ORACLE UNIVERSITY

Oracle Linux 7: Advanced Administration

Student Guide - Volume I
D90758GC10
Edition 1.0 | September 2015 | D92963

ORACLE

Author
Craig McBride

Technical Contributors
and Reviewers
Avi Miller
Elena Zannoni
Wim Coekaerts
Harald Van Breederode
Joel Goodman
Manish Kapur
Yasar Akthar
Antoinette O'Sullivan
Gavin Bowe
Steve Miller
Herbert Van Den Bergh
Todd Vierling
John Haxby

Editors
Malavika Jinka
Aju Kumar

Graphic Editors
Kavya Bellur
Maheshwari Krishnamurthy

Seema Bopaiah
Publishers
Veena Narasimhan
Pavithran Adka
Raghunath M

Contents

1

Introduction

Course Goals

In this course, you learn how to:

- Configure network addressing and name services
- Configure authentication and directory services
- Configure Pluggable Authentication Modules (PAM)
- Configure web and email services
- Perform a Kickstart installation
- Configure Samba services
- Perform advanced software package management
- Perform advanced storage administration
- Perform advanced network configuration
- Configure Oracle Cluster File System version 2 (OCFS2)

Course Goals

- Configure iSCSI targets and initiators
- Configure device multipathing
- Configure control groups (Cgroups)
- Configure Kernel-based Virtual Machine (KVM)
- Configure Linux containers (LXC)
- Configure containers by using Docker
- Configure SELinux
- Enable core dump and perform core dump analysis
- Configure dynamic tracing (DTrace)

Schedule

Session	Module
Day 1	Lesson 1: Introduction Lesson 2: Network Addressing and Name Services Lesson 3: Authentication and Directory Services
Day 2	Lesson 4: Pluggable Authentication Modules (PAM) Lesson 5: Web and Email Services Lesson 6: Installing Oracle Linux by Using Kickstart Lesson 7: Samba Services
Day 3	Lesson 8: Advanced Software Package Management Lesson 9: Advanced Storage Administration Lesson 10: Advanced Networking

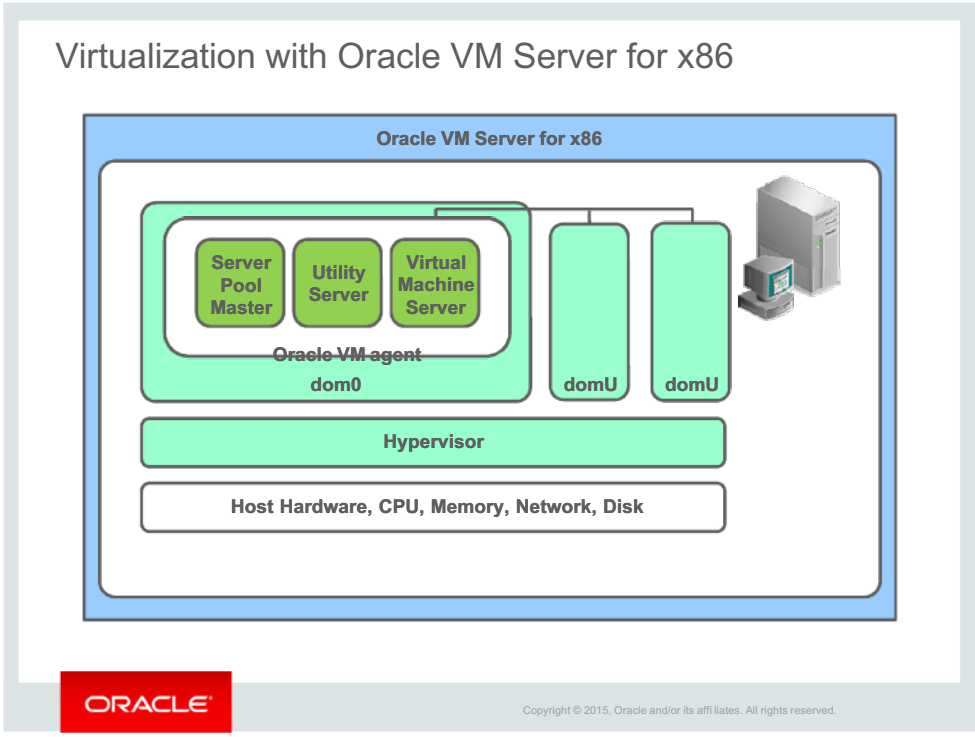
Schedule

Session	Module
Day 4	Lesson 11: OCFS2 and Oracle Clusterware Lesson 12: iSCSI and Multipathing Lesson 13: Control Groups (Cgroups) Lesson 14: Virtualization with Linux
Day 5	Lesson 15: Linux Containers (LXC) Lesson 16: Docker Lesson 17: SELinux Lesson 18: Core Dump Analysis Lesson 19: Dynamic Tracing with DTrace

Objectives

After completing this lesson, you should be able to:

- Describe the classroom environment used for the practice sessions
- Start, log in to, and stop a virtual machine on your student desktop



Virtualization

Virtualization allows you to use one server and its computing resources to run one or more guest operating system and application images concurrently, sharing those resources among the guests.

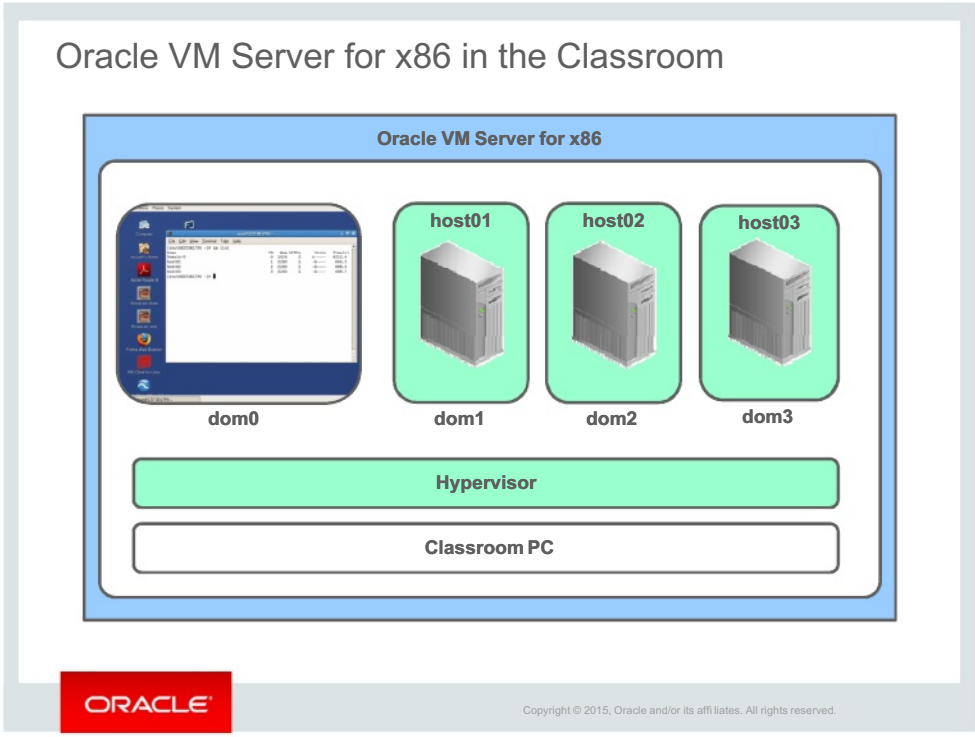
Hypervisor

A hypervisor is virtualization software, also known as a virtual machine monitor (VMM), which creates and runs virtual machines. There are two different types of hypervisors:

- A type 2 hypervisor, such as VirtualBox, which runs on the host operating system and in turn runs the guest virtual machines. A type 2 hypervisor is a distinct software layer.
- A type 1 hypervisor, such as Oracle VM Server for x86 or VMware ESX, which provides a small footprint host operating system and exposes the server's resources to the guest virtual machines that run directly on top of the hypervisor. Because this type of hypervisor communicates directly with the hardware, it is known as a bare metal hypervisor.

Oracle VM Server for x86 Domains

Oracle VM Server for x86 guests are referred to as *domains*. Dom0 is always present, providing management services for the other domains running on the same server.



Self-Contained Multi-Host Environment

Your student PC is running Oracle VM Server for x86, where you can run up to three guests (as required) to work through the practice sessions. Guests running on your machine can see each other and can see outside the environment. Out of the box, Oracle VM Server for x86 does not offer a GUI front end; however, your dom0 has been modified to include the GNOME interface. When you log in to the machine, you are presented with a graphical interface that can also act as an X-server for your guests.

Logging In to Your Machine

Log in as the `vncuser` user (password is `vnctech`). This logs you in to dom0 and the GNOME GUI. When you are logged in, the simplest way to control your machine is from terminal sessions initiated from the GNOME desktop.

Where to Find Your Guests

The guest VMs reside in their own directories under the `/OVM/running_pool` directory on dom0. For example, the files for the host01 VM reside in the `/OVS/running_pool/host01` directory and the files for the host02 VM reside in the `/OVS/running_pool/host02` directory.

Working with Classroom Virtual Machines

- Use the `xm` command-line tool to manually manage guests.
  - `xm list`: Lists the currently active guests
  - `xm create vm.cfg`: Starts a VM guest
  - `xm shutdown -w <VM_name>`: Gracefully shuts down the specified VM
  - `xm destroy <VM_name>`: Performs a nongraceful shutdown
  - `xm reset <VM_name>`: Resets the specified VM
- Use the following commands to connect to VM guests:
  - `ssh <VM_name>`
  - `vncviewer` (provide `localhost:<VNC_port_number>` when prompted)
- Each practice specifies whether to use `ssh` or `vncviewer`.

Starting, Stopping, and Listing Guests

When you are logged in to dom0, you can switch to `root` (password is `oracle`) in a terminal session and use the `xm` command-line tool to manually manage guests on the machine.

- **`xm list`**: Lists all the currently active guests, including dom0 itself
- **`xm create vm.cfg`**: Creates a running instance of the specified VM
- **`xm shutdown -w <VM_name>`**: Shuts down the specified VM and waits for the action to complete before returning control to you
- **`xm destroy <VM_name>`**: Immediately shuts down the specified VM
- **`xm reset <VM_name>`**: Resets the specified VM. Use this command when you are unable to connect to the VM.

Connecting to Guests

The practice exercises direct you to become the `root` user on dom0 and use secure shell (`ssh`) or `vncviewer` to connect from dom0 to your guests. For example, to use secure shell to create a connection from dom0 to the host01 guest:

```
# ssh host01
```

The `root` password is `oracle` (all lowercase) on all the guests.

Using `vncviewer`

In some of the practices, you are instructed to use the `vncviewer` command, instead of the `ssh` command, to connect to your guest VM. You can use `vncviewer` in all practices, but it is required only when specified in the applicable practices.

Before running the `vncviewer` command, you first must determine the port number by running the following command from dom0:

```
# xm list -l host03 | grep location
            (location 0.0.0.0:5903)
            (location 3)
```

In this example, the port number for host03 is `5903`. Each host has a different port number.

Port numbers are assigned when the virtual machine is started. Therefore, run the above `xm list` command before running `vncviewer` because the port number can change.

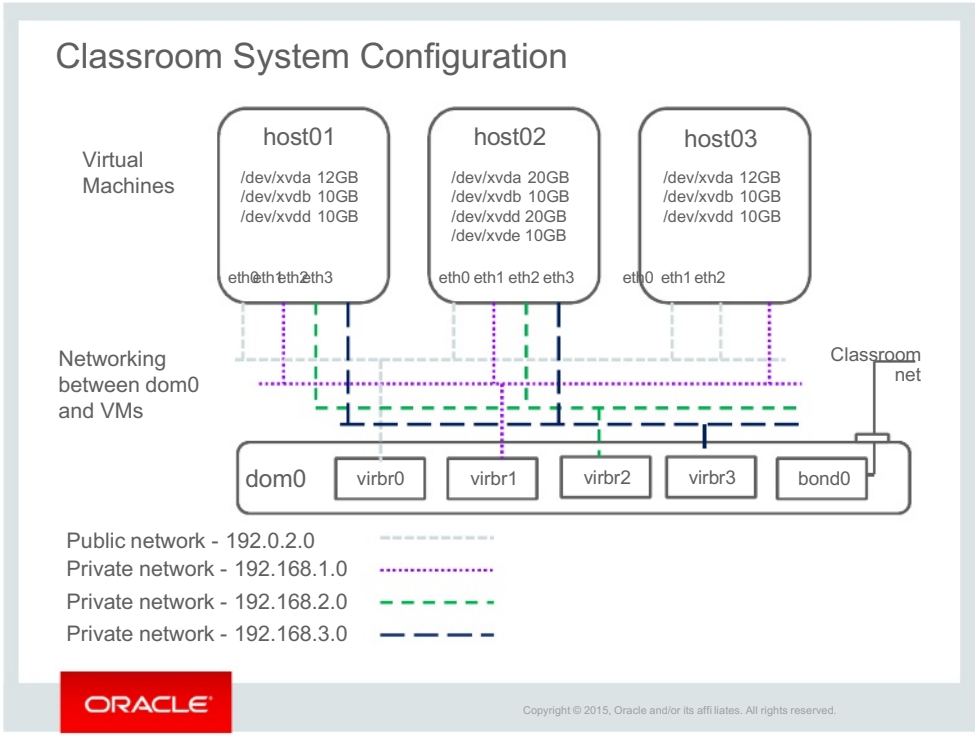
Run the `vncviewer` command and you are prompted for **VNC Viewer: Connection Details**. Enter `localhost:<port_number>`, substituting the port number obtained from the previous `xm list` command. Example:

```
# vncviewer&
localhost:5903
```

Instead of entering `5903` as the port number, you can omit "`590`" and enter "`3`" as follows:

```
localhost:3
```

You are instructed to use `vncviewer` in the practices that require access to the GNOME desktop.



Network Configuration

This slide shows the network configuration of each classroom system, as well as the disk devices for each VM guest. Communication to the classroom network is provided through the `bond0` interface on each classroom system. In addition, four network bridges are configured on dom0:

- `virbr0`: A Xen bridge interface used by the VM guests on the `192.0.2` public subnet. The IP address of `virbr0` is `192.0.2.1`.
- `virbr1`: A Xen bridge interface used by the VM guests on the `192.168.1` private subnet. The IP address of `virbr1` is `192.168.1.1`.
- `virbr2`: A Xen bridge interface used by the VM guests on the `192.168.2` private subnet. The IP address of `virbr2` is `192.168.2.1`.
- `virbr3`: A Xen bridge interface used by the VM guests on the `192.168.3` private subnet. The IP address of `virbr3` is `192.168.3.1`.

The host01 and host02 VM guests have the same network configuration. These VM guests have four virtual network interfaces.

- `eth0`: This network interface is on the `192.0.2` subnet. On host01, `eth0` has an IP address of `192.0.2.101`. On host02, `eth0` has an IP address of `192.0.2.102`.
- `eth1`: This network interface is on the `192.168.1` subnet. host01 obtains an IP address by using DHCP for `eth1`. The `eth1` interface on host02 has an IP address of `192.168.1.102`.
- `eth2`: This network interface is on the `192.168.2` subnet and has no IP address.
- `eth3`: This network interface is on the `192.168.2` subnet and has no IP address.

The host03 VM guest has two virtual network interfaces on the `192.0.2` subnet, `eth0` and `eth1`, and one interface, `eth2`, on the `192.168.1` subnet.

- `eth0`: On host03, `eth0` has an IP address of `192.0.2.103`.
- `eth1`: On host03, `eth1` has an IP address of `192.0.2.104`.
- `eth2`: On host03, `eth2` has an IP address of `192.168.1.103`.

The `192.0.2` interfaces are used in the iSCSI multipathing practice.

Dom0 is configured as a DNS server for name resolution on the Internet. Dom0 has the following entries in the `/etc/resolv.conf` file:

```
# cat /etc/resolv.conf
search edu.oracle.com
nameserver 192.0.2.1
nameserver 152.68.154.3
nameserver 10.216.106.3
```

The `edu.oracle.com` search domain and the three name servers provide name resolution services on the Internet.

Disk Configuration

The host01 and host03 VM guests have the same disk configuration. Each of these VMs has three virtual block devices.

- `/dev/xvda`: This is a 12 GB system disk that contains the Oracle Linux operating system.
- `/dev/xvdb`: This is a 10 GB utility disk that is used in the various hands-on practices.
- `/dev/xvdd`: This is a 10 GB utility disk that is used in the hands-on practices.

The host02 VM guest has four virtual block devices.

- `/dev/xvda`: This is a 20 GB system disk that contains the Oracle Linux operating system.
- `/dev/xvdb`: This is a 10 GB shared disk that is used in the OCFS2 practice.
- `/dev/xvdd`: This is a 20 GB utility disk that is used in the Linux Containers practice.
- `/dev/xvde`: This is a 10 GB utility disk that is used in the iSCSI practice.

There are four additional virtual machines, not shown in the slide, which are used in specific practices.

Local Yum Repository

- Your VMs are configured to access a local Yum Repository on **dom0**.
- The following `vm.repo` file exists on each VM, which points to the repository on **dom0** (`192.0.2.1`):

```
# cat /etc/yum.repos.d/vm.repo
[OL7.1Dom0]
Name="Oracle Linux 7.1 Dom0 Repo"
baseurl=http://192.0.2.1/repo/OracleLinux/OL7/1/x86_64
enabled=1
gpgkey=http://192.0.2.1/repo/OracleLinux/OL7/1/x86_64/RPM-
    GPG-KEY-oracle
gpgcheck=1
```

A local Yum repository exists on dom0. You use the `yum` command to install and upgrade software packages on your VMs from this local Yum repository. A `vm.repo` file exists in the `/etc/yum.repos.d` directory, which points to this local Yum repository.

```
# ls /etc/yum.repos.d
public-yum-ol7.repo    vm.repo
```

All repositories are disabled in the `public-yum-ol7.repo` file. Only `vm.repo` has an enabled repository, which points to the local Yum Repository on dom0.

On dom0, the Oracle Linux 7.1 ISO files exist in the following directory:

```
# ls /var/www/html/repo/OracleLinux/OL7/1/x86_64
addons  images    RELEASE-NOTES-U1-en       RPM-GPG-KEY-oracle
EFI     isolinux  RELEASE-NOTES-U1-en.html  TRANS.TBL
EULA    LiveOS    repodata
GPL     Packages  RPM-GPG-KEY
```

Summary

In this lesson, you should have learned how to:

- Log in to your classroom PC
- Start and stop the guest VMs on your classroom PC
- Log in to the guest VMs on your classroom PC

Practice 1: Overview

This practice covers the following topics:

- Logging in to your classroom PC *
- Exploring the dom0 configuration and directory structure
- Starting, stopping, and listing VM guests
- Logging in to each VM guest
- Exploring the VM configurations
- Logging off from your classroom PC

* See the appendix titled “Remote Access Options” for information about connecting to the classroom PC remotely.

2

Network Addressing and Name Services

Objectives

After completing this lesson, you should be able to:

- Describe DHCP
- Configure a DHCP server
- Start and stop a DHCP server
- Configure a DHCP client and request a lease
- Describe DNS
- Describe nameserver types
- Describe BIND
- Configure a cache-only nameserver
- Describe and configure zone files
- Describe and configure reverse name resolution
- Use the `rndc` utility
- Use the `host` and `dig` utilities

Introduction to DHCP

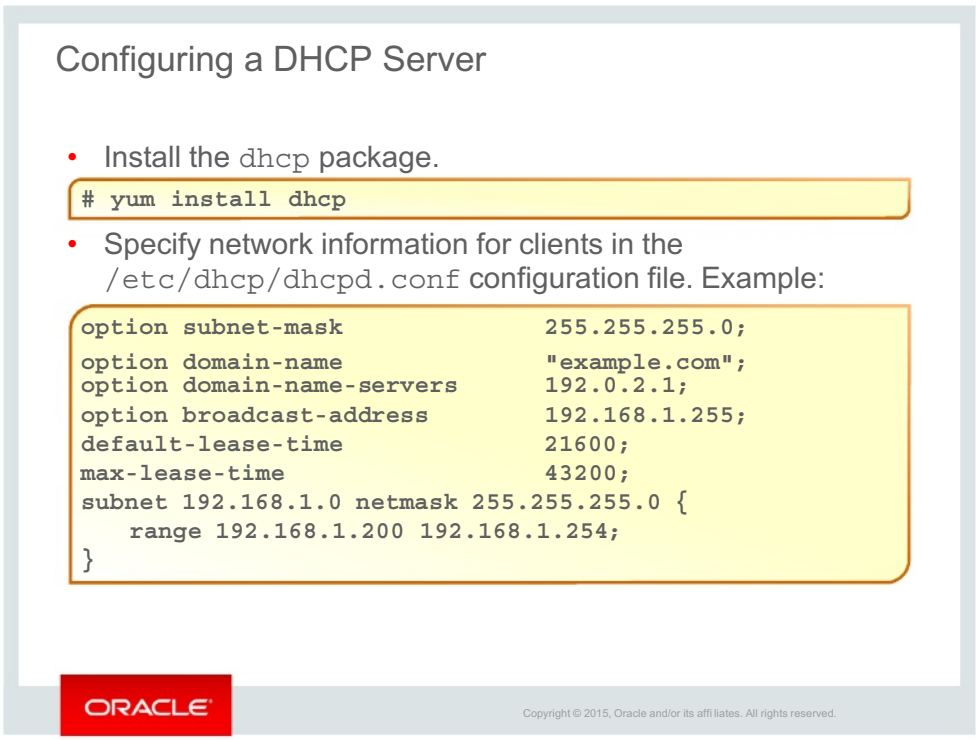
- Client machines automatically obtain network configuration information from a DHCP server.
- The client “leases” the network information.
  - The terms of the lease are configurable.
  - The lease is renewed automatically by the client while the network is in use.
- The DHCP server can provide static IP addresses.
- DHCP is broadcast based.
  - This requires the client and the server to be on the same subnet.

Dynamic Host Configuration Protocol (DHCP) allows client machines to automatically obtain network configuration information from a DHCP server each time they connect to the network. The DHCP server is configured with a range of IP addresses and other network configuration parameters.

When the client machine is configured to use DHCP, the client daemon, `dhclient`, contacts the server daemon, `dhcpd`, to obtain the networking parameters. Because DHCP is broadcast based, both the client and the server must be on the same subnet.

The server provides a lease on the IP address to the client. The client can request specific terms of the lease, such as its duration. The server can also be configured to limit the terms of the lease. While connected to the network, `dhclient` automatically renews the lease before it expires. You can configure the DHCP server to provide the same IP address each time to specific clients.

The advantages of using DHCP include ease of adding a new client machine to the network and centralized management of IP addresses. In addition, the number of total IP addresses needed is reduced because IP addresses can be reused. DHCP is also useful if you want to change the IP addresses of a large number of systems. Instead of reconfiguring each system individually, edit the DHCP configuration file on the server and enter the new set of IP addresses.



To configure a system as a DHCP server, install the `dhcp` package:

```
# yum install dhcp
```

The main configuration file for DHCP is `/etc/dhcp/dhcpd.conf`. Use this file to store network information for the clients. A sample configuration file is also installed with the `dhcp` package: `/usr/share/doc/dhcp-<version>/dhcpd.conf.sample`. You can copy this file to `/etc/dhcp/dhcpd.conf` and use it as a template.

Options

Information in the `option` lines is sent to each client when it requests a lease. Each option declaration begins with the "option" keyword, followed by the option name, and the option data. Each option declaration must be terminated with a semicolon (`;`). See the `dhcp-options(5)` man page for a description of available options.

Lease Times

There are time-related configuration entries in the sample configuration file. These are described as follows. Each of these entries is also terminated with a semicolon.

- **`default-lease-time`**: Specifies the number of seconds the IP lease remains valid if the client requesting the lease does not specify a duration
- **`max-lease-time`**: Specifies the maximum number of seconds allowed for a lease

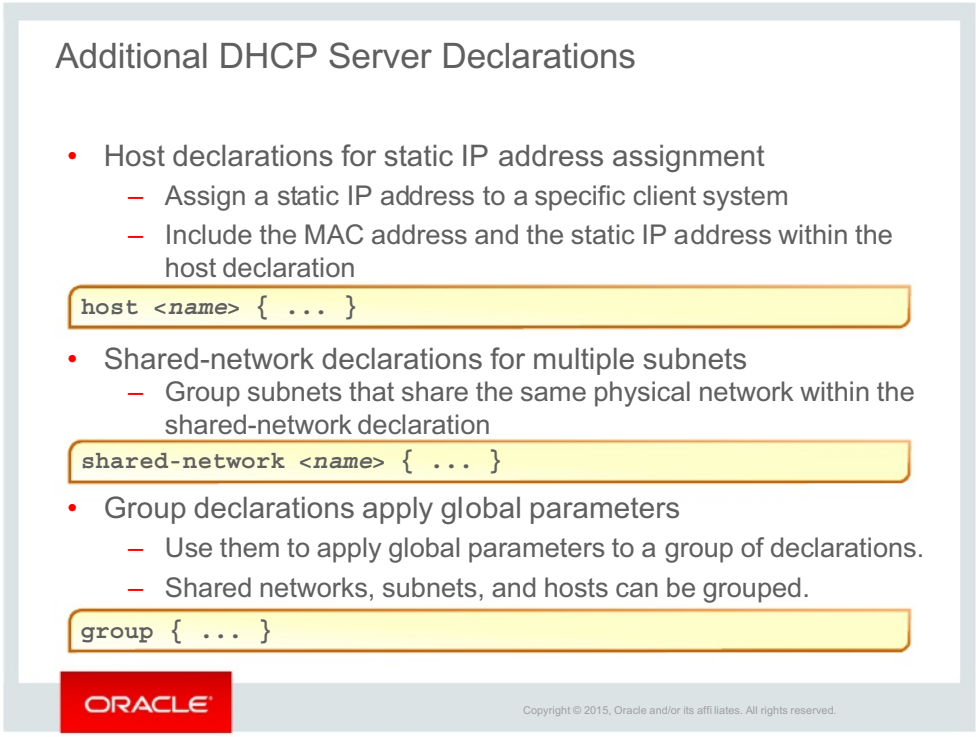
Subnet Declaration

The `subnet` declaration defines a range of IP addresses that the DHCP server can assign to clients. The example in the slide includes a subnet declaration for the `192.168.1.0` subnet. The declaration defines a netmask value and a range of IP addresses between `192.168.1.200` and `192.168.1.254`. The range of addresses is surrounded by curly braces ({ }).

You can declare multiple subnets and specify parameters inside or outside of the braces. Parameters specified within the braces apply to the clients on the subnet. In the following example, two subnet declarations are defined. The options apply only to the clients on the `192.168.1.0` subnet:

```
subnet 192.168.1.0 netmask 255.255.255.0 {

     option domain-name              "example.com";
     option domain-name-servers      192.168.1.1;

     range 192.168.1.200 192.168.1.254;

}

subnet 192.168.2.0 netmask 255.255.255.0 {

     range 192.168.2.200 192.168.2.254;

}
```

Parameters configured outside of a subnet declaration are global and apply to all client systems.



Host Declaration

Use a `host` declaration to provide a static IP address to a specific client system. Include the MAC address of the client and the static IP address to be assigned to the client. Example:

```
host host01 {
      hardware ethernet      00:16:3E:00:01:01;
      fixed-address          192.168.1.101;
      max-lease-time         84600;
}
```

In this example, IP address of `192.168.1.101` is always assigned to the system with the MAC address of `00:16:3E:00:01:01`. The `max-lease-time` included within the declaration is specific to this host and overrides the global parameter defined outside the curly brackets.

Shared-Network Declaration

Declare all subnets that share the same physical network within a `shared-network` declaration. Parameters within the shared network, but outside the enclosed subnet declarations, are considered to be global parameters.

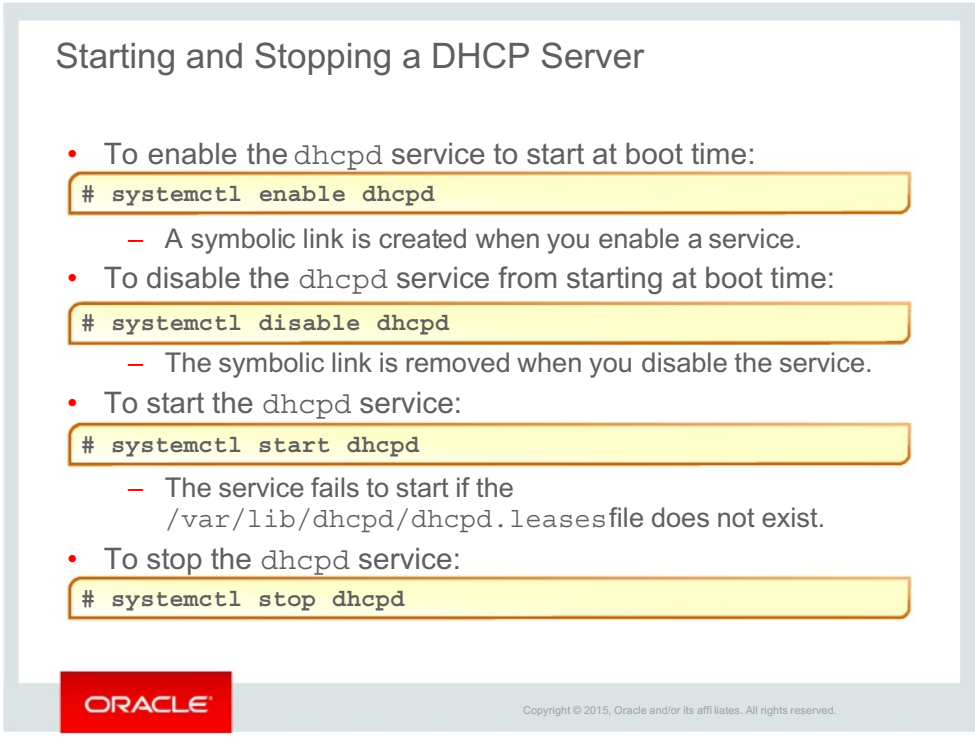
The following is an example of a shared-network declaration with two subnets. The `routers` parameter applies to both subnets:

```
shared-network name {
     option routers  192.168.0.254;
     subnet 192.168.1.0 netmask 255.255.252.0 {
          range 192.168.1.200 192.168.1.254;
     }
     subnet 192.168.2.0 netmask 255.255.252.0 {
          range 192.168.2.200 192.168.2.254;
     }
}
```

Group Declaration

Use the `group` declaration to apply global parameters to a group of declarations. Shared networks, subnets, and hosts can be grouped. The following is an example of a `group` declaration with two `host` declarations:

```
group {
     option routers  192.168.1.254;
     host host01 {
          hardware ethernet     00:16:3E:00:01:01;
          fixed-address         192.168.1.101;
     }
     host host02 {
          hardware ethernet     00:16:3E:00:01:02;
          fixed-address         192.168.1.102;
     }
}
```



Use the `systemctl` command to enable the `dhcpd` service to start at boot time:

```
# systemctl enable dhcpd
ln -s '/usr/lib/systemd/system/dhcpd.service'
'/etc/systemd/system/multi-user.target.wants/dhcpd.service'
```

Notice that the command enables a service by creating a symbolic link for the lowest-level system-state target at which the service starts. In the example, the command creates the symbolic link `dhcpd.service` for the `multi-user` target.

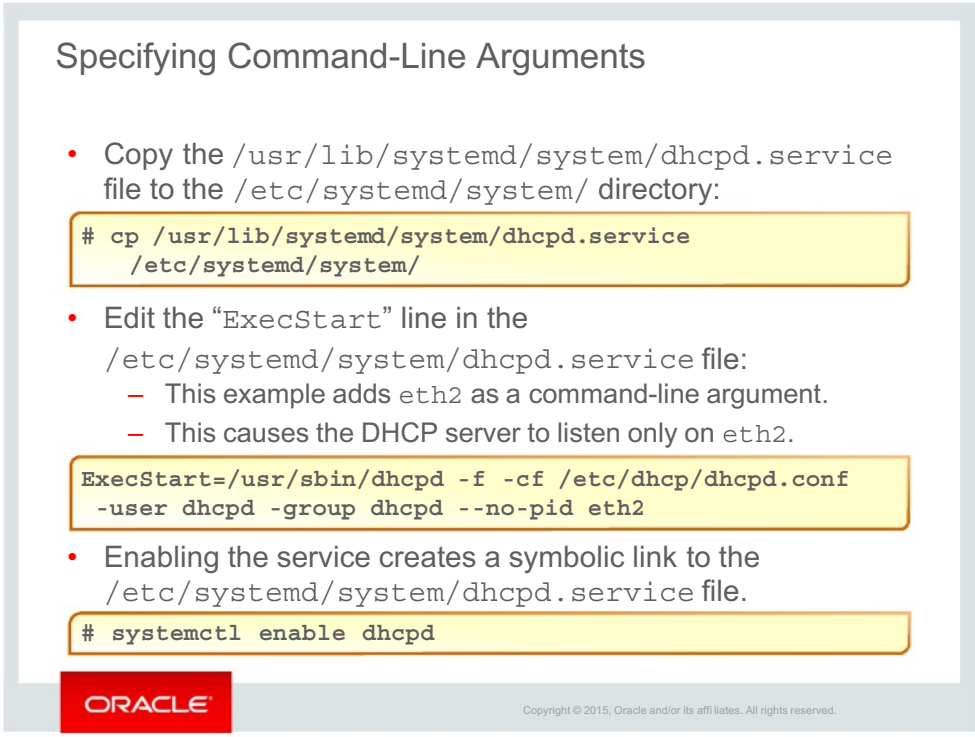
Use the `systemctl` command to disable the `dhcpd` service from starting at boot time. The symbolic link is removed when the service is disabled:

```
# systemctl disable httpd
rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

Use the `systemctl` command to start the `dhcpd` service:

```
# systemctl start httpd
```

The `dhcpd` service fails to start if the `/var/lib/dhcpd/dhcpd.leases` file does not exist. You can use the `touch` command to create the file. The `dhcpd.leases` file stores the client lease information. Do not edit this file.



To specify command-line arguments and options when the `dhcpd` service is started, copy the `/usr/lib/systemd/system/dhcpd.service` file to the `/etc/systemd/system/` directory:

```
# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
```

You can then edit the `/etc/systemd/system/dhcpd.service` file and append command-line arguments and options to the `ExecStart` line.

For example, if your DHCP server has multiple network interfaces (`eth0`, `eth1`, `eth2`) but you want only the `dhcpd` service to listen for DHCP requests on `eth2`, include `eth2` as a command-line argument:

```
# vi /etc/systemd/system/dhcpd.service
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user
dhcpd -group dhcpd --no-pid eth2
```

When you enable the service to start at boot time, a symbolic link is created to the `dhcpd.service` file in the `/etc/systemd/system/` directory rather than the file in the `/usr/lib/systemd/system/` directory:

```
# systemctl enable dhcpd
ln -s '/etc/systemd/system/dhcpd.service'
'/etc/systemd/system/multi-user.target.wants/dhcpd.service'
```

Refer to the `dhcpd(8)` man page for additional command-line options and arguments. Some of the available options are described:

- **`-p <port>`**: Specifies the UDP port number on which `dhcpd` listens. The default is port 67.
- **`-f`**: Runs the `dhcpd` as a foreground process instead of a background daemon. This is helpful when debugging a problem.
- **`-d`**: Logs the DHCP server daemon to the standard error descriptor. This is helpful when debugging. If this is not specified, `dhcpd` logs all output using `syslog`.
- **`-cf <filename>`**: Specifies the location of the configuration file. The default configuration file is `/etc/dhcp/dhcpd.conf`.
- **`-lf <filename>`**: Specifies the location of the lease database file. The default lease file is `/var/lib/dhcpd/dhcpd.leases`.
- **`-q`**: Specifies to be quiet at startup. This suppresses printing of the entire copyright message when starting the daemon.
- **`--no-pid`**: Disables writing pid (Process ID) files. With this option, the service does not check for an existing server process.

Configuring a DHCP Client

1. Install the `dhclient` package.
2. Set `BOOTPROTO=dhcp` in the `/etc/sysconfig/network-scripts/ifcfg-<device>` file.
3. Ensure that the `network` service is running on the client.
4. Optionally, enter any custom configuration information in the DHCP client configuration file, `/etc/dhcp/dhclient.conf`.
5. Run the `dhclient` command to request a lease from the server.

After being configured to use DHCP, the `dhclient` command runs at boot time.

To configure a system as a DHCP client, install the `dhclient` package:

```
# yum install dhclient
```

Change the `BOOTPROTO` directive in the `/etc/sysconfig/network-scripts/ifcfg-<interface>` file for the device to `dhcp`. For example, to use DHCP on `eth1`, perform the following:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth1
BOOTPROTO=dhcp
```

You also need to start the network service on the DHCP client.

```
# systemctl start network
```

The next time the client system connects to the network, `dhclient` requests a lease from the DHCP server and configures the client's network interface. You can also run `dhclient` from the command line to request a lease and make a connection:

```
$ dhclient
```

To request on a specific interface, include the interface as an argument. The following example only requests a lease for `eth1`:

```
$ dhclient eth1
```

The DHCP client configuration file, `/etc/dhcp/dhclient.conf`, is required only for custom configurations. A sample file exists in `/usr/share/doc/dhclient-<version>/dhclient.conf.example`.

When the client has requested and established a lease, information about the lease is stored in `/var/lib/dhclient/dhclient.leases`. Example:

```
# cat /var/lib/dhclient/dhclient.leases
lease {
      interface "eth1";
      fixed-address 192.168.1.251;
      option subnet-mask 255.255.255.0;

      option dhcp-lease-time 21600;
      option dhcp-message-type 5;
      option domain-name-servers 192.0.2.1;
      option dhcp-server-identifier 192.168.1.103;
      option broadcast-address 192.168.1.255;
      option domain-name "example.com";
      renew 5 2015/04/17 20:18:58;
      rebind 6 2015/04/17 23:15:26;
      expire 6 2015/04/17 00:00:26;
}
```

Introduction to DNS

- DNS is a network service that maps, or resolves, domain names to their respective IP addresses.
  - `wiki.us.oracle.com` > `139.185.51.248`
- DNS performs the function of the `/etc/hosts` file, but on the Internet.
- The DNS database is hierarchical and distributed.
  - Each level of hierarchy is divided by a period (`.`)
- Resolution occurs from right to left:
  1. `com` is resolved.
  2. `oracle.com` is resolved.
  3. `us.oracle.com` is resolved.
  4. The IP address of `wiki.us.oracle.com` is returned to the client.
- DNS servers are called nameservers.

Domain Name System (DNS) is a network service that maps, or resolves, domain names to their respective IP addresses. It reduces the need for users to remember IP addresses because they can refer to machines on the network by name. The mapping done by `/etc/hosts` on a small local area network (LAN) is handled by DNS on large networks, including the Internet.

