Design and Implementation of Private Cloud Infrastructure based on OpenStack for Strategic Organizations

M. Swapna*[1], Devanahalli Sunil Archana
Research Centre Imarat (RCI), Hyderabad- 500 069, India
E-mail: swapna@rcilab.in, sunil.archana@rcilab.in*

ABSTRACT

Cloud technologies have become indispensable part of organizations today. As an evolving paradigm for anytime, anywhere, cost effective computing, cloud computing has attracted huge attention among academics and industry. Many commercial companies are providing public cloud solutions through internet. In public clouds, control over the entire cloud infrastructure and the data hosted on it remains with the company providing cloud solution. However, government and defense organizations require that control over the cloud infrastructure and the hosted data is retained with them. Hence private clouds are more suitable for strategic organizations.

This paper discusses design choices and challenges involved in implementing an Open Stack based private cloud system for an organization and customizing it for developing strategic applications while saving recurrent costs and also providing data security. It also focuses on automating the entire deployment with minimal manual work. As a case study for PaaS, deep learning frameworks have been provided on cloud for training deep learning models.

Keywords*: Open Source clouds**, OpenStack, Private Cloud, Design of private Cloud*

1. INTRODUCTION

Cloud Computing provides a shared pool of computing resources (storage, network, memory and CPU) to users using cloud enabling technology called virtualization. This gives users a perception that they have access to unlimited computing resources.

The National Institute of Standards and Technology (NIST) [1] definition of cloud computing describes it as a “model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. This model is composed of three service models namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). It elaborates on IaaS as "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications “. The control sphere of consumer here is over operating systems, storage and applications deployed. Management of cloud infrastructure is out of customer control sphere. In PaaS model, consumer is provided with programming framework, such as programming language, libraries, tools to build and develop applications. Consumer control sphere is within deployed applications and over configuration settings of development environment. Management of underlying infrastructure and operating system is out of consumer control sphere. In SaaS model, consumer is provided access to applications running on cloud infrastructure. The control sphere of consumer is over application configuration settings. Management of underlying infrastructure, operating system or development environment is out of consumer control sphere.

The IaaS cloud architecture consists of different layers as shown in Figure 1. The resource pool consists of all physical hardware. Hypervisor runs natively on physical hardware and

[1]Corresponding author: E-mail: swapna@rcilab.in

enables sharing of resources of physical hardware among of multiple virtual machines hosted on physical hardware. IaaS software runs on hypervisor and controls pool of storage, compute and network resources. Most of the open source cloud solutions support IaaS service model.

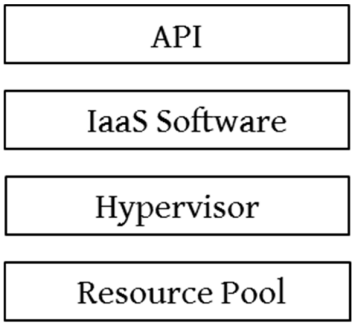

Figure 1: Different layers of OpenStack IaaS Cloud [2]

The paper is structured as follows. Section 2 points to literature review of existing open source cloud solutions. Section 3 formulates the problem definition and section 4 explains the setup of private cloud computing system and its deployment plan. Section 5 presents discussion on important design aspects. Section 6 presents challenges faced during the implementation, remedies and case study of hosting deep learning frameworks on cloud for application development and finally section 7 provides conclusions and future work.

2. LITERATURE REVIEW

Five major implementations of Open-Source cloud solutions for IaaS namely OpenStack, CloudStack, OpenNebula, Eucalyptus and Nimbus dominate the market. TheoLynn et. al presents comprehensive overview of features of these top five open source frameworks and their comparative analysis based on 19 criteria [3]. OpenStack was chosen for our deployment keeping in mind, its modular architecture, wide adoption, community support, free access to source code, developer documentation and its suitability to private deployments.

