RHCE-UPDATER

Red Hat Certified Engineer Updater: Red Hat Enterprise Linux Version 4 to Version 5

RHCE-UPDATER-RHEL5-en-2-20070323

Table of Contents

RHCE-UPDATER - Red Hat Certified Engineer Updater: Red Hat Enterprise Linux Version 4 to Version 5

Unit 1 - Version 5 RHCE Updater

Unit 1

Version 5 RHCE Updater

1-1

Objectives

Upon completion of this unit, you should be able to:

- List and use the new RHCT and RHCE competencies in version 5
- List and use new tools in version 5
- The `/etc/pki` directory
- Use other useful tools

1-2

Copyright

- The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2007 Red Hat, Inc.
- No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.
- This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.
- If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please email training@redhat.com or phone toll-free (USA) +1 866 626 2994 or +1 919 754 3700.

1-3

Overview

- This document assists candidates for the RHCE certification who have studied under version 4 in studying for the version 5 exam

This document assists candidates for the Red Hat Certified Engineer (RHCE) certification who have studied under Red Hat Enterprise Linux, version 4, and wish to take one of these certification exams under Red Hat Enterprise Linux, version 5. This document does not present a complete list of differences between the releases; rather, it assists RHCE candidates in their preparation for the certification exam.

This document is divided into two sections: these first few pages give an overview of RHCE-related changes in version 5. Pages following this give specific details on those changes.

This document does not present a comprehensive list of material new in version 5 of Red Hat Enterprise Linux. Nor does it contain a discussion of all items new in the RH131, RH133, RH253, or RH300 courses. Rather, it is aimed specifically at presenting information relevant to continuing or new competencies in the RHCE exam.

New Competencies for the Version 5 RHCT/RHCE certifications

- Access Control Lists
- SELinux
- NTP

New competencies have been defined for the RHCT and RHCE certifications. Although none of these skills are new in version 5, the expectation that an RHCE can manipulate these subsystems is new.

- Access Control Lists. Although ACLs were included in version 4 of the operating system and in the version 4 RH131 and RH133 courses, the ability to set, list, and understand ACLs is now considered a competency for the version 5 RHCT and RHCE certifications.

- SELinux. Although the SELinux subsystem was included in version 4 of the RH131, RH133, and RH300 courses, this is now a new competency in the version 5 RHCT and RHCE exams. SELinux is a complex topic and a complete understanding of the nature and configuration of SELinux is not expected. However, an RHCT or RHCE is expected to understand what SELinux is and what it is trying to accomplish; how to turn SELinux on and off and how to set it to enforce and not enforce rules; how to list SELinux security contexts for files and processes; how to set an appropriate security context for a file.

- Network Time Protocol (NTP). The ability to configure NTP has been part of Red Hat Enterprise Linux for some time; it is now an RHCT and RHCE competency.

See the attached pages for information on these new competencies.

New in Version 5

- Red Hat Enterprise Linux variants: Server and Client
- Kernel versions
- **yum**
- **resize2fs** replaces **ext2online**

New features in Red Hat Enterprise Linux 5 include the following:

- Red Hat Enterprise Linux variants. The variants of the operating system have changed. In version 4, the variants were AS, ES, WS, and Desktop. Under version 5, the variants are Server and Client.
- Kernel versions. Kernel versions available under x86 hardware have changed.
- **yum**. The **yum** software installation command is now available.
- **resize2fs**. The **resize2fs** command is now the proper tool to use to resize an ext3 filesystem in place of the obsolete **ext2online** command.

See the attached pages for information on these new features.

The /etc/pki Directory

- SSL-related files now located in /etc/pki
- Items located in /etc/pki include:
  - pki/rpm-gpg/*: Red Hat's RPM GPG keys
  - dovecot/certs/dovecot.pem: **dovecot**'s digital certificate
  - tls/certs/localhost.crt and tls/private/localhost.key: Apache's SSL keys

Starting with the version 5, the location of SSL-related keys is now in /etc/pki. Items in this directory include:

- pki/rpm-gpg/*: Red Hat's RPM GPG keys: the GPG keys used by the **rpm** command to validate RPM packages are now located in this directory.
- dovecot/certs/dovecot.pem: **dovecot**'s digital certificate: in order for **dovecot** to run encrypted **pop** or **imap** services (**pop3s** and **imaps**), **dovecot** needs a digital certificate, now located here.
- tls/certs/localhost.crt and tls/private/localhost.key: Apache uses two digital certificates when running in encrypted mode: a public key called localhost.crt, and a private key, called localhost.key.

Additional Useful Tools

- Initial RAM disk
- **rsync**
- **ip**
- **ethtool**

Starting with the version 5 version of the RH131 and RH133 courseware, the following useful tools are now emphasized more than in the past.

- Initial RAM disk. Although the initial RAM disk has been available for some time, the version 5 version of the RH131 and RH133 courses have additional information on this topic, including instructions on how to build an initial RAM disk with an additional module.
- The **rsync** command. Although **rsync** has been available in Red Hat Enterprise Linux for some time, the version 5 version of the RH131 and RH133 courses now include information on this topic as a tool for backing up data.
- The **ip** command. This command is now considered a standard mechanism for manipulating network configuration.
- The **ethtool** command. In the past, both **mii-tool** and **ethtool** were discussed as tools for displaying information about and configuring network interface cards. In the version 5 version of RH131 and RH133, we now consider **ethtool** to be the preferred tool; we no longer discuss **mii-tool** in this class.

