A Simple Guide to Oracle Cluster File System (OCFS2) using iSCSI on Oracle Cloud Infrastructure

Article Number: 151 | Rating: Unrated | Last Updated: Sat, Jun 2, 2018 9:26 PM

A Simple Guide to Oracle Cluster File System (OCFS2) using iSCSI on Oracle Cloud Infrastructure

By: Gilson Melo | Senior Product Manager

Oracle Cluster File System version 2 (OCFS2) is a general-purpose shared-disk file system intended for use in clusters to increase storage performance and availability. Almost any application can use OCFS2 because it provides local file-system semantics. 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 the available file-system functionality to fail over and run on another node in the event that a node fails.

This blog describes the deployment steps for Oracle Cluster File System (OCFS2) on Oracle Cloud Infrastructure using iSCSI as the underlying storage.

Here is an example of the OCFS2 architecture that will be used for this tutorial on Oracle Cloud Infrastructure. It spreads resources across availability domains (ADs) for fault tolerance, which is the recommended configuration.

Why iSCSI?

As explained in the public documentation, iSCSI is is an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. By carrying SCSI commands over IP networks, iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. iSCSI can used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet, and can enable location-independent data storage and retrieval.

iSCSI enables clients (called Initiators) to send SCSI commands (CDBs) to SCSI storage devices (LinuxIOs) on remote servers. It is a popular SAN protocol, allowing organizations to consolidate storage into data center storage arrays while providing hosts (such as database and web servers) with the illusion of locally-attached disks. Unlike traditional Fibre Channel, which requires special-purpose cabling, iSCSI can be run over long distances using existing network infrastructure.

OCFS2

OCFS2 has a large number of features that make it suitable for deployment in an enterprise-level computing environment:

- *- Support for ordered and write-back data journaling that provides file system consistency in the event of power failure or system crash.*
- *- Block sizes ranging from 512 bytes to 4 KB, and file-system cluster sizes ranging from 4 KB to 1 MB (both in increments of powers of 2). The maximum supported volume size is 16 TB, which corresponds to a cluster size of 4 KB. A volume size as large as 4 PB is theoretically possible for a cluster size of 1 MB, although this limit has not been tested.*

- *- Extent-based allocations for efficient storage of very large files.*
- *- Optimized allocation support for sparse files, inline-data, unwritten extents, hole punching, reflinks, and allocation reservation for high performance and efficient storage.*
- *- Indexing of directories to allow efficient access to a directory even if it contains millions of objects.*
- *- Metadata checksums for the detection of corrupted inodes and directories.*
- *- Extended attributes to allow an unlimited number of name:value pairs to be attached to file system objects such as regular files, directories, and symbolic links.*
- *- Advanced security support for POSIX ACLs and SELinux in addition to the traditional file-access permission model.*
- *- Support for user and group quotas.*
- *- Support for heterogeneous clusters of nodes with a mixture of 32-bit and 64-bit, little-endian (x86, x86_64, ia64) and big-endian (ppc64) architectures.*
- *- An easy-to-configure, in-kernel cluster-stack (O2CB) with a distributed lock manager (DLM), which manages concurrent access from the cluster nodes.*
- *- Support for buffered, direct, asynchronous, splice and memory-mapped I/O.*
- *- A tool set that uses similar parameters to the ext3 file system.*

Getting Started

This tutorial requires at least three Oracle Bare Metal instances; one instance will be the iSCSI Target server and the remaining two ones will be used as the iSCSI Initiator Servers and also as nodes of the OCFS2 cluster with a local mount point to the OCFS2 volume.

Below is a summary of the configuration steps required for this architecture:

1. Configure your iSCSI Target and Initiator bare metal Instances

2. Set up your OCFS2/O2CB cluster Nodes

3. Create your OCFS2 file system and mount point

You also need to open ports 7777 and 3260 on the Oracle Cloud Infrastructure Dashboard. Edit the VCN Security List and either open all ports for your tenancy Internal Network (*NOT PUBLIC NETWORK*) as shown below for network 172.0.0.0/16

Source: 172.0.0.0/16

IP Protocol: All Protocols

Allows: all traffic for all ports

or open only the required 7777 and 3260 ports for the internal network and here is an example for port 7777:

Source: 172.0.0.0/16

IP Protocol: TCP

Source Port Range: All

Destination Port Range: 7777

Allows: TCP traffic for ports: 7777

Make sure DNS is working properly and your bare metal instances can communicate properly across your tenancy availability domains (ADs). Here is a quick example of /etc/resolv.conf based on this setup

```
$ cat /etc/resolv.conf

; generated by /usr/sbin/dhclient-script

search baremetal.oraclevcn.com publicsubnetad3.baremetal.oraclevcn.com publicsubnetad1.baremet

nameserver 169.254.169.254
```

As you can see above, all ADs DNS entries are available in that resolv.conf file.

