IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

Firewall as a Service in Cloud Infrastructure

Pavan Harapanahalli B[1], Suresh Rathinam[2], Minal Moharir[3]

Computer Science, R V College of Engineering, Banglore, Karnataka, India[1, 3]

Cloud Infrastructure Service, Unifylabs System Pvt Ltd, Banglore, Karnataka, India[2]

Abstract: OpenStack is most widely used cloud platform to set up private cloud. OpenStack offers Compute, Identity, Image and Networking services to deploy cloud platform. Firewall as a Service (FWaaS) of OpenStack secures tenant's network. OpenStack installers automate OpenStack services deployment to reduce human efforts involved. Currently none of the installers support FwaaS deployment nor support different scenarios of OpenStack Network service deployment like Distributed Virtual Routing (DVR), Virtual Router Redundancy Protocol (VRRP). The proposed work focuses on automating OpenStack basic services deployment with different scenarios of Network Service and FWaaS with Graphical User Interface (GUI) using Ansible tool. This work has made OpenStack service deployment very easy with GUI, reduced human efforts and errors due to human mistakes. This work efficiently sets up OpenStack cloud with FWaaS supporting different scenarios of OpenStack-Neutron on multi cluster environment.

Keywords: Cloud Computing, Distributed Virtual Routing, Firewall as a Service, OpenStack, Virtual Router Redundancy Protocol, Ansible.

I. INTRODUCTION

OpenStack is open source software which manages large pool of cloud resources. Compute, storage, network are main resources OpenStack offers to its tenants. These resources are configured and managed through OpenStack REST full API, or through command line interface, or using OpenStack dash board [1, 2]. OpenStack architecture consists of a control node, one or more compute nodes and network node. Optionally it may consist of one or more storage nodes. Three node architecture of OpenStack consists of one control node, one compute node and one network node [3].

A. OpenStack Services

Basic OpenStack Services are as follow [4].

Identity Service: Keystone is project name given for Identity service of OpenStack. Keystone provides an authentication service for all other OpenStack services. It provides accessible endpoints for all OpenStack services, using which users/tenants and other OpenStack services interact.

Compute Service: Compute service is named as Nova in OpenStack. Nova creates and manages Virtual Machine (VM) instances.

Network Service: Network Service is called as Neutron. Neutron is responsible for creation of project networks between VMs and connecting VMs to external network.

Neutron controls and manages virtual switches and virtual routers. Image Service: Images Service in OpenStack is known as Glance. Glance is responsible for Storing and retrieving virtual machines disk images. OpenStack Nova makes use of Glance during instance provisioning. Controller node runs Identity service, Image Service, management portions of Compute and Networking, Networking plug-in and the dashboard. Network node runs Networking plug-in and several agents that provision tenant networks and provide switching, routing, NAT and DHCP services. L3 agent is responsible for routing and NAT operations. DHCP agent is responsible for DHCP service. OpenvSwitch agent or Linux Bridge agent is responsible for switching operations. This node also handles external (Internet) connectivity for tenant virtual machine instances.

Compute nodes runs hypervisor portion of Compute that operates tenant virtual machines or instances.

Compute nodes also runs the Networking plug-in and agents that connect tenant networks to instances and provide security groups services. A security group is named collection of network access rules that are use to limit the types of traffic that have access to instances.

II. RELATED WORK

OpenStack supports VLAN, Generic Routing Encapsulation Routing (GRE) and Virtual Extensible LAN (VXLAN) for scalability. VLAN uses only 12-bit VLAN ID to identify each network uniquely; only 4096 network with different VLAN ID is possible, which is considered to be very low for scalability. Tunnelling technologies like GRE and VXLAN can be used to provide high scalability [5-9].

OpenStack supports different technologies for Layer 3 and Layer 2 operations. Linux Bridge provides same functionality as normal layer 2 switch provides, but Linux Bridge (LB) is not flexible for virtual and multitenant cloud environment. Alternative to Linux Bridge is OpenvSwitch (OVS), and it is highly flexible for virtual environments [10-12].

