Oracle Linux Advanced Administration

Student Guide – Volume I

D84498GC10

Edition 1.0

April 2014

D86117

Author

Craig McBride

Technical Contributors and Reviewers

Michele Dady

Avi Miller

Elena Zannoni

Wim Coekaerts

Al Flournoy

Harald Van Breederode

Joel Goodman

Manish Kapur

Yasar Akthar

Ozgur Yuksel

Antoinette O'Sullivan

Gavin Bowe

Nick Alcock

Dwight Engen

Wayne Lewis

Herbert Van Den Bergh

Tim Hill

Kris Van Hees

John Haxby

Graphic Editor

Maheshwari Krishnamurthy

Editors

Smita Kommini

Daniel Milne

Publishers

Veena Narasimhan

Michael Sebastian Almeida

Jobi Varghese

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS
The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

Oracle Systems Learning Stream

A New Innovative Continuous Learning Solution

Ideal for supplementing what you learned in the classroom

Oracle Learning Streams

Videos from Oracle Experts 🎬

Continuously Updated Content 🔁

Connect Live with Instructors 👥

ORACLE UNIVERSITY

Oracle Systems Learning Stream

Keep Your Skills Current through Continuous Learning

Expert Delivered

Access to hundreds of instructional videos delivered by Oracle subject matter experts

Training Across Your Infrastructure

For technical and business professionals looking to regularly broaden and deepen their knowledge

Continuously Refreshed Content

Covers technical, new features, how-to information and more…. on Oracle Hardware, Software, Operating System and Virtualization Solutions

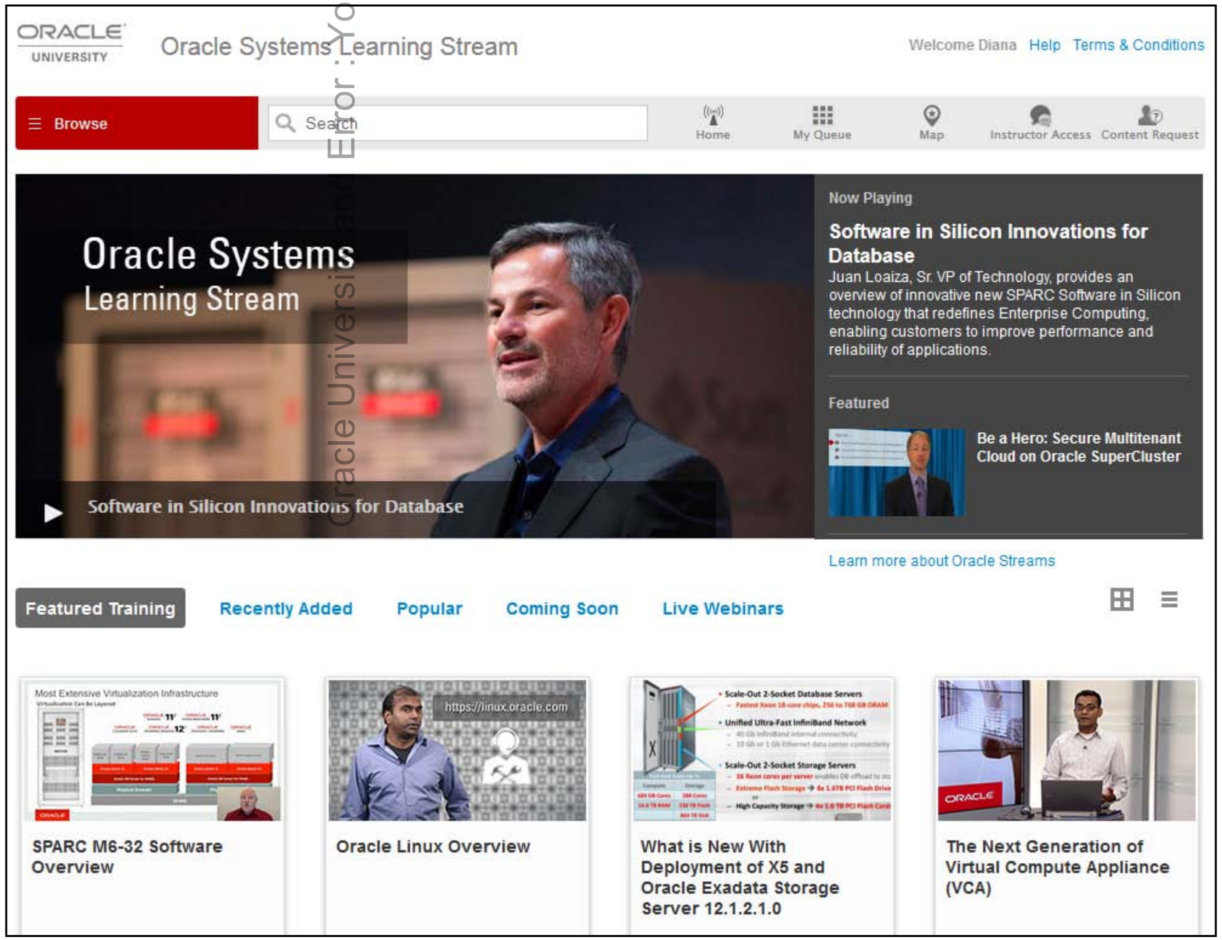
Request Topics that Interest You

Subscription Service

Preview the Oracle Systems Learning Stream

NOW!

education.oracle.com/streams/systems



ORACLE UNIVERSITY Oracle Systems Learning Stream

Welcome Diana Help Terms & Conditions

Software in Silicon Innovations for Database

Now Playing

Software in Silicon Innovations for Database

Juan Loaiza, Sr. VP of Technology, provides an overview of innovative new SPARC Software in Silicon technology that redefines Enterprise Computing, enabling customers to improve performance and reliability of applications.

Featured

Be a Hero: Secure Multitenant Cloud on Oracle SuperCluster

Learn more about Oracle Streams

SPARC M6-32 Software Overview

Oracle Linux Overview

What is New With Deployment of X5 and Oracle Exadata Storage Server 12.1.2.1.0

The Next Generation of Virtual Compute Appliance (VCA)

ORACLE

1

Introduction

Course Goals

This course teaches you how to:

- Configure network addressing and name services
- Configure authentication and directory services
- Configure web and email services
- Perform a Kickstart installation
- Configure Samba services
- Perform advanced software package management
- Perform advanced storage administration
- Configure Oracle Cluster File System version 2 (OCFS2)
- Configure iSCSI targets and initiators
- Configure device multipathing

Course Goals

This course teaches you how to:

- Configure an XFS file system
- Configure a Btrfs file system
- Configure control groups (Cgroups)
- Configure Kernel-based Virtual Machine (KVM)
- Configure Linux containers (LXC)
- Configure SELinux
- Enable core dump and perform core dump analysis
- Configure dynamic tracing (DTrace)

ORACLE

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: Web and Email Services
	Lesson 5: Installing Oracle Linux by Using Kickstart
	Lesson 6: Samba Services
	Lesson 7: Advanced Software Package Management
Day 3	Lesson 8: Advanced Storage Administration
	Lesson 9: OCFS2 and Oracle Clusterware
	Lesson 10: iSCSI and Multipathing

Schedule

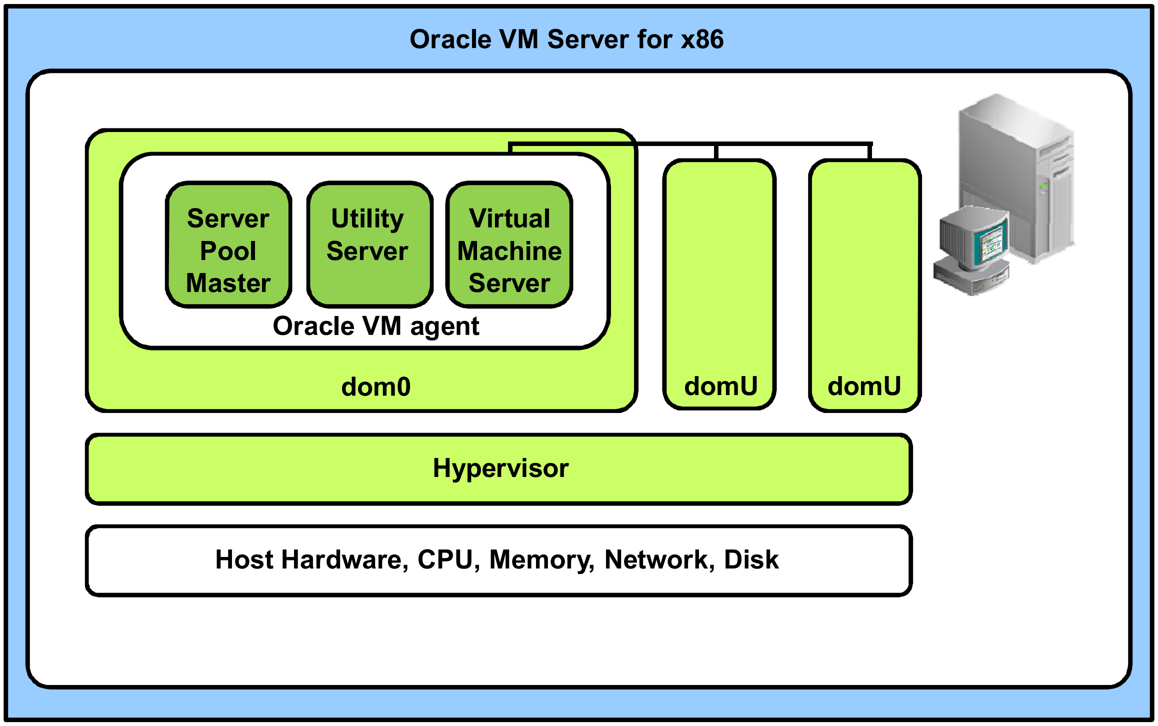
Session	Module
Day 4	Lesson 11: XFS File System Lesson 12: Btrfs File System Lesson 13: Control Groups (cgroups) Lesson 14: Virtualization with Linux
Day 5	Lesson 15: Linux Containers (LXC) Lesson 16: SELinux Lesson 17: Core Dump Analysis Lesson 18: Dynamic Tracing with DTrace

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 with Oracle VM Server for x86



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

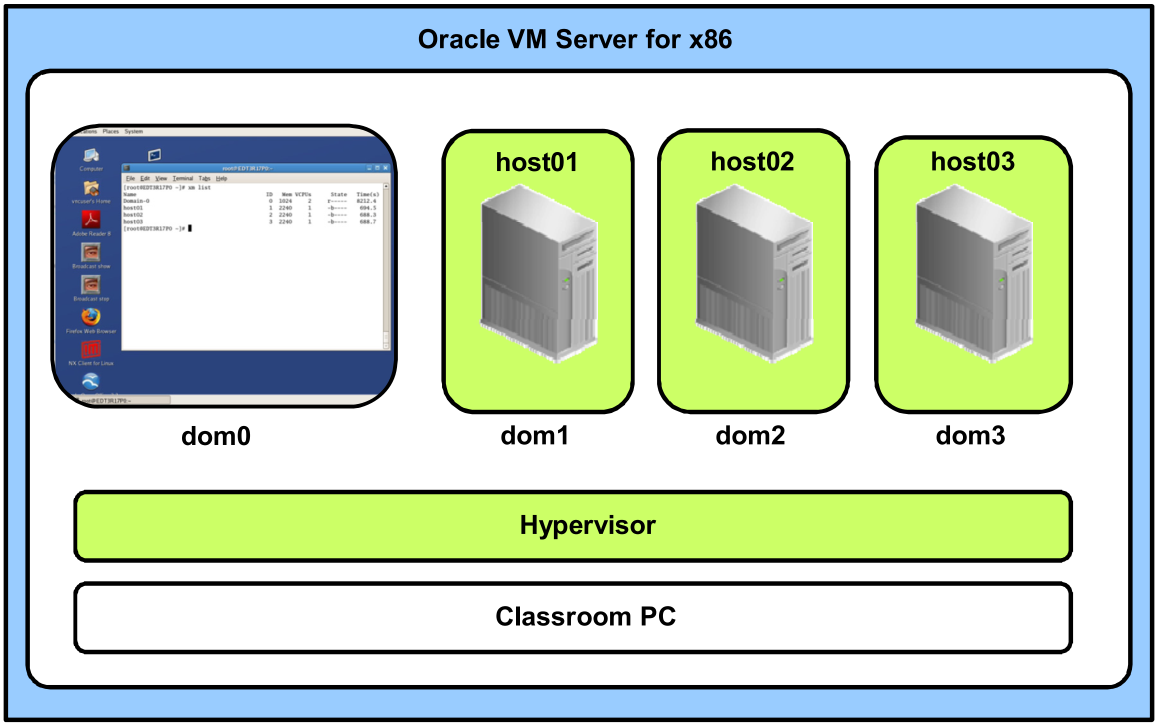
A hypervisor is virtualization software, also known as a virtual machine monitor (VMM), that creates and runs the virtual machines. There are two different types of hypervisors:

- A type 2 hypervisor such as VirtualBox that 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 that 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.

Oracle VM Server for x86 in the Classroom



```
[root@EDT3R17P0 ~]# xm list
Name                                      ID   Mem VCPUs      State   Time(s)
Domain-0                                   0  1024     2     r-----    8212.4
host01                                     1  2240     1     -b----     694.5
host02                                     2  2240     1     -b----     688.3
host03                                     3  2240     1     -b----     688.7
[root@EDT3R17P0 ~]# 
```

ORACLE

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. Activity Guides provide hands-on practice exercises that enforce the material covered in the Student Guides. The Activity Guides tell you which VMs must be running for each practice exercise. Note that the name of a VM does not necessarily have to be the same as the host name of the server that runs within the VM.

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`*`>`: Non-graceful shut down.
  - `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 using 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:

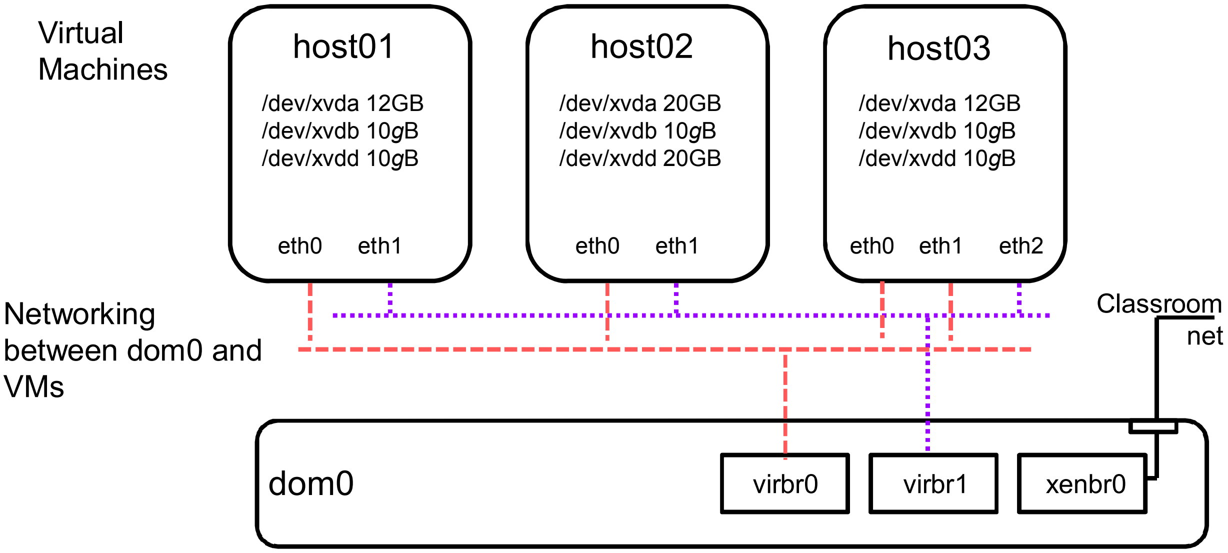
```
    # vncviewer&
    localhost:5903
```

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

```
    localhost:3
```

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

Classroom System Configuration



Virtual Machines

host01
/dev/xvda 12GB
/dev/xvdb 10gB
/dev/xvdd 10gB
eth0 eth1

host02
/dev/xvda 20GB
/dev/xvdb 10gB
/dev/xvdd 20GB
eth0 eth1

host03
/dev/xvda 12GB
/dev/xvdb 10gB
/dev/xvdd 10gB
eth0 eth1 eth2

Networking between dom0 and VMs

Classroom net

dom0
virbr0 virbr1 xenbr0

Public network - 192.0.2.0 - - - - - - - - -

Private network - 192.168.1.0 ····················

Network Configuration

This slide shows the network configuration of each classroom system, as well as the disk devices for each VM guest. Dom0 on each system has a single physical network interface, `eth0`. Three network bridges are configured on Dom0:

- `xenbr0`: Communication to the classroom network is through this bridge. The IP address of `xenbr0` is the same as the IP address assigned to `eth0`.
- `virbr0`: This is 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`: This is a Xen bridge interface used by the VM guests on the `192.168.1` private subnet. This private subnet is used in the OCFS2 practice. The IP address of `virbr1` is `192.168.1.1`.

The host01 and host02 VM guests have the same network configuration. These VM guests have two 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 using DHCP for `eth1`. The `eth1` interface on host02 has an IP address of `192.168.1.102`.

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 example.com us.oracle.com
nameserver 192.0.2.1
nameserver 152.68.154.3
nameserver 10.216.106.3
nameserver 193.32.3.252
```

The example.com search domain is offered by the DNS service running on dom0 (nameserver 192.0.2.1). The us.oracle.com search domain and the three additional name servers provide additional 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 two virtual block devices, each 20 GB in size.

- /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/xvdb: This is a 20 GB utility disk that is used in the Linux Containers practice.

There are three additional virtual machines, not shown in the slide, which are used in specific practices:

- host04: This VM is used in “Practices for Lesson 7: Advanced Software Package Management.” This VM is pre-configured to access Oracle’s Public Yum Server.
- host05: This VM is used in “Practices for Lesson 14: Virtualization with Linux.” You configure this VM to run Kernel-based Virtual Machine (KVM).
- host06: This VM is used in “Practices for Lesson 12: Btrfs File System.” You install Oracle Linux 6.5 on this VM from the UEK Boot ISO.

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
[OL6U5Dom0]
Name="Oracle Linux 6 U5 Dom0 Repo"
baseurl=http://192.0.2.1/repo/OracleLinux/OL6/5/base/x86_64
enabled=1
gpgkey=http://192.0.2.1/repo/OracleLinux/OL6/5/base/x86_64/R
   PM-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-ol6.repo   vm.repo
```

All repositories are disabled in the `public-yum-ol6.repo` file. Only `vm.repo` has an enabled repository, which points to the local Yum Repository on dom0.

On dom0, the Oracle Linux 6.5 ISO files exist in the following directory:

```
# ls /OVS/OL6U5REPO
EFI     Packages      ResilientStorage
EULA    README-en     RPM-GPG-KEY
...
```

The web server URL is pointing to `OL6U5REPO`:

```
# ls -l /var/www/html/repo/OracleLinux/OL6/5/base
lrwxrwxrwx ...  x86_64 -> /OVS/OL6U5REPO
```

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
- 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 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, the `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.

Configuring a DHCP Server

- Install the `dhcp` package.

```
# yum install dhcp
```

- The `/etc/dhcp/dhcpd.conf` configuration file specifies:
  - Options
  - Lease times
  - Subnet declarations with range of IP addresses
- Specify command-line arguments and options in `/etc/sysconfig/dhcpd`. Example:
  - `DHCPDARGS=eth1`
- Start the service after changing the configuration:

```
# service dhcpd start
```

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

```
    # yum install dhcp
```

DHCP Configuration File

The main configuration file for DHCP is `/etc/dhcp/dhcpd.conf`. Use this file to store network information for the clients. The following is a sample `dhcpd.conf` configuration file:

```
    option subnet-mask                 255.255.255.0;
    option domain-name                 "example.com";
    option domain-name-servers         192.0.2.1;
    option broadcast-address           192.168.1.255;
    default-lease-time                 21600;
    max-lease-time                     43200;
    subnet 192.168.1.0 netmask 255.255.255.0 {
         range 192.168.1.200 192.168.1.254;
    }
```

A sample file exists in `/usr/share/doc/dhcp-<`*`version`*`>/dhcpd.conf.sample`.

Options

Information in the `option` lines is sent to each client when it requests a lease. In this example, the `subnet-mask`, `broadcast-address`, `routers` (default gateway IP address), and `DNS server` IP are sent to the client. The default gateway and DNS server are the same system in this example. Each option declaration is terminated with a semicolon (`;`). Run the `man 5 dhcp-options` command 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 includes a range of IP address that the DHCP server can assign to clients. This example defines a range between `192.168.1.200` and `192.168.1.254`.

You can declare multiple subnets and specify parameters within the braces (`{ }`). Parameters specified within the braces apply to the clients on the subnet. Parameters configured outside of a subnet declaration are global and apply to all client systems.

Start the DHCP Service

You can specify command-line options and arguments by placing them in `/etc/sysconfig/dhcpd`. For example, to configure the server to only start the service on a specific interface, edit the `/etc/sysconfig/dhcpd` file and add the name of the interface to the `DHCPDARGS` directive. The following example starts the service on `eth1` only:

```
DHCPDARGS=eth1
```

Start the service as follows. After making any changes to the configuration, restart the DHCP service.

```
# service dhcpd start
```

The server fails to start if the `/var/lib/dhcpd/dhcpd.leases` file does not exist. If the service fails to start, you can use the `touch` command to create the file:

```
# touch /var/lib/dhcpd/dhcpd.leases
```

This file stores the client lease information. Do not edit this file.

To ensure that the `dhcpd` service starts at boot time, enter the following command to enable the service for run levels 2, 3, 4, and 5:

```
# chkconfig dhcpd on
```

Additional DHCP Server Declarations

- Host declarations for static IP address assignment
  - Assign a static IP address to a specific client system.
  - Include the MAC address and the static IP address within the host declaration.
- Shared-network declarations for multiple subnets
  - Group subnets that share the same physical network within the shared-network declaration.
- Group declarations apply global parameters
  - Use to apply global parameters to a group of declarations.
  - Shared networks, subnets, and hosts can be grouped.

Host Declaration

To provide a static IP address to a specific client system, use a `host` declaration and 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;
     }
}
```

Configuring a DHCP Client

1. Install the `dhclient` package.
2. Set `NETWORKING=yes` in `/etc/sysconfig/network`.
3. Set `BOOTPROTO=dhcp` in the `/etc/sysconfig/network-scripts/ifcfg-<device>` file.
4. Optionally, enter any custom configuration information in the DHCP client configuration file, `/etc/dhclient.conf`.
5. Run the `dhclient` command to request a lease from the server.
   – After being configured to use DHCP, this 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:

- Edit `/etc/sysconfig/network-scripts/ifcfg-eth1`
- Set `BOOTPROTO=dhcp`

You also need to set `NETWORKING=yes` in the `/etc/sysconfig/network` file.

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/dhclient.conf`, is only required for custom configurations. A sample file exists in `/usr/share/doc/dhclient-`*`<version>`*`/dhclient.conf.sample`. The following example specifies the use of DHCP on a single interface, `eth1`, which has a MAC address of `00:16:3e:00:02:02`:

```
# cat /etc/dhclient.conf
interface “eth1” {
     send dhcp-client-identifier 1:00:16:3e:00:02:02;
}
```

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 192.168.1.200;
     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 1 2012/02/20 20:18:58;
     rebind 1 2012/02/20 23:15:26;
     expire 2 2012/02/20 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 sub-domain of `com`, `us` is a sub-domain 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
- BIND `named` options are set in `/etc/sysconfig/named`.

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. To install the `bind` package, type:

```
# yum install bind
```

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
- **`/var/named`:** The default directory in which zone files are stored
- **`/etc/named.rfc1912.zones`:** The base configuration file for implementing a caching-only nameserver
- **`/var/named/named.ca`:** A file that contains a list of the 13 root authoritative DNS servers

Use the following commands to ensure that the `named` service starts at boot time and to start the service. Restart `named` after making any configuration changes:

```
# chkconfig named on
# service named start
```

BIND `named` options are set in the `/etc/sysconfig/named` file.

DNS Cache-Only Nameserver

- The default BIND configuration files provide a caching-only nameserver service.
- `/etc/named.conf` defines:
  - `options`: Global server configuration options
  - `logging`: Enables logging
  - `zone`: Specifies authoritative servers for the root domain (`/var/named/named.ca`)
  - `include`: Includes `/etc/named.rfc1912.zones`
- `/etc/named.rfc1912.zones`
  - Specifies five predefined zones

A caching-only nameserver is installed by default with the `bind` package. This DNS cache nameserver is not authoritative; it only stores the results of queries in memory. The default BIND configuration files provide this caching-only nameserver service.

The following is the default `/etc/named.conf`:

```
options {
     listen-on port 53 { 127.0.0.1; };
     listen-on-v6 port 53 { ::1; };
     directory  “/var/named”;
     dump-file  “/var/named/data/cache_dump.db”;
     statistics-file “/var/named/data/named_stats.txt”;
     memstatistics-file
     “/var/named/data/named_mem_stats.txt”;
     allow-query     { localhost; };
     recursion yes;
```

Default `/etc/named.conf` file, continued:

```
        dnssec-enable yes;
        dnssec-validation yes;
        dnssec-lookaside auto;

        /* Path to ISC DLV key */
        bindkeys-file “/etc/named.iscdlv.key”;
    };

    logging {
        channel default_debug {
            file “data/named.run”;
            severity dynamic;
        };
    };

    zone “.” IN {
        type hint;
        file “named.ca”;
    };

    include “/etc/named.rfc1912.zones”;
```

/etc/named.conf

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 (only `localhost` in this example) 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.
- **`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) using the `/etc/named.iscdlv.key`

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. The value `dynamic` means assume the global level defined by either the command-line parameter `-d` or by running the `rndc trace` command.

The `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 `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.

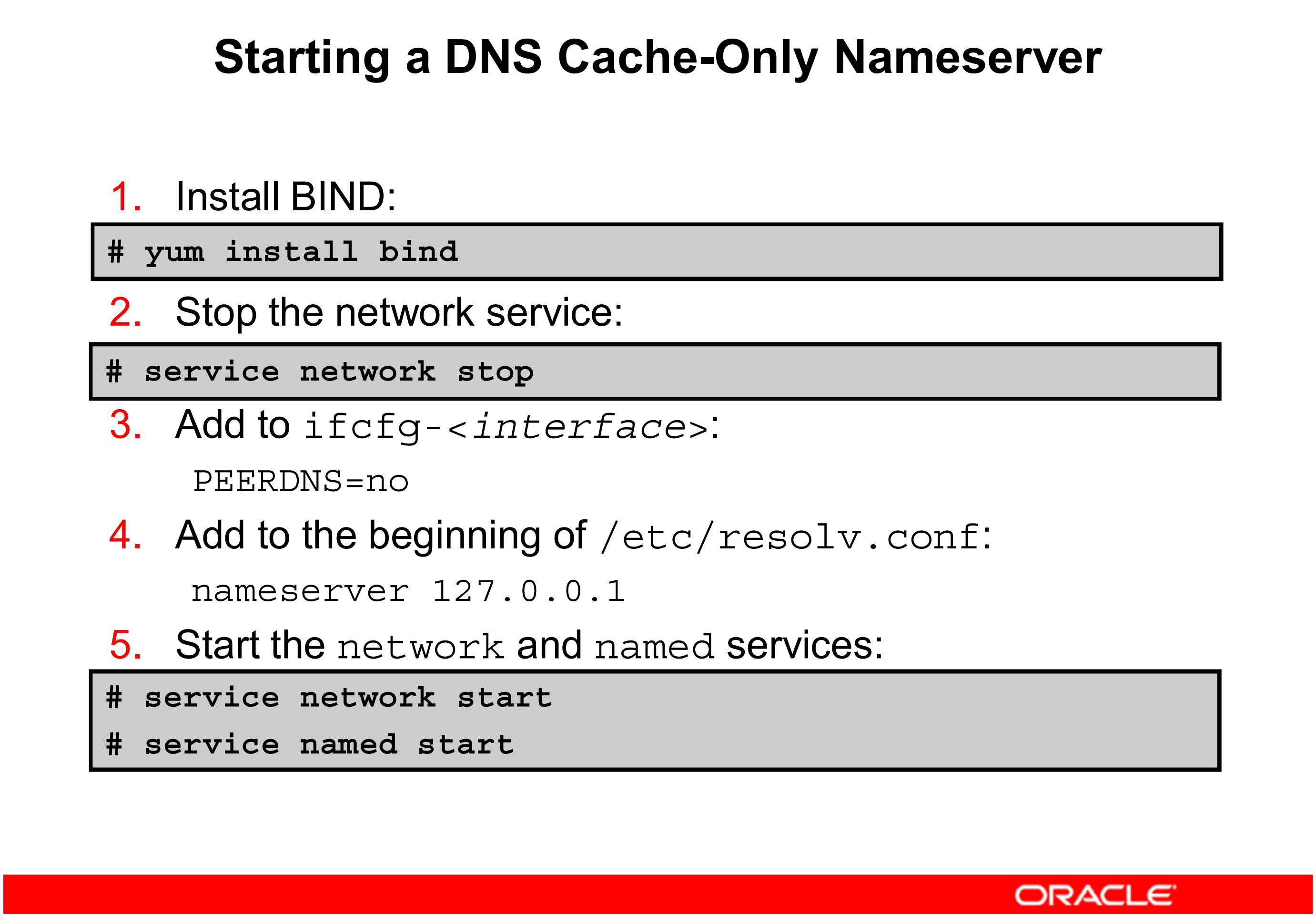
/etc/named.rfc1912.zones

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 this 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
- **`1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.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

Zone options are included for each of these five zones:

- **`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 (`/var/named` in this example)
- **`allow-update`**: Specifies which hosts are allowed to dynamically update information in their zone. Dynamic updates are set to `none` for these zones, meaning they are not allowed.



Starting a DNS Cache-Only Nameserver

1. Install BIND:

```
# yum install bind
```

2. Stop the network service:

```
# service network stop
```

3. Add to `ifcfg-<interface>`:

```
PEERDNS=no
```

4. Add to the beginning of `/etc/resolv.conf`:

```
nameserver 127.0.0.1
```

5. Start the `network` and `named` services:

```
# service network start
# service named start
```

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. Shut down the active network interfaces:

```
# service network stop
```

3. Add the following line to the `/etc/sysconfig/network-scripts/ifcfg-<interface>` file. This line prevents `dhclient` from overwriting `/etc/resolv.conf` when starting the `network` service while using DHCP:

```
PEERDNS=no
```

4. 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
```

5. Start the `network` and `named` services:

```
# service network start
# service named start
```

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`

ORACLE

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 using the format: `IN NS dns.example.com.`

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

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`.

