

# Red Hat RHCSA/RHCE Certification Preparation Study Guide



**redhat.**

## RedHat RHCSA / RHCE 7

RHCSA (EX200) and RHCE (EX300) exams

- \* EX200 - Red Hat Certified System Administrator (RHCSA)
- \* EX300 - Red Hat Certified Engineer (RHCE)

**RHCSA (Red Hat Certified System Administrator)** is a certification exam from Red Hat company, which provides an open source operating system and software to the enterprise community. It also provides support, training and consulting services for the organizations.

## **RHCSA: Red Hat Certified System Administrator**

### **RED HAT ENTERPRISE LINUX 7**



**A 15-article series brought to you by Tecmint.com**

#### RHCSA Exam Preparation Guide

**RHCSA** exam is the certification obtained from Red Hat Inc, after passing the exam (codename **EX200**). RHCSA exam is an upgrade to the **RHCT (Red Hat Certified Technician)** exam, and this upgrade is compulsory as the Red Hat Enterprise Linux was upgraded. The main variation between RHCT and RHCSA is that RHCT exam based on RHEL 5, whereas RHCSA certification is based on RHEL 6 and 7, the courseware of these two certifications are also vary to a certain level.

This Red Hat Certified System Administrator (RHCSA) is essential to perform the following core system administration tasks needed in Red Hat Enterprise Linux environments:

1. Understand and use necessary tools for handling files, directories, command-environments line, and system-wide / packages documentation.
2. Operate running systems, even in different run levels, identify and control processes, start and stop virtual machines.
3. Set up local storage using partitions and logical volumes.
4. Create and configure local and network file systems and its attributes (permissions, encryption, and ACLs).
5. Setup, configure, and control systems, including installing, updating and removing software.
6. Manage system users and groups, along with use of a centralized LDAP directory for authentication.
7. Ensure system security, including basic firewall and SELinux configuration.

To view fees and register for an exam in your country, check the [RHCSA Certification](#) page.

In this 15-article RHCSA series, titled Preparation for the RHCSA (Red Hat Certified System Administrator) exam, we will go to cover the following topics on the latest releases of **Red Hat Enterprise Linux 7**.

**Part 1: Reviewing Essential Commands & System Documentation**

**Part 2:** [How to Perform File and Directory Management in RHEL 7](#)

**Part 3:** [How to Manage Users and Groups in RHEL 7](#)

**Part 4:** [Editing Text Files with Nano and Vim / Analyzing text with grep and regexps](#)

**Part 5:** [Process Management in RHEL 7: boot, shutdown, and everything in between](#)

**Part 6:** [Using ‘Parted’ and ‘SSM’ to Configure and Encrypt System Storage](#)

**Part 7:** [Using ACLs \(Access Control Lists\) and Mounting Samba / NFS Shares](#)

**Part 8:** [Securing SSH, Setting Hostname and Enabling Network Services](#)

**Part 9:** [Installing, Configuring and Securing a Web and FTP Server](#)

**Part 10:** [Yum Package Management, Automating Tasks with Cron and Monitoring System Logs](#)

**Part 11:** [Firewall Essentials and Control Network Traffic Using FirewallD and Iptables](#)

**Part 12:** [Automate RHEL 7 Installations Using ‘Kickstart’](#)

**Part 13:** [Mandatory Access Control Essentials with SELinux](#)

**Part 14:** [Setting Up LDAP-based Authentication in RHEL 7](#)

**Part 15:** [Essentials of Virtualization and Guest Administration with KVM](#)

In this **Part 1** of the RHCSA series, we will explain how to enter and execute commands with the correct syntax in a shell prompt or terminal, and explained how to find, inspect, and use system documentation.

## RHCSA: Red Hat Certified System Administrator

### RED HAT ENTERPRISE LINUX 7



#### Part 1: Reviewing essential commands & system documentation

RHCSA: Reviewing Essential Linux Commands – Part 1

##### Prerequisites:

At least a slight degree of familiarity with basic Linux commands such as:

1. [cd command](#) (change directory)
2. [ls command](#) (list directory)

3. [cp command](#) (copy files)
4. [mv command](#) (move or rename files)
5. [touch command](#) (create empty files or update the timestamp of existing ones)
6. rm command (delete files)
7. mkdir command (make directory)

The correct usage of some of them are anyway exemplified in this article, and you can find further information about each of them using the suggested methods in this article.

Though not strictly required to start, as we will be discussing general commands and methods for information search in a Linux system, you should try to install **RHEL 7** as explained in the following article. It will make things easier down the road.

1. [Red Hat Enterprise Linux \(RHEL\) 7 Installation Guide](#)

## Interacting with the Linux Shell

If we log into a Linux box using a **text-mode** login screen, chances are we will be dropped directly into our default shell. On the other hand, if we login using a graphical user interface (GUI), we will have to open a shell manually by starting a terminal. Either way, we will be presented with the user prompt and we can start typing and executing commands (a command is executed by pressing the **Enter** key after we have typed it).

Commands are composed of two parts:

1. the name of the command itself, and
2. arguments

Certain arguments, called **options** (usually preceded by a hyphen), alter the behavior of the command in a particular way while other arguments specify the objects upon which the command operates.

The **type** command can help us identify whether another certain command is built into the shell or if it is provided by a separate package. The need to make this distinction lies in the place where we will find more information about the command. For shell built-ins we need to look in the shell's man page, whereas for other binaries we can refer to its own man page.

```
gacanepa@dev2:~$ type cd
cd is a shell builtin
gacanepa@dev2:~$ type top
top is /usr/bin/top
gacanepa@dev2:~$ type type
type is a shell builtin
gacanepa@dev2:~$ type less
less is /usr/bin/less
gacanepa@dev2:~$ http://www.tecmint.com
```

Check Shell built in Commands

In the examples above, **cd** and **type** are shell built-ins, while **top** and **less** are binaries external to the shell itself (in this case, the location of the command executable is returned by **type**).

## RHCSA Series: Reviewing Essential Commands & System Documentation – Part 1

Other well-known shell built-ins include:

1. [echo command](#): Displays strings of text.
2. [pwd command](#): Prints the current working directory.

```
gacanepa@dev2:~$ echo "This is Part 1 of the RHCSA series brought to you by Tecmint.com"
This is Part 1 of the RHCSA series brought to you by Tecmint.com
gacanepa@dev2:~$ pwd
/home/gacanepa
gacanepa@dev2:~$
```

<http://www.tecmint.com>

### More Built in Shell Commands

#### [exec command](#)

Runs an external program that we specify. Note that in most cases, this is better accomplished by just typing the name of the program we want to run, but the **exec** command has one special feature: rather than create a new process that runs alongside the shell, the new process replaces the shell, as can be verified by subsequent.

```
# ps -ef | grep [original PID of the shell process]
```

When the new process terminates, the shell terminates with it. Run **exec top** and then hit the **q** key to quit top. You will notice that the shell session ends when you do, as shown in the following screencast:

#### [export command](#)

Exports variables to the environment of subsequently executed commands.

#### [history Command](#)

Displays the command history list with line numbers. A command in the history list can be repeated by typing the command number preceded by an exclamation sign. If we need to edit a command in history list before executing it, we can press **Ctrl + r** and start typing the first letters associated with the command. When we see the command completed automatically, we can edit it as per our current need:

This list of commands is kept in our home directory in a file called **.bash\_history**. The history facility is a useful resource for reducing the amount of typing, especially when combined with command line editing. By default, bash stores the last 500 commands you have entered, but this limit can be extended by using the **HISTSIZE** environment variable:

```
gacanepa@dev2:~$ echo $HISTSIZE  
500  
gacanepa@dev2:~$ export HISTSIZE=1000  
gacanepa@dev2:~$ echo $HISTSIZE  
1000  
gacanepa@dev2:~$ █ http://www.tecmint.com
```

## Linux history Command

But this change **as** performed above, will not be persistent on our next boot. In order to preserve the change in the **HISTSIZE** variable, we need to edit the **.bashrc** file by hand:

```
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)  
HISTSIZE=1000
```

**Important:** Keep in mind that these changes will not take effect until we restart our shell session.

## alias command

With no arguments or with the **-p** option prints the list of aliases in the form alias **name=value** on standard output. When arguments are provided, an alias is defined for each name whose value is given.

With **alias**, we can make up our own commands or modify existing ones by including desired options. For example, suppose we want to alias **ls** to **ls --color=auto** so that the output will display regular files, directories, symlinks, and so on, in different colors:

```
# alias ls='ls --color=auto'  
root@dev2:~# ls  
backups calcurse.txt Release.key sent swapkeyfile trace.scap  
root@dev2:~# alias ls='ls --color=auto'  
root@dev2:~# ls  
backups calcurse.txt Release.key sent swapkeyfile trace.scap  
root@dev2:~# alias  
alias ls='ls --color=auto'  
root@dev2:~# █ http://www.tecmint.com
```

## Linux alias Command

**Note:** That you can assign any name to your “**new command**” and enclose as many commands as desired between single quotes, but in that case you need to separate them by semicolons, as follows:

```
# alias myNewCommand='cd /usr/bin; ls; cd; clear'  
exit command
```

The **exit** and **logout** commands both terminate the shell. The **exit** command terminates any shell, but the **logout** command terminates only login shells—that is, those that are launched automatically when you initiate a text-mode login.

If we are ever in doubt as to what a program does, we can refer to its man page, which can be invoked using the **man command**. In addition, there are also man pages for important files (inittab, fstab, hosts, to name a few), library functions, shells, devices, and other features.

#### Examples:

1. **man uname** (print system information, such as kernel name, processor, operating system type, architecture, and so on).
2. **man inittab** (init daemon configuration).

Another important source of information is provided by the **info** command, which is used to read info documents. These documents often provide more information than the man page. It is invoked by using the **info keyword** followed by a command name, such as:

```
# info ls  
# info cut
```

In addition, the **/usr/share/doc** directory contains several subdirectories where further documentation can be found. They either contain plain-text files or other friendly formats.

Make sure you make it a habit to use these three methods to look up information for commands. Pay special and careful attention to the syntax of each of them, which is explained in detail in the documentation.

#### Converting Tabs into Spaces with expand Command

Sometimes text files contain tabs but programs that need to process the files don't cope well with tabs. Or maybe we just want to convert tabs into spaces. That's where the **expand** tool (provided by the GNU coreutils package) comes in handy.

For example, given the file **NumbersList.txt**, let's run **expand** against it, changing tabs to one space, and display on standard output.

```
# expand --tabs=1 NumbersList.txt  
root@dev2:~# cat NumbersList.txt  
One      Two      Three     Four  
Five      Six      Seven     Eight  
Nine      Ten      Eleven    Twelve  
root@dev2:~# expand --tabs=1 NumbersList.txt  
One Two Three Four  
Five Six Seven Eight  
Nine Ten Eleven Twelve  
root@dev2:~#
```

<http://www.tecmint.com>

#### Linux expand Command

The **unexpand** command performs the reverse operation (converts spaces into tabs).

Display the first lines of a file with head and the last lines with tail

By default, the **head** command followed by a **filename**, will display the first 10 lines of the said file. This behavior can be changed using the **-n** option and specifying a certain number of lines.

```
# head -n3 /etc/passwd
# tail -n3 /etc/passwd
root@dev2:~# head -n3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
root@dev2:~# tail -n3 /etc/passwd
btsync:x:1005:1005:,,,:/home/btsync:/bin/bash
varnish:x:116:123::/home/varnish:/bin/false
varnishlog:x:117:124::/home/varnishlog:/bin/false
root@dev2:~# █ http://www.tecmint.com
```

## Linux head and tail Command

One of the most interesting features of **tail** is the possibility of displaying data (last lines) as the input file grows (**tail -f my.log**, where my.log is the file under observation). This is particularly useful when monitoring a log to which data is being continually added.

## Merging Lines with paste

The **paste** command merges files line by line, separating the lines from each file with tabs (by default), or another delimiter that can be specified (in the following example the fields in the output are separated by an equal sign).

```
# paste -d= file1 file2
```

```
root@dev2:~# cat file1
1
2
3
4
5
6
root@dev2:~# cat file2
one
two
three
four
five
six
root@dev2:~# paste -d= file1 file2
1=one
2=two
3=three
4=four
5=five
6=six
root@dev2:~# █ http://www.tecmint.com
```

## Merge Files in Linux

Breaking a file into pieces using split command

The **split** command is used split a file into two (or more) separate files, which are named according to a prefix of our choosing. The splitting can be defined by size, chunks, or number of lines, and the resulting files can have a numeric or alphabetic suffixes. In the following example, we will split **bash.pdf** into files of size **50 KB** (-b 50KB), using numeric suffixes (-d):

```
# split -b 50KB -d bash.pdf bash_
root@dev2:~# ls -lh | grep bash | grep -v grep
-rw-r--r-- 1 root root 290K Feb 21 12:18 bash.pdf
root@dev2:~# split -b 50KB -d bash.pdf bash_
root@dev2:~# ls -lh | grep bash | grep -v grep
-rw-r--r-- 1 root root 49K Feb 21 12:18 bash_00
-rw-r--r-- 1 root root 49K Feb 21 12:18 bash_01
-rw-r--r-- 1 root root 49K Feb 21 12:18 bash_02
-rw-r--r-- 1 root root 49K Feb 21 12:18 bash_03
-rw-r--r-- 1 root root 49K Feb 21 12:18 bash_04
-rw-r--r-- 1 root root 46K Feb 21 12:18 bash_05
-rw-r--r-- 1 root root 290K Feb 21 12:18 bash.pdf
root@dev2:~# █ http://www.tecmint.com
```

## Split Files in Linux

You can merge the files to recreate the original file with the following command:

```
# cat bash_00 bash_01 bash_02 bash_03 bash_04 bash_05 > bash.pdf
```

## Translating characters with tr command

The **tr** command can be used to translate (change) characters on a one-by-one basis or using character ranges. In the following example we will use the same **file2** as previously, and we will change:

1. lowercase o's to uppercase,
2. and all lowercase to uppercase

```
# cat file2 | tr o O
# cat file2 | tr [a-z] [A-Z]
root@dev2:~# cat file2
one
two
three
four
five
six
root@dev2:~# cat file2 | tr o O
One
two
three      Change only lowercase o's
four        to uppercase
five
six
root@dev2:~# cat file2 | tr [a-z] [A-Z]
ONE
TWO      Change all lowercase
THREE    to uppercase
FOUR
FIVE
SIX
root@dev2:~# http://www.tecmint.com
```

## Translate Characters in Linux

### Reporting or deleting duplicate lines with uniq and sort command

The **uniq** command allows us to report or remove duplicate lines in a file, writing to **stdout** by default. We must note that **uniq** does not detect repeated lines unless they are adjacent. Thus, **uniq** is commonly used along with a preceding **sort** (which is used to sort lines of text files).

By default, **sort** takes the first field (separated by spaces) as key field. To specify a different key field, we need to use the **-k** option. Please note how the output returned by sort and uniq change as we change the key field in the following example:

```
# cat file3
# sort file3 | uniq
# sort -k2 file3 | uniq
# sort -k3 file3 | uniq
```

```
root@dev2:~# cat file3
Jane Doe 111
Dave Null 114
Peter Cramp 113
Dave Smith 112
Dave Null 114
Peter Cramp 115
root@dev2:~# sort file3 | uniq
Dave Null 114
Dave Smith 112
Jane Doe 111
Peter Cramp 113
Peter Cramp 115
root@dev2:~# sort -k2 file3 | uniq
Peter Cramp 113
Peter Cramp 115
Jane Doe 111
Dave Null 114
Dave Smith 112
root@dev2:~# sort -k3 file3 | uniq
Jane Doe 111
Dave Smith 112
Peter Cramp 113
Dave Null 114
Peter Cramp 115
root@dev2:~# http://www.tecmint.com
```

## Remove Duplicate Lines in Linux

### Extracting text with cut command

The **cut** command extracts portions of input lines (from **stdin** or files) and displays the result on standard output, based on number of bytes (-b), characters (-c), or fields (-f).

When using **cut** based on fields, the default field separator is a tab, but a different separator can be specified by using the **-d** option.

```
# cut -d: -f1,3 /etc/passwd # Extract specific fields: 1 and 3 in this case
# cut -d: -f2-4 /etc/passwd # Extract range of fields: 2 through 4 in this example
```

```
root@dev2:~# cut -d: -f1,3 /etc/passwd
root:0
daemon:1
bin:2
sys:3
sync:4
games:5
man:6
lp:7
root@dev2:~# cut -d: -f2-4 /etc/passwd
x:0:0
x:1:1
x:2:2
x:3:3
x:4:65534
x:5:60
x:6:12
x:7:7
x:8:8
http://www.tecmint.com
```

## Extract Text From a File in Linux

Note that the output of the two examples above was truncated for brevity.

#### Reformatting files with fmt command

**fmt** is used to “clean up” files with a great amount of content or lines, or with varying degrees of indentation. The new paragraph formatting defaults to no more than 75 characters wide. You can change this with the **-w** (width) option, which set the line length to the specified number of characters.

For example, let’s see what happens when we use **fmt** to display the **/etc/passwd** file setting the width of each line to 100 characters. Once again, output has been truncated for brevity.

```
# fmt -w100 /etc/passwd
root@dev2:~# fmt -w100 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
http://www.tecmint.com
```

#### File Reformatting in Linux

#### Formatting content for printing with pr command

**pr** paginates and displays in columns one or more files for printing. In other words, **pr** formats a file to make it look better when printed. For example, the following command:

```
# ls -a /etc | pr -n --columns=3 -h "Files in /etc"
```

Shows a listing of all the files found in **/etc** in a printer-friendly format (3 columns) with a custom header (indicated by the **-h** option), and numbered lines (**-n**).

```
root@dev2:~# ls -a /etc | pr -n --columns=3 -h "Files in /etc"
http://www.tecmint.com
2015-02-21 12:47          Files in /etc          Page 1

 1  .           57  environment      113  lvm
 2  ..          58  exim4          114  magic
 3  acpi         59  fonts          115  magic.mime
 4  adduser.conf 60  foomatic      116  mailcap
 5  adjtime       61  fstab          117  mailcap.order
 6  aide          62  fstab.d        118  mailname
 7  aliases       63  gai.conf       119  mail.rc
```

#### File Formatting in Linux

## Summary

In this article we have discussed how to enter and execute commands with the correct syntax in a shell prompt or terminal, and explained how to find, inspect, and use system

documentation. As simple as it seems, it's a large first step in your way to becoming a **RHCSA**.

If you would like to add other commands that you use on a periodic basis and that have proven useful to fulfill your daily responsibilities, feel free to share them with the world by using the comment form below. Questions are also welcome. We look forward to hearing from you!

## RHCSA Series: How to Perform File and Directory Management – Part 2

In this article, **RHCSA Part 2: File and directory management**, we will review some essential skills that are required in the day-to-day tasks of a system administrator.

### **RHCSA: Red Hat Certified System Administrator**

#### **RED HAT ENTERPRISE LINUX 7**



#### **Part 2: File and directory management**

### RHCSA: Perform File and Directory Management – Part 2

#### **Create, Delete, Copy, and Move Files and Directories**

File and directory management is a critical competence that every system administrator should possess. This includes the ability to create / delete text files from scratch (the core of each program's configuration) and directories (where you will organize files and other directories), and to find out the type of existing files.

The [touch command](#) can be used not only to create empty files, but also to update the access and modification times of existing files.

```
[root@dodev01 ~]# touch testfile http://www.tecmint.com
[root@dodev01 ~]# ls -l
total 0
-rw-r--r-- 1 root root 0 Mar  9 18:20 testfile
[root@dodev01 ~]# echo "Adding some content..." > testfile
[root@dodev01 ~]# ls -l
total 4
-rw-r--r-- 1 root root 23 Mar  9 18:21 testfile
[root@dodev01 ~]# touch testfile
[root@dodev01 ~]# ls -l
total 4
-rw-r--r-- 1 root root 23 Mar  9 18:22 testfile
[root@dodev01 ~]# 
```

### touch command example

You can use `file [filename]` to determine a file's type (this will come in handy before launching your preferred text editor to edit it).

```
[root@dodev01 ~]# file testfile
testfile: ASCII text
[root@dodev01 ~]# 
```

### file command example

and `rm [filename]` to delete it.

```
[root@dodev01 ~]# ls
testfile
[root@dodev01 ~]# rm testfile
rm: remove regular file ‘testfile’? y
[root@dodev01 ~]# ls
[root@dodev01 ~]# 
```

### rm command example

As for directories, you can create directories inside existing paths with `mkdir [directory]` or create a full path with `mkdir -p [/full/path/to/directory]`.

```
[root@dodev01 ~]# ls
[root@dodev01 ~]# mkdir dir1
[root@dodev01 ~]# ls
dir1
[root@dodev01 ~]# mkdir dir1/dir2/dir3
mkdir: cannot create directory 'dir1/dir2/dir3': No such file or directory
[root@dodev01 ~]# mkdir -p dir1/dir2/dir3
[root@dodev01 ~]# ls -lR
.:
total 4
drwxr-xr-x 3 root root 4096 Mar  9 18:40 dir1

./dir1:
total 4
drwxr-xr-x 3 root root 4096 Mar  9 18:40 dir2

./dir1/dir2:
total 4
drwxr-xr-x 2 root root 4096 Mar  9 18:40 dir3

./dir1/dir2/dir3:
total 0
[root@dodev01 ~]#
```

<http://www.tecmint.com>

## mkdir command example

When it comes to removing directories, you need to make sure that they're empty before issuing the `rmdir [directory]` command, or use the more powerful (handle with care!) `rm -rf [directory]`. This last option will force remove recursively the `[directory]` and all its contents – so use it at your own risk.

## Input and Output Redirection and Pipelining

The command line environment provides two very useful features that allows to redirect the **input** and **output** of commands from and to files, and to send the output of a command to another, called redirection and pipelining, respectively.

To understand those two important concepts, we must first understand the three most important types of I/O (Input and Output) streams (or sequences) of characters, which are in fact special files, in the \*nix sense of the word.

1. **Standard input** (aka `stdin`) is by default attached to the keyboard. In other words, the keyboard is the standard input device to enter commands to the command line.
2. **Standard output** (aka `stdout`) is by default attached to the screen, the device that “receives” the output of commands and display them on the screen.
3. **Standard error** (aka `stderr`), is where the status messages of a command is sent to by default, which is also the screen.

In the following example, the output of `ls /var` is sent to **stdout** (the screen), as well as the result of `ls /tecmint`. But in the latter case, it is **stderr** that is shown.

```
[root@dodev01 ~]# ls /var
bin cache crash db empty ftp games gopher kernels lib local lock log mail
[root@dodev01 ~]# ls /tecmint
ls: cannot access /tecmint: No such file or directory
[root@dodev01 ~]# 
```

<http://www.tecmint.com>

## Input and Output Example

To more easily identify these special files, they are each assigned a file descriptor, an abstract representation that is used to access them. The essential thing to understand is that these files, just like others, can be redirected. What this means is that you can capture the output from a file or script and send it as input to another file, command, or script. This will allow you to store on disk, for example, the output of commands for later processing or analysis.

To redirect stdin (fd 0), stdout (fd 1), or stderr (fd 2), the following operators are available.

Redirection Operator	Effect
>	Redirects standard output to a file containing standard output. If the destination file exists, it will be overwritten.
>>	Appends standard output to a file.
2>	Redirects standard error to a file containing standard output. If the destination file exists, it will be overwritten.
2>>	Appends standard error to the existing file.
&>	Redirects both standard output and standard error to a file; if the specified file exists, it will be overwritten.
<	Uses the specified file as standard input.
<>	The specified file is used for both standard input and standard output.

As opposed to redirection, pipelining is performed by adding a vertical bar (|) after a command and before another one.

Remember:

1. **Redirection** is used to send the output of a command to a file, or to send a file as input to a command.
2. **Pipelining** is used to send the output of a command to another command as input.

## Examples Of Redirection and Pipelining

### Example 1: Redirecting the output of a command to a file

There will be times when you will need to iterate over a list of files. To do that, you can first save that list to a file and then read that file line by line. While it is true that you can iterate over the output of ls directly, this example serves to illustrate redirection.

```
# ls -1 /var/mail > mail.txt
[root@dodev01 ~]# ls -1 /var/mail > mail.txt
[root@dodev01 ~]# cat mail.txt
dave
gacanepa
phil
[root@dodev01 ~]#
```

<http://www.tecmint.com>

### Redirect output of command tot a file

#### Example 2: Redirecting both stdout and stderr to /dev/null

In case we want to prevent both stdout and stderr to be displayed on the screen, we can redirect both file descriptors to /dev/null. Note how the output changes when the redirection is implemented for the same command.

```
# ls /var /tecmint
# ls /var/ /tecmint &> /dev/null
[root@dodev01 ~]# ls /var /tecmint
ls: cannot access /tecmint: No such file or directory
/var:
adm cache crash db empty ftp games gopher kerberos lib loca
[root@dodev01 ~]# ls /var /tecmint &> /dev/null
[root@dodev01 ~]#
```

<http://www.tecmint.com>

### Redirecting stdout and stderr ouput to /dev/null

#### Example 3: Using a file as input to a command

While the classic syntax of the [cat command](#) is as follows.

```
# cat [file(s)]
```

You can also send a file as input, using the correct redirection operator.

```
# cat < mail.txt
```

```
[root@dodev01 ~]# cat < mail.txt
dave
gacanepa
phil
[root@dodev01 ~]#
```

 http://www.tecmint.com

### cat command example

#### Example 4: Sending the output of a command as input to another

If you have a large directory or process listing and want to be able to locate a certain file or process at a glance, you will want to pipeline the listing to grep.

Note that we use two pipelines in the following example. The first one looks for the required keyword, while the second one will eliminate the actual grep command from the results. This example lists all the processes associated with the apache user.

```
# ps -ef | grep apache | grep -v grep
```

```
[root@dodev01 ~]# ps -ef | grep apache | grep -v grep
apache 21004 671 0 Mar01 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 21666 671 0 Mar02 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22332 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22341 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22346 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22348 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22349 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22350 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22351 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache 22352 671 0 Mar03 ? 00:00:00 /usr/sbin/httpd -DFOREGROUND
[root@dodev01 ~]#
```

 http://www.tecmint.com

### Send output of command as input to another

If you need to transport, backup, or send via email a group of files, you will use an archiving (or grouping) tool such as [tar](#), typically used with a compression utility like **gzip**, **bzip2**, or **xz**.

Your choice of a compression tool will be likely defined by the compression speed and rate of each one. Of these three compression tools, **gzip** is the oldest and provides the least compression, **bzip2** provides improved compression, and **xz** is the newest and provides the best compression. Typically, files compressed with these utilities have **.gz**, **.bz2**, or **.xz** extensions, respectively.

Command	Abbreviation	Description
-create	c	Creates a tar archive
-concatenate	A	Appends tar files to an archive

-append	r	Appends non-tar files to an archive
-update	u	Appends files that are newer than those in an archive
-diff or -compare	d	Compares an archive to files on disk
-list	t	Lists the contents of a tarball
-extract or -get	x	Extracts files from an archive
Operation modifier	Abbreviation	Description
-directory dir	c	Changes to directory dir before performing operations
-same-permissions and -same-owner	p	Preserves permissions and ownership information, respectively.
-verbose	v	Lists all files as they are read or extracted; if combined with -list, it also displays file sizes, ownership, and timestamps
-exclude file	-	Excludes file from the archive. In this case, file can be an actual file or a pattern.
-gzip or -gunzip	z	Compresses an archive through gzip
-bzip2	j	Compresses an archive through bzip2
-xz	J	Compresses an archive through xz

Example 5: Creating a tarball and then compressing it using the three compression utilities

You may want to compare the effectiveness of each tool before deciding to use one or another. Note that while compressing small files, or a few files, the results may not show much differences, but may give you a glimpse of what they have to offer.

```
# tar cf ApacheLogs-$(date +%Y%m%d).tar /var/log/httpd/*           # Create an ordinary tarball
# tar czf ApacheLogs-$(date +%Y%m%d).tar.gz /var/log/httpd/*        # Create a tarball and compress with gzip
# tar cjf ApacheLogs-$(date +%Y%m%d).tar.bz2 /var/log/httpd/*       # Create a tarball and compress with bzip2
# tar cJf ApacheLogs-$(date +%Y%m%d).tar.xz /var/log/httpd/*        # Create a tarball and compress with xz
```

```
[root@dodev01 ~]# tar cf ApacheLogs-$(date +%Y%m%d).tar /var/log/httpd/*
tar: Removing leading `/' from member names
[root@dodev01 ~]# tar czf ApacheLogs-$(date +%Y%m%d).tar.gz /var/log/httpd/*
tar: Removing leading `/' from member names
[root@dodev01 ~]# tar cjf ApacheLogs-$(date +%Y%m%d).tar.bz2 /var/log/httpd/*
tar: Removing leading `/' from member names
[root@dodev01 ~]# tar cJf ApacheLogs-$(date +%Y%m%d).tar.xz /var/log/httpd/*
tar: Removing leading `/' from member names
[root@dodev01 ~]# ls -l
total 340
-rw-r--r-- 1 root root 276480 Mar  7 11:06 ApacheLogs-20150307.tar
-rw-r--r-- 1 root root  20114 Mar  7 11:07 ApacheLogs-20150307.tar.bz2
-rw-r--r-- 1 root root  26806 Mar  7 11:06 ApacheLogs-20150307.tar.gz
-rw-r--r-- 1 root root  19004 Mar  7 11:07 ApacheLogs-20150307.tar.xz
[root@dodev01 ~]#
```

<http://www.tecmint.com>

## tar command examples

Example 6: Preserving original permissions and ownership while archiving and when

If you are creating backups from users' home directories, you will want to store the individual files with the original permissions and ownership instead of changing them to that of the user account or daemon performing the backup. The following example preserves these attributes while taking the backup of the contents in the `/var/log/httpd` directory:

```
# tar cJf ApacheLogs-$(date +%Y%m%d).tar.xz /var/log/httpd/* --same-permissions --same-owner
```

## Create Hard and Soft Links

In Linux, there are two types of links to files: **hard links** and **soft** (aka symbolic) links. Since a hard link represents another name for an existing file and is identified by the same **inode**, it then points to the actual data, as opposed to symbolic links, which point to filenames instead.

In addition, hard links do not occupy space on disk, while symbolic links do take a small amount of space to store the text of the link itself. The downside of hard links is that they can only be used to reference files within the filesystem where they are located because inodes are unique inside a filesystem. Symbolic links save the day, in that they point to another file or directory by name rather than by inode, and therefore can cross filesystem boundaries.

The basic syntax to create links is similar in both cases:

```
# ln TARGET LINK_NAME          # Hard link named LINK_NAME to file
named TARGET
# ln -s TARGET LINK_NAME      # Soft link named LINK_NAME to file
named TARGET
```

Example 7: Creating hard and soft links

There is no better way to visualize the relation between a file and a hard or symbolic link that point to it, than to create those links. In the following screenshot you will see that the file and

the hard link that points to it share the same inode and both are identified by the same disk usage of 466 bytes.

On the other hand, creating a hard link results in an extra disk usage of 5 bytes. Not that you're going to run out of storage capacity, but this example is enough to illustrate the difference between a hard link and a soft link.

```
[root@dodev01 ~]# ls -li
total 4
913936 -rw-r--r-- 1 root root 466 Mar  9 16:16 rhcsa
[root@dodev01 ~]# ln rhcsa rhcsa_hardlink
[root@dodev01 ~]# ln -s rhcsa rhcsa_softlink
[root@dodev01 ~]# ls -li
total 8
913936 -rw-r--r-- 2 root root 466 Mar  9 16:16 rhcsa
913936 -rw-r--r-- 2 root root 466 Mar  9 16:16 rhcsa_hardlink
913933 lrwxrwxrwx 1 root root   5 Mar  9 16:16 rhcsa_softlink -> rhcsa
[root@dodev01 ~]#
```

<http://www.tecmint.com>

### Difference between a hard link and a soft link

A typical usage of symbolic links is to reference a versioned file in a Linux system. Suppose there are several programs that need access to file **fooX.Y**, which is subject to frequent version updates (think of a library, for example). Instead of updating every single reference to **fooX.Y** every time there's a version update, it is wiser, safer, and faster, to have programs look to a symbolic link named just **foo**, which in turn points to the actual **fooX.Y**.

Thus, when **X** and **Y** change, you only need to edit the symbolic link **foo** with a new destination name instead of tracking every usage of the destination file and updating it.

## Summary

In this article we have reviewed some essential file and directory management skills that must be a part of every system administrator's tool-set. Make sure to review other parts of this series as well in order to integrate these topics with the content covered in this tutorial.

Feel free to let us know if you have any questions or comments. We are always more than glad to hear from our readers.

## RHCSA Series: How to Manage Users and Groups in RHEL 7 – Part 3

Managing a **RHEL 7** server, as it is the case with any other Linux server, will require that you know how to add, edit, suspend, or delete user accounts, and grant users the necessary permissions to files, directories, and other system resources to perform their assigned tasks.

### **RHCSA: Red Hat Certified System Administrator**

### **RED HAT ENTERPRISE LINUX 7**



### **Part 3: User and group management**

#### RHCSA: User and Group Management – Part 3

#### **Managing User Accounts**

To add a new user account to a RHEL 7 server, you can run either of the following two commands as root:

```
# adduser [new_account]  
# useradd [new_account]
```

When a new user account is added, by default the following operations are performed.

1. His/her home directory is created (`/home/username` unless specified otherwise).
2. These `.bash_logout`, `.bash_profile` and `.bashrc` hidden files are copied inside the user's home directory, and will be used to provide environment variables for his/her user session. You can explore each of them for further details.
3. A mail spool directory is created for the added user account.
4. A group is created with the same name as the new user account.

The full account summary is stored in the `/etc/passwd` file. This file holds a record per system user account and has the following format (fields are separated by a colon):

```
[username]:[x]:[UID]:[GID]:[Comment]:[Home directory]:[Default shell]
```

1. These two fields [username] and [Comment] are self explanatory.
2. The second field 'x' indicates that the account is secured by a shadowed password (in /etc/shadow), which is used to logon as [username].
3. The fields [UID] and [GID] are integers that shows the User IDentification and the primary Group IDentification to which [username] belongs, equally.

Finally,

1. The [Home directory] shows the absolute location of [username]'s home directory, and
2. [Default shell] is the shell that is committed to this user when he/she logs in to the system.

Another important file that you must become familiar with is /etc/group, where group information is stored. As it is the case with /etc/passwd, there is one record per line and its fields are also delimited by a colon:

```
[Group name]:[Group password]:[GID]:[Group members]
```

where,

1. [Group name] is the name of group.
2. Does this group use a group password? (An "x" means no).
3. [GID]: same as in /etc/passwd.
4. [Group members]: a list of users, separated by commas, that are members of each group.

After adding an account, at anytime, you can edit the user's account information using **usermod**, whose basic syntax is:

```
# usermod [options] [username]
```

#### **Read Also:**

- [15 'useradd' Command Examples](#)
- [15 'usermod' Command Examples](#)

**EXAMPLE 1:** Setting the expiry date for an account

If you work for a company that has some kind of policy to enable account for a certain interval of time, or if you want to grant access to a limited period of time, you can use the --expiredate flag followed by a date in **YYYY-MM-DD** format. To verify that the change has been applied, you can compare the output of

```
# chage -l [username]
```

before and after updating the account expiry date, as shown in the following image.

```
[root@rhel7 ~]# chage -l tecmint
Last password change : Mar 14, 2015
Password expires : never
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
[root@rhel7 ~]# usermod --expiredate=2015-03-19 tecmint
[root@rhel7 ~]# chage -l tecmint
Last password change : Mar 14, 2015
Password expires : never
Password inactive : never
Account expires : Mar 19, 2015
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
[root@rhel7 ~]#
```

## Change User Account Information

### EXAMPLE 2: Adding the user to supplementary groups

Besides the primary group that is created when a new user account is added to the system, a user can be added to supplementary groups using the combined **-aG**, or **--append -groups** options, followed by a comma separated list of groups.

### EXAMPLE 3: Changing the default location of the user's home directory and / or changing its shell

If for some reason you need to change the default location of the user's home directory (other than **/home/username**), you will need to use the **-d**, or **--home** options, followed by the absolute path to the new home directory.

If a user wants to use another shell other than bash (for example, **sh**), which gets assigned by default, use **usermod** with the **--shell** flag, followed by the path to the new shell.

### EXAMPLE 4: Displaying the groups an user is a member of

After adding the user to a supplementary group, you can verify that it now actually belongs to such group(s):

```
# groups [username]
# id [username]
```

The following image depicts **Examples 2 through 4**:

```
[root@rhel7 ~]# usermod --append --groups gacanepa,users --home /tmp --shell /bin/sh tecmint
[root@rhel7 ~]# grep -i tecmint /etc/passwd
tecmint:x:1001:1001::/tmp:/bin/sh
[root@rhel7 ~]# groups tecmint
tecmint : tecmint users gacanepa
[root@rhel7 ~]# id tecmint
uid=1001(tecmint) gid=1001(tecmint) groups=1001(tecmint),100(users),1000(gacanepa)
[root@rhel7 ~]#
```

## Adding User to Supplementary Group

In the example above:

```
# usermod --append --groups gacanepa,users --home /tmp --shell /bin/sh tecmint
```

To remove a user from a group, omit the `--append` switch in the command above and list the groups you want the user to belong to following the `--groups` flag.

### EXAMPLE 5: Disabling account by locking password

To disable an account, you will need to use either the `-L` (lowercase L) or the `-lock` option to lock a user's password. This will prevent the user from being able to log on.

### EXAMPLE 6: Unlocking password

When you need to re-enable the user so that he can log on to the server again, use the `-U` or the `-unlock` option to unlock a user's password that was previously blocked, as explained in **Example 5** above.

```
# usermod --unlock tecmint
```

The following image illustrates **Examples 5** and **6**:

```
[root@rhel7 ~]# usermod --lock tecmint
[root@rhel7 ~]# su gacanepa
[gacanepa@rhel7 root]$ su tecmint
Password:
su: Authentication failure
[gacanepa@rhel7 root]$ su -
Password:
Last login: Sat Mar 14 09:41:24 EDT 2015 on pts/0
[root@rhel7 ~]# usermod --unlock tecmint
[root@rhel7 ~]# exit
logout
[gacanepa@rhel7 root]$ su tecmint
Password:
sh-4.2$
```

When root locks a user account, he or she will not be able to log on. Login attempts will fail with a "Authentication failure" message until root unlocks the account. In that case, the user will be allowed to use the system and will be presented with his/her default shell at the command line.

## Lock Unlock User Account

## EXAMPLE 7: Deleting a group or an user account

To delete a group, you'll want to use **groupdel**, whereas to delete a user account you will use **userdel** (add the **-r** switch if you also want to delete the contents of its home directory and mail spool):

```
# groupdel [group_name]          # Delete a group
# userdel -r [user_name]         # Remove user_name from the system, along
with his/her home directory and mail spool
```

If there are files owned by **group\_name**, they will not be deleted, but the group owner will be set to the **GID** of the group that was deleted.

## Listing, Setting and Changing Standard ugo/rwx Permissions

The well-known [ls command](#) is one of the best friends of any system administrator. When used with the **-l** flag, this tool allows you to view a list a directory's contents in long (or detailed) format.

However, this command can also be applied to a single file. Either way, the first 10 characters in the output of **ls -l** represent each file's attributes.

The first char of this **10-character** sequence is used to indicate the file type:

1. **-** (hyphen): a regular file
2. **d**: a directory
3. **l**: a symbolic link
4. **c**: a character device (which treats data as a stream of bytes, i.e. a terminal)
5. **b**: a block device (which handles data in blocks, i.e. storage devices)

The next nine characters of the file attributes, divided in groups of three from left to right, are called the file mode and indicate the read (**r**), write(**w**), and execute (**x**) permissions granted to the file's owner, the file's group owner, and the rest of the users (commonly referred to as "**the world**"), respectively.

While the read permission on a file allows the same to be opened and read, the same permission on a directory allows its contents to be listed if the execute permission is also set. In addition, the execute permission in a file allows it to be handled as a program and run.

File permissions are changed with the **chmod** command, whose basic syntax is as follows:

```
# chmod [new_mode] file
```

where **new\_mode** is either an octal number or an expression that specifies the new permissions. Feel free to use the mode that works best for you in each case. Or perhaps you already have a preferred way to set a file's permissions – so feel free to use the method that works best for you.

The octal number can be calculated based on the binary equivalent, which can in turn be obtained from the desired file permissions for the owner of the file, the owner group, and the

world. The presence of a certain permission equals a power of 2 (r=2<sup>2</sup>, w=2<sup>1</sup>, x=2<sup>0</sup>), while its absence means 0. For example:

r	w	x	r	-	-	r	-	-
4	2	1	4	0	0	4	0	0
$4+2+1=7$			$4+0+0=4$			$4+0+0=4$		

## File Permissions

To set the file's permissions as indicated above in octal form, type:

```
# chmod 744 myfile
```

Please take a minute to compare our previous calculation to the actual output of `ls -l` after changing the file's permissions:

```
[root@rhel7 ~]# ls -l myfile
-rw-r--r--. 1 root root 0 Mar 13 20:49 myfile
[root@rhel7 ~]# chmod 744 myfile
[root@rhel7 ~]# ls -l myfile
-rwxr--r--. 1 root root 0 Mar 13 20:49 myfile
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Long List Format

**EXAMPLE 8:** Searching for files with 777 permissions

As a security measure, you should make sure that files with **777** permissions (read, write, and execute for everyone) are avoided like the plague under normal circumstances. Although we will explain in a later tutorial how to more effectively locate all the files in your system with a certain permission set, you can -by now- combine `ls` with `grep` to obtain such information.

In the following example, we will look for file with **777** permissions in the `/etc` directory only. Note that we will use pipelining as explained in [Part 2: File and Directory Management](#) of this **RHCSA** series:

```
# ls -l /etc | grep rwxrwxrwx
[root@rhel7 ~]# ls -l /etc | grep rwxrwxrwx
1-rwxrwxrwx. 1 root root 56 Mar 13 20:03 favicon.png -> /usr/share/icons/hicolor/16x16/apps/fedora-logo-icon.png
1-rwxrwxrwx. 1 root root 22 Mar 13 20:04 grub2.cfg -> ../boot/grub2/grub.cfg
1-rwxrwxrwx. 1 root root 11 Mar 13 20:02 init.d -> rc.d/init.d
1-rwxrwxrwx. 1 root root 38 Mar 13 20:07 localtime -> ../usr/share/zoneinfo/America/New_York
1-rwxrwxrwx. 1 root root 17 Mar 13 20:02 mtab -> /proc/self/mounts
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc0.d -> rc.d/rc0.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc1.d -> rc.d/rc1.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc2.d -> rc.d/rc2.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc3.d -> rc.d/rc3.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc4.d -> rc.d/rc4.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc5.d -> rc.d/rc5.d
1-rwxrwxrwx. 1 root root 10 Mar 13 20:02 rc6.d -> rc.d/rc6.d
1-rwxrwxrwx. 1 root root 13 Mar 13 20:03 rc.local -> rc.d/rc.local
1-rwxrwxrwx. 1 root root 14 Mar 13 20:02 system-release -> redhat-release
All of these files are symbolic links to the actual
files they point to. In this case, it is OK to leave
the 777 permissions set.
But please beware of regular files or directories
with the same permissions set!
By default, new files are created with 644
permissions, while 755 permissions are assigned
to new directories.
http://www.tecmint.com
```

## Find All Files with 777 Permission

EXAMPLE 9: Assigning a specific permission to all users

Shell scripts, along with some binaries that all users should have access to (not just their corresponding owner and group), should have the execute bit set accordingly (please note that we will discuss a special case later):

```
# chmod a+x script.sh
```

**Note:** That we can also set a file's mode using an expression that indicates the owner's rights with the letter u, the group owner's rights with the letter g, and the rest with o. All of these rights can be represented at the same time with the letter a. Permissions are granted (or revoked) with the + or - signs, respectively.

```
[root@rhel7 ~]# ls -l script.sh
-rw-r--r--. 1 root root 0 Mar 13 21:21 script.sh
[root@rhel7 ~]# chmod a+x script.sh
[root@rhel7 ~]# ls -l script.sh
-rwxr-xr-x. 1 root root 0 Mar 13 21:21 script.sh
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Set Execute Permission on File

A long directory listing also shows the file's owner and its group owner in the first and second columns, respectively. This feature serves as a first-level access control method to files in a system:

```
[root@rhel7 ~]# ls -la /home/gacanepa
total 12
drwx----- 2 gacanepa gacanepa 59 Mar 13 20:02 .
drwxr-xr-x. 3 root      root     21 Mar 13 20:07 ..
-rw-r--r--. 1 gacanepa gacanepa 18 Jan 29 2014 .bash_logout
-rw-r--r--. 1 gacanepa gacanepa 193 Jan 29 2014 .bash_profile
-rw-r--r--. 1 gacanepa gacanepa 231 Jan 29 2014 .bashrc
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Check File Owner and Group

To change file ownership, you will use the **chown** command. Note that you can change the file and group ownership at the same time or separately:

```
# chown user:group file
```

**Note:** That you can change the user or group, or the two attributes at the same time, as long as you don't forget the colon, leaving user or group blank if you want to update the other attribute, for example:

```
# chown :group file          # Change group ownership only
# chown user: file           # Change user ownership only
```

EXAMPLE 10: Cloning permissions from one file to another

If you would like to “**clone**” ownership from one file to another, you can do so using the **--reference** flag, as follows:

```
# chown --reference=ref_file file
```

where the owner and group of **ref\_file** will be assigned to file as well:

```
[root@rhel7 ~]# ls -l testfile
-rw-r--r--. 1 lp mail 0 Mar 13 21:54 testfile
[root@rhel7 ~]# ls -l yetanotherfile
-rw-r--r--. 1 root root 0 Mar 13 21:54 yetanotherfile
[root@rhel7 ~]# chown --reference=testfile yetanotherfile
[root@rhel7 ~]# ls -l yetanotherfile
-rw-r--r--. 1 lp mail 0 Mar 13 21:54 yetanotherfile
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Clone File Ownership

### Setting Up SETGID Directories for Collaboration

Should you need to grant access to all the files owned by a certain group inside a specific directory, you will most likely use the approach of setting the **setgid** bit for such directory. When the **setgid** bit is set, the effective **GID** of the real user becomes that of the group owner.

