



Oracle International College Academy Use Only

# Oracle Linux 7: Advanced Administration

Student Guide - Volume II  
D90758GC10  
Edition 1.0 | September 2015 | D92964

Learn more from Oracle University at [oracle.com/education/](http://oracle.com/education/)

**Author**

Craig McBride

**Technical Contributors  
and Reviewers**

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

**Editors**

Malavika Jinka  
Aju Kumar

**Graphic Editors**

Kavya Bellur  
Maheshwari Krishnamurthy  
Seema Bopaiah

**Publishers**

Veena Narasimhan  
Pavithran Adka  
Raghunath M

**Copyright © 2015, Oracle and/or its affiliates. All rights reserved.**

**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

## 1 Introduction

- Course Goals 1-2
- Schedule 1-4
- Objectives 1-6
- Virtualization with Oracle VM Server for x86 1-7
- Oracle VM Server for x86 in the Classroom 1-8
- Working with Classroom Virtual Machines 1-9
- Classroom System Configuration 1-11
- Local Yum Repository 1-13
- Summary 1-14
- Practice 1: Overview 1-15

## 2 Network Addressing and Name Services

- Objectives 2-2
- Introduction to DHCP 2-3
- Configuring a DHCP Server 2-4
- Additional DHCP Server Declarations 2-6
- Starting and Stopping a DHCP Server 2-8
- Specifying Command-Line Arguments 2-9
- Configuring a DHCP Client 2-11
- Introduction to DNS 2-13
- Nameserver Types 2-14
- BIND 2-15
- Starting a DNS Cache-Only Nameserver 2-16
- Configuring an Authoritative Nameserver 2-17
- Zone Files 2-18
- The /etc/named.conf File 2-20
- The /etc/named.rfc1912.zones File 2-22
- Reverse Name Resolution 2-23
- rndc Utility 2-25
- host and dig Utilities 2-27
- Quiz 2-28
- Summary 2-29
- Practice 2: Overview 2-30

### **3 Authentication and Directory Services**

- Objectives 3-2
- Authentication Options 3-3
- Authentication Configuration GUI 3-4
- NIS Authentication 3-6
- Lightweight Directory Access Protocol (LDAP) 3-7
- OpenLDAP 3-9
- OpenLDAP Server Directories 3-10
- OpenLDAP Server Utilities 3-11
- OpenLDAP Client Utilities 3-12
- OpenLDAP Server Configuration 3-13
- The ldapmodify Utility 3-14
- The slappasswd Utility 3-15
- Loading the Standard Schemas 3-16
- Populating an OpenLDAP Directory 3-17
- Using the migrationtools Utilities 3-18
- Configuring LDAP Authentication 3-20
- Configuring User Authentication from an OpenLDAP Client 3-22
- Configuring Winbind Authentication 3-24
- Winbind Security Model Options 3-26
- Configuring Kerberos Authentication 3-28
- IPA Identity Management and Authentication Services 3-29
- Configuring Advanced Options 3-30
- Configuring Password Options 3-32
- System Security Services Daemon 3-34
- Configuring SSSD Services 3-35
- Configuring SSSD Domains 3-37
- Quiz 3-39
- Summary 3-40
- Practice 3: Overview 3-41

### **4 Pluggable Authentication Modules (PAM)**

- Objectives 4-2
- Introduction to PAM 4-3
- PAM Configuration Files 4-4
- PAM Authentication Modules 4-5
- PAM Module Types 4-6
- PAM Control Flags 4-7
- PAM: Example #1 4-9
- PAM: Example #2 4-11

Quiz 4-13  
Introduction to SELinux 4-14  
Summary 4-16  
Practice 4: Overview 4-17

## 5 Web and Email Services

Objectives 5-2  
Apache HTTP Server 5-3  
Configuring Apache 5-4  
Testing Apache 5-6  
Apache Containers 5-7  
Apache Virtual Hosts 5-9  
Quiz 5-11  
Email Program Classifications 5-12  
Email Protocols 5-13  
Postfix SMTP Server 5-15  
Sendmail SMTP Server 5-16  
Configuring Sendmail on a Client 5-18  
Quiz 5-19  
Summary 5-20  
Practice 5: Overview 5-21

## 6 Installing Oracle Linux by Using Kickstart

Objectives 6-2  
Kickstart Installation Method 6-3  
Kickstart File 6-4  
Verifying the Kickstart File 6-6  
Beginning a Kickstart Installation 6-7  
Rescue Mode 6-8  
Quiz 6-9  
Summary 6-10  
Practice 6: Overview 6-11

## 7 Samba Services

Objectives 7-2  
Introduction to Samba 7-3  
Samba Daemons and Services 7-4  
Samba Server Configuration 7-5  
Samba Server 7-7  
Samba Server Types 7-8  
Accessing Linux Shares from Windows 7-10

Accessing Windows Shares from Linux 7-12  
Samba Utilities 7-13  
Quiz 7-15  
Summary 7-16  
Practice 7: Overview 7-17

## 8 Advanced Software Package Management

Objectives 8-2  
Software Management with RPM and Yum 8-3  
RPM Packages 8-5  
The Binary RPM Build Process 8-6  
BUILD Directory Structure 8-7  
spec File to Build a Binary RPM Package 8-8  
spec File: Example 8-10  
Managing RPM-Based Software with Yum 8-11  
Yum Cache 8-12  
Yum History 8-14  
Extending Yum Functionality with Plug-Ins 8-16  
Popular Yum Plug-Ins 8-18  
Managing Errata 8-19  
Important Resources for Errata Information 8-21  
PackageKit Software Package Manager GUI 8-22  
Using PackageKit Software Update 8-23  
PackageKit Commands: Summary 8-24  
Quiz 8-25  
Summary 8-27  
Practice 8: Overview 8-28

## 9 Advanced Storage Administration

Objectives 9-2  
Access Control Lists (ACLs) 9-3  
getfacl and setfacl Utilities 9-4  
Disk Quotas 9-6  
Enabling Disk Quotas 9-7  
Summary of Quota Commands 9-9  
Encrypted Block Devices 9-12  
cryptsetup Utility 9-13  
Making an Encrypted Device Usable 9-15  
kpartx Utility 9-16  
Udev: Introduction 9-18  
Udev Rule Files and Directories 9-19

Sample Udev Rules	9-20
udevadm Utility	9-22
Creating a Symbolic Link to a Device Node	9-24
Quiz	9-25
Summary	9-27
Practice 9: Overview	9-28

## 10 Advanced Networking

Objectives	10-2
Network Bonding: Configuration	10-4
Using the NetworkManager GUI to Configure Network Bonding	10-6
Network Bonding Modes	10-7
Network Bonding Link Monitoring	10-9
Using the nmcli Utility to Configure Network Bonding	10-10
Using the nmcli Utility to Add the Slaves to the Bond	10-12
Activate the Bond	10-14
Viewing Network Bonding Information	10-15
Virtual Local Area Networks: Introduction	10-17
Using the NetworkManager GUI to Configure 802.1Q VLAN Tagging	10-18
Using the nmcli Utility to Configure VLAN Tagging	10-19
Viewing VLAN Information	10-21
Virtual Private Networks: Introduction	10-22
The libreswan RPM Package	10-23
Site-to-Site VPN	10-24
Site-to-Site VPN: Configuration	10-25
Example “sitetosite” Connection	10-27
Quiz	10-28
Summary	10-29
Practice 10: Overview	10-30

## 11 OCFS2 and Oracle Clusterware

Objectives	11-2
OCFS2: Introduction	11-3
OCFS2 Features	11-5
Using OCFS2	11-7
Preparing for OCFS2	11-8
OCFS2 Software	11-10
Kernel Configuration	11-12
Configuring Cluster Layout	11-13
o2cb Utility	11-15
OCFS2 Heartbeat	11-17

O2CB Cluster Timeouts 11-19  
o2cb Initialization Script 11-20  
mkfs.ocfs2 Utility 11-22  
Mounting OCFS2 Volumes 11-25  
OCFS2 Tuning and Debugging 11-27  
Quiz 11-30  
Oracle Clusterware: Introduction 11-34  
Oracle Clusterware: Hardware Requirements 11-35  
Oracle Clusterware Files 11-36  
Summary 11-37  
Practice 11: Overview 11-38

## **12 iSCSI and Multipathing**

Objectives 12-2  
Introduction to iSCSI 12-3  
Configuring an iSCSI Server 12-4  
targetcli Utility 12-5  
Backstores 12-6  
Creating an iSCSI Target 12-8  
Creating iSCSI LUNs 12-10  
Creating ACLs 12-11  
iSCSI Initiators 12-12  
Configuring an iSCSI Initiator 12-13  
iscsiadm Utility 12-14  
iSCSI Discovery 12-15  
iSCSI Initiator Sessions 12-17  
iSCSI Block Devices 12-19  
Quiz 12-21  
Device Mapper Multipathing 12-24  
DM-Multipath Files 12-25  
DM-Multipath Configuration File 12-26  
defaults Attributes in /etc/multipath.conf 12-27  
blacklist Section in /etc/multipath.conf 12-29  
multipaths Section in /etc/multipath.conf 12-30  
devices Section in /etc/multipath.conf 12-31  
Multipath Identifiers 12-32  
mpathconf Utility 12-33  
multipath Utility 12-35  
multipathd Daemon 12-37  
iSCSI Multipathing 12-39  
Quiz 12-41

Summary 12-43  
Practice 12: Overview 12-44

## 13 Control Groups (Cgroups)

Objectives 13-2  
Control Groups: Introduction 13-3  
cgroup Subsystems (Resource Controllers) 13-4  
View Mounted Resource Controllers 13-5  
cgroup Subsystem Parameters 13-6  
cgroup Implementation in Oracle Linux 7 13-7  
cgroup Hierarchies 13-8  
The /sys/fs/cgroup/systemd Directory 13-10  
systemd Slice Units 13-12  
systemd Scope Units 13-14  
The systemd-cgls Utility 13-15  
Displaying the cgroup Tree of Specific Services and Scopes 13-16  
Viewing cgroup Resource Control Settings 13-18  
Controlling Access to System Resources 13-20  
Modifying Unit Configuration Files 13-22  
The systemd-run Utility 13-23  
Quiz 13-25  
Summary 13-28  
Practice 13: Overview 13-29

## 14 Virtualization with Linux

Objectives 14-2  
Virtualization: Introduction 14-3  
Virtualization Concepts 14-4  
Virtualization Modes 14-5  
Linux and Xen Integration 14-7  
Running Linux in a Virtual Machine 14-8  
Oracle VM Server for X86 14-9  
Oracle VM Server for x86 Components 14-10  
Linux as a Guest OS with Oracle VM Server for X86 14-12  
Linux as a Guest OS with Oracle VM VirtualBox 14-13  
VMware vSphere 14-14  
Linux as a Guest OS with VMware vSphere 14-16  
Microsoft Hyper-V and Windows Azure 14-18  
Linux as a Guest OS with Microsoft Hyper-V and Windows Azure 14-20  
Linux as a Virtualization Provider 14-22  
libvirt 14-23

Installing KVM and libvirt 14-24  
Getting Started with virt-manager: Connections 14-26  
Virtual Networks 14-28  
Working with Storage 14-30  
Creating Virtual Machines 14-32  
Managing the Life Cycle of a Virtual Machine 14-33  
Quiz 14-34  
Summary 14-35  
Practice 14: Overview 14-36

## 15 Linux Containers (LXC)

Objectives 15-2  
Linux Containers: Introduction 15-3  
Linux Container Resource Isolation 15-4  
Linux Container Configuration File 15-6  
Setting up a System for Linux Containers 15-8  
Linux Container Template Scripts 15-9  
lxc-create Utility 15-10  
lxc-oracle Template Options 15-12  
Using the lxc-oracle Template 15-13  
Starting and Stopping a Container 15-15  
lxc-start Utility 15-16  
lxc-execute Utility 15-17  
lxc-attach Utility 15-18  
lxc-ls and lxc-info Utilities 15-19  
lxc-console Utility 15-20  
lxc-freeze and lxc-unfreeze Utilities 15-21  
lxc-cgroup Utility 15-22  
Summary of Linux Container Utilities 15-23  
Quiz 15-25  
Summary 15-27  
Practice 15: Overview 15-28

## 16 Docker

Objectives 16-2  
Docker: Introduction 16-3  
Docker Images 16-4  
The Docker Hub Registry 16-5  
The Oracle Linux Repository 16-6  
Installing and Starting Docker 16-7  
The docker Utility 16-9

Using Btrfs as the Storage Engine	16-10
Searching the Docker Hub Registry for Images	16-12
Downloading Images from Docker Hub	16-13
Running an Application Inside a Container	16-15
Running an Interactive Docker Container	16-17
Listing Containers and Viewing Container Logs	16-18
Display All Information for a Container or an Image	16-19
Creating a New Container	16-21
Starting, Stopping, and Removing a Container	16-22
Running Additional Commands in a Running Container	16-23
Creating an Image from a Container	16-24
Creating an Image from a Dockerfile	16-25
Save and Load an Image or a Container	16-26
Oracle WebLogic Docker Certification	16-27
Quiz	16-28
Summary	16-30
Practice 16: Overview	16-31

## 17 Security Enhanced Linux (SELinux)

Objectives	17-2
Introduction to SELinux	17-3
SELinux Packages	17-4
SELinux Administration GUI	17-6
SELinux Modes	17-7
Setting a Mode	17-8
SELinux Policies	17-9
SELinux Booleans	17-11
getsebool and setsebool Utilities	17-12
SELinux File Labeling	17-14
SELinux Context	17-15
Changing the Context File Type	17-17
Confined SELinux Users	17-19
SELinux Utilities: Summary	17-21
Quiz	17-23
Summary	17-24
Practice 17: Overview	17-25

## 18 Core Dump Analysis

Objectives	18-2
System Core Collection: Kexec and Kdump	18-3
Kdump Configuration File	18-5

Kdump Setup Configuration GUI	18-7
Kernel Tuning Parameters	18-9
Magic SysRq Keys	18-11
crash Utility	18-12
Downloading kernel-debuginfo RPM Packages	18-13
Initial crash Output	18-15
Using the crash Utility	18-16
Symbolic Display crash Commands	18-17
System State crash Commands	18-19
Utility crash Commands	18-25
Session Control crash Commands	18-26
General Guidelines for Using crash	18-27
Quiz	18-28
Summary	18-31
Practice 18: Overview	18-32

## 19 Dynamic Tracing with DTrace

Objectives	19-2
DTrace: Introduction	19-3
Reasons to Use DTrace on Linux	19-4
DTrace 0.4 in UEK R3	19-6
DTrace-Enabled Applications	19-8
ULN Channels for DTrace 0.4	19-9
Enabling DTrace	19-10
DTrace Probes	19-12
DTrace Providers	19-13
dtrace Provider	19-15
profile Provider	19-16
syscall Provider	19-17
sdt Provider	19-18
proc Provider	19-19
sched Provider	19-20
io Provider	19-21
Enabling Probes	19-22
DTrace Actions	19-24
Built-in D Variables	19-26
trace() Built-in Function	19-27
DTrace: Examples	19-28
D Scripts	19-32
Using Predicates in a D Script	19-34
D Script: Example	19-36

Quiz 19-40

Summary 19-44

Practice 19: Overview 19-45

## A Appendix: NIS Configuration

Objectives A-2

NIS Authentication A-3

NIS Maps A-4

NIS Server Configuration A-6

NIS Client Configuration A-8

Implementing NIS Authentication A-9

Quiz A-11

Summary A-12

Practice A: Overview A-13



## OCFS2 and Oracle Clusterware

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# 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



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## 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 nonclustered 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 nonclustered 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 interact with it 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 who use the Unbreakable Enterprise Kernel.

Both on-disk and network protocol compatibility is 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 and 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](https://blogs.oracle.com/wim/entry/dm_nfs)):

```
# mount |grep ocfs2
ocfs2_dlmfs on /dlm type ocfs2_dlmfs (rw,relatime)
/dev/mapper/ovspoolfs on
/poolfsmnt/0004fb000005000031bfabb596af47 type ocfs2
(rw,_netdev,heartbeat=global)
/dev/mapper/SATA_ST3500320AS_5QM1EYTX on
/OVS/Repositories/0004fb000003000a826cf5f901b13de 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](mailto: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



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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

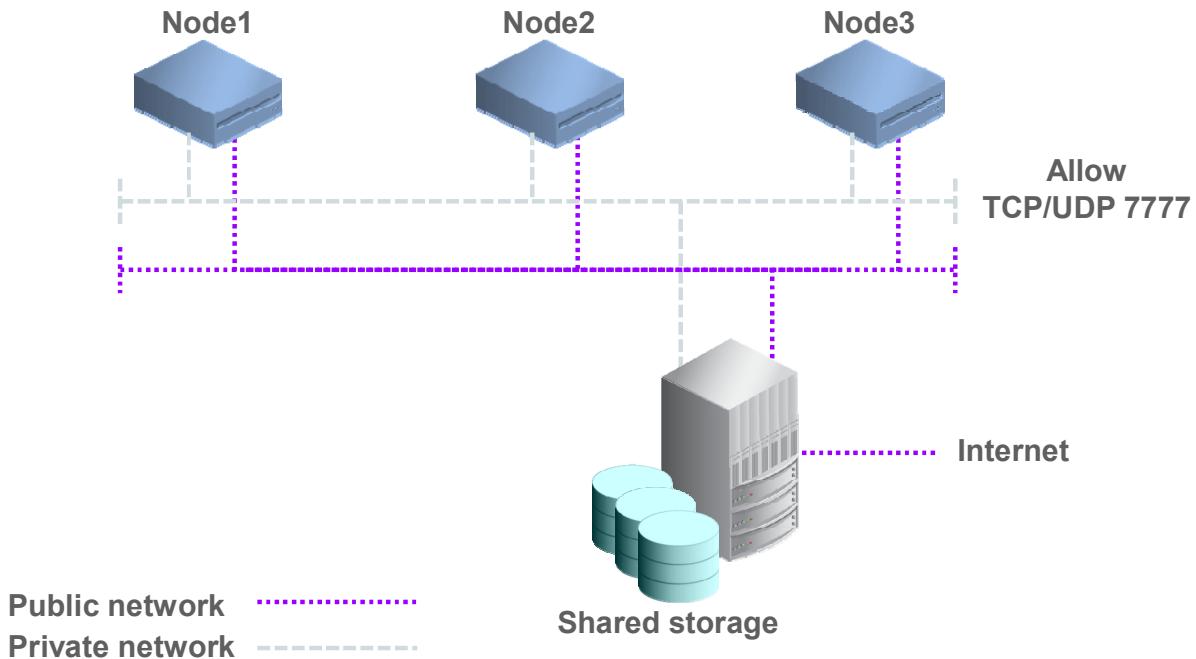
1. Configure shared storage.
2. Configure a private network for nodes (recommended).
3. Configure the firewall.
4. Install the Unbreakable Enterprise Kernel (UEK).
5. Install the `ocfs2-tools` RPM package.
6. Configure the `panic_on_oops` and `panic` kernel settings.
7. Configure the cluster layout configuration.
  - 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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

# Preparing for OCFS2



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 xvdb /ovs/running_pool/host*/vm.cfg
host01/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,xvdb,w!', 
host02/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,xvdb,w!', 
host03/vm.cfg: 'file:/OVS/sharedDisk/physDisk1.img,xvdb,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 `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 `ip addr` command displays the network configuration for the host01 node, with `eth0` on the public network and `eth1` on the private network:

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue ...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet addr:127.0.0.1/8 scope host lo
    ...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
    link/ether 00:16:3e:00:01:01 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.101/24 brd 192.0.2.255 scope global eth0
    ...
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc ...
    link/ether 00:16:3e:00:02:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.200/24 brd 192.168.1.255 scope global eth1
    ...
```

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 `firewalld` 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 trust this port number, or disable `firewalld`.

## 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 command to install the user-space RPM:

```
# yum install ocfs2-tools
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 `ocfs2-tools` packages.

```
# yum install ocfs2-tools
```

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

```
# rpm -q1 ocfs2-tools
/etc/init.d/o2cb
/etc/init.d/ocfs2
/etc/sysconfig/o2cb
/sbin/debugfs.ocfs2
/sbin/fsck.ocfs2
```

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

```
/sbin/mkfs.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/lib/systemd/system/o2cb.service  
/usr/lib/systemd/system/ocfs2.service  
/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 3.8.13-55.1.6.el7uek.x86\_64) are:

```
# ls -R /lib/modules/3.8.13-55.1.6.el7uek.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
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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, nonclustered 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 to 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]`



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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>` for 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`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 = 5678675678ABCFE309888C34A0DB6B2
    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](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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 load the modules:

```
# service o2cb load
```

- Use `o2cb` to configure the cluster stack and cluster timeout values:

```
# service o2cb configure
```

- Use `o2cb` to start the cluster stack, named `mycluster` in this example:

```
# service o2cb online mycluster
```

- Use `o2cb` to view the status and settings of the cluster stack:

```
# service o2cb status
```

- Use `systemctl` to enable the `o2cb` service:

```
# systemctl enable o2cb
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `/etc/init.d/o2cb` initialization script to load the modules, to configure the cluster stack and cluster timeout values, to start the cluster, and to view the status and settings of the cluster. Run the initialization script commands on each node of the cluster. The following example uses the `service` command, which calls the `/etc/init.d/o2cb` initialization script, to load the modules:

```
# service o2cb load
Loading filesystem "configfs": OK
Loading stack plugin "o2cb": OK
Loading filesystem "ocfs2_dlmfs": OK
Creating directory '/dlm': OK
Mounting ocfs2_dlmfs filesystem at /dlm: OK
```

The `configfs` file system 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.

The following command allows you to configure the cluster stack and the cluster timeout values:

```
# 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.

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

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

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

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

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.

Use the following command to start the cluster, named mycluster in this example:

```
# service o2cb online mycluster
```

```
Loading filesystem "configfs": OK
```

```
Loading stack plugin "o2cb": OK
```

```
Loading filesystem "ocfs2_dlmfs": OK
```

```
Mounting ocfs2_dlmfs filesystem at /dlm: OK
```

```
Setting cluster stack "o2cb": OK
```

```
Registering O2CB cluster "mycluster": OK
```

```
Setting O2CB cluster timeouts : OK
```

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

```
# service o2cb status
```

```
...
```

```
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
```

```
Please run 'systemctl enable o2cb.service' to enable o2cb
```

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
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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, and 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
  - `--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`.
  - `--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: 10736369664 (2621184 clusters) (2621184 blocks)
Cluster groups: 82 (tail covers 8448 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`



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 mycluster
```

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:

```
# systemctl enable o2cb
# systemctl enable ocfs2
```

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. Stop the `firewalld` service or trust port 7777 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`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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%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: ...
  Number of blocks: 2621184
  Block size: 4096
  Number of clusters: 2621184
  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          ...                                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 nonclustered environment.
- d. OCFS2 file systems can be mounted and used across multiple architectures at the same time.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Quiz



Which of the following are valid OCFS2 commands?

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



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Oracle Clusterware: Introduction

- 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).



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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). GI comprises Oracle Clusterware and Oracle Automatic Storage Management (ASM). Oracle Clusterware and Oracle ASM can only co-exist. 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 11: Overview

The practices cover the following topics:

- Preparing your environment for OCFS2 configuration
- Verifying that the required software is installed
- Configuring the cluster layout
- Configuring and starting the O2CB cluster stack
- Creating an OCFS2 volume
- Mounting an OCFS2 volume
- Tuning and debugging OCFS2



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

These practices assume that you completed Practices 2-1 and 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 11-1.

If you did complete Practices 2-1 and 2-2, and `eth1` on **host01** has an IP address, skip task 1 in Practice 11-1 and go directly to task 2.

## iSCSI and Multipathing

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

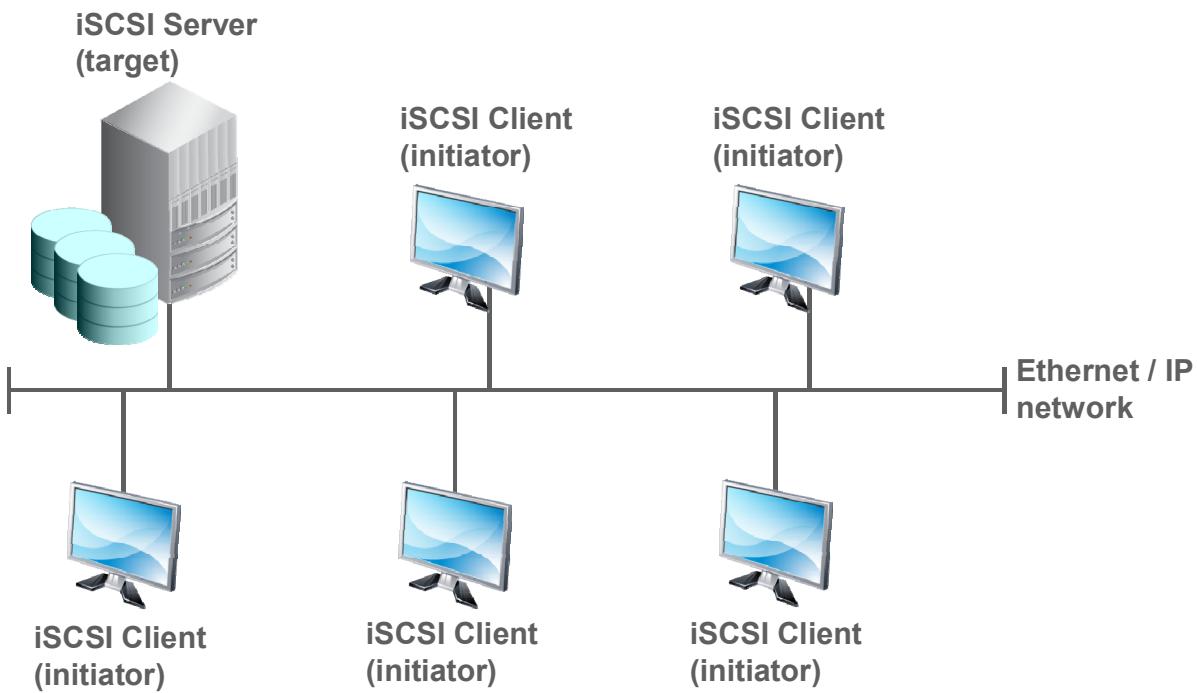
After completing this lesson, you should be able to:

- Configure an iSCSI server
- Use the `targetcli` administration shell
- Configure an iSCSI software initiator
- Use the `iscsiadm` utility
- Describe Device Mapper Multipathing
- Use the `mpathconf` and `multipath` utilities
- Configure iSCSI multipathing



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Introduction to iSCSI



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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). 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.

# Configuring an iSCSI Server

- Oracle Linux 7 uses the Linux-IO (LIO) kernel target subsystem for iSCSI.
  - LIO supports a number of storage fabrics including FCoE, iSER, and SRP.
  - All storage fabrics are managed with the `targetcli` utility.
- To configure an iSCSI server, install the `targetcli` software package:

```
# yum install targetcli
```

- Use the `systemctl` command to enable and start the `target` service:

```
# systemctl enable target
# systemctl start target
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Linux 7 uses the Linux-IO (LIO) kernel target subsystem for iSCSI. In addition to iSCSI, LIO supports a number of storage fabrics including Fibre Channel over Ethernet (FCoE), iSCSI access over Mellanox InfiniBand networks (iSER), and SCSI access over Mellanox InfiniBand networks (SRP). In Oracle Linux 7, all storage fabrics are managed with the `targetcli` utility.

To configure an Oracle Linux system as an iSCSI server, begin by installing the `targetcli` software package:

```
# yum install targetcli
```

Installing the `targetcli` software package also installs the `python-rtslib` package, which provides the `/usr/lib/systemd/system/target.service` file.

Before using the `targetcli` utility to create, delete, and view storage targets, use the `systemctl` command to enable and start the `target` service on the iSCSI server.

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

## targetcli Utility

- Use the `targetcli` utility to configure the iSCSI target.
- Run `targetcli` to enter an administration shell.

```
# targetcli  
/>
```

- From the `targetcli` shell prompt, use the `help` command to view the available commands.
- Commands exist to:
  - List the current configuration (`ls`)
  - Traverse the storage object hierarchy (`cd`)
  - Create storage objects, targets, LUNs, ACLs (`create`)
  - Change or view global parameters (`set`, `get`)
  - Save the configuration and exit the `targetcli` shell (`exit`)
- Commands can be arguments to `targetcli`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `targetcli` utility is the administration shell for creating, editing, and viewing the configuration of the kernel's target subsystem. Run `targetcli` to enter the configuration shell.

```
# targetcli  
targetcli shell version 2.1.fb37  
Copyright 2011-2013 by Datera, Inc and others.  
For help on commands, type 'help'.  
/>>
```

Run the `help` command from the `targetcli` prompt to view the available commands.

```
/> help
```

The following are some of the available `targetcli` commands:

- `ls`: View the object hierarchy.
- `cd`: Traverse the object hierarchy.
- `create`: Create storage objects, targets, LUNs, network portals, access control lists.
- `exit`: Exit the `targetcli` shell and automatically save the configuration.

You can also enter `targetcli <command>` to run `<command>` without entering the shell.

## Backstores

- Backstores are local storage resources that the kernel target uses to “back” the SCSI devices it exports.
- The following types of backstores are supported:
  - `block`: Linux block devices
  - `fileio`: File on a mounted file system
  - `pscsi`: Pass-through SCSI devices
  - `ramdisk`: Memory copy RAM disks
- To create a `block` backstore from the `targetcli` shell:

```
/> cd /backstores/block
/backstores/block> create name=LUN_1 dev=/dev/xvdb
```

- To create a `fileio` backstore from the `targetcli` shell:

```
/> cd /backstores/fileio
/backstores/fileio> create name=LUN_3 /root/disk1.img 5G
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Backstores are the different kinds of local storage resources that the kernel target uses to “back” the SCSI devices it exports to client systems. The mappings to local storage resources that each backstore creates are called storage objects.

Use the `targetcli ls` command to list the different types of backstores.

```
# targetcli ls /backstores
o- backstores ..... [ ... ]
  | o- block ..... [Storage Objects: 0]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
```

The types of backstores are described as follows:

- `block`: Linux block devices such as `/dev/sda`
- `fileio`: Any file on a mounted file system such as `/tmp/disk1.img`
- `pscsi`: Any storage object that supports pass-through SCSI commands
- `ramdisk`: Memory copy RAM disks

Access the `targetcli` shell to create backstores.

```
# targetcli
/ >
```

### Create a block backstore

The following commands create block backstores. Use the `cd` command to change to the `/backstore/block` directory.

```
/ > cd /backstore/block
/backstore/block
```

Use the `create` command to create two block storage objects.

```
/backstores/block> create name=LUN_1 dev=/dev/xvdb
/backstores/block> create name=LUN_2 dev=/dev/xvde
```

### Create a fileio backstore

The following commands create a fileio backstore. From the `targetcli` shell, use the `cd` command to change to the `/fileio` directory.

```
/ > cd /backstore/fileio
/backstore/fileio
```

Use the `create` command to create a fileio storage object.

```
/backstores/fileio> create name=LUN_3 /root/disk1.img 5G
```

### List the backstores

Use the `ls` command to view the backstores.

```
/ > ls /backstores
o- backstores ..... [....]
  o- block ..... [Storage Objects: 2]
    | o- LUN_1 ..... [/dev/xvdb (10.0GiB) write-thru deactivated]
    | o- LUN_2 ..... [/dev/xvde (10.0GiB) write-thru deactivated]
  o- fileio ..... [Storage Objects: 1]
    | o- LUN_3 ... [/root/disk1.img (5.0GiB) write-back deactivated]
  o- pscsi ..... [Storage Objects: 0]
  o- ramdisk ..... [Storage Objects: 0]
```

Note that the block storage objects have write-thru deactivated and the fileio storage objects have write-back deactivated by default. The fileio storage objects can support either write-back or write-thru operation. Write-back enables the local filesystem cache, which improves performance but also increases the risk of data loss. Specify “`write-back`” when creating the fileio storage object to enable write-back operation. For example:

```
/backstores/fileio> create name=<name> <file> <size> write_back
```

See the following for more information about backstores: <http://linux-iscsi.org/wiki/LIO>

# Creating an iSCSI Target

- To create an iSCSI target from the `targetcli` shell:

```
/> cd /iscsi
/iscsi> create iqn.2015-07.com.example.host02:tgt1
Created target iqn.2015-07.com.example.host02:tgt1.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0),
port 3260.
```

- This command creates an iSCSI target named `tgt1` and uses the iSCSI Qualified Name (IQN) identifier.
- This command also creates a Target Portal Group (TPG).
  - A TPG allows iSCSI to support multiple complete configurations within a single target.
  - A default network portal of `0.0.0.0:3260` is also created if the `auto_add_default_portal` global parameter is `true`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To create an iSCSI target from the `targetcli` shell, use the `cd` command to change to the `/iscsi` directory.

```
/> cd /iscsi
/iscsi>
```

Use the `create` command without any arguments to create an iSCSI target by using a default target name. By default, 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 following example uses the `create` command to create an IQN (iSCSI Qualified Name) called `iqn.2015-07.com.example.host02` with a target named `tgt1`.

```
/iscsi> create iqn.2015-07.com.example.host02:tgt1
```

A Target Portal Group (TPG) is created as a result of this command. Use the `ls` command to view the TPG hierarchy.

```
/iscsi> ls
o- iscsi ..... [Targets: 1]
  o- iqn.2015-07.com.example.host02:tgt1 ..... [TPGs: 1]
    o- tpg1 ..... [no-gen-acls, no-auth]
      o- acls ..... [ACLs: 0]
      o- luns ..... [LUNs: 0]
      o- portals ..... [Portals: 1]
        o- 0.0.0.0:3260 ..... [OK]
```

Note that the initial TPG hierarchy is empty with exception of a network portal.

A network portal is an IP address:port pair. An iSCSI target is accessed by remote systems through the network portal. The default portal of `0.0.0.0:3260` allows the iSCSI server to listen on all IPv4 addresses on port 3260. You can delete the default portal and configure portals as needed. Both IPv4 and IPv6 addresses are supported.

To allow remote systems to access an iSCSI target on port 3260, either disable the `firewalld` service on the iSCSI server or configure `firewalld` to trust the `3260/tcp` port. The following example uses `firewall-cmd` to open the `3260/tcp` port for the `firewalld` service.

```
# firewall-cmd --permanent --add-port=3260/tcp
```

If you include the `--permanent` option when adding a port, use the `firewall-cmd` command to reload the configuration.

```
# firewall-cmd -reload
```

## Creating iSCSI LUNs

- Logical Unit Numbers (LUNs) link previously defined storage objects with a target.
  - The kernel target exports these SCSI Logical Units to remote systems.
- From the `targetcli` shell, use the `cd` command to change to the `luns` directory within the `<target/TGP>` hierarchy.

```
/> cd iqn.2015-07.com.example.host02:tgt1/tpg1/luns
/iscsi/iqn.20.../luns>
```

- Use the `create` command to create new LUNs for a target.

```
/iscsi/iqn.20.../luns> create /backstores/block/LUN_1
    lun1
/iscsi/iqn.20.../luns> create /backstores/block/LUN_2
    lun2
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The kernel target exports SCSI Logical Units to remote systems. Use the `targetcli` shell to link previously defined storage objects with a target, and to specify which Logical Unit Number (LUN) the device uses.

The following example uses the `create` command to create two new LUNs for a target. From the `targetcli` shell, begin by using the `cd` command to change to the `luns` directory within the `<target/TGP>` hierarchy.

```
/> cd iqn.2015-07.com.example.host02:tgt1/tpg1/luns
/iscsi/iqn.20.../luns>
```

The following commands create two LUNs from the previously defined block storage objects.

```
/iscsi/iqn.20.../luns> create /backstores/block/LUN_1 lun1
/iscsi/iqn.20.../luns> create /backstores/block/LUN_2 lun2
```

Use the `ls` command to view the LUNs.

```
/iscsi/iqn.20.../luns> ls
o- luns ..... [LUNs: 2]
  o- lun1 ..... [block/LUN_1 (/dev/xvdb)]
  o- lun2 ..... [block/LUN_2 (/dev/xvde)]
```

## Creating ACLs

- Access Control Lists (ACLs) restrict access to LUNs from remote systems.
- Create an ACL for each initiator to enforce authentication when the initiator connects to the target.
  - This gives a specific initiator exclusive access to a specific target.
- From the `targetcli` shell, use the `cd` command to change to the `acls` directory within the `<target/TGP>` hierarchy.

```
/> cd iqn.2015-07.com.example.host02:tgt1/tpg1/acls
/iscsi/iscsi/iqn.20.../acls>
```

- Use the `create` command to create an ACL for an initiator.

```
/iscsi/iscsi/iqn.20.../acls> create iqn.2015-
07.com.example.host02:host03
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Access Control Lists (ACLs) restrict access to LUNs from remote systems. You can create an ACL for each initiator to enforce authentication when the initiator connects to the target. This allows you to give a specific initiator exclusive access to a specific target.

The following example uses the `create` command to create an ACL for an initiator. From the `targetcli` shell, begin by using the `cd` command to change to the `acls` directory within the `<target/TGP>` hierarchy.

```
/> cd iqn.2015-07.com.example.host02:tgt1/tpg1/acls
/iscsi/iscsi/iqn.20.../acls>
```

The following command creates an ACL for the **host03** initiator.

```
/iscsi/iscsi/iqn.20.../acls> create iqn.2015-
07.com.example.host02:host03
```

Use the `ls` command to view the ACLs.

```
/iscsi/iscsi/iqn.20.../acls> ls
o- acls ..... [ACLs: 2]
  o- iqn.2015-07.com.example.host02:host03 ..... [Mapped LUNs:2]
    | o- mapped_lun1 ..... [lun1 block/LUN_1 (rw) ]
    | o- mapped_lun2 ..... [lun2 block/LUN_2 (rw) ]
```

## 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

An iSCSI client functions as an 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.

# Configuring an iSCSI Initiator

- To configure an iSCSI initiator, install the `iscsi-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
  - Edit `/etc/iscsi/initiatorname.iscsi` and provide the initiator name that you configured as an ACL on the target.
    - `InitiatorName=iqn.2015-07.com.example.host02:host03`
  - Use the `systemctl` utility to enable and start `iscsid`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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

```
# yum install iscsi-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

Edit the `/etc/iscsi/initiatorname.iscsi` file and replace the `InitiatorName` parameter with the initiator name that you previously configured as ACL on the target.