Normal, or forward, resolution returns an IP address when the domain name is known. The zone characteristics are defined in `/etc/named.conf`. Example:

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

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
- **`file`**: Specifies `master-example.com` as the name of the zone file and its location
- **`allow-update`**: This example uses a Transaction SIGnatures (TSIG) 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

The following is an example of the `A` (address) records in the `master-example.com` zone file:

```
# cat /var/named/data/master-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
```

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
```

host and dig Utilities

- host and dig are command-line tools to perform DNS lookups.
- The host command has more options.
- Examples of queries using host:

```
# host
# host -a dns.example.com
# host -a host01
# host -a 192.0.2.101
```

- Examples of queries using dig:

```
# dig dns.example.com
# dig -x 192.0.2.101
# dig example.com NS
# dig example.com A
```

The host utility and the dig utility 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, should have you learned how to:

- Describe DHCP
- Configure 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
- DNS configuration

ORACLE

3

Authentication and Directory Services

Objectives

After completing this lesson, you should be able to:

- Describe authentication options
- Describe the Authentication Configuration Tool
- Describe NIS
- Configure NIS server and NIS client
- Configure NIS authentication
- Describe LDAP
- Describe OpenLDAP
- Describe OpenLDAP server and client utilities
- Configure LDAP authentication
- Configure Winbind authentication
- Configure Kerberos authentication
- Describe and configure 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)
  - Winbind
- Authentication Configuration Tool:
  - 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, then 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 using services such as Lightweight Directory Access Protocol (LDAP), Network Information Service (NIS), and Winbind. Additionally, both LDAP and NIS data files can use Kerberos authentication.

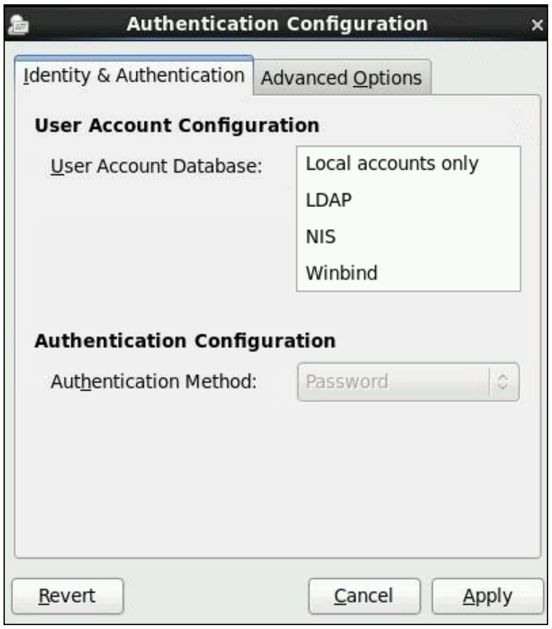
NIS simplifies the maintenance of common administration files such as `/etc/passwd` by keeping them in a central database and having clients retrieve information from this database server.

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 tool for selecting user databases and configuring associated authentication options—the Authentication Configuration Tool,

Authentication Configuration Tool

```
system-config-authentication
```

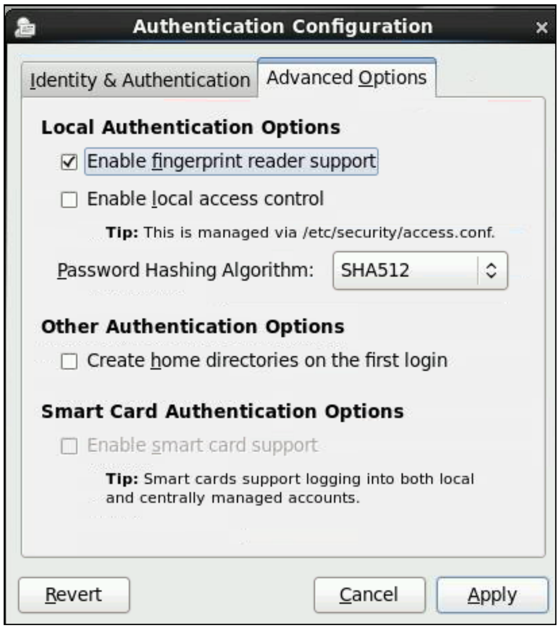


User Account Configuration

User Account Database: Local accounts only / LDAP / NIS / Winbind

Authentication Configuration

Authentication Method: Password



Local Authentication Options

☑ Enable fingerprint reader support

□ Enable local access control

Tip: This is managed via /etc/security/access.conf.

Password Hashing Algorithm: SHA512

Other Authentication Options

□ Create home directories on the first login

Smart Card Authentication Options

□ Enable smart card support

Tip: Smart cards support logging into both local and centrally managed accounts.

To use the Authentication Configuration Tool, enter the command:

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

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 four options for the User Account Database field:

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

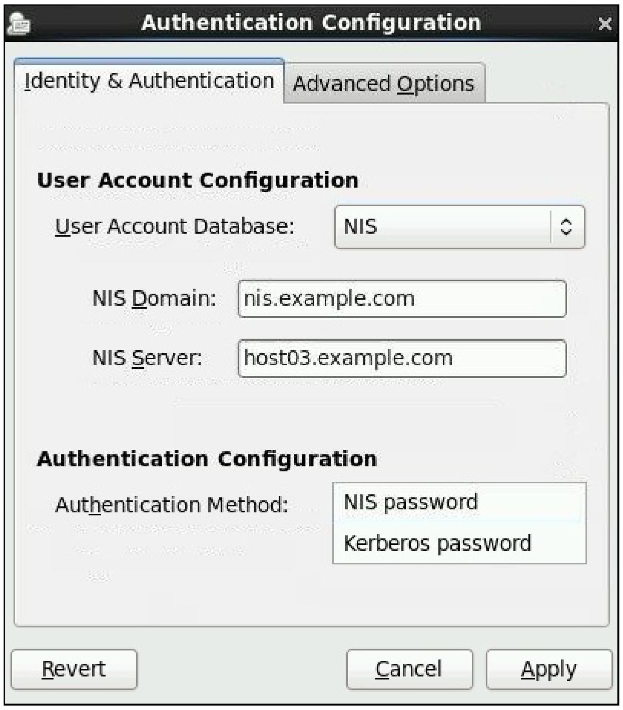
Additional User Account Configuration fields appear, depending on which user account database is selected. The Authentication Configuration section of the page also changes, depending on which user account database is selected.

Advanced Options

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

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



User Account Configuration

User Account Database: NIS

NIS Domain: nis.example.com

NIS Server: host03.example.com

Authentication Configuration

Authentication Method: NIS password
Kerberos password

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
- 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 non-secure network to prove their identity to one another in a secure manner.

NIS Maps

- NIS maps are indexed administrative files within an NIS domain.
- NIS maps are generated from standard administrative “source” files such as:
  - `/etc/hosts`
  - `/etc/passwd`
  - `/etc/shadow`
  - `/etc/group`
- You define which maps to build in `/var/yp/Makefile`.
- Maps are generated by using the `ypinit` command.
- Some source files generate two maps, for example:
  - `passwd.byname`: `/etc/passwd` indexed on username
  - `passwd.byuid`: `/etc/passwd` indexed on UID

The administrative files within an NIS domain are called NIS maps. NIS maps are generated from standard configuration “source” files such as:

- **`/etc/hosts`**: Maps IP addresses to host names
- **`/etc/passwd`**: Lists user information
- **`/etc/shadow`**: Provides shadow passwords for users
- **`/etc/group`**: Defines groups and group members
- **`/etc/gshadow`**: Provides shadow passwords for groups
- **`/etc/networks`**: Maps IP network addresses to network names
- **`/etc/services`**: Maps service names and port numbers
- **`/etc/rpc`**: Maps RPC program names and numbers
- **`/etc/protocols`**: Maps protocol numbers to protocol names

Source files are converted to dbm-format files called maps. Each map is indexed on one field. Records from these indexed maps are retrieved by specifying a value from the indexed field. Some source files have two maps. For example, `/etc/passwd` has two maps:

- **`passwd.byname`**: Indexed on username
- **`passwd.byuid`**: Indexed on UID

NIS was originally called Yellow Pages but was renamed by Sun Microsystems. The names of NIS utilities, files, and directories still begin with the letters “yp” for yellow pages.

You specify which NIS maps to build in `/var/yp/Makefile`. NIS maps are created by running the `ypinit –m` command on the server and are stored in the `/var/yp/`*`domainname`* directory on the server. The following is a sample listing of NIS maps:

```
# ls /var/yp/nis.example.com
group.bygid   mail.aliases  protocols.byname   services.byname
group.byname  netid.byname  protocols.bynumber hosts.byaddr
passwd.byname rpc.byname    hosts.byname.      passwd.byuid
rpc.bynumber
```

You can assign nicknames to maps to make it easier to refer to them. The `/var/yp/nicknames` file contains a list of commonly used nicknames:

```
# cat /var/yp/nicknames
passwd     passwd.byname
group      group.byname
networks   networks.byaddr
hosts      hosts.byname
protocols  protocols.bynumber
services   services.byname
aliases    mail.aliases
ethers     ethers.byname
```

NIS Server Configuration

- Install the `ypserv` package.
- Specify the NIS domain name in `/etc/sysconfig/network`:
  - `NISDOMAIN=`*`nisdomainname`*
- The main configuration file for the NIS server is:
  - `/etc/ypserv.conf`
- Specify which hosts are allowed access to the NIS server in:
  - `/var/yp/securenets`
- Specify which NIS maps to create in:
  - `/var/yp/Makefile`
- Start the services: `ypserv`, `ypxfrd`, `yppasswdd`
- Run `ypinit -m` to create the maps.

To begin configuring a system as an NIS server, install the `ypserv` package:

```
# yum install ypserv
```

Specify the NIS domain name by creating an entry in the `/etc/sysconfig/network` file:

```
NISDOMAIN=nisdomainname
```

The main configuration file for the NIS server, `/etc/ypserv.conf`, specifies server options and access rules. Some options are set by default. Options have the following format:

```
option: value
```

Access rules specify which hosts and domains can access which NIS maps. Access rules have the following format:

```
host:domain:map:security
```

The *`host`* and *`domain`* fields specify the IP address and NIS domain the rule applies to, respectively. The *`map`* field is the name of the map the rule applies to. The *`security`* field is either `none` (always allow access), `port` (allow access if from port < 1024), or `deny` (never allow access). The following example grants access to anyone logging in from an IP address in the range of `192.0.2.1` to `192.0.2.255` (spaces around colons are required):

```
192.0.2.1/24 : * : * : none
```

Create the /var/yp/securenets file to restrict access to your NIS server to certain hosts. This file prevents unauthorized systems from sending RPC requests to the NIS server and retrieving NIS maps. NIS accepts requests from systems whose IP addresses are included in this file and ignores requests from other systems. Each line contains a netmask and IP address. The following examples accept requests from the local system and from systems with IP addresses starting with 192.0.2:

```
255.255.255.255 127.0.0.1
255.255.255.0   192.0.2.0
```

Edit /var/yp/Makefile to set options and specify which NIS maps to create. The all: target in /var/yp/Makefile specifies the maps to create:

```
all:  passwd group hosts rpc services netid protocols mail \
  # netgrp shadow publickey networks ethers bootparams printcap \
  # amd.home auto.master auto.home auto.local. passwd.adjunct \
  # timezone locale netmasks
```

Run chkconfig to cause the following services to start when the system enters multiuser mode:

```
# chkconfig ypserv on
# chkconfig ypxfrd on
```

The rpc.ypxfrd daemon is used to speed up the distribution of very large NIS maps from an NIS master to NIS slave servers. It runs on the master server only, not the slave server.

```
# chkconfig yppasswdd on
```

The rpc.yppasswdd daemon is the password update daemon and allows normal NIS users to change their password in an NIS shadow map.

Start the three services:

```
# service ypserv start
# service ypxfrd start
# service yppasswdd start
```

Run the command ypinit on the master server to create the NIS maps. The ypinit utility builds the domain subdirectory in /var/yp for the domain. After building the subdirectory, ypinit gathers information from the passwd, group, hosts, rpc, services, netid, protocols, and mail files (or whatever was targeted as all: in /var/yp/Makefile) in the /etc directory and builds the database. Use the absolute path name of ypinit and include the -m option when running the command on the master server.

```
# find / -name ypinit
/usr/lib64/yp/ypinit
# /usr/lib64/yp/ypinit -m
```

NIS Client Configuration

- Install the following packages:

```
# yum install yp-tools
# yum install ypbind
```

- Specify NIS domain name in `/etc/sysconfig/network`:
  – `NISDOMAIN=`*`nisdomainname`*
- Specify the NIS server in `/etc/yp.conf`:
  – `ypserver` *`hostname`*
- Start the `ypbind` service:

```
# service ypbind start
```

To begin configuring a system as an NIS client, install the following packages:

```
# yum install yp-tools
# yum install ypbind
```

Specify the name of the NIS domain that the system belongs to, by creating an entry in the `/etc/sysconfig/network` file:

```
NISDOMAIN=nisdomainname
```

You can use the `nisdomainname` command to view and set the name as well, but this is not persistent across a reboot:

```
# nisdomainname nis.example.com
# nisdomainname
nis.example.com
```

Specify the NIS server in the `/etc/yp.conf` file. You can specify one or more NIS servers (masters and/or slaves):

```
ypserver hostname
```

Start the `ypbind` service:

```
# service ypbind start
```

Implementing NIS Authentication

- Select NIS as the user account database by using either of the following:
  - Authentication Configuration Tool
  - Command line: `authconfig`
- Add a new user, and create a password.
- Update the NIS maps.
- Update `/etc/nsswitch.conf`.
- Log in as `new_user` from an NIS client to test the authentication.
- NIS command summary:
  - `yppasswd`: Change the NIS password.
  - `ypcat`: Display the contents of an NIS map.
  - `ypmatch`: Search an NIS map.
  - Others

After configuring the NIS server and NIS clients, open the Authentication Configuration Tool by entering the command:

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

Select NIS as the user account database. The NIS Domain and NIS Server fields are automatically filled in if NIS is configured. NIS authentication can also be enabled and configured from the command line by using the `authconfig` command. The syntax is as follows:

```
authconfig --enablenis --nisdomain nisdomainname --nisserver
hostname --update
```

For example, if `nisdomain` is `nis.example.com` and the NIS server is `host03.example.com`, enter the following:

```
# authconfig --enablenis --nisdomain nis.example.com --nisserver
host03.example.com --update
```

Add an NIS User

To create a new NIS user account, add the user on the NIS server:

```
# useradd new_user
```

This command updates the `/etc/passwd` file and creates a home directory on the NIS server.

Create a password for the new NIS user:

```
# passwd new_user
```

This command updates the `/etc/shadow` file with the hashed password.

Update the NIS maps by running `ypinit -m` from the NIS server:

```
# /usr/lib64/yp/ypinit -m
```

This command updates the NIS maps, adding `new_user` to the `passwd` and `shadow` maps.

Update `nsswitch.conf`

Whether a system uses NIS, local files, DNS, or a combination as the source of information, and in what order, is determined by the `/etc/nsswitch.conf` file. To query the `passwd`, `shadow`, and `group` NIS maps first, make the following changes to `/etc/nsswitch.conf`:

```
passwd:    nis files
shadow:    nis files
group:     nis files
```

To test the authentication, log in as `new_user` from a remote NIS client.

```
host02 login: new_user
Password:
No directory /home/new_user!
Logging in with home = "/"
```

You can log in, but the home directory, `/home/new_user`, is on the NIS server. Use NFS or automounter to mount the remote home directory.

Use the `yppasswd` command to change the NIS password:

```
# yppasswd
Changing NIS account information for new_user on
host03.example.com.
Please enter old password:
Changing NIS password for new_user on host03.example.com.
Please enter new password:
Please retype new password:
The NIS password has been changed on host03.example.com.
```

Summary of NIS Commands

The following summarizes the various NIS commands:

- **`ypinit`:** Update the NIS server maps.
- **`ypwhich`:** Display the NIS server name.
- **`ypcat`:** Display the contents of an NIS file.
  - Example: `ypcat passwd`
- **`ypmatch`:** Search an NIS map.
  - Example: `ypmatch new_user passwd`
- **`yptest`:** Test the NIS configuration.
- **`yppasswd`:** Change the NIS password.
- **`yppush`:** Update NIS slaves from the NIS master.

Complete NIS Practices

- You can optionally stop the lecture portion of this lesson and complete the NIS practices in the activity guide.
- The NIS practices include the following:
  - Practice 3-1: Configuring an NIS Server
  - Practice 3-2: Configuring an NIS Client
  - Practice 3-3: Implementing NIS Authentication
  - Practice 3-4: Testing NIS Authentication
  - Practice 3-5: Auto-mounting a User Home Directory
- After completing the NIS practices, you can continue with the lecture.

Because this is a long lesson, you can optionally stop the lecture here and perform the NIS practices. You can then resume the lecture and complete the remaining LDAP practices after reaching the end of this lesson.

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.

ORACLE

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: 500
gidNumber: 500
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: 556
memberUid: oracle
memberUid: student1
memberUid: student2
```

LDIF (LDAP Data Interchange Format) 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`.
- To start the service:

```
# service slapd start
```

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
- **`mod_authz_ldap`:** Contains the LDAP authorization module for the Apache HTTP Server

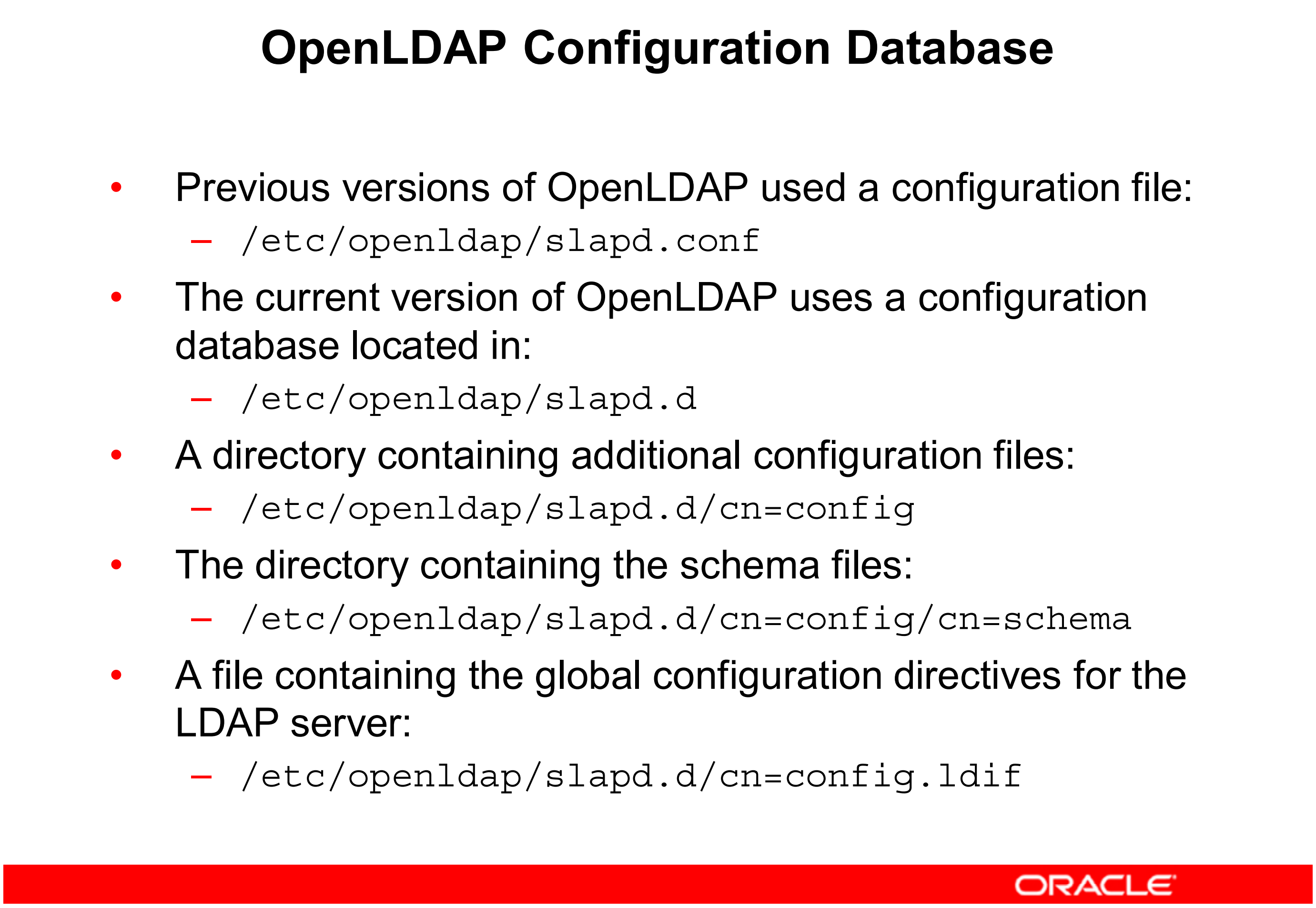
Install the packages by using the `yum install <package_name>` command.

Run `chkconfig` to cause the `slapd` service to start when the system enters multiuser mode:

```
# chkconfig slapd on
```

Start the `slapd` service:

```
# service slapd start
```



OpenLDAP Configuration Database

- 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`
- A directory containing additional configuration files:
  - `/etc/openldap/slapd.d/cn=config`
- The directory containing the schema files:
  - `/etc/openldap/slapd.d/cn=config/cn=schema`
- A file containing the global configuration directives for the LDAP server:
  - `/etc/openldap/slapd.d/cn=config.ldif`

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/slapd.d/cn=config/cn=schema`:** The directory containing the schema files

The `/etc/openldap/slapd.d` directory also contains LDAP definitions that were previously located in the following directory:

`/etc/openldap/schema`

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/admin/schema.html`

A `README` file, containing descriptions of the installed schema files, is located at:

```
/usr/share/doc/openldap-servers-version/README.schema
```

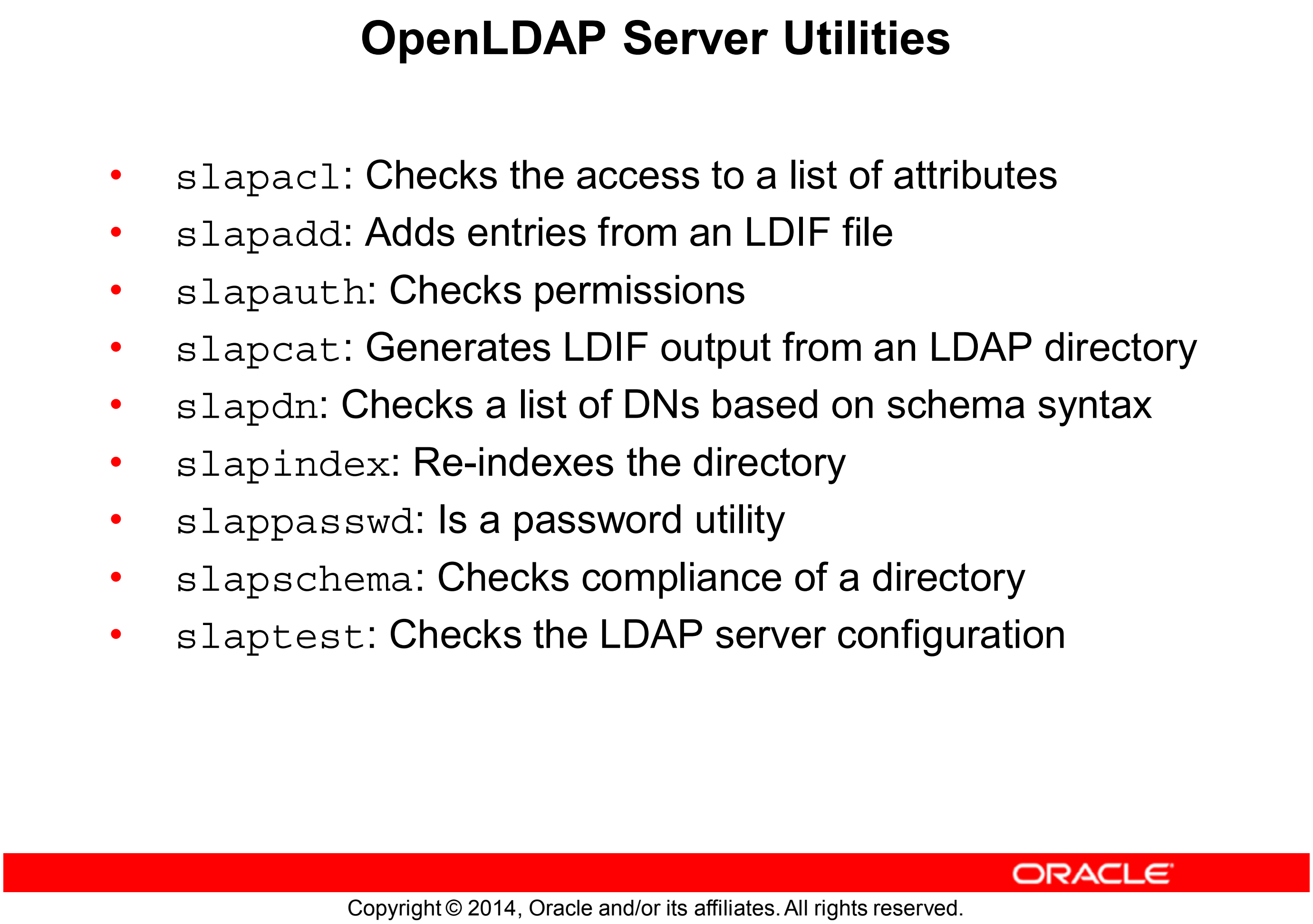
Additional configuration files and directories include:

- `/etc/openldap/slapd.d/cn=config.ldif`: A file containing the global configuration directives for the LDAP server
- `/etc/openldap/slapd.d/cn=config`: A directory containing additional configuration files

Refer to `man slapd-config` for a description of all the configuration directives.

The *OpenLDAP Software Administrator's Guide* also describes the configuration. A copy of this administrator's guide is located at:

http://www.openldap.org/doc/admin24/OpenLDAP-Admin-Guide.pdf



OpenLDAP Server Utilities

- slapacl: Checks the access to a list of attributes
- slapadd: Adds entries from an LDIF file
- slapauth: Checks permissions
- slapcat: Generates LDIF output from an LDAP directory
- slapdn: Checks a list of DNs based on schema syntax
- slapindex: Re-indexes the directory
- slappasswd: Is a password utility
- slapschema: Checks compliance of a directory
- slaptest: Checks the LDAP server configuration

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

OpenLDAP Client Utilities

- `ldapadd`: Adds entries to an LDAP directory
- `ldapmodify`: Modifies entries in an LDAP directory
- `ldapcompare`: Compares a given attribute with an entry
- `ldapdelete`: Deletes entries from an LDAP directory
- `ldapexop`: Performs extended LDAP operations
- `ldapmodrdn`: Modifies the RDN value of an 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

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 migrationtools
```

  - The `migrationtools` package is optional but allows you to migrate information from existing name services.
- Copy sample files into appropriate directories:
  - Copy the `slapd.conf` file into `/etc/openldap`.
  - Copy the sample `DB_CONFIG` file into `/var/lib/ldap`.
  - Change ownership and group on both to `ldap`.
- Use `slappasswd` to create an encrypted password.
- Update `slapd.conf` with encrypted password and domain.
- Use `slaptest` to convert configuration file to database.
- Start the `slapd` service.

To configure an OpenLDAP server, install the following packages:

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

The `openldap-clients` package is also installed as a dependency. The `migrationtools` package is optional but it provides a set of Perl scripts, which allows you migrate users, groups, and other information from existing name services.

The current version of OpenLDAP uses a configuration database, which can be an edited directory. However a configuration file, used on earlier versions, is provided, which can be edited and then converted to the new format. To use the sample `slapd.conf` file, copy the file from the `/usr/share/openldap-servers` directory into the `/etc/openldap` directory:

```
    # cp usr/share/openldap-servers/slapd.conf.obsolete
    /etc/openldap/slapd.conf
```

Use the `slappasswd` command to create an encrypted user password:

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

The encrypted password shown is a sample only.

After providing a password of your choice, the encrypted password is displayed. Edit the `/etc/openldap/slapd.conf` file and set the `rootpw` directive to the encrypted password:

```
    rootpw          {SSHA}4SOiIaqwQYftwkdr1FbqVNEmI3Am0wJT
```

Also in the `slapd.conf` file change all occurrences of `dc=my-domain` to the domain name component of your OpenLDAP server, for example:

```
    suffix     "dc=example,dc=com"
    rootdn     "cn=Manager,dc=example,dc=com"
```

A sample `DB_CONFIG` file is also provided in the `/usr/share/openldap-servers` directory. Copy this sample file into the `/var/lib/ldap` directory:

```
    # cp /usr/share/openldap-servers/DB_CONFIG.example
    /var/lib/ldap/DB_CONFIG
```

The `/var/lib/ldap` directory (and contents) needs the owner and group set to `ldap`. Use the `chown –R ldap.ldap` command to change ownership:

```
    # chown –R ldap.ldap /var/lib/ldap
```

Before converting the `slapd.conf` file to the new configuration database format, remove the contents of the `/etc/openldap/slapd.d` directory:

```
    # rm -rf /etc/openldap/slapd.d/*
```

Use the `slaptest` command to convert the configuration to the new format.

```
    # slaptest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d
```

The `/etc/openldap/slad.d` directory also needs the owner and group set to `ldap`. Use the `chown –R ldap.ldap` command:

```
    # chown –R ldap.ldap /etc/openldap/slapd.d
```

Use the `service` command to start the `slapd` service.

```
    # sevice slapd start
```

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.

You also need to populate the LDAP directory as discussed in the next slide.

Populating an OpenLDAP Directory

- Create a base domain text file in the LDIF format.
- Use the `ldapadd` command to import the base information to the LDAP directory.
- 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.
  - Extract existing information into text files.
  - Use the `migrate_passwd.pl` command to migrate user information LDAP format.
- Use the `ldapadd` command to import migrated information to the LDAP directory.

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 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
```

Use the `ldapadd` command to import the base information to the LDAP directory. Provide the password created with the `slappasswd` command (for example, `oracle`):

```
# ldapadd -x -W -D “cn=Manager,dc=example,dc=com” -f base.ldif
Enter LDAP Password: oracle
adding new entry “dc=example,dc=com”
adding new entry “ou=People,dc=example,dc=com”
adding new entry “ou=Group,dc=example,dc=com”
```

The `migrationtools` Perl scripts allow you to migrate users, groups, and other information from existing name services. You first must update the `migrate_common.ph` file for the correct domain. The `migrationtools` files are installed in the `/usr/share/migrationtools` directory. The following example sets two directives to the correct 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 and groups with UID and GID in the 500 range and writes the output to `passwd` and `group` text files:

```
# grep “:5[0-9][0-9]” /etc/passwd > passwd
# grep “:5[0-9][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:

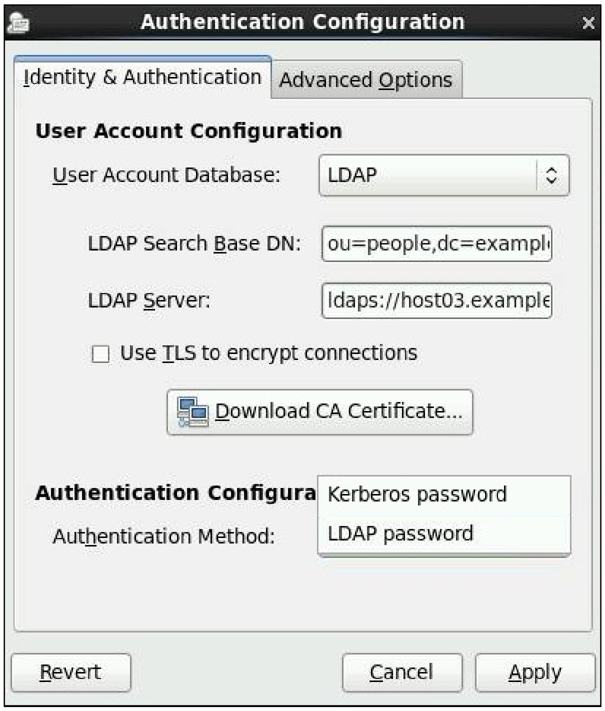
```
# ldapadd -x -W -D “cn=Manager,dc=example,dc=com” -f users.ldif
# ldapadd -x -W -D “cn=Manager,dc=example,dc=com” -f group.ldif
```

Use the `ldapsearch` command to search for specific entries in the LDAP directory. For example, to display the `students` group entry:

```
# ldapsearch –x “cn=students” -b “dc=example,dc=com”
...
# students, Group, example.com
dn: cn=students,ou=Group,dc=example,dc=com
objectClass: posixGroup
objectClass: top
cn: students
userPassword:: e2NyeXB0...
gidNumber: 556
memberUid: oracle
memberUid: student1
memberUid: student2
...
```

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


Authentication Configuration
Identity & Authentication
Advanced Options
Download CA Certificate...
Revert
Cancel
Apply

User Account Configuration

User Account Database: LDAP

LDAP Search Base DN: ou=people,dc=exampl

LDAP Server: ldaps://host03.example

□ Use TLS to encrypt connections

Authentication Configura

Authentication Method: Kerberos password
LDAP password

To configure LDAP authentication, in the Authentication Configuration Tool, 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:389`

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
```

Configuring User Authentication from an OpenLDAP Client

- Install the following packages on the client:
  - `openldap-clients`, `pam_ldap`, `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/`
  - Update `/etc/nslcd.conf` and `/etc/pam_ldap.conf`
- 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.
- Set `USELDAP=yes` in `/etc/sysconfig/authconfig`.
- 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:

```
openldap-clients
pam_ldap
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/
```

You also need to configure the `/etc/nslcd.conf` and `/etc/pam_ldap.conf` files to point to the LDAP server, for example (both directives are in lowercase in these files):

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

The `pam_ldap` module allows OpenLDAP clients to authenticate against LDAP directories, and to change their passwords in the directory. 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 system-auth
...
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        sufficient    pam_ldap.so use_first_pass

account     sufficient    pam_succeed_if.so uid < 500 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:

```
passwd:   files ldap
shadow:   files ldap
group:    files ldap
```

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

```
USELDAP=yes
```

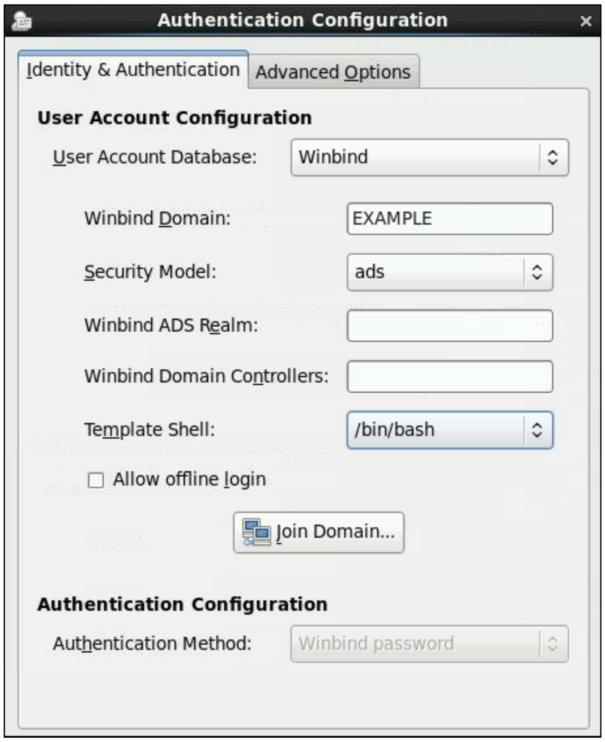
Use the `service` command to start the `nslcd` service:

```
# service nslcd start
```

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



Authentication Configuration

Identity & Authentication | Advanced Options

User Account Configuration

User Account Database: Winbind

Winbind Domain: EXAMPLE

Security Model: ads

Winbind ADS Realm:

Winbind Domain Controllers:

Template Shell: /bin/bash

□ Allow offline login

Join Domain...

