Red Hat Certified System Administrator (RHCSA) Cheat Sheet

Fabrizio Di Carlo

Contents

Why this notes

The Red Hat Certification Program[1] 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. Unfortunately, due to mathematical intractability of most Bayesian models, the reader is only shown simple, artificial examples. This can leave the user with a so-what feeling about Bayesian inference. In fact, this was the author's own prior opinion.

After some recent success of Red Hat, I decided to investigate the subject again but after some Google's query about the topic I was not able anything of interesing, except some valid books[3] but

[1] `http://www.redhat.com/training/certifications/`

[2] `http://www.redhat.com/training/certifications/rhcsa/`

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

these is nothing that I can't consult on my eBook's reader or that I can share with my friends, so I started to write a collaborative notes on GitHub.

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.

Using the book

The book can be read in three different ways, starting from most recommended to least recommended:

1. The most recommended option is to clone the repository to download the files to your local machine. **RHCSA Cheat Sheet** it was written to be edited and modified by anyone. It is written as a single text file using the format markdown and prepared so as to be able to automatically generate its versions in ePub and Mobi (for e-Book's readers) as well as PDF using PanDoc.

In order to function you need to have the following software installed (at least on my computer):

- Pandoc
- Geany

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`

- KindleGen

2. The second, preferred, option is to use the GitHub viewer site, which display the Cheat Sheet in the browser example. The contents are updated synchronously as commits are made to the book. You can use the Contents section above to link to the chapters.

3. **Compiled** versions are available! Look in the compile/ directory.

Edit and regeneration

To edit the text you can simply change the files:

```
$ chapters/[number of chapter].markdown  <- Single chapters
$ title.txt                              <- Title and author
$ metadata.xml  <- License and language
```

To regenerate the text for sharing there is a script:

```
$ ruby build.rb
```

The language MultiMarkdown to handle text in the book is quite easy to use and practice should be understandable without great effort. A guide to the use of language is also available here:

- Pandoc's markdown

Development

This book has an unusual development design. The content is open-sourced, meaning anyone can be an author. Authors submit content or revisions using the GitHub interface.

What to contribute?

The current chapter list is not finalized. If you see something that is missing (Bash, networks, commands, tricks etc.), feel free to start there.

- Cleaning up code and making code more polish
- Giving better explanations
- Spelling/grammar mistakes
- Suggestions
- Contributing to the RH notebook styles

Commiting

- All commits are welcome, even if they are minor ;)
- If you are unfamiliar with Github, you can email me contributions to the email below.

Contributions and Thanks

One final thanks: **RHCSA Cheat Sheet** would never have been realized in this form without the contributions of explanation:

- `https://github.com/karlseguin/the-little-mongodb-book`
- `http://brizzled.clapper.org/blog/2010/11/26/writing-markdown-and`

Contact the main author, Fabrizio Di Carlo at `dicarlo.fabrizio@gmail.com` or @fdicarlo

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[5] 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
```

[5] Gianluca Varisco http://it.linkedin.com/in/gvarisco

```
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.

Or you can simply use:

```
$ ls -l /usr/share/doc | grep ruby
drwxr-xr-x  2 root root  4096 ott 21 11:58 libmysql-ruby
```

```
drwxr-xr-x  2 root root  4096 ott 21 11:59 libruby
drwxr-xr-x 10 root root  4096 ott 24 19:21 libruby1.8
drwxr-xr-x  8 root root  4096 feb 23 18:59 libruby1.9.1
drwxr-xr-x  2 root root  4096 feb 23 18:59 libruby1.9.1-dbg
drwxr-xr-x  3 root root  4096 ott 24 19:22 libtcltk-ruby1.8
drwxr-xr-x  3 root root  4096 feb 23 18:59 libtcltk-ruby1.9.1
drwxr-xr-x  2 root root  4096 ott 21 11:58 ruby
drwxr-xr-x  2 root root  4096 ott 24 19:21 ruby1.8
drwxr-xr-x  2 root root  4096 feb 23 18:59 ruby1.9.1
drwxr-xr-x  2 root root  4096 feb 23 18:59 ruby1.9.1-dev
drwxr-xr-x  3 root root  4096 feb 23 18:59 ruby1.9.1-examples
drwxr-xr-x  2 root root  4096 feb 23 18:59 ruby1.9.1-full
drwxr-xr-x  2 root root  4096 feb 18 19:45 ruby-dev
drwxr-xr-x  3 root root  4096 ott 21 12:03 rubygems
drwxr-xr-x  2 root root  4096 ott 21 11:58 ruby-mysql
```