See the attached pages for information on these tools.

New Competencies for the Version 5 RHCT/RHCE Certifications

1-9

Access Control Lists (ACLs)

- Grant rwx access to files and directories for multiple users or groups
  - **mount -o acl */directory***
  - **getfacl *file*|*directory***
  - **setfacl -m u:gandolf:rwx *file*|*directory***
  - **setfacl -m g:nazgul:rw *file*|*directory***
  - **setfacl -m d:u:frodo:rw *directory***
  - **setfacl -x u:samwise *file*|*directory***

1-10

The ext3 filesystem includes support for access control lists which allow finer grained control of file system permissions than are possible with the standard three access categories that are normally provided. Many filesystem commands, such as **cp** and **mv**, have been modified to copy the associated ACL's for a file.

In order to enable ACLs on a file system, the file system must be mounted with the **acl mount** option. Filesystems created during installation include the **acl** flag in their default mount option. To remount the `/home/` directory with the **acl** option, run the following:

```
[root@stationX]# mount -o remount,acl /home/
```

To view the ACL's for a file, use the **getfacl** command:

```
[root@stationX]# getfacl /home/schedule.txt
getfacl: Removing leading '/' from absolute path names
# file: home/schedule.txt
# owner: root
# group: root
user::rw-
user:bob:rwx
group::r--
group:admins:rw
mask::rwx
other::r--
```

ACL's can be modified using the **setfacl** command:

```
[root@stationX]# setfacl -m u:visitor:rx /home/schedule.txt
```

The above command would grant the user visitor read and execute access to the file `/home/schedule.txt`.

To remove (expunge) an ACL:

```
[root@stationX]# setfacl -x u:visitor /home/schedule.txt
```

On directories, default access control lists can be used. If we wanted all newly created content of a directory to be readable and writable by the user student:

```
[root@stationX]# setfacl -m d:u:student:rw /home/share/project/
```

SELinux

- Mandatory Access Control (MAC) -vs- Discretionary Access Control (DAC)
- A rule set called the *policy* determines how strict the control
- Processes are either restricted or unconfined
- The policy defines what resources restricted processes are allowed to access
- Any action that is not explicitly allowed is, by default, denied

1-11

In an effort to deal with ever increasing threats to their data systems, the US government assigned the National Security Agency (NSA) with the task of developing a single set of rules that all other agencies would follow in handling confidential information. By evaluating previous breaches, the NSA determined that a major hurdle to securing data was internal users bypassing local security. In some cases, users would inadvertently open access to a system. A classic example would be a user executing the command **chmod 777 ~**. To the user, this may seem an easy way to allow co-workers to share files. They may not realize that this gives everyone in the world access to potentially confidential information.

In more extreme cases, users would intentionally disable security for more insidious reasons. Both of these situations are examples of a user having the discretion to control the access to their system. The NSA felt the solution was to have systems implement *Mandatory Access Control* (MAC) over the users. In MAC, a set of rules, known as the *policy*, identify what a process is allowed to do. Anything that is not explicitly permitted is, by default, denied. Ideally, different policies could be implemented depending upon how strict the security needs.

The NSA's first implementation of MAC was a system called Mach, which introduced the concept of *Type Enforcement* (TE). Objects (files, directories, resources) were assigned a type value. Users and processes were also assigned a type value. The policy contains rules that allow a user or process to access objects. It was possible for a policy to be so *strict*, it was difficult to manage, so occasionally users and processes were allowed to be *unconfined*. This meant the policy did not apply to that user or process.

The NSA decided Mach made a better security framework than operating system. To get this framework deployed on production systems, the NSA needed access to the OS source code. They took advantage of the open source Linux kernel, and developed a set of patches to enable MAC. These patches became known as Security Enhanced Linux, or SELinux for short. They were later refined and incorporated into the kernel source by Red Hat.

SELinux, continued

- All files and processes have a *security context*
- The context has several elements, depending on the security needs
  - user:role:type:sensitivity:category
  - `user_u:object_r:tmp_t:s0:c0`
  - Not all systems will display s0:c0
- **ls -Z**
- **ps -Z**
  - Usually paired with other options, such as **-e**

There is an old UNIX saying that "everything is a file". Traditional file access is controlled by user, group, and permission settings. To SELinux, everything is an object and access is controlled by security elements stored in the the inode's extended attribute fields. Collectively, the elements are called the security context. Presently, there are five supported elements, although all five may not be present on all systems:

user	Indicates the type of user that is logged into the system. If a user logs in as root, they will have the user value of `root`. Other users will have a value of `user_u`. If they escalate their privileges with **su**, they will still have the user value of `user_u`. Processes have a value of `system_u`.
role	Defines the purpose of the particular file, process, or user. Files have the role of `object_r`. Processes get the role of `system_r`. Users also have the role of `system_r`, because (to Linux) users are similar to processes.
type	Used by Type Enforcement to specify the nature of the data in a file or processes. Rules within the policy say what process types can access which file types.
sensitivity	A security classification sometimes used by government agencies.
category	Similar to group, but can block root's access to confidential data.

