Canonical OpenStack on Dell EMC DSS 9000 Hardware

Reference Architecture Guide

Dell EMC Converged Platforms and Solutions
September 2017

A Dell EMC Reference Architecture

Revisions

Date	Description
September 2017	Initial release

Table of Contents

Trademarks

Glossary of Terms

Table 1 on page 8 describes terms that are used throughout this document.

Table 1 Terminology

Term	Description
AMT	Active Management Technology
bcache	Allows SSDs to work as cache for one or more slower hard disk drives
BMC	Baseband Management Controller
Bundles	Ready-to-run collections of applications which have been modeled to work together. This can include particular configurations and relations between the software to be deployed.
Charms	Collections of scripts that contain all of the operations necessary to deploy, configure, scale, and maintain cloud applications easily with Juju.
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DSS	Dell Scalable Solutions
HA	High Availability
Hardware Availability Zones	MAAS has a concept zone where server hardware can be placed into different rack, and each rack can be placed in single zone. Or, within same rack, hardware can be divided based upon the power redundancy or the slots within rack. Dell EMC recommends that you place different services in different hardware zone.
HPC	High Performance Computing
Intel RSD	Intel Rack Scale Design
IPAM	IP address management
IPMI	Intelligent Platform Management Interface
Juju	An open source application modeling tool.
Landscape	Landscape is the leading management tool to deploy, monitor and manage Ubuntu servers.
LVM	Logical Volume Manager

Term	Description
MAAS	Metal As A Service is an open source hardware management tool.
MLAG	Multi Chassis Link Aggregation Group
PXE	Preboot Execution Environment
RAID	Redundant Array of Independent Disks
REST API	Representational State Transfer Application Program Interface

Chapter 1 Overview

This document provides a complete Reference Architecture Guide for Canonical OpenStack (Mitaka) on Dell EMC DSS 9000 compute, storage and network hardware. It discusses the Dell EMC DSS 9000 hardware specifications and the tools and services to configure the hardware and software: the foundation cluster and the OpenStack cluster. It also discusses other tools used for the monitoring and management of the cluster in detail, and how all of these components work together in the ecosystem.

This document provides the deployment steps and references to configuration and automation scripts, developed by Dell and Canonical, in the deployment process. Finally, tools for validation of the deployed solution and expected results are provided; including those for OPNFV, targeted for telco customers.

Executive Summary

An OpenStack® based cloud is now a common need for many organizations. Dell EMC and Canonical have worked together to build a jointly-engineered and validated architecture that details software, hardware, and integration points of all solution components. The architecture provides prescriptive guidance and recommendations for:

- Hardware design
  - Infrastructure nodes
  - Compute nodes
  - Controller nodes
  - Storage nodes
- Network design
- Software layout
- System configurations

Dell EMC DSS-9000 Overview

The Dell EMC DSS 9000 is modular rack scale infrastructure that is based on hyperscale principles, meaning that it is:

- Software-defined
- Readily optimized for multiple workloads
- Rooted in open standards
- Specifically designed for easy large scale management.

The Dell EMC DSS 9000 is especially relevant for telco carriers and cloud service providers in a competitive arena being upended by trends like Big Data Analytics, the Internet of Things, and Edge Computing.

DSS 9000 rack level solution is comprised of pools of compute, storage and networking resources which are managed through a single point of rack management. Each DSS 9000 allows combinations of compute and storage sleds, along with shared power, cooling and networking to provide maximum configuration flexibility. This Reference Architecture uses one DSS 9000 half rack. For more information regarding the DSS 9000 Hardware, refer to Chapter 2 Hardware Specifications.

Dell EMC is a founding member of the OpenStack Foundation, and a gold member 2011.

OpenStack Mitaka

This architecture guide is based on the OpenStack Mitaka - the 13th release of the most widely deployed open source software for building clouds. The Mitaka release was designed and built by an international community of 2,336 developers, operators and users from 345 organizations. Please visit https://www.openstack.org/software/mitaka for information regarding OpenStack Mitaka.

Introduction

Dell EMC and Canonical designed this architecture guide to make it easy for Dell EMC and Canonical customers to build their own operational readiness cluster and design their initial offerings using the current releases. Dell EMC and Canonical provide the support and services that the customers need to stand up production-ready OpenStack clusters.

The solution is based on Canonical Mitaka OpenStack Platform that provides Long Term Support (LTS). The code base for Canonical OpenStack Platform is evolving at a very rapid pace. Please see https://www.ubuntu.com/info/release-end-of-life for more information.

Core Components

Table 2 on page 11 displays core solution components.

Table 2 Core Components

Component	Code Name
Block Storage	Cinder with Ceph
Image Service	Glance with Ceph
Compute	Nova with KVM
Identity	Keystone
Networking	Neutron

Optional Components

Optional Components on page 11 displays optional solution components.

Table 3 Optional Components

Component	Code Name
Telemetry	Ceilometer
Orchestration	Heat

Component	Code Name
DNS as a Service	Designate
Validation Testing	Tempest
Dashboard	Horizon

Note: Before using Tempest, review the Tempest documentation at http://docs.openstack.org/developer/tempest/.

The standards-based APIs are the same between all OpenStack deployments, and they enable customers and vendor ecosystems to operate across multiple clouds. The site-specific infrastructure combines open and proprietary software, Dell EMC hardware, and operational processes to deliver cloud resources as a service.

The implementation choices for each cloud infrastructure are highly specific to the requirements of each site. Many of these choices can be standardized and automated using the tools in this Reference Architecture. Conforming to best practices helps reduce operational risk by leveraging the accumulated experience of Dell EMC, Canonical and the broader OpenStack community.

Canonical Metal as a Service (MAAS) is used as a bare metal and VM provisioning tool. The foundation cluster is composed of MAAS and other services running in High Availability (HA) that are used to deploy, manage and update the OpenStack cluster nodes. In the Dell EMC Solution for Canonical OpenStack, the OpenStack Controllers, Computes and Ceph Storage servers comprise the OpenStack cluster.

OpenStack and Canonical

This Reference Architecture is based upon the Canonical Distribution of Ubuntu OpenStack. Canonical, the company behind the Ubuntu, and a Platinum member of the OpenStack Foundation, was the first company to commercially distribute and support OpenStack. Ubuntu is the reference operating system for OpenStack deployments, making it the easiest way to build an OpenStack cloud, and since 2011 the latest version of OpenStack has been included in every Ubuntu release. The release schedules of the two projects are synchronized, ensuring that OpenStack updates and releases are immediately available on widely deployed releases of Ubuntu.

Canonical Foundation Cloud services provide the means to design, deploy, manage, and support customer cloud in POC, development, pre-production, and production environments.

Canonical’s Reference Architecture is delivered on a hyper-converged infrastructure approach, where any of the servers can accommodate more than one specific OpenStack role or service simultaneously. This hyper-converged approach has many benefits, including simplicity of operation and management overhead.

Canonical can also deploy OpenStack in a more traditional manner, grouping server per role:

- Controllers
- Storage
- Computes

MAAS (Metal as a Service) Physical Cloud

MAAS is a complete automation of physical servers for data center operation efficiency on premises, is open source, and supported by Canonical.

MAAS is Metal as Service. It lets you treat physical servers like virtual machines (instances) in the cloud. Rather than having to manage each server individually, MAAS turns your bare metal into an elastic cloud-like resource.

MAAS provides management of a large number of physical machines by creating a single resource pool out of them. Participating machines can then be provisioned automatically and used as normal. When those machines are no longer required they are "released" back into the pool. MAAS integrates all the tools you require in one smooth experience. It includes:

- Web UI (optimized for mobile devices)
- Ubuntu, CentOS, Windows, RHEL and SUSE installation support open source IP address management (IPAM)
- Full API/CLI support
- High availability
- IPv6 support
- Inventory of components
- DHCP and DNS for other devices on the network
- DHCP relay integration
- VLAN and fabric support
- NTP for the entire infrastructure
- Hardware testing
- Composable hardware support

Table 4 on page 13 displays key features of MAAS.

Table 4 Key MAAS Features

Feature	Description
Automation	Automatic discovery and registration of every device on the network. BMC (IPMI, AMT and more) and PXE (IPv4and IPv6) automation.
Fast Deployment	Zero-touch deployment of Ubuntu, CentOS, Windows, RHEL and SUSE. Deploys Linux distributions in less than 5 minutes.
Machine Configuration	Configures the machine’s network interfaces with bridges, VLANs, bonds and more. Creates advanced file system layouts with RAID, bcache, LVM and more.

Feature	Description
DevOps Integration	Integration with DevOps automation tools like conjure-up, Juju, Chef, Puppet, SALT, Ansible and more.
Chassis Management	Full chassis convergence. Dynamic hardware resource management with Intel RSD.
Network Management	Observes and catalogs every IP address on the network (IPAM). Built-in highly available DHCP (active-passive) and DNS (active-active).
Service Tracking	Monitors and tracks critical services to ensure proper operations.
Manage	Comes with a REST API, Web UI and CLI.

Juju Modeling Tool

Juju is an open source application modeling tool that allows you to deploy, configure, scale, and operate cloud infrastructures quickly and efficiently on public clouds such as AWS, GCE, and Azure; along with private clouds such as MAAS, OpenStack, and VSphere.

Its store allows access to a wide range of best practice solutions which you can deploy with a single command. You can use Juju from the command line or through its powerful graphical representation of the model in the GUI.

Why use Juju?

Whether it involves deep learning, container orchestration, real-time big data or stream processing, big software needs operations to be open source and automated.

Juju is the best way to encapsulate all the ops knowledge required to automate the behavior of your application.

Landscape Systems Management Tool

The Landscape systems management tool helps you monitor, manage and update your entire Ubuntu infrastructure from a single interface. Part of Canonical's Ubuntu Advantage support service, Landscape brings you intuitive systems management tools combined with world-class support.

Landscape is the most cost-effective way to support and monitor large and growing networks of desktops, servers and clouds; to reduce your team’s efforts on day-to-day management with Landscape; and to take control of your infrastructure.

The Landscape charm will deploy Landscape Dedicated Server (LDS), and must be connected to other charms to be fully functional. It has a client server model where landscape agents are deployed on the service host, to manage and monitor. See Figure 1 on page 15.

