ORACLE LINUX

openstack CLOUD SOFTWARE

ORACLE VM

Oracle OpenStack for Oracle Linux High Availability Guide

Release 1.0

ORACLE WHITE PAPER | SEPTEMBER 2014

ORACLE

Contents

Introduction

A High Availability (HA) cluster is a set of servers (nodes) managed by clusterware to work together for minimizing application downtime as much as possible. Clusterware enables servers to communicate with each other, so that they appear to function as a collective unit. In an active-passive HA cluster, there is one active server running the application, and one or more nodes standing by. If any problem were to occur on the active node that prevents it from running the application, the clusterware is able to detect this problem and start the protected applications on one of the standby nodes.

Oracle Clusterware provides the infrastructure necessary to run Oracle Real Application Clusters (Oracle RAC). Oracle Clusterware can also manage user applications and other resources, such as virtual IP addresses. This document shows you how to use Oracle Clusterware to manage Oracle OpenStack nodes to make its services HA.

Active-Passive HA Deployment Architecture

How to deploy OpenStack requires careful planning, in very large deployment, different services should be distributed on different nodes for redundancy, isolation and better performance. The OpenStack servers can be grouped into the following categories:

- Controller node
- Networking controller node
- Service node
- Compute node

The controller node is also known as the *API* node. This node contains the API services which work as an endpoint or service hub for all OpenStack clients (dashboard or command line tools). When a request arrives on controller node, the API services route the request to backend service nodes. When the request is processed, the results are returned to the API node, and forwarded to the requester. In a small deployment scenario, the major OpenStack services may be installed together with the API services on the controller node.

The networking controller node is a node where networking agent services are running. These agents include the Layer 2 Agent, Layer 3 Agent, Metadata Agent, LBaaS Agent, and DHCP Agent. These agents can also be deployed on the controller node, but since all virtual machine traffic passes through these agents, in networking traffic intensive deployments, dedicated servers should be used to run these agents.

The service node is any node running OpenStack services, these services can be any of Glance, Cinder, Swift, Nova, and so on. These services nodes are responsible for processing requests distributed from the controller nodes.

The compute node is where the virtual machine is running. Only the nova-compute service and the Layer 2 Agent (neutron-openvswitch-agent) services are running on this node.

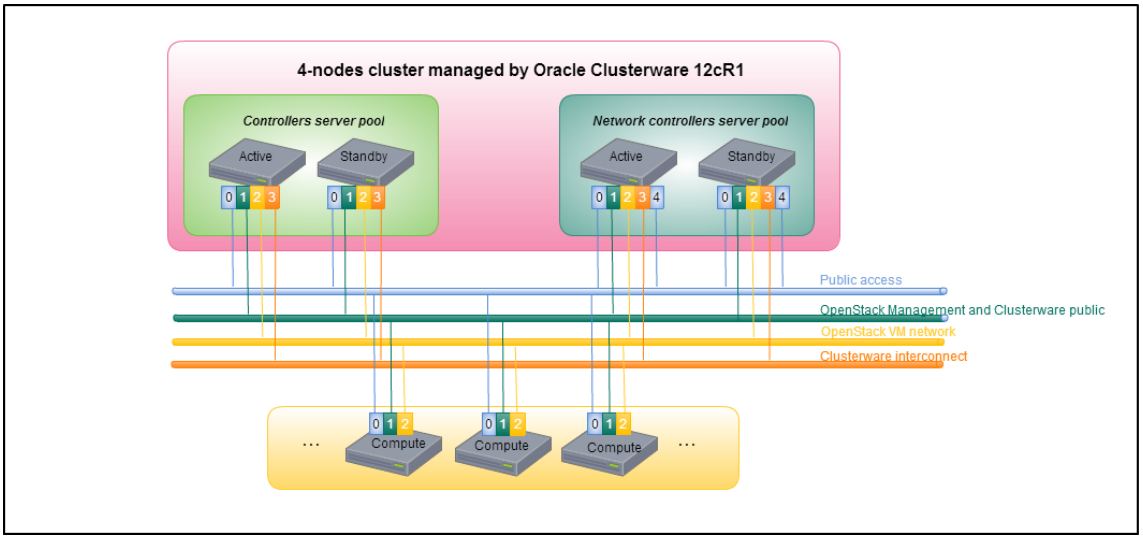

Figure 1. Active-Passive HA Deployment Architecture

Figure 1 shows the typical active-passive HA configuration. All four nodes are configured in one cluster, two controller nodes are grouped in one server pool while the other two network controller nodes are in another server pool. When registering the OpenStack services, different services should be placed in different server pools, based on their functions.

In this Guide, we assume the controller nodes are running not only the API services, but also some other service, such as: Glance, Cinder, and Nova. This Guide provides commands to register not only the API services, but also other services.

Active-Passive HA Networking

In an OpenStack active-passive HA configuration, four networks should be configured, as shown in the following table.

TABLE 1. OPENSTACK ACTIVE-PASSIVE HA CONFIGURATION NETWORKS

Network	Purpose	Nodes
Public access	A network for SSH logins	All nodes
OpenStack management and Clusterware public	A private network used to manage OpenStack and Clusterware	All nodes
OpenStack VM	A network for virtual machine traffic	All nodes
Clusterware interconnect	A network for Clusterware interconnect	Controller and network controller nodes

In some deployments, the storage traffic may be isolated in an independent network for better I/O performance. This Guide assumes the network topology shown in the above table.