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

Authors of [13] studies different OpenStack installers. Currently available installers though install OpenStack services efficiently lacks GUI features i.e. users not have privileges to select among available OpenStack Networking technologies between VLAN, VXLAN, GRE based on users requirement. And also none of the currently available installers support different scenarios of OpenStack Neutron nor support Firewall as a Service. This work automates OpenStack service deployment with user friendly GUI, support different scenarios of Neutron and Firewall as a Service.

III. OPENSTACK NETWORK SCENARIOS

This section describes architectures of Neutron Networking scenarios. Neutron can be deployed in one among the following three networking scenarios. Description of each of them is given below.

1) Classic scenario.
2) Distributed Virtual Routing (DVR) scenario.
3) Virtual Router Redundancy Protocol (VRRP) scenario.

A. Classic Scenario

This scenario describes installation of OpenStack Neutron either using Linux Bridge or OpenvSwitch agent. Here either Linux Bridge or OpenvSwitch agent manages virtual switches. Currently available installers install OpenStack networking service with classic OpenvSwitch scenario. Users have to choose either Linux Bridge or OpenvSwitch agent. Figure 1 shows architecture of classic scenario.

On network node

1) Either Linux Bridge agent or OpenvSwitch agent manage virtual switches and connectivity among them.
2) Dynamic Host Configuration Protocol (DHCP) agent manages DHCP service for assigning IP addresses for VMs.
3) L3 agent handles routing and NAT operation for instances, routing traffic between project and external networks.

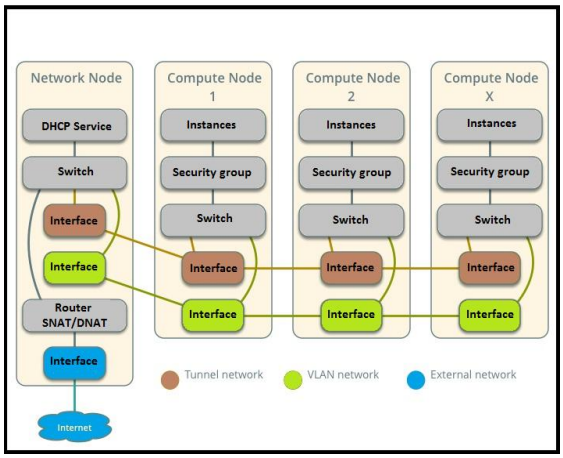

Figure 1: Architecture of Neutron classic scenario

On compute nodes

1) Either Linux Bridge agent or OpenvSwitch agent manage virtual switches and connectivity among them.
2) Linux Bridge agent handles security groups for instances.

B. Distributed Virtual Routing (DVR) Scenario

In previous scenario, to route a data traffic from VM to external network, routing operation is resides completely in network node. To eliminate single point of failure of network node DVR scenario has proposed. In DVR scenario, routing operation resides in compute nodes also. Figure 2 shows architecture of DVR scenario.

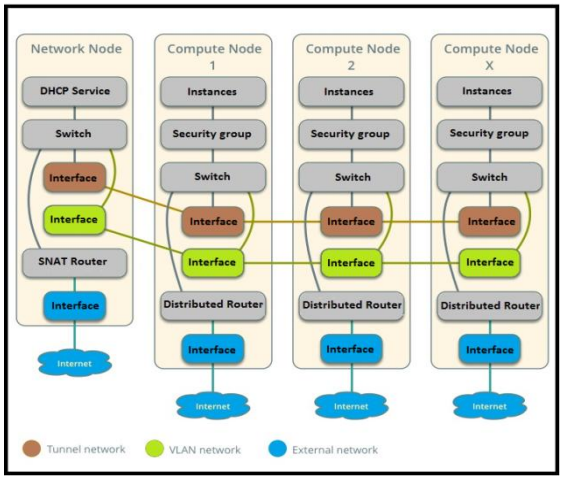

Figure 2: Architecture of Neutron DVR scenario

On network node

1) Open v Switch agent manages virtual switches and connectivity between them.
2) L3 agent handles routing and NAT operation for instances with a fixed IP address, routing traffic between project and external networks.
3) DHCP agent handles DHCP services for VMs.

