Oracle Linux 8

Managing Shared File Systems

F29522-12
January 2024

Oracle Linux 8 Managing Shared File Systems,

F29522-12

Contents

4 Managing Samba

Preface

Oracle Linux 8: Managing Shared File Systems provides information about managing shared file systems in Oracle Linux 8.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About Shared File System Management

This chapter includes a brief description of shared files systems and includes information about the types of shared file systems that are provided in Oracle Linux 8 and how they're used. The chapters of this guide describe how to manage each of the shared file systems types that are described in this chapter.

For information about managing local file systems in Oracle Linux, see Oracle Linux 8: Managing Local File Systems.

A *shared file system* is a type of file system that enables sevarl users to access the same files across different operating systems or over a network at the same time. The shared file system approach provides many benefits. Most notably, using shared file systems can improve performance and scalability and can reduce the time that administrators spend managing data.

Oracle Linux8 includes support for several file systems types. The following distributed and shared file systems are discussed in length in their respective sections:

- Network File System (NFS) : Managing the Network File System
- Oracle Cluster File System Version 2 (OCFS2): Managing the Oracle Cluster File System Version 2 in Oracle Linux
- Samba: Managing Samba

2
Managing the Network File System

This chapter includes information about managing the Network File System (NFS) in Oracle Linux 8, including tasks for configuring, administering, and using NFS.

For information about local file system management in Oracle Linux, see Oracle Linux 8: Managing Local File Systems.

About NFS

NFS (Network File System) is a distributed file system that enables a client system to access files over a network as though the files were on local storage.

An NFS server can share directory hierarchies in its local file systems with remote client systems over an IP-based network. After an NFS server exports a directory, NFS clients mount this directory, provided that the clients have been granted the appropriate permissions. To the client systems, the directory appears as if it were a local directory. The benefits of using NFS include centralized storage provisioning, improved data consistency, and reliability.

Supported Versions of NFS

The following versions of NFS are supported in Oracle Linux 8:

- NFS version 3 (NFSv3), specified in RFC 1813.
- NFS version 4 (NFSv4), specified in RFC 7530.
- NFS version 4 minor version 1 (NFSv4.1), specified in RFC 5661.
- NFS version 4 minor version 2 (NFSv4.2), specified in RFC 7862 .

> **Note:**
>
> NFSv2 is no longer supported.

NFSv3 provides safe, asynchronous writes, and efficient error handling. NFSv3 also supports 64-bit file sizes and offsets, which enable clients to access more than 2 GB of file data.

NFSv3 relies on Remote Procedure Call (RPC) services, which are controlled by the `rpcbind` service. The `rpcbind` service responds to requests for an RPC service and then sets up connections for the requested service. In addition, separate services are used to handle locking and mounting protocols, as configuring a firewall to cope with the various ports that are used by all these services can be complex and error-prone.

> **Note:**
>
> In previous Oracle Linux releases, NFSv3 also used the User Datagram Protocol (UDP). However, in Oracle Linux 8, NFS over UDP is no longer supported. Further, UDP is disabled in the NFS server by default in this release.

NFSv4 can work through firewalls and the Internet. Also, NFSv4 doesn't require the `rpcbind` service. In addition, NFSv4 supports access Control Lists (ACLs), and uses stateful operations.

NFSv4 requires the Transmission Control Protocol (TCP) running over an IP network. As mentioned, NFSv4 doesn't use `rpcbind`; as such, the NFS server listens on TCP port 2049 for service requests. The mounting and locking protocols are also integrated into the NFSv4 protocol, which means that separate services are also not required for these protocols. These refinements make firewall configuration for NFSv4 no more difficult than for a service such as HTTP.

Note that in Oracle Linux 8, NFS clients mount by using NFSv4.2 (the default version), but fall back to NFSv4.1 when the server doesn't support NFSv4.2. The mount later falls back to NFSv4.0 and then to NFSv3.

About NFS Services

In Oracle Linux 8, NFS versions rely on Remote Procedure Calls (RPC) between clients and servers. To share or mount NFS file systems, the following required services work together, depending on which version of NFS is implemented. Note that all of these services are started automatically:

nfsd
Server kernel module that services requests for shared NFS file systems.

rpcbind
Service that accepts port reservations from local RPC services, which are made available or advertised so that the corresponding remote RPC services can access them and also hat the client is allowed to access it.

rpc.mountd
Process that is used by an NFS server to process mount requests from NFSv3 clients. The service checks that the requested NFS share is currently exported by the NFS server.

rpc.nfsd
Process that enables explicit NFS versions and protocols the server advertises to be defined.

lockd
Kernel thread that runs on both clients and servers. The `lockd` process implements the Network Lock Manager (NLM) protocol, which enables NFSv3 clients to lock files on the server. The daemon is started automatically whenever the NFS server is run and whenever an NFS file system is mounted.

rpc-statd
Process that implements the Network Status Monitor (NSM) RPC protocol, which notifies NFS clients when an NFS server is restarted without being gracefully brought down. The `rpc-statd` service is automatically started by the `nfs-server` service. This service does not require configuration by the user and is not used with NFSv4.

rpc-idmapd
Process that provides NFSv4 client and server upcalls, which map between on-the-wire NFSv4 names (strings in the form of user@domain) and local UIDs and GIDs. Note that for the `idmapd` process to function with NFSv4, you must configure the `/etc/idmapd.conf` file. Note that only NFSv4 uses the `rpc-idmapd` process.

> **Note:**
>
> The mounting and locking protocols are incorporated into the NFSv4 protocol. Also, the server listens on TCP port 2049. For this reason, NFSv4 does not need to interact with the `rpcbind`, `lockd`, and `rpc-statd` services. However, the `nfs-mountd` service is still required to set up exports on the NFS server; but, the service is not involved in any over-the-wire operations.
>
> The `rpc-idmapd` service only handles upcalls from the kernel and is not itself directly involved in any over-the-wire operations. The service, however, might make naming service calls, which do result in over-the-wire lookups.

Configuring an NFS Server

You can configure an NFS server in Oracle Linux 8 in the following ways:

- By editing the `/etc/exports` file manually.

  Exports can also be added to files that you create in the `/etc/exports.d` directory.
- By using the `exportfs` command.

Configuring an NFS Server by Editing the /etc/exports File

The following steps describe how to configure an NFS server by editing the `/etc/exports` file.

> **Note:**
>
> You can also add exports to files that you create in the `/etc/exports.d` directory in a similar fashion.

1. If it is not yet in the system, install the `nfs-utils` package.

   ```
   sudo dnf install nfs-utils
   ```

2. Edit the `/etc/exports` file to define the directories that the server makes available for clients to mount, for example:

```
/var/folder 192.0.2.102(rw,async)
/usr/local/apps *(all_squash,anonuid=501,anongid=501,ro)
/var/projects/proj1 192.168.1.0/24(ro) mgmtpc(rw)
```

   Each entry includes the local path to the exported directory, followed by a list of clients that can mount the directory and client-specific exports options in parentheses. No spaces should separate a client specifier and the parenthesized list of options that apply to that client.

   The following information explains the file entries in greater detail:

   - Only the client system with the IP address 192.0.2.102 can mount the `/var/folder` directory with read and write permissions. All writes to the disk are asynchronous. Therefore, the server does not wait for write requests to be written to disk before responding to further requests from the client.
   - As indicated by the wildcard (*), all of the clients can mount the `/usr/local/apps` directory as read-only. All connecting users, including `root` users, are mapped to the local, unprivileged user with UID 501 and GID 501.
   - All of the clients on the 192.168.1.0/24 subnet can mount the `/var/projects/proj1` directory as read-only. However, the client system named `mgmtpc` can mount the directory with read-write permissions.

   For more information, see the `exports(5)` manual page.

3. If the server serves NFSv4 clients, edit the `/etc/idmapd.conf` file's definition for the Domain parameter by specifying the server's domain name.

```
Domain = mydom.com
```

   This setting prevents the owner and group from being unexpectedly listed as the anonymous user or group (`nobody` or `nogroup`) on NFS clients when the `all_squash` mount option is not specified.

4. If you need to enable access through the firewall for NFSv4 clients only, use the following commands:

```
sudo firewall-cmd --permanent --zone=zone --add-service=nfs
```

   This configuration assumes that `rpc.nfsd` listens for client requests on the default TCP port 2049.

5. If you need to enable access through the firewall for NFSv3 and NFSv4 clients, do the following:

   a. Edit the `/etc/nfs.conf` file to create port settings for handling network mount requests and status monitoring. Additionally, set the TCP port on which the network lock manager should listen, for example:

```
# Ports that various services should listen on.

[mountd]
```

```
port = 892

[statd]
port = 662

[lockd]
port = 32803
```

If any port is in use, NFS fails to start. Use the `lsof -i` command to locate an unused port and then amend the setting in the `/etc/nfs.conf` file as appropriate.