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 &
```

Questions

- Enter a command that lists all users who use bash as their default shells.
- Search a string root in /etc/passwd file and save in /somefile.

- What command compresses the /home directory into an archive in bzip2 format, in a file named homearch.tar.bz2?
- Enter a command that creates a /home/testuser/smb.conf file soft-linked to the /etc/samba/smb.conf file.

Operate running systems

Boot, reboot, and shut down a system normally.

Basically, they are referring to these actions on the command line. I'm sure everyone is able to do this on a pc, but not necessarily a live Red Hat Enterprise Linux Server remotely.

The commands are simple for the server.

Reboot

```
$ sudo reboot
```

Another way

```
$ sudo shutdown -r now
```

Changing to init 6 will reboot as well, which is what init 6 does.

```
$ sudo init 6
```

Shutdown

On the same note, init 0 calls all of the shutdown scripts and gracefully shuts down your machine.

```
$ sudo init 0
```

Surprisingly you can also use the shutdown command to shutdown completely by using the -h switch.

```
$ sudo shutdown -h
```

I think we all know how to boot the computer, so that shouldn't be a problem. :-)

Boot systems into different runlevels manually.

Red Hat Enterprise Linux is similar to most other linux distributions in its core functionality. The ability to run the operating system in multiple run levels is an important skill to have.

If you type into your terminal:

```
$ runlevel
```

you should see a number as the output.

```
$ N 3
```

This is the runlevel my server was running at the time this was written.

There are 6 runlevels:

- Runlevel 0 - Halt
- Runlevel 1 - Single User mode. Most services turned off, including networking. Used to perform maintenance on the server usually. Boots logged into roots account, no password.
- Runlevel 2 - This is basic functions, multi-user mode, without any networking.
- Runlevel 3 - This is what servers usually run in, as it provides all of the services of the normal server, without the graphical user interface.
- Runlevel 4 - Doesn't really get used.
- Runlevel 5 - This provides the same functions of runlevel 3, along with services to allow for desktop functionality (graphical user interface).

- Runlevel 6 - Reboot

The command to jump runlevels is actually really easy. Just type init followed by the runlevel you want to switch into.

```
$ init 1
```

The above command would turn off most services and drop you into single user mode.

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

Booting into single user mode is the easiest way to gain access to a Red Hat Enterprise Linux server.

This is only feasible if you have access to the physical console, which you will on the RHCSA and RHCE exams.

- At the beginning of the boot process you should see the grub menu pop up with a countdown and some kernel options (or perhaps just one option).
- It should be counting down at this point and says: "Press any key to enter the menu". In this case you would hit any key.
- At the bottom of the screen there is an explanation of the few options that are available to use on this page. One of these options is "e" for edit. Hit "e" to edit the boot kernel options. NOTE: (You can also use "a" for append, although they both accomplish the same thing.)
- You would now edit the main kernel options, adding either "single" or even just "1" at the end. Once you have completed that hit enter, the "b" for boot.

- You are now in single user mode, and be auto logged in as root.

NOTE: for Red Hat Enterprise Linux 6.0 there is a bug that will prevent you from changing your root password in single user mode. This is a result of SELinux. For this situation you would want to temporarily disable SELinux.

```
# setenforce 0
```

Now you should be allowed to change your root password.

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

A few commands to help you identify processes on the exam are ps and top. These are commands that you will actually use extensively to monitor systems in the workplace.

ps - report a snapshot of the current processes.

ps helps you see what processes are being run, what files and commands they are being run with, who they are being run by, as well and their process ids. All the above items are crucial when troubleshooting issues on a Red Hat Enterprise Linux 6 system.

a few good examples pulled from a man page:

EXAMPLES

```
# To see every process on the system using standard syntax:
$ ps -e
$ ps -ef
$ ps -eF
```