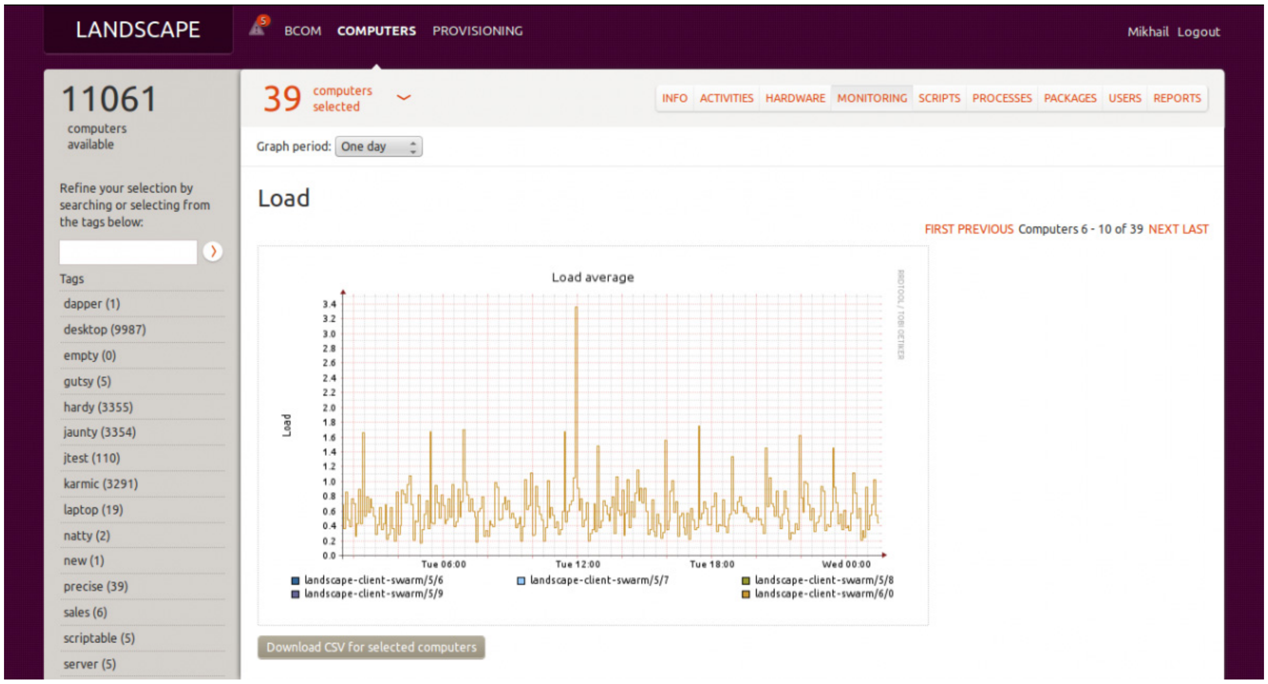


Figure 1 Landscape Dashboard

As part of Canonical's Reference Architecture, this service is deployed by default, where your whole infrastructure will be managed and monitored through Landscape. Table 5 on page 15 displays the features that help Landscape to be part of your Canonical OpenStack infrastructure.

Table 5 Landscape Features

Feature	Description
Systems Management	• Manage desktop, server and cloud deployments • Up to 40,000 machines with a single instance • Create custom profiles for managing different machine classes • Easily install, update, rollback and remove software • Define policies for automated updates and security patches • Configure users and groups
Monitor Your Machines at Scale	• Set alerts for updates on specific machines • Graph trends of temperature, disk, memory usage and system load • List all processes running on a system and remotely kill rogue processes • Build your own graphs with custom metrics
Maintain Security and Compliance	• Patch compliance - keep systems secure and up to date • Role Based Access Control (RBAC)

Feature	Description
	• Automated audit logging and compliance reporting • Regulatory compliance is significantly simplified with custom reporting
Control Your Inventory	• Quickly track full software package information for all registered machines • Gather asset information in real time • Create dynamic search groups to perform operations on categories of machines • Easily access any machine property
Package Repository Management	• Mirror and stage internal or external APT repositories • Upload and manage custom packages

Software Version

This Reference Architecture includes the software versions displayed in Table 6 on page 16.

Table 6 Software Versions

Component	Version
Ubuntu	16.04 LTS (kernel: 4.4)
OpenStack	Mitaka LTS (2016.4)
MAAS	2.2.x
Juju	2.2.x
OpenStack Charms	17.02

Chapter 2 Hardware Specifications

The base validated Reference Architecture solution is based on the Dell EMC DSS 9000. The Reference Architecture uses the following Rack & Server Specifications.

Dell EMC DSS-9000 Rack Specifications

Table 7 DSS-9000 Rack Specifications

Component Type	Component Description	Quantity
Rack	DSS 9000 half rack with 3 blocks of storage nodes	1
Chassis	DSS 9000 Full Width SLED with up to 12x 3.5' HDDs	12
Data Switches	Dell Networking S4048-ON 10G switch	2
iDRAC Switch	Dell Networking PowerConnect 5424	1
Control Cables	1 GbE copper cables with RJ45 connector	12
Data Cables	10 GbE QSPF+ copper cables	48

Dell EMC DSS-9000 Server Specifications

Table 8 DSS-9000 Server Specifications

Component Type	Component Description	Quantity
Processor	Intel® Xeon® E5-2650 v4	2
Memory	256 GB total memory (4x 32 GB or 8 x16 GB RDIMMs)	4
Drive Controller	PERC H730	1
Hard Drive	4 TB 7.2K RPM SATA 6 Gbps 3.5 in. Hot-plug Hard Drive	10
Network Card	Intel Ethernet X520 2 x 10 Gb SFP+ DCS Mezzanine card	1
Network Card	Intel Ethernet X520 2 x 10 Gb SFP+ onboard LOM	1
OOB License	iDRAC8 Express, integrated Dell EMC Remote Access Controller, Express	1
Boot Disks	Seagate 2 x 1 TB 2.5'' disks	2
SSD	Intel SSD 2 x 800 GB (S3510)	2

Rack Layout

The reference deployment of Canonical OpenStack on Dell EMC DSS 9000 servers utilizes three nodes as infrastructure nodes, and nine nodes to deploy OpenStack services in a hyper-converged manner. See Table 9 and Table 10 on page 18.

Table 9 Infrastructure Nodes

Node Name	Node Role
Rack1-Block3-Sled4	Infra # 1 (MAAS, Juju, Landscape, etc.)
Rack1-Block3-Sled3	Infra # 2 (MAAS, Juju, Landscape, etc.)
Rack1-Block3-Sled2	Infra # 3 (MAAS, Juju, Landscape, etc.)

Table 10 OpenStack Nodes

Node Name	Node Role
Rack1-Block3-Sled1	Compute, Storage
Rack1-Block2-Sled4	Compute, Storage
Rack1-Block2-Sled3	Compute, Storage
Rack1-Block2-Sled2	Compute, Storage
Rack1-Block2-Sled1	Compute, Storage
Rack1-Block1-Sled4	Compute, Storage
Rack1-Block1-Sled3	Compute, Storage
Rack1-Block1-Sled2	Neutron/Controller, Storage
Rack1-Block1-Sled1	Neutron/Controller, Storage

Figure 2 on page 19 displays the DSS 9000 rack layout scheme.

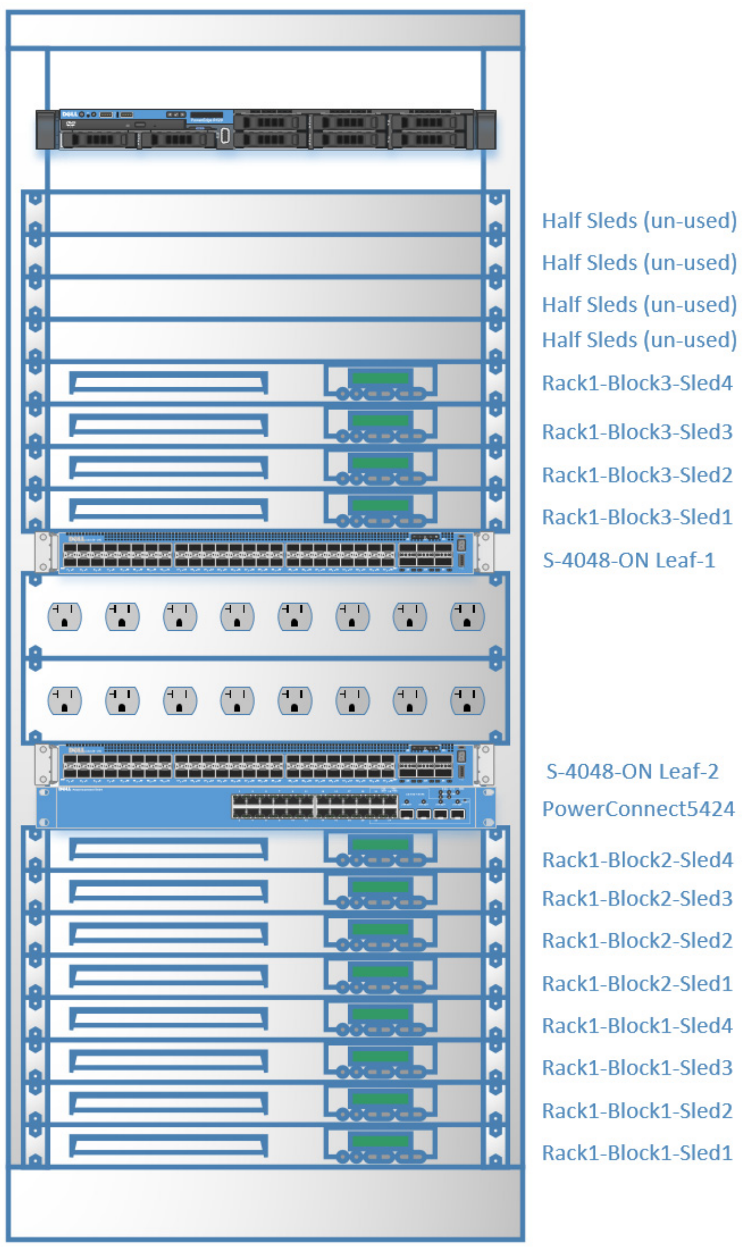


Figure 2 DSS 9000 Rack Layout

Hardware Configuration Notes

The Dell EMC DSS 9000 configurations are used with 10 GbE networking. To ensure that the network is HA ready, two network cards are required for each node. The DSS-9000 servers must be configured, as follows, for the Dell EMC Canonical OpenStack Mitaka Solution:

- BIOS
- iDRAC

- RAID
- Network

Make sure the Physical and Virtual Disks are in Ready State, and the virtual disks are auto-configured to RAID-0. The IPMI over LAN must be enabled in the iDRAC.

