Red Hat Reference Architecture Series

Deploying Red Hat Enterprise Linux OpenStack Platform 7

with RHEL-OSP director

Jacob Liberman

Version 2.0, November 9, 2015

Table of Contents

100 East Davie Street
Raleigh NC 27601 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is: CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Send feedback to refarch-feedback@redhat.com

Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any of the reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

https://www.facebook.com/rhrefarch

Follow us on Twitter:

https://twitter.com/RedHatRefArch

Plus us on Google+

https://plus.google.com/u/0/b/114152126783830728030/

Executive Summary

This reference architecture describes a realistic use case for deploying a microservices architecture application on a Red Hat Enterprise Linux OpenStack Platform 7 cluster. It begins with steps for deploying RHEL-OSP 7 on baremetal servers via RHEL-OSP director, Red Hat's new deployment toolchain. Next it describes Red Hat's approach to implementing highly available OpenStack. The reference architecture concludes with instructions for implementing a microservices architecture that provides shopping cart functionality via a multi-tier web application.

This reference architecture has been updated for RHEL-OSP director 7.1.

Architecture Overview

Red Hat Enterprise Linux OpenStack Platform (RHEL-OSP) delivers an integrated foundation to create, deploy, and scale an OpenStack cloud. RHEL-OSP 7, Red Hat's 5th iteration of OpenStack Platform, is based on the community Kilo OpenStack release. This version introduces RHEL-OSP director, Red Hat's new deployment toolchain. RHEL-OSP director combines functinality from the upstream `TripleO` and `Ironic` projects with components from Red Hat's previous installers.

Red Hat JBoss Enterprise Application Platform 6 (EAP) is a fully-certified Java EE platform to quickly deploy and develop enterprise applications. This reference architecture describes a realistic use case for deploying an EAP 6 Microservices Architecture (MSA) on a high availability RHEL-OSP 7 cluster. A microservices architecture is a modular enterprise application where individual instances or containers run single services and communicate via lightweight protocols and APIs. The EAP 6 MSA used in this reference architecture is a multi-tier shopping cart that processes customer transactions and logs them in a backend database.

The complete reference architecture use case provides a comprehensive, end-to-end example of deploying a RHEL-OSP 7 cloud on baremetal using OpenStack director then implementing the microservices architecture via `Heat` templates. This reference architecture complements existing RHEL-OSP documentation by providing a *comprehensive example* of deploying a complex enterprise web application on OpenStack, demonstrating RHEL-OSP 7's features and tools in a realistic context.

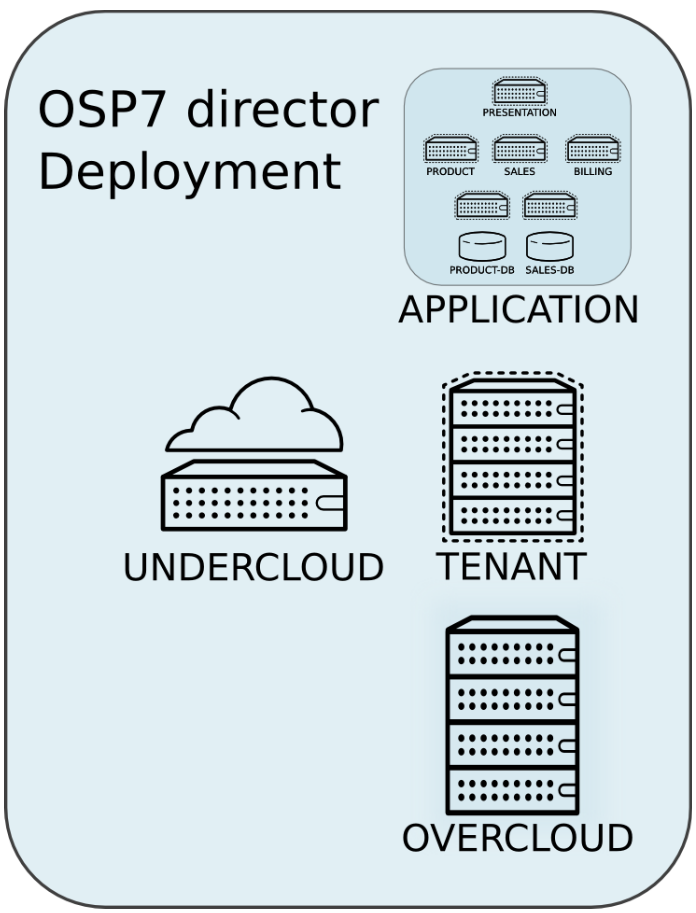

Figure 2.1: OpenStack Platform director

The first section of this reference architecture introduces the principal components: Red Hat Enterprise Linux OpenStack Platform 7, RHEL-OSP director, and a microservices architecture built with Red Hat JBoss Enterprise Application Platform 6. It also describes Red Hat's approach to making OpenStack high availability. Core OpenStack services are managed and monitored in a High Availability (HA) cluster. A load balancer provides access to the service endpoints. There are no direct connections from the clients to the the services. This approach allows administrators to manage, operate, and scale services together or independently.

The second section of the paper describes the lab environment, hardware, and software used to implement and test the reference architecture.

The third section documents the installation and configuration procedure. All of the steps listed in this document were performed by the Red Hat Systems Engineering team. The complete use case was deployed in the Systems Engineering lab on bare metal servers using RHEL-OSP director and generally available code.

This reference architecture has been updated for RHEL-OSP director 7.1.

OpenStack Platform 7 director

RHEL-OSP delivers an integrated foundation to create, deploy, and scale a secure and reliable public or private OpenStack cloud. RHEL-OSP starts with the proven foundation of Red Hat Enterprise Linux and integrates Red Hat's OpenStack Platform technology to provide a production-ready cloud platform backed by an ecosystem of more than 350 certified partners.

RHEL-OSP 7 is based on the community Kilo OpenStack release. This release is Red Hat's fifth iteration of RHEL-OSP. It has been successfully deployed by Red Hat customers worldwide across diverse vertical industries including financial, telecommunications, and education.

RHEL-OSP 7 introduces RHEL-OSP director, a cloud installation and lifecycle management tool chain. Director is the first RHEL-OSP installer to deploy OpenStack on and with OpenStack. This section of the paper introduces RHEL-OSP director's architecture and features:

- Simplified deployment through ready-state provisioning of bare metal resources.
- Flexible network definitions
- High availability via tight integration with the Red Hat Enterprise Linux Server High Availability Add-on
- Integrated setup and installation of Red Hat Ceph Storage 1.3
- Content management via the Red Hat Content Delivery Network or Red Hat Satellite Server.

Ready State Provisioning and Server Roles

RHEL-OSP director is a converged installer. It combines mature upstream OpenStack deployment projects (`TripleO` and `Ironic`) with components from Red Hat's past RHEL-OSP installers.

`TripleO` stands for *OpenStack on OpenStack*. `TripleO` is an upstream OpenStack project that uses an existing OpenStack environment to install a production OpenStack environment. The deployment environment is called the `undercloud`. The production environment is called the `overcloud`.

The `undercloud` is the `TripleO` control plane. It uses native OpenStack APIs and services to deploy, configure, and manage the production OpenStack deployment. The `undercloud` defines the overcloud with `Heat` templates then deploys it via the `Ironic` baremetal provisioning service. RHEL-OSP director includes predefined `Heat` templates for the basic server roles that comprise the overcloud. Customizable templates allow director to deploy, redeploy, and scale complex overclouds in a repeatable fashion.

`Ironic` is a community bare-metal provisioning project. Director uses `Ironic` to deploy the overcloud servers. `Ironic` gathers information about baremetal servers via a discovery mechanism known as introspection. `Ironic` pairs servers with bootable images and installs them via PXE and remote power management.

RHEL-OSP director deploys all servers with the same generic image by injecting `Puppet modules` into the image to tailor it for specific server roles. It then applies host-specific customizations via Puppet including network and storage configurations.

While the `undercloud` is primarily used to deploy OpenStack, the `overcloud` is a functional cloud available to run virtual machines and workloads. Servers in the following roles comprise the overcloud:

Control

This role provides endpoints for REST-based API queries to the majority of the OpenStack services. These include Compute, Image, Identity, Block, Network, and Data processing. The controller can run as a standalone server or as a HA cluster.

Compute

This role provides the processing, memory, storage, and networking resources to run virtual machine instances. It runs the KVM hypervisor by default. New instances are spawned across compute nodes in a round-robin fashion based on resource availability.

Block storage

This role provides external block storage for HA controller nodes via the OpenStack Block Storage service `Cinder`.

Ceph storage

Ceph is a distributed object store and file system. This role deploys Object Storage Daemon (OSD) nodes for Ceph clusters. It also installs the Ceph Monitor service on the controller.

Object storage

This role provides external Account, Container, and Object (ACO) storage for the OpenStack Object Storage service, `Swift`, by installing a `Swift` proxy server on the controller nodes.

> The overcloud requires at least one controller and one compute node. It runs independently from the undercloud once it is installed. This reference architecture uses the Control, Compute, and Ceph storage roles.

RHEL-OSP director also includes **advanced hardware configuration** tools from the eNovance SpinalStack installer. These tools validate server hardware prior to installation. **Profile matching** lets administrators specify hardware requirements for each server role. RHEL-OSP director only matches servers that meet minimum hardware requirements for each role. Profile to matching is performed after introspection but prior to deployment.

RHEL-OSP director also supports pre-installation **benchmark collection**. Servers boot to a customized RAMdisk and run a series of benchmarks. The benchmarks report performance outliers prior to installation.

RHEL-OSP 7 requires Red Hat Enterprise Linux 7 Server on all servers. Supported guest operating systems can be found at https://access.redhat.com/articles/973163. Deployment limitations are listed at https://access.redhat.com/articles/1436373.

Network Isolation

OpenStack requires multiple network functions. While it is possible to collapse all network functions onto a single network interface, isolating communication streams in their own physical or virtual networks provides better performance and scalability.

RHEL-OSP director supports isolating network traffic by type. One or more network traffic types can be assigned to a physical, virtual, or bonded interface. Multiple traffic types can be combined across the same physical interfaces or switches. Each OpenStack service is bound to an IP on a particular network. In a cluster a service virtual IP is shared among all of the HA controllers.

RHEL-OSP director supports network isolation for the following traffic types:

Provisioning

The control plane installs the overcloud via this network. All cluster nodes must have a physical interface attached to the provisioning network. This network carries DHCP/PXE and TFTP traffic so it must be provided on a dedicated interface or a native VLAN to the boot interface. The provisioning interface can act as a default gateway for the overcloud if there is no other gateway on the network.

RHEL-OSP director 7.1 supports static IPs on the provisioning network. Using static IPs requires additional parameters in the network isolation Heat templates for setting static IPs, routes, and DNS servers. Configure network isolation describes the additional parameters for configuring static IPs on the provisioning network.

Disable PXE on the remaining interfaces to ensure the servers boot from this network.

External

This network provides overcloud nodes with external connectivity. Controller nodes connect the external network to an `Open vSwitch` bridge and forward traffic originating from hypervisor instances through it. The `Horizon` service and OpenStack public API endpoints can share this network or they can be broken out to an optional public API network.

Internal API

This network exposes internal OpenStack API endpoints for the overcloud nodes. It handles inter-service communication between both core OpenStack services and the supporting services. By default this network also hosts cluster management traffic used by HA services to share data and track cluster state for automated failover. It is common practice to break the cluster management traffic out to a separate network if it affects performance or scaling. Supporting service traffic from the state database, message bus, and hostname resolution is also delivered via this network.

Tenant

Virtual machines communicate over the tenant network. It supports three modes of operation: VXLAN, GRE, and VLAN. VXLAN and GRE tenant traffic is delivered via software tunnels on a single VLAN. In VLAN mode, individual VLANs correspond to tenant networks.

Storage

This network carries storage communication including `Ceph`, `Cinder`, and `Swift` traffic. The virtual machine instances communicate with the storage servers via this network. Data-intensive OpenStack deployments should isolate storage traffic on a dedicated high bandwidth interface, i.e. 10 GB interface. The `Glance` API, `Swift` proxy, and `Ceph Public interface` services are all delivered via this network.

Storage Management

Storage management communication can generate large amounts of network traffic. This network is shared between the front and back end storage nodes. Storage controllers use this network to access data storage nodes. This network is also used for storage clustering and replication traffic.

Network traffic types are assigned to network interfaces through `Heat` template customizations prior to deploying the overcloud. RHEL-OSP director supports several network interface types including physical interfaces, bonded interfaces, and either tagged or native 802.1Q VLANs.

Disable DHCP on unused interfaces to avoid unwanted routes and network loops.

Network Types by Server Role

The previous section discussed server roles. Each server role requires access to specific types of network traffic. Figure 3.1 Network Topology depicts the network roles by server type in this reference architecture.

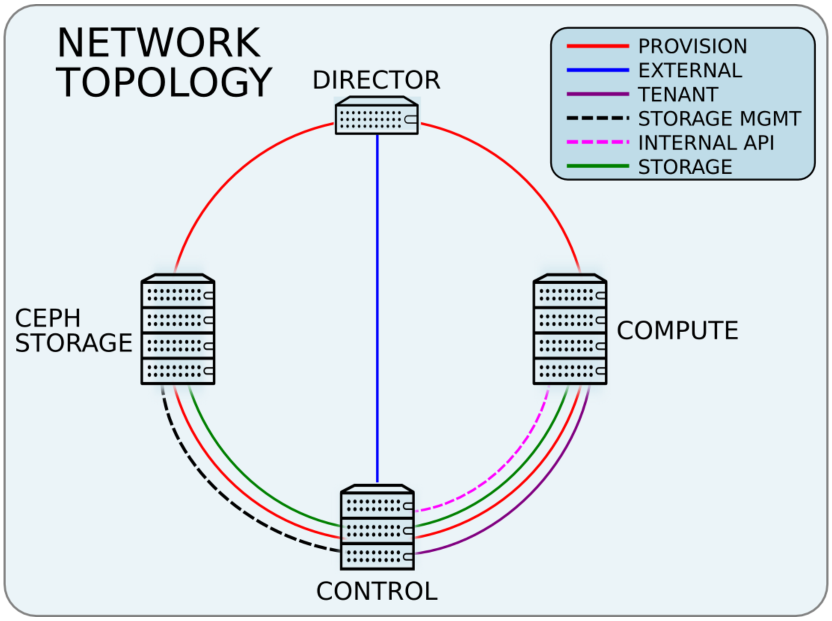

Figure 3.1 Network Topology

The network isolation feature allows RHEL-OSP director to segment network traffic by particular network types. When using network isolation, each server role must have access to its required network traffic types.

By default, RHEL-OSP director collapses all network traffic to the provisioning interface. This configuration is suitable for evaluation, proof of concept, and development environments. It is not recommended for production environments where scaling and performance are primary concerns.

Table 1 Network type by server role summarizes the required network types by server role.

Table 1. Network type by server role

Role	Network
Undercloud	External
	Provisioning
Control	External
	Provisioning
	Storage Mgmt
	Tenant
	Internal API
	Storage

Role	Network
Compute	Provisioning
	Tenant
	Internal API
	Storage
Ceph/Block/Object Storage	Provisioning
	Storage Mgmt
	Storage

Tenant Network Types

RHEL-OSP 7 supports tenant network communication through the OpenStack Networking (`Neutron`) service. OpenStack Networking supports overlapping IP address ranges across tenants via the Linux kernel's `network namespace` capability. It also supports three default networking types:

VLAN segmentation mode

Each tenant is assigned a network subnet mapped to a 802.1q VLAN on the physical network. This tenant networking type requires VLAN-assignment to the appropriate switch ports on the physical network.