Authentication Configuration

Authentication Method: Winbind password

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

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

This package includes the `winbindd` daemon, which is 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.

Enter the Windows domain to connect to. Select the security model to use for Samba clients. The security model options are `user`, `server`, `domain`, and `ads`. Selection of a security model updates the `security` directive in the `[global]` section of `/etc/samba/smb.conf`.

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
  - 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.

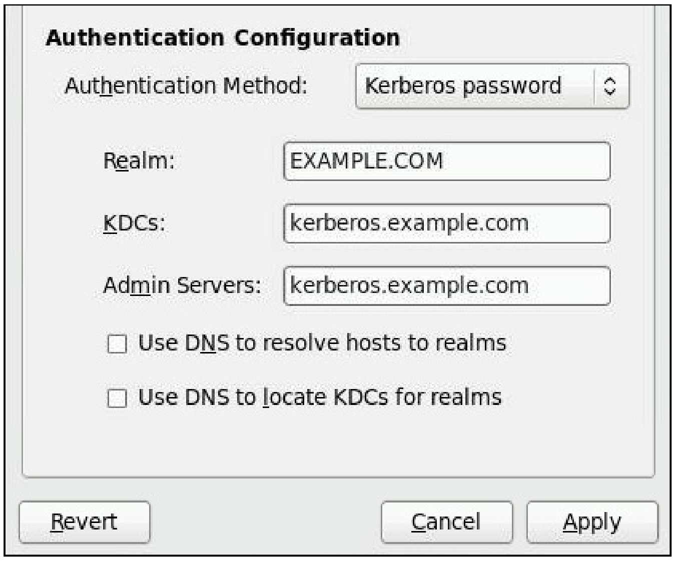
Assuming that Kerberos has been initialized, click the Join Domain button to attempt to join the domain immediately. Alternatively, use the following command to join:

```
    # 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



Authentication Configuration

Authentication Method: Kerberos password

Realm: EXAMPLE.COM

KDCs: kerberos.example.com

Admin Servers: kerberos.example.com

□ Use DNS to resolve hosts to realms

□ Use DNS to locate KDCs for realms

Both LDAP and NIS 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
# yum install krb5-workstation
```

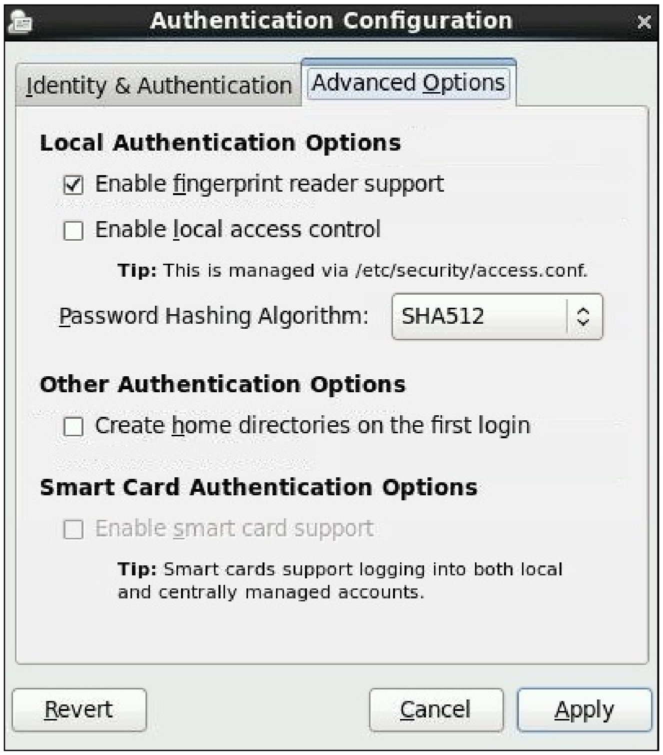
Select Kerberos password as the Authentication Method. You are then prompted for information required to connect to the Kerberos realm:

- **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.

Configuring Advanced Options



Local Authentication Options

☑ Enable fingerprint reader support

☐ Enable local access control

Tip: This is managed via /etc/security/access.conf.

Password Hashing Algorithm: SHA512

Other Authentication Options

☐ Create home directories on the first login

Smart Card Authentication Options

☐ Enable smart card support

Tip: Smart cards support logging into both local and centrally managed accounts.

Clicking the Advanced Options tab allows you to 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 : origin
```

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 `origin` 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.

System Security Services Daemon

- System Security Services Daemon (SSSD) provides access to remote identity and authentication providers.
- SSSD acts as an intermediary between local clients and any back-end providers.
  - Reduces the load on back-end providers
  - Allows offline authentication
  - Allows for single-user accounts
- Install the packages:

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

- Start the service:

```
# authconfig --enablesssd --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
# yum install sssd-client
```

To cause SSSD to start when the system enters multiuser mode, enter either of the following:

```
# chkconfig sssd on
# 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:
  - Specify specialized services that run together with SSSD.
  - Specify 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
```

- Identity provider: `ldap`, `local`, or `proxy`
- Authentication provider: `ldap`, `krb5`, `proxy`, or `none`
- Provider-specific directives
- 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`:** Support 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 are 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:

```
# service sssd start
```

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 NIS
- Configure NIS server and NIS client
- Configure NIS authentication
- Describe LDAP
- Describe OpenLDAP
- Describe OpenLDAP server and client utilities
- Configure LDAP authentication
- Configure Winbind authentication
- Configure Kerberos authentication
- Describe and configure SSSD services and domains

Practice 3: Overview

The practices for this lesson cover the following:

- Configuring an NIS server
- Configuring an NIS client
- Implementing NIS authentication
- Testing the NIS authentication
- Auto-mounting the user home directory
- Configuring an OpenLDAP server
- Implementing OpenLDAP authentication
- Authenticating from an OpenLDAP client

4

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

Apache HTTP Server

- Apache HTTP Web Server is included with Oracle Linux.
- Install the package:

```
# yum install httpd
```

- Start the HTTP daemon:

```
# service httpd start
```

- The main configuration file is:
  - /etc/httpd/conf/httpd.conf
- The auxiliary configuration directory is:
  - /etc/httpd/conf.d
- Check for configuration errors:

```
# service httpd configtest
```

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
```

Start the HTTP daemon by entering the following command:

```
# service httpd start
```

To ensure that `httpd` starts at boot time, enter the following command to enable the service for run levels 2, 3, 4, and 5:

```
# chkconfig httpd on
```

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.

Restart the `httpd` service after making any configuration changes. You can check for configuration errors by running the following command:

```
# service httpd configtest
```

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/current/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. An example follows:

```
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.

```
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

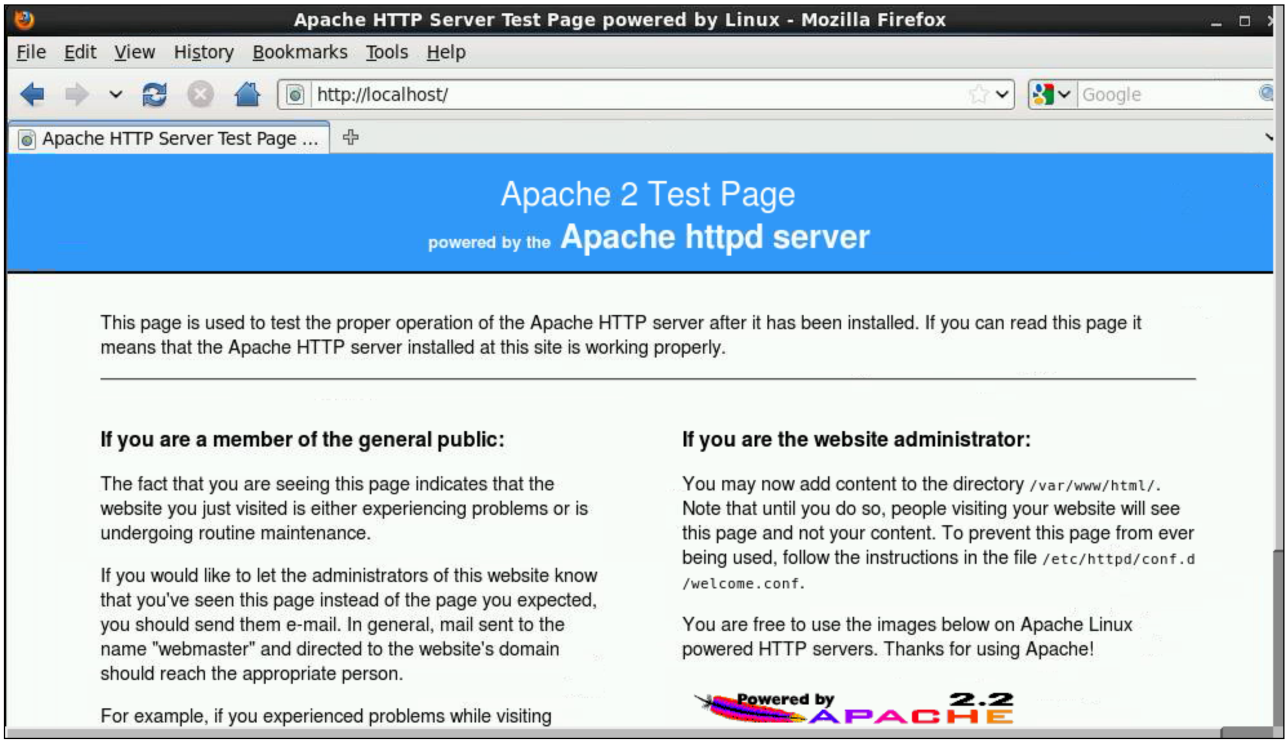
Specifies the order in which `Allow` and `Deny` directives are evaluated. `deny,allow` evaluates `deny` directives first and then evaluates `allow` directives. The following example grants access to clients from the `example.com` domain only, by first denying access to all and then allowing it from `.example.com`:

```
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


Apache HTTP Server Test Page powered by Linux - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/
Google
Apache HTTP Server Test Page ...

Apache 2 Test Page
powered by the Apache httpd server

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the images below on Apache Linux powered HTTP servers. Thanks for using 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
- `<Limit` *`method`*`>`
  - Limits access control directives to specified methods
- Containers can be nested.

Apache containers are special configuration directives that group other directives. Containers use XML-style tags, meaning that the beginning of a container is `<`*`name`*`>` and the end is `</`*`name`*`>`. The 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 `.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] …>

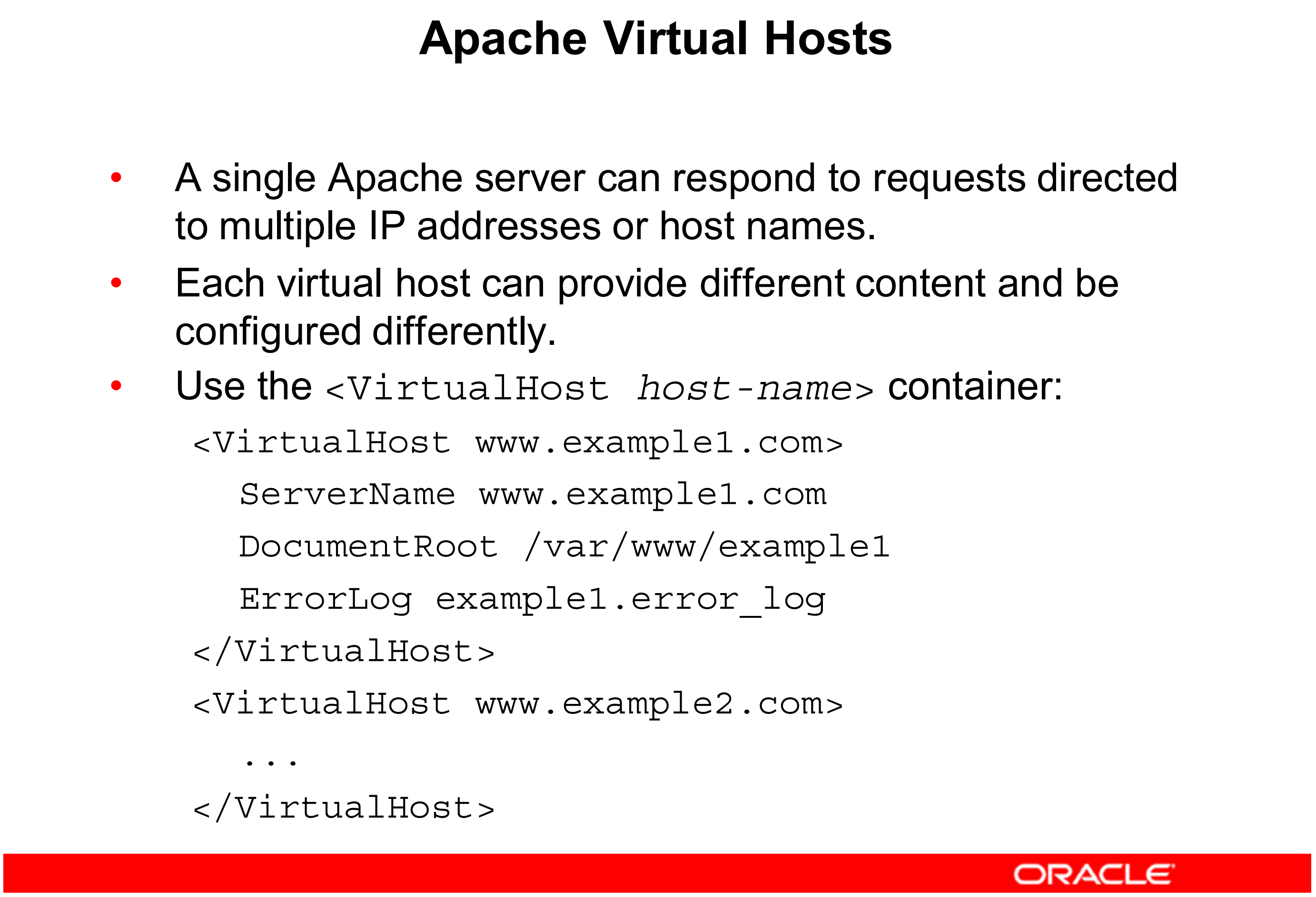
This container is the opposite of the `Limit` container in that it limits access control directives to all except specified methods.

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 different 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:

```
NameVirtualHost *:80
<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

Email Program Classifications

- Mail User Agent (MUA):
  - An email client application to create and read email messages
  - Some MUAs are capable of sending outbound messages to MTA.
  - Some MUAs are capable of retrieving messages from remote servers by using POP or IMAP.
- Mail Transfer Agent (MTA):
  - An email server that transports email messages by using SMTP
  - Examples: Sendmail, Postfix, Fetchmail
- Mail Delivery Agent (MDA):
  - Invoked by MTA
  - Puts incoming email in the recipient's mailbox file
  - Examples: Procmail or `mail`

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 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 Transport 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 used to transfer 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 originating 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:

```
# service dovecot start
```

To ensure that the service starts when booting to multi-user mode:

```
# chkconfig dovecot on
```

By default, `dovecot` runs IMAP and POP together with their secure versions using Secure Socket 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.conf`. Refer to `/usr/share/doc/doc/dovecot*` for more information.

Postfix SMTP Server

- This is the default MTA with Oracle Linux.
- The main configuration files are in the `/etc/postfix` directory:
  - `access`: Specifies which hosts can connect to Postfix
  - `main.cf`: The global Postfix configuration file
  - `master.cf`: Specifies how Postfix processes interact
  - `transport`: Maps email addresses to relay hosts
- Restart the service after making any configuration changes:

```
# service postfix restart
```

- Refer to www.postfix.org for complete documentation.

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:

```
# service postfix restart
```

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:

```
# service sendmail restart
# 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:

```
# service sendmail restart
```

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 re-generate 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:

```
# service sendmail restart
```

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 originates on the local system. It does not forward email originating 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.

```
# service sendmail restart
```

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 4: Overview

The practices for this lesson cover the following:

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

5

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
- Use the Kickstart Configurator
- Start a Kickstart installation
- Boot into Rescue mode to correct boot problems

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 need to 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 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 you would respond to 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 pre-installation and post-installation 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 disk partition. In this example, the `root` password is encrypted:

- `lang en_US.UTF-8`
- `keyboard us`
- `rootpw --iscrypted $6$...`
- `timezone --utc America/Denver`
- `part / --fstype=ext4 --size=2000`

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 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 pre-installation section contains commands to run on the system immediately after the Kickstart file has been parsed but before the installation begins. The pre-installation section begins with the `%pre` command and ends with the `%end` command. You can access the network in the `%pre` section only by using 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.

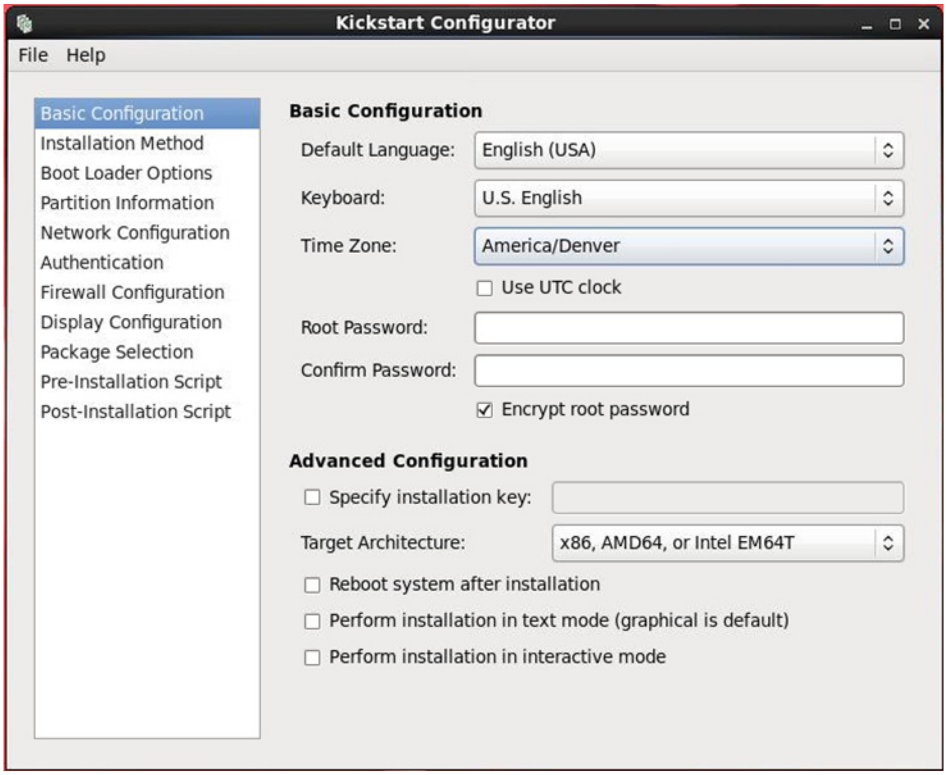
```
%pre
%include http://192.0.2.1/scripts/config-partitions
%end
```

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

Both the pre-installation section and post-installation section must be placed at the end of the Kickstart file.

Refer to the Oracle Linux documentation for Kickstart file options and additional examples: http://docs.oracle.com/cd/E37670_01/

Kickstart Configurator



Basic Configuration

Default Language: English (USA)

Keyboard: U.S. English

Time Zone: America/Denver

□ Use UTC clock

Root Password:

Confirm Password:

☑ Encrypt root password

Advanced Configuration

□ Specify installation key:

Target Architecture: x86, AMD64, or Intel EM64T

□ Reboot system after installation

□ Perform installation in text mode (graphical is default)

□ Perform installation in interactive mode

A Kickstart Configurator is available to create or modify a Kickstart file using a graphical user interface (GUI). The Kickstart Configurator allows you to specify system information, packages to install, and pre- and post-installation scripts without needing to remember the correct syntax of the Kickstart file.

To use the Kickstart Configurator, install the package as follows:

```
    # yum install system-config-kickstart
```

To launch the Kickstart Configurator, use the following command:

```
    # system-config-kickstart
```

Select options from the list on the left side of the GUI. This slide displays prompts associated with the Basic Configuration option selected. In this screen you can specify the Default Language, Keyboard, Time Zone, Root Password, and Advanced Configuration options. Select Installation Method from the list of options to specify either a new installation or an upgrade. You can also specify to install or upgrade from DVD, NFS, FTP, HTTP, or from a hard drive. Continue selecting options from the left side to build the Kickstart file.

You can start with an existing Kickstart file by selecting **File > Open** from the menu bar. You can also preview a Kickstart file as you are creating it by selecting **File > Preview**. Select **File > Save** after you have provided all necessary information for the Kickstart file.

Beginning a Kickstart Installation

- Boot from boot media or Oracle Linux install media.
- Press Esc at the boot menu to get to the boot prompt.
- Enter a special `ks` command at the boot prompt.
- If the Kickstart file is located on a boot CD, enter:

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

- If the Kickstart file is on an HTTP server, enter:

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

- Provide network interface information if the network server is not running DHCP. 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
```

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

- Boot into rescue mode to correct boot problems.
- Rescue mode boots from installation media.
- File systems are mounted under `/mnt/sysimage`.
- Use `chroot` to change the root partition of the rescue mode environment.

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

- You can then access files and use Linux utilities to fix the boot problem.

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, re-install 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.

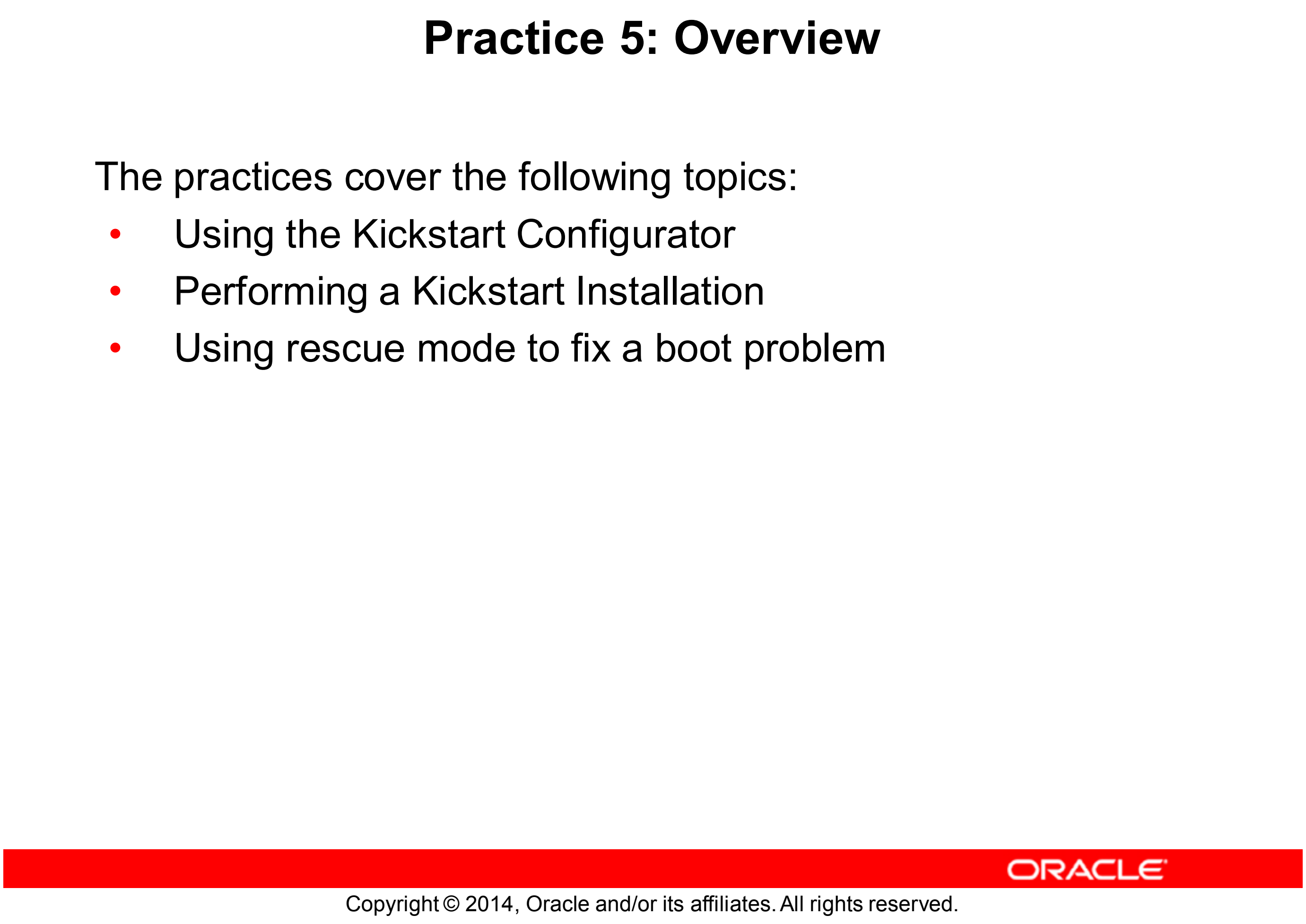
a. True

b. False

Summary

In this lesson, you should have learned how to:

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



Practice 5: Overview

The practices cover the following topics:

- Using the Kickstart Configurator
- Performing a Kickstart Installation
- Using rescue mode to fix a boot problem

6

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`

ORACLE

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 the `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:

```
# service smb start
# service nmb start
```

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:

```
# service winbind start
```

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 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 `man smb.conf` 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:

```
# service smb start
```

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

```
# service smb restart
# service smb reload
```

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

Use the `chkconfig` command to automatically start the service at boot time. Example:

```
# chkconfig smb on
```

Samba Server

- Three different ways to configure a Samba server:
  - Stand-alone server
  - Domain member server
  - Domain controller
- Difference between workgroup and domain:
  - A Windows workgroup is a smaller, peer-to-peer network with no centralized management.
  - A Windows domain is a larger network of computers that share security and access control. Centralized management is provided by a domain controller.

There are three different ways 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.
- Stand-alone server:
  - Can be a workgroup server or a member of a workgroup.
- Domain member server:
  - Logs in to a domain controller and is subject to the domain’s security rules.
- Domain controller:
  - Can only be a domain controller in a Windows NT domain, not 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 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 non-secure 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 can be a member of an Active Directory but it cannot operate as an Active Directory 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."

Accessing Linux Shares from Windows

- Browse using the `\\`*`servername`*`\`*`sharename`* syntax.
  - *`servername`* is the Linux Samba server.
- Provide a Windows username and a Samba password.
- The Windows username must map to the Linux username.
  - `/etc/samba/smbuser` maps Linux > Windows usernames.
  - `oracle = wuser`
- Use the `smbpasswd` command to add a Samba password.

```
# smbpasswd -a oracle
```

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 -t 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` command with the following format to mount Samba shares:

```
mount -t 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 findsmb
/usr/bin/findsmb
# rpm -qf /usr/bin/findsmb
samba-client-<version>
```

The Samba command-line utilities include the following:

- **`findsmb`:** Lists information about systems that respond to SMB name queries on a subnet
- **`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

- **`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.
- **`nmblookup`:** Is used to query NetBIOS names and map them to IP addresses
- **`pdbedit`:** Manages accounts located in the SAM database (the database of Samba users)
- **`rpcclient`**: Is a tool for executing client-side Microsoft RPCs functions
- **`smbcacls`:** Modifies Windows ACLs on files and directories shared by a Samba server or a Windows server
- **`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 6: 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

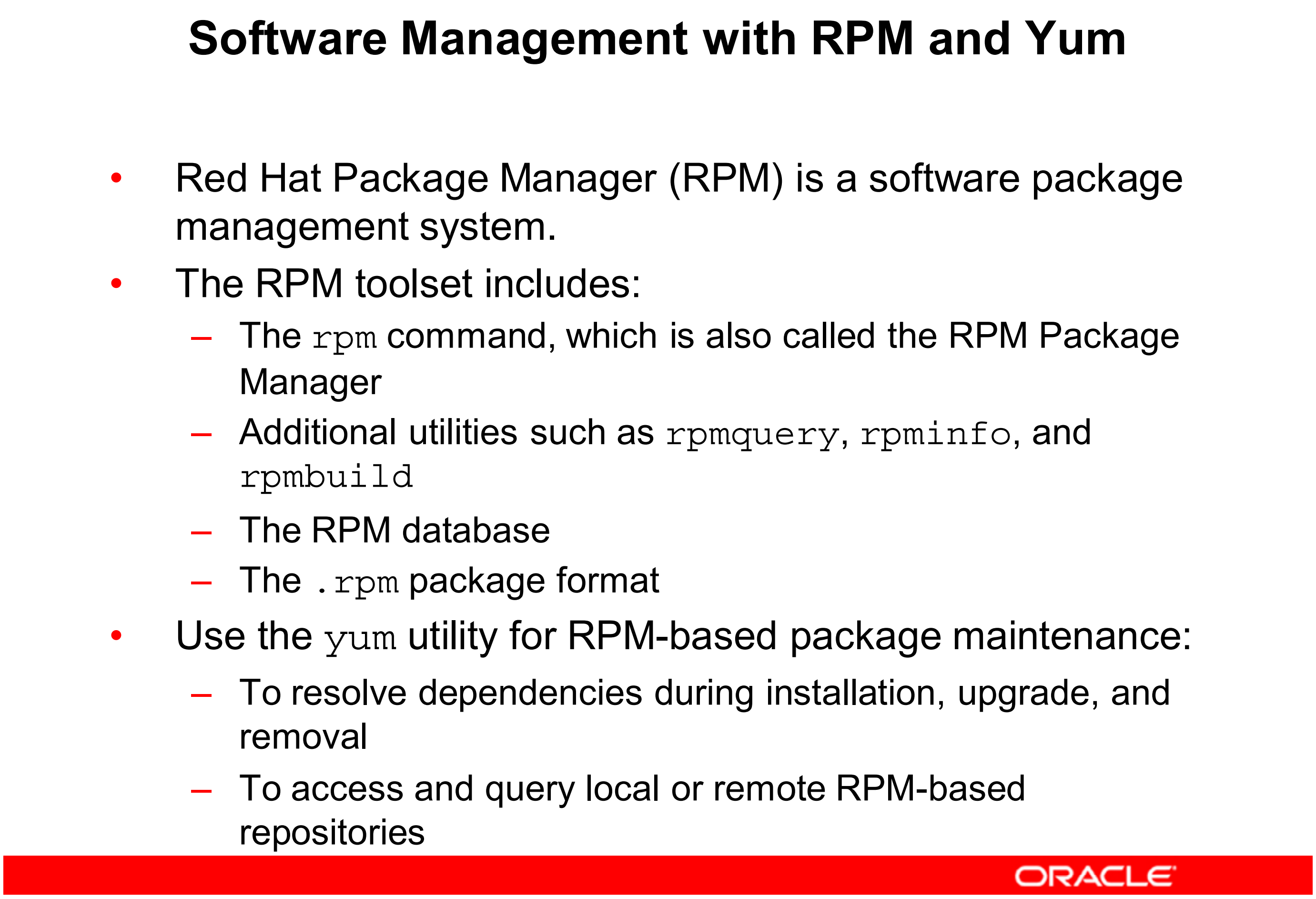
7

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



Software Management with RPM and Yum

- Red Hat Package Manager (RPM) is a software package management system.
- The RPM toolset includes:
  - The `rpm` command, which is also called the RPM Package Manager
  - Additional utilities such as `rpmquery`, `rpminfo`, and `rpmbuild`
  - The RPM database
  - The `.rpm` package format
- Use the `yum` utility for RPM-based package maintenance:
  - To resolve dependencies during installation, upgrade, and removal
  - To access and query local or remote RPM-based repositories

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-4.1.2-15.el6_4.x86_64
        /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
...
```

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 using the RPM format and they 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: refresh-packagekit, security
Setting up Install Process
Package bash-4.1.2-15.el6_4.x86_64 already installed ...
Nothing to do
```

- An RPM source package file (for example `bash-4.1.2-15.el6_4.src.rpm`) contains the source code and all necessary files to recreate the binary package file.

```
# yumdownloader --source bash
Loaded plugins: refresh-packagekit
bash-4.1.2-15.el6_4.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: downloadonly, refresh-packagekit, security
Installed Packages
Name        : bash
Arch        : x86_64
Version     : 4.1.2
Release     : 15.el6_4
....
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.
- `%defattr` directive sets default permissions and owner and group attributes for files in the `%files` section.

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

Group:
       Applications/Communications
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 $RPM_BUILD_ROOT
#make install DESTDIR=$RPM_BUILD_ROOT
install -d $RPM_BUILD_ROOT/usr/local/bin
install hello
       $RPM_BUILD_ROOT/usr/local/bin/hello

%clean
rm -rf $RPM_BUILD_ROOT


%files
%defattr(-,root,root,-)
/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.el6.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 `downloadonly` plug-in (to download a package without installing it)
- 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
```

The following command displays the last 20 transactions in Yum history:

```
# yum history list
Loaded plugins: downloadonly, refresh-packagekit, security
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 `yum-presto` package:

```
# yum history info 6
Loaded plugins: downloadonly, refresh-packagekit, security
Transaction ID : 6
Begin time     : ...
...
Return-Code    : Success
Command Line   : install yum-presto
Transaction performed with:
...
```

In this example, transaction ID `6` installed the `yum-presto` package. Transaction ID `7` de-installed the `yum-presto` package. The following command performs transaction ID `6` again, which installs `yum-presto`:

```
# yum history redo 6
Loaded plugins: downloadonly, refresh-packagekit, security
...
Repeating transaction 6...
...
Installing:
 yum-presto    ...
...
```

Extending Yum Functionality with Plug-Ins

- Yum uses plug-ins to extend its functionality.
- Yum plug-ins are installed as packages.
- Yum plug-ins reside in `/usr/lib/yum-plugins`.
- Each plug-in has an associated configuration file in `/etc/yum/pluginconf.d`.
- To disable a plug-in for a single command:

```
# yum update --disableplugin=presto
```

Yum uses plug-ins to extend its functionality.

Each plug-in has a configuration file located in `/etc/yum/pluginconf.d`. For example, the configuration file for the `security` plug-in contains:

```
# cat /etc/yum/pluginconf.d/security.conf
[main]
enabled=1
```

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 de-installing 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 /usr/lib/yum-plugins
downloadonly.py      refresh-packagekit.py      security.py
downloadonly.pyc     refresh-packagekit.pyc     security.pyc
downloadonly.pyo     refresh-packagekit.pyo     security.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

- `security`
  - To manage errata (security, bug fixes, and enhancements)
- `downloadonly`
  - To download packages without installing them
- `presto`
  - To download the delta of RPM packages instead of the full packages
- `fs-snapshot`
  - To create a snapshot of a file system before updating your system.

`security`

The package name is `yum-plugin-security`.

This plug-in allows you to use the `yum` command to obtain a list of the errata available for your system, by using filters that limit the output to an errata type. You learn about the `security` plug-in in the next slide.

`fs-snapshot`

The package name is `yum-plugin-fs-snapshot`.

This plug-in allows you to create a snapshot of a file system before updating your system. If you decide that you do not want to keep the changes, you can roll back to the snapshot. The `root` file system must be created on a logical volume or on a Btrfs volume.

`presto`

The package name is `yum-presto`.

With the `presto` plug-in, you can download the delta of RPM packages. A delta package contains only the changes between the installed package and the package residing in the repository. Downloading only delta RPM packages results in faster downloads. However, the packages must be rebuilt after they are downloaded and before they are installed, resulting in additional processing time and resources on the target server.

downloadonly

The package name is `yum-plugin-downloadonly`.

With this plug-in installed, you can download a package and its dependent packages without installing the package(s). Example:

```
# yum install vsftpd --downloadonly
```

The packages are downloaded to the Yum cache, but you can specify a different directory with the `--downloaddir` option. A package is not downloaded if it is already installed. You can use the `yumdownloader` command to download a package that is installed.

