Chapter 1. Improve Command-line Productivity

Abstract

Goal	Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various Red Hat Enterprise Linux utilities.
Objectives	• Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various Red Hat Enterprise Linux utilities. • Run repetitive tasks with `for` loops, evaluate exit codes from commands and scripts, run tests with operators, and create conditional structures with `if` statements. • Create regular expressions to match data, apply regular expressions to text files with the `grep` command, and use `grep` to search files and data from piped commands.
Sections	• Write Simple Bash Scripts (and Guided Exercise) • Loops and Conditional Constructs in Scripts (and Guided Exercise) • Match Text in Command Output with Regular Expressions (and Guided Exercise)
Lab	Improve Command-line Productivity

Write Simple Bash Scripts

Objectives

Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various Red Hat Enterprise Linux utilities.

Create and Execute Bash Shell Scripts

You can accomplish many system administration tasks by using command-line tools. More complex tasks often require chaining together multiple commands that pass results between them. By using the Bash shell environment and scripting features, you can combine Linux commands into *shell scripts* to solve repetitive real-world problems.

A Bash shell script is an executable file that contains a list of commands, and possibly with programming logic to control decision-making in the overall task. When well-written, a shell script is a powerful command-line tool on its own, and you can use it with other scripts.

Shell scripting proficiency is essential for system administrators in any operational environment. You can use shell scripts to improve the efficiency and accuracy of routine task completion.

Although you can use any text editor, advanced editors such as `vim` or `emacs` understand Bash shell syntax and can provide color-coded highlighting. This highlighting helps to identify common scripting errors such as improper syntax, unmatched quotes, parentheses, brackets, and braces, and other structural mistakes.

Specify the Command Interpreter

The first line of a script begins with the `#!` notation, which is commonly referred to as `she-bang` or `hash-bang`, from the names of those two characters, `sharp` or `hash` and `bang`. This notation is an interpreter directive to indicate the command interpreter and command options to process the remaining lines in the file. For Bash syntax script files, the first line is the following directive:

```
#!/usr/bin/bash
```

Execute a Bash Shell Script

A shell script file must have execute permissions to run it as an ordinary command. Use the `chmod` command to modify the file permissions. Use the `chown` command, if needed, to grant execute permission only for specific users or groups.

If the script is stored in a directory that is listed in the shell's `PATH` environmental variable, then you can run the shell script by using only its file name, similar to running compiled commands. Because PATH parsing runs the first matching file name that is found, always avoid using existing command names to name your script files. If a script is not in a PATH directory, then run the script by using its absolute path name, which you can determine by querying the file with the `which` command. Alternatively, run a script in your current working directory by using the `.` directory prefix, such as `./scriptname`.

```
[user@host ~]$ which hello
~/bin/hello
[user@host ~]$ echo $PATH
/home/user/.local/bin:/home/user/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/u
sr/local/bin
```

Quote Special Characters

Some characters and words have a special meaning to the Bash shell. To use these characters for their literal values, rather than for their special meanings, you *escape* them in the script. Use the backslash character (`\`), single quotes (`''`), or double quotes (`""`) to remove (or *escape*) the special meaning of these characters.

The backslash character removes the special meaning of the single character that immediately follows the backslash. For example, to use the `echo` command to display the `# not a comment` literal string, the `#` hash character must not be interpreted as a comment.

The following example shows the backslash character (`\`) modifying the hash character so it is not interpreted as a comment:

```
[user@host ~]$ echo # not a comment


[user@host ~]$ echo \# not a comment
# not a comment
```

To escape more than one character in a text string, either use the backslash character multiple times, or enclose the whole string in single quotes (`''`) to interpret literally. Single quotes preserve the literal meaning of all characters that they enclose. Observe the backslash character and single quotes in these examples:

```
[user@host ~]$ echo # not a comment #

[user@host ~]$ echo \# not a comment #
# not a comment
[user@host ~]$ echo \# not a comment \#
# not a comment #
[user@host ~]$ echo '# not a comment #'
# not a comment #
```

Use double quotation marks to suppress *globbing* (file name pattern matching) and shell expansion, but still allow command and variable substitution. Variable substitution is conceptually the same as command substitution, but might use optional brace syntax. Observe the following examples of various quotation mark forms.

Use single quotation marks to interpret all enclosed text literally. Besides suppressing globbing and shell expansion, single quotation marks also direct the shell to suppress command and variable substitution. The question mark (`?`) is included inside the quotations, because it is a *metacharacter* that also needs escaping from expansion.

```
[user@host ~]$ var=$(hostname -s); echo $var
host
[user@host ~]$ echo "***** hostname is ${var} *****"
***** hostname is host *****
[user@host ~]$ echo Your username variable is \$USER.
Your username variable is $USER.
[user@host ~]$ echo "Will variable $var evaluate to $(hostname -s)?"
Will variable host evaluate to host?
[user@host ~]$ echo 'Will variable $var evaluate to $(hostname -s)?'
Will variable $var evaluate to $(hostname -s)?
[user@host ~]$ echo "\"Hello, world\""
"Hello, world"
[user@host ~]$ echo '"Hello, world"'
"Hello, world"
```

Provide Output from a Shell Script

The `echo` command displays arbitrary text by passing the text as an argument to the command. By default, the text is sent to *standard output (STDOUT)*. You can send text elsewhere by using output redirection. In the following simple Bash script, the `echo` command displays the `"Hello, world"` message to STDOUT, which defaults to the screen device.

```
[user@host ~]$ cat ~/bin/hello
#!/usr/bin/bash


echo "Hello, world"


[user@host ~]$ hello
Hello, world
```

Note

This user can run `hello` at the prompt because the `~/bin` (`/home/user/bin`) directory is in the user's `PATH` variable and the `hello` script has executable permission. The PATH parser finds the script first, if no other executable file called `hello` is found in any earlier `PATH` directory. Your home directory's `bin` subdirectory is intended to store your personal scripts.

The `echo` command is widely used in shell scripts to display informational or error messages. Messages helpfully indicate a script's progress, and can be directed to standard output or standard error, or be redirected to a log file for archiving. When you display error messages, good programming practice is to redirect error messages to STDERR to separate them from normal program output.

```
[user@host ~]$ cat ~/bin/hello
#!/usr/bin/bash


echo "Hello, world"
echo "ERROR: Houston, we have a problem." >&2


[user@host ~]$ hello 2> hello.log
```

```
Hello, world
[user@host ~]$ cat hello.log
ERROR: Houston, we have a problem.
```

The `echo` command is also helpful to debug a problematic shell script. Adding echo statements in a script, to display variable values and other runtime information, can help to clarify how a script is behaving.

Guided Exercise: Write Simple Bash Scripts

In this exercise, you write a simple Bash script with a sequence of commands and run it from the command line.

Outcomes

- Write and execute a simple Bash script.
- Redirect the output of a simple Bash script to a file.

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

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

```
[student@workstation ~]$ lab start console-write
```

Instructions

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

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

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

4. Create and execute a simple Bash script.
    1. Use the `vim` command to create the `firstscript.sh` file under your home directory.

```
[student@servera ~]$ vim firstscript.sh
```

2. Insert the following text, and save the file. The number of hash signs (#) is arbitrary.

```
3. #!/usr/bin/bash
4. echo "This is my first bash script" > ~/output.txt
5. echo "" >> ~/output.txt
```

```
echo "#####################################################" >> ~/output
.txt
```

6. Use the `bash` command to execute the script.

```
[student@servera ~]$ bash firstscript.sh
```

7. Review the output file that the script generated.

```
8. [student@servera ~]$ cat output.txt
9. This is my first bash script
10.
```

```
#####################################################
```

5. Add more commands to the `firstscript.sh` script, execute it, and review the output.
   1. Use the Vim text editor to edit the `firstscript.sh` script.

```
[student@servera ~]$ vim firstscript.sh
```

The following output shows the expected content of the `firstscript.sh` file:

```
#!/usr/bin/bash
#
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/output
.txt
echo "LIST BLOCK DEVICES" >> ~/output.txt
echo "" >> ~/output.txt
```

```
lsblk >> ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/output.txt
echo "FILESYSTEM FREE SPACE STATUS" >> ~/output.txt
echo "" >> ~/output.txt
df -h >> ~/output.txt
echo "#####################################################" >> ~/output.txt
```

2. Make the `firstscript.sh` file executable by using the `chmod` command.

```
[student@servera ~]$ chmod a+x firstscript.sh
```

3. Execute the `firstscript.sh` script.

```
[student@servera ~]$ ./firstscript.sh
```

4. Review the output file that the script generated.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####################################################
LIST BLOCK DEVICES

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0     11:0    1  558K  0 rom
vda    252:0    0   10G  0 disk
├─vda1 252:1    0    1M  0 part
├─vda2 252:2    0  200M  0 part /boot/efi
├─vda3 252:3    0  500M  0 part /boot
└─vda4 252:4    0  9.3G  0 part /
vdb    252:16   0    5G  0 disk
vdc    252:32   0    5G  0 disk
vdd    252:48   0    5G  0 disk

```

```
#####################################################
FILESYSTEM FREE SPACE STATUS

Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        844M     0  844M   0% /dev
tmpfs           888M     0  888M   0% /dev/shm
tmpfs           355M  9.4M  346M   3% /run
/dev/vda4       9.4G  1.7G  7.7G  18% /
/dev/vda3       495M  161M  335M  33% /boot
/dev/vda2       200M  7.6M  193M   4% /boot/efi
tmpfs           178M     0  178M   0% /run/user/1000
```

```
#####################################################
```

6. Remove the exercise files and return to the `workstation` machine.
    1. Delete the `firstscript.sh` and `output.txt` files.

    ```
    [student@servera ~]$ rm firstscript.sh output.txt
    ```

    2. Return to the `workstation` machine as the `student` user.

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

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish console-write
```

This concludes the section.

Guided Exercise: Write Simple Bash Scripts

In this exercise, you write a simple Bash script with a sequence of commands and run it from the command line.

Outcomes

- Write and execute a simple Bash script.
- Redirect the output of a simple Bash script to a file.

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

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

```
[student@workstation ~]$ lab start console-write
```

Instructions

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

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

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

4. Create and execute a simple Bash script.
    1. Use the `vim` command to create the `firstscript.sh` file under your home directory.

    ```
    [student@servera ~]$ vim firstscript.sh
    ```

    2. Insert the following text, and save the file. The number of hash signs (#) is arbitrary.

    ```
    #!/usr/bin/bash
    echo "This is my first bash script" > ~/output.txt
    echo "" >> ~/output.txt
    ```

```
echo "#####################################################" >> ~/output
.txt
```

6. Use the `bash` command to execute the script.

```
[student@servera ~]$ bash firstscript.sh
```

7. Review the output file that the script generated.

```
8. [student@servera ~]$ cat output.txt
9. This is my first bash script
10.
```

```
#####################################################
```

5. Add more commands to the `firstscript.sh` script, execute it, and review the output.
   1. Use the Vim text editor to edit the `firstscript.sh` script.

```
[student@servera ~]$ vim firstscript.sh
```

The following output shows the expected content of the `firstscript.sh` file:

```
#!/usr/bin/bash
#
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/output
.txt
echo "LIST BLOCK DEVICES" >> ~/output.txt
echo "" >> ~/output.txt
lsblk >> ~/output.txt
echo "" >> ~/output.txt
echo "#####################################################" >> ~/output
.txt
echo "FILESYSTEM FREE SPACE STATUS" >> ~/output.txt
echo "" >> ~/output.txt
```

```
df -h >> ~/output.txt

echo "#####################################################" >> ~/output
.txt
```

2. Make the `firstscript.sh` file executable by using the `chmod` command.

```
[student@servera ~]$ chmod a+x firstscript.sh
```

3. Execute the `firstscript.sh` script.

```
[student@servera ~]$ ./firstscript.sh
```

4. Review the output file that the script generated.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####################################################
LIST BLOCK DEVICES

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0     11:0    1  558K  0 rom
vda    252:0    0   10G  0 disk
├─vda1 252:1    0    1M  0 part
├─vda2 252:2    0  200M  0 part /boot/efi
├─vda3 252:3    0  500M  0 part /boot
└─vda4 252:4    0  9.3G  0 part /
vdb    252:16   0    5G  0 disk
vdc    252:32   0    5G  0 disk
vdd    252:48   0    5G  0 disk

#####################################################
FILESYSTEM FREE SPACE STATUS

Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        844M     0  844M   0% /dev
tmpfs           888M     0  888M   0% /dev/shm
tmpfs           355M  9.4M  346M   3% /run
/dev/vda4       9.4G  1.7G  7.7G  18% /
/dev/vda3       495M  161M  335M  33% /boot
/dev/vda2       200M  7.6M  193M   4% /boot/efi
tmpfs           178M     0  178M   0% /run/user/1000
```

```
#####################################################
```

6. Remove the exercise files and return to the `workstation` machine.
    1. Delete the `firstscript.sh` and `output.txt` files.

    ```
    [student@servera ~]$ rm firstscript.sh output.txt
    ```

    2. Return to the `workstation` machine as the `student` user.

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

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish console-write
```

This concludes the section.

Guided Exercise: Loops and Conditional Constructs in Scripts

In this exercise, you use loops to efficiently print the hostname from multiple servers.

Outcomes

- Create a `for` loop to iterate through a list of items from the command line and in a shell script.

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

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

```
[student@workstation ~]$ lab start console-commands
```

Instructions

1. Use the `ssh` and `hostname` commands to print the hostname of the `servera` and `serverb` machines to standard output.

```
2. [student@workstation ~]$ ssh student@servera hostname
3. servera.lab.example.com
4. [student@workstation ~]$ ssh student@serverb hostname
```

```
serverb.lab.example.com
```

5. Create a `for` loop to execute the `hostname` command on the `servera` and `serverb` machines.

```
6. [student@workstation ~]$ for HOST in servera serverb
7. do
8. ssh student@${HOST} hostname
9. done
10. servera.lab.example.com
```

```
serverb.lab.example.com
```

11. Create a shell script in the `/home/student/bin` directory to execute the same `for` loop. Ensure that the script is included in the `PATH` environment variable.
    1. Create the `/home/student/bin` directory to store the shell script, if the directory does not exist.

    ```
    [student@workstation ~]$ mkdir ~/bin
    ```

    2. Verify that the `bin` subdirectory of your home directory is in your `PATH` environment variable.

    ```
    3. [student@workstation ~]$ echo $PATH
    ```

    ```
    /home/student/.local/bin:/home/student/bin:/sbin:/bin:/usr/sbin:/usr/bin
    :/usr/local/sbin:/usr/local/bin:/home/student/.venv/labs/bin
    ```

    4. Create a shell script called `printhostname.sh` in the `/home/student/bin` directory to perform the `for` loop, and add the following content in the file.

    ```
    5. [student@workstation ~]$ vim ~/bin/printhostname.sh
    6. #!/usr/bin/bash
    7. #Execute for loop to print server hostname.
    8. for HOST in servera serverb
    9. do
    10.   ssh student@${HOST} hostname
    11. done
    ```

    ```
    exit 0
    ```

    12. Give the created script executable permission.

    ```
    [student@workstation ~]$ chmod +x ~/bin/printhostname.sh
    ```

    13. Run the script from your home directory.

    ```
    14. [student@workstation ~]$ printhostname.sh
    15. servera.lab.example.com
    ```

```
serverb.lab.example.com
```

16. Verify that the exit code of your script is 0.

```
17. [student@workstation ~]$ echo $?
```

```
0
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish console-commands
```

This concludes the section.

Match Text in Command Output with Regular Expressions

Objectives

Create regular expressions to match data, apply regular expressions to text files with the `grep` command, and use `grep` to search files and data from piped commands.

Write Regular Expressions

Regular expressions provide a pattern matching mechanism to find specific content. The `vim`, `grep`, and `less` commands can use regular expressions. Programming languages such as Perl, Python, and C also support regular expressions, but might differ slightly in syntax.

Regular expressions are a unique language, with their own syntax and rules. This section introduces regular expression syntax as implemented in bash, with examples.

Describe a Simple Regular Expression

The simplest regular expression is an exact match of the string to search. An exact match is when the characters in the regular expression match the type and order of the string.

Imagine that a user is looking through the following file for all occurrences of the pattern `cat`:

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

The `cat` string is an exact match of the `c` character, followed by the `a` and `t` characters with no other characters between. Searching the file with the `cat` string as the regular expression returns the following matches:

```
cat
concatenate
category
educated
vindication
```

Match the Start and End of a Line

The regular expression would match the search string anywhere on the line on which it occurred: the beginning, middle, or end of the word or line. Use a *line anchor* metacharacter to control where on a line to look for a match.

To match only at the beginning of a line, use the caret character (`^`). To match only at the end of a line, use the dollar sign (`$`).

With the same file as for the previous example, the `^cat` regular expression would match two lines.

```
cat
category
```

The `cat$` regular expression would find only one match, where the `cat` characters are at the end of a line.

```
cat
```

Locate lines in the file that end with `dog`, by using an end-of-line anchor to create the `dog$` regular expression, which matches two lines:

```
dog
chilidog
```

To locate a line that contains only the search expression exactly, use both the beginning and end-of-line anchors. For example, to locate the word `cat` when it is both at the beginning and the end of a line simultaneously, use `^cat$`.

```
cat
```

Basic and Extended Regular Expression

The two types of regular expressions are basic regular expressions and extended regular expressions.

One difference between basic and extended regular expressions is in the behavior of the `|`, `+`, `?`, `(`, `)`, `{`, and `}` special characters. In basic regular expression syntax, these characters have a special meaning only if they are prefixed with a backslash `\` character. In extended regular expression syntax, these characters are special unless they are prefixed with a backslash `\` character. Other minor differences apply to how the `^`, `$`, and `*` characters are handled.

The `grep`, `sed`, and `vim` commands use basic regular expressions.
The `grep` command `-E` option, the `sed` command `-E` option, and the `less` command use extended regular expressions.

Wildcard and Multiplier Usage in Regular Expressions

Regular expressions use a dot character (`.`) as a wildcard to match any single character on a single line. The `c.t` regular expression searches for a string that contains a `c`, followed by any single character, followed by a `t`. Example matches might include `cat`, `concatenate`, `vindication`, `cut`, and `c$t`.

With an unrestricted wildcard, you cannot predict the character that matches the wildcard. To match specific characters, replace the unrestricted wildcard with appropriate characters.

The use of bracket characters, such as in the `c[aou]t` regular expression, matches patterns that start with a `c`, followed by an `a`, `o`, or `u`, followed by a `t`. Possible matching expressions can have the `cat`, `cot`, and `cut` strings.

Multipliers are an often used mechanism with wildcards. Multipliers apply to the previous character or wildcard in the regular expression. An often used multiplier is the asterisk (`*`) character. When used in a regular expression, the asterisk multiplier matches zero or more occurrences of the multiplied expression. You can use the asterisk with expressions, in addition to characters.

For example, the `c[aou]*t` regular expression might match `coat` or `coot`. A regular expression of `c.*t` matches `cat`, `coat`, `culvert`, and even `ct` (matching zero characters between the `c` and the `t`). Any string that starts with a `c`, is followed by zero or more characters, and ends with a `t` must be a match.

Another type of multiplier indicates a more precise number of characters in the pattern. An example of an explicit multiplier is the `'c.\{2\}t'` regular expression, which matches any word that begins with a `c`, followed by exactly any two characters, and ends with a `t`. The `'c.\{2\}t'` expression would match two words in the following example:

```
cat
coat
convert
cart
covert
cypher
```

Note

This course introduced two metacharacter text parsing mechanisms: shell *pattern matching* (also known as file globbing or file name expansion), and *regular expressions*. Both mechanisms use similar metacharacters, such as the asterisk character (`*`), but have differences in metacharacter interpretation and rules.

Pattern matching is a shell technique to specify multiple file names on the command line. Regular expressions represent any form or pattern in text strings, no matter how complex. Regular expressions are internally supported by many text processing commands, such as `grep`, `sed`, `awk`, `python`, and `perl`, and in many applications.

Table 1.1. Basic and Extended Regular Expression Syntax

Basic syntax	Extended syntax	Description
.		The period (`.`) matches any single character.
?		The preceding item is optional and is matched at most once.
*		The preceding item is matched zero or more times.
+		The preceding item is matched one or more times.
\{n\}	{n}	The preceding item is matched exactly `n` times.
\{n,\}	{n,}	The preceding item is matched `n` or more times.
\{,m\}	{,m}	The preceding item is matched at most `m` times.
\{n,m\}	{n,m}	The preceding item is matched at least `n` times, but not more than `m` times.
[:alnum:]		Alphanumeric characters: `[:alpha:]` and `[:digit:]`; in the 'C' locale and ASCII character encoding, this expression is the same as `[0-9A-Za-z]`.
[:alpha:]		Alphabetic characters: `[:lower:]` and `[:upper:]`; in the 'C' locale and ASCII character encoding, this expression is the same as `[A-Za-z]`.
[:blank:]		Blank characters: space and tab.
[:cntrl:]		Control characters. In ASCII, these characters have octal codes 000 through 037, and 177 (DEL).
[:digit:]		Digits: `0 1 2 3 4 5 6 7 8 9`.
[:graph:]		Graphical characters: `[:alnum:]` and `[:punct:]`.
[:lower:]		Lowercase letters; in the 'C' locale and ASCII character encoding: `a b c d e f g h i j k l m n o p q r s t u v w x y z`.
[:print:]		Printable characters: `[:alnum:]`, `[:punct:]`, and space.
[:punct:]		Punctuation characters; in the 'C' locale and ASCII character encoding: `! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ' { | } ~`.
[:space:]		Space characters: in the 'C' locale, it is tab, newline, vertical tab, form feed, carriage return, and space.

Basic syntax	Extended syntax	Description
[:upper:]		Uppercase letters: in the 'C' locale and ASCII character encoding, it is: `A B C D E F G H I J K L M N O P Q R S T U V W X Y Z`.
[:xdigit:]		Hexadecimal digits: `0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f`.
\b		Match the empty string at the edge of a word.
\B		Match the empty string provided that it is not at the edge of a word.
\<		Match the empty string at the beginning of a word.
\>		Match the empty string at the end of a word.
\w		Match word constituent. Synonym for `[_[:alnum:]]`.
\W		Match non-word constituent. Synonym for `[^_[:alnum:]]`.
\s		Match white space. Synonym for `[[:space:]]`.
\S		Match non-white space. Synonym for `[^[:space:]]`.

Match Regular Expressions from the Command Line

The `grep` command uses regular expressions to isolate matching data. You can use the `grep` command to match data in a single file or in multiple files. When you use `grep` to match data in multiple files, it prints the file name followed by a colon character and then the lines that match the regular expression.

Isolating Data with the grep Command

The `grep` command specifies a regular expression and a file to parse for matches.

```
[user@host ~]$ grep '^computer' /usr/share/dict/words
computer
computerese
computerise
computerite
computerizable
computerization
computerize
computerized
computerizes
computerizing
computerlike
computernik
```

```
computers
```

Note

It is recommended practice to use single quotation marks to encapsulate the regular expression to protect any shell metacharacters (such as the `$`, `*`, and `{}` characters). Encapsulating the regular expression ensures that the command and not the shell interprets the characters.

The `grep` command can process output from other commands by using a pipe operator character (`|`). The following example shows the `grep` command parsing lines from the output of another command.

```
[root@host ~]# ps aux | grep chrony
chrony     662  0.0  0.1  29440  2468 ?        S    10:56   0:00 /usr/sbin/chronyd
```

The grep Command Options

The `grep` command has many options for controlling how it parses lines.

Table 1.2. Table of Common grep Options

Option	Function
`-i`	Use the provided regular expression and do not enforce case sensitivity (run case-insensitive).
`-v`	Display only lines that do *not* contain matches to the regular expression.
`-r`	Search for data that matches the regular expression recursively in 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`	If multiple `-e` options are used, then multiple regular expressions can be supplied and are used with a logical OR.
`-E`	Use extended regular expression syntax instead of basic regular expression syntax when parsing the provided regular expression.

View the `man` pages to find other options for the `grep` command.

Examples of the grep Command

The following examples use various configuration files and log files.

Regular expressions are case-sensitive by default. Use the `grep` command `-i` option to run a case-insensitive search. The following example shows an excerpt of the `/etc/httpd/conf/httpd.conf` configuration file.

```
[user@host ~]$ cat /etc/httpd/conf/httpd.conf
...output omitted...
ServerRoot "/etc/httpd"


#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on a specific IP address, but note that if
# httpd.service is enabled to run at boot time, the address may not be
# available when the service starts.  See the httpd.service(8) man
# page for more information.
#
#Listen 12.34.56.78:80
Listen 80
...output omitted...
```

The following example searches for the `serverroot` regular expression in the `/etc/httpd/conf/httpd.conf` configuration file.

```
[user@host ~]$ grep -i serverroot /etc/httpd/conf/httpd.conf
# with "/", the value of ServerRoot is prepended -- so 'log/access_log'
# with ServerRoot set to '/www' will be interpreted by the
# ServerRoot: The top of the directory tree under which the server's
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# same ServerRoot for multiple httpd daemons, you will need to change at
ServerRoot "/etc/httpd"
```

Use the `grep` command `-v` option to *reverse* search the regular expression. This option displays only the lines that do *not* match the regular expression.

In the following example, all lines, regardless of case, that *do not* contain the `server` regular expression are returned.

```
[user@host ~]$ grep -v -i server /etc/hosts
127.0.0.1 localhost.localdomain localhost
172.25.254.254 classroom.example.com classroom
172.25.254.254 content.example.com content
172.25.254.254 materials.example.com materials
### rht-vm-hosts file listing the entries to be appended to /etc/hosts

172.25.250.9     workstation.lab.example.com workstation
172.25.250.254   bastion.lab.example.com bastion
172.25.250.220  utility.lab.example.com utility
172.25.250.220  registry.lab.example.com registry
```

To view a file without the distraction of comment lines, use the `grep` command `-v` option. In the following example, the regular expression matches and excludes all the lines that begin with a hash character (`#`) or a semicolon (`;`) character in the `/etc/systemd/system/multi-user.target.wants/rsyslog.service` file. In that file, the hash character at the beginning of a line indicates a general comment, whereas the semicolon character refers to a commented variable value.

```
[user@host ~]$ grep -v '^[#;]' \
/etc/systemd/system/multi-user.target.wants/rsyslog.service
[Unit]
Description=System Logging Service
Documentation=man:rsyslogd(8)
Documentation=https://www.rsyslog.com/doc/

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/rsyslog
ExecStart=/usr/sbin/rsyslogd -n $SYSLOGD_OPTIONS
```

```
ExecReload=/usr/bin/kill -HUP $MAINPID

UMask=0066

StandardOutput=null

Restart=on-failure


LimitNOFILE=16384


[Install]

WantedBy=multi-user.target
```

The `grep` command `-e` option can search for more than one regular expression at a time. The following example, which uses a combination of the `less` and `grep` commands, locates all occurrences of `pam_unix`, `user root`, and `Accepted publickey` in the `/var/log/secure` log file.

```
[root@host ~]# cat /var/log/secure | grep -e 'pam_unix' \

-e 'user root' -e 'Accepted publickey' | less

Mar  4 03:31:41 localhost passwd[6639]: pam_unix(passwd:chauthtok): password changed
for root

Mar  4 03:32:34 localhost sshd[15556]: Accepted publickey for devops from 10.30.0.167
port 56472 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFCt+wowZLNzNlBT0

Mar  4 03:32:34 localhost systemd[15560]: pam_unix(systemd-user:session): session ope
ned for user devops(uid=1001) by (uid=0)
```

To search for text in a file that you opened with the `vim` or `less` commands, first enter the slash character (`/`) and then type the pattern to find. Press **Enter** to start the search. Press **N** to find the next match.

```
[root@host ~]# vim /var/log/boot.log

...output omitted...

[^[[0;32m  OK  ^[[0m] Finished ^[[0;1;39mdracut pre-pivot and cleanup hook^[[0m.^M

         Starting ^[[0;1;39mCleaning Up and Shutting Down Daemons^[[0m...^M

[^[[0;32m  OK  ^[[0m] Stopped target ^[[0;1;39mRemote Encrypted Volumes^[[0m.^M

[^[[0;32m  OK  ^[[0m] Stopped target ^[[0;1;39mTimer Units^[[0m.^M

[^[[0;32m  OK  ^[[0m] Closed ^[[0;1;39mD-Bus System Message Bus Socket^[[0m.^M

/Daemons

[root@host ~]# less /var/log/messages
```

```
...output omitted...
Mar  4 03:31:19 localhost kernel: pci 0000:00:02.0: vgaarb: setting as boot VGA devic
e
Mar  4 03:31:19 localhost kernel: pci 0000:00:02.0: vgaarb: VGA device added: decodes
=io+mem,owns=io+mem,locks=none
Mar  4 03:31:19 localhost kernel: pci 0000:00:02.0: vgaarb: bridge control possible
Mar  4 03:31:19 localhost kernel: vgaarb: loaded
Mar  4 03:31:19 localhost kernel: SCSI subsystem initialized
Mar  4 03:31:19 localhost kernel: ACPI: bus type USB registered
Mar  4 03:31:19 localhost kernel: usbcore: registered new interface driver usbfs
Mar  4 03:31:19 localhost kernel: usbcore: registered new interface driver hub
Mar  4 03:31:19 localhost kernel: usbcore: registered new device driver usb
/device
```

Guided Exercise: Match Text in Command Output with Regular Expressions

In this lab, you search for text in the system logs and the output of commands to find information more efficiently.

Outcomes

- Efficiently search for text in log files and configuration files.

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

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start console-regex
```

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

```
2. [student@workstation ~]$ ssh student@servera
```

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

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

6. Use the `grep` command to find the GID and UID for the `postfix` and `postdrop` groups and users. To do so, use the `rpm -q --scripts` command, which queries the information for a specific package and shows the scripts that are used as part of the installation process.

```
[root@servera ~]# rpm -q --scripts postfix | grep -e 'user' -e 'group'
# Add user and groups if necessary
/usr/sbin/groupadd -g 90 -r postdrop 2>/dev/null
/usr/sbin/groupadd -g 89 -r postfix 2>/dev/null
/usr/sbin/groupadd -g 12 -r mail 2>/dev/null
/usr/sbin/useradd -d /var/spool/postfix -s /sbin/nologin -g postfix -G mail -M -r -u 89 postfix 2>/dev/null
```

```
  setgid_group=postdrop \
```

13. Modify the previous regular expression to display the first two messages in the `/var/log/maillog` file. In this search, you do not need to use the caret character (^), because you are not searching for the first character in a line.

```
[root@servera ~]# grep 'postfix' /var/log/maillog | head -n 2
Apr  1 15:27:16 servera postfix/postfix-script[3121]: starting the Postfix mail system
```

```
Apr  1 15:27:16 servera postfix/master[3123]: daemon started -- version 3.5.9, configuration /etc/postfix
```

16. Find the name of the `queue` directory for the `Postfix` server. Search the `/etc/postfix/main.cf` configuration file for all information about queues. Use the `grep` command `-i` option to ignore case distinctions.

```
[root@servera ~]# grep -i 'queue' /etc/postfix/main.cf
# testing.  When soft_bounce is enabled, mail will remain queued that
# The queue_directory specifies the location of the Postfix queue.
queue_directory = /var/spool/postfix
```

```
# QUEUE AND PROCESS OWNERSHIP
# The mail_owner parameter specifies the owner of the Postfix queue
# is the Sendmail-compatible mail queue listing command.
```

```
# setgid_group: The group for mail submission and queue management
```

24. Confirm that the `postfix` service writes messages to the `/var/log/messages` file. Use the `less` command and then the slash character (/) to search the file. Press **n** to move to the next entry that matches the search. Press **q** to quit the `less` command.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Apr  1 15:27:15 servera systemd[1]: Starting Postfix Mail Transport Agent...
...output omitted...
Apr  1 15:27:16 servera systemd[1]: Started Postfix Mail Transport Agent.
...output omitted...
```

```
/Postfix
```

31. Use the `ps aux` command to confirm that the `postfix` server is currently running. Use the `grep` command to limit the output to the necessary lines.

```
[root@servera ~]# ps aux | grep postfix
root        3123  0.0  0.2  38172  4384 ?        Ss   15:27   0:00 /usr/libexe
c/postfix/master -w
postfix     3124  0.0  0.4  45208  8236 ?        S    15:27   0:00 pickup -l -
t unix -u
postfix     3125  0.0  0.4  45252  8400 ?        S    15:27   0:00 qmgr -l -t
unix -u
```

```
root        3228  0.0  0.1 221668  2288 pts/0    S+   15:55   0:00 grep --colo
r=auto postfix
```

36. Confirm that the `qmgr`, `cleanup`, and `pickup` queues are correctly configured. Use the `grep` command `-e` option to match multiple entries in the same file. The `/etc/postfix/master.cf` file is the configuration file.

```
[root@servera ~]# grep -e qmgr -e pickup -e cleanup /etc/postfix/master.cf
pickup    unix  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
```

```
40. qmgr      unix  n       -       n       300     1       qmgr
```

```
#qmgr     unix  n       -       n       300     1       oqmgr
```

41. Return to the `workstation` machine as the `student` user.

```
42. [root@servera ~]# exit
43. logout
44. [student@servera ~]$ exit
45. logout
46. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish console-regex
```

This concludes the section.

Summary

- Create and execute Bash scripts to accomplish administration tasks.
- Use loops to iterate through a list of items from the command line and in a shell script.
- Use conditional structures to incorporate decision-making into shell scripts.
- Search for text in log and configuration files by using regular expressions and the `grep` command.

Chapter 2. Schedule Future Tasks

Abstract

Goal	Schedule tasks to execute at a specific time and date.
Objectives	• Set up a command to run once at a future time. • Schedule commands to run on a repeating schedule with a user's `crontab` file. • Schedule commands to run on a repeating schedule with the system `crontab` file and directories. • Enable and disable `systemd` timers, and configure a timer that manages temporary files.
Sections	• Schedule a Deferred User Job (and Guided Exercise) • Schedule Recurring User Jobs (and Guided Exercise) • Schedule Recurring System Jobs (and Guided Exercise) • Manage Temporary Files (and Guided Exercise) • Schedule Future Tasks (Quiz)

Schedule a Deferred User Job

Objectives

Set up a command to run once at a future time.

Describe Deferred User Tasks

Sometimes you might need to run one or more commands at a specific future time. An example is a user who schedules a long-running maintenance task to occur in the middle of the night. Another example is a system administrator who is working on a firewall configuration and queues a safety job to reset the firewall settings to a former working state in ten minutes' time. The system administrator then deactivates the job before it runs, unless the new firewall configuration worked.

These scheduled commands are called *tasks* or *jobs*, and the *deferred* term indicates that these tasks run in the future.

One available solution for Red Hat Enterprise Linux users to schedule deferred tasks is the `at` command, which is installed and enabled by default. The `at` package provides the `atd` system daemon and the `at` and `atq` commands to interact with the daemon.

Any user can queue jobs for the `atd` daemon by using the `at` command.
The `atd` daemon provides 26 queues, identified from `a` to `z`, where jobs in alphabetically later queues get lower system priority (with higher *nice* values, as discussed in a later chapter).

Schedule Deferred User Tasks

Use the `at` *`TIMESPEC`* command to start entering a new job to schedule.
The `at` command then reads from `STDIN` (your keyboard) to obtain the commands to run. When manually entering commands, complete the input by pressing **Ctrl**+**D** on an empty line. You can use input redirection from a script file for entering more complex commands. For example, use the `at now +5min <` *`myscript`* command to schedule the commands in `myscript` to start in 5 minutes, without needing to type the commands manually in a terminal window.

The `at` command `TIMESPEC` argument accepts natural time specifications to describe when a job should run. For example, specify a time as `02:00pm`, `15:59`, `midnight`, or even `teatime`, followed by an optional date or number of days in the future.

The `TIMESPEC` argument expects time and date specifications in that order. If you provide the date and not the time, then the time defaults to the current time. If you provide the time and not the date, then the date is considered to be matched, and the jobs run when the time next matches.

The following example shows a job schedule without providing the date.
The `at` command schedules the job for today or tomorrow depending whether the time has passed.

```
[user@host ~]$ date
Wed May 18 21:01:18 CDT 2022
[user@host ~]$ at 21:03 < myscript
job 3 at Wed May 18 21:03:00 2022
[user@host ~]$ at 21:00 < myscript
job 4 at Thu May 19 21:00:00 2022
```

The man pages for the `at` command and other documentation sources use lowercase to write the natural time specifications. You can use lowercase, sentence case, or uppercase. Here are examples of time specifications that you can use:

- `now +5min`
- `teatime tomorrow` (teatime is `16:00`)
- `noon +4 days`
- `5pm august 3 2021`

For other valid time specifications, refer to the local `timespec` document listed in the references.

Inspect and Manage Deferred User Jobs

For an overview of the pending jobs for the current user, use the `atq` or the `at -l` command.

```
[user@host ~]$ atq
28  Mon May 16 05:13:00 2022 a user
29  Tue May 17 16:00:00 2022 h user
30  Wed May 18 12:00:00 2022 a user
```

In the preceding output, every line represents a different scheduled future job. The following description applies to the first line of the output:

- **`28`** is the unique job number.
- **`Mon May 16 05:13:00 2022`** is the execution date and time for the scheduled job.

- **`a`** indicates that the job is scheduled with the default queue `a`.
- **`user`** is the owner of the job (and the user that the job runs as).

Important

Unprivileged users can view and manage only their own jobs. The `root` user can view and manage all jobs.

Use the `at -c` *`JOBNUMBER`* command to inspect the commands that run when the `atd` daemon executes a job. This command shows the job's environment, which is set from the user's environment when they created the job, and the command syntax to run.

Remove Jobs from Schedule

The `atrm` *`JOBNUMBER`* command removes a scheduled job. Remove the scheduled job when you no longer need it, for example, when a remote firewall configuration succeeded, and you do not need to reset it.

Guided Exercise: Schedule a Deferred User Job

In this exercise, you use the `at` command to schedule several commands to run at specified future times.

Outcomes

- Schedule a job to run at a specified future time.
- Inspect the commands that a scheduled job runs.
- Delete the scheduled jobs.

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

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

```
[student@workstation ~]$ lab start scheduling-at
```

Instructions

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

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

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

4. Schedule a job to run in two minutes from now. Save the output of the `date` command to the `/home/student/myjob.txt` file.
    1. Pass the `date >> /home/student/myjob.txt` string as the input to the `at` command, so that the job runs in two minutes from now.

    ```
    2. [student@servera ~]$ echo "date >> /home/student/myjob.txt" | at now +2m
       in
    3. warning: commands will be executed using /bin/sh
    ```

    ```
    job 1 at Thu Feb 16 18:51:16 2023
    ```

    4. List the scheduled jobs.

    ```
    5. [student@servera ~]$ atq
    ```

    ```
    1       Thu Feb 16 18:51:16 2023 a student
    ```

    6. Monitor the deferred jobs queue in real time. After the `atd` daemon executes, it removes the job from the queue.

       The command updates the output of the `atq` command every two seconds, by default. After the `atd` daemon removes the deferred job from the queue, press **Ctrl**+**c** to exit the `watch` command and return to the shell prompt.

    ```
    [student@servera ~]$ watch atq
    Every 2.0s: atq              servera.lab.example.com: Thu Feb 16 17:58:5
    0 2023


    1       Thu Feb 16 18:51:16 2023 a student
    ```

    7. Verify that the contents of the `/home/student/myjob.txt` file match the output of the `date` command.

The output matches the output of the `date` command, which confirms that the scheduled job executed successfully.

```
[student@servera ~]$ cat myjob.txt
Thu Feb 16 06:51:16 PM EDT 2023
```

5. Interactively schedule a job in the `g` queue that runs at `teatime` (16:00). The job should print the `It's teatime` message to the `/home/student/tea.txt` file. Append the new messages to the `/home/student/tea.txt` file.

```
6. [student@servera ~]$ at -q g teatime
7. warning: commands will be executed using /bin/sh
8. at> echo "It's teatime" >> /home/student/tea.txt
9. at> Ctrl+d
```

```
job 2 at Fri Feb 17 16:00:00 2023
```

10. Interactively schedule another job with the `b` queue that runs at `16:05`. The job should print `The cookies are good` message to the `/home/student/cookies.txt` file. Append the new messages to the `/home/student/cookies.txt` file.

```
11. [student@servera ~]$ at -q b 16:05
12. warning: commands will be executed using /bin/sh
13. at> echo "The cookies are good" >> /home/student/cookies.txt
14. at> Ctrl+d
```

```
job 3 at Fri Feb 17 16:05:00 2023
```

15. Inspect the commands in the pending jobs.
    1. View the job numbers of the pending jobs.

       Note the job numbers in the output, which might vary on your system. Use the job numbers from your system.

```
[student@servera ~]$ atq
2       Fri Feb 17 16:00:00 2023 g student
3       Fri Feb 17 16:05:00 2023 b student
```

2. View the commands in the pending job number *2*. Replace the job number if it changed for you.

   The job executes an `echo` command that appends the `It's teatime` message to the `/home/student/tea.txt` file.

```
[student@servera ~]$ at -c 2
...output omitted...
echo "It's teatime" >> /home/student/tea.txt
marcinDELIMITER1d7be6a7
```

3. View the commands in the pending job number *3*. Replace the job number if it changed for you.

   The job executes an `echo` command that appends the message `The cookies are good` to the `/home/student/cookies.txt` file.

```
[student@servera ~]$ at -c 3
...output omitted...
echo "The cookies are good" >> /home/student/cookies.txt
marcinDELIMITER44662c6f
```

16. View the job number of a job that runs at `teatime` (16:00), and remove it by using the `atrm` command.

```
17. [student@servera ~]$ atq
18. 2       Fri Feb 17 16:00:00 2023 g student
19. 3       Fri Feb 17 16:05:00 2023 b student
```

```
[student@servera ~]$ atrm 2
```

20. Verify that the scheduled job to run at `teatime` (16:00) no longer exists.
    1. View the list of pending jobs, and confirm that the scheduled job to run at `teatime` (16:00) no longer exists.

```
2. [student@servera ~]$ atq
```

```
3       Fri Feb 17 16:05:00 2023 b student
```

    3. Return to the `workstation` machine as the `student` user.

```
4. [student@servera ~]$ exit
5. logout
6. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish scheduling-at
```

This concludes the section.

Schedule Recurring User Jobs

Objectives

Schedule commands to run on a repeating schedule with a user's `crontab` file.

Describe Recurring User Jobs

Recurring jobs are scheduled to run repeatedly. Red Hat Enterprise Linux systems provide the `crond` daemon, which is enabled and started by default.
The `crond` daemon reads multiple configuration files: one per user, and a set of system-wide files. Each user has a personal file that they edit with the `crontab -e` command. When executing recurring jobs, these configuration files provide detailed control to users and administrators. If the scheduled job is not written to use redirection, then the `crond` daemon emails any generated output or errors to the job owner.

Schedule Recurring User Jobs

Use the `crontab` command to manage scheduled jobs. The following list shows the commands that a local user can use to manage their jobs:

Table 2.1. Examples Of The `crontab` Command

Command	**Intended use**
`crontab -l`	List the jobs for the current user.
`crontab -r`	Remove all jobs for the current user.
`crontab -e`	Edit jobs for the current user.
`crontab` *`filename`*	Remove all jobs, and replace them with jobs that are read from *filename*. This command uses `stdin` input when no file is specified.

A privileged user might use the `crontab` command `-u` option to manage jobs for another user. The `crontab` command is never used to manage system jobs, and using the `crontab` commands as the `root` user is not recommended due to the ability to exploit personal jobs that are configured to run as `root`. Configure such privileged jobs as described in the later section that describes recurring system jobs.

Describe User Job Format

The `crontab -e` command invokes the `vim` editor by default unless the `EDITOR` environment variable is set for another editor. Each job must use a unique line in the `crontab` file. Follow these recommendations for valid entries when writing recurring jobs:

- Empty lines for ease of reading
- Comments on lines that start with the number sign (`#`)
- Environment variables with a `NAME=value` format, which affects all lines after the line where they are declared

Standard variable settings include the `SHELL` variable, to declare the shell that is used for interpreting the remaining lines of the `crontab` file. The `MAILTO` variable determines who should receive the emailed output.

Note

The ability to send an email requires additional system configuration for a local mail server or an SMTP relay.

The fields in the `crontab` file appear in the following order:

- Minutes

- Hours
- Day of month
- Month
- Day of week
- Command

The command executes when the *Day of month* or *Day of week* fields use the same value other than the `*` character. For example, to run a command on the 11th day of every month, and every Friday at 12:15 (24-hour format), use the following job format:

```
15 12 11 * Fri command
```

The first five fields all use the same syntax rules:

- Use the `*` character to execute in every possible instance of the field.
- A number to specify the number of minutes or hours, a date, or a day of the week. For days of the week, `0` equals Sunday, `1` equals Monday, `2` equals Tuesday, and so on. `7` also equals Sunday.
- Use `x-y` for a range, which includes the `x` and `y` values.
- Use `x,y` for lists. Lists might include ranges as well, for example, `5,10-13,17` in the `Minutes` column, for a job to run at 5, 10, 11, 12, 13, and 17 minutes past the hour.
- The `*/x` indicates an interval of `x`; for example, `*/7` in the `Minutes` column runs a job every seven minutes.

Additionally, 3-letter English abbreviations are used for months or days of the week, for example, Jan, Feb, and Mon, Tue.

The last field contains the full command with options and arguments to execute with the default shell. If the command contains an unescaped percentage sign (`%`), then that percentage sign is treated as a newline character, and everything after the percentage sign passes to the command as `stdin` input.

Examples of Recurring User Jobs

The following job executes the `/usr/local/bin/yearly_backup` command at exactly 09:00 on 3 February, every year. February is represented as the number 2 in the example, because it is the second month of the year.

```
0 9 3 2 * /usr/local/bin/yearly_backup
```

The following job sends an email that contains the `Chime` word to the owner of this job every five minutes between and including 09:00 and 16:00, but only on each Friday in July.

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

The preceding `9-16` range of hours means that the job timer starts at the ninth hour (09:00) and continues until the end of the sixteenth hour (16:59). The job starts executing at `09:00` with the last execution at `16:55`, because five minutes after `16:55` is `17:00`, which is beyond the given scope of hours.

If a range is specific for the hours instead of a single value, then all hours within the range will match. Therefore, with the hours of `9-16`, this example matches every five minutes from 09:00 through 16:55.

Note

This example job sends the output as an email, because `crond` recognizes that the job allowed output to go to the `STDIO` channel without redirection. Because cron jobs run in a background environment without an output device (known as a *controlling terminal*), `crond` buffers the output and creates an email to send it to the specified user in the configuration. For system jobs, the email is sent to the `root` account.

The following job runs the `/usr/local/bin/daily_report` command every working day (Monday to Friday) two minutes before midnight.

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

The following job executes the `mutt` command to send the `Checking in` mail message to the `developer@example.com` recipient every working day (Monday to Friday), at 9 AM.

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

Guided Exercise: Schedule Recurring User Jobs

In this exercise, you schedule commands to run on a repeating schedule as a non-privileged user, with the `crontab` command.

Outcomes

- Schedule recurring jobs to run as a non-privileged user.
- Inspect the commands that a scheduled recurring job runs.
- Remove scheduled recurring jobs.

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

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

```
[student@workstation ~]$ lab start scheduling-cron
```

Instructions

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

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

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

4. Schedule a recurring job as the `student` user that appends the current date and time to the `/home/student/my_first_cron_job.txt` file every two minutes. Use the `date` command to display the current date and time. The job must run only from one day before to one day after the current time. The job must not run on any other day.
    1. Use the `date` command to display the current date and time. Note the day of the week, which you need for the next steps.

```
[student@workstation ~]$ date
Wed Mar 15 07:33:01 PM EDT 2023
```

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

Note

You can use the `date -d "last day" +%a` command to display the day before the current time, and the `date -d "next day" +%a` command to display the day after the current time.

```
[student@servera ~]$ date -d "last day" +%a
Tue
[student@servera ~]$ date -d "next day" +%a
Thu
```

4. Open the `crontab` file with the default text editor.

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

5. Insert the following line. Replace the range of days from one day before to one day after the current time:

```
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

6. Press **Esc** and type `:wq` to save the changes and exit the editor. When the editor exits, you should see the following output:

```
7. ...output omitted...
8. crontab: installing new crontab
```

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

5. Use the `crontab -l` command to list the scheduled recurring jobs. Inspect the command that you scheduled to run as a recurring job in the preceding step.

   Verify that the job runs the `/usr/bin/date` command and appends its output to the `/home/student/my_first_cron_job.txt` file.

```
[student@servera ~]$ crontab -l
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

6. Instruct your shell prompt to sleep until the `/home/student/my_first_cron_job.txt` file is created because of the

successful execution of the recurring job that you scheduled. Wait for your shell prompt to return.

The `while` command uses `! test -f` to continue to run a loop, and sleeps for one second until the `my_first_cron_job.txt` file is created in the `/home/student` directory.

```
[student@servera ~]$ while ! test -f my_first_cron_job.txt; do sleep 1s; done
```

7. Verify that the contents of the `/home/student/my_first_cron_job.txt` file match the output of the `date` command.

```
8. [student@servera ~]$ cat my_first_cron_job.txt
```

```
Wed Mar 15 07:40:01 PM EDT 2023
```

9. Remove all the scheduled recurring jobs for the `student` user.
    1. Remove all the scheduled recurring jobs for the `student` user.

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

    2. Verify that no recurring jobs exist for the `student` user.

    ```
    3. [student@servera ~]$ crontab -l
    ```

    ```
    no crontab for student
    ```

    4. Return to the `workstation` machine as the `student` user.

    ```
    5. [student@servera ~]$ exit
    6. logout
    7. Connection to servera closed.
    ```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish scheduling-cron
```

This concludes the section.

Schedule Recurring System Jobs

Objectives

Schedule commands to run on a repeating schedule with the system `crontab` file and directories.

Recurring System Jobs

System administrators often need to run recurring jobs. It is best to run these jobs from system accounts rather than from user accounts. Schedule these jobs with system-wide crontab files instead of with the `crontab` command. Job entries in the system-wide crontab files are similar to the users' crontab entries. The system-wide crontab files have an extra field before the command field to specify the user that runs the command.

The `/etc/crontab` file has a syntax diagram in the comments.

```
SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root


# For details see man 4 crontabs


# Example of job definition:

# .---------------- minute (0 - 59)

# |  .------------- hour (0 - 23)

# |  |  .---------- day of month (1 - 31)

# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...

# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sa
t

# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

The `/etc/crontab` file and other files in the `/etc/cron.d/` directory define the recurring system jobs. Always create custom crontab files in the `/etc/cron.d/` directory to schedule recurring system jobs. Place the custom crontab file in the `/etc/cron.d` directory to prevent a package update from overwriting the `/etc/crontab` file. Packages that require recurring system jobs place their crontab files in the `/etc/cron.d/` directory with the job entries. Administrators also use this location to group related jobs into a single file.

The crontab system also includes repositories for scripts to run every hour, day, week, and month. These repositories are placed in the `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/`, and `/etc/cron.monthly/` directories. These directories contain executable shell scripts, not crontab files.

Note

Use the `chmod +x` *`script_name`* command to make a script executable. A script must be executable to run.

Run Periodic Commands with Anacron

The `run-parts` command also runs the daily, weekly, and monthly jobs from the `/etc/anacrontab` configuration file.

The `/etc/anacrontab` file ensures that scheduled jobs always run and are not skipped accidentally because the system was turned off or hibernated. For example, when a system job that runs daily was not executed at a specified time because the system was rebooting, then the job is completed when the system becomes ready. A delay might occur before the job starts, if specified in the `Delay in minutes` parameter in the `/etc/anacrontab` file.

Files in the `/var/spool/anacron/` directory determine the daily, weekly, and monthly jobs. When the `crond` daemon starts a job from the `/etc/anacrontab` file, it updates the timestamps of those files. With this timestamp, you can determine the last time that the job executed. The syntax of the `/etc/anacrontab` file is different from the regular `crontab` configuration files. The `/etc/anacrontab` file contains four fields per line, as follows.

`Period in days`

Defines the interval in days for the job to run on a recurring schedule. This field accepts an integer or a macro value. For example, the macro `@daily` is equivalent to the `1` integer, which executes the job daily. Similarly, the macro `@weekly` is equivalent to the `7` integer, which executes the job weekly.

`Delay in minutes`
: Defines the time that the `crond` daemon must wait before it starts the job.

`Job identifier`
: Identifies the unique name of the job in the log messages.

`Command`
: The command to be executed.

The `/etc/anacrontab` file also contains environment variable declarations with the `NAME=value` syntax. The `START_HOURS_RANGE` variable specifies the time interval for the jobs to run. Jobs do not start outside this range. When a job does not run within this time interval on a particular day, then the job must wait until the next day for execution.

Systemd Timer

The `systemd` timer unit activates another unit of a different type (such as a service), whose unit name matches the timer unit name. The timer unit allows timer-based activation of other units. The `systemd` timer unit logs timer events in system journals for easier debugging.

Sample Timer Unit

The `sysstat` package provides the `systemd` timer unit, called the `sysstat-collect.timer` service, to collect system statistics every 10 minutes. The following output shows the contents of the `/usr/lib/systemd/system/sysstat-collect.timer` configuration file.

```
...output omitted...
[Unit]
Description=Run system activity accounting tool every 10 minutes


[Timer]
OnCalendar=*:00/10
```

```
[Install]
WantedBy=sysstat.service
```

The `OnCalendar=*:00/10` option signifies that this timer unit activates the corresponding `sysstat-collect.service` unit every 10 minutes. You might specify more complex time intervals.

For example, a `2022-04-* 12:35,37,39:16` value against the `OnCalendar` option causes the timer unit to activate the corresponding service unit at the `12:35:16`, `12:37:16`, and `12:39:16` times, every day during April 2022. You might also specify relative timers with the `OnUnitActiveSec` option. For example, with the `OnUnitActiveSec=15min` option, the timer unit triggers the corresponding unit to start 15 minutes after the last time that the timer unit activated its corresponding unit.

Important

Do not modify any units in the configuration files under the `/usr/lib/systemd/system` directory, because the `systemd` unit overrides the configuration changes in that file. Create a copy of the configuration file in the `/etc/systemd/system` directory, and then modify the copied file to prevent any update to the provider package from overriding the changes. If two files exist with the same name in the `/usr/lib/systemd/system` and `/etc/systemd/system` directories, then the `systemd` timer unit parses the file in the `/etc/systemd/system` directory.

After you change the timer unit configuration file, use the `systemctl daemon-reload` command to ensure that the `systemd` timer unit loads the changes.

```
[root@host ~]# systemctl daemon-reload
```

After reloading the `systemd` daemon configuration, use the `systemctl` command to activate the timer unit.

```
[root@host ~]# systemctl enable --now <unitname>.timer
```

Guided Exercise: Schedule Recurring System Jobs

In this exercise, you schedule commands to run on various schedules by adding configuration files to the system crontab directories.

Outcomes

- Schedule a recurring system job to count the number of active users.
- Update the `systemd` timer unit that gathers system activity data.

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

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

```
[student@workstation ~]$ lab start scheduling-system
```

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

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

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

6. Schedule a recurring system job that generates a log message to indicate the number of active users in the system. This job must run daily and use the `w -h | wc -l` command to retrieve the number of active users in the system. Use the `logger` command to generate the log message of currently active users.
    1. Create the `/etc/cron.daily/usercount` script file with the following content:

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
```

```
logger "There are currently ${USERCOUNT} active users"
```

4. Make the script file executable.

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

7. Install the `sysstat` package. The timer unit must trigger the service unit every ten minutes to collect system activity data with the `/usr/lib64/sa/sa1` shell script. Change the timer unit configuration file to collect the system activity data every two minutes.
   1. Install the `sysstat` package.

```
2. [root@servera ~]# dnf install sysstat
3. ...output omitted...
4. Is this ok [y/N]: y
5. ...output omitted...
```

```
Complete!
```

6. Copy the `/usr/lib/systemd/system/sysstat-collect.timer` file to the `/etc/systemd/system/sysstat-collect.timer` file.

```
7. [root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
```

```
/etc/systemd/system/sysstat-collect.timer
```

8. Edit the `/etc/systemd/system/sysstat-collect.timer` file for the timer unit to run every two minutes. Replace any occurrence of the `10 minutes` string with `2 minutes` throughout the unit configuration file, including the occurrences in the commented lines. Use the `vim /etc/systemd/system/sysstat-collect.timer` command to edit the configuration file.

   From these changes, the `sysstat-collect.timer` unit triggers the `sysstat-collect.service` unit every two minutes, and collects the system activity data in a binary file in the `/var/log/sa` directory.

```
...output omitted...
# Activates activity collector every 2 minutes


[Unit]
Description=Run system activity accounting tool every 2 minutes
```

```
[Timer]
OnCalendar=*:00/2

[Install]
WantedBy=sysstat.service
```

9. Notify the `systemd` daemon of the changes.

```
[root@servera ~]# systemctl daemon-reload
```

10. Activate the `sysstat-collect.timer` unit.

```
11. [root@servera ~]# systemctl enable --now sysstat-collect.timer
```

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

12. Wait until the binary file is created in the `/var/log/sa` directory.

    The `while` command, `ls /var/log/sa | wc -l` returns `0` when the file does not exist, or returns `1` when the file exists. The `while` command pauses for one second when the file is not present. The `while` loop exits when the file is present.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ];  \
do sleep 1s; done
```

13. Verify that the binary file in the `/var/log/sa` directory was modified within two minutes.

```
14. [root@servera ~]# ls -l /var/log/sa
15. total 4
16. -rw-r--r--. 1 root root 2540 Apr  5 04:08 sa05
17. [root@servera ~]# date
```

```
Tue Apr  5 04:08:29 AM EDT 2022
```

18. Return to the `workstation` machine as the `student` user.

```
19. [root@servera ~]# exit
```

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

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish scheduling-system
```

This concludes the section.

Manage Temporary Files

Objectives

Enable and disable `systemd` timers, and configure a timer that manages temporary files.

Manage Temporary Files

Most critical applications and services use temporary files and directories. Some applications and users use the `/tmp` directory to hold transient working data, whereas other applications use task-specific locations such as daemon- and user-specific volatile directories under `/run`, which exist only in memory. When the system reboots or loses power, memory-based file systems are self-cleaning.

Commonly, daemons and scripts operate correctly only when their expected temporary files and directories exist. Additionally, purging temporary files on persistent storage is necessary to prevent disk space issues or stale working data.

Red Hat Enterprise Linux includes the `systemd-tmpfiles` tool, which provides a structured and configurable method to manage temporary directories and files.

At system boot, one of the first `systemd` service units to launch is the `systemd-tmpfiles-setup` service. This service runs the `systemd-tmpfiles` command `--create --remove` option, which reads instructions from the `/usr/lib/tmpfiles.d/*.conf`, `/run/tmpfiles.d/*.conf`, and `/etc/tmpfiles.d/*.conf` configuration files. These configuration files list files and directories that the `systemd-tmpfiles-setup` service is instructed to create, delete, or secure with permissions.

Clean Temporary Files with a Systemd Timer

To prevent long-running systems from filling up their disks with stale data, a `systemd` timer unit called `systemd-tmpfiles-clean.timer` triggers at a regular interval the `systemd-tmpfiles-clean.service` unit, which executes the `systemd-tmpfiles --clean` command.

A `systemd` timer unit configuration has a `[Timer]` section to indicate how to start the service with the same name as the timer.

Use the following `systemctl` command to view the contents of the `systemd-tmpfiles-clean.timer` unit configuration file.

```
[user@host ~]$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
#  SPDX-License-Identifier: LGPL-2.1-or-later
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)
ConditionPathExists=!/etc/initrd-release
```

```
[Timer]
OnBootSec=15min
OnUnitActiveSec=1d
```

In the preceding configuration, the `OnBootSec=15min` parameter indicates that the `systemd-tmpfiles-clean.service` unit gets triggered 15 minutes after the system boots up. The `OnUnitActiveSec=1d` parameter indicates that any further trigger to the `systemd-tmpfiles-clean.service` unit happens 24 hours after the service unit was last activated.

Change the parameters in the `systemd-tmpfiles-clean.timer` unit configuration file to meet your requirements. For example, a `30min` value for the `OnUnitActiveSec` parameter triggers the `systemd-tmpfiles-clean.service` unit 30 minutes after the service unit is last activated. As a result, the `systemd-tmpfiles-clean.service` unit gets triggered every 30 minutes after the changes are recognized.

After you change the timer unit configuration file, use the `systemctl daemon-reload` command to ensure that the `systemd` daemon loads the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

Clean Temporary Files Manually

The `systemd-tmpfiles --clean` command parses the same configuration files as the `systemd-tmpfiles --create` command, but instead of creating files and directories, it purges all files that were not accessed, changed, or modified more recently than the maximum age that is defined in the configuration file.

For detailed information about the format of the configuration files for the `systemd-tmpfiles` service, see the `tmpfiles.d`(5) man page. The syntax consists of the following columns: Type, Path, Mode, UID, GID, Age, and Argument. Type refers to the action for the `systemd-tmpfiles` service to take; for example, `d` to create a directory if it does not exist, or z to recursively restore SELinux contexts, file permissions, and ownership.

The following command purges a configuration with explanations:

```
d /run/systemd/seats 0755 root root -
```

When you create files and directories, create the `/run/systemd/seats` directory if it does not exist, with the `root` user and the `root` group as owners, and with permissions of `rwxr-xr-x`. If this directory does exist, then take no action. The `systemd-tmpfiles` service does not purge this directory automatically.

```
D /home/student 0700 student student 1d
```

Create the `/home/student` directory if it does not exist. If it does exist, then remove all its contents. When the system runs the `systemd-tmpfiles --clean` command, it removes from the directory all files that you did not access, change, or modify for more than one day.

```
L /run/fstablink - root root - /etc/fstab
```

Create the `/run/fstablink` symbolic link, to point to the `/etc/fstab` directory. Never automatically purge this line.

Configuration File Precedence

The `systemd-tmpfiles-clean` service configuration files can exist in three places:

- `/etc/tmpfiles.d/*.conf`
- `/run/tmpfiles.d/*.conf`
- `/usr/lib/tmpfiles.d/*.conf`

Use the files in the `/etc/tmpfiles.d/` directory to configure custom temporary locations, and to override vendor-provided defaults. The files in the `/run/tmpfiles.d/` directory are volatile files, which normally daemons use to manage their own runtime temporary files. Relevant RPM packages provide the files in the `/usr/lib/tmpfiles.d/` directory; therefore do not edit these files.

If a file in the `/run/tmpfiles.d/` directory has the same file name as a file in the `/usr/lib/tmpfiles.d/` directory, then the service uses the file in the `/run/tmpfiles.d/` directory. If a file in the `/etc/tmpfiles.d/` directory has the same file name as a file in either the `/run/tmpfiles.d/` or the `/usr/lib/tmpfiles.d/` directories, then the service uses the file in the `/etc/tmpfiles.d/` directory.

Given these precedence rules, you can override vendor-provided settings by copying the relevant file to the `/etc/tmpfiles.d/` directory and then editing it. By using these configuration locations correctly, you can manage administrator-configured settings from a central configuration management system, and package updates do not overwrite your configured settings.

Note

When testing new or modified configurations, apply only the commands from a single configuration file at a time. Specify the name of the single configuration file on the `systemd-tmpfiles` command line.

Guided Exercise: Manage Temporary Files

In this exercise, you configure `systemd-tmpfiles` to change how quickly it removes temporary files from the `/tmp` directory, and also to periodically purge files from another directory.

Outcomes

- Configure `systemd-tmpfiles` to remove unused temporary files from the `/tmp` directory.
- Configure `systemd-tmpfiles` to periodically purge files from another directory.

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

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

```
[student@workstation ~]$ lab start scheduling-tempfiles
```

Instructions

1. Log in to the `servera` system as the `student` user and switch to the `root` user.

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

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

6. Configure the `systemd-tmpfiles` service to clean the `/tmp` directory of any unused files from the last five days. Ensure that a package update does not overwrite the configuration files.
   1. Copy the `/usr/lib/tmpfiles.d/tmp.conf` file to the `/etc/tmpfiles.d` directory.

```
2. [root@servera ~]# cp /usr/lib/tmpfiles.d/tmp.conf \
```

```
/etc/tmpfiles.d/tmp.conf
```

   3. Search for the configuration line in the `/etc/tmpfiles.d/tmp.conf` file that applies to the `/tmp` directory. Replace the existing age of the temporary files in that configuration line with the new age of 5 days. Remove from the file all the other lines, including the commented lines. You can use the `vim /etc/tmpfiles.d/tmp.conf` command to edit the configuration file.

      In the configuration, the `q` type is the same as the `d` type, and instructs the `systemd-tmpfiles` service to create the `/tmp` directory if it does not exist. The directory's octal permissions must be set to `1777`. Both the owning user and group of the `/tmp` directory must be `root`.
      The `/tmp` directory must not contain the unused temporary files from the last five days.

      The `/etc/tmpfiles.d/tmp.conf` file should appear as follows:

```
q /tmp 1777 root root 5d
```

   4. Verify the `/etc/tmpfiles.d/tmp.conf` file configuration.

      Because the command does not return any errors, it confirms that the configuration settings are correct.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
```

7. Add a new configuration that ensures that the `/run/momentary` directory exists, and that user and group ownership is set to the `root` user. The octal

permissions for the directory must be `0700`. The configuration must purge from this directory any files that remain unused in the last 30 seconds.

1. Create the `/etc/tmpfiles.d/momentary.conf` file with the following content.

   With the configuration, the `systemd-tmpfiles` service ensures that the `/run/momentary` directory exists and that its octal permissions are set to `0700`. The ownership of the `/run/momentary` directory must be the `root` user and group. The service purges from this directory any file if it remains unused for 30 seconds.

   ```
   [root@servera ~]# vim /etc/tmpfiles.d/momentary.conf
   d /run/momentary 0700 root root 30s
   ```

2. Verify the `/etc/tmpfiles.d/momentary.conf` file configuration. The command creates the `/run/momentary` directory if it does not exist.

   Because the command does not return any errors, it confirms that the configuration settings are correct.

   ```
   [root@servera ~]# systemd-tmpfiles --create \
   /etc/tmpfiles.d/momentary.conf
   ```

3. Verify that the `systemd-tmpfiles` command creates the `/run/momentary` directory with the appropriate permissions, owner, and group owner.

   The octal permission for the `/run/momentary` directory is set to `0700`, and the user and group ownership are set to `root`.

   ```
   [root@servera ~]# ls -ld /run/momentary
   drwx------. 2 root root 40 Apr  4 06:35 /run/momentary
   ```

8. Verify that the `systemd-tmpfiles --clean` command removes from the `/run/momentary` directory any file that is unused in the last 30 seconds, based on the `systemd-tmpfiles` configuration for the directory.
   1. Create the `/run/momentary/test` file.

   ```
   [root@servera ~]# touch /run/momentary/test
   ```

2. Configure your shell prompt not to return for 30 seconds.

```
[root@servera ~]# sleep 30
```

3. After your shell prompt returns, clean stale files from the `/run/momentary` directory, based on the referenced rule in the `/etc/tmpfiles.d/momentary.conf` configuration file.

   The command removes the `/run/momentary/test` file, because it remains unused for 30 seconds. This behavior is based on the referenced rule in the `/etc/tmpfiles.d/momentary.conf` configuration file.

```
[root@servera ~]# systemd-tmpfiles --clean \
/etc/tmpfiles.d/momentary.conf
```

4. Verify that the `/run/momentary/test` file does not exist.

```
5. [root@servera ~]# ls -l /run/momentary/test
```

```
ls: cannot access '/run/momentary/test': No such file or directory
```

6. Return to the `workstation` machine as the `student` user.

```
7.  [root@servera ~]# exit
8.  logout
9.  [student@servera ~]$ exit
10. logout
11. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish scheduling-tempfiles
```

This concludes the section.

Quiz: Schedule Future Tasks

Choose the correct answers to the following questions.

1.

2.

1. Which command displays all the user jobs that you scheduled to run as deferred jobs?

A	[✓ `atq`]
B	`atrm`
C	`at -c`
D	`at --display`

3. CheckResetShow Solution

4.

5.

2. Which command removes the deferred user job with the

job
number 5?