For detailed hardware configurations of Dell EMC DSS 9000 Solution for Canonical OpenStack Platform, consult your Dell EMC sales representative and services at openstack@dell.com.

Caution: Please ensure that the firmware on your hardware is up to date.

Chapter 3 Network Architecture

A DSS 9000 rack solution is agnostic to the Top of Rack switch a customer may choose. This reference architecture uses is the Dell EMC S4048-ON. It is an x48 SFP+ port network switch running Cumulus Linux OS 2.5.X. Two of these switches are used to implement high availability on the data network. A Dell EMC PowerConnect 5424 is used for management.

Dell EMC Networking

This topic discusses the following Dell Networking switches:

- S4048-ON 10 GbE Switch on page 21
- PowerConnect 5424 1 GbE Switch on page 21

S4048-ON 10 GbE Switch

The Dell EMC Networking S-Series S4048-ON is an ultra-low-latency 10/40 GbE top-of-rack (ToR) switch built for applications in high performance data center and computing environments. Leveraging a non-blocking switching architecture, the S4048-ON delivers line-rate L2 and L3 forwarding capacity with ultra-low-latency to maximize network performance. Table 11 on page 21 displays the switch specifications.

Table 11 S4048-ON 10 GbE Switch Specifications

Variable	Description
SFP+ Ports	48 x 10 GbE SFP+ Ports
QSFP+ Ports	6 x 40 GbE QSFP+ Ports
RJ45 Ports	1 x Console/Management Port
Operating System	Cumulus Linux OS 2.5.6

Refer to the S4048-ON switch specification sheet for more information.

PowerConnect 5424 1 GbE Switch

The PowerConnect 5424 delivers 24 ports of wire-speed, Gigabit Ethernet. It provides 48 Gbps for switching capacity and 35.6 Mbps forwarding rate. With 24 built-in copper Gigabit Ethernet ports in a 1U form factor, the switches offer flexibility with their four SFP transceiver slots, which can be used in lieu of up to 4 copper ports to support fiber media. Table 12 on page 21 presents the switch specifications.

Table 12 PowerConnect 5424 1 GbE Switch Specifications

Variable	Description
SFP+ Ports	4 x 10 GbE SFP+ Ports
QSFP+ Ports	None

Variable	Description
RJ45 Ports	24 x 1 Gigabit Ethernet Ports
Operating System	Marvell®

Refer to the PowerConnect 5424 switch specification sheet for more information.

Infrastructure Layout

The network consists of the following major network infrastructure layouts:

- **Data Network Infrastructure** - The server NICs and the aggregation switches.
- **Management Network Infrastructure** - The BMC management network, consisting of iDRAC ports and the out-of-band management ports of the switches, aggregated into a 1-rack unit (RU) PowerConnect 5424 switch. This 1-RU switch in turn can connect to one of the aggregation switch to create a separate network with a separate VLAN.

Network Components

The network is comprised of the following component blocks:

- Server Nodes on page 22
- Aggregation Switches on page 23
- VLANs on page 23
- Out of Band Management Network on page 23

Server Nodes

In order to create a highly-available solution, the network must be resilient to loss of a single network switch, network interface card (NIC) or bad cable. To achieve this, the network configuration uses channel bonding across the servers and switches.

All nodes' endpoints are terminated to switch ports that have been configured for LACP bonding mode, across two Dell Networking S4048-ON switches configured with a MLAG across them. For details regarding network configuration on the servers, please contact your Dell EMC services and sales representative.

Table 13 Supported Channel Bonding Modes

Node Type	Channel Bonding Type
Infrastructure Nodes	802.3ad (LACP mode 4)
OpenStack Compute, Storage	802.3ad (LACP mode 4)
OpenStack Controller/Neutron, Storage	802.3ad (LACP mode 4)
Compute/Storage for Scaling	802.3ad (LACP mode 4)

Aggregation Switches

The reference implementation uses two Dell Networking S4048-ON switches. There is a redundant physical 40GbE connection between the two switches. The recommended architecture uses MLAG between the two aggregation switches.

VLANs

The reference implementation uses at a minimum eight (8) separate networks through Layer-2 VLANs. Multiple networks can be combined into single subnet based on end user requirements. See Table 14 on page 23.

Table 14 OpenStack Networks

VLAN	Description
admin	Used for admin-level access to services, including for automating administrative tasks.
internal	Used for internal endpoints and communications between most of the services.
public	Used for public service endpoints, e.g., using the OpenStack CLI to upload images to Glance.
external	Used by neutron to provide outbound access for tenant networks.
data	Used mostly for guest compute traffic between tenants and between tenants and OpenStack services.
storage(data)	Used by clients of the Ceph/Swift storage backend to consume block and object storage contents.
storage(cluster)	Used for replicating persistent storage data between units of Ceph/Swift.
OOB Management	Used for the iDRAC network.

Out of Band Management Network

The Management network of all the servers is aggregated into the PowerConnect 5424 in the reference architecture. One interface on the OOB switch provides an uplink to a router/jump host. The Leaf 2 of the aggregation switch also has one interface connected to the OOB network.

The Out of Band (OOB) Management network is used for several functions:

- The highly available software uses it to reboot and partition servers.
- When an uplink to a router is added and the iDRACs configured to use it as a gateway, there are tools for monitoring the servers and gather metrics on them. Discussion of this topic is beyond the scope of this document. Please contact your Dell EMC sales representative for these tools.

Network Configuration Notes

The reference architecture uses Cumulus Linux 2.5.x as the network operating system on the Dell Networking S4048-ON switches. The following configurations must be performed on these switches:

- Configure two switches in MLAG. Refer to the Cumulus Linux Official documentation to learn more details.
- All the switch ports being used must be added to bonding interface.
- Have the MTU on the bonded interfaces set to 9000 (jumbo frames).
- Configure VLANs to accommodate all the data network traffic on the bonded interfaces.

For detailed network configurations of the Dell EMC DSS 9000 Solution for Canonical OpenStack Platform, consult your Dell EMC sales representative and services at openstack@dell.com.

Caution: Please ensure that the firmware on your network hardware is up to date.

Chapter 4 Foundation Cloud: Infrastructure Components

The foundation cluster or the infrastructure nodes are composed of the following services and tools:

- MAAS
- Juju
- Landscape
- Monitoring
- Log Aggregation

All these services are configured for High Availability.

This chapter provides details about how each of these components work.

How MAAS works

MAAS has a tiered architecture with a central Postgres database backing a region controller (regiond) that deals with operator requests. Distributed Rack Controllers (rackd) provide high-bandwidth services to multiple racks. The controller itself is stateless and horizontally scalable, presenting only a REST API.

Rack Controller (rackd) provides DHCP, IPMI, PXE, TFTP and other local services. They cache large items like operating system install images at the rack level for performance but maintain no exclusive state other than credentials to talk to the controller.

High availability in MAAS

MAAS is mission critical service, providing infrastructure coordination upon which HPC and cloud infrastructures depend. High availability in the region controller is achieved at the database level. The region controller will automatically switch gateways to ensure high availability of services to network segments in the event of a rackd failure.

MAAS can scale from a small set of servers to many racks of hardware in a datacenter. High-bandwidth activities (such as the initial operating system installation) are handled by the distributed gateways enabling massively parallel deployments.

The Node Lifecycle

Each machine (node) managed by MAAS goes through a lifecycle - from *New* to *Enlistment*, further to *Commissioned*, and finally to *Ready State*. There are also special statuses as *Broken* and *Testing*.

New

New machines that PXE-boot on a MAAS network will be enlisted automatically if MAAS can detect their BMC parameters. During the Enlistment phase MAAS will ensure that it can control the power status of the machine through its BMC. Another option is to add machines through the API by supplying BMC credentials.

Commissioning

In the *Commissioning* phase MAAS will collect all data about the machine. This includes detailed hardware inventory like CPU model, memory setup, disks, chipsets, etc., but also information about network connectivity. This information can later be used in deployments. It is in this phase where one can also supply custom commissioning scripts that can update firmware, configure hardware RAID, etc.

Ready

A machine that is successfully commissioned is considered *Ready*. It will have configured BMC credentials (on IPMI based BMCs) for ongoing power control, ensuring that MAAS can start or stop the machine and allocate or (re)deploy it with a fresh operating system.

Allocated

Ready machines can be *Allocated* to users, who can configure network interface bonding and addressing, and disks, such as LVM, RAID, bcache or partitioning.

Deploying

Users then can ask MAAS to turn the machine on and install a complete operating system from scratch without any manual intervention, configuring network interfaces, disk partitions, and more.

Releasing

When a user has finished with the machine they can release it back to the shared pool of capacity. You can ask MAAS to ensure that there is a full disk-wipe of the machine when that happens.

Install MAAS

In this Reference Architecture MAAS is installed in HA fashion using Canonical’s Foundation Cloud process. Only customers who bought Canonical Foundation Cloud SKUs will be able to use this document to install and configure MAAS in HA in an automated fashion.

Configure Your Hardware

You need one small server for MAAS, and at least one server that can be managed with a BMC. Dell EMC recommends that you have the MAAS server provide DHCP and DNS on a network to which the managed machines are connected.

Install Ubuntu Server

Download Ubuntu Server 16.04 LTS, and follow the step-by-step installation instructions on your MAAS server.

MAAS Installation

This section describes the following MAAS installation topics:

- Prerequisites on page 27
- Infrastructure Nodes Requirements on page 27

Prerequisites

Three infrastructure nodes for fully HA, preinstalled with the latest Ubuntu LTS 16.04.1, must be available to host MAAS, the Juju controllers and other runtime and monitoring tools. The nodes must have SSH access to the root user enabled through `authorized_keys`.

Infrastructure Nodes Requirements

Three infrastructure nodes must be already preinstalled, and they host multiple services intended to support building and operating the OpenStack solution, including:

- MAAS and its dependencies, including PostgreSQL
- Juju controllers
- Landscape Dedicated Server
- Monitoring and alerting systems
- Log aggregation and analysis systems
- Capacity planning

Infrastructure nodes host these services either on the bare metal or in KVM virtual machines.

Infrastructure Nodes must have network access to:

- The PXE and BMC networks in order to commission and provision machines
- The various APIs which must be monitored. In order to monitor OpenStack, the nodes must have access to the OpenStack APIs
- Externally, to the Ubuntu archives and other online services, in order to obtain images, packages, and other reference data

To provide HA, Infrastructure Nodes must:

- Be placed in separate Hardware Availability Zones
- Have bonded network interfaces in order to provide resiliency from switch or NIC failures
- Have the MTU on the bonded interfaces set to 9000 (jumbo frames)
- Have a bridge (*br0*) interface active which has the primary bond (typically *bond0*) as its only member.