```
ps -ely

# To see every process on the system using BSD syntax:
$ ps ax
$ ps axu

# To print a process tree:
$ ps -ejH
$ ps axjf

# To get info about threads:
$ ps -eLf
$ ps axms

# To get security info:
$ ps -eo euser,ruser,suser,fuser,f,comm,label
$ ps axZ
$ ps -eM

# To see every process running as root (real & effective ID) in user
$ ps -U root -u root u
```

There is plenty of more info on this in the man pages as well as a plethoura of information on the web for ps.

top - display Linux tasks

At its most basic usage you can just type:

```
$ top
```

There is a whole lot of options that go along with that command: “man top” to see them all.

renice — alter priority of running processes

As stated in the description, renice is a linux utility to change the priority of a process. This could obviously come in handy while trying to keep a process at bay.

Example from the man page:

```
renice +1 987 -u daemon root -p 32
```

This would change the priority of process ID’s 987 and 32, and all processes owned by users daemon and root.

Man Page http://linux.die.net/man/8/renice

kill - terminate a process

Like it states in the name, this kills processes. Once you have identified the process you would like to kill with top or ps, you would use the kill command to terminate that process.

The most common implementation of this is:

```
# kill 2342
```

If that doesnt kill the process you would use the -9 switch, which will take out most any process.

```
# kill -9 2342
```

NOTE: The -9 command should be used with caution. Make sure you are killing the right pid, otherwise terrible things may transpire, especially on the RHCSA or the RHCE, where time is of concern.

Locate and interpret system log files.

Most of the logs you will deal with are going to be located in /var/log/. There are some exceptions to this, such as apache vhosts. Many people write the logs for a specific virtual host in a folder with the web content.

Aside from the occasional exception, this is the spot.

Logs are written in a way that makes them easy to parse through with text processing tools like cat, grep, and awk.

One example would be searching for Failed logins in /var/log/secure

```
$ cat /var/log/messages | grep Failed | less
Apr  1 16:17:06 mytest sshd[19632]: Failed password for root from 84.
Apr  1 16:17:09 mytest sshd[19634]: Failed password for root from 84.
Apr  1 16:17:13 mytest sshd[19636]: Failed password for root from 84.
Apr  1 22:13:40 mytest sshd[19741]: Failed password for bin from 200.
Apr  1 22:13:43 mytest sshd[19744]: Failed password for bin from 200.
Apr  1 22:13:46 mytest sshd[19747]: Failed password for bin from 200.
Apr  1 22:13:49 mytest sshd[19749]: Failed password for bin from 200.
Apr  1 22:13:52 mytest sshd[19751]: Failed password for bin from 200.
Apr  1 22:13:55 mytest sshd[19753]: Failed password for bin from 200.
Apr  2 06:05:01 mytest sshd[20102]: Failed password for root from 117
```

By using the cat command we are able to read all contents of the file, but thats a lot of stuff. We only want to see Failed logins. We then pipe the result of cat, into grep and process the text there. Grep picks out any line that contains Failed. I then piped it to less to output only the last 10 lines.So now you can see all these failed password attempts on our test server. Wow.

Lets say we just want to process the text, and get a count of how many logins were failed in this file. We can pipe the output into wc -l, which counts lines.

```
$ cat /var/log/messages | grep Failed | wc -l
90
```

Thats a lot of failed logins. That lets us know we should probably tighten up our security a bit, maybe change the port for ssh. Thats something that is covered down the road though. But you can see the value in combining text processing utilities in order to get a clean final result.

Some key tools to look at are:

- cat - http://man.he.net/man1/cat
- tail - http://man.he.net/man1/tail
- head - http://linux.die.net/man/1/head
- wc - http://linux.die.net/man/1/wc
- less - http://linux.die.net/man/1/less
- more - http://linux.die.net/man/1/more
- grep - http://linux.die.net/man/1/grep
- awk - http://linux.die.net/man/1/awk
- sed - http://linux.die.net/man/1/sed

Access a virtual machine's console.

Knowing how to access the virtual machines console is essential, and if you have never used it could take a few minutes to figure out.

There are two ways to pull it up, one from the gui menu in on the desktop, and the other with the following command in terminal:

```
$ virt-manager
```

For images of what it looks like, if you have never seen it, check out Red Hat's website on the Virtual Machine Manager. Its fairly straightforward.

http://virt-manager.et.redhat.com/index.html

Start and stop virtual machines.

Starting and stopping virtual machines is just like starting and stopping real machines.

The only main difference, is that you can start and stop them right from the . Virtual Machine Manager

The Red Hat Enterprise Linux 6 Virtual Machine Manager has a fairly intuitive graphical user interface, with obvious start and stop buttons for each machine. If you haven't used it at all, It would be good to download a trial copy of Red Hat, to get an idea of how to navigate the program.

List VMs on the system

```
# virsh list
```

Start a VM

```
# virsh create /etc/libvert/qemu/vm_file_name.xml
```

Kill a VM

Shutsdown without notification

```
#virsh destroy domain-id
```

Shutdown a VM

Shutdown down with notification

```
#virsh shutdown domain-id
```

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

There are a few things to consider when dealing with network services:

- You want to make sure the service is running, if not start it.
- You want to be able to restart the service, to reload a config file that you may have changed.
- You want to have the ability to turn the service off, if you don't plan on using it.
- Also, you need to be able to set the service up to start on boot, or vise versa.

Service management takes place with the service command. Go figure.

To start the httpd service, you would type:

```
$ sudo /sbin/service httpd start
```

To stop it:

```
$ sudo /sbin/service httpd stop
```

To restart it:

```
$ sudo /sbin/service httpd restart
```

To reload it (refresh configs without stopping and starting):

```
$ sudo /sbin/service httpd reload
```

How do you know what services you can do that with? Well that can be listed with the tool that will handing startup programs. chkconfig.

chkconfig is used to manage what runlevel a program with automatically start or get killed in. To list all your services you would just type

```
$ sudo /sbin/chkconfig --list
```

Thats a big list. But you get the idea, you can see how they are either on or off in each runlevel. To narrow down the list we can use grep to process the list and filter out say, our httpd service.

```
$ sudo /sbin/chkconfig --list | grep httpd
httpd              0:off   1:off   2:on    3:on    4:on    5:on    6:
```

So we can see that in runlevels 2-5 we have httpd on in. If that were not the case, and my server rebooted, when it came back up all my sites would be disabled, until I manually went in and started the service.

To change the values of that you would just run chkconfig followed by the service and whether you want to on or off in the main runlevels.

```
$ sudo /sbin/chkconfig httpd on
```

You can actually control what runlevels you want the service on in as well by adding the –level switch followed by the runlevels.

```
$ sudo /sbin/chkconfig --level 45 httpd off
$ sudo /sbin/chkconfig --list | grep httpd
httpd           0:off   1:off   2:on    3:on    4:off   5:off   6:off
```

I turned off the httpd service for runlevel 4 and 5 there.

Configure local storage

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

The official tool is now parted, but you can still use fdisk to create partitions. I'm a fan of fdisk, so thats what I will be using here.

List partitions

To list all partitions that are on your server, you would issue the fdisk command, with the list switch.

```
$ fdisk -l
```

Create new partitions

In order to create new partitions you would first have to open the device in fdisk. I will be opening /dev/sdb and creating both a primary and extended partition. We use the n command to create a new partition.

```
$ sudo fdisk /dev/sdb

Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4, default 1): 1
First sector (2048-8388607, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-8388607, default 8388607)
```

```
Command (m for help): p

Disk /dev/sdb: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd26a7e50

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      514047      256000   83  Linux

Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
e
Partition number (1-4, default 2):
Using default value 2
First sector (514048-8388607, default 514048):
Using default value 514048
Last sector, +sectors or +size{K,M,G} (514048-8388607, default 838860

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

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

Delete partitions

Deleting partitions is even easier. You would just type d at the fdisk prompt, tell it which partition number you are deleting, and the write the changes with the w flag.

```
$ sudo fdisk /dev/sdb

 Command (m for help): p

 Disk /dev/sdb: 4294 MB, 4294967296 bytes
 255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
 Disk identifier: 0x08bafe2e

   Device Boot      Start         End      Blocks   Id  System
 /dev/sdb1            2048     1026047      512000   83  Linux
 /dev/sdb2         1026048     1538047      256000    5  Extended

 Command (m for help): d
 Partition number (1-5): 2

 Command (m for help): p

 Disk /dev/sdb: 4294 MB, 4294967296 bytes
 255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
 Units = sectors of 1 * 512 = 512 bytes
 Sector size (logical/physical): 512 bytes / 512 bytes
 I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk identifier: 0x08bafe2e

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     1026047      512000   83  Linux

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

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

Set partition type

In order to use the disk we have to set a partition type. In this case we just make it a general Linux format #83. If it were going to be part of a raid array, we would make it Linux raid auto, #fd.