Thus, any user can access a file under the privileges granted to the group owner of such file. In addition, when the **setgid** bit is set on a directory, newly created files inherit the same group as the directory, and newly created subdirectories will also inherit the **setgid** bit of the parent directory.

```
# chmod g+s [filename]
```

To set the **setgid** in octal form, prepend the number **2** to the current (or desired) basic permissions.

```
# chmod 2755 [directory]
```

## Conclusion

A solid knowledge of **user and group management**, along with standard and special Linux permissions, when coupled with practice, will allow you to quickly identify and troubleshoot issues with file permissions in your **RHEL 7** server.

I assure you that as you follow the steps outlined in this article and use the system documentation (as explained in [Part 1: Reviewing Essential Commands & System Documentation](#) of this series) you will master this essential competence of system administration.

Feel free to let us know if you have any questions or comments using the form below.

## RHCSA Series: Editing Text Files with Nano and Vim / Analyzing text with grep and regexps – Part 4

Every system administrator has to deal with text files as part of his daily responsibilities. That includes editing existing files (most likely configuration files), or creating new ones. It has been said that if you want to start a holy war in the Linux world, you can ask sysadmins what their favorite text editor is and why. We are not going to do that in this article, but will present a few tips that will be helpful to use two of the most widely used text editors in RHEL 7: **nano** (due to its simplicity and easiness of use, specially to new users), and **vi/m** (due to its several features that convert it into more than a simple editor). I am sure that you can find many more reasons to use one or the other, or perhaps some other editor such as **emacs** or **pico**. It's entirely up to you.

### RHCSA: Red Hat Certified System Administrator

### RED HAT ENTERPRISE LINUX 7



### Part 4: Editing text files with Nano and Vim / Analyzing text with grep and regexps in RHEL 7

RHCSA: Editing Text Files with Nano and Vim – Part 4

#### Editing Files with Nano Editor

To launch **nano**, you can either just type **nano** at the command prompt, optionally followed by a **filename** (in this case, if the file exists, it will be opened in edition mode). If the file does not exist, or if we omit the filename, nano will also be opened in edition mode but will present a blank screen for us to start typing:



## Nano Editor

As you can see in the previous image, **nano** displays at the bottom of the screen several functions that are available via the indicated shortcuts (^, aka caret, indicates the **Ctrl** key). To name a few of them:

1. **Ctrl + G**: brings up the help menu with a complete list of functions and descriptions:  
Ctrl + X: exits the current file. If changes have not been saved, they are discarded.
2. **Ctrl + R**: lets you choose a file to insert its contents into the present file by specifying a full path.

```
GNU nano 2.3.1                                         New Bu

Main nano help text

The nano editor is designed to emulate the functionality and ease-
the program version, the current filename being edited, and whether
The status line is the third line from the bottom and shows import

The notation for shortcuts is as follows: Control-key sequences ar
pressing the Escape (Esc) key twice. Escape-key sequences are not
on your keyboard setup. Also, pressing Esc twice and then typing
The following keystrokes are available in the main editor window.

^G      (F1)          Display this help text
^X      (F2)          Close the current file buffer / Exit from n
^O      (F3)          Write the current file to disk
^J      (F4)          Justify the current paragraph

^R      (F5)          Insert another file into the current one
^W      (F6)          Search for a string or a regular expression
^Y      (F7)          Go to previous screen
^V      (F8)          Go to next screen

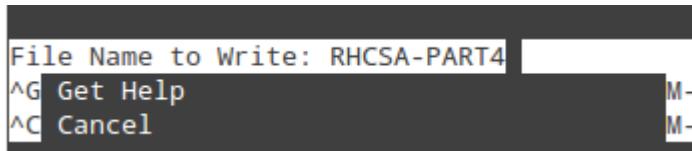
^K      (F9)          Cut the current line and store it in the cu
^U      (F10)         Uncut from the cutbuffer into the current l
^K      (F11)         Display the position of the cursor
^T      (F12)         Invoke the spell checker, if available

M-\   (M-|)         Go to the first line of the file
M-/   (M-?)         Go to the last line of the file

^_   (F13)  (M-G)   Go to line and column number
^R   (F14)  (M-R)   Replace a string or a regular expression
^A   (F15)  (M-A)   Mark text at the cursor position
```

Nano Editor Help Menu

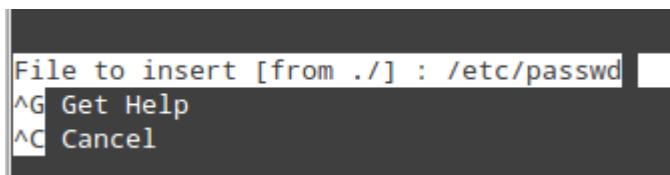
1. **Ctrl + O**: saves changes made to a file. It will let you save the file with the same name or a different one. Then press Enter to confirm.



A screenshot of the Nano Editor in save mode. The title bar says "File Name to Write: RHCSA-PART4". Below it is a menu with two options: "Get Help" and "Cancel".

Nano Editor Save Changes Mode

1. **Ctrl + X**: exits the current file. If changes have not been saved, they are discarded.
2. **Ctrl + R**: lets you choose a file to insert its contents into the present file by specifying a full path.



A screenshot of the Nano Editor in insert mode. The title bar says "File to insert [from ./] : /etc/passwd". Below it is a menu with two options: "Get Help" and "Cancel".

Nano: Insert File Content to Parent File

will insert the contents of **/etc/passwd** into the current file.

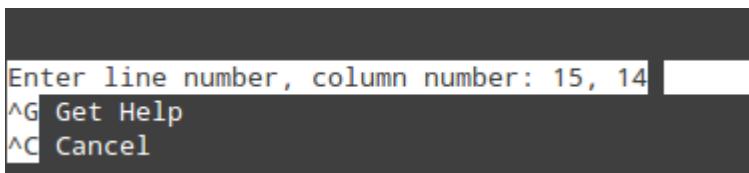
1. **Ctrl + K**: cuts the current line.
2. **Ctrl + U**: paste.
3. **Ctrl + C**: cancels the current operation and places you at the previous screen.

To easily navigate the opened file, nano provides the following features:

1. **Ctrl + F and Ctrl + B** move the cursor forward or backward, whereas **Ctrl + P** and **Ctrl + N** move it up or down one line at a time, respectively, just like the arrow keys.
2. **Ctrl + space and Alt + space** move the cursor forward and backward one word at a time.

Finally,

1. **Ctrl + \_ (underscore)** and then entering **X,Y** will take you precisely to Line **X**, column **Y**, if you want to place the cursor at a specific place in the document.



A screenshot of the Nano Editor in goto mode. The title bar says "Enter line number, column number: 15, 14". Below it is a menu with two options: "Get Help" and "Cancel".

Navigate to Line Numbers in Nano

The example above will take you to **line 15, column 14** in the current document.

If you can recall your early Linux days, specially if you came from Windows, you will probably agree that starting off with nano is the best way to go for a new user.

## Editing Files with Vim Editor

**Vim** is an improved version of **vi**, a famous text editor in Linux that is available on all POSIX-compliant \*nix systems, such as RHEL 7. If you have the chance and can install vim, go ahead; if not, most (if not all) the tips given in this article should also work.

One of vim's distinguishing features is the different modes in which it operates:

1. **Command mode** will allow you to browse through the file and enter commands, which are brief and case-sensitive combinations of one or more letters. If you need to repeat one of them a certain number of times, you can prefix it with a number (there are only a few exceptions to this rule). For example, yy (or Y, short for yank) copies the entire current line, whereas 4yy (or 4Y) copies the entire current line along with the next three lines (4 lines in total).
2. **In ex mode**, you can manipulate files (including saving a current file and running outside programs or commands). To enter ex mode, we must type a colon (:) starting from command mode (or in other words, Esc + :), directly followed by the name of the ex-mode command that you want to use.
3. **In insert mode**, which is accessed by typing the letter i, we simply enter text. Most keystrokes result in text appearing on the screen.
4. We can always enter command mode (regardless of the mode we're working on) by pressing the Esc key.

Let's see how we can perform the same operations that we outlined for **nano** in the previous section, but now with **vim**. Don't forget to hit the Enter key to confirm the vim command!

To access vim's full manual from the command line, type **:help** while in command mode and then press **Enter**:

```

help.txt      For Vim version 7.4. Last change: 2012 Dec 06

          VIM - main help file

Move around: Use the cursor keys, or "h" to go left,
              "j" to go down, "k" to go up, "l" to go right
Close this window: Use ":q<Enter>".
Get out of Vim: Use ":qa!<Enter>" (careful, all changes are lost)

Jump to a subject: Position the cursor on a tag (e.g. bars) and press :h
With the mouse:   Use ":set mouse=a" to enable the mouse (in xterm)
                  Double-click the left mouse button on a tag
Jump back:        Type CTRL-T or CTRL-O (repeat to go further back)

Get specific help: It is possible to go directly to whatever you want
                   on, by giving an argument to the :help command
                   It is possible to further specify the context
                           help-context

          WHAT           PREPEND     EXAMPLE
Normal mode command    (nothing)  :help

help.txt [Help][RO]
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin

```

## vim Edito Help Menu

The upper section presents an index list of contents, with defined sections dedicated to specific topics about **vim**. To navigate to a section, place the cursor over it and press **Ctrl + ]** (closing square bracket). Note that the bottom section displays the current file.

1. To save changes made to a file, run any of the following commands from command mode and it will do the trick:

```

:wq!
:x!
ZZ (yes, double Z without the colon at the beginning)

```

2. To exit discarding changes, use **:q!**. This command will also allow you to exit the help menu described above, and return to the current file in command mode.

3. Cut **N** number of lines: type **Ndd** while in command mode.
4. Copy **M** number of lines: type **Myy** while in command mode.
5. Paste lines that were previously cutted or copied: press the **P** key while in command mode.
6. To insert the contents of another file into the current one:

```
:r filename
```

For example, to insert the contents of `/etc/fstab`, do:

```
~  
~  
:r /etc/fstab
```

Insert Content of File in vi Editor

7. To insert the output of a command into the current document:

```
:r! command
```

For example, to insert the date and time in the line below the current position of the cursor:

```
~  
~  
:r! $(which date)
```

Insert Time an Date in vi Editor

In another article that I wrote for, ([Part 2 of the LFCS series](#)), I explained in greater detail the keyboard shortcuts and functions available in vim. You may want to refer to that tutorial for further examples on how to use this powerful text editor.

## Analyzing Text with Grep and Regular Expressions

By now you have learned how to create and edit files using **nano** or **vim**. Say you become a text editor ninja, so to speak – now what? Among other things, you will also need how to search for regular expressions inside text.

A regular expression (also known as “**regex**” or “**regexp**”) is a way of identifying a text string or pattern so that a program can compare the pattern against arbitrary text strings. Although the use of regular expressions along with grep would deserve an entire article on its own, let us review the basics here:

1. The simplest regular expression is an alphanumeric string (i.e., the word “**svm**”) or two (when two are present, you can use the | (**OR**) operator):

```
# grep -Ei 'svm|vmx' /proc/cpuinfo
```

The presence of either of those two strings indicate that your processor supports virtualization:

```
grep -Ei 'svm|vmx' /proc/cpuinfo  
ou vme de pse tsc msr pae mce cx8 apic  
t_tsc rep_good nopl nonstop_tsc extd_a  
ot lbrv svm_lock nrip_save  
ou vme de pse tsc msr pae mce cx8 apic  
t_tsc rep_good nopl nonstop_tsc extd_a  
ot lbrv svm_lock nrip_save
```

Regular Expression Example

## 2. A second kind of a regular expression is a range list, enclosed between square brackets.

For example, `c[aeiou]t` matches the strings cat, cet, cit, cot, and cut, whereas `[a-z]` and `[0-9]` match any lowercase letter or decimal digit, respectively. If you want to repeat the regular expression **X** certain number of times, type `{x}` immediately following the regexp.

For example, let's extract the **UUIDs** of storage devices from `/etc/fstab`:

```
# grep -Ei '[0-9a-f]{8}-([0-9a-f]{4}-){3}[0-9a-f]{12}' -o /etc/fstab
[root@rhel7 ~]# grep -Ei '[0-9a-f]{8}-([0-9a-f]{4}-){3}[0-9a-f]{12}' -o /etc/fstab
2d78977c-2a59-4962-bf1f-33d9b5f5076f
[root@rhel7 ~]# grep -i UUID /etc/fstab
UUID=2d78977c-2a59-4962-bf1f-33d9b5f5076f /boot          xfs      defaults
[root@rhel7 ~]#
```

### Extract String from a File

The first expression in brackets `[0-9a-f]` is used to denote lowercase hexadecimal characters, and `{8}` is a quantifier that indicates the number of times that the preceding match should be repeated (the first sequence of characters in an **UUID** is a 8-character long hexadecimal string).

The parentheses, the `{4}` quantifier, and the hyphen indicate that the next sequence is a 4-character long hexadecimal string, and the quantifier that follows (`{3}`) denote that the expression should be repeated 3 times.

Finally, the last sequence of 12-character long hexadecimal string in the **UUID** is retrieved with `[0-9a-f]{12}`, and the `-o` option prints only the matched (non-empty) parts of the matching line in `/etc/fstab`.

## 3. POSIX character classes.

Character Class	Matches...
<code>[:alnum:]</code>	Any alphanumeric [a-zA-Z0-9] character
<code>[:alpha:]</code>	Any alphabetic [a-zA-Z] character
<code>[:blank:]</code>	Spaces or tabs
<code>[:cntrl:]</code>	Any control characters (ASCII 0 to 32)
<code>[:digit:]</code>	Any numeric digits [0-9]
<code>[:graph:]</code>	Any visible characters
<code>[:lower:]</code>	Any lowercase [a-z] character
<code>[:print:]</code>	Any non-control characters

[:space:]	Any whitespace
[:punct:]	Any punctuation marks
[:upper:]	Any uppercase [A-Z] character
[:xdigit:]	Any hex digits [0-9a-fA-F]
:word:	Any letters, numbers, and underscores [a-zA-Z0-9_]

For example, we may be interested in finding out what the used **UIDs** and **GIDs** (refer to [Part 2](#) of this series to refresh your memory) are for real users that have been added to our system. Thus, we will search for sequences of 4 digits in **/etc/passwd**:

```
# grep -Ei [:digit:]{4} /etc/passwd
[root@rhel7 ~]# grep -Ei [:digit:]{4} /etc/passwd
gacanepa:x:1000:1000:Gabriel A. Cánepa:/home/gacanepa:/bin/bash
tecmint:x:1001:1001::/tmp:/bin/sh
[root@rhel7 ~]#
```

### Search For a String in File

The above example may not be the best case of use of regular expressions in the real world, but it clearly illustrates how to use **POSIX** character classes to analyze text along with **grep**.

## Conclusion

In this article we have provided some tips to make the most of **nano** and **vim**, two text editors for the command-line users. Both tools are supported by extensive documentation, which you can consult in their respective official web sites (links given below) and using the suggestions given in [Part 1](#) of this series.

### Reference Links

<http://www.nano-editor.org/>  
<http://www.vim.org/>

## RHCSA Series: Process Management in RHEL 7: Boot, Shutdown, and Everything in Between – Part 5

We will start this article with an overall and brief revision of what happens since the moment you press the **Power** button to turn on your **RHEL 7** server until you are presented with the login screen in a command line interface.

# RHCSA: Red Hat Certified System Administrator

## RED HAT ENTERPRISE LINUX 7



### Part 5: Process management in RHEL 7: boot, shutdown, and everything in between

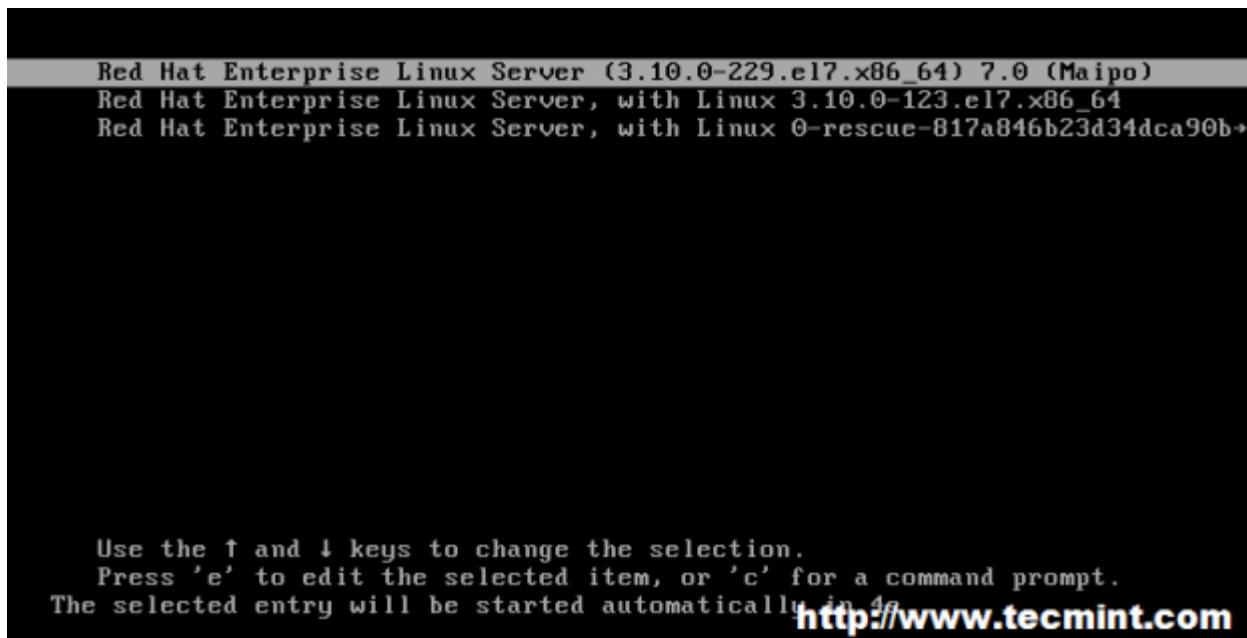
#### Linux Boot Process

Please note that:

1. the same basic principles apply, with perhaps minor modifications, to other Linux distributions as well, and
2. the following description is not intended to represent an exhaustive explanation of the boot process, but only the fundamentals.

#### Linux Boot Process

1. The **POST** (Power On Self Test) initializes and performs hardware checks.
2. When the **POST** finishes, the system control is passed to the first stage boot loader, which is stored on either the boot sector of one of the hard disks (for older systems using BIOS and MBR), or a dedicated (U)EFI partition.
3. The first stage boot loader then loads the second stage boot loader, most usually **GRUB** (**GRAnd Unified Boot Loader**), which resides inside **/boot**, which in turn loads the kernel and the initial RAM-based file system (also known as **initramfs**, which contains programs and binary files that perform the necessary actions needed to ultimately mount the actual root filesystem).
4. We are presented with a splash screen that allows us to choose an operating system and kernel to boot:



## Boot Menu Screen

5. The kernel sets up the hardware attached to the system and once the root filesystem has been mounted, launches process with **PID 1**, which in turn will initialize other processes and present us with a login prompt.

**Note:** That if we wish to do so at a later time, we can examine the specifics of this process using the [dmesg command](#) and filtering its output using the tools that we have explained in previous articles of this series.

The screenshot shows a terminal window with a black background and white text. It displays the output of the command "ps -o ppid,pid,uname,comm --ppid=1", which lists various system processes. Above the command output, there is a small black box containing the text: "Red Hat Enterprise Linux  
Kernel 3.10.0-229.el7.x86\_64 on an x86\_64  
rhel7 login: \_". A red watermark "http://www.tecmint.com" is visible in the bottom right corner.

PPID	PID	USER	COMMAND
1	456	root	systemd-journal
1	461	root	lvmetad
1	467	root	systemd-udevd
1	537	root	auditd
1	559	root	firewalld
1	562	avahi	avahi-daemon
1	563	root	rsyslogd
1	564	root	tuned
1	568	root	systemd-logind
1	569	dbus	dbus-daemon
1	571	root	crond
1	575	root	agetty
1	653	root	NetworkManager
1	733	polkitd	polkitd
1	1206	root	sshd
1	1211	root	rhsmdcertd
1	1690	root	master

## Login Screen and Process PID

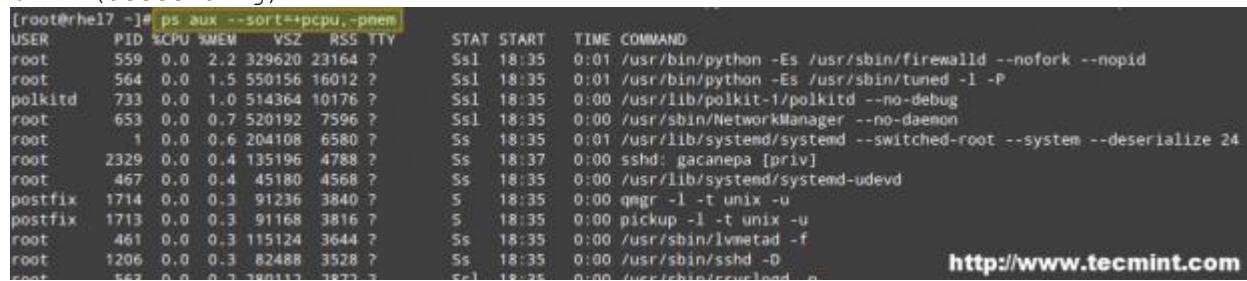
In the example above, we used the well-known **ps command** to display a list of current processes whose parent process (or in other words, the process that started them) is **systemd** (the system and service manager that most modern Linux distributions have switched to) during system startup:

```
# ps -o ppid,pid,uname,comm --ppid=1
```

Remember that the **-o** flag (short for **-format**) allows you to present the output of **ps** in a customized format to suit your needs using the keywords specified in the **STANDARD FORMAT SPECIFIERS** section in **man ps**.

Another case in which you will want to define the output of **ps** instead of going with the default is when you need to find processes that are causing a significant CPU and / or memory load, and sort them accordingly:

```
# ps aux --sort=+pcpu          # Sort by %CPU (ascending)
# ps aux --sort=-pcpu          # Sort by %CPU (descending)
# ps aux --sort=+pmem          # Sort by %MEM (ascending)
# ps aux --sort=-pmem          # Sort by %MEM (descending)
# ps aux --sort=+pcpu,-pmem    # Combine sort by %CPU (ascending) and
%MEM (descending)
```



The screenshot shows a terminal window with the following output:

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	559	0.0	2.2	329620	23164	?	Ssl	18:35	0:01	/usr/bin/python -Es /usr/sbin/firewalld --nofork --nrepid
root	564	0.0	1.5	550156	16012	?	Ssl	18:35	0:01	/usr/bin/python -Es /usr/sbin/tuned -l -P
polkitd	733	0.0	1.0	514364	10176	?	Ssl	18:35	0:00	/usr/lib/polkit-1/polkitd --no-debug
root	653	0.0	0.7	520192	7596	?	Ssl	18:35	0:00	/usr/sbin/NetworkManager --no-daemon
root	1	0.0	0.6	204108	6580	?	Ss	18:35	0:01	/usr/lib/systemd/systemd --switched-root --system --deserialize 24
root	2329	0.0	0.4	135196	4788	?	Ss	18:37	0:00	sshd: gacanepa [priv]
root	467	0.0	0.4	45180	4568	?	Ss	18:35	0:00	/usr/lib/systemd/systemd-udevd
postfix	1714	0.0	0.3	91236	3840	?	S	18:35	0:00	qmgr -l -t unix -u
postfix	1713	0.0	0.3	91168	3816	?	S	18:35	0:00	pickup -l -t unix -u
root	461	0.0	0.3	115124	3644	?	Ss	18:35	0:00	/usr/sbin/lvmetad -f
root	1206	0.0	0.3	82488	3528	?	Ss	18:35	0:00	/usr/sbin/sshd -D
root	563	0.0	0.2	380112	3872	?	Ssl	18:35	0:00	/usr/sbin/crond -n

## Customize ps Command Output

## An Introduction to SystemD

Few decisions in the Linux world have caused more controversies than the adoption of **systemd** by major Linux distributions. Systemd's advocates name as its main advantages the following facts:

**Read Also:** [The Story Behind ‘init’ and ‘systemd’](#)

1. Systemd allows more processing to be done in parallel during system startup (as opposed to older **SysVinit**, which always tends to be slower because it starts processes one by one, checks if one depends on another, and then waits for daemons to launch so more services can start), and
2. It works as a dynamic resource management in a running system. Thus, services are started when needed (to avoid consuming system resources if they are not being used) instead of being launched without a valid reason during boot.
3. Backwards compatibility with **SysVinit** scripts.

**Systemd** is controlled by the **systemctl** utility. If you come from a **SysVinit** background, chances are you will be familiar with:

1. the **service** tool, which -in those older systems- was used to manage SysVinit scripts, and
2. the [chkconfig](#) utility, which served the purpose of updating and querying runlevel information for system services.

3. **shutdown**, which you must have used several times to either restart or halt a running system.

The following table shows the similarities between the use of these legacy tools and **systemctl**:

<b>Legacy tool</b>	<b>Systemctl equivalent</b>	<b>Description</b>
service name start	systemctl start name	Start name (where name is a service)
service name stop	systemctl stop name	Stop name
service name condrestart	systemctl try-restart name	Restarts name (if it's already running)
service name restart	systemctl restart name	Restarts name
service name reload	systemctl reload name	Reloads the configuration for name
service name status	systemctl status name	Displays the current status of name
service -status-all	systemctl	Displays the status of all current services
chkconfig name on	systemctl enable name	Enable name to run on startup as specified in the unit file (the file to which the symlink points). The process of enabling or disabling a service to start automatically on boot consists in adding or removing symbolic links inside the /etc/systemd/system directory.
chkconfig name off	systemctl disable name	Disables name to run on startup as specified in the unit file (the file to which the symlink points)
chkconfig -list name	systemctl is-enabled name	Verify whether name (a specific service) is currently enabled
chkconfig -list	systemctl -type=service	Displays all services and tells whether they are enabled or disabled
shutdown -h now	systemctl poweroff	Power-off the machine (halt)
shutdown -r now	systemctl reboot	Reboot the system

**Systemd** also introduced the concepts of units (which can be either a service, a mount point, a device, or a network socket) and targets (which is how systemd manages to start several related process at the same time, and can be considered -though not equal- as the equivalent of runlevels in **SysVinit-based** systems).

## Summing Up

Other tasks related with process management include, but may not be limited to, the ability to:

1. Adjust the execution priority as far as the use of system resources is concerned of a process:

This is accomplished through the **renice** utility, which alters the scheduling priority of one or more running processes. In simple terms, the scheduling priority is a feature that allows the kernel (present in versions => **2.6**) to allocate system resources as per the assigned execution priority (aka niceness, in a range from **-20** through **19**) of a given process.

The basic syntax of **renice** is as follows:

```
# renice [-n] priority [-gpu] identifier
```

In the generic command above, the first argument is the priority value to be used, whereas the other argument can be interpreted as process **IDs** (which is the default setting), process group IDs, user IDs, or user names. A normal user (other than root) can only modify the scheduling priority of a process he or she owns, and only increase the niceness level (which means taking up less system resources).

```
[gacanepa@rhel7 ~]$ ps -efl | grep top | grep -v grep
0 5 gacanepa 10079 10053 0 80 0 - 32473 poll_s 21:32 tty1 00:00:00 top
[gacanepa@rhel7 ~]$ renice -1 -u gacanepa
renice: failed to set priority for 10000 (user ID): Permission denied
[gacanepa@rhel7 ~]$ renice -1 -p 10079
renice: failed to set priority for 10079 (process ID): Permission denied
[gacanepa@rhel7 ~]$ renice +1 -p 10079
10079 (process ID) old priority 0, new priority 1
[gacanepa@rhel7 ~]$ ps -efl | grep top | grep -v grep
0 5 gacanepa 10079 10053 0 81 0 - 32473 poll_s 21:32 tty1 00:00:00 top
[gacanepa@rhel7 ~]$
```

As we can see, user gacanepa is running the top command with a priority of 0. If he tries to decrease this value, he will get an error message that says that he does not have permissions to do so. In addition, he can't decrease the priority of any process he owns. However, he can increase this value. This goes to show that a regular user cannot modify the scheduling priority of a given process in such a way that it will require more system resources as result.

<http://www.tecmint.com>

## Process Scheduling Priority

2. Kill (or interrupt the normal execution) of a process as needed:

In more precise terms, killing a process entitles sending it a signal to either finish its execution gracefully (**SIGTERM=15**) or immediately (**SIGKILL=9**) through the [kill or pkill commands](#).

The difference between these two tools is that the former is used to terminate a specific process or a process group altogether, while the latter allows you to do the same based on name and other attributes.

In addition, **pkill** comes bundled with **pgrep**, which shows you the PIDs that will be affected should **pkill** be used. For example, before running:

```
# pkill -u gacanepa
```

It may be useful to view at a glance which are the **PIDs** owned by **gacanepa**:

```
# pgrep -l -u gacanepa
[root@rhel7 ~]# pgrep -l -u gacanepa
2333 sshd
2334 bash
10053 bash
10079 top
[root@http://www.tecmint.com
```

## Find PIDs of User

By default, both **kill** and **pkill** send the **SIGTERM** signal to the process. As we mentioned above, this signal can be ignored (while the process finishes its execution or for good), so when you seriously need to stop a running process with a valid reason, you will need to specify the **SIGKILL** signal on the command line:

```
# kill -9 identifier          # Kill a process or a process group
# kill -s SIGNAL identifier   # Idem
# pkill -s SIGNAL identifier  # Kill a process by name or other
                             # attributes
```

## Conclusion

In this article we have explained the basics of the **boot process** in a **RHEL 7** system, and analyzed some of the tools that are available to help you with managing processes using common utilities and systemd-specific commands.

Note that this list is not intended to cover all the bells and whistles of this topic, so feel free to add your own preferred tools and commands to this article using the comment form below. Questions and other comments are also welcome.

## RHCSA Series: Using ‘Parted’ and ‘SSM’ to Configure and Encrypt System Storage – Part 6

In this article we will discuss how to set up and configure local system storage in **Red Hat Enterprise Linux 7** using classic tools and introducing the **System Storage Manager** (also known as **SSM**), which greatly simplifies this task.

# RHCSA: Red Hat Certified System Administrator

## RED HAT ENTERPRISE LINUX 7



### Part 6: Using parted and ssm to configure and encrypt system storage in RHEL 7

RHCSA: Configure and Encrypt System Storage – Part 6

Please note that we will present this topic in this article but will continue its description and usage on the next one (Part 7) due to vastness of the subject.

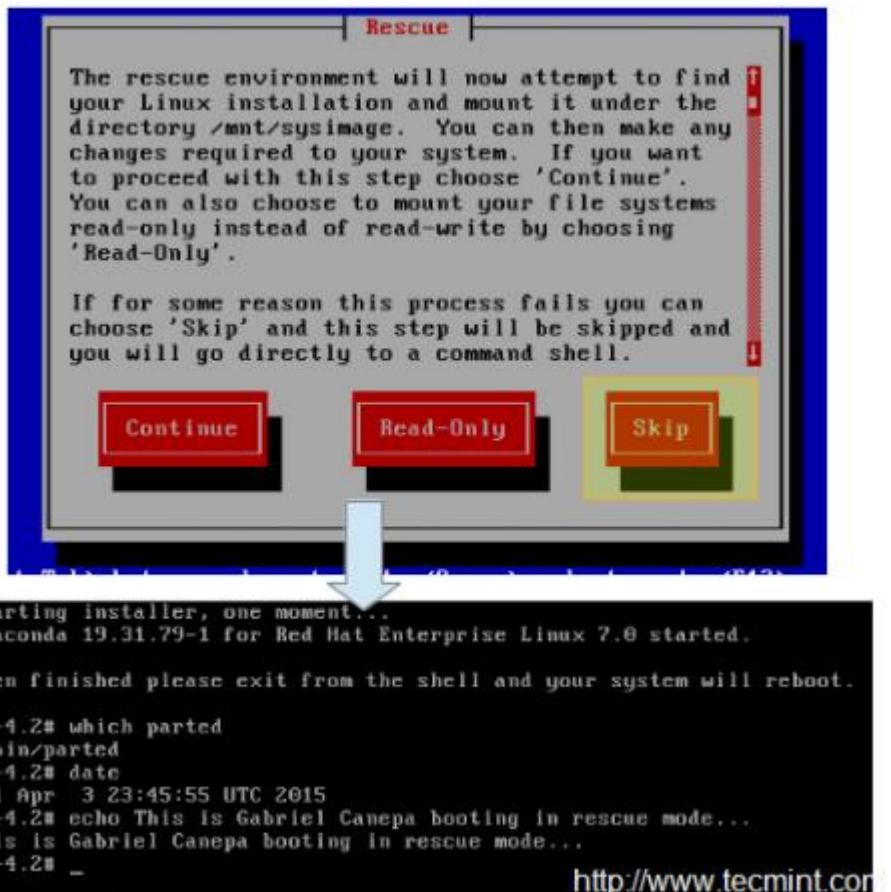
#### Creating and Modifying Partitions in RHEL 7

In RHEL 7, **parted** is the default utility to work with partitions, and will allow you to:

1. Display the current partition table
2. Manipulate (increase or decrease the size of) existing partitions
3. Create partitions using free space or additional physical storage devices

It is recommended that before attempting the creation of a new partition or the modification of an existing one, you should ensure that none of the partitions on the device are in use (`umount /dev/partition`), and if you're using part of the device as swap you need to disable it (`swapoff -v /dev/partition`) during the process.

The easiest way to do this is to boot **RHEL** in **rescue mode** using an installation media such as a **RHEL 7** installation **DVD** or **USB** (**Troubleshooting → Rescue a Red Hat Enterprise Linux system**) and Select **Skip** when you're prompted to choose an option to mount the existing Linux installation, and you will be presented with a command prompt where you can start typing the same commands as shown as follows during the creation of an ordinary partition in a physical device that is not being used.



## RHEL 7 Rescue Mode

To start **parted**, simply type.

```
# parted /dev/sdb
```

Where `/dev/sdb` is the device where you will create the new partition; next, type **print** to display the current drive's partition table:

```
[root@rhel7 ~]# parted /dev/sdb
GNU Parted 3.1
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: /dev/sdb: unrecognised disk label
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
(parted) -
```

<http://www.tecmint.com>

## Create New Partition

As you can see, in this example we are using a virtual drive of **5 GB**. We will now proceed to create a **4 GB** primary partition and then format it with the **xfs** filesystem, which is the default in **RHEL 7**.

You can choose from a variety of file systems. You will need to manually create the partition with **mkpart** and then format it with **mkfs.fstype** as usual because **mkpart** does not support many modern filesystems out-of-the-box.

In the following example we will set a label for the device and then create a primary partition (p) on /dev/sdb, which starts at the **0%** percentage of the device and ends at **4000 MB (4 GB)**:

```
(parted) mklabel msdos
(parted) mkpart
Partition type? primary/extended? p
File system type? [ext2]?
Start? 0%
End? 4000
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  4000MB  3999MB  primary

(parted) █
```

<http://www.tecmint.com>

#### Label Partition Name

Next, we will format the partition as **xfs** and print the partition table again to verify that changes were applied:

```
# mkfs.xfs /dev/sdb1
# parted /dev/sdb print
```

```
[root@rhel7 ~]# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=256    agcount=4, agsize=244096 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=0     finobt=0
data     =                      bsize=4096   blocks=976384, imaxpct=25
                                =                      sunit=0   swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0 ftype=0
log      =internal log          bsize=4096   blocks=2560, version=2
                                =                      sectsz=512  sunit=0 blks, lazy-count=1
realtime =none                  extsz=4096   blocks=0, rtextents=0
[root@rhel7 ~]# parted /dev/sdb print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB 4000MB 3999MB  primary   xfs

[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Format Partition as XFS Filesystem

For older filesystems, you could use the **resize** command in **parted** to resize a partition. Unfortunately, this only applies to ext2, fat16, fat32, hfs, linux-swap, and reiserfs (if libreiserfs is installed).

Thus, the only way to resize a partition is by deleting it and creating it again (so make sure you have a good backup of your data!). No wonder the default partitioning scheme in **RHEL 7** is based on **LVM**.

To remove a partition with **parted**:

```
# parted /dev/sdb print
# parted /dev/sdb rm 1
```

```
[root@rhel7 ~]# parted /dev/sdb print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1        1049kB 4000MB  3999MB  primary   xfs

[root@rhel7 ~]# parted /dev/sdb rm 1
Information: You may need to update /etc/fstab.

[root@rhel7 ~]# parted /dev/sdb print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags

```

**When you create or remove a partition, you will get a reminder that you may need to update /etc/fstab.**

**To remove a partition, you will use the rm command of parted followed by the partition number (1 in this case)**

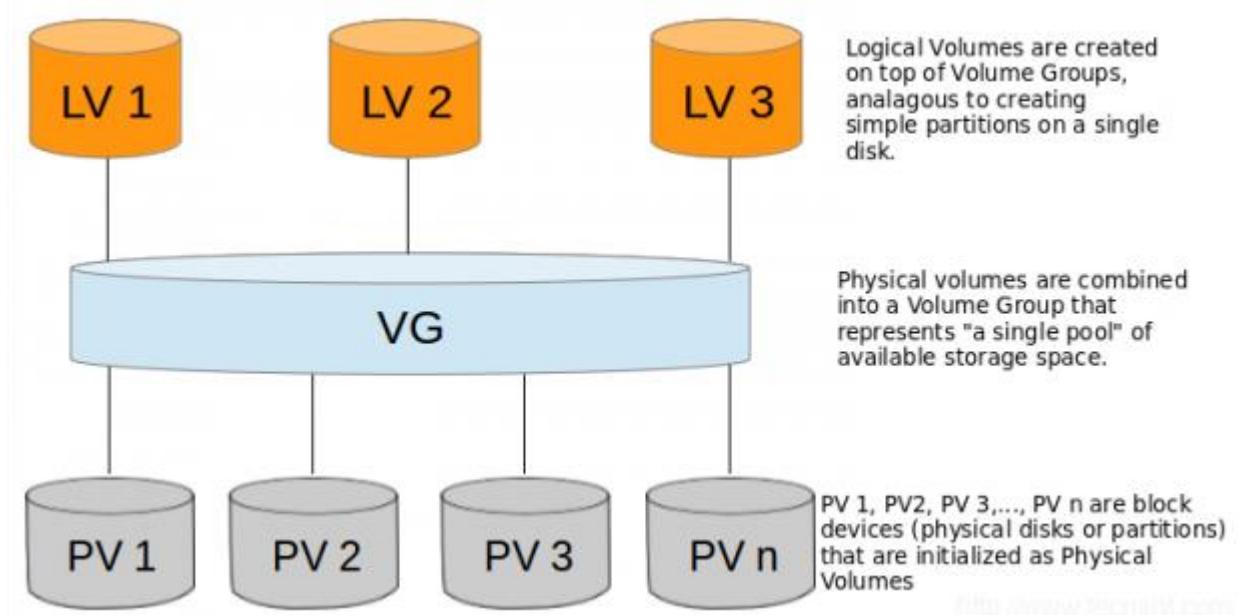
<http://www.tecmint.com>

Remove or Delete Partition

## The Logical Volume Manager (LVM)

Once a disk has been partitioned, it can be difficult or risky to change the partition sizes. For that reason, if we plan on resizing the partitions on our system, we should consider the possibility of using **LVM** instead of the classic partitioning system, where several physical devices can form a volume group that will host a defined number of logical volumes, which can be expanded or reduced without any hassle.

In simple terms, you may find the following diagram useful to remember the basic architecture of **LVM**.



## Basic Architecture of LVM

### Creating Physical Volumes, Volume Group and Logical Volumes

Follow these steps in order to set up **LVM** using classic volume management tools. Since you can expand this topic reading the [LVM series on this site](#), I will only outline the basic steps to set up LVM, and then compare them to implementing the same functionality with SSM.

**Note:** That we will use the whole disks `/dev/sdb` and `/dev/sdc` as **PVs (Physical Volumes)** but it's entirely up to you if you want to do the same.

1. Create partitions `/dev/sdb1` and `/dev/sdc1` using **100%** of the available disk space in **/dev/sdb** and **/dev/sdc**:

```
# parted /dev/sdb print
# parted /dev/sdc print
```

```
[root@rhel7 ~]# parted /dev/sdb print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  5369MB  5368MB  primary

[root@rhel7 ~]# parted /dev/sdc print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdc: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  5369MB  5368MB  primary

[root@rhel7 ~]#
```

## Create New Partitions

2. Create 2 physical volumes on top of `/dev/sdb1` and `/dev/sdc1`, respectively.

