Red Hat Certified System Administrator (RHCSA) Cheat Sheet

Fabrizio Di Carlo

Contents

Disclaimer

Hi! This notes you're reading is an Alpha Release of the final notes, which should be ready for "the press" for June 2013. If you are reading this, most likely you have got to make editing.

If this is the case, please open an account on GitHub.com and send me your user on `dicarlo.fabrizio@gmail.com`. So that I can enable you to reading and edit the text version of the notes. You will find this version, once authorized, at:
`https://github.com/fdicarlo/RHCSA_cs`

Thanks a lot for your edit!

Fabrizio.

Why this notes

Times ago some of my friends began to tell me to start the Red Hat Certification Program[1], I'm not a SysAdmin (I came from Engineering in Computer Science and I want to become a UX guy) but I know Linux and its power, I started to use it when I was 14 and now I'm 26, so I googled about it.

The Red Hat Certification Program are IT Professional certifications for Red Hat products and general Linux related skills such as system administration on Red Hat Enterprise Linux, all certifications are given after passing exams. The program distinguishes itself in that the exams are performance-based, meaning that students must perform tasks on a live system, rather than answering multiple choice questions.

RHCSA[2] is the entry-level certification that focuses on actual competencies at system administration, including installation and configuration of a Red Hat Linux system and attaching it to a live network running network services. To achieve the RHCSA certification the student must pass EX200, a half-day hands-on lab exam. The minimum passing score for the exam is 210 out of 300 possible points (70%). There is no prerequisite for the exam, but Red Hat recommends preparing for the exam by taking courses in Red Hat System Administration (RH124 or RH135) if one does not have previous experience. RHCSA was launched in 2002 as Red Hat Certified Technician (RHCT). As of July 2009 there were 30,000 RHCTs. In November 2010 it was renamed to RHCSA.

I googled also about some notes, Cheat sheet or books but except some valid books[3] these is nothing that I can't consult on

[1] `https://www.redhat.com/training/courses/`

[2] `https://www.redhat.com/training/courses/ex200/`

[3] Michael Jang's RHCSA/RHCE Red Hat Linux Certification

my eBook's reader or that I can share with my friends, so I started to write a collaborative notes on GitHub.

Some details

As I said I'm not a SysAdmin but I'm simple Linux passionate, I wrote (and I'm writing) this ebook not for money but following my passion, my knowledge and the "Exam objectives" so, for sure, you can find some mistakes or something wrong, please send me a mail or update the notes.

Study Guide (Exams EX200 & EX300) `http://www.amazon.com/RHCSA-Linux-Certification-Study-Edition/dp/0071765654` and Damian Tommasino's Hands-on Guide to the Red Hat Exams: RHCSA and RHCE Cert Guide and Lab Manual `http://www.amazon.com/Hands-Guide-Red-Exams-Certification/dp/0321767950`

Red Hat Certified System Administrator (RHCSA) Exam objectives[4]

Red Hat reserves the right to add, modify, and remove objectives. Such changes will be made public in advance through revisions to this document.

RHCSA exam candidates should be able to accomplish the tasks below without assistance. These have been grouped into several categories.

Understand and use essential tools:

- Access a shell prompt and issue commands with correct syntax.
- Use input-output redirection (>, >>, |, 2>, etc.).
- Use grep and regular expressions to analyze text.
- Access remote systems using ssh and VNC.
- Log in and switch users in multiuser runlevels.
- Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2.
- Create and edit text files.
- Create, delete, copy, and move files and directories.
- Create hard and soft links.
- List, set, and change standard ugo/rwx permissions.

[4]Red Hat Certified System Administrator (RHCSA) Exam objectives (EX200): `https://www.redhat.com/training/courses/ex200/examobjective`

- Locate, read, and use system documentation including man, info, and files in /usr/share/doc.

Note: Red Hat may use applications during the exam that are not included in Red Hat Enterprise Linux for the purpose of evaluating candidate's abilities to meet this objective.

Operate running systems:

- Boot, reboot, and shut down a system normally.
- Boot systems into different runlevels manually.
- Use single-user mode to gain access to a system.
- Identify CPU/memory intensive processes, adjust process priority with renice, and kill processes.
- Locate and interpret system log files.
- Access a virtual machine's console.
- Start and stop virtual machines.
- Start, stop, and check the status of network services.