```
InitiatorName=iqn.2015-07.com.example.host02:host03
```

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

```
# systemctl enable iscsid
# systemctl start iscsid
```

## 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 192.0.2.102
```

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

```
# iscsiadm -m discoverydb -t st -p 192.0.2.102
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 192.0.2.102
192.0.2.102:3260,1 iqn.2011-12.com.example.mypc:tgt1
192.0.2.102:3260,1 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
192.0.2.102,3260
```

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 192.0.2.102:

```
# iscsiadm -m discoverydb -t st -p 192.0.2.102
# BEGIN RECORD 6.2.0-873-28
discovery.startup = manual
discovery.type = sendtargets
discovery.sendtargets.address = 192.0.2.102
discovery.sendtargets.port = 3260
discovery.sendtargets.auth.authmethod = None
discovery.sendtargets.auth.username = <empty>
discovery.sendtargets.auth.password = <empty>
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.reopen_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 192.0.2.102 -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 192.0.2.102 -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 192.0.2.102 -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 192.0.2.102:3260 -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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 192.0.2.102:3260 -l
Login to [iface: default, target: iqn.2011-12.com.example.mypc:tgt1, portal: 192.0.2.102:3260] 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:

```
# iscsiadadm -m session
tcp: [1] 192.0.2.102:3260,1 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:

```
# iscsiadadm -m session -P 1
Target: iqn.2011-12.com.example.mypc:tgt1
        Current Portal: 192.0.2.102:3260,1
        Persistent Portal: 192.0.2.102:3260,1
        *****
        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 about the attached SCSI devices. In this example, the target defines two LUNs:

```
# iscsiadadm -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: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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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, 25165824 sectors
 /dev/xvda1      *     2048     1026047      512000    83    Linux
 /dev/xvda2        1026048    25165823    12069888    8e    Linux LVM
Disk /dev/xvdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Disk /dev/xvdd: 10.7 GB, 10737418240 bytes, 20971520 sectors
Disk /dev/mapper/ol-root: 11.1 GB, 11068768256 bytes, ...
Disk /dev/mapper/ol-swap: 1287 MB, 1287651328 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, 20971520 sectors
Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
```

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

```
# tail -f /var/log/messages
...
<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
Partition type:
  p  primary (0 primary, 0 extended, 4 free)
  e  extended
Select (default p): ENTER
Using default response: p
Partition number (1-4, default 1): ENTER
First sector (8192-20971519, default 8192): ENTER
Using default value 8192
Last sector, +sectors or +size{K,M,G} (8192-20971519, default 20971519): +1G
Partition 1 of type Linux and of size 1 GiB is set
Command (m for help): w
...
# mkfs -t ext4 /dev/sda1
...
```

Make a mount point and mount the file system:

```
# mkdir /iscsi_dev
# mount /dev/sda1 -o _netdev /iscsi_dev
# df -h
Filesystem      Size   Used   Avail   Use%   Mounted on
...
/dev/sda1      976M   2.6M   907M    1%   /iscsi_dev
```

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_dev      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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



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

- a. targetcli
- b. tgtadm
- c. tgt-admin
- d. tgt-setup-lun
- e. iscsiamd



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



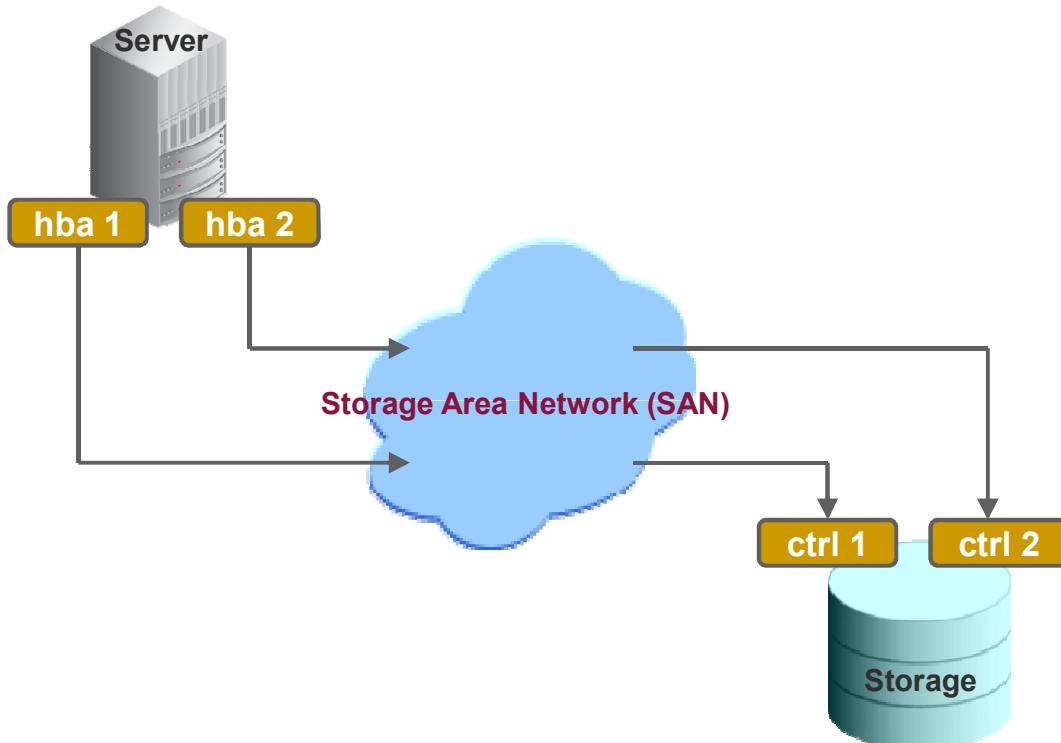
Which of the following statements are true?

- 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Device Mapper Multipathing



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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:
  - `/usr/sbin/multipath` – Utility that detects and configures multiple paths to devices
  - `/usr/sbin/multipathd` – DM-Multipath daemon
  - `/usr/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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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:

- `/usr/sbin/multipath` – Device mapper target auto-configuration that is used to detect and configure multiple paths to devices
- `/usr/sbin/multipathd` – Multipath daemon that checks for failed paths, and reconfigures the multipath map to regain connectivity. This daemon executes `/usr/sbin/multipath` when events occur.
- `/usr/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`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 file is 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.

The `multipath.conf` file contains the following sections, and each section contains one or more attributes or subsections.

- `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. Settings in this section have top priority.
- `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.

## defaults Attributes in /etc/multipath.conf

```
defaults {
    udev_dir                  /dev
    polling_interval           10
    path_selector               "round-robin 0"
    path_grouping_policy       multibus
    prio                      alua
    path_checker                readsector0
    rr_min_io                  100
    max_fds                     8192
    rr_weight                   priorities
    fallback                    immediate
    no_path_retry               fail
    user_friendly_names        yes
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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

- `prio` – One of the following methods is 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 is 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`.
- `fallback` – One of the following methods is used to manage path group fallback:
  - `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 (5)` man page for additional attributes and details.

## blacklist Section in /etc/multipath.conf

```
blacklist {
    wwid "*"
    devnode "^sd[a-z]"
        device {
            vendor
            product
        }
}

blacklist_exceptions {
    wwid "360000970000292602744533030303730"
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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
        product
    }
}
```

## multipaths Section in /etc/multipath.conf

```
multipaths {  
    multipath {  
        wwid      3600508b4000156d700012000000b0000  
        alias          yellow  
        fallback        manual  
        no_path_retry     5  
    }  
    multipath {  
        wwid      360000970000292602744533032443941  
        alias          green  
    }  
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 `fallback` 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"  
        path_grouping_policy "multibus"  
        path_checker     "directio"  
        features         "0"  
        hardware_handler "0"  
        prio             "const"  
        rr_weight        "uniform"  
    }  
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DM-Multipath includes support for the most common storage arrays. Run either of the following commands to view information on supported devices:

```
# multipathd show config  
# multipath -t
```

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>`



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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...
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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: /usr/sbin/mpathconf <command>
```

Commands :

Enable: `--enable`

Disable: `--disable`

Set `user_friendly_names` (Default `y`): `--user_friendly_names <y|n>`

Set `find_multipaths` (Default `y`): `--find_multipaths <y|n>`

Load the `dm-multipath` modules on enable (Default `y`): `--with ...`

start/stop/reload `multipathd` (Default `n`): `--with_multipathd ...`

By default, the `mpathconf` utility loads the `device-mapper-multipath` kernel module and enables the `user_friendly_names` and the `find_multipaths` attributes.

Run `mpathconf` without any arguments to display the status of DM-Multipath.

```
# mpathconf
multipath is enabled
find_multipaths is enabled
user_friendly_names is enabled
dm_multipath modules is not loaded
multipathd is not running
```

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 `multipath.conf(5)` man page 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_multipathd <y|n>` – If set to `y`, this runs `systemctl enable multipathd` to start `multipathd` on `--enable`, and runs `systemctl stop multipathd` on `--disable`.

Always reload the `multipathd` service after making any changes to the `/etc/multipath.conf` configuration file.

```
# systemctl reload multipathd
```

## multipath Utility

- Use the multipath utility to list and configure multiple paths to devices.
- To list the maximum multipath topology:

```
# multipath -ll
...
mpathb(36001405346939038cc9480caf0dd9a9d) dm-3 LIO-ORG
    ,IBLOCK
size=10g features='0' hwhandler='0' wp=rw
`-- policy='service-time 0' prio=1 status=active
    '- 2:0:0:2 sdb 8:16    active ready running
mpatha(36001405a7c28190541f4d61880050090) dm-2 LIO-ORG
    ,IBLOCK
size=10g features='0' hwhandler='0' wp=rw
`-- policy='service-time 0' prio=1 status=active
    '- 2:0:0:1 sda 8:0    active ready running
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The multipath utility is the device mapper target auto-configuration, 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, the `major:minor` format, the multipath map name (for example, `mpathN`), or the WWID to specify a device.

A sample output of the `multipath -ll` command is as follows:

```
# multipath -ll
...
mpathb(36001405346939038cc9480caf0dd9a9d) dm-3 LIO-ORG ,IBLOCK
size=10g features='0' hwhandler='0' wp=rw
`-- policy='service-time 0' prio=1 status=active
   `-- 2:0:0:2 sdb 8:16      active ready running
mpatha(36001405a7c28190541f4d61880050090) dm-2 LIO-ORG ,IBLOCK
size=10g features='0' hwhandler='0' wp=rw
`-- policy='service-time 0' prio=1 status=active
   `-- 2:0:0:1 sda 8:0      active ready running
```

The output for `mpathb` is described as follows:

- `mpathb` – User-friendly device name
- `36001405346939038cc9480caf0dd9a9d` – Unique WWID
- `dm-0` – sysfs file name
- `LIO-ORG` – Vendor name
- `IBLOCK` – Linux BLOCK device
- `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='service-time 0'` – Path selector algorithm
- `prio=1` – Path group priority
- `status=active` – Path group state
- `2:0:0:2` – SCSI information: host, channel, scsi\_id, and LUN
- `sdb` – Linux device name
- `8:16` – 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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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
<date_time> | ux_socket_listen error
# systemctl stop multipathd
# multipathd -d
<date_time> | mpatha: load table [0 20971520 multipath 0 0 1 1
service-time 0 1 1 8:0 1]
<date_time> | mpatha: event checker started
<date_time> | 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
2:0:0:1  sda    8:0     1    active   ready    running   .....
2:0:0:2  sdb    8:16    1    active   ready    running   .....
...
multipathd> list maps
name      sysfs  uuid
mpatha  dm-2   36001405a7c28190541f4d61880050090
mpathb  dm-3   36001405346939038cc9480caf0dd9a9d
multipathd> list devices
available block devices:
  sda  devnode whitelisted, monitored
  sdb  devnode whitelisted, monitored
  sr0  devnode blacklisted, unmonitored
  dm-0 devnode blacklisted, unmonitored
...
multipathd> quit
```

# iSCSI Multipathing

- From the initiator, enable DM-Multipath:

```
# mpathconf --enable
```

- Start the multipathd daemon:

```
# systemctl start multipathd
```

- The DM-Multipath devices now exist:

```
# ls -l /dev/mapper
crw-rw---- ... control
lrwxrwxrwx ... mpatha -> ../dm-2
lrwxrwxrwx ... mpathb -> ../dm-3
```

- Create a file system on /dev/mapper/mpatha.
- Create a mount point and mount the file system.
- If one of the network interfaces fails on the initiator, I/O continues through the remaining active interface.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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
```

Before starting the multipathd daemon, there are no mpathN devices in /dev/mapper:

```
# ls /dev/mapper/mpatha  
ls: cannot access /dev/mapper/mpatha: No such file or directory
```

After starting the multipathd daemon, the mpathN device is created in /dev/mapper:

```
# systemctl start multipathd  
# ls /dev/mapper/mpatha  
/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.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## 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. The `multipathd` daemon also has an interactive mode enabled with the `-k` option.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Configure an iSCSI server
- Use the `targetcli` administration shell
- Configure an iSCSI software initiator
- Use the `iscsiadm` utility
- Describe Device Mapper Multipathing
- Use the `mpathconf` and `multipath` utilities
- Configure iSCSI multipathing



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 12: Overview

These practices cover the following topics:

- Configuring an iSCSI server (target)
- Configuring an iSCSI client (initiator)
- Configuring iSCSI multipathing



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Control Groups (Cgroups)

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the purpose of control groups
- Describe control group subsystems and parameters
- Describe control group implementation in Oracle Linux 7
- Describe the control group hierarchy model
- Describe `systemd` slice units and scope units
- Use the `systemd-cgls` utility
- Display the cgroup tree of specific services and scopes
- Display system resource control settings
- Control access to system resources
- Modify unit configuration files
- Use the `systemd-run` utility



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Control Groups: Introduction

- Control groups (cgroups) enable you to allocate computing resources to specific processes or tasks. You can:
  - Assign a set of CPU cores and memory nodes to a specific group of tasks
  - Specify the relative share of CPU time available to the tasks in a cgroup
  - Specify memory limits for the tasks in a cgroup
  - Report the CPU time and memory used by cgroup tasks
  - Deny or allow tasks in a cgroup access to specific devices
- A cgroup subsystem, or kernel resource controller, represents a single kernel resource, such as memory or CPU cores.
- A cgroup associates a group of processes or tasks with a set of parameters for one or more cgroup subsystems.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Control groups (cgroups) provide a mechanism to put Linux processes (tasks) into groups ensuring that critical workloads get the system resources (CPU, memory, and I/O) that they need. You can allocate system resources, track usage, and impose limits on the cgroups.

cgroups provide fine-grained control of CPU, I/O, and memory resources. You can associate a set of CPU cores and memory nodes with a group of processes that make up an application or a group of applications. This enables the subsetting of larger systems; more fine-grained control over memory, CPUs, and devices; and the isolation of applications.

For example, with very large NUMA systems, you make the best use of system resources by compartmentalizing. cgroups give you a great deal of control over how to set up a system, which memory to give, and which CPUs to give to an individual task. You can pin processes to the same NUMA node and use NUMA-local memory. cgroups facilitate database consolidation on large NUMA servers, I/O throttling support, and device whitelisting. cgroups work inside virtual guests as well.

A cgroup subsystem is a kernel resource controller that applies limits or acts on a group of Linux processes. A subsystem represents a single kernel resource, such as memory or CPU cores. A cgroup associates a group of processes or tasks with a set of parameters for one or more subsystems. Processes assigned to each cgroup are subject to the subsystem parameters.

## cgroup Subsystems (Resource Controllers)

A subsystem is a kernel resource controller that applies limits or acts on a group of processes. Subsystems include:

- `blkio` – Sets limits on I/O bandwidth for block devices
- `cpu` – Provides CPU access to cgroup tasks
- `cpuacct` – Reports CPU resources usage
- `cpuset` – Assigns individual CPUs and memory nodes
- `devices` – Grants or denies access to devices
- `freezer` – Suspends or resumes cgroup tasks
- `memory` – Reports or limits memory use of cgroup tasks
- `net_cls` – Tags outgoing network packets with an identifier, which can then be assigned a different priority
- `perf_event` – Enables monitoring by using the `perf` tool
- `hugetbl` – Allows large page sizes of virtual memory



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Following are brief descriptions of the cgroup subsystems:

- `blkio` – Sets limits on I/O bandwidth for block devices
- `cpu` – Schedules CPU access to cgroup tasks. It is mounted together with the `cpuacct` controller on the same mount.
- `cpuacct` – Reports CPU resources used by tasks in a cgroup tasks
- `cpuset` – Assigns individual CPUs and memory nodes (for systems with NUMA architectures) to cgroup tasks
- `devices` – Grants or denies cgroup tasks access to devices
- `freezer` – Suspends or resumes cgroup tasks
- `memory` – Sets limits on memory use of cgroup tasks and reports on memory resources
- `net_cls` – Tags outgoing network packets with an identifier. You can configure the Linux traffic controller (`tc`) to assign different priorities to packets from different cgroups.
- `perf_event` – Enables monitoring of cgroups by using the `perf` tool
- `hugetbl` – Allows large page sizes of virtual memory and enforces resource limits on these pages

The `perf_event` and `hugetbl` resource controllers are new for Oracle Linux 7.

## View Mounted Resource Controllers

- View `/proc/cgroups` to list the currently mounted resource controllers:

```
# cat /proc/cgroups
```

- Use the `lssubsys` command to list the hierarchies.

```
# lssubsys -m
cpuset      /sys/fs/cgroup/cpuset
cpu,cpuacct  /sys/fs/cgroup/cpu,cpuacct
memory      /sys/fs/cgroup/memory
devices      /sys/fs/cgroup/devices
freezer     /sys/fs/cgroup/freezer
net_cls      /sys/fs/cgroup/net_cls
blkio       /sys/fs/cgroup/blkio
perf_event   /sys/fs/cgroup/perf_event
hugetlb     /sys/fs/cgroup/hugetlb
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

View `/proc/cgroups` to see the currently mounted resource controllers.

```
# cat /proc/cgroups
```

...

You can also use the `lssubsys` command to list the hierarchies. You must install the `libcgroup-tools` package to run the `lssubsys` utility. The following example includes the mount points (`-m`):

```
# lssubsys -m
...

```

Include the subsystem(s) arguments to list specific hierarchies, as in the following example:

```
# lssubsys -m blkio cpu
blkio      /sys/fs/cgroup/blkio
cpu,cpuacct  /sys/fs/cgroup/cpu,cpuacct
```

## cgroup Subsystem Parameters

- Each subsystem has specific parameters that enable resource control and reporting mechanisms.
- Subsystem parameters allow you to set limits, restrict access, or define allocations for each subsystem.
- Understanding the specific parameters helps you understand the possibilities for controlling resources with cgroups.
- Each subsystem cgroup also has a set of common files, including the `tasks` file:
  - The `tasks` file keeps track of processes associated with the cgroup and the associated subsystem parameter settings.
  - The `tasks` file contains all the process IDs (PIDs) assigned to the cgroup.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Each subsystem has specific parameters that enable resource control and reporting mechanisms. The subsystem parameters are the heart of cgroup resource controls. They allow you to set limits, restrict access, or define allocations for each subsystem. Understanding the specific parameters helps you to understand the possibilities for controlling resources with cgroups.

In addition to the subsystem parameters, each subsystem directory contains the following files:

- `cgroup.clone_children`
- `cgroup.event_control`
- `cgroup.procs`
- `notify_on_release`
- `release_agent`
- `tasks`

The `tasks` file keeps track of the processes associated with the cgroup and the associated subsystem parameter settings. The `tasks` file contains all the process IDs (PIDs) assigned to the cgroup.

# cgroup Implementation in Oracle Linux 7

- In previous versions of Oracle Linux, the cgroup mechanism was optional.
  - Utilities and files were provided by the `libcgroup` package.
  - Resource controllers were mounted under `/cgroup`.
  - You could define control groups, their parameters, and mount points in the `/etc/cgconfig.conf` file.
  - You could use utilities, such as `cgcreate`, `cgdelete`, `cgget`, `cgset`, and others, to manage the cgroups.
  - You needed to start the `cgconfig` service.
- In Oracle Linux 7, cgroups are an integral part of `systemd`.
  - You can use `systemctl` commands, or modify `systemd` unit files to manage system resources.
  - The only exception is the `net_prio` resource controller.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In previous versions of Oracle Linux, cgroup management utilities and files were provided by the `libcgroup` package. Resource controllers were mounted under the `/cgroup` directory. You could traverse the `/cgroup` hierarchy to view current control group hierarchies, parameter assignments, and associated tasks. You could define cgroups, their parameters, and mount points in the `/etc/cgconfig.conf` file. Utilities were provided by the `libcgroup` package, such as `cgcreate`, `cgdelete`, `cgget`, `cgset`, and others, to manage the cgroup mechanism. In previous versions of Oracle Linux, you also needed to start the `cgconfig` service.

In Oracle Linux 7, cgroups are no longer optional. cgroups in Oracle Linux 7 are an integral part of `systemd`. cgroup hierarchies are bound with the `systemd` unit tree. You now can use `systemctl` commands, or modify `systemd` unit files to manage system resources.

There is one resource controller, `net_prio`, that has not been compiled in the kernel. The `net_prio` resource controller requires that you load the `netprio_cgroup` kernel module and that you also install the `libcgroup-tools` package. The `libcgroup-tools` package provides the `/etc/cgconfig.conf` file, the `cgconfig` service, and the cgroup utilities needed to configure and use the `net_prio` resource controller. To avoid conflicts, do not use `libcgroup` tools for the main resource controllers.

## cgroup Hierarchies

- `systemd` automatically mounts hierarchies for kernel resource controllers in `/sys/fs/cgroup`:

```
# ls -l /sys/fs/cgroup
drwxr-xr-x. .... blkio
lrwxrwxrwx. .... cpu -> cpu,cpuacct
lrwxrwxrwx. .... cpuacct -> cpu,cpuacct
drwxr-xr-x. .... cpu,cpuacct
drwxr-xr-x. .... cpuset
drwxr-xr-x. .... devices
drwxr-xr-x. .... hugetlb
drwxr-xr-x. .... memory
drwxr-xr-x. .... net_cls
drwxr-xr-x. .... perf_event
drwxr-xr-x. .... systemd
```

- Subsystem parameters are stored in these directories.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

`systemd` automatically mounts hierarchies for kernel resource controllers in the `/sys/fs/cgroup` directory. For example:

```
# ls /sys/fs/cgroup
blkio  cpuacct      cpuset    freezer   memory   perf_event
cpu    cpu,cpuacct  devices  hugetlb  net_cls  systemd
```

The `cpu` resource controller is mounted together with the `cpuacct` controller on the same mount. The mount point is “`cpu,cpuacct`” and `cpu` and `cpuacct` are symbolic links to this mount point:

```
lrwxrwxrwx. .... cpu -> cpu,cpuacct
lrwxrwxrwx. .... cpuacct -> cpu,cpuacct
drwxr-xr-x. .... cpu,cpuacct
```

The parameters for each resource controller are defined in the respective directories:

```
# ls /sys/fs/cgroup/blkio
blkio.io_merged    blkio.throttle_io_service_bytes ...
blkio.io_queued   blkio.throttle.io_serviced ...
...
```

Partial listings of the remaining resource controller directories are listed in the following:

```
# ls /sys/fs/cgroup/cpu,cpuacct
cpu.shares      cpu.rt_period_us  cpu.rt_runtime_us  ...
cpu.stat        cpuacct.stat    cpuacct.usage       ...
...

# ls /sys/fs/cgroup/cpuset
cpuset.cpus     cpuset.mems    cpuset.cpu_exclusive ...
...

# ls /sys/fs/cgroup/devices
devices.allow   devices.deny    devices.list      ...
...

# ls /sys/fs/cgroup/freezer
freezer.state  ...

...

# ls /sys/fs/cgroup/hugetbl
hugetlb.2MB.failcnt  hugetlb.2MB.limit_in_bytes ...
...

# ls /sys/fs/cgroup/memory
memory.limit_in_bytes  memory.soft_limit_in_bytes ...
...

# ls /sys/fs/cgroup/net_cls
net_cls.classid ...
...

# ls /sys/fs/cgroup/perf_event
...
```

The kernel-uek-doc package provides documentation of all cgroup subsystems in the /usr/share/doc/kernel-doc-<version>/Documentation/cgroups directory. Following is a partial listing:

```
# ls /usr/share/doc/kernel-doc-<version>/Documentation/cgroups
00-INDEX          cpusets.txt  memory.txt
blkio-controller.txt  devices.txt  net_prio.txt
...
```

# The /sys/fs/cgroup/systemd Directory

Hierarchy of systemd unit types:

- Slice
- Scope
- Service

## `system.slice/`

Contains services and other system processes

- `sshd.service`
- `atd.service`
- `httpd.service`
- `firewalld.service`
- `network.service`
- `named.service`
- more...

## `user.slice/`

Contains `user-UID.slice/`

Contains user processes that run as scopes (transient cgroups )

- `/bin/dd...`
- `/sbin/ssh...`
- more...



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Included in the `/sys/fs/cgroup` directory is a `systemd` directory. The `systemd` directory contains a hierarchy of `slice`, `scope`, and `service` units for the cgroup tree.

```
# ls /sys/fs/cgroup/systemd
system.slice  user.slice  ...
...
```

The `system.slice` directory contains services and other system processes. For example:

```
# ls /sys/fs/cgroup/systemd/system.slice
abrt-ccpp.service  abrtd.service  abrt-oops.service ...
chronyd.service    colord.service  cpupower.service ...
...
...
```

The `user.slice` directory is the top-level directory for all user sessions. For example:

```
# ls /sys/fs/cgroup/systemd/user.slice
user-0.slice  user-42.slice  user-1000.slice ...
...
...
```

In this example, a directory exists for processes owned by UID 0 (root), UID 42 (gdm), and UID 1000 (oracle user).

The `user.slice` hierarchy contains user processes that run with transient cgroups called scopes. In this example, the `user.slice/user-1000.slice` directory contains a “scope” directory:

```
# ls /sys/fs/cgroup/systemd/user.slice/user-1000.slice  
session-3.scope ...  
...
```

## systemd Slice Units

- A slice is a concept for hierarchically managing resources of a group of processes.
- For each slice, certain resource limits can be set that apply to all processes of all units contained in that slice.
- Slices do not contain processes, they organize a hierarchy in which scopes and services are placed.
  - The actual processes are contained in scopes or in services.
- There are four slices that are created by default:
  - `- .slice` – The root slice
  - `system.slice` – Contains all system services
  - `user.slice` – Contains all user sessions
  - `machine.slice` – Contains all virtual machines and Linux containers



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A slice is a concept for hierarchically managing resources of a group of processes. This management is performed by creating a node in the cgroup tree. For each slice, certain resource limits can be set that apply to all processes of all units contained in that slice.

Slices are organized hierarchically in a tree. The name of the slice encodes the location in the tree. The name consists of a dash-separated series of names, which describes the path to the slice from the root slice. The root slice is named `- .slice`. Use the following command to list all loaded slices:

```
# systemctl -t slice -a
UNIT           LOAD  ACTIVE SUB    DESCRIPTION
-.slice         loaded active active  Root Slice
system-getty.slice  loaded active active  system-getty.slice
system-lvm2\x2dpvscan.slice  loaded inactive dead ...
system-systemd\x2dfsck.slice  loaded inactive dead ...
system.slice   loaded active active  System Slice
user-0.slice   loaded active active  user-0.slice
user-1000.slice  loaded active active  user-1000.slice
...
```

You can use the `find` command to list the location and file name of slice units:

```
# find / -name "*.slice"
...
/sys/fs/cgroup/systemd/user.slice
/sys/fs/cgroup/systemd/user.slice/user-1000.slice
/sys/fs/cgroup/systemd/user.slice/user-0.slice
/sys/fs/cgroup/systemd/user.slice/user-42.slice
/sys/fs/cgroup/systemd/system.slice
/sys/fs/cgroup/systemd/system.slice/system-lvm2\x2dpvscan.slice
/sys/fs/cgroup/systemd/system.slice/system-systemd\x2dfsck.slice
/sys/fs/cgroup/systemd/system.slice/system-getty.slice
/usr/lib/systemd/system/-.slice
/usr/lib/systemd/system/user.slice
/usr/lib/systemd/system/system.slice
/usr/lib/systemd/system/machine.slice
```

Slices do not contain processes, they organize a hierarchy in which scopes and services are placed. The actual processes are contained in scopes or in services.

By default, service and scope units are placed in `system.slice`. Virtual machines and containers are placed in `machine.slice`. User sessions are placed in `user.slice`.

## systemd Scope Units

- Scope units are similar to service units in that they both encapsulate processes.
  - Service unit processes are started by `systemd` based on service unit configuration files.
  - Scope unit processes are started by arbitrary processes via `fork()` and are registered by `systemd` at runtime.
- The main purpose of scope units is grouping worker processes for managing system resources.
- User sessions, virtual machines, and Linux containers are treated as scopes.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

`systemd` services units encapsulate processes that are started and stopped by `systemd` based on unit configuration files. Scope units encapsulate processes that are started and stopped by arbitrary processes via `fork()` and are then registered by `systemd` at runtime.

The main purpose of scope units is for grouping worker processes of a system service organization and for managing resources. User sessions, virtual machines, and Linux containers are treated as scopes. Use the following command to list all loaded scope units on your system:

```
# systemctl -t scope -a
UNIT          LOAD ACTIVE SUB      DESCRIPTION
session-3.scope  loaded active running Session 3 of user oracle
...

```

You can use the `find` command to list the location and file name of scope units:

```
# find / -name "*.scope"
...
/run/systemd/system/session-3.scope
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/session-
3.scope
```

## The `systemd-cgls` Utility

- Use the `systemd-cgls` command to display the slice and cgroup hierarchy.

```
# systemd-cgls
|-1 /usr/lib/systemd/system --switched-root -system
|-user.slice
| |-user-1000.slice
| | -session-3.scope
...
|-system.slice
...
```

- Note that services and scopes contain process and are placed in slices.
- Slices do not contain processes of their own with the exception of PID 1.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `systemd-cgls` command to display the slice and cgroup hierarchy. The `systemd-cgls` command recursively shows the contents of the cgroup hierarchy in a tree.

```
# systemd-cgls
|-1 /usr/lib/systemd/system --switched-root --system ...
|-user.slice
| |-user-1000.slice
| | -session-3.scope
| |   |-2029 gdm-session-worker [pam/gdm-password]
| |   |-2021 /usr/bin/gnome-keyring-daemon --daemonize ...
...
|-system.slice
| |-system-hostnamed.service
| | |-5653 /usr/lib/systemd/systemd-hostnamed
| |-udisk2.service
| | |-2258 /usr/lib/udisk2/udisksd --no-debug
...
```

## Displaying the cgroup Tree of Specific Services and Scopes

- Use the `systemctl status` command to view the tasks associated with a service:

```
# systemctl status sshd.service
...
Main PID: 1334 (sshd)
CGroup: /system.slice/sshd.service
| -1334 /usr/sbin/sshd -D
...
```

- Use the `systemctl status` command to view the tasks associated with a scope:

```
# systemctl status session-3.scope
...
CGroup: /user.slice/user-1000.slice/session-3.scope
| -2009 gdm-session-worker [pam/gdm-password]
| -2021 /usr/bin/gnome-keyring-daemon ...
...
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can also use the `systemctl` command to view the processes associated with a service and with a scope, but not slices. Recall that slices do not contain processes of their own. The following example shows that the `sshd` service is loaded and active. The `systemctl status` command shows the tasks in the cgroup that the service implements. The `ps` command verifies the PID of the `sshd` service. The PID of `sshd` is 1334 in this example:

```
# systemctl -t service
...
sshd.service loaded active running OpenSSH server daemon
...
# systemctl status sshd
...
CGroup: /system.slice/sshd.service
| -1334 /usr/sbin/sshd -D
...
# ps -ef |grep 1334
root    1334      1      ... /usr/sbin/sshd -D
```

The following example shows the tasks associated with a scope:

```
# systemctl -t scope
...
session-3.scope loaded active running Session 3 of user oracle
...
# systemctl status session-3.scope
...
CGroup: /user.slice/user-1000.slice/session-3.scope
| -2009 gdm-session-worker [pam/gdm-password]
| -2021 /usr/bin/gnome-keyring-daemon --daemonize ...
| -2031 dnome-session --session gnome-classic
| -2039 dbus-launch --sh-syntax --exit-with-session
| -2040 /bin/dbus-daemon --fork --print-pid 4 ...
| -2105 /usr/libexec/gvfsd
| -2138 /usr/libexec//gvfsd-fuse
/run/user/1000/gvfs...
| -2182 /usr/bin/ssh-agent /bin/sh -c exec -l /bin/...
| -2203 /usr/libexec/at-spi-bus-launcher
| -2207 /bin/dbus-daemon --config-file=/etc/at-
spi2/...
| -2211 /usr/libexec/at-spi2-registryd --use-gnome-...
| -2223 /usr/libexec/gnome-settings-daemon
| -2229 /usr/bin/pulseaudio -start
| -2255 /usr/libexec/gvfs-udisk2-volume-monitor
...
| -21843 /bin/bash
| -21874 su -
| -21881 -bash
...
...
```

# Viewing cgroup Resource Control Settings

- Use the `systemctl show` command to view the properties of a `systemd` unit.
  - Included in the output are resource control settings.
- For example, to view the “CPU” resource settings:

```
# systemctl show sshd | grep -i cpu
CPUAccounting=no
CPUShares=1024
...
```

- To view the “memory” resource settings:

```
# systemctl show user-1000.slice | grep -i mem
MemoryAccounting=no
MemoryLimit=18446744073709551615
...
```

- See the `systemd.resource-control(5)` man page.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `systemctl show` command to view the properties of a `systemd` unit. Included in the output are resource control settings. See the `systemd.resource-control(5)` man page for a description resource control unit settings. The following example displays all properties for the `sshd` service unit:

```
# systemctl show sshd
Id=sshd.service
Names=sshd.service
...
...
```

To view CPU resource settings for the `sshd` service:

```
# systemctl show sshd | grep -i cpu
LimitCPU=18446744073709551615
CPUSchedulingPolicy=0
CPUSchedulingPriority=0
CPUSchedulingResetOnFork=no
CPUAccounting=no
CPUShares=1024
```

Descriptions of some of the “CPU” properties are provided by the `systemd.resource-control(5)` man page:

- `CPUAccounting=on`: Turn on CPU usage accounting for this unit. Set to “off” to turn accounting off. Note that turning on CPU accounting for one unit might also implicitly turn it on for all units contained in the same slice and for all its parent slices and units contained therein.
- `CPUShares=1024`: Assign the specified overall CPU time share weight to the processes executed. This controls the “cpu.shares” control group attribute, which defaults to 1024. For details of about this control group attribute, see the `/usr/share/doc/kernel-doc-<version>/Documentation/scheduler/sched-design-CFS.txt` file.

Descriptions of the remaining “CPU” properties are provided by the `systemd.exec(5)` man page and other man pages:

- `LimitCPU=18446744073709551615`: Controls resource limits for executed processes. See the `setrlimit(2)` man page for details.
- `CPUSchedulingPolicy=0`: Sets the CPU scheduling policy for executed processes. See the `sched_setscheduler(2)` man page for details.
- `CPUSchedulingPriority=0`: Sets the CPU scheduling priority for executed processes. The available priority range depends on the selected CPU scheduling policy. See the `sched_setscheduler(2)` man page for details.
- `CPUSchedulingResetOnFork=no`: Takes a Boolean argument. If true, elevated CPU scheduling priorities and policies are reset when the executed processes fork and can hence not leak into child processes. See the `sched_setscheduler(2)` man page for details. Defaults to false.

To view memory resource settings:

```
# systemctl show sshd | grep -i mem
LimitMEMLOCK=65536
MemoryAccounting=no
MemoryLimit=18446744073709551615
```

Descriptions of some of the “mem” properties are provided by the `systemd.resource-control(5)` man page:

- `MemoryAccounting=on`: Turn on process and kernel memory accounting for this unit. Set to “off” to turn accounting off. Note that turning on memory accounting for one unit might also implicitly turn it on for all units contained in the same slice and for all its parent slices and units contained therein.
- `MemoryLimit=18446744073709551615`: Specify the limit on maximum memory usage of the executed processes. The limit specifies how much process and kernel memory can be used by tasks in this unit. Takes a memory size in bytes. If the value is suffixed with K, M, G, or T, the specified memory size is parsed as Kilobytes, Megabytes, Gigabytes, or Terabytes (with the base 1024), respectively. This controls the “memory.limit\_in\_bytes” control group attribute. For details about this control group attribute, see the `/usr/share/doc/kernel-doc-<version>/Documentation/cgroups/memory.txt` file.

See the `systemd.exec(5)` and `setrlimit(2)` man pages for description of the `LimitMEMLOCK` property. This property defines the maximum number of bytes of memory that can be locked into RAM.

# Controlling Access to System Resources

- Use the `systemctl set-property` command to change resource control settings.
- The changes are applied immediately and persist across system reboots.
  - With the `--runtime` option, changes are not persistent.
- To change the `CPUShares` property to 512:

```
# systemctl set-property sshd.service CPUShares=512
```

- This command creates the following directory, populates the directory with the `cpu` and `cpuacct` subsystem parameters, and updates the `cpu.shares` parameter:

```
# cat
  /sys/fs/cgroup/cpu,cpuacct/system.slice/sshd.service/
    cpu.shares
512
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `systemctl set-property` command to change resource control settings and control access to system resources. Resource control setting properties described in the `systemd.resource-control(5)` man page can be changed at runtime. The changes are applied immediately and persist across system reboots. If you specify the `--runtime` option, the setting does not persist across system reboots.

The following example changes the `CPUShares` property from the default of 1024 to 512. This halves the access the processes in the cgroup have to CPU time.

```
# systemctl set-property sshd.service CPUShares=512
```

The change is effective immediately:

```
# systemctl show --property=CPUShares sshd
CPUShares=512
```

The change is also written to the configuration unit file so that it persists across reboot:

```
# cat
/sys/fs/cgroup/cpu,cpuacct/system.slice/sshd.service/cpu.shares
512
```

You can change multiple properties in the same command. Assigning an empty value to a parameter resets the parameter value to the default.

Note that the `/sys/fs/cgroup/cpu,cpuacct/system.slice/sshd.service` directory is first created and populated with `cpu,cpuacct` resource controller parameters when the `CPUShares` property is set.

For example, before setting any “memory” kernel resource parameters, the “memory” directory does not exist:

```
# ls /sys/fs/cgroup/memory/system.slice/sshd.service
ls: cannot access /sys/fs/cgroup/memory/system.slice/sshd.service:
No such file or directory
```

Using the `systemctl set-property` command to set a “memory” property creates and populates the directory. The following example limits the maximum amount of memory tasks in the `sshd.service` cgroup can use to 1 GB:

```
# systemctl set-property sshd MemoryLimit=1G MemoryAccounting=true
```

Note that the directory is created and populated with the memory subsystem parameters:

```
# ls /sys/fs/cgroup/memory/system.slice/sshd.service/
cgroup.clone_children           memory.max_usage_in_bytes
cgroup.event_control            memory.memsw.failcnt
cgroup.procs                     memory.memsw.limit_in_bytes
...
memory.limit_in_bytes          tasks
```

Note that the `memory.limit_in_bytes` parameter is set to the new value, 1 GB:

```
# ls /sys/fs/cgroup/memory/system.slice/sshd.service/
# cat memory.limit_in_bytes
1073741824
```

Also note that the contents of the `tasks` file in this directory contains the PID of the `sshd` process. The PID is 1337 in this example:

```
# cat tasks
1337
# ps -ef | grep 1337
root 1337      1  ...  /usr/sbin/sshd -D
...
```

# Modifying Unit Configuration Files

- You can define resource control settings in unit configuration files.
- Do not edit `/usr/lib/systemd/system/` unit files.
  - Copy the unit configuration file you want to edit from the `/usr/lib/systemd/system/` directory to the `/etc/systemd/system/` directory.
- Change the resource settings for a service under the `[Service]` heading in the unit configuration file.
  - Refer to the `systemd.resource-control(5)` man page for descriptions of resource control settings.
- After editing, reload all unit files and restart the service.

```
# systemctl daemon-reload
# systemctl restart <service>
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

`systemd` units are defined by unit configuration files located in the following directories:

- `/usr/lib/systemd/system`: `systemd` units distributed with installed RPM packages.
- `/run/systemd/system`: `systemd` units created at runtime. This directory takes precedence over the directory with installed service units.
- `/etc/systemd/system`: `systemd` units created and managed by the system administrator. This directory takes precedence over the directory with runtime units.

In addition to changing resource control settings from the command line using the `systemctl` command, you can define resource control settings in these unit configuration files.

Do not edit files in the `/usr/lib/systemd/system` directory. Copy the desired unit configuration file from this directory into the `/etc/systemd/system` directory and edit this file. For example, to define resource control settings for the `sshd` service:

```
# cp /usr/lib/systemd/system/sshd.service /etc/systemd/system/
# vi /etc/systemd/system/sshd.service
...
```

Change the resource settings for a service under the `[Service]` heading in the unit configuration file.

# The `systemd-run` Utility

- Use the `systemd-run` command to run a program in a transient scope or service unit.
  - When run in a service unit, the command runs in a clean execution environment. `systemd-run` starts the service asynchronously in the background and immediately returns.
  - When run in a scope unit, the command inherits the execution environment of `systemd-run`. Execution is synchronous and returns only when the command finished.
- Syntax:
 

```
systemd-run --unit=name --scope --slice=sname command
```

  - `--unit` – Specify unit name.
  - `--scope` – Create a scope unit instead of service unit.
  - `--slice` – Specify a new or existing slice.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `systemd-run` command to run a program in a transient scope or service unit. This allows you to set limits on resources consumed by the service during its runtime. The transient unit is removed automatically as soon as the service is stopped.

When a command is run as a transient service unit, the command is started and managed by `systemd` like any other service and is included in the output of the `systemctl` command like any other unit. The command runs in a clean and detached execution environment.

`systemd-run` starts the service asynchronously in the background and immediately returns.

When a command is run as a transient scope unit, the command is started directly by `systemd-run` and inherits the execution environment of the caller. The command is managed by `systemd` and is included in the output of `systemctl`. Execution is synchronous and execution returns only when the command finished.

Syntax for the `systemd-run` command follows:

```
systemd-run --unit=<name> --scope --slice=<slice_name> <command>
```

If `--unit` is not specified, the unit name is automatically generated. A transient service unit is created by default. Include the optional `--scope` parameter to create a transient scope unit. Use the `--slice` parameter to specify a new or existing slice. Otherwise, the transient service or scope is created in `system.slice`. Provide the command to run as the final argument.

Additional `systemd-run` options include the following:

- `--description` – Create a description of the unit
- `--remain-after-exit` – Keep the service until explicitly stopped. This is useful to collect runtime information about the service after it finished running.
- `--help` – Display usage with additional options

The following example runs the `dd` command as a transient service:

```
# systemd-run dd if=/dev/zero of=/dev/null bs=1024
Running as unit run-8928.service.
```

The “Running as ...” message displays to confirm the service starts successfully. The unit name, `run-8928.service`, is automatically generated because the `--unit=name` option is not specified.

