ORACLE
UNIVERSITY

Hardware and Software
Engineered to Work Together

Oracle Linux 7: System Administration

Student Guide | Volume 2
D88168GC10
Edition 1.0 | January 2015 | D89907

Learn more from Oracle University at **oracle.com/education/**

ORACLE

Author

Craig McBride

Technical Contributors and Reviewers

Yasar Akthar

Gavin Bowe

Avi Miller

Chris Potter

Tim Hill

Manish Kapur

Wim Coekaerts

Al Flournoy

Joel Goodman

Harald Van Breederode

Michele Dady

Steve Miller

Antoinette O'Sullivan

Editors

Malavika Jinka

Raj Kumar

Smita Kommini

Graphic Designer

Seema Bopaiah

Publishers

Joseph Fernandez

Giri Venugopal

Contents

13

Btrfs File System

Objectives

After completing this lesson, you should be able to:

- Describe the features of the Btrfs file system
- Create a Btrfs file system
- Create Btrfs subvolumes and snapshots
- Take a snapshot of a file in a Btrfs subvolume
- Mount Btrfs subvolumes and snapshots
- Defragment and resize a Btrfs file system
- Add and remove devices in a Btrfs file system
- Check and repair the integrity of a Btrfs file system
- Convert ext file systems to Btrfs

Btrfs: Introduction

- Jointly developed by a number of companies
- Extent-based file storage
- 50 TB maximum file size, 50 TB maximum file system size
- All data and metadata written via copy-on-write
- Readable and writable snapshots
- Integrated volume management and RAID capabilities
- CRCs for all metadata and data
- Online resizing and defragmentation
- Transparent compression
- Efficient storage for small files
- SSD optimizations and TRIM support

Btrfs is an open-source, general-purpose file system for Linux. The name derives from the use of B-trees to store internal file system structures. Different names are used for the file system, including "Butter F S" and "B-tree F S." Development of Btrfs began at Oracle in 2007, and now a number of companies (including Red Hat, Fujitsu, Intel, SUSE, and many others) are contributing to the development effort. Btrfs is included in the mainline Linux kernel.

Btrfs provides extent-based file storage with a maximum file size of 50 TB and a maximum file system size of 50 TB. All data and metadata is copy-on-write. This means that blocks of data are not changed on disk. Btrfs just copies the blocks and then writes out the copies to a different location. Not updating the original location eliminates the risk of a partial update or data corruption during a power failure. The copy-on-write nature of Btrfs also facilitates file system features such as replication, migration, backup, and restoration of data.

Btrfs allows you to create both readable and writable snapshots. A snapshot is a copy of an entire Btrfs subvolume taken at a given point in time. The snapshots appear as normal directories and you can access the snapshot as you would any other directory. Writable snapshots allow you to roll back a file system to a previous state. You can take a snapshot, perform a system upgrade, and reboot into the snapshot if the upgrade causes problems. All snapshots are writable by default but you also have the option to create read-only snapshots. Read-only snapshots are useful for a backup and then can be deleted when the backup completes.

Btrfs allows a file system to span multiple devices. This is different from logical volume management (LVM) style of volume management. Btrfs does not create block devices; it just creates subvolumes in the file system that can then be mounted like a regular file system.

Btrfs also has built-in RAID support for RAID-0, RAID-1, and RAID-10 levels. Btrfs’s RAID is not a multi-disk RAID like the software RAID devices created by using the `mdadm` command. It is not block RAID either because it does not mirror block devices. Btrfs’s RAID just ensures that for every block, there are “x” amount of copies. For RAID-1, for example, Btrfs just stores two copies of everything on two different devices.

Btrfs maintains CRCs for all metadata and data so everything is checksummed to preserve the integrity of data against corruption. With a RAID-1 or RAID-10 configuration, if checksum fails on the first read, data is pulled off from another copy.

Btrfs has online resizing and defragmentation. You can add or remove devices while the file systems remain online. When a device is removed, the extents stored on it are redistributed to the other devices in the file system. You can also replace devices while Btrfs is online. Btrfs rebalances the extents across the new disk and then you can drop the old disk from a Btrfs array.

Btrfs has transparent compression and currently supports two compression methods: `zlib` and LZO (the default). LZO offers a better compression ratio, whereas `zlib` offers faster compression. Btrfs can determine whether the blocks can be compressed and, therefore, compresses only when possible. You enable compression and specify the compression method by using a `mount` option. For example, to enable LZO or `zlib` compression:

```
# mount -o compress=lzo|zlib <device> <mount_point>
```

You can also force Btrfs to always compress data:

```
# mount -o compress-force <device> <mount_point>
```

Btrfs provides efficient storage for small files. All Linux file systems address storage in block sizes, for example 4 KB. With other file systems, a file that is smaller than 4 KB wastes the leftover space. Btrfs stores these smaller files directly into the metadata, thereby providing a significant performance advantage over other file systems when creating and reading small files.

Btrfs automatically detects solid state drives (SSD) and turns off all optimizations for rotational media. For example, on spinning disks, it is important to store related data close together to reduce seeking. This requires CPU cycles to get good data locality on spinning disks, which is not as important with SSD. TRIM support is also an optimization for SSD. It tells the SSD which blocks are no longer needed and are available to be written over.

Btrfs with Oracle Linux

- Btrfs is production-ready for Oracle Linux since the UEK R2 release.
- Btrfs is currently under technology preview with the RHCK.
- Use the latest Oracle Linux update release and latest UEK to get the most stability and benefits.
- Refer to the Oracle Linux 7 and UEK R3 release notes at http://docs.oracle.com/cd/E52668_01/index.html.

Btrfs is considered production-ready with Oracle Linux since the release of the Unbreakable Enterprise Kernel (UEK) R2 (version 2.6.39). See the press release at http://www.oracle.com/us/corporate/press/1555025.

Btrfs is currently under technology preview with the Red Hat compatible kernel (RHCK). It is strongly recommended to use the latest Oracle Linux update release and latest UEK to get the most stability and benefits.

Several notable features are implemented for Btrfs in UEK R3. Refer to the following release notes at http://docs.oracle.com/cd/E52668_01/index.html:

- Oracle Linux 7 Release Notes
- Oracle Linux Unbreakable Enterprise Kernel Release 3 Release Notes
- Oracle Linux Unbreakable Enterprise Kernel Release 3 Quarterly Update 2 Release Notes
- Oracle Linux Unbreakable Enterprise Kernel Release 3 Quarterly Update 3 Release Notes

Creating a Btrfs File System

- Btrfs utilities are provided by the `btrfs-progs` software package.

```
# rpm -ql btrfs-progs
```

- Use the `mkfs.btrfs` command to create a file system.

```
mkfs.btrfs [options] block_device [block_device ...]
```

- To create a Btrfs file system across two devices:

```
# mkfs.btrfs /dev/sdb /dev/sdc
```

- Mount the Btrfs file system by using the `mount` command, referencing either device:

```
# mount /dev/sdb /btrfs
```


The Btrfs utilities are provided by the `btrfs-progs` software package. Use the following command to list the files provided by the package.

```
# rpm -ql btrfs-progs
```

Use the `mkfs.btrfs` command to create a Btrfs file system. The syntax is:

```
mkfs.btrfs [options] block_device [block_device ...]
```

You can create a Btrfs file system on a single device or on multiple devices. Devices can be disk partitions, loopback devices (disk images in memory), multipath devices, or LUNs that implement RAID in hardware.

Some of the available options for the `mkfs.btrfs` command are:

- **`-A`** ***`offset`*** – Specify the offset from the start of the device for the file system. The default is `0`, which is the start of the device.
- **`-b`** ***`size`*** – Specify the size of the file system. The default is all the available storage.
- **`-d`** ***`type`*** – Specify how the file system data is spanned across the devices. The *`type`* argument must be `raid0`, `raid1`, `raid10`, or `single`.
- **`-l`** ***`size`*** – Specify the leaf size, the least data item in which Btrfs stores data. The default is the page size.
- **`-L`** ***`name`*** – Specify a label name for the file system.

- **`-m`** ***`profile`*** – Specify how the file system metadata is spanned across the devices. The *`profile`* argument must be `raid0`, `raid1`, `raid10`, `single`, or `dup`.
- **`-M`** – Mix data and metadata chunks together for more efficient space utilization. This option affects performance for larger file systems, and is recommended only for file systems that are 1 GB or smaller.
- **`-n`** ***`size`*** – Specify the node size. The default is the page size.
- **`-s`** ***`size`*** – Specify the sector size, which is the minimum block allocation.
- **`-V`** – Print the `mkfs.btrfs` version and exit.

`mkfs.btrfs`: Examples

To create a Btrfs file system on a single block device (for example, `/dev/sdb`):

```
# mkfs.btrfs /dev/sdb
```

To create a Btrfs file system on two block devices (for example, `/dev/sdb` and `/dev/sdc`):

```
# mkfs.btrfs /dev/sdb /dev/sdc
```

The default configuration for a file system with multiple devices is:

- **`-d raid0`** – Stripe the file system data across all devices.
- **`-m raid1`** – Mirror the file system metadata across all devices.

To create a Btrfs file system with multiple devices (`/dev/sdb` and `/dev/sdc`) and stripe both the data and the metadata:

```
# mkfs.btrfs -m raid0 /dev/sdb /dev/sdc
```

To create a Btrfs file system with multiple devices (`/dev/sdb` and `/dev/sdc`) and mirror both the data and the metadata:

```
# mkfs.btrfs -d raid1 /dev/sdb /dev/sdc
```

When you specify a single device, metadata is duplicated on that device unless you specify only a single copy. To create a Btrfs file system on a single block device (for example, `/dev/sdb`) and to specify not to duplicate the metadata:

```
# mkfs.btrfs -m single /dev/sdb
```

For RAID-10 data or metadata, you must specify an even number of at least four devices. To create a Btrfs file system and stripe the data and metadata across mirrored devices (RAID-10):

```
# mkfs.btrfs -d raid10 -m raid10 /dev/sd[bcde]
```

Mounting the File System

Use the `mount` command or make an entry in `/etc/fstab` as you would when mounting any other type of Linux file system. You can reference either device when your file system contains multiple devices. You can also reference the file system label or the UUID. Example:

```
# mount /dev/sdb /btrfs
```

`btrfs` Utility

- The `btrfs` utility requires a subcommand.

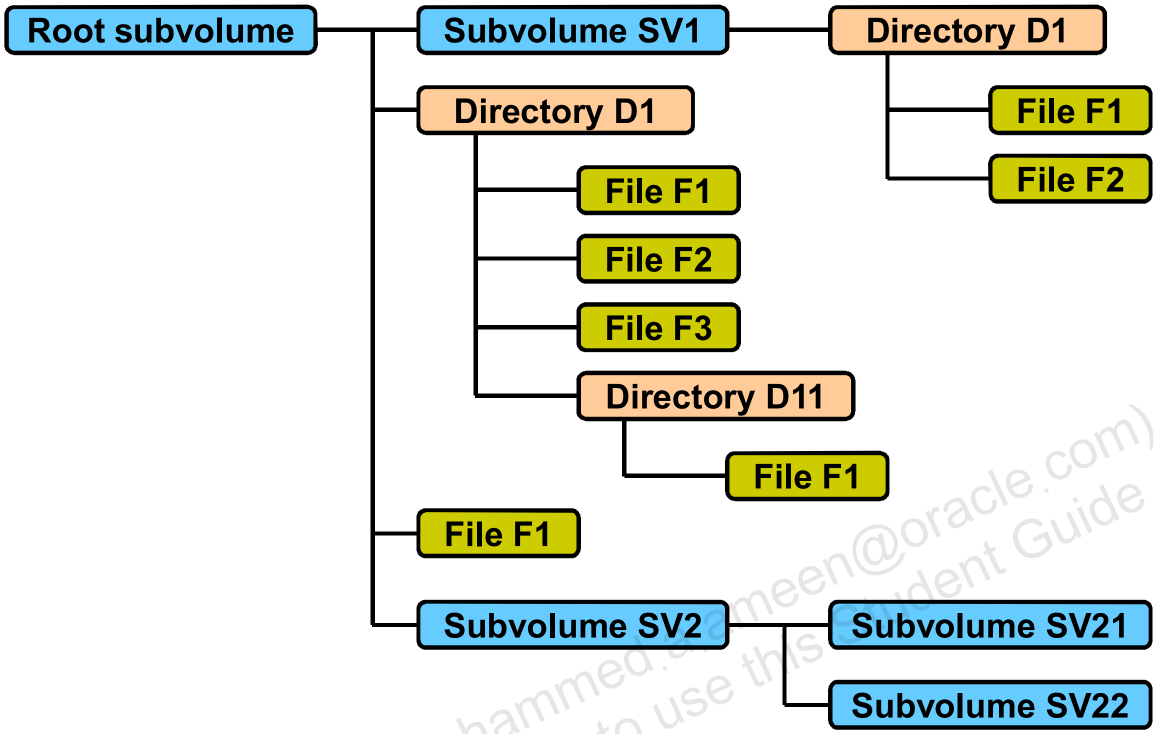
```
# btrfs
usage: btrfs [--help] [--version] <group> [<group>...]
   <command> [<args>]
...
```

- Available subcommands include:
  - `subvolume`
  - `filesystem`
  - `device | replace`
  - `scrub`
  - `check | rescue | restore`
  - `inspect-internal`
  - `send | receive`
  - `quota | qgroup`

Use the `btrfs` command to manage and display information about a Btrfs file system. The command requires a subcommand. Enter `btrfs` without any arguments to list the subcommands:

```
# btrfs
Usage: btrfs [--help] [--version] <group> [<group>...] <command>
[<args>]
        btrfs subvolume create [-i <qgroupid>] [<dest>/]<name>
                Create a subvolume
        btrfs subvolume delete <subvolume> [<subvolume>...]
                Delete subvolume(s)
...
        btrfs filesystem df <path>
                Show space usage information for a mount point
        btrfs filesystem show [--all-devices] [<uuid>|<label>]
                Show the structure of a filesystem
...
```

Btrfs Subvolumes



This slide illustrates a Btrfs file system hierarchy that consists of subvolumes, directories, and files. Btrfs subvolumes are named B-trees that hold files and directories. Subvolumes can also contain subvolumes, which are themselves named B-trees that can also hold files and directories. The top level of a Btrfs file system is also a subvolume, and is known as the *root subvolume*.

The root subvolume is mounted by default and Btrfs subvolumes appear as regular directories within the file system. However, a subvolume can be mounted and only files and directories in the subvolume are accessible. The following example lists the hierarchy displayed in the slide, with the default root subvolume mounted on `/btrfs`:

```
# ls -l /btrfs
drwxr-xr-x ... SV1
drwxr-xr-x ... D1
-rw-r--r-- ... F1
drwxr-xr-x ... SV2
```

Mounting the `SV1` subvolume or the `SV2` subvolume on `/btrfs` allows access only to the files and directories within the respective subvolumes. Remount the root subvolume to gain access to the entire hierarchy.

Use the `btrfs subvolume` command to manage and report on Btrfs subvolumes. A list of the available subvolume commands is as follows:

```
# btrfs subvolume
usage: btrfs subvolume <command> <args>
     btrfs subvolume create [-i <qgroupid>] [<dest>/]<name>
          Create a subvolume
     btrfs subvolume delete <subvolume> [<subvolume>...]
          Delete a subvolume(s)
     btrfs subvolume list [options] [-G [+|-]value] [-C [+|-
]value] [--sort=gen,ogen,rootid,path] <path>
          List subvolumes (and snapshots)
     btrfs subvolume snapshot [-r] <source> <dest>|
[<dest>/]<name>
          btrfs subvolume snapshot [-r] [-i <qgroupid>] <source>
<dest>|[<dest>/]<name>
     btrfs subvolume get-default <path>
          Get the default subvolume of a filesystem
     btrfs subvolume set-default <subvolid> <path>
          Set the default subvolume of a filesystem
     btrfs subvolume find-new <path> <lastgen>
          List the recently modified files in a filesystem
     btrfs subvolume show <subvol-path>
          Show more information of the subvolume
```

The word "`subvolume`" in the `btrfs` command can be abbreviated to "`sub`". For example, both of the following commands are valid:

```
# btrfs subvolume create /btrfs/SV1
# btrfs sub create /btrfs/SV1
```

The abbreviation applies to other `btrfs` subcommands as well. For example, both of the following subcommands are valid:

```
# btrfs filesystem df /btrfs
# btrfs file df /btrfs
```

`btrfs subvolume` Utilities

- Use the `btrfs subvolume create` command to create a subvolume on a mounted Btrfs file system, such as:

```
# btrfs subvolume create /btrfs/SV1
```

- The subvolume appears as a normal directory when the `ls` command is used (only a partial output is shown):

```
# ls -l /btrfs
drwxr-xr-x   ...   SV1
```

- Use the `btrfs subvolume list` command to view the subvolumes in a Btrfs file system, as in this example:

```
# btrfs subvolume list /btrfs
ID 258 gen 10 top level 5 path SV1
```


Use the `btrfs subvolume create` command to create a subvolume. The following example creates a subvolume named `SV1` on a Btrfs file system mounted on `/btrfs`:

```
# btrfs subvolume create /btrfs/SV1
Create subvolume '/btrfs/SV1'
```

The subvolume appears as a regular directory. The following example creates a regular directory in `/btrfs` and then displays the content:

```
# mkdir /btrfs/D1
# ls -l /btrfs
drwxr-xr-x ... D1
drwxr-xr-x ... SV1
```

Use the `btrfs subvolume list` command to view only the subvolumes in a Btrfs file system, as in this example:

```
# btrfs subvolume list /btrfs
ID 258 gen 10 top level 5 path SV1
```

This command also displays the subvolume ID (`258`), root ID generation of the B-tree (`10`), and the top-level ID (`5`). These fields are described later in this lesson.

Btrfs Snapshots

- A snapshot is a point-in-time copy of a subvolume.
- Snapshots are created quickly and initially consume very little disk space.
- Use the `btrfs subvolume snapshot` command to create a snapshot of a subvolume.
- The following example creates a writable/readable snapshot named `SV1-snap` of the `SV1` subvolume:

```
# btrfs subvolume snapshot /btrfs/SV1 /btrfs/SV1-snap
```

- Use the `-r` option to create a read-only snapshot:

```
# btrfs subvolume snapshot -r /btrfs/SV1 /btrfs/SV1-
  rosnap
```


Btrfs subvolumes can be snapshotted and cloned, which creates additional B-trees. A snapshot starts as a copy of a subvolume taken at a point in time. You can make a snapshot writable and use it as an evolving clone of the original subvolume. Or you can use the snapshot as a stable image of a subvolume for backup purposes or for migration to other systems. Snapshots can be created quickly and they initially consume very little disk space.

Use the `btrfs subvolume snapshot` command to create a writable/readable snapshot of a subvolume. The following example creates a snapshot of the `SV1` subvolume:

```
# btrfs subvolume snapshot /btrfs/SV1 /btrfs/SV1-snap
Create a snapshot of '/btrfs/SV1' in '/btrfs/SV1-snap'
```

Use the `btrfs subvolume snapshot -r` option to create a read-only snapshot:

```
# btrfs subvolume snapshot -r /btrfs/SV1 /btrfs/SV1-rosnap
Create a readonly snapshot of '/btrfs/SV1' in '/btrfs/SV1-
rosnap'
```

The snapshots appear as a regular directory when the `ls` command is used. Snapshots also appear in the output of the `btrfs subvolume list` command.

Taking a Snapshot of a File

- Use the `cp --reflink` command to take a snapshot of a file.
- A new file shares the same disk blocks as the original file.
- The copy operation is almost instantaneous and also saves disk space.
- This operation works only within the boundaries of the same Btrfs file system and within the same subvolume.
- Example:

```
# cp --reflink /btrfs/SV1/file /btrfs/SV1/copy_of_file
```

ORACLE

You can use the `cp --reflink` command to take a snapshot of a file. With this option, the file system does not create a new link pointing to an existing inode, but instead creates a new inode that shares the same disk blocks as the original copy. The new file appears to be a copy of the original file but the data blocks are not duplicated. This allows the copy to be almost instantaneous and also saves disk space. As the file's content diverges over time, its amount of required storage grows. One restriction is that this operation can work only within the boundaries of the same file system and within the same subvolume.

The following example copies a file by using the `cp --reflink` command. The space used is given both before and after the copy operation. Note that the space used does not increase.

```
# df –h /btrfs
Filesystem    Size   Used   Avail   Use%   Mounted on
/dev/sdb       16G   8.2M    14G     1%    /btrfs
# cp --reflink /btrfs/SV1/vmlinuz* /btrfs/SV1/copy_of_vmlinuz
# df –h
Filesystem    Size   Used   Avail   Use%   Mounted on
/dev/sdb       16G   8.2M    14G     1%    /btrfs
```

Mounting a Subvolume or Snapshot

- To mount a subvolume or snapshot, you must first determine the ID number.
- Use the `btrfs subvolume list` command to display the ID numbers, as in this example:

```
# btrfs subvolume list /btrfs
ID 258 gen 12 top level 5 path SV1
ID 259 gen 9 top level 5 path SV1-snap
```

- Use the `btrfs subvolume set-default` command to change the ID number to the entity to be mounted:

```
# btrfs subvolume set-default 259 /btrfs
```

- Unmount and remount the file system.
- Alternatively, use `-o subvolid=#` when mounting the file system, but this does not change the default subvolume ID.

By default, Linux mounts the parent Btrfs volume, which has an ID of `0`. In this example, the following `mount` command was issued before creating any subvolumes and snapshots:

```
# mount /dev/sdb /btrfs
```

The subvolume `SV1` was created in `/btrfs`. The `ls` command shows the subvolume:

```
# ls -l /btrfs
drwx------   ...   SV1
```

The following example copies files into `SV1`, creates a snapshot of `SV1`, and verifies that both the subvolume and the snapshot contain the same files:

```
# cp /boot/vmlinuz-3.8.13-35* /btrfs/SV1
# btrfs sub snapshot /btrfs/SV1 /btrfs/SV1-snap
# ls /btrfs/SV1*
/btrfs/SV1:
vmlinuz-3.8.13-35.3.1.el7uek.x86_64
/btrfs/SV1-snap:
vmlinuz-3.8.13-35.3.1.el7uek.x86_64
```

If you unmount /btrfs and remount it, the parent Btrfs volume is mounted by default:

```
# ls /btrfs
SV1  SV1-snap
# umount /btrfs
# mount /dev/sdb /btrfs
# ls /btrfs
SV1  SV1-snap
```

You can, however, mount a btrfs subvolume or snapshot as though it were a disk device. If you mount a snapshot instead of its parent subvolume, you effectively roll back the state of the file system to the time that the snapshot was taken.

The following example copies a file to SV1 so that the content is different from SV1-snap:

```
# cp ~/test-file /btrfs/SV1
# ls /btrfs/SV1*
/btrfs/SV1:
test-file       vmlinuz-3.8.13-35.3.1.el7uek.x86_64
/btrfs/SV1-snap:
vmlinuz-3.8.13-35.3.1.el7uek.x86_64
```

To mount a subvolume or snapshot, you must first determine the ID number of the subvolume that you want to mount. Use the btrfs subvolume list command to display the ID numbers. In the following example, the ID of the root subvolume is 5:

```
# btrfs subvolume list /btrfs
ID 258 gen 12 top level 5 path SV1
ID 259 gen 9 top level 5 path SV1-snap
```

Use the btrfs subvolume set-default command to set the default subvolume of a file system. For example, to mount the SV1 Btrfs subvolume, which has an ID of 258:

```
# btrfs subvolume set-default 258 /btrfs
```

You then need to unmount and remount the Btrfs file system. The root level then contains the contents of the SV1 subvolume and the root subvolume is no longer visible:

```
# umount /btrfs
# mount /dev/sdb /btrfs
# ls /btrfs
test-file       vmlinuz-3.8.13-35.3.1.el7uek.x86_64
```

You can also use the -o subvolid option to the mount command to mount the root subvolume or a subvolume or snapshot. For example, to mount the root subvolume:

```
# umount /btrfs
# mount -o subvolid=5 /dev/sdb /btrfs
# ls /btrfs
SV1  SV1-snap
```

`btrfs filesystem` Utilities

- Use the `btrfs filesystem` command to manage and report on Btrfs file systems.
- Available commands include:
  - `btrfs filesystem df`
  - `btrfs filesystem show`
  - `btrfs filesystem sync`
  - `btrfs filesystem defragment`
  - `btrfs filesystem resize`
  - `btrfs filesystem balance`
  - `btrfs filesystem label`
- For example, to display the file system label:

```
# btrfs filesystem label /btrfs
Btrfs
```


Use the `btrfs filesystem` command to manage and report on Btrfs file systems. A partial list of the available commands is as follows:

```
# btrfs filesystem
Usage: btrfs filesystem [<group>] <command> [<args>]
    btrfs filesystem df <path>
        Show space usage information for a mount point
    btrfs filesystem show [options|<path>|<uuid>]
        Show the structure of a filesystem
    btrfs filesystem sync <path>
        Force a sync on a filesystem
    btrfs filesystem defragment [options] <file>|<dir> [...]
        Defragment a file or a directory
    btrfs filesystem resize [devid:][+/-]<newsize>[gkm]|...
        Resize a filesystem
...
```

`btrfs filesystem df` Utility

- Use the `btrfs filesystem df` command to show accurate space usage information for a mount point:

```
# btrfs filesystem df /btrfs
Data, RAID1: total=1.00GiB, used=5.18MiB
Data, single: total=8.00MiB, used=0.00
System, RAID1: total=8.00MiB, used=16.00KiB
System, single: total=4.00MiB, used=0.00
Metadata, RAID1: total=1.00GiB, used=112.00KiB
Metadata, single: total=8.00MiB, used=0.00
```

- Btrfs allocates space on disks in chunks.
  - A chunk is 1 GB for data and 256 MB for metadata.
  - A chunk also has a specific RAID profile associated with it.

Some information is presented when you create a Btrfs file system. The following example creates a Btrfs file system with two 5 GB devices (`/dev/sdb` and `/dev/sdc`) and mirrors both the data and the metadata (metadata is mirrored by default):

```
# mkfs.btrfs –L Btrfs –d raid1 /dev/sdb /dev/sdc
...adding device /dev/sdc id 2
fs created label Btrfs on /dev/sdb
     nodesize 16384 leafsize 16384 sectorsize 4096 size 10.00GiB
Btrfs v3.12
```

The preceding output shows that the block size is 4 KB with a total of 10 GiB of space. But because the array is RAID1, you can fit only 5 GB of data on this file system. You actually have less than 5 GB because space is needed for the metadata as well. The example continues with creating a mount point and mounting the file system:

```
# mkdir /btrfs
# mount /dev/sdb /brtfs
```

As previously discussed, you can mount by referencing either device in the array, the LABEL, or the UUID.

Even the /proc/mounts file does not show the second device for the Btrfs file system:

```
# grep btrfs /proc/mounts
/dev/sdb /btrfs btrfs rw,seclabel,relatime,space_cache 0 0
```

For example, the following command copies a file to the Btrfs file system:

```
# cd /btrfs
# cp /boot/vmlinuz-3.10* .
# ls -l
-rwxr-xr-x ... Vmlinuz-3.10...
```

When the file system is mounted and has a file copied to it, the output of the df command produces inaccurate information for the Btrfs file system:

```
# sync
# df -h
Filesystem  Size  Used  Avail  Use%  Mounted on
...
/dev/sdb     10g   11M   8.0G    1%  /btrfs
```

This output shows that the file system has a size of 10 G, which is not accurate because this is a RAID-1 array. To get accurate space information for a Btrfs file system, use the btrfs filesystem df command:

```
# btrfs filesystem df /btrfs
Data, RAID1: total=1.00GiB, used=5.18MiB
Data, single: total=8.00MiB, used=0.00
System, RAID1: total=8.00MiB, used=16.00KiB
System, single: total=4.00MiB, used=0.00
Metadata, RAID1: total=1.00GiB, used=112.00KiB
Metadata, single: total=8.00MiB, used=0.00
```

Btrfs allocates space on disks in chunks. A chunk is 1 GB for data and 256 MB for metadata. A chunk also has a specific RAID profile associated with it, which allows Btrfs to have different allocation profiles for data and for metadata. The output of the btrfs filesystem df command shows that it has allocated only a 1 GB chunk of RAID-1 at this time.

Btrfs is not yet actually “RAIDing” the entire device. For example, if you specify RAID-1 for metadata and RAID-0 for data, metadata writes are mirrored across all the disks and data writes are striped across the disks.

The output of the btrfs filesystem df command shows that you are currently using 5.18 MB. The disk (system RAID1) has a total allocated space of 8 MB and has used 16 KB. Metadata is allocated 1 GB of space as well; it has used 112 KB of it.

`btrfs filesystem show`|`sync` Utilities

- Use the `btrfs filesystem show` command to display the structure of a file system, as in this example:

```
# btrfs filesystem show
Label: Btrfs  uuid: ...
        Total devices 2 FS bytes used 9.69MiB
        devid 1 size 5.00GiB used 2.03GiB path /dev/sdc
        devid 2 size 5.00GiB used 2.01GiB path /dev/sdb
```

- Use the `btrfs filesystem sync` command to force a sync for the file system:

```
# btrfs filesystem sync /btrfs
FSSync '/btrfs'
```


Use the `btrfs filesystem show` command to display the structure of a file system. The syntax follows:

```
btrfs filesystem show [options|<path>|<uuid>
```

If you omit the optional `path` and `uuid`, the command shows information about all the Btrfs file systems.

The following example displays the structure of a Btrfs file system:

```
# btrfs filesystem show
Label: Btrfs  uuid: ...
        Total devices 2 FS bytes used 9.69MiB
        devid    1 size 5.00GiB used 2.03GiB path /dev/sdc
        devid    2 size 5.00GiB used 2.01GiB path /dev/sdb
```

Use the `btrfs filesystem sync` command to force a sync for the file system. The file system must be mounted. To force a sync of the file system mounted on `/btrfs`:

```
# btrfs filesystem sync /btrfs
FSSync '/btrfs'
```

btrfs filesystem defragment Utility

- Use the `btrfs filesystem defragment` command to defragment a file system, file, or directory.
- To defragment a file system:

```
# btrfs filesystem defragment /btrfs
```

- To defragment and compress a file system:

```
# btrfs filesystem defragment -c /btrfs
```

- Set up automatic defragmentation by specifying the `autodefrag` option with the `mount` command:

```
# mount -o autodefrag /dev/sdb /btrfs
```


Btrfs provides online defragmentation of a file system, file, or directory. The online defragmentation facility reorganizes data into contiguous chunks wherever possible to create larger sections of available disk space and to improve read and write performance. Use the `btrfs filesystem defragment` command to defragment a file or a directory.

```
btrfs filesystem defragment [options] <file>|<dir> [...]
```

The available options include the following:

- `-v` – Verbose
- `-c` – Compress file contents while defragmenting.
- `-r` – Defragment files recursively.
- `-f` – Flush file system after defragmenting.
- `-s` *`start`* – Defragment only from byte *`start`* onward.
- `-l` *`len`* – Defragment only up to *`len`* bytes.
- `-t` *`size`* – Defragment files only at least *`size`* bytes.

You can set up automatic defragmentation by specifying the `-o autodefrag` option when you mount the file system. Do not defragment with kernels up to version `2.6.37` if you have created snapshots or made snapshots of files by using the `cp --reflink` option. Btrfs in these earlier kernels unlinks the copy-on-write copies of data.

btrfs filesystem resize Utility

- Use the `btrfs filesystem resize` command to resize a file system.
- To accommodate the resizing, you must have space available on the underlying devices.
- To reduce the file system by 2 GB:

```
# btrfs filesystem resize -2G /btrfs
```

- To increase the file system by 2 MB:

```
# btrfs filesystem resize +2M /btrfs
```

- To have the file system occupy all available space:

```
# btrfs filesystem resize max /btrfs
```


Btrfs provides online resizing of a file system. Use the `btrfs filesystem resize` command to resize a file system. You must have space available to accommodate the resizing because the command has no effect on the underlying devices. The syntax is as follows:

```
btrfs filesystem resize [devid:][+/-]<newsize>[gkm]|[devid:]max
<path>
```

Descriptions of the parameters:

- `+ newsize` – Increases the file system size by *newsize* amount
- `- newsize` – Decreases the file system size by *newsize* amount
- *newsize* – Specifies the *newsize* amount
- `g`, `k`, or `m` – Specifies the unit of *newsize* (GB, KB, or MB). If no units are specified, the parameter defaults to bytes.
- `max` – Specifies that the file system occupies all available space

For example, to reduce the size of the file system by 2 GB:

```
# btrfs filesystem resize -2G /btrfs
Resize '/btrfs/' of '-2G'
```

`btrfs device` Utilities

- Use the `btrfs device` command to manage devices on Btrfs file systems.
- Available commands include:
  - `btrfs device add|delete|scan|ready|stats`
- The `btrfs device scan` command scans physical devices looking for members of a Btrfs volume.
  - This allows a multiple-disk Btrfs file system to be mounted without specifying all the disks on the `mount` command.
- `Udev` automatically runs `btrfs device scan` on boot.
- The `btrfs device ready` command checks whether all devices are in cache for mounting.
- The `btrfs device stats` command shows IO stats.

Use the `btrfs device` command to manage devices on Btrfs file systems. A list of the available commands is as follows:

```
# btrfs device
Usage: btrfs device <command> [<args>]
    btrfs device add [options] <device> [<device>...] <path>
        Add a device to a filesystem
    btrfs device delete <device> [<device>...] <path>
        Remove a device from a filesystem
    btrfs device scan [<--all-devices>|<device>| [<device>...]]
        Scan devices for a btrfs filesystem
...
```

The `btrfs device scan` command scans physical devices looking for members of a Btrfs volume. This command allows a multiple-disk Btrfs file system to be mounted without specifying all the disks on the `mount` command.

You do not need to run `btrfs device scan` from the command line, because `udev` automatically runs `btrfs device scan` on boot.

btrfs device Utility: Examples

- Use the `btrfs device add` command to add a device to a mounted file system, as in this example:

```
# btrfs device add /dev/sdd /btrfs
```

- Use the `btrfs filesystem balance` command after adding a device:

```
# btrfs filesystem balance /btrfs
```

- Use the `btrfs device delete` command to remove a device from a file system:

```
# btrfs device delete /dev/sdd /btrfs
```


Use the `btrfs device add` command to add a device to a file system. In this example, the current file system structure is as follows:

```
# btrfs filesystem show
Label: Btrfs  uuid: ...
        Total devices 1 FS bytes used 4.86MiB
        devid    1 size 5.00GB used 276.00MiB path /dev/sdb
```

The `btrfs filesystem df` command shows:

```
# btrfs filesystem df /btrfs
Data, single: total=8.00MiB, used=4.74MiB
System, single: total=4.00MiB, used=16.00KiB
Metadata, single: total=264.00MiB, used=112.00KiB
```

The output of the `df` command shows:

```
# df -h /btrfs
Filesystem  Size  Used  Avail  Use%  Mounted on
/dev/sdb    5.0G  4.9M   4.8G    1%  /btrfs
```

Add a 5 GB disk, /dev/sdd, to the file system mounted on /btrfs by using the btrfs device add command:

```
# btrfs device add /dev/sdd /btrfs
```

The output of the btrfs filesystem show command shows the newly added device:

```
# btrfs file show
Label: Btrfs  uuid: ...
        Total devices 2 FS bytes used 4.86MiB
        devid    1 size 5.00GiB used 276.00MiB path /dev/sdb
        devid    2 size 5.00GiB used 0.00 path /dev/sdc
```

The output of the btrfs filesystem df command shows no difference after adding the new device:

```
# btrfs filesystem df /btrfs
Data, single: total=8.00MiB, used=4.74MiB
System, single: total=4.00MiB, used=16.00KiB
Metadata, single: total=264.00MiB, used=112.00KiB
```

There is no difference in the output because the newly added device has not yet been allocated for either data or metadata.

The additional size is reflected in the output of df:

```
# df -h /btrfs
Filesystem  Size  Used  Avail  Use%  Mounted on
/dev/sdb     10g  4.9M   9.8G    1%  /btrfs
```

After adding a device, it is recommended that you run the following balance command on the file system:

```
# btrfs filesystem balance /btrfs
```

Running this command redistributes space by balancing the chunks of the file system across all the devices. This command also reclaims any wasted space.

Use the btrfs device delete command to remove a device from a file system.

Example:

```
# btrfs device delete /dev/sdd /btrfs
```

`btrfs scrub` Utilities

- Use the `btrfs scrub` command to manage scrubbing on Btrfs file systems.
- Scrubbing is performed in the background by default. It attempts to report and repair bad blocks on the file system.
- Available commands include:
  - `btrfs scrub start`
  - `btrfs scrub cancel`
  - `btrfs scrub resume`
  - `btrfs scrub status`

You can initiate a check of the entire file system by triggering a file system scrub job. The scrub job runs in the background by default and scans the entire file system for integrity. It automatically attempts to report and repair any bad blocks that it finds along the way. Instead of going through the entire disk drive, the scrub job deals only with data that is actually allocated. Depending on the allocated disk space, this is much faster than performing an entire surface scan of the disk.

Scrubbing involves reading all the data from all the disks and verifying checksums. If any values are not correct, the data can be corrected by reading a good copy of the block from another drive. The scrubbing code also scans on read automatically. It is recommended that you scrub high-usage file systems once a week and all other file systems once a month.

The following is a partial list of the available `btrfs scrub` commands:

```
# btrfs scrub
Usage: btrfs scrub <command> [options] <path>|<device>
     btrfs scrub start [-BdqrR] [-c ioprio_class ...
          Start a new scrub
...
     btrfs scrub status [-dR] <path>|<device>
          Show status of running or finished scrub
```

`btrfs scrub` Utility: Examples

- Use the `btrfs scrub start` command to start a scrub on all the devices of a file system or on a single device.

```
# btrfs scrub start /btrfs
```

- Use the `btrfs scrub status` command to get the status of a scrub job. The following example includes detailed scrub information about each device in the file system:

```
# btrfs scrub status -dR /btrfs
```

- Use the `btrfs scrub cancel` command to cancel a running scrub job:

```
# btrfs scrub cancel /btrfs
```

- Use the `btrfs scrub resume` command to resume a previously canceled or interrupted scrub:

```
# btrfs scrub resume /btrfs
```


Use the `btrfs scrub start` command to start a scrub on all the devices of a file system or on a single device. The syntax is as follows:

```
btrfs scrub start [-BdqrR] [-c ioprio_class ...
```

Description of options:

- `-B` – Do not run in the background and print statistics when finished.
- `-d` – Print separate statistics for each device of the file system. This option is used in conjunction with the `-B` option.
- `-q` – Run in quiet mode, omitting error messages and statistics.
- `-r` – Run in read-only mode, not correcting any errors.
- `-R` – Raw print mode. Print full data instead of summary.
- `-c ioprio_class` – Set IO priority class (see `ionice(1)` man page).
- `-n ioprio_classdata` – Set IO priority classdata (see `ionice(1)` man page).

The following example starts a scrub on the Btrfs file system that is mounted on `/btrfs`.

```
# btrfs scrub start /btrfs
scrub started on /brtfs, fsid ... (pid=...)
```

Use the `btrfs scrub status` command to get the status of a scrub job. One option is available:

- `-d` – Print separate statistics for each device of the file system.

The following is a partial output from the `btrfs scrub status` command:

```
# btrfs scrub status /btrfs
Scrub status for ...
        scrub started at ... and finished after 0 seconds
        total bytes scrubbed: 10.60MiB with 1 errors
        error details: csum=1
        corrected errors: 1, uncorrectable errors: 0,
        unverified errors: 0
```

You can also cancel a running scrub job. Progress is saved in the scrub progress file and you can resume scrubbing later.

To cancel a scrub:

```
# btrfs scrub cancel /btrfs
```

To resume a canceled or interrupted scrub job:

```
# btrfs scrub resume /btrfs
```

The `scrub resume` command has the same options as the `scrub start` command.

Btrfs stores the last two minutes, at 30-second intervals, of root ID generations. Btrfs continues to keep rolling these generations, even if there are no changes in the file system.

If a scrub does not correct errors, you can use the following mount option to roll back to a known good B-tree, given that the rest of the tree is available because of copy-on-write:

```
# mount -o recovery /dev/xvdb /btrfs
```

Converting Ext File Systems to Btrfs

- Use the `btrfs-convert` utility to convert an ext2, ext3, or ext4 file system to a Btrfs file system.
- To convert a non-`root` ext file system:
  1. Unmount the ext file system.
  2. Use `fsck` to check the integrity of the ext file system.
  3. Use the `btrfs-convert` utility to convert the file system.
  4. Edit `/etc/fstab` and change the file system type to `btrfs`.
  5. Mount the converted file system on the original mount point.
- The syntax of the `btrfs-convert` utility is as follows:

```
btrfs-convert <device>
```

- You cannot convert the `root` file system or a bootable partition, such as `/boot`, to Btrfs.

Btrfs supports the conversion of ext2, ext3, and ext4 file systems to Btrfs file systems. The original ext file system metadata is stored in a snapshot named `ext#_saved` so that the conversion can be reversed if necessary.

Use the `btrfs-convert` utility to convert an ext file system. Always make a backup copy before converting a file system. To convert a non-`root` ext file system, perform the steps listed in the slide.

You cannot convert the `root` file system or a bootable partition, such as `/boot`, to Btrfs.

Quiz

Which of the following statements are true?

a. Btrfs is a general-purpose file system.
b. All Btrfs data and metadata is written via copy-on-write.
c. Btrfs supports only readable snapshots.
d. Btrfs supports online resizing and defragmentation.

Quiz

Which of the following statements are true?

a. Btrfs has built-in RAID support for RAID-0, RAID-1, RAID-5, RAID-6, and RAID-10.

b. Btrfs supports transparent compression.

c. Btrfs automatically detects and optimizes solid state drives.

d. Oracle Linux with UEK2 is the first release to officially support Btrfs.

Quiz

Which of the following are valid `btrfs` commands?

a. `btrfs subvolume create`
b. `btrfs snapshot create`
c. `btrfs filesystem show`
d. `btrfs device create`

Summary

In this lesson, you should have learned how to:

- Describe the features of the Btrfs file system
- Create a Btrfs file system
- Create Btrfs subvolumes and snapshots
- Take a snapshot of a file in a Btrfs subvolume
- Mount Btrfs subvolumes and snapshots
- Defragment and resize a Btrfs file system
- Add and remove devices in a Btrfs file system
- Check and repair the integrity of a Btrfs file system
- Convert ext file systems to Btrfs

Practice 13: Overview

This practice covers the following topics:

- Creating Btrfs file systems with different specifications
- Resizing a Btrfs file system
- Adding a disk to and removing a disk from a Btrfs file system
- Creating a Btrfs subvolume and snapshot
- Mounting a subvolume and a snapshot
- Taking a snapshot of a file by using the `cp --reflink` command
- Corrupting data on a Btrfs file system and recovering from data corruption

14

Storage Administration

Objectives

After completing this lesson, you should be able to:

- Describe the Linux device mapper
- Describe Logical Volume Manager (LVM)
- Configure LVM components
- Back up and restore volume group metadata
- Describe LVM thin provisioning
- Describe snapper
- Describe Linux kernel multi-disk (MD) driver
- Describe RAID and configure RAID devices

Logical Volume Manager (LVM)

- LVM is a tool to facilitate the management of physical volumes, volume groups, and logical volumes.
  - Physical volume (PV): A physical storage device
  - Volume group (VG): Physical volumes are grouped together into storage pools called volume groups.
  - Logical volume (LV): Each volume group is divided into multiple LVs.
- File systems are created on LVs.
- Use LVM to increase the size of VGs and LVs “on the fly” (without interrupting operations).

ORACLE

The Linux device mapper (DM) provides an abstraction layer on top of the actual storage block devices and provides the foundation for Logical Volume Manager (LVM2), RAID, encryption, and other storage features. LVM2 manages multiple physical volumes and also supports mirroring and striping of logical volumes to provide redundancy and increase performance. To assist in understanding LVM, the following terms are defined:

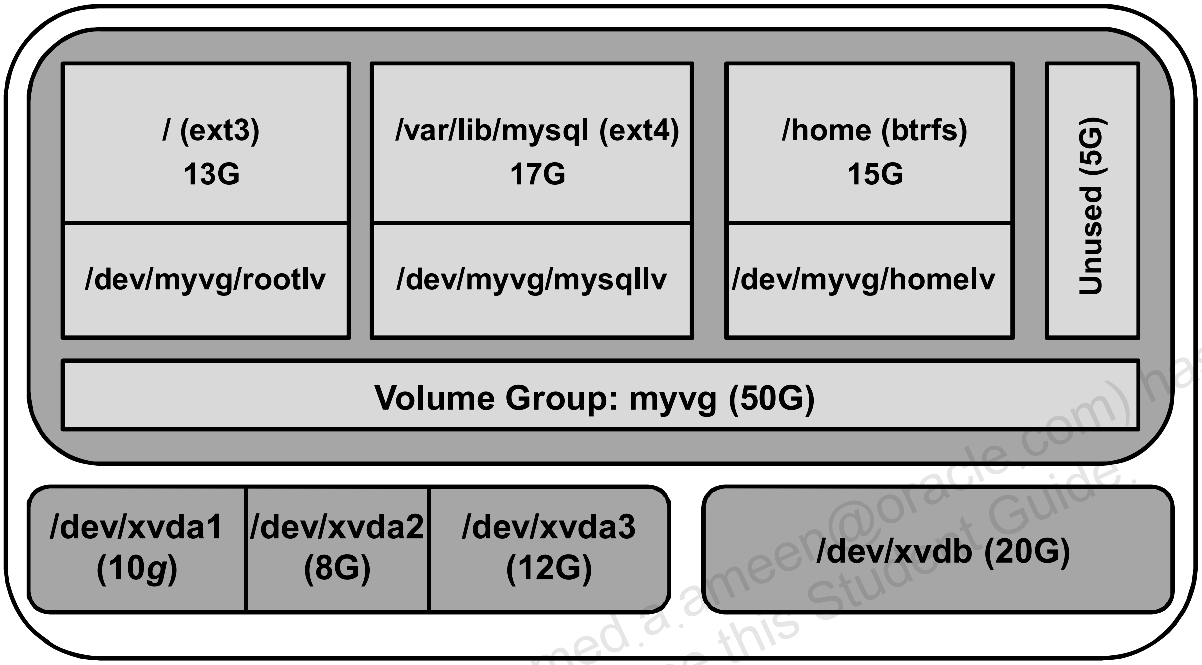
- **Physical volumes:** These are physical storage devices (hard drives, partitions, arrays).
- **Volume groups:** Physical volumes are grouped together into volume groups.
- **Logical volumes:** Each volume group is divided into multiple logical volumes.

Each logical volume is analogous to a standard disk partition. Logical volumes, therefore, function as partitions that can span multiple physical disks.

File systems, such as ext3 or ext4, can be created on logical volumes and connected to the directory hierarchy through mount points. As these “partitions” become filled with data, use LVM to increase their capacity from free space in the volume group. New physical storage devices are added to volume groups to increase the capacity of these groups.

With LVM, capacity is expanded in logical volumes “on the fly” (dynamically) without the need to back up the data on standard partitions, modify the partition table, and restore the data. Logical volume management does not interrupt usage and is transparent to users.

