Cork Institute of Technology

Department of Computing

Building a Cloud-Based Network Management System in VMware and OpenStack

Thesis Report

May-2017

By

Sohail Akhter

Student ID: R00132956

Masters in Cloud Computing

Cork Institute of Technology

Supervisor: David Murphy

Declaration

This report is submitted in partial fulfilment of the requirements for the Degree of Master of Cloud Computing in the Cork Institute of Technology. It represents substantially the result of my own work except where explicitly indicated in the text.

Abstract

This thesis explores the challenges of building cloud based Network Management System (NMS) using VMware based and Redhat OpenStack private cloud platforms. The aim of this thesis is to identify the issues that will arise during cloud infrastructure deployment using both technologies and when deploying NMS in both cloud solutions. Both cloud solutions are discussed in terms of challenges that are faced during deployment, scalability, manageability, and recovery in the case of failure. As companies are moving towards cloud based infrastructure, they demand from product developers to build products that can run in their cloud infrastructure. From product development company's perspective, there is huge pressure on them to build products with speed, elasticity, improved quality, features rich and flexible to run in many environments. To achieve this development teams are moving towards Continuous Integration (CI) where they are continuously building and testing software using CI tools such as Jenkins. VMware and OpenStack are two of the leading cloud platform providers, each allowing efficient utilization of physical infrastructure in cloud environment. These cloud environments will be used by more than 2000 developers in product development units. It will help to reduce pressure on developers by providing a flexible and elastic testing environment that allows them to fail and try again and build high quality, feature-rich products. This serves as a guide on how to design and build OpenStack and VMware based clouds by highlighting issues faced during deployment, expansion and operations of these cloud environments.

Acknowledgements

I would like to thank David Murphy at Cork Institute of Technology for his time, guidance and valuable feedback over the course of this research project. I would also like to thank my colleagues in Ericsson for their assistance over the past two years. Finally, a sincere thank you to my wife Nageena and son Mohid for their patience and generosity in providing me with time to conduct and write this thesis report.

Table of Contents

Tables

Figures

1. Introduction

1.1 Research Problem, Scope, Objective

Today's businesses are operating in a business climate of breakneck speed and continuous change. Most companies now using cloud services as part of their IT strategy. Cloud services are enabling businesses to transform their IT operations so that they can deliver customer value more effectively and respond to these changing market forces faster.

Ericsson is world leader in telecommunication world providing equipment, software, and services to enable transformation through mobility (Ericsson, 2017). Ericsson has product development unit is situated in Athlone. Athlone site is R&D center for Ericsson's network management products. Currently all the application development and testing is performed on physical hardware. Physical hardware deployment takes weeks to make it ready for developers as application runs on pre-defined set of hardware and tested firmware for compute and storage systems. Same hardware is deployed at customer premises to run the application.

Customers are now demanding application that can run on their cloud infrastructure and they don't want application tightly coupled with hardware. There are huge demands on product development units to build software that can meet the demands of rapidly changing market. Teams are transformed from traditional waterfall development approach to Agile ways of working where they have to deliver new or enhanced product features in every sprint. Developers need an environment that is promptly available and they could deploy/destroy it as many time without any necessary infrastructure teams support. Cloud environment will help developers to innovate and experiment as they have new instances available for experimentation purposes. There is push from management for efficient utilization of physical infrastructure.

The thesis will discuss the challenges that are faced during the deployment of VMware based and Red OpenStack cloud that will be used for deployment of Network management system. Issues will be discussed that are faced during deployment, infrastructure expansion and operational activities.

Developers will use this environment to develop the application and test how it will work at customer cloud infrastructure as well as functional testing. Actual product design and strategy to move to cloud is not scope of this project.

1.2 Motivation

All the work conducted for this thesis will carried out at Ericsson Athlone' s test environment that provides infrastructure services to Product development units. I am part of cloud infrastructure team that has been given the task to build the cloud environment using VMware based and Openstack cloud. So all the challenges that will be discussed based on actual deployments.

1.3 Research Questions

Currently my research questions include but are not limited to –

RQ1. Deployment Challenge(s)

How VMware based and OpenStack private cloud environments are deployed and how easy is deployment and what are challenges faced?

RQ2. Expansion Challenge(s)

How to expand the environment with physical hardware and what are challenges faced?

RQ3. Operational Challenge(s)

What are the issues faced in day to day operations of both environments?

RQ4. Failure Recovery Challenge(s)

How both environments behaved in case of hardware failure?

RQ5. Application Migration Challenge(s)

How the application is moved/migrated to cloud environments?

1.4 Background Research

For the purposes of this research we will deploy physical Redhat Openstack and VMware vCloud director based cloud environments as well as an EMC VNX 5400 Storage Area Network (SAN) and network based on Extreme switches at core, access, and distribution layers. After successful implementation same environments will be provided to developers in product development to deploy and test the Network Management System.

For my research project I have chosen The NOVA University research methodology. This methodology was chosen as I felt it best suits the comparison of characteristics within cloud computing implementations using VMware and OpenStack. Case study based research method will be used and it will be exploratory case study to find how NMS application behaves in different cloud environments. The research aim is deductive. It will help us to decide and take key actions before deploying NMS in customer's different cloud environments.

This thesis will require an examination of recent relevant studies in VMware based and Openstack cloud. In performing background research CIT library will be the primary source of relevant material and information. IEEE Xplore and ACM electronic library will be used to gain access to journal and magazines on the subject matter in question.

Network management System's product development unit is key stakeholder in this implementation. As the NMS application is deployed in both cloud environment, they will confirm that it is behaving same as bare metal installation.

2. Literature Review

2.1 Introduction to Cloud Computing

NIST (National Institute of Standards and Technology) is renowned institution all over the world for its work in the field of Information Technology. According to NIST (Mell & Grance, 2011), cloud computing is defined to consist of

Three service models

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Four deployment models

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

Five essential characteristics

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Cloud has its foundations in virtualization. Humble, et al., (2016) explain that virtualization in computer science is something that is not real. Virtualization software is called hypervisor or virtual machine monitor. Physical system that runs virtualization is called host and virtual machines that are installed on that hypervisor are called guest. Hypervisor provide virtualization management tasks, such as providing virtual hardware, allocating resources in real time, VM life cycle management, migrating of VMs, defining policies for virtual machine management, and so on. Hypervisors are categorized as either Type 1 or Type 2 based on where they reside in hardware. If hypervisor is running on top of hardware it is Type 1 hypervisor. Type 1 hypervisors are also called Bare Metal, Embedded, or Native Hypervisors. VMware ESXi, Citrix Xen Serve and Microsoft Hyper-V are good examples of type 1 hypervisor. Following figure provides an illustration of Type 1 hypervisor.

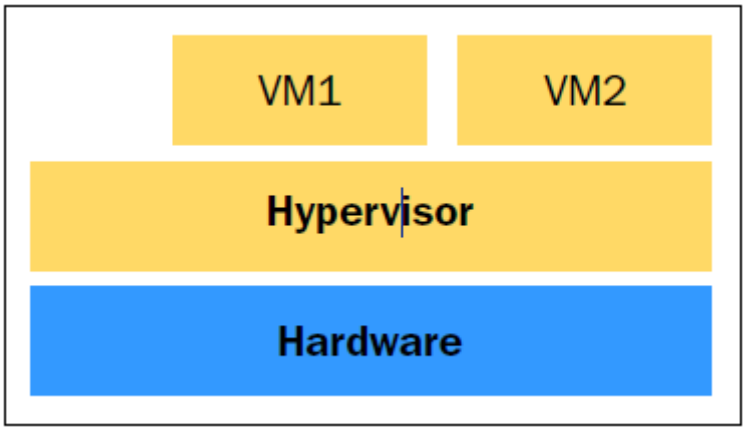

Figure 1 - Type 1 Hypervisor (Source: Humble, et al., 2016)

Type 2 hypervisor reside on top of operating system. They are also called hosted hypervisors. Type 2 hypervisors are dependent on host operating system for their operations. VMware workstation, VMware Player, VMware Fusion, Oracle virtual box are examples of type 2 hypervisors. Following figure provides an illustration of Type 2 hypervisor.

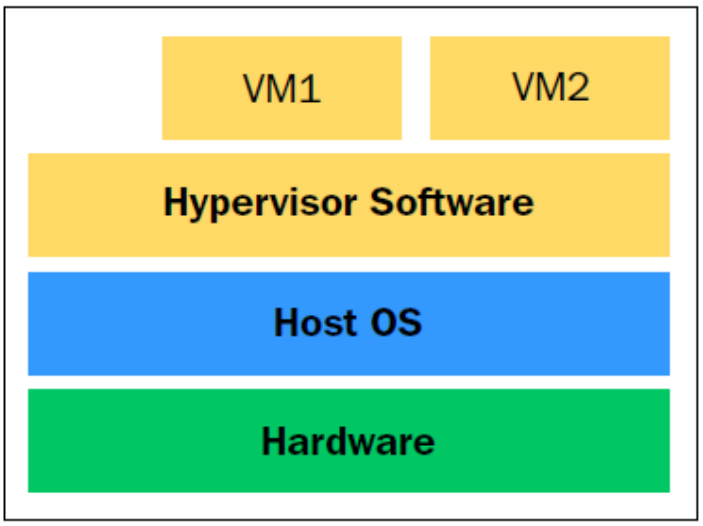

Figure 2 - Type 2 Hypervisor (Source: Humble, et al., 2016)

To build end to end cloud computing platform is a challenging job for most of the IT companies. They must have fulfilled all complicated cloud features while implementation (Grossman & Robert , 2009). The primary function in cloud based companies, is to utilize huge amounts of compute services and manage storage and network. Virtualization technology helps to allow efficient hardware utilization through creating new virtual machine image. Cloud monitoring tools are used to manage cloud based applications. These monitoring tools provide safe cloud environment by identifying and resolving critical issues. According to Service Level Agreement (SLA) cloud service provider should provide highly available and scalable service to customers. Automation in business processes is a new trend in cloud industry to automate process and task. Task automation helps to reduce production cost when large number of users send request for cloud services. The key function in this process is to provision new virtual machine and assign resources for new user. To build a user friendly environment is main the main challenge and its possible only through web based portals. Web portals are needed to count usage of cloud resources to calculate cost (Behrendt, et al., 2011).

2.2 VMware vCloud suite

The VMware vSphere product suite is collection of products and features that provides virtualization functionality to build VMware based cloud on top of that. ESXi host, vCenter Server, Update Manager, Distributed Resource Scheduling, High availability are some of the products and features of vSphere suite. VMware vCloud suite provides a bundle solution to build a cloud. It consists of vSphere suite and other additional products like vCloud director (Lowe & Marshall, 2014). Following figure shows high level architecture and different components that are used in VMware vCloud solution.

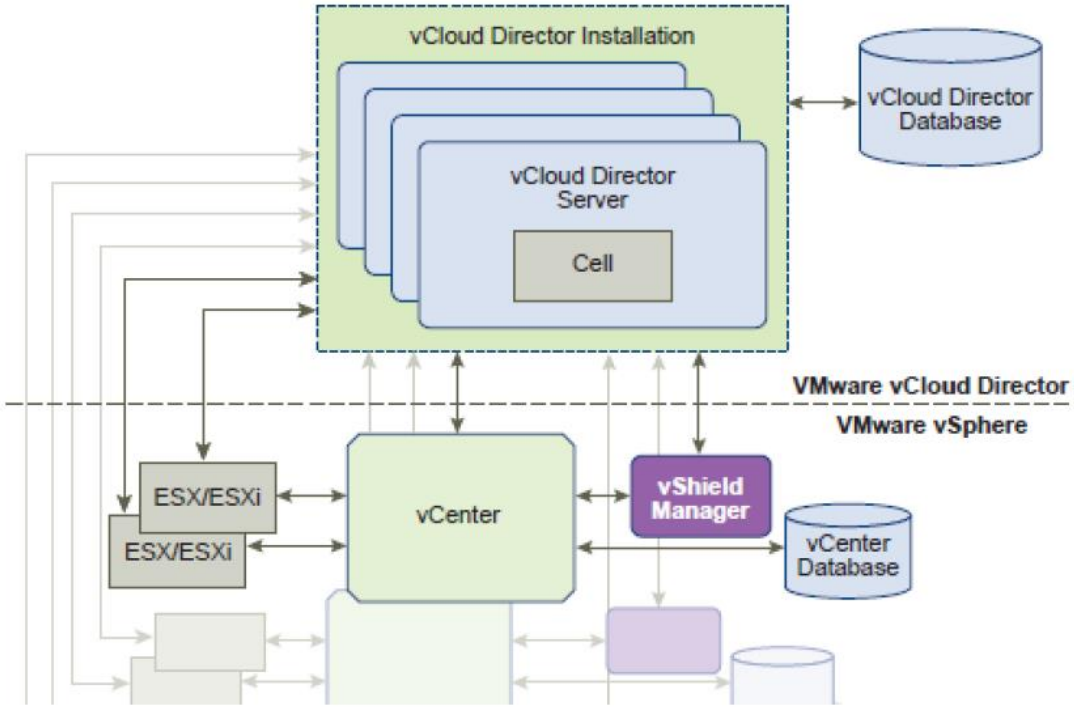

Figure 3 - vCloud Architecture (Source: VMware.com)

There can be multiple vCloud Director Servers in simple cloud deployment. These servers run a collection of services called a vCloud Director cell.

At least one VMware vCenter Server system, a VMware vCloud Networking and Security server, and one or more VMware ESXi hosts needs to be associated with vCloud Director server. For each vCenter Server system managed by vCloud Director, there must be one vCloud Networking and Security server.