The `find` command displays the location of the `run-8928.service` file:

```
# find / -name run-8928.service
/run/systemd/system/run-8928.service
/sys/fs/cgroup/cpu,cpuacct/system.slice/run-8928.service
/sys/fs/cgroup/systemd/system.slice/run-8928.service
```

Note that the service is placed in the `system.slice` hierarchy by default. The service is included as output of the `systemctl` command:

```
# systemctl
...
run-8928.service loaded active running /bin/dd if=/dev/zero ...
...
```

You can use the `top` utility to show that the `dd` command is using much of the CPU and memory resources. You can also use the `systemd-cgtop` command, which shows top cgroups by their resource usage:

```
# systemd-cgtop
Path                      Tasks    %CPU   Memory  Input/s ...
/
          159    100.0  744.2M   -
/system.slice                48     86.1   5.7M   -
/system.slice/run-8928.service  1     86.1     -   -
...
```

Transient cgroups are released automatically as soon as the processes they contain finish. You can also use the `systemctl stop` command to stop the service:

```
# systemctl stop run-8928
```

## Quiz



cgroups enable you to do which of the following?

- a. Specify the relative share of CPU time available to the tasks in a cgroup.
- b. Specify memory limits for the tasks in a cgroup.
- c. Suspend the tasks in a cgroup.
- d. Report the CPU time and memory used by the tasks in a cgroup.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. `systemd` automatically mounts hierarchies for kernel resource controllers in the `/cgroup/` directory.
- b. The `system.slice` directory contains services and other system processes.
- c. The `user.slice` directory is the top-level directory for all user sessions.
- d. Slices do not contain processes, they organize a hierarchy in which scopes and services are placed.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following commands allow you to run a program in a transient scope or service unit?

- a. systemctl
- b. systemd-cgls
- c. systemd-cgtop
- d. systemd-run



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe the purpose of control groups
- Describe control group subsystems and parameters
- Describe control group implementation in Oracle Linux 7
- Describe the control group hierarchy model
- Describe `systemd` slice units and scope units
- Use the `systemd-cgls` utility
- Display the cgroup tree of specific services and scopes
- Display system resource control settings
- Control access to system resources
- Modify unit configuration files
- Use the `systemd-run` utility



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 13: Overview

This practice covers the following topics:

- Exploring cgroup integration into `systemd`
- Exploring cgroup hierarchies and cgroup subsystem parameters
- Controlling access to system resources



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.



# Virtualization with Linux

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe virtualization and its benefits
- Explain how Linux as a virtual guest supports the different virtualization modes
- Outline the support for Linux as a guest operating system (OS) with various virtualization solutions
- Describe the KVM hypervisor
- Use the libvirt tools to create and manage KVM virtual guests



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson begins with a general discussion of virtualization technology. There is an overview of the major virtualization products. For each virtualization product, this lesson discusses the features, the major components, the integration package and, if it exists, the cloud product associated with the virtualization product. There are many virtualization products that are not discussed in this lesson. The following URL provides a comparison of platform virtual machine packages:

[http://everything.explained.at/Comparison\\_of\\_platform\\_virtual\\_machines/](http://everything.explained.at/Comparison_of_platform_virtual_machines/).

This lesson concludes with a discussion of Kernel-based Virtual machine (KVM). In the practice associated with this lesson, you create and manage virtual guests in a KVM environment.

# Virtualization: Introduction

- The ability to run a virtual instance of:
  - An operating system
  - An application environment
  - A storage or network resource
- Virtualization provides the following benefits:
  - Better hardware utilization
  - Server consolidation
  - Faster deployments
  - Foundation for cloud computing



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Virtualization is usually associated with running one or more virtual instances of an operating system on a physical host.

But virtualization is not limited to running an operating system (OS) on a virtual machine.

Virtualization extends to:

- Creating virtual application environments (for example, with Linux Containers or Oracle Solaris Zones)
- I/O virtualization: Virtualizing a physical Host Bus Adapter (HBA) by using NPIV, which stands for N-Port ID Virtualization or virtualizing a network card by using SR-IOV, which stands for single root I/O virtualization

Virtualization provides many benefits including:

- Increasing your hardware utilization
- Consolidating your servers into fewer, more powerful, and better-utilized platforms
- The ability to quickly provide environments for development and test operations
- Providing the foundation for creating private or public cloud environments

In this topic, you explore how you can:

- Deploy Linux in a virtual environment
- Create a virtual environment with Oracle Linux

# Virtualization Concepts

- The virtual environment runs on a physical host.
  - The virtual instances on the physical host are called virtual machines.
  - Virtual machines run a guest OS like Oracle Linux or Microsoft Windows.
- Virtual machines are managed by a layer called the hypervisor.
  - A type-1 hypervisor runs directly on the hardware of the physical host.
  - A type-2 hypervisor runs in the operating system installed on the physical host.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A hypervisor provides a virtualized environment for the guests running on the physical platform. The hypervisor creates and runs the virtual machines. The hypervisor can be:

- A software layer
- In the firmware of the physical host, for example, Oracle VM Server for SPARC
- Partly in the hardware, to supplement the software layer with virtualization hardware extensions (Intel VT-x or AMD-V)

Server virtualization products, such as Oracle VM Server for x86 or Microsoft Hyper-V, run directly on the physical platform. These products contain a hypervisor layer that communicates directly with the hardware. For this reason, this kind of hypervisor is called a **type-1 hypervisor** or bare-metal hypervisor.

A **type-2 hypervisor**, also known as a host-based hypervisor, is designed to run within a traditional operating system. This type of hypervisor adds a distinct layer to the OS and the running virtual guest becomes a third software layer. Examples of this type of virtualization include Parallels and Oracle VM VirtualBox.

KVM is considered both a type-1 and type-2 (type-1/2) hypervisor, because KVM turns the Linux kernel into a bare-metal hypervisor, but the OS running on the virtualization host is a full OS.

# Virtualization Modes

- Virtualization must handle:
  - Disk devices and network devices
  - Privileged instructions
  - Memory access and paging
  - Buses, interrupts, and timekeeping
  - BIOS and the booting process
- With full virtualization, all aspects of a guest OS are emulated.
- With paravirtualization, the guest OS communicates with the hypervisor by using hypercalls.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Full Virtualization

For a fully hardware-virtualized machine (HVM) type of guest, all aspects of the virtual machine are virtualized. The guest OS running on the virtual machine does not need to know that it is running in a virtual environment and can run unmodified. Privileged instructions issued by the guest OS are trapped by the hypervisor and translated into safe, emulated instructions on the system hardware.

Full virtualization takes advantage of the virtualization hardware extensions offered by the physical server. These hardware extensions are required to run fully virtualized virtual machines.

Many products such as VMware and Microsoft's virtualization offerings use full virtualization, although each vendor also supports some type of virtualization-aware interface in the virtual machine to optimize access to I/O devices.

## Paravirtualization

With paravirtualization, the guest running on the virtual machine is virtualization aware.

Paravirtualization is not an all-or-nothing mode: It represents a spectrum of support for the virtual environment.

## Full Paravirtualization

Paravirtualized or PV guests run a modified version of the guest operating system, which is virtualization aware. With full paravirtualization, the PV guests do not require the presence of hardware virtualization extensions on the host processor. This type of guest is supported by just a few hypervisors, such as Xen-based virtualization products.

You can find the list of operating systems that can run as Xen PV guests at this location: [http://wiki.xen.org/wiki/DomU\\_Support\\_for\\_Xen](http://wiki.xen.org/wiki/DomU_Support_for_Xen). This list includes major Linux distributions and Oracle Solaris 11.

**Note:** Xen is a type-1 hypervisor that allows guests to run either as fully paravirtualized (PV guests) or as hardware virtualized (HVM guests), with or without paravirtualized drivers. Xen is available as open source and is used by virtualization products such as Citrix XenServer and Oracle VM Server for x86. Find more information about Xen and virtualization at this site: [http://wiki.xenproject.org/wiki/Xen\\_Overview](http://wiki.xenproject.org/wiki/Xen_Overview).

## Paravirtualized Drivers

You can install paravirtualized (PV) drivers in your hardware-virtualized (HVM) guest OS to optimize access to disk and network devices. PV drivers are idealized device drivers that map operations to the real device drivers in the virtualization host. Oracle VM Server for x86, which is Xen-based, refers to this type of guest as Hardware Virtualized Machine (HVM) with PV drivers. Most virtualization vendors offer this type of paravirtualization support for selected operating systems.

In the Xen virtual environment, an HVM guest that can also support virtualized interrupts and timers in addition to running PV drivers is said to run in PVHVM mode. The Oracle Linux kernel provides this type of paravirtualization support, called paravirt-ops. Paravirt-ops is discussed further in the next slide.

You can find more information about the virtualization spectrum available for Xen-type virtual environments like Oracle VM Server for x86 and Citrix XenServer at this site:

[http://wiki.xen.org/wiki/Virtualization\\_Spectrum](http://wiki.xen.org/wiki/Virtualization_Spectrum).

As mentioned previously, most virtualization solutions offer paravirtualization support for selected operating systems. This paravirtualization support by the major virtualization providers is highlighted throughout this lesson.

# Linux and Xen Integration

- Pvops stands for paravirt-ops.
- The pvops kernel:
  - Contains all PV drivers
  - Can determine whether the underlying system supports full virtualization or paravirtual operations
  - Allows the guest running Linux to switch into PV, HVM, or HVM with PV drivers mode at boot time
- The Oracle Linux Unbreakable Enterprise Kernel (UEK) is a pvops kernel.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Paravirt-ops, or pvops, provides the support in the Linux kernel for virtual guests to run as PV guests on a Xen hypervisor, including Xen-based Oracle VM, with its Oracle VM Server for x86 platform.

With its pvops kernel, a Linux OS can boot natively on a physical host or boot as a guest OS in a virtual machine. If booting in a virtual machine, the Linux with pvops kernel can support HVM operation if the underlying hypervisor supports only full virtualization.

Several Linux distributions offer pvops support, including Oracle Linux, Red Hat Enterprise Linux (RHEL), Fedora, and Debian. RHEL provides the network and block storage paravirtualized drivers, but the tighter integration with Xen, which provides support for the management domain (called control domain or dom0), was removed starting with RHEL 6. Red Hat is now adopting KVM as its virtualization platform. You learn about dom0 with Oracle VM Server for x86 and KVM later in this lesson.

Oracle Linux 5 and later versions provide pvops support.

# Running Linux in a Virtual Machine

- You can run Linux in a virtual machine with:
  - Oracle VM Server for x86
  - Microsoft Hyper-V
  - Oracle VM VirtualBox
  - VMware vSphere
  - Citrix XenServer
  - KVM
- Most virtualization providers offer integration packages to optimize Linux operations in the virtual machine.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can run Linux as a guest operating system with most virtualization products. The list in the slide contains only a partial list of virtualization solutions. If you plan to run Oracle Linux in a virtual environment, you can access information about support for Oracle Linux with the major virtualization products at My Oracle Support, and search for Oracle Linux support policies.

## Running Linux in a Virtual Environment

All virtualization products offer integration and services when running Linux as a virtual OS. This support can be provided through the emulation layer, with specialized drivers, or by support already present in the Linux distribution.

Virtualization providers offer integration packages to provide additional support such as:

- Heartbeat, which detects whether the virtual machine is running
- Integrated shutdown, where you can shut down the virtual machine from the virtualization management component
- Mouse support to help with mouse synchronization
- Messaging, which allows communication as key/value pairs between the virtual machine and the management layer

For Hyper-V, this integration package is called Linux Integration Services. For Oracle VM, it is called Guest Additions, and for VMware, it is called VMware Tools.

# Oracle VM Server for X86

- Oracle VM is an enterprise-class server virtualization solution.
- Oracle VM Server for x86 is the component of Oracle VM to deploy virtual workloads on the x86 platform.
  - It includes a type-1 Xen hypervisor.
- Oracle VM Server for x86 is the x86 virtualization platform for Oracle's cloud computing solutions.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Oracle VM

Oracle VM is Oracle's server virtualization solution for both x86 and SPARC architectures and supports a variety of workloads such as Linux, Windows, and Oracle Solaris.

### Oracle VM Server for x86

Oracle VM Server for x86 is part of the Oracle VM virtualization solution and provides the x86 server virtualization component for both Oracle and non-Oracle workloads. In addition to Oracle Linux and Oracle Solaris, Oracle VM Server for x86 supports Red Hat, CentOS, and SUSE Linux Enterprise Server. You can find a complete list of supported guest operating systems in the Oracle VM Release Notes, Part Number E35329-08 or newer.

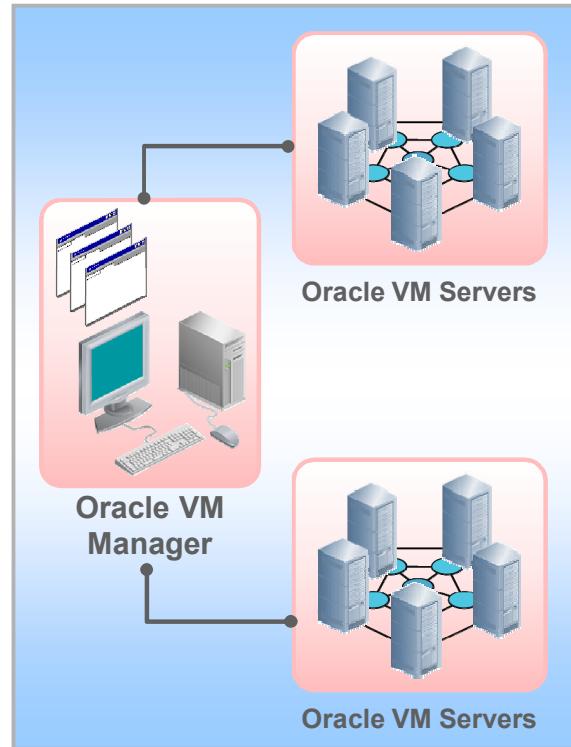
### Oracle Cloud Solution

Oracle Enterprise Manager Cloud Control (EMCC) offers a solution that enables you to create, manage, and monitor a private cloud. Oracle VM Server for x86 is the virtualization platform for Oracle cloud computing service models:

- Infrastructure as a Service (IaaS), which makes available resources such as processing, networking, and storage
- Platform as a Service (PaaS), which makes available platforms onto which you can deploy applications
- Software as a Service (SaaS), which makes available an application that you can use and customize

# Oracle VM Server for x86 Components

- Each Oracle VM server in the environment is a separate virtualization platform.
- The Oracle VM Manager tracks and manages the resources in your virtual environment.
- Client access to Oracle VM includes:
  - Web-based UI
  - CLI
  - Web Services API



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Oracle VM Servers

Oracle VM Server for x86 is installed on a physical host. The Xen hypervisor runs directly on the host hardware. A special virtual machine, called dom0, is the host operating system running on top of the Xen hypervisor, and acts as the control domain. Each virtual guest runs in a domain called domU. Dom0, as opposed to domU-type domains, is allowed to run privileged instructions and provides drivers for the hardware on the host platform.

## Oracle VM Manager

The Oracle VM Manager tracks and manages the resources available in your virtual environment. These resources include the resources in each of the Oracle VM servers, as well as the connected networks and storage. If an action is required on the resources, the Oracle VM Manager delegates an Oracle VM server to carry out the task.

You use the Oracle VM Manager to create virtual machines, and the virtual machines run on Oracle VM servers.

The diagram in the slide shows Oracle VM servers in groupings called server pools. You use the Oracle VM Manager to create pools. Most virtualization products offer a pooling feature that is the basis for other features such as high availability and live migration. With live migration, you can migrate a running virtual machine from one virtualization host to another virtualization host. Oracle VM Server for x86 offers this feature.

## Oracle VM Management Interfaces

Oracle VM Server for x86 offers several ways to access and manage your environment:

- The Oracle VM Manager user interface (UI)

You access the Oracle VM Manager by using a browser-based UI. The Oracle VM Manager UI is the most widely used management interface. From the UI, you can perform nearly all administrative functions.

- The Oracle VM command-line interface (CLI)

With the Oracle VM CLI, you can automate configuration and operational functions by writing scripts that include embedded Oracle VM CLI commands.

- The Oracle VM Web Services

The Oracle VM Web Services offer you a programmatic interface to Oracle VM. You can use the Representational State Transfer (REST) or Simple Object Access Protocol (SOAP) communication protocols from within Java, Python, or any other language that supports access to web services to configure, manage, or monitor your virtual environment.

## Oracle VM Training

Oracle University offers several courses for Oracle VM. Go to

<http://education.oracle.com/virtualization> and click Server Virtualization.

# Linux as a Guest OS with Oracle VM Server for X86

- Supported Linux guests can run using PV, HVM, or HVM with PV driver virtualization modes.
- Install Linux from an ISO file or a template from the Oracle Software Delivery Cloud.
- Install the Oracle VM Guest Additions package in the virtual machine for bi-directional messaging and first-boot configuration.
- Manage the life cycle of your Linux guests from the Oracle VM Manager.
- Deploy Linux within a cloud infrastructure with Enterprise Manager Cloud Control.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Oracle Linux as a Guest with Oracle VM Server for x86

Oracle Linux is very well suited for running in a virtual machine with Oracle VM Server for x86. Oracle Linux supports paravirtualization (PV) modes and full virtualization mode (HVM).

Oracle VM Server for x86 has been designed and tested to handle mission-critical enterprise workloads.

## Oracle VM Guest Additions

The Oracle VM Guest Additions form a bi-directional messaging channel between Oracle VM Manager and the guest OS. This communication channel allows first-boot installation configuration. Nearly all of the Oracle VM templates available from the Oracle Software Delivery Cloud already have Guest Additions installed.

## Oracle VM Server for x86 and Oracle Enterprise Manager Cloud Control

You can use Oracle Enterprise Manager Cloud Control to build cloud solutions based on Oracle VM Server for x86. Oracle Enterprise Manager Cloud Control extends the functionality of Oracle VM by offering centralized monitoring, self-service provisioning, and configuration management, including a facility to store cloud resources such as templates and patches in a software library. Find out more about Oracle Enterprise Manager Cloud Control at <http://www.oracle.com/technetwork/oem/cloud-mgmt-496758.html>.

# Linux as a Guest OS with Oracle VM VirtualBox

## Oracle VM VirtualBox:

- Is the most popular virtualization software
- Incorporates a type-2 (host-based) hypervisor
- Runs on Mac OS X, Windows, Linux, or Oracle Solaris
- Supports several flavors of Linux, running as guest OS on a virtual machine:
  - Oracle Linux, RHEL, Ubuntu, Debian, SUSE, Mandriva, Fedora, CentOS, Oracle Solaris, and more
- Is a great choice for desktop virtualization



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle VM VirtualBox is a general-purpose virtualization product for x86 hardware. It is a type-2 hypervisor virtualizer that you install in an already existing OS, such as Windows, Linux, MAC OS X, or Oracle Solaris. It provides full virtualization only. Though you can use Oracle VM VirtualBox for server provisioning, its desktop extension pack makes it an excellent choice for desktop virtualization. You can find more information about VirtualBox extension packs at this location: <https://www.virtualbox.org/manual/ch01.html#intro-installing>.

VirtualBox supports several operating systems running as guests in a virtual machine. Find the list of supported guest operating systems at [https://www.virtualbox.org/wiki/Guest\\_OSes](https://www.virtualbox.org/wiki/Guest_OSes).

VirtualBox is a great choice for evaluating Oracle Linux as a platform for running your business applications, because you can run Oracle Linux in a VirtualBox virtual machine on a desktop. At this site, <http://www.oracle.com/technetwork/articles/servers-storage-admin/evaluating-linux-vb-1934676.html>, you can find information about evaluating Oracle Linux, plus a link to an Oracle Linux VM download for VirtualBox.

## VMware vSphere

- vSphere is a family of products providing VMware's virtualization platform.
- ESXi is vSphere's bare-metal (type-1) hypervisor.
  - You install ESXi on a 64-bit processor host.
- vCenter is the management layer for vSphere.
  - You install vCenter on a 64-bit Microsoft Windows platform.
- vSphere is the building block for VMware's vCloud offering.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

### VMware vSphere

VMware vSphere is a virtualization platform that comprises several virtualization products, associated tools, and components.

### VMware ESXi

A VMware virtualization environment includes one or more ESXi physical hosts that form the virtualization layer. The ESXi host is a type-1 or bare-metal hypervisor. You attach a storage and networking infrastructure to your ESXi hosts and use the vSphere management layer to make these resources available to deploy virtual machines.

### VMware vCenter Server

vCenter is the management layer for vSphere. It provides a single point of control to manage the storage and network resources configured to the ESXi hosts and to assign these resources to the virtual machines. vCenter also provides:

- User access control by connecting to Active Directory
- A repository of management information in an Oracle, Microsoft SQL Server, or IBM DB2 database. Stored information includes host and virtual machine configurations, user permissions and roles, resource inventory such as storage resources, and performance statistics.

You install the vCenter Server on a 64-bit Microsoft Windows platform. You can have a single, stand-alone instance of vCenter or join the instances into a group. You can also deploy your vCenter instances into a highly available configuration.

### **VMware Cloud Offering**

With VMware vCloud, you can build a vSphere-based private cloud. You can find more information about vCloud at this location: <http://www.vmware.com/products/vcloud-suite/>.

# Linux as a Guest OS with VMware vSphere

- vSphere provides several client interfaces to create virtual machines:
  - A vSphere local client and web client
  - Several CLI tools (for example, `vmkfstools` and `vmware-cmd`)
  - A vSphere Web Services API
- Provision a guest OS in the virtual machine by:
  - Installing from media
  - Deploying from a template
  - Cloning an existing virtual machine
- VMware offers paravirtual drivers: PVSCSI and VMXNET 3



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## vSphere Clients

You manage your vSphere environment by accessing the vCenter server using vSphere clients:

- vSphere Client: A locally installed program to connect remotely to the vCenter Server
- vSphere Web Client: A web program to connect remotely to the vCenter Server
- vSphere CLI: Includes commands like `vmkfstools`, a tool to manage virtual disks and physical storage on an ESXi server, and `vmware-cmd`, a command used to perform virtual machine operations remotely, like starting or stopping a virtual machine
- vSphere PowerCLI: Based on Microsoft PowerShell, provides cmdlets to create, manage, and monitor virtual machines
- vSphere Web Client SDK: Provides a programmatic interface using web services to create customized tasks for your vSphere environment

## Creating a Virtual Machine from Installation Media

Using the vSphere client or web client, you can:

- Create a virtual machine
- Assign it a name
- Specify a guest operating system, for example, Oracle Linux (64-bit)

- Create a network connection
- Create a virtual disk from an already configured storage source (datastore). A datastore is storage space across multiple ESXi servers.

After creating the virtual machine, connect to an ISO image on your local disk and install the OS in the virtual machine.

### **Creating a Virtual Machine from a Template**

Using the vSphere client or web client, you can deploy a virtual machine from a template or clone an existing virtual machine. The new virtual machine inherits the software and configured properties present in the template or virtual machine. You can further customize your Linux guest during the template deployment or clone operation.

### **Paravirtualization with vSphere**

VMware offers these paravirtual drivers:

- PVSCSI, a paravirtual SCSI adapter
- VMXNET 3, a paravirtual network driver

You select PVSCSI adapters to achieve greater throughput and lower CPU utilization for storage operations. PVSCSI adapters yield the best performance in SAN environments.

VMXNET is a paravirtual network interface that is designed to reduce the I/O virtualization overhead and therefore increase performance.

In most cases, you obtain paravirtual drivers by installing VMware Tools in your virtual machine. VMware Tools is VMware's integration package for virtual machines.

# Microsoft Hyper-V and Windows Azure

- Hyper-V is a Windows Server role that turns the server into a type-1 virtualization provider.
- In addition to a UI, Hyper-V provides Windows Powershell cmdlets to manage the virtual environment.
- Linux Integration Services (LIS) provides integration between the OS running in the Hyper-V virtual machine.
  - The Hyper-V LIS package is already built in Oracle Linux 6.4 and newer releases.
- Windows Azure is Microsoft's cloud solution, based on Hyper-V.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Microsoft Hyper-V

Microsoft Hyper-V is a Windows server role that turns a Windows server, such as Windows Server 2012 R2 into a type-1 hypervisor. Hyper-V requires a 64-bit processor and hardware-assisted virtualization.

Manage Hyper-V with:

- Hyper-V Manager, a GUI tool
- A Hyper-V module for Windows Powershell, providing cmdlets for management tasks.

Hyper-V provides a software package called Linux Integration Services (LIS) that provides integration between the OS running in the Hyper-V virtual machine and the physical host. This package is already available in Oracle Linux, starting with Oracle Linux 6.4. In addition to timekeeping, virtual machine heartbeat detection, and integrated shutdown features, the package provides an information exchange capability between the running Linux virtual machine and the Hyper-V server. This functionality is similar to the messaging function available for Oracle Linux with the Oracle VM Guest Additions package.

## Windows Azure

Windows Azure is a cloud platform that allows you to quickly build, deploy, and manage scalable solutions.

As part of Windows Azure Compute services, you create virtual machines that use either Windows Server or the Linux operating system, including Oracle Linux. If the applications that you want to deploy run on Hyper-V, they also run on Windows Azure.

Windows Azure is based on a customized version of Microsoft Hyper-V called the Windows Azure Hypervisor. This hypervisor provides the virtualization services. Windows Azure contains additional components that manage the storage and computing resources in the Microsoft datacenters that are hosting Windows Azure.

# Linux as a Guest OS with Microsoft Hyper-V and Windows Azure

- The Linux guest OS runs in an isolated environment called a child partition.
  - The parent partition (or Management OS) loads the hypervisor.
- Hyper-V offers a form of paravirtualization called Enlightened I/O.
  - Recent releases of many Linux distributions contain the paravirtual drivers for Hyper-V's Enlightened I/O.
- To build Linux virtual machines in Azure, access the Management Portal and select a template from the Image Gallery.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Hyper-V Partitions

With Hyper-V, a virtual guest OS runs in a child partition, which isolates the guest OS from other partitions on the physical host. The child partition is also called a virtual machine. The parent partition, also called the Management OS, loads the hypervisor and contains the virtualization stack and the virtualization tools. The virtualization stack in the parent partition has direct access to the hardware devices.

## Paravirtualization with Hyper-V

Hyper-V supports a form of paravirtualization called Enlightened I/O, for networking, storage, graphics, and other input devices. Enlightened I/O provides increased performance by bypassing a layer of emulated hardware. The guest OS must support Enlightened I/O. The drivers for Enlightened I/O are delivered directly in recent Linux distributions, including Oracle Linux, Red Hat, and CentOS. For other distributions, you can obtain the Enlightened I/O drivers by downloading the Linux Integration Services from the Microsoft Download Center at <http://www.microsoft.com/en-us/download/default.aspx>.

## How to Create an Oracle Linux Virtual Machine with Windows Azure

After signing up for a particular program (for example, the Compute services program), create a virtual machine by using an image from the Image Gallery in the Windows Azure Management portal. Or you can upload a .vhd disk image as a file to Windows Azure. This vhd-type disk must already contain a bootable OS.

You can create stand-alone virtual machines or place your virtual machines in the same cloud service to allow the virtual machines to communicate or to provide load balancing to your applications.

# Linux as a Virtualization Provider

- With Linux Containers (LXC)
  - Containers allow you to run multiple user-space versions of Linux on the same host without the need of a hypervisor.
- With KVM, your Linux host becomes a type-1/2 hypervisor.
  - Guests run as ordinary user-space processes.
  - Guests are hardware accelerated and fully virtualized.
- KVM requires hardware virtualization extensions on Intel or AMD physical platforms.
- Use `libvirt` API and tools to manage KVM guests.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

With Oracle Linux, there are two ways to configure virtual environments, Linux Containers (LXC) and KVM.

## Linux Containers (LXC)

LXC is a virtualization technology that allows you to partition system resources on the control host into virtual instances called containers, which have their own process and network space. This technology provides isolation for the application(s) running in the container, while allowing resource adjustment to the container. The container environment is similar to a standard Linux OS but depends on the control host's kernel. LXC is covered in a separate lesson titled “Linux Containers (LXC).”

## KVM

KVM was first developed at Qumranet, which Red Hat bought in 2008. With KVM, your Oracle Linux host becomes a type-1/2 hypervisor. KVM requires hardware virtualization extensions on your Intel or AMD physical platform. KVM is a full virtualization solution, mainly for x86 hardware.

- Full virtualization means that the virtualization solution provides full emulation for the guest operating system running in the virtual machine including emulation for networking, interrupts and timers, and even provides an emulated BIOS.
- With full virtualization, you can run unmodified versions of the guest operating system in the virtual machine, either Linux or Windows.

## libvirt

- libvirt is a toolkit to manage your virtualized environment.
- libvirt is not specific to KVM:
  - Other virtualization technologies such as LXC, Xen, and VirtualBox use libvirt.
- libvirt offers an API and tools such as:
  - virt-install: Create KVM guests
  - virsh: Perform operational tasks on KVM guests
  - virt-manager: GUI to manage KVM guests
- Guest metadata is stored in XML format.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

KVM uses libvirt, an API and toolkit, to manage your virtualized environment. The libvirt toolkit can interact with several virtualizers, including these hypervisors:

- KVM/QEMU
- Xen hypervisor
- Linux Container System (LXC)
- VirtualBox hypervisor
- VMware ESX hypervisor

Find the complete list at <http://libvirt.org/>.

libvirt provides local and remote management of virtual machines. For secure remote management, you can use libvirt with TLS encryption and x509 certificates. For authentication, you can select Kerberos and Simple Authentication and Security Layer (SASL). SASL allows applications to exchange information securely.

The libvirt toolkit provides the tools and APIs to manage virtual machines: provision, start, stop, modify, migrate, monitor, and delete.

With libvirt tools, the virtual machine or domain metadata is described using the XML format. By default, the virtual machine XML configuration files reside in the /etc/libvirt/qemu directory.

## Installing KVM and libvirt

- The KVM modules are present in mainline Linux, as of 2.6.20.
    - With Oracle and Red Hat Linux, KVM is already available.
  - Get started by installing the following libvirt packages:
- ```
# yum install qemu-kvm qemu-img virt-manager libvirt libvirt-python python-virtinst libvirt-client
```
- Or if installing Oracle Linux:
    - Select Server with GUI as the Base Environment from the Software Selection.
    - Select Virtualization Client, Virtualization Hypervisor, and Virtualization Tools from the Add-Ons list.
    - This environment provides a full graphical user interface, which is required if you want to use the `virt-manager` application.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

KVM support is already present in Red Hat and Oracle Linux. If using other distributions, consult the documentation for a list of packages to install for KVM support.

To verify that `kvm` is present, use the `lsmod` command:

```
# lsmod | grep kvm
kvm_amd           51654    4
kvm               432586    1 kvm_amd
```

For administering your KVM guests, install the `libvirt` tools.

### Installing the Packages Separately

The selection of packages to install depends on what you plan to do. Install the following packages to get started with KVM:

```
# yum install qemu-kvm qemu-img virt-manager libvirt libvirt-
python python-virtinst libvirt-client
```

- `qemu-kvm`: Provides the user-level KVM emulator
- `qemu-img`: Provides the disk image manager
- `virt-install`: Provides the `virt-install` command for creating virtual machines

- `libvirt`: Provides the server and host side libraries for interacting with hypervisors and host systems
- `virt-manager`: Provides a graphical tool for administering virtual machines
- `libvirt-python`: Contains a module that permits applications written in the Python programming language to use the interface supplied by the `libvirt` API
- `libvirt-client`: Provides the client-side APIs and libraries for accessing `libvirt` servers. This package includes the `virsh` command line tool.

## Using the Software Groups

You can use the `yum` command to install the virtualization software groups to your existing Linux host:

- Virtualization Client: Clients for installing and managing virtualization instances
- Virtualization Platform: Provides an interface for accessing and controlling virtualized guests and containers
- Virtualization Hypervisor: Smallest possible virtualization host installation
- Virtualization Tools: Tools for offline and live virtual image management

Example:

```
# yum groupinstall "Virtualization Client"
```

## Adding Virtualization Support When Installing Linux

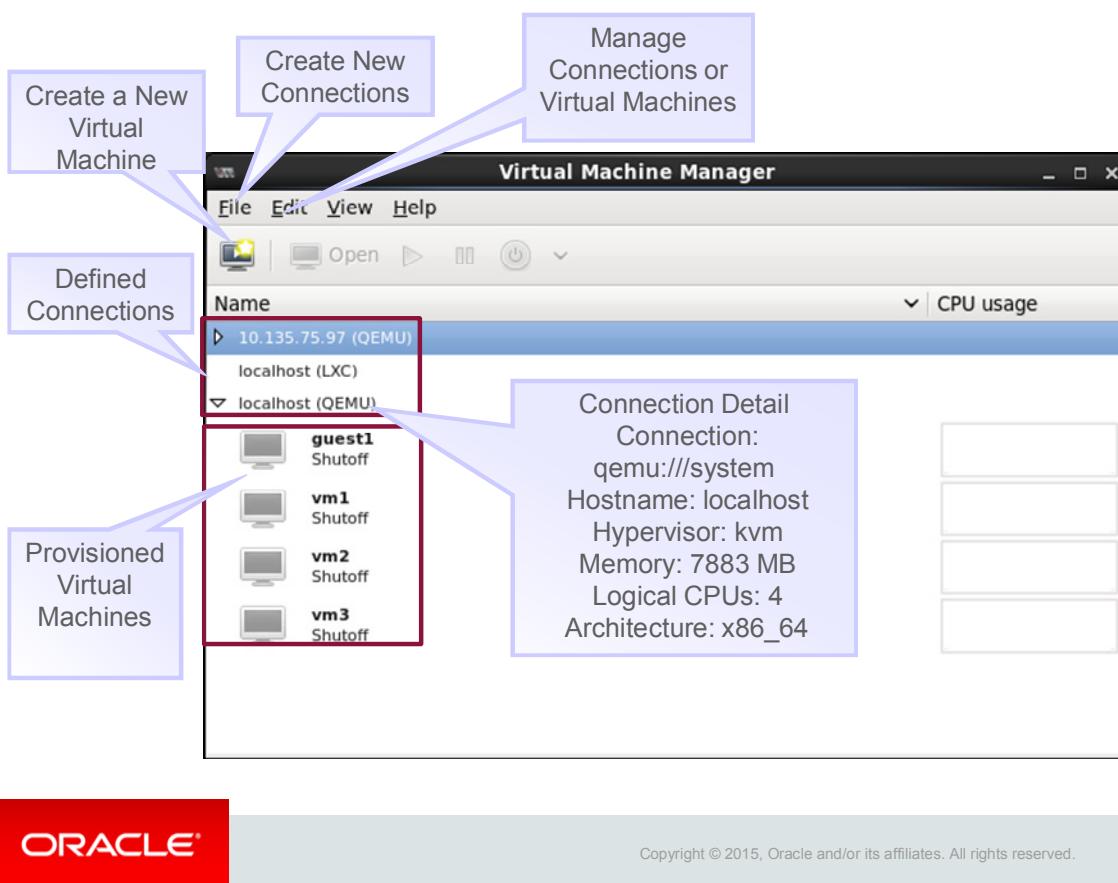
If installing Oracle Linux:

- Select Server with GUI as the Base Environment from the Software Selection
- Select Virtualization Client, Virtualization Hypervisor, and Virtualization Tools from the Add-Ons list

This provides an environment with full graphical user interface, which is required if you plan to use the `virt-manager` graphical user interface.

Alternatively, select Virtualization Host as the Base Environment from the Software Selection during installation, and select Virtualization Platform from the Add-Ons list. This provides a minimal virtualization server.

# Getting Started with virt-manager: Connections



Use the `virt-manager` command to launch the graphical user interface.

Because `virt-manager` supports several types of hypervisors, you must create one or more connections to specify which hypervisor to use for your virtual machines.

You can access the hypervisor locally or remotely.

## Creating a Connection

When creating a connection to the KVM hypervisor, specify the hypervisor as QEMU/KVM.

**Note:** QEMU is used with KVM to provide emulation for components like a NIC, disk device, or graphics adapter.

If accessing the hypervisor remotely, you must specify the necessary `ssh` parameters for the connection to succeed.

## Creating a Virtual Machine

After creating the connection, you can provision virtual machines for that connection by using one of the following two methods:

- Click the “Create a new virtual machine” icon, located in the toolbar.
- Highlight the target connection and select New from the shortcut menu.

You can also create new virtual machines with the `virsh` or `virt-install` commands. If you do not specify parameters with the `virsh` command, you start an interactive session:

```
# virsh
Welcome to virsh, the virtualization interactive terminal.

Type: 'help' for help with commands
      'quit' to quit
virsh # version
Compiled against library: libvirt 1.2.8
Using library: libvirt 1.2.8
Using API: QEMU 1.2.8
Running hypervisor: QEMU 1.5.3
virsh # quit
```

Example of creating a virtual machine with `virt-install`:

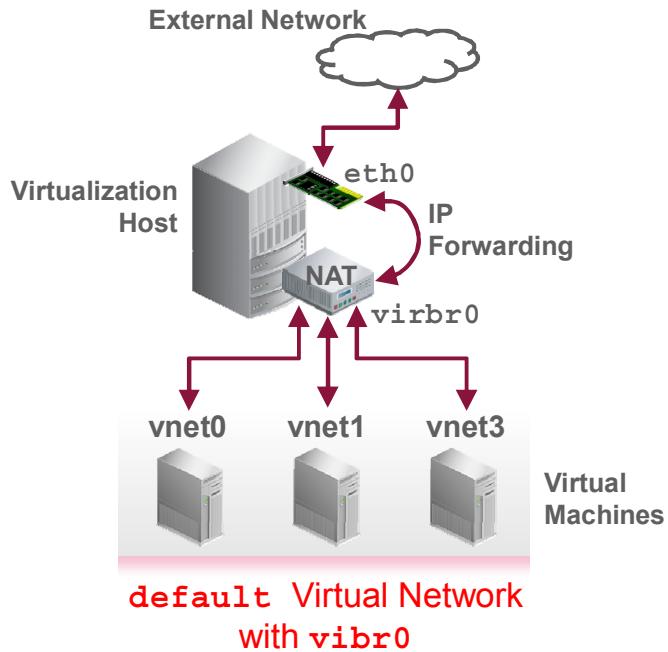
```
# virt-install --connect qemu:///system --virt-type kvm --name test --ram 500 --disk path=/var/lib/libvirt/images/test.img,size=4 --cdrom /home/user01/Downloads/OracleLinux-R7-U1-Server-x86_64-dvd.iso --os-variant oel7
```

```
Starting install...
Allocating 'test.img'
| 4.0 GB    00:00
Creating domain...
| 0 B        00:00
Connected to domain test
Escape character is ^]
```

```
Google, Inc.
Serial Graphics Adapter 10/14/11
...
```

# Virtual Networks

- A virtual network acts like a virtual network switch.
- The default virtual network is created when the libvirt daemon is started for the first time.
- Guest NICs connect to the virtual network `virbr0` bridge.
- Use `virt-manager` or `virsh net-*` commands to manage virtual networks.



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Before creating your virtual guests, you must create virtual networks to enable networking for the guests.

## Virtual Network Using NAT and IP Forwarding

Virtual machines connect to virtual networks. When the `libvирtd` daemon starts for the first time, it defines a virtual network called `default`. This virtual network uses a virtual network switch, or bridge, called `virbr0`, to which the guests attach.

The diagram in the slide shows the virtual interfaces, `vnet0`, `vnet1`, and `vnet3`, connecting to the virtual switch, `virbr0`, which is a bridge. These virtual interfaces are the back-end NICs created on the virtualization host, and they correspond to virtual network interfaces in the virtual guests. The `virbr0` bridge is not attached to any physical NIC on the virtualization host. Instead, NAT and IP forwarding are used to forward packets from the virtual guests to the external network. You can find more information about virtual networking at this site: <http://wiki.libvirt.org/page/VirtualNetworking>.

## Virtual Network in Routed Mode

In addition to virtual networks using NAT and IP forwarding, you can set up a virtual network that uses a virtual switch (or bridge) in routed mode, where the virtual switch is connected to a physical NIC on the virtualization host.

## Private Virtual Network

If virtual guests running on a virtualization host need to communicate with each other but do not require a connection to an external network, you can create a virtual network in private or isolated mode. This type of network still allows virtual guests to acquire an IP address but does not support traffic into or out of the virtualization host.