```
# pvcreate /dev/sdb1
# pvcreate /dev/sdc1
[root@rhel7 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
[root@rhel7 ~]# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
[root@rhel7 ~]#
```

## Create Two Physical Volumes

Remember that you can use `pvdisplay /dev/sd{b,c}1` to show information about the newly created **PVs**.

3. Create a **VG** on top of the **PV** that you created in the previous step:

```
# vgcreate tecmint_vg /dev/sd{b,c}1
[root@rhel7 ~]# vgcreate tecmint_vg /dev/sd{b,c}1
  Volume group "tecmint_vg" successfully created
[root@rhel7 ~]#
```

## Create Volume Group

Remember that you can use `vgdisplay tecmint_vg` to show information about the newly created **VG**.

4. Create three logical volumes on top of **VG tecmint\_vg**, as follows:

```

# lvcreate -L 3G -n vol01_docs tecmint_vg
# lvcreate -L 1G -n vol02_logs tecmint_vg
# lvcreate -l 100%FREE -n vol03_homes tecmint_vg
[vol01_docs → 3 GB]
[vol02_logs → 1 GB]
[vol03_homes → 6
GB]

[root@rhel7 ~]# lvcreate -L 3G -n vol01_docs tecmint_vg
Logical volume "vol01_docs" created.
[root@rhel7 ~]# lvcreate -L 1G -n vol02_logs tecmint_vg
Logical volume "vol02_logs" created.
[root@rhel7 ~]# lvcreate -l 100%FREE -n vol03_homes tecmint_vg
Logical volume "vol03_homes" created.
[root@rhel7 ~]#

```

## Create Logical Volumes

Remember that you can use **lvdisplay tecmint\_vg** to show information about the newly created **LVs** on top of **VG tecmint\_vg**.

**5.** Format each of the logical volumes with **xfs** (do NOT use xfs if you're planning on shrinking volumes later!):

```

# mkfs.xfs /dev/tecmint_vg/vol01_docs
# mkfs.xfs /dev/tecmint_vg/vol02_logs
# mkfs.xfs /dev/tecmint_vg/vol03_homes

```

**6.** Finally, mount them:

```

# mount /dev/tecmint_vg/vol01_docs /mnt/docs
# mount /dev/tecmint_vg/vol02_logs /mnt/logs
# mount /dev/tecmint_vg/vol03_homes /mnt/homes

```

## Removing Logical Volumes, Volume Group and Physical Volumes

**7.** Now we will reverse the **LVM** implementation and remove the **LVs**, the **VG**, and the **PVs**:

```

# lvremove /dev/tecmint_vg/vol01_docs
# lvremove /dev/tecmint_vg/vol02_logs
# lvremove /dev/tecmint_vg/vol03_homes
# vgremove /dev/tecmint_vg
# pvremove /dev/sd{b,c}1

```

**8.** Now let's install **SSM** and we will see how to perform the above in **ONLY 1 STEP!**

```
# yum update && yum install system-storage-manager
```

We will use the same names and sizes as before:

```

# ssm create -s 3G -n vol01_docs -p tecmint_vg --fstype ext4 /mnt/docs
/dev/sd{b,c}1
# ssm create -s 1G -n vol02_logs -p tecmint_vg --fstype ext4 /mnt/logs
/dev/sd{b,c}1
# ssm create -n vol03_homes -p tecmint_vg --fstype ext4 /mnt/homes
/dev/sd{b,c}1

```

Yes! **SSM** will let you:

1. initialize block devices as physical volumes
2. create a volume group
3. create logical volumes
4. format LVs, and
5. mount them using only one command

**9.** We can now display the information about **PVs**, **VGs**, or **LVs**, respectively, as follows:

```
# ssm list dev
# ssm list pool
# ssm list vol
[root@rhel7 ~]# ssm list dev
-----[redacted]-----
Device      Free     Used     Total   Pool      Mount point
-----[redacted]-----
/dev/sda            30.00 GB          PARTITIONED
/dev/sda1           500.00 MB          /boot
/dev/sda2    0.00 KB  29.51 GB  29.51 GB  rhel
/dev/sdb            5.00 GB
-----[redacted]-----
/dev/sdb1           5.00 GB  tecmint_vg
/dev/sdc            5.00 GB
-----[redacted]-----
/dev/sdc1           5.00 GB  tecmint_vg

[root@rhel7 ~]# ssm list pool
-----[redacted]-----
Pool      Type  Devices     Free     Used     Total
-----[redacted]-----
rhel      lvm   1        0.00 KB  29.51 GB  29.51 GB
tecmint_vg  lvm   2        0.00 KB  9.99 GB   9.99 GB

[root@rhel7 ~]# ssm list vol
-----[redacted]-----
Volume      Pool      Volume size   FS      FS size     Free     Type      Mount point
-----[redacted]-----
/dev/rhel/root    rhel      27.48 GB  xfs    27.46 GB  26.27 GB  linear    /
/dev/rhel/swap    rhel      2.03 GB
-----[redacted]-----
/dev/tecmint_vg/vol01_docs  tecmint_vg  3.00 GB  ext4    3.00 GB  2.73 GB  linear    /mnt/docs
/dev/tecmint_vg/vol02_logs   tecmint_vg  1.00 GB  ext4    1.00 GB  922.20 MB linear    /mnt/logs
/dev/tecmint_vg/vol03_homes  tecmint_vg  5.99 GB  ext4    5.99 GB  5.45 GB  linear    /mnt/homes
-----[redacted]-----
/dev/sda1           500.00 MB xfs   496.67 MB 343.77 MB part    /boot

[root@rhel7 ~]#
```

Check Information of PVs, VGs, or LVs

**10.** As we already know, one of the distinguishing features of LVM is the possibility to **resize** (expand or decrease) logical volumes without downtime.

Say we are running out of space in **vol02\_logs** but have plenty of space in **vol03\_homes**. We will resize **vol03\_homes** to **4 GB** and expand **vol02\_logs** to use the remaining space:

```
# ssm resize -s 4G /dev/tecmint_vg/vol03_homes
```

Run **ssm list pool** again and take note of the free space in **tecmint\_vg**:

```
-----[redacted]-----
Pool      Type  Devices     Free     Used     Total
-----[redacted]-----
rhel      lvm   1        0.00 KB  29.51 GB  29.51 GB
tecmint_vg  lvm   2        1.99 GB  8.00 GB   9.99 GB
-----[redacted]-----
```

## Check Volume Size

Then do:

```
# ssm resize -s+1.99 /dev/tecmint_vg/vol02_logs
```

**Note:** that the plus sign after the `-s` flag indicates that the specified value should be added to the present value.

**11.** Removing logical volumes and volume groups is much easier with ssm as well. A simple,

```
# ssm remove tecmint_vg
```

will return a prompt asking you to confirm the deletion of the **VG** and the **LVs** it contains:

```
[root@rhel7 ~]# ssm remove tecmint_vg
Do you really want to remove volume group "tecmint_vg" containing 3 logical volumes? [y/n]: y
Do you really want to remove active logical volume vol01_docs? [y/n]: y
Logical volume "vol01_docs" successfully removed
Do you really want to remove active logical volume vol02_logs? [y/n]: y
Logical volume "vol02_logs" successfully removed
Do you really want to remove active logical volume vol03_homes? [y/n]: y
Logical volume "vol03_homes" successfully removed
Volume group "tecmint_vg" successfully removed
[root@rhel7 ~]# █
```

## Remove Logical Volume and Volume Group

## Managing Encrypted Volumes

**SSM** also provides system administrators with the capability of managing encryption for new or existing volumes. You will need the **cryptsetup** package installed first:

```
# yum update && yum install cryptsetup
```

Then issue the following command to create an encrypted volume. You will be prompted to enter a **passphrase** to maximize security:

```
# ssm create -s 3G -n vol01_docs -p tecmint_vg --fstype ext4 --encrypt luks
/mnt/docs /dev/sd{b,c}1
# ssm create -s 1G -n vol02_logs -p tecmint_vg --fstype ext4 --encrypt luks
/mnt/logs /dev/sd{b,c}1
# ssm create -n vol03_homes -p tecmint_vg --fstype ext4 --encrypt luks
/mnt/homes /dev/sd{b,c}1
```

Our next task consists in adding the corresponding entries in **/etc/fstab** in order for those logical volumes to be available on boot. Rather than using the device identifier (**/dev/something**).

We will use each **LV's UUID** (so that our devices will still be uniquely identified should we add other logical volumes or devices), which we can find out with the **blkid** utility:

```
# blkid -o value UUID /dev/tecmint_vg/vol01_docs
# blkid -o value UUID /dev/tecmint_vg/vol02_logs
```

```
# blkid -o value UUID /dev/tecmint_vg/vol03_homes
```

In our case:

```
[root@rhel7 ~]# blkid -o value UUID /dev/tecmint_vg/vol01_docs  
ba77d113-f849-4ddf-8048-13860399fca8  
crypto_LUKS  
[root@rhel7 ~]# blkid -o value UUID /dev/tecmint_vg/vol02_logs  
58f89c5a-f694-4443-83d6-2e83878e30e4  
crypto_LUKS  
[root@rhel7 ~]# blkid -o value UUID /dev/tecmint_vg/vol03_homes  
92245af6-3f38-4e07-8dd8-787f4690d7ac  
crypto_LUKS  
[root@rhel7 ~]#
```

Find Logical Volume UUID

Next, create the **/etc/crypttab** file with the following contents (change the **UUIDs** for the ones that apply to your setup):

```
docs UUID=ba77d113-f849-4ddf-8048-13860399fca8 none  
logs UUID=58f89c5a-f694-4443-83d6-2e83878e30e4 none  
homes UUID=92245af6-3f38-4e07-8dd8-787f4690d7ac none
```

And insert the following entries in **/etc/fstab**. Note that **device\_name** (**/dev/mapper/device\_name**) is the mapper identifier that appears in the first column of **/etc/crypttab**.

```
# Logical volume vol01_docs:  
/dev/mapper/docs    /mnt/docs      ext4    defaults      0      2  
# Logical volume vol02_logs  
/dev/mapper/logs    /mnt/logs      ext4    defaults      0      2  
# Logical volume vol03_homes  
/dev/mapper/homes   /mnt/homes     ext4    defaults      0      2
```

Now **reboot (systemctl reboot)** and you will be prompted to enter the **passphrase** for each **LV**. Afterwards you can confirm that the mount operation was successful by checking the corresponding mount points:

```
Please enter passphrase for disk tecmint_vg-vol01_docs (docs) on /mnt/docs!:  
*****_  
  
Please enter passphrase for disk tecmint_vg-vol02_logs (logs) on /mnt/logs!:  
*****_  
  
Please enter passphrase for disk tecmint_vg-vol03_homes (homes) on /mnt/homes!:  
*****_*  
  
Red Hat Enterprise Linux  
Kernel 3.10.0-229.1.2.el7.x86_64 on an x86_64  
  
Hint: Num Lock on  
  
rhel7 login: root  
Password:  
Last login: Sun Apr  5 00:53:14 on tty1  
[root@rhel7 ~]# mount | grep mnt  
/dev/mapper/docs on /mnt/docs type ext4 (rw,relatime,seclabel,data=ordered)  
/dev/mapper/logs on /mnt/logs type ext4 (rw,relatime,seclabel,data=ordered)  
/dev/mapper/homes on /mnt/homes type ext4 (rw,relatime,seclabel,data=ordered)  
[root@rhel7 ~]# _
```

Verify Logical Volume Mount Points

## Conclusion

In this tutorial we have started to explore how to set up and configure system storage using classic volume management tools and **SSM**, which also integrates filesystem and encryption capabilities in one package. This makes SSM an invaluable tool for any sysadmin.

Let us know if you have any questions or comments – feel free to use the form below to get in touch with us!

## RHCSA Series: Using ACLs (Access Control Lists) and Mounting Samba / NFS Shares – Part 7

In the last article ([RHCSA series Part 6](#)) we started explaining how to set up and configure local system storage using **parted** and **ssm**.

**RHCSA: Red Hat Certified System Administrator**

<http://www.tecmint.com>

**RED HAT ENTERPRISE LINUX 7**



**redhat.**<sup>®</sup>

**Part 7: Using ACLs and mounting Samba / NFS shares in RHEL 7**

RHCSA Series:: Configure ACL's and Mounting NFS / Samba Shares – Part 7

We also discussed how to create and mount encrypted volumes with a password during system boot. In addition, we warned you to avoid performing critical storage management operations on mounted filesystems. With that in mind we will now review the most used file system formats in **Red Hat Enterprise Linux 7** and then proceed to cover the topics of mounting, using, and **unmounting** both manually and automatically network filesystems (**CIFS** and **NFS**), along with the implementation of access control lists for your system.

### Prerequisites

Before proceeding further, please make sure you have a **Samba** server and a **NFS** server available (note that **NFSv2** is no longer supported in **RHEL 7**).

During this guide we will use a machine with **IP 192.168.0.10** with both services running in it as server, and a **RHEL 7** box as client with IP address **192.168.0.18**. Later in the article we will tell you which packages you need to install on the client.

### File System Formats in RHEL 7

Beginning with **RHEL 7**, **XFS** has been introduced as the default file system for all architectures due to its high performance and scalability. It currently supports a maximum filesystem size of **500 TB** as per the latest tests performed by Red Hat and its partners for mainstream hardware.

Also, XFS enables **user\_xattr** (extended user attributes) and **acl** (POSIX access control lists) as default mount options, unlike ext3 or ext4 (ext2 is considered deprecated as of RHEL 7), which means that you don't need to specify those options explicitly either on the command line or in **/etc/fstab** when mounting a XFS filesystem (if you want to disable such options in this last case, you have to explicitly use **no\_acl** and **no\_user\_xattr**).

Keep in mind that the extended user attributes can be assigned to files and directories for storing arbitrary additional information such as the mime type, character set or encoding of a file, whereas the access permissions for user attributes are defined by the regular file permission bits.

## Access Control Lists

As every system administrator, either beginner or expert, is well acquainted with regular access permissions on files and directories, which specify certain privileges (**read**, **write**, and **execute**) for the owner, the group, and “the world” (all others). However, feel free to refer to [Part 3 of the RHCSA series](#) if you need to refresh your memory a little bit.

However, since the standard **ugo/rwx** set does not allow to configure different permissions for different users, **ACLs** were introduced in order to define more detailed access rights for files and directories than those specified by regular permissions.

In fact, **ACL-defined** permissions are a superset of the permissions specified by the file permission bits. Let's see how all of this translates is applied in the real world.

1. There are two types of **ACLs**: **access ACLs**, which can be applied to either a specific file or a directory), and **default ACLs**, which can only be applied to a directory. If files contained therein do not have a ACL set, they inherit the default ACL of their parent directory.
2. To begin, ACLs can be configured per user, per group, or per an user not in the owning group of a file.
3. **ACLs** are set (and removed) using **setfacl**, with either the **-m** or **-x** options, respectively.

For example, let us create a group named **tecmint** and add users **johndoe** and **davenull** to it:

```
# groupadd tecmint
# useradd johndoe
# useradd davenull
# usermod -a -G tecmint johndoe
# usermod -a -G tecmint davenull
```

And let's verify that both users belong to supplementary group **tecmint**:

```
# id johndoe
# id davenull
[root@rhel7 ~]# id johndoe
uid=1002(johndoe) gid=1002(johndoe) groups=1002(johndoe),1001(tecmint)
[root@rhel7 ~]# id davenull
uid=1003(davenull) gid=1003(davenull) groups=1003(davenull),1001(tecmint)
[root@rhel7 ~]#
```

## Verify Users

Let's now create a directory called **playground** within **/mnt**, and a file named **testfile.txt** inside. We will set the group owner to **tecmint** and change its default ugo/rwx permissions to 770 (read, write, and execute permissions granted to both the owner and the group owner of the file):

```
# mkdir /mnt/playground  
# touch /mnt/playground/testfile.txt  
# chmod 770 /mnt/playground/testfile.txt
```

Then switch user to **johndoe** and **davenull**, in that order, and write to the file:

```
echo "My name is John Doe" > /mnt/playground/testfile.txt  
echo "My name is Dave Null" >> /mnt/playground/testfile.txt
```

So far so good. Now let's have user **gacanepa** write to the file – and the write operation will fail, which was to be expected.

But what if we actually need user **gacanepa** (who is not a member of group **tecmint**) to have write permissions on **/mnt/playground/testfile.txt**? The first thing that may come to your mind is adding that user account to group **tecmint**. But that will give him write permissions on **ALL** files were the write bit is set for the group, and we don't want that. We only want him to be able to write to **/mnt/playground/testfile.txt**.

```
# touch /mnt/playground/testfile.txt  
# chown :tecmint /mnt/playground/testfile.txt  
# chmod 777 /mnt/playground/testfile.txt  
# su johndoe  
$ echo "My name is John Doe" > /mnt/playground/testfile.txt  
$ su davenull  
$ echo "My name is Dave Null" >> /mnt/playground/testfile.txt  
$ su gacanepa  
$ echo "My name is Gabriel Canepa" >> /mnt/playground/testfile.txt
```

```
[root@rhel7 ~]# touch /mnt/playground/testfile.txt  
[root@rhel7 ~]# chown :tecmint /mnt/playground/testfile.txt  
[root@rhel7 ~]# chmod 770 /mnt/playground/testfile.txt  
[root@rhel7 ~]# su johndoe  
[johndoe@rhel7 root]$ echo "My name is John Doe" > /mnt/playground/testfile.txt  
[johndoe@rhel7 root]$ su davenull  
Password:  
[davenull@rhel7 root]$ echo "My name is Dave Null" >> /mnt/playground/testfile.txt  
[davenull@rhel7 root]$ su gacanepa  
Password:  
[gacanepa@rhel7 root]$ echo "My name is Gabriel Canepe" >> /mnt/playground/testfile.txt  
bash: /mnt/playground/testfile.txt: Permission denied  
[gacanepa@rhel7 root]$ █
```

<http://www.tecmint.com>

## Manage User Permissions

Let's give user **gacanepa** read and write access to **/mnt/playground/testfile.txt**.

Run as root,

```
# setfacl -R -m u:gacanepa:rwx /mnt/playground
```

and you'll have successfully added an **ACL** that allows **gacanepa** to write to the test file. Then switch to user **gacanepa** and try to write to the file again:

```
$ echo "My name is Gabriel Canepa" >> /mnt/playground/testfile.txt
```

To view the **ACLs** for a specific file or directory, use **getfacl**:

```
# getfacl /mnt/playground/testfile.txt
[root@rhel7 ~]# getfacl /mnt/playground/testfile.txt
getfacl: Removing leading '/' from absolute path names
# file: mnt/playground/testfile.txt
# owner: root
# group: tecmint
user::rwx
user:gacanepa:rwx
group::rwx
mask::rwx
other::---
[root@rhel7 ~]#
```

<http://www.tecmint.com>

### Check ACLs of Files

To set a **default ACL** to a directory (which its contents will inherit unless overwritten otherwise), add **d:** before the rule and specify a directory instead of a file name:

```
# setfacl -m d:o:r /mnt/playground
```

The ACL above will allow users not in the owner group to have read access to the future contents of the **/mnt/playground** directory. Note the difference in the output of **getfacl /mnt/playground** before and after the change:

The screenshot shows two terminal sessions side-by-side. The left session, labeled 'BEFORE', shows the original ACL for the /mnt/playground directory. The right session, labeled 'AFTER', shows the ACL after the default ACL was set using the command '# setfacl -m d:o:r /mnt/playground'. The 'AFTER' session's output includes three new entries under the 'default' section: 'default:user::rwx', 'default:group::rwx', and 'default:other::r--'.

```
[root@rhel7 ~]# getfacl /mnt/playground
getfacl: Removing leading '/' from absolute path names
# file: mnt/playground
# owner: tecmint
# group: tecmint
user::rwx
user:gacanepa:rwx
group::rwx
mask::rwx
other::---
[root@rhel7 ~]#
```

**BEFORE**

```
[root@rhel7 ~]# setfacl -m d:o:r /mnt/playground
[root@rhel7 ~]# getfacl /mnt/playground
getfacl: Removing leading '/' from absolute path names
# file: mnt/playground
# owner: tecmint
# group: tecmint
user::rwx
user:gacanepa:rwx
group::rwx
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:other::r--
[root@rhel7 ~]#
```

**AFTER**

<http://www.tecmint.com>

### Set Default ACL in Linux

[Chapter 20 in the official RHEL 7 Storage Administration Guide](#) provides more ACL examples, and I highly recommend you take a look at it and have it handy as reference.

## Mounting NFS Network Shares

To show the list of NFS shares available in your server, you can use the **showmount** command with the **-e** option, followed by the machine name or its IP address. This tool is included in the **nfs-utils** package:

```
# yum update && yum install nfs-utils
```

Then do:

```
# showmount -e 192.168.0.10
```

and you will get a list of the available NFS shares on **192.168.0.10**:

```
[root@rhel7 ~]# showmount -e 192.168.0.10
Export list for 192.168.0.10:
/NFS-SHARE/mydir 192.168.0.18
/NFS-SHARE 192.168.0.18
[root@rhel7 ~]# http://www.tecmint.com
```

### Check Available NFS Shares

To mount NFS network shares on the local client using the command line on demand, use the following syntax:

```
# mount -t nfs -o [options] remote_host:/remote/directory /local/directory
```

which, in our case, translates to:

```
# mount -t nfs 192.168.0.10:/NFS-SHARE /mnt/nfs
```

If you get the following error message: “**Job for rpc-statd.service failed. See “systemctl status rpc-statd.service” and “journalctl -xn” for details.**”, make sure the **rpcbind** service is enabled and started in your system first:

```
# systemctl enable rpcbind.socket
# systemctl restart rpcbind.service
```

and then **reboot**. That should do the trick and you will be able to mount your NFS share as explained earlier. If you need to mount the NFS share automatically on system boot, add a valid entry to the **/etc/fstab** file:

```
remote_host:/remote/directory /local/directory nfs options 0 0
```

The variables **remote\_host**, **/remote/directory**, **/local/directory**, and **options** (which is optional) are the same ones used when manually mounting an NFS share from the command line. As per our previous example:

```
192.168.0.10:/NFS-SHARE /mnt/nfs nfs defaults 0 0
```

### Mounting CIFS (Samba) Network Shares

**Samba** represents the tool of choice to make a network share available in a network with \*nix and Windows machines. To show the Samba shares that are available, use the **smbclient** command with the **-L** flag, followed by the machine name or its IP address. This tool is included in the **samba-client** package:

You will be prompted for root's password in the remote host:

```
# smbclient -L 192.168.0.10
[root@rhel7 ~]# smbclient -L 192.168.0.10
Enter root's password:
Domain=[REDACTED] OS=[Unix] Server=[Samba 3.6.6]

      Sharename          Type      Comment
      -----          ----      -----
      print$            Disk      Printer Drivers
      gacanepa          Disk      gacanepa's home
      IPC$              IPC       IPC Service (debian server)
      SamsungML1640Series Printer  Samsung ML-1640 Series
      PDF                Printer   PDF
      EPSON_Stylus_CX3900 Printer  EPSON Stylus CX3900
Domain=[REDACTED] OS=[Unix] Server=[Samba 3.6.6]

      Server           Comment
      -----
      Workgroup        Master
      -----
[REDACTED]
[root@rhel7 ~]#
```

<http://www.tecmint.com>

### Check Samba Shares

To mount Samba network shares on the local client you will need to install first the **cifs-utils** package:

```
# yum update && yum install cifs-utils
```

Then use the following syntax on the command line:

```
# mount -t cifs -o credentials=/path/to/credentials/file
//remote_host/samba_share /local/directory
```

which, in our case, translates to:

```
# mount -t cifs -o credentials=~/smbcredentials //192.168.0.10/gacanepa
/mnt/samba
```

where **smbcredentials**:

```
username=gacanepa
password=XXXXXX
```

is a hidden file inside root's home (**/root/**) with permissions set to **600**, so that no one else but the owner of the file can read or write to it.

Please note that the **samba\_share** is the name of the Samba share as returned by **smbclient -L remote\_host** as shown above.

Now, if you need the Samba share to be available automatically on system boot, add a valid entry to the **/etc/fstab** file as follows:

```
//remote_host:/samba_share /local/directory cifs options 0 0
```

The variables **remote\_host**, **/samba\_share**, **/local/directory**, and options (which is optional) are the same ones used when manually mounting a Samba share from the command line. Following the definitions given in our previous example:

```
//192.168.0.10/gacanepa /mnt/samba    cifs  
credentials=/root/smbcredentials,defaults 0 0
```

## Conclusion

In this article we have explained how to set up **ACLs** in Linux, and discussed how to mount **CIFS** and **NFS** network shares in a **RHEL 7** client.

I recommend you to practice these concepts and even mix them (go ahead and try to set ACLs in mounted network shares) until you feel comfortable. If you have questions or comments feel free to use the form below to contact us anytime. Also, feel free to share this article through your social networks.

## RHCSA Series: Securing SSH, Setting Hostname and Enabling Network Services – Part 8

As a system administrator you will often have to log on to remote systems to perform a variety of administration tasks using a terminal emulator. You will rarely sit in front of a real (physical) terminal, so you need to set up a way to log on remotely to the machines that you will be asked to manage.

In fact, that may be the last thing that you will have to do in front of a physical terminal. For security reasons, using **Telnet** for this purpose is not a good idea, as all traffic goes through the wire in unencrypted, plain text.

In addition, in this article we will also review how to configure network services to start automatically at boot and learn how to set up network and hostname resolution statically or dynamically.

### RHCSA: Red Hat Certified System Administrator

#### RED HAT ENTERPRISE LINUX 7

<http://www.tecmint.com>



#### Part 8: Securing SSH, setting hostname, and enabling network services in RHEL 7

RHCSA: Secure SSH and Enable Network Services – Part 8

### Installing and Securing SSH Communication

For you to be able to log on remotely to a **RHEL 7** box using **SSH**, you will have to install the **openssh**, **openssh-clients** and **openssh-servers** packages. The following command not only will install the remote login program, but also the secure file transfer tool, as well as the remote file copy utility:

```
# yum update && yum install openssh openssh-clients openssh-servers
```

Note that it's a good idea to install the server counterparts as you may want to use the same machine as both client and server at some point or another.

After installation, there is a couple of basic things that you need to take into account if you want to secure remote access to your SSH server. The following settings should be present in the `/etc/ssh/sshd_config` file.

1. Change the port where the sshd daemon will listen on from **22** (the default value) to a high port (**2000** or greater), but first make sure the chosen port is not being used.

For example, let's suppose you choose port **2500**. Use [netstat](#) in order to check whether the chosen port is being used or not:

```
# netstat -npltu | grep 2500
```

If **netstat** does not return anything, you can safely use port **2500** for sshd, and you should change the Port setting in the configuration file as follows:

```
Port 2500
```

2. Only allow **protocol 2**:

```
Protocol 2
```

3. Configure the authentication timeout to 2 minutes, do not allow root logins, and restrict to a minimum the list of users which are allowed to login via ssh:

```
LoginGraceTime 2m
PermitRootLogin no
AllowUsers gacanepa
```

4. If possible, use key-based instead of password authentication:

```
PasswordAuthentication no
RSAAuthentication yes
PubkeyAuthentication yes
```

This assumes that you have already created a key pair with your user name on your client machine and copied it to your server as explained here.

1. [Enable SSH Passwordless Login](#)

## Configuring Networking and Name Resolution

1. Every system administrator should be well acquainted with the following system-wide configuration files:

1. **/etc/hosts** is used to resolve names <---> IPs in small networks.

Every line in the `/etc/hosts` file has the following structure:

```
IP address - Hostname - FQDN
```

For example,

```
192.168.0.10    laptop  laptop.gabrielcanepa.com.ar
```

**2.** `/etc/resolv.conf` specifies the IP addresses of DNS servers and the search domain, which is used for completing a given query name to a fully qualified domain name when no domain suffix is supplied.

Under normal circumstances, you don't need to edit this file as it is managed by the system. However, should you want to change DNS servers, be advised that you need to stick to the following structure in each line:

```
nameserver - IP address
```

For example,

```
nameserver 8.8.8.8
```

**3.** `3. /etc/host.conf` specifies the methods and the order by which hostnames are resolved within a network. In other words, tells the name resolver which services to use, and in what order.

Although this file has several options, the most common and basic setup includes a line as follows:

```
order bind,hosts
```

Which indicates that the resolver should first look in the nameservers specified in `resolv.conf` and then to the `/etc/hosts` file for name resolution.

**4.** `4. /etc/sysconfig/network` contains routing and global host information for all network interfaces. The following values may be used:

```
NETWORKING=yes|no  
HOSTNAME=value
```

Where `value` should be the Fully Qualified Domain Name (FQDN).

```
GATEWAY=XXX.XXX.XXX.XXX
```

Where **XXX.XXX.XXX.XXX** is the IP address of the network's gateway.

```
GATEWAYDEV=value
```

In a machine with multiple NICs, `value` is the gateway device, such as **enp0s3**.

**5.** Files inside `/etc/sysconfig/network-scripts` (network adapters configuration files).

Inside the directory mentioned previously, you will find several plain text files named.

```
ifcfg-name
```

Where `name` is the name of the NIC as returned by **ip link show**:

```
[root@rhel7 ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 08:00:27:4e:59:37 brd ff:ff:ff:ff:ff:ff
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Check Network Link Status

For example:

```
[root@rhel7 ~]# ls /etc/sysconfig/network-scripts/
ifcfg-enp0s3  ifdown-eth  ifdown-post  ifdown-Team
ifcfg-lo       ifdown-ippp  ifdown-ppp   ifdown-Team
ifdown         ifdown-ipv6  ifdown-routes  ifdown-tun
ifdown-bnep    ifdown-isdn  ifdown-sit    ifup
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Network Files

Other than for the **loopback** interface, you can expect a similar configuration for your NICs. Note that some variables, if set, will override those present in `/etc/sysconfig/network` for this particular interface. Each line is commented for clarification in this article but in the actual file you should avoid comments:

```
HWADDR=08:00:27:4E:59:37 # The MAC address of the NIC
TYPE=Ethernet # Type of connection
BOOTPROTO=static # This indicates that this NIC has been assigned a static
IP. If this variable was set to dhcp, the NIC will be assigned an IP
address by a DHCP server and thus the next two lines should not be present
in that case.
IPADDR=192.168.0.18
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
NM_CONTROLLED=no # Should be added to the Ethernet interface to prevent
NetworkManager from changing the file.
NAME=enp0s3
UUID=14033805-98ef-4049-bc7b-d4bea76ed2eb
ONBOOT=yes # The operating system should bring up this NIC during boot
```

## Setting Hostnames

In **Red Hat Enterprise Linux 7**, the **hostnamectl** command is used to both query and set the system's hostname.

To display the current hostname, type:

```
# hostnamectl status
```

```
[root@rhel7 ~]# hostnamectl status
  Static hostname: rhel7
    Icon name: computer
    Chassis: n/a
  Machine ID: 817a846b23d34dca90b4c8bea548570f
    Boot ID: 32300e1241074873b03f42b7d67aa6aa
  Virtualization: oracle
Operating System: Red Hat Enterprise Linux
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
    Kernel: Linux 3.10.0-229.1.2.el7.x86_64
  Architecture: x86_64
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Check System Hostname

To change the hostname, use

```
# hostnamectl set-hostname [new hostname]
```

For example,

```
# hostnamectl set-hostname cinderella
```

For the changes to take effect you will need to restart the **hostnamed** daemon (that way you will not have to log off and on again in order to apply the change):

```
# systemctl restart systemd-hostnamed
[root@rhel7 ~]# hostnamectl status
  Static hostname: rhel7
    Icon name: computer
    Chassis: n/a
  Machine ID: 817a846b23d34dca90b4c8bea548570f
    Boot ID: 32300e1241074873b03f42b7d67aa6aa
  Virtualization: oracle
Operating System: Red Hat Enterprise Linux
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
    Kernel: Linux 3.10.0-229.1.2.el7.x86_64
  Architecture: x86_64
[root@rhel7 ~]# hostnamectl set-hostname cinderella
[root@rhel7 ~]# systemctl restart systemd-hostnamed
[root@rhel7 ~]# hostnamectl status
  Static hostname: cinderella
    Icon name: computer
    Chassis: n/a
  Machine ID: 817a846b23d34dca90b4c8bea548570f
    Boot ID: 32300e1241074873b03f42b7d67aa6aa
  Virtualization: oracle
Operating System: Red Hat Enterprise Linux
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
    Kernel: Linux 3.10.0-229.1.2.el7.x86_64
  Architecture: x86_64
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Set System Hostname

In addition, **RHEL 7** also includes the **nmcli** utility that can be used for the same purpose. To display the hostname, run:

```
# nmcli general hostname
```

and to change it:

```
# nmcli general hostname [new hostname]
```

For example,

```
# nmcli general hostname rhel7
[root@rhel7 ~]# nmcli general hostname
cinderella
[root@rhel7 ~]# nmcli general hostname rhel7
[root@rhel7 ~]# nmcli general hostname
rhel7
[root@rhel7 ~]# http://www.tecmint.com
```

## Set Hostname Using nmcli Command

### Starting Network Services on Boot

To wrap up, let us see how we can ensure that network services are started automatically on boot. In simple terms, this is done by creating symlinks to certain files specified in the **[Install]** section of the service configuration files.

In the case of **firewalld** (**/usr/lib/systemd/system/firewalld.service**):

```
[Install]
WantedBy=basic.target
Alias=dbus-org.fedoraproject.FirewallD1.service
```

To enable the service:

```
# systemctl enable firewalld
```

On the other hand, disabling firewalld entails removing the symlinks:

```
# systemctl disable firewalld
[root@rhel7 ~]# systemctl enable firewalld
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/system/basic.target.wants/firewalld.service'
[root@rhel7 ~]# systemctl disable firewalld
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
[root@rhel7 ~]# http://www.tecmint.com
```

## Enable Service at System Boot

## Conclusion

In this article we have summarized how to install and secure connections via **SSH** to a **RHEL** server, how to change its name, and finally how to ensure that network services are started on boot. If you notice that a certain service has failed to start properly, you can use **systemctl status -l [service]** and **journald -xn** to troubleshoot it.

Feel free to let us know what you think about this article using the comment form below. Questions are also welcome. We look forward to hearing from you!

## RHCSA Series: Installing, Configuring and Securing a Web and FTP Server – Part 9

A web server (also known as a **HTTP** server) is a service that handles content (most commonly web pages, but other types of documents as well) over to a client in a network.

A FTP server is one of the oldest and most commonly used resources (even to this day) to make files available to clients on a network in cases where no authentication is necessary since FTP uses **username** and **password** without encryption.

The web server available in **RHEL 7** is version **2.4** of the Apache HTTP Server. As for the FTP server, we will use the Very Secure Ftp Daemon (aka **vsftpd**) to establish connections secured by TLS.

## RHCSA: Red Hat Certified System Administrator



### Part 9: Installing, configuring, and securing a Web and FTP server in RHEL 7

#### RHCSA: Installing, Configuring and Securing Apache and FTP – Part 9

In this article we will explain how to install, configure, and secure a web server and a FTP server in RHEL 7.

#### Installing Apache and FTP Server

In this guide we will use a RHEL 7 server with a static IP address of **192.168.0.18/24**. To install Apache and VSFTPD, run the following command:

```
# yum update && yum install httpd vsftpd
```

When the installation completes, both services will be disabled initially, so we need to start them manually for the time being and enable them to start automatically beginning with the next boot:

```
# systemctl start httpd
# systemctl enable httpd
# systemctl start vsftpd
# systemctl enable vsftpd
```

In addition, we have to open ports **80** and **21**, where the web and ftp daemons are listening, respectively, in order to allow access to those services from the outside:

```
# firewall-cmd --zone=public --add-port=80/tcp --permanent
# firewall-cmd --zone=public --add-service=ftp --permanent
# firewall-cmd --reload
```

To confirm that the web server is working properly, fire up your browser and enter the IP of the server. You should see the test page:



## Confirm Apache Web Server

As for the ftp server, we will have to configure it further, which we will do in a minute, before confirming that it's working as expected.

## Configuring and Securing Apache Web Server

The main configuration file for **Apache** is located in `/etc/httpd/conf/httpd.conf`, but it may rely on other files present inside `/etc/httpd/conf.d`.

Although the default configuration should be sufficient for most cases, it's a good idea to become familiar with all the available options as described in the [official documentation](#).

As always, make a backup copy of the main configuration file before editing it:

```
# cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.$(date +%Y%m%d)
```

Then open it with your preferred text editor and look for the following variables:

1. **ServerRoot**: the directory where the server's configuration, error, and log files are kept.
2. **Listen**: instructs Apache to listen on specific IP address and / or ports.
3. **Include**: allows the inclusion of other configuration files, which must exist. Otherwise, the server will fail, as opposed to the **IncludeOptional** directive, which is silently ignored if the specified configuration files do not exist.
4. **User and Group**: the name of the user/group to run the httpd service as.
5. **DocumentRoot**: The directory out of which Apache will serve your documents. By default, all requests are taken from this directory, but symbolic links and aliases may be used to point to other locations.
6. **ServerName**: this directive sets the hostname (or IP address) and port that the server uses to identify itself.

The first security measure will consist of creating a dedicated user and group (i.e. **tecmint/tecmint**) to run the web server as and changing the default port to a higher one (**9000** in this case):

```
ServerRoot "/etc/httpd"
Listen 192.168.0.18:9000
User tecmint
Group tecmint
DocumentRoot "/var/www/html"
ServerName 192.168.0.18:9000
```

You can test the configuration file with.

```
# apachectl configtest
```

and if everything is **OK**, then restart the web server.

```
# systemctl restart httpd
```

and don't forget to enable the new port (and disable the old one) in the firewall:

```
# firewall-cmd --zone=public --remove-port=80/tcp --permanent
# firewall-cmd --zone=public --add-port=9000/tcp --permanent
# firewall-cmd --reload
```

Note that, due to **SELinux** policies, you can only use the ports returned by

```
# semanage port -l | grep -w '^http_port_t'
```

for the web server.

If you want to use another port (i.e. TCP port **8100**), you will have to add it to **SELinux** port context for the **httpd** service:

```
# semanage port -a -t http_port_t -p tcp 8100
```

```
[root@rhel7 ~]# semanage port -l | grep -w '^http_port_t'
http_port_t
[root@rhel7 ~]# semanage port -a -t http_port_t -p tcp 8100
[root@rhel7 ~]# semanage port -l | grep -w '^http_port_t'
http_port_t
[root@rhel7 ~]# http://www.tecmint.com
```

## Add Apache Port to SELinux Policies

To further secure your Apache installation, follow these steps:

1. The user Apache is running as should not have access to a shell:

```
# usermod -s /sbin/nologin tecmint
```

2. Disable directory listing in order to prevent the browser from displaying the contents of a directory if there is no **index.html** present in that directory.

Edit `/etc/httpd/conf/httpd.conf` (and the configuration files for virtual hosts, if any) and make sure that the **Options** directive, both at the top and at Directory block levels, is set to **None**:

```
Options None
```

3. Hide information about the web server and the operating system in HTTP responses. Edit `/etc/httpd/conf/httpd.conf` as follows:

```
ServerTokens Prod
ServerSignature Off
```

Now you are ready to start serving content from your `/var/www/html` directory.

## Configuring and Securing FTP Server

As in the case of Apache, the main configuration file for **Vsftpd** (`/etc/vsftpd/vsftpd.conf`) is well commented and while the default configuration should suffice for most applications, you should become acquainted with the documentation and the man page (`man vsftpd.conf`) in order to operate the ftp server more efficiently (I can't emphasize that enough!).

In our case, these are the directives used:

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
chroot_local_user=YES
allow_writeable_chroot=YES
listen=NO
```

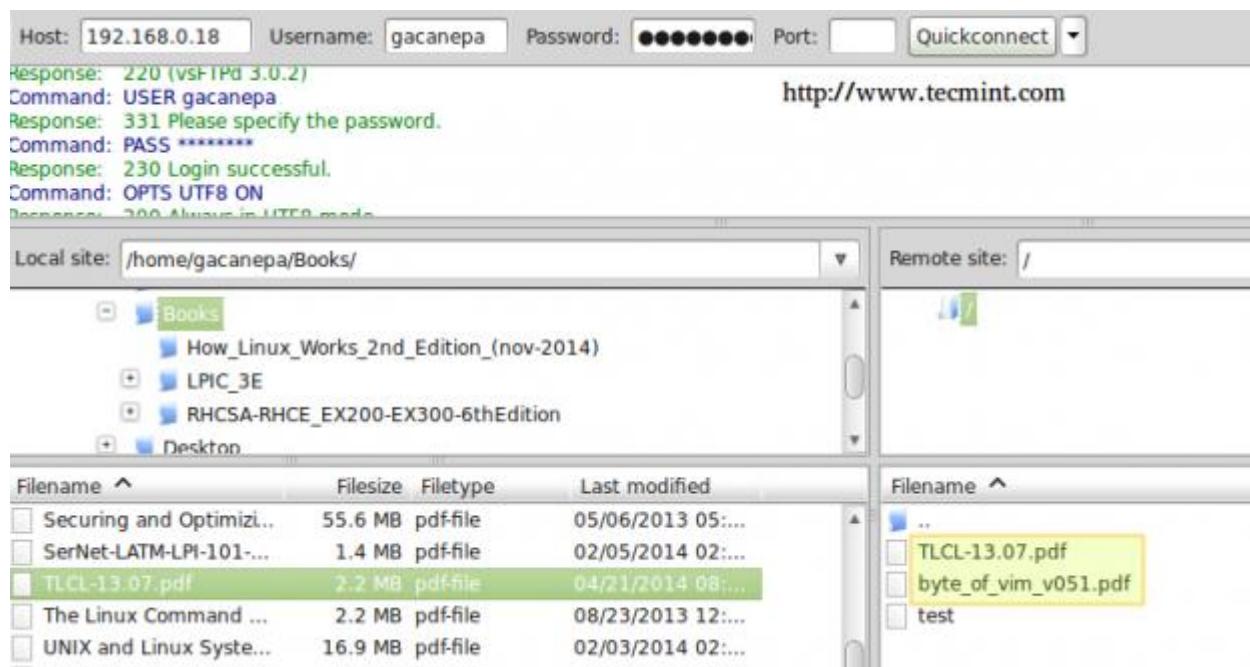
```
listen_ipv6=YES  
pam_service_name=vsftpd  
userlist_enable=YES  
tcp_wrappers=YES
```

By using `chroot_local_user=YES`, local users will be (by default) placed in a chroot'ed jail in their home directory right after login. This means that local users will not be able to access any files outside their corresponding home directories.

Finally, to allow ftp to read files in the user's home directory, set the following **SELinux** boolean:

```
# setsebool -P ftp_home_dir on
```

You can now connect to the ftp server using a client such as **Filezilla**:



### Check FTP Connection

Note that the `/var/log/xferlog` log records downloads and uploads, which concur with the above directory listing:

```
[root@rhel7 log]# tail -f xferlog  
Sat May 2 00:40:42 2015 1 ::ffff:192.168.0.103 0 /test b_o_r gacanepa ftp 0 * c  
Sat May 2 00:58:54 2015 1 ::ffff:192.168.0.103 1545847 /byte_of_vim_v051.pdf b_i_r gacanepa ftp 0 * c  
Sat May 2 00:59:11 2015 1 ::ffff:192.168.0.103 2119958 /TLCL-13.07.pdf b_i_r gacanepa ftp 0 * c
```

### Monitor FTP Download and Upload

**Read Also:** [Limit FTP Network Bandwidth Used by Applications in a Linux System with Trickle](#)

## Summary

In this tutorial we have explained how to set up a web and a ftp server. Due to the vastness of the subject, it is not possible to cover all the aspects of these topics (i.e. virtual web hosts). Thus, I recommend you also check other excellent articles in this website about [Apache](#).

## RHCSA Series: Yum Package Management, Automating Tasks with Cron and Monitoring System Logs – Part 10

In this article we will review how to install, update, and remove packages in **Red Hat Enterprise Linux 7**. We will also cover how to automate tasks using **cron**, and will finish this guide explaining how to locate and interpret system logs files with the focus of teaching you why all of these are essential skills for every system administrator.

### RHCSA: Red Hat Certified System Administrator



### Part 10: Managing packages, automating tasks, and inspecting system logs in RHEL 7

RHCSA: Yum Package Management, Cron Job Scheduling and Log Monitoring – Part 10

#### Managing Packages Via Yum

To install a package along with all its dependencies that are not already installed, you will use:

```
# yum -y install package_name(s)
```

Where **package\_name(s)** represent at least one real package name.

For example, to install **httpd** and **mlocate** (in that order), type.

```
# yum -y install httpd mlocate
```

**Note:** That the letter **y** in the example above bypasses the confirmation prompts that yum presents before performing the actual download and installation of the requested programs. You can leave it out if you want.

By default, **yum** will install the package with the architecture that matches the OS architecture, unless overridden by appending the package architecture to its name.

For example, on a **64 bit** system, **yum install package** will install the **x86\_64** version of package, whereas **yum install package.x86** (if available) will install the **32-bit** one.

There will be times when you want to install a package but don't know its exact name. The **search all** or **search** options can search the currently enabled repositories for a certain keyword in the package name and/or in its description as well, respectively.

For example,

```
# yum search log
```

will search the installed repositories for packages with the word **log** in their names and summaries, whereas

```
# yum search all log
```

will look for the same keyword in the package description and **url** fields as well.

Once the search returns a package listing, you may want to display further information about some of them before installing. That is when the **info** option will come in handy:

```
# yum info logwatch
```

```
[root@rhel7 ~]# yum search log
Loaded plugins: product-id, subscription-manager
=====
apache-commons-logging.noarch : Apache Commons Logging
avalon-logkit.noarch : Java logging toolkit
dialog.i686 : A utility for creating TTY dialog boxes
dialog.x86_64 : A utility for creating TTY dialog boxes
gnome-system-log.x86_64 : A log file viewer for GNOME
log4cxx.i686 : A port to C++ of the Log4j project
log4cxx.x86_64 : A port to C++ of the Log4j project
log4j.noarch : Java logging package
logrotate.x86_64 : Rotates, compresses, removes and mails system log files
logwatch.noarch : A log file analysis program
compmii-logicalfile-doc.noarch : CIM logicalfile provider documentation
```

```
[root@rhel7 ~]# yum info logwatch
Loaded plugins: product-id, subscription-manager
Available Packages
Name        : logwatch
Arch       : noarch
Version    : 7.4.0
Release   : 28.20130522svn140.el7
Size      : 400 k
Repo      : rhel-7-server-rpms/7Server/x86_64
Summary   : A log file analysis program
URL       : http://www.logwatch.org/
License   : MIT
Description: Logwatch is a customizable, pluggable program that monitors your logs for a given period of time and generates reports that you wish with the detail that you desire. It can be run on many systems.
[root@rhel7 ~]# http://www.tecmint.com
```

## Search Package Information

You can regularly check for updates with the following command:

```
# yum check-update
```

The above command will return all the installed packages for which an update is available. In the example shown in the image below, only **rhel-7-server-rpms** has an update available:

```
[root@rhel7 ~]# yum check-update
Loaded plugins: product-id, subscription-manager
rhel-7-server-rpms
[root@rhel7 ~]# http://www.tecmint.com
```

## Check For Package Updates

You can then update that package alone with,

```
# yum update rhel-7-server-rpms
```

If there are several packages that can be updated, **yum update** will update all of them at once.

Now what happens when you know the name of an executable, such as **ps2pdf**, but don't know which package provides it? You can find out with **yum whatprovides "\*/[executable]"**:

```
# yum whatprovides "*/ps2pdf"
[root@rhel7 ~]# yum whatprovides "*/ps2pdf"
Loaded plugins: product-id, subscription-manager
ghostscript-9.07-16.el7.i686 : A PostScript interpreter a
Repo      : rhel-7-server-rpms
Matched from:
Filename   : /usr/bin/ps2pdf

ghostscript-9.07-16.el7.x86_64 : A PostScript interpreter
Repo      : rhel-7-server-rpms
Matched from:
Filename   : /usr/bin/ps2pdf http://www.tecmint.com
```

## Find Package Belongs to Which Package

Now, when it comes to removing a package, you can do so with **yum remove package**. Easy, huh? This goes to show that yum is a complete and powerful package manager.

```
# yum remove httpd
```

**Read Also:** [20 Yum Commands to Manage RHEL 7 Package Management](#)

## Good Old Plain RPM

**RPM** (aka **RPM Package Manager**, or originally **RedHat Package Manager**) can also be used to install or update packages when they come in form of standalone .rpm packages.

It is often utilized with the **-Uvh** flags to indicate that it should install the package if it's not already present or attempt to update it if it's installed (**-U**), producing a verbose output (**-v**) and a progress bar with hash marks (**-h**) while the operation is being performed. For example,

```
# rpm -Uvh package.rpm
```

Another typical use of **rpm** is to produce a list of currently installed packages with code>**rpm -qa** (short for query all):

```
# rpm -qa
[root@rhel7 ~]# rpm -qa
perl-Pod-Perldoc-3.20-4.el7.noarch
jansson-2.4-6.el7.x86_64
filesystem-3.2-18.el7.x86_64
perl-Pod-Usage-1.63-3.el7.noarch
xdg-utils-1.1.0-0.16.20120809git.el7.noarch
perl-Time-Local-1.2300-2.el7.noarch
perl-PathTools-3.40-5.el7.x86_64
perl-File-Temp-0.23.01-3.el7.noarch
perl-Filter-1.49-3.el7.x86_64
popt-1.13-16.el7.x86_64
vim-filesystem-7.4.160-1.el7.x86_64
systemd-libs-208-20.el7_1.2.x86_64
openssl-libs-1.0.1e-42.el7_1.4.x86_64
kernel-tools-3.10.0-329.1.2.el7.x86_64
http://www.tecmint.com
```

## Query All RPM Packages

**Read Also:** [20 RPM Commands to Install Packages in RHEL 7](#)

## Scheduling Tasks using Cron

Linux and other Unix-like operating systems include a tool called **cron** that allows you to schedule tasks (i.e. commands or shell scripts) to run on a periodic basis. Cron checks every minute the **/var/spool/cron** directory for files which are named after accounts in **/etc/passwd**.