GRE overlay mode

This mode isolates tenant traffic in virtual tunnels to provide Layer 2 network connectivity between virtual machine instances on different hypervisors. GRE does not require changes to the network switches and supports more unique network IDs than VLAN segmentation.

VXLAN

This overlay method similar to GRE. VXLAN combines the ease and scalability of GRE with superior performance. This is the default mode of operation for Red Hat Enterprise Linux OpenStack Platform 7 as of the Y1 release.

Although Red Hat certifies third-party network plug-ins, RHEL-OSP director uses the `ML2` network plugin with the `Open vSwitch` driver by default.

RHEL-OSP director does not deploy legacy (`Nova`) networking.

High Availability

RHEL-OSP director's approach to high availability OpenStack leverages Red Hat's internal expertise with distributed cluster systems. Most of the technologies discussed in this section are available through the Red Hat Enterprise Linux Server High Availability Add On. These technologies are bundled with RHEL-OSP 7 to provide cluster services for production deployments.

Cluster Manager and Proxy Server

Two components drive HA for all core and non-core OpenStack services: the **cluster manager** and the **proxy server**.

The cluster manager is responsible for the startup and recovery of an inter-related services across a set of physical machines. It tracks the cluster's internal state across multiple machines. State changes trigger appropriate responses from the cluster manager to ensure service availability and data integrity.

Cluster manager benefits

1. Deterministic recovery of a complex, multi-machine application stack.
2. State awareness of other cluster machines to co-ordinate service startup and failover.
3. Shared quorum calculation to determine majority set of surviving cluster nodes after a failure.
4. Data integrity through fencing. Machines running a non-responsive process are isolated to ensure they are not still responding to remote requests. Machines are typically fenced via a remotely accessible power switch or IPMI controller.
5. Automated recovery of failed instances to prevent additional load-induced failures.

In RHEL-OSP's HA model, clients do not directly connect to service endpoints. Connection requests are routed to service endpoints by a proxy server.

Proxy server benefits

1. Connections are load balanced across service endpoints
2. Service requests can be monitored in a central location
3. Cluster nodes can be added or removed without interrupting service

RHEL-OSP director uses `HAproxy` and `Pacemaker` to manage HA services and load balance connection requests. With the exception of `RabbitMQ` and `Galera`, `HAproxy` distributes connection requests to active nodes in a round-robin fashion. `Galera` and `RabbitMQ` use persistent options to ensure requests go only to active and/or synchronized nodes. `Pacemaker` checks service health at 1 second intervals. Timeout settings vary by service.

Benefits of combining Pacemaker and HAproxy

The combination of `Pacemaker` and `HAproxy`:

- Detects and recovers machine and application failures
- Starts and stops OpenStack services in the correct order
- Responds to cluster failures with appropriate actions including resource failover and machine

restart and fencing

- Provides a thoroughly tested code base that has been used in production clusters across a variety of use cases

The following services deployed by RHEL-OSP director do not use the `HAproxy` server:

1. `RabbitMQ`
2. `memcached`
3. `mongodb`

These services have their own failover and HA mechanisms. In most cases the clients have full lists of all service endpoints and try them in a round robin fashion. Individual cluster services are discussed in the following section.

> RHEL-OSP director uses `Pacemaker` and `HAproxy` for clustering. Red Hat also supports manually deployed RHEL-OSP 7 clustered with `keepalived` and `HAproxy`. Manual installation is beyond the scope of this document.

Cluster models: Segregated versus Collapsed

Cluster services can be deployed across cluster nodes in different combinations. The two primary approaches are *segregated* and *collapsed*.

Segregated clusters run each service on dedicated clusters of three or more nodes. Components are isolated and can be scaled individually. Each service has its own virtual IP address. Segregating services offers flexibility in service placement. Multiple services can be run on the same physical nodes, or, in an extreme case, each service can run on its own dedicated hardware.

Figure 3.2 Segregated Cluster Services depicts OpenStack service deployed in a segregated cluster model. Red Hat supports RHEL-OSP 7 services deployed in a segregated model but it is beyond the scope of this document.

Collapsed clusters run every service and component on the same set of three or more nodes. Cluster services share the same virtual IP address set. Collapsed services require fewer physical machines and are simpler to implement and manage.

Previous Red Hat Enterprise Linux OpenStack Platform installers deployed segregated clusters. RHEL-OSP director deploys overclouds as collapsed clusters. All controller nodes run the same services. Service endpoints are bound to the same set of virtual IP addresses. The undercloud is not clustered.

Figure 3.3 Collapsed Cluter Services depicts RHEL-OSP director's default approach to deploying collapsed HA OpenStack services.

Segregated and collapsed are the dominant approaches to implementing HA clusters but hybrid approaches are also possible. Segregate one or more components expected to cause a bottleneck into individual clusters. Collapse the remainder. Deploying a mixed cluster is beyond the scope of this document.

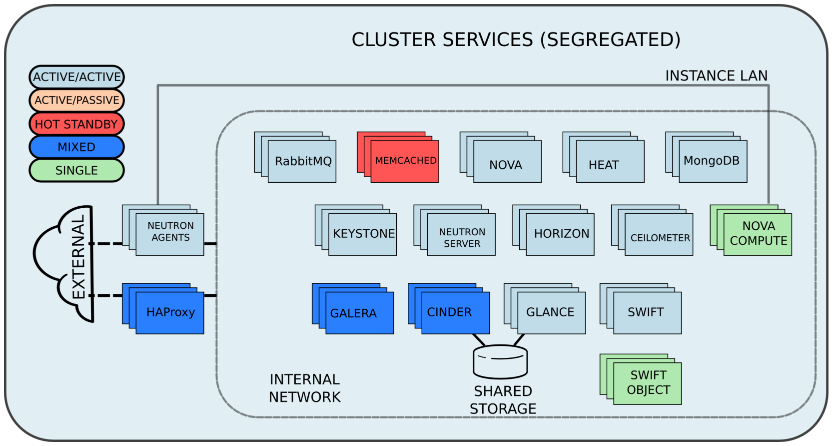

Figure 3.2: Segregated Cluster Services

Cluster Services and Quorum

Each clustered service operates in one of the following modes:

Active/active

Requests are load balanced between multiple cluster nodes running the same services. Traffic intended for failed nodes is sent to the remaining nodes.

Active/passive

A redundant copy of a running service is brought online when the primary node fails.

Hot Standby

Connections are only routed to one of several active service endpoints. New connections are routed to a standby endpoint if the primary service endpoint fails.

Mixed

Mixed has one of two meanings: services within a group run in different modes, or the service runs active/active but is used as active/passive. Mixed services are explained individually.

Single

Each node runs an independent cluster manager that only monitors its local service.

A cluster **quorum** is the majority node set when a failure splits the cluster into two or more partitions. In this situation the majority fences the minority to ensure both sides are not running the same services — a so-called *split brain* situation. **Fencing** is the process of isolating a failed machine — typically via remote power control or networked switches — by powering it off. This is necessary to ensure data integrity.

Although `Pacemaker` supports up to 16 cluster nodes, Red Hat recommends an odd number of cluster members to help ensure quorum during cluster communication failure. RHEL-OSP director requires three active cluster members to achieve quorum.

Cluster Modes for Core Services

This section of the paper describes RHEL-OSP director’s default cluster mode for each OpenStack service.

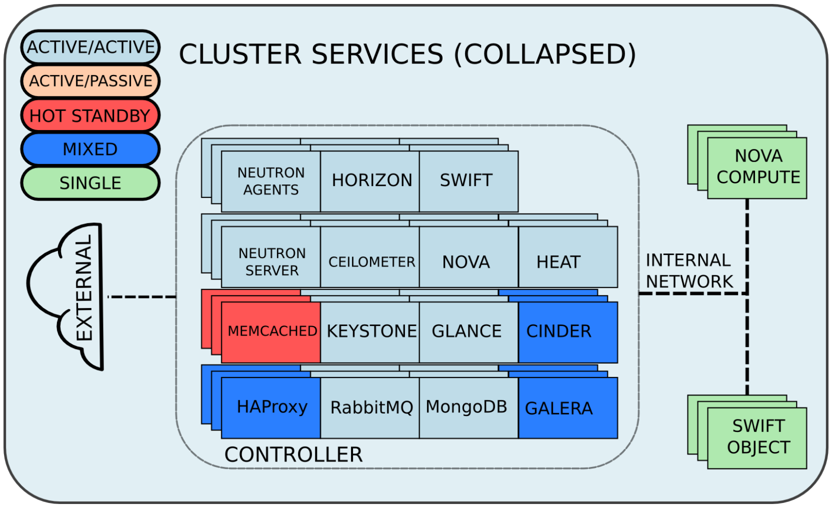

Figure 3.3: Collapsed Cluter Services

The following table lists service mode by service.

Table 2. Core Service Cluster Modes and Description

Service	Mode	Description
Ceilometer	Active/active	Measures usage of core OpenStack components. It is used within `Heat` to trigger application autoscaling.
Cinder	Mixed	Provides persistent block storage to virtual machines. All services are active/active except `cinder-volume` runs active/passive to prevent a potential race condition.
Glance	Active/active	Discovers, catalogs, and retrieves virtual machine images.
Horizon	Active/active	Web management interface runs via `httpd` in active/active mode.
Keystone	Active/active	Common OpenStack authentication system runs in `httpd`.
Neutron server	Active/active	`Neutron` allows users to define and join networks on demand.
Neutron agents	Active/active	Support Layer 2 and 3 communication plus numerous virtual networking technologies including `ML2` and `Open vSwitch`.
Nova	Active/active	Provides compute capabilities to deploy and run virtual machine instances.
Swift proxy server	Active/active	Routes data requests to the appropriate `Swift` ACO server.

Cluster Modes for Supporting Services

The majority of the core OpenStack services run in active/active mode. The same is true for the supporting services, although several of them field connection requests directly from clients rather than `HAproxy`.

The following table lists the cluster mode for the non-core OpenStack services.

Table 3. Supporting Service Cluster Modes and Description

Service	Mode	Description
Replicated state database	Active/active	`Galera` replicates databases to decrease client latency and prevent lost transactions. `Galera` runs in active/active mode but connections are only sent to one active node at a time to avoid lock contention.
Database cache	Hot standby	Memory caching system. `HAproxy` does not manage `memcached` connections because replicated access is still experimental.
Message bus	Active/active	`AMQP` message bus coordinates job execution and ensures reliable delivery. Not handled by `HAproxy`. Clients have a full list of `RabbitMQ` hosts.
NoSQL database	Active/active	NoSQL database `mongodb` supports `Ceilometer` and `Heat`. Not managed by `HAproxy`. `Ceilometer` servers have a full list of `mongodb` hosts.
Load Balancer	Active/active	The load balancer `HAproxy` runs in active/activde mode but is accessed via a set of active/passive virtual IP addresses.

Compute Node and Swift ACO Clustering

RHEL-OSP installs compute nodes and `Swift` storage servers as single-node clusters in order to monitor their health and that of the services running on them.

In the event that a compute node fails, `Pacemaker` restarts compute node services in the following order:

1. `neutron-ovs-agent`
2. `ceilometer-compute`
3. `nova-compute`

In the event that a `Swift` ACO node fails, `Pacemaker` restarts `Swift` services in the following order:

1. `swift-fs`

2. `swift-object`
3. `swift-container`
4. `swift-account`

If a service fails to start, the node where the service is running will be fenced in order to guarantee data integrity.

> This article contains more information regarding HA clustering and RHEL-OSP: Understanding RHEL OpenStack Platform High Availability

Ceph Storage Integration

Red Hat Ceph Storage 1.3 is a distributed data object store designed for performance, reliability, and scalability. RHEL-OSP 7 director can deploy an integrated Ceph cluster in the overcloud. The integrated Ceph cluster acts as a storage virtualization layer for `Glance` images, `Cinder` volumes, and `Nova` ephemeral storage. Figure 3.4 Ceph Integration depicts RHEL-OSP 7 director Ceph cluster integration from a high level. The Ceph cluster consists of two types of daemons: Ceph OSD and Ceph Monitor. The **Ceph OSD Daemon** stores data in pools striped across one or more disks. Ceph OSDs also replicate, rebalance, and recover data, and report data usage.

The **Ceph Monitor** maintains a master copy of the Ceph storage map and the current state of the storage cluster. Ceph clients consult the Ceph monitor to receive the latest copy of the storage map then communicate directly with the primary data-owning OSD.

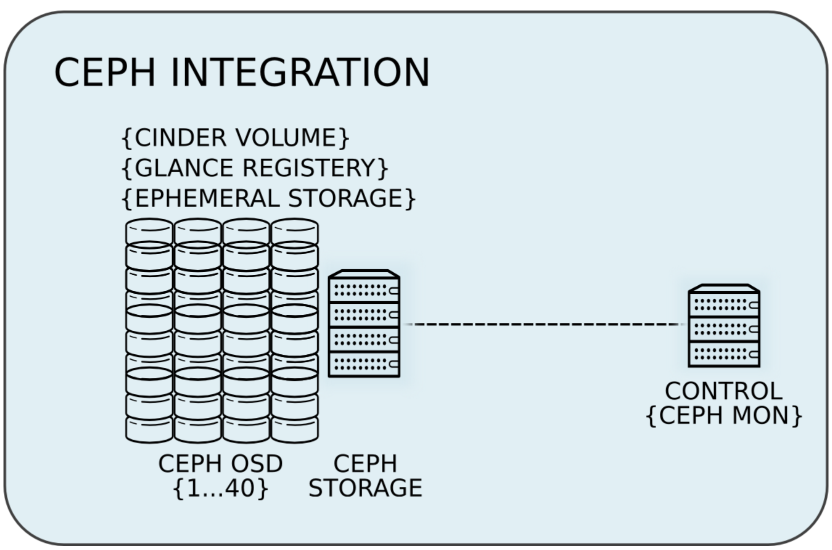

Figure 3.4: Ceph Integration

RHEL-OSP director can install a Ceph cluster with one or more OSD servers. By default the OSD server will use free space on its primary disk for the OSD storage device. Additional OSDs can be configured through Puppet customization prior to deploying the overcloud. Ceph performance scales with the number of OSD disks. The Ceph monitor is installed on the controller nodes whenever a Ceph storage role is deployed in the overcloud.

This reference architecture includes a 4-node Ceph cluster. Each node has 10 OSD disks (40 total). The OSDs in the reference architecture store `Glance` images, host `Cinder` volumes, and provide `Nova` instances with ephemeral storage.

Consult Ceph documentation for more information on Red Hat Ceph Storage 1.3. This reference architecture details how to install and run Ceph with standalone versions of Red Hat Enterprise Linux OpenStack Platform.

Reference Architecture Configuration Details

This section of the paper discusses the reference architecture use case. It includes an overview of the objective and workflow. This section also describes the test environment used to execute the use case in the Red Hat Systems Engineering lab.

