Exploring Advanced Features in OpenStack

Project Report

Team Members:

Abhijit Bhandarkar – B00659257

Nishant Shah – B00659531

Vidhi Kamdar – B00659440

Contents:

Introduction

- Maintaining, configuring, networking of servers and managing resources became extremely difficult, complex and consumed a lot of time.
- This is when virtualization came in the role.
- Virtualization added hypervisors over servers which enabled users to deploy application softwares on virtual machines rather than physical machines.
- But with increasing number of servers, there were as many or more hypervisors to manage and there was a need to automate those underlying complex tasks from the developers.
- OpenStack turns all sets of hypervisors from different data centers into pools of resources and makes it easy to manage those resources from the GUI tools such as dashboard or command line tools.

What is OpenStack?

- OpenStack is a set of open-source software tools for building and managing cloud computing platforms for public and private clouds. It is used by major companies such as RedHat, HP, Dell, etc.
- OpenStack falls is an Infrastructure as a Service (IaaS). computing for end users in a remote environment, where the actual software runs as a service on reliable and scalable servers rather than on each end-user's computer.
- OpenStack lets users deploy virtual machines and other instances that handle different tasks for managing a cloud environment on the fly. Can be used for Orchestration and hybrid clouds

Basic components of OpenStack:

- **Nova Compute Engine:**
    - Used for deploying and managing virtual machines and other instances to handle computing tasks
- **Swift Object Storage:**
    - Storage system for objects and files where you can refer to a unique identifier referring to the file or piece of information and let OpenStack decide where to store this information.
- **Cinder Block Storage:**
    - Block storage component where you can refer to files by their specific location on a disk drive.
- **Neutron Networking:**
    - Provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently.

- **Horizon Dashboard:**
    - Dashboard i.e., the graphical interface for OpenStack
- **Keystone Identity Services:**
    - Provides identity services for OpenStack. It is essentially a central list of the users of the OpenStack cloud, mapped against all of the services provided by the cloud, which they have permission to use.
- **Glance mage Service:**
    - Glance provides image services which can be used as templates when deploying new virtual machine instances.

OpenStack Installation:

- Either setup Local Virtual Machine or create one on the cloud.
- We created an instance on Google Cloud Platform with the following configuration.
    - CentOS 7 on Google Cloud Platform
    - CPU Platform: Intel Haswell; 4 vCPUs
    - Memory: 15 GB; Disk Space: 100 GB
    - Packstack utility to deploy parts of OpenStack
    - NOVA Compute, NEUTRON Networking, GLANCE Image Service, SWIFT Object Storage, CINDER Block Storage, Horizon Dashboard

GNOME for GUI:

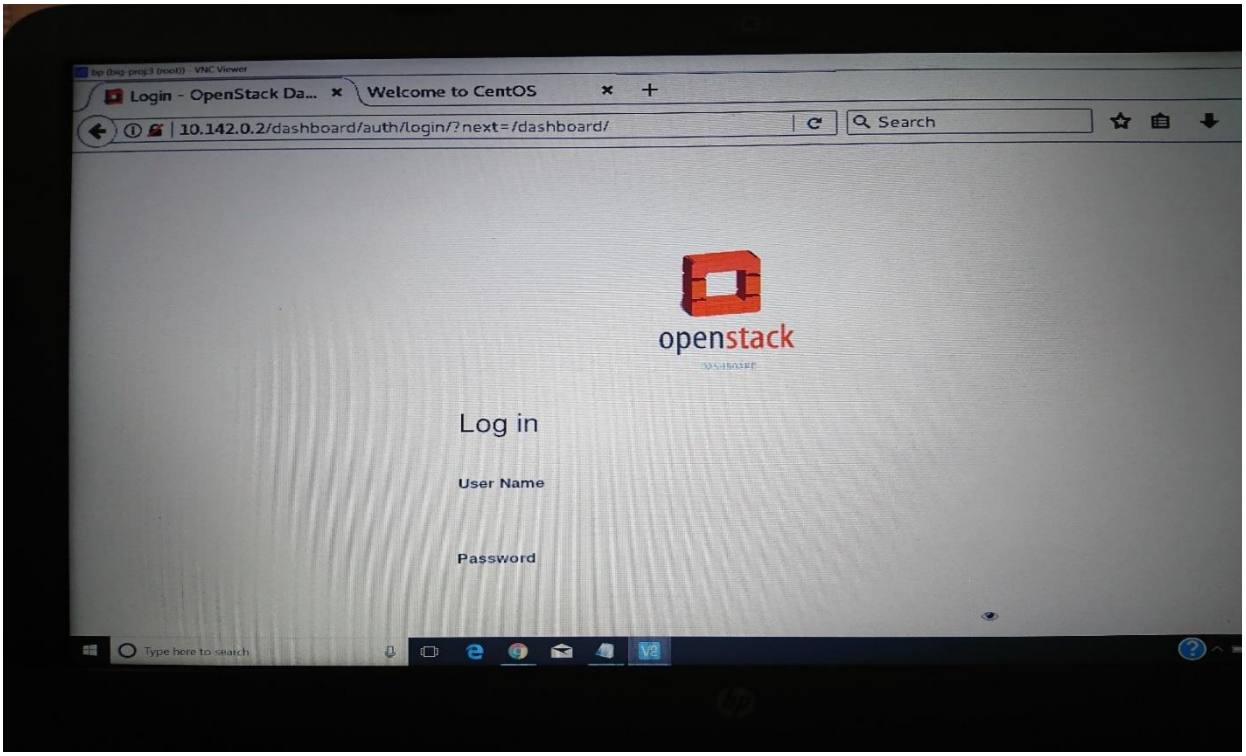
- GNOME for GUI access
- TigerVNC on CentOS and RealVNC on local system
- Setup service, password, and firewall rules
- Start VNCserver and connect to OpenStack Dashboard via Internal IP

Commands to start VNC Server and set up firewall rules:

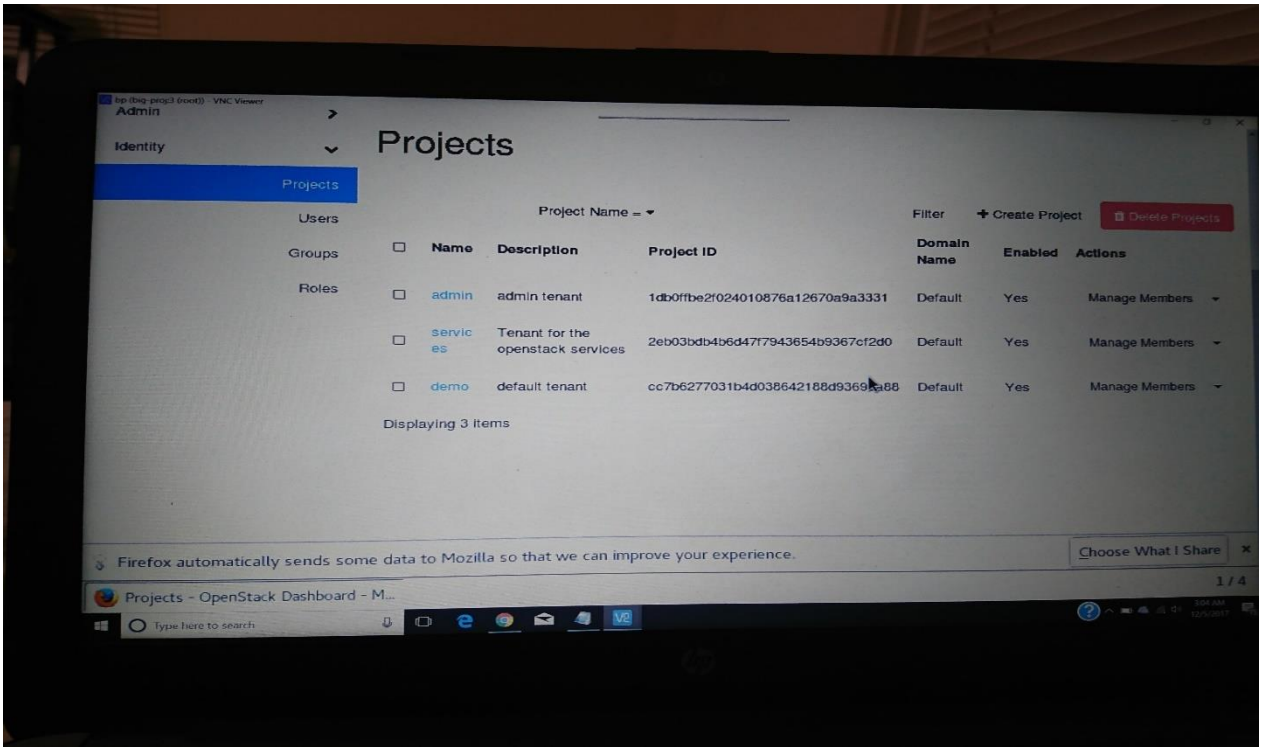
- Enable VNC service at startup
    - systemctl enable vncserver@:1.service
- Allow VNC service in firewall.
    - firewall-cmd --permanent --add-service vnc-server
    - systemctl restart firewalld.service
- Start the service
    - systemctl start vncserver@:1.service
- And your VNC Server is ON
- To stop VNC service
    - systemctl stop vncserver@:1.service
- Disable VNC service from startup
    - systemctl disable vncserver@:1.service
- To stop firewall
    - systemctl stop firewalld.service

