Providing High Availability to the OpenStack Cloud Controller on Oracle Solaris with Oracle Solaris Cluster

Release 1.0
# Table of Contents

## Introduction

Oracle Solaris Cluster Features

Overview of the High Availability Cloud Controller Configuration

Oracle Solaris Cluster Framework and Infrastructure Setup

  Configuration assumptions

  Configure zpools on shared storage and a dataset per node for later usage

  Zone cluster installation

OpenStack Components on the HA Cloud Controller

  Database - MySQL

    Configure cluster infrastructure objects for HA MySQL within zone cluster os-db

    Install and configure the MySQL database within zone cluster os-db

    Configure the MySQL database as HA MySQL failover cluster resource os-mysql-rs within zone cluster os-db

  Message Queue - RabbitMQ

    Configure cluster infrastructure objects for HA rabbitmq within zone cluster os-mq

    Configure rabbitmq using the cluster option with mirrored queues within zone cluster os-mq

    Configure rabbitmq as scalable cluster resources within zone cluster os-mq

Configure cluster infrastructure objects for OpenStack components running within zone cluster os-api

Keystone
Create Keystone database within zone cluster os-db..........................................................22
Configure Keystone within zone cluster os-api.................................................................22
Configure the Keystone SMF service as a failover cluster resource within zone cluster os-api..................................................................................................................23
Glance................................................................................................................................23
Create the Glance database within zone cluster os-db......................................................25
Configure Glance services within zone cluster os-api.....................................................26
Configure Glance SMF services as failover cluster resources within zone cluster os-api..................................................................................................................27
Nova..................................................................................................................................28
Create the Nova database within zone cluster os-db......................................................29
Configure Nova services within zone cluster os-api.....................................................29
Configure the Nova SMF services as failover cluster resources within zone cluster os-api..................................................................................................................30
Horizon............................................................................................................................32
Configure the horizon/https services within zone cluster os-api..................................33
Configure horizon/https services as HA Apache scalable cluster resource within zone cluster os-api............................................................................................................33
Neutron...........................................................................................................................34
Elastic Virtual Switch (EVS) Controller........................................................................34
Neutron server..............................................................................................................37
Neutron L3 and Neutron DHCP agent...........................................................................39
Cinder............................................................................................................................44
Create the Cinder database within zone cluster os-db

Configure the Cinder services within zone cluster os-api

Configure the Cinder SMF services as failover cluster resources within zone cluster os-api

Cinder-volume services when using the ZFSSAISCSIDriver volume driver for the ZFS Storage Appliance

Configure the cinder-volume SMF services within zone cluster os-api

Configure the cinder-volume SMF services as GDSv2 based failover cluster resources within zone cluster os-api

Swift

Configure swift-proxy-server service within zone cluster os-api

Configure the swift-proxy-server SMF service as a failover cluster resource within zone cluster os-api

OpenStack Components Not Running on the HA Cloud Controller

References

Appendix

SMF manifest for service system/cluster/osc-clpriv-ip-forwarding-disable

SMF method script for service system/cluster/osc-clpriv-ip-forwarding-disable

SMF wrapper script to manage the cinder-volume services
Introduction

Oracle Solaris delivers a complete OpenStack distribution which is integrated with its core technologies such as Oracle Solaris Zones, the ZFS file system, or its image packaging system (IPS). OpenStack in Oracle Solaris 11.2 helps IT organizations to create an enterprise-ready Infrastructure as a Service (IaaS) cloud, so that users can quickly create virtual networking and compute resources by using a centralized web-based portal [1].

Any enterprise-type OpenStack deployment requires a highly available OpenStack infrastructure that can sustain individual system failures [11],[12].

Oracle Solaris Cluster is deeply integrated with Oracle Solaris technologies and provides a rich set of features which provide high availability to Oracle Solaris based OpenStack services [2].

The primary goals of the Oracle Solaris Cluster software are to maximize service availability through fine-grained monitoring and automated recovery of critical services and to prevent data corruption through proper fencing.

Figure 1 below depicts just one example on a highly available physical node OpenStack infrastructure deployment. The two cloud controller cluster nodes in yellow represent the highly available OpenStack cloud controller configuration described in the sections to follow. Of course, the OpenStack cloud controller can be deployed on a larger cluster, based on the capacity required by the deployment and on the qualified configurations documented within the Oracle Solaris Cluster 4 Compatibility Guide [3].

The usage of OpenStack Swift object storage nodes is optional. High availability for the Swift object storage is achieved by Swift's clustering features by configuring a Swift ring [4]. Only the swift-proxy-server service will be hosted by the HA cloud controller.

The clustered Oracle ZFS Storage Appliance (ZFS SA) is used to provide highly available shared storage to the cloud controller cluster, and is also used to provide Cinder volume storage through the corresponding Oracle ZFS Storage Appliance iSCSI Cinder driver [5].
This white paper describes further how to provide high availability to an OpenStack cloud controller that is deployed on Oracle Solaris with Oracle Solaris Cluster.
Oracle Solaris Cluster Features

The Oracle Solaris Cluster environment extends the Oracle Solaris operating system into a cluster operating system. A cluster is a collection of one or more nodes that belong exclusively to that collection. The following explains some of the key features offered by Oracle Solaris Cluster and used in this example to provide high availability to the OpenStack cloud controller services.

» Resources, Resource Groups, Resource Types, and the Resource Group Manager

The Resource Group Manager (RGM) subsystem manages applications put under Oracle Solaris Cluster control and the resources that they require to function. More specifically, the RGM ensures that cluster applications have the resources they need on the cluster nodes they are running. A cluster application is called a data service, often abbreviated to just a service.

The RGM uses specific agents to manage each data service. An RGM-managed resource is an instance of a resource type that is defined cluster wide. Several resource types are pre-defined as either generic, such as a logical hostname, or application specific, such as HA for MySQL. RGM-managed resources are placed into groups, called resource groups, so that they can be managed as a unit. A resource group is migrated as a unit if a failover or switchover is initiated on the resource group.

To allow orchestration of multiple resource groups and resources within the cluster, the RGM provides a rich set of resource group affinities and resource dependencies types. They are used to specify the various relations that exist between application resources, even distributed across global-cluster and zone-cluster nodes.

Optionally one can configure a set of load limits for each cluster node and assign load factors to resource groups. When starting a resource group, the RGM chooses a node from the resource group's preferred node list that best satisfies the configured load distribution policy.

» Global Cluster and Zone Clusters

In a cluster that runs on the Oracle Solaris Cluster framework, two types of clusters can exist: a global cluster that exists by default, and zone clusters which can be added. A global cluster consists of the set of Oracle Solaris global zones, while a zone cluster consists of a set of non-global zones (one per global-cluster node) that are configured to behave as a separate “virtual” cluster.

The ability to run multiple zone clusters within a global cluster, and the rich set of resource group affinities and resource dependencies provided by the RGM, enables deployment of multi-tiered applications, and securely separates application layers from each other. The ability to automatically orchestrate application components to orderly start and stop during normal operation, as well as during reconfigurations triggered by failures, is preserved with zone clusters. Further, the ability of Oracle Solaris zones to configure resource management, such as to specifically assign system resources like CPU and memory, can be leveraged with zone clusters.

» Failover, Scalable, and Multi-Master Applications

A cluster application that runs on only one node at a time is a failover application. A cluster application that runs an instance on multiple nodes at a time is either a multi-master or scalable application. The later takes advantage of Oracle Solaris Cluster built-in network load balancing.

» Failover and Scalable IP Addresses

The Oracle Solaris Cluster software provides two mechanisms through which client applications can access services on the cluster: logical IP addresses (also called logical hostnames, abbreviated as LH) and global IP addresses (also called shared addresses or scalable addresses, abbreviated as SA).

Logical IP addresses are used by failover services. Global IP addresses are used for scalable services that can run on more than one node concurrently. The cluster framework provides a load balancing feature by the global networking service for scalable outbound IP traffic, to distribute TCP and/or UDP client requests across the
configured nodes for a scalable service. The load balancer can be configured for the packet distribution mechanism and load weights to be used.

» Failover and Scalable Mount Points

The SUNW.HAStoragePlus resource type is designed to make local and global file system configurations highly available. You can also use the SUNW.HAStoragePlus resource type to synchronize the startup of resources and device groups on which the resources depend.

The SUNW.ScalMountPoint resource type represents a scalable file-system mount point, where the file-system type can be a QFS shared file system or a network-attached storage (NAS) device, such as an NFS share from an Oracle ZFS Storage Appliance.

» Data Service for MySQL

Oracle Solaris Cluster provides a standard data service to manage a MySQL database, which implements a mechanism for orderly startup and shutdown, fault monitoring, and automatic failover of the MySQL service.

» Data Service for Apache

Oracle Solaris Cluster provides a standard data service to manage the Apache web server as a failover or scalable service for http and https access, which implements a mechanism for orderly startup and shutdown, fault monitoring, and automatic failover.

» Data Service for Oracle Solaris Zones

This standard data service provides high availability for Oracle Solaris Zones through three components in a failover or multi-master configuration:

sczbt: The orderly booting, shutdown, and fault monitoring of an Oracle Solaris zone.

sczsh: The orderly startup, shutdown, and fault monitoring of an application within the Oracle Solaris zone (managed by sczbt), using scripts or commands.

sczsmf: The orderly startup, shutdown, and fault monitoring of an Oracle Solaris Service Management Facility (SMF) service within the Oracle Solaris zone managed by sczbt.

With Oracle Solaris Cluster 4.2, the sczbt component supports cold migration (boot and shutdown) of solaris and solaris10 branded zones, and cold and warm migration for kernel zones on Oracle Solaris 11.2. Using warm migration (suspend and resume of a kernel zone) can minimize planned downtime, for example, in case a cluster node is overloaded or needs to be shut down for maintenance.

» Data Service for HA SMF Proxy Resources

Oracle Solaris Cluster includes three SMF proxy resource types that can be used to enable SMF services to run with Oracle Solaris Cluster in a failover, multi-master, or scalable configuration. The SMF proxy resource types enables you to encapsulate a set of interrelated SMF services into a single resource, the SMF proxy resource, to be managed by Oracle Solaris Cluster. In this feature, SMF manages the availability of SMF services on a single node. Oracle Solaris Cluster extends this capability to multiple nodes and helps to provide high availability and scalability of the SMF services.

» Generic Data Service (GDSv2)

The Generic Data Service is a mechanism for making simple network-aware and non-network-aware applications highly available by plugging them into the Oracle Solaris Cluster Resource Group Manager (RGM) framework. This mechanism does not require coding a data service, which you typically must do to make an application highly available. It can be used for cases where no standard data service is part of Oracle Solaris Cluster exists for a given application.

More information about the Oracle Solaris Cluster capabilities can be found within the Oracle Solaris Cluster Concepts Guide [2] and within the Oracle Solaris Cluster 4.2 documentation library [6].
Overview of the High Availability Cloud Controller Configuration

Figure 2 below reflects the high-level setup of the HA OpenStack cloud controller. In this example, the services running on the OpenStack cloud controller are organized within four runtime environments:

- Dedicated zone cluster to host the MySQL database (named `os-db`), used by all OpenStack components
- Dedicated zone cluster to host the central messaging service RabbitMQ (named `os-mq`), used by all OpenStack components
- Dedicated zone cluster for all OpenStack services (named `os-api`) that can be run within a non-global Solaris zone
- Global cluster to host the Elastic Virtual Switch (EVS) controller within a failover zone and the Neutron agent services

This is an example on how to achieve secure isolation between services, and concurrently define all the required dependencies for proper startup and stopping between services which is orchestrated by the cluster framework.

The granularity of service separation can be further increased, specifically within the OpenStack services hosted by the `os-api` zone cluster. Additional zone clusters can be created and also distributed across multiple physical cluster nodes, depending on runtime performance requirements for individual OpenStack services. Only services that are required to run within the global zone context are configured within the global cluster.

The OpenStack cloud controller provides various API endpoints. From an initial setup perspective, high availability needs to be considered already at installation time.

Oracle Solaris Cluster offers built-in HA data services for several OpenStack components (like MySQL, Apache), which are used in this implementation.

Most OpenStack components are managed by one or more corresponding SMF services on Oracle Solaris. If Oracle Solaris Cluster does not offer the corresponding HA data service, for most of those SMF services the data service for HA SMF Proxy resources is used to provide HA for those components.

The generic approach to provide HA for OpenStack SMF services can be summarized as follows:

- **Failover services (stateful active/passive)**
  - Configure either a SUNW.HAStoragePlus or SUNW.ScalMountPoint cluster resource to store dynamic file system content
  - Configure a SUNW.LogicalHostname resource for the service endpoint(s)
  - Configure a SUNW.Proxy_SMF_failover resource for each SMF service

- **Scalable service (stateless active/active)**
  - Ensure static content is replicated identical across all cluster nodes or zones
  - Configure a failover resource group with SUNW.SharedAddress resource for the service endpoint(s)
  - Configure a scalable resource group with a SUNW.Proxy_SMF_scalable resource for each SMF service

The cluster load balancer is used to distribute client access between the scalable SMF services.

Other OpenStack components that need to access the highly available OpenStack endpoints have to be configured to use the logical hostname that is managed by the cluster.
Figure 2: HA OpenStack cloud controller
When customizing the various OpenStack components' configuration files for a failover service, directories and file system paths that point to `/var/lib/<component>/` hosting dynamic data have to be changed to point to a directory on shared storage that is managed by the cluster. IP addresses for the various services have to be defined and configured to use the logical or scalable cluster IP in those configuration files as well. This is done to make sure that the failover service is switched together with the storage data and service IP-address.