To confirm on which ports RPC services are listening, use the `rpcinfo -p` command.

b. Restart the firewall service and configure the firewall to allow NFSv3 connections:

```
sudo firewall-cmd --permanent --zone=zone --add-port=2049/tcp --add-
port=111/tcp --add-port=32803/tcp --add-port=892/tcp --add-
port=662/tcp
```

c. Reboot the server.

```
sudo systemctl reboot
```

6. Start the `nfs-server` service and configure the service to start following a system reboot:

```
sudo systemctl enable --now nfs-server
```

7. Display a list of the exported file systems.

```
sudo showmount -e


Export list for host01.mydom.com
/var/folder 192.0.2.102
/usr/local/apps *
/var/projects/proj1 192.168.1.0/24 mgmtpc
```

The `exportfs` command on the server displays the same information as the `showmount -e` command.

```
sudo /usr/sbin/exportfs -v
```

The `showmount -a` command displays all of the current clients and all of the file systems that the clients have mounted.

> **Note:**
>
> To enable use of the `showmount` command from NFSv4 clients, specify a port number to the `MOUNTD_PORT` parameter in `/etc/nfs.conf`. Then, create a firewall rule to enable access to this TCP port.

Configuring an NFS Server by Using the exportfs Command

The `exportfs` command enables the administrator to export or unexport directories selectively, and eliminates the need to restart the NFS service. By providing the appropriate options, the `exportfs` command writes the exported file systems to the `/var/lib/nfs/etab` file. Changes to the list of exported file systems are effective immediately because the `nfs-mountd` service refers to the `etab` file for determining access privileges to a file system.

If used without any options, `exportfs` displays a list of currently exported file systems.

Options that you can specify with the `exportfs` command include the following:

-r
Refreshes the list of exported directories in the `/var/lib/nfs/etab` file by incorporating any changes that were made to the list in the `/etc/exports` file.

-a
Exports all of the file systems that are specified in the `/etc/exports` file. This option can be combined with other options, which determines the action the command performs.

-u
Unexports all of the shared directories.

> **Note:**
>
> The `exportfs -ua` command suspends NFS file sharing, but keeps all NFS services running. To re-enable NFS sharing, use the `exportfs -r` command.

-v
Specifies a verbose logging, which displays information about the file systems that are being exported or unexported in greater detail.

For more information, see the `exportfs(8)`, `exports(5)`, and `showmount(8)` manual pages.

Mounting an NFS File System

1. If it's not yet on the system, install the `nfs-utils` package.

   ```
   sudo dnf install nfs-utils
   ```

2. Display the file systems that the NFS server exports.

   ```
   sudo showmount -e host01.mydom.com
   ```

The output of the previous command would be similar to the following:

```
Export list for host01.mydom.com
/var/folder 192.0.2.102
/usr/local/apps *
/var/projects/proj1 192.168.1.0/24 mgmtpc
```

> **Note:**
>
> Some servers don't accept querying of this information but can still export NFS file systems.

3. Mount an exported NFS file system on an available mount point.

   ```
   sudo mount -t nfs -r -o nosuid host01.mydoc.com:/usr/local/apps /apps
   ```

   Typically, when mounting an NFS file system, the `-t nfs` option can be omitted.

   This example mounts the `/usr/local/apps` directory that's exported by `host01.mydoc.com` with read-only permissions on `/apps`. The `nosuid` option prevents remote users from gaining higher privileges by running a `setuid` program.

4. To configure the system to mount an NFS file system at boot time, add an entry for the file system to the `/etc/fstab` file, as shown in the following example:

   ```
   host01.mydoc.com:/usr/local/apps      /apps      nfs      ro,nosuid  0 0
   ```

For more information, see the `mount(8)`, `nfs(5)`, and `showmount(8)` manual pages.

3

Managing the Oracle Cluster File System Version 2 in Oracle Linux

This chapter includes information about managing the Oracle Cluster File System Version 2 (OCFS2) in Oracle Linux 8. It includes tasks for configuring, administering, and troubleshooting OCFS2.

In Oracle Linux 8, Oracle Cluster File Ssystem Version 2 (OCFS2) is supported on Unbreakable Enterprise Kernel (UEK) releases *only*, starting with Unbreakable Enterprise Kernel Release 6 (UEK R6).

For information about local file system management in Oracle Linux, see Oracle Linux 8: Managing Local File Systems.

About OCFS2

OCFS2 (Oracle Cluster File System Version 2) is a general-purpose shared-disk file system that is intended for use with clusters. OCFS2 offers high performance as well as high availability. Optionally, you can also mount an OCFS2 volume on a standalone, non-clustered system.

Using OCFS2 offers the following benefits:

- You can use the `reflink` command with OCFS2 to create copy-on-write clones of individual files. You can also use the `cp --reflink` command in a similar way that you would on a Btrfs file system. Typically, such clones enable you to save disk space when storing multiple copies of very similar files, such as virtual machine (VM) images or Linux Containers.
  Note that when using the `reflink` command, the resulting file system behaves like a clone of the original files ystem, which means that their UUIDs are identical. When using the `reflink` command to create a clone, you must change the UUID by using the `tunefs.ocfs2` command. See Querying and Changing Volume Parameters.
- Mounting a local OCFS2 file system enables you to subsequently migrate the file system to a cluster file system without requiring any conversion.
- OCFS2 provides local file-system semantics. Therefore, almost all applications can use OCFS2. Applications that are cluster-aware can use cache-coherent parallel I/O from multiple cluster nodes to balance activity across the cluster, or they can use of the available file-system functionality to fail over and run on another node in the event that a node fails.

OCFS2 Use Cases

The following are some typical use cases for OCFS2.

Load Balancing Use Case

You can use OCFS2 nodes to share resources between client systems. For example, the nodes could export a shared file system by using Samba or NFS. To distribute service requests between the nodes, you can use round-robin DNS, a network load balancer; or, you can specify which node should be used on each client.

Oracle Real Application Cluster Use Case

Oracle Real Application Cluster (RAC) uses its own cluster stack, Cluster Synchronization Services (CSS). You can use O2CB in conjunction with CSS, but note that each stack is configured independently for timeouts, nodes, and other cluster settings. You can use OCFS2 to host the voting disk files and the Oracle cluster registry (OCR), but not the grid infrastructure user's home, which must exist on a local file system on each node.

Because both CSS and O2CB use the lowest node number as a tie breaker in quorum calculations, ensure that the node numbers are the same in both clusters. If necessary, edit the O2CB configuration file, `/etc/ocfs2/cluster.conf`, to make the node numbering consistent. Then, update this file on all of the nodes. The change takes effect when the cluster is restarted.

Oracle Database Use Case

Specify the `noatime` option when mounting volumes that host Oracle datafiles, control files, redo logs, voting disk, and OCR. The `noatime` option disables unnecessary updates to the access time on the inodes.

Specify the `nointr` mount option to prevent signals interrupting I/O transactions that are in progress.

By default, the `init.ora` parameter `filesystemio_options` directs the database to perform direct I/O to the Oracle datafiles, control files, and redo logs. You should also specify the `datavolume` mount option for volumes that contain the voting disk and OCR. Do *not* specify this option for volumes that host the Oracle user's home directory or Oracle E-Business Suite.

To prevent database blocks from becoming fragmented across a disk, ensure that the file system cluster size is at minimum as large as the database block size, which is typically 8KB. If you specify the file system usage type as `datafiles` when using the `mkfs.ocfs2` command, the file system cluster size is set to 128KB.

To enable multiple nodes to maximize throughput by concurrently streaming data to an Oracle datafile, OCFS2 deviates from the POSIX standard by not updating the modification time (`mtime`) on the disk when performing non-extending direct I/O writes. The value of `mtime` is updated in memory. However, OCFS2 does not write the value to disk unless an application extends or truncates the file or performs a operation to change the file metadata, such as using the `touch` command. This behavior leads to results with different nodes reporting different time stamps for the same file. Use the following command to view the on-disk timestamp of a file:

```
sudo debugfs.ocfs2 -R "stat /file_path" device | grep "mtime:"
```

Setting Up an OCFS2 Cluster

A cluster consists of members called nodes. For best performance, each node in the cluster should have at least two network interfaces. The first interface is connected to a public network to allow general access to the systems, while the second interface is used for private communications between the nodes and the *cluster heartbeat*. This second interface determines how the cluster nodes coordinate their access to shared resources and how they monitor each other's state.