Note: The bridge inherits the MTU of the underlying device, so there is no need to set its MTU explicitly.

Figure 3 on page 28 displays the logical design of MAAS.

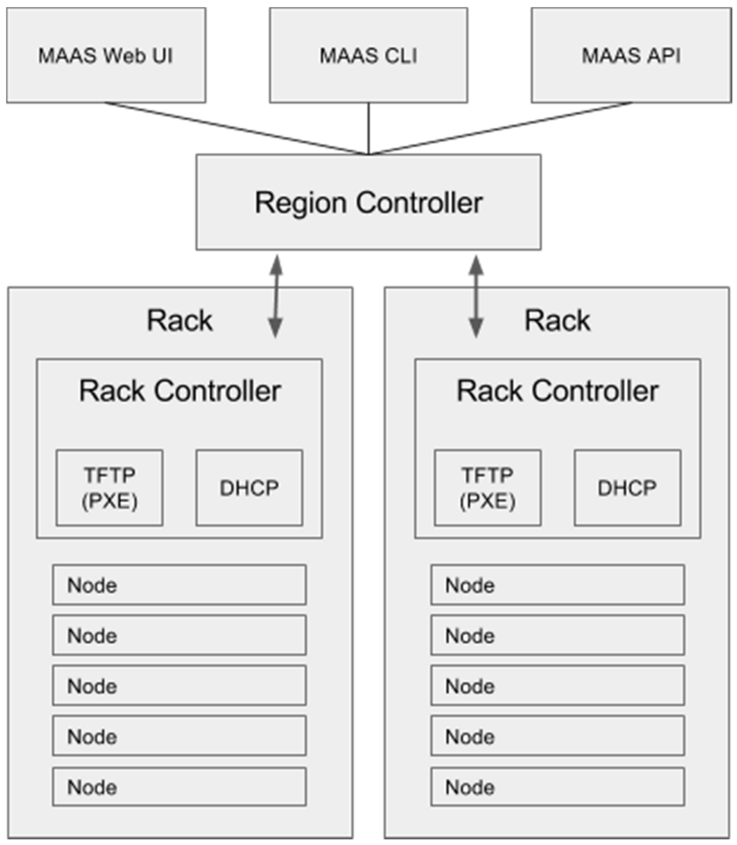


Figure 3 MAAS Logical Design

MAAS Initial Configurations

This section describes the following MAAS initial configurations:

- MAAS Credentials on page 28
- Enlist and Commission Servers on page 28

MAAS Credentials

The MAAS admin user will be created automatically and password will be randomly generated. For your convenience, password is stored in the `maas-pass` file. Username is defined in the `infra.yaml` file.

Enlist and Commission Servers

Now MAAS is ready to enlist and commission machines. To perform that task:

1. Set all the servers to **PXE boot**.
2. Boot each machine once. You should see these machines appear in MAAS.
3. If your machines do not have an IPMI based BMC, edit them and enter their BMC details.
4. Select all the machines and *Commission* them by clicking on the **Take action** button.
5. When machines have a *Ready* status you can start deploying the services.

Juju Components

For an overview of Juju, refer to Juju Modeling Tool on page 14. This section discusses the working of different components of Juju.

Juju Controller - the Heart of Juju

Juju’s controller manages all the machines in your running models, responding to the events that are triggered throughout the system. It also manages scale-out, configuration, and placement of all your models and applications.

Charms

Charms encapsulate a single application, and all the code and know-how it takes to operate the application, such us how to combine and work with other related applications or how to upgrade it.

Charms also allow a hierarchy, with subordinate charms to complement a main service.

Bundles

Bundles may also be optimized for different deployment scenarios of the same software. For example, a scale-out, production-ready version like the Canonical Distribution of Kubernetes, or a development-friendly test version like Kubernetes Core.

Bundles perform the following functions:

- Install
- Configure
- Connect
- Upgrade and update
- Scale-out and scale-back
- Perform health checks
- Undertake operational actions
- Benchmark

Provision

Specify the number of machines you want and how you want them to be deployed, or let Juju do it automatically.

Deploy

Deploy your services, or (re)deploy your entire application infrastructure to another cloud, with a few clicks of your mouse.

Monitor and Manage

The Juju controller manages:

- Multiple models

- All VMs in all your running models
- Scale out, configure and placement
- User accounts and identification
- Sharing and access

Comparing Juju to Any Configuration Management Tool

Juju provides a higher level of abstraction, supplying the tools to manage the full scope of operations beyond deployment and configuration management, regardless of the machine on which it runs.

One of the main advantages of Juju is its dynamic configuration ability, which enables you to:

- Reconfigure services on the fly
- Add, remove, or change relationships between services
- Scale in or out with ease, sharing the operational knowledge and making the most of the wider community.

Figure 4 on page 30 displays a representation of the Juju client with MaaS.

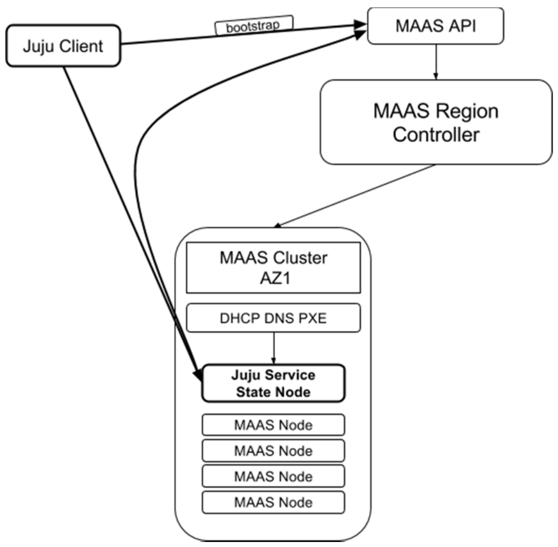


Figure 4 Juju Client with MAAS

Monitoring

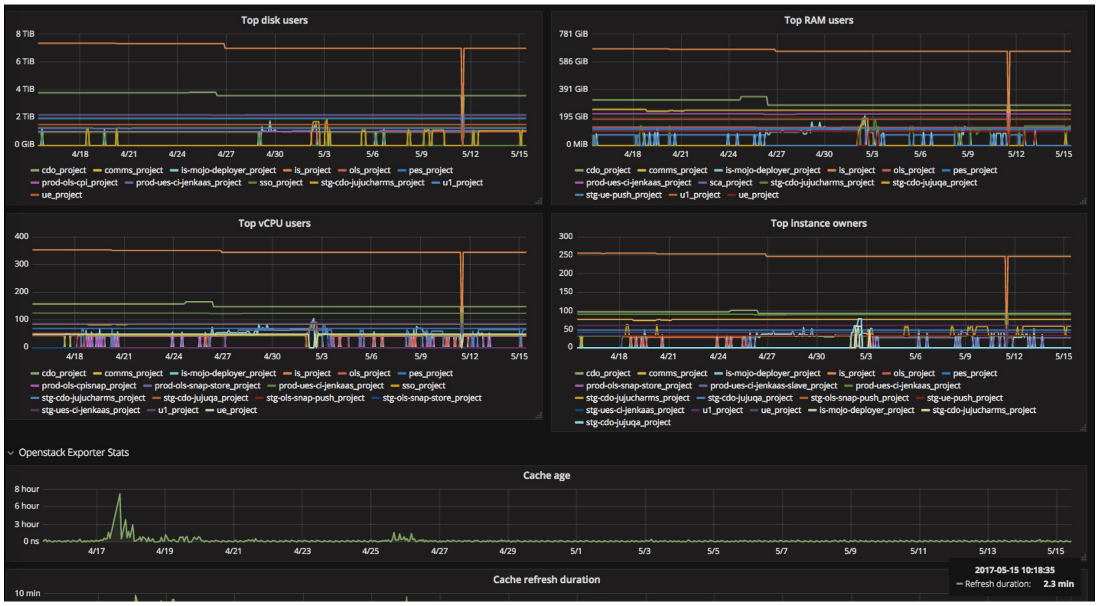
Canonical OpenStack includes a monitoring suite based on the best open source tools available.

Observability Tools

The Canonical OpenStack monitoring suite retrieves information from the different OpenStack components and infrastructure monitors, and combines it in a configurable portal, giving the customer visibility to all the different metrics. The portal aggregates the relevant information from an operational perspective, differentiating the different components (Compute, Network, and Storage).

The Canonical observability tool enables both customers and operators to zoom in on the details of any of the higher-level graphs to obtain further information. The portal includes also an efficient time series database, which enables tracking the evolution of the cloud metrics and health status over time.

Figure 5 on page 31 displays the monitoring tool dashboard.



Top disk users
cdo_project comms_project is-mojo-deployer_project is_project ols_project pes_project prod-ols-cpi_project prod-ues-ci-jenkaas_project sso_project stg-cdo-jujucharms_project u1_project ue_project
Top RAM users
cdo_project comms_project is-mojo-deployer_project is_project ols_project pes_project prod-ues-ci-jenkaas_project sca_project stg-cdo-jujucharms_project stg-cdo-jujuqa_project stg-ue-push_project u1_project ue_project
Top vCPU users
Top instance owners
Openstack Exporter Stats
Cache age
Cache refresh duration
2017-05-15 10:18:35
Refresh duration: 2.3 min

Figure 5 Monitoring Tool Dashboard

Alerting and Continuous Service Checks

Canonical OpenStack encapsulates different operations as actions in the charms. Some of these operations are related to service tests, and produce alerts when the tests are not successful. The number of tests we add to our monitoring system grows over time. Our customers benefit, not only from our internal CI/CD systems, but also from the different checks that we implement in the many different clouds Canonical OpenStack deploys and manages.

