IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Deployment of Cloud Infrastructure using Puppet Configuration Tool

Smita S Ereshimi[1], Geetha V[2], Suresh Rathinam[3]

M. Tech in Information Technology, Dept. of ISE, R. V. College of Engineering, Bengaluru, India[1]

Assistant Professor, Dept. of ISE, R. V. College of Engineering, Bengaluru, India[2]

Principal Architect, Cloud Infrastructure Solutions, Unifylabs Systems Pvt Ltd, Bengaluru, India[3]

Abstract: Deployment of cloud infrastructure is very critical and complex task to administrators as well as developers. Problems faced while deploying cloud environment are - more time and effort to install, exhaustive line by line configuration, adding and updating OpenStack services is difficult, need thorough knowledge to install and troubleshoot, does not support parallel execution, no user friendly interface. To address these issues and to automate the deployment of OpenStack cloud infrastructure, Puppet Configuration Management Tool and Python has been used. Automated scripts have been written for deploying OpenStack Infrastructure modules. This project provides light weight appliance, deployment portal, updating and managing all the features and services of OpenStack and KVM (Kernel Virtual Machine).

Keywords: Cloud infrastructure, Open Stack, Puppet.

I. INTRODUCTION

The project, Deployment of cloud Infrastructure targets Infrastructure as a Service offering of cloud computing. In Cloud Computing space there are many Infrastructure as a Service offerings are available like OpenStack, Apache Mesos, Cloud Stack, Core Os etc. But comparing to all IaaS services of cloud computing, OpenStack is a very robust, current industry trend, widely adopted and is very actively contributed opensource cloud platform. So this project using OpenStack infrastructure to build a cloud environment very quickly.
OpenStack is a group of open source projects and it is referred as a cloud operating system. It manages large pools of resources like networking, storage and compute throughout data centres. Administrators can control resources through dashboard. It is very popular and rapidly developing cloud computing platform for the creating a IaaS cloud and it has very huge competitive and innovative market [1]. IaaS create a public and private cloud infrastructure very quickly and it will create, manage and maintain virtual machines efficiently [2]. If users ask for virtual machines OpenStack will provision very quickly, user doesn't need to worry about hardware and other requirements. OpenStack has very robust access controls. Accessing and resource utilization are controlled at the level of users, roles, and projects.

The below diagram is the basic OpenStack architecture storage, compute and network is a fundamental services of OpenStack administrators are accessing these services resources through dashboard. User can access these services using the applications through the APIs.

The OpenStack system consists several key projects that developers can install separately. This project include services like identity service (Keystone), image service (Glance), compute (Nova), block storage (cinder), networking (Neutron), dashboard (Horizon), object storage (Swift). Developers can install any of these projects independently and configure them on different nodes.

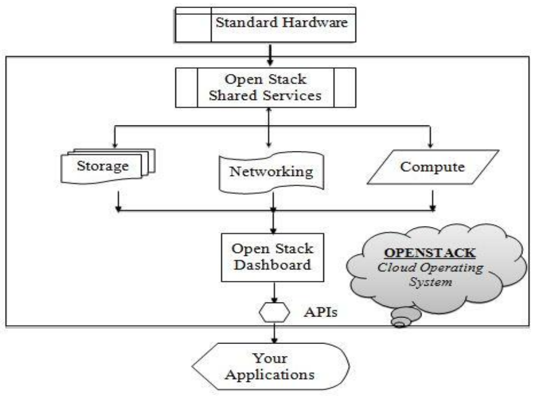


Fig. 1.OpenStack Architecture

II. RELATED WORK

In physical computer installing and configuration of cloud infrastructure is an arduous and complex process whereas the configuration of cloud infrastructure in virtual machine show a great number of performance issues. In this paper deploying the cloud infrastructure with more opening to the cloud software to a large number of new services that could be deployed with lower time , lower complexity and extensible[3].

OpenSim is a simulator of OpenStack service. It creates OpenStack environment and cloud applications. So using OpenSim simulator anyone can deploy cloud environment easily, flexible, less time consuming, etc. [4].

This paper reviews, the three open source technologies, namely OpenStack, OpenDaylight and Open vSwitch. Benefits of each open source technology is different from the stand points of flexibility, cost, customizability and audit ability factors[5].

In this paper deploying the cloud infrastructure in multiple virtual machines on single physical machine which increases the resources utilization and reduce the power consumption. The benefit of virtualized technology is on its on-demand resource allocation and flexible management. It covers the infrastructure required for deployment of OpenStack [6].