> **Important:**
>
> Both network interfaces must be connected through a network switch. Additionally, you must ensure that all of the network interfaces are configured and working before configuring the cluster.

Planning for an OCFS2 Cluster

Aside from setting up the proper equipment, prepare the following for the cluster:

- Designated cluster members, specifically, their hostnames and corresponding IP addresses.
- Heartbeat mode to use in the cluster.

In a cluster configuration, small packets travel throughout the entire setup over a specific UDP port, including all the networks configured for cluster. These packets establish routes between the cluster nodes and become indicators of whether the cluster network is up or down. Essentially, the packets are the way you can determine the health of the cluster. For this reason, these packets are also called *heartbeats*.

You can configure a cluster to run in either of the following heartbeat modes:

- Local heartbeat thread for each shared device (default heartbeat mode).

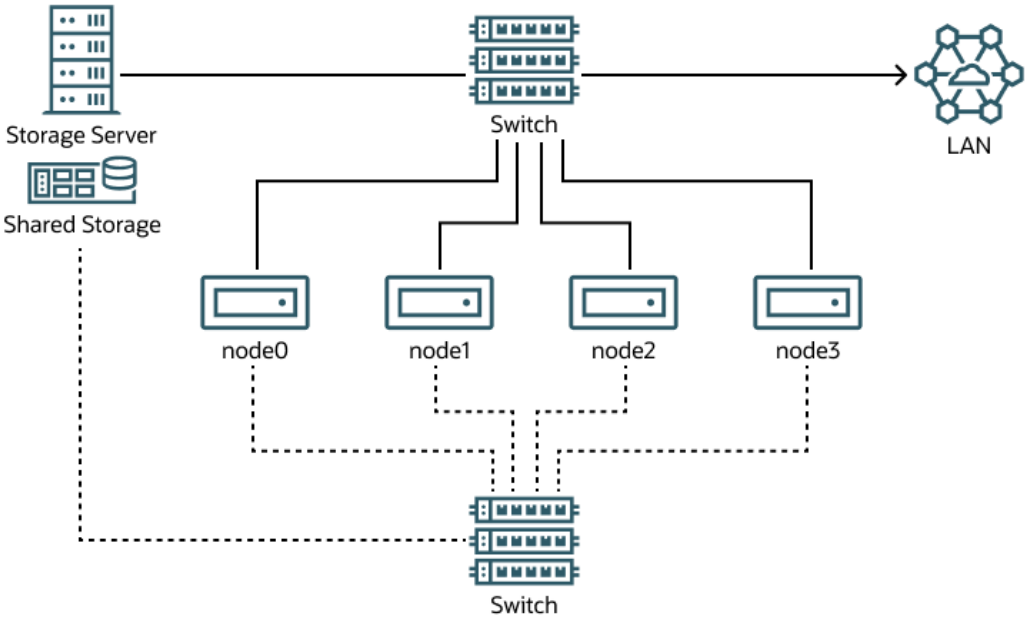
  In this configuration, a node starts a heartbeat thread when the node mounts an OCFS2 volume and stops the thread when the node unmounts the volume. CPU overhead is large on nodes that mount a large number of OCFS2 volumes because each mount requires a separate heartbeat thread. Likewise, a large number of mounts increases the risk of a node becoming isolated out of the cluster, also known as *fencing*, due to a heartbeat I/O timeout on a single mount.

- Global heartbeat on specific shared devices.

  This mode enables you to configure any OCFS2 volume as a global heartbeat device, provided that the volume occupies a whole disk device and not a partition. In this mode, the heartbeat to the device starts when the cluster becomes online and stops when the cluster goes offline. This mode is recommended for clusters that mount a large number of OCFS2 volumes. A node fences itself out of the cluster if a heartbeat I/O timeout occurs on more than half of the global heartbeat devices. To provide redundancy against failure of one of the devices, you should configure at least three global heartbeat devices.

The following figure shows a cluster of four nodes that are connected by using a network switch to a LAN and a network storage server. The nodes and storage server are also connected by using a switch to a private network that is used for the local cluster heartbeat.

Figure 3-1 Cluster Configuration by Using a Private Network



Although you can configure and use OCFS2 without using a private network, such a configuration increases the probability of a node fencing itself out of the cluster due to an I/O heartbeat timeout.

The following table provides recommendations for minimum cluster size settings for different file system size ranges:

File System Size	Suggested Minimum Cluster Size
1 GB - 10 GB	8K
10GB - 100 GB	16K
100 GB - 1 TB	32K
1 TB - 10 TB	64K
10 TB - 16 TB	128K

Installing the Cluster Software

You can configure a cluster whose nodes use mixed versions of OCFS2 and the UEK software. This configuration is supported for scenarios where you are performing a rolling update of a cluster. In these cases, the cluster node that is running the lowest version of the software determines the set of usable features. However, you should preferably use the same version of the OCFS2 software and a compatible UEK release on all the nodes of the cluster to ensure efficiency in cluster operations.

For a tutorial on how to configure OCFS2, see Use Oracle Cloud Cluster File System Tools on Oracle Cloud Infrastructure.

> **Important:**
>
> Perform the following procedure on each designated node.

1. Display designated node's kernel version.

   ```
   sudo uname -r
   ```

   The output on all the nodes should display the same version, for example, 5.4.17-2136.302.7.2.1.el8uek.x86_64. If necessary, update each node to ensure that all of the nodes are running the same kernel version.

2. Install the OCFS2 packages.

   ```
   sudo dnf install -y ocfs2-tools
   ```

   > **Note:**
   >
   > If you want to use the global heartbeat feature, install the `ocfs2-tools-1.8.0-11` or later package as well.

3. Configure the firewall to allow access on the interface that the cluster uses for private cluster communication.

   By default, the cluster uses both TCP and UDP over port 7777.

   ```
   sudo firewall-cmd --permanent --add-port=7777/tcp --add-port=7777/udp
   ```

4. Disable SELinux.
   With a text editor, open the `/etc/selinux/config` file and disable SELinux on the appropriate line entry in the file, as shown in this example in bold.

   ```
   # This file controls the state of SELinux on the system.
   # SELINUX= can take one of these three values:
   #     enforcing - SELinux security policy is enforced.
   #     permissive - SELinux prints warnings instead of enforcing.
   #     disabled - No SELinux policy is loaded.
   SELINUX=disabled
   ```

Configuring the Cluster Layout

1. On any one of the designated cluster member, create a cluster definition.

   ```
   sudo o2cb add-cluster cluster-name
   ```

2. Add the current system and all of the other designated nodes to the cluster.

   ```
   sudo o2cb add-node cluster-name hostname --ip ip_address
   ```

The IP address should be the IP address that is used by the node for private communication in the cluster.

Suppose that you want to create a cluster `mycluster` out of the following systems as identified by their host names. Further, you are using `ol-sys1` to complete this configuration.

- `ol-sys0`: 10.1.0.100
- `ol-sys1`: 10.1.0.110
- `ol-sys2`: 10.1.0.120

To add nodes to `mycluster`, you would type the following commands on `ol-sys1`:

```
sudo o2cb add-node mycluster ol-sys0 --ip 10.1.0.100
sudo ol2cb add-node mycluster ol-sys1 --ip 10.1.0.110
sudo ol2cb add-node mycluster ol-sys2 --ip 10.1.0.120
```

> **Note:**
>
> OCFS2 only supports IPv4 addresses.

3. If you want the cluster to run in global heartbeat mode, do the following:

   a. Run the following command on every cluster device:

   ```
   sudo o2cb add-heartbeat cluster-name device-name
   ```

   For example, for `mycluster`, you set the cluster heartbeat on `/dev/sdd`, `/dev/sdg`, and `/dev/sdj` as follows:

   ```
   sudo o2cb add-heartbeat mycluster /dev/sdd
   sudo o2cb add-heartbeat mycluster /dev/sdg
   sudo o2cb add-heartbeat mycluster /dev/sdj
   ```

   b. Set the cluster's heartbeat mode to global.

   ```
   sudo o2cb heartbeat-mode cluster-name global
   ```

> **Important:**
>
> You must configure the global heartbeat feature to use whole disk devices. Global heartbeat devices on disk partitions are not supported.

4. Copy the cluster `/etc/ocfs2/cluster.conf` file to each node in the cluster.
5. (Optional) Display information about the cluster.