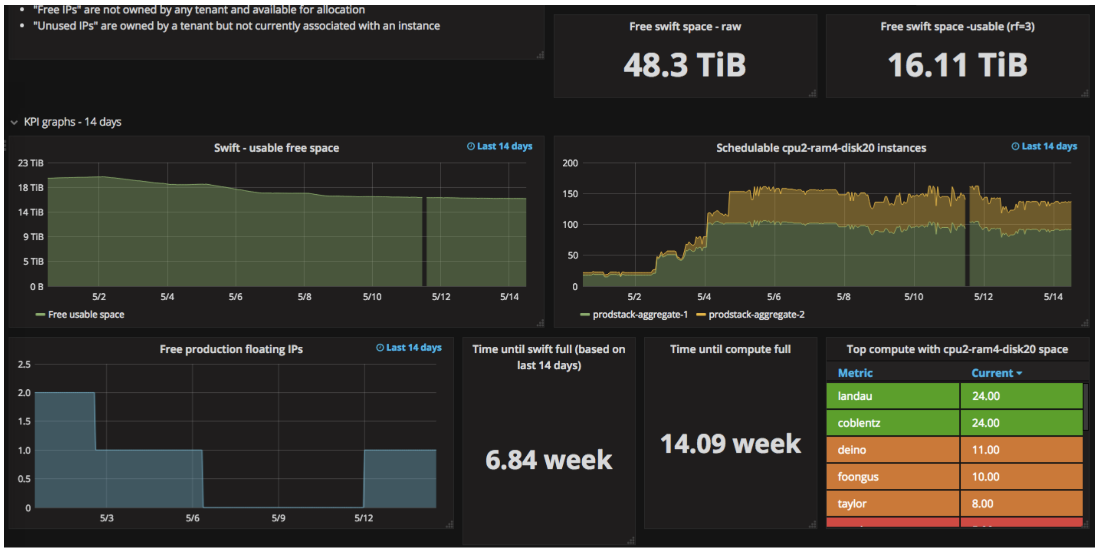
The different tests performed regularly on the customer's clouds are aggregated and accessible to the customer through a Thruk (https://www.thruk.org/) web interface implementation. Canonical can also integrate the underlying Nagios monitoring and alerting system with the existing tools that the customer may be using.

Canonical OpenStack implements Nagios, the Industry Standard in IT Infrastructure Monitoring, which retrieves information of the status of network, servers, storage and services for the deployment. Nagios also implements an alerting system that can be integrated with the operations center, sending any notification in real-time. This alerting enables the customer to have visibility of any issues related to their clouds.

Capacity Planning

The Canonical OpenStack monitoring suite includes both a capacity threshold alert and capacity forecast based on the consumption pattern for the cloud. This will provide our customers with information about when their resources are expected to be exhausted allowing them to plan in advance with their hardware provider.

Figure 6 on page 32 displays capacity planning parameters statistics on the Canonical OpenStack monitoring suite dashboard.



- "Free IPs" are not owned by any tenant and available for allocation
- "Unused IPs" are owned by a tenant but not currently associated with an instance

Free swift space - raw: 48.3 TiB

Free swift space -usable (rf=3): 16.11 TiB

Time until swift full (based on last 14 days): 6.84 week

Time until compute full: 14.09 week

Top compute with cpu2-ram4-disk20 space

Metric	Current
landau	24.00
coblentz	24.00
deino	11.00
foongus	10.00
taylor	8.00

Figure 6 Capacity Planning Parameters Statistics

Log Aggregation

Canonical OpenStack also implements the Elasticsearch suite for log aggregation, giving customers easy visibility to the different logs from their cloud services, without accessing them directly.

These services are integrated with Canonical OpenStack as part of the charms, fulfilling the same requirements around upgradeability and operation. Figure 7 on page 32 displays charms in the Juju dashboard.

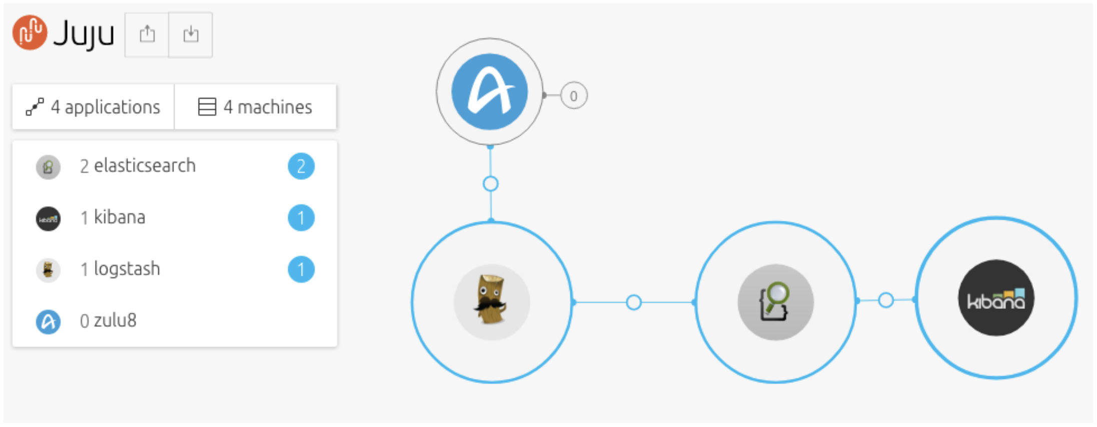


Figure 7 Juju Dashboard

Landscape

The Landscape systems management tool helps you monitor, manage, and update your entire Ubuntu infrastructure from a single interface. Part of Canonical's Ubuntu Advantage support service, Landscape includes intuitive systems management tools combined with world-class support.

This charm deploys Landscape Dedicated Server (LDS), and must be connected to other charms to be fully functional.

For more information about Landscape, go to http://www.ubuntu.com/management.

For an overview and more details about Landscape, refer to Landscape Systems Management Tool on page 14.

Chapter 5 Foundation Cloud: OpenStack Components

This chapter presents detailed information about the OpenStack components included as charms in Foundation Cloud.

Storage Charms

Ceph is a distributed storage and network file system designed to provide excellent performance, reliability, and scalability. Canonical uses Ceph by default for storage, but this can be replaced by, or complemented with, any other storage solution.

ceph-monitor

The Ceph charm has two pieces of mandatory configuration for which no defaults are provided:

- fsid
- monitor-secret

Caution: These two pieces of configuration **must not be changed after bootstrap**. Attempting to do so will cause a reconfiguration error, and new service units will not join the existing Ceph cluster.

At a minimum you must provide a `juju config` file during initial deployment with the fsid and monitor-secret options (the contents of `ceph.yaml` below):

By default the Ceph cluster will not bootstrap until three (3) service units have been deployed and started. This ensures that a quorum is achieved prior to adding storage devices.

This charm uses the new-style Ceph deployment as reverse-engineered from the Chef cookbook at https://github.com/ceph/ceph-cookbooks, although Dell EMC selected a different strategy to form the monitor cluster. Since we do not know the names *or* addresses of the machines in advance, Dell EMC uses the `_relation-joined_hook` to wait for all three nodes to come up, and then write their addresses to `ceph.conf` in the `mon host` parameter. After we initialize the monitor cluster a quorum forms quickly, and OSD bring-up proceeds.

ceph-osd

This charm provides the Ceph OSD personality for expanding storage capacity within a Ceph deployment.

ceph-radosgw

This charm provides the RADOS HTTP gateway supporting S3 and Swift protocols for object storage.

cinder

This charm provides the Cinder volume service for OpenStack. It is intended to be used alongside the other OpenStack components, starting with the Folsom release. Cinder is made up of three (3) separate services:

- An API service
- A scheduler
- A volume service

glance

This charm provides the Glance image service for OpenStack. It is intended to be used alongside the other OpenStack components, starting with the Essex release in Ubuntu 12.04.

Glance may be deployed in a number of ways. This charm focuses on three (3) main configurations. All require the existence of the other core OpenStack services deployed via Juju charms, specifically:

- mysql
- keystone
- nova-cloud-controller

The following assumes these services have already been deployed.

- Local Storage
- Swift backed storage
- Ceph backed storage

In this Reference Architecture we use Ceph backed storage. Figure 8 on page 35 displays Ceph components.

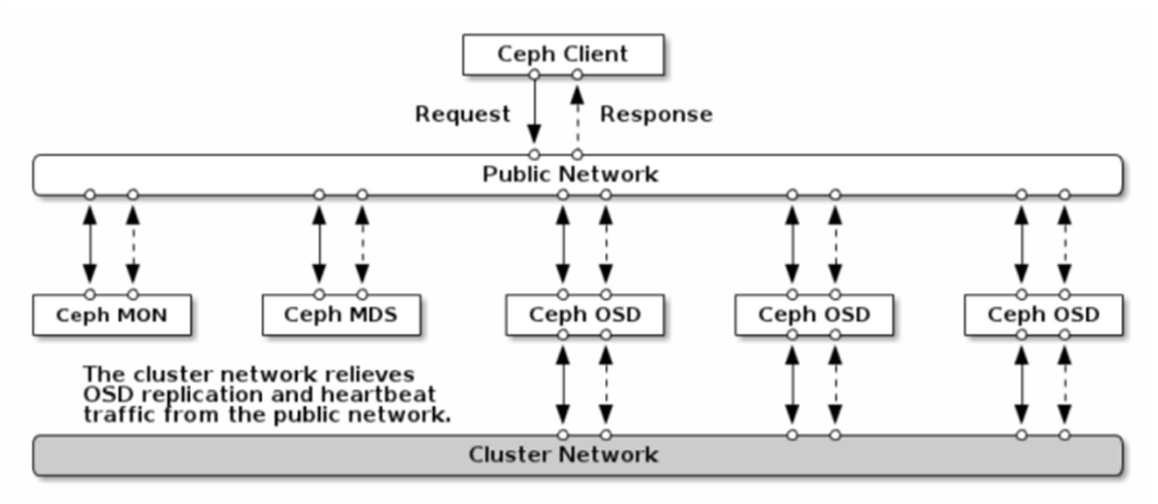


Figure 8 Ceph Components

Compute Charms

Foundation Cloud uses Nova Compute, the OpenStack compute service.

nova-compute

This charm provides Nova Compute. Its target platform is Ubuntu (preferably LTS) + OpenStack.

The following interfaces are provided:

- **cloud-compute** - Used to relate with (at least) one or more of:
  - nova-cloud-controller
  - glance
  - ceph
  - cinder
  - mysql
  - ceilometer-agent
  - rabbitmq-server
  - neutron

In this deployment Canonical uses both KVM and LXD as a hypervisor.

Resources Charms

This topic describes the resource charms used by Foundation Cloud.

HA/Clustering

There are two mutually exclusive HA options available:

- Virtual IP address(es)
- DNS

In this Reference Architecture, deployment and testing the HA option used is the VIP. In both cases a relationship to `hacluster` is required, which provides the Corosync back end HA functionality. To use virtual IP address(es) the clustered nodes must be on the same subnet, such that:

- The VIP is a valid IP address on the subnet for one of the node's interfaces
- Each node has an interface in said subnet

The VIP becomes a highly-available API endpoint.

At a minimum, the configuration option, `vip`, must be set in order to use virtual IP HA. If multiple networks are being used, a VIP should be provided for each network, separated by spaces. Optionally, `vip_iface` or `vip_cidr` may be specified.