# Working with Storage

- Use libvirt tools to create storage pools and allocate volumes for virtual guests.
  - Storage pools can be iSCSI, NFS, disk devices, file systems or directories, or even LVM Volume Group.
  - Volumes support several format types, including raw, qcow2, and vmdk.
- You can also assign storage outside of storage pools.
  - Ensure that the storage (for example, an NFS share) is available when the virtual guest boots.
- Consider using virtio, which offers a paravirtualized driver to accelerate disk operations.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You do not need to create storage pools to assign storage to your virtual guests. If you use storage pools, the libvirt tools ensure that the assigned storage residing in a storage pool is available when you attempt to boot your virtual guest.

libvirt supports several storage pool types. A few of these types are described here:

- Directory pool: You create volumes in the directory, selecting among several format types, such as qcow2, which is a file format used by QEMU. qcow2 uses a disk optimization scheme to delay storage allocation until needed. The vmdk file format listed in the slide is a file format developed by VMware.
- iSCSI pool: The storage pool is created on an existing iSCSI target.
- NFS pool: The storage pool is created on an existing NFS share.
- Disk device: You can use a disk device, such as a USB stick, as a storage pool. A volume is created in the storage pool by creating partitions on the disk. For example, you create a storage pool with device /dev/sdb and create a volume as /dev/sdb1.

virtio is a paravirtualization standard for both storage and networking. The virtio paravirtualized drivers reduce I/O latency. The guest must support paravirtualization to use virtio.

To enable `virtio` operations, you specify `virtio` when allocating a virtual disk (or a virtual network interface), or you can change the model type for the virtual disk to `virtio` at a later time.

For example, if you examine XML file `vm3.xml` in `/etc/libvirt/qemu` (on the virtualization host) that describes virtual guest `vm3`, you find the following disk section under the devices section:

```
<disk type='file' device='disk'>
    <driver name='qemu' type='raw' cache='none' />
    <source file='/var/lib/libvirt/images/vm3.img' />
    <target dev='vda' bus='virtio' />
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06'
              function='0x0' />
</disk>
```

Note that `virtio` is selected as the bus type.

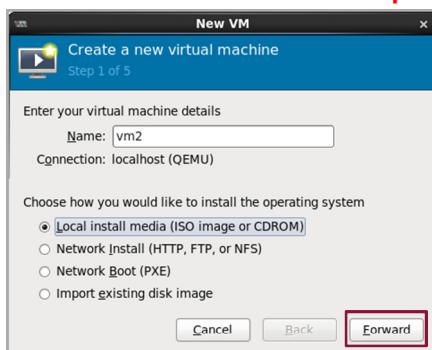
# Creating Virtual Machines

- Using `virt-install`

```
virt-install \
--name=vm1 \
--file=/var/lib/libvirt/images/vm1.dsk \
--file-size=8 --nonparse
...
--network bridge=br0
--os-type=linux --os-variant=oe16
```

- Using `virt-manager`

Step 1



Step 5



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `virt-install` command from the command line or in a script to create virtual machines. You need root privileges to use the `virt-install` command.

Make sure that you use the appropriate `os-type` and `os-variant` for the OS in your virtual machines. Use the following command to list the `os-variant`:

```
# osinfo-query os
```

You can create a virtual machine using the following installation methods:

- ISO image or CDROM
- Network install: You must make the installation tree accessible by using HTTP, FTP, or NFS.
- PXE network install: For this installation method to succeed, the virtual machine must be able to acquire an IP address.
- Using an existing disk image: The disk image must contain an already installed, bootable OS.

Hot plugging operations are allowed with KVM when also supported by the guest OS. This means that you can dynamically increase the number of virtual CPUs (vCPUs), or add virtual disks or network interfaces to your virtual machines. Virtual disks hotplug operations are supported for only some bus types.

# Managing the Life Cycle of a Virtual Machine

The following operations are supported for virtual machines:

- Pause
- Start and shutdown, including force shutdown
- Cloning
- Migration
- Delete



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Shutdown and Pause Operations

You can use the pause function before attempting to clone your virtual machine. Use the resume function to un-pause the virtual machine.

To shut down a virtual machine using a `libvirt` tool, the `acpid` daemon must be running in the guest OS. If not running, use force shutdown from `virt-manager` or shut down the virtual machine from its console.

## Cloning

The cloning operation copies the disk images of an existing machine to create a new virtual machine. It automatically assigns a new MAC address and updates the virtual machine unique ID called UUID. You can clone by using the `virt-manager` UI or use the `virt-clone` command. Pause the virtual machine before attempting to clone it.

## Migration

You can migrate a virtual machine to another virtualization host, if both the source and target hosts have the same architecture. You must also ensure that both hosts have access to the guest storage. Find more information about KVM migration at this location: <http://www.linux-kvm.org/page/Migration>

## Delete Operation

When you delete a virtual machine, you have the choice to retain the virtual machine's storage files.

## Quiz



What statements are true regarding paravirtualized drivers?

- a. They are idealized drivers installed in the virtual guest.
- b. They optimize I/O operations.
- c. They map operations to real drivers on the virtualization host.
- d. Most virtualization solutions offer them.
- e. All of the above



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe virtualization and its benefits
- Explain how Linux as a virtual guest supports the different virtualization modes
- Outline the support for Linux as a guest OS with various virtualization solutions
- Describe the KVM hypervisor
- Use the libvirt tools to create and manage KVM virtual guests



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 14: Overview

This practice covers the following topics:

- Preparing your virtualization host for KVM operations
- Starting the Virtual Machine Manager
- Creating a virtual machine with an existing virtual disk
- Accessing the virtual machine's console
- Cloning the virtual machine's disk
- Adding paravirtualization to the virtual machine's NIC operations



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Linux Containers (LXC)

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the purpose of Linux Containers
- Describe container configuration parameters
- Set up a system for Linux Containers
- Describe Linux Container template scripts
- Create a Linux Container by using the Oracle template script
- Use Linux Container utilities to start and stop a container
- Use additional Linux Container utilities



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Linux Containers: Introduction

- Linux Containers (LXC) are the next step up from cgroups.
- LXC provides application and operating system isolation.
- Containers do not require a hypervisor.
- They are similar to Oracle Solaris Zones in that:
  - They both provide virtualization at the application level
  - One kernel is shared by many zones or containers
- Containers rely on the cgroups functionality.
- Containers rely on namespace isolation that is similar to chroot.
- Processes within a container can have their own process ID space, file system structure, and network interfaces.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Linux Containers (LXC) are the next step up from cgroups for using system resources more efficiently. Whereas cgroups allow you to isolate system resources, containers provide application and operating-system isolation. Containers allow you to run multiple user-space versions of Linux on the same host without the need of a hypervisor. You can isolate environments and control how resources are allocated without the virtualization overhead.

LXC are similar to Oracle Solaris Zones in that they are virtualization at the application level, above the kernel. One operating system kernel is shared by many zones or containers. Because the kernel is shared, you are limited to the modules and drivers that it has loaded. The difference between zones and containers is more at the implementation level and in the way it is integrated into the operating system.

Containers rely on the cgroup's functionality but also rely on namespace isolation, similar to chroot. Within each container, processes can have their own private view of the operating system with its own process ID space, file system structure, and network interfaces.

Containers can be useful for:

- Running different copies of application configurations on the same server
- Running multiple versions of Oracle Linux on the same server
- Creating sandbox environments for testing and development
- Controlling the resources allocated to user environments

You can also use Btrfs subvolumes as a way to quickly create containers.

# Linux Container Resource Isolation

- Containers provide:
  - Resource management through control groups (cgroups)
  - Process isolation through namespaces
- A user-space container object provides resource isolation and control for an application or a system.
- Containers rely on a set of kernel functionalities to be active.
  - LXC is fully functional beginning with kernel 2.6.29.
- Use the `lxc-checkconfig` command to get information about your kernel configuration.
  - This utility reads `/proc/config.gz` or `/boot/config*`.
- Define the resources to isolate for an application.
- Running a system within a container is easier than an application.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Containers provide system resource management through control groups (cgroups) and process isolation through namespaces. They use these functionalities to provide a user-space container object that can then provide full resource isolation and resource control for an application or a system.

Containers rely on a set of kernel functionalities, such as namespaces, control groups, networking, and file capabilities, to be active. Beginning with kernel 2.6.29 or later, LXC is fully functional. Running with an older kernel version causes LXC to work with a restricted number of functionalities, or can even cause LXC to fail. You can use the `lxc-checkconfig` command to get information about your kernel configuration. This utility reads the `/proc/config.gz` file if it is found, or reads the `/boot/config*` file for your active kernel version and displays the status (enabled/disabled) of kernel functionalities.

Before running an application in a container, identify the resources to isolate. By default, the process IDs, the SysV IPCs, and the mount points are isolated. With the default configuration, you can run a simple shell command within a container. When running an application, for example `sshd`, provide a new network stack and a new host name. To avoid container conflicts, specify a `root` file system for the container. Running a system in a container is easier than running an application because you do not care about specific resource isolation when running a system. Everything is isolated when running a system.

The following is sample output from the `lxc-checkconfig` command:

```
# lxc-checkconfig
Kernel configuration not found at /proc/config.gz; search...
Kernel configuration found at /boot/config-3.8.13-55.1.6...
--- Namespaces ---
Namespaces: enabled
Utsname namespaces: enabled
Ipc namespaces: enabled
Pid namespaces: enabled
User namespaces: missing
Network namespaces: enabled
Multiple /dev/pts instances: enabled

--- Control groups ---
Cgroup: enabled
Cgroup clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
File capabilities: enabled
...
```

Notice that the UEK R3 (3.8.13) does not have the necessary support for user namespaces. Without this support, you cannot set `lxc.id_map` entries, which allow you to run a container under another UID instead of `root`. Support for user namespaces is available beginning with kernel version 3.13.

# Linux Container Configuration File

Create a file to define the system resources for the container.

- **Architecture and Utsname:**
  - Architecture and host name for the container
- **Network:**
  - Virtual network interfaces
- **TTY:**
  - Pseudo tty, console output, and number of available ttys
- **Mount points:**
  - Different places to be mounted
- **Root file system:**
  - Root file system for the container
- **Control groups:**
  - Configuration for the different cgroup subsystems



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Creating a container defines a set of system resources to be virtualized and isolated by a process that uses the container. As mentioned, PIDs, SysV IPCs, and mount points are virtualized and isolated by default. Other system resources are shared across containers until they are explicitly defined in the Linux Container configuration file. For example, if a network is defined in the configuration file, a network stack is created for the container. Otherwise, the container shares the same network stack with the host that creates the container.

Following is a partial list of system resources that can be defined for a container. See the `lxc.container.conf (5)` man page for complete list and descriptions:

- **Architecture** – Specifies the architecture for the container
- **Utsname** – Specifies the host name for the container
- **Network** – Specifies how the network is virtualized in the container. Network virtualization works at layer 2. You can define multiple virtual network interfaces.
- **TTY** – Specifies the pseudo tty, console output, and available ttys
- **Mount points** – Specifies the different places to be mounted
- **Root file system** – Specifies the root file system for the container, which can be different from the root file system for the host system
- **Control groups** – Specifies the configuration for the different cgroup subsystems

You can define the following configuration keys. Define each key on a separate line using the `key = value` format.

### **Architecture / Utsname**

- `lxc.arch`: Specifies the architecture for the container. Valid values are `x86`, `i686`, `x86_64`, and `amd64`.
- `lxc.utsname`: Specifies the host name for the container

### **Network**

- `lxc.network.type`: Specifies the type of network virtualization to be used for the container. Valid values are `none`, `empty`, `veth`, `vlan`, `macvlan`, and `phys`.
- `lxc.network.flags`: Specifies an action to do for the network. A value of `up` activates the network.
- `lxc.network.link`: Specifies the interface to be used for the real network traffic
- `lxc.network.name`: Specifies the network interface in the container
- `lxc.network.hwaddr`: Specifies the MAC address of the container's network interface
- `lxc.network.ipv4`: Specifies the IPv4 address for the virtualized interface. You also specify the broadcast address on the same line, immediately after the IPv4 address.
- `lxc.network.ipv4.gateway`: Specifies the IPv4 address to use as the gateway in the container
- `lxc.network.ipv6`: Specifies the IPv6 address for the virtualized interface
- `lxc.network.ipv6.gateway`: Specifies the IPv6 address to use as the gateway in the container
- `lxc.network.script.up`: Specifies a script to be executed after creating and configuring the network used from the host side

### **TTY**

- `lxc.pts`: Specifies the number of pseudo ttys allowed for a `pts` instance
- `lxc.console`: Specifies a path to a file where the console output is written
- `lxc.tty`: Specifies the number of ttys available to the container

### **Mount Points**

- `lxc.mount`: Specifies a file location `fstab` format that contains the mount information
- `lxc.mount.entry`: Specifies a mount point corresponding to a line in the `fstab` format

### **Root File System**

- `lxc.rootfs`: Specifies the `root` file system for the container. If it is not specified, the container shares its `root` file system with the host system.
- `lxc.rootfs.mount`: Specifies where to recursively bind `lxc.rootfs` before pivoting. This ensures success of the `pivot_root(8)` system call.
- `lxc.pivotdir`: Specifies where to pivot the original root file system under `lxc.rootfs`. The default is `/mnt`.

### **Control Groups**

- `lxc.cgroup.[subsystem name]`: Specifies the control group subsystem parameter to be set. An example is `lxc.cgroup.cpuset.cpus`.

Example configuration files are in the `/usr/share/doc/lxc/examples` directory.

# Setting up a System for Linux Containers

- Install the LXC packages:

```
# yum install lxc
```

- The libvirt and lxc-libs packages are installed as dependencies.
  - The /container directory is automatically created.

- The LXC template scripts create containers in /container.
  - Edit the script if you want to use a different directory.
- Mount a Btrfs file system on /container to take advantage of the Btrfs snapshot feature. For example:

```
# mkfs.btrfs /dev/xvdd
# mount /dev/xvdd /container
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The LXC package is required to create and use Linux Containers. Installing this package also installs the libvirt and lxc-libs packages as dependencies and creates the /container/ directory. Install the lxc package as follows:

```
# yum install lxc
```

The Btrfs file system can be used for a container repository. New instances can then be cloned and spawned quickly, without requiring significant additional disk space. Btrfs allows you to create a subvolume that contains the base template for the containers, and to create containers from writable snapshots of the template.

The following example creates a Btrfs file system on /dev/xvdd and mounts the file system on /container:

```
# mkfs.btrfs /dev/xvdd
# mount /dev/xvdd /container
```

# Linux Container Template Scripts

- Template scripts define the settings for the system resources that are assigned to a container.
- Several template scripts are available:

```
# ls /usr/share/lxc/templates
lxc-alpine    lxc-centos   lxc-fedora   lxc-oracle   ...
...
```

- Configuration settings for the container are stored in the /container/<name>/config file.

```
# lxc-create -n ol-test -t oracle
# ls /container/ol-test
config      rootfs
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Template scripts define the settings for the different system resources that are assigned to a running container. Each template script defines different resources but examples include the container host name, network configuration, mount points, the root file system, number of available ttys, and other settings. The following is a list of the current template scripts:

```
# ls /usr/share/lxc/templates
lxc-alpine    lxc-centos   lxc-fedora   lxc-oracle   ...
lxc-altlinux  lxc-cirros   lxc-gentoo   lxc-plamo
lxc-archlinux lxc-debian   lxc-openmandriva lxc-sshd
lxc-busybox   lxc-download lxc-opensuse  lxc-ubuntu
```

Each template script is used to generate a specific type of Linux Container. The `lxc-altlinux` script is for generating an ALT Linux Container, the `lxc-busybox` script is for generating a BusyBox container, the `lxc-oracle` script is for generating an Oracle Linux Container, and so on.

Use the short name (omit the `lxc-`) when referring to the template script. The following example uses the `lxc-create` command to create a container named “ol-test” from the `lxc-oracle` template script:

```
# lxc-create -n ol-test -t oracle
```

## lxc-create Utility

- Use the `lxc-create` command to create a container. The syntax is:

```
lxc-create -n name [-f config] [-t template] [-B
<backing-store>] [-- template-options]
```

- The command creates an object directory that defines the resources that the container can use or can see:

```
/container/<name>/
```

- A configuration file is optional. The default isolation is:
  - Processes
  - SysV inter-process communication (IPC)
  - Mount points
- Template scripts in the `/usr/share/lxc/templates` directory also provide configuration information.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `lxc-create` command to create a container. This command creates a system object directory in `/container`, which stores configuration and user information. The object provides a definition of the different resources that an application can use or can see. The syntax is:

```
lxc-create -n <container_name> [-f <config_file>] [-t
<template>] [-B <backing-store>] [-- template-options]
```

The options are described as follows:

- `-n <container_name>` – Specifies the name of the container. This name becomes the name of the system object directory, `/container/<container_name>`.
- `-f <config_file>` – Specifies the file used to configure the virtualization and isolation functionality of the container. If a configuration file is not specified, the container is created with the default isolation: processes, SysV IPC, and mount points.
- `-t <template>` – Specifies the short name of an existing template script. Template scripts are located in `/usr/share/lxc/templates`. Configuration information is provided in these template scripts.
- `-B <backing-store>` – Specifies either `dir`, `btrfs`, `lvm`, `loop`, or `best` as the backing-store
- `-- template-options` – Specifies arguments to pass to the template

The `-B <backing-store>` options are described further.

- `dir`: This is the default. The container root file system is a directory under `/container/<container_name>/rootfs`.
- `btrfs`: If `/container` is a Btrfs file system, the `rootfs` directory is a subvolume when it is created by `lxc-create`. For example:  

```
# btrfs sub list /container
ID 257 gen 12 top level 5 path ol-test/rootfs
```
- `lvm`: This specifies to use an LVM block device. Additional options are available when using the `lvm` parameter.
  - `--lvname lvname1`: Specifies creation of a logical volume named `lvname1` rather than the container name, which is the default logical volume
  - `--vgname vgname1`: Specifies creation of the logical volume in the volume group `vgname1` rather than `lxc`, which is the default volume group
  - `--thin-pool thinpool1`: Specifies creation of the logical volume as a thin-provisioned volume in the pool named `thinpool1` rather than `lxc`, which is the default
  - `--fstype FSTYPE`: Specifies creation of an `FSTYPE` file system on the logical volume, rather than `ext4`, which is the default file system type
  - `--fssize SIZE`: Specifies creation of a logical volume and file system of `SIZE` rather than the default size of `1G`
- `best`: This specifies to use, in order, `btrfs`, `zfs`, `lvm`, and finally a directory backing store.

## lxc-oracle Template Options

- To view options supported by the `lxc-oracle` template:
 

```
# lxc-create -t oracle -h
```

  - Options are displayed in the notes.
- In decreasing order of preference, the recommended methods for creating a new container are:
  1. Use `-R` (optionally with `-u` to specify an internal mirror of Public Yum for updates using its hierarchical directory structure).
  2. Use `--baseurl` to install from a local single-release Yum repository containing the specific RPMs to use for install (optionally with `-u`, as above).
  3. Use `-P` (inplace patch) or `-t` (copy and patch).



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the following command to display the list of options supported by the template:

```
lxc-create -t <template> -h
```

Use the short name when requesting template options. For example, to display options supported by the `lxc-oracle` template:

```
# lxc-create -t oracle -h
...
-a | --arch=<arch>          architecture (ie. i386, x86_64)
-R | --release=<release>    release to download for the new ...
--rootfs=<path>             rootfs path
-r | --rpms=<rpm name>      additional rpms to install into ...
-u | --url=<url>             replace yum repo url (ie Oracle ...
                             use package repository (file:///...
                             arch and release must also be ...
-t | --templatefs=<path>     copy/clone rootfs at path instead ...
-P | --patch=<path>          only patch the rootfs at path for ...
...
...
```

## Using the lxc-oracle Template

- This template simplifies the creation of Oracle Linux containers.
- This template creates the container by using packages from Public Yum, from a local Yum repository, or from an existing rootfs.
- To create an Oracle Linux 6.5 container with i386 packages:

```
# lxc-create -n ol65-32 -B btrfs -t oracle -- -a i386 -R
6.5
```

- To create the latest Oracle Linux 7 container with x86\_64 packages:

```
# lxc-create -n ol7L-64 -B btrfs -t oracle -- -a x86_64
-R 7.latest
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The lxc-oracle template supports creating an Oracle Linux container using packages from Public Yum, or from a local Yum repository, to create a new rootfs. In addition, this template can create a container based on an existing root file system.

The following example uses the lxc-oracle template to create an Oracle Linux 6.5 container named ol65-32 from public-yum with i386 packages:

```
# lxc-create -n ol65-32 -B btrfs -t oracle -- -a i386 -R 6.5
```

To use the template and to create the latest Oracle Linux version 7 container from Public Yum with x86\_64 packages:

```
# lxc-create -n ol7L-64 -B btrfs -t oracle -- -a x86_64 -R
7.latest
```

The following example creates an Oracle Linux container from an existing rootfs. The container rootfs is a Btrfs snapshot if /path/to/rootfs is a Btrfs subvolume on the same Btrfs file system.

```
# lxc-create -n ol7L-64 -B btrfs -t oracle -- -t /path/to/rootfs
```

If you do not specify template options, the container defaults to the same architecture and Oracle Linux version as the host.

Each of the examples specifies using btrfs as the backing-store.

Create a container named `ol-test` using the `lxc-oracle` template. If you do not specify template options, the container defaults to the same architecture and Oracle Linux version as the host.

```
# lxc-create -n ol-test -B btrfs -t oracle
...
Host is OracleServer 7.1
Create configuration file /container/ol-test/config
Yum installing release 7.1 for x86_64
...
Transaction Summary
=====
...
Complete!
Rebuilding rpm database
Patching container rootfs /container/ol-test/rootfs for ...
Configuring container for Oracle Linux 7.1
Added container user:oracle password:oracle
Added container user:root password:root
Container : /container/ol-test/rootfs
Config     : /container/ol-test/config
Network    : eth0 () on virbr0
```

As you can see from the output, the `lxc-oracle` template creates an `oracle` user. It sets the initial `root` and `oracle` user passwords for the container and prints them out so that you can log in.

The `lxc-oracle` template uses the `virbr0` bridge set up by `libvирtd`. The following shows the default network configuration:

```
# grep network /container/ol-test/config
lxc.network.type = veth
lxc.network.link = virbr0
lxc.network.name = eth0
lxc.network.mtu = 1500
lxc.network.hwaddr = ...
...
```

The `veth` type of network virtualization is a peer network device with one side assigned to the container and the other side attached to a bridge specified by `lxc.network.link`.

# Starting and Stopping a Container

- There are two ways to run an application or a system in a container:

```
lxc-start -n name -f configfile cmd
lxc-execute -n name -f configfile -- cmd
```

- The `lxc-execute` command runs the specified command using an intermediate process, `/usr/sbin/init.lxc`.
  - `init.lxc` has a PID of 1. The first process of the application has a PID of 2.
- The `lxc-start` command runs the specified command directly in the container.
  - The PID of the first process is 1. If no command is specified, `lxc-start` runs `/sbin/init`.
- Use `lxc-stop` to kill all the processes in the container.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `lxc-start` or `lxc-execute` command to run an application or a system inside a container. If you did not create the container before starting the application, the container uses the configuration file that is passed as an argument to the command. If there is no configuration file, the container uses the default isolation. The syntax for both `lxc-start` and `lxc-execute` is similar; they both accept a command to run within the container as an argument:

```
lxc-start -n <container_name> ... [command]
lxc-execute -n <container_name> ... [-- command]
```

The `lxc-execute` command runs the specified command using an intermediate process, `/usr/sbin/init.lxc`. In the container, `init.lxc` has a PID of 1 and the first process of the application has a PID of 2.

The `lxc-start` command runs the specified command directly in the container. The PID of the first process is 1. If no command is specified, `lxc-start` runs `/sbin/init`.

Therefore, `lxc-execute` is best suited for running an application within a container, whereas `lxc-start` is best suited for running a system in the container.

When the application stops, the container is also stopped. You can also use the `lxc-stop` command to kill all the processes in the container.

## lxc-start Utility

- Is mainly used to run a system container. The syntax is:

```
lxc-start -n <name> [-f <config_file>] [-c
    <console_device>] [-d] [-s <KEY=VAL>] [<command>]
```

- Runs the specified [<command>] in the container
- If no command is specified, uses the default /sbin/init to run a system container.
  - -d: Runs the container as a daemon
  - -f: Specifies a configuration file that overrides a container created using lxc-create
  - -c: Specifies a device to use for the container's console
  - -s <KEY=VAL>: Overrides configuration settings



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

System containers emulate an entire Linux OS booting from scratch. They provide their own init program, so `init.lxc` is not needed. Use `lxc-start` for system containers and use `lxc-execute` for running a single program, which generally shares the rootfs with the host system. The complete syntax of `lxc-start` is as follows:

```
lxc-start -n <name> [-f <config_file>] [-c <console_device>]
    [-d] [-s <KEY=VAL>] [<command>]
```

This command runs the specified [<command>] inside the container, which is identified by the `-n <name>` argument. If no [<command>] is specified, `lxc-start` uses the default `/sbin/init` to run a system container.

If the container does not exist, that is, it was not already created by the `lxc-create` command, the container is set up as defined in `<config_file>`. If no configuration is defined, the default isolation (processes, SysV IPC, and mount points) is used.

The remaining options are described as follows:

- -d – Specifies to run the container as a daemon. If an error occurs, nothing is displayed.
- -c <console\_device> – Specifies a device to use for the container's console. If not specified, the current terminal is used unless the `-d` option is specified.
- -s <KEY=VAL> – Assigns a value (VAL) to a configuration key (KEY). This overrides assignments in the configuration file.

## lxc-execute Utility

- Is mainly used to run an application in an isolated environment. The syntax is:

```
lxc-execute -n <name> [-f <config_file>] [-s <KEY=VAL>]
[-- <command>]
```

- Runs the specified [*<command>*] inside the container via an intermediate process, /usr/sbin/init.lxc
  - The init.lxc process had a PID of 1.
  - The init.lxc process waits for the specified command and all child processes to end before init.lxc ends.
- Example:

```
# lxc-execute -n ol-test -f
/usr/share/doc/lxc/examples/lxc-macvlan.conf
/bin/bash
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The complete syntax of `lxc-execute` is as follows:

```
lxc-execute -n <name> [-f <config_file>] [-s <KEY=VAL>]
[-- <command>]
```

This command runs the specified [*<command>*] inside the container via an intermediate process, /usr/sbin/init.lxc. The init.lxc process has a PID of 1, is executed when `lxc-execute` is run, and invokes the command that you pass to `lxc-execute`.

After launching the specified command, init.lxc waits for the command and all the child processes to end before it ends. This allows it to support daemons inside the container. The options are similar to `lxc-start`, except that there is no `-c console_device` option and no option to run the specified command as a daemon using `-d`.

The following example launches the Bash shell inside with a predefined configuration file. You do not need to pre-create the `test` container with `lxc-create`, you can just use `lxc-execute` to run it.

```
# lxc-execute -n ol-test -f /usr/share/doc/lxc/examples/lxc-
macvlan.conf /bin/bash
```

## lxc-attach Utility

- Is used to start a process inside a running container. The syntax is:

```
lxc-attach -n <name> [-a <arch>] [-e] [-s <namespaces>]
[-R] [--keep-env] [--clear-env] [-- <command>]
```

- Runs the specified [<command>] inside the container.
  - The container must be currently running.
  - If no command is specified, the default shell of the user running lxc-attach is executed inside the container.
- Examples:

```
# lxc-attach -n ol-test -- ps -ef
# lxc-attach -n ol-test -- systemctl start sshd
# lxc-attach -n ol-test -- ip link delete eth1
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The complete syntax of lxc-attach is as follows:

```
lxc-attach -n <name> [-a <arch>] [-e] [-s <namespaces>] [-R] [--keep-env] [--clear-env] [-- <command>]
```

This command starts the specified [<command>] inside a running container. If no command is specified, the default shell of the user running the lxc-attach command is executed.

The options are described as follows:

- a <arch> – Specifies the architecture that the kernel must run
- e – Specifies that the new process not be added to the container's cgroups and not drop its capabilities before executing
- s <namespaces> – Specifies the namespaces to attach to, as a pipe-separated list (for example, NETWORK|IPC|MOUNT|PID|UTSNAME)
- R – Specifies that /proc and /sys be remounted to reflect namespace contexts
- keep-env|--clear-env – Specifies to either keep or clear the current environment for attached programs

The following example starts the sshd service in the container:

```
# lxc-attach -n ol-test -- systemctl start sshd
```

## lxc-ls and lxc-info Utilities

- Use the `lxc-ls` command to display the containers that the host system is running.

```
# lxc-ls  
ol-test
```

- Use the `lxc-info` command to display information for a container.

```
# lxc-info -n ol-test  
Name:          ol-test  
State:         RUNNING  
PID:           7451  
IP:            192.168.122.175  
CPU use:       1.39 seconds  
BlkIO use:     1.69 MiB  
Memory use:   26.87 MiB  
...
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `lxc-ls` command to display the containers that the host system is running. The command accepts the options of the `ls` command. For example:

```
# lxc-ls -l  
drwxrwx---. 1 root root <date_time> ol-test
```

Use the `lxc-info` command to display information about a container.

```
# lxc-info -n ol-test  
Name:          ol-test  
State:         RUNNING  
PID:           7451  
IP:            192.168.122.175  
CPU use:       1.39 seconds  
BlkIO use:     1.69 MiB  
...
```

There are options to the `lxc-info` command to display specific information, such as just print the container's state (`-s`), or just print the container's IP address (`-i`).

## lxc-console Utility

- Container must be in RUNNING state and must be configured with ttys:
  - /sbin/agetty processes are running in the container.
- Use the lxc-console command to access the container through the ttys. The syntax is:

```
lxc-console -n <name> [-t <ttynum>]
```

- Specify the number of ttys available to the container in a configuration file:

```
lxc.tty = #
```

- To exit an lxc-console session, press Ctrl + A and then Q.
- You can also use ssh to log in if the container is running sshd and has an IP address.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

If the container is in RUNNING state and is configured with ttys (that is, /sbin/agetty processes have started running in the container), you can use the lxc-console command to access the container through the ttys. The syntax of lxc-console is as follows:

```
lxc-console -n <name> [-t <ttynum>]
```

You can optionally specify the tty number to connect. If it is not specified, a tty number is automatically selected by the container. You are prompted for a login and a password:

```
# lxc-console -n ol-test
ol-test login: root
password:
```

Use the following configuration entry to define the number of ttys available:

```
lxc.tty = #
```

To exit the lxc-console session, type Ctrl + A followed by Q.

You can also log in by using the ssh command if the container has an IP address assigned to the virtual network interface and the container has the sshd service running.

## lxc-freeze and lxc-unfreeze Utilities

- Use the `lxc-freeze` command to stop all processes that belong to a container.

```
# lxc-freeze -n ol-test
# lxc-info -n ol-test
Name:          ol-test
State:         FROZEN
PID:           4354
...
```

- Use the `lxc-unfreeze` command to resume all processes:

```
# lxc-unfreeze -n ol-test
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `lxc-freeze` command to stop all processes that belong to a container (for example, to accommodate job scheduling). This command puts all the processes in an uninterruptible state. The state changes to FROZEN.

```
# lxc-freeze -n ol-test
# lxc-info -n ol-test
Name:          ol-test
State:         FROZEN
PID:           4354
...
```

Use the `lxc-unfreeze` command to resume all processes.

```
# lxc-unfreeze -n ol-test
# lxc-info -n ol-test
Name:          ol-test
State:         RUNNING
...
```

## lxc-cgroup Utility

- When a container is started, a control group (cgroup) is created and associated with the container.
- Use the `lxc-cgroup` command to view and modify the cgroup subsystem parameters set in the container.
- To display a cgroup subsystem parameter:

```
# lxc-cgroup -n ol-test cpu.shares
```

- To set the value of a cgroup subsystem parameter:

```
# lxc-cgroup -n ol-test cpu.shares 500
```

- To set a persistent value, add the entry to the container's configuration file (or template script before `lxc-create`):

```
lxc.cgroup.cpu.shares = 500
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

When a container is started, a control group (cgroup) is created and associated with the container. Use the `lxc-cgroup` command to view and modify the cgroup subsystem parameters set in the container. To display the current value of a cgroup subsystem parameter, provide the parameter as an argument.

```
# lxc-cgroup -n ol-test cpu.shares  
1024
```

To set the value of a cgroup subsystem parameter, pass the parameter and the value as arguments.

```
# lxc-cgroup -n ol-test cpu.shares 500  
# lxc-cgroup -n ol-test cpu.shares  
500
```

To make the cgroup subsystem values persistent, add the settings to the container's configuration file or add the settings to the template script before creating the container.

```
lxc.cgroup.cpu.shares = 500  
lxc.cgroup.devices.allow = c 1:3 rwm
```

# Summary of Linux Container Utilities

- `lxc-autostart` – Starts, stops, or kills containers with the `lxc.start.auto` configuration parameter set
- `lxc-clone` – Clones a new container from an existing one
- `lxc-monitor` – Monitors the state of a container
- `lxc-snapshot` – Creates, lists, and restores container snapshots
- `lxc-unshare` – Runs a task in a cloned set of namespaces
- `lxc-usernsexec` – Runs a task as the `root` user in a new user namespace
- `lxc-wait` – Waits for a container to change to a specified state



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary of LXC utilities:

- `lxc-attach` – Starts a process inside a running container
- `lxc-autostart` – Processes containers with `lxc.start.auto` set
- `lxc-cgroup` – Sets or displays the value of a cgroup subsystem parameter
- `lxc-checkconfig` – Checks the current kernel for LXC support
- `lxc-clone` – Clones a new container from an existing one
- `lxc-config` – Queries LXC system configuration
- `lxc-console` – Logs in to a container
- `lxc-create` – Creates a persistent container object
- `lxc-destroy` – Destroys a container object
- `lxc-execute` – Runs an application inside a container
- `lxc-freeze` – Stops all the processes belonging to a container
- `lxc-info` – Displays information for a container
- `lxc-ls` – Lists the containers belonging to a host system
- `lxc-monitor` – Monitors the state of a container

- `lxc-snapshot` – Creates, lists, and restores container snapshots
- `lxc-start` – Runs a system inside a container
- `lxc-stop` – Kills all the processes inside a container
- `lxc-unfreeze` – Resumes all the processes belonging to a container
- `lxc-unshare` – Runs a task in a cloned set of namespaces
- `lxc-usernsexec` – Runs a task as the `root` user in a new user namespace
- `lxc-wait` – Waits for a container to change to a specified state

## Quiz



Which of the following statements are true?

- a. Containers provide application and operating system isolation.
- b. Each container has its own copy of the Linux kernel.
- c. Containers rely on the cgroup's functionality but also rely on namespace isolation (similar to chroot).
- d. By default, the process IDs, SysV IPCs, and mount points are isolated.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true about LXC template scripts?

- a. Template scripts define the settings for the different system resources that are assigned to a running container.
- b. Template scripts are located in the `/usr/share/lxc/templates` directory.
- c. Each template script is used to generate a specific type of Linux Container.
- d. Template scripts are arguments to the `lxc-start` and `lxc-execute` commands.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe the purpose of Linux Containers
- Describe container configuration parameters
- Set up a system for Linux Containers
- Describe Linux Container template scripts
- Create a Linux Container by using the Oracle template script
- Use Linux Container utilities to start and stop a container
- Use additional Linux Container utilities



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 15: Overview

This practice covers the following topics:

- Preparing your system to create a Linux Container
- Creating a Linux Container from a template
- Exploring container configuration
- Using `lxc` commands



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Docker

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the purpose of Docker
- Describe the use of Docker images
- Describe the Docker Hub Registry
- Install and start Docker
- Search and download images from the Docker Hub Registry
- View Docker container logs and information about images
- Create, start, stop, and remove a Docker container
- Create a Docker image from a Docker container or Dockerfile
- Save a Docker container or image for exporting
- Load an exported Docker container or image
- Describe Oracle WebLogic Docker Certification



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Docker: Introduction

- Docker provides a means for building and packaging applications into lightweight containers.
- Docker consists of the following components:
  - Docker Engine – Runtime and packaging tool
  - Docker Hub Registry – A cloud service for sharing applications
- Use Docker to create image-based application containers.
  - Application is packaged with the individual runtime stack
  - Container is independent from the host operating system
- With Docker, you can transfer the container to any machine that runs Docker and run the application without any compatibility issues.
- Applications that run in Docker containers are small, fast, and secure.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Docker is an open platform management tool for Linux Containers. It provides a means for developers and system administrators to build and package applications into lightweight containers. Docker consists of the following components:

- Docker Engine – A portable, lightweight runtime and packaging tool
- Docker Hub – A cloud service for sharing applications and automating workflows

Docker is used to create image-based application containers. Image-based containers package an application with the individual runtime stack into a single container. This makes the container independent from the host operating system and kernel version. As a result, you can run the same application, unchanged, on laptops, data center virtual machines, and any cloud. You can transfer this container to another machine that runs Docker and run the application without any compatibility issues.

The following lists additional advantages of running applications within Docker containers:

- Docker images contain only the content needed to run an application so they are smaller than virtual machines, which require the entire operating system.
- A Docker container runs faster than an application that includes the overhead of an entire virtual machine.
- A Docker container includes its own network interfaces, file system, and memory, which allows the application running in the container to be isolated and secured from other processes on the host computer.

# Docker Images

- Docker containers are built from Docker images. Example:
- ```
# docker create --name guest oraclelinux:7 /bin/bash
```
- Container name is guest.
  - Image name is oraclelinux:7.
  - Application running in the container is /bin/bash.
- Each Docker image consists of a series of layers:
    - The base layer provides the runtime environment.
    - Each instruction adds another layer.
    - The application container runs in the top layer.
  - Docker provides the capabilities to build images or update existing images.
  - Instructions for building images are stored in a Dockerfile.
  - Docker images can be shared with other users.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Docker containers are built from Docker images. You create Docker containers to run your applications from these Docker images. The following example creates a Docker container named “guest” from a Docker image named “oraclelinux:7” and runs /bin/bash.

```
# docker create --name guest oraclelinux:7 /bin/bash
```

Images are read-only templates that can contain an entire operating system with services and additional applications installed. Docker provides the capabilities to build images or update existing images. You can create Docker images from the command line or you can store the instructions to build an image in a Dockerfile. Docker reads this Dockerfile and executes the instructions when you initiate the build of a Docker image.

Each image starts from a base image (for example, oraclelinux) a base Oracle Linux image. Each Docker image consists of a series of layers that are built from these base images. Each instruction in the Dockerfile creates a new layer in the image. Each time you make a change to a Docker image, only that layer is updated or added. Docker uses unionfs to combine these layers into a single image. The unionfs file system service allows files and directories of separate file systems to be overlaid into a single file system.

After you create a Docker image, you can share the images by storing them in Docker registries. These registries can be private or public. Docker Hub is the public Docker registry that acts as a Software-as-a-Service platform for sharing and managing Docker containers.

# The Docker Hub Registry

- The Docker Hub Registry provides a means to share images.
- Docker Hub is owned and maintained by Docker, Inc. and is located at <https://registry.hub.docker.com/>.
- Oracle images are available at the Oracle Linux repository at [https://registry.hub.docker.com/\\_oraclelinux/](https://registry.hub.docker.com/_oraclelinux/).
- You need to create an account on Docker Hub to use all available services including:
  - User authentication
  - Automated image builds and work-flow tools
  - Integration with GitHub and BitBucket
  - Uploading images to Docker Hub
- You can search for Docker images and download Docker images without a Docker Hub account.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Docker Hub Registry hosts applications as Docker images and provides services that allow you to create and manage a Docker environment. The Docker Hub Registry is owned and maintained by Docker, Inc. and is located at <https://registry.hub.docker.com/>.

Docker Hub provides a number of repositories and each repository can contain a number of images. Oracle makes Docker images available on the “oraclelinux” repository that you can download and use with the Docker Engine. The Oracle Linux repository is located at [https://registry.hub.docker.com/\\_oraclelinux/](https://registry.hub.docker.com/_oraclelinux/). Oracle does not have any control otherwise over the content of the Docker Hub Registry site or any other repositories.

In addition to hosting Docker images, Docker Hub provides services such as user authentication, automated image builds and work-flow tools, and integration with GitHub and BitBucket. To use these Docker services, you need to create a Docker Hub account. You can create an account at <https://hub.docker.com/account/signup/>. You can also create an account from the command line by using the following command:

```
# docker login
```

You can search for Docker images and pull (download) images from the Docker Hub without having an account. To push (upload) images, leave comments on an image or repository, and use all available Docker Hub services, you need a Docker Hub account.