   ```
   sudo o2cb list-cluster cluster-name
   ```

A 3-node cluster `mycluster` with a local heartbeat would display information similar to the following:

```
node:
    name = ol-sys0
    cluster = mycluster
    number = 0
    ip_address = 10.1.0.100
    ip_port = 7777

node:
        name = ol-sys1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.110
        ip_port = 7777

node:
        name = ol-sys2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.120
        ip_port = 7777

cluster:
        name = mycluster
        heartbeat_mode = local
        node_count = 3
```

The same cluster but with a global heartbeat would have the following information:

```
node:
        name = ol-sys0
        cluster = mycluster
        number = 0
        ip_address = 10.1.0.100
        ip_port = 7777

node:
        name = ol-sys1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.110
        ip_port = 7777

node:
        name = ol-sys2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.120
        ip_port = 7777

cluster:
        name = mycluster
```

```
        heartbeat_mode = global
        node_count = 3

heartbeat:
        cluster = mycluster
        region = 7DA5015346C245E6A41AA85E2E7EA3CF

heartbeat:
        cluster = mycluster
        region = 4F9FBB0D9B6341729F21A8891B9A05BD

heartbeat:
        cluster = mycluster
        region = B423C7EEE9FC426790FC411972C91CC3
```

The heartbeat regions are represented by the UUIDs of their block devices.

Configuring and Starting the O2CB Cluster Stack

> **Important:**
>
> Perform the following procedure on each designated node.

1. Configure the node.

   ```
   sudo /sbin/o2cb.init configure
   ```

   The configuration process prompts you for additional information. Some of the parameters you need to set are the following:

   - Load the O2CB driver on boot: Specify `y` or `n`, which is the default setting.
   - Cluster to start at boot: Specify the name of your cluster. The name must match that as found in the `/etc/ocfs2/cluster.conf` file.
   - For the rest of the parameters for which you are prompted for information, the default settings are typically sufficient. However, you can specify values other than the default if necessary.

2. Verify the settings for the cluster stack.

   ```
   sudo /sbin/o2cb.init status
   ```

   A cluster that uses local heartbeat mode would display information similar to the following:

   ```
   Driver for "configfs": Loaded
   Filesystem "configfs": Mounted
   Stack glue driver: Loaded
   Stack plugin "o2cb": Loaded
   Driver for "ocfs2_dlmfs": Loaded
   Filesystem "ocfs2_dlmfs": Mounted
   ```

```
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Local
Checking O2CB heartbeat: Active
```

A cluster that uses global heartbeat mode would display information similar to the following:

```
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Global
Checking O2CB heartbeat: Active
  7DA5015346C245E6A41AA85E2E7EA3CF /dev/sdd
  4F9FBB0D9B6341729F21A8891B9A05BD /dev/sdg
  B423C7EEE9FC426790FC411972C91CC3 /dev/sdj
```

3. Enable the `o2cb` and `ocfs2` services so that they start at boot time after networking is enabled.

```
sudo systemctl enable o2cb


sudo systemctl enable ocfs2
```

4. Using a text editor, set the following kernel settings for cluster operations in the `/etc/sysctl.d/99-sysctl.conf` file:

   - `kernel.panic` = 30
     This setting specifies the number of seconds after a panic before a system automatically resets itself. The default value is zero, which means that the system hangs after a panic. A non-zero value enables the system to automatically reset. If you require a memory image to be created before the system hangs, assign a higher value.
   - `kernel.panic_on_oops` = 1
     This setting enables the system to panic if a kernel `oops` occurs. Thus, if a kernel thread required for cluster operation crashes, the system resets itself. Otherwise, another node might not be able to tell whether a node is slow to respond or unable to respond, which eventually causes all cluster operations to hang.

5. Apply the configuration by running the following command:

```
sudo sysctl -p
```

The `o2cb.init` command accepts other subcommands to enable you to administer the cluster, such as the following:

- `/sbin/o2cb.init status`: Check the status of the cluster stack.
- `/sbin/o2cb.init online`: Stop the cluster stack.
- `/sbin/o2cb.init offline`: Start the cluster stack.
- `/sbin/o2cb.init unload`: Unload the cluster stack.

To determine other available subcommands, type the command `o2cb.init` by itself.

Working With OCFS2 Volumes

To configure OCFS2 volumes, you use `mkfs.ocfs2` as the main command. To enable you to create the volume according to specific requirements, the command accepts different options and arguments, including the following:

`-b blocksize, --block-size blocksize`
Specifies the unit size for I/O transactions to and from the file system, and the size of inode and extent blocks. The supported block sizes are 512 (512 bytes), 1K, 2K, and 4K. The default and recommended block size is 4K (4 kilobytes).

`-C clustersize, --cluster-size clustersize`
Specifies the unit size for space used to allocate file data. The supported cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).

`--fs-feature-level=feature-level`
Enables you select a set of file-system features from the following choices:

- `default`: Enables support for the sparse files, unwritten extents, and inline data features.
- `max-compat`: Enables only those features that are understood by older versions of OCFS2.
- `max-features`:

  Enables all features that OCFS2 currently supports.

`--fs_features=feature`
Enables you to enable or disable individual features such as support for sparse files, unwritten extents, and backup superblocks. For more information, see the `mkfs.ocfs2(8)` manual page.

`-J journalsize, --journal-size journalsize`
Specifies the size of the write-ahead journal. If not specified, the size is determined from the file system usage type that you specify to the `-T` option, and, otherwise, from the volume size. The default size of the journal is 64M (64 MB) for `datafiles`, 256M (256 MB) for `mail`, and 128M (128 MB) for `vmstore`.

`-L label, --label label`
Specifies a descriptive name for the volume that allows you to identify it easily on different cluster nodes.

`-N number-of-slots, --node-slots number-of-slots`
Determines the maximum number of nodes that can concurrently access a volume, which is limited by the number of node slots for system files such as the file-system journal. For best performance, set the number of node slots to at least twice the number of nodes. If you subsequently increase the number of node slots, performance can suffer because the journal will no longer be contiguously laid out on the outer edge of the disk platter.

`-T file-system-usage-type`
Specifies the type of usage for the file system, which is one of the following three options:

- `datafiles`: Database files are typically few in number, fully allocated, and relatively large. Such files require few metadata changes, and do not benefit from having a large journal.
- `mail`: Mail server files are typically many in number, and relatively small. Such files require many metadata changes, and benefit from having a large journal.
- `vmstore`: Virtual machine image files are typically few in number, sparsely allocated, and relatively large. Such files require a moderate number of metadata changes and a medium sized journal.

Creating and Mounting OCFS2 Volumes

When creating OCFS2 volumes, keep the following additional points in mind:

- Do not create an OCFS2 volume on an LVM logical volume, as LVM is not cluster-aware.
- After you have created an OCFS2 volume, you cannot change the block and cluster size of that volume. You can use the `tunefs.ocfs2` command to modify other settings for the file system, with certain restrictions. For more information, see the `tunefs.ocfs2(8)` manual page.
- If you intend that the volume store database files, do not specify a cluster size that is smaller than the block size of the database.
- The default cluster size of 4 KB is not suitable if the file system is larger than a few gigabytes.

1. Create an OCFS2 volume.

```
sudo mkfs.ocfs2 -L "myvol" /dev/sdc1
```

This command creates the volume with a label on the specified device. Without additional options or arguments, specifications, the creates a volume that uses default values for some of its properties, such as 4 KB block and cluster size, eight node slots, 256 MB journal, and support for default file system features. This volume with default settings is suitable for file systems that are no larger than a few gigabytes.

> **Tip:**
>
> Ensure that the device corresponds to a partition so that you can refer to the label when mounting the volume.

The options you use with the `mkfs.ocfs2` command determines the volume you are creating. Consider the following examples:

- Create a labeled volume for use as a database.

  ```
  sudo mkfs.ocfs2 -L "dbvol" -T datafiles /dev/sdd2
  ```

  In this case, the cluster size is set to 128 KB and the journal size to 32 MB.

- Create a volume with specific property settings.

  ```
  sudo mkfs.ocfs2 -C 16K -J size=128M -N 16 --fs-feature-level=max-
  features --fs-features=norefcount /dev/sde1
  ```

  In this case, cluster and journal sizes are specified, as well as the number of node slots. Likewise, all of the supported features are enabled for use except `norefcount` trees.

2. On each cluster member, mount the created volume.

   a. Create a mount point.

   ```
   sudo mkdir /u01
   ```

   b. Mount the volume.

   ```
   sudo mount -L myvol /u01
   ```

   c. Check the status of the heartbeat mode.

   ```
   sudo o2cb.init status
   ```

   The heartbeat becomes active after the volume is mounted.

Optionally, you can automate the mount operation by adding an entry to the `/etc/fstab`, for example:

```
myvol   /u01    ocfs2     _netdev,defaults  0 0
```

In the entry, `_netdev` informs the system to mount an OCFS2 volume at boot time only after networking is started. Likewise, the system should unmount the file system before networking is stopped.