LVM Configuration: Example



This slide illustrates a possible LVM configuration. There are four physical volumes (PV), with three of these being partitions on one drive, and the fourth being an entire hard drive:

- xvda1: 10 GB
- xvda2: 8 GB
- xvda3: 12 GB
- xvdb: 20 GB

All of the PV are grouped into a single volume group (VG), named `myvg`. The storage capacity of this group is 50 GB, which is the total space of the four PV.

The volume group is divided into three logical volumes (LV). The following lists the LV name, the size, the mount point, and the file system type of each logical volume:

- `rootlv`, 13 GB, `/` (root), ext3
- `mysqllv`, 17 GB, `/var/lib/mysql`, ext4
- `homelv`, 15 GB, `/home`, btrfs

Finally, the illustration shows that there is 5 GB of unused space in the VG. This is available to be allocated to any of the existing logical volumes, or to a new logical volume.

Physical Volume Utilities

- Use the `pvcreate` command to create physical volumes:

```
# pvcreate -v /dev/xvdd1 /dev/xvdd2
```

- The following commands display physical volumes:
  - `pvdisplay`
  - `pvs`
  - `pvscan`
- Use the `pvremove` command to remove physical volumes:

```
# pvremove /dev/xvdd1
```

- Additional PV commands are available:
  - `pvchange`: Change the attributes of physical volumes.
  - `pvresize`: Resize physical volumes.
  - `pvck`: Check the consistency of physical volumes.
  - `pvmove`: Move extents from one physical volume to another.

The first step in implementing LVM is to create physical volumes. In addition to creating physical volumes, commands exist to display the attributes of physical volumes, remove physical volumes, and perform other functions on physical volumes.

Creating Physical Volumes

Use the `pvcreate` command to create physical volumes. The syntax is:

```
pvcreate [options] device
```

You can initialize multiple disks or partitions for use by LVM in the same command. For example, the following command initializes two partitions. The `–v` option makes the output more verbose:

```
# pvcreate -v /dev/xvdd1 /dev/xvdd2
    Set up physical volume for "/dev/xvdd1" with ...
    Zeroing start of device /dev/xvdd1
    Writing physical volume data to disk "/dev/xvdd1"
Physical volume "/dev/xvdd1" successfully created
    Set up physical volume for "/dev/xvdd2" with ...
    Zeroing start of device /dev/xvdd2
...
```

Displaying Physical Volumes

Use the pvdisplay command to display attributes of physical volumes.

```
# pvdisplay
  "/dev/xvdd1" is a new physical volume of "1.00 GiB"
  --- NEW Physical volume ---
  PV Name        /dev/xvdd1
...
```

In addition to pvdisplay, two other commands list information about physical volumes. The pvs command reports information about physical volumes in a more condensed form. The pvscan command scans all disks for physical volumes. Example:

```
# pvs
  PV          VG   Fmt   Attr Psize  PFree
  /dev/xvdd1       lvm2  a--  1.00g  1.00g
  /dev/xvdd2       lvm2  a--  1.00g  1.00g
# pvscan
  PV /dev/xvdd1      lvm2  [1.00GiB]
  PV /dev/xvdd2      lvm2  [1.00GiB]
  Total: 2 [2.00GiB] / in use: 0 [0   ] / in no VG: 2 [2.00 GiB]
```

Removing Physical Volumes

Use the pvremove command to remove a physical volume, for example:

```
# pvremove /dev/xvdd1
  Labels on physical volume "/dev/xvdd1" successfully wiped
# pvdisplay /dev/xvdd1
  No physical volume label read from /dev/xvdd1
  Failed to read physical volume "/dev/xvdd1"
```

Additional PV Commands

The following are other commands that are associated with the manipulation of physical volumes:

- **pvchange**: Change the attributes of physical volumes.
- **pvresize**: Resize physical volumes.
- **pvck**: Check the consistency of physical volumes.
- **pvmove**: Move extents from one physical volume to another.

Volume Group Utilities

- Use the `vgcreate` command to create volume groups:

```
# vgcreate -v myvolg /dev/xvdd1 /dev/xvdd2
```

- The following commands display volume groups:
  - `vgdisplay`
  - `vgs`
  - `vgscan`
- Use the `vgremove` command to remove volume groups:

```
# vgremove myvolg
```

- Additional VG commands are available, for example:
  - `vgchange`: Change volume group attributes.
  - `vgck`: Check the consistency of volume groups.
  - `vgextend`: Add physical volumes to a volume group.
  - `vgreduce`: Remove physical volumes from a volume group.

The next step in implementing LVM is to assign the physical volumes to an existing or new volume group.

Creating a Volume Group

Use the `vgcreate` command to create a new volume group. Space in a volume group is divided into "extents." The default physical extent size is 4 MB. The syntax is:

```
vgcreate [options] volume_group_name physical_volume
```

For example, to create a volume group named `myvolg` by using the `/dev/xvdd1` and `/dev/xvdd2` physical volumes with a default physical extent size of 4 MB, enter:

```
# vgcreate -v myvolg /dev/xvdd1 /dev/xvdd2
   Adding physical volume '/dev/xvdd1' to volume group 'myvolg'
   Adding physical volume '/dev/xvdd2' to volume group 'myvolg'
   Archiving volume group "myvolg" metadata (seqno 0).
   Creating volume group backup "/etc/lvm/backup/myvolg" ...
 Volume group "myvolg" successfully created
```

Displaying Volume Groups

Use the `vgdisplay` command to display attributes of volume groups:

```
# vgdisplay
  --- Volume group ---
  VG Name         myvolg
  System ID
  Format          lvm2
  ...
```

In addition to `vgdisplay`, two other commands list information about volume groups. The `vgs` command reports information about volume groups in a more condensed form. The `vgscan` command scans all disks for volume groups and rebuilds caches. Example:

```
# vgs
  VG      #PV  #LV  #SN  Attr    Vsize  VFree
  myvolg    2    0    0  wz--n-  5.01g  5.01g
# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "myvolg" using metadata type lvm2
```

Removing Volume Groups

Use the `vgremove` command to remove a volume group, for example:

```
# vgremove myvolg
  Volume group "myvolg" successfully removed
# vgdisplay
  No volume groups found
```

Additional VG Commands

The following commands are used to manipulate volume groups:

- **`vgcfgbackup`**: Back up volume group configurations.
- **`vgcfgrestore`**: Restore volume group configurations.
- **`vgchange`**: Change volume group attributes.
- **`vgck`**: Check the consistency of volume groups.
- **`vgconvert`**: Change the volume group metadata format.
- **`vgexport`**: Unregister volume groups from the system.
- **`vgextend`**: Add physical volumes to a volume group.
- **`vgimport`**: Register an exported volume group with the system.
- **`vgmerge`**: Merge volume groups.
- **`vgmknodes`**: Create special files for volume group devices in `/dev`.
- **`vgreduce`**: Remove physical volumes from a volume group.
- **`vgrename`**: Rename a volume group.
- **`vgsplit`**: Move physical volumes into a new or existing volume group.

The use of the **`vgcfgbackup`** and **`vgcfgrestore`** commands is discussed in a later slide.

Logical Volume Utilities

- Use the `lvcreate` command to create logical volumes:

```
# lvcreate -v --size 2g --name myvol myvolg
```

- The following commands display logical volumes:
  - `lvdisplay`
  - `lvs`
  - `lvscan`
- Use the `lvremove` command to remove logical volumes:

```
# lvremove myvolg/myvol
```

- Additional LV commands are available, for example:
  - `lvchange`: Change the attributes of logical volumes.
  - `lvextend`: Add space to a logical volume.
  - `lvreduce`: Reduce the size of a logical volume.

ORACLE

The next step in implementing LVM is to create logical volumes from the space allocated to volume groups.

Creating Logical Volumes

Use the `lvcreate` command to create a new logical volume. This command automatically creates the block device nodes in the `/dev` directory. The syntax is:

```
lvcreate [options] --size <size> --name LV_name VG_name
```

The `--size` option defines the size of the logical volume by allocating logical extents from the free physical extent pool of the volume group. For example, to create a logical volume named `myvol` from the volume group named `myvolg` with a size of 2 GB, enter:

```
# lvcreate -v --size 2g --name myvol myvolg
    Setting logging type to disk
    Finding volume group "myvolg"
    Archiving volume group "myvolg" metadata (seqno 1).
    Creating logical volume myvol
    Create volume group backup "/etc/lvm/backup/myvolg" ...
...
```

Displaying Logical Volumes

Use the `lvdisplay` command to display the attributes of logical volumes.

```
# lvdisplay
  --- Logical volume ---
  LV Path           /dev/myvolg/myvol
  LV Name           myvol
  VG Name           myvolg
  LV UUID...
...
```

In addition to `lvdisplay`, two other commands list information about logical volumes. The `lvs` command reports information about logical volumes in a more condensed form. The `lvscan` command scans all disks for logical volumes. Example:

```
# lvs
  LV    VG     Attr       LSize  Pool Origin Data% Move Log Cpy...
  myvol myvolg -wi-a----- 2.00g
# lvscan
  ACTIVE    '/dev/myvolg/myvol' [2.00 GiB] inherit
```

Removing Logical Volumes

Use the `lvremove` command to remove a logical volume. You must include the volume group name as well as the logical volume name. You are prompted to confirm your request. Example:

```
# lvremove myvol
  Volume group "myvol" not found
  Skipping volume group myvol
# lvremove myvolg/myvol
  Do you really want to remove active logical volume myvol? ...
  Logical volume "myvol" successfully removed
```

Additional LV Commands

The following commands are used to manipulate logical volumes:

- **`lvchange`**: Change the attributes of logical volumes.
- **`lvconvert`**: Change logical volume layout.
- **`lvextend`**: Add space to a logical volume.
- **`lvmdiskscan`**: List devices that may be used as physical volumes.
- **`lvmsadc`**: Collect activity data.
- **`lvmsar`**: Create activity report.
- **`lvreduce`**: Reduce the size of a logical volume.
- **`lvrename`**: Rename a logical volume.
- **`lvresize`**: Resize a logical volume.

Making Logical Volumes Usable

- Final steps:
  - Create a file system on the logical volume.
  - Create a mount point.
  - Attach the logical volume to the directory hierarchy.
- The `lvcreate` command creates two entries in the `/dev` directory for each logical volume. Example:
  - `/dev/mapper/myvolg-myvol`
  - `/dev/myvolg/myvol`
- Either of these block device names are usable as arguments to the `mkfs` command:

```
# mkfs -t ext4 /dev/mapper/myvolg-myvol
# mkfs -t ext4 /dev/myvolg/myvol
```


The last step in implementing LVM is to create a file system on the logical volume, create a mount point, and attach the logical volume to the directory hierarchy. There is nothing new here, these steps were discussed in the lesson titled "Partitions, File Systems, and Swap." Logical volumes do not require a file system to be usable. For example, they can be used as Automatic Storage Management (ASM) disks or as a raw device.

The only thing different from creating a file system on a disk partition, and creating a file system on a logical volume, is the name of the block device in the `/dev` directory. The `lvcreate` command creates two entries in the `/dev` directory for each logical volume. For example, when creating the logical volume named `myvol` from the volume group named `myvolg`, the following two block device names in the `/dev` directory were automatically created:

```
/dev/mapper/myvolg-myvol
/dev/myvolg/myvol
```

Use either of these device names as arguments to the `mkfs` command when making a file system. For example, to make an ext4 file system on the `myvol` logical volume, enter either of the following commands:

```
# mkfs -t ext4 /dev/mapper/myvolg-myvol
# mkfs -t ext4 /dev/myvolg/myvol
```

The blkid command displays the same output (and the same UUIDs) when querying either of the logical volume device names:

```
# blkid /dev/mapper/myvolg-myvol
/dev/mapper/myvolg-myvol: UUID="9fa64e..." TYPE="ext4"
# blkid /dev/myvolg/myvol
/dev/myvolg/myvol: UUID="9fa64e..." TYPE="ext4"
```

Create a mount point and mount the new logical volume file system, for example:

```
# mkdir /test
# mount /dev/mapper/myvolg-myvol /test
```

Create an entry in /etc/fstab to mount the file system at boot time.

Backing Up and Restoring Volume Group Metadata

- LVM metadata contains configuration details of volume groups.
- Metadata backups and archives are automatically created on every volume group and logical volume configuration change.
  - Backups are stored in `/etc/lvm/backup`.
  - Archives are stored in `/etc/lvm/archive`.
- Configuration settings are stored in `/etc/lvm/lvm.conf`.
- Use the `vgcfgbackup` command to manually back up LVM metadata.
- Use the `vgcfgrestore` command to restore from a backup to recover from corrupted or missing metadata.

LVM metadata contains configuration details of LVM volume groups. By default, metadata backups and archives are automatically created on every volume group and logical volume configuration change. Settings can be changed in the LVM configuration file, `/etc/lvm/lvm.conf`. The `lvm dumpconfig` command displays configuration settings.

Metadata backups are stored in the `/etc/lvm/backup` directory. Metadata archives are stored in the `/etc/lvm/archive` directory. You can manually back up the metadata by using the `vgcfgbackup` command. For example, the following command backs up the metadata of the `myvolg` volume group to the `/etc/lvm/backup/myvolg` file:

```
# vgcfgback myvolg
```

Omit the name of the volume group to back up metadata for all volume groups. Use the `-f <filename>` option to give the backup file a specific file name.

The following are examples of error messages you might get if the metadata area is corrupted or incorrect:

```
Couldn't find device with uuid '...'.
Couldn't find all physical volumes for volume group myvolg.
```

You can use the `vgcfgrestore` command to restore volume group metadata from a backup. Provide the name of the volume group as an argument to the `vgcfgrestore` command.

LVM Thin Provisioning

- LVM thin provisioning allows you to over-commit the physical storage.
- You can create file systems which are larger than the available physical storage.
- Use the `lvcreate` command to create a thin pool:

```
# lvcreate -L 100m -T myvolg/mythinpool
```

- Use the `lvcreate` command to create a thin volume.
  - A thin volume is a virtual disk inside a thin pool.
  - The size of the virtual disk can be greater than the size of the thin pool.

```
# lvcreate -V 1g -T myvolg/mythinpool -n mythinvol
```

- Use the `lvs` command to monitor the allocated pool data and add more capacity when it starts to become full.

LVM thin provisioning allows you to create virtual disks inside a thin pool. The size of the virtual disk can be greater than the available space in the thin pool. This allows you to over-commit the physical storage and create file systems which are larger than the actual available physical storage. It is important that you monitor the thin pool and add more capacity when it starts to become full.

Thin pools are created using the `lvcreate` command and as such, they are essentially logical volumes. Use either the `-T` option, or the `--thin` option, or the `--thinpool` option when creating a thin pool. The following example creates a thin pool named `mythinpool` from the `myvolg` volume group that is 100m in size:

```
# lvcreate -L 100m -T myvolg/mythinpool
```

This command creates a logical volume as shown by the following command:

```
# lvs
  LV         VG     Attr       LSize    Pool     Origin Data%...
  mythinpool myvolg twi-a-tz-- 100.00m                   0.00
```

The "Data%" column shows the allocated pool data. The example shows 0.00% because virtual thin volumes have not yet been created in this thin-pool. You can also use the `lvdisplay` command to show the "Allocated pool data" percentage.

The thin pool logical volume is not mountable. That is, there is no entry in the `/dev` directory:

```
# ls /dev/myvolg*
ls: cannot access /dev/myvolg*: No such file or directory
```

Use the `lvcreate` command with the `-V` option to create a thin volume (a virtual disk) from a thin pool. The following example creates a 1 GB thin volume named `mythinvol` in the `myvolg/mythinpool` thin pool. Note that the size of the thin volume is larger than the size of the thin pool that contains it.

```
# lvcreate -V 1g -T myvolg/mythinpool -n mythinvol
```

This command creates a thin volume as shown by the following command:

```
# lvs
  LV         VG     Attr       LSize   Pool       Origin Data%...
  mythinpool myvolg twi-a-tz-- 100.00m                     0.00
  mythinvol  myvolg Vwi-a-tz--   1.00g mythinpool          0.00
```

Note the difference in attributes. The thin volume has a 'V' attribute for virtual disk. The Data% column shows 0.00 until you create a file system on the thin volume.

The virtual disk thin volume has a `/dev` entry shown as follows. In this example, the entry is a symbolic link to the `dm-4` block device.

```
# ls -l /dev/myvolg*
lrwxrwxrwx ... mythinvol -> ../dm-4
```

You can create a file system on this thin volume and mount it. For example:

```
# mkfs.ext4 /dev/myvolg/mythinvol
# mkdir /myvol
# mount /dev/myvolg/mythinvol /myvol
```

Output of the `df` command shows the size of the file system is 976M, which is an over-allocation of the available storage in the thin pool.

```
# df -h
Filesystem    Size  Used  Avail  Use%  Mounted on
...
/dev/mapper/myvolg-mythinvol
              976M  2.6M   907M    1%  /myvol
```

Copy some data to `/myvol` then run the `lvs` command to show the allocated pool data.

```
# cp /boot/vmlinuz* /myvol
# lvs
  LV         VG     Attr       LSize   Pool       Origin Data%...
  mythinpool myvolg twi-a-tz-- 100.00m                    49.00
  mythinvol  myvolg Vwi-a-tz--   1.00g mythinpool          4.79
```

This shows you have used 49% of the allocated pool data. This also shows that the thin volume has used 4.79% of 1 GB. You can use the `lvextend` command to add space to a thin pool logical volume.

Snapper

- Command-line utility in Oracle Linux 7 to create and manage snapshots
- Supports Btrfs and LVM thin volumes
- Requires a configuration file for each Btrfs and LVM volume
  - To create `myvol1_snap` configuration file for ext4 file system on LVM thin volume mounted on `/myvol1`:

```
# snapper -c myvol1_snap create-config -f "lvm(ext4)"
   /myvol1
```

  - Entry is added to `/etc/sysconfig/snapper`.
  - `.snapshots` directory is created in the `/myvol1` directory.
  - Configuration file, `myvol1_snap`, is created in the `/etc/snapper/configs/` directory.

ORACLE

Snapper is a command-line utility in Oracle Linux 7 used to create and manage snapshots of LVM thin volumes. It can create, delete, and compare snapshots and revert changes done between snapshots. Snapper also allows for the easy creation and management of snapshots for Btrfs. Use the following command to install the snapper software package:

```
# yum install snapper
```

The snapper software package includes a `cron.hourly` file to create snapshots and a `cron.daily` file to clean up old snapshots.

```
# ls -l /etc/cron*/snapper
-rwxr-xr-x ... /etc/cron.daily/snapper
-rwxr-xr-x ... /etc/cron.hourly/snapper
```

To create a snapshot using snapper, a configuration file is required for the LVM thin volume or Btrfs subvolume. The LVM and Btrfs volumes must also have a mounted file system. Use the `create-config` command to create the configuration file. The following example creates a configuration file named `myvol_snap` for an LVM ext4 file system mounted on `/myvol1`:

```
# snapper -c myvol1_snap create-config -f "lvm(ext4)" /myvol1
```

This command adds an entry to `/etc/sysconfig/snapper`, creates a `.snapshots` directory in the `/myvol1` directory, and creates the configuration file, `myvol1_snap`, in the `/etc/snapper/configs/` directory.

Snapper

- There are three types of snapshots that you can create:
  - pre, post, single
  - Always associate a pre snapshot with a post snapshot
  - All snapshots have an associated number.
- To create a pre snapshot:

```
# snapper -c myvol1_snap create -t pre -p
```

- To create a post snapshot:

```
# snapper -c myvol1_snap create -t post -pre-num 4 -p
```

- To list the differences in a pre snapshot (#4) and a post snapshot (#5):

```
# snapper -c myvol1_snap diff 4..5
```


Each /etc/snapper/configs/* file describes a snapper configuration. See the snapper-configs(5) man page for a description of the parameters in the snapper configuration file.

The hourly cron job performs automatic snapshot creation but you can also create snapshots manually. There are three types of snapshots that you can create by using snapper:

- **pre**: Use to record the state of a volume before a modification. Pre snapshots should always have a corresponding post snapshot.
- **post**: Use to record the state of a volume after a modification.
- **single**: These snapshots have no special relationship to other snapshots.

Use the create -t command to specify the type of snapshot to create. Possible values are single, pre, and post.

The following example creates a pre snapshot of the /myvol1 volume. The -p option causes snapper to display the number of the snapshot. In this example, the number is 4.

```
# snapper -c myvol1_snap create -t pre -p
4
```

The snapshots are stored by snapshot number in the .snapshots subdirectory of the volume.

The following example creates a post snapshot of the `/myvol1` volume. The `--pre-num 4` option references the associated pre snapshot number. The `-p` option causes snapper to display the number of the snapshot. In this example, the number is `5`.

```
# snapper -c myvol1_snap create -t post -pre-num 4 -p
5
```

Use the snapper `status` command to display the files and directories that have been added, removed, or modified between a pre snapshot and a post snapshot. Example:

```
# snapper -c myvol1_snap status 4..5
```

Use the snapper `diff` command to display the differences between the contents of the files in a pre snapshot and a post snapshot. Example:

```
# snapper -c myvol1_snap diff 4..5
```

Use the snapper `list` command to list the snapshots that exist for a volume defined by the snapper configuration file. Example:

```
# snapper -c myvol1_snap list
```

Use the snapper `delete` command to delete a snapshot number. Example:

```
# snapper -c myvol1_snap delete 1
```

Use the snapper `undochange` command to revert the contents of a volume defined by a configuration file to the pre snapshot contents. Example:

```
# snapper -c myvol1_snap undochange 4..5
```

Redundant Array of Independent Disks (RAID)

- The multi-disk (MD) driver supports software RAID.
  - MD organizes disk drives into RAID devices (arrays).
- Common RAID levels are supported:
  - Linear RAID: Concatenated drives
  - RAID-0: Striping
  - RAID-1: Mirroring
  - RAID-5: Distributed parity
  - RAID-6: Dual-distributed parity
  - Nested RAID levels:
    - RAID 0+1: Mirrored striping
    - RAID 1+0 (or RAID 10): Striped mirror

In addition to logical volume management with LVM2, the Linux kernel supports "software RAID" with the multi-disk (MD) driver. MD organizes disk drives into RAID devices, or arrays, and provides different RAID levels. RAID devices are virtual devices created from two or more real block devices.

RAID combines multiple disk drives into an array and allows data to be spread across the drives to increase capacity, achieve redundancy, and increase performance.

Supported RAID Levels

The following RAID levels are the most commonly used levels supported by Oracle Linux:

- **Linear RAID:** Linear RAID simply groups drives together to create a larger virtual drive. Data is written to the first drive until it is full, and then it is written to the next drive. There is no redundancy or performance benefit. Reliability is actually decreased, because the entire array cannot be used if any one drive fails.
- **RAID-0:** RAID-0 is called striping and provides an increase in performance but offers no redundancy. Data is broken down into stripes and written across all the drives, rather than filling up the first drive before moving on to the next as is the case with Linear RAID. In addition, as is the case with Linear RAID, the entire array cannot be used if any one drive fails.

- **RAID-1:** RAID-1 is called mirroring and provides redundancy by writing identical data to each drive in the array. If one drive fails, the mirror drive satisfies I/O requests. RAID-1 is expensive because the same information is written to all of the disks in the array.
- **RAID-5:** RAID-5 is the most common type of RAID and uses striping with distributed parity. RAID-5 is able to recover from the loss of one drive in the array. Parity information is calculated based on the contents of the rest of the drives in the array. This information is used to reconstruct data when one drive in the array fails. The reconstructed data also satisfies I/O requests to the failed drive before it is replaced, and repopulates the failed disk after it has been replaced. With RAID-5, the parity is distributed across all drives in the array.
- **RAID-6:** RAID-6 uses striping with double distributed parity. RAID-6 is able to recover from the loss of two drives in the array. RAID-6 is commonly used when data redundancy and preservation, and not performance, is of most importance.

Nested RAID Levels

Nested RAID levels, also known as hybrid RAID, combine standard RAID levels for additional performance and/or redundancy. One example is **RAID 0+1**, which is a mirror (RAID-1) of striped (RAID-0) disks. Another example is **RAID 1+0**, sometimes called **RAID 10**, which is a stripe of mirrors.

Many Oracle Database customers use one of these nested RAID levels, but have the RAID implemented in the storage area network (SAN) arrays. RAID 5 and RAID 6 provide the redundancy needed, but add overhead to calculate parity. This can impact the performance of write-intensive databases.

`mdadm` Utility

- Use the `mdadm` command to build, manage, and monitor Linux MD devices (software RAID devices).
- To create a device:

```
# mdadm --create /dev/md0 --level=1 --raid-devices=2
   /dev/xvdd1 /dev/xvdd2
```

- To query a device:

```
# mdadm --query /dev/md0
# mdadm --detail /dev/md0
# cat /proc/mdstat
```


Creating a RAID Device

The `mdadm` command is used to build, manage, and monitor Linux MD devices (software RAID devices). The basic syntax to create a new RAID array is:

```
mdadm --create <md_device> --level=<RAID_level> --raid-devices=<#> devices
```

For example, to create a RAID-1 device (`/dev/md0`) consisting of two block devices (`/dev/xvdd1` and `/dev/xvdd2`), enter:

```
# mdadm --create /dev/md0 --level=1 –raid-devices=2 /dev/xvdd1
/dev/xvdd2
```

Information about the RAID device to be created is displayed along with the following prompt:

```
Continue creating array?
```

Respond with “y” to create the array. View the `/proc/mdstat` file to check the status of your MD RAID devices:

```
# cat /proc/mdstat
Personalities : [raid1]
mdo : active raid1 xvdd2[1] xvdd1[0]
```

Querying a RAID Device

You can also use the `mdadm` command to view information about the RAID device:

```
# mdadm --query /dev/md0
/dev/md0: 2.01GiB raid1 2 devices, 0 spares.
```

To see even more detail, enter:

```
# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : ...
     Raid Level : raid1
     Array Size : 1048000 (023.61 MiB 1073.15 MB)
  Used Dev Size : 1048000 (023.61 MiB 1073.15 MB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent
    Update Time : ...
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
...
```

Making RAID Devices Usable

1. Create a file system on the RAID device.
2. Create a mount point.
3. Attach the RAID device to the directory hierarchy.
4. Update the `mdadm` configuration file:
   - `/etc/mdadm.conf`
   - `ARRAY /dev/md0 devices=/dev/xvdd1,/dev/xvdd2`

ORACLE

The last step in implementing a RAID device is to create a file system on the device, create a mount point, and attach the device to the directory hierarchy. Again, this is nothing new. These steps are necessary for standard partitions, logical volumes, and RAID devices. You can create logical volumes on top of RAID block devices.

Assuming that the RAID device name is `/dev/md0` and that you want to create an ext4 file system on the device and mount it to `/raid`, enter:

```
# mkfs -t ext4 /dev/md0
# mkdir /raid
# mount /dev/md0 /raid
```

The last step is to update the `mdadm` configuration file, `/etc/mdadm.conf`. It is useful to store the RAID configuration information in this file. This helps `mdadm` to assemble existing arrays at system boot. You can either copy and adapt the sample configuration file from `/usr/share/doc/mdadm-3.2.1/mdadm.conf-example`, or create the file from scratch. The following entry would suffice for the RAID device created in the practices for this lesson:

```
ARRAY /dev/md0 devices=/dev/xvdd1,/dev/xvdd2
```

Quiz

Which of the following commands is used to build, manage, and monitor software RAID devices?

a. `raidadm`

b. `lvadm`

c. `mdadm`

d. `dmsetup`

Summary

In this lesson, you should have learned how to:

- Describe the Linux device mapper
- Describe Logical Volume Manager (LVM)
- Configure LVM components
- Back up and restore volume group metadata
- Describe LVM thin provisioning
- Describe snapper
- Describe Linux kernel multi-disk (MD) driver
- Describe RAID and configure RAID devices

Practice 14: Overview

The practices for this lesson cover the following:

- Creating Linux LVM partitions
- Creating a logical volume
- Creating a file system and mounting a logical volume
- Backing up volume group metadata
- Creating a logical volume snapshot
- Increasing the capacity of a logical volume
- Restoring volume group metadata
- Creating a thinly provisioned logical volume
- Using snapper with LVM thin provisioned logical volumes
- Creating a RAID device

15

Network Configuration

Objectives

After completing this lesson, you should be able to:

- Describe network interface configuration files
- Describe additional network configuration files
- Start the network service
- Use the `ethtool` utility
- Describe NetworkManager
- Use the NetworkManager GUI
- Use the Network Connections Editor
- Use the `nmcli` utility
- Use the `nmtui` utility
- Describe ARP and the ARP cache
- Use the `ip` utility

Network Interface File Names

- Each physical network device has an associated network interface configuration file.
- Network interface configuration files are located in the `/etc/sysconfig/network-scripts` directory.
- Naming scheme automatically assigns interface names that are predictable. For example:

```
# ls/etc/sysconfig/network-scripts/ifcfg*
ifcfg-enp134s1f0   ifcfg-enp134s1f1   ifcfg-lo
```

  – Names persist across system reboots, hardware reconfiguration, and kernel and device driver updates.
- Naming scheme can be changed by boot parameters:
  – `biosdevname=1`
  – `net.ifnames=0`

Linux handles network communications through software configuration files and the physical networking devices in your system. Each physical network device has an associated configuration file, named `ifcfg-<interface>`, located in the `/etc/sysconfig/network-scripts` directory.

In the following example, there are two Ethernet interfaces, represented by `ifcfg-enp134s1f0` and `ifcfg-enp134s1f1`, and one loopback interface (`ifcfg-lo`). The system uses these files during the boot process to configure the network interfaces.

```
# ls /etc/sysconfig/network-scripts/ifcfg*
ifcfg-enp134s1f0   ifcfg-enp134s1f1   ifcfg-lo
```

In Oracle Linux 7, `systemd` and udev support different network interface name schemes. The default is to assign fixed names based on firmware, topology, and location. In the example, the network interfaces are on a PCI card and are named "`en`" for Ethernet followed by `p<bus>s<slot>`, and `f<function_number>`. The following shows the names in `/sys`:

```
# ls -lR /sys | grep 134
lrwxrwxrwx... enp134s1f0 ->
/sys/devices/pci0000:80/0000:80:11.0/0000:86:01.0/net/enp134s1f0
lrwxrwxrwx... enp134s1f1 ->
/sys/devices/pci0000:80/0000:80:11.0/0000:86:01.1/net/enp134s1f1
```

This naming scheme automatically assigns interface names that are predictable and ensures that the names persist across system reboots, hardware reconfiguration, and updates to device drivers and the kernel.

By default, `systemd` assigns a two character prefix based on the type of interface:

- **`en`**: Ethernet
- **`wl`**: Wireless LAN (WLAN)
- **`ww`**: Wireless wide area network (WWAN)

The prefix is followed by a suffix based on the hardware configuration, system bus configuration, or MAC address of the device, described as follows:

- **`o`*`N`***: Onboard device with index number *`N`*. For example, `eno1`.
- **`s`*`S`*`[f`*`F`*`] [d`*`D`*`]`**: Hot-plug device with slot number *`S`*, optional function number *`F`*, and optional device ID *`D`*
- **`x`*`M`***: Device with MAC address *`M`*
- **`p`*`B`*`s`*`S`*`[f`*`F`*`] [d`*`D`*`]`**: PCI device with bus number *`B`*, slot number *`S`*, optional function number *`F`*, and optional device ID *`D`*. For example:
  - `enp134s1f0`: Ethernet, bus number 134, slot number 1, function number 0
  - `enp134s1f1`: Ethernet, bus number 134, slot number 1, function number 1
- **`p`*`B`*`s`*`S`*`[f`*`F`*`] [u`*`P`*`] [c`*`C`*`] [i`*`I`*`]`**: USB device with bus number *`B`*, slot number *`S`*, optional function number *`F`*, optional port number *`P`*, optional configuration number *`C`*, and optional interface number *`I`*

The kernel assigns a legacy, unpredictable network interface name (`eth`*`N`* and `wlan`*`N`*) only if it cannot discover any information about the device that would allow it to disambiguate the device from other such devices. You can use the `net.ifnames=0` boot parameter to reinstate the legacy naming scheme. The following gives the network interface names in `/sys` when using the `net.ifnames=0` boot parameter:

```
# ls -lR /sys | grep eth
lrwxrwxrwx... eth0 ->
/sys/devices/pci0000:80/0000:80:11.0/0000:86:01.0/net/eth0
lrwxrwxrwx... eth1 ->
/sys/devices/pci0000:80/0000:80:11.0/0000:86:01.1/net/eth1
```

The name of embedded network interfaces, PCI card network interfaces, and virtual function network interfaces can also be changed by the `biosdevname` udev helper utility. This feature requires that you install the `biosdevname` software package, and that you enable the `biosdevname` boot option as follows:

```
# yum install biosdevname
biosdevname=1
```

Note that using the `net.ifnames` or `biosdevname` boot parameters to change the naming scheme can render existing firewall rules invalid. This is discussed further in the lesson titled "Security Administration." Changing the naming scheme can also affect other software that refers to legacy network interface names.

Network Interface File Parameters

- Configuration parameters include:
  - `TYPE=Ethernet`
  - `BOOTPROTO=none`
  - `DEFROUTE=yes`
  - `NAME=eth0`
  - `ONBOOT=yes`
  - `HWADDR=00:14:4F:8D:B0:BC`
  - `IPADDR0=10.150.36.203`
  - `PREFIX0=23`
  - `GATEWAY0=10.150.36.1`
  - `DNS1=152.68.154.3`
  - `DNS2=10.216.106.3`
  - `DOMAIN=us.oracle.com`

The system reads the network interface files during the boot process to determine which interfaces to bring up and how to configure them. The following is a sample:

```
# cat /etc/sysconfig/network-scripts/ifcfg-enp134s1f0
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=enp134s1f0
UUID=...
ONBOOT=yes
HWADDR=00:14:4F:8D:B0:BC
```

```
IPADDR0=10.150.36.203
PREFIX0=23
GATEWAY0=10.150.36.1
DNS1=152.68.154.3
DNS2=10.216.106.3
DOMAIN=us.oracle.com
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
```

A description of some of these configuration parameters follows:

TYPE=*device_type*: The type of network interface device

BOOTPROTO=*protocol*: Where *protocol* is one of the following:

- **none**: No boot-time protocol is used.
- **bootp**: Use BOOTP (bootstrap protocol).
- **dhcp**: Use DHCP (Dynamic Host Configuration Protocol).

DEFROUTE|IPV6_DEFROUTE=*answer*: Where *answer* is one of the following:

- **yes**: This interface is set as the default route for IPv4|IPv6 traffic.
- **no**: This interface is not set as the default route.

IPV6INIT=*answer*: Where *answer* is one of the following:

- **yes**: Enable IPv6 on this interface. If **IPV6INIT=yes**, the following parameters could also be set in this file:
  - **IPV6ADDR**=IPv6 address
  - **IPV6_DEFAULTGW**=The default route through the specified gateway
- **no**: Disable IPv6 on this interface.

IPV4_FAILURE_FATAL|IPV6_FAILURE_FATAL=*answer*: Where *answer* is one of the following:

- **yes**: This interface is disabled if IPv4 or IPv6 configuration fails.
- **no**: This interface is not disabled if configuration fails.

ONBOOT=*answer*: Where *answer* is one of the following:

- **yes**: This interface is activated at boot time.
- **no**: This interface is not activated at boot time.

HWADDR=*MAC-address*: The hardware address of the Ethernet device

IPADDR*N*=*address*: The IPv4 address assigned to the interface

PREFIX*N*=*N*: Length of the IPv4 netmask value

GATEWAY*N*=*address*: The IPv4 gateway address assigned to the interface. Because an interface can be associated with several combinations of IP address, network mask prefix length, and gateway address, these are numbered starting from `0`.

DNS*N*=*address*: The address of the Domain Name Servers (DNS)

DOMAIN=*DNS_search_domain*: The DNS search domain

Refer to http://docs.oracle.com/cd/E37670_01/E41138/html/ol_about_netconf.html for a description of network interface configuration file parameters.

Additional Network Configuration Files

- `/etc/hosts` associates host names with IP addresses.
  - Larger networks would use DNS to perform this resolution.
  - Specify the IP address of the loopback device.
- `/etc/resolv.conf`:
  - Provides access to DNS
  - Identifies DNS name server(s) and search domain
- `/etc/sysconfig/network` specifies global information for all network interfaces.
- `/etc/nsswitch.conf` lists the order of host name searches.

In addition to the individual network interface configuration files in the `/etc/sysconfig/network-scripts` directory, there are other, more global network configuration files. These files are:

- **`/etc/hosts`**
- **`/etc/resolv.conf`**
- **`/etc/sysconfig/network`**
- **`/etc/nsswitch.conf`**

`/etc/hosts`

This file associates host names with IP addresses. It resolves, or looks up, an IP address when the host name is known. Larger networks would use Domain Name Service (DNS) to perform this resolution. Even if using DNS, include in this file a line specifying the IP address of the loopback device (127.0.0.1) as `localhost.localdomain`. A sample `/etc/hosts` file follows. The first column contains the IP address. The second column is the fully qualified host names. Additional columns contain host name aliases:

```
# cat /etc/hosts
127.0.0.1        localhost.localdomain       localhost
192.0.2.101      host01.example.com          host01
```

/etc/resolv.conf

The resolver configuration file provides access to DNS. This file usually has at least two lines, one line specifying the IP address of a DNS server (or name server) and the other specifying the search domain. The following example shows three name servers and the search domain:

```
# cat /etc/resolv.conf
search us.oracle.com
nameserver 152.68.154.3
nameserver 10.216.106.3
nameserver 193.32.3.252
```

/etc/sysconfig/network

This file specifies global network settings. For example, you can specify the default gateway in this file:

```
# cat /etc/sysconfig/network
GATEWAY=192.0.2.1
```

/etc/nsswitch.conf

This file is the system databases and name service switch configuration file. It provides sources for common configuration databases and name resolution mechanisms. Entries in this file identify the database name in the first field, then a colon, and then a list of possible resolution mechanisms in the second field. The order in which the mechanisms are listed determines the order in which queries on the specified database are resolved.

The following example indicates that host name resolution is attempted first by querying local files, that is, `/etc/hosts`, and then by querying the DNS server if the host name is not resolved:

```
# cat /etc/nsswitch.conf
...
hosts:    files dns
...
```

Starting the Network Service

- Use the `systemctl` command to start, stop, and restart the network service:

```
# systemctl restart network
```

- Interface control scripts in `/etc/sysconfig/network-scripts` can also be used.
- Use `ifup <interface_name>` to activate an interface:

```
# ifup enp134s1f0
```

- Use `ifdown <interface_name>` to deactivate an interface:

```
# ifdown enp134s1f0
```


The `systemd` service unit for networking is named `network.service` shown as follows:

```
# systemctl list-units -type service | grep network
network.service loaded active exited LSB: Bring up/down network
```

Use the `systemctl` command and specify `network` to start, stop, or view the status of network interfaces. The following example starts and stops all network interfaces:

```
# systemctl restart network
```

You can also use interface control scripts, located in the `/etc/sysconfig/network-scripts` directory, to start and stop network interfaces. The `ifup` and `ifdown` interface scripts are symbolic links to scripts in the `/usr/sbin` directory. When either of these scripts are called, they require the interface to be specified as an argument. For example, to bring the `enp134s1f0` interface down:

```
# ifdown enp134s1f0
```

To bring the `enp134s1f0` interface up:

```
# ifup enp134s1f0
```

There are additional interface control scripts in the `/etc/sysconfig/network-scripts` directory to start and stop different types of interfaces such as PPP, ISDN, PPP, and IPv6.

The `ethtool` Utility

- `ethtool` is used to query and set low-level network interface properties.
- Changes made by `ethtool` do not persist after a reboot.
- To show current low-level properties on an interface:

```
# ethtool enp134s1f0
...
```

- Use the `-s` option to set low-level properties on an interface. Example:

```
# ethtool -s enp134s1f0 speed 1000 autoneg on duplex
  full
```


The `ethtool` command allows you to query and set properties of the network device. This is useful for diagnosing possible mismatched settings that affect performance. The settings that `ethtool` controls are considered low-level or device settings.

The changes that `ethtool` makes are not permanent and do not persist through a reboot. To make the changes permanent, change the `/etc/sysconfig/network-scripts/ifcfg-<interface>` file for the device.

`ethtool` can be used to configure options such as speed, full or half duplex, autonegotiate, and other properties. To display a list of available options, use the `-h` option:

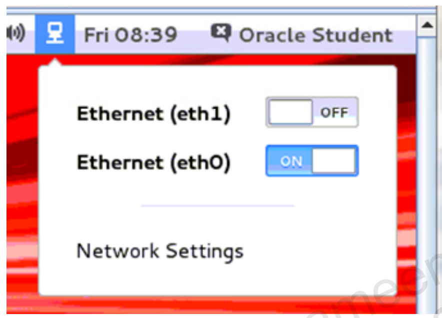
```
# ethtool -h
        ethtool -s|--change DEVNAME    Change generic options
                [ speed %d ]
                [ duplex half|full]
...
```

The following example configures the `enp134s1f0` interface to 1000 Mb/sec, full duplex, and enables autonegotiate:

```
# ethtool -s enp134s1f0 speed 1000 autoneg on duplex full
```

NetworkManager

- NetworkManager:
  - Dynamically detects and configures network connections
  - Includes a GNOME Notification Area network icon
- Click the icon to display status and to manage network connections.



NetworkManager is the default networking service in Oracle Linux 7. It dynamically detects and configures network connections and also attempts to keep network interfaces up and active. Use the following commands to ensure that the package is installed and that the service is started:

```
# yum install NetworkManager
# systemctl start NetworkManager
```

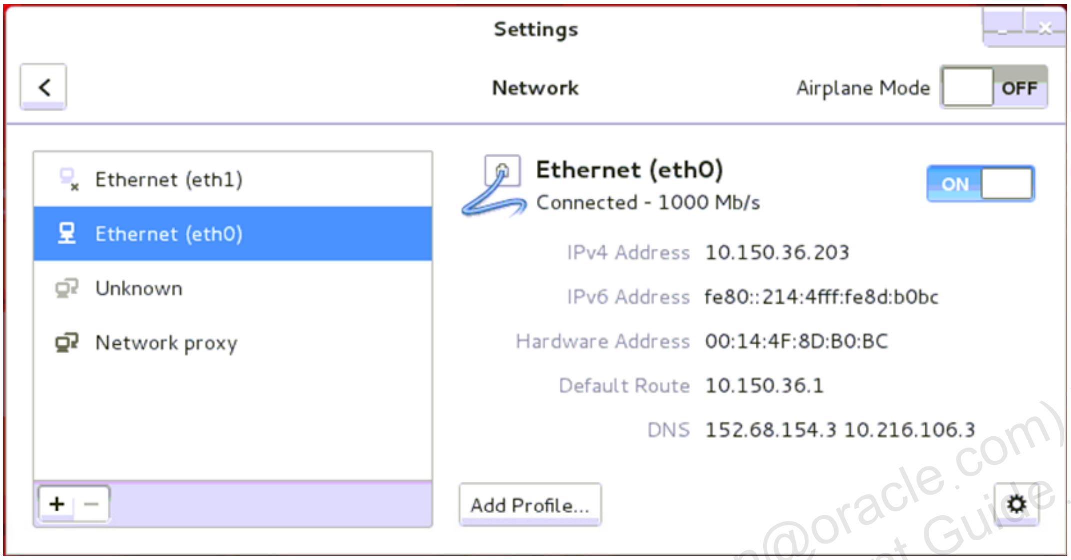
Run the following command to ensure that NetworkManager starts at boot time:

```
# systemctl enable NetworkManager
```

The GNOME Shell provides a network icon in the Notification Area, which represents network connection states as reported by NetworkManager. Click the network icon in the Notification Area to view the status of the network interfaces and also to manage network connections. A sample status screen is shown in the slide. In this example, there are two networking interfaces, `eth0` and `eth1`. The `eth0` interface is enabled (`ON`) and `eth1` is disabled (`OFF`). On this screen you can:

- Click a network interface entry to enable or disable the specific network interface. The ON/OFF toggle switch changes each time you click an entry.
- Click the Network Settings entry to display the Network Settings window.

Network Settings Editor



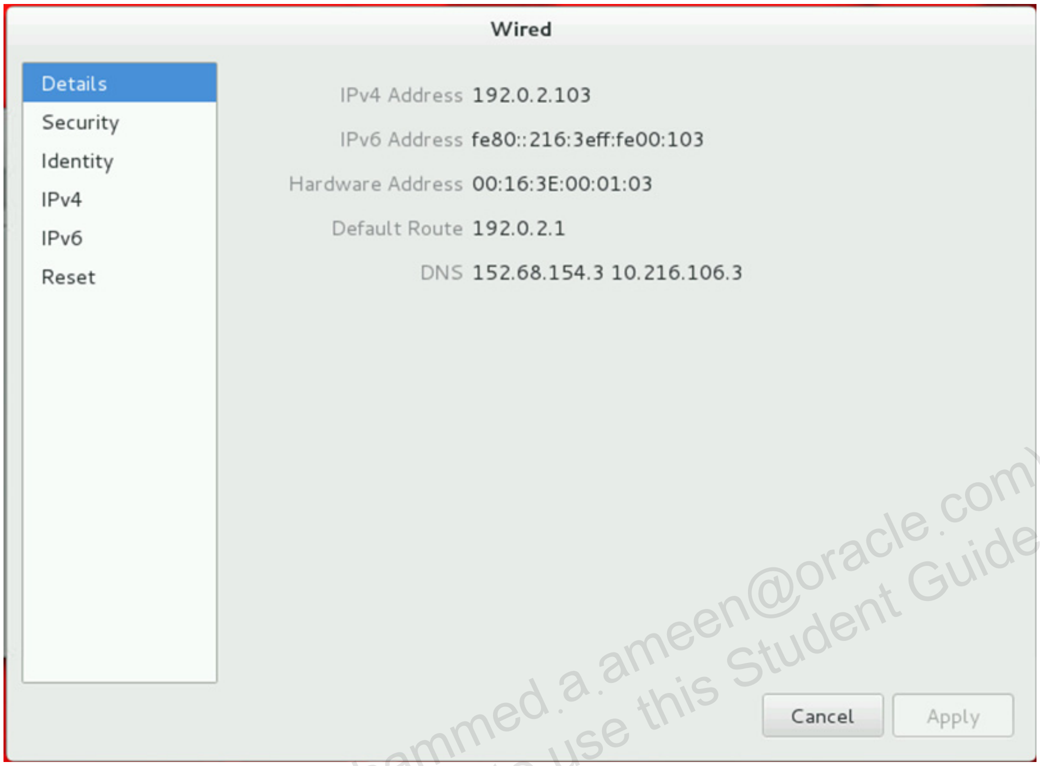
The Network Settings editor window is shown in the slide. Each of the existing network devices is listed in the left-hand pane. Details are displayed for the selected entry on the right side of the window. You can toggle the **ON/OFF** switch to enable or disable the selected entry. You can also toggle the **ON/OFF** switch to enable or disable airplane mode.

Click the plus sign (+) to add a new connection type. Click the minus sign (-) to delete the selected entry. You can create any of the following connection types. These connection types are covered in another course.