The following Table 1 outlines the example mapping of cluster-managed IP addresses to the OpenStack services:

<table>
<thead>
<tr>
<th>Logical Hostname</th>
<th>network port</th>
<th>OpenStack service</th>
<th>runs in</th>
</tr>
</thead>
<tbody>
<tr>
<td>os-db-lh</td>
<td>3306/tcp</td>
<td>MySQL database</td>
<td>zone cluster os-db</td>
</tr>
<tr>
<td>os-mq-sa (SA)</td>
<td>5672/tcp</td>
<td>RabbitMQ</td>
<td>zone cluster os-mq</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>5000/tcp</td>
<td>keystone AUTH_URL</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>35357/tcp</td>
<td>keystone SERVICE_ENDPOINT identity</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>8776/tcp</td>
<td>cinder-api</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>9292/tcp</td>
<td>glance-api</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>8773/tcp</td>
<td>nova-api-ec2</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>8080/tcp</td>
<td>swift-proxy-server</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>9696/tcp</td>
<td>neutron-server</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>8774/tcp</td>
<td>nova-api-osapi-compute</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>9191/tcp</td>
<td>glance-registry</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-lh</td>
<td>3333/tcp</td>
<td>nova-objectstore</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-sa (SA)</td>
<td>80/tcp</td>
<td>horizon dashboard (apache)</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-api-sa (SA)</td>
<td>443/tcp</td>
<td>horizon dashboard (apache)</td>
<td>zone cluster os-api</td>
</tr>
<tr>
<td>os-evscon-lh</td>
<td>22/tcp</td>
<td>EVS controller</td>
<td>failover solaris zone</td>
</tr>
<tr>
<td></td>
<td>67/tcp</td>
<td>neutron-dhcp-agent</td>
<td>Global Zone</td>
</tr>
<tr>
<td></td>
<td>53/tcp</td>
<td>neutron-dhcp-agent</td>
<td>Global Zone</td>
</tr>
</tbody>
</table>

Table 1: Cluster IP address and port mapping to OpenStack services

When configuring Keystone, endpoints for the various OpenStack service components are defined. To change a specific standard OpenStack service into a highly available service with Oracle Solaris Cluster, the Keystone endpoint needs to be configured to use the highly available IP address managed by the cluster framework.

**Oracle Solaris Cluster Framework and Infrastructure Setup**

Instructions in this white paper provide details on how to set up the OpenStack components under Oracle Solaris Cluster control. As a prerequisite, the Oracle Solaris Cluster Image Packaging System (IPS) packages need to be installed on the Oracle Solaris systems, which should form the cluster, and the Oracle Solaris Cluster framework has to be configured. The Oracle Technology Network provides technical resources for Oracle Solaris Cluster [7], specifically the article "How to Install and Configure a Two-Node Cluster" [8] and the Oracle Solaris Cluster Software Installation Guide [9].
**Configuration assumptions**

This white paper assumes that the following configuration is used for HA OpenStack cloud controller cluster nodes:

» At least two physical systems are configured as a global cluster with Oracle Solaris 11.2 SRU 7 and Oracle Solaris Cluster 4.2 SRU 3 or newer. You can install more nodes in the cluster as required. In this example the global cluster node names are cc-node-a and cc-node-b.

» The Image Packaging System publishers for Oracle Solaris (solaris) and Oracle Solaris Cluster (ha-cluster) are already configured for the corresponding support repositories on the cluster nodes.

» Oracle Solaris Cluster 4.2 SRU 3 or later is installed and configured with the ha-cluster-full group package.

» This example uses the OpenStack Havana release, which is delivered by Oracle Solaris 11.2.

» Each node has two spare network interfaces to be used as private interconnects, also known as transports, and at least two network interfaces configured as IPMP group sc_ipmp0, which is connected to the public cluster network and is also used for the OpenStack service endpoints. Further network adapters (protected by either IPMP or IP aggregation) can be configured as a dedicated network between the cloud controller and nova compute nodes that are managed by Neutron and the EVS. In this example, an aggregation aggr1 is used to configure a tagged VLAN for that purpose.

» The IP addresses used for the cluster nodes, zone cluster nodes, logical hostnames, and shared addresses resolve properly on all nodes and zones to the host aliases used in this example:

  cc-node-a, cc-node-b, os-db-z1, os-db-z2, os-db-lh, os-mq-z1, os-mq-z2, os-mq-sa, os-api-z1, os-api-z2, os-api-lh, os-api-sa, os-evscon-lh.

» An Oracle ZFS Storage Appliance is configured on the cluster to provide shared storage. For more information, see "How to Install an Oracle ZFS Storage Appliance in a Cluster." [10].

» The cluster hardware is a supported configuration for Oracle Solaris Cluster 4.2 software. For more information, see the Oracle Solaris Cluster 4 Compatibility Guide [3].

» The global cluster nodes are configured as NTP clients, to synchronize time from a common source [9].

» The path /usr/cluster/bin has been added to the PATH variable defined within the root user .profile on all cluster nodes and zone cluster nodes.

» Throughout the examples, the short name for the cluster commands are used, for example, clrs instead of clresource. A list of cluster commands and their corresponding short names can be viewed by executing cluster list-cmds -v.

**Conventions on the command output:**

» When a shell prompt is shown as:

  cc-node-a# command

  execute command on only the specific node shown, cc-node-a.

» When a shell prompt is shown as:

  all global nodes# command

  execute command on all global cluster nodes.
When the shell prompt is shown as:

```
both os-db zones# command
```

execute command on both the zone cluster nodes as shown, in this example os-db-z1 and os-db-z2.

### Configure zpools on shared storage and a dataset per node for later usage

Configure a zpool named `os-db_zp` with mount point `/failover/os-db` to host the OpenStack MySQL database:

```
cc-node-a# zpool create -m /failover/os-db os-db_zp <shared storage device 1 cxtydz>
cc-node-a# zpool export os-db_zp
```

Configure a zpool named `os-api_zp` with mount point `/failover/os-api` to host the data for OpenStack services configured as failover services:

```
cc-node-a# zpool create -m /failover/os-api os-api_zp <shared storage device 2 cxtydz>
cc-node-a# zpool export os-api_zp
```

Configure a zpool named `os-gz-api_zp` with mount point `/failover/os-gz-api` to host the data for the `neutron-l3-agent` and `neutron-dhcp-agent` services:

```
cc-node-a# zpool create -m /failover/os-gz-api os-gz-api_zp <shared storage device 3 cxtydz>
cc-node-a# zpool export os-gz-api_zp
```

Configure a dedicated dataset with mount point `/zones` for the zone cluster node zonepath within the rpool on all global cluster nodes:

```
all-global nodes# zfs create -o mountpoint=/zones rpool/zones
```

### Zone cluster installation

Create zone cluster `os-db`, which will host the HA MySQL configuration. The zone cluster nodes are named `os-db-z1` and `os-db-z2`. The zone cluster is allowed to use the logical hostname `os-db-lh` and zpool `os_db_zp`:

```
cc-node-a# vi /var/tmp/os-db.txt
create
set brand=solaris
set zonepath=/zones/os-db
set ip-type=shared
set enable_priv_net=true
set autoboot=true
add node
set physical-host=cc-node-a
set hostname=os-db-z1
add net
set address=os-db-z1
set physical=sc_ipmp0
end
end
add node
set physical-host=cc-node-b
set hostname=os-db-z2
add net
```
Create zone cluster os-mq, which will host the HA RabbitMQ configuration. The zone cluster nodes are named os-mq-z1 and os-mq-z2. The shared address os-mq-sa will be used for this zone cluster:

```
cc-node-a# vi /var/tmp/os-mq.txt
create
set brand=solaris
set zonepath=/zones/os-mq
set ip-type=shared
set enable_priv_net=true
set autoboot=true
add node
set physical-host=cc-node-a
set hostname=os-mq-z1
add net
set address=os-mq-z1
set physical=sc_ipmp0
end
end
add node
set physical-host=cc-node-b
set hostname=os-mq-z2
add net
set address=os-mq-z2
set physical=sc_ipmp0
end
end
add net
set address=os-mq-sa
end
commit
exit
```

```
cc-node-a# clzc configure -f /var/tmp/os-mq.txt os-mq
```

## Note: You can re-use the sysconfig profile used to setup zone cluster os-db
cc-node-a# clzc install -c /var/tmp/os-db.xml/sc_profile.xml os-mq

cinode-a# clzc boot os-mq

=> on both nodes: zlogin -C -e @ os-mq
    and verify the zone boots correctly on each node

Create zone cluster os-api, which will host the OpenStack services for the cloud controller configuration. The zone cluster nodes are named os-api-z1 and os-api-z2. The zone cluster is allowed to use the logical hostname os-api-lh, the shared address os-api-sa, and zpool os_api_zp:

cc-node-a# vi /var/tmp/os-api.txt
create
set brand=solaris
set zonopath=/zones/os-api
set ip-type=shared
set enable_priv_net=true
set autoboot=true
add node
set physical-host=cc-node-a
set hostname=os-api-z1
add net
set address=os-api-z1
set physical=sc_ipmp0
end
end
add node
set physical-host=cc-node-b
set hostname=os-api-z2
add net
set address=os-api-z2
set physical=sc_ipmp0
end
end
add net
set address=os-api-lh
end
add net
set address=os-api-sa
end
add dataset
set name=os-api_zp
end
commit
exit

cc-node-a# clzc configure -f /var/tmp/os-api.txt os-api

## Note: You can re-use the sysconfig profile used to setup zone cluster os-db
cc-node-a# clzc install -c /var/tmp/os-db.xml/sc_profile.xml os-api

cc-node-a# clzc boot os-api

=> on both nodes: zlogin -C -e @ os-api
    and verify the zone boots correctly on each node
OpenStack Components on the HA Cloud Controller

Database - MySQL

Within a multi-node configuration of OpenStack on Oracle Solaris, the backend database is typically MySQL. The following OpenStack components store data within the MySQL database and are thus dependent on it:

» Keystone

» Glance (glance-db)

» Nova (nova-conductor, nova-scheduler, nova-cert, nova-api-osapi-compute)

» Neutron (neutron-server)

» Cinder (cinder-db)

» Heat (heat-db)

Providing HA for the MySQL database is critical for these OpenStack services to operate properly.

The HA MySQL data service is used to provide HA for the MySQL database [14]. This data service provides a variety of supported MySQL topologies. This example shows on how to configure MySQL as a failover database. Based on requirements to distribute load for the database, other supported topologies may be used.

Configure cluster infrastructure objects for HA MySQL within zone cluster os-db

Configure the resource group os-db-rg, logical hostname os-db-lh-rs, and HA zpool os-db-hasp-rs resources within zone cluster os-db for HA MySQL:

```
os-db-z1# clrt register SUNW.HAStoragePlus
os-db-z1# clrg create os-db-rg
os-db-z1# clrslh create -g os-db-rg -h os-db-lh os-db-lh-rs
both os-db zones # mkdir -p /failover/os-db
os-db-z1# clrs create -t SUNW.HAStoragePlus -g os-db-rg -p zpools=os-db_zp \ os-db-hasp-rs
os-db-z1# clrg online -eM os-db-rg
```

Install and configure the MySQL database within zone cluster os-db

Install the MySQL binaries and set up the database by using file system /failover/os-db on the HA zpool os-db_zp and the IP-address os-db-lh under cluster control:

```
both os-db zones# pkg install mysql-55 mysql-55/client
cc-node-a# zfs create os-db_zp/mysql
os-db-z1# mkdir /failover/os-db/mysql/logs
os-db-z1# mkdir /failover/os-db/mysql/innodb
os-db-z1# cp /etc/mysql/5.5/my.cnf /failover/os-db/mysql/
os-db-z1# vi /failover/os-db/mysql/my.cnf
--- my.cnf.orig Thu Jul  3 11:02:56 2014
+++ my.cnf Thu Jul  3 11:07:24 2014
@@ -18,7 +18,7 @@
```

[client]
#password = your_password
port = 3306
-socket = /tmp/mysql.sock
+socket = /tmp/os-db-lh.sock

# Here follows entries for some specific programs

@@ -25,7 +25,7 @@
# The MySQL server
[mysqld]
port = 3306
-socket = /tmp/mysql.sock
+socket = /tmp/os-db-lh.sock
 skip-external-locking
 key_buffer_size = 16K
 max_allowed_packet = 1M
@@ -45,6 +45,8 @@
 #skip-networking
 server-id = 1

+bind-address=os-db-lh
 +
 # Uncomment the following if you want to log updates
 #log-bin=mysql-bin
@@ -59,18 +61,18 @@
 #binlog_direct_non_transactional_updates=TRUE

 # Uncomment the following if you are using InnoDB tables
-#innodb data_home_dir = /var/mysql/5.5
-#innodb data_file_path = ibdata1:10M:autoextend
-#innodb log_group_home_dir = /var/mysql/5.5
+innodb data_home_dir = /failover/os-db/mysql/innodb
 +innodb data_file_path = ibdata1:10M:autoextend
 +innodb log_group_home_dir = /failover/os-db/mysql/innodb

 # You can set .._buffer_pool_size up to 50 ~ 80 %
 # if RAM but beware of setting memory usage too high
-#innodb buffer_pool_size = 16M
-#innodb additional_mem_pool_size = 2M
+innodb buffer_pool_size = 16M
 +innodb additional_mem_pool_size = 2M

 # Set .._log_file_size to 25 % of buffer pool size
-#innodb log_file_size = 5M
-#innodb log_buffer_size = 8M
-#innodb flush_log_at_trx_commit = 1
-#innodb lock_wait_timeout = 50
+innodb log_file_size = 5M
 +innodb log_buffer_size = 8M
 +innodb flush_log_at_trx_commit = 1
 +innodb lock_wait_timeout = 50

 [mysqldump]
 quick

 both os-db zones
 cd /etc/mysql/5.5
 both os-db zones
 mv my.cnf my.cnf.orig
 both os-db zones
 ln -s /failover/os-db/mysql/my.cnf .
os-db-z1# chown -R mysql:mysql /failover/os-db/mysql
Installing MySQL system tables...  
OK

Filling help tables...  
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER!
To do so, start the server, then issue the following commands:

/usr/mysql/5.5/bin/mysqld --defaults-file=/failover/os-db/mysql/my.cnf
--user=mysql --basedir=/usr/mysql/5.5
--datadir=/failover/os-db/mysql --pid-file=/failover/os-db/mysql/mysql.pid

Alternatively you can run:
/usr/mysql/5.5/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:

cd /usr/mysql/5.5 ; /usr/mysql/5.5/bin/mysqld_safe

You can test the MySQL daemon with mysql-test-run.pl

cd /usr/mysql/5.5/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /usr/mysql/5.5/scripts/mysqlbug script!