Querying and Changing Volume Parameters

Use the `tunefs.ocfs2` command to query or change volume parameters.

For example, to find out the label, UUID, and number of node slots for a volume, you would use the following command:

```
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb


Label = myvol
UUID = CBB8D5E0C169497C8B52A0FD555C7A3E
NumSlots = 4
```

You would generate a new UUID for a volume by using the following commands:

```
sudo tunefs.ocfs2 -U /dev/sda
```

```
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb
```

```
Label = myvol
UUID = 48E56A2BBAB34A9EB1BE832B3C36AB5C
NumSlots = 4
```

Creating a Local OCFS2 File System

The following procedure describes how to create an OCFS2 file system to be mounted locally, which is not associated with a cluster.

To create an OCFS2 file system that is to be mounted locally, use the following command syntax:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 [options] device
```

Consider the following example:, you would create a locally mountable OCFS2 volume on `/dev/sdc1`, with one node slot and the label `localvol`, as follows:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 -L "localvol" /dev/sdc1
```

The command creates a locally mountable volume with the label `localvol` and a single node slot.

To convert a local OCTFS2 file system to cluster use, use the `tunefs.ocfs2` utility as follows:

```
sudo umount /dev/sdc1
```

```
sudo tunefs.ocfs2 -M cluster --fs-features=clusterinfo -N 8 /dev/sdc1
```

The previous example also increases the number of node slots from 1 to 8, to allow up to eight nodes to mount the file system.

Troubleshooting OCFS2 Issues

Refer to the following information when investigating how to resolve issues that you might encounter when administering OCFS2.

Recommended Debugging Tools and Practices

Use the following tools to troubleshoot OCFS2 issues:

- It is recommended that you set up `netconsole` on the nodes to capture an oops trace.

- Use the `tcpdump` command to capture the DLM's network traffic between nodes. For example, to capture TCP traffic on port 7777 for the private network interface `em2`, you could use the following command:

  ```
  sudo tcpdump -i em2 -C 10 -W 15 -s 10000 -Sw /tmp/`hostname -
  s`_tcpdump.log \
  -ttt 'port 7777' &
  ```

- Use the `debugfs.ocfs2` command to trace events in the OCFS2 driver, determine lock statuses, walk directory structures, examine inodes, and so on. This command is similar in behavior to the `debugfs` command that is used for the `ext3` file system.

  For more information, see the `debugfs.ocfs2(8)` manual page.

- Use the `o2image` command to save an OCFS2 file system's metadata, including information about inodes, file names, and directory names, to an image file on another file system. Because the image file contains only metadata, it is much smaller than the original file system. You can use the `debugfs.ocfs2` command to open the image file and analyze the file system layout to determine the cause of a file system corruption or performance problem.

  For example, to create the image `/tmp/sda2.img` from the OCFS2 file system on the device `/dev/sda2`, you would use the following command:

  ```
  sudo o2image /dev/sda2 /tmp/sda2.img
  ```

  For more information, see the `o2image(8)` manual page.

Mounting the debugfs File System

OCFS2 uses the `debugfs` file system to enable userspace access to information about its in-kernel state. You must mount the `debugfs` file system to use the `debugfs.ocfs2` command.

For example, to mount the `debugfs` file system, add the following line to the `/etc/fstab` file:

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

Then, run the `mount -a` command.

Configuring OCFS2 Tracing

You can use the following commands and methods to trace issues in OCFS2.

Commands for Tracing OCFS2 Issues

The following list describes several commands that are useful for tracing issues.

`debugfs.ocfs2 -l`
Lists all of the trace bits and their statuses.

debugfs.ocfs2 -l SUPER allow|off|deny
Enables, disables, or disallows tracing for the superblock, respectively. If you specify `deny`, then even if another tracing mode setting implicitly allows it, tracing is still disallowed.

debugfs.ocfs2 -l HEARTBEAT ENTRY EXIT allow
Enable heartbeat tracing.

debugfs.ocfs2 -l HEARTBEAT off ENTRY EXIT deny
Disable heartbeat tracing. `ENTRY` and `EXIT` parameters are set to `deny`, as these parameters exist in all trace paths.

debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow
Enable tracing for the file system.

debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE allow
Disable tracing for the file system.

debugfs.ocfs2 -l ENTRY EXIT DLM DLM_THREAD allow
Enable tracing for the DLM.

debugfs.ocfs2 -l ENTRY EXIT deny DLM DLM_THREAD allow
Disable tracing for the DLM.

OCFS2 Tracing Methods and Examples

One method that you can use to obtain a trace is to first enable the trace, sleep for a short while, and then disable the trace. As shown in the following example, to avoid unnecessary output, you should reset the trace bits to their default settings after you have finished tracing:

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow && sleep 10 &&
sudo debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

To limit the amount of information that is displayed, enable only the trace bits that are relevant to diagnosing the problem.

If a specific file system command, such as `mv`, is causing an error, you might use commands that are shown in the following example to trace the error:

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow


mv source destination & CMD_PID=$(jobs -p %-)


echo $CMD_PID


sudo debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

Because the trace is enabled for all mounted OCFS2 volumes, knowing the correct process ID can help you to interpret the trace.

For more information, see the `debugfs.ocfs2(8)` manual page.

Debugging File System Locks

If an OCFS2 volume hangs, you can use the following procedure to determine which locks are busy and which processes are likely to be holding the locks.

In the following procedure, the `Lockres` value refers to the lock name that is used by DLM, which is a combination of a lock-type identifier, inode number, and a generation number. The following table lists the various lock types and their associated identifier.

Table 3-1 DLM Lock Types

Identifier	Lock Type
D	File data
M	Metadata
R	Rename
S	Superblock
W	Read-write

1. Mount the debug file system.

   ```
   sudo mount -t debugfs debugfs /sys/kernel/debug
   ```

2. Dump the lock statuses for the file system device, which is `/dev/sdx1` in the following example:

   ```
   echo "fs_locks" | sudo debugfs.ocfs2 /dev/sdx1 | sudo tee /tmp/
   fslocks


   Lockres: M000000000000006672078b84822 Mode: Protected Read
   ...
   ```

3. Use the `Lockres` value from the previous output to obtain the inode number and generation number for the lock.

   ```
   sudo echo "stat lockres-value" | sudo debugfs.ocfs2 -n /dev/sdx1
   ```

   For example, for the `Locres` value `M000000000000006672078b84822` from the previous step, the command output might resemble the following:

   ```
   Inode: 419616   Mode: 0666   Generation: 2025343010 (0x78b84822)
   ...
   ```

4. Determine the file system object to which the inode number in the previous output relates.

   ```
   sudo echo "locate inode" | sudo debugfs.ocfs2 -n /dev/sdx1
   ```

For example, for the `Inode` value `419616` from the previous step, the command output might resemble the following:

```
419616 /linux-2.6.15/arch/i386/kernel/semaphore.c
```

5. Obtain the lock names that are associated with the file system object, which in the previous step's output is `/linux-2.6.15/arch/i386/kernel/semaphore.c`. Thus, you would type:

```
sudo echo "encode /linux-2.6.15/arch/i386/kernel/semaphore.c" | sudo
debugfs.ocfs2 -n /dev/sdx1


M000000000000006672078b84822 D000000000000006672078b84822
W000000000000006672078b84822
```

In the previous example, a metadata lock, a file data lock, and a read-write lock are associated with the file system object.

6. Determine the DLM domain of the file system by running the following command:

```
sudo echo "stats" | sudo debugfs.ocfs2 -n /dev/sdX1 | grep UUID: | while
read a b ; do echo $b ; done


82DA8137A49A47E4B187F74E09FBBB4B
```

7. Using the values of the DLM domain and the lock name that enables debugging for the DLM, run the following command:

```
sudo echo R 82DA8137A49A47E4B187F74E09FBBB4B M000000000000006672078b84822
| sudo tee /proc/fs/ocfs2_dlm/debug
```

8. Examine the debug messages by using the `dmesg | tail` command, for example:

```
struct dlm_ctxt: 82DA8137A49A47E4B187F74E09FBBB4B, node=3, key=965960985
  lockres: M000000000000006672078b84822, owner=1, state=0 last used: 0,
  on purge list: no granted queue:
      type=3, conv=-1, node=3, cookie=11673330234144325711,
ast=(empty=y,pend=n),
      bast=(empty=y,pend=n)
    converting queue:
    blocked queue:
```

The DLM supports three lock modes: no lock (`type=0`), protected read (`type=3`), and exclusive (`type=5`). In the previous example, the lock is owned by node 1 (`owner=1`) and node 3 has been granted a protected-read lock on the file-system resource.