The Oracle Linux firewall also needs to be configured to open the relevant ports on the local bare metal instances as well.

```
iSCSI Initiator Nodes

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

$ sudo firewall-cmd --reload

iSCSI Target

$ sudo firewall-cmd --permanent --add-port=3260/tcp

$ sudo firewall-cmd --reload
```

Environment

ROLE	INSTANCE	IP
iSCSI Target	target.publicsubnetad3.baremetal.oraclevcn.com	172.0.2.
iSCSI Initiator - OCFS2 Node1	initiator1.publicsubnetad1.baremetal.oraclevcn.com	172.0.0.
iSCSI Initiator - OCFS2 Node2	initiator2.publicsubnetad2.baremetal.oraclevcn.com	172.0.1.

Storage Configuration

Provision a bare metal HighIO1.36 to be able to use the NVMe disk(s) as the iSCSI targets. You can also use virtual machine (VM) shapes with attached block storage disks - either option will work. Use the proper shape based on your workload requirements. For this tutorial, we are going to use a HighIO1.36 compute instance shape and combine all 4 NVMe disks into a single LVM logical volume (~12Tb) for the OCFS2 volume.

LVM Logical Volume

The HighIO1.36 bare metal shape has 4 NVMe disks (~12TB). This is the process to create a single logical volume that will be used as an iSCSI Target:

```
$ sudo pvcreate /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
```

```
$ sudo vgcreate vg_ocfs2 /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
```

```
$ sudo  lvcreate -L 11500G -n target vg_ocfs2
```

iSCSI target

You can create your iSCSI target with authentication or without authentication. For this tutorial we will not use CHAP authentication.

Configuring iSCSI Targets without CHAP Authentication

Install the targetcli package on the server.

```
$ sudo yum install targetcli -y
```

Once you installed the package, enter the following command to get a iSCSI CLI for an interactive prompt.

```
$ sudo targetcli
Warning: Could not load preferences file /root/.targetcli/prefs.bin.
targetcli shell version 2.1.fb41
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.
>
```

Use an existing logical volume (/dev/vg_ocfs2/target) as a block-type backing store for

storage object “ocfs2“.

```
/> cd backstores/block

/backstores/block> create name=ocfs2 dev=/dev/vg_ocfs2/target

Created block storage object ocfs2 using /dev/vg_ocfs2/target.
```

Create a target

```
/backstores/block> cd /iscsi

/iscsi> create iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa

Created target iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa.

Created TPG 1.

Global pref auto_add_default_portal=true

Created default portal listening on all IPs (0.0.0.0), port 3260.

/iscsi>
```

By default, authentication is enabled, so disable it for this tutorial

```
cd /iscsi/iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa/tpg1/acls
```

```
/iscsi/iqn.20...3fa/tpg1/acls> create iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa

Created Node ACL for iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa

/iscsi/iqn.20...3fa/tpg1/acls> cd /iscsi/iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa/tpg

/iscsi/iqn.20...653673fa/tpg1> set attribute authentication=0

Parameter authentication is now '0'.

/iscsi/iqn.20...653673fa/tpg1> set attribute generate_node_acls=1

Parameter generate_node_acls is now '1'.
```

Create a LUN under the target. The LUN should use the previously mentioned backing storage object named “ocfs2”

```
/iscsi/iqn.20...653673fa/tpg1> cd /iscsi/iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa/t

/iscsi/iqn.20...3fa/tpg1/luns> pwd

/iscsi/iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa/tpg1/luns

/iscsi/iqn.20...3fa/tpg1/luns>create /backstores/block/ocfs2

Created LUN 0.
```