To use DNS high availability there are several prerequisites. However, DNS HA does not require the clustered nodes to be on the same subnet.

- Currently the DNS HA feature is only available for MAAS 2.0 or greater environments.

- MAAS 2.0 requires Juju 2.0 or greater.
- The clustered nodes must have static or "reserved" IP addresses registered in MAAS.
- The DNS hostname(s) must be pre-registered in MAAS before use with DNS HA.

At a minimum, the configuration option, `dns-ha`, must be set to *true*, and at least one or more of the following hostnames must be set, in order to use DNS HA:

- os-public-hostname
- os-internal-hostname
- os-internal-hostname

The charm will throw an exception in the following circumstances:

- If neither `vip` nor `dns-ha` is set and the charm is related to `hacluster`
- If both `vip` and `dns-ha` are set, as they are mutually exclusive
- If `dns-ha` is set and none of the `os-{admin,internal,public}`-hostname(s) are set

All the OpenStack services will be deployed in HA, and each service will have three (3) units; each one running on a LXC container in a separate hypervisor. The Canonical OpenStack provides high availability for all OpenStack services as well as highly available Juju. Figure 9 on page 38 displays different types of high availability used in Canonical OpenStack.

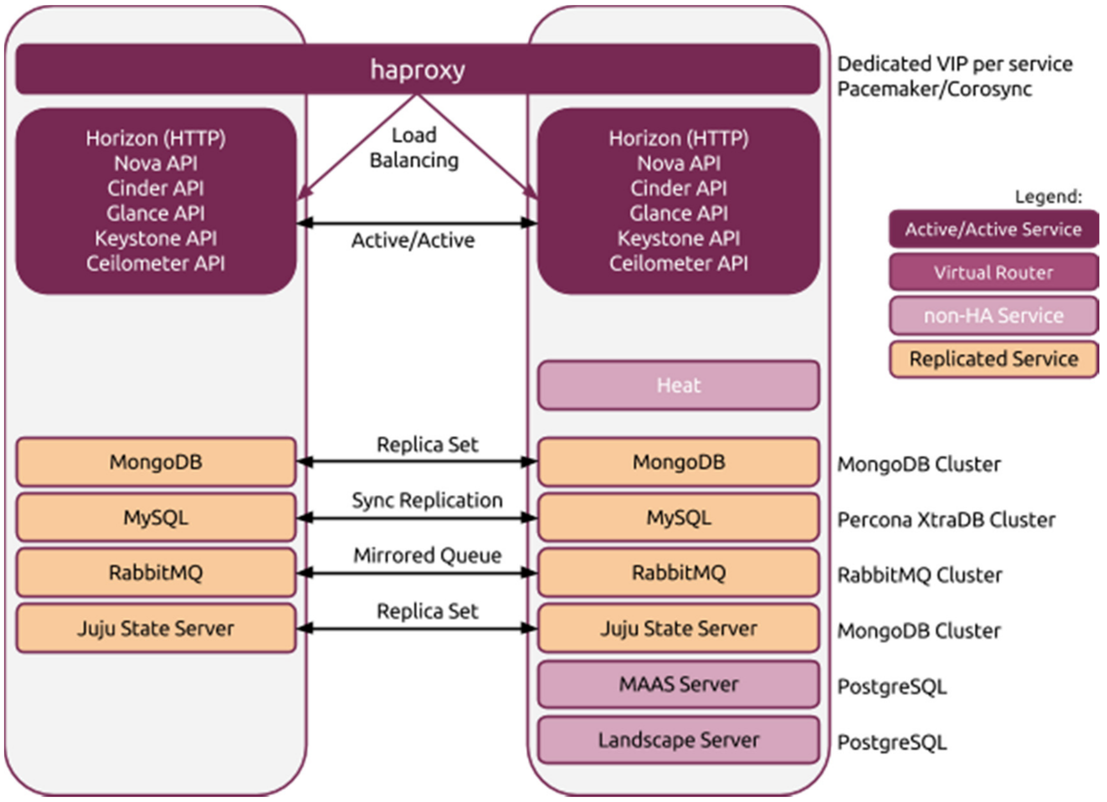


Figure 9 HA Clustering of Foundation Cloud and OpenStack Services

Network Space Support

OpenStack charms support the use of Juju Network Spaces, allowing the charm to be bound to network space configurations managed directly by Juju. API endpoints can be bound to distinct network spaces supporting the network separation of public, internal, and admin endpoints. Access to the underlying MySQL instance can also be bound to a specific space using the `shared-db` relation.

Heat Orchestration

Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates, in the form of text files that can be treated like code. Heat requires the existence of the other core OpenStack services deployed via Juju charms; specifically:

- mysql
- rabbitmq-server
- keystone
- nova-cloud-controller

RabbitMQ-Server on page 39 assumes these services have already been deployed.

RabbitMQ-Server

RabbitMQ is an implementation of AMQP, the emerging standard for high performance enterprise messaging. The RabbitMQ server is a robust and scalable implementation of an AMQP broker. This charm deploys RabbitMQ server and provides AMQP connectivity to clients.

When more than one unit of the charm is deployed the charm will bring up a native RabbitMQ cluster. The process of clustering the units together takes some time.

Note: Due to the nature of asynchronous hook execution, it is possible that client relationship hooks may be executed before the cluster is complete. In some cases, this can lead to client charm errors. Single unit deployments behave as expected.

OpenStack-Dashboard

The OpenStack Dashboard provides a Django-based web interface for use by both administrators and users of an OpenStack Cloud. It allows you to manage Nova, Glance, Cinder, and Neutron resources within the cloud.

nova-cloud-controller

`nova-cloud-controller` is the Cloud controller node for OpenStack Nova. It contains:

- nova-scheduler
- nova-api
- nova-network
- nova-objectstore

If console access is required, then `console-proxy-ip` should be set to a client accessible IP address that resolves to the `nova-cloud-controller`. If running in HA mode, then the public VIP is used if `console-proxy-ip` is set to *local*.

Note: The console access protocol is configured into a guest when it is created; if you change it then console access for existing guests will stop working.

Percona Cluster

Percona XtraDB Cluster is a high availability and high scalability solution for MySQL clustering. Percona XtraDB Cluster integrates Percona Server with the Galera library of MySQL high availability solutions in a single product package, which enables you to create a cost-effective MySQL cluster. This charm deploys Percona XtraDB Cluster onto Ubuntu.

Note: Percona XtraDB Cluster is not a 'scale-out' MySQL solution. Reads and writes are channeled through a single service unit and synchronously replicated to other nodes in the cluster. Reads/writes are as slow as the slowest node you have in your deployment.

Keystone

This charm provides Keystone, the OpenStack identity service. Support for SSL and https endpoint is provided via a set of configuration options on the charm. Two types are supported:

- **use-https** - If enabled, this option tells Keystone to configure the identity endpoint as HTTPS. Under this model the keystone charm will either use the CA as provided by the user, or will generate its own and sync across peers. The certificate will be distributed to all service endpoints, which will be configured to use HTTPS.
- **https-service-endpoints** - If enabled, this option tells Keystone to configure ALL endpoints as HTTPS. Under this model the keystone charm will either use the CA as provided by the user, or will generate its own and synchronize across peers. The certificate will be distributed to all service endpoints, which will be configured to use HTTPS as well as configuring themselves to be used as HTTPS.

aodh Alarming

Ceilometer aims to deliver a single point of contact for billing systems, providing all the counters they need in order to establish customer billing across all current and future OpenStack components:

- The delivery of counters must be traceable and auditable
- The counters must be easily extensible to support new projects
- Agents performing data collections should be independent of the overall system

aodh provides the Alarming service as part of OpenStack telemetry.

Ceilometer

This charm provides the Ceilometer service for OpenStack. It is intended to be used alongside the other OpenStack components, starting with the Folsom release.

Ceilometer is made up of two (2) separate services:

- An API service
- A collector service

This charm enables the services to be deployed in different combination, depending upon user preferences and requirements.

Network Charms

This section describes the various network charms included with Foundation Cloud.

neutron-api

This principle charm provides the OpenStack Neutron API service, which was previously provided by the `nova-cloud-controller` charm. When this charm is related to the `nova-cloud-controller` charm, the `nova-cloud controller` charm will:

1. Shut down its API service
2. Deregister it from keystone
3. Inform the compute nodes of the new neutron URL

Network Space Support

This charm supports the use of Juju Network Spaces, allowing the charm to be bound to network space configurations managed directly by Juju.

Note: This is only supported with Juju 2.0 and above.

API endpoints can be bound to distinct network spaces supporting the network separation of public, internal, and admin endpoints. Access to the underlying MySQL instance can also be bound to a specific space using the `shared-db` relation.

To use this feature, use the `--bind` option when deploying the charm:

```
$ juju deploy neutron-api --bind "public=public-space internal=internal-space
admin=admin-space shared-db=internal-space"
```

Alternatively, these can also be provided as part of a Juju native bundle configuration:

```
neutron-api:
  charm: cs:xenial/neutron-api
  num_units: 1
  bindings:
    public: public-space
    admin: admin-space
    internal: internal-space
    shared-db: internal-space
```

Note: Spaces must be configured in the underlying provider prior to attempting to use them.

Note: Existing deployments using `os-*`-network configuration options will continue to function; these options are preferred over any network space binding provided if set.

neutron-gateway

Neutron provides flexible software defined networking (SDN) for OpenStack.

This charm is designed to be used in conjunction with the rest of the OpenStack related charms in the charm store, to virtualize the network that Nova Compute instances plug into. It is designed as a replacement for nova-network. However, it does not yet support all of the features of `nova-network` (such as multihost), so it may not be suitable for all environments.

Neutron supports a rich plugin/extension framework for proprietary networking solutions, and supports:

- OVS
- (in core) Nicira NVP
- NEC
- Cisco
- Cplane
- Others

By default this architecture uses the **Open vSwitch (OVS)**.

See the upstream Neutron documentation for more details.

The gateway provides two key services:

- L3 network routing
- DHCP services

These are both required in a fully-functional Neutron OpenStack deployment. All internal and external network types are configured with `bridge-mappings` and `data-port`, and the flat-network-providers configuration option of the `neutron-api` charm. Once deployed, you can configure the network specifics using `neutron net-create`.

neutron-openvswitch

This subordinate charm provides the Neutron Open vSwitch configuration for a compute node. Once deployed it takes over the management of the Neutron base and plugin configuration on the compute node. This charm supports DPDK fast packet processing as well.