Configure local storage:

- List, create, delete, and set partition type for primary, extended, and logical partitions.
- Create and remove physical volumes, assign physical volumes to volume groups, and create and delete logical volumes.
- Create and configure LUKS-encrypted partitions and logical volumes to prompt for password and mount a decrypted file system at boot.

- Configure systems to mount file systems at boot by Universally Unique ID (UUID) or label.
- Add new partitions and logical volumes, and swap to a system non-destructively.

Create and configure file systems:

- Create, mount, unmount, and use ext2, ext3, and ext4 file systems.
- Mount, unmount, and use LUKS-encrypted file systems.
- Mount and unmount CIFS and NFS network file systems.
- Configure systems to mount ext4, LUKS-encrypted, and network file systems automatically.
- Extend existing unencrypted ext4-formatted logical volumes.
- Create and configure set-GID directories for collaboration.
- Create and manage Access Control Lists (ACLs).
- Diagnose and correct file permission problems.

Deploy, configure, and maintain systems:

- Configure networking and hostname resolution statically or dynamically.
- Schedule tasks using cron.
- Configure systems to boot into a specific runlevel automatically.

- Install Red Hat Enterprise Linux automatically using Kickstart.
- Configure a physical machine to host virtual guests.
- Install Red Hat Enterprise Linux systems as virtual guests.
- Configure systems to launch virtual machines at boot.
- Configure network services to start automatically at boot.
- Configure a system to run a default configuration HTTP server.
- Configure a system to run a default configuration FTP server.
- Install and update software packages from Red Hat Network, a remote repository, or from the local file system.
- Update the kernel package appropriately to ensure a bootable system.
- Modify the system bootloader.

Manage users and groups:

- Create, delete, and modify local user accounts.
- Change passwords and adjust password aging for local user accounts.
- Create, delete, and modify local groups and group memberships.
- Configure a system to use an existing LDAP directory service for user and group information.

Manage security:

- Configure firewall settings using system-config-firewall or iptables.
- Set enforcing and permissive modes for SELinux.
- List and identify SELinux file and process context.
- Restore default file contexts.
- Use boolean settings to modify system SELinux settings.
- Diagnose and address routine SELinux policy violations.

Understand and use essential tools

Access a shell prompt and issue commands with correct syntax

This is first requirement should stop anyone who may not know, or may have never used a shell prompt from attempting the test. If you can open your terminal, navigate and type commands then you have accomplished this. If not, then you should check out the basics and start there.

Alternatively

Ctrl+Alt+F1 to **F6** are the virtual consoles provided by the getty/agetty programs. Ctrl+Alt+F7 is the console where your X server is running. The GUI (Gnome/KDE or any other) runs over X. So to get back into your GUI window manager: type: **Ctrl+Alt+F7**.

Use input-output redirection (>, >>, |, 2>, etc.)

Input output redirection is one of the base skills you will need as a sysadmin. On the exam you will have to be able to redirect data from one command into another, and/or into a file.

Some examples:

```
$ echo "this is input" > file.txt
```

or

```
$ cat /var/log/messages | less
```

You can easily redirect input / output to any file other than the screen. This is achieved in Linux using input and output redirection symbols:

- “>” Output redirection
- “<” Input redirection

Using a combination of these symbols and the standard file descriptors you can achieve complex redirection tasks quite easily.

- “>” overwright
- “<” send into a command or file
- “>>” append
- “<<” append into a command or file
- “|” funnel into
- “2>” redirect errors
- “2>&1” redirect errors to std out

Use grep and regular expressions to analyze text

RHCSA requirements state that you must know how to use grep to analyze text. This is actually going to be pretty necessary to do many administration tasks on a daily basis.

Grep returns any lines that have characters, words, or expressions that match your query.

Basic usage examples of this include:

- Find “Permission Denied” entries in a log file

  $ grep -r “Permission Denied” /path/to/logfile/

- Find “Permission Denied” entries in a log file by using output redirection

  $ cat /path/to/file/ | grep “Permission Denied”

Access remote systems using ssh and VNC

SSH

SSH is such an integrated part of this exam that its kind of weird that this is one of the official requirements. But nonetheless there are a number of different options that you can apply to make you more efficient in exam.