Verify the target server configuration.

```
/iscsi/iqn.20...3fa/tpg1/luns> cd /

/> ls

o- / ..... [...]

 o- backstores ..... [...]

 | o- block ..... [Storage Objects: 2]

 | | o- ocfs2 ..... [/dev/vg_ocfs2/target (11.3TiB) write-thru acti

 | | o- sbd ..... [/dev/vg_ocfs2/sbd (5.0GiB) write-thru activa

 | o- fileio ..... [Storage Objects: 0]

 | o- pscsi ..... [Storage Objects: 0]

 | o- ramdisk ..... [Storage Objects: 0]

 o- iscsi ..... [Targets: 1]

 | o- iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa .....

 |  o- tpg1 ..... [gen-acls, no-auth]

 |  o- acls ..... [ACLs: 0]

 |  o- luns ..... [LUNs: 2]

 |  | o- lun0 ..... [block/ocfs2 (/dev/vg_ocfs2/target)]
```

```
|   | o- lun1 ..... [block/sbd (/dev/vg_ocfs2/sbd)]
|   o- portals ..... [Portals: 1]
|     o- 0.0.0.0:3260 ..... [OK]
o- loopback ..... [Targets: 0]
/>
/> saveconfig
/> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
```

Enable and restart the target service.

```
$ sudo systemctl enable target.service
$ sudo systemctl start target.service
```

iSCSI Initiator

Configure the Initiator without CHAP authentication

It is time to configure your bare metal iSCSI Initiator1 and Initiator2 nodes to use an iSCSI target as storage. The following steps need to be performed on both iSCSI Initiator Nodes. Iscsi-initiator-utils package should be installed by default in Oracle Linux image but double check with the below command.

```
$ sudo yum install iscsi-initiator-utils -y
```

Discover the target using this command:

```
$ sudo iscsiadm -m discovery -t st -p  172.0.2.40

172.0.2.40:3260,1 iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa
```

Edit this file and add the iscsi initiator name

```
$ sudo vi /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa
```

Restart and enable the initiator service.

```
$ sudo systemctl enable iscsid.service

$ sudo systemctl restart iscsid.service
```

Login to the discovered target.

```
$ sudo iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa -p 172.0.2.4
```

```
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa, por
```

```
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.x8664:sn.4131653673fa, portal: 1
```

Both iSCSI Initiator bare metal Instances should be able to see the new disk now

```
$ sudo fdisk -l |grep sdb
```

```
Disk /dev/sdb: 12455.4 GB, 12455405158400 bytes, 24326963200 sectors
```

OCFS2

Creating the Configuration File for the Cluster Stack

Install the required OCFS2 packages

```
$ sudo yum install ocfs2-tools-devel ocfs2-tools -y
```

Now, create the configuration file by using the o2cb command or a text editor. Lets use the following command to create a cluster definition.

```
$ sudo o2cb add-cluster ociocfs2
```

The above command creates the configuration file /etc/ocfs2/cluster.conf if it does not already exist.

For each node, use the following command to define the node.

```
$ sudo o2cb add-node ociocfs2 initiator1 --ip 172.0.0.41
```

```
$ sudo o2cb add-node ociocfs2 initiator2 --ip 172.0.1.42
```

NOTE: The name of the node must be same as the value of the system's HOSTNAME that is configured in /etc/sysconfig/network and the IP address is the one that the node will use for private communication in the cluster. You need to ***copy*** the cluster configuration file /etc/ocfs2/cluster.conf to each node in the cluster. Any changes that you make to the cluster configuration file do not take effect until you restart the cluster stack.

The following /etc/ocfs2/cluster.conf configuration file defines a 2-node cluster named ociocfs2 with a local heartbeat which is the configuration used for this tutorial.

```
$ sudo cat /etc/ocfs2/cluster.conf
```

```
cluster:
```

```
    heartbeat_mode = local
```

```
    node_count = 2
```

```
    name = ociocfs2
```

```
node:
        number = 0
        cluster = ociocfs2
        ip_port = 7777
        ip_address = 172.0.0.41
        name = initiator1

node:
        number = 1
        cluster = ociocfs2
        ip_port = 7777
        ip_address = 172.0.1.42
        name = initiator2
```

Configuring the Cluster Stack

Run the following command on each node of the cluster:

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

Configuring the O2CB driver.

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

Load O2CB driver on boot (y/n) [y]:
Cluster stack backing O2CB [o2cb]:
Cluster to start on boot (Enter "none" to clear) [ocfs2]: ociocfs2
Specify heartbeat dead threshold (>=7) [31]:
Specify network idle timeout in ms (>=5000) [30000]:
Specify network keepalive delay in ms (>=1000) [2000]:
```

```
Specify network reconnect delay in ms (>=2000) [2000]:

Writing O2CB configuration: OK

checking debugfs...