A `at -c 5`

[✓ B `atrm 5`]

C `at 5`

D `at --delete 5`

6. CheckResetShow Solution

7.

8.

3. Which command displays all the scheduled recurring user jobs for the currently logged-in user?

A `crontab -r`

[✓ B `crontab -l`]

C `crontab -u`

D `crontab -V`

9. CheckResetShow Solution

10.

11.

4. Which job format executes the `/usr/local/bin/daily_backup` command hourly from 9 AM to 6 PM on all days from Monday through Friday?

A `00 * * * Mon-Fri /usr/local/bin/daily_backup`

B `* */9 * * Mon-Fri /usr/local/bin/daily_backup`

C `00 */18 * * * /usr/local/bin/daily_backup`

D [✓ `00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`]

12. CheckResetShow Solution

13.

14.

5. Which directory contains the shell scripts to run daily?

A `/etc/cron.d`

B `/etc/cron.hourly`

C [✓ `/etc/cron.daily`]

D `/etc/cron.weekly`

15. CheckResetShow Solution

16.

17.

6. Which configuration file defines the settings for the system jobs that run daily, weekly, and monthly?

A	`/etc/crontab`
B	[✓ `/etc/anacrontab`]
C	`/etc/inittab`
D	`/etc/sysconfig/crond`

18. CheckResetShow Solution

19.

20.

7. Which `systemd` unit regularly triggers the cleanup of temporary files?

A	[✓ `systemd-tmpfiles-clean.timer`]
B	`systemd-tmpfiles-clean.service`
C	`dnf-makecache.timer`
D	`unbound-anchor.timer`

21. CheckResetShow Solution

Summary

- Deferred jobs or tasks are scheduled to run once in the future.
- Recurring user jobs execute the user's tasks on a repeating schedule.
- Recurring system jobs accomplish, on a repeating schedule, administrative tasks with system-wide impact.
- The `systemd` timer units can execute both the deferred and recurring jobs.

Chapter 3. Analyze and Store Logs

Abstract

Goal	Locate and accurately interpret system event logs for troubleshooting purposes.
Objectives	• Describe the basic Red Hat Enterprise Linux logging architecture to record events.

	• Interpret events in the relevant syslog files to troubleshoot problems or to review system status. • Find and interpret entries in the system journal to troubleshoot problems or review system status. • Configure the system journal to preserve the record of events when a server is rebooted. • Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events that are recorded by the system journal and logs.
Sections	• Describe System Log Architecture (and Quiz) • Review Syslog Files (and Guided Exercise) • Review System Journal Entries (and Guided Exercise) • Preserve the System Journal (and Guided Exercise) • Maintain Accurate Time (and Guided Exercise)
Lab	Analyze and Store Logs

Describe System Log Architecture

Objectives

Describe the basic Red Hat Enterprise Linux logging architecture to record events.

System Logging

The operating system kernel and other processes record a log of events that happen when the system is running. These logs are used to audit the system and to troubleshoot problems. You can use text utilities such as the `less` and `tail` commands to inspect these logs.

Red Hat Enterprise Linux uses a standard logging system that is based on the syslog protocol to log the system messages. Many programs use the logging system to record events and to organize them into log files. The `systemd-journald` and `rsyslog` services handle the syslog messages in Red Hat Enterprise Linux 9.

The `systemd-journald` service is at the heart of the operating system event logging architecture. The `systemd-journald` service collects event messages from many sources:

- System kernel
- Output from the early stages of the boot process
- Standard output and standard error from daemons
- Syslog events

The `systemd-journald` service restructures the logs into a standard format and writes them into a structured, indexed system journal. By default, this journal is stored on a file system that does not persist across reboots.

The `rsyslog` service reads syslog messages that the `systemd-journald` service receives from the journal when they arrive. The `rsyslog` service then processes the syslog events, and records them to its log files or forwards them to other services according to its own configuration.

The `rsyslog` service sorts and writes syslog messages to the log files that do persist across reboots in the `/var/log` directory. The service also sorts the log messages to specific log files according to the type of program that sent each message and the priority of each syslog message.

In addition to syslog message files, the `/var/log` directory contains log files from other services on the system. The following table lists some useful files in the `/var/log` directory.

Table 3.1. Selected System Log Files

Log file	**Type of stored messages**
`/var/log/messages`	Most syslog messages are logged here. Exceptions include messages about authentication and email processing, scheduled job execution, and purely debugging-related messages.
`/var/log/secure`	Syslog messages about security and authentication events.
`/var/log/maillog`	Syslog messages about the mail server.
`/var/log/cron`	Syslog messages about scheduled job execution.
`/var/log/boot.log`	Non-syslog console messages about system startup.

Some applications do not use the `syslog` service to manage their log messages. For example, the Apache Web Server saves log messages to files in a subdirectory of the `/var/log` directory.

Quiz: Describe System Log Architecture

Choose the correct answer to the following questions:

1.

2.

1. Which log file stores most syslog messages, except for the ones about authentication, mail, scheduled jobs, and debugging?

A	`/var/log/maillog`
B	`/var/log/boot.log`
C	[✓ `/var/log/messages`]
D	`/var/log/secure`

3. CheckResetShow Solution

4.

5.

2. Which log file stores

syslog
messages
about security
and
authentication
operations in
the system?

A `/var/log/maillog`

B `/var/log/boot.log`

C `/var/log/messages`

D [✓ `/var/log/secure`]

6. CheckResetShow Solution

7.

8.

3. Which service sorts and organizes syslog messages into files in the `/var/log` directory?

A [✓ `rsyslog`]

B `systemd-journald`

C `auditd`

D `tuned`

9. CheckResetShow Solution

10.

11.

4. Which
directory

accommodates the human-readable syslog files?

A	`/sys/kernel/debug`
B	`/var/log/journal`
C	`/run/log/journal`
D	[✓ `/var/log`]

12. CheckResetShow Solution

13.

14.

5. Which file stores syslog messages about the mail server?

A	`/var/log/lastlog`
B	[✓ `/var/log/maillog`]
C	`/var/log/tallylog`
D	`/var/log/boot.log`

15. CheckResetShow Solution

16.

17.

6. Which file stores syslog messages about scheduled jobs?

A	[✓ `/var/log/cron`]
B	`/var/log/tallylog`
C	`/var/log/spooler`
D	`/var/log/secure`

18. CheckResetShow Solution

19.

20.

7. Which file stores console messages about system startup?

A	`/var/log/messages`
B	`/var/log/cron`
C	[✓ `/var/log/boot.log`]
D	`/var/log/secure`

Review Syslog Files

Objectives

Interpret events in relevant syslog files to troubleshoot problems or review system status.

Log Events to the System

Many programs use the syslog protocol to log events to the system. Each log message is categorized by facility (the subsystem that produces the message) and priority (the message's severity).

The following table lists the standard syslog facilities:

Table 3.2. Overview of Syslog Facilities

Code	Facility	Facility description
0	kern	Kernel messages
1	user	User-level messages
2	mail	Mail system messages
3	daemon	System daemon messages
4	auth	Authentication and security messages
5	syslog	Internal syslog messages
6	lpr	Printer messages
7	news	Network news messages
8	uucp	UUCP protocol messages
9	cron	Clock daemon messages
10	authpriv	Non-system authorization messages
11	ftp	FTP protocol messages
16-23	local0 to local7	Custom local messages

The following table lists the standard syslog priorities in descending order:

Table 3.3. Overview of Syslog Priorities

Code	Priority	Priority description
0	emerg	System is unusable
1	alert	Action must be taken immediately

Code	Priority	Priority description
2	crit	Critical condition
3	err	Non-critical error condition
4	warning	Warning condition
5	notice	Normal but significant event
6	info	Informational event
7	debug	Debugging-level message

The `rsyslog` service uses the facility and priority of log messages to determine how to handle them. Rules configure this facility and priority in the `/etc/rsyslog.conf` file and in any file in the `/etc/rsyslog.d` directory with the `.conf` extension. Software packages can add rules by installing an appropriate file in the `/etc/rsyslog.d` directory.

Each rule that controls how to sort syslog messages has a line in one of the configuration files. The left side of each line indicates the facility and priority of the syslog messages that the rule matches. The right side of each line indicates which file to save the log message in (or where else to deliver the message). An asterisk (`*`) is a wildcard that matches all values.

For example, the following line in the `/etc/rsyslog.d` file would record messages that are sent to the `authpriv` facility at any priority to the `/var/log/secure` file:

```
authpriv.*                    /var/log/secure
```

Sometimes, log messages match more than one rule in the `rsyslog.conf` file. In such cases, one message is stored in more than one log file. The `none` keyword in the priority field indicates that no messages for the indicated facility are stored in the given file, to limit stored messages.

Instead of being logged to a file, syslog messages can also be printed to the terminals of all logged-in users. The `rsyslog.conf` file has a setting to print all the syslog messages with the `emerg` priority to the terminals of all logged-in users.

Sample Rules of the rsyslog Service

```
#### RULES ####
```

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
.emerg                                                 :omusrmsg:

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

Note

The syslog subsystem has many more features beyond the scope of this course. To explore further, refer to the `rsyslog.conf`(5) man page and the extensive HTML documentation at `/usr/share/doc/rsyslog/html/index.html` that the `rsyslog-doc` package provides.

Log File Rotation

The `logrotate` command rotates log files to prevent them from taking too much space in the `/var/log` directory. When a log file is rotated, it is renamed with an extension that indicates the rotation date. For example, the previous `/var/log/messages` file is renamed to the `/var/log/messages-20220320` file when it is rotated on 2022-03-20. After the previous log file rotates, it creates a log file and notifies the service that wrote the log file.

After rotations during typically four weeks, the earliest log file is discarded to free disk space. A scheduled job runs the `logrotate` command daily to see the rotation requirement of any log files. Most log files rotate weekly; the `logrotate` command rotates some log files faster, or more slowly, or when they reach a specific size.

Analyze a Syslog Entry

Log messages start with the earliest message at the start and the latest message at the end of the log file. The `rsyslog` service uses a standard format for recording entries in log files. The following example explains the anatomy of a log message in the `/var/log/secure` log file.

```
Mar 20 20:11:48 localhost sshd[1433]: Failed password for student from 172.25.0.10 po
rt 59344 ssh2
```

- **`Mar 20 20:11:48`** : Records the time stamp of the log entry.
- **`localhost`** : The host that sends the log message.
- **`sshd[1433]`** : The program or process name and PID number that sent the log message.
- **`Failed password for …`**: The message that was sent.

Monitor Log Events

Monitoring log files for events is helpful to reproduce issues. The `tail -f` *`/path/to/file`* command outputs the last ten lines of the specified file and continues to output newly written lines in the file.

For example, to monitor for failed login attempts, run the `tail` command in one terminal, and then run in another terminal the `ssh` command as the `root` user while a user tries to log in to the system.

In the first terminal, run the `tail` command:

```
[root@host ~]# tail -f /var/log/secure
```

In the second terminal, run the `ssh` command:

```
[root@host ~]# ssh root@hosta
root@hosta's password: redhat
...output omitted...
[root@hostA ~]#
```

The log messages are visible in the first terminal.

```
...output omitted...
Mar 20 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254 port
56801 ssh2
Mar 20 09:01:13 host sshd[2712]: pam_unix(sshd:session): session opened for user root
by (uid=0)
```

Send Syslog Messages Manually

The `logger` command sends messages to the `rsyslog` service. By default, the `logger` command sends the message to the user type with the `notice` priority (`user.notice`) unless specified otherwise with the `-p` option. It is helpful to test any change to the `rsyslog` service configuration.

To send a message to the `rsyslog` service to be recorded in the `/var/log/boot.log` log file, execute the following `logger` command:

```
[root@host ~]# logger -p local7.notice "Log entry created on host"
```

Guided Exercise: Review Syslog Files

In this exercise, you reconfigure the `rsyslog` service to write specific log messages to a new file.

Outcomes

- Configure the `rsyslog` service to write all log messages with the `debug` priority to the `/var/log/messages-debug` log file.

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

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

```
[student@workstation ~]$ lab start logs-syslog
```

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

```
2. [student@workstation ~]$ ssh student@servera
3. ...output omitted...
4. [student@servera ~]$ sudo -i
5. [sudo] password for student: student
```

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

6. Configure the `rsyslog` service on the `servera` machine to log all messages with the `debug` or higher priority, for any service to the new `/var/log/messages-debug` log file by changing the `/etc/rsyslog.d/debug.conf` configuration file.
    1. Create the `/etc/rsyslog.d/debug.conf` file with the necessary entries to redirect all log messages with the `debug` or higher priority to the `/var/log/messages-debug` log file.

    ```
    *.debug /var/log/messages-debug
    ```

    This configuration line logs syslog messages with any facility and with the `debug` or higher priority level:

- The wildcard (*****) in the facility field of the configuration line indicates any facility of log messages.
- The `rsyslog` service writes the matching messages to the `/var/log/messages-debug` log file.

2. Restart the `rsyslog` service.

```
[root@servera ~]# systemctl restart rsyslog
```

7. Verify that all the log messages with the `debug` priority appear in the `/var/log/messages-debug` log file.
   1. Generate a log message with the `user` type and the `debug` priority.

```
[root@servera ~]# logger -p user.debug "Debug Message Test"
```

   2. View the last ten log messages from the `/var/log/messages-debug` log file, and verify that you see the `Debug Message Test` message among the other log messages.

```
[root@servera ~]# tail /var/log/messages-debug
Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd" swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://www.rsyslog.com"] exiting on signal 15.
Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.
Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is not set, auto correcting this to TZ=/etc/localtime  [v8.37.0-9.el8 try http://www.rsyslog.com/e/2442 ]
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd" swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://www.rsyslog.com"] start
```

```
Feb 13 18:27:58 servera root[25416]: Debug Message Test
```

11. Return to the `workstation` system as the `student` user.

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

```
16. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish logs-syslog
```

This concludes the section.

Review System Journal Entries

Objectives

Find and interpret entries in the system journal to troubleshoot problems or review system status.

Find Events on the System Journal

The `systemd-journald` service stores logging data in a structured, indexed binary file called a *journal*. This data includes extra information about the log event. For example, for syslog events, this information includes the priority of the original message and the *facility*, which is a value that the `syslog` service assigns to track the process that originated a message.

Important

In Red Hat Enterprise Linux, the memory-based `/run/log` directory holds the system journal by default. The contents of the `/run/log` directory are lost when the system is shut down. You can change the `journald` directory to a persistent location, which is discussed later in this chapter.

To retrieve log messages from the journal, use the `journalctl` command. You can use the `journalctl` command to view all messages in the journal, or to search for specific events based on options and criteria. If you run the command as `root`, then you have full access to the journal. Although regular users can also use the `journalctl` command, the system restricts them from seeing certain messages.

```
[root@host ~]# journalctl
...output omitted...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on PipeWire Multimedia
System Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Starting Create User's Volatile F
iles and Directories...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on D-Bus User Message B
us Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Sockets.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Finished Create User's Volatile F
iles and Directories.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Basic System.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started User Manager for UID 0.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Main User Target.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Startup finished in 90ms.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started Session 6 of User root.
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session open
ed for user root(uid=0) by (uid=0)
Mar 15 04:42:17 host.lab.example.com systemd[1]: Starting Hostname Service...
Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.
lines 1951-2000/2000 (END) q
```

The `journalctl` command highlights important log messages; messages with the `notice` or `warning` priority are in bold text, whereas messages with the `error` priority or higher are in red text.

The key to successful use of the journal for troubleshooting and auditing is to limit journal searches to show only relevant output.

By default, the `journalctl` command `-n` option shows the last 10 log entries. You can adjust the number of log entries with an optional argument that specifies how many log entries to display. For example, to review the last five log entries, you can run the following `journalctl` command:

```
[root@host ~]# journalctl -n 5

Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.

Mar 15 04:42:47 host.lab.example.com systemd[1]: systemd-hostnamed.service: Deactivat
ed successfully.

Mar 15 04:47:33 host.lab.example.com systemd[2127]: Created slice User Background Tas
ks Slice.

Mar 15 04:47:33 host.lab.example.com systemd[2127]: Starting Cleanup of User's Tempor
ary Files and Directories...

Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's Tempor
ary Files and Directories.
```

Similar to the `tail` command, the `journalctl` command `-f` option outputs the last 10 lines of the system journal and continues to output new journal entries when the journal appends them. To exit the `journalctl` command `-f` option, use the **Ctrl**+**C** key combination.

```
[root@host ~]# journalctl -f

Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's Tempor
ary Files and Directories.

Mar 15 05:01:01 host.lab.example.com CROND[2197]: (root) CMD (run-parts /etc/cron.hou
rly)

Mar 15 05:01:01 host.lab.example.com run-parts[2200]: (/etc/cron.hourly) starting 0an
acron

Mar 15 05:01:01 host.lab.example.com anacron[2208]: Anacron started on 2022-03-15

Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.daily' in 29 m
in.

Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.weekly' in 49
min.

Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.monthly' in 69
min.

Mar 15 05:01:01 host.lab.example.com anacron[2208]: Jobs will be executed sequentiall
y

Mar 15 05:01:01 host.lab.example.com run-parts[2210]: (/etc/cron.hourly) finished 0an
acron

Mar 15 05:01:01 host.lab.example.com CROND[2196]: (root) CMDEND (run-parts /etc/cron.
hourly)

^C

[root@host ~]#
```

To help to troubleshoot problems, you can filter the output of the journal by the priority of the journal entries. The `journalctl` command `-p` option shows the journal entries with a specified priority level (by name or by number) or higher.
The `journalctl` command processes the `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, and `emerg` priority levels, in ascending priority order.

As an example, run the following `journalctl` command to list journal entries with the `err` priority or higher:

```
[root@host ~]# journalctl -p err

Mar 15 04:22:00 host.lab.example.com pipewire-pulse[1640]: pw.conf: execvp error 'pac
tl': No such file or direct

Mar 15 04:22:17 host.lab.example.com kernel: Detected CPU family 6 model 13 stepping
3

Mar 15 04:22:17 host.lab.example.com kernel: Warning: Intel Processor - this hardware
has not undergone testing by Red Hat and might not be certif>

Mar 15 04:22:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3) aborted
matching pattern /dev/discs/disc*

Mar 15 04:22:20 host.lab.example.com smartd[669]: In the system's table of devices NO
devices found to scan
```

You can show messages for a specified systemd unit by using the `journalctl` command `-u` option and the unit name.

```
[root@host ~]# journalctl -u sshd.service

May 15 04:30:18 host.lab.example.com systemd[1]: Starting OpenSSH server daemon...

May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on 0.0.0.0 port 22.

May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on :: port 22.

May 15 04:30:18 host.lab.example.com systemd[1]: Started OpenSSH server daemon.

May 15 04:32:03 host.lab.example.com sshd[1796]: Accepted publickey for user1 from 17
2.25.250.254 port 43876 ssh2: RSA SHA256:1UGy...>

May 15 04:32:03 host.lab.example.com sshd[1796]: pam_unix(sshd:session): session open
ed for user user1(uid=1000) by (uid=0)

May 15 04:32:26 host.lab.example.com sshd[1866]: Accepted publickey for user2 from ::
1 port 36088 ssh2: RSA SHA256:M8ik...

May 15 04:32:26 host.lab.example.com sshd[1866]: pam_unix(sshd:session): session open
ed for user user2(uid=1001) by (uid=0)

lines 1-8/8 (END) q
```

When looking for specific events, you can limit the output to a specific time frame. To limit the output to a specific time range, the `journalctl` command has the `--since` option and the `--until` option. Both options take a time argument in the *"YYYY-MM-DD hh:mm:ss"* format (the double quotation marks are required to preserve the space in the option).

The `journalctl` command assumes that the day starts at 00:00:00 when you omit the time argument. The command assumes the current day when you omit the day argument. Both options take `yesterday`, `today`, and `tomorrow` as valid arguments in addition to the date and time field.

As an example, run the following `journalctl` command to list all journal entries from today's records:

```
[root@host ~]# journalctl --since today
...output omitted...
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Session 8 of User student.
Mar 15 05:04:20 host.lab.example.com sshd[2255]: pam_unix(sshd:session): session open
ed for user student(uid=1000) by (uid=0)
Mar 15 05:04:20 host.lab.example.com systemd[1]: Starting Hostname Service...
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Hostname Service.
Mar 15 05:04:50 host.lab.example.com systemd[1]: systemd-hostnamed.service: Deactivat
ed successfully.
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Starting Mark boot as successful.
..
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Finished Mark boot as successful.
lines 1996-2043/2043 (END) q
```

Run the following `journalctl` command to list all journal entries from `2022-03-11 20:30:00` to `2022-03-14 10:00:00`:

```
[root@host ~]# journalctl --since "2022-03-11 20:30" --until "2022-03-14 10:00"
...output omitted...
```

You can also specify all entries since a relative time to the present. For example, to specify all entries in the last hour, you can use the following command:

```
[root@host ~]# journalctl --since "-1 hour"
```

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

Note

You can use other, more sophisticated time specifications with the `--since` and `--until` options. For some examples, see the `systemd.time`(7) man page.

In addition to the visible content of the journal, you can view additional log entries if you turn on the verbose output. You can use any displayed extra field to filter the output of a journal query. The verbose output is useful to reduce the output of complex searches for certain events in the journal.

```
[root@host ~]# journalctl -o verbose
Tue 2022-03-15 05:10:32.625470 EDT [s=e7623387430b4c14b2c71917db58e0ee;i...]
    _BOOT_ID=beaadd6e5c5448e393ce716cd76229d4
    _MACHINE_ID=4ec03abd2f7b40118b1b357f479b3112
    PRIORITY=6
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=systemd
    _UID=0
    _GID=0
    _TRANSPORT=journal
    _CAP_EFFECTIVE=1ffffffffff
    TID=1
    CODE_FILE=src/core/job.c
    CODE_LINE=744
    CODE_FUNC=job_emit_done_message
    JOB_RESULT=done
    _PID=1
    _COMM=systemd
    _EXE=/usr/lib/systemd/systemd
    _SYSTEMD_CGROUP=/init.scope
    _SYSTEMD_UNIT=init.scope
    _SYSTEMD_SLICE=-.slice
    JOB_TYPE=stop
    MESSAGE_ID=9d1aaa27d60140bd96365438aad20286
    _HOSTNAME=host.lab.example.com
    _CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 31
    _SELINUX_CONTEXT=system_u:system_r:init_t:s0
    UNIT=user-1000.slice
    MESSAGE=Removed slice User Slice of UID 1000.
    INVOCATION_ID=0e5efc1b4a6d41198f0cf02116ca8aa8
    JOB_ID=3220
    _SOURCE_REALTIME_TIMESTAMP=1647335432625470
lines 46560-46607/46607 (END) q
```

The following list shows some fields of the system journal that you can use to search for relevant lines to a particular process or event:

- *_COMM* is the command name.
- *_EXE* is the path to the executable file for the process.
- *_PID* is the PID of the process.
- *_UID* is the UID of the user that runs the process.
- *_SYSTEMD_UNIT* is the `systemd` unit that started the process.

You can combine multiple system journal fields to form a granular search query with the `journalctl` command. For example, the following `journalctl` command shows all related journal entries to the `sshd.service` `systemd` unit from a process with PID `2110`.

```
[root@host ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=2110
Mar 15 04:42:16 host.lab.example.com sshd[2110]: Accepted publickey for root from 172
.25.250.254 port 46224 ssh2: RSA SHA256:1UGybTe52L2jzEJa1HLVKn9QUCKrTv3ZzxnMJol1Fro
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session open
ed for user root(uid=0) by (uid=0)
```

Entries

In this exercise, you search the system journal for entries to record events that match specific criteria.

Outcomes

- Search the system journal for entries to record events based on different criteria.

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

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

```
[student@workstation ~]$ lab start logs-systemd
```

Instructions

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

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

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

4. Use the `journalctl` command `_PID=1` option to display only log events that originate from the `systemd` PID 1 process on the `servera` machine. To quit from the `journalctl` command, press **q**. The following output is an example and might differ on your system:

```
[student@servera ~]$ journalctl _PID=1
Mar 15 04:21:14 localhost systemd[1]: Finished Load Kernel Modules.
Mar 15 04:21:14 localhost systemd[1]: Finished Setup Virtual Console.
Mar 15 04:21:14 localhost systemd[1]: dracut ask for additional cmdline parameters was skipped because all trigger condition checks failed.
Mar 15 04:21:14 localhost systemd[1]: Starting dracut cmdline hook...
Mar 15 04:21:14 localhost systemd[1]: Starting Apply Kernel Variables...
```

```
11. lines 1-5 q
```

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

12. Use the `journalctl` command `_UID=81` option to display all log events that originated from a system service with a UID of 81 on the `servera` machine.

```
13. [student@servera ~]$ journalctl _UID=81
```

```
Mar 15 04:21:17 servera.lab.example.com dbus-broker-lau[727]: Ready
```

14. Use the `journalctl` command `-p warning` option to display log events with a `warning` or higher priority on the `servera` machine.

```
15. [student@servera ~]$ journalctl -p warning
16. Mar 15 04:21:14 localhost kernel: wait_for_initramfs() called before rootfs_in
    itcalls
17. Mar 15 04:21:14 localhost kernel: ACPI: PRMT not present
18. Mar 15 04:21:14 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG inform
    ation, can't access extended PCI configuration space under this bridge.
19. Mar 15 04:21:14 localhost kernel: device-mapper: core: CONFIG_IMA_DISABLE_HTAB
    LE is disabled. Duplicate IMA measurements will not be recorded in the IMA log
    .
20. ...output omitted...
21. Mar 15 04:21:18 servera.lab.example.com NetworkManager[769]: <warn>  [16473324
    78.5504] device (eth0): mtu: failure to set IPv6 MTU
22. Mar 15 04:21:27 servera.lab.example.com chronyd[751]: System clock wrong by -0
    .919695 seconds
23. Mar 15 04:22:34 servera.lab.example.com chronyd[751]: System clock wrong by 0.
    772805 seconds
24. Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error: kex_exchange_identi
    fication: Connection closed by remote host
25. lines 1-19/19 (END) q
```

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

26. Display all recorded log events in the past 10 minutes from the current time on the `servera` machine.

```
27. [student@servera ~]$ journalctl --since "-10min"
28. Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly' start
    ed
```

```
Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly' termi
nated
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error: kex_exchange_identi
fication: Connection closed by remote host
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: Connection closed by 172.2
5.250.9 port 45370
Mar 15 05:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for stu
dent from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixC
FCt+wowZLNzNlBT0
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Created slice User Slice o
f UID 1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Runtime Dire
ctory /run/user/1000...
Mar 15 05:41:14 servera.lab.example.com systemd-logind[739]: New session 1 of
user student.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Finished User Runtime Dire
ctory /run/user/1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Manager for
UID 1000...
...output omitted...
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Sockets.
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Timers.
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Mark boot as su
ccessful after the user session has run 2 minutes.
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Daily Cleanup o
f User's Temporary Directories.
lines 1-48 q
```

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

44. Use the `journalctl` command `--since` and `_SYSTEMD_UNIT="sshd.service"` options to display all the recorded log events that originated from the `sshd` service since `09:00:00` this morning on the `servera` machine.

    Note

    Online classrooms typically run on the UTC time zone. To obtain results that start at 9:00 AM in your local time zone, adjust your --since value by the amount of your offset from UTC. Alternatively, ignore the local time and use a value of 9:00 to locate journal entries that occurred since 9:00 for the `servera` time zone.

```
[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
Mar 15 09:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for stu
dent from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixC
FCt+wowZLNzNlBT0
Mar 15 09:41:15 servera.lab.example.com sshd[1105]: pam_unix(sshd:session): se
ssion opened for user student(uid=1000) by (uid=0)
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: Accepted publickey for stu
dent from 172.25.250.9 port 45374 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixC
FCt+wowZLNzNlBT0
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: pam_unix(sshd:session): se
ssion opened for user student(uid=1000) by (uid=0)
```

45. Return to the `workstation` system as the `student` user.

```
46. [student@servera ~]$ exit
47. logout
48. Connection to servera closed.
```

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

Finish

Preserve the System Journal

Objectives

Configure the system journal to preserve the record of events when a server is rebooted.

System Journal Storage

By default, Red Hat Enterprise Linux 9 stores the system journal in the `/run/log` directory, and the system clears the system journal after a reboot. You can change the configuration settings of the `systemd-journald` service in the `/etc/systemd/journald.conf` file so that the journals persist across a reboot.

The `Storage` parameter in the `/etc/systemd/journald.conf` file defines whether to store system journals in a volatile manner or persistently across a reboot. Set this parameter to `persistent`, `volatile`, `auto`, or `none`, as follows:

- `persistent`: Stores journals in the `/var/log/journal` directory, which persists across reboots. If the `/var/log/journal` directory does not exist, then the `systemd-journald` service creates it.
- `volatile`: Stores journals in the volatile `/run/log/journal` directory. Because the `/run` file system is temporary and exists only in the runtime memory, the data in it, including system journals, does not persist across a reboot.
- `auto`: If the `/var/log/journal` directory exists, then the `systemd-journald` service uses persistent storage; otherwise it uses volatile storage. This action is the default if you do not set the `Storage` parameter.
- `none`: Do not use any storage. The system drops all logs, but you can still forward the logs.

The advantage of persistent system journals is that the historical data is available immediately at boot. However, even with a persistent journal, the system does not keep all data forever. The journal has a built-in log rotation mechanism that triggers monthly. In addition, the system does not allow the journals to get larger than 10% of the file system that they are on, or leaving less than 15% of the file system free. You can modify these values for both the runtime and persistent journals in the `/etc/systemd/journald.conf` configuration file.

The `systemd-journald` process logs the current limits on the size of the journal when it starts. The following command output shows the journal entries that reflect the current size limits:

```
[user@host ~]$ journalctl | grep -E 'Runtime Journal|System Journal'

Mar 15 04:21:14 localhost systemd-journald[226]: Runtime Journal (/run/log/journal/4e
c03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.

Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: Runtime Journal (/run/log
/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.

Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: System Journal (/run/log/
journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 4.0G, 4.0G free.
```

Note

In the previous `grep` command, the vertical bar (|) symbol acts as an *or* operator. That is, the `grep` command matches any line with either the `Runtime Journal` string or the `System Journal` string from the `journalctl` command output. This command fetches the current size limits on the volatile (`Runtime`) journal store and on the persistent (`System`) journal store.

Configure Persistent System Journals

Configure the `systemd-journald` service as follows to preserve system journals persistently across a reboot:

- Create the `/var/log/journal` directory.

```
[root@host ~]# mkdir /var/log/journal
```

- Set the `Storage` parameter to the `persistent` value in the `/etc/systemd/journald.conf` file. Run your chosen text editor as the superuser to edit the `/etc/systemd/journald.conf` file.

```
[Journal]
Storage=persistent
...output omitted...
```

- Restart the `systemd-journald` service to apply the configuration changes.

```
[root@host ~]# systemctl restart systemd-journald
```

If the `systemd-journald` service successfully restarts, then the service creates subdirectories in the `/var/log/journal` directory. The subdirectory in the `/var/log/journal` directory has hexadecimal characters in its long name and contain files with the `.journal` extension. The `.journal` binary files store the structured and indexed journal entries.

```
[root@host ~]# ls /var/log/journal
4ec03abd2f7b40118b1b357f479b3112
[root@host ~]# ls /var/log/journal/4ec03abd2f7b40118b1b357f479b3112
system.journal  user-1000.journal
```

Although the system journals persist after a reboot, the `journalctl` command output includes entries from the current system boot as well as from the previous system boots. To limit the output to a specific system boot, use the `journalctl` command `-b` option. The following `journalctl` command retrieves the entries from the first system boot only:

```
[root@host ~]# journalctl -b 1
...output omitted...
```

The following `journalctl` command retrieves the entries from the second system boot only. The argument is meaningful only if the system was rebooted at least twice:

```
[root@host ~]# journalctl -b 2
...output omitted...
```

You can list the system boot events that the `journalctl` command recognizes, by using the `--list-boots` option.

```
[root@host ~]# journalctl --list-boots
  -6 27de... Wed 2022-04-13 20:04:32 EDT—Wed 2022-04-13 21:09:36 EDT
  -5 6a18... Tue 2022-04-26 08:32:22 EDT—Thu 2022-04-28 16:02:33 EDT
  -4 e2d7... Thu 2022-04-28 16:02:46 EDT—Fri 2022-05-06 20:59:29 EDT
  -3 45c3... Sat 2022-05-07 11:19:47 EDT—Sat 2022-05-07 11:53:32 EDT
  -2 dfae... Sat 2022-05-07 13:11:13 EDT—Sat 2022-05-07 13:27:26 EDT
  -1 e754... Sat 2022-05-07 13:58:08 EDT—Sat 2022-05-07 14:10:53 EDT
   0 ee2c... Mon 2022-05-09 09:56:45 EDT—Mon 2022-05-09 12:57:21 EDT
```

The following `journalctl` command retrieves the entries from the current system boot only:

```
[root@host ~]# journalctl -b
...output omitted...
```

Note

When debugging a system crash with a persistent journal, usually you must limit the journal query to the reboot before the crash happened. You can use the `journalctl` command `-b` option with a negative number to indicate how many earlier system boots to include in the output. For example, the `journalctl -b -1` command limits the output to only the previous boot.

Maintain Accurate Time

Objectives

Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Administer Local Clocks and Time Zones

System time synchronization is critical for log file analysis across multiple systems. Also, some services might require time synchronization to work correctly. Machines use the Network Time Protocol to provide and obtain correct time information over the internet. A machine might get accurate time information from public NTP services, such as the NTP Pool Project. Another option is to sync with a high-quality hardware clock to serve accurate time to local clients.

The `timedatectl` command shows an overview of the current time-related system settings, including the current time, time zone, and NTP synchronization settings of the system.

```
[user@host ~]$ timedatectl
               Local time: Wed 2022-03-16 05:53:05 EDT
           Universal time: Wed 2022-03-16 09:53:05 UTC
                 RTC time: Wed 2022-03-16 09:53:05
                Time zone: America/New_York (EDT, -0400)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

You can list a database of time zones with the `timedatectl` command `list-timezones` option:

```
[user@host ~]$ timedatectl list-timezones
```

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...output omitted...
```

The Internet Assigned Numbers Authority (IANA) provides a public time zone database, and the `timedatectl` command bases the time zone names on that database. IANA names time zones based on the continent or ocean, and then typically (not always) the largest city within the time zone region. For example, most of the US Mountain time zone is `America/Denver`.

Some localities inside the time zone have different daylight saving time rules. For example, in the US, much of the state of Arizona (US Mountain time) does not change to daylight saving time, and is in the `America/Phoenix` time zone.

Use the `tzselect` command to identify the correct time zone name. This command interactively prompts the user with questions about the system's location, and outputs the name of the correct time zone. It does not change the system's time zone setting.

The `root` user can change the system setting to update the current time zone with the `timedatectl` command `set-timezone` option. For example, the following `timedatectl` command updates the current time zone to `America/Phoenix`.

```
[root@host ~]# timedatectl set-timezone America/Phoenix
[root@host ~]# timedatectl
               Local time: Wed 2022-03-16 03:05:55 MST
           Universal time: Wed 2022-03-16 10:05:55 UTC
                 RTC time: Wed 2022-03-16 10:05:55
                Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Note

You can set a server's time zone to Coordinated Universal Time (UTC). The `tzselect` command does not include the name of the UTC time zone. Use the `timedatectl set-timezone UTC` command to set the system's current time zone to `UTC`.

Use the `timedatectl` command `set-time` option to change the system's current time. You might specify the time in the *"YYYY-MM-DD hh:mm:ss"* format, where you can omit either the date or the time. For example, the following `timedatectl` command changes the time to `09:00:00`.

```
[root@host ~]# timedatectl set-time 9:00:00
[root@host ~]# timedatectl
               Local time: Fri 2019-04-05 09:00:27 MST
           Universal time: Fri 2019-04-05 16:00:27 UTC
                 RTC time: Fri 2019-04-05 16:00:27
                Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Note

The previous example might fail with the "Failed to set time: Automatic time synchronization is enabled" error message. In that case, first disable the automatic time synchronization before manually setting the date or time, as explained after this note.

The `timedatectl` command `set-ntp` option enables or disables NTP synchronization for automatic time adjustment. The option requires either a `true` or a `false` argument to turn it on or off. For example, the following `timedatectl` command turns off NTP synchronization.

```
[root@host ~]# timedatectl set-ntp false
```

Note

In Red Hat Enterprise Linux 9, the `timedatectl set-ntp` command adjusts whether the `chronyd` NTP service is enabled. Other Linux distributions might use this setting to adjust a different NTP or a *Simple Network Time Protocol (SNTP)* service.

Enabling or disabling NTP with other utilities in Red Hat Enterprise Linux, such as in the graphical GNOME `Settings` application, also updates this setting.

Configure and Monitor the chronyd Service

The `chronyd` service tracks the usually inaccurate local *Real-Time Clock (RTC)* by synchronizing it to the configured NTP servers. If no network connectivity is available, then the `chronyd` service calculates the RTC clock drift, and records it in the file that the `driftfile` value specifies in the `/etc/chrony.conf` configuration file.

By default, the `chronyd` service uses servers from the NTP Pool Project to synchronize time and requires no additional configuration. You might need to change the NTP servers for a machine that runs on an isolated network.

The *stratum* of the NTP time source determines its quality. The stratum determines the number of hops that the machine is away from a high-performance reference clock. The reference clock is a `stratum 0` time source. An NTP server that is directly attached to the reference clock is a `stratum 1` time source. A machine that synchronizes time from the NTP server is a `stratum 2` time source.

The server and the peer are the two categories of time sources that you can declare in the `/etc/chrony.conf` configuration file. The server is one stratum above the local NTP server, and the peer is at the same stratum level. You can define multiple servers and peers in the `chronyd` configuration file, one per line.

The first argument of the `server` line is the IP address or DNS name of the NTP server. Following the server IP address or name, you can list a series of options for the server. Red Hat recommends using the `iburst` option, because then the `chronyd` service takes four measurements in a short time period for a more accurate initial clock synchronization after the service starts. For more information about the `chronyd` configuration file options, use the `man 5 chrony.conf` command.

As an example, with the following `server classroom.example.com iburst` line in the `/etc/chrony.conf` configuration file, the `chronyd` service uses the `classroom.example.com` server as the NTP time source.

```
# Use public servers from the pool.ntp.org project.
...output omitted...
server classroom.example.com iburst
...output omitted...
```

Restart the service after pointing the `chronyd` service to the `classroom.example.com` local time source.

```
[root@host ~]# systemctl restart chronyd
```

The `chronyc` command acts as a client to the `chronyd` service. After setting up NTP synchronization, verify that the local system is seamlessly using the NTP server to synchronize the system clock, by using the `chronyc sources` command. For more verbose output with additional explanations about the output, use the `chronyc sources -v` command.

```
[root@host ~]# chronyc sources -v

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /             'x' = may be in error, '~' = too variable, '?' = unusable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 172.25.254.254                3   6    17    26  +2957ns[+2244ns] +/-   25ms
```

The asterisk character (`*`) in the s (Source state) field indicates that the `chronyd` service uses the `classroom.example.com` server as a time source and is the NTP server that the machine is currently synchronized to.

Summary

- The `systemd-journald` and `rsyslog` services capture and write log messages to the appropriate files.
- The `/var/log` directory contains log files.
- Periodic rotation of log files prevents them from filling up the file-system space.
- The `systemd` journals are temporary and do not persist across a reboot.
- The `chronyd` service helps to synchronize time settings with a time source.
- You can update the time zone of the server based on its location.

Chapter 4. Archive and Transfer Files

Abstract

Goal	Archive and copy files from one system to another.
Objectives	• Archive files and directories into a compressed file with `tar`, and extract the contents of an existing `tar` archive. • Transfer files to or from a remote system securely with SSH. • Efficiently and securely synchronize the contents of a local file or directory with a remote server copy.
Sections	• Manage Compressed tar Archives (and Guided Exercise) • Transfer Files Between Systems Securely (and Guided Exercise) • Synchronize Files Between Systems Securely (and Guided Exercise)

Lab	Archive and Transfer Files

Manage Compressed tar Archives

Objectives

Archive files and directories into a compressed file with `tar`, and extract the contents of an existing `tar` archive.

Create Archives from the Command Line

An *archive* is a single regular file or device file that contains multiple files. The device file could be a tape drive, flash drive, or other removable media. When using a regular file, archiving is analogous to the `zip` utility and similar variations that are popular on most operating systems.

Note

The original, ubiquitous `zip` compression and file packaging utility uses the PKZIP (Phil Katz's ZIP for MSDOS systems) algorithm, which has evolved significantly, and is supported on RHEL with the `zip` and `unzip` commands. Many other compression algorithms have been developed since `zip` was introduced, and each has its advantages. For creating compressed archives for general use, any `tar`-supported compression algorithm is acceptable.

Archive files are used to create manageable personal backups, or to simplify transferring a set of files across a network when other methods, such as `rsync`, are unavailable or might be more complex. Archive files can be created with or without using compression to reduce the archive file size.

On Linux, the `tar` utility is the common command to create, manage, and extract archives. Use the `tar` command to gather multiple files into a single archive file. A *tar archive* is a structured sequence of file metadata and data with an index so you can extract individual files.

Files can be compressed during creation by using one of the supported compression algorithms. The `tar` command can list the contents of an archive without extracting, and can extract original files directly from both compressed and uncompressed archives.

Options of the tar Utility

One of the following `tar` command actions is required to perform a `tar` operation:

- **`-c`** or **`--create`** : Create an archive file.
- **`-t`** or **`--list`** : List the contents of an archive.
- **`-x`** or **`--extract`** : Extract an archive.

The following `tar` command general options are often included:

- **`-v`** or **`--verbose`** : Show the files that are being archived or extracted during the `tar` operation.
- **`-f`** or **`--file`** : Follow this option with the archive file name to create or open.
- **`-p`** or **`--preserve-permissions`** : Preserve the original file permissions when extracting.
- **`--xattrs`** : Enable extended attribute support, and store extended file attributes.
- **`--selinux`** : Enable SELinux context support, and store SELinux file contexts.

The following `tar` command compression options are used to select an algorithm:

- **`-a`** or **`--auto-compress`** : Use the archive's suffix to determine the algorithm to use.
- **`-z`** or **`--gzip`** : Use the `gzip` compression algorithm, which results in a `.tar.gz` suffix.
- **`-j`** or **`--bzip2`** : Use the `bzip2` compression algorithm, which results in a `.tar.bz2` suffix.
- **`-J`** or **`--xz`** : Use the `xz` compression algorithm, which results in a `.tar.xz` suffix.

Note

The `tar` command still supports the legacy option style that does not use a dash (`-`) character. You might find this syntax in legacy scripts or documentation, and the behavior is essentially the same. For command consistency, Red Hat recommends using the short- or long-option styles instead.

Create an Archive

To create an archive with the `tar` command, use the create and file options with the archive file name as the first argument, followed by a list of files and directories to include in the archive.

The `tar` command recognizes absolute and relative file name syntax. By default, tar removes the leading forward slash (`/`) character from absolute file names, so that files are stored internally with relative path names. This technique is safer, because extracting absolute path names always overwrites existing files. With files that are archived with relative path names, files can be extracted to a new directory without overwriting existing files.

The following command creates the `mybackup.tar` archive to contain the `myapp1.log`, `myapp2.log`, and `myapp2.log` files from the user's home directory. If a file with the same name as the requested archive exists in the target directory, then the `tar` command overwrites the file.

```
[user@host ~]$ tar -cf mybackup.tar myapp1.log myapp2.log myapp3.log
[user@host ~]$ ls mybackup.tar
mybackup.tar
```

A user must have read permissions on the target files that are being archived. For example, creating an archive in the `/etc` directory requires `root` privileges, because only privileged users can read all `/etc` files. An unprivileged user can create an archive of the `/etc` directory, but the archive excludes files that the user cannot read, and directories for which the user lacks the read and execute permissions.

In this example, the `root` user creates the `/root/etc-backup.tar` archive of the `/etc` directory.

```
[root@host ~]# tar -cf /root/etc-backup.tar /etc
tar: Removing leading `/' from member names
```

Important

Extended file attributes, such as access control lists (ACL) and SELinux file contexts, are not preserved by default in an archive. Use the `--acls`, `--selinux`, and `--xattrs` options to include POSIX ACLs, SELinux file contexts, and other extended attributes, respectively.

List Archive Contents

Use the `tar` command `t` option to list the file names from within the archive that are specified with the `f` option. The files list with relative name syntax, because the leading forward slash was removed during archive creation.

```
[root@host ~]# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract Archive Contents

Extract a tar archive into an empty directory to avoid overwriting existing files. When the `root` user extracts an archive, the extracted files preserve the original user and group ownership. If a regular user extracts files, then the user becomes the owner of the extracted files.

List the contents of the `/root/etc.tar` archive and then extract its files to the `/root/etcbackup` directory:

```
[root@host ~]# mkdir /root/etcbackup
[root@host ~]# cd /root/etcbackup
[root@host etcbackup]# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
[root@host etcbackup]# tar -xf /root/etc.tar
```

When you extract files from an archive, the current `umask` is used to modify each extracted file's permissions. Instead, use the `tar` command `p` option to preserve the original archived permissions for extracted files. The `--preserve-permissions` option is enabled by default for a superuser.

```
[user@host scripts]# tar -xpf /home/user/myscripts.tar
...output omitted...
```

Create a Compressed Archive

The `tar` command supports these compression methods, and others:

- **`gzip`** compression is the earlier, fastest method, and is widely available across platforms.
- **`bzip2`** compression creates smaller archives but is less widely available than `gzip`.
- **`xz`** compression is newer, and offers the best compression ratio of the available methods.

The effectiveness of any compression algorithm depends on the type of data that is compressed. Previously compressed data files, such as picture formats or RPM files, typically do not significantly compress further.

Create the `/root/etcbackup.tar.gz` archive with `gzip` compression from the contents of the `/etc` directory:

```
[root@host ~]# tar -czf /root/etcbackup.tar.gz /etc
tar: Removing leading `/' from member names
```

Create the `/root/logbackup.tar.bz2` archive with `bzip2` compression from the contents of the `/var/log` directory:

```
[root@host ~]$ tar -cjf /root/logbackup.tar.bz2 /var/log
tar: Removing leading `/' from member names
```

Create the `/root/sshconfig.tar.xz` archive with `xz` compression from the contents of the `/etc/ssh` directory:

```
[root@host ~]$ tar -cJf /root/sshconfig.tar.xz /etc/ssh
tar: Removing leading `/' from member names
```

After creating an archive, verify its table of contents with the `tar` command `tf` options. It is not necessary to specify the compression option when listing a compressed archive file, because the compression type is read from the archive's header. List the archived content in the `/root/etcbackup.tar.gz` file, which uses the `gzip` compression:

```
[root@host ~]# tar -tf /root/etcbackup.tar.gz
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract Compressed Archive Contents

The `tar` command can automatically determine which compression was used, so it is not necessary to specify the compression option. If you do include an incorrect compression type, `tar` reports that the specified compression type does not match the file's type. In the following example, the `tar` command uses the `-z` option, which indicates `gzip` compression, but the file name extension is `.xz`, which indicates `xz` compression:

```
[root@host ~]# tar -xzf /root/etcbackup.tar.xz


gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
```

Listing a compressed `tar` archive works in the same way as listing an uncompressed `tar` archive. Use the `tar` command with the `tf` option to verify the content of the compressed archive before extracting its contents:

```
[root@host logbackup]# tar -tf /root/logbackup.tar
var/log/
var/log/lastlog
var/log/README
var/log/private/
...output omitted...
```

The `gzip`, `bzip2`, and `xz` algorithms are also implemented as stand-alone commands for compressing individual files without creating an archive. With these commands, you cannot create a single compressed file of multiple files, such as a directory. As previously discussed, to create a compressed archive of multiple files, use the `tar` command with your preferred compression option. To uncompress a single compressed file or a compressed archive file without extracting its contents, use the `gunzip`, `bunzip2`, and `unxz` stand-alone commands.

The `gzip` and `xz` commands provide an `-l` option to view the uncompressed size of a compressed single or archive file. Use this option to verify that enough space is available before uncompressing or extracting a file.

```
[user@host ~]$ gzip -l file.tar.gz
         compressed        uncompressed  ratio uncompressed_name
          221603125           303841280  27.1% file.tar
[user@host ~]$ xz -l file.xz
Strms  Blocks   Compressed Uncompressed  Ratio  Check   Filename
    1       1    195.7 MiB    289.8 MiB  0.675  CRC64   file.xz
```

Transfer Files Between Systems Securely

Objectives

Transfer files to or from a remote system securely with SSH.

Transfer Remote Files with the Secure File Transfer Program

The `OpenSSH` suite securely runs shell commands on remote systems. Use the *Secure File Transfer Program (SFTP)* to interactively upload to or download files from an SSH server. This program is part of the `OpenSSH` suite. A session with the `sftp` command uses the secure authentication mechanism and encrypted data transfer to and from the SSH server.

Specify a remote location for the source or destination of the files to copy. For the format of the remote location, use `[user@]host:/path`. The `user@` part of the argument is optional. If this part is missing, then the `sftp` command uses your current local username. When you run the `sftp` command, your terminal provides an `sftp>` prompt.

```
[user@host ~]$ sftp remoteuser@remotehost
remoteuser@remotehost's password: password
Connected to remotehost.
sftp>
```

The interactive `sftp` session accepts various commands that work the same way in the remote file system as in the local file system, such as the `ls`, `cd`, `mkdir`, `rmdir`, and `pwd` commands. The `put` command uploads a file to the remote system. The `get` command downloads a file from the remote system. The `exit` command exits the `sftp` session.

List the available `sftp` commands by using the `help` command in the `sftp` session:

```
sftp> help
Available commands:
bye                                 Quit sftp
cd path                             Change remote directory to 'path'
chgrp [-h] grp path                 Change group of file 'path' to 'grp'
chmod [-h] mode path                Change permissions of file 'path' to 'mode'
chown [-h] own path                 Change owner of file 'path' to 'own'
...output omitted...
```

In an `sftp` session, you might run some commands on your local host. For most available commands, add the `l` character before the command. For example, the `pwd` command prints the current working directory on the remote host. To print the current working directory on your local host, use the `lpwd` command.

```
sftp> pwd
Remote working directory: /home/remoteuser
sftp> lpwd
Local working directory: /home/user
```

The next example uploads the `/etc/hosts` file on the local system to the newly created `/home/remoteuser/hostbackup` directory on the `remotehost` machine.

The `sftp` session expects that the `put` command is followed by a local file in the connecting user's home directory, in this case the `/home/remoteuser` directory:

```
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
Uploading /etc/hosts to /home/remoteuser/hostbackup/hosts
/etc/hosts                                 100%  227     0.2KB/s   00:00
```

To copy a whole directory tree recursively, use the `sftp` command `-r` option. The following example recursively copies the `/home/user/directory` local directory to the `remotehost` machine.

```
sftp> put -r directory
Uploading directory/ to /home/remoteuser/directory
Entering directory/
file1                                      100%    0     0.0KB/s   00:00
file2                                      100%    0     0.0KB/s   00:00
sftp> ls -l
drwxr-xr-x    2 student  student        32 Mar 21 07:51 directory
```

To download the `/etc/yum.conf` file from the remote host to the current directory on the local system, execute the `get /etc/yum.conf` command, and then exit the `sftp` session.

```
sftp> get /etc/yum.conf
Fetching /etc/yum.conf to yum.conf
/etc/yum.conf                              100%  813     0.8KB/s   00:00
sftp> exit
[user@host ~]$
```

To get a remote file with the `sftp` command on a single command line, without opening an interactive session, use the following syntax. You cannot use single command-line syntax to put files on a remote host.

```
[user@host ~]$ sftp remoteuser@remotehost:/home/remoteuser/remotefile
Connected to remotehost.
```

```
Fetching /home/remoteuser/remotefile to remotefile

remotefile                                               100%    7    15.7KB/
s   00:00
```

Transfer Files with Secure Copy Protocol

Warning

The `scp` command, which system administrators widely use to copy files to and from remote systems, is based on a historical `rcp` protocol that was not designed with security considerations. The `scp` command has a known code injection issue such that an attacker could execute arbitrary commands on the remote server. For this reason, `scp` is not covered in this course.

Although some vulnerabilities were fixed in recent years, not all can be fixed while maintaining backward compatibility. For this reason, Red Hat recommends no longer using the `scp` command in new applications or scripts, and instead using other utilities such as the `sftp` or `rsync` commands to copy files to or from a remote host.

You can find more information about this issue in https://access.redhat.com/security/cve/cve-2020-15778.

The `scp` Secure Copy command, which is also part of the `OpenSSH` suite, copies files from a remote system to the local system, or from the local system to a remote system. The command uses the SSH server to authenticate and encrypt data during transfer.

You can specify a remote location for the source or destination of the files that you are copying. As with the `sftp` command, the `scp` command uses `[user@]host` to identify the target system and username. If you do not specify a user, then the command attempts to log in with your local username as the remote username. When you run the command, your `scp` client authenticates to the remote SSH server as with the `ssh` command, by using key-based authentication or by prompting you for your password.

Guided Exercise: Transfer Files Between Systems Securely

In this exercise, you copy files from a remote system to a local directory with `sftp`.

Outcomes

- Copy files from a remote host to a directory on the local machine.

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

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

```
[student@workstation ~]$ lab start archive-transfer
```

Instructions

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

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

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

4. Use the `sftp` command to copy the `/etc/ssh` directory from the `serverb` machine to the `/home/student/serverbackup` directory on the `servera` machine.
    1. On the `servera` machine, create a `/home/student/serverbackup` directory.

    ```
    [student@servera ~]$ mkdir ~/serverbackup
    ```

    2. Use the `sftp` command to open a session to the `serverb` machine. Only the `root` user can read all the content in the `/etc/ssh` directory. When prompted, enter `redhat` as the password.

    ```
    3. [student@servera ~]$ sftp root@serverb
    4. root@serverb's password: redhat
    5. Connected to serverb.
    ```

```
sftp>
```

6. Change the local current directory to the newly created `/home/student/serverbackup` directory.

```
sftp> lcd /home/student/serverbackup/
sftp> lpwd
```

```
Local working directory: /home/student/serverbackup
```

9. Recursively copy the `/etc/ssh` directory from the `serverb` machine to the `/home/student/serverbackup` directory on the `servera` machine.

```
sftp> get -r /etc/ssh
Fetching /etc/ssh/ to ssh
Retrieving /etc/ssh
Retrieving /etc/ssh/sshd_config.d
50-redhat.conf                                  100%  719   881.5KB/s
00:00
Retrieving /etc/ssh/ssh_config.d
50-redhat.conf                                  100%  581   347.4KB/s
00:00
01-training.conf                                100%   36    25.8KB/s
00:00
moduli                                          100%  565KB  71.9MB/s
00:00
ssh_config                                      100% 1921     1.1MB/s
00:00
ssh_host_rsa_key                                100% 2602     7.2MB/s
00:00
ssh_host_rsa_key.pub                            100%  565     1.6MB/s
00:00
ssh_host_ecdsa_key                              100%  505     1.6MB/s
00:00
ssh_host_ecdsa_key.pub                          100%  173   528.6KB/s
00:00
ssh_host_ed25519_key                            100%  399     1.0MB/s
00:00
ssh_host_ed25519_key.pub                        100%   93   275.8KB/s
00:00
```

```
sshd_config                                       100% 3730    10.3MB/s
00:00
```

26. Exit from the `sftp` session. Verify that the `/etc/ssh` directory from the `serverb` machine is copied to the `/home/student/serverbackup` directory on the `servera` machine.

```
sftp> exit
[student@servera ~]$ ls -lR ~/serverbackup
/home/student/serverbackup:
total 4
drwxr-xr-x. 4 student student 4096 Mar 21 12:01 ssh

/home/student/serverbackup/ssh:
total 600
-rw-r--r--. 1 student student 578094 Mar 21 12:01 moduli
-rw-r--r--. 1 student student   1921 Mar 21 12:01 ssh_config
drwxr-xr-x. 2 student student     52 Mar 21 12:01 ssh_config.d
-rw-------. 1 student student   3730 Mar 21 12:01 sshd_config
drwx------. 2 student student     28 Mar 21 12:01 sshd_config.d
-rw-r-----. 1 student student    505 Mar 21 12:01 ssh_host_ecdsa_key
-rw-r--r--. 1 student student    173 Mar 21 12:01 ssh_host_ecdsa_key.pub
-rw-r-----. 1 student student    399 Mar 21 12:01 ssh_host_ed25519_key
-rw-r--r--. 1 student student     93 Mar 21 12:01 ssh_host_ed25519_key.pub
-rw-r-----. 1 student student   2602 Mar 21 12:01 ssh_host_rsa_key
-rw-r--r--. 1 student student    565 Mar 21 12:01 ssh_host_rsa_key.pub

/home/student/serverbackup/ssh/ssh_config.d:
total 8
-rw-r--r--. 1 student student  36 Mar 21 12:01 01-training.conf
-rw-r--r--. 1 student student 581 Mar 21 12:01 50-redhat.conf

/home/student/serverbackup/ssh/sshd_config.d:
total 4
```

```
-rw-------. 1 student student 719 Mar 21 12:01 50-redhat.conf
```

5. Return to the `workstation` system as the `student` user.

```
6. [student@servera ~]$ exit
7. logout
8. Connection to servera closed.
```

```
[student@workstation]$
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish archive-transfer
```

This concludes the section.

Synchronize Files Between Systems Securely

Objectives

Efficiently and securely synchronize the contents of a local file or directory with a remote server copy.

Synchronize Remote Files and Directories

The `rsync` command is another way to copy files from one system to another system securely. The tool uses an algorithm that minimizes copied data by synchronizing only the changed portions of files. If two files or directories are similar between two servers, then the `rsync` command copies only the differences between the file systems.

An advantage of the `rsync` command is that it copies files securely and efficiently between a local system and a remote system. Whereas an initial directory synchronization takes about the same time as copying it, subsequent synchronizations copy only the differences over the network, which substantially accelerates updates.

Use the `rsync` command `-n` option for a dry run. A dry run simulates what happens when the command is executed. The dry run shows the changes that the `rsync` command would perform when executing the command. Perform a dry run before the actual `rsync` command operation to ensure that no critical files are overwritten or deleted.

When synchronizing with the `rsync` command, two standard options are the `-v` and `-a` options.

The `rsync` command `-v` or `--verbose` option provides a more detailed output. This option is helpful for troubleshooting and viewing live progress.

The `rsync` command `-a` or `--archive` option enables "archive mode". This option enables recursive copying and turns on many valuable options to preserve most characteristics of the files. Archive mode is the same as specifying the following options:

Table 4.1. Options Enabled with `rsync -a` (Archive Mode)

Option	Description
`-r`, `--recursive`	Synchronize the whole directory tree recursively
`-l`, `--links`	Synchronize symbolic links
`-p`, `--perms`	Preserve permissions
`-t`, `--times`	Preserve time stamps
`-g`, `--group`	Preserve group ownership
`-o`, `--owner`	Preserve the owner of the files
`-D`, `--devices`	Preserve device files

Archive mode does not preserve hard links, because it might add significant time to the synchronization. Use the `rsync` command `-H` option to preserve hard links too.

Note

To include extended attributes when syncing files, add these options to the `rsync` command:

- `-A` to preserve Access Control Lists (ACLs)
- `-X` to preserve SELinux file contexts

You can use the `rsync` command to synchronize the contents of a local file or directory with a file or directory on a remote machine, with either machine as the source. You can also synchronize the contents of two local files or directories on the same machine.

Like the `sftp` command, the `rsync` command specifies remote locations in the `[user@]host:/path` format. The remote location can be either the source or the destination system, provided that one of the two machines is local.

You must be the `root` user on the destination system to preserve file ownership. If the destination is remote, then authenticate as the `root` user. If the destination is local, then you must run the `rsync` command as the `root` user.

In this example, synchronize the local `/var/log` directory to the `/tmp` directory on the `hosta` system:

```
[root@host ~]# rsync -av /var/log hosta:/tmp
root@hosta's password: password
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...
sent 9,783 bytes  received 290,576 bytes  85,816.86 bytes/sec
total size is 11,585,690  speedup is 38.57
```

In the same way, the `/var/log` remote directory on the `hosta` machine synchronizes to the `/tmp` directory on the `host` machine:

```
[root@host ~]# rsync -av hosta:/var/log /tmp
root@hosta's password: password
receiving incremental file list
log/boot.log
log/dnf.librepo.log
log/dnf.log
```

```
...output omitted...

sent 9,783 bytes  received 290,576 bytes  85,816.86 bytes/sec
total size is 11,585,690  speedup is 38.57
```

The following example synchronizes the contents of the `/var/log` directory to the `/tmp` directory on the same machine:

```
[user@host ~]$ su -
Password: password
[root@host ~]# rsync -av /var/log /tmp
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...
log/tuned/tuned.log

sent 11,592,423 bytes  received 779 bytes  23,186,404.00 bytes/sec
total size is 11,586,755  speedup is 1.00
[user@host ~]$ ls /tmp
log  ssh-RLjDdarkKiW1
[user@host ~]$
```

Important

Correctly specifying a source directory trailing slash is important. A source directory with a trailing slash synchronizes the contents of the directory *without* including the directory itself. The contents are synced directly into the destination directory. Without the trailing slash, the source directory itself will sync to the destination directory. The source directory's contents are in the new subdirectory in the destination.

Bash **Tab**-completion automatically adds a trailing slash to directory names.

In this example, the content of the `/var/log/` directory is synchronized to the `/tmp` directory instead of the `log` directory being created in the `/tmp` directory.

```
[root@host ~]# rsync -av /var/log/ /tmp
sending incremental file list
./
README
boot.log
...output omitted...
tuned/tuned.log

sent 11,592,389 bytes  received 778 bytes  23,186,334.00 bytes/sec
total size is 11,586,755  speedup is 1.00
[root@host ~]# ls /tmp
anaconda                  dnf.rpm.log-20190318  private
audit                     dnf.rpm.log-20190324  qemu-ga
boot.log                  dnf.rpm.log-20190331  README
...output omitted...
```

Summary

- The `tar` command creates an archive file from a set of files and directories. This command also extracts and lists files from an archive file.
- The `tar` command provides a set of compression methods to reduce archive size.
- Besides providing a secure remote shell, the `SSH` service also provides the `sftp` command to transfer files securely to and from a remote system that runs the `SSH` server.
- The `rsync` command securely and efficiently synchronizes files between two directories, of which either one can be on a remote system.

Chapter 5. Tune System Performance

Abstract

Goal	Improve system performance by setting tuning parameters and adjusting the scheduling priority of processes.
Objectives	• Optimize system performance by selecting a tuning profile that the `tuned` daemon manages. • Prioritize or deprioritize specific processes, with the `nice` and `renice` commands.
Sections	• Adjust Tuning Profiles (and Guided Exercise) • Influence Process Scheduling (and Guided Exercise)
Lab	Tune System Performance

Adjust Tuning Profiles

Objectives

Optimize system performance by selecting a tuning profile that the `tuned` daemon manages.

Tune Systems

System administrators optimize the performance of a system by adjusting device settings based on various use case workloads. The `tuned` daemon applies tuning adjustments both statically and dynamically by using tuning profiles that reflect particular workload requirements.

Configure Static Tuning

The `tuned` daemon applies system settings when a service starts or on selecting a new tuning profile. Static tuning configures predefined `kernel` parameters in profiles that the `tuned` daemon applies at runtime. With static tuning, the `tuned` daemon sets kernel parameters for overall performance expectations, without adjusting these parameters as activity levels change.

Configure Dynamic Tuning

With dynamic tuning, the `tuned` daemon monitors system activity and adjusts settings according to runtime behavior changes. Dynamic tuning continuously adjusts tuning to fit the current workload, starting with the initial declared settings in your selected tuning profile.

For example, storage devices experience high use during startup and login, but have minimal activity when user workloads consist of using web browsers and email clients. Similarly, CPU and network devices experience activity increases during peak usage throughout a workday. The `tuned` daemon monitors the activity of these components, and adjusts parameter settings to maximize performance during high-activity times and to reduce settings during low activity. Predefined tuning profiles provide performance parameters that the `tuned` daemon uses.

To monitor and adjust parameter settings, the `tuned` daemon uses modules called monitor and tuning plug-ins, respectively.

Monitor plug-ins analyze the system and obtain information from it, so the tuning plug-ins use this information for dynamic tuning. At this moment, the `tuned` daemon ships with three monitor plug-ins:

- `disk`: Monitors the disk load based on the number of I/O operations for every disk device.
- `net`: Monitors the network load based on the number of transferred packets per network card.
- `load`: Monitors the CPU load for every CPU.

Tuning plug-ins tune the individual subsystems. They use the data from the monitor plug-ins and the performance parameters from the predefined tuning profiles. Among others, the `tuned` daemon ships with the following tuning plug-ins:

- `disk`: Sets different disk parameters, for example, the disk scheduler, the spin-down timeout, or the advanced power management.
- `net`: Configures the interface speed and the Wake-on-LAN (WoL) functionality.
- `cpu`: Sets different CPU parameters, for example, the CPU governor or the latency.

By default, dynamic tuning is disabled. You can enable it by setting the `dynamic_tuning` variable to 1 in the `/etc/tuned/tuned-main.conf` configuration file. If you enable dynamic tuning, then the `tuned` daemon periodically monitors the system and adjusts the tuning settings to runtime behavior changes. You can set the time in seconds between updates by using the `update_interval` variable in the `/etc/tuned/tuned-main.conf` configuration file.

```
[root@host ~]$ cat /etc/tuned/tuned-main.conf
...output omitted...
# Dynamicaly tune devices, if disabled only static tuning will be used.
dynamic_tuning = 1
...output omitted...
# Update interval for dynamic tunings (in seconds).
# It must be multiply of the sleep_interval.
update_interval = 10
...output omitted...
```

The tuned Utility

A minimal Red Hat Enterprise Linux installation includes the `tuned` package by default. You can install and enable the package manually by using the following commands:

```
[root@host ~]$ dnf install tuned
...output omitted...
[root@host ~]$ systemctl enable --now tuned
Created symlink /etc/systemd/system/multi-user.target.wants/tuned.service → /usr/lib/
systemd/system/tuned.service.
```

The `tuned` application provides profiles in the following categories:

- Power-saving profiles

- Performance-boosting profiles

The performance-boosting profiles include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

The next table shows a list of the tuning profiles that are distributed with Red Hat Enterprise Linux 9:

Table 5.1. Tuning Profiles Distributed with Red Hat Enterprise Linux 9

Tuned Profile	**Purpose**
`balanced`	Ideal for systems that require a compromise between power saving and performance.
`powersave`	Tunes the system for maximum power saving.
`throughput-performance`	Tunes the system for maximum throughput.
`accelerator-performance`	Tunes the same as `throughput-performance`, and also reduces the latency to less than 100 µs.
`latency-performance`	Ideal for server systems that require low latency at the expense of power consumption.
`network-throughput`	Derived from the `throughput-performance` profile. Additional network tuning parameters are applied for maximum network throughput.
`network-latency`	Derived from the `latency-performance` profile. Enables additional network tuning parameters to provide low network latency.
`desktop`	Derived from the `balanced` profile. Provides faster response of interactive applications.
`hpc-compute`	Derived from the `latency-performance` profile. Ideal for high-performance computing.
`virtual-guest`	Tunes the system for maximum performance if it runs on a virtual machine.

Tuned Profile	Purpose
`virtual-host`	Tunes the system for maximum performance if it acts as a host for virtual machines.
`intel-sst`	Optimized for systems with Intel Speed Select Technology configurations. Use it as an overlay on other profiles.
`optimize-serial-console`	Increases responsiveness of the serial console. Use it as an overlay on other profiles.

The `tuned` application stores the tuning profiles under the `/usr/lib/tuned` and `/etc/tuned` directories. Every profile has a separate directory, and inside the directory the `tuned.conf` main configuration file and, optionally, other files.

```
[root@host ~]# cd /usr/lib/tuned
[root@host tuned]# ls
accelerator-performance  hpc-compute          network-throughput       throughput-per
formance
balanced                 intel-sst            optimize-serial-console  virtual-guest
desktop                  latency-performance  powersave                virtual-host
functions                network-latency      recommend.d
[root@host tuned]$ ls virtual-guest
tuned.conf
```

A typical `tuned.conf` configuration file looks as follows:

```
[root@host tuned]# cat virtual-guest/tuned.conf
#
# tuned configuration
#

[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance
```

```
[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up.  Workloads
# that mostly use file mappings may be able to use even higher values.
#
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 30

# Filesystem I/O is usually much more efficient than swapping, so try to keep
# swapping low.  It's usually safe to go even lower than this on systems with
# server-grade storage.
vm.swappiness = 30
```

The `[main]` section in the file might include a summary of the tuning profile. This section also accepts the `include` parameter, for the profile to inherit all the settings from the referenced profile.