When executing commands, any output is mailed to the owner of the **crontab** (or to the user specified in the **MAILTO** environment variable in the **/etc/crontab**, if it exists).

Crontab files (which are created by typing **crontab -e** and pressing **Enter**) have the following format:

**\* \* \* \* \* /absolute/path/to/command/or/script**



## Crontab Entries

Thus, if we want to update the local file database (which is used by locate to find files by name or pattern) every second day of the month at **2:15 am**, we need to add the following **crontab** entry:

```
15 02 2 * * /bin/updatedb
```

The above crontab entry reads, “**Run /bin/updatedb on the second day of the month, every month of the year, regardless of the day of the week, at 2:15 am**”. As I’m sure you already guessed, the star symbol is used as a wildcard character.

After adding a cron job, you can see that a file named root was added inside **/var/spool/cron**, as we mentioned earlier. That file lists all the tasks that the **crond** daemon should run:

```
# ls -l /var/spool/cron
[root@rhel7 ~]# ls -l /var/spool/cron
total 4
-rw-----. 1 root root 26 May  4 21:53 root
[root@rhel7 ~]# cat /var/spool/cron/root
15 02 2 * * /bin/updatedb
[root@rhel7 ~]# http://www.tecmint.com
```

#### Check All Cron Jobs

In the above image, the current user’s crontab can be displayed either using **cat /var/spool/cron/root** or,

```
# crontab -l
```

If you need to run a task on a more fine-grained basis (for example, twice a day or three times each month), cron can also help you to do that.

For example, to run **/my/script** on the 1st and **15th** of each month and send any output to **/dev/null**, you can add two **crontab** entries as follows:

```
01 00 1 * * /myscript > /dev/null 2>&1
01 00 15 * * /my/script > /dev/null 2>&1
```

But in order for the task to be easier to maintain, you can combine both entries into one:

```
01 00 1,15 * * /my/script > /dev/null 2>&1
```

Following the previous example, we can run **/my/other/script** at **1:30 am** on the first day of the month every three months:

```
30 01 1 1,4,7,10 * /my/other/script > /dev/null 2>&1
```

But when you have to repeat a certain task every “**x**” minutes, hours, days, or months, you can divide the right position by the desired frequency. The following **crontab** entry has the exact same meaning as the previous one:

```
30 01 1 */3 * /my/other/script > /dev/null 2>&1
```

Or perhaps you need to run a certain job on a fixed frequency or after the system boots, for example. You can use one of the following string instead of the five fields to indicate the exact time when you want your job to run:

```
@reboot      Run when the system boots.  
@yearly      Run once a year, same as 00 00 1 1 *.  
@monthly     Run once a month, same as 00 00 1 * *.  
@weekly      Run once a week, same as 00 00 * * 0.  
@daily       Run once a day, same as 00 00 * * *.  
@hourly      Run once an hour, same as 00 * * * *.
```

**Read Also:** [11 Commands to Schedule Cron Jobs in RHEL 7](#)

## Locating and Checking Logs

System logs are located (and rotated) inside the **/var/log** directory. According to the Linux Filesystem Hierarchy Standard, this directory contains miscellaneous log files, which are written to it or an appropriate subdirectory (such as **audit**, **httpd**, or **samba** in the image below) by the corresponding daemons during system operation:

```
# ls /var/log  
[root@rhel7 ~]# ls /var/log  
anaconda    cron        dmesg      lastlog      maillog-20150504  messages-201  
audit        cron-20150418  dmesg.old   maillog      messages  
boot.log     cron-20150420  firewalld   maillog-20150418  messages-20150418  ppp  
btmp        cron-20150428  grubby     maillog-20150420  messages-20150420  rhsn  
bttmp-20150501 cron-20150504  httpd      maillog-20150428  messages-20150428  samba  
[root@rhel7 ~]# █  
http://www.tecmint.com
```

### Linux Log Files Location

Other interesting logs are **dmesg** (contains all messages from kernel ring buffer), **secure** (logs connection attempts that require user authentication), **messages** (system-wide messages) and **wtmp** (records of all user logins and logouts).

Logs are very important in that they allow you to have a glimpse of what is going on at all times in your system, and what has happened in the past. They represent a priceless tool to troubleshoot and monitor a Linux server, and thus are often used with the **tail -f** command to display events, in real time, as they happen and are recorded in a log.

For example, if you want to display **kernel-related** events, type the following command:

```
# tail -f /var/log/dmesg
```

Same if you want to view access to your web server:

```
# tail -f /var/log/httpd/access.log
```

## Summary

If you know how to efficiently manage packages, schedule tasks, and where to look for information about the current and past operation of your system you can rest assure that you

will not run into surprises very often. I hope this article has helped you learn or refresh your knowledge about these basic skills.

Don't hesitate to drop us a line using the contact form below if you have any questions or comments.

## RHCSA Series: Firewall Essentials and Network Traffic Control Using FirewallD and Iptables – Part 11

In simple words, a **firewall** is a security system that controls the incoming and outgoing traffic in a network based on a set of predefined rules (such as the packet destination / source or type of traffic, for example).

### RHCSA: Red Hat Certified System Administrator

### RED HAT ENTERPRISE LINUX 7



### Firewall Essentials to Control Network Traffic Using FirewallD and Iptables

#### RHCSA: Control Network Traffic with FirewallD and Iptables – Part 11

In this article we will review the basics of **firewalld**, the default dynamic firewall daemon in **Red Hat Enterprise Linux 7**, and **iptables** service, the legacy firewall service for Linux, with which most system and network administrators are well acquainted, and which is also available in **RHEL 7**.

#### A Comparison Between FirewallD and Iptables

Under the hood, both **firewalld** and the **iptables** service talk to the **netfilter** framework in the kernel through the same interface, not surprisingly, the **iptables** command. However, as opposed to the **iptables** service, **firewalld** can change the settings during normal system operation without existing connections being lost.

**Firewalld** should be installed by default in your RHEL system, though it may not be running. You can verify with the following commands (**firewall-config** is the user interface configuration tool):

```
# yum info firewalld firewall-config
```

```
[root@rhel7 ~]# yum info firewalld firewall-config
Loaded plugins: product-id, subscription-manager
Installed Packages
Name        : firewalld
Arch       : noarch
Version    : 0.3.9
Release   : 11.el7
Size      : 2.3 M
Repo      : installed
From repo : rhel-7-server-rpms
Summary   : A firewall daemon with D-BUS interface providing a dynamic firewall
URL       : http://fedorahosted.org/firewalld
License   : GPLv2+
Description: firewalld is a firewall service daemon that provides a dynamic customizable
             firewall with a D-BUS interface.

Available Packages
Name        : firewall-config
Arch       : noarch
Version    : 0.3.9
Release   : 11.el7
Size      : 103 k
Repo      : rhel-7-server-rpms/7Server/x86_64
Summary   : Firewall configuration application
URL       : http://fedorahosted.org/firewalld
License   : GPLv2+
Description: The firewall configuration application provides an configuration interface for
             firewalld.
http://www.tecmint.com
```

## Check FirewallD Information

and,

```
# systemctl status -l firewalld.service
[root@rhel7 ~]# systemctl status -l firewalld.service
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
     Active: active (running) since Fri 2015-05-08 19:03:04 EDT; 15min ago
       Main PID: 633 (firewalld)
          CGroup: /system.slice/firewalld.service
                     └─633 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

May 08 19:03:00 rhel7 systemd[1]: Starting firewalld - dynamic firewall daemon...
May 08 19:03:04 rhel7 systemd[1]: Started firewalld - dynamic firewall daemon.
[root@rhel7 ~]# http://www.tecmint.com
```

## Check FirewallD Status

On the other hand, the **iptables** service is not included by default, but can be installed through.

```
# yum update && yum install iptables-services
```

Both daemons can be started and enabled to start on boot with the usual **systemd** commands:

```
# systemctl start firewalld.service | iptables-service.service
# systemctl enable firewalld.service | iptables-service.service
```

## **Read Also:** [Useful Commands to Manage Systemd Services](#)

As for the configuration files, the iptables service uses `/etc/sysconfig/iptables` (which will not exist if the package is not installed in your system). On a RHEL 7 box used as a cluster node, this file looks as follows:

```
[root@node01 ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.21 on Tue Apr 28 18:41:14 2015
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [262:28166]
-A INPUT -p tcp -m state --state NEW -m tcp --dport 7790 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 7789 -j ACCEPT
-A INPUT -m addrtype --dst-type MULTICAST -j ACCEPT
-A INPUT -p igmp -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2224 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m multiport --dports 5404,5405 -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Tue Apr 28 18:41:14 2015
[root@node01 ~]# 
```

<http://www.tecmint.com>

## Iptables Firewall Configuration

Whereas firewalld store its configuration across two directories, `/usr/lib/firewalld` and `/etc/firewalld`:

```
# ls /usr/lib/firewalld /etc/firewalld
[root@rhel7 ~]# ls /usr/lib/firewalld/ /etc/firewalld/
/etc/firewalld/:
firewalld.conf  icmptypes  lockdown-whitelist.xml  services  zones

/usr/lib/firewalld/:
icmptypes  services  zones
[root@rhel7 ~]# 
```

<http://www.tecmint.com>

## FirewallD Configuration

We will examine these configuration files further later in this article, after we add a few rules here and there. By now it will suffice to remind you that you can always find more information about both tools with.

```
# man firewalld.conf
# man firewall-cmd
# man iptables
```

Other than that, remember to take a look at [Reviewing Essential Commands & System Documentation – Part 1](#) of the current series, where I described several sources where you can get information about the packages installed on your **RHEL 7** system.

## Using Iptables to Control Network Traffic

You may want to refer to [Configure Iptables Firewall – Part 8](#) of the **Linux Foundation Certified Engineer (LFCE)** series to refresh your memory about **iptables** internals before proceeding further. Thus, we will be able to jump in right into the examples.

[Example 1: Allowing both incoming and outgoing web traffic](#)

TCP ports **80** and **443** are the default ports used by the Apache web server to handle normal (**HTTP**) and secure (**HTTPS**) web traffic. You can allow incoming and outgoing web traffic through both ports on the **enp0s3** interface as follows:

```
# iptables -A INPUT -i enp0s3 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -o enp0s3 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
# iptables -A INPUT -i enp0s3 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -o enp0s3 -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
```

[Example 2: Block all \(or some\) incoming connections from a specific network](#)

There may be times when you need to block all (or some) type of traffic originating from a specific network, say **192.168.1.0/24** for example:

```
# iptables -I INPUT -s 192.168.1.0/24 -j DROP
```

will drop all packages coming from the **192.168.1.0/24** network, whereas,

```
# iptables -A INPUT -s 192.168.1.0/24 --dport 22 -j ACCEPT
```

will only allow incoming traffic through port **22**.

[Example 3: Redirect incoming traffic to another destination](#)

If you use your **RHEL 7** box not only as a software firewall, but also as the actual hardware-based one, so that it sits between two distinct networks, IP forwarding must have been already enabled in your system. If not, you need to edit `/etc/sysctl.conf` and set the value of `net.ipv4.ip_forward` to **1**, as follows:

```
net.ipv4.ip_forward = 1
```

then save the change, close your text editor and finally run the following command to apply the change:

```
# sysctl -p /etc/sysctl.conf
```

For example, you may have a printer installed at an internal box with **IP 192.168.0.10**, with the **CUPS** service listening on port **631** (both on the print server and on your firewall). In order to forward print requests from clients on the other side of the firewall, you should add the following iptables rule:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 631 -j DNAT --to 192.168.0.10:631
```

Please keep in mind that **iptables** reads its rules sequentially, so make sure the default policies or later rules do not override those outlined in the examples above.

## Getting Started with FirewallD

One of the changes introduced with **firewalld** are **zones**. This concept allows to separate networks into different zones level of trust the user has decided to place on the devices and traffic within that network.

To list the active zones:

```
# firewall-cmd --get-active-zones
```

In the example below, the **public zone** is active, and the **enp0s3** interface has been assigned to it automatically. To view all the information about a particular zone:

```
# firewall-cmd --zone=public --list-all
[root@rhel7 ~]# firewall-cmd --get-active-zones
public
  interfaces: enp0s3
[root@rhel7 ~]# firewall-cmd --zone=public --list-all
public (default, active)
  interfaces: enp0s3
  sources:
  services: dhcpcv6-client ftp ssh
  ports: 9000/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules: http://www.tecmint.com
```

### List all FirewallD Zones

Since you can read more about **zones** in the [RHEL 7 Security guide](#), we will only list some specific examples here.

#### Example 4: Allowing services through the firewall

To get a list of the supported services, use.

```
# firewall-cmd --get-services
```

```
[root@rhel7 ~]# firewall-cmd --get-services
RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpcv6 dhcpcv6-client dns
ftp high-availability http https imaps ipp ipp-client ipsec kerberos kpasswd lda
p ldaps libvirt libvirt-tls mdns mountd ms-wbt mysql nfs ntp openvpn pmcd pmproxy
pmwebapi pmwebapis pop3s postgresql proxy-dhcp radius rpc-bind samba samba-client
smtp ssh telnet tftp tftp-client transmission-client vnc-server wbem-https
[root@rhel7 ~]# [ ] http://www.tecmint.com
```

## List All Supported Services

To allow **http** and **https** web traffic through the firewall, effective immediately and on subsequent boots:

```
# firewall-cmd --zone=MyZone --add-service=http
# firewall-cmd --zone=MyZone --permanent --add-service=http
# firewall-cmd --zone=MyZone --add-service=https
# firewall-cmd --zone=MyZone --permanent --add-service=https
# firewall-cmd --reload
```

If code>—zone is omitted, the default zone (you can check with **firewall-cmd –get-default-zone**) is used.

To remove the rule, replace the word add with remove in the above commands.

## Example 5: IP / Port forwarding

First off, you need to find out if masquerading is enabled for the desired zone:

```
# firewall-cmd --zone=MyZone --query-masquerade
```

In the image below, we can see that **masquerading** is enabled for the **external zone**, but not for **public**:

```
[root@rhel7 ~]# firewall-cmd --zone=external --query-masquerade
yes
[root@rhel7 ~]# firewall-cmd --zone=public --query-masquerade
no [ ] http://www.tecmint.com
```

## Check Masquerading Status

You can either enable masquerading for public:

```
# firewall-cmd --zone=public --add-masquerade
```

or use masquerading in **external**. Here's what we would do to replicate **Example 3** with **firewalld**:

```
# firewall-cmd --zone=external --add-forward-
port=port=631:proto=tcp:toport=631:toaddr=192.168.0.10
```

And don't forget to reload the firewall.

You can find further examples on [Part 9](#) of the RHCSA series, where we explained how to allow or disable the ports that are usually used by a web server and a ftp server, and how to change the corresponding rule when the default port for those services are changed. In addition, you may want to refer to the firewalld wiki for further examples.

**Read Also:** [Useful FirewallD Examples to Configure Firewall in RHEL 7](#)

## Conclusion

In this article we have explained what a **firewall** is, what are the available services to implement one in **RHEL 7**, and provided a few examples that can help you get started with this task. If you have any comments, suggestions, or questions, feel free to let us know using the form below. Thank you in advance!

[RHCSA Series: Automate RHEL 7 Installations Using ‘Kickstart’ – Part 12](#)

Linux servers are rarely standalone boxes. Whether it is in a datacenter or in a lab environment, chances are that you have had to install several machines that will interact one with another in some way. If you multiply the time that it takes to install **Red Hat Enterprise Linux 7** manually on a single server by the number of boxes that you need to set up, this can lead to a rather lengthy effort that can be avoided through the use of an unattended installation tool known as **kickstart**.

In this article we will show what you need to use **kickstart** utility so that you can forget about babysitting servers during the installation process.

## RHCSA: Red Hat Certified System Administrator

RED HAT ENTERPRISE LINUX 7



**Part 12: Automating RHEL 7 installations using kickstart**

RHCSA: Automatic Kickstart Installation of RHEL 7

## Introducing Kickstart and Automated Installations

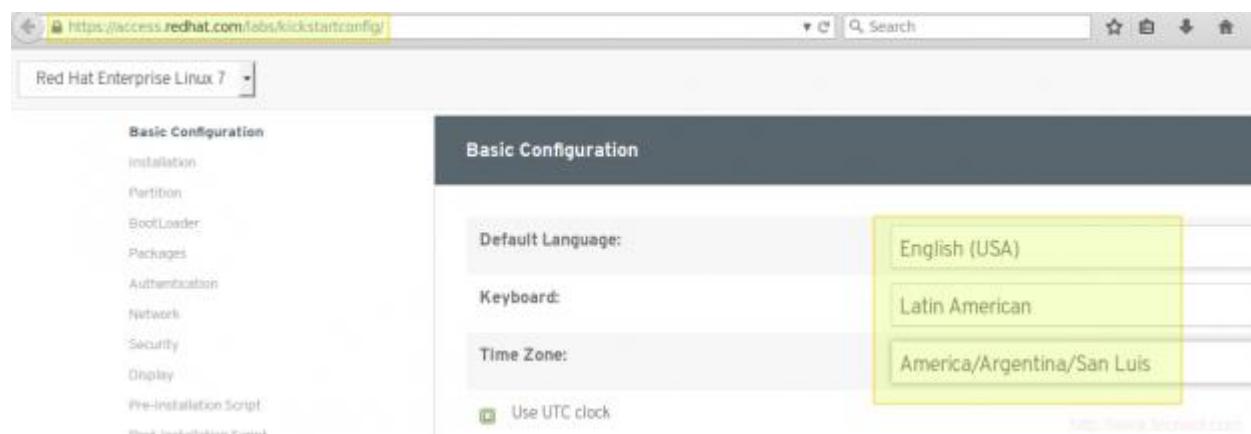
**Kickstart** is an automated installation method used primarily by Red Hat Enterprise Linux (and other Fedora spin-offs, such as CentOS, Oracle Linux, etc.) to execute unattended operating system installation and configuration. Thus, kickstart installations allow system administrators to have identical systems, as far as installed package groups and system configuration are concerned, while sparing them the hassle of having to manually install each of them.

## Preparing for a Kickstart Installation

To perform a kickstart installation, we need to follow these steps:

1. **Create a Kickstart file**, a plain text file with several predefined configuration options.
2. **Make the Kickstart file available on removable media, a hard drive or a network location**. The client will use the **rhel-server-7.0-x86\_64-boot.iso** file, whereas you will need to make the full ISO image (**rhel-server-7.0-x86\_64-dvd.iso**) available from a network resource, such as a HTTP or FTP server (in our present case, we will use another RHEL 7 box with IP **192.168.0.18**).
3. **Start the Kickstart installation**

To create a kickstart file, login to your **Red Hat Customer Portal** account, and use the [Kickstart configuration tool](#) to choose the desired installation options. Read each one of them carefully before scrolling down, and choose what best fits your needs:



Kickstart Configuration Tool

If you specify that the installation should be performed either through **HTTP**, **FTP**, or **NFS**, make sure the firewall on the server allows those services.

Although you can use the Red Hat online tool to create a kickstart file, you can also create it manually using the following lines as reference. You will notice, for example, that the

installation process will be in **English**, using the latin american keyboard layout and the America/Argentina/San\_Luis time zone:

```
lang en_US
keyboard la-latin1
timezone America/Argentina/San_Luis --isUtc
rootpw $1$5s0tDvRo$In4KTmX7OmcOW9HUvWtfn0 --iscrypted
#platform x86, AMD64, or Intel EM64T
text
url --url=http://192.168.0.18//kickstart/media
bootloader --location=mbr --append="rhgb quiet crashkernel=auto"
zerombr
clearpart --all --initlabel
autopart
auth --passalgo=sha512 --useshadow
selinux --enforcing
firewall --enabled
firstboot --disable
%packages
@base
@backup-server
@print-server
%end
```

In the online configuration tool, use **192.168.0.18** for HTTP Server and **/kickstart/tecmint.bin** for HTTP Directory in the Installation section after selecting HTTP as installation source. Finally, click the **Download** button at the right top corner to download the kickstart file.

In the kickstart sample file above, you need to pay careful attention to.

```
url --url=http://192.168.0.18//kickstart/media
```

That directory is where you need to extract the contents of the DVD or ISO installation media. Before doing that, we will mount the ISO installation file in **/media/rhel** as a loop device:

```
# mount -o loop /var/www/html/kickstart/rhel-server-7.0-x86_64-dvd.iso
/media/rhel
```

## Mount RHEL ISO Image

Next, copy all the contents of **/media/rhel** to **/var/www/html/kickstart/media**:

```
# cp -R /media/rhel /var/www/html/kickstart/media
```

When you're done, the directory listing and disk usage of **/var/www/html/kickstart/media** should look as follows:

## Kickstart Media Files

Now we're ready to kick off the kickstart installation.

Regardless of how you choose to create the kickstart file, it's always a good idea to check its syntax before proceeding with the installation. To do that, install the **pykickstart** package.

```
# yum update && yum install pykickstart
```

And then use the **ksvalidator** utility to check the file:

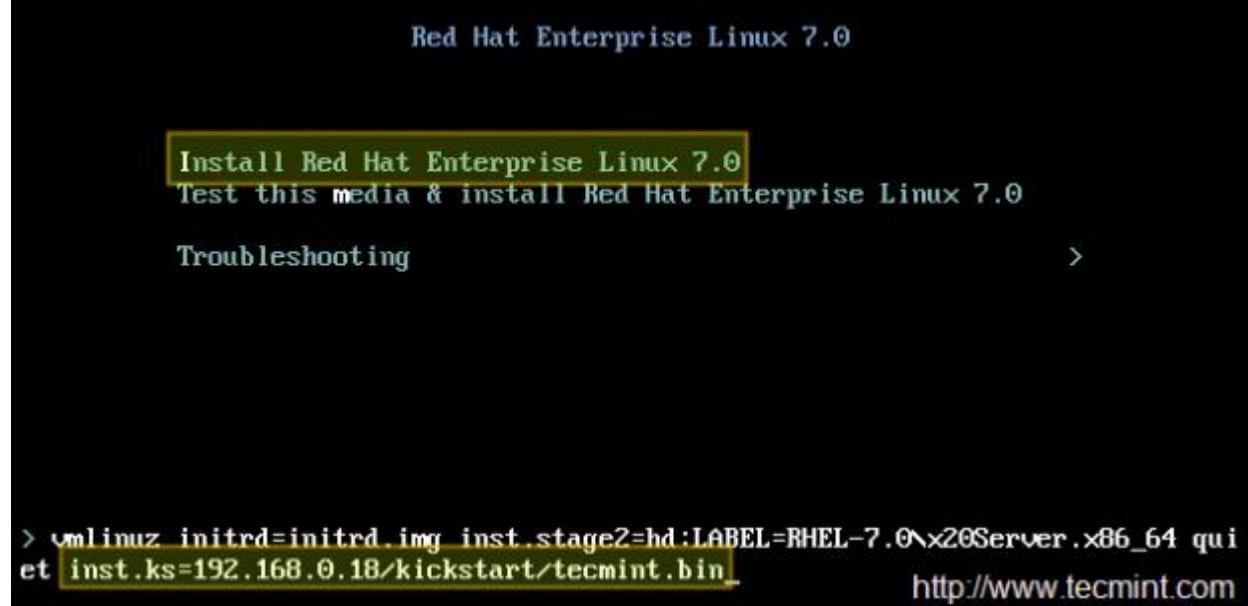
```
# ksvalidator /var/www/html/kickstart/tecmint.bin
```

If the syntax is correct, you will not get any output, whereas if there's an error in the file, you will get a warning notice indicating the line where the syntax is not correct or unknown.

## Performing a Kickstart Installation

To start, boot your client using the **rhel-server-7.0-x86\_64-boot.iso** file. When the initial screen appears, select **Install Red Hat Enterprise Linux 7.0** and press the **Tab** key to append the following stanza and press **Enter**:

```
# inst.ks=http://192.168.0.18/kickstart/tecmint.bin
```



RHEL Kickstart Installation

## RHEL Kickstart Installation

Where **tecmint.bin** is the kickstart file created earlier.

When you press **Enter**, the automated installation will begin, and you will see the list of packages that are being installed (the number and the names will differ depending on your choice of programs and package groups):

```
Installing iwl8000-firmware (578/591)
Installing iwl135-firmware (579/591)
Installing libertas-sd8686-firmware (580/591)
Installing iwl3945-firmware (581/591)
Installing man-pages (582/591)
Installing redhat-indexhtml (583/591)
Installing libertas-usb8388-firmware (584/591)
Installing iwl6000g2a-firmware (585/591)
Installing libertas-sd8787-firmware (586/591)
Installing iwl6000g2b-firmware (587/591)
Installing iwl2000-firmware (588/591)
Installing iw15150-firmware (589/591)
Installing man-pages-overrides (590/591)
Installing words (591/591)
Performing post-installation scriptlet http://www.tecmint.com
```

## Automatic Kickstart Installation of RHEL 7

When the automated process ends, you will be prompted to remove the installation media and then you will be able to boot into your newly installed system:



## RHEL 7 Boot Screen

Although you can create your kickstart files manually as we mentioned earlier, you should consider using the recommended approach whenever possible. You can either use the online configuration tool, or the **anaconda-ks.cfg** file that is created by the installation process in root's home directory.

This file actually is a kickstart file, so you may want to install the first box manually with all the desired options (maybe modify the logical volumes layout or the file system on top of each one) and then use the resulting **anaconda-ks.cfg** file to automate the installation of the rest.

In addition, using the online configuration tool or the **anaconda-ks.cfg** file to guide future installations will allow you to perform them using an encrypted root password out-of-the-box.

## Conclusion

Now that you know how to create kickstart files and how to use them to automate the installation of Red Hat Enterprise Linux 7 servers, you can forget about babysitting the installation process. This will give you time to do other things, or perhaps some leisure time if you're lucky.

Either way, let us know what you think about this article using the form below. Questions are also welcome!

**Read Also:** [Automated Installations of Multiple RHEL/CentOS 7 Distributions using PXE and Kickstart](#)

[RHCSA Series: Mandatory Access Control Essentials with SELinux in RHEL 7 – Part 13](#)

During this series we have explored in detail at least two access control methods: standard **ugo/rwx** permissions ([Manage Users and Groups – Part 3](#)) and access control lists ([Configure ACL's on File Systems – Part 7](#)).

## RHCSA: Red Hat Certified System Administrator

RED HAT ENTERPRISE LINUX 7



## Part 13: SELinux essentials and examples in RHEL 7

RHCSA Exam: SELinux Essentials and Control FileSystem Access

Although necessary as first level permissions and access control mechanisms, they have some limitations that are addressed by **Security Enhanced Linux** (aka **SELinux** for short).

One of such limitations is that a user can expose a file or directory to a security breach through a poorly elaborated **chmod** command and thus cause an unexpected propagation of access rights. As a result, any process started by that user can do as it pleases with the files

owned by the user, where finally a malicious or otherwise compromised software can achieve root-level access to the entire system.

With those limitations in mind, the **United States National Security Agency (NSA)** first devised **SELinux**, a flexible mandatory access control method, to restrict the ability of processes to access or perform other operations on system objects (such as files, directories, network ports, etc) to the least permission model, which can be modified later as needed. In few words, each element of the system is given only the access required to function.

In **RHEL 7**, **SELinux** is incorporated into the kernel itself and is enabled in **Enforcing** mode by default. In this article we will explain briefly the basic concepts associated with **SELinux** and its operation.

## SELinux Modes

SELinux can operate in three different ways:

1. **Enforcing**: SELinux denies access based on SELinux policy rules, a set of guidelines that control the security engine.
2. **Permissive**: SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode.
3. **Disabled** (self-explanatory).

The `getenforce` command displays the current mode of SELinux, whereas `setenforce` (followed by a **1** or a **0**) is used to change the mode to **Enforcing** or **Permissive**, respectively, during the current session only.

In order to achieve persistence across logouts and reboots, you will need to edit the `/etc/selinux/config` file and set the `SELINUX` variable to either **enforcing**, **permissive**, or **disabled**:

```
# getenforce  
# setenforce 0  
# getenforce  
# setenforce 1  
# getenforce  
# cat /etc/selinux/config
```

```
[root@rhel7 ~]# getenforce  
Enforcing  
[root@rhel7 ~]# setenforce 0  
[root@rhel7 ~]# getenforce  
Permissive  
[root@rhel7 ~]# setenforce 1  
[root@rhel7 ~]# getenforce  
Enforcing  
[root@rhel7 ~]#
```

```
[root@rhel7 ~]# cat /etc/selinux/config  
  
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.  
#       permissive - SELinux prints warnings instead of enforcing.  
#       disabled - No SELinux policy is loaded.  
SELINUX=enforcing
```

## Set SELinux Mode

Typically you will use `setenforce` to toggle between SELinux modes (enforcing to permissive and back) as a first troubleshooting step. If SELinux is currently set to **enforcing** while

you're experiencing a certain problem, and the same goes away when you set it to **permissive**, you can be confident you're looking at a SELinux permissions issue.

## SELinux Contexts

A SELinux context consists of an access control environment where decisions are made based on SELinux user, role, and type (and optionally a level):

1. A SELinux user complements a regular Linux user account by mapping it to a SELinux user account, which in turn is used in the SELinux context for processes in that session, in order to explicitly define their allowed roles and levels.
2. The concept of role acts as an intermediary between domains and SELinux users in that it defines which process domains and file types can be accessed. This will shield your system against vulnerability to privilege escalation attacks.
3. A type defines an SELinux file type or an SELinux process domain. Under normal circumstances, processes are prevented from accessing files that other processes use, and from accessing other processes, thus access is only allowed if a specific SELinux policy rule exists that allows it.

Let's see how all of that works through the following examples.

### EXAMPLE 1: Changing the default port for the sshd daemon

In [Securing SSH – Part 8](#) we explained that changing the default port where **sshd** listens on is one of the first security measures to secure your server against external attacks. Let's edit the `/etc/ssh/sshd_config` file and set the port to **9999**:

```
Port 9999
```

Save the changes, and restart sshd:

```
# systemctl restart sshd
# systemctl status sshd
[root@rhel7 ~]# systemctl restart sshd
[root@rhel7 ~]# systemctl status sshd
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
   Active: activating (auto-restart) (Result: exit-code) since Mon 2015-05-25 16:03:23
     Process: 3583 ExecStart=/usr/sbin/sshd -D $OPTIONS (code=exited, status=255)
   Main PID: 3583 (code=exited, status=255)

May 25 16:03:23 rhel7 systemd[1]: sshd.service: main process exited, code=exited
May 25 16:03:23 rhel7 systemd[1]: Unit sshd.service entered failed state.
[root@rhel7 ~]#
```

<http://www.tecmint.com>

### Restart SSH Service

As you can see, sshd has failed to start. But what happened?

A quick inspection of `/var/log/audit/audit.log` indicates that sshd has been denied permissions to start on port **9999** (SELinux log messages include the word “AVC” so that

they might be easily identified from other messages) because that is a reserved port for the **JBoss Management** service:

```
# cat /var/log/audit/audit.log | grep AVC | tail -1
[root@rhel7 ~]# cat /var/log/audit/audit.log | grep AVC | tail -1
type=AVC msg=audit(1432584371.456:568): avc: denied { name_bind } for pid=3599 comm="sshd" src=9999 scont
u:object_r:jboss_management_port_t:s0 tclass=tcp_socket
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Inspect SSH Logs

At this point you could disable **SELinux** (but don't!) as explained earlier and try to start **sshd** again, and it should work. However, the **semanage** utility can tell us what we need to change in order for us to be able to start sshd in whatever port we choose without issues.

Run,

```
# semanage port -l | grep ssh
```

to get a list of the ports where SELinux allows sshd to listen on.

```
[root@rhel7 ~]# semanage port -l | grep ssh
ssh_port_t          tcp      22
[root@rhel7 ~]# getenforce
Enforcing
[root@rhel7 ~]# systemctl is-active sshd
activating
[root@rhel7 ~]# systemctl restart sshd
[root@rhel7 ~]# systemctl is-active sshd
activating
[root@rhel7 ~]# semanage port -a -t ssh_port_t -p tcp 9999
ValueError: Port tcp/9999 already defined
[root@rhel7 ~]# semanage port -l | grep 9999
jboss_management_port_t      tcp      4712, 4447, 7600, 9123, 9999, http://www.tecmint.com:18001
```

sshd will not start as long as the port where we told it to listen on is not allowed by the SELinux ssh\_port\_t context

## Semanage Tool

So let's change the port in `/etc/ssh/sshd_config` to Port **9998**, add the port to the **ssh\_port\_t context**, and then restart the service:

```
# semanage port -a -t ssh_port_t -p tcp 9998
# systemctl restart sshd
# systemctl is-active sshd
[root@rhel7 ~]# semanage port -a -t ssh_port_t -p tcp 9998
[root@rhel7 ~]# systemctl restart sshd
[root@rhel7 ~]# systemctl is-active sshd
active
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Semanage Add Port

As you can see, the service was started successfully this time. This example illustrates the fact that SELinux controls the TCP port number to its own port type internal definitions.

## EXAMPLE 2: Allowing httpd to send access sendmail

This is an example of SELinux managing a process accessing another process. If you were to implement [mod\\_security and mod\\_evasive along with Apache in your RHEL 7](#) server, you need to allow **httpd** to access **sendmail** in order to send a mail notification in the wake of a **(D)DoS** attack. In the following command, omit the **-P** flag if you do not want the change to be persistent across reboots.

```
# semanage boolean -l | grep httpd_can_sendmail
# setsebool -P httpd_can_sendmail 1
# semanage boolean -l | grep httpd_can_sendmail
[root@rhel7 ~]# semanage boolean -l | grep httpd_can_sendmail
httpd_can_sendmail BEFORE (off , off) Allow httpd to can sendmail
[root@rhel7 ~]# setsebool -P httpd_can_sendmail 1
[root@rhel7 ~]# semanage boolean -l | grep httpd_can_sendmail
httpd_can_sendmail AFTER (on , on) Allow httpd to can sendmail
[root@rhel7 ~]# 
```

<http://www.tecmint.com>

## Allow Apache to Send Mails

As you can tell from the above example, **SELinux boolean** settings (or just booleans) are true / false rules embedded into SELinux policies. You can list all the booleans with `semanage boolean -l`, and alternatively pipe it to `grep` in order to filter the output.

## EXAMPLE 3: Serving a static site from a directory other than the default one

Suppose you are serving a static website using a different directory than the default one (`/var/www/html`), say **/websites** (this could be the case if you're storing your web files in a shared network drive, for example, and need to mount it at **/websites**).

- Create an **index.html** file inside **/websites** with the following contents:

```
<html>
<h2>SELinux test</h2>
</html>
```

If you do,

```
# ls -lZ /websites/index.html
```

you will see that the **index.html** file has been labeled with the **default\_t** SELinux type, which Apache can't access:

```
[root@rhel7 websites]# ls -lZ
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
```

## Check SELinux File Permission

- Change the **DocumentRoot** directive in `/etc/httpd/conf/httpd.conf` to **/websites** and don't forget to update the corresponding `Directory` block. Then, restart Apache.

c). Browse to `http://<web server IP address>`, and you should get a 503 Forbidden HTTP response.

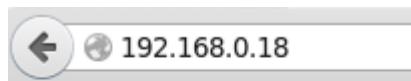
d). Next, change the label of `/websites`, recursively, to the `httpd_sys_content_t` type in order to grant Apache read-only access to that directory and its contents:

```
# semanage fcontext -a -t httpd_sys_content_t "/websites(/.*)?"
```

e). Finally, apply the SELinux policy created in d):

```
# restorecon -R -v /websites
```

Now restart Apache and browse to `http://<web server IP address>` again and you will see the html file displayed correctly:



Verify Apache Page

## Summary

In this article we have gone through the basics of **SELinux**. Note that due to the vastness of the subject, a full detailed explanation is not possible in a single article, but we believe that the principles outlined in this guide will help you to move on to more advanced topics should you wish to do so.

If I may, let me recommend two essential resources to start with: the [NSA SELinux page](#) and the [RHEL 7 SELinux User's and Administrator's guide](#).

Don't hesitate to let us know if you have any questions or comments.

## RHCSA Series: Setting Up LDAP-based Authentication in RHEL 7 – Part 14

We will begin this article by outlining some **LDAP** basics (what it is, where it is used and why) and show how to set up a LDAP server and configure a client to authenticate against it using **Red Hat Enterprise Linux 7** systems.

# RHCSA: Red Hat Certified System Administrator

## RED HAT ENTERPRISE LINUX 7

<http://www.tecmint.com>



### Part 14: Setting up LDAP-based authentication in RHEL 7

RHCSA Series: Setup LDAP Server and Client Authentication – Part 14

As we will see, there are several other possible application scenarios, but in this guide we will focus entirely on **LDAP-based** authentication. In addition, please keep in mind that due to the vastness of the subject, we will only cover its basics here, but you can refer to the documentation outlined in the summary for more in-depth details.

For the same reason, you will note that I have decided to leave out several references to many pages of LDAP tools for the sake of brevity, but the corresponding explanations are at a fingertip's distance (**man ldapadd**, for example).

That said, let's get started.

#### Our Testing Environment

Our test environment consists of two **RHEL 7** boxes:

**Server:** 192.168.0.18. **FQDN:** rhel7.mydomain.com  
**Client:** 192.168.0.20. **FQDN:** ldapclient.mydomain.com

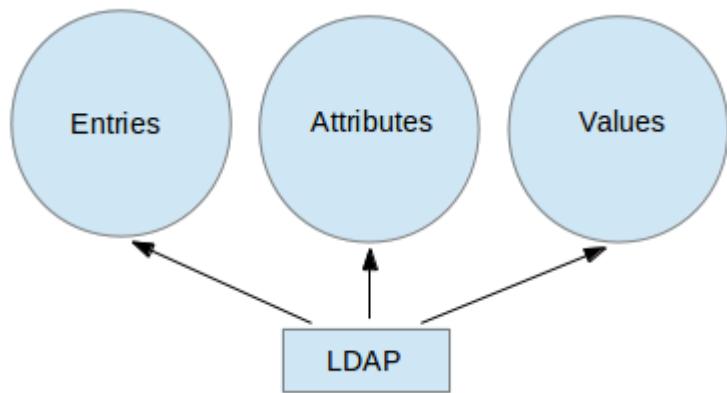
If you want, you can use the machine installed in [Part 12: Automate RHEL 7 installations](#) using Kickstart as client.

#### What is LDAP?

**LDAP** stands for **Lightweight Directory Access Protocol** and consists in a set of protocols that allows a client to access, over a network, centrally stored information (such as a directory of login shells, absolute paths to home directories, and other typical system user information, for example) that should be accessible from different places or available to a large number of end users (another example would be a directory of home addresses and phone numbers of all employees in a company).

Keeping such (and more) information centrally means it can be more easily maintained and accessed by everyone who has been granted permissions to use it.

The following diagram offers a simplified diagram of **LDAP**, and is described below in greater detail:



LDAP Diagram

Explanation of above diagram in detail.

1. An **entry** in a LDAP directory represents a single unit or information and is uniquely identified by what is called a Distinguished Name.
2. An **attribute** is a piece of information associated with an entry (for example, addresses, available contact phone numbers, and email addresses).
3. Each attribute is assigned one or more **values** consisting in a space-separated list. A value that is unique per entry is called a Relative Distinguished Name.

That being said, let's proceed with the server and client installations.

## Installing and Configuring a LDAP Server and Client

In **RHEL 7**, LDAP is implemented by **OpenLDAP**. To install the server and client, use the following commands, respectively:

```
# yum update && yum install openldap openldap-clients openldap-servers  
# yum update && yum install openldap openldap-clients nss-pam-ldapd
```

Once the installation is complete, there are some things we look at. The following steps should be performed on the server alone, unless explicitly noted:

1. Make sure **SELinux** does not get in the way by enabling the following **booleans** persistently, both on the server and the client:

```
# setsebool -P allow_ypbind=0 authlogin_nsswitch_use_ldap=0
```

Where **allow\_ypbind** is required for LDAP-based authentication, and **authlogin\_nsswitch\_use\_ldap** may be needed by some applications.

## 2. Enable and start the service:

```
# systemctl enable slapd.service  
# systemctl start slapd.service
```

Keep in mind that you can also disable, restart, or stop the service with [systemctl](#) as well:

```
# systemctl disable slapd.service  
# systemctl restart slapd.service  
# systemctl stop slapd.service
```

## 3. Since the **slapd** service runs as the **ldap** user (which you can verify with **ps -e -o pid,uname,comm | grep slapd**), such user should own the **/var/lib/ldap** directory in order for the server to be able to modify entries created by administrative tools that can only be run as root (more on this in a minute).

Before changing the ownership of this directory recursively, copy the sample database configuration file for **slapd** into it:

```
# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG  
# chown -R ldap:ldap /var/lib/ldap
```

## 4. Set up an OpenLDAP administrative user and assign a password:

```
# slappasswd
```

as shown in the next image:

```
[root@rhel7 ~]# slappasswd  
New password:  
Re-enter new password:  
{SSHA}ZhsLQTl [REDACTED] eQ7Di7PAcRNq  
[root@rhel7 ~]#
```

Set LDAP Admin Password

and create an **LDIF** file (**ldaprootpasswd.ldif**) with the following contents:

```
dn: olcDatabase={0}config,cn=config  
changetype: modify  
add: olcRootPW  
olcRootPW: {SSHA}PASSWORD
```

where:

1. **PASSWORD** is the hashed string obtained earlier.
2. **cn=config** indicates global config options.
3. **olcDatabase** indicates a specific database instance name and can be typically found inside **/etc/openldap/slapd.d/cn=config**.

Referring to the theoretical background provided earlier, the **ldaprootpasswd.ldif** file will add an entry to the LDAP directory. In that entry, each line represents an attribute: value pair

(where dn, changetype, add, and olcRootPW are the attributes and the strings to the right of each colon are their corresponding values).

You may want to keep this in mind as we proceed further, and please note that we are using the same Common Names (`cn=`) throughout the rest of this article, where each step depends on the previous one.

**5.** Now, add the corresponding LDAP entry by specifying the **URI** referring to the ldap server, where only the protocol/host/port fields are allowed.

```
# ldapadd -H ldapi:/// -f ldaprootpasswd.ldif
```

The output should be similar to:

```
[root@rhel7 ~]# [ldapadd -H ldapi:/// -f ldaprootpasswd.ldif]
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={0}config,cn=config"

[root@rhel7 ~]#
```

## LDAP Configuration

and import some basic LDAP definitions from the `/etc/openldap/schema` directory:

```
# for def in cosine.ldif nis.ldif inetorgperson.ldif; do ldapadd -H
ldapi:/// -f /etc/openldap/schema/$def; done
[root@rhel7 ~]# for def in cosine.ldif nis.ldif inetorgperson.ldif; do ldapadd -H ldapi:/// -f /etc/openldap/schema/$def; done
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=cosine,cn=schema,cn=config"

SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=nis,cn=schema,cn=config"

SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=inetorgperson,cn=schema,cn=config"

[root@rhel7 ~]#
```

## LDAP Definitions

**6.** Have LDAP use your domain in its database.

Create another **LDIF** file, which we will call `ldapdomain.ldif`, with the following contents, replacing your domain (in the Domain Component `dc=`) and password as appropriate:

```
dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by
dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth"
```

```

read by dn.base="cn=Manager,dc=mydomain,dc=com" read by * none

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=mydomain,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=Manager,dc=mydomain,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}PASSWORD

dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange by
dn="cn=Manager,dc=mydomain,dc=com" write by anonymous auth by self write
by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=Manager,dc=mydomain,dc=com" write by * read

```

Then load it as follows:

```

# ldapmodify -H ldap:// -f ldapdomain.ldif
[root@rhel7 ~]# vim ldapdomain.ldif
[root@rhel7 ~]# ldapmodify -H ldap:// -f ldapdomain.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={1}monitor,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

modifying entry "olcDatabase={2}hdb,cn=config"

[root@rhel7 ~]#

```

## LDAP Domain Configuration

7. Now it's time to add some entries to our LDAP directory. Attributes and values are separated by a colon (:) in the following file, which we'll name baseldapdomain.ldif:

```

dn: dc=mydomain,dc=com
objectClass: top
objectClass: dcObject
objectclass: organization
o: mydomain com
dc: mydomain

```

```

dn: cn=Manager,dc=mydomain,dc=com
objectClass: organizationalRole
cn: Manager
description: Directory Manager

dn: ou=People,dc=mydomain,dc=com
objectClass: organizationalUnit
ou: People

dn: ou=Group,dc=mydomain,dc=com
objectClass: organizationalUnit
ou: Group

```

Add the entries to the LDAP directory:

```

# ldapadd -x -D cn=Manager,dc=mydomain,dc=com -W -f baseldapdomain.ldif
[root@rhel7 ~]# vim baseldapdomain.ldif
[root@rhel7 ~]# ldapadd -x -D cn=Manager,dc=mydomain,dc=com -W -f baseldapdomain.ldif
Enter LDAP Password:
adding new entry "dc=mydomain,dc=com"

adding new entry "cn=Manager,dc=mydomain,dc=com"

adding new entry "ou=People,dc=mydomain,dc=com"

adding new entry "ou=Group,dc=mydomain,dc=com"

[root@rhel7 ~]#

```

Add LDAP Domain Attributes and Values

**8.** Create a LDAP user called **ldapuser** (**adduser ldapuser**), then create the definitions for a LDAP group in **ldapgroup.ldif**.