- Basic ssh access is simple:

  $ ssh user@host

- ssh to a custom port:

  $ ssh -p port_number user@host

- ssh bringing X (required to run programs like system-config-users remotely)

  $ ssh -X user@host

- ssh as another user (another way)

  $ ssh -l user host

- Display debugging messages as it connects. Useful if you have having some issues connecting to a certain machine.

  $ ssh -v user@host

Those are the main options for ssh, as always "man ssh" to see all the other magic.

VNC

On the remote machine, that you will be connecting to, you should have tigervnc-server installed.

```
$ yum install tigervnc-server
```

This puts a config file on your remote machine in /etc/sysconfig/vncservers

```
VNCSERVERS="2:myusername"
VNCSERVERARGS[2]="-geometry 800x600 -nolisten tcp -nohttpd"
```

Aside from changing "username" you want it to look like this. All we did to change it, is remove the "-localhost" directive. This would have restricted us from connecting from a remote system without a tunnel setup. Since this is an exam and not the real world, we can disable that.

Set up your password on the remote machine by running

```
$ vncpasswd
```

And finally start your vncserver

```
$ vncserver :1
```

The output should look like this:

```
[root@rhel6 ~]# vncserver :1
New 'rhel6.local:1 (root)' desktop is rhel6.local:1
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/rhel6.local:1.log
```

The default vnc client on Red Hat Enterprise Linux 6 is tigervnc. If it is not already installed on the system:

```
$ yum install tigervnc
```

To connect to the newly setup vncserver just type:

```
$ vncviewer rhel6.local:5901
```

(replace rhel6.local with your remote host)

Log in and switch users in multiuser runlevels.

If you have followed along to this point, you have logged in, and most likely been in either runlevel 3 or runlevel 5. Runlevels determine how much of the systems services are actually running. Most common runlevel for servers is going to be 3, most services that are not GUI oriented (including the Gnome Desktop) are turned off. Runlevel 5 is what you see when you boot into the desktop environment.

Switching between these levels is fairly straightforward. To switch to runlevel 3 type:

```
$ init 3
```

Then to see what runlevel you are in type:

```
$ runlevel
```

Switching between users, also straightforward. To switch to bob, assuming bob is a user on the system:

```
$ su - bob  #note: We put the dash in there to gain the users login p
            # So if I switched to root and didnt use the - operator,
            #I wouldn't have /usr/sbin in my path.
```

Switching to root is a common task.

```
$ su -
```

Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2.

tar

Create a tar file from a folder called test1:

```
$ tar cvf test1.tar test1
```

- c = create
- v = verbose
- f = file

Extract test1.tar

```
$ tar xvf test1.tar
```

- x = extract
- v = verbose
- f = file

List contents of tar archive

```
$ tar tf test1.tar
```

star

Man page for star:

```
man star
```

gzip

This is most commonly used in combination with tar, using the z switch. Tar itself does not compress, it just packs.

```
$ tar cvzf test1.tar.gz test1
```

Although it can be used by itself

```
$ gzip test1
```

gunzip test1.gz

note that this does not preserve the .gz file, it extracts it and removes it.

bzip2

bzip2 uses a different algorithm to compress files than the other tools, but very similar options

Create a bzip2 file

```
$ bzip2 test1
```

note that this does not preserve the original file(s), it will compress and delete the uncompressed version also does not compress directories, only files. Check out the man page:

```
man bzip2
```

Create and edit text files.

Using a command line text editor is a skill that is absolutely necessary. Without it most tasks cannot be performed. The default editor used it vim. vim is a enhanced version of vi, which is not quite as pretty as vim.

To use vim simply type vim and then the filename

```
$ vim filename.txt
```

There are options that can be given such as:

- -R Open in read-only
- -b Start in binary mode

There is also a list of options for using vim once you are editing, to move around and actually edit. It may seem cumbersome at first, but once you are used to vim you will love it.

Check out the man page for more options:

```
man vim
```

Create, delete, copy, and move files and directories.

Administering a system requires moving, copying, and deleting files and directories. These are tasks that you will encounter on a constant basis and are essential to the RHCE.