This configuration file is useful when creating new tuning profiles, because you can use one of the provided profiles as a basis, and then add or modify the parameters to configure. To create or modify tuning profiles, copy the tuning profile files from the `/usr/lib/tuned` directory to the `/etc/tuned` directory and then modify them. If profile directories exist with the same name under the `/usr/lib/tuned` and `/etc/tuned` directories, the latter always take precedence. Thus, never directly modify files in the `/usr/lib/tuned` system directory.

The rest of the sections in the `tuned.conf` file use the tuning plug-ins to modify parameters in the system. In the previous example, the `[sysctl]` section modifies the `vm.dirty_ratio` and `vm.swappiness` kernel parameters through the `sysctl` plug-in.

Manage Profiles from the Command Line

Use the `tuned-adm` command to change the settings of the `tuned` daemon. The `tuned-adm` command queries current settings, lists available profiles, recommends a tuning profile for the system, changes profiles directly, or turns off tuning.

You can identify the currently active tuning profile with the `tuned-adm active` command.

```
[root@host ~]# tuned-adm active
Current active profile: virtual-guest
```

The `tuned-adm list` command lists all available tuning profiles, including both built-in profiles and the custom-created tuning profiles.

```
[root@host ~]# tuned-adm list
Available profiles:
- accelerator-performance     - Throughput performance based tuning with ...
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- latency-performance         - Optimize for deterministic performance at ...
- network-latency             - Optimize for deterministic performance at ...
...output omitted...
Current active profile: virtual-guest
```

Use the `tuned-adm profile_info` command for information about a given profile.

```
[root@host ~]$ tuned-adm profile_info network-latency
Profile name:
network-latency


Profile summary:
Optimize for deterministic performance at the cost of increased power consumption, fo
cused on low latency network performance
...output omitted..
```

If no profile is specified, then the `tuned-adm profile_info` command shows the information for the active tuning profile:

```
[root@host ~]$ tuned-adm active
```

```
Current active profile: virtual-guest
[root@host ~]$ tuned-adm profile_info
Profile name:
virtual-guest

Profile summary:
Optimize for running inside a virtual guest
...output omitted..
```

Use the `tuned-adm profile` *`profilename`* command to switch to a different active profile that better matches the system's current tuning requirements.

```
[root@host ~]$ tuned-adm profile throughput-performance
[root@host ~]$ tuned-adm active
Current active profile: throughput-performance
```

The `tuned-adm recommend` command can recommend a tuning profile for the system. The system uses this mechanism to determine the default profile after its installation.

```
[root@host ~]$ tuned-adm recommend
virtual-guest
```

Note

The `tuned-adm recommend` command bases its recommendation on various system characteristics, including whether the system is a virtual machine and other predefined selected categories during system installation.

To revert the setting changes that the current profile applied, either switch to another profile or deactivate the tuned daemon. Turn off the `tuned` application tuning activity by using the `tuned-adm off` command.

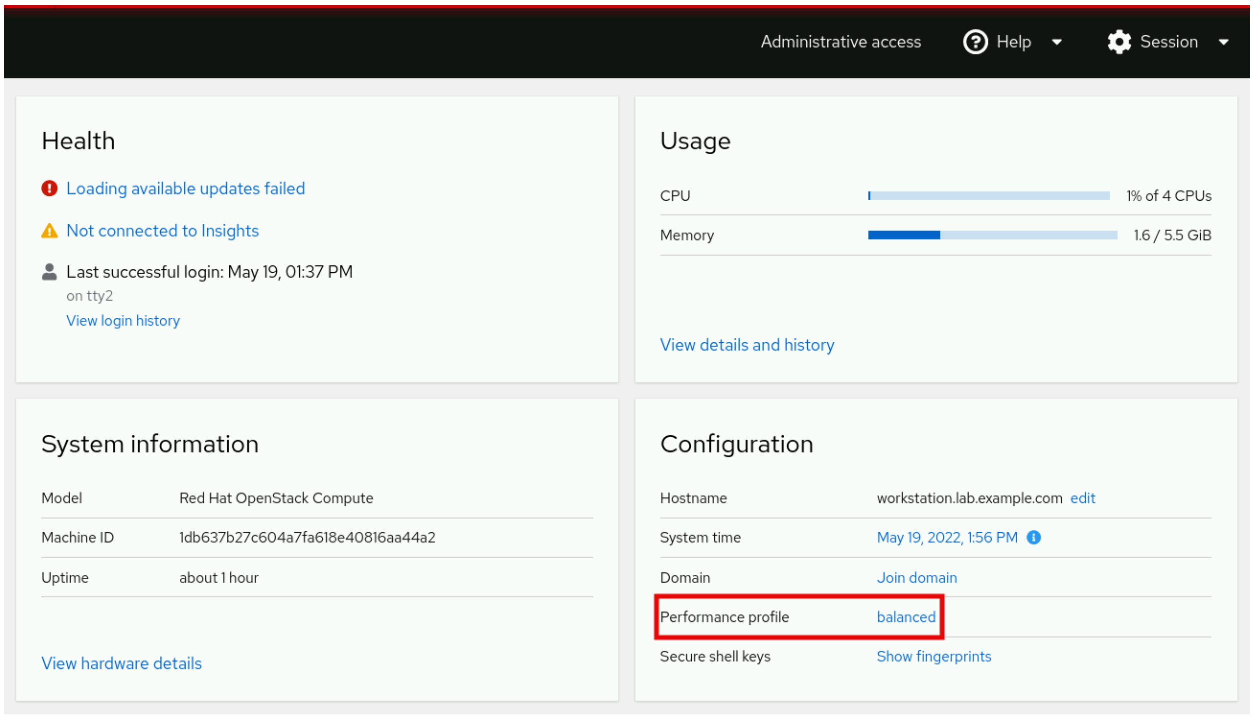
```
[root@host ~]$ tuned-adm off
[root@host ~]$ tuned-adm active
No current active profile.
```

Manage Profiles with the Web Console

To manage system performance profiles with the web console, you must log in and escalate privileges. Privilege escalation mode permits the user to execute commands, with administrative privileges, that modify system performance profiles. Because changing tuning profiles modifies some system parameters, you must do it with administrative privileges.

You can switch to the administrative access mode in the web console by clicking the **Limited access** or the **Turn on administrative access** buttons. Then, enter your password when prompted. After you escalate privileges, the **Limited access** button changes to **Administrative access**. As a security reminder, always toggle back to limited access mode after completing the system task that requires administrative privileges.

As a privileged user, click the **Overview** menu option in the left navigation bar. The **Performance profile** field displays the current active profile.



Health

Loading available updates failed

Not connected to Insights

Last successful login: May 19, 01:37 PM
on tty2
View login history

Usage

CPU		1% of 4 CPUs
Memory		1.6 / 5.5 GiB

View details and history

System information

Model	Red Hat OpenStack Compute
Machine ID	1db637b27c604a7fa618e40816aa44a2
Uptime	about 1 hour

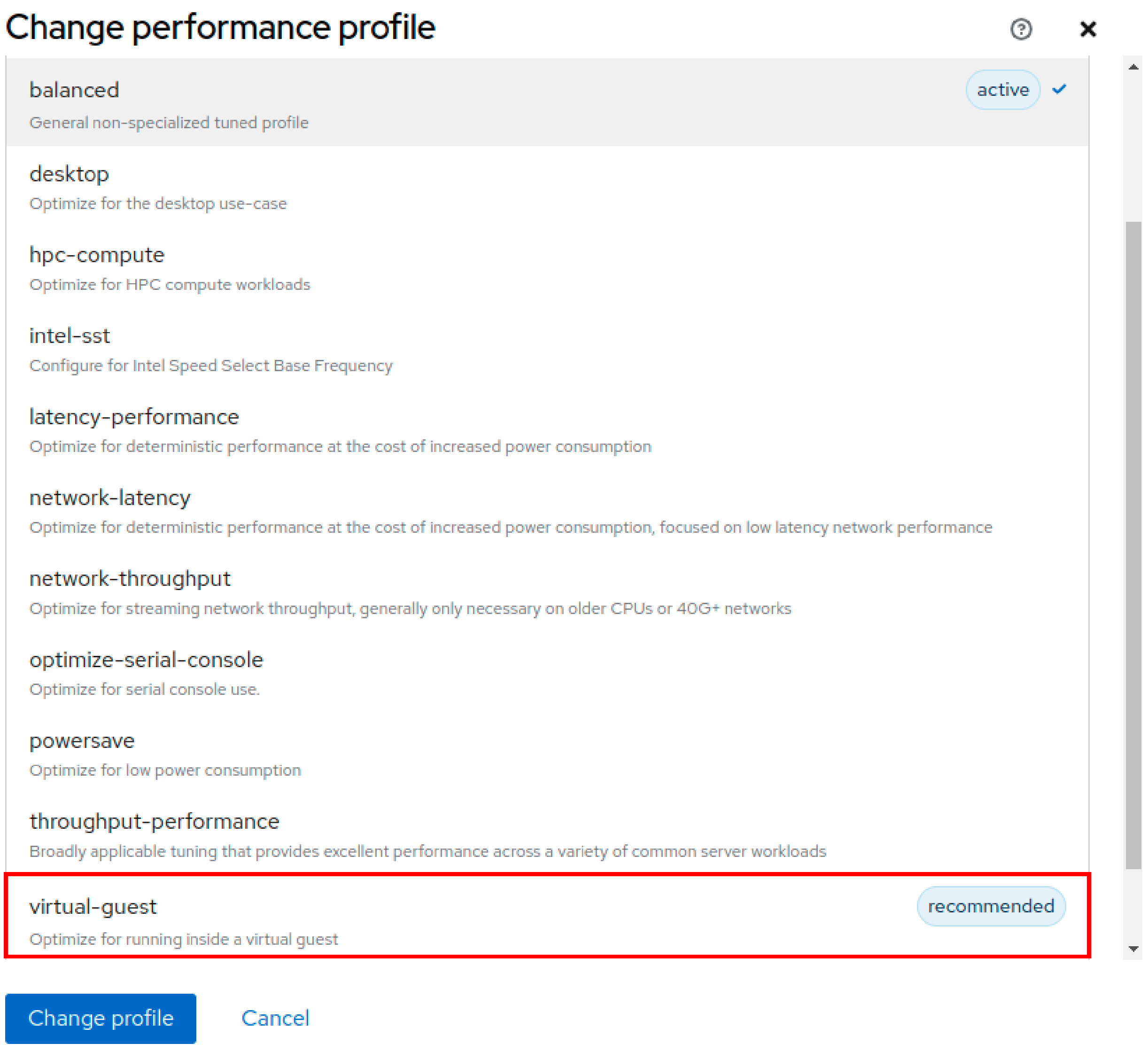
View hardware details

Configuration

Hostname	workstation.lab.example.com edit
System time	May 19, 2022, 1:56 PM
Domain	Join domain
Performance profile	balanced
Secure shell keys	Show fingerprints

Figure 5.1: Active performance profile

To select a different profile, click the active profile link. In the **Change performance profile** user interface, scroll through the profile list to select one that best suits the system purpose, and click the **Change profile** button.



Change performance profile

balanced — active ✓
General non-specialized tuned profile

desktop
Optimize for the desktop use-case

hpc-compute
Optimize for HPC compute workloads

intel-sst
Configure for Intel Speed Select Base Frequency

latency-performance
Optimize for deterministic performance at the cost of increased power consumption

network-latency
Optimize for deterministic performance at the cost of increased power consumption, focused on low latency network performance

network-throughput
Optimize for streaming network throughput, generally only necessary on older CPUs or 40G+ networks

optimize-serial-console
Optimize for serial console use.

powersave
Optimize for low power consumption

throughput-performance
Broadly applicable tuning that provides excellent performance across a variety of common server workloads

virtual-guest — recommended
Optimize for running inside a virtual guest

Figure 5.2: Select a preferred performance profile

To verify changes, return to the main **Overview** page, and confirm that it displays the active profile in the **Performance profile** field.

Guided Exercise: Adjust Tuning Profiles

In this exercise, you tune server performance by activating the `tuned` service and applying a tuning profile.

Outcomes

- Configure a system to use a tuning profile.

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

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

```
[student@workstation ~]$ lab start tuning-profiles
```

Instructions

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

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

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

4. Verify that the `tuned` package is installed, enabled, and started.
   1. Verify that the `tuned` package is installed.

```
[student@servera ~]$ dnf list tuned
...output omitted...
Installed Packages
```

```
tuned.noarch                  2.18.0-1.el9                     @System
```

   5. Verify that the service is enabled.

```
[student@servera ~]$ systemctl is-enabled tuned
```

```
enabled
```

7. Verify that the service is currently running.

```
[student@servera ~] systemctl is-active tuned
```

```
active
```

5. List the available tuning profiles and identify the active profile.

```
[student@servera ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- accelerator-performance     - Throughput performance based tuning with disabled higher latency STOP states
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- latency-performance         - Optimize for deterministic performance at the cost of increased power consumption
- network-latency             - Optimize for deterministic performance at the cost of increased power consumption, focused on low latency network performance
- network-throughput          - Optimize for streaming network throughput, generally only necessary on older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent performance across a variety of common server workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
```

```
Current active profile: virtual-guest
```

22. Review the `tuned.conf` configuration file for the current active profile, `virtual-guest`. You can find the `tuned.conf` configuration file in the `/usr/lib/tuned/virtual-guest` directory. The `virtual-guest` tuning profile is based on the `throughput-performance` profile, but it sets different values for the `vm.dirty_ratio` and `vm.swappiness` parameters. Verify that the `virtual-guest` tuning profile applies these values on your system.

1. Review the `virtual-guest` configuration file in the `/usr/lib/tuned/virtual-guest` directory. Verify the values for the `vm.dirty_ratio` and `vm.swappiness` parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/virtual-guest/tuned.conf
#
# tuned configuration
#

[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance

[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up.  Workloads
# that mostly use file mappings may be able to use even higher values.
#
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 30

# Filesystem I/O is usually much more efficient than swapping, so try to keep
# swapping low.  It's usually safe to go even lower than this on systems with
# server-grade storage.
```

```
vm.swappiness = 30
```

24. Verify that the tuning profile applies these values on your system.

```
[student@servera ~]$ sysctl vm.dirty_ratio
vm.dirty_ratio = 30
```

```
[student@servera ~]$ sysctl vm.swappiness
```

```
vm.swappiness = 30
```

23. Review the `tuned.conf` configuration file for the `virtual-guest` parent's tuning profile, `throughput-performance`. You can find it in the `/usr/lib/tuned/throughput-performance` directory. Notice that the `throughput-performance` tuning profile sets a different value for the `vm.dirty_ratio` and `vm.swappiness` parameters, although the `virtual-guest` profile overwrites them. Verify that the `virtual-guest` tuning profile applies the value for the `vm.dirty_background_ratio` parameter, which it inherits from the `throughput-performance` profile.
    1. Review the `throughput-performance` configuration file in the `/usr/lib/tuned/throughput-performance` directory. Verify the values for the `vm.dirty_ratio`, `vm.swappiness`, and `vm.dirty_background_ratio` parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/throughput-performance/tuned.conf
#
# tuned configuration
#

[main]
summary=Broadly applicable tuning that provides excellent performance across a variety of common server workloads

...output omitted...

[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up.  Workloads
# that mostly use file mappings may be able to use even higher values.
#
```

```
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 40

# Start background writeback (via writeback threads) at this percentage (system
# default is 10%)
vm.dirty_background_ratio = 10

# PID allocation wrap value.  When the kernel's next PID value
# reaches this value, it wraps back to a minimum PID value.
# PIDs of value pid_max or larger are not allocated.
#
# A suggested value for pid_max is 1024 * <# of cpu cores/threads in system>
# e.g., a box with 32 cpus, the default of 32768 is reasonable, for 64 cpus,
# 65536, for 4096 cpus, 4194304 (which is the upper limit possible).
#kernel.pid_max = 65536

# The swappiness parameter controls the tendency of the kernel to move
# processes out of physical memory and onto the swap disk.
# 0 tells the kernel to avoid swapping processes out of physical memory
# for as long as possible
# 100 tells the kernel to aggressively swap processes out of physical memory
# and move them to swap cache
vm.swappiness=10

```

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

43. Verify that the `virtual-guest` tuning profile applies the inherited `vm.dirty_background_ratio` parameter.

```
[student@servera ~]$ sysctl vm.dirty_background_ratio
```

```
vm.dirty_background_ratio = 10
```

24. Change the current active tuning profile to `throughput-performance`, and then confirm the results. Verify that
the `vm.dirty_ratio` and `vm.swappiness` parameters change to the values in the `throughput-performance` configuration file.
    1. Change the current active tuning profile.

```
[student@servera ~]$ sudo tuned-adm profile throughput-performance
```

    2. Confirm that `throughput-performance` is the active tuning profile.

```
3. [student@servera ~]$ sudo tuned-adm active
```

```
Current active profile: throughput-performance
```

    4. Verify the values for the `vm.dirty_ratio` and `vm.swappiness` parameters.

```
5. [student@servera ~]$ sysctl vm.dirty_ratio
6. vm.dirty_ratio = 40
7. [student@servera ~]$ sysctl vm.swappiness
```

```
vm.swappiness = 10
```

25. Return to the `workstation` machine as the `student` user.

```
26. [student@servera ~]$ exit
27. logout
28. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish tuning-profiles
```

This concludes the section.

Influence Process Scheduling

Objectives

Prioritize or deprioritize specific processes, with the `nice` and `renice` commands.

Linux Process Scheduling

Modern computer systems use multi-core, multi-thread CPUs that can execute many instruction threads simultaneously. The largest high-performing supercomputers can have hundreds or thousands of CPUs with hundreds of processing cores and thread structures per CPU, and can process millions of instruction threads in parallel. Although a single user who runs many applications can saturate the typical desktop system or personal workstation with CPU activity, a correctly sized and configured workstation is designed to match the user's intended workload. However, the typical enterprise or internet server handles many hundreds or thousands of users and application requests each second, which can result in CPU saturation. All systems under CPU load experience scenarios that require handling more process threads than the available system CPU processing units to schedule the threads immediately.

Linux and other operating systems use a technique called *time-slicing* or *multitasking* for process management. The operating system *process scheduler* rapidly switches between process threads on each available CPU core. This behavior gives the impression that many processes are running at the same time.

Process Priorities

Process *priority* sets the importance of each process. Linux implements *scheduling policies* that define the rules by which processes are organized and prioritized to obtain CPU processing time. The various Linux scheduling policies might be designed to handle interactive application requests, or non-interactive batch application processing, or real-time application requirements. Real-time scheduling policies still use process priorities and queues. However, current, non-real-time (*normal*) scheduling policies use the Completely Fair Scheduler (CFS), which instead organizes processes that are awaiting CPU time into a binary search tree. This process priority uses the `SCHED_NORMAL` or `SCHED_OTHER` policies as the default scheduling policy.

Processes that run under the `SCHED_NORMAL` policy are assigned a *static* real-time priority of 0, to ensure that all system real-time processes have a higher priority than normal processes. However, this static priority value is not included when organizing normal process threads for CPU scheduling. Instead, the CFS scheduling algorithm arranges normal process threads into a time-weighted binary tree, where the first item has the lowest previously spent CPU time, and the last item has the most cumulative CPU time.

Nice Value

The order of the binary tree is additionally influenced by a user-modifiable, per-process *nice* value, which ranges from -20 (increased priority) to 19 (decreased priority), with a default of 0. Processes inherit their starting nice value from their parent process. All users can adjust the nice value to decrease priority, but only `root` can increase its priority.

A higher nice value indicates a decrease in the process priority from the default, or *making the process nicer* to other processes. A lower nice value indicates an increase in the process priority from the default, or *making the process less nice* to other processes.

Increasing the nice value lowers the thread's position, and decreasing the value raises the thread's position.

Important

Generally, priorities determine only indirectly the amount of CPU time that a process thread receives. On a non-saturated system with available CPU capacity, every process is scheduled for immediate CPU time, for as much time as each process wants. Relative process importance, as managed in the binary tree, determines only which threads are selected and placed on CPUs first.

On a CPU-saturated system, where more waiting threads than CPU processing units exist, higher-priority (lower nice) process threads are placed first, until all CPU units are busy, while the lower-priority (higher nice) threads initially must wait in the binary tree. However, the Completely Fair Scheduler is designed to balance process importance, nice values, and previous cumulative CPU time, and dynamically adjusts the binary tree such that all processes obtain fair CPU time.

Permission to Modify Nice Values

Privileged users can decrease the nice value of a process, to make a process less nice. A process is then repetitively placed higher in the binary tree, and therefore is scheduled more often. On a saturated system, the overall CPU time available to other processes is reduced.

Unprivileged users can only increase the nice value on their own processes, which makes their own processes nicer, and therefore lowers their placement in the binary tree. Unprivileged users cannot decrease their processes' nice values to raise their importance, nor can they adjust the nice values for another user's process.

Viewing Nice Values

Nice values map to a priority value, and both values are available for viewing in process listing commands. A nice value of -20 maps to a 0 priority in the `top` command. A nice value of 19 maps to a 39 priority in the `top` command.

Figure 5.3: Priorities and nice values as reported by the top command

In the preceding figure, the nice values are aligned with the priority values that are used by the `top` command. The `top` command displays the process priority in the `PR` column, and the nice value in the `NI` column. The `top` priority numbering scheme, which displays real-time process priorities as negative numbers, is a legacy of internal priority algorithms.

The following output is the summary and a partial process listing in the `top` command:

```
Tasks: 192 total,   1 running, 191 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  1.6 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  1.6 si,  0.0 st
MiB Mem :   5668.6 total,   4655.6 free,    470.1 used,    542.9 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   4942.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0  172180  16232  10328 S   0.0   0.3   0:01.49 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

The `ps` command displays process nice values, when using the default formatting options.

The following `ps` command lists all processes with their process ID, process name, nice value, and scheduling class. The processes are sorted in descending order by nice value. In the `CLS` scheduling class column, `TS` stands for *time sharing*, which is another name for the normal scheduling policies, including `SCHED_NORMAL`. Other `CLS` values, such as `FF` for *first in first out* and `RR` for *round robin*, indicate real-time processes. Real-time processes are not assigned nice values, as indicated by the dash (-) in the `NI` column. Advanced scheduling policies are taught in the *Red Hat Performance Tuning: Linux in Physical, Virtual, and Cloud (RH442)* course.

```
[user@host ~]$ ps axo pid,comm,nice,cls --sort=-nice
  PID COMMAND          NI CLS
   33 khugepaged       19  TS
   32 ksmd              5  TS
  814 rtkit-daemon      1  TS
    1 systemd           0  TS
    2 kthreadd          0  TS
    5 kworker/0:0-cgr   0  TS
    7 kworker/0:1-rcu   0  TS
    8 kworker/u4:0-ev   0  TS
   15 migration/0       -  FF
...output omitted...
```

Start Processes with User-set Nice Values

When a process is created, it inherits its parent's nice value. When a process starts from the command line, it inherits its nice value from the shell process. Typically, new processes run with the default nice value of 0.

The following example starts a process from the shell, and displays the process's nice value. Note the use of the PID option in the `ps` command to specify the requested output.

Note

This command was chosen for demonstration for its low resource consumption.

```
[user@host ~]$ sleep 60 &
[1] 2667
[user@host ~]$ ps -o pid,comm,nice 2667
  PID COMMAND          NI
 2667 sleep            0
```

All users can use the `nice` command to start commands with a default or higher nice value. Setting a higher value by default ensures that the new process is a lower priority than your current working shell, and would less likely affect your current interactive session.

The following example starts the same command as a background job with the default nice value, and displays the process's nice value:

```
[user@host ~]$ nice sleep 60 &
```

```
[1] 2736
[user@host ~]$ ps -o pid,comm,nice 2736
  PID COMMAND          NI
 2736 sleep            10
```

Use the `nice` command `-n` option to apply a user-defined nice value to the starting process. The default is to add 10 to the process's current nice value. The following example starts a background job with a user-defined nice value of `15` and displays the result:

```
[user@host ~]$ nice -n 15 sleep 60 &
[1] 2673
[user@host ~]$ ps -o pid,comm,nice 2740
  PID COMMAND          NI
 2740 sleep            15
```

Change the Nice Value of an Existing Process

You can change the nice value of an existing process with the `renice` command. This example uses the process ID from the previous example to change from the current nice value of 15 to a new nice value of 19.

```
[user@host ~]$ renice -n 19 2740
2740 (process ID) old priority 15, new priority 19
```

You can also use the `top` command to change the nice value on an existing process. From the `top` interactive interface, press the **r** key to access the `renice` command. Enter the process ID, and then enter the new nice value.

Guided Exercise: Influence Process Scheduling

In this exercise, you adjust the scheduling priority of processes with the `nice` and `renice` commands, and observe the effects on process execution.

Outcomes

- Adjust scheduling priorities for processes.

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

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start tuning-procscheduling
```

Important

This exercise uses commands that perform an endless checksum on a device file and intentionally use significant CPU resources.

Instructions

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

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

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

4. Determine the number of CPU cores on the `servera` machine, and then start two instances of the `sha1sum /dev/zero &` command for each core.
   1. Use the `grep` command to parse the number of existing virtual processors (CPU cores) from the `/proc/cpuinfo` file.

   ```
   2. [student@servera ~]$ grep -c '^processor' /proc/cpuinfo
   ```

   ```
   2
   ```

   3. Use a looping command to start multiple instances of the `sha1sum /dev/zero &` command. Start two instances for each virtual processor that was indicated in the previous step. In this example, a `for` loop creates four instances. The PID values in your output might vary from the example.

   ```
   4. [student@servera ~]$ for i in {1..4}; do sha1sum /dev/zero & done
   5. [1] 1132
   6. [2] 1133
   ```

```
7. [3] 1134
```

```
[4] 1135
```

5. Verify that the background jobs are running for each of the `sha1sum` processes.

```
6. [student@servera ~]$ jobs
7. [1]   Running                 sha1sum /dev/zero &
8. [2]   Running                 sha1sum /dev/zero &
9. [3]-  Running                 sha1sum /dev/zero &
```

```
[4]+  Running                 sha1sum /dev/zero &
```

10. Use the `ps` and `pgrep` commands to display the percentage of CPU usage for each `sha1sum` process.

```
11. [student@servera ~]$ ps u $(pgrep sha1sum)
12. USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
13. student  1132 49.6  0.1 225336  2288 pts/0    R    11:40   2:40 sha1sum /dev/ze
    ro
14. student  1133 49.6  0.1 225336  2296 pts/0    R    11:40   2:40 sha1sum /dev/ze
    ro
15. student  1134 49.6  0.1 225336  2264 pts/0    R    11:40   2:40 sha1sum /dev/ze
    ro
```

```
student  1135 49.6  0.1 225336  2280 pts/0    R    11:40   2:40 sha1sum /dev/ze
ro
```

16. Terminate all `sha1sum` processes, and then verify that no jobs are running.
    1. Use the `pkill` command to terminate all running processes with the `sha1sum` name pattern.

```
2. [student@servera ~]$ pkill sha1sum
3. [2]   Terminated              sha1sum /dev/zero
4. [4]+  Terminated              sha1sum /dev/zero
5. [1]-  Terminated              sha1sum /dev/zero
```

```
[3]+  Terminated              sha1sum /dev/zero
```

    6. Verify that no jobs are running.

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

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

17. Start multiple instances of the `sha1sum /dev/zero &` command, and then start one additional instance of the `sha1sum /dev/zero &` command with a nice level of 10. Start at least as many instances as the number of system virtual processors. In this example, three regular instances are started, plus another with a higher nice level.
    1. Use looping to start three instances of the `sha1sum /dev/zero &` command.

```
[student@servera ~]$ for i in {1..3}; do sha1sum /dev/zero & done
[1] 1207
[2] 1208
```

```
[3] 1209
```

    5. Use the `nice` command to start the fourth instance with a nice level of 10.

```
[student@servera ~]$ nice -n 10 sha1sum /dev/zero &
```

```
[4] 1210
```

18. Use the `ps` and `pgrep` commands to display the PID, percentage of CPU usage, nice value, and executable name for each process. The instance with the nice value of 10 displays a lower percentage of CPU usage than the other instances.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1207 64.2   0 sha1sum
 1208 65.0   0 sha1sum
 1209 63.9   0 sha1sum
```

```
 1210  8.2  10 sha1sum
```

24. Use the `sudo renice` command to lower the nice level of a process from the previous step. Use the PID value of the process instance with the nice level of 10 to lower its nice level to 5.

```
[student@servera ~]$ sudo renice -n 5 1210
[sudo] password for student: student
```

```
1210 (process ID) old priority 10, new priority 5
```

27. Repeat the `ps` and `pgrep` commands to display the CPU percentage and nice level.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1207 62.9   0 sha1sum
 1208 63.2   0 sha1sum
 1209 63.2   0 sha1sum
```

```
 1210 10.9   5 sha1sum
```

33. Use the `pkill` command to terminate all running processes with the `sha1sum` name pattern.

```
[student@servera ~]$ pkill sha1sum
```

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

35. Return to the `workstation` machine as the `student` user.

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

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

Important

Verify that you have terminated all exercise processes before leaving this exercise.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish tuning-procscheduling
```

This concludes the section.

Summary

- The `tuned` service automatically modifies device settings to meet specific system needs based on a predefined selected tuning profile.
- To revert all changes of the selected profile to the system settings, either switch to another profile or deactivate the `tuned` service.
- The system assigns a relative priority to a process to determine its CPU access. This priority is called the `nice` value of a process.
- The `nice` command assigns a priority to a process when it starts.
- The `renice` command modifies the priority of a running process.

Chapter 6. Manage SELinux Security

Abstract

Goal	Protect and manage server security by using SELinux.
Objectives	• Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system. • Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command, and apply the context defined by the SELinux policy to files and directories with the `restorecon` command. • Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult `man` pages that end with `_selinux` to find useful information about SELinux Booleans. • Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.
Sections	• Change the SELinux Enforcement Mode (and Guided Exercise) • Control SELinux File Contexts (and Guided Exercise) • Adjust SELinux Policy with Booleans (and Guided Exercise) • Investigate and Resolve SELinux Issues (and Guided Exercise)
Lab	Manage SELinux Security

Change the SELinux Enforcement Mode

Objectives

Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system.

SELinux Architecture

Security Enhanced Linux (SELinux) is a critical security feature of Linux. Access to files, ports, and other resources is controlled at a granular level. Processes are permitted to access only the resources that their SELinux policy or Boolean settings specify.

File permissions control file access for a specific user or group. However, file permissions do not prevent an authorized user with file access from using a file for an unintended purpose.

For example, with write access to a file, other editors or programs can still open and modify a structured data file that is designed for only a specific program to write to, which could result in corruption or a data security issue. File permissions do not stop such undesired access, because they do not control *how* a file is used but only *who* is allowed to read, write, or run a file.

SELinux consists of application-specific policies that the application's developers define to declare what actions and accesses are allowed for each binary executable, configuration file, and data file that the application uses. This policy is known as a *targeted policy*, because one policy defines an application's activities. Policies declare the predefined labels that are configured on individual programs, files, and network ports.

SELinux Usage

SELinux enforces a set of access rules that explicitly define allowed actions between processes and resources. Any action that is not defined in an access rule is not allowed. Because only defined actions are allowed, applications with a poor security design are still protected from malicious use. Applications or services with a targeted policy run in a *confined* domain, whereas an application without a policy runs *unconfined* but without any SELinux protection. Individual targeted policies can be disabled to assist with application and security policy development and debugging.

SELinux has the following operational modes:

- **Enforcing** : SELinux enforces the loaded policies. This mode is the default in Red Hat Enterprise Linux.
- **Permissive** : SELinux loads the policies and is active, but instead of enforcing access control rules, it logs access violations. This mode is helpful for testing and troubleshooting applications and rules.
- **Disabled** : SELinux is turned off. SELinux violations are not denied or logged. Disabling SELinux is strongly discouraged.

Important

Starting in Red Hat Enterprise Linux 9, SELinux can be fully disabled only by using the `selinux=0` kernel parameter at boot. RHEL no longer supports setting the `SELINUX=disabled` option in the `/etc/selinux/config` file.

Starting in RHEL 9, disabling SELinux in the `/etc/selinux/config` file results in SELinux starting and performing active enforcement, but without loading any policies. Because policy rules define allowed actions, if no policies are loaded then all actions are denied. This behavior is intentional, and is designed to block malicious attempts to circumvent SELinux protection.

Basic SELinux Concepts

The primary goal of SELinux is to protect user data from improper use by compromised applications or system services. Most Linux administrators are familiar with the standard user, group, and world file permission security model, which is known as *Discretionary Access Control (DAC)* because administrators set file permissions as they need. SELinux provides an additional layer of object-based security, which is defined in granular rules, which are known as *Mandatory Access Control (MAC)* because MAC policies apply to all users and cannot be bypassed for specific users by discretionary configuration settings.

For example, a web server's open firewall port allows remote anonymous access to a web client. However, a malicious user that accesses that port might try to compromise a system through an existing vulnerability. If an example vulnerability compromises the permissions for the `apache` user and group, then a malicious user might directly access the `/var/www/html` document root content, or the system's `/tmp` and `/var/tmp` directories, or other accessible files and directories.

SELinux policies are security rules that define how specific processes access relevant files, directories, and ports. Every resource entity, such as a file, process, directory, or port, has a label called an *SELinux context*. The context label matches a defined SELinux policy rule to allow a process to access the labeled resource. By default, an SELinux policy does not allow any access unless an explicit rule grants access. When no allow rule is defined, all access is disallowed.

SELinux labels have `user`, `role`, `type`, and `security level` fields. Targeted policy, which is enabled in RHEL by default, defines rules by using the `type` context. Type context names typically end with `_t`.

Figure 6.1: SELinux file context

Policy Access Rule Concepts

For example, a web server process is labeled with the `httpd_t` type context. Web server files and directories in the `/var/www/html/` directory and other locations are labeled with the `httpd_sys_content_t` type context. Temporary files in the `/tmp` and `/var/tmp` directories have the `tmp_t` type contexts as a label. The web server's ports have the `http_port_t` type context as a label.

An Apache web server process runs with the `httpd_t` type context. A policy rule permits the Apache server to access files and directories that are labeled with the `httpd_sys_content_t` type context. By default, files in the `/var/www/html` directory have the `httpd_sys_content_t` type context. A web server policy has by default no `allow` rules for using files that are labeled `tmp_t`, such as in the `/tmp` and `/var/tmp` directories, thus disallowing access. With SELinux enabled, a malicious user who uses a compromised Apache process would still not have access to the `/tmp` directory files.

A MariaDB server process runs with the `mysqld_t` type context. By default, files in the `/data/mysql` directory have the `mysqld_db_t` type context. A MariaDB server can access the `mysqld_db_t` labeled files, but has no rules to allow access to files for other services, such as `httpd_sys_content_t` labeled files.

Figure 6.2: SELinux decision-making flow

Many commands that list resources use the `-Z` option to manage SELinux contexts. For example, the `ps`, `ls`, `cp`, and `mkdir` commands all use the `-Z` option.

```
[root@host ~]# ps axZ
LABEL                               PID TTY      STAT   TIME COMMAND
system_u:system_r:kernel_t:s0         2 ?        S      0:00 [kthreadd]
system_u:system_r:kernel_t:s0         3 ?        I<     0:00 [rcu_gp]
system_u:system_r:kernel_t:s0         4 ?        I<     0:00 [rcu_par_gp]
...output omitted...
[root@host ~]# systemctl start httpd
[root@host ~]# ps -ZC httpd
LABEL                               PID TTY          TIME CMD
system_u:system_r:httpd_t:s0       1550 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0       1551 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0       1552 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0       1553 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0       1554 ?        00:00:00 httpd
[root@host ~]# ls -Z /var/www
system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
system_u:object_r:httpd_sys_content_t:s0 html
```

Change the SELinux Mode

Use the `getenforce` command to view the current SELinux mode. Use the `setenforce` command to change the SELinux mode.

```
[root@host ~]# getenforce
Enforcing
[root@host ~]# setenforce
usage:  setenforce [ Enforcing | Permissive | 1 | 0 ]
[root@host ~]# setenforce 0
[root@host ~]# getenforce
Permissive
[root@host ~]# setenforce Enforcing
```

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

Alternatively, set the SELinux mode at boot time with a kernel parameter. Pass the `enforcing=0` kernel parameter to boot the system into `permissive` mode, or pass `enforcing=1` to boot into `enforcing` mode. Disable SELinux by passing the `selinux=0` kernel parameter, or pass `selinux=1` to enable SELinux.

Red Hat recommends rebooting the server when you change the SELinux mode from `Permissive` to `Enforcing`. This reboot ensures that the services that are started in permissive mode are confined in the next boot.

Set the Default SELinux Mode

To configure SELinux persistently, use the `/etc/selinux/config` file. In the following default example, the configuration sets SELinux to the `enforcing` mode. The comments list other valid values, such as the `permissive` and `disabled` modes.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
...output omitted...
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#    grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
```

```
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protecte
d.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Guided Exercise: Change the SELinux Enforcement Mode

In this lab, you manage SELinux modes, both temporarily and persistently.

Outcomes

- View and set the current SELinux mode.

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

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

```
[student@workstation ~]$ lab start selinux-opsmode
```

Instructions

1. On the `workstation` machine, use the `ssh` command to log in to the `servera` machine as the `student` user and then switch to the `root` user.

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

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

6. Change the default SELinux mode to permissive.
   1. Use the `getenforce` command to verify the current SELinux mode on the `servera` machine.

   ```
   2. [root@servera ~]# getenforce
   ```

   ```
   Enforcing
   ```

   3. Use the `vim /etc/selinux/config` command to edit the configuration file. Change the `SELINUX` parameter from `enforcing` to `permissive` mode.

   ```
   [root@servera ~]# vim /etc/selinux/config
   ```

   4. Use the `grep` command to confirm that the `SELINUX` parameter displays the `permissive` mode.

   ```
   5. [root@servera ~]# grep '^SELINUX' /etc/selinux/config
   6. SELINUX=permissive
   ```

   ```
   SELINUXTYPE=targeted
   ```

   7. Use the `setenforce` command to change the `SELINUX` parameter to the `permissive` mode and verify the change.

   ```
   8. [root@servera ~]# setenforce 0
   9. [root@servera ~]# getenforce
   ```

   ```
   Permissive
   ```

7. Change the default SELinux mode back to the `enforcing` mode in the configuration file.
   1. Use the `vim /etc/selinux/config` command to edit the configuration file. Change the `SELINUX` parameter from `permissive` to `enforcing` mode.

   ```
   [root@servera ~]# vim /etc/selinux/config
   ```

   2. Use the `grep` command to confirm that the `SELINUX` parameter sets the `enforcing` mode on booting.

   ```
   3. [root@servera ~]# grep '^SELINUX' /etc/selinux/config
   ```

```
SELINUX=enforcing
```

```
SELINUXTYPE=targeted
```

8. Set the SELinux mode to `enforcing` on the command line. Reboot the `servera` machine and verify the SELinux mode.
   1. Use the `setenforce` command to set the current SELinux mode to the `enforcing` mode. Use the `getenforce` command to confirm that SELinux is set to the `enforcing` mode.

```
[root@servera ~]# setenforce 1
[root@servera ~]# getenforce
```

```
Enforcing
```

   4. Reboot the `servera` machine to implement the persistent configuration.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
```

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

   8. Log in to the `servera` machine and verify the SELinux mode.

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

```
Enforcing
```

9. Return to the `workstation` machine as the `student` user.

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

```
14. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish selinux-opsmode
```

This concludes the section.

Control SELinux File Contexts

Objectives

Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command and apply the context defined by the SELinux policy to files and directories with the `restorecon` command.

Initial SELinux Context

All resources, such as processes, files, and ports, are labeled with an SELinux *context*. SELinux maintains a file-based database of file labeling policies in the `/etc/selinux/targeted/contexts/files/` directory. New files obtain a default label when their file name matches an existing labeling policy.

When a new file's name does not match an existing labeling policy, the file inherits the same label as the parent directory. With labeling inheritance, all files are always labeled when created, regardless of whether an explicit policy exists for a file.

When files are created in default locations that have an existing labeling policy, or when a policy exists for a custom location, then new files are labeled with a correct SELinux context. However, if a file is created in an unexpected location without an existing labeling policy, then the inherited label might not be correct for the new file's intended purpose.

Furthermore, copying a file to a new location can cause that file's SELinux context to change, where the new context is determined by the new location's labeling policy, or from parent directory inheritance if no policy exists. A file's SELinux context can be preserved during copying to retain the context label that was determined for the file's original location. For example, the `cp -p` command preserves all file attributes where possible, and the `cp --preserve=context` command preserves only SELinux contexts, during copying.

Note

Copying a file always creates a file inode, and that inode's attributes, including the SELinux context, must be initially set, as previously discussed.

However, moving a file does not typically create an inode if the move occurs within the same file system, but instead moves the existing inode's file name to a new location. Because the existing inode's attributes do not need to be initialized, a file that is moved with `mv` preserves its SELinux context unless you set a new context on the file with the `-Z` option.

After you copy or move a file, verify that it has the appropriate SELinux context and set it correctly if necessary.

The following example demonstrates how this process works.

Create two files in the `/tmp` directory. Both files receive the `user_tmp_t` context type.

Move the first file, and copy the second file, to the `/var/www/html` directory.

- The moved file retains the file context that was labeled from the original `/tmp` directory.
- The copied file has a new inode and inherits the SELinux context from the destination `/var/www/html` directory.

The `ls -Z` command displays the SELinux context of a file. Observe the label of the files that are created in the `/tmp` directory.

```
[root@host ~]# touch /tmp/file1 /tmp/file2
[root@host ~]# ls -Z /tmp/file*
unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
```

```
unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

The `ls -Zd` command displays the SELinux context of the specified directory. Note the label on the `/var/www/html` directory and the files inside it.

```
[root@host ~]# ls -Zd /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
[root@host ~]# ls -Z /var/www/html/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

Move one file from the `/tmp` directory to the `/var/www/html` directory. Copy the other file to the same directory. Note the resulting label on each file.

```
[root@host ~]# mv /tmp/file1 /var/www/html/
[root@host ~]# cp /tmp/file2 /var/www/html/
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The moved file retained its original label and the copied file inherited the destination directory label. The `unconfined_u` is the SELinux user role, `object_r` is the SELinux role, and `s0` is the (lowest possible) sensitivity level. Advanced SELinux configurations and features use these values.

Change the SELinux Context

You can manage the SELinux context on files with the `semanage fcontext`, `restorecon`, and `chcon` commands.

The recommended method to change the context for a file is to create a file context policy by using the `semanage fcontext` command, and then to apply the specified context in the policy to the file by using the `restorecon` command. This method ensures that you can relabel the file to its correct context with the `restorecon` command whenever necessary. The advantage of this method is that you do not need to remember what the context is supposed to be, and you can correct the context on a set of files.

The `chcon` command changes the SELinux context directly on files, without referencing the system's SELinux policy. Although `chcon` is useful for testing and debugging, changing contexts manually with this method is temporary. Although file contexts that you can change manually survive a reboot, they might be replaced if you run `restorecon` to relabel the contents of the file system.

Important

When an SELinux system *relabel* occurs, all files on a system are labeled with their policy defaults. When you use `restorecon` on a file, any context that you change manually on the file is replaced if it does not match the rules in the SELinux policy.

The following example creates a directory with a `default_t` SELinux context, which it inherited from the `/` parent directory.

```
[root@host ~]# mkdir /virtual
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

The `chcon` command sets the file context of the `/virtual` directory to the `httpd_sys_content_t` type.

```
[root@host ~]# chcon -t httpd_sys_content_t /virtual
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

Running the `restorecon` command resets the context to the default value of `default_t`. Note the `Relabeled` message.

```
[root@host ~]# restorecon -v /virtual
Relabeled /virtual from unconfined_u:object_r:httpd_sys_content_t:s0 to unconfined_u:
object_r:default_t:s0
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

Define SELinux Default File Context Policies

The `semanage fcontext` command displays and modifies the policies that determine the default file contexts. You can list all the file context policy rules by running the `semanage fcontext -l` command. These rules use extended regular expression syntax to specify the path and file names.

When viewing policies, the most common extended regular expression is `(/.*)?`, which is usually appended to a directory name. This notation is humorously called *the pirate*, because it looks like a face with an eye patch and a hooked hand next to it.

This syntax is described as "a set of characters that begin with a slash and followed by any number of characters, where the set can either exist or not exist". Stated more simply, this syntax matches the directory itself, even when empty, and also matches almost any file name that is created within that directory.

For example, the following rule specifies that the `/var/www/cgi-bin` directory, and any files in it or in its subdirectories (and in their subdirectories, and so on), have the `system_u:object_r:httpd_sys_script_exec_t:s0` SELinux context, unless a more specific rule overrides this one.

```
/var/www/cgi-bin(/.*)?  all files  system_u:object_r:httpd_sys_script_exec_t:s0
```

Note

The `all files` field option from the previous example is the default file type that `semanage` uses when you do not specify one. This option applies to all file types that you can use with `semanage`; they are the same as the standard file types as in the *Control Access to Files* chapter in the *Red Hat System Administration I* (RH124) course. You can get more information from the `semanage-fcontext`(8) man page.

Basic File Context Operations

The following table is a reference for the `semanage fcontext` command options to add, remove, or list SELinux file context policies.

Table 6.1. The `semanage fcontext` Command

Option	Description
`-a, --add`	Add a record of the specified object type.
`-d, --delete`	Delete a record of the specified object type.
`-l, --list`	List records of the specified object type.

To manage SELinux contexts, install the `policycoreutils` and `policycoreutils-python-utils` packages, which contain the `restorecon` and `semanage` commands.

To reset all files in a directory to the default policy context, first use the `semanage fcontext -l` command to locate and verify that the correct policy exists with the intended file context. Then, use the `restorecon` command on the wildcarded directory name to reset all the files recursively. In the following example, view the file contexts before and after using the `semanage` and `restorecon` commands.

First, verify the SELinux context for the files:

```
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

Then, use the `semanage fcontext -l` command to list the default SELinux file contexts:

```
[root@host ~]# semanage fcontext -l
...output omitted...
/var/www(/.*)?       all files    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

The `semanage` command output indicates that all the files and subdirectories in the `/var/www/` directory have the `httpd_sys_content_t` context by default. Running `restorecon` command on the wildcarded directory restores the default context on all files and subdirectories.

```
[root@host ~]# restorecon -Rv /var/www/
Relabeled /var/www/html/file1 from unconfined_u:object_r:user_tmp_t:s0 to unconfined_
u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The following example uses the `semanage` command to add a context policy for a new directory. First, create the `/virtual` directory with an `index.html` file inside it. View the SELinux context for the file and the directory.

```
[root@host ~]# mkdir /virtual
[root@host ~]# touch /virtual/index.html
```

```
[root@host ~]# ls -Zd /virtual/

unconfined_u:object_r:default_t:s0 /virtual

[root@host ~]# ls -Z /virtual/

unconfined_u:object_r:default_t:s0 index.html
```

Next, use the `semanage fcontext` command to add an SELinux file context policy for the directory.

```
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
```

Use the `restorecon` command on the wildcarded directory to set the default context on the directory and all files within it.

```
[root@host ~]# restorecon -RFvv /virtual

Relabeled /virtual from unconfined_u:object_r:default_t:s0 to system_u:object_r:httpd
_sys_content_t:s0

Relabeled /virtual/index.html from unconfined_u:object_r:default_t:s0 to system_u:obj
ect_r:httpd_sys_content_t:s0

[root@host ~]# ls -Zd /virtual/

drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/

[root@host ~]# ls -Z /virtual/

-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

Use the `semanage fcontext -l -C` command to view any local customizations to the default policy.

```
[root@host ~]# semanage fcontext -l -C

SELinux fcontext     type         Context


/virtual(/.*)?       all files    system_u:object_r:httpd_sys_content_t:s0
```

Guided Exercise: Control SELinux File Contexts

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

Outcomes

- Configure the `Apache` HTTP server to publish web content from a non-standard document root.

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

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

```
[student@workstation ~]$ lab start selinux-filecontexts
```

Instructions

1. Log in to `servera` as the `student` user and switch to the `root` user.

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

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

7. Configure Apache to use a document directory in a non-standard location.
   1. Create the `/custom` directory.

   ```
   [root@servera ~]# mkdir /custom
   ```

   2. Create the `index.html` file in the `/custom` directory that contains the `This is SERVERA.` text.

   ```
   [root@servera ~]# echo 'This is SERVERA.' > /custom/index.html
   ```

3. Configure Apache to use the new directory location. Edit the Apache `/etc/httpd/conf/httpd.conf` configuration file, and replace the two occurrences of the `/var/www/html` directory with the `/custom` directory. You can use the `vim /etc/httpd/conf/httpd.conf` command to do so. The following example shows the expected content of the `/etc/httpd/conf/httpd.conf` file.

```
[root@servera ~]# cat /etc/httpd/conf/httpd.conf
...output omitted...
DocumentRoot "/custom"
...output omitted...
<Directory "/custom">
```

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

8. Start and enable the Apache web service and confirm that the service is running.
   1. Start and enable the Apache web service by using the `systemctl` command.

```
[root@servera ~]# systemctl enable --now httpd
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.servic
e → /usr/lib/systemd/system/httpd.service.
```

   3. Verify that the service is running.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; ven
dor preset: disabled)
     Active: active (running) since Wed 2022-04-06 05:21:19 EDT; 22s ago
       Docs: man:httpd.service(8)
   Main PID: 1676 (httpd)
...output omitted...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Starting The Apache
HTTP Server...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Started The Apache H
TTP Server.
```

```
Apr 06 05:21:19 servera.lab.example.com httpd[1676]: Server configured,
listening on: port 80
```

9. Open a web browser on `workstation` and try to view the `http://servera/index.html` web page. You get an error message that you do not have permission to access the file.
10. To grant access to the `index.html` file on `servera`, you must configure the SELinux context. Define an SELinux file context rule that sets the context type to `httpd_sys_content_t` for the `/custom` directory and all the files under it.

```
11. [root@servera ~]# semanage fcontext -a \
```

```
-t httpd_sys_content_t '/custom(/.*)?'
```

12. Correct the file contexts in the `/custom` directory.

```
13. [root@servera ~]# restorecon -Rv /custom
14. Relabeled /custom from unconfined_u:object_r:default_t:s0 to unconfined_u:obje
    ct_r:httpd_sys_content_t:s0
```

```
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to unconf
ined_u:object_r:httpd_sys_content_t:s0
```

15. Try to view `http://servera/index.html` again in the web browser on the `workstation` machine. You should see the `This is SERVERA.` message.
16. Return to the `workstation` machine as the `student` user.

```
17. [root@servera ~]# exit
18. logout
19. [student@servera ~]$ exit
20. logout
21. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish selinux-filecontexts
```

This concludes the section.

Adjust SELinux Policy with Booleans

Objectives

Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult `man` pages that end with `_selinux` to find useful information about SELinux Booleans.

SELinux Booleans

An application or service developer writes an SELinux targeted policy to define the allowed behavior of the targeted application. A developer can include optional application behavior in the SELinux policy that can be enabled when the behavior is allowed on a specific system. SELinux Booleans enable or disable the SELinux policy's optional behavior. With Booleans, you can selectively tune the behavior of an application.

These optional behaviors are application-specific, and must be discovered and selected for each targeted application. Service-specific Booleans are documented in that service's SELinux man page. For example, the web server `httpd` service has its `httpd(8)` man page, and an `httpd_selinux(8)` man page to document its SELinux policy, including the supported process types, file contexts, and the available Boolean-enabled behaviors. The SELinux man pages are provided in the `selinux-policy-doc` package.

Use the `getsebool` command to list available Booleans for the targeted policies on this system, and the current Boolean status. Use the `setsebool` command to enable or disable the running state of these behaviors. The `setsebool -P` command option makes the setting persistent by writing to the policy file. Only privileged users can set SELinux Booleans.

```
[root@host ~]# getsebool -a
abrt_anon_write --> off
```

```
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...output omitted...
```

Example httpd Policy Boolean

The `httpd` service policy includes the `httpd_enable_homedirs` Boolean, which enables the sharing of home directories with `httpd`. Typically, a user's local home directory is accessible to the user only when logged in to the local system. Alternatively, home directories are shared and accessed by using a remote file sharing protocol, such as NFS. In both scenarios, home directories are not shared by using `https`, by default, and are not available to the user through a browser.

```
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

You can enable sharing and enable users to access their home directories with a browser. When enabled, the `httpd` service shares home directories that are labeled with the `user_home_dir_t` file context. Users can then access and manage their home directory files from a browser.

Manage the Policy Boolean

Setting SELinux Booleans with the `setsebool` command without the `-P` option is temporary, and settings return to the persistent values after rebooting. View additional information with the `semanage boolean -l` command, which lists the Booleans from the policy files, including whether a Boolean is persistent, the default and current values, and a short description.

```
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (off   ,  off)  Allow httpd to enable homedirs
[root@host ~]# setsebool httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,  off)  Allow httpd to enable homedirs
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

To list only Booleans with a current setting that is different from the default setting at boot, use the `semanage boolean -l -C` command. This example has the same result as the previous example, without requiring the `grep` filtering.

```
[root@host ~]# semanage boolean -l -C
SELinux boolean                State  Default Description


httpd_enable_homedirs          (on   ,   off)  Allow httpd to enable homedirs
```

The previous example temporarily set the current value for the `httpd_enable_homedirs` Boolean to `on`, until the system reboots. To change the default setting, use the `setsebool -P` command to make the setting persistent. The following example sets a persistent value, and then views the Boolean's information from the policy file.

```
[root@host ~]# setsebool -P httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,   on)  Allow httpd to enable homedirs
```

Use the `semanage boolean -l -C` command again. The Boolean is displayed despite the appearance that the current and default settings are the same. However, the -c option matches when the current setting is different from the default setting from the last boot. For this `httpd_enable_homedirs` example, the original default boot setting was `off`.

```
[root@host ~]# semanage boolean -l -C
SELinux boolean                State  Default Description


httpd_enable_homedirs          (on   ,   on)  Allow httpd to enable homedirs
```

Guided Exercise: Adjust SELinux Policy with Booleans

In this exercise, you configure Apache to publish web content from users' home directories.

Outcomes

- Configure the Apache web service to publish web content from the user's home directory.

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

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

```
[student@workstation ~]$ lab start selinux-booleans
```

Instructions

1. On the `workstation` machine, use the `ssh` command to log in to the `servera` machine as the `student` user and then switch to the `root` user.

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

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

6. Edit the `/etc/httpd/conf.d/userdir.conf` configuration file to enable the Apache feature so that users can publish web content from their home directory. Comment out the line in the `IfModule` section that sets the `UserDir` variable to the `disabled` value, and uncomment the line that sets the `UserDir` variable to the `public_html` value.

```
[root@servera ~]# vim /etc/httpd/conf.d/userdir.conf
<IfModule mod_userdir.c>
...output omitted...
    # UserDir disabled

...output omitted...
    UserDir public_html

...output omitted...
```

```
</IfModule>
```

16. Start and enable the Apache web service.

```
[root@servera ~]# systemctl enable --now httpd
```

17. Open another terminal window, and use the ssh command to log in to the servera machine as the student user. Create the index.html web content file in the ~/public_html directory.
    1. In another terminal window, use the ssh command to log in to the servera machine as the student user.

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

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

    4. Use the mkdir command to create the ~/public_html directory.

    ```
    [student@servera ~]$ mkdir ~/public_html
    ```

    5. Create the index.html file with the following content:

    ```
    6. [student@servera ~]$ echo 'This is student content on SERVERA.' > \
    ```

    ```
    ~/public_html/index.html
    ```

    7. For the Apache web service to serve the contents of the /home/student/public_html directory, it must be allowed to share files and subdirectories in the /home/student directory. When you created the /home/student/public_html directory, it was automatically configured to allow anyone with home directory permission to access its contents.

       Change the /home/student directory permissions to allow the Apache web service to access the public_html subdirectory.

    ```
    [student@servera ~]$ chmod 711 ~
    [student@servera ~]$ ls -ld ~
    drwx--x--x. 16 student student 4096 Nov  3 09:28 /home/student
    ```

18. Open a web browser on the `workstation` machine and enter the `http://servera/~student/index.html` address. An error message states that you do not have permission to access the file.
19. Switch to the other terminal, and use the `getsebool` command to see whether any Booleans restrict access to home directories for the `httpd` service.

```
20. [root@servera ~]# getsebool -a | grep home
21. ...output omitted...
22. httpd_enable_homedirs --> off
```

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

23. Use the `setsebool` command to enable persistent access to the home directory for the `httpd` service.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

24. Verify that you can now see the `This is student content on SERVERA.` message in the web browser after entering the `http://servera/~student/index.html` address. You might need to close and reopen your web browser to see the message.
25. Return to the `workstation` machine as the `student` user.

```
26. [root@servera ~]# exit
27. logout
28. [student@servera ~]$ exit
29. logout
30. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish selinux-booleans
```

This concludes the section.

Investigate and Resolve SELinux Issues

Objectives

Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.

Troubleshoot SELinux Issues

When applications unexpectedly fail to work due to SELinux access denials, methods and tools are available to resolve these issues. It is helpful to start by understanding some fundamental concepts and behaviors when SELinux is enabled.

- SELinux consists of targeted policies that explicitly define allowable actions.
- A policy entry defines a labeled process and a labeled resource that interact.
- The policy states the process type, and the file or port context, by using labels.
- The policy entry defines one process type, one resource label, and the explicit action to allow.
- An action can be a system call, a kernel function, or another specific programming routine.
- If no entry is created for a specific process-resource-action relationship, then the action is denied.
- When an action is denied, the attempt is logged with useful context information.

Red Hat Enterprise Linux provides a stable targeted SELinux policy for almost every service in the distribution. Therefore, it is unusual to have SELinux access problems with common RHEL services when they are configured correctly. SELinux access problems occur when services are implemented incorrectly, or when new applications have incomplete policies. Consider these troubleshooting concepts before making broad SELinux configuration changes.

- Most access denials indicate that SELinux is working correctly by blocking improper actions.
- Evaluating denied actions requires some familiarity with normal, expected service actions.

- The most common SELinux issue is an incorrect context on new, copied, or moved files.
- File contexts can be fixed when an existing policy references their location.
- Optional Boolean policy features are documented in the `_selinux` man pages.
- Implementing Boolean features generally requires setting additional non-SELinux configuration.
- SELinux policies do not replace or circumvent file permissions or access control list restrictions.

When a common application or service fails, and the service is known to have a working SELinux policy, first see the service's `_selinux` man page to verify the correct context type label. View the affected process and file attributes to verify that the correct labels are set.

Monitor SELinux Violations

The SELinux troubleshooting service, from the `setroubleshoot-server` package, provides tools to diagnose SELinux issues. When SELinux denies an action, an Access Vector Cache (AVC) message is logged to the `/var/log/audit/audit.log` security log file. The SELinux troubleshooting service monitors for AVC events and sends an event summary to the `/var/log/messages` file.

The AVC summary includes an event unique identifier (UUID). Use the `sealert -l UUID` command to view comprehensive report details for the specific event. Use the `sealert -a /var/log/audit/audit.log` command to view all existing events.

Consider the following example sequence of commands on a standard Apache web server. You create `/root/mypage` and move it to the default Apache content directory (`/var/www/html`). Then, after starting the Apache service, you try to retrieve the file content.

```
[root@host ~]# touch /root/mypage
[root@host ~]# mv /root/mypage /var/www/html
[root@host ~]# systemctl start httpd
[root@host ~]# curl http://localhost/mypage
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
```

```
</head><body>

<h1>Forbidden</h1>

<p>You don't have permission to access this resource.</p>

</body></html>
```

The web server does not display the content, and returns a `permission denied` error. An AVC event is logged to the `/var/log/audit/audit.log` and `/var/log/messages` files. Note the suggested `sealert` command and UUID in the `/var/log/messages` event message.

```
[root@host ~]# tail /var/log/audit/audit.log

...output omitted...

type=AVC msg=audit(1649249057.067:212): avc:  denied  { getattr } for  pid=2332 comm=
"httpd" path="/var/www/html/mypage" dev="vda4" ino=9322502 scontext=system_u:system_r
:httpd_t:s0 tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

...output omitted

[root@host ~]# tail /var/log/messages

...output omitted...

Apr  6 08:44:19 host setroubleshoot[2547]: SELinux is preventing /usr/sbin/httpd from
getattr access on the file /var/www/html/mypage. For complete SELinux messages run: s
ealert -l 95f41f98-6b56-45bc-95da-ce67ec9a9ab7

...output omitted...
```

The `sealert` output describes the event, and includes the affected process, the accessed file, and the attempted and denied action. The output includes advice for correcting the file's label, if appropriate. Additional advice describes how to generate a new policy to allow the denied action. Use the given advice only when it is appropriate for your scenario.

Important

The `sealert` output includes a confidence rating, which indicates the level of confidence that the given advice will mitigate the denial. However, that advice might not be appropriate for your scenario.

For example, if the AVC denial is because the denied file is in the wrong location, then advice that states either to adjust the file's context label, or to create a policy for this location and action, although technically accurate, is not the correct solution for your scenario. If the root cause is a wrong location or file name, then moving or renaming the file and then restoring a correct file context is the correct solution instead.

```
[root@host ~]# sealert -l 95f41f98-6b56-45bc-95da-ce67ec9a9ab7
SELinux is preventing /usr/sbin/httpd from getattr access on the file /var/www/html/m
ypage.


*****  Plugin restorecon (99.5 confidence) suggests   ************************


If you want to fix the label.
/var/www/html/mypage default label should be httpd_sys_content_t.
Then you can run restorecon. The access attempt may have been stopped due to insuffic
ient permissions to access a parent directory in which case try to change the followi
ng command accordingly.
Do
# /sbin/restorecon -v /var/www/html/mypage


*****  Plugin catchall (1.49 confidence) suggests   **************************


If you believe that httpd should be allowed getattr access on the mypage file by defa
ult.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp



Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:admin_home_t:s0
Target Objects                /var/www/html/mypage [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
```

```
...output omitted...


Raw Audit Messages

type=AVC msg=audit(1649249057.67:212): avc:  denied  { getattr } for  pid=2332 comm="
httpd" path="/var/www/html/mypage" dev="vda4" ino=9322502 scontext=system_u:system_r:
httpd_t:s0 tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0


type=SYSCALL msg=audit(1649249057.67:212): arch=x86_64 syscall=newfstatat success=no
exit=EACCES a0=ffffff9c a1=7fe9c00048f8 a2=7fe9ccfc8830 a3=100 items=0 ppid=2329 pid=
2332 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48
tty=(none) ses=4294967295 comm=httpd exe=/usr/sbin/httpd subj=system_u:system_r:httpd
_t:s0 key=(null)


Hash: httpd,httpd_t,admin_home_t,file,getattr
```

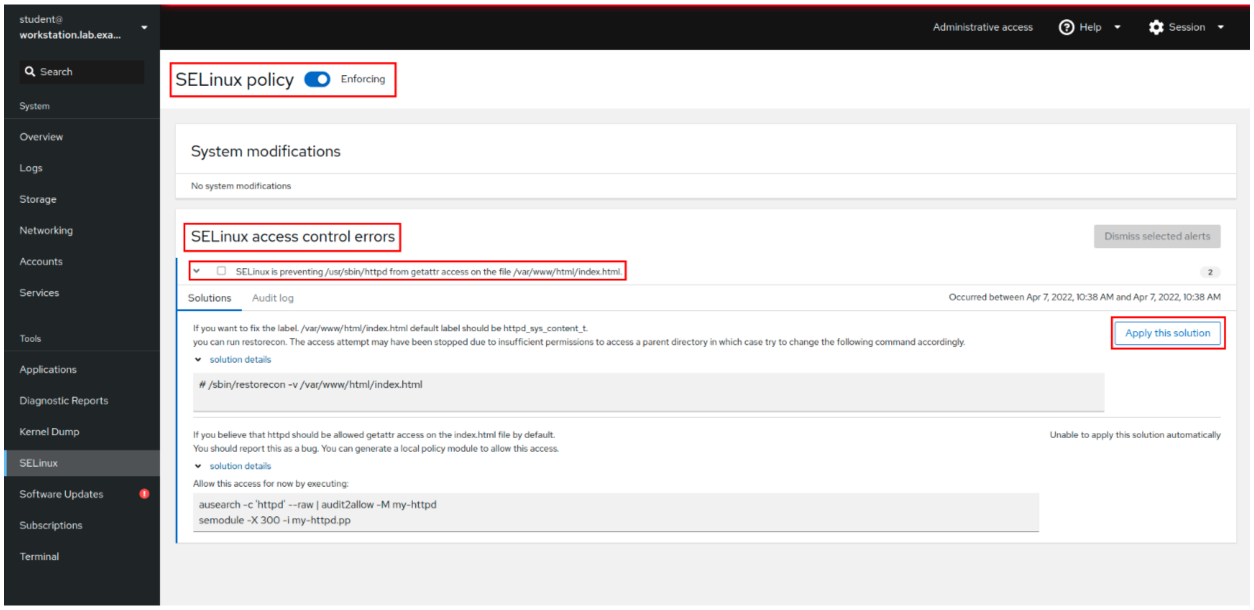
In this example, the accessed file is in the correct location, but does not have the correct SELinux file context. The `Raw Audit Messages` section displays information from the `/var/log/audit/audit.log` event entry. Use the `restorecon /var/www/html/mypage` command to set the correct context label. To correct multiple files recursively, use the `restorecon -R` command on the parent directory.

Use the `ausearch` command to search for AVC events in the `/var/log/audit/audit.log` log file. Use the `-m` option to specify the `AVC` message type and the `-ts` option to provide a time hint, such as `recent`.

```
[root@host ~]# ausearch -m AVC -ts recent
----
time->Tue Apr  6 13:13:07 2019
type=PROCTITLE msg=audit(1554808387.778:4002): proctitle=2F7573722F7362696E2F68747470
64002D44464F524547524F554E44
type=SYSCALL msg=audit(1554808387.778:4002): arch=c000003e syscall=49 success=no exit
=-13 a0=3 a1=55620b8c9280 a2=10 a3=7ffed967661c items=0 ppid=1 pid=9340 auid=42949672
95 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295
comm="httpd" exe="/usr/sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1554808387.778:4002): avc:  denied  { name_bind } for  pid=9340 co
mm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:re
served_port_t:s0 tclass=tcp_socket permissive=0
```

Troubleshoot SELinux Issues with the Web Console

The RHEL web console includes tools for troubleshooting SELinux issues. Select **SELinux** from the menu on the left. The SELinux policy window displays the current enforcing state. The **SELinux access control errors** section lists current SELinux issues.



System modifications

No system modifications

SELinux access control errors

SELinux is preventing /usr/sbin/httpd from getattr access on the file /var/www/html/index.html. 2

Occurred between Apr 7, 2022, 10:38 AM and Apr 7, 2022, 10:38 AM

If you want to fix the label. /var/www/html/index.html default label should be httpd_sys_content_t.
you can run restorecon. The access attempt may have been stopped due to insufficient permissions to access a parent directory in which case try to change the following command accordingly.

```
# /sbin/restorecon -v /var/www/html/index.html
```

If you believe that httpd should be allowed getattr access on the index.html file by default.
You should report this as a bug. You can generate a local policy module to allow this access.

Unable to apply this solution automatically

Allow this access for now by executing:

```
ausearch -c 'httpd' --raw | audit2allow -M my-httpd
semodule -X 300 -i my-httpd.pp
```

Figure 6.3: SELinux policy and errors in the web console

Click the **>** character to display event details. Click **solution details** to display all event details and advice. You can click **Apply the solution**.

After correcting the issue, the **SELinux access control errors** section should remove that event from view. If the `No SELinux alerts` message appears, then you have corrected all current SELinux issues.

Guided Exercise: Investigate and Resolve SELinux Issues

In this lab, you learn how to troubleshoot SELinux security denials.

Outcomes

- Gain experience with SELinux troubleshooting tools.

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

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

```
[student@workstation ~]$ lab start selinux-issues
```

Instructions

1. From a web browser on the `workstation` machine, open the `http://servera/index.html` web page. An error message states that you do not have permission to access the file.
2. Use the `ssh` command to log in to `servera` as the `student` user. Use the `sudo -i` command to switch to the `root` user.