Install Oracle Grid Infrastructure

Oracle's OpenStack release leverages Oracle Grid Infrastructure to provide HA. To begin setting up your environment to use active-passing HA and OpenStack, you first need to install the Oracle Grid Infrastructure Release 12c Release 1 or later. Then you should create a cluster on two controller nodes, and on two network controller nodes. See the *Oracle Grid Infrastructure Installation Guide* for information on how to install and configure Oracle Grid Infrastructure:

http://docs.oracle.com/database/121/CWLIN/

After Oracle Grid Infrastructure is installed, and the cluster is configured, you can check the cluster status using command:

```
# /u01/app/12.1.0/grid/bin/crsctl check cluster -all
```

You should see output similar to the following to show all cluster services are online.

```
**************************************************************
ctr1:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
ctr2:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
ctr3:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
ctr4:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
```

You can check the status of the servers using the command:

```
# /u01/app/12.1.0/grid/bin/crsctl stat server
```

You should see output similar to the following:

```
NAME=ctr1
STATE=ONLINE
NAME=ctr2
STATE=ONLINE
NAME=ctr3
STATE=ONLINE
NAME=ctr4
STATE=ONLINE
```

In order to place services on different servers, you should create server pools respectively for controller nodes and Neutron nodes, all controller nodes related services should be set up on the two controller nodes only. Similarly, all networking related services should be set up on the two Neutron nodes. For example, to add two controller nodes (**ctr1** and **ctr2**) to a server pool (**controller_sp**), enter:

```
# /u01/app/12.1.0/grid/bin/crsctl add serverpool controller_sp -attr "PARENT_POOLS=Generic,
SERVER_NAMES=ctr1 ctr2"
```

To add two network nodes (**ctr3** and **ctr4**) to a server pool (**network_sp**), enter:

```
# /u01/app/12.1.0/grid/bin/crsctl add serverpool network_sp -attr "PARENT_POOLS=Generic,
SERVER_NAMES=ctr3 ctr4"
```

Deploy OpenStack on Nodes

After installing and configuring the cluster, you should next install and deploy OpenStack on the nodes.

To install the Neutron agent services on a dedicated node, specify the following parameters, which are different from the parameter used to define the controller hosts:

- --neutron-l3-hosts
- --neutron-dhcp-hosts
- --neutron-lbaas-hosts
- --neutron-metadata-hosts

For example, enter:

```
# packstack --install-host=10.20.20.1,10.20.20.3 --neutron-server-host=10.20.20.1 --neutron-l3-
hosts=10.20.20.2 --neutron-dhcp-hosts=10.20.20.2 --neutron-lbaas-hosts=10.20.20.2 --neutron-metadata-
hosts=10.20.20.2
```

In the above command, the IP address of the controller node is 10.20.20.1, the IP address of the compute node is 10.20.20.3, and the network services are installed on a dedicated node with the IP address of 10.20.20.2. Unlike other Neutron agent services, the neutron-server service should be installed on the controller node.

Run this command again using the appropriate IP addresses for your environment to deploy the OpenStack software on the standby controller and network nodes.

After the packstack installation succeeds, stop all OpenStack related services on all nodes using the following script:

```
#!/bin/sh

SERVICES=`ls /etc/init.d/neutron* | cut -d/ -f4`
SERVICES="$SERVICES `ls /etc/init.d/openstack* | cut -d/ -f4`"
SERVICES="$SERVICES mongod rabbitmq-server mysqld"

for s in $SERVICES; do
    service $s stop
    chkconfig $s off
done
```

Note that this script also disables services in chkconfig so the services are not automatically started on rebooting the node. Having these services start automatically may cause issues with Oracle Clusterware.

Configure iptables for Oracle Clusterware Traffic

When installing Oracle Clusterware, one interface should be dedicated for the Oracle Clusterware interconnect network (private network). The traffic in this network must not be blocked by the iptables service. The iptables service is a critical part of OpenStack. The Neutron service provides state-of-the-art networking functionality by combining the power of the L2 Agent (openvswitch by default) and iptables. You should add a rule to the `INPUT` chain of iptables to accept any packages from the Oracle Clusterware interconnect network interface. In this example, we use the interface named **eth2** for the interconnect network. To insert the rule, enter a command similar to:

```
# iptables -I INPUT 1 -i eth2 -j ACCEPT
```

After this rule is inserted, save the rules to make them persistent after the node is rebooted.

```
# service iptables save
```

Install OpenStack HA Package

By using customizable *action* scripts and application agent programs, as well as resource attributes that you assign to applications and processes, Oracle Clusterware can manage all these entities to provide HA. Action scripts define the start, stop, and check operations for a resource. The start action is invoked while starting the resource, the stop action for stopping the resource, and the check action while checking the running status of a resource.

To make OpenStack HA, you should register the OpenStack services with Oracle Clusterware by supplying action scripts and resource attributes. The opentack-ha-utils RPM includes all the action scripts required for OpenStack services.

This package can be installed from the Oracle OpenStack channel using the command:

```
# yum install -y openstack-ha-utils
```

This package should be installed on two controller nodes and two network controller nodes. You do not need to install it on the other service nodes and compute nodes, which are not configured in the cluster.