To view the security context of a file, use the **ls** command's **-Z** option:

```
[root@shadex4 ~]# ls -Z /root/anaconda-ks.cfg  /var/log/messages
-rw-r--r--  root root   user_u:object_r:user_home_t   anaconda-ks.cfg
-rw-------  root root   system_u:object_r:var_log_t   /var/log/messages
```

Generally speaking, files inherit a directory's security context:

```
[root@stationX ~]# ls -Zd /etc /etc/hosts
drwxr-xr-x  root root   system_u:object_r:etc_t   /etc
-rw-r--r--  root root   system_u:object_r:etc_t   /etc/hosts
```

Some files, however, get a unique security context, for added security:

```
[root@stationX ~]# ls -Z /etc/shadow /etc/aliases
-r--------  root root   system_u:object_r:shadow_t        /etc/shadow
```

```
rw-r--r--  root root   system_u:object_r:etc_aliases_t   /etc/aliases
```

If a system were running under the most secure configuration, everything would be restricted by SELinux. This is not practical in most cases. For this reason, SELinux is being deployed in phases. In Red Hat Enterprise Linux 4, SELinux was protecting 13 processes. In the initial release of Red Hat Enterprise Linux 5, this count had increased to 88.

To determine if a process is protected, use to **ps** command's **-Z** option:

```
[root@stationX ~]# ps -ZC syslogd,bash
LABEL                          PID   TTY    TIME      CMD
system_u:system_r:syslogd_t    1888  ?      00:00:00  syslogd
user_u:system_r:unconfined_t   2583  pts/0  00:00:00  bash
```

Any process whose type is *unconfined_t*, is not yet restricted by SELinux. To view the entire process stack, use either **ps -eZ** or **ps Zax**.

*Note:*A restricted process is sometimes called *protected*, though it is the data that is protected, not the process.

SELinux: Targeted Policy

- The targeted policy is loaded at install time
- Most local processes are *unconfined*
- Principally uses the type element for *type enforcement*
- The security context can be changed with **chcon**
  - **chcon -t tmp_t /etc/hosts**
- Safer to use **restorecon**
  - **restorecon /etc/hosts**

The source code that allows SELinux to protect a process is integrated into the kernel, but the rules that define how SELinux enforces security is defined in the policy. The NSA's most secure policy is called the *strict* policy. By default, Red Hat uses the *targeted* policy. The targeted policy "targets" certain processes to be restricted, and then enforces their access to files and resources. Other policies also exist, such as the *Multi Level Security* (MLS). All policies could be thought of as a subset of the Strict policy.

The policy defines the elements that can be used, and whether users are allowed to manipulate the elements. Since the targeted policy uses Type Enforcement as its principal security mechanism, it pays the most attention to the type element of the security context. The policy allows the **chcon** command to change the security context.

```
[root@stationX ~]# ls -Z install.log
-rw-r--r--  root root   root:object_r:user_home_t   install.log
[root@stationX ~]# chcon -t etc_t install.log
[root@stationX ~]# ls -Z install.log
-rw-r--r--  root root   root:object_r:etc_t         install.log
```

The **chcon** command can only use types that are defined in the policy. Rather than memorizing all the possible types that can be used, **chcon --reference** can take the security context from one object, and apply it to another.

```
[root@stationX ~]# ls -Z anaconda-ks.cfg
-rw-------  root root   system_u:object_r:user_home_t   anaconda-ks.cfg
[root@stationX ~]# chcon --reference /etc/shadow anaconda-ks.cfg
[root@stationX ~]# ls -Z anaconda-ks.cfg
-rw-------  root root   system_u:object_r:shadow_t      anaconda-ks.cfg
```

A safer alternative is the **restorecon** command. With **restorecon**, the policy determines and applies the object's default context.

```
[root@stationX ~]# restorecon /root/*
[root@stationX ~]# ls -Z /root
-rw-------  root root   root:object_r:user_home_t    anaconda-ks.cfg
-rw-r--r--  root root   root:object_r:user_home_t    install.log
```

SELinux: Management

- Modes: Enforcing, Permissive, Disabled
  - Changing enforcement is allowed in the Targeted policy
  - **getenforce**
  - **setenforce 0 | 1**
  - Disable from GRUB with **selinux=0**
- `/etc/sysconfig/selinux`
- **system-config-securitylevel**
  - Change mode, Disabling requires reboot
- **system-config-selinux**
  - Booleans
- **setroubleshootd**
  - Advises on how to avoid errors, not ensure security!

The policy for a system is defined in the `/etc/sysconfig/selinux` file as is the mode of operation. The policy can be Disabled, Enforcing, or Permissive. Disabled means the policy is ignored. Permissive is a mode for troubleshooting or development that logs policy violations, but does not prevent programs from running. Enforcing is the default mode.