```
3. [student@workstation ~]$ ssh student@servera
4. ...output omitted...
5. [student@servera ~]$ sudo -i
6. [sudo] password for student: student
```

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

7. Use the `less` command to view the contents of the `/var/log/messages` file. You use the **/** character and search for the `sealert` text. Press the **n** key until you reach the last occurrence, because previous exercises might also have generated SELinux messages. Copy the suggested `sealert` command so that you can use it in the next step. Use the **q** key to quit the `less` command.

```
8. [root@servera ~]# less /var/log/messages
9. ...output omitted...
10. Apr  7 04:52:18 servera setroubleshoot[20715]: SELinux is preventing /usr/sbin/httpd from getattr access on the file /custom/index.html. For complete SELinux messages run: sealert -l 9a96294a-239b-4568-8f1e-9f35b5fb472b
```

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

11. Run the suggested `sealert` command. Note the source context, the target objects, the policy, and the enforcing mode. Find the correct SELinux context label for the file that the `httpd` service tries to serve.
    1. Run the `sealert` command.

       The output explains that the `/custom/index.html` file has an incorrect context label.

       ```
       [root@servera ~]# sealert -l 9a96294a-239b-4568-8f1e-9f35b5fb472b
       ```

```
SELinux is preventing /usr/sbin/httpd from getattr access on the file /c
ustom/index.html.


*****  Plugin catchall_labels (83.8 confidence) suggests   *************
******


If you want to allow httpd to have getattr access on the index.html file
Then you need to change the label on /custom/index.html
Do
# semanage fcontext -a -t FILE_TYPE '/custom/index.html'
where FILE_TYPE is one of the following: NetworkManager_exec_t, NetworkM
anager_log_t, NetworkManager_tmp_t, abrt_dump_oops_exec_t, abrt_etc_t, a
brt_exec_t, abrt_handle_event_exec_t, abrt_helper_exec_t, abrt_retrace_c
oredump_exec_t, abrt_retrace_spool_t, abrt_retrace_worker_exec_t, abrt_t
mp_t, abrt_upload_watch_tmp_t, abrt_var_cache_t, abrt_var_log_t, abrt_va
r_run_t, accountsd_exec_t, acct_data_t, acct_exec_t, admin_crontab_tmp_t
, admin_passwd_exec_t, afs_logfile_t, aide_exec_t, aide_log_t, alsa_exec
_t, alsa_tmp_t, amanda_exec_t, amanda_log_t, amanda_recover_exec_t, aman
da_tmp_t, amtu_exec_t, anacron_exec_t, anon_inodefs_t
...output omitted...


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:default_t:s0
Target Objects                /custom/index.html [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          servera.lab.example.com
Source RPM Packages           httpd-2.4.51-7.el9_0.x86_64
Target RPM Packages
SELinux Policy RPM            selinux-policy-targeted-34.1.27-1.el9.noar
ch
Local Policy RPM              selinux-policy-targeted-34.1.27-1.el9.noar
ch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
```

```
Host Name                      servera.lab.example.com
Platform                       Linux servera.lab.example.com
                               5.14.0-70.2.1.el9_0.x86_64 #1 SMP PREEMPT
Wed Mar
                               16 18:15:38 EDT 2022 x86_64 x86_64
Alert Count                    4
First Seen                     2022-04-07 04:51:38 EDT
Last Seen                      2022-04-07 04:52:13 EDT
Local ID                       9a96294a-239b-4568-8f1e-9f35b5fb472b


Raw Audit Messages
type=AVC msg=audit(1649321533.406:1024): avc:  denied  { getattr } for
pid=20464 comm="httpd" path="/custom/index.html" dev="vda4" ino=25571802
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:def
ault_t:s0 tclass=file permissive=0


...output omitted...
```

2. Verify the SELinux context for the directory from where the `httpd` service serves the content by default, `/var/www/html`. The `httpd_sys_content_t` SELinux context is appropriate for the `/custom/index.html` file.

```
[root@servera ~]# ls -ldZ /var/www/html
```

```
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 6 Mar 2
1 11:47 /var/www/html
```

12. The `Raw Audit Messages` section of the `sealert` command contains information from the `/var/log/audit/audit.log` file. Use the `ausearch` command to search the `/var/log/audit/audit.log` file. The `-m` option searches on the message type. The `-ts` option searches based on time. The following entry identifies the relevant process and file that cause the alert. The process is the `httpd` Apache web server, the file is `/custom/index.html`, and the context is `system_r:httpd_t`.

```
[root@servera ~]# ausearch -m AVC -ts today
...output omitted...
----
time->Thu Apr  7 04:52:13 2022
```

```
17. type=PROCTITLE msg=audit(1649321533.406:1024): proctitle=2F7573722F7362696E2F6
    874747064002D44464F524547524F554E44

18. type=SYSCALL msg=audit(1649321533.406:1024): arch=c000003e syscall=262 success
    =no exit=-13 a0=ffffff9c a1=7fefc403d850 a2=7fefc89bc830 a3=100 items=0 ppid=2
    0461 pid=20464 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48
    sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
```

```
type=AVC msg=audit(1649321533.406:1024): avc:  denied  { getattr } for  pid=20
464 comm="httpd" path="/custom/index.html" dev="vda4" ino=25571802 scontext=sy
stem_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0 tclass=
file permissive=0
```

19. Resolve the issue by applying the `httpd_sys_content_t` context.

```
20. [root@servera ~]# semanage fcontext -a \

21. -t httpd_sys_content_t '/custom(/.*)?'

22. [root@servera ~]# restorecon -Rv /custom

23. Relabeled /custom from unconfined_u:object_r:default_t:s0 to unconfined_u:obje
    ct_r:httpd_sys_content_t:s0
```

```
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to unconf
ined_u:object_r:httpd_sys_content_t:s0
```

24. Again, try to view `http://servera/index.html`. The `This is SERVERA.` message is displayed.
25. Return to the `workstation` machine as the `student` user.

```
26. [root@servera ~]# exit

27. logout

28. [student@servera ~]$ exit

29. logout

30. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish selinux-issues
```

This concludes the section.

Summary

- Use the `getenforce` and `setenforce` commands to manage the SELinux mode of a system.
- The `semanage` command manages SELinux policy rules.
  The `restorecon` command applies the context that the policy defines.
- Booleans are switches that change the behavior of the SELinux policy. You can enable or disable them to tune the policy.
- The `sealert` command displays useful information to help with SELinux troubleshooting.

Chapter 7. Manage Basic Storage

Abstract

Goal	Create and manage storage devices, partitions, file systems, and swap spaces from the command line.
Objectives	• Create storage partitions, format them with file systems, and mount them for use. • Create and manage swap spaces to supplement physical memory.
Sections	• Add Partitions, File Systems, and Persistent Mounts (and Guided Exercise) • Manage Swap Space (and Guided Exercise)
Lab	Manage Basic Storage

Add Partitions, File Systems, and Persistent Mounts

Objectives

Create storage partitions, format them with file systems, and mount them for use.

Partition Disks

Disk partitioning divides a hard drive into multiple logical storage *partitions*. You can use partitions to divide storage based on different requirements, and this division provides many benefits:

- Limit available space to applications or users.
- Separate operating system and program files from user files.
- Create a separate area for memory swapping.

- Limit disk space use to improve the performance of diagnostic tools and backup imaging.

MBR Partition Scheme

The *Master Boot Record* (MBR) partitioning scheme is the standard on systems that run BIOS firmware. This scheme supports a maximum of four primary partitions. On Linux systems, with extended and logical partitions, you can create up to 15 partitions. With a 32-bit partition size, disks that are partitioned with MBR can have a size of up to 2 TiB.

Figure 7.1: MBR partitioning of the /dev/vdb storage device

The 2 TiB disk and partition size limit is now a common and restrictive limitation. Consequently, the legacy MBR scheme is superseded by the *GUID Partition Table (GPT)* partitioning scheme.

GPT Partition Scheme

For systems that run *Unified Extensible Firmware Interface (UEFI)* firmware, GPT is the standard for disk partitioning, and addresses the limitations of the MBR scheme. A GPT provides a maximum of 128 partitions. The GPT scheme allocates 64 bits for logical block addresses, to support partitions and disks of up to eight zebibytes (ZiB) or eight billion tebibytes (TiB).

Figure 7.2: GPT partitioning of the /dev/vdb storage device

GPT partitioning offers additional features and benefits over MBR. A GPT uses a *globally unique identifier* (GUID) to identify each disk and partition. A GPT makes the partition table redundant, with the primary GPT at the head of the disk, and a backup secondary GPT at the end of the disk. A GPT uses a checksum to detect errors in the GPT header and partition table.

Manage Partitions

An administrator can use a *partition editor* program to change a disk's partitions, such as creating and deleting partitions, and changing partition types.

The standard partition editor on the command line in Red Hat Enterprise Linux is `parted`. You can use the `parted` partition editor with storage that uses either the MBR partitioning scheme or the GPT partitioning scheme.

The `parted` command takes as its first argument the device name that represents the entire storage device or disk to modify, followed by subcommands. The following example uses the `print` subcommand to display the partition table on the disk that is the `/dev/vda` block device (the first "virtualized I/O" disk detected by the system).

```
[root@host ~]# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
 2      10.7GB  53.7GB  42.9GB  primary  xfs
```

Use the `parted` command without a subcommand to open an interactive partitioning session.

```
[root@host ~]# parted /dev/vda
```

```
GNU Parted 3.4
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
 2      10.7GB  53.7GB  42.9GB  primary  xfs

(parted) quit
[root@host ~]#
```

By default, the `parted` command displays sizes in powers of 10 (KB, MB, GB). You can change the unit size with the `unit` parameter, which accepts the following values:

- **`s`** for sector
- **`B`** for byte
- **`MiB`** , **`GiB`** , or **`TiB`** (powers of 2)
- **`MB`** , **`GB`** , or **`TB`** (powers of 10)

```
[root@host ~]# parted /dev/vda unit s print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start      End        Size       Type     File system  Flags
 1      2048s      20971486s  20969439s  primary  xfs          boot
 2      20971520s  104857535s  83886016s  primary  xfs
```

As shown in the previous example, you can also specify multiple subcommands (here, `unit` and `print`) on the same line.

Write the Partition Table on a New Disk

To partition a new drive, first write a disk label. The disk label indicates which partitioning scheme to use. Use `parted` to write an MBR disk label or a GPT disk label.

```
[root@host ~]# parted /dev/vdb mklabel msdos


[root@host ~]# parted /dev/vdb mklabel gpt
```

Warning

The `mklabel` subcommand wipes the existing partition table. Use the `mklabel` subcommand when the intent is to reuse the disk without regard to the existing data. If a new label moves the partition boundaries, then all data in existing file systems becomes inaccessible.

Create MBR Partitions

The following instructions create an MBR disk partition. Specify the disk device to create the partition on.

Run the `parted` command and specify the disk device name as an argument, to start in interactive mode. The session displays `(parted)` as a subcommand prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Use the `mkpart` subcommand to create a primary or extended partition.

```
(parted) mkpart
```

```
Partition type?  primary/extended? **primary**
```

Note

If you need more than four partitions on an MBR-partitioned disk, then create three primary partitions and one extended partition. The extended partition serves as a container within which you can create multiple logical partitions.

Indicate the file-system type to create on the partition, such as `xfs` or `ext4`. This value is only a useful partition type label, and does not create the file system.

```
File system type?  [ext2]? xfs
```

To list the supported file-system types, use the following command:

```
[root@host ~]# parted /dev/vdb help mkpart
  ...output omitted...
  mkpart PART-TYPE [FS-TYPE] START END     make a partition

    PART-TYPE is one of: primary, logical, extended
    FS-TYPE is one of: udf, btrfs, nilfs2, ext4, ext3, ext2, f2fs, fat32, fat16,
    hfsx, hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), ntfs,
    reiserfs, hp-ufs, sun-ufs, xfs, apfs2, apfs1, asfs, amufs5, amufs4, amufs3,
    amufs2, amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2,
    affs1, affs0, linux-swap, linux-swap(new), linux-swap(old)

   'mkpart' makes a partition without creating a new file system on the
    partition.  FS-TYPE may be specified to set an appropriate partition
    ID.
```

Specify the disk sector to start the new partition on.

```
Start? 2048s
```

The `s` suffix provides the value in sectors, or uses the `MiB`, `GiB`, `TiB`, `MB`, `GB`, or `TB` suffixes. The `parted` command defaults to the `MB` suffix. The `parted` command rounds provided values to satisfy disk constraints.

When the `parted` command starts, it retrieves the disk topology from the device, such as the disk physical block size. The `parted` command ensures that the start position that you provide correctly aligns the partition with the disk structure, to optimize performance. If the start position results in a misaligned partition, then the `parted` command displays a warning. With most disks, a start sector that is a multiple of 2048 is safe.

Specify the disk sector where the new partition should end, and exit `parted`. You can specify the end as a size or as an ending location.

```
End? 1000MB
(parted) quit
Information: You may need to update /etc/fstab.


[root@host ~]#
```

When you provide the end position, the `parted` command updates the partition table on the disk with the new partition details.

Run the `udevadm settle` command. This command waits for the system to detect the new partition and to create the associated device file in the `/dev` directory. The prompt returns when the task is done.

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

As an alternative to interactive mode, you can create a partition in a single command:

```
[root@host ~]# parted /dev/vdb mkpart primary xfs 2048s 1000MB
```

Create GPT Partitions

The GPT scheme also uses the `parted` command to create partitions. Specify the disk device to create the partition on.

As the `root` user, execute the `parted` command and specify the disk device name as an argument.

```
[root@host ~]# parted /dev/vdb
```

```
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Use the `mkpart` subcommand to begin creating the partition. With the GPT scheme, each partition is given a name.

```
(parted) mkpart
Partition name?  []? userdata
```

Indicate the file-system type to create on the partition, such as `xfs` or `ext4`. This value does not create the file system, but is a useful partition type label.

```
File system type?  [ext2]? xfs
```

Specify the disk sector that the new partition starts on.

```
Start? 2048s
```

Specify the disk sector for the new partition to end, and exit `parted`. When you provide the end position, the `parted` command updates the GPT on the disk with the new partition details.

```
End? 1000MB
(parted) quit
Information: You may need to update /etc/fstab.


[root@host ~]#
```

Run the `udevadm settle` command. This command waits for the system to detect the new partition and to create the associated device file in the `/dev` directory. The prompt returns when the task is done.

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

As an alternative to interactive mode, you can create a partition in a single command:

```
[root@host ~]# parted /dev/vdb mkpart userdata xfs 2048s 1000MB
```

Delete Partitions

The following instructions apply for both the MBR and GPT partitioning schemes. Specify the disk that contains the partition to remove.

Run the `parted` command with the disk device as the only argument.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Identify the partition number of the partition to delete.

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name       Flags
 1      1049kB  1000MB  999MB  xfs          usersdata
```

Delete the partition, and exit `parted`. The `rm` subcommand immediately deletes the partition from the partition table on the disk.

```
(parted) rm 1
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

As an alternative to interactive mode, you can delete a partition in a single command:

```
[root@host ~]# parted /dev/vdb rm 1
```

Create File Systems

After a block device is created, the next step is to add a file system to it. Red Hat Enterprise Linux supports multiple file-system types, and XFS is the recommended default.

As the `root` user, use the `mkfs.xfs` command to apply an XFS file system to a block device. For an ext4 file system, use the `mkfs.ext4` command.

```
[root@host ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=60992 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=243968, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Mount File Systems

After you add the file system, the last step is to mount the file system to a directory in the directory structure. When you mount a file system to the directory hierarchy, user-space utilities can access or write files on the device.

Manually Mount File Systems

Use the `mount` command to manually attach a device to a *mount point* directory location. The `mount` command requires a device and a mount point, and can include file-system mount options. File-system options customize the behavior of the file system.

```
[root@host ~]# mount /dev/vdb1 /mnt
```

You also use the `mount` command to view currently mounted file systems, the mount points, and their options.

```
[root@host ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Persistently Mount File Systems

Manually mounting a file system is a good way to verify that a formatted device is accessible and is working as expected. However, when the server reboots, the system does not automatically mount the file system again.

To configure the system to automatically mount the file system during system boot, add an entry to the `/etc/fstab` file. This configuration file lists the file systems to mount at system boot.

The `/etc/fstab` file is a white-space-delimited file with six fields per line.

```
[root@host ~]# cat /etc/fstab


#
# /etc/fstab
# Created by anaconda on Thu Apr 5 12:05:19 2022
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=a8063676-44dd-409a-b584-68be2c9f5570   /        xfs   defaults   0 0
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838   /dbdata  xfs   defaults   0 0
```

The first field specifies the device. This example uses a UUID to specify the device. File systems create and store the UUID in the partition super block at creation time. Alternatively, you could use the device file, such as `/dev/vdb1`.

The second field is the directory mount point, from which the block device is accessible in the directory structure. The mount point must exist; if not, create it with the `mkdir` command.

The third field contains the file-system type, such as `xfs` or `ext4`.

The fourth field is the comma-separated list of options to apply to the device. `defaults` is a set of commonly used options. The `mount`(8) man page documents the other available options.

The fifth field is used by the `dump` command to back up the device. Other backup applications do not usually use this field.

The last field, the `fsck` order field, determines whether to run the `fsck` command at system boot to verify that the file systems are clean. The value in this field indicates the order in which `fsck` should run. For XFS file systems, set this field to `0`, because XFS does not use `fsck` to verify its file-system status. For ext4 file systems, set it to `1` for the root file system, and to `2` for the other ext4 file systems. By using this notation, the `fsck` utility processes the root file system first, and then verifies file systems on separate disks concurrently, and file systems on the same disk in sequence.

Note

An incorrect entry in `/etc/fstab` might render the machine non-bootable. Verify that an entry is valid by manually unmounting the new file system and then by using `mount /*mountpoint*` to read the `/etc/fstab` file, and remount the file system with that entry's mount options. If the `mount` command returns an error, then correct it before rebooting the machine.

Alternatively, use the `findmnt --verify` command to parse the `/etc/fstab` file for partition usability.

When you add or remove an entry in the `/etc/fstab` file, run the `systemctl daemon-reload` command, or reboot the server, to ensure that the `systemd` daemon loads and uses the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

Red Hat recommends the use of UUIDs to persistently mount file systems, because block device names can change in certain scenarios, such as if a cloud provider changes the underlying storage layer of a virtual machine, or if disks are detected in a different order on a system boot. The block device file name might change, but the UUID remains constant in the file-system's super block.

Use the `lsblk --fs` command to scan the block devices that are connected to a machine and retrieve the file-system UUIDs.

```
[root@host ~]# lsblk --fs
NAME   FSTYPE  FSVER  LABEL    UUID            FSAVAIL FSUSE% MOUNTPOINTS
vda
├─vda1
├─vda2 xfs            boot     49dd...75fdf    312M    37%    /boot
└─vda3 xfs            root     8a90...ce0da    4.8G    48%    /
```

Guided Exercise: Add Partitions, File Systems, and Persistent Mounts

In this exercise, you create a partition on a new storage device, format it with an XFS file system, configure it to mount at boot, and mount it for use.

Outcomes

- Use the `parted`, `mkfs.xfs`, and other commands to create a partition on a new disk, format it, and persistently mount it.

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

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

```
[student@workstation ~]$ lab start storage-partitions
```

Instructions

1. Log in to `servera` as the `student` user and switch to the `root` user.

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

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

6. Create an `msdos` disk label on the `/dev/vdb` device.

```
[root@servera ~]# parted /dev/vdb mklabel msdos
```

```
Information: You may need to update /etc/fstab.
```

8. Add a 1 GB primary partition. For correct alignment, start the partition at the 2048 sector. Set the partition file-system type to XFS.
   1. Use `parted` interactive mode to create the partition.

   ```
   [root@servera ~]# parted /dev/vdb
   GNU Parted 3.4
   Using /dev/vdb
   Welcome to GNU Parted! Type 'help' to view a list of commands.
   (parted) mkpart
   Partition type?  primary/extended? primary
   File system type?  [ext2]? xfs
   Start? 2048s
   End? 1001MB
   (parted) quit
   ```

   ```
   Information: You may need to update /etc/fstab.
   ```

   Because the partition starts at the 2048 sector, the previous command sets the end position to 1001 MB to get a partition size of 1000 MB (1 GB).

Alternatively, you can perform the same operation with the following non-interactive command: `parted /dev/vdb mkpart primary xfs 2048s 1001 MB`

12. Verify your work by listing the partitions on the `/dev/vdb` device.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
```

```
 1      1049kB  1001MB  1000MB  primary
```

21. Run the `udevadm settle` command. This command waits for the system to register the new partition, and returns when it is done.

```
[root@servera ~]# udevadm settle
```

9. Format the new partition with the XFS file system.

```
[root@servera ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=61056 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=244224, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
```

```
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

20. Configure the new file system to mount to the `/archive` directory persistently.

1. Create the `/archive` directory.

```
[root@servera ~]# mkdir /archive
```

2. Discover the UUID of the `/dev/vdb1` device. The UUID in the output is probably different on your system.

```
3. [root@servera ~]# lsblk --fs /dev/vdb

4. NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL F
   SUSE% MOUNTPOINTS

5. vdb
```

```
└─vdb1 xfs                881e856c-37b1-41e3-b009-ad526e46d987
```

6. Add an entry to the `/etc/fstab` file. Replace the UUID with the one that you discovered from the previous step.

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

```
UUID=881e856c-37b1-41e3-b009-ad526e46d987 /archive xfs defaults  0 0
```

8. Update the `systemd` daemon for the system to register the new `/etc/fstab` file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

9. Mount the new file system with the new entry in the `/etc/fstab` file.

```
[root@servera ~]# mount /archive
```

10. Verify that the new file system is mounted on the `/archive` directory.

```
11. [root@servera ~]# mount | grep /archive
```

```
/dev/vdb1 on /archive type xfs (rw,relatime,seclabel,attr2,inode64,logbu
fs=8,logbsize=32k,noquota)
```

21. Reboot `servera`. After the server rebooted, log in and verify that the `/dev/vdb1` device is mounted on the `/archive` directory. When done, log out from `servera`.
    1. Reboot `servera`.

```
2. [root@servera ~]# systemctl reboot
3. Connection to servera closed by remote host.
4. Connection to servera closed.
```

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

5. Wait for `servera` to reboot and log in as the `student` user.

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

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

8. Verify that the `/dev/vdb1` device is mounted on the `/archive` directory.

```
9. [student@servera ~]$ mount | grep /archive
```

```
/dev/vdb1 on /archive type xfs (rw,relatime,seclabel,attr2,inode64,logbu
fs=8,logbsize=32k,noquota)
```

10. Return to the `workstation` machine as the `student` user.

```
11. [student@servera ~]$ exit
12. logout
13. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-partitions
```

This concludes the section.

Manage Swap Space

Objectives

Create and manage swap spaces to supplement physical memory.

Swap Space Concepts

A *swap space* is an area of a disk under the control of the Linux kernel memory management subsystem. The kernel uses swap space to supplement the system RAM by holding inactive pages in memory. A system's *virtual memory* encompasses the combined system RAM and swap space.

When the memory usage on a system exceeds a defined limit, the kernel searches through RAM to look for idle memory pages that are assigned to processes. The kernel writes the idle pages to the swap area and reassigns the RAM pages to other processes. If a program requires access to a page on disk, then the kernel locates another idle page of memory, writes it to disk, and recalls the needed page from the swap area.

Because swap areas are on disk, swap is slow when compared with RAM. Although swap space augments system RAM, do not consider swap space as a sustainable solution for insufficient RAM for your workload.

Swap Space Calculation

Administrators should size the swap space based on the memory workload on the system. Application vendors sometimes provide recommendations for calculating swap space. The following table provides guidance based on the total physical memory.

Table 7.1. RAM and Swap Space Recommendations

RAM	Swap space	Swap space if allowing for hibernation
2 GB or less	Twice the RAM	Three times the RAM
Between 2 GB and 8 GB	Same as RAM	Twice the RAM
Between 8 GB and 64 GB	At least 4 GB	1.5 times the RAM
More than 64 GB	At least 4 GB	Hibernation is not recommended

The laptop and desktop hibernation function uses the swap space to save the RAM contents before powering off the system. When you turn the system back on, the kernel restores the RAM contents from the swap space and does not need a complete boot. For those systems, the swap space must be greater than the amount of RAM.

The Knowledgebase article in References at the end of this section gives more guidance about sizing the swap space.

Create Swap Space

To create a swap space, perform the following steps:

- Create a partition with a file-system type of `linux-swap`.
- Place a swap signature on the device.

Create a Swap Partition

Use the `parted` command to create a partition of the appropriate size and set its file-system type to `linux-swap`. In the past, tools determined from the partition file-system type whether to activate the device; however, that requirement is no longer the case. Even though utilities no longer use the partition file-system type, administrators can determine the partition's purpose from that type.

The following example creates a 256 MB partition.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
```

```
 1      1049kB  1001MB  1000MB               data

(parted) mkpart
Partition name?  []? swap1
File system type?  [ext2]? linux-swap
Start? 1001MB
End? 1257MB
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name   Flags
 1      1049kB  1001MB  1000MB                  data
 2      1001MB  1257MB  256MB   linux-swap(v1)  swap1

(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

After creating the partition, run the `udevadm settle` command. This command waits for the system to detect the new partition and to create the associated device file in the `/dev` directory. The command returns only when it is finished.

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

Format Swap Space

The `mkswap` command applies a swap signature to the device. Unlike other formatting utilities, the `mkswap` command writes a single block of data at the beginning of the device, and leaves the rest of the device unformatted so that the kernel can use it for storing memory pages.

```
[root@host ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

Activate Swap Space

You can use the `swapon` command to activate a formatted swap space.

Use `swapon` with the device as a parameter, or use `swapon -a` to activate all the listed swap spaces in the `/etc/fstab` file. Use the `swapon --show` and `free` commands to inspect the available swap spaces.

```
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      134688     1536436       16748      201912     1576044
Swap:             0           0           0
[root@host ~]# swapon /dev/vdb2
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      135044     1536040       16748      201952     1575680
Swap:        249852           0      249852
```

You can deactivate a swap space with the `swapoff` command. If pages are written to the swap space, then the `swapoff` command tries to move those pages to other active swap spaces or back into memory. If the `swapoff` command cannot write data to other places, then it fails with an error, and the swap space stays active.

Activate Swap Space Persistently

Create an entry in the `/etc/fstab` file to ensure an active swap space at system boot. The following example shows a typical line in the `/etc/fstab` file based on the previously created swap space.

```
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   defaults   0 0
```

The example uses the UUID as the first field. When you format the device, the `mkswap` command displays that UUID. If you lost the output of `mkswap`, then use the `lsblk --fs` command. As an alternative, you can use the device name in the first field.

The second field is typically reserved for the mount point. However, for swap devices, which are not accessible through the directory structure, this field takes the `swap` placeholder value. The `fstab(5)` man page uses a `none` placeholder value; however, a `swap` value gives more informative error messages if something goes wrong.

The third field is the file-system type. The file-system type for swap space is `swap`.

The fourth field is for options. The example uses the `defaults` option.
The `defaults` option includes the `auto` mount option, which activates the swap space automatically at system boot.

The final two fields are the `dump` flag and the `fsck` order. Swap spaces do not require backing up or file-system checking, and so these fields should be set to zero.

When you add or remove an entry in the `/etc/fstab` file, run the `systemctl daemon-reload` command, or reboot the server, for `systemd` to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

Set Swap Space Priority

By default, the system uses swap spaces in series, meaning that the kernel uses the first activated swap space until it is full, and then it starts using the second swap space. You can instead define a priority for each swap space to force a particular order.

To set the priority, use the `pri` option in the `/etc/fstab` file. The kernel uses the swap space with the highest priority first. The default priority is -2.

The following example shows three defined swap spaces in the `/etc/fstab` file. The kernel uses the last entry first, because its priority is set to 10. When that space is full, it uses the second entry, because its priority is set to 4. Finally, it uses the first entry, which has a default priority of -2.

```
UUID=af30cbb0-3866-466a-825a-58889a49ef33   swap   swap   defaults  0 0
```

```
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   pri=4     0 0
UUID=fbd7fa60-b781-44a8-961b-37ac3ef572bf   swap   swap   pri=10    0 0
```

Use the `swapon --show` command to display the swap space priorities.

When swap spaces have the same priority, the kernel writes to them in a round-robin fashion.

Guided Exercise: Manage Swap Space

In this exercise, you create and format a partition as a swap space, and activate it persistently.

Outcomes

- Create a partition and a swap space on a disk by using the GPT partitioning scheme.

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

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

```
[student@workstation ~]$ lab start storage-swap
```

Instructions

1. Log in to `servera` as the `student` user and switch to the `root` user.

```
2. [student@workstation ~]$ ssh student@servera
3. ...output omitted...
4. [student@servera ~]$ sudo -i
5. [sudo] password for student: student
```

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

6. Inspect the `/dev/vdb` disk. The disk already has a partition table and uses the GPT partitioning scheme. Also, it has an existing 1 GB partition.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
```

```
 1      1049kB  1001MB  1000MB               data
```

15. Add a new partition of 500 MB for use as a swap space. Set the partition type to `linux-swap`.
    1. Create the `myswap` partition. Because the disk uses the GPT partitioning scheme, you must give a name to the partition. Notice that the start position, 1001 MB, is the end of the existing first partition.
       The `parted` command ensures that the new partition immediately follows the previous one, without any gap. Because the partition starts at the 1001 MB position, the command sets the end position to 1501 MB to get a partition size of 500 MB.

```
[root@servera ~]# parted /dev/vdb mkpart myswap linux-swap \
1001MB 1501MB
```

```
Information: You may need to update /etc/fstab.
```

    4. Verify your work by listing the partitions on the `/dev/vdb` disk. The size of the new partition is not exactly 500 MB. The difference in size is because the `parted` command must align the partition with the disk layout.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

```

```
12. Number  Start   End     Size    File system  Name    Flags
13.  1      1049kB  1001MB  1000MB               data
```

```
 2      1001MB  1501MB  499MB                myswap swap
```

14. Run the `udevadm settle` command. This command waits for the system to register the new partition, and returns when it is done.

```
[root@servera ~]# udevadm settle
```

16. Initialize the new partition as a swap space.

```
17. [root@servera ~]# mkswap /dev/vdb2
18. Setting up swapspace version 1, size = 476 MiB (499118080 bytes)
```

```
no label, UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328
```

19. Enable the new swap space.
    1. Verify that creating and initializing the swap space does not yet enable it for use.

    ```
    [root@servera ~]# swapon --show
    ```

    2. Enable the new swap space.

    ```
    [root@servera ~]# swapon /dev/vdb2
    ```

    3. Verify that the new swap space is now available.

    ```
    4. [root@servera ~]# swapon --show
    5. NAME      TYPE      SIZE USED PRIO
    ```

    ```
    /dev/vdb2 partition 476M   0B   -2
    ```

    6. Disable the swap space.

    ```
    [root@servera ~]# swapoff /dev/vdb2
    ```

    7. Confirm that the swap space is disabled.

```
[root@servera ~]# swapon --show
```

20. Enable the new swap space at system boot.
    1. Use the `lsblk` command with the `--fs` option to discover the UUID of the `/dev/vdb2` device. The UUID in the output is different on your system.

```
2. [root@servera ~]# lsblk --fs /dev/vdb2

3. NAME FSTYPE FSVER LABEL UUID                                 FSAVAIL F
   SUSE% MOUNTPOINTS
```

```
vdb2 swap   1           762735cb-a52a-4345-9ed0-e3a68aa8bb97
```

    4. Add an entry to the `/etc/fstab` file. In the following command, replace the UUID with the one that you discovered from the previous step.

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

```
UUID=762735cb-a52a-4345-9ed0-e3a68aa8bb97  swap  swap  defaults  0 0
```

    6. Update the `systemd` daemon for the system to register the new `/etc/fstab` file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

    7. Enable the swap space by using the entry in the `/etc/fstab` file.

```
[root@servera ~]# swapon -a
```

    8. Verify that the new swap space is enabled.

```
9. [root@servera ~]# swapon --show

10.NAME      TYPE      SIZE USED PRIO
```

```
/dev/vdb2 partition 476M   0B   -2
```

21. Reboot the `servera` machine. After the server reboots, log in and verify that the swap space is enabled. When done, log out from `servera`.
    1. Reboot the `servera` machine.

```
2. [root@servera ~]# systemctl reboot

3. Connection to servera closed by remote host.
```

```
Connection to servera closed.
```

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

5. Wait for `servera` to reboot and log in as the `student` user.

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

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

8. Verify that the swap space is enabled.

```
[student@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
```

```
/dev/vdb2 partition 476M   0B   -2
```

11. Return to the `workstation` machine as the `student` user.

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

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-swap
```

This concludes the section.

Summary

- The `parted` command adds, modifies, and removes partitions on disks with the MBR or the GPT partitioning scheme.
- The `mkfs.xfs` command creates XFS file systems on disk partitions.
- The `/etc/fstab` file contains devices that must be persistently mounted.
- The `mkswap` command initializes swap spaces.

Chapter 8. Manage Storage Stack

Abstract

Goal	Create and manage logical volumes that contain file systems or swap spaces from the command line.
Objectives	• Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information. • Analyze the multiple storage components that make up the layers of the storage stack.
Sections	• Create and Extend Logical Volumes (and Guided Exercise) • Manage Layered Storage (and Guided Exercise)
Lab	Manage Storage Stack

Create and Extend Logical Volumes

Objectives

Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information.

Logical Volume Manager Overview

Use the *Logical Volume Manager (LVM)* system to create logical storage volumes as a layer on the physical storage. This storage system provides greater flexibility than using physical storage directly. LVM hides the hardware storage configuration from the software, and enables you to resize volumes without stopping applications or unmounting file systems. LVM provides comprehensive command-line tools to manage storage.

Physical devices

Logical volumes use physical devices for storing data. These devices might be disk partitions, whole disks, RAID arrays, or SAN disks. You must initialize the device as an LVM physical volume. An LVM physical volume must use the entire physical device.

Physical Volumes (PVs)

LVM uses the underlying physical device as the LVM physical volume. LVM tools segment the physical volumes into *Physical Extents (PEs)* to form small chunks of data that act as the smallest storage block on a PV.

Volume Groups (VGs)

Volume groups are storage pools that are made from one or more PVs. It is the functional equivalent of a whole disk in physical storage. A PV must be allocated only to a single VG. LVM sets the PE size automatically, although it is possible to specify it. A VG might consist of unused space and several logical volumes.

Logical Volumes (LVs)

Logical volumes are created from free physical extents in a VG, and are provided as the storage device for applications, users, and operating systems. LVs are a collection of *Logical Extents (LEs)*, which map to physical extents. By default, each LE gets mapped to one PE. Setting specific LV options changes this mapping; for example, mirroring causes each LE to map to two PEs.

Logical Volume Manager Workflow

Creating LVM storage requires building structures in a logical workflow.

- Determine the physical devices that are used for creating physical volumes, and initialize these devices as LVM physical volumes.
- Create a volume group from multiple physical volumes.
- Create the logical volumes from the available space in the volume group.

- Format the logical volume with a file system and mount it, or activate it as swap space, or pass the raw volume to a database or storage server for advanced structures.

Figure 8.1: Logical Volume Manager workflow

Note

The examples here use a `/dev/vdb` device name and its storage partitions. The device names on your classroom system might be different. Use the `lsblk`, `blkid`, or `cat /proc/partitions` commands to identify your system's devices.

Build LVM Storage

Creating a logical volume involves creating physical device partitions, physical volumes, and volume groups. After creating an LV, format the volume and mount it to access it as storage.

Prepare Physical Devices

Partitioning is optional when already present. Use the `parted` command to create a partition on the physical device. Set the physical device to the `Linux LVM` partition type. Use the `udevadm settle` command to register the new partition with the kernel.

```
[root@host ~]# parted /dev/vdb mklabel gpt mkpart primary 1MiB 769MiB
...output omitted...
[root@host ~]# parted /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted /dev/vdb set 1 lvm on
[root@host ~]# parted /dev/vdb set 2 lvm on
[root@host ~]# udevadm settle
```

Create Physical Volumes

Use the `pvcreate` command to label the physical partition as an LVM physical volume. Label multiple devices simultaneously by using space-delimited device names as arguments to the `pvcreate` command. This example labels the `/dev/vdb1` and `/dev/vdb2` devices as PVs that are ready for creating volume groups.

```
[root@host ~]# pvcreate /dev/vdb1 /dev/vdb2
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Creating devices file /etc/lvm/devices/system.devices
```

Create a Volume Group

The `vgcreate` command builds one or more physical volumes into a volume group. The first argument is a volume group name, followed by one or more physical volumes to allocate to this VG. This example creates the `vg01` VG by using the `/dev/vdb1` and `/dev/vdb2` PVs.

```
[root@host ~]# vgcreate vg01 /dev/vdb1 /dev/vdb2
  Volume group "vg01" successfully created
```

Create a Logical Volume

The `lvcreate` command creates a logical volume from the available PEs in a volume group. Use the `lvcreate` command to set the LV name and size, and the VG name to contain this logical volume. This example creates the `lv01` LV with 300 MiB in the `vg01` VG.

```
[root@host ~]# lvcreate -n lv01 -L 300M vg01
  Logical volume "lv01" created.
```

This command might fail if the volume group does not have enough free physical extents. The LV size rounds up to the next PE size value when the size does not exactly match.

The `lvcreate` command `-L` option requires sizes in bytes, mebibytes (binary megabytes, 1048576 bytes), and gibibytes (binary gigabytes), or similar. The lowercase `-l` requires sizes that are specified as a number of physical extents. The following commands are two choices for creating the same LV with the same size:

- **`lvcreate -n lv01 -L 128M vg01`** : create an LV of size 128 MiB, rounded to the next PE.
- **`lvcreate -n lv01 -l 32 vg01`** : create an LV of size 32 PEs at 4 MiB each, total 128 MiB.

Create a Logical Volume with Deduplication and Compression

RHEL 9 uses an LVM VDO implementation for managing VDO volumes. The previous python-based VDO management tools are still available but are no longer needed.

The *Virtual Data Optimizer (VDO)* provides inline block-level deduplication, compression, and thin provisioning for storage. Configure a VDO volume to use up to 256 TB of physical storage. Manage VDO as a type of LVM logical volume (LVs), similar to LVM thinly provisioned volumes. An LVM VDO is composed of two logical volumes:

VDO pool LV

> This LV stores, deduplicates, compresses data, and sets the size of the VDO volume that is backed by the physical device. VDO is deduplicated and compresses each VDO LV separately, because each VDO pool LV can hold only one VDO LV.

VDO LV

> A virtual device is provisioned on top of the VDO pool LV, and sets the logical size of the VDO volume to store the data before deduplication and compression occur.

LVM VDO presents the deduplicated storage as a regular logical volume (LV). The VDO volume can be formatted with standard file systems, or shared as a block device, or used to build other storage layers, the same as any normal logical volume.

To use VDO deduplication and compression, install the `vdo` and `kmod-kvdo` packages.

```
[root@host ~]# dnf install vdo kmod-kvdo
```

Verify that the selected LVM volume group has enough free storage capacity. Use the `lvcreate` command with the `--type vdo` parameter to create a VDO LV.

```
[root@host ~]# lvcreate --type vdo --name vdo-lv01 --size 5G vg01
    Logical blocks defaulted to 523108 blocks.
    The VDO volume can address 2 GB in 1 data slab.
    It can grow to address at most 16 TB of physical storage in 8192 slabs.
    If a larger maximum size might be needed, use bigger slabs.
```

```
  Logical volume "vdo-lv01" created.
```

Create a File System on the Logical Volume

Specify the logical volume by using either the `/dev/vgname/lvname` traditional name, or the `/dev/mapper/vgname-lvname` kernel device mapper name.

Use the `mkfs` command to create a file system on the new logical volume.

```
[root@host ~]# mkfs -t xfs /dev/vg01/lv01
...output omitted...
```

Create a mount point by using the `mkdir` command.

```
[root@host ~]# mkdir /mnt/data
```

To make the file system available persistently, add an entry to the `/etc/fstab` file.

```
/dev/vg01/lv01 /mnt/data xfs defaults 0 0
```

Mount the LV by using the `mount` command.

```
[root@host ~]# mount /mnt/data/
```

Note

You can mount a logical volume by name or by UUID, because LVM parses the PVs by looking for the UUID. This behavior is successful even when the VG was created by using a name, because the PV always contains a UUID.

Display LVM Component Status

LVM provides various utilities to display the status information of PV, VG, and LV. Use the `pvdisplay`, `vgdisplay`, and `lvdisplay` commands to show the status information of the LVM components.

The associated `pvs`, `vgs`, and `lvs` commands are commonly used and show a subset of the status information, with one line for each entity.

Display Physical Volume Information

The `pvdisplay` command displays information about the PVs. Use the command without arguments to list information of all PVs on the system. Providing the name of the PV as the argument to the command shows the information that is specific to the PV.

```
[root@host ~]# pvdisplay /dev/vdb1
  --- Physical volume ---

  PV Name               /dev/vdb1

  VG Name               vg01

  PV Size               731.98 MiB / not usable 3.98 MiB
  Allocatable           yes

  PE Size               4.00 MiB
  Total PE              182

  Free PE               107
  Allocated PE          75
  PV UUID               zP0gD9-NxTV-Qtoi-yfQD-TGpL-0Yj0-wExh2N
```

`PV Name` shows the device name.

`VG Name` shows the volume group where the PV is allocated.

`PV Size` shows the physical size of the PV, including unusable space.

`PE Size` shows the physical extent size.

`Free PE` shows the PE size available in the VG to create new LVs or extend existing LVs.

Display Volume Group Information

The `vgdisplay` command shows the information about volume groups. To list information about all VGs, use the command without arguments. Provide the name of the VG as an argument to list information that is specific to the VG.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2

  VG Size               1012.00 MiB
  PE Size               4.00 MiB

  Total PE              253
  Alloc PE / Size       75 / 300.00 MiB

  Free  PE / Size       178 / 712.00 MiB
  VG UUID               jK5M1M-Yvlk-kxU2-bxmS-dNjQ-Bs3L-DRlJNc
```

`VG Name` shows the name of the volume group.

`VG Size` displays the total size of the storage pool that is available for LV allocation.

`Total PE` shows the total size of PE units.

`Free PE / Size` shows the available space in the VG to create or extend LVs.

Display Logical Volume Information

The `lvdisplay` command displays information about logical volumes. Use the command without arguments to list information of all LVs. Providing the name of the LV as an argument displays the information that is specific to the LV.

```
[root@host ~]# lvdisplay /dev/vg01/lv01
```

```
--- Logical volume ---

LV Path                /dev/vg01/lv01
LV Name                lv01

VG Name                vg01
LV UUID                FVmNel-u25R-dt3p-C5L6-VP2w-QRNP-scqrbq
LV Write Access        read/write
LV Creation host, time servera.lab.example.com, 2022-04-07 10:45:34 -0400
LV Status              available
# open                 1

LV Size                300.00 MiB

Current LE             75
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     8192
Block device           253:0
```

`LV Path` shows the device name of the LV.

`VG Name` shows the VG that is used for creating this LV.

`LV Size` shows the total size of the LV. Use the file-system tools to determine the free and used space for the LV.
`Current LE` shows the number of logical extents that this LV uses.

Extend and Reduce LVM Storage

After creating a logical volume, you can extend the volume to expand the file system. You might need to extend PV or VG to increase the storage capacity of the LV.

Extend a Volume Group Size

You might need to add more disk space to extend a VG. You can add physical volumes to a VG to extend its available size.

Prepare the physical device and create the physical volume when not present.

```
[root@host ~]# parted /dev/vdb mkpart primary 1072MiB 1648MiB
...output omitted...
[root@host ~]# parted /dev/vdb set 3 lvm on
...output omitted...
[root@host ~]# udevadm settle
[root@host ~]# pvcreate /dev/vdb3
  Physical volume "/dev/vdb3" successfully created.
```

The `vgextend` command adds the new PV to the VG. Provide the VG and PV names as arguments to the `vgextend` command.

```
[root@host ~]# vgextend vg01 /dev/vdb3
  Volume group "vg01" successfully extended
```

This command extends the `vg01` VG by the `/dev/vdb3` PV size.

Extend a Logical Volume Size

A benefit of using logical volumes is increasing their size without experiencing any downtime. Add free physical extents to the LV from the VG, to extend the LV capacity and to expand the file system. Use the `vgdisplay` command to confirm that the volume group has enough free space for the LV extension.

Use the `lvextend` command to extend the LV.

```
[root@host ~]# lvextend -L +500M /dev/vg01/lv01
  Size of logical volume vg01/lv01 changed from 300.00 MiB (75 extents) to 800.00 MiB
(200 extents).
  Logical volume vg01/lv01 successfully resized.
```

This command increases the size of the `lv01` logical volume by 500 MiB. The (+) plus sign in front of the size means adding this value to the existing size; otherwise, without the plus sign, the value defines the final size of the LV.

The `lvextend` command `-l` option expects the number of PE as the argument. The `lvextend` command `-L` option expects sizes in bytes, mebibytes, gibibytes, and similar.

Extend an XFS File System to the Logical Volume Size

The `xfs_growfs` command helps to expand the file system to occupy the extended LV. The target file system must be mounted before you use the `xfs_growfs` command. You can continue to use the file system when resizing.

```
[root@host ~]# xfs_growfs /mnt/data/
...output omitted...
data blocks changed from 76800 to 204800
```

Important

Always run the `xfs_growfs` command after executing the `lvextend` command. Use the `lvextend` command `-r` option to run the two steps consecutively. After extending the LV, resize the file system by using the `fsadm` command. This option supports several other file systems.

Extend an EXT4 File System to the Logical Volume Size

The steps for extending LV by using the `ext4` file system are the same as for LV by using the `XFS` file system, except for the step to resize the file system.

The `resize2fs` command expands the file system to occupy the new extended LV. You can continue to use the file system when resizing.

```
[root@host ~]# resize2fs /dev/vg01/lv01
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/vg01/lv01 to 256000 (4k) blocks.
The filesystem on /dev/vg01/lv01 is now 256000 (4k) blocks long.
```

The primary difference between `xfs_growfs` and `resize2fs` is the argument that is passed to identify the file system. The `xfs_growfs` command takes the mount point as an argument, and the `resize2fs` command takes the LV name as an argument. The `xfs_growfs` command supports only an online resize, whereas the `resize2fs` command supports both online and offline resizing. You can resize an `ext4` file system up or down, but you can resize an `XFS` file system only up.

Extend Swap Space Logical Volumes

You can extend the LVs that are used as swap space; however, the process differs from expanding the `ext4` or `XFS` file system. Logical volumes that are used as swap space must be offline to extend them.

Use the `swapoff` command to deactivate the swap space on the LV.

```
[root@host ~]# swapoff -v /dev/vg01/swap
swapoff /dev/vg01/swap
```

Use the `lvextend` command to extend the LV.

```
[root@host ~]# lvextend -L +300M /dev/vg01/swap
  Size of logical volume vg01/swap changed from 500.00 MiB (125 extents) to 800.00 Mi
B (200 extents).
  Logical volume vg01/swap successfully resized.
```

Use the `mkswap` command to format the LV as a swap space.

```
[root@host ~]# mkswap /dev/vg01/swap
mkswap: /dev/vg01/swap: warning: wiping old swap signature.
Setting up swapspace version 1, size = 800 MiB (838856704 bytes)
no label, UUID=25b4d602-6180-4b1c-974e-7f40634ad660
```

Use the `swapon` command to activate the swap space on the LV.

```
[root@host ~]# swapon /dev/vg01/swap
```

Reduce Volume Group Storage

Reducing a VG involves removing unused PVs from the VG. The `pvmove` command moves data from extents on one PV to extents on another PV with enough free extents in the same VG. You can continue to use the LV from the same VG when reducing. Use the `pvmove` command `-A` option to automatically back up the metadata of the VG after a change. This option uses the `vgcfgbackup` command to back up metadata automatically.

```
[root@host ~]# pvmove /dev/vdb3
```

Warning

Before using the `pvmove` command, back up the data that is stored on all LVs in the VG. An unexpected power loss during the operation might leave the VG in an inconsistent state, which might cause a loss of data on LVs.

Use the `vgreduce` command to remove a PV from a VG.

```
[root@host ~]# vgreduce vg01 /dev/vdb3
  Removed "/dev/vdb3" from volume group "vg01"
```

Important

The GFS2 and XFS file systems do not support shrinking, so you cannot reduce the size of an LV.

Remove LVM Storage

Use the `lvremove`, `vgremove`, and `pvremove` commands to remove an LVM component that is no longer required.

Prepare the File System

Move to another file system all data that must be kept. Use the `umount` command to unmount the file system, and then remove any `/etc/fstab` entries that are associated with this file system.

```
[root@host ~]# umount /mnt/data
```

Warning

Removing a logical volume destroys any data that is stored on the logical volume. Back up or move your data **before** you remove the logical volume.

Remove the Logical Volume

Use the `lvremove` *`DEVICE-NAME`* command to remove a logical volume that is no longer needed. Unmount the LV file system before running this command. The command prompts for confirmation before removing the LV.

```
[root@host ~]# lvremove /dev/vg01/lv01
Do you really want to remove active logical volume vg01/lv01? [y/n]: y
  Logical volume "lv01" successfully removed.
```

The LV's physical extents are freed and available to assign to existing or new LVs in the volume group.

Remove the Volume Group

Use the `vgremove` *`VG-NAME`* command to remove a volume group that is no longer needed.

```
[root@host ~]# vgremove vg01
  Volume group "vg01" successfully removed
```

The VG's physical volumes are freed and available to assign to existing or new VGs on the system.

Remove the Physical Volumes

Use the `pvremove` command to remove physical volumes that are no longer needed. Use a space-delimited list of PV devices to remove more than one device at a time. This command deletes the PV metadata from the partition (or disk). The partition is now available for reallocation or reformatting.

```
[root@host ~]# pvremove /dev/vdb1 /dev/vdb2
  Labels on physical volume "/dev/vdb1" successfully wiped.
  Labels on physical volume "/dev/vdb2" successfully wiped.
```

Guided Exercise: Create and Extend Logical Volumes

In this lab, you create and extend a physical volume, volume group, logical volume, and an XFS file system. You also persistently mount the logical volume file system.

Outcomes

- Create physical volumes, volume groups, and logical volumes with LVM tools.
- Create file systems on logical volumes and persistently mount them.
- Extend the volume group to include an additional physical volume.
- Resize the logical volume when the file system is still mounted and in use.

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

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

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

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

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

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

6. Create the physical device partition on the `/dev/vdb` storage device.
    1. Create two partitions of 256 MiB each and set to the Linux LVM type. Use the `first` and `second` names for these partitions.

```
[root@servera ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb mkpart first 1MiB 258MiB
Information: You may need to update /etc/fstab.
```

```

[root@servera ~]# parted /dev/vdb set 1 lvm on
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb mkpart second 258MiB 514MiB
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb set 2 lvm on
```

```
Information: You may need to update /etc/fstab.
```

15. Register the new partitions with the kernel.

```
[root@servera ~]# udevadm settle
```

16. List the partitions on the `/dev/vdb` storage device. In the `Number` column, the `1` and `2` values correspond to the `/dev/vdb1` and `/dev/vdb2` device partitions. The `Flags` column indicates the partition type.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End    Size   File system  Name    Flags
 1      1049kB  271MB  269MB               first   lvm
```

```
 2      271MB   539MB  268MB               second  lvm
```

7. Label the two new partitions as physical volumes.

```
[root@servera ~]# pvcreate /dev/vdb1 /dev/vdb2
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Creating devices file /etc/lvm/devices/system.devices
```

11. Create the `servera_group` volume group by using the two new PVs.

```
12. [root@servera ~]# vgcreate servera_group /dev/vdb1 /dev/vdb2
```

```
  Volume group "servera_group" successfully created
```

13. Create the `servera_volume` logical volume with a size of 400 MiB. This command creates the `/dev/servera_group/servera_volume` LV without a file system.

```
14. [root@servera ~]# lvcreate -n servera_volume -L 400M servera_group
```

```
  Logical volume "servera_volume" created.
```

15. Format the newly created LV and mount it persistently.
    1. Format the `servera_volume` LV with the XFS file system.

    ```
    2. [root@servera ~]# mkfs -t xfs /dev/servera_group/servera_volume
    ```

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

    3. Create the `/data` directory as a mount point.

    ```
    [root@servera ~]# mkdir /data
    ```

    4. To persistently mount the newly created file system, add the following content in the `/etc/fstab` file:

    ```
    /dev/servera_group/servera_volume /data xfs defaults 0 0
    ```

    5. Mount the `servera_volume` LV.

    ```
    [root@servera ~]# mount /data
    ```

16. Verify that the mounted file system is accessible, and display the status information of the LVM.
    1. Verify that you can copy files to the `/data` directory.

    ```
    2. [root@servera ~]# cp -a /etc/*.conf /data
    3. [root@servera ~]# ls /data | wc -l
    ```

```
32
```

4. View the PV status information. The output shows that the PV uses the `servera_group` VG. The PV has a size of 256 MiB and a physical extent size of 4 MiB.

   The VG contains 63 PEs, of which 27 PEs are available for allocation, and 36 PEs are currently allocated to LVs. Use the following calculation for allocating the volume size in MiBs:

   - Total 252 MiB (63 PEs x 4 MiB)
   - Free 108 MiB (27 PEs x 4 MiB)
   - Allocated 144 MiB (36 PEs x 4 MiB)

```
[root@servera ~]# pvdisplay /dev/vdb2
  --- Physical volume ---
  PV Name               /dev/vdb2
  VG Name               servera_group
  PV Size               256.00 MiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              63
  Free PE               27
  Allocated PE          36
  PV UUID               FKKFYJ-wJiR-1jt2-sfy3-yjPy-TylN-LG92jj
```

5. View the VG status information of the `servera_group` VG. The output shows a VG size of `508` MiB with a PE size of `4` MiB. The available size from the VG is `108` MiB.

```
6. [root@servera ~]# vgdisplay servera_group
7.   --- Volume group ---
8.   VG Name               servera_group
9.   System ID
10.  Format                lvm2
11.  Metadata Areas        2
12.  Metadata Sequence No  2
```

```
13.   VG Access             read/write
14.   VG Status             resizable
15.   MAX LV                0
16.   Cur LV                1
17.   Open LV               1
18.   Max PV                0
19.   Cur PV                2
20.   Act PV                2
21.   VG Size               508.00 MiB
22.   PE Size               4.00 MiB
23.   Total PE              127
24.   Alloc PE / Size       100 / 400.00 MiB
25.   Free  PE / Size       27 / 108.00 MiB
```

```
      VG UUID               g0ahyT-90J5-iGic-nnb5-G6T9-tLdK-dX8c9M
```

26. View the status information for the `servera_volume` LV. The output shows the VG name for creating the LV. It also shows an LV size of `400` MiB and an LE size of `100`.

```
27. [root@servera ~]# lvdisplay /dev/servera_group/servera_volume
28.   --- Logical volume ---
29.   LV Path                /dev/servera_group/servera_volume
30.   LV Name                servera_volume
31.   VG Name                servera_group
32.   LV UUID                93MfUt-esgT-B5HM-r1p5-DVZH-n5cn-J5e2tw
33.   LV Write Access        read/write
34.   LV Creation host, time servera.lab.example.com, 2022-04-11 03:25:12 -040
    0
35.   LV Status              available
36.   # open                 1
37.   LV Size                400.00 MiB
38.   Current LE             100
39.   Segments               2
40.   Allocation             inherit
41.   Read ahead sectors     auto
```

```
42.  - currently set to     8192
```

```
  Block device          253:0
```

43. View the free disk space in human-readable units. The output shows the total size of `395` MiB with the available size of `372` MiB.

```
44. [root@servera ~]# df -h /data
45. Filesystem                               Size  Used Avail Use% Mounted on
```

```
/dev/mapper/servera_group-servera_volume  395M   24M  372M   6% /data
```

17. Create the physical resource on the `/dev/vdb` storage device.
    1. Create an additional partition of 512 MiB and set it to the Linux LVM type. Use the `third` name for this partition.

```
2.  [root@servera ~]# parted /dev/vdb mkpart third 514MiB 1026MiB
```

```
[root@servera ~]# parted /dev/vdb set 3 lvm on
```

3. Register the new partition with the kernel.

```
[root@servera ~]# udevadm settle
```

4. Add the new partition as a PV.

```
5.  [root@servera ~]# pvcreate /dev/vdb3
```

```
  Physical volume "/dev/vdb3" successfully created.
```

18. Using the newly created disk space, extend the file system on the `servera_volume` to be a total size of 700 MiB.
    1. Extend the `servera_group` VG by using the new `/dev/vdb3` PV.

```
2.  [root@servera ~]# vgextend servera_group /dev/vdb3
```

```
  Volume group "servera_group" successfully extended
```

3. Extend the existing `servera_volume` LV to 700 MiB.

```
4.  [root@servera ~]# lvextend -L 700M /dev/servera_group/servera_volume
5.    Size of logical volume servera_group/servera_volume changed from 400.00
    MiB (100 extents) to 700.00 MiB (175 extents).
```

```
  Logical volume servera_group/servera_volume successfully resized.
```

6. Extend the XFS file system by using the free space on the LV.

```
[root@servera ~]# **xfs_growfs /data**
...output omitted...
```

```
data blocks changed from 102400 to 179200
```

19. Verify that the LV size is extended, and that the contents are still present in the volume.

   1. Verify the size of the extended LV by using the `lvdisplay` command.

```
[root@servera ~]# lvdisplay /dev/servera_group/servera_volume
  --- Logical volume ---
  LV Path                /dev/servera_group/servera_volume
  LV Name                servera_volume
  VG Name                servera_group
  LV UUID                mLQhsD-hyL0-KC2B-2nug-o2Nc-0znS-Q428fK
  LV Write Access        read/write
  LV Creation host, time servera.lab.example.com, 2022-04-12 06:04:12 -0400
  LV Status              available
  # open                 1
  LV Size                700.00 MiB
  Current LE             175
  Segments               3
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
```

```
  Block device           253:0
```

   18. Verify the new file-system size. Verify that the previously copied files are still present.

```
[root@servera ~]# df -h /data
Filesystem                               Size  Used Avail Use% Mounted on
```

```
/dev/mapper/servera_group-servera_volume  695M   26M  670M   4% /data
[root@servera ~]# ls /data | wc -l
```

```
32
```

20. Return to the `workstation` machine as the `student` user.

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

```
Connection to servera closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

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

This concludes the section.

Manage Layered Storage

Objectives

Analyze the multiple storage components that make up the layers of the storage stack.

Storage Stack

Storage in RHEL is composed of multiple layers of drivers, managers, and utilities that are mature, stable, and full of modern features. Managing storage requires familiarity with stack components, and recognizing that storage configuration affects the boot process, application performance, and the ability to provide needed storage features for specific application use cases.

Figure 8.2: Storage stack

Previous sections in the Red Hat System Administration courses presented XFS file systems, network storage sharing, partitioning, and the Logical Volume Manager. This section shows the bottom-to-top RHEL storage stack and introduces each layer.

This section also covers Stratis, the daemon that unifies, configures, and monitors the underlying RHEL storage stack components, and provides automated local storage management from either the CLI or the RHEL web console.

Block Device

Block devices are at the bottom of the storage stack, and present a stable, consistent device protocol that enables including almost any block device transparently in a RHEL storage configuration. Most block devices today are accessed through the RHEL SCSI device driver, and appear as a SCSI device, including earlier ATA hard drives, solid-state devices, and common enterprise host bus adapters (HBAs). RHEL also supports iSCSI, Fibre Channel over Ethernet (FCoE), virtual machine driver (`virtio`), serial-attached SCSI (SAS), Non-Volatile Memory Express (NVMe), and other block devices.

An iSCSI target can be a dedicated physical device in a network or an iSCSI software-configured logical device on a networked storage server. The target is the endpoint in a SCSI protocol bus communication, to access the storage as Logical Unit Numbers (LUNs).

The Fibre Channel over Ethernet (FCoE) protocol transmits Fibre Channel frames over Ethernet networks. Typically, each data center has dedicated LAN and Storage Area Network (SAN) cabling, which is uniquely configured for its traffic. With FCoE, both traffic types can be combined into a larger, converged, Ethernet network architecture. FCoE benefits include lower hardware and energy costs.

Multipath

A path is a connection between a server and the underlying storage. Device Mapper multipath (`dm-multipath`) is a RHEL native multipath tool for configuring redundant I/O paths into a single, path-aggregated logical device. A logical device that is created by using the device mapper (`dm`) appears as a unique block device in the `/dev/mapper/` directory for each LUN that is attached to the system.

You can also implement storage multipath redundancy by using network bonding when the storage, such as iSCSI and FCoE, uses network cabling.

Partitions

A block device can be further divided into partitions. Partitions might consume the entire block device size, or divide the block device for creating multiple partitions. You can use these partitions to create a file system, LVM devices, or directly for database structures or other raw storage.

RAID

A Redundant Array of Inexpensive Disks (RAID) is a storage virtualization technology that creates large logical volumes from multiple physical or virtual block device components. Different forms of RAID volumes offer data redundancy, performance improvement, or both, by implementing mirroring or striping layouts.

LVM supports RAID levels 0, 1, 4, 5, 6, and 10. RAID logical volumes that LVM creates and manages use the Multiple Devices (`mdadm`) kernel drivers. When not using LVM, Device Mapper RAID (`dm-raid`) provides a device mapper interface to the `mdadm` kernel drivers.

Logical Volume Manager

LVM physical volumes, volume groups, and logical volumes were discussed in a previous section. LVM can take almost any form of physical or virtual block devices, and can build storage as new logical storage volumes, and effectively hides the physical storage configuration from applications and other storage clients.

You can stack LVM volumes and implement advanced features such as encryption and compression for each part of the stack. The stack LVM volumes have mandated rules and recommended practices to follow for practical layering for specific scenarios. You can use case-specific recommendations from the *Configuring and Managing Logical Volumes* user guide.

LVM supports *LUKS encryption*, where a lower block device or partition is encrypted and presented as a secure volume to create a file system on top. The practical advantage for LUKS over file-system or file-based encryption is that a LUKS-encrypted device does not allow public visibility or access to the file-system structure. The LUKS-encrypted device ensures that a physical device remains secure even when removed from a computer.

LVM now incorporates VDO deduplication and compression as a configurable feature of regular logical volumes. You can use LUKS encryption and VDO together with logical volumes, where the LVM LUKS encryption is enabled underneath the LVM VDO volume.

File System or Other Use

The top layer of the stack is typically a file system, and can be used as raw space for databases or custom application data requirements. RHEL supports multiple file-system types, and recommends XFS for most modern use cases. XFS is required when the utility that implements LVM is Red Hat Ceph Storage or the Stratis storage tool.

Database server applications consume storage in different ways, depending on their architecture and size. Some smaller databases store their structures in regular files that are contained in a file system. Because of the additional overhead or restrictions of file system access, this architecture has scaling limits. Larger databases that bypass file system caching, and that use their own caching mechanisms, create their database structures on raw storage. Logical volumes are suitable for database and other raw storage use cases.

Red Hat Ceph Storage creates its own storage management metadata structures on raw devices, to create Ceph Object Storage Devices (OSDs). In the latest Red Hat Ceph Storage versions, Ceph uses LVM to initialize disk devices for use as OSDs. More information is available in the *Cloud Storage with Red Hat Ceph Storage* (CL260) course.

Stratis Storage Management

Stratis is a local storage management tool that Red Hat and the upstream Fedora community developed. Stratis configures initial storage, changes a storage configuration, and uses advanced storage features.

Important

Stratis is currently available as a Technology Preview, but is expected to be supported in a later RHEL 9 version. For information about Red Hat scope of support for Technology Preview features, see the Technology Features Support Scope document.

Red Hat encourages customers to provide feedback when deploying Stratis.

Stratis runs as a service that manages pools of physical storage devices, and transparently creates and manages volumes for the newly created file systems.

Stratis builds file systems from shared pools of disk devices by using the *thin provisioning* concept. Instead of immediately allocating physical storage space to the file system when you create it, Stratis dynamically allocates that space from the pool as the file system stores more data. Therefore, the file system might appear to be 1 TiB, but might have only 100 GiB of real storage that is allocated to it from the pool.

You can create multiple pools from different storage devices. From each pool, you can create one or more file systems. Currently, you can create up to 2^{24} file systems per pool.

Stratis builds the components that make up a Stratis-managed file system from standard Linux components. Internally, Stratis uses the Device Mapper infrastructure that LVM also uses. Stratis formats the managed file systems with XFS.

Figure 8.3: Stratis architecture illustrates how Stratis assembles the elements of its storage management solution. Stratis assigns block storage devices such as hard disks or SSDs to pools. Each device contributes some physical storage to the pool. Then, Stratis creates file systems from the pools, and maps physical storage to each file system as needed.

Figure 8.3: Stratis architecture

Stratis Administration Methods

To manage file systems with the Stratis storage management solution, install the `stratis-cli` and `stratisd` packages. The `stratis-cli` package provides the `stratis` command, which sends reconfiguration requests to the `stratisd` system daemon. The `stratisd` package provides the `stratisd` service, which handles reconfiguration requests, and manages and monitors Stratis block devices, pools, and file systems.

Stratis administration is included in the RHEL web console.

Warning

Reconfigure file systems created by Stratis only with Stratis tools and commands.

Stratis uses stored metadata to recognize managed pools, volumes, and file systems. Manually configuring Stratis file systems with non-Stratis commands can result in overwriting that metadata, and can prevent Stratis from recognizing the file system volumes that it previously created.

Install and Enable Stratis

To use Stratis, ensure that your system has the required software and that the `stratisd` service is running. Install the `stratis-cli` and `stratisd` packages, and start and enable the `stratisd` service.

```
[root@host ~]# dnf install stratis-cli stratisd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
[root@host ~]# systemctl enable --now stratisd
```

Create Stratis Pools

Create pools of one or more block devices by using the `stratis pool create` command. Then, use the `stratis pool list` command to view the list of available pools.

```
[root@host ~]# stratis pool create pool1 /dev/vdb
[root@host ~]# stratis pool list
Name                  Total Physical   Properties            UUID
pool1   5 GiB / 37.63 MiB / 4.96 GiB      ~Ca,~Cr   11f6f3c5-5...
```

Warning

The `stratis pool list` command displays the storage space in use and the available pool space. Currently, if a pool becomes full, then further data that is written to the pool's file systems is quietly discarded.

Use the `stratis pool add-data` command to add block devices to a pool. Then, use the `stratis blockdev list` command to verify the block devices of a pool.

```
[root@host ~]# stratis pool add-data pool1 /dev/vdc
[root@host ~]# stratis blockdev list pool1
Pool Name   Device Node   Physical Size   Tier
pool1       /dev/vdb              5 GiB   Data
pool1       /dev/vdc              5 GiB   Data
```

Manage Stratis File Systems

Use the `stratis filesystem create` command to create a file system from a pool. The links to the Stratis file systems are in the `/dev/stratis/pool1` directory. Use the `stratis filesystem list` command to view the list of available file systems.

```
[root@host ~]# stratis filesystem create pool1 fs1
[root@host ~]# stratis filesystem list
Pool Name   Name   Used      Created             Device                 UUID
pool1       fs1    546 MiB   Apr 08 2022 04:05   /dev/stratis/pool1/fs1   c7b5719...
```

Create a Stratis file system snapshot by using the `stratis filesystem snapshot` command. Snapshots are independent of the source file systems. Stratis dynamically allocates the snapshot storage space, and uses an initial 560 MB to store the file system's journal.

```
[root@host ~]# stratis filesystem snapshot pool1 fs1 snapshot1
```

Persistently Mount Stratis File Systems

You can persistently mount Stratis file systems by editing the `/etc/fstab` file and specifying the details of the file system. Use the `lsblk` command to display the UUID of the file system to use in the `/etc/fstab` file to identify the file system. You can also use the `stratis filesystem list` command to obtain the UUID of the file system.