After installation, you can access the action scripts in the /opt/openstack-ha/ directory.

Configure HA for MySQL

To enable HA for the MySQL service:

1. **Create a virtual IP address resource and bind the MySQL service to this address**. A virtual IP address resource is a standard resource for which Oracle Clusterware has a built-in action file to manage its start/stop/check operations. Create a virtual IP address resource using the command:

   ```
   # /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.11 -vipname=mysqlVIP -
   user=root
   ```

   You should see output similar to the following:

   ```
   Production Copyright 2007, 2008, Oracle. All rights reserved
   2014-08-10 21:00:47: Creating Resource Type
   2014-08-10 21:00:47: Executing /u01/app/12.1.0/grid/bin/crsctl add type app.appvip net1.type -
   basetype ora.cluster vip net1.type -file /u01/app/12.1.0/grid/crs/template/appvip.type
   2014-08-10 21:00:47: Executing cmd: /u01/app/12.1.0/grid/bin/crsctl add type app.appvip_net1.type
   -basetype ora.cluster_vip_net1.type -file /u01/app/12.1.0/grid/crs/template/appvip.type
   2014-08-10 21:00:47: Create the Resource
   2014-08-10 21:00:47: Executing /u01/app/12.1.0/grid/bin/crsctl add resource mysqlVIP -type
   app.appvip_net1.type -attr "USR_ORA_VIP=10.10.10.11,START_DEPENDENCIES=hard(ora.net1.network)
   pullup(ora.net1.network),STOP_DEPENDENCIES=hard(ora.net1.network),ACL='owner:root:rwx,pgrp:root:r-
   x,other::r--,user:root:r-x',HOSTING MEMBERS=ctr1,APPSVIP FAILBACK="
   2014-08-10 21:00:47: Executing cmd: /u01/app/12.1.0/grid/bin/crsctl add resource mysqlVIP -type
   app.appvip_net1.type -attr "USR_ORA_VIP=10.10.10.11,START_DEPENDENCIES=hard(ora.net1.network)
   pullup(ora.net1.network),STOP_DEPENDENCIES=hard(ora.net1.network),ACL='owner:root:rwx,pgrp:root:r-
   x,other::r--,user:root:r-x',HOSTING_MEMBERS=ctr1,APPSVIP_FAILBACK="
   ```

2. **Move the MySQL data directory to shared storage**. The MySQL data directory is located at /var/lib/mysql. Remount this directory to shared storage, where both the controller nodes have access. The shared storage may be a NAS export (for example, an. NFS file system), or a cluster file system based on shared LUN (for example, an OCFS2 file system). This example uses an NFS file system named **myfileserver.example.com:/mysql_data**, which is exported to both controller nodes.

   To move the MySQL data directory, enter:

   ```
   # mount myfileserver.example.com:/mysql data /mnt
   # mv /var/lib/mysql/* /mnt
   # umount /mnt
   ```

3. Register the shared data directory and MySQL service with Oracle Clusterware. Like the virtual IP address, the /var/lib/mysql mount also needs be registered with Oracle Clusterware. First, make sure this mount exists before the MySQL service is started. To do this, specify the NFS export in the /etc/openstack-ha-data.conf file.

```
MYSQL_DATA_EXPORT='myfileserver.example.com:/mysql_data'
MYSQL_DATA_PATH='/var/lib/mysql'
```

The same setting should be applied on the other controller node. The action script for this resource is /opt/openstack-ha/mysql-data.scr. This action script reads the above setting while performing the start/stop/check operations.

Register this resource:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource mysql-data -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/mysql-
data.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,STAR
T_DEPENDENCIES=hard(mysqlVIP),STOP_DEPENDENCIES=hard(mysqlVIP)"
```

Where:

PLACEMENT: Specifies how Oracle Clusterware selects a cluster server on which to start a resource. Valid values are `balanced`, `favored`, or `restricted`.

If you set the `PLACEMENT` attribute to `favored` or `restricted`, then you must also assign values to the `SERVER_POOLS` and `HOSTING_MEMBERS` attributes. If you set the value of the `PLACEMENT` attribute to `balanced`, then the `HOSTING_MEMBERS` attribute is not required.

CHECK_INTERVAL: Specifies the interval, in seconds, between which the check action is repeated. Shorter intervals enable more frequent checks, but also increase resource consumption if you use the script agent.

START_DEPENDENCIES: Specifies a set of relationships Oracle Clusterware considers when starting a resource.

STOP_DEPENDENCIES: Specifies a set of relationships Oracle Clusterware considers when stopping a resource.

RESTART_ATTEMPTS: Specifies the number of times Oracle Clusterware attempts to restart a resource on the resource's current server before attempting to relocate it. A value of `1` indicates that Oracle Clusterware only attempts to restart the resource once on a server. A second failure causes Oracle Clusterware to attempt to relocate the resource. A value of `0` indicates that there is no attempt to restart, but Oracle Clusterware always tries to fail the resource over to another server.

For more information on the attributes of Oracle Clusterware resource, see the *Oracle Clusterware Administration and Deployment Guide* at:

http://docs.oracle.com/cd/E11882_01/rac.112/e16794/toc.htm

For the MySQL service, use the /opt/openstack-ha/mysqld.scr script as action script when performing the registration. This resource depends on the mysql-data resource, which mounts the shared data storage before the mysqld service is started.