Screenshots of our OpenStack Dashboard

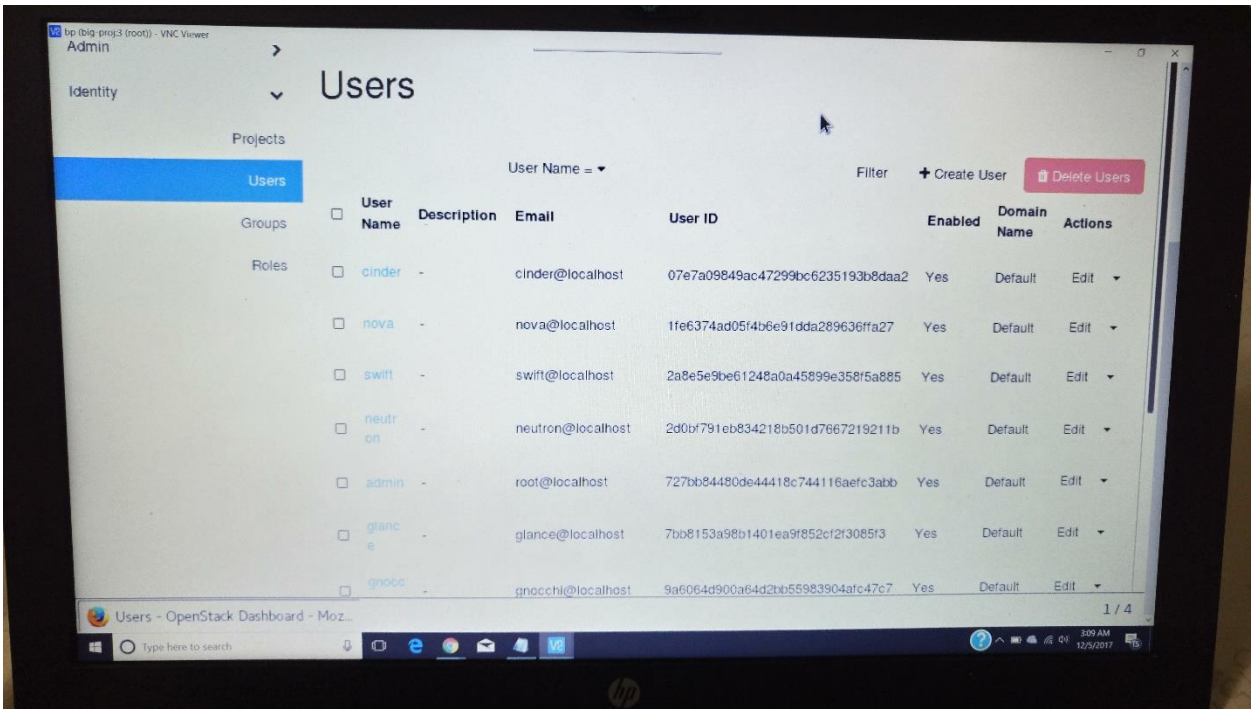
- **Dashboard Log in screen**



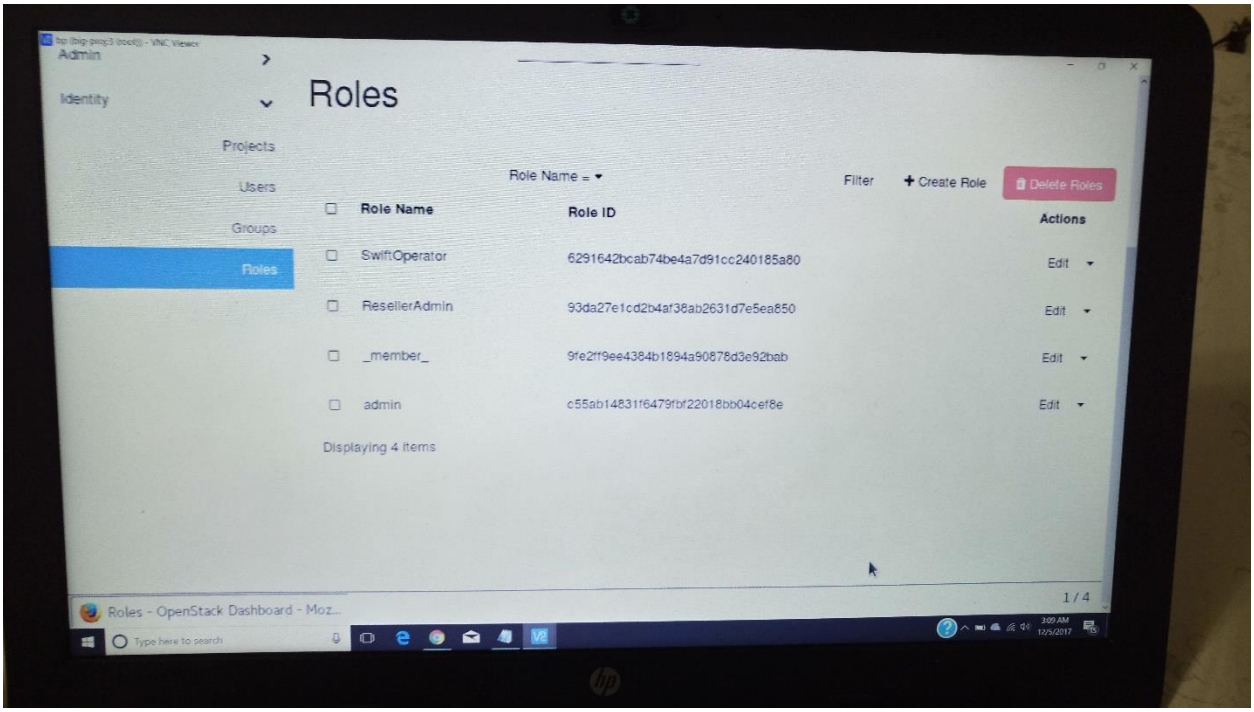
- **Projects**



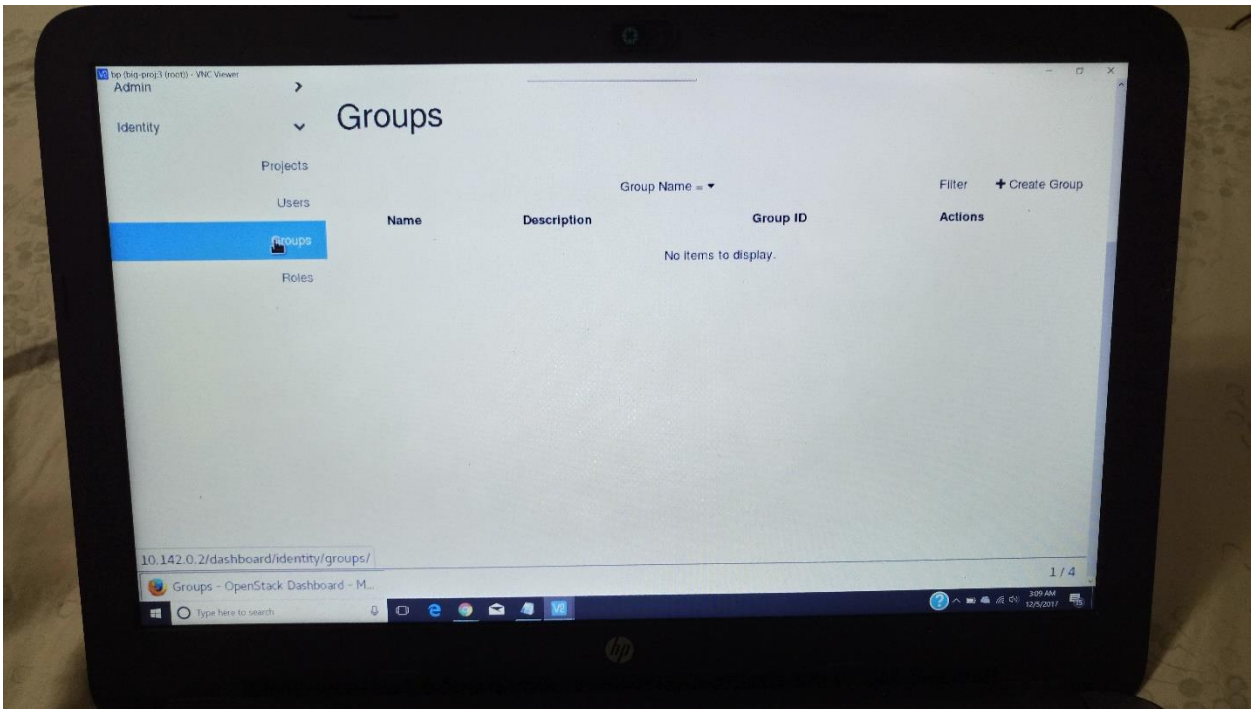
- **Users**



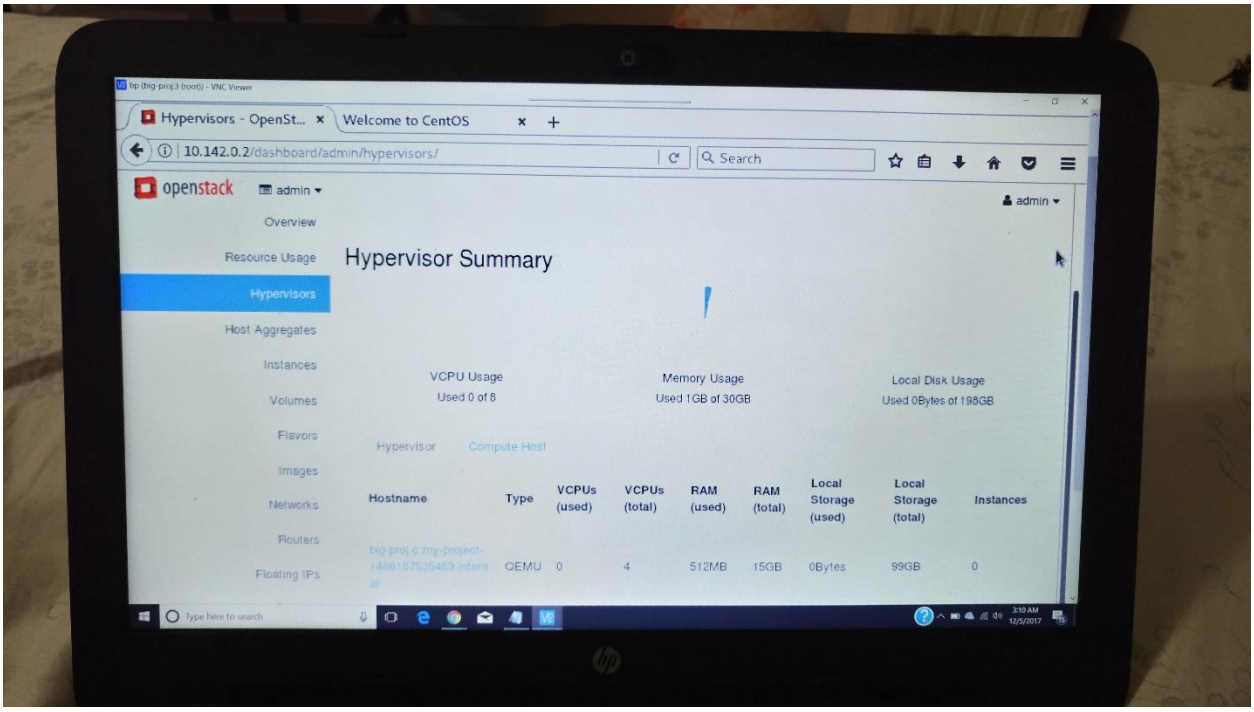
- **Roles**



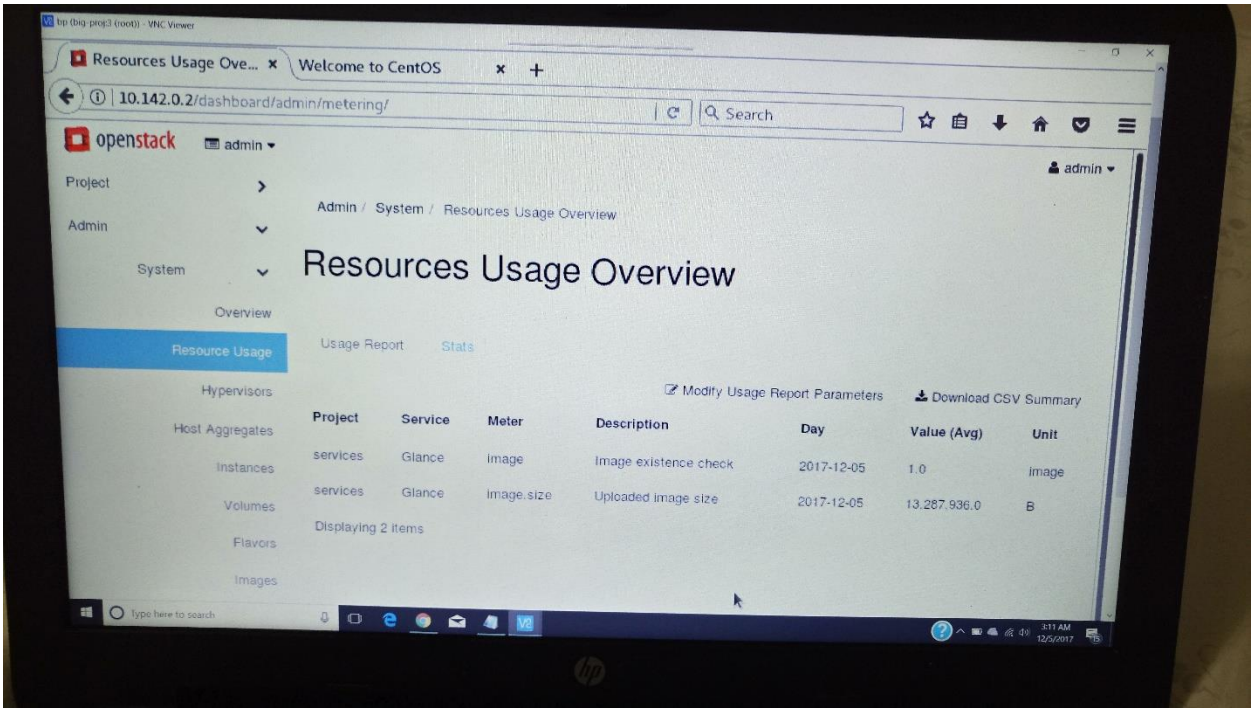
- **Groups**



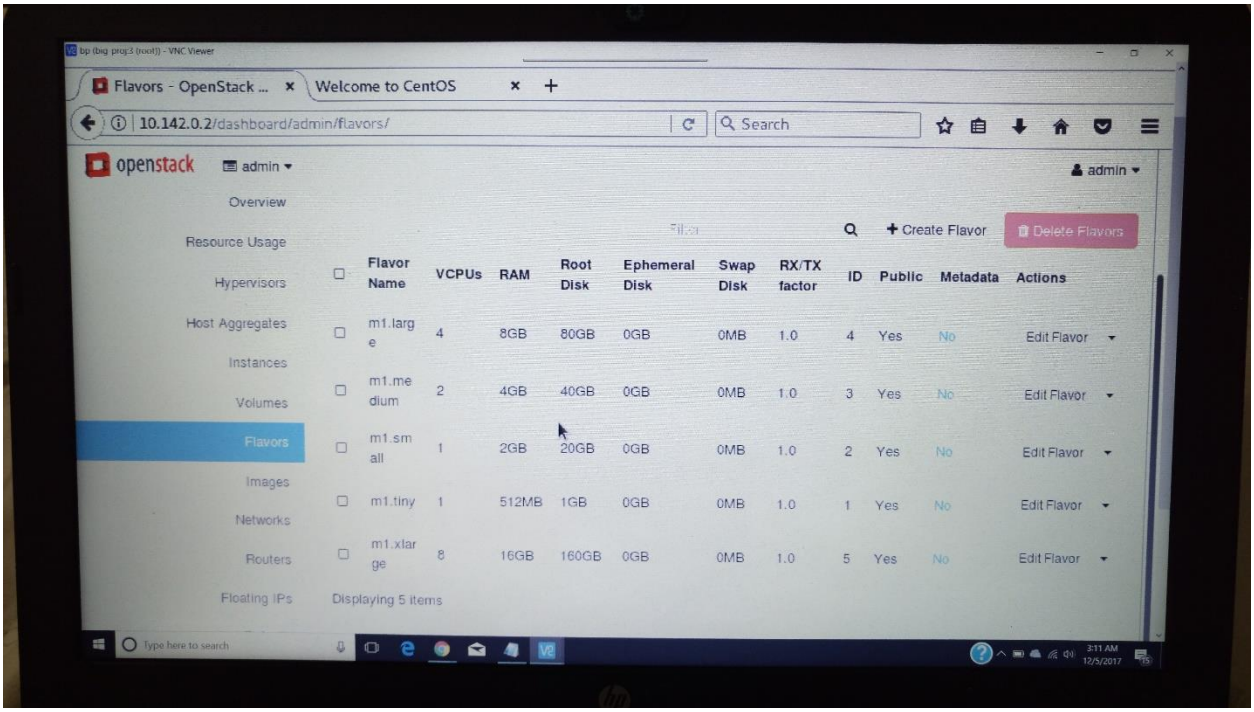
- **Hypervisors**



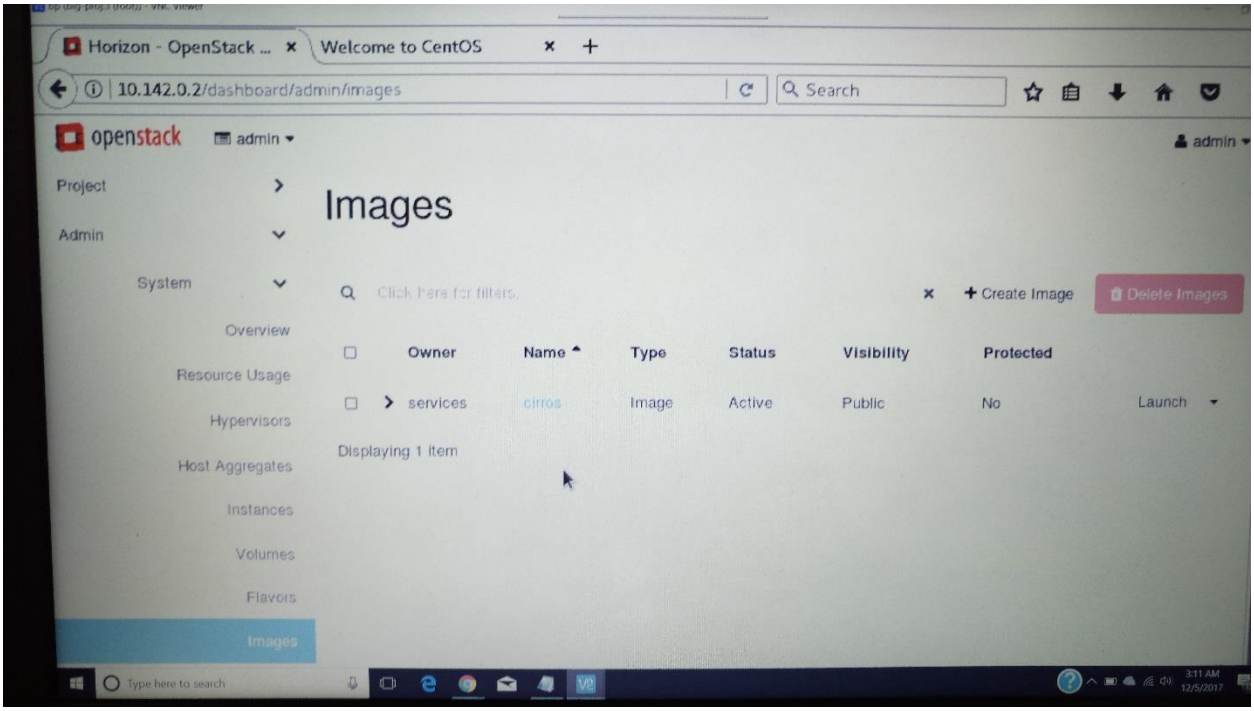
- **Resource Usage**