```
$ fdisk /dev/sdb

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): L

0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solari
1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/
2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/
3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/
4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS
6  FAT16           42  SFS             87  NTFS volume set db  CP/M /
```

```
7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell U
8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS ac
a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/
b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedS
c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS f
e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (F
10  OPUS           55  EZ-Drive        a7  NeXTSTEP        f0  Linux
11  Hidden FAT12   56  Golden Bow      a8  Darwin UFS      f1  Speed
12  Compaq diagnost 5c  Priam Edisk    a9  NetBSD          f4  Speed
14  Hidden FAT16 <3 61  SpeedStor      ab  Darwin boot     f2  DOS s
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMwar
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMwar
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANst
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix
Hex code (type L to list codes): 83

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

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

Redhat actually encourages the use of parted nowadays, with works a little differently. To see information about that, see this link http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storag disk-storage-parted.html

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

Create and remove physical volumes

Creating a physical volume in LVM is the first step in the LVM setup. Its the part where you actually tell Red Hat Enterprise Linux that you want a disk to be used for LVM.

pvcreate is the command used to add the physical volumes, or physical partitions.

```
~] pvcreate /dev/sdb
~] Physical volume "/dev/sdb" successfully created
```

pvremove is used to disassociate the volume from LVM.

```
~] pvremove /dev/sdb
~] Labels on physical volume "/dev/sdb" successfully wiped
```

Assign physical volumes to volume groups

Once that physical volume has been created we can add it to a volume group with the vgcreate or vgextend (if the volume group has already been created).

If the volume group does not exist, you can create it and add physical volumes in one shot:

```
~] vgextend MyVolGroup /dev/sdc
No physical volume label read from /dev/sdc
Physical volume "/dev/sdc" successfully created
Volume group "MyVolGroup" successfully extended
```

To assign a new physical volume to an existing volume group we use vgextend:

```
~] vgextend MyVolGroup /dev/sdc
Volume group "MyVolGroup" successfully extended
```

Similarly if we want to remove /dev/sdc from that group we would run ‘’vgreduce”:

```
~] vgreduce MyVolGroup /dev/sdc
Removed "/dev/sdc" from volume group "MyVolGroup"
```

Create and delete logical volumes

Logical Volumes have similar commands to create and delete as Volume Groups and Physical Volumes.

To create a new logical volume:

```
~] lvcreate -L 100M MyVolGroup
Logical volume "lvol0" created
```

To display the volume after for confirmation:

```
~] lvdisplay MyVolGroup
--- Logical volume ---
LV Name                /dev/MyVolGroup/lvol0
VG Name                MyVolGroup
LV UUID                zwLMev-i63w-7Jpk-XuqZ-VGl7-89Ov-WpoewP
LV Write Access        read/write
LV Status              available
# open                 0
LV Size                100.00 MiB
Current LE             25
```

```
Segments                1
Allocation              inherit
Read ahead sectors      auto
- currently set to      256
Block device            253:2
```

To delete the logical volume you would use the LV Name listed in the results of lvdisplay:

```
:::bash ~] lvremove /dev/MyVolGroup/lvol0 Do you really want to remov
```

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

This is a brand new objective that was not present on the RHEL5 requirements. There are a few steps to this, but once you go through it a few times its not too bad.