All vCloud Director servers in the group share a single vCloud Director database. The group connects to one or more vCenter Server systems and the ESXi hosts that they manage. vCloud Networking and Security servers provide network security services and automatically deploy VMware vShield Edge virtual appliances on demand from vCloud Director. (VMware, 2014)

VMware ESXi is a Type 1 Hypervisor, runs on bare metal hardware. It is able to run many virtual guests on a single native server. Tesfamicael, et al., (2015) mention that, software, tool like VMware vCloud is a widely used solution to build and manage cloud infrastructure in most IT companies. vCloud is a layer of VMware based cloud suite, build on the VMware ESXi servers to create private cloud and

hybrid cloud platform. This is mainly used in multitenant environment, to pool hardware resources into virtual machines in datacentres. These infrastructure resources can be accessed by users with web-based tools. VMware vShield is another key component to provide complete security for data and application, improve visible control in virtual cloud (Tesfamicael, et al., 2015).

The VMware vCloud Director is a part of the VMware based cloud software bundle. The software stack of VMware consists of three layers: ESXi hypervisor, vCenter server and vCloud Director. This software stack is used to create a virtual cloud environment and access cloud services. The immense architecture of ESXi is specially designed to manage cloud resources and offer better performance of VMs with high consolidation (Padala, 2013).

2.3 OpenStack

OpenStack is primarily designed and developed by Rackspace and NASA (National Aeronautics and space Administration) (Mullerikkal, et al., 2015). The specific goal assigns to OpenStack is to create simple, standard and scalable cloud computing infrastructure. OpenStack offers Infrastructure as a Service (IaaS) to consumers with components like Glance, Nova, Horizon, Neutron, Keystone. A Nova component of OpenStack is used to handle VMs, including create, manage and delete VMs. Neutron helps to provide network based services, Glance is responsible to store images in the cloud while Keystone deals with authentication and authorization of security credentials (Rosado, et al., 2014). Sefraoui, et al. (2012) confer that, OpenStack bring out resource utilization with successfully configuring VM on the cloud. In case of Linux Container (LXC) hypervisor, VM owner will responsible to assign parameters to VM through OpenStack Cgroups. Yang, et al., (2013) research for dynamic resource allocation using live migration of VM in OpenStack cloud platform. This live migration can be achieved through dynamic resource scheduling with load balancing of VMs and hosts.

There are multiple Open Source cloud toolkits are easily available on Internet for example OpenStack, CloudStack and Eucalyptus. Among these solutions OpenStack is the most popular in IT industry. OpenStack is purely based on Open-source platform and enables IT sector to build and manage Infrastructure as a Service (IaaS) oriented cloud services. One major advantage of Open Source is, it gives transparent solution and avoid possibility of vendor lock-in. Open-source can easily utilize private cloud resources of consumer and offers better services through web portal (OpenStack, 2015).

2.3.1 OpenStack Components

OpenStack solution is a new foundation of datacenter tasks to perform multiple functionalities over the cloud. As OpenStack works as a modern software solution over traditional operating systems, it manages network, storage, and compute pool. OpenStack project is fabricated with some vital components and each one has service oriented and message passing feature (Lifflancer, et al., 2009). Each component in OpenStack architecture plays an important role. Advanced Message Queuing Protocol (AMQP) provides a communication path to these components. This protocol keeps message requests in queue and process them with short response time. AMPQ protocol avoids a process deadlock situation, where the process does not need to wait on another process for completion. A RESTful API is a basic element of each component of OpenStack architecture to expose tasks towards outside (Rodriguez, 2008).

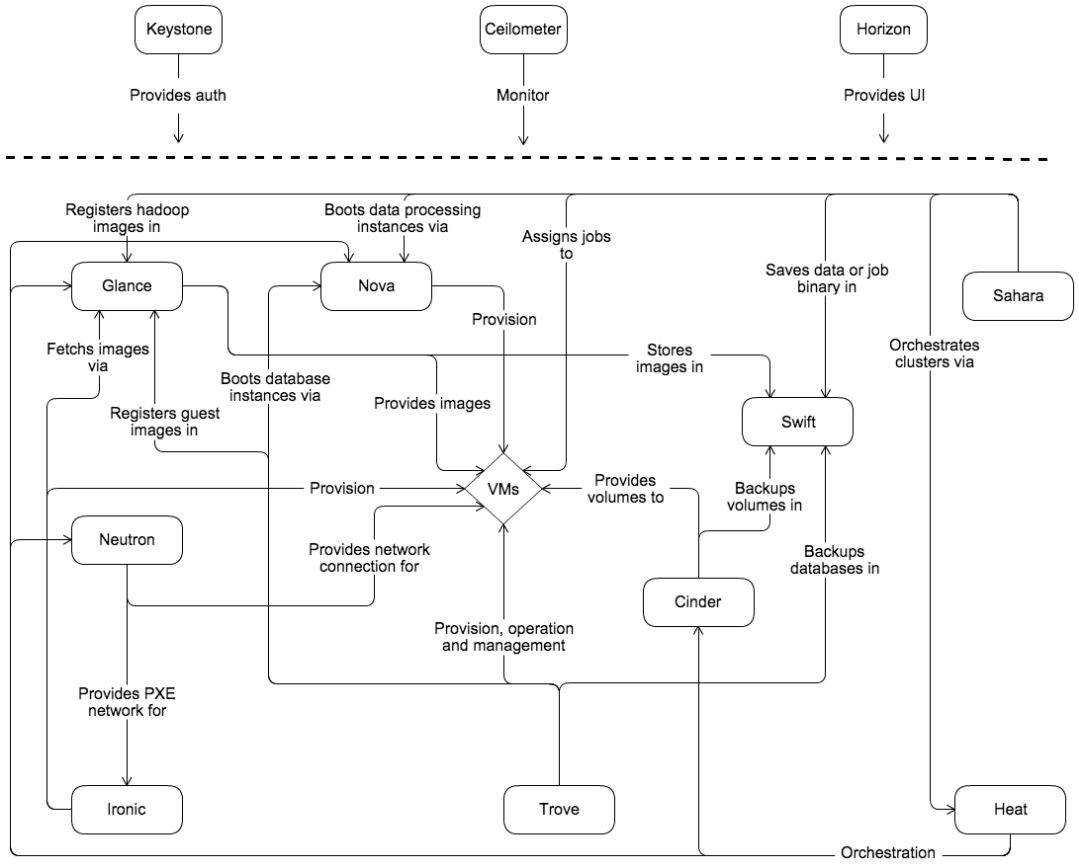

Figure 4 - OpenStack Architecture (Source: Openstack.org)

According to OpenStack (2015), key components of OpenStack is shown in above Figure. The introduction of each component is as follows:

2.3.1.1. Keystone - Identity Service

This element is known as a centralized identity integral to store user credentials like authentication and authorization details (OpenStack, 2017). To manage a Service Catalogue and maintain API register, keystone installs integrals with OpenStack software package into the system. The Service Catalogue in keystone is used to discover cloud services and their credentials, so they can run on authenticated node (OpenStack, 2017).

2.3.1.2. Nova - Compute Service

Nova is a well-known OpenStack service for cloud computing platform. It enables the provisioning and management of CPU resources. Nova acts as a controller to interface with Virtual Machine Monitor (VMM) or Hypervisor. Nova takes care of communication with different hypervisors using virtualization drivers. Every supported hypervisor has a driver associated with it (Shrivastwa & Sarat, 2015)

2.3.1.3. Glance – Image Service

Virtual Image or Glance is used to create a virtual image of physical machines. Glance is mandatory service. Without Glance, Nova service will not know where to pick its images. The central repository collects all virtual machine images for storing and managing purpose. The metadata of central repository contains the actual location of the virtual image in memory. Virtual images can be stored in the file system as well as Object Storage Service (OpenStack, 2017).

2.3.1.4. Neutron – Network Service

In OpenStack cloud, the cloud service provider has always controlled their services through IP addressing. In a multi-tenant environment, it is difficult to provide cloud services with high availability. Neutron service of OpenStack, offers the simplest way to connect virtual machines in cloud networking paradigm (OpenStack, 2015).

2.3.1.5. Cinder – Block Storage Service

Cinder creates an abstraction layer to manage device pool in an OpenStack based cloud infrastructure. Block level storage is mainly used to utilize workloads like a sensitive database or virtual machine. Cinders with suitable block driver is used to provide appropriate cloud services. The API (Application Programming Interface) for Cinder service grants consumers to request cloud resources without knowing its implementation (OpenStack, 2015).

2.3.1.6. Swift – Object Storage Service

Swift is an OpenStack service, works on object level, to store data this service called Object Storage service. The main goal of swift is to create secure, scalable and low cost paradigm of data provider. The data which is not changed regularly and rarely retrieved, like data usage in banks or government offices, such type of data is suitable for Swift (Rosado, et al., 2014).

2.3.1.7. Horizon – Dashboard

Horizon is a Dashboard interface of OpenStack. It provides graphical user interface for OpenStack components. It relies on the OpenStack APIs to present much of this functionality. Horizon is Django web application (Campbell, 2016).

2.3.1.8. Heat – Orchestration

Heat is the OpenStack orchestration service. It automatically configures and deploys resources in stacks. Templates are used to define Orchestration stacks. Templates describe tasks in terms of resources, parameters, inputs, constraints and dependencies. The Orchestration service also runs Heat Orchestration Template (HOT) templates that are written in YAML (OpenStack, 2016).

2.3.1.9. Ceilometer – Telemetry

Telemetry service was originally designed for OpenStack cloud resources billing systems support. Service collects system information and stores in the form of samples. These samples provide data about anything that can be billed. It also captures event notifications against various actions that are happening the system. List of meters are increasing that can be used to perform different actions other thank billing (OpenStack, 2016).

2.4 OpenStack Vs VMware based Cloud

According to Sahasrabudhe & Sonawani (2014), cloud computing is a new platform for resource utilization. There is increasing use of VMware based cloud and OpenStack cloud solutions. The main advantage of OpenStack components is; it allows to build hybrid cloud infrastructure. Cloud resources are consisting of computing, network and storage. These resources can be utilized in a way that, end user can achieve better performance and cloud provider can gain resource utilization. OpenStack is a well-known cloud based platform used to build Infrastructure as a Service (IaaS). VMware based cloud products that can use directly to map OpenStack services are vCloud Director and vCloud Automation Center. Although, OpenStack toolkit does not include its own hypervisor, but it allows Xen, ESXi and KVM hypervisors.

2.4.1 Basic Approach

OpenStack is an open source platform to provide cloud computing services to consumers. Its toolkit includes, storage, networking and compute technologies, as well as it supports many hypervisors like Hype-V, KVM, ESXi and Xen. On the other side VMware ESXi is a well-known hypervisor but not a cloud platform or toolkit. vCloud Automation center or vCloud director are the VMware products that map to OpenStack. (Xie, 2013).

2.4.2 Installation

When installing VMware vCloud the experience is very similar every time you install. It has solid documentation, a well-established hardware compatibility.

Installing OpenStack, where weeks can be spent researching on the latest release or finding solutions to undocumented installation challenges. It is not uncommon for those tasked with the deployment of Openstack to participate in developer level communication channels including IRC and mailing lists (Dafni, 2015).

2.4.3 System Convergence

The single vendor “full stack” nature of VMware vCloud provides integrated service layers that are necessarily very well optimized to operate together. There is no opportunity to mix these layers; instead, the full stack is written, tested, sold, and supported as a single product. This necessarily means that everybody who uses the VMware vCloud solution is using the same software stack.

OpenStack is a suite of discrete products that can be implemented piecemeal or as a whole. The fact that very few organizations run the same OpenStack software stack as each other means that convergence is less of a focus for developers than is each product’s functionality and stability as a single unit (Dafni, 2015).

2.4.4 Access Type

OpenStack is using Open API or command line interface to allow customers to access web services. Horizon-Dashboard is a widely used customer access component of OpenStack. VMware allows web access through Web console, API and Windows client (Sahasrabudhe & Sonawani, 2014).

2.4.5 Network and Storage Component

OpenStack allows network switching and SDN (Software Defined Network) pluggable extensions such as OVS (OpenFlow software switch) (Al-Najjar, et al., 2016). VMware based cloud supports network

switching with VMware NSX, in the SDN network environment. For storage purpose OpenStack uses the Pluggable Cinder component. Whereas, VMware based cloud uses the SAN (Storage Attached Network) and iSCSI (Internet Small Computer System Interface) (Sahasrabudhe & Sonawani, 2014).

2.4.6 Image Management

OpenStack Glance service is specially developed for image management. Glance allows to create custom types of images and flavors. On the other hand, in vCloud director Catalogs are used and vAPP templates and OVF files can be uploaded. (Sahasrabudhe & Sonawani, 2014).

2.4.7 Scheduling

OpenStack includes scheduling component in its toolkit called the Nova-scheduler. Nova-scheduler, like vSphere DRS, makes decision on initial placement of VM using metrics like compute resources that are available. But it has not the capability to do load-balancing or power management of VMs. DRS dynamically balances computing capacity across compute resources and intelligently allocates available resources among the virtual machines based on predefined rules that reflect business needs and changing priorities (Sahasrabudhe & Sonawani, 2014).

3. NMS Introduction

3.1 NMS Physical Deployment

Ericsson's Network Management System called Ericsson Network Manager (ENM) is the next generation Ericsson product designed to provide a unified network management system for multi-technology and heterogeneous networks.

It offers greater network upgrade flexibility with zero downtime, the capacity to scale from very small to very large networks, and an intuitive User Interface which provides common tools for all applications with extensive integrated help and videos simplifying common network operations.

Ericsson Network Manager (ENM) provides powerful and unified performance and configuration management, software, hardware, and fault management, as well as security, self-monitoring, and system administration.
It offers simplified network management with advanced querying and filtering capabilities, shorter troubleshooting lead times, and faster implementation of network changes.

ENM is deployed on Linux IT Platform (LITP), a lightweight plugin-based platform that offers a robust, simple, consistent, and automated way to install, upgrade, and execute all software and 3PP in a variety of environments.

The following figure illustrate typical network layouts of 11 blade ENM 40K deployments. 40K is the number of nodes supported in this hardware layout.