- **VPN:** Virtual Private Network
- **Bond:** Combine multiple network connections into a single logical interface
- **Team:** A new implementation of the bonding concept offered in Oracle Linux 7
- **Bridge:** A link-layer device that forwards traffic between networks based on MAC addresses
- **VLAN:** Virtual Local Area Network

You also have the option to **Add Profile**. A profile is a named collection of network settings that can be applied to a network interface. You can define more than one profile for an interface and apply each profile as needed. When you add a profile, NetworkManager creates a new configuration file and then opens the same window used for editing an existing connection.

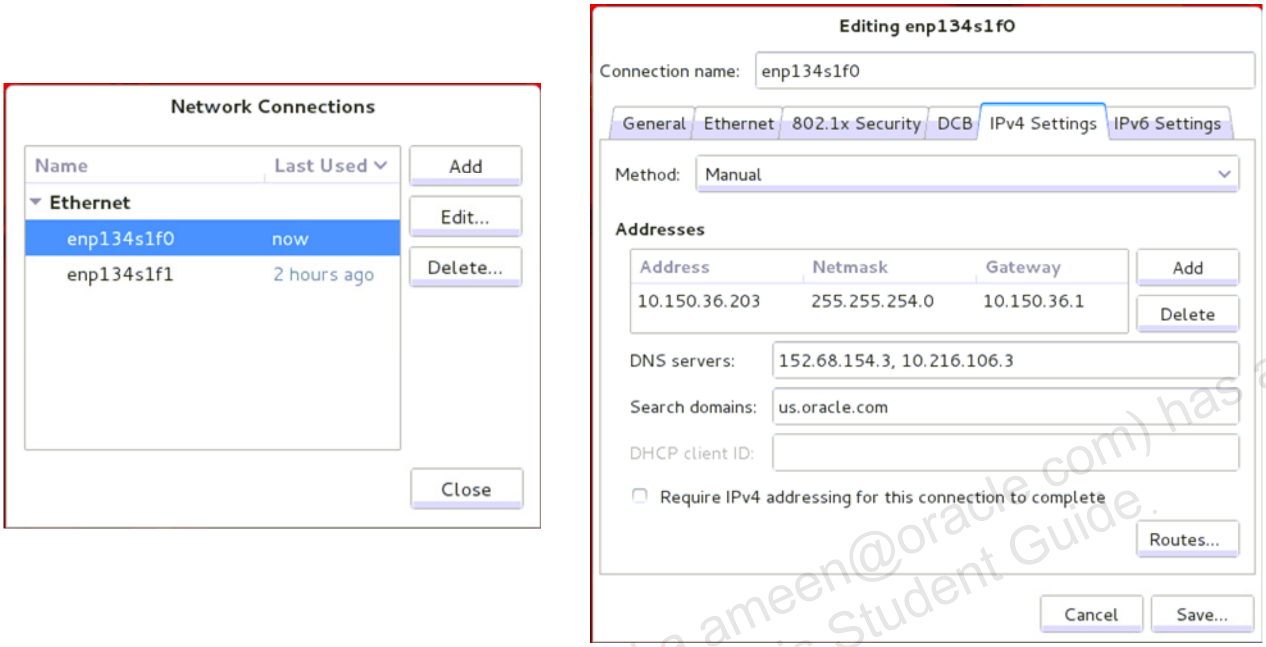
Edit an Existing Network Connection



To edit an existing interface, select the interface from the list on the left of the Network Settings editor window, and click the gear icon in the lower right-hand corner of the window. The window shown in the slide displays. You can then select from the following list of options:

- **Details:** Displays the same information displayed on the Network Settings editor window such as IPv4 and IPv6 addresses, hardware address, default route, and any DNS servers
- **Security:** Enable or disable 802.1x security, choose an authentication method, and provide additional authentication information based on the chosen method. Available authentication methods are MD5 (message-digest algorithm), TLS (Transport Layer Security), FAST (Flexible Authentication via Secure Tunneling), Tunneled TLS, or PEAP (Protected Extensible Authentication Protocol).
- **Identify:** Specify the device Name, MAC address, cloned address, and MTU
- **IPv4:** Enable or disable IPv4, specify DHCP or IPv4 address, netmask, and gateway, enable or disable DNS, and specify DNS servers.
- **IPv6:** Enable or disable IPv6, specify DHCP or IPv6 address, enable or disable DNS, specify DNS servers, enable or disable routes, and specify route address and prefix.
- **Reset:** Specify Reset to reset the settings for the network or Forget to remove all details and do not attempt to automatically connect.

Network Connections Editor



ORACLE

You can also use the Network Connections editor window to add, delete, or edit network interface information. The following commands install the required package and display the Network Connections window:

```
# yum install nm-connection-editor
# nm-connection-editor
```

The example in the slide shows the use of the Network Connections window to configure IPv4 settings for an Ethernet connection. The window on the left shows the existing Ethernet connections. Select an entry from the list and then click **Edit** to modify parameters. Tabs on the Editing window include:

- **General:** Specify to automatically connect and other general parameters.
- **Ethernet:** Specify the MAC address and the MTU.
- **802.1x Security:** Enable 802.1x security and specify authentication information.
- **DCB:** Enable Data Center Bridging (DCB) and specify parameters.
- **IPv4 Settings:** Specify IPv4 settings as shown in the slide.
- **IPv6 Settings:** Specify IPv6 settings.

The `nmcli` Utility

- Command-line tool used to control NetworkManager.
  - Useful for scripting and for controlling NetworkManager without a GUI
- The command provides five different categories, or objects:
  - `general`: NetworkManager's general status and operations
  - `networking`: Overall networking control
  - `radio`: NetworkManager radio switches
  - `connection`: NetworkManager's connections
  - `device`: Devices managed by NetworkManager
- See the `nmcli-examples(5)` manual page for examples.

ORACLE

NetworkManager includes a command-line tool, `nmcli`, which is used to control NetworkManager. You can use `nmcli` to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status. The syntax is:

```
nmlci OPTIONS OBJECT { COMMAND | help }
```

There are five different objects: general, networking, radio, connection, and device. Use the `help` argument to display the options and information about the five different objects:

```
# nmcli help
...
OPTIONS
...
OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
```

The `nmcli general` Object

- The `nmcli general` object provides the following commands:
  - `status`: Show the overall status of NetworkManager.
  - `hostname`: Get or change the system hostname. The system hostname is stored in `/etc/hostname`.
  - `permissions`: Show permissions for the various authenticated operations that NetworkManager provides.
  - `logging`: Get or change NetworkManager logging level for domains. See the `NetworkManager.conf(5)` man page for description of log levels and domains.

Use the `nmcli general` object to show NetworkManager status and permissions. This command also allows you view and change the system hostname and the NetworkManager logging level. The following command provides help on the `nmcli general` object:

```
# nmcli general help
Usage: nmcli general { COMMAND | help }
COMMAND := { status | hostname | permissions | logging }
...
```

Some examples of using this command follow. Use the following command to display the overall status of NetworkManager. The `status` argument is the default and can be omitted.

```
# nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  disabled
```

The `hostname` argument is used to display or change the system hostname. The hostname is stored in the `/etc/hostname` file. The following example changes the hostname to `my_host.us.oracle.com` and updates the `/etc/hostname` file:

```
# nmcli general hostname my_host.us.oracle.com
```

The `permissions` argument shows the permissions a caller has for the various authenticated operations that NetworkManager provides. The following example shows permissions for enabling and disabling networking, changing Wi-Fi and WWAN state, modifying connections, and other operations:

```
# nmcli general permissions
PERMISSION                                                 VALUE
org.freedesktop.NetworkManager.enable-disable-network      yes
org.freedesktop.NetworkManager.enable-disable-wifi         yes
org.freedesktop.NetworkManager.enable-disable-wwan         yes
org.freedesktop.NetworkManager.enable-disable-wimax        yes
org.freedesktop.NetworkManager.sleep-wake                  yes
org.freedesktop.NetworkManager.network-control             yes
org.freedesktop.NetworkManager.wifi.share.protected        yes
org.freedesktop.NetworkManager.wifi.share.open             yes
org.freedesktop.NetworkManager.settings.modify.system      yes
org.freedesktop.NetworkManager.settings.modify.own         yes
org.freedesktop.NetworkManager.settings.modify.hostname    yes
```

The `logging` argument is used to get and change NetworkManager logging level for domains. Without any argument, the current logging level and domains are shown as follows:

```
# nmcli general logging
LEVEL  DOMAINS
INFO   PLATFORM,RFKIL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,IP4,IP6,
AUTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,
DEVICE,OLPC,WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,TEAM,
CONCHECK,DCB
```

To change logging state, provide the level and/or domain parameters using the following syntax:

```
nmcli general logging [level <log level>] [domains <log domains>]
```

The logging level can be one of the following (listed in order of verbosity):

- **ERR**: Logs only critical errors
- **WARN**: Logs warnings that might reflect operation
- **INFO**: Logs various informational messages that are useful for tracking state and operations
- **DEBUG**: Enables verbose logging for debugging purposes

The following example sets the logging level to `DEBUG` for the `IPv4` domain:

```
# nmcli general logging level DEBUG domains IP4
```

The following example sets the logging level to `INFO` for all domains:

```
# nmcli general logging level INFO domains ALL
```

For information on configuring NetworkManager logging and for domain descriptions, see the `NetworkManager.conf(5)` man page.

The `nmcli networking` Object

- The `nmcli networking` object provides the following commands:
  - `on`: Enable networking by NetworkManager.
  - `off`: Disable networking by NetworkManager.
  - `connectivity [check]`: Get network connectivity state.
- To display networking status:

```
# nmcli networking
enabled
```

- To get network connectivity state:

```
# nmcli networking connectivity check
full
```

  - Possible states: `none`, `portal`, `limited`, `full`, `unknown`.

Use the `nmcli networking` object to show NetworkManager networking status, or to enable and disable networking. Disabling networking removes the configuration from all devices and changes them to the "unmanaged" state. The following command provides help on the `nmcli networking` object:

```
# nmcli networking help
Usage: nmcli networking { COMMAND | help }
COMMAND := { [ on | off | connectivity ] }
...
```

Some examples of using this command are given. The following sequence of commands displays the networking status and then disables and enables networking:

```
# nmcli networking
enabled
# nmcli networking off
# nmcli networking
disabled
# nmcli networking on
```

The `connectivity` argument shows the network connectivity state. An optional `check` argument tells NetworkManager to recheck the connectivity. Without the `check` argument, the command displays the most recent known connectivity state without rechecking. The following example includes the `check` argument:

```
# nmcli networking connectivity check
full
```

Possible states are:

- **`none`**: The host is not connected to any network.
- **`portal`**: The host is behind a captive portal and cannot reach the full Internet.
- **`limited`**: The host is connected to a network, but it has no access to the Internet.
- **`full`**: The host is connected to a network and has full access to the Internet.
- **`unknown`**: The connectivity status cannot be determined.

The `nmcli radio` Object

- The `nmcli radio` object provides the following commands:
  - `wifi [ on | oof ]`: Get or set status of Wi-Fi in NetworkManager.
  - `wwan [ on | oof ]`: Get or set status of WWAN (mobile broadband).
  - `wimax [ on | oof ]`: Get or set status of WiMAX (Worldwide Interoperability for Microwave Access).
    - WiMAX support is a compile-time decision.
  - `all [ on | oof ]`: Get or set status of all the above.
- To display status of all the radio switches:

```
# nmcli radio
WIFI-HW   WIFI      WWAN-HW   WWAN
enabled   enabled   enabled   enabled
```


Use the `nmcli radio` object to show radio switch status, or to enable and disable the switches. The following command provides help on the `nmcli radio` object:

```
# nmcli radio help
Usage: nmcli radio { COMMAND | help }
COMMAND := { [ all | wifi | wwan | wimax ] }
...
```

Some examples of using this command are given. The following sequence of commands displays the radio switch status and then disables Wi-Fi in NetworkManager:

```
# nmcli radio
WIFI-HW   WIFI      WWAN-HW   WWAN
enabled   enabled   enabled   enabled
# nmcli radio wifi off
# nmcli radio
WIFI-HW   WIFI       WWAN-HW   WWAN
enabled   disabled   enabled   enabled
```

The `nmcli connection` Object

- NetworkManager stores all network configuration information as connections.
- Connections contain all the information required to create or connect to a network.
- There can be multiple connections for a given device.
- Only one connection can be active on a device at a time.
- The `nmcli connection` object provides the following commands, as displayed, by using the `help` argument:

```
# nmcli connection help
Usage: nmcli connection { COMMAND | help }
COMMAND := {show | up | down | add | modify | edit |
    delete | reload | load }
```


Use the `nmcli connection` object to start, stop, and manage network connections.

NetworkManager stores all network configuration information as connections. Connections contain all the information, such as MAC address and IP address, required to create or connect to a network. A connection is active when a device uses that connection's configuration to create or connect to a network.

There can be multiple connections for a given device but only one of them can be active on that device at any given time. The additional connections can be used to allow quick switching between different networks and configurations. For example, you can have a connection defined for a network interface that uses static IP addressing. You could have a second connection defined for the same network interface that uses DHCP.

The following command provides help on the `nmcli connection` object:

```
# nmcli connection help
Usage: nmcli connection { COMMAND | help }
COMMAND := { show | up | down | add | modify | edit | delete |
reload | load }
...
```

The various commands for the `nmcli connection` object are described on the following slides.

The `nmcli connection show` Command

- Use the show argument to list connection profiles.
- Include the `--active` option to list only the active profiles.

```
# nmcli connection show --active
NAME         UUID  TYPE             DEVICE
virbr0       ...   bridge           virbr0
enp134s1f0   ...   802-3-ethernet   eth0
```

- View detailed information by specifying an <*ID*> keyword and associated value:

```
# nmcli connection show id enp134s1f0
connection.id:                enp134s1f0
connection.type:              802-3-ethernet
connection.autoconnect:       yes
...
```


Use the `show` argument to list connection profiles. Include the `--active` option to list only the active profiles. Example:

```
# nmcli connection show --active
NAME         UUID  TYPE             DEVICE
virbr0       ...   bridge           virbr0
enp134s1f0   ...   802-3-ethernet   eth0
```

You can also view detailed information for a specific connection by specifying an optional <*ID*> keyword followed by an associated value. The <*ID*> can be `id`, `uuid`, `path`, or `apath`. The following example uses the `id` keyword to show detailed information for the `enp134s1f0` connection. Only partial output is shown:

```
# nmcli connection show id enp134s1f0
...
802-3-ethernet.mac-address:  00:14:4F:8D:B0:BC
ipv4.addresses:              { ip = 10.150.36.203/23, gw = ... }
GENERAL.DBUS-PATH:
/org/freedesktop/NetworkManager/ActiveConnection/2
...
```

The `nmcli connection up|down` Commands

- Use the `up` argument to activate a connection.
- The connection is specified by its ID, UUID, or D-Bus path.
- The following example uses the connection ID:

```
# nmcli connection up id enp134s1f1
Connection successfully activated (D-Bus active path:
   /org/freedesktop/NetworkManager/ActiveConnection/4)
```

- Use the `down` argument to deactivate a connection.

```
# nmcli connection down id enp134s1f1
```

- If the connection is set to autoconnect, the connection starts automatically on the disconnected device again.
- In this case, use the `nmcli device disconnect` command instead of `nmcli connection down`.

Use the `up` argument to activate a connection. The connection is specified by its name, UUID, or D-Bus path. When requiring a particular device to activate the connection on, use the `ifname` option with the interface name.

The following example activates the `enp134s1f1` connection. The `show` argument is issued before and after to illustrate the result of the `up` argument:

```
# nmcli connection show
NAME         UUID   TYPE             DEVICE
enp134s1f0   ...    802-3-ethernet   eth0
enp134s1f1   ...    802-3-ethernet
# nmcli connection up id enp134s1f1
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
# nmcli connection show
NAME         UUID   TYPE             DEVICE
enp134s1f0   ...    802-3-ethernet   eth0
enp134s1f1   ...    802-3-ethernet   eth1
```

Use the `down` argument to deactivate a specific active connection. The following example deactivates the `enp134s1f1` connection. The `show` argument is issued before and after to illustrate the result of the `down` argument:

```
# nmcli connection show
NAME        UUID  TYPE            DEVICE
virbr0      ...   bridge          virbr0
enp134s1f0  ...   802-3-ethernet  eth0
enp134s1f1  ...   802-3-ethernet  eth1
# nmcli connection down id enp134s1f1
# nmcli connection show
NAME        UUID  TYPE            DEVICE
virbr0      ...   bridge          virbr0
enp134s1f0  ...   802-3-ethernet  eth0
enp134s1f1  ...   802-3-ethernet
```

If the connection has the "`connection.autoconnect`" flag set to "yes," the connection automatically starts on the disconnected device again. In this case, use the `nmcli device disconnect` command instead of the `nmcli connection down` command.

The `nmcli connection add` Command

- Use the `add` argument to add a connection for NetworkManager.
- The command accepts the following options:
  - Common options: `type`, `ifname`, `con-name`, more
  - Type-specific options: `mac`, `mtu`, `ssid`, more
  - IP options: `ip4`, `ip6`, `gw4`, `gw6`
- Example:

```
# nmcli connection add con-name new-eth0 ifname eth0
   type ethernet ip4 192.168.2.100/24 gw4 192.168.2.1
Connection 'new-eth0' (<UUID>) successfully added.
```


ORACLE

Use the `add` argument to add a connection for NetworkManager. The syntax follows:

```
nmcli connection add  COMMON_OPTIONS TYPE_SPECIFIC OPTIONS
IP_OPTIONS
```

The COMMON_OPTIONS for the `add` argument are described:

- **`type <type>`**: Connection type. Valid types of connections are `ethernet`, `wifi`, `wimax`, `pppoe`, `gsm`, `cdma`, `infiniband`, `bluetooth`, `vlan`, `bond`, `bond-slave`, `team`, `team-slave`, `bridge`, `bridge-slave`, `vpn`, and `olpc-mesh`.
- **`ifname <ifname>`**: Interface to bind the connection to. A special value of `"*"` can be used for interface-independent connections.
- **`con-name <connection_name>`**: Connection name. This is optional. When not provided, a default name is generated, `<type>[-<ifname>][-<num>]`.
- **`autoconnect yes|no`**: Whether the connection profile can be automatically activated. This is optional. The default is `yes`.
- **`save yes|no`**: Whether the connection is persistent. This is optional. The default is `yes`.

Some of the `TYPE_SPECIFIC OPTIONS` for the `add` argument are given on the next page. Refer to the `nmcli(1)` man page for the complete list of options.

The following lists the `TYPE_SPECIFIC OPTIONS` for Ethernet and WiFi connections:

- **`ethernet` TYPE_SPECIFIC OPTIONS:**
  - **`mac`** <***`MAC_address`***>: MAC address of the device this connection is locked to
  - **`cloned-mac`** <***`cloned_MAC_address`***>: Clone MAC address
  - **`mtu`** <***`MTU`***>: MTU
- **`wifi` TYPE_SPECIFIC OPTIONS:**
  - **`ssid`** <***`SSID`***>: SSID
  - **`mac`** <***`MAC_address`***>: MAC address of the device this connection is locked to
  - **`cloned-mac`** <***`cloned_MAC_address`***>: Clone MAC address
  - **`mtu`** <***`MTU`***>: MTU

The `IP_OPTIONS` for the `add` argument are described:

- **`ip4`** <***`IPv4_address`***> **`gw4`** <***`IPv4_address`***>: IPv4 addresses
- **`ip6`** <***`IPv6_address`***> **`gw6`** <***`IPv6_address`***>: IPv6 addresses

The following example adds an Ethernet connection. The `nmcli connection show` command is issued afterwards to view the results. Only partial output is shown.

```
# nmcli connection add con-name new-eth0 ifname eth0 type ethernet
ip4 192.168.2.100/24 gw4 192.168.2.1
Connection 'new-eth0' (<UUID>) successfully added.
# nmcli connection show
NAME        UUID        TYPE              DEVICE
new-eth0    ...         802-3-ethernet
```

Each new connection creates an associated network interface configuration file in the `/etc/sysconfig/network-scripts` directory. For example:

```
# ls /etc/sysconfig/network-scripts/ifcfg*
ifcfg-enp134s1f0   ifcfg-enp134s1f1   ifcfg-lo   ifcfg-new-eth0
```

The `nmcli connection edit` Command

- Use the `edit` argument to edit a connection by using an interactive editor. Example:

```
# nmcli connection edit new-eth0
...
You may edit the following settings: connection, 802-3-
   ethernet (ethernet), 802-1x, ipv4, ipv6, dcb
nmcli>
```

  - Use the '?' key or type 'help' to view commands.
- Use the `edit` argument without specifying a connection to add a new connection. Example:

```
# nmcli connection edit new-eth0
...
Enter connection type:
...
```


Use the `edit` argument to edit an existing connection, identified by the connection ID, UUID, or D-Bus path. The following example specifies editing of the `new-eth0` connection:

```
# nmcli connection edit new-eth0
===| nmcli interactive connection editor |===
Editing existing '802-3-ethernet' connection: 'new-eth0'
Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property
description.
You may edit the following settings: connection, 802-3-ethernet
(ethernet), 802-1x, ipv4, ipv6, dcb
nmcli>
```

Use the '?' key or type 'help' to display the available commands. Only partial output follows:

```
nmcli> ?
...
set    [<setting>.<prop> <value>]   :: set property value
quit                                :: exit nmcli
```

Use the `edit` argument without specifying a connection identifier to add a new connection. The interactive editor guides you through the connection editing. The following example adds a new Ethernet connection:

```
# nmcli connection edit
Valid connection types: generic, 802-3-ethernet (ethernet), pppoe,
802-11-wireless (wiji), wimax, gsm, cdma, infiniband, adsl,
bluetooth, vpn, 802-11-olpc-mesh (olpc-mesh), vlan, bond, team,
bridge, bond-slave, team-slave, bridge-slave
Enter connection type: 802-3-ethernet
===| nmcli interactive connection editor |===
Adding a new '802-3-ethernet' connection
Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property
description.
You may edit the following settings: connection, 802-3-ethernet
(ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> set connection.id new-eth1
nmcli> set connection.interface-name eth1
nmcli> set connection.autoconnect yes
nmcli> set 802-3-ethernet.mtu auto
nmcli> set ipv4.method manual
nmcli> set ipv4.addresses 192.168.2.101/24 192.168.2.1
nmcli> set ipv6.method auto
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in
an immediate activation of the connection.
Do you still want to save? [yes] ENTER
Connection 'new-eth1' (<UUID>) successfully saved.
nmcli> quit
```

The following `connection show` command lists the new '`new-eth1`' connection:

```
# nmcli connection show
NAME      UUID               TYPE             DEVICE
...
new-eth1  ...                802-3-ethernet   eth1
...
```

A new network interface configuration file is created in the `/etc/sysconfig/network-scripts` directory:

```
# ls /etc/sysconfig/network-scripts/ifcfg*
ifcfg-enp134s1f0   ifcfg-enp134s1f1   ifcfg-lo   ifcfg-new-eth0
ifcfg-new-eth1
```

The nmcli connection modify Command

- Use the modify argument to modify properties in the connection profile.
  - The following example modifies the IPv4 DNS server address property (ipv4.dns) for the new-eth1 connection:

```
# nmcli connection modify new-eth1 ipv4.dns 152.68.154.3
```

  - Use the '+' prefix to append a value:

```
# nmcli connection modify new-eth1 +ipv4.dns
   10.216.106.3
```


Use the modify argument to modify one or more properties in the connection profile. Identify the connection to modify by its ID, UUID, or D-Bus path. The provided value overwrites the existing property value.

Use an empty value ("") to set the property value to the default. You can use the + prefix for the property name to append an item to the existing value, or use the - prefix to remove a specified value. The following example modifies the IPv4 DNS server address. The show argument displays the values before and after the modification:

```
# nmcli connection show new-eth1
...
ipv4.dns:
...
# nmcli connection modify new-eth1 ipv4.dns 152.68.154.3
# nmcli connection show new-eth1
...
ipv4.dns:          152.68.154.3
...
```

The `nmcli connection delete | reload | load` Commands

- Use the `delete` argument to delete a configured connection. For example:

```
# nmcli connection delete new-eth1
```

- Use the `reload` argument to reload all connection files from disk. For example:

```
# nmcli connection reload
```

- Use the `load` argument to load or reload one or more configuration files from disk. For example:

```
# nmcli connection load /etc/sysconfig/network-
  scripts/ifcfg-new-eth0
```

- Set the `monitor-connection-files` to `true` to enable the auto-loading feature.

The remaining three arguments to the `nmcli connection` command are given in the slide.

Use the `delete` argument to delete a configured connection profile. Identify the connection to delete by its ID, UUID, or D-Bus path.

Use the `reload` argument to reload all configuration files from disk. Use this command to tell NetworkManager to re-read the connection profiles from disk whenever a change was made to them. Set the `monitor-connection-files` to `true` to enable the auto-loading feature. In this case, NetworkManager reloads connection files any time they change.

Use the `load` argument to load or reload one or more specific configuration files from disk. This is not needed if the auto-loading feature is enabled for the connection.

The `nmcli device` Object

- The `nmcli device` object provides the following commands:
  - `status`: Display the status of all devices.
  - `show [<ifname>]`: Show detailed information about devices.
  - `connect <ifname>`: Connect the device.
  - `disconnect <ifname>`: Disconnect the device.
  - `wifi list | connect | rescan`: List Wi-Fi access points. Connect to a Wi-Fi network. Rescan for available access points.
- To display status of all the devices:

```
# nmcli device
DEVICE       TYPE      STATE                    CONNECTION
enp134s1f0   ethernet  connected                enp134s1f0
```


Use the `nmcli device` object to show and manage network interfaces. The following command provides help on the `nmcli device` object:

```
# nmcli device help
Usage: nmcli device { COMMAND | help }
COMMAND := { status | show | connect | disconnect | wifi }
...
```

Some examples of using this command are given. The following sequence of commands displays the status of all devices. The `status` argument is the default.

```
# nmcli device
DEVICE       TYPE      STATE                    CONNECTION
enp134s1f0   ethernet  connected                enp134s1f0
enp134s1f1   ethernet  disconnected
lo           loopback  unmanaged
```

The following example displays detailed information about a device:

```
# nmcli device show
GENERAL.DEVICE:             enp134s1f0
GENERAL.TYPE:               ethernet
GENERAL.HWADDR:             00:14:4F:8D:B0:BC
GENERAL.MTU:                1500
...
IP4.ADDRESS[1]:             ip = 10.150.36.203/23, gw = 10.150.36.1
IP4.DNS[1]:                 152.68.154.3
...
GENERAL.DEVICE:             enp134s1f1
GENERAL.TYPE:               ethernet
GENERAL.HWADDR:             00:14:4F:8D:B0:BD
GENERAL.MTU:                1500
...
```

The following example shows the effect of using the `disconnect` and `connect` arguments:

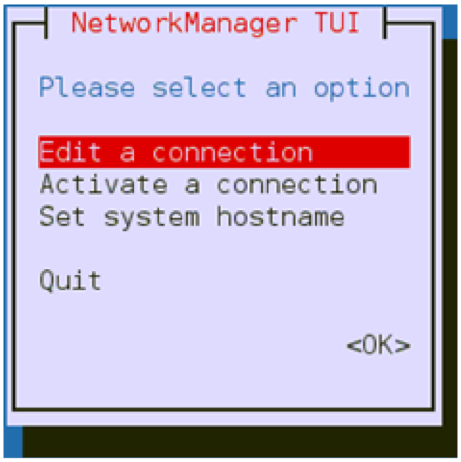
```
# nmcli device disconnect enp134s1f0
# nmcli device
DEVICE      TYPE      STATE                          CONNECTION
enp134s1f0  ethernet  disconnected
...
# nmcli device connect enp134s1f0
Device 'enp134s1f0' successfully activated with <UUID>.
# nmcli device
DEVICE      TYPE      STATE                          CONNECTION
enp134s1f0  ethernet  connected                      enp134s1f0
...
```

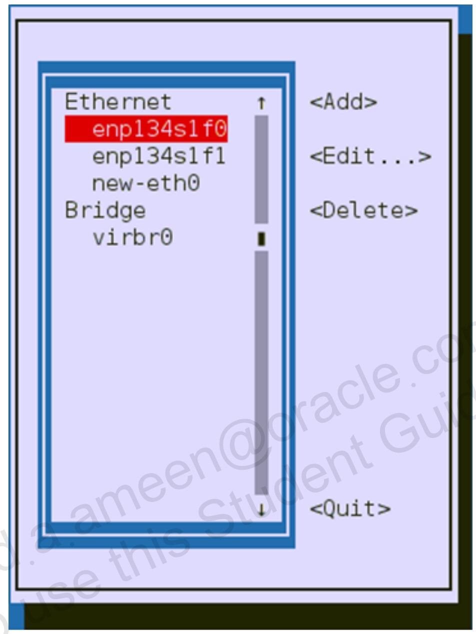
The "`nmcli device wifi`" command provides the following arguments:

- **`list`**: List available Wi-Fi access points.
- **`connect <(B)SSID>`**: Connect to a Wi-Fi network specified by Service Set Identifier (SSID) or Basic Service Set Identifier (BSSID).
- **`rescan`**: Request that NetworkManager re-scan for available Wi-Fi access points.

The `nmtui` Utility

```
# nmtui
```





ORACLE

A text-based user interface (TUI) for NetworkManager exists to add or edit a network connection, to activate a connection, and to set the system hostname. Enter the `nmtui` command to display the first of the two screens shown in the slide. Use the Tab key or arrow keys to highlight a selection.

Highlight Edit a connection and press Enter to display the second screen. From this screen, you can edit and delete an existing connection, or add a new connection.

Use the following command to install the package that provides the `nmtui` utility:

```
# yum install NetworkManager-tui
```

The `ip` Utility

- Use to display and manipulate devices, routing, policy routing, and tunnels.
- Replaces the `ifconfig` command
- Provides a number of OBJECT arguments, such as:
  – `link`: Network device
  – `address` (or `addr`): IPv4 or IPv6 address on a device
  – `route`: Routing table entry
- Provides a number of COMMANDS for each OBJECT, such as:
  – `add`, `change`, `del`, `show`, more
- Use `help` to show COMMANDS available for an OBJECT.

```
# ip addr help
```

ORACLE

You can use the `ip` command to display the status of an interface, configure network properties, or for debugging or tuning the network. The `ip` command replaces the `ifconfig` command, which is deprecated. The syntax of the `ip` utility follows:

```
ip [OPTIONS] OBJECT {COMMAND | help}
```

OBJECT

The following describes the OBJECT field:

- **`link`**: Network interface device
- **`address`** (or **`addr`**): Protocol (IPv4 or IPv6) address on a device
- **`addrlabel`**: Label configuration for protocol address selection.
- **`route`**: Routing table entry
- **`rule`**: Rule in routing policy database
- **`neighbour`** (or **`neigh`**): Manage ARP or NDISC cache entries.
- **`ntable`**: Manage the neighbor cache's operation.
- **`tunnel`**: Tunnel over IP
- **`tuntap`**: Manage TUN/TAP devices.
- **`maddress`** (or **`maddr`**): Multicast address

The OBJECT field description continues:

- **mroute**: Multicast routing cache entry.
- **mrule**: Rule in multicast routing policy database
- **monitor**: Watch for netlink messages.
- **xfrm**: Manage IPSec policies.
- **netns**: Manage network namespaces.
- **l2tp**: Tunnel ethernet of IP (L2TPv3)
- **tcp_metrics** (or **tcpmetrics**): Manage TCP metrics.

COMMAND

The COMMAND specifies the action to perform on the object. The set of possible actions depends on the object type. In general, you can add, delete, and show objects. Some objects do not allow all of these operations, some objects allow more operations. The help command displays a list of available commands and syntax for a specified object. The following example gives the commands available for the addr object. Only partial output is shown.

```
# ip addr help
Usage: ip addr {add|change|replace} IFADDR dev STRING [ LIFE...
       ip addr del IFADDR dev STRING
       ip addr {show|save|flush} [ dev STRING ] [ scope ...
       ip addr {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
...
```

As shown in this example, the commands for the addr object are add, change, replace, del (or delete), show, save, flush, showdump, and restore. The addr object is discussed further in the next slide.

OPTIONS

The following describes the available OPTIONS for the ip utility:

- **-V**, **-Version**: Display the version of the ip utility.
- **-b**, **-batch** <***FILENAME***>: Read commands from <*FILENAME*> or from standard input and execute them. A failure in a command in batch mode terminates ip.
- **-force**: Do not terminate ip on errors in batch mode.
- **-s**, **-stats**, **-statistics**: Display more information. If the option appears twice or more, the amount of information increases.
- **-l**, **-loops** <***COUNT***>: Specify the maximum number of loops the 'ip addr flush' logic attempts. The default is 10. Zero (0) means loop until all addresses are removed.
- **-f**, **-family** <***FAMILY***>: Specify the protocol family: inet, inet6, bridge, ipx, dnet, or link. The link family is a special identifier meaning that no networking protocol is involved. The respective shortcuts are -4, -6, -B, -I, -D, and -0.
- **-o**, **-oneline**: Output each record on a single line.
- **-r**, **-resolve**: Use the system's name resolver to print DNS names instead of host addresses.

The `ip addr` Object

- Use the `ip addr` object to show and manage IPv4 or IPv6 address on a device.
  - Changes made with `ip` are not persistent.
- To show the status of all active devices:

```
# ip addr
```

- To add an IPv4 address to a network interface device:

```
# ip addr add 192.168.50.5/24 dev enp134s1f0
```

- To remove an IPv4 address from a device:

```
# ip addr del 192.168.50.5/24 dev enp134s1f0
```

- To remove all IP addresses from a device:

```
# ip addr flush dev enp134s1f0
```


Use the `ip addr` object to show and manage IPv4 or IPv6 address on a device. The following example shows IP status for all active devices. The `show` command is the default.

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp134s1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
    inet 10.150.36.203/23 brd 10.150.37.255 scope global
enp134s1f0
    inet6 fe80::214:4fff:fe8d:b0bc/64 scope link
       valid_lft forever preferred_lft forever
...
```

The following example uses the add argument to add the IPv4 address 192.168.50.5/24 to the enp134s1f0 interface. The show argument is given afterwards to display the result. This example assumes the interface already has 10.150.36.203/23 assigned to it.

```
# ip addr add 192.168.50.5/24 dev enp134s1f0
# ip addr show enp134s1f0
...
link/ether 00:14:4f:8d:b0:bc brd: ff:ff:ff:ff:ff:ff
inet 10.150.36.203/23 brd 10.150.37.255 scope global enp134s1f0
inet 192.168.50.5/24 scope global enp134s1f0
inet6 fe80::214:4fff:fe8d:b0bc/64 scope link
...
```

Use the del argument to delete the IPv4 address. Example:

```
# ip addr del 192.168.50.5/24 dev enp134s1f0
# ip addr show enp134s1f0
...
link/ether 00:14:4f:8d:b0:bc brd: ff:ff:ff:ff:ff:ff
inet 10.150.36.203/23 brd 10.150.37.255 scope global enp134s1f0
inet6 fe80::214:4fff:fe8d:b0bc/64 scope link
...
```

Use the flush argument to remove all the IPv4 addresses assigned to an interface. Example:

```
# ip addr flush dev enp134s1f0
# ip addr show enp134s1f0
...
link/ether 00:14:4f:8d:b0:bc brd: ff:ff:ff:ff:ff:ff
...
```

Any settings that you configure for network interfaces using ip do not persist across system reboots. To make the changes permanent, set the properties in the /etc/sysconfig/network-scripts/ifcfg-<*interface*> file.

Refer to the ip-address(8) man page for more information on using commands available for the ip addr object.

The `ip link` Object

- Use the `ip link` object to show and manage the attributes of network interfaces on the system.
  - Changes made with `ip` are not persistent.
- To show the status of a specific device:

```
# ip link show enp134s1f0
```

- To bring a specific network interface down:

```
# ip link set enp134s1f0 down
```

- To bring a specific network interface up:

```
# ip link set enp134s1f0 up
```

- To change a device attribute, for example, MTU:

```
# ip link set enp134s1f0 mtu 1000
```


Use the `ip link` object to show and manage the state of network interface devices on the system. The following example gives the commands available for the `link` object. Only partial output is shown.

```
# ip link help
Usage: ip link add [link DEV] [name] NAME
       ...
       ip link delete DEV type TYPE [ARGS]
       ...
       ip link set {dev DEVICE | group DEVGROUP} [{up | down}]
       ...
       ip link show [DEVICE | group GROUP] [up]
...
```

As this example shows, the commands for the `link` object are `add`, `delete`, `set`, and `show`. The `TYPE` argument can be any of the following: `vlan`, `veth`, `vcan`, `dummy`, `ifb`, `macvlan`, `can`, `bridge`, `ipoib`, `ip6tnl`, `ipip`, `sit`, or `vxlan`.

Refer to the `ip-link(8)` man page for more information.

The following example shows the status of all active devices. The `show` argument is the default. Notice that the output is similar to that of the `ip addr` command, but without the IP address information.

```
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp134s1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
...
```

Use the `set` argument to change device attributes. The `up` and `down` arguments change the state of the device. The following example brings the `enp134s1f0` interface down and then back up. The `show` argument displays the results of the `set` argument.

```
# ip link set enp134s1f0 down
# ip link show enp134s1f0
2: enp134s1f0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast
state DOWN mode DEFAULT qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
# ip link set enp134s1f0 up
# ip link show enp134s1f0
2: enp134s1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
```

The following example uses the `set` argument to change the MTU attribute to 1000:

```
# ip link set enp134s1f0 mtu 1000
# ip link show enp134s1f0
2: enp134s1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1000 qdisc
pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
```

Address Resolution Protocol (ARP)

- ARP resolves an IP address to the MAC address.
- IP addresses and associated MAC addresses are cached in an ARP table.
  - By default, entries are cached for 60 seconds.
- Use the `ip neigh` object to display, add, or delete entries in the ARP table.
  - The `arp` command is deprecated.
- To display all entries:

```
# ip neigh
...
```

- To delete all entries:

```
# ip neigh flush all
```


ARP resolves an IP address to the MAC address. The MAC address is a 48-bit physical hardware address, which is burned into the network interface card (NIC). Network applications use the IP address to communicate with another device but the MAC address is needed to ensure network packets are delivered. The following example uses the `ip addr show` command to display the MAC address, `00:14:4f:8d:b0:bc`, and the IP address, `10.150.36.203`, for the `enp134s1f0` network interface:

```
# ip addr show enp134s1f0
2: enp134s1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP qlen 1000
    link/ether 00:14:4f:8d:b0:bd brd ff:ff:ff:ff:ff:ff
    inet 10.150.36.203/23 brd 10.150.37.255 scope global enp...
```

For performance reasons, ARP caches resolve IP addresses and associate MAC addresses in an ARP table (or cache). By default, entries are cached for 60 seconds. This value can be modified on a per–network interface basis. For example, the following file stores the timeout value for the `enp134s1f0` interface:

```
# cat /proc/sys/net/ipv4/neigh/enp134s1f0/gc_stale_time
60
```

Use the `ip neigh` object to display the ARP table, to delete an ARP entry, or to add an entry to the table. The `ip neigh` object replaces the `arp` command, which is deprecated. The ARP table is also known by another name, the IP neighbor table.

Use the following command to display the commands available for the `ip neigh` object. Only partial output is displayed.

```
# ip neigh help
Usage: ip neigh { add | del | change | replace } { ADDR ...
...
       ip neigh {show|flush} [ to PREFIX ]  [ dev DEV ] [ nud
STATE ]
```

The `ip neigh` object commands are summarized as follows:

- **`ip neigh add`**: Add a new neighbor entry.
- **`ip neigh change`**: Change an existing entry.
- **`ip neigh replace`**: Add a new entry or change an existing entry.
- **`ip neigh delete`**: Delete a neighbor entry.
- **`ip neigh show`**: List neighbor entries.
- **`ip neigh flush`**: Flush neighbor tables.

The following example displays the ARP table. The `show` command is the default.

```
# ip neigh
10.150.36.201 dev enp134s1f0 lladdr 00:14:4f:a6:90:9d STALE
10.150.36.202 dev enp134s1f0 lladdr 00:14:4f:9a:69:d7 STALE
10.150.36.204 dev enp134s1f0 lladdr 00:14:4f:9e:d9:15 STALE
```

The following example clears all entries in the ARP table with verbosity:

```
# ip -s -s neigh flush all
10.150.36.201 dev enp134s1f0 lladdr 00:14:4f:a6:90:9d ref 1 ...
10.150.36.202 dev enp134s1f0 lladdr 00:14:4f:9a:69:d7 ref 1 ...
10.150.36.204 dev enp134s1f0 lladdr 00:14:4f:9e:d9:15 ref 1 ...
*** Round 1, deleting 3 entries ***
*** Flush is complete after 1 round ***
```

The following example removes entries in the ARP table on device `enp134s1f0`:

```
# ip neigh flush dev enp134s1f0
```

Refer to the `ip-neighbour(8)` man page for more information.

The `ip route` Object

- Use the `ip route` object to display or manipulate the IP routing table.
- The default route, `GATEWAY`, is configured in the `/etc/sysconfig/network` file.
- To display the routing table:

```
# ip route
```

- To add an entry to the routing table:

```
# ip route add default via 10.150.36.2 dev enp134s1f0
  proto static
# ip route add 192.0.2.1 via 10.150.36.2 dev enp134s1f0
```

- Configure permanent static routes in the `/etc/sysconfig/network-scripts/route-`*`interface`* file.

The `ip route` utility displays or manipulates the IP routing table. Its primary use is to set up static routes to specific hosts or networks through a network interface.

To create a default route, include a `GATEWAY` entry in the `/etc/sysconfig/network` file. Use the `GATEWAYDEV` parameter to designate a specific interface. Example:

```
# cat /etc/sysconfig/network
GATEWAY=10.150.36.1
GATEWAYDEV=enp134s1f0
```

Network traffic destined for hosts on another network would be handled by the `10.150.36.1` gateway on the local area network.

Displaying the Routing Table

Use the `ip route` command to display the routing table. Example:

```
# ip route
default via 10.150.36.1 dev enp134s1f0 proto static metric 1024
10.150.36.0/23 dev enp134s1f0 proto kernel scope link src ...
192.168.122.0/24 dev virbr0 proto kernel scope link src ...
```

In this example, the gateway IP address of `10.150.36.1` was obtained from the entry in the `/etc/sysconfig/network` file. Refer to the `ip-route(8)` man pages for more information.

You can also use the `netstat -r` command to display the route table:

```
# netstat -r
```

Adding a Route

Use the `ip route add` command to add a static route. The following example adds a default route, which is used if no other route matches. All network packages using this route are "gatewayed" through the 192.0.2.2 IP address:

```
# ip route add default via 10.150.36.2 dev enp134s1f0 proto static
```

The following example adds a static route to a host address via a specific network interface.

```
# ip route add 192.0.2.1 via 10.150.36.2 dev enp134s1f0
```

Deleting a Route

Use the `ip route delete` command to delete an entry from the routing table, for example:

```
# ip route delete default via 10.150.36.2
# ip route delete 192.0.2.1
```

Configuring Permanent Static Routes

Any changes that you make to the routing table by using `ip route` do not persist across system reboots. To make static routes permanent, configure them for each interface. Static route configuration is stored in a `/etc/sysconfig/network-scripts/`*`route-interface`* file. For example, static routes for the `enp134s1f0` interface would be stored in the `/etc/sysconfig/network-scripts/route-enp134s1f0` file.

The route-interface file has two formats:

- IP command arguments
- Network/netmask directives

The IP command arguments format uses the following syntax:

```
x.x.x.x/x via x.x.x.x dev interface
```

Use the term `default` to create a default gateway, for example:

```
default via x.x.x.x dev interface
```

The following example creates a static route to the `192.168.2.0/24` subnet through an `enp134s1f1` interface (`10.10.10.1`):

```
# cat /etc/sysconfig/network-scripts/route-enp134s1f1
198.168.2.0/24 via 10.10.10.1 dev enp134s1f1
```

You can also use the network/netmask directives format for route-interface files. The format is as follows:

```
ADDRESS0=X.X.X.X NETMASK0=X.X.X.X GATEWAY0=X.X.X.X
```

The following example shows use of the IP command arguments to define the same entry:

```
ADDRESS0=198.168.2.0
NETMASK0=255.255.255.0
GATEWAY0=10.10.10.1
```

Start at `0` (as shown) and increment by one for each additional static route.

Quiz

Which of the following statements is true?

a. Network interface configuration files are located in the `/etc/sysconfig/network` directory.

b. The NetworkManager GUI can be used only to configure wired Ethernet devices.

c. Routing tables can be displayed by using the `netstat -r` command.

d. The `ipup eth0` command activates the `eth0` interface.

Quiz

Which of the following tasks can you perform using the `nmcli` utility?

a. Disable and enable networking operations
b. Manage network connection profiles
c. Change the system host name
d. Manage routing
e. Manage the ARP cache

Quiz

Which of the following tasks can you perform using the `ip` utility?

a. Manage network devices
b. Manage network addressing
c. Change the system host name
d. Manage routing
e. Manage the ARP cache

Summary

In this lesson, you should have learned how to:

- Describe network interface configuration files
- Describe additional network configuration files
- Start the network service
- Use the `ethtool` utility
- Describe NetworkManager
- Use the NetworkManager GUI
- Use the Network Connections Editor
- Use the `nmcli` utility
- Use the `nmtui` utility
- Describe ARP and the ARP cache
- Use the `ip` utility

Practice 15: Overview

The practices for this lesson cover the following:

- Configuring the `eth1` network interface
- Using NetworkManager with the GNOME GUI
- Using the Network Connection editor
- Using the `nmcli` utility
- Using the `nmtui` utility
- Using the `ip` utility

16

File Sharing

Objectives

After completing this lesson, you should be able to:

- Describe NFS
- Configure NFS server and client
- Describe the `exportfs` utility
- Describe and configure automounter
- Describe and configure `vsftpd`

Introduction to NFS

- NFS allows a Linux server to share directory hierarchies with Linux clients over a network.
- NFS servers export the directory, and NFS clients mount the exported directory.
- Oracle Linux 7 supports two versions:
  - NFSv3
  - NFSv4
- NFS relies on Remote Procedure Calls (RPC) between clients and servers.
- Several `nfs` and `rpc` services work together, depending on which version of NFS is implemented.

A Network File system (NFS) allows a server to share directory hierarchies (file systems) with remote systems over a network. NFS servers *export* the directory and NFS clients *mount* the exported directory. The server directory then appears to the client systems as if they were local directories. NFS reduces storage needs and improves data consistency and reliability, because users are accessing files that are stored on a centralized server.

Oracle Linux 7 does not support NFS version 2 (NFSv2). The following two versions are supported:

- NFS version 3 (NFSv3), specification is RFC 1813
- NFS version 4 (NFSv4), specification is RFC 3530

NFS relies on Remote Procedure Calls (RPC) between clients and servers. RPC services are controlled by the `rpcbind` service. The `rpcbind` service replaces `portmap`, which was used in previous versions of Linux to map RPC program numbers to IP address port number combinations. `rpcbind` responds to requests for RPC services and sets up connections to the requested RPC service.

`rpcbind` is not used with NFSv4, because the server listens on well-known TCP port 2049. The mounting and locking protocols have also been incorporated into the NFSv4 protocol, so NFSv4 does not interact with the `lockd` and `rpc.statd` daemons either.

NFS Server and RPC Processes

- Starting the `nfs-server` service starts the NFS server and other RPC processes.