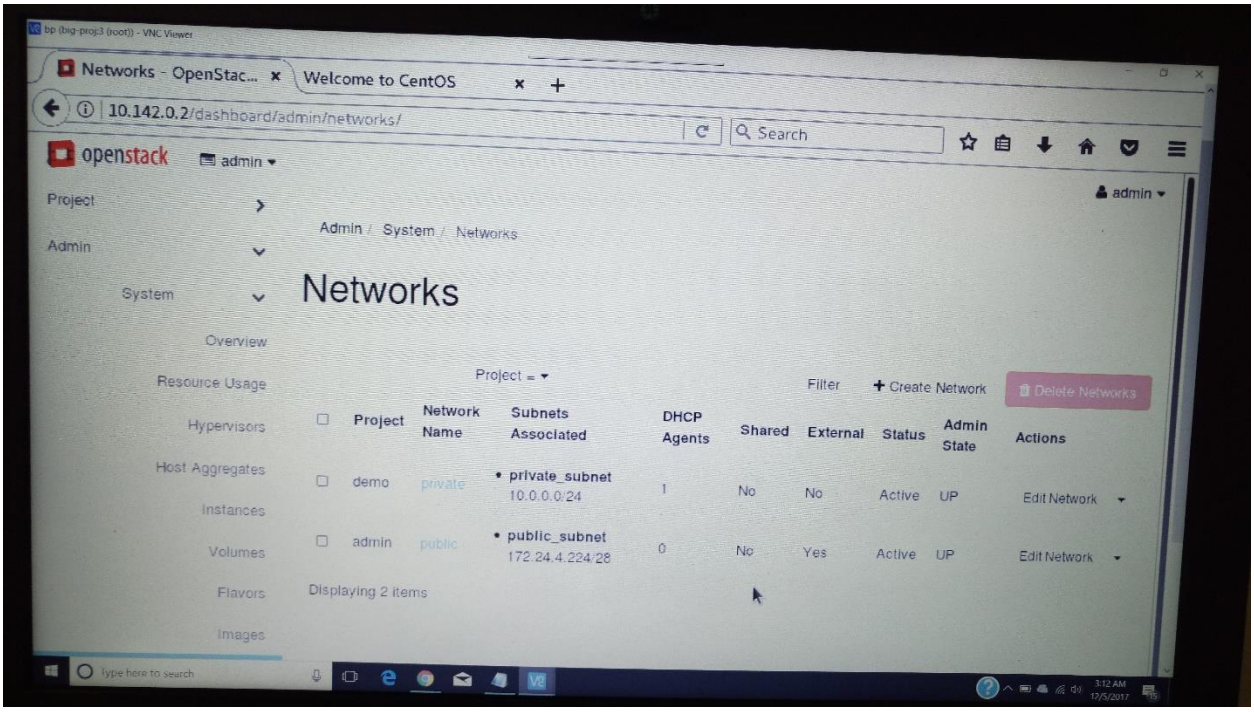
- **Flavors**



- **Images**



- **Networks**


Networks - OpenStac...
Welcome to CentOS
10.142.0.2/dashboard/admin/networks/
Search
openstack
admin
Project
Admin
System
Overview
Resource Usage
Hypervisors
Host Aggregates
Instances
Volumes
Flavors
Images
Admin / System / Networks
Networks
Project =
Filter
+ Create Network
Delete Networks
Project
Network Name
Subnets Associated
DHCP Agents
Shared
External
Status