OpenStack is free and open-source software for deploying Infrastructure as a Service (IaaS). The technology behind OpenStack consists of a series of interrelated projects delivering various services for cloud infrastructure solution. Each OpenStack service provides one or more endpoints through which user can access resources and perform operations. Some of the important services of OpenStack are compute service, image service, object storage, identity service, networking service, block storage, orchestration service, accounting service, bare metal service and dashboard service. Compute service is provided by **Nova**. Nova manages life cycle of virtual machines, from spin up to termination of virtual machines. Image service is provided by **Glance.** It provides a repository of VM images and snapshots. Object storage is provided by **Swift.** Identity service is handled by **Keystone**. It manages authentication, authorization and OpenStack service information. It provides means to access various OpenStack services by different users. Networking service is provided by **Neutron**. It manages networks and IP addresses. Block storage i.e. persistent block level storage devices for use by VM are provided by **Cinder.** Orchestration service is provided by **Heat.** Heat provides an orchestration engine to launch group of machines, networks and other resources based on templates. A heat template is used to describe infrastructure resources (servers, volume, users) for an application in text file. Account of user's usage of different components of OpenStack

ISBN - 978-93-95038-50-8

is provided by **Ceilometer**. **Ironic** provides bare metal service where instances of physical hardware can be spawned. All of these services can be managed though a web based dashboard service called **Horizon** or through APIs or through command line interface. Horizon provides a web based UI for users and administrators. Interaction between these services is illustrated in Figure 2. We have used Kernel Virtual Machine (KVM) hypervisor in our cloud deployment. Choice of KVM hypervisor is based on two reasons:

(i) It is default hypervisor for OpenStack and widely adopted hypervisor in the OpenStack community.

(ii) It is feature complete and free from license charges and restrictions.

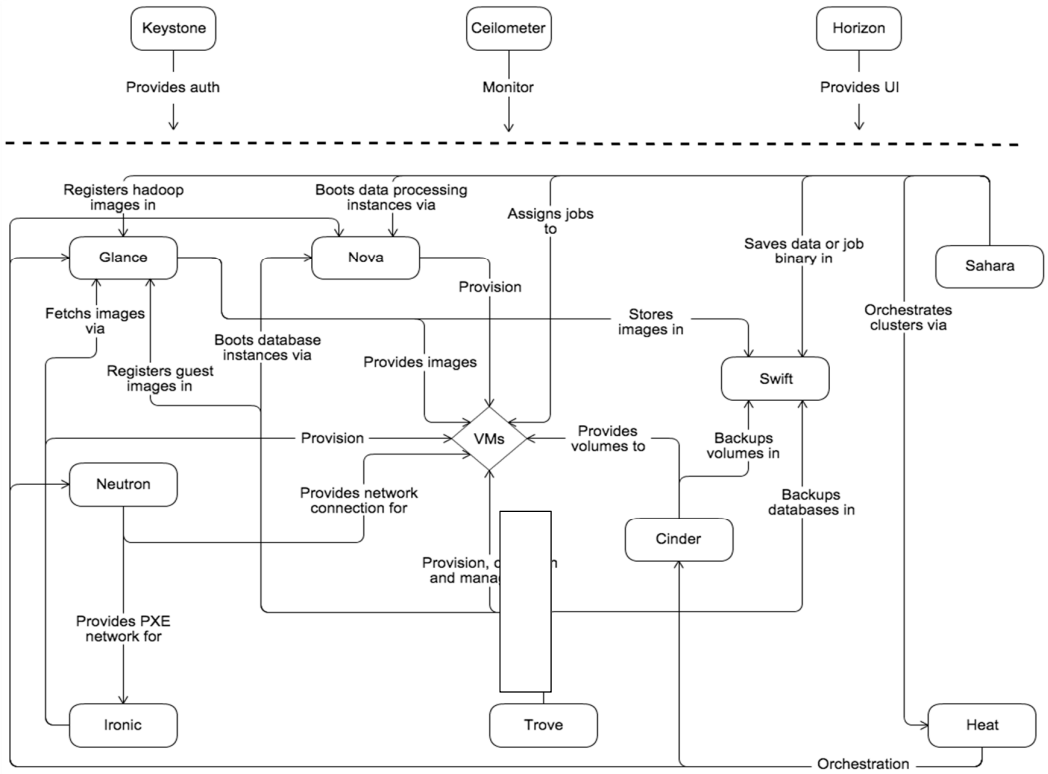

Figure 2: OpenStack core services interaction. (Source: docs.openstack.org) [4]