Objective

This use case provides a comprehensive example for deploying an Red Hat JBoss Enterprise Application Platform 6 microservices architecture on a high availability Red Hat Enterprise Linux OpenStack Platform 7 cloud using RHEL-OSP director. The Red Hat Systems Engineering team validated all commands on bare metal servers using generally available software. The use case highlights many of RHEL-OSP director's features including:

- High Availability
- Network isolation
- Advanced Profile Matching
- Ceph integration
- Ceph customization
- Satellite subscription

The use case concludes with instructions for installing the EAP 6 microservices architecture via `Heat`. The microservices architecture demonstrates OpenStack's ability to deploy and run a complex application typical to a production cloud. The microservices architecture in this example is a multi-tier shopping cart that includes a web presentation layer, product and customer databases, and sales, billing, and product microservices.

Workflow

Figure 4.1 Reference Architecture Workflow depicts a high-level overview of the use case workflow.

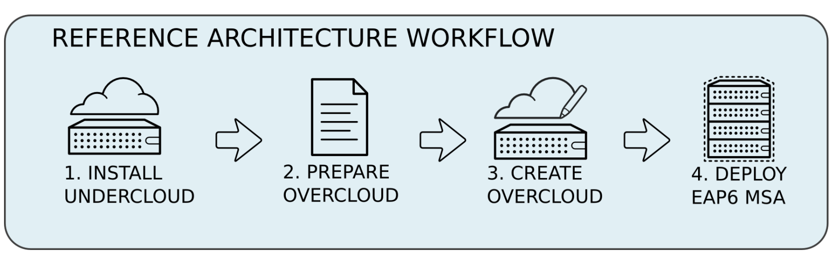

Figure 4.1 Reference Architecture Workflow

The use case is divided into the following steps:

1. **Install the undercloud**:

   a. Download the RHEL-OSP deployment package.
   b. Deploy the undercloud baremetal server.

2. **Prepare the overcloud**:
   a. Import overcloud disk images.
   b. Discover baremetal servers for overcloud deployment.
   c. Match the servers to hardware profiles.
   d. Customize the Ceph OSDs.
   e. Define the network isolation configuration.

3. **Create the overcloud**:
   a. Deploy the overcloud via `Heat` templates.
   b. Configure HA fencing devices.
   c. Test the overcloud deployment.

4. **Deploy the EAP 6 MSA**:
   a. Configure the tenant
   b. Deploy EAP 6 microservices application via `Heat` templates
   c. Test the application

Conceptual Diagram of the Solution Stack

Figure 4.2 Reference Architecture Diagram depicts the deployed solution stack.

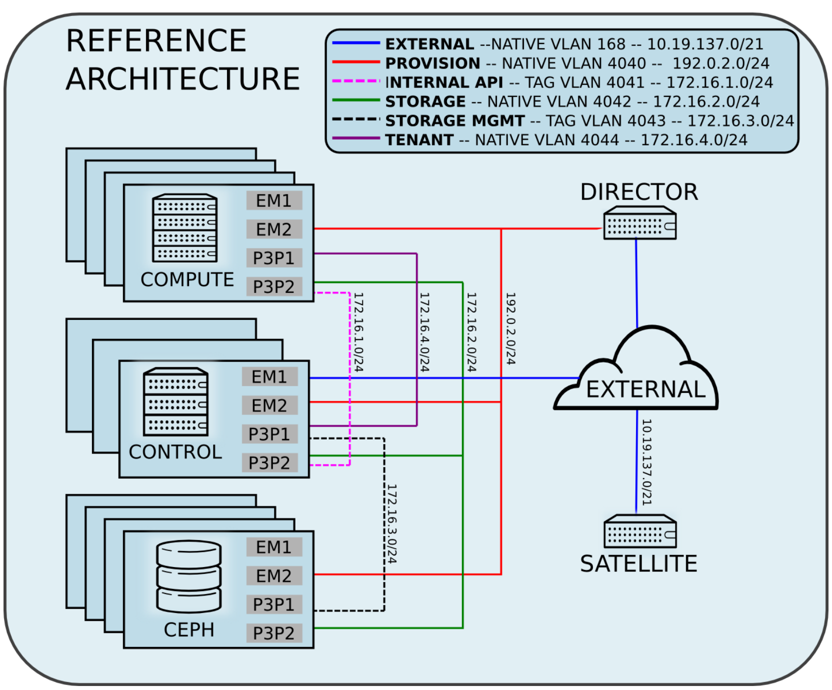

Figure 4.2: Reference Architecture Diagram

The Network Topology section of this paper describes the networking components in detail.

Server Roles

As depicted in Figure 4.2 Reference Architecture Diagram, the use case requires 12 bare metal servers deployed with the following roles:

- 1 undercloud server
- 3 cloud controllers
- 4 compute nodes
- 4 Ceph storage servers

Servers are assigned to roles based on their hardware characteristics.

Table 4. Server hardware by role

Role	Count	Model
Undercloud	1	Dell PowerEdge R720xd
Cloud controller	3	Dell PowerEdge M520
Compute node	4	Dell PowerEdge M520
Ceph storage server	4	Dell PowerEdge R510

Appendix C Hardware specifications lists hardware specifics for each server model.

Network Topology

Figure 4.2 Reference Architecture Diagram details and describes the network topology of this reference architecture.

Each server has two Gigabit interfaces (nic1:2) and two 10-Gigabit interfaces (nic3:4) for network isolation to segment OpenStack communication by type.

The following network traffic types are isolated:

- Provisioning
- Internal API
- Storage
- Storage Management
- Tenant
- External

There are six isolated networks but only four physical interfaces. Two networks are isolated on each physical 10 Gb interface using a combination of tagged and native VLANs.

The RHEL-OSP 7 network isolation feature supports bonded interfaces. Limitations in the Systems Engineering lab precluded the use of bonded interfaces in this reference architecture. Bonded interfaces are recommended for production deployments.

Table 5. Network isolation

Role	Interface	VLAN ID	Network	VLAN Type	CIDR
Undercloud	nic1	168	External	Native	10.19.137.0/21
	nic2	4040	Provisioning	Native	192.0.2.0/24
Control	nic1	168	External	Native	10.19.137.0/21
	nic2	4040	Provisioning	Native	192.0.2.0/24
	nic3	4043	Storage Mgmt	Tagged	172.16.3.0/24
	nic3	4044	Tenant	Native	172.16.4.0/24
	nic4	4041	Internal API	Tagged	172.16.1.0/24
	nic4	4042	Storage	Native	172.16.2.0/24
Compute	nic2	4040	Provisioning	Native	192.0.2.0/24
	nic3	4044	Tenant	Native	172.16.4.0/24
	nic4	4041	Internal API	Tagged	172.16.1.0/24
	nic4	4042	Storage	Native	172.16.2.0/24
Ceph storage	nic2	4040	Provisioning	Native	192.0.2.0/24
	nic3	4043	Storage Mgmt	Tagged	172.16.3.0/24
	nic4	4042	Storage	Native	172.16.2.0/24

All switch ports must be added to their respective VLANs prior to deploying the overcloud.

Deciding how to isolate networks is a crucial decision when designing for performance and scalability. There is no one-size-fits-all approach. Hardware constraints and workload characteristics must dictate this design decision. This paper shares an approach to using cloud benchmarks to guide RHEL-OSP design decisions for performance and scaling.

A complete archive of the configuration files and supporting scripts used in this reference architecture is available at https://access.redhat.com/node/1610453/40/0.

Install the undercloud

This section lists the steps to install and configure Red Hat Enterprise Linux OpenStack Platform 7 with RHEL-OSP director in the Red Hat Systems Engineering lab.

Prepare the undercloud server

The following steps are to be performed within the undercloud server as the `root` user unless otherwise specified.

1. Install the operating system.

```
# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.1 (Maipo)
```

2. Set the hostname using the `hostnamectl` command.

```
# hostnamectl set-hostname rhos0.osplocal
# hostnamectl set-hostname --transient rhos0.osplocal
# export HOSTNAME=rhos0.osplocal
# hostname
rhos0.osplocal
```

3. Register the system with `subscription-manager`.

```
# subscription-manager register --org syseng --activationkey OSP7-undercloud
The system has been registered with ID:
84e0fb33-24b0-4a1d-968e-e80352daa4f6
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status:       Subscribed
```

In this example the system is registered to a satellite server via an organization activation key. The product documentation includes instructions for registering directly via `subscription-manager`. The Red Hat Satellite 6.0 Provisioning Guide includes instructions for creating an organization activation key.

4. List active repositories.

```
# yum repolist
Loaded plugins: langpacks, product-id, rhnplugin, subscription-manager
This system is receiving updates from RHN Classic or Red Hat
Satellite.
repo id                                                     repo name status
rhel-7-server-extras-rpms/x86_64                            Red Hat Enterprise Linux 7 Server - Extras
(RPMs) 89
rhel-7-server-openstack-7.0-rpms/7Server/x86_64             Red Hat OpenStack 7.0 for RHEL 7 (RPMs)
497
rhel-7-server-optional-rpms/7Server/x86_64                  Red Hat Enterprise Linux 7 Server -
Optional (RPMs) 5,674
rhel-7-server-rpms/7Server/x86_64                           Red Hat Enterprise Linux 7 Server (RPMs)
7,392
rhel-x86_64-server-7                                        Red Hat Enterprise Linux Server (v. 7 for
64-bit x86_64) 7,424
repolist: 21,076
```

Appendix D Required channels lists the required channels.

5. Create the `stack` user. Use this account for all deployment tasks.

```
# useradd stack
# echo 'stack:password' | chpasswd
# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
stack ALL=(root) NOPASSWD:ALL
# chmod 0440 /etc/sudoers.d/stack
# id stack
uid=1000(stack) gid=1000(stack) groups=1000(stack)
```

Deploy the Control Plane

1. Switch to the `stack` user account.

```
# su - stack
$ id
uid=1000(stack) gid=1000(stack) groups=1000(stack) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
```

2. Install the RHEL-OSP director plugin.

```
$ sudo yum install -y -q python-rdomanager-oscplugin
$ sudo rpm -q python-rdomanager-oscplugin
python-rdomanager-oscplugin-0.0.10-8.el7ost.noarch
```

3. Create the *undercloud.conf*. This file contains configuration data for the undercloud installation.

```
[DEFAULT]

image_path = .
local_ip = 192.0.2.1/24
local_interface = eno4
masquerade_network = 192.0.2.0/24
dhcp_start = 192.0.2.5
dhcp_end = 192.0.2.24
network_cidr = 192.0.2.0/24
network_gateway = 192.0.2.1
discovery_interface = br-ctlplane
discovery_iprange = 192.0.2.100,192.0.2.120
discovery_runbench = false
undercloud_debug = true

[auth]
```

`eno4` is the provisioning network interface. Blank passwords are auto-generated by the installer. Accept `br-ctlplane` as the default discovery interface.

`discovery_runbench` is set to *false*. When enabled, the `Ironic` instances are booted to a RAMdisk image that runs a series of benchmark tests and reports outliers. This is beyond the scope of this reference architecture.

> Leaving the passwords undefined in the *undercloud.conf* will generate random passwords during installation.

> Installing with SSL support is beyond the scope of this reference architecture. For details on how to use SSL support for the undercloud see the product documentation or Appendix E Deploying undercloud with SSL.

4. Install the undercloud.

```
$ openstack undercloud install | tee uc.out

... [Output Abbreviated] ...
#######
instack-install-undercloud complete.

The file containing this installation's passwords is at
/home/stack/undercloud-passwords.conf.

There is also a stackrc file at /home/stack/stackrc.

These files are needed to interact with the OpenStack services, and should be secured.
#######
```

5. Source *stackrc* to set environment variables for interacting with the undercloud.

```
$ source stackrc
$ env | grep OS_
OS_PASSWORD=7f1dbeead29fe7b1ca96fcf4bec20efb1717f6db
OS_AUTH_URL=http://192.0.2.1:5000/v2.0
OS_USERNAME=admin
OS_TENANT_NAME=admin
OS_NO_CACHE=True
```

6. Verify all services are active.

> This command output was truncated for brevity. Verify all services are *active*. Appendix F Undercloud Service List lists the OpenStack services running on the undercloud.

```
$ openstack-service status
neutron-dhcp-agent (pid 16458) is active
neutron-openvswitch-agent (pid 17750) is active
neutron-server (pid 16517) is active
openstack-ceilometer-alarm-evaluator (pid 16101) is active
openstack-ceilometer-alarm-notifier (pid 16033) is active
openstack-ceilometer-api (pid 16068) is active
openstack-ceilometer-central (pid 15998) is active
openstack-ceilometer-collector (pid 15965) is active
openstack-ceilometer-notification (pid 15932) is active
...
```

Deploy the overcloud

This section describes steps for deploying the overcloud.

Create the images

1. Download and extract the RHEL-OSP 7 discovery, deployment, and overcloud images.

   Download the images from:
   https://access.redhat.com/downloads/content/191/ver=7/rhel---7/7/x86_64/product-downloads

```
$ mkdir images
$ cd images
$ cp ../.tar .*
$ ls
overcloud-full-7.1.0-39.tar discovery-ramdisk-7.1.0-39.tar deploy-ramdisk-ironic-7.1.0-39.tar
```

2. Extract the images from the tar archives.

```
$ tar xf deploy-ramdisk-ironic-7.1.0-39.tar
$ tar xf discovery-ramdisk-7.1.0-39.tar
$ tar xf overcloud-full-7.1.0-39.tar
$ ls
deploy-ramdisk-ironic-7.1.0-39.tar  discovery-ramdisk-7.1.0-39.tar overcloud-full-7.1.0-39.tar
overcloud-full.vmlinuz
deploy-ramdisk-ironic.initramfs     discovery-ramdisk.initramfs overcloud-full.initrd
deploy-ramdisk-ironic.kernel        discovery-ramdisk.kernel overcloud-full.qcow2
```

3. Upload the images to `Glance`.

```
$ openstack overcloud image upload
```

4. List the images.

```
$ openstack image list
+--------------------------------------+------------------------+
| ID                                   | Name                   |
+--------------------------------------+------------------------+
| 179a49cb-cda8-410f-b78d-0b8d31df59bf | bm-deploy-ramdisk      |
| 4266cce9-92f7-4c4f-85b0-908271b95241 | bm-deploy-kernel       |
| 841ba92b-2183-45c7-8779-f0105471323c | overcloud-full         |
| b17decc0-72f5-48ef-9ad1-85d371a3e0f8 | overcloud-full-initrd  |
| c55c1359-c9a7-40a7-983f-d4d7610954bb | overcloud-full-vmlinuz |
+--------------------------------------+------------------------+
```

Register and introspect the nodes.

1. Create the host definition file. *openstack-ironic-discoverd* uses this file to discover nodes and populate the `Ironic` database.

```
{
  "nodes": [
    {
      "pm_password": "PASSWORD",
      "pm_type": "pxe_ipmitool",
      "mac": [
        "d4:ae:52:b2:20:d2"
      ],
      "cpu": "24",
      "memory": "49152",
      "disk": "500",
      "arch": "x86_64",
      "pm_user": "root",
      "pm_addr": "10.19.143.153"
    },
    ...
```

mac is the MAC address of the provisioning interface. The **pm_** entries refer to the hardware management interface.

> The example below is truncated for brevity.