```
# systemctl start nfs
```

- Several `nfsd` kernel threads are started. The number of threads are defined in `/proc/fs/nfsd/threads`.
- RPC process includes:
  - `rpc.statd`: Implements monitoring protocol (NSM) between NFS client and NFS server
  - `rpc.mountd`: NFS mount daemon that implements the server side of the mount requests from NFSv3 clients
  - `rpc.idmapd`: Maps NFSv4 names and local UIDs and GIDs
  - `rpc.rquotad`: Provides user quota information for remote users

Starting the `nfs-server` service starts the NFS server and other RPC processes needed to service requests for shared NFS file systems. You can use the short name "`nfs`" rather than "`nfs-server`" when starting the service. Example:

```
# systemctl start nfs
```

This is the NFS server process that implements the user level part of the NFS service. The main functionality is handled by the `nfsd` kernel module. The user space program merely specifies what sort of sockets the kernel server listens on, what NFS versions it supports, and how many `nfsd` kernel threads it uses. Use the `ps -e` command to show the number of running threads. Only partial output is shown:

```
# ps -e |grep nfs
... nfsd
... nfsd
...
```

The number of `nfsd` threads to run is defined in the `/proc/fs/nfsd/threads` file. In this example, 8 `nfsd` threads are specified:

```
# cat /proc/fs/nfsd/threads
8
```

Starting the `nfs-server` service also starts the RPC processes. You can use the `ps -e` command to display the names of the RPC processes. Only partial output is shown:

```
# ps -e |grep rpc
... rpciod
... rpcbind
... rpc.statd
... rpc.mountd
... rpc.idmapd
... rpc.rquotad
```

rpc.statd

This process implements the Network Status Monitor (NSM) RPC protocol, which notifies NFS clients when an NFS server is restarted without being gracefully brought down. This is not used with NFSv4.

rpc.mountd

This is the NFS mount daemon that implements the server side of the mount requests from NFSv3 clients. It checks that the requested NFS share is currently exported by the NFS server, and that the client is allowed to access it. For NFSv4, the `rpc.mountd` daemon is required only on the NFS server to set up the exports.

rpc.idmapd

This provides NFSv4 client and server upcalls, which map between on-the-wire NFSv4 names (which are strings in the form of `user@domain`) and local UIDs and GIDs. For `idmapd` to function with NFSv4, `/etc/idmapd.conf` must be configured. This service is required for use with NFSv4, although not when all hosts share the same DNS domain name.

rpc.rquotad

This process provides user quota information for remote users. It is started automatically by the `nfs` service and does not require user configuration. The results are used by the `quota` command to display user quotas for remote file systems and by the `edquota` command to set quotas on remote file systems.

lockd

This is a kernel thread that runs on both clients and servers. It implements the Network Lock Manager (NLM) protocol, which allows NFSv3 clients to lock files on the server. It is started automatically whenever the NFS server is run and whenever an NFS file system is mounted.

nfslock

Starting this service starts the RPC processes that allow NFS clients to lock files on the server.

NFS Server Configuration

- Install the `nfs-utils` package.
- Configuration file for the NFS server is `/etc/exports`.
  - It contains a list of exported directory hierarchies that remote systems can mount.
- The format of `/etc/exports` entries is:

```
dir client1(options) [client2(options)...]
```

- Example:

```
/export/directory 192.0.2.102(rw,async)
```

- Client options include (defaults are listed first):
  - `ro / rw`
  - `sync / async`
  - `wdelay / no_wdelay`
  - `no_all_squash / all_squash`

To begin configuring a system as an NFS server, install the `nfs-utils` package:

```
# yum install nfs-utils
```

The main configuration file for the NFS server is `/etc/exports`. This file stores a list of exported directory hierarchies that remote systems can mount. The format for entries is:

```
export-point client1(options) [client2(options) … ]
```

The `export-point` is the absolute path name of the directory hierarchy to be exported. One or more client systems, each with specific options, can mount `export-point`. There are no spaces between the client attribute and the open bracket. When no client options are specified, the following default settings apply:

- **`ro`:** Read-only. Client hosts cannot change the data shared on the file system. To allow client hosts to make changes to the file system, specify the `rw` (read/write) option.
- **`sync`:** The NFS server replies to requests only after changes made by previous requests are written to disk. `async` specifies that the server does not have to wait.
- **`wdelay`:** The NFS server delays committing write requests when it suspects another write request is imminent. To disable the delay, use the `no_wdelay` option. `no_wdelay` is available only if the default `sync` option is also specified.

- **`root_squash`:** Prevents `root` users connected remotely from having `root` privileges, effectively "squashing" the power of the remote `root` user. Requests appear to come from the user `nfsnobody`, an unprivileged user on the local system, or as specified by `anonuid`. To disable root squashing, specify the `no_root_squash` option.
- **`no_all_squash`:** Does not change the mapping of remote users. To squash every remote user (including `root`), use the `all_squash` option.

To specify the user ID (UID) and group ID (GID) that the NFS server assigns to remote users, use the `anonuid` and `anongid` options as follows:

```
export-point client(anonuid=uid,anongid=gid)
```

The `anonuid` and `anongid` options allow you to create a special user and group account for remote NFS users to share. By default, access control lists (ACLs) are supported by NFS. To disable this feature, specify the `no_acl` option when exporting the file system.

You can use wildcard characters, such as (`*`) and (`?`) in client names. You can also export directories to all hosts on an IP network. To do this, specify an IP address and netmask pair as address/netmask. Either of the following forms is valid:

- `192.168.1.0/24`
- `192.168.1.0/255.255.255.0`

Other client options exist. Refer to `man exports` for descriptions of all options.

`/etc/exports` Examples

In the following example, a client system with the IP address of `192.0.2.102` can mount the `/export/directory` with read/write permissions. All writes to the disk are asynchronous:

```
/export/directory 192.0.2.102(rw,async)
```

The following example exports the `/exports/apps` directory to all clients, converts all connecting users to the local anonymous `nfsnobody` user, and makes the directory read-only:

```
/exports/apps *(all_squash, ro)
```

The following example exports the `/spreadsheets/proj1` directory with read-only permissions to all clients on the `192.168.1.0` subnet, and read/write permissions to the client system named `mgmtpc`:

```
/spreadsheets/proj1 192.168.1.0/24(ro) mgmtpc(rw)
```

Starting the NFS Service

- Start `rpcbind` before starting the `nfs` services:

```
# systemctl start rpcbind
# systemctl start nfs
# systemctl start nfslock
```

- Use the `systemctl enable` command to automatically start the services at boot time.

```
# systemctl enable nfs-server
```

- Specify configuration options and arguments in `/etc/sysconfig/nfs`.
- To display exported file systems:

```
# showmount -e
```


The `rpcbind` service must be started before starting `nfs`. The following command checks if the `rpcbind` service is enabled and running.

```
# systemctl status rpcbind
```

If the `rpcbind` service is running, the `nfs` service can be started. Restart `nfs` after making any configuration changes in `/etc/exports` or run the `exportfs -a` command.

```
# systemctl start nfs
```

Check if the `nfslock` service is enabled and running. Starting this service starts the RPC processes that allow NFS clients to lock files on the server.

```
# systemctl status nfslock
```

Use the `systemctl enable` command to automatically start the services at boot time. Use the full name of `nfs-server` when enabling the NFS service.

```
# systemctl enable nfs-server
```

Specify configuration options and arguments by placing them in `/etc/sysconfig/nfs`. This file contains several comments to assist you in specifying options are arguments.

Use the `showmount -e` command to display exported file systems:

```
# showmount -e
```

`exportfs` Utility

- `exportfs` exports or unexports directories, and is run from the command line.
  - No need to change `/etc/exports`
  - No need to restart NFS service
- Syntax for the command:

```
exportfs [options] [client:dir ...]
```

- Example:

```
# exportfs -i -o rw *:/Dev
```

- This example does the following:
  - Exports `/Dev` to all clients systems (`*`)
  - Allows read/write permission (`-o rw`)
  - Ignores `/etc/exports` entries (`-i`)

ORACLE

You can also configure an NFS server from the command line by using `exportfs`. This command allows the `root` user to selectively export or unexport directories without changing `/etc/exports` and without restarting the NFS service. The syntax for the command is:

```
exportfs [options] [client:dir ...]
```

The `client` argument is the name of the client system that `dir` is exported to. The `dir` argument is the absolute path name of the directory being exported. The following is a list of some of the options:

- **`-r`:** Re-export the entries in `/etc/exports` and synchronize `/var/lib/nfs/etab` with `/etc/exports`. The `/var/lib/nfs/etab` file is the master export table. `rpc.mountd` reads this file when a client sends an NFS `mount` command.
- **`-a`:** Export the entries in `/etc/exports` but do not synchronize `/var/lib/nfs/etab`. Run `exportfs -a` after making any configuration changes.
- **`-i`:** Ignore the entries in `/etc/exports` and use only command-line arguments.
- **`-u`:** Unexport one or more directories.
- **`-o`:** Specify client options as specified in `/etc/exports`.

NFS Client Configuration

- Install the `nfs-utils` package.
- Use the `mount` command to mount exported file systems.
- Syntax for the command:

```
mount -t nfs -o options host:/remote/export
   /local/directory
```

- Example:

```
# mount -t nfs -o ro,nosuid abc:/home /abc_home
```

- This example does the following:
  - It mounts `/home` from remote host `abc` on local mount point `/abc_home`.
  - File system is mounted read-only and users are prevented from running a `setuid` program (`-o ro,nosuid` options).
- Update `/etc/fstab` to mount NFS shares at boot time.

To begin configuring a system as an NFS client, install the `nfs-utils` package:

```
# yum install nfs-utils
```

Use the `mount` command to mount exported file systems (NFS shares) on the client side. Syntax for the command is:

```
mount -t nfs -o options host:/remote/export /local/directory
```

The following are descriptions of the arguments:

- **`-t nfs`**: Indicates that the file system type is `nfs`. With this option, `mount` uses NFSv4 if the server supports it; otherwise, it uses NFSv3.
- **`-o options`**: A comma-delimited list of mount options
- **`host:/remote/export`**: The host name exporting the file system, followed by a colon, followed by the absolute path name of the NFS share
- **`/local/directory`**: The mount point on the client system

For example, to mount the `/home` directory exported from host `abc` with read-only permissions (`ro` option) on local mount point `/abc_home`, and prevent remote users from gaining higher privileges by running a `setuid` program (`nosuid` option):

```
# mount -t nfs -o ro,nosuid abc:/home /abc_home
```

For a list of options used for NFS mounts, see the MOUNT OPTIONS section of the `nfs(5)` man pages. For a list of client mount options, see the FILESYSTEM INDEPENDENT MOUNT OPTIONS section of the `mount(8)` man pages.

To mount NFS shares at boot time, add entries to the file system mount table, `/etc/fstab`. Entries are in the following format:

```
server:/exported-filesystem local_mount_point nfs options 0 0
```

For example, the `/etc/fstab` entry that replicates the `mount` command on the previous page is:

```
abc:/home  /abc_home  nfs  ro,nosuid  0    0
```

The `df` command displays mounted file systems, including NFS-mounted file systems. For NFS mounts, the "File system" column displays the `server:/exported-filesystem` information. Use the `-T` option to include a "Type" column:

```
# df -hT
Filesystem   Type  Size  Used  Avail  Use%  Mounted on
...
host03:/Dev  nfs4  976M  2.5M   907M    1%  /remote_dev
```

Automounting File Systems

- Remote file systems are mounted only when accessed.
- Install the `autofs` package.
  - `autofs`: Kernel module
  - `automount`: Userspace daemon
- The main configuration file is `/etc/auto.master`.
- Format of `/etc/auto.master` entries:

```
/key        map-file        [options]
```

- Example:

```
# cat /etc/auto.master
/-  auto.direct
/misc    /etc/auto.misc
/net  -hosts
+auto.master
```

ORACLE

Automounting is an alternative to creating NFS mount entries in `/etc/fstab` or using the `mount` command from the command line to mount NFS shares. Automounting mounts remote file systems when they are accessed, rather than maintaining these remote mounts at all times. When the remote file systems are inactive, they are unmounted. This frees up system resources and improves overall system performance.

To implement automounting, first install the `autofs` package:

```
# yum install autofs
```

To start the `autofs` service:

```
# systemctl start autofs
```

The main configuration file, known as the master map file, is `/etc/auto.master`. This file lists mount points, known as keys, and corresponding map files that indicate which remote file systems can be mounted on the key. The format for entries in `/etc/auto.master` is:

```
/key        map-file          [options]
```

Automounting supports direct maps, indirect maps, and host maps. Direct maps use a special key, `/-`, in `/etc/auto.master`. Indirect maps specify a relative path name in their map files. Host maps use a special map, `-hosts`, in the `/etc/auto.master` file. Entries preceded with a plus sign (`+`) include a map from its source as if it were present in the master map.

Direct Maps

- Direct maps always have a key of /- in /etc/auto.master. Example:

```
/-      auto.direct
```

- Sample entry in auto.direct map file:

```
/usr/man        -ro,soft         host01:/usr/man
```

- Format of direct and indirect map files:

```
key                  [options]          location
```

  – The "key" is the absolute path name of the mount point for direct mounts.
  – The "location" is an exported NFS file system, or a local file system of any supported file system type.
  – Mount options included in map files override options specified in the master map file, /etc/auto.master.

ORACLE

The following entry in the /etc/auto.master file is an example of a direct map:

```
/-          auto.direct
```

Direct maps always have a key of /-. The map file in this example is auto.direct. With direct maps, the map file contains the absolute path name of the directory to be mounted. The following is an example of the contents of the auto.direct file:

```
/usr/man    -ro,soft    host01:/usr/man
```

This entry mounts the file system /usr/man from the server host01 on the local /usr/man mount point. automount creates the /usr/man directory if it does not already exist. If /usr/man does exist and is not empty, the mounted file system hides the local existing file system.

Direct map files and indirect map files have the following format:

```
key    [options]    location
```

The key can be a single directory name for an indirect map or the absolute path name of the mount point for direct mounts. Mount options can be included in map files. Any options specified in map files override options specified in the master map file. The location is the exported NFS file system, a local file system, or any other supported file system type.

Indirect Maps

- Example of an indirect map entry in `/etc/auto.master`:

```
/misc      /etc/auto.misc
```

- Sample entries in `auto.misc` map file:

```
xyz     -fstype=nfs                             host01:/xyz
cd      -fstype=iso9600,ro,nosuid,nodev         :/dev/cdrom
abc     -fstype=ext3                            :/dev/hda1
```

- The "`key`" in the indirect map file is relative to the `autofs` mount point, `/misc`, defined in `/etc/auto.master`.
- For example:
  - `cd /misc/xyz` mounts the `/xyz` directory from machine `host01` locally on `/misc/xyz`.
  - `cd /misc/abc` mounts the ext3 file system on local device `/dev/hda1` on `/misc/abc`.

The following entry in the `/etc/auto.master` file is an example of an indirect map:

```
/misc      /etc/auto.misc
```

Indirect maps are more common than direct maps. The following is an example of an indirect map file named `/etc/auto.misc`:

```
# cat /etc/auto.misc
xyz       -fstype=nfs                         host01:/xyz
cd        -fstype=iso9600,ro,nosuid,nodev     :/dev/cdrom
abc       -fstype=ext3                        :/dev/hda1
kernel    -ro,soft,intr             ftp.kernel.org:/pub/linux
windoz    -fstype=smbfs                       ://windoz/c
```

The key field is relative to the actual location of the `autofs` mount point, `/misc`, from the master map file, `/etc/auto.master`.

For example, entering the `cd /misc/xyz` command mounts the `/xyz` directory from machine `host01` locally on `/misc/xyz`. Only the `/misc` mount point needs to exist on the local machine. For indirect maps, the key is created when the file system is accessed and then removed when the file system is unmounted.

The second and third entries are examples of automounting local file systems:

```
cd        -fstype=iso9600,ro,nosuid,nodev       :/dev/cdrom
abc       -fstype=ext3                          :/dev/hda1
```

The location field is the local file system path preceded with a colon (`:`). Entering the `ls /misc/cd` command would display the contents of the `iso` file on the `cdrom`. Entering the `ls /misc/abc` command would display the contents of the ext3 file system on the `hda1` device.

The fourth line is an NFS mount (excluding the `–fstype` option defaults to NFS), which mounts the `/pub/kernel` directory from `ftp.kernel.org` on local mount point `/misc/kernel`:

```
kernel    -ro,soft,intr              ftp.kernel.org:/pub/linux
```

The last line mounts a share exported from a Windows machine on `/misc/windoz`:

```
windoz    -fstype=smbfs              ://windoz/c
```

Host Maps

- Example of a host map entry in /etc/auto.master:

```
/net    -hosts
```

- The automount daemon creates a subdirectory under the "key" directory for every server listed in /etc/hosts.
- For example, entering the following command mounts all exports from host03 over the /net/host03 directory:

```
# cd /net/host03
```

- Entering the ls command list exported file systems from host03

The following entry in the /etc/auto.master file is an example of a host map:

```
/net        -hosts
```

When -hosts is given as the map, the automount daemon creates a subdirectory under the "key" directory, /net, for every server listed in the /etc/hosts file.

For example, entering the following command mounts all exports from host03 over the /net/host03 directory:

```
# cd /net/host03
```

All exports are mounted with the "no-suid,nodev,intr" options by default.

Introduction to `vsftpd`

- `vsftpd` allows a system to function as an FTP server.
- `vsftpd` includes the following configuration files and directories:
  – `/etc/vsftpd/vsftpd.conf`
  – `/etc/vsftpd/ftpusers`
  – `/etc/vsftpd/user_list`
  – `/var/ftp`
- To start the service:

```
# systemctl start vsftpd
```

- To start automatically at boot time:

```
# systemctl enable vsftpd
```


ORACLE

File Transfer Protocol (FTP) is a commonly used method of downloading and uploading files between systems on a network. FTP sites are typically public sites that allow anonymous users to log in and download software and documentation without needing a user account on the remote system.

The FTP server daemon included with Oracle Linux is called “very secure FTP,” or `vsftpd`. To install the `vsftpd` package:

```
# yum install vsftpd
```

The following configuration files are installed with the package:

- **`/etc/vsftpd/vsftpd.conf`**: The main configuration file for `vsftpd`
- **`/etc/vsftpd/ftpusers`**: A list of users not allowed to log in to `vsftpd`
- **`/etc/vsftpd/user_list`**: This file contains users who are denied access when the `userlist_deny` directive is set to `YES` (default) in `/etc/vsftpd/vsftpd.conf` or users who are allowed access when `userlist_deny` is set to `NO`.
- **`/var/ftp`**: The directory containing files served by `vsftpd`. It also contains the `/var/ftp/pub` directory for anonymous users.

To start the `vsftpd` service:

```
# systemctl start vsftpd
```

`vsftpd` Configuration Options

- Local and anonymous users can download files by default.
  - `local_enable=YES`
  - `anonymous_enable=YES`
- Users can upload files by default, too.
  - `write_enable=YES`
- Additional configuration parameters in `/etc/vsftpd/vsftpd.conf` include:
  - `userlist_enable`
  - `userlist_deny`
  - `no_anon_password`
  - `xferlog_enable`
  - `xferlog_file`

The `vsftpd` service allows local and anonymous users to log in without any additional configuration. When a user logs in, they can download files from the `/var/ftp` directory on the `vsftpd` server and upload files by default.

These and other options are configured in `/etc/vsftpd/vsftpd.conf`. The following lists some of the more common configuration parameters:

- **`userlist_enable`:** This setting causes `vsftpd` to read `/etc/vsftpd/user_list` and use that as a list of users to allow or not allow on the server.
- **`userlist_deny`:** When set to `yes`, `vsftpd` blocks all users in the `user_list`. When set to `no`, it allows only users in the `user_list`.
- **`local_enable`:** This setting allows users in `/etc/passwd` to log in with their accounts.
- **`anonymous_enable`:** This setting allows anonymous connections to the server.
- **`no_anon_password`:** This setting allows anonymous connections without a password (otherwise, users must provide an email address as a password).
- **`write_enable`:** When set to `yes`, this setting allows users to upload files to the server and create directories.
- **`anon_mkdir_write_enable`:** When set to `yes`, this setting allows anonymous users to create directories.

- **`anon_other_write_enable`**: When set to `yes`, this setting allows anonymous users to make other changes to the file system, such as deleting, renaming, and modifying existing files.
- **`anon_upload_enable`**: This setting allows anonymous users to upload files to the server.
- **`ascii_download_enable`**: This setting allows conversion of text files transferred from the server to other operating systems. This can be a good idea if you are transferring text files from UNIX systems to Mac OS or Windows.
- **`ascii_upload_enable`**: This setting allows conversion of text files uploaded to the server.
- **`xferlog_enable`**: This setting activates logging of uploads and downloads.
- **`xferlog_file`**: This setting names the upload/download log file. The default is `/var/log/vsftpd.log`.

Quiz

Which of the following statements are true?

a. NFS allows Linux clients to mount exported file systems from remote Linux systems.

b. Automounter allows NFS shares to be automatically mounted.

c. The `vsftpd` daemon enables a system to be configured as an FTP server.

Summary

In this lesson, you should have learned how to:

- Describe NFS
- Configure NFS server and client
- Describe the `exportfs` utility
- Describe and configure automounter
- Describe and configure `vsftpd`

Practice 16: Overview

The practices for this lesson cover the following:

- Configuring an NFS server and an NFS client
- Using automounter
- Configuring an FTP server
- Downloading a file from an FTP server

17

OpenSSH

Objectives

After completing this lesson, you should be able to:

- Describe OpenSSH
- Describe OpenSSH configuration files
- Configure OpenSSH server and client
- Use the `ssh` command
- Use the `scp` command
- Use the `sftp` command
- Use the `ssh-keygen` command
- Use `ssh-agent` and `ssh-add`

Introduction to OpenSSH

OpenSSH:

- Is a suite of secure network connectivity tools:
  - `ssh`: Secure shell command
  - `scp`: Secure copy command
  - `sftp`: Secure file transfer protocol (FTP) command
  - `sshd`: The OpenSSH daemon
  - `ssh-keygen`: Creates ECDSA or RSA authentication keys
- Is a secure alternative to `telnet`, `rcp`, `rsh`, `rlogin`, and `ftp`
- Encrypts all communication between the client and server
- Supports both the SSH1 and SSH2 protocols
- Provides X11 forwarding and port forwarding

ORACLE

OpenSSH (Secure Shell) is a suite of network connectivity tools that provides secure communications between systems. OpenSSH tools include the following:

- **`ssh`:** Secure shell logs on or runs a command on a remote system
- **`scp`:** Secure copy
- **`sftp`:** Secure `ftp` (file transfer protocol)
- **`sshd`:** The OpenSSH daemon
- **`ssh-keygen`:** Creates ECDSA or RSA host/user authentication keys:
  - ECDSA (Elliptic Curve Digital Signature Algorithm)
  - RSA is named for the designers Rivest, Shamir, and Adleman.

Unlike other tools such as `telnet`, `rcp`, `rsh`, `rlogin`, and `ftp`, OpenSSH tools encrypt all communication between the client and server systems, including passwords. Each network packet is encrypted by using a key known only by the local and remote systems.

OpenSSH supports both versions of SSH, SSH protocol version 1 (SSH1) and SSH protocol version 2 (SSH2). Additionally, OpenSSH provides a secure means to use graphical applications over a network by using X11 forwarding. It also provides a way to secure otherwise insecure TCP/IP protocols by using port forwarding.

OpenSSH Configuration Files

- Global files are stored in the `/etc/ssh` directory.
- User files are stored in the `~/.ssh` directory.
- Global files include:
  - `ssh_config`: The default OpenSSH client configuration file
  - `sshd_config`: The configuration file for the `sshd` daemon
  - Various ECDSA and RSA public and private key files
  - PAM configuration file: `/etc/pam.d/sshd`
  - Configuration file for `sshd`: `/etc/sysconfig/sshd`
- User files include:
  - `config`: Overrides global `ssh_config` file
  - `known_hosts`: Contains host keys of SSH servers accessed by the user
  - Various user ECDSA and RSA public and private key files

OpenSSH clients and servers have several configuration files. Global configuration files are stored in the `/etc/ssh` directory. User configuration files are stored in an `.ssh` directory in user home directories (`~/.ssh`).

/etc/ssh: Global Files

The following are brief descriptions of the global configuration files:

- **moduli**: Contains key exchange information used to establish a secure connection
- **ssh_config**: The default OpenSSH client configuration file. Entries are overridden by a user's `~/.ssh/config` file.
- **sshd_config**: The configuration file for the `sshd` daemon
- **ssh_host_ecdsa_key**: The ECDSA private key used by the `sshd` daemon
- **ssh_host_ecdsa_key.pub**: The ECDSA public key used by the `sshd` daemon
- **ssh_host_key**: The RSA private key for version SSH1
- **ssh_host_key.pub**: The RSA public key for version SSH1
- **ssh_host_rsa_key**: The RSA private key for version SSH2
- **ssh_host_rsa_key.pub**: The RSA public key for version SSH2

There is also a PAM configuration file for the `sshd` daemon, `/etc/pam.d/sshd`, and a configuration file for the `sshd` service, `/etc/sysconfig/sshd`.

`~/.ssh`: User Files

OpenSSH creates the `~/.ssh` directory and the `known_hosts` file automatically when you connect to a remote system. The following are brief descriptions of the user-specific configuration files:

- **`authorized_keys`**: Contains a list of authorized public keys for SSH servers. The server authenticates the client by checking its signed public key within this file.
- **`id_ecdsa`**: The ECDSA private key of the user
- **`id_ecdsa.pub`**: The ECDSA public key of the user
- **`id_rsa`**: The RSA private key for version SSH2
- **`id_rsa.pub`**: The RSA public key for version SSH2
- **`identity`**: The RSA private key for version SSH1
- **`identity.pub`**: The RSA public key for version SSH1
- **`known_hosts`**: Contains host keys of SSH servers accessed by the user. OpenSSH automatically adds entries each time the user connects to a new server.

OpenSSH Configuration

- To configure an OpenSSH server:
  - The following packages are installed by default:
    - `openssh`
    - `openssh-server`
  - Start the `sshd` daemon:

```
# systemctl start sshd
```

  - Configure the service to start at boot time:

```
# systemctl enable sshd
```

- To configure an OpenSSH client:
  - The following packages are installed by default:
    - `openssh`
    - `openssh-clients`
  - There are no services to start on the client.

OpenSSH Server

To begin configuring a system as an OpenSSH server, install the following packages (these are installed by default):

```
# yum install openssh
# yum install openssh-server
```

Start the `sshd` daemon:

```
# systemctl start sshd
```

Use the `systemctl` command to automatically start the `sshd` service at boot time:

```
# systemctl enable sshd
```

OpenSSH Client

To configure a system as an OpenSSH client, install the following packages (these are installed by default):

```
# yum install openssh
# yum install openssh-clients
```

There are no services to start for OpenSSH clients.

Using OpenSSH Utilities

- All OpenSSH utilities require a remote user account.
- The first time you connect to an OpenSSH server, the OpenSSH client prompts you to confirm that you are connected to the correct system:

```
$ ssh host03
The authenticity of host 'host03 (192.0.2.103)' can't be
   established. ECDSA key fingerprint is …
Are you sure you want to continue connecting (yes/no)?
   Yes
Warning: Permanently added 'host03,192.0,2,103' (ECDSA)
   to the list of known hosts.
```

- The user's `~/.ssh/known_hosts` file is updated.

ORACLE

All of the OpenSSH tools require that you have a user account on the remote system. Each time you attempt to connect to a remote system, you must provide a username and password for the remote system.

When you connect to an OpenSSH server for the first time, the OpenSSH client prompts you to confirm that you are connected to the correct system. The following example uses the `ssh` command to connect to a remote host named `host03`:

```
$ ssh host03
The authenticity of host 'host03 (192.0.2.103)' can't be
established. ECDSA key fingerprint is ...
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'host03,192.0,2,103' (ECDSA) to the
list of known hosts.
```

Host validation is one of OpenSSH's major features. The command checks to make sure that you are connecting to the host that you think you are connecting to. When you enter `yes`, the client appends the server's public host key to the user's `~/.ssh/known_hosts` file, creating the `~/.ssh` directory if necessary. The next time you connect to the remote server, the client compares this key to the one the server supplies. If the keys match, you are not asked if you want to continue connecting.

If someone tries to trick you into logging in to their machine so that they can sniff your SSH session, you will receive a warning similar to the following:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The RSA host key for ... has changed,
and the key for the according IP address ...
is unchanged. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
Offending key for IP in /home/<user>/.ssh/known_hosts:10
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is ...
Please contact your system administrator.
Add correct host key in /home/<user>/.ssh/known_hosts to get rid
of this message.
Offending key in /home/<user>/.ssh/known_hosts:53
RSA host key for ... has changed and you have requested strict
checking.
Host key verification failed.
```

If you ever get a warning like this, stop and determine whether there is a reason for the remote server’s host key to change (such as if SSH was upgraded or the server itself was upgraded). If there is no good reason for the host key to change, do not try to connect to that machine until you have resolved the situation.

Using the `ssh` Command

- The `ssh` command:
  - Allows you to connect to a remote system
  - Allows you to execute a command on a remote system
- The format of the command is:

```
ssh [options] [user@]host [command]
```

- Examples:

```
$ ssh host03
$ ssh root@host03
$ ssh host03 ls
```


The `ssh` command allows you to connect to a remote system, or to execute a command on a remote system.

The format of the `ssh` command is:

```
ssh [options] [user@]host [command]
```

The *`host`* argument is the name of the OpenSSH server that you want to connect to, and is the only required argument. For example, to connect to a remote host named `host03`, enter only the following:

```
$ ssh host03
```

This command attempts to connect to the remote host with the same username that you are logged on as on the local system. You are prompted for only the remote user's password. To connect to a remote host as a different user, provide the *`user@`* argument:

```
$ ssh root@host03
```

To execute a command on a remote system, include the command as an argument. `ssh` logs you in, executes the command, and then closes the connection, for example:

```
$ ssh host03 ls
```

Using the scp Command

- Use `scp` to copy files or directories to or from a remote system.
- To copy to a remote system, the format is:

```
scp [options] local-file [user@]to-host[:remote-file]
```

- Examples:

```
$ scp test host03
$ scp test host03:new_test
```

- To copy from a remote system, the format is:

```
scp [options] [user@]from-host:remote-file local-file
```

- Examples:

```
$ scp host03:new_test .
$ scp host03:new_test newer_test
```


The `scp` command allows you to copy files or directories (use the `-r` option to copy directories) between remote systems. A connection is established, files are copied, and the connection closes.

To copy a file to a remote system (upload), the format of the `scp` command is:

```
scp [options] local-file [user@]to-host[:remote-file]
```

For example, to copy a file named `test` to the remote user's home directory on `host03`:

```
$ scp test host03
```

To copy the same file to the same location but rename it to `new_test`:

```
$ scp test host03:new_test
```

To copy a file from a remote system (download), the format of the `scp` command is:

```
scp [options] [user@]from-host:remote-file local-file
```

For example, to copy a file named `new_test` from user's home directory on remote `host03`:

```
$ scp host03:new_test .
```

To copy a file named `new_test` from user's home directory on remote `host03` and rename it to `newer_test`:

```
$ scp host03:new_test newer_test
```

Using the `sftp` Command

- `sftp` is a secure alternative to, and is functionally the same as, `ftp`.
- The format to connect to a remote system is:

```
sftp [options] [user@]host
```

- Example:

```
$ sftp host03
```

- You are presented with the `sftp>` prompt after connecting:

```
sftp>
```

- Enter `help` or `?` to display a list of `sftp` commands.
- To upload a file (copy to remote system):

```
sftp> put filename
```

- To download a file (copy from a remote system):

```
sftp> get filename
```

ORACLE

The `sftp` command is a secure alternative to `ftp` and is functionally the same as `ftp`. Use `sftp` instead of `ftp` when logging on to a server that is running the OpenSSH daemon, `sshd`.

The format to connect to a remote system is:

```
sftp [options] [user@]host
```

The following example assumes that you are logged on to your local system as user `oracle` and are connecting to a remote system named `host03`:

```
$ sftp host03
Connecting to host03...
oracle@host03's password:
sftp>
```

After providing the correct password, you are presented with an `sftp>` prompt as shown. Enter `help` or `?` to display a list of available commands. The following example uploads a file, or copies the file from the local system to the remote system:

```
sftp> put newer
```

Enter `exit`, `quit`, or `bye` to close the connection and exit `sftp`.

Using the `ssh-keygen` Command

- The `ssh-keygen` command generates authentication key pairs.
- Use the `-t` option to specify the key type. Example:

```
$ ssh-keygen -t rsa
```

- `ssh-keygen` generates two keys:
  - Private key
  - Public key
- Specify a passphrase to encrypt the private part of the key.
- To allow remote connectivity without supplying a password:
  1. Copy the public key to `~/.ssh` on the remote system.
  2. Do one of the following:
     - Rename the public key file name to `authorized_keys`.
     - Append the public key to the `authorized_keys` file on the remote system to allow connection from multiple clients.

Use the `ssh-keygen` command to generate a public/private authentication key pair. Authentication keys allow a user to connect to a remote system without supplying a password. Keys must be generated for each user separately. If you generate key pairs as the `root` user, only the `root` can use the keys.

The following example creates the public and private parts of an RSA key:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:...
The key's randomart image is:...
```

Use the `-t` option to specify the type of key to create. Possible values are "`rsa1`" for protocol version 1, and "`dsa`", "`ecdsa`", or "`rsa`" for protocol version 2.

You have the option of specifying a passphrase to encrypt the private part of the key. If you encrypt your personal key, you must supply the passphrase each time you use the key. This prevents an attacker, who has access to your private key and can impersonate you and access all the computers you have access to, from being able to do so. The attacker still needs to supply the passphrase.

The `ssh-key` command in the example generated two keys in the `~/.ssh` directory:

```
$ ls ~/.ssh
id_rsa
id_rsa.pub
```

To log on to, or copy files to, a remote system without supplying a password, copy the public key (`~/.ssh/id_rsa.pub` in this example) to `~/.ssh/authorized_keys` on the remote system. Set the remote `~/.ssh` directory permissions to `700`. You can then use the `ssh` or `scp` tools to access the remote system without supplying a password.

To allow multiple connections, append the public key to the `authorized_keys` file on the remote system instead of copying it. The following example appends the public key:

```
$ cat id_rsa.pub >> authorized_keys
```

You can improve system security even further by disabling the standard password authentication, and enforcing the key-based authentication. To do so, set the `PasswordAuthentication` option to `no` in the `/etc/ssh/sshd_config` configuration file as follows:

```
PasswordAuthentication no
```

This disallows users whose keys are not in the `authorized_keys` file of the specific user on the server to connect via `ssh`. The connection is denied and the following message appears:

```
$ ssh host01
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Setting the `PasswordAuthentication` option to `yes`, which is the default, permits a user to use a password for authentication.

Using `ssh-agent`

- `ssh-agent` is an authentication agent that handles passwords for SSH private keys.
  – Use `ssh-keygen` to generate authentication key pairs.
  – Provide a passphrase, for example "`password`", when creating the key pairs.
  – Copy the public key to `~/.ssh/authorized_keys` on the remote system.
- To use `ssh-agent`:

```
$ exec ssh-agent $SHELL
```

- Use `ssh-add` to add the keys:

```
$ ssh-add
Enter passphrase for /home/oracle/.ssh/id_rsa: password
Identity added: /home/oracle/.ssh/id_rsa ...
```


ORACLE

The `ssh-agent` program is an authentication agent that handles passwords for SSH private keys. Use `ssh-add` to add the keys to the list maintained by `ssh-agent`. After you add a private key password to `ssh-agent`, you do not need to enter it each time you connect to a remote host with your public key.

Use the **`ssh-keygen`** command to generate authentication key pairs as described in the previous slide. Provide a passphrase, for example "`password`", when creating the key pairs. Copy the public key to `~/.ssh/authorized_keys` on the remote system as described in the previous slide.

To add the private key password to `ssh-agent`, enter the following command:

```
$ exec ssh-agent $SHELL
```

The next step is to use the `ssh-add` command to add the key.

```
$ ssh-add
Enter passphrase for /home/oracle/.ssh/id_rsa: password
Identity added: /home/oracle/.ssh/id_rsa ...
```

In this example, the passphrase is remembered for only the current login session and is forgotten when you log out.

Quiz

OpenSSH connectivity tools encrypt all network traffic, including passwords.

a. True

b. False

Summary

In this lesson, you should have learned how to:

- Describe OpenSSH
- Describe OpenSSH configuration files
- Configure OpenSSH server and client
- Use the `ssh` command
- Use the `scp` command
- Use the `sftp` command
- Use the `ssh-keygen` command
- Use `ssh-agent` and `ssh-add`

Practice 17: Overview

The practices for this lesson cover the following:

- Connecting to a remote server by using `ssh`
- Configuring OpenSSH to connect without a password

18

Security Administration

Objectives

After completing this lesson, you should be able to:

- Describe the `chroot` jail
- Use the `chroot` utility
- Describe packet-filtering firewalls
- Describe `firewalld`
- Configure `firewalld` packet filters
- Describe `iptables`
- Configure `iptables` packet filters
- Describe TCP wrappers
- Configure TCP wrappers

Several methods of securing your computer system are covered in this lesson.

`chroot` Jail

- The `chroot` utility changes the apparent root directory.
  - A program (process) runs with a root directory other than `/`.
  - The artificial root directory is called a `chroot` jail.
- To the process, it appears that it is running in the root directory.
- A `chroot` jail limits the directory access of a potential attacker.
- A `chroot` jail is not intended to:
  - Defend against intentional tampering by privileged (`root`) users
  - Be used to block low-level access to system devices by privileged users
- The `chroot` jail directory must be populated with all files required by the process at their expected locations.

As the name implies, a `chroot` operation changes the apparent root directory for a running process and its children. It allows you to run a program (process) with a root directory other than `/`. The program cannot see or access files outside the designated directory tree.

For example, you can run a program and specify its root directory as `/home/oracle/jail`. In this case, the program's root directory is actually `/home/oracle/jail`. The program would not be aware of, or able to access, any files above this directory in the hierarchy.

This artificial root directory is called a `chroot` jail. Its purpose is to limit the directory access of a potential attacker. The `chroot` jail locks down a given process and any user ID it is using so that the user sees only the directory that the process is running in. To the process, it appears that it is running in the root directory.

The `chroot` mechanism is not intended to defend against intentional tampering by privileged (`root`) users. It is also not intended by itself to be used to block low-level access to system devices by privileged users. A `chroot root` user can still create device nodes and mount the file systems on them.

For a `chroot` process to successfully start, the `chroot` directory must be populated with all required program files, configuration files, device nodes, and shared libraries at their expected locations.

`chroot` Utility

- To use a `chroot` jail, use the following command:

```
# chroot new_root [command]
```

- The *new_root* directory becomes the artificial root.
- `chroot` changes to *new_root* and runs the optional command.
  – Alternatively, it runs the `SHELL` variable if the command is omitted.
- The command fails unless the necessary files are copied into the *new_root* directory before running `chroot`:

```
# chroot /home/oracle/jail
chroot: failed to run command '/bin/bash': No such file
   or directory
```


To use a `chroot` jail, use the following command (*new_root* must be an existing directory):

```
# chroot new_root [command]
```

The *new_root* directory becomes the artificial root directory. `chroot` changes to *new_root* and runs the optional command. Without specifying a command as an argument, `chroot` changes to *new_root* and runs the value of the `SHELL` environment variable or `/bin/sh` if `SHELL` is not set.

For example, assuming `SHELL` is set to `/bin/bash`, and the `/home/oracle/jail` directory exists, running the `chroot` command results in the following:

```
# chroot /home/oracle/jail
chroot: failed to run command '/bin/bash': No such file or
directory
```

The `/home/oracle/jail` directory takes the name of `/`. `chroot` cannot find the `/bin/bash` within this `chroot` jail and returns the error message.

To implement a `chroot` jail, create the new root directory structure and copy all the necessary files into this new root directory before running the `chroot` command.

Implementing a `chroot` Jail

- Make the necessary directories and copy all required files into these directories:

```
$ mkdir /home/oracle/jail/bin
$ cp /bin/bash /home/oracle/jail/bin
```

- Determine whether any shared libraries are required:

```
$ ldd /bin/bash
```

- Create the `lib` (or `lib64`) directory and copy all required shared libraries into this directory:

```
$ mkdir /home/oracle/jail/lib64
$ cp /lib64/{...} /home/oracle/jail/lib64
```

- Execute the `chroot` command (as `root`):

```
# chroot /home/oracle/jail
```


To implement a `chroot` jail and run `/bin/bash`, create the `bin` directory in the artificial root directory (`/home/oracle/jail` in this example), and copy `/bin/bash` into this directory:

```
$ mkdir /home/oracle/jail/bin
$ cp /bin/bash /home/oracle/jail/bin
```

The `/bin/bash` command is dynamically linked to shared libraries. These libraries must also be copied into the `chroot` jail.

Use the `ldd` command to determine which libraries are required by the `/bin/bash` command:

```
$ ldd /bin/bash
linux-vdso.so.1 => (0x0000...)
libtinfo.so.5 => /lib64/libtinfo.so.5 (0x0000...)
libdl.so.2 => /lib64/libdl.so.2 (0x0000...)
libc.so.6 => /lib64/libc.so.6 (0x0000...)
/lib64/ld-linux-x86-64.so.2 (0x0000...)
```

Copy each of these files into a `lib64` directory in the artificial root directory.

Make the `lib64` directory and copy the shared libraries into this directory:

```
$ mkdir /home/oracle/jail/lib64
$ cp /lib64/{libtinfo.so.5,libdl.so.2,libc.so.6,ld-linux-x86-64.so.2} /home/oracle/jail/lib64
```

Now that all the required files are in their expected locations, running the `chroot` command (as `root`) results in the following:

```
# chroot /home/oracle/jail
bash-4.1#
```

The command succeeded this time and the `/bin/bash` program executed. Entering `pwd` to print the current directory displays `/`, even though the actual directory is `/home/oracle/jail`:

```
bash-4.1# pwd
/
```

The `pwd` command runs because it is a shell built-in command. Running any other command fails because `bash` cannot find the command. The process assumes it is in the root directory and has no visibility or knowledge of any files above this directory in the hierarchy.

For example, running the `ls` command fails:

```
bash-4.1# ls
bash: ls: command not found
```

Use the `exit` command to exit the `chroot` jail.

```
bash-4.1# exit
exit
#
```

Running Services in a `chroot` Jail

- DNS and FTP includes `chroot` jail options.
- DNS:
  - Install the `bind-chroot` package.
  - `/var/named/chroot` becomes the `chroot` for BIND files.
- FTP (`vsftpd` daemon):
  - Anonymous users are automatically placed in a `chroot` jail.
  - `/var/ftp` appears as `/`.
  - Local user home directories can be configured as `chroot` jails.
  - Set options in the `/etc/vsftpd/vsftpd.conf` file.

Two services are set up to take advantage of `chroot` jails. You can set up DNS so that `named` runs in a jail. The `vsftpd` FTP server can automatically start `chroot` jails for clients.

DNS in `chroot` Jail

The `bind-chroot` package allows you to set up `named` to run in a `chroot` jail. When you install this package, the `/var/named/chroot` directory is created and becomes the `chroot` jail directory for all BIND files.

- The `/var/named` directory becomes `/var/named/chroot/var/named`.
- `/etc/named*` files become `/var/named/chroot/etc/named*` files.

Installing this package also sets the `ROOTDIR` shell variable to `/var/named/chroot` in the `/etc/sysconfig/named` file.

The advantage of running `named` in a `chroot` jail is that if a hacker enters your system via a BIND exploit, the hacker's access to the rest of your system is isolated to the files under the `chroot` jail directory.

FTP Clients in `chroot` Jail

By default, anonymous users are placed in a `chroot` jail. When an anonymous user logs in to a `vsftpd` server, the user's home directory is `/var/ftp`. However, all that the user sees is `/`.

For example, a directory named `/var/ftp/upload` appears as `/upload` to an anonymous user. This prohibits anonymous users from being able to access any files above `/var/ftp` in the directory hierarchy.

Local users that access a `vsftpd` server are placed in their home directory. You can enable options in the `/etc/vsftpd/vsftpd.conf` file to put local users in a `chroot` jail, where the artificial root directory is the user's home directory. The following options exist in the `vsftpd` configuration file to implement a `chroot` jail for local users:

- `chroot_list_enable`
- `chroot_local_user`
- `chroot_list_file`

When a local user logs in to the `vsftpd` server, the `chroot_list_enable` directive is checked. If this directive is set to YES, the service checks the `/etc/vsftpd/chroot_list` file (by default) or another file specified by the `chroot_list_file` directive.

Another directive is then checked, `chroot_local_user`. If this directive is set to YES, then the `chroot_list` becomes a list of users to NOT `chroot`. If this directive is set to NO, the user is put into a `chroot` jail in his home directory.

Introduction to Packet-filtering Firewalls

- Packet filtering firewalls accept or deny network packets.
- The Linux kernel has built-in packet filtering functionality.
  - Netfilter is the kernel component that stores filtering rules.
- Two services are available in Oracle Linux 7 to create, maintain, and display the rules stored by Netfilter:
  - `firewalld`
  - `iptables`
- The default firewall service in Oracle Linux 7 is `firewalld`.
- `firewalld` offers several advantages over `iptables`:
  - Changes do not require restart of `firewalld` service.
  - Networks can be separated into different zones based on the level of trust.

A packet filtering firewall reads incoming network packets and filters (allows or denies) each data packet based on the header information in the packet. You can create packet filters, or rules, that determine which packets are accepted and which are rejected. For example, you can create a rule to block a port. If a request is made to the port that is blocked by the firewall, the request is ignored. If a service is listening on a blocked port, it does not receive the packets and is effectively disabled.

The Linux kernel has built-in packet filtering functionality called Netfilter. Netfilter consists of a set of tables that store rules that the kernel uses to control network packet filtering. Oracle Linux provides the `firewalld` service and the `iptables` services to manage the rules stored by Netfilter.

In Oracle Linux 7, the default firewall service is `firewalld`. You can configure `firewalld` by using the `firewall-cmd` command-line interface. You can also use the `firewall-config` GUI to configure `firewalld`.

The `firewalld`-based firewall has the following advantages over `iptables`:

- Unlike the `iptables` command, the `firewall-cmd` command does not restart the firewall and disrupt established TCP connections.
- `firewalld` supports dynamic zones. Zones are discussed in subsequent slides.
- `firewalld` supports D-Bus for better integration with services that depend on firewall configuration.

Introduction to `firewalld`

- A dynamic firewall manager for Oracle Linux 7
- Supports firewall (network) zones
- Supports IPv4, IPv6, and Ethernet bridges
- Provides a D-BUS interface
- Provides two configuration modes:
  - Runtime: Configuration changes are immediate.
  - Permanent: Changes are written to configuration files and are applied when the `firewalld` service restarts.
- Configuration files exist in two directories:
  - `/usr/lib/firewalld`: Contains default configuration files. Do not make changes to these files.
  - `/etc/firewalld`: Configuration changes are written to files in this directory.