Admin State
Actions
demo
private
private_subnet
10.0.0.0/24
1
No
No
Active
UP
Edit Network
admin
public
public_subnet
172.24.4.224/28
0
No
Yes
Active
UP
Edit Network
Displaying 2 items
Type here to search

PART 1

TROVE

Database as a Service

Databases in Traditional IT Systems:

- Provisioning by DBA
- Database management by Specialists
- Waterfall Development
- few large machines/bare metal

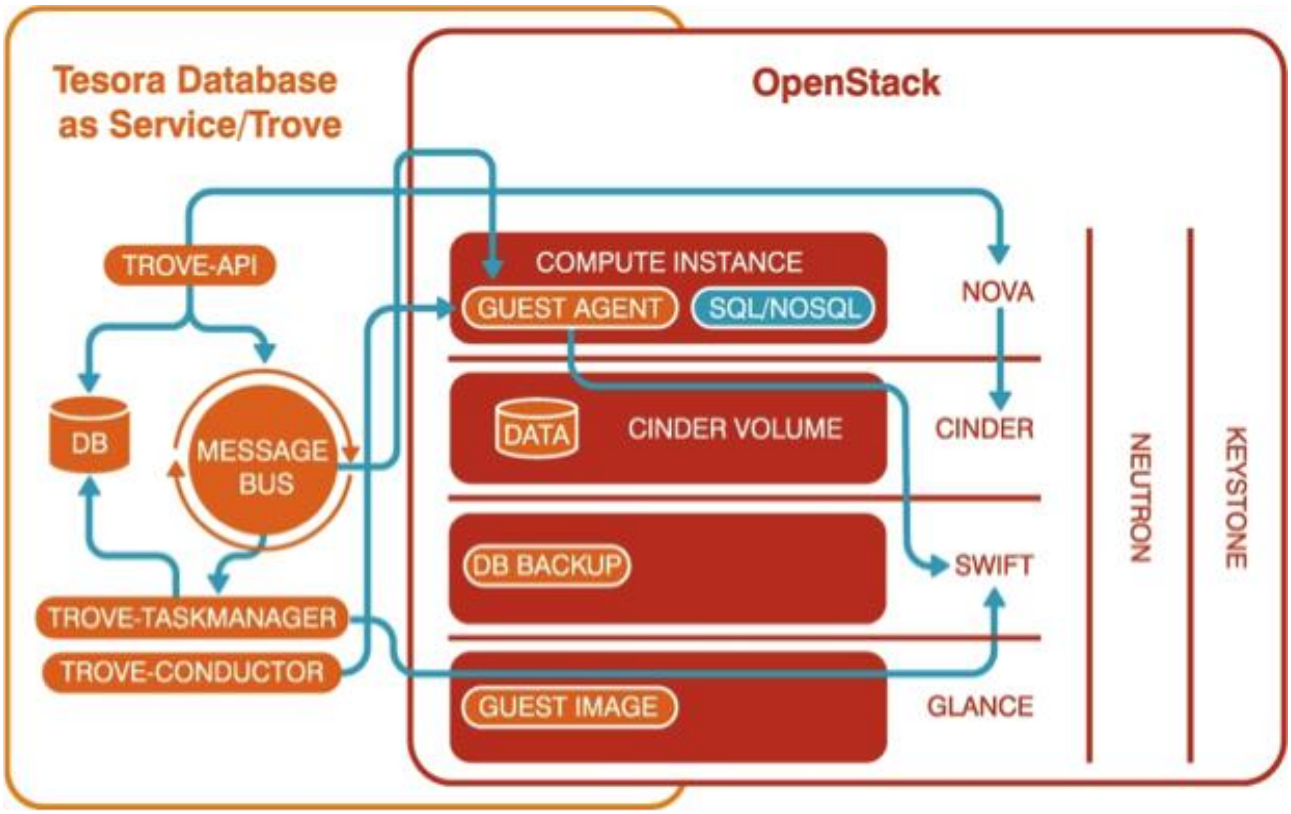
Databases on Cloud:

- Self-service provisioning
- Developers manage their own databases
- Agile Development
- many small machines/virtualization
- many data management technologies

TROVE Mission Statement:

To provide scalable and reliable Cloud Database as a Service provisioning functionality for both relational and non-relational database engines, and to continue to improve its fully-featured and extensible open source framework.

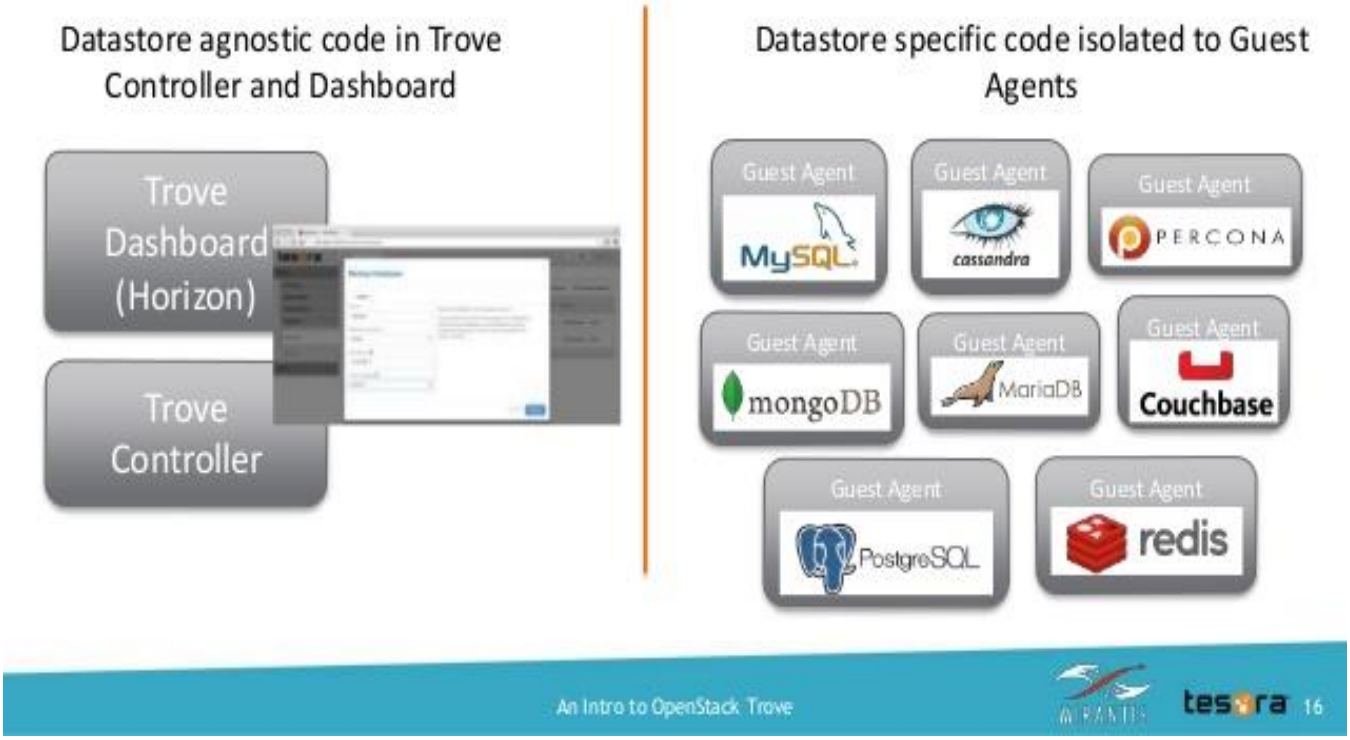
TROVE Architecture:



Source : http://www.odbms.org/2016/02/10-things-you-should-know-about-openstack-trove-the-open-source-database-as-a-service/

Why Trove?

- Main goal is to make it quick and easy to deploy and manage databases of all kinds
- It automates complex database administrative tasks including deployment, configuration, patching, backups, restores, and monitoring
- Single framework to operate 13 different DBMS technologies in a consistent way



Source: https://image.slidesharecdn.com/webinarintrotrovemirantis26feb2015-150226141250-conversion-gate01/95/openstack-and-trove-with-mirantis-and-tesora-16-638.jpg?cb=1424981648

- It creates a pluggable architecture where many different types of databases can be supported
- Automates the lifecycle management of the database instances it provisions
- Can be accessed through a web-based GUI, a command line interface, or a set of RESTful APIs

Trove Terminology:

- Trove Instance
- Guest Image
- Datastore
- Flavor
- Guest Agent
- Configuration Group

How to create a Database Instance using TROVE:

- To create a database instance called PS_troveinstance, with volume size 2 GB, user PS_user, password PS_password and MySQL datastore:
    - $ trove create –size 2 –users PS_user:PS_password –datastore MySQL PS_troveinstance
- To get the ID of the database instance:
    - $ trove list PS_troveinstance
- Finally, create database:
    - $ trove database-create PS_troveinstance PS_trovedb
- Alternatively, combining above commands as:
    - $ trove create –size 2 —-database PS_trovedb users PS_user:PS_password –datastore MySQL PS_troveinstance
- We now have a MySQL database server containing a database called PS_trovedb

PART 2

PUPPET

A Configuration Management Tool

Why do we need Configuration Management Tools?

The major tasks of System Administrator and people working in DevOps are as follows:

- Server Installation
- Server Configuration
- Modification/Change server configuration
- Maintenance of the servers

Each of the above tasks is very tedious. It takes a lot of time as well as efforts to complete server installation, configuration and maintenance. Also, there are huge chances of human errors in above activities which eventually can lead to entire system failure.
When the number of servers increase to thousands or more than that then it becomes a very hectic job.