InnoDB: The first specified data file /failover/os-db/mysql/innodb/ibdata1 did not
exist:
InnoDB: Setting file /failover/os-db/mysql/innodb/ibdata1 size to
10 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Setting log file /failover/os-db/mysql/innodb/ib_logfile0 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Setting log file /failover/os-db/mysql/innodb/ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
InnoDB: Waiting for the background threads to start
InnoDB: 5.5.31 started; log sequence number 0
[Note] Server hostname (bind-address): 'os-db-lh'; port: 3306
[Note] - 'os-db-lh' resolves to '10.134.96.132';
[Note] Server socket created on IP: '10.134.96.132'.
[Note] Event Scheduler: Loaded 0 events
Version: '5.5.31' socket: '/tmp/os-db-lh.sock' port: 3306 MySQL Community Server (GPL)

The following is only needed for the mysql_secure_installation script:

```
os-db-z1# cd /tmp
os-db-z1# ln os-db-lh.sock mysql.sock
os-db-z1# export PATH=${PATH}:/usr/mysql/5.5/bin
os-db-z1# /usr/mysql/5.5/bin/mysql_secure_installation
```

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current password for the root user. If you've just installed MySQL, and you haven't set the root password yet, the password will be blank, so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL root user without the proper authorisation.

Set root password? [Y/n] y
New password: <mysqlrootpassword>
Re-enter new password: <mysqlrootpassword>
Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n] Y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? [Y/n] Y

... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MySQL installation should now be secure.

Thanks for using MySQL!

os-db-z1# /usr/mysql/5.5/bin/mysql -S /tmp/os-db-lh.sock -uroot -p'<mysqlrootpassword>'

mysql> use mysql;
Database changed

mysql> GRANT ALL ON *.* TO 'root'@'os-db-z1' IDENTIFIED BY '<mysqlrootpassword>';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON *.* TO 'root'@'os-db-z2' IDENTIFIED BY '<mysqlrootpassword>';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE user SET Grant_priv='Y' WHERE User='root' AND Host='os-db-z1';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> UPDATE user SET Grant_priv='Y' WHERE User='root' AND Host='os-db-z2';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> GRANT ALL ON *.* TO 'root'@'os-db-lh' IDENTIFIED BY '<mysqlrootpassword>';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE user SET Grant_priv='Y' WHERE User='root' AND Host='os-db-lh';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

Configure the MySQL database as HA MySQL failover cluster resource os-mysql-rs within zone cluster os-db

Configure the MySQL database instance as cluster resource os-mysql-rs within resource group os-db-rg:

os-db-z1# mkdir /failover/os-db/cluster-config
os-db-z1# cd /failover/os-db/cluster-config
os-db-z1# cp /opt/SUNWscmys/util/ha_mysql_config .
os-db-z1# cp /opt/SUNWscmys/util/mysql_config .

os-db-z1# vi ha_mysql_config
RS=os-mysql-rs
RG=os-db-rg
PORT=3306
LH=os-db-lh-rs
SCALABLE=
LB_POLICY=
RS_PROP=
HAS_RS=os-db-hasp-rs
ZONE=
ZONE_BT=
PROJECT=
BASEDIR=/usr/mysql/5.5
DATADIR=/failover/os-db/mysql
MYSQLUSER=mysql
MYSQLHOST=os-db-lh
FMUSER=fmuser
FMPASS=
LOGDIR=/failover/os-db/mysql/logs
CHECK=YES
NDB_CHECK=

os-db-z1# vi mysql_config
MySQL_BASE=/usr/mysql/5.5
MYSQL_USER=root
MYSQL_PASSWD=
MYSQL_HOST=os-db-lh
FMUSER=fmuser
FMPASS=
MYSQL_SOCK=/tmp/os-db-lh.sock
MYSQL_NIC_HOSTNAME="os-db-z1 os-db-z2 os-db-lh"
MYSQL_DATADIR=/failover/os-db/mysql
NDB_CHECK=

os-db-z1# /opt/SUNWscmys/util/mysql_register \
    -f /failover/os-db/cluster-config/mysql_config
    sourcing /failover/os-db/cluster-config/mysql_config.
MySQL version 5 detected on 5.11.
Enter the MySQL admin password. => <mysqlrootpassword>
Re enter the MySQL admin password.
Check if the MySQL server is running and accepting connections.
Enter the MySQL fault monitor user password. => <fmuserpassword>
Reenter the MySQL fault monitor user password.
Add faultmonitor user (fmuser) with its password with Process, Select, Reload, and Shutdown privileges to user table for MySQL database for host os-db-z1.
Add SUPER privilege for fmuser@os-db-z1.
Add faultmonitor user (fmuser) with its password with Process, Select, Reload, and Shutdown privileges to user table for MySQL database for host os-db-z2.
Add SUPER privilege for fmuser@os-db-z2.
Add faultmonitor user (fmuser) with its password with Process, Select, Reload, and Shutdown privileges to user table for MySQL database for host os-db-lh.
Add SUPER privilege for fmuser@os-db-lh.
Create test-database sc3_test_database.
Grant all privileges to sc3_test_database for faultmonitor-user fmuser for host os-db-z1.
Grant all privileges to sc3_test_database for faultmonitor-user fmuser for host os-db-z2.
Grant all privileges to sc3_test_database for faultmonitor-user fmuser for host os-db-1h.

Flush all privileges.

MySQL configuration for HA is done.

```
    os-db-z1# kill -TERM $(cat /failover/os-db/mysql/mysql.pid)
```

```
    os-db-z1# /opt/SUNWscmys/util/ha_mysql_register 
           -f /failover/os-db/cluster-config/ha_mysql_config -e
```

Sourcing /failover/os-db/cluster-config/ha_mysql_config and create a working copy under /opt/SUNWscmys/util/ha_mysql_config.work.

remove the working copy /opt/SUNWscmys/util/ha_mysql_config.work.

Enter the password for the faultmonitoring user. => <fmuserpassword>

Re enter the password for the faultmonitoring user.

Encrypt the password.

```
    os-db-z1# cd
    os-db-z1# clrg switch -n os-db-z2 os-db-rg
```

```
    os-db-z2# /opt/SUNWscmys/util/ha_mysql_register 
           -f /failover/os-db/cluster-config/ha_mysql_config -e
```

Sourcing /failover/os-db/cluster-config/ha_mysql_config and create a working copy under /opt/SUNWscmys/util/ha_mysql_config.work.

remove the working copy /opt/SUNWscmys/util/ha_mysql_config.work.

Enter the password for the faultmonitoring user.

Re enter the password for the faultmonitoring user. => <fmuserpassword>

Encrypt the password.

```
    os-db-z1# clrg switch -n os-db-z1 os-db-rg
```

```
    os-db-z1# clrt register SUNW.gds
    os-db-z1# /opt/SUNWscmys/util/ha_mysql_register 
           -f /failover/os-db/cluster-config/ha_mysql_config
```

Sourcing /failover/os-db/cluster-config/ha_mysql_config and create a working copy under /opt/SUNWscmys/util/ha_mysql_config.work.

Registration of resource os-mysql-rs succeeded.

remove the working copy /opt/SUNWscmys/util/ha_mysql_config.work.

```
    os-db-z1# clrs enable os-mysql-rs
```

Once MySQL is started, under cluster control enable and disable the MySQL service by enabling and disabling the os-mysql-rs cluster resource through the Oracle Solaris Cluster CLI or BUI.

Example to enable the MySQL service through the CLI:

```
    os-db-z1# clrs enable os-mysql-rs
```

Example to disable the MySQL service through the CLI:

```
    os-db-z1# clrs disable os-mysql-rs
```

More information about the HA MySQL data service can be found within the HA for MySQL data service guide [14].
Message Queue - RabbitMQ

Oracle Solaris provides RabbitMQ as message queue software for the OpenStack configuration. The following OpenStack components make use of messages queues stored and managed by the rabbitmq SMF service and are thus dependent on it:

» Glance (glance-api)

» Nova (nova-conductor, nova-scheduler, nova-cert, nova-api-osapi-compute)

» Neutron (neutron-server, neutron-l3-agent, neutron-dhcp-agent)

» Cinder (cinder-scheduler, cinder-api)

Providing high availability for the rabbitmq service is critical for those OpenStack components to operate properly. There are also OpenStack services running outside the HA cloud controller, which make use of the rabbitmq service (such as nova-compute).

The rabbitmq SMF service can be made HA by using its own capability to cluster rabbitmq instances [17], and to then configure mirrored messages queues [18]. That way, each instance has knowledge and content for each queue. Load balancing is delivered through the usage of the SUNW.SharedAddress resource type (provided by the Oracle Solaris Cluster built-in load balancer). Thus, all rabbitmq clients only need to be configured for one IP address (os-mq-sa); HA is transparent for them. The rabbitmq instances can be started through the SUNW.Proxy_SMF_scalable resource type as a scalable service.

Configure cluster infrastructure objects for HA rabbitmq within zone cluster os-mq

Configure the failover resource group os-mq-sa-rg for the shared address resource os-mq-sa-rs within zone cluster os-mq, which will be used by the highly available rabbitmq service later on:

```
os-mq-z1# clrg create os-mq-sa-rg
os-mq-z1# clrssa create -g os-mq-sa-rg -h os-mq-sa os-mq-sa-rs
os-mq-z1# clrg online -eM os-mq-sa-rg
```

Configure rabbitmq using the cluster option with mirrored queues within zone cluster os-mq

Install the rabbitmq binaries and create an Erlang cookie, which can be used by both rabbitmq instances, and configure rabbitmq with the cluster option and mirrored queues on both zone cluster zones os-mq-z1 and os-mq-z2. The rabbitmq service will make use of the scalable IP-address os-mq-sa which is managed by the os-mq-sa-rs cluster resource:

```
both os-mq zones# pkg install rabbitmq
## Just create an erlang cookie that can be used by both instances by enabling
## and disabling the rabbitmq SMF service on one node, then copy that cookie to
## the other node.
os-mq-z1# svcadm enable rabbitmq
os-mq-z1# svcadm disable rabbitmq
os-mq-z1# scp /var/lib/rabbitmq/.erlang.cookie os-mq-z2:/var/lib/rabbitmq/
os-mq-z2# chown rabbitmq:bin /var/lib/rabbitmq/.erlang.cookie
os-mq-z2# chmod 400 /var/lib/rabbitmq/.erlang.cookie
os-mq-z1# vi /etc/rabbitmq/rabbitmq-env.conf
NODENAME=rabbit@os-mq-z1
NODE_IP_ADDRESS=os-mq-sa
CONFIG_FILE=/etc/rabbitmq/rabbitmq
```
Configure rabbitmq as scalable cluster resources within zone cluster os-mq

Create the scalable cluster resource group rabbitmq-rg with resource os-rabbitmq-rs, using the SUNW.Proxy_SMF_scalable resource type to manage rabbitmq as a scalable service. The cluster load balancer is configured to use the shared address os-mq-sa with TCP port 5672, with an even client access distribution between the rabbitmq instances:

```
os-mq-z1# clrg create -S rabbitmq-rg
both os-mq zones# mkdir -p /opt/HA-OpenStack/etc
both os-mq zones# vi /opt/HA-OpenStack/etc/rabbitmq-svclist.txt
<svc:/application/rabbitmq:default>,<lib/svc/manifest/application/rabbitmq.xml>
os-mq-z1# clrt register SUNW.Proxy_SMF_scalable
```

```
os-mq-z1# clrs create -d -g rabbitmq-rg -t SUNW.Proxy_SMF_scalable
  -p proxied_service_instances=/opt/HA-OpenStack/etc/rabbitmq-svclist.txt
  -p Resource_dependencies=os-mq-sa-rs -p Port_list="5672/tcp"
  -p Load_balancing_policy="Lb_weighted"
  -p Load_balancing_weights="1@1,1@2" os-rabbitmq-rs
```
Once `rabbitmq` is started under cluster control, enable and disable the `rabbitmq` service by enabling and disabling the `os-rabbitmq-rs` cluster resource through the Oracle Solaris Cluster CLI or BUI.

Configure cluster infrastructure objects for OpenStack components running within zone cluster `os-api`

Configure the resource group `os-api-rg` with the logical hostname resource `os-api-lh-rs` and HA zpool resource `os-api-hasp-rs`, and the resource group `os-api-sa-rg` with the shared address resource `os-api-sa-rs` for all OpenStack components running within zone cluster `os-api`:

```
os-api-z1# clrg create os-api-rg
os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

```os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

```
os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

```os-api-z1# clrg create os-api-rg
os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

```
os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

```
os-api-z1# clrg create os-api-rg
os-api-z1# clrs enable os-rabbitmq-rs
os-api-z1# clrg online -eM rabbitmq-rg
```

Keystone

The Keystone identity service provides authentication and authorization services between users, administrators, and OpenStack services. Keystone is required to be available for all OpenStack components, therefore high availability for this service is essential.

The following SMF service will be managed by the HA SMF proxy failover resource:

```sVC:/application/openstack/keystone:default```

In order for Keystone to use the configured cluster objects, the following configuration files and options need to be amended:

```
<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/keystone/keystone.conf</td>
<td>admin_token</td>
<td>ADMIN</td>
</tr>
<tr>
<td></td>
<td>bind_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>log_dir</td>
<td>/failover/os-api/keystone/log</td>
</tr>
<tr>
<td></td>
<td>connection</td>
<td>mysql://keystone:&lt;keystonepassword&gt;@os-db-lh/keystone</td>
</tr>
</tbody>
</table>
```

Table 2: Keystone configuration parameters

In addition, other parameters not related to the HA setup may be amended as required [1].

The communication endpoints for the various OpenStack services configured within Keystone need to use the highly available IP addresses managed by the cluster for those services. In this example, the endpoints use the IP address that corresponds to `os-api-lh`.
Create Keystone database within zone cluster os-db

Create the Keystone database within MySQL and ensure that the Keystone user has access from all nodes and zones that are required to connect to the Keystone database. Access to the Keystone database needs to be directed to the logical hostname os-db-lh:

```
os-db-z1# mysql -h os-db-lh -u root -p
mysql> CREATE DATABASE keystone;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED by '<keystonepassword>';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '<keystonepassword>';
Query OK, 0 rows affected (0.00 sec)
```

Configure Keystone within zone cluster os-api

Install the Keystone packages within zone cluster os-api and amend the configuration file as described within the OpenStack installation guide [1] and Table 2:

```
both os-api zones# pkg install keystone mysql-55/client 
                   library/python-2/python-mysql-26 library/python-2/sqlalchemy-26
both os-api zones# vi /etc/keystone/keystone.conf
                 => amend configuration parameters as described in [1] and table 2

## Create directories used by keystone on shared storage:
both os-api zones# mkdir /failover/os-api/keystone
both os-api zones# mkdir /failover/os-api/keystone/log
both os-api zones# chown -R keystone:keystone /failover/os-api/keystone
## run db_sync for keystone on os-api zone cluster
both os-api zones# su - keystone -c "keystone-manage db_sync"
both os-api zones# svcadm enable keystone
both os-api zones# su - keystone -c "keystone-manage pki_setup"
## Sync the ssl config across zone cluster nodes:
both os-api zones# cd /etc/keystone
both os-api zones# cpf ssl.tar ssl ec2rc
both os-api zones# scp os-api-z1:/etc/keystone/ssl.tar .
both os-api zones# tar xpf ssl.tar
```

The Keystone package provides the sample script /usr/demo/openstack/keystone/sample_data.sh to populate the Keystone database with an administrative user using the SERVICE_TOKEN and SERVICE_ENDPOINT environment variables, the various OpenStack component users with their password, and the corresponding communication endpoints. Regardless of whether you set up the endpoints using the sample script or do it manually, you have to ensure that the logical hostnames that are configured with the corresponding cluster resources are used, as summarized in Table 1.
For this example, the OS_AUTH_URL variable needs to be defined as http://os-api-lh:5000/v2.0 and CONTROLLER_PUBLIC_ADDRESS, CONTROLLER_ADMIN_ADDRESS, and CONTROLLER_INTERNAL_ADDRESS need to be set to os-api-lh.