The **getenforce** command can be used determine the system's current mode. The targeted policy allows the use of the **setenforce** command to toggle between the Enforcing (1) and Permissive (0) mode:

```
[root@stationX ~]# getenforce
Enforcing
[root@stationX ~]# setenforce 0
[root@stationX ~]# getenforce
Permissive
```

The only way to disable SELinux is to change `/etc/sysconfig/selinux`, and reboot, or use the **selinux=0** Grub kernel option.

The GUI tool, **system-config-selinux**, allows for changes of a few other SELinux options. The targeted policy allows certain features to be controlled through a set of *booleans*. These are predefined functions that can selectively turn off enforcement of certain daemons. It would be preferable to ensure objects have the correct security context rather than shutting off security features.

When an application attempts something that is not authorized by the policy, SELinux blocks access and an error is logged to `/var/log/messages`. The application is often unaware of why it failed. This can make troubleshooting difficult. To help in the troubleshooting process, the **setroubleshootd** daemon will alert you of the error by placing a warning icon on the alert panel. Clicking on the icon will display a possible fix for the error. It is important to realize that the proposed solution may not be the best solution for the problem.

Network Time Protocol

- Workstation hardware clocks tend to drift over time without correction
- Many application require accurate timing
- Time synchronization makes system logs easier to analyze
- NTP counters the drift by manipulating the length of a second
- NTP clients should use three time servers
- Config file: `/etc/ntp.conf`
- Config tool: **system-config-date**

The PC hardware clock is not accurate enough for many applications. It tends to drift over time. Many applications require exact timing and may react poorly if the clock is reset suddenly to a new time.

NTP counters the drift by modifying the length of a second, much like tuning the pendulum of a old fashioned clock. If the system's time is behind the average of the time servers the second is made shorter so that the system clock races towards the correct time. Thus the time difference is reduced gently without disturbing other applications. However if the time differs to greatly, NTP ceases to work. In this case the clock must be reset manually with **ntpdate**.

Having three central time servers allows clients to reject bogus synchronization messages if one of the servers' NTP daemons or clocks malfunctions. It is possible for a client to synchronize with fewer time servers if necessary but it is less secure.

NTP is configured in `/etc/ntp.conf`. For simple client configurations however it is often enough to use the graphical configuration tool **system-config-date** which is automatically run during firstboot .

```
server 192.168.0.1
server 192.168.0.2
server clock.example.com
driftfile /var/lib/ntp/drift
```

New in Version 5

Red Hat Enterprise Linux

- Enterprise-targeted operating system
- Focused on mature open source technology
- 18-24 month release cycle
  - Certified with leading OEM and ISV products
- Purchased with one year Red Hat Network subscription and support contract
  - Support available for seven years after release
  - Up to 24x7 coverage plans available

The Red Hat Enterprise Linux product family is designed specifically for organizations planning to use Linux in production settings. All products in the Red Hat Enterprise Linux family are built on the same software foundation, and maintain the highest level of ABI/API compatibility across releases and errata. Extensive support services are available: a one year support contract and Update Module entitlement to Red Hat Network are included with purchase. Various Service Level Agreements are available that may provide up to 24x7 coverage with guaranteed one hour response time. Support will be available for up to seven years after a particular release.

Red Hat Enterprise Linux is released on an eighteen to twenty-four month cycle. It is based on code developed by the open source community and adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors such as Dell, IBM, Fujitsu, BEA, and Oracle. Red Hat Enterprise Linux provides a high degree of standardization through its support for five processor architectures five processor architectures (Intel x86-compatible, AMD AMD64/Intel 64, Intel Itanium 2, IBM POWER, and IBM mainframe on System z).

Kernel Images and Variants

- Architectures supported: x86, x86_64, IA64/Itanium, PowerPC64, s390x.
- Three kernel versions available for x86:
  - Regular: one or more processors but 4GB of RAM or less
  - PAE: multiple processors and up to 64G of RAM
  - Xen: needed for virtualization
- Kernels always installed under `/boot/vmlinuz-*`

The general idea behind providing a variety of kernel versions is that the more features are built in, the more overhead it will bring. For 32bit x86, there is a great deal of difference between the kernels.

The standard kernel is usually installed by default. It support multiple processors, a feature known as Symmetric MultiProcessing (SMP). Memory support is limited to 4 GB physical memory though the amount of *virtual* memory could be larger through the use of swap space. The memory limit per process is 3 GB.

The hugemem kernel adds total memory support of up to 64 GB of RAM and uses the 4/4 memory split. This means that processes and the kernel will both be able to use up to 4 GB of physical memory. This comes at a cost, however: all switches from user to kernel space will require the address space to be remapped and this can hurt performance significantly.

The Xen kernel, (i.e. the kernel-xen RPM package) contains the Xen-enabled kernel for both the host and guest operating systems as well as the hypervisor. Xen is a virtual machine that can securely run multiple operating systems in their own sandboxed domains.