```
# /u01/app/12.1.0/grid/bin/crsctl add resource mysqld -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-
ha/mysqld.scr,PLACEMENT=restricted,SERVER POOLS=controller sp,CHECK INTERVAL=30,RESTART ATTEMPTS=2
,START_DEPENDENCIES=hard(mysql-data),STOP_DEPENDENCIES=hard(mysql-data)"
```

Start the MySQL services:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource mysqlVIP mysql-data mysql -n ctr1
```

4. **Modify the SQL connection setting in all OpenStack services configuration files on both nodes to connect to the virtual IP address**. Open the configuration files listed in table below. Edit the

`sql_connection` or `connection` entry to use the new virtual IP address. The changes should be made to the configuration files on both nodes.

TABLE 2. MYSQL CONNECTION STRING CONFIGURATION FILES

Service	Configuration File
Keystone	/etc/keystone/keystone.conf
Glance	/etc/glance/glance-api.conf /etc/glance/glance-registry.conf
Nova	/etc/nova/nova.conf
Cinder	/etc/cinder/cinder.conf
Neutron	/etc/neutron/neutron.conf

Configure HA for RabbitMQ

To enable HA for the RabbitMQ service:

1. **Create a virtual IP address resource and bind the RabbitMQ service to this address**. A virtual IP address resource is an Oracle Clusterware resource with a built-in action file to manage its start/stop/check operations. You can create a virtual IP address resource using the following command:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.12 -vipname=rabbitmqVIP -
user=root
```

   Here, we use a different virtual IP address from that of the MySQL service. This means that if MySQL is relocated to the standby node, the RabbitMQ service can remain running on the active node, without interruption.

   The RabbitMQ service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address.

2. **Move the RabbitMQ data directory to shared storage**. The RabbitMQ data directory is located at /var/lib/rabbitmq. Remount this directory to shared storage where both controller nodes have access. The shared storage may be a NAS export (for example, an. NFS file system), or a cluster file system based on shared LUN (for example, an OCFS2 file system). This example uses an NFS file system named **myfileserver.example.com:/rabbitmq_data**, which is exported to both controller nodes.

   Move the RabbitMQ data files, including the cookie, to the export, using the command:

```
# mount myfileserver.example.com:/rabbitmq_data /mnt
# mv /var/lib/rabbitmq/* /mnt
# umount /mnt
```

3. **Register the shared data directory and RabbitMQ service with Oracle Clusterware**. Like the virtual IP address, the /var/lib/rabbitmq mount should be registered with Oracle Clusterware.

   First, specify the NFS export in the /etc/openstack-ha-data.conf file. The same setting should be applied on the other controller node.

```
RABBITMQ_DATA_EXPORT='myfileserver.example.com:/rabbitmq_data'
RABBITMQ_DATA_PATH='/var/lib/rabbitmq'
```

   The action script for this resource is /opt/openstack-ha/rabbitmq-data.scr. This action script reads the above setting during the start/stop/check operations.

   Register this resource with the command:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource rabbitmq-data -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/rabbitmq-
data.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,STAR
T_DEPENDENCIES=hard(rabbitmqVIP),STOP_DEPENDENCIES=hard(rabbitmqVIP)"
```

For the RabbitMQ service, use the /opt/openstack-ha/rabbitmq-server.scr action script when performing the registration. This resource depends on the rabbitmq-data resource, which mounts the shared data storage before the rabbitmq-server service is started.

```
# /u01/app/12.1.0/grid/bin/crsctl add resource rabbitmq-server -type cluster_resource -attr
"ACTION SCRIPT=/opt/openstack-ha/rabbitmq-
server.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,ST
ART_DEPENDENCIES=hard(rabbitmq-data),STOP_DEPENDENCIES=hard(rabbitmq-data)"
```

Start the RabbitMQ services:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource rabbitmqVIP rabbitmq-data rabbitmq-server -n ctr1
```

4. **Modify the RabbitMQ connection settings in all OpenStack services configuration files on both nodes to connect to the virtual IP address**. Open the configuration files listed in the table below. Edit the `rabbit_host` and `rabbit_hosts` entries to use the new virtual IP address of the RabbitMQ service. The changes should be made to the configuration files on both nodes.

TABLE 3. RABBITMQ CONNECTION STRING CONFIGURATION FILES

Service	Configuration File
Glance	/etc/glance/glance-api.conf
Nova	/etc/nova/nova.conf
Cinder	/etc/cinder/cinder.conf
Neutron	/etc/neutron/neutron.conf

Configure HA for OpenStack

This section describes the steps required to configure HA for the OpenStack services. The OpenStack services you should configure for HA are:

- Keystone
- Cinder
- Glance
- Swift
- Nova
- Neutron

Configure HA for Keystone

To enable HA for the Keystone service:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for the Keystone service with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.13 -vipname=keystoneVIP -
user=root
```

The Keystone service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address. However, if you want to bind to the virtual IP address exclusively, edit the `public_bind_host` and `admin_bind_host` entries in the /etc/keystone/keystone.conf file.