# The Oracle Linux Repository

The screenshot shows a Mozilla Firefox browser window displaying the Docker Hub Registry. The URL in the address bar is [https://registry.hub.docker.com/\\_/oraclelinux/](https://registry.hub.docker.com/_/oraclelinux/). The page title is "oraclelinux Repository | Docker Hub Registry - Repositories of Docker Images". The main content area shows the "oraclelinux" repository as an "OFFICIAL REPO". It lists several supported tags: latest, 7.1, 7.0, 6.6, and 5.11. A "Tags" tab is visible above the list. To the right, there are "Properties" (creation date 2015-01-01 17:28:42, library) and "Settings" (Webhooks) sections. Below the tags list, it says "Supported tags and respective Dockerfile links". A note at the bottom indicates that more information can be found in the [relevant manifest file](#) ([library/oraclelinux](#)) in the [docker-library/official-images GitHub repo](#). The Oracle logo is visible in the bottom left corner of the browser window.

This screen shows the official Oracle Linux images that are available directly from the Docker Hub Registry. The Oracle Linux repository is located at [https://registry.hub.docker.com/\\_/oraclelinux/](https://registry.hub.docker.com/_/oraclelinux/).

# Installing and Starting Docker

- The Docker package is available for Oracle Linux 7 from:
  - Add-ons Channel on ULN (Unbreakable Linux Network)
  - ol7\_addons Public Yum repository
- Subscribe to the ULN channel or enable the Yum repository.
- Use the following command to install Docker:

```
# yum install docker
```

- Enable and start the docker service:

```
# systemctl enable docker  
# systemctl enable docker
```

- Docker uses devicemapper as the default storage driver.
- Oracle Linux 7 supports use of Btrfs as the storage driver.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Docker package is available for Oracle Linux 7 from the Add-ons Channel on ULN (Unbreakable Linux Network) or from the ol7\_addons Public Yum repository. If using ULN, use the web interface at <https://linux.oracle.com/> to subscribe to the “Oracle Linux 7 Addons (x86\_64)” channel. If using Public Yum, enable the ol7\_addons repository as follows.

```
# vi /etc/yum.repos.d/public-yum-ol7.repo  
[ol7_addons]  
enabled=1
```

You can then use the following command to install the package:

```
# yum install docker
```

Use the systemctl command to enable and start the docker service.

```
# systemctl enable docker  
# systemctl start docker
```

By default, Docker uses devicemapper as the storage driver. With Oracle Linux 7, you can configure the Docker Engine to use Btrfs. This allows you to take advantage of the snapshot features of Btrfs. Refer to the *Oracle Linux 7 Administrator’s Guide* for the procedure to configure the Btrfs storage driver for Docker.

The following command displays all the files that are installed from the docker package:

```
# rpm -ql docker
/etc/docker
/etc/sysconfig/docker
/etc/sysconfig/docker-network
/etc/sysconfig/docker-storage
/etc/udev/rules.d/80-docker.rules
/usr/bin/docker
/usr/lib/systemd/system/docker.service
/usr/libexec/docker
/usr/libexec/docker/dockerinit
/usr/share/bash-completion/completions/docker
/usr/share/doc/docker-1.6.1
/usr/share/doc/docker-1.6.1/AUTHORS
...
/usr/share/man/man1/docker-attach.1.qz
/usr/share/man/man1/docker-build.1.qz
/usr/share/man/man1/docker-commit.1.qz
...
/usr/share/man/man1/docker.1.qz
/usr/share/man/man5/Dockerfile.5.qz
/var/lib/docker
```

You can see that, in addition to the docker binaries and configuration files, documentation and man pages are installed for all of the docker commands.

The /var/lib/docker directory is empty until the docker service is started. The following series of commands displays the contents of the directory before and after starting docker.

```
# ls /var/lib/docker
# systemctl enable docker
# systemctl start docker
# ls /var/lib/docker
containers      graph      linkgraph.db          tmp      volumes
devicemapper   init      repositories-devicemapper  trust
```

## The docker Utility

- The `docker` command-line interface has over 30 commands.
- Run `docker` without any arguments, or refer to the `docker(1)` man page for a list of commands.
- A partial list of the `docker` commands are:
  - `docker build`: Build a Docker image from a Dockerfile
  - `docker create`: Create a new container
  - `docker exec`: Run a command in a running container
  - `docker info`: Display systemwide information
  - `docker kill`: Kill a running container
  - `docker pull`: Download an image or a repository
  - `docker run`: Run a command in a new container
  - `docker search`: Search Docker Hub for images
  - `docker start`: Start a stopped container



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `docker` command-line interface has over 30 commands. Refer to the `docker(1)` man page for a list of the commands. The `docker info` command displays systemwide information about the Docker installation. See the `docker-info(1)` man page for more information. Note that the default storage driver is `devicemapper` and that data and metadata are stored in loop devices: `/dev/loop0` and `/dev/loop1`.

```
# docker info
Containers: 0
Images: 0
Storage Driver: devicemapper
  Pool Name: docker-252:0-37294159-pool
  Pool Blocksize: 65.54 kB
  Backing Filesystem: xfs
  Data file: /dev/loop0
...
...
```

Users other than `root` can run `docker` commands if you add them to the `docker` group and reconfigure the `docker` service. Refer to the *Oracle Linux 7 Administrator's Guide* for the procedure to enable non-`root` users to run `docker` commands.

# Using Btrfs as the Storage Engine

- Docker uses devicemapper devices as the default storage engine.
- You can use Btrfs instead of devicemaper to take advantage of the snapshot features of the Btrfs file system.
- Perform the following steps:
  1. Create a Btrfs file system on a block device.
  2. Create the /etc/systemd/system/var-lib-docker.mount file.
  3. Enable and start the var-lib-docker.mount target.
  4. Set SELinux to Permissive mode.
  5. Set the OPTIONS variable in /etc/sysconfig/docker.
  6. Edit /etc/systemd/system/docker.service.
  7. Enable and start the docker service.
- See the Oracle Linux Admin Guide for details.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Docker uses devicemapper devices as the default storage engine. To use Btrfs as the storage engine, perform the following steps. Note that Red Hat Enterprise Linux (RHEL) removes the Btrfs storage driver from their build of Docker, both on the Extra Packages for Enterprise Linux (EPEL) repository and the version released for RHEL7. Oracle restores the Btrfs storage driver into the version built for Oracle Linux.

Use the `systemctl` command to stop the `docker` service.

Use the `mkfs.btrfs` command to create a Btrfs file system on a block device.

Use the `blkid` command to determine the UUID of the Btrfs file system.

Create the `/etc/systemd/system/var-lib-docker.mount` file as follows:

```
[Unit]
Description = Docker Image Store
[Mount]
What = UUID=<UUID for the Btrfs file system>
Where = /var/lib/docker
Type = btrfs
[Install]
WantedBy = multi-user.target
```

Use the `systemctl` command to enable and start the `var-lib-docker.mount` target.

```
# systemctl enable var-lib-docker.mount
# systemctl start var-lib-docker.mount
```

Set the SELinux mode to “Permissive.” SELinux does not currently support the Btrfs storage driver. SELinux is covered in the lesson titled “Security Enhanced Linux (SELinux)” of this course.

Edit the `/etc/sysconfig/docker` file and set the `OPTIONS` variable as follows:

```
OPTIONS=-s btrfs
```

Copy the `/usr/lib/systemd/system/docker.service` file to `/etc/systemd/system/docker.service`. Edit `/etc/systemd/system/docker.service` and add `Requires` and `After` entries for the `var-lib-docker.mount` target as follows:

```
[Unit]
Description=Docker Application Container Engine
Documentation=http://docs.docker.com
Requires=var-lib-docker.mount
After=network.target docker.socket
Requires=docker.socket
After=var-lib-docker.mount
[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/docker
EnvironmentFile=/etc/sysconfig/docker-storage
ExecStart=/usr/bin/docker -d -H fd:// $OPTIONS
$DOCKER_STORAGE_OPTIONS
LimitNOFILE=1048576
LimitNPROC=1048576
[Install]
WantedBy=multi-user.target
```

If your system needs to use a web proxy to access the Docker Hub Registry, edit the `/etc/sysconfig/docker` file and add the following lines. Replace `<proxy_URL:port>` with the appropriate URL and port number for your web proxy.

```
HTTP_PROXY="proxy_URL:port"
HTTPS_PROXY="proxy_URL:port"
```

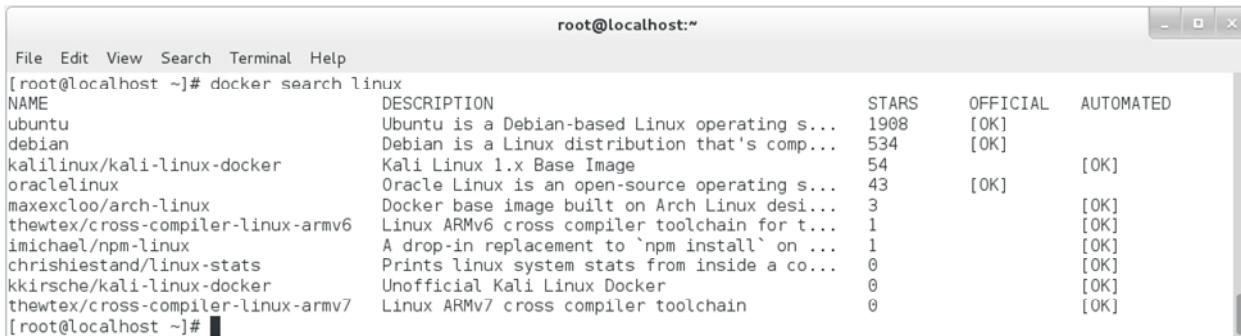
Use the `systemctl` command to enable and start the `docker` service.

```
# systemctl enable docker
# systemctl start docker
```

The `docker info` command now shows Btrfs as the storage driver:

```
# docker info
Storage Driver: btrfs
```

# Searching the Docker Hub Registry for Images



```
root@localhost:~# docker search linux
NAME          DESCRIPTION                           STARS  OFFICIAL  AUTOMATED
ubuntu        Ubuntu is a Debian-based Linux operating s... 1908  [OK]
debian        Debian is a Linux distribution that's comp... 534   [OK]
kalilinux/kali-linux-docker  Kali Linux 1.x Base Image 54    [OK]
oraclelinux   Oracle Linux is an open-source operating s... 43    [OK]
maxexcloo/arch-linux  Docker base image built on Arch Linux desi... 3     [OK]
thewtex/cross-compiler-linux-armv6  Linux ARMv6 cross compiler toolchain for t... 1     [OK]
imichael/npm-linux  A drop-in replacement to `npm install` on ... 1     [OK]
chrishiestand/linux-stats  Prints linux system stats from inside a co... 0     [OK]
kkirsche/kali-linux-docker  Unofficial Kali Linux Docker 0     [OK]
thewtex/cross-compiler-linux-armv7  Linux ARMv7 cross compiler toolchain 0     [OK]
[root@localhost ~]#
```

- The Name column can include the containing repository.
- Stars are a measure of popularity.
- Official repositories are certified by vendors or contributors.
- Automated images are built by the Docker Hub's automated build process.

**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker search` command to search the Docker Hub for images. The slide shows the output of the following command:

```
# docker search linux
```

This command searches the Docker Hub for “linux” images. The output includes the name of the repository/image, a description, the number of stars awarded, whether the image is official, and whether it is automated. The name column is in the following form and can include the containing repository to provide a unique identification:

```
<repository_name>/<image_name>
```

Stars measure the popularity of images. Anyone with a Docker Hub account can “star” an image if they like it. The following example searches for “ubuntu” images with at least 3 stars:

```
# docker search -s 3 ubuntu
```

| NAME           | DESCRIPTION                   | STARS | ... |
|----------------|-------------------------------|-------|-----|
| ubuntu         | Ubuntu is a Debian-based ...  | 1908  | ... |
| ubuntu-upstart | Upstart is an event-based ... | 27    | ... |

An “official” repository is certified by a vendor or contributor to Docker. An “automated” image is built by the Docker Hub’s automated build process.

# Downloading Images from Docker Hub

- Use the `docker pull` command to download a repository or an image from Docker Hub. Examples:

```
# docker pull oraclelinux
...
# docker pull ubuntu
...
# docker pull nimmis/ubuntu
...
```

- Use the `docker images` command to list images stored in the local Docker repository. Example:

| # docker images |        |              |             |     |  |
|-----------------|--------|--------------|-------------|-----|--|
| REPOSITORY      | TAG    | IMAGE ID     | CREATED     | ... |  |
| mimmis/ubuntu   | latest | bd569b81d746 | 3 days ago  | ... |  |
| ubuntu          | latest | d2a0ecffe6fa | 3 days ago  | ... |  |
| oraclelinux     | latest | 8a2b759d9dd8 | 3 weeks ago | ... |  |



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker pull` command to download an image or a repository from the Docker Hub Registry to your local system. If there is more than one image for a repository, then all images for that repository are downloaded. The following example downloads the `oraclelinux` repository with multiple images. Each layer of the image is pulled down.

```
# docker pull oraclelinux
Pulling repository oraclelinux
...
Status: Downloaded newer image for oraclelinux:latest
```

Use the `docker images` command to list images stored in the local Docker repository.

| REPOSITORY    | TAG    | IMAGE ID     | CREATED     | VIRTUAL SIZE |
|---------------|--------|--------------|-------------|--------------|
| mimmis/ubuntu | latest | bd569b81d746 | 3 days ago  | 331.7 MB     |
| ubuntu        | latest | d2a0ecffe6fa | 3 days ago  | 188.3 MB     |
| oraclelinux   | latest | 8a2b759d9dd8 | 3 weeks ago | 189.5 MB     |

Each image in a repository is distinguished by TAG and IMAGE ID.

The following shows the location of the downloaded images:

```
# docker pull oraclelinux
latest: Pulling from oraclelinux
8c3e49cb06dc: Pull complete
90fed8b1ceab: Pull complete
8a2b759d9dd8: Already exists
oraclelinux:latest: The image you are pulling has been verified...
Digest: sha256:...
Status: Downloaded newer image for oraclelinux:latest

# find /var/lib/docker -name "*8c3e49cb06dc*"
/var/lib/docker/devicemapper/metadata/8c3e49cb06dc...
/var/lib/docker/devicemapper/mnt/8c3e49cb06dc...
/var/lib/docker/graph/8c3e49cb06dc...

# ls /var/lib/docker/metadata
8a2b759d9dd8... base
8c3e49cb06dc... deviceset-metadata
90fed8b1ceab... transaction-metadata
```

You can see that each layer, including the `base` layer, is downloaded.

# Running an Application Inside a Container

- Use the `docker run` command to run applications inside containers.
- Include the Docker image you want to use.
  - If the image is not available locally, Docker pulls it down.
- Examples:

```
# docker run oraclelinux /bin/echo "Hello"
Hello
# docker run fedora /bin/echo "Hello"
Unable to find image 'fedora:latest' locally
latest: Pulling from fedora
...
Hello
```

- In both examples, the container stops after the `/bin/echo` command runs.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker run` command to run an application inside a container. This command starts a process with its own file system, its own networking, and its own isolated process tree. The following syntax does not include all of the available options for the command:

```
docker run [OPTION...] IMAGE [COMMAND] {ARG...}
```

The `IMAGE` that starts the process can define defaults related to the process to run in the container, the networking, and more, but `docker run` options override settings in the `IMAGE`. If the `IMAGE` is not available locally, `docker run` pulls the image in the same way as the `docker pull` command before it starts the container in the `IMAGE`.

The slide shows two examples of using `docker run`. The first example uses the “`oraclelinux`” image, which already exists on the local machine. Docker uses the image to create a new Oracle Linux environment and then runs the `echo` command to display “Hello”.

The second example uses the `fedora` image, which is not present on the local system. Docker pulls the image from Docker Hub and then uses the image to create a new Fedora environment and runs the `echo` command.

In both examples, the Docker containers stop after “Hello” is displayed to the screen. Use the `docker ps` command to list containers and no containers are displayed:

```
# docker ps
```

The `docker images` command shows that the “latest” image from the “fedora” repository was downloaded from Docker Hub when using the `docker run` command:

| REPOSITORY    | TAG    | IMAGE ID     | CREATED     | VIRTUAL SIZE |
|---------------|--------|--------------|-------------|--------------|
| mimmis/ubuntu | latest | bd569b81d746 | 3 days ago  | 331.7 MB     |
| ubuntu        | latest | d2a0ecffe6fa | 3 days ago  | 188.3 MB     |
| oraclelinux   | latest | 8a2b759d9dd8 | 3 weeks ago | 189.5 MB     |
| fedora        | latest | ded7cd95e059 | 6 weeks ago | 186.5 MB     |

# Running an Interactive Docker Container

- Use the `docker run` command with the `-t` and `-i` options to run an interactive container. Examples:

```
# cat /etc/oracle-release
Oracle Linux Server release 7.1
# docker run -t -i oraclelinux:6.6 /bin/bash
# cat /etc/oracle-release
Oracle Linux Server release 6.6
# exit
```

```
# cat /etc/oracle-release
Oracle Linux Server release 7.1
# docker run -t -i ubuntu /bin/bash
# cat /etc/os-release
NAME="Ubuntu"
VERSION="14.04.2 LTS, Trusty Tahr"
...
# exit
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `-t` and `-i` options with the `docker run` command to run an interactive container. These options are described:

- `-t`: Allocate a pseudo-tty and attach to STDIN (standard input) of a container
- `-i`: Keep STDIN of a container open

The slide shows two examples of using `docker run` commands with the `-t` and `-i` options. In both examples, Oracle Linux 7.1 is running on the local system:

```
# cat /etc/oracle-release
Oracle Linux Server version 7.1
```

The first example uses the “`oraclelinux:6.6`” image. Docker uses the image to create a new Oracle Linux environment and then runs the `bash` shell command. The OS version of this image is Oracle Linux 6.6.

```
# cat /etc/oracle-release
Oracle Linux Server version 6.6
```

The second example uses the `ubuntu` image. Because no specific image was specified, Docker uses the “`latest`” Ubuntu image to create a new Ubuntu environment and then runs the `bash` shell command. The OS version of this image is Ubuntu 14.04.2.

You can use the `exit` command or press `CTRL + D` to exit an interactive container.

# Listings Containers and Viewing Container Logs

- Use the `docker ps` command to list running containers.
  - Use the `-a` option to include stopped containers.

```
# docker ps -a
CONTAINER ID IMAGE COMMAND ... NAMES
8deb0f6dfa41 oraclelinux:6 "/bin/bash" ... lonely_hop
27f03b3484b0 ubuntu:14.04 "/bin/echo Hello" sharp_bag
```

- Output includes the container ID, container name, the image used to create the container, the command, and status.
- Use the `docker logs` command to view container logs.

```
# docker logs lonely_hop
[root@ 8deb0f6dfa41 /]# ls
bin boot dev etc home lib lib64 media mnt opt proc ...
[root@ 8deb0f6dfa41 /]# ps
 PID TTY      TIME CMD
  1 ?    00:00:00 bash
 15 ?    00:00:00 ps
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker ps` command to list information about Docker containers. By default, only running containers are listed. Include the `-a` option with the `docker ps` command to show all containers. Output includes a unique container ID and unique container name which are automatically generated when the container is created. Output of the `docker ps` command also includes the image that was used to create the container, the command that is running in the container, and status information. Status information includes when the container was created, and how long the container has been running.

```
# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
...
...
```

The `docker logs` command looks inside the container and returns its standard output. You can use either the container ID or the container name as an argument to the command.

```
# docker logs 8deb0f6dfa41
...
```

In the example in the slide, the `docker logs` command shows that the `ls` and `ps` commands were executed in the container. The `docker ps` and `docker logs` commands need to be executed from outside the container, that is from another terminal window.

## Display All Information for a Container or an Image

- Use the `docker inspect` command to view low-level information on a container or an image.
  - The output is provided in JavaScript Object Notation (JSON) format by default.
- The following example displays all information for the “`lonely_hop`” container.

```
# docker inspect lonely_hop
```

- Use the `-f {{ .sectionsubsection }}` option to display a specific piece of information.
- The following example displays the IP address:

```
# docker inspect -f {{ .NetworkSettings.IPAddress }}  
lonely_hop  
172.17.0.10
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker inspect` command to view all the information available for a container or an image. The following notes page shows some of the information that is displayed.

Use the `-f {{ .sectionsubsection }}` option to display a specific piece of information. The following example displays only network settings:

```
# docker inspect -f {{ .NetworkSettings }} lonely_hop  
map [IPv6Gateway: Gateway:172.17.42.1 GlobalIPv6Address:  
GlobalIPv6PrefixLen:0 IPAddress:172.17.0.10 IPPrefixLen:16 ...
```

The following example displays the IP address:

```
# docker inspect -f {{ .NetworkSettings.IPAddress }} lonely_hop  
172.17.0.10
```

The following example displays the process ID:

```
# docker inspect -f {{ .State.Pid }} lonely_hop  
3378
```

The following example displays the command running in the container:

```
# docker inspect -f {{ .Config.Cmd }} lonely_hop  
[/bin/bash]
```

```
# docker inspect lonely_hop
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": true,
        "AttachStdin": false,
        "AttachStdout": true,
        "Cmd": [
            "/bin/bash"
        ],
        "CpuShares": 0,
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": null,
        "ExposedPorts": null,
        "Hostname": "a95b84655815",
        "Image": "oraclelinux:6.6",
        "Labels": {},
        "MacAddress": "",
        "Memory": 0,
        "MemorySwap": 0,
        "NetworkDisabled": false,
        ...
    },
    "Created": "date...",
    "Driver": "devicemapper",
    "ExecDriver": "native-0.2",
    "HostConfig": {
        "Binds": null,
        "CapAdd": null,
        "CapDrop": null,
        "CgroupParent": "",
        ...
    },
    "HostnamePath": "/var/lib/docker/containers/a95b84655815...
    ...
}
```

# Creating a New Container

- Use the `docker create` command to create a new container to start at a later time.
  - The `docker run` command also creates a new container but runs a process in a new container immediately.
  - The two commands have the same syntax and options.
- Following example creates a new container named “guest”:

```
# docker create -i -t --name guest oraclelinux:6.6
/bin/bash
ca3fa0f731f8...
```

- Output of command is a unique container ID.
- The new container does not start immediately.

```
# docker ps -a | grep ca3fa0f731f8
ca3fa0f731f8  oraclelinux:6  "/bin/bash"  ...  guest
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `docker run` command runs a process in a new container. You can also use the `docker create` command to create a container that you can start at a later time. The syntax and available options for `docker create` are similar to the `docker run` syntax.

The example in the slide creates a new container named “guest” from the `oraclelinux:6.6` image, and when started, runs the `bash` shell command. A container name is automatically generated if you omit the `--name` option.

```
# docker create -i -t --name guest oraclelinux:6.6 /bin/bash
ca3fa0f731f8...
```

The output of the command is a very long unique container ID. The container does not start immediately as shown by the `docker ps` command, which only shows running containers by default:

```
# docker ps
```

You need to run `docker ps -a` to show all containers. The example pipes the output to `grep` and searches part of the container ID:

```
# docker ps -a | grep ca3fa
ca3fa0f731f8  oraclelinux:6  "/bin/bash"  ...  guest
```

# Starting, Stopping, and Removing a Container

- Use the `docker start` command to start a container.
  - Include the `-a` and `-i` options to get a shell prompt.
- Provide either the container ID or container name. Example:

```
# docker start -a -i guest
[root@guest] #
```

- From inside a container, use the `exit` command or CTRL-d to stop the container.
- From outside a container, use the `docker stop` command.

```
# docker stop guest
```

- Use the `docker rm` command to remove a container.
  - Include the `-f` option to remove a running container.

```
# docker rm -f guest
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker start` command to start an existing container. Use the `-a` and `-i` options to attach the current shell's STDIN (standard input), STDOUT (standard output), and STDERR (standard error) to the container and also cause all signals to be forwarded to the container.

```
# docker start -a -i guest
```

From inside a container, use the `exit` command or CTRL-d to stop the container. From outside the container, that is from another terminal window, use `docker stop` command to stop a container.

```
# docker stop guest
```

Use the `docker rm` command to remove a container. You can remove multiple containers in a single command. Use the `-f` option to remove a running container.

```
# docker rm guest
```

All three of these `docker` commands accept either the container ID or the container name as an argument.

The `docker rm` command removes a container. Use the `docker rmi` command to remove an image:

```
# docker rmi <IMAGE>
```

# Running Additional Commands in a Running Container

- Use the `docker exec` command to run additional processes in a running container.
  - Use the `-t` and `-i` options to run an interactive process.
- The following example starts an interactive `bash` shell in the “guest” container:

```
# docker exec -t -i guest /bin/bash
[root@guest] #
```

- The following example uses the `docker exec` command to start the `sshd` service on the “guest” container:

```
# docker exec guest service sshd start
Generating SSH2 RSA host key: [ OK ]
Generating SSH1 RSA host key: [ OK ]
Generating SSH2 DSA host key: [ OK ]
Starting sshd: [ OK ]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the `docker exec` command to run a command in a running container. Similar to the `docker run` command, include the `-t` and `-i` options to run an interactive command. Provide either the container ID or the container name as an argument.

In the following example, the `docker exec` command starts a new interactive `bash` shell in the “guest” container:

```
# docker exec -t -i guest /bin/bash
[root@guest] #
```

The following example uses the `docker exec` command to start the `sshd` service on the “guest” container. The “guest” container is running Oracle Linux 6.6. In this example, the `-t` and `-i` options are not needed.

```
# docker exec guest service sshd start
Generating SSH2 RSA host key: [ OK ]
Generating SSH1 RSA host key: [ OK ]
Generating SSH2 DSA host key: [ OK ]
Starting sshd: [ OK ]
```

The `sshd` service is started on the container and control returns to the initiating host system.

## Creating an Image from a Container

- Use the `docker commit` command to save the current state of a container as a new image.
- The following example creates a container, installs the `httpd` package, and then creates a new image:

```
# docker run -i -t --name newguest oraclelinux:7
    /bin/bash
[newguest]# yum install httpd
[newguest]# exit
# docker commit -m="Installed httpd on OL7" -a="My Name"
    newguest mymod/httpd:v1
```

- The `docker images` command now lists the new image:

| REPOSITORY  | TAG | IMAGE ID | CREATED | VIRTUAL SIZE |
|-------------|-----|----------|---------|--------------|
| mymod/httpd | v1  | ...      |         |              |
| ...         |     |          |         |              |



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can save the current state of a container as a new image by using the `docker commit` command. This is useful if you have modified a container and want to commit the changes to a new image for later use.

The example in the slide creates a new container named “newguest” from the `oraclelinux:7` image and runs the `bash` shell command in the container.

```
# docker run -i -t --name newguest oraclelinux:7 /bin/bash
```

From within the container, the `yum` command is used to install the `httpd` package.

```
[newguest]# yum install httpd
```

Use the `exit` command to stop a running container.

```
[newguest]# exit
```

The `docker commit` command saves the changes to a new image. Use the `-m` option to provide a message describing the changes. Use the `-a` option to provide author information. Provide the container ID or container name, the image name, and a tag. Example:

```
# docker commit -m="Installed httpd on OL7" -a="My Name"
    newguest mymod/httpd:v1
```

The output of the `docker images` command now includes the new image.

# Creating an Image from a Dockerfile

- A Dockerfile contains the set of instructions for building a Docker image.
  - Create a new directory and create the file, named Dockerfile, in a new directory.
- Use the docker build command to create the image from the Dockerfile.
  - Use the -t option to specify the repository name and tag for the resulting image.
  - Provide the path to Dockerfile.
- Example:

```
# mkdir /var/dockerfiles
# cd /var/dockerfiles
# vi Dockerfile
...
# docker build -t="mymod/httpd:v2" .
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the docker build command to create a new image from the instructions contained in a file named “Dockerfile”. The format of the Dockerfile is:

#### ***INSTRUCTION arguments***

The instruction is not case-sensitive but convention is to capitalize the instruction to distinguish it from the arguments. Docker runs the instructions in a Dockerfile in order. You build a new image from a base image. The first instruction is FROM and specifies the base image to use. Example:

```
FROM oraclelinux:7.1
```

Use the RUN instruction to specify the commands to run in a new layer on top of the current image and commit the results. Example:

```
RUN yum -y install httpd
```

The ENTRYPOINT instruction specifies the command that the container created from the image runs. Example:

```
ENTRYPOINT /usr/sbin/httpd -D FOREGROUND
```

Refer to the dockerfile(5) man page for a description of all the instructions. The following URL also provides descriptions, usage, and examples of all the available Dockerfile instructions: <https://docs.docker.com/reference/builder/>.

## Save and Load an Image or a Container

- Use the `docker save` command to save an image to a tar archive.
  - The following example saves all `oraclelinux` images to the `oraclelinux-all.tar` file.

```
# docker save oraclelinux > oraclelinux-all.tar
```

- Use the `-o` option to write the output to a file rather than to STDOUT.
  - The following example saves the `oraclelinux:latest` image to the `ol-71.tar` file.

```
# docker save -o=ol-71.tar oraclelinux:latest
```

- Use the `docker load` command to load an image from a tar file to a local Docker repository. Example:

```
# docker load --input oraclelinux-all.tar
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can create tar files of images and containers to use on systems that do not have access to Docker Hub.

Use the `docker save` command to save images to a tar file. You can either save all images in a repository to a tar file, or save a specific image to a tar file. Create the tar file either by redirecting STDOUT to a tar file or use the `-o` option to specify an output tar file name.

The following example redirects STDOUT to save all images in the `oraclelinux` repository to `oraclelinux-all.tar`:

```
# docker save oraclelinux > oraclelinux-all.tar
```

Use the `docker load` command to load an image from a tar file to a local Docker repository. The following example loads the images from the `oraclelinux-all.tar` file:

```
# docker load --input oraclelinux-all.tar
```

# Oracle WebLogic Docker Certification

- Oracle WebLogic Server is now certified to run on Docker containers.
- Dockerfiles and supporting scripts to build Oracle WebLogic Server Docker images are available on GitHub.
- Docker images are built as an extension of existing Oracle Linux image 7.0, with JDK 7, and the Oracle WebLogic Server 12c (12.1.3) installations.
- Dockerfiles and scripts enable users to create clustered and non-clustered Oracle WebLogic Server domain configurations.
- Each server running in the resulting domain configurations runs in its Docker container, and is capable of communicating as required with other servers.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle WebLogic Server is now certified to run on Docker containers. As part of the certification, Oracle is releasing Dockerfiles and supporting scripts to build images for Oracle WebLogic Server on GitHub: <https://github.com/oracle/docker/tree/master/OracleWebLogic>

These images are built as an extension of existing Oracle Linux image 7.0, with JDK 7, and the Oracle WebLogic Server 12c (12.1.3) installations.

These Dockerfiles and scripts enable users to create clustered and non-clustered Oracle WebLogic Server domain configurations, including both development and production running on a single host operating system or VMs. Each server running in the resulting domain configurations runs in its Docker container, and is capable of communicating as required with other servers.

For documentation on how to use these Dockerfiles and scripts, see the following:  
<http://www.oracle.com/technetwork/middleware/weblogic/overview/weblogic-server-docker-containers-2491959.pdf>.

You can also view a video demo of Oracle WebLogic Server running on Docker containers at <https://www.youtube.com/watch?v=SyznNrN1xFM&feature=youtu.be>.

You can see the Oracle WebLogic Docker certification announcement at [https://blogs.oracle.com/WebLogicServer/entry/oracle\\_weblogic\\_server\\_now\\_running](https://blogs.oracle.com/WebLogicServer/entry/oracle_weblogic_server_now_running).

## Quiz



Which of the following statements are true?

- a. Docker images contain only the content needed to run an application so they are smaller than virtual machines.
- b. A Docker container includes its own network interfaces, file system, and memory.
- c. An application running in a Docker container is isolated and secured from other processes on the host computer.
- d. With Docker containers, you can run the same application, unchanged, on laptops, data center virtual machines, and any cloud.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true about Docker containers?

- a. Docker containers are created from Docker images.
- b. You can run an application as a daemon inside a Docker container.
- c. You can run an interactive application inside a Docker container.
- d. Users other than `root` can run `docker` commands.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe the purpose of Docker
- Describe the use of Docker images
- Describe the Docker Hub Registry
- Install and start Docker
- Search and download images from the Docker Hub Registry
- View Docker container logs and information about images
- Create, start, stop, and remove a Docker container
- Create a Docker image from a Docker container or Dockerfile
- Save a Docker container or image for exporting
- Load an exported Docker container or image
- Describe Oracle WebLogic Docker Certification



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 16: Overview

This practice covers the following topics:

- Using `sftp` to upload Docker package and images
- Installing and configuring Docker
- Using the `docker` command-line interface



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.



# Security Enhanced Linux (SELinux)

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe SELinux
- Describe SELinux packages
- Use the SELinux Administration GUI
- Describe SELinux modes
- Describe SELinux policies
- Describe SELinux Booleans
- Describe SELinux file labeling, context, and users
- Use SELinux utilities



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Introduction to SELinux

- SELinux:
  - Was created by the National Security Agency
  - Is an enhancement to the Linux kernel
  - Implements a type of security called MAC
- Standard Linux security is based on DAC:
  - DAC provides minimal protection.
  - Access to objects is based on user identity and ownership.
- MAC can define security policy:
  - It provides granular permissions for all users, programs, processes, files, and devices.
  - SELinux policy rule determines whether access is allowed.
- The Linux kernel checks and enforces MAC rules after it checks DAC rules.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Standard Linux security is based on Discretionary Access Control (DAC). DAC provides minimal protection from broken software or malware running as a normal user or `root`. Access to files and devices is based solely on user identity and ownership. Malicious or flawed software can do anything with the files and resources it controls through the user that started the process. If the user is `root` or the application is `setuid` or `setgid` to `root`, the process can have `root`-level control over the entire file system.

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

MAC under SELinux allows you to define a security policy that provides granular permissions for all users, programs, processes, files, and devices. Access control decisions are based on all the security-relevant information available, and not just authenticated user identity. When security-relevant access takes place, such as when a process attempts to open a file, the operation is intercepted in the kernel by SELinux. If an SELinux policy rule allows the operation, it continues; otherwise, the operation is blocked and the process receives an error. The kernel checks and enforces MAC rules after it checks DAC rules. SELinux policy rules are not used if DAC rules deny access first.

# SELinux Packages

- **policycoreutils**: Provides the policy core utilities that are required for basic operation of SELinux
- **policycoreutils-python**: Provides additional tools
- **libselinux-utils**: Provides additional SELinux tools
- **libselinux**: Provides an API for SELinux tools
- **setroubleshoot-server**: Translates denial messages into detailed descriptions that are viewed by using `sealert`
- **libselinux-python**: Contains the Python bindings for developing SELinux applications
- **selinux-policy**: Provides the SELinux Reference Policy
- **selinux-policy-targeted**: Provides targeted policies. For MLS, install `selinux-policy-mls`.
- Other packages are available but not installed by default.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are many different SELinux software packages, some of these are installed by default, and some are not. The following is a list of the SELinux packages that are installed by default:

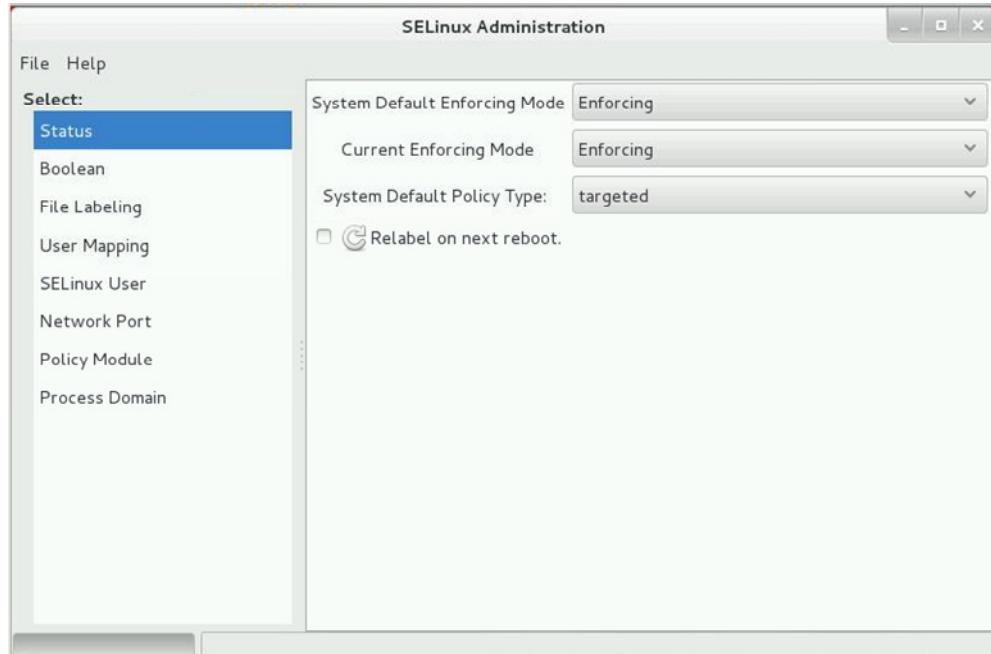
- **policycoreutils**: Provides the `sestatus`, `restorecon`, `secon`, `setfiles`, `semodule`, `load_policy`, and `setsebool` tools for operating and managing SELinux
- **policycoreutils-python**: Provides additional utilities such as `semanage`, `audit2allow`, `audit2why`, and `chcat` for operating and managing SELinux
- **libselinux-utils**: Provides the `avcstat`, `getenforce`, `getsebool`, `matchpathcon`, `selinuxconlist`, `selinuxdefcon`, `selinuxenabled`, `setenforce`, and `toggelsebool` tools
- **libselinux**: Provides an API for SELinux applications to get and set process and file security contexts and to obtain security policy decisions
- **setroubleshoot-server**: Translates denial messages, produced when access is denied by SELinux, into detailed descriptions that are viewed by using `sealert`
- **libselinux-python**: Contains Python bindings for developing SELinux applications
- **selinux-policy**: Provides the SELinux Reference Policy. The SELinux Reference Policy is a complete SELinux policy and is used as a basis for other policies, such as the SELinux targeted policy.
- **selinux-policy-targeted**: Provides the targeted policy. For MLS policy, install `selinux-policy-mls`.

The following is a partial list of SELinux packages that are not installed by default:

- **policycoreutils-gui**: Provides `system-config-selinux`, which is a graphical tool for managing SELinux
- **setools-console**: Provides the Tresys Technology SETools distribution, several tools and libraries for analyzing and querying policy, audit log monitoring and reporting, and file context management
- **mcstrans**: Translates levels, such as `s0-s0:c0.c1023`, to an easier-to-read form, such as `SystemLow-SystemHigh`

# SELinux Administration GUI

`system-config-selinux`



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The SELinux Administration graphical user interface (GUI) is a tool used to manage SELinux. To run this GUI, install the `policycoreutils-gui` package:

```
# yum install policycoreutils-gui
```

Enter the following command to display the GUI:

```
# system-config-selinux
```

The GUI displays a list of options on the left side. The right side of the GUI changes as different options are selected from the left.

The example shown in the slide is the Status view. From this view, you can change:

- **Enforcing Mode:** Enforcing, Permissive, Disabled
- **Policy Type:** Targeted, MLS (Multi-Level Security)
- **Relabel on next reboot:** Labels all files at boot time with an SELinux context

The `sestatus` command displays the SELinux mode and the SELinux policy being used.

```
# sestatus
Loaded policy name:          targeted
Current mode:                enforcing
```

## SELinux Modes

- **Enforcing:** This is the default state. Access is denied to users and programs unless permitted by SELinux security policy rules.
- **Permissive:** Security policy rules are not enforced but denial messages are logged.
- **Disabled:** No security policy is loaded. Only DAC rules are used for access control.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

SELinux runs in one of three modes (or states).

### **Enforcing**

This is the default state that enforces SELinux security policy. Access is denied to users and programs unless permitted by SELinux security policy rules. All denial messages are logged as AVC (Access Vector Cache) Denials.

### **Permissive**

This is a diagnostic state. The security policy rules are not enforced, but SELinux sends denial messages to a log file. This allows you to see what would have been denied if SELinux were running in enforcing mode.

### **Disabled**

SELinux does not enforce a security policy because no policy is loaded in the kernel. Only DAC rules are used for access control.