```
# yumdownloader vsftpd
```

Using the Yum Security Plug-In

- To install the Yum `security` plug-in:

```
# yum install yum-plugin-security
```

- To list all of the available errata:

```
# yum updateinfo list
```

- To filter errata information:
  - By security priority (important, moderate, low)

```
# yum updateinfo list --sec-severity=Important
```

  - By erratum

```
# yum updateinfo --advisory ELSA-2014-0097
```

  - By CVE

```
# yum updateinfo info --cve CVE-2013-5896
```

The Yum `security` plug-in is installed automatically in newer Oracle Linux releases. If it is not installed on your system, use the following command to install it:

```
# yum install yum-plugin-security
```

The Yum `security` plug-in allows you to view information about errata, but also to update your Linux system by using options that act like filters to select only certain updates from the errata. You can find the list of available options by reading the `man` page:

```
# man yum-security
```

Obtaining Errata Information

To list all of the available errata:

```
# yum updateinfo list
...
ELSA-2014-0043 Moderate/Sec.  bind-libs-32:9.8.2-0.23.rc1.el...
ELSA-2014-0043 Moderate/Sec.  bind-utils-32:9.8.2-0.23.rc1.e...
ELSA-2013-1866 Moderate/Sec.  ca-certificates-2013.1.95-65.1...
ELSA-2013-1866 bugfix         dmidecode-1:2.11-2.el6_1.x86_64
...
```

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: Important, moderate or low.

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
```

Important Resources for Errata Information

- Refer to the following for errata listings:
  - https://linux.oracle.com/errata
- Refer to the following for CVE listings:
  - https://linux.oracle.com/cve
- Finding important errata and CVE information on ULN:
  - https://blogs.oracle.com/linux/entry/finding_information_on_important_errata
- Updates to errata on ULN and public-yum.oracle.com:
  - https://blogs.oracle.com/linux/entry/updates_to_errata_on_uln

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.

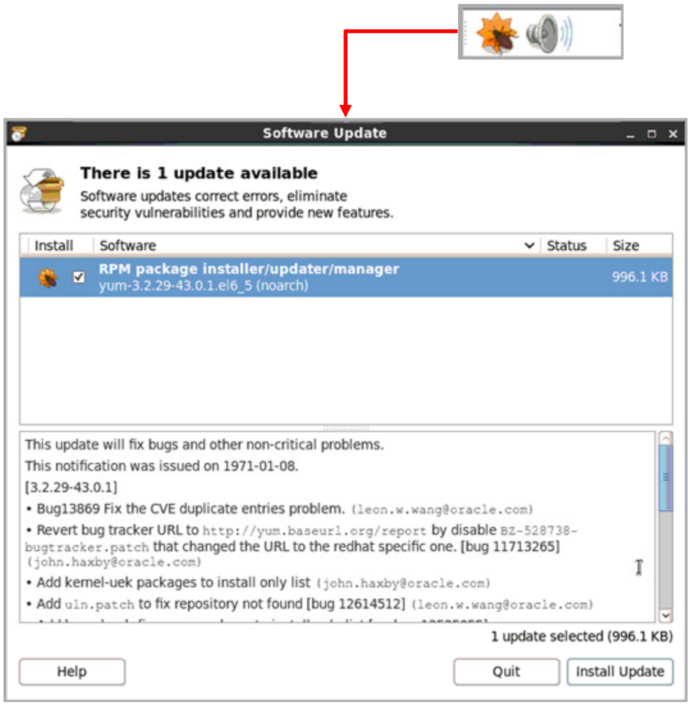
The PackageKit program also depends on a Yum plug-in called `refresh-packagekit`, which is installed from the `PackageKit-yum-plugin` package. This plug-in tells PackageKit to check for updates when `yum` exits. This plug-in is listed every time you invoke the `yum` command. The following example shows the `refresh-packagekit` as a loaded plug-in:

```
# yum info bash
Loaded plugins: downloadonly, refresh-packagekit, security
Installed Packages
Name         : bash
Arch         : x86_64
Version      : 4.1.2
...
```

Using PackageKit Software Update

Launch Software Update through one of the following methods:

- Click the starburst icon in the notification area.
- Select Software Updates from the Administration menu.
- Run `gpk-update-viewer` from the command line.



There is 1 update available

Software updates correct errors, eliminate security vulnerabilities and provide new features.

Install	Software	Status	Size
☑	**RPM package installer/updater/manager** yum-3.2.29-43.0.1.el6_5 (noarch)		996.1 KB

This update will fix bugs and other non-critical problems.

This notification was issued on 1971-01-08.

[3.2.29-43.0.1]

- Bug13869 Fix the CVE duplicate entries problem. (leon.w.wang@oracle.com)
- Revert bug tracker URL to http://yum.baseurl.org/report by disable BZ-528738-bugtracker.patch that changed the URL to the redhat specific one. [bug 11713265] (john.haxby@oracle.com)
- Add kernel-uek packages to install only list (john.haxby@oracle.com)
- Add uln.patch to fix repository not found [bug 12614512] (leon.w.wang@oracle.com)

1 update selected (996.1 KB)

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.

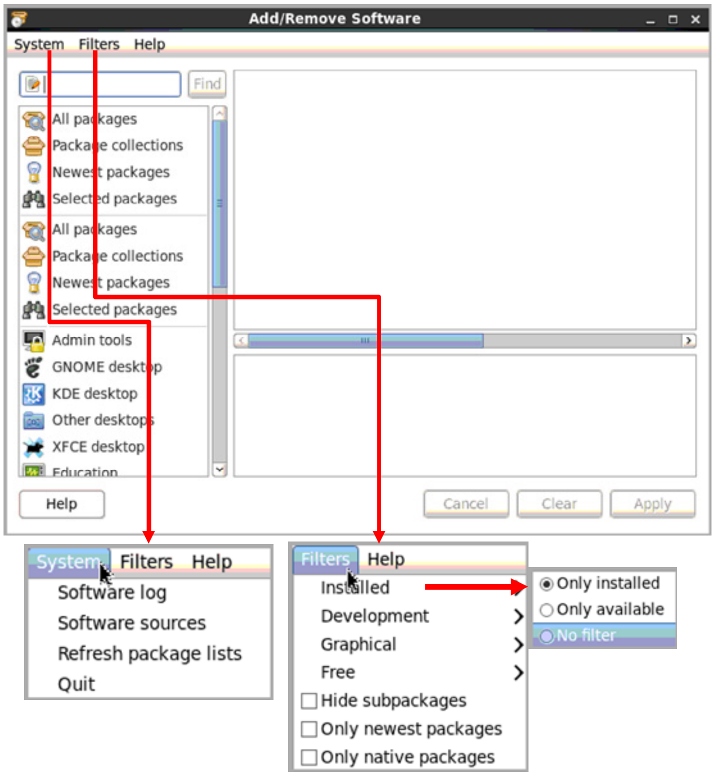
You can use the Software Update and the Add/Remove Software programs as a non-privileged user. When requesting to add/update/remove packages, you are prompted for the `root` password.

You can customize the settings for Software Update by selecting Software Update Preferences from the GNOME Preferences drop-down list. From the Update Settings window, you can change the interval at which PackageKit checks for updates and whether to install updates automatically.

Using PackageKit Add/Remove Software

Launch Add/Remove Software by either of the following methods:

- Select Add/Remove Software from the Administration menu.
- Run `gpk-application` from the command line.



From the Add/Remove Software window, you can find and install packages as well as remove packages. You can select several packages for the chosen operation and the selected packages are acted on when you click the Apply button.

To limit the list of packages displayed for the various categories (All packages, Newest packages, or packages from a Package collection), you set filters by selecting Filters from the menu bar. The filters are shown in the small window at the bottom right in the slide. For example, you can hide the packages that are listed only as dependencies of other packages by selecting the “Hide subpackages” check box, or you can display “Only installed” or “Only available” packages from the Installed filter selections.

Because Yum is the package management used by PackageKit, PackageKit looks for packages in repositories, either local or remote. You can select which repositories are enabled by selecting “Software sources” from the System entry in the menu bar. From the Software Sources window, you select repositories to enable them or deselect repositories to disable them. This action is similar to the `yum-config-manager --enable` *`<repository name>`* or `yum-config-manager --disable` *`<repository name>`* commands. You can enable and disable repositories from the Add/Remove Software program, but you cannot add a new repository or delete an existing one.

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-repo` → Software Sources
  - `gpk-update-icon` → Alert for available updates
- 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-application` program launches the Add/Remove Software graphical program as discussed in the previous slide. Similarly, the `gpk-update-viewer` launches the Software Update graphical program. The `gpk-log` and `gpk-repo` commands start graphical programs that are usually launched as selections from the Add/Remove Software graphical program. The `gpk-prefs` command is equivalent to selecting System > Preferences > Software Updates from the GNOME desktop.

The `gpk-update-icon` program displays the starburst icon in the notification area on the GNOME desktop when software updates are available. The program is usually started when the system boots:

```
# ps -ef | grep gpk
oracle   ...   gpk-update-icon
root     ...   grep gpk
```

The `pkcon` is the CLI equivalent to the Software Update and Add/Remove Software graphical programs. The following example installs an update to the `yum` command that was flagged as an update in the Software Update program.

```
# pkcon install yum
Simulating install            [=========================]
Starting                      [=========================]
Running                       [=========================]
Resolving dependencies        [=========================]
Proceed with changes? [N/y]   [=========================]
Installing                    [=========================]
Waiting for authentication    [=========================]
Starting                      [=========================]
Running                       [                         ] (0%)
                              [=========================]
Resolving dependencies        [=========================]
Downloading packages          [=========================]
Testing changes               [=========================]
Installing updates            [=========================]
Cleaning up packages          [=========================]
Scanning applications         [=========================]
```

In this example, the `yum` command is updated to `yum-3.2.29-43.0.1.el6_5`. You can use the `gpk-log` command to view the log entry for this update.

Disabling PackageKit Software Updates

- Many Linux administrators choose to manage software by using Yum commands directly.
- To disable the Software Updates program from the desktop:
  - From System > Preferences > Software Updates, change the frequency at which the Software Updates program refreshes its list of updates to “Never.”
  - From System > Preferences > Startup Applications Preferences, deselect the PackageKit Update Applet.
  - Log off from your desktop session.

Most Linux administrators prefer to manage software installation and updates by using the Yum commands directly, rather than use the graphical interfaces offered by the PackageKit programs.

Even if you decide to maintain the software on your system without the PackageKit programs, you might get this message when executing `yum` commands:

```
Another app is currently holding the yum lock; waiting for it to
exit...
The other application is: PackageKit
```

The solution is to disable the Software Updates applet that checks for software updates for your environment. You can disable the `refresh-packagekit` Yum plug-in and even remove the PackageKit programs, but this solution affects all users.

To disable the `refresh-packagekit` Yum plug-in, set `enabled=0` in the configuration file:

```
# cat /etc/yum/pluginconf.d/refresh-packagekit.conf
[main]
enabled=0
```

Quiz

Which statements are true regarding 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`.

ORACLE

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 7: Overview

In the practices for this lesson, you:

- Learn to manage Yum plug-ins
- Create a binary RPM package
- Manage software updates with PackageKit's Software Update program
- Work with Yum history and Yum cache

ORACLE

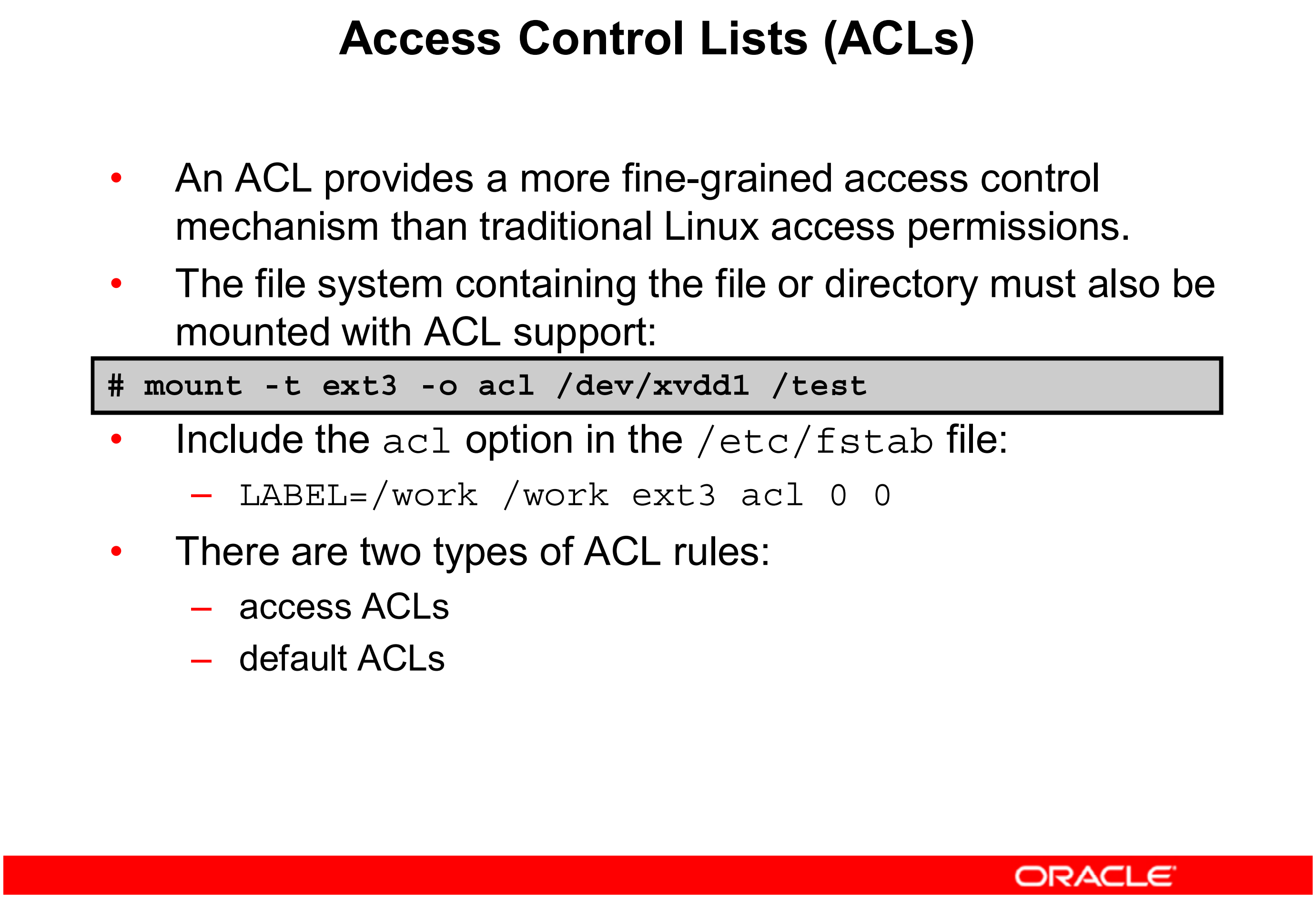
8

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



Access Control Lists (ACLs)

- An ACL provides a more fine-grained access control mechanism than traditional Linux access permissions.
- The file system containing the file or directory must also be mounted with ACL support:

```
# mount -t ext3 -o acl /dev/xvdd1 /test
```

- Include the `acl` option in the `/etc/fstab` file:
  - `LABEL=/work /work ext3 acl 0 0`
- There are two types of ACL rules:
  - access ACLs
  - default ACLs

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 partition
```

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`**: Specifies access information for a single file or directory
- **`default ACLs`**: Pertains 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. The following example adds a rule to the ACL for the test file that gives user oracle 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 –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`
- SELinux in Enforcing mode results in “Permission denied” messages.

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 `quotacheck`. The `quotacheck` 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
```

If you get "permission denied" messages, disable `selinux`:

```
# echo 0 > /selinux/enforce
```

Edit the `/etc/selinux/config` file and change "`SELINUX=enforcing`" to "`SELINUX=disabled`". SELinux is discussed in "Lesson 16: SELinux."

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.

`cryptsetup` Utility

- Use the `cryptsetup` command to create and activate encrypted volumes, and to manage authentication.
- The `cryptsetup` command includes the Linux Unified Key Setup (LUKS) extension, a disk encryption standard.
- The basic syntax of the `cryptsetup` command is:
  - `cryptsetup [options] [action] [action args]`
- To initialize a volume and set an initial key:

```
# cryptsetup luksFormat /dev/xvdd1
```

- To open the partition and create the device mapping:

```
# cryptsetup luksOpen /dev/xvdd1 cryptfs
```

- The device mapping file is:
  - `/dev/mapper/cryptfs`

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: <enter passphrase>
Verify passphrase: <enter passphrase again>
```

To open the partition and create the device mapping, enter:

```
# cryptsetup luksOpen /dev/xvdd1 cryptfs
Enter passphrase for /dev/xvdd1: <enter 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-cbs-essiv:sha256
  keysize: 256 bits
  device:  /dev/xvdd1
  offset:  4096 sectors
  size:    6309386 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
```

Making an Encrypted Device Usable

1. Create a file system on the encrypted device.
2. Create a mount point.
3. Attach the encrypted device to the directory hierarchy.
4. Update the `/etc/crypttab` configuration file:
   – `cryptfs   /dev/xvdd1   none   luks`
5. Add the file system to `/etc/fstab`.
6. Enter the passphrase to mount the encrypted file system during boot.

ORACLE

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 `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
```

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-2.6.32-220.el6.x86_64
config-2.6.32-300.3.1.el6uek.x86_64
efi
grub
initramfs-2.6.32-220.el6.x86_64.img
initramfs-2.6.32-300.3.1.el6uek.x86_64.img
…
```

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-2.6.32-220.el6.x86_64
config-2.6.32-300.3.1.el6uek.x86_64
efi
grub
initramfs-2.6.32-220.el6.x86_64.img
initramfs-2.6.32-300.3.1.el6uek.x86_64.img
…
```

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
```

The following procedure provides an example of using the kpartx command when working with Oracle VM templates. This example downloads and extracts the template for Oracle Linux 6 Update 5:

1. Select “Oracle VM 3 Templates (OVF) for Oracle Linux 6 Media Pack for x86_64 (64 bit)” from the Oracle Software Delivery Cloud (https://edelivery.oracle.com/linux).
2. Download “Oracle Linux 6 Update 5 template (OVF) – Paravirtualized x86_64 (64 bit)” from the list of download options. The downloaded file name is V42906.01.zip.
3. Use the unzip command to unzip the downloaded ZIP file. The resulting file name is OVM_OL6U5_x86_64_PVM.ova.
4. Use the tar command to extract the ova file. Three files are extracted, one of which is named System.img.
5. The System.img file is compressed but you must rename it to be de-compressed. Use the mv command to rename System.img to System.img.gz.
6. Use the gunzip command to uncompress the System.img.gz file. The resulting file name is System.img.
7. Use the kpartx -l command to list the partitions found on the System.img file:

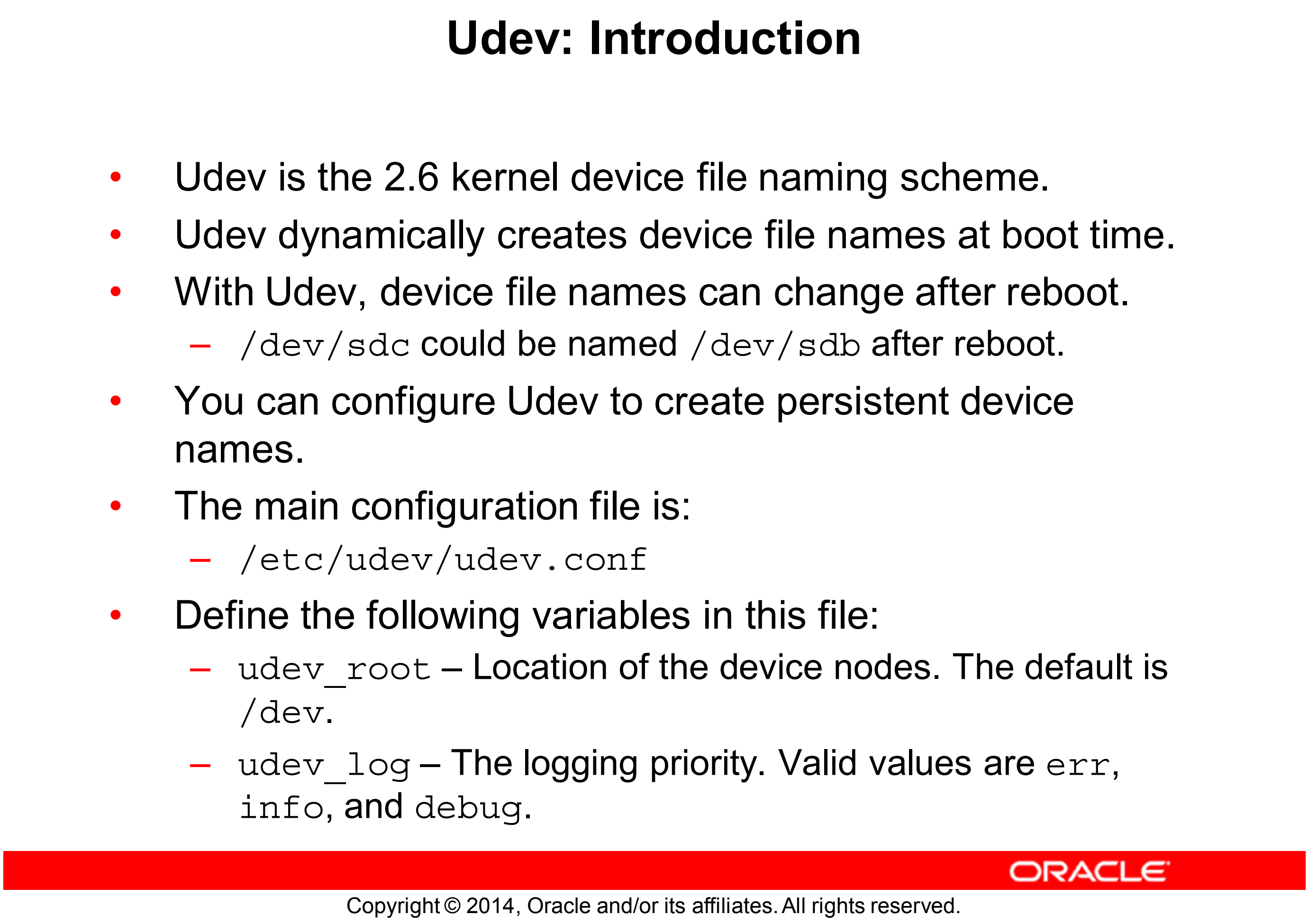
   ```
   # kpartx -l system.img
   loop0p1 : 0 1028096 /dev/loop0 2048
   loop0p2 : 0 19941376 /dev/loop0 210944
   loop0p3 : 0 4194304 /dev/loop0 20971520
   ```

8. Use the kpartx -a command to add device mappings for the detected partitions on the System.img file. The device mappings are created in the /dev/mapper directory:

   ```
   # kpartx -a System.img
   # ls /dev/mapper
   control  loop0p1  loop0p2  loop0p3
   ```

9. The root file system is found on /dev/mapper/loop0p2. Use the mount command to mount this device on /mnt. You can then use the ls command to display the contents of the root file system on the Oracle VM template.

   ```
   # mount /dev/mapper/loop0p2 /mnt
   # ls /mnt
   bin  dev home lib64      media opt  root selinux sys u01 var
   boot etc lib  lost+found mnt   proc sbin srv     tmp usr
   ```



Udev: Introduction

- Udev is the 2.6 kernel device file naming scheme.
- Udev dynamically creates device file names at boot time.
- 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.
- The main configuration file is:
  - `/etc/udev/udev.conf`
- Define the following variables in this file:
  - `udev_root` – Location of the device nodes. The default is `/dev`.
  - `udev_log` – The logging priority. Valid values are `err`, `info`, and `debug`.

Udev is the device file naming mechanism in the 2.6 Linux kernel. Udev dynamically creates or removes device node files at boot time, by default in the `/dev` directory, for all types of devices. Udev is executed if a kernel device is added or removed from the system. On device creation, Udev reads the `sysfs` file system for the given device to collect device attributes such as label, serial number, or bus device number.

Note that 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 originally 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.

The main configuration file is `/etc/udev/udev.conf`. Define the following variables in this file:

- `udev_root` – Location of the device nodes in the file system. The default is `/dev`.
- `udev_log` – The logging priority. Valid values are `err`, `info`, and `debug`.

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
  - `/dev/.udev/rules.d/` – The temporary rules directory
- Custom rule files override default rules files of the same name.
- Rules files are sorted and processed in lexical order.
- Sample rules file names:
  - `10-console.rules`
  - `50-udev.rules`

ORACLE

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.
- `/dev/.udev/rules.d/` – The temporary rules directory

Rules files are required 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
10-console.rules  10-dm.rules  11-dm-lvm.rules  13-dm-disk.rules
# ls /etc/udev/rules.d
40-hplip.rules  56-hpmud_support.rules  60-pcmia.rules
```

Sample Udev Rules

```
SUBSYSTEM==“block”, SYMLINK{unique}+=“block/%M:%m”
SUBSYSTEM!=“block”, SYMLINK{unique}+=“char/%M:%m”
KERNEL==“tty[0-9]”, GROUP=“tty”, MODE=“0620”
# mem
KERNEL==“mem|kmem|port|nvram”, GROUP=“kmem”, MODE=“0640”
# block
SYBSYSTEM==“block”, GROUP=“disk”
# network
KERNEL==“tun”, MODE=“0666”
# cpu
KERNEL==“cpu[0-9]*”, MODE=“0444”
# do not delete static device nodes
ACTION==“remove”, NAME==“?*”,
   TEST==“/lib/udev/devices/$name”,
   OPTIONS+=“ignore_remove”
```

This slide contains selected entries from the `/lib/udev/rules.d/50-udev-default.rules` file. This rules file contains over 100 entries. The selected entries assist in describing the syntax of the rules files.

Comments begin with a `#` sign. Each non-commented 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.
- `KERNEL`: Match the name of the event device.
- `SYMLINK`: Match the name of the symlink targeting the node. A `SYMLINK` key can be set in the preceding rules. There can be multiple symlinks but only one needs to match.
- `SUBSYSTEM`: Match the subsystem of the event device.
- `NAME`: Match the name of the node or network interface. You can set a `NAME` key in the preceding rules.
- `TEST{`*`octal mode mask`*`}`: Test the existence of a file. You can specify *`octal mode mask`*.

Other match keys include `DEVPATH`, `DRIVER`, `ATTR{`*`filename`*`}`, `KERNELS`, `SUBSYSTEMS`, `DRIVERS`, `ATTRS{`*`filename`*`}`, `ENV{`*`key`*`}`, `PROGRAM`, and `RESULT`.

Assignment Keys

The following keys can have values assigned to them:

- `NAME` – The name of the node to be created, or the name the network interface is renamed to
- `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`*`}`, `RUN`, `LABEL`, `GOTO`, `IMPORT{`*`type`*`}`, and `WAIT_FOR`.

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 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/sda`:

```
# udevadm info --query=all --name=/dev/sda
```

- To print all `sysfs` properties of `/dev/sda`:

```
# udevadm info --attribute-walk --name=/dev/sda
```

- 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
  test     simulation run
```

You can also obtain usage for each of the `udevadm` commands. For example, to get help on using the `info` command:

```
# udevadm info --help
Usage: udevadm info OPTIONS ...
```

Some examples follow. To query the Udev database for the device path of `/dev/sda`:

```
# udevadm info --query=path --name=/dev/sda
/devices/pci0000:00/0000:00:0d.0/host2/target2:0:0/2:0:0:0/block
/sda
```

To query the Udev database for the symlinks for `/dev/sda`:

```
# udevadm info --query=symlink --name=/dev/sda
block/8:0
disk/by-id/ata-VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-id/scsi-SATA_VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-path/pci-0000:00:0d.0-scsi-0:0:0:0
```

To query the Udev database for all device information for `/dev/sda`:

```
# udevadm info --query=all --name=/dev/sda
P:/devices/pci0000:00/0000:00:0d.0/host2/target2:0:0/2:0:0:0/blo
ck/sda
N:sda
S:block/8:0
...
```

Enter the following to print all `sysfs` properties of `/dev/sda`. 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/sda
looking at device
‘/devices/pci0000:00/0000:00:0d.0/host2/target2:0:0/2:0:0:0/bloc
k/sda’:
KERNEL==“sda”
SUBSYSTEM==“block”
DRIVER==“”
ATTR{range}==“16”
...
looking at parent device
‘/devices/pci0000:00/0000:00:0d.0/host2/target2:0:0/2:0:0:0’:
KERNELS==“2:0:0:0”
SUBSYSTEMS==“scsi”
DRIVER==“sd”
ATTR{device_blocked}==“0”
...
looking at parent device ‘/devices/pci0000:00’:
KERNELS==“pci0000:00”
...
```

Changing /dev/sdb to /dev/my_disk: Example

- Create a rules file:

```
# vi /etc/udev/rules.d/10-local.rules
KERNEL=="sdb", SUBSYSTEM==“block", NAME=“my_disk”
```

- Run start_udev to process the rules files:

```
# start_udev
Starting udev:                     [  OK  ]
```

- The file name has changed:

```
# ls /dev/sd* /dev/my*
/dev/my_disk /dev/sda /dev/sda1 /dev/sda2
```

ORACLE

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 renames /dev/sdb to /dev/my_disk:

```
KERNEL=="sdb", SUBSYSTEM==“block", NAME=“my_disk”
```

Before restarting Udev, the current /dev/sd* output is:

```
# ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sdb  /dev/sdc  /dev/sdd
```

Run start_udev to process the rules files:

```
# start_udev
Starting udev:                       [  OK  ]
```

The file name has changed:

```
# ls /dev/sd* /dev/my*
/dev/my_disk  /dev/sda  /dev/sda1  /dev/sda2  /dev/sdc  /dev/sdd
```

Remove the 10-local.rules file and run start_udev to return to default names.