A typical 11-blade layout contains the following blades:

- 6 blades in a Service cluster
- 3 blades in a Scripting cluster
- 2 blades in a Database cluster

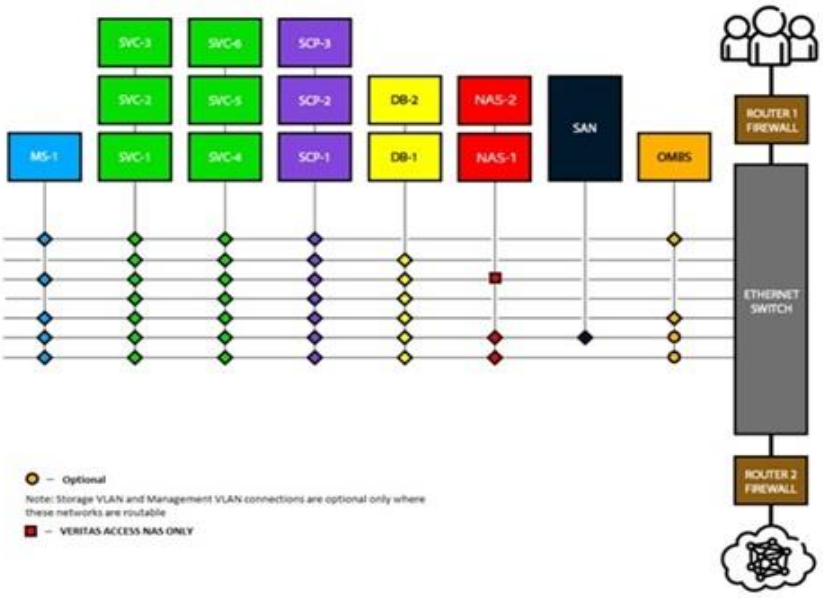

Figure 5 - ENM Hardware Layout

The following figure show the configuration of Virtual Machines (VMs), databases, and services located on each node for specific deployments.

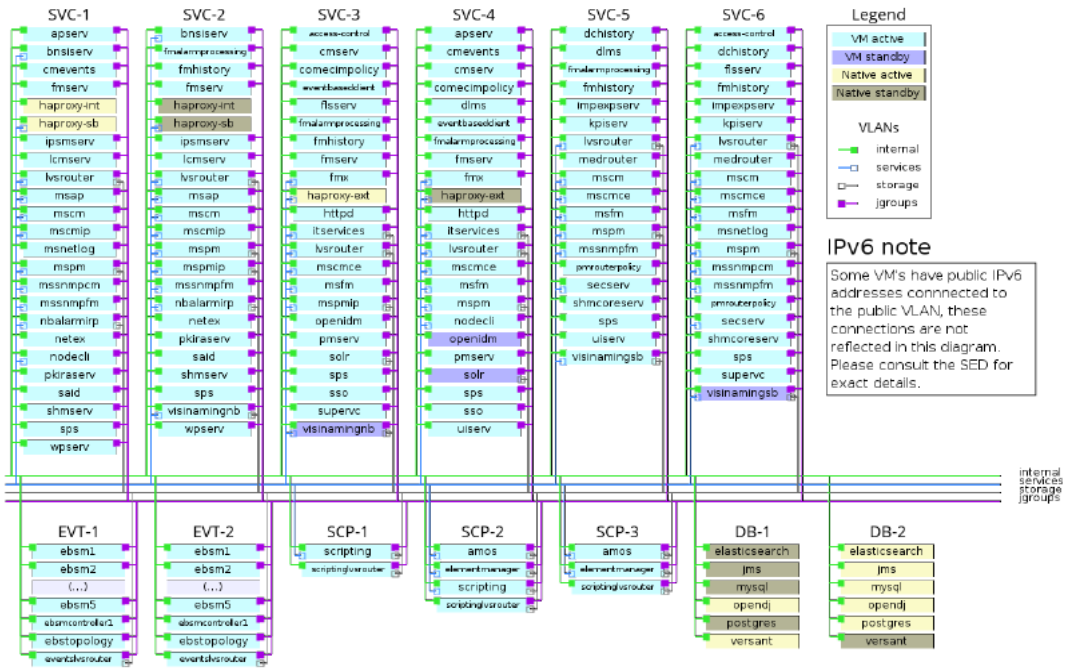

Figure 6 - ENM KVM Layout

4. VMware vCloud Architecture Design Overview

4.1 vCloud Components

The following figure shows the components that comprise the VMware vCloud and how they are interacted.

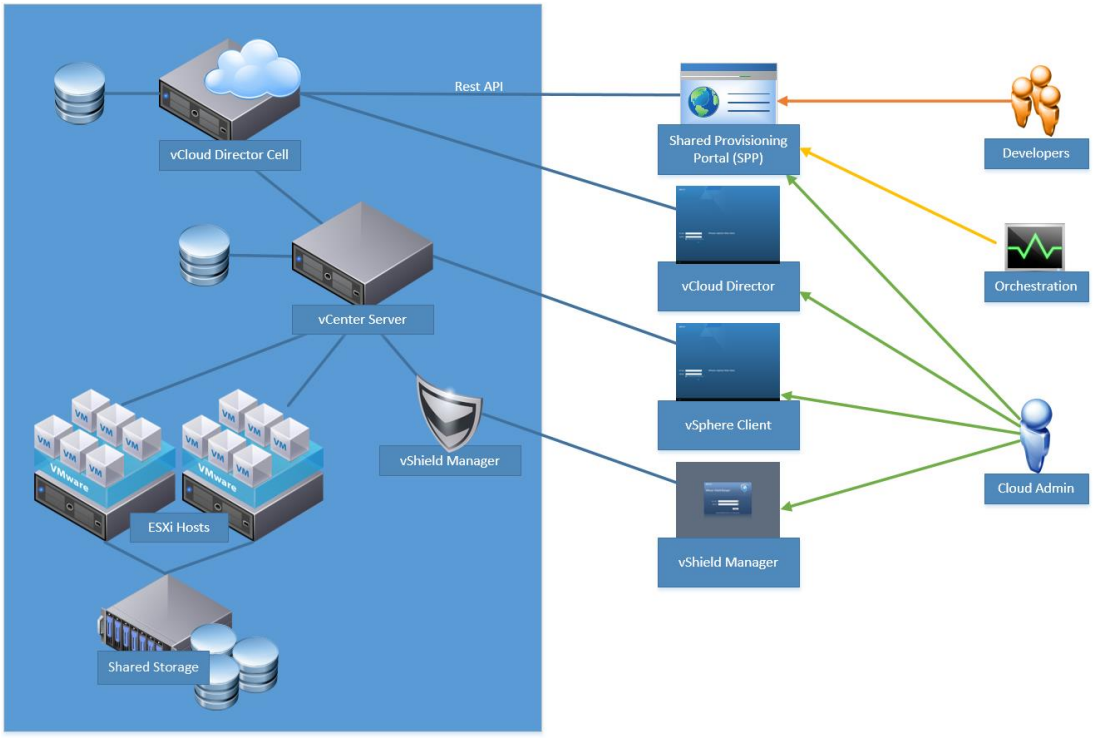

Figure 7 - vCloud High Level Layout

Following table shows high level description of VMware based cloud solution's components and customized Self Provisioning Portal

vCloud Component	Description
VMware vCloud Director	Software layer that abstracts virtual resources and exposes cloud artifacts to consumers. Includes: • VMware vCloud Director Server (one or more instances, each installed on a Linux VM and called as as a *cell*). • VMware vCloud Director Database (one instance per clustered set of vCloud Director cells).
VMware vSphere	Platform that provides abstraction of physical infrastructure layer for vCloud. Includes:

	• VMware ESXi hosts (different clusters for management and cloud workloads) • VMware vCenter Server (multiple instances managing management clusters and resource groups reserved for vCloud consumption) with databases, single sign-on instances and web client servers
VMware vCloud Network and Security (vShield)	Provides perimeter network security for virtual datacenters with Edge Gateways. Includes: • Edges (deployed automatically as virtual appliances on resource group hosts by vCloud Director). • vCloud Networking and Security (vCNS) Manager (one instance per resource group vCenter Server).
Self-Provisioning Portal (SPP)	In house-build tool with web interface to create/delete their vApps. Add vApps and ISO images to catalog.

Table 1 - vCloud Components

4.2 vCloud Conceptual Architecture

4.2.1 vCloud Logical Design

vCloud infrastructure is divided into two main building blocks.

- **Management Pod -** contains the core and optional components and services required to run vCloud instances. Core vCloud components such as VMware vCenter Servers and vCloud Director Cells are some of the examples.
- **Resource Pod** – Represents vCloud-dedicated resources for end-user consumption. Resource group includes vSphere clusters (vSphere hosts managed by a vCenter Server), under the control of vCloud Director. The following table provides an overview of the management and resource clusters for the design.

vCloud Pod	Description
Management Pod	Cluster dedicated to running management servers including: • vCenter Server Appliances • ESXi hosts • vCloud Director cells and associated database schemas • vCNS Manager for vCloud Director • vCNS Edge Devices • VMware Update Manager Servers • NFS Servers
Resource Pods	Clusters dedicated to running end-user servers: • ESXi hosts • User-created virtual machines

Table 2 - vCloud Infrastructure Layout

4.2.2 Management Components

A dedicated management cluster that does not share resources with any other solution is used, and a separate vCenter Server is used to manage the vCloud Management Pod. This provides stability and reliability for the management components.

The management components are the server components, deployed as virtual machines, which are required to operate a vCloud solution. The following list summarizes the management components installed as VM within the management cluster.

- vCenter Server
- vCenter Update Manager
- vCNS Manager appliance
- vCloud Director cells
- NFS server

- vCloud Microsoft SQL Database
- vCNS Edge Appliance (Load Balancer)

4.2.3 Compute Logical Design

The compute logical design contains the resources provided by the underlying ESXi hosts clusters. The management cluster will be an ESXi hosts cluster that is dedicated to running the servers required to manage the vCloud infrastructure.

The vSphere cluster in the Management Pod is dedicated to running servers required to manage the vCloud environment. Similarly, vSphere cluster in resource cluster is dedicated to running end user's guest machines.

VMware High Availability (HA) is configured on the management and compute clusters to provide recovery of virtual machines in the event of an ESXi host failure.

4.2.4 vCenter Server Design

Installing a vCenter Server creates the central point for configuring, provisioning, and managing virtualized IT environments. It must be install before we can add the hosts and data centers to be managed and monitored.

vCenter Server is deployed as a virtual machine within the Management Pod. This leverages the benefits available, such as HA, that protects the vCenter Server virtual machine in the event of hardware failure.

vCenter Update Manager is implemented as a component part of this solution for monitoring and managing the patch levels of the ESXi hosts only. A Microsoft SQL database server is deployed to support the vCenter Update Manager service. The default Microsoft SQL Server 2008 Express installation is only supported for up to 5 hosts, so Microsoft SQL Server 2012 Standard is installed.

The Microsoft SQL database is deployed on the same Windows Server as vCenter Update Manager.

4.2.5 vCloud Director Design

4.2.5.1. vCloud Director Cells

Two vCloud director cell per POD are used. Each cell is automatically assigned a role. When load balancer receives communication request, they fall into four categories.

User Interface (UI). This is the main Web console that administrators and operators use to manage vCloud Director.

API. The API consists of commands that can be issued to vCloud Director from other systems and scripts through the API. Some commands and functions can only be issued though the API.

Virtual Machine Remote Console (VMRC). This is the pop-out console that an operator can open on any virtual machine running in vCloud Director.

Image Transfer. This is the system that allows files and images like .ISO files to be uploaded into vCloud Director.

A master cell (selected by vCloud Director) coordinates the role assignment to vCloud Director cells

4.2.5.2. vCloud Director database

vCloud Director cells point to a shared database where all configuration and state data is stored. If the database is lost or unavailable, the entire vCloud instance is also down.

4.2.5.3. Transfer Storage

When multiple vCloud Director cells are used a shared temporary storage for uploads, downloads and cloning operations across different vCenters has to be available to all vCloud Director cells.

4.2.5.4. Networking

vCloud Director has requirements with respect to network connections. The first of these requirements is to provide dedicated connections for the HTTP and Console Proxy services. In addition, it is common to have additional connections to management and potentially dedicated storage networks for access to NFS storage. The following figure provides an overview of the network configuration used for this design.

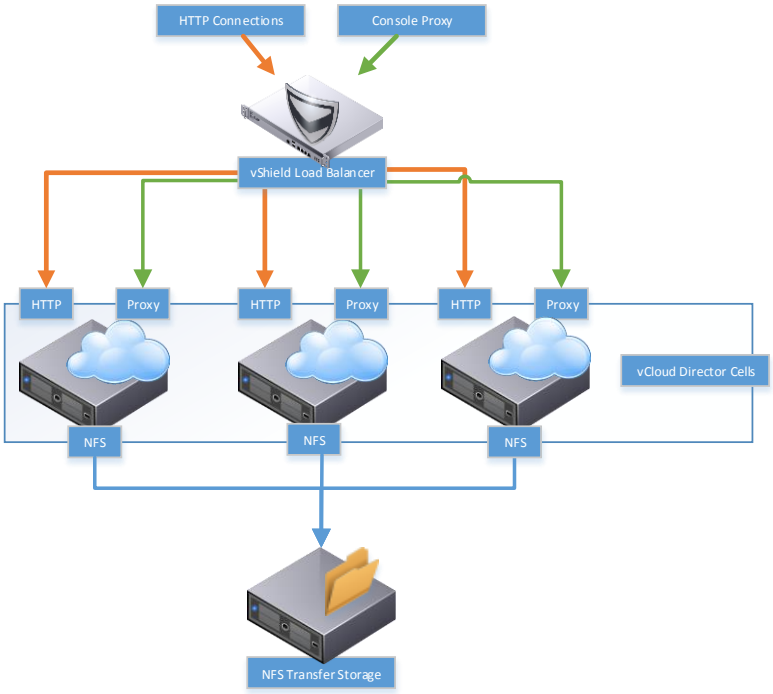

Figure 8 - vCloud Network (Source: VMware.com)

In the case of this design the vCloud cells are deployed in conjunction with a pair of VMware vCNS Edge Gateway load balancers to manage requests between the HTTP and Console Proxy connections of the cells.