```
# adduser ldapuser
# vi ldapgroup.ldif
```

Add following content.

```

dn: cn=Manager,ou=Group,dc=mydomain,dc=com
objectClass: top
objectClass: posixGroup
gidNumber: 1004

```

where **gidNumber** is the GID in **/etc/group** for ldapuser) and load it:

```
# ldapadd -x -W -D "cn=Manager,dc=mydomain,dc=com" -f ldapgroup.ldif
```

**9.** Add a LDIF file with the definitions for user ldapuser (**ldapuser.ldif**):

```

dn: uid=ldapuser,ou=People,dc=mydomain,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: ldapuser
uid: ldapuser

```

```
uidNumber: 1004
gidNumber: 1004
homeDirectory: /home/ldapuser
userPassword: {SSHA}fiN0YqzbDuDI0Fpqg9UudWmjZQY28S3M
loginShell: /bin/bash
gecos: ldapuser
shadowLastChange: 0
shadowMax: 0
shadowWarning: 0
```

and load it:

```
# ldapadd -x -D cn=Manager,dc=mydomain,dc=com -W -f ldapuser.ldif
[root@rhel7 ~]# vim ldapuser.ldif
[root@rhel7 ~]# systemctl restart slapd
[root@rhel7 ~]# ldapadd -x -D cn=Manager,dc=mydomain,dc=com -W -f ldapuser.ldif
Enter LDAP Password:
adding new entry "uid=ldapuser,ou=People,dc=mydomain,dc=com"

[root@rhel7 ~]#
```

## LDAP User Configuration

Likewise, you can delete the user entry you just created:

```
# ldapdelete -x -W -D cn=Manager,dc=mydomain,dc=com
"uid=ldapuser,ou=People,dc=mydomain,dc=com"
```

### 10. Allow communication through the firewall:

```
# firewall-cmd --add-service=ldap
```

### 11. Last, but not least, enable the client to authenticate using LDAP.

To help us in this final step, we will use the **authconfig** utility (an interface for configuring system authentication resources).

Using the following command, the home directory for the requested user is created if it doesn't exist after the authentication against the LDAP server succeeds:

```
# authconfig --enableldap --enableldapauth --ldapserver=rhel7.mydomain.com
--ldapbasedn="dc=mydomain,dc=com" --enablemkhomedir --update
```

```
Red Hat Enterprise Linux  
Kernel 3.10.0-229.4.2.el7.x86_64 on an x86_64  
  
Hint: Num Lock on  
  
[ldapclient login: ldapuser  
Password:  
You are required to change your password immediately (password aged)  
password expired 16596 days ago  
New password:  
Retype new password:  
Creating directory '/home/ldapuser'.  
Last login: Tue Jun  9 00:36:06 on tty1  
[ldapuser@ldapclient ~]$ _
```

## LDAP Client Configuration

### Summary

In this article we have explained how to set up basic authentication against a LDAP server. To further configure the setup described in the present guide, please refer to [Chapter 13 – LDAP Configuration](#) in the RHEL 7 System administrator's guide, paying special attention to the security settings using TLS.

Feel free to leave any questions you may have using the comment form below.

## RHCSA Series: Essentials of Virtualization and Guest Administration with KVM – Part 15

If you look up the word virtualize in a dictionary, you will find that it means “**to create a virtual (rather than actual) version of something**”. In computing, the term virtualization refers to the possibility of running multiple operating systems simultaneously and isolated one from another, on top of the same physical (hardware) system, known in the virtualization schema as **host**.

### **RHCSA: Red Hat Certified System Administrator**

### **RED HAT ENTERPRISE LINUX 7**



### **Part 15: Essentials of virtualization and guest administration with KVM in RHEL 7**

## RHCSA Series: Essentials of Virtualization and Guest Administration with KVM – Part 15

Through the use of the virtual machine monitor (also known as **hypervisor**), virtual machines (referred to as **guests**) are provided virtual resources (i.e. CPU, RAM, storage, network interfaces, to name a few) from the underlying hardware.

With that in mind, it is plain to see that one of the main advantages of virtualization is cost savings (in equipment and network infrastructure and in terms of maintenance effort) and a substantial reduction in the physical space required to accommodate all the necessary hardware.

Since this brief how-to cannot cover all virtualization methods, I encourage you to refer to the documentation listed in the summary for further details on the subject.

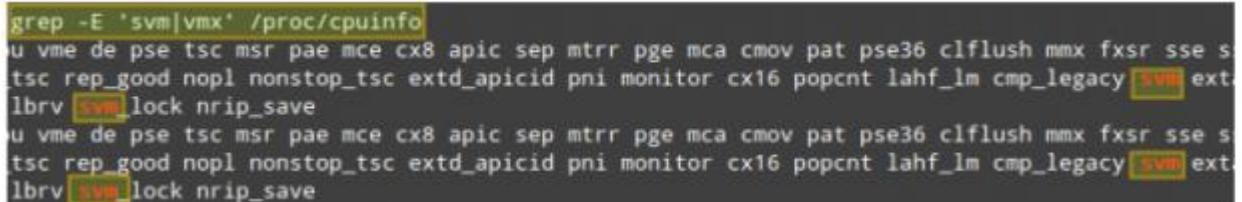
Please keep in mind that the present article is intended to be a starting point to learn the basics of virtualization in **RHEL 7** using **KVM** (Kernel-based Virtual Machine) with command-line utilities, and not an in-depth discussion of the topic.

### **Verifying Hardware Requirements and Installing Packages**

In order to set up virtualization, your CPU must support it. You can verify whether your system meets the requirements with the following command:

```
# grep -E 'svm|vmx' /proc/cpuinfo
```

In the following screenshot we can see that the current system (with an **AMD** microprocessor) supports virtualization, as indicated by **svm**. If we had an Intel-based processor, we would see **vmx** instead in the results of the above command.



```
grep -E 'svm|vmx' /proc/cpuinfo
u vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse s
tsc rep_good nopl nonstop_tsc extd_apicid pni monitor cx16 popcnt lahf_lm cmp_legacy svm ext
lbrv svm lock nrip_save
u vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse s
tsc rep_good nopl nonstop_tsc extd_apicid pni monitor cx16 popcnt lahf_lm cmp_legacy svm ext
lbrv svm lock nrip_save
```

## Check KVM Support

In addition, you will need to have virtualization capabilities enabled in the firmware of your host (**BIOS** or **UEFI**).

Now install the necessary packages:

1. **qemu-kvm** is an open source virtualizer that provides hardware emulation for the KVM hypervisor whereas **qemu-img** provides a command line tool for manipulating disk images.
2. **libvirt** includes the tools to interact with the virtualization capabilities of the operating system.
3. **libvirt-python** contains a module that permits applications written in Python to use the interface supplied by libvirt.
4. **libguestfs-tools**: miscellaneous system administrator command line tools for virtual machines.
5. **virt-install**: other command-line utilities for virtual machine administration.

```
# yum update && yum install qemu-kvm qemu-img libvirt libvirt-python
libguestfs-tools virt-install
```

Once the installation completes, make sure you start and enable the **libvirtd** service:

```
# systemctl start libvirtd.service
# systemctl enable libvirtd.service
```

By default, each virtual machine will only be able to communicate with the rest in the same physical server and with the host itself. To allow the guests to reach other machines inside our LAN and also the Internet, we need to set up a bridge interface in our host (say **br0**, for example) by,

1. adding the following line to our main NIC configuration (most likely `/etc/sysconfig/network-scripts/ifcfg-enp0s3`):

```
BRIDGE=br0
```

**2.** creating the configuration file for **br0** (`/etc/sysconfig/network-scripts/ifcfg-br0`) with these contents (note that you may have to change the IP address, gateway address, and DNS information):

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=static
IPADDR=192.168.0.18
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
NM_CONTROLLED=no
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=br0
ONBOOT=yes
DNS1=8.8.8.8
DNS2=8.8.4.4
```

**3.** finally, enabling packet forwarding by making, in `/etc/sysctl.conf`,

```
net.ipv4.ip_forward = 1
```

and loading the changes to the current kernel configuration:

```
# sysctl -p
```

Note that you may also need to tell **firewalld** that this kind of traffic should be allowed. Remember that you can refer to the article on that topic in this same series ([Part 11: Network Traffic Control Using FirewallD and Iptables](#)) if you need help to do that.

## Creating VM Images

By default, VM images will be created to `/var/lib/libvirt/images` and you are strongly advised to not change this unless you really need to, know what you're doing, and want to handle SELinux settings yourself (such topic is out of the scope of this tutorial but you can refer to **Part 13** of the RHCSA series: [Mandatory Access Control Essentials with SELinux](#) if you want to refresh your memory).

This means that you need to make sure that you have allocated the necessary space in that filesystem to accommodate your virtual machines.

The following command will create a virtual machine named `tecmint-virt01` with 1 virtual CPU, 1 GB (=1024 MB) of RAM, and 20 GB of disk space (represented by `/var/lib/libvirt/images/tecmint-virt01.img`) using the **rhel-server-7.0-x86\_64-dvd.iso** image located inside **/home/gacanepa/ISOs** as installation media and the **br0** as network bridge:

```
# virt-install \
--network bridge=br0
--name tecmint-virt01 \
--ram=1024 \
--vcpus=1 \
--disk path=/var/lib/libvirt/images/tecmint-virt01.img,size=20 \
--graphics none \
--cdrom /home/gacanepa/ISOs/rhel-server-7.0-x86_64-dvd.iso
--extra-args="console=tty0 console=ttyS0,115200"
```

If the installation file was located in a HTTP server instead of an image stored in your disk, you will have to replace the **-cdrom** flag with **-location** and indicate the address of the online repository.

As for the **--graphics** none option, it tells the installer to perform the installation in text-mode exclusively. You can omit that flag if you are using a GUI interface and a VNC window to access the main VM console. Finally, with **--extra-args** we are passing kernel boot parameters to the installer that set up a serial VM console.

The installation should now proceed as a regular (real) server now. If not, please review the steps listed above.

## Managing Virtual Machines

These are some typical administration tasks that you, as a system administrator, will need to perform on your virtual machines. Note that all of the following commands need to be run from your host:

### 1. List all VMs:

```
# virsh list --all
```

From the output of the above command you will have to note the **Id** for the virtual machine (although it will also return its name and current status) because you will need it for most administration tasks related to a particular VM.

### 2. Display information about a guest:

```
# virsh dominfo [VM Id]
```

### 3. Start, restart, or stop a guest operating system:

```
# virsh start | reboot | shutdown [VM Id]
```

### 4. Access a VM's serial console if networking is not available and no X server is running on the host:

```
# virsh console [VM Id]
```

**Note** that this will require that you add the serial console configuration information to the `/etc/grub.conf` file (refer to the argument passed to the `-extra-args` option when the VM was created).

## 5. Modify assigned memory or virtual CPUs:

First, shutdown the guest:

```
# virsh shutdown [VM Id]
```

Edit the VM configuration for RAM:

```
# virsh edit [VM Id]
```

Then modify

```
<memory>[Memory size here without brackets]</memory>
```

Restart the VM with the new settings:

```
# virsh create /etc/libvirt/qemu/tecmint-virt01.xml
```

Finally, change the memory dynamically:

```
# virsh setmem [VM Id] [Memory size here without brackets]
```

For CPU:

```
# virsh edit [VM Id]
```

Then modify

```
<cpu>[Number of CPUs here without brackets]</cpu>
```

For further commands and details, please refer to **table 26.1** in Chapter 26 of the RHEL 5 Virtualization guide (that guide, though a bit old, includes an exhaustive list of virsh commands used for guest administration).

## SUMMARY

In this article we have covered some basic aspects of virtualization with **KVM** in **RHEL 7**, which is both a vast and a fascinating topic, and I hope it will be helpful as a starting guide for you to later explore more advanced subjects found in the official [RHEL virtualization getting started](#) and [deployment / administration guides](#).

In addition, you can refer to the preceding articles in [this KVM series](#) in order to clarify or expand some of the concepts explained here.

# RHCE Series: How to Setup and Test Static Network Routing – Part 1

by [Gabriel Cánepa](#) | Published: July 29, 2015 | Last Updated: March 11, 2016

**Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)**

**RHCE (Red Hat Certified Engineer)** is a certification from Red Hat company, which gives an open source operating system and software to the enterprise community. It also gives training, support and consulting services for the companies.



## RHCE Exam Preparation Guide

This **RHCE (Red Hat Certified Engineer)** is a performance-based exam (codename **EX300**), who possesses the additional skills, knowledge, and abilities required of a senior system administrator responsible for Red Hat Enterprise Linux (RHEL) systems.

**Important:** [Red Hat Certified System Administrator](#) (RHCSA) certification is required to earn RHCE certification.

Following are the exam objectives based on the **Red Hat Enterprise Linux 7** version of the exam, which will going to cover in this RHCE series:

**Part 1: How to Setup and Test Static Routing in RHEL 7**

**Part 2: [How to Perform Packet Filtering, Network Address Translation and Set Kernel Runtime Parameters](#)**

**Part 3: [How to Produce and Deliver System Activity Reports Using Linux Toolsets](#)**

**Part 4: [Using Shell Scripting to Automate Linux System Maintenance Tasks](#)**

**Part 5: [How to Manage System Logs \(Configure, Rotate and Import Into Database\) in RHEL 7](#)**

- Part 6:** [Setting Up Samba and Configure FirewallD and SELinux to Allow File Sharing on Clients](#)
- Part 7:** [Setting Up NFS Server with Kerberos-based Authentication for Linux Clients](#)
- Part 8:** [Implementing HTTPS through TLS using Network Security Service \(NSS\) for Apache](#)
- Part 9:** [How to Setup Postfix Mail Server \(SMTP\) using null-client Configuration](#)
- Part 10:** [Install and Configure Caching-Only DNS Server in RHEL/CentOS 7](#)
- Part 11:** [Setup and Configure Network Bonding or Teaming in RHEL/CentOS 7](#)
- Part 12:** [Create Centralized Secure Storage using iSCSI Target / Initiator on RHEL/CentOS 7](#)
- Part 13:** [Setting Up “NTP \(Network Time Protocol\) Server” in RHEL/CentOS 7](#)

To view fees and register for an exam in your country, check the [RHCE Certification](#) page.

In this **Part 1** of the **RHCE** series and the next, we will present basic, yet typical, cases where the principles of static routing, packet filtering, and network address translation come into play.



RHCE: Setup and Test Network Static Routing – Part 1

Please note that we will not cover them in depth, but rather organize these contents in such a way that will be helpful to take the first steps and build from there.

## Static Routing in Red Hat Enterprise Linux 7

One of the wonders of modern networking is the vast availability of devices that can connect groups of computers, whether in relatively small numbers and confined to a single room or several machines in the same building, city, country, or across continents.

However, in order to effectively accomplish this in any situation, network packets need to be routed, or in other words, the path they follow from source to destination must be ruled somehow.

Static routing is the process of specifying a route for network packets other than the default, which is provided by a network device known as the default gateway. Unless specified otherwise through static routing, network packets are directed to the default gateway; with static routing, other paths are defined based on predefined criteria, such as the packet destination.

Let us define the following scenario for this tutorial. We have a Red Hat Enterprise Linux 7 box connecting to router #1 [192.168.0.1] to access the Internet and machines in 192.168.0.0/24.

A second router (**router #2**) has two network interface cards: **enp0s3** is also connected to **router #1** to access the Internet and to communicate with the RHEL 7 box and other machines in the same network, whereas the other (**enp0s8**) is used to grant access to the **10.0.0.0/24** network where internal services reside, such as a web and / or database server.

This scenario is illustrated in the diagram below:



Static Routing Network Diagram

In this article we will focus exclusively on setting up the routing table on our **RHEL 7** box to make sure that it can both access the Internet through **router #1** and the internal network via **router #2**.

In **RHEL 7**, you will use the [ip command](#) to configure and show devices and routing using the command line. These changes can take effect immediately on a running system but since they are not persistent across reboots, we will use **ifcfg-enp0sX** and **route-enp0sX** files inside **/etc/sysconfig/network-scripts** to save our configuration permanently.

To begin, let's print our current routing table:

```
# ip route show
[root@rhel7 ~]# ip route show
default via 192.168.0.1 dev enp0s3
169.254.0.0/16 dev enp0s3  scope link  metric 1002
192.168.0.0/24 dev enp0s3  proto kernel  scope link  src 192.168.0.18
[root@rhel7 ~]#
```

#### Check Current Routing Table

From the output above, we can see the following facts:

1. The default gateway's IP address is **192.168.0.1** and can be accessed via the **enp0s3** NIC.

2. When the system booted up, it enabled the zeroconf route to **169.254.0.0/16** (just in case). In few words, if a machine is set to obtain an IP address through DHCP but fails to do so for some reason, it is automatically assigned an address in this network. Bottom line is, this route will allow us to communicate, also via **enp0s3**, with other machines who have failed to obtain an IP address from a DHCP server.
3. Last, but not least, we can communicate with other boxes inside the **192.168.0.0/24** network through **enp0s3**, whose IP address is **192.168.0.18**.

These are the typical tasks that you would have to perform in such a setting. Unless specified otherwise, the following tasks should be performed in **router #2**:

Make sure all NICs have been properly installed:

```
# ip link show
```

If one of them is down, bring it up:

```
# ip link set dev enp0s8 up
```

and assign an IP address in the **10.0.0.0/24** network to it:

```
# ip addr add 10.0.0.17 dev enp0s8
```

Oops! We made a mistake in the IP address. We will have to remove the one we assigned earlier and then add the right one (**10.0.0.18**):

```
# ip addr del 10.0.0.17 dev enp0s8
# ip addr add 10.0.0.18 dev enp0s8
```

Now, please note that you can only add a route to a destination network through a gateway that is itself already reachable. For that reason, we need to assign an IP address within the **192.168.0.0/24** range to **enp0s3** so that our RHEL 7 box can communicate with it:

```
# ip addr add 192.168.0.19 dev enp0s3
```

Finally, we will need to enable packet forwarding:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

and stop / disable (just for the time being – until we cover packet filtering in the next article) the firewall:

```
# systemctl stop firewalld
# systemctl disable firewalld
```

Back in our **RHEL 7** box (**192.168.0.18**), let's configure a route to **10.0.0.0/24** through **192.168.0.19** (**enp0s3** in **router #2**):

```
# ip route add 10.0.0.0/24 via 192.168.0.19
```

After that, the routing table looks as follows:

```
# ip route show
[root@rhel7 ~]# ip route show
default via 192.168.0.1 dev enp0s3
10.0.0.0/24 via 192.168.0.19 dev enp0s3
169.254.0.0/16 dev enp0s3 scope link metric 1002
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.18
[root@rhel7 ~]#
```

## Confirm Network Routing Table

Likewise, add the corresponding route in the machine(s) you're trying to reach in **10.0.0.0/24**:

```
# ip route add 192.168.0.0/24 via 10.0.0.18
```

You can test for basic connectivity using **ping**:

In the **RHEL 7** box, run

```
# ping -c 4 10.0.0.20
```

where **10.0.0.20** is the IP address of a web server in the **10.0.0.0/24** network.

In the web server (**10.0.0.20**), run

```
# ping -c 192.168.0.18
```

where **192.168.0.18** is, as you will recall, the IP address of our RHEL 7 machine.

Alternatively, we can use [tcpdump](#) (you may need to install it with **yum install tcpdump**) to check the 2-way communication over TCP between our RHEL 7 box and the web server at **10.0.0.20**.

To do so, let's start the logging in the first machine with:

```
# tcpdump -qnnvvv -i enp0s3 host 10.0.0.20
```

and from another terminal in the same system let's **telnet** to port **80** in the web server (assuming **Apache** is listening on that port; otherwise, indicate the right port in the following command):

```
# telnet 10.0.0.20 80
```

The **tcpdump** log should look as follows:

```
[root@rhel7 ~]# tcpdump -qnnvvv -i enp0s3 host 10.0.0.20
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 65535 bytes
23:12:22.682531 IP (tos 0x10, ttl 64, id 61663, offset 0, flags [DF], proto TCP (6), length 60)
  192.168.0.18.36676 > 10.0.0.20.80: tcp 0
23:12:22.683875 IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto TCP (6), length 60)
  10.0.0.20.80 > 192.168.0.18.36676: tcp 0
23:12:22.683965 IP (tos 0x10, ttl 64, id 61664, offset 0, flags [DF], proto TCP (6), length 52)
```

## Check Network Communication between Servers

Where the connection has been properly initialized, as we can tell by looking at the 2-way communication between our **RHEL 7** box (**192.168.0.18**) and the web server (**10.0.0.20**).

Please remember that these changes will go away when you restart the system. If you want to make them persistent, you will need to edit (or create, if they don't already exist) the following files, in the same systems where we performed the above commands.

Though not strictly necessary for our test case, you should know that **/etc/sysconfig/network** contains system-wide network parameters. A typical **/etc/sysconfig/network** looks as follows:

```
# Enable networking on this system?  
NETWORKING=yes  
# Hostname. Should match the value in /etc/hostname  
HOSTNAME=yourhostnamehere  
# Default gateway  
GATEWAY=XXX.XXX.XXX.XXX  
# Device used to connect to default gateway. Replace X with the appropriate  
number.  
GATEWAYDEV=enp0sX
```

When it comes to setting specific variables and values for each NIC (as we did for router #2), you will have to edit **/etc/sysconfig/network-scripts/ifcfg-enp0s3** and **/etc/sysconfig/network-scripts/ifcfg-enp0s8**.

Following our case,

```
TYPE=Ethernet  
BOOTPROTO=static  
IPADDR=192.168.0.19  
NETMASK=255.255.255.0  
GATEWAY=192.168.0.1  
NAME=enp0s3  
ONBOOT=yes
```

and

```
TYPE=Ethernet  
BOOTPROTO=static  
IPADDR=10.0.0.18  
NETMASK=255.255.255.0  
GATEWAY=10.0.0.1  
NAME=enp0s8  
ONBOOT=yes
```

for **enp0s3** and **enp0s8**, respectively.

As for routing in our client machine (192.168.0.18), **we will need to edit /etc/sysconfig/network-scripts/route-enp0s3:**

```
10.0.0.0/24 via 192.168.0.19 dev enp0s3
```

Now **reboot** your system and you should see that route in your table.

## Summary

In this article we have covered the essentials of static routing in **Red Hat Enterprise Linux 7**. Although scenarios may vary, the case presented here illustrates the required principles and the procedures to perform this task. Before wrapping up, I would like to suggest you to take a look at [Chapter 4](#) of the **Securing and Optimizing Linux** section in The Linux Documentation Project site for further details on the topics covered here.

Free ebook on **Securing & Optimizing Linux: The Hacking Solution (v.3.0)** – This 800+ eBook contains comprehensive collection of Linux security tips and how to use them safely and easily to configure Linux-based applications and services.

In the next article we will talk about packet filtering and network address translation to sum up the networking basic skills needed for the RHCE certification.

As always, we look forward to hearing from you, so feel free to leave your questions, comments, and suggestions using the form below.

## [How to Perform Packet Filtering, Network Address Translation and Set Kernel Runtime Parameters – Part 2](#)

As promised in **Part 1** (“[Setup Static Network Routing](#)”), in this article (**Part 2 of RHCE** series) we will begin by introducing the principles of packet filtering and network address translation (NAT) in **Red Hat Enterprise Linux 7**, before diving into setting runtime kernel parameters to modify the behavior of a running kernel if certain conditions change or needs arise.



RHCE: Network Packet Filtering – Part 2

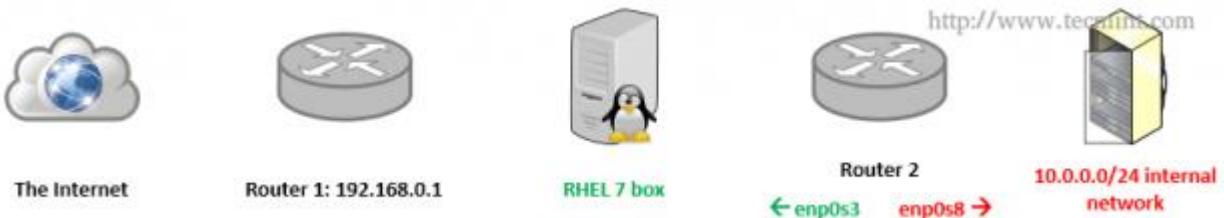
## **Network Packet Filtering in RHEL 7**

When we talk about packet filtering, we refer to a process performed by a firewall in which it reads the header of each data packet that attempts to pass through it. Then, it filters the packet by taking the required action based on rules that have been previously defined by the system administrator.

As you probably know, beginning with **RHEL 7**, the default service that manages firewall rules is [firewalld](#). Like iptables, it talks to the netfilter module in the Linux kernel in order to examine and manipulate network packets. Unlike iptables, updates can take effect immediately without interrupting active connections – you don't even have to restart the service.

Another advantage of **firewalld** is that it allows us to define rules based on pre-configured service names (more on that in a minute).

In **Part 1**, we used the following scenario:



Static Routing Network Diagram

However, you will recall that we disabled the firewall on **router #2** to simplify the example since we had not covered packet filtering yet. Let's see now how we can enable incoming packets destined for a specific service or port in the destination.

First, let's add a permanent rule to allow inbound traffic in **enp0s3** (**192.168.0.19**) to **enp0s8** (**10.0.0.18**):

```
# firewall-cmd --permanent --direct --add-rule ipv4 filter FORWARD 0 -i enp0s3 -o enp0s8 -j ACCEPT
```

The above command will save the rule to **/etc/firewalld/direct.xml**:

```
# cat /etc/firewalld/direct.xml
[root@router2 ~]# cat /etc/firewalld/direct.xml
<?xml version="1.0" encoding="utf-8"?>
<direct>
  <rule priority="0" table="filter" ipv="ipv4" chain="FORWARD">-i enp0s3 -o enp0s8 -j ACCEPT</rule>
</direct>
[root@router2 ~]#
```

### Check Firewalld Saved Rules

Then enable the rule for it to take effect immediately:

```
# firewall-cmd --direct --add-rule ipv4 filter FORWARD 0 -i enp0s3 -o enp0s8 -j ACCEPT
```

Now you can telnet to the web server from the **RHEL 7** box and run [tcpdump](#) again to monitor the TCP traffic between the two machines, this time with the firewall in **router #2** enabled.

```
# telnet 10.0.0.20 80
# tcpdump -qnnvvv -i enp0s3 host 10.0.0.20
```

What if you want to only allow incoming connections to the web server (**port 80**) from **192.168.0.18** and block connections from other sources in the **192.168.0.0/24** network?

In the web server's firewall, add the following rules:

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" source
address="192.168.0.18/24" service name="http" accept'
# firewall-cmd --add-rich-rule 'rule family="ipv4" source
address="192.168.0.18/24" service name="http" accept' --permanent
# firewall-cmd --add-rich-rule 'rule family="ipv4" source
address="192.168.0.0/24" service name="http" drop'
# firewall-cmd --add-rich-rule 'rule family="ipv4" source
address="192.168.0.0/24" service name="http" drop' --permanent
```

Now you can make HTTP requests to the web server, from **192.168.0.18** and from some other machine in **192.168.0.0/24**. In the first case the connection should complete successfully, whereas in the second it will eventually timeout.

To do so, any of the following commands will do the trick:

```
# telnet 10.0.0.20 80
# wget 10.0.0.20
```

I strongly advise you to check out the [Firewalld Rich Language](#) documentation in the Fedora Project Wiki for further details on rich rules.

## Network Address Translation in RHEL 7

**Network Address Translation (NAT)** is the process where a group of computers (it can also be just one of them) in a private network are assigned an unique public IP address. As result, they are still uniquely identified by their own private IP address inside the network but to the outside they all “seem” the same.

In addition, NAT makes it possible that computers inside a network sends requests to outside resources (like the Internet) and have the corresponding responses be sent back to the source system only.

Let's now consider the following scenario:



## Network Address Translation

In **router #2**, we will move the **enp0s3** interface to the external zone, and **enp0s8** to the internal zone, where masquerading, or NAT, is enabled by default:

```
# firewall-cmd --list-all --zone=external
# firewall-cmd --change-interface=enp0s3 --zone=external
# firewall-cmd --change-interface=enp0s3 --zone=external --permanent
# firewall-cmd --change-interface=enp0s8 --zone=internal
# firewall-cmd --change-interface=enp0s8 --zone=internal --permanent
```

For our current setup, the internal zone – along with everything that is enabled in it will be the default zone:

```
# firewall-cmd --set-default-zone=internal
```

Next, let's reload firewall rules and keep state information:

```
# firewall-cmd --reload
```

Finally, let's add **router #2** as default gateway in the web server:

```
# ip route add default via 10.0.0.18
```

You can now verify that you can ping **router #1** and an external site (**tecmint.com**, for example) from the web server:

```
# ping -c 2 192.168.0.1
# ping -c 2 tecmint.com
[root@webserver ~]# ping -c 2 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=63 time=1.91 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=63 time=1.12 ms

--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 1.121/1.516/1.912/0.397 ms
[root@webserver ~]# ping -c 2 tecmint.com
PING tecmint.com (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=51 time=323 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=51 time=257 ms

--- tecmint.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1259ms
rtt min/avg/max/mdev = 257.981/290.800/323.619/32.819 ms
[root@webserver ~]# _
```

<http://www.tecmint.com>

## Verify Network Routing

## Setting Kernel Runtime Parameters in RHEL 7

In Linux, you are allowed to change, enable, and disable the kernel runtime parameters, and RHEL is no exception. The **/proc/sys** interface (**sysctl**) lets you set runtime parameters on-the-fly to modify the system's behavior without much hassle when operating conditions change.

To do so, the echo shell built-in is used to write to files inside **/proc/sys/<category>**, where **<category>** is most likely one of the following directories:

1. **dev**: parameters for specific devices connected to the machine.
2. **fs**: filesystem configuration (quotas and inodes, for example).
3. **kernel**: kernel-specific configuration.
4. **net**: network configuration.
5. **vm**: use of the kernel's virtual memory.

To display the list of all the currently available values, run

```
# sysctl -a | less
```

In **Part 1**, we changed the value of the **net.ipv4.ip\_forward** parameter by doing

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

in order to allow a Linux machine to act as router.

Another runtime parameter that you may want to set is **kernel.sysrq**, which enables the **Sysrq** key in your keyboard to instruct the system to perform gracefully some low-level functions, such as rebooting the system if it has frozen for some reason:

```
# echo 1 > /proc/sys/kernel/sysrq
```

To display the value of a specific parameter, use **sysctl** as follows:

```
# sysctl <parameter.name>
```

For example,

```
# sysctl net.ipv4.ip_forward
# sysctl kernel.sysrq
```

Some parameters, such as the ones mentioned above, require only one value, whereas others (for example, **fs.inode-state**) require multiple values:

```
[root@rhel7 ~]# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
[root@rhel7 ~]# sysctl kernel.sysrq
kernel.sysrq = 16
[root@rhel7 ~]# sysctl fs.inode-state
fs.inode-state = 17099 329 0 0 0 0 0
[root@rhel7 ~]#
```

<http://www.tecmint.com>

## Check Kernel Parameters

In either case, you need to read the kernel's documentation before making any changes.

Please note that these settings will go away when the system is rebooted. To make these changes permanent, we will need to add **.conf** files inside the **/etc/sysctl.d** as follows:

```
# echo "net.ipv4.ip_forward = 1" > /etc/sysctl.d/10-forward.conf
```

(where the number **10** indicates the order of processing relative to other files in the same directory).

and enable the changes with

```
# sysctl -p /etc/sysctl.d/10-forward.conf
```

## Summary

In this tutorial we have explained the basics of packet filtering, network address translation, and setting kernel runtime parameters on a running system and persistently across reboots. I hope you have found this information useful, and as always, we look forward to hearing from you!

Don't hesitate to share with us your questions, comments, or suggestions using the form below.

## How to Produce and Deliver System Activity Reports Using Linux Toolsets – Part 3

As a system engineer, you will often need to produce reports that show the utilization of your system's resources in order to make sure that: 1) they are being utilized optimally, 2) prevent bottlenecks, and 3) ensure scalability, among other reasons.

# Red Hat Certified Engineer (RHCE)



## Part 3: How To Produce And Deliver System Activity Reports Using Native Linux tools And the Sysstat / Dstat Toolsets In RHEL 7

### RHCE: Monitor Linux Performance Activity Reports – Part 3

Besides the well-known native Linux tools that are used to check disk, memory, and CPU usage – to name a few examples, Red Hat Enterprise Linux 7 provides two additional toolsets to enhance the data you can collect for your reports: **sysstat** and **dstat**.

In this article we will describe both, but let's first start by reviewing the usage of the classic tools.

#### Native Linux Tools

With **df**, you will be able to report disk space and inode usage of by filesystem. You need to monitor both because a lack of space will prevent you from being able to save further files (and may even cause the system to crash), just like running out of inodes will mean you can't link further files with their corresponding data structures, thus producing the same effect: you won't be able to save those files to disk.

```
# df -h          [Display output in human-readable form]
# df -h --total [Produce a grand total]
```

```
[root@rhel7 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root 28G  8.5G  20G  31% /
devtmpfs        488M   0  488M  0% /dev
tmpfs           497M   0  497M  0% /dev/shm
tmpfs           497M  6.6M  491M  2% /run
tmpfs           497M   0  497M  0% /sys/fs/cgroup
/dev/sda1       497M 191M  307M  39% /boot
[root@rhel7 ~]# df -h --total
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root 28G  8.5G  20G  31% /
devtmpfs        488M   0  488M  0% /dev
tmpfs           497M   0  497M  0% /dev/shm
tmpfs           497M  6.6M  491M  2% /run
tmpfs           497M   0  497M  0% /sys/fs/cgroup
/dev/sda1       497M 191M  307M  39% /boot
total          30G  8.7G  22G  29% -
[root@rhel7 ~]#
```

### Check Linux Total Disk Usage

```
# df -i                      [Show inode count by filesystem]
# df -i --total              [Produce a grand total]
[root@rhel7 ~]# df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
/dev/mapper/rhel-root 28811264 45596 28765668  1% /
devtmpfs        124888   377 124511  1% /dev
tmpfs           127185     1 127184  1% /dev/shm
tmpfs           127185   446 126739  1% /run
tmpfs           127185     13 127172  1% /sys/fs/cgroup
/dev/sda1       512000   342 511658  1% /boot
[root@rhel7 ~]# df -i --total
Filesystem      Inodes IUsed   IFree IUse% Mounted on
/dev/mapper/rhel-root 28811264 45596 28765668  1% /
devtmpfs        124888   377 124511  1% /dev
tmpfs           127185     1 127184  1% /dev/shm
tmpfs           127185   446 126739  1% /run
tmpfs           127185     13 127172  1% /sys/fs/cgroup
/dev/sda1       512000   342 511658  1% /boot
total          29829707 46775 29782932  1% -
[root@rhel7 ~]#
```

### Check Linux Total inode Numbers

With **du**, you can estimate file space usage by either file, directory, or filesystem.

For example, let's see how much space is used by the **/home** directory, which includes all of the user's personal files. The first command will return the overall space currently used by the entire **/home** directory, whereas the second will also display a disaggregated list by sub-directory as well:

```
# du -sch /home
# du -sch /home/*
```

```
[root@rhel7 ~]# du -sch /home
1016K  /home
1016K  total
[root@rhel7 ~]# du -sch /home/*
16K    /home/davenull
972K   /home/gacanepa
16K    /home/johndoe
12K    /home/tecmint
1016K  total
[root@rhel7 ~]#
```

## Check Linux Directory Disk Size

### Don't Miss:

1. [12 ‘df’ Command Examples to Check Linux Disk Space Usage](#)
2. [10 ‘du’ Command Examples to Find Disk Usage of Files/Directories](#)

Another utility that can't be missing from your toolset is **vmstat**. It will allow you to see at a quick glance information about processes, CPU and memory usage, disk activity, and more.

If run without arguments, **vmstat** will return averages since the last reboot. While you may use this form of the command once in a while, it will be more helpful to take a certain amount of system utilization samples, one after another, with a defined time separation between samples.

For example,

```
# vmstat 5 10
```

will return 10 samples taken every 5 seconds:

```
[root@rhel7 ~]# vmstat
procs -----memory----- ---swap-- -----io---- -system- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
2 0 0 562540 764 347968 0 0 62 23 31 47 0 1 99 0 0
[root@rhel7 ~]# vmstat 5 10
procs -----memory----- ---swap-- -----io---- -system- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
2 0 0 562544 764 347968 0 0 62 23 31 47 0 1 99 0 0
0 0 0 562548 764 347968 0 0 0 0 8 10 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 8 11 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 9 10 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 8 10 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 8 10 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 7 11 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 8 10 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 9 13 0 0 100 0 0
0 0 0 562548 764 347968 0 0 0 0 9 11 0 0 100 0 0
[root@rhel7 ~]#
```

## Check Linux System Performance

As you can see in the above picture, the output of vmstat is divided by columns: **procs** (processes), **memory**, **swap**, **io**, **system**, and **cpu**. The meaning of each field can be found in the **FIELD DESCRIPTION** sections in the man page of **vmstat**.

Where can **vmstat** come in handy? Let's examine the behavior of the system before and during a **yum update**:

```
# vmstat -a 1 5
[root@rhel7 ~]# vmstat -a 1 5 BEFORE
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 815660 109212 143544 0 0 89 22 47 81 1 1 98 1 0
0 0 0 815652 109212 143544 0 0 0 0 33 60 0 0 100 0 0
0 0 0 815652 109212 143544 0 0 0 0 30 52 0 0 100 0 0
0 0 0 815652 109212 143544 0 0 0 0 31 54 0 0 100 0 0
0 0 0 815652 109212 143544 0 0 0 0 33 57 0 0 100 0 0
[root@rhel7 ~]# vmstat -a 1 5 DURING
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 454716 106660 296716 0 0 78 55 54 80 1 1 98 1 0
0 0 0 454728 106728 296352 0 0 0 0 249 167 3 2 95 0 0
0 0 0 454316 106890 296372 0 0 0 0 205 111 5 2 93 0 0
0 0 0 454196 107120 296408 0 0 0 0 298 140 5 2 93 0 0
0 0 0 454012 107172 296359 0 0 0 0 318 157 5 4 91 0 0
[root@rhel7 ~]# vmstat -a 1 5
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 453752 107192 296776 0 0 78 55 54 80 1 1 98 1 0
0 0 0 453600 107284 296844 0 0 0 0 213 151 2 2 96 0 0
0 0 0 453412 107304 296872 0 0 0 0 112 90 2 0 98 0 0
2 0 0 469048 106732 282932 0 0 564 1978 615 432 31 16 42 12 0
0 1 0 466784 108028 283844 0 0 84 585 1049 590 66 22 8 5 0
[root@rhel7 ~]# vmstat -a 1 5
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 463480 110432 284380 0 0 79 57 56 81 1 1 98 1 0
0 0 0 460912 110416 284436 0 0 0 0 201 115 6 8 86 0 0
1 0 0 460296 110416 284436 0 0 0 0 356 71 31 2 67 0 0
3 0 0 446368 110416 296852 0 0 4 145 941 204 76 12 10 1 0
0 0 0 515040 110284 230704 0 0 0 252 579 490 19 12 55 4 0
[root@rhel7 ~]# dracut -033-241.el7_1.5.x86_64.rpm | 301 kB 00:00:03
(3/5): dracut-config-rescue-033-241.el7_1.5.x86_64.rpm | 45 kB 00:00:01
(4/5): dracut-network-033-241.el7_1.5.x86_64.rpm | 82 kB 00:00:01
(5/5): bind-libs-lite-9.9.4-18.el7_1.3.x86_64.rpm | 712 kB 00:00:10
Total 116 kB/s | 1.2 MB 00:10
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating : dracut-033-241.el7_1.5.x86_64 1/10
  Updating : 32-bind-license-9.9.4-18.el7_1.3.noarch 2/10
  Updating : 32-bind-libs-lite-9.9.4-18.el7_1.3.x86_64 3/10
  Updating : dracut-config-rescue-033-241.el7_1.5.x86_64 4/10
  Updating : dracut-network-033-241.el7_1.5.x86_64 5/10
  Cleanup : dracut-network-033-241.el7_1.5.x86_64 6/10
  Cleanup : dracut-config-rescue-033-241.el7_1.3.x86_64 7/10
  Cleanup : 32-bind-libs-lite-9.9.4-18.el7_1.2.x86_64 8/10
  Cleanup : 32-bind-license-9.9.4-18.el7_1.3.noarch 9/10
  Cleanup : dracut-033-241.el7_1.3.x86_64 10/10
rhel-7-server-rpm@7Server:x86_64/productid | 1.7 kB 00:00:00
  Verifying : 32-bind-libs-lite-9.9.4-18.el7_1.3.x86_64 1/10
  Verifying : dracut-config-rescue-033-241.el7_1.5.x86_64 2/10
  Verifying : dracut-033-241.el7_1.5.x86_64 3/10
  Verifying : dracut-network-033-241.el7_1.5.x86_64 4/10
  Verifying : 32-bind-license-9.9.4-18.el7_1.3.noarch 5/10
  Verifying : dracut-033-241.el7_1.3.x86_64 6/10
  Verifying : dracut-network-033-241.el7_1.5.x86_64 7/10
  Verifying : 32-bind-libs-lite-9.9.4-18.el7_1.2.x86_64 8/10
  Verifying : dracut-config-rescue-033-241.el7_1.3.x86_64 9/10
  Verifying : 32-bind-license-9.9.4-18.el7_1.2.noarch 10/10
```

## Vmstat Linux Performance Monitoring

Please note that as files are being modified on disk, the amount of **active** memory increases and so does the number of blocks written to disk (**bo**) and the CPU time that is dedicated to user processes (**us**).

Or during the saving process of a large file directly to disk (caused by **dsync**):

```
# vmstat -a 1 5
# dd if=/dev/zero of=dummy.out bs=1M count=1000 oflag=dsync
[root@rhel7 ~]# vmstat -a 1 5 BEFORE
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 443712 109972 297288 0 0 39 46 38 50 1 1 98 0 0
0 0 0 443712 109972 297288 0 0 0 0 11 16 0 0 100 0 0
0 0 0 443696 109972 297308 0 0 0 0 13 15 0 0 100 0 0
0 0 0 443696 109972 297308 0 0 0 0 15 18 0 0 100 0 0
0 0 0 443696 109972 297308 0 0 0 0 15 18 0 0 99 1 0
[root@rhel7 ~]# vmstat -a 1 5
procs      memory----- swap----- io----- system----- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
2 0 0 71596 498928 278364 0 0 39 156 38 51 1 1 98 0 0
2 0 0 69388 512212 270164 0 0 0 3869 627 1128 0 27 65 6 0
1 0 0 67326 531416 255256 0 0 0 92251 689 1387 0 20 70 10 0
0 1 0 66100 566116 221984 0 0 0 69699 537 1018 0 17 75 6 0
0 0 0 81300 566116 207552 0 0 0 16401 223 311 0 11 87 2 0
[root@rhel7 ~]# dd if=/dev/zero of=dummy.out bs=1M count=1000 oflag=dsync
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 12.7199 s, 82.4 MB/s
[root@rhel7 ~]# 
```

## VmStat Linux Disk Performance Monitoring

In this case, we can see a yet larger number of blocks being written to disk (**bo**), which was to be expected, but also an increase of the amount of CPU time that it has to wait for I/O operations to complete before processing tasks (**wa**).

**Don't Miss:** [Vmstat – Linux Performance Monitoring](#)

## Other Linux Tools

As mentioned in the introduction of this chapter, there are other tools that you can use to check the system status and utilization (they are not only provided by **Red Hat** but also by other major distributions from their officially supported repositories).

The **sysstat** package contains the following utilities:

1. **sar** (collect, report, or save system activity information).
2. **sadf** (display data collected by sar in multiple formats).
3. **mpstat** (report processors related statistics).
4. **iostat** (report CPU statistics and I/O statistics for devices and partitions).
5. **pidstat** (report statistics for Linux tasks).
6. **nfsiostat** (report input/output statistics for NFS).
7. **cfsiostat** (report CIFS statistics) and
8. **sa1** (collect and store binary data in the system activity daily data file).
9. **sa2** (write a daily report in the **/var/log/sa** directory) tools.

whereas **dstat** adds some extra features to the functionality provided by those tools, along with more counters and flexibility. You can find an overall description of each tool by running **yum info sysstat** or **yum info dstat**, respectively, or checking the individual man pages after installation.

To install both packages:

```
# yum update && yum install sysstat dstat
```

The main configuration file for **sysstat** is **/etc/sysconfig/sysstat**. You will find the following parameters in that file:

```
# How long to keep log files (in days).  
# If value is greater than 28, then log files are kept in  
# multiple directories, one for each month.  
HISTORY=28  
# Compress (using gzip or bzip2) sa and sar files older than (in days):  
COMPRESSAFTER=31  
# Parameters for the system activity data collector (see sadc manual page)  
# which are used for the generation of log files.  
SADC_OPTIONS="-S DISK"  
# Compression program to use.  
ZIP="bzip2"
```

When **sysstat** is installed, two cron jobs are added and enabled in **/etc/cron.d/sysstat**. The first job runs the system activity accounting tool every **10 minutes** and stores the reports in **/var/log/sa/saXX** where **XX** is the day of the month.

Thus, **/var/log/sa/sa05** will contain all the system activity reports from the 5th of the month. This assumes that we are using the default value in the **HISTORY** variable in the configuration file above:

```
*/10 * * * * root /usr/lib64/sa/sa1 1 1
```

The second job generates a daily summary of process accounting at **11:53 pm** every day and stores it in **/var/log/sa/sarXX** files, where **XX** has the same meaning as in the previous example:

```
53 23 * * * root /usr/lib64/sa/sa2 -A
```

For example, you may want to output system statistics from **9:30 am** through **5:30 pm** of the sixth of the month to a **.csv** file that can easily be viewed using **LibreOffice Calc** or **Microsoft Excel** (this approach will also allow you to create charts or graphs):

```
# sadf -s 09:30:00 -e 17:30:00 -dh /var/log/sa/sa06 -- | sed 's/;/,/g' > system_stats20150806.csv
```

You could alternatively use the **-j** flag instead of **-d** in the **sadf** command above to output the system stats in **JSON** format, which could be useful if you need to consume the data in a web application, for example.

	#	hostname	interval	timestamp	CPU	%user	%nice	%system	%wait	%steal	%idle[...]
1	2	vps		600 2015-08-06 09:40:01 UTC	-2	0.03	0	0.01	0.01	0.03	99.93
2	3	vps		600 2015-08-06 09:50:01 UTC	-3	0.05	0	0.01	0.04	0.01	99.89
3	4	vps		600 2015-08-06 10:00:01 UTC	-2	0.04	0	0.01	0.02	0.03	99.91
4	5	vps		600 2015-08-06 10:10:01 UTC	-2	0.04	0	0.01	0.02	0.03	99.9
5	6	vps		600 2015-08-06 10:20:01 UTC	-2	0.07	0	0.01	0.05	0.03	99.84
6	7	vps		600 2015-08-06 10:30:01 UTC	-3	0.04	0	0.01	0.01	0.02	99.92
7	8	vps		600 2015-08-06 10:40:01 UTC	-3	0.14	0	0.02	0.04	0.04	99.77
8	9	vps		600 2015-08-06 10:50:02 UTC	-2	0.06	0	0.01	0.02	0.02	99.9
9	10	vps		599 2015-08-06 11:00:01 UTC	-2	0.04	0	0.01	0.01	0.02	99.91
10	11	vps		600 2015-08-06 11:10:01 UTC	-3	0.06	0	0.01	0.15	0.03	99.75
11	12	vps		600 2015-08-06 11:20:01 UTC	-3	0.08	0	0.01	0.04	0.02	99.85
12	13	vps		600 2015-08-06 11:30:01 UTC	-2	0.03	0	0.01	0.03	0.02	99.91
13											

## Linux System Statistics

Finally, let's see what **dstat** has to offer. Please note that if run without arguments, **dstat** assumes **-cdnrgy** by default (short for CPU, disk, network, memory pages, and system stats, respectively), and adds one line every second (execution can be interrupted anytime with **Ctrl + C**):

```
# dstat
```

```
[root@vps ~]# dstat
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read writ| recv send| in out| int csw
0 0 100 0 0 0| 670B 1149B| 0 0| 0 0| 36 52
0 0 100 0 0 0| 0 0| 524B 834B| 0 0| 72 91
0 0 100 0 0 0| 0 0| 466B 354B| 0 0| 44 59
0 0 98 1 0 0| 0 0| 36k 328B| 354B| 0 0| 38 60
0 0 100 0 0 0| 0 0| 466B 354B| 0 0| 42 57
0 0 100 0 0 0| 0 0| 374B 354B| 0 0| 45 65
0 0 100 0 0 0| 0 0| 374B 354B| 0 0| 37 57
0 0 100 0 0 0| 0 0| 466B 354B| 0 0| 40 55 ^C
[root@vps ~]#
```

## Linux Disk Statistics Monitoring

To output the stats to a **.csv** file, use the **--output** flag followed by a file name. Let's see how this looks on LibreOffice Calc:

dstat 0.7.2 CSV output											
1	A	B	C	D	E	F	G	H	I	J	K
2	Author:	Doug Weevers < <a href="mailto:doug@weevers.com">doug@weevers.com</a> >					URL:	<a href="http://doug.weevers.co">http://doug.weevers.co</a>			
3	Host:	vps					User:	root			
4	Created:	dstat --output /var/www/gabrielcanepa.com.ar/public_html/dstat.csv					Date:	08 Aug 2015 14:52:21			
5											
6	Total CPU usage										
7	idle	sys	user	wai	hiq	siq	read	recv	send	in	out
8	0.067	0.067	0.012	99.608	0.224	0	0				
9	0	0	0	100	0	0	0	0	0	0	0
10	0	0	0	100	0	0	0	0	0	0	0
11	0	0	0	100	0	0	0	0	0	0	0
12	0	0	0	100	0	0	0	0	0	0	0
13	0	0	0	100	0	0	0	0	0	0	0
14	0	0	0	100	0	0	0	0	0	0	0
15	0	0	0	100	0	0	0	0	0	0	0
16	0	0	0.5	99.5	0	0	0	0	0	0	0
17	0	0	0	100	0	0	0	0	0	0	0
18	0	0	0	100	0	0	0	0	0	0	0

## Monitor Linux Statistics Output

I strongly advise you to check out the man page of [dstat](#) along with the man page of [sysstat](#) in PDF format for your reading convenience. You will find several other options that will help you create custom and detailed system activity reports.

## Don't Miss: [Sysstat – Linux Usage Activity Monitoring Tool](#)

## Summary

In this guide we have explained how to use both native Linux tools and specific utilities provided with **RHEL 7** in order to produce reports on system utilization. At one point or another, you will come to rely on these reports as best friends.

You will probably have used other tools that we have not covered in this tutorial. If so, feel free to share them with the rest of the community along with any other suggestions / questions / comments that you may have- using the form below.

We look forward to hearing from you.

## Using Shell Scripting to Automate Linux System Maintenance Tasks – Part 4

Some time ago I read that one of the distinguishing characteristics of an effective system administrator / engineer is laziness. It seemed a little contradictory at first but the author then proceeded to explain why:



RHCE Series: Automate Linux System Maintenance Tasks – Part 4

if a sysadmin spends most of his time solving issues and doing repetitive tasks, you can suspect he or she is not doing things quite right. In other words, an effective system administrator / engineer should develop a plan to perform repetitive tasks with as less action on his / her part as possible, and should foresee problems by using,

for example, the tools reviewed in Part 3 – [Monitor System Activity Reports Using Linux Toolsets](#) of this series. Thus, although he or she may not seem to be doing much, it's because most of his / her responsibilities have been taken care of with the help of shell scripting, which is what we're going to talk about in this tutorial.

### What is a shell script?

In few words, a shell script is nothing more and nothing less than a program that is executed step by step by a shell, which is another program that provides an interface layer between the Linux kernel and the end user.

By default, the shell used for user accounts in **RHEL 7** is bash (**/bin/bash**). If you want a detailed description and some historical background, you can refer to [this Wikipedia article](#).

To find out more about the enormous set of features provided by this shell, you may want to check out its **man page**, which is downloaded in PDF format at [\(Bash Commands\)](#). Other than that, it is assumed that you are familiar with Linux commands (if not, I strongly advise

you to go through [A Guide from Newbies to SysAdmin](#) article in **Tecmint.com** before proceeding). Now let's get started.

## Writing a script to display system information

For our convenience, let's create a directory to store our shell scripts:

```
# mkdir scripts  
# cd scripts
```

And open a new text file named `system_info.sh` with your preferred text editor. We will begin by inserting a few comments at the top and some commands afterwards:

```
#!/bin/bash

# Sample script written for Part 4 of the RHCE series
# This script will return the following set of system information:
# -Hostname information:
echo -e "\e[31;43m***** HOSTNAME INFORMATION *****\e[0m"
hostnamectl
echo ""
# -File system disk space usage:
echo -e "\e[31;43m***** FILE SYSTEM DISK SPACE USAGE *****\e[0m"
df -h
echo ""
# -Free and used memory in the system:
echo -e "\e[31;43m ***** FREE AND USED MEMORY *****\e[0m"
free
echo ""
# -System uptime and load:
echo -e "\e[31;43m***** SYSTEM UPTIME AND LOAD *****\e[0m"
uptime
echo ""
# -Logged-in users:
echo -e "\e[31;43m***** CURRENTLY LOGGED-IN USERS *****\e[0m"
who
echo ""
# -Top 5 processes as far as memory usage is concerned
echo -e "\e[31;43m***** TOP 5 MEMORY-CONSUMING PROCESSES *****\e[0m"
ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6
echo ""
echo -e "\e[1;32mDone.\e[0m"
```

Next, give the script execute permissions:

```
# chmod +x system_info.sh
```

and run it:

```
./system_info.sh
```

Note that the headers of each section are shown in color for better visualization:

```
[root@rhel7 scripts]# ./system_info.sh http://www.tecmint.com
***** HOSTNAME INFORMATION *****
  Static hostname: rhel7
    Icon name: computer
    Chassis: n/a
  Machine ID: 817a846b23d34dca90b4c8bea548570f
    Boot ID: 91e202c094d8464980a2f3782b82306b
  Virtualization: oracle
Operating System: Red Hat Enterprise Linux
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:GA:server
    Kernel: Linux 3.10.0-229.7.2.el7.x86_64
  Architecture: x86_64

***** FILE SYSTEM DISK SPACE USAGE *****
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root  28G  9.5G  19G  35% /
devtmpfs        488M   0  488M   0% /dev
tmpfs          497M   0  497M   0% /dev/shm
tmpfs          497M  6.6M  491M   2% /run
tmpfs          497M   0  497M   0% /sys/fs/cgroup
/dev/sda1       497M 191M  307M  39% /boot

***** FREE AND USED MEMORY *****
              total     used      free      shared  buff/cache   available
Mem:      1017480    111716    693664       6824    212100     747500
Swap:     2129916       0  2129916

***** SYSTEM UPTIME AND LOAD *****
20:50:33 up  2:31,  2 users,  load average: 0.00, 0.01, 0.05

***** CURRENTLY LOGGED-IN USERS *****

```

## Server Monitoring Shell Script

That functionality is provided by this command:

```
echo -e "\e[COLOR1;COLOR2m<YOUR TEXT HERE>\e[0m"
```

Where **COLOR1** and **COLOR2** are the foreground and background colors, respectively (more info and options are explained in this entry from the [Arch Linux Wiki](#)) and <**YOUR TEXT HERE**> is the string that you want to show in color.

## Automating Tasks

The tasks that you may need to automate may vary from case to case. Thus, we cannot possibly cover all of the possible scenarios in a single article, but we will present three classic tasks that can be automated using shell scripting:

- 1) update the local file database, 2) find (and alternatively delete) files with **777** permissions, and 3) alert when filesystem usage surpasses a defined limit.

Let's create a file named `auto_tasks.sh` in our `scripts` directory with the following content:

```
#!/bin/bash

# Sample script to automate tasks:
# -Update local file database:
```

```

echo -e "\e[4;32mUPDATING LOCAL FILE DATABASE\e[0m"
updatedb
if [ $? == 0 ]; then
    echo "The local file database was updated correctly."
else
    echo "The local file database was not updated correctly."
fi
echo ""

# -Find and / or delete files with 777 permissions.
echo -e "\e[4;32mLOOKING FOR FILES WITH 777 PERMISSIONS\e[0m"
# Enable either option (comment out the other line), but not both.
# Option 1: Delete files without prompting for confirmation. Assumes GNU
version of find.
#find -type f -perm 0777 -delete
# Option 2: Ask for confirmation before deleting files. More portable
across systems.
find -type f -perm 0777 -exec rm -i {} +;
echo ""

# -Alert when file system usage surpasses a defined limit
echo -e "\e[4;32mCHECKING FILE SYSTEM USAGE\e[0m"
THRESHOLD=30
while read line; do
    # This variable stores the file system path as a string
    FILESYSTEM=$(echo $line | awk '{print $1}')
    # This variable stores the use percentage (XX%)
    PERCENTAGE=$(echo $line | awk '{print $5}')
    # Use percentage without the % sign.
    USAGE=${PERCENTAGE%?}
    if [ $USAGE -gt $THRESHOLD ]; then
        echo "The remaining available space in $FILESYSTEM is
critically low. Used: $PERCENTAGE"
    fi
done < <(df -h --total | grep -vi filesystem)

```

Please note that there is a space between the two < signs in the last line of the script.

```

[root@rhel7 scripts]# ./auto_tasks.sh
UPDATING LOCAL FILE DATABASE
The local file database was updated correctly.

LOOKING FOR FILES WITH 777 PERMISSIONS
rm: remove regular empty file './file.txt'? y

CHECKING FILE SYSTEM USAGE
The remaining available space in /dev/mapper/rhel-root is critically low. Used: 35%
The remaining available space in /dev/sda1 is critically low. Used: 39%
The remaining available space in total is critically low. Used: 33%
[root@rhel7 scripts]#

```

Shell Script to Find 777 Permissions

## Using Cron

To take efficiency one step further, you will not want to sit in front of your computer and run those scripts manually. Rather, you will use **cron** to schedule those tasks to run on a periodic

basis and sends the results to a predefined list of recipients via email or save them to a file that can be viewed using a web browser.

The following script (filesystem\_usage.sh) will run the well-known **df -h** command, format the output into a HTML table and save it in the **report.html** file:

```
#!/bin/bash
# Sample script to demonstrate the creation of an HTML report using shell
scripting
# Web directory
WEB_DIR=/var/www/html
# A little CSS and table layout to make the report look a little nicer
echo "<HTML>
<HEAD>
<style>
.titulo{font-size: 1em; color: white; background:#0863CE; padding: 0.1em
0.2em;}
table
{
border-collapse:collapse;
}
table, td, th
{
border:1px solid black;
}
</style>
<meta http-equiv='Content-Type' content='text/html; charset=UTF-8' />
</HEAD>
<BODY>" > $WEB_DIR/report.html
# View hostname and insert it at the top of the html body
HOST=$(hostname)
echo "Filesystem usage for host <strong>$HOST</strong><br>
Last updated: <strong>$(date)</strong><br><br>
<table border='1'>
<tr><th class='titulo'>Filesystem</td>
<th class='titulo'>Size</td>
<th class='titulo'>Use %</td>
</tr>" >> $WEB_DIR/report.html
# Read the output of df -h line by line
while read line; do
echo "<tr><td align='center'>" >> $WEB_DIR/report.html
echo $line | awk '{print $1}' >> $WEB_DIR/report.html
echo "</td><td align='center'>" >> $WEB_DIR/report.html
echo $line | awk '{print $2}' >> $WEB_DIR/report.html
echo "</td><td align='center'>" >> $WEB_DIR/report.html
echo $line | awk '{print $5}' >> $WEB_DIR/report.html
echo "</td></tr>" >> $WEB_DIR/report.html
done < <(df -h | grep -vi filesystem)
echo "</table></BODY></HTML>" >> $WEB_DIR/report.html
```

In our **RHEL 7** server (**192.168.0.18**), this looks as follows:

Filesystem usage for host **rhel7**  
Last updated: **Tue Aug 11 11:19:34 EDT 2015**

Filesystem	Size	Use %
/dev/mapper/rhel-root	28G	35%
devtmpfs	488M	0%
tmpfs	497M	0%
tmpfs	497M	2%
tmpfs	497M	0%
/dev/sda1	497M	39%

## Server Monitoring Report

You can add to that report as much information as you want. To run the script every day at **1:30 pm**, add the following crontab entry:

```
30 13 * * * /root/scripts/filesystem_usage.sh
```

## Summary

You will most likely think of several other tasks that you want or need to automate; as you can see, using shell scripting will greatly simplify this effort. Feel free to let us know if you find this article helpful and don't hesitate to add your own ideas or comments via the form below.

## [How to Manage System Logs \(Configure, Rotate and Import Into Database\) in RHEL 7 – Part 5](#)

In order to keep your **RHEL 7** systems secure, you need to know how to monitor all of the activities that take place on such systems by examining log files. Thus, you will be able to detect any unusual or potentially malicious activity and perform system troubleshooting or take another appropriate action.

# Red Hat Certified Engineer (RHCE)

<http://www.tecmint.com>



## Part 5: How To Manage System Logs (Configure, Rotate, and Import Into Database) In RHEL 7

RHCE Exam: Manage System Logs Using Rsyslogd and Logrotate – Part 5

In **RHEL 7**, the [rsyslogd](#) daemon is responsible for system logging and reads its configuration from **/etc/rsyslog.conf** (this file specifies the default location for all system logs) and from files inside **/etc/rsyslog.d**, if any.

### Rsyslog Configuration

A quick inspection of the [rsyslog.conf](#) will be helpful to start. This file is divided into 3 main sections: **Modules** (since rsyslog follows a modular design), **Global** directives (used to set global properties of the rsyslogd daemon), and **Rules**. As you will probably guess, this last section indicates what gets logged or shown (also known as the selector) and where, and will be our focus throughout this article.

A typical line in **rsyslog.conf** is as follows:



### Rsyslog Configuration

In the image above, we can see that a selector consists of one or more pairs **Facility:Priority** separated by semicolons, where Facility describes the type of message (refer to [section 4.1.1 in RFC 3164](#) to see the complete list of facilities available for rsyslog) and Priority indicates its severity, which can be one of the following self-explanatory words:

1. debug

2. info
3. notice
4. warning
5. err
6. crit
7. alert
8. emerg

Though not a priority itself, the keyword **none** means no priority at all of the given facility.

**Note:** That a given priority indicates that all messages of such priority and above should be logged. Thus, the line in the example above instructs the **rsyslogd** daemon to log all messages of priority info or higher (regardless of the facility) except those belonging to **mail**, **authpriv**, and **cron** services (no messages coming from this facilities will be taken into account) to **/var/log/messages**.

You can also group multiple facilities using the colon sign to apply the same priority to all of them. Thus, the line:

```
*.info;mail.none;authpriv.none;cron.none           /var/log/messages
```

Could be rewritten as

```
*.info;mail,authpriv,cron.none           /var/log/messages
```

In other words, the facilities **mail**, **authpriv**, and **cron** are grouped and the keyword **none** is applied to the three of them.

### Creating a custom log file

To log all daemon messages to **/var/log/tecmint.log**, we need to add the following line either in **rsyslog.conf** or in a separate file (easier to manage) inside **/etc/rsyslog.d**:

```
daemon.*      /var/log/tecmint.log
```

Let's restart the daemon (note that the service name does not end with a **d**):

```
# systemctl restart rsyslog
```

And check the contents of our custom log before and after restarting two random daemons:

```
[root@rhel7 ~]# cat /etc/rsyslog.d/tecmint.conf
daemon.*          /var/log/tecmint.log
[root@rhel7 ~]# cat /var/log/tecmint.log
[root@rhel7 ~]# systemctl stop httpd
[root@rhel7 ~]# systemctl stop firewalld
[root@rhel7 ~]# systemctl start httpd
[root@rhel7 ~]# systemctl start firewalld
[root@rhel7 ~]# cat /var/log/tecmint.log
Aug 20 20:20:57 rhel7 systemd: Stopping The Apache HTTP Server...
Aug 20 20:20:58 rhel7 systemd: Stopped The Apache HTTP Server.
Aug 20 20:21:05 rhel7 systemd: Stopping firewalld - dynamic firewall daemon...
Aug 20 20:21:06 rhel7 systemd: Stopped firewalld - dynamic firewall daemon.
Aug 20 20:21:12 rhel7 systemd: Starting The Apache HTTP Server...
Aug 20 20:21:12 rhel7 systemd: Started The Apache HTTP Server.
Aug 20 20:21:18 rhel7 systemd: Starting firewalld - dynamic firewall daemon...
Aug 20 20:21:18 rhel7 systemd: Started firewalld - dynamic firewall daemon.
[root@rhel7 ~]#
```

The log is initially empty

Stop and start the firewall and web services

The log now contains information about the daemon status history.

<http://www.tecmint.com>

## Create Custom Log File

As a self-study exercise, I would recommend you play around with the facilities and priorities and either log additional messages to existing log files or create new ones as in the previous example.

## Rotating Logs using Logrotate

To prevent log files from growing endlessly, the **logrotate** utility is used to rotate, compress, remove, and alternatively mail logs, thus easing the administration of systems that generate large numbers of log files.

**Logrotate** runs daily as a **cron** job (`/etc/cron.daily/logrotate`) and reads its configuration from `/etc/logrotate.conf` and from files located in `/etc/logrotate.d`, if any.

As with the case of **rsyslog**, even when you can include settings for specific services in the main file, creating separate configuration files for each one will help organize your settings better.

Let's take a look at a typical `logrotate.conf`:

```

# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4                                → Global settings
                                         (apply to all services
                                         that are logged)

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# uncomment this if you want your log files compressed
#compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}

```

<http://www.tecmint.com>

## Logrotate Configuration

In the example above, **logrotate** will perform the following actions for **/var/loh/wtmp**: attempt to rotate only once a month, but only if the file is at least **1 MB** in size, then create a brand new log file with permissions set to **0664** and ownership given to user **root** and group **utmp**. Next, only keep one archived log, as specified by the **rotate** directive:

```

[root@rhel7 ~]# ls -lh /var/log | grep wtmp
-rw-rw-r--. 1 root utmp 264K Aug 20 20:20 wtmp

```

<http://www.tecmint.com>

## Logrotate Logs Monthly

Let's now consider another example as found in **/etc/logrotate.d/httpd**:

The left terminal window shows the contents of /etc/logrotate.d/httpd:

```
[root@rhel7 ~]# cat /etc/logrotate.d/httpd
/var/log/httpd/*log {
    missingok
   notifempty
    sharedscripts
    delaycompress
    postrotate
        /bin/systemctl reload httpd.service > /dev/null
    endscript
}
```

Annotations explain: "These settings apply to all logs located within /var/log/httpd", "Action to be performed after rotating these logs".

The right terminal window shows the output of ls -l /var/log/httpd:

```
[root@rhel7 ~]# ls -l /var/log/httpd
total 56
-rw-r--r--. 1 root root 1380 Aug 20 20:20 access_log
-rw-r--r--. 1 root root 3863 Jul 23 20:40 access_log-20150723
-rw-r--r--. 1 root root 6210 Jul 27 18:31 access_log-20150727
-rw-r--r--. 1 root root 2947 Aug 3 20:43 access_log-20150803
-rw-r--r--. 1 root root 6157 Aug 20 19:38 access_log-20150820
-rw-r--r--. 1 root root 2278 Aug 20 20:21 error_log
-rw-r--r--. 1 root root 2395 Jul 23 20:40 error_log-20150723
-rw-r--r--. 1 root root 7705 Jul 27 18:31 error_log-20150727
-rw-r--r--. 1 root root 3280 Aug 3 20:43 error_log-20150803
-rw-r--r--. 1 root root 7588 Aug 20 19:38 error_log-20150820
[root@rhel7 ~]#
```

Annotation: "http://www.tecmint.com"

## Rotate Apache Log Files

You can read more about the settings for **logrotate** in its man pages ([man logrotate](#) and [man logrotate.conf](#)). Both files are provided along with this article in PDF format for your reading convenience.

As a system engineer, it will be pretty much up to you to decide for how long logs will be stored and in what format, depending on whether you have **/var** in a separate partition / logical volume. Otherwise, you really want to consider removing old logs to save storage space. On the other hand, you may be forced to keep several logs for future security auditing according to your company's or client's internal policies.

## Saving Logs to a Database

Of course examining logs (even with the help of tools such as **grep** and regular expressions) can become a rather tedious task. For that reason, **rsyslog** allows us to export them into a database (**OTB** supported RDBMS include MySQL, MariaDB, PostgreSQL, and Oracle).

This section of the tutorial assumes that you have already installed the **MariaDB** server and client in the same RHEL 7 box where the logs are being managed:

```
# yum update && yum install mariadb mariadb-server mariadb-client rsyslog-
mysql
# systemctl enable mariadb && systemctl start mariadb
```

Then use the **mysql\_secure\_installation** utility to set the password for the root user and other security considerations:

Annotations explain: "Leave blank and press Enter this time", "Enter a password of your choosing", "Then continue with the execution of mysql\_secure\_installation".

```
Enter current password for root (enter for none): Leave blank and press OK, successfully used password, moving on... Enter this time

Setting the root password ensures that nobody can log into the MariaDB root user without the proper authorisation.

Set root password? [Y/n] y → Enter a password of your choosing
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables...
... Success!
```

Annotation: "Then continue with the execution of mysql\_secure\_installation"

Annotation: "http://www.tecmint.com"

## Secure MySQL Database

**Note:** If you don't want to use the **MariaDB** root user to insert log messages to the database, you can configure another user account to do so. Explaining how to do that is out of the scope of this tutorial but is explained in detail in [MariaDB knowledge base](#). In this tutorial we will use the root account for simplicity.

Next, download the **createDB.sql** script from [GitHub](#) and import it into your database server:

```
# mysql -u root -p < createDB.sql
[root@rhel7 ~]# mysql -u root -p < createDB.sql
Enter password:
[root@rhel7 ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 5.5.41-MariaDB MariaDB Server

Copyright (c) 2000, 2014, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE Syslog;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [Syslog]> SHOW TABLES;
+-----+
| Tables_in_Syslog      |
+-----+
| SystemEvents          |
| SystemEventsProperties |
+-----+
2 rows in set (0.00 sec)

MariaDB [Syslog]> █
```

<http://www.tecmint.com>

## Save Server Logs to Database

Finally, add the following lines to **/etc/rsyslog.conf**:

```
$ModLoad ommysql
$ActionOmmysqlServerPort 3306
*.* :ommysql:localhost,Syslog,root,YourPasswordHere
```

Restart **rsyslog** and the database server:

```
# systemctl restart rsyslog
# systemctl restart mariadb
```

## Querying the Logs using SQL syntax

Now perform some tasks that will modify the logs (like stopping and starting services, for example), then log to your DB server and use standard SQL commands to display and search in the logs:

```
USE Syslog;
SELECT ReceivedAt, Message FROM SystemEvents;

MariaDB [Syslog]> SELECT ReceivedAt, Message FROM SystemEvents;
+-----+-----+
| ReceivedAt      | Message
+-----+-----+
| 2015-08-21 10:15:20 | [origin software="rsyslogd" swVersion="7.4.7" x-pid="4010" x-i...
| 2015-08-21 10:15:20 | Stopping System Logging Service...
| 2015-08-21 10:15:20 | Starting System Logging Service...
| 2015-08-21 10:15:20 | Started System Logging Service.
| 2015-08-21 10:15:38 | Stopping MariaDB database server...
| 2015-08-21 10:20:01 | Created slice user-0.slice.
| 2015-08-21 10:20:01 | Starting Session 29 of user root.
| 2015-08-21 10:20:01 | Started Session 29 of user root.
| 2015-08-21 10:20:01 | (root) CMD (/usr/lib64/sa/sa1 1 1)
+-----+
9 rows in set (0.00 sec)
```

<http://www.tecmint.com>

## Query Logs in Database

## Summary

In this article we have explained how to set up system logging, how to rotate logs, and how to redirect the messages to a database for easier search. We hope that these skills will be helpful as you prepare for the [RHCE exam](#) and in your daily responsibilities as well.

As always, your feedback is more than welcome. Feel free to use the form below to reach us.

## [Setting Up Samba and Configure FirewallD and SELinux to Allow File Sharing on Linux/Windows Clients – Part 6](#)

Since computers seldom work as isolated systems, it is to be expected that as a system administrator or engineer, you know how to set up and maintain a network with multiple types of servers.

In this article and in the next of this series we will go through the essentials of setting up **Samba** and **NFS** servers with Windows/Linux and Linux clients, respectively.

# Red Hat Certified Engineer (RHCE)

Tecmint  
Linux Howto's Guide



## Part 6: How to set up a Samba server and configure firewalld and SELinux to allow file sharing In RHEL 7

### RHCE: Setup Samba File Sharing – Part 6

This article will definitely come in handy if you're called upon to set up file servers in corporate or enterprise environments where you are likely to find different operating systems and types of devices.

Since you can read about the background and the technical aspects of both Samba and NFS all over the Internet, in this article and the next we will cut right to the chase with the topic at hand.

#### Step 1: Installing Samba Server

Our current testing environment consists of two **RHEL 7 boxes** and one **Windows 8 machine**, in that order:

1. Samba / NFS server [box1 (RHEL 7): 192.168.0.18],
2. Samba client #1 [box2 (RHEL 7): 192.168.0.20]
3. Samba client #2 [Windows 8 machine: 192.168.0.106]

The screenshot shows two terminal windows. The left window is titled 'root@box1:~ 82x41' and displays the command 'cat /etc/redhat-release' followed by the output 'Red Hat Enterprise Linux Server release 7.1 (Maipo)'. The right window is titled 'root@box2:~ 82x41' and also displays the command 'cat /etc/redhat-release' followed by the same output 'Red Hat Enterprise Linux Server release 7.1 (Maipo)'.

#### Testing Setup for Samba

On **box1**, install the following packages:

```
# yum update && yum install samba samba-client samba-common
```

On **box2**:

```
# yum update && yum install samba samba-client samba-common cifs-utils
```

Once the installation is complete, we're ready to configure our share.

## Step 2: Setting Up File Sharing Through Samba

One of the reason why **Samba** is so relevant is because it provides file and print services to **SMB/CIFS** clients, which causes those clients to see the server as if it was a Windows system (I must admit I tend to get a little emotional while writing about this topic as it was my first setup as a new Linux system administrator some years ago).

[Adding system users and setting up permissions and ownership](#)

To allow for group collaboration, we will create a group named **finance** with two users (**user1** and **user2**) with [useradd command](#) and a directory **/finance** in **box1**.

We will also change the group owner of this directory to **finance** and set its permissions to **0770** (read, write, and execution permissions for the owner and the group owner):

```
# groupadd finance
# useradd user1
# useradd user2
# usermod -a -G finance user1
# usermod -a -G finance user2
# mkdir /finance
# chmod 0770 /finance
# chgrp finance /finance
```

## Step 3: Configuring SELinux and Firewalld

In preparation to configure **/finance** as a Samba share, we will need to either disable **SELinux** or set the proper **boolean** and security context values as follows (otherwise, SELinux will prevent clients from accessing the share):

```
# setsebool -P samba_export_all_ro=1 samba_export_all_rw=1
# getsebool -a | grep samba_export
# semanage fcontext -at samba_share_t "/finance(/.*)?"
# restorecon /finance
```

In addition, we must ensure that Samba traffic is allowed by the [firewalld](#).

```
# firewall-cmd --permanent --add-service=samba
# firewall-cmd --reload
```

## Step 4: Configure Samba Share

Now it's time to dive into the configuration file **/etc/samba/smb.conf** and add the section for our share: we want the members of the **finance** group to be able to browse the contents of **/finance**, and save / create files or subdirectories in it (which by default will have their permission bits set to **0770** and **finance** will be their group owner):

**smb.conf**

```
[finance]
comment=Directory for collaboration of the company's finance team
browsable=yes
path=/finance
```

```
public=no
valid users=@finance
write list=@finance
writeable=yes
create mask=0770
Force create mode=0770
force group=finance
```

Save the file and then test it with the **testparm** utility. If there are any errors, the output of the following command will indicate what you need to fix. Otherwise, it will display a review of your Samba server configuration:

```
[root@box1 ~]# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Processing section "[homes]"
Processing section "[printers]"
Processing section "[finance]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
```

## Test Samba Configuration

Should you want to add another share that is open to the public (meaning without any authentication whatsoever), create another section in **/etc/samba/smb.conf** and under the new share's name copy the section above, only changing **public=no** to **public=yes** and not including the valid users and write list directives.

## Step 5: Adding Samba Users

Next, you will need to add **user1** and **user2** as Samba users. To do so, you will use the **smbpasswd** command, which interacts with Samba's internal database. You will be prompted to enter a password that you will later use to connect to the share:

```
# smbpasswd -a user1
# smbpasswd -a user2
```

Finally, restart **Samba**, enable the service to start on boot, and make sure the share is actually available to network clients:

```
# systemctl start smb
# systemctl enable smb
# smbclient -L localhost -U user1
# smbclient -L localhost -U user2
```

```
[root@box1 ~]# smbclient -L localhost -U user1
Enter user1's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Sharename      Type      Comment
      -----      ----      -----
      finance       Disk      Directory for collaboration of the company's finance team
      IPC$         IPC       IPC Service (Samba Server Version 4.1.12)
      user1        Disk      Home Directories
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Server          Comment
      -----
      Workgroup      Master
      -----
[root@box1 ~]# smbclient -L localhost -U user2
Enter user2's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Sharename      Type      Comment
      -----      ----      -----
      finance       Disk      Directory for collaboration of the company's finance team
      IPC$         IPC       IPC Service (Samba Server Version 4.1.12)
      user2        Disk      Home Directories
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Server          Comment
      -----
      Workgroup      Master
      -----
[root@box1 ~]#
```

## Verify Samba Share

At this point, the Samba file server has been properly installed and configured. Now it's time to test this setup on our **RHEL 7** and **Windows 8** clients.

### Step 6: Mounting the Samba Share in Linux

First, make sure the Samba share is accessible from this client:

```
# smbclient -L 192.168.0.18 -U user2
[root@box2 ~]# smbclient -L 192.168.0.18 -U user2
Enter user2's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Sharename      Type      Comment
      -----      ----      -----
      IPC$         IPC       IPC Service (Samba Server Version 4.1.12)
      finance       Disk      Directory for collaboration of the company's finance team
      user2        Disk      Home Directories
Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.12]

      Server          Comment
      -----
      Workgroup      Master
      -----
[root@box2 ~]#
```

## Mount Samba Share on Linux

(repeat the above command for user1)

As any other storage media, you can mount (and later unmount) this network share when needed:

```
# mount //192.168.0.18/finance /media/samba -o username=user1
[root@box2 ~]# mount | grep finance → The Samba share has not been mounted yet
[root@box2 ~]# mount //192.168.0.18/finance /media/samba -o username=user1
Password for user1@//192.168.0.18/finance: ***** → Enter the password for user1
[root@box2 ~]# mount | grep finance
//192.168.0.18/Finance on /media/samba type cifs (rw,relatime,vers=1.0,cache=strict,username=user1,do
posixpaths,serverino,acl,rsize=1048576,wsize=65536,actimeo=1) → The Samba share is now mounted
[root@box2 ~]#
```

## Mount Samba Network Share

(where **/media/samba** is an existing directory)

or permanently, by adding the following entry in **/etc/fstab** file:

**fstab**

```
//192.168.0.18/finance /media/samba cifs
credentials=/media/samba/.smbcredentials,defaults 0 0
```

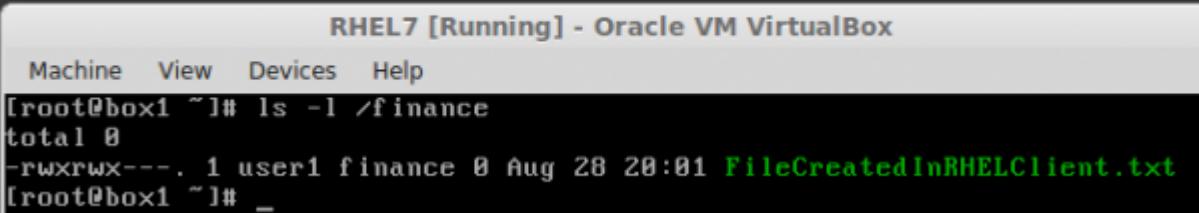
Where the hidden file **/media/samba/.smbcredentials** (whose permissions and ownership have been set to **600** and **root:root**, respectively) contains two lines that indicate the username and password of an account that is allowed to use the share:

**.smbcredentials**

```
username=user1
password=PasswordForUser1
```

Finally, let's create a file inside **/finance** and check the permissions and ownership:

```
# touch /media/samba/FileCreatedInRHELClient.txt
[root@box2 ~]# touch /media/samba/FileCreatedInRHELClient.txt
[root@box2 ~]# 
```



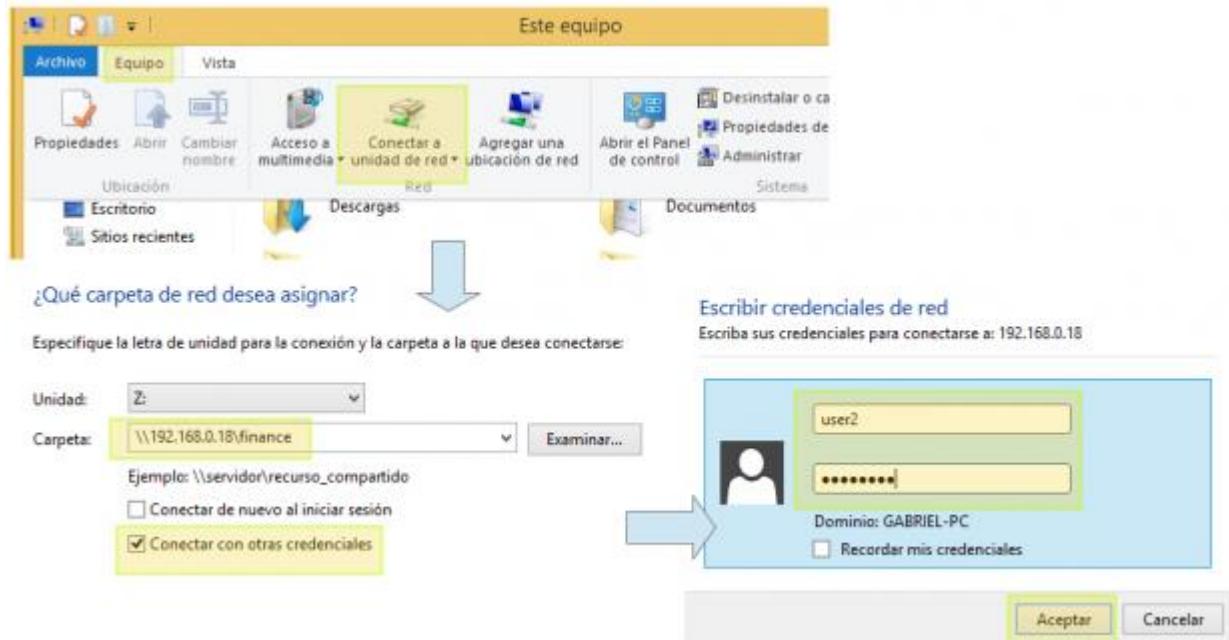
```
RHEL7 [Running] - Oracle VM VirtualBox
Machine View Devices Help
[root@box1 ~]# ls -l /finance
total 0
-rwxrwx--- 1 user1 finance 0 Aug 28 20:01 FileCreatedInRHELClient.txt
[root@box1 ~]# _
```

## Create File in Samba Share

As you can see, the file was created with **0770** permissions and ownership set to **user1:finance**.

## Step 7: Mounting the Samba Share in Windows

To mount the Samba share in Windows, go to **My PC** and choose **Computer**, then **Map network drive**. Next, assign a letter for the drive to be mapped and check Connect using different credentials (the screenshots below are in Spanish, my native language):



### Mount Samba Share in Windows

Finally, let's create a file and check the permissions and ownership:

Nombre	Fecha de modifica...	Tipo	Tamaño
FileCreatedInRHELClient	28/08/2015 09:01 ...	Documento de tex...	0 KB
FileCreatedInWindowsClient	28/08/2015 09:27 ...	Documento de tex...	0 KB

```
[root@box1 ~]# ls -l /finance
total 0
-rwxrwx---. 1 user1 finance 0 Aug 28 20:01 FileCreatedInRHELClient.txt
-rwxrwx---. 1 user2 finance 0 Aug 28 20:27 FileCreatedInWindowsClient.txt
[root@box1 ~]#
```

```
# ls -l /finance
```

This time the file belongs to **user2** since that's the account we used to connect from the Windows client.

## Summary

In this article we have explained not only how to set up a **Samba** server and two clients using different operating systems, but also [how to configure the firewalld](#) and [SELinux on the server](#) to allow the desired group collaboration capabilities.

Last, but not least, let me recommend the reading of the online [man page of smb.conf](#) to explore other configuration directives that may be more suitable for your case than the scenario described in this article.

As always, feel free to drop a comment using the form below if you have any comments or suggestions.

## [Setting Up NFS Server with Kerberos-based Authentication for Linux Clients – Part 7](#)

In the last article of this series, we reviewed [how to set up a Samba share over a network](#) that may consist of multiple types of operating systems. Now, if you need to set up file sharing for a group of Unix-like clients you will automatically think of the **Network File System**, or **NFS** for short.



RHCE Series: Setting Up NFS Server with Kerberos Authentication – Part 7

In this article we will walk you through the process of using **Kerberos-based** authentication for **NFS** shares. It is assumed that you already have set up a NFS server and a client. If not, please refer to [install and configure NFS server](#) – which will list the necessary packages that need to be installed and explain how to perform initial configurations on the server before proceeding further.

In addition, you will want to configure both [SELinux](#) and [firewalld](#) to allow for file sharing through NFS.

The following example assumes that your **NFS** share is located in **/nfs** in **box2**:

```
# semanage fcontext -a -t public_content_rw_t "/nfs(/.*)?"  
# restorecon -R /nfs  
# setsebool -P nfs_export_all_rw on  
# setsebool -P nfs_export_all_ro on
```

(where the **-P** flag indicates persistence across reboots).

Finally, don't forget to:

### Create NFS Group and Configure NFS Share Directory

1. Create a group called **nfs** and add the **nfsnobody** user to it, then change the permissions of the **/nfs** directory to **0770** and its group owner to **nfs**. Thus, **nfsnobody** (which is mapped to the client requests) will have write permissions on the share) and you won't need to use **no\_root\_squash** in the **/etc(exports** file.