The `firewalld` service is a dynamic firewall manager for Oracle Linux 7. It provides support for network zones that allow you to define trusted services for specific network connections. Trusted services are network services, such as `ssh` and `dhcp`, which are accessible from other systems. The `firewalld` service has support for IPv4, IPv6, and for Ethernet bridges.

The `firewalld` service also provides a D-BUS interface. Services or applications already using D-BUS can add or request changes to firewall rules directly through the D-BUS interface. Refer to the `firewalld.dbus(5)` man page for more information.

The `firewalld` service has two types of configuration options:

- **Runtime:** Changes to firewall settings take effect immediately but are not permanent. Changes made in runtime configuration mode are lost when the `firewalld` service is restarted.
- **Permanent:** Changes to firewall settings are written to configuration files. These changes are applied when the `firewalld` service restarts.

Configuration files for `firewalld` exist in two directories:

- **`/usr/lib/firewalld`**: Contains default configuration files. Do not make changes to these files. An upgrade of the `firewalld` package overwrites this directory.
- **`/etc/firewalld`**: Changes to the default configuration files are stored in this directory. Files in this directory overload the default configuration files.

`firewalld` Zones

- A `firewalld` zone defines the following firewall features:
  - Services: Predefined or custom services that are trusted
  - Ports: Additional ports and associated protocols to trust
  - Masquerading: Translate IPv4 addresses to a single external address.
  - Port Forwarding: Forward inbound network traffic to an alternative port or IPv4 address.
  - ICMP Filter: Block selected ICMP messages.
  - Rich Rules: Extend existing `firewalld` rules to include additional source and destination addresses, and logging.
  - Interfaces: Network interfaces bound to the zone. The zone for an interface is specified with the `ZONE=<zone>` in the `/etc/sysconfig/network-scripts/ifcfg` file. If the option is missing, the interface is bound to the default zone.

ORACLE

The `firewalld` service allows you to separate networks into different zones based on the level of trust you want to place on the devices and traffic within a specific network. For each zone you can define the following features:

- **Services:** Predefined or custom services to trust. Trusted services are a combination of ports and protocols that are accessible from other systems and networks.
- **Ports:** Additional ports or port ranges and associated protocols that are accessible from other systems and networks.
- **Masquerading:** Translate IPv4 addresses to a single external address. With masquerading enabled, addresses of a private network are mapped to and hidden behind a public address.
- **Port Forwarding:** Forward inbound network traffic from a specific port or port range to an alternative port on the local system, or to a port on another IPv4 address.
- **ICMP Filter:** Block selected Internet Control Message Protocol messages.
- **Rich Rules:** Extend existing `firewalld` rules to include additional source and destination addresses and logging and auditing actions.
- **Interfaces:** Network interfaces bound to the zone. The zone for an interface is specified with the `ZONE=option` in the `/etc/sysconfig/network-scripts/ifcfg` file. If the option is missing, the interface is bound to the default zone.

Predefined `firewalld` Zones

- The `firewalld` software package includes a set of predefined network zones in the following directory:

```
# ls /usr/lib/firewalld/zones/
block.xml   drop.xml    home.xml   public.xml   work.xml
dmz.xml     external.xml   internal.xml   trusted.xml
```

- The zone files contain preset settings, which can be applied to a network interface. For example:

```
# grep -i service /usr/lib/firewalld/zones/public.xml
<service name="ssh"/>
<service name="dhcpv6-client"/>
```

- In this example, network interfaces bound to the `public` zone trust only two services, `ssh` and `dhcpv6-client`.

The `firewalld` package includes a set of predefined network zones. Zone settings for each zone are stored in the following directory:

```
# ls /usr/lib/firewalld/zones/
block.xml   drop.xml       home.xml       public.xml   work.xml
dmz.xml     external.xml   internal.xml   trusted.xml
```

The zone files contain a description and preset settings, which can be applied to a network interface. Example:

```
# cat /usr/lib/firewalld/zones/public.xml
...
<description>For use in public areas. You do not trust the other
computers on networks to not harm your computer. Only selected
incoming connections are accepted.</description>
<service name="ssh"/>
<service name="dhcpv6-client"/>
```

In this example, all network interfaces bound to the `public` zone trust only two services, `ssh` and `dhcpv6-client`.

A brief explanation of each zone follows:

- **drop**: Any incoming network packets are dropped, there is no reply. Only outgoing network connections are possible.
- **block**: Any incoming network connections are rejected with an `icmp-host-prohibited` message for IPv4 and `icmp6-adm-prohibited` for IPv6. Only network connections initiated from within the system are possible.
- **home**: For use in home areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.
- **public**: For use in public areas. You do not trust the other computers on the network to not harm your computer. Only selected incoming connections are accepted.
- **work**: For use in work areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.
- **dmz**: For computers in your demilitarized zone that are publicly accessible with limited access to your internal network. Only selected incoming connections are accepted.
- **external**: For use on external networks with masquerading enabled especially for routers. You do not trust the other computers on the network to not harm your computer. Only selected incoming connections are accepted.
- **internal**: For use on internal networks. You mostly trust the other computers on the networks to not harm your computer. Only selected incoming connections are accepted.
- **trusted**: All network connections are accepted.

Setting the Default `firewalld` Zone

- After an initial installation, the `public` zone is the default zone as specified in the configuration file, `/etc/firewalld/firewalld.conf`.

```
# grep -i defaultzone /etc/firewalld/firewalld.conf
DefaultZone=public
```

- Network interfaces are bound to the default zone unless specified with `ZONE=<zone>` in the `ifcfg` file.
- You can use the `firewall-cmd` command to change the default zone:

```
# firewall-cmd --set-default-zone=work
success
```

- You can also use the `firewall-config` GUI to change the default zone.

The `public` zone is initially defined as the default zone but this can be changed by editing the `/etc/firewalld/firewalld.conf` file. Network connections are bound to the default zone unless the zone is specified in the `ifcfg` file with the `ZONE=<zone>` option.

The following command shows the interfaces that are bound to the `public` zone:

```
# firewall-cmd --get-active-zone
public
  interfaces: eth0 eth1
```

You can use the `firewall-cmd` command to change the default zone. The following sequence of commands displays the default zone, then changes the default zone to the `work` zone.

```
# firewall-cmd --get-default-zone
public
# firewall-cmd --set-default-zone=work
success
```

You can also use the `firewall-config` GUI to change the default zone. From the menu bar, select **Options->Change Default Zone**, and then select a zone from a pop-up list.

`firewalld` Services

- A `firewalld` service is a combination of local ports and protocols and destination addresses.
- A `firewalld` service can also include Netfilter kernel modules that are automatically loaded when a service is enabled.
- The `firewalld` software package includes a set of predefined services in the following directory:

```
# ls /usr/lib/firewalld/services/
amanda-client.xml ipp-client.xml mysql.xml rpc-bind.xml
bacula-client.xml ipp.xml    nfs.xml   samba-client.xml
...
```

- Services can be enabled for a zone in Runtime mode.
- Service definitions can only be edited in Permanent mode.

A `firewalld` service is a combination of local ports and protocols and destination addresses. For each service, you can limit network traffic to a particular destination address and Internet Protocol (IPv4 or IPv6). A service can also include Netfilter kernel modules that are automatically loaded when a service is enabled.

Trusted services are accessible from all hosts and networks. You can choose to trust a service or choose not to trust a service for a selected zone at any time. In **Runtime** configuration mode, changes are implemented immediately without the need to restart the `firewalld` service or to disrupt existing network connections and services.

The `firewalld` package includes a set of predefined services. Configuration files for these services are stored in the following directory:

```
# ls /usr/lib/firewalld/services/
amanda-client.xml    ipp-client.xml    mysql.xml    rpc-bind.xml
bacula-client.xml    ipp.xml           nfs.xml      samba-client.xml
bacula.xml           ipsec.xml         ntp.xml      samba.xml
...
```

Service definition settings can only be changed in the **Permanent** configuration mode.

The service files contain a description and preset settings. Example:

```
# cat /usr/lib/firewalld/services/samba.xml
...
<description>This option allows you to access and participate in
Winfows file and printer sharing networks. You need the samba
package installed for this option to be useful.</description>
<port protocol="udp" port="137"/>
<port protocol="udp" port="138"/>
<port protocol="tcp" port="139"/>
<port protocol="tcp" port="445"/>
<module name="nf_conntrack_netbios_ns"/>
...
```

Starting `firewalld`

- To start `firewalld`:

```
# systemctl start firewalld
```

- To ensure `firewalld` starts at boot time:

```
# systemctl enable firewalld
```

- To check if `firewalld` is running:

```
# systemctl status firewalld
# firewall-cmd --state
```

- Three methods to configure the `firewalld` service:
  - `firewall-cmd`: Command-line interface
  - `firewall-config`: Graphical user interface
  - Edit various XML configuration files.

Use the following command to install the `firewalld` package and the GUI tool.

```
# yum install firewalld firewall-config
```

To start `firewalld`, run the following commands as root:

```
# systemctl start firewalld
# systemctl enable firewalld
```

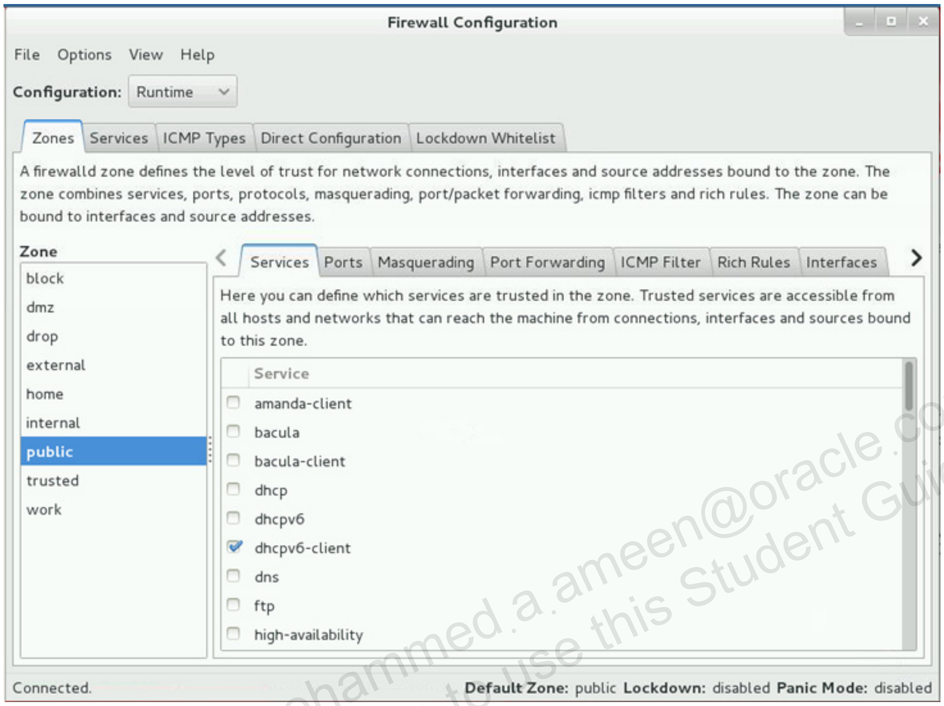
Use either of the following commands to check if `firewalld` is running.

```
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service...
   Active: active (running) since ...
...
# firewall-cmd --state
running
```

The `firewalld` service can be configured by using the `firewall-config` GUI, by using the `firewall-cmd` command-line interface, and by editing the various XML configuration files.

The `firewalld` Configuration Tool

```
# firewall-config
```



The slide shows the Firewall Configuration GUI, which can be used to configure `firewalld`. Enter the following command to start the GUI:

```
# firewall-config
```

The word “Connected” in the lower left corner indicates that the `firewalld` service is running. Start `firewalld` if “No connection. Waiting …” appears in the lower left corner.

The Configuration drop-down menu offers two options:

- **Runtime:** Changes to current firewall settings take effect immediately. Changes are not permanent and are lost when the `firewalld` service restarts.
- **Permanent:** Changes are written to configuration files and are applied when the `firewalld` service restarts. You can restart the `firewalld` from the GUI by selecting **Options->Reload Firewall** from the menu bar. If you select **Permanent**, an additional row of menu options appears allowing you to Add, Edit, and Remove a Zone, a Service, or an ICMP Type.

The GUI has several tabs – Zones, Services, ICMP Types, Direct Configuration, and LockDown WhiteList – which allow you to configure different firewall characteristics. Selecting each tab displays a short description. The slide shows the Zones tab selected and gives a short description of a zone. The bottom right shows the **Default Zone**, which is `public`.

The `firewall-cmd` Utility

- Command-line interface to configure the `firewalld` service.
- To get help on the `firewall-cmd` command:

```
# firewall-cmd --help
```

- To list information for all zones:

```
# firewall-cmd --list-all-zones
```

- To permit access by HTTP clients for the `public` zone:

```
# firewall-cmd --zone=public --add-service=http
```

- To permanently permit access by HTTP clients for the `public` zone:

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

ORACLE

The command-line tool `firewall-cmd` is part of the `firewalld` application, which is installed by default. It can be used to make permanent and non-permanent runtime changes.

Enter the following command to view the help output.

```
# firewall-cmd --help
...
```

The `firewall-cmd` command offers categories of options such as General, Status, Permanent, Zone, IcmpType, Service, Adapt and Query Zones, Direct, Lockdown, Lockdown Whitelist, and Panic. Refer to the `firewall-cmd(1)` man page for more information.

Some examples are given. Use the following command to list information for all zones. Only partial output is displayed.

```
# firewall-cmd --list-all-zones
public (default, active)
  interfaces: eth0 eth1
  sources:
  services: dhcpv6-client ssh
  ports:
...
```

To permit access by HTTP clients for the `public` zone:

```
# firewall-cmd --zone=public --add-service=http
success
```

To list services that are allowed for the public zone:

```
# firewall-cmd --zone=work --list-services
dhcpv6-client http ssh
```

Using this command only changes the **Runtime** configuration and does not update the configuration files.

The following sequence of commands shows that configuration changes made in **Runtime** configuration mode are lost when the `firewalld` service is restarted:

```
# systemctl restart firewalld
# firewall-cmd --zone=work --list-services
dhcpv6-client ssh
```

To make changes permanent, use the `--permanent` option. Example:

```
# firewall-cmd --permanent --zone=public --add-service=http
success
```

Changes made in **Permanent** configuration mode are not implemented immediately. Example:

```
# firewall-cmd --zone=work --list-services
dhcpv6-client ssh
```

However, changes made in **Permanent** configuration are written to configuration files. Restarting the `firewalld` service reads the configuration files and implements the changes. Example:

```
# systemctl restart firewalld
# firewall-cmd --zone=work --list-services
dhcpv6-client http ssh
```

Introduction to `iptables`

- Another firewall mechanism available in Oracle Linux 7.
- Firewall configuration information is stored in the `/etc/sysconfig/` directory:
  - `iptables` file is used for IPv4 configuration
  - `ip6tables` file is used for IPv6 configuration
- Every configuration change flushes old firewall rules and reloads new firewall rules.
- Stop the `firewalld` service before using `iptables`.
- Use the `iptables` command-line utility to create firewall configuration rules.

The `iptables` firewall mechanism is also available in Oracle Linux 7. Firewall configuration information is stored in the `/etc/sysconfig/` directory.

- IPv4 information is stored in the `/etc/sysconfig/iptables` file.
- IPv6 information is stored in the `/etc/sysconfig/ip6tables` file.

With the `iptables` service, every configuration change flushes all the old firewall rules and reads all the new rules from the configuration file.

In Oracle Linux 7 the `firewalld` service is enabled by default and the `iptables` and ip6tables services are disabled. To use the `iptables` service, you must first stop and disable the `firewalld` service. Use the following command to stop the `firewalld` service:

```
# systemctl stop firewalld
```

Use the following command to disable the `firewalld` service so that `firewalld` does not start when your system boots:

```
# systemctl disable firewalld
```

Use the `iptables` command-line interface to configure the `iptables` service. Previous versions of Oracle Linux included a GUI for configuring `iptables` but this tool no longer exists in Oracle Linux 7.

`iptables` Terminology

- The Netfilter component is a set of tables:
  - `Filter`: The default table
  - `NAT`: The Network Address Translation table
  - `Mangle`: The table used to alter certain fields in a packet
- Tables store rules, which consist of:
  - One or more match criteria
  - A single action, or target, such as `ACCEPT`, `DROP`, `REJECT`
- Rules are stored in chains. `Filter` table chains are:
  - `INPUT`: Inbound packets pass through this chain.
  - `OUTPUT`: Outbound packets pass through this chain.
  - `FORWARD`: Packets not addressed to the local system pass through this chain.

The Netfilter component of `iptables` is a set of tables. The three main tables are described as follows:

- **`Filter`:** The default table. This table is used primarily to `DROP` or `ACCEPT` packets based on their content.
- **`NAT`:** The Network Address Translation table. Packets that create new connections are routed through this table.
- **`Mangle`:** This table is used to alter certain fields in a packet.

These tables store rules that the kernel uses to make network packet filtering decisions. A rule consists of one or more criteria and a single action, or target. If the criteria in a rule match the information in a network packet header, the action or target is applied to the packet. Examples of targets include:

- **`ACCEPT`:** Continue processing the packet.
- **`DROP`:** End the packet's life without notice.
- **`REJECT`:** Similar to `DROP`, except it notifies the sending system that the packet was blocked. Use `DROP` if you do not want the sender to be notified.

Rules are stored in chains. Each rule in a chain is applied, in order, to a packet until a match is found. If there is no match, the chain's policy, or default action, is applied to the packet.

Each Netfilter table has several built-in chains. The default Netfilter table, named `filter`, contains the following built-in chains:

- **`INPUT`:** Inbound packets to the local system pass through this chain.
- **`OUTPUT`:** Packets created locally pass through this chain.
- **`FORWARD`:** Packets not addressed to the local system pass through this chain.

These chains are permanent and cannot be deleted. You can create additional, user-defined chains in this `filter` table.

Beginning `iptables` Maintenance

- To start the service:

```
# systemctl start iptables
```

- To configure iptables to start at boot time:

```
# systemctl enable iptables
```

- To list iptables:

```
# iptables -L [chain]
```

- Each chain has a default policy:
  – The action to take (ACCEPT or DROP) if no rules match
- To set default policy:

```
# iptables -P chain DROP|ACCEPT
```

- To save configuration changes:

```
# service iptables save
```


The firewall rules are active only if the `iptables` service is running. Start the service as follows. After changing the configuration, save the configuration and restart the service:

```
# systemctl start iptables
```

To ensure that `iptables` starts at boot time, enter the following command:

```
# systemctl enable iptables
```

Run the same commands for `ip6tables` if you are using IPv6.

Use the `iptables` command to create, maintain, and display the rules stored by Netfilter. Several options exist for the command. Long or short options are allowed. For example, to add rules to the end of a chain, use either of the following:

```
# iptables --append ...
# iptables -A ...
```

To remove rules from a chain, use either of the following:

```
# iptables --delete ...
# iptables -D ...
```

Use `iptables -h` or `iptables --help` to display all options.

Use the -L option, or --list, to list the current rules:

```
# iptables –L
Chain INPUT (policy ACCEPT)
target  prot  opt  source    destination
ACCEPT  all   --   anywhere  anywhere   state RELATED, ESTABLISHED
ACCEPT  icmp  --   anywhere  anywhere
ACCEPT  all   --   anywhere  anywhere
ACCEPT  tcp   --   anywhere  anywhere   state NEW tcp dpt:ftp
ACCEPT  tcp   --   anywhere  anywhere   state NEW tcp dpt:ssh
REJECT  all   --   anywhere  anywhere   reject-with icmp-host-...

Chain FORWARD (policy ACCEPT)
target  prot  opt  source    destination
REJECT  all   --   anywhere  anywhere   reject-with icmp-host-...

Chain OUTPUT (policy ACCEPT)
target  prot  opt  source    destination
```

The rules in all three chains (INPUT, FORWARD, OUTPUT) of the default table, filter, are displayed. Include the chain as an argument to limit output to a specific chain. For example, to list the rules in the INPUT chain only:

```
# iptables –L INPUT
```

Policies

Each iptables chain consists of a default policy and zero or more rules, which together define the overall ruleset for the firewall. If the information in a network packet header does not match any rule, the chain's policy, or default action, is applied to the packet. In this example, the policy for each chain is ACCEPT.

The default policy for a chain can be either DROP or ACCEPT. A more secure system would have a default of DROP and would allow only specific packets on a case-by-case basis. Set the default policy as follows, providing either the DROP or ACCEPT argument. This example blocks all incoming and outgoing network packets:

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
```

The FORWARD chain routes network traffic to its destination node. To create a DROP policy for these packets and to restrict internal clients from inadvertent exposure to the Internet, use the following rule:

```
# iptables -P FORWARD DROP
```

After establishing the default policies for each chain, create and save additional rules to meet your particular network and security requirements.

To save the rules to the /etc/sysconfig/iptables file so that they are loaded when the iptables service starts, use the following command:

```
# service iptables save
```

Adding a Rule by Using the `iptables` Utility

- To add a rule to a chain, use the following syntax:

```
iptables [-t <table>] -A <chain> <rule_specs> -j
   <target>
```

- Command-line options and arguments:
  - `-t <table>`: Defaults to the `Filter` table if omitted
  - `-A <chain>`: Appends a rule to `<chain>`
  - `rule_specs`: Specifies the rule criteria
  - `-j <target>`: Specifies the action to take if a match occurs
- Example:

```
# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

  - Accept incoming packets if protocol is TCP and destination port is 80 (`http`).

To add a rule to a chain, use the following syntax:

```
iptables [-t <table>] -A <chain> <rule_specs> -j <target>
```

The command-line options and arguments are described as follows:

- `-t <table>` option: Specifies the table (`filter`, `nat`, `mangle`). If omitted, the `filter` table is used by default.
- `-A <chain>` option: Appends a rule to `<chain>`. The chain value depends on the table. If the table is `filter`, the possible chains are `INPUT`, `OUTPUT`, and `FORWARD`.
- `rule_specs`: Specifies the rule criteria, or how to match a network packet.
- `-j <target>` option: Specifies the target of the rule, or what action to take if the packet matches the rule. The target value depends on the table. If the table is `filter`, the possible targets are `ACCEPT`, `DROP`, and `REJECT`.

The following example allows access to TCP port 80 on the firewall:

```
# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

Because no table is defined, the rule is written to the `filter` table. The chain is `INPUT`, so the rule is applied to incoming packets. The `rule_specs` consists of `-p tcp -m tcp --dport 80`. If information in the packet header matches the rule, the action taken is `ACCEPT`.

The rule_specs in this example are defined as follows:

- **-p tcp:** Matches if the packet uses the TCP protocol. The protocol can also use the long option, --protocol. The specified protocol can be any protocol name or number listed in the /etc/protocols file. When omitted, the default is all.
- **-m tcp:** The -m option specifies match extensions. Match extensions are loaded implicitly when -p or --protocol is specified, or explicitly using the -m or --match option followed by the matching module name. Various extra command-line options become available, depending on the specific module. The module name in this example is tcp. Use the -h or --help option after the module has been specified to receive help specific to that module. For example (the optional exclamation point [!] matches packets that do not match the criterion):

```
# iptables -p tcp –h
...
tcp match options:
[!] --tcp-flags mask comp  match when TCP flags & mask == comp
                           (Flags: SYN ACK FIN RST URG PSH ALL...
[!] --syn                  match when only SYN flag set
                           (equivalent to --tcp-flags SYN, RST...
[!] --source-port port[:port]
 --sport ...
                           match source port(s)
[!] --destination-port port[:port]
 --dport ...
                           match destination port(s)
[!] --tcp-option number    match if TCP option set
```

- **--dport 80:** Matches if the destination port is 80

Save any changes so that they are loaded when the iptables service is started, using the following command:

```
# service iptables save
```

The new entry appears in the /etc/sysconfig/iptables file:

```
-A INPUT –p tcp –m tcp --dport 80 –j ACCEPT
```

The iptables -L output displays the new entry as follows:

```
ACCEPT  tcp   --   anywhere  anywhere   tcp dpt:http
```

The TCP destination port of 80 is represented as http in the output because the http daemon listens for client requests on port 80.

`iptables` Rule Specs

- `-p, --protocol` *`protocol`*: Matches if the packet uses *`protocol`*
- `-s, --source` *`address`*`[/`*`mask`*`]`: Matches if the packet came from *`address`*
- `-d, --destination` *`address`*`[/`*`mask`*`]`: Matches if the packet is going to *`address`*
- `-j, --jump` *`target`*: Specifies what to do if the packet matches the rule specification
- `-i, --in-interface` *`name`*: Matches if the packet came from interface *`name`*
- `-o, --out-interface` *`name`*: Matches if the packet is to be sent to interface *`name`*

The `-p` (or `--protocol`) rule specification as shown in the previous example is commonly used as match criterion. The following describes some additional rule specifications for matching a network packet. Each of the rule specs can be preceded with an exclamation point (`!`) to have the inverse effect—that is, to match packets that do not match the criterion:

- **`-p, --protocol`** ***`protocol`***: Matches if the packet uses *`protocol`*
- **`-s, --source`** ***`address`*****`[/`*****`mask`*****`]`**: Matches if the packet came from *`address`*. The *`address`* can be a name or IP address and can include the optional *`mask`* with an IP address. The `--src` option is an alias and can also be used.
- **`-d, --destination`** ***`address`*****`[/`*****`mask`*****`]`**: Matches if the packet is going to *`address`*. The *`address`* can be specified as described in the `--source` option. The `--dst` option can also be used.
- **`-j, --jump`** ***`target`***: Specifies what to do if the packet matches the rule specification
- **`-g, --goto`** ***`chain`***: Specifies that the processing continues in a user-specified chain
- **`-i, --in-interface`** ***`name`***: Matches if the packet came from the *`name`* interface
- **`-o, --out-interface`** ***`name`***: Matches if the packet is to be sent to the *`name`* interface

More `iptables` Options

- `-D, --delete` *`chain rule_spec|rule_number`*: Removes a rule from *`chain`*
- `-I, --insert` *`chain rule_spec|rule_number`*: Inserts a rule in *`chain`* above an existing rule
- `-R, --replace` *`chain rule_spec|rule_number`*: Replaces an existing rule in *`chain`*
- `-F, --flush [`*`chain`*`]`: Deletes rules in *`chain`*
- `-N, --new-chain` *`chain`*: Creates a user-defined *`chain`*
- `-X, --delete-chain` *`chain`*: Deletes a user-defined *`chain`*

Three `iptables` options have been discussed; the `-A` (or `--append`) option to add a rule to the end of a chain, the `-L` (or `--list`) option to list all rules, and the `-P` (or `--policy`) option to set the default policy. The following describes some of the other options available with the `iptables` command:

- **`-D, --delete`** ***`chain rule_spec|rule_number`***: Removes a rule from *`chain`*. Define the rule to be removed by *`rule_spec`* or the *`rule_number`*. To display rule numbers, use the following command:

  ```
  # iptables -L --line-numbers
  ```

- **`-I, --insert`** ***`chain rule_spec|rule_number`***: Inserts a rule in *`chain`* above an existing rule that is specified by *`rule_spec`* or *`rule_number`*. If no existing rule is specified, the rule is inserted at the beginning of the chain.
- **`-R, --replace`** ***`chain rule_spec|rule_number`***: Replaces an existing rule in *`chain`*
- **`-F, --flush [`*`chain`*`]`**: Deletes rules in *`chain`*. If you omit the *`chain`* argument, all rules in all chains are deleted.
- **`-N, --new-chain`** ***`chain`***: Creates a new user-defined *`chain`*
- **`-X, --delete-chain`** ***`chain`***: Deletes a user-defined *`chain`*

NAT Table

- The Netfilter kernel subsystem provides a `nat` table in addition to the default `filter` table to facilitate NAT.
- Use the following option to specify the `nat` table:

```
# iptables -t nat ...
```

- Built-in chains for the `nat` table:
  - `PREROUTING`: Alters packets when they arrive
  - `OUTPUT`: Alters locally generated packets before they are sent out
  - `POSTROUTING`: Alters packets before they are sent out
- Targets for the `nat` table:
  - `DNAT`: Alters the destination IP address
  - `SNAT`: Alters the source IP address
  - `MASQUERADE`: Facilitates use with DHCP

Network Address Translation (NAT) is a process where a network device, usually a firewall, assigns a public address to a computer (or group of computers) inside a private network. NAT places private IP subnetworks behind one or a small pool of public IP addresses, masquerading all requests to one source rather than several. The main use of NAT is to limit the number of public IP addresses an organization or company must use, for both economy and security purposes.

The Netfilter kernel subsystem provides a `nat` table in addition to the default `filter` table to facilitate NAT. The `nat` table is consulted when a packet that creates a new connection is encountered. Use the `iptables -t` *`<table>`* option to specify the `nat` table when adding, deleting, replacing, or displaying rules:

```
# iptables -t nat ...
```

Whereas the built-in chains for the `filter` table are `INPUT`, `OUTPUT`, and `FORWARD`, the following built-in chains exist for the `nat` table:

- **`PREROUTING`:** Alters packets, such as destination address, when they arrive
- **`OUTPUT`:** Alters locally generated packets before they are sent out
- **`POSTROUTING`:** Alters packets before they are sent out

The targets for the `filter` table are `DROP`, `ACCEPT`, and `REJECT`. The `nat` table has specific targets as well:

- **`DNAT`:** Alters the destination IP address on an inbound packet so that it is routed to another host
- **`SNAT`:** Alters the source IP address on an outbound packet so that it appears to come from a fixed IP address, such as a firewall or router
- **`MASQUERADE`:** Differs from `SNAT` in that it checks for an IP address to apply to each outbound packet, making it suitable for use with DHCP

The following example specifies that the `nat` table use the built-in `PREROUTING` chain to forward incoming HTTP requests to a dedicated HTTP server at `172.31.0.23`. The rule changes the destination address of the packet.

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT
--to 172.31.0.23:80
```

The following example allows LAN nodes with private IP addresses to communicate with external public networks:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

This rule masks requests from LAN nodes with the IP address of the firewall's external device (in this case, `eth0`). `POSTROUTING` allows packets to be altered as they are leaving the firewall's external device. The `-j MASQUERADE` target masks the private IP address of a node with the external IP address of the firewall/gateway.

TCP Wrappers

- A TCP wrapper provides basic traffic filtering of incoming network traffic.
- Specifically, a TCP wrapper provides or denies access to “wrapped” network services.
- Use the `ldd` command to determine whether a network service is wrapped (linked to `libwrap.a`):

```
# ldd /usr/sbin/sshd | grep libwrap
libwrap.so.0 => /lib64/libwrap.so.0 ...
```

- TCP wrappers rely on two configuration files as the basis for access control:
  - `/etc/hosts.allow`
  - `/etc/hosts.deny`
- These files determine whether client access to network service is allowed or denied.

TCP wrappers provide basic traffic filtering of incoming network traffic. Access to “wrapped” network services running on a Linux server from other systems can be allowed or denied. A TCP wrapped service is one that has been compiled against the `libwrap.a` library. Use the `ldd` command to determine whether a network service is linked to `libwrap.a`. The following example determines the absolute path name of the `sshd` service, and then lists the shared libraries linked to the `sshd` service, using the `grep` command to search for the `libwrap` library:

```
# which sshd
/usr/sbin/sshd
# ldd /usr/sbin/sshd | grep libwrap
libwrap.so.0 => /lib64/libwrap.so.0 (0x00007f769e067000)
```

TCP wrappers rely on two configuration files as the basis for access control:

- `/etc/hosts.allow`
- `/etc/hosts.deny`

When a client attempts to connect to a network service on a remote system, these files are used to determine whether client access is allowed or denied.

TCP Wrappers Configuration

- Configuration files:
  - `/etc/hosts.allow`: Defines rules that allow client access to server daemons
  - `/etc/hosts.deny`: Defines rules that deny client access to server daemons
- The format for entries is the same for both files:

```
daemon_list : client_list [: command]
vsftpd : 192.168.2.*
```

- The `/etc/hosts.allow` file is read first:
  - If the daemon-client pair matches, access is granted.
  - The entry in `/etc/hosts.deny` is ignored if the entry in `/etc/hosts.allow` grants access.

Use `/etc/hosts.allow` and `/etc/hosts.deny` to define rules that selectively allow or deny clients access to server daemons on local system. The format for entries is as follows for both files:

```
daemon_list : client_list [: command]
```

A description of each field follows:

- **`daemon_list`**: A comma-separated list of daemons, or keyword `ALL` for all daemons
- **`client_list`**: A comma-separated list of clients, or keyword `ALL` for all clients
- **`command`**: An optional command that is executed when a client tries to access a server daemon

To allow client access, add the client host name or IP address in `/etc/hosts.allow`. To deny client access, add its name or IP address in `/etc/hosts.deny`.

The `/etc/hosts.allow` file is read first and is read from top to bottom. If a daemon-client pair matches the first line in the file, access is granted. If the line is not a match, the next line is read and the same check is performed. If all lines are read and no match occurs, the `/etc/hosts.deny` file is read, starting at the top. If a daemon-client pair match is found in the deny file, access is denied. If no rules for the daemon-client pair are found in either file, or if neither file exists, access to the service is granted.

Because access rules in `hosts.allow` are applied first, they take precedence over rules specified in `hosts.deny`. Therefore, if access to a service is allowed in `hosts.allow`, a rule denying access to that same service in `hosts.deny` is ignored.

The following are some examples of entries in the `/etc/hosts.allow` file:

To allow clients on the `192.168.2` subnet to access FTP (daemon is `vsftpd`):

```
vsftpd : 192.168.2.*
```

To allow all clients to access `ssh`, `scp`, and `sftp` (daemon is `sshd`):

```
sshd : ALL
```

Place the following entry in the `/etc/hosts.deny` file to deny FTP service to all clients except subnet `192.168.2.*` (this assumes the previous entry of `vsftpd:192.168.2.*` exists in `/etc/hosts.allow`):

```
vsftpd : ALL
```

Use the `.domain` syntax to represent any hosts from a given domain. The following example allows connections to `vsftpd` from any host in the `example.com` domain (if the entry is in `/etc/hosts.allow`):

```
vsftpd : .example.com
```

If this entry appears in `/etc/hosts.deny`, the connection is denied.

TCP Wrapper Command Options

- Use the optional *command* argument to send connection banners, warn of attacks, and enhance logging.
- To display the contents of a banner file:
  - vsftpd : ALL : banners /etc/banners/
- To append an entry to a log file:

```
ALL : 200.182.68.0 : spawn /bin/echo `date` %c %d >>
   /var/log/intruder_alert
```

- To elevate the logging level:

```
sshd : ALL : severity emerg
```

- To deny access from /etc/hosts.allow:

```
sshd : .example.com : deny
```


TCP wrappers are capable of more than allowing and denying access to services. With the optional *command* argument, they can send connection banners, warn of attacks from particular hosts, and enhance logging.

To implement a TCP wrapper banner for a service, use the `banner` option. This example implements a banner for `vsftpd`. You need to create a banner file anywhere on the system, giving it the same name as the daemon. In this example, the file is called `/etc/banners/vsftpd` and contains the following lines:

```
220-Hello, %c
220-All activity on ftp.example.com is logged.
220-Inappropriate use results in access privileges being
removed.
```

The `%c` token supplies a variety of client information. The `%d` token (not shown) expands to the name of the daemon that the client attempted to connect to. For this banner to be displayed to incoming connections, add the following line to the `/etc/hosts.allow` file:

```
vsftpd : ALL : banners /etc/banners/
```

TCP wrappers can warn you of potential attacks from a host or network by using the spawn directive. The spawn directive executes any shell command. In this example, access is being attempted from the 200.182.68.0/24 network. Place the following line in the /etc/hosts.deny file to deny any connection attempts from that network, and to log the attempts to a special file:

```
ALL : 200.182.68.0 : spawn /bin/echo `date` %c %d >>
/var/log/intruder_alert
```

To allow the connection and log it, place the spawn directive in the /etc/hosts.allow file.

The following entry in /etc/hosts.deny denies all client access to all services (unless specifically permitted in /etc/hosts.allow) and logs the connection attempt:

```
ALL : ALL : spawn /bin/echo “%c tried to connect to %d and was
blocked” >> /var/log/tcpwrappers.log
```

The log level can be elevated by using the severity option. Assume that anyone attempting to ssh to an FTP server is an intruder. To denote this, place an emerg flag in the log files instead of the default flag, info, and deny the connection. To do this, place the following line in /etc/hosts.deny:

```
sshd : ALL : severity emerg
```

This uses the default authpriv logging facility, but elevates the priority from the default value of info to emerg, which posts log messages directly to the console.

The following example states that if a connection to the SSH daemon (sshd) is attempted from a host in the example.com domain, execute the echo command to append the attempt to a special log file, and deny the connection. Because the optional deny directive is used, this line denies access even if it appears in the /etc/hosts.allow file:

```
sshd : .example.com \
: spawn /bin/echo `/bin/date` access denied >> /var/log/sshd.log \
: deny
```

Each option field (spawn and deny) is preceded by the backslash (\) to prevent failure of the rule due to length.

Refer to the man page on hosts_options for additional information and examples.

Quiz

Which of the following has a specific purpose of allowing or denying access to network services?

a. `chroot` jail
b. `firewalld`
c. `iptables`
d. TCP wrappers

Quiz

Which of the following statements are true?

a. The default firewall service in Oracle Linux 7 is `firewalld`.
b. When using `firewalld`, configuration changes do not require restart of `firewalld` service.
c. With `iptables`, networks can be separated into different zones based on the level of trust.
d. Both the `firewalld` and the `iptables` services include a GUI to make configuration changes.

ORACLE

Summary

In this lesson, you should have learned how to:

- Describe the `chroot` jail
- Use the `chroot` utility
- Describe packet-filtering firewalls
- Describe `firewalld`
- Configure `firewalld` packet filters
- Describe `iptables`
- Configure `iptables` packet filters
- Describe TCP wrappers
- Configure TCP wrappers

Practice 18: Overview

The practices for this lesson cover the following:

- Configuring a `chroot` jail
- Configuring a `chroot` jail for `ftp` users
- Exploring `firewalld`
- Configuring `firewalld`
- Configuring `iptables`
- Configuring a TCP wrapper

19

Oracle on Oracle

Objectives

After completing this lesson, you should be able to:

- Prepare your Oracle Linux server for Oracle Database installation
- Create Oracle software user and group accounts
- Set kernel parameters for Oracle Database
- Set Oracle database shell limits
- Configure HugePages
- Configure Oracle Database Smart Flash Cache (DBSFC)
- Use Oracle pre-install RPM
- Install and use ASMLib

Oracle Software User Accounts

- The Oracle database software owner:
  - Is commonly named `oracle`
  - Runs the OUI and has full privileges to install, uninstall, and patch Oracle software
  - Cannot be `root`
- The owner of the `httpd` process is:
  - A low-privileged OS user
  - Usually provided by the `nobody` user
- Database operations require a few more users:
  - Members of `OSOPER` group can start, stop, back up, and recover the database.
  - Members of the `OSDBA` group have `OSOPER` privileges, can create and drop database, and create other `OSDBA` members.

The Oracle software installation requires a Linux user to be a designated Oracle software owner. The Oracle software owner runs the OUI (Oracle Universal Installer) to install Oracle Database and has full privileges to install, uninstall, and patch the Oracle software. The OUI cannot be run as the `root` user. The name of the Oracle software owner is commonly `oracle`, but you can use a different name.

The Oracle software installation also requires a low-privileged OS user to be the owner of the `httpd` process. This is usually provided by the `nobody` user.

Database operations require a few more users. A user who is a member of the `OSOPER` group can start, stop, back up, and recover the database. A user who is a member of the `OSDBA` group can create, drop database, and create other DBA privileged users, in addition to the privileges of the `OSOPER`.

Ordinary database users can have OS accounts on the database server, but it is not necessary. It is common for database users to connect to the database through a client or application server without any OS account. OS user accounts might be required by the database application for batch jobs or specialized external processes. The Oracle default installation does not require any ordinary database user to have OS accounts.

With Oracle Grid Infrastructure & ASM there is a user called `grid` and three groups: `asmadmin`, `asmdba`, and `asmoper`. The owner of the Grid Infrastructure is commonly the "grid" user.

Oracle Software Group Accounts

- OSDBA:
  - This is commonly named `dba`.
  - Members of the OSDBA group have database administration privileges (`SYSDBA`).
- OSOPER:
  - This is commonly named `oper`.
  - Members of the OSOPER group have limited database administration privileges (`SYSOPER`).
- Oracle Inventory group:
  - This is commonly named `oinstall`.
  - All installed Oracle software is registered in this inventory.
  - Oracle software owner (`oracle`) is a member of this group.

The Oracle Database Installation Guide names three group identifiers:

- **`OSDBA` (`dba`):** Identifies OS accounts that have database administration privileges (`SYSDBA`)
- **`OSOPER` (`oper`):** Identifies OS accounts that have limited database administration privileges (`SYSOPER`)
- **Oracle Inventory group (`oinstall`):** Identifies the owner of the Oracle software

An `OSDBA` group is the only group that must be created to manage the database files. By default, this group is `dba`, but can have a different group name. `SYSDBA` is a high-level administrative privilege much like that of the `root` user on Linux. The members of the `OSDBA` group own the database files and have the privilege to connect to the database without a password, using `AS SYSDBA` through OS authentication.

The `OSOPER` group members connect to the database using the `AS SYSOPER` mechanism. This group has a restricted set of privileges. Each database can have its own `OSDBA` and `OSOPER` groups.

During installation, one inventory is created per system and all Oracle software installed on a server is registered in this inventory. The inventory group name is `oinstall`, and the Oracle software owner (`oracle`) is a member of this group. This user is also a member of the `OSDBA` and `OSOPER` groups.

Oracle Database 12c introduces new operating system groups:

- SYSBACKUP: Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations either from RMAN or SQL*Plus.
- SYSDG: Facilitates Data Guard operations. The user can perform operations either with Data Guard Broker or with the DGMGRL command-line interface.
- SYSKM: Facilitates Transparent Data Encryption keystore operations.

Each of these accounts provides a designated user for the new administrative privilege with the same name. See the following for more information: http://docs.oracle.com/database/121/ADMIN/dba.htm#ADMIN11042 and http://docs.oracle.com/database/121/ADMIN/dba.htm#ADMIN11052.

System Resource Tuning

- An Oracle database instance requires certain system resources.
- Shared memory must be adjusted for database use.
- Shared memory system uses semaphores, which must be adjusted.
- Each dedicated server process requires a network port.
- Larger network buffers are recommended.
- The maximum number of open files per process must be increased.
- Shell limit settings must be increased.

The Oracle Database instance requires certain system resources. Kernel resources are controlled by kernel parameters. Shell limits are controlled by the settings in the shell configuration files.

Oracle uses SYSV UNIX shared memory. The kernel parameters for shared memory must be adjusted for database use. The shared memory system also uses semaphores to coordinate shared memory access. Every Oracle instance requires a set of semaphores.

The Oracle instance communicates via network connections. Each dedicated server process requires a network port. In a shared server environment, each dispatcher requires a port.

Oracle recommends that you change the network buffers to allow larger defaults for send and receive buffers and a larger maximum buffer size. These changes are helpful to optimize network performance when there are high-bandwidth applications, such as RAC and GigE network interfaces.

Because an Oracle database often has a large number of open files, the kernel default setting for the maximum number of open files per process is too small.

Shell limit settings are typically used to prevent any one user from consuming so many resources that it prevents other users from being able to work. The typical user settings are too low for the Oracle software owner. The `oracle` user can have hundreds of processes executing and thousands of files open.

Linux Shared Memory

- Three shared memory-related kernel parameters:
  - `SHMMNI`: The maximum number of system-wide shared memory segments
  - `SHMMAX`: The maximum size of each segment
  - `SHMALL`: The maximum number of shared memory pages system wide
- For Oracle database, set `SHMMAX` >= the largest SGA.
- Shared memory kernel parameters are set in: `/etc/sysctl.conf`
- Parameters are viewable in:

```
# ls /proc/sys/kernel/sh*
shmall      shmmax      shmmni
```


The following are the memory-related kernel parameters:

- **SHMMNI:** The maximum number of system-wide shared memory segments
- **SHMMAX:** The maximum size of each segment
- **SHMALL:** The maximum number of shared memory pages system wide

Shared memory is allocated in segments. A segment is not necessarily as large as the maximum size; it is only as big as is allocated. If a process needs a larger shared memory area than can be allocated in one segment, it allocates multiple segments. Database instances often allocate multiple segments to accommodate a large System Global Area (SGA).

For Oracle Database, the `SHMMAX` parameter limits the size of each of the shared memory segments on the system. It should be equal to or larger than the largest SGA on the system; otherwise the SGA is made up of multiple memory segments.

The memory-related kernel parameters are set in the `/etc/sysctl.conf` file:

- `kernel.shmmni = 4096`
- `kernel.shmmax = 4398046511104`
- `kernel.shmall = 1073741824`

Semaphores

- Semaphores are a method of controlling access to critical resources.
- The Oracle instance uses semaphores to control access to shared memory.
- Semaphores are allocated based on the `PROCESSES` initialization parameter.
- All four semaphore parameters are set by a single `kernel.sem` parameter in `/etc/sysctl.conf`:
  - `semmsl`: Maximum number of semaphores per set
  - `semmns`: Total number of semaphores in the system
  - `semopm`: Maximum number of operations per `semop` call
  - `semmni`: Maximum number of semaphore sets

Semaphores are a robust method of controlling access to critical resources. The Oracle instance uses semaphores primarily to control access to shared memory. Semaphores are allocated based on the `PROCESSES` initialization parameter. The `PROCESSES` initialization parameter determines the maximum number of operating system processes that can be connected to Oracle Database concurrently.

Each Oracle instance tries to allocate two semaphore sets at startup. Immediately after startup, the instance releases one set of semaphores. This method prevents exhaustion of the semaphore resources. Each set allocates at least as many semaphores as the value of `PROCESSES`. If it does not, the Oracle instance gets more sets to satisfy the number of semaphores that it needs. If the instance cannot allocate enough semaphores (either in one set or in multiple sets), the instance does not start.

You can adjust the kernel parameters for semaphores. Semaphore settings are positional. All four of the semaphore parameters are set by a single kernel parameter, `kernel.sem`, in `/etc/sysctl.conf` and viewable in `/proc/sys/kernel/sem`. The four parameters are:

- **`semmsl`:** Maximum number of semaphores per set
- **`semmns`:** Total number of semaphores in the system
- **`semopm`:** Maximum number of operations per `semop` call
- **`semmni`:** Maximum number of semaphore sets

A `semop` call is a call to a function that actually uses the semaphores (for example, testing, setting, and clearing).

The following are the minimum required values. System administrators and DBAs might need to tune these values higher for production workloads, as per the documentation.