2. Import the node definitions to the `Ironic` database.

```
$ openstack baremetal import --json ~/instackenv.json
$ openstack baremetal list
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
| UUID                                 | Name | Instance UUID | Power State | Provision State |
Maintenance |
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
| 1adc6792-0bd6-4bd2-b8fc-4d9867d74597 | None | None          | power off   | available       | False
|
| 382ab2a5-b5c0-4017-b59f-82eee0fb9864 | None | None          | power off   | available       | False
|
| 84efb518-15e6-45c7-8f6a-56a5097c0b85 | None | None          | power off   | available       | False
|
| 15ca1ded-0914-469f-af63-3340f91bc56a | None | None          | power off   | available       | False
|
| 8e6c96ad-c039-498d-8bd2-61a489bbae87 | None | None          | power off   | available       | False
|
| 84e34eb3-2352-49c8-8748-8bc6b6185587 | None | None          | power off   | available       | False
|
| abb19869-b92f-42b3-9db1-f69f6ee00f2e | None | None          | power off   | available       | False
|
| db878d37-5b7a-4140-8809-1b50d4ddbec4 | None | None          | power off   | available       | False
|
| d472af62-5547-4f9a-8fbb-fc8556eb4110 | None | None          | power off   | available       | False
|
| c8400dc0-4246-44ee-a406-9362381d7ce1 | None | None          | power off   | available       | False
|
| 0c7af223-1a7d-43cd-a0ff-19226872e09c | None | None          | power off   | available       | False
|
| 5f52affb-cfe2-49dc-aa89-b57d99e5372a | None | None          | power off   | available       | False
|
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
```

3. Assign a kernel and ramdisk to the nodes

```
$ openstack baremetal configure boot
```

4. Introspect the nodes to discover their hardware attributes.

```
$ openstack baremetal introspection bulk start
...
```

Bulk introspection time will vary based on node count and boot time. For this reference architecture bulk introspection lasted approximately 3 minutes per node.

5. Use **journalctl** to view introspection progress in a separate terminal.

```
$ sudo journalctl -l -u openstack-ironic-discoverd -u
openstack-ironic-discoverd-dnsmasq -u openstack-ironic-conductor |
grep -i finished
Aug 28 09:23:46 rhos0.osplocal ironic-discoverd[22863]:
INFO:ironic_discoverd.process:Introspection finished successfully for node 1adc6792-0bd6-4bd2-b8fc-
4d9867d74597
Aug 28 09:24:53 rhos0.osplocal ironic-discoverd[22863]:
INFO:ironic_discoverd.process:Introspection finished successfully for node 84efb518-15e6-45c7-8f6a-
56a5097c0b85
```

6. Verify nodes completed introspection without errors.

```
$ openstack baremetal introspection bulk status
+--------------------------------------+----------+-------+
| Node UUID                            | Finished | Error |
+--------------------------------------+----------+-------+
| 1adc6792-0bd6-4bd2-b8fc-4d9867d74597 | True     | None  |
| 382ab2a5-b5c0-4017-b59f-82eee0fb9864 | True     | None  |
| 84efb518-15e6-45c7-8f6a-56a5097c0b85 | True     | None  |
| 15ca1ded-0914-469f-af63-3340f91bc56a | True     | None  |
| 8e6c96ad-c039-498d-8bd2-61a489bbae87 | True     | None  |
| 84e34eb3-2352-49c8-8748-8bc6b6185587 | True     | None  |
| abb19869-b92f-42b3-9db1-f69f6ee00f2e | True     | None  |
| db878d37-5b7a-4140-8809-1b50d4ddbec4 | True     | None  |
| d472af62-5547-4f9a-8fbb-fc8556eb4110 | True     | None  |
| c8400dc0-4246-44ee-a406-9362381d7ce1 | True     | None  |
| 0c7af223-1a7d-43cd-a0ff-19226872e09c | True     | None  |
| 5f52affb-cfe2-49dc-aa89-b57d99e5372a | True     | None  |
+--------------------------------------+----------+-------+-
```

Configure hardware profiles

1. Create the default flavor for baremetal deployments.

```
$ openstack flavor create --id auto --ram 4096 --disk 40 --vcpus 1 baremetal
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 40                                   |
| id                         | e3f8358d-983f-4383-8379-50cbbf5bf970 |
| name                       | baremetal                            |
| os-flavor-access:is_public | True                                 |
| ram                        | 4096                                 |
| rxtx_factor                | 1.0                                  |
| swap                       |                                      |
| vcpus                      | 1                                    |
+----------------------------+--------------------------------------+
```

2. Set properties for the baremetal flavor.

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local"
baremetal
+----------------------------+-----------------------------------------------------+
| Field                      | Value                                               |
+----------------------------+-----------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                               |
| OS-FLV-EXT-DATA:ephemeral  | 0                                                   |
| disk                       | 40                                                  |
| id                         | e3f8358d-983f-4383-8379-50cbbf5bf970                |
| name                       | baremetal                                           |
| os-flavor-access:is_public | True                                                |
| properties                 | capabilities:boot_option='local', cpu_arch='x86_64' |
| ram                        | 4096                                                |
| rxtx_factor                | 1.0                                                 |
| swap                       |                                                     |
| vcpus                      | 1                                                   |
+----------------------------+-----------------------------------------------------+
```

3. Install *ahc-tools*. This package contains reporting and matching tools for automatic health checks.

```
$ sudo yum install -y -q ahc-tools
$ sudo rpm -qa | grep ahc-tools
ahc-tools-0.1.1-6.el7ost.noarch
```

4. Create the AHC configuration file.

```
$ sudo cp /etc/ironic-discoverd/discoverd.conf /etc/ahc-tools/ahc-tools.conf
$ sudo sed -i 's/\[discoverd/\[ironic/' /etc/ahc-tools/ahc-tools.conf
$ sudo chmod 0600 /etc/ahc-tools/ahc-tools.conf
```

5. View */etc/ahc-tools/ahc-tools.conf*.

```
$ sudo cat /etc/ahc-tools/ahc-tools.conf
[ironic]
debug = false
os_auth_url = http://192.0.2.1:5000/v2.0
identity_uri = http://192.0.2.1:35357
os_username = ironic
os_password = d5ba7515326d740725ea74bf0aec65fb079c0e19
os_tenant_name = service
dnsmasq_interface = br-ctlplane
database = /var/lib/ironic-discoverd/discoverd.sqlite
ramdisk_logs_dir = /var/log/ironic-discoverd/ramdisk/
processing_hooks =
ramdisk_error,root_device_hint,scheduler,validate_interfaces,edeploy
enable_setting_ipmi_credentials = true
keep_ports = added
ironic_retry_attempts = 6
ironic_retry_period = 10

[swift]
username = ironic
password = d5ba7515326d740725ea74bf0aec65fb079c0e19
tenant_name = service
os_auth_url = http://192.0.2.1:5000/v2.0
```

6. Create the AHC spec files. These files contain matching rules that determine which profile gets assigned to each node.

> Servers are matched to profiles by the order they are listed in this file.

```
$ for i in $(ls /etc/ahc-tools/edeploy/{*.specs,state}); do echo $i && cat $i; done
/etc/ahc-tools/edeploy/ceph.specs
[
  ('disk', '$disk', 'size', 'gt(400)'),
]
/etc/ahc-tools/edeploy/compute.specs
[
 ('cpu', '$cpu', 'cores', '8'),
  ('memory', 'total', 'size', 'ge(64000000000)'),
]
/etc/ahc-tools/edeploy/control.specs
[
 ('cpu', '$cpu', 'cores', '8'),
('disk', '$disk', 'size', 'gt(100)'),
 ('memory', 'total', 'size', 'ge(64000000000)'),
 ]
/etc/ahc-tools/edeploy/state
[('ceph', '4'), ('control', '3'), ('compute', '*')]
```

This configuration defines:

- Minimum disk size of 400 GB for Ceph servers
- 8 cores per CPU and 64 GB RAM for compute nodes
- 8 cores per CPU, minimum 100 GB disk size and 64 GB RAM for controllers
- The state file specifies that AHC should match 3 controllers, 4 Ceph storage servers, and the remainder as compute nodes.

View Appendix A of the eDeploy User's Guide for an exhaustive list of the hardware components and settings that can be matched in a specification file.

7. This loop creates a hardware profile for each node type defined in the state file.

```
$ for i in ceph control compute; do openstack flavor create --id auto --ram 4096 --disk 40 --vcpus 1 $i;
openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local"
--property "capabilities:profile"="$i" $i; done
...
$ openstack flavor list
+--------------------------------------+-----------+------+------+-----------+-------+-----------+
| ID                                   | Name      |  RAM | Disk | Ephemeral | VCPUs | Is Public |
+--------------------------------------+-----------+------+------+-----------+-------+-----------+
| 3bd3c59f-16c4-4090-94b5-0d90e1f951fa | compute   | 4096 |   40 | 0         |     1 | True      |
| 9a9c0a68-550a-4736-9b6d-f4aa1cc68a1f | ceph      | 4096 |   40 | 0         |     1 | True      |
| a3d47c7e-04dc-47e3-8fca-b19ea31d0ed2 | control   | 4096 |   40 | 0         |     1 | True      |
| e3f8358d-983f-4383-8379-50cbbf5bf970 | baremetal | 4096 |   40 | 0         |     1 | True      |
+--------------------------------------+-----------+------+------+-----------+-------+-----------+
```

8. Assign `Ironic` nodes to their corresponding profiles.

```
$ sudo ahc-match
```

9. View the profile assigned to each node.

```
$ for i in $(ironic node-list | awk ' /available/ { print $2 } ');
do ironic node-show $i | grep capabilities; done
|                         | u'cpus': u'24', u'capabilities':u'profile:ceph,boot_option:local'}   |
|                         | u'cpus': u'24', u'capabilities':u'profile:ceph,boot_option:local'}   |
|                         | u'cpus': u'24', u'capabilities':u'profile:ceph,boot_option:local'}   |
|                         | u'cpus': u'24', u'capabilities':u'profile:ceph,boot_option:local'}   |
|                         | u'cpus': u'32', u'capabilities':u'profile:control,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:control,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:control,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:compute,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:compute,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:compute,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:compute,boot_option:local'}  |
|                         | u'cpus': u'32', u'capabilities':u'profile:compute,boot_option:local'}  |
```

In this example, the 4 R510 servers are assigned to the Ceph profile, 3 M520 servers are assigned to the control profile, and the remainder are assigned to the compute profile.

Configure network isolation

Network isolation assigns specific types of OpenStack network traffic to specific interfaces or bonds. This section describes how network isolation was configured for this reference architecture. Configure network isolation by defining networks in environment files. Pass the environment files to `Heat`.

The network isolation environment files used in this section produce the network described in Reference Architecture Diagram.

1. Define isolated networks in *network-environment.yaml*.

```
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/nic-configs/controller.yaml
  OS::TripleO::CephStorage::Net::SoftwareConfig: /home/stack/nic-configs/ceph-storage.yaml
  #OS::TripleO::ObjectStorage::Net::SoftwareConfig: /home/stack/nic-configs/swift-storage.yaml
  #OS::TripleO::BlockStorage::Net::SoftwareConfig: /home/stack/nic-configs/cinder-storage.yaml

parameter_defaults:
  ControlPlaneSubnetCidr: "24"
  ControlPlaneDefaultRoute: 192.0.2.1
  EC2MetadataIp: 192.0.2.1
  DnsServers: ['10.19.143.247','10.11.5.19']
  NeutronExternalNetworkBridge: "br-ex"
  InternalApiNetCidr: 172.16.1.0/24
  StorageNetCidr: 172.16.2.0/24
  StorageMgmtNetCidr: 172.16.3.0/24
  TenantNetCidr: 172.16.4.0/24
  ExternalNetCidr: 10.19.136.0/21
  InternalApiAllocationPools: [{'start': '172.16.1.10', 'end': '172.16.1.100'}]
  StorageAllocationPools: [{'start': '172.16.2.10', 'end': '172.16.2.200'}]
  StorageMgmtAllocationPools: [{'start': '172.16.3.10', 'end': '172.16.3.200'}]
  TenantAllocationPools: [{'start': '172.16.4.10', 'end': '172.16.4.200'}]
  ExternalAllocationPools: [{'start': '10.19.137.121', 'end': '10.19.137.151'}]
  InternalApiNetworkVlanID: 4041
  StorageNetworkVlanID: 4042
  StorageMgmtNetworkVlanID: 4043
  TenantNetworkVlanID: 4044
  ExternalNetworkVlanID: 168
  ExternalInterfaceDefaultRoute: 10.19.143.254
  BondInterfaceOvsOptions:
      "bond_mode=balance-tcp lacp=active other-config:lacp-fallback-ab=true"
```

The *resource_registery* section defines role-specific configuration. In this example, the paths to *cinder-storage.yaml* and *swift-storage.yaml* are commented out because resources of these types are not used in this reference architecture.

The *parameter_defaults* section defines default parameters used across the resource registry. These include CIDRs, VLAN IDs, and IP allocation pools for each network, as well as the external network bridge created by `Open vSwitch`.

The parameters defined in this file match the network configuration used in the reference architecture.

In most cases *NeutronExternalNetworkBridge* would be set to "''" in order to support multiple floating IP VLANs or physical interfaces. In this case there was only one floating IP network on the native VLAN of bridge *br-ex*, so the bridge was specified directly for performance reasons.

> RHEL-OSP director 7.1 includes additional parameter defaults for assigning static IP addresses to the provisioning network. Static IP addresses ensure the provisioning network continues to function even if the servers' DHCP leases expire and are not renewed. The following text block contains example parameters that should be customized to match the environment.

```
parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ControlPlaneSubnetCidr:          # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max) to add to resolv.conf.
    type: json
  EC2MetadataIp:                   # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
  ControlPlaneDefaultRoute:        # Gateway router for the provisioning network (or Undercloud IP)
    default: METADATA_IP_ADDR      # default to the undercloud
    description: The subnet CIDR of the control plane network.
    type: string
```

2. Create the *nic-configs* files to define network configuration for each interface by server role.

```
$ mkdir ~/nic-configs
$ ls ~/nic-configs
ceph-storage.yaml  cinder-storage.yaml  compute.yaml  controller.yaml swift-storage.yaml
```

Complete examples of each network configuration file are in Appendix I NIC Confguration Files. These examples include the updated syntax for RHEL-OSP director 7.1 to use static IP addresses on the provisioning network as well as:

1. Enforcing that the provisioning network will not use DHCP
2. Specifying the DNS servers with the new DnsServer parameter
3. Making the external network the default network for the controller nodes
4. Using the provisioning network as the default gateway for the compute and Ceph storage nodes

> `Swift` and `Cinder` servers are not used in this reference architecture. Their files are included for completeness but not called by the installer.

Customize Ceph Storage

Like network isolation, Ceph is customized by passing `Heat` additional environment files. The customization produce the Ceph cluster depicted in Figure 3.4 Ceph integration.

In this reference architecture ten SAS disks in each R510 are configured as OSD drives. The journal for each OSD is created as a separate partition on the OSD drive. This is the recommended journal configuration for Ceph OSDs when SSD drives are not used.

1. Configure Ceph OSD disks as single-drive RAID 0 virtual disks for best performance. Ceph data is protected through replication across OSDs so RAID is not recommended.
2. Initialize the virtual disks to remove all partition and MBR data.
3. Create a *templates* directory for `Heat` template customization.

    ```
    $ mkdir ~/templates
    $ cp -rp /usr/share/openstack-tripleo-heat-templates/ ~/templates
    ```

4. Edit *~/templates/openstack-tripleo-heat-templates/puppet/hieradata/ceph.yaml* to include the Ceph customizations. This example includes the additional OSDs accepting the Puppet defaults for journaling.