The DNS database is hierarchical and distributed. Each level of the hierarchy is divided by a period (`.`). Consider the following example of a fully qualified domain name (FQDN):

```
wiki.us.oracle.com.
```

The root domain is represented by a period (`.`) and is frequently omitted except in zone files. The top-level domain in this example is `com`, `oracle` is a subdomain of `com`, `us` is a subdomain of `oracle`, and `wiki` is the host name. For administrative purposes, each of these domains is grouped into zones. A DNS server, called a nameserver, holds all the information to resolve all domains within a zone. The DNS server for a zone also holds pointers to DNS servers responsible for resolving a domain's subdomains.

When a client requests resolution of `wiki.us.oracle.com` from a nameserver, if the nameserver cannot resolve the FQDN, it queries a root nameserver, which returns the nameserver that can resolve `com`. This nameserver is queried and returns the nameserver to resolve `oracle.com`. This nameserver is queried and returns the nameserver to resolve `us.oracle.com`, which is queried and returns the IP address for the FQDN to the client.

Nameserver Types

- Authoritative nameservers:
  - Answer queries about names that are part of their zones only
  - Can be either primary (master) or secondary (slave)
- The primary nameserver holds the master copy of zone data.
- Secondary nameservers copy zone data from master nameserver or another slave nameserver.
- Caching-only, or recursive, nameservers:
  - Offer resolution services, but are not authoritative
  - Cache the answers from previous queries
  - Respond to queries from the cache if possible
    - Otherwise, they forward the query to an authoritative server.

An authoritative nameserver responds to queries about names that are part of their zones only. Authoritative nameservers can be either primary (master) nameservers or secondary (slave) nameservers. Each zone has at least one authoritative DNS server. A DNS query returns information about a domain and specifies which DNS server is authoritative for that domain.

A primary nameserver, or master nameserver, is the authoritative server that holds the master copy of zone data. Secondary nameservers, or slave nameservers, are also authoritative but copy zone information from the master nameserver or from another slave nameserver. A nameserver can also serve as a primary or secondary server for multiple zones at the same time.

Caching-only nameservers, or recursive nameservers, offer resolution services but they are not authoritative for any zone. These DNS cache nameservers store answers to previous queries in cache (memory) for a fixed period of time. When a caching-only nameserver receives a query, it answers from cache if it can. If it does not have the answer in cache, it forwards the query to an authoritative server.

Although it is not recommended for reasons of security, nameservers can also be configured to give authoritative answers to queries in some zones, while acting as a caching-only nameserver for all other zones.

BIND

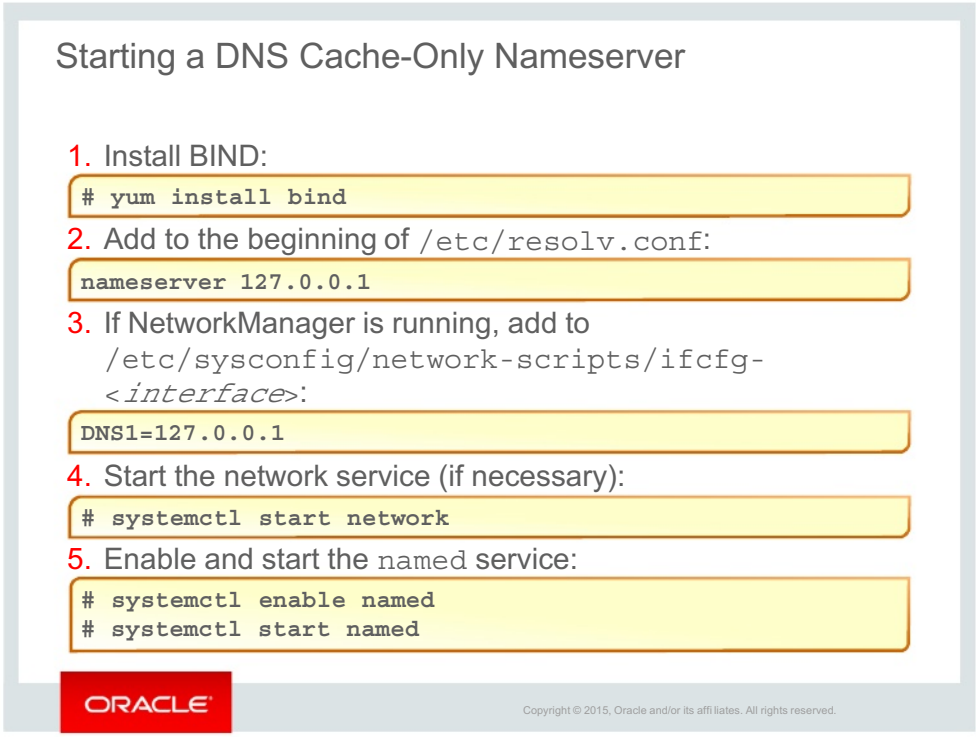
- The DNS server included in Oracle Linux is called BIND.
- The BIND server daemon is `named`.
- The remote administration utility is `rndc`.
- Configuration files and directories include:
  - `/etc/named.conf`: The main configuration file
  - `/var/named`: The default directory for storing zone files
  - `/etc/named.rfc1912.zones`: The base configuration file for implementing a caching-only nameserver
  - `/var/named/named.ca`: Contains a list of the 13 root authoritative DNS servers
- The default installation provides a caching-only nameserver.

The DNS server included in Oracle Linux is called Berkeley Internet Name Domain (BIND). BIND includes the DNS server daemon, `named`, tools for working with DNS such as the `rndc` administration utility, and several configuration files.

The primary BIND configuration files and directories are:

- **`/etc/named.conf`:** The main configuration file that lists the location and characteristics of all your domain's zone files. See the `named.conf(5)` man page for configuration options.
- **`/var/named`:** The default directory in which zone files are stored. This is specified by the "`directory`" option in the `/etc/named.conf` file.
- **`/etc/named.rfc1912.zones`:** The base configuration file for implementing a caching-only nameserver. This file contains five defined zones.
- **`/var/named/named.ca`:** A file that contains IP addresses of the 13 root authoritative DNS servers for the root domain.

The default installation of the `bind` package provides a caching-only nameserver. This nameserver is not authoritative for any domain. It stores only the results of queries in memory.



To configure a system as a DNS cache-only nameserver, perform the following steps (as the `root` user):

1. Install the `bind` package:

   ```
   # yum install bind
   ```

2. Add the following line to the beginning of the `/etc/resolv.conf` file. This line indicates use of the local system as the primary nameserver:

   ```
   nameserver 127.0.0.1
   ```

3. If NetworkManager is running, add the following line to the `/etc/sysconfig/network-scripts/ifcfg-<interface>` file:

   ```
   DNS1=127.0.1.1
   ```

4. Ensure that the network service is running:

   ```
   # systemctl start network
   ```

5. Enable and start the `named` service:

   ```
   # systemctl enable named
   # systemctl start named
   ```

Configuring an Authoritative Nameserver

- Define the zones in `/etc/named.conf`. Example:

```
zone "example.com" {
    type master;
    file "data/master-example.com";
    allow-update { key "rndckey"; };
    notify yes;
};
```

- This example specifies the `type` as `master` meaning the nameserver is authoritative for the `example.com` domain.

An authoritative nameserver responds to queries about names that are part of their zones. To configure an authoritative nameserver, edit the `/etc/named.conf` file and define the zone file characteristics. Multiple zone files can be defined.

The example in the slide specifies a zone file for the `example.com` domain. Zone options included in this example are:

- **`type`**: Specifies the `master` zone type, meaning the nameserver is authoritative for this zone, and the zone file resides on this system. Other valid zone types are `delegation-only`, `forward`, `hint`, and `slave`.
- **`file`**: Specifies "`data/master-example.com`" as the name of the zone file and its location. If the `directory` option is set to "`/var/named`", the absolute file name of the zone file is `/var/named/data/master-example.com`.
- **`allow-update`**: BIND uses a shared secret key algorithm to grant privileges to hosts. This example specifies the `rndc` (Remote Name Daemon Control) key, which ensures that a shared secret key exists on both primary and secondary nameservers before allowing a transfer.
- **`notify`**: Specifies whether to notify the secondary nameservers when a zone is updated

Zone Files

- Zone files:
  - Store information about domains in the DNS database
  - Contain directives (optional) and resource records
- Resource records contain (not all fields are required):
  - `Name`: The domain name or IP address
  - `TTL`: Time to live
  - `Class`: Always `IN` for Internet
  - `Type`: Record type
  - `Data`: Varies with record type
- Common types of resource records include:
  - `A`, `CNAME`, `MX`, `NS`, `PTR`, `SOA`

Information about domains in the DNS database is stored in zone files. A zone file consists of directives and resource records. Directives tell the nameserver to perform tasks or apply special settings to the zone. Resource records define the parameters of the zone and store host information. Directives are optional, but resource records are required.

A resource record has the following fields (some fields are optional, depending on the `Type`):

- **`Name`:** The domain name or IP address
- **`TTL`:** Time to live, maximum time a record is cached before checking for a newer one
- **`Class`:** Always `IN` for Internet
- **`Type`:** Record type
- **`Data`:** Varies with record type

More than 30 types of resource records exist. The more common ones are:

- **`A`:** IPv4 address
- **`CNAME`:** Canonical name or alias
- **`MX`:** Mail exchange, specifies the destination for mail addressed to the domain
- **`NS`:** Nameserver, specifies the system that provides DNS records for the domain
- **`PTR`:** Maps an IP address to a domain name for reverse name resolution
- **`SOA`:** Start of authority, designates the start of a zone

The following is an example of a zone file:

```
$TTL 86400      ; 1 day
example.com IN SOA dns.example.com. root@example.com. (
                   57         ; serial
                   28800      ; refresh (8 hours)
                   7200       ; retry (2 hours)
                   2419200    ; expire (4 weeks)
                   86400      ; minimum (1 day)
                   )
            IN NS dns.example.com.
dns         IN         A       192.0.2.1
example.com IN         A       192.0.2.1
host01      IN         A       192.0.2.101
host02      IN         A       192.0.2.102
host03      IN         A       192.0.2.103
```

The `$TTL` entry is a directive that defines the default time to live for all resource records in the zone. Each resource record can have a `TTL` value, which overrides this global directive.

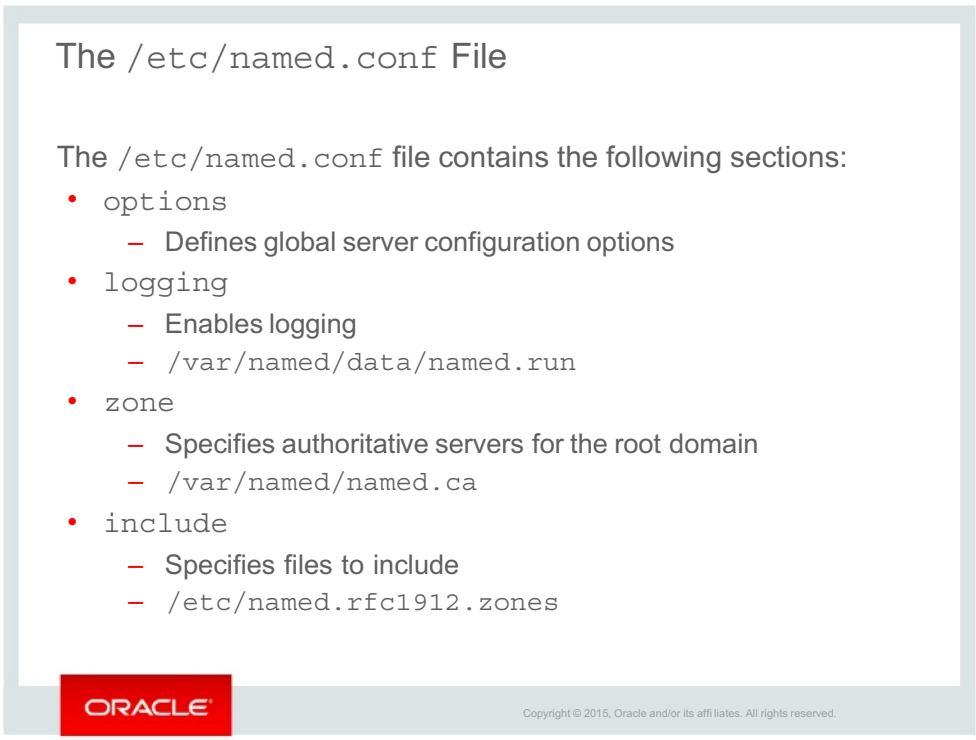
The next line in the example is the SOA record. All zone files must have one SOA record. The following information is included in the SOA record:

- **`example.com`:** The name of the domain
- **`dns.example.com.`:** The FQDN of the nameserver
- **`root@example.com.`:** The email address of the user who is responsible for the zone
- **`serial`:** A numerical value that is incremented each time the zone file is altered to indicate when it is time for the `named` service to reload the zone
- **`refresh`:** The elapsed time after which the primary nameserver notifies secondary nameservers to refresh their database
- **`retry`:** The time to wait after which a refresh fails before trying to refresh again
- **`expire`:** The time after which the zone is no longer authoritative and the root nameservers must be queried
- **`minimum`:** The amount of that time that other nameservers cache the zone's information

The `NS` (Nameserver) record announces authoritative nameservers for a particular zone by using the format: `IN NS dns.example.com.`

The `A` (Address) records specify the IP address to be assigned to a name by using the format:

```
hostname IN A IP-address
```



The default configuration of the `/etc/named.conf` file provides a caching-only nameserver. The file has four main sections described as follows.

`options`

The `options` statement defines global server configuration options and sets defaults for other statements. The following options are defined in the default `/etc/named.conf` file:

- **`listen-on`:** Instructs `named` to listen on port 53 on the local system for both IPv4 and IPv6 queries
- **`directory`:** Specifies the default working directory for the `named` service
- **`dump-file`:** Specifies the location where BIND dumps the database (cache) in the event of a crash
- **`statistics-file`:** Specifies the location to which data is written when the command `rndc stats` is issued
- **`memstatistics-file`:** Specifies the location to which BIND memory usage statistics are written
- **`allow-query`:** Specifies which IP addresses (`localhost` by default) are allowed to query the server
- **`recursion`:** Instructs the nameserver to perform recursive queries. Recursive queries cause a nameserver to query another nameserver if necessary to respond with an answer.

The options defined in the default `/etc/named.conf` file continue:

- **`dnssec-enable`:** Specifies that a secure DNS service is being used
- **`dnssec-validation`:** Instructs the nameserver to validate replies from DNSSEC-enabled (signed) zones
- **`dnssec-lookaside`:** Enables DNSSEC Lookaside Validation (DLV) by using `/etc/named.iscdlv.key`

`logging`

The `logging` statement turns on logging and causes messages to be written to the `data/named.run` file. The `severity` parameter controls the logging level. A `severity` value of `dynamic` means assume the global level defined by either the command-line parameter `-d` or by running the `rndc trace` command. The default logging statement follows:

```
logging {
      channel default_debug {
            file "data/named.run";
            severity dynamic;
};
```

`zone`

The default `zone` section specifies the initial set of root servers by using a `hint` zone, whose name is a period (`.`). This zone specifies that the nameserver must look in `/var/named/named.ca` for IP addresses of authoritative servers for the root domain when the nameserver starts or does not know which nameserver to query. The default zone section follows:

```
zone "." IN {
      type hint;
      file "named.ca";
};
```

Zone options include the following:

- **`type`:** Specifies the zone type, such as `master`, `delegation-only`, `forward`, `hint`, or `slave`. Type `master` designates the nameserver as authoritative for this zone. A zone is set as `master` if the zone file resides on this system.
- **`file`:** Specifies the name of the zone file, which is stored in the working directory defined by the `directory` option
- **`allow-update`:** Specifies which hosts are allowed to dynamically update information in their zone

`include`

The `include` statement allows files to be included. This can be done for readability, ease of maintenance, or so that potentially sensitive data can be placed in a separate file with restricted permissions. This `include` statement includes the `/etc/named.rfc1912.zones` file as though it were present in this file.

The /etc/named.rfc1912.zones File

This file specifies five predefined zones:

- `localhost.localdomain`
  - Specifies that `localhost.localdomain` points to `127.0.0.1`
- `localhost`
  - Sets up the normal server on the local system
- `1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa`
  - Sets up IPv6 reverse name resolution
- `1.0.0.127.in-addr.arpa`
  - Sets up IPv4 reverse name resolution
- `0.in-addr.arpa`
  - Specifies that IP addresses that start with 0 have their reverse lookup handled by the local server

The `/etc/named.rfc1912.zones` file is listed in the `include` section of the `/etc/named.conf` file. The `/etc/named.rfc1912.zones` file contains five zone sections.

Domains are grouped into zones and zones are configured through the use of zone files. The `zone` statement defines the characteristics of a zone, the location of its zone file, and zone-specific options, which override the global options statements. The following zones are defined in the `/etc/named.rfc1912.zones` file:

- **`localhost.localdomain`:** Specifies that `localhost.localdomain` points to `127.0.0.1`, preventing the local server from looking upstream for this information
- **`localhost`:** Sets up the normal server on the local system
- **`1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa`:** Sets up IPv6 reverse name resolution
- **`1.0.0.127.in-addr.arpa`:** Sets up IPv4 reverse name resolution
- **`0.in-addr.arpa`:** Specifies that IP addresses that start with 0 have their reverse lookup handled by the local server, preventing the local server from looking upstream

Reverse Name Resolution

- Normal, or forward, resolution returns an IP address when the domain name is known.
- Reverse name resolution returns the domain name when an IP address is known.
- DNS implements reverse name resolution by use of the following special domains:
  - `in-addr.arpa`: For IPv4
  - `ip6.arpa`: For IPv6
- Zone names reverse the network portion of the IP address and append the special domain name:
  - `2.0.192.in-addr.arpa`
- Resource records use type `PTR`.

DNS also provides reverse name resolution, which returns a domain name for a given IP address. DNS implements reverse name resolution by use of the following special domains:

- **`in-addr.arpa`:** For IPv4
- **`ip6.arpa`:** For IPv6

The zone characteristics are defined in `/etc/named.conf`, for example:

```
zone "2.0.192.in-addr.arpa" IN {
      type master;
      file "data/reverse-192.0.2";
      allow-update { key "rndckey"; };
      notify yes;
};
```

The zone name consists of `in-addr.arpa` preceded by the network portion of the IP address for the domain. In this example, the network is `192.0.2`, which in reverse is `2.0.192`.

Resource records in these domains have `Name` fields that contain IP addresses and `Type` fields of `PTR`.

The following is an example of the `2.0.192.in-addr.arpa` zone file:

```
$TTL 86400      ; 1 day
2.0.192.in-addr.arpa IN SOA dns.example.com. root@example.com. (
                    57         ; serial
                    28800      ; refresh (8 hours)
                    7200       ; retry (2 hours)
                    2419200    ; expire (4 weeks)
                    86400      ; minimum (1 day)
                    )
             IN NS dns.example.com.
1            IN        PTR       dns
1            IN        PTR       example.com
101          IN        PTR       host01
102          IN        PTR       host02
103          IN        PTR       host03
```

rndc Utility

- rndc is a command-line administration tool for named.
- Use the rndc key to prevent unauthorized access.
- The rndc key is generated by using the following command:

```
# rndc-confgen -a
```

- Configure named to use the key in /etc/named.conf.
- Type rndc to display usage of the utility and a list of available commands.

The rndc utility is a command-line tool to administer the named service, both locally and from a remote machine. To prevent unauthorized access to the service, rndc must be configured to listen on the selected port (port 953 by default), and an identical key must be used by both the service and the rndc utility. The rndc key is generated by using the following command:

```
# rndc-confgen -a
wrote key file "/etc/rndc.key"
```

This command creates the /etc/rndc.key file, which contains the key. To configure named to use the key, include the following entries in /etc/named.conf:

```
include "/etc/rndc.key";
controls {
      inet 127.0.0.1 allow { localhost; } keys { "rndckey"; }
};
```

The include statement allows files to be included so that potentially sensitive data can be placed in a separate file with restricted permissions. To ensure that only root can read the file, enter the following:

```
# chmod o-rwx /etc/rndc.key
```

The `controls` statement defines access information and the various security requirements necessary to use the `rndc` command.

- **`inet`**: The example allows you to control `rndc` from a console on the localhost (`127.0.0.1`).
- **`keys`**: Keys are used to authenticate various actions and are the primary access control method for remote administration. The example specifies using `rndckey`, which is defined in the `/etc/rndc.key` include file.

Type `rndc` to display usage of the utility and a list of available commands:

```
# rndc
Usage: rndc [-c config] [-s server] [-p port] [-key fey-file] [-y
key] [-V] command
Command is one of the following:
reload     Reload configuration file and zones
...
reconfig   Reload configuration file and new zones only.
stats      Write server statistics to the statistics file.
querylog   Toggle query logging.
dumpdb     Dump cache(s) to the dump file (named_dump.db)
stop       Save pending updates to master files and stop the
server.
halt       Stop the server without saving pending updates.
...
status     Display status of the server
...
```

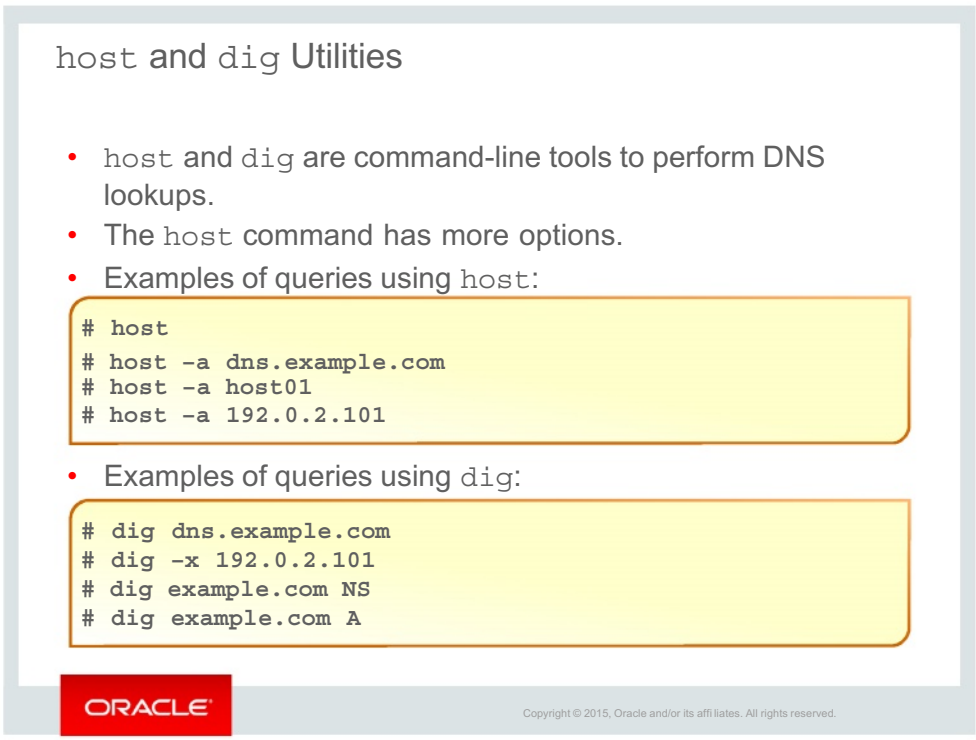
The following is an example of some of the `rndc` commands:

Use the `rndc status` command to check the current status of the `named` service:

```
# rndc status
number of zones: 3
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/1000
tcp clients: 0/100
server is up and running
```

Use the `rndc reload` command to reload both the configuration file and zones:

```
# rndc reload
server reload successful
```



The `host` and `dig` utilities are used for performing DNS lookups. The `host` utility returns the same information as `dig` and has more options. When no arguments are given, `host` displays a summary of its command-line arguments and options. Include the `a` option to `host` for more verbose output.

To look up the IP address for `host01`:

```
# host host01
# dig host01.example.com
```

To perform a reverse lookup, that is, to query DNS for the domain name that corresponds to an IP address:

```
# host 192.0.2.101
# dig -x 192.0.2.101
```

To query DNS for the IP address that corresponds to a domain:

```
# host dns.example.com
# dig dns.example.com
```

Quiz

Which of the following is the main configuration file for BIND?

a. `/var/bind.conf`
b. `/var/named.conf`
c. `/etc/named.conf`
d. `/etc/bind.conf`

Summary

In this lesson, you should have learned how to:

- Describe DHCP
- Configure a DHCP server
- Start and stop a DHCP server
- Configure a DHCP client and request a lease
- Describe DNS
- Describe nameserver types
- Describe BIND
- Configure a cache-only nameserver
- Describe and configure zone files
- Describe and configure reverse name resolution
- Use the `rndc` utility
- Use the `host` and `dig` utilities

Practice 2: Overview

The practices for this lesson cover the following:

- Configuring a DHCP server
- Configuring a DHCP client
- Viewing and Testing the DNS configuration
- Configuring a caching-only nameserver

3

Authentication and Directory Services

Objectives

After completing this lesson, you should be able to:

- Describe authentication options
- Describe the Authentication Configuration Tool
- Describe LDAP
- Describe OpenLDAP
- Describe OpenLDAP server and client utilities
- Configure LDAP authentication
- Configure Winbind authentication
- Configure Kerberos authentication
- Describe IPA Identity Management and Authentication Services
- Describe SSSD services and domains

Authentication Options

- Authentication is the verification of the identity of a user.
- Local account verification authenticates user information from local files:
  - `/etc/passwd`
  - `/etc/shadow`
- A local system can also access other directory services:
  - Network Information Service (NIS)
  - Lightweight Directory Access Protocol (LDAP)
  - Identity Policy Audit (IPA)
  - Winbind
- The Authentication Configuration GUI provides for the selection of user account databases and authentication configurations.

Authentication is the verification of the identity of a user. A user logs in by providing a username and a password and is authenticated by comparing this information to data stored on the system. If the login credentials match and the user account is active, the user is authenticated and can successfully access the system.

The information to verify the user can be located on the local system in the `/etc/passwd` and `/etc/shadow` files. A local system can also reference data stored on remote systems by using services such as Lightweight Directory Access Protocol (LDAP), Network Information Service (NIS), Identity Policy Audit (IPA), and Winbind. Additionally, LDAP, IPA, and NIS data files can use Kerberos authentication.

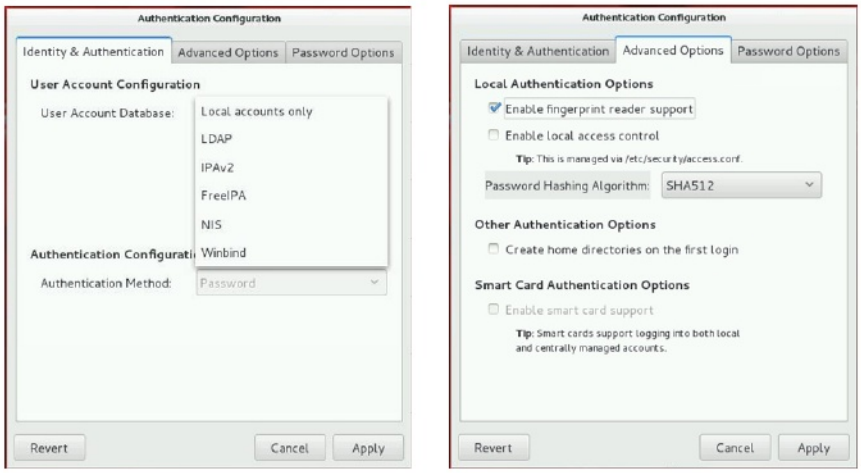
NIS simplifies the maintenance of common administration files such as `/etc/passwd` by keeping them in a central database and having clients retrieve information from this database server.

An LDAP directory can hold many types of information, including usernames, network services, and authentication data. Much like NIS, LDAP clients contact a centralized server to access this information.

Oracle Linux includes a GUI for selecting user databases and configuring associated authentication options—the Authentication Configuration GUI.

Authentication Configuration GUI

`system-config-authentication`



To use the Authentication Configuration GUI, enter the command:

```
# system-config-authentication
```

There are three tabs on the Authentication Configuration GUI: Identify & Authentication, Advanced Options, and Password Options. The slide shows the first two tabs.

Identity & Authentication

Click this tab to select how users should be authenticated. Under the User Account Configuration section of the page, select one of the six options for the User Account Database field:

- Local accounts only: Users and passwords are checked against local system accounts.
- LDAP
- IPAv2
- FreeIPA
- NIS
- Winbind

Additional User Account Configuration fields appear, depending on which user account database is selected. The Authentication Configuration section of the page also changes, depending on which user account database is selected.

Advanced Options

Click this tab to enable fingerprint reader support, enable local access control by using the `/etc/security/access.conf` file, configure smart card authentication options, change the password hashing algorithm, and configure other authentication options.

Password Options

Click this tab to configure minimal password requirements, which include the length of the password and the number of character classes. You can also configure the required character classes and the maximal consecutive character repetition values.

NIS Authentication

- NIS Domain:
  - A network of systems that share a common set of configuration files
- NIS Server:
  - A single system that stores the configuration files
- Authentication Method:
  - NIS password
  - Kerberos password

See the NIS Configuration Appendix for additional information.

NIS was among the first directory services, but it has largely been replaced by other technologies, such as LDAP. NIS stores administrative information such as usernames, passwords, and host names on a centralized server. Client systems on the network can access this common data. This allows users the freedom to move from machine to machine without having to remember different passwords and copy data from one machine to another. Storing administrative information centrally, and providing a means of accessing it from networked systems, also ensures the consistency of that data.

An NIS network of systems is called an NIS domain. Each system within the domain has the same NIS domain name, which is different from a DNS domain name. The DNS domain is used throughout the Internet to refer to a group of systems. An NIS domain is used to identify systems that use files on an NIS server. An NIS domain must have exactly one master server but can have multiple slave servers. When using the Authentication Configuration Tool and selecting NIS as the user account database, you are prompted to enter the NIS Domain and the NIS Server.

For Authentication Method, NIS allows simple NIS password authentication or Kerberos authentication. Kerberos is an authentication protocol that allows nodes communicating over a nonsecure network to prove their identity to one another in a secure manner.

See the appendix titled “NIS Configuration” for details on configuring NIS authentication.

Lightweight Directory Access Protocol (LDAP)

- LDAP is a protocol for accessing directory services.
- A directory is a hierarchical database.
- LDAP can also be used to authenticate users.
- An entry is the basic unit of information in a directory.
- Each entry has attributes.
- Required attributes are defined in a schema.
- Entries are uniquely identified and referenced by their distinguished name (DN).
- Example of a DN:
  - `uid=oracle,ou=People,dc=example,dc=com`
- LDIF is a plain-text representation of a DN.

Lightweight Directory Access Protocol (LDAP) is used to access centrally stored information over a network. LDAP servers store information in a database called a directory, which is optimized for searches. Directory entries are arranged in a hierarchical tree-like structure. This directory can store a variety of information such as names, addresses, phone numbers, network services, printers, and other types of data. LDAP can also be used to authenticate users, allowing users to access their accounts from any machine on the LDAP network.

An entry is the basic unit of information within an LDAP directory. Each entry has one or more attributes. Each attribute has a name, a type, or description, and one or more values. An example of a type would be `cn` for a common name, or `mail` for an email address. In addition, LDAP allows you to control which attributes are required and which are optional through the use of a special attribute called `objectClass`. The values of the `objectClass` attribute determine the schema rules that the entry must obey.

Each entry is uniquely identified and referenced by its distinguished name (DN). The DN is constructed by taking the name of the entry itself (called the "relative distinguished name" or RDN) and concatenating the names of its ancestor entries. For example, the DN for a user with an RDN of `uid=oracle` would be something like `uid=oracle,ou=People,dc=example,dc=com`. In this example, `ou` stands for "organizational unit" and `dc` stands for "domain component."

The following is an example of the information needed for the `oracle` user:

```
dn: uid=oracle,ou=People,dc=example,dc=com
uid: oracle
cn: Oracle Student
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0...
shadowLastChange: 15880
shadowMin: 0
shadowMax: 9999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/oracle
gecos: Oracle Student
```

The following is an example group:

```
dn: cn=students,ou=Group,dc=example,dc=com
objectClass: posixGroup

objectClass: top
cn: students
userPassword:: e2NyeXB0...
gidNumber: 1008
memberUid: oracle
memberUid: student1
memberUid: student2
```


LDAP Data Interchange Format (LDIF) is a plain-text representation of an LDAP entry. It takes the following form:

```
[id] dn: distinguished_name
attribute_type: attribute_value...
attribute_type: attribute_value...
```

The optional `id` number is determined by the application that is used to edit the entry. Each entry can contain as many *attribute_type* and *attribute_value* pairs as needed, as long as they are all defined in a corresponding schema file.

OpenLDAP

- OpenLDAP is an open-source implementation of LDAP.
- Packages include:
  - `openldap`: OpenLDAP libraries
  - `openldap-clients`: Client command-line utilities
  - `openldap-servers`: Server package; includes `slapd`
  - `nss-pam-ldapd`: Required for LDAP authentication
- OpenLDAP service is the stand-alone LDAP daemon, `slapd`.
- Use the `systemctl` utility to enable and start the service:

```
# systemctl enable slapd
# systemctl start slapd
```

Oracle Linux includes OpenLDAP, which is an open source implementation of the LDAP protocols. To begin configuring a system as an OpenLDAP server, install the following OpenLDAP packages:

- **`openldap`:** Contains the libraries necessary to run the OpenLDAP server and client applications
- **`openldap-clients`:** Contains the command-line utilities for viewing and modifying directories on an LDAP server
- **`openldap-servers`:** Contains the services and utilities to configure and run an LDAP server. This includes the stand-alone LDAP daemon, `slapd`.
- **`nss-pam-ldapd`:** Contains `nslcd`, a local LDAP name service that allows a user to perform local LDAP queries. This package is only required to authenticate by using OpenLDAP.

Additional OpenLDAP packages, not required for a standard configuration, are:

- **`compat-openldap`:** Includes older versions of the OpenLDAP-shared libraries that might be required by some applications
- **`bind-dyndb-ldap`:** A new LDAP driver for BIND9. It allows you to read data and also write data back (DNS Updates) to an LDAP backend.

OpenLDAP Server Directories

- Previous versions of OpenLDAP used a configuration file:
  - `/etc/openldap/slapd.conf`
- The current version of OpenLDAP uses a configuration database located in:
  - `/etc/openldap/slapd.d`
- The directory containing additional configuration files:
  - `/etc/openldap/slapd.d/cn=config`
- The directory containing the schema files:
  - `/etc/openldap/schema`
- The directory containing the database:
  - `/var/lib/ldap`

Previous versions of OpenLDAP used a configuration file:

`/etc/openldap/slapd.conf`

OpenLDAP now uses a configuration database located in the following directory:

`/etc/openldap/slapd.d`

The following list summarizes the OpenLDAP configuration that is stored in the `/etc/openldap` directory:

- **`/etc/openldap/ldap.conf`**: The configuration file for client applications
- **`/etc/openldap/slapd.d`**: The directory containing the `slapd` configuration
- **`/etc/openldap/schema`**: The directory containing the schema files

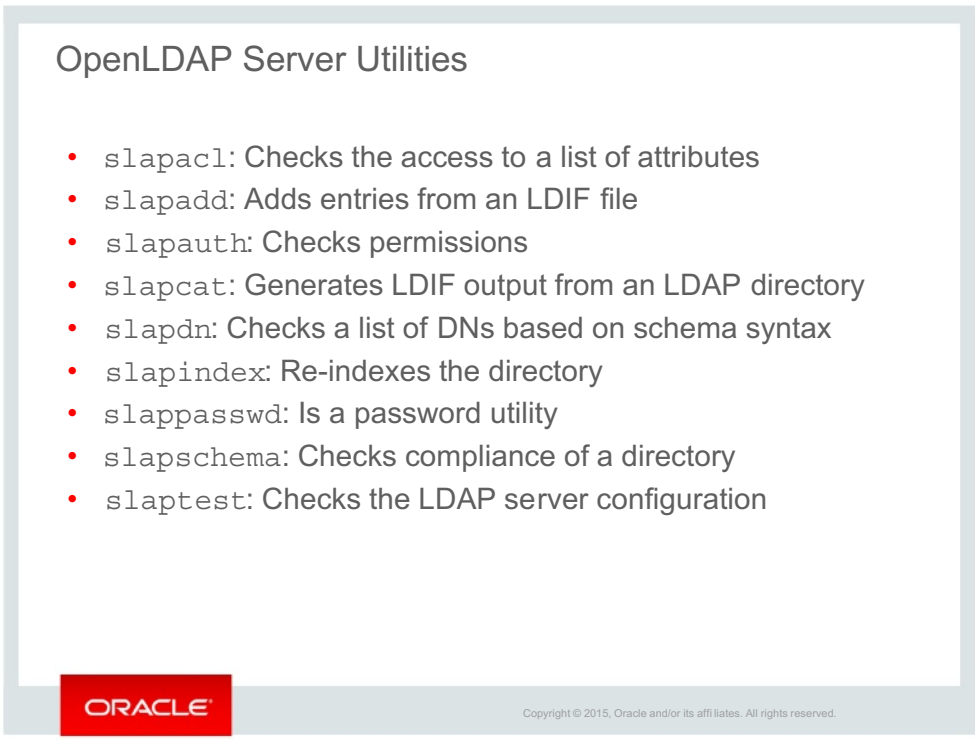
The schema used by OpenLDAP can be extended to support additional attribute types and object classes. This is described at:

`http://www.openldap.org/doc/admin24/schema.html`

OpenLDAP uses one of two varieties of the Berkeley DB storage format:

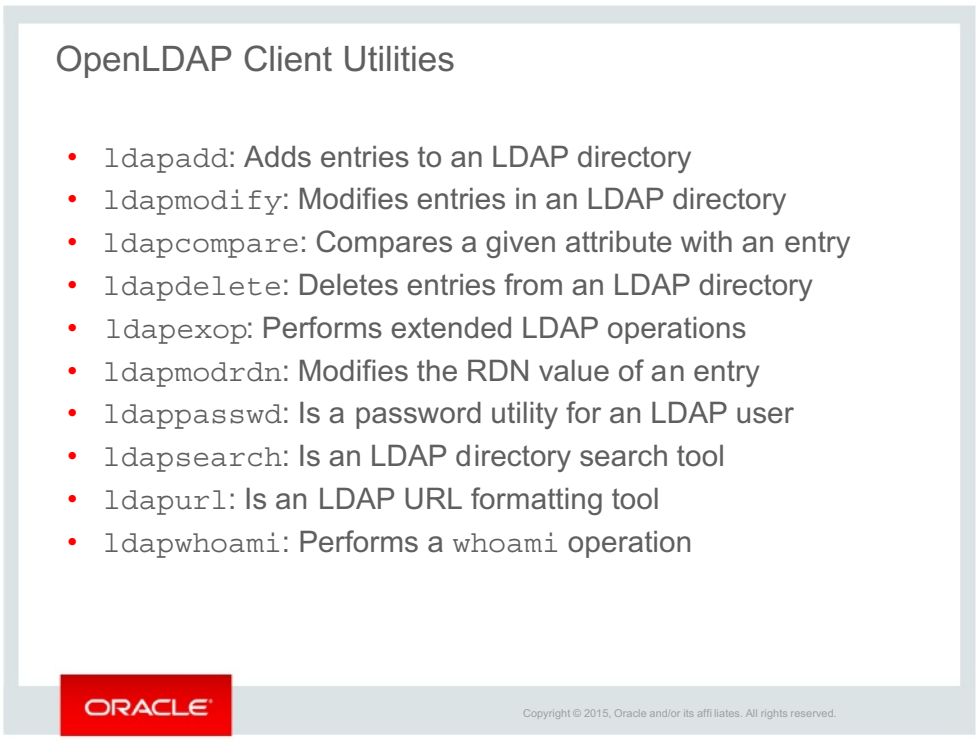
- **`bdb`**: The standard Berkeley DB format
- **`hdb`**: A newer version for hierarchical databases like LDAP

The database is stored in the `/var/lib/ldap` directory.