Setting cluster stack "o2cb": OK

Registering O2CB cluster "ociocfs2": OK

Setting O2CB cluster timeouts : OK

Starting global heartbeat for cluster "ociocfs2": OK
```

Explanation of the above options can be found in OCFS2

To verify the settings for the cluster stack, enter the /sbin/o2cb.init status command:

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

Driver for "configfs": Loaded

Filesystem "configfs": Mounted

Stack glue driver: Loaded

Stack plugin "o2cb": Loaded

Driver for "ocfs2_dlmfs": Loaded

Filesystem "ocfs2_dlmfs": Mounted
```

```
Checking O2CB cluster "ociocfs2": Online
  Heartbeat dead threshold: 31
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Local
Checking O2CB heartbeat: Active
Debug file system at /sys/kernel/debug: mounted
```

In this example, the cluster is online and is using local heartbeat mode. If no volumes have been configured, the O2CB heartbeat is shown as Not Active rather than Active.

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

```
$ sudo systemctl enable o2cb
$ sudo systemctl enable ocfs2
```

These settings allow the node to mount OCFS2 volumes automatically when the system starts.

Configuring the Kernel for Cluster Operation

For the correct operation of the cluster, you must configure the kernel settings shown in the following table:

KERNEL SETTING	DESCRI
panic	Specifie itself. If the va the pani To enab (vmcore seconds
panic_on_oops	Specifie cluster c be able operatio

On each node, enter the following commands to set the recommended values for panic and panic_on_oops:

```
$ sudo sysctl kernel.panic=30
$ sudo sysctl kernel.panic_on_oops=1
```

To make the change persist across reboots, add the following entries to the /etc/sysctl.conf

file:

```
# Define panic and panic_on_oops for cluster operation

kernel.panic=30

kernel.panic_on_oops=1
```

Starting and Stopping the Cluster Stack

The following table shows the commands that you can use to perform various operations on the cluster stack.

COMMAND	DESCRI
/sbin/o2cb.init status	Check t
/sbin/o2cb.init online	Start the
/sbin/o2cb.init offline	Stop the
/sbin/o2cb.init unload	Unload

Creating OCFS2 volumes

Use mkfs.ocfs2 command to create an OCFS2 volume on a device. If you want to label the volume and mount it by specifying the label, the device must correspond to a partition. You cannot mount an unpartitioned disk device by specifying a label.

```
$ sudo mkfs.ocfs2 -L "ocfs2" /dev/sdb

mkfs.ocfs2 1.8.6
```

Cluster stack: classic o2cb

Label: ocfs2

Features: sparse extended-slotmap backup-super unwritten inline-data strict-journal-super xattr inde

Block size: 4096 (12 bits)

Cluster size: 4096 (12 bits)

Volume size: 12455405158400 (3040870400 clusters) (3040870400 blocks)

Cluster groups: 94274 (tail covers 512 clusters, rest cover 32256 clusters)

Extent allocator size: 780140544 (186 groups)

Journal size: 268435456

Node slots: 16

Creating bitmaps: done

Initializing superblock: done

Writing system files: done

Writing superblock: done

Writing backup superblock: 6 block(s)

Formatting Journals: done

Growing extent allocator: done

```
Formatting slot map: done

Formatting quota files: done

Writing lost+found: done

mkfs.ocfs2 successful
```

Mounting OCFS2 Volumes

As shown in the following example, specify "_netdev" and "nofail" options in /etc/fstab if you want the system to mount an OCFS2 volume at boot time after networking is started, and to unmount the file system before networking is stopped.

```
$ sudo mkdir /ocfs2

$ sudo vi /etc/fstab

#include the below line to mount your ocfs2 after a restart

/dev/sdb /ocfs2 ocfs2     _netdev,defaults   0 0
```

Run "mount -a" to mount your OCFS2 partition based on the fstab entry you created above and your OCFS2 using iSCSI on bare metal instances setup is concluded. Validate your configuration by verifying your mounted file system devices as the following images illustrates. You should have a cluster file system mounted on /ocfs2 on both Initiator1 and Initiator2 Oracle Linux 7.4 Nodes.

Finally, you're finished! Your applications can use this storage as they would any local file storage.
Planning your environment thoughtfully and making use of Availability Domains and capabilities such as Oracle Cluster File System can help you increase the performance and availability of the solutions you build on Oracle Cloud Infrastructure.
https://docs.oracle.com/cd/E37670_01/E37355/html/ol_ocfs2.html