About yum

- Front-end to **rpm**
  - Designed to resolve package dependencies
  - Can locate packages across multiple repositories
- Replacement for **up2date**

Development of RPM cemented the future of Linux by greatly simplifying installation of software. As the operating system became more complex, RPM began to show a few weaknesses, primarily its inability to resolve dependencies.

```
[root@stationX ~]# rpm -ivh x3270-x11-*
warning: x3270-x11-3.3.4p7-3.el5.1.x86_64.rpm: Header V3 DSA signature: NOKEY,
key ID 897da07a
error: Failed dependencies:
       x3270 = 3.3.4p7 is needed by x3270-x11-3.3.4p7-3.el5.1.x86_64
[root@stationX Server]#
```

Unfortunately, there is no way to look at the above output and determine if the suggested RPM, x3270, also has a dependency. As a matter of fact it is possible that dozens of RPMs would have to be installed. If the need RPMs were not available in the current directory, it would be up to the user to located and install each.

To solve the problem of dependency resolution and package location, volunteer programmers at Duke University developed Yellow dog Update, Modified, or YUM for short. The system is based on repositories that hold RPMs and a repodata filelist. The yum application can call upon several repositories for dependency resolution, fetch the RPMs, and install the needed packages.

```
[root@stationX ~]# yum install x3270-x11
... output truncated ...
Dependencies Resolved

================================================================
Package                 Arch       Version          Repository        Size
================================================================
Installing:
x3270-x11               x86_64     3.3.4p7-3.el5.1  server1           424 k
Installing for dependencies:
x3270                   x86_64     3.3.4p7-3.el5.1  server1           142 k

Transaction Summary
================================================================
Install      2 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 566 k
Is this ok [y/N]:
```

Using yum

- Install/Remove/Update
  - **yum install** *package...*
  - **yum remove** *package...*
  - **yum update** *[package...]*

1-20

The command-line utility **yum** gives you an easy way to manage the packages on your system:

```
[root@stationX ~]# yum install firefox
```

The above command will search the configured repositories for a package named firefox, and if found will install the latest version, pulling in dependencies if needed.

```
[root@stationX ~]# yum remove mypackage
```

The above command will try to remove the package named mypackage from your system. If any other package depends on mypackage **yum** will prompt you about this, giving you the option to remove those packages as well.

```
[root@stationX ~]# yum update [mypackage...]
```

If any packages are specified on the command-line **yum** will search the configured repositories for updated versions of those packages and install them. When no packages are specified **yum** will search for updates to all of your currently installed packages.

Searching packages/files

- Searching packages
  - **yum search** *searchterm*
  - **yum list** *(all|available|extras|installed|recent|updates)*
  - **yum info** *packagename*
- Searching files
  - **yum whatprovides** *filename*

yum search *searchterm* will search all known package names and descriptions for *searchterm*:

```
[root@stationX ~]# yum search cairo
```

yum list *searchterm* will search all known package names for *searchterm*. *searchterm* can include wildcards:

```
[root@stationX ~]# yum list '*irefo*'
```

yum info *package...* will search all the package database for *package* and display some info about the package.

```
[root@stationX ~]# yum info '*irefo*'
```

yum whatprovides *filename* will search all packages (both installed and available) for *filename*. This can be useful when you know the filename of an executable/library you need, but you don't know the package name.

```
[root@stationX ~]# yum whatprovides /usr/sbin/sendmail
```

Red Hat Network Client

- Registration
  - Run **rhn_register**
  - Select the updates location (RHN or local satellite/proxy)
  - Enter Account information
- Interactive usage
  - **yum** plugin for downloading packages from RHN
  - Configuration in `/etc/yum/pluginconf.d/rhn-plugin.conf`
- Remote management
  - **rhnsd** polls RHN every four hours
  - **rhn_check** polls imediately

rhn_register will create configuration files for **yum** as well as registering your system with RHN or your satellite-server.

RHN can be used to perform remote administration of collections of machines. First, actions for the machine (such as specific package installation or upgrades) are queued for the machine using the RHN account.

Client machines use the rhnsd daemon to poll RHN periodically for queued actions. By default, rhnsd polls every 4 hours, though this can be adjusted in `/etc/sysconfig/rhn/rhnsd`. The rhnsd daemon uses the **/usr/sbin/rhn_check** command to actually perform the poll and administer any queued actions. Notably, the rhnsd daemon does not open any server networking ports.

Configuring Additional Repositories

- Create a file in `/etc/yum.repos.d` for your repository
- Required information
  - [*repo-name*]
  - name=*A nice description*
  - baseurl=*http://yourserver.com/path/to/repo*
  - enabled=*1*
  - gpgcheck=*1*

Inside a repository declaration you can use variables like `$releasever` and `$basearch` to be substituted with the relevant information for your installation. Using this you can roll out one repository-file for use on multiple architectures or releases.

Creating a private repository

- Create a directory to hold your packages
- Make this directory available by http/ftp
- Install the **createrepo** RPM
- Run **createrepo -v `/package/directory`**
- This will create a `repodata` subdirectory and the needed support files

The **createrepo** creates the support files necessary for a yum repository. These files will put into the `repodata` subdirectory.

`repomd.xml` - The `repomd.xml` contains timestamps and checksum values for the other three files. Once a client has established a connection with a server, it caches all files, and only refreshes the cache if `repomd.xml` indicates the repo has changed.

`primary.xml.gz` - The `primary.xml.gz` file contains the list of all the RPMs in the repository, as well as dependency information. It also contains the information that would normally be returned by **rpm -qlp**.