On compute nodes

1) Open v Switch agent manages virtual switches and connectivity between them.
2) L3 agent for managing routing and NAT operations for instances with a floating IP address, routing traffic between project and external networks to eliminate single point of failure.
3) L3 agent manages routing for instances with a fixed or floating IP address using project networks on the same distributed virtual router.
4) Linux Bridge agent manages security groups for VMs.

C.Virtual Router Redundancy Protocol (VRRP) Scenario

This scenario describes a high-availability (HA) implementation of the OpenStack Networking service. This scenario installs Neutron either using Linux Bridge or OpenvSwitch. VRRP provide high availability routing by supporting random distribution of routing on different network nodes. Figure 3 shows architecture of VRRP scenario.

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

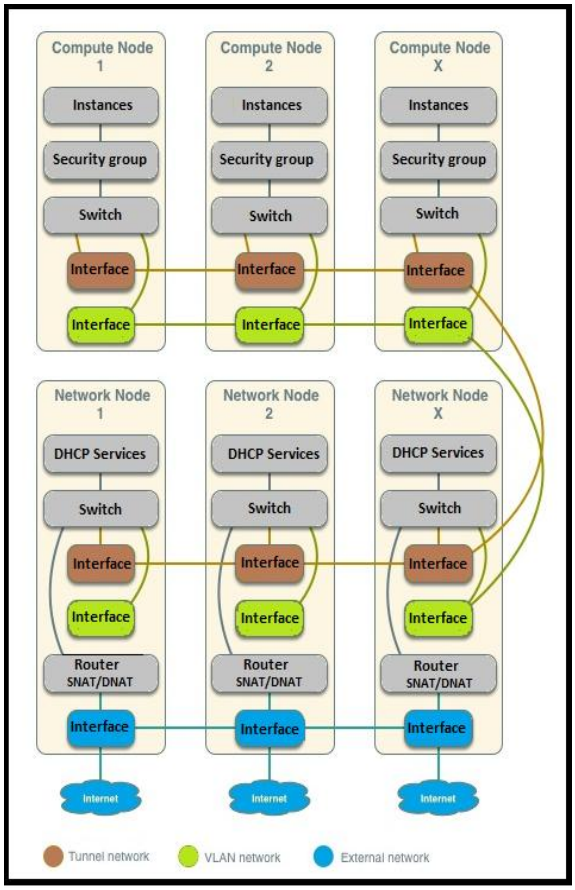

Figure 3: Architecture of Neutron VRRP scenario

On network nodes

1) Either Linux Bridge agent or OpenvSwitch agent manage virtual switches and connectivity among them.
2) DHCP agent manages DHCP service for assigning IP addresses for VMs.
3) L3 agent handles routing and NAT operation for instances, routing traffic between project and external networks.

On compute nodes

1) Either Linux Bridge agent or OpenvSwitch agent manage virtual switches and connectivity among them.
2) Linux Bridge agent handles security groups for instances.

IV. FIREWALL AS A SERVICE (FWAAS)

By deploying FWaaS [14] in cloud infrastructure users can create firewall rule, firewall policies, firewalls using Networking API calls to protect tenants' networks. Neutron FWaaS is configured on the nodes running the Neutron L3 agent, and Neutron Server API configured on the controller node to pick up the service.

Users can also expose the FWaaS feature in Horizon (Dashboard) on the controller node. With a Neutron firewall in place on the L3 router, any traffic traversing that router will be inspected there before it is allowed to continue.