To overcome all the above issues, we require Configuration Management Tools.

What is Configuration Management?

Configuration Management is a practice of handling changes systematically so that a system maintains its integrity over the time. It also allows access to an accurate historical record of system state.

What is Puppet?

Puppet is a configuration management tool that is used to deploy, configure and manage the server machines in a controlled manner.

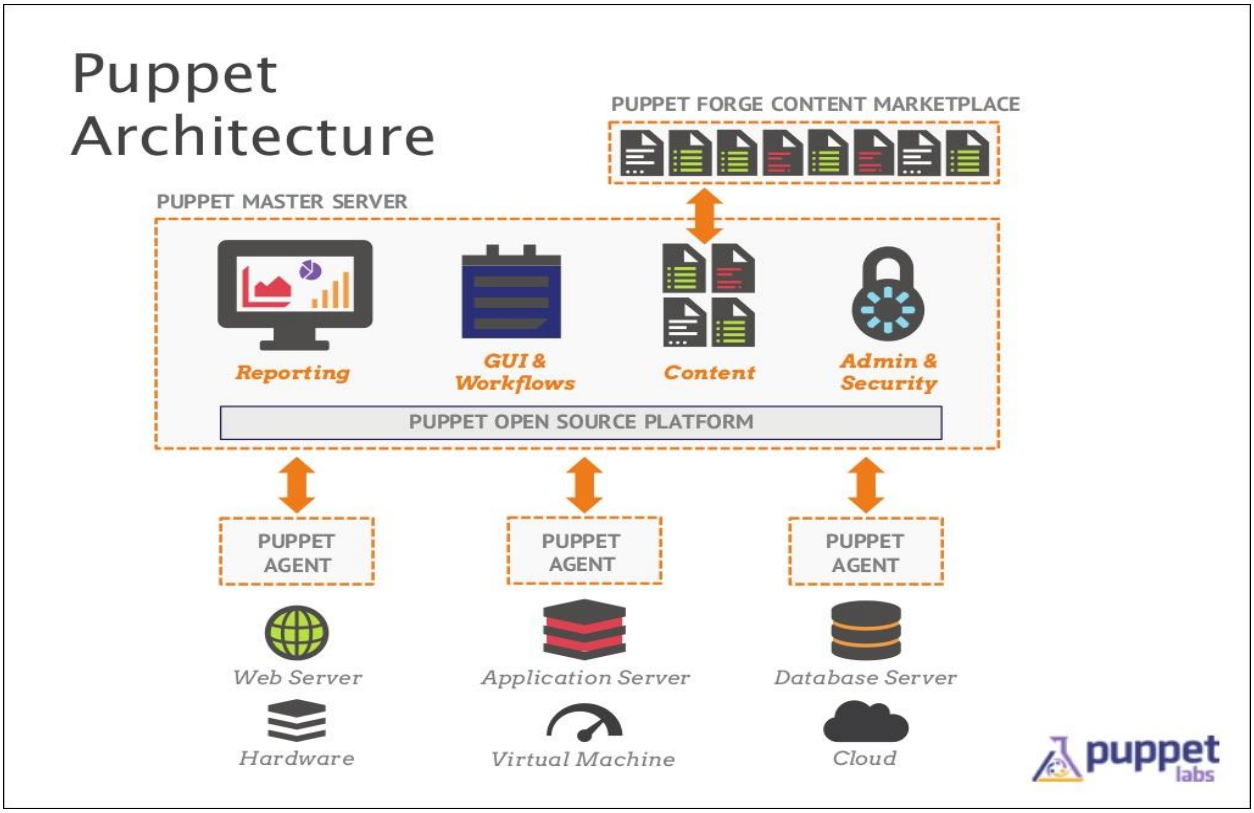
- Puppet is a very powerful tool which helps in the concept of Infrastructure as code
- Written in Ruby DSL language that helps in converting a complete infrastructure in code format, which can be easily managed and configured.

Features of Puppet

- Puppet provides a way to automatically configure newly provisioned systems
- Controls all the steps, right from bootstrapping to the end of the server life
- Can group the servers according to roles
  Example: Web servers, Database Servers
- Used as either a local system command line tool or in a client-server relationship
- Server acts as a Puppet Master and applies configuration to multiple client system using Puppet Agent
- Domain specific language is loosely modeled on Ruby and works on Unix/Linux and Windows
- Maintain consistency across nodes
  Example: If a change is done locally then it is rolled back to the original configuration

Puppet Architecture

- Puppet usually follows client-server architecture. The client is known as Agent and the server is known as master.
- It can also be used as a stand-alone application.



Source : https://www.slideshare.net/joshbeard/puppet-overview-28908346

Puppet Components

Puppet Master

- Puppet master is a service that runs on the main server which is used to manage the entire clients
- The master machine contains all the configuration for different hosts

Puppet Agent

- It's a service which runs on each node and talks to the master
- The connection between master and agent is made in a secure encrypted channel with the help of SSL

Config Repository

- This is the repo where all nodes and server-related configurations are saved and pulled when required.

Facts

- Facts are the details related to the node or the master machine, which are basically used for analyzing the current status of any node.
- On the basis of facts, changes are done on any target machine.
- There are pre-defined and custom facts in Puppet.

Catalog

- All the manifest files or configuration which are written in Puppet are first converted to a compiled format called catalog and later those catalogs are applied on the target machine.
- Describes the desired states of the particular client/node
- Stores all the resources to be managed