- For `semmsl`: 250 or the largest `PROCESSES` parameter of an Oracle database plus 50
- For `semmns`: 32000 or sum of the `PROCESSES` parameters for each Oracle database, adding the largest one twice, and adding an additional 25 to 50 for each database
- For `semopm`: 100
- For `semmni`: 128

Because these parameters are positional, the following illustrates setting the parameters as indicated in `/etc/sysctl.conf`:

```
kernel.sem = 250 32000 100 128
```

Parameters are viewable in:

```
# cat /proc/sys/kernel/sem
250     32000     100    128
```

Network Tuning

- Socket parameters:
  - An IP port is assigned to a server process when it starts.
  - An IP port is used to communicate with the user process.
  - The default range is 32768 through 61000.

```
# cat /proc/sys/net/ipv4/ip_local_port_range
9000    65500
```

- TCP/IP window size parameters:
  - Define read (rmem) and write (wmem) window sizes.
  - Set the default and maximum memory allocated for the network send and receive buffers.

```
# ls /proc/sys/net/core/[rw]mem*
rmem_default        wmem_default
rmem_max            wmem_max
```


An IP port is assigned to a database-dedicated server process when it starts. The IP port is used to communicate with the user process. By default, the range available is 32768 through 61000. In some databases with a very large number of users, the default range of ports that is available to non-root processes might not be adequate. In the following example, the IP port range is set to be from port 9000 through 65500:

```
# cat /proc/sys/net/ipv4/ip_local_port_range
9000    65500
```

On systems that use a firewall, a shared server configuration, or connection multiplexing, the number of needed ports can be greatly reduced.

TCP/IP window size parameters define the read (rmem) and write (wmem) window sizes for a TCP/IP packet. These parameters set the default and maximum memory allocated for the network send and receive buffers. Defaults are defined, and because TCP/IP communications occur with other machines, which can have different settings, you can adjust the sizes upward to attain compatibility. You cannot adjust them beyond the specified maximum value.

```
# ls /proc/sys/net/core/[rw]mem*
rmem_default   (262144)       wmem_default   (262144)
rmem_max      (4194304)       wmem_max      (1048576)
```

Setting the File Handles Parameter

- The File Handles parameter (`fs.file-max`) determines the maximum number of file handles that the Linux kernel allocates.
- The Oracle database background processes open all data files, logs, and other supporting files.
- The parameter must be high enough to include all the data files within your database and all supporting files.
- Set the kernel parameter in `/etc/sysctl.conf`:

  ```
  fs.file-max = 6815744
  ```
- View the setting in:

```
# cat /proc/sys/fs/file-max
6815744
```


The File Handles parameter (`fs.file-max`) determines the maximum number of file handles that the Linux kernel allocates. The Oracle database background processes open all the data files in addition to redo logs, the alert log, and other supporting files. Therefore, `fs.file-max` must be high enough to include all the data files within your database and all supporting files.

This value is set in `/etc/sysctl.conf` and is viewable in `/proc/sys/fs/file-max`:

```
# cat /proc/sys/fs/file-max
6815744
```

Asynchronous IO (AIO)

- AIO is the kernel subsystem used to ensure that Oracle databases run properly on Linux.
- AIO allows a process to initiate several I/O operations without having to block or wait for any to complete.
- The process can retrieve the results of the I/O later.
- Set the maximum number of allowable concurrent requests kernel parameter in `/etc/sysctl.conf`:

  `fs.aio-max-nr = 1048576`
- View the setting in:

```
# cat /proc/sys/fs/aio-max-nr
1048576
```


The Asynchronous IO (AIO) kernel subsystem is used to make system calls asynchronously in a generic fashion to ensure that Oracle databases run properly on Linux. The idea behind AIO is to allow a process to initiate several I/O operations without having to block or wait for any to complete. At some later time, or after being notified of I/O completion, the process can retrieve the results of the I/O.

The `/proc/sys/fs/aio-max-nr` file is the maximum number of allowable concurrent requests.

```
# cat /proc/sys/fs/aio-max-nr
1048576
```

Oracle-Related Shell Limits

- Three shell limits must be set for the `oracle` user:
  - `nofile`: Number of open file descriptors
  - `nproc`: Number of processes available to a single user
  - `stack`: Size of the stack segment of the process
- Soft limit versus hard limit
  - A hard limit can be changed only by `root`.
  - A soft limit can be changed by the user, up to the value of the hard limit.
- Define limits in the `/etc/security/limits.conf` file.
- Edit the `/etc/pam.d/login` file.
- A user can change a soft limit by using the `ulimit` command, for example:

```
$ ulimit -Sn 50
```


You must set three limits for an Oracle database to function properly. These apply to the `oracle` Linux user. You can typically set these limits to a high value.

The `nofile` limit is the maximum number of files that the user can have open at one time. The `oracle` user opens initialization files, data files, redo log files, and other files; therefore, this limit needs to be set high enough to have all those files open simultaneously.

The `nproc` limit is the maximum number of processes a given user can run at once. The `oracle` Linux user owns and starts all the background processes, server processes, and possibly the parallel query and dispatcher processes. This number must be set high enough to accommodate that. You must set this parameter high enough to manage the highest number of sessions in the database, plus some for other processes.

The `stack` limit is the size of the stack segment of the process.

For each of these settings, there is a soft limit and a hard limit. The hard limit can be changed only by the `root` user. The soft limit serves as the limit for the resource at any given time, which the user cannot exceed. But the user can change the soft limit, up to the value of the hard limit. The purpose of a limit is to prevent runaway situations where resources are being used up beyond what was intended by the processes running in the user space. Allowing the soft limit to be adjusted by the user, but never exceeding the `root`-defined hard limit, provides flexibility along with control.

Setting Shell Limits

The following example sets hard and soft limits for the `oracle` user. Two different files are modified:

1. Add the following to the `/etc/security/limits.conf` file:

   ```
   oracle soft nproc 16384
   oracle hard nproc 16384
   oracle soft nofile 1024
   oracle hard nofile 65536
   oracle soft stack 10240
   oracle hard stack 32768
   ```

2. Add or edit the following lines in the `/etc/pam.d/login` file:

   ```
   session required pam_limits.so
   ```

The `pam_limits.so` file is a Pluggable Authentication Module (PAM) that sets limits on the system resources that can be obtained in a user session. By default, limits are taken from the `/etc/security/limits.conf` file.

After a user has started a shell, the user can use the `ulimit` command to adjust the hard limit and soft limit for this specific shell. The hard limit cannot be increased after it is set, and the soft limit cannot be increased above the hard limit. In the following example, the `ulimit` command has no effect; it is setting the hard limit and soft limit to the same value that they have already been set to:

```
$ ulimit -u 16384 -n 65536
```

If the user issues the `ulimit -Sn 50` command (which sets the soft limit for the number of open files to 50), any attempt to open more than that results in an error. The user could still set it higher (for example, `ulimit -Sn 100`), which would result only in errors when the number of open file requests exceeds 100. However, the soft limit cannot be set higher than the hard limit.

Because a process inherits these settings from the shell (from which it is started) at the time that it is started, if you change the settings, any processes would have to be restarted for them to take effect. For example, if the shell limit values were changed, the Oracle database would have to be shut down and restarted.

HugePages

- HugePages:
  – Allow larger pages to manage memory
  – Are crucial for faster Oracle database performance
  – Are useful in both 32- and 64-bit configurations
  – Are integrated into the Linux kernel with release 2.6
  – Have been back-ported to some 2.4 kernels (2.4.21), but are implemented differently
  – Decrease page table overhead
  – Provide faster overall memory performance
  – Must be reserved during system startup
  – Are not swappable—there is no page-in/page-out overhead
- HugePage sizes vary from 2 MB to 256 MB, based on the kernel version and the hardware architecture.

HugePages is a feature of the Linux kernel. HugePages allow larger pages to manage memory as the alternative to small 4 KB page sizes (16 KB for IA64). HugePages are crucial for faster Oracle database performance on Linux if you have large RAM and SGA. If your combined database SGA is large (for example, more than 8 GB—but HugePages can also be important for smaller databases), you need HugePages configured. The HugePages feature is useful in both 32- and 64-bit configurations and is integrated into the Linux kernel with release 2.6.

HugePages Facts/Features

- The HugePages feature is back-ported to some 2.4 kernels. Kernel versions 2.4.21-* have this feature, but it is implemented in a different way. The difference from the 2.6 implementation is the organization within the source code and the kernel parameters that are used for configuring HugePages.
- HugePages can be allocated dynamically, but they must be reserved during system startup. Otherwise, the allocation might fail, because the memory is already paged in mostly 4 KB.
- HugePages are not subject to reservation/release after system startup unless there is system administrator intervention (basically changing the HugePages configuration).
- HugePages are not swappable; therefore, there is no page-in/page-out mechanism overhead. HugePages are universally regarded as pinned (never swapped to secondary storage).

- No kswapd operations: The kernel swap daemon, kswapd, gets very busy if there is a very large area to be paged (13 million page table entries for 50 GB memory) and uses an incredible amount of CPU resource. When HugePages are used, kswapd is not involved in managing them.
- HugePages allow fewer translations to be loaded into the Translation Lookaside Buffer (TLB). A TLB is a buffer (or cache) in a CPU that contains parts of the page table. This is a fixed-size buffer used for faster virtual address translation. A hugetlb is an entry in the TLB that points to a HugePage. HugePages are implemented via hugetlb entries (a HugePage is handled by a "hugetlb page entry"). The "hugetlb" term is also used synonymously with a HugePage.
- TLB entries cover a larger part of the address space when using HugePages. There are fewer TLB misses before the entire SGA, or most of it, is mapped in the TLB.
- Fewer TLB entries for the SGA also means more room for other parts of the address space.
- Decreased page table overhead: A page table is the data structure of a virtual memory system in an operating system to store the mapping between virtual addresses and physical addresses. This means that on a virtual memory system, the memory is accessed by first accessing a page table and then accessing the actual memory location implicitly.
- Eliminated page table lookup overhead: Because the pages are not subject to replacement, page table lookups are not required.
- Faster overall memory performance: On virtual memory systems, each memory operation is actually two abstract memory operations. Because there are fewer pages to work on, the possible bottleneck on page table access is clearly avoided.
- Oracle 11*g* Automatic Memory Management (AMM) and HugePages are not compatible. You must disable AMM on 11*g* to be able to use HugePages.

Size of a HugePage

HugePage sizes vary from 2 MB to 256 MB based on kernel version and hardware architecture. The following table shows the sizes of HugePages on different configurations:

Hardware Platform	Source Code Tree	Kernel 2.4	Kernel 2.6
Linux x86 (IA32)	i386	4 MB	4 MB
Linux x86-64 (AMD64, EM64T)	x86_64	2 MB	2 MB
Linux Itanium (IA64)	ia64	256 MB	256 MB
IBM Power Based Linux (PPC64)	ppc64/powerpc	N/A	16 MB
IBM zSeries Based Linux	s390	N/A	N/A
IBM S/390 Based Linux	s390	N/A	N/A

Configuring HugePages

Configuring your Linux OS for HugePages is a delicate process. If you do not configure properly, the system can experience serious problems such as:

- HugePages not used (HugePages_Total = HugePages_Free), wasting the amount of memory configured for HugePages
- Poor database performance
- System running out of memory or excessive swapping
- Some or all database instances cannot be started
- Crucial system services failing (for example, CRS)

Configuring HugePages

- Guidelines exist for different OS versions and hardware architectures.
- Configuring HugePages on 64-bit Linux:
  – Set the `memlock` user limit in `/etc/security/limits.conf` slightly smaller than installed RAM.
  – Disable AMM by setting `MEMORY_TARGET` and `MEMORY_MAX_TARGET` to zero.
  – Use the `hugepages_settings.sh` script to calculate the recommended value for the `vm.nr_hugepages` parameter.
  – Edit `/etc/sysctl.conf` and set the `vm.nr_hugepages` parameter.
  – Reboot your system.
- To check: `grep HugePages /proc/meminfo`

General guidelines exist to configure HugePages for more than one Oracle RDBMS instance. The following guidelines exist for the different OS versions and hardware architectures :

- “How to Configure RHEL 3.0 32-bit for Very Large Memory with ramfs and hugepages”
- “How to Configure Asianux 1.0 32-bit for Very Large Memory with ramfs and hugepages”
- “How to Configure RHEL 4 32-bit for Very Large Memory with ramfs and hugepages”
- “How to Configure SuSE SLES 9 32-bit for Very Large Memory with ramfs and hugepages”
- “How to Configure HugePages on 64-bit Linux”

HugePages on 64-bit Linux

The following are the configuration steps for configuring HugePages on 64-bit Linux. The configuration steps provided here are primarily for Oracle Linux, but the same concepts and configurations apply to other Linux distributions. These configuration steps guide you to do a persistent system configuration, which requires a reboot of the system.

Step 1: Have the `memlock` user limit set in the `/etc/security/limits.conf` file. Set the value (in KB) slightly smaller than installed RAM. If you have 64-GB RAM installed, set:

```
soft    memlock     60397977
hard    memlock     60397977
```

There is no harm in setting this value larger than your SGA requirements. The parameters are set by default on:

- Oracle Linux with Oracle-validated package installed
- Oracle Exadata DB compute nodes

Step 2: Log on again to the Oracle product owner account (for example, `oracle`) and check the `memlock` limit:

```
$ ulimit -l
60397977
```

Step 3: If you have Oracle Database 11*g* or later, the default database created uses the Automatic Memory Management (AMM) feature, which is incompatible with HugePages. Disable AMM before proceeding. To disable AMM, set the initialization parameters `MEMORY_TARGET` and `MEMORY_MAX_TARGET` to `0` (zero).

Step 4: Make sure that all your database instances are up (including ASM instances) as they would run on production. Use the `hugepages_settings.sh` script in Document 401749.1 to calculate the recommended value for the `vm.nr_hugepages` kernel parameter:

```
$ ./hugepages_settings.sh
...
Recommended setting: vm.nr_hugepages = 1496
```

You can also calculate a proper value for the parameter yourself but that is not advised if you do not have extensive experience with HugePages.

Step 5: Edit the `/etc/sysctl.conf` file and set the `vm.nr_hugepages` parameter:

```
vm.nr_hugepages = 1496
```

This causes the parameter to be set properly with each reboot.

Step 6: Stop all the database instances and reboot the server.

The performed configuration is based on the RAM installed and combined size of SGA of database instances that you are running. If any of the following changes occur, revise your HugePages configuration to make it suitable to the new memory framework:

- Changes to the amount of RAM installed for the Linux OS
- New database instance(s) introduced
- Changes to SGA size or configuration for one or more database instances

Check and Validate the Configuration

After the system is rebooted, make sure that your database instances (including the ASM instances) are started. Automatic startup via OS configuration or CRS, or manual startup (whichever method you use) has been performed. Check the HugePages state from `/proc/meminfo`:

```
# grep HugePages /proc/meminfo
HugePages_Total:    1496
HugePages_Free:      485
HugePages_Rsvd:      446
HugePages_Surp:        0
```

The values in the output vary. For a valid configuration, ensure that the `HugePages_Free` value is smaller than `HugePages_Total` and that there are some `HugePages_Rsvd`. The sum of `Hugepages_Free` and `HugePages_Rsvd` can be smaller than your total combined SGA as instances allocate pages dynamically and proactively as needed.

Oracle Database Smart Flash Cache (DBSFC)

- DBSFC:
  - Is available for both Oracle Solaris and Oracle Linux customers with the 11*g* R2 database or the 12*c* database
  - Allows you to extend the Oracle Buffer Cache in memory (SGA) using secondary flash-based storage
  - Helps with read-only/read-mostly workloads
- When a block gets modified, it is modified in the standard database buffer cache, written to disk and copied over into the flash cache.
- A subsequent read can then be from this fast storage instead of from the originating data files.
- See http://www.oracle.com/technetwork/articles/servers-storage-admin/smart-flash-cache-oracle-perf-361527.html.

The Oracle Database Smart Flash Cache (DBSFC) feature is available for both Oracle Solaris and Oracle Linux customers with the 11*g* R2 database and the 12*c* database. DBSFC allows you to extend the Oracle Buffer Cache in memory (SGA) by using secondary flash-based storage. This flash-based storage can be presented to the database through a file on a file system on flash storage, through a raw disk device (flash-based), or by adding flash storage to Oracle ASM and creating a region inside ASM. See http://www.oracle.com/technetwork/articles/servers-storage-admin/smart-flash-cache-oracle-perf-361527.html for more information. Note that multiple devices can be used in 12*c* for the server side flash cache.

DBSFC is a read-only cache extension that helps with read-only/read-mostly workloads. It contains clean blocks that are removed from the buffercache/sga and now first get placed in this extended cache. A subsequent read can then be from this fast storage instead of from the originating data files. When a block gets modified, it is modified in the standard database buffer cache, written to disk, and copied over into the flash cache.

The white paper referenced previously provides DBSFC configuration details for Oracle Linux. But to summarize, specify `DB_FLASH_CACHE_FILE` and `DB_FLASH_CACHE_SIZE` in the Oracle Initialization File, `init.ora`. These initialization parameters are also described at http://docs.oracle.com/database/121/ADMIN/memory.htm#ADMIN13395.

Oracle Pre-Install RPM

Oracle RDBMS Pre-Install RPM for Oracle Linux:

- Completes most pre-installation configuration tasks
- Downloads and installs various software packages and specific versions needed for database installation
- Creates the user `oracle` and the groups `oinstall` and `dba`
- Modifies kernel parameters in `/etc/sysctl.conf`
- Sets hard and soft shell resource limits in `/etc/security/limits.d` directory
- Sets `numa=off` in the kernel boot parameters for x86_64 machines

The Oracle RDBMS Pre-install RPM package is designed specifically for Oracle Linux to aid in the installation of the Oracle Database. You can complete most pre-installation configuration tasks by using this package, which is now available from the Unbreakable Linux Network or from the Oracle Public Yum repository.

This package was formerly known as `oracle-validated`. For Oracle Linux 6 and newer, the name of the package was changed to `oracle-rdbms-server-<version>-preinstall`. As of this writing, there are two versions of the `oracle-rdbms-preinstall` RPM:

- `oracle-rdbms-server-11gR2-preinstall`
- `oracle-rdbms-server-12cR1-preinstall`

The pre-install RPM configures an Oracle Linux machine so that you can immediately run the OUI database installation. The pre-install package is available for x86_64 only. Specifically, the package:

- Downloads and installs the various software packages and specific versions needed for database installation, with package dependencies resolved via `yum`
- Creates the user `oracle` and the groups `oinstall` and `dba`, which are the defaults used during database installation

The pre-install package also performs the following tasks:

- It modifies kernel parameters in `/etc/sysctl.conf` to change settings for shared memory, semaphores, the maximum number of file descriptors, and so on.
- The "11*g* R2" package sets hard and soft shell resource limits in `/etc/security/limits.conf`, such as the number of open files, the number of processes, and stack size to the minimum required based on the Oracle Database 11*g* Release 2 Server installation requirements. The "12*c* R1" version sets limits by using a file in the `/etc/security/limits.d` directory.
- It sets `numa=off` in the kernel boot parameters for x86_64 machines.

Further details for the "11*g* R2" version are available at http://oss.oracle.com/pipermail/el-errata/2012-March/002727.html.

Oracle ASM

- For stand-alone or Oracle RAC databases, you must have space available on Oracle ASM.
  - Creating Oracle Clusterware files on block or raw devices is no longer supported.
- ASM performs the functions of a volume manager and a file system.
- ASM consists of a specialized Oracle instance and a set of disk groups that are managed through the ASM instance.
- A disk group is a set of disk devices that ASM manages.
  - Each disk device can be a partition, a logical volume, a RAID array, or a single disk.
  - ASM spreads data evenly across the disk group to optimize performance and usage.

If you install stand-alone or Oracle RAC Databases, you must have space available on Oracle ASM for Oracle Clusterware files (voting disks and Oracle Cluster Registries), and for Oracle Database files. Creating Oracle Clusterware files on block or raw devices is no longer supported for new installations.

ASM consists of a specialized Oracle instance and a set of disk groups that are managed through the ASM instance. ASM performs the functions of a volume manager and a file system. ASM can be used for single instance or clustered databases. When using Oracle ASM for either the Oracle Clusterware files or Oracle Database files, Oracle creates one Oracle ASM instance on each node in the cluster, regardless of the number of databases.

The ASM instance manages disks in disk groups. An ASM instance must be configured and running before a database instance can access ASM files. This configuration is performed automatically if the Database Configuration Assistant is used for database creation.

A disk group is a set of disk devices that ASM manages as a single unit. Each disk device is a block device: a partition, logical volume, a RAID array, or a single disk. ASM spreads data evenly across all the devices in the disk group to optimize performance and usage. You can add or remove disk devices from a disk group without shutting down the database. When you add or remove devices, ASM rebalances the files across the disk group. You can create multiple disk groups to handle specific tasks, such as database backup and recovery operations, in addition to database file storage activities.

Grid Installation Owner and ASMOPER

During installation, in the Privileged Operating System Groups window, it is now optional to designate a group as the OSOPER for ASM group. If you choose to create an OSOPER for ASM group, then you can enter a group name configured on all cluster member nodes for the OSOPER for ASM group. In addition, the Oracle Grid Infrastructure installation owner no longer is required to be a member.

Oracle ASM Job Role Separation Option with SYSASM

The SYSASM privilege that was introduced in Oracle ASM 11*g* release 1 (11.1) is now fully separated from the SYSDBA privilege. If you choose to use this optional feature, and designate different operating system groups as the OSASM and the OSDBA groups, then the SYSASM administrative privilege is available only to members of the OSASM group. The SYSASM privilege can also be granted by using password authentication on the Oracle ASM instance.

OSASM is an operating system group that is used exclusively for Oracle ASM. Members of the OSASM group can connect as SYSASM by using operating system authentication and have full access to Oracle ASM.

You can designate OPERATOR privileges (a subset of the SYSASM privileges, including starting and stopping Oracle ASM) to members of the OSOPER for ASM group.

Providing system privileges for the storage tier by using the SYSASM privilege instead of the SYSDBA privilege provides a clearer division of responsibility between Oracle ASM administration and database administration, and helps to prevent different databases that use the same storage from accidentally overwriting each other's files.

ASM Library Driver (ASMLib)

- ASMLib simplifies the management of ASM disks.
- ASMLib has three components:
  - `oracleasm-support`: Provides user space shell scripts, and is included with the Oracle Linux distribution
  - `oracleasmlib`: Provides the user space library, and is installed from Unbreakable Linux Network (ULN)
  - `oracleasm`: Is the kernel driver included in `kernel-uek`
- To configure ASMLib:

```
# oracleasm configure -i
```

- To mark disks as ASM disks:

```
# oracleasm createdisk ASM_DISK_NAME candidate_disk
```


If you intend to use ASM for database storage for Linux, Oracle recommends that you install the ASMLib RPMs to simplify storage administration.

ASMLib is free, optional software for the ASM feature of Oracle Database. ASMLib simplifies the management and discovery of ASM disks and makes I/O processing and kernel resource usage with ASM storage more efficient. It provides persistent paths and permissions for storage devices used with ASM, eliminating the need for updating `udev` or `devlabel` files with storage device paths and permissions.

ASMLib also contains Linux data integrity features. To enable Oracle application-to-disk data integrity checking, ASMLib must be used. The ASMLib kernel driver is what connects the data integrity dots between Oracle database and ASM. See the following for more information: http://oss.oracle.com/~mkp/docs/data-integrity-webcast.pdf.

ASMLib updates are delivered via Unbreakable Linux Network (ULN) for both Oracle Linux or Red Hat Enterprise Linux installations. Refer to the following for more information: http://ovmjira.us.oracle.com/confluence/display/OLPM/Oracle+Linux+FAQ#OracleLinuxFAQ-ASMLib.

ASMLib has three components:

- **`oracleasm-support`:** This package provides user space shell scripts.
- **`oracleasmlib`:** This package provides the user space library and is closed source.
- **`oracleasm`:** This is the kernel driver and is included in `kernel-uek`.

The `oracleasm-support` package is included with the Oracle Linux distribution. Install the `oracleasmlib` package from ULN. The `oracleasm` kernel driver is included in the UEK. You do not need to install any driver package when using this kernel.

The following webpage describes getting ASMLib from ULN: http://www.oracle.com/technetwork/server-storage/linux/uln-095759.html.

Oracle ASMLib Release Notes for Oracle Linux 7 are available from: http://www.oracle.com/technetwork/server-storage/linux/release-notes-092521.html.

Oracle ASMLib Downloads for Oracle Linux 7 are available from: http://www.oracle.com/technetwork/server-storage/linux/asmlib/ol7-2352094.html.

The full installation guide is part of the *Oracle Database Documentation*.

Configuring ASMLib

Configure ASMLib by logging in as `root` and entering the following command:

```
# oracleasm configure -i
```

You are prompted to provide the following information:

- The default user to own the driver interface
- The default group to own the driver interface
- Whether to scan for Oracle ASM disks on boot

The user to own the driver interface is the same user that owns the software installation, typically `oracle`. The group to own the driver interface is the group used for DBAs, typically `dba`. You want to scan for Oracle ASM disks on boot.

If you enter the command `oracleasm` configured without the `-i` flag, then you are shown the current configuration. After it is configured, to load and initialize the ASMLib driver, run the `oracleasm` utility with the `init` option as shown:

```
# oracleasm init
```

Marking Disks as ASM Disks

A disk that is configured for use with ASM is known as a candidate disk. For OUI to recognize partitions as Oracle ASM disk candidates, you must log in as `root` and mark the disk partitions that Oracle ASM can use. Disks are marked by using the `createdisk` option. Use the following syntax, where *`ASM_DISK_NAME`* is the name of the Oracle ASM disk group, and *`candidate_disk`* is the name of the disk device that you want to assign to that disk group:

```
# oracleasm createdisk ASM_DISK_NAME candidate_disk
```

Meaningful names can be assigned for each disk. You can create multiple disk groups. By providing descriptive names to each disk, you have an easier time assigning disks to disk groups when creating the ASM instance. When choosing names for drives, consider using the physical location of the drive in the name. Example:

```
# oracleasm createdisk VOL1 /dev/sda1
# oracleasm createdisk VOL2 /dev/sdb1
# oracleasm createdisk VOL3 /dev/sde1
```

Using ASMLib Commands

Available options for the `oracleasm` script:

- `configure`: Configure the ASM library driver.
- `init/exit`: Change the behavior of the ASMLib when the system starts.
- `createdisk`: Mark a disk device for use with ASM.
- `deletedisk`: Unmark a named disk device.
- `querydisk`: Determine whether a disk device or disk name is being used by the ASMLib.
- `listdisks`: List the disk names of marked disks.
- `scandisks`: Enable cluster nodes to identify which shared disks have been marked as ASMLib disks on another node.

ORACLE

To administer the Automatic Storage Management library driver and disks, use the `oracleasm` initialization script with different options. The following summarizes the available options for the `oracleasm` script:

- **`configure`:** Use this to configure the Automatic Storage Management library driver.
- **`init/exit`:** Use this to change the behavior of the ASMLib when the system starts. The `init` option causes the ASMLib driver to load when the system starts.
- **`createdisk`:** Use this to mark a disk device for use with the ASMLib and give it a name.
- **`deletedisk`:** Use this to unmark a named disk device. Do not use this command to unmark disks that are being used by an ASM disk group. You must drop the disk from the disk group before you unmark it. The syntax is as follows:

  ```
  # oracleasm deletedisk DISKNAME
  ```

- **`querydisk`:** Use this to determine whether a disk device or disk name is being used by the ASMLib. The syntax is as follows:

  ```
  # oracleasm querydisk {DISKNAME|devicename}
  ```

- **listdisks:** Use this to list the disk names of marked ASMLib disks.
- **scandisks:** Use this to enable cluster nodes to identify which shared disks have been marked as ASMLib disks on another node.

ASM Rebalance Operations

ASM attempts to use that same amount of space on all the disks of a disk group. The data is striped and mirrored across all the disks of a disk group at the file level. Even though the disk group has a default for mirroring and striping, each file can have its own stripe and mirror properties.

There are two modes of striping:

- 1 MB allocation units
- 128 KB units

The redundancy can be set to one of the following:

- **Normal:** Normal redundancy is two-way mirroring.
- **High:** High redundancy is three-way mirroring.
- **External:** External redundancy does no mirroring. It assumes that the disk volumes are mirrored by some external means, such as RAID 1 arrays.

When a disk is added to a disk group, a rebalance operation is started. ASM moves a set of data blocks (allocation units) from the existing disks to the new disk. The number of allocation units moved is proportional to the size of the new disk compared to the total size of the disk group. If a disk is dropped from the disk group, or fails, then the data is redistributed across the remaining disks to re-establish the redundancy requirements.

The rebalance operation is controlled through an ASM instance parameter or by a parameter associated with the operation. This parameter is named `ASM_POWER_LIMIT` and can be set from `0-11`. A setting of `0` stops the rebalance, and `11` takes all the resources that can be effectively used to minimize the time to complete the operation. A setting of `1` is the default to prevent rebalance operations from interfering with normal database operations.

Whenever a disk group is altered by adding or dropping disks, a rebalance operation is triggered. If there is insufficient remaining disk space for a drop operation, the `alter` command fails. The `alter` disk group command does not complete until the rebalance operation is finished.

Oracle ASM Filter Driver

In Oracle Database 12*c* there is a new ASM filter driver (ASMFD) that prevents accidental corruption or deletion of the ASM devices. Refer to the following for more information: http://docs.oracle.com/database/121/LADBI/oraclerestart.htm#LADBI8076. Steps to configure Oracle ASMFD are provided at:
http://docs.oracle.com/database/121/OSTMG/asminst.htm#OSTMG95909.

Quiz

Which of the following statements is true regarding ASM?

a. ASMLib is required to use ASM.
b. The `oracleasm` kernel driver is included in the Red Hat Compatible Kernel (RHCK).
c. For Oracle Universal Installer (OUI) to recognize partitions as Oracle ASM disk candidates, you must mark the disk partitions that Oracle ASM can use.
d. A RAID array cannot be included in an ASM disk group.

Summary

In this lesson, you should have learned how to:

- Prepare your Oracle Linux server for Oracle Database installation
- Create Oracle software user and group accounts
- Set kernel parameters for Oracle Database
- Set Oracle database shell limits
- Configure HugePages
- Configure Oracle Database Smart Flash Cache (DBSFC)
- Use Oracle pre-install RPM
- Install and use ASMLib

Practice 19: Overview

The practices for this lesson cover the following:

- Using `sftp` to upload `oracle` packages
- Installing and running Oracle RDBMS Pre-install
- Preparing disks for ASM use
- Installing and configuring ASMLib

20

System Monitoring

Objectives

After completing this lesson, you should be able to:

- Use the `sosreport` utility
- Use the `iostat`, `mpstat`, `vmstat`, `sar`, `top`, `iotop`, and `strace` utilities
- Use the `netstat` and `tcpdump` utilities
- Use the Wireshark network analyzer GUI
- Use the OSWatcher Black Box (OSWbb) tool
- Use OSWatcher Analyzer (OSWbba)
- Describe Enterprise Manager Ops Center
- Describe Linux Patch and Provisioning using Enterprise Manager Ops Center
- Describe Spacewalk

`sosreport` Utility

- The `sosreport` utility:
  – Collects debugging information about a system
  – Stores the information in a compressed file in `/var/tmp`
- Run the tool as follows:

```
# sosreport
...
Please enter your first initial and last name...
Please enter the case number...:
```

- `sosreport` uses plug-ins. Options exist to manage plug-ins:
  – `-l`: List the status of all available plug-ins.
  – `-n` *`PLUGNAME`*: Do not load specified plug-in(s).
  – `-e` *`PLUGNAME`*: Enable the specified plug-in(s).

ORACLE

The `sosreport` tool collects information about a system, such as hardware configuration, installed software packages, configuration, and operational state. This information is stored in a single compressed file in the `/var/tmp` directory, and the file can be sent to a support representative to assist in troubleshooting a problem. The `sosreport` tool replaces an earlier version of the tool called `sysreport`.

To run the tool, first install the `sos` package:

```
# yum install sos
```

Run the report as the `root` user. The version of the tool is displayed along with a short description of the tool and the output it produces. You are prompted to press Enter to continue or Ctrl + C to quit.

```
# sosreport
...
Press ENTER to continue, or CTRL-C to quit.
```

Press Enter to start. You are prompted as follows:

```
Please enter your first initial and last name [host03...]:
Please enter the case number you are generating this report for:
```

The name and case number that you provide becomes part of the file name created by the tool. After the tool completes, you can uncompress the file and view the contents, by running the following commands:

```
# cd /var/tmp
# xz -d <sosfile>.xz
# tar xvf <sosfile>.tar
```

Extracting the file creates a directory, which includes the output of several system status commands as well as the contents of some configuration directories on your system. The following is a sample list of the output collected on a system named `host03`:

```
# ls /var/tmp/sosreport-host03*
boot/      etc/      lib/    proc/    sos_commands/    uptime
chkconfig  free      lsmod   ps       sos_logs/        usr/
date       hostname  lsof    pstree   sos_reports/     var/
...
```

The `sosreport` uses plug-ins, which can be turned on and off. Use the following command to list the plug-ins, which are enabled and disabled, and plug-in options:

```
# sosreport –l
The following plugins are currently enabled:
abrt               ABRT log dump
acpid              acpid related information
anaconda           Anaconda / Installation information
...
The following plugins are currently disabled:
apache            inactive  Apache related information ...
ceph              inactive  information on CEPH
cloudforms        inactive  Cloudforms related information...
...
The following plugin options are available:
abrt.backtraces     off   collect backtraces for every ...
auditd.syslogsize   15    max size (MiB) of logs to collect
...
```

Additional options exist to control the plug-ins and the tool. The following is a partial list:

- **`-n`** ***`PLUGNAME`***: Do not load specified plug-in(s).
- **`-e`** ***`PLUGNAME`***: Enable the specified plug-in(s).
- **`-o`** ***`PLUGNAME`***: Enable only the specified plug-in(s), disable all others.
- **`-k`** ***`PLUGNAME.PLUGOPT=[VALUE]`***: Specify options for plug-ins.
- **`-a`**: Enable all (Boolean) options for all loaded plug-ins.
- **`--tmp-dir`** ***`DIRECTORY`***: Specify an alternative temporary directory.
- **`--name`** ***`NAME`***: Specify a name to be used for the archive.
- **`--ticket-number`** ***`NUMBER`***: Specify a ticket number to be used for the archive.

`iostat` Utility

- The `iostat` utility:
  - Reports CPU and I/O statistics
  - Is used during performance analysis to balance I/O load
- The `iostat` utility report has the following sections:
  - CPU utilization
  - Device utilization
- Include the `-x` option for extended statistics:

```
# iostat -x
```

- Execute `iostat` continuously at a specific *interval*, up to *count* times:

```
# iostat interval count
```

- For example, to run `iostat` every 10 seconds for 5 times:

```
# iostat 10 5
```

ORACLE

The `iostat` command is used for monitoring system input/output device loading by observing the time that the physical disks are active in relation to their average transfer rates. This information can be used to change system configuration to better balance the input/output load between physical disks and adapters.

```
# iostat
Linux 3.8.13-44.1.1.el7uek.x86_64 (host03.example.com)
11/17/2014    _x64_64_   (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          25.99    0.78    7.43   12.77    0.00   53.03

Device:   tps    kB_read/s    kB_wrtn/s    kB_read  kB_wrtn
xvda     0.23         1.24         1.57     887799   978180
xvdb     0.00         0.01         0.00       5221        0
```

The first line displays the Linux kernel version, host name, current date, architecture, and number of CPUs on your system.

CPU Utilization Report

The next two lines display CPU statistics. For multiprocessor systems, the CPU values are global averages among all processors. The columns are defined as follows:

- **`%user`**: The percentage of CPU used while executing applications at the user level
- **`%nice`**: The percentage of CPU used while executing at the user level with nice priority
- **`%system`**: The percentage of CPU used while executing at the system (kernel) level
- **`%iowait`**: The percentage of time the CPU(s) were idle while the system had an outstanding disk I/O request
- **`%steal`**: The percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor
- **`%idle`**: The percentage of time that the CPU was (or the CPUs were) idle and the system did not have an outstanding disk I/O request

Device Utilization Report

The remaining lines in the example display statistics on a per–physical device or per-partition basis. You can include block devices and partitions as arguments to the `iostat` command. If no arguments are included, the report displays all devices that the kernel has statistics for. The columns are defined as follows:

- **`Device`**: Device or partition name as listed in the `/dev` directory
- **`tps`**: Number of transfers (I/O request) per second issued to the device
- **`kB_read/s`**: Amount of data read from the device expressed in number of kilobytes per second.
- **`kB_wrtn/s`**: Amount of data written to the device expressed in number of kilobytes per second
- **`kB_read`**: Total number of kilobytes read
- **`kB_wrtn`**: Total number of kilobytes written

More detailed statistics can be included by providing different options to the `iostat` command. Some of the command-line options are listed:

- **`-c`**: Display the CPU utilization report.
- **`-d`**: Display the device utilization report.
- **`-m`**: Display statistics in megabytes per second.
- **`-x`**: Display extended statistics.

Multiple reports can be run at different intervals by using *interval* and *count* arguments. The following example displays `6` reports at `2`-second intervals for all devices:

```
# iostat –d 2 6
```

mpstat Utility

- The mpstat utility:
  - Collects and displays performance statistics for all CPUs
  - Is used during performance analysis to determine CPU utilization
- Use the -P ALL option to include average usage of all CPUs:

```
# mpstat -P ALL
```

- Execute mpstat continuously at a specific *interval*, up to *count* times:

```
# mpstat interval count
```

- For example, to run mpstat every 2 seconds for 5 times:

```
# mpstat 2 5
```


The mpstat command collects and displays performance statistics for all logical CPUs in the system. When a CPU is occupied by a process, it is unavailable for processing other requests. These other processes must wait until the CPU is free. The mpstat command provides CPU usage to help you identify CPU-related performance problems.

```
# mpstat
Linux 3.8.13-44.1.1.el7uek.x86_64 (host03.example.com)
11/17/2014    _x64_64_    (2 CPUs)
04:14:55 PM  CPU  %usr  %nice  %sys  %iowait  %irq  %soft
%steal  %guest  %idle
04:14:55 PM  all  0.77   0.00  0.20     0.01  0.00   0.00
0.00      0.00   99.02
```

The first line displays the Linux kernel version, host name, current date, architecture, and number of CPUs on your system.

The first column is a time stamp. The remaining columns are defined as follows:

- **CPU:** Processor number starting at 0. The keyword all indicates that statistics are calculated as averages among all processors.

- **%usr**: Percentage of CPU used while executing at the user level
- **%nice**: Percentage of CPU used while executing at the user level with nice priority
- **%sys**: Percentage of CPU used while executing at the system (kernel) level. This does not include time spent servicing hardware and software interrupts.
- **%iowait**: Percentage of time the CPU was (or the CPUs were) idle during which the system had an outstanding disk I/O request
- **%irq**: Percentage of time spent by the CPU(s) to service hardware interrupts
- **%soft**: Percentage of time spent by the CPU(s) to service software interrupts
- **%steal**: Percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor
- **%guest**: Percentage of time spent by the CPU or CPUs to run a virtual processor
- **%idle**: Percentage of time that the CPU was (or the CPUs were) idle and the system did not have an outstanding disk I/O request

Similar to the `iostat` utility, `mpstat` allows multiple reports to run at different intervals. Use the following arguments:

```
# mpstat interval count
```

If you omit the *count* argument, the report runs at *interval* continuously. Press Ctrl + C to stop the report. The following example displays a report every 3 seconds, until terminated by pressing Ctrl + C:

```
# mpstat 3
```

The `-P` option followed by the keyword `ALL` displays statistics for processors. To report on a specific CPU, include the processor number as an argument to `-P`. The following displays 5 separate reports at 2-second intervals of all processors, and includes an average line:

```
# mpstat -P ALL 2 5
```

`vmstat` Utility

- The `vmstat` utility:
  - Monitors system memory usage
  - Is useful for detecting shortages of physical memory
- The `vmstat` report has six sections:
  - `Processes`: Number of processes in wait or sleep states
  - `Memory`: Amount of memory free, and amount used for virtual memory, buffers, and cache
  - `Swap`: Number of page-ins and page-outs
  - `IO`: Number of blocks received and sent
  - `System`: Number of interrupts and context switches
  - `CPU time`: Percentages for user, kernel, idle, iowait, and stolen
- **Recommended:** Run the utility with a delay interval:

```
# vmstat 5
```

- Additional options are available.

ORACLE

The `vmstat` command allows you to monitor your system's memory usage. It shows how much virtual memory there is, and how much is free and paging activity. You can observe page-ins and page-outs as they happen. This is extremely useful for detecting shortages of physical memory, which can adversely affect system performance.

The `vmstat` output contains more than just memory statistics. Output is broken up into six sections: `procs`, `memory`, `swap`, `io`, `system`, and `cpu`. To prevent the sample output from wrapping, the output is shown in two parts. As with `iostat` and `mpstat`, `vmstat` accepts *`interval`* and *`count`* arguments. The following example runs `3` reports `5` seconds apart:

```
# vmstat 5 3
procs ----------memory----------  ---swap--
 r  b   swpd   free   buff  cache    si   so
 1  0  13344   1444   1308  19692     0  168
 1  0  13856   1640   1308  18524    64  516
 3  0  13856   1084   1308  18316    56   64
```

This portion of the sample output shows only the first three sections. These three sections are described before the remaining three sections are shown.

The first two columns give information about processes:

- **r**: Number of processes that are in a wait state. These processes are not doing anything but waiting to run.
- **b**: Number of processes that were in sleep mode and were interrupted since the last update

The next four columns give information about memory:

- **swpd**: Amount of virtual memory used
- **free**: Amount of idle memory
- **buff**: Amount of memory used as buffers
- **cache**: Amount of memory used as cache

The next two columns give information about swap:

- **si**: Amount of memory swapped in from disk (per second)
- **so**: Amount of memory swapped out to disk (per second)

Nonzero `si` and `so` numbers indicate that there is not enough physical memory and that the kernel is swapping memory to disk.

The remaining three sections of the `vmstat` report:

```
# vmstat 5 3
-----io----  --system-- -----cpu------
   bi    bo    in   cs   us sy id wa st
  129    42  1505  713   20 11 69  0  0
  379   129  4341  646   24 34 42  0  0
   14     0   320 1022   84  9  7  0  0
```

The first two columns give information about I/O (input-output):

- **bi**: Number of blocks per second received from a block device
- **bo**: Number of blocks per second sent to a block device

The next two columns give the following system information:

- **in**: Number of interrupts per second, including the clock
- **cs**: Number of context switches per second

The last five columns give the percentages of total CPU time:

- **us**: Percentage of CPU cycles spent on user processes
- **sy**: Percentage of CPU cycles spent on system (kernel) processes
- **id**: Percentage of CPU cycles spent idle
- **wa**: Percentage of CPU cycles spent waiting for I/O
- **st**: Percentage of CPU cycles stolen from a virtual machine

Additional information can be included by providing different options to the `vmstat` command. Some of the command-line options are listed:

- **-a**: Display active and inactive memory.
- **-f**: Display the number of forks since boot.
- **-t**: Add a time stamp to the output.
- **-d**: Report the disk statistics.

`sar` Utility

- Provided by the `sysstat` package:
  - `sar`: Collects and displays ALL system activities statistics
  - `sadc`: The `sar` back-end tool that does the data collection
  - `sa1`: A script that runs `sadc` and stores system activities in a binary data file. `sa1` runs from `cron`.
  - `sa2`: Creates daily summary of the collected statistics. `sa2` runs from `cron`.
  - `pidstat`: Reports statistics based on the process ID (PID)
  - `cifsiostat`: Generates CIFS statistics
- Many options exist for `sar`:
  - `-A`, `-r`, `-b`, `-B`, `-d`, `-S`, and more
- You can specify *interval* and *count* parameters.

The `iostat` and `mpstat` commands are provided by the `sysstat` package. Additional resource monitoring tools, including `sar` and `sadc` (system activity data collector), are also provided by this package. See http://sebastien.godard.pagesperso-orange.fr/ for more information on `sysstat`. The following is a partial list of the files provided by the package:

```
# rpm -ql sysstat
/etc/cron.d/sysstat               /usr/bin/sadf
/etc/sysconfig/sysstat            /usr/bin/sar
/etc/sysconfig/sysstat.ioconf     /usr/lib64/sa
/usr/bin/cifsiostat               /usr/lib64/sa/sa1
/usr/bin/iostat                   /usr/lib64/sa/sa2
/usr/bin/mpstat                   /usr/lib64/sa/sadc
/usr/bin/nfsiostat                /var/log/sa
/usr/bin/pidstat
```

The `sadc` command collects system resource utilization data and writes it to a file. The `sadc` command is normally run by the `sa1` script, which is invoked by `cron` via the `/etc/cron.d/sysstat` file. By default, `cron` runs the `sa1` script every 10 minutes.

The `sar` command produces system utilization reports based on the data collected by `sadc`. The `sar` command is normally run by the `sa2` script, which is also invoked by `cron` via the `/etc/cron.d/sysstat` file. By default, `cron` runs the `sa2` script once a day at 23:53, allowing it to produce a report for the entire day's data. Example:

```
# cat /etc/cron.d/sysstat
*/10 * * * * root /usr/lib64/sa/sa1 ...
53 23 * * * root /usr/lib64/sa/sa2 ...
```

The `sa1` script logs output into `sysstat` binary log file format, and the `sa2` script reports it back in human-readable format. By default, the data is written to files in the `/var/log/sa` directory. The files are named `sa<dd>`, where `<dd>` is the current day's two-digit date. Running the `sar` command without any options uses the current daily data file as the data source. Use the `-f filename` option to specify a different data source. Sample output from `sar` is shown here:

```
# sar
Linux 3.8.13-44.1.1.el7uek.x86_64 (host03.example.com)... (1 CPU)
11:00:01 AM CPU %user %nice %system %iowait %steal %idle
11:10:01 AM all  0.01  0.00    0.01    0.09   0.00 99.89
11:20:01 AM all  0.02  0.00    0.09    0.11   0.00 99.79
11:30:01 AM all  0.08  0.00    0.11    0.32   0.00 99.49
11:40:01 AM all  0.01  3.69    7.41    6.68   0.00 82.21
Average:    all  ...
```

Many options exist for the `sar` command including the following:

- **`-A`:** Display all the statistics saved in the current daily data file.
- **`-r`:** Display memory utilization statistics.
- **`-b`:** Report I/O and transfer rate statistics.
- **`-B`:** Report paging statistics.
- **`-d`:** Report activity for each block device.
- **`-S`:** Report swap space usage statistics.
- **`-W`:** Report swapping statistics.

The `sar` command also accepts *`interval`* and *`count`* parameters. If the `interval` parameter is set to zero, `sar` displays the average statistics for the time since the system was started. Reports are generated continuously if the `interval` parameter is specified without the `count` parameter.

`top` Utility

- The `top` utility monitors system processes in real time.
- The upper section of the `top` output displays load averages, number of running and sleeping tasks, and overall CPU and memory usage.
- The lower section has a sorted list of processes, owner, running time, and CPU and memory usage.
- `top` sorts the list by most CPU-intensive tasks, and refreshes the list every three seconds by default.
- `top` provides an interactive interface for manipulating processes:
  – `h` or `?`: Display the help screen.
  – `F` or `f`: Display the field management screen.
  – `i`: Toggle the display of all tasks or just active tasks.
  – `q`: Quit.