3. PROBLEM DEFINITION

The goal of this work is to design a scalable architecture for private cloud and implement a private cloud solution aiming at maximum automation and minimal amount of manual work. The Storage solution of the private cloud should provide for redundancy and also data security.

4. METHODOLOGY

4.1. SETTING UP THE PRIVATE CLOUD

OpenStack based private cloud computing system was installed on group of servers mounted on two racks in lab environment. The hardware used for the setup consists of 22 high performance dell power edge servers. The servers are deployed in the following fashion.

I. Three servers as controller nodes, running all sub-services except nova-compute.

II. Sixteen servers as compute nodes which run nova-compute.

III. Three servers as storage nodes with ceph storage cluster configured.

Each compute and controller server has 20 cores, 128 GB RAM and attached 4TB hard disk. Each storage server has 8 cores, 256GB RAM, 9×4TB hard disk. Each server has 4 Network Interface Cards (NICs). Servers are connected using 1 Gigabit ethernet switch for cloud management and 10 Gigabit ethernet switch for VM traffic. All compute nodes are mounted on single rack. Controller nodes, storage nodes and switches are mounted on the second rack. Figure 2 shows distribution of OpenStack services across different nodes.

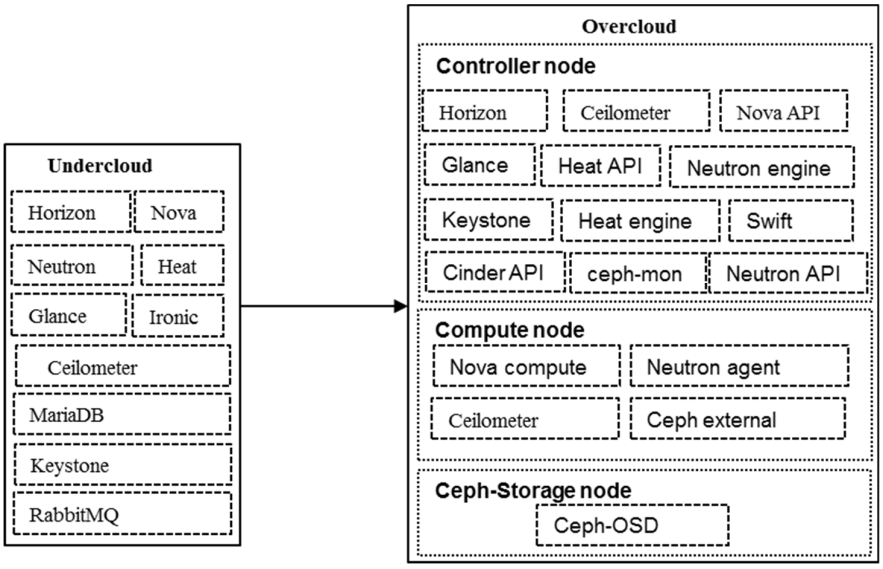

Figure 3: OpenStack Services deployment overview

4.2. DEPLOYMENT PLAN

TripleO (OpenStack On OpenStack) was used to deploy OpenStack. TripleO is a tool to install and manage OpenStack cloud using OpenStack's own services building on Nova, Neutron and Heat. Its aim is to install, test, manage and update it's overcloud servers. Deployment of OpenStack has been done in two steps: Undercloud deployment and overcloud deployment. Undercloud or director is a single node OpenStack installation deployed on virtual machine. This contains necessary OpenStack components to deploy and manage overcloud or OpenStack cloud. Undercloud and Overcloud deployments are elaborated in subsequent subsections. Figure 3 shows services deployment in overcloud and undercloud. Figure 4 shows architecture of the designed cloud computing system.