2. **Register the Keystone service with Oracle Clusterware**. Since the Keystone service already has a generic service lifecycle management script (/etc/init.d/openstack-keystone), it can be registered with Oracle Clusterware using the command:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource openstack-keystone -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-
keystone.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,
START_DEPENDENCIES=hard(keystoneVIP),STOP_DEPENDENCIES=hard(keystoneVIP)"
```

   Start the Keystone service:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource keystoneVIP openstack-keystone -n ctr1
```

3. **Create a Keystone service endpoint using the virtual IP address**. Run `keystone service-list` to query the Keystone service UUID. Then run `keystone endpoint-create` to create an endpoint for the virtual IP address

```
# keystone service-list | grep keystone
| 8f6da7beb4c949a08b667782ae0fa281 |  keystone  |   identity   |   OpenStack Identity Service   |

# keystone endpoint-create --region RegionOne --service-id 8f6da7beb4c949a08b667782ae0fa281 --
publicurl 'http://10.10.10.13:5000/v2.0' --adminurl 'http://10.10.10.13:35357/v2.0' --internalurl
'http://10.10.10.13:5000/v2.0'
```

4. **Modify any other service configuration files to access the virtual IP address Keystone service endpoint**. Open the configuration files specified in the table below. Edit the `auth_host`, `*_auth_url`, and `auth_url` entries to use the virtual IP address of the Keystone service. The changes should be made to files on all nodes.

TABLE 4. KEYSTONE SERVICE CONFIGURATION FILES

Service	Configuration File
Glance	/etc/glance/glance-api.conf /etc/glance/glance-registry.conf /etc/glance/glance-cache.conf
Nova	/etc/nova/nova.conf
Cinder	/etc/cinder/api-paste.ini
Neutron	/etc/neutron/neutron.conf /etc/neutron/api-paste.ini /etc/neutron/metadata_agent.ini
Swift	/etc/swift/proxy-server.conf
Horizon	/etc/openstack-dashboard/local_settings (change `OPENSTACK_HOST` entry)

5. **Synchronize the PKI certificate files and keys for both nodes**. Both controller nodes should have identical PKI certificate files and private keys. If they are different on the nodes, when the Keystone service is relocated from one node to the other, all authentication and authorization requests fail. By default, the PKI files are in /etc/keystone/ssl/private and /etc/keystone/ssl/certs. The location of these files can be changed in the signing section of the /etc/keystone/keystone.conf file.

```
# scp /etc/keystone/ssl/private/  root@ctr2:/etc/keystone/ssl/private/
# scp /etc/keystone/ssl/certs/  root@ctr2:/etc/keystone/ssl/certs/
```

Configure HA for Cinder

Cinder is the OpenStack volume service. If not explicitly specified, the Cinder API service and its other services are installed on the controller node by packstack. To gain the best performance and scalability, the controller node should only have the Cinder API service installed.

To enable HA for the Cinder services:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for Cinder with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.14 -vipname=cinderVIP -
user=root
```

   The Cinder service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address. However, if you want to bind to the virtual IP address exclusively, edit the `osapi_volume_listen` entry in the /etc/cinder/cinder.conf file.

2. **Register the Cinder services with Oracle Clusterware**. The Cinder services are:

   - openstack-cinder-api
   - openstack-cinder-backup
   - openstack-cinder-scheduler
   - openstack-cinder-volume

   For improved performance and security, you should deploy the Cinder API service on a controller node, separate from the other Cinder services, which should be deployed to other nodes In this case, only the Cinder API service needs to be registered with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource openstack-cinder-api -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-cinder-
api.scr,PLACEMENT=restricted,SERVER_POOLS=controller sp,CHECK INTERVAL=30,RESTART ATTEMPTS=2,START
_DEPENDENCIES=hard(cinderVIP),STOP_DEPENDENCIES=hard(cinderVIP)"
```

   Start the Cinder API service:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource cinderVIP openstack-cinder-api -n ctrl
```

   Alternatively, if all Cinder services are deployed in the controller node, register all Cinder services with Oracle Clusterware, for example:

```
for s in api backup scheduler volume; do
    /u01/app/12.1.0/grid/bin/crsctl add resource openstack-cinder-$s -type cluster_resource -attr
"ACTION SCRIPT=/opt/openstack-ha/openstack-cinder-
$s.scr,PLACEMENT=restricted,SERVER POOLS=controller sp,CHECK INTERVAL=30,RESTART ATTEMPTS=2,START
DEPENDENCIES=hard(cinderVIP),STOP DEPENDENCIES=hard(cinderVIP)"
Done
```

   Since both controller nodes may provide the volume service, change the default storage backend from iSCSI to NFS to make sure the volumes can be accessed from both controller nodes.

   Create a text file named **nfsshares** in /etc/cinder/ on both controller nodes. Add an entry to the /etc/cinder/nfsshares files for each NFS share the Cinder volume service should access for backend storage. Each entry should be a separate line, and should use the following format:

```
HOST:SHARE
```

   Where:

   HOST: Specifies the IP address or hostname of the NFS server.

   SHARE: Specifies the absolute path to an existing and accessible NFS share.

   Change ownership of the /etc/cinder/nfsshares file to be owned by the root user and a member of the cinder group. Also make the /etc/cinder/nfsshares file to be readable by members of the cinder group, for example:

```
# chown root:cinder /etc/cinder/nfsshares
# chmod 0640 /etc/cinder/nfsshares
```

   Configure the Cinder volume service to use the /etc/cinder/nfsshares file. Open the /etc/cinder/cinder.conf configuration file and change the `nfs_shares_config` configuration key to `/etc/cinder/nfsshares`.

Start the Cinder services:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource cinderVIP openstack-cinder-api openstack-cinder-
backup openstack-cinder-scheduler openstack-cinder-volume -n ctrl
```

3. **Create a new Cinder service endpoint using the virtual IP address**. Run `keystone service-list` to query the Cinder service UUID, then run `keystone endpoint-create` to create the endpoint for the virtual IP address. As there are two versions of the Cinder service, you should create two endpoints. For example:

```
# keystone service-list | grep cinder
| 04b2c1501ee34b5b982418a28e7459a2 |   cinder   |    volume    |         Cinder Service         |
| 69d9642f795247e78ea74b6e5e86c516 | cinder_v2  |   volumev2   |       Cinder Service v2        |

# keystone endpoint-create --region RegionOne --service-id 04b2c1501ee34b5b982418a28e7459a2 --
publicurl 'http://10.10.10.14:8776/v1/%(tenant id)s' --adminurl
'http://10.10.10.14:8776/v1/%(tenant_id)s' --internalurl
'http://10.10.10.14:8776/v1/%(tenant_id)s'

# keystone endpoint-create --region RegionOne --service-id 69d9642f795247e78ea74b6e5e86c516  --
publicurl 'http://10.10.10.14:8776/v2/%(tenant_id)s' --adminurl
'http://10.10.10.14:8776/v2/%(tenant id)s' --internalurl
'http://10.10.10.14:8776/v2/%(tenant_id)s'
```

Configure HA for Glance

To enable HA for the Glance services:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for the Glance service with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.15 -vipname=glanceVIP -
user=root
```

The Glance service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address. However, if you want to bind to the virtual IP address exclusively, edit the `bind_host` entry in the /etc/glance/glance-api.conf and /etc/glance/glance-registry.conf files.

2. **Register the Glance services with Oracle Clusterware**. The Glance services include:

   - openstack-glance-api
   - openstack-glance-registry
   - openstack-glance-scrubber

   For improved performance and security, you should deploy the Glance API service on a controller node, separate from the other Glance services, which should be deployed to other nodes In this case, only the Glance API service needs to be registered with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource openstack-glance-api -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-glance-
api.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,START
_DEPENDENCIES=hard(glanceVIP),STOP_DEPENDENCIES=hard(glanceVIP)"
```

Start the Glance API service:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource glanceVIP openstack-glance-api -n ctrl
```

Alternatively, if all the Glance services are deployed on a controller node, you should register all the services with Oracle Clusterware. As both controller nodes may provide the image service, make the image storage shared for these two nodes. If the Glance service fails over from an active node to a standby node, the images can still be retrieved from the standby node. The shared storage can be an NFS filesystem or an OCFS2 filesystem. In this example, we use the name **myfileserver.example.com:/glance_data** for the NFS export. Modify the /etc/openstack-ha-data.conf file as below. The same setting should be applied on the other controller node.

```
GLANCE_DATA_EXPORT='myfileserver.example.com:/glance_data'
GLANCE_DATA_PATH='/var/lib/glance'
```

Before you register the resource for the NFS export, create two directories in this export:

```
# mount myfileserver.example.com:/glance_data /mnt
# cd /mnt
# mkdir images
# mkdir scrubber
# chown -R glance:glance images
# chown -R glance:glance scrubber
# cd -
# umount /mnt
```

Register a resource to make sure this shared directory is mounted before the Glance services are started.

```
# /u01/app/12.1.0/grid/bin/crsctl add resource glance-data -type cluster resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/glance-
data.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,STAR
T_DEPENDENCIES=hard(glanceVIP),STOP_DEPENDENCIES=hard(glanceVIP)"
```

Register the Glance services using the script:

```
for s in api registry scrubber; do
    /u01/app/12.1.0/grid/bin/crsctl add resource openstack-glance-$s -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-glance-
$s.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,START_
DEPENDENCIES=hard(glance-data),STOP DEPENDENCIES=hard(glance-data)"
Done
```

Start the Glance services:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource glanceVIP glance-data openstack-glance-api
openstack-glance-registry openstack-glance-scrubber -n ctr1
```

3. Create a Glance service endpoint using the virtual IP address. First, run `keystone service-list` to query the Glance service UUID, then run `keystone endpoint-create` to create the endpoint for the virtual IP address:

```
# keystone service-list | grep glance
| a6033e6852334703ae21f9af68f7de98 |   glance   |    image     |    Openstack Image Service     |

# keystone endpoint-create --region RegionOne --service-id a6033e6852334703ae21f9af68f7de98 --
publicurl 'http://10.10.10.15:9292' --adminurl 'http://10.10.10.15:9292' --internalurl
'http://10.10.10.15:9292'
```

Modify all other service configuration files to access the virtual IP address for the Glance service endpoint. Open the configuration files specified in the table below. Edit the `glance_host` and `glance_api_servers` entries. The changes should be made to files on all nodes.