9. Use the following command to search for processes that are in an uninterruptable sleep state, which are indicated by the `D` flag in the `STAT` column:

```
ps -e -o pid,stat,comm,wchan=WIDE-WCHAN-COLUMN
```

Note that at least one of the processes that are in the uninterruptable sleep state is responsible for the hang on the other node.

If a process is waiting for I/O to complete, the problem could be anywhere in the I/O subsystem, from the block device layer through the drivers, to the disk array. If the hang concerns a user lock (`flock()`), the problem could lie with the application. If possible, kill the holder of the lock. If the hang is due to lack of memory or fragmented memory, you can free up memory by killing non-essential processes. The most immediate solution is to reset the node that is holding the lock. The DLM recovery process can then clear all of the locks owned by the dead node owned; thus, enabling the cluster to continue to operate.

Configuring the Behavior of Fenced Nodes With Kdump

If the hearbeat mechanism detects that a node with a mounted OCFS2 volume has lost contact with the other cluster nodes, that node is removed from the cluster in a process called *fencing*. Fencing prevents other nodes from hanging while attempting to access resources that are held by the fenced node. By default, a fenced node automatically restarts to be able to quickly rejoin the cluster.

However, under some circumstances, this default behavior might not be desirable. For example, if a node frequently restarts for no apparent reason, then causing the node to panic instead of restarting is preferable to enable you to troubleshoot the issue. By enabling Kdump on the node, you can obtain a `vmcore` crash dump from the fenced node, which you can analyze to diagnose the cause of frequent node restarts.

To configure a node to panic at the next fencing, set `fence_method` to `panic` by runnng the following command on the node after the cluster starts:

```
echo "panic" | sudo tee /sys/kernel/config/cluster/cluster-name/
fence_method
```

To set the value after each system reboot, add the same line to the `/etc/rc.local` file.

To restore the default behavior, change the value of `fence_method` back to `reset`.

```
echo "reset" | sudo tee /sys/kernel/config/cluster/cluster-name/
fence_method
```

Likewise, remove the `panic` line from `/etc/rc.local` if the line exists in the file.

4
Managing Samba

This chapter includes information about managing Samba in Oracle Linux 8.

For information about local file system management in Oracle Linux, see Oracle Linux 8: Managing Local File Systems.

About Samba

Samba is an open source implementation of the Server Message Block (SMB) protocol that enables Oracle Linux to interoperate with clients that are running different OSs, including Windows, other Linux flavors, and macOS.

You can configure Samba to share both file and print resources from a Linux server. For example, you might use Samba to configure a Linux host to create a file share location that Windows users on the network can access using the following syntax:

```
\\samba_server\share_name
```

In the preceding example:

- *samba_server* is the IP address or a network resolvable host name of the system running the Samba service.
- *share_name* is the name of the resource as defined in the Samba configuration file `/etc/samba/smb.conf`.

> **Note:**
>
> The format of the location path can vary depending upon the client operating system. For example, on Unix and Linux based OSs, including macOS, the path format might appear as follows: `smb://samba_server/share_name`.

Samba includes capability for integrating with a Windows workgroup and an Active Directory (AD) domain.

Samba implements the Distributed Computing Environment Remote Procedure Call (DCE RPC) protocol that's used by Microsoft Windows to provision file and print services for Windows clients.

Samba uses the NetBIOS over TCP/IP protocol so that computer applications that depend on the NetBIOS API can work on TCP/IP networks.

About Samba Services

The Samba server consists of the following important services:

`smb` Service

The `smb` service enables file sharing and printing services by using the SMB protocol. This service is also responsible for resource locking and for authenticating connecting users.
The `smb systemd` service starts and stops the `smbd` daemon. The following is an example command:

```
sudo systemctl start smb.service
```

To use the `smbd` service, you need to install the `samba` package on the system.

`nmb` Service

The `nmb` (NetBIOS Message Block) service provides host name and IP resolution by using the NetBIOS over IPv4 protocol. The `nmb` service also enables browsing of the SMB network to locate domains, workgroups, hosts, file shares, and printers.
The `nmb systemd` service starts and stops the `nmbd` daemon. The following is an example command:

```
sudo systemctl start nmb.service
```

To use the `nmbd` service, you need to install the `samba` package on the system.

`winbind` Service

The `winbind` service is a Name Service Switch (NSS) daemon for resolving AD Users and Groups. The daemon enables AD Users to securely access services that are hosted on the Samba server.
The `winbind systemd` service starts and stops the `winbindd` daemon. The following is an example command:

```
sudo systemctl start winbind.service
```

To use the `winbindd` service, install the `samba-winbind` package.

> **Note:**
>
> If you're setting up Samba as a domain member, you must start the `winbind` service before starting the `smb` service. Otherwise, domain users and groups aren't available to the local system.

About the Samba Configuration File

Samba uses the `/etc/samba/smb.conf` file to manage Samba configuration.

`smb.conf` File Structure Overview

The `smb.conf` file consists of several sections that you can configure to support the required services for a specific Samba configuration. Consider the following sample extract from an `smb.conf` file:

```
#========== Global Settings =======
[global]
security = ADS
realm = EXAMPLE.REALM
password server = krbsvr.example.com
.
.
.
load printers = yes
printing = cups
printcap name = cups

#========== Share Definitions =========

[homes]
comment = User home directories
path = /data/pchome/%S
valid users = %S, WWW.EXAMPLE.COM\%S
browsable = no
read only = no
guest ok = no

[printers]
comment = All Printers
path = /var/spool/samba
printable = yes


[test_share]
comment = Shared /usr/local/test_share directory created for tests
path = /usr/local/test_share
valid users = @examplegroup
browsable = yes
read only = no
```

The following list describes the sections in the preceding configuration example:

`[global]`
This section contains global settings for the Samba server.
In the preceding example, the `security` parameter value of `ADS` means the server is a member of an AD domain that's running in native mode. In this scenario, Samba relies on tickets issued by the Kerberos server to authenticate clients who want to access local services.

[homes]
The `[homes]` section provides a personal share for users that log onto the Samba server. In the example, the location of each user's home directory is set by the line `path = /data/pchome/%S` (the `%S` macro is substituted with the username). The settings for `browsable = no` and `read only = no` prevent other users from browsing home directories, while granting full access to valid users.

> **Important:**
>
> Be careful with settings, such as security, especially in the special sections `[global]`, `[homes]`, and `[printers]`.
> For example, if guest access is specified in the `[homes]` section, all home directories are visible to all clients without a password.
> You might consider using the `invalid users` parameter for users such as `root` and other users with administrative privileges.

[printers]
Specifies support for print services. The `path` parameter specifies the location of a spooling directory that receives print jobs from Windows clients before submitting them to the local print spooler.
Samba advertises all locally configured printers on the server.

[*test_share*]
Specifies a share named *test_share*, which grants users belonging to group *examplegroup* browsing and write permissions to the `/usr/local/test_share` directory.

> **Note:**
>
> The `read only = no` configuration entry is essential to ensure Samba shares the directory as a writeable share.

For more information see `/etc/samba/smb.conf.example`, `smb.conf(5)` manual page, and https://wiki.samba.org/index.php/User_Documentation

Using the `testparm` Program to Validate Samba Configuration File Content

You use the `testparm` program to validate a Samba configuration file after making configuration changes. The `testparm` program detects invalid parameters and values, as well as any incorrect settings such as incorrect ID mappings.

> **Note:**
>
> `testparm` checks a configuration file for internal correctness only. The `testparm` command isn't capable of testing whether configured services are available or work as expected.

The following example shows how you might use the command to test a copy of the file you're working on:

```
sudo testparm /etc/samba/smb.conf.my_copy


Load smb config files from /etc/samba/smb.conf.my_copy
Loaded services file OK.
...
```

If, instead of a copy, you want to test the default Samba configuration file, you do not have to specify the file as a parameter. You can simply run testparm as follows:

```
sudo testparm
```

If the `testparm` command reports any errors or misconfiguration in the configuration file, you must fix the problem and then reissue the command.

For more information, see the `testparm(1)` manual page.

Best Practice When Editing Samba Configuration

Samba services reload their configuration as follows:

- Most configuration values are reloaded automatically, every 3 minutes.
- You can also manually request a reload, for example by using the `smbcontrol all reload-config command`.

> **Note:**
>
> Some parameters, such as `security`, require a restart of the `smb` service to take effect.