You can verify the configured Keystone endpoints:

```
os-api-z1# su - keystone -c "env SERVICE_ENDPOINT=http://os-api-lh:35357/v2.0 \ SERVICE_TOKEN=ADMIN keystone endpoint-list"
```

Disable the manually started Keystone SMF service:

```
os-api-z1# svcadm disable keystone
```

Configure the Keystone SMF service as a failover cluster resource within zone cluster os-api

Configure the Keystone SMF service as SUNW.Proxy_SMF_failover resource os-keystone-rs within resource group os-api-rg, using the logical hostname os-api-lh and zpool os-api_zp. Within the global cluster, define a resource dependency on the HA MySQL resource os-mysql-rs:

```
both os-api zones# mkdir -p /opt/HA-OpenStack/etc
both os-api zones# vi /opt/HA-OpenStack/etc/keystone-svclist.txt
<svc:/application/openstack/keystone:default>,/lib/svc/manifest/application/openstack/keystone.xml>

os-api-z1# clrt register Proxy_SMF_failover
os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \ -p proxied_service_instances=/opt/HA-OpenStack/etc/keystone-svclist.txt \ -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \ os-keystone-rs

## also need a resource dependency on HA MySQL, configure in the global cluster:
cc-node-a# clrs set -Z os-api -p Resource_dependencies+=os-db:os-mysql-rs \ os-keystone-rs
```

Once Keystone is started under cluster control, enable and disable the Keystone service by enabling and disabling the os-keystone-rs cluster resource through the Oracle Solaris Cluster CLI or BUI.

---

**Glance**

The Glance image store service stores disk images of virtual machines (VM), which are used to deploy VM instances. In Oracle Solaris, Glance images are Unified Archives. The images can be stored in a variety of locations, from simple file systems to object storage systems such as OpenStack Swift. Glance has a RESTful API that enables you to query image metadata as well as retrieve the image.

In this configuration, the Unified Archives are stored on a ZFS file system. The corresponding zpool os-api_zp is managed by the SUNW.HAStoragePlus resource os-api-hasp-rs.

The following SMF services will be managed by corresponding HA SMF proxy failover resources:

- `svc:/application/openstack/glance/glance-api:default`
- `svc:/application/openstack/glance/glance-db:default`
- `svc:/application/openstack/glance/glance-registry:default`
In order for the Glance services to use the configured cluster objects, the following configuration files and options have to be amended in addition to options that you normally set up:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/glance/glance-api.conf</td>
<td>bind_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>log_file</td>
<td>/failover/os-api/glance/log/api.log</td>
</tr>
<tr>
<td></td>
<td>sql_connection</td>
<td>mysql://glance:&lt;glancepwd&gt;@os-db-lh/glance</td>
</tr>
<tr>
<td></td>
<td>registry_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>notifier_strategy</td>
<td>rabbit</td>
</tr>
<tr>
<td></td>
<td>rabbit_host</td>
<td>os-mq-sa</td>
</tr>
<tr>
<td></td>
<td>filesystem_store_datadir</td>
<td>/failover/os-api/glance/images/</td>
</tr>
<tr>
<td></td>
<td>swift_store_auth_address</td>
<td>os-api-lh:5000/v2.0/</td>
</tr>
<tr>
<td></td>
<td>s3_store_host</td>
<td>os-api-lh:8080/v1.0/</td>
</tr>
<tr>
<td></td>
<td>scrubber_datadir</td>
<td>/failover/os-api/glance/scrubber</td>
</tr>
<tr>
<td></td>
<td>image_cache_dir</td>
<td>/failover/os-api/glance/image-cache/</td>
</tr>
<tr>
<td></td>
<td>auth_url</td>
<td><a href="http://os-api-lh:5000/v2.0/">http://os-api-lh:5000/v2.0/</a></td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
<td><a href="http://os-api-lh:35357">http://os-api-lh:35357</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>signing_dir</td>
<td>/failover/os-api/glance/keystone-signing</td>
</tr>
<tr>
<td>/etc/glance/glance-cache.conf</td>
<td>log_file</td>
<td>/failover/os-api/glance/log/image-cache.log</td>
</tr>
<tr>
<td></td>
<td>image_cache_dir</td>
<td>/failover/os-api/glance/image-cache/</td>
</tr>
<tr>
<td></td>
<td>registry_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>auth_url</td>
<td><a href="http://os-api-lh:5000/v2.0/">http://os-api-lh:5000/v2.0/</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>filesystem_store_datadir</td>
<td>/failover/os-api/glance/images/</td>
</tr>
<tr>
<td></td>
<td>swift_store_auth_address</td>
<td>os-api-lh:5000/v2.0/</td>
</tr>
<tr>
<td></td>
<td>s3_store_host</td>
<td>os-api-lh:8080/v1.0/</td>
</tr>
<tr>
<td>/etc/glance/glance-registry.conf</td>
<td>bind_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>log_file</td>
<td>/failover/os-api/glance/log/registry.log</td>
</tr>
<tr>
<td></td>
<td>sql_connection</td>
<td>mysql://glance:&lt;glancepwd&gt;@os-db-lh/glance</td>
</tr>
<tr>
<td></td>
<td>auth_url</td>
<td><a href="http://os-api-lh:5000/v2.0/">http://os-api-lh:5000/v2.0/</a></td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
<td><a href="http://os-api-lh:35357">http://os-api-lh:35357</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>signing_dir</td>
<td>/failover/os-api/glance/keystone-signing</td>
<td></td>
</tr>
<tr>
<td>/etc/glance/glance-api-paste.ini</td>
<td>auth_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>auth_port</td>
<td>35357</td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
</tr>
<tr>
<td>/etc/glance/glance-registry-paste.ini</td>
<td>auth_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>auth_port</td>
<td>35357</td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
</tr>
<tr>
<td>/etc/glance/glance-scrubber.conf</td>
<td>log_file</td>
<td>/failover/os-api/glance/log/scrubber.log</td>
</tr>
<tr>
<td></td>
<td>scrubber_datadir</td>
<td>/failover/os-api/glance/scrubber</td>
</tr>
<tr>
<td></td>
<td>registry_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>auth_url</td>
<td><a href="http://os-api-lh:5000/v2.0/">http://os-api-lh:5000/v2.0/</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>glance</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;glanceadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>filesystem_store_datadir</td>
<td>/failover/os-api/glance/images/</td>
</tr>
<tr>
<td></td>
<td>swift_store_auth_address</td>
<td>os-api-lh:5000/v2.0/</td>
</tr>
<tr>
<td></td>
<td>s3_store_host</td>
<td>os-api-lh:8080/v1.0/</td>
</tr>
</tbody>
</table>

Table 3: Glance configuration parameters

In addition, other parameters that are not related to the HA setup may be amended as required [1].

Other OpenStack components required to access the Glance endpoints have to use the highly available IP address managed by the cluster for those services. In this example, the access to glance-api and glance-registry has to be configured for the IP address os-api-lh.

Create the Glance database within zone cluster os-db

Create the Glance database within MySQL and ensure that the Glance user has access from all nodes and zones that are required to connect to the Glance database. Access to the Glance database has to be directed to the logical hostname os-db-lh:

```
os-db-z1# mysql -h os-db-lh -u root -p
mysql> CREATE DATABASE glance;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED by 
   
   '<glancepassword>';
Query OK, 0 rows affected (0.00 sec)
```
Configure Glance services within zone cluster os-api

Install the Glance packages within zone cluster os-api and amend the configuration files as described within the OpenStack installation guide [1] and Table 3:

```
both os-api zones# pkg install glance glanceclient
both os-api zones# cd /etc/glance
## Amend the following config files for glance on all os-api zones:
both os-api zones# vi glance-api.conf
both os-api zones# vi glance-cache.conf
both os-api zones# vi glance-registry.conf
both os-api zones# vi glance-api-paste.ini
both os-api zones# vi glance-registry-paste.ini
both os-api zones# vi glance-scrubber.conf
```

Create the required directories within zpool os-api_zp which is managed by the cluster framework:

```
os-api-z1# mkdir -p /failover/os-api/glance/log
os-api-z1# mkdir /failover/os-api/glance/images
os-api-z1# mkdir /failover/os-api/glance/image-cache
os-api-z1# mkdir /failover/os-api/glance/keystone-signing
os-api-z1# chmod 700 /failover/os-api/glance/keystone-signing
os-api-z1# mkdir /failover/os-api/glance/scrubber
os-api-z1# chown -R glance:glance /failover/os-api/glance
```

Create the Glance database instance and enable the Glance SMF services:

```
os-api-z1# su - glance -c "glance-manage db_sync"
os-api-z1# svcadm enable glance-db
os-api-z1# svcadm enable glance-api
os-api-z1# svcadm enable glance-registry
os-api-z1# svcadm enable glance-scrubber
```

Verify that the Glance services run correctly, then disable them:

```
os-api-z1# su - glance -c "env OS_AUTH_URL=http://os-api-lh:5000/v2.0 "
       OS_USERNAME=glance OS_PASSWORD=<glanceadminpassword> "
       OS_TENANT_NAME=service glance image-list"
os-api-z1# svcadm disable glance-scrubber
os-api-z1# svcadm disable glance-registry
os-api-z1# svcadm disable glance-api
os-api-z1# svcadm disable glance-db
```
Configure Glance SMF services as failover cluster resources within zone cluster os-api

Configure the Glance SMF services as SMF proxy failover resources within resource group os-api-rg. Within the global cluster, define the resource dependencies on the HA MySQL and HA RabbitMQ resources, where required:

```bash
## glance-db
both os-api zones# vi /opt/HA-OpenStack/etc/glance-db-svclist.txt
<svc:/application/openstack/glance/glance-db:default>,
</lib/svc/manifest/application/openstack/glance-db.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \ 
    -p proxied_service_instances=/opt/HA-OpenStack/etc/glance-db-svclist.txt \ 
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \ 
    os-glance-db-rs

## also need a resource dependency on HA MySQL, configure in the global cluster:
cc-node-a# clrs set -Z os-api -p Resource_dependencies=os-db:os-mysql-rs \ 
    os-glance-db-rs

os-api-z1# clrs enable os-glance-db-rs

## glance-api
both os-api zones# vi /opt/HA-OpenStack/etc/glance-api-svclist.txt
<svc:/application/openstack/glance/glance-api:default>,
</lib/svc/manifest/application/openstack/glance-api.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \ 
    -p proxied_service_instances=/opt/HA-OpenStack/etc/glance-api-svclist.txt \ 
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \ 
    -p Resource_dependencies=os-glance-db-rs os-glance-api-rs

## also need a resource dependency on HA SMF rabbitmq, configure in the 
## global cluster:
cc-node-a# clrs set -Z os-api -p Resource_dependencies+=os-mq:os-rabbitmq-rs \ 
    os-glance-api-rs

os-api-z1# clrs enable os-glance-api-rs

## glance-registry
both os-api zones# vi /opt/HA-OpenStack/etc/glance-registry-svclist.txt
<svc:/application/openstack/glance/glance-registry:default>,
</lib/svc/manifest/application/openstack/glance-registry.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \ 
    -p proxied_service_instances=/opt/HA-OpenStack/etc/glance-registry-svclist.txt \ 
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \ 
    -p Resource_dependencies=os-glance-db-rs os-glance-registry-rs

os-api-z1# clrs enable os-glance-registry-rs

## glance-scrubber
both os-api zones# vi /opt/HA-OpenStack/etc/glance-scrubber-svclist.txt
<svc:/application/openstack/glance/glance-scrubber:default>,
</lib/svc/manifest/application/openstack/glance-scrubber.xml>

## Note: The previous content needs to be all one line, without the \ character
```
Once the various glance-* SMF services are started under cluster control, you need to enable and disable the Glance services by enabling and disabling the corresponding os-glance-*-rs cluster resources by using the Oracle Solaris Cluster CLI or BUI.

**Nova**

The Nova compute virtualization service provides a cloud computing fabric controller that supports a variety of virtualization technologies. In Oracle Solaris, virtual machine (VM) instances are kernel zones or non-global zones. Zones are scalable high-density virtual environments with low virtualization overhead. Kernel zones additionally provide independent kernel versions, enabling independent upgrade of VM instances, which is desirable for a multi-tenant cloud.

The following SMF services will be managed by corresponding HA SMF proxy failover resources:

- `svc:/application/openstack/nova/nova-api-ec2:default`
- `svc:/application/openstack/nova/nova-api-osapi-compute:default`
- `svc:/application/openstack/nova/nova-cert:default`
- `svc:/application/openstack/nova/nova-conductor:default`
- `svc:/application/openstack/nova/nova-objectstore:default`
- `svc:/application/openstack/nova/nova-scheduler:default`

In order for the Nova services to use the configured cluster objects, the following configuration files and options have to be amended in addition to options that you normally set up:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/nova/nova.conf</td>
<td>my_ip</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>state_path</td>
<td>/failover/os-api/nova</td>
</tr>
<tr>
<td></td>
<td>neutron_url</td>
<td><a href="http://os-api-lh:9696">http://os-api-lh:9696</a></td>
</tr>
<tr>
<td></td>
<td>neutron_admin_username</td>
<td>neutron</td>
</tr>
<tr>
<td></td>
<td>neutron_admin_password</td>
<td>&lt;neutronadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>neutron_admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>neutron_admin_auth_url</td>
<td><a href="http://os-api-lh:5000/v2.0">http://os-api-lh:5000/v2.0</a></td>
</tr>
<tr>
<td></td>
<td>rabbit_host</td>
<td>os-mq-sa</td>
</tr>
</tbody>
</table>
Table 4: Nova configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rabbit_port</td>
<td>5672</td>
</tr>
<tr>
<td>rabbit_hosts</td>
<td>$rabbit_host:$rabbit_port</td>
</tr>
<tr>
<td>rabbit_userid</td>
<td>guest</td>
</tr>
<tr>
<td>rabbit_password</td>
<td>&lt;rabbitmqpassword&gt;</td>
</tr>
<tr>
<td>connection</td>
<td>mysql://nova:&lt;novapassword&gt;@os-db-lh/nova</td>
</tr>
<tr>
<td>auth_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td>auth_port</td>
<td>35357</td>
</tr>
<tr>
<td>auth_protocol</td>
<td>http</td>
</tr>
<tr>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td>admin_user</td>
<td>nova</td>
</tr>
<tr>
<td>admin_password</td>
<td>&lt;novaadmpassword&gt;</td>
</tr>
<tr>
<td>/etc/nova/api-paste.ini</td>
<td>auth_uri</td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
</tr>
<tr>
<td></td>
<td>signing_dir</td>
</tr>
</tbody>
</table>

In addition, other parameters that are not related to the HA setup may be amended as required [1].