First step is to create a partition with fdisk or parted. We will use fdisk here.

```
~] fdisk -c -u /dev/sdb

Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4, default 1): 1
First sector (2048-8388607, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-8388607, default 8388607)
```

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 83

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

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

Now that the partition is created, we have to luks encrypt it. First we fill it with random data for security:

```
 ~] dd if=/dev/urandom of=/dev/sdb1 bs=1M
dd: writing '/dev/sdb1': No space left on device
201+0 records in
200+0 records out
209715200 bytes (210 MB) copied, 26.0497 s, 8.1 MB/s
```

Then we can encrypt the partition with luksFormat:

```
 ~] cryptsetup luksFormat /dev/sdb1

WARNING!
========
This will overwrite data on /dev/sdb1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
```

Now that the partition is encrypted, we open it and give it a label. The label is the name that it will show up as under /dev/mapper/

```
~] cryptsetup luksOpen /dev/sdb1 mynew_data
Enter passphrase for /dev/sdb1:
```

Once the partition is setup and luks encrypted, it will be available in the /dev/mapper/ directory. You can do an ls on the /dev/mapper/ directory to confirm.

```
~] ls /dev/mapper/
control  mynew_data  VolGroup-lv_root  VolGroup-lv_swap
```

Next steps involve creating a filesystem, adding the partition into the /etc/crypttab file, as well as in the /etc/fstab file in order to configure automounting on boot.

```
~] mkfs.ext4 /dev/mapper/mynew_data
mke2fs 1.41.14 (22-Dec-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
50800 inodes, 202752 blocks
10137 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67371008
25 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
```

```
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 25 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override
~] vim /etc/crypttab
```

In the /etc/crypttab file you would simply place the name of the encrypted device, as well as the path to the device:

mynew_data /dev/sdb1

Then we make the directory and add an entry into fstab, so that it mounts on boot:

```
~] mkdir /mynew_data
~] vim /etc/fstab

# Add the following:
/dev/mapper/mynew_data /mynew_data ext4    defaults    1 2
```

Thats it. You should run the mount command in order to verify your entries are correct in fstab, to prevent any boot issues.

```
~] mount -a

~] mount
# .... lots of data here that im leaving out
/dev/mapper/mynew_data on /mynew_data type ext4 (rw,relatime,seclabel
```

Awesome, try that a few times and you should be good to go on setting up luks encrypted partitions.

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

Configuring a filesystem to mount via UUID or label is an essential part of managing filesystems and partitions on the Red Hat Enterprise Linux system, and will most probably be something you will see on an RHCSA/RHCE exam.

First we will configure mounting at boot time via UUID. To find the UUID of a device you have to issue just one command:

```
~] blkid
/dev/sda1: UUID="183e5753-fbe7-4cf7-b974-f6cb9a326a33" TYPE="ext4"
/dev/sda2: UUID="1OJDNK-4gpP-s3YE-cK7o-1urJ-cXHk-jPnuF7" TYPE="LVM2_m
/dev/sdb1: UUID="1c1fa5a2-11e5-4d6b-89e9-61a15dcbe0f6" TYPE="crypto_L
/dev/mapper/VolGroup-lv_swap: UUID="dc82e68f-f1b9-409d-a1f7-c556eb6eb
/dev/sdc: UUID="vFLamh-rudP-T1jc-ZrrH-LTgD-FUuq-IHUBgM" TYPE="LVM2_me
/dev/mapper/VolGroup-lv_root: UUID="5bbc084b-1af0-464f-8629-9490a75ca
/dev/mapper/mynew_data: UUID="f8b694a6-916d-4ffa-8e5c-a7ed8ab25b5d" T
```

Once you have the UUID you can head over to /etc/fstab to create the entry. Here we will pick our new luks partition.

```
~] vim /etc/fstab
Inside of fstab we need to add a line. (if you already have a line fo

UUID=f8b694a6-916d-4ffa-8e5c-a7ed8ab25b5d /mynew_data ext4 defaults 1
```

Then write/save the file and quit :wq You can confirm that this is entered correctly by using the mount command:

```
~] mount -a
```

```
~] mount
..ommitted data...
/dev/mapper/mynew_data on /mynew_data type ext4 (rw,relatime,seclabel
```

Now to mount a filesystem via label requires another step, to label the filesystem. Luckily this is done in one easy step using e2label. I am going to label the filesystem luksdrive

```
~] e2label /dev/mapper/mynew_data luksdrive
```

Now we can unmount the filesystem, change our fstab to use a label, and run mount a again to see it mounted via label instead.

```
~] umount /mynew_data/
```

Verify its unmounted

```
~] mount
```

Then edit /etc/fstab this time using LABEL=luksdrive in place of UUID. So the line should look like:

```
LABEL=luksdrive         /mynew_data             ext4    defaults
Run mount -a and mount to confirm:

~] mount -a

~] mount
..ommitted data...
/dev/mapper/mynew_data on /mynew_data type ext4 (rw,relatime,seclabel
```

Thats all there is to that. I would try that out a number of times to make sure you have the process down. Repitition is key.