Chapter 6 Management and Monitoring Tools

This chapter describes the services which have been deployed to manage and monitor your OpenStack cloud. Canonical has a specially designed architecture for their customers to manage and monitor OpenStack clouds. Those services are part of Canonical Foundation Cloud Build (FCB) services. If customers have different requirements as part of FCB, we design the architecture for customers and do the deployments.

Designate

Designate provides DNSaaS services for OpenStack:

- A REST API for domain/record management Multi-tenant
- Integrated with Keystone for authentication
- Framework in place to integrate with Nova and Neutron notifications

MongoDB

MongoDB is a high-performance, open source, schema-free, document-oriented data store that is easy to deploy, manage, and use. MongoDB is network-accessible, written in C++, and offers the following features:

- Collection oriented storage - easy storage of object-style data
- Full index support, including on inner objects
- Query profiling
- Replication and fail-over support
- Efficient storage of binary data including large objects (e.g., videos)
- Auto-sharing for cloud-level scalability (Q209)
- High performance, scalability, and reasonable depth of functionality are the goals for the project

This charm deploys MongoDB in three configurations:

- Single node
- Replica set
- Shared clusters

In this Reference Architecture we deploy as a three node Replica set configuration.

Note: By default, the MongoDB application is installed from the Ubuntu archive, **except for arm64 platforms**. The version of MongoDB in the archive is known to have issues on arm64, so by default this charm will use `ppa:mongodb-arm64/ppa`, which contains backported fixes for this architecture.

PostgreSQL

PostgreSQL is a powerful, open source, object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It is fully ACID-compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL: 2008 data types, including:

- INTEGER
- NUMERIC
- BOOLEAN
- CHAR
- VARCHAR
- DATE
- INTERVAL
- TIMESTAMP

PostgreSQL also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation (http://www.postgresql.org/docs/manuals/).

Kibana

Kibana is a search front end for Logstash. This charm provides Kibana from http://kibana.org/.

Elasticsearch

Elasticsearch is a flexible and powerful open source, distributed, real-time search and analytics engine. Architected from the ground up for use in distributed environments where reliability and scalability are must haves, Elasticsearch gives you the ability to move easily beyond simple full-text search. Through its robust set of APIs and query DSLs, plus clients for the most popular programming languages, Elasticsearch delivers on the near limitless promises of search technology.

HAproxy

HAproxy is a TCP/HTTP reverse proxy that is particularly suited for high availability environments. It features connection persistence through HTTP cookies, load balancing, header addition, modification, and deletion both ways. It has request-blocking capabilities, and provides an interface to display server status.

Prometheus

Prometheus is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

Grafana

Grafana is the leading graph and dashboard builder for visualizing time series metrics.

nrpe Nagios Add-on

Nagios is a host/service/network monitoring and management system. The purpose of this add-on is to allow you to execute Nagios plugins on a remote host in as transparent a manner as possible. This program runs as a background process on the remote host and processes command execution requests from the `check_nrpe` plugin on the Nagios host.

Filebeat

As the next-generation Logstash Forwarder, Filebeat tails logs and quickly sends this information to Logstash for further parsing and enrichment, or to Elasticsearch for centralized storage and analysis.

Logical OpenStack Service Architecture

Figure 10 on page 45 presents how and where services are deployed in the Canonical OpenStack Reference Architecture. Canonical supports multiple custom architectures based on customer requirement of their production OpenStack. Canonical's Foundation cloud services organization helps deploy and manage Cloud for Dell EMC and Canonical customers.

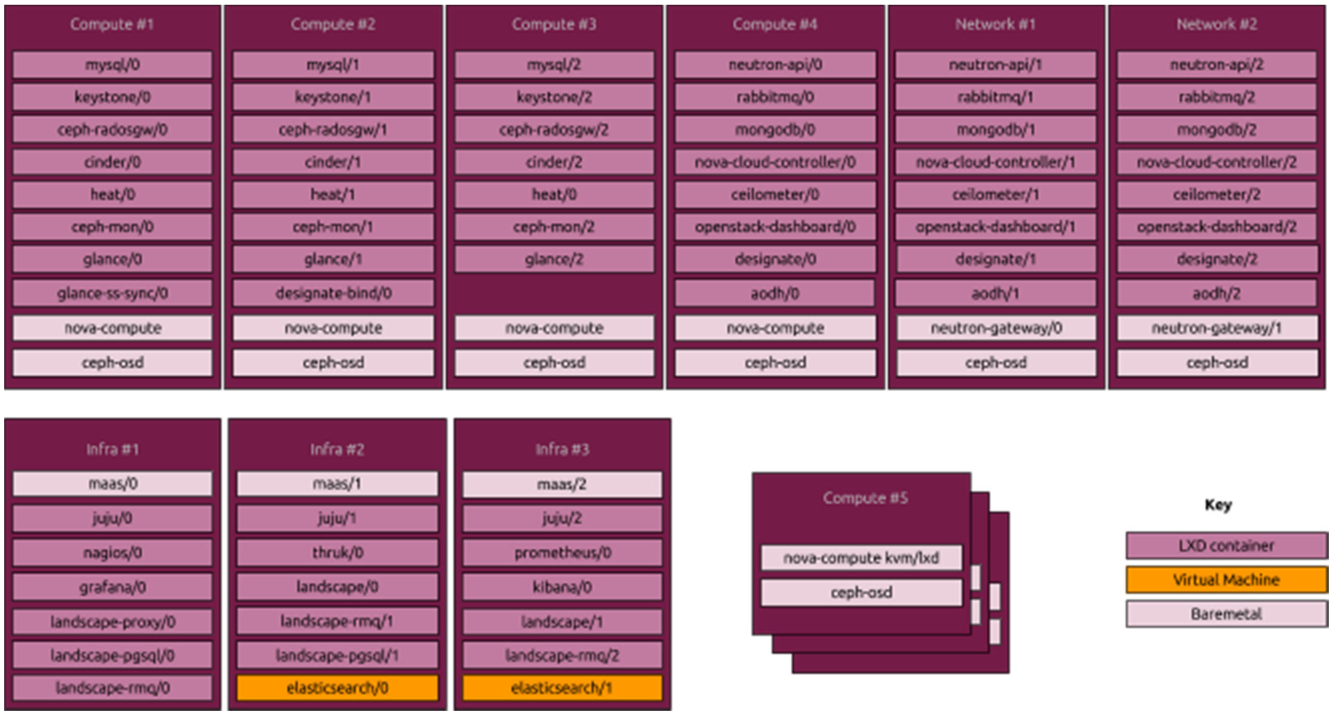


Figure 10 Infrastructure and OpenStack Components Distributed Service Architecture

Chapter 7 OpenStack Validation Testing

This chapter describes validating an OpenStack deployment with:

- Test Cases
- Tempest
- Rally

Test Cases

Canonical used OPNFV `functest` to run tempest and rally test cases on the above deployed environment for test and verification purpose. Canonical used only functional test cases for this environment. Performance test cases are out of scope of this document, but can be delivered as part of a professional services engagement.

Tempest

Tempest is designed to be useful for a large number of different environments. This is useful for:

- Gating commits to OpenStack core projects
- Validating OpenStack cloud implementations for correctness
- Performing as a burn-in tool for OpenStack clouds

As such Tempest tests come in many flavors, each with their own rules and guidelines. The Tempest repository is structured as follows:

- **tempest/** - main directory
- **api/** - API tests
- **scenario/** - complex scenario tests
- **tests/** - unit tests for Tempest internals

Each of these directories contains different types of tests. What belongs in each directory, the rules and examples for good tests, are documented in a *README.rst* file in the directory.

Run the following command to start the `functest` test cases:

```
$ functest run test case tempest-sanity
```

Tempest Test Summary

The Tempest test summary results were:

```
Tempest - INFO -
Tempest - INFO - | Verification
Tempest - INFO - | Status                          | finished
Tempest - INFO - | Duration                | 0:21:30
Tempest - INFO - | Run arguments           | concurrency: 1
Tempest - INFO - | Verifier type           | tempest (namespace: openstack)
Tempest - INFO - | Tests count             | 105
Tempest - INFO - | Tests duration, sec     | 1276.104
Tempest - INFO - | Success                 | 104
Tempest - INFO - | Skipped                 | 1
Tempest - INFO - | Expected failures       | 0
Tempest - INFO - | Failures                | 0
```

Tempest Test Cases

The following test cases were ran on this architecture after deployment:

```
api.compute.flavors.test_flavors.FlavorsV2TestJSON.test_get_flavor
api.compute.flavors.test_flavors.FlavorsV2TestJSON.test_list_flavors
api.compute.security_groups.test_security_group_rules.SecurityGroupRulesTestJSON.test_security_group_rules_create
api.compute.security_groups.test_security_group_rules.SecurityGroupRulesTestJSON.test_security_group_rules_list
api.compute.security_groups.test_security_groups.SecurityGroupsTestJSON.test_security_groups_create_list_delete
api.compute.servers.test_attach_interfaces.AttachInterfacesTestJSON.test_add_remove_fixed_ip
api.compute.servers.test_create_server.ServersTestJSON.test_list_servers
api.compute.servers.test_create_server.ServersTestJSON.test_verify_server_details
api.compute.servers.test_create_server.ServersTestManualDisk.test_list_servers
api.compute.servers.test_create_server.ServersTestManualDisk.test_verify_server_details
api.compute.servers.test_server_actions.ServerActionsTestJSON.test_reboot_server_hard
api.compute.servers.test_server_addresses.ServerAddressesTestJSON.test_list_server_addresses
api.compute.servers.test_server_addresses.ServerAddressesTestJSON.test_list_server_addresses_by_network
api.identity.admin.v2.test_services.ServicesTestJSON.test_list_services
api.identity.admin.v2.test_users.UsersTestJSON.test_create_user
api.identity.admin.v3.test_credentials.CredentialsTestJSON.test_credentials_create_get_update_delete
api.identity.admin.v3.test_domains.DefaultDomainTestJSON.test_default_domain_exists
api.identity.admin.v3.test_domains.DomainsTestJSON.test_create_update_delete_domain
api.identity.admin.v3.test_endpoints.EndPointsTestJSON.test_update_endpoint
api.identity.admin.v3.test_groups.GroupsV3TestJSON.test_group_users_add_list_delete
api.identity.admin.v3.test_policies.PoliciesTestJSON.test_create_update_delete_policy
api.identity.admin.v3.test_regions.RegionsTestJSON.test_create_region_with_specific_id
api.identity.admin.v3.test_roles.RolesV3TestJSON.test_role_create_update_show_list
api.identity.admin.v3.test_services.ServicesTestJSON.test_create_update_get_service
api.identity.admin.v3.test_trusts.TrustsV3TestJSON.test_get_trusts_all
api.identity.v2.test_api_discovery.TestApiDiscovery.test_api_media_types
api.identity.v2.test_api_discovery.TestApiDiscovery.test_api_version_resources
api.identity.v2.test_api_discovery.TestApiDiscovery.test_api_version_statuses
api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_media_types
api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_version_resources
api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_version_statuses
api.image.v2.test_images.BasicOperationsImagesTest.test_delete_image
api.image.v2.test_images.BasicOperationsImagesTest.test_register_upload_get_image_file
api.image.v2.test_images.BasicOperationsImagesTest.test_update_image
api.network.test_extensions.ExtensionsTestJSON.test_list_show_extensions
api.network.test_floating_ips.FloatingIPTestJSON.test_create_floating_ip_specifying_a_fixed_ip_address
api.network.test_floating_ips.FloatingIPTestJSON.test_create_list_show_update_delete_floating_ip
api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_network
api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_port
```