The frequent reloading of configuration values does not give you much time to validate any changes you're planning to make to `/etc/samba/smb.conf`. Therefore, as best practice, first test the changes on a copy of the configuration file. The following steps describe how you might do this:

1. Make a copy of the samba configuration file.

   ```
   sudo cp /etc/samba/smb.conf /etc/samba/samba.conf.mycopy
   ```

2. Edit the copy of the file after opening it in an editor of your choice, for example `vi`:

   ```
   sudo vi /etc/samba/samba.conf.mycopy
   ```

3. Validate the changes using `testparm`:

   ```
   sudo testparm /etc/samba/smb.conf.my_copy
   ```

4. Overwrite the original file with the copy you have validated:

```
sudo mv /etc/samba/samba.conf.my_copy /etc/samba/smb.conf
```

5. Use the `smbcontrol all reload-config` command to reload the configuration:

```
sudo smbcontrol all reload-config
```

About Samba Server Roles

The following sections give an overview of different roles you can configure for a Samba server.

Standalone

You can configure the Samba server role as a standalone server in small networks, for instance peer-to-peer workgroups, where the server isn't required to be part of a domain.

To provide a Windows user with authenticated access to a share on a standalone server, you create the following accounts on the Samba server:

- **A Local Linux Account**

  The local Linux account is required to validate access to local file system objects.

- **A Samba Account**

  In a standalone configuration, Samba authenticates users to a local database rather than a domain controller. You use the Samba `smbpasswd` command to create such accounts.

In addition to authenticated access, you can also enable guest access for users to connect to some services without authentication.

Domain Member of an Active Directory Domain

> **Note:**
>
> Oracle Linux doesn't support running Samba as an AD domain controller (DC)

You configure a Samba server's role to be a domain member of an Active Directory (AD) domain when you need to set up Samba shares in an AD domain network.

The Samba AD domain member setup requires the following:

- **Installation of `Kerberos`**

  The Samba server uses `Kerberos` to authenticate Windows AD users against the domain controller.

- **Installation of the `winbind` service**

  The `winbind` service provides information about Windows AD users and groups to the Linux OS. Hence, when a Samba server is configured as an AD member, you

do not need to manually create local Linux users and groups for authenticated access to the Samba shares.

- **Configuration of ID Mapping Back Ends**

  Samba provides various ID mapping methods, referred to as back ends, that can be configured to map each Linux `GID` and `UID` to its corresponding Windows `SID`. You choose which back ends you want to use and configure them in the `/etc/samba/smb.conf` file.

  See ID Mapping Back Ends in the Active Domain Member Setup

ID Mapping Back Ends in the Active Domain Member Setup

Linux and Windows each use a different ID system to identify groups and users.

Linux assigns unique GID and UID numbers to groups and users, whereas Windows uses a Security Identifier values (SID) for each user and group.

The winbind service maintains the necessary mapping of each Linux GID and UID to its corresponding Windows SID. However, you're responsible for specifying which of the available mapping methods, or back ends as they're called in Samba, are to be used for this mapping.

Overview of ID Mapping in the Samba Configuration File

You configure the mapping in the `[global]` section of the Samba configuration file `/etc/samba/smb.conf`.

Consider the following example extract from an `/etc/samba/smb.conf` file:

```
#========== Global Settings =======
[global]
security = ADS
.
.
.
#.......................................
#   Using tdb backend for default* domain.
#   UID/GID range 1000000-2000000
#.......................................
idmap config * : backend = tdb
idmap config * : range = 1000000-2000000


#.......................................
#   Using rid backend to map EXAMPLE.COM users
#   UID/GID range 10000-49999
#.......................................
idmap config EXAMPLE.COM : backend  = rid
idmap config EXAMPLE.COM: range = 10000-49999



#.......................................
```

```
#   Using rid backend to map EXAMPLE.NET users
#   UID/GID range 50000-99999
#.....................................
idmap config EXAMPLE.NET : backend  = rid
idmap config EXAMPLE.NET : range = 50000 -99999
```

The preceding example extract shows the following configurations:

- The Samba server is a member of the *EXAMPLE.COM* AD domain and uses the `rid` backend to map `SIDs` belonging to that domain. The backend is authoritative for those `SIDs` that the `rid` method translates to `UIDs` and `GIDs` within the range specified in the file (`10000-49999`).
- The Samba server also provides share access to a trusted AD domain *EXAMPLE.NET*. The trusted domain is also configured to use the `rid` backend. The range for *EXAMPLE.NET* is `50000-99999`.
- The default `*` domain uses backend `tdb`. The `tdb` range is specified as `1000000-2000000`

> **WARNING:**
>
> - ID ranges must not overlap.
> - Only one range can be assigned to a domain.
> - Once a range has been set and Samba has started using the range, you can only increase the upper number of the range. Any other change to the range can result in new ID assignments, and thus a loss of file ownership data.

The following sections give a further overview on using the different backends to configure ID Mapping for domains.

For more information, see `/etc/samba/smb.conf.example` and the `smb.conf(5)` manual pages. See also https://wiki.samba.org/index.php/User_Documentation and upstream documentation.

Domains That Require ID Mapping Configuration

You need to configure ID Mapping for the following:

- The AD domain of which the Samba server is a member.
- Each trusted AD domain that might access the Samba Server.
- The `*` default domain.

  The default domain includes Samba built-in accounts and groups, such as `BUILTIN\Administrators`.

Available Back Ends

The following table describes the most commonly used back ends and their different use cases.

Table 4-1 The Most Commonly Used ID Mapping Back Ends

Back End	Domains With Which the Back End Can Be Used
tdb	Use with * default domain only.
ad	Use with AD domains.
rid	Use with AD domains.
autorid	Can be used both with AD, and * default domain.

The following sections give an overview of the back ends listed in the preceding table.

tdb Mapping Back End

The tdb back end is the default back end used by winbindd for storing Security Identifier (SID), UID, and GID mapping tables.

The tdb back end must only be used for the * default domain.

The default domain includes Samba built-in accounts and groups, such as BUILTIN\Administrators.

The tdb back end is a writeable backend that needs to allocate new user and group IDs to create new mappings.

The ID mappings are local to the server.

ad Mapping Back End

The ad backend enables winbind to read the id mappings from an AD server that uses RFC2307 schema extensions.

For example, when using the ad backend, you set a user's Linux UID number by entering its value in their AD account's uidNumber attribute.

Some attributes that you set in the Windows AD Server are listed in the following table's first column and the corresponding Linux value each one maps to in the second column:

Table 4-2 Table of Attributes on the AD Server When ad Mapping is Used

Attribute Set on Windows AD Server	Corresponding Linux Value to Which AD Attribute Maps
uidNumber	UID
gidNumber	GID
sAMAccountName	User Name or Group Name

> **Note:**
>
> - The list in the preceding table is given as an overview. See upstream documentation for more attributes.
> - The mapping IDs must be within the range configured in `/etc/samba/smb.conf`. Objects with IDs outside the range will not be available on the Samba server.

Advantages of `ad` include the following:

- `UIDs` and `GIDs` are consistent on all Samba servers that use `ad`.
- The ID values are not stored in a local database, so there is less chance of local data corruption and loss of file ownership data.

`rid` Mapping Back End

The `rid` back end is an algorithmic mapping scheme that uses the `RID` (relative identifier) portion of the Windows `SID` to map Windows groups and Users to `UIDs` and `GIDs`.

Advantages of `rid` include the following:

- All domain user accounts and groups are automatically available on the domain member providing the mapped ID falls within the domain's `rid` range specified in `/etc/samba/smb.conf`.
- No attributes need to be set for domain users and groups.

`autorid` Mapping Back End

The `autorid` back end works in a similar way to the `rid` ID mapping back end, but one advantage of `autorid` is that it can automatically assign IDs for different domains. You can use the `autorid` back end for the following:

- The `*` default domain and additional domains, without the need to create ID mapping configurations for each of the additional domains.
- Only for specific domains.

Configuring a Samba Standalone Server

The following procedure shows one way of configuring a Samba standalone server:

1. Install the `samba` package:

   ```
   sudo dnf install samba
   ```

2. Edit the `/etc/samba/smb.conf` file and configure the various sections to support the services that are required for the specific configuration.

> **Note:**
>
> See Best Practice When Editing Samba Configuration for more information on editing a Samba configuration file.

Consider the following example:

```
#========== Global Settings =========
[global]

workgroup = EXAMPLE_WORKGROUP
netbios name = Server_Netbios_Name

security = user
role = standalone server
passdb backend = tdbsam
log file = /var/log/samba/%m
log level = 1

#========== File Share =========

[shareexample]
path = /srv/samba/shareexample/
read only = no
```

The preceding example configures a standalone server named *Server_Netbios_Name* with the workgroup *EXAMPLE_WORKGROUP* with the following settings:

- `role = standalone server`

  Type of server that you're setting up.