TABLE 5. GLANCE SERVICE CONFIGURATION FILES

Service	Configuration File
Nova	/etc/nova/nova.conf
Cinder	/etc/cinder/cinder.conf

Configure HA for Swift

Swift is a highly available, distributed, eventually consistent object/blob store. You can use Swift to store lots of data efficiently, safely, and cheaply. In a typical Swift deployment architecture, it requires multiple nodes to run different services of Swift to ensure data duplication and better performance. In all Swift services, the proxy service is responsible for tying together the rest of the Swift architecture. For each request, it looks up the location of the account, container, or object in the ring and routes the request accordingly. The public API is also exposed through the proxy server, so this service can be regarded as the API service for Swift. In this example we register only the Swift proxy service in Oracle Clusterware.

To enable HA for the Swift proxy service:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for the Swift proxy service with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.16 -vipname=swiftVIP -
user=root
```

2. **Bind the Swift proxy service to the virtual IP address**. Modify the Swift configuration file to bind its service to the virtual IP address. Edit the `bind_host` entry in the/etc/swift/proxy-server.conf file to use the virtual IP address of the Swift proxy service.

3. **Register the Swift proxy service with Oracle Clusterware**. Register the Swift proxy service with Oracle Clusterware with the command:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource openstack-swift-proxy -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-swift-
proxy.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,STA
RT_DEPENDENCIES=hard(swiftVIP),STOP_DEPENDENCIES=hard(swiftVIP)"
```

   Start the Swift proxy service:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource swiftVIP openstack-swift-proxy -n ctrl
```

4. **Create a Keystone service endpoint for the Swift proxy service using the virtual IP address**. First, run `keystone service-list` to query the Swift proxy service UUID, then run `keystone endpoint-create` to create an endpoint for the virtual IP address. For example:

```
# keystone service-list | grep swift
| 52ca8fa4545a40dcb75bc3ca3f22b535 |   swift    | object-store | Openstack Object-Store Service |
| aa2a4812478946c3a5a058bf423540e8 |  swift s3  |      s3      |      Openstack S3 Service      |

# keystone endpoint-create --region RegionOne --service-id 52ca8fa4545a40dcb75bc3ca3f22b535 --
publicurl 'http://10.10.10.16:8080/v1/AUTH_%(tenant_id)s' --adminurl 'http://10.10.10.16:8080' --
internalurl 'http://10.10.10.16:8080/v1/AUTH_%(tenant_id)s'

# keystone endpoint-create --region RegionOne --service-id aa2a4812478946c3a5a058bf423540e8 --
publicurl 'http://10.10.10.16:8080' --adminurl 'http://10.10.10.16:8080' --internalurl
'http://10.10.10.16:8080'
```

5. Modify other service configuration files to access this virtual IP address for the Swift proxy service endpoint. Open the configuration files specified in table below. Edit the `backup_swift_url` entry. The changes should be made to files on both nodes.

TABLE 6. SWIFT PROXY SERVICE CONFIGURATION FILES

Service	Configuration File
Cinder	/etc/cinder/cinder.conf

Configure HA for Nova

To enable HA for the Nova services:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for the Nova service with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.17 -vipname=novaVIP -user=root
```

   The Nova service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address. However, if you want to bind to the virtual IP address exclusively, edit the `ec2_listen` and `osapi_compute_listen` entries in the /etc/nova/nova.conf file.

2. **Register the Nova services with Oracle Clusterware**. The Nova services include:
   - openstack-nova-api
   - openstack-nova-cert

- openstack-nova-conductor
- openstack-nova-console
- openstack-nova-consoleauth
- openstack-nova-metadata-api
- openstack-nova-novncproxy
- openstack-nova-scheduler
- openstack-nova-spicehtml5proxy
- openstack-nova-xvpvncproxy

For improved performance and security, you should deploy the Nova API service on a controller node, separate from the other Nova services, which should be deployed to other nodes In this case, only the Nova API service needs to be registered with Oracle Clusterware, for example:

```
# /u01/app/12.1.0/grid/bin/crsctl add resource openstack-nova-api -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-nova-
api.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,START
_DEPENDENCIES=hard(novaVIP),STOP_DEPENDENCIES=hard(novaVIP)"
```

Start the Nova API service:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource novaVIP openstack-nova-api -n ctrl
```

Alternatively, if all the Nova services are deployed on a controller node, you should register all the services with Oracle Clusterware, for example:

```
for s in api cert conductor console consoleauth metadata-api novncproxy scheduler spicehtml5proxy
xvpvncproxy; do
    /u01/app/12.1.0/grid/bin/crsctl add resource openstack-nova-$s -type cluster_resource -attr
"ACTION_SCRIPT=/opt/openstack-ha/openstack-nova-
$s.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,START_
DEPENDENCIES=hard(novaVIP),STOP_DEPENDENCIES=hard(novaVIP)"
Done
```

Start all the Nova services with the command:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource novaVIP openstack-nova-api openstack-nova-cert
openstack-nova-conductor openstack-nova-consoleauth openstack-nova-novncproxy openstack-nova-
scheduler openstack-nova-spicehtml5proxy -n ctrl
```

The openstack-nova-console, openstack-nova-metadata-api, and openstack-nova-xvpvncproxy service are not enabled by default, so there is no need to start these services