Some of the most important commands are the ones that we will list below.

ls - List contents of a directory

list the contents of the home directory.

```
$ ls /home/
```

cp - Copy a file or group of files to another location on the machine.

copy file1 as file2

```
$ cp file1 file2
```

mv - Move a file or directory

move a directory to the /tmp directory

```
$ mv directory /tmp/
```

cd - Change directory

- navigate into the /home/ directory
  $ cd /home/
- navigate from home into the /etc directory, using the .. to reverse out of the directory
  $ cd ../etc/

rm - remove files or directories.

- remove file1
  $ rm file1
- remove directory with all contents (Caution when using this!)
  $ rm -rf directory1/

touch - create a new blank file

- create a blank file named myfile.txt

  $ touch myfile.txt

mkdir - create a new directory

- create a directory in the present working directory

  $ mkdir directory1

pwd - Get the present working directory

- find the present working directory. Handy when you need to see where in the system you are.

  $ pwd
  $ /home/david/

head- Display first lines of a file, default to 10 lines

- display the first 10 lines of file1

  head file1

  ::bash

- display the first 50 lines of file1

  head -50 file1

tail -Display last lines of a file, default to 10 lines

- display the last 10 lines of file1

  tail file1

- display the last 50 lines of file1

  tail -50 file1

Create hard and soft links.

Hard Links

A hard link is a link where two files are really the same file.

Watch how when we create a file, and link to it with a hard link, the inodes (exact location on the harddisk) are the same.

```
$ touch file.txt
$ ln file.txt file1.txt
$ ls -li file*
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file.txt
```

When we create a third file linking it to the original, we see the same thing. They all are have an inode of 524594.

```
$ ln file.txt file2.txt
$ ls -li file*
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file2.txt
524594 -rw-r--r--. 3 root root 0 Mar 21 12:54 file.txt
```

What happens if we delete the original file?

```
$ rm file.txt
rm: remove regular empty file 'file.txt'? y
$ ls -li file*
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file1.txt
524594 -rw-r--r--. 2 root root 0 Mar 21 12:54 file2.txt
```

As you can see, the other two files are in tact and have not been removed, even though the original file is gone. That is because they are all the same file, when you make a hard link to it you are just putting another reference to it with a different name. Until the last file with that inode gets deleted, that file lives on.

Lets put some text in file2.txt and see what happens

```
$ echo "things" >> file2.txt
$ ls -li file*
524594 -rw-r--r--. 2 root root 7 Mar 21 13:01 file1.txt
524594 -rw-r--r--. 2 root root 7 Mar 21 13:01 file2.txt
$ cat file1.txt
things
$ cat file2.txt
things
```

As you can see, the files both grew to 7 bytes, and when we look inside each one, they both have the same text. That's because they are the same.

Soft Links

A soft link is much different from a hard link. Most people relate hard links to shortcuts in Windows. When you put a shortcut on your Desktop, it is just a link to the something on your computer. If you delete it no biggie, its just a link. Soft links are the same way.

```
$ touch testfile.txt
$ ln -s testfile.txt testfile1.txt
$ ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testf
```

```
524725 -rw-r--r--. 1 root root  0 Mar 21 13:11 testfile.txt
```

Here we created a file, testfile.txt, and then ran ln -s to create a soft link. When we ran ls -li we see that now the inodes are different, and testfile1.txt shows highlighted with an arrow to testfile.txt.

OK, so now lets repeat what we did above for hard links. I will make another soft link, linking to the original file testfile.txt and call it testfile2.txt. Then I'll delete the original and ls -li

```
$ ln -s testfile.txt testfile2.txt
$ ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testf
524727 lrwxrwxrwx. 1 root root 12 Mar 21 13:15 testfile2.txt -> testf
524725 -rw-r--r--. 1 root root  0 Mar 21 13:11 testfile.txt
$ rm testfile.txt
rm: remove regular empty file 'testfile.txt'? y
$ ls -li testfile*
524726 lrwxrwxrwx. 1 root root 12 Mar 21 13:11 testfile1.txt -> testf
524727 lrwxrwxrwx. 1 root root 12 Mar 21 13:15 testfile2.txt -> testf
```

If we try and cat the two files, to see the contents, we get an error. We can no longer access these files, they are broken links.

```
$ cat testfile*
cat: testfile1.txt: No such file or directory
cat: testfile2.txt: No such file or directory
```

List, set, and change standard ugo/rwx permissions.

