RHCE
BOOT CAMP

System Administration

redhat®

CERTIFIED
ENGINEER

X WINDOWS

- X Windows was developed in the 1980's to provide an intelligent GUI system for UNIX.
- It is an extremely simple client/server model, that is exceptionally easy to extend, hence it's power and world-wide adoption.

LAYERS

- X Windows is built on a layered concept:
  - X Server
  - Window Manager
  - Desktop
- Also, a display manager runs to provide login services.

WINDOW MANAGERS

- Special type of X Clients which encapsulate other clients, allowing them to be moved, resized, or “iconified.” They also provide the desktop theme, configurable menus, panel utilities, and session management. RHEL ships with `metacity`, `kwin` and `twm`. These window managers provide the core functionality of the GUI.

- Generally a desktop is run in addition to the window manager, though `twm` is provided as a fallback if a desktop won’t start

DESKTOPS

- Fully integrated graphical environments, sitting on top of a window manager. Usually provides copy/paste features, lots of extra tools/utilities to run and configure a graphical environment. RHEL ships with GNOME and KDE.

- Configured with `DESKTOP` environment variable, set in

  `/etc/sysconfig/desktop`

DISPLAY MANAGER

- X equivalent of the text-based login program. RHEL 5 ships with gdm and kdm. Display managers are usually started in run-level 5 from the /etc/X11/prefdm script.

- Configured with `DISPLAYMANAGER` environment variable, set in `/etc/sysconfig/desktop`

XFREE86

- XFree86 was the first open source clone of the X Window system, released in 1991.
- XFree86 formed the de facto GUI platform for Linux, and indeed all of X Windows development for the ‘90s and into the early 2000’s
- Unfortunately, in 2004 the XFree86 project adopted a license change which GNU did not particularly care for, and almost all distributors switched to X.Org.

X.ORG

- The X.Org Server stepped into the picture in 2004 as a splinter off of the XFree86 project.

- Since they didn't muck with the license, most distributors jumped over to X.Org for their X Windows needs, and to this day X.Org remains the GUI platform of choice for Linux implementations.

CONFIGURATION

- Configuring X Windows often requires at least Bachelors in Computer Science with a Minor in Great Luck.
- The main configuration file for X.Org is `/etc/X11/xorg.conf`.
- Reading the associated man pages is a must.
- Relying on the GUI configuration tools to help with X Windows configs is a Good Idea

CONFIGURING X WINDOWS

- For initial configuration of Xorg, you can use the command `system-config-display`.

- This can be run with the `--noui` and `--reconfig` options to completely removing any existing configurations and replace it with conservative settings.

NETWORK CONFIGURATION

- There are two main approaches to configuring a machine for network access:
  - Static configuration
  - Dynamic configuration
- Static configuration uses set parameters for the configuration, which is known by the machine and the network and never changes. Generally used with servers.
- Dynamic configuration configures network machines on the fly, where a service on the network provides all configuration parameters to a machine when it joins the network. Generally used with workstations.

DYNAMIC CONFIGURATION

- Dynamic configuration is the easiest to use.
- The machine just needs to set up it's interfaces with the DHCP protocol.
- DHCP: Dynamic Host Configuration Protocol.
- A lease is obtained from the DHCP server, providing all network configuration details for the client. The lease expires after some amount of time and is renewed by the client to maintain network access.

STATIC CONFIGURATION

- Static configuration requires four configuration parameters in order to allow full network functionality:
  - IP Address
  - Netmask
  - Default Gateway or Router
  - DNS Server(s)

DNS?

- Domain Name Service: This is the glue between network names and IP addresses.

- Remember: Humans like names, computers like numbers. DNS is a service like so many others, mapping names to numbers and numbers to names. Mostly a convenience.

- Also provides for email functionality, geographic load balancing and limited service failover capabilities.

STATIC CONFIGURATION

- The first two components of static configuration are IP address and netmask.
- These provide LAN-level access
- `ifconfig`: Network Interface configuration tool
- Basic idea:
  - `ifconfig eth0 192.168.0.100 netmask 255.255.255.0`

GATEWAYS

- The third configuration parameter is the default gateway.
- Provides access to *inter-networking*, or moving from just the local LAN to other LAN's
- `route`: Kernel routing table tool
  - Displays and manipulates network routing table
  - `route add default gw 192.168.0.1`

DNS SERVERS

- Final piece of configuration information.
- List of one or more IP addresses which provide the DNS service, allowing name to IP address mapping
- Very simple to configure. Add `nameserver` lines to `/etc/resolv.conf`:
  - `nameserver 192.168.7.15`
- Also consider `/etc/nsswitch.conf`

STATIC CONFIGURATION

- Once all four pieces of information are configured on the system, full network service will be available.
- Best practice:
  - Configure IP address and netmask. Check LAN connectivity.
  - Configure default gateway. Check intra-LAN connectivity.
  - Configure DNS: Check name resolution.