```
[root@host ~]# lsblk --output=UUID /dev/stratis/pool1/fs1
UUID
c7b57190-8fba-463e-8ec8-29c80703d45e
```

The following example shows an entry in the `/etc/fstab` file to mount a Stratis file system persistently. This example entry is a single long line in the file. The `x-systemd.requires=stratisd.service` mount option delays mounting the file system until the `systemd` daemon starts the `stratisd` service during the boot process.

```
UUID=c7b57190-8fba-463e-8ec8-29c80703d45e /dir1 xfs defaults,x-systemd.requires=strat
isd.service 0 0
```

Important

If you do not include the `x-systemd.requires=stratisd.service` mount option in the `/etc/fstab` file for each Stratis file system, then the machine fails to start correctly, and aborts to `emergency.target` the next time that you reboot it.

Warning

Do not use the `df` command to query Stratis file system space.

The `df` command reports that any mounted Stratis-managed XFS file system is 1 TiB, regardless of the current allocation. Because the file system is thinly provisioned, a pool might not have enough physical storage to back the entire file system. Other file systems in the pool might use up all the available storage.

Therefore, it is possible to consume the whole storage pool, even if the `df` command reports that the file system has available space. Writes to a file system with no available pool storage can fail.

Instead, always use the `stratis pool list` command to monitor a pool's available storage accurately.

Guided Exercise: Manage Layered Storage

In this exercise, you use Stratis to create file systems from pools of storage that physical storage devices provide.

Outcomes

- Create a thin-provisioned file system by using the Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thin-provisioned file system.

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

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

```
[student@workstation ~]$ lab start lvm-stratis
```

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

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

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

6. Install the `stratisd` and `stratis-cli` packages.

```
[root@servera ~]# dnf install stratisd stratis-cli
...output omitted...
```

```
9.  Is this ok [y/N]: y
10. ...output omitted...
```

```
Complete!
```

11. Activate the `stratisd` service.

```
[root@servera ~]# systemctl enable --now stratisd
```

12. Ensure that the `stratispool1` Stratis pool exists on the `/dev/vdb` block device.
    1. Create the `stratispool1` Stratis pool.

    ```
    [root@servera ~]# stratis pool create stratispool1 /dev/vdb
    ```

    2. Verify the availability of the `stratispool1` pool. Note the size of the pool.

    ```
    3.  [root@servera ~]# stratis pool list
    4.  Name                          Total Physical   Properties              UU
        ID
    ```

    ```
    stratispool1   5 GiB / 37.63 MiB / 4.96 GiB      ~Ca,~Cr   3557c389-7...
    ```

13. Expand the capacity of the `stratispool1` pool by adding the `/dev/vdc` block device.
    1. Add the `/dev/vdc` block device to the `stratispool1` pool.

    ```
    [root@servera ~]# stratis pool add-data stratispool1 /dev/vdc
    ```

    2. Verify the size of the `stratispool1` pool. The `stratispool1` pool size increases when you add the block device.

    ```
    3.  [root@servera ~]# stratis pool list
    4.  Name                           Total Physical   Properties            UUI
        D
    ```

    ```
    stratispool1   10 GiB / 41.63 MiB / 9.96 GiB      ~Ca,~Cr   3557c389-7..
    .
    ```

    5. Verify the block devices that are currently members of the `stratispool1` pool.

```
6. [root@servera ~]# stratis blockdev list stratispool1
7. Pool Name     Device Node  Physical Size  Tier
8. stratispool1  /dev/vdb             5 GiB  Data
```

```
stratispool1  /dev/vdc             5 GiB  Data
```

14. Add a thin-provisioned file system called `stratis-filesystem1` in the `stratispool1` pool. Mount the file system on the `/stratisvol` directory. Create a file on the `stratis-filesystem1` file system called `file1` that contains the text `Hello World!`. Modify the `/etc/fstab` file to persistently mount the file system on the `/stratisvol` directory.
    1. Create the thin-provisioned `stratis-filesystem1` file system on the `stratispool1` pool. It might take up to a minute for the command to complete.

    ```
    [root@servera ~]# stratis filesystem create stratispool1 stratis-filesys
    tem1
    ```

    2. Verify the availability of the `stratis-filesystem1` file system, and note its current usage. The usage of the file system increases on demand in the later steps.

```
3. [root@servera ~]# stratis filesystem list
4. Pool Name      Name                  Used      Created             Devic
   e                                        UUID
```

```
stratispool1   stratis-filesystem1   546 MiB   Apr 08 2022 07:12   /dev/
stratis/stratispool1/stratis-filesystem1   48e8...
```

5. Create the `/stratisvol` directory.

```
[root@servera ~]# mkdir /stratisvol
```

6. Mount the `stratis-filesystem1` file system on the `/stratisvol` directory.

```
7. [root@servera ~]# mount /dev/stratis/stratispool1/stratis-filesystem1 \
```

```
/stratisvol
```

8. Create the `/stratisvol/file1` text file.

```
[root@servera ~]# echo "Hello World!" > /stratisvol/file1
```

9. Unmount the `/stratisvol` volume.

```
[root@servera ~]# umount /stratisvol
```

10. Obtain the UUID of the file system. The UUID would be different in your system.

```
[root@servera ~]# lsblk --output=UUID \
/dev/stratis/stratispool1/stratis-filesystem1
UUID
```

```
d18cb4fc-753c-473a-9ead-d6661533b475
```

14. Modify the `/etc/fstab` file to persistently mount the file system on the `/stratisvol` directory. To do so, use the `vim /etc/fstab` command and add the following line. Replace the UUID with the correct one for your system.

```
UUID=d18c... /stratisvol xfs defaults,x-systemd.requires=stratisd.service 0 0
```

15. Update the `systemd` daemon with the new `/etc/fstab` configuration file.

```
[root@servera ~]# systemctl daemon-reload
```

16. Mount the `stratisvol` volume and verify that the `stratis-filesystem1` volume is mounted on the `/stratisvol` directory.

```
[root@servera ~]# mount /stratisvol
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-3557...fbd3-thin-fs-48e8...9ebe on /stratisvol type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,sunit=2048,swidth=2048,
```

```
noquota)
```

15. Reboot your system and verify that the file system is persistently mounted across reboots.

```
16. [root@servera ~]# systemctl reboot
17. ...output omitted...
18. [student@workstation ~]$ ssh student@servera
19. ...output omitted...
20. [student@servera ~]$ sudo -i
21. [sudo] password for student: student
22. [root@servera ~]# mount
23. ...output omitted...
24. /dev/mapper/stratis-1-3557...fbd3-thin-fs-d18c...b475 on /stratisvol type xfs
    (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,sunit=2048,swidth=2
    048,
```

```
noquota,x-systemd.requires=stratisd.service)
```

25. Verify that the `stratis-filesystem1` thin-provisioned file system dynamically grows as the data on the file system grows.
    1. View the current usage of the `stratis-filesystem1` file system.

    ```
    2. [root@servera ~]# stratis filesystem list
    3. Pool Name      Name                   Used      Created              Devic
       e                                          UUID
    ```

    ```
    stratispool1   stratis-filesystem1   546 MiB   Apr 08 2022 07:12   /dev/
    stratis/stratispool1/stratis-filesystem1   48e8...
    ```

    4. Create a 2 GiB file on the `stratis-filesystem1` file system. It might take up to a minute for the command to complete.

    ```
    [root@servera ~]# dd if=/dev/urandom of=/stratisvol/file2 bs=1M count=20
    48
    ```

    5. Verify the used space in the `stratis-filesystem1` file system.

       The output shows that the used space in the `stratis-filesystem1` file system increased. The used-space increase confirms that the thin-provisioned file system dynamically expands as needed.

    ```
    [root@servera ~]# stratis filesystem list
    Pool Name      Name                   Used      Created              Devic
    e                                          UUID
    ```

```
stratispool1   stratis-filesystem1   2.60 GiB   Apr 08 2022 07:12   /dev
/stratis/stratispool1/stratis-filesystem1   48e8...
```

26. Create a snapshot called `stratis-filesystem1-snap` of the `stratis-filesystem1` file system. The snapshot provides you with access to any file that you delete from the `stratis-filesystem1` file system.
    1. Create a snapshot of the `stratis-filesystem1` file system. It might take up to a minute for the command to complete.

```
2. [root@servera ~]# stratis filesystem snapshot stratispool1 \
```

```
stratis-filesystem1 stratis-filesystem1-snap
```

    3. Verify the availability of the snapshot.

```
4. [root@servera ~]# stratis filesystem list
5. Pool Name      Name                       Used       Created
   Device                                              UUID
6. stratispool1   stratis-filesystem1-snap   2.73 GiB   Apr 08 2022 07:22
   /dev/stratis/stratispool1/stratis-filesystem1-snap   5774...
```

```
stratispool1   stratis-filesystem1       2.73 GiB   Apr 08 2022 07:12
/dev/stratis/stratispool1/stratis-filesystem1       48e8...
```

    7. Remove the `/stratisvol/file1` file.

```
8. [root@servera ~]# rm /stratisvol/file1
```

```
rm: remove regular file '/stratisvol/file1'? y
```

    9. Create the `/stratisvol-snap` directory.

```
[root@servera ~]# mkdir /stratisvol-snap
```

    10. Mount the `stratis-filesystem1-snap` snapshot on the `/stratisvol-snap` directory.

```
11. [root@servera ~]# mount /dev/stratis/stratispool1/stratis-filesystem1-
    snap \
```

```
/stratisvol-snap
```

    12. Verify that you can still access the file that you deleted from the `stratis-filesystem1` file system in the snapshot.

```
13. [root@servera ~]# cat /stratisvol-snap/file1
```

```
Hello World!
```

27. Unmount the `/stratisvol` and `/stratisvol-snap` volumes.

```
28. [root@servera ~]# umount /stratisvol-snap
```

```
[root@servera ~]# umount /stratisvol
```

29. Remove the `stratis-filesystem1` thin-provisioned file system and the `stratis-filesystem1-snap` snapshot from the system.
    1. Destroy the `stratis-filesystem1-snap` snapshot.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesy
    stem1-snap
    ```

    2. Destroy the `stratis-filesystem1` file system.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesy
    stem1
    ```

    3. Return to the `workstation` system as the `student` user.

    ```
    4. [root@servera ~]# exit
    5. logout
    6. [student@servera ~]$ exit
    7. logout
    8. Connection to servera closed.
    ```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish lvm-stratis
```

This concludes the section.

Summary

- You can use LVM to create flexible storage by allocating space on multiple storage devices.
- Physical volumes, volume groups, and logical volumes are managed by the `pvcreate`, `vgreduce`, and `lvextend` commands.
- Logical volumes can be formatted with a file system or swap space, and they can be mounted persistently.
- Storage can be added to volume groups, and logical volumes can be extended dynamically.
- Storage stack uses layers and components to manage storage efficiently.
- Virtual Data Optimizer (VDO) uses LVM for compression and deduplication of data.
- You can use Stratis to configure initial storage or to enable advanced storage features.

Chapter 9. Access Network-Attached Storage

Abstract

Goal	Access network-attached storage with the NFS protocol.
Objectives	• Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command. • Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.
Sections	• Manage Network-Attached Storage with NFS (and Guided Exercise) • Automount Network-Attached Storage (and Guided Exercise)
Lab	Access Network-Attached Storage

Manage Network-Attached Storage with NFS

Objectives

Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command.

Accessing Exported NFS Directories

The *Network File System* (NFS) is an internet standard protocol that Linux, UNIX, and similar operating systems use as their native network file system. NFS is an open standard that supports native Linux permissions and file-system attributes.

By default, Red Hat Enterprise Linux 9 uses NFS version 4.2. RHEL fully supports both NFSv3 and NFSv4 protocols. NFSv3 might use either a TCP or a UDP transport protocol, but NFSv4 supports only TCP connections.

NFS servers *export* directories. NFS clients mount exported directories to an existing local mount point directory. NFS clients can mount exported directories in multiple ways:

- **Manually** by using the `mount` command.
- **Persistently at boot** by configuring entries in the `/etc/fstab` file.
- **On demand** by configuring an automounter method.

The automounter methods, which include the `autofs` service and the `systemd.automount` facility, are discussed in the `Automount Network-Attached Storage` section. You must install the `nfs-utils` package to obtain the client tools for manually mounting, or for automounting, to obtain exported NFS directories.

```
[root@host ~]# dnf install nfs-utils
```

RHEL also supports mounting *shared* directories from Microsoft Windows systems by using the same methods as for the NFS protocol, by using either the Server Message Block (SMB) or the Common Internet File System (CIFS) protocols. Mounting options are protocol-specific, and depend on your Windows Server or Samba Server configuration.

Query a Server's Exported NFS Directories

The NFS protocol changed significantly between NFSv3 and NFSv4. The method to query a server to view the available exports is different for each protocol version.

NFSv3 used the RPC protocol, which requires a file server that supports NFSv3 connections to run the `rpcbind` service. An NFSv3 client connects to the `rpcbind` service at port 111 on the server to request NFS service. The server responds with the current port for the NFS service. Use the `showmount` command to query the available exports on an RPC-based NFSv3 server.

```
[root@host ~]# showmount --exports server


Export list for server

/shares/test1
```

```
/shares/test2
```

The NFSv4 protocol eliminated the use of the legacy RPC protocol for NFS transactions. Use of the `showmount` command on a server that supports only NFSv4 times out without receiving a response, because the `rpcbind` service is not running on the server. However, querying an NFSv4 server is simpler than querying an NFSv3 server.

NFSv4 introduced an *export tree* that contains all of the paths for the server's exported directories. To view all of the exported directories, mount the root (`/`) of the server's export tree. Mounting the export tree's root provides browseable paths for all exported directories, as children of the tree's root directory, but does not mount ("bind") any of the exported directories.

```
[root@host ~]# mkdir /mountpoint
[root@host ~]# mount server:/ /mountpoint
[root@host ~]# ls /mountpoint
```

To mount an NFSv4 export when browsing the mounted export tree, change directory to an exported directory path. Alternatively, use the `mount` command with an exported directory's full path name to mount a single exported directory. Exported directories that use Kerberos security do not allow mounting or accessing a directory when browsing an export tree, even though you can view the export's path name. Mounting Kerberos-protected shares requires additional server configuration and the use of Kerberos user credentials, which are discussed in the *Red Hat Security: Identity Management and Active Directory Integration (RH362)* training course.

Manually Mount Exported NFS Directories

After you identify the NFS export to mount, create a local mount point if it does not yet exist. Although the `/mnt` directory is available for use as a temporary mount point, recommended practice is not to use `/mnt` for long-term or persistent mounting.

```
[root@host ~]# mkdir /mountpoint
```

As with local volume file systems, mount the NFS export to access its contents. NFS shares can be mounted temporarily or permanently, only by a privileged user.

```
[root@host ~]# mount -t nfs -o rw,sync server:/export /mountpoint
```

The `-t nfs` option specifies the NFS file-system type. However, when the `mount` command detects the *server:/export* syntax, the command defaults to the NFS type. With the `-o` flag, you can add a list of comma-separated options to the `mount` command. In the example, the `rw` option specifies that the exported file system is mounted with read/write access. The `sync` option specifies synchronous transactions to the exported file system. This method is strongly recommended for all production network mounts where transactions must be completed or else return as failed.

Using a manual `mount` command is not persistent. When the system reboots, that NFS export is not still mounted. Manual mounts are useful for providing temporary access to an exported directory, or for test mounting an NFS export before persistently mounting it.

Persistently Mount Exported NFS Directories

To persistently mount an NFS export, edit the `/etc/fstab` file, and add the mount entry with similar syntax to manual mounting.

```
[root@host ~]# vim /etc/fstab
...
server:/export  /mountpoint  nfs  rw  0 0
```

Then, you can mount the NFS export by using only the mount point. The `mount` command obtains the NFS server and mount options from the matching entry in the `/etc/fstab` file.

```
[root@host ~]# mount /mountpoint
```

Unmount Exported NFS Directories

As a privileged user, unmount an NFS export with the `umount` command. Unmounting a share does not remove its entry in the `/etc/fstab` file, if that file exists. Entries in the `/etc/fstab` file are persistent and are remounted during boot.

```
[root@host ~]# umount /mountpoint
```

A mounted directory can sometimes fail to unmount, and returns an error that the device is busy. The device is busy because either an application is keeping a file open within the file system, or some user's shell has a working directory in the mounted file-system's root directory or below it.

To resolve the error, check your own active shell windows, and use the `cd` command to leave the mounted file system. If subsequent attempts to unmount the file system still fail, then use the `lsof` (*list open files*) command to query the mount point. The `lsof` command returns a list of open file names and the process which is keeping the file open.

```
[root@host ~]# lsof  /mountpoint
COMMAND  PID   USER  FD   TYPE  DEVICE  SIZE/OFF  NODE  NAME
program  5534  user  txt  REG   252.4   910704    128   /home/user/program
```

With this information, gracefully close any processes that are using files on this file system, and retry the unmount. In critical scenarios only, when an application cannot be closed gracefully, kill the process to close the file. Alternatively, use the `umount -f` option to force the unmount, which can cause loss of unwritten data for all open files.

Guided Exercise: Manage Network-Attached Storage with NFS

In this exercise, you modify the `/etc/fstab` file to persistently mount an NFS export at boot time.

Outcomes

- Test an NFS server with the `mount` command.
- Configure NFS exports in the `/etc/fstab` configuration file to save changes even after a system reboots.

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

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

```
[student@workstation ~]$ lab start netstorage-nfs
```

Instructions

A shipping company uses a central NFS server, `serverb`, to host various exported documents and directories. Users on `servera`, who are all members of the `admin` group, need access to the persistently mounted NFS export.

The following list provides the environment characteristics for completing this exercise:

- The `serverb` machine exports the `/shares/public` directory, which contains some text files.
- Members of the `admin` group (`admin1`, `sysmanager1`) have read and write access to the `/shares/public` exported directory.
- The mount point on `servera` must be the `/public` directory.
- All user passwords are set to `redhat`.
- The `nfs-utils` package is already installed.

1. Log in to `servera` as the `student` user and switch to the `root` user.
    1. Log in to `servera` as the `student` user and switch to the `root` user.

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

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

2. Test the NFS server on `serverb` with `servera` as the NFS client.
    1. Create the `/public` mount point on the `servera` machine.

    ```
    [root@servera ~]# mkdir /public
    ```

    2. On `servera`, verify that the `/shares/public` NFS export from `serverb` successfully mounts to the `/public` directory.

```
3.  [root@servera ~]# mount -t nfs \
```

```
serverb.lab.example.com:/shares/public /public
```

4. List the contents of the mounted NFS export.

```
5.  [root@servera ~]# ls -l /public
6.  total 16
7.  -rw-r--r--. 1 root admin 42 Apr  8 22:36 Delivered.txt
8.  -rw-r--r--. 1 root admin 46 Apr  8 22:36 NOTES.txt
9.  -rw-r--r--. 1 root admin 20 Apr  8 22:36 README.txt
```

```
-rw-r--r--. 1 root admin 27 Apr  8 22:36 Trackings.txt
```

10. Explore the `mount` command options for the mounted NFS export.

```
11. [root@servera ~]# mount | grep public
12. serverb.lab.example.com:/shares/public on /public type nfs4
13. (rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync
14. ,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
```

```
local_lock=none,addr=172.25.250.11)
```

15. Unmount the NFS export.

```
[root@servera ~]# umount /public
```

3. Configure `servera` so that the `/shares/public` export is persistently mounted.
    1. Edit the `/etc/fstab` file.

    ```
    [root@servera ~]# vim /etc/fstab
    ```

    Add the following line to the end of the file:

    ```
    serverb.lab.example.com:/shares/public  /public  nfs  rw,sync  0 0
    ```

    2. Mount the exported directory.

    ```
    [root@servera ~]# mount /public
    ```

3. List the contents of the exported directory.

```
4.  [root@servera ~]# ls -l /public
5.  total 16
6.  -rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
7.  -rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
8.  -rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
```

```
-rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
```

9. Reboot the `servera` machine.

```
[root@servera ~]# systemctl reboot
```

4. After `servera` is finished rebooting, log in to `servera` as the `admin1` user and test the persistently mounted NFS export.
   1. Log in to `servera` as the `admin1` user.

```
2.  [student@workstation ~]$ ssh admin1@servera
```

```
[admin1@servera ~]$
```

   3. Test the NFS export that is mounted on the `/public` directory.

```
4.  [admin1@servera ~]$ ls -l /public
5.  total 16
6.  -rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
7.  -rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
8.  -rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
9.  -rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
10. [admin1@servera ~]$ cat /public/NOTES.txt
11. ###In this file you can log all your notes###
12. [admin1@servera ~]$ echo "This is a test" > /public/Test.txt
13. [admin1@servera ~]$ cat /public/Test.txt
```

```
This is a test
```

   14. Return to the `workstation` machine as the `student` user.

```
15. [admin1@servera ~]$ exit
```

```
16. logout
```

```
Connection to servera closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish netstorage-nfs
```

This concludes the section.

Automount Network-Attached Storage

Objectives

Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.

Mount NFS Exports with the Automounter

The *automounter* is a service (`autofs`) that automatically mounts file systems and NFS exports on demand, and automatically unmounts file systems and NFS exports when the mounted resources are no longer in current use.

The automounter function was created to solve the problem that unprivileged users do not have sufficient permissions to use the mount command. Without use of the mount command, normal users cannot access removable media such as CDs, DVDs, and removable disk drives. Furthermore, if a local or remote file system is not mounted at boot time by using the `/etc/fstab` configuration, then a normal user cannot mount and access those unmounted file systems.

The automounter configuration files are populated with file-system mount information, in a similar way to `/etc/fstab` entries. Although `/etc/fstab` file systems mount during system boot and remain mounted until system shutdown or other intervention, automounter file systems do not necessarily mount during system boot. Instead, automounter-controlled file systems mount on demand, when a user or application attempts to enter the file-system mount point to access files.

Automounter Benefits

Resource use for automounter file systems is equivalent to file systems that are mounted at boot, because a file system uses resources only when a program is reading and writing open files. Mounted but idle file systems and unmounted file systems use almost no resources.

The automounter advantage is that by unmounting the file system each time that it is no longer in use, the file system is protected from unexpected corruption when it is open. When the file system is directed to mount again, the `autofs` service uses the most current mount configuration, unlike an `/etc/fstab` mount, which might still use a configuration that was mounted months ago during the last system boot. Additionally, if your NFS server configuration includes redundant servers and paths, then the automounter can select the fastest connection each time that a new file system is requested.

The Automounter `autofs` Service Method

The `autofs` service supports the same local and remote file systems as in the `/etc/fstab` file, including NFS and SMB file sharing protocols, and supports the same protocol-specific mount options, including security parameters. File systems that are mounted through the automounter are available by default to all users, but can be restricted through access permission options.

Because the automounter is a client-side configuration that uses the standard `mount` and `umount` commands to manage file systems, automounted file systems in use exhibit the same behavior to file systems that are mounted by using `/etc/fstab`. The difference is that an automounter file system remains unmounted until the mount point is accessed, which causes the file system to mount immediately, and to remain mounted when the file system is in use. When all files on the file system are closed, and all users and processes leave the mount point directory, the automounter unmounts the file system after a minimal timeout.

Direct and Indirect Map Use Cases

The automounter supports both direct and indirect mount-point mapping, to handle the two types of demand mounting. A *direct* mount is when a file system mounts to an unchanging, known mount point location. Almost all the file system mounts that you configured, before learning about the automounter, are examples of direct mounts. A *direct* mount point exists as a permanent directory, the same as other normal directories.

An *indirect* mount is when the mount point location is not known until the mount demand occurs. An example of an indirect mount is the configuration for remote-mounted home directories, where a user's home directory includes their username in the directory path. The user's remote file system is mounted to their home directory, only after the automounter learns which user specified to mount their home directory, and determines the mount point location to use.
Although *indirect* mount points appear to exist, the `autofs` service creates them when the mount demand occurs, and deletes them again when the demand ended and the file system is unmounted.

Configure the Automounter Service

The process to configure an automount has many steps.

First, you must install the `autofs` and `nfs-utils` packages.

```
[user@host ~]$ sudo dnf install autofs nfs-utils
```

These packages contain all requirements to use the automounter for NFS exports.

Create a Master Map

Next, add a master map file to `/etc/auto.master.d`. This file identifies the base directory for mount points, and identifies the mapping file to create the automounts.

```
[user@host ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

The name of the master map file is mostly arbitrary (although typically meaningful), and it must have an extension of `.autofs` for the subsystem to recognize it. You can place multiple entries in a single master map file; alternatively, you can create multiple master map files, each with its own logically grouped entries.

Include the following content in the master map entry for indirectly mapped mounts:

```
/shares  /etc/auto.demo
```

This entry uses the `/shares` directory as the base for indirect automounts. The `/etc/auto.demo` file contains the mount details. Use an absolute file name. The `auto.demo` file must be created before starting the `autofs` service.

Create an Indirect Map

Now, create the mapping files. Each mapping file identifies the mount point, mount options, and source location to mount for a set of automounts.

```
[user@host ~]$ sudo vim /etc/auto.demo
```

The mapping file-naming convention is `/etc/auto.name`, where *name* reflects the content of the map.

```
work  -rw,sync  serverb:/shares/work
```

The format of an entry is *mount point*, *mount options*, and *source location*. This example shows an indirect mapping entry. Direct maps and indirect maps that use wildcards are covered later in this section.

Known as the *key* in the man pages, the `autofs` service automatically creates and removes the mount point. In this case, the fully qualified mount point is `/shares/work` (see the master map file). The `autofs` service creates and removes the `/shares` and `/shares/work` directories as needed.

In this example, the local mount point mirrors the server's directory structure. However, this mirroring is not required; the local mount point can have an arbitrary name. The `autofs` service does not enforce a specific naming structure on the client.

Mount options start with a dash character (`-`) and are comma-separated with no white space. The file-system mount options for manual mounting are also available when automounting. In this example, the automounter mounts the export with read/write access (`rw` option), and the server is synchronized immediately during write operations (`sync` option).

Useful automounter-specific options include `-fstype=` and `-strict`. Use `fstype` to specify the file-system type, for example `nfs4` or `xfs`, and use `strict` to treat errors when mounting file systems as fatal.

The source location for NFS exports follows the `host:/pathname` pattern, in this example `serverb:/shares/work`. For this automount to succeed, the NFS server, `serverb`, must *export* the directory with read/write access, and the user that requests access must have standard Linux file permissions on the directory. If `serverb` exports the directory with read/only access, then the client gets read/only access even if it requested read/write access.

Wildcards in an Indirect Map

When an NFS server exports multiple subdirectories within a directory, then the automounter can be configured to access any of those subdirectories with a single mapping entry.

Continuing the previous example, if `serverb:/shares` exports two or more subdirectories, and they are accessible with the same mount options, then the content for the `/etc/auto.demo` file might appear as follows:

```
*  -rw,sync  serverb:/shares/&
```

The mount point (or key) is an asterisk character (*), and the subdirectory on the source location is an ampersand character (&). Everything else in the entry is the same.

When a user attempts to access `/shares/work`, the `*` key (which is `work` in this example) replaces the ampersand in the source location and `serverb:/exports/work` is mounted. As with the indirect example, the `autofs` service creates and removes the `work` directory automatically.

Create a Direct Map

A direct map is used to map an NFS export to an absolute path mount point. Only one direct map file is necessary, and can contain any number of direct maps.

To use directly mapped mount points, the master map file might appear as follows:

```
/-  /etc/auto.direct
```

All direct map entries use `/-` as the base directory. In this case, the mapping file that contains the mount details is `/etc/auto.direct`.

The content for the `/etc/auto.direct` file might appear as follows:

```
/mnt/docs  -rw,sync  serverb:/shares/docs
```

The mount point (or key) is always an absolute path. The rest of the mapping file uses the same structure.

In this example, the `/mnt` directory exists, and the `autofs` service does not manage it. The `autofs` service creates and removes the full `/mnt/docs` directory automatically.

Start the Automounter Service

Lastly, use the `systemctl` command to start and enable the `autofs` service.

```
[user@host ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/lib
/systemd/system/autofs.service.
```

The Alternative `systemd.automount` Method

The systemd daemon can automatically create unit files for entries in the `/etc/fstab` file that include the `x-systemd.automount` option. Use the `systemctl daemon-reload` command after modifying an entry's mount options, to generate a new unit file, and then use the `systemctl start` *`unit`*`.automount` command to enable that automount configuration.

The naming of the unit is based on its mount location. For example, if the mount point is `/remote/finance`, then the unit file is named `remote-finance.automount`. The systemd daemon mounts the file system when the `/remote/finance` directory is initially accessed.

This method can be simpler than installing and configuring the `autofs` service. However, a `systemd.automount` unit can support only absolute path mount points, similar to `autofs` direct maps.

Guided Exercise: Automount Network-Attached Storage

In this exercise, you create direct-mapped and indirect-mapped automount-managed mount points that mount NFS file systems.

Outcomes

- Install required packages for the automounter.
- Configure direct and indirect automounter maps, with resources from a preconfigured NFSv4 server.
- Describe the difference between direct and indirect automounter maps.

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

This start script determines whether `servera` and `serverb` are reachable on the network. The script alerts you if those servers are not available. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. The script also creates users and groups that are needed on both `servera` and `serverb`.

```
[student@workstation ~]$ lab start netstorage-autofs
```

Instructions

An internet service provider uses a central server, `serverb`, to host shared directories with important documents that must be available on demand. When users log in to `servera`, they need access to the automounted shared directories.

The following list provides the environment characteristics for completing this exercise:

- The `serverb` machine exports the `/shares/indirect` directory, which in turn contains the `west`, `central`, and `east` subdirectories.
- The `serverb` machine also exports the `/shares/direct/external` directory.
- The `operators` group consists of the `operator1` and `operator2` users. They have read and write access to the `/shares/indirect/west`, `/shares/indirect/central`, and `/shares/indirect/east` exported directories.

- The `contractors` group consists of the `contractor1` and `contractor2` users. They have read and write access to the `/shares/direct/external` exported directory.
- The expected mount points for `servera` are `/external` and `/internal`.
- The `/shares/direct/external` exported directory is automounted on `servera` with a *direct* map on `/external`.
- The `/shares/indirect/west` exported directory is automounted on `servera` with an *indirect* map on `/internal/west`.
- The `/shares/indirect/central` exported directory is automounted on `servera` with an *indirect* map on `/internal/central`.
- The `/shares/indirect/east` exported directory is automounted on `servera` with an *indirect* map on `/internal/east`.
- All user passwords are set to `redhat`.
- The `nfs-utils` package is already installed.

1. Log in to `servera` and install the required packages.
    1. Log in to `servera` as the `student` user and switch to the `root` user.

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

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

    6. Install the `autofs` package.

    ```
    [root@servera ~]# dnf install autofs
    ...output omitted...
    Is this ok [y/N]: y
    ...output omitted...
    ```

    ```
    Complete!
    ```

2. Configure an automounter direct map on `servera` with exports from `serverb`. Create the direct map with files that are named `/etc/auto.master.d/direct.autofs` for the master map and `/etc/auto.direct` for the mapping file. Use the `/external` directory as the main mount point on `servera`.
    1. Test the NFS server and export before you configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/direct/external /mnt
[root@servera ~]# ls -l /mnt
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:15 README.txt
```

```
[root@servera ~]# umount /mnt
```

7. Create a master map file named `/etc/auto.master.d/direct.autofs`, insert the following content, and save the changes.

```
/-  /etc/auto.direct
```

8. Create a direct map file named `/etc/auto.direct`, insert the following content, and save the changes.

```
/external   -rw,sync,fstype=nfs4     serverb.lab.example.com:/shares/dir
ect/external
```

3. Configure an automounter indirect map on `servera` with exports from `serverb`. Create the indirect map with files that are named `/etc/auto.master.d/indirect.autofs` for the master map and `/etc/auto.indirect` for the mapping file. Use the `/internal` directory as the main mount point on `servera`.
   1. Test the NFS server and export before you configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/indirect /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

```
[root@servera ~]# umount /mnt
```

9. Create a master map file named `/etc/auto.master.d/indirect.autofs`, insert the following content, and save the changes.

```
/internal   /etc/auto.indirect
```

10. Create an indirect map file named `/etc/auto.indirect`, insert the following content, and save the changes.

```
*   -rw,sync,fstype=nfs4     serverb.lab.example.com:/shares/indirect/&
```

4. Start the `autofs` service on `servera`, and enable it to start automatically at boot time.
   1. Start and enable the `autofs` service on `servera`.

```
2. [root@servera ~]# systemctl enable --now autofs
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.servi
ce → /usr/lib/systemd/system/autofs.service.
```

5. Test the direct automounter map as the `contractor1` user. When done, exit from the `contractor1` user session on `servera`.
   1. Switch to the `contractor1` user.

```
2. [root@servera ~]# su - contractor1
```

```
[contractor1@servera ~]$
```

   3. List the `/external` mount point.

```
4. [contractor1@servera ~]$ ls -l /external
5. total 4
```

```
-rw-r--r--. 1 root contractors 22 Apr  7 23:34 README.txt
```

   6. Review the content and test the access to the `/external` mount point.

```
7. [contractor1@servera ~]$ cat /external/README.txt
8. ###External Folder###
9. [contractor1@servera ~]$ echo testing-direct > /external/testing.txt
10.[contractor1@servera ~]$ cat /external/testing.txt
```

```
testing-direct
```

   11. Exit from the `contractor1` user session.

```
[contractor1@servera ~]$ exit
logout
```

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

6. Test the indirect automounter map as the `operator1` user. When done, log out from `servera`.
   1. Switch to the `operator1` user.

```
[root@servera ~]# su - operator1
```

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

   3. List the `/internal` mount point.

```
[operator1@servera ~]$ ls -l /internal
```

```
total 0
```

Note

With an automounter indirect map, you must access each exported subdirectory for them to mount. With an automounter direct map, after you access the mapped mount point, you can immediately view and access the subdirectories and content in the exported directory.

   5. Test the `/internal/west` automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/west/
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/west/README.txt
###West Folder###
[operator1@servera ~]$ echo testing-1 > /internal/west/testing-1.txt
[operator1@servera ~]$ cat /internal/west/testing-1.txt
testing-1
[operator1@servera ~]$ ls -l /internal
total 0
```

```
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

16. Test the `/internal/central` automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/central
total 4
-rw-r--r--. 1 root operators 21 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/central/README.txt
###Central Folder###
[operator1@servera ~]$ echo testing-2 > /internal/central/testing-2.txt
[operator1@servera ~]$ cat /internal/central/testing-2.txt
testing-2
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
```

```
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

28. Test the `/internal/east` automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/east
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/east/README.txt
###East Folder###
[operator1@servera ~]$ echo testing-3 > /internal/east/testing-3.txt
[operator1@servera ~]$ cat /internal/east/testing-3.txt
testing-3
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
```

```
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

41. Test the `/external` automounter exported directory access.

```
42. [operator1@servera ~]$ ls -l /external
```

```
ls: cannot open directory '/external': Permission denied
```

43. Return to the `workstation` machine as the `student` user.

```
44. [operator1@servera ~]$ exit
45. logout
46. [root@servera ~]# exit
47. logout
48. [student@servera ~]$ exit
49. logout
```

```
Connection to servera closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish netstorage-autofs
```

This concludes the section.

Summary

- Mount and unmount an NFS share from the command line.
- Configure an NFS share to mount automatically at startup.
- Configure the automounter with direct and indirect maps, and describe their differences.

Chapter 10. Control the Boot Process

Abstract

Goal	Manage the boot process to control offered services and to troubleshoot and repair problems.
Objectives	• Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target. • Log in to a system and change the root password when the current root password is lost. • Manually repair file-system configuration or corruption issues that stop the boot process.
Sections	• Select the Boot Target (and Guided Exercise) • Reset the Root Password (and Guided Exercise) • Repair File-system Issues at Boot (and Guided Exercise)
Lab	Control the Boot Process

Select the Boot Target

Objectives

Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target.

Describe the Red Hat Enterprise Linux 9 Boot Process

Modern computer systems are complex combinations of hardware and software. Starting from an undefined, powered-down state to a running system with a login prompt requires many pieces of hardware and software to work together. The following list gives a high-level overview of the tasks for a physical `x86_64` system that boots Red Hat Enterprise Linux 9. The list for `x86_64` virtual machines is similar, except that the hypervisor handles some hardware-specific steps in software.

- The machine is powered on. The system firmware, either modern UEFI or earlier BIOS, runs a *Power On Self Test (POST)* and starts to initialize the hardware.

  The system BIOS or UEFI is configured by pressing a specific key combination, such as **F2**, early during the boot process.

- The UEFI boot firmware is configured by searching for a bootable device, which searches for or configures the *Master Boot Record (MBR)* on all disks.

  The system BIOS or UEFI configuration is configured by pressing a specific key combination, such as **F2**, early during the boot process.

- The system firmware reads a boot loader from disk and then passes control of the system to the boot loader. On a Red Hat Enterprise Linux 9 system, the boot loader is the *GRand Unified Bootloader version 2 (GRUB2)*.

  The `grub2-install` command installs GRUB2 as the boot loader on the disk for BIOS systems. Do not use the `grub2-install` command directly to install the UEFI boot loader. RHEL 9 provides a prebuilt `/boot/efi/EFI/redhat/grubx64.efi` file, which contains the required authentication signatures for a Secure Boot system. Executing `grub2-install` directly on a UEFI system generates a new `grubx64.efi` file without those required signatures. You can restore the correct `grubx64.efi` file from the `grub2-efi` package.

- GRUB2 loads its configuration from the `/boot/grub2/grub.cfg` file for BIOS, and from the `/boot/efi/EFI/redhat/grub.cfg` file for UEFI, and displays a menu to select which kernel to boot.

  GRUB2 is configured by using the `/etc/grub.d/` directory and the `/etc/default/grub` file. The `grub2-mkconfig` command generates the /boot/grub2/grub.cfg or /boot/efi/EFI/redhat/grub.cfg files for BIOS or UEFI, respectively.

- After you select a kernel, or the timeout expires, the boot loader loads the kernel and *initramfs* from disk and places them in memory.
  An initramfs image is an archive with the kernel modules for all the required hardware at boot, initialization scripts, and more. In Red Hat Enterprise Linux 9, the initramfs image contains a bootable root file system with a running kernel and a systemd unit.

  The initramfs image is configured by using the /etc/dracut.conf.d/ directory, the dracut command, and the lsinitrd command to inspect the initramfs file.

- The boot loader hands control over to the kernel, and passes in any specified options on the kernel command line in the boot loader, and the location of the initramfs image in memory.

  The boot loader is configured by using the /etc/grub.d/ directory, the /etc/default/grub file, and the grub2-mkconfig command to generate the /boot/grub2/grub.cfg file.

- The kernel initializes all hardware for which it can find a driver in the initramfs image, and then executes the /sbin/init script from the initramfs image as PID 1. On Red Hat Enterprise Linux 9, the /sbin/init script is a link to the systemd unit.

  The script is configured by using the kernel init= command-line parameter.

- The systemd unit from the initramfs image executes all units for the initrd.target target. This unit includes mounting the root file system on disk to the /sysroot directory.

  Configured by using the /etc/fstab file.

- The kernel switches (pivots) the root file system from the initramfs image to the root file system in the /sysroot directory. The systemd unit then re-executes itself by using the installed copy of the systemd unit on the disk.
- The systemd unit looks for a default target, which is either passed in from the kernel command line or is configured on the system. The systemd unit then starts (and stops) units to comply with the configuration for that target, and

solves dependencies between units automatically. A `systemd` unit is a set of units that the system activates to reach the intended state. These targets typically start a text-based login or a graphical login screen.

Configured by using the `/etc/systemd/system/default.target` file and the `/etc/systemd/system/` directory.

Figure 10.1: Boot process for BIOS-based and UEFI-based systems

Power Off and Reboot

To power off or reboot a running system from the command line, you can use the `systemctl` command.

The `systemctl poweroff` command stops all running services, unmounts all file systems (or remounts them read-only when they cannot be unmounted), and then powers down the system.

The `systemctl reboot` command stops all running services, unmounts all file systems, and then reboots the system.

You can also use the shorter version of these commands, `poweroff` and `reboot`, which are symbolic links to their `systemctl` equivalents.

Note

The `systemctl halt` and `halt` commands are also available to stop the system. Unlike the `poweroff` command, these commands do not power off the system; they bring down a system to a point where it is safe to power it off manually.

Select a Systemd Target

A `systemd` target is a set of `systemd` units that the system must start to reach an intended state. The following table lists the most important targets:

Table 10.1. Commonly Used Targets

Target	Purpose
`graphical.target`	This target supports multiple users, and provides graphical- and text-based logins.
`multi-user.target`	This target supports multiple users, and provides text-based logins only.
`rescue.target`	This target provides a single-user system to enable repairing your system.
`emergency.target`	This target starts the most minimal system for repairing your system when the `rescue.target` unit fails to start.

A target can be a part of another target. For example, the `graphical.target` unit includes the `multi-user.target` unit, which in turn depends on the `basic.target` unit and others. You can view these dependencies with the following command:

```
[user@host ~]$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   │ ├─paths.target
*   │ ├─slices.target
*   │ ├─sockets.target
*   │ ├─sysinit.target
*   │ │ ├─cryptsetup.target
*   │ │ ├─integritysetup.target
*   │ │ ├─local-fs.target
...output omitted...
```

To list the available targets, use the following command:

```
[user@host ~]$ systemctl list-units --type=target --all
  UNIT                      LOAD      ACTIVE   SUB    DESCRIPTION
  ---------------------------------------------------------------------------
  basic.target              loaded    active   active Basic System
...output omitted...
  cloud-config.target       loaded    active   active Cloud-config availability
  cloud-init.target         loaded    active   active Cloud-init target
  cryptsetup-pre.target     loaded    inactive dead   Local Encrypted Volumes (Pre)
  cryptsetup.target         loaded    active   active Local Encrypted Volumes
...output omitted...
```

Select a Target at Runtime

On a running system, administrators can switch to a different target by using the `systemctl isolate` command.

```
[root@host ~]# systemctl isolate multi-user.target
```

Isolating a target stops all services that the target does not require (and its dependencies), and starts any required services that are not yet started.

Not all targets can be isolated. You can isolate only targets where `AllowIsolate=yes` is set in their unit files. For example, you can isolate the graphical target, but not the `cryptsetup` target.

```
[user@host ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@host ~]$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

Set a Default Target

When the system starts, the `systemd` unit activates the `default.target` target. Normally, the default target the `/etc/systemd/system/` directory is a symbolic link to either the `graphical.target` or the `multi-user.target` targets. Instead of editing this symbolic link by hand, the `systemctl` command provides two subcommands to manage this link: `get-default` and `set-default`.

```
[root@host ~]# systemctl get-default
```

```
multi-user.target
[root@host ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/graphic
al.target.
[root@host ~]# systemctl get-default
graphical.target
```

Select a Different Target at Boot Time

To select a different target at boot time, append the `systemd.unit=`*`target`*`.target` option to the kernel command line from the boot loader.

For example, to boot the system into a rescue shell where you can change the system configuration with almost no services running, append the following option to the kernel command line from the boot loader:

```
systemd.unit=rescue.target
```

This configuration change affects only a single boot, and is a useful tool to troubleshoot the boot process.

To use this method to select a different target, use the following procedure:

1. Boot or reboot the system.
2. Interrupt the boot loader menu countdown by pressing any key (except **Enter**, which would initiate a normal boot).
3. Move the cursor to the kernel entry to start.
4. Press **e** to edit the current entry.
5. Move the cursor to the line that starts with `linux` which is the kernel command line.
6. Append `systemd.unit=`*`target`*`.target`, for example, `systemd.unit=emergency.target`.
7. Press **Ctrl**+**x** to boot with these changes.

Guided Exercise: Select the Boot Target

In this exercise, you determine the default target into which a system boots, and boot that system into other targets.

Outcomes

- Update the system default target and use a temporary target from the boot loader.

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

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start boot-selecting
```

Instructions

1. On the `workstation` machine, open a terminal and confirm that the default target is `graphical.target`.

```
2. [student@workstation ~]$ systemctl get-default
```

```
graphical.target
```

3. On the `workstation` machine, switch to the `multi-user` target manually without rebooting. Use the `sudo` command and if prompted, use `student` as the password.

```
4. [student@workstation ~]$ sudo systemctl isolate multi-user.target
```

```
[sudo] password for student: student
```

5. Access a text-based console. Use the **Ctrl**+**Alt**+**F1** key sequence by using the relevant button or menu entry. Log in as the `root` user by using `redhat` as the password.

   Note

Reminder: If you are using the terminal through a web page, then you can click the `Show Keyboard` icon in the menu on the right side of the screen under your web browser's URL bar.

```
workstation login: root
Password: redhat
[root@workstation ~]#
```

6. Configure the `workstation` machine to automatically boot into the `multi-user` target, and then reboot the `workstation` machine to verify. When done, change the default `systemd` target back to the `graphical` target.
   1. Set the default target.

```
2. [root@workstation ~]# systemctl set-default multi-user.target
3. Removed /etc/systemd/system/default.target.
```

```
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/s
ystem/multi-user.target.
```

   4. Reboot the `workstation` machine. After reboot, the system presents a text-based console and not a graphical login screen.

```
[root@workstation ~]# systemctl reboot
```

   5. Log in as the `root` user.

```
6. workstation login: root
7. Password: redhat
8. Last login: Thu Mar 28 14:50:53 on tty1
```

```
[root@workstation ~]#
```

   9. Set the default `systemd` target back to the `graphical` target.

```
10. [root@workstation ~]# systemctl set-default graphical.target
11. Removed /etc/systemd/system/default.target.
```

```
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/s
ystem/graphical.target.
```

This step concludes the first part of the exercise, where you practice setting the default `systemd` target.

7. In this second part of the exercise, you practice by using rescue mode to recover the system.

   Access the boot loader by rebooting `workstation` again. From within the boot loader menu, boot into the `rescue` target.

   1. Initiate the reboot.

      ```
      [root@workstation ~]# systemctl reboot
      ```

   2. When the boot loader menu appears, press any key to interrupt the countdown (except **Enter**, which would initiate a normal boot).
   3. Use the cursor keys to highlight the default boot loader entry.
   4. Press **e** to edit the current entry.
   5. Using the cursor keys, navigate to the line that starts with `linux`.
   6. Press **End** to move the cursor to the end of the line.
   7. Append `systemd.unit=rescue.target` to the end of the line.

      Note

      If it is difficult for you to read the text in the console, then consider changing the resolution when you edit the kernel line in the boot loader entry.

      To change the console resolution, add either `video=640x480` or `vga=ask` to the line that starts with the `linux` word, after `sysemd.unit=rescue.target`. With the `video=640x480` argument, the text console should be displayed at approximately an 80-column width. If you use `vga=ask` instead, then you can choose the resolution that best suits your environment at boot time.

   8. Press **Ctrl**+**x** to boot by using the modified configuration.
   9. Log in to rescue mode. You might need to press **Enter** to get a clean prompt.

```
10.Give root password for maintenance
```

```
11. (or press Control-D to continue): redhat
```

```
[root@workstation ~]#
```

8. Confirm that in rescue mode, the root file system is in read/write mode.

```
9.  [root@workstation ~]# mount
10. ...output omitted...
11. /dev/vda4 on / type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize
    =32k,noquota)
```

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

12. Press **Ctrl**+**d** to continue with the boot process.

    The system presents a graphical login. Log in as the `student` user.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-selecting
```

This concludes the section.

Reset the Root Password

Objectives

Log in to a system and change the root password when the current root password is lost.

Reset the Root Password from the Boot Loader

One task that every system administrator should be able to accomplish is resetting a lost `root` password. This task is trivial if the administrator is still logged in, either as an unprivileged user but with full `sudo` access, or as `root`. This task becomes slightly more involved when the administrator is not logged in.

Several methods exist to set a new `root` password. A system administrator could, for example, boot the system by using a Live CD, mount the root file system from there, and edit `/etc/shadow`. This section explores a method that does not require the use of external media.

On Red Hat Enterprise Linux 9, the scripts that run from the `initramfs` image can be paused at certain points, to provide a `root` shell, and then continue when that shell exits. This script is mostly meant for debugging, and also to reset a lost `root` password.

Starting from Red Hat Enterprise Linux 9, if you install your system from a DVD, then the default kernel asks for the `root` password when you try to enter maintenance mode. Thus, to reset a lost `root` password, you must use the rescue kernel.

To access that `root` shell, follow these steps:

1. Reboot the system.
2. Interrupt the boot-loader countdown by pressing any key, except **Enter**.
3. Move the cursor to the rescue kernel entry to boot (the entry with the *rescue* word in its name).
4. Press **e** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with `linux`).
6. Append `rd.break`. With that option, the system breaks just before the system hands control from the `initramfs` image to the actual system.
7. Press **Ctrl**+**x** to boot with the changes.
8. Press **Enter** to perform maintenance when prompted.

At this point, the system presents a `root` shell, and the root file system on the disk is mounted read-only on `/sysroot`. Because troubleshooting often requires modifying the root file system, you must remount the root file system as read/write. The following step shows how the `remount,rw` option to the `mount` command remounts the file system where the new option (`rw`) is set.

Important

Because the system has not yet enabled SELinux, any file that you create does not have SELinux context. Some tools, such as the `passwd` command, first create a temporary file, and then replace it with the file that is intended for editing, which effectively creates a file without SELinux context. For this reason, when you use the `passwd` command with `rd.break`, the `/etc/shadow` file does not receive SELinux context.

To reset the `root` password, use the following procedure:

1. Remount `/sysroot` as read/write.

```
sh-5.1# mount -o remount,rw /sysroot
```

2. Switch into a `chroot` jail, where `/sysroot` is treated as the root of the file-system tree.

```
sh-5.1# chroot /sysroot
```

3. Set a new `root` password.

```
sh-5.1# passwd root
```

4. Ensure that all unlabeled files, including `/etc/shadow` at this point, get relabeled during boot.

```
sh-5.1# touch /.autorelabel
```

5. Type `exit` twice. The first command exits the `chroot` jail, and the second command exits the `initramfs` debug shell.

At this point, the system continues booting, performs a full SELinux relabeling, and then reboots again.

Recovery of a Cloud Image-based System

If your system was installed by deploying and modifying one of the official cloud images instead of using the installer, then some system configuration aspects might differ for the boot process.

The procedure to use the `rd.break` option to get a root shell is similar to the previously outlined procedure, with some minor changes.

If your system was deployed from a Red Hat Enterprise Linux cloud image, then your boot menu does not have a rescue kernel by default. However, you can use the default kernel to enter maintenance mode by using the `rd.break` option without entering the root password.

The kernel prints boot messages and displays the root prompt on the system console. Prebuilt images might have multiple `console=` arguments on the kernel command line in the bootloader. Even though the system sends the kernel messages to all the consoles, the root shell that the `rd.break` option sets up uses the last console that is specified on the command line. If you do not get your prompt, then you might temporarily reorder the `console=` arguments when you edit the kernel command line in the boot loader.

Inspect Logs

Looking at the logs of previously failed boots can be useful. If the system journals persist across reboots, then you can use the `journalctl` tool to inspect those logs.

Remember that by default, the system journals are kept in the `/run/log/journal` directory, and the journals are cleared when the system reboots. To store journals in the `/var/log/journal` directory, which persists across reboots, set the `Storage` parameter to `persistent` in the `/etc/systemd/journald.conf` file.

```
[root@host ~]# vim /etc/systemd/journald.conf
...output omitted...
[Journal]
Storage=persistent
...output omitted...
[root@host ~]# systemctl restart systemd-journald.service
```

To inspect the logs of a previous boot, use the `journalctl` command `-b` option. Without any arguments, the `journalctl` command `-b` option displays only messages since the last boot. With a negative number as an argument, it displays the logs of previous boots.

```
[root@host ~]# journalctl -b -1 -p err
```

This command shows all messages that are rated as an error or worse from the previous boot.

Repair Systemd Boot Issues

To troubleshoot service startup issues at boot time, Red Hat Enterprise Linux 8 and later versions provide the following tools:

Enable the Early Debug Shell

By enabling the `debug-shell` service with the `systemctl enable debug-shell.service` command, the system spawns a `root` shell on `TTY9` (**Ctrl**+**Alt**+**F9**) early during the boot sequence. This shell is automatically logged in as `root`, so that administrators can debug the system when the operating system is still booting.

Warning

Disable the `debug-shell.service` service when you are done debugging, because it leaves an unauthenticated `root` shell open to anyone with local console access.

Alternatively, to activate the debug shell during the boot by using the GRUB2 menu, follow these steps:

1. Reboot the system.
2. Interrupt the boot-loader countdown by pressing any key, except **Enter**.
3. Move the cursor to the kernel entry to boot.
4. Press **e** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with `linux`).
6. Append `systemd.debug-shell`. With this parameter, the system boots into the debug shell.
7. Press **Ctrl**+**x** to boot with the changes.

Use the Emergency and Rescue Targets

By appending either `systemd.unit=rescue.target` or `systemd.unit=emergency.target` to the kernel command line from the boot loader, the system enters into a rescue or emergency shell instead of starting normally. Both of these shells require the `root` password.

The emergency target keeps the root file system mounted read-only, while the rescue target waits for the `sysinit.target` unit to complete, so that more of the system is initialized, such as the logging service or the file systems. The root user at this point cannot change `/etc/fstab` until the drive is remounted in a read write state with the `mount -o remount,rw /` command.

Administrators can use these shells to fix any issues that prevent the system from booting normally, for example, a dependency loop between services, or an incorrect entry in `/etc/fstab`. Exiting from these shells continues with the regular boot process.

Identify Stuck Jobs

During startup, `systemd` spawns various jobs. If some of these jobs cannot complete, then they block other jobs from running. To inspect the current job list, administrators can use the `systemctl list-jobs` command. Any jobs that are listed as running must complete before the jobs that are listed as waiting can continue.

Guided Exercise: Reset the Root Password

In this exercise, you reset the `root` password on a system.

Outcomes

- Reset the lost `root` user password.

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

This command runs a start script that determines whether the `servera` machine is reachable on the network. It also resets the `root` password to a random string and sets a higher time-out for the GRUB2 menu.

```
[student@workstation ~]$ lab start boot-resetting
```

Instructions

1. Reboot `servera`, and interrupt the countdown in the boot-loader menu.
    1. Locate the icon for the `servera` console, as appropriate for your classroom environment, and then open the console.

    Send **Ctrl**+**Alt**+**Del** to your system by using the relevant button or menu entry.

    2. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.
2. Edit the rescue kernel boot-loader entry, in memory, to abort the boot process just after the kernel mounts all the file systems, but before it hands over control to `systemd`.
    1. Use the cursor keys to highlight the rescue kernel entry (the one with the *rescue* word in its name).
    2. Press **e** to edit the current entry.
    3. Use the cursor keys to navigate to the line that starts with `linux`.
    4. Press **End** to move the cursor to the end of the line.
    5. Append `rd.break` to the end of the line.

       Note

       If it is difficult for you to see the text in the console, then consider changing the resolution when editing the kernel line in the boot loader entry.

       To change the console resolution, add either `video=640x480` or `vga=ask` on the line that starts with the `linux` word, after `rd.break`. For most consoles, a resolution of `640x480` is enough. By using `vga=ask`, you can choose a more suitable resolution for your environment.

    6. Press **Ctrl**+**x** to boot by using the modified configuration.
3. Press **Enter** to perform maintenance. At the `sh-5.1#` prompt, remount the `/sysroot` file-system as read/write, and then use the `chroot` command to enter a `chroot` jail at `/sysroot`.

```
sh-5.1# mount -o remount,rw /sysroot
...output omitted...
```

```
sh-5.1# chroot /sysroot
```

6. Change the `root` password back to `redhat`.

```
sh-5.1# passwd root
Changing password for user root.
```

```
9.  New password: redhat
10. BAD PASSWORD: The password is shorter than 8 characters
11. Retype new password: redhat
```

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

12. Configure the system to automatically perform a full SELinux relabeling after booting. This step is necessary because the `passwd` command re-creates the `/etc/shadow` file without an SELinux context.

```
sh-5.1# touch /.autorelabel
```

13. Type `exit` twice to continue booting your system as usual. The system runs an SELinux relabel operation, and then reboots automatically. When the system is up, verify your work by logging in as `root` at the console.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-resetting
```

This concludes the section.

Repair File-system Issues at Boot

Objectives

Manually repair file-system configuration or corruption issues that stop the boot process.

File-system Issues

During the boot process, the `systemd` service mounts the persistent file systems that are defined in the `/etc/fstab` file.

Errors in the `/etc/fstab` file or corrupted file systems can block a system from completing the boot process. In some failure scenarios, the system breaks out of the boot process and opens an emergency shell that requires the `root` user password.

The following list describes some common file-system mounting issues when parsing the `/etc/fstab` file during the boot process:

Corrupted file system

The `systemd` service attempts to repair the file system. If the problem cannot be automatically repaired, then the system opens an emergency shell.

Nonexistent device or UUID

The `systemd` service times out waiting for the device to become available. If the device does not respond, then the system opens an emergency shell.

Note

If the mount point is not present, then Red Hat Enterprise Linux 9 automatically creates it during the boot process.

Repair File-system Issues at Boot

To access a system that cannot complete booting because of file-system issues, the `systemd` architecture provides an `emergency` boot target, which opens an emergency shell that requires the `root` password for access.

The next example demonstrates the boot process output when the system finds a file-system issue and switches to the `emergency` target:

```
...output omitted...
[*     ] A start job is running for /dev/vda2 (27s / 1min 30s)
[ TIME ] Timed out waiting for device /dev/vda2.
[DEPEND] Dependency failed for /mnt/mountfolder
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Mark need to relabel after reboot.
```

```
...output omitted...
[  OK  ] Started Emergency Shell.
[  OK  ] Reached target Emergency Mode.
...output omitted...
Give root password for maintenance
(or press Control-D to continue):
```

The `systemd` daemon failed to mount the `/dev/vda2` device and timed out. Because the device is not available, the system opens an emergency shell for maintenance access.

To repair file-system issues when your system opens an emergency shell, first locate the errant file system, and then find and repair the fault. Now reload the `systemd` configuration to retry the automatic mounting.

Use the `mount` command to find which file systems are currently mounted by the `systemd` daemon.

```
[root@host ~]# mount
...output omitted...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

If the root file system is mounted with the `ro` (read-only) option, then you cannot edit the `/etc/fstab` file. Temporarily remount the root file system with the `rw` (read/write) option, if necessary, before opening the `/etc/fstab` file. With the remount option, an in-use file system can change its mount parameters without unmounting the file system.

```
[root@host ~]# mount -o remount,rw /
```

Try to mount all the file systems that are listed in the `/etc/fstab` file by using the `mount --all` option. This option mounts processes on every file-system entry, but skips those file systems that are already mounted. The command displays any errors that occur when mounting a file system.

```
[root@host ~]# mount --all
mount: /mnt/mountfolder: mount point does not exist.
```

In this scenario, where the `/mnt/mountfolder` mount directory does not exist, create the `/mnt/mountfolder` directory before reattempting the mount. Other error messages can occur, including typing errors in the entries, or wrong device names or UUIDs.

After you corrected all issues in the `/etc/fstab` file, inform the `systemd` daemon to register the new `/etc/fstab` file by using the `systemctl daemon-reload` command. Then, reattempt mounting all the entries.

```
[root@host ~]# systemctl daemon-reload
[root@host ~]# mount --all
```

Note

The `systemd` service processes the `/etc/fstab` file by transforming each entry into a `.mount` type `systemd` unit configuration and then starting the unit as a service. The `daemon-reload` option requests the `systemd` daemon to rebuild and reload all unit configurations.

If the `mount --all` command succeeds without further errors, then the final test is to verify that file-system mounting is successful during a system boot. Reboot the system and wait for the boot to complete normally.

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

For quick testing in the `/etc/fstab` file, use the `nofail` mount entry option. Using the `nofail` option in an `/etc/fstab` entry enables the system to boot even if that file-system mount is unsuccessful. This option must not be used with production file systems that must always mount. With the `nofail` option, an application could start when its file-system data is missing, with possibly severe consequences.

Repair File-system Issues at Boot

Objectives

Manually repair file-system configuration or corruption issues that stop the boot process.

File-system Issues

During the boot process, the `systemd` service mounts the persistent file systems that are defined in the `/etc/fstab` file.

Errors in the `/etc/fstab` file or corrupted file systems can block a system from completing the boot process. In some failure scenarios, the system breaks out of the boot process and opens an emergency shell that requires the `root` user password.

The following list describes some common file-system mounting issues when parsing the `/etc/fstab` file during the boot process:

Corrupted file system
: The `systemd` service attempts to repair the file system. If the problem cannot be automatically repaired, then the system opens an emergency shell.

Nonexistent device or UUID
: The `systemd` service times out waiting for the device to become available. If the device does not respond, then the system opens an emergency shell.

Note

If the mount point is not present, then Red Hat Enterprise Linux 9 automatically creates it during the boot process.

Repair File-system Issues at Boot

To access a system that cannot complete booting because of file-system issues, the `systemd` architecture provides an `emergency` boot target, which opens an emergency shell that requires the `root` password for access.

The next example demonstrates the boot process output when the system finds a file-system issue and switches to the `emergency` target:

```
...output omitted...
[*     ] A start job is running for /dev/vda2 (27s / 1min 30s)
```

```
[ TIME ] Timed out waiting for device /dev/vda2.
[DEPEND] Dependency failed for /mnt/mountfolder
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Mark need to relabel after reboot.
...output omitted...
[  OK  ] Started Emergency Shell.
[  OK  ] Reached target Emergency Mode.
...output omitted...
Give root password for maintenance
(or press Control-D to continue):
```

The `systemd` daemon failed to mount the `/dev/vda2` device and timed out. Because the device is not available, the system opens an emergency shell for maintenance access.

To repair file-system issues when your system opens an emergency shell, first locate the errant file system, and then find and repair the fault. Now reload the `systemd` configuration to retry the automatic mounting.

Use the `mount` command to find which file systems are currently mounted by the `systemd` daemon.

```
[root@host ~]# mount
...output omitted...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

If the root file system is mounted with the `ro` (read-only) option, then you cannot edit the `/etc/fstab` file. Temporarily remount the root file system with the `rw` (read/write) option, if necessary, before opening the `/etc/fstab` file. With the remount option, an in-use file system can change its mount parameters without unmounting the file system.

```
[root@host ~]# mount -o remount,rw /
```

Try to mount all the file systems that are listed in the `/etc/fstab` file by using the `mount --all` option. This option mounts processes on every file-system entry, but skips those file systems that are already mounted. The command displays any errors that occur when mounting a file system.

```
[root@host ~]# mount --all
mount: /mnt/mountfolder: mount point does not exist.
```

In this scenario, where the `/mnt/mountfolder` mount directory does not exist, create the `/mnt/mountfolder` directory before reattempting the mount. Other error messages can occur, including typing errors in the entries, or wrong device names or UUIDs.

After you corrected all issues in the `/etc/fstab` file, inform the `systemd` daemon to register the new `/etc/fstab` file by using the `systemctl daemon-reload` command. Then, reattempt mounting all the entries.

```
[root@host ~]# systemctl daemon-reload
[root@host ~]# mount --all
```

Note

The `systemd` service processes the `/etc/fstab` file by transforming each entry into a `.mount` type `systemd` unit configuration and then starting the unit as a service. The `daemon-reload` option requests the `systemd` daemon to rebuild and reload all unit configurations.

If the `mount --all` command succeeds without further errors, then the final test is to verify that file-system mounting is successful during a system boot. Reboot the system and wait for the boot to complete normally.

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

For quick testing in the `/etc/fstab` file, use the `nofail` mount entry option. Using the `nofail` option in an `/etc/fstab` entry enables the system to boot even if that file-system mount is unsuccessful. This option must not be used with production file systems that must always mount. With the `nofail` option, an application could start when its file-system data is missing, with possibly severe consequences.

Summary

- The `systemctl reboot` and `systemctl poweroff` commands reboot and power down a system, respectively.
- The `systemctl isolate *target-name*.target` command switches to a new target at runtime.
- The `systemctl get-default` and `systemctl set-default` commands can query and set the default target.
- You can use the `rd.break` option on the kernel command line to interrupt the boot process before control is handed over from the `initramfs` image. The root file system is mounted read-only under `/sysroot`.
- The emergency target can diagnose and fix file-system issues.

Chapter 11. Manage Network Security

Abstract

Goal	Control network connections to services with the system firewall and SELinux rules.
Objectives	• Accept or reject network connections to system services with `firewalld` rules. • Verify that network ports have the correct SELinux type for services to bind to them.
Sections	• Manage Server Firewalls (and Guided Exercise)

	• Control SELinux Port Labeling (and Guided Exercise)
Lab	Manage Network Security

Manage Server Firewalls

Objectives

Accept or reject network connections to system services with `firewalld` rules.

Firewall Architecture Concepts

The Linux kernel provides the `netfilter` framework for network traffic operations such as packet filtering, network address translation, and port translation. The `netfilter` framework includes *hooks* for kernel modules to interact with network packets as they traverse a system's network stack. Fundamentally, `netfilter` hooks are kernel routines that intercept events (for example, a packet that enters an interface) and run other related routines (for example, firewall rules).

The nftables Framework

The `nftables` packet classification framework builds on the `netfilter` framework to apply firewall rules to network traffic. In Red Hat Enterprise Linux 9, the `nftables` framework is the system firewall core, and it replaces the deprecated `iptables` framework.

The `nftables` framework provides many advantages over `iptables`, including improved usability and more efficient rule sets. For example, the `iptables` framework required a rule for each protocol, but `nftables` rules can apply to both IPv4 and IPv6 traffic simultaneously. The `iptables` framework required using different tools, such as `iptables`, `ip6tables`, `arptables`, and `ebtables`, for each protocol. By contrast, the `nftables` framework uses the single `nft` user-space utility to manage all protocols through a single interface.

Note

Convert earlier `iptables` configuration files into their `nftables` equivalents by using the `iptables-translate` and `ip6tables-translate` utilities.

The firewalld Service

The `firewalld` service is a dynamic firewall manager, and is the recommended front end to the `nftables` framework. The Red Hat Enterprise Linux 9 distribution includes the `firewalld` package.

The `firewalld` service simplifies firewall management by classifying network traffic into *zones*. A network packet's assigned zone depends on criteria such as the source IP address of the packet or the incoming network interface. Each zone has its own list of ports and services that are either open or closed.

Note

For laptops or other machines that often change networks, the `NetworkManager` service can automatically set the firewall zone for a connection. This service is useful when switching between home, work, and public wireless networks. A user might want their system's `sshd` service to be reachable when connected to their home or corporate networks, but not when connected to a public wireless network in the local coffee shop.

The `firewalld` service ensures the source address for every incoming packet into the system. If that source address is assigned to a specific zone, then the rules for that zone apply. If the source address is not assigned to a zone, then the `firewalld` service associates the packet with the zone for the incoming network interface, and the rules for that zone apply. If the network interface is not associated with a zone, then the `firewalld` service sends the packet to the default zone.

The default zone is not a separate zone but rather an assigned designation to an existing zone. Initially, the `firewalld` service designates the `public` zone as default, and maps the `lo` loopback interface to the `trusted` zone.

Most zones allow traffic through the firewall if it matches a list of particular ports and protocols, such as `631/udp`, or a predefined service configuration, such as `ssh`. Normally, if the traffic does not match a permitted port and protocol or service, then it is rejected. The `trusted` zone, which permits all traffic by default, is an exception.

Predefined Zones

The `firewalld` service uses predefined zones, which you can customize. By default, all zones allow any incoming traffic that is part of an existing session that system initiated, and also allow all outgoing traffic. The following table details the initial zone configuration.

Table 11.1. 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` predefined services.
`internal`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `mdns`, `ipp-client`, `samba-client`, or `dhcpv6-client` predefined 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` predefined services.
`public`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` or `dhcpv6-client` predefined services. *The default zone for newly added network interfaces.*
`external`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` predefined service. Outgoing IPv4 traffic that is forwarded through this zone is *masqueraded* to appear that it originated from the IPv4 address of the outgoing network interface.
`dmz`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` predefined service.
`block`	Reject all incoming traffic unless related to outgoing traffic.
`drop`	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of available predefined zones and their intended use, see the `firewalld.zones`(5) man page.