Permissions rule on compooters. Controlling them, essential. Linux has a number of different tools to do this, we list the essentials for the exam here.

ls

This is one of the most common commands used when probing a filesystem. ls lists the files in a directory, and the -l switch shows permissions, ownership, size, and date modified

```
[root@rhel6 ~]# ls -l
total 28
-rw-------. 1 root root  1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 root root 15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root root  5337 Mar 21 15:37 install.log.syslog
```

chmod

Permissions are as follows:

- 1 execute
- 2 write
- 4 read

..or in letter format

- x execute
- w write
- r read

note: the first bit is reserved for type, files are -, directories are d, links are l

For example, to change all three above files to 777 or readable, writable, and executable by all:

```
# chmod changes permission bits, either with numeric or letter permis
[root@rhel6 ~]# chmod 777 ./*
[root@rhel6 ~]# ls -l
total 28
-rwxrwxrwx. 1 root root  1403 Mar 21 15:40 anaconda-ks.cfg
-rwxrwxrwx. 1 root root 15932 Mar 21 15:39 install.log
-rwxrwxrwx. 1 root root  5337 Mar 21 15:37 install.log.syslog
```

A more reasonable permissions set would be to allow others to read files, but only allow the owner to read, write, and execute.

```
[root@rhel6 ~]# chmod 644 ./*
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 root root  1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 root root 15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root root  5337 Mar 21 15:37 install.log.syslog
```

Directories usually have a similar permissions set. They allow owner to rwx, but everyone else to rx. 755 would be the numerical value.

If directories are not executable, you cannot change into them with cd. cd essentially executes itself on the directory when you use it.

If we want to use the letter format as opposed to numbers. we combine the ugo/rwx to apply permissions. To give the group permissions to execute install.log we combine g+x:

```
[root@rhel6 ~]# chmod g+x install.log
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 nobody nobody  1403 Mar 21 15:40 anaconda-ks.cfg
```

```
-rw-r-xr--. 1 david  root   15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root   david   5337 Mar 21 15:37 install.log.syslog
```

chown

chown is used to change ownership of files and directories.

Using the same group of files, we can change the owner from root to david on install.log.

```
[root@rhel6 ~]# chown david.david install.log
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 root  root   1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 david david 15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root  root   5337 Mar 21 15:37 install.log.syslog
```

We can also change just group on a file, to allow the group certain permissions. Here we change install.log.syslog to be owned by group david, but still owner is root.

```
[root@rhel6 ~]# chown root.david install.log.syslog
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 root  root   1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 david david 15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root  david  5337 Mar 21 15:37 install.log.syslog
```

If we don't want anyone to see have access, we could change it to a user like nobody. In this case, everyone would be able to read it, but nobody could write and execute anaconda-ks.config.

```
[root@rhel6 ~]# chown nobody.nobody anaconda-ks.cfg
```

```
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 nobody nobody  1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 david  david  15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root   david   5337 Mar 21 15:37 install.log.syslog
```

chgrp

chgrp does the same thing as chown does, except it only changes the group. Handy if you just want to apply group permissions to a group of files that have various owners.

```
[root@rhel6 ~]# chgrp root install.log
[root@rhel6 ~]# ls -l
total 28
-rw-r--r--. 1 nobody nobody  1403 Mar 21 15:40 anaconda-ks.cfg
-rw-r--r--. 1 david  root   15932 Mar 21 15:39 install.log
-rw-r--r--. 1 root   david   5337 Mar 21 15:37 install.log.syslog
```

Locate, read, and use system documentation including man, info, and files in /usr/share/doc.

Man pages, docs, and info are all saving graces on the exams (Thanks to Gianluca for this tips).

The most commonly used help pages are man pages.

```
$ man vim
```

This will give you the manual pages, with descriptions of the options, examples, and information about the application. If you don't remember exactly what man page you need, but you know what utility you are trying to use you can search man pages.