ONE MORE THING...

- `ifconfig` and `route` directly manipulate the running kernel, and are not permanent changes to the system. After a reboot, changes will be lost.

- To make IP address, netmask and gateway changes permanent, you have to edit two configuration files:

  - `/etc/sysconfig/network-scripts/ifcfg-eth0`

  - `/etc/sysconfig/network`

/ETC/SYSCONFIG/NETWORK

```
NETWORKING={yes|no}

HOSTNAME=<fqdn>

GATEWAY=<gateway ip>

NISDOMAIN=<nis domain name>
```

IFCFG-* FILES

- To configure a device to use dhcp, the ifcfg file should contain the following:

```
DEVICE=eth0

BOOTPROTO=dhcp

ONBOOT=yes
```

IFCFG-* FILES

- To configure a device with static settings, the ifcfg file should contain the following:

```
DEVICE=eth0

BOOTPROTO=static

IPADDR=<ip>

NETMASK=<netmask>

ONBOOT=yes

GATEWAY=<gateway ip>  * Only if different than default
```

LAB

1. Determine your current network settings (which were assigned by DHCP) and change your machine to a static network configuration using these settings.

2. When you are satisfied with your configuration, restart the network service to put your changes into effect.

3. Test your connectivity to `server1` to make sure you are still online.

4. Refer back to DHCP settings if necessary to correct any mistakes in your static configuration.

PRINTING

- The new, and preferred printing system for Linux, is CUPS - the Common Unix Printing System.
- CUPS supports IPP protocol (based on HTTP/1.1) and can communicate with LPD print servers
- CUPS can be administered several ways:
  - by manually editing `/etc/cups/cupsd.conf` and `/etc/cups/printers.conf`
  - by using the `system-config-printer` tool, or
  - by using the web administration interface on port 631

CUPS

- CUPS tools and commands:
  - `lpstat`: used to view status of configured printers
  - `lp`: Create a print request
  - `cancel`: Cancel a pending print request
  - `lprm`: Cancel a pending print request
  - `lpadmin`: printer access control

PRINTER CONTROL

- Printing under CUPS is a two-step process.
  - First, a job is *spooled* or *queued* for printing in the print spool.
  - Second, the cups daemon pulls jobs from the print queue and feeds them to the appropriate printer.
- Access to the print queue is managed with the `accept` and `reject` commands
- Whether `cupsd` hands print jobs to the printer is controlled with the `enable` and `disable` commands.

LAB

1. Configure a “dummy” printer which doesn't correspond to any actual device, and make this the default queue on your machine.

2. Submit a text file to be printed from the command line.

3. View the status of the print job (which should never complete) using both the CLI and Web interfaces.

CRON

- `crond` is the cron daemon. Cron provides for the ability to execute commands on a regular basis.

- Generally used to run hourly, daily and weekly type system maintenance scripts.

- Also useful to run reports, cleanup jobs and much, much more.

USING CRON

- Cron is controlled through crontab files.
  - There are system-wide crons, accessible under

    `/etc/cron.*`

  - Every user has their own crontab, accessible through the `crontab` command

SYSTEM CRONS

- `/etc/crontab` defines the system cron jobs.
  - Many distributions use the `run-parts` script to execute all scripts found in `/etc/cron.hourly`, `/etc/cron.daily`, etc on the appropriate schedule.
- `/etc/crontab` defines the times for each schedule: hourly, daily, weekly, monthly

CRONTAB

- `crontab`: View, edit or remove crontabs
  - The `-l` option prints the crontab. The `-e` option opens the crontab for editing. The `-r` option removes the crontab.
- Root can work with the crontab for any user by specifying the username on the command line:
  - `crontab -e -u bob`

CRONTAB SYNTAX

- There are two main components to a crontab entry:
  - The timespec specifies when the command should be run
  - The command is what gets executed every time the timespec is matched

CRONTAB TIMESPECS

- The timespec is broken down into 5 fields, separated by spaces:
  - `minute hour day-of-month month day-of-week`
- Each field can contain a number, a range of numbers, a comma-separated list of numbers, an asterisk or a number slash division rate
- Mostly self-explanatory - some examples will help...