`filelists.xml.gz` - The `filelists.xml.gz` file contains a list of all the files in all the RPMS. This is used by queries such as **yum whatprovides**.

`other.xml.gz` - The `other.xml.gz` file contains additional information, including the changelogs for the RPMs.

`comps.xml` - The optional `comps.xml` file contains information about package groups. This allows group installations and optimizes dependency resolution.

The addition or deletion of files within the repository requires a **createrepo** to be run again.

Resizing Logical Volumes

- Growing Volumes
  - **lvextend** can grow logical volumes
  - **resize2fs** can grow EXT3 filesystems online
  - **vgextend** adds new physical volumes to an existing volume group.
- Shrinking volumes
  - Filesystem must be reduced first
  - Requires a filesystem check and cannot be performed online
  - **lvreduce** can then reduce the volume.
  - Volume Groups can be reduced with:

```
pvmove /dev/hda3
vgreduce vg0 /dev/hda3
```

Logical volumes can be resized dynamically while preserving the data on the volume, if the volume's filesystem supports resizing. The **lvextend** command allows resizing of an e or ext3-based logical volume. **resize2fs** can be used to grow mounted ext2 and ext3 filesystems. **lvextend** must be called first to grow the logical volume.

The following commands grow the mounted `/dev/vg0/data` filesystem:

```
# lvextend -L +500M /dev/vg0/data
# resize2fs /dev/vg0/data
```

For other filesystems, the **lvextend** utility can be used to add unallocated extents in the volume group to a logical volume. Then native utilities for the filesystem can be used to expand it to fill the volume. To reduce a filesystem, first use the native utilities to shrink the filesystem, then run **lvreduce** to shrink the logical volume.

Additional physical volumes can be added to a volume group to provide more unallocated extents to assign to logical volumes. The physical volumes need to be setup with **pvcreate**, then added to the volume group with the **vgextend** command.

Physical volumes can also be removed from a volume group. This is useful for removing an old disk from the volume group. The **pvmove** command can redistribute extents from the physical volume being decommissioned to the other physical volumes in the volume group. In its simplest mode, **pvmove** takes the name of the physical volume to be removed as its argument; for example:

```
# pvmove /dev/hda3
```

Once there are no extents in use on the old physical volume, it can be removed from the volume group with the **vgreduce** command:

```
# vgreduce vg0 /dev/hda3
```

Some commands are available to help you gather information about the state of your physical volumes, volume groups, and logical volumes. Three useful commands are **pvdisplay**, **vgdisplay**, and **lvdisplay**.

Additional Useful Tools

1-26

Managing the initrd Image

- The initial RAM disk provides modules loaded early in the boot process.
- This file is located under `/boot/initrd-$(uname -r).img`
- Extra modules sometimes need to be added due to:
  - New hardware added to the system. i.e. SCSI controller
  - New features needed such as USB devices.
  - Module needs to load automatically at boot time.
- Use **mkinitrd** and the **--with** option to rebuild with an extra module:

```
mkinitrd --with=module_name /boot/initrd-$(uname -r).img \
$(uname -r)
```

The initial RAM disk (also called initrd) provides the system with a series of modules that are not provided by the kernel image but which are still critical to boot the system correctly. These modules are usually associated with storage devices and filesystems, but may support other features and hardware peripherals.

The initrd file is located under `/boot/initrd-$(uname -r).img`. Although the filename typically uses the format above, the name may be modified as long as the Grub configuration file, `/boot/grub/grub.conf` refers to the right filename.

The initial RAM disk sometimes needs to be rebuilt due to new hardware (such as a new SCSI or SATA controller), new features required early during the boot process, or in order to load a module automatically. To rebuild the initrd file, the **mkinitrd** command is provided. Several options are supported, such as **--with**, which makes it possible to list modules that should be forced into the initial RAM disk. A typical command would look like:

```
mkinitrd --with=usb_storage /boot/initrd-$(uname -r).img $(uname -r)
```

where **$(uname -r)** is replaced with the current kernel version.

rsync: Efficient File Sync

- Efficiently copies files to or from remote systems
- Uses secure **ssh** connections for transport
  - **rsync *.conf barney:/home/joe/configs/**
- Faster than **scp** - copies differences in like files

Efficient network copies: ***rsync***

rsync is a program that works in much the same way that the older **rcp** does, but has many more options and uses the **rsync** remote-update protocol to greatly increase the speed of file transfers when the destination file already exists.

The **rsync** remote-update protocol allows **rsync** to transfer just the differences between two sets of files using an efficient checksum-search algorithm. This utility is useful for tasks like updating web content, because it will only transfer the changed files.

Useful options to **rsync**

-e *command*	specifies an external, rsh-compatible program to connect with (usually **ssh**)
-a	recurses subdirectories, preserving permissions, ownership, etc
-r	recurses subdirectories without preserving permissions, etc.
--partial	continues partially downloaded files
--progress	prints a progress bar while transferring
-P	is the same as **--partial --progress**

See the rsync(1) man page for a complete list

Network Inferfaces

- Networking scripts refer to logical interface names:
  - Ethernet: `eth0, eth1 ...`
  - Dial-up: `ppp0, ppp1 ...`
  - Loopback: `lo`
- Display network interfaces by using:
  - **ifconfig -a**
  - **ip link [show]**

Logical Interface Naming

The Linux kernel names interfaces with a specific prefix depending on the type of interface. For example, all Ethernet interfaces start with `eth`, regardless of the specific hardware vendor. Following the prefix, each interface is numbered, starting at zero. For example, `eth0`, `eth1`, and `eth2` would refer to the first, second, and third Ethernet interfaces.