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 the device file-naming mechanism introduced in the 2.4 Linux kernel.

b. You can define Udev rules to create persistent device names.

c. Run the `udevadm` 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 8: 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

9

OCFS2 and Oracle Clusterware

Objectives

After completing this lesson, you should be able to:

- Describe the purpose and features of Oracle Cluster File System 2 (OCFS2)
- Prepare for an OCFS2 configuration
- Install the OCFS2 software packages
- Configure kernel settings for OCFS2
- Configure the cluster layout
- Describe the OCFS2 heartbeat
- Configure and start the O2CB cluster service
- Create and mount an OCFS2 volume
- Use OCFS2 tuning and debugging utilities
- Provide an introduction to Oracle Clusterware

OCFS2: Introduction

- OCFS2 is a shared disk cluster file system, which allows multiple nodes to access the same disk at the same time.
  - You can also use OCFS2 on a non-clustered system.
- OCFS (Release 1) was created specifically for use by Oracle Real Application Clusters (RAC).
- OCFS2 is a general-purpose cluster file system.
  - You can store any files on OCFS2.
  - You interact with OCFS2 the same way you do on a regular file system.
  - OCFS2 file systems can be mounted and used across multiple architectures at the same time.
- The latest release of OCFS2 requires the Oracle Linux Unbreakable Enterprise Kernel.

OCFS2 is a general-purpose, POSIX-compliant, and shared disk cluster file system. It allows multiple nodes to read from and write to files on the same disk at the same time. It behaves on all nodes exactly like a local file system, and can be used in a non-clustered environment. Files, directories, and their content are always in sync across all nodes regardless of which node updates the files. For example, if you add a line to a file on node 1, it appears immediately on node 2.

OCFS (Release 1) was released in December 2002 to enable Oracle Real Application Cluster (RAC) users to run the clustered database without having to deal with RAW devices. The file system was designed to store database-related files, such as data files, control files, redo logs, and archive logs. OCFS refers to the file system in the 2.4 Linux Kernel.

OCFS2 is the next generation of the Oracle Cluster File System. It is designed to be a general-purpose cluster file system. You can store any files on OCFS2 and you interact with OCFS2 the same way you do on a regular file system. OCFS2 is being used in middleware clusters (Oracle E-Business Suite), appliances such as SAP's Business Intelligence Accelerator, and virtualization with Oracle VM Server for x86.

OCFS2 was developed as a native Linux cluster file system at Oracle. It was submitted for inclusion and accepted into the mainline kernel in January 2006. It was included in the 2.6.16 kernel and is the first native cluster file system to be included in the Linux kernel.

Oracle has done a number of releases. OCFS2 Release 1.4 added new features such as sparse files, unwritten extents, inline data, and shared writeable mmap. OCFS2 Release 1.6 added REFLINKs, indexed directories, metadata checksums, extended attributes, quotas, POSIX ACLs, and allocation reservations. OCFS2 is now available to customers using the Unbreakable Enterprise Kernel.

Both on-disk and network protocol compatibility are maintained across all releases of the file system. The on-disk format changes are managed by a set of feature flags that you can turn on and off. The file system in the kernel detects these features during the mount operation and continues only if it understands all the features. In the event of feature incompatibility, you have the option of either disabling the feature or upgrading the file system to a new release.

The network protocol version is negotiated by the nodes to ensure that all nodes understand the active protocol version. OCFS2 file systems can be mounted and used across multiple architectures at the same time. For example, you can mount a volume on a cluster with x86, ia64, or ppc nodes at the same time.

OCFS2 is currently being used in Oracle VM Server for x86 in both the management domain, to host virtual machine images, and in the guest domain, to allow virtual machines to share a file system. The following example shows mounted OCFS2 file systems. The following example is for Oracle VM 3 where the shared storage for the cluster is NFS based. (For a discussion on using device mapper NFS in Oracle VM, see https://blogs.oracle.com/wim/entry/dm_nfs):

```
# mount |grep ocfs2

ocfs2_dlmfs on /dlm type ocfs2_dlmfs (rw)

/dev/mapper/ovspoolfs on
/poolfsmnt/0004fb000005000031bfbabbc596af47 type ocfs2
(rw,_netdev,heartbeat=global)

/dev/mapper/SATA_ST3500320AS_5QM1EYTX on
/OVS/Repositories/0004fb0000030000a826cf5f901b13de type ocfs2
(rw,heartbeat=none)
```

Since its inclusion in the mainline Linux kernel, other developers and companies have contributed to the development efforts. All new features are included in the mainline Linux kernel tree and all bug fixes are applied to all active kernel trees. The source code of the OCFS2 file system is available under the GNU General Public License (GPL) version 2. Support for the file system is included as part of the Oracle Linux support contract. The OCFS2 development community also provides email support for all users via the ocfs2-users@oss.oracle.com mailing list.

The OCFS2 home page is http://oss.oracle.com/projects/ocfs2/. Documentation, mailing lists, the source code repository, and additional information are available on this page.

OCFS2 Features

- Variable block and cluster sizes for different types of data
- Extent-based allocations for efficient storage of large files
- Sparse files, inline-data, unwritten extents, hole punching, REFLINKS, and allocation reservation
- Extended attributes by attaching `name:value` pairs to file system objects
- POSIX ACLs and SELinux for additional security
- Metadata checksums to detect silent corruption in inodes and directories
- Indexed directories to allow quick lookups of a directory entry in a very large directory
- User and group quotas to limit file system usage
- JBD2 journaling to provide file system consistency

Selected features of the file system:

- **Variable Block and Cluster Sizes** – OCFS2 has two main allocation units, blocks and clusters. OCFS2 supports block sizes ranging from 512 bytes to 4 KB and cluster sizes ranging from 4 KB to 1 MB. Cluster size is always greater than or equal to block size.
- **Extent-based Allocations** – Tracks allocated space in ranges of clusters, making it especially efficient for storing very large files
- **Optimized Allocations** – Supports sparse files, inline data, unwritten extents, hole punching, REFLINKS, and allocation reservation for higher performance and efficient storage. The REFLINK feature allows you to create multiple writeable snapshots of regular files. It involves an on-disk change. Enable the `refcount` file system feature to activate.
- **Extended Attributes** – Supports attaching an unlimited number of `name:value` pairs to file system objects such as regular files, directories, and symbolic links. Use the `setfattr` command to attach extended attributes. This feature involves an on-disk change. Enable the `xattr` file system feature to activate.
- **Advanced Security** – Supports POSIX ACLs and SELinux in addition to the traditional file access permission model. Use the `setfacl` command to assign users specific permissions to an object. Both these security extensions require the `xattr` feature.

- **Metadata Checksums** – Detects silent corruption in inodes and directories. This feature makes the file system compute and validate the checksums of metadata objects, such as inodes and directories, to ensure metadata integrity. It also stores an error correction code that is capable of fixing single bit errors. This feature entails an on-disk change. Enable the `metaecc` file system feature to activate.
- **Indexed Directories** – Allows quick lookups of a directory entry in a very large directory. It also results in faster creates and unlinks, which increases overall performance. This feature entails an on-disk change. Enable the `indexed-dirs` file system feature to activate.
- **Quotas** – Supports user and group quotas by using standard utilities such as `quota`, `setquota`, `quotacheck`, and `quotaon`. This feature involves an on-disk change. Enable the `usrquota` and `grpquota` file system features to activate.
- **JBD2 Journaling** – Supports both ordered and writeback data journaling modes to provide file system consistency in the event of power failure or system crash. Journaling block device (JBD2) allows the file system to grow beyond 16 TB. Journal files in OCFS2 are stored as node local system files. Each node has exclusive access to its journal, and retains a cluster lock on it for the duration of its mount.
- **Discontiguous Block Group** – Dynamically allocates space for inodes when required. This feature allows the allocators to grow in small, variable-sized chunks. It involves an on-disk change. Enable the `discontig-bg` file system feature to activate.
- **Endian and Architecture Neutral** – Supports a cluster of nodes with mixed architectures. The file system allows concurrent mounts on nodes running 32-bit and 64-bit, little-endian (x86, x86_64, ia64) and big-endian (ppc64) architectures.
- **In-kernel Cluster Stack (O2CB)** – Includes an easy-to-configure, in-kernel cluster stack with a Distributed Lock Manager (DLM).
- **Multiple Cluster Stacks** – In addition to its own in-kernel cluster stack (O2CB), enables functioning with user-space cluster stacks such as Pacemaker (`pcmk`), CMAN (`cman`), and no cluster stack (local mount)
- **Buffered, Direct, Asynchronous, Splice and Memory Mapped I/Os** – Supports all modes of I/Os for maximum flexibility and performance. OCFS2 is fully cache coherent.
- **Comprehensive Tools Support** – Provides a familiar EXT3-style tool-set that uses similar parameters for ease of use

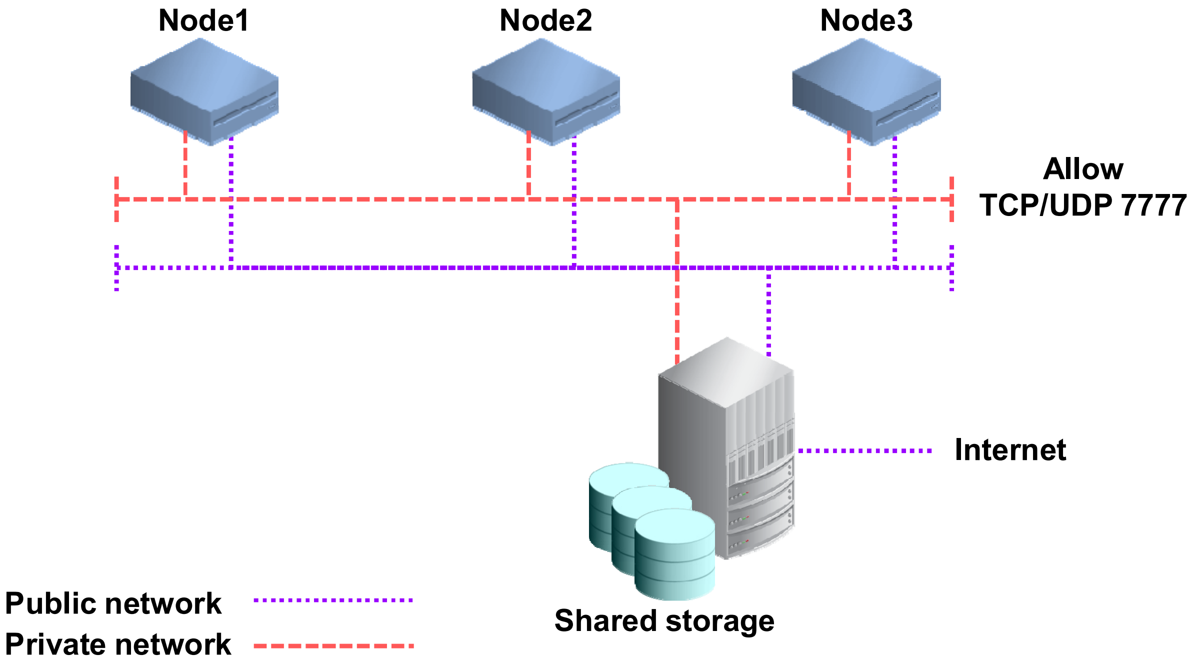
Using OCFS2

1. Configure shared storage.
2. Configure a private network for nodes (recommended).
3. Configure the firewall.
4. Install the Unbreakable Enterprise Kernel (UEK).
5. Install the `ocfs2-tools` RPM package.
6. Configure the `panic_on_oops` and `panic` kernel settings.
7. Configure the cluster layout configuration.
   – Use the `/sbin/o2cb` utility.
8. Configure and start the O2CB cluster service.
   – Use the `/etc/init.d/o2cb` initialization script.
9. Create OCFS2 volumes.
   – Use the `mkfs.ocfs2` utility.
10. Mount the OCFS2 volumes.

ORACLE

The high-level steps to install and use OCFS2 are summarized in this slide.

Preparing for OCFS2



This slide illustrates a sample configuration with shared storage for cluster nodes, a private network between the nodes (`192.168.1.0`), and the requirement to allow access on the private network for TCP and UDP port `7777`.

Each node in the cluster requires access to shared storage. For cluster file systems, typically this shared storage device is a shared SAN disk, an iSCSI device, or a shared virtual device on an NFS server.

In the example in the slide, each node (guest VM) shares `/dev/xvdb` as configured with the following entry in the `vm.cfg` files for each VM:

```
[dom0]# grep hdb /OVS/running_pool/host*/vm.cfg
host01/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,hdb,w!',
host02/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,hdb,w!',
host03/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,hdb,w!',
```

A shared disk is the same as a normal virtual disk, except that it uses `w!` in `vm.cfg` instead of just `w`. The entries in all `vm.cfg` files point to the same `.img` file with the same `hdb` (which translates to `xvdb`) identifier with `w!`.

It is also recommended that you configure a private interconnect between the nodes.

OCFS2 requires the nodes to be alive on the network, and sends regular keepalive packets (heartbeat) to ensure that they are alive. A private interconnect is recommended to avoid a network delay being interpreted as a node disappearing on the network, which could lead to node self-fencing. You can use OCFS2 without using a private network, but such a configuration increases the probability of a node fencing itself out of the cluster due to an I/O heartbeat timeout.

In this example, each node has a network interface on the public network (`192.0.2.0`) and an interface on the private network (`192.168.1.0`). The following `ifconfig` command displays the network configuration for the host01 node, with `eth0` on the public network and `eth1` on the private network:

```
# ifconfig
eth0       Link encap:Ethernet HWaddr 00:16:3E:00:01:01
           inet6 addr:192.0.2.101 ...
...
eth1       Link encap:Ethernet HWaddr 00:16:3E:00:02:01
           inet6 addr:192.168.1.200 ...
...
lo         Link encap:Local Loopback
           inet addr:127.0.0.1 ...
...
```

The host02 and host01 VMs have the same configuration. The host03 VM guest has a slightly different configuration in that it has two network interfaces on the public network, `eth0` and `eth1`, and `eth2` on the private network.

The O2CB cluster also requires iptables to be disabled or modified to allow network traffic on the private network interface. By default, the cluster uses both TCP and UDP over port `7777`. This port number is specified in the cluster configuration file. You can either write an iptable rule to allow access, or disable iptables.

OCFS2 Software

- OCFS2 software has a kernel component and a user-space component.
- The kernel component is bundled with the Unbreakable Enterprise Kernel beginning with Oracle Linux 5.
  - The kernel component includes the core file system and the O2CB cluster stack.
- The user-space component is provided by the `ocfs2-tools` RPM.
  - This RPM provides the command-line interface utilities that are used to format, tune, mount, and check the file system.
- Use the following commands to upgrade the kernel and install the user-space RPM:

```
# yum install kernel-uek
# yum install ocfs2-tools
```

OCFS2 software has a kernel component and a user-space component:

- **Kernel** – The kernel component is bundled with the Unbreakable Enterprise Kernel beginning with Oracle Linux 5. This component includes the core file system and the O2CB cluster stack.
- **User-space** – The `ocfs2-tools` package provides the command-line interface utilities that are used to format, tune, mount, and check the file system. The graphical user interface (`ocfs2console` package) is no longer supported.

Use the `yum` command to install the UEK kernel (`kernel-uek`) and the `ocfs2-tools` packages. You need to reboot your system after upgrading to a new kernel.

```
# yum install kernel-uek
# yum install ocfs2-tools
```

The following is a list of files included with the `ocfs2-tools` package:

```
# rpm -ql ocfs2-tools
/etc/init.d/o2cb
/etc/init.d/ocfs2
/etc/sysconfig/o2cb
```

List of files in the `ocfs2-tools` package (continued):

```
/sbin/debugfs.ocfs2
/sbin/fsck.ocfs2
/sbin/mount.ocfs2
/sbin/mounted.ocfs2
/sbin/o2cb
/sbin/o2cb_ctl
/sbin/o2image
/sbin/ocfs2_hb_ctl
/sbin/tunefs.ocfs2
/usr/bin/o2info
/usr/sbin/o2hbmonitor
```

The package installs documentation in the `/usr/share/doc/ocfs2-tools-<version>` directory. Also, `man` pages are installed for many of the OCFS2 utilities.

The OCFS2 kernel modules included with the UEK kernel (version `3.8.13-26.1.1.el6uek.x86_64`) are:

```
# ls -R /lib/modules/3.8.13-26.1.1.el6uek.x86_64/kernel/fs/ocfs2
| grep .ko
ocfs2.ko
ocfs2_stackglue.ko
ocfs2_stack_o2cb.ko
ocfs2_stack_user.ko
ocfs2_nodemanager.ko
ocfs2_dlm.ko
ocfs2_dlmfs.ko
```

Ideally, each node in the cluster is running the same version of the OCFS2 software and a compatible version of the Oracle Linux Unbreakable Enterprise Kernel. A cluster can run with mixed versions of OCFS2 and kernel, for example, when performing a rolling update of a cluster. The cluster node that is running the lowest version of the software determines the set of usable features.

Kernel Configuration

- Two kernel settings must be configured for O2CB to function properly:
  - `panic_on_oops` – Enable this to change a kernel oops into a panic.
  - `panic` – Specify the number of seconds after a panic that the system is auto-reset.
- To manually enable `panic_on_oops` and set a 30-second timeout for reboot on `panic`:

```
# echo 1 > /proc/sys/kernel/panic_on_oops
# echo 30 > /proc/sys/kernel/panic
```

- Configure persistent settings in `/etc/sysctl.conf`:

```
kernel.panic_on_oops = 1
kernel.panic = 30
```

You must set two kernel settings for the O2CB cluster stack to function properly. The first kernel setting is `panic_on_oops`, which you must enable to change a kernel oops into a panic. If a kernel thread required for O2CB crashes, the system is reset to prevent a cluster hang.

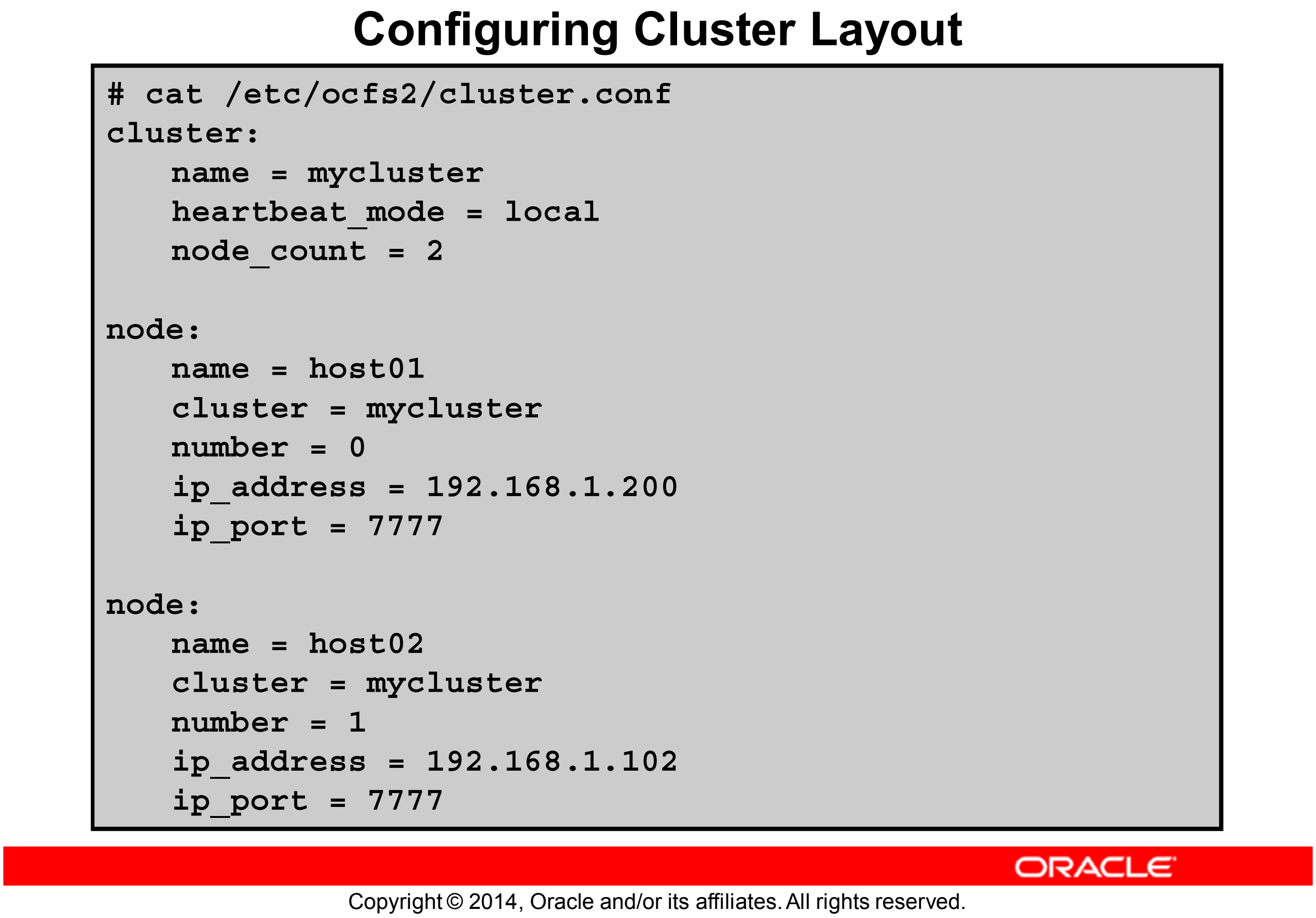
The other kernel setting is `panic`, which specifies the number of seconds after a panic that the system is automatically reset. The default setting is zero, which disables auto-reset, in which case the cluster requires manual intervention. The recommended setting is 30 seconds but you can set it higher for large systems.

To manually enable `panic_on_oops` and set a 30-second timeout for reboot on `panic`:

```
    # echo 1 > /proc/sys/kernel/panic_on_oops
    # echo 30 > /proc/sys/kernel/panic
```

To make these settings persistent, add the following entries to the `/etc/sysctl.conf` file:

```
    kernel.panic_on_oops = 1
    kernel.panic = 30
```



Configuring Cluster Layout

```
# cat /etc/ocfs2/cluster.conf
cluster:
    name = mycluster
    heartbeat_mode = local
    node_count = 2

node:
    name = host01
    cluster = mycluster
    number = 0
    ip_address = 192.168.1.200
    ip_port = 7777

node:
    name = host02
    cluster = mycluster
    number = 1
    ip_address = 192.168.1.102
    ip_port = 7777
```

This slide shows a sample cluster layout configuration. By default, the cluster layout configuration file for OCFS2 is `/etc/ocfs2/cluster.conf`. It is not necessary to configure this file when mounting an OCFS2 volume on a stand-alone, non-clustered system.

After creating this configuration file on one node in the cluster, copy the file to all nodes in the cluster. If you edit this file on any node, ensure that the other nodes are updated as well. When adding a new node to the cluster, update this configuration file on all nodes before mounting the OCFS2 file system from the new node.

The example configuration file in the slide has two sections (or stanzas):

- **Cluster –** This stanza specifies the parameters for the cluster. The configuration file typically contains only one cluster stanza. It can contain multiple cluster stanzas; however, only one cluster can be active at any time.
- **Node –** This stanza specifies the parameters for the individual nodes in the cluster. The configuration file typically contains multiple node stanzas.

You can use a text editor to manually create the `/etc/ocfs2/cluster.conf` file or use the `o2cb` utility to create and modify the configuration file. It is recommended that you use the `o2cb` command to modify the configuration file. This command ensures that entries in the configuration file are formatted correctly.

If you edit the configuration file manually, note the following guidelines:

- The `cluster:` and `node:` headings must start in the first column and end with a colon (`:`).
- Each parameter entry must be indented by one tab space.
- A blank line must separate each stanza that defines the cluster or a node.

The example in the slide describes a two-node cluster. In this example, the `cluster:` stanza parameters are:

- `name` – Specifies the name of the cluster
- `heartbeat_mode` – Specifies either the `local` or `global` heartbeat mode. The default is `local`. OCFS2 heartbeat is discussed later in this lesson.
- `node_count` – Specifies the total number of nodes in the cluster

In this example, the `node:` stanza parameters are:

- `name` – Specifies the host name of the node. The node name must match the host name but does not need to include the domain name.
- `cluster` – Specifies the name of the cluster to which the node belongs
- `number` – Specifies a unique node number from 0–254. When adding a new node, the number defaults to the lowest unused node number.
- `ip_address` – Specifies the IP address of the node. It is recommended that this address should be a private interface address.
- `ip_port` – Specifies the IP port number that the cluster uses for private cluster communication. By default, the cluster uses both TCP and UDP over port 7777.

`o2cb` Utility

- Use the `/sbin/o2cb` command to add, remove, and list information in the cluster configuration file.
- The syntax is:

```
o2cb [--config-file=path] [-h|--help] [-v|--verbose] [-
   V|--version] COMMAND [ARGS]
```

- Some of the available *`COMMAND`* parameters:
  - `add-cluster <cluster-name>`
  - `add-node <cluster-name> <node-name> [--ip <ip-address>] [--port <port>] [--number <node-number>]`
  - `heartbeat-mode <cluster-name> [local|global]`
  - `list-cluster <cluster-name> [--oneline]`
  - `list-nodes <cluster-name> [--oneline]`

Use the `o2cb` utility to add, remove, and list information in the cluster layout configuration file. You can also use this utility to register and unregister the cluster, and to start and stop the global heartbeat. The syntax for the command is:

```
    o2cb [--config-file=path] [-h|--help] [-v|--verbose] [-V|--
    version] COMMAND [ARGS]
```

Use the `--config-file=`*`path`* option to override the default configuration file, `/etc/ocfs2/cluster.conf`.

Available `COMMAND` parameters include:

- `add-cluster <cluster-name>` – Adds the specified `<cluster-name>` to the configuration file. The configuration file can specify multiple clusters but only one cluster can be active.
- `remove-cluster <cluster-name>` – Removes the specified `<cluster-name>` from the configuration file. This command also removes all nodes and heartbeat regions associated with the cluster.
- `add-node <cluster-name> <node-name>` – Adds the specified `<node-name>` to the specified `<cluster-name>` to the configuration file. You can optionally specify an IP address `[--ip <addr>]`, port number `[--port <port>]`, and node number `[--number <node>]`.

- `remove-node` *`<cluster-name>`* *`<node-name>`* – Removes the specified *`<node-name>`* from the specified *`<cluster-name>`* from the configuration file
- `add-heartbeat` *`<cluster-name>`* `[`*`uuid|device`*`]` – Adds a heartbeat region for the specified *`<cluster-name>`* to the configuration file
- `remove-heartbeat` *`<cluster-name>`* `[`*`uuid|device`*`]` – Removes a heartbeat region for the specified *`<cluster-name>`* from the configuration file
- `heartbeat-mode` *`<cluster-name>`* `[local|global]` – Specifies the heartbeat mode for the specified *`<cluster-name>`* in the configuration file
- `list-clusters` – Lists all the cluster names in the configuration file
- `list-cluster` *`<cluster-name>`* `[--oneline]` – Lists all the nodes and heartbeat regions associated with the specified *`<cluster-name>`* in the configuration file. The optional `--oneline` argument displays the output in a condensed format.
- `list-nodes` *`<cluster-name>`* `[--oneline]` – Lists all the nodes associated with the specified *`<cluster-name>`* in the configuration file
- `list-heartbeats` *`<cluster-name>`* `[--oneline]` – Lists all the heartbeat regions associated with the specified *`<cluster-name>`* in the configuration file
- `register-cluster` *`<cluster-name>`* – Registers the specified *`<cluster-name>`* in the configuration file with `configfs`
  - The preceding `configfs` is referred to as a synthetic (or virtual) file system. This is a generic kernel component, which is also used by `netconsole` and `fs/dlm`. OCFS2 tools use it to communicate the list of nodes in the cluster, details of the heartbeat device, and cluster timeouts to the in-kernel node manager. The `/etc/init.d/o2cb` initialization script mounts this file system at `/sys/kernel/config`.
- `unregister-cluster <cluster-name>` – Unregisters the specified *`<cluster-name>`* from `configfs`
- `start-heartbeat` *`<cluster-name>`* – Starts the global heartbeat on all regions for the specified *`<cluster-name>`* in the configuration file. It silently exits if global heartbeat has not been enabled.
- `stop-heartbeat` *`<cluster-name>`* – Stops the global heartbeat on all regions for the specified *`<cluster-name>`*

Examples of Using the `o2cb` Command

To create a cluster named `mycluster`:

```
# o2cb add-cluster mycluster
```

To add the `host01` node to `mycluster`:

```
# o2cb add-node mycluster host01 --ip 192.168.1.200
```

To specify `/dev/sda1` as a global heartbeat device:

```
# o2cb add-heartbeat mycluster /dev/sda1
```

To enable global heartbeat:

```
# o2cb heartbeat-mode mycluster global
```

To list `mycluster` configuration information:

```
# o2cb list-cluster mycluster
```

OCFS2 Heartbeat

- The O2CB cluster stack uses a heartbeat to determine whether a node is dead or alive.
- If a node loses network connectivity, it fences itself off from the cluster.
  - OCFS2 panics the node that is fenced off.
- There are two heartbeat modes: `local` and `global`.
  - Local – The heartbeat is started when a volume is mounted.
  - Global – The heartbeat is started when the O2CB cluster stack is started.
- The default heartbeat mode is local and is recommended for small clusters (fewer than five mounts).
- Heartbeat mode and heartbeat regions are configured in `/etc/ocfs2/cluster.conf`.

Connectivity of nodes and storage devices must be assured to prevent file system corruption. OCFS2 sends regular keepalive packets (heartbeat) to ensure that the nodes are alive. There are two types of heartbeat modes: *local* and *global*.

Local Heartbeat

In local heartbeat mode, a heartbeat is established when a volume is mounted by a node. The O2CB cluster stack does disk heartbeat on a per-mount basis on an area on disk, called the heartbeat file, which is reserved during format. The heartbeat thread is started and stopped automatically during mount and unmount. Each node that has a file system mounted writes every two seconds to its block in the heartbeat file. Each node opens a TCP connection to every node that establishes a heartbeat. If the TCP connection is lost for more than 10 seconds, the node is considered dead, even if the heartbeat is continuing.

If a node loses network connectivity to more than half of the heartbeating nodes, it has lost the quorum and fences itself off from the cluster. A *quorum* is the group of nodes in a cluster that are allowed to operate on the shared storage. *Fencing* is the act of forcefully removing a node from a cluster. A node with OCFS2 mounted fences itself when it realizes that it does not have quorum in a degraded cluster. It does this so that other nodes do not attempt to continue trying to access its resources. OCFS2 panics the node that is fenced off. A surviving node then replays the journal of the fenced node to ensure that all updates are on disk.

Local heartbeat requires as many heartbeat threads as there are mounts. This becomes a problem on clusters having five or more mounts. While each heartbeat I/O is small—one sector write and a maximum of 255 sectors read every two seconds—the amount of I/O operations per second (IOPS) can add up. Also, because the heartbeat is started on every mount, the mount is slow due to the need to wait for the heartbeat thread to stabilize. And because the number of mounts on each node in a cluster can vary, a node must self-fence if the heartbeat I/O times out to even one device.

Global Heartbeat

A solution to this problem is *global* heartbeat. This heartbeat scheme decouples the mount with the heartbeat. It allows you to mount 50 or more volumes without the additional heartbeat I/O overhead. Mounts are faster because there is no need to wait for the heartbeat thread to stabilize. And the loss of one heartbeat device need not force the node to self-fence.

With global heartbeat, you can configure heartbeat devices on all nodes. The heartbeat is started when the O2CB cluster stack is started. All nodes in the cluster ensure that the devices are the same on all nodes. A node self-fences if the heartbeat I/O times out on 50% or more of the devices. A recommendation is to set up at least three heartbeat devices. Any fewer and the node must self-fence on losing just one device.

The list of heartbeat devices is stored in `/etc/ocfs2/cluster.conf`. The notation includes a new `heartbeat:` stanza that has the heartbeat region and the cluster name. Use the region and not the device name so as to not force stable and consistent device names across the cluster. The heartbeat device is either an existing `ocfs2` volume that you mount, or an `ocfs2` volume that is specifically formatted as a heartbeat device, using the `mkfs.ocfs2 -H` command.

The cluster stanza has a heartbeat mode that is set to local or global. A cluster can have up to 32 heartbeat regions. Regions are named using the Universally Unique Identifier (UUID). The following example specifies two heartbeat stanzas for the `mycluster` cluster:

```
heartbeat:
        region = 908A022988C34A0DB6BC38C43C6B1461
        cluster = mycluster

heartbeat:
        region = 5678675678ABCDFE309888C34A0DB6B2
        cluster = mycluster


cluster:
        node_count = 10
        heartbeat_mode = global
        name = mycluster
```

Refer to https://oss.oracle.com/projects/ocfs2-tools/src/branches/global-heartbeat/documentation/o2cb/heartbeat-configuration.txt for additional information about heartbeat configuration. Also see https://blogs.oracle.com/wim/entry/ocfs2_global_heartbeat for an example of configuring global heartbeat.