## Setting a Mode

- Set the mode from the SELinux GUI.
- Define the SELINUX directive in /etc/selinux/config:
  - SELINUX=enforcing
  - SELINUX=permissive
  - SELINUX=disabled
- Use the setenforce command to set either enforcing (1) or permissive (0) mode. Mode does not persist after a reboot:

```
# setenforce 1  
# setenforce 0
```

- To view the current SELinux mode:

```
# getenforce  
Enforcing
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are multiple ways of setting the SELinux mode. One way is to select the mode from the Status view in the SELinux GUI.

You can also edit the main configuration file for SELinux, /etc/selinux/config. Set the mode by changing the SELINUX directive in this file. For example, to set the mode to enforcing:

```
SELINUX=enforcing
```

The setenforce command is used to change between enforcing and permissive modes. Changes made with this command do not persist across reboots.

To change to enforcing mode:

```
# setenforce 1
```

To change to permissive mode:

```
# setenforce 0
```

### Display the Current Mode

Use the getenforce command to view the current SELinux mode:

```
# getenforce  
Enforcing
```

## SELinux Policies

- SELinux implements one of two different policies:
  - **Targeted:** Applies access controls to targeted processes
  - **MLS:** Multi-Level Security
- Select the policy type from the SELinux GUI or define the SELINUXTYPE directive in /etc/selinux/config:
  - SELINUXTYPE=targeted
  - SELINUXTYPE=mls
- Targeted policy:
  - Targeted processes run in a confined domain.
  - All network services are targeted processes.
  - Access to files is limited.
- Non-targeted processes run in an unconfined domain.
  - Access to files is not limited.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The SELinux policy describes the access permissions for all users, programs, processes, files, and devices they act upon. SELinux implements one of two different policies:

- **Targeted:** This default policy applies access controls to certain (targeted) processes.
- **MLS:** Multi-Level Security

Select the policy type from the SELinux GUI, or set the SELINUXTYPE directive in the /etc/selinux/config file. Example:

```
SELINUXTYPE=targeted
```

With the targeted policy, targeted processes run in their own domain, called a confined domain. In a confined domain, the files that a targeted process has access to are limited. If a confined process is compromised by an attacker, the attacker's access to resources and the possible damage they can do is also limited. SELinux denies access to these resources and logs the denial.

Only specific services are placed into these distinct security domains that are confined by the policy. For example, a user runs in a completely unconfined domain while services that listen on a network for client requests, such as named, httpd, and sshd, run in a specific, confined domain tailored to its operation. Processes that run as the Linux root user and perform tasks for users, such as the passwd application, are also confined.

Processes that are not targeted run in an unconfined domain. SELinux policy rules allow processes running in unconfined domains almost all access. If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data. DAC rules still apply in an unconfined domain.

The following are examples of unconfined domains:

- `initrc_t` domain: `init` programs run in this unconfined domain.
- `kernel_t` domain: Unconfined kernel processes run in this domain.
- `unconfined_t` domain: Linux users logged in to the system run in this domain.

Many domains that are protected by SELinux have `man` pages describing how to customize their policies.

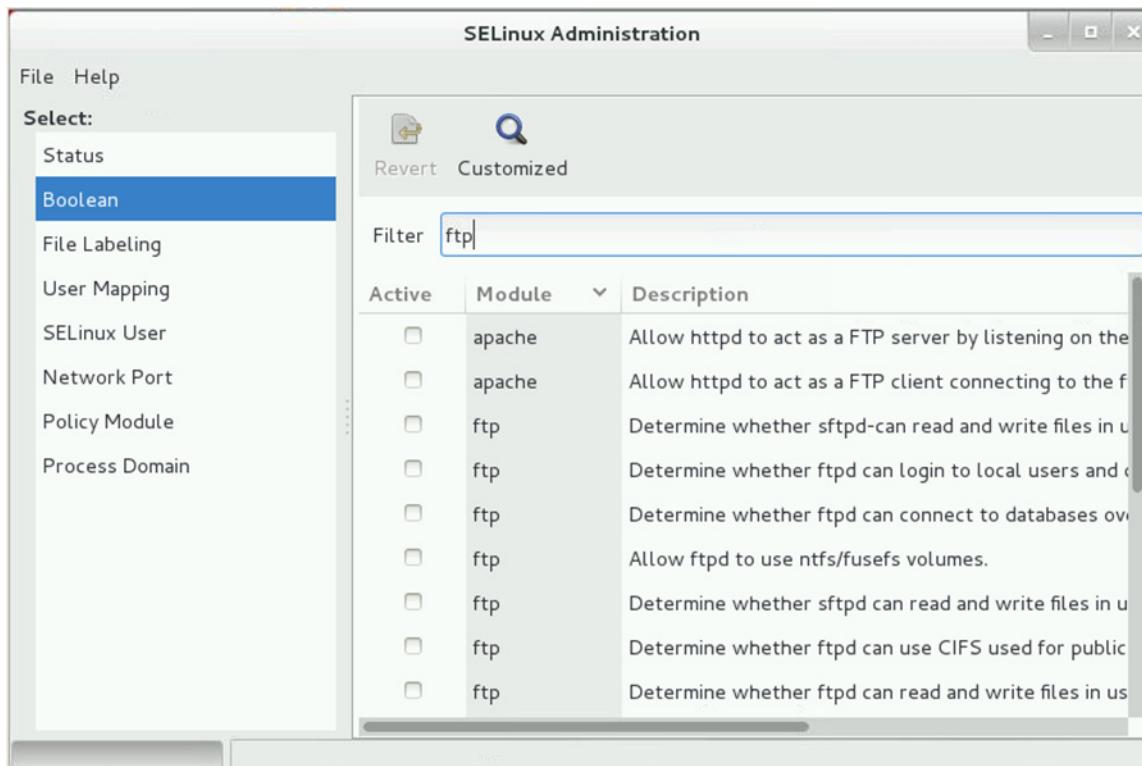
The configuration for each policy is installed in the `/etc/selinux/<SELINUXTYPE>` directories. The following example shows a partial listing of the `/etc/selinux` directory with both targeted and MLS policies installed:

```
# ls -l /etc/selinux
-rw-r--r--. root  root  config
drwxr-xr-x. root  root  mls
drwxr-xr-x. root  root  targeted
```

The targeted policy is installed by default, but the MLS policy is not. To use the MLS policy, install the `selinux-policy-mls` package:

```
# yum install selinux-policy-mls
```

# SELinux Booleans



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A given SELinux policy can be customized by enabling or disabling a set of policy Booleans. Booleans allow parts of SELinux policy to be changed at run time, without any knowledge of SELinux policy writing. This allows changes without reloading or recompiling SELinux policy. The `system-config-selinux` command displays the SELinux GUI, which, via the Boolean view (shown in the slide), allows customization of these Booleans. In the Boolean view, the Active check box indicates whether a Boolean is on or off. You can sort by any of the three columns (Active, Module, or Description).

You can also display this list from the command line by using the following command:

```
# semanage boolean -l
  SELinux boolean      State  Default Description
  ftp_home_dir        (off   ,  off)   Determine whether ftp ...
  smartmon_3ware       (off   ,  off)   Determine whether smart...
  mpd_enable_homedirs (off   ,  off)   Determine whether mpd ...
  ...

```

In the sample listing, the `ftp_home_dir` Boolean is off, which prevents the FTP daemon from reading and writing to files in user home directories.

## getsebool and setsebool Utilities

- Use the `getsebool` command to view the status of a Boolean.
  - To view the status of all Booleans:

```
# getsebool -a
```

- To view the status of a specific Boolean:

```
# getsebool use_nfs_home_dirs
use_nfs_home_dirs --> off
```

- Use the `setsebool` command to change the status of a Boolean. Example:

```
# setsebool use_nfs_home_dirs on
```

- Use the `-P` option to make the change persistent. Example:

```
# setsebool -P use_nfs_home_dirs on
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

### Displaying Booleans

You can also use the `getsebool` command to list Booleans. This command displays statuses but no descriptions.

To display all Booleans and their statuses:

```
# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...
```

Include the Boolean name as an argument to display the status of a specific Boolean. Multiple Boolean arguments are also allowed:

```
# getsebool allow_ftpd_use_nfs mozilla_read_content
allow_ftpd_use_nfs --> off
mozilla_read_content --> off
```

## Setting Booleans

Use the `setsebool` command to configure Booleans from the command line. The syntax is:

```
setsebool <Boolean> on|off
```

For example, the following sequence of commands displays the current status of a Boolean, then enables it to allow the `syslogd` daemon to send mail, and then displays the status again:

```
# getsebool logging_syslogd_can_sendmail
logging_syslogd_can_sendmail --> off
# setsebool logging_syslogd_can_sendmail on
# getsebool logging_syslogd_can_sendmail
logging_syslogd_can_sendmail --> on
```

To make the change persistent across reboots, use the `-P` option:

```
# setsebool -P logging_syslogd_can_sendmail on
/sys/fs/selinux Directory
```

You can also view and change the value of Booleans in the `/sys/fs/selinux` directory. The Boolean files are stored in the `/sys/fs/selinuxBOOLEANS` directory:

```
# ls /sys/fs/selinuxBOOLEANS
abrt_anon_write
abrt_handle_event
...
zebra_write_config
zoneminder_anon_write
zoneminder_run_sudo
```

To view the value of a specific Boolean:

```
# cat /sys/fs/selinuxBOOLEANS/abrt_anon_write
0 0
```

A value of 1 indicates that the Boolean is on, while 0 indicates off. The first number indicates the current value of the Boolean. The second number represents the pending value of the Boolean.

To turn the `abrt_anon_write` Boolean on:

```
# echo 1 > /sys/fs/selinuxBOOLEANS/abrt_anon_write
```

View the contents of the file:

```
# cat /sys/fs/selinuxBOOLEANS/abrt_anon_write
0 1
```

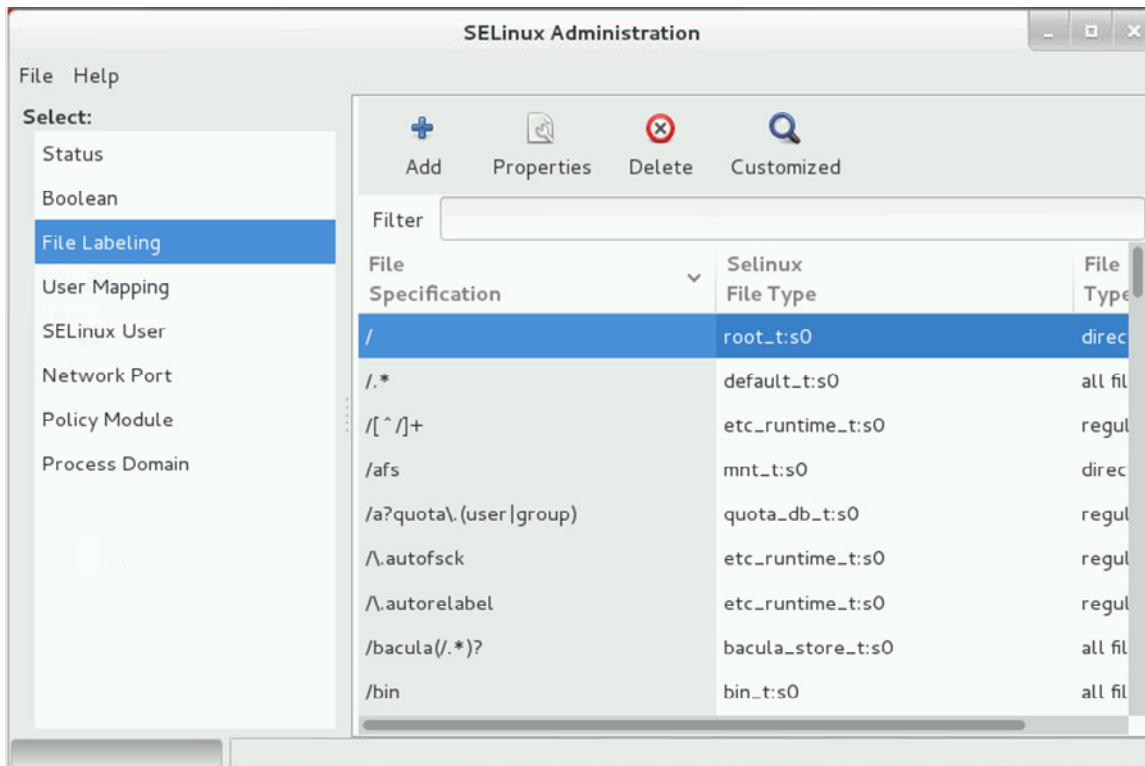
To commit the new value:

```
# echo 1 > /sys/fs/selinux/commit_pending_booleans
```

The value has now changed:

```
# cat /sys/fs/selinuxBOOLEANS/abrt_anon_write
1 1
# getsebool abrt_anon_write
abrt_anon_write --> on
```

# SELinux File Labeling



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The SELinux GUI's File Labeling view is shown in the slide. All files, directories, devices, and processes have a security context (or label) associated with them. For files, this context is stored in the extended attributes of the file system. Problems with SELinux often arise from the file system being mislabeled. If you see an error message containing `file_t`, that is usually a good indicator that you have a problem with file system labeling.

There are several ways to relabel the file system:

- Create the `/ .autorelabel` file and reboot.
- The Status view in the SELinux GUI provides an option to relabel on next reboot.
- Three command-line utilities, `restorecon`, `setfiles`, and `fixfiles`, relabel files.

To view the file system context information from the command line, use the `ls -Z` command:

```
# ls -Z
-rw-----. root root system_u:object_r:admin_home_t:s0 anaco...
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 initi...
```

This information is also stored in the `/etc/selinux/<SELINUXTYPE>/contexts/files` directory.

## SELinux Context

- Context is additional information about users, files, and processes.
- Access control is based on this information:
  - **SELinux user:** Linux users are mapped to SELinux users.
  - **Role:** An attribute of RBAC that acts as an intermediary between domains and SELinux users
  - **Type:** An attribute of TE that defines a domain for processes
  - **Level:** Optional information; an attribute of MLS and MCS
- Use the `-Z` option to display context information:

```
# ls -Z
# ps -Z
# id -Z
```

- The format is `user:role:type:level`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The SELinux context contains additional information such as SELinux user, role, type, and level. Access control decisions on processes, Linux users, and files are based on this context information.

To view the SELinux context information about files, use the `ls -Z` command:

```
# ls -Z
-rw-----. root root system_u:object_r:admin_home_t:s0 anaco...
```

To view the SELinux context information about processes, use the `ps -Z` command:

```
# ps -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 ... su
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 ... bash
```

To view the SELinux context associated with your Linux user, use the `id -Z` command:

```
# id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

SELinux context is displayed by using the following syntax:

`user:role:type:level`

## SELinux User

There are SELinux users in addition to the regular Linux users. SELinux users are part of an SELinux policy. The policy is authorized for a specific set of roles and for a specific MLS (Multi-Level Security) range. Each Linux user is mapped to an SELinux user as part of the policy. This allows Linux users to inherit the restrictions and security rules and mechanisms placed on SELinux users.

To define what roles and levels they can enter, the mapped SELinux user identity is used in the SELinux context for processes in that session. Use the SELinux Administration GUI to display user mapping. You can also view a list of mappings between SELinux and Linux user accounts from the command line:

| #          | <code>semanage login -l</code> |                |         |
|------------|--------------------------------|----------------|---------|
| Login Name | SELinux User                   | MLS/MCS Range  | Service |
| _default_  | unconfined_u                   | s0-s0:c0.c1023 | *       |
| root       | unconfined_u                   | s0-s0:c0.c1023 | *       |
| system_u   | system_u                       | s0-s0:c0.c1023 | *       |

Linux users are mapped to the SELinux \_default\_ login by default, which is mapped to the SELinux unconfined\_u user. The last column, MLS/MCS Range, is the level used by MLS and MCS (Multi-Category Security).

## Role

Role is an attribute of the Role-Based Access Control (RBAC) security model. The role serves as an intermediary between domains and SELinux users. SELinux users are authorized for roles, roles are authorized for domains, and processes run in their own separate domains. The roles determine which domains you can enter, and ultimately, which files you can access.

## Type

Type is an attribute of Type Enforcement (TE). The type defines a type for files, and defines a domain for processes. Processes are separated from each other by running in their own domains. This separation prevents processes from accessing files used by other processes, as well as preventing processes from accessing other processes. SELinux policy rules define how types can access each other, whether it is a domain accessing a type, or a domain accessing another domain.

## Level

Level is an attribute of MLS and MCS. An MLS range is a pair of levels, written as lowlevel-highlevel if the levels differ, or lowlevel if the levels are identical (s0-s0 is the same as s0). Each level is a sensitivity-category pair, with categories being optional. If there are categories, the level is written as sensitivity:category-set. If there are no categories, it is written as sensitivity.

If the category set is a contiguous series, it can be abbreviated. For example, c0.c3 is the same as c0,c1,c2,c3. The `/etc/selinux/targeted/setrans.conf` file is the Multi-Category Security translation table for SELinux and maps levels to human-readable form such as `s0 : c0 . c1023=SystemHigh`. Do not edit this file with a text editor; use the `semanage` command to make changes.

Use the `chcon` command to change the SELinux context for files. Changes made with the `chcon` command do not survive a file system relabel or the execution of the `restorecon` command. When using `chcon`, provide all or part of the SELinux context to change.

# Changing the Context File Type

- SELinux requires that KVM image files have the `virt_image_t` label applied to them.

```
# ls -dz /var/lib/libvirt/images
drwx... root root system_u:object_r:virt_image_t:s0
```

- To change the label on a new directory, `/virtstore`:

```
# semanage fcontext -a -t virt_image_t "/virtstore(/.*)?"
# restorecon -R -v /virtstore
# ls -dz /virtstore
drwx... root root system_u:object_r:virt_image_t:s0
```

- The `semanage` command adds the `/virtstore` directory to the SELinux targeted policy file.
- The `restorecon` command reads the policy file and updates the SELinux security contexts.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the lesson titled “Virtualization with Linux,” you learned that KVM virtual machine disk images are created in the `/var/lib/libvirt/images` directory by default. SELinux requires that image files have the `virt_image_t` label applied to them. You can use the `ls -Z` command to confirm that this label is applied to the `/var/lib/libvirt/images` directory:

```
# ls -dz /var/lib/libvirt/images
drwx... root root system_u:object_r:virt_image_t:s0 /var/...
```

You can use a different directory for your virtual machine images but you need to add the new directory to your SELinux policy and relabel it first. The following steps are used to add the `/virtstore` directory to the targeted SELinux policy and relabel the directory:

```
# semanage fcontext -a -t virt_image_t "/virtstore(/.*)?"
```

The above command adds the `/virtstore` directory to the SELinux policy by appending a line to the following file:

```
# cat /etc/selinux/targeted-contexts/files/file_contexts.local
/virtstore(/.*)? system_u:object_r:virt_image_t:s0
```

You still need to set the new security context on the directory and all files in the directory.

You can use any of the following commands to change the SELinux contexts on the /virtstore directory:

- **fixfiles**: Fixes the security context on file systems
- **restorecon**: Resets the security context on one or more files
- **setfiles**: Initializes the security context on one or more files

Each of these commands reads the files in /etc/selinux/targeted/contexts/files directory.

The following example shows the SELinux contexts before running the `restorecon` command:

```
# ls -dZ /virtstore
drwx... root root system_u:object_r:unlabeled_t:s0
```

Notice that the SELinux type is set to `unlabeled_t`. The following example runs the `restorecon` command to change the type as defined in the /etc/selinux/targeted/contexts/files/file\_contexts.local file:

```
# restorecon -R -v /virtstore
# ls -dZ /virtstore
drwx... root root system_u:object_r:virt_image_t:s0
```

There are also SELinux Booleans that affect KVM when launched by `libvirt`. Two of these Booleans are listed as follows:

- **`virt_use_nfs`**: Allow `virt` to manage NFS files.
- **`virt_use_samba`**: Allow `virt` to manage CIFS files.

These Booleans need to be enabled when using NFS or SAMBA shares, respectively, for storing virtual machine disk images. There are additional SELinux Booleans that affect KVM. Some of these are listed as follows:

```
# getsebool -a | grep virt
...
virt_use_commem --> off
virt_use_execmem --> off
virt_use_fusefs --> off
virt_use_nfs --> off
virt_use_rawip --> off
virt_use_samba --> off
virt_use_sanlock --> off
virt_use_usb --> on
virt_use_xserver --> off
```

## Confined SELinux Users

- Confined SELinux users and their associated domains:
  - `guest_u`: The domain for the user is `guest_t`.
  - `staff_u`: The domain for the user is `staff_t`.
  - `user_u`: The domain for the user is `user_t`.
  - `xguest_x`: The domain for the user is `xguest_t`.
- SELinux can confine Linux users by mapping Linux users to SELinux users.
- Use `semanage` to map a Linux user to an SELinux user:

```
# semanage login -a -s user_u newuser
```
- Booleans are available to change user behavior when running applications in home directories and in `/tmp`.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Linux users are mapped to the SELinux `_default_login` by default, which is mapped to the SELinux `unconfined_u` user. However, SELinux can confine Linux users, to take advantage of the security rules and mechanisms applied to them, by mapping Linux users to SELinux users.

A number of confined SELinux users exist in SELinux policy. The following is a list of confined SELinux users and their associated domains:

- `guest_u`: The domain for the user is `guest_t`.
- `staff_u`: The domain for the user is `staff_t`.
- `user_u`: The domain for the user is `user_t`.
- `xguest_x`: The domain for the user is `xguest_t`.

Linux users in the `guest_t`, `xguest_t`, and `user_t` domains can run set user ID (setuid) applications only if the SELinux policy permits it (such as `passwd`). They cannot run the `su` and `sudo` setuid applications to become the `root` user.

Linux users in the `guest_t` domain have no network access and can log in only from a terminal. They can log in with `ssh` but cannot use `ssh` to connect to another system.

The only network access Linux users in the `xguest_t` domain have is Firefox for connecting to webpages.

Linux users in the `xguest_t`, `user_t`, and `staff_t` domains can log in using the X Window System and a terminal.

By default, Linux users in the `staff_t` domain do not have permissions to execute applications with the `sudo` command.

By default, Linux users in the `guest_t` and `xguest_t` domains cannot execute applications in their home directories or `/tmp`, preventing them from executing applications in directories they have write access to. This helps prevent flawed or malicious applications from modifying files that users own.

By default, Linux users in the `user_t` and `staff_t` domains can execute applications in their home directories and `/tmp`.

### Mapping Linux Users to SELinux Users

Use the `semanage login -a` command to map a Linux user to an SELinux user. For example, to map the Linux `newuser` user to the SELinux `user_u` user, run the following command:

```
# semanage login -a -s user_u newuser
```

The `-a` option adds a new record and the `-s` option specifies the SELinux user. The last argument, `newuser`, is the Linux user that you want mapped to the specified SELinux user.

### Booleans for Users Executing Applications

Some Booleans are available to change user behavior when running applications in their home directories and in `/tmp`. Use the `setsebool -P <boolean> on|off` command:

To allow Linux users in the `guest_t` domain to execute applications in their home directories and `/tmp`:

```
# setsebool -P guest_exec_content on
```

To allow Linux users in the `xguest_t` domain to execute applications in their home directories and `/tmp`:

```
# setsebool -P xguest_exec_content on
```

To prevent Linux users in the `user_t` domain from executing applications in their home directories and `/tmp`:

```
# setsebool -P user_exec_content off
```

To prevent Linux users in the `staff_t` domain from executing applications in their home directories and `/tmp`:

```
# setsebool -P staff_exec_content off
```

## SELinux Utilities: Summary

- **sestatus**: Displays SELinux status
- **semanage**: Is an SELinux policy management tool
- **semodule**: Manages SELinux policy modules
- **setsebool**: Sets SELinux Boolean value
- **avcstat**: Displays SELinux AVC statistics
- **getenforce**: Reports the current SELinux mode
- **getsebool**: Reports SELinux Boolean values
- **seinfo**: Is an SELinux policy query tool
- **seseach**: Is an SELinux policy query tool
- **fixfiles**: Fixes the security context on file systems
- **restorecon**: Resets the security context on files



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide lists some of the more commonly used command-line utilities for managing and operating SELinux. More utilities are provided. The `policycoreutils` package installs the following utilities:

- **fixfiles**: Fixes the security context on file systems
- **load\_policy**: Loads a new SELinux policy into the kernel
- **restorecon**: Resets the security context on one or more files
- **setfiles**: Initializes the security context on one or more files
- **secon**: Displays the SELinux context from a file, program, or user input
- **semodule\_package**: Creates an SELinux policy module package
- **restorecond**: Is a daemon that watches for file creation and sets the default file context
- **semodule**: Manages SELinux policy modules
- **sestatus**: Displays SELinux status
- **setsebool**: Sets SELinux Boolean value

The `libselinux-utils` package installs the following utilities:

- `avcstat`: Displays SELinux AVC statistics
- `getenforce`: Reports the current SELinux mode
- `getsebool`: Reports SELinux Boolean values
- `matchpathcon`: Queries the system policy and displays the default security context associated with the file path
- `selinuxconlist`: Displays all of the SELinux context reachable for a user
- `selinuxdefcon`: Displays the default SELinux context for a user
- `selinuxenabled`: Indicates whether SELinux is enabled
- `setenforce`: Modifies the SELinux mode

The `setools-console` package installs the following utilities:

- `findcon`: An SELinux file context search tool
- `sechecker`: An SELinux policy checking tool
- `sediff`: An SELinux policy difference tool
- `seinfo`: An SELinux policy query tool
- `sesearch`: An SELinux policy query tool

The `policycoreutils-python` package installs the following utilities:

- `semanage`: Is an SELinux policy management tool
- `audit2allow, audit2why`: Generates SELinux policy `allow/don't_audit` rules from logs of denied operations
- `chcat`: Changes or removes the security category for each file or user
- `sandbox`: Runs a command in an SELinux sandbox
- `semodule_package`: Creates an SELinux policy module package

The `policycoreutils-gui` package installs the following utilities:

- `system-config-selinux`: SELinux Administration GUI
- `selinux-polengui`: SELinux policy generation tool

## Quiz



A given SELinux policy can be customized by enabling or disabling which of the following?

- a. Modes
- b. Policies
- c. Booleans
- d. Contexts
- e. Labels
- f. Users



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe SELinux
- Describe SELinux packages
- Use the SELinux Administration GUI
- Describe SELinux modes
- Describe SELinux policies
- Describe SELinux Booleans
- Describe SELinux file labeling, context, and users
- Use SELinux utilities



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 17: Overview

The practices for this lesson cover the following:

- Exploring SELinux files and directories
- Configuring SELinux Booleans
- Configuring SELinux Context



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.



# Core Dump Analysis

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe Kexec and Kdump
- Configure Kdump to capture kernel vmcore dump
- Describe kernel parameters that can cause a panic
- Use magic SysRq keys
- Use the `crash` utility for analyzing core dumps



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# System Core Collection: Kexec and Kdump

- Use Kexec and Kdump to ensure the creation of reliable kernel vmcores for diagnostic purposes.
- To use Kdump, install the following package:

```
# yum install kexec-tools
```

- To enable Kdump, reserve a portion of the system memory for the capture kernel.
  - Use the `crashkernel=<size>` boot parameter.  
Append to the kernel line in the GRUB configuration file.

```
linux16 /vmlinuz-3.8.13-68.3.3.el7uek ...
crashkernel=128M
```

- Enable the Kdump service and reboot the system:

```
# systemctl enable kdump
# systemctl reboot
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Kdump is the Linux kernel crash dumping mechanism. In the event of a system crash, Kdump provides a memory dump (vmcore) image. This image can assist in determining the cause of the crash. It is highly recommended that you enable the Kdump feature.

Kexec and Kdump together ensure faster bootup and the creation of reliable kernel vmcores for diagnostic purposes. Kexec is a fast-boot mechanism that allows booting a Linux kernel from the context of an already running kernel without going through BIOS. Kdump uses Kexec to boot into a second kernel whenever the system crashes. The crash dump is captured from the context of a freshly booted kernel and not from the context of the crashed kernel. This second kernel boots with very little memory and captures the dump image.

To enable and use Kdump, install the following package:

```
# yum install kexec-tools
```

Enabling Kdump requires you to reserve a portion of the system memory for the capture kernel. This portion of memory is unavailable for other uses. The amount of memory that is reserved for the Kdump kernel is represented by the `crashkernel` boot parameter. This is appended to the kernel line in the GRUB configuration file, `/boot/grub2/grub.cfg`. The following example enables Kdump and reserves 128 MB of memory:

```
linux16 /vmlinuz-3.8.13-68.3.3.el7uek ... crashkernel=128M
```

In addition to reserving memory, you can designate the starting address (offset) of this reserved memory. For example, adding the following option to the kernel line reserves 128 MB of memory, starting at physical address 0x01000000 (16 MB):

```
crashkernel=128M@16M
```

To set the offset to 48M:

```
crashkernel=128M@48M
```

If you have more than 128 GB RAM, use the following setting:

```
crashkernel=512M@64M
```

If more control is needed over the size and placement of the reserved memory, use the following format:

```
crashkernel=range1:size1[,range2:size2,...] [@offset]
```

The range<n> value specifies a range of values that are matched against the amount of physical RAM present in the system. The corresponding size<n> value specifies the amount of Kexec memory to reserve.

The following example tells Kexec to reserve 64 MB of RAM if the system contains between 512 MB and 2 GB of memory. If the system contains more than 2 GB of physical memory, reserve 128 MB:

```
crashkernel=512M-2G:64M,2G-:128M
```

On x86\_64 systems with at least 2 GB of memory, you can allocate memory for kdump automatically. Use the following parameter to automatically allocate memory for kdump:

```
crashkernel=auto
```

After adding the `crashkernel` parameter to the `/boot/grub2/grub.cfg` file, reboot your system so that memory is reserved for the capture kernel. The `free -m` command correctly shows that less memory is available for the system.

Use the `systemctl` command to enable the Kdump service to start at boot time. Use the `systemctl` command to start the Kdump service.

```
# systemctl enable kdump
# systemctl start kdump
```

This loads your kernel-kdump image via Kexec, leaving your system ready to capture a vmcore on crashing. You can test by force-crashing your system using the following command:

```
# echo c > /proc/sysrq-trigger
```

This causes panic output to be displayed, followed by the system restarting into the Kdump kernel. When the boot process gets to the point where it starts the Kdump service, the vmcore is copied to disk to the default location, `/var/crash/<YYYY-MM-DD-HH:MM>/vmcore`. The system then reboots back into the normal kernel.

Note that Kdump is not supported on Xen domU guests. Virtualized systems can use the `xm dump-core` command for panics.

# Kdump Configuration File

- The configuration file for Kdump is `/etc/kdump.conf`.
- By default, the target location to write the `vmcore` is the `/var/crash` directory on the local system.
  - Set the `path` directive to change this.
- Capture only the bare essentials for kernel fault debugging and compress the core:
 

```
core_collector makedumpfile -d 31 -c
```

  - This strips zero pages, pagecache pages, and user pages.
  - Compressing turns a 64 GB core into around 500 MB.
  - Always compress before uploading to Oracle Support.
- The default action to take, in case dumping to an intended target fails, is to reboot.
  - Set the `default` directive to change this.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The configuration file for Kdump is `/etc/kdump.conf`. The default target location for the `vmcore` is the `/var/crash` directory on the local file system, which is represented as follows:

```
path /var/crash
```

To write to a different local directory, edit the `path` directive and provide the absolute path. Example:

```
path /
```

To write directly to a device, edit the `raw` directive and specify the device name. Example:

```
raw /dev/sdal
```

To write to a remote system by using NFS, use the `nfs` directive followed by the FQDN of the remote system, then a colon (:), and then the directory path. Example:

```
nfs host01.example.com:/export/crash
```

To write to a remote machine by using SSH, use the `ssh` directive followed by a valid username, the @ sign, and the host name, in that order. Example:

```
ssh root@host02.example.com
```

Modify the filtering level for the vmcore dump using the `core_collector` directive in the `/etc/kdump.conf` file. To exclude certain pages from the dump, use the `-d <value>` parameter where `<value>` is a sum of values of the pages that you want to exclude. Use the following values for the pages:

- 1: zero page
- 2: cache page
- 4: cache private
- 8: user data
- 16: free page

The recommendation is to exclude all these pages as follows. Add the values (the total for all is 31) and provide the sum as the argument to the `-d` (dump level) option:

```
core_collector makedumpfile -d 31 -c
```

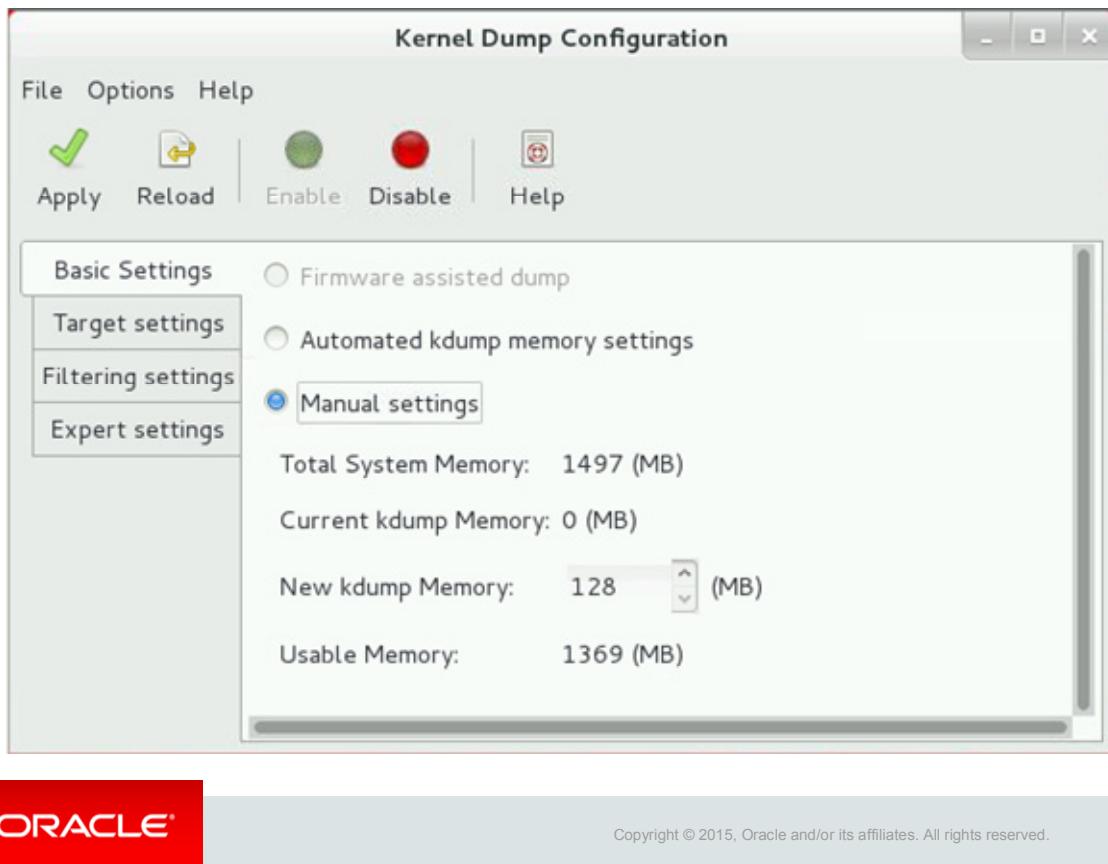
The `-c` option enables dump file compression. To exclude only zero (1) and free (16) pages:

```
core_collector makedumpfile -d 17 -c
```

The default action to take if dumping to the intended target fails is to reboot. Other possible actions are `halt`, `poweroff`, `shell`, or `dump_to_rootfs`, which means dump vmcore to `rootfs` from `initramfs` context and reboot. To change this, set the `default` directive in `/etc/kdump.conf`, as in this example:

```
default poweroff
```

# Kdump Setup Configuration GUI



You can also enable Kdump from a GUI. Enter the following command to use the Kernel Dump Configuration GUI:

```
# system-config-kdump
```

The GUI appears as shown in the slide. Click the Enable button to configure the `kdump` daemon to start at boot time.

Four tabs appear on the left side of the GUI. The Basic Settings tab allows you to select the amount of memory to reserve for Kdump.

Use the Target Settings tab to specify the target location for the vmcore dump. You can store the dump image in a local file system or store it remotely using NFS or SSH. The default is to store the vmcore file in the `/var/crash` directory of the local file system. The following targets are supported:

- **Raw device:** All locally attached raw disks and partitions
- **Local file system:** Any ext2, ext3, ext4, btrfs, or xfs file system on directly attached disk drives, hardware RAID logical drives, LVM devices, and mdraid arrays
- **Remote directory:** Remote directories accessed by using NFS or SSH over IPv4 and remote directories accessed using iSCSI over software initiators

Unsupported targets include:

- Remote directories on the `rootfs` file system accessed using NFS
- Remote directories accessed using iSCSI over hardware initiators
- Remote directories accessed over IPv6
- Remote directories accessed using SMB/CIFS or FCoE (Fibre Channel over Ethernet)
- Remote directories accessed using wireless NICs
- Multipath-based storage

The Filtering Settings tab allows you to select the filtering level for the vmcore dump. You can choose to exclude any or all of the following from the dump:

- zero page
- cache page
- cache private
- user data
- free page

The Expert Settings tab allows you to choose which kernel and initial RAM disk to use. From this tab you can also customize the options that are passed to the kernel and the core collector program. You can choose what to do when dumping to the intended target fails. The following options are available:

- **reboot**: Reboot the system and lose the core that you are trying to retrieve. This is the default action.
- **halt**: Halt the system after attempting to capture a vmcore, regardless of success or failure.
- **poweroff**: Power the system off.
- **shell**: Drop to an interactive shell session inside `initramfs` from where you can try to record the core manually. Exiting the shell reboots the system.
- **dump to rootfs and reboot**: Dump vmcore to `rootfs` from `initramfs` context and reboot.

Click Apply to save any changes.

# Kernel Tuning Parameters

- The following parameters can conditionally bring down the system.
- Out of memory:
  - `vm.panic_on_oom` – Panics when out of memory
  - `kernel.panic_on_oops` – Panics when a BUG occurs
- Hung Task Monitoring
  - `kernel.hung_task_timeout_secs` – Displays a message if tasks are not scheduled within the timeout period
  - `kernel.hung_task_panic` – Panics if the preceding condition is hit
  - Gives a point-in-time crash core for further analysis



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `vm.panic_on_oom` parameter (`/proc/sys/vm/panic_on_oom`) enables or disables a kernel panic in an out-of-memory (OOM) situation. When set to 0 (default), the kernel's OOM-killer scans through the entire task list and attempts to kill some rogue memory-hogging process to avoid a panic. When set to 2, the kernel always panics when an OOM condition occurs. When set to 1, the kernel normally panics but can survive in certain conditions. If a process limits allocations to certain nodes using memory policies or cpusets, and those nodes reach memory exhaustion status, one process can be killed by the OOM-killer. No panic occurs in this case because other nodes' memory might be free. This means that the system as a whole might not have reached an out-of-memory condition yet. Settings of 1 and 2 are for failover of clustering. Select the setting according to your failover policy.

The `kernel.panic_on_oops` parameter (`/proc/sys/kernel/panic_on_oops`) controls the kernel's behavior when an oops or BUG is encountered. When set to 0, the system tries to continue operations. When set to 1 (default), the system delays a few seconds (to give the `klogd` kernel log daemon time to record the oops output), and then panics.

The `kernel.hung_task_timeout_secs` parameter (in the `/proc/sys/kernel/` directory) is set to 120 by default. This causes a message to be generated when a task is stuck in D (disk sleep) state for 120 seconds. A process is put in D state while waiting for `read()` or `write()` return. A process cannot be killed or interrupted while in D state.

If the `kernel.hung_task_panic` parameter (in the `/proc/sys/kernel/` directory) is enabled by setting the value to 1, it causes the kernel to panic if any user or kernel thread sleeps in state `TASK_UNINTERRUPTIBLE` for more than `kernel.hung_task_timeout_secs` seconds. The default setting for the `kernel.hung_task_panic` parameter is 0, or disabled.

# Magic SysRq Keys

| Magic SysRq Key                                                                   | How to Invoke Magic SysRq                                                                                                  |
|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| M: Dump <b>system memory</b> statistics.                                          | <b>Console:</b><br>Alt + Sys Rq + <cmd>                                                                                    |
| P, W: Dump the stack for all <b>processors</b> .                                  | <b>Serial Console:</b><br><Break> <cmd>                                                                                    |
| T: Dump the <b>kernel stack trace</b> for all processes.                          | <b>Command Line:</b><br>echo t > /proc/sysrq-trigger                                                                       |
| C: Immediately cause a system <b>crash</b> .                                      | <b>Oracle VM dom0:</b><br>xm sysrq <domain ID> <cmd>                                                                       |
| S ... U... B: Emergency Sync all disks, Unmount disks, reboot                     | <b>Ensure kernel.sysrq = 1</b>                                                                                             |
| Some of these operations (such as stack trace) dump a lot of data (1 MB or more). | <b>These operations take full priority in the kernel. Use these carefully. Do not run them in your monitoring scripts.</b> |



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The magic SysRq (system request) key is a key combination that is understood by the Linux kernel, which allows you to execute low-level commands regardless of the system's state. It is often used to recover from, or diagnose, a system hang. The slide lists some of the common uses and the method to invoke. The left column lists the Magic SysRq keys and right column lists the different methods of invoking the keys.