Predefined Services

The `firewalld` service includes predefined configurations for common services, to simplify setting firewall rules. For example, instead of researching the relevant ports for an NFS server, use the predefined `nfs` configuration create rules for the correct ports and protocols. The following table lists some predefined service configurations that might be active in your default `firewalld` zone.

Table 11.2. Selected Predefined Firewalld Services

Service name	Configuration
`ssh`	Local SSH server. Traffic to 22/tcp.
`dhcpv6-client`	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network.
`ipp-client`	Local IPP printing. Traffic to 631/udp.
`samba-client`	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
`mdns`	Multicast DNS (mDNS) local-link name resolution. Traffic to 5353/udp to the 224.0.0.251 (IPv4) or ff02::fb (IPv6) multicast addresses.
`cockpit`	Red Hat Enterprise Linux web-based interface for managing and monitoring your local and remote system. Traffic to 9090 port.

The `firewalld` package includes many predefined service configurations. You can list the services with the `firewall-cmd --get-services` command.

```
[root@host ~]# firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule amanda-client amanda-k5-client amqp amqps
apcupsd audit bacula bacula-client bb bgp bitcoin bitcoin-rpc bitcoin-testnet
bitcoin-testnet-rpc bittorrent-lsd ceph ceph-mon cfengine cockpit collectd
...output omitted...
```

If the predefined service configurations are not appropriate for your scenario, then you can manually specify the required ports and protocols. You can use the web console graphical interface to review predefined services and manually define more ports and protocols.

Configure the firewalld Daemon

The following list shows two common ways that system administrators use to interact with the `firewalld` service:

- The web console graphical interface
- The `firewall-cmd` command-line tool

Configure Firewall Services with the Web Console

To manage firewall services with the web console, you must log in and escalate privileges. You can escalate privileges by clicking the **Limited access** or **Turn on administrative access** buttons. Then, enter your password when prompted. The administrative mode elevates privileges based on your user's sudo configuration. As a security reminder, remember to toggle back to limited access mode after you perform the system task that requires administrative privileges.

Click the **Networking** option in the left navigation menu to display the **Firewall** section in the main networking page. Click the **Edit rules and zones** button zones to navigate to the **Firewall** page.

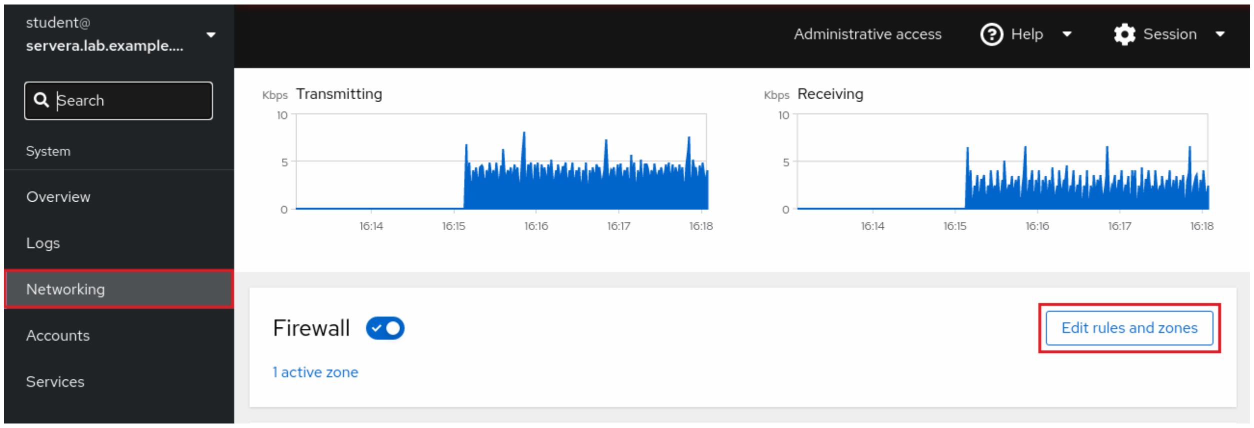
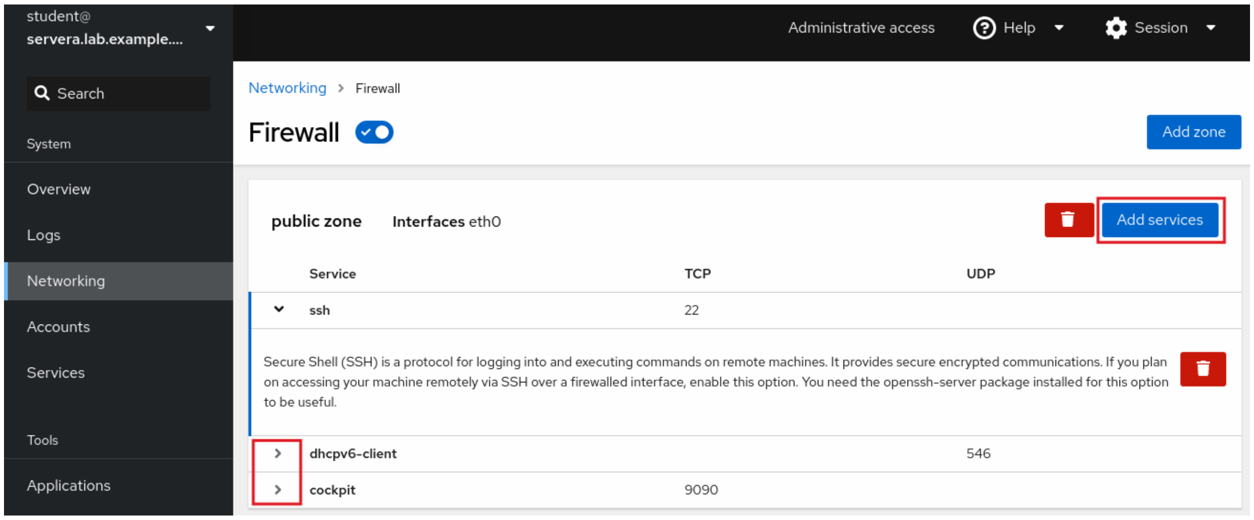


Figure 11.1: The web console networking page

The **Firewall** page displays active zones and their allowed services. Click the arrow (**>**) button to the left of a service name to view its details. To add a service to a zone, click the **Add services** button in the upper right corner of the applicable zone.



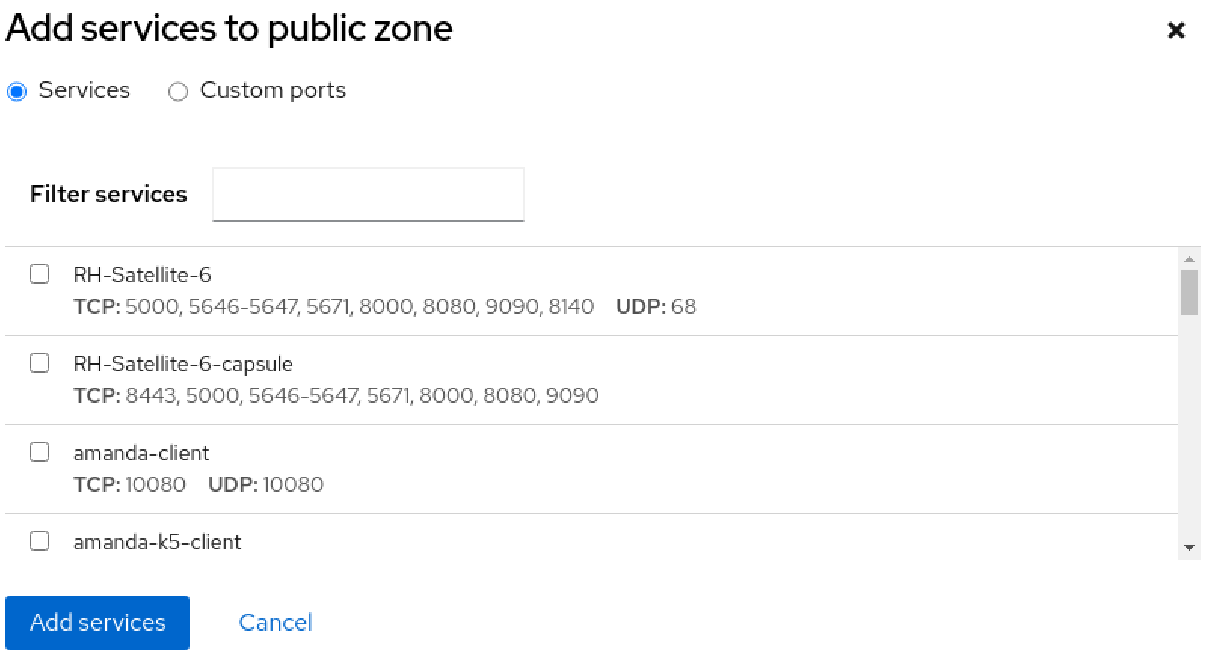
public zone Interfaces eth0

Service	TCP	UDP
ssh	22	
dhcpv6-client		546
cockpit	9090	

Secure Shell (SSH) is a protocol for logging into and executing commands on remote machines. It provides secure encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface, enable this option. You need the openssh-server package installed for this option to be useful.

Figure 11.2: The web console firewall page

The **Add Services** page displays the available predefined services.



RH-Satellite-6
TCP: 5000, 5646-5647, 5671, 8000, 8080, 9090, 8140 UDP: 68

RH-Satellite-6-capsule
TCP: 8443, 5000, 5646-5647, 5671, 8000, 8080, 9090

amanda-client
TCP: 10080 UDP: 10080

amanda-k5-client

Figure 11.3: The web console add services menu

To select a service, scroll through the list or enter a selection in the **Filter services** text box. In the following example, the `http` string filters the options to web-related services. Select the checkbox to the left of the service to allow it through the firewall. Click the **Add services** button to complete the process.

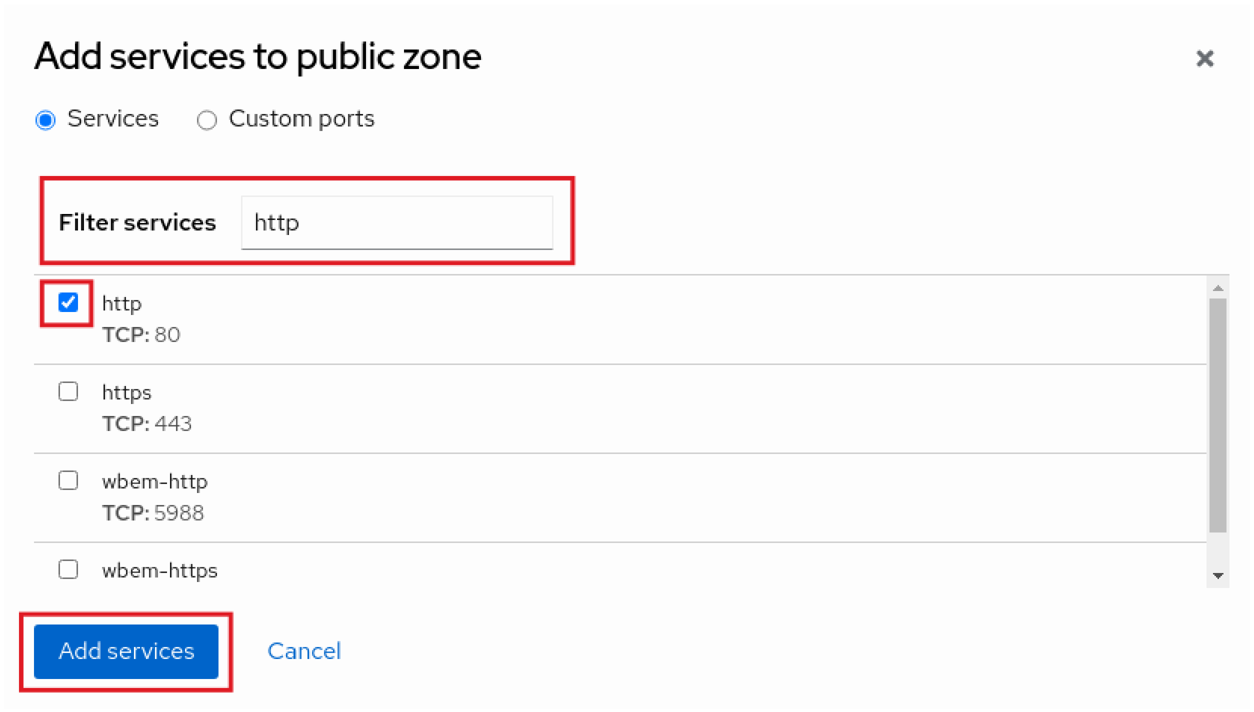
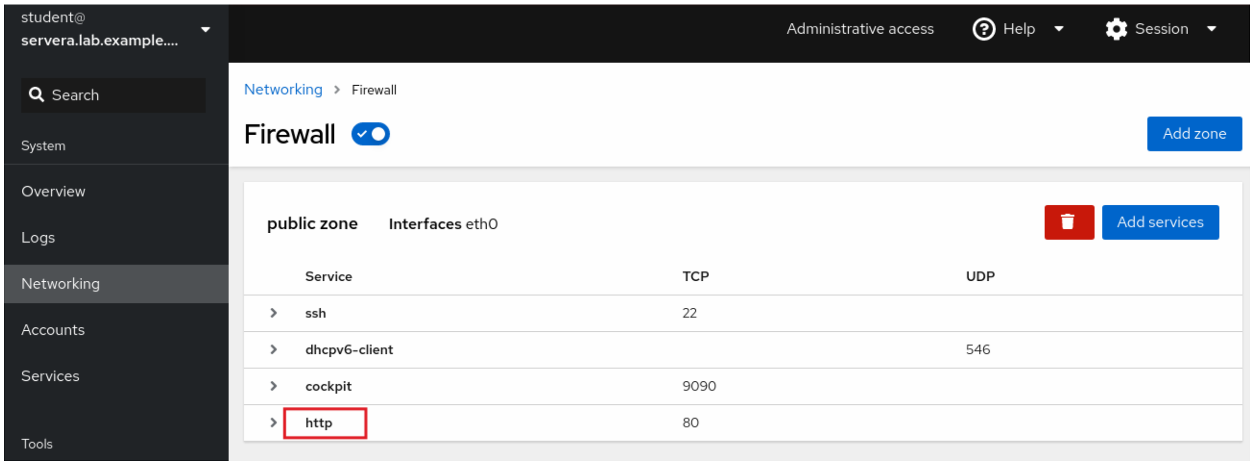


Figure 11.4: The web console add services menu options

The interface returns to the **Firewall** page, where you can review the updated allowed services list.



Service	TCP	UDP
ssh	22	
dhcpv6-client		546
cockpit	9090	
http	80	

Figure 11.5: The web console firewall overview

Configure the Firewall from the Command Line

The `firewall-cmd` command interfaces with the `firewalld` daemon. It is installed as part of the `firewalld` package, and is available for administrators who prefer to work on the command line, for working on systems without a graphical environment, or for scripting a firewall setup.

The following table lists often use `firewall-cmd` commands, along with an explanation. Most commands work on the *runtime* configuration, unless the `--permanent` option is specified. If the `--permanent` option is specified, then you must activate the setting by also running the `firewall-cmd --reload` command, which reads the current permanent configuration and applies it as the new runtime configuration. Many of the listed commands take the `--zone=`*`ZONE`* option to find which zone they affect. Where a netmask is required, use CIDR notation, such as 192.168.1/24.

firewall-cmd commands	**Explanation**
`--get-default-zone`	Query the current default zone.
`--set-default-zone=`*`ZONE`*	Set the default zone. This default zone changes both the runtime and the permanent configuration.
`--get-zones`	List all available zones.
`--get-active-zones`	List all zones that are currently in use (with an interface or source that is tied to them), along with their interface and source information.
`--add-source=`*`CIDR`* `[--zone=`*`ZONE`*`]`	Route all traffic from the IP address or network/netmask to the specified zone. If no `--zone=` option is provided, then the default zone is used.
`--remove-source=`*`CIDR`* `[--zone=`*`ZONE`*`]`	Remove the rule that routes all traffic from the zone that comes from the IP address or network. If no `--zone=` option is provided, then the default zone is used.
`--add-interface=`*`INTERFACE`* `[--zone=`*`ZONE`*`]`	Route all traffic from *`INTERFACE`* to the specified zone. If no `--zone=` option is provided, then the default zone is used.
`--change-interface=`*`INTERFACE`* `[--zone=`*`ZONE`*`]`	Associate the interface with ZONE instead of its current zone. If no `--zone=` option is provided, then the default zone is used.
`--list-all [--zone=`*`ZONE`*`]`	List all configured interfaces, sources, services, and ports for *`ZONE`*. If no `--zone=` option is provided, then the default zone is used.

firewall-cmd commands	**Explanation**
`--list-all-zones`	Retrieve all information for all zones (interfaces, sources, ports, and services).
`--add-service=`*`SERVICE`* `[--zone=`*`ZONE`*`]`	Allow traffic to *`SERVICE`*. If no `--zone=` option is provided, then the default zone is used.
`--add-port=`*`PORT/PROTOCOL`* `[--zone=`*`ZONE`*`]`	Allow traffic to the *`PORT/PROTOCOL`* ports. If no `--zone=` option is provided, then the default zone is used.
`--remove-service=`*`SERVICE`* `[--zone=`*`ZONE`*`]`	Remove *`SERVICE`* from the allowed list for the zone. If no `--zone=` option is provided, then the default zone is used.
`--remove-port=`*`PORT/PROTOCOL`* `[--zone=`*`ZONE`*`]`	Remove the *`PORT/PROTOCOL`* ports from the allowed list for the zone. If no `--zone=` option is provided, then the default zone is used.
`--reload`	Drop the runtime configuration and apply the persistent configuration.

The following example sets the default zone to `dmz`, assigns all traffic coming from the `192.168.0.0/24` network to the `internal` zone, and opens the network ports for the `mysql` service on the `internal` zone.

```
[root@host ~]# firewall-cmd --set-default-zone=dmz
[root@host ~]# firewall-cmd --permanent --zone=internal \
--add-source=192.168.0.0/24
[root@host ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@host ~]# firewall-cmd --reload
```

As another example, to add all the incoming traffic from the `172.25.25.11` single IPv4 address to the `public` zone, use the following commands:

```
[root@host ~]# firewall-cmd --permanent --zone=public \
--add-source=172.25.25.11/32
[root@host ~]# firewall-cmd --reload
```

Note

For situations where the basic syntax is not enough, you can add *rich-rules* to write complex rules. If even the rich-rules syntax is not enough, then you can also use Direct Configuration rules (which use raw `nft` syntax mixed in with `firewalld` rules). These advanced configurations are beyond the scope of this chapter.

Guided Exercise: Manage Server Firewalls

In this exercise, you control access to system services by adjusting system firewall rules with the `firewalld` service.

Outcomes

- Configure firewall rules to control access to services.

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

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

```
[student@workstation ~]$ lab start netsecurity-firewalls
```

Instructions

1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

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

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

6. Install the `httpd` and `mod_ssl` packages. These packages provide the Apache web server and the necessary extensions for the web server to serve content over SSL.

```
[root@servera ~]# dnf install httpd mod_ssl
...output omitted...
```

```
9.  Is this ok [y/N]: y
10. ...output omitted...
```

```
Complete!
```

11. Create the `/var/www/html/index.html` file. Add one line of text that reads: `I am servera.`

```
[root@servera ~]# echo 'I am servera.' > /var/www/html/index.html
```

12. Start and enable the `httpd` service.

```
13. [root@servera ~]# systemctl enable --now httpd
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /u
sr/lib/systemd/system/httpd.service.
```

14. Return to the `workstation` machine as the `student` user.

```
15. [root@servera ~]# exit
16. logout
17. [student@servera ~]$ exit
18. logout
19. Connection to servera closed.
```

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

20. From `workstation`, try to access the web server on `servera` by using both the `80/TCP` clear-text port and the `443/TCP` SSL encapsulated port. Both attempts should fail.
    1. The `curl` command should fail.

    ```
    2. [student@workstation ~]$ curl http://servera.lab.example.com
    ```

    ```
    curl: (7) Failed to connect to servera.lab.example.com port 80: No route
    to host
    ```

    3. The `curl` command with the `-k` option for insecure connections should also fail.

    ```
    4. [student@workstation ~]$ curl -k https://servera.lab.example.com
    ```

```
curl: (7) Failed to connect to servera.lab.example.com port 443: No route to host
```

21. Verify that the `firewalld` service on `servera` is enabled and running.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-04-13 11:22:50 EDT; 7min ago
       Docs: man:firewalld(1)
   Main PID: 768 (firewalld)
      Tasks: 2 (limit: 10798)
     Memory: 39.9M
        CPU: 584ms
     CGroup: /system.slice/firewalld.service
             └─768 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Apr 13 11:22:49 servera.lab.example.com systemd[1]: Starting firewalld - dynamic firewall daemon...
```

```
Apr 13 11:22:50 servera.lab.example.com systemd[1]: Started firewalld - dynamic firewall daemon.
```

39. Add the `https` service to the `public` firewall zone.
    1. Verify that the default firewall zone is set to the `public` zone.

    ```
    [root@servera ~]# firewall-cmd --get-default-zone
    ```

    ```
    public
    ```

    3. If the earlier step does not return `public` as the default zone, then correct it with the following command:

```
[root@servera ~]# firewall-cmd --set-default-zone public
```

4. Add the `https` service to the permanent configuration for the `public` network zone. Confirm your configuration.

```
[root@servera ~]# firewall-cmd --permanent --add-service=https
success
[root@servera ~]# firewall-cmd --reload
success
[root@servera ~]# firewall-cmd --permanent --zone=public --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client https ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
```

```
  rich rules:
```

40. From `workstation`, open Firefox and log in to the web console that is running on `servera` to verify the `https` service to the `public` firewall zone.
    1. Open Firefox and navigate to `https://servera.lab.example.com:9090` to access the web console. Click **Advanced** and **Accept the Risk and Continue** to accept the self-signed certificate.
    2. Log in as the `student` user with `student` as the password.
    3. Click **Turn on administrative access** and enter the `student` password again.
    4. Click **Networking** in the left navigation bar.
    5. Click **Edit rules and zones** in the **Firewall** section of the **Networking** page.

    6. Verify that the `https` service is listed in the Service column.

41. Return to a terminal on `workstation`, and verify your work by attempting to access the `servera` web server.

    1. Return to the `workstation` machine as the `student` user.

```
2. [root@servera ~]# exit
3. logout
4. [student@servera ~]$ exit
5. logout
6. Connection to servera closed.
```

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

    7. Verify the access to the `http://servera.lab.example.com` web server.

```
8. [student@workstation ~]$ curl http://servera.lab.example.com
```

```
curl: (7) Failed to connect to servera.lab.example.com port 80: No route
to host
```

    9. Verify the access to the `http://servera.lab.example.com` web server through the port 443 for insecure connection.

```
10. [student@workstation ~]$ curl -k https://servera.lab.example.com
```

```
I am servera.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from earlier exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish netsecurity-firewalls
```

This concludes the section.

Control SELinux Port Labeling

Objectives

Verify that network ports have the correct SELinux type for services to bind to them.

SELinux Port Labeling

In addition to file context and process type labeling, SELinux labels network ports with an SELinux context. SELinux controls network access by labeling the network ports and including rules in a service's targeted policy. For example, the SSH targeted policy includes the `22/TCP` port with an `ssh_port_t` port context label. In the HTTP policy, the default `80/TCP` and `443/TCP` ports use an `http_port_t` port context label.

When a targeted process attempts to open a port for listening, SELinux verifies that the policy includes entries that enable the binding of the process and the context. SElinux can then block a rogue service from taking over ports that other legitimate network services use.

Manage SELinux Port Labeling

If a service attempts to listen on a nonstandard port, and the port is not labeled with the correct SELinux type, then SELinux might block the attempt. You can correct this problem by changing the SELinux context on the port.

Typically, the `targeted` policy already labeled all expected ports with the correct type. For example, because port `8008/TCP` is often used for web applications, that port is already labeled with `http_port_t`, which is the default port type that a web server uses. Individual ports can be labeled with only one port context.

List Port Labels

Use the `grep` command to filter the port number.

```
[root@host ~]# grep gopher /etc/services
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
```

Use the `semanage` command to list the current port label assignments.

```
[root@host ~]# semanage port -l
...output omitted...
http_cache_port_t       tcp   8080, 8118, 8123, 10001-10010
http_cache_port_t       udp   3130
http_port_t             tcp   80, 81, 443, 488, 8008, 8009, 8443, 9000
...output omitted...
```

Use the `grep` command to filter the SELinux port label by using the service name.

```
[root@host ~]# semanage port -l | grep ftp
ftp_data_port_t                tcp      20
ftp_port_t                     tcp      21, 989, 990
ftp_port_t                     udp      989, 990
tftp_port_t                    udp      69
```

A port label can appear in the list many times for each supported networking protocol.

Use the `grep` command to filter the SELinux port label by using the port number.

```
[root@host ~]# semanage port -l | grep -w 70
gopher_port_t                  tcp      70
gopher_port_t                  udp      70
```

Manage Port Bindings

Use the `semanage` command to assign new port labels, remove port labels, and modify existing ones.

Important

Almost all of the services that are included in the RHEL distribution provide an SELinux policy module, which includes that service's default port contexts. You cannot change default port labels by using the `semanage` command. Instead, you must modify and reload the targeted service's policy module. Writing and generating policy modules is not discussed in this course.

You can label a new port with an existing port context label (type). The `semanage port` command's `-a` option adds a new port label; the `-t` option denotes the type; and the `-p` option denotes the protocol.

```
[root@host ~]# semanage port -a -t port_label -p tcp|udp PORTNUMBER
```

In the following example, enable the `gopher` service to listen on the `71/TCP` port:

```
[root@host~]# semanage port -a -t gopher_port_t -p tcp 71
```

To view local changes to the default policy, use the `semanage port` command's -
c option.

```
[root@host~]# semanage port -l -C
SELinux Port Type              Proto    Port Number


gopher_port_t                  tcp      71
```

The targeted policies include many port types.

Service-specific SELinux man pages are named by using the service name plus `_selinux`. These man pages include service-specific information on SELinux types, Booleans, and port types, and are not installed by default. To view a list of all of the available SELinux man pages, install the package and then run a `man -k` keyword search for the `_selinux` string.

```
[root@host ~]# dnf -y install selinux-policy-doc
[root@host ~]# man -k _selinux
```

Use the `semanage` command for deleting a port label, with the `-d` option. In the following example, remove the binding of port `71/TCP` to the `gopher_port_t` type:

```
[root@host ~]# semanage port -d -t gopher_port_t -p tcp 71
```

To change a port binding, when requirements change, use the `-m` option. This option is more efficient than deleting the earlier binding and adding the latest one.

For example, to modify port `71/TCP` from `gopher_port_t` to `http_port_t`, use the following command:

```
[root@server ~]# semanage port -m -t http_port_t -p tcp 71
```

View the modification by using the `semanage` command.

```
[root@server ~]# semanage port -l -C
SELinux Port Type              Proto    Port Number

http_port_t                    tcp      71
[root@server ~]# semanage port -l | grep http
http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t              udp      3130
http_port_t                    tcp      71, 80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t            tcp      5988
pegasus_https_port_t           tcp      5989
```

Guided Exercise: Control SELinux Port Labeling

In this lab, you configure your system to allow HTTP access on a nonstandard port.

Outcomes

- Configure a web server that is running on `servera` to successfully serve content that uses a nonstandard port.

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

This command determines whether the `servera` machine is reachable on the network, installs the `httpd` service, and configures the firewall on `servera` to allow HTTP connections.

```
[student@workstation ~]$ lab start netsecurity-ports
```

Instructions

Your organization is deploying a new custom web application. The web application is running on a nonstandard port, in this case, `82/TCP`.

A junior administrator already configured the application on your `servera` host. However, the web server content is not accessible.

1. Log in to `servera` as the `student` user and switch to the `root` user.

```
2. [student@workstation ~]$ ssh student@servera
3. ...output omitted...
4. [student@servera ~]$
5. [student@servera ~]$ sudo -i
6. [sudo] password for student: student
```

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

7. Try to fix the web content problem by restarting the `httpd` service.
   1. Restart the `httpd.service`. This command is expected to fail.

```
2. [root@servera ~]# systemctl restart httpd.service
3. Job for httpd.service failed because the control process exited with error code.
```

```
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

   4. View the status of the `httpd` service. Note the `permission denied` error.

```
5. [root@servera ~]# systemctl status -l httpd.service
6. ● httpd.service - The Apache HTTP Server
7.    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
8.    Active: failed (Result: exit-code) since Mon 2019-04-08 14:23:29 CEST; 3min 33s ago
9.      Docs: man:httpd.service(8)
10.   Process: 28078 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
11.  Main PID: 28078 (code=exited, status=1/FAILURE)
12.    Status: "Reading configuration..."
13.
14. Apr 08 14:23:29 servera.lab.example.com systemd[1]: Starting The Apache HTTP Server...
```

```
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission denied: AH00072: make_sock: could not bind to address [::]:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission denied: AH00072: make_sock: could not bind to address 0.0.0.0:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: no listening sockets available, shutting down
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: AH00015: Unable to open logs
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service: Main process exited, code=exited, status=1/FAILURE
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service: Failed with result 'exit-code'.
```

```
Apr 08 14:23:29 servera.lab.example.com systemd[1]: Failed to start The Apache HTTP Server.
```

21. Verify whether SELinux is blocking `httpd` from binding to the `82/TCP` port.

```
[root@servera ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port 82.

*****  Plugin bind_ports (99.5 confidence) suggests   ***********************

If you want to allow /usr/sbin/httpd to bind to network port 82
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 82 where PORT_TYPE is one of the following: http_cache_port_t, http_port_t, jboss_management_port_t, jboss_messaging_port_t, ntop_port_t, puppet_port_t.
...output omitted...
Raw Audit Messages
```

```
37. type=AVC msg=audit(1554726569.188:852): avc:  denied  { name_bind } for
    pid=28393 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0 tcon
    text=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
```

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

8. Configure SELinux to allow the `httpd` service to bind to the `82/TCP` port, and then restart the `httpd.service` service.
   1. Find an appropriate port type for the `82/TCP` port.

      The `http_port_t` type includes the default HTTP ports, `80/TCP` and `443/TCP`. This type is the correct port type for the web server.

      ```
      [root@servera ~]# semanage port -l | grep http
      http_cache_port_t              tcp      8080, 8118, 8123, 10001-10010
      http_cache_port_t              udp      3130
      http_port_t                    tcp      80, 81, 443, 488, 8008, 8009, 84
      43, 9000
      pegasus_http_port_t            tcp      5988
      pegasus_https_port_t           tcp      5989
      ```

   2. Assign the `82/TCP` port the `http_port_t` type.

      ```
      [root@servera ~]# semanage port -a -t http_port_t -p tcp 82
      ```

   3. Restart the `httpd.service` service. This command should succeed.

      ```
      [root@servera ~]# systemctl restart httpd.service
      ```

9. Verify that you can now access the web server that runs on the `82/TCP` port.

```
10. [root@servera ~]# curl http://servera.lab.example.com:82
```

```
Hello
```

11. In a different terminal window, verify whether you can access the new web service from `workstation`.

```
12. [student@workstation ~]$ curl http://servera.lab.example.com:82
```

```
curl: (7) Failed to connect to servera.example.com:82; No route to host
```

That error means that you still cannot connect to the web service from `workstation`.

13. On `servera`, open the `82/TCP` port on the firewall.
    1. Open the `82/TCP` port in the permanent configuration, for the default zone on the firewall, on `servera`.

```
2. [root@servera ~]# firewall-cmd --permanent --add-port=82/tcp
```

```
success
```

3. Activate your firewall changes on `servera`.

```
4. [root@servera ~]# firewall-cmd --reload
```

```
success
```

14. Access the web service from `workstation`.

```
15. [student@workstation ~]$ curl http://servera.lab.example.com:82
```

```
Hello
```

16. Return to the `workstation` system as the `student` user.

```
17. [root@servera ~]# exit
18. logout
19. [student@servera ~]$ exit
20. logout
21. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish netsecurity-ports
```

This concludes the section.

Summary

- The `netfilter` framework enables kernel modules to inspect every packet that traverses the system, including all incoming, outgoing, or forwarded network packets.
- The `firewalld` service simplifies management by classifying all network traffic into zones. Each zone has its own list of ports and services. The `public` zone is set as the default zone.
- The `firewalld` service ships with predefined services. You can list these services by using the `firewall-cmd --get-services` command.
- SELinux policy controls network traffic by labeling the network ports. For example, the `ssh_port_t` label is associated with the `22/TCP` port. When a process wants to listen on a port, SELinux verifies whether the port's associated label is allowed to bind that port label.
- Use the `semanage` command to add, delete, and modify labels.

Chapter 12. Install Red Hat Enterprise Linux

Abstract

Goal	Install Red Hat Enterprise Linux on servers and virtual machines.
Objectives	• Install Red Hat Enterprise Linux on a server. • Explain Kickstart concepts and architecture, create a Kickstart file with the Kickstart Generator website, modify an existing Kickstart file with a text editor and check its syntax with `ksvalidator`, publish a Kickstart file to the installer, and install Kickstart on the network. • Install a virtual machine on your Red Hat Enterprise Linux server with the web console.
Sections	• Install Red Hat Enterprise Linux (and Guided Exercise) • Automate Installation with Kickstart (and Guided Exercise) • Install and Configure Virtual Machines (and Quiz)
Lab	Install Red Hat Enterprise Linux

Install Red Hat Enterprise Linux

Objectives

Install Red Hat Enterprise Linux on a server.

Installation Media

Red Hat provides different forms of installation media that you can download from the Customer Portal website by using your active subscription.

- A binary image file in ISO 9660 format that contains the *Anaconda* Red Hat Enterprise Linux installation program, and the BaseOS and AppStream package repositories. These repositories contain the needed packages to complete the installation without additional repositories.
- A smaller "boot ISO" image file that contains Anaconda requires a configured network to access package repositories that are made available by using HTTP, FTP, or NFS.
- A QCOW2 image contains a prebuilt system disk that is ready to deploy as a virtual machine in cloud or enterprise virtual environments. Red Hat uses QCOW2 as the standard image format for KVM-based virtualization.
- Source code (human-readable programming language instructions) for Red Hat Enterprise Linux. The source DVDs have no documentation. This image helps to compile or develop your software according to the Red Hat Enterprise Linux version.

Red Hat Enterprise Linux supports the following architectures:

- AMD, Intel, and ARM 64-bit architectures
- IBM Power Systems (Little Endian, LC servers, and AC servers)
- IBM Z 64-bit

After downloading, create bootable installation media according to the instructions in the reference section.

Build Images by Using Image Builder

The Red Hat Image Builder tool helps to create customized images of Red Hat Enterprise Linux. Image Builder enables administrators to build custom system images for deployment on cloud platforms or on virtual environments for specialized use cases.

Use the `composer-cli` command or the Red Hat web console interface to access Image Builder.

Install Red Hat Enterprise Linux Manually

By using the binary DVD or boot ISO, administrators install a new RHEL system on a bare-metal server or on a virtual machine. The Anaconda program supports two installation methods: manual and automated.

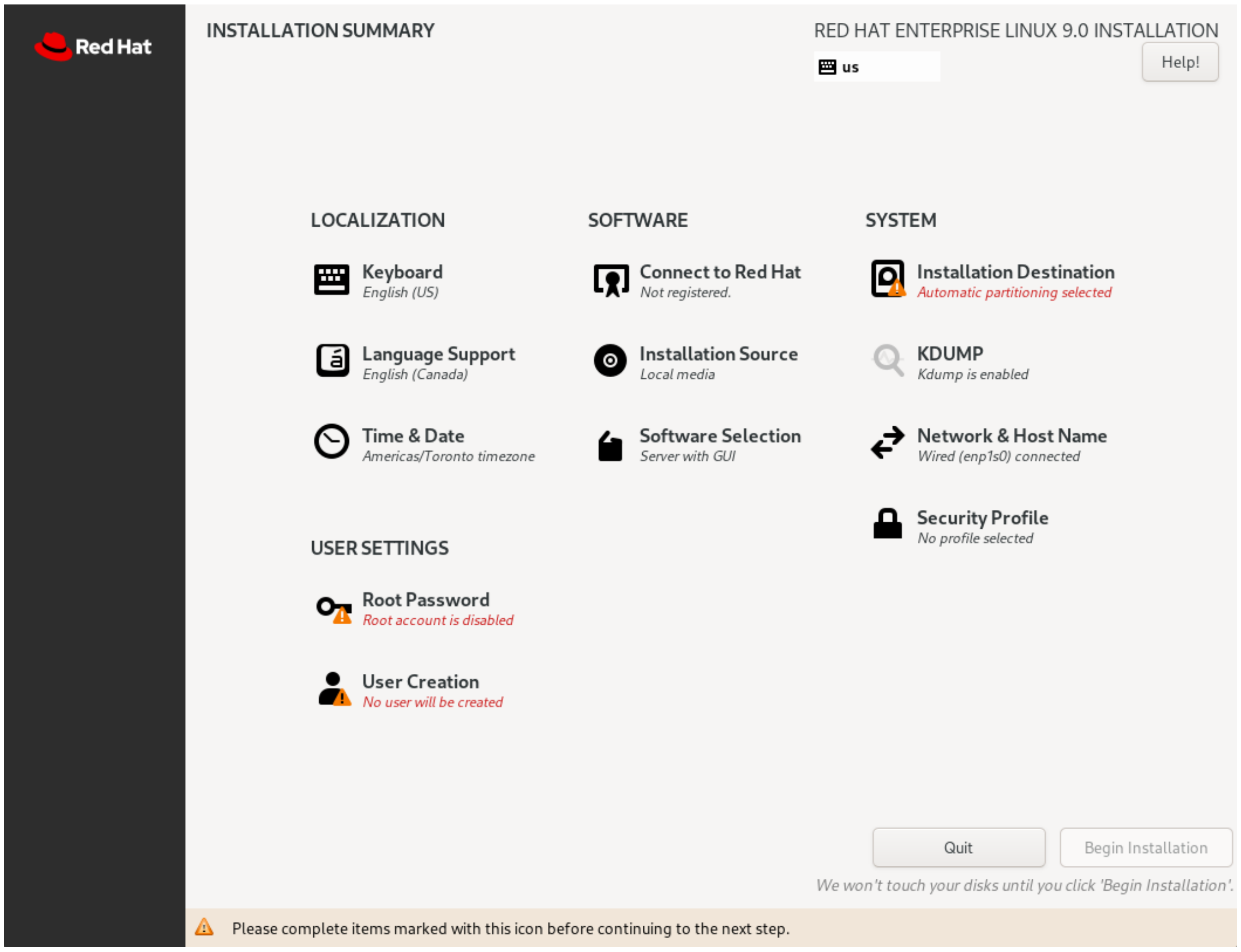
- The manual installation interacts with the user to query how Anaconda installs and configures the system.
- The automated installation uses a *Kickstart* file to direct Anaconda how to install the system.

Install RHEL by Using the Graphical Interface

Anaconda starts as a graphical application when you boot the system from the binary DVD or from the boot ISO.

At the **WELCOME TO RED HAT ENTERPRISE LINUX 9** screen, select the language, and click **Continue**. Individual users can choose a preferred language after installation.

Anaconda presents the **INSTALLATION SUMMARY** window, the central interface to customize parameters before beginning the installation.



LOCALIZATION

- Keyboard — English (US)
- Language Support — English (Canada)
- Time & Date — Americas/Toronto timezone

SOFTWARE

- Connect to Red Hat — Not registered.
- Installation Source — Local media
- Software Selection — Server with GUI

SYSTEM

- Installation Destination — Automatic partitioning selected
- KDUMP — Kdump is enabled
- Network & Host Name — Wired (enp1s0) connected
- Security Profile — No profile selected

USER SETTINGS

- Root Password — Root account is disabled
- User Creation — No user will be created

We won't touch your disks until you click 'Begin Installation'.

Please complete items marked with this icon before continuing to the next step.

Figure 12.1: Installation summary window

From this window, configure the installation parameters by selecting the icons in any order. Select an item to view or to edit. In any item, click **Done** to return to this central screen.

Anaconda marks mandatory items with a triangle warning symbol and message. The orange status bar at the bottom of the screen reminds you to complete the required information before the installation begins.

Complete the following items as needed:

- **Keyboard**: Add keyboard layouts.
- **Language Support**: Select additional languages to install.
- **Time & Date**: Select the system's location city by clicking the interactive map or selecting it from the lists. Specify the local time zone even when using *Network Time Protocol (NTP)*.

- **Connect to Red Hat**: Register the system with your Red Hat account and select the *system purpose*. The system purpose feature enables the registration process to automatically attach the most appropriate subscription to the system. You must first connect to the network by using the **Network & Host Name** icon to register the system.
- **Installation Source**: Provide the source package location that Anaconda requires for installation. The installation source field already refers to the DVD when using the binary DVD.
- **Software Selection**: Select the base environment to install, and add any add-ons. The **Minimal Install** environment installs only the essential packages to run Red Hat Enterprise Linux.
- **Installation Destination**: Select and partition the disks for Red Hat Enterprise Linux to install to. To complete this task, the administrator must know partitioning schemes and file-system selection criteria. The default radio button for automatic partitioning allocates the selected storage devices by using all available space.
- **KDUMP**: The *kdump* kernel crash dump feature collects information about the state of the system memory when the kernel crashes. Red Hat engineers analyze a kdump file to identify the cause of a crash. Use this Anaconda item to enable or to disable kdump.
- **Network & Host Name**: Detected network connections are listed on the left. Select a connection to display its details. By default, Anaconda activates the network automatically. Click **Configure** for the selected network connection.
- **Security Profile**: By activating a security profile, Anaconda applies restrictions and recommendations that the selected profile defines during installation.
- **Root Password**: The installation program prompts to set a `root` password. The final stage of the installation process continues only after you define a `root` password.
- **User Creation**: Create an optional non-root account. Creating a local, general-use account is a recommended practice. You can also create accounts after the installation is complete.

Note

When setting the `root` user password, Red Hat Enterprise Linux 9 enables an option to lock the `root` user access to the system. Red Hat Enterprise Linux 9 also enables password-based SSH access to the `root` user.

After you complete the installation configuration, and resolve all warnings, click **Begin Installation**. Clicking **Quit** aborts the installation without applying any changes to the system.

When the installation finishes, click **Reboot**. Anaconda displays the **Initial Setup** screen when installing a graphical desktop. Accept the license information and optionally register the system with the subscription manager. You might skip system registration until later.

Troubleshoot the Installation

During a Red Hat Enterprise Linux 9 installation, Anaconda provides two virtual consoles. The first virtual console has five windows that the `tmux` software terminal multiplexer supplies. You can access that console by using **Ctrl**+**Alt**+**F1**. The second virtual console, which is displayed by default, shows the Anaconda graphical interface. You can access it by using **Ctrl**+**Alt**+**F6**.

The `tmux` terminal provides a shell prompt in the second window in the first virtual console. You can use the terminal to enter commands to inspect and troubleshoot the system while the installation continues. The other windows provide diagnostic messages, logs, and additional information.

The following table lists the keystroke combinations to access the virtual consoles and the `tmux` terminal windows. In the `tmux` terminal, the keyboard shortcuts are performed in two actions: press and release **Ctrl**+**B**, and then press the number key of the window to access. In the `tmux` terminal, you can also use **Alt**+**Tab** to rotate the current focus between the windows.

Key sequence	**contents**
Ctrl+**Alt**+**F1**	Access the `tmux` terminal multiplexer.
Ctrl+**B 1**	In the `tmux` terminal, access the main information page for the installation process.
Ctrl+**B 2**	In the `tmux` terminal, provide a root shell. Anaconda stores the installation log files in the `/tmp` directory.
Ctrl+**B 3**	In the `tmux` terminal, display the contents of the `/tmp/anaconda.log` file.

Key sequence	**contents**
Ctrl+**B 4**	In the `tmux` terminal, display the contents of the `/tmp/storage.log` file.
Ctrl+**B 5**	In the `tmux` terminal, display the contents of the `/tmp/program.log` file.
Ctrl+**Alt**+**F6**	Access the Anaconda graphical interface.

Note

For compatibility with earlier Red Hat Enterprise Linux versions, the virtual consoles from **Ctrl**+**Alt**+**F2** through **Ctrl**+**Alt**+**F5** also present root shells during installation.

uided Exercise: Install Red Hat Enterprise Linux

In this exercise, you reinstall one of your servers with a minimal installation of Red Hat Enterprise Linux.

Outcomes

- Manually install Red Hat Enterprise Linux 9.

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

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

```
[student@workstation ~]$ lab start installing-install
```

Instructions

1. Access the `servera` console and reboot the system into the installation media.
    1. Locate the `servera` console icon in your classroom environment. Open the console.
    2. To reboot, send **Ctrl**+**Alt**+**Del** to your system by using the relevant keyboard, virtual, or menu entry.

    3. When the boot loader menu appears, select the **Install Red Hat Enterprise Linux 9** menu entry.
2. Keep the default selected language and click **Continue**.
3. Click **Network & Host Name** to configure the network.
    1. Enter `servera.lab.example.com` in the **Host Name** field and then click **Apply**.
    2. Click **Configure** and then click the **IPv4 Settings** tab.
    3. Enter the following details:

Field	**Value**
Address	`172.25.250.10`
Netmask	`24`
Gateway	`172.25.250.254`
DNS servers	`172.25.250.254`

        4. Note
        5. Because Red Hat Enterprise Linux is already installed on the `servera.lab.example.com` machine, the values for each of the required fields exist. For a clean server, you must complete this information.
    6. Click **Save** to save the network configuration, and then click **Done**.
4. Click **Installation Source** to configure the network installation source.
    1. Select the **On the network** option.
    2. Enter `content.example.com/rhel9.0/x86_64/dvd` in the network field.
    3. Click **Done**.
5. Click **Installation Destination** to select the installation disk.
    1. Select the `vda` disk and then click **Done**.
    2. On the next page, click **Reclaim space**.

        Note

        The `/dev/vda` disk already has partitions and file systems from the previous installation. With this selection, you can wipe the disk for the new installation.

    3. Click **Delete all**, and then click **Reclaim space**.
6. Click **Software Selection**, select **Minimal Install** from the **Base Environment** list, and then click **Done**.
7. Set `redhat` as the password for the `root` user.

    1. Click **Root Password** and enter `redhat` in the **Root Password** field.
    2. Enter `redhat` in the **Confirm** field.
    3. Click **Done** twice because the password fails the dictionary check.
8. Add the `student` user and set `student` as the password.
    1. Click **User Creation**.
    2. Enter `student` in the **Full Name** field. The **User name** field automatically fills `student` as the username.
    3. Select **Make this user administrator** to enable the `student` user to use the `sudo` command to run commands as the `root` user.
    4. Enter `student` in the **Password** field.
    5. Enter `student` in the **Confirm password** field.
    6. Click the **Done** button twice because the password fails the dictionary check.
9. Click **Begin Installation**.
10. Click **Reboot System** when the installation is complete.
11. When the system displays the login prompt, log in as the `student` user.
12. After you validate the installation, reset the `servera` machine from the web page of the classroom environment.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish installing-install
```

This concludes the section.

Automate Installation with Kickstart

Objectives

Explain Kickstart concepts and architecture, create a Kickstart file with the Kickstart Generator website, modify an existing Kickstart file with a text editor and check its syntax with `ksvalidator`, publish a Kickstart file to the installer, and install Kickstart on the network.

Introduction to Kickstart

The *Kickstart* feature of Red Hat Enterprise Linux automates system installations. You can use Kickstart text files to configure disk partitioning, network interfaces, package selection, and customize the installation. The Anaconda installer uses Kickstart files for a complete installation without user interaction. The Kickstart feature is similar and uses an unattended installation answer file for Microsoft Windows.

Kickstart files begin with a list of commands that define how to install the target machine. The installer ignores comment lines, which start with the number sign (`#`) character. Additional sections begin with a directive, which start with the percentage sign (`%`) character, and end on a line with the `&end` directive.

The `%packages` section specifies which software to include on installation. Specify individual packages by name, without versions. The at sign (`@`) character denotes package groups (either by group or ID), and the `@^` characters denote environment groups (groups of package groups). Lastly, use the `@module:stream/profile` syntax to denote module streams.

Groups have mandatory, default, and optional components. Normally, Kickstart installs mandatory and default components. To exclude a package or a package group from the installation, precede it with a hyphen (`-`) character. Excluded packages or package groups might still install if they are mandatory dependencies of other requested packages.

A Kickstart configuration file typically includes one or more `%pre` and `%post` sections, which contain scripts that further configure the system. The `%pre` scripts execute before any disk partitioning is done. Typically, you use `%pre` scripts to initialize a storage or network device that the remainder of the installation requires. The `%post` scripts execute after the initial installation is complete. Scripts within the `%pre` and `%post` sections can use any available interpreter on the system, including Bash or Python. Avoid the use of a `%pre` section, because any errors that occur within it might be difficult to diagnose.

Lastly, you can specify as many sections as you need, in any order. For example, you can have two `%post` sections, and they are interpreted in order of appearance.

Note

The RHEL Image Builder is an alternative installation method to Kickstart files. Rather than a text file that provides installation instructions, Image Builder creates an image with all the required system changes. The RHEL Image Builder can create images for public clouds such as Amazon Web Services and Microsoft Azure, or for private clouds such as OpenStack or VMware. See this section's references for more information about the RHEL Image Builder.

Installation Commands

The following Kickstart commands configure the installation source and method:

- `url`: Specifies the URL that points to the installation media.

```
url --url="http://classroom.example.com/content/rhel9.0/x86_64/dvd/"
```

- `repo`: Specifies where to find additional packages for installation. This option must point to a valid `DNF` repository.

```
repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel9.0
/x86_64/dvd/AppStream/
```

- `text`: Forces a text mode installation.
- `vnc`: Enables the VNC viewer so you can access the graphical installation remotely over VNC.

```
vnc --password=redhat
```

Device and Partition Commands

The following Kickstart commands configure devices and partitioning schemes:

- `clearpart`: Removes partitions from the system before creating partitions.

```
clearpart --all --drives=vda,vdb
```

- `part`: Specifies the size, format, and name of a partition. Required unless the `autopart` or `mount` commands are present.

```
part /home --fstype=ext4 --label=homes --size=4096 --maxsize=8192 --grow
```

- `autopart`: Automatically creates a root partition, a swap partition, and an appropriate boot partition for the architecture. On large enough drives (50 GB+), this command also creates a `/home` partition.
- `ignoredisk`: Prevents Anaconda from modifying disks, and is useful alongside the `autopart` command.

```
ignoredisk --drives=sdc
```

- `bootloader`: Defines where to install the bootloader. Required.

```
bootloader --location=mbr --boot-drive=sda
```

- `volgroup`, `logvol`: Creates LVM volume groups and logical volumes.

```
• 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`: Initialize disks whose formatting is unrecognized.

Network Commands

The following Kickstart commands configure networking-related features:

- `network`: Configures network information for the target system. Activates network devices in the installer environment.

```
network --device=eth0 --bootproto=dhcp
```

- `firewall`: Defines the firewall configuration for the target system.

```
firewall --enabled --service=ssh,http
```

Location and Security Commands

The following Kickstart commands configure security, language, and region settings:

- `lang`: Sets the language to use on the installed system. Required.

```
lang en_US
```

- `keyboard`: Sets the system keyboard type. Required.

```
keyboard --vckeymap=us
```

- `timezone`: Defines the time zone and whether the hardware clock uses UTC. Required.

```
timezone --utc Europe/Amsterdam
```

- `timesource`: Enables or disables NTP. If you enable NTP, then you must specify NTP servers or pools.

```
timesource --ntp-server classroom.example.com
```

- `authselect`: Sets up authentication options. Options that the `authselect` command recognizes are valid for this command. See authselect(8).
- `rootpw`: Defines the initial `root` user password. Required.

```
• rootpw --plaintext redhat
• or
```

```
rootpw --iscrypted $6$KUnFfrTzO8jv.PiH$YlBbOtXBkWzoMuRfb0.SpbQ....XDR1UuchoMG1
```

- `selinux`: Sets the SELinux mode for the installed system.

```
selinux --enforcing
```

- `services`: Modifies the default set of services to run under the default `systemd` target.

```
services --disabled=network,iptables,ip6tables --enabled=NetworkManager,firewa
lld
```

- `group`, `user`: Creates a local group or user on the system.

-
  ```
  group --name=admins --gid=10001
  ```

  ```
  user --name=jdoe --gecos="John Doe" --groups=admins --password=changeme --&#xF
  EFF;plaintext
  ```

Miscellaneous Commands

The following Kickstart commands configure logging the host power state on completion:

- `logging`: This command defines how Anaconda handles logging during the installation.

  ```
  logging --host=loghost.example.com
  ```

- `firstboot`: If enabled, then the Set up Agent starts the first time that the system boots. This command requires the `initial-setup` package.

  ```
  firstboot --disabled
  ```

- `reboot`, `poweroff`, `halt`: Specify the final action when the installation completes. The default setting is the `halt` option.

Note

Most Kickstart commands have multiple available options. Review the *Kickstart Commands and Options* guide in this section's references for more information.

Example Kickstart File

In the following example, the first part of the Kickstart file consists of the installation commands, such as disk partitioning and installation source.

```
#version=RHEL9


# Define system bootloader options

bootloader --append="console=ttyS0 console=ttyS0,115200n8 no_timer_check net.ifnames=
0  crashkernel=auto" --location=mbr --timeout=1 --boot-drive=vda
```

```
# Clear and partition disks

clearpart --all --initlabel

ignoredisk --only-use=vda

zerombr

part / --fstype="xfs" --ondisk=vda --size=10000


# Define installation options

text

repo --name="appstream" --baseurl="http://classroom.example.com/content/rhel9.0/x86_6
4/dvd/AppStream/"

url --url="http://classroom.example.com/content/rhel9.0/x86_64/dvd/"


# Configure keyboard and language settings

keyboard --vckeymap=us

lang en_US


# Set a root password, authselect profile, and selinux policy

rootpw --plaintext redhat

authselect select sssd

selinux --enforcing

firstboot --disable


# Enable and disable system services

services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"


# Configure the system timezone and NTP server

timezone America/New_York --utc

timesource --ntp-server classroom.example.com
```

The second part of a Kickstart file contains the `%packages` section, with details of which packages and package groups to install, and which packages not to install.

```
%packages


@core
```

```
chrony
cloud-init
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
-plymouth

%end
```

The last part of the Kickstart file contains a `%post` installation script.

```
%post

echo "This system was deployed using Kickstart on $(date)" > /etc/motd

%end
```

You can also specify a Python script with the `--interpreter` option.

```
%post --interpreter="/usr/libexec/platform-python"

print("This line of text is printed with python")

%end
```

Note

In a Kickstart file, missing required values cause the installer to interactively prompt for an answer or to abort the installation entirely.

Kickstart Installation Steps

Use the following steps to automate the installation of Red Hat Enterprise Linux with the Kickstart feature:

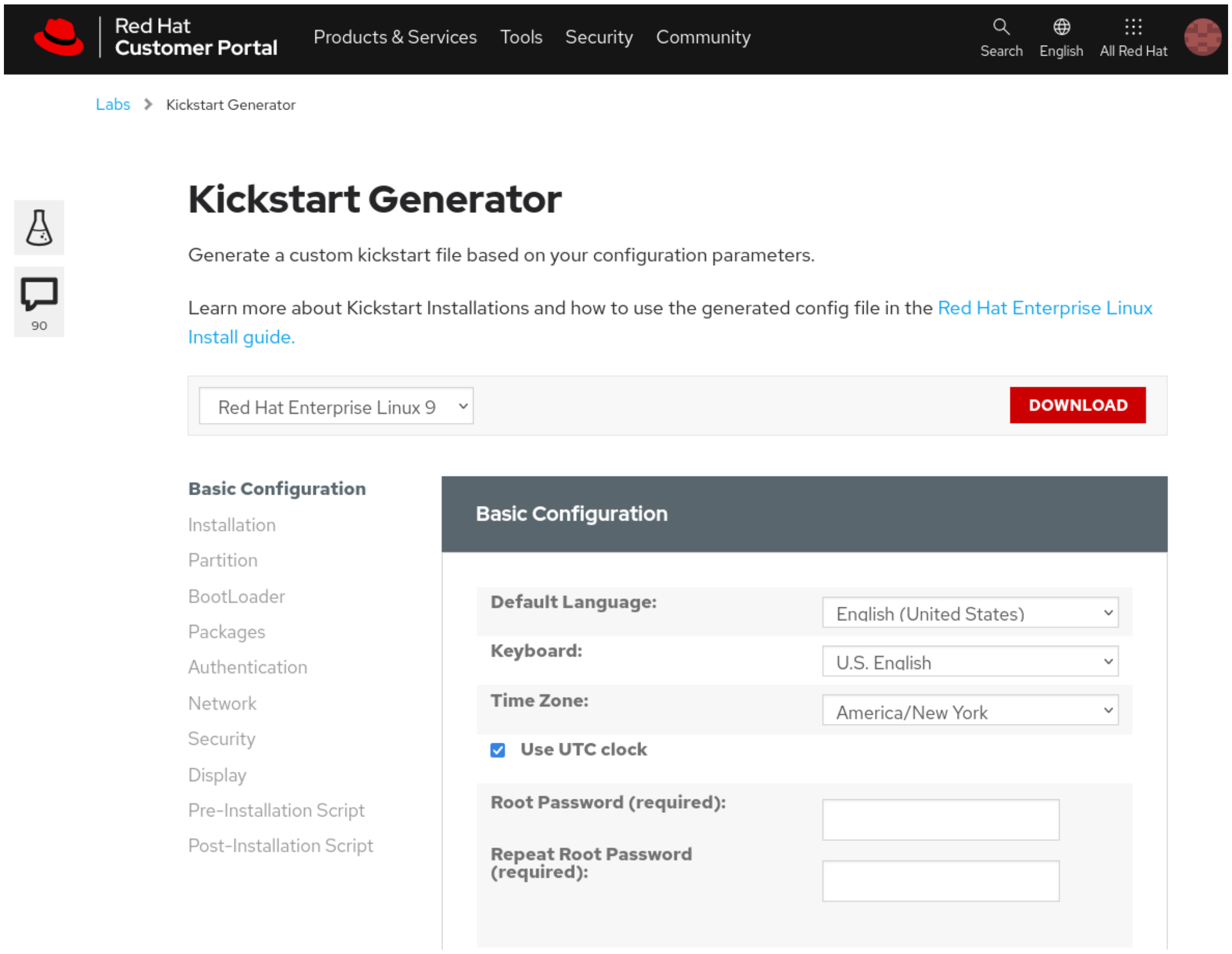
1. Create a Kickstart file.
2. Publish the Kickstart file so that the Anaconda installer can access it.
3. Boot the Anaconda installer and point it to the Kickstart file.

Create a Kickstart File

You can use either of these methods to create a Kickstart file:

- Use the Kickstart Generator website.
- Use a text editor.

The Kickstart Generator site at `https://access.redhat.com/labs/kickstartconfig`` presents dialog boxes for user inputs, and creates a Kickstart configuration file with the user's choices. Each dialog box corresponds to the configurable items in the Anaconda installer.


Red Hat Customer Portal

Kickstart Generator

Generate a custom kickstart file based on your configuration parameters.

Learn more about Kickstart Installations and how to use the generated config file in the Red Hat Enterprise Linux Install guide.

Basic Configuration

Default Language:	English (United States)
Keyboard:	U.S. English
Time Zone:	America/New York
✓ Use UTC clock	
Root Password (required):	
Repeat Root Password (required):	

Figure 12.2: The Red Hat Customer Portal Kickstart Generator

Creating a Kickstart file from scratch is complex, so first try to edit an existing file. Every installation creates a `/root/anaconda-ks.cfg` file that contains the Kickstart directives that are used in the installation. This file is a good starting point to create a Kickstart file.

The `ksvalidator` utility checks for syntax errors in a Kickstart file. It ensures that keywords and options are correctly used, but it does not validate URL paths, individual packages, groups, nor any part of `%post` or `%pre` scripts. For example, if the `firewall --disabled` directive is misspelled, then the `ksvalidator` command might produce one of the following errors:

```
[user@host ~]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:


Unknown command: frewall
```

```
[user@host ~]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:


no such option: --dsabled
```

The `ksverdiff` utility displays syntax differences between different operating system versions. For example, the following command displays the Kickstart syntax changes between RHEL 8 and RHEL 9:

```
[user@host ~]$ ksverdiff -f RHEL8 -t RHEL9
The following commands were removed in RHEL9:
device deviceprobe dmraid install multipath


The following commands were deprecated in RHEL9:
autostep btrfs method


The following commands were added in RHEL9:
timesource
...output omitted...
```

The `pykickstart` package provides the `ksvalidator` and `ksverdiff` utilities.

Publish the Kickstart File to Anaconda

Provide the Kickstart file to the installer by placing it in one of these locations:

- A network server that is available at installation time by using FTP, HTTP, or NFS.
- An available USB disk or CD-ROM.
- A local hard disk on the system.

The installer must access the Kickstart file to begin an automated installation. Usually, the file is made available via an FTP, web, or NFS server. Network servers help with Kickstart file maintenance, because changes can be made once, and then immediately be used for future installations.

By providing Kickstart files on USB or CD-ROM, is also convenient. The Kickstart file can be embedded in the boot media that starts the installation. However, when the Kickstart file is changed, you must generate new installation media.

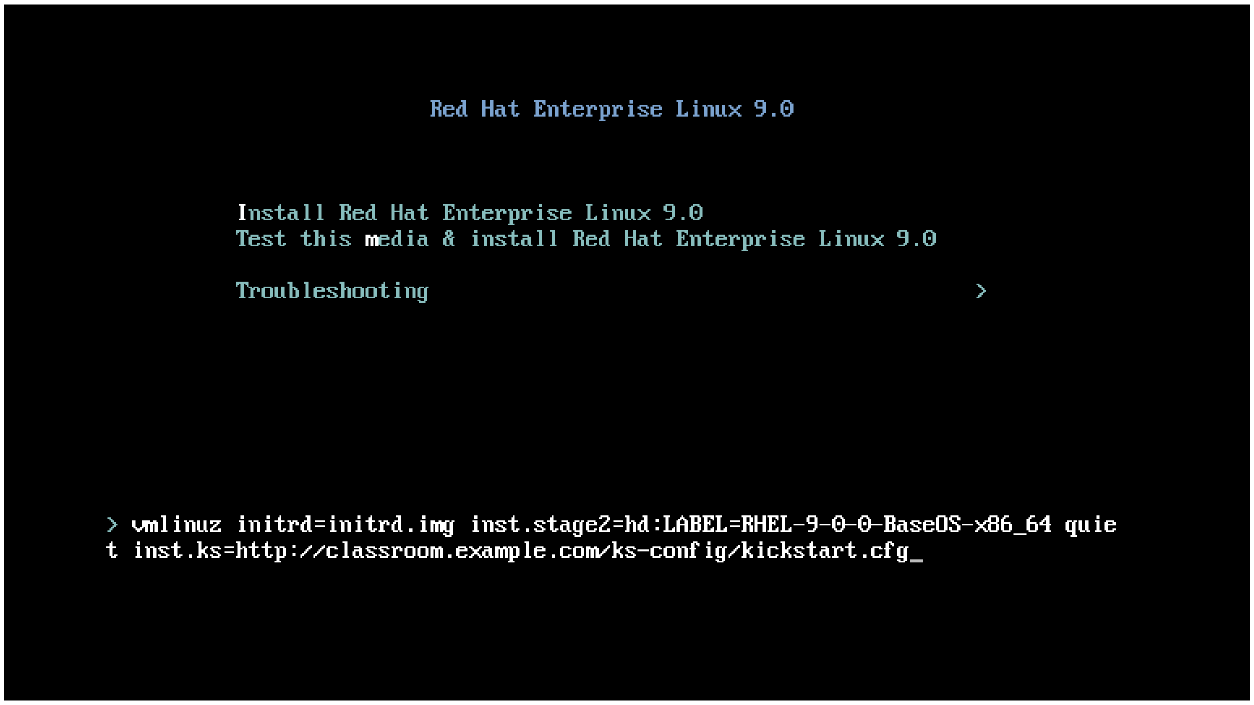
Providing the Kickstart file on a local disk enables a quick rebuild of a system.

Boot Anaconda and Point to the Kickstart File

After a Kickstart method is chosen, the installer is told where to locate the Kickstart file by passing the `inst.ks=LOCATION` parameter to the installation kernel.

Consider the following examples:

- `inst.ks=http://server/dir/file`
- `inst.ks=ftp://server/dir/file`
- `inst.ks=nfs:server:/dir/file`
- `inst.ks=hd:device:/dir/file`
- `inst.ks=cdrom:device`



```
                    Red Hat Enterprise Linux 9.0


          Install Red Hat Enterprise Linux 9.0
          Test this media & install Red Hat Enterprise Linux 9.0

          Troubleshooting                                      >





    > vmlinuz initrd=initrd.img inst.stage2=hd:LABEL=RHEL-9-0-0-BaseOS-x86_64 quie
    t inst.ks=http://classroom.example.com/ks-config/kickstart.cfg_
```

Figure 12.3: Specifying the Kickstart file location during installation

For virtual machine installations by 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 by using installation media, and press the **Tab** key to interrupt the boot process. Add an `inst.ks=LOCATION` parameter to the installation kernel.

Guided Exercise: Automate Installation with Kickstart

In this lab, you create a kickstart file and validate the syntax.

Outcomes

- Create a kickstart file.
- Validate the kickstart file's syntax.