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.

Configure networking

Networking is a big part of the RHCSA and RHCE. If you aren't super comfortable with configuring networking via network config files, then its probably a good idea to use the network management tools available.

In Red Hat Enterprise Linux you can type "setup" at the command line. This will open up the Text Mode Setup Utility, which allows you to configure network, firewall, athentication, keyboard, RHN, and System Services.

If you do venture into the networking configuration files, these are the important ones:

- /etc/hosts The static table lookup for hostnames
- /etc/resolv.conf The resolver configuration file
- /etc/sysconfig/network Contains hostname setting
- /etc/sysconfig/network-scripts/ifcfg-eth0 The first network device configuration

These are files that will be essential to know about during the exam if you are altering config files.

There will no doubt be some need to configure networks during the RHCSA and RHCE, so either way be prepared to fix network connections.

Configuring the hostname

Configuring the hostname can be done in the /etc/sysconfig/network file. Edit this file with the updated hostname and then on reboot, the new hostname will be reflected.

```
~] vim /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=rhel-01
```

Schedule tasks using cron.

Cron is a utility used to schedule tasks to run at a certain time on various intervals. First is to make sure its installed, although it is installed by default on a normal installation.

```
~] rpm -qa | grep cron
cronie-1.4.4-2.el6.x86_64
cronie-anacron-1.4.4-2.el6.x86_64
```

The easiest way to get guidance on how to use a utility is to use the man page. In this case the proper documentation is kind of hidden.

```
~] man 5 crontab
```

This page lays out the options for cron, why its not found by simply using "man cron" is beyond me, but its not.

The format for this goes as follows:

```
*    *    *    *    *  command to be executed
-    -    -    -    -
|    |    |    |    |
```

```
|    |    |    |    |
|    |    |    |    +----- day of week (0 - 6) (Sunday=0)
|    |    |    +---------- month (1 - 12)
|    |    +--------------- day of month (1 - 31)
|    +-------------------- hour (0 - 23)
+------------------------- min (0 - 59)
```

An example of a cron job would be configuring a job to run every day on minute 0 hour 12 daily, or daily at 12:00pm.

```
0 12 * * * /bin/echo "some job" >> echo.log
```

Another example would be to run a job weekly at 3:30pm on Sunday

```
30 15 * * 0 /bin/echo "another job" >> echo.log
```

Configure systems to boot into a specific runlevel automatically.

Depending on what the system running Red Hat Enterprise Linux 6 is going to be used for, you will want it to boot into the appropriate runlevel.

The file that controls the runlevel that a system boots into is the /etc/inittab.

```
# Default runlevel. The runlevels used are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have net
#   3 - Full multiuser mode
#   4 - unused
```

```
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
```

In this case the default runlevel that this system will boot into is runlevel 5. Make note of what not to do. As noted above, do not set the default runlevel to 0 or 6, which is shutdown and reboot, for obvious reasons.

Install Red Hat Enterprise Linux automatically using Kickstart.

Installing a system via Kickstart comes in pretty useful in real life. Whether there is time to do that in the 2.5 hours that they give you on the exam is questionable, but regardless its an objective.

There are a few ways to create a kickstart file, that would be used in the automatic installation of a Redhat Enterprise Linux 6 system. Theres always writing the thing from scratch, which while always an optoin, is not so efficient. Besides that there is:

```
system-config-kickstart (requires installing this application)
```

using the anaconda-ks.cfg that was created during an installation.

On the exam you would probably (hopefully) be provided with a premade kickstart file, so we work from there.