The `top` command provides an ongoing look at processor activity in real time. It displays a list of the most CPU-intensive processes or tasks on the system and provides a limited interactive interface for manipulating processes. The following is a partial example of the `top` output:

```
# top
top - 03:55:32 up 21 days, 21:11 3 users, load average: ...
Tasks: 151 total, 1 running, 149 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.3 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0%hi ...
KiB Mem: 2056632 total, 1339688 used,  716944 free, 204764 ...
KiB Swap: 4286460 total,      0 used, 4286460 free, 759712 ...
PID   USER    PR NI  VIRT  RES  SHR S %CPU %MEM   TIME+   COMMAND
1744  root    20  0  125m  23m 7872 S  0.3  0.2  1:57:94  Xorg
   1  root    20  0 19448 1556 1244 S  0.0  0.1  0:03:54  init
16437 oracle 20  0  282m  13m 9408 S  0.3  0.1  0:24:55  gnom...
```

This sample listing is a point-in-time view of the output that `top` produces. The output is dynamic and refreshes every 3 seconds by default.

The output is divided into two main sections. The upper section displays general information such as the load averages during the last 1, 5, and 15 minutes (same output as the `uptime` command), number of running and sleeping tasks, and overall CPU and memory usage. The following keys change the output displayed in the upper section:

- **`l`:** Toggles load average and uptime on and off
- **`m`:** Toggles memory and swap usage on and off
- **`t`:** Toggles tasks and CPU states on and off

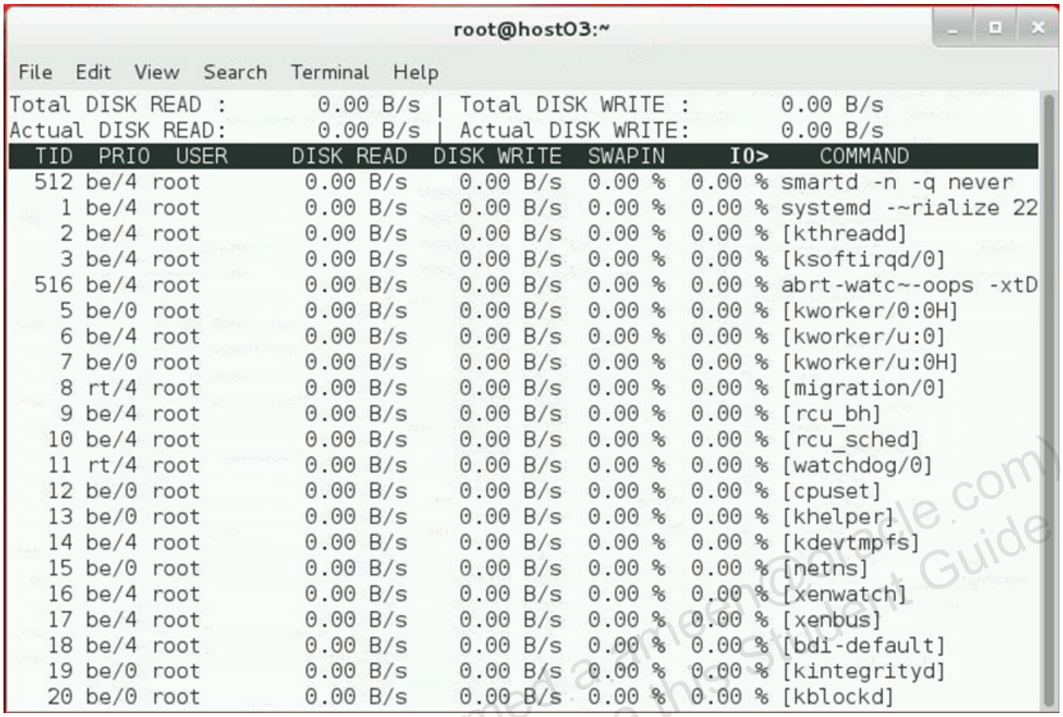
The lower section displays a sorted list of processes (usually by CPU usage) and their PIDs (process ID number), the user who owns the process, running time, and CPU and memory that the processes use. The following describes the columns in the lower section:

- **`PID`:** Task's unique process ID
- **`USER`:** Effective username of the task's owner
- **`PR`:** Priority of the task
- **`NI`:** Nice value of the task. A negative value means higher priority, a positive value means lower priority. Zero in this field means priority is not adjusted in determining a task's dispatchability.
- **`VIRT`:** Total amount of virtual memory used by the task. It includes all code, data, and shared libraries, plus pages that have been swapped out.
- **`RES`:** Non-swapped physical memory (resident size) a task has used
- **`SHR`:** Amount of shared memory used by a task. This memory could potentially be shared with other processes.
- **`S`:** Status of the task, which can be one of: `D` (uninterruptible sleep), `R` (running), `S` (sleeping), `T` (traced or stopped), or `Z` (zombie)
- **`%CPU`:** Task's share of the elapsed CPU time (CPU usage) since the last screen update, expressed as a percentage of total CPU time
- **`%MEM`:** Task's currently used share of available physical memory (memory usage)
- **`TIME+`:** Total CPU time that the task has used since it started
- **`COMMAND`:** Command-line or program name used to start a task

There are several keystroke commands that can be used while `top` is running. The following is a partial list:

- **`h` or `?`:** Displays a list of available commands (help screen)
- **`F` or `f`:** Field Management
  - Allows you to select columns to display
  - Allows you to rearrange order of columns
  - Allows you to sort by a specific column
- **`O` or `o`:** Allows you to set a filter
- **`d` or `s`:** Allows you to change the refresh interval
- **`c`:** Toggles the display of command-line and program name
- **`i`:** Toggles the display of all tasks or just active tasks
- **`S`:** Toggles the cumulative time on and off. When on, each process is listed with the CPU time that it and its dead children have used. When off, programs that fork into many separate tasks appear less demanding.
- **`u`:** Allows you to display only those tasks owned by a specific user
- **`k`:** Allows you to kill a process
- **`q`:** Allows you to exit or quit the `top` utility

iotop Utility



The `iotop` command is a Python program. `iotop` has a user interface similar to `top`, but it is used for monitoring swap and disk I/O on a per-process basis. If you are getting more disk activity on your system than you would like, `iotop` can help identify which process or processes are responsible for the excessive I/O.

The `iotop` command requires a kernel version 2.6.20 or higher and Python 2.5 or higher. Run `uname -r` to obtain your kernel version and `python -V` to get the Python version.

The top of the output displays the sum of the DISK READ and DISK WRITE bandwidth in B/s (bytes per second). After this is a list of all processes running on the system. Each process has a column labeled DISK READ and DISK WRITE, as well as SWAPIN and IO. The COMMAND column displays the name of the process.

By default `iotop` monitors all users on the system and all processes. Several options are available. The following is a partial list of options to `iotop`:

- **`-h`**: Display help and a list of options.
- **`-o`**: Show only processes and threads actually doing I/O.
- **`-u`** ***`USER`***: Show specific *`USER`* processes.
- **`-a`**: Show accumulated I/O instead of bandwidth.

Press the letter Q to exit.

`strace` Utility

- The `strace` utility is a debugging tool.
- It prints the system calls made by another program or process.
- Each line contains the system call name, followed by its arguments in parentheses and its return value.
- Errors typically return a value of `-1` and have the `errno` symbol and error string appended.
- Signals are printed as a signal symbol and a signal string.
- Output is printed on standard error or to the file specified with the `-o` option.

The `strace` command is a debugging tool, which prints a list of all the system calls made by another program or process. It displays the system calls that are called by a process and the signals that are received by a process. It is particularly useful in determining why a program continually crashes or does not behave as expected.

Each line in the trace contains the system call name, followed by its arguments in parentheses and its return value. Following is a partial output from stracing the `ls` command:

```
# strace ls
execve("/bin/ls, ["ls"], [/* 29 vars */]) = 0
brk(0)...
mmap(NULL, 4096 ...
access("/etc/ld.so.preload, R_OK)...
open("/etc/ld.so.cache", O_RDONLY)...
...
```

Errors typically return a value of `-1` and have the `errno` symbol and error string appended. Signals are printed as a signal symbol and a signal string. Output is printed on standard error or to the file specified with the `-o` option.

`netstat` Utility

- The `netstat` utility displays various network-related information.
- The `netstat` command without options displays a list of open sockets for each address family (AF).
- Several options exist:
  - `-A`: Specify the address family.
  - `-r`: Display the route table.
  - `-i`: Display network interface information.
  - `-s`: Display summary statistics for each protocol.
  - `-g`: Display multicast group membership information.
  - `-n`: Display IP addresses instead of the resolved names.
  - `-c`: Print information every second continuously.
  - `-e`: Display extended information.

ORACLE

The `netstat` command displays current TCP/IP network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. The `ss` command provides dump socket statistics but also shows information similar to `netstat`.

A number of command-line options and arguments exist, but `netstat` by itself displays a list of open sockets. Sockets are the interface between the user process and the network protocol stacks in the kernel. The protocol modules are grouped into protocol families such as `AF_INET`, `AF_IPX`, and `AF_PACKET`, and socket types such as `SOCK_STREAM` or `SOCK_DGRAM`. If you do not specify any address families, the active sockets of all configured address families are printed.

To specify the address families (low-level protocols) for which connections are to be shown, use the `-A` option followed by a comma-separated list of address family keywords. Possible address family keywords are `inet`, `inet6`, `unix`, `ipx`, `ax25`, `netrom`, and `ddp`. Example:

```
# netstat -A unix
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags Type   State   I-Node Path
unix  2      [ ]   DGRAM          7137   @/org/kernel/udev/udevd
...
```

Some of the other options for `netstat` are listed:

- **`-r` or `--route`:** Display the kernel routing table:

```
# netstat -r
Destination Gateway   Genmask  Flags MSS Window  irtt  Iface
default     192.0.2.1 0.0.0.0  UG      0      0     0  eth0
...
```

- **`-i` or `-I=`*`iface`*:** Display a table of all network interfaces or the specified *`iface`*:

```
# netstat -I=eth0
Iface  MTU Met   RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-DRP TX-OV...
eth0  1500   0 1131204      0     16      0 174989      0     0...
```

- **`-s` or `--statistics`:** Display summary statistics for each protocol:

```
# netstat -s
Ip:
     106564 total packets received
     0 forwarded
     0 incoming packets discarded
     10427 incoming packets delivered
     106069 requests sent out
Icmp:
     ...
```

`-l` or `--listening`: Display all ports that have a process currently listening for input.

```
# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp        0      0 *:pop3s       *:*             LISTEN
...
```

- **`-g` or `--groups`:** Display multicast group membership information for IPv4 and IPv6:

```
# netstat -g
Interface  RefCnt  Group
---------  ------  -----
lo         1       224.0.0.1
eth0       1       224.0.0.251
...
```

- **`-n` or `--numeric`:** Display IP addresses instead of the resolved names.
- **`-c` or `--continuous`:** Print information every second continuously.
- **`-e` or `--extend`:** Display additional information. Use this option twice for maximum detail.
- **`-p` or `--program`:** Show the PID and name of the program to which each socket belongs.

Any invalid option or argument displays a help screen listing usage and a brief description of available options.

`tcpdump` Utility

- The `tcpdump` utility is a packet-capture utility for network troubleshooting.
- Traffic is captured based on a specified filter.
- A variety of options exist, including:
  - `-D`: Print a list of network interfaces.
  - `-i`: Specify an interface on which to capture.
  - `-c`: Specify the number of packets to receive.
  - `-v`, `-vv`, `-vvv`: Increase the level of detail (verbosity).
  - `-w`: Write captured data to a file.
  - `-r`: Read captured data from a file.
- You can also specify host, source, or destination of traffic, and a specific protocol to capture.
- Boolean operators (`AND`, `OR`, `NOT`) allow complex filters.

The `tcpdump` utility allows you to capture packets that flow within your network to assist in network troubleshooting. The following are several examples of using `tcpdump` with different options.

To print a list of network interfaces available on which `tcpdump` can capture packets:

```
# tcpdump -D
1.eth0
2.eth1
3.any (Pseudo-device that captures on all interfaces)
4.lo
```

For each network interface, a number and an interface name is printed. The interface name or the number can be supplied to the `-i` flag to specify an interface on which to capture.

```
# tcpdump -i 1
listening on eth0, link-type EN10MB (Ethernet), capture size...
03:57:25.845920 ARP, Request who-has host02.example.com tell...
03:57:25.846093 ARP, Reply host02.example.com is-at 00:16:3e...
```

In this example, output is continuous until terminated by pressing Ctrl + C.

To exit tcpdump after receiving a specific number of packets, use the -c (count) option followed by the number of packets to receive. The following example captures two packets:

```
# tcpdump -i 1 -c2
...
2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

As shown in this example, when tcpdump finishes capturing packets, it reports the following:

- **packets captured:** This is the number of packets that tcpdump has received and processed.
- **packets received by filter:** A filter can be specified on the command line and only those packets that match the defined filter are processed by tcpdump and counted.
- **packets dropped by kernel:** This is the number of packets that were dropped due to a lack of buffer space. Use the -B option to set the buffer size.

To increase the detail (verbosity) of the output, use the -v option, or -vv for even more verbose output, or -vvv for the most verbose level of output:

```
# tcpdump -i 1 -v
# tcpdump -i 1 -vv
# tcpdump -i 1 -vvv
```

Using the tcpdump utility with the -w option allows you to write captured data to a file. This allows the captured data to be read by other network analysis tools, such as Wireshark. The following example captures data to a file named capture_file:

```
# tcpdump -i 1 -v -c2 -w capture_file
```

You can also read captured data from a file by using the -r option:

```
# tcpdump -r capture_file
```

Many other options and arguments can be used with tcpdump. The following are some specific examples of the power of the tcpdump utility.

To display all traffic between two hosts (represented by variables *host1* and *host2*):

```
# tcpdump host host1 and host2
```

To display traffic from only a source (src) or destination (dst) host:

```
# tcpdump src host
# tcpdump dst host
```

Provide the protocol as an argument to display only traffic for a specific protocol, for example tcp, udp, icmp, arp:

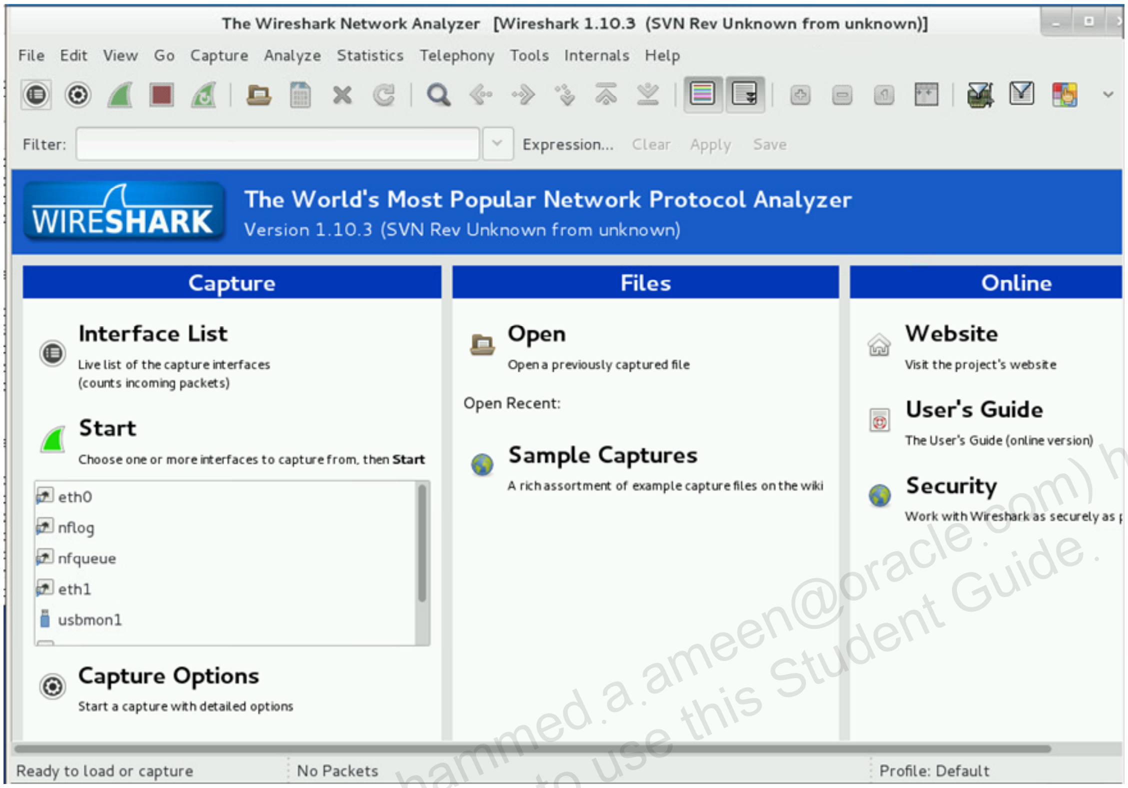
```
# tcpdump protocol
```

To filter based on a source or destination port:

```
# tcpdump src port ftp
# tcpdump dst port http
```

The tcpdump utility also accepts Boolean operators (AND, NOT, OR) and grouping of operators, allowing you to create complex filters for capturing network data.

Wireshark



The slide shows the Wireshark GUI. Wireshark is a network protocol analyzer that allows you to interactively browse packet data from a live network or from a previously saved capture file. The GUI is provided by the `wireshark-gnome` RPM but you also need to install the same version of the `wireshark` RPM. Documentation for Wireshark is installed in the `/use/share/wireshark` directory.

As indicated on the GUI, you can start a capture from any available network interface. Each live capture can be saved to a file for future analysis. You also open a previously captured file for analysis. Various capture options can be selected such as the following:

- Capture packets in promiscuous mode.
- Stop the capture after a specified number of packets, bytes, or a time period.
- Enable MAC name resolution.
- Enable network name resolution.

You can also filter a capture based on MAC address, IP address, protocol, or create your own filter expression. Wireshark provides packet search capabilities as well as packet coloring rules.

Also included with the Wireshark package is `tshark`, a text-based network protocol analyzer. `tshark` also allows you to capture packet data from a live network, or read packets from a previously saved capture file.

OSWatcher Black Box (OSWbb)

- OSWbb collects and archives operating system and network metrics to aid in diagnosing performance issues.
- OSWbb includes a built-in analyzer called OSWbba.
- Download the OSWbb TAR file from My Oracle Support (MOS).
- To install OSWbb, use the `tar` command:

```
# tar xvf oswbb732.tar
```

- To start OSWbb, use the following command:

```
# ./startOSWbb.sh
```

- To stop OSWbb, use the following command:

```
# ./stopOSWbb.sh
```


ORACLE

The Oracle OSWatcher Black Box (OSWbb) product is a collection of shell scripts intended to collect and archive operating system and network metrics to aid in diagnosing performance issues. OSWbb operates as a set of background processes on the server and gathers data on a regular basis, invoking such UNIX utilities as `vmstat`, `netstat`, `iostat`, `top`, and others.

Beginning with release 4.0.0, OSWbb includes a built-in analyzer called OSWbba, which analyzes the data that OSWbb collects. It provides information on system slowdowns, hangs, and other performance problems. It also provides the ability to graph `vmstat` and `iostat` data.

OSWbb is particularly useful for Oracle Real Application Clusters (RAC) and Oracle Grid Infrastructure configurations. OSWbb is included in the RAC-DDT (Diagnostic Data Tool) script file, but is not installed by RAC-DDT.

Installing OSWbb

You must install OSWbb on each node where data is to be collected. For RAC or shared disk systems, each node requires an OSWbb installation into a unique directory (for example, `/oswbb_node1` and `/oswbb_node2`). OSWbb is available through MOS Doc ID 301137.1 and can be downloaded as a TAR file named `oswbb732.tar`. After downloading the TAR file, copy the file to the directory where OSWbb is to be installed and run the following command:

```
# tar xvf oswbb732.tar
```

Extracting the TAR file creates a directory named `oswbb`, which contains all the files associated with OSWbb.

```
# ls oswbb
analysis/          ifconfigusub.sh  oswib.sh      tarupfiles.sh
call_du.sh         iosub.sh         oswnet.sh     tar_up_partial...
call_sar.sh        locks/           oswrds.sh     tmp/
call_uptime.sh     ltop.sh          oswsub.sh     topaix.sh
data/              mpsub.sh         profile/      vmsub.sh
docs/              nfssub.sh        psmemsub.sh   xtop.sh
Example_extras.txt  OSWatcherFM.sh  src
Exampleprivate.net  OSWatcher.sh    startOSWbb.sh
gif/               oswbba.jar       stopOSWbb.sh
```

Starting OSWbb

To start the OSWbb utility, execute the `startOSWbb.sh` shell script. The `startOSWbb.sh` script accepts two optional arguments that control the frequency (in seconds) that data is collected and the number of hours worth of data to archive. If you do not enter any arguments, the script runs with default values of `30` and `48`, meaning collect data every 30 seconds and store the last 48 hours of data in archive files.

The following example starts the tool and collects data at 60-second intervals and logs the last 10 hours of data to archive files. Some of the output produced when starting the tool is shown:

```
# ./startOSWbb.sh 60 10
Testing for discovery of OS Utilities...
VMSTAT found on your system.
IOSTAT found on your system.
MPSTAT found on your system.
NETSTAT found on your system.
TOP found on your system.
Testing for discovery of OS CPU COUNT
...
Starting Data Collection...
oswbb heartbeat: date/time
oswbb heartbeat: date/time (60 seconds later)
...
```

Stopping OSWbb

To stop OSWbb, execute the `stopOSWbb.sh` shell script. This terminates all processes associated with OSWbb and is the normal, graceful mechanism for stopping the tool.

OSWbb Diagnostic Data Output

- OSWatcher.sh is the main controlling script that spawns other scripts to collect diagnostic data.
- The data is stored in hourly archive files:
  - *<node_name>_<OS_utility>_YY.MM.DD.HH24*.dat
- Subdirectories are created in the archive directory.
- oswiostat: Contains the output from the iostat utility
- oswmeminfo: The contents of the /proc/meminfo file
- oswmpstat: Contains the output from the mpstat utility
- oswnetstat: Contains the output from the netstat utility
- oswprvtnet: Contains the status of RAC private networks
  - Requires you to manually create an executable file named private.net, which runs traceroute commands

The OSWatcher.sh shell script is the main controlling script that spawns individual shell processes to collect specific kinds of data by using UNIX operating system diagnostic utilities. Control is passed to individually spawned operating system data collector processes, which in turn collect specific data, time-stamp the data output, and append the data to files.

Data collectors exist for the ifconfig, top, vmstat, iostat, mpstat, netstat, and ps utilities, and for /proc/meminfo and /proc/slabinfo. There is also an optional collector for tracing private networks. The collected data files are stored in the archive subdirectory, which is created when OSWbb is started for the first time. The archive directory contains 10 subdirectories, one for each data collector.

```
# ls archive
oswifconfig/  oswmeminfo/  oswnetstat/  oswps/  oswtop/
oswiostat/    oswmpstat/   oswprvtnet/  oswslabinfo/  oswvmstat/
```

The data is stored in hourly archive files during the time that OSWbb is running. Files are named using the following format:

```
<node_name>_<OS_utility>_YY.MM.DD.HH24.dat
```

Each entry in the file contains a time stamp prefixed by *** characters. The contents of each of the 10 archive directories are described here:

oswiostat

OSWbb runs the `iostat` utility at the specified interval and stores the data in this directory. By default, `iostat` produces extended output (`-x` option). Look for average service times, `svctm`, greater than 20 msec for long durations and high average wait times, `await`, as indicators of performance problems.

oswmeminfo

OSWbb reads the `/proc/meminfo` file at the specified interval and stores the data in this directory. Information about available memory, `MemTotal`, and swap, `SwapTotal`, is included in this file.

oswmpstat

OSWbb runs the `mpstat` utility at the specified interval and stores the data in this directory. Be aware of involuntary context switches and the number of times a CPU failed to obtain a mutex. Values consistently greater than 200 per CPU cause system time to increase.

oswnetstat

OSWbb runs the `netstat` utility at the specified interval and stores the data in this directory. Each protocol type has a specific set of measures associated with it. Network analysis requires evaluation of these measurements on an individual level and all together to examine the overall health of the network communications.

The information in the upper section of the report helps diagnose network problems when there is connectivity but response is slow. The lower section of the report contains protocol statistics. The TCP protocol is used more often than UDP in Oracle database and applications. Many performance problems associated with the network involve the retransmission of the TCP packets. Some implementations for RAC use UDP for the interconnect protocol, instead of TCP. The statistics in the lower section of the report are not divided up on a per-interface basis so you need to compare these to the interface statistics in the upper portion of the report.

oswprvtnet

Information about the status of RAC private networks is collected and stored in this directory only if you have configured private network tracing. This requires you to manually add entries for these private networks into an executable file named `private.net` file located in the `oswbb` directory.

An example of what this file looks like is named `Exampleprivate.net` with samples for each operating system: `solaris`, `linux`, `aix`, `hp`, and so on, in the `oswbb` directory. This file can be edited and renamed `private.net` or a new file named `private.net` can be created. This file contains entries for running the `traceroute` command to verify RAC private networks. The following is an example of a `private.net` entry on Linux:

```
traceroute -r -F node1
traceroute -r -F node2
```

In this example, `node1` and `node2` are two nodes in addition to the `hostnode` of a three-node RAC cluster. If the `private.net` file does not exist or is not executable, then no data is collected and stored under the `oswprvtnet` directory. Review the collected data to ensure that the network interface is up and responding and that the network is reachable. If `traceroute` indicates that the target interface is not on a directly connected network, validate that the address is correct or the switch it is plugged in to is on the same VLAN.

OSWbb Diagnostic Data Output

- `oswps`: Contains the output from the `ps` utility
- `oswslabinfo`: Contents of the `/proc/slabinfo` file
  - Contains statistics on the kernel slab cache
- `oswtop`: Contains the output from the `top` utility
- `oswvmstat`: Contains the output from the `vmstat` utility
- `oswifconfig`: Contains the output from the `ifconfig` utility

The remaining data collection directories are described as follows.

oswps

OSWbb runs the `ps` command at the specified interval and stores the data in this directory. The `ps` command lists all the processes currently running on the system and provides information about CPU consumption, process state, priority of the process, and other information. OSWbb runs the command with the `-elf` option.

The information in the `ps` command is helpful supporting information for RAC diagnostics. For example, the status of a process before a system crash might be important for root cause analysis. To discover the amount of memory that a process consumes is another example of how this data can be used.

oswslabinfo

OSWbb reads the `/proc/slabinfo` file at the specified interval and stores the data in this directory. Frequently used objects in the Linux kernel have their own cache. This file gives statistics on the kernel slab cache.

For each slab cache entry, the file includes the cache name, the number of currently active objects (memory blocks), the total number of available objects, the size of each object in bytes, the number of pages with at least one active object, the total number of allocated pages, and the number of pages per slab.

oswtop

OSWbb runs the `top` utility at the specified interval and stores the data in this directory.

The load average line displays the load averages over the last 1, 5, and 15 minutes. Load average is defined as the average number of processes in the run queue. A runnable UNIX process is one that is available right now to consume CPU resources and is not blocked on I/O or on a system call. The higher the load average, the more work your machine is doing.

The three numbers are the average of the depth of the run queue over the last 1, 5, and 15 minutes. It is important to determine what the average load of the system is through benchmarking and then look for deviations. A dramatic rise in the load average can indicate a serious performance problem.

The tasks line displays the total number of processes running at the time of the last update. It also indicates how many processes exist, how many are sleeping (blocked on I/O or a system call), how many are stopped (someone in a shell has suspended it), and how many are actually assigned to a CPU. Like load average, the total number of processes on a healthy machine usually varies just a small amount over time. Suddenly having a significantly larger or smaller number of processes could be a warning sign.

The memory line reflects how much real and swap memory your system has, and how much is free. Real memory is the amount of RAM installed in the system, or the physical memory. Swap is virtual memory stored on the machine's disk. Performance deteriorates when a computer runs out of physical memory and starts using swap space.

Look for a large run queue. A large number of processes waiting in the run queue might be an indication that your system does not have sufficient CPU capacity. Also look for processes that are consuming lots of CPU, these processes can possibly be tuned.

oswvmstat

OSWbb runs the `vmstat` utility at the specified interval and stores the data in this directory. Again, when trying to determine the cause of performance problems, a large run queue can indicate CPU saturation. Also look at CPU usage to determine whether more CPUs are required. Memory bottlenecks are determined by the scan rate. If this rate is continuously over 200 pages per second, then there is a memory shortage. Disk problems might exist if the number of processes blocked exceeds the number of processes on the run queue.

oswifconfig

OSWbb runs the `ifconfig –a` utility at the specified interval and stores the data in this directory. The `ifconfig` command displays the current status of network interfaces. The `ifconfig -a` command utility is most commonly used to troubleshoot RAC network interface issues. The output of this command is used with the output of `netstat` and `private.net` to diagnose any network interface issues on your server.

OSWatcher Analyzer (OSWbba)

- OSWbba is a graphing and analysis utility that is included with OSWbb v4.0.0 and higher.
- OSWbba graphically displays data collected, and generates reports.
- OSWbba includes a built-in analyzer to provide details on performance problems.
  - The ability to create a graph and analyze this information relieves you of manually inspecting all the files.
- To start OSWbba, use the following command:

```
# java -jar oswbba.jar -i ~/oswbb/archive
```

- The OSWbba menu provides options to graph and analyze the collected data.

OSWatcher Analyzer (OSWbba) is a graphing and analysis utility that comes bundled with OSWbb v4.0.0 and higher. OSWbba allows you to graphically display the data that is collected, to generate reports containing these graphs, and provides a built-in analyzer to analyze the data and provide details on any performance problems that it detects. The ability to create a graph and analyze this information relieves you of manually inspecting all the files.

OSWbba replaces the OSWg utility. This was done to eliminate the confusion caused by having multiple tools in support named OSWatcher. OSWbba is supported only for data collected by OSWbb and no other tool.

OSWbba is written in Java and requires a minimum of Java Version 1.4.2 or higher. OSWbba can run on any UNIX X Windows or PC Windows platform. OSWbba uses Oracle Chartbuilder, which requires an X Windows environment.

OSWbba parses all the OSWbb `vmstat`, `iostat`, and `top` utility log files contained in the `oswbb/archive` directory. When the data is parsed, you are presented with a command-line menu that has options for displaying graphs, creating binary GIF files of these graphs, and generating an HTML report containing all the graphs with a narrative on what to look for, and the ability to self-analyze the files that OSWbb creates.

OSWbba requires no installation. It comes shipped as a stand-alone Java JAR file with OSWbb v4.0.0 and higher.

Starting OSWbba

Before starting the OSWbba utility, run the following command to ensure that you have Java Version 1.4.2 or higher installed on your system. In this example, the version is `1.7.0_51`:

```
# java –version
java version "1.7.0_51"
...
```

OSWbba requires an input directory to run. This input directory is the fully qualified path name of the `archive` directory containing the OSWbb logs. The `archive` directory must have the same directory structure as the `archive` directory for OSWbb. It must contain the subdirectories; `oswvmstat`, `oswiostat`, `oswps`, `oswtop`, and `oswnetstat`. Use the `-i` *<archive_directory>* option to specify the input directory:

```
# java -jar oswbba.jar -i ~/oswbb/archive
Starting OSW Analyzer V7.3.1
...
Parsing Data. Please Wait...
Parsing Completed.
```

After the parsing completes, the following menu is displayed, providing options to create a graph and analyze the collected data:

```
Enter 1 to Display CPU Process Queue Graphs
Enter 2 to Display CPU Utilization Graphs
Enter 3 to Display CPU Other Graphs
Enter 4 to Display Memory Graphs
Enter 5 to Display Disk IO Graphs
Enter 6 to Generate All CPU Gif Files
Enter 7 to Generate All Memory Gif Files
Enter 8 to Generate All Disk Gif Files
Enter L to Specify Alternate Location of Gif Directory
Enter T to Alter Graph Time Scale Only (Does not change ...)
Enter D to Return to Default Time Scale
Enter R to Remove Currently Displayed Graphs
Enter A to Analyze Data
Enter S to Analyze Subset of Data (Changes analysis dataset ...)
Enter P to Generate A Profile
Enter X to Export Parsed Data to File
Enter Q to Quit Program

Please Select an Option:
```

The first three options display graphs of specific CPU components of `vmstat`. All options are described as follows:

- `Option 1` – Displays the process run, wait, and block queues
- `Option 2` – Displays CPU utilization graphs for system, user, and idle
- `Option 3` – Displays graphs for context switches and interrupts
- `Option 4` – Displays memory graphs for free memory and available swap
- `Option 5` – Uses the extended disk statistics option of `iostat` to display a list of all devices. The device name along with the average service time of each device is listed. You can then select one of the devices from the list. Graphs are available for reads/second, writes/second, service time, and percent busy. Example:

```
The Following Devices and Average Service Times Are Ready to
   Display:
   Device Name             Average Service Times in
      Milliseconds
   xvda                    0.03258620689655172
   scd0
   xvdb
   xvdd
Specify A Case Sensitive Device name to View (Q to exit):
```

- `Options 6,7,8` – Generate images of the graph for the specific category (CPU, memory, disk) to a file. The file is created in the OSWbba directory by default.
- `Option L` – Allows you to specify an alternative location for the image files that you create using options 6, 7, and 8
- `Option T` – Allows you to specify a different subset of time to graph. The default time span is based on the entire time span of the logs. For example, if OSWbb keeps the last 48 hours of logs in the archive, the default graph contains all 48 hours of data. You can specify to graph a two-hour period, for example, out of the entire 48-hour collection.
- `Option D` – Resets the graphing time scale back to the time encompassing the entire log collection
- `Option R` – Removes all previously displayed graphs from the screen
- `Option A` – Analyzes the files in the archive and produces a report
- `Option S` – Analyzes a subset of data
- `Option P` – Generates an HTML profile
- `Option X` – Exports parsed data to a file
- `Option Q` – Exits the program

Analyzing OSWbb Archive Files

- Start the analyzer from the OSWBBA directory.
- Select `Option A` from the OSWbba menu.
- You can also run the analyzer from the command line:

```
# java -jar oswbba.jar -i ~/oswbb/archive -A
```

- The analyzer output is divided into eight sections:
  - Section 1: System Status
  - Section 2: System Slowdown
  - Section 3: System General Findings
  - Section 4: CPU Detailed Findings
  - Section 5: Memory Detailed Findings
  - Section 6: Disk Detailed Findings
  - Section 7: Network Detailed Findings
  - Section 8: Process Detailed Findings

Select Option A from the OSWbba menu to analyze the files in the `archive` directory and produce a report. You need to be in the directory where OSWbba is installed to run the analyzer. You can also run the analyzer directory from the command line by including the `-A` option:

```
# java -jar oswbba.jar -i ~/oswbb/archive -A
A new analysis file analysis/host...txt has been created.
```

The following is a sample analysis file name:

```
# ls ~/oswbb/analysis
host...txt
```

The analyzer output is divided into sections for easy readability.

- Section 1: System Status
- Section 2: System Slowdown
- Section 3: System General Findings
- Section 4: CPU Detailed Findings
- Section 5: Memory Detailed Findings
- Section 6: Disk Detailed Findings
- Section 7: Network Detailed Findings
- Section 8: Process Detailed Findings

Section 1 provides a quick status of each major subsystem. Example:

```
Section 1: System Status
...
Subsystem        Status
CPU              OK
MEMORY           OK
I/O              OK
NET              OK
```

Other possible status values are `Warning`, `Critical`, and `Unknown`.

Section 2 provides a system slowdown summary ordered by impact. This section lists:

- Slowdown time and duration
- Most likely causes of slowdown
- Offending process
- Advice on what to do

Section 3 provides system general findings such as the following:

- CPU run queue observed very high spikes.
- Severe memory swapping was observed.

Section 4 provides a summary of CPU metrics collected in the archive. The following metrics are reported:

- Number of snapshots in the archive
- Number of snapshots with a high CPU run queue
- Times when the run queue was reported high
- `root` processes with high CPU consumption
- Other processes with high CPU consumption

Section 5 provides a summary of memory metrics collected in the archive. The following metrics are reported:

- Process swap queue
- Scan rate
- Snapshot times when scan rate was high

Section 6 provides detailed disk findings. Only devices that are busy more than 50% are included in the report. The following metrics are reported:

- Device percent busy for devices with percent busy > 50%
- Device service time for devices with service time > 10 msec
- Device throughput for devices with percent busy > 50%

Section 7 provides detailed network findings including data link findings, IP findings, UDP findings, and TCP findings.

Section 8 provides detailed process findings ordered by time as well as top processes increasing memory.

Enterprise Manager Ops Center

Enterprise Manager Ops Center:

- Provides management services for operating systems, virtual machine, servers, storage, and networks
- Enables you to provision, update (patch), monitor, and manage assets in one or more data centers from a single console
- Includes built-in integration with My Oracle Support, with automatic service request generation
- Has the following architecture components:
  - Enterprise Controller
  - Proxy Controller
  - Agent Controller
  - User Interface
  - Knowledge Base

Enterprise Manager Ops Center provides management services for operating systems (including Oracle Linux), virtual machines, servers, storage, and networks. You can provision, update (patch), monitor, and manage the physical and virtual managed assets in one or more of your data centers from a single console, regardless of where the asset is located. It includes built-in integration with My Oracle Support, with automatic service request generation.

Key Features

The software provides features tailored for administrating the data center infrastructure, including the following:

- **Dashboards:** View assets including a graphical representation of the status and membership.
- **Incident Management:** Monitor assets according to rules and thresholds that you set.
- **Integration with Enterprise Manager Cloud Control:** View configuration, health and performance information, and incidents of managed assets using either product.
- **Profiles For Assets:** Create software profiles and operational profiles that contain your custom executable scripts.
- **Operational Plans:** Deploy a single script as an operational profile. You can use the scripts to perform specific tasks in your environment, such as configuration options, or to assist in incident management.

- **Deployment Plans:** Combine one or more profiles and scripts to create a multi-task plan that provisions operating systems or firmware efficiently and consistently.
- **Plan Management:** Use the provided default templates, profiles, and plans to create and deploy plans.
- **Hardware Management:** Update system component firmware and track hardware configuration changes over time.
- **Virtualization Management:** Manage virtual assets such as Oracle Solaris Zones, Oracle VM Server for SPARC, Oracle VM Server for x86, and their guests.
- **Reports:** Create reports for assets and activities and export the reports as files.

Architecture

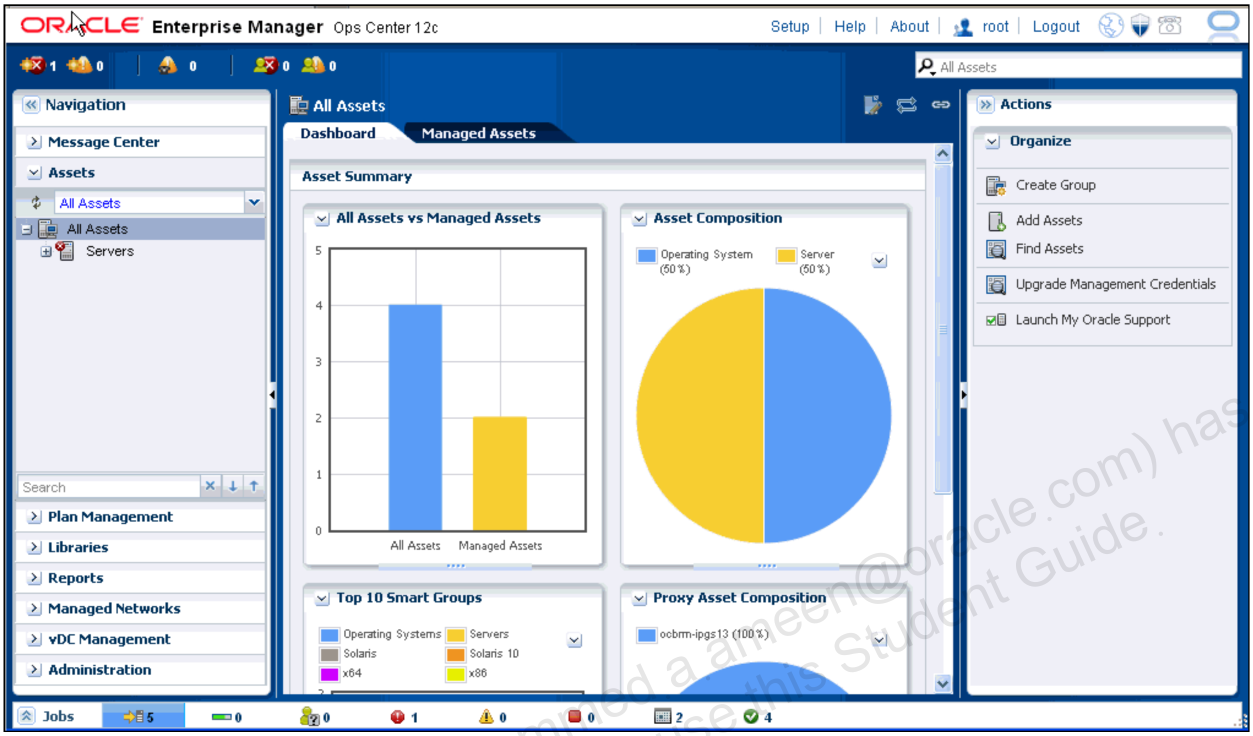
The Enterprise Controller, Proxy Controller, Agent Controller, and user interface are the major architectural components, along with Knowledge Base, which is hosted by Oracle Corporation and accessed through the Internet.

- **Enterprise Controller:** This is the central server for Enterprise Manager Ops Center. All operations, or jobs, are initiated from the Enterprise Controller. It manages firmware and operating system images, plans, profiles, and policies. It connects to the Internet to get access to contract information, to create service requests, and to download updates. You can also operate the software in Disconnected mode if your site policy does not allow an Internet connection.
- **Proxy Controller:** This distributes the operation load and provides for fan-out capabilities to minimize network load. It links the managed assets to the Enterprise Controller and performs operations that must be located close to the managed assets, such as operating-system provisioning. You can install the Proxy Controller and Enterprise Controller software on the same system, but to enhance performance and scalability, the preferred method is to install the Proxy Controller on a separate machine.
- **Agent Controllers:** The Agent Controller is lightweight Java software that identifies the asset and responds to a Proxy Controller. When an operating system is agent-managed, the agent receives the command from its Proxy Controller, performs the required action, and notifies the Proxy Controller of the results. When an operating system is agentlessly managed, the Proxy Controller uses SSH to perform tasks and to monitor the operating system. You can use many of the monitoring and management features without installing an Agent Controller on the operating system. Hardware management does not require the Agent Controller. Instead, a Proxy Controller runs commands on the hardware system and reports the results to the Enterprise Controller.
- **Knowledge Base and Package Repository:** This stores metadata about Oracle Solaris and Linux operating system components. The metadata includes patch dependencies, standard patch compatibilities, withdrawn patches, and rules for download and deployment. Knowledge Base keeps track of the URLs for the operating systems and retrieves the components from the appropriate vendor download site.

The entire Ops Center product is included as a default part of all Systems support agreements. This means that every customer of Oracle’s servers, storage, network equipment, operating systems, and virtualization technology can add Ops Center to their data center management suite.

To see a demonstration of the product, visit http://www.youtube.com/watch?v=tRWTWDBUIQU.

Enterprise Manager Ops Center GUI



This slide displays the Enterprise Manager Ops Center browser interface. The format of the information is in text, tables, graphs, and charts, and the information is organized into hierarchies and tabs. You can view the information and perform actions according to the role that you have been assigned.

The interface is consists of five panes:

- **Masthead:** The top pane displays the global functions and information about the Enterprise Manager Ops Center instance.
- **Navigation Pane:** The left pane consists of several drawers that display assets and objects that are managed by the Enterprise Manager Ops Center instance.
- **Actions Pane:** The right pane displays the actions that operate on the object currently selected in the Navigation pane. The actions of the Actions pane are enabled or disabled based on the state of the object or your role.
- **Jobs Pane:** The bottom pane displays the number of jobs in Enterprise Manager Ops Center, categorized by the status of respective jobs.
- **Center Pane:** This pane displays detailed information of the object that is currently selected in the Navigation pane.

To learn more about an incident, place your cursor over the incident icons in the left-side corner of the user interface.

Enterprise Manager Ops Center Provisioning

- Provisioning Firmware
  - Updates firmware from the Enterprise Controller library to managed servers, chassis, or storage devices
  - Action is controlled by a Firmware Profile.
- Provisioning Operating Systems
  - Enables you to install supported operating systems from the software library on the Enterprise Controller onto managed assets
  - Action is controlled by an OS Provisioning Profile.
- Applying Deployment Plans
  - Apply profiles in sequence to combine OS provisioning, updates, software installation, script execution and monitoring

Enterprise Manager Ops Center facilitates automated firmware provisioning and OS provisioning by using a combination of image libraries, profiles, and deployment plans. This allows you to perform consistent installation of an asset by using a deployment plan outlining a combination of an OS Provisioning, OS Update, Software Installation and Update, and post-install scripting, and assigning a monitoring profile.

The images, profiles, and deployment plans are stored on the Enterprise Controller.

Provisioning can be performed on a single asset, or a group of assets.

Enterprise Manager Ops Center Patching

- Installs, updates, and removes software and patches
- Reduces the complexity of updating a large number of systems
- Automates patching without user interaction
- Automatically manages the patch and software dependencies
- Provides version control and rollback capability
- Supports an update simulation capability
- Action controlled by an OS Update Profile

Enterprise Manager Ops Center is designed to reduce the complexity of updating a large number of systems, standardize the patch installation process, minimize down time, track changes, and automate patching without user interaction.

You control the update process, the level of automation, the scheduling, and the number of concurrent updates. You can apply customized controls for one system or a group of systems and schedule the updates to deploy during periods of low usage.

Enterprise Manager Ops Center Monitoring

- Enterprise Manager Ops Center provides monitoring capabilities for:
  - Hardware
  - Operating Systems
  - Storage Devices
  - Switches
- You can configure thresholds on system-defined parameters to trigger alerts.
  - OS performance statistics
  - Hardware status (temperature, fan speed, voltage, and so on)
  - Power consumption
- Monitoring profiles can be defined and assigned to assets.

The software is designed to make it easy to monitor and manage a large numbers of assets from a single console. It provides end-to-end server awareness and robust monitoring capabilities for the hardware, storage devices, and operating systems in your data center.

You can track system-defined parameters for hardware power consumption, hardware status (temperature, fan speed, and voltage), and key OS statistics (load, CPU, memory.)

For more robust monitoring, the software uses editable rules and event thresholds to monitor your systems. A rule defines a specific monitored resource and the rule parameter defines when an alert is triggered.

Spacewalk

- Spacewalk is a full lifecycle management tool for RPM-based Linux distributions.
  – The community project can be found at https://fedorahosted.org/spacewalk/.
  – Documentation is available at http://linux.oracle.com/documentation/spacewalk/.
- The RPMs for Oracle Linux are available at Public Yum.
  – Spacewalk Server for OL6 x86_64
  – Spacewalk Client for OL7 x86_64
  – Spacewalk Client for OL6 i386 and x86_64 architectures
  – Spacewalk Client for OL5 i386 and x86_64 architectures
- Spacewalk Server requires either Oracle Database 11*g*R2 or PostgreSQL as the back-end database.

Spacewalk is the open source upstream project of Red Hat Satellite Server. It is a full lifecycle management tool for RPM-based Linux distributions. Spacewalk is similar to Oracle Enterprise Manager in that it allows you to install and update software on your systems, provision systems, and monitor and manage your Oracle Linux systems. The community project can be found at https://fedorahosted.org/spacewalk/. Documentation is available at http://linux.oracle.com/documentation/spacewalk/.

Oracle has made a few changes to Spacewalk to ensure easy and complete support for Oracle Linux. The Spacewalk Server is available for Oracle Linux 6 x86_64 architecture at:

- http://public-yum.oracle.com/repo/OracleLinux/OL6/spacewalk20/server/

The back-end database for Spacewalk Server can be either Oracle Database 11*g*R2 or PostgreSQL. Oracle only supports the use of Oracle Database 11*g*R2 as the back-end database.

Spacewalk Client is available for Oracle Linux 7 x86_64, Oracle Linux 6 i386 and x86_64, and Oracle Linux 5 i386 and x86_64. The RPMs are available at:

- http://public-yum.oracle.com/repo/OracleLinux/OL7/spacewalk22/client
- http://public-yum.oracle.com/repo/OracleLinux/OL6/spacewalk20/client/
- http://public-yum.oracle.com/repo/OracleLinux/OL5/spacewalk20/client/

Spacewalk Features and Functionality

- Manages software updates
- Allows update staging through multiple environments
- Central web-based administration interface
- Allows scheduling of mass updates to thousands of servers
- Allows delivery of software updates targeting specific errata
- Mirrors content from ULN and Public Yum locally
- Can manage internal and 3rd-party Yum repositories
- Geographic distribution using Spacewalk Proxy and Inter-Spacewalk Sync (ISS)

Spacewalk provides software content management. It allows update staging through multiple environments, such as development environments, test environments, quality assurance (QA), near-production, and production environments. You can promote packages through your environments. And you can promote servers through your environments.

Spacewalk has a web based administration interface. It also includes comprehensive application program interface (API) and command-line tools. Spacewalk allows mass updates of thousands of servers at once. You can group servers using System Groups and assign servers to one or more system groups. Spacewalk allows delivery of software updates from specific channels and also by errata or common vulnerabilities and exposures (CVEs). You can have Spacewalk send the patches that resolve a CVE to the affected servers.

Spacewalk can manage internal and 3rd party Yum repositories. The repositories do not need to come from Oracle, or from Red Hat, or from an official repository source. You can set up internal repositories within Spacewalk and manually push packages into them. You can set up repositories to sync packages from an upstream source.

Spacewalk supports geographic distribution. Instead of having multiple clients all connecting to the same server, you can set up multiple complete Spacewalk instances and use Inter-Spacewalk Sync (ISS) to link them together. The other option is to use Spacewalk Proxy. This is a proxy that sits between servers and the main Spacewalk instance. Spacewalk Proxy can provide packages and first-level processing for downstream clients.

Spacewalk Features and Functionality

- Provisioning of new physical and virtual servers
  - Supports PXE-based deployments using kickstart
  - Automatically registers new servers
  - Can redeploy existing servers using PXE
  - Can create new virtual instances
- Monitoring
  - Spacewalk clients regularly report their status to Spacewalk.
  - OSAD provides near real-time triggering of actions on clients.
  - `rhncfg` provides local configuration file management and remote actions.
- Auditing
  - Spacewalk can trigger OpenSCAP-based XCCDF testing and provide results.

ORACLE

Spacewalk supports provisioning of both physical and virtual servers. Physical server provisioning is based on Preboot Execution Environment (PXE) or booting over the network. Spacewalk supports multiple kickstart configurations. You can have a kickstart configuration for different versions of Oracle Linux and a kernel-based virtual machine (KVM) configuration for virtual instances. If you enable the Spacewalk client channel, it will automatically register your server with Spacewalk as part of the kickstart process.

Spacewalk can create Xen and KVM virtual instances. Note that it is not supported on Oracle VM. You cannot run the Spacewalk client that creates virtual images on Oracle VM. It also cannot replace Oracle VM Manager but it can create KVM instances on Oracle Linux. It uses Spacewalk Proxy so you do not need to kickstart from the central location. Spacewalk configure its proxies to do local kickstarting of the geographical locations.

Spacewalk supports monitoring. By default, Spacewalk clients report their status to the Spacewalk server every 4 hours. There is an additional client tool called OSAD (Open Source Architecture Daemon) that provides triggering of actions on clients. There is also a local configuration file manager called `rhncfg` that allows you to send files and directories from your Spacewalk server down to your Spacewalk client. It also supports remote actions so you can send commands down to the client.

From an auditing perspective, Spacewalk can trigger OpenSCAP-based XCCDF testing on a daily or weekly basis.

Quiz

Which of the following utilities allows you to collect system information for sending to Oracle support?

a. `sosreport`

b. `sar`

c. OSWbb

d. `strace`

Summary

In this lesson, you should have learned how to:

- Use the `sosreport` utility
- Use the `iostat`, `mpstat`, `vmstat`, `sar`, `top`, `iotop`, and `strace` utilities
- Use the `netstat` and `tcpdump` utilities
- Use the Wireshark network analyzer GUI
- Use the OSWatcher Black Box (OSWbb) tool
- Use OSWatcher Analyzer (OSWbba)
- Describe Enterprise Manager Ops Center
- Describe Linux Patch and Provisioning using Enterprise Manager Ops Center
- Describe Spacewalk

Practice 20: Overview

The practices for this lesson cover the following topics:

- Using `sosreport` to collect system information
- Using standard Linux performance monitoring tools
- Installing and using OSWatcher
- Using OSWatcher Analyzer

21

System Logging

Objectives

After completing this lesson, you should be able to:

- Describe Oracle Linux 7 system logging options
- Describe the contents of the `rsyslog` configuration file
- Describe `rsyslog` filter options
- Describe facility/priority-based filters
- Describe `rsyslog` actions
- Describe `rsyslog` templates
- Configure log rotation
- Describe `logwatch`
- Describe `journald`
- Use the `journalctl` utility

System Logging: Introduction

- Log files store system, kernel, service, and application messages.
  – Most log files are located in the `/var/log/` directory.
- Some log files are controlled by the `rsyslogd` daemon.
  – `/etc/rsyslog.conf` is the main configuration file.
  – It contains global directives, modules, and rules.
- With Oracle Linux 7, log files can also be managed by the `journald` daemon.
  – `journald` is a component of `systemd`.
  – `journald` captures various system messages, indexes them, and stores them in `/run/log/journal/`.
  – Use the `journalctl` utility to view the journal logs.

Log files contain messages about the system, the kernel, services, and applications. Most of these log files are located in the `/var/log/` directory. Some log files are controlled by the `rsyslogd` daemon. The main configuration file for `rsyslogd` is `/etc/rsyslog.conf`, which contains global directives, modules, and rules.

- **Global Directives:** Configuration options that apply to the `rsyslogd` daemon
- **Modules:** Dynamically loaded modules that provide additional functionality and associated configuration directives
- **Rules:** Define a filter, which is a subset of `rsyslog` messages; and an action, which specifies what to do with the messages

With Oracle Linux 7, log files can also be managed by the `journald` daemon. The `journald` daemon is a component of `systemd` that captures various system messages, indexes them, and stores them in the `/run/log/journal/` directory. You can use the `journalctl` utility to view the journal logs.

This lesson begins with a discussion of `rsyslogd` and ends with a discussion of `journald`.

rsyslog Configuration

- Global Directives:
  - Configuration options that apply to the rsyslogd daemon
  - All configuration directives must begin with a dollar sign ($).