III. OVERVIEW OF IMPLEMENTATION

The deployment appliance consists of the template configurations required for deploying all in one node and multi node scenarios. This project provides light weight appliance, deployment portal, updating and managing all the features and services of OpenStack components and pre-requisites.
The web user interface is simple and is used to gather the basic information required for the installation of the cloud. The details gathered are all in one or multi node scenario, IP addresses, username, password and selecting the components required. Python validates the inputs like ip addresses, username and password before proceeding to the component selection. All in one node deploys all the OpenStack components in one node and multi node deploys components on four nodes namely – controller, network, storage and compute. Python compiles the input into automated scripts for deploying OpenStack Infrastructure modules and pre-requisites.
This project used Puppet Configuration Management Tool to install and configure OpenStack components on target nodes. Puppet Master holds the repository of all the configurations that are to be pushed to the intended target agent nodes[7]. Python based web User Interface is used to gather all the information about the cloud environment roles, services and configuration and it compiles all the information and prepares puppet modules and manifests, configure prerequisites and puppet agent, then push the modules and manifests to the required target nodes.
This project supports CentOS 7 operating system for the target nodes. All nodes should have a static IP address and connectivity to the internet. The Turnkey Appliance should be able to communicate with the target nodes on the given IP address. Minimal installation of CentOS 7 operating system is sufficient and no other additional packages are required to be installed by the user/administrator. All the target nodes should have the hostnames set.
Following are the pre-requisites that are installed on the target node(s) both All in one and Multi node scenarios. /etc/hosts file is updated with the respective hostname(s) and static ip addresses of turnkey appliance and the target nodes. NTP is very much essential for ensuring time synchronisation across the nodes so that Puppet jobs work seamlessly. Selinux has been disabled as per the recommendations for installing OpenStack. Firewall is disabled as it is not necessary at the moment. Puppet agent is installed & configured on all the target nodes and registered with the Puppet Master. Puppet agent certificate is generated and registered with the Puppet Master. OpenStack Kilo repository is being installed on all the target nodes.
The OpenStack Components and pre-requisites are installed by the puppet agent using the compiled templates available on the Puppet master as per the components selection and scenario selection. Following components are installed for multimode scenario. RabbitMQ, memcached, mariadb, mongodb, keystone, glance_auth, cinder api, nova_api, neutron server are installed on the controller node. Neutron_router is installed on the network node. Glance_api, cinder volume, cirros, and horizon is installed on the storage node. Neutron agent and nova compute is installed on the Compute node. KVM is automatically installed on the compute node as it is a dependency. All in one node functions as the controller, network, storage and compute node hence all the components mentioned above are installed on the same node.
The Progress of the components installation is being tracked by the web user interface continuously and is able to display a check mark once the installation / configuration of the component is completed. The other alternate way to track the progress is to see the output of the main python script which generates the User Interface and is the core of the entire project. In-case there is an interruption to the deployment due to unplanned outage the setup can be resumed only by restarting the target node and running the puppet agent –t command manually. Once all the components are installed the url of the OpenStack Dashboard is displayed. Using the Template

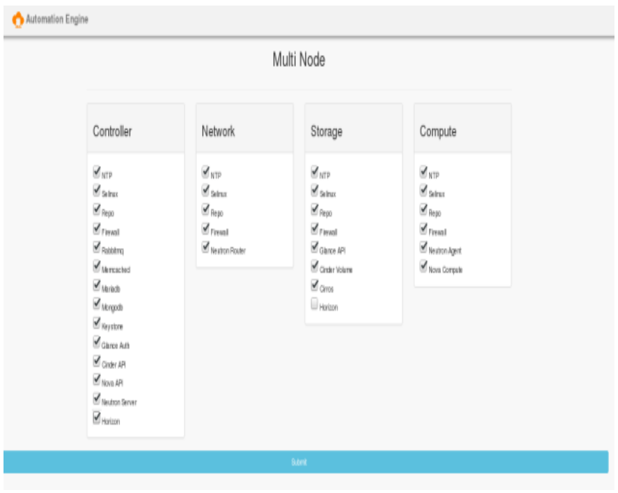
After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

IV. RESULTS

The purpose of this project is to reduce the overhead of the exhaustive manual configuration that is to be performed by administrators and developers in building a cloud infrastructure. The project is built in a modular fashion to facilitate addition or removal of OpenStack components as necessary. The execution model of the project is parallel so that cloud can be deployed at shorter span of time and zero manual effort across large number of servers in massive deployment scenarios. The project user interface is built as a very simple intuitive user friendly wizard and can be used by any person who need not have the knowledge about OpenStack or cloud or the advanced configurations underneath it.

The administrator and user can deploy cloud infrastructure in either all in one node or multimode scenario. UI Collects information about IP address and credential of target node. Python code internally will check given IP addresses and credentials are available then it will display servers status and hostnames.