Other OpenStack components required to access the nova endpoints have to make use of the high available IP address that is managed by the cluster for those services. In this example, the access to nova-api-* services has to be configured for the IP address os-api-lh.

Create the Nova database within zone cluster os-db

Create the Nova database within MySQL and ensure the Nova user has access from all nodes and zones that are required to connect to the Nova database. Access to the Nova database has to be directed to the logical hostname os-db-lh:

```
os-db-lh# mysql -h os-db-lh -u root -p
mysql> CREATE DATABASE nova;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED by \\
    '<novapassword>';  
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY '<novapassword>,'
    Query OK, 0 rows affected (0.00 sec)
```

Configure Nova services within zone cluster os-api

Install the Nova packages within zone cluster os-api and amend the configuration files as described within the OpenStack installation guide [1] and Table 4:

```
both os-api zones# pkg install nova novaclient
```
both os-api zones# cd /etc/nova

## Amend the following config files for nova on all os-api zones:

both os-api zones# vi nova.conf
both os-api zones# vi api-paste.ini

=> amend configuration parameters for all the files as described in [1] and table 4

Create the required directories within the zpool os-api_zp, which is managed by the cluster framework:

os-api-z1# mkdir /failover/os-api/nova
os-api-z1# mkdir /failover/os-api/nova/keystone-signing
os-api-z1# chmod 700 /failover/os-api/nova/keystone-signing
os-api-z1# chown -R nova:nova /failover/os-api/nova

Create the Nova database instance and enable the Nova SMF services:

os-api-z1# su - nova -c "nova-manage db sync"

os-api-z1# svcadm enable nova-conductor
os-api-z1# svcadm restart rad:local

os-api-z1# svcadm enable nova-scheduler
os-api-z1# svcadm enable nova-cert
os-api-z1# svcadm enable nova-objectstore
os-api-z1# svcadm enable nova-api-osapi-compute
os-api-z1# svcadm enable nova-api-ec2

Verify that the Nova endpoints are configured correctly, then disable them:

os-api-z1# su - nova -c "env OS_AUTH_URL=http://os-api-lh:5000/v2.0 \
    OS_PASSWORD=<novaadmpassword> OS_USERNAME=nova OS_TENANT_NAME=service \
    nova endpoints"

os-api-z1# svcadm disable nova-api-ec2
os-api-z1# svcadm disable nova-api-osapi-compute
os-api-z1# svcadm disable nova-objectstore
os-api-z1# svcadm disable nova-cert
os-api-z1# svcadm disable nova-scheduler
os-api-z1# svcadm disable nova-conductor

Configure the Nova SMF services as failover cluster resources within zone cluster os-api

Configure the Nova SMF services as SMF proxy failover cluster resources within resource group os-api-rg.

Within the global cluster, define the resource dependencies on the HA MySQL and HA RabbitMQ resources, where required:

## nova-conductor

both os-api zones# vi /opt/HA-OpenStack/etc/nova-conductor-svclist.txt

<svc:/application/openstack/nova/nova-conductor:default>, \
</lib/svc/manifest/application/openstack/nova/conductor.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
    -p proxied_service_instances=/opt/HA-OpenStack/etc/nova-conductor-svclist.txt \
    -m "PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER"
-p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \\  
  os-nova-conductor-rs

## also need a resource dependency on HA MySQL and on HA SMF rabbitmq, configure in 
## the global cluster:
cc-node-a# clrs set -Z os-api \
  -p Resource_dependencies=os-db:os-mysql-rs,os-mq:os-rabbitmq-rs \
  os-nova-conductor-rs

os-api-z1# clrs enable os-nova-conductor-rs

## nova-scheduler
both os-api zones# vi /opt/HA-OpenStack/etc/nova-scheduler-svclist.txt 
<svc://application/openstack/nova/nova-scheduler:default>,\ 
</lib/svc/manifest/application/openstack/nova-scheduler.xml>

## Note: The previous content needs to be all on one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
  -p proxied_service_instances=/opt/HA-OpenStack/etc/nova-scheduler-svclist.txt \
  -p Resource_dependencies_offline_restart=os-api-hasp-rs \ 
  -p Resource_dependencies=os-nova-conductor-rs os-nova-scheduler-rs

## also need a resource dependency on HA MySQL and on HA SMF rabbitmq, configure in 
## the global cluster:
cc-node-a# clrs set -Z os-api \
  -p Resource_dependencies+=os-db:os-mysql-rs,os-mq:os-rabbitmq-rs \
  os-nova-scheduler-rs

os-api-z1# clrs enable os-nova-scheduler-rs

## nova-cert
both os-api zones# vi /opt/HA-OpenStack/etc/nova-cert-svclist.txt 
<svc://application/openstack/nova/nova-cert:default>,\ 
</lib/svc/manifest/application/openstack/nova-cert.xml>

## Note: The previous content needs to be all on one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
  -p proxied_service_instances=/opt/HA-OpenStack/etc/nova-cert-svclist.txt \
  -p Resource_dependencies_offline_restart=os-api-hasp-rs \ 
  -p Resource_dependencies=os-nova-conductor-rs os-nova-cert-rs

## also need a resource dependency on HA MySQL and on HA SMF rabbitmq, configure in 
## the global cluster:
cc-node-a# clrs set -Z os-api \
  -p Resource_dependencies+=os-db:os-mysql-rs,os-mq:os-rabbitmq-rs os-nova-cert-rs

os-api-z1# clrs enable os-nova-cert-rs

## nova-objectstore
both os-api zones# vi /opt/HA-OpenStack/etc/nova-objectstore-svclist.txt 
<svc://application/openstack/nova/nova-objectstore:default>,\ 
</lib/svc/manifest/application/openstack/nova-objectstore.xml>

## Note: The previous content needs to be all on one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
  -p proxied_service_instances=/opt/HA-OpenStack/etc/nova-objectstore-svclist.txt \
  -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \ 
  os-nova-objectstore-rs

os-api-z1# clrs enable os-nova-objectstore-rs
Horizon

Horizon is the OpenStack dashboard where you can manage the cloud infrastructure and computing infrastructure. The dashboard provides a web-based user interface to OpenStack services.

The Horizon dashboard is provided through an Apache web server. The HA Apache data service is used to manage the Apache httpd service. It can be configured as failover or scalable service.

Assuming the /etc/apache2/2.2/samples-conf.d/openstack-dashboard-tls.conf is used, in order for the Horizon service to use the configured cluster objects, the following configuration files and options have to be amended, in addition to options that you normally set up:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/openstack_dashboard/local_settings.py</td>
<td>OPENSTACK_HOST</td>
<td>&quot;os-api-lh&quot;</td>
</tr>
<tr>
<td>/etc/apache2/2.2/conf.d/openstack-dashboard-tls.conf</td>
<td>RedirectPermanent</td>
<td>/horizon <a href="https://os-api-sa.example.com/horizon">https://os-api-sa.example.com/horizon</a></td>
</tr>
<tr>
<td></td>
<td>ServerName</td>
<td>os-api-sa.example.com</td>
</tr>
</tbody>
</table>

Table 5: Horizon configuration parameters
In addition, other parameters not related to the HA setup may be amended as required [1].

OpenStack users accessing the Horizon dashboard have to make use of the highly available IP address managed by the cluster for the HA Apache service. In this example, the user web browser has to access the URL https://os-api-sa.example.com/horizon.

Configure the horizon/https services within zone cluster os-api

Install the horizon packages within zone cluster os-api and amend the configuration file as described within the OpenStack installation guide [1] and Table 5:

```
both os-api zones# pkg install horizon memcached
both os-api zones# cd /etc/openstack_dashboard
os-api-z1# openssl req -new -x509 -nodes -out horizon.crt -keyout horizon.key
  => enter information required to create the certificate
      use the logical hostname used to access the OpenStack dashboard, e.g. os-api-sa.example.com

os-api-z1# chmod 0600 horizon.*
os-api-z1# scp horizon.* os-api-z2:/etc/openstack_dashboard
both os-api zones# cd /etc/apache2/2.2
both os-api zones# cp samples-conf.d/openstack-dashboard-tls.conf conf.d/
both os-api zones# cp -p /etc/openstack_dashboard/local_settings.py /etc/openstack_dashboard/local_settings.py.orig
both os-api-zones# vi /etc/apache2/2.2/conf.d/openstack-dashboard-tls.conf
both os-api zones# vi /etc/openstack_dashboard/local_settings.py
  => amend configuration parameters for all the files as described in [1] and table 5
both os-api zones# svcadm enable apache22
=> verify login at the OpenStack dashboard portal within a web browser:
    https://os-api-sa.example.com/horizon
=> admin / <adminpassword>
Note: While the login should succeed, since at this point the Neutron component is not yet configured, you will see a failure message saying “Connection to neutron failed: ...”, which is to be expected at this point.
both os-api zones# svcadm disable apache22
os-api-z1# scp /var/lib/openstack_dashboard/.secret_key_store 
    os-api-z2:/var/lib/openstack_dashboard/
both os-api-z2# chown webservd:webservd /var/lib/openstack_dashboard/.secret_key_store
```

Configure horizon/https services as HA Apache scalable cluster resource within zone cluster os-api

Configure horizon/https services as SUNW.apache scalable cluster resource os-apache22-rs within resource group os-apache22-rg. The cluster load balancer is configured to use the shared address os-api-sa with TCP ports 80 and 443, with an even client access distribution between the Apache instances:

```
os-api-z1# clrt register SUNW.apache
os-api-z1# clrg create -S os-apache22-rg
os-api-z1# clrs create -d -g os-apache22-rg -t SUNW.apache \
  -p Bin_dir=/usr/apache2/2.2/bin -p Scalable=true \
  -p Load_balancing_policy="Lb_weighted" \
```
-p Load_balancing_weights="1@1,1@2" \
-p Resource_dependencies=os-api-sa-rs -p Port_list=80/tcp,443/tcp \
    os-apache22-rs

os-api-z1# clrg online -eM os-apache22-rg

=> verify login at the OpenStack dashboard portal within a web browser:
    https://os-api-sa.example.com/horizon

=> admin / <adminpassword>

Note: While the login should succeed, since at this point the Neutron component is not yet configured, you will see a failure message saying “Connection to neutron failed: ...”, which is to be expected at this point.

Once Apache is started under cluster control, you need to enable and disable the Apache service by enabling and disabling the os-apache22-rs cluster resource by using the Oracle Solaris Cluster CLI or BUI.

More information about the HA Apache data service can be found within the HA for Apache data service guide [15].

Neutron

The Neutron network virtualization service provides network connectivity for other OpenStack services on multiple OpenStack systems and for VM instances. In Oracle Solaris, network virtualization services are provided through the Elastic Virtual Switch (EVS) capability, which acts as a single point of control for servers and switches in virtualized environments. Applications can drive their own behavior for prioritizing network traffic across the cloud. Neutron provides an API for users to dynamically request and configure virtual networks. These networks connect interfaces from other OpenStack services, such as VNICs from Nova VM instances. The Neutron API supports extensions to provide advanced network capabilities such as quality of service (QoS), access control lists (ACLs), and network monitoring.

Neutron on Oracle Solaris makes use of the plugin for the elastic virtual switch (EVS). As such, it has a runtime dependency on the EVS controller.

The Neutron configuration has three main parts on the cloud controller:

- neutron-server is configured within zone cluster os-api
- EVS controller is configured within a non-global Oracle Solaris zone, which can fail over between the global cluster nodes
- neutron-l3-agent and neutron-dhcp-agent are configured within the global cluster as failover services

Install the Neutron packages within zone cluster os-api

Install the Neutron packages within zone cluster os-api. This will also create the user neutron, which is required to exist in order to set up the SSH keys to communicate with the EVS controller:

both os-api zones# pkg install neutron

Elastic Virtual Switch (EVS) Controller

The EVS controller is a single instance within the overall OpenStack Havana deployment, which makes it a single point of failure (SPOF). Providing HA for the EVS controller is important to ensure that changes can be made through the neutron-server component. Already-deployed VM instances are not affected by an EVS controller outage, but any new deployments or application of network configuration changes would fail.
Since the EVS controller implementation has a fixed location for its own private database file within /etc/evs, currently the only way to provide HA is to wrap the EVS controller into a failover zone.