```
$IncludeConfig /etc/rsyslog.d/*.conf
```

- Modules:
  - Are dynamically loaded using the $ModLoad global directive
  - Provide additional functionality and configuration directives
  - Categories of modules include *Input*, *Output*, *Parser*, *Message modification*, *String generator*, and *Library*.
- Rules:
  - Specifies a *filter* (cron.*) and *action* (log all cron messages to /var/log/cron)

```
cron.*   /var/log/cron
```


The /etc/rsyslog.conf configuration file contains global directives, modules, and rules.

Global Directives

Global directives specify configuration options that apply to the rsyslogd daemon. All configuration directives are specified on a single line and must begin with a dollar sign ($). The following is an example of a global directive to include all configuration files found in the /etc/rsyslog.d directory:

```
$IncludeConfig /etc/rsyslog.d/*.conf
```

A list of all available configuration directives and their descriptions can be found at http://www.rsyslog.com/doc/rsyslog_conf_global.html.

Modules

rsyslog has a modular design. This enables functionality to be dynamically loaded from modules. Each module provides configuration directives. Modules must be loaded for their configuration directives and functionality to be available. The following example uses the $ModLoad global directive to load the imjournal module:

```
$ModLoad imjournal
```

The imjournal module transfers data acquired by journald to rsyslogd. The omjournal module is available to transfer data from rsyslogd to journald.

Following describes the main categories of `rsyslogd` modules:

- **Input modules:** Gather messages from various sources. Input module names always start with the `im` prefix (examples: `imfile`, `imjournal`).
- **Output modules:** Output messages to various targets such as across a network, storing them in a database, or encrypting them. Output module names always start with the `om` prefix (examples: `omsnmp`, `omjournal`).
- **Parser modules:** Use the message parsers to parse the message content of any received messages. The name of a parser module always starts with the `pm` prefix (examples: `pmciscoios`, `pmlastmsg`).
- **Message modification modules:** Change the content of an `rsyslog` message. Names of these modules always start with the `mm` prefix (examples: `mmcount`, `mmfields`).
- **String generator modules:** Generate strings based on the message content and co-operate with the template feature provided by `rsyslog`. The name of a string generator module always starts with the `sm` prefix (examples: `smfile`, `smfwd`).

Any output that is generated by `rsyslog` can be modified and formatted by using templates.

- **Library modules:** Library modules provide functionality for other loadable modules. These modules cannot be configured and are loaded automatically by `rsyslog` when needed.

Messages are received by input modules and then passed to one or many parser modules, which generate the in-memory representation of the message and might also modify the message itself. The internal representation is passed to output modules, which might output a message and also modify message object content.

A list of available modules and detailed descriptions can be found at http://www.rsyslog.com/doc/rsyslog_conf_modules.html.

Rules

Every rule consists of two fields, a *filter* field and an *action* field. A *filter* specifies a subset of `rsyslog` messages to select. An *action* specifies what to do with the selected messages. To define a rule in the `/etc/rsyslog.conf` configuration file, define both a *filter and* an *action* on one line and separate them with one or more spaces or tabs.

Following are examples of rules defined in the `/etc/rsyslog.conf` file. Lines beginning with the `#` sign are comments.

```
# Log all kernel messages to the console.
kern.*   /dev/console

# Log all the mail messages in one place.
mail.*   /var/log/maillog

# Log cron stuff
cron.*   /var/log/cron
```

`rsyslog` Filter Options

There are three different ways to filter `rsyslog` messages:

- Facility/priority-based filters:
  - Filters are based on two conditions: facility and priority.
  - Facility specifies the subsystem that produces the message.
  - Priority represents the priority of the message.
- Property-based filters:
  - Filter by comparing a property of the message to a value

```
:msg, contains, "error"
```

- Expression-based filters:
  - Filter according to arithmetic, Boolean, or string operations
  - Use `rsyslog` scripting language. Syntax:

```
if EXPRESSION then ACTION else ACTION
```


The `rsyslogd` daemon offers three different ways to filter `rsyslog` messages:

Facility/Priority-Based Filters

Facility/priority-based filters filter `rsyslog` messages based on two conditions: *facility* and *priority*. Facility specifies the subsystem that produces the message. Examples of facilities include `mail`, `kernel`, and `cron`. Priority represents the priority of the message. Examples of priorities include `debug (7)`, `warning (4)`, and `alert (1)`.

Property-Based Filters

Filter `rsyslog` messages by any property, such as `timegenerated` or `msg`. You can compare a property to a value by using one of several property-based compare operations. Compare operations include `contains`, `isequal`, and `startswith`. The following example filters for messages that contain the string "`error`" in the message text (`msg`):

```
:msg, contains, "error"
```

Expression-Based Filters

Select `rsyslog` messages according to arithmetic, Boolean, or string operations by using an `rsyslog` scripting language. The following shows the basic syntax of expression-based filters:

```
if EXPRESSION then ACTION else ACTION
```

Facility/Priority-Based Filters

Messages are filtered based on two conditions: Facility and priority.

- Syntax to create a filter (or selector):

```
Facility.Priority
```

- Select all `auth` `rsyslog` messages with any priority:

```
auth.*
```

- Select all `mail` `rsyslog` messages with priority `err` and higher:

```
mail.err
```

- Select all `user` `rsyslog` messages except those with `info` or `debug` priority:

```
user.!info,!debug
```

ORACLE

Facility/priority-based filters select `rsyslog` messages based on two conditions: *facility* and *priority*. A facility-priority pair is called a selector. To create a selector, use the syntax:

```
Facility.Priority
```

Facility

Facility specifies the subsystem that produces a specific `rsyslog` message and can be represented by one of the following keywords:

- **`auth/authpriv`**: Security/authorization messages
- **`cron`**: `crond` messages
- **`daemon`**: Other system daemons
- **`kern`**: Kernel messages
- **`lpr`**: Line printer subsystem
- **`mail`**: Mail system
- **`news`**: Network news subsystem
- **`syslog`**: Messages generated internally by `rsyslogd`
- **`user`**: User-level messages
- **`uucp`**: UUCP subsystem
- **`local0 through local7`**: Local use

Priority

Priority can be represented by one of these keywords (listed in an ascending order). All messages of the specified priority and higher are logged according to the given action.

- **`debug`**: Debug-level messages
- **`info`**: Informational messages
- **`notice`**: Normal bug significant condition
- **`warning`**: Warning conditions
- **`err`**: Error conditions
- **`crit`**: Critical conditions
- **`alert`**: Action must be taken immediately.
- **`emerg`**: System is unstable.

The following are examples of facility/priority-based selectors. To select all `mail` messages with priority `err` and higher:

```
mail.err
```

Special characters can be used. Use an asterisk (`*`) to specify all facilities or priorities. For example, to select all `auth` messages with any priority:

```
auth.*
```

Use a comma (`,`) to specify multiple facilities and priorities. For example, to select both the `uucp` and `news` facilities with priority of `warning` or higher:

```
uucp,news.warning
```

Use a semicolon (`;`) to define multiple selectors on one line. Example:

```
*.info;mail.none;auth.none;cron.none
```

Use an equal sign (=) to specify a single priority. All other priorities are ignored. For example, to select `cron` messages of only `emerg` priority:

```
cron.=emerg
```

Precede a priority with an exclamation mark (`!`) to select all `rsyslog` messages except those with the defined priority. The following example selects all `user` messages, except those with the `info` or `debug` priority:

```
user.!info,!debug
```

rsyslog Actions

- Actions specify what to do with the filtered messages.
- Options include:
  - Save rsyslog messages to log files.
  - Send rsyslog messages over the network.
  - Send rsyslog messages to specific users.
  - Execute a program.
  - Write rsyslog messages into a database.
  - Discard rsyslog messages.
- To save cron messages to /var/log/cron.log:

```
cron.* /var/log/cron.log
```

- To send rsyslog messages over the network:

```
*.* @example.com:18
```


Actions specify what to do with the messages filtered out by a selector. The following are some of the available actions.

Saving rsyslog Messages to Log Files

To save an rsyslog message to a log file, specify the absolute path to the log file after the selector. The following example selects all cron messages and the action saves them to the /var/log/cron.log log file:

```
cron.* /var/log/cron.log
```

You can specify an existing tty or /dev/console device to send rsyslog messages to standard output.

Sending rsyslog Messages over the Network

Use the following syntax to forward rsyslog messages to a remote machine:

```
@[zNUMBER]HOST:[PORT]
```

Use a single at sign (@) to specify UDP as the transport protocol. Use a double at sign (@@) to specify TCP. The optional *zNUMBER* field enables a level of zlib compression from 1 to 9. The *HOST* field specifies the receiving host. The optional *PORT* field specifies the port number on the receiving host.

For example, to forward messages to `192.0.2.101` using the UDP protocol:

```
*.* @192.0.2.101
```

To forward messages to port `18` on “`host02.example.com`” using the TCP protocol:

```
*.* @@host02example.com:18
```

Sending `rsyslog` Messages to Specific Users

Specify the username to send `rsyslog` messages to. Separate usernames with a comma (`,`) to specify more than one user. Use an asterisk (`*`) to send messages to every user that is currently logged on. The following example sends all kernel messages to user `joe`:

```
kern.* joe
```

Executing a Program

You can execute a program for selected `rsyslog` messages. To specify a program to be executed, prefix it with a caret character (`^`). Specify a template that formats the received message and passes it to the specified executable as a one-line parameter. The following example processes all kernel messages by the template `knl` and passes them on to the `knl-prog` program. Templates are discussed in the next slide.

```
kern.* ^knl-prog;knl
```

Write `rsyslog` Messages into a Database

You can use the database writer action to write selected `rsyslog` messages directly into a database table. The database writer uses the following syntax:

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;[TEMPLATE]
```

The *`PLUGIN`* field specifies the plug-in that performs the database writing. `rsyslog` provides support for MySQL and PostgreSQL databases. MySQL integration requires the `rsyslog-mysql` software package. PostgreSQL requires the `rsyslog-pgsql` package. You also need to load the `ommysql` module for MySQL and the `ompgsql` module for PostgreSQL.

Discarding `rsyslog` Messages

Use the tilde character (`~`) to discard selected messages. The following rule discards any `news` messages:

```
news.* ~
```

You can specify multiple actions for a selector by specifying subsequent actions on a new line and preceding the actions with an ampersand character (`&`). Specify the selector on the first action line. The following is an example of a rule with multiple actions:

```
kern.* joe
& ^knl-prog;knl
& @192.0.2.101
```

In the preceding example, all `kernel` messages are:

- Sent to user `joe`
- Processed by the template `knl` and passed on to the `knl-prog` executable
- Forwarded to `192.0.2.101` by using the UDP protocol

`rsyslog` Templates

Templates modify and format output generated by `rsyslog`.

- Syntax to create a template:

```
$template TEMPLATE_NAME,"text %PROPERTY% text", [OPTION]
```

- Templates can be used to generate dynamic file names:

```
$template DynamicFile,
"/var/log/%timegenerated%-test.log"
```

- Example of a template definition :

```
$template class, "Time: %timestamp%, Facility:
   %syslogfacility-text%, Priority: %syslogpriority-
   text%, Hostname: %hostname%, Message: %msg%\n"
```

- Example of using a template in a rule:

```
*.*   /var/log/logfile;class
```


Use templates to modify and format `rsyslog` output. The following is the syntax to create a template:

```
$template TEMPLATE_NAME,"text %PROPERTY% text", [OPTION]
```

The fields are described as follows:

- **`$template`**: Directive that defines a template
- ***`TEMPLATE_NAME`***: Name of the template
- **`"text"`**: Actual template text surrounded by quotation marks
- ***`%PROPERTY%`***: Specific message content surrounded by percent signs
- ***`OPTION`***: Specifies options that modify the template functionality

Templates can be used to generate dynamic file names. Specify a property as a part of the file path to create a new file for each unique property. For example, use the `timegenerated` property to generate a unique file name for each `rsyslog` message:

```
$template DynamicFile,"/var/log/%timegenerated%-test.log"
```

Specify the template name in a rule to modify `rsyslog` output. Dynamic files are represented by a template and a question mark (`?`) prefix. Example:

```
*.* ?DynamicFile
```

Properties

You can use properties inside a template to reference specific contents of an `rsyslog` message. Use the following syntax to define a property inside a template:

```
%PROPERTY_NAME[:FROM_CHAR:TO_CHAR:OPTION]%
```

The fields are described as follows:

- ***PROPERTY_NAME***: Name of a property
- ***FROM_CHAR*** and ***TO_CHAR***: Range of characters the specified property acts upon
- ***OPTION***: Property options

A list of available properties and descriptions can be found at http://www.rsyslog.com/doc/property_replacer.html.

The following property represents the entire message text of an `rsyslog` message:

```
%msg%
```

The following example represents the first two characters of the message text:

```
%msg:1:2%
```

The following property represents the host name in an `rsyslog` message:

```
%hostname%
```

The following property represents the facility from the message in text form:

```
%syslogfacility-text%
```

Template: Example

The following example defines a template named `class` that formats an `rsyslog` message to output the message's time stamp, facility in text form, priority in text form, host name, and message text, and ends with a new line:

```
$template class, "Time: %timestamp%, Facility: %syslogfacility-
text%, Priority: %syslogpriority-text%, Hostname: %hostname%,
Message: %msg%\n"
```

To use the template for `/var/log/logfile` messages, include the template name as follows:

```
*.*   /var/log/logfile;class
```

Configuring Log Rotation (`logrotate`)

- `logrotate` is a utility to automatically manage log files.
  - It runs as a daily `cron` job using the `/etc/cron.daily/logrotate` file.
- The `/etc/logrotate.conf` file is the global configuration file for all logs.
  - The `/etc/logrotate.d/` directory contains a separate configuration file for any specific log file.
- Configuration options include:
  - How often to rotate files
  - The number of rotated log files to keep
  - Scripts to run before or after rotating
  - Specify log files to be mailed
  - Enable compression of log files

Most log files are located in the `/var/log` directory. Some services such as `cups`, `httpd`, and `samba` have a directory within `/var/log` for their log files.

The `logrotate` utility helps manage log files automatically by rotating, compressing, mailing, and removing each as you specify. Rotating means saving a series of log files, renaming each file as a new one is saved. Log files are rotated so file sizes do not become too large. Rotating allows you to keep log information for future reference.

Some files in `/var/log` have numbers at the end of the file name. These numbers represent a rotated log with the time stamp added to the log file name.

Normally `logrotate` is run as a daily `cron` job using the `/etc/cron.daily/logrotate` file. The main configuration file for `logrotate` is `/etc/logrotate.conf`. There are also configuration files in the `/etc/logrotate.d` directory. You can configure how often to rotate files:

- Daily
- Weekly
- Monthly

You also can specify the number of rotated log files to keep. These parameters are configured in the `/etc/logrotate.conf` configuration file.

`/etc/logrotate.conf` File

The following is a sample `/etc/logrotate.conf` configuration file:

```
# cat /etc/logrotate.conf
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
#compress
```

In the example, log files are rotated weekly, rotated log files are kept for four weeks, and all rotated log files are compressed by `gzip` into the `.gz` format.

`/etc/logrotate.d/` Directory

You can create a separate configuration file for any specific log file in the `/etc/logrotate.d` directory and define any configuration options there. These options override the global options in `/etc/logrotate.conf` and also define additional options. Oracle Linux provides a few separate configuration files by default:

```
# ls /etc/logrotate.d
chrony  libvirtd      numad  samba    up2date  yum
cups    libvirtd.lxc  ppp    snapper  vsftpd
...
```

The following is an example of the `/etc/logrotate.d/chrony` configuration file:

```
# cat /etc/logrotate.d/chrony
/var/log/chrony/*.log {
     missingok
     nocreate
     sharedscripts
     postrotate
       /usr/libexec/chrony-helper command cyclelogs > /dev/nul...
     endscript
}
```

The options in the `/etc/logrotate.d/chrony` configuration file are described as follows:

- **`missingok`:** If the log file is missing, do not issue an error message.
- **`nocreate`:** New log files are not created.
- **`postrotate/endscript`:** The lines between these directives are executed after the log file is rotated.
- **`sharedscripts`:** The `postrotate` script runs only once, not once for each log that is rotated.

For a full list of directives and configuration options, refer to the `logrotate(8)` man page.

logwatch Utility

- logwatch is a utility to perform basic log file monitoring and analysis.
  - It runs as a daily cron job by using the /etc/cron.daily/0logwatch file.
- Configuration files are located in the following directory:
  - /etc/logwatch/conf/
- A HOWTO-Customize-Logwatch file exists in the following directory:
  - /usr/share/doc/logwatch-<*version*>/
- logwatch can also be run from command line. Example:

```
# logwatch --help
```


logwatch is a customizable log monitoring system. It goes through system logs for a given time period and reports on specific areas of interest.

It might be necessary to install the logwatch package. After it is installed, logwatch is configured by default to run each night from /etc/cron.daily/0logwatch and email a report to the root user.

You can customize the output of logwatch by modifying variables in the /etc/logwatch/conf/ directory. A HOWTO-Customize-Logwatch file exists in the /usr/share/doc/logwatch-<*version*>/ directory. Following are some of the options you can configure:

- Level of detail
- Log file to report on
- Name of a service to report on
- Username to mail the report to
- File name to save the report to

You can also run logwatch from the command line with various options. Run the following command to get information about using logwatch:

```
# logwatch --help
```

Introduction to `journald`

- Is a logging service included with `systemd`
- Collects and stores logging data in structured, indexed journals.
  - Supports advanced query options and faster search times
- Adds structured metadata to the messages that assist in troubleshooting
- Can be used together with, or in place of, `rsyslogd`
- Journals are nonpersistent by default.
  - Are stored in `/run/log/journal/`
- To configure persistent journal data storage:

```
# mkdir -p /var/log/journal
# systemctl restart systemd-journald
```


Log files can also be managed by the `journald` daemon, which is part of `systemd`. `journald` collects and stores logging information from the kernel, from system services, and from user processes. `journald` stores data in structured, indexed journals that support advanced query options and faster search times than traditional log files. `journald` adds structured metadata to the messages that assist in troubleshooting. `journald` can be used together with, or in place of, `rsyslogd`.

By default, the journal stores log data in `/run/log/journal`. The `/run` mount point is a `tmpfs` file system that is mounted at boot time.

```
# mount | grep run
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,mode=755)
```

A `tmpfs` file system stores its files in virtual memory. It is a temporary file system in the sense that data is lost at reboot or if the file system is unmounted. In addition, the amount of logged data depends on free memory. When you run out of free memory, the oldest entries in the journal are deleted.

You can make journal data persistent by creating the `/var/log/journal/` directory and then restarting the `systemd-journald` service.

```
# mkdir -p /var/log/journal
# systemctl restart systemd-journald
```

`journalctl` Utility

- Use the `journalctl` command to view the journal logs.

```
# journalctl
```

- Output is displayed one page at a time.
- Time stamps are converted to your local time zone.
- Priority of entries is visibly marked.
  - Entries with error priority and higher are red.
  - Entries with notice and warning priority are in bold font.
- The beginning of the boot process has a special entry.
- With no options, all log data is displayed.
- By default, oldest entries are listed first.
- A number of query options are available. See:

```
# journalctl -h
```


Use the `journalctl` command to view the journal logs. By default, the listed entries include a time stamp, the host name, the application that performed the operation, and the actual message.

```
# journalctl
-- Logs begin at ..., end at ...
<date_time> <host name> systemd-journal[65]: ...
...
```

The output of the command is formatted as follows:

- Entries are displayed one page at a time.
- Time stamps are converted to your local time zone.
- Priority of entries is visibly marked. Entries with error priority and higher are red. Entries with notice and warning priority are in bold font.
- The beginning of the boot process is indicated with a special entry.

When running the `journalctl` command without any options or arguments, all log data is displayed, including rotated logs. Oldest entries are listed first. A number of options are available for the `journalctl` command. Examples of some of the options are given on the next page.

Use the `-r` option to display the newest log entries first.

```
# journalctl -r
-- Logs begin at ..., end at ...
<date_time> <host name> CROND[27325]: (root) CMD (/usr...
<date_time> <host name> systemd[1]: Started session ...
...
```

Use the `-n <number>` option to display a specific number of the most recent log entries. The following example displays the three most recent log entries.

```
# journalctl -n 3
-- Logs begin at ..., end at ...
<date_time> <host name> systemd[1]: Started session ...
<date_time> <host name> systemd[1]: Started session ...
<date_time> <host name> CROND[27452]: (root) CMD (/usr...
```

Use the `-p <priority>` option to display only log entries of a specific `<priority>`. Valid priorities are `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, and `emerg`. The following example displays only `crit` log entries. Entries with `err` priority and higher are in red.

```
# journalctl -p crit
-- Logs begin at ..., end at ...
<date_time> host03.example.com smartd[512]: Problem creating ...
<date_time> host03.example.com smartd[512]: In the system's ...
<date_time> host03.example.com firewalld[506]: 2014-11-10 ...
...
```

Use the `-u <systemd_unit>` option to display only log entries for the specified `systemd` unit. The following example displays only log entries associated with the `crond` unit.

```
# journalctl -u crond
-- Logs begin at ..., end at ...
<date_time> <host name> systemd[1]: Starting Command ...
<date_time> <host name> systemd[1]: Started Command ...
<date_time> <host name> crond[578]: (CRON) INFO (RAN...
...
```

Use the `-o <output_form>` option to format the output. Valid output formats are `short`, `short-iso`, `short-precise`, `short-monotonic`, `verbose`, `export`, `json`, `json-pretty`, `json-see`, and `cat`. Refer to the `journalctl(1)` man page for a description of the output formats. The following example displays log entries using the `verbose` format.

```
# journalctl -o verbose
-- Logs begin at ..., end at ...
<date_time>
  PRIORITY=6
  _TRANSPORT=driver
  MESSAGE=Runtime journal is using 8.0M (max 100.1M, leaving ...
...
```

`journald` Metadata

- `journald` adds structured metadata to the messages that assist in troubleshooting.
- To view all metadata for all journal entries:

```
# journalctl -o verbose
```

- To view a list of the metadata fields:

```
# journalctl <TAB> <TAB>
```

- To view a list of unique values that occur in a specific metadata field, use the following syntax:

```
# journalctl -F <fieldname>
```

- To filter log entries for a specific metadata value, use the following syntax:

```
# journalctl <fieldname>=<value>
```


`journald` adds structured metadata to the messages that assist in troubleshooting. The following command (also shown on the previous page) displays all metadata for all journal entries:

```
# journalctl -o verbose
```

You can view a list of the metadata fields by pressing the `<TAB>` key twice after the `journalctl` command:

```
# journalctl <TAB> <TAB>
```

For a description of the metadata fields, see the `systemd.journal-fields(7)` man page. Metadata values are usually text-based but can include binary data. Metadata fields can also have multiple values but this is usually not the case. Metadata can be used for message filtering to assist in troubleshooting.

To view a list of unique values that occur in a specific metadata field, use the following syntax:

```
# journalctl -F <fieldname>
```

You can specify a *`<value>`* for a *`<fieldname>`* to show only log entries that match the condition. Use the following syntax:

```
# journalctl <fieldname>=<value>
```

You can specify multiple values for one field and you can specify multiple field-value pairs.

Quiz

Which of the following entries in `/etc/rsyslog.conf` cause `warning`, `err`, `crit`, `alert`, and `emerg` messages from the kernel to be logged?

a. `kern.*`

b. `kern.warning`

c. `kern.err`

d. `*.kern`

Quiz

Which of the following statements are true?

a. To use the `journald` service, you must stop `rsyslogd`.
b. Journals are nonpersistent by default.
c. The `journald` service adds structured metadata to the messages that assist in troubleshooting.
d. Use the `journald` command to view the journal log files.

Summary

In this lesson, you should have learned how to:

- Describe Oracle Linux 7 system logging options
- Describe the contents of the `rsyslog` configuration file
- Describe `rsyslog` filter options
- Describe facility/priority-based filters
- Describe `rsyslog` actions
- Describe `rsyslog` templates
- Configure log rotation
- Describe `logwatch`
- Describe `journald`
- Use the `journalctl` utility

Practice 21: Overview

The practices for this lesson cover the following:

- Configuring system logging
- Using `rsyslog` templates
- Using `logwatch`
- Using `journald`

22

Troubleshooting

Objectives

After completing this lesson, you should be able to:

- Describe the two-phased approach to troubleshooting
- Describe the type of information needed to troubleshoot a problem
- Describe the available operating system logs to assist in troubleshooting
- Use the `dmesg` utility
- Describe the available troubleshooting resources
- Describe causes of common problems
- Describe troubleshooting boot problems
- Describe typical causes of NFS problems

Two-Phased Approach to Troubleshooting

- Fault Analysis Phase
  - State the problem.
  - Gather information.
  - Identify what is and what is not working.
- Fault Diagnosis Phase
  - Based on the fault analysis findings and past experiences, determine the most probable causes of the fault.
  - Test and verify the probable causes.
  - Take corrective action.
  - Ensure you do not introduce any new problems.
- Document the results of the fault analysis and fault diagnosis phases.

Use a two-phased approach to troubleshooting. Begin with the fault analysis phase in which you state the problem and gather as much information as you can about the problem. Problem information can be gathered from error messages, log files, historical information such as previous problems and associated resolutions, and Oracle bug and support websites. Recent system changes are also an important source of information about a system fault.

In the second phase, you determine the most likely causes of the problem from the information you collected. When possible, take into consideration past experiences with diagnosing similar issues. You then test and verify your list of most likely causes. Through a process of elimination, you identify the actual cause of the fault while simultaneously verifying that you can correct the problem and not introduce any new problems.

Always document the steps you took to isolate and correct the problem for future reference.

Gathering Information

- Get a complete description of the server.
- Describe exactly what the problem is.
  - Symptoms
  - Error messages
- Who is experiencing the problem?
  - One user or several users
- Can the problem be reproduced?
  - Steps to reproduce the problem
  - Is it an intermittent problem?
- Does the problem occur only at certain times of the day or certain days of the week?
- Have any changes been made to the server?

When a problem occurs, the first things to think about are how to locate the problem. The more information you have, the better. Ask probing questions to help clarify the problem. Some users might have difficulty in answering the questions, but any extra information they can provide might help you find the problem.

Knowing when the problem occurs might help you determine the cause. The problem might occur only at a certain time of day. Or perhaps the problem occurred after a change was made to the server or peripherals, or by some new way in which clients are using the server.

Knowing who is experiencing the problem can help determine if the problem exists in a particular part of the network, or if the problem is application dependent. Determine if one person, one group of users, or a larger group is experiencing the problem.

If the problem can be reproduced, determine what steps are needed to reproduce the problem. Also determine if the problem can be reproduced on another system and by another user. A good procedure to remedy hard-to-reproduce problems is to perform general maintenance on the system, such as bringing the system up to date on patches.

Operating System Logs

- Files under `/var/log`:
  - `boot.log` – Messages from bootup
  - `messages` – Standard system error messages
  - `anaconda` – O/S install logs
  - `dmesg` – Log of boot messages showing hardware errors
  - Other logs exist for mail, `cron`, security, and so on.
  - Other directories in `/var/log/` exist for cups, httpd, samba, and so on.
- You can monitor a log file in real time by using the following command:

```
# tail -f <logfile>
```

- To view the journal in live view:

```
# journalctl -f
```


Linux maintains several system logs that help you administer your systems by informing you of important events. Checking system messages is a logical early step when trying to determine the probable cause of a system fault. At a glance, a system message can provide you with the following information :

- Process name/PID number
- Message ID number
- Facility that generated the message (for example, the kernel or a system daemon)
- Level of severity of the message (for example: emergency, error, warning, notice, or information)
- Message

Probably the most important log is the file `/var/log/messages` that records a variety of events, including system error messages, system startups, and system shutdowns. Like most other Linux files, the file contains ASCII text, so you can view it with a text editor or the text processing commands.

You can monitor a log file in real time by using the `tail -f <logfile>` command. Use the `journalctl -f` command to monitor the journal in live view. These commands keep the file open and new messages are appended to the file. Use the `CTRL-C` command to close the file.

`dmesg` Utility

- `dmesg`: Print out a buffer showing latest hardware issues.
- The command prints only a memory structure (kernel ring buffer) in the memory.
- `dmesg` does not have time stamps.
- The buffer can truncate when it is full.
  - `/var/log/boot.log`
  - `/var/log/dmesg*`

The `dmesg` command is used to examine or control the kernel ring buffer. Messages related to the operation of the kernel are written to the ring buffer. A ring buffer is of a constant size and the oldest messages are removed when new messages are written.

Hardware-related information is available in `dmesg` output. This includes memory related issues, CPU information, and information about devices.

See the `dmesg(1)` man page for more information.

Troubleshooting Resources

- Man pages provide the usage of a command and the available options and configuration parameters.
- Many commands and services have a `-d`/`-D` option for debugging or a `-v`/`-V` option for verbose.
- The `/usr/share/doc/` directory contains information about packages installed on your system plus release notes and manuals.
- Oracle Linux 7 administration guides:
  - http://docs.oracle.com/cd/E52668_01/
- The My Oracle Support website contains knowledge articles and other helpful information.
  - https://support.oracle.com/

ORACLE

One important factor when troubleshooting a problem is knowing what configuration files are used, the type of information stored in configuration files, and what services need to be running.

Many resources are available to assist in troubleshooting. The Linux man pages provide configuration file parameters and the available options to the commands and services. Often there is a `-d` or `-D` option to a command, which allows you to turn on various levels of debugging to assist in troubleshooting a problem. These `-d` or `-D` options are normally turned off or not specified by default because they display more information than you need when everything is running smoothly. However, when a problem occurs, the detailed information can be useful in troubleshooting where the problem might be.

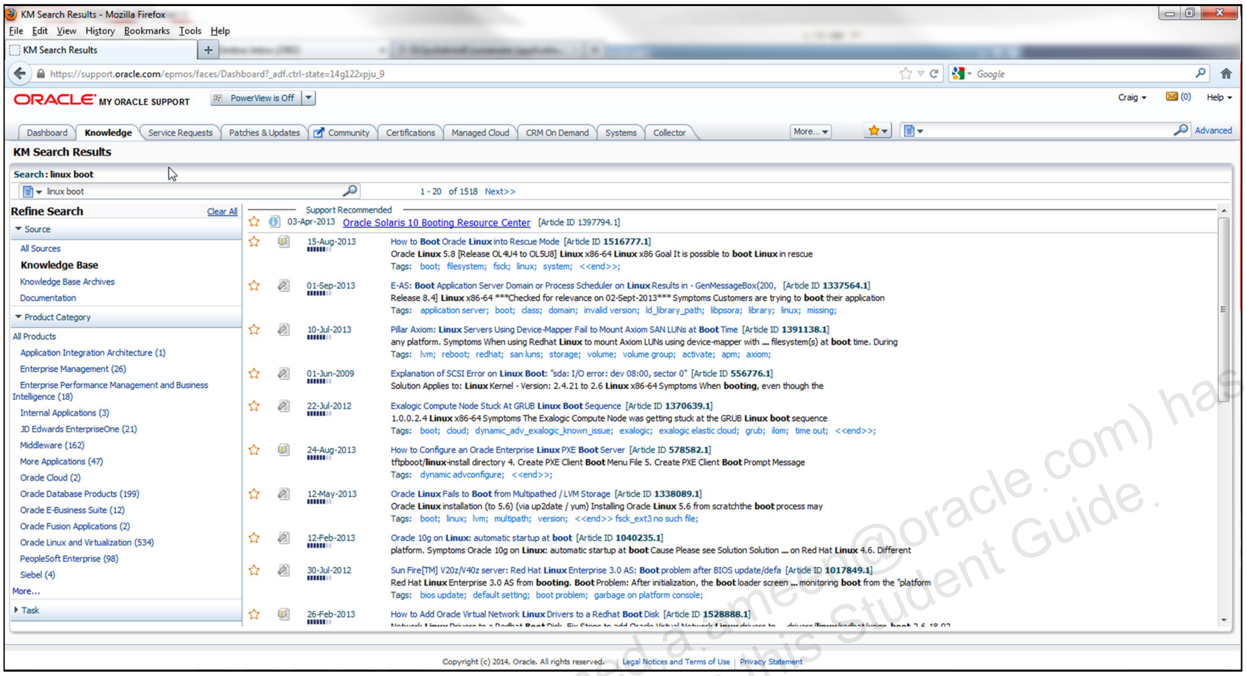
The `/usr/share/doc` directory is another helpful resource. It is the central documentation directory and contains various documentation and release notes for your system.

The administration guides at the URL listed in the slide is another source of information.

The My Oracle Support (MOS) website contains valuable information to assist in troubleshooting system problems.

Internet searches can be helpful in troubleshooting. Entering the exact error message from a log file can often point you to a resolution to your problem.

My Oracle Support



The My Oracle Support website (https://support.oracle.com/) contains knowledge articles and other helpful information to assist in troubleshooting a problem.

The slide shows the results of a search for "linux boot." A list of knowledge articles related to the query is displayed. Click each article to view the details.

My Oracle Support requires a user login and password.

Causes of Common Problems

- Service(s) not running:
  - Use the `systemctl` command to start a service or check the status of a service.
  - Use the `systemctl enable` command to start a service at boot time.
- Configuration errors:
- Firewall (`firewalld` and `iptables`) is prohibiting a connection.
  - Stop the service and test to determine if a firewall is blocking.
- PAM is prohibiting authentication:
  - View `/var/log/secure` for authentication error messages.
- SELinux is denying a connection:
  - Set SELinux to permissive mode and test.

Use the `systemctl` command to ensure a particular daemon (or service) is running. For example, to obtain the status of the `sshd` daemon:

```
# systemctl status sshd
```

Always start (or restart) a service whenever making a change to an associated configuration file. For example, after changing parameters in a network interface file in the `/etc/sysconfig/network-scripts/` directory, restart the `network` service:

```
# systemctl restart network
```

Use the `systemctl enable` command to configure a service to start at boot time. For example, to configure the `vsftpd` daemon to start at boot time:

```
# systemctl enable  vsftpd
```

This command does not actually start a service. You need to run the following command to start the service:

```
# systemctl start vsftpd
```

Ensure configuration files contain valid information. Each service has at least one associated configuration file. Refer to the man pages or administration guides for configuration file parameters. Configuration files often contain comments that describe configuration file parameters.

Many of the system configuration files are located in the `/etc/sysconfig/` directory. Refer to the `/usr/share/doc/initscripts-<version>/sysconfig.txt` file for information about these files.

Do not make kernel setting changes from the command line. To preserve custom settings for kernel features, add them to the `/etc/sysctl.conf` file. Changes made in the `/etc/sysctl.conf` file take effect immediately when issuing the following command:

```
# sysctl -p
```

The `firewalld` and `iptables` services (firewall) are often the cause of a problem with a client-server process. Use the `systemctl stop firewalld|iptables` command to temporarily stop `firewalld` and `iptables` respectively and re-test to determine if the problem is resolved. If so, you can create a rule to open a specific port and restart the service.

PAM modules might be causing authentication errors. Entries are usually written to the `/var/log/secure` log file when PAM is denying access. PAM is covered in another course.

SELinux stands for "Security-Enhanced Linux" and is covered in another course in the Oracle Linux curriculum map. SELinux is often the cause of a problem. You can use the `sestatus` command to display information about SELinux.

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

From this output, you can see that SELinux is enabled and is in enforcing mode. You can temporarily change SELinux to "permissive" mode and re-test to see if the problem is fixed. Use the `setenforce 0` command to temporarily change SELinux to "permissive" mode.

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

Notice the "Current mode" is now set to "Permissive."

The `sestatus` command also reports the status of SELinux:

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

To permanently change the mode, edit the `/etc/selinux/config` file and change the `SELINUX` directive to "permissive" or "disabled."

Troubleshooting Boot Problems

- Configuration errors in the following files can prevent your system from booting:
  - `/boot/grub2/grub.cfg`
  - `/etc/fstab`
- Boot into rescue mode to correct boot problems.
  - Rescue mode boots from installation media.
  - File systems are mounted under `/mnt/sysimage`.
  - Use `chroot` to change the root partition of the rescue mode environment.
  - Then use `vi`, `fsck`, `rpm`, and other utilities to fix the boot problem.
- Use the `grub2-install` to re-install the boot loader.

Knowing the normal sequence of events that occurs in the boot process, and knowing at which point in the process a system had problems, are key to diagnosing and fixing boot-time problems. Configuration errors in important files, such as `/boot/grub2/grub.cfg` and `/etc/fstab` can prevent your system from booting.

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

Rescue mode attempts to mount your file systems under `/mnt/sysimage`. The `/mnt/sysimage` is a temporary root partition, not the root partition of the file system used during normal operations. You can use the `chroot` command to change the root partition of the rescue mode environment to the root partition of your file system. You can then correct any errors in configuration files, run `fsck` to check and repair a file system, use `rpm` to install or upgrade software packages, and other commands to rescue your environment.

You can re-install the GRUB boot loader if it has been corrupted or overwritten by another operating system. Use the `grub2-install` command to re-install the boot loader.

Typical Causes of NFS Problems

- The `rpcbind` or NFS daemons are not running:
  - NFS daemons are `nfs` and `nfslock`.
- Syntax errors:
  - On client `mount` command
  - In `/etc/exports` file on server
- Permission problems:
  - Check UIDs and GIDs.
- Firewall is blocking NFS packets:
  - Check `firewalld` and `iptables` rules or stop the service.
- DNS host name resolution:
  - Ensure `/etc/resolv.conf` contains correct entries.

Typical causes of NFS problems are given in the slide. Check and ensure that the `rpcbind`, `nfs`, and `nfslock` daemons are running on the server. Always start the `rpcbind` service first because the NFS services need `rpcbind` to be running.

Another common problem is syntax errors either in the `mount` command on the client or in the `/etc/exports` file on the server. The format for entries in the `/etc/exports` file is:

```
export-point client1(options) [client2(options) ... ]
```

A common error is inserting a space in the `client(options)` argument. A space after the client identifier and the bracket causes the options to be ignored.

If the NFS file system mounts but you cannot access it, check the permissions and the GIDs and UIDs. NFS requests contain numeric UIDs and GIDs. Just because the username is the same on both the client and the server, it does not mean the UIDs and GIDs are the same.

Firewalls can filter packets necessary for NFS. Check your `firewalld` and `iptables` rules and service. The NFS service uses port `2049`. The `rpcbind` service uses port `111`.

Host name resolution provided by DNS must also be configured properly for NFS to work. Check the `/etc/resolv.conf` file and ensure you are querying the correct DNS server.

Quiz

Which of the following commands is useful in determining if your system has hardware-related errors?

a. `service`

b. `ps`

c. `lsmod`

d. `dmesg`

Summary

In this lesson, you should have learned:

- The two-phased approach to troubleshooting
- The type of information needed to troubleshoot a problem
- The available operating system logs to assist in troubleshooting
- Use of the `dmesg` utility
- The available troubleshooting resources
- Causes of common problems
- Troubleshooting boot problems
- Typical causes of NFS problems

ORACLE

Practice 22: Overview

The practices for this lesson involve troubleshooting some common problems including:

- System boots into single-user mode.
- Status commands fail.
- A `cron` job fails to run.
- A user cannot log in.
- File system does not mount.
- Logical volume space is exhausted.
- There are network connectivity problems.
- There are NFS permission problems.
- You cannot log in to remote hosts using `ssh`.
- Log file is not getting updated.

In these practices, you configure a scenario and verify that everything works. You are directed to run a program that introduces an error. You are given some hints, or things to look at and check for, with regard to diagnosing the problem. Refer to preceding lessons when necessary, and attempt to diagnose and fix the problem.