Table 1: Specification of workstation hosting director and services VM

Processor	x86_64
Cores	20
Threads	2
Virtualization	KVM
Memory	256GB
Disk	4TB

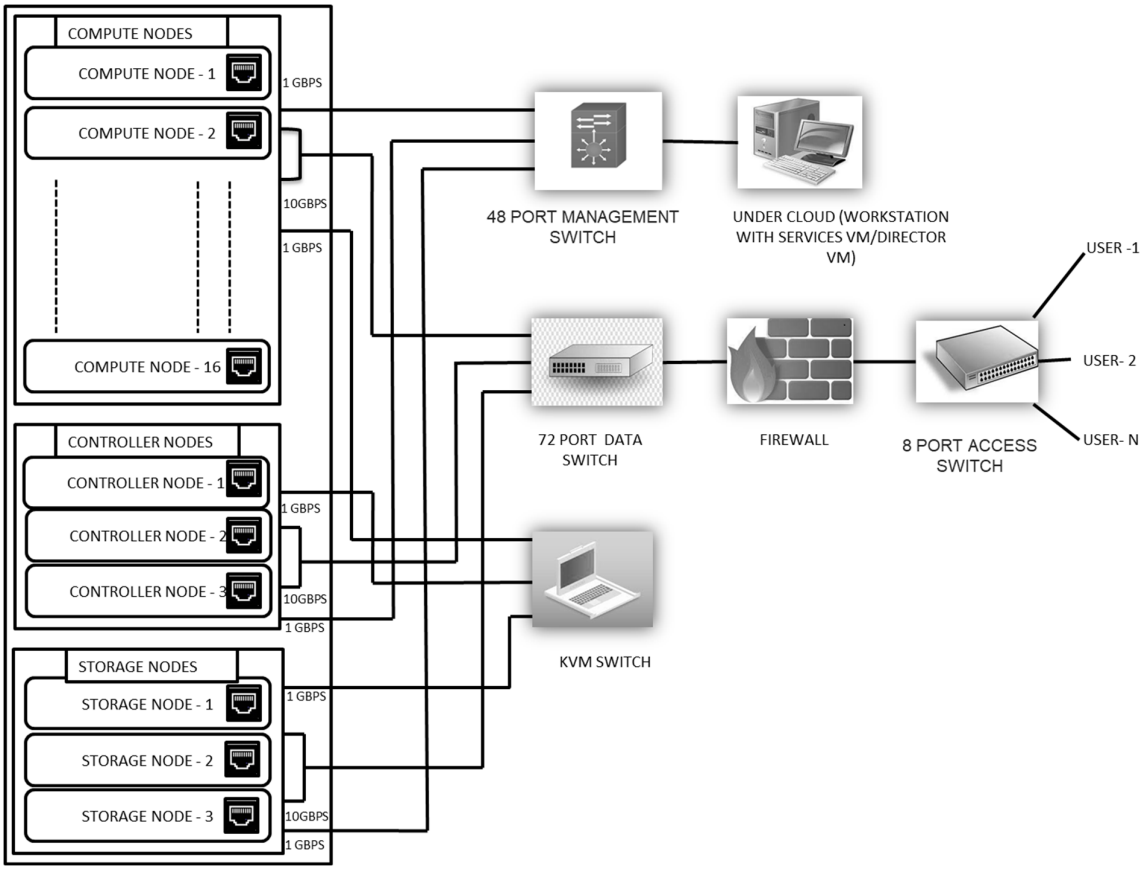

Figure 4: OpenStack based Private Cloud Architecture

4.3. UNDERCLOUD DEPLOYMENT

A dedicated workstation (specification in Table 1) was used for deploying undercloud.