The `openldap-servers` package also includes the following utilities:

- **`slapacl`:** Checks the access to a list of attributes
- **`slapadd`:** Adds entries from an LDIF file to an LDAP directory
- **`slapauth`:** Checks a list of IDs for authentication and authorization permissions
- **`slapcat`:** Generates LDIF output from an LDAP directory
- **`slapdn`:** Checks a list of distinguished names (DNs) based on schema syntax
- **`slapindex`:** Re-indexes the directory. Run `slapindex` whenever indexing options are changed in the configuration file.
- **`slappasswd`:** Is a password utility for creating an encrypted user password
- **`slapschema`:** Checks compliance of a database with the corresponding schema
- **`slaptest`:** Checks the LDAP server configuration



The `openldap-clients` package installs the following utilities:

- **`ldapadd`:** Adds entries to an LDAP directory either from a file or from standard input. `ldapadd` is a symbolic link to `ldapmodify -a`.
- **`ldapmodify`:** Modifies entries in an LDAP directory
- **`ldapcompare`:** Compares a given attribute with an LDAP directory entry
- **`ldapdelete`:** Deletes entries from an LDAP directory
- **`ldapexop`:** Performs extended LDAP operations
- **`ldapmodrdn`:** Modifies the RDN value of an LDAP directory entry
- **`ldappasswd`:** Is a password utility for an LDAP user
- **`ldapsearch`:** Is an LDAP directory search tool
- **`ldapurl`:** Is an LDAP URL formatting tool
- **`ldapwhoami`:** Performs a `whoami` operation on an LDAP server

There are several LDAP client software applications that provide a graphical user interface (GUI) for maintaining LDAP directories, but none of them are included in Oracle Linux.

OpenLDAP Server Configuration

- Install the packages:

```
# yum install openldap-servers openldap-clients
  migrationtools
```

  - The `openldap-clients` package provides the `ldap` command-line utilities.
  - The `migrationtools` package is optional but allows you to migrate information from existing name services.
- Enable and start the `slapd` service.
  - This creates the database in the `/var/lib/ldap` directory.
- Update the files in the configuration directory:
  - `/etc/openldap/slapd.d/cn=config`
- Use the `ldapmodify` and `ldapadd` commands to update these files.

To configure an OpenLDAP server, install the following packages:

```
# yum install openldap-servers openldap-clients migrationtools
```

The `openldap-clients` package provides the LDAP command-line utilities used to update the configuration database. The `migrationtools` package is optional but it provides a set of Perl scripts, which allows you to migrate users, groups, and other information from existing name services.

Use the `systemctl` command to enable and start the `slapd` service.

```
# systemctl enable slapd
# systemctl start slapd
```

Use `ldapmodify` and `ldapadd` commands to update the files in the configuration directory:

```
# ls /etc/openldap/slapd.d/cn=config
-rw-------. 1 ldap ldap ... olcDatabase={0}config.ldif
-rw-------. 1 ldap ldap ... olcDatabase={-1}frontend.ldif
-rw-------. 1 ldap ldap ... olcDatabase={1}monitor.ldif
-rw-------. 1 ldap ldap ... olcDatabase={2}hdb.ldif
```

The ldapmodify Utility

- The following example uses the ldapmodify utility to set the olcSuffix directive in the olcDatabase={2}hdb.ldif file to "dc=example,dc=com":

```
# ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
> dn: olcDatabase={2}hdb,cn=config
>
> changetype: modify
> replace: olcSuffix
> olcSuffix: dc=example,dc=com
>
> EOF
```

- After issuing the ldapmodify command, the prompt changes to >.
- The command terminates after entering the final "EOF".

The example in the slide uses the ldapmodify command to update the olcSuffix parameter in the olcDatabase={2}hdb.ldif file. The example sets olcSuffix to "dc=example,dc=com".

Use the ldapmodify utility to update the OpenLDAP domain component in the configuration database. The default setting after an initial installation is:

```
dc=my-domain,dc=com
```

You can use the grep command and search for occurrences of "my-domain" in the configuration database directory:

```
# grep my-domain /etc/openldap/slapd.d/cn=config/*
olcDatabase={1}monitor.ldif: ,cn=auth" read by
dn.base="cn=Manager,dc=my-domain,dc=com" read by * none
olcDatabase={2}hdb.ldif:olcSuffix: dc=my-domain,dc=com
olcDatabase={2}hdb.ldif:olcRootDN: cn=Manager,dc=my-
domain,dc=com
```

In this example, "my-domain" needs to be changed to your company's DN in two files:

- **olcDatabase={1}monitor.ldif**
- **olcDatabase={2}hdb.ldif**

The slappasswd Utility

- Use the slappasswd command to create an encrypted user password. Example:

```
# slappasswd
New password: <password>
Re-enter new password: <password>
{SSHA}4SOiIaqwQYftwkdr1FbqVNEmI3Am0wJT
```

- Use the ldapmodify command to add the olcRootPW directive to the olcDatabase={2}hdb.ldif file.
  - Set the value of olcRootPW to the encrypted result of the slappasswd command.
- Subsequent modifications to the OpenLDAP directory require the password. Example:

```
# ldapadd -x -W -D "cn=Manager,dc=example,dc=com" -f
   base.ldif
Enter LDAP Password:
```

Use the slappasswd command to create an encrypted user password. The encrypted password shown is a sample only.

```
# slappasswd
New password: <password>
Re-enter new password: <password>
{SSHA}4SOiIaqwQYftwkdr1FbqVNEmI3Am0wJT
```

Use the ldapmodify command to add the olcRootPW directive to the olcDatabase={2}hdb.ldif file. Set the value of olcRootPW to the encrypted result of the slappasswd command:

```
# ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
> dn: olcDatabase={2}hdb,cn=config
> changetype: modify
>
> add: olcRootPW
> olcRootPW: {SSHA}4SOiIaqwQYftwkdr1FbqVNEmI3Am0wJT
>
> EOF
```

Loading the Standard Schemas

- The attributes of each data object in an OpenLDAP directory are defined in a schema.
  - The schema must be loaded into the configuration database before the object they define can be used.
  - The standard schema are provided as LDIF files in the `/etc/openldap/schema` directory.
- The following four schemas provide the objects and attributes of a typical organization:
  - `core`, `cosine`, `inetorgperson`, `nis`
- Use the `ldapadd` command to load the schema. Example:

```
# ldapadd -Q -Y EXTERNAL -H ldapi:/// -f
   /etc/openldap/schema/core.ldif
```

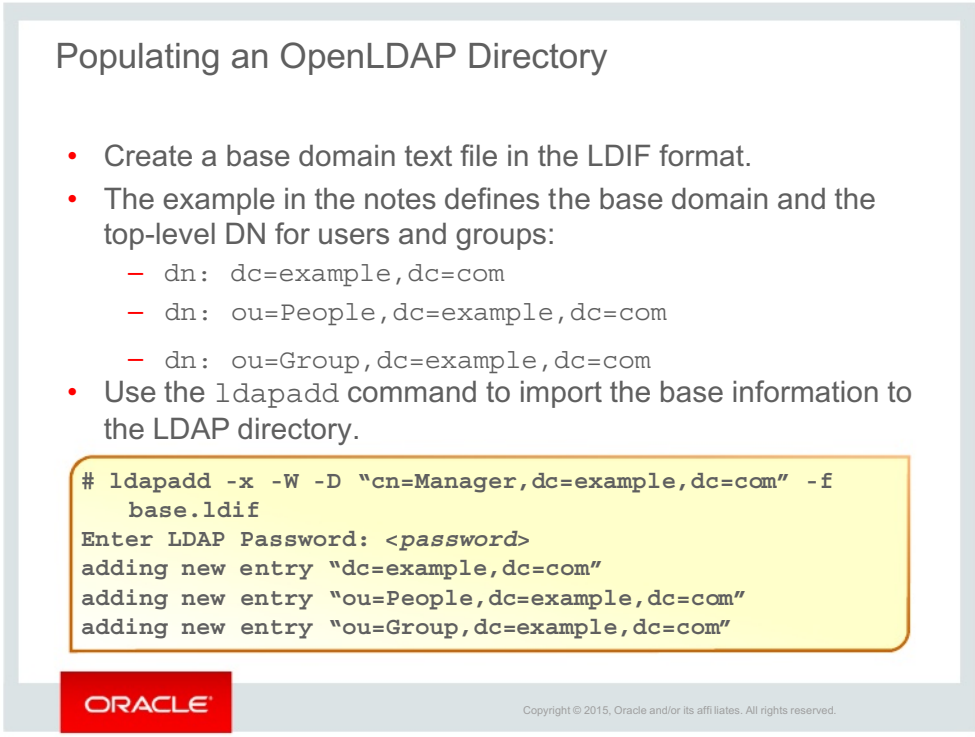
The attributes of each data object in an OpenLDAP directory are defined in a schema. The schema must be loaded into the configuration database before the object they define can be used. The standard schema are provided as LDIF files in the `/etc/openldap/schema` directory.

```
# ls /etc/openldap/schema
collective.ldif     cosine.schema   java.ldif     openldap.schema
collective.schema   duaconf.ldif    java.schema   pmi.ldif
...
```

The following four schemas define the basic objects and attributes needed to describe a typical organization: `core`, `cosine`, `inetorgperson`, `nis`.

Use the `ldappadd` utility to load the schemas.

```
# ldapadd -Q -Y EXTERNAL -H ldapi:/// -f
/etc/openldap/schema/core.ldif
adding new entry "cn=core,cn=schema,cn=config"
# ldapadd -Q -Y EXTERNAL -H ldapi:/// -f
/etc/openldap/schema/cosine.ldif
adding new entry "cn=cosine,cn=schema,cn=config"
```



You can create a text file containing base information for the LDAP directory and then use the `ldapadd` command to import the information into the directory. Create the base information in the LDIF format. The following example contains top-level information for users and groups:

```
dn: dc=example,dc=com
dc: example
objectClass: top
objectClass: domain
dn: ou=People,dc=example,dc=com
ou: People
objectClass: top
objectClass: organizationalUnit
dn: ou=Group,dc=example,dc=com
ou: Group
objectClass: top
objectClass: organizationalUnit
```

Using the migrationtools Utilities

- The migrationtools Perl scripts allow you to migrate information from existing name services.
- The migrationtools files are installed in the /usr/share/migrationtools directory.
- Update the migrate_common.ph file for the correct domain.

```
# vi /usr/share/migrationtools/migrate_common.ph
$DEFAULT_MAIL_DOMAIN = "example.com";
$DEFAULT_BASE = "dc=example,dc=com";
```

- Extract existing information into text files.

```
# grep ":100[0-9]" /etc/passwd > passwd"
# grep ":100[0-9]" /etc/group > group
```

The migrationtools Perl scripts allow you to migrate users, groups, and other information from existing name services. The migrationtools files are installed in the /usr/share/migrationtools directory.

You must first update the migrate_common.ph file for the correct domain. The following example sets two directives to the "example.com" domain:

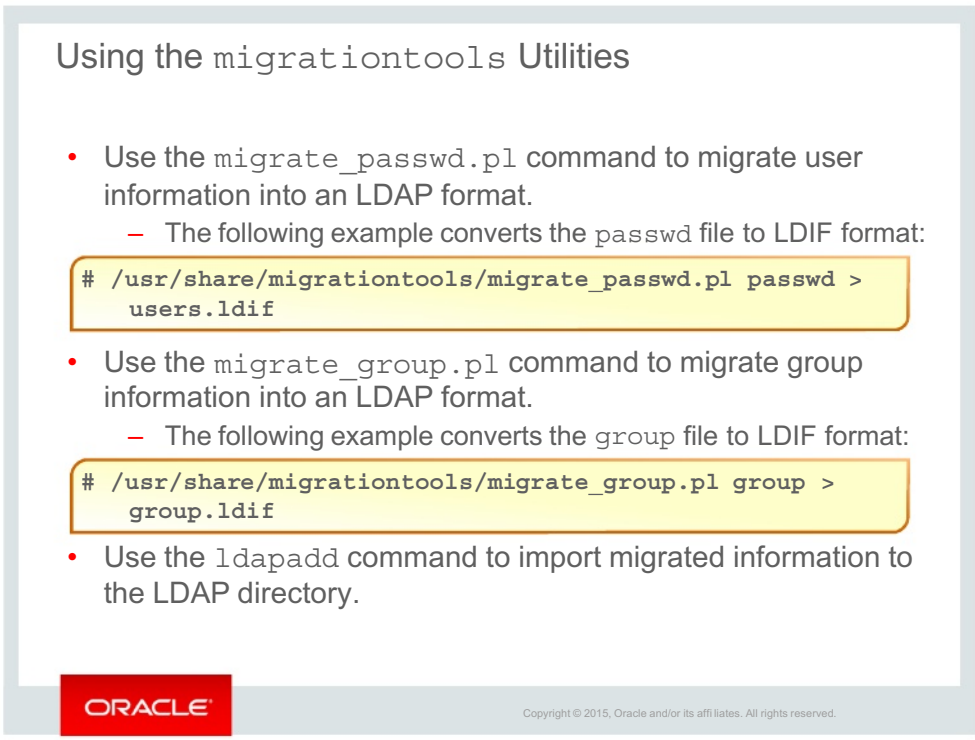
```
# vi /usr/share/migrationtools/migrate_common.ph
$DEFAULT_MAIL_DOMAIN = "example.com";
$DEFAULT_BASE = "dc=example,dc=com";
```

You can use the grep command to extract existing users and groups into text files. The following example extracts users from the /etc/passwd file with UID in the 1000-1009 range and writes the output to the passwd file:

```
# grep ":100[0-9]" /etc/passwd > passwd
```

The following example extracts groups with GID in the 1000-1009 range and writes the output to the group file:

```
# grep ":100[0-9]" /etc/group > group
```



Run the `migrate_passwd.pl` command to migrate user information in the `passwd` file into an LDAP format. This example redirects the output to `users.ldif`:

```
# /usr/share/migrationtools/migrate_passwd.pl passwd >
users.ldif
```

Run the `migrate_group.pl` command to migrate group information in the `group` file into an LDAP format. This example redirects the output to `group.ldif`:

```
# /usr/share/migrationtools/migrate_group.pl group > group.ldif
```

You can then use the `ldapadd` command to import the user and group information to the LDAP directory:

```
# ldapadd -x -W -D "cn=Manager,dc=example,dc=com" -f users.ldif
Enter LDAP Password: <password>
adding new entry "uid=oracle,ou=People,dc=example,dc=com"

...
# ldapadd -x -W -D "cn=Manager,dc=example,dc=com" -f group.ldif
Enter LDAP Password: <password>
adding new entry "cn=oracle,ou=Group,dc=example,dc=com"
...
```

Configuring LDAP Authentication

- LDAP Search Base DN
  - The distinguished name (DN), or root suffix, for the user directory
- LDAP Server
  - The URL of the LDAP server
- Authentication Method
  - LDAP password
  - Kerberos password

To configure LDAP authentication from the Authentication Configuration Tool on the OpenLDAP server, select LDAP as the user account database. You are then prompted to enter:

- LDAP Search Base DN
- LDAP Server

For LDAP Search Base DN, enter the DN, or the root suffix, for the user directory. The LDAP directory is hierarchical, so all the user entries used for identity and authentication exist below this parent entry. An example of a base DN would be `dc=example,dc=com`.

For LDAP Server, enter the URL of the LDAP server and optionally include the port number. Examples of this would be:

- **`ldap://host03.example.com:389`**
- **`ldaps://host03.example.com`**

You also have the option to “Use TLS to encrypt connections.” TLS stands for Transport Layer Security. It provides a secure connection by encrypting the connections to the LDAP server. Selecting TLS enables the “Download CA Certificate” button. CA stands for certificate authority or certification authority and is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key. Clicking the “Download CA Certificate” button prompts you to enter the URL from which to download the CA certificate.

Do not select “Use TLS to encrypt connections” if the server URL uses a secure protocol (`ldaps`).

For Authentication Method, select either one of the following:

- LDAP password
- Kerberos password

The LDAP password option uses Pluggable Authentication Modules (PAM) applications for LDAP authentication. This option requires either a secure URL or the use of TLS to connect to the LDAP server.

You can also enable and configure LDAP from the command line by using the `authconfig` command. To use an LDAP identity data store, use the `--enableldap` flag. To use LDAP as the authentication source, use the `--enableldapauth` flag. Include the LDAP server name, the base DN for the user suffix, and TLS information if used. Use the full LDAP URL, including the protocol (`ldap` or `ldaps`) and the port number.

The following is an example:

```
# authconfig --enableldap --enableldapauth
--ldapserver=ldap://host03.example.com:389
--ldapbasedn="dc=example,dc=com" --enableldaptls
--ldaploadcacert=https://ca.server.example.com/caCert.crt --update
```

Basic configuration of the OpenLDAP server is now complete.

OpenLDAP uses port `389` to communicate over the network. Ensure that the firewall is configured (or disabled) to allow access to this port. When using `firewalld`, add the `ldap` service. Example:

```
# firewall-cmd --permanent --zone=public --add-service=ldap
```

If you configure LDAP over Secure Sockets Layer (SSL), the port number is `636`. When using `firewalld`, add the `ldaps` service.

```
# firewall-cmd --permanent --zone=public --add-service=ldaps
```

Configuring User Authentication from an OpenLDAP Client

- Install the following packages on the client:

```
# yum install openldap-clients nss-pam-ldapd
```

- Edit the `/etc/openldap/ldap.conf` file on the client to point to the LDAP server:

```
BASE        dc=example,dc=com
URI         ldap://192.0.2.103/
```

  - Also update `/etc/nslcd.conf`
- Set `USELDAP=yes` in `/etc/sysconfig/authconfig`.
- Enable the `pam_ldap` module in the `/etc/pam.d/system-auth` file.
- Add `ldap` to the `passwd`, `shadow`, and `group` directives in the `/etc/nsswitch.conf` file.
- Start the `nslcd` service.

OpenLDAP clients can search a remote server for information. Users on an OpenLDAP client can also be authenticated over the network by the server. This allows all user accounts to exist only in the LDAP directory. Users can then log in from clients and be authenticated by the server. Install the following packages to configure an OpenLDAP client:

```
# yum install openldap-clients nss-pam-ldapd
```

Edit the `/etc/openldap/ldap.conf` file on the client to point to the LDAP server. Example:

```
BASE        dc=example,dc=com
URI         ldap://192.0.2.103/
```

You also need to configure the `/etc/nslcd.conf` file to point to the LDAP server. This is the configuration file for the Naming Services LDAP Client Daemon. Example (both directives are in lowercase in these files):

```
uri         ldap://192.0.2.103/
base        dc=example,dc=com
```

Edit the `/etc/sysconfig/authconfig` file and set `USELDAP=yes`:

```
USELDAP=yes
```

The pam_ldap module allows OpenLDAP clients to authenticate against LDAP directories, and to change their passwords in the directory. Pluggable Authentication Modules (PAM) is covered in a subsequent lesson in this course.

Changes are needed to the /etc/pam.d/system-auth file to enable this PAM module. Items in bold are added to this file.

```
# vi /etc/pam.d/system-auth
...
auth        requisite     pam_succeed_if.so uid >= 1000 quiet
auth        sufficient    pam_ldap.so use_first_pass
...
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     [default=bad success=ok user_unknown=ignore]
pam_ldap.so
...
password    sufficient    pam_unix.so sha512 shadow nullok
try_first_pass use_authtok
password    sufficient    pam_ldap.so use_authtok
...
session     required      pam_unix.so
session     optional      pam_ldap.so
session     optional      pam_mkhomedir.so skel=/etc/skel
umask=077
```

Add ldap to the passwd, shadow, and group directives in the /etc/nsswitch.conf file:

```
# vi /etc/nsswitch.conf
passwd:   files ldap
shadow:   files ldap
group:    files ldap
```

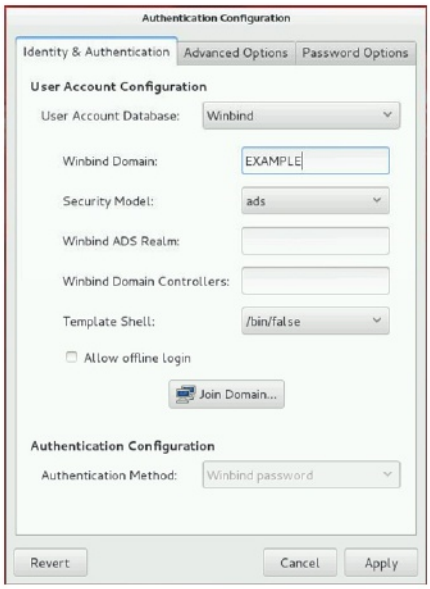
Use the systemctl command to start the nslcd service:

```
# systemctl start nslcd
```

You can now log in to the OpenLDAP client as a user, which has an account only on the LDAP server. The new "session" entry in the /etc/pam.d/system-auth file, which loads the pam_mkhomedir module, creates the user's home directory on the client if needed.

Configuring Winbind Authentication

- Winbind Domain
- Security Model
  - ads, domain, server, user
- Winbind ADS Realm
  - ads only
- Winbind Domain Controllers
  - ads, domain only
- Template Shell
  - ads, domain only



To configure Winbind authentication, install the `samba-winbind` package:

```
# yum install samba-winbind
```

This package includes the `winbindd` daemon, controlled by the `winbind` service. `winbind` is a client-side service used to connect to Windows servers. It resolves user and group information on a Windows server, allowing Linux to understand Windows users and groups.

Select Winbind as the user account database in the Authentication Configuration Tool. You are then prompted for information required to connect to a Microsoft workgroup, Active Directory, or Windows NT domain controller.

Select the security model to use for Samba clients. The security model options are:

- **`ads`**: This configures Samba to act as a domain member in an Active Directory Server realm. To use `ads`, you need to install the `krb5-server` package and configure Kerberos.
- **`domain`**: Samba validates the user by authenticating through a Windows domain controller.
- **`server`**: A local Samba server validates the user by authenticating it through another server.
- **`user`**: This requires a client to log in with a valid username and password.

To complete Winbind authentication configuration, provide the following information:

- **Winbind ADS Realm:** The Active Directory realm that the Samba server joins. This is required only when using the `ads` security model.
- **Winbind Domain Controllers:** The domain controller to use
- **Template Shell:** The login shell to use for Windows NT user account settings
- **Allow offline login:** Allows user authentication while the system is offline. Authentication information is stored in a local cache provided by System Security Services Daemon (SSSD).

Winbind authentication can also be configured from the command line by using the `authconfig` command. For `user` and `server` security models, only the domain (or workgroup) name and the domain controller host names are required:

```
# authconfig --enablewinbind --enablewinbindauth --smbsecurity
user|server --enablewinbindoffline --smbservers=ad.example.com
--smbworkgroup=EXAMPLE --update
```

For `ads` and `domain` security models, specify using the `--smbsecurity` flag and append the template shell and realm (`ads` only) flags to the previous example:

```
--smbrealm EXAMPLE.COM --winbindtemplateshell=/bin/sh -update
```

Winbind Security Model Options

- User Security Model
  - This model requires a client to log in with a valid username and password.
  - The client can mount multiple shares without specifying a separate username and password for each instance.
- Server Security Model
  - This model presents numerous security issues.
- Domain Security Model
  - The Samba server has a domain security trust account.
  - Samba authenticates username and password through a domain controller.
- Activity Directory Server (ADS) Security Model
  - Samba acts as a domain member in an ADS realm.

User Security Model

User-level security is the default setting for Samba. It requires a client to log in with a valid username and password. This mode does support encrypted passwords. If the server accepts the client's username and password, the client can then mount multiple shares without specifying a password for each instance.

Server Security Model

In this model, a local Samba server validates the username and password by authenticating it through another server, such as a Windows NT server. Do not use this model, because it presents numerous security issues. It was designed before Samba could act as a domain member server.

Domain Security Model

In this model, the Samba server has a machine account (domain security trust account) and Samba validates the username and password by authenticating it through a domain controller.

Activity Directory Server (ADS) Security Model

In this model, Samba is configured to act as a domain member in an ADS realm. Windows requires Kerberos tickets for Active Directory authentication. In addition, a machine account must be created on the Active Directory domain server.

Install the `krb5-server` package and configure Kerberos before attempting to join a domain.

```
# yum install krb5-server
```

Use the `kinit` command to create an administrative Kerberos ticket as follows:

```
# kinit administrator@EXAMPLE.COM
```

The `kinit` command references the Active Directory administrator account and Kerberos realm. It then obtains and caches a Kerberos ticket, which is required for authentication.

Assuming that Kerberos has been initialized, click the Join Domain button to attempt to join the domain immediately. Alternatively, use the following command to join:

```
# net ads join -S <active_directory_server> -U
administrator%password
```

This command creates the appropriate machine account on the Active Directory server and grants permissions to the Samba domain member server to join the domain.

Configuring Kerberos Authentication

- Realm
  - The realm for the Kerberos server
- KDCs
  - The servers that issue Kerberos tickets
- Admin Servers
  - The servers running the `kadmind` process in the realm

LDAP, NIS, and IPA authentication support Kerberos authentication. Kerberos provides a secure connection over standard ports. It also uses credentials caching with SSSD, which allows offline logins.

Install the following packages required for Kerberos authentication:

```
# yum install krb5-libs krb5-workstation
```

Select Kerberos password as the Authentication Method. You are then prompted for information required to connect to the Kerberos realm:

- **Realm:** The realm for the Kerberos server is the network that uses Kerberos. It is composed of Key Distribution Centers (KDCs) and client systems.
- **KDCs:** A comma-separated list of servers that issue Kerberos tickets
- **Admin Servers:** A comma-separated list of administration servers that run the `kadmind` process in the realm

You can also select the check boxes to use DNS to resolve server host name and to find additional KDCs within the realm.

You can also use the `authconfig` command to configure Kerberos authentication.

IPA Identity Management and Authentication Services

- There are two versions of Identity Policy Audit (IPA) that you can choose from the Authentication Configuration GUI:
  - FreeIPA (effectively IPAv1) and IPAv2
- These are open source projects that combine the capabilities of Linux, LDAP (389 Directory Server), Kerberos, NTP, DNS, SSSD, and certificate authority functionality.
- You can use them to:
  - Enable single sign-on authentication for all your systems, services, and applications
  - Define Kerberos authentication and authorization policies for your identities
  - Control services like DNS, sudo, SELinux, and autofs
  - Integrate with Microsoft Active Directory
- See https://www.freeipa.org for more information.

There are two versions of IPA that you can choose from the Authentication Configuration GUI: FreeIPA (effectively IPAv1) and IPAv2. These are open source projects that provide centralized identity management and authentication services. They combine the capabilities of Linux, LDAP (389 Directory Server), Kerberos, NTP, DNS, SSSD, and certificate authority functionality.

FreeIPA and IPAv2 allow you to enable single sign-on authentication for all your systems, services, and applications. They allow you to define Kerberos authentication and authorization policies for your identities. You can control services like DNS, sudo, SELinux, and autofs. You can also integrate FreeIPA and IPAv2 with other Identity Management systems like Microsoft Active Directory.

FreeIPA supports identity management and authentication of users and groups, and does not require you to join your system to an IPA realm. When you select FreeIPA as the user account database, you are prompted to enter information about LDAP and Kerberos configuration.

IPAv2 supports identity management and authentication of machines and requires you to join your system to an IPA realm. You are prompted to enter information about the IPA domain configuration, optionally choose to configure NTP, and click Join Domain to create a machine account on the IPA server. After your system has obtained permission to join the IPA realm, you can select and configure the authentication method.

Configuring Advanced Options

The slide shows the Advanced Options tab from the Authentication Configuration GUI. On this tab, you can configure local authentication options that define authentication behavior on the local system. You can also configure your system to automatically create home directories the first time that a user logs in, and you can enable smart card authentication. Each of these configuration options is discussed.

Enable Fingerprint Reader Support

Assuming that the appropriate hardware is in place, this allows fingerprint scans to be used to authenticate local users rather than using other credentials. Use the following command to enable fingerprint reader support, from the command line:

```
# authconfig --enablefingerprint --update
```

Enable Local Access Control

This checks the `/etc/security/access.conf` file for local user authorization rules. This file specifies combinations for logins that are accepted or refused. The syntax of the entries is:

```
permission : users : srcin
```

A plus sign (`+`) in the `permission` field means that the login is granted. Login is denied if the field contains a minus sign (`-`). The `users` field can be a username, group, or the `ALL` keyword. The `srcin` field is a host name, network, TTY (terminal), or the `ALL` or `NONE` keywords.

Password Hashing Algorithm

This sets the hashing algorithm to use to encrypt locally stored passwords. The options are:

- DESCRYPT
- BIGCRYPT
- MD5
- SHA256
- SHA512

You can also look at the password field in `/etc/shadow` to determine the algorithm. The field starts with a specific set of characters, depending on the hashing algorithm used, for example:

- MD5 starts with `$1$`
- SHA-256 starts with `$5$`
- SHA-512 starts with `$6$`

To determine the current algorithm from the command line:

```
# authconfig --test |grep hashing
```

You can also change the hashing algorithm from the command line. The following example changes it to SHA512:

```
# authconfig --passalgo=sha512 --update
```

Other Authentication Options

To enable the creation of user home directories at the first login, from the command line:

```
# authconfig --enablemkhomedir --update
```

Smart Card Authentication Options

A system can accept smart cards (or tokens) to authenticate users. The appropriate hardware must be available and the following package must be installed:

```
# yum install pam_pkcs11
```

Enabling smart card support prompts for additional configuration information:

- **Require smart card for login:** This disables Kerberos password authentication and all other methods of authentication for logging in to the system.
- **Card removal action:** Sets the system's response to a smart card being removed during an active session. Options are `Ignore`, meaning that the system continues functioning, and `Lock`, which immediately locks the screen.

To enable smart card use, from the command line:

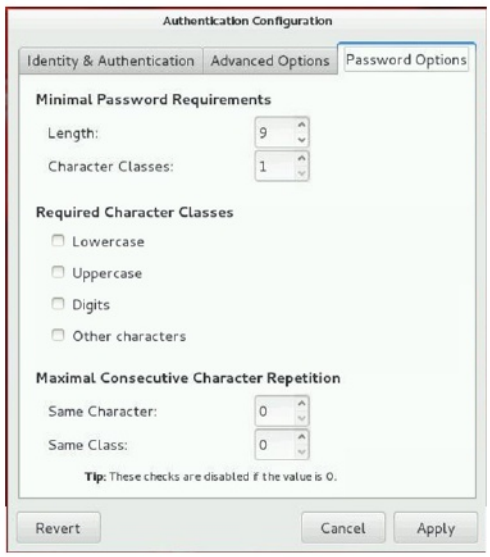
```
# authconfig --enablesmartcard --update
```

To enable smart cards and lock the system when the smart card is removed:

```
# authconfig --enablesmartcard --smartcardaction=0 --update
```

Setting `--smartcardaction=1` does not lock the system when the smart card is removed.

Configuring Password Options



The slide shows the Password Options tab from the Authentication Configuration GUI. From this tab, you can configure password complexity, which is a combination of length and variation of character classes. You can specify what types of characters can be used in a password and how those characters can be used within the password.

Minimal Password Requirements

Use the “Length” field to specify the minimum length of the password. Use the “Character Classes” field to specify the minimum number of character classes, which must be used in the password.

Required Character Classes

In this section, you can enable character classes, which must be used for passwords. For example, if the “Digits” check box is selected, a digit must be used in every password. All character types are allowed but the selection of a character class means the character class is required.

Maximal Consecutive Character Repetition

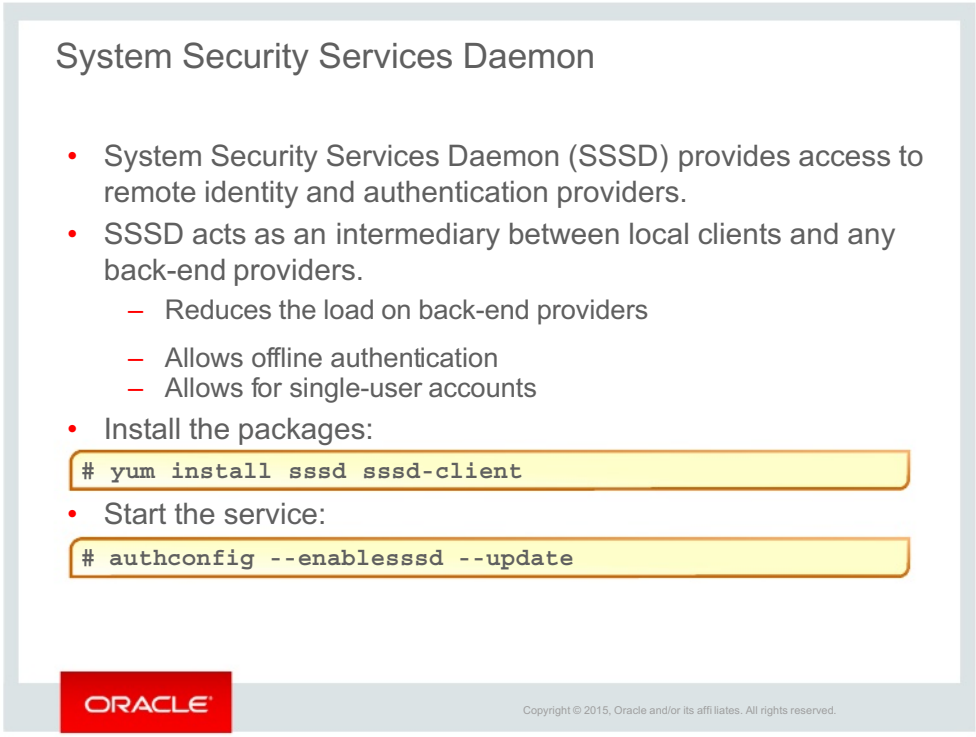
Set the number of times that a character or character class can be repeated consecutively. A zero setting means there is no repeat limit. The “Same Character” field sets how often a single character can be repeated. The “Same Class” field sets how many times any character from a character class can be repeated.

Password complexity can also be configured from the command line by using the `authconfig` command with the following options:

- **`--passminlen`**: The minimum length of a password
- **`--passminclass`**: The minimum number of different types of characters that must be used in a password
- **`--passmaxrepeat`**: The number of times a character can be repeated consecutively
- **`--passmaxclassrepeat`**: The number of times the same character can be used
- **`--enablerequpper`**: The password requires uppercase letters
- **`--enablereqlower`**: The password requires lowercase letters
- **`--enablereqdigit`**: The password requires numbers
- **`--enablereqother`**: The password requires special characters

The following example sets the minimum length of the password to eight characters, requires three different types of character classes, does not allow characters of classes to be repeated more than two times, and requires both numbers and special characters.

```
# authconfig --passminlen=8 --passminclass=3 --passmaxrepeat=2
--passmaxclassrepeat=2 --enablereqdigit --enablereqother
--update
```



The System Security Services Daemon (SSSD) provides access to remote identity and authentication providers. Providers are configured as back ends with SSSD acting as an intermediary between local clients and any configured back-end provider. The local clients connect to SSSD and then SSSD contacts the providers. Benefits of SSSD include:

- **Reduced load:** Clients do not have to contact the identification/authentication servers directly; they need to contact only SSSD.
- **Offline authentication:** SSSD can, optionally, keep a cache of user identities and credentials, allowing users to authenticate offline.
- **Single-user accounts:** SSSD maintains network credentials, allowing users to connect to network resources by authenticating with their local username on their local machine.

Install the following SSSD packages:

```
# yum install sssd sssd-client
```

To cause SSSD to start when the system boots, enter either of the following:

```
# systemctl enable sssd
# authconfig --enablesssd --update
```

Configuring SSSD Services

- The main configuration file is `/etc/sssd/sssd.conf`.
- SSSD services are configured in separate sections of this file.
- `[sssd]` section specifies:
  - Specialized services that run together with SSSD
  - Identity domains
- `[nss]` section:
  - Configuration parameters for NSS service
- `[pam]` section:
  - Configuration parameters for PAM service

The main configuration file for SSSD is `/etc/sssd/sssd.conf`. SSSD services and domains are configured in separate sections of this file, each beginning with a name of the section in square brackets. The following are examples:

```
[sssd]
[nss]
[pam]
```

`[sssd]` Section

SSSD functionality is provided by specialized services that run together with SSSD. These specialized services are started and restarted by a special service called "monitor." Monitor options and identity domains are configured in the `[sssd]` section of `/etc/sssd/sssd.conf`. The following is an example:

```
[sssd]
domains = LDAP
services = nss, pam
```

The `domains` directive can define multiple domains. Enter them in the order in which you want them to be queried. The `services` directive lists the services that are started when `sssd` itself starts.

Services Sections

Each of the specialized services that run together with SSSD is configured in separate sections in `/etc/sssd/sssd.conf`. For example, the `[nss]` section is used to configure the Name Service Switch (NSS) service. The `[pam]` section is used to configure the PAM service.

Configuring the NSS Service

Included in the `sssd` package is an NSS module, `sssd_nss`, which instructs the system to use SSSD to retrieve user information. This is configured in the `[nss]` section of `/etc/sssd/sssd.conf`. The following is an example that includes only a partial list of configurable directives:

```
[nss]
filter_groups = root
filter_users = root
reconnection_retries = 3
entry_cache_timeout = 300
```

The `filter_users` and `filter_groups` directives tell SSSD to exclude certain users and groups from being fetched from the NSS database. The `reconnection_retries` directive specifies the number of times to attempt to reconnect in the event of a data provider crash. The `enum_cache_timeout` directive specifies, in seconds, how long `sssd_nss` caches requests information about all users.

Configuring the PAM Service

The `sssd` package also provides a PAM module, `sssd_pam`, which is configured in the `[pam]` section of `/etc/sssd/sssd.conf`. The following is an example that includes only a partial list of configurable directives:

```
[pam]

reconnection_retries = 3
offline_credentials_expiration = 2
offline_failed_login_attempts = 3
offline_failed_login_delay = 5
```

The `offline_credentials_expiration` directive specifies, in days, how long to allow cached logins if the authentication provider is offline.

The `offline_failed_login_attempts` directive specifies how many failed login attempts are allowed if the authentication provider is offline.

To update the PAM configuration to reference all of the SSSD modules, use the `authconfig` command as follows to enable SSSD for system authentication:

```
# authconfig --update --enablesssd --enablesssdauth
```

This command auto-generates the PAM configuration file to include the necessary `pam_sss.so` entries.