Configure the non-global zone evs-controller as an HA for Oracle Solaris Zones failover cluster resource

The following procedure configures the EVS controller within a non-global Oracle Solaris zone. That zone is then configured as a failover zone by using the HA for Oracle Solaris Zones data service [16].

```bash
## Configure EVS controller - configured within a solaris failover zone:
## register the SUNW.HAStoragePlus resource type
cc-node-a# clrt register SUNW.HAStoragePlus

## create resource group to manage the EVS controller resources
cc-node-a# clrg create -n cc-node-a,cc-node-b evs-controller-rg

## Create a dedicated zpool for OpenStack EVS controller zone path
cc-node-a# zpool create -m /ha-zones ha-zones_zp <shared storage device 4 cxtydz>
cc-node-a# zpool export ha-zones_zp

## register zpool ha-zones_zp as SUNW.HAStoragePlus resource
cc-node-a# clrs create -t SUNW.HAStoragePlus -g evs-controller-rg \
    -p zpools=ha-zones_zp ha-zones-hasp-rs
cc-node-a# clrg online -eM evs-controller-rg

## Configure the solaris zone to host the EVS controller with additional privileges:
## limitpriv=default,sys_dl_config is needed in case the EVS controller is to
## be run within a non-global zone
both cc-node-a|b# zonecfg -z evs-controller 'create -b ; \
    set zonepath=/ha-zones/solaris/evs-controller ; set autoboot=false; \ 
    set ip-type=shared; add attr; set name=osc-ha-zone; set type=boolean; \ 
    set value=true; end; add net; set physical=sc_ipmp0; \ 
    set address=os-evscon-lh/24; end; set limitpriv=default,sys_dl_config;'

cc-node-a# zoneadm -z evs-controller install
cc-node-a# zoneadm -z evs-controller boot
    => in a different root shell, start
        # zlogin -C -e @ evs-controller
            => go through sysidcfg setup

cc-node-a# zoneadm -z evs-controller shutdown
cc-node-a# zoneadm -z evs-controller detach -F

cc-node-a# clrg switch -n cc-node-b evs-controller-rg
cc-node-b# zoneadm -z evs-controller attach
cc-node-b# zlogin -C -e @ evs-controller
cc-node-b# zoneadm -z evs-controller boot
cc-node-b# zoneadm -z evs-controller shutdown
cc-node-b# zoneadm -z evs-controller detach -F

both cc-node-a|b# cd /opt/SUNWsczone/sczbt/util
both cc-node-a|b# cp -p sczbt_config sczbt_config.evs-controller-rs
both cc-node-a|b# vi sczbt_config.evs-controller-rs
R$=evs-controller-rs
R$=evs-controller-rg
```

---

35 | PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER
Configure the EVS controller within the non-global zone evs-controller

The EVS controller configuration has two main parts:

» Configuration of the EVS controller properties

> Setup of the password-less SSH login between EVS controller and various nodes and zones that interact with the EVS controller

The following procedure installs the EVS controller packages, performs the SSH setup between zone cluster nodes (os-api-z1 and os-api-z1) and the non-global zone hosting the EVS controller (os-evscon-lh), and configures the EVS controller properties used by this example. More information on the EVS controller setup can be found at [1] and [19].

## Ensure that host aliases get resolved which need to interact with the EVS controller. This includes all global cluster nodes, zone cluster nodes and logical hostnames managed by the cluster.

## This can be achieved by amending /etc/inet/hosts to include the IP addresses and corresponding host aliases.

```bash
os-evscon-lh# vi /etc/inet/hosts
```

## Enable root login via ssh

```bash
os-evscon-lh# vi /etc/sshd_config
PermitRootLogin yes
```

## Setup ssh keys for evsuser, neutron and root

```bash
os-api-z1# su - neutron -c "ssh-keygen -N "" -f /var/lib/neutron/.ssh/id_rsa -t rsa"
```

```bash
os-api-z1# scp /var/lib/neutron/.ssh/id_rsa.pub \
    root@os-evscon-lh:/var/user/evsuser/.ssh/authorized_keys
```
### Neutron server

The Neutron-server component requires the MySQL and rabbitmq service to be online and available, otherwise it will fail to start. As part of its startup, it establishes an SSH connection to the EVS controller. As such, it also has an offline-restart dependency on the EVS controller resource.

The following SMF service will be managed by the HA SMF proxy failover resource:

```
svc:/application/openstack/neutron/neutron-server:default
```

In order for the Neutron-server service to use the configured cluster objects, the following configuration files and options need to be amended:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/neutron/neutron.conf</td>
<td>state_path</td>
<td>/failover/os-api/neutron</td>
</tr>
<tr>
<td></td>
<td>bind_host</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>rabbit_host</td>
<td>os-mq-sa</td>
</tr>
<tr>
<td></td>
<td>rabbit_userid</td>
<td>guest</td>
</tr>
<tr>
<td></td>
<td>rabbit_password</td>
<td>&lt;rabbitmqpassword&gt;</td>
</tr>
</tbody>
</table>
In addition, other parameters not related to the HA setup may be amended as required [1].

Other OpenStack components that are required to access the Neutron-server endpoint need to make use of the highly available IP address that is managed by the cluster for this service. In this example, access to neutron-server needs to be configured for the IP address os-api-lh.

Create the Neutron database within zone cluster os-db

Create the Neutron database within MySQL and ensure that the Neutron user has access from all nodes and zones that are required to connect to the Neutron database. Access to the Neutron database needs to be directed to the logical hostname os-db-lh:

```
os-db-z1# mysql -h os-db-lh -u root -p
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE neutron;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED by 'netronadmpassword';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'netronadmpassword';
Query OK, 0 rows affected (0.00 sec)
os-api-z1# vi /etc/neutron/neutron.conf
```

Verify that the Neutron-server SMF service runs correctly, then disable it:

```
both os-api zones# vi /etc/neutron/neutron.conf
both os-api zones# vi /etc/neutron/plugins/evs/evs_plugin.ini
=> amend configuration parameters for all the files as described in [1] and Table 6
```
Configure the `neutron-server` SMF service as a failover cluster resource within zone cluster `os-api`.

Configure the `neutron-server` SMF service as SMF proxy failover resource `os-neutron-server-rs` within resource group `os-api-rg`. Within the global cluster, define the required resource dependency on the HA MySQL, HA Rabbitmq, and HA Zones `sczbt` resources:

```
both os-api zones# vi /opt/HA-OpenStack/etc/neutron-server-svclist.txt
</svc:/application/openstack/neutron/neutron-server:default>,
</lib/svc/manifest/application/openstack/neutron-server.xml>
## Note: The previous content needs to be all one line, without the \ character
```

```
os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
   -p proxied_service_instances=/opt/HA-OpenStack/etc/neutron-server-svclist.txt \
   -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \
   os-neutron-Server-rs
```

```
## also need a resource dependency on HA MySQL and HA SMF rabbitmq, configure in 
## the global cluster:
cc-node-a# clrs set -Z os-api \
   -p Resource_dependencies=os-db:os-mysql-rs,os-mq:os-rabbitmq-rs \
   os-neutron-server-rs
```

```
## also need a offline restart resource dependency on HA sczbt for the 
## evs-controller non-global zone, configure in the global cluster:
cc-node-a# clrs set -Z os-api \
   -p Resource_dependencies_offline_restart=global:evs-controller-rs \
   os-neutron-server-rs
```

```
os-api-z1# clrs enable os-neutron-server-rs
```

Once the `neutron-server` SMF service is started under cluster control, you need to enable and disable the `neutron-server` service by enabling and disabling the corresponding `os-neutron-server-rs` cluster resource by using the Oracle Solaris Cluster CLI or BUI.

**Neutron L3 and Neutron DHCP agent**

The Oracle Solaris 11.2 implementation of OpenStack Neutron supports a provider router with a private networks deployment model. In this deployment model, each tenant can have one or more private networks and all the tenant networks share the same router. The router in this model provides connectivity to the outside world for the tenant VM instances. The router performs bidirectional NAT on the interface that connects the router to the external network. All the gateway interfaces are instantiated on the node that is running the `neutron-l3-agent` SMF service. Though a cloud can have many Neutron instances, you need only one `neutron-l3-agent` per cloud. As such, it is important to provide high availability for the `neutron-l3-agent`.

The `neutron-dhcp-agent` provides DHCP service for the configured tenant networks and is only run as a single instance service. Again, it is important to provide high availability for this service.

The EVS can be configured with two i2-types: VLAN or VxLAN. The combination of VxLAN and IPMP does not work. As such, when configured on a cluster node, it is necessary to use either VLAN or use dedicated separate
network links that are not managed by IPMP. In this example, a dedicated separate network aggregation that is configured with VLANs is used.

The neutron-l3-agent and neutron-dhcp-agent component require the rabbitmq service to be online and available. They also depend on the EVS controller being online and available. The neutron-l3-agent needs to be started before the neutron-dhcp-agent service is started.

The following SMF services will be managed by the HA SMF proxy failover resource:

```shell
» svc:/application/openstack/neutron/neutron-l3-agent:default

» svc:/application/openstack/neutron/neutron-server:default
```

In order for the neutron-l3-agent and neutron-dhcp-agent services to use the configured cluster objects, the following configuration files and options need to be amended in addition to options that you normally set up:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/neutron/neutron.conf</td>
<td>state_path</td>
<td>/failover/os-gz-api/neutron</td>
</tr>
<tr>
<td></td>
<td>rabbit_host</td>
<td>os-mq-sa</td>
</tr>
<tr>
<td></td>
<td>rabbit_password</td>
<td>&lt;rabbitmqpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>rabbit_port</td>
<td>5672</td>
</tr>
<tr>
<td></td>
<td>rabbit_userid</td>
<td>guest</td>
</tr>
<tr>
<td></td>
<td>auth_uri</td>
<td><a href="http://os-api-lh:5000/v2.0">http://os-api-lh:5000/v2.0</a></td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
<td><a href="http://os-api-lh:35357">http://os-api-lh:35357</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>neutron</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;neutronadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>connection</td>
<td>mysql://neutron:&lt;neutronpassword&gt;@os-db-lh/neutron</td>
</tr>
<tr>
<td>/etc/neutron/l3_agent_ini</td>
<td>external_network_datalink</td>
<td>&lt;nic used for external network uplink&gt;</td>
</tr>
<tr>
<td></td>
<td>evs_controller</td>
<td>ssh://evsuser@os-evscon-lh.example.com</td>
</tr>
<tr>
<td>/etc/neutron/dhcp_agent.ini</td>
<td>evs_controller</td>
<td>ssh://evsuser@os-evscon-lh.example.com</td>
</tr>
<tr>
<td>/etc/neutron/plugins/evs/evs_plugin.ini</td>
<td>evs_controller</td>
<td>ssh://evsuser@os-evscon-lh.example.com</td>
</tr>
<tr>
<td></td>
<td>sql_connection</td>
<td>mysql://neutron:&lt;neutronpassword&gt;@os-db-lh/neutron</td>
</tr>
</tbody>
</table>

Table 7: neutron-l3-agent and neutron-dhcp-agent configuration parameters

In addition, other parameters that are not related to the HA setup may be amended as required [1].

Configure the neutron-l3-agent and neutron-dhcp-agent services within the global cluster

Install the Neutron packages and configure the neutron-l3-agent and neutron-dhcp-agent services within the global cluster as described within the OpenStack installation guide [1] and Table 7:

```shell
both cc-node-a|b# pkg install neutron mysql-55/client \
 both cc-node-a|b# cd /etc/neutron
```
both cc-node-a|b# vi neutron.conf
both cc-node-a|b# vi l3_agent.ini
both cc-node-a|b# vi dhcp_agent.ini
both cc-node-a|b# cd /etc/neutron/plugins/evs
both cc-node-a|b# vi evs_plugin.ini

=> amend configuration parameters for all the files as described in [1]
and table 7

Create resource group os-gz-api-rg and the SUNW.HAStoragePlus resource to manage zpool os-gz-api_zp within the global cluster:

cc-node-a# clrg create -n cc-node-a,cc-node-b os-gz-api-rg
cc-node-a# clrs create -t SUNW.HAStoragePlus -g os-gz-api-rg \ 
-p zpools=os-gz-api_zp os-gz-api-hasp-rs
cc-node-a# clrg online -eM os-gz-api-rg

Create the required directories within zpool os-gz-api_zp managed by the cluster framework:

cc-node-a# mkdir -p /failover/os-gz-api/neutron/keystone-signing
cc-node-a# mkdir -p /failover/os-gz-api/neutron/lock
cc-node-a# chown -R neutron:neutron /failover/os-gz-api/neutron
cc-node-a# chmod 700 /failover/os-gz-api/neutron/keystone-signing

In order for the EVS to correctly work across multiple systems, SSH authentication needs to be set-up by using the pre-shared public key between the host where OpenStack components use the evsadm command and the EVS controller. The following procedure creates the SSH setup between neutron-l3-agent and neutron dhcp-agent services that are running on the global cluster nodes (cc-node-a and cc-node-b) and the non-global zone that hosts the EVS controller (os-evscon-lh):

both cc-node-a|b# su - root -c "ssh-keygen -N '' -f /root/.ssh/id_rsa -t rsa"
both cc-node-a|b# su - neutron -c "ssh-keygen -N '' \ 
- f /var/lib/neutron/.ssh/id_rsa -t rsa"
cc-node-a# cat /root/.ssh/id_rsa.pub /var/lib/neutron/.ssh/id_rsa.pub \ 
> /var/tmp/cc-node-a-neutron-root-ssh-pub.pub
cc-node-a# scp /var/tmp/cc-node-a-neutron-root-ssh-pub.txt os-evscon-lh:/var/tmp
cc-node-b# cat /root/.ssh/id_rsa.pub /var/lib/neutron/.ssh/id_rsa.pub \ 
> /var/tmp/cc-node-b-neutron-root-ssh-pub.txt
cc-node-b# scp /var/tmp/cc-node-b-neutron-root-ssh-pub.txt os-evscon-lh:/var/tmp
os-evscon-lh# cat /var/tmp/cc-node-a-neutron-root-ssh-pub.txt \ 
> /var/tmp/cc-node-b-neutron-root-ssh-pub.txt >> \ 
> /var/user/evsuser/.ssh/authorized_keys
both cc-node-a|b# su - neutron -c "ssh evuser@os-evscon-lh.example.com whoami"
both cc-node-a|b# su - root -c "ssh evuser@os-evscon-lh.example.com whoami"
evuser@os-evscon-lh$ scp /var/user/evsuser/.ssh/id_rsa.pub root@cc-node-a:/tmp
evuser@os-evscon-lh$ scp /var/user/evsuser/.ssh/id_rsa.pub root@cc-node-b:/tmp
both cc-node-a|b# cat /tmp/id_rsa.pub >> /root/.ssh/authorized_keys
both cc-node-a|b# su - evuser
evuser@cc-node-a|b$ cat /tmp/id_rsa.pub >> .ssh/authorized_keys

evuser@os-evscon-lh$ ssh root@cc-node-a.example.com true
evuser@os-evscon-lh$ ssh root@cc-node-b.example.com true
evuser@os-evscon-lh$ ssh evuser@cc-node-a.example.com true
evuser@os-evscon-lh$ ssh evuser@cc-node-b.example.com true
For more information on the EVS controller SSH key distribution, refer to the “Configure the Network Node” section within the *Installing and Configuring OpenStack in Oracle Solaris 11.2* user guide [1].