V. PROPOSED WORK AND IMPLEMENTATION

For implementation, this work has used Python 2.7 and Ansible [15] playbook. Ansbile playbook is an efficient tool used to configure group of nodes remotely. Ansible playbooks contain several tasks to be executed on group of nodes. These Ansible playbooks are stored in .yml format hence Ansible playbooks are also called as yml files in this work. Python is used to handle GUI. Architecture of proposed system is shown in figure 4.

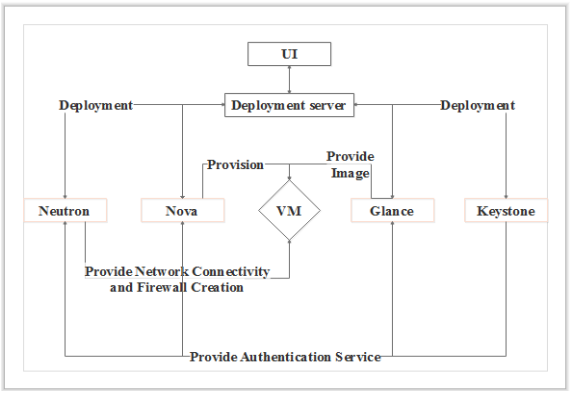

Figure 4: System architecture of proposed work

A deployment server which is running an application, gets configuration inputs from users and generates yml files for different OpenStack services for different nodes, for example yml file is created for keystone service will configure control node to deploy keystone. These yml files contain several tasks such as installing OpenStack packages, configuring files, starting services, etc to configure control node, compute nodes and network nodes. Following yml files will be generated to deploy proposed services.

Keystone: One yml file to configure control node, because keystone service is configured only on control node.

Glance: One yml file to configure control node.

Nova: One yml files to configure control and one to configure compute nodes.

Neutron: One yml files to configure control node, one yml file configure compute nodes, one yml file network node/s (Incase of VRRP).

FWaaS: One yml file to configure network node/s. In case of DVR one yml file to configure compute nodes. One yml file to configure control node.

These yml files are executed one by one in an order, to deploy Keystone, Glance, Nova, Neutron (with selected scenario, tunneling technology and virtual switch technology) and FWaaS. Any error that occurs during execution will be thrown to GUI with details. This system will not let users to deploy next service until users recovers the error. Most of the errors are mainly because of user entered values through GUI text boxes such as wrong IP addresses or URLs. In most of the cases errors can be fixed by re-entering these values correctly. Ansible is designed in such way that it erases previous configuration and changes the configuration as per new inputs.

DOI 10.17148/IJARCCE.2016.5929

IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 9, September 2016

VI. PERFORMANCE ANALYSIS

There are several methods in which OpenStack services can be deployed. Let's analyze human efforts involved in these methods. OpenStack services can be deployed manually; it's been observed that users have to execute about an average 356 operations on three node architecture to deploy these basic services with FWaaS [14]. And also there is a possibility that human can make mistakes in any of these operations, to fix these errors there is need of more human effort. Obviously human effort is directly proportion to consumption of time.

Another approach is using only yml (without GUI) files to automate these human operations to deploy OpenStack Services, it completely reduces human interaction with a process of cloud deployment and hence mistakes of humans can be reduced completely. But human effort is still needed to create these yml files again and again for different services for different nodes. One more approach is generating yml files based on user inputs through GUI. In this approach human effort is to enter details like IP address, URLs, selecting technologies that users wants to deploy etc. In this approach there may be possibility of error occurrence because of the values entered through text boxes of GUI, such as IP addresses etc.

But human efforts required to deploy cloud have reduced significantly in this approach. Figure 5 shows a comparison of error occurrences in manual approach and automated approach. From the figure it is observed that error occurrence reduces significantly in automated approach, and error occurrence drastically reduces with increase in multi cluster size, this is due to the fact that one single yml file is enough to execute similar tasks on group of nodes.