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

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

```
[student@workstation ~]$ lab start installing-kickstart
```

Instructions

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

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

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

4. Create the `/home/student/kickstart.cfg` file by copying the contents of the `/root/anaconda-ks.cfg` file using privileged access.

```
5. [student@servera ~]$ sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg
```

```
[sudo] password for student: student
```

6. Make the following changes to the `/home/student/kickstart.cfg` file.
   1. Comment out the reboot command:

```
#reboot
```

2. Modify the `repo` commands to specify the `classroom` server's BaseOS and AppStream repositories:

```
3. repo --name="BaseOS" --baseurl="http://classroom.example.com/content/rhe
   l9.0/x86_64/dvd/BaseOS/"
```

```
repo --name="Appstream" --baseurl="http://classroom.example.com/content/
rhel9.0/x86_64/dvd/AppStream/"
```

4. Modify the `url` command to specify the `classroom` server's HTTP installation source:

```
url --url="http://classroom.example.com/content/rhel9.0/x86_64/dvd/"
```

5. Comment out the `network` command:

```
#network  --bootproto=dhcp --device=link --activate
```

6. Modify the `rootpw` command to set the `root` user's password to `redhat`.

```
rootpw --plaintext redhat
```

7. Modify the `authselect` command to set the `sssd` service as the identity and authentication source.

```
authselect select sssd
```

8. Modify the `services` command to disable and enable services.

```
services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"
```

9. Comment out the `part` commands and add the `autopart` command:

```
10.# Disk partitioning information
11.ignoredisk --only-use=vda
12.#part biosboot --fstype="biosboot" --size=1
13.#part /boot/efi --fstype="efi" --size=100 --fsoptions="..."
14.#part / --fstype="xfs
```

```
autopart
```

15. Delete all of the content between the `%post` section and its `%end` directive. Add the `echo "Kickstarted on $(date)" >> /etc/issue` line.

```
16. %post --erroronfail
17. echo "Kickstarted on $(date)" >> /etc/issue
```

```
%end
```

18. Modify the `%packages` section to include only the following content:

```
19. %packages
20. @core
21. chrony
22. dracut-config-generic
23. dracut-norescue
24. firewalld
25. grub2
26. kernel
27. rsync
28. tar
29. httpd
30. -plymouth
```

```
%end
```

31. Save and exit the file.

7. Validate the Kickstart file for syntax errors. If no errors are shown, then the command has no output.

```
[student@servera ~]$ ksvalidator kickstart.cfg
```

8. Copy the `kickstart.cfg` file to the `/var/www/html/ks-config` directory.

```
9. [student@servera ~]$ sudo cp ~/kickstart.cfg /var/www/html/ks-config
```

```
[sudo] password for student: student
```

10. Return to the `workstation` machine as the `student` user.

```
11. [student@servera ~]$ exit
12. logout
13. Connection to servera closed.
```

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish installing-kickstart
```

This concludes the section.

Install and Configure Virtual Machines

Objectives

Install a virtual machine on your Red Hat Enterprise Linux server with the web console.

Introducing KVM Virtualization

Virtualization is a feature to support dividing a single physical machine into multiple *virtual machines (VM)*, each of which can run an independent operating system.

Red Hat Enterprise Linux supports *KVM (Kernel-based Virtual Machine)*, a full virtualization solution that is built into the standard Linux kernel. KVM can run multiple Windows and Linux guest operating systems.

Figure 12.4: KVM virtualization

In Red Hat Enterprise Linux, you can manage KVM from the command line with the `virsh` command, or graphically with the web console's Virtual Machines tool.

KVM virtual machine technology is available across all Red Hat products, from stand-alone physical instances of Red Hat Enterprise Linux to the Red Hat OpenStack Platform:

- Physical hardware systems run Red Hat Enterprise Linux to provide KVM virtualization. Red Hat Enterprise Linux is typically a *thick* host, a system that supports VMs and also providing other local and network services, applications, and management functions.
- *Red Hat Virtualization (RHV)* provides a centralized web interface that administrators can use to manage an entire virtual infrastructure. It includes advanced features such as KVM migration, redundancy, and high availability. A *Red Hat Virtualization Hypervisor* is a tuned version of Red Hat Enterprise Linux solely for provisioning and supporting VMs.
- *Red Hat OpenStack Platform (RHOSP)* provides the foundation to create, deploy, and scale a public or a private cloud.
- *Red Hat OpenShift Virtualization* includes RHV components to enable deployment of containers on bare metal.

On systems where SELinux is enabled, sVirt isolates guests and the hypervisor. Each virtual machine process is labeled and is automatically allocated a unique level, and the associated virtual disk files are given matching labels.

Virtual Machine Types

Two main virtual machine types are available to install the guest operating system on. The earlier of the two uses the i440FX chip set, and provides ISA and PCI bridges, an IDE controller, and VGA and Ethernet adapters. The Q35 machine type is the later of the two, and should be used in most cases. It supports later virtual hardware interfaces such as PCIe and SATA, and includes an SM Bus controller.

In Red Hat Enterprise Linux 8 and later versions, UEFI and Secure Boot support for virtual machines is provided by Open Virtual Machine Firmware (OVMF), in the `edk2-ovmf` package.

Supported Guest Operating Systems

The list of Red Hat supported guest operating systems varies depending on the product in use. The RHV, RHOSP, and OpenShift Virtualization products all include Server Virtualization Validation Program (SVVP) certified drivers, and supports more supported Windows guest operating systems. A RHEL system that runs KVM only has SVVP support for specific subscriptions.

Review the following documents to determine whether a guest operating system is supported.

- RHV, RHOSP, and OpenShift Virtualization: Certified Guest Operating Systems in Red Hat OpenStack Platform, Red Hat Virtualization, OpenShift Virtualization and Red Hat Enterprise Linux with KVM

Configure a Red Hat Enterprise Linux Physical System as a Virtualization Host

Administrators can configure a Red Hat Enterprise Linux system as a virtualization host, as appropriate for development, testing, training, or when needing to work in multiple operating systems at the same time.

Install the Virtualization Tools

Install the `Virtualization Host` DNF package group to prepare a system to become a virtualization host.

```
[root@host ~]# dnf group list | grep -i virt
   Virtualization Host
[root@host ~]# dnf group info "Virtualization Host"
...output omitted...
Environment Group: Virtualization Host
 Description: Minimal virtualization host.
 Mandatory Groups:
   Base
   Core
   Standard
   Virtualization Hypervisor
   Virtualization Tools
 Optional Groups:
   Debugging Tools
   Network File System Client
   Remote Management for Linux
   Virtualization Platform

[root@host ~]# dnf group install "Virtualization Host"
...output omitted...
```

Verify the System Requirements for Virtualization

KVM requires either an Intel processor with the Intel VT-x and Intel 64 extensions for x86-based systems, or an AMD processor with the AMD-V and the AMD64 extensions. To verify your hardware and the system requirements, use the `virt-host-validate` command.

```
[root@host ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                            : PASS
  QEMU: Checking if device /dev/kvm exists                              : PASS
  QEMU: Checking if device /dev/kvm is accessible                       : PASS
  QEMU: Checking if device /dev/vhost-net exists                        : PASS
  QEMU: Checking if device /dev/net/tun exists                          : PASS
  QEMU: Checking for cgroup 'cpu' controller support                    : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support                : PASS
  QEMU: Checking for cgroup 'cpuset' controller support                 : PASS
  QEMU: Checking for cgroup 'memory' controller support                 : PASS
  QEMU: Checking for cgroup 'devices' controller support                : PASS
  QEMU: Checking for cgroup 'blkio' controller support                  : PASS
  QEMU: Checking for device assignment IOMMU support                    : PASS
  QEMU: Checking for secure guest support                               : PASS
```

The system must pass all the validation items to operate as a KVM host.

Install a Virtual Machine from the Command Line

Install the `virt-install` package, and then use the `virt-install` command to deploy a virtual machine from the command line. Many installation sources are supported, such as ISO, NFS, HTTP, and FTP. The following example uses the `virt-install` command's `--cdrom` option to install the guest from an ISO image.

```
[root@host ~]# virt-install --name demo --memory 4096 --vcpus 2 --disk size=40 \
--os-type linux --cdrom /root/rhel.iso
...output omitted...
```

Manage Virtual Machines with the Web Console

The `libvirt-client` package provides the `virsh` command, to manage virtual machines from the command line. Alternatively, you can create and manage virtual machines graphically by using the web console Virtual Machines plug-in.

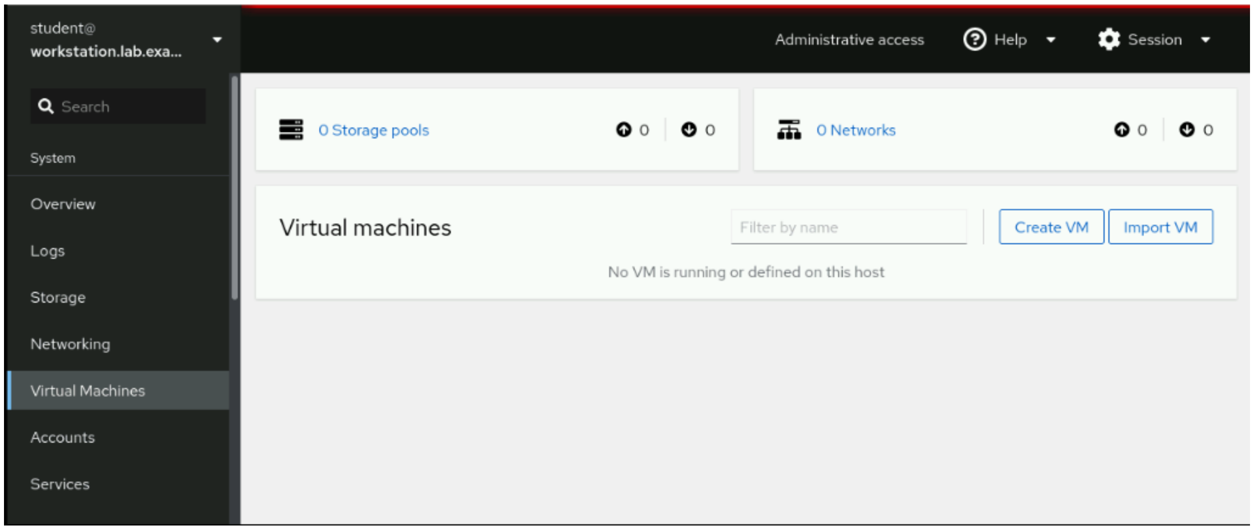
Install the `cockpit-machines` package to add the **Virtual Machines** menu to the web console.

```
[root@host ~]# dnf install cockpit-machines
```

If the web console is not already running, then start and enable it.

```
[root@host ~]# systemctl enable --now cockpit.socket
```

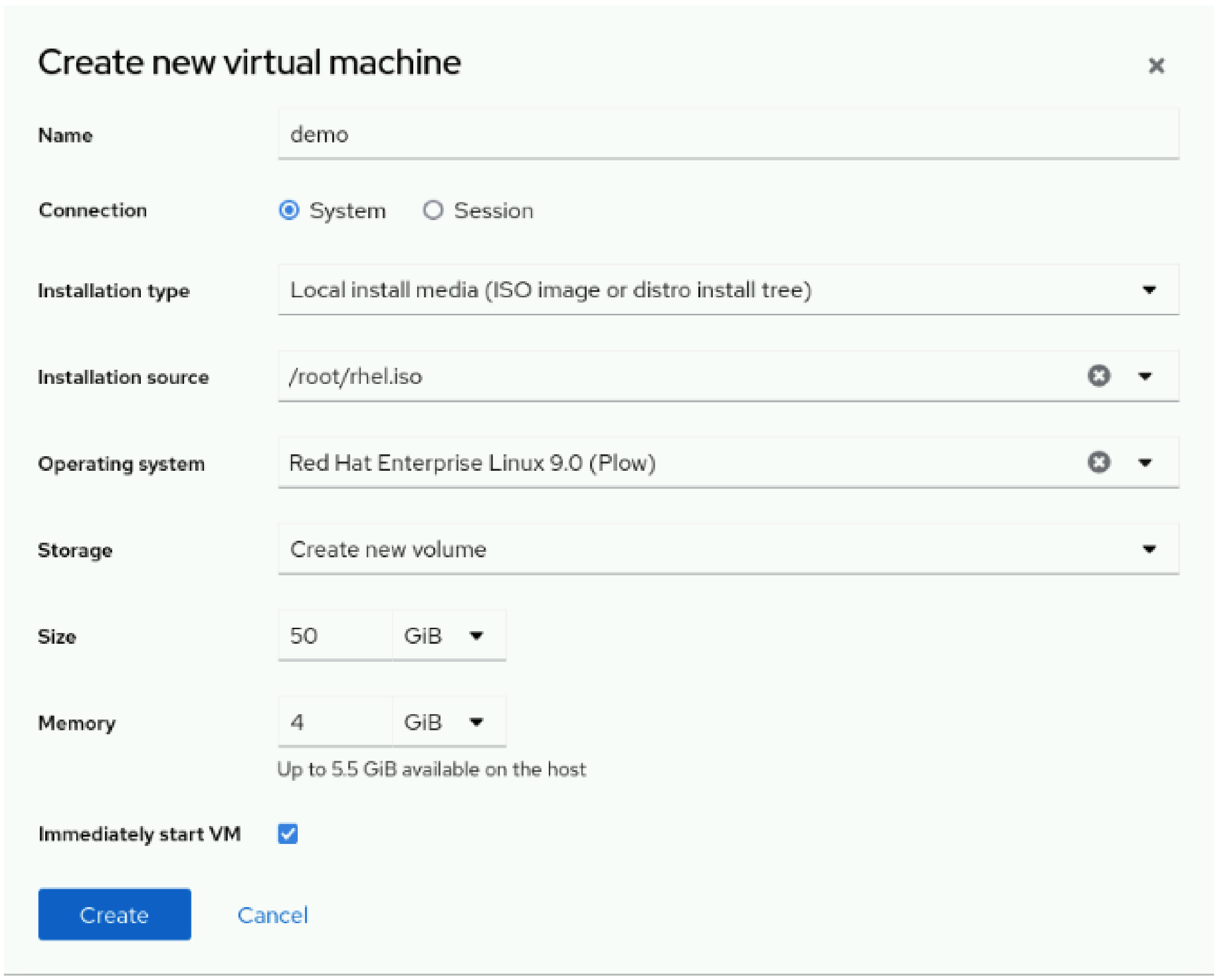
In a web browser that is running on the local machine, navigate to `https://localhost/9090` and log in to the web console. Switch to administrative mode and enter your password, if necessary.



No VM is running or defined on this host

Figure 12.5: Managing virtual machines in the web console

To create a virtual machine with the web console, access the **Virtual Machines** menu. From there, click **Create VM** and enter the VM configuration in the **Create New Virtual Machine** window. If you are using the web console for the first time after installing the Virtual Machines plug-in, then you must reboot your system to start the `libvirt` virtualization.



Create new virtual machine

Name	demo
Connection	[✓ System] Session
Installation type	Local install media (ISO image or distro install tree)
Installation source	/root/rhel.iso
Operating system	Red Hat Enterprise Linux 9.0 (Plow)
Storage	Create new volume
Size	50 GiB
Memory	4 GiB Up to 5.5 GiB available on the host
Immediately start VM	✓

Figure 12.6: Creating a virtual machine in the web console

- **Name** sets a *domain* name for the virtual machine configuration. This name is unrelated to the hostname that you give the virtual machine during installation.
- **Installation type** is the method for accessing the installation media. Choices include the local file system, or an HTTPS, FTP, or NFS URL, or PXE.
- **Installation source** provides the path to the installation source.

- **Operating system** defines the virtual machine's operating system. The virtualization layer presents hardware emulation to be compatible with the chosen operating system.
- **Storage** defines whether to create a storage volume or to use an existing one.
- **Size** is the disk size when creating a new volume. Associate additional disks with the VM after installation.
- **Memory** is the amount of RAM to provide to the new VM.
- **Immediately start VM** indicates whether the VM immediately starts after you click **Create**.

Click **Create** to create the VM, and click **Install** to start the operating system installation. The web console displays the VM console from which you can install the system.

Quiz: Install and Configure Virtual Machines

Choose the correct answers to the following questions:

1.

2.

1. Which package enables you to select the OVMF firmware for a virtual machine?

A `open-firmware`

B [✓ `edk2-ovmf`]

C `core-ovmf`

D `ovmf`

E `virt-open-firmware`

3. CheckResetShow Solution

4.

5.

2. Which two components are required to configure your system as a virtualization host, and to manage virtual machines with the web console? (Choose two.)

A [✓ The `Virtualization Host` package group]

B The `openstack` package group

C [✓ The `cockpit-machines` package]

D The `Virtualization Platform` package group

E The `kvm` DNF module

F The `cockpit-virtualization` package

6. CheckResetShow Solution

7.

8.

3. Which command verifies that your system supports virtualization?

A `grep kvm /proc/cpuinfo`

B `virsh validate`

C [✓ `virt-host-validate`]

D `rhv-validate`

E `cockpit-validate`

9. CheckResetShow Solution

10.

11.

4. Which two tools can you use to start and stop your virtual machines on a Red Hat Enterprise Linux system? (Choose two.)

A `vmctl`

B libvirtd

C [✓ `virsh`]

D `neutron`

E [✓ the web console]

12. CheckResetShow Solution

Previous Next

Summary

- The RHEL 9 binary DVD includes Anaconda and all required repositories for installation.
- The RHEL 9 boot ISO includes the Anaconda installer, and can access repositories over the network during installation.
- The Kickstart system can perform unattended installations.
- You can create Kickstart files by using the Kickstart Generator website or by copying and editing `/root/anaconda-ks.cfg`.
- The `Virtualization Host` DNF package group provides the packages for a RHEL system to become a virtualization host.
- The `cockpit-machines` package adds the **Virtual Machines** menu to Cockpit.

Previous Next

Chapter 13. Run Containers

Abstract

Goal	Obtain, run, and manage simple lightweight services as containers on a single Red Hat Enterprise Linux server.
Objectives	• Explain container concepts and the core technologies for building, storing, and running containers.

	• Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers. • Provide persistent storage for container data by sharing storage from the container host, and configure a container network. • Configure a container as a `systemd` service, and configure a container service to start at boot time.
Sections	• Container Concepts (and Quiz) • Deploy Containers (and Guided Exercise) • Manage Container Storage and Network Resources (and Guided Exercise) • Manage Containers as System Services (and Guided Exercise)
Lab	Run Containers

Container Concepts

Objectives

Explain container concepts and the core technologies for building, storing, and running containers.

Container Technology

Software applications typically depend on system libraries, configuration files, or services that their runtime environment provides. Traditionally, the runtime environment for a software application is installed in an operating system that runs on a physical host or a virtual machine. Administrators then install application dependencies on top of the operating system.

In Red Hat Enterprise Linux, packaging systems such as RPM help administrators to manage application dependencies. When you install the `httpd` package, the RPM system ensures that the correct libraries and other dependencies for that package are also installed.

The major drawback to traditionally deployed software applications is that these dependencies are entangled with the runtime environment. An application might require earlier or later versions of supporting software than the software that is provided with the operating system. Similarly, two applications on the same system might require different and incompatible versions of the same software.

One way to resolve these conflicts is to package and deploy the application as a *container*. A container is a set of one or more processes that are isolated from the rest of the system. Software containers provide a way to package applications and to simplify their deployment and management.

Think of a physical shipping container. A shipping container is a standard way to package and ship goods. It is labeled, loaded, unloaded, and transported from one location to another as a single box. The container's contents are isolated from the contents of other containers so that they do not affect each other. These underlying principles also apply to software containers.

Red Hat Enterprise Linux supports containers by using the following core technologies:

- *Control Groups (cgroups)* for resource management
- *Namespaces* for process isolation
- SELinux and *Seccomp* (Secure Computing mode) to enforce security boundaries

Note

For a deeper discussion of container architecture and security, refer to the "Ten Layers of Container Security" white paper.

Differences Between Containers and Virtual Machines

Containers provide many of the same benefits as virtual machines, such as security, storage, and network isolation.

Both technologies isolate their application libraries and runtime resources from the host operating system or hypervisor, and vice versa.

Figure 13.1: Comparison between virtualization and containerization

Containers and virtual machines interact differently with hardware and the underlying operating system.

A virtual machine has the following characteristics:

- Enables multiple operating systems to run simultaneously on a single hardware platform.
- Uses a hypervisor to divide hardware into multiple virtual hardware systems.
- Requires a complete operating system environment to support the application.

A container has the following characteristics:

- Runs directly on the host operating system, and it shares resources with all containers on the system.
- Shares the host's kernel, but it isolates the application processes from the rest of the system.
- Requires far fewer hardware resources than virtual machines, so containers are also quicker to start.
- Includes all dependencies, such as system and programming dependencies, and configuration settings.

Note

Some applications might not be suitable to run as a container. For example, applications that access low-level hardware information might need more direct hardware access than containers generally provide.

Rootless and Rootful Containers

On the container host, you can run containers as the root user or as a regular, unprivileged user. Containers that a privileged user runs are called *rootful containers*. Containers that non-privileged users run are called *rootless containers*.

A rootless container is not allowed to use system resources that are usually reserved for privileged users, such as access to restricted directories, or to publish network services on restricted ports (below 1024). This feature prevents a possible attacker from gaining root privileges on the container host.

Although you can run containers directly as `root` if necessary, this scenario weakens the security of the system if a bug enables an attacker to compromise the container.

Design a Container-based Architecture

Containers efficiently reuse hosted applications and make them portable. Containers can be moved from one environment to another, such as from development to production. You can save multiple versions of a container and access each one as needed.

Containers are typically temporary, or *ephemeral*. You can permanently save in persistent storage the data that a running container generates, but the containers themselves usually run when needed, and then they stop and are removed. A new container process is started the next time that particular container is needed.

You could install a complex software application with multiple services in a single container. For example, a web server might need to use a database and a messaging system. However, using one container for multiple services is hard to manage.

A better design runs in separate containers each component, the web server, the database, and the messaging system. This way, updates and maintenance to individual application components do not affect other components or the application stack.

Container Management Tools

Red Hat Enterprise Linux provides a set of container tools that you can use to run containers in a single server:

- `podman` manages containers and container images.
- `skopeo` inspects, copies, deletes, and signs images.
- `buildah` creates container images.

These tools are compatible with the Open Container Initiative (OCI). With these tools, you can manage any Linux containers that OCI-compatible container engines create, such as Podman or Docker. These tools are designed to run containers under Red Hat Enterprise Linux on a single-node container host.

In this chapter, you use the `podman` and `skopeo` utilities to run and manage containers and existing container images.

Note

Using `buildah` to construct your own container images is beyond the scope of this course. It is covered in the *Red Hat OpenShift I: Containers & Kubernetes* (DO180) Red Hat Training course.

Container Images and Registries

To run containers, you must use a *container image*. A container image is a static file that contains codified steps, and it serves as a blueprint to create containers. The container images package an application with all its dependencies, such as its system libraries, programming language runtimes and libraries, and other configuration settings.

Container images are built according to specifications, such as the Open Container Initiative (OCI) image format specification. These specifications define the format for container images, as well as the metadata about the container host operating systems and hardware architectures that the image supports.

A *container registry* is a repository for storing and retrieving container images. A developer *pushes* or uploads container images to a container registry. You can *pull* or download container images from a registry to a local system to run containers.

You might use a public registry that contains third-party images, or you might use a private registry that your organization controls. The source of your container images matters. As with any other software package, you must know whether you can trust the code in the container image. Policies vary between registries about whether and how they provide, evaluate, and test container images that are submitted to them.

Red Hat distributes certified container images through two main container registries that you can access with your Red Hat login credentials:

- `registry.redhat.io` for containers that are based on official Red Hat products
- `registry.connect.redhat.com` for containers that are based on third-party products

The Red Hat Container Catalog (`https://access.redhat.com/containers`) provides a web-based interface to search these registries for certified content.

Note

Red Hat provides the *Universal Base Image (UBI)* image as an initial layer to build containers. The UBI image is a minimized container image that can be a first layer for an application build.

You need a Red Hat Developer account to download an image from the Red Hat registries. You can use the `podman login` command to authenticate to the registries. If you do not provide a registry URL to the `podman login` command, then it authenticates to the default configured registry.

```
[user@host ~]$ podman login registry.lab.example.com
Username: RH134
Password: EXAMPLEPASSWORD
Login Succeeded!
```

You can also use the `podman login` command `--username` and `--password-stdin` options, to specify the user and password to log in to the registry. The `--password-stdin` option reads the password from stdin. Red Hat does not recommend using the `--password` option to provide the password directly, because this option stores the password in the log files.

```
[user@host ~]# echo $PASSWORDVAR | podman login --username RH134 \
--password-stdin registry.access.redhat.com
```

To verify that you are logged in to a registry, use the `podman login` command `--get-login` option.

```
[user01@rhel-vm ~]$ podman login registry.access.redhat.com --get-login
RH134
[user01@rhel-vm ~]$ podman login quay.io --get-login
Error: not logged into quay.io
```

In the preceding output, the `podman` utility is authenticated to the `registry.access.redhat.com` registry with the `RH134` user credentials, but the `podman` utility is not authenticated to the `quay.io` registry.

Configure Container Registries

The default configuration file for container registries is the `/etc/containers/registries.conf` file.

```
[user@host ~]$ cat /etc/containers/registries.conf
# For more information on this configuration file, see containers-registries.conf(5).
#
...output omitted...

unqualified-search-registries = ["registry.fedoraproject.org", "registry.access.redha
t.com", "registry.centos.org", "quay.io", "docker.io"]

# [[registry]]
# # The "prefix" field is used to choose the relevant [[registry]] TOML table;
# # (only) the TOML table with the longest match for the input image name
# # (taking into account namespace/repo/tag/digest separators) is used.
# #
# # The prefix can also be of the form: *.example.com for wildcard subdomain
# # matching.
# #
# # If the prefix field is missing, it defaults to be the same as the "location" fiel
d.
# prefix = "example.com/foo"
#
# # If true, unencrypted HTTP as well as TLS connections with untrusted
# # certificates are allowed.
# insecure = false
#
# # If true, pulling images with matching names is forbidden.
# blocked = false
#
```

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

Because Red Hat recommends using a non-privileged user to manage containers, you can create a `registries.conf` file for container registries in the `$HOME/.config/containers` directory. The configuration file in this directory overrides the settings in the `/etc/containers/registries.conf` file, and is used when Podman runs in rootless mode.

If you do not specify the fully qualified name of the container image when using `podman` commands, then the list of registries in the `unqualified-search-registries` section of this file is used to search for the container image.

```
[user@host ~]$ podman pull ubi
```

If you do specify the fully qualified name of the container image from the command line, then the container utility does not search in this section. The `unqualified-search-registries` section can be left blank to ensure that you use the fully qualified name of the container image.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/ubi:latest
```

Note

Red Hat recommends always using the fully qualified name of container images.

Configure settings for container registries in the `[[registry]]` sections of the file. Use a separate `[[registry]]` section to configure settings for each container registry.

```
[[registry]]
location = "registry.lab.example.com"
insecure = true
blocked = false
```

- The `location` setting specifies the location of the container registry.
- If the `insecure` setting is set to `true`, then you can use unencrypted HTTP as well as TLS connections with untrusted certificates to access the registry.
- If the `blocked` setting is set to `true`, then images cannot be downloaded from that registry.

Note

This classroom runs a private insecure registry that is based on Red Hat Quay to provide container images. This registry meets the classroom need; however, you would not expect to work with insecure registries in real-world scenarios. For more information about this software, see https://access.redhat.com/products/red-hat-quay

Container Files to Build Container Images

A *container file* is a text file with the instructions to build a container image. A container file usually has a *context* that defines the path or URL where its files and directories are located. The resulting container image consists of read-only layers, where each layer represents an instruction from the container file.

The following output is an example of a container file that uses the UBI image from the `registry.access.redhat.com` registry, installs the `python3` package, and prints the `hello` string to the console.

```
[user@host ~]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python3
CMD ["/bin/bash", "-c", "echo hello"]
```

Note

Creating a container file and its usage instructions are out of scope for this course. For more information about container files, refer to the DO180 course.

Container Management at Scale

New applications increasingly use containers to implement functional components. Those containers provide services that other parts of the application consume. In an organization, managing a growing number of containers might become an overwhelming task.

Deploying containers at scale in production requires an environment that can adapt to the following challenges:

- The platform must ensure the availability of containers that provide essential services.
- The environment must respond to application usage spikes by increasing or decreasing the number of running containers and by load balancing the traffic.
- The platform must detect the failure of a container or a host and react accordingly.
- Developers might need an automated workflow to deliver later application versions transparently and securely.

Kubernetes is an orchestration service that deploys, manages, and scales container-based applications across a cluster of container hosts. Kubernetes redirects traffic to your containers with a load balancer, so that you can scale the number of containers that provide a service. Kubernetes also supports user-defined health checks to monitor your containers and to restart them if they fail.

Red Hat provides a distribution of Kubernetes called Red Hat OpenShift. Red Hat OpenShift is a set of modular components and services that are built on top of the Kubernetes infrastructure. It provides additional features, such as remote web-based management, multitenancy, monitoring and auditing, advanced security features, application lifecycle management, and self-service instances for developers.

Red Hat OpenShift is beyond the scope of this course. You can learn more about it at `https://www.openshift.com`

Note

In the enterprise, individual containers are not generally run from the command line. Instead, it is preferable to run containers in production with a Kubernetes-based platform, such as Red Hat OpenShift.

However, you might need to use commands to manage containers and images manually or at a small scale. This chapter focuses on this use case to improve your grasp of the core concepts behind containers, how they work, and how they can be useful.

Quiz: Container Concepts

Choose the correct answers to the following questions:

1.

2.

1. Which Red Hat Enterprise Linux tool runs containers?

A	buildah
B	container
C	[✓ podman]
D	skopeo

3. CheckResetShow Solution

4.

5.

2. Which two statements describe container technology? (Choose two.)

A	Containers package complete operating systems, with the addition of library dependencies.

B [✓ Containers run processes that are isolated from the rest of the system.]

C Each container includes its own kernel version and libraries.

D [✓ Containers provide a standard way to package applications to ease deployment and management.]

6. CheckResetShow Solution

7.

8.

3. Which two statements are true about container images? (Choose two.)

A [✓ Container images package an application with all of its needed runtime dependencies.]

B Container images that work with Docker cannot work with Podman.

C Container images can run only on a container host with the same installed software version in the image.

D [✓ Container images serve as blueprints for creating containers.]

9. CheckResetShow Solution

10.

11.

4. Which three core technologies are used to implement containers in Red Hat Enterprise Linux? (Choose three.)

A Hypervisor code for hosting VMs

[✓ B Control Groups (cgroups) for resource management]

[✓ C Namespaces for process isolation]

D Full operating system for compatibility with the container's host

[✓ E SELinux and Seccomp for security]

12. CheckResetShow Solution

13.

14.

5. Which sentence is true about container files?

A A container file is an executable file that runs a container.

B A container file is an executable file that builds a container image.

C A container file is a compressed file that contains libraries and configuration for a container.

D A container file is a text file with the instructions to build a container.

E [✓ A container file is a text file with the instructions to build a container image.]

15. CheckResetShow Solution

Previous Next

Deploy Containers

Objectives

Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers.

The Podman Utility

Podman is a fully featured container engine from the `container-tools` meta-package to manage *Open Container Initiative (OCI)* containers and images. The `podman` utility does not use a daemon to function, and so developers do not need a privileged user account on the system to start or stop containers. Podman provides many subcommands to interact with containers and images. The following list shows subcommands that are used in this section:

Table 13.1. Podman Commands

Command	Description
`podman build`	Build a container image with a container file.
`podman run`	Run a command in a new container.
`podman images`	List images in local storage.
`podman ps`	Print information about containers.
`podman inspect`	Display configuration of a container, image, volume, network, or pod.
`podman pull`	Download an image from a registry.

Command	Description
`podman cp`	Copy files or directories between a container and the local file system.
`podman exec`	Execute a command in a running container.
`podman rm`	Remove one or more containers.
`podman rmi`	Remove one or more locally stored images.
`podman search`	Search a registry for an image.

For more information about each subcommand by using the man pages, append the subcommand to the `podman` command with a hyphen to separate the two. For example, the `podman-build` man page explains the use of the `podman build` subcommand.

To cover the topics in this lecture, imagine the following scenario.

As a system administrator, you are tasked to run a container that is based on the RHEL 8 UBI container image called `python38` with the `python-38` package. You are also tasked to create a container image from a container file, and to run a container called `python36` from that container image. The container image that is created with the container file must have the `python36:1.0` tag. Identify the differences between the two containers. Also, ensure that the installed `python` packages in the containers do not conflict with the installed Python version in your local machine.

Install Container Utilities

The `container-tools` meta-package contains required utilities to interact with containers and container images. To download, run, and compare containers on your system, you install the `container-tools` meta-package with the `dnf install` command. Use the `dnf info` command to view the version and contents of the `container-tools` package:

```
[root@host ~]# dnf install container-tools
...output omitted...
[user@host ~]$ dnf info container-tools
...output omitted...
Summary      : A meta-package witch container tools such as podman, buildah,
             : skopeo, etc.
License      : MIT
Description  : Latest versions of podman, buildah, skopeo, runc, conmon, CRIU,
             : Udica, etc as well as dependencies such as container-selinux
             : built and tested together, and updated.
...output omitted...
```

The `container-tools` meta-package provides the needed `podman` and `skopeo` utilities to achieve the assigned tasks.

Download a Container Image from a Registry

First, you ensure that the `podman` utility is configured to search and download containers from the `registry.redhat.io` registry. The `podman info` command displays configuration information of the `podman` utility, including the configured registries.

```
[user@host ~]$ podman info
...output omitted...
insecure registries:
  registries: []
registries:
  registries:
  - registry.redhat.io
  - quay.io
  - docker.io
...output omitted...
```

The `podman search` command searches for a matching name image by using the specified list of registries in the `registries.conf` file. By default, Podman searches in all unqualified-search registries.

Note

The `unqualified-search-registries` directive is a list of registries that Podman uses to search or pull an image when a not fully qualified name image such as `registry.redhat.io/ubi9/python-39` is used. You can obtain more information from the `containers-registries.conf(5)` man page.

Depending on the Docker distribution API that is implemented with the registry, some registries might not support the search feature.

Use the `podman search` command to display a list of images on the configured registries that contain the `python-38` package.

```
[user@host ~]$ podman search python-38
NAME                                          DESCRIPTION
registry.access.redhat.com/ubi7/python-38     Python 3.8 platform for building and ru
nning applications
```

```
registry.access.redhat.com/ubi8/python-38     Platform for building and running Pytho
n 3.8 applications

...output omitted...
```

The `registry.access.redhat.com/ubi8/python-38` image seems to match the criteria for the required container.

You can use the `skopeo inspect` command to examine different container image formats from a local directory or a remote registry without downloading the image. This command output displays a list of the available version tags, exposed ports of the containerized application, and metadata of the container image. You use the `skopeo inspect` command to verify that the image contains the required `python-38` package.

```
[user@host ~]$ skopeo inspect docker://registry.access.redhat.com/ubi8/python-38

{

    "Name": "registry.access.redhat.com/ubi8/python-38",

    "Digest": "sha256:c6e522cba2cf2b3ae4a875d5210fb94aa1e7ba71b6cebd902a4f4df73cb090b
8",

    "RepoTags": [

...output omitted...

        "1-68",

        "1-77-source",

        "latest"

...output omitted...

        "name": "ubi8/python-38",

        "release": "86.1648121386",

        "summary": "Platform for building and running Python 3.8 applications",

...output omitted...
```

The `registry.access.redhat.com/ubi8/python-38` image contains the required package and it is based on the required image. You use the `podman pull` command to download the selected image to the local machine. You can use the fully qualified name of the image from the preceding output to avoid ambiguity on container versions or registries.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/python-38

Trying to pull registry.access.redhat.com/ubi8/python-38:latest...

Getting image source signatures

Checking if image destination supports signatures

Copying blob c530010fb61c done
```

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

Then, you use the `podman images` command to display the local images.

```
[user@host ~]$ podman images
REPOSITORY                              TAG     IMAGE ID      CREATED     SIZE
registry.access.redhat.com/ubi8/python-38 latest  a33d92f90990  1 hour ago  901 MB
```

Create a Container Image from a Container File

You are provided with the following container file to create the container image in the `python36-app` directory:

```
[user@host python36-app]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python36
CMD ["/bin/bash", "-c", "sleep infinity"]
```

The previous container file uses the `registry.access.redhat.com/ubi8/ubi:latest` image as the base image. The container file then installs the `python36` package and runs the `sleep infinity` bash command to prevent the container from exiting.

Normally, a container runs a process, and then exits after that process is complete. The `sleep infinity` command prevents the container from exiting, because the process never completes. You can then test, develop, and debug inside the container.

After examining the container file, you use the `podman build` command to build the image. The syntax for the `podman build` command is as follows:

```
[user@host ~]$ podman build -t NAME:TAG DIR
```

NAME
: Name for the new image.

TAG
: Tag for the new image. If the tag is not specified, then the image is automatically tagged as `latest`.

DIR
: Path to the working directory. The container file must be in the working directory. If the working directory is the current directory, then you designate it by a dot (`.`). Use the `-f` flag to specify a different directory from the current one.

In the following example, you use the `podman build` command `-t` option to provide the `python36` name and the `1.0` tag for the new image. The container file is in the current directory.

```
[user@host python36-app]$ podman build -t python36:1.0 .
STEP 1/3: FROM registry.access.redhat.com/ubi8/ubi:latest
STEP 2/3: RUN dnf install -y python36
...output omitted...
STEP 3/3: CMD ["/bin/bash", "-c", "sleep infinity"]
COMMIT python36:1.0
--> 35ab820880f
Successfully tagged localhost/python36:1.0
35ab820880f1708fa310f835407ffc94cb4b4fe2506b882c162a421827b156fc
```

The last line of the preceding output shows the container image ID. Most Podman commands use the first 12 characters of the container image ID to refer to the container image. You can use this short ID or the name of a container or a container image as arguments for most Podman commands.

Note

If a version number is not specified in the tag, then the image is created with the `:latest` tag. If an image name is not specified, then the image and tag fields show the `<none>` string.

You use the `podman images` command to verify that the image is created with the defined name and tag.

```
[user@host ~]$ podman images
REPOSITORY                                 TAG    IMAGE ID     CREATED      SIZE
localhost/python36                         1.0    35ab820880f1 3 minute ago 266 MB
registry.access.redhat.com/ubi8/python-38 latest a33d92f90990 1 hour ago   901 MB
```

You then use the `podman inspect` command to view the low-level information of the container image and verify that its content matches the requirements for the container.

```
[user@host ~]$ podman inspect localhost/python36:1.0
...output omitted...
               "Cmd": [
                    "/bin/bash",
                    "-c",
                    "sleep infinity"
               ],
...output omitted...
               {
                    "created": "2022-04-18T19:47:52.708227513Z",
                    "created_by": "/bin/sh -c dnf install -y python36",
                    "comment": "FROM registry.access.redhat.com/ubi8/ubi:latest"
               },
...output omitted...
```

The output of the `podman inspect` command shows the `registry.access.redhat.com/ubi8/ubi:latest` base image, the `dnf` command to install the `python36` package, and the `sleep infinity` bash command that is executed at runtime to prevent the container from exiting.

Note

The `podman inspect` command output varies from the `python-38` image to the `python36` image, because you created the `/python36` image by adding a layer with changes to the existing `registry.access.redhat.com/ubi8/ubi:latest` base image, whereas the `python-38` image is itself a base image.

Run Containers

Now that you have the required container images, you can use them to run containers. A container can be in one of the following states:

Created
: A container that is created but is not started.

Running
: A container that is running with its processes.

Stopped
: A container with its processes stopped.

Paused
: A container with its processes paused. Not supported for rootless containers.

Deleted
: A container with its processes in a dead state.

The `podman ps` command lists the running containers on the system. Use the `podman ps -a` command to view all containers (that are created, stopped, paused, or running) in the machine.

You use the `podman create` command to create the container to run later. To create the container, you use the ID of the `localhost/python36` container image. You also use the `--name` option to set a name to identify the container. The output of the command is the long ID of the container.

```
[user@host ~]$ podman create --name python36 dd6ca291f097

c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec
```

Note

If you do not set a name for the container with the `podman create` or `podman run` command with the `--name` option, then the `podman` utility assigns a random name to the container.

You then use the `podman ps` and `podman ps -a` commands to verify that the container is created but is not started. You can see information about the `python36` container, such as the short ID, name, and the status of the container, the command that the container runs when started, and the image to create the container.

```
[user@host ~]$ podman create --name python36 dd6ca291f097

c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec

[user@host ~]$ podman ps

CONTAINER ID  IMAGE       COMMAND     CREATED     STATUS      PORTS       NAMES

[user@host ~]$ podman ps -a

CONTAINER ID  IMAGE                   COMMAND               CREATED        STATUS
PORTS       NAMES

c54c7ee28158  localhost/python36:1.0  /bin/bash -c slee...  5 seconds ago  Created
python36
```

Now that you verified that the container is created correctly, you decide to start the container, so you run the `podman start` command. You can use the name or the container ID to start the container. The output of this command is the name of the container.

```
[user@host ~]$ podman start python36

python36

[user@host ~]$ podman ps

CONTAINER ID  IMAGE                   COMMAND               CREATED        STATUS
PORTS       NAMES
```

```
c54c7ee28158  localhost/python36:1.0  /bin/bash -c slee...  6 minutes ago  Up 3 secon
ds ago              python36
```

Run a Container from a Remote Repository

You can use the `podman run` command to create and run the container in one step. The `podman run` command runs a process inside a container, and this process starts the new container.

You use the `podman run` command `-d` option to run a container in *detached mode*, which runs the container in the background instead of in the foreground of the session. In the example of the `python36` container, you do not need to provide a command for the container to run, because the `sleep infinity` command was already provided in the container file that created the image for that container.

To create the `python38` container, you decide to use the `podman run` command and to refer to the `registry.access.redhat.com/ubi8/python-38` image. You also decide to use the `sleep infinity` command to prevent the container from exiting.

```
[user@host ~]$ podman run -d --name python38 \

registry.access.redhat.com/ubi8/python-38 \

sleep infinity


a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462

[user@host ~]$ podman ps

CONTAINER ID  IMAGE                                      COMMAND
CREATED         STATUS             PORTS       NAMES

c54c7ee28158  localhost/python36:1.0                     /bin/bash -c slee...
37 minutes ago  Up 30 minutes ago              python36

a60f71a1dc1b  registry.access.redhat.com/ubi8/python-38:latest  sleep infinity
32 seconds ago  Up 33 seconds ago              python38
```

Important

If you run a container by using the fully qualified image name, but the image is not yet stored locally, then the `podman run` command first pulls the image from the registry and then runs.

Environment Isolation in Containers

Containers isolate the environment of an application. Each container has its own file system, networking, and processes. You can notice the isolation feature when you look at the output of the `ps` command and compare it between the host machine and a running container.

You first run the `ps -ax` command on the local machine, and the command returns an expected result with many processes.

```
[root@host ~]# ps -ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:01 /usr/lib/systemd/systemd --switched-root --system --dese
riali
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
...output omitted...
```

The `podman exec` command executes a command inside a running container. The command takes the name or ID of the container as the first argument and the following arguments as commands to run inside the container. You use the `podman exec` command to view the running processes in the `python36` container. The output from the `ps -ax` command looks different, because it is running different processes from the local machine.

```
[student@host ~]$ podman exec python38 ps -ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /usr/b
in/sleep infinity
      7 ?        R      0:00 ps -ax
```

You can use the `sh -c` command to encapsulate the command to execute in the container. In the following example, the `ps -ax > /tmp/process-data.log` command is interpreted as the command to be executed in the container. If you do not encapsulate the command, then Podman might interpret the greater-than character (`>`) as part of the `podman` command instead of as an argument to the `podman exec` option.

```
[student@host ~]$ podman exec python38 sh -c 'ps -ax > /tmp/process-data.log'
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /usr/b
in/sleep infinity
      7 ?        R      0:00 ps -ax
```

You decide to compare the installed `python` version on the host system with the installed `python` version on the containers.

```
[user@host ~]$ python3 --version
```

```
Python 3.9.10
[user@host ~]$ podman exec python36 python3 --version
Python 3.6.8
[user@host ~]$ podman exec python38 python3 --version
Python 3.8.8
```

File-system Isolation in Containers

Developers can use the file-system isolation feature to write and test applications for different versions of programming languages without the need to use multiple physical or virtual machines.

You create a simple bash script that displays `hello world` on the terminal in the `/tmp` directory.

```
[user@host ~]$ echo "echo 'hello world'" > /tmp/hello.sh
```

The `/tmp/hello.sh` file exists on the host machine, and does not exist on the file system inside the containers. If you try to use the `podman exec` to execute the script, then it gives an error, because the `/tmp/hello.sh` script does not exist in the container.

```
[user@host ~]$ stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19              Blocks: 8          IO Block: 4096   regular file
Device: fc04h/64516d    Inode: 17655599    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ user)   Gid: ( 1000/ user)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2022-04-19 21:47:40.101601412 -0400
Modify: 2022-04-19 21:47:36.497558132 -0400
Change: 2022-04-19 21:47:36.497558132 -0400
 Birth: 2022-04-19 21:45:24.785976758 -0400


[user@host ~]$ podman exec python38 stat /tmp/hello.sh
stat: cannot statx '/tmp/hello.sh': No such file or directory
```

The `podman cp` command copies files and directories between host and container file systems. You can copy the `/tmp/hello.sh` file to the `python38` container with the `podman cp` command.

```
[user@host ~]$ podman cp /tmp/hello.sh python38:/tmp/hello.sh
```

```
[user@host ~]$ podman exec python38 stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19              Blocks: 8          IO Block: 4096   regular file
Device: 3bh/59d  Inode: 12280058    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/ default)   Gid: (    0/    root)
Access: 2022-04-20 01:47:36.000000000 +0000
Modify: 2022-04-20 01:47:36.000000000 +0000
Change: 2022-04-20 02:02:04.732982187 +0000
 Birth: 2022-04-20 02:02:04.732982187 +0000
```

After the script is copied to the container file system, it can be executed from within the container.

```
[user@host ~]$ podman exec python38 bash /tmp/hello.sh
hello world
```

Remove Containers and Images

You can remove containers and images by using the `podman rm` and `podman rmi` commands, respectively. Before you remove a container image, any existing running containers from that image must be removed.

You decide to remove the `python38` container and its related image. If you try to remove the `registry.access.redhat.com/ubi8/python-38` image when the `python38` container exists, then it gives an error.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Error: Image used by a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
: image is in use by a container
```

You must stop the container before you can remove it. To stop a container, use the `podman stop` command.

```
[user@host ~]$ podman stop python38
```

After you stop the container, use the `podman rm` command to remove the container.

```
[user@host ~]$ podman rm python38
```

```
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
```

When the container no longer exists, you can remove the `registry.access.redhat.com/ubi8/python-38` image with the `podman rmi` command.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Untagged: registry.access.redhat.com/ubi8/python-38:latest
Deleted: a33d92f90990c9b1bad9aa98fe017e48f30c711b49527dcc797135352ea57d12
```

Guided Exercise: Deploy Containers

In this exercise, you use container management tools to build an image, run a container, and query the running container environment.

Outcomes

- Configure a container image registry and create a container from an existing image.
- Create a container by using a container file.
- Copy a script from a host machine into containers and run the script.
- Delete containers and images.

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

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

```
[student@workstation ~]$ lab start containers-deploy
```

Instructions

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

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

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

4. Install the `container-tools` meta-package.

```
[student@servera ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
Complete!
```

10. Configure the `registry.lab.example.com` classroom registry in your home directory. Log in to the container registry with `admin` as the user and `redhat321` as the password.
    1. Create the `/home/student/.config/containers` directory.

    ```
    [student@servera ~]$ mkdir -p /home/student/.config/containers
    ```

    2. Create the `/home/student/.config/containers/registries.conf` file with the following contents:

    ```
    unqualified-search-registries = ['registry.lab.example.com']

    [[registry]]
    location = "registry.lab.example.com"
    insecure = true
    ```

    ```
    blocked = false
    ```

    8. Verify that the classroom registry is added.

    ```
    [student@servera ~]$ podman info
    ...output omitted...
    registries:
      registry.lab.example.com:
        Blocked: false
        Insecure: true
        Location: registry.lab.example.com
    ```

```
    MirrorByDigestOnly: false
    Mirrors: null
    Prefix: registry.lab.example.com
  search:
  - registry.lab.example.com
```

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

21. Log in to the classroom registry.

```
[student@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
```

```
Login Succeeded!
```

11. Run the `python38` container in detached mode from an image with the `python 3.8` package and based on the `ubi8` image. The image is hosted on a remote registry.
    1. Search for a *python-38* container in the `registry.lab.example.com` registry.

```
[student@servera ~]$ podman search registry.lab.example.com/
NAME                                          DESCRIPTION
...output omitted...
registry.lab.example.com/ubi8/python-38
registry.lab.example.com/ubi8/httpd-24
```

```
registry.lab.example.com/rhel8/php-74
```

7. Inspect the image.

```
[student@servera ~]$ skopeo inspect \
docker://registry.lab.example.com/ubi8/python-38
...output omitted...
        "description": "Python 3.8 available as container is a base plat
form for building and running various Python 3.8 applications and framew
orks.
```

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

12. Pull the `python-38` container image.

```
[student@servera ~]$ podman pull registry.lab.example.com/ubi8/python-38
Trying to pull registry.lab.example.com/ubi8/python-38:latest...
...output omitted...
```

```
671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

16. Verify that the container is downloaded to the local image repository.

```
[student@servera ~]$ podman images
REPOSITORY                               TAG     IMAGE ID      CREATED
SIZE
```

```
registry.lab.example.com/ubi8/python-38  latest  671cc3cb4298  5 days ag
o  901 MB
```

19. Start the `python38` container.

```
[student@servera ~]$ podman run -d --name python38 \
registry.lab.example.com/ubi8/python-38 sleep infinity
```

```
004756b52d3d3326545f5075594cffa858afd474b903288723a3aa299e72b1af
```

22. Verify that the container was created.

```
[student@servera ~]$ podman ps
CONTAINER ID  IMAGE                                             COMMAND
CREATED              STATUS                 PORTS       NAMES
```

```
004756b52d3d  registry.lab.example.com/ubi8/python-38:latest    sleep in
finity  About a minute ago  Up About a minute ago              python38
```

12. Build a container image called `python39:1.0` from a container file, and use the image to create a container called `python39`.
    1. Examine the container file in the `/home/student/python39` directory.

```
[student@servera ~]$ cat /home/student/python39/Containerfile
FROM registry.lab.example.com/ubi9-beta/ubi:latest
RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms]\nbaseurl = http://content
.example.com/rhel9.0/x86_64/dvd/BaseOS\nenabled = true\ngpgcheck = false
```

```
\nname = Red Hat Enterprise Linux 9.0 BaseOS (dvd)\n[rhel-9.0-for-x86_64
-appstream-rpms]\nbaseurl = http://content.example.com/rhel9.0/x86_64/dv
d/AppStream\nenabled = true\ngpgcheck = false\nname = Red Hat Enterprise
Linux 9.0 Appstream (dvd)'>/etc/yum.repos.d/rhel_dvd.repo
```

```
RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_64-baseos-
rpms --enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3
```

5. Create the container image from the container file.

```
[student@servera ~]$ podman build -t python39:1.0 /home/student/python39
/.
STEP 1/4: FROM registry.lab.example.com/ubi9-beta/ubi:latest
...output omitted...
STEP 2/4: RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms] ...
...output omitted...
STEP 3/4: RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_
64-baseos-rpms --enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python
3
...output omitted...
STEP 4/4: CMD ["/bin/bash", "-c", "sleep infinity"]
...output omitted...
Successfully tagged localhost/python39:1.0
```

```
80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
```

16. Verify that the container image exists in the local image repository.

```
[student@servera ~]$ podman images
REPOSITORY                            TAG    IMAGE ID      CREATED
SIZE
localhost/python39                    1.0    80e68c195925  3 minutes a
go  266 MB
registry.lab.example.com/ubi8/python-38 latest 671cc3cb4298  5 days ago
901 MB
```

```
registry.lab.example.com/ubi9-beta/ubi  latest fca12da1dc30  4 months ag
o   235 MB
```

21. Inspect the `python39` container.

```
[student@servera ~]$ podman inspect localhost/python39:1.0
...output omitted...
```

```
24.            "comment": "FROM registry.lab.example.com/ubi9-beta/ubi:latest"
25. ...output omitted...
26.            "created_by": "/bin/sh -c yum install --disablerepo=*
27. --enablerepo=rhel-9.0-for-x86_64-baseos-rpms --enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3"
28. ...output omitted...
29.            "created_by": "/bin/sh -c #(nop) CMD [\"/bin/bash\", \"-c\", \"sleep infinity\"]",
```

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

30. Create the `python39` container.

```
31. [student@servera ~]$ podman create --name python39 localhost/python39:1.0
```

```
3db4eabe9043224a7bdf195ab5fd810bf95db98dc29193392cef7b94489e1aae
```

32. Start the `python39` container.

```
33. [student@servera ~]$ podman start python39
```

```
python39
```

34. Verify that the container is running.

```
35. [student@servera ~]$ podman ps
36. CONTAINER ID  IMAGE                                            COMMAND               CREATED             STATUS             PORTS       NAMES
37. 004756b52d3d  registry.lab.example.com/ubi8/python-38:latest    sleep infinity        33 minutes ago      Up 33 minutes ago              python38
```

```
3db4eabe9043  localhost/python39:1.0                           /bin/bash -c slee...  About a minute ago  Up 42 seconds ago              python39
```

13. Copy the `/home/student/script.py` script into the `/tmp` directory of the running containers, and run the script on each container.
    1. Copy the `/home/student/script.py` python script into the `/tmp` directory in both containers.

```
2.  [student@servera ~]$ podman cp /home/student/script.py python39:/tmp/scr
    ipt.py
```

```
[student@servera ~]$ podman cp /home/student/script.py python38:/tmp/scr
ipt.py
```

3. Run the Python script in both containers, and then run the Python script on the host.

```
4.  [student@servera ~]$ podman exec -it python39 python3 /tmp/script.py
5.  This script was not run on the correct version of Python
6.  Expected version of Python is 3.8
7.  Current version of python is 3.9
8.  [student@servera ~]$ podman exec -it python38 python3 /tmp/script.py
9.  This script was correctly run on Python 3.8
10. [student@servera ~]$ python3 /home/student/script.py
11. This script was not run on the correct version of Python
12. Expected version of Python is 3.8
```

```
Current version of python is 3.9
```

14. Delete containers and images. Return to `workstation`.

1. Stop both containers.

```
2.  [student@servera ~]$ podman stop python39 python38
3.  ...output omitted...
4.  python38
```

```
python39
```

5. Remove both containers.

```
6.  [student@servera ~]$ podman rm python39 python38
7.  3db4eabe9043224a7bdf195ab5fd810bf95db98dc29193392cef7b94489e1aae
```

```
004756b52d3d3326545f5075594cffa858afd474b903288723a3aa299e72b1af
```

8. Remove both container images.

```
9.  [student@servera ~]$ podman rmi localhost/python39:1.0 \
registry.lab.example.com/ubi8/python-38:latest \
registry.lab.example.com/ubi9-beta/ubi
Untagged: localhost/python39:1.0
Untagged: registry.lab.example.com/ubi8/python-38:latest
Deleted: 80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
Deleted: 219e43f6ff96fd11ea64f67cd6411c354dacbc5cbe296ff1fdbf5b717f01d89a
```

```
Deleted: 671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

16. Return to `workstation`.

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

```
Connection to servera closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish containers-deploy
```

This concludes the section.

Manage Container Storage and Network Resources

Objectives

Provide persistent storage for container data by sharing storage from the container host, and configure a container network.

Manage Container Resources

You can use containers to run a simple process and exit.

You can also configure a container to run a service continuously, such as a database server. If you run a service continuously, you might eventually need to add more resources to the container, such as persistent storage or access to more networks.

You can use different strategies to configure persistent storage for containers:

- For large deployments on an enterprise container platform, such as Red Hat OpenShift, you can use sophisticated storage solutions to provide storage to your containers without knowing the underlying infrastructure.
- For small deployments on a single container host, and without a need to scale, you can create persistent storage from the container host by creating a directory to mount on the running container.

When a container, such as a web server or database server, serves content for clients outside the container host, you must set up a communication channel for those clients to access the content of the container. You can configure *port mapping* to enable communication to a container. With port mapping, the requests that are destined for a port on the container host are forwarded to a port inside the container.

Imagine that you must perform the following tasks:

- Create a containerized database named `db01`, which is based on MariaDB.
- Configure the container port mapping and host firewall to allow traffic on port 3306/tcp.
- Configure the `db01` container to use persistent storage with the appropriate SELinux context.
- Add the appropriate network configuration so that the `client01` container can communicate with the `db01` container by using DNS.

Environment Variables for Containers

Some container images enable passing environment variables to customize the container at creation time. You can use environment variables to set parameters to the container to tailor for your environment without the need to create your own custom image. Usually, you would not modify the container image, because it would add layers to the image, which might be harder to maintain.

You use the `podman run -d registry.lab.example.com/rhel8/mariadb-105` command to run a containerized database, but you notice that the container fails to start.

```
[user@host ~]$ podman run -d registry.lab.example.com/rhel8/mariadb-105 \
--name db01
20751a03897f14764fb0e7c58c7456425859502612417 9de4456d26c49c435ad
[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                              COMMAND         CREA
TED         STATUS                      PORTS       NAMES
20751a03897f  registry.lab.example.com/rhel8/mariadb-105:latest  run-mysqld      29 s
econds ago  Exited (1) 29 seconds ago               db01
```

You use the `podman container logs` command to investigate the reason of the container status.

```
[user@host ~]$ podman container logs db01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[a-zA-Z0-9_]+$')
  MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]+$')
  MYSQL_DATABASE (regex: '^[a-zA-Z0-9_]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]+$')
Or both.
...output omitted...
```

From the preceding output, you determine that the container did not continue to run, because the required environment variables were not passed to the container. So you inspect the `mariadb-105` container image to find more information about the environment variables to customize the container.

```
[user@host ~]$ skopeo inspect docker://registry.lab.example.com/rhel8/mariadb-105
...output omitted...
        "name": "rhel8/mariadb-105",
        "release": "40.1647451927",
        "summary": "MariaDB 10.5 SQL database server",
        "url": "https://access.redhat.com/containers/#/registry.access.redhat.com/rhel8/mariadb-105/
images/1-40.1647451927",
        "usage": "podman run -d -e MYSQL_USER=user -e MYSQL_PASSWORD=pass -e MYSQL_DATABASE=db -p 3306:3306 rhel8/mariadb-105",
        "vcs-ref": "c04193b96a119e176ada62d779bd44a0e0edf7a6",
        "vcs-type": "git",
        "vendor": "Red Hat, Inc.",
...output omitted...
```

The `usage` label from the output provides an example of how to run the image. The `url` label points to a web page in the Red Hat Container Catalog that documents environment variables and other information about how to use the container image.

The documentation for this image shows that the container uses the 3306 port for the database service. The documentation also shows that the following environment variables are available to configure the database service:

Table 13.2. Environment Variables for the `mariadb` Image

Variable	Description
`MYSQL_USER`	Username for the MySQL account to create
`MYSQL_PASSWORD`	Password for the user account
`MYSQL_DATABASE`	Database name
`MYSQL_ROOT_PASSWORD`	Password for the root user (optional)

After examining the available environment variables for the image, you use the `podman run` command `-e` option to pass environment variables to the container, and use the `podman ps` command to verify that it is running.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
```

```
[user@host ~]$ podman ps
CONTAINER ID  IMAGE                                             COMMAND     CREATED
STATUS            PORTS       NAMES
4b8f01be7fd6  registry.lab.example.com/rhel8/mariadb-105:latest  run-mysqld  6 second
s ago  Up 6 seconds ago              db01
```

Container Persistent Storage

By default, when you run a container, all of the content uses the container-based image. Given the ephemeral nature of container images, all of the new data that the user or the application writes is lost after removing the container.

To persist data, you can use host file-system content in the container with the `--volume (-v)` option. You must consider file-system level permissions when you use this volume type in a container.

In the MariaDB container image, the `mysql` user must own the `/var/lib/mysql` directory, the same as if MariaDB was running on the host machine. The directory to mount into the container must have `mysql` as the user and group owner (or the UID and GID of the `mysql` user, if MariaDB is not installed on the host machine). If you run a container as the `root` user, then the UIDs and GIDs on your host machine match the UIDs and GIDs inside the container.

The UID and GID matching configuration does not occur the same way in a rootless container. In a rootless container, the user has root access from within the container, because Podman launches a container inside the user namespace.

You can use the `podman unshare` command to run a command inside the user namespace. To obtain the UID mapping for your user namespace, use the `podman unshare cat` command.

```
[user@host ~]$ podman unshare cat /proc/self/uid_map
         0       1000          1
         1     100000      65536
[user@host ~]$ podman unshare cat /proc/self/gid_map
         0       1000          1
         1     100000      65536
```

The preceding output shows that in the container, the root user (UID and GID of 0) maps to your user (UID and GID of `1000`) on the host machine. In the container, the UID and GID of 1 maps to the UID and GID of 100000 on the host machine. Every UID and GID after 1 increments by 1. For example, the UID and GID of 30 inside a container maps to the UID and GID of 100029 on the host machine.

You use the `podman exec` command to view the `mysql` user UID and GID inside the container that is running with ephemeral storage.

```
[user@host ~]$ podman exec -it db01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
```

You decide to mount the `/home/user/db_data` directory into the `db01` container to provide persistent storage on the `/var/lib/mysql` directory of the container. You then create the `/home/user/db_data` directory, and use the `podman unshare` command to set the user namespace UID and GID of `27` as the owner of the directory.

```
[user@host ~]$ mkdir /home/user/db_data
[user@host ~]$ podman unshare chown 27:27 /home/user/db_data
```

The UID and GID of `27` in the container maps to the UID and GID of `100026` on the host machine. You can verify the mapping by viewing the ownership of the `/home/user/db_data` directory with the `ls` command.

```
[user@host ~]$ ls -l /home/user/
total 0
drwxrwxr-x. 3  100026  100026 18 May  5 14:37 db_data
...output omitted...
```

Now that the correct file-system level permissions are set, you use the `podman run` command `-v` option to mount the directory.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql \
registry.lab.example.com/rhel8/mariadb-105
```

You notice that the `db01` container is not running.

```
[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                           COMMAND         CREA
TED         STATUS                     PORTS       NAMES
dfdc20cf9a7e  registry.lab.example.com/rhel8/mariadb-105:latest  run-mysqld      29 s
econds ago  Exited (1) 29 seconds ago              db01
```

The `podman container logs` command shows a permission error for the `/var/lib/mysql/db_data` directory.

```
[user@host ~]$ podman container logs db01
...output omitted...
---> 16:41:25     Initializing database ...
---> 16:41:25     Running mysql_install_db ...
mkdir: cannot create directory '/var/lib/mysql/db_data': Permission denied
Fatal error Can't create database directory '/var/lib/mysql/db_data'
```

This error happens because of the incorrect SELinux context that is set on the `/home/user/db_data` directory on the host machine.

SELinux Contexts for Container Storage

You must set the `container_file_t` SELinux context type before you can mount the directory as persistent storage to a container. If the directory does not have the `container_file_t` SELinux context, then the container cannot access the directory. You can append the Z option to the argument of the `podman run` command `-v` option to automatically set the SELinux context on the directory.

So you use the `podman run -v /home/user/db_data:/var/lib/mysql:Z` command to set the SELinux context for the `/home/user/db_data` directory when you mount it as persistent storage for the `/var/lib/mysql` directory.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
registry.lab.example.com/rhel8/mariadb-105
```

You then verify that the correct SELinux context is set on the `/home/user/db_data` directory with the `ls` command `-Z` option.

```
[user@host ~]$ ls -Z /home/user/
system_u:object_r:container_file_t:s0:c81,c1009 db_data
...output omitted...
```

Assign a Port Mapping to Containers

To provide network access to containers, clients must connect to ports on the container host that pass the network traffic through to ports in the container. When you map a network port on the container host to a port in the container, the container receives network traffic that is sent to the host network port.

For example, you can map the 13306 port on the container host to the 3306 port on the container for communication with the MariaDB container. Therefore, traffic that is sent to the container host port 13306 would be received by MariaDB that is running in the container.

You use the `podman run` command `-p` option to set a port mapping from the `13306` port from the container host to the `3306` port on the `db01` container.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
registry.lab.example.com/rhel8/mariadb-105
```

Use the `podman port` command `-a` option to show all container port mappings in use. You can also use the `podman port db01` command to show the mapped ports for the `db01` container.

```
[user@host ~]$ podman port -a
1c22fd905120     3306/tcp -> 0.0.0.0:13306
[user@host ~]$ podman port db01
3306/tcp -> 0.0.0.0:13306
```

You use the `firewall-cmd` command to allow port `13306` traffic into the container host machine to redirect to the container.

```
[root@host ~]# firewall-cmd --add-port=13306/tcp --permanent
[root@host ~]# firewall-cmd --reload
```

Important

A rootless container cannot open a privileged port (ports below `1024`) on the container. That is, the `podman run -p 80:8080` command does not normally work for a running rootless container. To map a port on the container host below `1024` to a container port, you must run Podman as root or otherwise adjust the system.

You can map a port above `1024` on the container host to a privileged port on the container, even if you are running a rootless container. The `8080:80` mapping works if the container provides service listening on port `80`.

DNS Configuration in a Container

Podman v4.0 supports two network back ends for containers, Netavark and CNI. Starting with RHEL 9, systems use Netavark by default. To verify which network back end is used, run the following `podman info` command.

```
[user@host ~]$ podman info --format {{.Host.NetworkBackend}}
netavark
```

Note

The `container-tools` meta-package includes the `netavark` and `aardvark-dns` packages. If Podman was installed as a stand-alone package, or if the `container-tools` meta-package was installed later, then the result of the previous command might be `cni`. To change the network back end, set the following configuration in the `/usr/share/containers/containers.conf` file:

```
[network]
...output omitted...
network_backend = "netavark"
```

Existing containers on the host that use the default Podman network cannot resolve each other's hostnames, because DNS is not enabled on the default network.

Use the `podman network create` command to create a DNS-enabled network. You use the `podman network create` command to create the network called `db_net`, and specify the subnet as `10.87.0.0/16` and the gateway as `10.87.0.1`.

```
[user@host ~]$ podman network create --gateway 10.87.0.1 \
--subnet 10.87.0.0/16 db_net
db_net
```

If you do not specify the `--gateway` or `--subnet` options, then they are created with the default values.

The `podman network inspect` command displays information about a specific network. You use the `podman network inspect` command to verify that the gateway and subnet were correctly set and that the new `db_net` network is DNS-enabled.

```
[user@host ~]$ podman network inspect db_net
[
     {
          "name": "db_net",
...output omitted...
          "subnets": [
               {
                    "subnet": "10.87.0.0/16",
                    "gateway": "10.87.0.1"
               }
          ],
...output omitted...
          "dns_enabled": true,
...output omitted...
```

You can add the DNS-enabled `db_net` network to a new container with the `podman run` command `--network` option. You use the `podman run` command `--network` option to create the `db01` and `client01` containers that are connected to the `db_net` network.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
--network db_net \
registry.lab.example.com/rhel8/mariadb-105
[user@host ~]$ podman run -d --name client01 \
--network db_net \
registry.lab.example.com/ubi8/ubi:latest \
sleep infinity
```

Because containers are designed to have only the minimum required packages, the containers might not have the required utilities to test communication, such as the `ping` and `ip` commands. You can install these utilities in the container by using the `podman exec` command.

```
[user@host ~]$ podman exec -it db01 dnf install -y iputils iproute
...output omitted...
[user@host ~]$ podman exec -it client01 dnf install -y iputils iproute
...output omitted...
```

The containers can now ping each other by container name. You test the DNS resolution with the `podman exec` command. The names resolve to IPs within the subnet that was manually set for the `db_net` network.

```
[user@host ~]$ podman exec -it db01 ping -c3 client01
PING client01.dns.podman (10.87.0.4) 56(84) bytes of data.
```

```
64 bytes from 10.87.0.4 (10.87.0.4): icmp_seq=1 ttl=64 time=0.049 ms
...output omitted...
--- client01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.049/0.060/0.072/0.013 ms

[user@host ~]$ podman exec -it client01 ping -c3 db01
PING db01.dns.podman (10.87.0.3) 56(84) bytes of data.
64 bytes from 10.87.0.3 (10.87.0.3): icmp_seq=1 ttl=64 time=0.021 ms
...output omitted...
--- db01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.021/0.040/0.050/0.013 ms
```

You verify that the IP addresses in each container match the DNS resolution with the `podman exec` command.

```
[user@host ~]$ podman exec -it db01 ip a | grep 10.8
    inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
    inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
[user@host ~]$ podman exec -it client01 ip a | grep 10.8
    inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
    inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
```

Multiple Networks to a Single Container

Multiple networks can be connected to a container at the same time to help to separate different types of traffic.

You use the `podman network create` command to create the `backend` network.

```
[user@host ~]$ podman network create backend
```

You then use the `podman network ls` command to view all the Podman networks.

```
[user@host ~]$ podman network ls
NETWORK ID    NAME        DRIVER
```

```
a7fea510a6d1  backend     bridge
fe680efc5276  db01        bridge
2f259bab93aa  podman      bridge
```

The subnet and gateway were not specified with the `podman network create` command `--gateway` and `--subnet` options.