4.3 VMware vSphere Architecture Design – Resource Cluster

Hardware pod design for the datacenter consisting of 7 HP C7000 chassis, L2 networking to the End of Rack switch and a fiber channel Storage Area Network (SAN). HP C7000 chassis has space for 16 blades, which gives a total of 112 blades per hardware pod.

To mitigate the risk of failure at the chassis level, four blades from each chassis are used to create each of the clusters. Out of 4 clusters in POD, one cluster has 28 hosts and 3 clusters has 27 hosts. Remaining 3 hosts are used in management cluster.

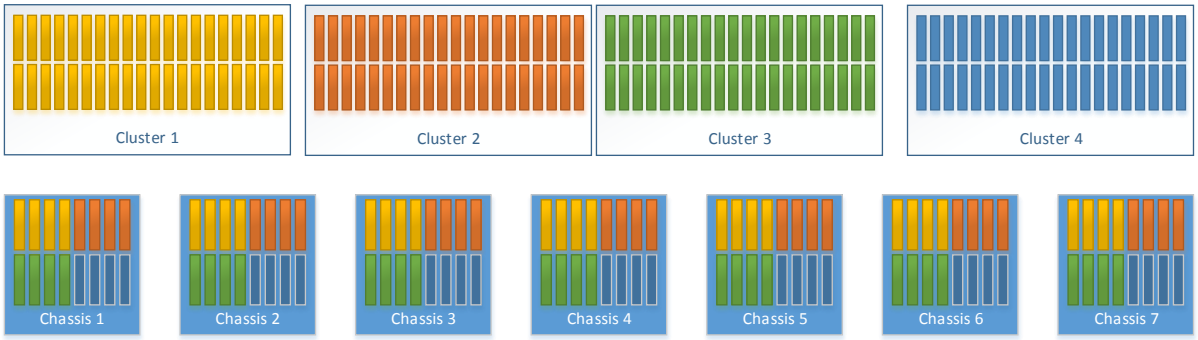

Figure 9 - Resource Cluster Layout

4.4 Deployment Flow

Here is the deployment flow at higher level

- Install ESXi hosts via a PXE Boot
- Create Cluster for Management POD
- Add ESXi Hosts to Management POD
- Remediate Hosts to create a baseline for VMware ESXi patches
- Create Host Profile
- Install the VMware Management Stack
- SPP Installation
- Compute Stack Installation
- Storage setup
- Compute Stack Preparation
- vCloud Director Configuration

4.5 Application deployment through vApp Templates

Currently VMware vCloud environment is used by developers to run and test the application. There is little or no effort to run this application in VMware based cloud environment. So application running on VMware based cloud can be called a cloud enabled application. Similar architecture and networking is used as we have in physical deployments. Two types of templates are created for developers. One template gives full functionality of the NMS application with high availability running and all KVMs are deployed across Services nodes (SVC1 to SVC4 VMs). This template is used for functional testing and high availability testing of the application. But it is not used for any load test. This template consists of some application supporting VMs like Selenium VM that automatically test the application's GUI

functionality and TAF VM that is test automation framework that does the functional testing of the application after deployment.

Another small vApp template called SSGID (Small Service Group Instance Deployment) is used for functional testing. It runs single instance of every KVM running in the application.

As a CI flow, Jenkins job will take the blank template with no application installed on VMs, install the latest version of the application. If install is successful, vApp will be added to catalog for all the teams to deploy their vApp from newly added template. If installation is failed, then it is not added to catalog and vApp is kept for troubleshooting purposes to find the root cause of the problem.

5. Redhat Openstack Cloud Overview

5.1 Introduction

Redhat OpenStack is based on OpenStack project TripleO, which is an abbreviation for "OpenStack-On-OpenStack". TripleO uses OpenStack components to install an operational OpenStack environment. It includes new OpenStack components that provision and control bare metal systems to use as OpenStack nodes. Red Hat OpenStack Platform uses two main concepts: An Undercloud and an Overcloud. The Undercloud installs and configures the Overcloud (Redhat, 2016).

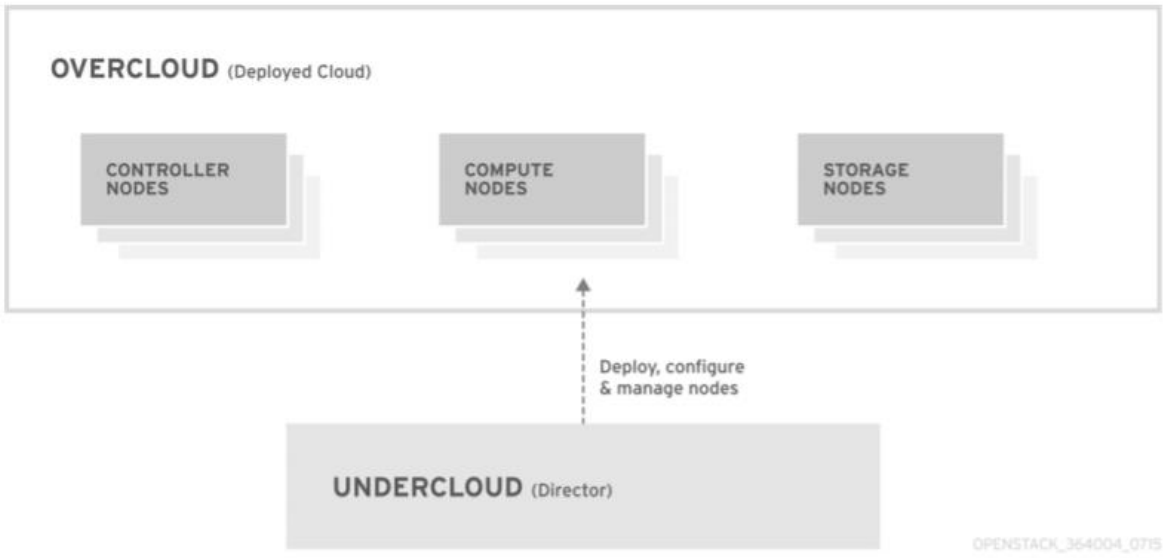

Figure 10 - Redhat OpenStack Platform (Source: Redhat Platform 9)

The Undercloud is also called director node. Director node is a single-system OpenStack installation that includes components for provisioning and managing the OpenStack nodes that form working OpenStack environment (the Overcloud).

Red hat Openstack Platform 9 director machines run on RHEL 7.2. In our implementation we have used KVM as director machine that is installed on Rackmount DL380 server. Active Red Hat subscription is also required for deployment. Detailed instruction for deployment can be found in Red hat documentation. Following figure shows tripleO deployment view.

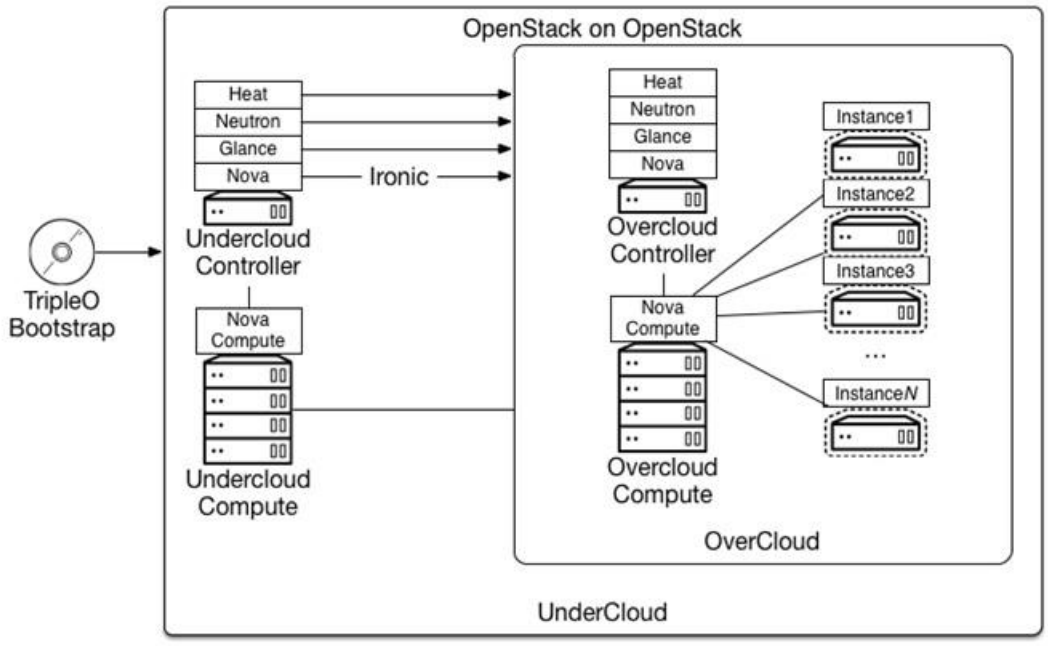

Figure 11 - Openstack on OpenStack (Source: TripleO.org)

Overcloud is defined with heat templates. There are standard templates are provided. We can customize for our environment. TripleO bootstrap contains overcloud images that are installed on compute and controller nodes.

Following figure shows high level physical infrastructure layout that we are using to build Redhat OpenStack cloud.

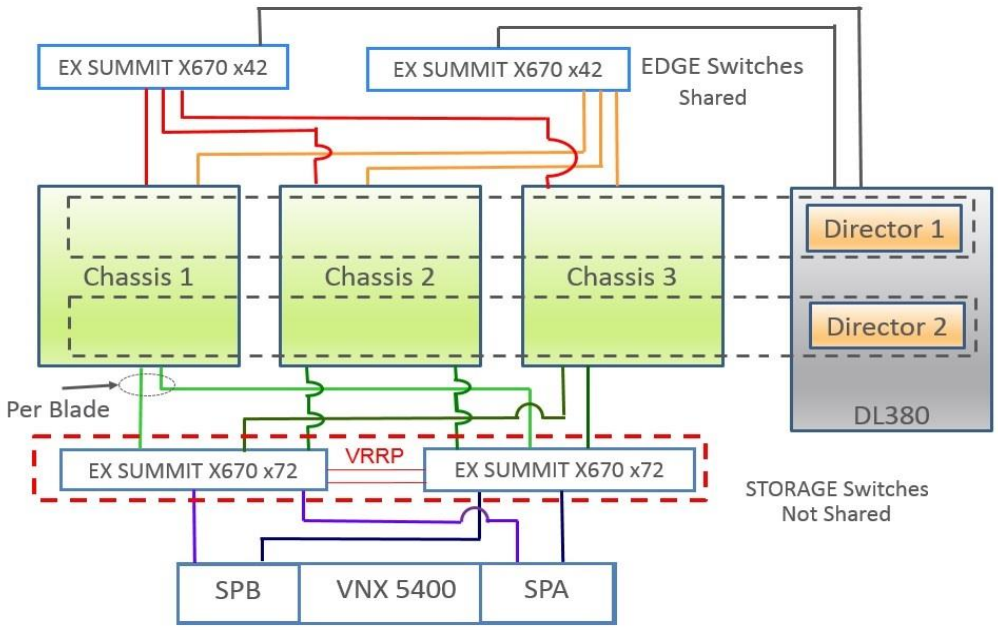

Figure 12 - Redhat OpenStack Network Layout

One Openstack POD consist of 24 HP blades split across three HP C7000 chassis. Each POD consist of 3 controller nodes and 21 compute nodes that are connected to backend Dell EMC VNX SAN storage through iSCSI. Following figure shows hardware layout in HP C7000 chassis.

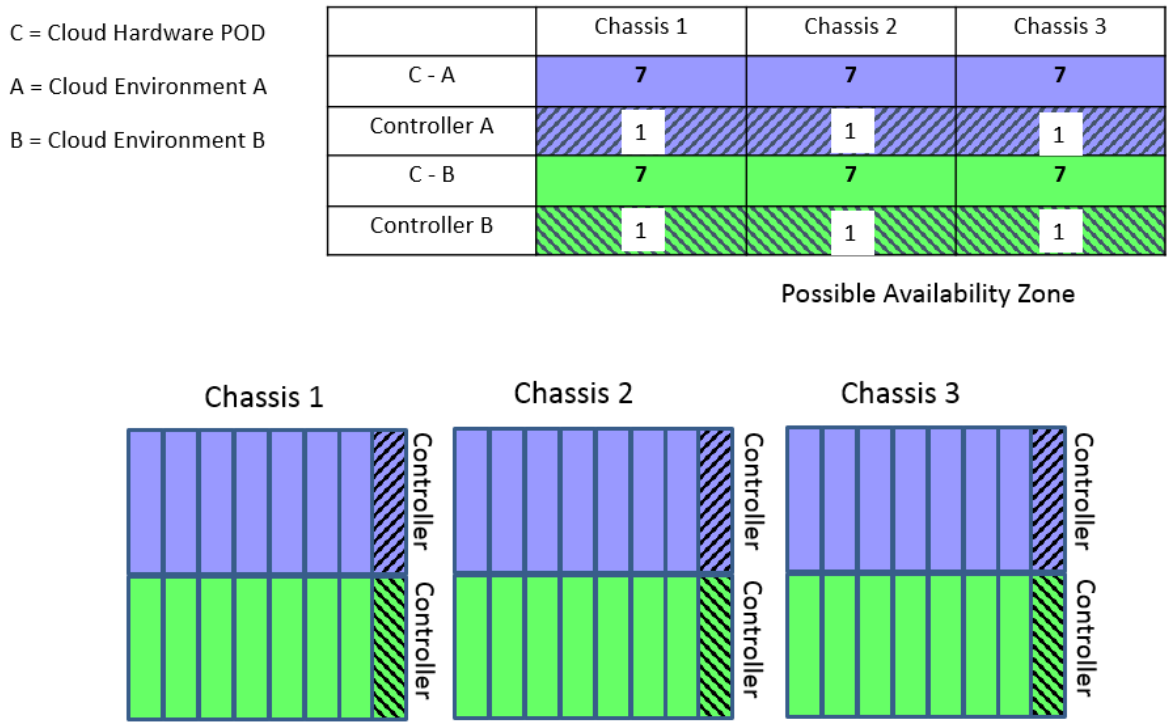

Figure 13 - Compute and Controller split across chassis

5.2 Installation Overview

At high level Redhat OpenStack deployment consist of following flow

- Install an Operating System to the undercloud host
- Prepare the undercloud services
- Install the overcloud services
- Prepare for overcloud deployment
- Deploy the overcloud
- Post Deployment tasks (Verification, Create Tenant, Create Users etc.)