Configuring SSSD Domains

- SSSD domains are also configured in separate sections of `/etc/sssd/sssd.conf`.
- The syntax is:

```
[domain/Name]
id_provider = type
auth_provider = type
provider_specific = value
global = value
```

- Supported identity providers include `ldap`, `local`, or `proxy`.
- Supported authentication providers include `ldap`, `krb5`, `proxy`, or `none`.
- Provider-specific directives apply to the specified provider.
- Global directives apply to all domains.

SSSD domains are a combination of an identity provider and an authentication method. SSSD works with LDAP identity providers (including OpenLDAP, Red Hat Directory Server, and Microsoft Active Directory) and can use native LDAP authentication or Kerberos authentication. When configuring a domain, you define both where the user information is stored and how those users are allowed to authenticate to the system.

Similar to SSSD services, SSSD domains are also configured in separate sections of the `/etc/sssd/sssd.conf` file. The services and the domains are identified in the `[sssd]` section. Example:

```
[sssd]
domains = LDAP
services = nss, pam
```

This example specifies an LDAP domain. The domain section of the configuration would begin with the following header:

```
[domain/LDAP]
```

The domain configuration section would then specify the identity provider, the authentication provider, and any specific configuration to access the information in those providers.

The following is an example of a domain section:

```
[domain/LDAP]
id_provider = ldap
ldap_uri = ldap://ldap.example.com
ldap_search_base = dc=example,dc=com
auth_provider = krb5
krb5_server = kerberos.example.com
krb5_realm = EXAMPLE.COM
min_id = 10000
max_id = 20000
```

Identity Provider

The `id_provider` specifies the data provider identity back end to use for this domain. Supported back ends are:

- **`proxy`:** Supports a legacy NSS provider
- **`local`:** SSSD internal local provider
- **`ldap`:** LDAP provider

The `ldap_uri` directive gives a comma-separated list of the URIs (Universal Resource Identifiers) of the LDAP servers, in order of preference, to which SSSD connects.

The `ldap_search_base` directive gives the base DN to use for performing LDAP user operations.

Authentication Provider

The `auth_provider` directive specifies the authentication provider used for the domain. If unspecified, the `id_provider` is used. Supported authentication providers are:

- **`ldap`:** Native LDAP authentication
- **`krb5`:** Kerberos authentication
- **`proxy`:** Relays authentication to some other PAM target
- **`none`:** Disables authentication explicitly

The `krb5_server` directive gives a comma-separated list of Kerberos servers, in order of preference, to which SSSD connects.

The `krb5_realm` directive gives the Kerberos realm to use for Simple Authentication and Security Layer (SASL)/Generic Security Services API (GSS-API) authentication. Configuration of SASL connections by using GSS-API is required before SSSD can use Kerberos to connect to the LDAP server.

The last two directives, `min_id` and `max_id`, are examples of global attributes that are available to any type of domain. Other attributes include cache and timeout settings. These two directives specify UID and GID limits for the domain. If a domain contains an entry that is outside these limits, it is ignored.

Start or restart the `sssd` service after making any configuration changes to domains or services:

```
# systemctl start sssd
```

Quiz

Which of the following stores information in a structure, called a directory, that is optimized for searches?

a. NIS
b. OpenLDAP
c. Winbind
d. Kerberos
e. SSSD

Summary

In this lesson, you should have learned how to:

- Describe authentication options
- Describe the Authentication Configuration Tool
- Describe LDAP
- Describe OpenLDAP
- Describe OpenLDAP server and client utilities
- Configure LDAP authentication
- Configure Winbind authentication
- Configure Kerberos authentication
- Describe IPA Identity Management and Authentication Services
- Describe SSSD services and domains

Practice 3: Overview

The practices for this lesson cover the following:

- Configuring an OpenLDAP server
- Implementing OpenLDAP authentication
- Authenticating from an OpenLDAP client

4

Pluggable Authentication Modules (PAM)

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of PAM
- Describe PAM configuration files
- Describe PAM authentication modules
- Describe PAM module types
- Describe PAM control flags
- Walk through PAM authentication examples

Introduction to PAM

- PAM allows you to configure how applications use authentication to verify the identity of a user.
- Configuration files are located in the `/etc/pam.d` directory.
- Each configuration file has the same, or a similar, name as the application it authenticates, for example:
  - `login`, `sudo`, `sshd`, `su`, `passwd`
- Each configuration file lists authentication modules that contain the authentication code.
- Authentication modules are shared libraries located in `/lib/security` (and `/lib64/security`).
- PAM documentation includes man pages for most modules and SAG in `/usr/share/doc/pam-<version>`.

Pluggable Authentication Modules (PAM) is an authentication mechanism that allows you to configure how applications use authentication to verify the identity of a user. Many system applications in Oracle Linux use PAM for authentication and authorization.

The PAM configuration files are located in the `/etc/pam.d` directory. Each of these files have names that are the same as, or similar to, the name of the application that they authenticate. For example, there is a configuration file for the `su` process named `su`.

Each PAM configuration file contains a group of directives that define the authentication module as well as any controls or arguments. The following example is from the `su` file:

```
account    sufficient    pam_succeed_if.so uid = 0 use_uid quiet
```

The four directives in this example are known by the following names:

- **`module_interface`**: `account`
- **`control_flag`**: `sufficient`
- **`module_name`**: `pam_succeed_if.so`
- **`module_arguments`**: `uid = 0 use_uid quiet`

The PAM authentication module name in this example is `pam_succeed_if.so`. The authentication modules are shared libraries and are located in the `/lib/security` and `/lib64/security` directories.

PAM Configuration Files

- Configuration files are located in the `/etc/pam.d` directory.
- Each PAM configuration file contains a list, or stack, of calls to authentication modules. Example:

```
# cat /etc/pam.d/su
auth       sufficient   pam_rootok.so

auth       required     pam_wheel.so use_uid
auth       include      system-auth
account    sufficient   pam_succeed_if.so uid = 0 use_uid
account    include      system-auth
password   include      system-auth
session    include      system-auth
session    optional     pam_xauth.so
```

- The authentication modules have a `.so` extension.
- The `system-auth` is a configuration file that is "included."

Each PAM configuration file contains a list, or stack, of calls to authentication modules. The example shows the contents of the `su` configuration file.

```
# cat /etc/pam.d/su
auth       sufficient   pam_rootok.so
auth       required     pam_wheel.so use_uid
auth       include      system-auth
account    sufficient   pam_succeed_if.so uid = 0 use_uid quiet
account    include      system-auth
password   include      system-auth
session    include      system-auth
session    optional     pam_xauth.so
```

This file references four authentication modules: `pam_rootok.so`, `pam_wheel.so`, `pam_succeed_if.so`, and `pam_xauth.so`. The `pam_wheel.so` and `pam_succeed_if.so` modules include module arguments.

The `system-auth` file is not an authentication module but is a common configuration file for PAM-ified services. Contents of the `system-auth` file are appended to the `su` configuration file and processed as if they were part of the file.

PAM Authentication Modules

- PAM provides several authentication modules in shared libraries.
- These are located in `/lib/security` (or `/lib64/security` for 64-bit Linux). Example:

```
# ls /lib64/security
pam_access.so      pam_limits.so       pam_chroot.so
pam_cap.so         pam_listfile.so     pam_sss.so
pam_chroot.so      pam_localuser.so    pam_stress.so
pam_console.so     pam_loginuid.so     pam_succeed_if.so
pam_cracklib.so    pam.mail.so         pam_systemd.so
pam_debug.so       pam_mkhomedir.so    pam_tally2.so
pam.deny.so        pam_motd.so         pam_time.so
...
```

- This pluggable, modular architecture provides flexibility in setting authentication policies.

PAM provides several authentication modules in shared libraries, which are located in `/lib/security` (or `/lib64/security` for 64-bit Linux).

This pluggable, modular architecture provides flexibility in setting authentication policies for a system. With the authentication code separate from the application code, you can configure the authentication mechanism for a given application without ever touching the application.

There are manual pages for most of the PAM modules. The following example lists a partial man page for the `pam_access` module.

```
# man pam_access
...
NAME
     pam_access - PAM module for logdaemon style login access...
SYNOPSIS
     pam_access.so [debug] [nodefgroup] [noaudit] [accessfile...
DESCRIPTION
     The pam_access PAM module is mainly for access manage...
...
```

PAM Module Types

- The first column in the `/etc/pam.d` configuration file (`auth` in this example) is the module interface:

```
auth     sufficient     pam_rootok.so
```

- Module types represent a different aspect of the authorization process.
- Four types are available:
  - `auth`: Proves the user is authorized to use the service
  - `account`: Determines whether an already authenticated user is allowed to use the service
  - `password`: Updates user authentication credentials
  - `session`: Configures and manages user sessions

The following lists partial contents of the `/etc/pam.d/su` configuration file.

```
auth       sufficient   pam_rootok.so
auth       required     pam_wheel.so use_uid
session    optional     pam_xauth.so
```

The first column is the PAM module interface, or module type indicator. Four module types are available. Only the `auth` and `account` types determine authorization to run a command.

- **`auth`:** Proves that the user is authenticated, or authorized to use the service. This can be done by requesting and verifying the validity of a password. Modules with this interface can also set credentials, such as group memberships or Kerberos tickets.
- **`account`:** Verifies whether an already authenticated user is allowed access to the application. For example, it could check whether a user account has expired or whether a user is allowed to use this service at a particular time of day.
- **`password`:** Is used when a user tries to update his or her authentication token
- **`session`:** Configures and manages user sessions. Performs tasks that are needed to allow access, such as mounting a user's home directory and unmounting when the service is terminated.

PAM Control Flags

- The second column in the `/etc/pam.d` configuration file (`sufficient` in this example) is the control flag:

```
auth     sufficient     pam_rootok.so
```

- Each PAM module generates a success or failure result.
- Control flags tell PAM what to do with the result:
  - `required`: The module must pass before access is granted. The user is not notified immediately if the module fails.
  - `requisite`: This is similar to `required` control flag except that the user is notified immediately if the module fails.
  - `sufficient`: Failure is not necessarily fatal, depending on other module test results.
  - `optional`: The module result is ignored unless this is the only module.
  - `include` | `substack`: This includes lines from another file.

The following lists partial contents of the `/etc/pam.d/su` configuration file.

```
auth       sufficient   pam_rootok.so
auth       required     pam_wheel.so use_uid
session    optional     pam_xauth.so
```

The second column is the PAM control flag. PAM reads the stack from top to bottom and calls the modules listed in the configuration file. Each PAM module generates a success or failure result when called. Control flags tell PAM what to do with the result.

Modules can be stacked in a particular order, and the control flags determine how important the success or failure of a particular module is to the overall goal of authenticating the user to the service. The available control flags are listed:

- **`required`:** All `required` modules are tried and all must "pass" before access is granted by PAM. If the module fails at this point, the user is not notified until all modules in the stack have been executed.
- **`requisite`:** Similar to `required`, in which success of the module is required for authentication to continue. However, if the module fails, no further modules are executed. The user is notified immediately of the first failed `required` or `requisite` module test.

- **`sufficient`:** Success indicates that this module type has succeeded and no subsequent `required` modules of this type are executed. Failure is not fatal to the stack of this module type, however. PAM processes the remaining modules listed to decide whether access is allowed.
- **`optional`:** The module result is generally ignored. A module flagged as `optional` becomes necessary for successful authentication when it is the only module in the stack for a particular service.
- **`include`:** Unlike the other controls, this does not relate to how the module result is handled. This control flag pulls in all lines of the given type from the configuration file specified as an argument to this control.
- **`substack`:** This control is similar to `include` in that it includes all lines of the given type from the configuration file specified as an argument. The differences from `include` is that evaluation of the `done` and `die` actions in a substack does not cause skipping the rest of the complete module stack, but only of the substack.

PAM also includes some predefined actions in the control flag field, which are included within brackets as follows:

```
[value1=action1 value2=action2 ...]
```

The use of brackets in the control flag field gives you full control of PAM's actions. When the result returned by a function matches `value`, `action` is evaluated.

PAM: Example #1

- The contents of the `/etc/pam.d/sshd` file are listed in the notes.
- The `pam_serpermit` PAM module allows or denies login depending on the SELinux enforcement state.
  - SELinux is covered in a subsequent lesson in this course.
- The `pam_nologin` module prevents users from logging in to the system when either the `/var/run/nologin` file or the `/etc/nologin` file exists.
- The `pam_selinux` module sets up the default SELinux security context for the next executed process.
- The `pam_loginuid` module records the user's login UID.
- The `pam_keyinit` module is the kernel session keyring initializer.

To see PAM in action, the following example steps through the processing of the `sshd` application's authentication stack. The contents of the `sshd` configuration file are listed.

```
# cat /etc/pam.d/sshd
#%PAM-1.0
auth       required     pam_sepermit.so
auth       substack     password-auth
auth       include      postlogin
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
session    required     pam_selinux.so close

session    required     pam_loginuid.so
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
session    include      postlogin
```

Lines that begin with # are comments and are not processed. There are three comment lines in this file. The remaining lines tell PAM to do something as part of the authentication process.

The first line is:

```
auth       required     pam_sepermit.so
```

This line uses the `pam_sepermit.so` module to attempt to authenticate the user. Because the control flag is `required`, this module is `required` to succeed if the authentication stack is to succeed.

There are manual pages for most of the PAM modules. The man page for `pam_serpermit` states that the PAM module allows or denies login depending on the SELinux enforcement state. SELinux is covered in a subsequent lesson in this course. The man page goes on to say that, "When the user which is logging in matches an entry in the configuration file, `sepermit.conf`, the user is allowed access only when the SELinux is in enforcing mode." Otherwise the user is denied access. The configuration file, `sepermit.conf`, also has a man page that describes the configuration options.

The next two lines include a couple of configuration files:

```
auth       substack     password-auth
auth       include      postlogin
```

These files are located in the `/etc/pam.d` directory. These files also have man pages. The `password-auth` configuration file is for applications that handle authentication from different types of devices via simultaneously running individual conversations instead of one aggregate conversation. The `postlogin` configuration file is included from all individual service configuration files that provide login service with shell or file access.

The fourth line is:

```
account    required     pam_nologin.so
```

This line uses the `pam_nologin.so` module to verify whether an already authenticated user is allowed access to the application. The man page for `pam_nologin` states that the PAM module prevents users from logging in to the system when `/var/run/nologin` or `/etc/nologin` exists. Because the control flag is `required`, this module is `required` to succeed if the authentication stack is to succeed.

The fifth and sixth lines are `account` and `password` type entries, which both include the `password-auth` configuration file. The remaining lines are `session` type entries that use the following PAM modules:

```
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
```

`pam_selinux` is a PAM module that sets up the default SELinux security context for the next executed process. These entries have module arguments, `close` and `open`. When a session is ended, the `close_session` part of the module restores old security contexts that were in effect before the change made by the `open_session` part of the module.

`pam_loginuid` is a PAM module that records the user's login UID, which is necessary for applications to be correctly audited.

`pam_keyinit` is the kernel session keyring initializer PAM module.

PAM: Example #2

Example # 2:

- Uses *value=action* pairs in the control flag field, allowing full control of PAM actions
  - `[user_unknown=ignore success=ok ignore=ignore default=bad]`
- Uses authentication module arguments
  - `pam_unix.so nullok try_first_pass`
  - `pam_succeed_if.so uid >= 500 quiet`
- Includes the contents of the common configuration file, `system-auth`
  - `system-auth` is included in nearly all individual service configuration files.
  - `system-auth` is auto-generated each time the `authconfig` command runs.

To demonstrate another example of PAM in action, the following is a partial listing of the `login` service's authentication stack:

```
# cat /etc/pam.d/login
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad]
pam_securetty.so
auth       substack   system-auth
account    required   pam_nologin.so
account    include    system-auth
password   include    system-auth
```

Processing of each line in the stack is described:

```
auth [user_unknown=ignore success=ok ignore=ignore default=bad
pam_securetty.so
```

The first noncommented line calls the `pam_securetty.so` module, which allows `root` logins only if logging in on a secure TTY, as defined in the `/etc/securetty` file. This module has no effect on nonroot users. This entry also contains specific actions for the control flag.

Several predefined control flag actions are available. This example uses the following actions and values:

- **user_unknown=ignore**:
  - user_unknown: The user is not known to the underlying authentication module.
  - ignore: The return status does not contribute to the return code.
- **success=ok**:
  - success: Successful function return
  - ok: If the module fails, the total stack state is fail. If the stack is already in fail status, the return code of this module does nothing.
- **ignore=ignore**:
  - ignore: Ignore underlying account modules regardless of whether the control flag is required, optional, or sufficient.
  - ignore: The return status does not contribute to the return code.
- **default=bad**:
  - default: All not explicitly mentioned values
  - bad: The return status is set to fail.

```
auth    substack      system-auth
```

The next line includes the contents of a common configuration file, system-auth, into the /etc/pam.d/login file. The system-auth configuration file is included in nearly all individual service configuration files. It checks that the user who is logging in is authorized to do so, including verification of the username and password. The system-auth configuration file is auto-generated each time the authconfig command is executed. An example of auth entries in a system-auth file are:

```
auth    required      pam_env.so
auth    sufficient    pam_fprintd.so
auth    sufficient    pam_unix.so nullok try_first_pass
auth    requisite     pam_succeed_if.so uid >= 1000 quiet
auth    required      pam_deny.so
```

The pam_env.so module allows the setting and unsetting of environmental variables. The pam_fprintd.so module is used for fingerprint authentication. The pam_unix.so module is the standard UNIX password authentication module. The nullok argument overrides the default action to not permit the user access to a service if the user's password is blank. The try_first_pass argument tries the previous stacked module's password before prompting the user for his or her password. The pam_succeed_if.so module tests account characteristics. In this case, it tests for uid >= 1000. The quiet argument means do not log success or failure to the system log file. The pam_deny.so module is the locking-out PAM module and can be used to deny access.

```
account     required    pam_nologin.so
```

The next line uses the pam_nologin.so module. This module prevents users from logging in to the system when the /var/run/nologin file or the /etc/nologin.txt file exists. This module has no affect on the root user.

Some of the remaining entries also include the contents of the system-auth file for module types of account, password, and session.

Quiz

Which of the following are examples of PAM module types?

a. requisite
b. required
c. auth
d. account
e. password
f. sufficient
g. session

Introduction to SELinux

- Standard Linux security is based on DAC.
- SELinux provides finer grained control.
- SElinux runs in three modes:
  - Enforcing
  - Permissive
  - Disabled
- Display the SELinux mode with `sestatus` or `getenforce` commands.
- SELinux also provides "Booleans."

In practice 4-1, you are presented with a new topic in Oracle Linux, SELinux. SELinux stands for “Security-Enhanced Linux” and is covered in a subsequent lesson in this course. This brief introduction helps you understand the tasks you are directed to perform, and why, in practice 4-1: Configuring PAM for a Single Login Session.

Standard Linux security is based on Discretionary Access Control (DAC). With DAC, access to files and devices are based solely on user identity and ownership. Each file can have read, write, and execute permissions for the owner of the file, for the group, and for other users.

SELinux was created by the US National Security Agency to provide a finer-grained level of control over files, processes, users, and applications in the system. It is an enhancement to the Linux kernel, and it implements a different type of security called Mandatory Access Control (MAC). MAC policy is centrally managed rather than being managed by the user.

SELinux runs in one of three modes:

- **Enforcing:** Access is denied to users and programs unless permitted by SELinux security policy rules.
- **Permissive:** The security policy rules are not enforced, but SELinux sends denial messages to a log file.
- **Disabled:** SELinux does not enforce a security policy because no policy is loaded in the kernel. Only DAC rules are used for access control.

You can use the `sestatus` command to display the SELinux mode as well as some additional information about SELinux.

```
# sestatus
SELinux status:    enabled
...
Current mode:      enforcing
...
```

You can use the `getenforce` command to display the SELinux mode. This command displays the current mode: “Enforcing,” “Permissive,” or “Disabled.” Example:

```
# getenforce
Enforcing
```

SELinux running in “Enforcing” mode is often the cause of a problem in Linux. In some of the practices in this course, you are directed to change the SELinux mode to “Permissive” to get a function of Linux to work properly. You can use the `setenforce` command to change the mode to either “Enforcing” (`1`) or “Permissive” (`0`). Example:

```
# setenforce 0
# getenforce
Permissive
```

SELinux also provides “Booleans,” which allow parts of a SELinux policy to be changed at run time, without reloading or recompiling a SELinux policy. You can display a list of Booleans, state information, and a description of the Boolean by running the following command:

```
# semanage boolean -l
SELINUX boolean   State  Default Description
ftp_home_dir      (off , off)    Allow ftp to read and write ...
...
```

You can change the state of a specific Boolean to either `on` or `off` by using the `setsebool` command. For example, to turn the `ftp_home_dir` Boolean to `on`:

```
# setsebool ftp_home_dir on
```

Use the `getsebool` command to display the state of a specific Boolean. Example:

```
# getsebool ftp_home_dir
ftp_home_dir --> on
```

SELinux provides many other features, functions, and configuration options. This brief introduction helps you understand the tasks you are directed to perform, and why, in the following practice:

- Practice 4-1: Configuring PAM for a Single Login Session

Summary

In this lesson, you should have learned how to:

- Describe the purpose of PAM
- Describe PAM configuration files
- Describe PAM authentication modules
- Describe PAM module types
- Describe PAM control flags
- Walk through PAM authentication examples

Practice 4: Overview

The practices for this lesson cover the following:

- Configuring PAM for a single login session
- Configuring PAM to prevent non-`root` login

SELinux is referenced in the following practice:

- Practice 4-1: Configuring PAM for a Single Login Session

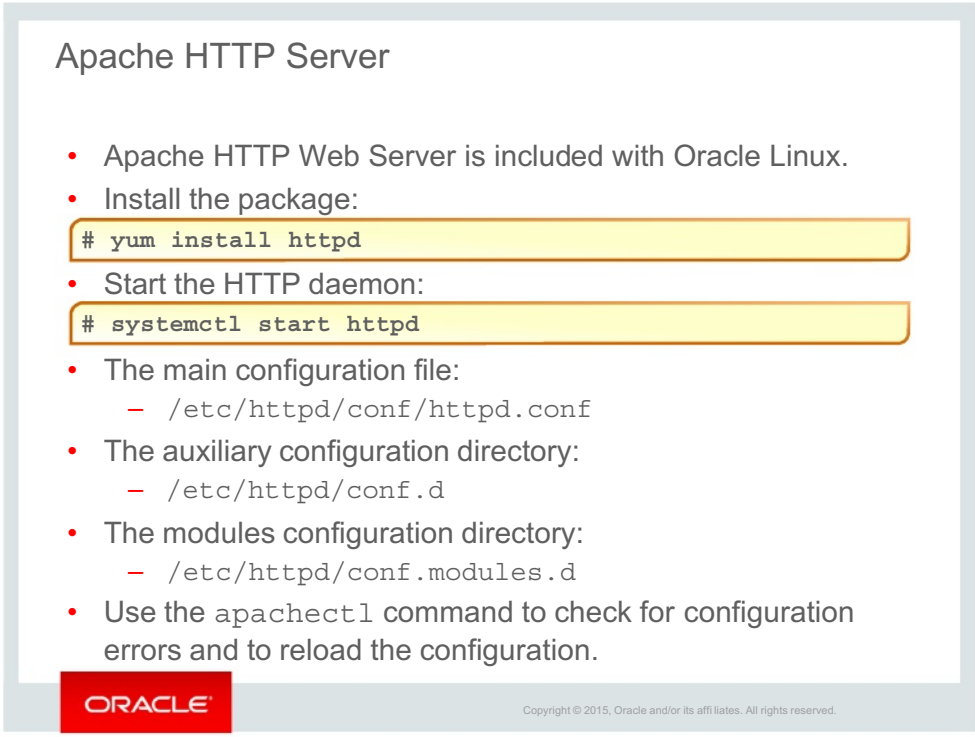
5

Web and Email Services

Objectives

After completing this lesson, you should be able to:

- Describe the Apache HTTP Web Server
- Configure Apache directives
- Configure Apache containers
- Configure Apache virtual hosts
- Describe email program classifications: MUA, MTA, MDA
- Describe email protocols: SMTP, POP, IMAP
- Describe the Postfix SMTP server
- Describe the Sendmail SMTP server
- Configure Sendmail on a client system



The Apache HTTP Server, an open-source web server developed by the Apache Software Foundation, is included with Oracle Linux. The Apache server is used to host web content. It responds to requests for content from web browsers, such as Internet Explorer and Firefox.

To configure your system as a web server, begin by installing the `httpd` software package.

```
# yum install httpd
```

Use the `systemctl` utility to enable the HTTP daemon to start at boot time and also to start the daemon immediately.

```
# systemctl enable|start httpd
```

The main configuration file for Apache is `/etc/httpd/conf/httpd.conf`. An auxiliary directory, `/etc/httpd/conf.d`, also exists to store configuration files that are included in the main configuration file. Configuration files that load modules are in the `/etc/httpd/conf.modules.d` directory.

A new `apachectl` command is available in Oracle Linux 7. The following example uses the `configtest` subcommand to check the configuration for possible errors.

```
# apachectl configtest
```

Use the `graceful` subcommand to reload the configuration without affecting active requests.

```
# apachectl graceful
```

Configuring Apache

Examples of configuration directives in the configuration file:

- `Listen 192.168.2.1:8080`
- `ServerName www.example.com:80`
- `ServerRoot /etc/httpd`
- `DocumentRoot /var/www/html`
- `UserDir enabled oracle`
- `ErrorLog logs/error_log`
- `LoadModule auth_basic_module modules/mod_auth_basic.so`
- `Order deny,allow`
- `Deny from all`
- `Allow from .example.com`
- `Timeout 60`

The main configuration file for Apache is `/etc/httpd/conf/httpd.conf` Apache runs as installed, but you can modify configuration directives in this file to customize Apache for your environment. Some of these directives are described here. An index of all the directives is available at http://httpd.apache.org/docs/2.4/mod/directives.html

`Listen [`*`IP address`*`:]`*`port`*

Tells the server to accept incoming requests on the specified port or IP address and port combination. By default, the server responds to requests on all IP interfaces on port 80. If you specify a port number other than 80, a request to the server must include the port number (as in `www.example.com:8080`). This is a required directive. Examples are as follows:

```
Listen 80
Listen 192.168.2.1:8080
```

`ServerName` ***`FQDN`*****`[:`***`port`**`]`**

Specifies the fully qualified domain name or IP address of the server and an optional port that Apache listens on. The FQDN must be able to be resolved by DNS. If no FQDN is specified, Apache performs a DNS reverse name lookup on the IP address. If no port is specified, the server uses the port from the incoming request, as shown in the following example:

```
ServerName www.example.com:80
```

ServerRoot *directory-path*

The top of the directory hierarchy under which the Apache server's configuration, error, and log files are kept. The default is `/etc/httpd`. Do not add a slash at the end of *directory-path*:

```
ServerRoot /etc/httpd
```

DocumentRoot *directory-path*

The top of the directory hierarchy that holds the Apache server content. Do not end the path name with a slash. The `apache` user needs read access to any files and execute access to the directory and any subdirectories in the hierarchy. The following is the default:

```
DocumentRoot /var/www/html
```

UserDir *directory-path* | *disabled* | *enabled user-list*

Allows users identified by the `user-list` argument to publish content from their home directories. The *directory-path* is the name of a directory in a user's home directory from which Apache publishes content. If *directory-path* is not defined, the default is `~/public_html`. The following example enables this feature for user `oracle`. Assuming that the `ServerName` is www.example.com, browsing to http://www.example.com/~oracle displays the `oracle` user's webpage.

```
UserDir enabled oracle
```

ErrorLog *filename* | *syslog*[:*facility*]

Specifies the name of the file, relative to `ServerRoot`, that Apache sends error messages to. Alternatively, `syslog` specifies that Apache must send errors to `rsyslogd`. The optional `facility` argument specifies which `rsyslogd` facility to use. The default facility is `local7`.

```
ErrorLog logs/error_log
```

LoadModule *module filename*

Apache, like the Linux kernel, uses external modules to extend functionality. These modules are called dynamic shared objects (DSOs). The *module* argument is the name of the DSO and *filename* is the path name of the module, relative to `ServerRoot`. More than 60 modules are included with Apache, and more than 50 of these are loaded by default. An index of all the modules is available at http://httpd.apache.org/docs/2.4/mod/.

```
LoadModule auth_basic_module modules/mod_auth_basic.so
```

Allow from *All* | *host* [*host* ...]

Specifies which clients can access content. `All` serves content to any client. Alternatively, you can list the specific hosts that are allowed access to content.

Deny from All | host [host ...]

Specifies which clients are not allowed access to content.

Order deny,allow | allow,deny

Specifies the order in which `Allow` and `Deny` directives are evaluated. `deny,allow` evaluates `deny` directives first and then `allow` directives. The following example grants access to clients from the `example.com` domain only, by first denying access to all and then allowing it from `.example.com`:

```
Order deny,allow
Deny from all
Allow from .example.com
```

Timeout *num*

Specifies the number of seconds Apache waits for network operations to finish. The default is 60.

Testing Apache

You can confirm that Apache is working by pointing a browser on the local system to http://localhost as shown. From a remote system, point a browser to `http://` followed by the `ServerName` directive that you specified in the configuration file. The test page, shown in the slide, confirms that Apache is working correctly.

To test the display of actual content, create an HTML file named `index.html` in the directory specified by the `DocumentRoot` directive (the default directory is `/var/www/html`). Apache automatically displays the `index.html` file in this directory, if it exists.

Apache Containers

- Containers are special directives that group other directives.
  - `<Directory directory-path>` applies directives to directories within *directory-path*.
  - `<IfModule module-name>` applies directives if *module-name* is loaded.
  - `<IfModule !module-name>` applies directives if *module-name* is not loaded.
  - `<Limit method [method] ...>` limits access control directives to specified methods.
- Containers can be nested.

Apache containers are special configuration directives that group other directives. The containers use XML-style tags, meaning that the beginning of a container is `<name>` and the end is `</name>`. An index of all the container directives is available at http://httpd.apache.org/docs/current/sections.htmlThe following are examples of containers:

`<Directory directory-path>`

This container applies directives to directories within *directory-path*. The example applies `Deny`, `Allow`, and `AllowOverride` directives to all files and directories within the `/var/www/html/test` directory hierarchy. Indenting is for readability only.

```
<Directory /var/www/html/test>
     Deny from all
     Allow from 192.168.2.
     AllowOverride All

</Directory>
```

The `AllowOverride` directive in this container specifies classes of directives that are allowed in `.htaccess` files. The `.htaccess` files are other configuration files that typically contain user authentication directives. The `ALL` argument to `AllowOverride` means that all classes of directives are allowed in the `.htaccess` files. There are classes of directives that control authorization, control client access, control directory indexing, and others.

<IfModule [!]*module-name*>

This container applies directives if *module-name* is loaded. With the optional exclamation point, Apache does the inverse; that is, it sets the directives in the container if the *module-name* is not loaded. An example is as follows:

```
<IfModule mod_userdir.c>
     UserDir disabled
</IfModule>
```

<Limit *method* [*method*] …>

This container limits access control directives to specified methods. An HTTP method specifies actions to perform on a Uniform Resource Identifier (URI). Examples of methods are `GET` (the default), `PUT`, `POST`, and `OPTIONS`. The following example disables HTTP uploads (`PUT`) from systems that are not in the `example.com` domain:

```
<Limit PUT>
     Order deny,allow
     Deny from all
     Allow from .example.com
</Limit>
```

<LimitExcept *method* [*method*] …>

This container is the opposite of the `Limit` container in that it limits access control directives to all except specified methods.

The following example uses the `LimitExcept` container but also illustrates that containers can be nested. This example controls access to `UserDir` directories by restricting these directories to be read-only:

```
<Directory /home/*/public_html>
     AllowOverride FileInfo AuthConfig Limit
     Options MultiViews Indexes SymLinksIfOwnerMatch \
     IncludesNoExec
     <Limit GET POST OPTIONS>
          Order allow,deny
          Allow from all
     </Limit>
     <LimitExcept GET POST OPTIONS>
          Order deny,allow
          Deny from all
     </LimitExcept>
</Directory>
```

The `Options` directive controls server features by directory. Some of these are described:

- **`MultiViews`:** Allows a page to be displayed in different languages, for example
- **`Indexes`:** Generates a directory listing if the `DirectoryIndex` directive is not set
- **`SymLinksIfOwnerMatch`:** Follows symbolic links if the file or directory being pointed to has the same owner as the link

Apache Virtual Hosts

- A single Apache server can respond to requests directed to multiple IP addresses or host names.
- Each virtual host can provide different content and be configured differently.
- Use the `<VirtualHost host-name>` container:

```
<VirtualHost www.example1.com>
  ServerName www.example1.com
  DocumentRoot /var/www/example1
  ErrorLog example1.error_log
</VirtualHost>
<VirtualHost www.example2.com>
  ...
</VirtualHost>
```

Apache supports virtual hosts, meaning that a single Apache server can respond to requests directed to multiple IP addresses or host names. Each virtual host can provide content and be configured differently.

You can configure virtual hosts in two ways:

- IP-based Virtual Hosts (host-by-IP)
- Name-based Virtual Hosts (host-by-name)

With host-by-IP, each virtual host has its own IP address and port combination. The Apache web server responds to the IP address that the host resolves as. Host-by-IP is required for serving HTTPS requests due to restrictions in the Secure Sockets Layer (SSL) protocol.

With host-by-name, all virtual hosts share the common IP address. Apache responds to the request by mapping the host name in the request to `ServerName` and `ServerAlias` directives in the particular virtual host's configuration file.

Use the `<VirtualHost host-name>` container to implement virtual hosts. After the first `VirtualHost` is defined, all of the content served by Apache must also be moved into virtual hosts.

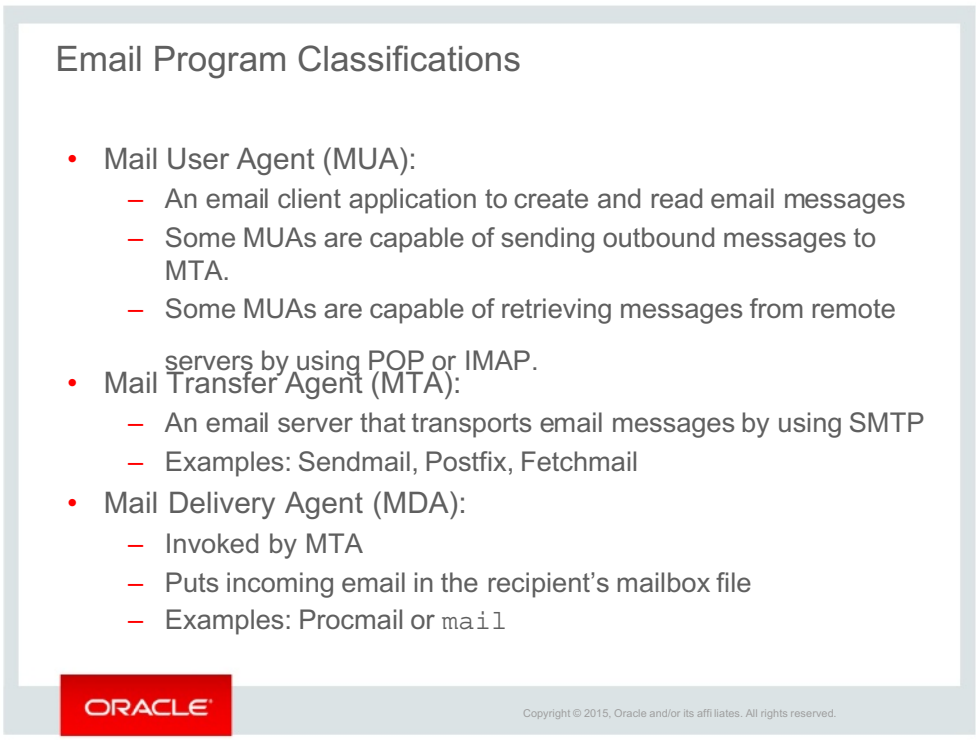
The following example is a simple name-based virtual hosts configuration:

```
<VirtualHost *:80>
     ServerName example1.com
     ServerAlias www.example1.com
     DocumentRoot /var/www/example1
     ErrorLog example1.error_log
</VirtualHost>
<VirtualHost *:80>
     ServerName example2.com
     ServerAlias www.example2.com
     DocumentRoot /var/www/example2
     ErrorLog example2.error_log
</VirtualHost>
```

Quiz

On which port does Apache listen for client requests by default?

a. 443
b. 8080
c. 80
d. 280



An email message is created by a mail client program called a Mail User Agent (MUA) and delivered to the recipient's email server by Mail Transfer Agents (MTAs). From here, a Mail Delivery Agent (MDA) puts the message in the recipient's mailbox file.

Mail User Agent (MUA)

An MUA is an email client application that allows you to create and read email messages, set up mailboxes to store and organize messages, and send outbound messages to an MTA. Many MUAs can also retrieve email messages from remote servers by using Post Office Protocol (POP) or Internet Message Access Protocol (IMAP).

Mail Transfer Agent (MTA)

An MTA transports email messages between systems by using Simple Mail Transfer Protocol (SMTP). It provides mail delivery services from a client program to a destination server, possibly traversing several MTAs along the way. Oracle Linux offers two MTAs, Postfix and Sendmail, and also includes a special purpose MTA called Fetchmail.

Mail Delivery Agent (MDA)

An MDA, such as Procmail, is invoked by the MTA to put incoming email in the recipient's mailbox file. MDAs perform the actual delivery. They distribute and sort messages on the local system for an email client application to access.

Email Protocols

- Simple Mail Transfer Protocol (SMTP):
  - This is a transport protocol (an MTA).
  - Specify the SMTP server when configuring the client program.
  - Configure relay restrictions to limit junk email.
- Post Office Protocol (POP):
  - An email access protocol
  - Used by client programs to retrieve email messages
- Internet Message Access Protocol (IMAP):
  - This is similar to POP.
  - Email is kept on the server when using IMAP.
- POP and IMAP services are provided by the `dovecot` package.

Several different network protocols are required to deliver email messages. These protocols work together to allow different systems, often running different operating systems and different email programs, to send and receive email. The most commonly used protocols for transferring email are described here:

Simple Mail Transfer Protocol (SMTP)

SMTP is considered to be a transport protocol (an MTA), as opposed to an email access protocol. It provides mail delivery services from a client program to a server, and from an srcinating server to the destination server. You must specify the SMTP server when configuring the client program. You can also specify a remote SMTP server for outgoing email.

SMTP does not require authentication. Anyone can send anyone email, including junk email, also known as spam or unsolicited bulk email. You can configure relay restrictions that limit users from sending email through your SMTP server. Servers without any restrictions are called open relay servers.

The SMTP programs provided by Oracle Linux are Postfix and Sendmail. Because these programs use SMTP, they are often referred to as SMTP server programs.

Post Office Protocol (POP)

POP is an email access protocol used by client programs to retrieve email messages from remote servers. Users on client systems usually have an email account on a server run by their employer or an Internet Service Provider (ISP). On Linux systems, the MUA on the receiving system either reads the mailbox file or retrieves the email from a remote SMTP server, using POP or IMAP. Unless you own a domain at which you want to receive email, you do not need to set up Sendmail as an incoming SMTP server.