```
ceph::profile::params::osd_journal_size: 1024
ceph::profile::params::osd_pool_default_pg_num: 128
ceph::profile::params::osd_pool_default_pgp_num: 128
ceph::profile::params::osd_pool_default_size: 3
ceph::profile::params::osd_pool_default_min_size: 1
#ceph::profile::params::osds: {/srv/data: {}}
ceph::profile::params::osds:
  '/dev/sdb': {}
  '/dev/sdc': {}
  '/dev/sdd': {}
  '/dev/sde': {}
  '/dev/sdf': {}
  '/dev/sdg': {}
  '/dev/sdh': {}
  '/dev/sdi': {}
  '/dev/sdj': {}
  '/dev/sdk': {}
ceph::profile::params::manage_repo: false
ceph::profile::params::authentication_type: cephx

ceph_pools:
  - volumes
  - vms
  - images

ceph_osd_selinux_permissive: true
```

This configuration file does not specify an OSD journal location. Omitting a custom location for the OSD journal instructs `Heat` to create a journal in the default location for each disk. The default location is a second partition on each disk.

By default Ceph creates one OSD per storage server using the remaining free space on the operating system disk. The OSD journal is configured as a 5 GB file on the disk. This configuration is only suitable for evaluation and proof of concept.

Deploy and Test the overcloud

This section describes how to deploy and test the overcloud defined in the previous section.

Deploy the overcloud servers

1. Use **ironic node-list** to verify all `Ironic` nodes are powered off, available for provisioning, and not in maintenance mode.

```
$ ironic node-list
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
| UUID                                 | Name | Instance UUID | Power State | Provision State |
Maintenance |
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
| 1adc6792-0bd6-4bd2-b8fc-4d9867d74597 | None | None          | power off   | available       | False
|
| 382ab2a5-b5c0-4017-b59f-82eee0fb9864 | None | None          | power off   | available       | False
|
| 84efb518-15e6-45c7-8f6a-56a5097c0b85 | None | None          | power off   | available       | False
|
| 15ca1ded-0914-469f-af63-3340f91bc56a | None | None          | power off   | available       | False
|
| 8e6c96ad-c039-498d-8bd2-61a489bbae87 | None | None          | power off   | available       | False
|
| 84e34eb3-2352-49c8-8748-8bc6b6185587 | None | None          | power off   | available       | False
|
| abb19869-b92f-42b3-9db1-f69f6ee00f2e | None | None          | power off   | available       | False
|
| db878d37-5b7a-4140-8809-1b50d4ddbec4 | None | None          | power off   | available       | False
|
| d472af62-5547-4f9a-8fbb-fc8556eb4110 | None | None          | power off   | available       | False
|
| c8400dc0-4246-44ee-a406-9362381d7ce1 | None | None          | power off   | available       | False
|
| 0c7af223-1a7d-43cd-a0ff-19226872e09c | None | None          | power off   | available       | False
|
| 5f52affb-cfe2-49dc-aa89-b57d99e5372a | None | None          | power off   | available       | False
|
+--------------------------------------+------+---------------+-------------+-----------------
+-------------+
```

2. Deploy the overcloud.

```
$ openstack overcloud deploy -e
/usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/network-environment.yaml --control-flavor control --compute-flavor compute \
--ceph-storage-flavor ceph --ntp-server 10.16.255.2 --control-scale 3 --compute-scale 4 \
--ceph-storage-scale 4 --block-storage-scale 0 --swift-storage-scale 0 \
-t 90 --templates /home/stack/templates/openstack-tripleo-heat-templates/ \
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \
--rhel-reg --reg-method satellite --reg-sat-url \
http://se-sat6.syseng.bos.redhat.com --reg-org syseng --reg-activation-key OSP-Overcloud
Deploying templates in the directory /home/stack/templates/openstack-tripleo-heat-templates
```

This lengthy command does the following:

- Specifies the location of *network-environment.yaml* to customize the network configurations.
- Specifies which flavors and how many control, compute, and ceph-storage nodes to instantiate.
- Specifies the location of the *storage-environment.yaml* for Ceph customization.
- Registers the overcloud servers with the lab satellite server using a predefined activation key.
- Enables tenant networking with the default VXLAN tenant networking type.

3. Watch deployment progress in a separate console window.

```
$ heat resource-list overcloud | grep CREATE_COMPLETE
| BlockStorage                      | 8565b42e-0b24-41ec-88d3-7d0d6bc18834 | OS::Heat::ResourceGroup |
CREATE_COMPLETE | 2015-08-28T16:25:53Z |
| ControlVirtualIP                  | c4926ff9-2ea7-40f1-9677-d7f26e3517db | OS::Neutron::Port |
CREATE_COMPLETE    | 2015-08-28T16:25:53Z |
| HeatAuthEncryptionKey             | overcloud-HeatAuthEncryptionKey-paa5lxc3ubon  |
OS::Heat::RandomString | CREATE_COMPLETE    | 2015-08-28T16:25:53Z |
| HorizonSecret                     | overcloud-HorizonSecret-mpgdt65yqsud          |
OS::Heat::RandomString | CREATE_COMPLETE    | 2015-08-28T16:25:53Z |
...
```

4. Run **nova-list** to view IP addresses for the overcloud servers.

```
$ nova list
...
| e50a67fa-ed75-4f39-a58f-47b51371f61d | overcloud-cephstorage-0 | ACTIVE | -          | Running     |
ctlplane=192.0.2.20 |
| e36b2f28-463c-4e01-91e0-8ed762a1c057 | overcloud-cephstorage-1 | ACTIVE | -          | Running     |
ctlplane=192.0.2.21 |
| 37c67128-8432-4330-afe7-ab3b01bdcb6e | overcloud-cephstorage-2 | ACTIVE | -          | Running     |
ctlplane=192.0.2.19 |
| 3ee07cc2-9adf-457f-94e6-705657ac3767 | overcloud-cephstorage-3 | ACTIVE | -          | Running     |
ctlplane=192.0.2.22 |
| e1f2801b-cb6e-4c55-a82a-476d0090f1d6 | overcloud-compute-0     | ACTIVE | -          | Running     |
ctlplane=192.0.2.8  |
| 17be9669-247b-434f-9ad2-8ab59740c1e9 | overcloud-compute-1     | ACTIVE | -          | Running     |
ctlplane=192.0.2.23 |
| be30827b-e3b4-4504-8afb-fe5ea42fda54 | overcloud-compute-2     | ACTIVE | -          | Running     |
ctlplane=192.0.2.7  |
| 6a2ee7e1-31b8-48da-b56b-0834ac6bf3b4 | overcloud-compute-3     | ACTIVE | -          | Running     |
ctlplane=192.0.2.24 |
| 520c5af6-fc91-4b93-bb95-93f947a7cc71 | overcloud-controller-0  | ACTIVE | -          | Running     |
ctlplane=192.0.2.9  |
| 23a2de54-e3c9-4c1d-aaff-75ef5993b7af | overcloud-controller-1  | ACTIVE | -          | Running     |
ctlplane=192.0.2.6  |
| 2afb18d3-3494-41da-951a-b72d68b4bf88 | overcloud-controller-2  | ACTIVE | -          | Running     |
ctlplane=192.0.2.10 |
```

5. Source the *overcloudrc* file to set environment variables for the overcloud.
6. Verify all `Nova` services and enabled and up.

```
$ nova service-list
...
| 3   | nova-scheduler   | overcloud-controller-0.localdomain | internal | enabled | up    | 2015-08-
28T21:56:01.000000 | -                |
| 6   | nova-scheduler   | overcloud-controller-2.localdomain | internal | enabled | up    | 2015-08-
28T21:56:03.000000 | -                |
| 9   | nova-scheduler   | overcloud-controller-1.localdomain | internal | enabled | up    | 2015-08-
28T21:56:04.000000 | -                |
| 12  | nova-consoleauth | overcloud-controller-1.localdomain | internal | enabled | up    | 2015-08-
28T21:56:03.000000 | -                |
| 15  | nova-consoleauth | overcloud-controller-2.localdomain | internal | enabled | up    | 2015-08-
28T21:56:03.000000 | -                |
| 18  | nova-consoleauth | overcloud-controller-0.localdomain | internal | enabled | up    | 2015-08-
28T21:56:04.000000 | -                |
| 21  | nova-conductor   | overcloud-controller-2.localdomain | internal | enabled | up    | 2015-08-
28T21:55:57.000000 | -                |
| 57  | nova-conductor   | overcloud-controller-0.localdomain | internal | enabled | up    | 2015-08-
28T21:55:57.000000 | -                |
| 105 | nova-conductor   | overcloud-controller-1.localdomain | internal | enabled | up    | 2015-08-
28T21:55:58.000000 | -                |
| 123 | nova-compute     | overcloud-compute-1.localdomain    | nova     | enabled | up    | 2015-08-
28T21:55:59.000000 | -                |
[ ... Output truncated ... ]
```

7. Verify all `Neutron` agents are alive and up.

```
$ neutron agent-list
...
| 2034c620-e2be-4fc3-8c7e-878125cccb46 | Open vSwitch agent | overcloud-compute-3.localdomain    | :-)
| True           | neutron-openvswitch-agent |
| 290a09bb-9878-4661-9c55-dee4c53f103c | Metadata agent     | overcloud-controller-2.localdomain | :-)
| True           | neutron-metadata-agent    |
| 369ef1fd-992a-462a-8569-128c329cf7b1 | Open vSwitch agent | overcloud-compute-2.localdomain    | :-)
| True           | neutron-openvswitch-agent |
| 42b35c58-dda0-4e55-b53f-5f7466acdac5 | Open vSwitch agent | overcloud-compute-0.localdomain    | :-)
| True           | neutron-openvswitch-agent |
| 45b4e429-1ad7-4678-aa8b-bc8afa8761ea | DHCP agent         | overcloud-controller-1.localdomain | :-)
| True           | neutron-dhcp-agent        |
| 91ff4990-6080-4fd2-98c2-b69cb5ea3d79 | L3 agent           | overcloud-controller-0.localdomain | :-)
| True           | neutron-l3-agent          |
[ ... Output truncated ... ]
```

8. **ssh** to a controller node and switch to root user. Find the controller IP address by running **nova list** on the undercloud.

```
$ ssh -l heat-admin 192.0.2.9
The authenticity of host '192.0.2.9 (192.0.2.9)' can't be established.
ECDSA key fingerprint is fe:a3:da:94:36:37:de:76:68:71:e0:70:cb:3a:00:aa.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.0.2.9' (ECDSA) to the list of known hosts.
$ sudo -i
```

9. Run **pcs status** to verify OpenStack services started correctly.

Run **pcs resource cleanup** if any of the services are not fully started.

```
# pcs status
Cluster name: tripleo_cluster
Last updated: Fri Aug 28 17:47:31 2015
Last change: Fri Aug 28 15:28:39 2015
Stack: corosync
Current DC: overcloud-controller-1 (2) - partition with quorum
Version: 1.1.12-a14efad
3 Nodes configured
112 Resources configured

Online: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]

Full list of resources:

 Clone Set: haproxy-clone [haproxy]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
ip-172.16.1.11 (ocf::heartbeat:IPaddr2): Started overcloud-controller-0
ip-10.19.137.121  (ocf::heartbeat:IPaddr2): Started overcloud-controller-1
...
```

Appendix G Overcloud Service List shows complete **pcs status** `Pacemaker` output for a deployed overcloud.

10. Verify that the provisioning network IP address is statically assigned.

```
# cat /etc/sysconfig/network-scripts/ifcfg-em2
# This file is autogenerated by os-net-config
DEVICE=em2
ONBOOT=yes
HOTPLUG=no
NM_CONTROLLED=no
BOOTPROTO=static
IPADDR=192.0.2.9
NETMASK=255.255.255.0
```

Tune Ceph storage

This section includes steps for increasing the number of Placement Groups (PGs) per pool. Ceph Placement Groups (PGs) aggregate objects within pools. PGs within a pool are distributed across OSDs for data durability and performance. By default OSP director creates 4 pools with 64 PGs and 3 replicas per pool. There are 40 OSDs which leaves 19.2 PGs per OSD. Ceph recommends at least 30 PGs per OSD.

Each pool has two properties that dictate its number of placement groups: *pg_num* (number of placement groups) and *pgp_num* (number of PGs for placement on OSD.) At the time of writing, customizing *pg_num* in *ceph.yaml* prior to deployment was not working. See BZ1252546 for details. Therefore, this reference architecture manually inceases *pg_num* and *pgp_num* to Ceph recommendations.

Figure 6.1 Ceph benchmark performance shows the relative performance impact of Ceph tuning on an IO microbenchmark.

1. **ssh** to a Ceph OSD node and switch to root user.

```
$ ssh -l heat-admin 192.0.2.20
Last login: Fri Aug 28 17:58:30 2015 from 192.0.2.1
$ sudo -i
```

2. Run **ceph -s** to verify all OSDs are up and in, pool count, and total free space.

```
# ceph -s
 cluster 7ced0d2a-4db6-11e5-86a4-90b11c56332a
 health HEALTH_WARN too few PGs per OSD (19 < min 30)
 monmap e2: 3 mons at {overcloud-controller-0=172.16.2.16:6789/0,overcloud-controller-
1=172.16.2.15:6789/0,overcloud-controller-2=172.16.2.21:6789/0}

        election epoch 6, quorum 0,1,2 overcloud-controller-1,overcloud-controller-0,overcloud-
controller-2
 osdmap e82: 40 osds: 40 up, 40 in
  pgmap v120: 256 pgs, 4 pools, 0 bytes data, 0 objects
        201 GB used, 37020 GB / 37221 GB avail
        256 active+clean
```

3. List the pools and pool stats. There are four pools configured for object storage, images, block storage, and ephemeral storage. There are 256 PGs total, 64 per pool.

```
# ceph osd lspools
0 rbd,1 images,2 volumes,3 vms,
# ceph pg stat
v120: 256 pgs: 256 active+clean; 0 bytes data, 201 GB used, 37020 GB /
37221 GB avail
```

4. View overall Ceph health.

```
# ceph health
HEALTH_WARN too few PGs per OSD (19 < min 30)
```

5. Increase per-pool *pg_num* and *pgp_num* to 256.

```
# for i in rbd images volumes vms; do
 ceph osd pool set $i pg_num 256;
 sleep 10
 ceph osd pool set $i pgp_num 256;
 sleep 10
done
set pool 0 pg_num to 256
set pool 0 pgp_num to 256
set pool 1 pg_num to 256
set pool 1 pgp_num to 256
set pool 2 pg_num to 256
set pool 2 pgp_num to 256
set pool 3 pg_num to 256
set pool 3 pgp_num to 256
```

The **sleep** statements are intended to ensure the cluster has time to complete the previous action before proceeding. If a large increase is needed increase *pg_num* in stages.

6. Re-run **ceph health** and **ceph pg stat**.