Two Virtual machines (VMs) were installed on this workstation. First VM (specification in Table 2) hosts DHCP, NTP and DNS services. Second VM (specification in Table 3) hosts director. Director and services were chosen to be hosted in virtual machines as they can be cloned for backup. In the director VM, TripleO client was installed. Upon successful installation of TripleO client, undercloud.conf file was generated. Configuration parameters of undercloud were edited and updated in undercloud.conf. After this, undercloud was installed using the command - openstack undercloud install. Successful installation of undercloud created two files named stackrc.conf and password.conf. This completed Undercloud deployment. Overcloud disk images are downloaded from internet and placed in local repository before commencing overcloud deployment.

Table 2 Specification for services VM

Host CPU	1
Memory	1024MB
Disk Storage	20GB

Table 3 Specification for director VM

Host CPU	10
Memory	24576MB
Storage	100GB

4.4. OVERCLOUD DEPLOYMENT

Deployment of overcloud comprises the following steps – registration, introspection, profile tagging, preparation of configuration files and deployment.

- Registration: Overcloud servers were registered in undercloud. Intelligent platform Management Interface (IPMI) related information like IDRAC IP address, username, password, MAC of PXE enabled interface of each server were specified in instackenv.json. Importing this json file completed registration.
- Introspection: Inspection of hardware nodes was done by running introspection agent though PXE. This agent fetched hardware properties of nodes and stored in local database. This information was used in profiling tags.
- Profile tagging: Each node was tagged into specific profiles – control, compute or ceph according to roles chosen for nodes.
- Deployment files were prepared using configuration files. TripleO client provided generic YAML configuration files. Entries in these files were edited to describe environment of overcloud according to the architecture. The environmental files used were *network*-environment.yaml, network-isolation.yaml, roles.yaml, layout.yaml, storage-environent.yaml, network-isolation.yaml and node-info.yaml.
- Deployment: Overcloud deployment was commenced using command - openstack overcloud deploy. Each round of deployment took 3 to 4 hours. After 8 weeks of rigorous installations, consulting and following up on forums, installation was concluded.

5. DISCUSSION ON DESIGN CHOICES

OpenStack based private cloud deployment as discussed in section 4 was mostly automated after a minimal amount of manual work including physical racking, interconnect, MAC-to-IP assignment and power configuration [5]. Our focus was on key design issues as they are hard to modify post deployment. Designing a cloud involves selecting hardware, hypervisor, cloud software and operating system, configuring storage and network interfaces. Rationale behind selection of KVM as hypervisor and OpenStack as cloud software has been discussed in Introduction section. This section sheds light on other design issues.

To ensure symmetry in hardware and its configurations of the physical nodes, servers with same hardware specification were chosen. This symmetry offers following advantages:

(i) Managing spare servers is easy as we are storing only one type of server and replacing of nodes is easy in case of node failures.

(ii) Troubleshooting of software issues are simplified as error terminology remains same throughout complete cloud system.

ISBN - 978-93-95038-50-8

By default, all servers had single redundant array of independent (RAID) disk. Owing to this, in case of any controller node failure, risk of cloud system failing is high. To mitigate this, 4TB disk was added and configured as RAID2 in all controller nodes.

OpenStack uses concept of isolated networks to host specific network traffic type i.e. different network traffic type traverses through different isolated networks (mentioned in Table 4). Ideally each network required one network interface. We have more isolated networks (six) and less network interface cards (four). To remedy this, Virtual Local Area Networks (VLAN) were used. 5 VLANs were configured (as shown in Table 5) to efficiently use available interfaces. Isolated networks were assigned as follows: One physical 1G network port was assigned for IPMI and provisioning network. One network port was assigned for VLAN20 (Internal API) as VMs communicate with higher bandwidth. Two 10G network ports are bonded using Link Aggregation Control Protocol (LACP). LACP offers aggregation, failover and load balancing. This bonded interface was assigned to VLAN30, VLAN40, VLAN50 and VLAN60.