For example, lets find all man pages relating to Ruby.

```
[root@rhel6 ~]# man -k ruby
erb (1)              - an embedded Ruby language interpreter
erb1.8 (1)           - an embedded Ruby language interpreter
erb1.9.1 (1)         - an embedded Ruby language interpreter
gem (1)              - the front end to RubyGems
gem1.8 (1)           - the front end to RubyGems
gem1.9.1 (1)         - the front end to RubyGems
irb (1)              - interactive ruby
irb1.8 (1)           - interactive ruby
irb1.9.1 (1)         - interactive ruby
rake1.9.1 (1)        - a ruby build program with capabilities similar
rdoc (1)             - Generate documentation from Ruby script files
rdoc1.8 (1)          - Generate documentation from Ruby script files
rdoc1.9.1 (1)        - Generate documentation from Ruby script files
ri (1)               - Ruby Information at your fingertips
ri1.8 (1)            - Ruby Information at your fingertips
ri1.9.1 (1)          - Ruby Information at your fingertips
ruby (1)             - Interpreted object-oriented scripting language
ruby1.8 (1)          - Interpreted object-oriented scripting language
ruby1.9.1 (1)        - Interpreted object-oriented scripting language
testrb (1)           - Automatic runnter for Test::Unit of Ruby
testrb1.8 (1)        - Automatic runnter for Test::Unit of Ruby
testrb1.9.1 (1)      - Automatic runnter for Test::Unit of Ruby
```

This is helpful output of results from the search. Really helpful in situations that you forgot the name of a certain utility.

Info is nearly identical, referencing the info docs. Its not quite as nice to use, and therefore is not as popular.

You could also get information from the /usr/share/docs. Here you can find other information about the program itself, or that particular version. The following output is a typical doc directory.

```
[root@rhel6 yum-3.2.27]# pwd
/usr/share/doc/yum-3.2.27
[root@rhel6 yum-3.2.27]# ls
AUTHORS  ChangeLog  COPYING  INSTALL  README  TODO
```

As you can see its very different, simply text files with license, readme, install instructions, etc. For most of your referencing in an exam situation, use the man pages.

Exam tip: If you dont get any output from man pages, try running the following command, which will build the man pages.

```
# first check for the package
[root@rhel6 ~]# rpm -qi man
# then if its installed try
[root@rhel6 ~]# makewhatis &
```

Operate running systems

Boot, reboot, and shut down a system normally.

Boot systems into different runlevels manually.

Use single-user mode to gain access to a system.

Identify CPU/memory intensive processes, adjust process priority with renice, and kill processes.

Locate and interpret system log files.

Access a virtual machine's console.

Start and stop virtual machines.

Start, stop, and check the status of network services.

Configure local storage

List, create, delete, and set partition type for primary, extended, and logical partitions.

Create and remove physical volumes, assign physical volumes to volume groups, and create and delete logical volumes.

Create and configure LUKS-encrypted partitions and logical volumes to prompt for password and mount a decrypted file system at boot.

Configure systems to mount file systems at boot by Universally Unique ID (UUID) or label.

Add new partitions and logical volumes, and swap to a system non-destructively.

Create and configure file systems

Create, mount, unmount, and use ext2, ext3, and ext4 file systems.

Mount, unmount, and use LUKS-encrypted file systems.

Mount and unmount CIFS and NFS network file systems.

Configure systems to mount ext4, LUKS-encrypted, and network file systems automatically.

Extend existing unencrypted ext4-formatted logical volumes.

Create and configure set-GID directories for collaboration.

Create and manage Access Control Lists (ACLs).

Diagnose and correct file permission problems.

Deploy, configure, and maintain systems

Configure networking and hostname resolution statically or dynamically.

Schedule tasks using cron.

Configure systems to boot into a specific runlevel automatically.

Install Red Hat Enterprise Linux automatically using Kickstart.

Configure a physical machine to host virtual guests.

Install Red Hat Enterprise Linux systems as virtual guests.

Configure systems to launch virtual machines at boot.

Configure network services to start automatically at boot.

Configure a system to run a default configuration HTTP server.

Configure a system to run a default configuration FTP server.

Install and update software packages from Red Hat Network, a remote repository, or from the local file system.

Update the kernel package appropriately to ensure a bootable system.

Modify the system bootloader.

Manage users and groups

Create, delete, and modify local user accounts.

Change passwords and adjust password aging for local user accounts.

Create, delete, and modify local groups and group memberships.

Configure a system to use an existing LDAP directory service for user and group information.

Manage security

Configure firewall settings using system-config-firewall or iptables.

Set enforcing and permissive modes for SELinux.

List and identify SELinux file and process context.

Restore default file contexts.

Use boolean settings to modify system SELinux settings.

Diagnose and address routine SELinux policy violations