OPENSTACK

What is Openstack?

- Openstack is a collection of open source technology delivering a massively scalable cloud operating system.
- OpenStack is a software package that provides a cloud platform for Public and Private cloud covering various use cases including Enterprise and Telecom.
- The main focus is on Infrastructure as a Service (IaaS) cloud and additional services built upon IaaS.
- It is a set of software that allows you to build your own cloud infrastructure and allows you to launch your own cloud infrastructure on which you can run your own application like development testing and deployment.
- Openstack releasing two versions every years.
- There are number of companies which contributing to the openstack: rackspace, cisco, IBM, redhat, vmware, dell, hp.
- Rackspace is the one who created the openstack in 2010 with NASA.
- Openstack is the future of cloud computing backed by some of biggest company.
- The openstack is managed by openstack foundation.

Advantages of an Openstack

I. Strong security

✓ One of the main reasons why OpenStack is so popular in the world of cloud computing is – it has outstanding security features that keep you secure all the time.

II. Open-source

✓ OpenStack is open-source that makes it the favourite cloud software for the developers and entrepreneurs. The source code of this project can be found at www.github.com/openstack/.

III. Development support

✓ OpenStack has been receiving a concrete development support from many prestigious companies and from the top developers of the IT industry for many years.

IV. An array of services for different tasks

✓ OpenStack provides you with an array of services that accomplish different necessary tasks for your private, public and hybrid cloud.

✓ OpenStack using a dashboard, Mistral for managing workflows, Sahara for providing Hadoop clusters, and Zaqar is for messaging on the cloud.

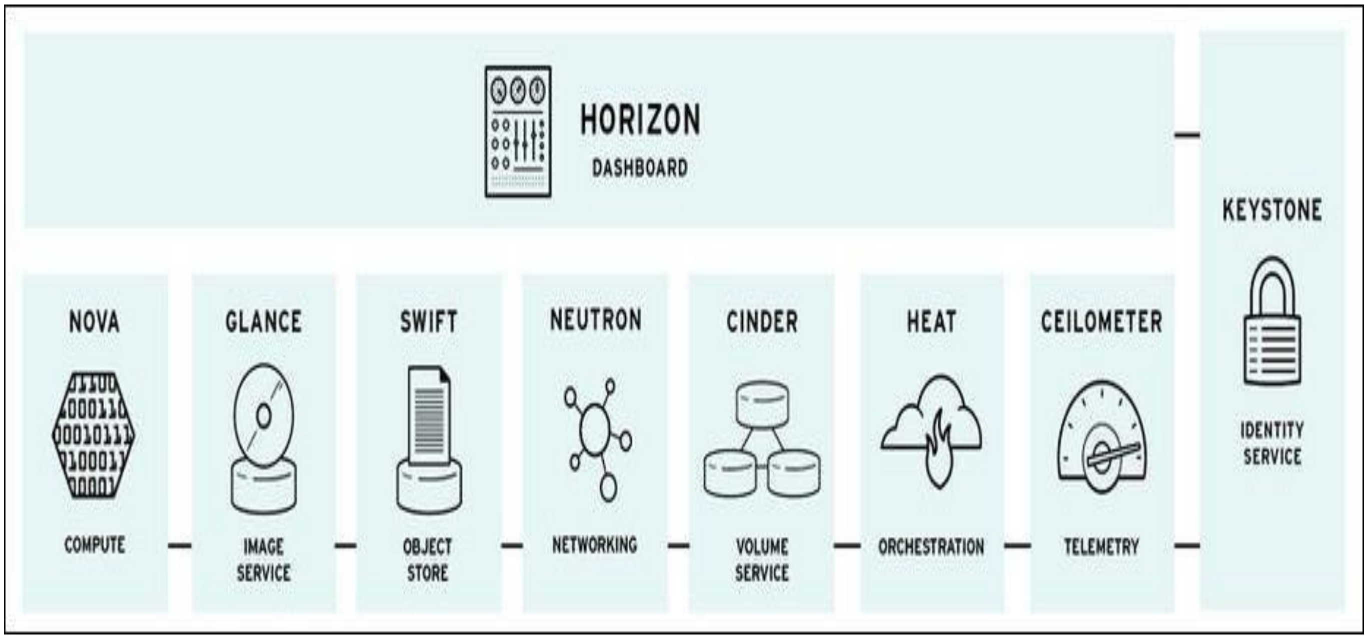
V. Easy to access and manage OpenStack

✓ OpenStack can be accessed and managed in several ways, which is a great benefit for you.