Lets say we have this info:

```
kickstart file = http://192.168.111.23/pub/ks/redhat6.kfg
ip of new install = 192.168.111.222 (same subnet)
netmask = 255.255.255.0
```

First we would boot the system with some sort of boot media, most likely the RHEL 6 CD-ROM #1 and at the boot prompt (when it asks you what you want to do) you would type a command like this, substituting your own info:

```
linux ks=http://192.168.111.23/pub/ks/redhat6.kfg append ip=192.168.1
```

As long as everying is configured correctly and the installation media is where is should be, then this should install pretty hands off. Of course, anything besides this already configured environment would just take way too much time to be included on the exam. As long as you know how to install via ks file, you are probably good.

Configure a physical machine to host virtual guests.

A default RHEL 6 system should come prepared to host virtual guests, minus the packages. In RHEL5 you had to make sure that you were running the xen kernel, which would require installing and booting into that kernel. RHEL 6 is simple, if it Virtualization is not installed, install it.

```
~] yum groupinstall "Virtualization"
```

That will install everything needed to run virtual guests on RHEL 6.

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.

Installing apache via yum on Red Hat Enterprise Linux 6 does most of the setup for you.

```
~]  yum install httpd

:::bash
~]  service httpd start
```

Now if you try to visit the main ip or domain of the server, you may run into an issue getting to the site. Whenever you enable a network service like a web server, you also have to allow the outside to use that service. We have to add an entry into iptables.

```
~]  iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

This would add an entry into iptables, but to survive a reboot we would have to save this.

```
~]  service iptables save
```

Now an easier way of doing this, is to use system-config-firewall, which is the gui/tui tool to configure a firewall.

```
~]  system-config-firewall
```

This may not make things perfect, but it can definitely give you a jump start to molding rules on an exam.

Configure a system to run a default configuration FTP server.

FTP

vsftpd installs with a default configuration that works for this requirement. So a basic:

```
~] yum install vsftpd
```

```
~] chkconfig vsftpd on
```

This will get your default server up and running. But what about firewall and selinux?

Iptables

For iptables you want to open up port 20 and 21, to allow ftp requests in.

```
~] iptables -I INPUT 5 -p tcp -m tcp --dport 20 -j ACCEPT
```

```
~] iptables -I INPUT 5 -p tcp -m tcp --dport 21 -j ACCEPT
```

Then remember to always save your iptables rules so they survive a reboot.

Selinux

SELinux is now a part of the exams, so you have to know how to apply the correct context to the directories that will be used by vsftpd.

Here's a tip: All this information is stored in man pages, so rather than memorizing, use the resources available. If you search for _selinux, then all services that have information on how to be configured with SELinux will show up. To search the man pages use:

```
~] man -k _selinux
ftpd_selinux         (8)  - Security-Enhanced Linux policy for ftp da

~] man ftpd_selinux
```

To make a ftp server's content available you can see it says to run the following:

```
semanage fcontext -a -t public_content_t "/var/ftp(/.*)?"

restorecon -F -R -v /var/ftp
```

Thats it. As long as you can install the application, vsftpd, open the correct ports in iptables, and set context in SELinux, then you are good to go on this objective.

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

Red Hat Network is pretty easy to work with.

```
~] rhn_register
```

And follow the instructions. Now, if you arent lucky enough to have a subscription to this wonderful service, then you will most likely using a repo that you create, or that is given to you.

Most common situation is having a remote repository that you need to pull packages from. Usually you be given a url to connect to looking something like this: http://myremote.com/repo/i386/. The yum repo files are located in /etc/yum.repos.d/ and end with a .repo extension. The format is simple to setup a repo on the fly.

```
[myremote]
name=myremote
baseurl=http://myremote.com/repo/i386/
enabled=1
gpgcheck=0
```

Those are the essential elements to pull packages via yum from that repo.

Setting up a local repo with a disk is almost the same with a few steps before. First the disk needs to be mounted, and the packages copied from Packages/ into another directory on the server. In this case we will use file:///directory/path/to/repo/ as the url, where /directory/path/to/repo/ is the directory that contains the rpm files.

Next the package creatrepo needs to be installed. Once installed cd into the directory and run:

```
~] createrepo .
```

Now that you have a repo setup, yum needs to know about it. Create a file named mylocal.repo in the /etc/yum.repos.d/ directory.

```
[mylocal]
name=mylocal
baseurl=file:///directory/path/to/repo/
enabled=1
gpgcheck=0
```

Run a yum command to test, and it should be pulling information about packages from the local repo.

```
~] yum list httpd
```

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