TIMESPEC EXAMPLES

- `0 23 * * *` *11pm every day*
- `30 * * * 1-5` *30 minutes after every hour, M-F*
- `0 7 1 * *` *7am, first of every month*
- `* * * * *` *Every single minute*
- `0,10,20,30,40,50 * * * *` *Every 10 minutes*
- `*/5 8-17 * * 1-5` *Every 5 minutes, 8am-5pm, M-F*

EXAMPLE CRONTAB

```
01 4 * * * /usr/local/bin/restart-webserver
00 8 1 * * /usr/bin/mail-report boss@mycompany.com
*/5 * * * * /monitor/bin/check-site -e admin@mycompany.com -o /var/log/check.log
```

- There are various additional options and features available to the cron system. Check the man pages for reference:
  - `cron`, `crontab` (sections 1 and 5)

LAB

1. Create a cronjob for the user root that checks the amount of available space on the system every Friday at 12:34pm.

2. Create a cronjob as a regular user that lists the contents of `/tmp` at 3:54am on Sunday, January 2.

LOGS

- One of the easiest places to find the cause of a problem is in the log files.

- Log files store informational messages from software. The types of messages include debug information, status information, warnings, errors and more.

- Some applications manage their own log files. Others use the system-wide logging package...

SYSLOG

- `syslog` - The system logger. A framework consisting of a library, a daemon, a configuration file and logs.
- Any application can use the library and log messages through `syslog` with simple function calls.
- Log messages consist of 3 parts:
  - Facility
  - Level
  - Message

SYSLOG

- The facility describes what part of the operating system generated the message, and is selected by the software:
  - `auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0-local7`
- The level represents the importance of the message, and is also chosen by the software:
  - `emergency, alert, critical, error, warning, notice, info, debug`

/ETC/SYSLOG.CONF

- `/etc/syslog.conf` defines where all of the log messages should go. Destinations include files, screens of logged in users, console, other syslog servers.
- Basic file format:
  - `facility.level  destination`
- Examples:
  - `*.err  /dev/console`
  - `mail.*  /var/log/maillog`
  - `*.info;mail.none;authpriv.none  /var/log/messages`

/VAR/LOG

- `maillog`: messages from the email subsystem
- `secure`: authentication and security messages
- `cron`: cron messages
- `boot.log`: boot messages
- `messages`: catch-all
- `dmesg` : hardware and kernel events generated before syslogd

LOGS

- As mentioned earlier, not all software uses the syslog framework to handle it's logging. Quite a bit of software manages it's own logs.

- This can make it difficult to track down all of the log locations on an unfamiliar system. The best way to handle this is to start from the init scripts...

LOCATING APPLICATION LOGS

- To track down the log file location for an application, you need to find it's configuration file so you can see where the logs are being written.

- Of course, finding the configuration file might be just as difficult, so it's best to start at the source.

- `init` starts all of the system services, and so there is an init script somewhere that is starting up the application in question.

- The init script almost always references the configuration file

LOCATING APPLICATION LOGS

- Now that the configuration file location is known, it only takes a few moments to scan through it and find out where logs are being written.

- As for the format of the log file, that's completely dependent on the application. Some will be similar to syslog, others, like Apache or Qmail, will be completely foreign looking.

- Fortunately, a little common sense and judicious application of Google Ointment will get the information you seek.

MAINTAINING LOGS

- `/etc/logrotate.conf`
  - This is the main configuration file for logrotate.
- `/etc/logrotate.d/`
  - EVERYTHING in this directory will be parsed as if it is a logrotate configuration file. Usually, applications such as Apache and Sendmail will have configuration files in this directory to control how their logs will be rotated.
- `logrotate -vf /etc/logrotate.conf`
  - Can be run as root at any time to force log rotation and check for errors.

LAB

1. Take a few minutes to browse through the various logs in `/var/log`. Familiarize yourself with the kinds of information available.

2. Browse the man page for `syslog.conf`

3. Find where the audit service keeps its log and add a corresponding new entry to your logrotate configuration.

```
slideshow.end();
```