The magic SysRq feature is controlled by the `kernel.sysrq` kernel parameter (`/proc/sys/kernel/sysrq`). Following is the list of possible values:

- 0 – Disable SysRq completely
- 1 – Enable all functions of SysRq
- 2 – Enable control of console logging level
- 4 – Enable control of keyboard (SAK, unraw)
- 8 – Enable debugging dumps of processes etc.
- 16 – Enable sync command
- 32 – Enable remount read-only
- 64 – Enable signalling of processes (term, kill, oom-kill)
- 128 – Allow reboot/poweroff
- 256 – Allow nicing of all RT tasks

## crash Utility

- Use the `crash` utility to analyze the state of the system while it is running or after a kernel crash has occurred.
  - Crash is merged with the GNU Debugger (`gdb`).
  - Crash can analyze core dumps created by multiple facilities.
- Syntax for `crash`:
 

```
# crash <vmcore> <kernel-debug data>
```

  - `vmcore` – The memory image
  - `vmlinux` – Part of the `kernel-debuginfo` package
- Download the `kernel-debuginfo` package from [oss.oracle.com](http://oss.oracle.com).
- The `vmlinux` file is installed in the `/usr/lib/debug/lib/modules/<kernel>` directory.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `crash` utility allows you to analyze the state of the Linux system while it is running or after a kernel crash has occurred. The utility has been merged with `gdb`, the GNU Debugger, so `crash` includes source code-level debugging capabilities. The `crash` utility is used to analyze core dumps created by the `kdump`, `netdump`, `diskdump`, `xm dump-core`, Linux Kernel Crash Dumps (LKCD), and `vissh dump` facilities. The syntax for `crash` is as follows:

```
crash [options] [vmlinux] [vmcore]
```

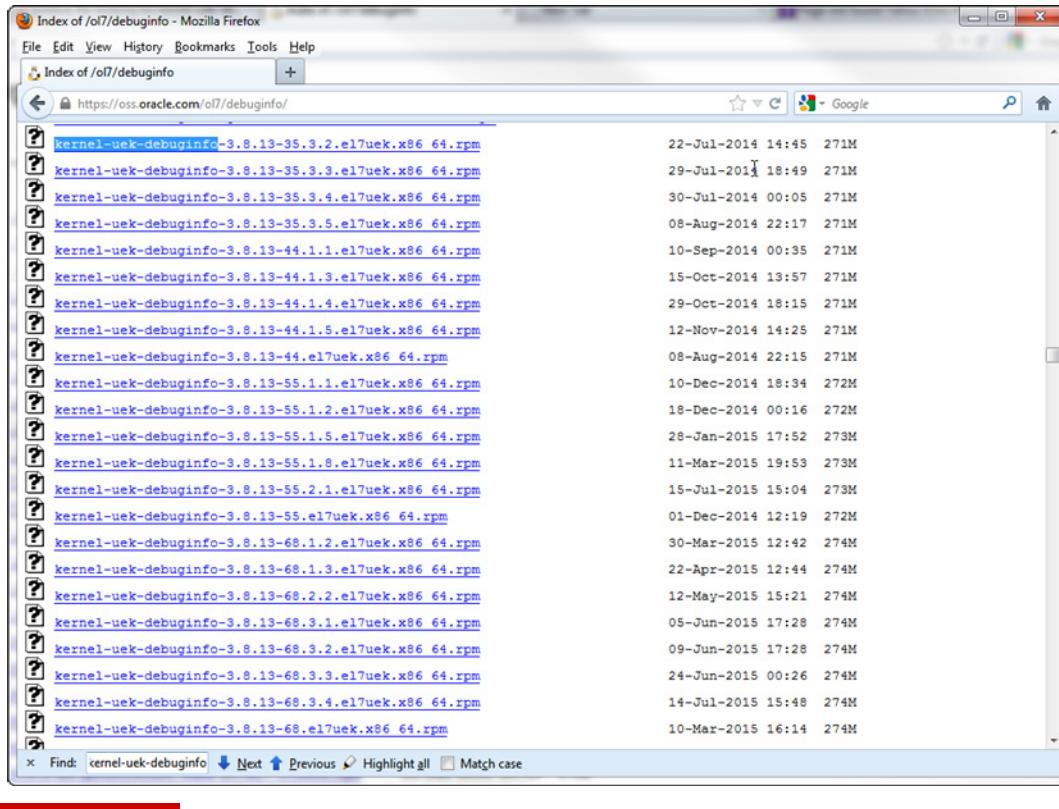
Optional `<vmlinux>` and `<vmcore>` arguments:

- `vmlinux` – A `vmlinux` kernel object file, often referred to as the *namelist*. The `vmlinux` file is part of the `kernel-debuginfo` package. This file is installed in the `/usr/lib/debug/lib/modules/<kernel>` directory.
- `vmcore` – The memory image, which is a kernel crash dump file created by any of the supported core dump facilities. Omit the `vmcore` argument to invoke the `crash` utility on a live system.

The following is an example with both `<vmlinux>` and `<vmcore>` arguments:

```
# crash /usr/lib/debug/lib/modules/3.8.13-
55.1.6.el7uek.x86_64/vmlinux /root/2015-0721-2154.29-
host03.59.core
```

# Downloading kernel-debuginfo RPM Packages



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To use the `crash` utility, you must install the `crash` and `kernel-debuginfo` RPM packages. The `crash` RPM is included in the Oracle Linux distribution. Download the `kernel-debuginfo` RPM packages from <https://oss.oracle.com/> as shown in the slide.

You need to determine the kernel version and architecture of the vmcore dump file to know which `kernel-debuginfo` RPM packages to download and install. If you have access to the system that produced the vmcore, use the `uname -r` command to determine the kernel version and architecture as follows:

```
# uname -r
3.8.13-55.1.6.el7uek.x86_64
```

In this example, the kernel version is `3.8.13-55.1.6.el7uek` and the architecture is `x86_64`. If you have only the vmcore file, use the `strings <vmcore>` command as follows:

```
# strings 2015-0721-2154.29-host03.59.core | grep -i boot_image
BOOT_IMAGE=/vmlinuz-3.8.13-55.1.6.el7uek.x86_64 ...
...
```

This example pipes the output to `grep -i boot_image` to show the kernel version and system architecture.

Download the matching `debuginfo` and `debuginfo-common` RPMs for the desired kernel version and architecture. You might need to check multiple URLs to locate the correct kernel version and architecture.

For Oracle Linux 5, browse to:

- <http://oss.oracle.com/el5/debuginfo>

For Oracle Linux 6, browse to:

- <http://oss.oracle.com/el6/debuginfo>

For Oracle Linux 7, browse to:

- <https://oss.oracle.com/ol7/debuginfo>

The slide highlights the `debuginfo` RPM package for kernel version `3.8.13-55.1.6.el7uek.x86_64`. For this kernel version and architecture, download the following two packages:

`kernel-uek-debuginfo-3.8.13-55.1.6.el7uek.x86_64`

`kernel-uek-debuginfo-common-3.8.13-55.1.6.el7uek.x86_64`

You can either extract only the `vmlinu`x executable from the `kernel-debuginfo` RPM, or install both RPMs on the system on which you are analyzing the core dump file.

Use the `yum update` command to install the latest version of `crash`, which supports dumps of systems running UEK2 and above:

```
# yum update crash
```

## Initial crash Output

```
KERNEL: /usr/lib/debug/lib/modules/3.8.../vmlinu
DUMPFILE: /root/2015-0721-2154.29-host03.59.core
CPUS: 1
DATE: Thu Jul 21 15:54:36 2015
UPTIME: 11:19:18
LOAD AVERAGE: 0.00, 0.01, 0.05
TASKS: 215
NODENAME: host03.example.com
RELEASE: 3.8.13-55.1.6.el7uek.x86_64
VERSION: #2 SMP Wed Feb 11 14:18:22 PST 2015
MACHINE: x86_64 (2992 Mhz)
MEMORY: 1.5 GB
PANIC: ""
PID: 0
COMMAND: "swapper"
TASK: ffffffff818ae420 [THREAD_INFO: ...]
CPU: 0
STATE: TASK_RUNNING (ACTIVE)
WARNING: panic task not found
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide displays an example of information that is initially displayed when `crash` is run. You can learn a lot about the state of a system from this initial output. The number of CPUs and the load average over the last 1 minute, last 5 minutes, and last 15 minutes are displayed. The number of tasks running, the amount of memory, the panic string, the command executing at the time the core dump was created, and additional information are displayed. In this example, the system did not panic. The core dump file was created using the `xm dump-core` command. The following example displays some of the initial `crash` information for an actual panic:

```
DUMPFILE: tmp/vmcore
PANIC: "Oops: 0002" (check log for details)
PID: 1696
COMMAND: "insmod"
TASK: c74de000
CPU: 0
STATE: TASK_RUNNING (PANIC)
```

In this example, an `insmod` attempt to install a module resulted in an “oops” violation.

## Using the crash Utility

- The `crash` utility provides an interactive prompt from which commands are executed:

```
crash>
```

- Use the `?` or `help` command to view a list of commands.
- Crash commands fall into the following categories:
  - Symbolic display – Displays kernel text and data structures
  - System state – Includes kernel-aware commands that examine various kernel subsystems on a system-wide or per-task basis
  - Utility functions – Include useful helper commands such as a calculator, translator, and search function
  - Session control – Includes commands that control the crash session itself



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After the initial `crash` output is displayed, the `crash` utility provides an interactive prompt from which `crash` commands are executed:

```
# crash /usr/lib/debug/lib/modules/3.8.13-
55.1.6.el7uek.x86_64/vmlinuX /root/2014-0220-0525.00-
host03.249.core
...
crash>
```

From the `crash>` prompt, you can enter `help` or `?` to display the `crash` commands. You can also enter `help <command>` to display usage information for a specific command. Each `crash` command falls into one of the categories listed in the slide.

# Symbolic Display crash Commands

This category of `crash` commands displays kernel data structures. Available commands are:

- `struct` – Displays a structure definition or the contents of a structure at a specified address
- `union` – Is the same as `struct`, but used for kernel data types that are defined as unions instead of structures
- `*` – Is the “Pointer-to” command that is used instead of `struct` or `union`. The `gdb` module determines whether the argument is a structure or a union, and then calls the appropriate function.
- `p` – Displays the contents of a kernel variable
- `sym` – Translates a kernel symbol name to its kernel virtual address or vice versa
- `dis` – Disassembles the source code instructions of a kernel function



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following symbolic display `crash` commands take advantage of the `gdb` integration to display kernel data structures symbolically:

- `struct` – Displays a structure definition or provides a formatted display of the contents of a structure at a specified address. Example:

```
crash> struct cpu
struct cpu {
    int node_id;
    int hotpluggable;
    struct device dev;
}
```

- `union` – Is the same as the `struct` command, but is used for kernel data types that are defined as unions instead of structures
- `*` – Is the “Pointer-to” command, which can be used instead of entering `struct` or `union`. The `gdb` module first determines whether the argument is a structure or a union, and then calls the appropriate function.

The following is an example of using the “Pointer-to” command:

```
crash> *page
struct page {
    unsigned long flags;
    struct address_space *mapping;
    struct {
        union {
            unsigned long index;
        };
    ...
}
```

SIZE: 64

- p – Displays the contents of a kernel variable. The arguments are passed on to the `gdb print` command for proper formatting. Example:

```
crash> p init_mm
init_mm = $1 = {
    mmap = 0x0,
    mm_rb = {
        rb_node = 0x0
    };
    mmap_cache = 0x0,
    ...
}
```

- whatis – Displays the definition of structures, unions, typedefs, or text/data symbols. Example:

```
crash> whatis linux_binfmt
struct linux_binfmt {
    struct list_head lh;
    struct module *module;
    int (*load_binary)(struct linux_binprm *);
    int (*load_shlib)(struct file *);
    int (*core_dump)(struct coredump_params *);
};
```

- sym – Translates a kernel symbol name to its kernel virtual address and section, or a kernel virtual address to its symbol name and section. You can also use this command to dump the complete list of kernel symbols (-l), or to query (-q) the symbol list for all symbols containing a given substring.
- dis – Disassembles the source code instructions of a complete kernel function, or from a specified address for a given number of instructions, or from the beginning of a function up to a specified address

## System State crash Commands

These crash commands are “kernel-aware” commands that display various kernel subsystems on a systemwide or per-task basis. Available commands are:

- **bt** – Displays a kernel stack backtrace of the current context. This is probably the most useful `crash` command.
- **dev** – Displays character and block device data, I/O port and memory usage (`-i`), and disk I/O statistics (`-d`)
- **files** – Displays information about open files
- **fuser** – Displays file users, or the tasks that reference a specified file name or inode address
- **irq** – Displays IRQ (interrupt request) data
- **kmem** – Displays several kernel memory subsystems such as `/proc/slabinfo` data at the time of the crash (`-s`)



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following system state `crash` commands are “kernel-aware” commands that display various kernel subsystems on a systemwide or per-task basis:

- **bt** – Displays a kernel stack backtrace of the current context. This is probably the most useful `crash` command. Because the initial context is the panic context, it shows the function trace leading up to the kernel panic. Example:

```
crash> bt
PID: 0  TASK: ffffffff818ae420  CPU: 0  COMMAND: "swapper/0"
#0 [ffffffffff8189be38] __schedule at ffffffff81591d12
#1 [ffffffffff8189bec0] default_idle at ffffffff8101e35d
#2 [ffffffffff8189bee0] cpu_idle at ffffffff8101dc19
```

- **dev** – Displays character and block device data. Example:

```
crash> dev
CHRDEV  NAME          CDEV      OPERATIONS
      1   mem          ffff88005c803840  memory_fops
      4  /dev/vc/0     ffffffff81dc7560  console_fops
      ...

```

- **files** – Displays information about open files. This command is context-sensitive, meaning that it acts on the current context unless a PID or task address is specified as an argument. Example:

```
crash> files
PID: 0 TASK: ffff88005b290680 CPU: 0 COMMAND: "crash"
ROOT: / CWD: /
FD          FILE          DENTRY          INODE
0 ffff8800599c... Ffff88003e65... Ffff88005a11... CHR /dev/pts/0
...
...
```

- **fuser** – Displays the tasks that are using the specified files or sockets. Example:

```
crash> fuser /dev/pts/0
PID          TASK          COMM          USAGE
2394  ffff88005b3544c0  "bash"        fd
3470  ffff88005b3544c0  "crash"       fd
...
```

- **irq** – Displays interrupt request (IRQ) data. Example:

```
crash> irq
IRQ      IRQ_DESC/_DATA      IRQACTION      NAME
0      ffff88005da74080  ffffffff81786000  "timer"
...
...
```

- **kmem** – Displays the state of several kernel memory subsystems. This command accepts a number of options. For example, the **-i** option displays general memory usage information:

```
crash> kmem -i
          PAGES      TOTAL      PERCENTAGE
TOTAL MEM  384258    1.5 GB      -----
          FREE     22955   89.7 MB      5% of TOTAL MEM
          USED    361303   1.4 GB      94% of TOTAL MEM
...
TOTAL SWAP 770047    2.9 GB      -----
SWAP USED   45      180 KB      0% of TOTAL SWAP
SWAP FREE  770002    2.9 GB      99% of TOTAL SWAP
crash> kmem -s
CACHE          NAME      OBJSIZE ALLOCATED TOTAL SLABS SSIZE
ffff88005b762fc0  rpc_buffers    2048          8      8      4      4k
ffff88005ae62f80  rpc_tasks      256          8     15      1      4k
...
...
```

# System State crash Commands

Additional “kernel-aware” commands:

- `log` – Displays the kernel message buffer chronologically. This command gives more data than the `dmesg` command.
- `mach` – Displays machine-specific data such as `cpuinfo` structure (`-c`) and physical memory map (`-m`)
- `mod` – Displays the list of currently installed kernel modules and can also load (`-s` or `-S`) and unload (`-d`) debug data
- `mount` – Displays information about currently mounted file systems
- `net` – Displays network-related data
- `ps` – Displays process status information
- `pte` – Translates the contents of a page table entry (PTE)
- `rung` – Displays the list of tasks on the run queue



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Additional system state category of crash commands:

- `log` – Displays the kernel message buffer chronologically. This is the same data that is displayed with the `dmesg` command but this output can include console logs that never reached the console ringbuffer and messages that were not written to syslog/disk.
- `mach` – Displays machine-specific data such as `cpuinfo` structure (`-c`) and physical memory map (`-m`). Example:

```
crash> mach
MACHINE TYPE: x86_64
MEMORY SIZE: 1.5 GB
CPUS: 1
HYPERVISOR: Xen HVM
PROCESSOR SPEED: 2992 Mhz
...
```

- `mod` – Displays the list of currently installed kernel modules. With the `-s` or `-S` options, it loads the debug data from the module object files if they are available, allowing the symbolic debugging capability of kernel modules. The `-d` option deletes the symbolic and debugging data of the specified module.

- **mount** – Displays information about currently mounted file systems such as `vfsmount` structure address, `super_block` structure address, file system type, device name, and mount point. Example:

```
crash> mount
      VFSMOUNT          SUPERBLK        TYPE     DEVNAME   DIRNAME
fffff88005c735d80 fffff88005c738800  rootfs   rootfs    /
fffff88005874cdc0 fffff88005c741c00  proc     proc      /proc
...
...
```

- **net** – Displays network-related data such as ARP cache, open network socket addresses, `net_device` address, IP addresses of each network device, and other information. Example:

```
crash> net
      NET_DEVICE      NAME      IP ADDRESS (ES)
fffff88005c30e000 lo        127.0.0.1
fffff88005bb28000 eth0     192.0.2.105
...
...
```

- **ps** – Displays process status information, but not as comprehensively as the `bt` command. The active task is highlighted by `>`. Example:

```
crash> ps
  PID  PPID CPU      TASK           ST %MEM   VSZ   RSS  COMM
> 0    0   0  ffffffff81781020 RU  0.0      0     0 [swapper]
  1    0   0  fffff88005c09a040 IN  0.1  19396 1424  init
...
...
```

- **pte** – Translates the hexadecimal contents of a page table entry (PTE) into its physical page address and page bit settings. If the PTE references a swap location, it displays the swap device and offset. Example:

```
crash> pte ffff88005b762fc0
      PTE          PHYSICAL      FLAGS
fffff88005b762fc0  f88005b762000 (DIRTY|PROTNONE|GLOBAL|NX)
crash> pte ffff88005ae62f80
```

| PTE               | PHYSICAL      | FLAGS             |
|-------------------|---------------|-------------------|
| fffff88005ae62f80 | f88005ae62000 | (DIRTY GLOBAL NX) |

- **rung** – Displays the list of tasks on the run queue of each CPU. Example:

```
crash> rung
CPU 0 RUNQUEUE: ffff88005fc121c0
      CURRENT: PID: 0  TASK: ffffffff81781020  COMMAND: "swapper"
      RT PRIO_ARRAY: ffff88005fc12310
      [no tasks queued]
```

# System State crash Commands

Additional “kernel-aware” commands:

- `sig` – Displays a task’s signal-handling data. You can include multiple task or PID numbers as arguments.
- `swap` – Displays information for each configured swap device. The `swapon -s` command displays the same data.
- `sys` – Displays the system information shown during crash initialization, or the system call table entries (-c)
- `task` – Displays the contents of `task_struct`. Multiple task or PID numbers can be entered.
- `timer` – Displays the timer queue entries
- `vm` – Displays a task’s virtual memory data
- `vtop` – Translates a user or kernel virtual address to its physical address
- `waitq` – Displays tasks blocked on the specified wait queue



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The remaining system state category of `crash` commands:

- `sig` – Displays a task’s signal-handling data. You can include multiple task or PID numbers as arguments. Example:

```
crash> sig
PID: 0  TASK: ffffffff81781020  COMMAND: "swapper"
SIGNAL_STRUC: ffffffff81782960  NR_THREADS: 1
      SIG      SIGACTION          HANDLER          MASK          FLAGS
      [1]  ffffffff81782d88  SIG_DFL  0000000000000000  0
      [2]  ffffffff81782da8  SIG_DFL  0000000000000000  0
...

```

- `swap` – Displays information for each configured swap device. The `swapon -s` command displays the same data in a slightly different format. Example:

```
crash> swap
  FILENAME        TYPE        SIZE      USED     PCT    PRIORITY
  /dm-1          PARTITION  3080188k    0k     0%      -1
```

- **sys** – Displays the system information shown during `crash` initialization, or the system call table entries (-c). Example:

```
crash> sys -c
NUM   SYSTEM CALL
  0  sys_read           .../fs/read)write.c: 403
  1  sys_write          .../fs/read)write.c: 421
```

- **task** – Displays the contents of `task_struct`. Each `task_struct` data structure describes a process or task in the system. You can enter multiple task or PID numbers. Example:

```
crash> task
PID: 0  TASK: ffffffff81781020  CPU: 0    COMMAND: "swapper"
struct task_struct {
    state = 0,
    stack = 0xffffffff81776000,
    usage = {
        counter = 2
    },
    flags = 4202752,
    ...
}
```

- **timer** – Displays the timer queue entries in chronological order
- **vm** – Displays a task's virtual memory data, including `mm_struct` address, page directory address, resident set size, and total virtual memory size. Example:

```
crash> vm
PID: 0  TASK: ffff88005b352680  CPU: 0    COMMAND: "crash"
          MM             PGD            RSS      TOTAL_CM
ffff880037fa62c0  ffff8800589bb000  165980k  301640k
          VMA            START          END      FLAGS   FILE
ffff88005bb69608  400000         a07000  8001875  /usr/bin/crash
    ...

```

- **vtop** – Translates a user or kernel virtual address to its physical address. Other information such as the PTE translation, the `vm_area_struct` data, and the `mem_map` page data is displayed. Example:

```
crash> vtop ffff88005b352680
          VIRTUAL          PHYSICAL
ffff88005b352680  5b352680
    ...

```

- **waitq** – Displays tasks blocked on the specified wait queue

## Utility crash Commands

The utility `crash` commands are helper commands. Available commands are:

- `ascii` – Translates a hexadecimal value to ASCII
- `btop` – Translates a hexadecimal value to its page number
- `eval` – Evaluates an expression
- `list` – Displays the contents of a linked list of structures
- `ptob` – Translates a page number to its physical address
- `ptov` – Translates a hexadecimal physical address into a kernel virtual address
- `search` – Searches memory space for a specified value
- `rd` – Reads or displays a specified amount of memory
- `wt` – Writes or modifies the contents of memory



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following utility `crash` commands are helper commands that serve a variety of functions:

- `ascii` – Translates a hexadecimal value to ASCII. If no argument is entered, an ASCII chart is displayed.
- `btop` – Translates a hexadecimal address to its page number
- `eval` – Evaluates an expression and displays the result in hexadecimal, decimal, octal, and binary. This command can also function as a calculator.
- `list` – Displays the contents of a linked list of structures
- `ptob` – Translates a page number to its physical address (byte value)
- `ptov` – Translates a hexadecimal physical address into a kernel virtual address
- `search` – Searches for a specified value within the user virtual, kernel virtual, or physical memory space. You can also provide a starting and ending point to search.
- `rd` – Reads or displays a specified amount of user, kernel, or physical memory in a specified format
- `wt` – Writes or modifies the contents of memory. Use this command with great care.

## Session Control crash Commands

The session control crash commands help control the execution of a crash session. Available commands are:

- alias – Creates an alias for a command string
- exit – Exits the crash session
- extend – Extends the crash command set by dynamically loading crash extension shared object libraries
- foreach – Allows you to execute a crash command on multiple tasks in the system
- gdb – Passes arguments to the GNU Debugger
- repeat – Repeats a command indefinitely
- set – Changes the crash context to a new task, or displays the current context
- q – Exits the crash session



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following session control crash commands control the crash session itself:

- alias – Creates an alias for a command string. Several built-in aliases are provided. Enter the alias command with no arguments to display the current list of aliases.
- exit – Exits the crash session. This command is the same as the q command.
- extend – Extends the crash command set by dynamically loading crash extension shared object libraries. Use the -u option to unload shared object libraries.
- foreach – Allows you to execute a crash command on multiple tasks in the system. It can be used with bt, vm, task, files, net, set, sig, and vtop commands.
- gdb – Passes arguments to the GNU Debugger for processing. Use the gdb help command to list classes of commands.
- repeat – Repeats a command indefinitely until stopped with Ctrl + C. This command is useful only on a live system.
- set – Changes the crash context to a new task, or is used to display the current context
- q – Exits the crash session. This command is the same as the exit command.

## General Guidelines for Using `crash`

- The `bt` command is often the first command you use after starting a crash session.
- To show the stack traces of an active task on each CPU:
 

```
crash> bt -a
```
- To see source file line numbers:
 

```
crash> bt -l
```
- To check for D state stuck processes:
 

```
crash> ps | grep UN
```
- To change the context to a different PID:
 

```
crash> set <PID>
```
- Other useful commands include `files`, `mount`, and `net`.
- Use `help <command>` to see options and examples.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The steps to debug a kernel crash vary according to the problem. The following provides some basic steps to follow. The `bt` command is often the first command you use after starting a crash session. The `bt` command shows the function trace leading up to the kernel panic. Use the `bt -a` command to show the stack traces of an active task on each CPU because there is often a relationship between the panicking tasks on one CPU and the running tasks on the other CPUs. If you see `cpu_idle` and `swapper`, it means nothing is running on that CPU. You can also use `bt` as an argument to the `foreach` command to display backtraces of all tasks. Use the `bt -l` command to see source file line numbers.

The `kmem -i` command provides a good summary of memory and swap use. Look for SLAB greater than 500 MB and SWAP in use. Use the command `ps | grep UN` to check for D state stuck processes. These processes contribute to the load average.

You can also redirect output to a file as follows. Look through the output for a process that is hung to set the PID. Use the `set` command to change the context to that PID:

```
crash> ps > ps.txt
crash> set <PID>
```

Commands that show open files (`files`), mount points (`mount`), and network devices with IP addresses (`net`) are often helpful. Use the `help <command>` in `crash` to see options.

## Quiz



Which of the following statements are true?

- a. To use kdump, first install the kdump-tools package.
- b. To enable kdump, append the crashkernel=<mem\_size> parameter to the kernel line in grub.cfg.
- c. The configuration file for kdump is /etc/kdump.conf.
- d. You cannot write a vmcore dump to a remote system.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. To use the `crash` utility, you must install the `crash` RPM and the `kernel-debuginfo` RPM packages.
- b. The `kernel-debuginfo` RPM packages for Oracle Linux 7 are available from <https://oss.oracle.com/ol7/debuginfo>.
- c. The `crash` utility can analyze core dumps created by the `kdump`, `netdump`, `diskdump`, `xm dump-core`, Linux Kernel Crash Dumps (LKCD), and `vissh` dump facilities.
- d. The `crash` utility has been merged with `gdb` (the GNU Debugger), so `crash` includes source code-level debugging capabilities.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. The `crash` utility provides commands to display kernel data structures.
- b. The `crash` utility provides commands to display various kernel subsystems on a systemwide or per-task basis.
- c. The `crash` utility provides useful helper commands such as a calculator, translator, and search function.
- d. The most useful `crash` command is `bt` (backtrace) because it shows the function trace leading up to the kernel panic.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Configure kdump to capture kernel vmcore dump
- Set kernel parameters that can cause a panic
- Use magic SysRq keys
- Use the `crash` utility for analyzing core dumps



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 18: Overview

This practice covers the following topics:

- Configuring kdump
- Creating a core dump file
- Preparing your system to analyze the vmcore
- Using the crash utility to analyze the vmcore



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Dynamic Tracing with DTrace

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the purpose of DTrace
- Enable DTrace on Oracle Linux
- Describe and view DTrace providers and probes
- Use the D programming language to enable probes and corresponding actions
- Use built-in D variables
- Use built-in D functions
- Create D scripts to explore your system



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## DTrace: Introduction

- DTrace is a performance analysis and troubleshooting tool.
- It has been available in the Oracle Solaris operating system since 2005, and was ported to Oracle Linux.
- DTrace allows dynamic tracing by defining probe points dynamically.
- Probes and actions at probe points are defined by commands and scripts written in the D language.
  - Probes are made available through providers.
- The current list of DTrace Oracle Linux providers is as follows:
  - dtrace, profile, syscall, proc, sched, io, sdt, and fasttrap



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DTrace is a comprehensive dynamic tracing facility originally developed for Oracle Solaris, and is now available with Oracle Linux. DTrace is a performance analysis and troubleshooting tool designed to give operational insights that allow you to tune and troubleshoot the OS and user-space programs. Use DTrace to explore your system and to understand how it works, to track down performance problems across many layers of software, or to locate the cause of a multitude of abnormal behavior problems. DTrace also provides Oracle Linux developers with a tool to analyze performance, and to better see how their systems work. DTrace enables higher quality application development, reduced down time, lower cost, and greater utilization of existing resources.

DTrace stands for Dynamic Tracing and provides an instrumentation technique that dynamically patches live running instructions with instrumentation code. At specific points of execution in the code, you can activate probes and designate actions, such as collecting and displaying information. DTrace allows you to dynamically define probe points. This means that they are not precompiled into the kernel. DTrace has providers, which are basically categories of probes. The providers of DTrace for Linux are listed in the slide.

DTrace provides a D programming language to enable probe points and associated actions. Use the D language from the command line, or create D script files when performing complex tracing. The D language is similar to the C programming language and awk.

## Reasons to Use DTrace on Linux

- Many tracing and profiling tools exist for Linux, for varying use cases:
  - Inconsistent syntax, scripting languages, and output formats
  - Lack of integration of both kernel and application tracing in a single tool
- DTrace provides an integrated solution.
- Administrators and developers know DTrace from Oracle Solaris.
- Existing D scripts can be used.
- Customers ask for it.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DTrace provides an integrated solution to the variety of tracing tools currently available for Linux. Available tracing and profiling tools include:

- `strace` – This tool traces system calls and signals made by a program.
- `pstack` – This tool prints a stack trace of a running process.
- `oprofile` – This is a system profiling tool that is capable of profiling all running code, including hardware and software interrupt handlers, kernel modules, the kernel, shared libraries, and applications. It uses hardware counters and can also count cache activity.
- `perf` – This tool is part of the Performance Counters for Linux (PCL) kernel-based subsystem. It tracks hardware events and can also measure software events.
- `stap` – This tool is the front end to Systemtap, which can collect information about a running Linux system for use in the diagnosis of performance or functional problems.
- `valgrind` – This is a suite of tools for debugging and profiling Linux executables. The tool suite includes memory leak checking, thread error detectors, and other profiler tools.
- `blktrace` – This tool is a block layer I/O tracing mechanism, which traces I/O traffic on block devices.

Each of these tools uses a different syntax, a different scripting language, and provides different output formats. There is a lack of integration of both kernel and application tracing. DTrace provides the integrated tracing solution in a single tool that customers have asked for.

Another reason for porting DTrace to Linux is that administrators and developers know DTrace from Oracle Solaris. These skills are transferable to DTrace for Oracle Linux. Existing Oracle Solaris D scripts can be used. A DTrace book titled *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD* (by Brendan Gregg and Jim Mauro, Prentice Hall 2011) contains hundreds of D scripts. There is also the DTraceToolkit, which is a collection of over 200 useful documented scripts. Some of the many tools developed using DTrace and the D programming language include the following:

- `dexplorer` – Automatically runs a collection of DTrace scripts to examine many areas of the system, and places the output in a meaningful directory structure that is tar'd and gzip'd
- `iosnoop` – Displays disk I/O activity in real time, allowing you to observe what is happening on your disks, including the PID and the responsible command. The output includes the block address and size of the disk operation.
- `iotop` – Displays top disk I/O events by process. This tracks disk I/O by process and prints a summary report that is refreshed every interval.
- `execsnoop` – Displays process activity. As processes are executed on the server, their details are printed out. This is especially useful in troubleshooting short-lived processes that are otherwise difficult to spot.
- `opensnoop` – Displays file opens. The file name and file handle are traced along with some process details.
- `rwttop` – Displays top read/write bytes by process. It prints a summary report that is refreshed at intervals. This measures reads and writes at the application level.
- `tcpsnoop` – Displays TCP network packets by process. This analyses TCP network packets and prints the responsible PID and UID, plus standard details such as IP address and port. This captures traffic of newly created TCP connections that were established while this program was running. It helps to identify processes causing TCP traffic.

The DTraceToolkit, the code for the previously mentioned D scripts, and many other D scripts are available at the following website: <http://www.brendangregg.com/dtrace.html>. Not all of the scripts in the DTrace book and in the DTrace Toolkit work in Linux. This is because DTrace for Linux continues to evolve and not all of the providers used in these scripts are available in DTrace for Linux. Another reason is where function or system calls are used (named), they probably differ between Solaris and Linux and the user must use the corresponding Linux kernel function name. However, many of the scripts do work in Linux and all the scripts are helpful in learning the capabilities of DTrace and the D programming language.

## DTrace 0.4 in UEK R3

- In UEK R3, DTrace support is integrated with the kernel.
- You also need to install the `dtrace-utils` and `dtrace-modules` packages that are available on ULN.
- Meta-provider support has been implemented:
  - Allows DTrace to instantiate providers dynamically on demand.
  - An example of a meta-provider is the `fasttrap` provider that is used for user-space tracing.
- User-space statically defined tracing (USDT) supports SDT-like probes in user-space executable and libraries.
- The SDT provider has been improved so that probes for kernel modules can now be enabled.
  - Previously you could enable probes only in the core kernel.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

As of this writing, the latest release is DTrace 0.4 in UEK R3. The following lists some of the additional features compared with DTrace 0.3.2 in UEK R2. Refer to the Unbreakable Enterprise Kernel Release 3 Release Notes at [http://docs.oracle.com/cd/E52668\\_01/](http://docs.oracle.com/cd/E52668_01/) for a complete list of additional features.

- In addition to using the UEK R3, to use DTrace 0.4 you must install the `dtrace-utils` and `dtrace-modules` packages that are available on the `ol7_x86_64_UEK3` and `ol7_x86_64_Dtrace_userspace` channels on the Unbreakable Linux Network (ULN).
- Meta-provider support has been implemented, which allows DTrace to instantiate providers dynamically on demand. An example of a meta-provider is the `fasttrap` provider that is used for user-space tracing.
- User-space statically defined tracing (USDT) supports Statically Defined Tracing (SDT)-like probes in user-space executable and libraries.
- USDT requires programs to be modified to include embedded static probe points. The `sys/sdt.h` header file is provided to support USDT, but you can also use the `-h` option to `dtrace` to generate a suitable header file from a provider description file.
- The SDT provider has been improved so that probes for kernel modules can now be enabled. Previously you could enable probes only in the core kernel.

The list of the additional features in DTrace 0.4 in UEK R3 compared with DTrace 0.3.2 in UEK R2 continues:

- To enable the use of USDT probes in DTrace-enabled programs, you must load the new `fasttrap` module:  
`# modprobe fasttrap`
- Currently, the `fasttrap` provider supports the use of USDT probes. It is not used to implement the `pid` provider.
- For more information, refer to the “Statically Defined Tracing of User Applications” chapter of the *Oracle Linux DTrace Guide*, at  
[http://docs.oracle.com/cd/E52668\\_01/E38608/html/index.html](http://docs.oracle.com/cd/E52668_01/E38608/html/index.html).

# DTrace-Enabled Applications

- A number of DTrace-enabled applications are available with the release of DTrace 0.4, including MySQL and PHP.
  - These applications have been instrumented to contain statically defined DTrace probes.
- DTrace-enabled versions of user-space applications are available in the playground repository of Oracle Public Yum.
  - Oracle does not offer support for these packages and does not accept any liability for their use.
- PostgreSQL 9.2.4 includes support for DTrace.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A number of DTrace-enabled applications are available with the release of DTrace 0.4, including MySQL and PHP. These applications have been instrumented to contain statically defined DTrace probes. You can find details about the probes for MySQL at: <http://dev.mysql.com/doc/refman/5.5/en/dba-dtrace-mysqld-ref.html>.

Details about the probes for PHP are available at <http://php.net/manual/features.dtrace.php>.

DTrace-enabled versions of user-space applications are available in the playground repository of Oracle Public Yum. There is no playground repository for Oracle Linux 7. See [http://public-yum.oracle.com/repo/OracleLinux/OL6/playground/latest/x86\\_64/](http://public-yum.oracle.com/repo/OracleLinux/OL6/playground/latest/x86_64/).

The packages that are provided in the playground repository are intended for experimentation only and you should not use them with production systems. Oracle does not offer support for these packages and does not accept any liability for their use.

PHP 5.4.20, PHP 5.5.4, and later versions can be built with DTrace support on Oracle Linux. See [https://blogs.oracle.com/opal/entry/using\\_php\\_dtrace\\_on\\_oracle](https://blogs.oracle.com/opal/entry/using_php_dtrace_on_oracle).

PostgreSQL 9.2.4 includes support for DTrace as described in <http://www.postgresql.org/docs/9.2/static/dynamic-trace.html>.

You can build a DTrace-enabled version of PostgreSQL by specifying the --enable-dtrace option to configure as described in <http://www.postgresql.org/docs/9.2/static/install-procedure.html>.

# ULN Channels for DTrace 0.4

| Name                                                                         | Label                        | Description                                                                                               | Packages |
|------------------------------------------------------------------------------|------------------------------|-----------------------------------------------------------------------------------------------------------|----------|
| Oracle Linux 7 Latest Optional Packages (x86_64)                             | ol7_x86_64_optional_latest   | All optional packages released for Oracle Linux 7 (x86_64) including the latest errata packages. (x86_64) | 4202     |
| Oracle Linux 7 Latest (x86_64)                                               | ol7_x86_64_latest            | All packages released for Oracle Linux 7 (x86_64) including the latest errata packages. (x86_64)          | 4393     |
| Oracle Linux 7 Update 1 installation media copy (x86_64)                     | ol7_x86_64_u1_base           | All packages released for Oracle Linux 7 Update 1 (x86_64). No errata included                            | 4384     |
| Oracle Linux 7 Update 1 Patch (x86_64)                                       | ol7_x86_64_u1_patch          | Updated packages published after release of Oracle Linux 7 Update 1 (x86_64)                              | 371      |
| Oracle Linux 7 GA installation media copy (x86_64)                           | ol7_x86_64_u0_base           | All packages released for Oracle Linux 7 GA (x86_64). No errata included                                  | 4315     |
| Oracle Linux 7 GA Patch (x86_64)                                             | ol7_x86_64_u0_patch          | Updated packages published after release of Oracle Linux 7 GA (x86_64)                                    | 583      |
| MySQL 5.6 for Oracle Linux 7 (x86_64)                                        | ol7_x86_64_MySQL56_community | Latest MySQL 5.6 packages for Oracle Linux 7 (x86_64)                                                     | 15       |
| MySQL 5.5 for Oracle Linux 7 (x86_64)                                        | ol7_x86_64_MySQL55_community | Latest MySQL 5.5 packages for Oracle Linux 7 (x86_64)                                                     | 15       |
| Oracle Linux 7 Dtrace Userspace Tools (x86_64) - Latest                      | ol7_x86_64_Dtrace_userspace  | The latest Dtrace userspace tools for Oracle Linux 7 (x86_64)                                             | 2        |
| Oracle Linux 7 Addons (x86_64)                                               | ol7_x86_64_addons            | Oracle Linux 7 Addons (x86_64)                                                                            | 59       |
| Unbreakable Enterprise Kernel Release 3 for Oracle Linux 7 (x86_64) - Latest | ol7_x86_64_UEKR3             | Latest packages for Unbreakable Enterprise Kernel Release 3 for Oracle Linux 7 (x86_64)                   | 25       |
| OFED supporting tool packages for                                            |                              | Latest OpenFabrics Enterprise Distribution (OFED) supporting                                              |          |

To install and configure DTrace 0.4, subscribe your system to the following ULN channels:

- Oracle Linux 7 Latest (x86\_64) (ol7\_x86\_64\_latest)
- Oracle Linux 7 Dtrace Userspace Tools (x86\_64) – Latest (ol7\_x86\_64\_Dtrace\_userspace)
- Unbreakable Enterprise Kernel Release 3 for Oracle Linux 7 (x86\_64) – Latest (ol7\_x86\_64\_UEKR3)

# Enabling DTrace

- Register your systems with the appropriate ULN channels.
- To update your system to UEK R3:

```
# yum update
```

- To install the DTrace utilities package:

```
# yum install dtrace-utils
```

- The DTrace kernel modules:
  - Are installed in `/lib/modules/`uname -r`/kernel/drivers/dtrace`
  - Use the `modprobe` command to load the kernel modules.
- Release notes are located in `/usr/share/doc/dtrace*`.
- DTrace guides and UEK R3 release notes are available at [http://docs.oracle.com/cd/E52668\\_01/](http://docs.oracle.com/cd/E52668_01/).



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After registering your systems with the appropriate channels on ULN, use the `yum` command to install the packages.

If your system is not already running the Unbreakable Enterprise Kernel Release 3 (UEK R3), use the `yum` command as follows to update your system to use UEK R3:

```
# yum update
```

After the `yum update` command completes, reboot your system. Your system should automatically boot the UEK R3 (version 3.8.13). Otherwise, select the Oracle Linux Server (3.8.13) kernel in the GRUB menu.

Use the `yum` command as follows to install the DTrace utilities package:

```
# yum install dtrace-utils
```

If the appropriate `dtrace-modules` package for the running kernel is not present on your system, running any `dtrace` command downloads and installs the package from ULN, for example:

```
# dtrace -l
```

Alternatively, you can use the `yum` command to install the `dtrace-modules-`uname -r`` package.

The DTrace modules are installed in `/lib/modules/<UEK_version>.x86_64/kernel/drivers/dtrace`.