Internet Message Access Protocol (IMAP)

IMAP is similar to POP in that it is an email access protocol used to retrieve email remotely. The IMAP server is provided by the same `dovecot` package that provides the POP server. There are no new software packages to install.

While POP email clients typically delete the message on the server after it has been successfully retrieved, email is kept on the server when using IMAP. The entire message is downloaded only when it is opened. Messages can be read or deleted while still on the server. Both POP and IMAP allow you to manage mail folders and create multiple mail directories to organize and store email.

Oracle Linux includes the `dovecot` package to implement both the POP and IMAP protocols. To install the package:

```
# yum install dovecot
```

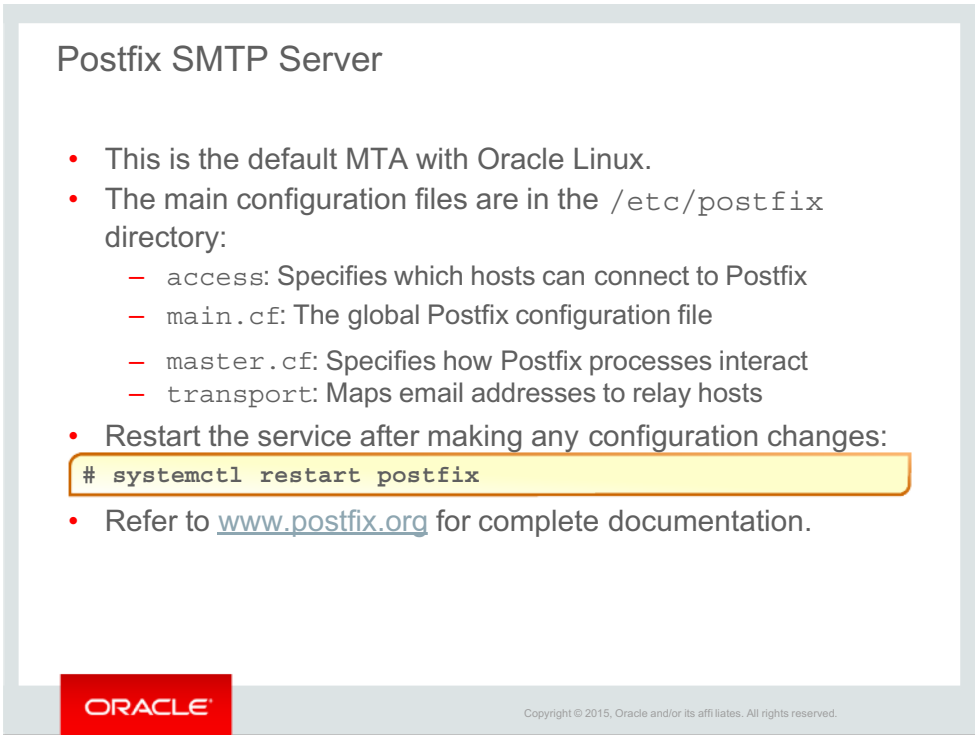
Start the daemon by entering the following command:

```
# systemctl start dovecot
```

To ensure that the service starts at boot time:

```
# systemctl enable dovecot
```

By default, `dovecot` runs IMAP and POP together with their secure versions using Secure Sockets Layer (SSL) encryption for client authentication and data transfer sessions. When starting `dovecot`, it reports that it started the IMAP server but it also starts the POP server. The servers provided by `dovecot` are configured to work as installed. You typically do not need to modify the configuration file, `/etc/dovecot/dovecot.conf`. Refer to `/usr/share/doc/doc/dovecot*` for more information.



Postfix and Sendmail are two MTAs (SMTP servers) included with Oracle Linux. Postfix is configured as the default MTA. It is easier to administer than Sendmail, but does not include as many features. Postfix has a modular design that consists of a master daemon and several smaller processes. It stores its configuration files in the `/etc/postfix` directory. Some of the configuration files are described. Refer to www.postfix.org for complete documentation.

- **`access`:** This file is used for access control and specifies which hosts are allowed to connect to Postfix.
- **`main.cf`:** This is the global Postfix configuration file in which most of the configuration options are specified.
- **`master.cf`:** This file specifies how the Postfix master daemon interacts with the smaller processes to deliver email.
- **`transport`:** This file maps email addresses to relay hosts.

By default, Postfix does not accept network connections from any system other than the local host. To enable mail delivery for other hosts, edit the `/etc/postfix/main.cf` file and configure the domain, host name, and network information for the other hosts. Restart the service after making any configuration changes:

```
# systemctl restart postfix
```

Sendmail SMTP Server

- This is an MTA included with Oracle Linux.
- One of the oldest and most common MTAs on the Internet
- Install two packages:

```
# yum install sendmail sendmail-cf
```

- Configuration files are located in `/etc/mail`:
  - `sendmail.mc`: Is the main configuration file
  - `access`: Specifies a relay host
  - `virtusertable`: Serves email to multiple domains
  - `mailertable`: Forwards email from one domain to another
- You must regenerate the configuration files after editing:

```
# systemctl restart sendmail
# make all -C /etc/mail
```

Sendmail is also included with Oracle Linux. It is one of the oldest and most commonly used MTAs on the Internet. The main purpose of Sendmail is to transfer email between systems, but it is highly configurable and capable of controlling almost every aspect of how email is handled.

To use Sendmail, install the following packages. The `sendmail-cf` package is required to configure Sendmail. The `procmail` package is installed as a dependency. In the default setup, the Sendmail MTA uses Procmail as the local MDA. The Procmail application writes email to the recipient's mailbox file.

```
# yum install sendmail sendmail-cf
```

The Sendmail configuration files are located in `/etc/mail`. The main configuration file is `sendmail.cf`, but it is not intended to be edited by using a text editor. Make any configuration changes in the `sendmail.mc` file and then generate a new `sendmail.cf` file by restarting the `sendmail` service:

```
# systemctl restart sendmail
```

You can also run the following `make` command, which calls the `Makefile` file in the `/etc/mail` directory and regenerates mail configuration files that have been modified.

```
# make all -C /etc/mail
```

Some of the other configuration files in the /etc/mail directory are described here:

- **access:** This file sets up a relay host. A relay host processes outbound mail for other systems. The default configuration is to relay mail only from the local host:

  ```
  Connect: localhost.localdomain        RELAY
  Connect: localhost                    RELAY
  Connect: 127.0.0.1                    RELAY
  ```

  To configure your system to relay mail from other systems (for example, the 192.168 subnet) add the following entry:

  ```
  Connect: 192.168                      RELAY
  ```

- **virtusertable:** This file serves email to multiple domains. Each line starts with the address that the email was sent to, followed by the address Sendmail forwards the email to. For example, the following entry forwards email addressed to any user at foo.org to the same username at example.com:

  ```
  @foo.org    %1@example.com
  ```

- **mailertable:** This file forwards email from one domain to another. The following example forwards email sent to the foo.org domain to the SMTP server for the example.com domain:

  ```
  foo.org     smtp:[example.com]
  ```

The configuration files in the /etc/mail directory have corresponding .db files. Example:

```
# ls /etc/mail/*table*
domaintable      mailertable      virtusertable
domaintable.db   mailertable.db   virtusertable.db
```

Make any configuration changes to the files without extensions. Sendmail uses the .db files, however. To update or regenerate the .db files for Sendmail, either restart the sendmail service or run the make command after making any configuration changes.

/etc/aliases

The /etc/aliases file can also be used to forward incoming email messages. Use the file to map inbound addresses to local users, files, commands, and remote addresses. The following example forwards mail sent to admin on the local system to several users, including user4, who is on a different system:

```
admin:      user1, user2, user3, user4@different.com
```

To direct email to a file, specify the absolute path name of the file in place of the destination address. The /etc/aliases file is writable only by the root user.

~/.forward

Individual users can forward incoming email messages by creating a .forward file in their home directory. Simply specify a different email address in the ~/.forward file. Example:

```
user1@host02.example.com
```

You can also specify another user, or a file, or a command to pipe the email to. Separate multiple entries with a comma or a newline.

Configuring Sendmail on a Client

- Sendmail on a client system simply relays outbound mail to an SMTP server.
- A remote SMTP server, typically an ISP, relays email to its destination.
- Edit the following line in `/etc/mail/sendmail.mc`:
  - `dnl define(`SMART_host', 'smtp.your.provider')dnl`
- Remove the `dnl` at the beginning of the line.
- Include the ISP's SMTP server name:
  - `define(`SMART_host', 'smtp.isp.com')dnl`
- Restart the `sendmail` service:

```
# systemctl restart sendmail
```

Sendmail on a client system simply relays outbound mail to an SMTP server. A remote SMTP server, typically an ISP, relays outbound email to its destination. The following example configuration sends email only to the SMTP server that srcinates on the local system. It does not forward email srcinating from other systems. This configuration does not handle inbound email either. Client systems normally use POP or IMAP to receive email.

To configure the system as described, locate the following entry in the main email configuration file, `/etc/mail/sendmail.mc`:

```
dnl define(`SMART_host', 'smtp.your.provider')dnl
```

The `dnl` at the beginning of the line is a comment. Delete these characters. Replace "`smtp.your.provider`" with the FQDN of your ISP's SMTP server. You can choose to delete the `dnl` characters at the end of the line, or not delete them. Assuming that your ISP's SMTP server is `smtp.isp.com`, change the line to appear as follows:

```
define(`SMART_host', 'smtp.isp.com')dnl
```

Restart the `sendmail` service, which regenerates the `sendmail.cf` file.

```
# systemctl restart sendmail
```

Send an email message to an account that you have on a remote server to ensure that `sendmail` is relaying your email.

Quiz

Sendmail is an example of what type of email program classification?

a. MUA
b. MTA
c. MDA
d. MMA

Summary

In this lesson, you should have learned how to:

- Describe the Apache HTTP Web Server
- Configure Apache directives
- Configure Apache containers
- Configure Apache virtual hosts
- Describe email program classifications: MUA, MTA, MDA
- Describe email protocols: SMTP, POP, IMAP
- Describe the Postfix SMTP server
- Describe the Sendmail SMTP server
- Configure Sendmail on a client system

Practice 5: Overview

The practice for this lesson covers the following:

- Configuring the Apache Web Server, Creating a test webpage, and Configuring two Apache virtual hosts

6

Installing Oracle Linux by Using Kickstart

Objectives

After completing this lesson, you should be able to:

- Describe the Kickstart installation method
- Describe the Kickstart file
- Verify the Kickstart file
- Start a Kickstart installation
- Boot into Rescue mode to correct boot problems

The Kickstart Configurator GUI for creating Kickstart files has not been updated for Oracle Linux 7. It does not reflect changes in Kickstart syntax between Oracle Linux 6 and Oracle Linux 7 and therefore is not covered in this lesson.

Kickstart Installation Method

To automate the installation of Oracle Linux:

- Create a Kickstart file that contains installation parameters
- Make the Kickstart file available on a boot disk, on a boot CD, or on the network
- Make the Oracle Linux installation tree available from the installation CD, from the ISO image stored on a hard drive, or over the network
- Use NFS, FTP, or HTTP to provide access to the installation tree over the network
- Initiate the Kickstart installation from the boot prompt

The Kickstart installation method allows you to perform an unattended installation of Oracle Linux. Requirements to implement Kickstart include the creation of a Kickstart file, which contains the answers to all the questions you are asked during a normal installation. You then must make the Kickstart file available, on a boot disk, on a CD, or on the network. Kickstart also needs access to the Oracle Linux installation tree, from the installation CD, from the ISO image stored on a hard drive, or over the network. You can use NFS, FTP, or HTTP to provide access to the installation tree over the network.

To start a Kickstart installation, boot the system from a boot disk or boot CD. Press the Esc key at the boot menu to display the boot prompt. At the boot prompt, enter a special `ks` command to begin the installation. The installation then runs unattended without prompting for additional information.

Kickstart File

- The Kickstart file contains answers to installation questions.
  - The Command section defines installation options and associated values.
  - The `%packages` section contains the names of package groups and individual package names to be installed.
  - The optional `%pre` section contains commands to run before the installation begins.
  - The optional `%post` section contains commands to run after the installation is completed.
- Every installation creates a Kickstart file, `/root/anaconda-ks.cfg`.
  - This file can be used as a template for future installations.

A Kickstart file contains the answers to all the questions posed during an installation. The Kickstart file contains the following sections:

- **Command section:** Defines the installation options and associated values
- **%packages section:** Defines the packages to install
- **%pre and %post sections:** Defines preinstallation and postinstallation commands

Every installation creates a Kickstart file, `/root/anaconda-ks.cfg`. This file can be used "as is" to repeat the installation, or it can be modified to specify different settings.

The following example demonstrates the syntax for providing system information in the Command section. The example defines values for the language, keyboard type, root password, time zone, and network. In this example, the `root` password is encrypted:

```
lang en_US.UTF-8
keyboard --vckeymap=us --xlayouts='us'
rootpw --iscrypted $6$...
timezone America/Denver --isUtc  --nontp
network --bootproto=static --device=eth0 --gateway=192.0.2.1 --
ip=192.0.2.107 --netmask=255.255.255.0 --ipv6=auto --activate
```

List the software packages that you want to install in the package section. Begin the section with the `%packages` command and end the section with the `%end` command. Packages can be specified by using the individual package name or by using the package group name. Group names begin with the `@` sign, whereas individual package names do not. Individual package names can also be specified by using wildcards. If you include the `-` character as a prefix to an individual package name, the package is not installed.

The following example includes both package group names and individual package names. The `pam*` entry is an example of using a wildcard.

```
%packages
@base
@client-mgmt-tools
@console-internet
@core
@system-admin-tools
pax
certmonger
pam*
krb5-workstation
perl-DBD-SQLite
%end
```

The optional preinstallation section contains commands to run on the system immediately after the Kickstart file has been parsed but before the installation begins. The preinstallation section begins with the `%pre` command and ends with the `%end` command. You can access the network in the `%pre` section by using only IP addresses. You cannot use host names or domain names in the `%pre` section because the name service has not yet been configured.

The following example runs the `config-partitions` script, which is stored on an HTTP server. You can store information such as disk partition parameters in a separate file, and use the `%include` command to access this file.

```
%pre
%include http://192.0.2.1/scripts/config-partitions
%end
```

The optional postinstallation section contains commands to run on the system after the installation is completed. The postinstallation section begins with the `%post` command and ends with the `%end` command.

Both the preinstallation and postinstallation sections must be placed at the end of the Kickstart file.

Refer to the *Oracle Linux 7 Installation Guide* for Kickstart file options and additional examples: http://docs.oracle.com/cd/E52668_01/index.html

Verifying the Kickstart File

- Use the `ksvalidator` utility to check for errors in the Kickstart file.
- It does not validate the syntax of `%pre` and `%post` scripts.
- Install the `pykickstart` package:

```
# yum install pykickstart
```

- Provide path to the Kickstart file as argument:

```
# ksvalidator /root/ks.cfg
The following problem occurred on line 15 of the
   kickstart file:
no such option: -b
```

You can use the `ksvalidator` utility to verify the syntax of a Kickstart file. The utility checks for proper quoting, valid options, and ensures that any required options have values.

The `ksvalidator` utility does not validate the syntax of the `%pre` and `%post` scripts. It also does not guarantee that a Kickstart file installs properly because `ksvalidator` cannot deal with the complexities of partitioning and what potentially already exists on the disk.

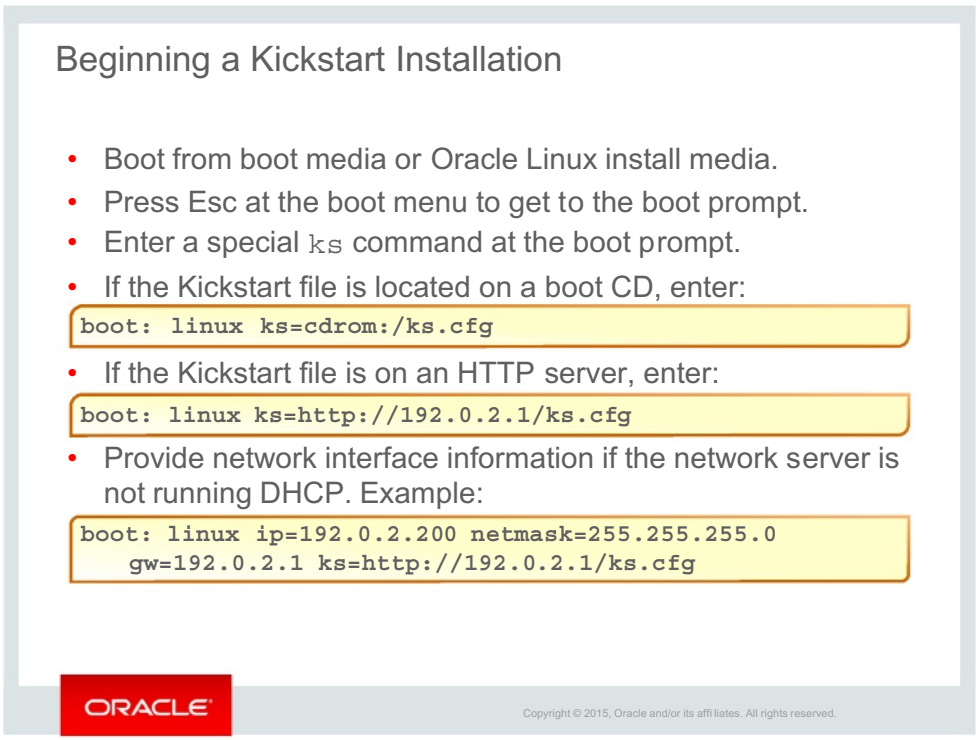
Install the `pykickstart` package:

```
# yum install pykickstart
```

Provide the Kickstart file name as an argument. For example:

```
# ksvalidator /root/ks.cfg
```

Output only occurs if syntax errors are found.



To begin a Kickstart installation, boot the system from boot media that you have made or from the Oracle Linux boot media. Press Esc at the boot menu to get to the boot prompt. Enter a special `ks` command at the boot prompt. The installation program looks for a Kickstart file if the `ks` command-line argument is passed to the kernel. For example, if the Kickstart file, `ks.cfg`, is located on a boot CD, boot the system with the CD in the drive and enter the following command at the boot prompt:

```
boot: linux ks=cdrom:/ks.cfg
```

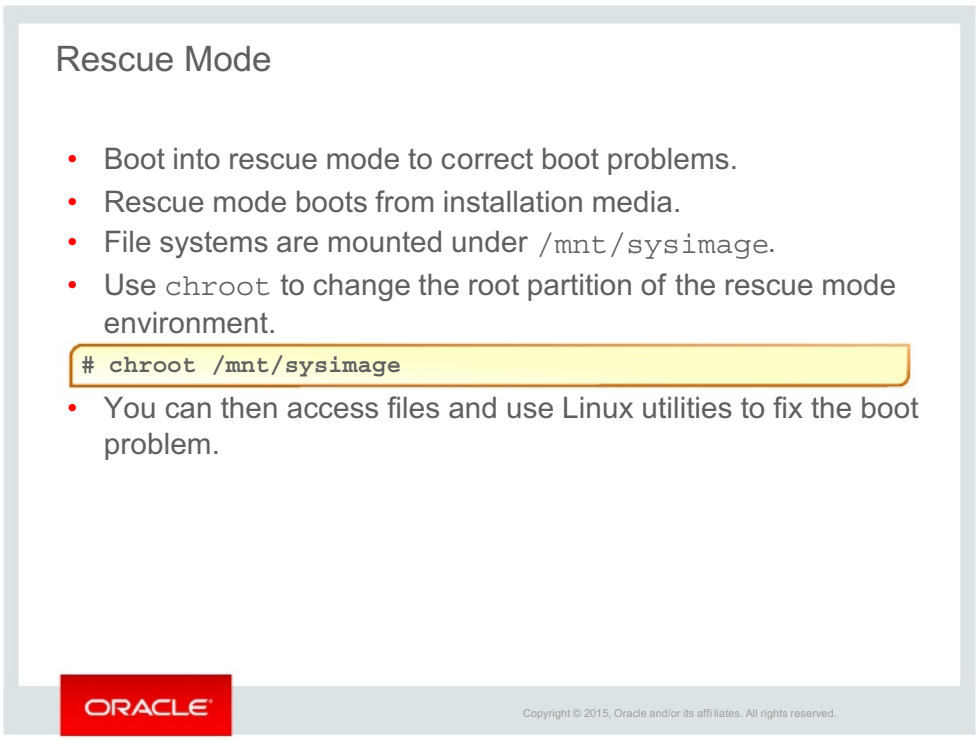
The following example is used when the Kickstart file is accessed from the network via HTTP. In the example, the HTTP server is `192.0.2.1` and the Kickstart file is located in the `/var/www/html/` directory on the HTTP server:

```
boot: linux ks=http://192.0.2.1/ks.cfg
```

This example assumes that DHCP is running on the `192.0.2.1` server. DHCP provides network interface configuration information for the system on which Oracle Linux is being installed. The system is then able to access the network server that is providing the installation tree.

If DHCP is not running on the installation server, you can include network interface configuration information in the boot command. Example:

```
boot: linux ip=192.0.2.200 netmask=255.255.255.0 gw=192.0.2.1
ks=http://192.0.2.1/ks.cfg
```



Rescue mode allows you to boot from the Oracle Linux installation media instead of booting from your system's hard drive. From rescue mode, you can access files on your hard drive and correct configuration errors, reinstall the boot loader, fix file system errors, or otherwise rescue your system. You might not be able to fix the boot problem, but at least you can get copies of important data files.

Rescue mode attempts to mount your file systems under `/mnt/sysimage`. The `/mnt/sysimage` is a temporary root partition, not the root partition of the file system used during normal operations. You can use the `chroot` command to change the root partition of the rescue mode environment to the root partition of your file system. Example:

```
# chroot /mnt/sysimage
```

You can then correct any errors in configuration files, run `fsck` to check and repair a file system, use `rpm` to install or upgrade software packages, and other commands to rescue your environment.

Quiz

The `%packages` section in the Kickstart file can contain package group names as well as individual package names.

a. True
b. False

Summary

In this lesson, you should have learned how to:

- Describe the Kickstart installation method
- Describe the Kickstart file
- Verify the Kickstart file
- Start a Kickstart installation
- Boot into rescue mode to correct boot problems

Practice 6: Overview

The practices cover the following topics:

- Performing a Kickstart installation
- Using rescue mode to fix a boot problem

7

Samba Services

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of Samba
- Describe Samba services and daemons
- Configure a Samba server
- Describe Samba server types
- Access Samba shares from a client

Introduction to Samba

- Samba:
  - Is an open source implementation of the Server Message Block (SMB) protocol
  - Allows Linux and Windows systems to share files and printers
- Samba packages included with Oracle Linux:
  - `samba`: SMB/CIFS server package
  - `samba-client`: Allows clients to access SMB/CIFS shares and printers
  - `samba-common`: Provides files necessary for both the server and client Samba packages
  - `samba-winbind`: Provides the `winbind` daemon and client tools
  - `samba-winbind-clients`: Provides the NSS library and PAM modules needed to communicate with `winbind`

Samba is an open-source implementation of the Server Message Block (SMB) protocol. It allows Linux to work with the Windows operating system, as both a server and a client. Samba shares Linux files and printers with Windows systems, and also gives Linux users access to files on Windows systems. Samba uses NetBIOS over TCP/IP (NetBT) protocols and does not need the NetBEUI (Microsoft Raw NetBIOS frame) protocol.

Several Samba packages are included with the Oracle Linux distribution:

- **`samba`:** Provides an SMB/Common Internet File System (CIFS) server that can be used to provide network services to SMB/CIFS clients
- **`samba-client`:** Provides some SMB/CIFS clients to complement the built-in SMB/CIFS file system in Linux. These clients allow access to SMB/CIFS shares and printing to SMB/CIFS printers.
- **`samba-common`:** Provides files necessary for both the server and client Samba packages
- **`samba-winbind`:** Provides the `winbind` daemon and client tools. `winbind` enables Linux membership in Windows domains and the use of Windows user and group accounts
- **`samba-winbind-clients`:** Provides the Network Security Services (NSS) library and Pluggable Authentication Modules (PAM) needed to communicate with `winbind`

Samba Daemons and Services

- The `samba` server package includes two daemons:
  - `smbd`: Provides file and print services for Samba clients
  - `nmbd`: NetBIOS nameserver
- The `samba-winbind` package includes one daemon:
  - `winbindd`: Resolves user/group information on Windows
- Each daemon has an associated service:
  - `smb`
  - `nmb`
  - `winbind`

Use `yum install <package_name>` to install the packages. The `samba` server package includes the following daemons and associated services:

- **`smbd`:** The server daemon that provides file-sharing and printing services to Windows clients. It is also responsible for user authentication, resource locking, and data sharing through the SMB protocol.
- **`nmbd`:** The NetBIOS nameserver daemon replies to name-service requests produced by SMB/CIFS in Windows-based systems. It also provides browsing support in the Windows Network Neighborhood view.

These daemons are controlled by their associated services, `smb` and `nmb`, for example:

```
# systemctl start smb
# systemctl start nmb
```

The `samba-winbind` package includes the `winbindd` daemon and associated service:

- **`winbindd`:** Resolves user and group information on a server running Windows and makes this information understandable by Linux

This daemon is controlled by the `winbind` service:

```
# systemctl start winbind
```

Samba Server Configuration

- The main configuration file for Samba is: `/etc/samba/smb.conf`
- The configuration file contains the following sections:
  - `[global]`: Defines global parameters
  - `[homes]`: Defines shares in the homes directory
  - `[printers]`: Defines printers
  - `[share name]`: Defines a share
- Example of share definition:

```
[tmp]
path = /tmp
writable = yes
guest ok = yes
```

The main configuration file for Samba is `/etc/samba/smb.conf`. This configuration file is divided into sections, each beginning with text surrounded by square brackets. With the exception of the `[global]` section, each section describes a shared resource, known as a "share." Typical sections are:

- **`[global]:`** Defines global parameters
- **`[homes]:`** Defines shares in the homes directory
- **`[printers]:`** Defines printers
- **`[share name]:`** Defines a share

Parameters within the section define the share attributes. Assuming that the global parameters are configured properly, the following example defines a share that gives any Windows user read-write permissions to the local `/tmp` directory:

```
[tmp]
comment = Insert a comment here
path = /tmp
writable = yes
guest ok = yes
```

Refer to the `smb.conf(5)` man page for a description of all the parameters that you can set in the configuration file. There are global parameters, security parameters, logging parameters, browser parameters, communication parameters, and share parameters. There are also several graphical user interfaces to configure and manage Samba. A list of these can be found at http://www.samba.org/samba/GUI/

`[homes]` Share

Samba provides this share to make it easy for users to share their Linux home directories with a Windows system. The following is an example:

```
[homes]
comment = Insert a comment here
browsable = no
writable = yes
```

These settings prevent users other than the owners from browsing home directories, while allowing logged-in owners full access.

Starting a Samba Server

To start a Samba server:

```
# systemctl start smb
```

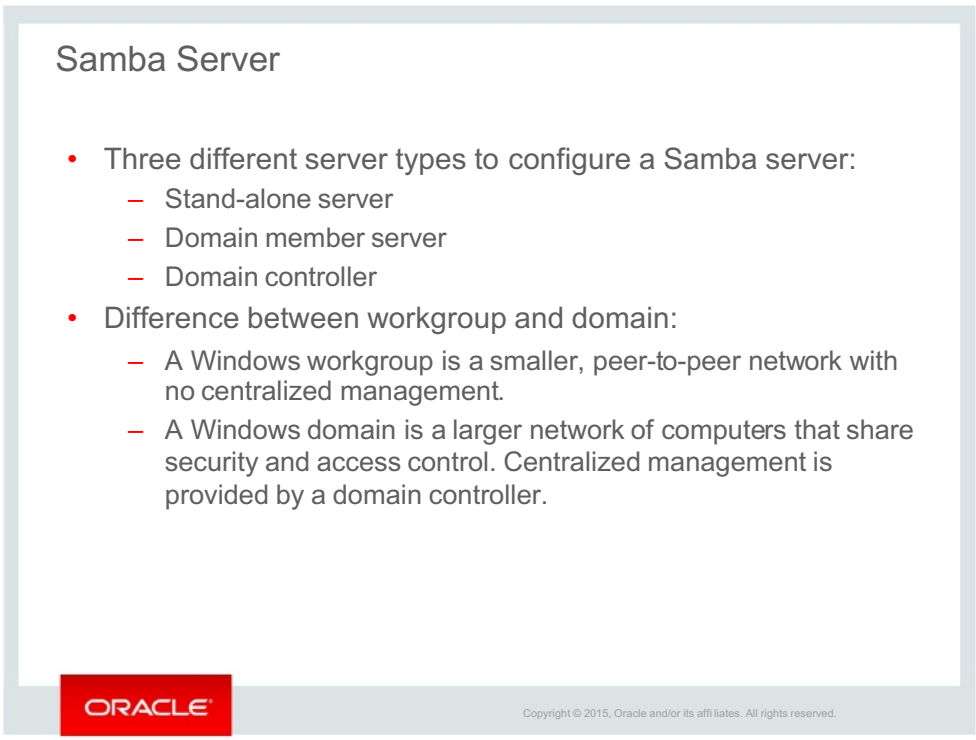
When making configuration changes to the `/etc/samba/smb.conf` file, issue a restart or reload:

```
# systemctl restart smb
# systemctl reload smb
```

The `reload` argument does not stop and start the `smb` service; it only reloads the configuration file.

Use the `systemctl` command to configure the service to start at boot time. Example:

```
# systemctl enable smb
```



There are three different server types to configure a Samba server:

- Stand-alone server
- Domain member server
- Domain controller

To understand the differences between the different server types, a brief introduction to Windows environments is necessary.

Windows Workgroups and Domains

Computers running Windows on a network belong to a workgroup or to a domain. Workgroup networks consist of a small number of computers when compared to a domain. Domains are best suited for corporate networks with many systems networked together.

A workgroup environment is a peer-to-peer network. Computers do not rely on each other for services. There is no centralized management. Each computer is sustainable on its own. Each computer has its own set of user accounts, its own access control, and its own resources. The systems can share resources, however, if configured to do so.

A domain is a trusted group of computers that share security and access control. Domains provide centralized management and security from a separate computer called a domain controller. Most modern Windows domains use Active Directory.

Samba Server Types

- Server type is configured in the `[global]` section of the `/etc/samba/smb.conf` file.
- A stand-alone server can be a workgroup server or a member of a workgroup.
- A domain member server logs in to a domain controller and is subject to the domain's security rules.
- A Samba server can be a domain controller in a Windows NT domain but not in an Active Directory domain.

Stand-Alone Server

A stand-alone Samba server can be a workgroup server or a member of a workgroup environment and does not participate in a Windows domain in any way. The following is an example of configuring the `[global]` directives in `/etc/samba/smb.conf` for a stand-alone server:

```
[global]
workgroup = workgroup_name
netbios name = netbios_name
security = share
```

The `security` parameter set to `share` indicates share-level security as opposed to user-level security. With share-level security, the server accepts only a password without an explicit username from the client. The server expects a password for each share, independent of the username. The use of share-level security is discouraged in favor of user-level security. There are four different ways to implement user-level security—user, server, domain, and ads—each of which is discussed in the "Authentication and Directory Services" lesson.

Domain Member Server

A domain member server is similar to a stand-alone server, but the server is logged in to a domain controller (either Windows or Samba) and is subject to the domain's security rules. An example of a domain member server would be a departmental server running Samba that has a machine account on the Primary Domain Controller (PDC). All of the department's clients still authenticate with the PDC, but the departmental server controls printer and network shares. To set up a domain member server, you must first join the domain or Active Directory by using the `net join` command before starting the `smb` service.

The following is an example of configuring `/etc/samba/smb.conf` to implement an Active Directory domain member server. Samba authenticates users for services being run locally, but is also a client of the Active Directory.

```
[global]
realm = EXAMPLE.COM
security = ADS
password server = kerberos.example.com
```

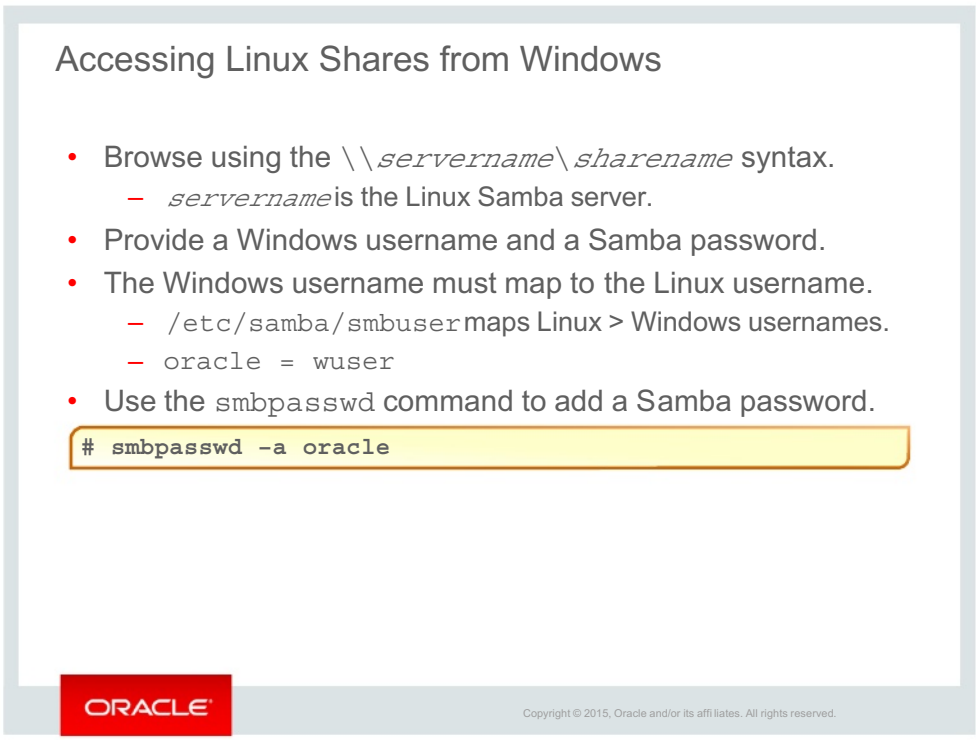
The `realm` directive identifies the Kerberos realm and must be capitalized. Kerberos is an authentication protocol that allows nodes communicating over a nonsecure network to prove their identity to one another. Windows requires Kerberos for Active Directory authentication. The `password server` directive is required only if Active Directory and Kerberos are running on different servers.

The following is an example of configuring `/etc/samba/smb.conf` to implement a Windows NT4-based domain member server. NT4-based domains do not use Kerberos in their authentication method.

```
[global]
workgroup = workgroup_name
netbios name = netbios_name
security = domain
```

Domain Controller

A Samba server cannot be configured as an Active Directory Primary Domain Controller (PDC) but it can be configured to appear as a Windows NT4-style domain controller. For Windows NT, a domain controller is similar to a Network Information Service (NIS) server in a Linux environment. They both host user and group information databases and other services. Domain controllers are mainly used for security, including the authentication of users accessing domain resources. Authentication services are discussed in the lesson titled "Authentication and Directory Services."



To access a share on a Linux Samba server from Windows, open My Computer or Explorer and enter the host name of the Samba server and the share name in the following format:

servername**sharename

If you enter *servername*, Windows displays the directories that the Linux system is sharing. You can also map a network drive to a share name by using the same syntax.

`smbusers` File

For a Windows user to access a Samba share on a Linux system, the user must provide a Windows username and a Samba password. The Windows username must be the same as the Linux username or must map to a Linux username. Samba stores these username maps in the `/etc/samba/smbusers` file. Users with the same username on Linux and Windows do not need an entry in this file, but they still need a Samba password.

The `/etc/samba/smbusers` file has two default entries:

```
root = administrator admin
nobody = guest pcguest smbguest
```

The first entry maps the Linux `root` user to the `administrator` and `admin` users in Windows. The second entry maps the Linux user `nobody` to three Windows usernames.

To map the Windows username of `wuser` to the Linux username of `oracle`, add the following entry to `/etc/samba/smbusers`:

```
oracle = wuser
```

Samba uses Samba passwords, not Linux passwords, to authenticate users. Add a password for the `oracle` user with the following command:

```
# smbpasswd -a oracle
New SMB password:
Retype new SMB password:
Added user oracle.
```

Accessing Windows Shares from Linux

- Utilities to query Samba servers:
  - `findsmb`
  - `smbtree`
- From GNOME and KDE desktops file managers:
  - Enter `smb:` in the location bar.
- Use the `smbclient` utility to connect to a Windows share from the command line:
  - `smbclient //<servername>/<sharename> [-U <username>]`
- Use the `mount.cifs` command to mount a Samba share.
- The `mount` command requires the `cifs-utils` package.

Use the `findsmb` command to query a subnet for Samba servers. The command displays the IP address, NetBIOS name, workgroup, operating system, and version for each server found.

You can also use the `smbtree` command, which is a text-based SMB network browser. It displays a hierarchy diagram with all the known domains, the servers in those domains, and the shares on the servers.

The GNOME and KDE desktops provide browser-based file managers to view Windows shares on the network. Enter `smb:` in the location bar of the file managers to browse shares.

Use the `smbclient` utility to connect to a Windows share from the command line. The format is as follows:

```
smbclient //<servername>/<sharename> [-U <username>]
```

The `smb:\>` prompt is displayed after successfully logging in. Type `help` to display a list of commands. Type `exit` to exit `smbclient`.

To mount Samba shares, install the `cifs-utils` package:

```
# yum install cifs-utils
```

Use the `mount.cifs` command with the following format to mount Samba shares:

```
mount.cifs //<servername>/<sharename> /mount-point -o
username=<username>,password=<password>
```

Samba Utilities

Samba packages include several command-line utilities like:

- `net`: Works like the `net` utility for Windows and MS-DOS
- `nmblookup`: Resolves NetBIOS names to IP addresses
- `smbstatus`: Displays the status of Samba server connections
- `smbtar`: Backs up and restores Windows-based share files and directories to a local Linux tape archive
- `testparm`: Checks the syntax of the `/etc/samba/smb.conf` file
- `wbinfo`: Displays information from the `winbindd` daemon
- `smbget`: A `wget`-like utility for downloading files over SMB

The following list summarizes the command-line utilities included with the Samba packages. Use the `which utility` command to display the absolute path name of the command. Include the output as an argument to the `rpm -qf` command to display which Samba package provides the command.