Table 4 Isolated networks in OpenStack

Isolated Network	Role	Connection
Intelligent Platform Management Interface (IPMI)	Power management	Controller, compute and ceph storage
Provisioning	Undercloud control plane for deployment and management	Controller, compute and ceph storage
External API	Public OpenStack API, Access Horizon and floating IPs	controller, compute and undercloud
Storage network	Access to storage resources from compute and controller nodes	Controller, compute and ceph storage
Internal API	OpenStack Internal API, Access and RPC between OpenStack components	Controller, compute and ceph storage
Storage management network	Replication and ceph-backend services.	Controller, ceph storage
Tenant network		Controller, compute

ISBN - 978-93-95038-50-8

Table 5 VLAN and network assignments

VLAN No.	Isolated Network
VLAN20	Internal network API
VLAN30	Storage management Network
VLAN40	Storage Network
VLAN50	Tenant network
VLAN60	External API network

6. IMPLEMENTATION CHALLENGES AND RESULTS

Implementation of cloud deployment comprises of downloading required packages and images, registration of nodes, introspection, profile tagging, preparation of configuration files, installing packages on all nodes simultaneously, and troubleshooting errors.

OpenStack's RDO packages of Pike[2] were used for deployment. OpenStack has placed its Redhat Package Manager (RPM) on ftp server. This RPM changes rapidly because of large contributing community and frequent updates. Use of incompatible packages causes deployment to err. To resolve this, we have prepared local satellite repository, where we downloaded all required packages once and used this as satellite repository for deployment.

Troubleshooting of errors during different stages of deployment required technical expertise on OpenStack. OpenStack has many components which are strongly related. Determining which component failed was difficult and time consuming. For example, common error "No valid host found" was encountered during overcloud deployment phase. Enabling debug or verbose did not give much information on what caused the error. There were many possible scenarios causing this error such as IPMI validation, provisioning state, monitoring compute services, identifying mismatch in properties of nodes and assigned flavors.

During the deployment, failure of even a single node to load boot images from PXE booting leads to failure of complete deployment, resulting in longer deployment time. Therefore, first eight compute nodes were deployed along with controller nodes and storage nodes. The remaining eight compute nodes were added to scale out the cloud by updating heat template.

The deployed cloud could provide IaaS, PaaS as service. As case study deep learning frameworks were provided on a virtual machine to train DL models. The benefit of providing deep learning framework on cloud is that large datasets can be used to train the models. The Qcow2[3] image of the VM was uploaded to Glance service of cloud. A user could login into the cloud and spawn the machine learning VM readily. As a case study

[2] Pike is the 16th release of OpenStack: open source infrastructure software

[3] Qcow2 stands for QEMU copy on write. It is a file format for disk images.

ISBN - 978-93-95038-50-8

7. Conclusions and Future Work

This paper presents setting up of multi-node OpenStack cloud and discusses design choices made while setting up private cloud for strategic organization. It also presents a case of providing deep learning frameworks on cloud. The future work will be to deploy container as a service, applications as a service, security analysis of this cloud and work on framework to enhance security of the private cloud.

Acknowledgements

This activity would not have been initiated without encouragement of late Dr. Manuj Sharma, Scientist G, DRDO. I am deeply thankful to him for proving valuable insights on architecture design and overall supervision.

References

1. Mell,P. and Grance,T. (2011), The NIST Definition of Cloud Computing, Special Publication(NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.SP.800-145 (Accessed Febraury 9,2023)
2. Hans Lindgren, Performance management for Cloud Services: Implementation and Evaluation of Schedulers for OpenStack. Kungliga Tekniska Hgskolan, Stockholm, 2013, page 10.
3. TheoLynn, Graham Hunt, David Corcoran, John Morrison, Philip Healy. “A Comparative Study of Current Open-Source Infrastructure as a Service frameworks”. In Proceedings of the 5th International Conference on Cloud Computing and Services Science – Volume 1: CLOSER, ISBN 978-989-758-104-5, pages 95-104. DOI: 10.5220/0005423300950104
4. OpenStack docs: Conceptual architecture.https://docs.openstack.org/ocata/admin-guide/common/get-started-conceptual-architecture.html.
5. Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, Joe Topjian. OpenStack Operations guide. Oreilly, 2014.