```
# ceph health
HEALTH_OK
# ceph pg stat
v180: 1024 pgs: 1024 active+clean; 0 bytes data, 201 GB used, 37020 GB
/ 37221 GB avail
```

Increase the PGs on only one Ceph node in the cluster.

Performance Impact of Ceph Tuning

This graphic illustrates the performance impact of increasing the OSD count from 4 to 40 and the PG count from 100 to 256. All performance numbers are relative to the default settings.

1. The *random read* performance improves slightly but does not benefit very much from increasing OSD or PG count. Random read performance is still limited by the average seek time on the disks.
2. Increasing the OSD count improves *sequential read* performance by more than 100% due to increased parallelism.
3. *sequential write* benefits from both OSD and PG increases and shows the largest relative improvement versus the default configuration.

These performance tests are are for illustrative purposes only and do not reflect the achievable performance of the machines on a real application workload.

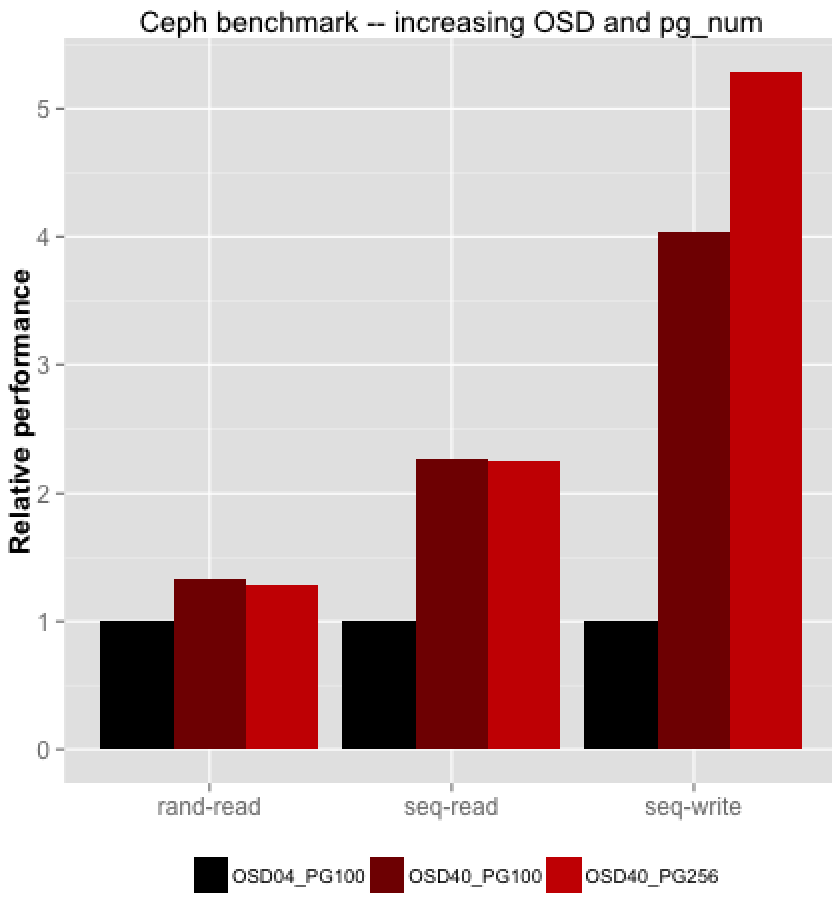

Figure 6.1: Ceph benchmark performance

Configure controller fencing

Fencing is an important concept for HA clusters. It is a method for bringing the cluster into a known state by removing members that are in an unknown state. In this reference architecture the controller IPMI interfaces act as fence devices. However, Red Hat Enterprise Linux OpenStack Platform director does not configure fencing. This section describes how the controller nodes were manually configured for fencing in this reference architecture.

Appendix H Example fencing Script shows an example script used to configure fencing in this reference architecture. This script configures each controller nodes IPMI as a fence device, constrains it so a controller cannot fence itself, and then enables all fence devices.

1. Run **configure_fence.sh**.

```
$ sh configure_fence.sh enable
Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: tripleo_cluster
 dc-version: 1.1.12-a14efad
 have-watchdog: false
 redis_REPL_INFO: overcloud-controller-1
 stonith-enabled: true
```

2. Verify fence devices are configured with **pcs status**.

```
$ ssh -l heat-admin 192.0.2.9 sudo pcs status | grep -i fence
 overcloud-controller-0-ipmi  (stonith:fence_ipmilan):  Started overcloud-controller-1
 overcloud-controller-1-ipmi (stonith:fence_ipmilan):  Started overcloud-controller-2
 overcloud-controller-2-ipmi  (stonith:fence_ipmilan):  Started overcloud-controller-0
```

Install and Configure EAP 6

This section describes the steps to install and configure an example Red Hat JBoss Enterprise Application Platform 6 application on the deployed cloud. The example application is a multi-tier web application with a shopping cart.

Red Hat JBoss Enterprise Application Platform 6 (EAP) is a fully certified Java™ EE platform for developing and deploying enterprise applications. This reference architecture documents the steps to deploy an EAP 6 application demonstrating Microservices Architecture (MSA) on RHEL-OSP 7. MSA is software architectural style that increases modularity to decrease complexity. Applications are developed from suites of small services, each running as an independent process in its own container or virtual machine. Each service has a single responsibility. The services communicate with standard lightweight protocols and APIs, such as REST over HTTP.

More information about Red Hat JBoss Enterprise Application Platform 6 can be found at http://red.ht/1NZrW0A.

The MSA application used in this reference architecture is an example of *Business-Driven* microservices. The services in the application do not communicate directly with one another. A web application aggregates and coordinates communication between the services. It acts as a perimeter between the application and the clients. By employing this presentation layer, the microservices remain independent from each other. They can be developed, scaled, and maintained independently, without leading to the complex dependency graph inherent to other MSA approaches.

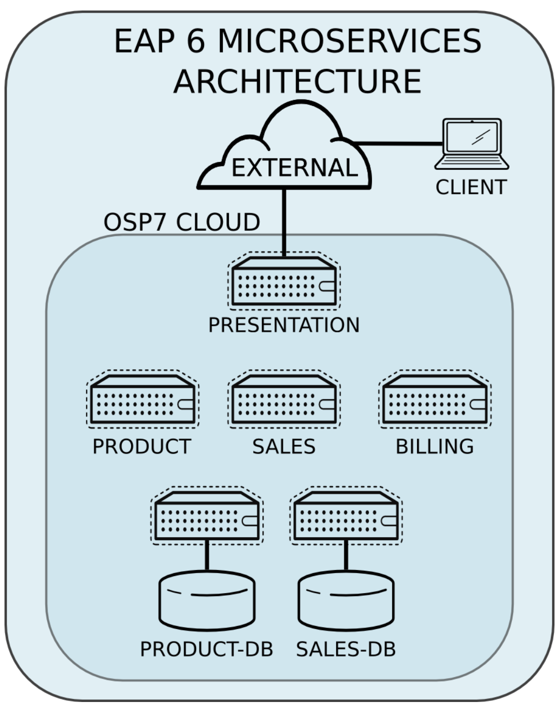

Figure 7.1: EAP6 Microservices Architecture

RHEL-OSP 7 provides a comprehensive platform for implementing, maintaining, and scaling an MSA application. Because microservices are independent, scalable services, they require a scalable platform to reach their potential. OpenStack provides a robust underlying tool set for automating service implementation, deployment, discovery, and scaling. This reference architecture demonstrates how to deploy and orchestrate an EAP6 MSA application using nested `Heat` templates. Directions for future work might include auto-scaling to multiple MSA applications via `Heat` and `Ceilometer` and load balancing between them via LBaaS.

Figure 7.1 EAP6 Microservices Architecture depicts the MSA application deployed in this reference architecture.

More information on deploying a Red Hat JBoss Enterprise Application Platform 6 6 MSA application can be found at Microservice Architecture: Building Microservices with JBOSS EAP 6.

Create the test environment

The MSA application does not exist in a vacuum. This section describes the steps for installing the supporting infrastructure around the application including a public `Neutron` network and subnet, a demo tenant and user, a Red Hat Enterprise Linux Server 7.1 `Glance` image, and a key pair for accessing the MSA application servers via **ssh**.

1. Source **overcloudrc**.

```
$ source overcloudrc
$ env | grep OS_
OS_PASSWORD=009fe566ba853020923a06c67c5c6a05fe7f9877
OS_AUTH_URL=http://10.19.137.121:5000/v2.0/
OS_USERNAME=admin
OS_TENANT_NAME=admin
OS_NO_CACHE=True
OS_CLOUDNAME=overcloud
```

2. Create an external network and subnet. This will be the public network for the MSA application.

```
$ neutron net-create ext-net -- --router:external=True --shared=True
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | b1f27b52-6229-41e7-a597-02a070320ab4 |
| mtu                       | 0                                    |
| name                      | ext-net                              |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| router:external           | True                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 346a061a7ef44605bd611efbe5d42b6e     |
+---------------------------+--------------------------------------+
```

3. Export *ext_net* network ID to pass it as a parameter to **heat stack-create**.

```
$ export ext_net_id=$(neutron net-show ext-net | awk ' / id/ { print $4 } ')
$ echo $ext_net_id
b1f27b52-6229-41e7-a597-02a070320ab4
```

4. Create a subnet on *ext-net*.

```
$ neutron subnet-create --name ext-net --allocation-pool=start=10.19.137.137,end=10.19.137.150
--gateway_ip=10.19.143.254 ext-net 10.19.136.0/21

Created a new subnet:
+-------------------+--------------------------------------------------+
| Field             | Value                                            |
+-------------------+--------------------------------------------------+
| allocation_pools  | {"start": "10.19.137.137", "end": "10.19.137.150"} |
| cidr              | 10.19.136.0/21                                   |
| dns_nameservers   |                                                  |
| enable_dhcp       | True                                             |
| gateway_ip        | 10.19.143.254                                    |
| host_routes       |                                                  |
| id                | aebba97c-443d-42da-b8ad-ecb94d3ac607             |
| ip_version        | 4                                                |
| ipv6_address_mode |                                                  |
| ipv6_ra_mode      |                                                  |
| name              | ext-net                                          |
| network_id        | b1f27b52-6229-41e7-a597-02a070320ab4             |
| subnetpool_id     |                                                  |
| tenant_id         | 346a061a7ef44605bd611efbe5d42b6e                 |
+-------------------+--------------------------------------------------+
```

5. Create a demo user.

```
$ openstack user create --password redhat demo
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 3082d95300e546a3aa3525d81d695d72 |
| name     | demo                             |
| username | demo                             |
+----------+----------------------------------+
```

6. Create a demo tenant.

```
$ openstack project create demo-tenant
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | None                             |
| enabled     | True                             |
| id          | a81f507d72c947739d911779f4403ae9 |
| name        | demo-tenant                      |
+-------------+----------------------------------+
```

7. Add the *_member_* role to the *demo-tenant* user.

```
$ openstack role add --user demo --project demo-tenant _member_
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 9fe2ff9ee4384b1894a90878d3e92bab |
| name  | _member_                         |
+-------+----------------------------------+
```

8. Create and source a *keystonerc* file for the demo user.

```
$ cat > ~/demorc << EOF
export OS_USERNAME=demo
export OS_TENANT_NAME=demo-tenant
export OS_PASSWORD=redhat
export OS_CLOUDNAME=overcloud
export OS_AUTH_URL=${OS_AUTH_URL}
export PS1='[\u@\h \W(demo_member)]\$ '
EOF
$ source ~/demorc
```

9. Upload a Red Hat Enterprise Linux Server 7.1 image for the MSA application servers. The image can be obtained by installing the *rhel-guest-image-7* package via `yum` from the **RHEL 7 Common** repo.

```
$ openstack image create --disk-format qcow2  --container-format bare --file /pub/rhel-guest-image-7.1-
20150224.0.x86_64.qcow2 rhel-server7.1
+------------------+--------------------------------------+
| Field            | Value                                |
+------------------+--------------------------------------+
| checksum         | b068d0e9531699516174a436bf2c300c     |
| container_format | bare                                 |
| created_at       | 2015-08-30T03:20:25.000000           |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | c5bfc6bd-2b3e-4a2b-ac29-985ec45c8092 |
| is_public        | False                                |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | rhel-server7.1                       |
| owner            | a81f507d72c947739d911779f4403ae9     |
| properties       | {}                                   |
| protected        | False                                |
| size             | 425956864                            |
| status           | active                               |
| updated_at       | 2015-08-30T03:20:38.000000           |
| virtual_size     | None                                 |
+------------------+--------------------------------------+
```

10. List the image.

```
$ openstack image list
+--------------------------------------+----------------+
| ID                                   | Name           |
+--------------------------------------+----------------+
| c5bfc6bd-2b3e-4a2b-ac29-985ec45c8092 | rhel-server7.1 |
+--------------------------------------+----------------+
```

11. Create a key pair for accessing the MSA application servers via **ssh**.

```
$ openstack keypair create demokp > ~/demokp.pem
$ chmod 600 ~/demokp.pem
$ openstack keypair list
+--------+-------------------------------------------------+
| Name   | Fingerprint                                     |
+--------+-------------------------------------------------+
| demokp | 94:55:b1:fa:cd:79:91:07:ad:b9:18:e4:1c:2b:00:22 |
+--------+-------------------------------------------------+
```

Deploy the MSA Application via Heat

`Heat Orchestration Template (HOT)` is the template format natively supported by `Heat`. These templates define resources. They accept user input to promote re-use.

The MSA application used in this reference architecture is defined as a series of nested templates. Each of the four services and two databases are defined in templates called by the master template. The private `Neutron` network, subnet, and router are also defined in a nested template.

The `Heat` templates used in this example are provided via the associated script archive. They are too lengthy to document.

1. Create the *templates/lib* directory.

```
$ mkdir ~/templates/lib
```

2. Extract the nested example templates to *templates/lib*.

```
$ ls templates/
eapws5_nested.yaml  lib  openstack-tripleo-heat-templates
$ ls templates/lib/
billing-service.yaml       private_network.yaml  product-service.yaml sales-service.yaml
presentation-service.yaml  product-db.yaml       sales-db.yaml
```

3. Deploy the MSA application with `Heat`.

```
$ heat stack-create --template-file templates/eapws5_nested.yaml eap6
--parameters="public_net_id=$ext_net_id"
+--------------------------------------+------------+--------------------+----------------------+
| id                                   | stack_name | stack_status       | creation_time        |
+--------------------------------------+------------+--------------------+----------------------+
| 91c16a0e-62a3-485a-a7ae-d999384fabf9 | eap6       | CREATE_IN_PROGRESS | 2015-08-30T03:59:15Z |
+--------------------------------------+------------+--------------------+----------------------+
```

4. Watch progress with **heat resource-list**.

```
$ heat resource-list eap6
...
| billing-service      | ca3dc55e-42a4-4501-a9ff-848856a4982d | file:///home/stack/templates/lib/billing-
service.yaml      | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| presentation-service | 5262f57a-846e-4ff5-8535-b66a049f0743 |
file:///home/stack/templates/lib/presentation-service.yaml | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| private_network      | aa0fda9e-dc55-4ec2-af62-1d184db5b409 |
file:///home/stack/templates/lib/private_network.yaml      | CREATE_COMPLETE    | 2015-08-30T03:59:16Z
| product-db           | 3b5ca63d-22f4-40cc-a691-79eec5a317b5 | file:///home/stack/templates/lib/product-
db.yaml           | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| product-service      | 605abef9-0001-4649-9a39-bfda3654f7a5 | file:///home/stack/templates/lib/product-
service.yaml      | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| sales-db             | cf49eed9-5e47-47db-95fb-e50baad04954 | file:///home/stack/templates/lib/sales-
db.yaml             | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| sales-service        | 14b0a0e0-8322-4548-9dd7-d52a29e7ebfa | file:///home/stack/templates/lib/sales-
service.yaml        | CREATE_IN_PROGRESS | 2015-08-30T03:59:16Z
| security_group       | f429d022-9608-4fd5-87b5-da7584f5b806 | OS::Neutron::SecurityGroup
| CREATE_COMPLETE    | 2015-08-30T03:59:16Z
```

5. View **nova list** after `Heat` creates the stack successfully.