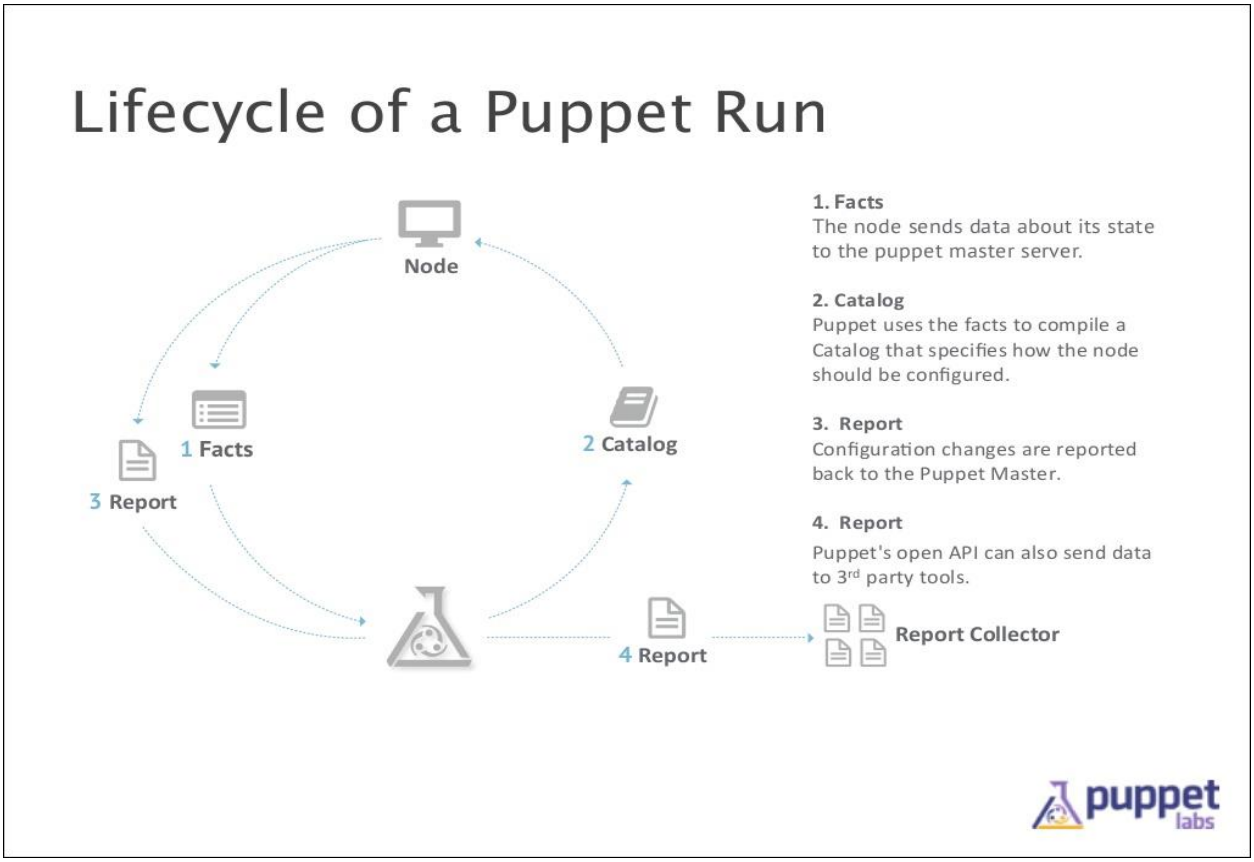
Manifests

- Manifest is a collection of resources which are coupled inside the function or classes to configure any target system.
- Resources can be files, packages, services, etc.
- They contain a set of Ruby code in order to configure a system.

Modules

- Module is the key building block of Puppet, which can be defined as a collection of resources, files, templates, etc.
- They can be easily distributed among different kinds of OS being defined that they are of the same flavor. As they can be easily distributed, one module can be used multiple times with the same configuration.
- Useful for organizing Puppet code

How Puppet Works?



Source : https://www.slideshare.net/joshbeard/puppet-overview-28908346

The above figure describes the Puppet master/agent topology in more detail.

- Puppet provides the ability to define which software and configuration a system requires and then maintain a specified state after an initial setup
- You use a declarative Domain Specific Language (DSL) that is similar to Ruby to define configuration parameters for a specific environment or infrastructure.
- Puppet discovers information about a system by using a utility called Facter, which is installed when you install the Puppet software package.
- The nodes that the master controls are those that have Puppet installed on them and are running the Puppet agent, which is a daemon.
- The configuration information that the agent collects about a node is sent to the Puppet master.
- The Puppet master then complies a catalog based on how the node should be configured.
- Each node uses that information to apply any necessary configuration updates to itself.

- Then the changes are reported back to master. Puppet master may send these reports to third party tools.
- Puppet works by using a pull mode, where agents poll the master at regular intervals to retrieve site-specific and node-specific configurations.

Steps to install Puppet on Ubuntu 16.04

Step 1: Environment Setup and configuration

On each machine (master and agents), edit the */etc/hosts* file. At the end of the file, specify the Puppet master server as follows, substituting the IP address for *your* Puppet master:

- *sudo nano /etc/hosts*

Add below in hosts file

- *puppet_ip_address puppet*

Install NTP
Time must be set accurately on puppet master that will be acting as a certificate authority to sign the certificates coming from the client nodes. We will use NTP for this purpose.

- *sudo apt-get install ntp ntpdate*
- *sudo ntpdate -u 0.ubuntu.pool.ntp.org*

Ensure that all the nodes are in same time zone using ***date*** command. If there are any discrepancies, change it accordingly.

Step 2 — Installing Puppet Server

Enable the official Puppet Labs collection repository with these commands

- *curl -O* https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb
- *sudo dpkg -i puppetlabs-release-pc1-xenial.deb*
- *sudo apt-get update*

Then, install the puppet server:

- *sudo apt-get install puppetserver*

Start Puppet server

- *sudo systemctl start puppetserver*

Step 3 — Installing the Puppet Agent

Enable the official Puppet Labs repository

- *wget https://apt.puppetlabs.com/puppetlabs-release-pc11-xenial.deb*
- *sudo dpkg -i puppetlabs-release-pc1-xenial.deb*
- *sudo apt-get update*

Install the Puppet agent package

- *sudo apt-get install puppet-agent*

Start Puppet

- *sudo systemctl start puppet*
- *sudo systemctl enable puppet*

Step 4 — Signing Certificates on Puppet Master

The first time Puppet runs on an agent node, it sends a certificate signing request to the Puppet master. Before Puppet Server will be able to communicate with and control the agent node, it must sign that particular agent node's certificate.

List current certificate requests

To list all unsigned certificate requests, run the following command on the Puppet master:

- *sudo /opt/puppetlabs/bin/puppet cert list*

There should be one request for each host you set up, that looks something like the following:

(SHA256)46:19:79:3F:70:19:0A:FB:DA:3D:C8:74:47:EF:C8:B0:05:8A:06:50:2B:40:B3:B9:26:35:F6:96:17:85:5E:7C

A + in front of a certificate indicates it has been signed. The absence of a plus sign indicates our new certificate has not been signed yet.

Sign requests

To sign a single certificate request, use the puppet cert sign command, with the hostname of the certificate as it is displayed in the certificate request.

- *sudo /opt/puppetlabs/bin/puppet cert sign hostname.localdomain*

Use the --all option to sign the remaining certificate:

- *sudo /opt/puppetlabs/bin/puppet cert sign –all*

Step 5- Verification

Once the Puppet master is signed your client certificate, run the following command on the client machine to test it.

sudo /opt/puppetlabs/bin/puppet agent –test

Conclusion about Puppet

- Puppet is a very powerful Configuration Tool and widely used for automation
- Makes the tasks of System Admin very easy
- Improves efficiency
- Reduces problems caused by human error

References:

OpenStack Installation:

1. https://www.youtube.com/watch?v=ozUvALQw5T8
2. http://www.itzgeek.com/how-tos/linux/centos-how-tos/install-gnome-gui-on-centos-7-rhel-7.html
3. http://www.krizna.com/centos/install-vnc-server-centos-7/

TROVE Installation:

1. https://wiki.openstack.org/wiki/Trove/installation
2. https://docs.openstack.org/mitaka/install-guide-obs/trove-install.html
3. https://dzone.com/articles/openstack-trove-benefits

PUPPET Installation:

1. https://www.youtube.com/watch?v=0yVJhb2VkVk
2. https://www.youtube.com/watch?v=eQe-Jh6eoMg
3. https://www.digitalocean.com/community/tutorials/how-to-install-puppet-4-on-ubuntu-16-04
4. https://www.youtube.com/watch?v=vOLaMDyYtcI