5.3 Undercloud Deployment

5.3.1 IPMI connections to overcloud servers

The introspection (hardware discovery) and the whole deployment uses Intelligent Power Management Interface (IPMI) as a way to control the power cycle of the servers. Make sure you can run ipmi commands from the Undercloud before running the introspection to avoid potential errors. This command is an example should return some information about the status of the server.

```
# ipmitool -I lanplus -H 10.87.246.64 -U <ipmi user> -P <password> chassis status
     System Power         : on
     Power Overload       : false
     Power Interlock      : inactive
     Main Power Fault     : false
     Power Control Fault  : false
     Power Restore Policy : previous
     Last Power Event     :
     Chassis Intrusion    : inactive
     Front-Panel Lockout  : inactive
     Drive Fault          : false
     Cooling/Fan Fault    : false
     Front Panel Control  : none
```

5.3.2 Registering Nodes in Ironic

The *instackenv.json* (/home/stack/insatckenv.json) is used to register bare metal nodes in ironic. The following example shows node information of one server.

```
{
       "nodes":[
```

```
            {
                "mac":[
                    "bb:bb:bb:bb:bb:bb"
],
                "cpu":"40",
                "memory":"262144",
                 "arch":"x86_64",
                "pm_type":"pxe_ipmitool",
                "pm_user":"cloud",
                "pm_password":"p@55w0rd!",
                "pm_addr":"10.141.4.159"
      ]
  }
}
```

Once that is done, we can register and configure the nodes by issuing:

```
$ openstack baremetal import --json ~/instackenv.json
$ openstack baremetal configure boot
$ Ironic node-list
```

5.3.3 Introspection

This is the step where the Undercloud "introspects" or inspects the hardware where the Overcloud is going to be deployed.

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

This command will start a process that will take a couple of minutes (the servers will reboot and PXE a ramdisk based image and will be powered off again). It will take some time depending how many nodes you are inspecting. In our case we have 3 controller and 21 compute nodes. If everything is OK it should finish in 30 minutes.

We can check the progress by executing below command

```
$ sudo journalctl -l -u openstack-ironic-inspector -u openstack-ironic-inspector-
dnsmasq -u openstack-ironic-conductor -f
```

When introspection completes, the introspection status should show "Finished = True"

```
(openstack) baremetal introspection bulk status
+--------------------------------------+----------+-------+
| Node UUID                            | Finished | Error |
+--------------------------------------+----------+-------+
| 58bf2451-ce26-4d8e-a44f-3cf8be8c7776 | True     | None  |
| 5a067218-aeb2-4477-a793-5ed2306dc4cc | True     | None  |
| ee5bc84a-06c6-4b45-84f0-6b17993ca153 | True     | None  |
| 380f4184-09d4-44a1-bc58-4e8401baf57a | True     | None  |
| 0b6f68fb-f5c9-412a-a640-1375e9a7b6b1 | True     | None  |
+--------------------------------------+----------+-------+
```

Figure 14 - Ironic Hardware Introspection

5.4 Overcloud Deployment

Before running the Overcloud deployment, make sure that the nodes provisioning state is available and not in maintenance mode.

```
$ ironic node-list
+--------------------------------------+-----------+--------------------------------------+-------------+--------------------+-------------+
| UUID                                 | Name      | Instance UUID                        | Power State | Provisioning State | Maintenance |
+--------------------------------------+-----------+--------------------------------------+-------------+--------------------+-------------+
| 58bf2451-ce26-4d8e-a44f-3cf8be8c7776 | dl360-406 | 42788271-e79f-418a-be70-0db4e304406a | power off   | available          | False       |
| 5a067218-aeb2-4477-a793-5ed2306dc4cc | dl360-405 | 39b1aac2-4afb-4f13-8681-feb06a333a6d | power off   | available          | False       |
| ee5bc84a-06c6-4b45-84f0-6b17993ca153 | dl360-428 | 1a431cc3-19c8-43a0-b3b0-02882064ab0f | power off   | available          | False       |
| 380f4184-09d4-44a1-bc58-4e8401baf57a | r630-026  | 0eb23ee2-05e4-43d5-8cbc-8f7fc0d96ac0 | power off   | available          | False       |
| 0b6f68fb-f5c9-412a-a640-1375e9a7b6b1 | r630-027  | 6165e35f-42bb-4079-a1be-6a972e6fd418 | power off   | available          | False       |
+--------------------------------------+-----------+--------------------------------------+-------------+--------------------+-------------+
```

Figure 15 - Overcloud Nodes Provisioning State

Overcloud deployment consist of customized YAML files according to your environment.

```
$ openstack overcloud deploy \
--verbose \
--templates  ~/templates/openstack-tripleo-heat-templates/ \
-e ~/templates/openstack-tripleo-heat-templates/environments/network-isolation.yaml
\
-e ~/templates/nic-configs/network-environment.yaml \
-e ~/templates/hostnames/HostnameMap.yaml \
-t 240 \
--validation-errors-fatal \
--control-scale 3 \
--compute-scale 21 \
--compute-flavor compute \
--control-flavor control \
--ntp-server 159.107.173.12
```

Here is the brief summary of all the options mentioned above

--templates – Overcloud is created using the Heat template collection in */usr/share/openstack-tripleo-heat-templates*

-e option adds an additional environment file to the Overcloud deployment

--control-scale 3 – Create three controller nodes.

--compute-scale 21 – create twenty-one compute nodes

--ntp-server 159.107.173.12 - Use an NTP server for time synchronization. This is useful for keeping Controller node cluster in synchronization.

-t 240 – deployment timeout in minutes

--compute-flavor compute – The flavour to use for compute nodes

--control—flavor control – The flavour to use for control nodes

--validations-errors-fatal - Pre deployment checks are performed as part of Overcloud creation process. If there are errors in pre-deployment checks this option will exit the deployment process.

A script is generated by director to configure and help authenticate interactions with Overcloud from the director host. This script file, overcloudrc, is saved in stack user's home directory. We run the following command to use this file.

```
$ source ~/overcloudrc
```

When above command is executed, necessary environment variables are loaded to interact with Overcloud from the director host's CLI. If we want to access undercloud, run the following command:

$ source ~/stackrc

If overcloud deployment finished successful we can check the status from director machine

```
$ openstack stack list
+--------------------------------------+------------+-----------------+----------------------+-
--------------------+
| ID                                   | Stack Name | Stack Status    | Creation Time        |
Updated Time        |
```

```
+--------------------------------------+------------+-----------------+----------------------+-
--------------------+
| c6765469-78ee-4297-bd55-47996f300bcb | overcloud  | UPDATE_COMPLETE | 2017-02-16T10:42:39 |
2017-02-16T12:27:59 |
+--------------------------------------+------------+-----------------+----------------------+-
--------------------+
```

5.5 Application deployment through Heat Templates

NMS application in OpenStack cloud is being redesigned to make it cloud-native. KVMs that run on physical hardware in physical deployment and on VMware VMs in VMware environment, are deployed as standard Linux instances. There is no Veritas cluster running to provide high availability in OpenStack environment. NMS application is deployed as heat templates, that deploy different stacks in OpenStack cloud for applications to be installed and running. Ericsson's own built orchestrator called VNF-LAF (Virtual Network Function, Life Cycle Automation Framework) is used to orchestrate whole application deployment. This tool is also called LCM (Life Cycle Manager). Currently this tool is in under development and only supports Application deployment. Following figure shows the VNF-LAF GUI.

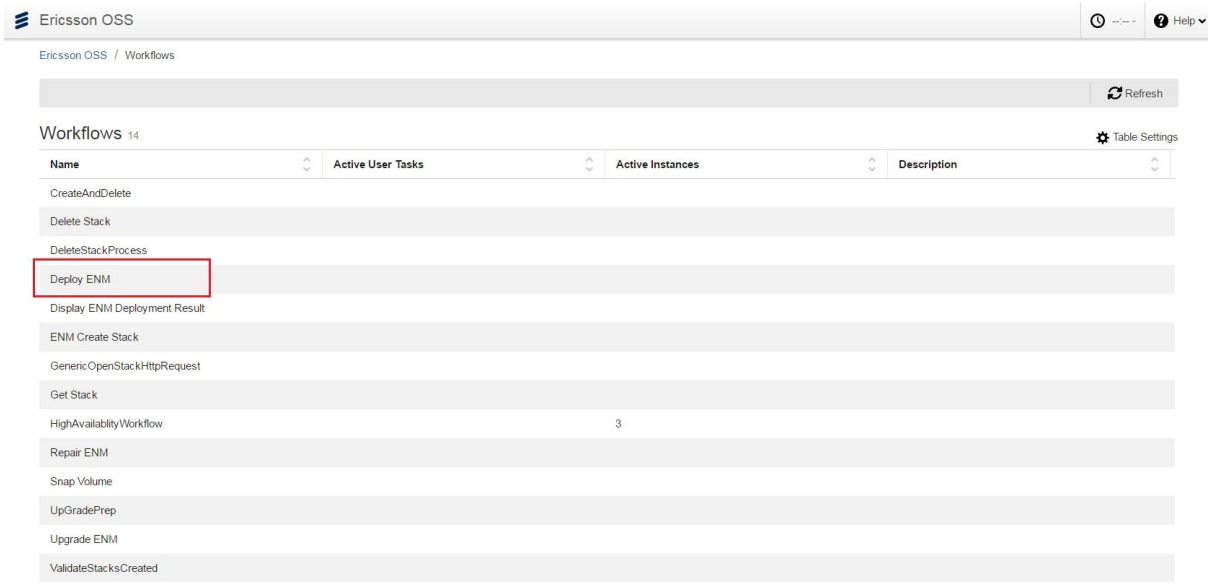

Figure 16 - VNF-LAF Orchestrator

Here is the flow of application deployment in OpenStack Cloud.

- Access OpenStack Client
- Upload Images to Glance Image store either CLI or through command line.
- Deploy VNF-LAF instance from heat template. During LAF instance deployment SED (Site Engineering Data) YAML file is also provided.
- Access the VNF-LAF GUI and click on Deploy ENM as highlighted in figure-15.

- If everything is OK, we will have running application in couple of minutes.

6. Openstack Challenges

6.1 Director Node failing to boot after Installation - Deployment

In Redhat Openstack solution director is Undercloud node that is responsible for installation of actual cloud i.e. overcloud. It was observed that director node was not booting upon reboot.

After investigation it is found that this is known bug in Redhat bugzilla (Peeler, 2016). If the director machine has more than 50GB disk space and it is installed through standard installation procedure, then RHEL installation auto-creates logical volume (LV) for home partition. Director installation disables auto activation of volumes.

This doesn't affect systems that only have root and swap LVs because these get activated when the kernel starts:

```
linux16 /vmlinuz-3.10.0-327.10.1.el7.x86_64 root=/dev/mapper/rhel_ospd8test-root ro
rd.lvm.lv=rhel_ospd8test/root rd.lvm.lv=rhel_ospd8test/swap rhgb quiet
LANG=en_AU.UTF-8
```

Additional volumes don’t get activated upon reboot, the boot process will attempt to find and mount the home LV, hang for a minute and half, then timeout and drop you to the emergency shell.

There are different recommendations given in Bugzilla, but the easiest is comment out following line in /etc/lvm/lvm.conf

Before

```
#auto_activation_volume_list = []
```

After

```
auto_activation_volume_list = []
```

6.2 PXE does not work during introspection - Deployment

Director runs introspection process on overcloud nodes (compute, controller etc.). Each node boots introspection agent over PXE from that process. Hardware data is collected from nodes and sent back to the director. Data is stored in swift object storage service running on the director. This hardware information is used for benchmarking, profile-tagging (compute, controller), and root disk assignment.

Director requires a node definition template file as mentioned in *section 5.3.2*. This template file contains hardware and power management details for nodes and uses JSON format file.

HP iLO interface provides out-of-band remote management features including power management and server monitoring. As our solution uses HP blade server, so per Redhat documentation (RHOSP9 Power Management Drivers) it should use *pxe_ilo* power management type (pm_type). But issues were faced and nodes were unable to pxe boot with *pxe_ilo* pm_type. HP support could not find any reason for the root cause of the problem. Changing it to standard *pxe_ipmitool* solved the issue and nodes were able to pxe boot.

This is the change in instackenv.json file

From

```
"pm_type":"pxe_ilo"
```

To

```
"pm_type":"pxe_ipmitool"
```

6.3 Misconfigured Cinder volume - Deployment

Cloud deployment is using one common Dell EMC VNX 5400 storage for two different OpenStack deployments. This can be seen in in Figure-12. There is storage pool created for each cloud deployment. As part of installation, EMC VNX storage driver is installed so that cinder can access VNX storage. Redhat Openstack cinder installation assign IQN (iSCSI qualified name) to controller nodes. This IQN can be checked from */etc/iscsi/initiatorname.iscsi* in controller node.

```
$ cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:a3446c45c8cd
```

Same name is assigned to all three controller nodes. But it is not a problem as cinder service runs only in one controller node in high available controller environment.

Meanwhile another Redhat Openstack cloud environment is deployed that is also accessing same VNX storage but different storage pool. Cinder assigned same IQN to controller nodes that it assigned to controllers in previous deployment.

If there is volume creation request on VNX storage, it assigns volume name based on initiator IQN. If there is already volume created from cloud 1 and request comes from cloud2 for volume creation, It

finds volume is already existing so the response will go back to cloud 2 with acknowledgement for a volume that it cannot access. Now if we try to create any image on that volume it will fail because cloud 2 has no access to that volume.