```
$ nova list
...
| a003370e-f8b2-4d76-bdb6-7b6064e155b1 | billing-service      | ACTIVE | -          | Running     | demo-
net=172.16.5.14, 10.19.137.145 |
| 839347fc-cce9-4025-8c28-8879eddb9bc6 | presentation-service | ACTIVE | -          | Running     | demo-
net=172.16.5.12, 10.19.137.146 |
| e1d5c9a0-634f-4b00-9922-0e3a0bd5ba3e | product-db           | ACTIVE | -          | Running     | demo-
net=172.16.5.11, 10.19.137.142 |
| 190388cc-28fb-4956-bcdf-65d5fb0388b4 | product-service      | ACTIVE | -          | Running     | demo-
net=172.16.5.13, 10.19.137.144 |
| c95c0fbd-2a49-42c1-9346-6a955754f905 | sales-db             | ACTIVE | -          | Running     | demo-
net=172.16.5.10, 10.19.137.143 |
| ab46dd07-3cec-43a8-a2fa-530729541475 | sales-service        | ACTIVE | -          | Running     | demo-
net=172.16.5.15, 10.19.137.141 |
```

6. The *cloud-init* service customizes instances post-boot. The *user-data* section of the nested templates includes the commands performed by *cloud-init* for each microservice instance. **ssh** to *presentation-service* to view */var/log/cloud-init.log* to track progress.

```
$ ssh -l cloud-user -i ~/demokp.pem 10.19.137.146
$ sudo -i
# tail /var/log/cloud-init.log
Sep  2 23:55:21 localhost cloud-init: 03:55:21,588 INFO [org.jboss.as.server] (ServerService Thread Pool
-- 39) JBAS015859:
Deployed "presentation.war" (runtime-name : "presentation.war")
Sep  2 23:55:21 localhost cloud-init: 03:55:21,887 INFO [org.jboss.as] (Controller Boot Thread)
JBAS015961: Http management
interface listening on http://172.16.5.12:9990/management
Sep  2 23:55:21 localhost cloud-init: 03:55:21,931 INFO [org.jboss.as] (Controller Boot Thread)
JBAS015951: Admin console
listening on http://172.16.5.12:9990
Sep  2 23:55:21 localhost cloud-init: 03:55:21,936 INFO [org.jboss.as] (Controller Boot Thread)
JBAS015874: JBoss EAP 6.4.0.GA
(AS 7.5.0.Final-redhat-21) started in 201375ms - Started 207 of 245 services (60 services are lazy,
passive or on-demand)
```

At the conclusion of *cloud-init* the Java application **standalone.sh** should be running. The entire stack creation and post-creation configuration can take up to 30 minutes depending on network conditions.

Test EAP server

This section describes a test procedure for the application.

1. Connect to a server via **ssh** and use **curl** verify the services are running.

```
$ ssh -l cloud-user -i ~/demokp.pem 10.19.137.144
$ sudo -i
# curl http://172.16.5.13:8080/product/products/?featured=true
[{"sku":10001,"name":"ABC HD32CS5002 32-inch LED TV","description":"HD
LED Picture Quality<p/>ConnectShare Movie<p/>Wide Color Enhancement<p/>Clear Motion Rate
60","length":29,"width":3,"height":17,"weight":17,"featured":true,"availability":52,"price":249.99,"image
":"TV"},{"sku":10002,"name":"ABC
HD42CS5002 42-inch LED TV","description":"HD LED Picture Quality<p/>ConnectShare Movie<p/>Wide Color
Enhancement<p/>Clear
Motion Rate
60","length":37,"width":2,"height":22,"weight":20,"featured":true,"availability":64,"price":424.95,"image
":"TV"}
...
```

2. Verify the databases are running and mounted on the persistent storage.

```
$ ssh -l cloud-user -i ~/demokp.pem 10.19.137.142
$ sudo -i
# mysql -e 'SHOW TABLES' product
+--------------------+
| Tables_in_product |
+--------------------+
| Keyword            |
| PRODUCT_KEYWORD    |
| Product            |
+--------------------+
# mount -v | grep mysql
/dev/vdb on /var/lib/mysql type ext4
(rw,relatime,seclabel,data=ordered)
```

3. From a client browser, access *presentation* via the floating IP address to make a test purchase: 'http://10.19.137.142:8080/presentation'

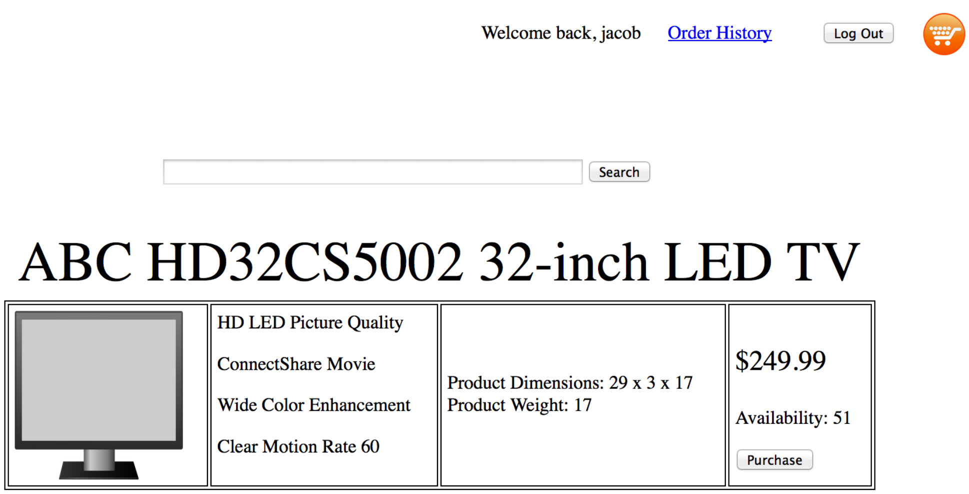

Figure 7.2: EAP6 Web Interface

Complete steps are described in Section 4.6 of this reference architecture: 2015 - Microservice Architecture: Building microservices with JBoss EAP 6

Conclusion

Red Hat Enterprise Linux OpenStack Platform 7 is Red Hat's 5th iteration of RHEL-OSP based on the Kilo community OpenStack release. The reference architecture introduces RHEL-OSP director — Red Hat's integrated management tool set — and describes Red Hat's approach to OpenStack HA. It also describes the steps performed by the Red Hat Systems Engineering team to deploy a highly available RHEL-OSP 7 cluster running a modern Red Hat JBoss Enterprise Application Platform 6 microservices application installed and configured via nested `Heat` templates. Every step in the reference architecture was tested with customer-available code on bare metal servers. It complements existing documentation by providing a comprehensive example of using RHEL-OSP director in a realistic environment with its own hardware and network constraints.

The use case provides:

1. undercloud installation steps
2. overcloud installation steps including post-installation configuration of the `Pacemaker` fence devices
3. A fully worked example of Ceph OSD and journal customization via `Puppet` hiera data accompanied by post-installation configuration steps for increasing placement groups
4. A fully worked network isolation configuration that collapses all networks onto four physical interfaces using both tagged and native VLANs
5. A detailed description of Red Hat's approach to highly available OpenStack including service placement and protection as implemented by RHEL-OSP director
6. A fully worked example of deploying a multi-tiered Red Hat JBoss Enterprise Application Platform 6 application using nested `Heat` templates

EAP microservices applications can be delivered via Red Hat OpenShift Enterprise using containers or another PaaS model. However, the use case is still valid from a DevOps perspective for staging and developing the application, and also for customers who are reluctant to move to PaaS because they have a substantial investment in existing IaaS infrastructure. Also, although it is not covered in this reference architecture, OpenStack can put additional muscle behind an EAP 6 microservices application when it is coupled with `Heat` and `Ceilometer` auto-scaling functionality. Auto-scaling the EAP 6 application behind a `Neutron` LBaaS has potential as an interesting direction for future work.

Appendix A: Contributors

1. Roger Lopez - content review, technical content review (RHEL-OSP)
2. Babak Mozaffari - technical content review (JBOSS EAP)
3. Dan Sneddon - technical content review (Network isolation)
4. Keith Schincke - technical content review (Ceph)
5. Andrew Beekhoff - technical content review (HA)
6. Steven Reichard — content review, technical content review (RHEL-OSP)
7. Scott Lewis — content review, messaging
8. Vinny Valdez — content review, technical content review (RHEL-OSP)

Appendix B: References

Ceph References

1. Ceph Placement Groups
2. Benchmark Ceph Cluster Performance
3. 2015 - RHEL OSP 5: Cinder Volume Performance on Inktank Ceph Enterprise 1.2.2
4. Red Hat Ceph Architecture Guide
5. 2015 - Deploying Highly Available Red Hat Enterprise Linux OpenStack Platform 6 with Ceph Storage

eDeploy and AHC References

1. eDeploy User's Guide
2. Automatic Health Check (AHC) - User Guide

Heat References

1. Heat Orchestration Template (HOT) specification
2. Debugging TripleO Heat templates

JBOSS EAP References

1. How can I execute the JBoss EAP 6 using Systemctl?
2. 2015 - Microservice Architecture: Building microservices with JBoss EAP 6

Red Hat OpenStack Platform References

1. 2015 - Guidelines and Considerations for Performance and Scaling your Red Hat Openstack 6 Cloud
2. Performance Tuning for RabbitMQ in Red Hat Enterprise Linux OpenStack Platform
3. Performance tuning the backend database for Red Hat Enterprise Linux OpenStack Platform
4. Red Hat Enterprise Linux 6 Virtualization Tuning and Optimization Guide
5. OpenStack Performance Optimization
6. Red Hat Enterprise Linux OpenStack Platform 7 Director Installation and Usage
7. Certified Guest Operating Systems in Red Hat Enterprise Linux OpenStack Platform and Red Hat Enterprise Virtualization Deployment Limits for Red Hat OpenStack Platform
8. The Red Hat Satellite 6.0 Provisioning Guide

9. Understanding RHEL OpenStack Platform High Availability (RHEL OSP7 and Director)

Appendix C: Hardware specifications

Table 6. Hardware specifications

Count	Model	Description
8	Dell PowerEdge M520	2x Intel Xeon CPU E5-2450 0 @ 2.10GHz, Broadcom 5720 1Gb Dual Port LOMs, Broadcom 57810S-k Dual Port 10Gb NIC, 6x DDR3 8192 MB @1333 MHZ DIMMs, 2 x 146GB SAS internal disk drives
4	Dell PowerEdge R510	2x Intel® Xeon® CPU X5650 @ 2.67 GHz (6 core), 2 x Broadcom NetXtreme II BCM5709S Gb Ethernet, 2x Emulex Corporation OneConnect 10Gb NIC, 6 x DDR3 8192 MB @1333 MHZ DIMMs, 12x 146GB SAS internal disk drives
1	Dell PowerEdge R720xd	2x Intel® Xeon® CPU X5650 @ 2.67 GHz (6 core), 2 x Broadcom NetXtreme II BCM5709S Gb Ethernet, 2x Emulex Corporation OneConnect 10Gb NIC, 6 x DDR3 8192 MB @1333 MHZ DIMMs, 12x 146GB SAS internal disk drives

Appendix D: Required channels

Red Hat Enterprise Linux OpenStack Platform 7 is available via the Red Hat Content Delivery Network or Red Hat Satellite Server.

Table 7. Required channels

Name	Description
rhel-7-server-extras-rpms	Red Hat Enterprise Linux 7 Server - Extras
rhel-7-server-openstack-7.0-director-rpms	Red Hat OpenStack 7.0 director for RHEL 7 (RPMs)
rhel-7-server-openstack-7.0-rpms	Red Hat OpenStack 7.0 for RHEL 7 (RPMs)
rhel-7-server-optional-rpms	Red Hat Enterprise Linux 7 Server - Optional (RPMs)
rhel-7-server-rpms	Red Hat Enterprise Linux 7 Server (RPMs)

This reference architecture uses a local satellite server for deployments and updates.

Appendix E: Deploying undercloud with SSL

This appendix describes steps for deploying the undercloud with SSL support.

> The product documentation includes additional information for installing the undercloud with SSL support.

1. Generate a private key file.

```
$ openssl genrsa -out privkey.pem 2048
Generating RSA private key, 2048 bit long modulus
......................................................................+++
.........+++
e is 65537 (0x10001)
```

2. Create the distinguished identifier for the certificate.

> Replace this example with appropriate environment-specific answers.

```
$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 365
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
\-----
Country Name (2 letter code) [XX]:*US*
State or Province Name (full name) []:*Texas*
Locality Name (eg, city) [Default City]:*Austin*
Organization Name (eg, company) [Default Company Ltd]:*Red Hat*
Organizational Unit Name (eg, section) []:*Systems Engineering*
Common Name (eg, your name or your server's hostname) []:*192.0.2.2*
Email Address []:*jliberma@redhat.com*
```

3. Write the certificate and key to *undercloud.pem*.

```
$ cat cacert.pem privkey.pem > undercloud.pem
```

4. Copy the combined SSL key to */etc/pki/instal-cert/*.

```
$ sudo mkdir /etc/pki/instack-certs
$ sudo cp ~/undercloud.pem /etc/pki/instack-certs/.
```

5. Set the SELinux context on the key certificate directory and files.

```
$ sudo semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.\*)?"
```

6. Run **restorecon** to enforce the new SELinux contexts.

```
$ sudo restorecon -R /etc/pki/instack-certs
```

7. Modify the undercloud.conf from the previous example to include:
   a. An undercloud public VIP
   b. An undercloud private VIP
   c. The location for the undercloud service certificate.

```
$ head undercloud.conf
[DEFAULT]

image_path = .
local_ip = 192.0.2.1/24
undercloud_public_vip = 192.0.2.2
undercloud_admin_vip = 192.0.2.3
undercloud_service_certificate = /etc/pki/instack-certs/undercloud.pem
local_interface = eno4
masquerade_network = 192.0.2.0/24
dhcp_start = 192.0.2.5
```

8. Install the undercloud with SSL support.

```
$ openstack undercloud install
...
#######
instack-install-undercloud complete.

The file containing this installation's passwords is at
/home/stack/undercloud-passwords.conf.

There is also a stackrc file at /home/stack/stackrc.

These files are needed to interact with the OpenStack services, and
should be
secured.

#######
```

9. Source *stackrc* and verify the OpenStack services have separate internal and public endpoint URLs.