Additionally, it is possible to assign multiple Layer 3 addresses to a single physical network adapter. In this case, the second, third, and so on layer 3 addresses would be assigned to individual “device aliases”. The naming of a “device alias” takes the logical adapter name appending a colon and a unique alias number: `eth0:1`, `eth0:2`, and `eth0:3` representing the second, third and fourth layer 3 addresses.

Display Hardware Configuration

The hardware address of network interfaces can be determined by running the **/sbin/ifconfig** command. Without options, **ifconfig** will only display the “active” interfaces. To view “all” interfaces, including “inactive” ones, use the **-a** option.

```
[root@stationX ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:9A:86:9D
          inet addr:172.31.53.128  Bcast:172.31.53.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9a:869d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3726 (3.6 KiB)  TX bytes:7310 (7.1 KiB)
          Interrupt:185 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2935 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2935 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:3908056 (3.7 MiB)  TX bytes:3908056 (3.7 MiB)
```

Another method to view “all” of your interfaces and their hardware addresses is to use **/sbin/ip link [show]**.

```
[root@stationX ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 00:0c:29:9a:86:9d brd ff:ff:ff:ff:ff:ff
3: sit0: <NOARP> mtu 1480 qdisc noop
   link/sit 0.0.0.0 brd 0.0.0.0
```

Speed and Duplex Settings

- Modules are configured to autonegotiate, by default
- Mismatches can cause intermittent to no communcation
- Manually overridden using:
  - **ethtool**
  - `ETHTOOL_OPTS` in `ifcfg-`*`ethX`*
  - `options` or `install` in `/etc/modprobe.conf` for older interface modules

1-30

Speed and Duplex

Network cards can negotiate speed and duplex settings with the "hub/switch" that they are attached to. Autonegotiation allows the network card and upstream "hub/switch" to transparently select the optimal setting.

When there is a mismatch in speed and/or duplex between the settings presumed by the card and the settings presumed by the upstream "hub/switch", communication becomes intermittent, at best, or non-existent. A mismatch will also exhibit error statistics of "overruns" or "frame/collsns" when viewing the interface through **/sbin/ifconfig** or **/sbin/ip -s link**

Manual Override

The speed and duplex settings for the card can be viewed and/or changed with **/sbin/ethtool**.

```
[root@stationX ~]# ethtool eth0
Settings for eth0:
       Supported ports: [ TP ]
       Supported link modes:   10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                               1000baseT/Full
       Supports auto-negotiation: Yes
       Advertised link modes:  10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                               1000baseT/Full
       Advertised auto-negotiation: Yes
       Speed: 100Mb/s
       Duplex: Full
       Port: Twisted Pair
       PHYAD: 0
       Transceiver: internal
       Auto-negotiation: on
       Supports Wake-on: umbg
       Wake-on: g
       Current message level: 0x00000007 (7)
       Link detected: yes
```

Manual changes are best made when the interface is not "in use". It is also recommended to turn off autonegotiation before "forcing" a manual setting. To manually force 100Mbps, full duplex operation on eth1:

```
[root@stationX ~]# ifdown eth1
[root@stationX ~]# ethtool -s eth1 autoneg off speed 100 duplex full
[root@stationX ~]# ifup eth1
```

To make the changes persist across a reboot, we need to incorporate the settings into the logical adapter interface configuration file, `ifcfg-`*`ethX`*, by adding the following `ETHTOOL_OPTS` line:

```
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

Note: If you are working with an older module that does not support the **ethtool** interface, you may need to turn to either specifying module parameters with an `options` line or executing **/sbin/mii-tool** via an `install` line, both in `/etc/modprobe.conf`.

Dynamic IPv4 Configuration

- Interface configuration defined in:
  - `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
  - Dynamic with line of: `BOOTPROTO=dhcp`
- Zero Configuration Networking
  - Uses `169.254.0.0/16`
  - Disabled with line of: `NOZEROCONF=yes` in `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
- Use **ifdown *`device`*; ifup *`device`*** to apply configuration changes

1-31

Dynamic IPv4 Configuration

Red Hat Enterprise Linux stores network interface configuration information in files in the `/etc/sysconfig/network-scripts/` directory. The file names are prefixed with `ifcfg-` followed by the logical adapter name. For example, the first Ethernet interface would be `ifcfg-eth0`.

DHCP (Dynamic Host Configuration Protocol) can be used to automatically obtain an IP address and other configuration parameters from a central server. The `BOOTPROTO` variable in the interface configuration file controls the use of **dhclient** to negotiate the lease:

```
[root@stationX ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0D:60:FB:CA:61
ONBOOT=yes
```

Zeroconf

If there is no DHCP server an address from the `169.254.0.0/16` network is automatically assigned. These addresses are non-routable.

The use of this IP address range can be disabled by adding a `NOZEROCONF=yes` to the interface configuration file, `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*.

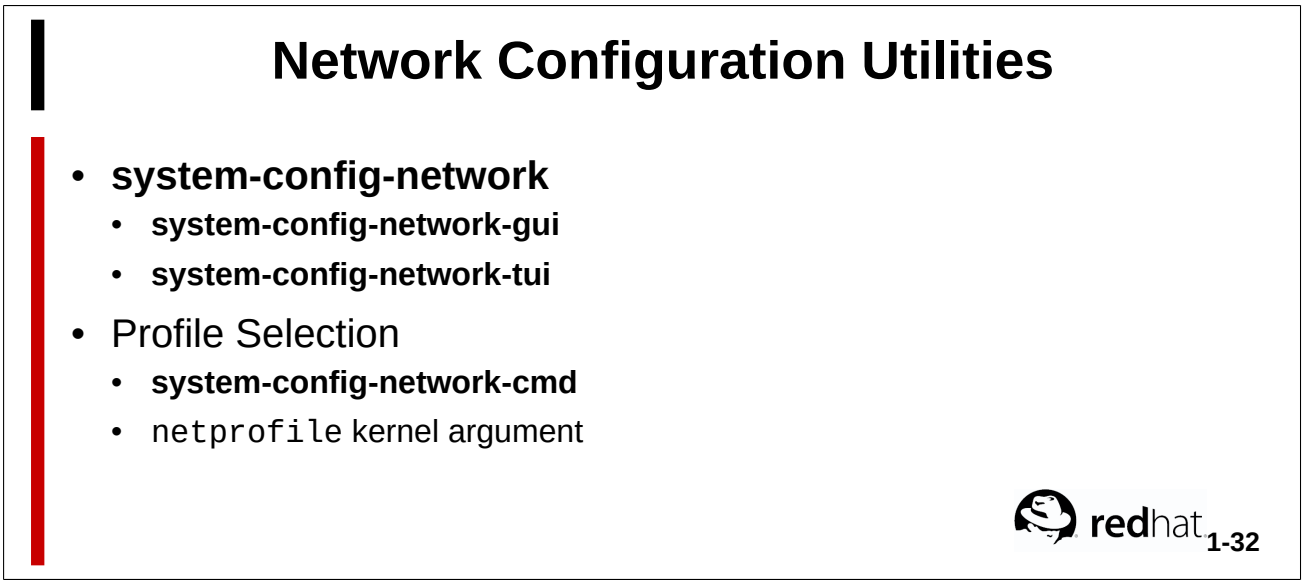
Activating Configuration Changes

After making changes to the interface configuration file, the process of bringing up a network interface involves the potential interaction of several utilities and processes. For example, a dialup interface would need to instruct the **pppd** daemon to dial the modem, or static routes might need to be added, or, as configured above, the **dhclient** daemon needs to negotiate a lease from a DHCP server. The **/sbin/ifup** and **/sbin/ifdown** scripts take care of all the extra tasks that need to be performed when activating and deactivating a network interface.

```
[root@stationX ~]# ifdown eth0
```

```
[root@stationX ~]# ifup eth0

Determining IP information for eth0... done.
```



Network Configuration Utilities

Through the Gnome or KDE panels, we can navigate to System->Administration->Network which will launch **system-config-network-gui**. From the command-line, we can enter **system-config-network** which will launch either the GUI version (**system-config-network-gui**) or the text based version (**system-config-network-tui**) depending on the environment the command has been executed from.

system-config-network provides a friendly method of modifying the files we have referred to throughout this unit. You can setup Ethernet, ISDN, PPP, xDSL, Token Ring, CIPE, and/or wireless network interfaces for static or dynamic configuration. You can define routes, hostnames, name servers, etc.

Profiles

The **system-config-network** framework, however, maintains a parallel directory structure under `/etc/sysconfig/networking/`. This opens up another feature, allowing the creation of *profiles*, the definition of multiple, alternative network configurations. When we transition from one profile to another, the files under `/etc/sysconfig/networking/` are *dynamically linked* to the normal network configuration files.

We can further automate the transition from one networking profile to another by using the **system-config-network-cmd** command directly or via a script.

```
[root@stationX ~]# system-config-network-cmd --profile ProfileName --activate
```

At boot time, we can also pass a network profile via a kernel argument perhaps as an alterative stanza in `/boot/grub/grub.conf`.

```
title RHEL5 with ProfileName networking
   root (hd0,0)
   kernel /vmlinuz-version ro root=LABEL=/ netprofile=ProfileName
   initrd /initrd-version.img
```

The use of profiles can simplify the network configuration of a system “on the move”. Usually applied to laptops, you could, for example, have one network profile for Work and another for Home. Create two entries in `/boot/grub/grub.conf` to boot your system with the appropriate network settings depending on where you are.

End of Unit 1

- Questions and Answers
- Summary
  - New Competencies: ACLs, SELinux, NTP
  - Changed Software: Server and Client varients, kernel versions, **yum**, the `/etc/pki` directory
  - Other useful tools: **mkinitrd**, **rsync**, **ip**, **ethtool**