After analysis it is found that same IQN across two deployments is causing the issue. So we assigned unique IQN to controller nodes in one of the deployment and after that it worked fine.

```
$  cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:podcbctl0
```

6.4 Failed Hardware Identification - Operation

When overcloud nodes are deployed from director machine they get name assigned in periodical order as shown below

```
$ openstack server list -c ID -c Name
+--------------------------------------+---------------------------------+
| ID                                   | Name                            |
+--------------------------------------+---------------------------------+
| 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 | overcloud-novacompute-prodd1-0  |
| 2ec447e7-002c-4201-87c0-16d1fde9274c | overcloud-novacompute-prodd1-1  |
| 593de817-7687-4fcd-8a46-9f8a035e911c | overcloud-novacompute-prodd1-2  |
| 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 | overcloud-controller-prodd1-0   |
| feb6b514-69b3-4c40-aed8-c79e9943e887 | overcloud-controller-prodd1-1   |
| f0b95444-6055-48d4-814a-fa90a033c693 | overcloud-controller-prodd1-2   |
+--------------------------------------+---------------------------------+
```

There is no mechanism to track which physical node is failed if there is hardware failure in compute or controller node because these are the names assigned by Openstack deployment. On the physical devices different naming convention is allocated as shown in following figure. It is only showing Blades in HP C7000 chassis onboard administrator GUI.

Figure 17 - Blade Names in HP C7000 Chassis

ID shown in above output has entry in ironic database from where these nodes are PXE boot.

```
$ ironic node-list
+--------------------------------------+-------------+--------------------------------------+
| UUID                                 | Name        | Instance UUID                        |
+--------------------------------------+-------------+--------------------------------------+
| 5384f95e-725c-4039-9253-62f9dfe06e74 | ieatosk5613 | 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 |
| ce67624a-6102-4c13-b297-c56ba4b1f6a5 | ieatosk6044 | 2ec447e7-002c-4201-87c0-16d1fde9274c |
| df2e76e4-a080-4d9b-820a-7ba6055de776 | ieatosk6642 | 593de817-7687-4fcd-8a46-9f8a035e911c |
| 5384f95e-725c-4039-9253-62f9dfe06e74 | ieatosk5613 | 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 |
| e8fb9aaf-6272-40d0-961c-1ad7fc710461 | ieatosk6321 | feb6b514-69b3-4c40-aed8-c79e9943e887 |
| 92287914-b737-4250-8c56-abb030cbe43a | ieatosk6649 | f0b95444-6055-48d4-814a-fa90a033c693 |
+--------------------------------------+-------------+--------------------------------------+
```

Above output has been omitted to show only relevant information. Ironic database is built from instackenv.json file that is discussed in *section 5.3.2*. To populate Name field in above output we add ***name*** value in instackenv.json file. But this information is not provided in standard installation document. So if the ironic database has been built then we can update Name field with below command.

```
#ironic node-update <UUID of the node> add name="ieatosk5613"
```

In future deployments name field added against each node in instackenv.json file

```
{
                "mac":[
                        "EC:B1:D7:A9:1F:B0"
```

```
            ],
            "cpu":"40",
            "memory":"262144",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"cloud",
            "name":"ieatosk5613",
            "pm_password":"cl0udroot",
            "pm_addr":"10.151.40.114"
        },
```

Once Ironic database is updated then we can track failed node by co-relating *Instance UUID* in ironic database with *ID* value in undercloud's Nova database. Following diagram explains the relation of both databases side by side

```
$ ironic node-list
+--------------------------------------+-------------+--------------------------------------+
| UUID                                 | Name        | Instance UUID                        |
+--------------------------------------+-------------+--------------------------------------+
| 5384f95e-725c-4039-9253-62f9dfe06e74 | ieatosk5613 | 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 |
| ce67624a-6102-4c13-b297-c56ba4b1f6a5 | ieatosk6044 | 2ec447e7-002c-4201-87c0-16d1fde9274c |
| df2e76e4-a080-4d9b-820a-7ba6055de776 | ieatosk6642 | 593de817-7687-4fcd-8a46-9f8a035e911c |
| 5384f95e-725c-4039-9253-62f9dfe06e74 | ieatosk5613 | 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 |
| e8fb9aaf-6272-40d0-961c-1ad7fc710461 | ieatosk6321 | feb6b514-69b3-4c40-aed8-c79e9943e887 |
| 92287914-b737-4250-8c56-abb030cbe43a | ieatosk6649 | f0b95444-6055-48d4-814a-fa90a033c693 |
+--------------------------------------+-------------+--------------------------------------+

$ openstack server list -c ID -c Name
+--------------------------------------+--------------------------------+
| ID                                   | Name                           |
+--------------------------------------+--------------------------------+
| 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 | overcloud-novacompute-prodd1-0 |
| 2ec447e7-002c-4201-87c0-16d1fde9274c | overcloud-novacompute-prodd1-1 |
| 593de817-7687-4fcd-8a46-9f8a035e911c | overcloud-novacompute-prodd1-2 |
| 7270c8ba-249f-4f22-89f5-f841bfe5c2e5 | overcloud-controller-prodd1-0  |
| feb6b514-69b3-4c40-aed8-c79e9943e887 | overcloud-controller-prodd1-1  |
| f0b95444-6055-48d4-814a-fa90a033c693 | overcloud-controller-prodd1-2  |
+--------------------------------------+--------------------------------+
```

Figure 18 - Ironic and Nova dbs relationship for instance ID

6.5 Misconfigured Controller behaving Compute - Operation

Nova consists of following four services

- Nova-Scheduler
- Nova-Consoleauth
- Nova-Conductor
- Nova-Compute

Nova compute service runs on compute nodes and remaining service run on controller node as show in following output

```
$ nova service-list

+-----+------------------+-----------------------------------------+----------+---------+-
------+----------------------------+-----------------+

| Id  | Binary           | Host                                    | Zone     | Status  |
State | Updated_at                 | Disabled Reason |

+-----+------------------+-----------------------------------------+----------+---------+-
------+----------------------------+-----------------+

| 2   | nova-scheduler   | overcloud-controller-prodd1-0.localdomain   | internal | enabled |
up    | 2017-04-23T14:27:55.000000 | -               |

| 11  | nova-consoleauth | overcloud-controller-prodd1-0.localdomain   | internal | enabled |
up    | 2017-04-23T14:27:55.000000 | -               |

| 170 | nova-conductor   | overcloud-controller-prodd1-2.localdomain   | internal | enabled |
up    | 2017-04-23T14:27:57.000000 | -               |

| 284 | nova-compute     | overcloud-novacompute-prodd1-3.localdomain  | nova     | enabled |
up    | 2017-04-23T14:27:59.000000 | -               |
```

If ***openstack-nova-compute*** is started mistakenly on controller node, then ***nova-service-list*** automatically picks in service polling and listing. In this case nova-scheduler includes that controller node in its hypervisors list and try to create instance on controller node. Instance might create but it will not be accessible. In order to avoid this situation, we have to stop nova-compute service from controller node and disable it so that it will not start automatically in future upon controller restart.

Nova Service is showing below output if compute service is running on controller node. Complete output is not being shown here.

```
| 71 | nova-compute     | overcloud-compute-8.localdomain    | nova     | enabled |
up    | 2017-01-03T17:28:17.000000 | -               |

| 74 | nova-compute     | overcloud-compute-9.localdomain    | nova     | enabled |
up    | 2017-01-03T17:28:18.000000 | -               |

| 77 | nova-compute     | overcloud-controller-0.localdomain | nova     | enabled |
up    | 2017-01-03T17:28:17.000000 | -               |
```

```
+----+------------------+-------------------------------------+----------+---------
```

Below steps are performed on controller node.

```
$ sudo systemctl disable openstack-nova-compute
```

```
$ sudo systemctl stop openstack-nova-compute
```

From director machine, source overcloudrc and delete the nova-compute service that is bind to controller node.

```
$ nova service-delete <ID of the service>
```

ID is retrieved from ***nova service-list*** command as shown in above output

6.6 Controller services restart upon compute Nodes Expansion

Overcloud deployment has been explained in *section 5.4*. Same steps are needed if we need to expand out cloud environment with compute or controller nodes. There are two options shown compute-scale and control-scale in deployment command. If our current compute-scale is 21 and we want to add three further compute nodes in compute capacity, then in deployment script we need to mention compute-scale as 24 not 3. Otherwise cloud environment will reduce to 3 compute nodes. This expansion process is service impacting and service are restarted on controller nodes during expansion. Redhat provided a workaround to avoid services restart but it overrides manual changes which are done on nodes after initial deployment (Redhat, 2017). So changes had to be manually restored after completion of scale out operation.

6.7 Network Nodes not synchronizing with ENM - Operation

NMS application uses LVS (Linux Virtual Server) for the communication of KVM that perform different functions like fault, performance, or configuration management for different network nodes. LVS performs NAT to connect KVMs with network elements/nodes. NMS application uses mediation nodes that perform different mediation functions like fault mediation. In the test environment network simulators are used that behave like real world network nodes. Application that holds these network simulator nodes is called netsim and it is in-house built software. It is reported by development teams that network nodes are not synchronizing with mediation nodes (Mediation KVMs)

Here is the flow of the packet from mediation node towards netsim node in physical environment.

- Packet leave mediation node
- Packet is received at internal address of LVS router

- LVS NATs the source address to its external address and forwards
- Packet arrives at netsim node and it replies back
- Packet arrives at LVS external interface and LVS forward it to internal interface
- Packet reply is sent back to mediation nodes

It is observed that packet is not reaching back to mediation nodes. Here is the diagrammatical depiction of the issue

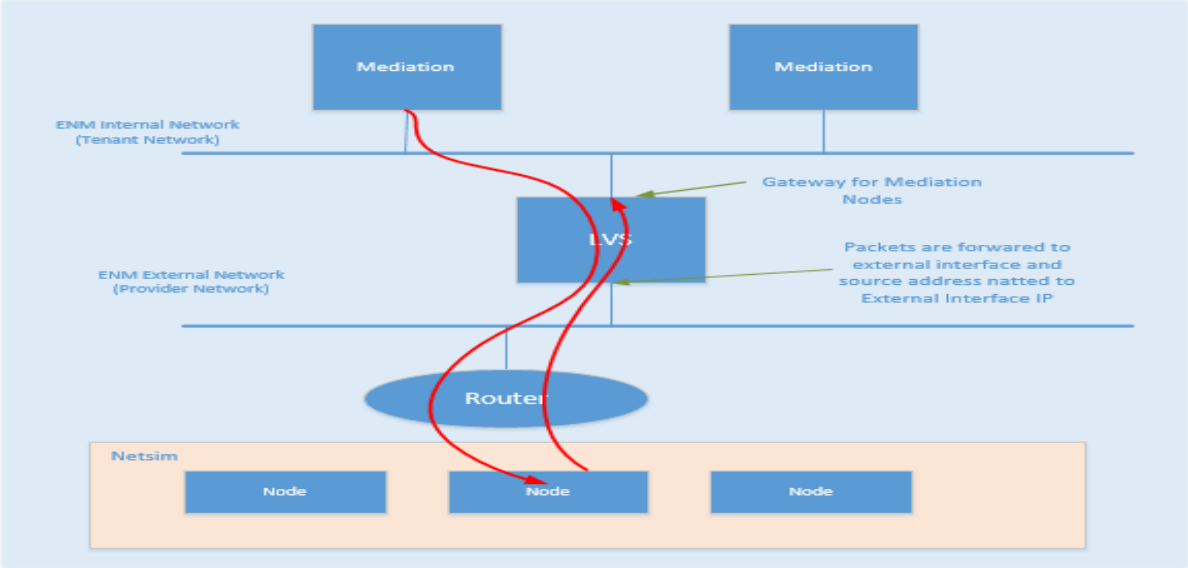

Figure 19 - Packet Flow from Mediation nodes towards Netsim Nodes

Neutron drops all ingress traffic on an instance's network interface that is connected to a tenant network and is not connected to a neutron's router or if that traffic is not from an address configured on the interface.

Since the LVS router is not a neutron defined router, neutron will drop all ingress traffic on the interface if the traffic source address is not defined on that interface. So product development team added one rule to allow all traffic on that interface. By Specifying 0.0.0.0/0 allows all traffic through.

6.8 VMs status upon Compute Node Failure

Currently there is no high availability mechanism in compute nodes. If compute nodes fail due to any hardware issue. There is no way to recover VMs running on that compute nodes. One option is to define orchestration on the basis of ceilometer logs that if compute node appears to be in trouble then migrate the VMs from that compute node to another compute Node.

6.9 Floating IP Networks Vs Provider Network – External Access

Instances in OpenStack get IP from tenant network. Tenant network is isolated from physical or external network. That is why same tenant network can be used in multiple projects. Floating IPs are used in OpenStack environment for instances to communicate with outside world. They are SNAT/DNAT (Source NAT and Destination NAT) that is created in iptables of qrouter network namespace. Floating IPs are assigned in round-robin fashion from floating IPs pool to the instance that need access to external provider network. There is no way through CLI to bind particular floating IP to particular instance. In each cloud POD there is a pool of floating IP. Every tenancy or project is assigned certain number of floating IPs out of that pool. Development teams are having issues that are unable to bind floating IP to certain instance as they have defined in their SED (Site Engineering Document). It is also observed that heat stack deletion commands fail to remove floating IP and subsequently fails to delete stack.

So It is requested to create provider networks that can be assigned to instances. So provider networks are created and instances have direct connectivity to external networks. In this case there is no NAT process from private to floating IPs. Also not all instances are allocated provider IPs. Some are still using tenant network only.

6.10 Operations Management tool(s)

Redhat OpenStack provides collection of open source tools alarms for monitoring, centralized logging and performance data collection of OpenStack nodes. For alarms monitoring *Sensu agent* is installed on all the controller and compute nodes. Agent collects alarms from agents and forwards to *sensu server*. Sensu server can be installed on physical hardware or it can be a VM. Sensu server forwards these alarms to presentation layer application called *Uchiwa*.