api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_subnet
api.network.test_networks.BulkNetworkOpsTest.test_bulk_create_delete_network
api.network.test_networks.BulkNetworkOpsTest.test_bulk_create_delete_port
api.network.test_networks.BulkNetworkOpsTest.test_bulk_create_delete_subnet
api.network.test_networks.NetworksIpV6Test.test_create_update_delete_network_subnet
api.network.test_networks.NetworksIpV6Test.test_external_network_visibility
api.network.test_networks.NetworksIpV6Test.test_list_networks
api.network.test_networks.NetworksIpV6Test.test_list_subnets
api.network.test_networks.NetworksIpV6Test.test_show_network
api.network.test_networks.NetworksIpV6Test.test_show_subnet
api.network.test_networks.NetworksTest.test_create_update_delete_network_subnet
api.network.test_networks.NetworksTest.test_external_network_visibility
api.network.test_networks.NetworksTest.test_list_networks
api.network.test_networks.NetworksTest.test_list_subnets
api.network.test_networks.NetworksTest.test_show_network
api.network.test_networks.NetworksTest.test_show_subnet
api.network.test_ports.PortsIpV6TestJSON.test_create_port_in_allowed_allocation_pools
api.network.test_ports.PortsIpV6TestJSON.test_create_port_with_no_securitygroups
api.network.test_ports.PortsIpV6TestJSON.test_create_update_delete_port
api.network.test_ports.PortsIpV6TestJSON.test_list_ports
api.network.test_ports.PortsIpV6TestJSON.test_show_port
api.network.test_ports.PortsTestJSON.test_create_port_in_allowed_allocation_pools
api.network.test_ports.PortsTestJSON.test_create_port_with_no_securitygroups
api.network.test_ports.PortsTestJSON.test_create_update_delete_port
api.network.test_ports.PortsTestJSON.test_list_ports
api.network.test_ports.PortsTestJSON.test_show_port
api.network.test_routers.RoutersIpV6Test.test_add_multiple_router_interfaces
api.network.test_routers.RoutersIpV6Test.test_add_remove_router_interface_with_port_id
api.network.test_routers.RoutersIpV6Test.test_add_remove_router_interface_with_subnet_id
api.network.test_routers.RoutersIpV6Test.test_create_show_list_update_delete_router
api.network.test_routers.RoutersTest.test_add_multiple_router_interfaces
api.network.test_routers.RoutersTest.test_add_remove_router_interface_with_port_id
api.network.test_routers.RoutersTest.test_add_remove_router_interface_with_subnet_id
api.network.test_routers.RoutersTest.test_create_show_list_update_delete_router
api.network.test_security_groups.SecGroupIPv6Test.test_create_list_update_show_delete_security_group
api.network.test_security_groups.SecGroupIPv6Test.test_create_show_delete_security_group_rule
api.network.test_security_groups.SecGroupIPv6Test.test_list_security_groups
api.network.test_security_groups.SecGroupTest.test_create_list_update_show_delete_security_group
api.network.test_security_groups.SecGroupTest.test_create_show_delete_security_group_rule
api.network.test_security_groups.SecGroupTest.test_list_security_groups
api.network.test_subnetpools_extensions.SubnetPoolsTestJSON.test_create_list_show_update_delete_subnetpools
api.network.test_versions.NetworksApiDiscovery.test_api_version_resources
api.orchestration.stacks.test_resource_types.ResourceTypesTest.test_resource_type_list
api.orchestration.stacks.test_resource_types.ResourceTypesTest.test_resource_type_show
api.orchestration.stacks.test_resource_types.ResourceTypesTest.test_resource_type_template
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_get_deployment_list
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_get_deployment_metadata
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_get_software_config
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_software_deployment_create_validate
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_software_deployment_update_no_metadata_change
api.orchestration.stacks.test_soft_conf.TestSoftwareConfig.test_software_deployment_update_with_metadata_change
api.orchestration.stacks.test_stacks.StacksTestJSON.test_stack_crud_no_resources
api.orchestration.stacks.test_stacks.StacksTestJSON.test_stack_list_responds

api.volume.test_volumes_actions.VolumesV1ActionsTest.test_attach_detach_volume_to_instance
api.volume.test_volumes_actions.VolumesV2ActionsTest.test_attach_detach_volume_to_instance
api.volume.test_volumes_get.VolumesV1GetTest.test_volume_create_get_update_delete
api.volume.test_volumes_get.VolumesV1GetTest.test_volume_create_get_update_delete_from_image
api.volume.test_volumes_get.VolumesV2GetTest.test_volume_create_get_update_delete
api.volume.test_volumes_get.VolumesV2GetTest.test_volume_create_get_update_delete_from_image
api.volume.test_volumes_list.VolumesV1ListTestJSON.test_volume_list
api.volume.test_volumes_list.VolumesV2ListTestJSON.test_volume_list
tempest.scenario.test_network_basic_ops.TestNetworkBasicOps.test_network_basic_ops
tempest.scenario.test_server_basic_ops.TestServerBasicOps.test_server_basic_ops
tempest.scenario.test_server_multinode.TestServerMultinode.test_schedule_to_all_nodes
tempest.scenario.test_volume_boot_pattern.TestVolumeBootPattern.test_volume_boot_pattern
tempest.scenario.test_volume_boot_pattern.TestVolumeBootPatternV2.test_volume_boot_pattern

Rally

OpenStack is, undoubtedly, a really huge ecosystem of cooperative services. Rally is a benchmarking tool that answers the question, "How does OpenStack work at scale?" To make this possible, Rally automates and unifies multi-node OpenStack deployment, cloud verification, benchmarking, and profiling. Rally does this in a generic way, making it possible to check whether OpenStack is going to work well on, for example, a 1k-servers installation under high load. Thus it can be used as a basic tool for an OpenStack CI/CD system that will continuously improve its SLA, performance and stability.

Rally Tests Summary

Table 15 on page 49 displays the Rally test results run on this architecture.

Table 15 Rally Test Results

Module	Test Cases	Results
authenticate	12	Pass (100%)
cinder	11	Pass (100%)
glance	19	Pass (100%)
heat	16	Pass (100%)
keystone	50	Pass (100%)
neutron	30	Pass (100%)
nova	26	Pass (100%)
quotas	12	Pass (100%)
requests	2	Pass (100%)

Rally Test Cases:

Rally has multiple test scenarios against each service. Below are the test scenarios listed which has been tested as part of Rally tests. Each scenario will have multiple repeated tests related to service tested.

test scenario Authenticate.validate_glance
test scenario Authenticate.keystone
test scenario Authenticate.validate_heat
test scenario Authenticate.validate_nova
test scenario Authenticate.validate_cinder
test scenario Authenticate.validate_neutron
test scenario GlanceImages.list_images
test scenario GlanceImages.create_image_and_boot_instances
test scenario CinderVolumes.create_and_extend_volume
test scenario CinderVolumes.create_and_delete_volume
test scenario CinderVolumes.create_from_volume_and_delete_volume
test scenario CinderVolumes.create_and_delete_snapshot
test scenario HeatStacks.create_suspend_resume_delete_stack
test scenario KeystoneBasic.create_tenant_with_users
test scenario KeystoneBasic.create_add_and_list_user_roles
test scenario KeystoneBasic.add_and_remove_user_role
test scenario KeystoneBasic.create_update_and_delete_tenant
test scenario KeystoneBasic.create_and_delete_service
test scenario KeystoneBasic.create_tenant
test scenario KeystoneBasic.create_user
test scenario KeystoneBasic.create_and_list_tenants
test scenario KeystoneBasic.create_and_delete_role
test scenario KeystoneBasic.get_entities
test scenario KeystoneBasic.create_and_list_users
test scenario NeutronNetworks.create_and_delete_ports
test scenario NeutronNetworks.create_and_list_routers
test scenario NeutronNetworks.create_and_delete_routers
test scenario NeutronNetworks.create_and_list_ports
test scenario NeutronNetworks.create_and_delete_subnets
test scenario NeutronNetworks.create_and_delete_networks
test scenario NeutronNetworks.create_and_list_networks
test scenario NeutronNetworks.create_and_list_subnets
test scenario NovaKeypair.boot_and_delete_server_with_keypair
test scenario Quotas.cinder_update
test scenario Quotas.neutron_update
test scenario Quotas.cinder_update_and_delete
test scenario Quotas.nova_update_and_delete
test scenario Quotas.nova_update
test scenario HttpRequests.check_request

Appendix A Update History

This appendix lists changes to this document, per major release.

Initial Release

First Architecture Guide for the Dell EMC Solution on Canonical OpenStack Platform.

Appendix B References

Please see the following resources for more information.

Dell EMC Documentation

- http://www.dell.com/en-us/work/learn/rack-scale-infrastructure
- http://en.community.dell.com/techcenter/cloud/w/wiki/12401.dell-emc-canonical-openstack-cloud-solutions

Canonical Documentation

- https://jujucharms.com/
- https://maas.io/
- https://wiki.ubuntu.com/
- https://www.canonical.com/
- https://www.ubuntu.com/
- BootStack: Fully managed operations OpenStack and Kubernetes

Canonical Services and Support

- Canonical Foundation Cloud Services
- OpenStack UA Support

OpenStack Documentation

- https://wiki.openstack.org/wiki/Main_Page
- https://wiki.opnfv.org/display/functest/Functest+1.+Getting+Started
- https://docs.openstack.org/charm-guide/latest/

To Learn More

If you need additional services or implementation help, please contact your Dell EMC sales representative, or email openstack@dell.com.