O2CB Cluster Timeouts

- An O2CB cluster has four configurable cluster timeouts:
  - Heartbeat Dead Threshold – Number of two-second iterations before a node is considered dead
  - Network Idle Timeout – Time in milliseconds before a network connection is considered dead
  - Network Keepalive Delay – Maximum delay in milliseconds before a keepalive packet is sent to another node
  - Network Reconnect Delay – Minimum delay in milliseconds between connection attempts
- Cluster timeout values are configured by using the following command:

```
# service o2cb configure
```

- Timeout values are stored in `/etc/sysconfig/o2cb`.

In addition to configuring the cluster layout in the `/etc/ocfs2/cluster.conf` file, the O2CB cluster stack configuration consists of cluster timeout configuration. O2CB has four configurable cluster timeouts that are specified in the `/etc/sysconfig/o2cb` file:

- **Heartbeat Dead Threshold** – Specifies the number of two-second iterations before a node is considered dead. To convert the timeout in seconds to the number of iterations, divide the timeout in seconds by 2 and add 1. For example, to specify a 60-second timeout, set the threshold to 31 ((60 / 2) + 1). This is the default timeout value. To specify a timeout value of 120 seconds, set the threshold to 61. The heartbeat threshold must be the same on all nodes of the cluster.
- **Network Idle Timeout** – Specifies the time in milliseconds (ms) before a network connection is considered dead. The default is 30,000 ms.
- **Network Keepalive Delay** – Specifies the maximum delay in milliseconds before a keepalive packet is sent to another node. If the node is alive, it is expected to respond. The default is 2,000 ms.
- **Network Reconnect Delay** – Specifies the minimum delay in milliseconds between connection attempts. It defaults to 2,000 ms.

Cluster timeouts are configured by using the `/etc/init.d/o2cb` initialization script.

`o2cb` Initialization Script

- Use `/etc/init.d/o2cb` to configure the cluster stack and cluster timeout values:

```
# service o2cb configure
```

- Use `o2cb` to check the status of the cluster stack:

```
# service o2cb status
```

- Use `o2cb` to start the cluster stack:

```
# service o2cb online
```

- Use `o2cb` to stop the cluster stack:

```
# service o2cb offline
```

- Use `o2cb` to unload the cluster stack:

```
# service o2cb unload
```

- Additional `o2cb` commands exist to manage the cluster.

Use the `/etc/init.d/o2cb` initialization script to configure the cluster stack, as well as to configure the cluster timeout values. Run the following command on each node of the cluster:

```
# service o2cb configure
Configuring the O2CB driver.
This will configure the on-boot properties of the O2CB driver.
The following questions will determine whether the driver is
loaded on boot. The current values will be shown in brackets
('[]'). Hitting <ENTER> without typing an answer will keep that
current value. Ctrl-C will abort.
Load O2CB driver on boot (y/n) [n]: y
```

Respond with `y` to load the cluster stack driver at boot time.

```
Cluster stack backing O2CB [o2cb]: ENTER
```

Provide the name of the cluster stack service. The default and correct response is `o2cb`.

```
Cluster to start on boot (Enter "none" to clear) [ocfs2]:
mycluster
```

Provide the name of the cluster that you created in the cluster layout configuration file.

The cluster stack configuration continues as follows, prompting for cluster timeout information:

```
Specify heartbeat dead threshold (>=7) [31]: ENTER
Specify network idle timeout in ms (>=5000) [30000]: ENTER
Specify network keepalive delay in ms (>=1000) [2000]: ENTER
Specify network reconnect delay in ms (>=2000) [2000]: ENTER
```

After responding to all the queries, additional information appears as follows:

```
Writing O2CB configuration: OK
Loading filesystem “configfs”” OK
Mounting configfs filesystem at /sys/kernel/config: OK
Loading stack plugin “o2cb”: OK
Loading filesystem “ocfs2_dlmfs”: OK
Creating directory ‘/dlm’: OK
Mounting ocfs2_dlmfs filesystem at /dlm: OK
Setting cluster stack "o2cb": OK
Starting O2CB cluster “mycluster”: OK
Setting O2CB cluster timeouts : OK
```

`configfs` is loaded and mounted on `/sys/kernel/config`. Another synthetic, or virtual, file system, `ocfs2_dlmfs`, is loaded and mounted on `/dlm`. OCFS2 uses DLM to manage concurrent access from cluster nodes. DLM itself uses `ocfs2_dlmfs`, which is separate from the actual OCFS2 file systems on your system.

Use the following command to view the settings for the cluster stack:

```
# service o2cb status
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster mycluster: Online
  Heartbeat dead threshold = 31
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Local
Checking O2CB heartbeat: Not active
```

The `o2cb` script has additional commands to manage the cluster. Enter the following command to display usage:

```
# service o2cb
```

`mkfs.ocfs2` Utility

- Use the `mkfs.ocfs2` command to create an OCFS2 volume on a device partition.
  - The utility requires the O2CB cluster service to be online.
- Some of the options:
  - `-b` *`<block-size>`* – Specifies the smallest unit of I/O performed by the file system. The default size is 4 KB.
  - `-c` *`<cluster-size>`* – Specifies the smallest unit of space allocated for file data. The default size is 4 KB.
  - `-L` *`<volume-label>`* – Specifies a name for the volume
  - `-T` *`<filesystem-type>`* – Specifies usage of the file system. Options are `datafiles`, `mail`, and `vmstore`.
- Example of creating an OCFS2 volume with a label:

```
# mkfs.ocfs2 -L "myvolume" /dev/xvdb1
```

Use the `mkfs.ocfs2` command to create an OCFS2 volume on a device. You cannot use this utility to overwrite an existing volume that is mounted by another node in the cluster. This utility also requires that the cluster service is online. Though not required, it is recommended that you create OCFS2 volumes only on partitions. Only partitioned volumes can be mounted by label. Labels are a must for ease of management in a cluster environment.

Some of the available options for `mkfs.ocfs2` are listed as follows. All the options, with the exception of block size and cluster size, can be changed by using `tunefs.ocfs2`.

- `-b|--block-size` *`<block-size>`* – Specifies the smallest unit of I/O performed by the file system, and the size of inode and extent blocks. Supported block sizes are 512 bytes, 1 KB, 2 KB, and 4 KB. The default size is 4 KB.
- `-c|--cluster-size` *`<cluster-size>`* – Specifies the smallest unit of space allocated for file data. Supported sizes are 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 512 KB, and 1 MB. The default size is 4 KB. When using the volume for storing database files, do not use a cluster size value smaller than the database block size.
- `-L|--label` *`<volume-label>`* – Specifies a name for the volume. In a cluster, nodes can detect devices in a different order, which could result in the same device having different names on different nodes. Labeling allows consistent naming for OCFS2 volumes across a cluster.

Additional options for the `mkfs.ocfs2` command include:

- `-T filesystem-type` – Specifies usage of the file system. Valid types are:
  - `mail` – Specify this type when you intend to use the file system as a mail server. Mail servers perform many metadata changes to many small files, which require the use of a large journal.
  - `datafiles` – Specify this type when you intend to use the file system for database files. These file types use fewer fully allocated large files, with fewer metadata changes, and do not benefit from a large journal.
  - `vmstore` – Specify this type when you intend to store virtual machine images. These file types are sparsely allocated large files and require moderate metadata updates.

When not using the `-T filesystem-type` option (and not specifying sizes), `mkfs.ocfs2` calculates defaults based on the volume size:

1. Block size:
   a) 512 bytes if volume is less than 3 MB
   b) 1 KB if volume is more than 256 MB
   c) 2 KB if volume is more than 512 MB
   d) 4 KB if volume is more than 1 GB
2. Cluster size 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB if volume is 4 GB, 8 GB, and so on (respectively)
3. Journal size: Minimum 4 MB and up to 256 MB depending on volume size
4. Space reserved for extent allocation files: 0.1 percent of volume size

- `-J|--journal-options <`*`options`*`>` – Specifies the size of the write-ahead journal. The defaults are 64 MB for `datafiles`, 128 MB for `vmstore`, and 256 MB for `mail`.
- `-N|--node-slots <`*`number-of-node-slots`*`>` – Specifies the number of nodes that can concurrently mount the volume. Valid numbers range from 1 to 255. The default is 4. It is recommended to create more node slots than initially required. If you add more node slots later using `tunefs.ocfs2`, the journal is not contiguous, which causes poor performance.
- `--fs-features=<[no]`*`feature`*`>` – Specifies a comma-separated list of file system features to be enabled or disabled. Precede the feature with `no` to disable.
- `--fs-feature-level=<`*`feature-level`*`>` – Specifies one of the following levels:
  - `max-compat` – Enables only the features available in previous versions of OCFS2
  - `default` – Enables support for sparse files, unwritten extents, and inline data
  - `max-features` – Enables all the features that OCFS2 currently supports

The following example creates a file system with all defaults, and assigns a label:

```
# mkfs.ocfs2 -L "myvolume" /dev/xvdb1

mkfs.ocfs2 1.8.0

Cluster stack: classic o2cb

Label: myvolume

Features: sparse extended-slotmap backup-super unwritten inline-
data strict-journal-super xattr indexed-dirs refcount discontig-
bg

Block size: 4096 (12 bits)

Cluster size: 4096 (12 bits)

Volume size: 5362847744 (1309289 clusters) (1309289 blocks)

Cluster groups: 41 (tail covers 19049 clusters, rest cover 32256
clusters)

Extent allocator size: 4194304 (1 groups)

Journal size: 67108864

Node slots: 4

...

mkfs.ocfs2 successful
```

Notice the default values of 4 KB blocks and clusters, four node slots, and the default file system features.

Mounting OCFS2 Volumes

- Use the `mount` and `umount` commands to mount and unmount OCFS2 volumes, as in the following example:

```
# mount -L myvolume /u01
```

- Expect a delay when mounting and unmounting a volume.
- The O2CB cluster stack must be online before mounting.
- Use the `_netdev` option in `/etc/fstab`:

```
/dev/xvdb1 /u01 ocfs2 _netdev,defaults 0 0
```

- Start the `ocfs2` initialization service to mount at boot time.
- Use the following `mount` options when mounting OCFS2 volumes that contain Oracle database files:
  - `noatime`
  - `nointr`

Use the `mount` and `umount` commands to mount and unmount an OCFS2 volume, as you would with any other type of file system. The following example creates a mount point, and then mounts the OCFS2 volume by label:

```
# mkdir /u01
# mount -L myvolume /u01
```

There is a delay in the clustered mount operation because it must wait for the node to join the DLM domain. A clustered unmount is also not instantaneous, because it involves migrating all mastered lock-resources to the other nodes in the cluster. If the mount fails, use the `dmesg` command to view error messages.

The cluster stack must be online for the mount to succeed. Check the status of the cluster stack with the following `status` command. Use the `online` command to bring the cluster stack online if necessary.

```
# service o2cb status
# service o2cb online
```

To auto-mount volumes on startup, create entries in the /etc/fstab file and configure the ocfs2 initialization service to start at boot time. The ocfs2 service runs after the o2cb service starts the cluster. The ocfs2 service mounts all OCFS2 volumes listed in /etc/fstab. Configure both services to start at boot time as follows:

```
# chkconfig o2cb on
# chkconfig ocfs2 on
```

OCFS2 supports many of the mount command options supported by other Linux file systems. The following option is required when creating OCFS2 volume entries in /etc/fstab:

- _netdev – Specifies that the file system resides on a device that requires network access. This prevents the system from attempting to mount these file systems until the network has been enabled. The mount.ocfs2 command transparently appends this option during mount. You must, however, explicitly specify this option in /etc/fstab.

The following example specifies the _netdev option in /etc/fstab to mount the OCFS2 volume at boot time:

```
/dev/xvdb1  /u01  ocfs2  _netdev,defaults  0  0
```

Use the following mount options when using OCFS2 volumes for Oracle data files, control files, redo logs, voting disk, and the Oracle Cluster Registry (OCR):

- noatime – Disables unnecessary updates to access time (atime) on inodes
- nointr – Disables signals from interrupting I/Os in progress. This mount option is enabled by default, starting with OCFS2 Release 1.6.

The following example is an entry in /etc/fstab for an OCFS2 volume that hosts Oracle database files:

```
/dev/xvdb1  /u01  ocfs2  noatime,nointr  0  0
```

The o2net process handles network communication for all mounts. It gets the list of active nodes from O2HB and sets up a TCP/IP communication channel with each live node. It sends regular keepalive packets to detect any interruption on the channels.

In the following example, the mount command fails and produces an error message:

```
# mount –L myvolume /u01
mount.ocfs2: Invalid argument while mounting /dev/xvdb1 on /u01.
Check ‘dmesg’ for more information on this error.
```

Using the dmesg command to display more information provides the following message:

```
o2net: Connection to node host01 (num 0) at 192.168.1.101:7777
shutdown, state 7
```

The cause of this error is the inability of the nodes to communicate over the network on port 7777. Create an iptable rule to allow communication or stop the iptables service to fix this problem.

OCFS2 Tuning and Debugging

- The following OCFS2 utilities exist for tuning and debugging:
  - `tunefs.ocfs2` – Change file system parameters.
  - `fsck.ocfs2` – Detect and fix on-disk errors.
  - `mounted.ocfs2` – Detect and list all OCFS2 volumes.
  - `debugfs.ocfs2` – Display file system structures.
  - `o2image` – Back up the OCFS2 file system metadata from a device to a specified image file.
- OCFS2 uses the virtual file system, `debugfs`, to expose in-kernel information to user space.
  - This allows you to list the file system cluster locks, dlm locks, dlm state, and other states.
  - Mount the file system at `/sys/kernel/debug`.

ORACLE

Use the `tunefs.ocfs2` command to change file system parameters. You can change all parameters except block size and cluster size. The options for `tunefs.ocfs2` are similar to the options for the `mkfs.ocfs2` command. Some changes require the cluster service to be online.

You can also use the `tunefs.ocfs2` command with the `-Q` option to query the file system for specific attributes. The following example queries block size (`%B`), cluster size (`%T`), number of node slots (`%N`), volume label (`%V`), and volume UUID (`%U`) for the OCFS2 volume on `/dev/xvdb1`. The following example specifies labels for each attribute and displays each attribute on a new line:

```
# tunefs.ocfs2 -Q “Block Size: %B\nCluster Size: %T\nNode Slots:
%N\nVolume Label: %V\nVolume UUID: %U\n” /dev/xvdb1
Block Size: 4096
Cluster Size: 4096
Node Slots: 4
Volume Label: myvolume
Volume UUID: EC42AFE1CF614B0BB03F4ACAA4E5BC28
```

Use the `fsck.ocfs2` command to detect and fix on-disk errors. The command expects the cluster service to be online to ensure that the volume is not in use by another node. Unmount the file system before running `fsck.ocfs2`. The following example runs a full scan of the file system:

```
# fsck.ocfs2 -f /dev/xvdb1
fsck.ocfs2 1.8.0
Checking OCFS2 filesystem in /dev/xvdb1:
Label: myvolume
UUID: EC42AFE1CF614B0BB03F4ACAA4E5BC28
Number of blocks: 1309289
Block size: 4096
Number of clusters: 1309289
Cluster size: 4096
Number of slots: 4
/dev/xvdb1 was run with -f, check forced.
Pass 0a: Checking cluster allocation chains
Pass 0b: Checking inode allocation chains
Pass 0c: Checking extent block allocation chains
Pass 1: Checking inodes and blocks.
Pass 2: Checking directory entries.
Pass 3: Checking directory connectivity.
Pass 4a: checking for orphaned inodes
Pass 4b: Checking inodes link counts.
All passes succeeded.
```

The file system super block stores critical information such as the block size, cluster size, and locations of the root and system directories. A backup super block exists by default, as shown by the following example:

```
# tunefs.ocfs2 –Q “%M\n” /dev/xvdb1
backup-super strict-journal-super
```

The super block is not backed up on devices smaller than 1 GB. On devices that are larger than 1 GB, the `mkfs.ocfs2` command makes up to six backup copies of the super block. The `fsck.ocfs2` command refers to these six backups by number (1–6). In the unlikely event that the super block is corrupted, you can specify a backup to recover the super block. The following example overwrites the super block with the second backup:

```
# fsck.ocfs2 -f –r 2 /dev/xvdb1
fsck.ocfs2 1.8.0
[RECOVER_BACKUP_SUPERBLOCK] recover superblock information from
backup block#1048576 <n> y
```

Respond to the query with `y` to initiate the recovery.

Use the `mounted.ocfs2` command to detect and list all OCFS2 volumes. This command scans all devices in `/proc/partitions`. The following example lists all OCFS2 devices:

```
# mounted.ocfs2 -d
Device       Stack  Cluster  F  UUID              Label
/dev/xvdb1   o2cb               EC42AFE1CF614...  Myvolume
```

The following example lists the nodes currently mounting each volume:

```
# mounted.ocfs2 -f
Device       Stack  Cluster  F  Nodes
/dev/xvdb1   o2cb               host01, host02
```

Use the `debugfs.ocfs2` command to display file system structures. This is an interactive file system debugger for OCFS2, and is modeled after `debugfs` for `ext3` file systems. It allows you to display directory structures, examine inodes and backup files, and trace events in the OCFS2 driver. The following example lists all trace bits and their statuses:

```
# debugfs.ocfs2 -l
```

You can control file system tracing by enabling and disabling specific trace bits by using the following syntax:

```
debugfs.ocfs2 -l <tracebit> allow|off|deny
```

OCFS2 uses the virtual file system, `debugfs`, to expose its in-kernel information to user space. This allows you to list the file system cluster locks, dlm locks, dlm state, and other states. Access the information by mounting the file system at `/sys/kernel/debug`. To auto-mount `debugfs`, add the following entry to `/etc/fstab`:

```
debugfs   /sys/kernel/debug   debugfs   defaults   0  0
```

Use the `o2image` command to back up the OCFS2 file system metadata from a device to a specified image file. This image file contains the file system skeleton, including inodes, directory names, and file names. The image file does not include any file data. The following example creates the image file, `/tmp/xvdb1.img`, from the file system on `/dev/xvdb1`:

```
# o2image /dev/xvdb1 /tmp/xvdb1.img
```

You can determine the cause of a file system corruption or performance problem by using `debugfs.ocfs2` to open the image file and analyze the file system layout.

Quiz

Which of the following statements are true?

a. OCFS2 allows multiple nodes to read from and write to files on the same disk at the same time.

b. OCFS2 is the first native cluster file system to be included in the Linux kernel.

c. OCFS2 cannot be used in a non-clustered environment.

d. OCFS2 file systems can be mounted and used across multiple architectures at the same time.

Quiz

Which of the following statements are true?

a. The cluster layout configuration file for OCFS2 is `/etc/ocfs2/cluster.conf`.

b. This cluster layout configuration file needs to be present only on the master node.

c. Use the `/sbin/o2cb` command to add, remove, and list information in the cluster configuration file.

d. The configuration file includes only two sections (or stanzas): cluster and node.

Quiz

Which of the following statements are true?

a. OCFS2 sends regular keepalive packets (heartbeat) to ensure that nodes are alive.

b. There are two types of heartbeat modes: local and global.

c. In local heartbeat mode, heartbeat is on a per-mount basis on an area on disk. The heartbeat is established when a volume is mounted by a node.

d. With global heartbeat, you specify the devices on which you want a heartbeat thread. The heartbeat is started when the O2CB cluster stack is started.

Quiz

Which of the following are valid OCFS2 commands?

a. `service o2cb configure`
b. `mkfs.ocfs2`
c. `mounted.ocfs2`
d. `tunefs.ocfs2`

Introduction to Oracle Clusterware

- Cluster software groups together individual servers so they can cooperate as a single system.
- To applications and users, the separate servers appear as if they are one server.
- Oracle Clusterware is the required cluster technology for the Oracle multi-instance database, Oracle Real Application Clusters (RAC).
- It is available free of charge for Oracle Linux x86 and x86-64 architectures.
- To receive support for Oracle Clusterware, a licensed Oracle product must be used in the clustered environment.
- Oracle Clusterware is part of Grid Infrastructure (GI) along with Oracle Automatic Storage Management (ASM).

Oracle Clusterware is cluster software that groups together individual servers so they can cooperate as a single system. Each server looks like any stand-alone server. However, each server has additional processes that communicate with each other. To applications and end users, the separate servers appear as if they are one server.

A fundamental component of Oracle Real Application Clusters (Oracle RAC), Oracle Clusterware enables high availability for applications and databases managed in the cluster environment including Oracle Single Instance Databases, Oracle Application Servers, Oracle Enterprise Manager components, third-party databases, and other applications.

Oracle Clusterware for Unbreakable Linux is available free of charge for Linux x86 and Linux x86-64. To receive support for Oracle Clusterware, a licensed Oracle product must be used in the clustered environment. Licensed Oracle products include Oracle VM Server for x86, Oracle VM Server for SPARC, Oracle Linux, Oracle Solaris, or any of the licensed Oracle applications.

Oracle Clusterware is part of Grid Infrastructure (GI). Grid Infrastructure comprises Oracle Clusterware and Oracle Automatic Storage Management (ASM). Oracle Clusterware and Oracle ASM can only co-exist together. They are installed and maintained together.

Oracle Clusterware Hardware Requirements

- One or more servers (nodes), each having a minimum of two network interface cards
  - One network interface on the public network
  - One network interface on the private, interconnect network
  - The interconnect network is known only to the cluster nodes.
- Cluster-aware storage that is connected to each node in the cluster
  - Oracle Database supports Storage Area Network (SAN) storage or Network Attached (NAS) storage.
- There are generally at least two connections from each server to the cluster-aware storage to provide redundancy.

ORACLE

A cluster is made up of one or more servers. A server in a cluster is similar to any stand-alone server, but a cluster requires a second network called the interconnect network. Therefore, the server minimally requires two network interface cards: one for the public network and one for the private network. The interconnect network is a private network using a switch (or multiple switches) that only the nodes in the cluster can access. Crossover cables are not supported for use with Oracle Clusterware interconnects.

If you are implementing the cluster for high availability, configure redundancy for all components of the infrastructure. Configure:

- A network interface for the public network (generally this is an internal LAN)
- A redundant network interface for the public network
- A network interface for the private interconnect network
- A redundant network interface for the private interconnect network

The cluster requires cluster-aware storage that is connected to each server in the cluster. This can be referred to as a multihost device. Oracle Database supports Storage Area Network (SAN) storage or Network Attached (NAS) storage. Similar to the network, there are generally at least two connections from each server to the cluster-aware storage to provide redundancy. Most servers have at least one local disk that is internal to the server. Often, this disk is used for the operating system binaries and you can also use this disk for the Oracle binaries. The benefit of each server having its own copy of the binaries is that it simplifies rolling upgrades.

Oracle Clusterware Files

- Oracle Clusterware requires two files:
  - A voting disk to record node membership information
  - The OCR to record cluster configuration information
- Oracle recommends that you configure multiple voting disks and the OCR for redundancy.
- The voting disk and the OCR must reside on shared storage.
- Oracle recommends storing these files in ASM which is also part of GI.

Oracle Clusterware requires two files: a voting disk to record node membership information and the OCR to record cluster configuration information. During the Oracle Clusterware installation, Oracle recommends that you configure multiple voting disks and the OCR. The voting disk and the OCR must reside on shared storage.

Voting Disk

Oracle Clusterware uses the voting disk to determine which instances are members of a cluster. The voting disk must reside on a shared disk. For high availability, Oracle recommends that you have a minimum of three voting disks. If you configure a single voting disk, use external mirroring to provide redundancy. You can have up to 32 voting disks in your cluster.

Oracle Cluster Registry (OCR)

Oracle Clusterware uses the OCR to store and manage information about the components that Oracle Clusterware controls, such as Oracle RAC databases, listeners, virtual IP addresses (VIPs), services, and applications. The OCR repository stores configuration information in a series of key-value pairs in a directory tree structure. Oracle recommends that you use a multiplexed OCR to ensure cluster high availability.

Summary

In this lesson, you should have learned how to:

- Prepare for an OCFS2 configuration
- Install the OCFS2 software packages
- Configure kernel settings for OCFS2
- Configure the cluster layout
- Configure and start the O2CB cluster service
- Create an OCFS2 volume
- Mount an OCFS2 volume
- Use OCFS2 tuning and debugging utilities
- Provide an introduction to Oracle Clusterware

Practice 9: Overview

This practice covers the following topics:

- Preparing your environment for OCFS2
- Installing and upgrading the required software packages
- Configuring the cluster layout
- Configuring and starting the O2CB cluster stack
- Creating an OCFS2 volume
- Mounting an OCFS2 volume
- Modifying the parameters of an existing OCFS2 volume
- Using OCFS2 debugging utilities

These practices assume that you completed Practice 2-1 and Practice 2-2, which used DHCP to assign an IP address to `eth1` on **host01**.

If you did not complete these practices, perform task 1 in Practice 9-1.

If you did complete Practice 2-1 and Practice 2-2, and `eth1` on **host01** has an IP address, skip task 1 in Practice 9-1 and go directly to task 2.

10

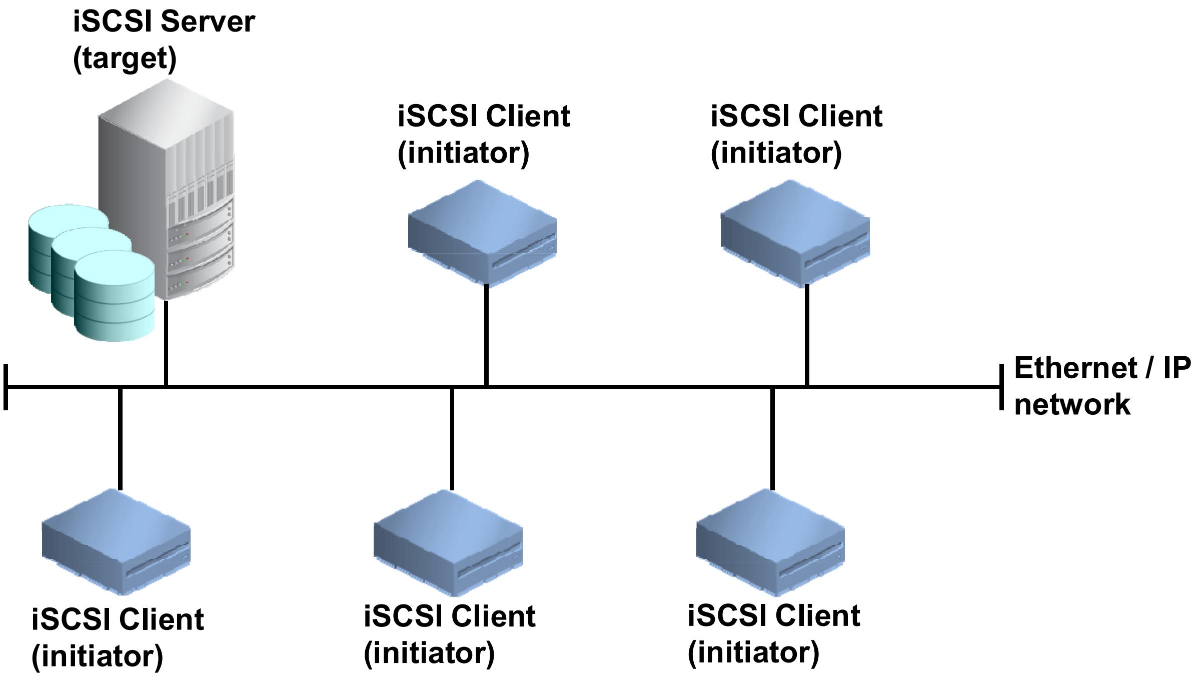
iSCSI and Multipathing

Objectives

After completing this lesson, you should be able to:

- Configure an iSCSI target
- Use the `tgtadm`, `tgt-admin`, and `tgt-setup-lun` utilities
- Configure an iSCSI software initiator
- Use the `iscsiadm` utility
- Describe Device Mapper Multipathing
- Use the `mpathconf` and `multipath` utilities
- Configure iSCSI multipathing

Introduction to iSCSI



Internet Small Computer System Interface (iSCSI) is an IP-based standard for connecting storage devices. iSCSI uses IP networks to encapsulate SCSI commands, allowing data to be transferred over long distances. iSCSI provides shared storage among a number of client systems. Storage devices are attached to servers (targets) and client systems (initiators) access the remote storage devices over IP networks. To the client systems, the storage devices appear to be locally attached. iSCSI uses the existing IP infrastructure and does not require any additional cabling, as is the case with Fibre Channel (FC) storage area networks.

This slide shows a simple Ethernet network with storage attached to an iSCSI server (target) and several client systems (initiators) able to access the shared storage over the network. The client initiators send SCSI commands over the IP network to the iSCSI target.

The iSCSI target can be a dedicated network-connected storage device, but can also be a general-purpose computer as is the case in this slide. Software to provide iSCSI target functionality is available for Oracle Linux and other operating systems. There are also specific-purpose operating systems that implement iSCSI target support. Two examples of open source network storage solutions are FreeNAS and Openfiler. The iSCSI part of this course focuses on the implementation of iSCSI targets and initiators in Oracle Linux.

iSCSI Target Files

- To configure an iSCSI server, install the `scsi-target-utils` software package:

```
# yum install scsi-target-utils
```

- The package installs several files, including the following:
  - `/etc/tgt/targets.conf` – iSCSI configuration file
  - `/usr/sbin/tgtd` – iSCSI target daemon
  - `/usr/sbin/tgtadm` – iSCSI target administration utility
  - `/usr/sbin/tgt-admin` – iSCSI target configuration script
  - `/usr/sbin/tgt-setup-lun` – Script that creates a target, adds a device to the target, and defines initiators that can connect to the target

ORACLE

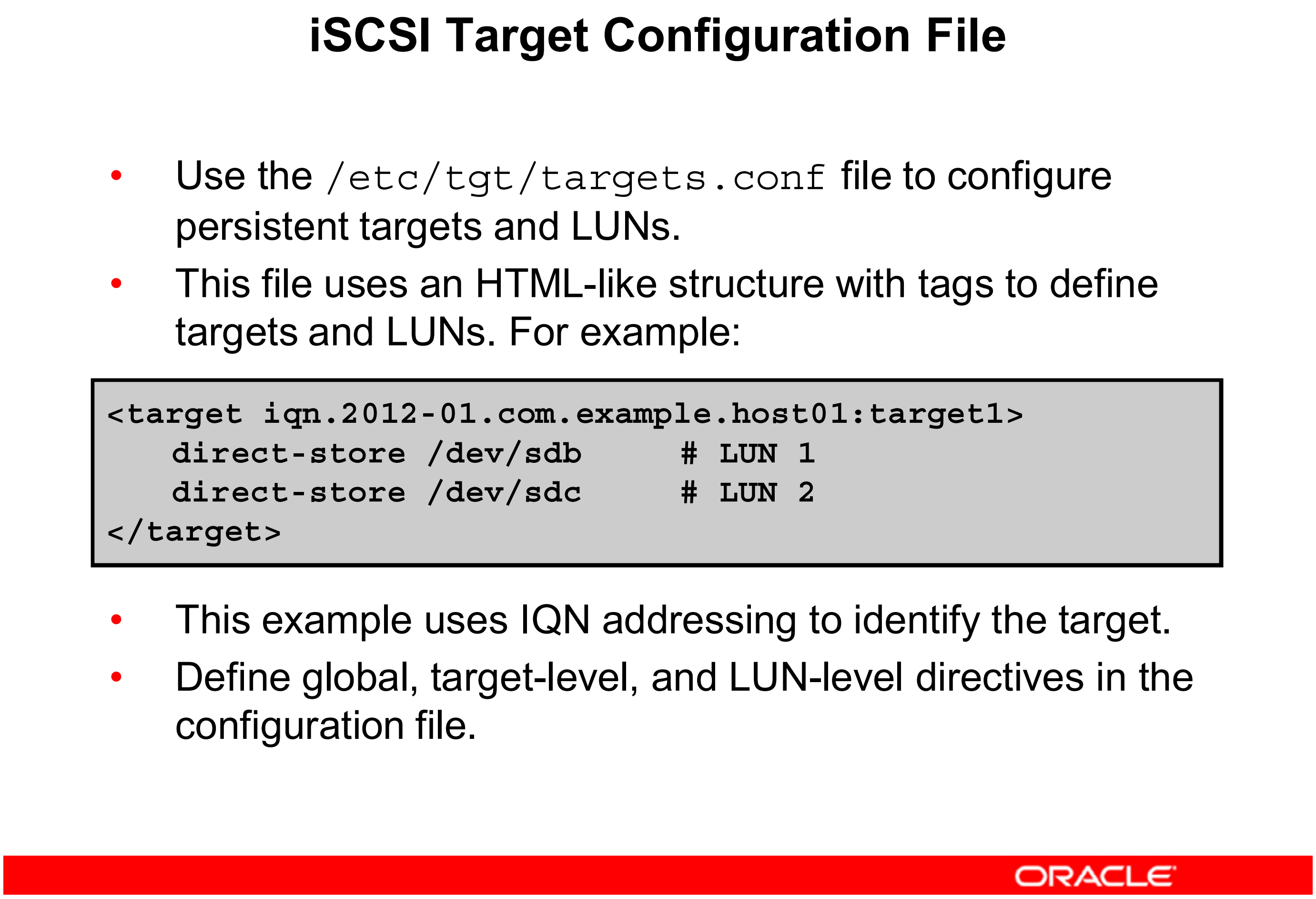
To configure an Oracle Linux system as an iSCSI server, install the `scsi-target-utils` software package:

```
    # yum install scsi-target-utils
```

The package installs several files, including the following:

- `/etc/tgt/targets.conf` – The iSCSI configuration file that is used to define targets and Logical Unit Numbers (LUNs)
- `/usr/sbin/tgtd` – The iSCSI target daemon
- `/usr/sbin/tgtadm` – The iSCSI target administration utility that is used to monitor and modify SCSI targets
- `/usr/sbin/tgt-admin` – A Perl script for creating persistent targets and LUNs
- `/usr/sbin/tgt-setup-lun` – A Bash script that creates a target, adds a device to the target, and defines initiators that can connect to the target

Man pages are also installed for `tgtadm`, `tgt-admin` and `tgt-setup-lun`.



iSCSI Target Configuration File

- Use the `/etc/tgt/targets.conf` file to configure persistent targets and LUNs.
- This file uses an HTML-like structure with tags to define targets and LUNs. For example:

```
<target iqn.2012-01.com.example.host01:target1>
   direct-store /dev/sdb     # LUN 1
   direct-store /dev/sdc     # LUN 2
</target>
```

- This example uses IQN addressing to identify the target.
- Define global, target-level, and LUN-level directives in the configuration file.