```
# groupadd nfs  
# usermod -a -G nfs nfsnobody  
# chmod 0770 /nfs  
# chgrp nfs /nfs
```

2. Modify the exports file (**/etc(exports**) as follows to only allow access from **box1** using **Kerberos** security (**sec=krb5**).

**Note:** that the value of **anongid** has been set to the **GID** of the **nfs** group that we created previously:

```
exports – Add NFS Share  
  
/nfs box1(rw,sec=krb5,anongid=1004)
```

3. Re-export (**-r**) all (**-a**) the NFS shares. Adding verbosity to the output (**-v**) is a good idea since it will provide helpful information to troubleshoot the server if something goes wrong:

```
# exportfs -arv
```

4. Restart and enable the NFS server and related services. Note that you don't have to enable **nfs-lock** and **nfs-idmapd** because they will be automatically started by the other services on boot:

```
# systemctl restart rpcbind nfs-server nfs-lock nfs-idmap  
# systemctl enable rpcbind nfs-server
```

### Testing Environment and Other Prerequisites

In this guide we will use the following test environment:

1. Client machine [**box1: 192.168.0.18**]
2. NFS / Kerberos server [**box2: 192.168.0.20**] (also known as **Key Distribution Center**, or **KDC** for short).

**Note:** that **Kerberos** service is crucial to the authentication scheme.

As you can see, the **NFS** server and the **KDC** are hosted in the same machine for simplicity, although you can set them up in separate machines if you have more available. Both machines are members of the `mydomain.com` domain.

Last but not least, **Kerberos** requires at least a basic schema of name resolution and the [Network Time Protocol](#) service to be present in both client and server since the security of Kerberos authentication is in part based upon the timestamps of tickets.

To set up name resolution, we will use the `/etc/hosts` file in both client and server:

host file – Add DNS for Domain

```
192.168.0.18      box1.mydomain.com      box1
192.168.0.20      box2.mydomain.com      box2
```

In **RHEL 7**, **chrony** is the default software that is used for **NTP** synchronization:

```
# yum install chrony
# systemctl start chronyd
# systemctl enable chronyd
```

To make sure **chrony** is actually synchronizing your system's time with time servers you may want to issue the following command two or three times and make sure the offset is getting nearer to zero:

```
# chronyc tracking
```

```
[root@box1 ~]# chronyc tracking
Reference ID      : 66.60.22.202 (argentino.microsulesybernabo.com.ar)
Stratum          : 5
Ref time (UTC)   : Fri Sep  4 00:18:34 2015
System time      : 0.000058842 seconds slow of NTP time
Last offset      : -0.043042619 seconds
RMS offset       : 0.043042619 seconds
Frequency        : 107.031 ppm slow
Residual freq    : -19.210 ppm
Skew             : 528.732 ppm
Root delay       : 0.056379 seconds
Root dispersion  : 0.012710 seconds
Update interval  : 65.2 seconds
Leap status      : Normal
```

1

```
[root@box1 ~]# chronyc tracking
Reference ID      : 66.60.22.202 (argentino.microsulesybernabo.com.ar)
Stratum          : 5
Ref time (UTC)   : Fri Sep  4 00:37:58 2015
System time      : 0.008076809 seconds fast of NTP time
Last offset      : 0.007414413 seconds
RMS offset       : 0.023549324 seconds
Frequency        : 58.135 ppm fast
Residual freq    : 1.626 ppm
Skew             : 103.616 ppm
Root delay       : 0.162798 seconds
Root dispersion  : 0.039265 seconds
Update interval  : 257.2 seconds
Leap status      : Normal
```

2

<http://www.tecmint.com>

Synchronize Server Time with Chrony

## Installing and Configuring Kerberos

To set up the **KDC**, install the following packages on both **server** and **client** (omit the server package in the client):

```
# yum update && yum install krb5-server krb5-workstation pam_krb5
```

Once it is installed, edit the configuration files (**/etc/krb5.conf** and **/var/kerberos/krb5kdc/kadm5.acl**) and replace all instances of **example.com** (lowercase and uppercase) with `mydomain.com` as follows.

Now create the **Kerberos** database (please note that this may take a while as it requires a some level of entropy in your system. To speed things up, I opened another terminal and ran **ping -f localhost** for 30-45 seconds):

```
# kdb5_util create -s
[root@box2 ~]# kdb5_util create -s                                         http://www.tecmint.com
Loading random data
Initializing database '/var/kerberos/krb5kdc/principal' for realm 'MYDOMAIN.COM',
master key name 'K/M@MYDOMAIN.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:                                         Enter password twice
Re-enter KDC database master key to verify:
[root@box2 ~]#
```

Create Kerberos Database

Next, enable **Kerberos** through the **firewall** and start / enable the related services.

**Important:** **nfs-secure** must be started and enabled on the client as well:

```
# firewall-cmd --permanent --add-service=kerberos
# systemctl start krb5kdc kadmin nfs-secure
# systemctl enable krb5kdc kadmin nfs-secure
```

Next, using the **kadmin.local** tool, create an admin principal for root:

```
# kadmin.local
# addprinc root/admin
```

And add the **Kerberos** server to the database:

```
# addprinc -randkey host/box2.mydomain.com
```

Same with the **NFS** service for both client (**box1**) and server (**box2**). Please note that in the screenshot below I forgot to do it for **box1** before quitting:

```
# addprinc -randkey nfs/box2.mydomain.com
# addprinc -randkey nfs/box1.mydomain.com
```

And **exit** by typing **quit** and pressing Enter:

```
[root@box2 ~]# kdb5_util create -s
Loading random data
Initializing database '/var/kerberos/krb5kdc/principal' for realm 'MYDOMAIN.COM',
master key name 'K/M@MYDOMAIN.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: Enter password twice
Re-enter KDC database master key to verify: Enter password twice
[root@box2 ~]# kadmin.local
Authenticating as principal root/admin@MYDOMAIN.COM with password.
kadmin.local: addprinc root/admin
WARNING: no policy specified for root/admin@MYDOMAIN.COM; defaulting to no policy
Enter password for principal "root/admin@MYDOMAIN.COM": Enter password twice
Re-enter password for principal "root/admin@MYDOMAIN.COM":
Principal "root/admin@MYDOMAIN.COM" created.
kadmin.local: addprinc -randkey host/box2.mydomain.com
WARNING: no policy specified for host/box2.mydomain.com@MYDOMAIN.COM; defaulting to no policy
Principal "host/box2.mydomain.com@MYDOMAIN.COM" created.
kadmin.local: addprinc -randkey nfs/box2.mydomain.com
WARNING: no policy specified for nfs/box2.mydomain.com@MYDOMAIN.COM; defaulting to no policy
Principal "nfs/box2.mydomain.com@MYDOMAIN.COM" created.
kadmin.local: quit
[root@box2 ~]# http://www.tecmint.com
```

## Add Kerberos to NFS Server

Then obtain and cache Kerberos ticket-granting ticket for root/admin:

```
# kinit root/admin
# klist
[root@box2 ~]# kinit root/admin
Password for root/admin@MYDOMAIN.COM:
[root@box2 ~]# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: root/admin@MYDOMAIN.COM

Valid starting     Expires            Service principal
09/05/2015 14:08:42  09/06/2015 14:08:42  krbtgt/MYDOMAIN.COM@MYDOMAIN.COM
[root@box2 ~]# http://www.tecmint.com
```

## Cache Kerberos

The last step before actually using **Kerberos** is storing into a **keytab** file (in the server) the principals that are authorized to use Kerberos authentication:

```
# kdadmin.local
# ktadd host/box2.mydomain.com
# ktadd nfs/box2.mydomain.com
# ktadd nfs/box1.mydomain.com
```

Finally, mount the share and perform a write test:

The screenshot shows two terminal windows. The left window on box1 has commands to mount the NFS share with Kerberos authentication and write to a file. The right window on box2 shows the file being created and its contents.

```
# mount -t nfs4 -o sec=krb5 box2:/nfs /mnt
# echo "Hello from Tecmint.com" > /mnt/greeting.txt
[root@box1 ~]# mount | grep krb5
box2:/nfs on /mnt type nfs4 (rw,relatime,vers=4.0,rsize=131072,wsize=131072,namlen
=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=krb5,clientaddr=192.168.0.18,lo
cal_lock=none,addr=192.168.0.20)
[root@box1 ~]# echo "Hello from Tecmint.com" > /mnt/greeting.txt
[root@box1 ~]#
```

```
* [root@box2 ~]# ls -l /nfs
total 4
-rw-r--r-- 1 nfsnobody nfs 23 Sep  5 21:18 greeting.txt
[root@box2 ~]#
```

http://www.tecmint.com

## Mount NFS Share

Let's now **unmount** the share, rename the **keytab** file in the client (to simulate it's not present) and try to mount the share again:

The screenshot shows a terminal session on box1 where the NFS share is unmounted, the keytab file is renamed, and then the share is remounted successfully. A note on the right side highlights the successful remount after keytab restoration.

```
# umount /mnt
# mv /etc/krb5.keytab /etc/krb5.keytab.orig
[root@box1 ~]# umount /mnt
[root@box1 ~]# mv /etc/krb5.keytab /etc/krb5.keytab.orig
[root@box1 ~]# mount -t nfs4 -o sec=krb5 box2:/nfs /mnt
mount.nfs4: access denied by server while mounting box2:/nfs
[root@box1 ~]# mv /etc/krb5.keytab.orig /etc/krb5.keytab
[root@box1 ~]# mount -t nfs4 -o sec=krb5 box2:/nfs /mnt
[root@box1 ~]# mount | grep krb5
box2:/nfs on /mnt type nfs4 (rw,relatime,vers=4.0,rsize=131072,wsize=131072,namlen
=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=krb5,clientaddr=192.168.0.18,lo
cal_lock=none,addr=192.168.0.20)
[root@box1 ~]#
```

Note how you can  
mount the share  
again after restoring  
the keytab file.

http://www.tecmint.com

## Mount Unmount Kerberos NFS Share

Now you can use the **NFS** share with **Kerberos-based** authentication.

## Summary

In this article we have explained how to set up **NFS** with **Kerberos** authentication. Since there is much more to the topic than we can cover in a single guide, feel free to check the online [Kerberos documentation](#) and since Kerberos is a bit tricky to say the least, don't hesitate to drop us a note using the form below if you run into any issue or need help with your testing or implementation.

## RHCE Series: Implementing HTTPS through TLS using Network Security Service (NSS) for Apache – Part 8

If you are a system administrator who is in charge of maintaining and securing a web server, you can't afford to not devote your very best efforts to ensure that data served by or going through your server is protected at all times.

# Red Hat Certified Engineer (RHCE)



 **Tecmint**.com  
Linux Howto's Guide

## Part 8: Implementing HTTPS through TLS using Network Security Service (NSS) for Apache in RHEL 7

RHCE Series: Implementing HTTPS through TLS using Network Security Service (NSS) for Apache – Part 8

In order to provide more secure communications between web clients and servers, the **HTTPS** protocol was born as a combination of **HTTP** and **SSL (Secure Sockets Layer)** or more recently, **TLS (Transport Layer Security)**.

Due to some serious security breaches, **SSL** has been deprecated in favor of the more robust **TLS**. For that reason, in this article we will explain how to secure connections between your web server and clients using **TLS**.

This tutorial assumes that you have already installed and configured your Apache web server. If not, please refer to following article in this site before proceeding further.

1. [Install LAMP \(Linux, MySQL/MariaDB, Apache and PHP\) on RHEL/CentOS 7](#)

### Installation of OpenSSL and Utilities

First off, make sure that **Apache** is running and that both **http** and **https** are allowed through the firewall:

```
# systemctl start http
# systemctl enable http
# firewall-cmd --permanent --add-service=http
# firewall-cmd --permanent --add-service=https
```

Then install the necessary packages:

```
# yum update && yum install openssl mod_nss crypto-utils
```

**Important:** Please note that you can replace **mod\_nss** with **mod\_ssl** in the command above if you want to use **OpenSSL** libraries instead of **NSS (Network Security Service)** to implement **TLS** (which one to use is left entirely up to you, but we will use NSS in this

article as it is more robust; for example, it supports recent cryptography standards such as **PKCS #11**).

Finally, uninstall **mod\_ssl** if you chose to use **mod\_nss**, or viceversa.

```
# yum remove mod_ssl
```

## Configuring NSS (Network Security Service)

After **mod\_nss** is installed, its default configuration file is created as **/etc/httpd/conf.d/nss.conf**. You should then make sure that all of the **Listen** and **VirtualHost** directives point to port **443** (default port for HTTPS):

**nss.conf – Configuration File**

```
Listen 443
VirtualHost _default_:443
```

Then restart **Apache** and check whether the **mod\_nss** module has been loaded:

```
# apachectl restart
# httpd -M | grep nss
[root@box1 ~]# httpd -M | grep nss
nss_module (shared)
[root@box1 ~]# █ http://www.tecmint.com
```

**Check Mod\_NSS Module Loaded in Apache**

Next, the following edits should be made in **/etc/httpd/conf.d/nss.conf** configuration file:

**1.** Indicate NSS database directory. You can use the default directory or create a new one. In this tutorial we will use the default:

```
NSSCertificateDatabase /etc/httpd/alias
```

**2.** Avoid manual passphrase entry on each system start by saving the password to the database directory in **/etc/httpd/nss-db-password.conf**:

```
NSSPassPhraseDialog file:/etc/httpd/nss-db-password.conf
```

Where **/etc/httpd/nss-db-password.conf** contains ONLY the following line and **mypassword** is the password that you will set later for the NSS database:

```
internal:mypassword
```

In addition, its permissions and ownership should be set to **0640** and **root:apache**, respectively:

```
# chmod 640 /etc/httpd/nss-db-password.conf
# chgrp apache /etc/httpd/nss-db-password.conf
```

**3.** Red Hat recommends disabling **SSL** and all versions of **TLS** previous to **TLSv1.0** due to the **POODLE SSLv3** vulnerability (more information [here](#)).

Make sure that every instance of the **NSSProtocol** directive reads as follows (you are likely to find only one if you are not hosting other virtual hosts):

```
NSSProtocol TLSv1.0,TLSv1.1
```

**4.** Apache will refuse to restart as this is a self-signed certificate and will not recognize the issuer as valid. For this reason, in this particular case you will have to add:

```
NSSEnforceValidCerts off
```

**5.** Though not strictly required, it is important to set a password for the NSS database:

```
# certutil -W -d /etc/httpd/alias
[root@box1 ~]# certutil -W -d /etc/httpd/alias
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password: Assign a password to the NSS
Re-enter password: certificate database
[root@box1 ~]# http://www.tecmint.com
```

Set Password for NSS Database

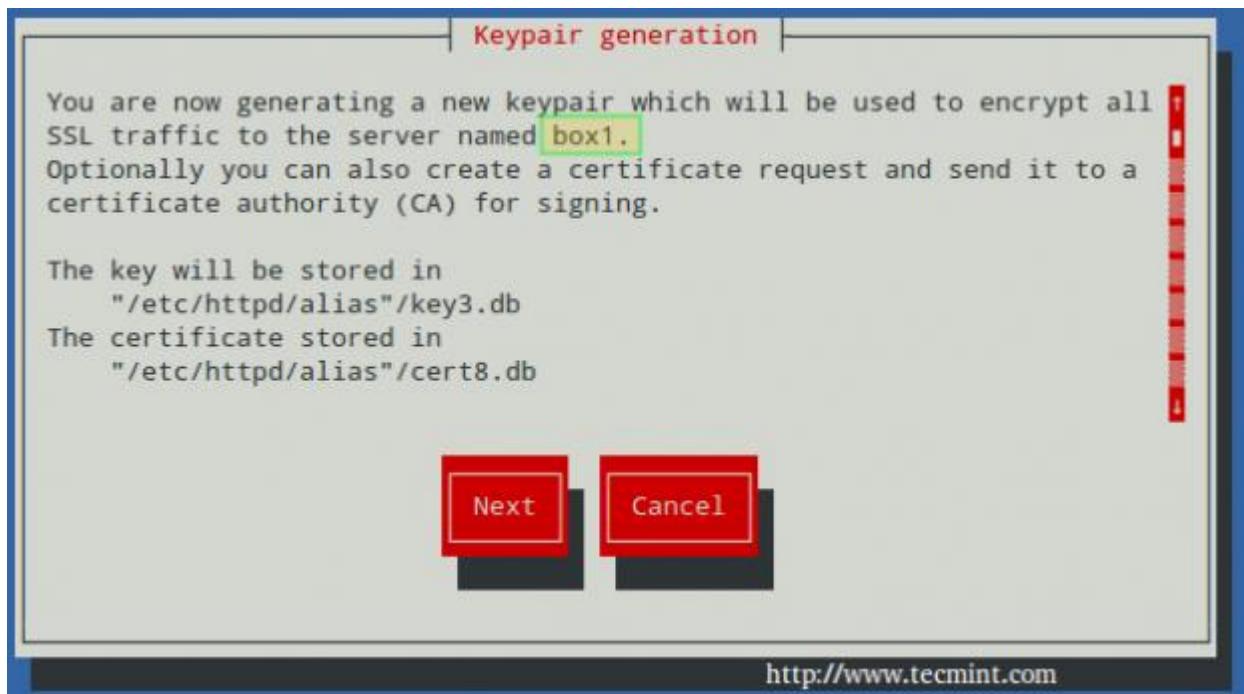
## Creating a Apache SSL Self-Signed Certificate

Next, we will create a self-signed certificate that will identify the server to our clients (please note that this method is not the best option for production environments; for such use you may want to consider buying a certificate verified by a 3rd trusted certificate authority, such as **DigiCert**).

To create a new NSS-compliant certificate for **box1** which will be valid for **365** days, we will use the **genkey** command. When this process completes:

```
# genkey --nss --days 365 box1
```

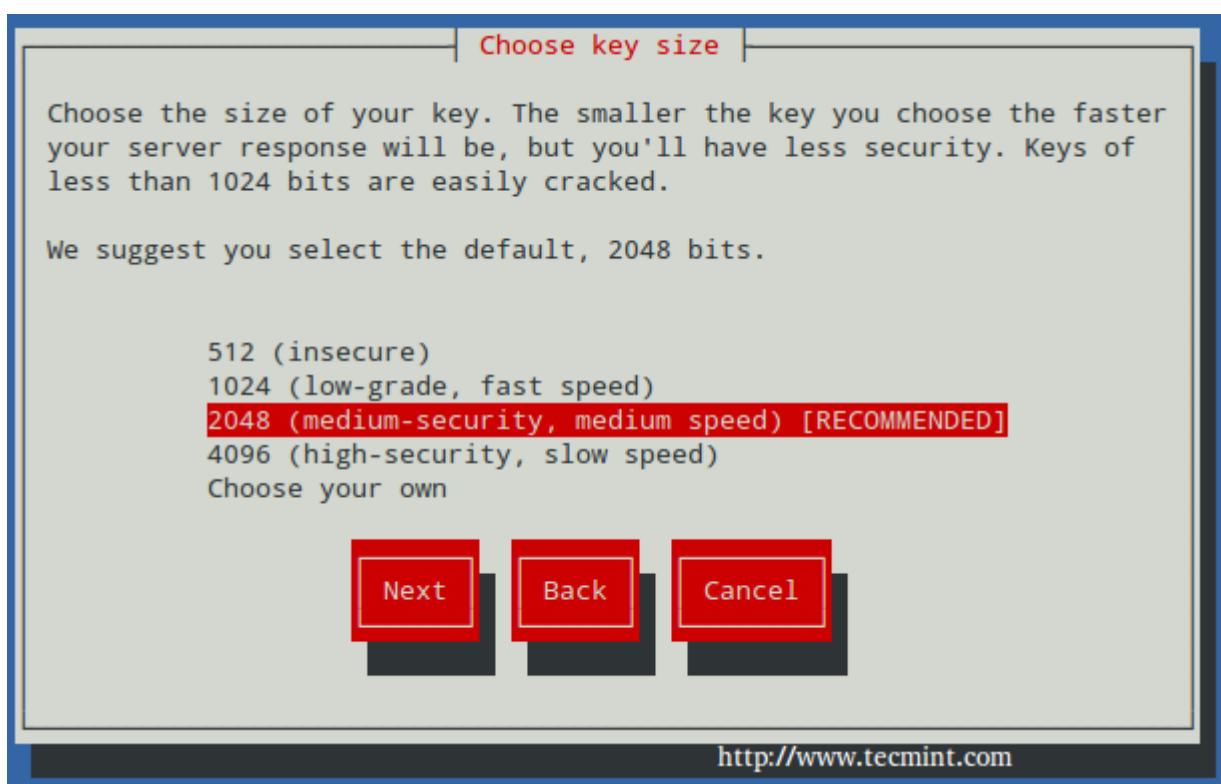
Choose **Next**:



<http://www.tecmint.com>

### Create Apache SSL Key

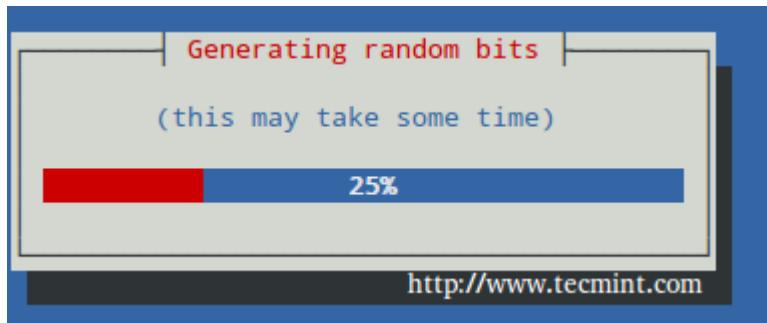
You can leave the default choice for the key size (**2048**), then choose **Next** again:



<http://www.tecmint.com>

### Select Apache SSL Key Size

Wait while the system generates random bits:



## Generating Random Key Bits

To speed up the process, you will be prompted to enter random text in your console, as shown in the following screencast. Please note how the progress bar stops when no input from the keyboard is received. Then, you will be asked to:

**1.** Whether to send the **Certificate Sign Request (CSR)** to a **Certificate Authority (CA)**: Choose **No**, as this is a self-signed certificate.

**2.** to enter the information for the certificate.

Finally, you will be prompted to enter the password to the NSS certificate that you set earlier:

```
# genkey --nss --days 365 box1
[root@box1 ~]# genkey --nss --days 365 box1
/usr/bin/certutil -S -n box1 -s "CN=Gabriel Canepa, O=Doing business with Tecmint, L=Villa Mercedes, ST=San Luis, C=AR"
tls/.rand.2965 -d /etc/httpd/alias -o /etc/pki/tls/certs/box1.crt
Enter Password or Pin for "NSS Certificate DB": Enter password here
Generating key. This may take a few moments... http://www.tecmint.com
```

## Apache NSS Certificate Password

At anytime, you can list the existing certificates with:

```
# certutil -L -d /etc/httpd/alias
[root@box1 ~]# certutil -L -d /etc/httpd/alias

Certificate Nickname                                     Trust Attributes
cacert                                                 SSL,S/MIME,JAR/XPI
Server-Cert                                            CT,C,C
alpha                                                 ,
box1                                                 ,P,
[root@box1 ~]# http://www.tecmint.com
```

## List Apache NSS Certificates

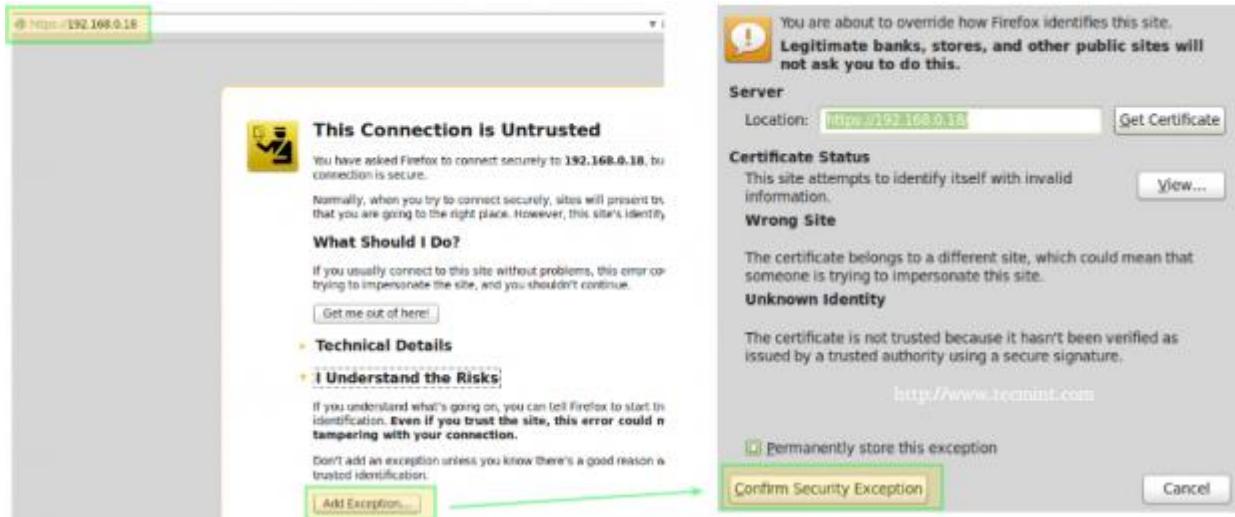
And delete them by name (only if strictly required, replacing box1 by your own certificate name) with:

```
# certutil -d /etc/httpd/alias -D -n "box1"
```

if you need to.c

## Testing Apache SSL HTTPS Connections

Finally, it's time to test the secure connection to our web server. When you point your browser to **https://<web server IP or hostname>**, you will get the well-known message “**This connection is untrusted**“:



### Check Apache SSL Connection

In the above situation, you can click on **Add Exception** and then **Confirm Security Exception** – but don't do it yet. Let's first examine the certificate to see if its details match the information that we entered earlier (as shown in the screencast).

To do so, click on **View... > Details** tab above and you should see this when you select Issuer from the list:

The screenshot shows a certificate viewer interface with the following sections:

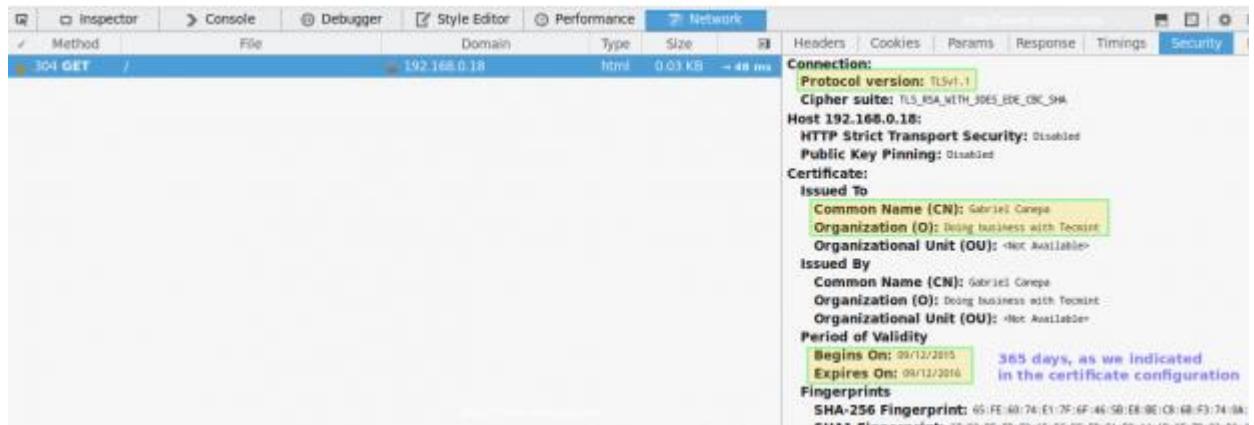
- Certificate Hierarchy:** Shows "Gabriel Canepa".
- Certificate Fields:**
  - Gabriel Canepa** (selected)
  - Certificate** (selected)
    - Version
    - Serial Number
    - Certificate Signature Algorithm** (highlighted in green)
    - Issuer** (highlighted in green)
  - Validity**
    - Not Before
    - Not After
  - Subject**
  - Subject Public Key Info**
- Field Value:**
  - CN = Gabriel Canepa
  - O = Doing business with Tecmint
  - L = Villa Mercedes
  - ST = San Luis
  - C = AR

The URL <http://www.tecmint.com> is shown at the bottom.

### Confirm Apache SSL Certificate Details

Now you can go ahead, confirm the exception (either for this time or permanently) and you will be taken to your web server's **DocumentRoot** directory via **https**, where you can inspect the connection details using your browser's builtin developer tools:

In **Firefox** you can launch it by right clicking on the screen, and choosing **Inspect Element** from the context menu, specifically through the **Network** tab:



### Inspect Apache HTTPS Connection

Please note that this is the same information as displayed before, which was entered during the certificate previously. There's also a way to test the connection using command line tools:

On the left (testing SSLv3):

```
# openssl s_client -connect localhost:443 -ssl3
```

On the right (testing TLS):

```
# openssl s_client -connect localhost:443 -tls1
[root@box1 ~]# openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
140299202217888:error:1408F108:SSL routines:SSL3_GET_RECORD:wrong version number:s
sl_pvt.c:339:
...
no peer certificate available
...
No client certificate CA names sent
...
SSL handshake has read 5 bytes and written 7 bytes
...
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
A connection that is attempted to be
secured using SSLv3 fails.
No cipher is negotiated, and even when
there's a handshake involved -as that way
client and server can decide that the version
number is wrong (bytes need to be sent)

SSL-Session:
Protocol : SSLv3
Cipher   : 0000
Session-ID:
Session-ID-ctx:
Master-Key:
Key-Agri : None
Krb5 Principal: None
PSK identity: None
PSK identity hint: None
Start Time: 1442224911
Timeout  : 7200 (sec)
Verify return code: 0 (ok)
[...]
[root@box1 ~]#
```

```
+ [root@box1 ~]# openssl s_client -connect localhost:443 -tls1
CONNECTED(00000003)
depth=0 C = AR, ST = San Luis, L = Villa Mercedes, O = Doing business with Tecmin
, CN = Gabriel Campana
verify error:num=18:self signed certificate
verify return:1
depth=0 C = AR, ST = San Luis, L = Villa Mercedes, O = Doing business with Tecmin
, CN = Gabriel Campana
verify return:1
Certificate Chain
0 s:/C=AR/ST=San Luis/L=Villa Mercedes/O=Doing business with Tecmin/CN=Gabriel
Campana
-----/i:/C=AR/ST=San Luis/L=Villa Mercedes/O=Doing business with Tecmin/CN=Gabriel
Campana
-----
Server certificate
-----BEGIN CERTIFICATE-----MIID6TCALmgAwIBAgIFAKPyhLQwDQYJKoZIhvchAQELBQAwDElMAkGAIUEBhMC
QVixETAPBgNVBAgTCPNbBBDkLzRcwQDVQHwE5WnwkxYSBNZAUjZWRIicxEK
MCIGATUEChMhR6SyhmcgYrVzam51c3Mg0210vCBzWntw50MrIcwFYQW00Dew5H
YWJyowNIEhNmWVTAwFe0xNTASWT1yNx3MzhaPw0xNjA5MTTyMzA3MzhaHgx
CaAIBgNVBYTAKFSMRgEoVwDVQGIwhTYwkgTHVpczEXMDUGA1UEB0wD0M1sbGEg
TWVyZVkZXNnJDAiBgNVBAsTQRwvWnIGJic2luZWRuIhdpdGggGVjbwLiidEX
MBUGA1UEAx0B0R2fLcm1lbCD0W51cGtaggELIM0GCSqS1b3DQEBAQUAA4IBBwAw
ggEKAoIBAQODVjIP1ak0BgjhAUUC2pA01vQdSY5wOHHhLZu3fCPM6/2s2kje
511Ms3X3b3uKh3k0wBrhF2zA182870N28EBvfrw9FozqbhvnIn70kafp2vzu
awvvpdpd0xDr2c18mm0D3hkrH0TsoevPNAr5/29nsE91RDFgvbAR/SHLC1X
F7b0cq-eqMrlaLFqBXIRAF7nsVcCtp18tAVg6L169w5mNzE5bCPvYBjJSz
r+04d50u+0qT4Q76qTgF07cik6N, L7T002xDGh77vqBLysje94L0CeJhTz4MT
r+7V1Mwub.../i:-----
```

On the other hand, a connection using TLS succeeds and as result, the certificate details are displayed. Since this output is truncated, you may want to refer to your own experiment for more details (not shown in this screenshot).

<http://www.tecmint.com>

## Testing Apache SSL and TLS Connections

Refer to the screenshot above for more details.

## Summary

As I'm sure you already know, the presence of **HTTPS** inspires trust in visitors who may have to enter personal information in your site (from **user names** and **passwords** all the way to financial / bank account information).

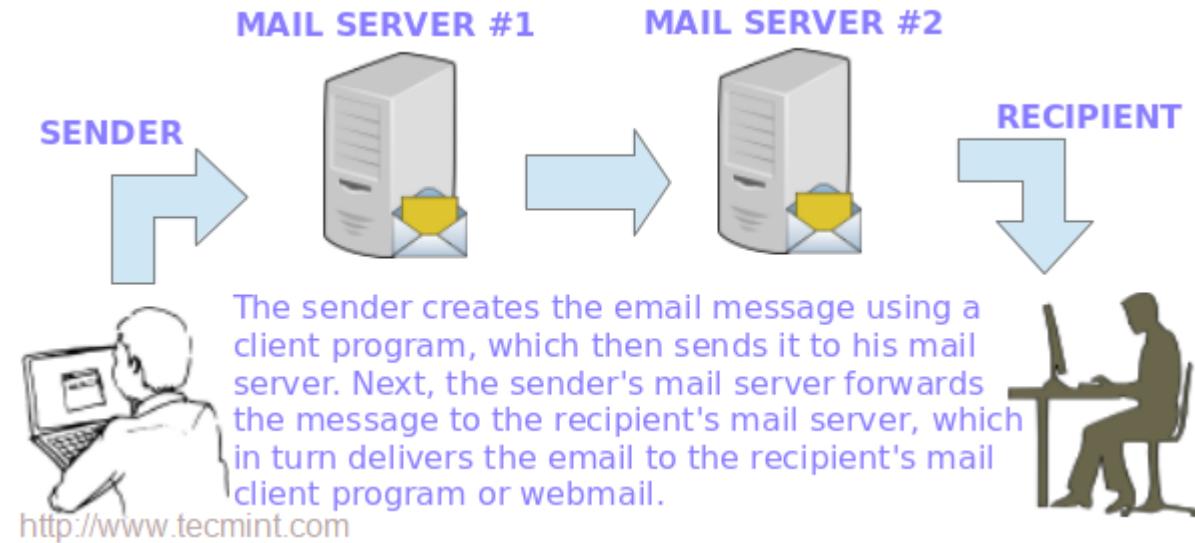
In that case, you will want to get a certificate signed by a trusted **Certificate Authority** as we explained earlier (the steps to set it up are identical with the exception that you will need to send the **CSR** to a **CA**, and you will get the signed certificate back); otherwise, a self-signed certificate as the one used in this tutorial will do.

For more details on the use of NSS, please refer to the online help about [mod-nss](#). And don't hesitate to let us know if you have any questions or comments.

## How to Setup Postfix Mail Server (SMTP) using null-client Configuration – Part 9

Regardless of the many online communication methods that are available today, email remains a practical way to deliver messages from one end of the world to another, or to a person sitting in the office next to ours.

The following image illustrates the process of email transport starting with the sender until the message reaches the recipient's inbox:



## How Mail Setup Works

To make this possible, several things happen behind the scenes. In order for an email message to be delivered from a client application (such as [Thunderbird](#), Outlook, or webmail services such as [Gmail](#) or [Yahoo! Mail](#)) to a mail server, and from there to the destination server and finally to its intended recipient, a **SMTP (Simple Mail Transfer Protocol)** service must be in place in each server.

That is the reason why in this article we will explain how to set up a **SMTP** server in **RHEL 7** where emails sent by local users (even to other local users) are forwarded to a central mail server for easier access.

In the exam's requirements this is called a **null-client** setup.

Our test environment will consist of an originating mail server and a central mail server or relayhost.

```
Original Mail Server: (hostname: box1.mydomain.com / IP: 192.168.0.18)
Central Mail Server: (hostname: mail.mydomain.com / IP: 192.168.0.20)
```

For name resolution we will use the well-known **/etc/hosts** file on both boxes:

192.168.0.18	box1.mydomain.com	box1
192.168.0.20	mail.mydomain.com	mail

## Installing Postfix and Firewall / SELinux Considerations

To begin, we will need to (in both servers):

### 1. Install Postfix:

```
# yum update && yum install postfix
```

**2.** Start the service and enable it to run on future reboots:

```
# systemctl start postfix  
# systemctl enable postfix
```

**3.** Allow mail traffic through the firewall:

```
# firewall-cmd --permanent --add-service=smtp  
# firewall-cmd --add-service=smtp  
[root@box1 ~]# firewall-cmd --list-services  
dhcpv6-client ftp http https samba ssh  
[root@box1 ~]# firewall-cmd --permanent --add-service=smtp  
success  
[root@box1 ~]# firewall-cmd --add-service=smtp  
success  
[root@box1 ~]# firewall-cmd --list-services  
dhcpv6-client ftp http https samba smtp ssh  
[root@box1 ~]#  
http://www.tecmint.com
```

Open Mail Server SMTP Port in Firewall

**4.** Configure Postfix on **box1.mydomain.com**.

Postfix's main configuration file is located in **/etc/postfix/main.cf**. This file itself is a great documentation source as the included comments explain the purpose of the program's settings.

For brevity, let's display only the lines that need to be edited (yes, you need to leave **mydestination** blank in the originating server; otherwise the emails will be stored locally as opposed to in a central mail server which is what we actually want):

Configure Postfix on **box1.mydomain.com**

```
myhostname = box1.mydomain.com  
mydomain = mydomain.com  
myorigin = $mydomain  
inet_interfaces = loopback-only  
mydestination =  
relayhost = 192.168.0.20
```

**5.** Configure Postfix on **mail.mydomain.com**.

Configure Postfix on **mail.mydomain.com**

```
myhostname = mail.mydomain.com  
mydomain = mydomain.com  
myorigin = $mydomain  
inet_interfaces = all  
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain  
mynetworks = 192.168.0.0/24, 127.0.0.0/8
```

And set the related **SELinux** boolean to true permanently if not already done:

```
# setsebool -P allow postfix local write mail spool on
[root@mail ~]# getsebool -a | grep postfix
postfix_local_write_mail_spool --> off
[root@mail ~]# setsebool -P allow_postfix_local_write_mail_spool on
[root@mail ~]# getsebool -a | grep postfix
postfix_local_write_mail_spool --> on
http://www.tecmint.com
```

## Set Postfix SELinux Permission

The above **SELinux** boolean will allow Postfix to write to the mail spool in the central server.

**5.** Restart the service on both servers for the changes to take effect:

```
# systemctl restart postfix
```

If **Postfix** does not start correctly, you can use following commands to troubleshoot.

```
# systemctl -l status postfix
# journalctl -xn
# postconf -n
```

## Testing the Postfix Mail Servers

To test the mail servers, you can use any **Mail User Agent** (most commonly known as **MUA** for short) such as [mail or mutt](#).

Since **mutt** is a personal favorite, I will use it in **box1** to send an email to user **tecmint** using an existing file (**mailbody.txt**) as message body:

```
# mutt -s "Part 9-RHCE series" tecmint@mydomain.com < mailbody.txt
[root@box1 ~]# cat mailbody.txt
My name is Gabriel Canepe and I'm a technical writer with Tecmint.com.
By the way, this is Part 9 of the RHCE series. Hope you're enjoying this guide!
[root@box1 ~]# mutt -s "Part 9-RHCE series" tecmint@mydomain.com < mailbody.txt
http://www.tecmint.com
```

## Test Postfix Mail Server

Now go to the central mail server (**mail.mydomain.com**), log on as user **tecmint**, and check whether the email was received:

```
# su - tecmint
# mail
```

```
[root@mail ~]# su - tecmint
Last login: Sat Sep 19 21:54:20 ART 2015 on pts/0
[tecmint@mail ~]$ mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/tecmint": 1 message 1 new
>N 1 root                               Sat Sep 19 21:53 23/923  "Part 9-RHCE series"
& 1
Message 1:
From root@mydomain.com Sat Sep 19 21:53:53 2015
Return-Path: <root@mydomain.com>
X-Original-To: tecmint@mydomain.com
Delivered-To: tecmint@mydomain.com
Date: Sat, 19 Sep 2015 21:53:52 -0300
From: root <root@>
To: tecmint@mydomain.com
Subject: Part 9-RHCE series
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
User-Agent: Mutt/1.5.21 (2010-09-15)
Status: R

My name is Gabriel Cánepa and I'm a technical writer with Tecmint.com.
By the way, this is Part 9 of the RHCE series. Hope you're enjoying this guide!
http://www.tecmint.com
```

## Check Postfix Mail Server Delivery

If the email was not received, check root's mail spool for a warning or error notification. You may also want to make sure that the **SMTP** service is running on both servers and that port **25** is open in the central mail server using [nmap command](#):

```
# nmap -PN 192.168.0.20
gacanepa@Mint13 ~ $ nmap -PN 192.168.0.20

Starting Nmap 5.21 ( http://nmap.org ) at 2015-09-19 21:25 ART
Nmap scan report for 192.168.0.20
Host is up (0.88s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
88/tcp    open  kerberos-sec
2049/tcp  open  nfs

Nmap done: 1 IP address (1 host up) scanned in 50.00 seconds
http://www.tecmint.com
```

## Troubleshoot Postfix Mail Server

### Summary

Setting up a **mail server** and a **relay host** as shown in this article is an essential skill that every system administrator must have, and represents the foundation to understand and install a more complex scenario such as a mail server hosting a live domain for several (even hundreds or thousands) of email accounts.

(Please note that this kind of setup requires a DNS server, which is out of the scope of this guide), but you can use following article to setup **DNS Server**:

1. [Setup Cache only DNS Server in CentOS/RHEL 07](#)

Finally, I highly recommend you become familiar with Postfix's configuration file (**main.cf**) and the program's man page. If in doubt, don't hesitate to drop us a line using the form below or using our forum, **Linuxsay.com**, where you will get almost immediate help from Linux experts from all around the world.

## Install and Configure Caching-Only DNS Server in RHEL/CentOS 7 – Part 10

DNS servers comes in several types such as master, slave, forwarding and cache, to name a few examples, with cache-only DNS being the one that is easier to setup. Since DNS uses the UDP protocol, it improves the query time because it does not require an acknowledgement.



The banner features a dark red background. At the top, the text "Red Hat Certified Engineer (RHCE)" is displayed in large, white, sans-serif font. Below this, there is a logo for Red Hat, featuring a stylized red hat icon and the word "redhat." to its right. To the right of the Red Hat logo is the Tecmint logo, which includes the text "Tecmint" with a blue square icon, and "Linux Howto's Guide" underneath. At the bottom of the banner, the text "Part 11: Setup Caching-Only DNS Server" is displayed in a large, white, sans-serif font.

### RHCE Series: Setup Cache-Only DNS in RHEL and CentOS 7 – Part 11

The cache-only DNS server is also known as resolver, which will query DNS records and fetch all the DNS details from other servers, and keep each query request in its cache for later use so that when we perform the same request in the future, it will serve from its cache, thus reducing the response time even more.

If you're looking to setup DNS Caching-Only Server in **CentOS/RHEL 6**, follow this guide here:

## [Setting Up Caching-Only DNS Name Server in CentOS/RHEL 6](#)

### My Testing Environment

DNS server	:	dns.tecmintlocal.com (Red Hat Enterprise Linux 7.1)
Server IP Address	:	192.168.0.18
Client	:	node1.tecmintlocal.com (CentOS 7.1)
Client IP Address	:	192.168.0.29

### Step 1: Installing Cache-Only DNS Server in RHEL/CentOS 7

**1.** The Cache-Only DNS server, can be installed via the **bind** package. If you don't remember the package name, you can do a quick search for the package name using the command below.

```
# yum search bind
[root@dns ~]# yum search bind
Loaded plugins: product-id, subscription-manager
=====
PackageKit-device-rebind.x86_64 : Device rebind functionality for PackageKit
bind.x86_64 : The Berkeley Internet Name Domain (BIND) DNS (Domain Name System) server
bind-chroot.x86_64 : A chroot runtime environment for the ISC BIND DNS server, named(8)
bind_dyndb_ldap.x86_64 : LDAP back-end plugin for RTND
```

Search DNS Bind Package

**2.** In the above result, you will see several packages. From those, we need to choose and install only **bind** and **bind-utils** packages using following yum command.

```
# yum install bind bind-utils -y
[root@dns ~]# yum install bind bind-utils -y
Loaded plugins: product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
  > Package bind.x86_64 33:9.9.4-29.el7_2.1 will be installed
```

Install DNS Bind in RHEL/CentOS 7

### Step 2: Configure Cache-Only DNS in RHEL/CentOS 7

**3.** Once DNS packages are installed we can go ahead and configure DNS. Open and edit `/etc/named.conf` using your preferred text editor. Make the changes suggested below (or you can use your settings as per your requirements).

```
listen-on port 53 { 127.0.0.1; any; };
allow-query      { localhost; any; };
allow-query-cache { localhost; any; };
```

```

// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named
// server as a caching only nameserver (as a localhost DNS resolver
//
// See /usr/share/doc/bind*/sample/ for example named configuration
//


options {
    listen-on port 53 { 127.0.0.1; any; };
    listen-on-v6 port 53 { ::1; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { localhost; any; };
    allow-query-cache { localhost; any; };
/*
 - If you are building an AUTHORITATIVE DNS server, do NOT
 - If you are building a RECURSIVE (caching) DNS server, yo
 recursion.

```

## Configure Cache-Only DNS in CentOS and RHEL 7

These directives instruct the DNS server to listen on **UDP** port **53**, and to allow queries and caches responses from **localhost** and any other machine that reaches the server.

**4.** It is important to note that the ownership of this file must be set to `root:named` and also if **SELinux** is enabled, after editing the configuration file we need to make sure that its context is set to `named_conf_t` as shown in Fig. 4 (same thing for the auxiliary file `/etc/named.rfc1912.zones`):

```
# ls -lZ /etc/named.conf
# ls -lZ /etc/named.rfc1912.zones
```

Otherwise, configure the SELinux context before proceeding:

```
# semanage fcontext -a -t named_conf_t /etc/named.conf
# semanage fcontext -a -t named_conf_t /etc/named.rfc1912.zones
```

**5.** Additionally, we need to test the DNS configuration now for some syntax error before starting the bind service:

```
# named-checkconf /etc/named.conf
```

**6.** After the syntax verification results seems perfect, restart the **named** service to take new changes into effect and also make the service to auto start across system boots, and then check its status:

```
# systemctl restart named
# systemctl enable named
# systemctl status named
```

```
[root@dns ~]# ls -lZ /etc/named.conf
-rw-r-----. root named system_u:object_r:named_conf_t:s0 /etc/named.conf
[root@dns ~]# ls -lZ /etc/named.rfc1912.zones
-rw-r-----. root named system_u:object_r:named_conf_t:s0 /etc/named.rfc1912.zones
[root@dns ~]# named-checkconf /etc/named.conf
[root@dns ~]# systemctl restart named
[root@dns ~]# systemctl enable named
ln -s '/usr/lib/systemd/system/named.service' '/etc/systemd/system/multi-user.target.wants/
[root@dns ~]# systemctl status named
named.service - Berkeley Internet Name Domain (DNS)
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled)
   Active: active (running) since Sat 2016-01-02 19:44:07 ART; 37s ago
     Main PID: 14550 (named)
        CGroup: /system.slice/named.service
                  └─14550 /usr/sbin/named -u named
```

## Configure and Start DNS Named Service

7. Next, open the port 53 on the firewall.

```
# firewall-cmd --add-port=53/udp
# firewall-cmd --add-port=53/udp --permanent
[root@dns ~]# firewall-cmd --add-port=53/udp
success
[root@dns ~]# firewall-cmd --add-port=53/udp --permanent
success
```

## Open DNS Port 53 on Firewall

### Step 3: Chroot Cache-Only DNS Server in RHEL and CentOS 7

8. If you wish to deploy the Cache-only DNS server within **chroot** environment, you need to have the package **chroot** installed on the system and no further configuration is needed as it by default hard-link to chroot.

```
# yum install bind-chroot -y
```

Once **chroot** package has been installed, you can restart named to take the new changes into effect:

```
# systemctl restart named
```

9. Next, create a symbolic link (also named `/etc/named.conf`) inside `/var/named/chroot/etc/`:

```
# ln -s /etc/named.conf /var/named/chroot/etc/named.conf
```

### Step 4: Configure DNS on Client Machine

10. Add the DNS Cache servers **IP 192.168.0.18** as resolver to the client machine. Edit `/etc/sysconfig/network-scripts/ifcfg-enp0s3` as shown in the following figure:

```
DNS=192.168.0.18
HWADDR=08:00:27:CB:39:E5
TYPE=Ethernet
BOOTPROTO=static
IPADDR=192.168.0.29
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
NM_CONTROLLED=no
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=enp0s3
UUID=8483ad3e-de0a-4cca-b8fc-d2cf9a1e169f
ONBOOT=yes
DNS=192.168.0.18
```

Configure DNS on Client Machine

And **/etc/resolv.conf** as follows:

```
nameserver 192.168.0.18
```

**11.** Finally it's time to check our cache server. To do this, you can use [dig utility](#) or [nslookup command](#).

Choose any website and query it twice (we will use **facebook.com** as an example). Note that with dig the second time the query is completed much faster because it is being served from the cache.