Example:

```
# which smbtree
/bin/smbtree
# rpm -qf /bin/smbtree
samba-client-<version>
```

The Samba command-line utilities include the following:

- **`smbtree`:** Is a text-based SMB network browser
- **`smbclient`:** Is an FTP-like client to access SMB/CIFS resources on servers
- **`smbpasswd`:** Is used to add or modify a user's SMB password
- **`smbcacls`:** Modifies Windows ACLs on files and directories shared by a Samba server or a Windows server
- **`nmblookup`:** Is used to query NetBIOS names and map them to IP addresses

- **net:** Is a tool for the administration of Samba and remote CIFS servers. It is designed to work like the `net` utility used for Windows and MS-DOS. The syntax is:

  `net <protocol> [options]`

  The `<protocol>` argument specifies the protocol to use when executing a command. Specify the type of server connection by using `ads` (Active Directory), `rap` (Win9x/NT3), or `rpc` (Windows NT4/2000/2003/2008). If the protocol argument is not specified, `net` automatically tries to identify it. Use `net -h` for online help and usage examples.
- **rpcclient**: Is a tool for executing client-side Microsoft RPCs functions
- **smbcontrol:** Sends control messages to running `smbd`, `nmbd`, or `winbindd` daemons
- **smbspool:** Sends a print file to an SMB printer
- **smbstatus:** Displays the status of current connections to a Samba server
- **smbtar:** Backs up and restores Windows-based share files and directories to a local Linux tape archive
- **testparm:** Checks the syntax of the `/etc/samba/smb.conf` file
- **wbinfo:** Displays information from the `winbindd` daemon (The `winbindd` daemon must be running.)
- **smbcquotas:** Manipulates quotas on NT file system (NTFS) SMB file shares
- **smbget:** Is a `wget`-like utility for downloading files over SMB

Quiz

Which of the following statements are true?

a. Samba allows Linux clients to mount exported file systems on remote Windows systems.
b. Samba allows Windows clients to mount exported file systems on remote Linux systems.
c. Samba allows Linux clients to mount exported file systems on remote Linux systems.

Summary

In this lesson, you should have learned how to:

- Describe the purpose of Samba
- Describe Samba services and daemons
- Configure a Samba server
- Describe Samba server types
- Access Samba shares from a client

Practice 7: Overview

The practices for this lesson cover the following:

- Configuring a Samba server
- Accessing Samba shares from a Samba client host
- Accessing a Linux Samba share from a Windows system

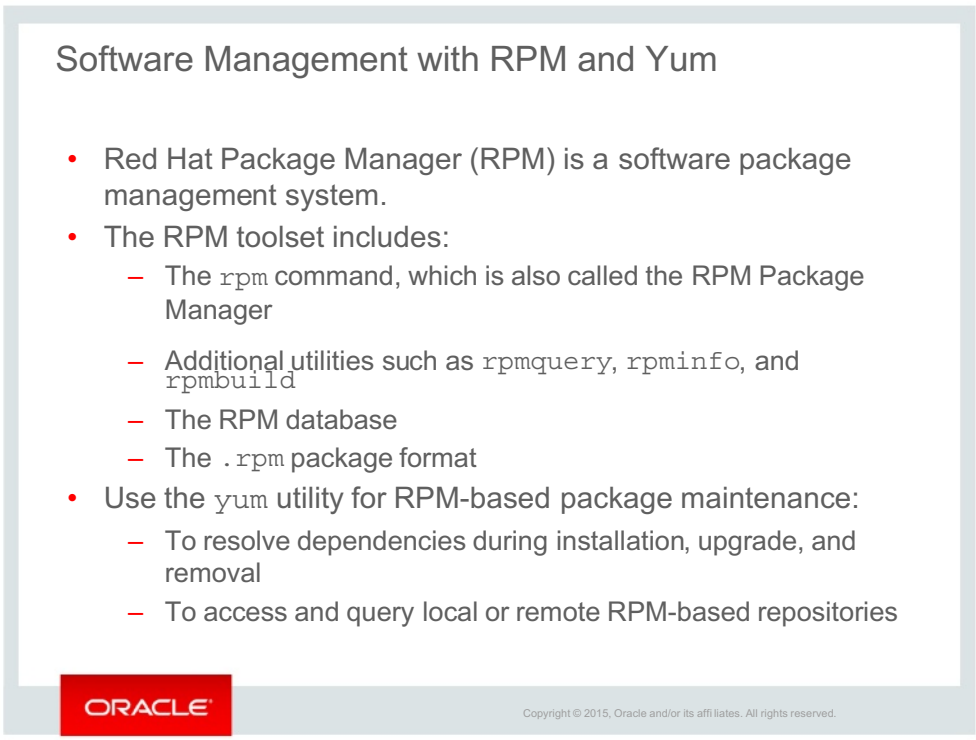
8

Advanced Software Package Management

Objectives

After completing this lesson, you should be able to:

- Describe the contents of an RPM package
- Perform a binary RPM build
- Use the tools to perform package maintenance with Yum
- Manage the Yum cache and Yum history
- Install and use Yum plug-ins
- Describe and use the programs offered by PackageKit



RPM Package Management

You can use the RPM package management system for packaging software in the Linux environment and manage these packages. The `rpm` command is the most widely used command for RPM package management.

With the `rpm` command, you can:

- Install, upgrade, and remove packages
- List and query installed packages. For example, use the `rpm -ql <package name>` command to list the files that make up a package.
- Manage the RPM database

In addition to the `rpm` command, you can install additional commands to manage your RPM-based software. Three of these commands are listed with an example for each command:

- The `rpminfo` command provides information about installed packages. The following example lists executable (`-e`) files included in the `bash` package:

```
# rpminfo -e bash
bash-4.2.46-12.el7.x86_64
        /usr/bin/bash       PIC
```

- The `rpmbuild` command builds source or binary RPM packages. The following example builds a binary package (`-bb`) in verbose (`-v`) mode:

```
# rpmbuild -bb -v myspecfile.spec
```

- The `rpmquery` command displays information about packages, the RPM database or other package-related items. The following example displays files in a package:

```
$ rpmquery -l xorg-x11-server-Xorg
/etc/X11/xorg.conf.d
/etc/pam.d/xserver
/etc/security/console.apps/xserver
/usr/bin/X
...
/usr/share/man/man5/xorg.conf.5.gz
/usr/share/man/man5/xorg.conf.d.5.gz
```

Note that the following `rpm` command produces the same output as the preceding one:

```
$ rpm -ql xorg-x11-server-Xorg
/etc/X11/xorg.conf.d
/etc/pam.d/xserver
/etc/security/console.apps/xserver
/usr/bin/X
...
/usr/share/man/man5/xorg.conf.5.gz
/usr/share/man/man5/xorg.conf.d.5.gz
```

Yum Package Management

Yum is also a package management system. It offers more functionality than is available with the RPM-based tools like the `rpm` command. Whether you use the `rpm` command or the `yum` command, the packages are the same. They are built by using the RPM format and are distributed in files with the `.rpm` identifier. Yum functionality is discussed later in this lesson.

RPM Packages

- An RPM binary package file (for example, `bash-4.1.2-15.el6_4.x86_64.rpm`) contains programs, configuration files, documentation.

```
# yum install bash
Loaded plugins: aliases, langpacks
Setting up Install Process
Package bash-4.2.46-12.el7.x86_64 already installed ...
Nothing to do
```

- An RPM source package file (for example, `bash-4.2.46-12.el7.src.rpm`) contains the source code and all necessary files to re-create the binary package file.

```
# yumdownloader --source bash
Loaded plugins: aliases, langpacks
bash-4.2.46-12.el7.src.rpm                  ...
```

RPM packages, either binary or source, are built into a file with the `.rpm` extension. All RPM files have the same format, though this format has changed over the years. The process to build RPM packages is similar for binary and source packages.

When you maintain the software in your RPM-based Linux environment, you install or upgrade the software from RPM binary packages. The slide provides an example of using the `yum` command to install the `bash` package. In this example, the `bash` package is already installed.

The slide also provides an example of using the `yumdownloader` command to download the RPM source packages for `bash`. You can install the source RPM package if you intend to modify the software. After modifying the software, you can rebuild the source RPM and the binary RPM package. If you install the binary RPM package after rebuilding it, you install the modified program(s) and associated files on your Linux system.

The Binary RPM Build Process

1. As the `root` user, install the RPM tools package called `rpmdevtools`.
2. As a user other than `root`:
    1. Create a directory structure for the build.
    2. Add the source files to a compressed (`.tar.gz`) file and store them in the `SOURCES` directory.
    3. Create the `spec` file.
    4. Build the binary package with the `rpmbuild -bb` command.
3. As the `root` user, install the package and verify the installation.
4. Optional: Upload the package to a repository.

The steps to build a binary RPM package are shown in the slide. Install the `rpmdevtools` package, which installs the `rpm-build` package as a dependency. These two packages provide the basic tools necessary to build binary and source RPM packages. These tools include the following utilities:

- `rpmdev-setuptree`: Creates the directory structure for the package build
- `rpmdev-newspec`: Creates a skeleton `spec` file
- `rpmbuild`: Builds the binary RPM (also used to create source RPM packages)

Other tools are installed as well. Use the following commands to list all the executables that are installed as part of the `rpmdevtools` package:

```
# rpm -ql rpmdevtools | grep bin
/usr/bin/annotate-output
...
/usr/bin/rpmdev-newspec
...
/usr/bin/rpmdev-setuptree
...
```

BUILD Directory Structure

- Use the `rpmdev-setuptree` command to build the directory structure for the RPM build process.
- This command creates the following directories:
  - `BUILD`
  - `RPMS`
  - `SOURCES`
  - `SPECS`
  - `SRPMS`

The directories listed in the slide are shown in alphabetical order.

In the following notes, the directories are listed as they are used during the binary (or source) RPM build process:

SPECS

This is the directory where you store the `spec` file for building a source or binary package.

SOURCES

This is the location where you store the files to build your package. The files can be source code or binary files.

BUILD

The files in the `SOURCES` directory are copied or extracted to this directory before building the software. This directory is used as temporary space to compile the software.

RPMS

The output of the RPM build process is an RPM package. If the build is a binary build, this package is stored in the `RPMS` directory.

SRPMS

This is the same as the `RPMS` directory except that it is used to store source RPMs.

spec File to Build a Binary RPM Package

- The spec file describes the package and lists the steps to build the software in the package.
- It contains the following sections:
  - Header: Describes the package with a collection of tags
  - %prep: Prepares files for the build
  - %build: Builds the software
  - %install: Copies files to their installation location
  - %clean: Cleans the build directory tree
  - %files: Identifies the files to be packaged

Header

This section describes the package by using tags and directives.

For example, the following tags describe a package: Name, Arch, Version, Release, URL, and License. The %description directive describes the usage for the package. After the package is built, the tags and directive information become part of the package.

You can display the tags and other information in an existing package by using the yum info *<package name>* command:

```
# yum info bash
Loaded plugins: aliases, langpacks
Installed Packages
Name        : bash
Arch        : x86_64
Version     : 4.2.46
Release     : 12.el7
....
Description : The GNU Bourne Again shell (Bash) is a shell ...
...
```

Each section following the header information is a step in the build process. Any command, script, macro or directive in a section is executed like a script when that step is executed.

Before each step is executed, several environment variables are set. For example, the value for the `RPM_BUILD_DIR` variable is set.

`%prep`

During the execution of this step, the source files are unpacked into the build directory. If present, patches are applied. The `%setup` macro is responsible for unpacking the source files.

`%build`

There are no special macros for this section. Generally, you specify one or more `make` commands in this section to build the software.

`%install`

The role of this section is to install the newly built software. This means that the files that make up the package along with their directory location are copied into a directory structure. After the files are copied, the build reads the list of files in the `%files` section and creates the binary RPM package.

`%clean`

At this step, the packaging is already done. This macro cleans the directory tree where the software is installed (default is variable `RPM_BUILD_ROOT`) or any directory specified in this section.

`%files`

This section lists the files that are to be part of the final RPM. In this section, you can use macros to set file and directory permissions.

As stated in the description of the sections in the `spec` file, you can use macros and directives to perform specific steps in each section.

Example:

- `%setup` macro in the `%prep` section unpacks the source files.
- `%config` directive in the `%files` section labels files as configuration files.

You can find more information about the tags, macros, and directives in the RPM-based `spec` file at this location: http://www.rpm.org/max-rpm/

spec File: Example

```
Name:          hello
Version:       1.0
Release:       1%{?dist)
Summary:       hello program

License:       GPL

#URL:
Source0:       hello-1.0.tar.gz

#BuildRequires:
#Requires:

%description
A program to display Hello
    World

%prep
%setup -q
```

```
%build


%install
rm -rf $RPM_BUILD_ROOT
#%make_install

install -d
      $RPM_BUILD_ROOT/usr/local/bin
install hello
      $RPM_BUILD_ROOT/usr/local/bin/h
    ello


%files
/usr/local/bin/hello


%changelog
```

Use the `rpmdev-newspec` command to create a skeleton`spec` file. The following example creates the `hello.spec` file in the `SPECS` directory:

```
# rpmdev-newspec SPECS/hello.spec
```

From the contents of the`hello.spec` file shown in the slide, you can gather the following information:

- Both the package and the program in the package are called `hello`.
- This is version 1.0 of the software being packaged (`Version: 1.0`).
- This is version 1 of the package itself with the distribution appended (`Release: 1.el7.x86_64`).
- There is no step for the `%build` section. The build process goes from the `%prep` section to the `%install` section without any build command or script. Generally, this section contains build instructions.
- There is only one file in the final package, the `hello` program, which is installed into the `/usr/local/bin` directory when the package is installed.

Managing RPM-Based Software with Yum

- Using Yum greatly simplifies package maintenance in your Linux environment.
- Yum includes:
  - The `yum` command
  - Several utilities such as `yum-config-manager`, `repoquery`, and `yumdownloader`
  - The ability to create, query, and control access to repositories
  - Plug-ins that extend Yum's functionality
  - Caching to increase performance for Yum operations

Yum tools, including the `yum` command, provide more services and functionality than is available with the `rpm` command and other RPM-based tools.

With Yum tools and plug-ins, you can:

- List software packages, both installed and available, in local or remote repositories
- Check for package dependencies (packages required to install a package)
- Check for dependent software (packages that depend on another package)
- Create new repositories and enable or disable access to existing repositories
- Speed up package installation by using cached information (Yum cache)
- Extend Yum's functionality with plug-ins such as the `aliases` plug-in (to create and view aliases for Yum commands)
- Use package management GUI tools such as PackageKit. PackageKit uses Yum tools.

PackageKit is discussed later in this lesson.

Note that when creating packages, either binary or source RPMs, you use RPM-type tools such as `rpmbuild` and `rpmdev-setuptree`. These commands were discussed in the RPM Packages topic earlier in this lesson.

Yum Cache

- `yum` stores temporary files in the `/var/cache/yum` directory.
- Temporary package files are deleted after a `yum` operation completes successfully.
- You can enable caching to retain package files in cache directories:
  - These packages can be reused when there is no network connection to repositories.
- Clean information in the cache with:

```
# yum clean metadata
# yum clean headers
# yum clean packages
# yum clean all
```

For some operations (for example, a `yum install` operation), Yum downloads the packages to install into the Yum cache. The cached packages are located in a subdirectory structure from `/var/cache/yum` that reflects the architecture, the distribution release, and the repository from where the packages were downloaded.

After successful installation, the packages are deleted from the cache. To retain the cached packages, change the `keepcache` setting to `1` in the `/etc/yum.conf` file as follows:

```
keepcache = 1
```

You can also change the location for the cache by modifying the `cachedir` parameter, which by default is set to:

```
cachedir=/var/cache/yum/$basearch/$releasever
```

Cleaning the Yum Cache

Clean the Yum cache to reclaim disk space or to clear errors due to corrupted metadata files.

To remove cached packages only, use:

```
# yum clean packages
```

To delete metadata for each enabled repository, use the following command:

```
# yum clean metadata
```

To delete package headers, use the following command:

```
# yum clean headers
```

To clean all cached information, use the following command:

```
# yum clean all
```

If you get the message “Metadata file does not match checksum” during a Yum operation, clearing the metadata from the cache might not help. In this case, adding the following line to `/etc/yum.conf` resolves the problem:

```
http_caching=none
```

Yum History

- Yum keeps detailed information about transactions in Yum history.
  - Each transaction is assigned an ID.
- Yum history is stored in `/var/lib/yum/history/`.
- To display the transactions:

```
# yum history list
# yum history info <transaction ID>
# yum history package-list <package name>
```

- To undo a transaction:

```
# yum history undo <transaction ID|last>
```

- To start a new history db:

```
# yum history new
```

ORACLE

The following command displays the last 20 transactions in Yum history:

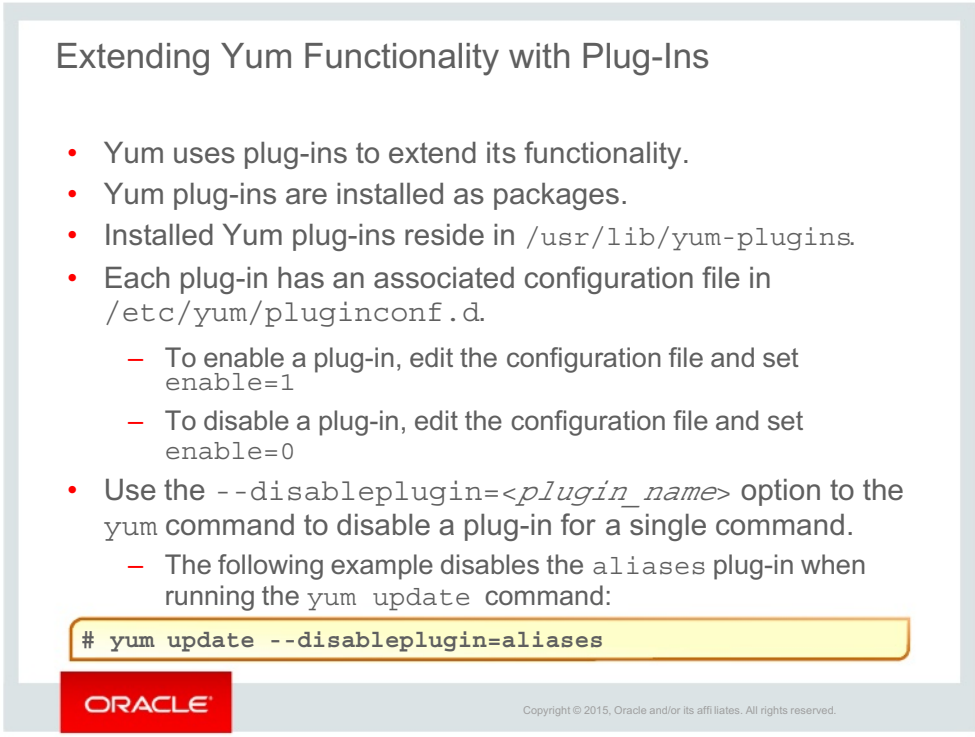
```
# yum history list
Loaded plugins: aliases, langpacks
ID     | Login user         | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     7 | root <root>        | 2014-01-28 03:13 | Erase          |    2
     6 | root <root>        | 2014-01-28 03:03 | Install        |    2
     5 | System <unset>     | 2014-01-27 09:31 | Update         |    1 <
     4 | root <root>        | 2014-01-27 08:10 | Install        |    4 >
     3 | root <root>        | 2014-01-27 07:46 | Install        |    1
     2 | root <root>        | 2014-01-27 07:45 | Update         |    2
     1 | System <unset>     | 2014-01-27 06:57 | Install        | 1131
history list
```

To obtain detailed information for transaction ID 6, which installed the rpmdevtools package:

```
# yum history info 6
Loaded plugins: aliases, langpacks
Transaction ID : 6
Begin time     : ...
...
Return-Code    : Success
Command Line   : install rpmdevtools
Transaction performed with:
...
```

In this example, transaction ID 6 installed the rpmdevtools package. Transaction ID 7 uninstalled the rpmdevtools package. The following command performs transaction ID 6 again, which installs rpmdevtools:

```
# yum history redo 6
Loaded plugins: aliases, langpacks
...
Repeating transaction 6...
...
Installing:
 rpmdevtools ...
...
```



Yum uses plug-ins to extend its functionality.

Each installed plug-in has a configuration file located in the `/etc/yum/pluginconf.d` directory. For example, the configuration file for the `langpacks` plug-in contains:

```
# cat /etc/yum/pluginconf.d/langpacks.conf
[main]
enabled=1
langpack_locales = en_US.UTF-8
```

By default, all plug-ins are enabled in `/etc/yum.conf` with the following statement:

```
plugins=1
```

Do not disable plug-ins globally from `/etc/yum.conf` by changing `plugins=1` to `0`. This action can cause problems with some Yum services. If you want to disable a plug-in without uninstalling it, you can:

- Disable the plug-in from its configuration file in `/etc/yum/pluginconf.d`.
- Disable the plug-in for a single operation by appending `--disableplugin=<plug-in name>` to the `yum` command.

Yum plug-ins are Python scripts or programs that are stored in `/usr/lib/yum-plugins` when they are installed:

```
# ls -l /usr/lib/yum-plugins
-rw-r--r-- ... langpacks.py
-rw-r--r-- ... langpacks.pyc
-rw-r--r-- ... langpacks.pyo
```

Each plug-in in this example has three files associated with it: a `.py`, a `.pyc`, and a `.pyo` file. These files are part of the Python application. For more information about these Python file extensions, see http://docs.python.org/release/1.5.1p1/tut/node43.html.

Popular Yum Plug-Ins

- `kabi`: Checks if newly installed kernel module packages conform with the kernel Application Binary Interface (kABI)
- `aliases`: Allows you to create and view aliases for Yum commands. It includes the `/etc/yum/aliases.conf` file, which contains many predefined Yum command aliases.
- `changelog`: Allows you to view package change logs before and after updating
- `tmprepo`: Allows you to use a temporary Yum repository. It ensures that these repositories are safe and does not allow GPG checking to be disabled.

kabi

The package name is `kabi-yum-plugin`. This plug-in checks newly installed kernel module packages to ensure that they conform with kABI.

aliases

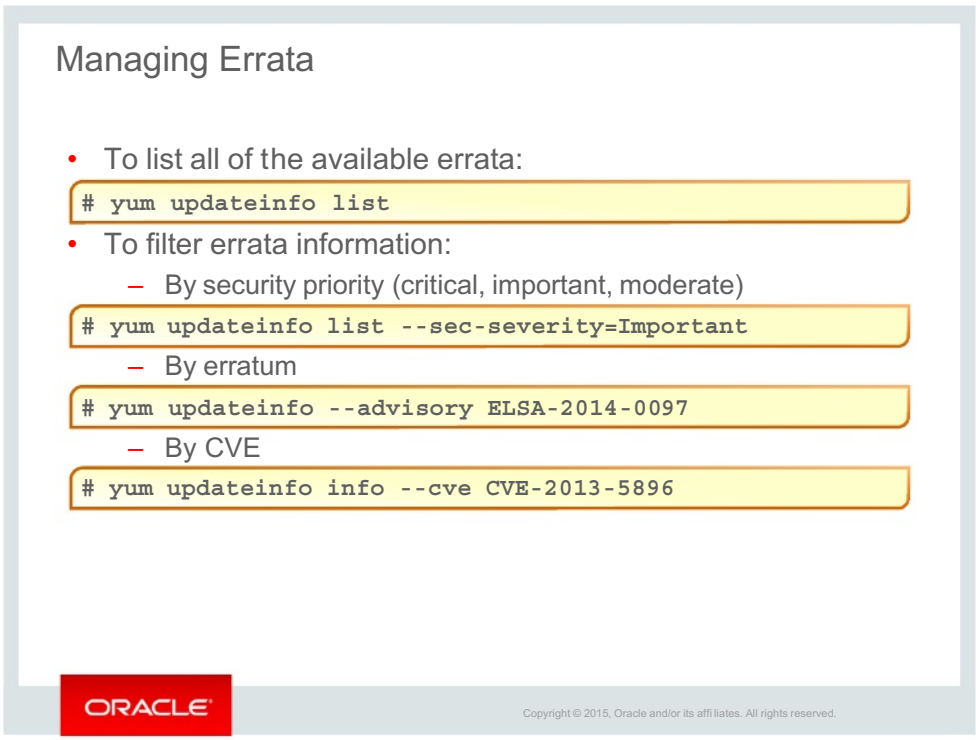
The package name is `yum-plugin-aliases`. This plug-in allows you to create and view aliases for Yum commands. This plug-in includes the `/etc/yum/aliases.conf` file, which contains many predefined Yum command aliases.

changelog

The package name is `yum-plugin-changelog`. This plug-in allows you to view package change logs before and after updating. Yum invokes the plug-in if either the `--changelog` option or the `changelog` command is used with the `yum` command.

tmprepo

The package name is `yum-plugin-tmprepo`. This plug-in provides the `--tmprepo=<URL:repo_file>` option to the `yum` command. The repository file is downloaded from the URL and enabled for a single Yum transaction. This plug-in attempts to ensure that temporary repositories are safe to use. The plug-in does not allow GNU Privacy Guard (GPG) checking to be disabled by default.



Use the `yum updateinfo` command to view information about errata (security, bug fixes, and enhancements), but also to update your Linux system by using options that act like filters to select only certain updates from the errata. In previous versions of Oracle Linux, this capability was provided by the `yum-security` plug-in. Oracle Linux 7 includes an updated version of Yum and the `yum-security` plug-in has been integrated into `yum`.

Obtaining Errata Information

To list all of the available errata:

```
# yum updateinfo list
...
ELSA-2015-0672 Moderate/Sec.  bind-libs-32:9.9.4-18.el7_1.1...
...
ELBA-2015-0741 bugfix         binutils-2.23.52.0.1-30.el7_1.1...
...
ELEA-2015-0969 enhancement    crash-7.0.9-5.el7_1.x86_64
...
ELSA-2015-0265 Critical/Sec.  firefox-31.5.0-2.0.1.el7_0...
...
```

This list contains all of the errata by errata ID. Errata include security patches, bug fixes, and feature enhancements. Security fixes are listed by priority: Critical, important, or moderate.

To obtain only the security errata with priority set to Important:

```
# yum updateinfo list --sec-severity=Important
ELSA-2014-0097 Important/Sec. java-1.6.0-openjdk-1:1.6.0.0-3...
ELSA-2014-0097 Important/Sec. java-1.6.0-openjdk-devel-1:1.6...
ELSA-2013-1801 Important/Sec. kernel-2.6.32-431.1.2.el6.x86_64
ELSA-2013-1801 Important/Sec. kernel-2.6.32-431.5.1.el6.x86_64
...
```

To list detailed information for a particular erratum from the previous list:

```
# yum updateinfo info --advisory ELSA-2014-0097
==============================================================
   java-1.6.0-openjdk security update
==============================================================
  Update ID : ELSA-2014-0097
     Release : Oracle Linux 6
        Type : security
      Status : final
      Issued : 2014-01-27
        CVEs : CVE-2013-5878
             : CVE-2013-5884
             ...
             : CVE-2014-0423
             : CVE-2014-0428
Description : [1:1.6.0.1-3.1.13.0]
             : - updated to icedtea 1.13.1
             :  -
             :    http://blog.fuseyism.com/index.php/2014/01/...
             ...
    Severity : Important
updateinfo info done
```

To obtain information for a particular Common Vulnerabilities and Exposures (CVE):

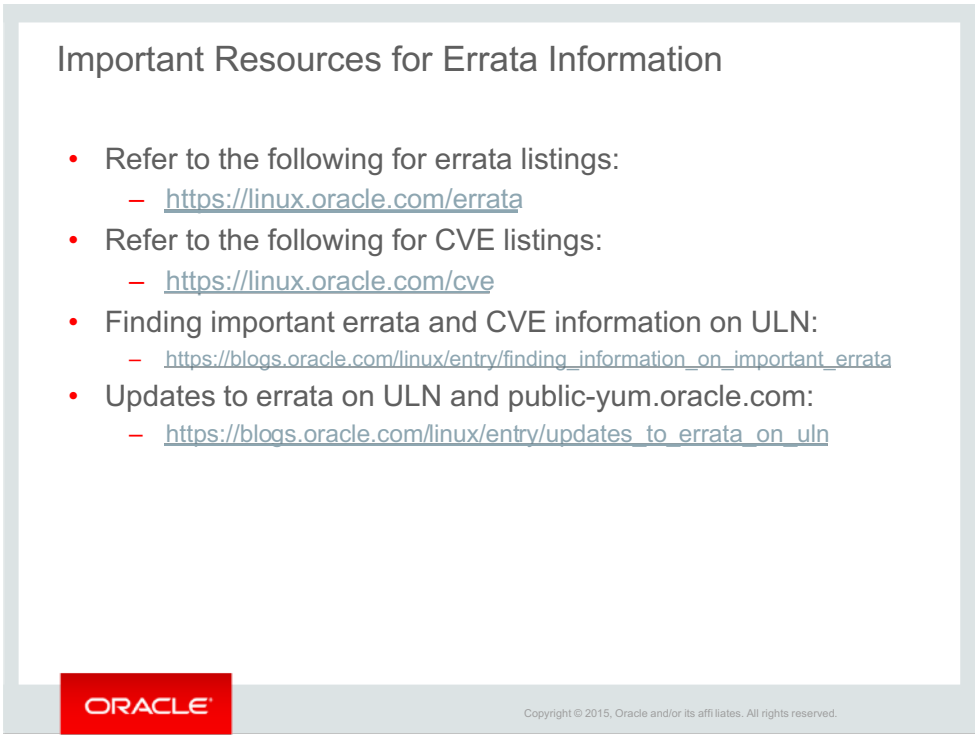
```
# yum updateinfo info --cve CVE-2013-5896
```

To update all packages for which security errata exist to the latest version of the packages:

```
# yum --security update
```

To update all packages for which security errata exist to the version that contains the security fix, ignoring newer releases of the packages:

```
# yum --security update-minimal
```



Refer to the following for errata listings: https://linux.oracle.com/errata

Using this link, you can view all errata releases available, listed by type, severity, advisory, summary, and release date. In addition, you are also able to filter this list by release and/or type (Bug, Security, or Enhancement).

Refer to the following for CVE listings: https://linux.oracle.com/cve

Using this link, you can find information on security errata by CVE identifier.

Read These Blogs

The following blog discusses finding important errata and CVE information on ULN: https://blogs.oracle.com/linux/entry/finding_information_on_important_errata

The following blog discusses updates to errata on ULN and public-yum.oracle.com: https://blogs.oracle.com/linux/entry/updates_to_errata_on_uln

PackageKit Software Package Manager GUI

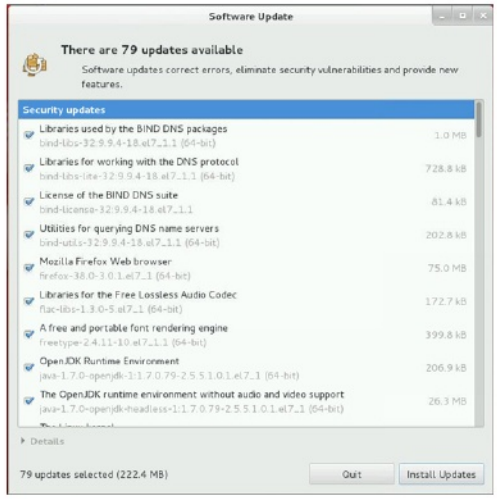
- PackageKit is designed to facilitate package management.
- It interfaces with a GUI front end. Examples:
  - GNOME (gnome-packagekit)
  - KDE (KPackageKit)
- The package operations are carried out by a back-end package management:
  - Yum (for Oracle Linux and Red Hat)
  - APT for Debian systems

The PackageKit program provides a graphical interface for package management systems for different Linux distributions. PackageKit for Oracle Linux and Red Hat distributions uses a Yum back end to perform operations on packages, such as installing, updating, and removing packages. The front-end graphical interface is GNOME, and this support is provided by the `gnome-packagekit` package and its dependencies.

Using PackageKit Software Update

Launch Software Update through one of the following methods:

- Select Software Updates from the Administration menu.
- Run `gpk-update-viewer` from the command line.



PackageKit provides a graphical tool to add or remove software and another tool to update your system.

For Oracle Linux users, installing updates with the PackageKit program is equivalent to installing fixes for bugs and/or security issues or enhancements announced in the errata and made available in the `_latest` repository for your particular Oracle Linux release. Only the updates for the currently installed software packages are shown.

You can use the Software Update and the Software programs as a nonprivileged user. When requesting to add/update/remove packages, you are prompted for the `root` password.

PackageKit Commands: Summary

- Commands to launch PackageKit graphical programs:
  - `gpk-application` → Add/Remove Software
  - `gpk-update-viewer` → Software Update
  - `gpk-log` → Software Log Viewer
  - `gpk-prefs` → Software Update Preferences
  - `gpk-distro-upgrade` → Upgrade Distribution Version
  - `gpk-install-catalog` → Install Package Catalogs
- Commands that interface with PackageKit:
  - `pkcon` → CLI client for PackageKit
  - `pkmon` → Program to monitor PackageKit operations
  - `pkgenpack` → Program to generate service packs with selected packages and their dependencies

The `gpk-update-viewer` launches the Software Update graphical program as discussed in the previous slide. Similarly, the `gpk-application` program launches the Software graphical program to install and remove packages. The `gpk-log` command allows you to view the software log. The `gpk-prefs` command allows you to change preferences for keeping your system up to date.

Following is a list of all the available "`gpk`" commands:

```
# ls /bin/gpk-*
gpk-application       gpk-install-local-file    gpk-log
gpk-dbus-service      gpk-install-mime-type     gpk-prefs
gpk-distro-upgrade    gpk-install-package-name  gpk-service-pack
gpk-install-catalog   gpk-install-provide-file  gpk-update-viewer
```

There are man pages for each of the "`gpk`" commands.

Similarly, you can view the available "`pk`" commands.

```
# ls /bin/pk*
pkcon     pkgenpack     pkmon     pkcheck     pkexec
...
```

Quiz

Which statements are true about Yum plug-ins?

a. Yum plug-ins extend the functionality of Yum.
b. New plug-ins are automatically included in the latest release of the Yum package.
c. Each plug-in has its own configuration file.
d. By default, all Yum plug-ins are enabled in `/etc/yum.conf`.

Quiz

Which of these commands can be used to clean information in the Yum cache?

a. `yum clean metadata`
b. `yum clean cache`
c. `yum clean packages`
d. `yum delete cache`

Summary

In this lesson, you should have learned how to:

- Describe the contents of an RPM package
- Perform a binary RPM build
- Use the tools to perform package maintenance with Yum
- Manage the Yum cache and Yum history
- Install and use Yum plug-ins
- Describe and use the programs offered by PackageKit

Practice 8: Overview

The practices for this lesson cover the following:

- Exploring the host04 VM
- Managing Yum plug-ins
- Using Yum utilities to manage errata and to download software
- Creating a binary RPM package
- Managing software updates with PackageKit's Software Update program
- Working with Yum history and Yum cache

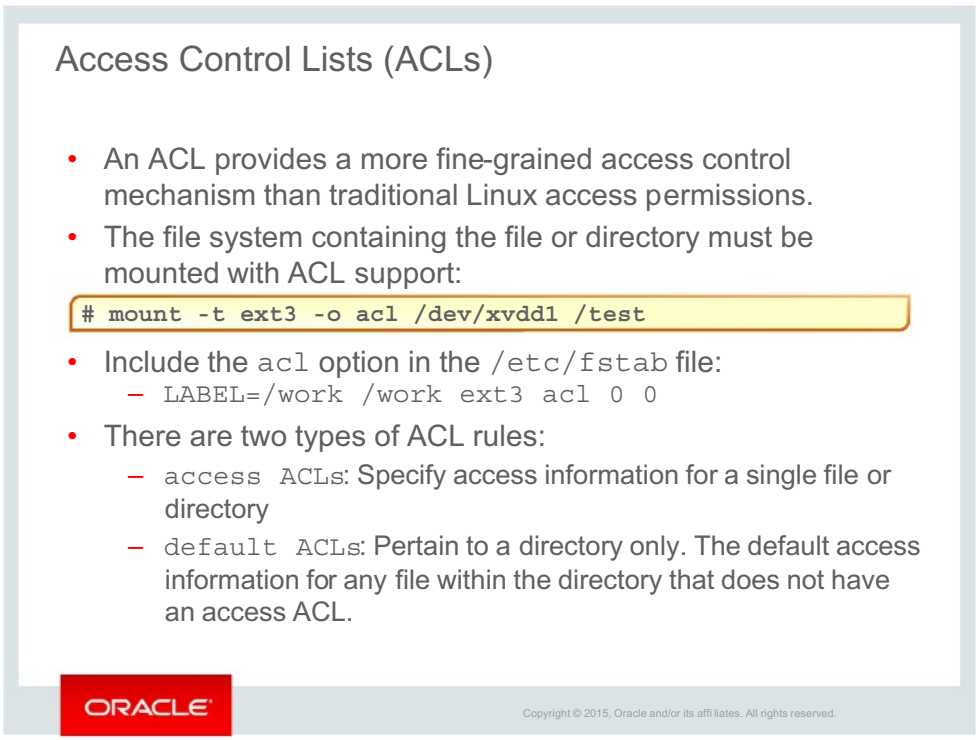
9

Advanced Storage Administration

Objectives

After completing this lesson, you should be able to:

- Describe access control lists (ACLs)
- Describe and configure disk quotas
- Describe the Linux `dm-crypt` device driver
- Describe and configure encrypted block devices
- Describe the `kpartx` utility
- Describe Udev
- Create Udev rules
- Use the `udevadm` utility



Traditional Linux access permissions for files and directories consist of setting a combination of read, write, and execute permissions for the owner of the file or directory, a member of the group the file or directory is associated with, and everyone else (other). Access control lists (ACLs) provide a finer-grained access control mechanism than these traditional Linux access permissions.

Before using ACLs for a file or directory, install the `acl` package:

```
# yum install acl
```

The file system containing the file or directory must also be mounted with ACL support. The following is the syntax to mount a local ext3 file system with ACL support:

```
# mount -t ext3 -o acl <device-name> <mount-point>
```

If the partition is listed in the `/etc/fstab` file, include the `acl` option:

```
LABEL=/work /work ext3 acl 0 0
```

An ACL consists of a set of rules that specify how a user or group can access the file or directory the ACL is associated with. There are two types of ACL rules:

- **`access ACLs`**: Specify access information for a single file or directory
- **`default ACLs`**: Pertain to a directory only. It specifies default access information for any file within the directory that does not have an access ACL.

getfacl and setfacl Utilities

- Use the getfacl utility to display a file's ACL.
  - When a file does not have an ACL, getfacl displays the same information as ls -l, although in a different format.
- Use the setfacl utility to add or modify rules.
- The rules are in the following form:
  - u:name:permissions
  - g:name:permissions
  - m:permissions
  - o:permissions
- To add an ACL rule that gives user oracle read and write permission to the test file:

```
# setfacl -m u:oracle:rwx test
```

Use the getfacl utility to display a file's ACL. When a file does not have an ACL, it displays the same information as ls -l, although in a different format. For example, the file test does not have an ACL:

```
# ls -l test
-rw-rw-r-- 1 oracle oracle 25 Mar 5 10:10 test
```

Sample getfacl output of the test file:

```
# getfacl test
# file: test
# owner: oracle
# group: oracle
user::rw-
group::rw-
other::r--
```

Use the setfacl utility to add or modify one or more rules in a file's ACL. The syntax is:

```
# setfacl -m <rules> <files>
```

The rules are in the following form:

- **u:name:permissions**: Sets the access ACL for a user (username or UID)
- **g:name:permissions**: Sets the access ACL for the group (group name or GID)
- **m:permissions**: Sets the effective rights mask. This is the union of all permissions of the owning group and all of the user and group entries.
- **o:permissions**: Sets the access ACL for everyone else (others)

The permissions are the traditional `r`, `w`, and `x` for read, write, and execute, respectively. The following example adds a rule to the ACL for the `test` file that gives the `oracle` user read and write permission to that file:

```
# setfacl -m u:oracle:rwx test
```

The output of `getfacl` includes the ACL rule:

```
# getfacl test
# file: test
# owner: oracle
# group: oracle
user::rw-
user:oracle:rwx
group::rw-
mask::rwx
other::r--
```

When a file has an ACL, `ls -l` displays a plus sign (+) following the permissions:

```
# ls -l test
-rw-rwxr--+ 1 oracle oracle 25 Mar 5 10:10 test
```

Use the `-x` option without specifying any permissions to remove rules for a user or group. To remove the ACL itself, use the `-b` option:

```
# setfacl -x u:oracle test
# setfacl -b test
```

To set a default ACL, add `d:` before the rule and specify a directory instead of a file name:

```
# setfacl -m d:o:rx /share
```

Disk Quotas

- Disk quotas limit file system disk usage:
  - For users and groups
  - To a number of blocks (disk space)
  - To a number of inodes (files)
- Set hard limits and soft limits on blocks and inodes.
  - Hard limits are maximums.
  - Soft limits have a grace period.
- Disk quota tools are used for:
  - Configuration
  - Reporting
  - Enabling/disabling
  - Maintaining accuracy

Set disk quotas for your users to restrict disk space and to notify you when usage is reaching a specified limit. Configure disk quotas for individual users as well as user groups. Quotas are set to limit the number of disk blocks (or disk space), as well as the number of inodes, which limit the number of files a user can create.