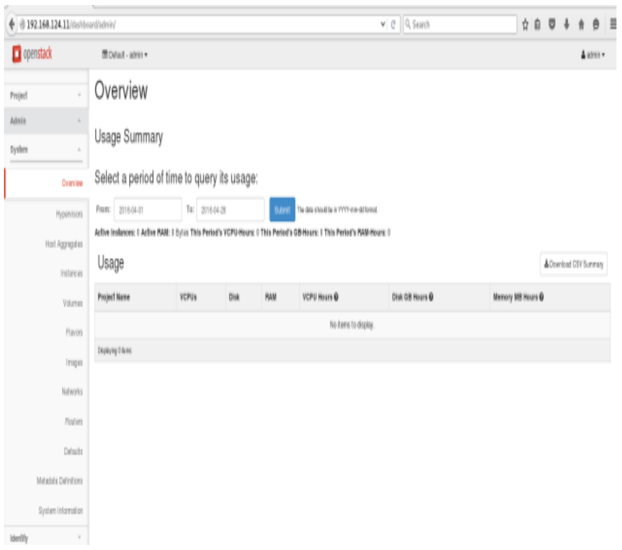
Then select the required OpenStack components to be deployed. Python code will compile the information into executable scripts, puppet manifests will be updated with respective values and selected components in the ui.



Controller	Network	Storage	Compute
✓ NTP	✓ NTP	✓ NTP	✓ NTP
✓ Selinux	✓ Selinux	✓ Selinux	✓ Selinux
✓ Repo	✓ Repo	✓ Repo	✓ Repo
✓ Firewall	✓ Firewall	✓ Firewall	✓ Firewall
✓ Rabbitmq	✓ Neutron Router	✓ Glance API	✓ Neutron Agent
✓ Memcached		✓ Cinder Volume	✓ Nova Compute
✓ Mariadb		✓ Cirros	
✓ Mongodb		☐ Horizon	
✓ Keystone			
✓ Glance Auth			
✓ Cinder API			
✓ Nova API			
✓ Neutron Server			
✓ Horizon			

Python will do the pre-check activities and it will prepare the setup by adding host entries, puppet master will install puppet agent to target node then puppet agent target node gets certificate from puppet master. Puppet master will give certificate approval to puppet agent node then puppet master will start deploying OpenStack components to target node.

After deploying all the components of OpenStack, it will display dashboard URL. User and administrator can open dashboard from using the URL and login using the hardcoded credentials i.e.



Overview

Usage Summary

Select a period of time to query its usage:

From: 2016-04-01 To: 2016-04-28 The date should be in YYYY-mm-dd format.

Active Instances: 0 Active RAM: 0 Bytes This Period's VCPU-Hours: 0 This Period's GB-Hours: 0 This Period's RAM-Hours: 0

Usage

Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours	Memory MB Hours
No items to display.						

Displaying 0 items

V. CONCLUSION

This project work is aimed at solving the deployment complexities of OpenStack cloud for administrators and users who do not have skills or knowledge in OpenStack deployment. Despite the fact that OpenStack is more popular and its components are maturing day by day there isn't a streamlined way of implementing OpenStack in a hassle free manner. This fact is the key-driver of the project. This project addresses two scenarios, all in one and multi node implementations. A lot of new components are being added to OpenStack in every release; this project is modular and can support the addition of new components. This project also addresses issues like configuration mismatch, human configuration errors, implementation time, complexity in monolithic deployments, etc.

VI. FUTURE ENHANCEMENTS

The following are the future enhancements

a) Add support for Operating systems like Ubuntu, RedHat Enterprise Linux
b) Add support for VMware vSphere Esxi and Hyper-V as compute nodes
c) Automatic deployment of Operating system on baremetal hardware
d) More customizable options for network and other components
e) Provision more nodes for Multi node
f) Add more OpenStack Services

REFERENCES

[1] Shalmali Suhas Sahasrabudhe "Comparing OpenStack and VMware" Advances in Electronics, Computers and Communications (ICAECC), 2014 International Conference on, 1 – 4, 2014.
[2] Thomas Voith, Karsten Oberle, Manuel Stein "A Path Supervision Framework – a key for service monitoring in Infrastructure as a Service (IaaS) Platforms" 36th EUROMICRO Conference on Software Engineering and Advanced Applications, 28, 2010.
[3] Enrique Chirivella-Perez "Hybrid and Extensible Architecture for Cloud Infrastructure Deployment" 2015 IEEE International Conference on Computer and Information Technology, 611, 2015.
[4] Dinkar Sitaram, H. L. Phalachandra "OpenSim: A Simulator of OpenStack Services", 2014 8th Asia Modelling Symposium, 90, 2014.
[5] Kapil Bakshi "Network Considerations for Open Source based Clouds" IEEE International conference, 2015.
[6] Robayet Nasim, Andreas J. Kassler "Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware" IEEE 38th Annual International Computers, Software and Applications Conference Workshops, 84-89, 2014.
[7] Johannes Wettinger, Michael Behrendt "Integrating Configuration Management with Model-Driven Cloud Management Based on TOSCA" 3rd International Conference on Cloud Computing and Service Science, 437-446, 2013.