The combination of Neutron with the EVS requires global IP forwarding on the global cluster nodes where the neutron-13-agent is configured to run. The Oracle Solaris Cluster private interconnect and the underlying network interfaces are required to disable IP forwarding for those interfaces. In order to ensure this requirement is implemented, the SMF service `system/cluster/osc-clpriv-ip-forwarding-disable` is provided in the appendix of this white paper (manifest in appendix section "SMF manifest for service system/cluster/osc-clpriv-ip-forwarding-disable" and method script in appendix section "SMF method script for service system/cluster/osc-clpriv-ip-forwarding-disable") and has to be installed on all global cluster nodes.

The following procedure also enables IP Filter and configures the EVS controller property and the external provider router for the neutron-13-agent services within the global cluster nodes for this example:

```bash
## Enable ipfilter and global IP forwarding
both cc-node-a|b# ipadm set-prop -p forwarding=on ipv4
both cc-node-a|b# svcadm enable ipfilter

## Install SMF service to disable IP forwarding on clprivnet and underlying network devices.
## Use the osc-clpriv-ip-forwarding-disable-manifest.xml and svc-osc-clpriv-ip-forwarding-disable method script from appendix to this white paper und put them into /var/tmp on all cluster nodes:
both cc-node-a|b# mkdir -p /opt/HA-OpenStack/bin
both cc-node-a|b# install -c /lib/svc/manifest/system/cluster -m 644 -u root -g sys /var/tmp/osc-clpriv-ip-forwarding-disable-manifest.xml
both cc-node-a|b# install -c /opt/HA-OpenStack/bin -m 755 -u root -g sys /var/tmp/svc-osc-clpriv-ip-forwarding-disable
both cc-node-a|b# svcadm restart manifest-import

## set the EVS controller property:
both cc-node-a|b# evsadm set-prop -p controller=ssh://evsuser@os-evscon-lh.example.com

## Configure the external provider router:
both cc-node-a|b# vi /etc/neutron/l3_agent.ini

Created a new Router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | e82c9561-0ee5-6dd2-b035-f25bff464d98 |
| name                  | provider_router                      |
| status                | ACTIVE                               |
| tenant_id             | 4fa1a0a7b198c30dace9ca46b40fc4a2     |
+-----------------------+--------------------------------------+

## Add the router id to the l3_agent.ini configuration file:
both cc-node-a|b# svcadm enable neutron-l3-agent
both cc-node-a|b# svcadm enable neutron-dhcp-agent
```

PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER
## verify that the neutron-l3-agent and neutron-dhcp-agent services run correctly

cn-node-a# svcadm disable neutron-l3-agent neutron-dhcp-agent

More information on the EVS controller setup can be found at [1] and [19].

Configure neutron-l3-agent and neutron-dhcp-agent SMF services as failover cluster resources within the global cluster

Configure neutron-l3-agent and neutron-dhcp-agent SMF services as SMF proxy failover resources within resource group os-gz-api-rg in the global cluster and define the required dependencies on the HA RabbitMQ service and the failover zone hosting the EVS controller. The neutron-dhcp-agent resource depends on the neutron-l3-agent resource, this ensures that the neutron-l3-agent service is started before the neutron-dhcp-agent service:

## Register HA SMF proxy failover RT in the global cluster

cn-node-a# clr register Proxy_SMF_failover

both cn-node-a|b# mkdir /opt/HA-OpenStack/etc

## neutron-l3-agent resource setup

both cn-node-a|b# vi /opt/HA-OpenStack/etc/neutron-l3-agent-svclist.txt
<svc:/application/openstack/neutron/neutron-l3-agent:default>,
</lib/svc/manifest/application/openstack/neutron-l3-agent.xml>

## Note: The previous content needs to be all one line, without the \ character

cn-node-a# clrs create -d -g os-gz-api-rg -t SUNW.Proxy_SMF_failover \
-p proxied_service_instances=/opt/HA-OpenStack/etc/neutron-l3-agent-svclist.txt \
-p Resource_dependencies_offline_restart=os-gz-api-hasp-rs \
-p Resource_dependencies=os-mq:os-rabbitmq-rs, Evans-controller-rs \
os-neutron-l3-agent-rs

cn-node-a# clrs enable os-neutron-l3-agent-rs

## neutron-dhcp-agent resource setup

both cn-node-a|b# vi /opt/HA-OpenStack/etc/neutron-dhcp-agent-svclist.txt
<svc:/application/openstack/neutron/neutron-dhcp-agent:default>,
</lib/svc/manifest/application/openstack/neutron-dhcp-agent.xml>

## Note: The previous content needs to be all one line, without the \ character

cn-node-a# clrs create -d -g os-gz-api-rg -t SUNW.Proxy_SMF_failover \
-p proxied_service_instances=/opt/HA-OpenStack/etc/neutron-dhcp-agent-svclist.txt \
-p Resource_dependencies_offline_restart=os-gz-api-hasp-rs -p \nResource_dependencies=os-mq:os-rabbitmq-rs, Evans-controller-rs, os-neutron-l3-agent- \
sos-neutron-dhcp-agent-rs

cn-node-a# clrs enable os-neutron-dhcp-agent-rs

Once the neutron-l3-agent and neutron-dhcp-agent SMF services are started under cluster control, you need to enable and disable those services by enabling and disabling the corresponding os-neutron-l3-agent-rs and neutron-dhcp-agent-rs cluster resources by using the Oracle Solaris Cluster CLI or BUI.
Cinder

The Cinder block storage service provides an infrastructure for managing block storage volumes in OpenStack. Cinder enables you to expose block devices and connect block devices to VM instances for expanded storage, better performance, and integration with enterprise storage platforms. In Oracle Solaris, Cinder uses ZFS for storage and uses iSCSI or Fibre Channel for remote access. ZFS provides integrated data services including snapshots, encryption, and deduplication.

While the two `cinder-volume` SMF services can’t be made highly available with Oracle Solaris Cluster on the HA cloud controller when setting `volume_driver` to `ZFSVolumeDriver` (default), `ZFSISCSIDriver`, or `ZFSFCDriver`, it is possible to configure high availability for the `cinder-volume` services when using `ZFSSAISCSIDriver` for `volume_driver` in combination with the Oracle ZFS Storage Appliance [5]. Setup of that configuration on the HA cloud controller is described in a dedicated sub-section, “Cinder-volume services when using the ZFSSAISCSIDriver volume driver for the ZFS Storage Appliance”.

The following SMF services will be managed by corresponding HA SMF proxy failover resources:

- `svc:/application/openstack/cinder/cinder-api:default`
- `svc:/application/openstack/cinder/cinder-backup:default`
- `svc:/application/openstack/cinder/cinder-db:default`
- `svc:/application/openstack/cinder/cinder-scheduler:default`

In order for the Cinder services to use the configured cluster objects, the following configuration files and options need to be amended in addition to options that you normally set up:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/cinder/cinder.conf</td>
<td>state_path</td>
<td>/failover/os-api/cinder</td>
</tr>
<tr>
<td></td>
<td>my_ip</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>sql_connection</td>
<td>mysql://cinder:&lt;cinderpassword&gt;@os-db-lh/cinder</td>
</tr>
<tr>
<td></td>
<td>rabbit_host</td>
<td>os-mq-sa</td>
</tr>
<tr>
<td>/etc/cinder/api-paste.ini</td>
<td>auth_uri</td>
<td><a href="http://os-api-lh:5000/v2.0">http://os-api-lh:5000/v2.0</a></td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
<td><a href="http://os-api-lh:35357">http://os-api-lh:35357</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>cinder</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;cinderadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>signing_dir</td>
<td>/failover/os-api/cinder/keystone-signing</td>
</tr>
</tbody>
</table>

Table 8: Cinder configuration parameters

In addition, other parameters not related to the HA setup may be amended as required [1].

Other OpenStack components that are required to access the Cinder endpoints have to make use of the highly available IP address managed by the cluster for those services. In this example, access to the `cinder-api` service has to be configured for the IP address `os-api-lh`. 

---

PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER

---
Create the Cinder database within zone cluster os-db

Create the Cinder database within MySQL and ensure that the Cinder user has access from all nodes and zones that are required to connect to the Cinder database. Access to the Cinder database needs to be directed to the logical hostname os-db-lh:

```bash
os-db-z1# mysql -h os-db-lh -u root -p
mysql> CREATE DATABASE cinder;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED by '<cinderpassword>';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY '<cinderpassword>';
Query OK, 0 rows affected (0.00 sec)
```

Configure the Cinder services within zone cluster os-api

Install the Cinder packages within zone cluster os-api and amend the configuration files as described within the OpenStack installation guide [1] and Table 8:

```bash
both cc-node-a|b# pkg install system/storage/iscsi/iscsi-target
both os-api zones# pkg install cinder
both os-api zones# cd /etc/cinder
both os-api zones# vi cinder.conf
both os-api zones# vi api-paste.ini
=> amend configuration parameters for all the files as described in [1] and table 8
```

Create the required directories within the zpool os-api_zp which is managed by the cluster framework:

```bash
os-api-z1# mkdir /failover/os-api/cinder
os-api-z1# mkdir /failover/os-api/cinder/keystone-signing
os-api-z1# chmod 700 /failover/os-api/cinder/keystone-signing
os-api-z1# chown -R cinder:cinder /failover/os-api/cinder
```

Create the Cinder database instance, enable the Cinder SMF services, ensure that the Cinder SMF services run correctly, and then disable the Cinder SMF services:

```bash
os-api-z1# su - cinder -c "cinder-manage db sync"
os-api-z1# svcadm enable cinder-db
os-api-z1# svcadm enable cinder-scheduler
os-api-z1# svcadm enable cinder-api
## verify that the cinder services run correctly
os-api-z1# svcadm disable cinder-api
os-api-z1# svcadm disable cinder-scheduler
os-api-z1# svcadm disable cinder-db
```
Configure the Cinder SMF services as failover cluster resources within zone cluster os-api

Configure the Cinder SMF services as SMF proxy failover failover resources within resource group os-api-rg.

Within the global cluster, define the resource dependencies on the HA MySQL and HA RabbitMQ resources, where required:

```bash
## cinder-db
both os-api zones# vi /opt/HA-OpenStack/etc/cinder-db-svclist.txt
<svc:/application/openstack/cinder/cinder-db:default>,
</lib/svc/manifest/application/openstack/cinder-db.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
    -p proxied_service_instances=/opt/HA-OpenStack/etc/cinder-db-svclist.txt \
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \
        os-cinder-db-rs

## also need a resource dependency on HA MySQL, configure in the global cluster:
cc-node-a# clrs set -Z os-api -p Resource_dependencies+=os-db:os-mysql-rs \
    os-cinder-db-rs

os-api-z1# clrs enable os-cinder-db-rs

## cinder-scheduler
both os-api zones# vi /opt/HA-OpenStack/etc/cinder-scheduler-svclist.txt
<svc:/application/openstack/cinder/cinder-scheduler:default>,
</lib/svc/manifest/application/openstack/cinder-scheduler.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
    -p proxied_service_instances=/opt/HA-OpenStack/etc/cinder-scheduler-svclist.txt \
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \
        os-cinder-scheduler-rs

## also need a resource dependency on HA SMF rabbitmq, configure in the global cluster:
cc-node-a# clrs set -Z os-api \
    -p Resource_dependencies+=os-mq:os-rabbitmq-rs os-cinder-scheduler-rs

os-api-z1# clrs enable os-cinder-scheduler-rs

## cinder-api
both os-api zones# vi /opt/HA-OpenStack/etc/cinder-api-svclist.txt
<svc:/application/openstack/cinder/cinder-api:default>,
</lib/svc/manifest/application/openstack/cinder-api.xml>

## Note: The previous content needs to be all one line, without the \ character

os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover \
    -p proxied_service_instances=/opt/HA-OpenStack/etc/cinder-api-svclist.txt \
    -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs \
        os-cinder-api-rs

## also need a resource dependency on HA SMF rabbitmq, configure in the global cluster:
cc-node-a# clrs set -Z os-api \
    -p Resource_dependencies+=os-mq:os-rabbitmq-rs os-cinder-api-rs

os-api-z1# clrs enable os-cinder-api-rs
```
Cinder-volume services when using the ZFSSAISSCISIDriver volume driver for the ZFS Storage Appliance

The Oracle ZFS Storage Appliance iSCSI Cinder driver enables the Oracle ZFS Storage Appliance to be used seamlessly as a block storage resource for Cinder. The driver provides the ability to create iSCSI volumes that can be allocated by a Cinder server to any virtual machine instantiated by the Nova service. The driver is delivered by the cloud/openstack/cinder package.

Follow the procedure described in section “How to Configure the ZFS Storage Appliance iSCSI Cinder Driver” within the Installing and Configuring OpenStack in Oracle Solaris 11.2 guide [5] to set up the cinder.akwf workflow on the Oracle ZFS Storage Appliance.

The following SMF services will be managed by corresponding GDSv2 based failover resources:

» svc:/application/openstack/cinder/cinder-volume:setup

» svc:/application/openstack/cinder/cinder-volume:default

In order for the cinder-volume services to use the iSCSI Cinder driver for the Oracle ZFS Storage Appliance, the following configuration files and options need to be amended, in addition to options that you normally set up and the options explained in Table 8:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/cinder/cinder.conf</td>
<td>volume_driver</td>
<td>cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISSCISIDriver</td>
</tr>
<tr>
<td></td>
<td>zfssa_host</td>
<td>&lt;name or IP address of the ZFSSA management host&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_auth_user</td>
<td>&lt;user name of the Cinder user on the ZFSSA&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_auth_password</td>
<td>&lt;password of the Cinder user on the ZFSSA&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_pool</td>
<td>&lt;pool name to be used to allocate volumes&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_target_portal</td>
<td>&lt;ZFSSA iSCSI target portal (data-ip:port)&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_project</td>
<td>&lt;name of the ZFSSA project&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_initiator_group</td>
<td>&lt;name of the initiator group&gt;</td>
</tr>
<tr>
<td></td>
<td>zfssa_target_interfaces</td>
<td>&lt;ZFSSA iSCSI target network interfaces&gt;</td>
</tr>
</tbody>
</table>

Table 9: Cinder configuration parameters for ZFS SA iSCSI driver

For more information on the specific parameters for the iSCSI driver for the Oracle ZFS Storage Appliance, refer to [5].