For Centralized logging *Fluentd* is used as log collect agent and log transformer. *Elasticsearch* is used as data store for log data and *Kibana* is the presentation layer for centralized logs.

Performance monitoring consist of *collectd* as data collection agent. Graphite is collection aggregator. It aggregates the data and then stores in *whisphedb*. *Grafana* is used as presentation layer.

Redhat document “Red Hat OpenStack Platform Operational tools” describes all these tools and the installation procedure. (Redhat OpenStack Team, 2016) .

These tools are very generic in nature and can be used with any open source applications. Documentation is only limited to installation and does not provide details about effective utilization

of these tools. At the time of writing this thesis report these tools are under evaluation to see if they provide good operational management of OpenStack cloud.

7. VMware Challenges

7.1 Nodes stop responding during vMotion

Before explaining the issue, it is important to understand the vMotion process. There is very good explanation on VMware blog (Gleed, 2011)

1. Shadow VM created on the destination host.
2. Copy each memory page from the source to the destination via the vMotion network. This is known as precopy.
3. Perform another pass over the VM's memory, copying any pages that changed during the last preCopy iteration.
4. Continue this iterative memory copying until no changed pages (outstanding to be-copied pages) remain.
5. Stun the VM on the source and resume it on the destination.

As explained in NMS product introduction that it uses KVMs on physical blade servers. These servers are named services (SVC), database (DB) and scripting (SCP) servers in the solution. Veritas cluster services are used for high availability in application. Same design approach is used in VMware based cloud and VMware's nested virtualization is adopted so the KVMs are created inside the VMware virtual machine. This implementation is not supported by both VMware and Redhat, but it is usable for developers to test their applications. Below figure shows the nested virtualization architecture on single ESXi host (Blade server in our case). This diagram is only showing Nested virtualization. But actual does not necessary consist of 4 VMware VMs and 4 KVMs per VM. It can have varying number of VMs.

Figure 20 - Nested Virtualization Overview

VMware based cloud NMS application uses small subset of the features of the application deployed on physical hardware. Standard template used by developers uses two SVC, a DB node as VMs and then KVM runs on these VMs. As part of Distributed Resource Scheduling, when any of SVC nodes move from one ESXi host to other ESXi host in the cluster (vMotion). It is observed during the stun process, KVMs are not aware of it and stop responding for some time and CPU lockups are observed. On SVC nodes it throws messages related to Veritas cluster that node is in trouble. Below error is seen in */var/log/messages* in most of KVMs

```
Jan 23 17:38:25 svc-1-mspm kernel: BUG: soft lockup - CPU#1 stuck for 145s!
[java:5733]
```

As mentioned earlier this is unsupported implementation, after discussion with VMware support it is decided that we set that DRS setting in vCenter from fully automated to partially automated. In case of partially automated DRS setting, VM is placed on best host upon power on. After that VMware DRS will only suggest migration recommendations but it will not migrate VM.

7.2 SCSI Enquiries not working in Linux Guests

The issue is caused by the fact that physical deployments are using the /dev/disk/by-id file to locate the disk id. The file /dev/disk/by-id is used to store an id of the disk based on the hardware serial number. For SAN storage it is a WWN that we get as an identifier for a LUN and this number matches the entry in /dev/disk/by-id. This is not the case in virtualized storage scenarios. The way that devices show up under */dev/disk/by-id* is when the *scsi_id* command queries the devices for VPD (SCSI inquiry Vital Product Data) and when the disk returns the information it can create the appropriate

link. SCSI inquiry VPD pages are not supported by default (VMware, n.d.) . There are two ways of checking for that data in RHEL VM.

```
# scsi_id -p 0x83 -g /dev/sda
# scsi_id -p 0x80 -g /dev/sda
```

0x83 is Device Identification Vital Product Data and **0x80** is Unit Serial Number (Elatov, 2012). Both commands are showing empty results in case of virtual machines.

After investigation it is found that there is a parameter *disk.enableUUID* in vSphere which can be set and which will ensure that there is a UUID assigned to the disk when it is allocated to VM. So following steps are performed to add entry in virtual machine configuration.

To enable disk UUID on a virtual machine

1. Power off the guest.
2. Select the guest and select **Edit Settings**.
3. Select the **Options** tab on top.
4. Select **General** under the **Advanced** section.
5. Select the **Configuration Parameters...** on right hand side.
6. Check to see if the parameter **disk.EnableUUID** is set, if it is there then make sure it is set to **TRUE**.

   If the parameter is not there, select **Add Row** and add it.
7. Power on the guest.

Now if you run the SCSI enquiry commands we can see the output.

```
# scsi_id -p 0x83  -g /dev/sda
36000c2994cb8ebf3f6b350c215b21f75
# scsi_id -p 0x80 -g /dev/sda
SVMware  Virtual disk   6000c2994cb8ebf3f6b350c215b21f75
```

7.3 Stranded Items in vCloud Director after Jenkins Jobs

As mentioned earlier in *section 4.5*, Jenkins jobs are used to install the application. There are Jenkins plugins that interact with VMware environment. Development teams run Jenkins jobs that deploy vApp in bulk. Once the jobs are successfully executed all the vApps that are created as part of job are deleted from vCloud. Sometimes it creates huge load on vCloud director if cloud director is busy in some other tasks also. As vCloud is the management layer on top of vSphere environment that directs

instruction to vSphere to perform all operations. Sometimes due to load on vCloud or vSphere side all the items does not delete. vCloud director tries to delete it again. If it fails, then these items are moved to Stranded items list in vCloud director. We can try to delete from Stranded items list in vCloud GUI. If it still fails to delete, then we have delete from login to vCenter and then locate it and delete it. This is permanent issue and manual intervention is required to clear all these items.

7.4 VXLAN offloading on ESXi host NICs

Virtual Extensible LAN (VXLAN) is encapsulation protocol for running an overlay network on existing Layer 3 infrastructure. An overlay network is a virtual network that is built on top of existing network Layer 2 and Layer 3 technologies to support elastic compute architectures. VXLAN makes it easier to scale out a cloud computing environment while logically isolating cloud apps and tenants.

We can create around 16 million VLANs through VXLAN. VXLAN is used for tenant networking. Our cloud solution is based on HPE C7000 chassis and BL460 Gen8 and Gen9 blades. ESXi is installed on blades. VXLAN has some overhead on compute hosts. HPE released a new NIC driver that offloads CPU overhead from ESXi and moves it to NIC. But we observed that NIC started to go offline after some time. Sometime single NIC go offline and sometime both go offline and host gets disconnected from vCenter also. VXLAN offloading functionality is available in previous version of the NIC but it is disabled by default unless it is manually enabled. But in newer version it is enabled as default. After analysis it is found that VXLAN process is causing issues. There is no issue with HP BL460c Gen9 blades having bnx2x NIC driver version *2.710.39.v55.2* or earlier. In these NIC drivers vxlan offloading is disabled.

Here is the esxcli command and its output in NIC version *2.710.39.v55.2*

```
# esxcli system module parameters list --module bnx2x | grep vxlan
enable_vxlan_ofld                    int                 Allow vxlan TSO/CSO
offload support.[Default is disabled, 1: enable vxlan offload, 0: disable vxlan
offload]
```

BNX2X NIC driver version *2.710.70.v55.7* or higher is problematic and you will get below output showing it is enabled by default.

```
# esxcli system module parameters list --module bnx2x | grep vxlan
disable_vxlan_filter                 int                 Enable/disable vxlan
filtering feature. Default:1, Enable:0, Disable:1


enable_vxlan_ofld                    int           0     Allow vxlan TSO/CSO
offload support.[Default is enabled, 1: enable vxlan offload, 0: disable vxlan
offload]
```

So disabling vxlan offloading from the NIC and rebooting host solved the issue. There is not fix from HPE or qlogic yet on this known issue.

These are the steps to disable vxlan offloading from NIC. Login to ESXi host and execute below command.

```
# esxcfg-module -s enable_vxlan_ofld=0 bnx2x
# esxcfg-module -g bnx2x
```

7.5 VMware tools for graceful shutdown of VM – Operation

As mentioned in section 4.4 that application installation is automated through Jenkins and VMware tools are also installed as part of the job. Sometimes installation fails through Jenkins and developers have to troubleshoot it. As part of troubleshooting they resume failed installation. Or sometimes some developers manually install the application to test some functionality. So VMs in these vApp are missing VMware tools. Out of many features of VMware tools, one is to gracefully shutdown or reboot the guest OS if user try to do it from vCenter tool box. As It is shown in figure-7 and Table-1 in section 4.1 that Self provisioning portal is provided to developers to deploy their vApps. This SPP has tools box that provides some of vApps actions by calling VMware APIs. It is shown in following figure. Some of the content in the figure are hidden for privacy purposes. Now if VMware tools are not installed and user try to stop the VM from the tool box in SPP. This action is mapped to *Stop Action* that is defined in vCloud director against that VM. It can be Shut Down or Power Off as shown in figure-19. Due to missing VMware tools vApp fails to stop. In order to get rid of this issue we have to install VMware tools in the VM so that they can be properly shut down.

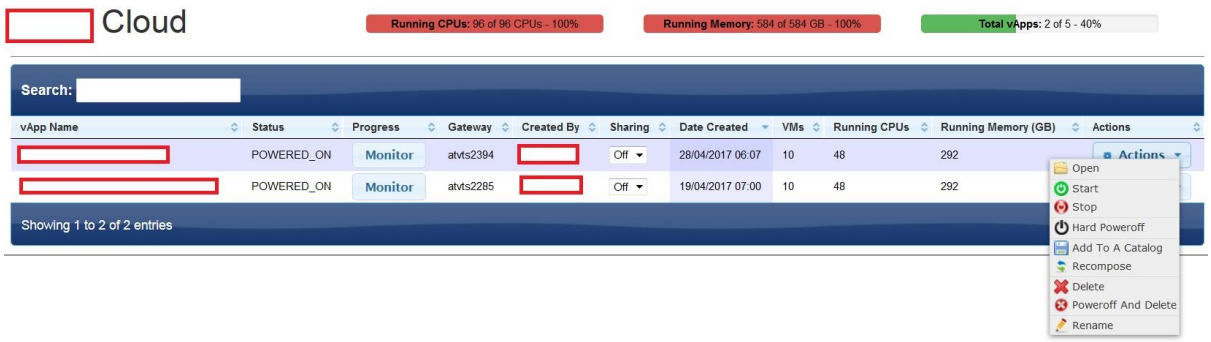

Figure 21 - Self Provisioning Portal

Here is the VMs Stop order in vCloud Director GUI.

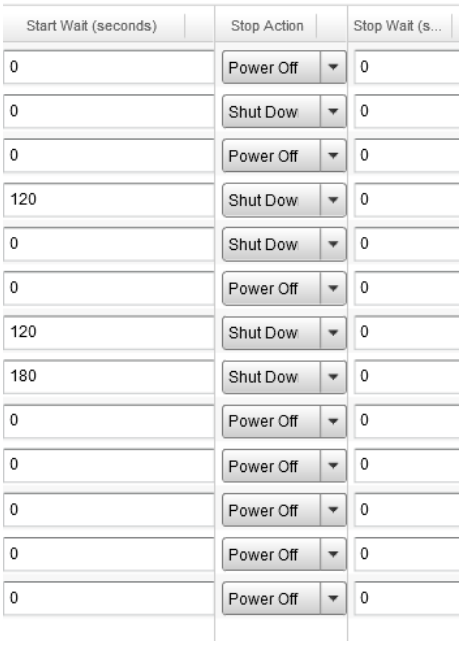

Figure 22 - VM Stop Action in vCloud Director

7.6 Windows based Management VMs Patching – Operation

Some applications like vCenter, Update Manager and vCloud database in vCloud solution are installed on Windows VMs. It is necessary to keep those machines update with Windows recommended patching. Extra care is need when choosing patches for Windows VM where have SQL database running. VMware has strict support on database level. If database is not as per their recommendation no support will be provided. It happened that automatic download and install of patches is chosen. Windows wait for some time after installation of patches to reboot machine. If machine is not rebooted, it will automatically reboot the machine. That is unplanned service interruption. Below diagram shows recommended setting if we are choosing Windows patching. So we need to choose that patches are download but we only install when we have maintenance window. Similarly, we need to uncheck the box so that Windows will not download the patches for SQL database.