3. **Create a Nova service endpoint using the virtual IP address**. First, run `keystone service-list` to query the Nova service UUID, then run `keystone endpoint-create` to create an endpoint for the virtual IP address:

```
# keystone service-list | grep nova
| 2e8324208b784c10be62b7533a111c1d |    nova    |   compute    |   Openstack Compute Service    |
| d59d368fc34b4055897fd6c51e074017 |  nova_ec2  |     ec2      |          EC2 Service           |

# keystone endpoint-create --region RegionOne --service-id 2e8324208b784c10be62b7533a111c1d --
publicurl 'http://10.10.10.17:8774/v2/%(tenant_id)s' --adminurl
'http://10.10.10.17:8774/v2/%(tenant_id)s' --internalurl
'http://10.10.10.17:8774/v2/%(tenant_id)s'

# keystone endpoint-create --region RegionOne --service-id d59d368fc34b4055897fd6c51e074017 --
publicurl 'http://10.10.10.17:8773/services/Cloud' --adminurl
'http://10.10.10.17:8773/services/Cloud' --internalurl 'http://10.10.10.17:8773/services/Cloud'
```

4. **Modify all other service configuration files to access the virtual IP address for the Nova services endpoint**. Open the configuration files specified in the table below. Edit the `nova_url` entry. The changes should be made to files on both network controller nodes.

TABLE 7. NOVA SERVICES CONFIGURATION FILES

Service	Configuration File
Neutron	/etc/neutron/neutron.conf

Configure HA for Neutron

The controller node has a number of Neutron services running to handle API requests. The services are neutron-server, and neutron-*-agent (the default is neutron-openvswitch-agent).

To enable HA for Neutron services:

1. **Register a virtual IP address resource with Oracle Clusterware**. Register a virtual IP address for the Neutron service with Oracle Clusterware, for example:

   ```
   # /u01/app/12.1.0/grid/bin/appvipcfg create -network=1 -ip=10.10.10.18 -vipname=neutronVIP -
   user=root
   ```

   The Neutron service, by default, is bound to the IP address 0.0.0.0, so it is bound to all interfaces. This means that there is no requirement to bind it to the virtual IP address. However, if you want to bind to the virtual IP address exclusively, edit the `bind_host` entry in the /etc/neutron/neutron.conf file.

2. **Register the Neutron services with Oracle Clusterware**. The Neutron services include:

   - neutron-server
   - neutron-openvswitch-agent

   Register these services with Oracle Clusterware:

   ```
   for s in server openvswitch-agent;do
       /u01/app/12.1.0/grid/bin/crsctl add resource neutron-$s -type cluster_resource -attr
   "ACTION_SCRIPT=/opt/openstack-ha/neutron-
   $s.scr,PLACEMENT=restricted,SERVER_POOLS=controller_sp,CHECK_INTERVAL=30,RESTART_ATTEMPTS=2,START_
   DEPENDENCIES=hard(neutronVIP),STOP_DEPENDENCIES=hard(neutronVIP)"
   Done
   ```

   Start the Neutron services:

   ```
   # /u01/app/12.1.0/grid/bin/crsctl start resource neutronVIP neutron-server neutron-openvswitch-
   agent -n ctrl
   ```

3. **Create a Neutron service endpoint using the virtual IP address**. First, run `keystone service-list` to query the Neutron service UUID, then run `keystone endpoint-create` to create the endpoint for the virtual IP address:

   ```
   # keystone service-list | grep neutron
   | beda3c73174b4024a3facd8d1649cd35 |  neutron   |   network    |   Neutron Networking Service   |

   # keystone endpoint-create --region RegionOne --service-id beda3c73174b4024a3facd8d1649cd35 --
   publicurl 'http://10.10.10.18:9696/' --adminurl 'http://10.10.10.18:9696/' --internalurl
   'http://10.10.10.18:9696/'
   ```

4. **Modify all other service configuration files to access the virtual IP address for the Neutron services endpoint**. On every compute node, change the `neutron_url` entry in the /etc/nova/nova.conf file.

Configure HA for Network Controller Nodes

The network controller node is a host where the Neutron agents are running. The Neutron network agents are:

- neutron-dhcp-agent
- neutron-l3-agent
- neutron-metadata-agent
- neutron-openvswitch-agent

» neutron-lbaas-agent

To protect these services using HA, register the services with Oracle Clusterware. You do not need to register a virtual IP address for these services. Make sure they are registered into the network controller server pool. For example:

```
for s in dhcp l3 metadata openvswitch lbaas; do
    /u01/app/12.1.0/grid/bin/crsctl add resource net.neutron-$s-agent -type cluster resource -attr
"ACTION SCRIPT=/opt/openstack-ha/neutron-$s-
agent.scr,PLACEMENT=restricted,SERVER POOLS=network sp,CHECK INTERVAL=30,RESTART ATTEMPTS=2"
Done
```

Start the Neutron services on the active network controller node:

```
# /u01/app/12.1.0/grid/bin/crsctl start resource net.neutron-dhcp-agent net.neutron-l3-agent
net.neutron-metadata-agent net.neutron-openvswitch-agent net.neutron-lbaas-agent -n ctr3
```

HA is now configured in your Oracle Openstack for Oracle Linux environment.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together