```
$ source ~stackrc
$ openstack endpoint show glance
    +--------------+----------------------------------+
    | Field        | Value                            |
    +--------------+----------------------------------+
    | adminurl     | http://192.0.2.1:9292/           |
    | enabled      | True                             |
    | id           | 6f715600451f433f98e38b72a5b70606 |
    | internalurl  | http://192.0.2.1:9292/           |
    | publicurl    | https://192.0.2.2:13292/         |
    | region       | regionOne                        |
    | service_id   | 8553ca00fa2c4aa98b1d60aa53df3f89 |
    | service_name | glance                           |
    | service_type | image                            |
    +--------------+----------------------------------+
```

Appendix F: Undercloud Service List

```
neutron-dhcp-agent
neutron-openvswitch-agent
neutron-server
openstack-ceilometer-alarm-evaluator
openstack-ceilometer-alarm-notifier
openstack-ceilometer-api
openstack-ceilometer-central
openstack-ceilometer-collector
openstack-ceilometer-notification
openstack-glance-api
openstack-glance-registry
openstack-heat-api-cfn
openstack-heat-api-cloudwatch
openstack-heat-api
openstack-heat-engine
openstack-ironic-api
openstack-ironic-conductor
openstack-ironic-discoverd-dnsmasq
openstack-ironic-discoverd
openstack-keystone
openstack-nova-api
openstack-nova-compute
openstack-nova-conductor
openstack-nova-consoleauth
openstack-nova-scheduler
openstack-swift-account-auditor
openstack-swift-account-reaper
openstack-swift-account-replicator
openstack-swift-account
openstack-swift-container-auditor
openstack-swift-container-replicator
openstack-swift-container-updater
openstack-swift-container
openstack-swift-object-auditor
openstack-swift-object-replicator
openstack-swift-object-updater
openstack-swift-object
openstack-swift-proxy
openstack-tuskar-api
```

Appendix G: Overcloud Service List

```
Cluster name: tripleo_cluster
Last updated: Tue Sep  8 12:41:33 2015
Last change: Tue Sep  8 11:47:03 2015
Stack: corosync
Current DC: overcloud-controller-2 (3) - partition with quorum
Version: 1.1.12-a14efad
3 Nodes configured
112 Resources configured

Online: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]

Full list of resources:

 ip-192.0.2.6   (ocf::heartbeat:IPaddr2):       Started overcloud-controller-0
 Clone Set: haproxy-clone [haproxy]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 ip-172.16.1.11 (ocf::heartbeat:IPaddr2):       Started overcloud-controller-1
 ip-10.19.137.121       (ocf::heartbeat:IPaddr2):       Started overcloud-controller-2
 ip-172.16.2.10 (ocf::heartbeat:IPaddr2):       Started overcloud-controller-0
 ip-172.16.1.10 (ocf::heartbeat:IPaddr2):       Started overcloud-controller-1
 Master/Slave Set: galera-master [galera]
     Masters: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 ip-172.16.3.10 (ocf::heartbeat:IPaddr2):       Started overcloud-controller-2
 Master/Slave Set: redis-master [redis]
     Masters: [ overcloud-controller-2 ]
     Slaves: [ overcloud-controller-0 overcloud-controller-1 ]
 Clone Set: mongod-clone [mongod]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: rabbitmq-clone [rabbitmq]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: memcached-clone [memcached]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-nova-scheduler-clone [openstack-nova-scheduler]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-l3-agent-clone [neutron-l3-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-alarm-notifier-clone [openstack-ceilometer-alarm-notifier]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-heat-engine-clone [openstack-heat-engine]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-api-clone [openstack-ceilometer-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-metadata-agent-clone [neutron-metadata-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-ovs-cleanup-clone [neutron-ovs-cleanup]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-netns-cleanup-clone [neutron-netns-cleanup]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-heat-api-clone [openstack-heat-api]
```

```
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-cinder-scheduler-clone [openstack-cinder-scheduler]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-nova-api-clone [openstack-nova-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-heat-api-cloudwatch-clone [openstack-heat-api-cloudwatch]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-collector-clone [openstack-ceilometer-collector]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-keystone-clone [openstack-keystone]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-nova-consoleauth-clone [openstack-nova-consoleauth]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-glance-registry-clone [openstack-glance-registry]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-notification-clone [openstack-ceilometer-notification]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-cinder-api-clone [openstack-cinder-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-dhcp-agent-clone [neutron-dhcp-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-glance-api-clone [openstack-glance-api]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-openvswitch-agent-clone [neutron-openvswitch-agent]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-nova-novncproxy-clone [openstack-nova-novncproxy]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: delay-clone [delay]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: neutron-server-clone [neutron-server]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: httpd-clone [httpd]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-central-clone [openstack-ceilometer-central]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-ceilometer-alarm-evaluator-clone [openstack-ceilometer-alarm-evaluator]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 Clone Set: openstack-heat-api-cfn-clone [openstack-heat-api-cfn]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
 openstack-cinder-volume        (systemd:openstack-cinder-volume):      Started overcloud-controller-2
 Clone Set: openstack-nova-conductor-clone [openstack-nova-conductor]
     Started: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]

PCSD Status:
  overcloud-controller-0: Online
  overcloud-controller-1: Online
  overcloud-controller-2: Online

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Appendix H: Example fencing Script

This script was used by the Red Hat Systems Engineering team to configure and test `Pacemaker` fencing. The script is not tested nor suitable for production use. It is included as a reference for manually configuring fencing or as an example for scripted configuration.

```
#!/bin/bash

source ~/stackrc
env | grep OS_
SSH_CMD="ssh -l heat-admin"

function usage {
        echo "USAGE: $0 [enable|test]"
        exit 1
}

function enable_stonith {
        # for all controller nodes
        for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d=)
        do
                echo $i
                # create the fence device
                $SSH_CMD $i 'sudo pcs stonith create $(hostname -s)-ipmi fence_ipmilan
pcmk_host_list=$(hostname -s) ipaddr=$(sudo ipmitool lan print 1 | awk " /IP Address  / { print \$4 } ")
login=root passwd=PASSWORD lanplus=1 cipher=1 op monitor interval=60sr'
                # avoid fencing yourself
                $SSH_CMD $i 'sudo pcs constraint location $(hostname -s)-ipmi avoids $(hostname -s)'
        done

        # enable STONITH devices from any controller
        $SSH_CMD $i 'sudo pcs property set stonith-enabled=true'
        $SSH_CMD $i 'sudo pcs property show'
}

function test_fence {

        for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d= | head -n 1)
        do
                # get REDIS_IP
                REDIS_IP=$($SSH_CMD $i 'sudo grep -ri redis_vip /etc/puppet/hieradata/' | awk
'/vip_data.yaml/ { print $2 } ')
        done
        # for all controller nodes
        for i in $(nova list | awk ' /controller/ { print $12 } ' | cut -f2 -d=)
        do
                if $SSH_CMD $i "sudo ip a" | grep -q $REDIS_IP
                then
                        FENCE_DEVICE=$($SSH_CMD $i 'sudo pcs stonith show $(hostname -s)-ipmi' | awk '
/Attributes/ { print $2 } ' | cut -f2 -d=)
                        IUUID=$(nova list | awk " /$i/ { print \$2 } ")
                        UUID=$(ironic node-list | awk " /$IUUID/ { print \$2 } ")
                else
```

```
                        FENCER=$i
                fi
        done 2>/dev/null

        echo "REDIS_IP $REDIS_IP"
        echo "FENCER $FENCER"
        echo "FENCE_DEVICE $FENCE_DEVICE"
        echo "UUID $UUID"
        echo "IUUID $IUUID"

        # stonith REDIS_IP owner
        $SSH_CMD $FENCER sudo pcs stonith fence $FENCE_DEVICE

        sleep 30

        # fence REDIS_IP owner to keep ironic from powering it on
        sudo ironic node-set-power-state $UUID off

        sleep 60

        # check REDIS_IP failover
        $SSH_CMD $FENCER sudo pcs status | grep $REDIS_IP
}

if [ "$1" == "test" ]
then
        test_fence
elif [ "$1" == "enable" ]
then
        enable_stonith
else
        usage
fi
```

Appendix I: NIC Confguration Files

This appendix includes the full text of the network isolation environment files used in this use case.

The *swift-storage.yaml* and *cinder-storage.yaml* are not shown because they were not used.

network-environment.yaml

```
resource_registry:
  OS::TripleO::Compute::Net::SoftwareConfig: /home/stack/nic-configs/compute.yaml
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/nic-configs/controller.yaml
  OS::TripleO::CephStorage::Net::SoftwareConfig: /home/stack/nic-configs/ceph-storage.yaml
  #OS::TripleO::ObjectStorage::Net::SoftwareConfig: /home/stack/nic-configs/swift-storage.yaml
  #OS::TripleO::BlockStorage::Net::SoftwareConfig: /home/stack/nic-configs/cinder-storage.yaml

parameter_defaults:
  ControlPlaneSubnetCidr: "24"
  ControlPlaneDefaultRoute: 192.0.2.1
  EC2MetadataIp: 192.0.2.1
  DnsServers: ['10.19.143.247','10.11.5.19']
  NeutronExternalNetworkBridge: "br-ex"
  InternalApiNetCidr: 172.16.1.0/24
  StorageNetCidr: 172.16.2.0/24
  StorageMgmtNetCidr: 172.16.3.0/24
  TenantNetCidr: 172.16.4.0/24
  ExternalNetCidr: 10.19.136.0/21
  InternalApiAllocationPools: [{'start': '172.16.1.10', 'end': '172.16.1.100'}]
  StorageAllocationPools: [{'start': '172.16.2.10', 'end': '172.16.2.200'}]
  StorageMgmtAllocationPools: [{'start': '172.16.3.10', 'end': '172.16.3.200'}]
  TenantAllocationPools: [{'start': '172.16.4.10', 'end': '172.16.4.200'}]
  ExternalAllocationPools: [{'start': '10.19.137.121', 'end': '10.19.137.151'}]
  InternalApiNetworkVlanID: 4041
  StorageNetworkVlanID: 4042
  StorageMgmtNetworkVlanID: 4043
  TenantNetworkVlanID: 4044
  ExternalNetworkVlanID: 168
  ExternalInterfaceDefaultRoute: 10.19.143.254
  BondInterfaceOvsOptions:
      "bond_mode=balance-tcp lacp=active other-config:lacp-fallback-ab=true"
```

controller.yaml

```
heat_template_version: 2015-04-30
```

```
description: >
  Software Config to drive os-net-config to configure VLANs for the
  controller role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ControlPlaneSubnetCidr:
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  EC2MetadataIp:
    description: The IP address of the EC2 metadata server.
    type:  string
  DnsServers:
    default: []
    description: A list of DNS servers (2 max) to add to resolv.conf.
    type: json
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ExternalNetworkVlanID:
    default: 168
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 4041
    description: Vlan ID for the internal_api network traffic.
```

```
    type: number
  StorageNetworkVlanID:
    default: 4042
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 4043
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 4044
    description: Vlan ID for the tenant network traffic.
    type: number
  ExternalInterfaceDefaultRoute:
    default: '10.19.143.254'
    description: Default route for the external network.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic2
              use_dhcp: false
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
            -
              type: ovs_bridge
              name: br-ex
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
              -
```

```
            ip_netmask: {get_param: ExternalIpSubnet}
          routes:
            -
              ip_netmask: 0.0.0.0/0
              next_hop: {get_param: ExternalInterfaceDefaultRoute}
          members:
            -
              type: interface
              name: nic1
              primary: true
        -
          type: ovs_bridge
          name: br-nic3
          use_dhcp: false
          addresses:
            -
              ip_netmask: {get_param: TenantIpSubnet}
          members:
            -
              type: interface
              name: nic3
              primary: true
            -
              type: vlan
              vlan_id: {get_param: StorageMgmtNetworkVlanID}
              addresses:
              -
                ip_netmask: {get_param: StorageMgmtIpSubnet}
        -
          type: ovs_bridge
          name: br-nic4
          use_dhcp: false
          addresses:
            -
              ip_netmask: {get_param: StorageIpSubnet}
          members:
            -
              type: interface
              name: nic4
              primary: true
            -
              type: vlan
              vlan_id: {get_param: InternalApiNetworkVlanID}
              addresses:
              -
                ip_netmask: {get_param: InternalApiIpSubnet}

outputs:
```

```
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

compute.yaml

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with a VLANs attached for the compute role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ControlPlaneSubnetCidr:
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  EC2MetadataIp:
    description: The IP address of the EC2 metadata server.
    type:  string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The subnet CIDR of the control plane network.
    type: string
  DnsServers:
    default: []
    description: A list of DNS servers (2 max) to add to resolv.conf.
    type: json
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
```

```
      type: string
    TenantIpSubnet:
      default: ''
      description: IP address/subnet on the tenant network
      type: string
    InternalApiNetworkVlanID:
      default: 4041
      description: Vlan ID for the internal_api network traffic.
      type: number
    StorageNetworkVlanID:
      default: 4042
      description: Vlan ID for the storage network traffic.
      type: number
    TenantNetworkVlanID:
      default: 4044
      description: Vlan ID for the tenant network traffic.
      type: number

  resources:
    OsNetConfigImpl:
      type: OS::Heat::StructuredConfig
      properties:
        group: os-apply-config
        config:
          os_net_config:
            network_config:
              -
                type: interface
                name: nic1
                use_dhcp: false
                defroute: false
              -
                type: interface
                name: nic2
                use_dhcp: false
                dns_servers: {get_param: DnsServers}
                addresses:
                  -
                    ip_netmask:
                      list_join:
                        - '/'
                        - - {get_param: ControlPlaneIp}
                          - {get_param: ControlPlaneSubnetCidr}
                routes:
                  -
                    ip_netmask: 169.254.169.254/32
                    next_hop: {get_param: EC2MetadataIp}
                  -
```

```
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
              -
                type: ovs_bridge
                name: br-nic3
                use_dhcp: false
                dns_servers: {get_param: DnsServers}
                addresses:
                  -
                    ip_netmask: {get_param: TenantIpSubnet}
                members:
                  -
                    type: interface
                    name: nic3
                    primary: true
              -
                type: ovs_bridge
                name: br-nic4
                use_dhcp: false
                addresses:
                  -
                    ip_netmask: {get_param: StorageIpSubnet}
                members:
                  -
                    type: interface
                    name: nic4
                    primary: true
                  -
                    type: vlan
                    vlan_id: {get_param: InternalApiNetworkVlanID}
                    addresses:
                    -
                      ip_netmask: {get_param: InternalApiIpSubnet}


outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

ceph-storage.yaml

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config to configure VLANs for the
```

```
  ceph-storage role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ControlPlaneSubnetCidr:
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  EC2MetadataIp:
    description: The IP address of the EC2 metadata server.
    type:  string
  ControlPlaneDefaultRoute:
    description: Default route for the control plane
    type: string
  DnsServers:
    default: []
    description: A list of DNS servers (2 max) to add to resolv.conf.
    type: json
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  StorageNetworkVlanID:
    default: 4042
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 4043
    description: Vlan ID for the storage mgmt network traffic.
    type: number

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
```

```
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              defroute: false
            -
              type: interface
              name: nic2
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: ovs_bridge
              name: br-nic3
              use_dhcp: false
              members:
                -
                  type: interface
                  name: nic3
                -
                  type: vlan
                  vlan_id: {get_param: StorageMgmtNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageMgmtIpSubnet}
            -
              type: interface
              name: nic4
              addresses:
                -
                  ip_netmask: {get_param: StorageIpSubnet}

outputs:
```

```
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```

Appendix J: Revision History

Table 8. Revision History

Revision 2.0	Monday, November 9, 2015	Jacob Liberman
- Updates for Y1 release		
Revision 1.0	Wednesday, September 9, 2015	Jacob Liberman
- Initial publication based on 0-day release and reviews - Ported to asciidoc template		