The configuration file for defining targets and LUNs is `/etc/tgt/targets.conf`. This file uses HTML-like structure with tags to define targets and LUNs. The file contains several sample configurations, all of which are commented out by default. An example is as follows:

```
<target iqn.2012-01.com.example.host01:target1>
     direct-store /dev/sdb     # LUN 1
     direct-store /dev/sdc     # LUN 2
</target>
```

In this example, the target is identified by an “iqn” identifier. This is an iSCSI Qualified Name (IQN), which uniquely identifies a target. IQN format addresses are most commonly used to identify a target. This address consists of the following fields:

- Literal iqn
- Date (in `yyyy-mm` format) that the naming authority took ownership of the domain
- Reversed domain name of the authority
- Optional “`:`” that prefixes a storage target name specified by the naming authority

Other address types include Extended Unique Identifier (EUI) and T11 Network Address Authority (NAA). The sample configuration file uses only IQN target addresses.

Global Directives

The configuration file defines global directives. A partial list of global directives is as follows:

- `<target <IQN>>` – Defines the start of a target definition. The target definition ends with `</target>`. Within the target block are target-level directives. Multiple targets can be defined.
- `default-driver <lld>` – Specifies a default low-level driver for all exported targets. Valid values are `iscsi` and `iser` (iSCSI Extensions for RDMA – Remote Direct Memory Access). The default is `iscsi`.
- `include <path>` – Includes the configuration from another configuration file identified by `<path>`
- `ignore-errors <yes|no>` – When set to `yes`, ignores errors from `tgtadm`. The default is `no`.
- `incomingdiscoveryuser <user> <password>` – Defines iSCSI incoming discovery authentication
- `outgoingdiscoveryuser <user> <password>` – Defines iSCSI outgoing discovery authentication

Many of the global directives can be overridden by defining them as specific target-level directives.

Target-Level Directives

Each target can export multiple block devices, or LUNs. A LUN represents an individually addressable (logical) SCSI device that is part of a physical SCSI device (target). A client system initiator negotiates with a target to establish connectivity to a LUN. Rather than mounting remote directories as would be done in NFS or CIFS environments, iSCSI systems format and directly manage file systems on iSCSI LUNs. A partial list of target-level directives is as follows:

- `backing-store <path>` – Defines a LUN, identified by `<path>`, which is exported by the target. This can be a regular file or block device. This directive is processed before `direct-store` directives.
- `direct-store <path>` – Defines a local SCSI device, identified by `<path>`. Device parameters, such as `VENDOR_ID` and `SERIAL_NUM` are passed to the new iSCSI LUN.
- `initiator-address <addr>` – Allows connections only from the specified IP address. The default is ALL if no address is specified.

LUN-Level Directives

These directives can be specified at the target-level and apply to all LUNs. Or they can be specified at the LUN level to apply to a specific LUN, for example:

```
<target iqn.2012-01.com.example.host01:target1>
     <direct-store /dev/sdb>     # LUN 1
          LUN directives
     </direct-store>
</target>
```

One example of a LUN-level directive is `write-cache <on|off>`, which defaults to `on`. Refer to the `target.conf` man page for details on all available directives and configuration examples.

`tgtadm` Utility

- Use the `tgtadm` utility to monitor and configure non-persistent iSCSI targets.
- The `tgtd` daemon must be running to use this utility:

```
# service tgtd start
```

- To show all the targets:

```
# tgtadm -o show -m target
```

- To add a new target:

```
# tgtadm -o new -m target --tid 3 --targetname
   iqn.2012.com.example.mypc:tgt3
```

- To add a new LUN (logical unit number `3`) to target ID `1`:

```
# tgtadm -o new -m logicalunit --tid 1 --lun 3 --
   backing-store /OVS/sharedDisk/physDisk5.img
```

The `tgtadm` utility is a binary executable file that is used to monitor and configure iSCSI targets. The following is a partial list of options for the `tgtadm` utility:

- `-L|--lld` *`<driver>`* – Specify a low-level driver. Valid values are `iscsi` and `iser`. The default is `iscsi`.
- `-m|--mode` *`<entity>`* – Specify the entity to perform an action on. Valid values are `target`, `logicalunit`, and `account`.
- `-o|--op` *`<action>`* – Specify the operation or action to perform. Valid values are `new`, `delete`, `show`, `update`, `bind`, and `unbind`.
- `-t|--tid` *`<id>`* – Specify the target ID.
- `-T|--targetname` *`<name>`* – Specify the target name.
- `-F|--force` – Use with the `delete` action to delete an active `I_T nexus` [a relationship between an initiator (`I`) and a target (`T`)].
- `-n|--name` *`<param>`*, `--value` *`<value>`* – Use with the `update` action to change the value *`<value>`* of a parameter *`<param>`*.
- `-l|--lun` *`<lun>`*: Specify a logical unit number.
- `-I|--initiator-address` *`<address>`*: Specify an initiator address for access control.

The following example is based on the contents of the configuration file listed below. The configuration file defines two targets with two LUNs for each target:

```
<target iqn.2011-12.com.example.mypc:tgt1>
     backing-store /OVS/sharedDisk/physDisk1.img
     backing-store /OVS/sharedDisk/physDisk2.img
     write-cache off
</target>
<target iqn.2011-12.com.example.mypc:tgt2>
     backing-store /OVS/sharedDisk/physDisk3.img
     backing-store /OVS/sharedDisk/physDisk4.img
     write-cache off
</target>
```

To show all the targets:

```
# tgtadm –o show –m target
Target 1: iqn.2011.12.com.example.mypc:tgt1
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET      00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            ...
        LUN: 1
            Type: disk
            SCSI ID: IET      00010001
            SCSI SN: beaf11
            Size: 10737 MB, Block size: 512
            Online: Yes
            Removable media: No
            Readonly: No
            Backing store type: rdwr
            Backing store path: /OVS/sharedDisk/physDisk1.img
...
```

To add a new target (targets are shown after the add):

```
# tgtadm –L iscsi -o new –m target --tid 3 --targetname
iqn.2012-11.com.example.mypc:tgt3
# tgtadm -o show –m target | grep Target
Target 1: iqn.2011-12.com.example.mypc:tgt1
Target 2: iqn.2011-12.com.example.mypc:tgt2
Target 3: iqn.2012-11.com.example.mypc:tgt3
```

To delete a target (the target must have no active `I_T nexus`, otherwise use the `delete --force` option):

```
# tgtadm -o delete –m target --tid 2
# tgtadm -o show -m target | grep Target
Target 1: iqn.2011-12.com.example.mypc:tgt1
Target 3: iqn.2012-11.com.example.mypc:tgt3
```

To add a new LUN (LUN `3`) to target ID `1`:

```
# tgtadm -o new -m logicalunit --tid 1 --lun 3 --backing-store
/OVS/sharedDisk/physDisk5.img
```

To add an IP address to the access control list for a target (only initiators with the address can access the target):

```
# tgtadm -o bind -m target --tid 1 --initiator-address 192.0.2.1
```

The `tgtadm` command is not persistent. Restarting the `tgtd` daemon reads the `/etc/tgt/targets.conf` file, and creates the targets and LUNs configured in this file.

```
# service tgtd restart
Stopping SCSI target daemon:             [  OK  ]
Starting SCSI target daemon:             [  OK  ]
# tgtadm -o show -m target | grep Target
Target 1: iqn.2011-12.com.example.mypc:tgt1
Target 2: iqn.2011-12.com.example.mypc:tgt2
```

`tgt-admin` Script

- The `tgt-admin` Perl script executes `tgtadm` commands.
- Pass an option to the script to do the following:
  - `-e` – Read `targets.conf` and run `tgtadm` commands.
  - `--delete` <*value*> – Delete all or selected targets.
  - `--offline` <*value*> – Pull all or selected targets in offline state.
  - `--online` <*value*> – Put all or selected targets in online state.
  - `--ready` <*value*> – Put targets in ready state.
  - `--update` <*value*> – Update configuration for targets.
  - `-s` – Show all the targets.
  - `-c` <*conf file*> – Specify an alternative configuration file.
  - `-h` – Show help for the `tgt-admin` script.

The `tgt-admin` Perl script uses `tgtadm` commands to create, delete, and show target information. For example, the following `tgt-admin` command executes the subsequent `tgtadm` command:

```
    # tgt-admin --show
    # tgtadm --op show --mode target
```

The following is a partial list of options for the `tgt-admin` script:

`-e|--execute` – Read `/etc/tgt/targets.conf` and execute `tgtadm` commands.

`-d|--delete` <*value*> – Delete all or selected targets.

`--offline` <*value*> – Put all or selected targets in offline state.

`--ready` <*value*> – Put all or selected targets in ready state.

`--update` <*value*> – Update configuration for all or selected targets.

`-s|--show` – Show all the targets.

`-c|--conf` <*conf file*> – Specify an alternative configuration file.

`-h|--help` – Display script usage.

`tgt-setup-lun` Script

- `tgt-setup-lun` is a Bash script that:
  - Creates a non-persistent target
  - Adds a device to the target
  - Optionally defines initiators that can connect to the target
- Syntax:

```
tgt-setup-lun -d device -n target_name [initiator_IP1
    initiator_IP2 ...]
```

- Example:

```
# tgt-setup-lun -d /OVS/sharedDisk/physDisk6.img -n
    new_tgt 192.0.2.101 192.0.2.102
Creating the new target (iqn.2001-04.com.<hostname>-
    new_tgt
...
```

The `tgt-setup-lun` Bash script creates a target, adds a device to the target, and defines initiators that can connect to the target. The `tgtd` daemon must be running to use this script. The syntax is as follows:

```
tgt-setup-lun -d device -n target_name [initiator_IP1...]
```

Provide a simple target name as the `-n` argument. The script appends an IQN format address, including the host name, to the target name you provide. If IP addresses are defined, they are added to the access list of allowed initiators for that target. If no IP addresses are specified, the target accepts any initiator.

The following example creates a target that uses `/OVS/sharedDisk/physDisk6.img` and allows connections from only IP addresses `192.0.2.101` and `192.0.2.102`:

```
# tgt-setup-lun -d /OVS/sharedDisk/physDisk6.img -n new_tgt
192.0.2.101 192.0.2.102

Creating the new target (iqn.2001-04.com.<hostname>-new_tgt

Adding a logical unit (/OVS/sharedDisk/physDisk6.img) to the
target

Accepting connections only from 192.0.2.101 192.0.2.102
```

iSCSI Initiators

- An iSCSI client functions as a SCSI initiator to access target devices on an iSCSI server.
- An iSCSI initiator sends SCSI commands over an IP network.
- There are two types of iSCSI initiators:
  - Software initiator – Uses an existing NIC
  - Hardware initiator – Uses a dedicated iSCSI HBA
- Oracle Linux and most other operating systems provide iSCSI software initiator functionality.

ORACLE

An iSCSI client functions as a SCSI initiator to access target devices on an iSCSI server. An iSCSI initiator sends SCSI commands over an IP network. There are two types of iSCSI initiators.

- **Software initiator:** A kernel-resident device driver uses the existing network interface card (NIC) and network stack to emulate SCSI devices. Some network cards offer TCP/IP Offload Engines (TOE), which perform much of the IP processing that is normally performed by server resources. These TOE cards have a built-in network chip, which creates the TCP frames. The Linux kernel does not support TOE directly; therefore, the card vendors write drivers for the OS.
- **Hardware initiator:** This uses a dedicated iSCSI Host Bus Adaptor (HBA) to implement iSCSI. Hardware initiators provide better performance because the processing is performed by the HBA rather than the server. You can also boot a server from iSCSI storage, which you cannot do with software initiators. The downside is that the cost of an iSCSI HBA is much higher than an Ethernet NIC.

Oracle Linux and most operating systems provide software initiator functionality.

iSCSI Initiator Files

- To configure an iSCSI initiator, install the `scsi-initiator-utils` software package:

```
# yum install iscsi-initiator-utils
```

- The package installs several files, including the following:
  - `/etc/iscsi/iscsid.conf` – The iSCSI initiator configuration file
  - `/sbin/iscsid` – The Open-iSCSI daemon
  - `/sbin/iscsiadm` – The Open-iSCSI administration utility that is used to discover and log in to iSCSI targets
  - `/sbin/iscsi-iname` – The iSCSI initiator name generation tool that is used to generate a unique iSCSI node name.
- Refer to http://www.open-iscsi.org for information about Open-iSCSI.

To configure an Oracle Linux system as an iSCSI initiator, install the `scsi-initiator-utils` software package. This package is the Linux Open-iSCSI Initiator.

```
# yum install scsi-initiator-utils
```

The package installs several files including the following:

- `/etc/iscsi/iscsid.conf`: The configuration file read by `iscsid` and `iscsiadm`. This file is heavily commented with descriptions for each configuration directive.
- `/sbin/iscsid`: The Open-iSCSI daemon that implements the control path and management facilities
- `/sbin/iscsiadm`: The Open-iSCSI administration utility used to discover and log in to iSCSI targets
- `/sbin/iscsi-iname`: The iSCSI initiator name generation tool used to generate a unique iSCSI node name

Refer to http://www.open-iscsi.org for information about Open-iSCSI. There is also a `README` file in the `/usr/share/doc/iscsi-initiator-utils-<version>/` directory, which describes Open-iSCSI.

`iscsiadm` Utility

- Use the `iscsiadm` utility to update, delete, insert, and query the persistent database:

```
# ls /var/lib/iscsi
ifaces  isns  nodes  send_targets  slp  static
```

- The `iscsiadm` modes include:
  - `-m discoverydb` – Query or update the database.
  - `-m discovery` – Discover iSCSI targets.
  - `-m node` – Perform an operation on a portal.
  - `-m session` – Perform an operation on a session.
  - `-m iface` – Perform an operation on a network interface.
- Additional options to `iscsiadm` include:
  - `-t` – Specify the discover type.
  - `-p` – Specify the iSCSI target portal.

Open-iSCSI persistent configuration is implemented as a database, which consists of a hierarchy of files and directories in the `/var/lib/iscsi/` directory:

```
# ls /var/lib/iscsi
ifaces  isns  nodes  send_targets  slp  static
```

Use the `iscsiadm` utility to update, delete, insert, and query the persistent database. Also use this utility to establish a session between a target and an initiator. Several different operational modes are available for the command.

- `discoverydb`: Updates or queries the Open-iSCSI database records
- `discovery`: Performs a discovery operation
- `node`: Performs an operation on a portal (IP:port) on an iSCSI target
- `session`: Performs an operation on a TCP connection between an initiator and a target
- `iface`: Performs an operation on a network interface

Additional options to `iscsiadm` include:

- `-type` – Specify the discover type.
- `-portal` – Specify the iSCSI target portal.

iSCSI Discovery

- Discovery makes targets available to the initiator.
- The following discovery methods are available:
  - SendTargets
  - Service Location Protocol (SLP)
  - Internet Storage Name Service (iSNS)
  - Static
- To discover targets using the SendTargets method:

```
# iscsiadm -m discovery -t st -p 192.0.2.1
```

- The database is updated by using the `iscsid.conf` settings.
- To browse the database:

```
# iscsiadm -m discoverydb -t st -p 192.0.2.1
```

Discovery is the process that makes the targets known to an initiator. It defines the method by which the iSCSI targets are found. The following discovery methods are available:

- **SendTargets:** This is a native iSCSI protocol that allows an iSCSI server to send a list of available targets to the initiator.
- **Service Location Protocol (SLP):** Servers use the SLP to announce available targets. The initiator can implement SLP queries to get information about these targets.
- **Internet Storage Name Service (iSNS):** Targets are discovered by interacting with one or more iSNS servers. iSNS servers record information about available targets. Initiators pass the address and an optional port of the iSNS server to discover targets.
- **Static:** The static target address is specified.

The following example uses the SendTargets discovery method to discover targets on IP address `192.0.2.1`. This command also starts the `iscsid` daemon if needed.

```
# iscsiadm -m discovery --type sendtargets -p 192.0.2.1
Starting iscsid:                                      [  OK  ]
192.0.2.1:3260,1 iqn.2011-12.com.example.mypc:tgt1
192.0.2.1:3260,1 iqn.2011-12.com.example.mypc:tgt2
```

After discovery, the `nodes` table and the `send_targets` tables in the database are updated:

```
# ls /var/lib/iscsi/nodes
iqn.2011-12.com.example.mypc:tgt1
iqn.2011-12.com.example.mypc:tgt2
iqn.2012-11.com.example.mypc:tgt3
# ls /var/lib/iscsi/send_targets
192.0.2.1,3260
```

You can view the database files and directories directly, or you can query the database with the `iscsiadm -m discoverydb` option.

The following example queries the `send_targets` table for entries from `192.0.2.1`:

```
# iscsiadm -m discoverydb –t st –p 192.0.2.1
# BEGIN RECORD 6.2.0-873.2.el6
discovery.startup = manual
discovery.type = sendtargets
discovery.sendtargets.address = 192.0.2.1
discovery.sendtargets.port = 3260
discovery.sendtargets.auth.authmethod = None
discovery.sendtargets.auth.username_in = <empty>
discovery.sendtargets.auth.password_in = <empty>
discovery.sendtargets.timeo.login_timeout = 15
discovery.sendtargets.use_discoveryd = No
discovery.sendtargets.discoveryd_poll_inval = 30
discovery.sendtargets.repoen_max = 5
discovery.sendtargets.timeo.auth_timeout = 45
discovery.sendtargets.timeo.active_timeout = 30
discovery.sendtargets.iscsi.MaxRecvDataSegmentLength = 32768
```

You can also use the `--discover` option to add (`-o new`), update (`-o update`), and delete (`-o delete`) records from the database. The following example adds new records in the database. If the discovery mechanism discovers records that are not in the database, they are created by using the `/etc/iscsi/iscsid.conf` discovery settings.

```
# iscsiadm -m discoverydb –t st –p 192.0.2.1 –o new --discover
```

The following example updates existing records in the database. If records are returned during discovery that currently exist in the database, they are updated with information from `/etc/iscsi/iscsid.conf`. No new records are added and stale records are not removed.

```
# iscsiadm -m discoverydb –t st –p 192.0.2.1 –o update --
discover
```

The following example deletes records from the database. If a record exists in the database, but is not returned during discovery, the record is removed from the database.

```
# iscsiadm -m discoverydb –t st –p 192.0.2.1 –o delete --
discover
```

iSCSI Initiator Sessions

- A session is a TCP connection between an initiator node port and a target node port.
- LUNs are not accessible until a session is established.
- Use the `-l` (or `--login`) option to establish a session:

```
# iscsiadm -m node -l
```

  - To log in to a specific target:

```
# iscsiadm -m node --targetname iqn.2011-
   12.com.example.mypc:tgt1 -p 192.0.2.1:3260 -l
```

- Use the `-u` (or `--logout`) option to close a session.
- To view session information:

```
# iscsiadm -m session [-P <printlevel>]
```

  - The print levels are `1`, `2`, and `3`. Each shows more detail.

After targets have been discovered by the initiator, the initiator needs to log in to access the LUNs. Logging in establishes a session, which is a TCP connection between an initiator node port and a target node port. Before logging in, no sessions are active. Example:

```
# iscsiadm -m session
iscsiadm: No active sessions
```

Use the following command to log in and establish a session. When a session is established, iSCSI control, data, and status messages are communicated over the session.

```
# iscsiadm -m node -l
```

To log in to a specific target:

```
# iscsiadm -m node --targetname iqn.2011-
12.com.example.mypc:tgt1 -p 192.0.2.1:3260 -l
Login to [iface: default, target: iqn.2011-
12.com.example.mypc:tgt1, portal: 192.0.2.1:3260] successful.
```

To log off from all targets:

```
# iscsiadm -m node -u
```

To log off from a specific target, use the same login command as shown, except use `-u`.

Running the `-m session` command after logging in shows an active session:

```
# iscsiadm -m session
tcp: [1] 192.0.2.1:3260,1 iqn.2011-12.com.example.mypc:tgt1
```

The TCP session includes a session ID (SID) of `1` in this example. Include `–P` *`<printlevel>`* for more detail. Valid `<printlevel>` values are `1`, `2`, and `3`.

The following example shows additional detail on the session:

```
# iscsiadm -m session –P 1
Target: iqn.2011-12.com.example.mypc:tgt1
     Current Portal: 192.0.2.1:3260,1
     Persistent Portal: 192.0.2.1:3260,1
           **********
           Interface:
           **********
           Iface Name: default
           Iface Transport: tcp
           Iface Initiatorname: iqn.1988-12.com.oracle:392a7cee2f
           Iface IPaddress: 192.0.2.101
           Iface HWaddress: <empty>
           Iface Netdev: <empty>
           SID: 1
           iSCSI Connection State: LOGGED IN
           iSCSI Session State: LOGGED IN
           Internal iscsid Session State: NO CHANGE
```

The following example provides the most detail, including information on the attached SCSI devices. In this example, the target defines two LUNs:

```
# iscsiadm -m session –P 3
Target: iqn.2011-12.com.example.mypc:tgt1
...
           ************************
           Attached SCSI devices:
           ************************
           Host Number: 4 State: running
           scsi10 Channel 00 Id 0 Lun:0
           scsi10 Channel 00 Id 0 Lun:1
                Attached scsi disk sda         State: running
           scsi10 Channel 00 Id 0 Lun:2
                Attached scsi disk sdb         State: running
```

iSCSI Block Devices

- After establishing a session, the LUNs are represented as SCSI (sd) block devices in the /dev directory.

```
# fdisk -l | grep /dev/sd
Disk /dev/sda: 10.7 GB, 10737418240 bytes
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
```

- View iSCSI initialization messages in the /var/log/messages file.
- Use partition and file system utilities on the LUNs as if they were locally attached SCSI disks.
- Mount the file systems with the _netdev option.
- This mount option requires the network to be up before mounting the file system.

After establishing a session, the LUNs are represented as block devices in the /dev directory. In an Oracle VM Server for x86 environment, disk devices appear as virtual block devices (xvd). This example also shows LVM volumes for the root and swap partitions:

```
# fdisk -l | grep /dev
Disk /dev/xvda: 12.9 GB, 12884901888 bytes
/dev/xvda1   *      1      64     512000    83   Linux
/dev/xvda2         64    1567   12069888    8e   Linux LVM
Disk /dev/xvdb: 10.7 GB, 10737418240 bytes
Disk /dev/xvdd: 10.7 GB, 10737418240 bytes
Disk /dev/mapper/vg_host01-lv-root: 9202 MB, 9202302976 bytes
Disk /dev/mapper/vg_host01-lv-swap: 3154 MB, 3154116608 bytes
```

After establishing a session, the iSCSI LUNs appear as SCSI (sd) devices:

```
# fdisk -l | grep /dev/sd
Disk /dev/sda: 10.7 GB, 10737418240 bytes
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
```

You can view the /var/log/messages file for initialization messages:

```
# tail -f /var/log/messages
<date> host03 kernel: scsi12 : iSCSI Initiator over TCP/IP
...
<date> host03 kernel: sd 4:0:0:1: [sda] Attached SCSI disk
...
<date> host03 kernel: sd 4:0:0:2: [sdb] Attached SCSI disk
...
```

You can now partition and create a file system on the LUNs by using utilities such as fdisk and mkfs:

```
# fdisk /dev/sda
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-10240, default 1): ENTER
Last cylinder, +cylinders or +size{K,M,G} (1-10240, default
10240): +5120
Command (m for help): w
...
# mkfs -t ext4 /dev/sda1
...
```

Make a mount point and mount the file system:

```
# mkdir /iscsi_lun1
# mount /dev/sda1 -o _netdev /iscsi_lun1
# df -h
Filesystem     Size   Used   Avail  Use%   Mounted on
...
/dev/sda1      5.0G   138M    4.6G    3%   /iscsi_lun1
```

When creating entries for iSCSI LUNs in the /etc/fstab file, use the _netdev option. This indicates that the file system resides on a device that requires network access. This option is used to prevent the system from attempting to mount the file system until the network has been enabled:

```
# vi /etc/fstab
/dev/sda1    /iscsi_lun1     ext4    _netdev   0   0
```

Quiz

Which of the following statements are true about iSCSI?

a. Storage devices are attached to servers (targets), and client systems (initiators) access the remote storage devices over IP networks.

b. iSCSI uses the existing IP infrastructure and does not require any additional cabling.

c. An Oracle Linux system can be configured both as an iSCSI target and an iSCSI initiator.

d. The configuration file for defining targets and LUNs is `/etc/tgt/targets.conf`.

Quiz

Which of the following commands allow you to configure an iSCSI server (target)?

a. `tgtadm`

b. `tgt-admin`

c. `tgt-setup-lun`

d. `iscsiadm`

Quiz

Which of the following statements are true?

a. Use the iscsiadm command to initiate discovery, which is the process that makes the targets known to an initiator.

b. Use the iscsiadm command from the initiator to establish a session, which is a TCP connection between an initiator node port and a target node port.

c. Mount the file systems created on iSCSI block devices with the _netdev option.

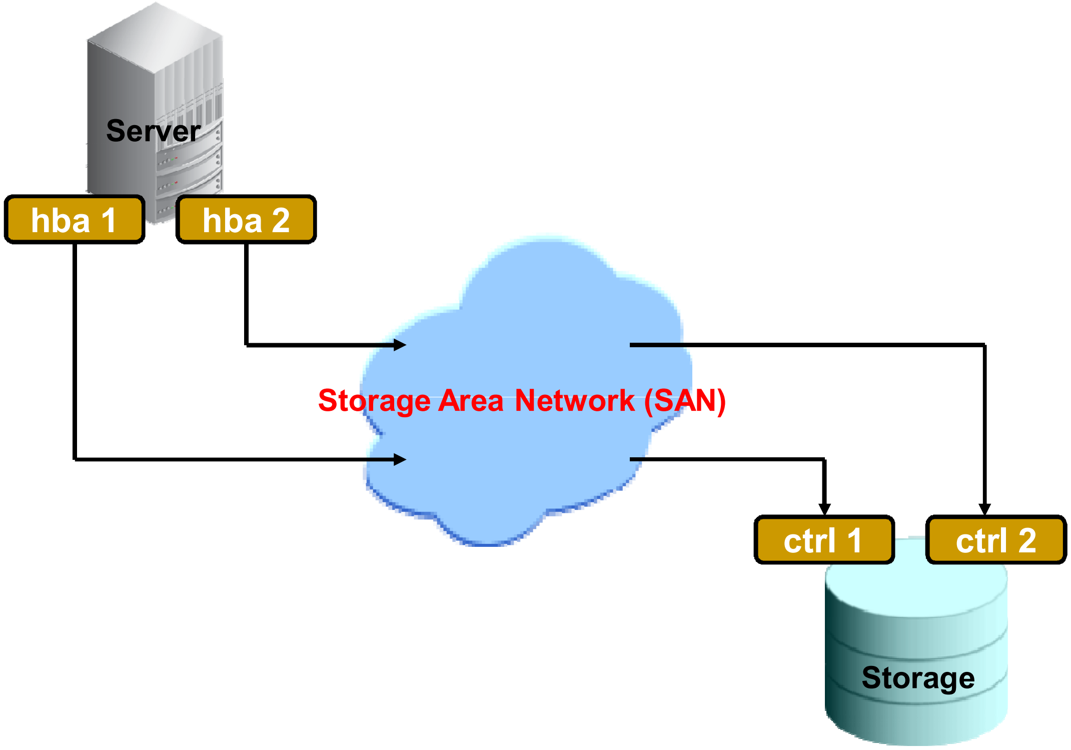
d. Mount the file systems created on iSCSI block devices with the errors=continue option.

Complete iSCSI Practices

- You can optionally stop the lecture portion of this lesson and complete the iSCSI practices in the activity guide.
- The iSCSI practices include the following:
  - Practice 10-1: Exploring a Configured iSCSI Server
  - Practice 10-2: Modifying the iSCSI Server (Target) Configuration
  - Practice 10-3: Configuring an iSCSI Client (Initiator)
- After completing the iSCSI practices, you can continue with the lecture.

Because this is a long lesson, you can optionally stop the lecture here and perform the iSCSI practices. You can then resume the lecture and complete the remaining multipathing practice after reaching the end of this lesson.

Device Mapper Multipathing



Device-Mapper Multipath (DM-Multipath) is a native Linux multipath tool that allows you to configure multiple I/O paths between a server and a storage device into a single path. Multiple paths to storage devices provide redundancy and failover capabilities, as well as improved performance and load balancing.

DM-Multipath can be configured in either of the following configurations:

- **Active/Passive (or Standby) –** Only half the paths are used for I/O. If any component in the active path fails, DM-Multipath switches I/O to the alternate path.
- **Active/Active –** In an active/active configuration, DM-Multipath can be configured to spread the I/O load across all paths in a round-robin fashion, or can dynamically balance the load.

The slide shows a simple DM-Multipath configuration through a storage area network (SAN). There are two I/O paths in this example:

- Host bus adapter (`hba1`), through the SAN, to Controller (`ctrl1`)
- Host bus adapter (`hba2`), through the SAN, to Controller (`ctrl2`)

Without DM-Multipath, each I/O path is a separate device even though the path connects the same server to the same storage device. DM-Multipath creates a single multipath device on top of the underlying devices.

DM-Multipath Files

- Install the `device-mapper-multipath` software package:

```
# yum install device-mapper-multipath
```

- Several files and directories are installed, including:
  - `/sbin/multipath` – Utility that detects and configures multiple paths to devices
  - `/sbin/multipathd` – DM-Multipath daemon
  - `/sbin/mpathconf` – Utility for configuring DM-Multipath
- Man pages exist for the `multipathd` daemon, the `mpathconf` and `multipath` utilities, and the `multipath.conf` configuration file.

To enable the DM-Multipath services on your system, install the `device-mapper-multipath` software package:

```
# yum install device-mapper-multipath
```

When installing the `device-mapper-multipath` package, `yum` also installs `device-mapper-multipath-libs` as a dependency package. The packages install several files, including the following:

- `/sbin/multipath` – Device mapper target auto-configurator that is used to detect and configure multiple paths to devices
- `/sbin/multipathd` – Multipath daemon that checks for failed paths, and reconfigures the multipath map to regain connectivity. This daemon executes `/sbin/multipath` when events occur.
- `/sbin/mpathconf` – Utility for configuring `device-mapper-multipath`. It creates or modifies `multipath.conf` and is also used to display the current status.

Man pages are also installed for `multipath`, `multipathd`, `multipath.conf`, and `mpathconf`.

DM-Multipath Configuration File

- The DM-Multipath configuration file, `/etc/multipath.conf`, contains the following sections:
  - `defaults` – Default settings that can be overwritten by the `devices` and `multipaths` sections
  - `blacklist` – Devices to be excluded from the multipath topology discovery
  - `blacklist_exceptions` – Blacklisted devices to be included in the multipath topology discovery
  - `multipaths` – Settings for individual multipath devices. Devices are identified by the `wwid` keyword.
  - `devices` – Settings for individual storage controller types. Controller types are identified by `vendor`, `product`, and `revision` keywords, which must match the `sysfs` information.
- Priority is `multipaths`, `devices`, and `defaults`.