You use the `podman network inspect` command to obtain the IP information of the `backend` network.

```
[user@host ~]$ podman network inspect backend
[
     {
          "name": "backend",
...output omitted...
          "subnets": [
               {
                    "subnet": "10.89.1.0/24",
                    "gateway": "10.89.1.1"
...output omitted...
```

You can use the `podman network connect` command to connect additional networks to a container when it is running. You use the `podman network connect` command to connect the `backend` network to the `db01` and `client01` containers.

```
[user@host ~]$ podman network connect backend db01
[user@host ~]$ podman network connect backend client01
```

Important

If a network is not specified with the `podman run` command, then the container connects to the default network. The default network uses the `slirp4netns` network mode, and the networks that you create with the `podman network create` command use the *bridge* network mode. If you try to connect a bridge network to a container by using the `slirp4netns` network mode, then the command fails:

```
Error: "slirp4netns" is not supported: invalid network mode
```

You use the `podman inspect` command to verify that both networks are connected to each container and to display the IP information.

```
[user@host ~]$ podman inspect db01
...output omitted...
                    "backend": {
                         "EndpointID": "",
                         "Gateway": "10.89.1.1",
                         "IPAddress": "10.89.1.4",
...output omitted...
                    },
                    "db_net": {
                         "EndpointID": "",
                         "Gateway": "10.87.0.1",
                         "IPAddress": "10.87.0.3",
...output omitted...
[user@host ~]$ podman inspect client01
...output omitted...
                    "backend": {
                         "EndpointID": "",
                         "Gateway": "10.89.1.1",
                         "IPAddress": "10.89.1.5",
...output omitted...
                    },
                    "db_net": {
                         "EndpointID": "",
                         "Gateway": "10.87.0.1",
                         "IPAddress": "10.87.0.4",
...output omitted...
```

The `client01` container can now communicate with the `db01` container on both networks. You use the `podman exec` command to ping both networks on the `db01` container from the `client01` container.

```
[user@host ~]$ podman exec -it client01 ping -c3 10.89.1.4 | grep 'packet loss'
```

```
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
[user@host ~]$ podman exec -it client01 ping -c3 10.87.0.3 | grep 'packet loss'
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
```

Guided Exercise: Manage Container Storage and Network Resources

In this exercise, you pass environment variables to a container during creation, mount persistent storage to a container, create and connect multiple container networks, and expose container ports from the host machine.

Outcomes

- Create container networks and connect them to containers.
- Troubleshoot failed containers.
- Pass environment variables to containers during creation.
- Create and mount persistent storage to containers.
- Map host ports to ports inside containers.

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

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

```
[student@workstation ~]$ lab start containers-resources
```

Instructions

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

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

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

4. Create the `frontend` container network. Create the `db_client` and `db_01` containers and connect them to the `frontend` network.
    1. Use the `podman network create` command `--subnet` and `--gateway` options to create the `frontend` network with the `10.89.1.0/24` subnet and the `10.89.1.1` gateway.

    ```
    [student@servera ~]$ podman network create --subnet 10.89.1.0/24 \
    --gateway 10.89.1.1 frontend
    ```

    ```
    frontend
    ```

    4. Log in to the `registry.lab.example.com` registry.

    ```
    [student@servera ~]$ podman login registry.lab.example.com
    Username: admin
    Password: redhat321
    ```

    ```
    Login Succeeded!
    ```

    8. Start a container named `db_client` in the background, and connect it to the `frontend` network. To be able to install packages in the `db_client` container, mount the `/etc/yum.repos.d` DNF repositories directory at the `/etc/yum.repos.d` container path. Run the `sleep infinity` command in the `db_client` container to prevent the container from exiting. Use the `registry.lab.example.com/ubi9-beta/ubi` image.

    ```
    [student@servera ~]$ podman run -d --name db_client \
    --network frontend \
    -v /etc/yum.repos.d:/etc/yum.repos.d \
    registry.lab.example.com/ubi9-beta/ubi \
    sleep infinity
    ```

    ```
    e20dfed7e392abe4b7bea3c25e9cb17ef95d16af9cedd50d68f997a663ba6c15
    ```

    14. Start in the background a container named `db_01` that is connected to the `frontend` network. Use the `registry.lab.example.com/rhel8/mariadb-105` image.

    ```
    [student@servera ~]$ podman run -d --name db_01 --network frontend \
    ```

```
registry.lab.example.com/rhel8/mariadb-105
```

```
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
```

17. View all containers.

```
[student@servera ~]$ podman ps -a
CONTAINER ID  IMAGE                                          COMMAND
CREATED       STATUS                  PORTS       NAMES
e20dfed7e392  registry.lab.example.com/ubi8/ubi:latest       sleep i
nfinity  56 seconds ago  Up 56 seconds ago             db_client
```

```
3e767ae6eea4  registry.lab.example.com/rhel8/mariadb-105:latest  run-mys
qld  1 second ago  Exited (1) 1 second ago             db_01
```

5. Troubleshoot the `db_01` container and determine why it is not running. Re-create the `db_01` container by using the required environment variables.
    1. View the container logs and determine why the container exited.

```
[student@servera ~]$ podman container logs db_01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[a-zA-Z0-9_]+$')
  MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]+$')
  MYSQL_DATABASE (regex: '^[a-zA-Z0-9_]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>,.?;:|]+$')
Or both.
```

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

11. Remove the `db_01` container and create it again with environment variables. Provide the required environment variables.

```
[student@servera ~]$ podman rm db_01
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
```

```
948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
```

21. View the current running containers.

```
[student@servera ~]$ podman ps
CONTAINER ID  IMAGE                                           COMMAND
CREATED         STATUS             PORTS       NAMES
e20dfed7e392  registry.lab.example.com/ubi8/ubi:latest          sleep i
nfinity  56 seconds ago  Up 56 seconds ago              db_client
```

```
948c4cd767b5  registry.lab.example.com/rhel8/mariadb-105:latest  run-mys
qld      11 seconds ago  Up 12 seconds ago              db_01
```

6. Create persistent storage for the containerized MariaDB service, and map the local machine 13306 port to the 3306 port in the container. Allow traffic to the 13306 port on the `servera` machine.
   1. Create the `/home/student/databases` directory on the `servera` machine.

   ```
   [student@servera ~]$ mkdir /home/student/databases
   ```

   2. Obtain the `mysql` UID and GID from the `db_01` container, and then remove the `db01` container.

   ```
   [student@servera ~]$ podman exec -it db_01 grep mysql /etc/passwd
   mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
   [student@servera ~]$ podman stop db_01
   db_01
   [student@servera ~]$ podman rm db_01
   ```

   ```
   948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
   ```

   8. Run the `chown` command inside the container namespace, and set the user and group owner to `27` on the `/home/student/database` directory.

   ```
   [student@servera ~]$ podman unshare chown 27:27 /home/student/databases/
   [student@servera ~]$ ls -l /home/student/
   ```

```
total 0
```

```
drwxr-xr-x. 2 100026 100026 6 May  9 17:40 databases
```

12. Create the `db_01` container, and mount the `/home/student/databases` directory from the `servera` machine to the `/var/lib/mysql` directory inside the `db_01` container. Use the z option to apply the required SELinux context.

```
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/student/databases:/var/lib/mysql:Z \
-p 13306:3306 \
```

```
registry.lab.example.com/rhel8/mariadb-105
```

21. Install the `mariadb` package in the `db_client` container.

```
[student@servera ~]$ podman exec -it db_client dnf install -y mariadb
...output omitted...
```

```
Complete!
```

24. Create the `crucial_data` table in the `dev_db` database in the `db_01` container from the `db_client` container.

```
[student@servera ~]$ podman exec -it db_client mysql -u dev1 -p -h db_01
Enter password: devpass
...output omitted...
MariaDB [(none)]> USE devdb;
Database changed
MariaDB [devdb]> CREATE TABLE crucial_data(column1 int);
Query OK, 0 rows affected (0.036 sec)

MariaDB [devdb]> SHOW TABLES;
```

```
+-----------------+
| Tables_in_devdb |
+-----------------+
| crucial_data    |
+-----------------+
1 row in set (0.001 sec)

MariaDB [devdb]> **quit**
```

```
Bye
```

42. Allow port 13306 traffic in the firewall on the `servera` machine.

```
[student@servera ~]$ sudo firewall-cmd --add-port=13306/tcp --permanent
[sudo] password for student: student
success
[student@servera ~]$ sudo firewall-cmd --reload
```

```
success
```

47. Open a second terminal on the `workstation` machine and use the MariaDB client to connect to the `servera` machine on port `13306`, to show tables inside the `db_01` container that are stored in the persistent storage.

```
[student@workstation ~]$ mysql -u dev1 -p -h servera --port 13306 \
devdb -e 'SHOW TABLES';
Enter password: devpass
+-----------------+
| Tables_in_devdb |
+-----------------+
| crucial_data    |
```

```
+-----------------+
```

7. Create a second container network called `backend`, and connect the `backend` network to the `db_client` and `db_01` containers. Test network connectivity and DNS resolution between the containers.

1. Create the `backend` network with the `10.90.0.0/24` subnet and the `10.90.0.1` gateway.

```
[student@servera ~]$ podman network create --subnet 10.90.0.0/24 \
--gateway 10.90.0.1 backend
```

```
backend
```

4. Connect the `backend` container network to the `db_client` and `db_01` containers.

```
[student@servera ~]$ podman network connect backend db_client
```

```
[student@servera ~]$ podman network connect backend db_01
```

6. Obtain the IP addresses of the `db_01` container.

```
[student@servera ~]$ podman inspect db_01
...output omitted...
               "Networks": {
                    "backend": {
                         "EndpointID": "",
                         "Gateway": "10.90.0.1",
                         "IPAddress": "10.90.0.3",
...output omitted...
                    "frontend": {
                         "EndpointID": "",
                         "Gateway": "10.89.1.1",
                         "IPAddress": "10.89.1.5",
```

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

19. Install the `iputils` package in the `db_client` container.

```
[student@servera ~]$ podman exec -it db_client dnf install -y iputils
...output omitted...
```

```
Complete!
```

22. Ping the `db_01` container name from the `db_client` container.

```
[student@servera ~]$ **podman exec -it db_client ping -c4 db_01**
PING db_01.dns.podman (10.90.0.3) 56(84) bytes of data.
...output omitted...
--- db_01.dns.podman ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3048ms
```

```
rtt min/avg/max/mdev = 0.043/0.049/0.054/0.004 ms
```

28. Exit the `servera` machine.

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

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

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish containers-resources
```

This concludes the section.

Manage Containers as System Services

Objectives

Configure a container as a `systemd` service, and configure a container service to start at boot time.

Manage Small Container Environments with systemd Units

You can run a container to complete a system task or to obtain the output of a series of commands. You also might want to run containers that run a service indefinitely, such as web servers or databases. In a traditional environment, a privileged user typically configures these services to run at system boot, and manages them with the `systemctl` command.

As a regular user, you can create a `systemd` unit to configure your rootless containers. You can use this configuration to manage your container as a regular system service with the `systemctl` command.

Managing containers based on `systemd` units is mainly useful for basic and small deployments that do not need to scale. For more sophisticated scaling and orchestration of many container-based applications and services, you can use an enterprise orchestration platform that is based on Kubernetes, such as Red Hat OpenShift Container Platform.

To discuss the topics in this lecture, imagine the following scenario.

As a system administrator, you are tasked to configure the `webserver1` container that is based on the `http24` container image to start at system boot. You must also mount the `/app-artifacts` directory for the web server content and map the 8080 port from the local machine to the container. Configure the container to start and stop with `systemctl` commands.

Requirements for systemd User Services

As a regular user, you can enable a service with the `systemctl` command. The service starts when you open a session (graphical interface, text console, or SSH), and it stops when you close the last session. This behavior differs from a system service, which starts when the system boots and stops when the system shuts down.

By default, when you create a user account with the `useradd` command, the system uses the next available ID from the regular user ID range. The system also reserves a range of IDs for the user's containers in the `/etc/subuid` file. If you create a user account with the `useradd` command `--system` option, then the system does not reserve a range for the user containers. As a consequence, you cannot start rootless containers with system accounts.

You decide to create a dedicated user account to manage containers. You use the `useradd` command to create the `appdev-adm` user, and use `redhat` as the password.

```
[user@host ~]$ sudo useradd appdev-adm
[user@host ~]$ sudo passwd appdev-adm
Changing password for user appdev-adm.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

You then use the `su` command to switch to the `appdev-adm` user, and you start to use the `podman` command.

```
[user@host ~]$ su appdev-adm
Password: redhat
[appdev-adm@host ~]$ podman info
ERRO[0000] XDG_RUNTIME_DIR directory "/run/user/1000" is not owned by the current use
r
[appdev-adm@host ~]$
```

Podman is a stateless utility and requires a full login session. Podman must be used within an SSH session, and cannot be used in a `sudo` or an `su` shell. So you exit the `su` shell and log in to the machine via SSH.

```
[appdev-adm@host ~]$ exit
[user@host ~]$ exit
[user@example ~]$ ssh appdev-adm@host
[appdev-adm@host ~]$
```

You then configure the container registry and authenticate with your credentials. You run the `http` container with the following command.

```
[appdev-adm@host ~]$ podman run -d --name webserver1 -p 8080:8080 -v \
~/app-artifacts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24
cde4a3d8c9563fd50cc39de8a4873dcf15a7e881ba4548d5646760eae7a35d81
```

```
[appdev-adm@host ~]$ podman ps

CONTAINER ID  IMAGE                                            COMMAND
CREATED        STATUS            PORTS                   NAMES

cde4a3d8c956  registry.access.redhat.com/ubi8/httpd-24:latest  /usr/bin/run-http...
4 seconds ago  Up 5 seconds ago  0.0.0.0:8080->8080/tcp  webserver1
```

Note

Remember to provide the right access to the directory that you mount from the host file system to the container. For any error when running a container, you can use the `podman container logs` command for troubleshooting.

Create systemd User Files for Containers

You can manually define `systemd` services in the `~/.config/systemd/user/` directory. The file syntax for user services is the same as for the system services files. For more details, review the `systemd.unit`(5) and `systemd.service`(5) man pages.

Use the `podman generate systemd` command to generate `systemd` service files for an existing container. The `podman generate systemd` command uses a container as a model to create the configuration file.

The `podman generate systemd` command `--new` option instructs the `podman` utility to configure the `systemd` service to create the container when the service starts, and to delete the container when the service stops.

Important

Without the `--new` option, the `podman` utility configures the service unit file to start and stop the existing container without deleting it.

You use the `podman generate systemd` command with the `--name` option to display the `systemd` service file that is modeled for the `webserver1` container.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1
...output omitted...


ExecStart=/usr/bin/podman start webserver1


ExecStop=/usr/bin/podman stop -t 10 webserver1
```

```
ExecStopPost=/usr/bin/podman stop -t 10 webserver1
...output omitted...
```

On start, the `systemd` daemon executes the `podman start` command to start the existing container.
On stop, the `systemd` daemon executes the `podman stop` command to stop the container. Notice that the `systemd` daemon does not delete the container on this action.

You then use the previous command with the addition of the `--new` option to compare the `systemd` configuration.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new
...output omitted...
ExecStartPre=/bin/rm -f %t/%n.ctr-id
ExecStart=/usr/bin/podman run --cidfile=%t/%n.ctr-id --cgroups=no-conmon --rm --sdnot
ify=conmon --replace -d --name webserver1 -p 8080:8080 -v /home/appdev-adm/app-artifa

cts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24


ExecStop=/usr/bin/podman stop --ignore --cidfile=%t/%n.ctr-id


ExecStopPost=/usr/bin/podman rm -f --ignore --cidfile=%t/%n.ctr-id
...output omitted...
```

On starting, the `systemd` daemon executes the `podman run` command to create and then start a new container. This action uses the `podman run` command `--rm` option, which removes the container on stopping.
On stopping, `systemd` executes the `podman stop` command to stop the container.

After `systemd` stops the container, `systemd` removes it by using the `podman rm -f` command.

You verify the output of the `podman generate systemd` command, and run the previous command with the `--files` option to create the `systemd` user file in the current directory. Because the `webserver1` container uses persistent storage, you choose to use the `podman generate systemd` command with the `--new` option. You then create the `~/.config/systemd/user/` directory and move the file to this location.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new --files
/home/appdev-adm/container-webserver1.service
[appdev-adm@host ~]$ mkdir -p ~/.config/systemd/user/
[appdev-adm@host ~]$ mv container-webserver1.service ~/.config/systemd/user/
```

Manage systemd User Files for Containers

Now that you created the `systemd` user file, you can use the `systemctl` command `--user` option to manage the `webserver1` container.

First, you reload the `systemd` daemon to make the `systemctl` command aware of the new user file. You use the `systemctl --user start` command to start the `webserver1` container. Use the name of the generated `systemd` user file for the container.

```
[appdev-adm@host ~]$ systemctl --user daemon-reload

[appdev-adm@host ~]$ systemctl --user start container-webserver1.service

[appdev-adm@host ~]$ systemctl --user status container-webserver1.service

● container-webserver1.service - Podman container-webserver1.service

     Loaded: loaded (/home/appdev-adm/.config/systemd/user/container-webserver1.servi
ce; disabled; vendor preset: disabled)

     Active: active (running) since Thu 2022-04-28 21:22:26 EDT; 18s ago

       Docs: man:podman-generate-systemd(1)

    Process: 31560 ExecStartPre=/bin/rm -f /run/user/1003/container-webserver1.servic
e.ctr-id (code=exited, status=0/SUCCESS)

   Main PID: 31600 (conmon)

...output omitted...

[appdev-adm@host ~]$ podman ps

CONTAINER ID  IMAGE                                            COMMAND
CREATED         STATUS             PORTS                   NAMES

18eb00f42324  registry.access.redhat.com/ubi8/httpd-24:latest  /usr/bin/run-http...
28 seconds ago  Up 29 seconds ago  0.0.0.0:8080->8080/tcp  webserver1

Created symlink /home/appdev-adm/.config/systemd/user/default.target.wants/container-
webserver1.service → /home/appdev-adm/.config/systemd/user/container-webserver1.servi
ce.
```

Important

When you configure a container with the `systemd` daemon, the daemon monitors the container status and restarts the container if it fails. Do not use the `podman` command to start or stop these containers. Doing so might interfere with the `systemd` daemon monitoring.

The following table summarizes the directories and commands that are used between `systemd` system and user services.

Table 13.3. Comparing System and User Services

Storing custom unit files	System services	`/etc/systemd/system/unit.service`
	User services	`~/.config/systemd/user/unit.service`
Reloading unit files	System services	`# systemctl daemon-reload`
	User services	`$ systemctl --user daemon-reload`

Starting and stopping a service	System services	`# systemctl start UNIT`
		`# systemctl stop UNIT`

	User services	`$ systemctl --user start UNIT`
		`$ systemctl --user stop UNIT`

Starting a service when the machine starts	System services	`# systemctl enable UNIT`

	User services	`$ loginctl enable-linger`
		`$ systemctl --user enable UNIT`

Configure Containers to Start at System Boot

At this point, the `systemd` service configuration is ready to run a container for a given user. However, the `systemd` service stops the container after a certain time if the user logs out from the system. This behavior occurs because the `systemd` service unit was created with the `--user` option, which starts a service at user login and stops it at user logout.

You can change this default behavior, and force your enabled services to start with the server and stop during the shutdown, by running the `loginctl enable-linger` command.

You use the `loginctl` command to configure the `systemd` user service to persist after the last user session of the configured service closes. You then verify the successful configuration with the `loginctl show-user` command.

```
[user@host ~]$ loginctl show-user appdev-adm
...output omitted...
Linger=no
[user@host ~]$ loginctl enable-linger
[user@host ~]$ loginctl show-user appdev-adm
...output omitted...
Linger=yes
```

To revert the operation, use the `loginctl disable-linger` command.

Manage Containers as Root with systemd

You can also configure containers to run as root and manage them with `systemd` service files. One advantage of this approach is that you can configure the service files to work the same as common `systemd` unit files, rather than as a particular user.

The procedure to set the service file as root is similar to the previously outlined procedure for rootless containers, with the following exceptions:

- Do not create a dedicated user for container management.
- The service file must be in the `/etc/systemd/system` directory instead of in the `~/.config/systemd/user` directory.
- You manage the containers with the `systemctl` command without the `--user` option.
- Do not run the `loginctl enable-linger` command as the `root` user.

For a demonstration, see the YouTube video from the Red Hat Videos channel that is listed in the References at the end of this section.

Guided Exercise: Manage Containers as System Services

In this exercise, you configure a container to manage it as a `systemd` service, and use `systemctl` commands to manage that container so that it automatically starts when the host machine starts.

Outcomes

- Create `systemd` service files to manage a container.
- Configure a container so you can manage it with `systemctl` commands.
- Configure a user account for `systemd` user services to start a container when the host machine starts.

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

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

```
[student@workstation ~]$ lab start containers-services
```

Instructions

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

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

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

4. Create a user account called `contsvc` and use `redhat` as the password. Use this user account to run containers as `systemd` services.
    1. Create the `contsvc` user. Set `redhat` as the password for the `contsvc` user.

```
[student@servera ~]$ sudo useradd contsvc
[sudo] password for student: student
[student@servera ~]$ sudo passwd contsvc
Changing password for user contsvc.
```

```
6. New password: redhat
7. BAD PASSWORD: The password is shorter than 8 characters
8. Retype new password: redhat
```

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

9. To manage the `systemd` user services with the `contsvc` account, you must log in directly as the `contsvc` user. You cannot use the `su` and `sudo` commands to create a session with the `contsvc` user.

   Return to the `workstation` machine as the `student` user, and then log in as the `contsvc` user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh contsvc@servera
...output omitted...
[contsvc@servera ~]$
```

5. Configure access to the `registry.lab.example.com` classroom registry in your home directory. Use the `/tmp/containers-services/registries.conf` file as a template.
   1. Create the `~/.config/containers/` directory.

```
[contsvc@servera ~]$ mkdir -p ~/.config/containers/
```

   2. The `lab` script prepares the `registries.conf` file in the `/tmp/containers-services/` directory. Copy that file to the `~/.config/containers/` directory.

```
3. [contsvc@servera ~]$ cp /tmp/containers-services/registries.conf \
```

```
~/.config/containers/
```

   4. Verify that you can access the `registry.lab.example.com` registry. If everything works as expected, then the command should list some images.

```
5. [contsvc@servera ~]$ podman search ubi
```

```
NAME                                        DESCRIPTION
registry.lab.example.com/ubi7/ubi
registry.lab.example.com/ubi8/ubi
```

```
registry.lab.example.com/ubi9-beta/ubi
```

6. Use the `/home/contsvc/webcontent/html/` directory as persistent storage for the web server container. Create the `index.html` test page with the `Hello World` line inside the directory.
    1. Create the `~/webcontent/html/` directory.

    ```
    [contsvc@servera ~]$ mkdir -p ~/webcontent/html/
    ```

    2. Create the `index.html` file and add the `Hello World` line.

    ```
    [contsvc@servera ~]$ echo "Hello World" > ~/webcontent/html/index.html
    ```

    3. Verify that the permission for others is set to `r-x` in the `webcontent/html` directory, and is set to `r--` in the `index.html` file. The container uses a non-privileged user that must be able to read the `index.html` file.

    ```
    [contsvc@servera ~]$ ls -ld webcontent/html/
    drwxr-xr-x. 2 contsvc contsvc 24 Aug 28 04:56 webcontent/html/
    [contsvc@servera ~]$ ls -l webcontent/html/index.html
    ```

    ```
    -rw-r--r--. 1 contsvc contsvc 12 Aug 28 04:56 webcontent/html/index.html
    ```

7. Use the `registry.lab.example.com/rhel8/httpd-24:1-163` image to run a container called `webapp` in detached mode. Redirect the 8080 port on the local host to the container 8080 port. Mount the `~/webcontent` directory from the host to the `/var/www` directory in the container.
    1. Log in to the `registry.lab.example.com` registry as the `admin` user with `redhat321` as the password.

    ```
    [contsvc@servera ~]$ podman login registry.lab.example.com
    Username: admin
    Password: redhat321
    ```

```
Login Succeeded!
```

5. Use the `registry.lab.example.com/rhel8/httpd-24:1-163` image to run a container called `webapp` in detached mode. Use the `-p` option to map the `8080` port on `servera` to the `8080` port in the container. Use the `-v` option to mount the `~/webcontent` directory on `servera` to the `/var/www` directory in the container.

```
6.  [contsvc@servera ~]$ podman run -d --name webapp -p 8080:8080 -v \
7.  ~/webcontent:/var/www:Z registry.lab.example.com/rhel8/httpd-24:1-163
8.  750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06
```

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

9. Verify that the web service is working on port 8080.

```
10. [contsvc@servera ~]$ curl http://localhost:8080
```

```
Hello World
```

8. Create a `systemd` service file to manage the `webapp` container with `systemctl` commands. Configure the `systemd` service so that when you start the service, the `systemd` daemon creates a container. After you finish the configuration, stop and then delete the `webapp` container. Remember that the `systemd` daemon expects that the container does not exist initially.
   1. Create and change to the `~/.config/systemd/user/` directory.

```
2.  [contsvc@servera ~]$ mkdir -p ~/.config/systemd/user/
```

```
[contsvc@servera ~]$ cd ~/.config/systemd/user
```

3. Create the unit file for the `webapp` container. Use the `--new` option so that `systemd` creates a container when starting the service, and deletes the container when stopping the service.

```
4.  [contsvc@servera user]$ podman generate systemd --name webapp --files --
    new
```

```
/home/contsvc/.config/systemd/user/container-webapp.service
```

5. Stop and then delete the `webapp` container.

```
6.  [contsvc@servera user]$ podman stop webapp
```

```
webapp
[contsvc@servera user]$ podman rm webapp
750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06
[contsvc@servera user]$ podman ps -a
```

```
CONTAINER ID  IMAGE       COMMAND     CREATED     STATUS      PORTS
NAMES
```

9. Reload the `systemd` daemon configuration, and then enable and start your new `container-webapp` user service. Verify the `systemd` service configuration, stop and start the service, and display the web server response and the container status.
   1. Reload the configuration to recognize the new unit file.

      ```
      [contsvc@servera user]$ systemctl --user daemon-reload
      ```

   2. Enable and start the `container-webapp` service.

      ```
      [contsvc@servera user]$ systemctl --user enable --now container-webapp
      ```

      ```
      Created symlink /home/contsvc/.config/systemd/user/default.target.wants/
      container-webapp.service → /home/contsvc/.config/systemd/user/container-
      webapp.service.
      ```

   4. Verify that the web server responds to requests.

      ```
      [contsvc@servera user]$ curl http://localhost:8080
      ```

      ```
      Hello World
      ```

   6. Verify that the container is running.

      ```
      [contsvc@servera user]$ podman ps
      CONTAINER ID  IMAGE                                         COMMAND
      CREATED        STATUS            PORTS                        NAMES
      ```

      ```
      3e996db98071  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/r
      un-http...  3 minutes ago  Up 3 minutes ago  0.0.0.0:8080->8080/tcp  web
      app
      ```

      Use the container ID information to confirm that the `systemd` daemon creates a container when you restart the service.

9. Stop the `container-webapp` service, and confirm that the container no longer exists. When you stop the service, the `systemd` daemon stops and then deletes the container.

```
10. [contsvc@servera user]$ systemctl --user stop container-webapp
11. [contsvc@servera user]$ podman ps --all
```

```
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
```

12. Start the `container-webapp` service, and then confirm that the container is running.

    The container ID is different, because the `systemd` daemon creates a container with the start instruction, and deletes the container with the stop instruction.

```
[contsvc@servera user]$ systemctl --user start container-webapp
[contsvc@servera user]$ podman ps
CONTAINER ID  IMAGE                                                    COMMAND
CREATED       STATUS             PORTS                    NAMES
4584b4df514c  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/r
un-http...  6 seconds ago  Up 7 seconds ago  0.0.0.0:8080->8080/tcp  web
app
```

10. Ensure that the services for the `contsvc` user start at system boot. When done, restart the `servera` machine.
    1. Run the `loginctl enable-linger` command.

```
[contsvc@servera user]$ loginctl enable-linger
```

2. Confirm that the `Linger` option is set for the `contsvc` user.

```
3. [contsvc@servera user]$ loginctl show-user contsvc
4. ...output omitted...
```

```
Linger=yes
```

5. Switch to the `root` user, and then use the `systemctl reboot` command to restart `servera`.

```
6. [contsvc@servera user]$ su -
```

```
Password: redhat
Last login: Fri Aug 28 07:43:40 EDT 2020 on pts/0
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
```

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

11. When the `servera` machine is up again, log in to `servera` as the `contsvc` user. Verify that the `systemd` daemon started the `webapp` container, and that the web content is available.

    1. Log in to `servera` as the `contsvc` user.

```
[student@workstation ~]$ ssh contsvc@servera
```

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

    3. Verify that the container is running.

```
[contsvc@servera ~]$ podman ps
CONTAINER ID  IMAGE                                                  COMMAND
CREATED        STATUS            PORTS                         NAMES
```

```
6c325bf49f84  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/r
un-http...  2 minutes ago  Up 2 minutes ago  0.0.0.0:8080->8080/tcp  web
app
```

    6. Access the web content.

```
[contsvc@servera ~]$ curl http://localhost:8080
```

```
Hello World
```

    8. Return to the `workstation` machine as the `student` user.

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

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

Finish

On the `workstation` machine, run the `lab finish containers-services` script to complete this exercise.

```
[student@workstation ~]$ lab finish containers-services
```

This concludes the section.

Summary

- Containers provide a lightweight way to distribute and run an application with its dependencies so that it does not conflict with installed software on the host.
- Containers run from container images that you can download from a container registry or create yourself.
- You can use container files with instructions to build a customized container image.
- Podman, which Red Hat Enterprise Linux provides, directly runs and manages containers and container images on a single host.
- Containers can be run as `root`, or as non-privileged rootless containers for increased security.
- You can map network ports on the container host to pass traffic to services that run in its containers.
- You can use environment variables to configure the software in containers at build time.
- Although container storage is temporary, you can attach persistent storage to a container by using the contents of a directory on the container host, for example.
- You can configure a `systemd` unit file to automatically run containers when the system starts.

Chapter 14. Comprehensive Review

Abstract

Goal	Review tasks from the *Red Hat System Administration II* course.
Objectives	• Review tasks from the *Red Hat System Administration II* course.
Sections	• Comprehensive Review
Labs	• Fix Boot Issues and Maintain Servers • Configure and Manage File Systems and Storage • Configure and Manage Server Security • Run Containers

Comprehensive Review

Objectives

Demonstrate knowledge and skills learned in *Red Hat System Administration II*.

Reviewing Red Hat System Administration II

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

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

Chapter 1, *Improve Command-line Productivity*

Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various Red Hat Enterprise Linux utilities.

- Run commands more efficiently by using advanced features of the Bash shell, shell scripts, and various Red Hat Enterprise Linux utilities.
- Run repetitive tasks with `for` loops, evaluate exit codes from commands and scripts, run tests with operators, and create conditional structures with `if` statements.
- Create regular expressions to match data, apply regular expressions to text files with the `grep` command, and use `grep` to search files and data from piped commands.

Chapter 2, *Schedule Future Tasks*

Schedule tasks to execute at a specific time and date.

- Set up a command to run once at a future time.
- Schedule commands to run on a repeating schedule with a user's `crontab` file.
- Schedule commands to run on a repeating schedule with the system `crontab` file and directories.
- Enable and disable `systemd` timers, and configure a timer that manages temporary files.

Chapter 3, *Analyze and Store Logs*

Locate and accurately interpret system event logs for troubleshooting purposes.

- Describe the basic Red Hat Enterprise Linux logging architecture to record events.
- Interpret events in the relevant syslog files to troubleshoot problems or to review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events that are recorded by the system journal and logs.

Chapter 4, *Archive and Transfer Files*

Archive and copy files from one system to another.

- Archive files and directories into a compressed file with `tar`, and extract the contents of an existing `tar` archive.
- Transfer files to or from a remote system securely with SSH.
- Efficiently and securely synchronize the contents of a local file or directory with a remote server copy.

Chapter 5, *Tune System Performance*

Improve system performance by setting tuning parameters and adjusting the scheduling priority of processes.

- Optimize system performance by selecting a tuning profile that the `tuned` daemon manages.
- Prioritize or deprioritize specific processes, with the `nice` and `renice` commands.

Chapter 6, *Manage SELinux Security*

Protect and manage server security by using SELinux.

- Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system.
- Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command, and apply the context defined by the SELinux policy to files and directories with the `restorecon` command.
- Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult `man` pages that end with `_selinux` to find useful information about SELinux Booleans.
- Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.

Chapter 7, *Manage Basic Storage*

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

- Create storage partitions, format them with file systems, and mount them for use.

- Create and manage swap spaces to supplement physical memory.

Chapter 8, *Manage Storage Stack*

Create and manage logical volumes that contain file systems or swap spaces from the command line.

- Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information.
- Analyze the multiple storage components that make up the layers of the storage stack.

Chapter 9, *Access Network-Attached Storage*

Access network-attached storage with the NFS protocol.

- Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command.
- Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.

Chapter 10, *Control the Boot Process*

Manage the boot process to control offered services and to troubleshoot and repair problems.

- Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target.
- Log in to a system and change the root password when the current root password is lost.
- Manually repair file-system configuration or corruption issues that stop the boot process.

Chapter 11, *Manage Network Security*

Control network connections to services with the system firewall and SELinux rules.

- Accept or reject network connections to system services with `firewalld` rules.
- Verify that network ports have the correct SELinux type for services to bind to them.

Chapter 12, *Install Red Hat Enterprise Linux*

Install Red Hat Enterprise Linux on servers and virtual machines.

- Install Red Hat Enterprise Linux on a server.
- Explain Kickstart concepts and architecture, create a Kickstart file with the Kickstart Generator website, modify an existing Kickstart file with a text editor and check its syntax with `ksvalidator`, publish a Kickstart file to the installer, and install Kickstart on the network.
- Install a virtual machine on your Red Hat Enterprise Linux server with the web console.

Chapter 13, *Run Containers*

Obtain, run, and manage simple lightweight services as containers on a single Red Hat Enterprise Linux server.

- Explain container concepts and the core technologies for building, storing, and running containers.
- Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers.
- Provide persistent storage for container data by sharing storage from the container host, and configure a container network.
- Configure a container as a `systemd` service, and configure a container service to start at boot time.

Lab: Fix Boot Issues and Maintain Servers

Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you troubleshoot and repair boot problems and update the system default target. You also schedule tasks to run on a repeating schedule as a normal user.

Outcomes

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

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

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

```
[student@workstation ~]$ lab start rhcsa-compreview1
```

Specifications

- On `workstation`, run the `/tmp/rhcsa-break1` script. This script causes an issue with the boot process on `serverb` and then reboots the machine. Troubleshoot the cause and repair the boot issue. When prompted, use `redhat` as the password of the `root` user.
- On `workstation`, run the `/tmp/rhcsa-break2` script. This script causes the default target to switch from the `multi-user` target to the `graphical` target on the `serverb` machine and then reboots the machine. On `serverb`, reset the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention. As the `student` user, use the `sudo` command for performing privileged commands. Use `student` as the password, when required.
- On `serverb`, schedule a recurring job as the `student` user that executes the `/home/student/backup-home.sh` script hourly between 7 PM and 9 PM every day except on Saturday and Sunday. Download the backup script from `http://materials.example.com/labs/backup-home.sh`. The `backup-home.sh` script backs up the `/home/student` directory from `serverb` to `servera` in

the `/home/student/serverb-backup` directory. Use the `backup-home.sh` script to schedule the recurring job as the `student` user. Run the command as an executable.

- Reboot the `serverb` machine and wait for the boot to complete before grading.

- On `workstation`, run the `/tmp/rhcsa-break1` script.

```
[student@workstation ~]$ sh /tmp/rhcsa-break1
```

- After the `serverb` machine boots, access the console and notice that the boot process stopped early. Consider a possible cause for this behavior.

1. Locate the icon for the `serverb` console, as appropriate for your classroom environment. Open the console and inspect the error. It might take a few seconds for the error to appear.
2. Press **Ctrl**+**Alt**+**Del** to reboot the `serverb` machine. When the boot-loader menu appears, press any key except **Enter** to interrupt the countdown.
3. Edit the default boot-loader entry, in memory, to log in to the emergency mode. Press **e** to edit the current entry.
4. Use the cursor keys to navigate to the line that starts with `linux`. Append `systemd.unit=emergency.target`.
5. Press **Ctrl**+**x** to boot with the modified configuration.
6. Log in to emergency mode. Use `redhat` as the `root` user's password.

```
7. Give root password for maintenance
8. (or press Control-D to continue): redhat
```

```
[root@serverb ~]#
```

- Remount the `/` file system with read and write capabilities. Use the `mount -a` command to try to mount all the other file systems.

1. Remount the `/` file system with read and write capabilities to edit the file system.

```
[root@serverb ~]# mount -o remount,rw /
```

2. Try to mount all the other file systems. Notice that one of the file systems does not mount.

```
3. [root@serverb ~]# mount -a
4. ...output omitted...
```

```
mount: /FakeMount: can't find UUID=fake.
```

5. Edit the `/etc/fstab` file to fix the issue. Remove or comment out the incorrect line.

```
6. [root@serverb ~]# vi /etc/fstab
7. ...output omitted...
```

```
#UUID=fake     /FakeMount  xfs   defaults    0 0
```

8. Update the `systemd` daemon for the system to register the new `/etc/fstab` file configuration.

```
9. [root@serverb ~]# systemctl daemon-reload
```

```
[ 206.828912] systemd[1]: Reloading.
```

10. Verify that `/etc/fstab` file is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

11. Reboot `serverb` and wait for the boot to complete. The system should now boot without errors.

```
[root@serverb ~]# systemctl reboot
```

- On `workstation`, run the `/tmp/rhcsa-break2` script. Wait for the `serverb` machine to reboot before proceeding.

```
[student@workstation ~]$ sh /tmp/rhcsa-break2
```

- On `serverb`, set the `multi-user` target as the current and default target.

1. Log in to `serverb` as the `student` user.

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

```
[student@serverb ~]$
```

4. Determine the default target.

```
5.  [student@serverb ~]$ systemctl get-default
```

```
graphical.target
```

6. Switch to the `multi-user` target.

```
7.  [student@serverb ~]$ sudo systemctl isolate multi-user.target
```

```
[sudo] password for student: student
```

8. Set the `multi-user` target as the default target.

```
9.  [student@serverb ~]$ sudo systemctl set-default multi-user.target
10. Removed /etc/systemd/system/default.target.
```

```
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
multi-user.target.
```

11. Reboot `serverb` and verify that the `multi-user` target is set as the default target.

```
12. [student@serverb ~]$ sudo systemctl reboot
13. Connection to serverb closed by remote host.
14. Connection to serverb closed.
```

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

15. After the system reboots, open an SSH session to `serverb` as the `student` user. Verify that the `multi-user` target is set as the default target.

```
16. [student@workstation ~]$ ssh student@serverb
17. ...output omitted...
18. [student@serverb ~]$ systemctl get-default
```

```
multi-user.target
```

- On `serverb`, schedule a recurring job as the `student` user that executes the `/home/student/backup-home.sh` script hourly between 7 PM and 9 PM on all days except Saturday and Sunday. Use the `backup-home.sh` script to schedule the recurring job. Download the backup script from `http://materials.example.com/labs/backup-home.sh`. Run the command as an executable.

1. On `serverb`, download the backup script from `http://materials.example.com/labs/backup-home.sh`. Use `chmod` to make the backup script executable.

```
2. [student@serverb ~]$ wget http://materials.example.com/labs/backup-home.sh
3. ...output omitted...
```

```
[student@serverb ~]$ chmod +x backup-home.sh
```

4. Open the crontab file with the default text editor.

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

5. Edit the file to add the following line:

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

Save the changes and exit the editor.

6. Use the `crontab -l` command to list the scheduled recurring jobs.

```
7. [student@serverb ~]$ crontab -l
```

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

- Reboot `serverb` and wait for the boot to complete before grading.

```
[student@serverb ~]$ sudo systemctl reboot
[sudo] password for student: student
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

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

```
[student@workstation ~]$ lab grade rhcsa-compreview1
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rhcsa-compreview1
```

This concludes the section.

Lab: Configure and Manage File Systems and Storage

Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you create a logical volume, mount a network file system, and create a swap partition that is automatically activated at boot. You also configure directories to store temporary files.

Outcomes

- Create a logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure a directory to store temporary files.

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

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

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

```
[student@workstation ~]$ lab start rhcsa-compreview2
```

Specifications

- On `serverb`, configure a new 1 GiB `vol_home` logical volume in a new 2 GiB `extra_storage` volume group. Use the unpartitioned `/dev/vdb` disk to create the partition.
- Format the `vol_home` logical volume with the `XFS` file-system type, and persistently mount it on the `/user-homes` directory.
- On `serverb`, persistently mount the `/share` network file system that `servera` exports on the `/local-share` directory. The `servera` machine exports the `servera.lab.example.com:/share` path.
- On `serverb`, create a 512 MiB swap partition on the `/dev/vdc` disk. Persistently mount the swap partition.
- Create the `production` user group. Create the `production1`, `production2`, `production3`, and `production4` users with the `production` group as their supplementary group.
- On `serverb`, configure the `/run/volatile` directory to store temporary files. If the files in this directory are not accessed for more than 30 seconds, then the system automatically deletes them. Set `0700` as the octal permissions for the directory. Use the `/etc/tmpfiles.d/volatile.conf` file to configure the time-based deletion of the files in the `/run/volatile` directory.

- On `serverb`, configure a new 1 GiB `vol_home` logical volume in a new 2 GiB `extra_storage` volume group. Use the unpartitioned `/dev/vdb` disk to create the partition.

1. Log in to `serverb` as the `student` user and switch to the `root` user.

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

```
4. [student@serverb ~]$ sudo -i
5. [sudo] password for student: student
```

```
[root@serverb ~]#
```

6. Create a 2 GiB partition on the `/dev/vdb` disk.

```
7. [root@serverb ~]# parted /dev/vdb mklabel msdos
8. ...output omitted...
9. [root@serverb ~]# parted /dev/vdb mkpart primary 1MiB 2GiB
10....output omitted...
11.[root@serverb ~]# parted /dev/vdb set 1 lvm on
```

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

12. Declare the `/dev/vdb1` block device as a physical volume.

```
13.[root@serverb ~]# pvcreate /dev/vdb1
```

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

14. Create the `extra_storage` volume group with the `/dev/vdb1` partition.

```
15.[root@serverb ~]# vgcreate extra_storage /dev/vdb1
```

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

16. Create the 1 GiB `vol_home` logical volume.

```
17.[root@serverb ~]# lvcreate -L 1GiB -n vol_home extra_storage
```

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

- Format the `vol_home` logical volume with the XFS file-system type, and persistently mount it on the `/user-homes` directory.

1. Create the `/user-homes` directory.

```
[root@serverb ~]# mkdir /user-homes
```

2. Format the `/dev/extra_storage/vol_home` partition with the XFS file-system type.

```
3. [root@serverb ~]# mkfs -t xfs /dev/extra_storage/vol_home
```

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

4. Persistently mount the `/dev/extra_storage/vol_home` partition on the `/user-homes` directory. Use the partition's UUID for the `/etc/fstab` file entry.

```
5. [root@serverb ~]# lsblk -o UUID /dev/extra_storage/vol_home
6. UUID
7. 988cf149-0667-4733-abca-f80c6ec50ab6
8. [root@serverb ~]# echo "UUID=988c...0ab6 /user-homes xfs defaults 0 0" \
9. >> /etc/fstab
```

```
[root@serverb ~]# mount /user-homes
```

- On `serverb`, persistently mount the `/share` network file system that `servera` exports on the `/local-share` directory. The `servera` machine exports the `servera.lab.example.com:/share` path.

1. Create the `/local-share` directory.

```
[root@serverb ~]# mkdir /local-share
```

2. Append the appropriate entry to the `/etc/fstab` file to persistently mount the `servera.lab.example.com:/share` network file system.

```
3. [root@serverb ~]# echo "servera.lab.example.com:/share /local-share \
```

```
nfs rw,sync 0 0" >> /etc/fstab
```

4. Mount the network file system on the `/local-share` directory.

```
[root@serverb ~]# mount /local-share
```

- On `serverb`, create a 512 MiB swap partition on the `/dev/vdc` disk. Activate and persistently mount the swap partition.

1. Create a 512 MiB partition on the `/dev/vdc` disk.

```
2. [root@serverb ~]# parted /dev/vdc mklabel msdos
```

```
3. ...output omitted...
4. [root@serverb ~]# parted /dev/vdc mkpart primary linux-swap 1MiB 513MiB
```

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

5. Create the swap space on the `/dev/vdc1` partition.

```
6. [root@serverb ~]# mkswap /dev/vdc1
```

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

7. Create an entry in the `/etc/fstab` file to persistently mount the swap space. Use the partition's UUID to create the `/etc/fstab` file entry. Activate the swap space.

```
8. [root@serverb ~]# lsblk -o UUID /dev/vdc1
9. UUID
10. cc18ccb6-bd29-48a5-8554-546bf3471b69
11. [root@serverb ~]# echo "UUID=cc18...1b69 swap swap defaults 0 0" >> /etc/fstab
```

```
[root@serverb ~]# swapon -a
```

- Create the `production` user group. Then, create the `production1`, `production2`, `production3`, and `production4` users with the `production` group as their supplementary group.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i; done
```

- On `serverb`, configure the `/run/volatile` directory to store temporary files. If the files in this directory are not accessed for more than 30 seconds, then the system automatically deletes them. Set `0700` as the octal permissions for the directory. Use the `/etc/tmpfiles.d/volatile.conf` file to configure the time-based deletion of the files in the `/run/volatile` directory.

1. Create the `/etc/tmpfiles.d/volatile.conf` file with the following content:

```
d /run/volatile 0700 root root 30s
```

2. Use the `systemd-tmpfiles --create` command to create the `/run/volatile` directory if it does not exist.

```
[root@serverb ~]# systemd-tmpfiles --create /etc/tmpfiles.d/volatile.conf
```

3. Return to the `workstation` machine as the `student` user.

```
4. [root@serverb ~]# exit
5. logout
6. [student@serverb ~]$ exit
7. logout
```

```
Connection to serverb closed.
```

Evaluation

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

```
[student@workstation ~]$ lab grade rhcsa-compreview2
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rhcsa-compreview2
```

This concludes the section.

Previous Next

Lab: Configure and Manage Server Security

Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

Outcomes

- Configure SSH key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

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

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

```
[student@workstation ~]$ lab start rhcsa-compreview3
```

Specifications

- On `serverb`, generate an SSH key pair for the `student` user. Do not protect the private key with a passphrase.
- Configure the `student` user on `servera` to accept login authentication with the SSH key pair that you generated on the `serverb` machine. The `student` user on `serverb` must be able to log in to `servera` via SSH without entering a password.
- On `servera`, check the `/user-homes/production5` directory permissions. Then, configure SELinux to run in the `permissive` mode by default.
- On `serverb`, verify that the `/localhome` directory does not exist. Then, configure the `production5` user's home directory to mount the `/user-homes/production5` network file system. The `servera.lab.example.com` machine exports the file system as the `servera.lab.example.com:/user-homes/production5` NFS share. Use the `autofs` service to mount the network

    share. Verify that the `autofs` service creates the `/localhome/production5` directory with the same permissions as on `servera`.
  - On `serverb`, adjust the appropriate SELinux Boolean so that the `production5` user may use the NFS-mounted home directory after authenticating with an SSH key. If required, use `redhat` as the password of the `production5` user.
  - On `serverb`, adjust the firewall settings to reject all connection requests from the `servera` machine. Use the `servera` IPv4 address (`172.25.250.10`) to configure the firewall rule.
  - On `serverb`, investigate and fix the issue with the failing Apache web service, which listens on port `30080/TCP` for connections. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

- On `serverb`, generate an SSH key pair for the `student` user. Do not protect the private key with a passphrase.

  1. Log in to `serverb` as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
```

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

  3. Use the `ssh-keygen` command to generate an SSH key pair. Do not protect the private key with a passphrase.

```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+ijpGqjEQSGBR8ORNchiRTHw/URQksVdHjsHqVBXeYI student@serverb.lab.example
.com
The key's randomart image is:
+---[RSA 3072]----+
|+BBX+o*+o..=+..  |
|+.O.oooo .oE+o . |
|.+ . . .. .+ .o  |
|.     o  .  o    |
| .     .S        |
|...    .         |
|.o.  ..          |
|o  .o  o         |
|. .o... .        |
```

```
+----[SHA256]-----+
```

- Configure the `student` user on `servera` to accept login authentication with the SSH key pair that you generated on the `serverb` machine. The `student` user on `serverb` must be able to log in to `servera` via SSH without entering a password.

  1. Send the public key of the newly generated SSH key pair to the `student` user on the `servera` machine.

```
[student@serverb ~]$ ssh-copy-id student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:shYfoFG0Nnv42pv7j+HG+FISmCAm4Bh5jfjwwSMJbrw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
student@servera's password: student

Number of key(s) added: 1

Now try logging in to the machine, with:   "ssh 'student@servera'"
```

```
and check to make sure that only the key(s) you wanted were added.
```

15. Verify that the `student` user can log in to `servera` from `serverb` without entering a password. Do not close the connection.

```
16. [student@serverb ~]$ ssh student@servera
17. ...output omitted...
```

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

- On `servera`, verify the `/user-homes/production5` directory permissions. Then, configure SELinux to run in the `permissive` mode by default.

1. Verify the `/user-homes/production5` directory permissions.

```
2. [student@servera ~]$ ls -ld /user-homes/production5
```

```
drwx------. 2 production5 production5 62 May  6 05:27 /user-homes/production5
```

3. Edit the `/etc/sysconfig/selinux` file to set the `SELINUX` parameter to the `permissive` value.

```
4. [student@servera ~]$ sudo vi /etc/sysconfig/selinux
5. ...output omitted...
6. #SELINUX=enforcing
7. SELINUX=permissive
```

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

8. Reboot the system.

```
9. [student@servera ~]$ sudo systemctl reboot
10. Connection to servera closed by remote host.
11. Connection to servera closed.
```

```
[student@serverb ~]$
```

- On `serverb`, verify that the `/localhome` directory does not exist. Then, configure the `production5` user's home directory to mount the `/user-homes/production5` network file system. The `servera.lab.example.com` machine exports the file system as the `servera.lab.example.com:/user-homes/production5` NFS share. Use the `autofs` service to mount the network share. Verify that the `autofs` service creates the `/localhome/production5` directory with the same permissions as on `servera`.

1. Verify that the `/localhome` directory does not exist.

```
[student@serverb ~]$ ls -ld /localhome
```

```
ls: cannot access '/localhome': No such file or directory
```

3. On `serverb`, switch to the `root` user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
```

```
[root@serverb ~]#
```

6. Install the `autofs` package.

```
[root@serverb ~]# dnf install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  autofs-1:5.1.7-27.el9.x86_64     libsss_autofs-2.6.2-2.el9.x86_64

```

```
Complete!
```

14. Create the `/etc/auto.master.d/production5.autofs` map file with the following content:

```
/- /etc/auto.production5
```

15. Determine the `production5` user's home directory.

```
[root@serverb ~]# getent passwd production5
```

```
production5:x:5001:5001::/localhome/production5:/bin/bash
```

17. Create the `/etc/auto.production5` file with the following content:

```
/localhome/production5 -rw servera.lab.example.com:/user-homes/production5
```

18. Restart the `autofs` service.

```
[root@serverb ~]# systemctl restart autofs
```

19. Verify that the `autofs` service creates the `/localhome/production5` directory on `serverb` with the same permissions as the `/user-homes/production5` directory on `servera`.

```
20. [root@serverb ~]# ls -ld /localhome/production5
```

```
drwx------. 2 production5 production5 62 May  6 05:52 /localhome/production5
```

- On `serverb`, adjust the appropriate SELinux Boolean so that the `production5` user may use the NFS-mounted home directory after authenticating with an SSH key. If required, use `redhat` as the password of the `production5` user.

1. Open a new terminal window and verify from `servera` that the `production5` user cannot log in to `serverb` with SSH key-based authentication. An SELinux Boolean is preventing the user from logging in. From `workstation`, open a new terminal and log in to `servera` as the `student` user.

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

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

4. Switch to the `production5` user. When prompted, use `redhat` as the password of the `production5` user.

```
5. [student@servera ~]$ su - production5
6. Password: redhat
```

```
[production5@servera ~]$
```

7. Generate an SSH key pair.

```
8. [production5@servera ~]$ ssh-keygen
9. Generating public/private rsa key pair.
10. Enter file in which to save the key (/home/production5/.ssh/id_rsa): Enter
```

```
Created directory '/home/production5/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production5/.ssh/id_rsa.
Your public key has been saved in /home/production5/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AbUcIBXneyiGIhr4wS1xzs3WqDvbTP+eZuSRn9HQ/cw production5@servera.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|    ..=++        |
|     . = o       |
|  . .   =    . . |
|.. * + o +  . . .|
|+ = = B S .. o o.|
|.+ + + . .+ . . E|
|. . . .  o o o   |
|    .= .  +.o    |
|    ooo .=+      |
```

```
+----[SHA256]-----+
```

29. Transfer the public key of the SSH key pair to the `production5` user on the `serverb` machine. When prompted, use `redhat` as the password of the `production5` user.

```
[production5@servera ~]$ ssh-copy-id production5@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/production5/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
production5@serverb's password: redhat
```

```

Number of key(s) added: 1

Now try logging in to the machine, with:   "ssh 'production5@serverb'"
```

```
and check to make sure that only the key(s) you wanted were added.
```

42. Use SSH public key-based authentication instead of password-based authentication to log in to `serverb` as the `production5` user. This command should fail.

```
[production5@servera ~]$ **ssh -o pubkeyauthentication=yes \**
**-o passwordauthentication=no production5@serverb**
```

```
production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic
,password).
```

45. On the terminal that is connected to `serverb` as the `root` user, set the `use_nfs_home_dirs` SELinux Boolean to `true`.

```
[root@serverb ~]# **setsebool -P use_nfs_home_dirs true**
```

46. Return to the terminal that is connected to `servera` as the `production5` user, and use SSH public key-based authentication instead of password-based authentication to log in to `serverb` as the `production5` user. This command should succeed.

```
[production5@servera ~]$ **ssh -o pubkeyauthentication=yes \**
**-o passwordauthentication=no production5@serverb**
...*output omitted*...
```

```
[production5@serverb ~]$
```

50. Exit and close the terminal that is connected to `serverb` as the `production5` user. Keep open the terminal that is connected to `serverb` as the `root` user.

- On `serverb`, adjust the firewall settings to reject all connection requests that originate from the `servera` machine. Use the `servera` IPv4 address (`172.25.250.10`) to configure the firewall rule.

1. Add the IPv4 address of `servera` to the `block` zone.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent
```

```
success
```

4. Reload the changes in the firewall settings.

```
[root@serverb ~]# firewall-cmd --reload
```

```
success
```

- On `serverb`, investigate and fix the issue with the failing Apache web service, which listens on port `30080/TCP` for connections. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

1. Restart the `httpd` service. This command fails to restart the service.

```
[root@serverb ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with error code.
```

```
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for details.
```

4. Investigate why the `httpd` service is failing. A permission error indicates that the `httpd` daemon failed to bind to port `30080/TCP` on startup. SELinux policies can prevent an application from binding to a non-standard port. Press **q** to quit the command.

```
[root@serverb ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
     Active: failed (Result: exit-code) since Mon 2022-05-02 13:20:46 EDT; 29s ago
       Docs: man:httpd.service(8)
    Process: 2322 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
   Main PID: 2322 (code=exited, status=1/FAILURE)
     Status: "Reading configuration..."
```

```
        CPU: 30ms

May 02 13:20:46 serverb.lab.example.com systemd[1]: Starting The Apache HTTP Server...
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied: AH00072: make_sock: could not bind to address [::]:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied: AH00072: make_sock: could not bind to address 0.0.0.0:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: no listening sockets available, shutting down
```

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

19. Determine whether an SELinux policy is preventing the `httpd` service from binding to the `30080/TCP` port. The log messages reveal that the `30080/TCP` port does not have the appropriate `http_port_t` SELinux context, and so SELinux prevents the `httpd` service from binding to the port. The log message also produces the syntax of the `semanage port` command, so that you can fix the issue.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
...output omitted...
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port 30080.

*****  Plugin bind_ports (92.2 confidence) suggests   ************************

If you want to allow /usr/sbin/httpd to bind to network port 30080
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 30080
    where PORT_TYPE is one of the following: http_cache_port_t, http_port_t, jboss_management_port_t, jboss_messaging_port_t, ntop_port_t, puppet_port_t.
```

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

31. Set the appropriate SELinux context on the `30080/TCP` port for the `httpd` service to bind to it.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
```

32. Restart the `httpd` service. This command should successfully restart the service.

```
[root@serverb ~]# systemctl restart httpd
```

33. Add the `30080/TCP` port to the default `public` zone.

```
34. [root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
35. success
36. [root@serverb ~]# firewall-cmd --reload
```

```
success
```

37. Return to the `workstation` machine as the `student` user.

```
38. [root@serverb ~]# exit
39. logout
40. [student@serverb ~]$ exit
41. logout
```

```
Connection to serverb closed.
```

Evaluation

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

```
[student@workstation ~]$ lab grade rhcsa-compreview3
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rhcsa-compreview3
```

This concludes the section.

Lab: Run Containers

Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

Outcomes

- Create rootless detached containers.
- Configure port mapping and persistent storage.
- Configure `systemd` for a container to manage it with `systemctl` commands.

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

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

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

```
[student@workstation ~]$ lab start rhcsa-compreview4
```

Specifications

- On `serverb`, configure the `podmgr` user with `redhat` as the password, and set up the appropriate tools for the `podmgr` user to manage the containers for this comprehensive review. Configure the `registry.lab.example.com` as the remote registry. Use `admin` as the user and `redhat321` as the password to authenticate. You can use the `/tmp/review4/registries.conf` file to configure the registry.
- The `/tmp/review4/container-dev` directory contains two directories with development files for the containers in this comprehensive review. Copy the two directories under the `/tmp/review4/container-dev` directory to the `podmgr` home directory. Configure

the `/home/podmgr/storage/database` subdirectory so that you can use it as persistent storage for a container.

- Create the `production` DNS-enabled container network. Use the `10.81.0.0/16` subnet and `10.81.0.1` as the gateway. Use this container network for the containers that you create in this comprehensive review.
- Create the `db-app01` detached container based on the `registry.lab.example.com/rhel8/mariadb-103` container image with the lowest tag number in the `production` network. Use the `/home/podmgr/storage/database` directory as persistent storage for the `/var/lib/mysql/data` directory of the `db-app01` container. Map the 13306 port on the local machine to the 3306 port in the container. Use the values of the following table to set the environment variables to create the containerized database.

Variable	**Value**
`MYSQL_USER`	`developer`
`MYSQL_PASSWORD`	`redhat`
`MYSQL_DATABASE`	`inventory`
`MYSQL_ROOT_PASSWORD`	`redhat`

- Create a `systemd` service file to manage the `db-app01` container. Configure the `systemd` service so that when you start the service, the `systemd` daemon keeps the original container. Start and enable the container as a `systemd` service. Configure the `db-app01` container to start at system boot.
- Copy the `/home/podmgr/db-dev/inventory.sql` script into the `/tmp` directory of the `db-app01` container, and execute the script inside the container. If you executed the script locally, then you would use the `mysql -u root inventory < /tmp/inventory.sql` command.
- Use the container file in the `/home/podmgr/http-dev` directory to create the `http-app01` detached container in the `production` network. The container image name must be `http-client` with the `9.0` tag. Map the 8080 port on the local machine to the 8080 port in the container.
- Use the `curl` command to query the content of the `http-app01` container. Verify that the output of the command shows the container name of the client and that the status of the database is up.

- On `serverb`, configure the `podmgr` user with `redhat` as the password and set up the appropriate tools for the `podmgr` user to manage the containers for this comprehensive review. Configure the `registry.lab.example.com` as the remote registry. Use `admin` as the user and `redhat321` as the password to authenticate. You can use the `/tmp/review4/registries.conf` file to configure the registry.

1. Log in to `serverb` as the `student` user.

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

```
[student@serverb ~]$
```

4. Install the `container-tools` meta-package.

```
[student@serverb ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
Complete!
```

10. Create the `podmgr` user and set `redhat` as the password for the user.

```
[student@serverb ~]$ sudo useradd podmgr
[student@serverb ~]$ sudo passwd podmgr
Changing password for user podmgr.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
```

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

17. Exit the `student` user session. Log in to the `serverb` machine as the `podmgr` user. If prompted, use `redhat` as the password.

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

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

```
[podmgr@serverb ~]$
```

23. Create the `~/.config/containers` directory.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/containers
```

24. Copy the `/tmp/review4/registries.conf` file to the container configuration directory in the home directory.

```
[podmgr@serverb ~]$ cp /tmp/review4/registries.conf ~/.config/containers/
```

25. Log in to the registry to verify the configuration.

```
[podmgr@serverb ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
```

```
Login Succeeded!
```

- The `/tmp/review4/container-dev` directory contains two directories with development files for the containers in this comprehensive review. Copy the two directories in the `/tmp/review4/container-dev` directory to the `podmgr` home directory. Configure the `/home/podmgr/storage/database` subdirectory so that you can use it as persistent storage for a container.

1. Copy the content of the `/tmp/review4/container-dev` directory to the `podmgr` home directory.

```
[podmgr@serverb ~]$ cp -r /tmp/review4/container-dev/* .
[podmgr@serverb ~]$ ls -l
total 0
drwxr-xr-x. 2 podmgr podmgr 27 May 10 21:52 db-dev
```

```
drwxr-xr-x. 2 podmgr podmgr 44 May 10 21:52 http-dev
```

6. Create the `/home/podmgr/storage/database` directory in the `podmgr` home directory. Set the appropriate permissions on the directory for the container to mount it as persistent storage.

```
[podmgr@serverb ~]$ mkdir -p storage/database
[podmgr@serverb ~]$ chmod 0777 storage/database
[podmgr@serverb ~]$ ls -l storage/
total 0
```

```
drwxrwxrwx. 2 podmgr podmgr 6 May 10 21:55 database
```

- Create the `production` DNS-enabled container network. Use the `10.81.0.0/16` subnet and `10.81.0.1` as the gateway. Use this container network for the containers that you create in this comprehensive review.

1. Create the `production` DNS-enabled container network. Use the `10.81.0.0/16` subnet and `10.81.0.1` as the gateway.

```
[podmgr@serverb ~]$ podman network create --gateway 10.81.0.1 \
--subnet 10.81.0.0/16 production
```

```
production
```

4. Verify that the DNS feature is enabled in the `production` network.

```
[podmgr@serverb ~]$ podman network inspect production
[
     {
          "name": "production",
...output omitted...
          "subnets": [
               {
                    "subnet": "10.81.0.0/16",
                    "gateway": "10.81.0.1"
               }
          ],
...output omitted...
          "dns_enabled": true,
...output omitted...
```

- Create the `db-app01` detached container based on the `registry.lab.example.com/rhel8/mariadb-103` container image with the lowest tag number in the `production` network. Use the `/home/podmgr/storage/database` directory as persistent storage for the `/var/lib/mysql/data` directory of the `db-app01` container. Map the 13306 port on the local machine to the 3306 port in the container. Use the values of the following table to set the environment variables to create the containerized database.

Variable	**Value**
`MYSQL_USER`	`developer`
`MYSQL_PASSWORD`	`redhat`
`MYSQL_DATABASE`	`inventory`
`MYSQL_ROOT_PASSWORD`	`redhat`

1. Search for the earliest version tag number of the `registry.lab.example.com/rhel8/mariadb` container image.

```
[podmgr@serverb ~]$ skopeo inspect \
docker://registry.lab.example.com/rhel8/mariadb-103
{
    "Name": "registry.lab.example.com/rhel8/mariadb-103",
    "Digest": "sha256:a95b678e52bb9f4305cb696e45c91a38c19a7c2c5c360ba6c681b10717394816",
    "RepoTags": [
        "1-86",
        "1-102",
        "latest"
```

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

11. Use the earliest version tag number from the output of the previous step to create the detached `db-app01` container in the `production` network. Use the `/home/podmgr/storage/database` directory as persistent storage for the

container. Map the 13306 port to the 3306 container port. Use the data in the table to set the environment variables for the container.

```
[podmgr@serverb ~]$ podman run -d --name db-app01 \
-e MYSQL_USER=developer \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
--network production -p 13306:3306 \
-v /home/podmgr/storage/database:/var/lib/mysql/data:Z \
registry.lab.example.com/rhel8/mariadb-103:1-86
...output omitted...
ba398d080e00ba1d52b1cf4f5959c477681cce343c11cc7fc39e4ce5f1cf2384
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                            COMMAND     CRE
ATED          STATUS             PORTS                        NAMES
```

```
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  20
seconds ago  Up 20 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
```

- Create a `systemd` service file to manage the `db-app01` container. Configure the `systemd` service so that when you start the service, the `systemd` daemon keeps the original container. Start and enable the container as a `systemd` service. Configure the `db-app01` container to start at system boot.

1. Create the `~/.config/systemd/user/` directory for the container unit file.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/systemd/user/
```

2. Create the `systemd` unit file for the `db-app01` container, and move the unit file to the `~/.config/systemd/user/` directory.

```
[podmgr@serverb ~]$ podman generate systemd --name db-app01 --files
/home/podmgr/container-db-app01.service
```

```
[podmgr@serverb ~]$ mv container-db-app01.service ~/.config/systemd/user/
```

5. Stop the `db-app01` container.

```
[podmgr@serverb ~]$ podman stop db-app01
```

```
db-app01
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                               COMMAND     CRE
ATED            STATUS                        PORTS                     NAMES
```

```
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  Abo
ut an hour ago  Exited (0) 3 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
```

10. Reload the user `systemd` service to use the new service unit.

```
[podmgr@serverb ~]$ systemctl --user daemon-reload
```

11. Start and enable the `systemd` unit for the `db-app01` container.

```
[podmgr@serverb ~]$ systemctl --user enable --now container-db-app01
Created symlink /home/podmgr/.config/systemd/user/default.target.wants/contain
er-db-app01.service → /home/podmgr/.config/systemd/user/container-db-app01.ser
vice.
[podmgr@serverb ~]$ systemctl --user status container-db-app01
● container-db-app01.service - Podman container-db-app01.service
     Loaded: loaded (/home/podmgr/.config/systemd/user/container-db-app01.serv
ice; disabled; vendor preset: disabled)
     Active: active (running) since Tue 2022-05-10 22:16:23 EDT; 7s ago
...output omitted...
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                               COMMAND     CRE
ATED         STATUS                    PORTS                        NAMES
```

```
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  59
seconds ago  Up About a minute ago  0.0.0.0:13306->3306/tcp  db-app01
```

21. Use the `loginctl` command to configure the `db-app01` container to start at system boot.

```
[podmgr@serverb ~]$ loginctl enable-linger
```

- Copy the `/home/podmgr/db-dev/inventory.sql` script into the `/tmp` directory of the `db-app01` container, and execute the script inside the container. If you executed the script locally, then you would use the `mysql -u root inventory < /tmp/inventory.sql` command.

1. Copy the `/home/podmgr/db-dev/inventory.sql` script into the `/tmp` directory of the `db-app01` container.

```
2. [podmgr@serverb ~]$ podman cp /home/podmgr/db-dev/inventory.sql \
```

```
db-app01:/tmp/inventory.sql
```

3. Execute the `inventory.sql` script in the `db-app01` container.

```
[podmgr@serverb ~]$ podman exec -it db-app01 sh -c 'mysql -u root inventory <
/tmp/inventory.sql'
```

- Use the container file in the `/home/podmgr/http-dev` directory to create the `http-app01` detached container in the `production` network. The container image name must be `http-client` with the `9.0` tag. Map the 8080 port on the local machine to the 8080 port in the container.

1. Create the `http-client:9.0` image with the container file in the `/home/podmgr/http-dev` directory.

```
2. [podmgr@serverb ~]$ podman build -t http-client:9.0 http-dev/
3. STEP 1/7: FROM registry.lab.example.com/rhel8/php-74:1-63
```

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

4. Create the `http-app01` detached container in the `production` network. Map the 8080 port from the local machine to the 8080 port in the container.

```
5. [podmgr@serverb ~]$ podman run -d --name http-app01 \
6. --network production -p 8080:8080 localhost/http-client:9.0
7. [podmgr@serverb ~]$ podman ps -a
8. CONTAINER ID  IMAGE                                       COMMAND     CRE
   ATED         STATUS             PORTS                     NAMES
9. ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  20
   minutes ago  Up 20 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
```

```
ee424df19621  localhost/http-client:9.0                        /bin/sh -c   4
seconds ago  Up 4 seconds ago   0.0.0.0:8080->8080/tcp   http-app01
```

- Query the content of the `http-app01` container. Verify that it shows the container name of the client and that the status of the database is up.

1. Verify that the `http-app01` container responds to http requests.

```
2. [podmgr@serverb ~]$ curl 127.0.0.1:8080
```

```
This is the server http-app01 and the database is up
```

- Return to the `workstation` machine as the `student` user.

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

Evaluation

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

```
[student@workstation ~]$ lab grade rhcsa-compreview4
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rhcsa-compreview4
```

This concludes the section.

Previous