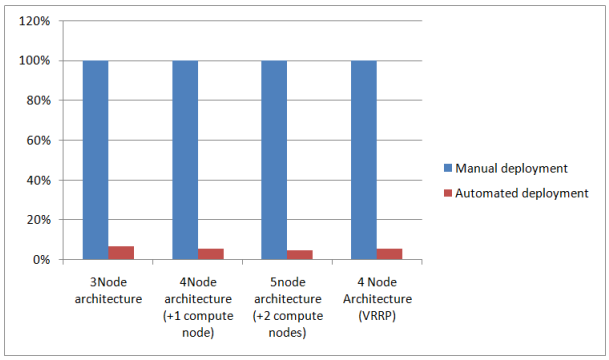

Figure 5: Comparison of error occurrences in manual approach and automated approach

VII. CONCLUSION AND FUTURE WORK

This work automates deployment of OpenStack Firewall as a service, Neutron with users selected network scenarios, technologies and other basic services of OpenStack with GUI. This system supports multi node cluster cloud environment. Behavior of firewall is tested on different Neutron scenarios. Still there is a need of system which will troubleshoot error that occurs during deployment, based on error patterns. And also there in need to automate other OpenStack services like Load Balancer as a Service etc.

REFERENCES

[1] O. Khedher, Mastering OpenStack. Birmingham: Packt Publishing, 2015.Print.
[2] O. Sefraoui, M. Aissaoui and M. Eleuldj, "OpenStack: Toward an Open-source Solution for Cloud Computing", International Journal of Computer Applications, vol. 55, no. 3, pp. 38-42, 2012.
[3] A. Leiter and R. Fekete, "An Openstack Integration Method with Linux Containers", 19th International ICIN Conference - Innovations in Clouds, Internet and Networks19th International ICIN Conference - Innovations in Clouds, Internet and Networks , Paris, pp. 219-221, 2016.
[4] Hewlett-Packard, "Red Hat Enterprise Linux OpenStack Platform on HP ConvergedSystem 700x", Technical white paper, November 2014.
[5] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, 2000.
[6] M. Mahalingam, D. Dutt, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, 2014.
[7] R. Davoli and M. Goldweber, "VXVDE: A Switch-Free VXLAN Replacement," 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, 2015, pp. 1-6.
[8] R. Kawashima and H. Matsuo, "Non-tunneling Edge-Overlay Model Using OpenFlow for Cloud Datacenter Networks," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, 2013, pp. 176-181.
[9] S. Jeuk, G. Salgueiro, F. Baker and S. Zhou, "Network segmentation in the cloud a novel architecture based on UCC and IID," Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on, Niagara Falls, ON, 2015, pp. 58-63.
[10] J. T. Yu, "Performance Evaluation of Linux Bridge," In Proc. Telecommunications System Management Conference, Louisville, KY, Apr. 2004.
[11] B. Pfaff, et al., "Extending Networking into the Virtualization Layer," In Proc. 8th ACM HotNets, New York, NY, Oct. 2009.
[12] F. Callegati, W. Cerroni, C. Contoli and G. Santandrea, "Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case," Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, Luxembourg, 2014, pp. 132-137.
[13] A. Awasthi and R. Gupta, "Comparison of OpenStack Installers", IJISET - International Journal of Innovative Science, Engineering & Technology, vol. 2, no. 9, pp. 744-748, 2015.
[14] K. Jackson, C. Bunch and E. Sigler, OpenStack Cloud Computing Cookbook - Third Edition. Packt Publishing, 2015.
[15] Red Hat, "The Benefits of Agentless Architecture", Technical white paper, October 2016.

BIOGRAPHIES

Pavan Harapanahalli B, Final year student of M.tech in Computer Network Engineering from R V College of Engineering, Banglore, Karnataka, India.

Suresh Rathinam, Principal Architect, Cloud Infrastructure Services, Unifylabs Systems Pvt Ltd, Banglore, Karnataka, India.

Minal Moharir, Associate Professor, Department of Computer Science, R V College of Engineering, Banglore, Karnataka, India.