Configure the cinder-volume SMF services within zone cluster os-api

Disable the Cinder services already under cluster control and amend the configuration files as described within the OpenStack installation guide [1], Table 8, and Table 9:

```
47 | PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER
```

os-api-z1# clrs disable os-cinder-api-rs os-cinder-scheduler-rs os-cinder-db-rs
both os-api zones# vi /etc/cinder/cinder.conf

```shell
=> amend configuration parameters for all the files as described in [1], table 8 and table 9. Ensure specifically that only the volume driver entry for cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISSCISIDriver is uncommented - all other entries for volume driver must be commented out.
```

Enable the Cinder services already under cluster control and enable the cinder-volume SMF services. Verify that the Cinder volume services get properly started, then disable the cinder-volume SMF services again:
Configure the cinder-volume SMF services as GDSv2 based failover cluster resources within zone cluster

The cinder-volume SMF services use a common SMF manifest, which defines two instance names (setup and default). The HA SMF proxy resource type is currently not able to manage an SMF service with such a manifest. The Generic Data Service (GDSv2) can be used to manage the two cinder-volume SMF services by using a helper script, which can be found in Appendix section “SMF wrapper script to manage the cinder-volume services”.

The following procedure describes how to register the GDSv2 resource type, install the helper script within the zone cluster nodes os-api-z1 and os-api-z2, and register the cinder-volume:setup and cinder-volume:default SMF services as failover cluster resources within resource group os-api-rg:

## Install the helper script to manage the cinder-volume SMF services with GDSv2.
## Use the script smf-wrapper from the appendix section of this white paper and
## put it into /var/tmp on all os-api zones:

```
both os-api zones# mkdir -p /opt/HA-OpenStack/bin
both os-api zones# install -c /opt/HA-OpenStack/bin -m 755 -u root -g sys \
/var/tmp/smf-wrapper
```

## cinder-volume:setup

```
os-api-z1# clrs create -d -g os-api-rg -t ORCL.gds -p PMF_managed=false \  -p start_exit_on_error=TRUE -p stop_exit_on_error=TRUE \  -p start_command="/opt/HA-OpenStack/bin/smf-wrapper start cinder-volume:setup" \  -p stop_command="/opt/HA-OpenStack/bin/smf-wrapper stop cinder-volume:setup" \  -p probe_command="/opt/HA-OpenStack/bin/smf-wrapper probe cinder-volume:setup" \  -p Resource_dependencies=os-cinder-db-rs os-cinder-volume-setup-rs
```

## cinder-volume:default

```
os-api-z1# clrs create -d -g os-api-rg -t ORCL.gds -p PMF_managed=false \  -p start_exit_on_error=TRUE -p stop_exit_on_error=TRUE \  -p start_command="/opt/HA-OpenStack/bin/smf-wrapper start cinder-volume:default" \  -p stop_command="/opt/HA-OpenStack/bin/smf-wrapper stop cinder-volume:default" \  -p probe_command="/opt/HA-OpenStack/bin/smf-wrapper probe cinder-volume:default" \  -p Resource_dependencies=os-cinder-volume-setup-rs os-cinder-volume-default-rs
```

## also need a resource dependency on HA MySQL and on HA SMF rabbitmq, configure in
## the global cluster:
```
cc-node-a# clrs set -p Resource_dependencies=os-db:os-mysql-rs,os-rabbitmq-rs \  os-cinder-volume-default-rs
```

```
os-api-z1# clrs enable os-cinder-volume-default-rs
```
Once the cinder-volume services are started under cluster control, you need to enable and disable the services by enabling and disabling the corresponding os-cinder-volume-setup-rs and os-cinder-volume-default-rs cluster resources.

More information about the Generic Data Service can be found within the Oracle Solaris Cluster Generic Data Service Guide [20].

Swift

The Swift object storage service provides redundant and scalable object storage services for OpenStack projects and users. Swift stores and retrieves arbitrary unstructured data using ZFS, and the data is then accessible via a RESTful API. The Swift components are typically deployed on dedicated storage nodes. High availability of the Swift storage is achieved by using Swift features to configure the Swift ring. In Swift, objects are protected by storing multiple copies of data so that, if one Swift storage node fails, the data can be retrieved from another Swift storage node. However, there is one Swift component that can run on the HA cloud controller, the swift-proxy-server service.

The proxy server processes are the public face of Swift, as they are the only ones that communicate with external clients. As a result, they are the first and last to handle an API request. All requests to and responses from the proxy use standard HTTP verbs and response codes.

The following SMF service will be managed by a corresponding HA SMF proxy failover resource:

```
svc:/application/openstack/swift/swift-proxy-server:default
```

In order for the swift-proxy-server to use the configured cluster objects, the following configuration files and options need to be amended in addition to options that you normally setup:

<table>
<thead>
<tr>
<th>configuration file</th>
<th>option</th>
<th>value to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/swift/swift.conf</td>
<td>swift_hash_path_suffix</td>
<td>&lt;swift_hash_path_suffix value&gt;</td>
</tr>
<tr>
<td></td>
<td>swift_hash_path_prefix</td>
<td>&lt;swift_hash_path_prefix value&gt;</td>
</tr>
<tr>
<td>/etc/swift/proxy-server.conf</td>
<td>bind_ip</td>
<td>os-api-lh</td>
</tr>
<tr>
<td></td>
<td>bind_port</td>
<td>8080</td>
</tr>
<tr>
<td></td>
<td>auth_uri</td>
<td><a href="http://os-api-lh:5000/">http://os-api-lh:5000/</a></td>
</tr>
<tr>
<td></td>
<td>identity_uri</td>
<td><a href="http://os-api-lh:35357">http://os-api-lh:35357</a></td>
</tr>
<tr>
<td></td>
<td>admin_tenant_name</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>admin_user</td>
<td>swift</td>
</tr>
<tr>
<td></td>
<td>admin_password</td>
<td>&lt;swiftadmpassword&gt;</td>
</tr>
<tr>
<td></td>
<td>signing_dir</td>
<td>/failover/os-api/swift/keystone-signing</td>
</tr>
</tbody>
</table>

Table 10: Swift proxy-server configuration parameters

In addition, other parameters that are not related to the HA setup may be amended as required [4].

Other OpenStack components that are required to access the swift-proxy-server endpoint need to make use of the highly available IP address that is managed by the cluster for this service. In this example, access to the swift-proxy-server service needs to be configured for the IP address os-api-lh.
Configure swift-proxy-server service within zone cluster os-api

Install the Swift packages within zone cluster os-api and amend the configuration files as described within the OpenStack Swift documentation [4] and Table 10:

```
both os-api zones# pkg install swift swiftclient
both os-api zones# cd /etc/swift
both os-api zones# vi swift.conf
both os-api zones# vi proxy-server.conf
=> amend configuration parameters for all the files as described in [4] and table 10

Note that the same configuration for those files needs to be also applied on all the Swift storage nodes.
```

Create the required directories within the zpool os-api_zp which is managed by the cluster framework:

```
os-api-z1# mkdir /failover/os-api/swift
os-api-z1# mkdir /failover/os-api/swift/keystone-signing
os-api-z1# chmod 700 /failover/os-api/swift/keystone-signing
os-api-z1# chown -R swift:swift /failover/os-api/swift
```

Follow the OpenStack Swift documentation [4] to configure the Swift rings. Set up the corresponding Swift storage nodes and replicate the required configuration information from /etc/swift on the proxy-server setup to the Swift storage nodes.

Enable the swift-proxy-server SMF service, verify the Swift setup, then disable the swift-proxy-server SMF service again:

```
os-api-z1# svcadm enable swift-proxy-server
## Verify the Swift setup
os-api-z1# svcadm disable swift-proxy-server
```

Configure the swift-proxy-server SMF service as a failover cluster resource within zone cluster os-api

```
both os-api zones# vi /opt/HA-OpenStack/etc/swift-proxy-server-svclist.txt
<svc:/application/openstack/swift/swift-proxy-server:default>,
</lib/svc/manifest/application/openstack/swift-proxy-server.xml>
## Note: The previous content needs to be all one line, without the \ character
os-api-z1# clrs create -d -g os-api-rg -t SUNW.Proxy_SMF_failover -p proxied_service_instances=/opt/HA-OpenStack/etc/swift-proxy-server-svclist.txt -p Resource_dependencies_offline_restart=os-api-hasp-rs,os-api-lh-rs os-swift-proxy-server-rs
os-api-z1# clrs enable os-swift-proxy-server-rs
```

Once the swift-server-proxy service is started under cluster control, you need to enable and disable the swift-proxy-server service by enabling and disabling the os-swift-proxy-server-rs cluster resource.

Once the swift-server-proxy service is started under cluster control, you need to enable and disable the swift-proxy-server service by enabling and disabling the os-swift-proxy-server-rs cluster resource.
OpenStack Components Not Running on the HA Cloud Controller

In this example, the following OpenStack components are running outside the HA cloud controller:

> nova-compute service on the Nova compute nodes

> swift-* services (except swift-proxy-server) on the Swift storage nodes

When configuring access to OpenStack endpoints for those services, configure the logical hostnames that correspond to the OpenStack endpoints as listed in Table 1.

For the Nova compute nodes, the SSH public keys have to be set up such that the Nova compute nodes can ssh into and from the EVS controller `os-evscon-lh.example.com` logical hostname as user `evsuser`. Further details are explained within the *Installing and Configuring OpenStack in Oracle Solaris 11.2* user guide [1].
References


Appendix

SMF manifest for service system/cluster/osc-clpriv-ip-forwarding-disable

Save this file with the name osc-clpriv-ip-forwarding-disable-manifest.xml:

```xml
<service_bundle type='manifest' name='ha-openstack:osc-clpriv-ip-forwarding-disable'>
  <service
    name='system/cluster/osc-clpriv-ip-forwarding-disable'
    type='service'
    version='1'>
    <create_default_instance enabled='true' />
    <single_instance />
    <dependency
      name='filesystem-minimal'
      grouping='require_all'
      restart_on='none'
      type='service'>
      <service_fmri value='svc:/system/filesystem/minimal:default'/>
    </dependency>
    <dependency
      name='scprivipd'
      grouping='require_all'
      restart_on='none'
      type='service'>
      <service_fmri value='svc:/system/cluster/scprivipd:default'/>
    </dependency>
    <exec_method
      type='method'
      name='start'
      exec='/opt/HA-OpenStack/bin/svc-osc-clpriv-ip-forward-disable'
      timeout_seconds='30'>
      <method_context>
        <method_credential user='root' group='root'/>
      </method_context>
    </exec_method>
    <exec_method
      type='method'
      name='stop'
      exec=':true'
      timeout_seconds='0' />
    <property_group name='startd' type='framework'>
      <propval name='ignore_error' type='astring' value='core,signal' />
      <propval name='duration' type='astring' value='transient' />
    </property_group>
    <stability value='Unstable' />
  </service>
</service_bundle>
```

Oracle Solaris Cluster service to disable IP forwarding for the network devices used by the cluster interconnect
SMF method script for service `system/cluster/osc-clpriv-ip-forwarding-disable`

Save this file with the name `svc-osc-clpriv-ip-forward-disable`:

```bash
#!/bin/ksh
#
# Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved.
#
# The purpose of this script is to ensure that IPv4 and IPv6 forwarding for
# clprivnet0 and the underlying network interfaces used by the Oracle
# Solaris Cluster private interconnect is disabled.
#
# By default IPv4 and IPv6 forwarding is globally disabled on a Solaris system.
# If IPv4 and/or IPv6 forwarding is enabled globally, this script ensures that
# IPv4 and IPv6 forwarding is disabled per interface for clprivnet0 and the
# underlying network interfaces used for the Oracle Solaris Cluster
# private interconnect.
#
# . /lib/svc/share/smf_include.sh

typeset -r IPADM=/usr/sbin/ipadm
.typeset -r ROUTEADM=/usr/sbin/routeadm
typeset -r SCCONF=/usr/cluster/bin/scconf
.typeset -r AWK=/usr/bin/awk
.typeset -r SED=/usr/bin/sed
.typeset -r HOSTNAME=/usr/bin/hostname
.typeset rc=${SMF_EXIT_OK}

.export LC_ALL=C

for iptype in ipv4 ipv6
do
  if ! [ $(ROUTEADM) -p ${iptype}-forwarding | 
     $(AWK) -F= 'BEGIN {RS="\""} $1 == "persistent" {FS="\"; print $2\"}'} == enabled ]
    then
      typeset nodename="${HOSTNAME}""" 
      typeset clprivadapter="${SCCONF} -pv | 
        $(SED) -n 's/^ *${nodename} Node transport adapters: *\]""}*\(.*\)/\1/p'""

      for interface in clprivnet0 ${clprivadapter}
        do
          if ! [ $(IPADM) show-ifprop -c -p forwarding -o current -m ${iptype} 
            ${interface} == on ]
            then
              $(IPADM) set-ifprop -t -p forwarding=off -m ${iptype} ${interface}
              if (( $? != 0 ))
                then
                  print -u2 "Error: Could not temporary disable" \ 
                      "IP forwarding for ${iptype} on ${interface}.)"
                  rc=1
                else
                  print "Temporary disable IP forwarding for" \ 
                      "${iptype} on ${interface}.)"
          fi
        done
  done

.exit $rc
```

54 | PROVIDING HIGH AVAILABILITY TO THE OPENSTACK CLOUD CONTROLLER ON ORACLE SOLARIS WITH ORACLE SOLARIS CLUSTER
SMF wrapper script to manage the cinder-volume services

Save this file with the name smf-wrapper:

```bash
#!/bin/ksh
#
# Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved.
#
typeset -r SVCADM=/usr/sbin/svcadm
typeset -r SVCS=/usr/bin/svcs
typeset -r BASENAME=/usr/bin/basename

typeset command_name="$(basename $0)"
typeset subcommand="$1"
typeset fmri="$2"

function usage {
    print -u2 "usage: ${command_name} start <fmri>"
    print -u2 "${command_name} stop <fmri>"
    print -u2 "${command_name} probe <fmri>"
    exit 1
}

## Main

if [[ -z ${fmri} ]]
then
    usage
fi

case "${subcommand}" in
    start)
        $(SVCADM) enable -st ${fmri}
        exit $? ;;
    stop)
        $(SVCADM) disable -s ${fmri}
        exit $? ;;
    probe)
        if [[ $(SVCS -Ho state ${fmri}) != online ]]
        then
            exit 100
        else
            exit 0
        fi
    *)
        usage
        exit 1
esac
```
Providing High Availability to the OpenStack Cloud Controller on Oracle Solaris with Oracle Solaris Cluster

April 2015

Author: Thorsten Früauf