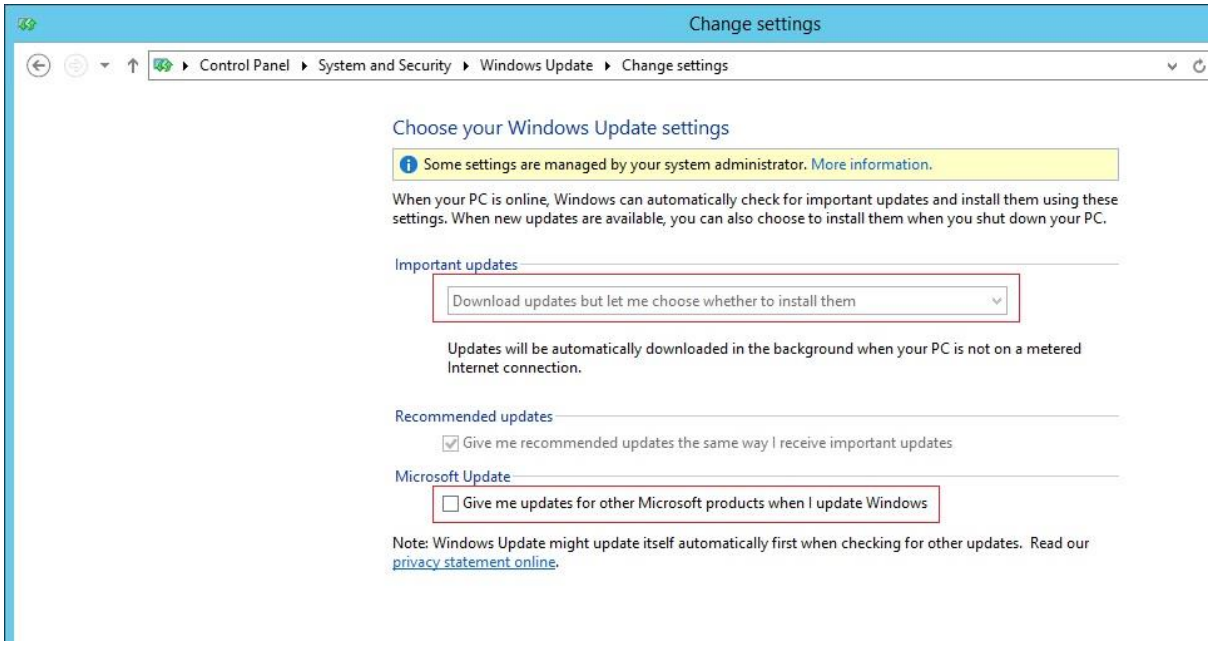

Figure 23 - Windows patching recommendations

7.7 Time synchronization in VMware tools for Linux Guests

It is observed that if VMware tools are installed on Linux guests along with NTP, after Daylight saving time (DST) and time is moved back/forward one hour. If the guest machines are powered on that were powered off before the DST or deployed from template, it will throw below error on console and boot process will hang

```
vmsvc[1174]: [ warning] [guestinfo] RecordRoutingInfo: Unable to collect
IPv4 routing table.
```

After analysis it is found that issue occurs when the Linux *iputilis* package causes a delay in the boot process. A warning message appears when the guestinfo plugin tool fails to parse the content from the /proc/net/route file. The guest operating system's clock is ahead of or behind the host on which it is running, causing the arping process to become unresponsive during boot. It is related to time synchronization library from vmware-tools. Time is resynchronized when you migrate the virtual machine using vMotion, take a snapshot, restore to a snapshot, shrink the virtual disk, or restart the VMware Tools service in the virtual machine (including rebooting the virtual machine) even time synchronization is disabled on the guest machine.

```
# vmware-toolbox-cmd timesync status
Disabled
```

VMware also suggest to use NTP rather than using vmware-tools time synchronization (VMware KB 1189, 2014).

In order to completely disable time synchronization through vmware-tools we need to perform below steps.

These steps can be performed either through vSphere desktop client or web client.

1. Select the virtual machine in the vSphere Client inventory and power it off.
2. On the **Summary** tab, click **Edit Settings**.
3. Click the **Options** tab and click **General** (under **Advanced**).
4. Click **Configuration Parameters**, then click **Add Row** and add this information:

Name	Value
tools.syncTime	0
time.synchronize.continue	0
time.synchronize.restore	0
time.synchronize.resume.disk	0
time.synchronize.shrink	0
time.synchronize.tools.startup	0
time.synchronize.tools.enable	0
time.synchronize.resume.host	0

Table 3 - Time Synchronization Parameters (Source VMware KB#1189)

8. Conclusion and Future Work

Cloud computing has seen significant growth in recent years and cloud is being used as part of IT strategy of most of the companies. Customers have demands from vendors to provide them cloud-native applications. They do not want applications having special hardware requirements. Similarly, vendors want to develop applications that can run in any cloud platform. For building feature-rich and readily available to market the software, development teams are adopting continuous Integration framework and Agile development where they want to have development environment that can be self-provisioned and readily available rather than waiting for days or weeks for physical environment setup.

In this thesis we explored the issues that are faced during the deployment, operations and expansion of VMware based and Redhat OpenStack cloud. These cloud environments are used for deploying

Network Management System application that is currently deployed in bare metal hardware. Normally significant effort is required from physical hardware deployment till application installation and it takes weeks to make available working system. We see couple of issues faced during the Redhat based OpenStack cloud deployment. Some of them are discussed in OpenStack challenges section of the report. Redhat OpenStack is based on TripleO project that is called “OpenStack on OpenStack”. Undercloud and overcloud are two cloud environments in TripleO project. Undercloud is used to build actual working cloud i.e. overcloud. If we have deployed director or undercloud server then there is server boot issue after undercloud software installation. Server boot hangs if it has logical volumes other than root and swap and disk size is larger than 50GB. Official installation document has no such information to avoid this issue. Similarly overcloud nodes like compute and controller did not PXE boot although correct power management type was chosen in configuration file as per documentation until standard power management (pm_type) type is applied in configuration. We see some issue due to customization in our environment like two cloud environment were sharing one VNX storage. After assigning unique iSCSI initiator names to controller nodes in each deployment we could deploy cinder volumes in storage.

When it comes VMware based cloud deployment, we don’t see any major issue if we are following strictly documentation and hardware used is in VMware’s hardware compatibility list. It is worth mentioning that we are using stable release of VMware vCloud suite and all the facts and figures are based on that release. At the time of writing this thesis installation on newer version was under development.

We found that compute nodes expansion is quick in Openstack cloud and in couple of minutes we have compute nodes available to be used. But controller nodes services restart during the expansion. During expansion we must mention the correct compute count by adding current number of expansion nodes count in existing nodes count. In VMware based cloud adding ESXi host to compute cluster in vCenter has couple of steps before it is available for use. It involves installing ESXi host, then patching and storage connectivity is needed before it is available for use.

For application deployment different approaches are used in both cloud environments. In OpenStack application is being redesigned to make it cloud-native as currently it has business case. In Physical deployment, KVMs are running on blade servers. Similar approach is used in VMware based cloud and nested virtualization is used for application deployment with slight modifications in DRS rule. Although

nested virtualization is unsupported but it provided good environment for developers for functional and high availability testing.

When it comes to operations management in OpenStack cloud, one of the issue is hardware failure in compute nodes. Redhat Controller nodes run in high availability mode. Failure of any controller node has no impact on running deployment. But if compute node fails, there is no mechanism of restart of instances running on that host to another compute host as we have in VMware HA compute or resource clusters. We do not have any further findings on issues related to compute nodes failure as we did not face any compute node failure while writing this thesis.

For majority of day to day operational issues VMware has good support setup in the form of customer services and freely available Knowledge base(s). OpenStack has huge community of open-source developers plus support from vendors if you have using OpenStack cloud from vendors like Redhat and Mirantis. But it is observed that troubleshooting OpenStack issues requires more skill and attention as compared to VMware based cloud.

Going forward we are planning to explore OpenStack cloud offering from Ericsson and other vendors like Mirantis and HP. Ericsson has built a telco grade cloud Ericsson Cloud Execution Environment (ECEE). It is built on top of Mirantis OpenStack with further enhancements from Ericsson. We are also building VMware Integrated OpenStack (VIO) cloud as it is another interesting option. VIO has preconfigured Openstack code that uses VMware OpenStack drivers and tools to install OpenStack cloud on top of VMware technologies. We have plans to explore VMware's vRealize Operation Management's management Pack for OpenStack that provides performance and availability monitoring for OpenStack infrastructure and services.

With all the above discussion taken into account, this thesis can act an important guide for cloud engineers to design and build an OpenStack or VMware based cloud by following design guidelines and issues discussed that appeared during deployment, expansion, and operations of both cloud solutions.

9. Bibliography

Al-Najjar, A., Pakzad, F. & Layeghy, S., 2016. *Link capacity estimation in SDN-based end-hosts.* s.l., 2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1-8.

Awasthi, A. & Gupta, R., 2016. *Multiple hypervisor based Open Stack cloud and.* Gurgaon, Haryana, India, 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), pp. 130-134.

Behrendt, et al., 2011. *Introduction and architecture overview ibm cloud computing reference architecture 2.0.* s.l., Draft Version V.

Campbell, T., 2016. *Troubleshooting OpenStack.* s.l.:Packt Publishing.

Chen, S.-C. & Hwang, R.-H., 2016. *A Scalable Integrated SDN and OpenStack Management System.* s.l., IEEE, pp. 532-537.

Chirammal, H. D., Mukhedkar, P. & Vettathu, A., 2016. *Mastering KVM Virtualization.* s.l.:Packt Publishing.

Dafni, R., 2015. *http://www.stratoscale.com/blog/industry-buzz/why-comparing-openstack-to-vmware-vcloud-is-like-comparing-android-to-apple-ios/.* [Online]
[Accessed 15 11 2016].

Dorney, A., 2015. *Australian Businesses Will Soon Benefit from Dedicated vCloud.* [Online]
Available at: https://blog.rackspace.com/australian-businesses-will-soon-benefit-from-dedicated-vcloud
[Accessed 1 3 2017].

Elatov, K., 2012. *Missing links to devices on Linux VMs under /dev/disk/by-id/.* [Online]
Available at: http://virtuallyhyper.com/2012/03/missing-links-to-devices-on-linux-vms-under-devdiskby-id/
[Accessed 3 11 2016].

Ericsson, 2017. *Ericsson.* [Online]
Available at: https://www.ericsson.com/about-us/company-facts
[Accessed 10 January 2017].

Gleed, K., 2011. *VMware vSphere Blog.* [Online]
Available at: https://blogs.vmware.com/vsphere/2011/02/vmotion-whats-going-on-under-the-covers.html
[Accessed 23 3 2017].

Grossman & Robert , L., 2009. The case for cloud computing. *IT professional,* Volume 11, pp. 23--27.

Hui, K., 2013. *OpenStack Nova-Scheduler and vSphere DRS.* [Online]
Available at: https://blog.rackspace.com/openstack-nova-scheduler-and-vsphere-drs
[Accessed 30 2 2017].

Jain, P., Datt, A., Goel, A. & Gupta, S. C., 2016. *Cloud Service Orchestration based Architecture of.* Jaipur, India, 2016 Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2453-2459.

Lifflancer, et al., 2009. *Clustering Versus Shared Nothing: A Case Study.* s.l., Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International, pp. 116--121.

Lowe, S. & Marshall, N., 2014. *Mastering VMware vSphere 5.5.* s.l.:Sybex.

Lowe, S. & Marshall, N., 2014. *Mastering VMware vSphere 5.5.* s.l.:Sybex.

Mell, P. & Grance, T., 2011. *NIST: SP 800-145.* [Online]
Available at: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
[Accessed 15 10 2016].

Mullerikkal, Paul , J., Sastri & Yedhu, 2015. *A Comparative Study of OpenStack and CloudStack.* s.l., Advances in Computing and Communications (ICACC), 2015 Fifth International Conference, pp. 81--84.

OpenStack, 2015. *OpenStack Open Source Cloud Computing Software.* [Online]
Available at: http://www.openstack.org/software/
[Accessed 22 2 2017].

OpenStack, 2016. *OpenStack Admin Guide.* [Online]
Available at: https://docs.openstack.org/admin-guide/
[Accessed 7 11 2016].

OpenStack, 2017. *Keystone, the OpenStack Identity Service.* [Online]
Available at: https://docs.openstack.org/developer/openstack-projects.html
[Accessed 19 3 2017].

OpenStack, T. D., 2017. *https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html/director_installation_and_usage/chap-introduction.* [Online]
Available at: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html/director_installation_and_usage/chap-introduction
[Accessed 10 03 2017].

Padala, P., 2013. *Resource Management in VMware Powered Cloud: Concepts and Techniques.* s.l., 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, pp. 581-581.

Parekh, P., 2015. *Dedicated VMware vCloud: A Managed Private Cloud Powered by VMware Technology and Rackspace Fanatical Support.* [Online]

Available at: https://blog.rackspace.com/dedicated-vmware-vcloud-a-managed-private-cloud-powered-by-vmware-technology-and-rackspace-fanatical-support
[Accessed 3 3 2017].

Peeler, J., 2016. *Red Hat Bugzilla.* [Online]
Available at: https://bugzilla.redhat.com/show_bug.cgi?id=1323024
[Accessed 25 11 2016].

Redhat, 2016. *Redhat OpenStack Platform.* [Online]
Available at: https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/9/html/product_guide/ch-understanding-rhosp
[Accessed 7 2 2017].

Redhat, 2017. *Redhat Knowledge Base.* [Online]
Available at: https://access.redhat.com/solutions/2345231
[Accessed 18 1 2017].

Rodriguez, A., 2008. *Restful web services: The basics.* s.l., IBM developerWorks, p. 1–11.

Rosado, Tiago , Bernardino & Jorge, 2014. *An overview of openstack architecture.* s.l., ,Proceedings of the 18th International Database Engineering \& Applications Symposium, ACM, pp. 366--367.

Sahasrabudhe, S. S. & Sonawani, S. S., 2014. *Comparing Openstack and VMware.* Pune, India, 2014 International Conference on Advances in Electronics Computers and Communications, pp. 1-4.

Sefraoui, et al., 2012. OpenStack: toward an open-source solution for cloud computing. *International Journal of Computer Applications Foundation of Computer Science,* Volume 55.

Shrivastwa, A. & Sarat, S., 2015. *Learning OpenStack.* s.l.:Packt Publishing.

Tesfamicael, A. D., Liu, V. & Caelli, W., 2015. *Performance Analysis of Secure Unified.* risbane, Australia, 2015 International Conference on Computational Intelligence and Communication Networks (CICN), pp. 1135-1140.

TripleO, 2017. *TripleO.* [Online]
Available at: http://tripleo.org/introduction/introduction.html
[Accessed 7 3 2017].

VMware.com, 2015. [Online]
Available at: http://www.vmware.com
[Accessed 20 02 2017].

VMware, 2014. *VMware vCloud Director: Install, Configure, Manage [V5.5] STUDENT eKit,* s.l.: VMware/Gilmore. VitalBook file..

VMware, n.d. *VMware KB.* [Online]
Available at:

https://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1029157

Wei, X., 2012. *Based on VMware Technology's Campus Network Cloud Platform Technology Research.* s.l., 2012 International Conference on Computer Science and Electronics Engineering, pp. 430-433.

Xie, L., 2013. *OpenStack Unlocked.* [Online]
Available at: https://www.mirantis.com/blog/cloud-prizefight-vmware-vs-openstack/
[Accessed 30 2 2017].

Yang, et al., 2013. *Implementation of a cloud iaas with dynamic resource allocation method using openstack.* s.l., 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 71--78.