```
# dig facebook.com
```

```
[root@node1 ~]# dig facebook.com
; <>> DiG 9.9.4-RedHat-9.9.4-18.el7_1.5 <>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46019
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITI
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;facebook.com.           IN      A
;; ANSWER SECTION:
facebook.com.        300     IN      A      66.220.158.6
;; AUTHORITY SECTION:
facebook.com.       172799   IN      NS      b.ns.facebook.com
facebook.com.       172799   IN      NS      a.ns.facebook.com
;; ADDITIONAL SECTION:
a.ns.facebook.com.  172799   IN      A       69.171.239.1
a.ns.facebook.com.  172799   IN      AAAA    2a03:2880:ffff::1
b.ns.facebook.com.  172799   IN      A       69.171.255.1
b.ns.facebook.com.  172799   IN      AAAA    2a03:2880:ffff::1
;; Query time: 528 msec
;; SERVER: 192.168.0.18#53(192.168.0.18) ←
;; WHEN: Sat Jan 02 18:14:10 EST 2016
;; MSG SIZE rcvd: 180
```

**1st time**

```
[root@node1 ~]# dig facebook.com
; <>> DiG 9.9.4-RedHat-9.9.4-18.el7_1.5 <>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46019
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITI
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;facebook.com.           IN      A
;; ANSWER SECTION:
facebook.com.        294     IN      A      66.220.158.6
;; AUTHORITY SECTION:
facebook.com.       172793   IN      NS      b.ns.facebook.com
facebook.com.       172793   IN      NS      a.ns.facebook.com
;; ADDITIONAL SECTION:
a.ns.facebook.com.  172793   IN      A       69.171.239.1
a.ns.facebook.com.  172793   IN      AAAA    2a03:2880:ffff::1
b.ns.facebook.com.  172793   IN      A       69.171.255.1
b.ns.facebook.com.  172793   IN      AAAA    2a03:2880:ffff::1
;; Query time: 9 msec
;; SERVER: 192.168.0.18#53(192.168.0.18)
;; WHEN: Sat Jan 02 18:14:16 EST 2016
;; MSG SIZE rcvd: 180
```

**2nd**

## Check Cache only DNS Queries

You can also use **nslookup** to verify that the DNS server is working as expected.

```
# nslookup facebook.com
[root@node1 ~]# nslookup facebook.com
Server:      192.168.0.18
Address:     192.168.0.18#53

Non-authoritative answer:
Name:   facebook.com
Address: 66.220.158.68

[root@node1 ~]#
```

## Checking DNS Query with nslookup

### Summary

In this article we have explained how to set up a DNS Cache-only server in **Red Hat Enterprise Linux 7** and **CentOS 7**, and tested it in a client machine. Feel free to let us know if you have any questions or suggestions using the form below.

## How to Setup and Configure Network Bonding or Teaming in RHEL/CentOS 7 – Part 11

When a system administrator wants to increase the bandwidth available and provide redundancy and load balancing for data transfers, a kernel feature known as network **bonding** allows to get the job done in a cost-effective way.

Read more about how to increase or bandwidth throttling in Linux

In simple words, bonding means aggregating two or more physical network interfaces (called slaves) into a single, logical one (called master). If a specific NIC (Network Interface Card) experiences a problem, communications are not affected significantly as long as the other(s) remain active.

Read more about network bonding in Linux systems here:

1. [Network Teaming or NiC Bondin in RHEL/CentOS 6/5](#)
2. [Network NIC Bonding or Teaming on Debian based Systems](#)

### Enabling and Configuring Network Bonding or Teaming

By default, the bonding kernel module is not enabled. Thus, we will need to load it and ensure it is persistent across boots. When used with the `--first-time` option, **modprobe** will alert us if loading the module fails:

```
# modprobe --first-time bonding
```

The above command will load the bonding module for the current session. In order to ensure persistency, create a `.conf` file inside `/etc/modules-load.d` with a descriptive name, such as `/etc/modules-load.d/bonding.conf`:

```
# echo "# Load the bonding kernel module at boot" > /etc/modules-
load.d/bonding.conf
# echo "bonding" >> /etc/modules-load.d/bonding.conf
```

Now reboot your server and once it restarts, make sure the bonding module is loaded automatically, as seen in **Fig. 1**:

```
[root@server ~]# lsmod | grep bonding
bonding               136630  0
[root@server ~]#
```

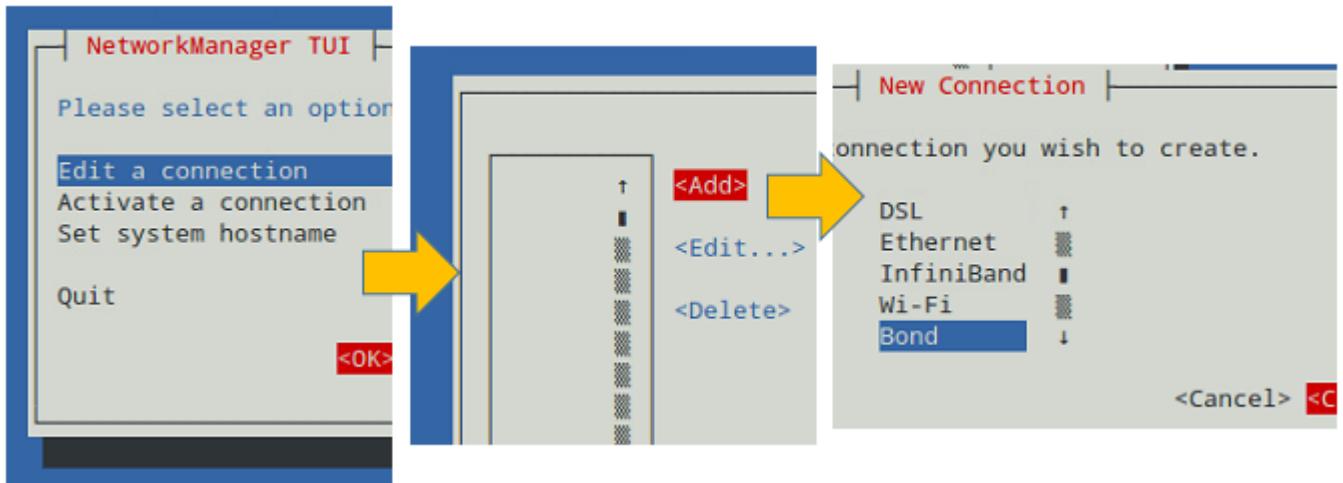
Check Network Bonding Module Loaded in Kernel

In this article we will use 3 interfaces (`enp0s3`, `enp0s8`, and `enp0s9`) to create a bond, named conveniently `bond0`.

To create `bond0`, we can either use **nmtui**, the text interface for controlling **NetworkManager**. When invoked without arguments from the command line, **nmtui** brings

up a text interface that allows you to edit an existing connection, activate a connection, or set the system hostname.

Choose **Edit connection** → **Add** → **Bond** as illustrated in Fig. 2:

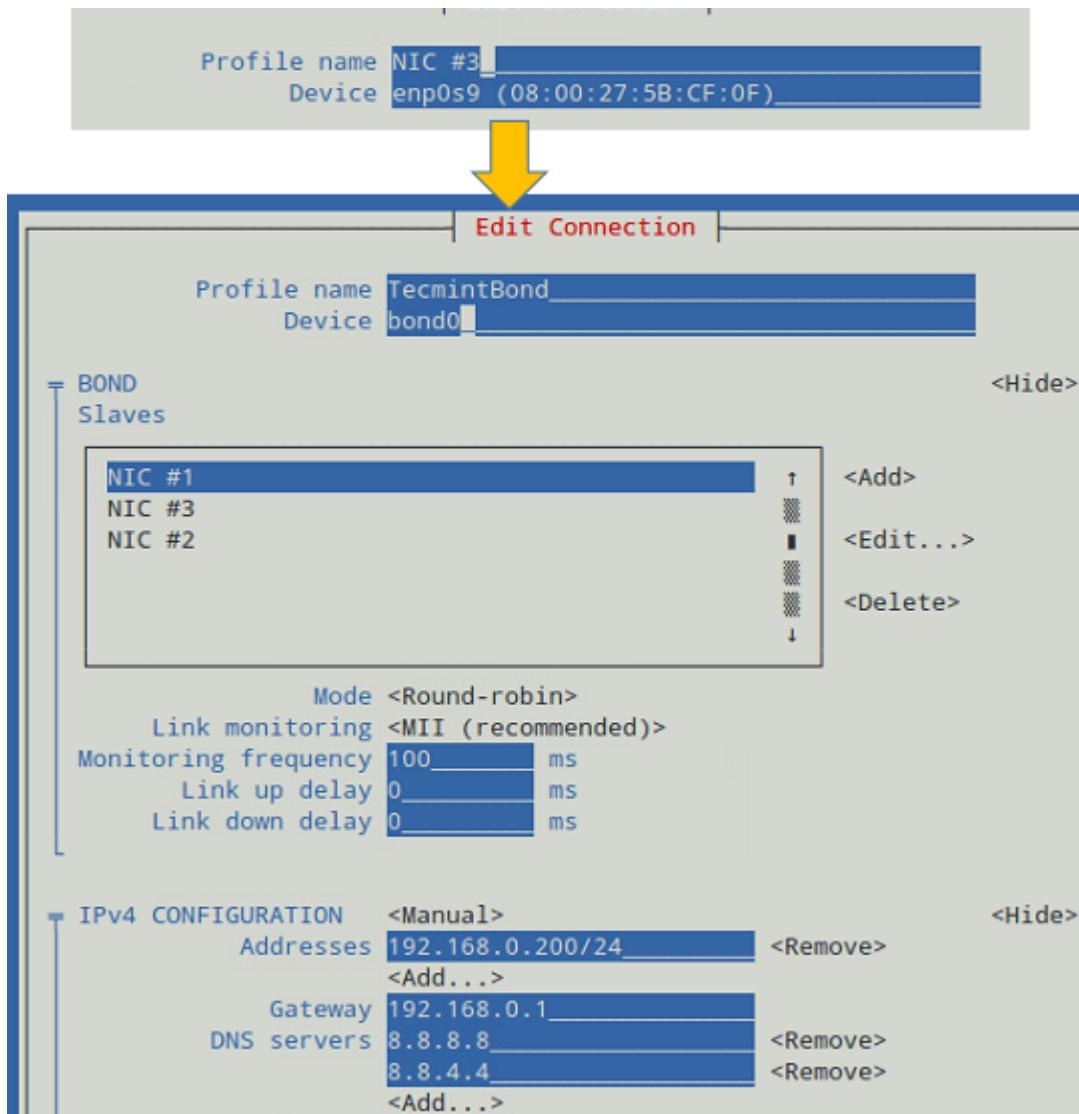


### Create Network Bonding Channel

In the **Edit Connection** screen, add the slave interfaces (`enp0s3`, `enp0s8`, and `enp0s9` in our case) and give them a descriptive (Profile) name (for example, `NIC #1`, `NIC #2`, and `NIC #3`, respectively).

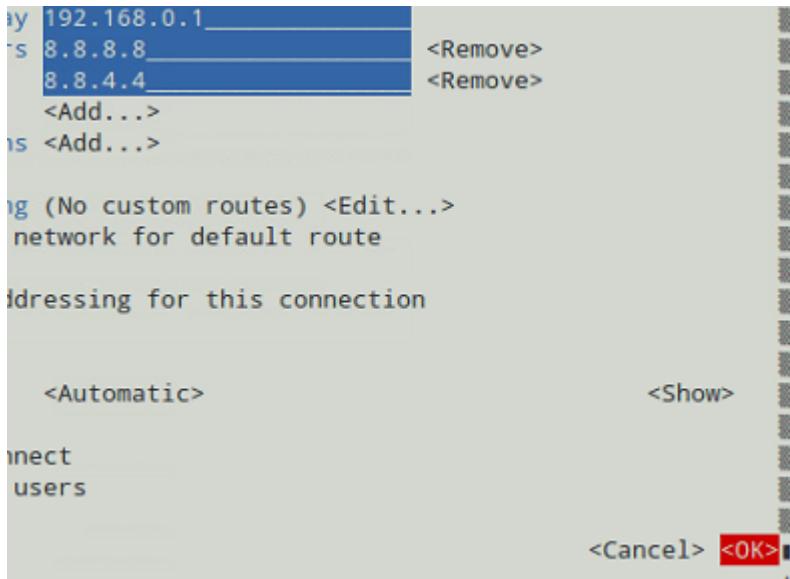
In addition, you will need to set a name and device for the bond (`TecmintBond` and `bond0` in Fig. 3, respectively) and an IP address for `bond0`, enter a gateway address, and the IPs of DNS servers.

**Note** that you do not need to enter the MAC address of each interface since **nmtui** will do that for you. You can leave all other settings as default. See Fig. 3 for more details.



## Network Bonding Teaming Configuration

When you're done, go to the bottom of the screen and choose **OK** (see Fig. 4):



### Configuration of bond0

And you're done. Now you can exit the text interface and return to the command line, where you will enable the newly created interface using [ip command](#):

```
# ip link set dev bond0 up
```

After that, you can see that `bond0` is UP and is assigned **192.168.0.200**, as seen in Fig. 5:

```
# ip addr show bond0
[root@server ~]# ip addr show bond0
5: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 08:00:27:5b:cf:0f brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.200/24 brd 192.168.0.255 scope global bond0
            valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe5b:cf0f/64 scope link tentative dadfailed
            valid_lft forever preferred_lft forever
[root@server ~]#
```

### Check Network Bond Interface Status

## Testing Network Bonding or Teaming in Linux

To verify that `bond0` actually works, you can either ping its IP address from another machine, or what's even better, watch the kernel interface table in real time (well, the refresh time in seconds is given by the `-n` option) to see how network traffic is distributed between the three network interfaces, as shown in Fig. 6.

The `-d` option is used to highlight changes when they occur:

```
# watch -d -n1 netstat -i
```

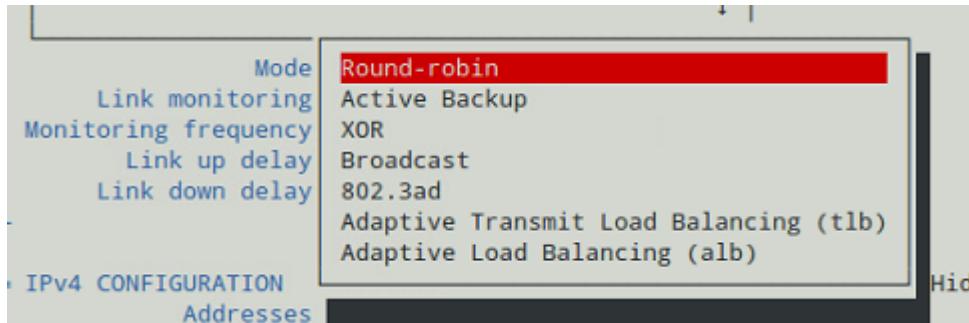
Every 1.0s: netstat -i											
Kernel Interface table											
Iface	MTU	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg	
bond0	1500	2344	0	0	0	889	0	0	0	BMmRU	
enp0s3	1500	804	0	0	0	295	0	0	0	BMsRU	
enp0s8	1500	802	0	0	0	297	0	0	0	BMsRU	
enp0s9	1500	1790	0	0	0	297	0	0	0	BMsRU	
lo	65536	73	0	0	0	73	0	0	0	LRU	

## Check Kernel Interface Table

It is important to note that there are several bonding modes, each with its distinguishing characteristics. They are documented in section 4.5 of the [Red Hat Enterprise Linux 7 Network Administration](#) guide. Depending on your needs, you will choose one or the other.

In our current setup, we chose the **Round-robin** mode (see Fig. 3), which ensures packets are transmitted beginning with the first slave in sequential order, ending with the last slave, and starting with the first again.

The Round-robin alternative is also called `mode 0`, and provides load balancing and fault tolerance. To change the bonding mode, you can use **nmtui** as explained before (see also Fig. 7):



## Changing Bonding Mode Using nmtui

If we change it to **Active Backup**, we will be prompted to choose a slave that will be the only one active interface at a given time. If such card fails, one of the remaining slaves will take its place and becomes active.

Let's choose `enp0s3` to be the primary slave, bring `bond0` down and up again, restart the network, and display the kernel interface table (see Fig. 8).

Note how data transfers (**TX-OK** and **RX-OK**) are now being made over `enp0s3` only:

```
# ip link set dev bond0 down
# ip link set dev bond0 up
# systemctl restart network
```

Mode <Active Backup>  
 Primary enp0s3  
 Link monitoring <MIPI (recommended)>  
 Monitoring frequency 100 ms  
 Link up delay 0 ms  
 Link down delay 0 ms

```
Every 1.0s: netstat -i

Kernel Interface table
Iface      MTU     RX-OK  RX-ERR  RX-DRP  RX-OVR    TX-OK  TX-ERR  TX-DRP  TX-OVR Flg
bond0      1500    5605    0       95 0     2439    0       0       0       BMmRU
enp0s3     1500    1497    0       0 0     898     0       0       0       BMsRU
enp0s8     1500    1355    0       48 0     774     0       0       0       BMsRU
enp0s9     1500    3984    0       47 0     767     0       0       0       BMsRU
lo         65536   265     0       0 0     265     0       0       0       LRU
```

### Bond Acting in Active Backup Mode

Alternatively, you can view the bond as the kernel sees it (see Fig. 9):

```
# cat /proc/net/bonding/bond0
```

```
[root@server ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: enp0s3 (primary_reselect always)
Currently Active Slave: enp0s3
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: enp0s3
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:4e:59:37
Slave queue ID: 0

Slave Interface: enp0s8
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:17:b6:c1
Slave queue ID: 0

Slave Interface: enp0s9
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:5b:cf:0f
Slave queue ID: 0
[root@server ~]#
```

Check Network Bond as Kernel

## Summary

In this chapter we have discussed how to set up and configure bonding in **Red Hat Enterprise Linux 7** (also works on **CentOS 7** and **Fedora 22+**) in order to increase bandwidth along with load balancing and redundancy for data transfers.

As you take the time to explore other bonding modes, you will come to master the concepts and practice related with this topic of the certification.

If you have questions about this article, or suggestions to share with the rest of the community, feel free to let us know using the comment form below.

## Create Centralized Secure Storage using iSCSI Target / Initiator on RHEL/CentOS 7 – Part 12

iSCSI is a block level Protocol for managing storage devices over TCP/IP Networks, specially over long distances. iSCSI target is a remote hard disk presented from an remote iSCSI server (or) target. On the other hand, the iSCSI client is called the **Initiator**, and will access the storage that is shared in the **Target** machine.

The following machines have been used in this article:

### Server (Target):

```
Operating System - Red Hat Enterprise Linux 7  
iSCSI Target IP - 192.168.0.29  
Ports Used : TCP 860, 3260
```

### Client (Initiator):

```
Operating System - Red Hat Enterprise Linux 7  
iSCSI Target IP - 192.168.0.30  
Ports Used : TCP 3260
```

## Step 1: Installing Packages on iSCSI Target

To install the packages needed for the target (we will deal with the client later), do:

```
# yum install targetcli -y
```

When the installation completes, we will start and enable the service as follows:

```
# systemctl start target  
# systemctl enable target
```

Finally, we need to allow the service in firewalld:

```
# firewall-cmd --add-service=iscsi-target  
# firewall-cmd --add-service=iscsi-target --permanent
```

And last but not least, we must not forget to allow the iSCSI target discovery:

```
# firewall-cmd --add-port=860/tcp  
# firewall-cmd --add-port=860/tcp --permanent  
# firewall-cmd --reload
```

## Step 2: Defining LUNs in Target Server

Before proceeding to defining LUNs in the **Target**, we need to create two logical volumes as explained in Part 6 of **RHCSA** series (“[Configuring system storage](#)”).

This time we will name them `vol_projects` and `vol_backups` and place them inside a volume group called `vg00`, as shown in Fig. 1. Feel free to choose the space allocated to each LV:

```
[root@server ~]# lvdisplay vg00/vol_backups vg00/vol_projects
--- Logical volume ---
LV Path          /dev/vg00/vol_projects
LV Name          vol_projects
VG Name          vg00
LV UUID          03os1b-encP-XVDX-wAHY-j1LK-iAdf-l8WD1y
LV Write Access  read/write
LV Creation host, time centos, 2016-02-20 19:27:39 -0500
LV Status        available
# open           1
LV Size          7.50 GiB
Current LE       1920
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:2

--- Logical volume ---
LV Path          /dev/vg00/vol_backups
LV Name          vol_backups
VG Name          vg00
LV UUID          E5xsjt-1KZ1-nP5d-qhNX-KzQi-trhv-67Nd9c
LV Write Access  read/write
LV Creation host, time centos, 2016-02-20 19:27:47 -0500
LV Status        available
# open           1
LV Size          8.49 GiB
Current LE       2174
Segments         3
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:3

[root@server ~]#
```

Fig 1: Two Logical Volumes Named `vol_projects` and `vol_backups`

After creating the **LVs**, we are ready to define the **LUNs** in the **Target** in order to make them available for the client machine.

As shown in **Fig. 2**, we will open a `targetcli` shell and issue the following commands, which will create two block backstores (local storage resources that represent the LUN the initiator will actually use) and an **Iscsi Qualified Name (IQN)**, a method of addressing the target server.

Please refer to [Page 32 of RFC 3720](#) for more details on the structure of the IQN. In particular, the text after the colon character (:tgt1) specifies the name of the target, while the text before (server:) indicates the hostname of the target inside the domain.

```
# targetcli
# cd backstores
# cd block
# create server.backups /dev/vg00/vol_backups
# create server.projects /dev/vg00/vol_projects
# cd /iscsi
# create iqn.2016-02.com.tecmint.server:tgt1
[root@server ~]# targetcli
targetcli shell version 2.1.fb41
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> cd backstores
/backstores> cd block
/backstores/block> create server.backups /dev/vg00/vol_backups
Created block storage object server.backups using /dev/vg00/vol_backups.
/backstores/block> create server.projects /dev/vg00/vol_projects
Created block storage object server.projects using /dev/vg00/vol_projects.
/backstores/block> cd /iscsi
/iscsi> create iqn.2016-02.com.tecmint.server:tgt1
Created target iqn.2016-02.com.tecmint.server:tgt1.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/iscsi> █
```

Fig 2: Define LUNs in Target Server

With the above step, a new **TPG (Target Portal Group)** was created along with the default portal (a pair consisting of an IP address and a port which is the way initiators can reach the target) listening on port **3260** of all IP addresses.

If you want to bind your portal to a specific IP (the Target's main IP, for example), delete the default portal and create a new one as follows (otherwise, skip the following **targetcli** commands. **Note that for simplicity we have skipped them as well**):

```
# cd /iscsi/inq.2016-02.com.tecmint.server:tgt1/tpg1/portals
# delete 0.0.0.0 3260
# create 192.168.0.29 3260
```

Now we are ready to proceed with the creation of LUNs. Note that we are using the backstores we previously created (`server.backups` and `server.projects`). This process is illustrated in Fig. 3:

```
# cd iqn.2016-02.com.tecmint.server:tgt1/tpg1/luns
# create /backstores/block/server.backups
# create /backstores/block/server.projects
```

```
/iscsi> cd iqn.2016-02.com.tecmint.server:tgt1/tpg1/luns  
/iscsi/inq.20...gt1/tpg1/luns> create /backstores/block/server.backups  
Created LUN 0.  
/iscsi/inq.20...gt1/tpg1/luns> create /backstores/block/server.projects  
Created LUN 1.  
/iscsi/inq.20...gt1/tpg1/luns> █
```

Fig 3: Create LUNs in iSCSI Target Server

The last part in the Target configuration consists of creating an [Access Control List to restrict access](#) on a per-initiator basis. Since our client machine is named “client”, we will append that text to the IQN. Refer to **Fig. 4** for details:

```
# cd .. /acls  
# create iqn.2016-02.com.tecmint.server:client  
/iscsi/inq.20...gt1/tpg1/luns> cd .. /acls  
/iscsi/inq.20...gt1/tpg1/acls> create iqn.2016-02.com.tecmint.server:client  
Created Node ACL for iqn.2016-02.com.tecmint.server:client  
Created mapped LUN 1.  
Created mapped LUN 0.
```

Fig 4: Create Access Control List for Initiator

At this point we can the **targetcli** shell to show all configured resources, as we can see in **Fig. 5**:

```
# targetcli  
# cd /  
# ls
```

```
[root@server ~]# targetcli
targetcli shell version 2.1.fb41
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/backstores/b...erver.backups> cd /
/> ls
o- /
  o- backstores
    | o- block ...
    | | o- server.backups ...
    | | o- server.projects ...
    | o- fileio ...
    | o- pscsi ...
    | o- ramdisk ...
  o- iscsi
    | o- iqn.2016-02.com.tecmint.server:tgt1
      o- tpg1 ...
        o- acls
          | o- iqn.2016-02.com.tecmint.server:client
            | o- mapped_lun0 ...
            | o- mapped_lun1 ...
          o- luns
            | o- lun0 ...
            | o- lun1 ...
          o- portals
            o- 0.0.0.0:3260
  o- loopback ...
/> [bl
```

Fig 5: User targetcli to Check Configured Resources

To quit the **targetcli** shell, simply type **exit** and press **Enter**. The configuration will be saved automatically to **/etc/target/saveconfig.json**.

As you can see in **Fig. 5** above, we have a portal listening on port **3260** of all IP addresses as expected. We can verify that using [netstat command](#) (see **Fig. 6**):

```
# netstat -npltu | grep 3260
[root@server ~]# netstat -tulnp | grep 3260
tcp      0      0 0.0.0.0:3260          0.0.0.0:*          LISTEN      -
[root@server ~]#
```

Fig 6: Verify iSCSI Target Server Port Listening

This concludes the **Target** configuration. Feel free to restart the system and verify that all settings survive a reboot. If not, make sure to open the necessary ports in the firewall configuration and to start the target service on boot. We are now ready to set up the Initiator and to connect to the client.

### Step 3: Setting up the Client Initiator

In the client we will need to install the **iscsi-initiator-utils** package, which provides the server daemon for the iSCSI protocol (**iscsid**) as well as **iscsiadm**, the administration utility:

```
# yum update && yum install iscsi-initiator-utils
```

Once the installation completes, open **/etc/iscsi/initiatorname.iscsi** and replace the default initiator name (commented in **Fig. 7**) with the name that was previously set in the **ACL** on the server (**iqn.2016-02.com.tecmint.server:client**).

Then save the file and run **iscsiadm** in discovery mode pointing to the target. If successful, this command will return the target information as shown in **Fig. 7**:

```
# iscsiadm -m discovery -t st -p 192.168.0.29
#InitiatorName=iqn.1994-05.com.redhat:39b330a79bcc
InitiatorName=iqn.2016-02.com.tecmint.server:client
```



```
[root@client ~]# iscsiadm -m discovery -t st -p 192.168.0.29
192.168.0.29:3260,1 iqn.2016-02.com.tecmint.server:tgt1
[root@client ~]#
```

**/etc/iscsi/initiatorname.iscsi**

Fig 7: Setting Up Client Initiator

The next step consists in restarting and enabling the **iscsid** service:

```
# systemctl start iscsid
# systemctl enable iscsid
```

and contacting the target in node mode. This should result in **kernel-level** messages, which when captured through **dmesg show** the device identification that the remote LUNs have been given in the local system (**sde** and **sdf** in **Fig. 8**):

```
# iscsiadm -m node -T iqn.2016-02.com.tecmint.server:tgt1 -p 192.168.0.29 -l
# dmesg | tail
[root@client ~]# iscsiadm -m node -T iqn.2016-02.com.tecmint.server:tgt1 -p 192.168.0.29 -l
Logging in to [iface: default, target: iqn.2016-02.com.tecmint.server:tgt1, portal: 192.168.0.29,3260] (mult
Login to [iface: default, target: iqn.2016-02.com.tecmint.server:tgt1, portal: 192.168.0.29,3260] successful
```



```
[root@client ~]# dmesg | tail
[ 1267.240785] sd 9:0:0:1: Attached scsi generic sg6 type
[ 1267.245328] sd 9:0:0:0: [sde] Write cache: enabled, r
[ 1267.245805] sd 9:0:0:1: [sdf] 15728640 512-byte logic
[ 1267.263945] sd 9:0:0:1: [sdf] Write Protect is off
[ 1267.263951] sd 9:0:0:1: [sdf] Mode Sense: 43 00 10 08
[ 1267.267680] sd 9:0:0:1: [sdf] Write cache: enabled, r
[ 1267.353148] sde: unknown partition table
[ 1267.354530] sdf: unknown partition table
[ 1267.372951] sd 9:0:0:1: [sdf] Attached SCSI disk
[ 1267.372965] sd 9:0:0:0: [sde] Attached SCSI disk
```

Fig 8: Connecting to iSCSI Target Server in Node Mode

From this point on, you can create partitions, or even LVs (and filesystems on top of them) as you would do with any other storage device. For simplicity, we will create a primary partition on each disk that will occupy its entire available space, and format it with ext4.

Finally, let's mount **/dev/sde1** and **/dev/sdf1** on **/projects** and **/backups**, respectively (note that these directories must be created first):

```
# mount /dev/sde1 /projects
# mount /dev/sdf1 /backups
```

Additionally, you can add two entries in **/etc/fstab** in order for both filesystems to be mounted automatically at boot using each filesystem's **UUID** as returned by **blkid**.

**Note** that the **\_netdev** mount option must be used in order to defer the mounting of these filesystems until the network service has been started:

```
[root@client ~]# blkid /dev/sde1
/dev/sde1: UUID="154feeb4-ad76-4a00-ba9c-02f4d05a3aca" TYPE="ext4"
[root@client ~]# blkid /dev/sdf1
/dev/sdf1: UUID="4d1cb829-156c-4753-a66d-2999b37881ff" TYPE="ext4"
[root@client ~]#
```

```
UUID=154feeb4-ad76-4a00-ba9c-02f4d05a3aca /projects ext4 defaults 0 0
UUID=4d1cb829-156c-4753-a66d-2999b37881ff /backups ext4 defaults 0 0
```

Fig 9: Find Filesystem UUID

You can now use these devices as you would with any other storage media.

## Summary

In this article we have covered how to set up and configure an iSCSI Target and an Initiator in **RHEL/CentOS 7** distributions. Although the first task is not part of the required competencies of the **EX300 (RHCE)** exam, it is needed in order to implement the second topic.

Don't hesitate to let us know if you have any questions or comments about this article – feel free to drop us a line using the comment form below.

Looking to setup iSCSI Target and Client Initiator on RHEL/CentOS 6, follow this guide: [Setting Up Centralized iSCSI Storage with Client Initiator](#).

## Setting Up “NTP (Network Time Protocol) Server” in RHEL/CentOS 7

**Network Time Protocol** – NTP- is a protocol which runs over port 123 UDP at Transport Layer and allows computers to synchronize time over networks for an accurate time. While time is passing by, computers internal clocks tend to drift which can lead to inconsistent time issues, especially on servers and clients logs files or if you want to replicate servers resources or databases.



# Setting Up NTP Server in RHEL/CentOS 7

NTP Server Installation in CentOS and RHEL 7

### Requirements:

1. [CentOS 7 Installation Procedure](#)
2. [RHEL 7 Installation Procedure](#)

### Additional Requirements:

1. [Register and Enable RHEL 7 Subscription for Updates](#)
2. [Configure Static IP Address on CentOS/Rhel 7](#)
3. [Disable and Remove Unwanted Services in CentOS/RHEL 7](#)

This tutorial will demonstrate how you can install and configure **NTP** server on **CentOS/RHEL 7** and automatically synchronize time with the closest geographically peers available for your server location by using NTP Public Pool Time Servers list.

### Step 1: Install and configure NTP daemon

**1.** NTP server package is provided by default from official **CentOS /RHEL 7** repositories and can be installed by issuing the following command.

```
# yum install ntp
[root@localhost ~]# yum install ntp
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.xservers.ro
 * extras: mirrors.biztelecom.ro
 * updates: mirrors.biztelecom.ro
Resolving Dependencies
--> Running transaction check
--> Package ntp.x86_64 0:4.2.6g5-18.el7.centos will be installed
--> Processing Dependency: ntpdate = 4.2.6g5-18.el7.centos for package: ntp-4.2.6g5-18.el7.centos.x86_64
--> Processing Dependency: libiptc.so.25()(64bit) for package: ntp-4.2.6g5-18.el7.centos.x86_64
--> Running transaction check
--> Package autogen-libiptc.x86_64 0:5.10-5.el7 will be installed
--> Package ntpdate.x86_64 0:4.2.6g5-18.el7.centos will be installed
--> Finished Dependency Resolution

Dependencies Resolved

-----  
Package           Arch      Version            Repository  Size  
installing:  
ntp               x86_64   4.2.6g5-18.el7.centos    base       539 k  
installing for dependencies:  
autogen-libiptc  x86_64   5.10-5.el7                    base       66 k  
ntpdate          x86_64   4.2.6g5-18.el7.centos    base       82 k  
  
Transaction Summary  
-----  
Install 1 Package (+2 Dependent packages)  
  
Total download size: 687 k  
Installed size: 1.6 M  
Is this ok (y/d/N): y  
  
http://www.tecmint.com
```

## Install NTP Server

**2.** After the server is installed, first go to official [NTP Public Pool Time Servers](http://www.pool.ntp.org/), choose your **Continent** area where the server physically is located, then search for your **Country** location and a list of NTP servers should appear.

The screenshot shows the homepage of the NTP pool for Romania. At the top, there's a navigation bar with links like 'JOIN THE POOL', 'USE THE POOL', and 'MANAGE SERVERS'. Below the navigation, the text 'Romania — ro.pool.ntp.org' is displayed, followed by a message encouraging users to join the pool if they have a static IP. A code block shows the configuration for the NTP pool, listing four servers: 'server 0.ro.pool.ntp.org', 'server 1.ro.pool.ntp.org', 'server 2.ro.pool.ntp.org', and 'server 3.ro.pool.ntp.org'. Further down, there are sections for 'IPv4' and 'IPv6', each listing active servers with their last update time. A note at the bottom says 'See all zones' with a link to <https://www.tecmint.com>.

## NTP Pool Server

3. Then open NTP daemon main configuration file for editing, comment the default list of Public Servers from **pool.ntp.org** project and replace it with the list provided for your country like in the screenshot below.

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst

server 0.ro.pool.ntp.org iburst
server 1.ro.pool.ntp.org iburst
server 2.ro.pool.ntp.org iburst
server 3.ro.pool.ntp.org iburst
http://www.tecmint.com
```

## Configure NTP Server

4. Further, you need to allow clients from your networks to synchronize time with this server. To accomplish this, add the following line to NTP configuration file, where **restrict** statement controls, what network is allowed to query and sync time – replace network IPs accordingly.

```
restrict 192.168.1.0 netmask 255.255.255.0 nomodify notrap
```

The **nomodify notrap** statements suggest that your clients are not allowed to configure the server or be used as peers for time sync.

5. If you need additional information for troubleshooting in case there are problems with your NTP daemon add a log file statement which will record all NTP server issues into one dedicated log file.

```
logfile /var/log/ntp.log
```

```
GNU nano 2.3.1                                         File: /etc/ntp.conf

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8

# Enable writing of statistics records.
#statistics clockstats cryptostats loopstats peerstats

# Disable the monitoring facility to prevent amplification attacks using ntpdc
# monlist command when default restrict does not include the noquery flag. See
# CVE-2013-5211 for more details.
# Note: Monitoring will not be disabled with the limited restriction flag.
#disable monitor

logfile /var/log/ntp.log
```

<http://www.tecmint.com>

## Enable NTP Logs

- After you have edited the file with all configuration explained above save and close **ntp.conf** file. Your final configuration should look like in the screenshot below.

```
GNU nano 2.3.1                                         File: /etc/ntp.conf

# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default nomodify notrap nopeer noquery

# Permit all access over the lohpach interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict ::1

# Hosts on local network are less restricted.
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

allow 192.168.1.0/24 network clients to synchronize time with this server
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst

server 0.ro.pool.ntp.org iburst
server 1.ro.pool.ntp.org iburst
server 2.ro.pool.ntp.org iburst
server 3.ro.pool.ntp.org iburst

broadcast 192.168.1.255 autokey      # broadcast server
broadcastclient                         # broadcast client
broadcast 224.0.1.1 autokey           # multicast server
multicastclient 224.0.1.1              # multicast client
unicastserver 239.255.254.254         # unicast server
unicastclient 239.255.254.254 autokey # unicast client

# Enable public key cryptography.
#crypto
```

## NTP Server Configuration

### Step 2: Add Firewall Rules and Start NTP Daemon

7. NTP service uses UDP port **123** on OSI transport layer (layer 4). It is designed particularly to resist the effects of variable latency (jitter). To open this port on RHEL/CentOS 7 run the following commands against Firewalld service.

```
# firewall-cmd --add-service=ntp --permanent
# firewall-cmd --reload
[root@pxe-server ~]# firewall-cmd --add-service=ntp --permanent
success
[root@pxe-server ~]# firewall-cmd --reload
success
[root@pxe-server ~]# _
```

<http://www.tecmint.com>

### Open NTP Port in Firewall

8. After you have opened Firewall port 123, start NTP server and make sure you enable it system-wide. Use the following commands to manage the service.

```
# systemctl start ntpd
# systemctl enable ntpd
# systemctl status ntpd
[root@localhost ~]# systemctl start ntpd
[root@localhost ~]# systemctl enable ntpd
In -s '/usr/lib/systemd/system/ntpd.service' '/etc/systemd/system/multi-user.target.wants/ntpd.service'
[root@localhost ~]# systemctl status ntpd
ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled)
   Active: active (running) since Sun 2014-09-14 16:28:41 EEST; 12s ago
     Main PID: 19471 (ntpd)
        CGroup: /system.slice/ntpd.service
                  └─19471 /usr/sbin/ntpd -u ntp:ntp -g

Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: proto: precision = 0.015 usec
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: 0.0.0.0 c01d 0d kern kernel time sync enabled
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: ntp_io: estimated max descriptors: 1024, initial socket boundary: 16
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listen and drop on 1 v6wildcard :: UDP 123
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listen normally on 2 lo 127.0.0.1 UDP 123
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listen normally on 3 eno1677736 192.168.1.25 UDP 123
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listen normally on 4 lo ::1 UDP 123
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: Listening on routing socket on fd #21 for interface updates
Sep 14 16:28:41 pxe-server.centos.lan systemd[1]: Started Network Time Service.
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: 0.0.0.0 c016 06 restart
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: 0.0.0.0 c012 02 freq_set kernel 0.000 PPM
Sep 14 16:28:41 pxe-server.centos.lan ntpd[19471]: 0.0.0.0 c011 01 freq_not_set
Sep 14 16:28:42 pxe-server.centos.lan ntpd[19471]: 0.0.0.0 c614 04 freq_mode
[root@localhost ~]#
```

<http://www.tecmint.com>

### Start NTP Service

### Step 3: Verify Server Time Sync

**9.** After NTP daemon has been started, wait a few minutes for the server to synchronize time with its pool list servers, then run the following commands to verify NTP peers synchronization status and your system time.

```
# ntpq -p
# date -R
[root@pxe-server ~]# ntpq -p
      remote          refid      st t when poll reach   delay    offset  jitter
=====+
*smtp.linux-expe 80.96.120.252    2 u 104 128 377    2.773   4.958   1.278
*main-fe0.b.astr 131.188.3.222    2 u 105 128 377   14.255   2.240   5.601
*shine3.bluepink 160.45.10.8     2 u   30 128 377    6.650   4.066   0.934
[root@pxe-server ~]# date
Sun Sep 14 22:32:09 EEST 2014
[root@pxe-server ~]#
```

<http://www.tecmint.com>

### Verify NTP Time Sync

**10.** If you want to query and synchronize against a pool of your choice use **ntpdate** command, followed by the server or servers addresses, as suggested in the following command line example.

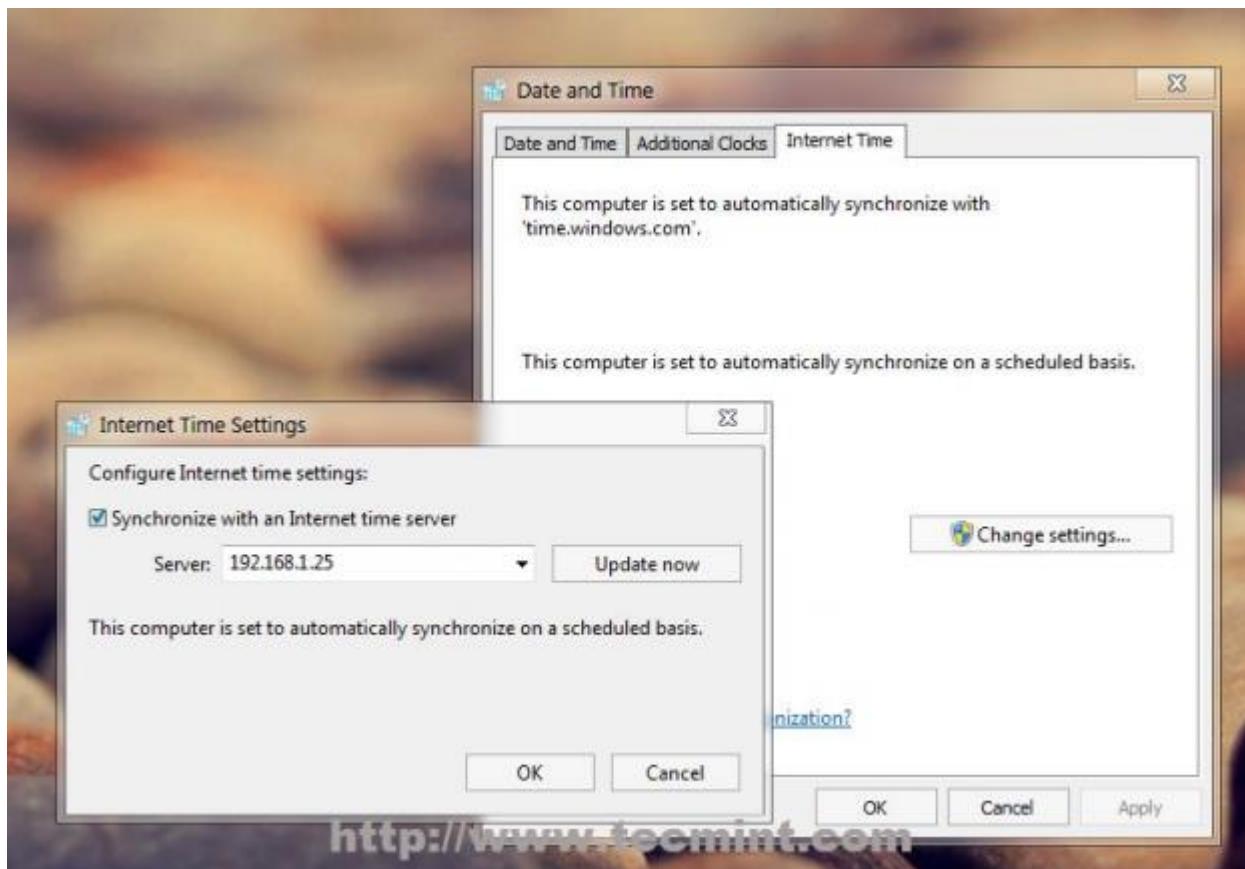
```
# ntpdate -q 0.ro.pool.ntp.org 1.ro.pool.ntp.org
[root@pxe-server ~]# ntpdate -q 0.ro.pool.ntp.org 1.ro.pool.ntp.org
server 93.190.144.19, stratum 2, offset -0.003678, delay 0.02756
server 91.216.151.201, stratum 2, offset -0.003264, delay 0.03261
server 91.208.179.1, stratum 2, offset -0.006003, delay 0.03307
server 176.126.213.2, stratum 2, offset -0.003138, delay 0.03677
server 89.149.54.18, stratum 2, offset -0.003436, delay 0.04388
server 78.96.7.8, stratum 2, offset 0.013856, delay 0.07767
server 78.96.7.25, stratum 2, offset 0.007590, delay 0.06664
server 89.149.54.30, stratum 2, offset -0.004126, delay 0.04311
14 Sep 22:50:00 ntpdate[3394]: adjust time server 91.216.151.201 offset -0.003264 sec
[root@pxe-server ~]_
```

<http://www.tecmint.com>

### Synchronize NTP Time

#### Step 4: Setup Windows NTP Client

**11.** If your windows machine is not a part of a Domain Controller you can configure Windows to synchronize time with your NTP server by going to Time from the right side of **Taskbar -> Change Date and Time Settings -> Internet Time tab -> Change Settings -> Check Synchronize with an Internet time server** -> put your server's IP or FQDN on **Server** field -> **Update now -> OK**.



## Synchronize Windows Time with NTP

That's all! Setting up a local NTP Server on your network ensures that all your servers and clients have the same time set in case of an Internet connectivity failure and they all are synchronized with each other.