- `passdb backend = tdbsam`

  Default setting in which Samba stores user accounts in the `/var/lib/samba/private/passdb.tdb` database.

- `log level = 1`

  Lowest level of logging.

- `log file = /var/log/samba/%m`

  Path to the log file. The `%m` variable is substituted with the NetBIOS name of the client machine.

- `[shareexample]`

  Name of the share is configured as *shareexample*. This is the name users use to access the share.

- `read only = no`

  Ensures that Samba shares the directory as a writeable share.

3. Verify the Samba configuration file:

```
sudo testparm
```

4. Create a local Linux user account without a home directory (`-M`) and without a login shell (`-s /sbin/nologin`):

```
sudo useradd -M -s /sbin/nologin exampleUser
```

5. Enable the user by setting a password:

```
sudo passwd exampleUser
     Changing password for user exampleUser.
     New password:
     Retype new password:
     passwd: all authentication tokens updated successfully.
```

> **Note:**
>
> The local password set with the `passwd` account is not the one used by Samba.
>
> However, setting a local password is required to enable the account (Samba denies access if the account is disabled locally).

6. Add *exampleUser* account to the Samba database:

```
sudo smbpasswd -a exampleUser
New SMB password:
Retype new SMB password:
Added user exampleUser
```

> **Note:**
>
> The Samba account password set in this step, using the `smbpasswd` command, is the password that will be used by Samba

7. Create a Linux Group:

```
sudo groupadd exampleGroup
```

Add the user *exampleUser* to the group *exampleGroup* created in the previous step:

```
sudo usermod -aG exampleGroup exampleUser
```

8. Make the share directory referenced in the named share in `/etc/samba/smb.conf` if it doesn't already exist:

```
sudo mkdir -p /srv/samba/shareexample/
```

> **Note:**
>
> If you run SELinux in `enforcing` mode, set the `samba_share_t` context on the directory so that SELinux allows Samba to read and write to it:
>
> ```
> sudo semanage fcontext -a -t samba_share_t "/srv/samba/
> shareexample(/.*)?"
> sudo restorecon -Rv /srv/samba/shareexample/
> ```

9. Set the group for the share directory to be the group you created in a previous step for the Samba users:

   ```
   sudo chgrp -R exampleGroup /srv/samba/shareexample/
   ```

10. Set permissions for the shared directory:

    ```
    sudo chmod 2770 /srv/samba/shareexample/
    ```

11. Open the required ports and reload the firewall configuration using the `firewall-cmd` utility:

    ```
    sudo firewall-cmd --permanent --add-service=samba
    sudo firewall-cmd --reload
    ```

12. Enable and start the `smb` service:

    ```
    systemctl enable --now smb
    ```

Configuring a Samba Server as an AD Member

The following procedure shows one way of configuring a Samba server as an AD member:

1. You need to install the following packages:

   - `realmd`

     The `realmd` tool is used to for joining `Kerberos` realms, such as Active Directory domains.

   - `oddjob` and `oddjob-mkhomedir`

     `oddjob` is a D-Bus service which runs jobs on behalf of client applications.

     `oddjob-mkhomedir` is an `oddjob` helper which creates and populates home directories.

   - `samba-winbind-clients`, `samba-winbind`, `samba-common-tools`, `samba-winbind-krb5-locator`, and `samba`.

     You can run the following command to install the required packages:

     ```
     sudo dnf install realmd \
      oddjob-mkhomedir \
     ```

```
oddjob  \
samba-winbind-clients \
samba-winbind  \
samba-common-tools  \
samba-winbind-krb5-locator  \
samba
```

2. Make a backup copy of the Samba configuration file:

   ```
   sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.copy
   ```

3. Use the `realm join` command to join AD domain. The following example assumes you want to join domain *`EXAMPLE.COM`*:

   ```
   sudo realm join --membership-software=samba \
    --client-software=winbind EXAMPLE.COM
   ```

   When you run the command as shown, `realm` does the following:

   - Creates the `/etc/samba/smb.conf` file with membership of the *EXAMPLE.COM* domain configured.
   - Adds the `winbind` module for user and group lookups to the `/etc/nsswitch.conf` file.
   - Updates the Pluggable Authentication Module (PAM) configuration files in the `/etc/pam.d/`directory
   - Starts and enables the `winbind` service.

4. Set up ID Mapping in the `/etc/samba/smb.conf` file required in the configuration.

   For further details on ID Mapping see ID Mapping Back Ends in the Active Domain Member Setup

5. Check that the entries in the `/etc/samba/smb.conf` file meet all configuration requirements.

   For more information on the configuration file see About the Samba Configuration File

6. Verify that the `winbind` service is running:

   ```
   sudo systemctl status winbind
   ```

   > **Important:**
   >
   > The `winbind` service must be running before you start the `smb` service. Otherwise, Samba can't retrieve domain user and group information.

7. After verifying that the `winbind` is running in the preceding step, start and enable the `smb` service:

   ```
   sudo systemctl enable --now smb
   ```

8. Perform verification steps such as the following:

   - Get the details of a domain user. The following assumes the details of user *exampleuser* in the *EXAMPLE.COM* domain are being retrieved:

     ```
     sudo getent passwd EXAMPLE.COM\\exampleuser
     ```

     ```
     EXAMPLE.COM\exampleuser:*:10000:10000::/home/
     exampleuser@EXAMPLE.COM:/bin/bash
     ```

   - Test the command to get users from the "`Domain Users`" group in the domain. The following assumes the details of users in the "`Domain Users`" group in the *EXAMPLE.COM* domain are being retrieved:

     ```
     sudo getent group "EXAMPLE.COM\Domain Users"
     ```

     ```
     EXAMPLE.COM\domain users:x:10000:exampleuser1,exampleuser2
     ```

   - Confirm that you can use domain users and groups when using file and directory commands. For example, to set the owner of the */srv/samba/shareexample/* directory to *EXAMPLE.COM*`\administrator` and the group to *EXAMPLE.COM*`\Domain Users` run the following command:

     ```
     sudo chown "EXAMPLE.COM\administrator":"EXAMPLE.COM\Domain
     Users" /srv/samba/shareexample
     ```

Accessing Samba Shares

The following tasks describe how to access Samba shares from a Windows client and an Oracle Linux client.

Accessing Samba Shares From a Windows Client

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

```
\\server_name\share_name
```

If you enter `\\server_name`, Windows displays the directories and printers that the server is sharing. You can also use the same syntax to map a network drive to a share name.

Accessing Samba Shares From a Client

To access a Samba share from an Oracle Linux host you can install the following packages:

- `samba-client`

  Installing the `samba-client` package gives you the `smbclient` utility that provides SFTP-like commands to access Samba shares. For example, `smbclient` provides a `get`

command for downloading a file from a remote Samba share, and a `put` command for uploading a file.

- `cifs-utils`

  Installing the `cifs-utils` package enables you to mount a Samba share.

Using `smbclient` Commands

The following steps give a brief overview of how you might use the `smbclient` commands:

1. Log onto a share *example_samba_share* hosted on server *example_samba_server* using account *EXAMPLE.COM/user1*:

   ```
   sudo smbclient -U "EXAMPLE.COM\user1" //example_samba_server/
   example_samba_share
   ```

2. Change to directory location */directory1/* :

   ```
   smb: \> cd /directory1/
   ```

3. Download file *ExampleFile.txt*:

   ```
   smb: \directory1\> get ExampleFile.txt
   ```

4. End the session:

   ```
   smb: \directory1\> exit
   ```

For more information, see the `smbclient(1)` manual page.

Mounting a Samba Share with `cifs`

To mount a Samba share, use a command similar to the following:

```
sudo mount -t cifs //server_name/share_name mountpoint -o
credentials=credfile
```

In the previous command, the credentials file contains settings for `username`, `password`, and `domain`:

```
username=username
password=password
domain=EXAMPLE.COM
```

The parameter to `domain` can be the name of a domain or a workgroup.

> **Caution:**
>
> Because the credentials file contains a plain-text password, use `chmod` to make it readable by only you, for example:
>
> ```
> sudo chmod 400 credfile
> ```

If the Samba server is a domain member server in an AD domain, and the current login session was authenticated by the Kerberos server in the domain, you can use the existing session credentials by specifying the `sec=krb5` option instead of a credentials file:

```
sudo mount -t cifs //server_name/share_name mountpoint -o sec=krb5
```

For more information, see `mount.cifs(8)` manual page.