The main configuration file for DM-Multipath is `/etc/multipath.conf`. This file is not created by the initial installation of the RPM package. However, the following sample files are installed in the `/usr/share/doc/device-mapper-multipath-<`*`version`*`>` directory:

- `multipath.conf` – Basic configuration file with some examples for DM-Multipath. This file is used to create the `/etc/multipath.conf` file.
- `multipath.conf.defaults` – A complete list of the default configuration values for the storage arrays supported by DM-Multipath
- `multipath.conf.annotated` – A list of configuration options with descriptions

The configuration file contains entries that use the following form:

```
<section> {
     <attribute> <value>
     <subsection> {
          <attribute> <value>
          }
}
```

Section Keywords

Each `<section>` contains one or more attributes or subsections. Valid sections are:

- `defaults` – Defines the default settings for DM-Multipath. These settings can be overwritten by the `devices` and `multipaths` sections.
- `blacklist` – Defines the devices to be excluded from the multipath topology discovery. Devices that are blacklisted are not grouped into a multipath device.
- `blacklist_exceptions` – Defines the devices to be included in the multipath topology discovery, even if the devices are listed in the `blacklist` section
- `multipaths` – Defines settings for individual multipath devices. Devices are identified by the `wwid` keyword.
- `devices` – Defines settings for individual storage controller types. Controller types are identified by `vendor`, `product`, and `revision` keywords, which must match the `sysfs` information about the device.

There is a priority for sections in the configuration file; `multipaths` has a higher priority than `devices`, which has a higher priority than `defaults`. When determining the attributes of a multipath device, multipath settings are read first, then device settings, and finally the default settings.

`defaults` Attributes in `/etc/multipath.conf`

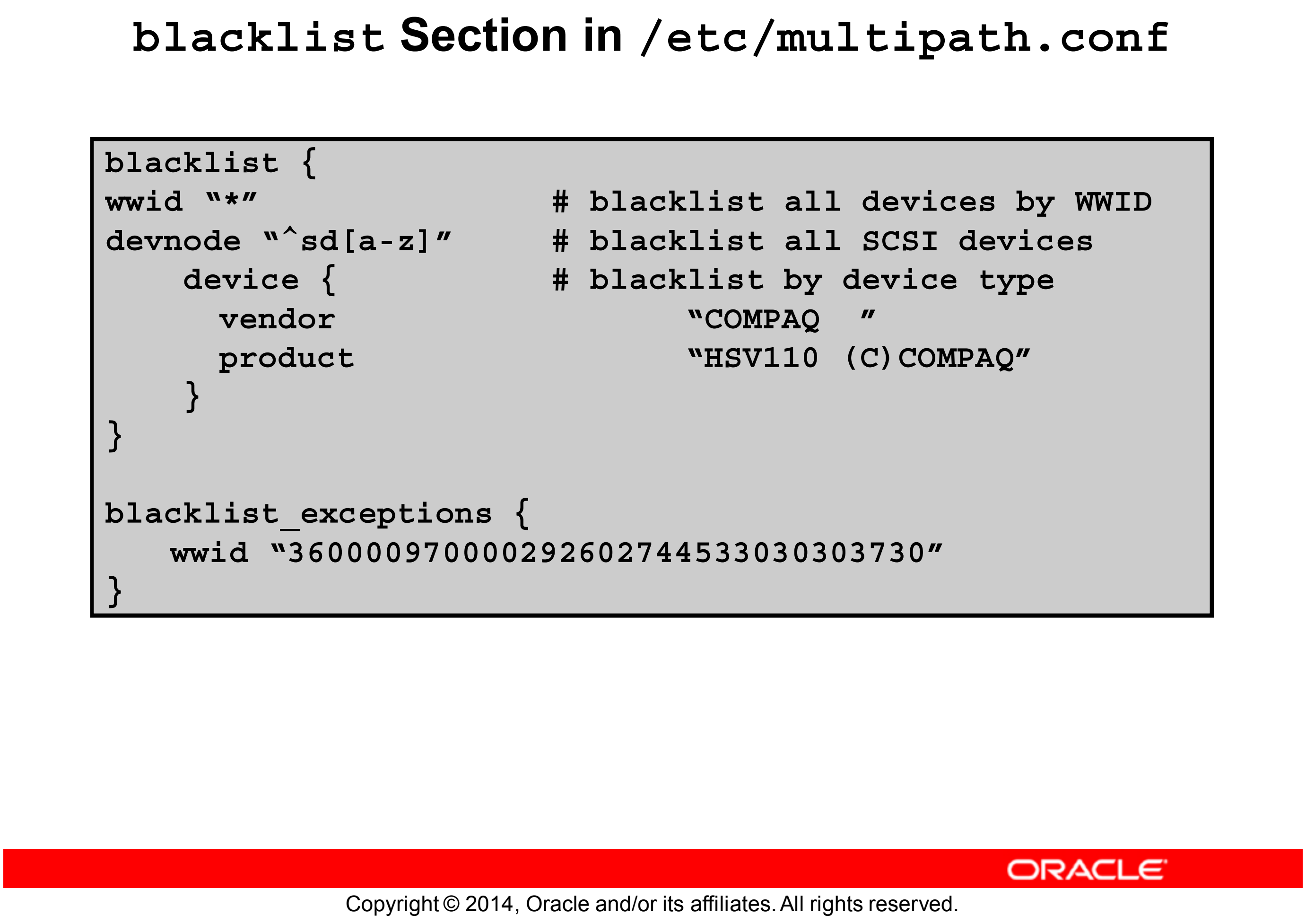
```
defaults {
  udev_dir                 /dev
  polling_interval         10
  path_selector            “round-robin 0”
  path_grouping_policy     multibus
  getuid_callout           “/lib/udev/scsi_id --
   whitelisted --device=/dev/%n”
  prio                     alua
  path_checker             readsector0
  rr_min_io                100
  max_fds                  8192
  rr_weight                priorities
  failback                 immediate
  no_path_retry            fail
  user_friendly_names      yes
}
```

A partial list of attributes defined in the `defaults` section of the configuration file is as follows:

- `udev_dir` – Directory where `udev` creates device nodes. The default is `/dev`.
- `polling_interval` – Interval in seconds that paths are checked. The default is 5 seconds.
- `path_selector` – One of the following path selector algorithms to use:
  - `round-robin 0`: Loop through every path sending the same amount of I/O to each. This is the default.
  - `queue-length 0`: Send I/O down a path with the least amount of outstanding I/O.
  - `service-time 0`: Send I/O down a path based on the amount of outstanding I/O and relative throughput.
- `path_grouping_policy` – Paths are grouped into path groups. The policy determines how path groups are formed. There are five different policies.
  - `failover`: One path per priority group
  - `multibus`: All paths in one priority group. This is the default.
  - `group_by_serial`: One priority group per storage controller (serial number)
  - `group_by_prio`: One priority group per priority value
  - `group_by_node_name`: One priority group per target node name

- getuid_callout – Command and arguments to get a unique path identifier. The default is /lib/udev/scsi_id --whitelisted --device=/dev/%n. The scsi_id command queries a SCSI device and generates a value that is unique across all SCSI devices. The --whitelisted option must be included to generate output.
- prio – One of the following methods used to obtain a path priority value:
    - const – Set a priority of one to all paths. This is the default.
    - emc – Generate the path priority for EMC storage arrays.
    - alua – Generate the path priority based on the SCSI-3 Asymmetric Logical Unit Access (ALUA) settings. ALUA allows a device to report the state of its ports to hosts. This state is used by hosts to prioritize paths and make failover and load-balancing decisions.
    - tpg_pref – Generate the path priority based on the SCSI-3 ALUA settings, using the preferred port bit.
    - ontap – Generate the path priority for NetApp storage arrays.
    - rdac – Generate the path priority for LSI/Engenio/NetApp E-Series Redundant Disk Array Controller (RDAC).
    - hp_sw – Generate the path priority for Compaq/HP controller in Active/Standby mode.
    - hds – Generate the path priority for Hitachi HDS Compaq/HP controller in active/standby mode.
- path_checker – One of the following methods used to determine the paths' state:
    - readsector0 – Read the first sector of the device. This is the default.
    - tur – Issue a Test Unit Ready (TUR) command to the device.
    - emc_clarrion – Query the EMC CLARiiON-specific EVPD page 0xC0 to determine the path state.
    - hp_sw – Check the path state for HP storage arrays with the Active/Standby firmware.
    - rdac – Check the path state for the LSI/Engenio/NetApp E-Series RDAC.
    - directio – Read the first sector with direct I/O.
- rr_min_io – The number of I/O to route to a path before switching to the next path in the same path group. This is for systems running kernels older than 2.6.31. Newer systems use rr_min_io_rq. The default is 1000.
- max_fds – The maximum number of file descriptors that can be opened by multipath and multipathd
- rr_weight – The path weight. Possible values are priorities or uniform.
- failback – One of the following methods to manage path group failback:
    - immediate – Fail back immediately to the highest priority path group that contains active paths.
    - manual – Do not perform automatic failback.
    - followover – Perform automatic failback only when the first path of a path group becomes active.
    - values > 0 – This indicates the time to defer failback in seconds.

Refer to the multipath.conf man page for additional attributes and details.



blacklist Section in /etc/multipath.conf

```
blacklist {
wwid “*”                 # blacklist all devices by WWID
devnode “^sd[a-z]”       # blacklist all SCSI devices
    device {             # blacklist by device type
      vendor                    “COMPAQ  ”
      product                   “HSV110 (C)COMPAQ”
    }
}

blacklist_exceptions {
   wwid “3600009700002926027445330303037 30”
}
```

Use the blacklist section in the /etc/multipath.conf file to exclude devices from being grouped into a multipath device. You can blacklist devices using any of the following identifiers. Use the same identifiers in the blacklist_exceptions section.

- WWID
- Device Name: Use the devnode keyword.
- Device Type: Use the device subsection.

The following example uses all three identifiers to blacklist devices:

```
blacklist {
     wwid 360000970000292602744533030303730
     devnode "^sd[a-z]"    # blacklist all SCSI devices
     device {
          vendor                   "COMPAQ  "
          product                  "HSV110 (C)COMPAQ"
     }
}
```

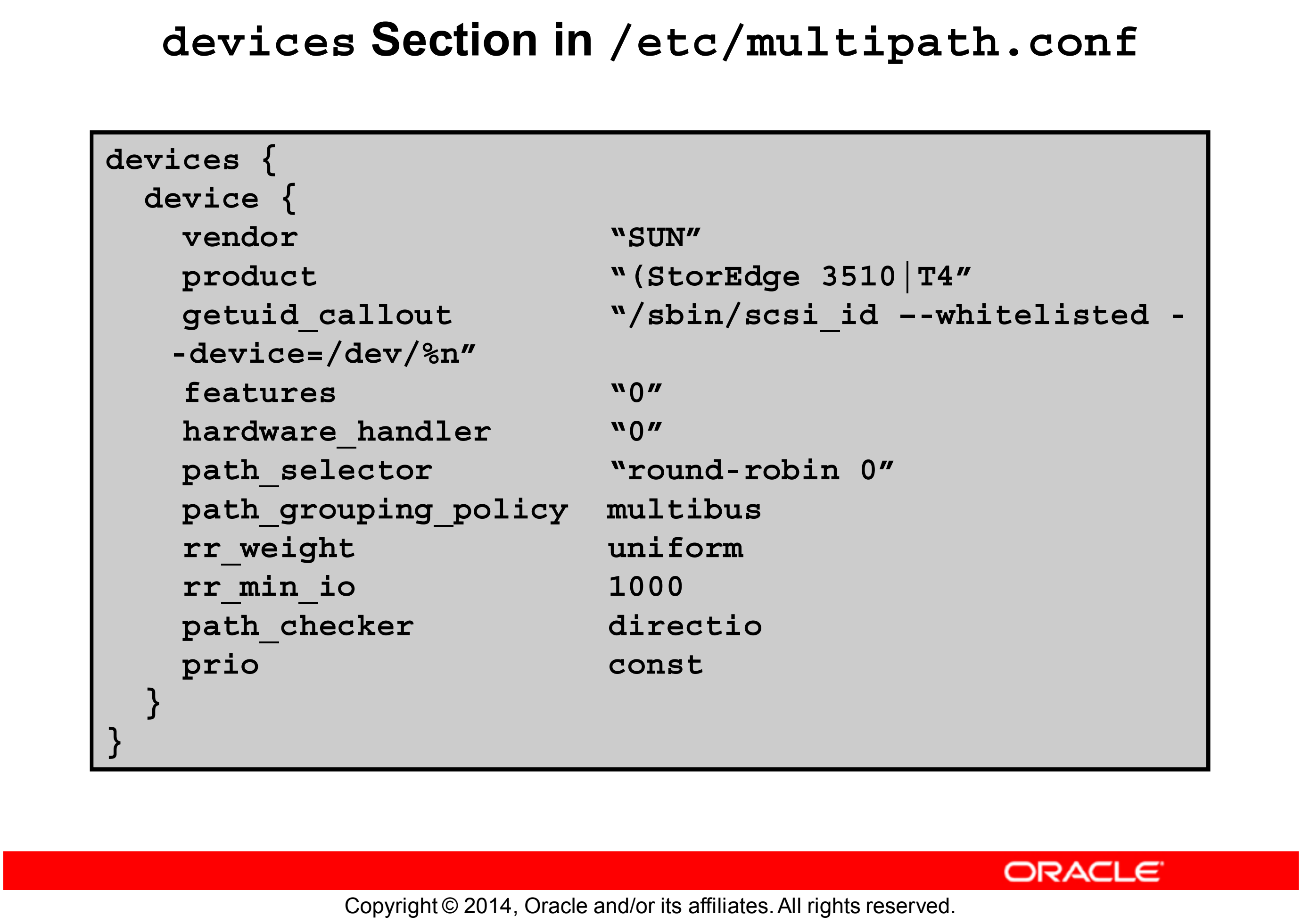
`multipaths` Section in `/etc/multipath.conf`

```
multipaths {
  multipath {
    wwid    3600508b4000156d700012000000b0000
    alias                   yellow
    failback                manual
    no_path_retry           5
  }
  multipath {
    wwid    360000970000292602744533032443941
    alias   green
  }
}
```

Set attributes in the `multipaths` section of the configuration file for each individual multipath device. These attributes apply to a specified multipath and override the attributes set in the `defaults` and `devices` sections.

This slide shows the settings that override the `failback` and `no_path_retry` default settings for the first WWID and set aliases for both WWIDs. Valid values for the `no_path_retry` attribute are:

- `<n>` – The number of retries until multipath stops the queueing and fails the path
- `fail` – Specifies immediate failure (no queueing)
- `queue` – Never stop queueing (queue forever until the path comes alive)



devices Section in /etc/multipath.conf

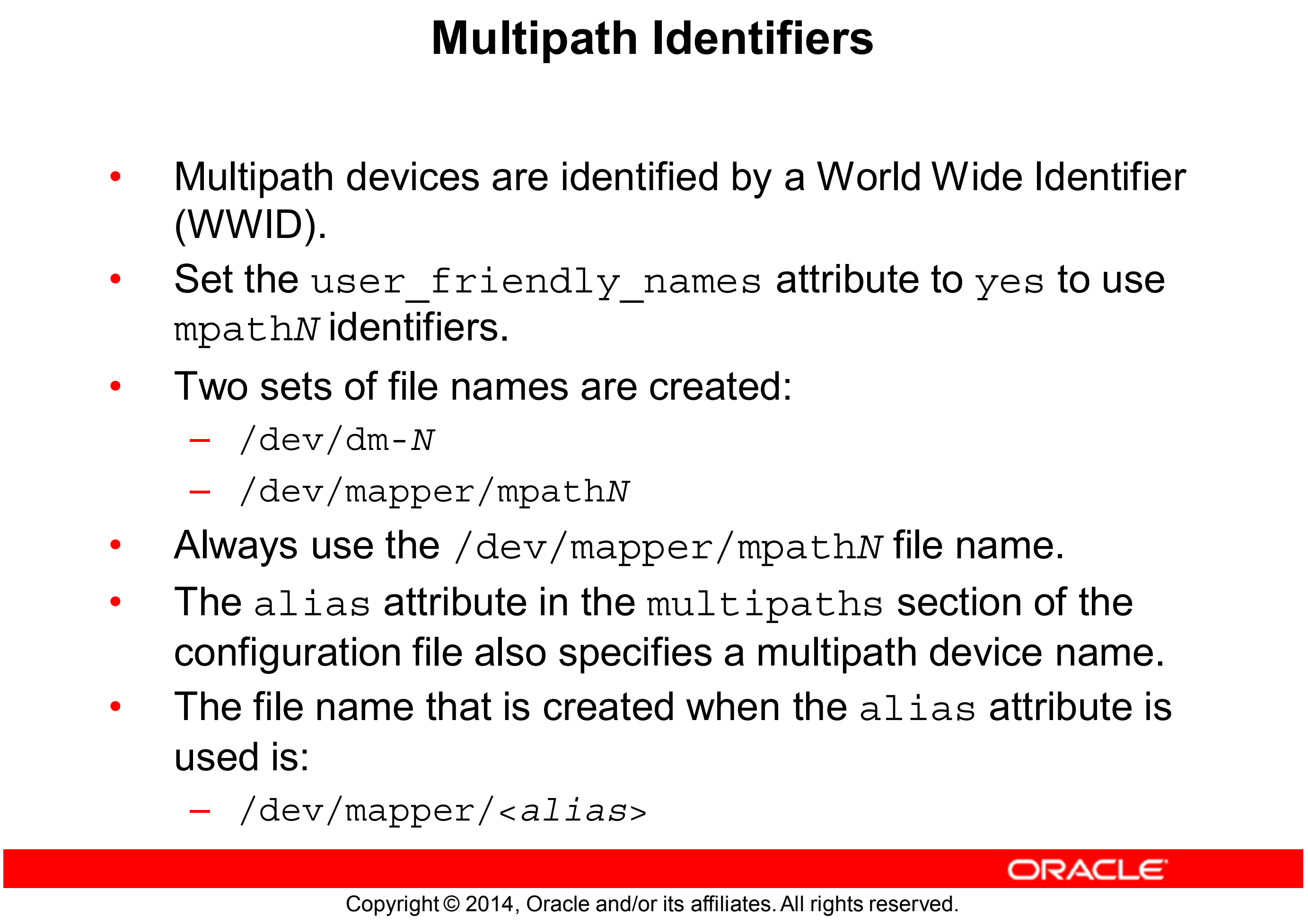
```
devices {
  device {
    vendor                  “SUN”
    product                 “(StorEdge 3510|T4”
    getuid_callout          “/sbin/scsi_id --whitelisted -
   -device=/dev/%n”
    features                “0”
    hardware_handler        “0”
    path_selector           “round-robin 0”
    path_grouping_policy    multibus
    rr_weight               uniform
    rr_min_io               1000
    path_checker            directio
    prio                    const
  }
}
```

DM-Multipath includes support for the most common storage arrays. These supported storage arrays and their default configuration values are stored in the `/usr/share/doc/device-mapper-multipath-<version>/multipath.conf.defaults` file. You can copy and paste from this file into the `/etc/multipath.conf` file to add a storage device.

To add a storage device that is not supported by default, obtain the `vendor`, `product`, and `revision` information from the `sysfs` file system for the storage device and add this to the `/etc/multipath.conf` file. View the following files to obtain this information:

- `/sys/block/`*`device_name`*`/device/vendor` – Vendor information
- `/sys/block/`*`device_name`*`/device/model` – Product information
- `/sys/block/`*`device_name`*`/device/rev` – Revision information

Include additional device attributes as required. The slide shows sample device settings for the Sun StorEdge 3510|T4 storage arrays.



Multipath Identifiers

- Multipath devices are identified by a World Wide Identifier (WWID).
- Set the `user_friendly_names` attribute to `yes` to use `mpathN` identifiers.
- Two sets of file names are created:
  - `/dev/dm-N`
  - `/dev/mapper/mpathN`
- Always use the `/dev/mapper/mpathN` file name.
- The `alias` attribute in the `multipaths` section of the configuration file also specifies a multipath device name.
- The file name that is created when the `alias` attribute is used is:
  - `/dev/mapper/<alias>`

By default, multipath devices are identified by a World Wide Identifier (WWID), which is globally unique. As an alternative, you can set the `user_friendly_names` attribute to `yes` in the `/etc/multipath.conf` file, which sets the multipath device to `mpathN` where *N* is the multipath group number.

The DM-Multipath tool uses two different sets of file names:

- `/dev/dm-N` – Never use these device names, because they are intended to be used only by the DM-Multipath tool.
- `/dev/mapper/mpathN` – Always use these device names to access the multipath devices. These names are persistent and are automatically created by device-mapper early in the boot process.

You can also use the `alias` attribute in the `multipaths` section of the configuration file to specify the name of a multipath device. The `alias` attribute overrides `mpathN` names.

Use either the `/dev/mapper/mpathN` name or the `/dev/mapper/<alias>` name when creating a partition, when creating an LVM physical volume, and when making and mounting a file system.

`mpathconf` Utility

- Use the `mpathconf` utility to read or create the DM-Multipath configuration file, or to display status.
- Options for the `mpathconf` utility include:
  - `--enable` – Removes any line that blacklists all device nodes from the configuration file
  - `--disable` – Adds a line that blacklists all device nodes to the configuration file
- Use the following command to display usage:

```
# mpathconf --help
```

- Run `mpathconf` without any arguments to display the status of DM-Multipath:

```
# mpathconf
multipath is enabled...
```

Use the `mpathconf` utility to configure DM-Multipath. The `mpathconf` utility creates or modifies the `/etc/multipath.conf` file, using a copy of the sample `multipath.conf` file in the `/usr/share/doc/device-mapper-multipath-<version>` directory as a template if necessary.

Run `mpathconf --help` to display usage:

```
# mpathconf --help
usage: /sbin/mpathconf <command>
Commands:
Enable: --enable
Disable: --disable
Set user_friendly_names (Default n): --user_friendly_names <y|n>
Set find_multipaths (Default n): --find_multipaths <y|n>
Load the dm-multipath modules on enable (Default y): --with ...
start/stop/reload multipathd (Default n): --with_multipathd ...
chkconfig on/off multipathd (Default y): --with_chkconfig <y|n>
```

By default, the mpathconf utility loads the device-mapper-multipath module and sets chkconfig to start the multipathd service automatically on reboot.

Run mpathconf without any arguments to display the status of DM-Multipath.

```
# mpathconf
multipath is disabled
find_multipaths is disabled
user_friendly_names is disabled
dm_multipath modules is not loaded
multipathd is chkconfiged off
```

For a basic failover configuration with all the defaults, run:

```
# mpathconf --enable
```

To enable the multipath configuration and start the multipathd service, run:

```
# mpathconf –-enable --with_multipathd y
```

The remaining commands are:

- --user_friendly_names <y|n> – If set to y, this adds the line user_friendly_names yes to the /etc/multipath.conf defaults section. If set to n, this removes the line.
- --find_multipaths <y|n> – If set to y, this adds the line find_multipaths yes to the /etc/multipath.conf defaults section. If set to n, this removes the line. Refer to the man page for multipath.conf for a description of find_multipaths.
- --with_module <y|n> – If set to y, this runs modprobe dm_multipath to install the multipath modules. This only works with the --enable command.
- --with_chkconfig <y|n> – If set to y, this runs chkconfig multipathd on to start multipathd on --enable , and runs chkconfig multipathd off on --disable.

Always reload the multipathd service after making any changes to the /etc/multipath.conf configuration file.

```
# service multipathd start|restart|reload
```

`multipath` Utility

- Use the `multipath` utility to list and configure multiple paths to devices.
- To list the maximum multipath topology:

```
# multipath -ll
mpatha(1IET_00030001) dm-0 IET,VIRTUAL-DISK
size=10g features=‘0’ hwhandler=‘0’ wp=rw
|-+- policy=‘round-robin 0’ prio=1 status=active
| `- 5:0:0:1 sda 8:0     active ready running
`-+- policy=‘round-robin 0’ prio=1 status=active
  `- 5:0:0:2 sdb 8:2     active ready running
```

The `multipath` utility is the device mapper target auto-configurator, which is used to detect and configure multiple paths to devices. Use the following command to display usage:

```
# multipath -h
```

Some of the available options are described as follows:

- `-v` *`<verbosity>`* – Specify the verbosity level when displaying paths and multipaths.
- `-l` – List the multipath topology.
- `-ll` – List the maximum multipath topology information.
- `-f` – Flush a multipath device map. Use `-F` to flush all multipath device maps.
- `-c` – Check if a device should be a path in a multipath device.
- `-p failover | multibus | group_by_serial | group_by_prio |group_by_node_name` – Force all maps to the specified path grouping policy.
- `-r` – Force device map reload.

You can optionally specify a device name to update only the device map that contains the specified device. Use the `/dev/sd#` format, or the `major:minor` format, or the multipath map name (for example, `mpath`*`N`*), or the WWID to specify a device.

A sample output of the `multipath -ll` command is as follows:

```
# multipath -ll
mpatha(1IET_00030001) dm-0 IET,VIRTUAL-DISK
size=10g features=‘0’ hwhandler=‘0’ wp=rw
|-+- policy=‘round-robin 0’ prio=1 status=active
| `- 5:0:0:1 sda 8:0     active ready running
`-+- policy=‘round-robin 0’ prio=1 status=active
  `- 5:0:0:2 sdb 8:2     active ready running
```

The output is described as follows:

- `mpatha` – User-defined alias name
- `1IET_00030001` – Unique identifier returned by the `scsi_id` utility
- `dm-0` – `sysfs` file name
- `IET` – Vendor name
- `VIRTUAL-DISK` – Product name

- `size=10g` – Size of the DM device
- `features=‘0’` – DM features supported
- `hwhandler=‘0’` – Hardware handler
- `wp=rw` – Write permission, set to read-write

- `policy=‘round-robin 0’` – Path selector algorithm
- `prio=1` – Path group priority
- `status=active` – Path group state

- `5:0:0:1, 5:0:0:2` – SCSI information: host, channel, scsi_id, and LUN
- `sda`, `sdb` – Linux device name
- `8:0`, `8:2` – Major and minor numbers
- `active ready running` – DM path and physical path state

`multipathd` Daemon

- The `multipathd` daemon checks for failed paths and reconfigures the multipath map.
- The daemon runs `multipath` when events occur.
- Options include:
  - `-d` – Run the daemon in the foreground and display all messages.
  - `-v <level>` – Set the verbosity level.
  - `-k` – Enter interactive mode.
- Use the `help` command from interactive mode to list the available commands.
- Press Ctrl + D or enter `quit` or `exit` to exit interactive mode.

The `multipathd` daemon checks for failed paths and reconfigures the multipath map. The daemon runs the `multipath` utility when events occur that require a device map reconfiguration. You can run the daemon in the foreground, which displays all messages using the `-d` option. You can set the verbosity level using the `-v <level>` option. The `multipathd` daemon also has an interactive mode enabled with the `-k` option. Some of the commands available in interactive mode include:

- `help` – List the available interactive commands.
- `list|show paths` – Show the paths that `multipathd` is monitoring and their state.
- `list|show maps|multipaths` – Show the multipath devices that `multipathd` is monitoring.
- `list|show topology` – Show the multipath topology. This gives the same output as using the `multipath -ll` command.
- `list|show config` – Show the current configuration that is derived from `/etc/multipath.conf`.
- `reconfigure` – Reconfigure the multipaths. This is triggered automatically after any hotplug event.
- `quit|exit` – End the interactive session.

To run the daemon in the foreground (the following example shows that the daemon is running, stops the daemon, and then starts the daemon in the foreground):

```
# multipathd -d
Nov 09 18:55:04 | process is already running
# service multipathd stop
Stopping multipathd daemon:                   [  OK  ]
# multipathd -d
Nov 09 18:56:37 | mpatha: load table [0 20971520 multipath 0 0 2
1 round-robin 0 1 1 8:0 1 round-robin 0 1 1 8:16 1]
Nov 09 18:56:37 | mpatha: event checker started
Nov 09 18:56:37 | path checkers start up
```

The following example runs the daemon in interactive mode:

```
# multipathd -k
multipathd> help
list|show paths
list|show paths format $format
...
multipathd> list paths
hcil    dev  dev_t  pri dm_st  chk_st  dev_st  next_check
#:#:#:# xvda 202:0  1   undef  ready   running orphan
#:#:#:# xvdb 202:16 1   undef  ready   running orphan
5:0:0:1 sda  8:0    1   active ready   running XXXXXXXXXX 20/20
5:0:0:2 sdb  8:0    1   active ready   running XXXXXXXXX. 19/20
multipathd> list maps
name   sysfs uuid
mpatha dm-0  1IET_00030001
multipathd> list devices
available block devices:
  ram0 devnode blacklisted, unmonitored
  ...
  xvda devnode whitelisted, monitored
  xvdb devnode whitelisted, monitored
  sr0  devnode blacklisted, unmonitored
  sda  devnode whitelisted, monitored
  dm-0 devnode blacklisted, unmonitored
  sdb  devnode whitelisted, monitored
multipathd> quit
```

iSCSI Multipathing

- From the initiator, enable DM-Multipath:

```
# mpathconf --enable
```

- Obtain a unique identifier for the device (`/dev/sda`):

```
# scsi_id --whitelisted --replace-whitespace --
   device=/dev/sda
1IET_00030001
```

- Create `defaults` and `multipath` entries in the `/etc/multipath.conf` file.
- Start the `multipathd` daemon:

```
# service multipathd start
```

- The DM-Multipath now exists: `/dev/mapper/mpatha`.

The procedure to configure DM-Multipath from an iSCSI initiator to an iSCSI target is presented, which assumes the following:

- The iSCSI target package is installed on the server.
- Targets and LUNs are configured on the iSCSI server.
- The iSCSI initiator package is installed on the client.
- The DM-Multipath package is installed on the client.
- The targets have been discovered by the client.
- An iSCSI session is active between the target and the initiator.
- The initiator has redundant network connections to the target.

Before enabling DM-Multipath on the client, there is no `/etc/multipath.conf` configuration file:

```
# ls /etc/multipath.conf
```

You can manually create this file or enable DM-Multipath with the `mpathconf` utility:

```
# mpathconf --enable
```

This command copies `multipath.conf` from the `/usr/share/doc/device-mapper-multipath-<`*`version`*`>` directory to the `/etc` directory.

Run the following command to enable DM-Multipath. Notice that the configuration file now exists:

```
# mpathconf --enable
# ls /etc/multipath.conf
/etc/multipath.conf
```

To configure multipath for /dev/sda, use the scsi_id command to obtain a unique identifier for the device. The --replace-whitespace option replaces all whitespace in the output with underscores.

```
# scsi_id --whitelisted --replace-whitespace --device=/dev/sda
1IET_00030001
```

Configure /etc/multipath.conf as follows:

```
# vi /etc/multipath.conf
defaults {
     user_friendly_names     yes
     getuid_callout          "/lib/udev/scsi_id --whitelisted --
replace-whitespace --device=/dev/%n"
}
multipaths {
     multipath {
          wwid 1IET_00030001
     }
}
```

This configuration performs the following:

- Enables the user_friendly_names option, which creates mpath*N* device names
- Defines the scsi_id command used to obtain a unique identifier for the /dev device
- Creates one multipath for wwid 1IET_00030001

1IET_00030001 is the expected output of the scsi_id command for /dev/sda.

Before starting the multipathd daemon, there are no mpath*N* devices in /dev/mapper:

```
# ls /dev/mapper/mpath*
ls: cannot access /dev/mapper/mpath*: No such file or directory
```

After starting the multipathd daemon, the mpath*N* device is created in /dev/mapper:

```
# service multipathd start
# ls /dev/mapper/mpath*
/dev/mapper/mpatha
```

Use the /dev/mapper/mpatha name when creating a partition, when creating an LVM physical volume, and when making and mounting a file system.

If one of the network interfaces fails on the initiator, I/O continues through the remaining active interface.

Quiz

Which of the following statements are true?

a. Device-Mapper Multipath (DM-Multipath) is a native Linux multipath tool that enables you to configure multiple I/O paths between a server and a storage device into a single logical path.

b. The main configuration file for DM-Multipath is `/etc/multipath.conf`.

c. Sections in the DM-Multipath configuration file include defaults, blacklist, blacklist_exceptions, multipaths, and devices.

d. In the DM-Multipath configuration file, the priority is `defaults`, `devices`, and `multipaths`.

ORACLE

Quiz

Which of the following statements are true?

a. Use the `mpathconf --enable` command to enable DM-Multipath.

b. Set the `user_friendly_names` attribute to `yes` to create `/dev/mapper/mpathN` file names to multipath devices.

c. Use the `multipath` utility to list multiple paths to devices.

d. Use the `scsi_id` command to obtain a unique identifier for a block device.

Summary

In this lesson, you should have learned how to:

- Configure an iSCSI target
- Use the `tgtadm`, `tgt-admin`, and `tgt-setup-lun` utilities
- Configure an iSCSI software initiator
- Use the `iscsiadm` utility
- Describe Device Mapper Multipathing
- Use the `mpathconf` and `multipath` utilities
- Configure iSCSI multipathing

Practice 10: Overview

These practices cover the following topics:

- Exploring a pre-configured iSCSI target on dom0
- Modifying iSCSI target configuration on dom0
- Configuring a VM guest as an iSCSI initiator
- Configuring iSCSI multipathing