```
# ls -l /lib/modules/`uname -r`/kernel/drivers/dtrace
-rw-r--r-- ... dt_perf.ko
-rw-r--r-- ... dtrace.ko
-rw-r--r-- ... dt_test.ko
-rw-r--r-- ... fasttrap.ko
-rw-r--r-- ... profile.ko
-rw-r--r-- ... sdt.ko
-rw-r--r-- ... systrace.ko
```

The developer's test suit uses the `dt_test` module. Note that this module is not for general use. This module is included in the distribution because it would be required if the test suite is made available as a package in the future.

The `dt_perf` module is for internal testing only and is provided as a “curiosity” item. It is of limited use for anyone other than developers of DTrace.

Use the `modprobe` command to load the modules. You don't have to load the `dtrace` module manually because loading any other module loads the `dtrace` module.

```
# modprobe fasttrap
# modprobe profile
# modprobe sdt
# modprobe systrace
# lsmod | grep dtrace
dtrace      135025  4 sdt,fasttrap,systrace,profile
ctf          941    1 dtrace
```

Additional resources include the following guides located at:

[http://docs.oracle.com/cd/E52668\\_01/](http://docs.oracle.com/cd/E52668_01/):

- *Oracle Linux DTrace Guide*
- *Oracle Linux DTrace Tutorial*

## DTrace Probes

- Instrumentation points are called *probes*.
- Each probe is associated with an action.
- Actions determine what data to record when a probe fires.
  - Any argument to the function
  - Any global variable in the kernel
  - A nanosecond time stamp of when the function was called
  - A stack trace to indicate what code called this function
- A probe:
  - Is made available by a provider
  - Identifies the instrumented function (for function-bound probes) and, in most cases, the containing module
  - Has a name
  - Has a unique integer identifier



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DTrace probes are tracing points, or instrumentation points, that enable you to record data at various points of interest. Each probe is associated with an action. When the probe fires, the action is executed. For example, you can define probes that fire upon entry into a kernel function, when a specific file opens, when a particular process starts, or when a line of code executes. Examples of data that you can display include:

- Any argument to the function
- Any global variable in the kernel
- A nanosecond time stamp of when the function was called
- A stack trace to indicate what code called this function

A probe:

- Is made available by a provider
- (For function-bound probes) identifies the instrumented function and, in most cases, the containing module
- Has a name
- Has a unique integer identifier

## DTrace Providers

- A probe is defined and implemented by a provider.
- A provider is a kernel module that enables its probes to trace data.
- The following providers are available:
  - dtrace, fasttrap, io, proc, profile, sched, sdt, and syscall
- To list all probes for all providers:

```
# dtrace -l
ID PROVIDER      MODULE      FUNCTION NAME
1   dtrace
...                                BEGIN
```

- Each line of output identifies a specific probe in the following form:

*provider:module:function:name*



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DTrace probes are implemented by providers. A provider is a kernel module that enables a requested probe to fire when it is hit. A provider receives information from DTrace about when a probe is to be enabled and transfers control to DTrace when an enabled probe is hit.

Use the `dtrace -l` command to list all probes and their respective providers:

```
# dtrace -l
ID PROVIDER      MODULE      FUNCTION NAME
1   dtrace
2   dtrace
3   dtrace
4   syscall      vmlinux      read entry
5   syscall      vmlinux      read return
...
...
```

Each line of output identifies a specific probe. Use the following syntax to uniquely identify each probe:

*provider:module:function:name*

The DTrace providers are summarized as follows:

- `dtrace` – Provides probes that relate to DTrace itself, such as BEGIN, ERROR, and END. You can use these probes to initialize DTrace's state before tracing begins, process its state after tracing has completed, and handle unexpected execution errors in other probes.
- `profile` – Provides probes associated with an interrupt that fires at a fixed, specified time interval. These probes are associated with the asynchronous interrupt event rather than with any particular point of execution. You can use these probes to sample some aspect of a system's state.
- `syscall` – Provides probes at the entry to and return from every system call. Because system calls are the primary interface between user-level applications and the operating system kernel, these probes can offer you an insight into the interaction between applications and the system.
- `sdt` – Creates probes at sites that a software programmer has formally designated. The Statically Defined Tracing (SDT) mechanism allows programmers to choose locations of interest to users of DTrace and to convey information about each location through the probe name.
- `proc` – Provides probes for monitoring process creation and termination, LWP (light-weight process) creation and termination, executing new program images, and sending and handling signals
- `sched` – Provides probes related to CPU scheduling. Because CPUs are the one resource that all threads must consume, the `sched` provider is very useful for understanding systemic behavior.
- `io` – Provides probes that relate to data input and output. The `io` provider enables quick exploration of behavior observed through I/O monitoring tools such as `iostat`.
- `fasttrap` – Supports user-space tracing of DTrace-enabled applications. The `fasttrap` provider makes available a single probe that fires each time that a DTrace-enabled user process executes an instruction.

## dtrace Provider

- The `dtrace` provider enables preprocessing, postprocessing, and D program error-processing capabilities.
- To list all probes for the `dtrace` provider:

```
# dtrace -l -P dtrace
ID PROVIDER      MODULE      FUNCTION NAME
1  dtrace          BEGIN
2  dtrace          END
3  dtrace          ERROR
```

- Three probes are provided by the `dtrace` provider:
  - `dtrace:::BEGIN` – Fires each time you start a new tracing request
  - `dtrace:::END` – Fires after all other probes
  - `dtrace:::ERROR` – Fires when a runtime error occurs in executing a clause for a probe



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use `dtrace` probes to initialize state (preprocessing) before tracing begins, process state after tracing has completed (postprocessing), and handle unexpected execution errors in other probes.

Only three probes are provided by the `dtrace` provider:

- `BEGIN` – Fires each time you start a new tracing request. Use this probe to initialize any state that is needed in other probes.
- `END` – Fires after all other probes. Use this probe to process state that has been gathered or to format the output.
- `ERROR` – Fires when a runtime error occurs in executing a clause for a probe

The syntax to uniquely identify each probe is as follows:

*provider:module:function:name*

The `dtrace` probes are not bound to a function or module; therefore, these probes are identified as follows:

- `dtrace:::BEGIN`
- `dtrace:::END`
- `dtrace:::ERROR`

## profile Provider

- The `profile` provider provides dynamic probes that fire at specific time intervals.
- You can use `profile` probes to sample an aspect of system state and provide periodic output or take periodic action.
- To list all probes for the `profile` provider:

```
# dtrace -l -P profile
ID PROVIDER    MODULE          FUNCTION NAME
635  profile
636  profile
...
642  profile
643  profile
644  profile
645  profile
646  profile
...

```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `profile` provider provides probes associated with a time-based interrupt that fires at a fixed, specified time interval. These probes are not associated with any particular point of execution. Use these probes to sample some aspect of system state such as the state of the current thread, the state of the CPU, or the current machine instruction.

The `profile-N` probes fire at a fixed interval at a high interrupt level on all CPUs.

The `tick-N` probes fire at fixed intervals at a high interrupt level on only one CPU per interval. The actual CPU might change over time.

The value of `N` defaults to rate-per-second. You can include an optional time suffix as follows:

- `nsec|ns` – Nanoseconds; for example, `tick-1ns`
- `usec|us` – Microseconds; for example, `tick-10us`
- `msec|ms` – Milliseconds; for example, `tick-100ms`
- `sec|s` – Seconds (this is the default)
- `min|m` – Minutes; for example, `tick-1m`
- `hour|h` – Hours; for example, `tick-1h`
- `day|d` – Days; for example, `tick-1d`
- `hz` – Hertz; the highest supported tick frequency is 5000 HZ (`tick-5000hz`).

## syscall Provider

- The `syscall` provider dynamically instruments the entry and return of every system call.
- A majority of the probes are provided by `syscall`.
- To list all probes for the `syscall` provider:

```
# dtrace -l -P syscall
ID PROVIDER      MODULE      FUNCTION NAME
4  syscall        vmlinux    read entry
5  syscall        vmlinux    read return
6  syscall        vmlinux    write entry
7  syscall        vmlinux   write return
8  syscall        vmlinux   open entry
9  syscall        vmlinux   open return
10 syscall        vmlinux  close entry
11 syscall        vmlinux  close return
...
...
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `syscall` provider makes available a probe at the entry to and return from every system call in the system. A system call changes the processor state from user mode to kernel mode, so that the CPU can access protected kernel memory. Because system calls are the primary interface between user-level applications and the operating system kernel, the `syscall` provider offers tremendous insight into application behavior with respect to the system.

A majority of the probes are provided by the `syscall` provider. The `syscall` provider provides a pair of probes for each system call:

- `entry` – Fires before the system call is entered
- `return` – Fires after the system call has completed but before control has transferred back to user level

For all `syscall` probes, the function name is set to the name of the instrumented system call and the module name is `vmlinux`. As an example, the name for `syscall` probe ID 4 is:

`syscall:vmlinux:read:entry`

## sdt Provider

- The `sdt` (Statically Defined Tracing) provider allows programmers to insert explicit probes in their applications.
- SDT probes are declared using the following macros from `<sys/sdt.h>`:
  - `DTRACE_PROBE`, `DTRACE_PROBE1`, `DTRACE_PROBE2`, `DTRACE_PROBE3`, and `DTRACE_PROBE4`
- The module name and function name correspond to the kernel module and function of the probe.
- The name of the probe depends on the name given in the `DTRACE_PROBEn` macro.
- To list the SDT probes you have installed:

```
# dtrace -l -P sdt -m <module>
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `sdt` (Statically Defined Tracing) provider allows programmers to insert explicit probes in their applications. Programmers can select locations of interest to users of DTrace and provide some information about each location through the probe name. Because the SDT probes that are defined for Oracle Linux kernel are likely to change over time, they are not listed here.

SDT probes are declared using the following macros from `<sys/sdt.h>`: `DTRACE_PROBE`, `DTRACE_PROBE1`, `DTRACE_PROBE2`, `DTRACE_PROBE3`, and `DTRACE_PROBE4`. The module name and function name of an SDT-based probe correspond to the kernel module and function of the probe. The name of the probe depends on the name given in the `DTRACE_PROBEn` macro.

DTrace includes the kernel module name and function name as part of the tuple identifying a probe; therefore, you do not need to include this information in the probe name. Use the following command on your driver module to list the probes you have installed and the full names that are seen by DTrace users:

```
# dtrace -l -P sdt -m <module>
```

## proc Provider

- The proc provider makes available all probes pertaining to the following activities:
  - Process creation and termination
  - LWP creation and termination
  - Executing new program images
  - Sending and handling signals
- To list all probes for the proc provider:

```
# dtrace -l -P proc
ID PROVIDER    MODULE           FUNCTION NAME
608  proc      vmlinux         __send_signal signal-discard
609  proc      vmlinux         __send_signal signal-send
...
626  proc      vmlinux         get_signal_to_deliver signal-handle
628  proc      vmlinux         schedule_tail lwp-start
629  proc      vmlinux         schedule_tail start
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The proc provider makes available probes pertaining to the following activities: process creation and termination, lightweight process (LWP) creation and termination, executing new program images, and sending and handling signals.

There are currently 13 proc probes, including the following:

- `start` – Fires in the context of a newly created process, before any user-level instructions are executed in the process
- `create` – Fires when a process (or process thread) is created using `fork()` or `vfork()`
- `exec` – Fires whenever a process loads a new process image using a variant of the `execve()` system call
- `exec-failure` – Fires when an `exec()` variant has failed
- `exec-success` – Fires when an `exec()` variant has succeeded
- `exit` – Fires when the current process is exiting
- `lwp-create` – Fires when a process thread is created
- `lwp-start` – Fires within the context of a newly created process or process thread
- `lwp-exit` – Fires when a process or process thread is exiting

## sched Provider

- The `sched` provider makes available all probes related to CPU scheduling.
- Probes provided by `sched` allow you to trace key scheduling events.
- To list all probes for the `sched` provider:

```
# dtrace -l -P sched
ID PROVIDER MODULE FUNCTION NAME
 604 sched vmlinux __schedule remain-cpu
 605 sched vmlinux __schedule sleep
 606 sched vmlinux __schedule preempt
 607 sched vmlinux __schedule off-cpu
 612 sched vmlinux dequeue_task dequeue
 623 sched vmlinux enqueue_task enqueue
 624 sched vmlinux finish_task_switch on-cpu
...
...
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `sched` provider makes available probes related to CPU scheduling. The `sched` provider dynamically traces key scheduling events. Because CPUs are the one resource that all threads must consume, the `sched` provider is very useful for understanding systemic behavior. For example, using the `sched` provider, you can understand when and why threads sleep, run, change priority, or wake other threads.

There are currently 12 `sched` probes, including the following:

- `on-cpu` – Fires when a CPU begins to execute a thread
- `off-cpu` – Fires when a thread is about to be taken off a CPU
- `surrender` – Fires when a CPU has been instructed by another CPU to make a scheduling decision, often because a higher-priority thread has become runnable
- `change-pri` – Fires whenever a thread's priority is about to be changed
- `enqueue` – Fires immediately before a runnable thread is enqueued to a run queue
- `dequeue` – Fires immediately before a runnable thread is dequeued from a run queue
- `tick` – Fires as a part of clock tick-based accounting
- `wakeup` – Fires immediately before the current thread wakes a thread sleeping on a synchronization object

## io Provider

- The `io` provider makes available all probes related to device input and output (I/O).
- To list all probes for the `io` provider:

```
# dtrace -l -P io
ID PROVIDER      MODULE          FUNCTION NAME
610    io        vmlinux       __wait_on_buffer wait-done
611    io        vmlinux       __wait_on_buffer wait-start
622    io        vmlinux      end_bio_bh_io_sync done
630    io        vmlinux      submit_bh start
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `io` provider makes available probes related to device input and output (I/O). You can explore behavior observed through I/O monitoring tools such as `iostat`.

The `io` probes are defined as follows:

- `start` – Fires when an I/O request is about to be made either to a peripheral device or to an NFS server
- `done` – Fires after an I/O request has been fulfilled
- `wait-start` – Fires immediately before a thread begins to wait, pending completion of a given I/O request
- `wait-done` – Fires when a thread finishes waiting for the completion of a given I/O request

## Enabling Probes

- Use the `dtrace` command without the `-l` option.
- You can specify probes to enable by provider (`-P`), by name (`-n`), by function (`-f`), and by module (`-m`).
- To enable all probes provided by the `syscall` provider:

```
# dtrace -P syscall
```

- To enable the `BEGIN` probe in the `dtrace` provider:

```
# dtrace -n dtrace:::BEGIN
```

- To enable probes in the `syscall` provider whose function name begins with “open”:

```
# dtrace -n syscall:::open*
```

- To enable every probe in the `vmlinux` module:

```
# dtrace -m vmlinux
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Probes are enabled with the `dtrace` command by specifying them without the list (`-l`) option. DTrace performs the associated action when the probe fires. The default action indicates only that a probe fired. No other data is recorded. You can enable (and list) probes by provider (`-P`), by name (`-n`), by function (`-f`), and by module (`-m`).

To enable all probes provided by the `syscall` provider:

```
# dtrace -P syscall
dtrace: description 'syscall' matched 592 probes
CPU      ID          FUNCTION:NAME
  0       36          ioctl:entry
  0       37          ioctl:return
  0       36          ioctl:entry
  0       37          ioctl:return
  0       30          rt_sigaction:entry
^C
```

From the output, you can see that the default action displays the CPU where the probe fired, the unique probe ID, the function where the probe fired, and the probe name.

The following example enables the BEGIN probe in the `dtrace` provider:

```
# dtrace -n dtrace:::BEGIN
dtrace: description 'dtrace:::BEGIN' matched 1 probe
CPU          ID          FUNCTION:NAME
0            1            :BEGIN
^C
```

The following example enables probes provided by the `syscall` provider in the `open` function:

```
# dtrace -n syscall:::open:
dtrace: description 'syscall:::open:' matched 2 probes
CPU          ID          FUNCTION:NAME
0            8            open:entry
0            9            open:return
0            8            open:entry
0            9            open:return
0            8            open:entry
^C
```

You can use the special characters \*, ?, and [ ] as wildcards in probe names. The preceding example matched two probes. The following example uses the \* character to list all probes in the `syscall` provider whose function name begins with "open":

```
# dtrace -l -n syscall:::open*:
ID PROVIDER      MODULE           FUNCTION NAME
  8 syscall        open entry
  9 syscall        open return
492 syscall       openat entry
493 syscall       openat return
586 syscall       open_by_handle_at entry
586 syscall       open_by_handle_at return
^C
```

The following example enables all probes in the `vmlinux` module, regardless of the provider to which they belong:

```
# dtrace -m vmlinux
dtrace: description 'vmlinux' matched 35 probes
CPU          ID          FUNCTION:NAME
0            628         __schedule:sleep
0            620         dequeue_task:dequeue
0            626         __schedule:off-cpu
0            609         finish_task_switch:on-cpu
^C
```

## DTrace Actions

- DTrace actions are taken when a probe fires.
- The default action indicates only that a probe fired.
  - No other data is recorded.
- Actions are programmable in the D language.
  - Enclose actions in braces { }.
  - Enclose probe and associated action in single quotes ` '.
- The following example uses the `trace()` function as the action:

```
# dtrace -n 'syscall:::entry {trace(timestamp)}'
dtrace: description 'syscall:::entry' matched 296 probes
CPU    ID      FUNCTION:NAME
0        6      write:entry 173635407378228
^C
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DTrace actions are taken when a probe fires. The default action displays the CPU where the probe fired, the unique probe ID, the function where the probe fired, and the probe name.

Use the D programming language to specify probes of interest and bind actions to those probes. The D programming language is similar to the C programming language and includes a set of functions and variables to help make tracing easier. The actions are listed as a series of statements enclosed in braces { }, following the probe name.

The following example uses the `trace()` function as the action to trace the time of entry to each system call:

```
# dtrace -n 'syscall:::entry {trace(timestamp)}'
dtrace: description 'syscall:::entry' matched 296 probes
CPU    ID      FUNCTION:NAME
0        6      write:entry 173635407378228
0        6      write:entry 173635407399180
0       36      ioctl:entry 173635407416780
^C
```

Surround the probe name and associated action with single quotation marks (` ').

In the following example, the probe of interest is the BEGIN probe provided by the `dtrace` provider. The actions are to call two functions, `trace()` and `exit()`. When specifying multiple actions, terminate each action with a semicolon (`;`).

```
# dtrace -n 'dtrace:::BEGIN {trace("hello, world");exit(0);}'  
dtrace: description 'dtrace:::BEGIN' matched 1 probe  
CPU          ID            FUNCTION:NAME  
0             1            :BEGIN  hello, world
```

The function `trace()` indicates that DTrace records the specified argument, the string "hello, world", when the `dtrace:::BEGIN` probe fires, and then prints it out.

The function `exit()` indicates that DTrace ceases tracing and exits the `dtrace` command.

## Built-in D Variables

- `errno` – Current errno value for system calls
- `execname` – Name of the current process' executable file
- `pid` – Process ID of the current process
- `tid` – Thread ID of the current thread
- `timestamp` – Time since boot in nanoseconds
- `probeprov` – Provider field of the current probe
- `probemod` – Module field of the current probe
- `probefunc` – Function field of the current probe
- `probename` – Name field of the current probe
- `curpsinfo` – Process state information for the current thread
- `curthread` – Internal kernel thread structure for the current thread



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The list in the slide gives some of the most useful built-in D variables. Refer to the *Oracle Linux DTrace Guide* for a complete list of built-in D variables.

## trace() Built-in Function

- `trace(pid)` – Traces the current process ID
- `trace(execname)` – Traces the name of the current executable
- `trace(curthread->t_pri)` – Traces the `t_pri` field of the current thread
- `trace(probefunc)` – Traces the function name of the probe
- `trace(timestamp / 1000)` – Traces the time stamp variable divided by 1000
- `trace("hello, world")` – Traces the string "hello, world"
- `trace(`max_pfn)` – Traces the `max_pfn` system variable



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are many built-in functions that perform actions. Previous examples use the `trace()` function. This function takes a D expression as its argument and traces the result to the directed buffer. All the examples in the slide are valid `trace()` actions.

The last example, `trace(`max_pfn)`, is an example of tracing the value of a system variable named `max_pfn`. DTrace instrumentation executes inside the Oracle Linux operating system kernel; therefore, in addition to accessing special DTrace variables and probe arguments, you can also access kernel data structures, symbols, and types. These capabilities enable advanced DTrace users, administrators, service personnel, and driver developers to examine the low-level behavior of the operating system kernel and device drivers.

D uses the backquote character (`) as a special scoping operator for accessing symbols that are defined in the operating system and not in your D program. For example, the Oracle Linux kernel contains a C declaration of a system variable named `max_pfn`. This variable is declared in C in the kernel source code as follows:

```
unsigned long max_pfn
```

Use the following statement in a D program script to trace the value of this variable:

```
trace(`max_pfn)
```

## DTrace: Examples

- Display commands executing on the system:

```
# dtrace -n 'proc:::exec {trace(stringof(arg0));}'
```

- Display new processes with arguments:

```
# dtrace -n 'proc:::exec-success {trace(curpsinfo->pr_psargs);}'
```

- Display files opened by processes using printf():

```
# dtrace -n 'syscall::open*:entry {printf("%s %s",
    execname,copyinstr(arg0));}'
```

- Display the number of system calls using aggregations:

```
# dtrace -n 'syscall:::entry {@num[probefunc] =
    count();}'
```

- With aggregation, totals appear after you press Ctrl + C.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Several examples are presented in subsequent slides. The following example displays commands that are executing on your system:

```
# dtrace -n 'proc:::exec {trace(stringof(arg0));}'
dtrace: description 'proc:::exec ' matched 1 probe
CPU      ID          FUNCTION:NAME
 0       617        do_execve_common:exec    /bin/ls
 0       617        do_execve_common:exec    /bin/bash
^C
```

The following example displays new processes with arguments:

```
# dtrace -n 'proc:::exec-success {trace(curpsinfo->pr_psargs);}'
dtrace: description 'proc:::exec-success ' matched 1 probe
CPU      ID          FUNCTION:NAME
 0       619  do_execve_common:exec-success /usr/sbin/sshd -R
 0       619  do_execve_common:exec-success /sbin/unix_chpwd root...
^C
```

The following example displays files opened by process. The example also introduces the `printf()` function, which is used to trace data, as well as to output the data and other text in a specific format that you describe. The `printf()` function tells DTrace to trace the data associated with each argument after the first argument ("`%s %s`"), known as the *format string*, and then to format the results using the rules described by the format string:

```
# dtrace -n 'syscall::open*:entry {printf("%s
%s", execname,copyinstr(arg0));}'
dtrace: description 'syscall::open*:entry' matched 3 probes
CPU ID FUNCTION:NAME
 0   8 open:entry vi /etc/ld.so.cache
 0   8 open:entry vi /lib64/libm.so.6
 0   8 open:entry crond /etc/passwd
 0   8 open:entry automount /proc/mounts
 0   8 open:entry man /etc/ld.so.cache
 0   8 open:entry man /lib64/libc.so.6
^C
```

The following example displays the number of system calls by system call using aggregations. DTrace provides several built-in functions for aggregating data that individual probes gather. The name of the aggregation is prefixed with the @ symbol. Press Ctrl + C to display the totals:

```
# dtrace -n 'syscall:::entry {@num[probefunc] = count();}'
dtrace: description 'syscall:::entry' matched 296 probes
^C
getgid                                1
geteuid                               1
inotify_add_watch                      1
newlstat                             1
rt_sigreturn                           1
setgid                                1
setresgid                            1
setuid                                1
close                                 2
fcntl                                2
getdents                             2
geteuid                               2
newfstat                             2
open                                 2
setgroups                            2
...
...
```

## DTrace: Examples

- Display the number of `write()` system calls and the number of `read()` system calls invoked by processes:

```
# dtrace -n 'syscall::write:entry,syscall::read:entry
{@[strjoin(probefunc,"() calls")] = count();}'
```

- Display disk size by process:

```
# dtrace -n 'io:::start {printf("%d %s %d",
pid,execname,args[0]->b_bcount);}'
```

- Display the number of `read()` system calls, excluding those initiated by the `dtrace` process:

```
# dtrace -n 'syscall::read:entry /execname != "dtrace"/
{@reads[execname, fds[arg0].fi.pathname] = count();}'
```

- This example uses a predicate.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following example counts the number of `write()` system calls and the number of `read()` system calls invoked by processes until you press Ctrl + C:

```
# dtrace -n 'syscall::write:entry,syscall::read:entry
{@[strjoin(probefunc,"() calls")] = count();}'
dtrace: description 'syscall::write:entry,syscall::read:entry '
matched 2 probes
^C
write() calls                      238
read() calls                        431
```

The following example displays disk size by process:

```
# dtrace -n 'io:::start {printf("%d %s %d",pid,execname,args[0]->b_bcount);}'
CPU      ID          FUNCTION:NAME
  0    602          submit_bh:start 342 jdb2/dm-0-8 4096
  0    602          submit_bh:start 1184 flush-252:0 4096
^C
```

The following example uses a predicate to selectively aggregate the number of `read()` system calls. The `read()` system calls initiated by the `dtrace` process are excluded because of the `/execname != "dtrace"` / predicate:

```
# dtrace -n 'syscall::read:entry /execname != "dtrace"/ {
@reads[execname, fds[arg0].fi.pathname] = count(); }'
^C

at-spi-registry    socket:[12950]                      1
bash               pipe:[15831]                        1
crond              /lib64/libnsl-2.12.so                1
crond              /lib64/security/pam_access.so      1
fdisk              /sys/devices/virtual/block/dm-0/dm-na 1
...
...
```

The D programming language does not provide control-flow constructs such as if-statements and loops. However, D does provide the ability to conditionally trace data and modify control flow. D uses logical expressions called *predicates* that can be used to prefix program clauses. A predicate expression is enclosed in // characters and is evaluated at probe firing time, before executing any of the statements associated with the corresponding clause. If the predicate evaluates to true (represented by any non-zero value), the statement list is executed. If the predicate is false (represented by a zero value), none of the statements are executed and the probe firing is ignored.

## D Scripts

- Example of a simple D script:

```
# cat hello.d
dtrace:::BEGIN
{
    trace("hello, world");
    exit(0);
}
```

- Use the `dtrace -s <script>` command to execute:

```
# dtrace -s hello.d
dtrace: script 'hello.d' matched 1 probe
CPU      ID            FUNCTION:NAME
  0        1            :BEGIN  hello, world
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Enabling multiple probes and multiple actions becomes difficult to manage on the command line. For complex tracing, DTrace supports D script files. Each D script file ends with a `.d` suffix and consists of a series of clauses. Each clause describes one or more probes to enable and an optional set of actions to perform when the probe fires. The actions are listed as a series of statements enclosed in braces `{ }`, following the probe name. Each statement ends with a semicolon `(;)`. The example D script in the slide provides the same functionality as the following command-line entry:

```
# dtrace -n 'dtrace:::BEGIN {trace("hello, world");exit(0);}'
dtrace: description 'dtrace:::BEGIN' matched 1 probe
CPU      ID            FUNCTION:NAME
  0        1            :BEGIN  hello, world
```

Use the `dtrace` command with the `-s <script>` option to execute the script:

```
# dtrace -s hello.d
dtrace: script 'hello.d' matched 1 probe
CPU      ID            FUNCTION:NAME
  0        1            :BEGIN  hello, world
```

## D Scripts

- Use the `dtrace -q -s <script>` command to execute the script in “quiet” mode:

```
# dtrace -qs hello.d
hello, world
```

- Alternatively, you can create executable DTrace interpreter files by beginning the script with the following line:

```
#!/usr/sbin/dtrace -s
dtrace:::BEGIN
{
    trace("hello, world");
    exit(0);
}
```

- Give the script execute permission and you can run the script by entering its name on the command line.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The quiet mode option, `dtrace -q`, instructs DTrace to record only the actions that are explicitly stated. This option suppresses the default output that is normally produced by the `dtrace` command. The following example shows the use of the `-q` option in the `hello.d` script:

```
# dtrace -q -s hello.d
hello, world
```

Alternatively, you can create executable DTrace interpreter files. Interpreter files have execute permission and always begin with the following line:

```
#!/usr/sbin/dtrace -s
```

Add the preceding line as the first line in a script to invoke the interpreter. Give the script execute permission and you can run the script by entering its name on the command line as follows:

```
# vi hello.d
#!/usr/sbin/dtrace -qs
...
# chmod +x hello.d
# ./hello.d
hello, world
```

# Using Predicates in a D Script

Example of predicates in a D script:

```
# cat countdown.d
dtrace:::BEGIN
{
    i = 5;
}
profile:::tick-1sec
/i > 0/
{
    trace(i--);
}
profile:::tick-1sec
/i == 0/
{
    trace("blastoff!");
    exit(0);
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The example in the slide implements a predicate in a D script. When executed, the script counts down from 5 and then prints a message and exits. The script uses the `dtrace:::BEGIN` probe to initialize an integer `i` to 10. The script then uses the `profile:::tick-1sec` probe to implement a timer that fires once per second. The first predicate, `/i > 0/`, checks whether the value of `i` is greater than 0. If the value is greater, the action is to use the `trace()` function to decrement the value of `i` by 1.

The script uses the `profile:::tick-1sec` probe a second time and the predicate evaluates if `i` is equal to zero, `/i == 0/`. If this predicate is true, the action is to use the `trace()` function to display the string "blastoff!". A second action is to `exit dtrace` and return to the shell prompt.

The following command executes the D script in the slide and displays the output:

```
# dtrace -s countdown.d
dtrace: script 'countdown.d' matched 3 probes
CPU      ID          FUNCTION:NAME
0        638          :tick-1sec  5
0        638          :tick-1sec  4
0        638          :tick-1sec  3
0        638          :tick-1sec  2
0        638          :tick-1sec  1
0        638          :tick-1sec  blastoff!
```

The following example uses a predicate to evaluate the process ID (`pid`) that is passed as a command-line argument (`$1`), and uses DTrace to observe every time the process performs a `read()` or `write()` system call:

```
# cat rw.d
syscall::read:entry,
syscall::write:entry
/pid == $1/
{
}
```

The following example expects the `pid` of the shell to be 9618. Executing the D script in one window and running shell commands in a second window reports `dtrace` probe firings as follows:

```
# dtrace -s rw.d 9618
dtrace: script 'rw.d' matched 2 probes
CPU      ID          FUNCTION:NAME
0        6            write:entry
0        4            read:entry
0        6            write:entry
0        4            read:entry
^C
```

## Conditional Expressions

D provides support for simple conditional expressions using the `?` and `:` operators. These operators enable a triplet of expressions to be associated where the first expression is used to conditionally evaluate one of the other two. For example, the following D statement could be used to set a variable `x` to one of two strings depending on the value of `i`:

```
x = i == 0 ? "zero" : "non-zero";
```

In this example, the expression `i == 0` is first evaluated. If the expression is true, “zero” is returned as the value of `x`. If the expression is false, “non-zero” is returned. These return values do not invoke a tracking function such as `trace()` or `printf()`. If you want to conditionally trace data, use a predicate instead.

## D Script: Example

The following script includes the use of an array:

```
# cat rwtime.d
syscall::read:entry,
syscall::write:entry
/pid == $1/
{
    ts[probefunc] = timestamp;
}
syscall::read:return,
syscall::write:return
/pid == $1 && ts[probefunc] != 0/
{
    printf("%d nsecs", timestamp - ts[probefunc]);
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The example script in the slide uses an array to trace the elapsed time for each system call. The script instruments both the entry to and return from `read()` and `write()`, and samples the time at each point. Then, on return from a given system call, the script computes the difference between the first and second time stamp. You could use separate variables for each system call, but it is easier to use an associative array indexed by the probe function name.

The first clause defines an array named `ts` and assigns the appropriate member the value of the DTrace variable `timestamp`. This variable returns the value of an always-incrementing nanosecond counter. When the entry time stamp is saved, the corresponding return probe samples `timestamp` again and reports the difference between the current time and the saved value.

The predicate on the return probe requires that DTrace is tracing the appropriate process and that the corresponding entry probe has already fired and assigned `ts[probefunc]` a non-zero value. This trick eliminates invalid output when DTrace first starts. If your shell is already waiting in a `read()` system call for input when you execute `dtrace`, the `read:return` probe fires without a preceding `read:entry` for this first `read()` and `ts[probefunc]` evaluates to zero because it has not yet been assigned.

The following example runs the script and displays an output. The command-line argument uses command substitution to obtain the PID of the bash shell:

```
# dtrace -s rwtime.d '/usr/bin/pgrep - bash'
dtrace: script 'rwtime.d' matched 4 probe
CPU ID          FUNCTION:NAME
  0  7          write:return 15645 nsecs
  0  7          write:return 11175 nsecs
  0  7          write:return 10616 nsecs
  0  5          read:return 3347303916 nsecs
  0  7          write:return 13968 nsecs
  0  7          write:return 11175 nsecs
  0  7          write:return 10616 nsecs
^C
```

## D Script: Example

- The `-C` option invokes the C preprocessor:

```
# dtrace -C -s activity.d
```

- Use D *pragmas* to enable DTrace options:

```
# pragma D option quiet
```

- Clause-local variables reuse storage for each D program clause that relates to a probe.
  - This is similar to automatic variables in C, C++, or Java.
  - Use `this->` notation to assign and reference.

```
this->pid = ((struct task_struct *)arg0)->pid;
time[this->pid] = timestamp;
p_pid[this->pid] = pid;
p_name[this->pid] = execname;
p_exec[this->pid] = "";
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The D script on the following page shows ongoing activity in terms of what program was executing, what its parent is, and how long it ran. Use the following command to run this script:

```
# dtrace -C -s activity.d
```

The `-C` option invokes the C preprocessor (see `man cpp`). You can use the C preprocessor in conjunction with your D programs by specifying the `dtrace -C` option.

This script introduces special D compiler directives called *pragmas*. DTrace is tuned by setting or enabling options and this is accomplished by using D pragmas. You can include D pragmas anywhere in a D script, including outside probe clauses. Begin the line with `#` as follows. This example sets the DTrace runtime `quiet` option (same as `dtrace -q`):

```
#pragma D option quiet
```

This script also introduces clause-local variables. These are variables whose storage is reused for each D program clause that relates to a probe. Clause-local variables are similar to automatic variables in a C, C++, or Java language program, which are active during each invocation of a function. Like all D program variables, clause-local variables are created on their first assignment. These variables are referenced and assigned by applying the `->` operator to the special identifier `this`.

```
# cat activity.d
#pragma D option quiet

proc::do_fork:create
{
    this->pid = ((struct task_struct *)arg0)->pid;
    time[this->pid] = timestamp;
    p_pid[this->pid] = pid;
    p_name[this->pid] = execname;
    p_exec[this->pid] = "";
}

proc::do_execve_common:exec
/p_pid[pid] != 0/
{
    p_exec[pid] = stringof(arg0);
}

proc::do_exit:exit
/p_pid[pid] != 0 && p_exec[pid] != ""/
{
    printf("%s (%d) executed %s (%d) for %d msec\n",
           p_name[pid], p_pid[pid], p_exec[pid], pid,
           (timestamp - time[pid]) / 1000);
}

proc::do_exit:exit
/p_pid[pid] != 0 && p_exec[pid] == ""/
{
    printf("%s (%d) forked itself (as %d) for %d msec\n",
           p_name[pid], p_pid[pid], pid,
           (timestamp - time[pid]) / 1000);
}
```

## Quiz



Which of the following statements are true?

- a. DTrace providers are categories of probes.
- b. DTrace probes are tracing or instrumentation points that enable you to record data at various points of interest.
- c. Oracle makes DTrace available through the ULN.
- d. DTrace for Oracle Linux is being ported to Oracle Solaris.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. DTrace probes are uniquely identified by using the provider:module:function:name format.
- b. dtrace probes are used to initialize state before tracing begins, to process state after tracing has completed, and to handle unexpected execution errors by other probes.
- c. The syscall probes are available for the entry to every system call and for the return from every system call.
- d. Other providers include sched, sdt, io, proc, profile, and fbt.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. DTrace probes can be enabled by provider (-P), by name (-n), by function (-f), by module (-m), and by label (-l).
- b. Use the D programming language to specify probes of interest and to bind actions to those probes.
- c. The following example uses the correct D programming language syntax:

```
# dtrace -n `syscall:::entry
    {trace(timestamp);exit(0);}'
```
- d. The `trace()` built-in function takes a D expression as an argument and traces the result to the directed buffer.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Quiz



Which of the following statements are true?

- a. DTrace provides several built-in functions for aggregating data that individual probes gather.
- b. The name of the aggregation is prefixed with the @ symbol.
- c. DTrace predicates provide the ability to conditionally trace data and modify control flow.
- d. A predicate expression is enclosed in %% characters.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Enable DTrace on Oracle Linux
- View DTrace providers and probes
- Use the D programming language to enable probes and corresponding actions
- Use built-in D variables
- Use built-in D functions
- Create D scripts to explore your system



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice 19: Overview

This practice covers the following topics:

- Installing DTrace packages
- Enabling DTrace modules
- Using DTrace from the command line to list providers and probes
- Using DTrace from the command line to enable probes and actions
- Creating and running D scripts



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.



# A

## Appendix: NIS Configuration

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this appendix, you should be able to:

- Describe NIS authentication
- Describe NIS maps
- Configure NIS server and NIS client
- Configure NIS authentication



**ORACLE®**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# 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



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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 nonsecure 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:
  - passwdbyname: /etc/passwd indexed on username
  - passwdbyuid: /etc/passwd indexed on UID



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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:

- passwdbyname: Indexed on username
- passwdbyuid: 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    mailaliases    protocolsbyname    servicesbyname
groupbyname    netidbyname    protocolsbynumber  servicesbyserv...
hosts.byaddr   passwdbyname  rpcbyname          ypservers
hostsbyname   passwdbyuid  rpcbynumber
```

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        passwdbyname
group         groupbyname
networks      networksbyaddr
hosts         hostsbyname
protocols     protocolsbynumber
services      servicesbyname
aliases       mailaliases
ethers        ethersbyname
```

# 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`
- Enable and start the services: `ypserv`, `ypxfrd`, `yppasswdd`
- Run `ypinit -m` to create the maps.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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/secrenets` 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
```

Use the `systemctl` utility to enable the following services to start at system boot time:

```
# systemctl enable ypserv
# systemctl enable ypxfrd
# systemctl enable yppasswdd
```

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 on the slave server.

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

Use the `systemctl` utility to start the three services:

```
# systemctl start ypserv
# systemctl start ypxfrd
# systemctl start yppasswdd
```

Run the `ypinit` command 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 the NIS domain name in /etc/sysconfig/network:
  - **NISDOMAIN=nisdomainname**
- Specify the NIS server in /etc/yp.conf:
  - **domain nisdomainname server nisservername**
- Start the ypbind service:

```
# service ypbind start
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

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):

```
domain nisdomainname server nisservername
```

Start the ypbind service:

```
# systemctl start ypbind
```

# 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



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After configuring the NIS server and NIS clients, open the Authentication Configuration Tool by entering the following 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.

## Quiz



Which of the following services need to be started on an NIS server?

- a. ypserv
- b. ypxfrd
- c. yppasswdd
- d. ypbind



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Summary

In this appendix, you should have learned how to:

- Describe NIS authentication
- Describe NIS maps
- Configure NIS server and NIS client
- Configure NIS authentication



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

## Practice A: Overview

The practices for this appendix cover the following:

- Configuring an NIS Server
- Configuring an NIS Client
- Implementing NIS Authentication
- Testing NIS Authentication
- Auto-mounting a User Home Directory
- Restoring the Systems to Their Original State



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