Hard limits define the maximum number of blocks (disk space) or inodes (files) for the user or group on the file system. Users can exceed soft limits for a certain period of time, called a "grace period." The grace period is configurable in time periods of days, hours, minutes, or seconds.

Various disk quota tools are summarized:

- **`quotacheck`:** This command creates disk usage tables `aquota.user` and `aquota.group`.
- **`edquota` / `setquota`:** These commands configure the quotas for users and groups. `edquota` is interactive and is also used to configure the grace period for soft limits.
- **`quota`:** Use this command to verify that quotas are set for users and groups.
- **`quotaon` / `quotaoff`:** Use these commands to enable and disable quotas.
- **`repquota`:** This command reports disk usage.
- **`quotacheck`:** Use this command to ensure the accuracy of quota reporting.

Enabling Disk Quotas

- Enable disk quotas with these file system mount options:
  - Use `usrquota` to enable for individual users
  - Use `grpquota` to enable for groups
- Use the `quotacheck` command to create a disk usage table for quota-enabled file systems:

```
# quotacheck -cug /home
```

- The above command creates the following files in the `/home` directory:
  - `aquota.user`
  - `aquota.group`

Enabling Quotas

Disk quotas are enabled by including mount options to entries in the `/etc/fstab` file. Include the `usrquota` mount option to configure disk quotas for individual users. Include the `grpquota` mount option to configure disk quotas for user groups. The following example enables both user and group quotas on the `/home` file system:

```
/dev/xvdb1   /home   ext4   usrquota,grpquota   0    0
```

After making any changes to entries in `/etc/fstab`, the file system must be unmounted and remounted for the change to take effect. Either run the `umount` command followed by the `mount` command to remount the file system, or use the `-o remount` option as follows:

```
# mount -o remount /home
```

Creating the Quota Database Files

After the `/etc/fstab` file is modified and the file system is remounted, run the `quotacheck` command. This command creates disk usage tables for the quota-enabled file system. Two files are created, `aquota.user` and `aquota.group`. Use the `-cug` options to the `quotacheck` command and include the quota-enabled file system mount point as an argument. Example:

```
# quotacheck -cug /home
```

To generate the table of file system disk usage, run the following command:

```
# quotacheck -avug
```

The options are described as follows:

- **-a**: Check all quota-enabled, locally mounted file systems.
- **-v**: Display verbose status information as the quota check proceeds.
- **-u**: Check user disk quota information.
- **-g**: Check group disk quota information.

Summary of Quota Commands

- `edquota`: Interactive utility to assign quotas and configure grace periods
- `setquota`: Command-line utility to assign quotas
- `quota`: Verify that quotas are set
- `quotaoff`: Disable quotas without modifying the limits
- `quotaon`: Enable quotas
- `repquota`: Report on disk usage
- `quotacheck`: Ensure the accuracy of quota reporting

Assigning Quotas per User

The `edquota` command is an interactive command to configure the quotas for a user. For example, to configure the quota for user `john`, enter the following command:

```
# edquota john
```

A text file opens in the default editor, defined by the EDITOR variable, and allows you to specify the limits for the user. The following is an example of the file:

```
Disk quotas for user john (uid 500)
Filesystem  blocks  soft  hard  inodes  soft  hard
/dev/xvdb1  400428     0     0   30412     0     0
```

The first column is the file system that has quota enabled. The second column is the number of blocks that the user is currently using. The next two columns are used to set soft and hard block limits for the user on the file system. The inodes column shows how many inodes the user is currently using. The last two columns are used to set the soft and hard inode limits for the user on the file system.

Make any changes to soft and hard limits in the text file and save and close the file. Repeat the `edquota` command for each user to whom you want to assign hard and soft limits on blocks and inodes.

Assigning Quotas per Group

The `edquota` command is also used to configure quotas for a group. Include the `-g group` option with the command. For example, to configure the quota for group teamA, enter the following command:

```
# edquota -g teamA
```

A text file opens, allowing you to set limits for the group. Set the soft and hard block limits, and the soft and hard inode limits for the group on the file system and save and close the file.

Setting Quotas from the Command Line

Use the `setquota` command to configure quotas from the command line. The syntax is:

```
setquota username block_soft_limit block_hard_limit
inode_soft_limit inode_hard_limit file_system
```

This command is also used to set quotas for groups. The `-p` option (prototype) allows quota settings from one user or group to be applied to another user or group.

Verifying Quotas

Use the `quota` command to verify that quotas are set. Enter the following to verify quotas for user `john`:

```
# quota john
```

Enter the following command to verify that quotas are set for group teamA:

```
# quota -g teamA
```

Setting the Grace Period

To configure the grace period for soft limits, use the `edquota -t` command. A text file opens allowing you to set both the block and inode grace periods as follows:

```
# edquota -t

Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem     Block grace period     Inode grace period
  /dev/xvdb1            7days                  7days
```

Make any changes and save and close the file.

Enabling and Disabling Quotas

One way to disable quotas is to set the limits to 0. If any of the values are set to 0, that limit is not set. Disable quotas without modifying the limits by using the `quotaoff` command. For example, to turn all user and group quotas off, enter:

```
# quotaoff -vaug
```

To enable quotas, use the `quotaon` command:

```
# quotaon -vaug
```

Alternatively, to enable quotas on a specific file system (for example `/home`), enter:

```
# quotaon -vug /home
```

Quota Reporting

Use the `repquota` command to report on disk usage. To view the report for all quota-enabled file systems, use the `-a` option:

```
# repquota -a
```

Include the file system as an argument to view the report for a specific quota-enabled file system. For example, to view the report for the `/home` file system, enter:

```
# repquota /home
```

Quota Accuracy

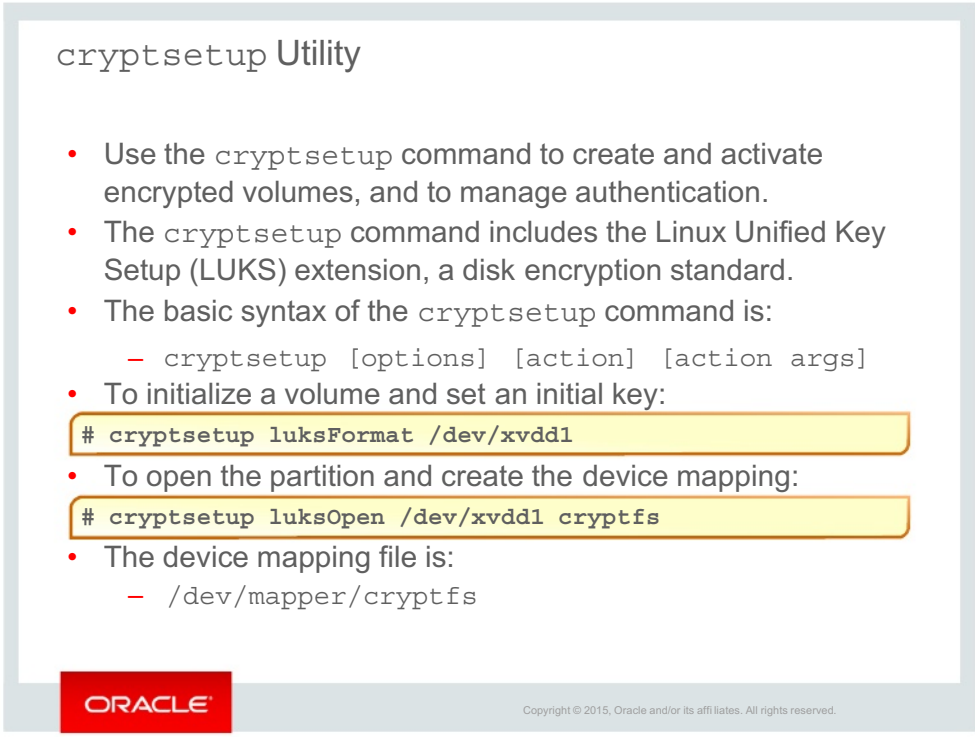
Run the `quotacheck` command to ensure the accuracy of quota reporting. Quota inaccuracies are caused by unclean system shutdowns. Unmount the file system before running this command. Disable quotas before running the `quotacheck` command and enable quotas afterwards. For example, to ensure the accuracy of quota reporting on the `/home` file system:

```
# quotaoff -vaug /home
# quotacheck -vaug /home
# quotaon -vaug /home
```

Encrypted Block Devices

- The `dm-crypt` device driver is used to encrypt block devices.
- Encrypted volumes can be stored on:
  - Disk partitions
  - Logical volumes
  - Disk images
- `dm-crypt` can encrypt:
  - All file systems supported by Linux
  - Swap space
  - RAID volumes
  - LVM physical volumes

The Linux device mapper also supports the creation of encrypted block devices using the `dm-crypt` device driver. Data on these encrypted devices can only be accessed by providing the correct password at boot time. Because the encryption takes place on the underlying block device, `dm-crypt` can be used for encrypting all file systems supported by Linux, as well as swap space. Encrypted volumes can be stored on disk partitions, logical volumes, and disk images. `dm-crypt` can also be configured to encrypt RAID volumes and LVM physical volumes.



Because the `dm-crypt` device mapper is concerned only with encryption of the block device, it relies on user-space tools, such as `cryptsetup`, to set up `dm-crypt` managed device-mapper mappings. Use `cryptsetup` to create and activate encrypted volumes and to manage authentication.

The `cryptsetup` command also provides commands to deal with the LUKS on-disk format. LUKS is a standard for hard disk encryption. It standardizes a partition header, as well as the format of the bulk data.

LUKS Actions

The following is a partial listing of valid LUKs actions:

- **`luksFormat`:** Initializes a LUKS partition and sets the initial key
- **`luksOpen`:** Opens the partition and creates the device mapping
- **`luksSuspend`:** Suspends the active device and wipes the encryption key from the kernel
- **`luksResume`:** Resumes the suspended device and reinstates the encryption key
- **`luksAddKey`:** Adds a new key passphrase
- **`luksRemoveKey`:** Removes the key from the LUKS device
- **`luksUUID`:** Prints the UUID if the device has a LUKS header
- **`luksClose`:** Closes the partition and removes the device mapping

Using the `cryptsetup` Command

The basic syntax of the `cryptsetup` command is:

```
cryptsetup [options] [action] [action args]
```

For example, to initialize a volume and set an initial key, enter the following command. A warning message appears, asking for confirmation to continue. You are prompted for the initial key (passphrase) twice to ensure that your password is typed correctly.

```
# cryptsetup luksFormat /dev/xvdd1
WARNING!
========
This will overwrite data on /dev/xvdd1 irrevocably.

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

To open the partition and create the device mapping, enter:

```
# cryptsetup luksOpen /dev/xvdd1 cryptfs
Enter passphrase for /dev/xvdd1: <passphrase>
```

The device mapping created in this example is `/dev/mapper/cryptsfs`. Create the file system on this device mapping file, not the physical device (`/dev/xvdd1`). The device mapping file is actually a symbolic link to `/dev/dm-0`:

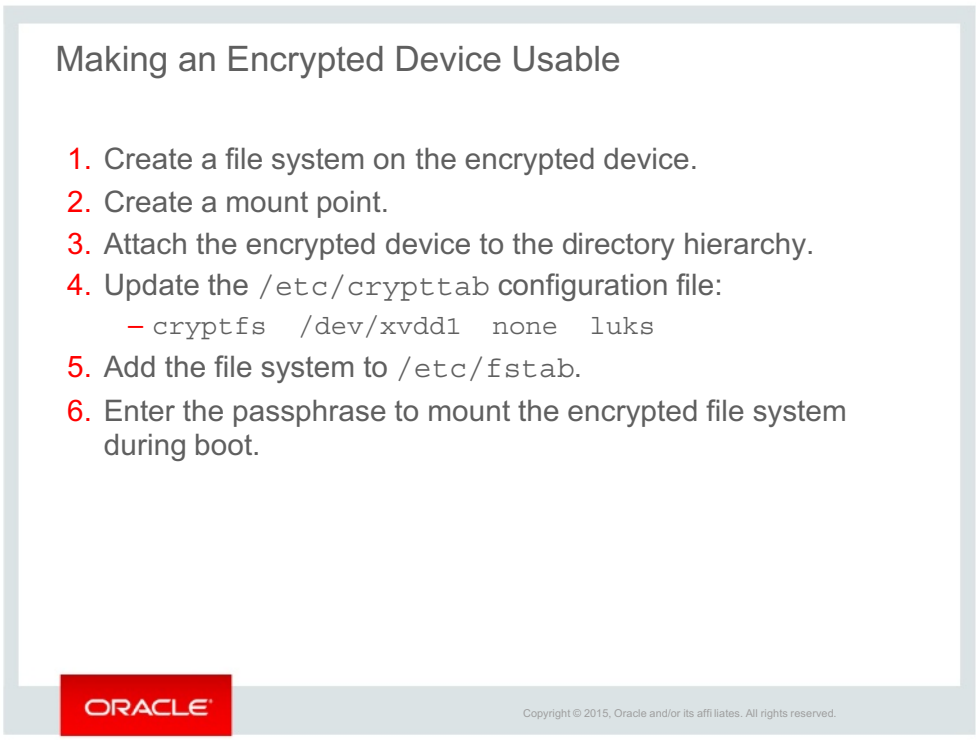
```
# ls -l /dev/mapper/cryptfs
lrwxrwxrwx, ... /dev/mapper/cryptfs -> ../dm-0
```

To check the status of the encrypted volume, enter:

```
# cryptsetup status cryptfs
/dev/mapper/cryptfs is active.
  type:  LUKS1
  cipher:  aes-xts-plain64
  keysize: 256 bits
  device:  /dev/xvdd1
  offset:  4096 sectors
  size:    2093056 sectors
  mode:    read/write
```

To close the partition and remove the device mapping, enter:

```
# umount /cryptfs (if the file system is mounted)
# cryptsetup luksClose /dev/mapper/cryptfs
```



As is the case with any block device, to make the encrypted device usable, you must create a file system on the device, create a mount point, and attach the device to the directory hierarchy.

Assuming that the encrypted device name is `/dev/mapper/cryptfs` and that you want to create an ext4 file system on the device and mount it to `/crypt`, enter:

```
# mkfs -t ext4 /dev/mapper/cryptfs
# mkdir /crypt
# mount /dev/mapper/cryptfs /crypt
```

Then update the configuration file, `/etc/crypttab`. This ensures that the encrypted file system is properly set up and mounted at boot time. The following entry is appropriate for your example:

```
# cat /etc/crypttab
# <target name>  <source device>  <key file>  <options>
cryptfs          /dev/xvdd1       none        luks
```

Finally, add the file system to `/etc/fstab` for the actual mounting to take place. You are prompted to enter your passphrase for the encrypted file system during the boot process.

kpartx Utility

- The kpartx utility is used to create device maps from partition tables.
- It reads block devices and creates device mappings of partitions in /dev/mapper.
- Using the system.img drive image as an example, to list partitions found on the drive image:

```
# kpartx -l system.img
```

- To add device mappings for the detected partitions:

```
# kpartx -a system.img
```

- You can now mount the partitions in /dev/mapper and view the files that they contain.
- To disconnect the device:

```
# kpartx -d system.img
```

ORACLE

The kpartx utility can be used to set up device mappings for the partitions of any partitioned block device. It reads partition tables on the specified device and creates device maps for the detected partitions. After running kpartx on a partitioned block device, device files are created in /dev/mapper. These files represent a disk partition or a disk volume.

Using the system.img file that you installed Oracle Linux on as an example, the following set of commands illustrates the usage of kpartx. Use the -l option to list any partitions that are found on the drive image.

```
# kpartx -l system.img
loop0p1 : 0 204800 /dev/loop0 2048
loop0p2 : 0 12288000 /dev/loop0 206848
loop0p3 : 0 4096000 /dev/loop0 212494848
loop0p4 : 0 2 /dev/loop0 16590848
```

The output shows that the drive image contains four partitions. The first column gives the names of the device files that are created. Before adding the device files, a listing of /dev/mapper shows no devices.

```
# ls /dev/mapper
control
```

To add the device mappings for the detected partitions, use the `-a` option.

```
# kpartx -a system.img
```

To view the new device mappings:

```
# ls /dev/mapper
control   loop0p1   loop0p2   loop0p3   loop0p4
```

You can now mount the partitions and view the files that they contain. For example, create a mount point and mount the first partition:

```
# mkdir /mnt/sysimage
# mount /dev/mapper/loop0p1 /mnt/sysimage
```

View the files on this first partition:

```
# ls /mnt/sysimage
config-3.10.0-229.el7.x86_64
config-3.8.13-55.1.6.el7uek.x86_64
...
```

The `/boot` file system is mounted on the first partition. As expected, the files on `/boot` (`/dev/xvda1`) are the same:

```
# df -kh | grep xvda1
/dev/xvda1   97M  46M  46M  50%  /boot
# ls /boot
config-3.10.0-229.el7.x86_64
config-3.8.13-55.1.6.el7uek.x86_64
...
```

To unmount the partition and disconnect the device, enter:

```
# umount /mnt/sysimage
# kpartx -d system.img
```

Notice that the mapping is gone in `/dev/mapper`:

```
# ls /dev/mapper
control
```

Udev: Introduction

- Udev dynamically creates device file names at boot time.
- Udev is now part of `systemd`.
  - The Udev daemon, `systemd-udevd`, receives device uevents directly from the kernel whenever a device is added or removed.
  - For every event, `systemd-udevd` executes matching instructions specified in Udev rules.
- With Udev, device file names can change after reboot.
  - `/dev/sdc` could be named `/dev/sdb` after reboot.
- You can configure Udev to create persistent device names.
- You can use these names in the file system mount table, `/etc/fstab`, or as an argument to the `mount` command.

Udev is the device manager for the Linux kernel. Udev dynamically creates or removes device node files at boot time in the `/dev` directory for all types of devices. Udev is now part of `systemd` as you can see by viewing the "`udev`" file names included with the `systemd` RPM package.

```
# rpm -ql systemd | grep udev
/etc/udev
...
```

The Udev daemon, `systemd-udevd`, receives device uevents directly from the kernel whenever a device is added or removed from the system. For every event, `systemd-udevd` executes matching instructions specified in Udev rules.

Device file names can change when disks are removed from the system due to failure. For example, devices are named `/dev/sda`, `/dev/sdb`, and `/dev/sdc` at boot time. But on the next reboot, `/dev/sdb` fails and what was previously `/dev/sdc` is named `/dev/sdb`. Any configuration references to `/dev/sdb` now contain content srcinally referenced by `/dev/sdc`.

The solution to avoid this type of situation is to guarantee consistent names for devices through reboots. You can configure Udev to create persistent names and use these names in the file system mount table, `/etc/fstab`, or as an argument to the `mount` command.

Udev Rule Files and Directories

- Udev rules determine how to identify devices and how to assign a persistent name.
- Udev rules files are located in the following directories:
  - `/lib/udev/rules.d/` – The default rules directory
  - `/etc/udev/rules.d/` – The custom rules directory
- Custom rules files override default rules files of the same name.
- Rules files are sorted and processed in lexical order.
- Sample rules file names:
  - `10-dm.rules`
  - `50-udev-default.rules`

Udev rules determine how to identify devices and how to assign a name that is persistent through reboots or disk changes. When Udev receives a device event, it matches the configured rules against the device attributes in `sysfs` to identify the device. Rules can also specify additional programs to run as part of device event handling.

Udev rules files are located in the following directories:

- `/lib/udev/rules.d/` – The default rules directory
- `/etc/udev/rules.d/` – The custom rules directory. These rules take precedence.

Rules files need to have unique names. Files in the custom rules directory override files of the same name in the default rules directory. Rules files are sorted and processed in lexical order. The following is a partial listing of rules files from the default and custom rules directories:

```
# ls /lib/udev/rules.d
100-balloon.rules  10-dm.rules  11-dm-lvm.rules
...
# ls /etc/udev/rules.d
70-persistent-ipoib.rules
```

Sample Udev Rules

```
SUBSYSTEM=="virtio-ports", KERNEL=="vport*",
   ATTR{name}=="?*", SYMLINK+="virtio-ports/$attr{name}"

SUBSYSTEM=="tty", KERNEL=="tty[0-9]*", GROUP="tty",
   MODE="0620"

SUBSYSTEM=="mem", KERNEL=="mem|kmem|port", GROUP="kmem",
   MODE="0640"

SUBSYSTEM=="usb", EBV{DEVTYPE}=="usb_device",
   ENV{ID_USB_INTERFACES}=="*:0701??:*", GROUP="lp"

SUBSYSTEM=="block", GROUP="disk"

SUBSYSTEM=="scsi_generic|scsi_tape", SUBSYSTEMS=="scsi",
   ATTRS{type}=="1|8", GROUP="tape"

KERNEL=="tun", MODE="0666", OPTIONS+="static_node-
   net/tun"
```


This slide contains selected entries from the `/lib/udev/rules.d/50-udev-default.rules` file. This rules file contains over 60 entries. The selected entries assist in describing the syntax of the rules files.

Comments begin with a # sign. Each noncommented line in a rules file consists of a list of one or more key-value pairs separated by a comma. There are two types of keys:

- Match keys
- Assignment keys

If all match keys match their respective value, the rule gets applied and the assignment keys are assigned the specified value. Each key has a distinct operation, depending on the operator. Valid operators are:

- `==`: Compare for equality
- `!=`: Compare for inequality
- `=`: Assign a value to a key
- `+=`: Add the value to the current values for the key
- `:=`: Assign the final value to the key. Disallow any later changes by any later rules.

Shell-style pattern matching (`*`, `?`, `[]`) is also supported in Udev rules.

Match Keys

The following key names are used to match against device properties. Some of the keys also match against properties of the parent devices in `sysfs`, and not just the device that has generated the event. If multiple keys are specified in a single rule, all these keys must match.

- `ACTION`: Match the name of the event action.
- `DEVPATH`: Match the devpath of the event device.
- `KERNEL`: Match the name of the event device.
- `NAME`: Match the name of a network interface. It can be used if the `NAME` key was set in one of the preceding rules.
- `SYMLINK`: Match the name of the symlink targeting the node. It can be used if a `SYMLINK` key was set in one of the preceding rules. There can be multiple symlinks but only one needs to match.
- `SUBSYSTEM`: Match the subsystem of the event device.
- `TEST{`*`octal mode mask`*`}`: Test the existence of a file. You can specify *`octal mode mask`*.

Other match keys include `DRIVER`, `ATTR{`*`filename`*`}`, `KERNELS`, `SUBSYSTEMS`, `DRIVERS`, `ATTRS{`*`filename`*`}`, `TAGS`, `ENV{`*`key`*`}`, `TAG`, `PROGRAM`, and `RESULT`.

Assignment Keys

The following keys can have values assigned to them:

- `NAME` – The name to use for a network interface. The name of a device node cannot be changed by Udev, only additional symlinks can be created.
- `SYMLINK` – The name of the symlink targeting the node
- `OWNER`, `GROUP`, `MODE` – The permissions for the device node
- `OPTIONS` – Rule and device options. The `ignore_remove` option used in the example means "Do not remove the device node when the device goes away."

Other assignment keys include `ATTR{`*`key`*`}`, `ENV{`*`key`*`}`, `TAG`, `RUN{type}`, `LABEL`, `GOTO`, `IMPORT{`*`type`*`}`, `WAIT_FOR`, and `OPTIONS`.

String Substitutions

The `NAME`, `SYMLINK`, `PROGRAM`, `OWNER`, `GROUP`, `MODE`, and `RUN` keys support many `printf`-like string substitutions. The substitutions used in the example are:

- `%M` – The kernel major number for the device
- `%m` – The kernel minor number for the device

Additional string substitutions are supported. Refer to the `udev(7)` man page for all supported substitutions and details on additional match keys, additional assignment keys, and additional rule and device options.

udevadm Utility

- The udevadm utility is a management tool for Udev.
- To query the Udev database for all device information for /dev/xvdd:

```
# udevadm info --query=all --name=/dev/xvdd
```

- To print all sysfs properties of /dev/xvdd:

```
# udevadm info --attribute-walk --name=/dev/xvdd
```

- These properties can be used in Udev rules to match the device.
- It prints all devices along the chain, up to the root of sysfs.

The udevadm utility is a userspace management tool for Udev. Among other functions, you can use udevadm to query sysfs and obtain device attributes to help in creating Udev rules that match a device. To display udevadm usage:

```
# udevadm --help
Usage: udevadm [--help] [--version] [--debug] COMMAND [OPTIONS]
  info          query sysfs or the udev database
  trigger       requests events from the kernel
  settle        wait for the event queue to finish
  control       control the udev daemon
  monitor       listen to kernel and udev events
  hwdb          maintain the hardware database index
  test          test an event run
  test-builtin  test a built-in command
```

You can also obtain usage for each of the udevadm commands. For example, to get help on using the info command:

```
# udevadm info --help
```

Some examples follow. To query the Udev database for the device path of /dev/xvdd:

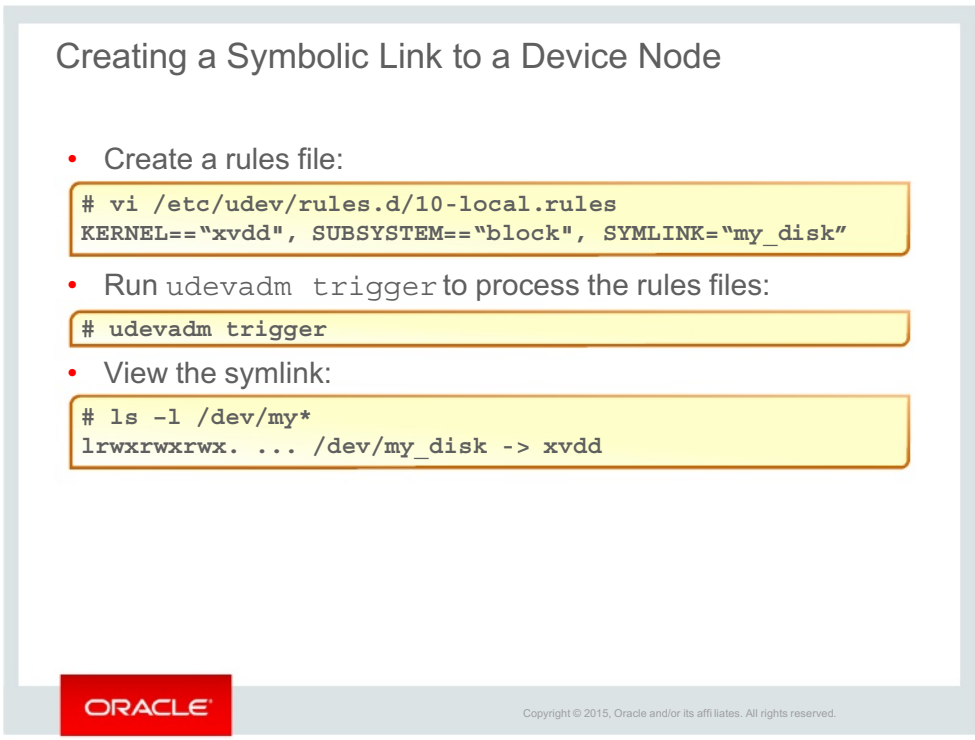
```
# udevadm info --query=path --name=/dev/xvdd
/devices/vbd-5696/block/xvdd
```

To query the Udev database for all device information for /dev/xvdd:

```
# udevadm info --query=all --name=/dev/xvdd
P: /devices/vbd-5696/block/xvdd
N: xvdd
E: DEVNAME=/dev/xvdd
E: DEVPATH=/devices/vbd-5696/block/xvdd
E: DEVTYPE=disk
E: MAJOR=202
E: MINOR=48
E: MPATH_SBIN_PATH=/sbin
E: SUBSYSTEM=block
E: TAGS=:systemd:
E: USEC_INITIALIZED=12403
```

Enter the following to print all sysfs properties of /dev/xvdd. These properties can be used in Udev rules to match the device. It prints all devices along the chain, up to the root of sysfs.

```
# udevadm info --attribute-walk --name=/dev/xvdd
...
 looking at device '/devices/vbd-5696/block/xvdd':
    KERNEL=="xvdd"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{ro}=="0"
    ATTR{size}=="20971520"
...
  looking at parent device '/devices/vbd-5696':
    KERNELS=="vbd-5696"
    SUBSYSTEMS=="xen"
    DRIVERS=="vbd"
    ATTR{devtype}=="vbd"
    ATTR{nodename}=="device/vbd/5696"
```



The order in which rules are evaluated is important. When creating your own rules, you want these evaluated before the defaults. Because rules are processed in lexical order, create a rules file with a file name such as `/etc/udev/rules.d/10-local.rules` for it to be processed first.

The following rule creates the `/dev/my_disk` symbolic link to the `/dev/xvdd` device node. You can create a Udev rule to change the name of a network interface but the name of a device node cannot be changed by Udev. Only additional symlinks can be created for device nodes.

```
KERNEL=="xvdd", SUBSYSTEM=="block", SYMLINK="my_disk"
```

Run `udevadm trigger` to process the rules files:

```
# udevadm trigger
```

The symlink now exists.

```
# ls -l /dev/my*
lrwxrwxrwx. ... /dev/my_disk -> xvdd
```

Remove the `10-local.rules` file and run `udevadm trigger` to remove the symlink.

Quiz

Which of the following can be encrypted?

a. All file systems supported by Linux
b. Swap space
c. RAID volumes
d. LVM physical volumes
e. All of the above

Quiz

Which of the following statements are true?

a. Udev is part of `systemd`.
b. You can define Udev rules to create persistent symbolic links to device nodes.
c. Run the `udevadm trigger` command to process rules files.
d. The `start_udev` utility is a user-space management tool for Udev.

Summary

In this lesson, you should have learned how to:

- Describe access control lists (ACLs)
- Describe and configure disk quotas
- Describe the Linux `dm-crypt` device driver
- Describe and configure encrypted block devices
- Describe the `kpartx` utility
- Configure Udev rules

Practice 9: Overview

The practices for this lesson cover the following:

- Creating and mounting a file system
- Implementing access control lists
- Setting disk quotas
- Encrypting a file system
- Using `kpartx`
- Exploring and configuring Udev rules

10

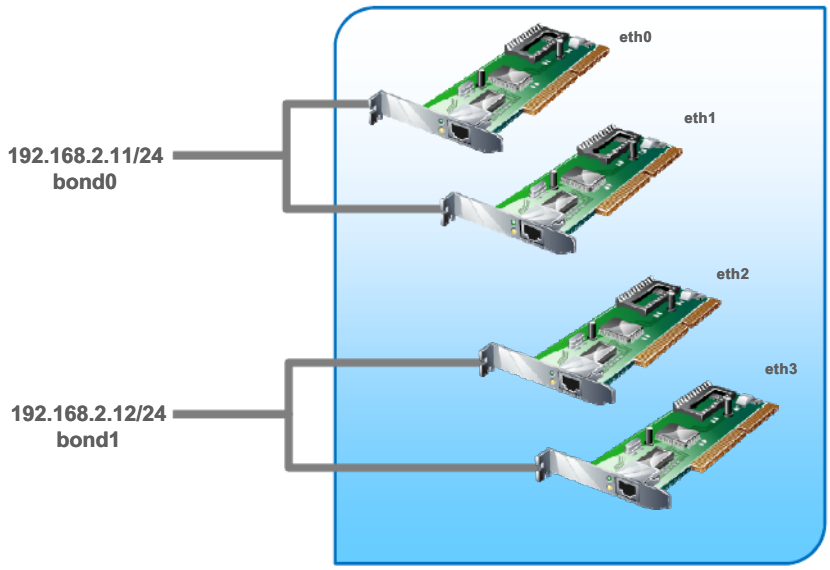
Advanced Networking

Objectives

After completing this lesson, you should be able to:

- Describe Network Bonding
- Configure Network Bonding
- Describe Virtual Local Area Networks (VLANs)
- Configure a VLAN
- Describe Virtual Private Networks (VPNs)
- Configure a Site-to-Site VPN

Network Bonding: Introduction



Network interface bonding is called by many names: Port Trunking, Channel Bonding, Link Aggregation, NIC teaming, and others. It combines or aggregates multiple network connections into a single channel bonding interface. This allows two or more network interfaces to act as one, to increase throughput and to provide redundancy or failover.

The Linux kernel comes with the bonding driver for aggregating multiple physical network interfaces into a single logical interface (for example, aggregating `eth0` and `eth1` into `bond0`). For each bonded interface you can define the mode and the link monitoring options. There are seven different mode options, each providing specific load balancing and fault tolerance characteristics.

The slide shows four network interface cards (NICs): `eth0`, `eth1`, `eth2`, and `eth3`. The `eth0` and `eth1` NICs are combined into a single `bond0` interface with an IP address of `192.168.2.11/24`. The `eth2` and `eth3` NICs are combined into a single `bond1` interface with an IP address of `192.168.2.12/24`.

Network Bonding: Configuration

- To configure network bonding:
  - Either manually create a bonding interface file in the `/etc/sysconfig/network-scripts/` directory
  - Or use the NetworkManager GUI
  - Or use the `nmtui` utility
  - Or use the `nmcli` utility
- The physical network interfaces included in the bonding interface are called “slaves.”
  - Each “slave” also has an interface file in the `/etc/sysconfig/network-scripts/` directory
  - The configuration file includes `MASTER` and `SLAVE` directives.

Creating a Bonding Interface File

You can manually create a bonding interface file in the `/etc/sysconfig/network-scripts` directory. You can also use the NetworkManager GUI, the `nmtui` text user interface, or use the `nmcli` command-line interface. You first create the bonding interface and then you add the physical network interfaces to the bond. These physical network interfaces are called “slaves.”

For example, in the previous slide, the slaves for the `bond0` interface are `eth0` and `eth1`. The slaves for `bond1` are `eth2` and `eth3`.

Creating Slave Interface Files

The “slaves” also have an associated file in the `/etc/sysconfig/network-scripts` directory. These files identify the bond by using the `MASTER` directive, and the slaves by using the `SLAVE` directive.

Examples of configuration files in the `/etc/sysconfig/network-scripts` directory are shown on the following page.

The following is an example of a bonding interface file:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BONDING_OPTS="miimon=1 updelay=0 downdelay=0 mode=active-backup"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
IPADDR=192.168.2.12
PREFIX=24
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
...
NAME=bond0
UUID=...
ONBOOT=yes
```

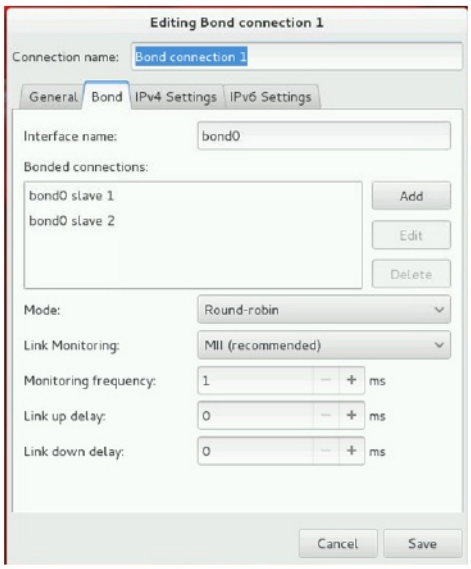
The following example defines the `eth0` physical network interface as a slave for `bond0`:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-eth0
TYPE=Ethernet
NAME=bond-slave-eth0

UUID=...
DEVICE=eth0
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

The following example defines the `eth1` physical network interface as a slave for `bond0`:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-eth1
TYPE=Ethernet
NAME=bond-slave-eth1
UUID=...
DEVICE=eth1
ONBOOT=yes

MASTER=bond0
SLAVE=yes
```

Using the NetworkManager GUI to Configure Network Bonding



The slide shows the NetworkManager GUI used to configure a network bonding connection. You can access this GUI by using the following command:

```
# nm-connection-editor
```

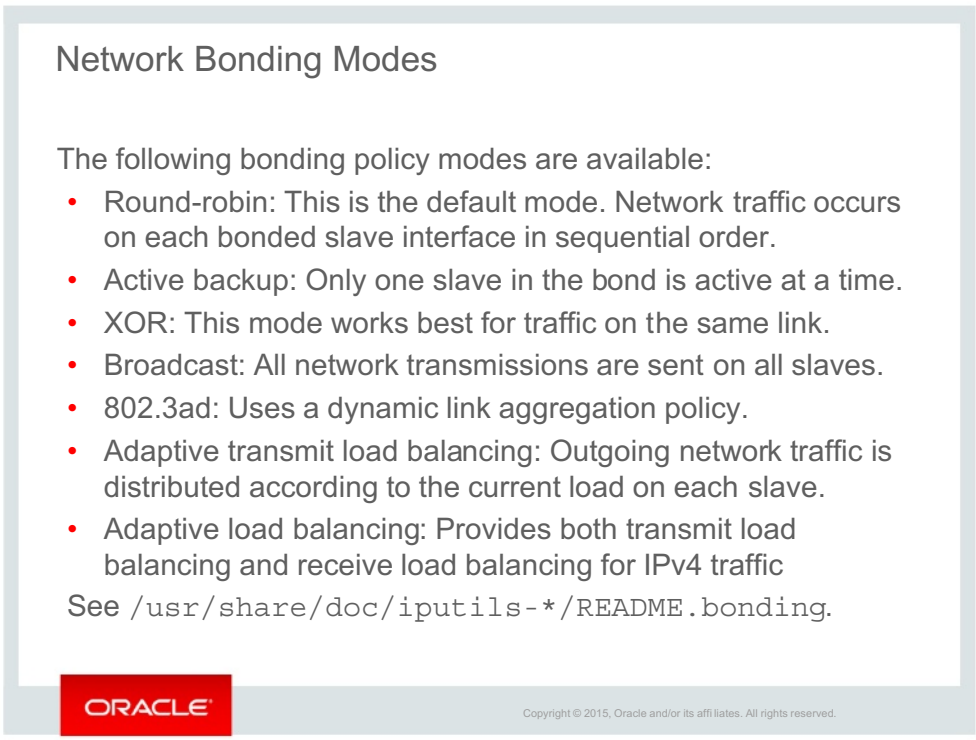
This command displays a Network Connections GUI, which includes an “Add” button. Click “Add” and then you are prompted to choose a connection type. Select “Bond” as the connection type to display the GUI shown in the slide.

You can also access the GUI shown in the slide by clicking the network icon in the GNOME Notification Area. You can then select “Network Settings” from the pop-up window. Click the plus sign (+) to add a new connection type. Then select “Bond” as the connection type to display the GUI shown in the slide.

From the “Bond” tab as shown in the slide, you can do the following:

- Provide a name for the bonded interface, which defaults to `bond0`.
- Click “Add” to add the physical network interfaces, which are known as “slaves.”
- Configure the Mode, which defaults to “Round-robin.”
- Configure Link Monitoring, which defaults to MII (Media Independent Interface).
- Configure the Monitoring frequency, Link up delay, and Link down delay.

Click “Save” and the interface files are created in `/etc/sysconfig/network-scripts/`.



The following bonding policy modes are available:

- **Round-robin:** This is the default mode. Network transmissions are in sequential order beginning with the first available slave. This mode provides load balancing and fault tolerance.
- **Active backup:** Only one slave in the bond is active. Another slave interface becomes active if the active slave interface fails. The bond's MAC address is externally visible on only one network adapter to avoid confusing a network switch. This mode provides fault tolerance.
- **XOR (exclusive-or):** Network transmissions are based on a transmit hash policy. The default policy derives the hash by using MAC addresses. In this mode, network transmission destined for specific peers are always sent over the same slave interface. This mode works best for traffic to peers on the same link or local network. This mode provides load balancing and fault tolerance.
- **Broadcast:** All network transmissions are sent on all slave interfaces. This mode provides fault tolerance.
- **802.3ad:** Uses an IEEE 802.3ad dynamic link aggregation policy. Aggregation groups share the same speed and duplex settings. This mode transmits and receives network traffic on all slaves in the active aggregator. This mode requires an 802.3ad-compliant network switch.

The remaining bonding policy modes are described:

- **Adaptive transmit load balancing (TLB):** Outgoing network traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode does not require any special switch support.
- **Adaptive load balancing (ALB):** This mode includes transmit load balancing (tlb) and receive load balancing (rlb) for IPv4 traffic and does not require any special switch support. Receive load balancing is achieved by ARP negotiation.

See the `/usr/share/doc/iputils-*/README.bonding` file for complete descriptions of the available bonding policy modes. This information is also available at http://www.kernel.org/doc/Documentation/networking/bonding.txt

Network Bonding Link Monitoring

The bonding driver supports two methods to monitor a slave's link state:

- MII (Media Independent Interface) monitor
  - This is the default, and recommended, link monitoring option.
  - It monitors the carrier state of the local network interface.
  - You can specify the monitoring frequency and the delay.
  - Delay times allow you to account for switch initialization.
- ARP monitor
  - This sends ARP queries to peer systems on the network and uses the response as an indication that the link is up.
  - You can specify the monitoring frequency and target addresses.

MII (Media Independent Interface) Monitor

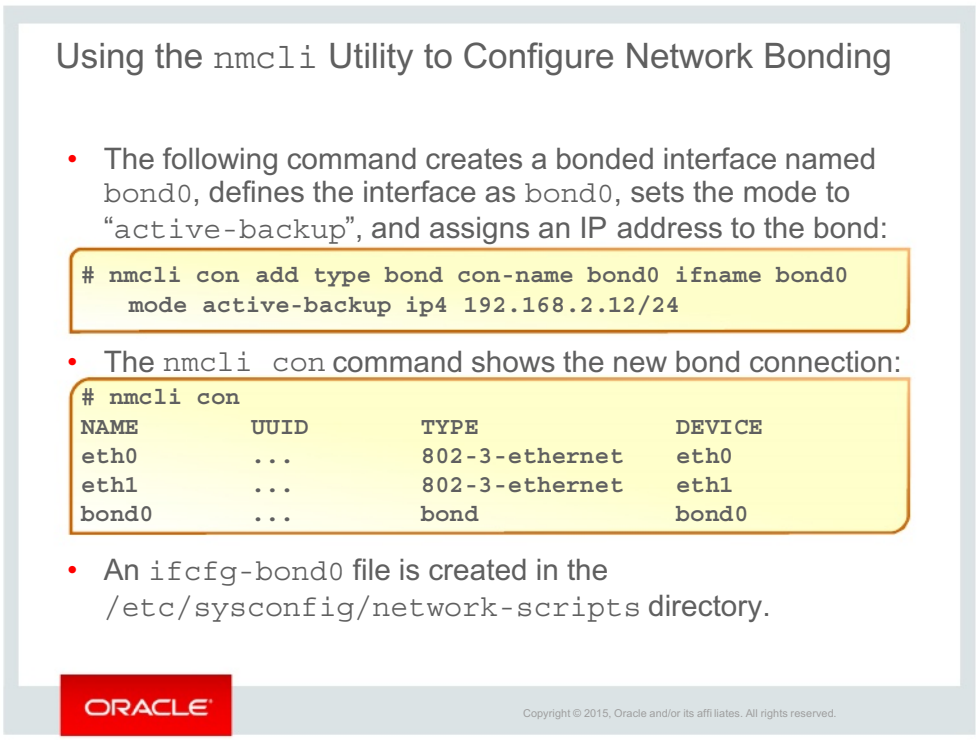
This is the default link monitoring option. This method monitors only the carrier state of the local network interface. It relies on the device driver for carrier state information, or queries the MII registers directly, or uses `ethtool` to attempt to obtain carrier state. You can specify the following information for MII monitoring:

- Monitoring frequency: The time in milliseconds between querying carrier state
- Link up delay: The time in milliseconds to wait before using a link that is up
- Link down delay: The time in milliseconds to wait before switching to another link when the active link is reported as down

ARP Monitor

This method of link monitoring sends APR queries to peer systems on the network and uses the response as an indication that the link is up. The ARP monitor relies on the device driver to keep the last receive time, and the transmit start time, updated. If the device driver is not updating these times, the ARP monitor fails any slaves that use that device driver. You can specify the following information for APR monitoring:

- Monitoring frequency: The time in milliseconds that ARP queries are sent
- ARP targets: A comma-separated list of IP addresses that ARP queries are sent to



Use the `nmcli connection` command without any arguments to view the existing network connections. You can shorten the "`connection`" argument to "`con`". Example:

```
# nmcli con
NAME        UUID    TYPE              DEVICE
eth0        ...     802-3-ethernet    eth0
eth1        ...     802-3-ethernet    eth1
```

Include the "`add type bond`" arguments, and any additional information to create a network bond connection. The following example creates a bonded interface named `bond0`, defines the interface as `bond0`, sets the mode to "`active-backup`", and assigns an IP address to the bonded interface.

```
# nmcli con add type bond con-name bond0 ifname bond0 mode
active-backup ip4 192.168.2.12/24
```

The `nmcli con` command shows the new bond connection.

```
# nmcli con
NAME        UUID    TYPE              DEVICE
bond0       ...     bond              bond0
```

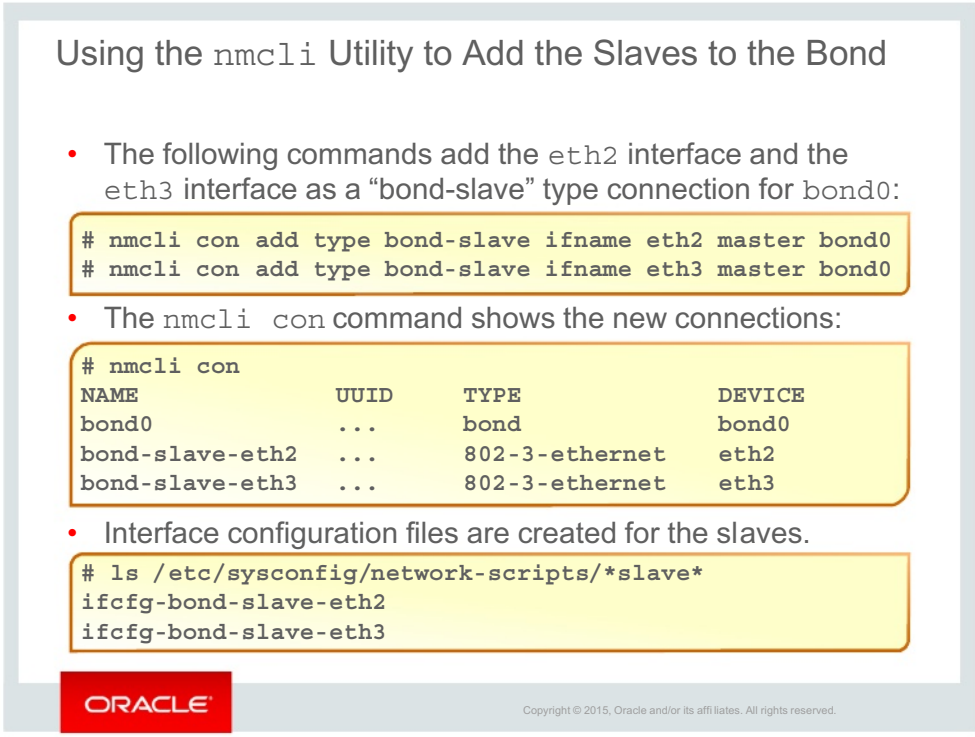
The `nmcli con add type bond` command creates an interface configuration file in the `/etc/sysconfig/network-scripts` directory. For example:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BONDING_OPTS=mode=active-backup
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
IPADDR=192.168.2.12
PREFIX=24
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=bond0
UUID=...
ONBOOT=yes
```

The `ip addr` command shows the new `bond0` interface:

```
# ip addr
...
6: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 ...
state UNKNOWN
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.12/24 brd 192.168.1.255 scope global bond0
```

Note that the IP address is assigned to the interface but the MAC address is all zeros.



For each interface that you want to bond, use the `nmcli con add type bond-slave` command. The following example adds the `eth2` interface as a bond slave. The command does not include the `con-name` argument so a name is automatically generated.

```
# nmcli con add type bond-slave ifname eth2 master bond0
Connection 'bond-slave-eth2' (<UUID>) successfully added.
```

The following example adds the `eth3` interface as a “bond-slave.”

```
# nmcli con add type bond-slave ifname eth3 master bond0
Connection 'bond-slave-eth3' (<UUID>) successfully added.
```

The `nmcli con` command shows the new connections.

```
# nmcli con
NAME                UUID    TYPE             DEVICE
bond0               ...     bond             bond0
bond-slave-eth2     ...     802-3-ethernet   eth2
bond-slave-eth3     ...     802-3-ethernet   eth3
```

The `nmcli con add type bond-slave` commands create interface configuration files in the `/etc/sysconfig/network-scripts` directory. For example:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-eth2
TYPE=Ethernet
NAME=bond-slave-eth2
UUID=...
DEVICE=eth2
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

The "bond-slave" configuration file for the `eth3` interface is shown:

```
# cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-eth3
TYPE=Ethernet
NAME=bond-slave-eth3
UUID=...
DEVICE=eth3
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

The `ip addr` command includes "SLAVE" for the `eth2` and `eth3` interfaces and also includes "master bond0".

```
# ip addr
...
4: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 ...
master bond0 state UP ...
    link/ether 00:16:3e:00:03:02 brd ff:ff:ff:ff:ff:ff
5: eth3: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 ...
master bond0 state UP ...
    link/ether 00:16:3e:00:03:02 brd ff:ff:ff:ff:ff:ff
6: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 ...
state UNKNOWN
    link/ether 00:16:3e:00:03:02 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.10/24 brd 192.168.1.255 scope global bond0
...
```

Note that the bond and the two slave entries have the same MAC address

Activate the Bond

- You can use the `nmcli` command to bring up the connections.
- Bring up the slaves first, and then bring up the bond interface.
- The following commands bring up the slaves:

```
# nmcli con up bond-slave-eth2
# nmcli con up bond-slave-eth3
```

- The following command brings up the `bond0` interface:

```
# nmcli con up bond0
```

You can use the `nmcli` command to bring up the connections. Bring up the slaves first, and then bring up the bond interface. The following commands bring up the slaves:

```
# nmcli con up bond-slave-eth2
# nmcli con up bond-slave-eth3
```

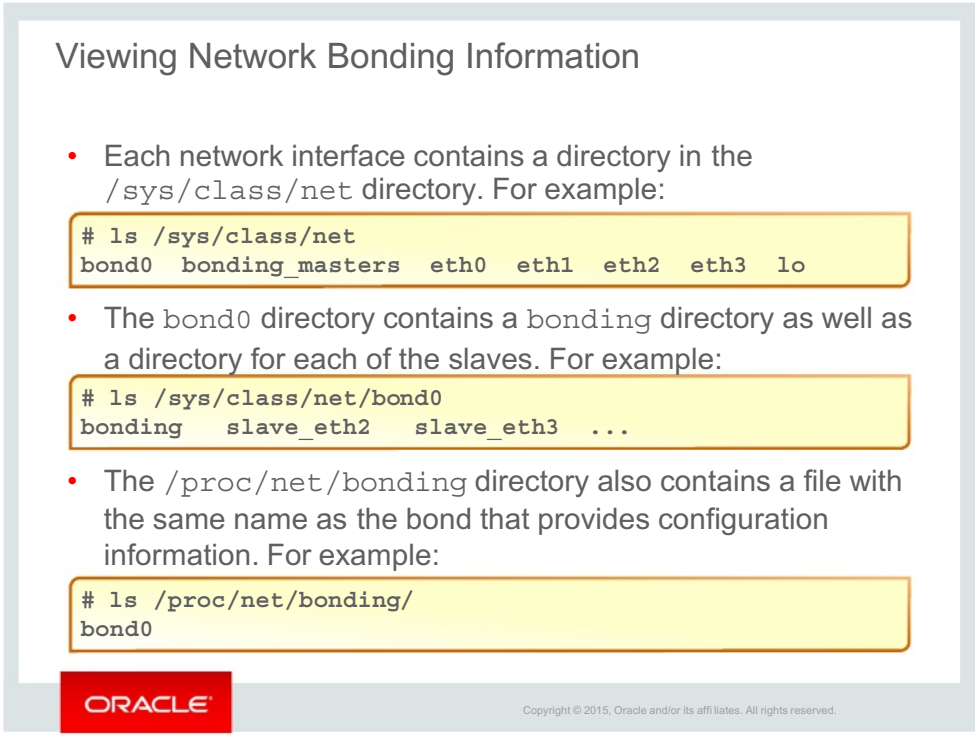
The following command brings up the `bond0` interface:

```
# nmcli con up bond0
```

The `ip addr` command, or the `ip link` command, now shows the slave and the bond interfaces that are UP.

```
# ip link
...
4: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ... state UP
5: eth3: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> ... state UP
6: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> ... state UP
...
```

Stopping the master bond interface also stops the slave interfaces.



Each network interface contains a directory in the `/sys/class/net` directory. For example:

```
# ls /sys/class/net
bond0  bonding_masters  eth0  eth1  eth2  eth3  lo
```

In this example, a network bond named `bond0` exists. A directory of the same name exists that contains configuration information for that bond. For example:

```
# ls /sys/class/net/bond0
addr_assign_type  carrier  ifalias  netdev_group  slave_eth3 ...
...
```

Within this directory is a `bonding` directory that contains information for the `bond0` interface. For example:

```
# ls /sys/class/net/bond0/bonding
active_slave  all_slaves_active  miimon  primary_reselect ...
...
```

There are also directories that contain information for each of the slaves. For example:

```
# ls /sys/class/net/bond0/slave_eth2
addr_assign_type  device  ifalias  mtu  rxbuf_cur  ...
```

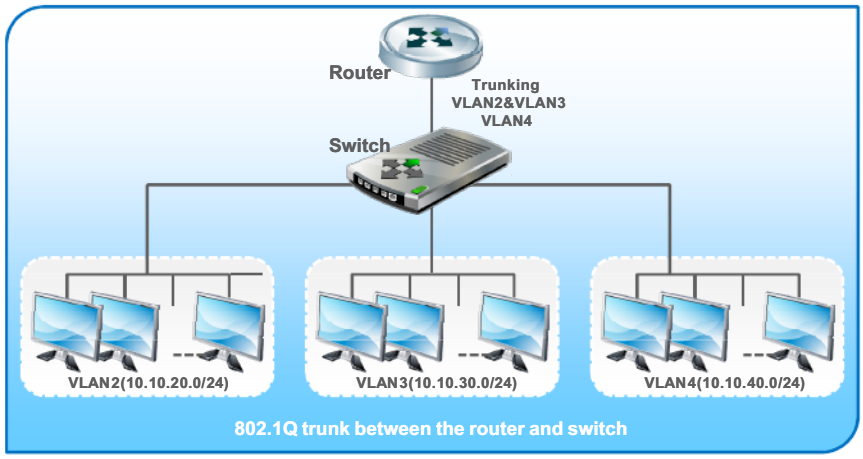
Following are some examples of viewing files in the `/sys/class/net` directory.

```
# cat /sys/class/net/bonding_masters
bond0
# cat /sys/class/net/bond0/operstate
up
# cat /sys/class/net/bond0/address
00:16:3e:00:03:01
# cat /sys/class/net/bond0/bonding/active_slave
eth2
# cat /sys/class/net/bond0/bonding/mode

active-backup 1
# cat /sys/class/net/bond0/bonding/slaves
eth2 eth3
```

Following is an example of viewing the `/proc/net/bonding/bond0` file.

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: eth2
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0

Down Delay (ms): 0

Slave Interface: eth2
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 00:16:3e:00:03:01
Slave queue ID: 0

Slave Interface: eth3
MII Status: up
Speed: Unknown

Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 00:16:3e:00:03:01
Slave queue ID: 0
```

Virtual Local Area Networks: Introduction



A VLAN is a type of local area network that does not have its own dedicated physical infrastructure, but instead uses another local area network to carry its traffic. The traffic is encapsulated so that a number of logically separate VLANs can be carried by the same physical LAN. With VLANs, you can create multiple distinct broadcast domains that are mutually isolated. With VLANs, network switches (not routers) create the broadcast domain.

Each VLAN is identified by a VID (VLAN Identifier) in the range 1 to 4094 inclusive. Switch ports are assigned to a VLAN ID, and all ports assigned to a single VLAN are in a single broadcast domain. The VID is stored in an extra 4-byte header that is added to the packet called the Tag. Adding a Tag to a packet is called tagging.

IEEE 802.1Q is a protocol for carrying VLAN traffic on Ethernet. There are two ways in which a machine can be connected to a switch carrying 802.1Q VLAN traffic:

- Via an *access* (or *untagged*) port, where VLAN support is handled by the switch (so the machine sees ordinary, *untagged* Ethernet frames)
- Via a *trunk* (or *tagged*) port, where VLAN support is handled by the attached machine (which sees 802.1Q-tagged Ethernet frames)

Linux has the ability to use an Ethernet interface as an 802.1Q trunk port, allowing it to concurrently send and receive traffic on multiple VLANs. This is provided by the 8021q kernel module.

Using the NetworkManager GUI to Configure 802.1Q VLAN Tagging

The slide shows the NetworkManager GUI used to configure an 802.1Q VLAN connection. You can access this GUI by using the following command:

```
# nm-connection-editor
```

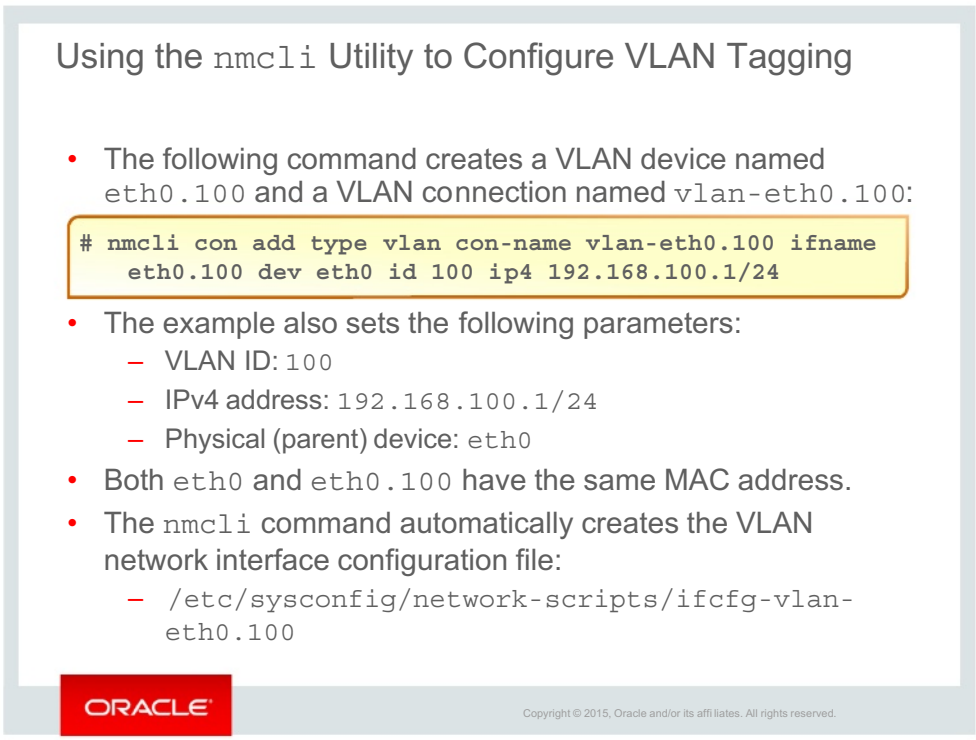
This command displays a Network Connections GUI, which includes an “Add” button. Click “Add” and then you are prompted to choose a connection type. Select “VLAN” as the connection type to display the GUI shown in the slide.

You can also access the GUI shown in the slide by clicking the network icon in the GNOME Notification Area. You can then select “Network Settings” from the pop-up window. Click the plus sign (+) to add a new connection type. Then select “VLAN” as the connection type to display the GUI shown in the slide.

From the “VLAN” tab as shown in the slide, you can do the following:

- Provide a name for the VLAN connection, which defaults to “VLAN connection 1.”
- Select the “Parent interface” from a dropdown list of available physical interfaces.
- Provide the “VLAN id,” which is set to `100` in this example.
- Provide the VLAN interface name,” which is set to `eth0.100` in this example.
- Optionally provide “Cloned MAC address” and “MTU” settings.

Click “Save” and the interface files are created in `/etc/sysconfig/network-scripts/`.



You can also use the `nmcli connection` command to create a VLAN connection. Include the "`add type vlan`" arguments and any additional information to create a VLAN connection. For example:

```
# nmcli con add type vlan con-name vlan-eth0.100 ifname eth0.100
dev eth0 id 100 ip4 192.168.100.1/24
```

The example defines the following attributes of the VLAN connection:

- `con-name vlan-eth0.100`: Specifies the name of the new VLAN connection
- `ifname eth0.100`: Specifies the interface to bind the connection to
- `dev eth0`: Specifies the physical (parent) device this VLAN is on
- `id 100`: Specifies the VLAN ID
- `ip4 192.168.100.1/24`: Specifies IPv4 address to assign to the interface

The `nmcli con` command shows the new VLAN connection.

```
# nmcli con
NAME              UUID    TYPE            DEVICE
vlan-eth0.100     ...     vlan            eth0.100
```

This command creates the `ifcfg-vlan-eth0.100` file. Following is the contents of this file:

```
# cat /etc/sysconfig/network-scripts/ifcfg-vlan-eth0.100
VLAN=yes
TYPE=Vlan
DEVICE=eth0.100
PHYSDEV=eth0
VLAN_ID=100
REORDER_HDR=0
BOOTPROTO=none
IPADDR=192.168.100.1
PREFIX=24
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=vlan-eth0.100
UUID=...

ONBOOT=yes
```

You can use the `ip addr` command to view the protocol address information for the network devices. The following shows the VLAN interface, `eth0.100`:

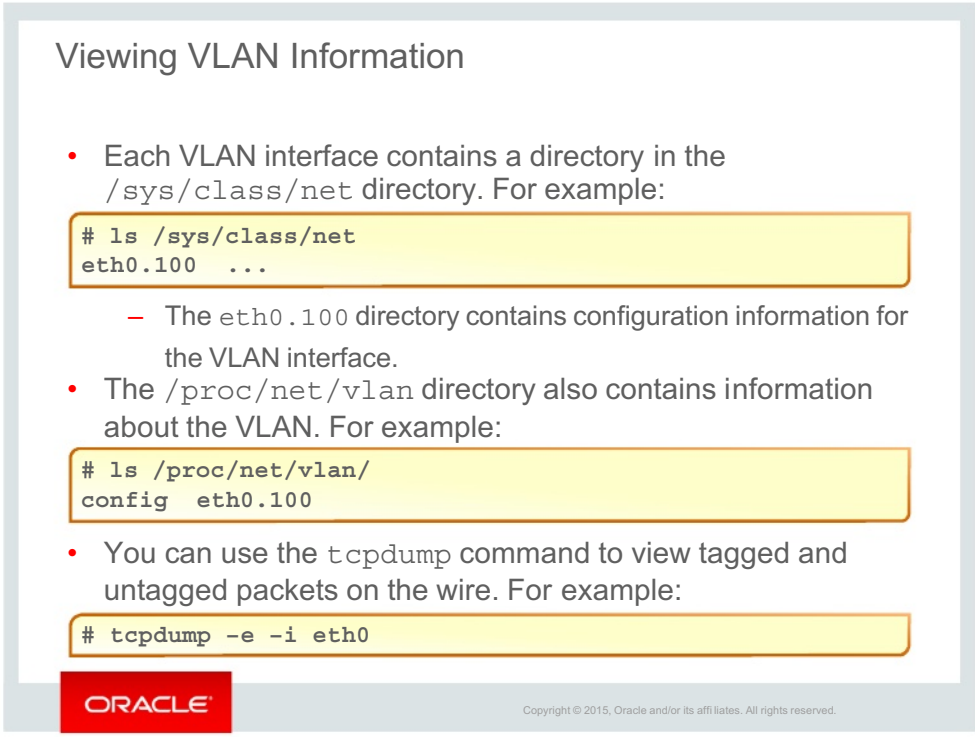
```
# ip addr
6: eth0.100@eth0: ...
     link/ether 00:16:3e:00:01:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.1/24 brd 192.168.100.255 scope ... eth0.100
...
```

The `nmcli dev` command shows the `eth0.100` device.

```
# nmcli dev
NAME            TYPE           STATE            CONNECTION
eth0.100        vlan           connected        vlan-eth0.100
```

The `nmcli con` command shows the `vlan-eth0.100` connection.

```
# nmcli con
NAME               UUID    TYPE              DEVICE
vlan-eth0.100      ...     vlan              eth0.100
```



Each network interface contains a directory in the `/sys/class/net` directory. For example:

```
# ls /sys/class/net
eth0.100  eth0  eth1  eth2  eth3  lo
```

In this example, a VLAN interface exists named `eth0.100` and a directory of the same name exists that contains configuration information for that interface. For example:

```
# ls /sys/class/net/eth0.100
addr_assign_type  carrier  flags  link_mode  power  ...
...
```

There are also files in the `/proc/net/vlan` directory that describe the VLAN interface. For example:

```
# ls /proc/net/vlan
config  eth0.100
```

You can use the `tcpdump` utility to see tagged and untagged packets to ensure traffic is showing up on the expected interfaces. The `-e` option specifies the Ethernet header that includes 802.1Q tags. Use the `-i` option to specify the interface. For example:

```
# tcpdump -e -i eth0
```

Virtual Private Networks: Introduction

- A virtual private network (VPN) is a secure connection between two or more endpoints over an untrusted network.
  - VPNs create a secure tunnel that typically uses authentication and encryption between the two endpoints.
- There are two main types of VPNs:
  - Site-to-Site: Each site has a VPN gateway that provides a secure link between client systems at each site.
  - Remote Access: Also referred to as mobile VPN or Road Warrior. Each client has VPN software and connects to a remote VPN gateway.
- IPSec (IP Security) is the preferred method for creating a VPN.
- IPSec includes various protocols including AH, ESP, and IKE.

A virtual private network (VPN) enables secure communication between two or more private networks over an untrusted network such as the Internet. VPNs create a secure tunnel that typically uses authentication and encryption between the two networks.

There are different types of VPNs:

- **Site-to-Site VPN:** Each site has a VPN server that encrypts data and provides a secure link between the sites. Client systems at each site do not have VPN client software, they send and receive normal TCP/IP traffic through the VPN gateway server.
- **Remote Access VPN:** A secure connection is made from an individual computer to a VPN gateway. Every host must have VPN client software.

IPSec (IP Security) is the preferred method for creating a VPN. IPSec is a layer 3 protocol and can protect any protocol that runs on top of IP. IPSec consists of various protocols and algorithms including the following:

- **Authentication Header (AH):** Protects the integrity of the entire packet including header information such as the source and destination IP addresses
- **Encapsulating Security Payload (ESP):** Protects the application-level data and provides both integrity checking and encryption
- **Internet Key Exchange (IKE):** Protocol that uses a system of automatic keying to derive a key for IPSec communications

The libreswan RPM Package

- libreswan is an open source IPSec implementation included with Oracle Linux 7.
  - Uses the Network Security Services (NSS) cryptographic library that is required for FIPS security compliance
  - Provides the ipsec command-line utility and a number of utilities in the /usr/libexec/ipsec directory
  - Provides the IPSec configuration directory, /etc/ipsec.d
  - Provides the main configuration file for IPSec, /etc/ipsec.conf
- Use the ipsec command to generate security keys, view security keys, and view connection status. Example:

```
# ipsec newhostkey --configdir /etc/ipsec.d --output
   /etc/ipsec.d/www.example.com.secrets
# ipsec showhostkey --left
# ipsec auto --status
```


In Oracle Linux 7, IPSec is provided by the libreswan RPM package. libreswan is an open source IPSec implementation. It uses the Network Security Services (NSS) cryptographic library that is required for Federal Information Processing Standard (FIPS) security compliance. Use the yum command to install this package and all required dependencies.

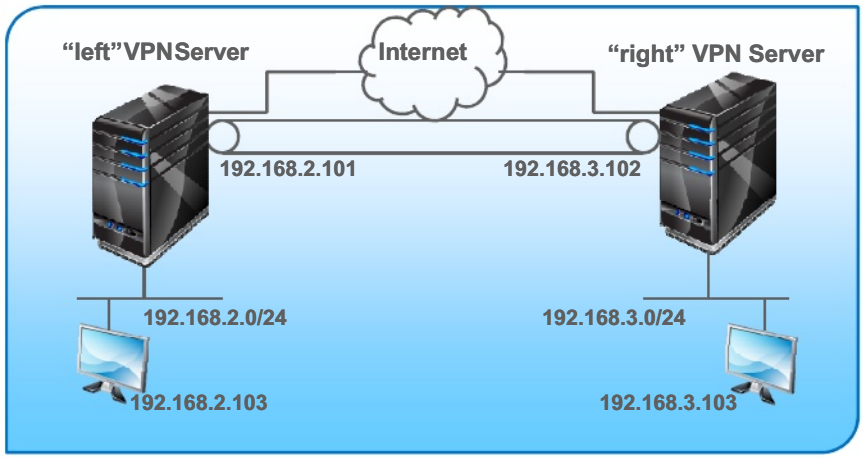
```
# yum install libreswan
```

The libreswan package provides the ipsec command-line utility, a number of utilities in the /usr/libexec/ipsec directory, and several other files including man pages and documentation. The IPSec configuration directory is /etc/ipsec.d and the main configuration file for IPSec is /etc/ipsec.conf.

The ipsec utility provides a number of commands. Use the ipsec --help command to view a list of the commands. The following describes a few of the ipsec commands:

- ipsec newhostkey: Generate a new RSA authentication key for a host. Include the -output <filename> secrets option to store the RSA authentication key.
- ipsec showhostkey: Display a host's authentication key. The output is suitable for copying and pasting the key in to the /etc/ipsec.conf configuration file.
- ipsec auto --status: View VPN connection status and supported ESP and IKE algorithms.

Site-to-Site VPN



The slides show a VPN configuration that is referred to as a site-to-site, or gateway-to-gateway VPN. In this configuration, there are two VPN servers that each have a public Internet IP address. The VPN servers also have connections to two separate private networks, `192.168.2.0/24` and `192.168.3.0/24`.

Each VPN server is running IPSec. IPSec is configured to provide a secure VPN tunnel connection between the two private networks.

The main configuration file for IPSec is `/etc/ipsec.conf`. You need to configure a "`sitetosite`" connection in this file and provide configuration information for each VPN server.

`libreswan` uses the terms "`left`" and "`right`" to refer to the VPN servers at each site. In this example, the server with the IP address of `192.168.2.101` is the "`left`" side of the connection and the server with the IP address of `192.168.3.102` is the "`right`" side of the connection. Parameters for the "`left`" and "`right`" VPN servers are defined for the "`sitetosite`" connection in the `/etc/ipsec.conf` file.

Site-to-Site VPN: Configuration

- On each VPN server:
  - Use the `sysctl` command to enable IP forwarding
  - Use the `ipsec newhostkey` command to generate an RSA authentication key
  - Use the `ipsec showhostkey --left` command to view the key on the left host
  - Use the `ipsec showhostkey --right` command to view the key on the right host
  - Copy and paste the output of each `ipsec showhostkey` command into the `/etc/ipsec.conf` file
  - Complete the "`sitetosite`" connection configuration in the `/etc/ipsec.conf` file (details in the next slide)
  - Add `firewalld` rules to trust the `ipsec` protocols
  - Use the `systemctl` command to start the `ipsec` service

For the site-to-site VPN configuration in the previous slide, each of the VPN servers acts as a router and forward packets destined for other hosts. Therefore, IP forwarding needs to be enabled on each VPN server. You can use the `sysctl` command to enable IPv4 forwarding on each VPN server:

```
# sysctl -w net.ipv4.ip_forward=1
```

On each VPN server, use the `ipsec newhostkey` command to generate an RSA authentication key. Example:

```
# ipsec newhostkey --configdir /etc/ipsec.d --output
/etc/ipsec.d/www.example.com.secrets
```

In this example, an RSA key pair is generated by using the Network Security Services (NSS) database and is written to the `/etc/ipsec.d/www.example.com.secrets` file.

In the example configuration in the slide titled "Site-to-Site VPN," the server with the IP address of `192.168.2.101` is the "`left`" side of the connection. From this server, use the `ipsec showhostkey --left` command to view the key on the left host. The `--left` option causes the output to be in the `ipsec.conf` format as a "`leftrsasigkey`" parameter.

```
# ipsec showhostkey --left
...
leftrsasigkey=...
```

From the "right" VPN server, use the ipsec showhostkey --right command to view the key on the right host. The --right option causes the output to be in the ipsec.conf format as a "rightrsasigkey" parameter.

```
# ipsec showhostkey --right
...
rightrsasigkey=...
```

Complete the "sitetosite" connection configuration in the /etc/ipsec.conf file with the "leftrsasigkey" parameter, the "rightrsasigkey" parameter, and the following information. This assumes the configuration represents the example in the "Site-to-Site VPN" slide.

```
# vi /etc/ipsec.conf
conn sitetosite
    leftid=192.168.1.101
    left=192.168.1.101
    leftsourceip=192.168.2.101
    leftsubnet=192.168.2.0/24
    leftrsasigkey=...

    rightid=192.168.1.102
    right=192.168.1.102
    rightsourceip=192.168.3.102
    rightsubnet=192.168.3.0/24
    rightrsasigkey=...

    authby=rsasig
    auto=start
```

Use the following command to check the syntax of the /etc/ipsec.conf file. If any errors are returned, the line number is included. If no errors are returned, the syntax is correct:

```
# /usr/libexec/ipsec/addconn --config /etc/ipsec.conf --
checkconfig
```

Copy the completed /etc/ipsec.conf file to the other VPN server so that both the "left" and the "right" VPN servers have the same configuration file.

Either add firewalld rules to trust the ipsec protocols, or stop the firewalld service on both the "left" and the "right" VPN servers. The following example uses the systemctl command to stop the firewalld service.

```
# systemctl stop firewalld
```

Use the systemctl command to start the ipsec service on both the "left" and the "right" VPN servers.

```
# systemctl start ipsec
```

Example "sitetosite" Connection

```
conn sitetosite
    leftid=192.168.1.101
    left=192.168.1.101
    leftsourceip=192.168.2.101
    leftsubnet=192.168.2.0/24
    leftrsasigkey=...

    rightid=192.168.1.102
    right=192.168.1.102
    rightsourceip=192.168.3.102
    rightsubnet=192.168.3.0/24
    rightrsasigkey=...

    authby=rsasig
    auto=start
```

The slide shows the "sitetosite" connection configuration in the /etc/ipsec.conf file. This assumes that the configuration represents the example in the "Site-to-Site VPN" slide. The parameters are described as follows. Refer to the ipsec.conf(5) man page for more information.

- **conn**: A connection specification defining a network connection to be made by using IPSec. In this example, the connection name is "sitetosite" but could be named whatever you want.
- **leftid**|**rightid**: Identifies the "left|right" VPN servers for authentication purposes. This could be an IP address or a fully qualified domain name.
- **left**|**right**: The IP address of the "left|right" VPN servers
- leftsourceip|rightsourceip: The IP address of the "left|right" VPN server to use when transmitting a packet to the other side of the link
- **leftsubnet**|**rightsubnet**: The private subnet behind the "left|right" participant
- **leftrsasigkey**|**rightrsasigkey**: The output of the ipsec showhostkey command for the "left|right" participant. This key is generated by using the ipsec newhostkey command.
- **authby**: How the two security gateways authenticate each other
- **auto**: What operation to perform at IPSec startup

Quiz

Which of the following statements are true?

a. Network bonding allows you to combine multiple physical network interfaces together into a single interface.
b. In network bonding, the physical interfaces are called “slaves” and the logical bonded interface is called “master.”
c. You need to load the 8021q kernel module to implement network bonding.
d. A VLAN does not have its own dedicated physical infrastructure and instead uses another LAN to carry its traffic.
e. ESP is the preferred method for creating a VPN in Oracle Linux 7.

Summary

In this lesson, you should have learned how to:

- Describe Network Bonding
- Configure Network Bonding
- Describe Virtual Local Area Networks (VLANs)
- Configure a VLAN
- Describe Virtual Private Networks (VPNs)
- Configure a Site-to-Site VPN

Practice 10: Overview

This practice covers the following topics:

- Configuring Network Bonding by Using the GUI
- Configuring Network Bonding from the Command Line
- Working with Bonded Interfaces
- Configuring 802.1Q VLAN Tagging by Using the GUI
- Configuring 802.1Q VLAN Tagging from the Command Line
- Working with VLAN Interfaces
- Configuring a Site-to-Site VPN