- GUI-based dashboard
- Command line tools
- APIs for programmers.

COMPONENTS OF AN OPENSTACK:

- OpenStack embraces a modular architecture to provide a set of core services that facilitates scalability and elasticity as core design tenets.
- This chapter briefly reviews OpenStack components, their use cases and security considerations.



i. **Nova**
   - This is the primary computing engine behind OpenStack.
   - This allows deploying and managing virtual machines and other instances to handle computing tasks.

ii. **Swift**
   - The storage system for objects and files is referred to as Swift.
   - In the traditional storage systems, files are referred to a location on the disk drive, whereas in OpenStack Swift files are referred to by a unique identifier and the Swift is in charge where to store the files.
   - The scaling is therefore made easier because the developers don’t have the worry about the capacity on a single system behind the software.
   - This makes the system in charge of the best way to make data backup in case of network or hardware problems.

iii. **Cinder**

- This is the respective component to the traditional computer access to specific disc locations.
- It is a block storage component that enables the cloud system to access data with higher speed in situations when it is an important feature.

iv. **Neutron**

- Neutron is the networking component of OpenStack.
- It makes all the components communicate with each other smoothly, quickly and efficiently.

v. **Horizon**

- In Horizon you can login with different user.
- This is the OpenStack dashboard. It’s the graphical interface to OpenStack and the first component that users starting with OpenStack will see.
- There is an OpenStack API that allows developers to access all the components individually, but the dashboard is the management platform for the system administrators to have a know what is going on in the cloud.

vi. **Keystone**

- This is the identity service who actually maintain who is using what kind of things?
- This is the component that provides identity services for OpenStack.
- Basically, this is a centralized list of all the users and their permissions for the services they use in the OpenStack cloud.

vii. **Glance**

- It is a component that provides image services or virtual copies of the hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances.

viii. **Ceilometer**

- Ceilometer provides data measurement services, thus enabling the cloud to offer billing services to individual users of the cloud.
- It measures system usage by each user for each of the components of the cloud and makes reporting available.

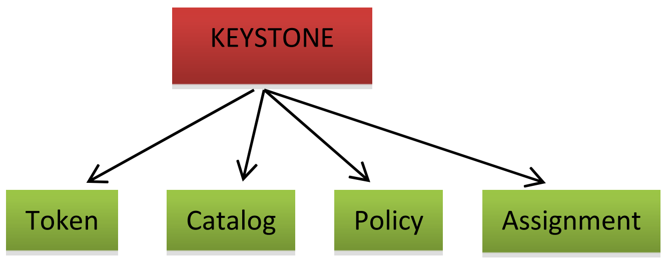
ix. **Heat**

- Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application.

- In this way, it helps to manage the infrastructure needed for a cloud service to run.

KEYSTONE:

- This is the authentication service used for authentication and authorization of an openstack. Every service that want to avail is first get to authenticate first.
- Keystone is an openstack service that provide API client authentication , service discovery and distributed multi-tenent authentication by implementing openstack identity API.
- It keep track of who and what is using.
- Keystone have four components.
    a. Token Service
    b. Catalog Service
    c. Policy service
    d. Assignment service.



i **Token Service:**
- It validate and manages token to authenticate the request once users credential already been verified.
- It is session based management service that token been uses.

ii **Catalog Services:**
- It provides you with endpoint registry which is used for endpoint discovery.
- Openstack is connected to the every other services with endpoints.

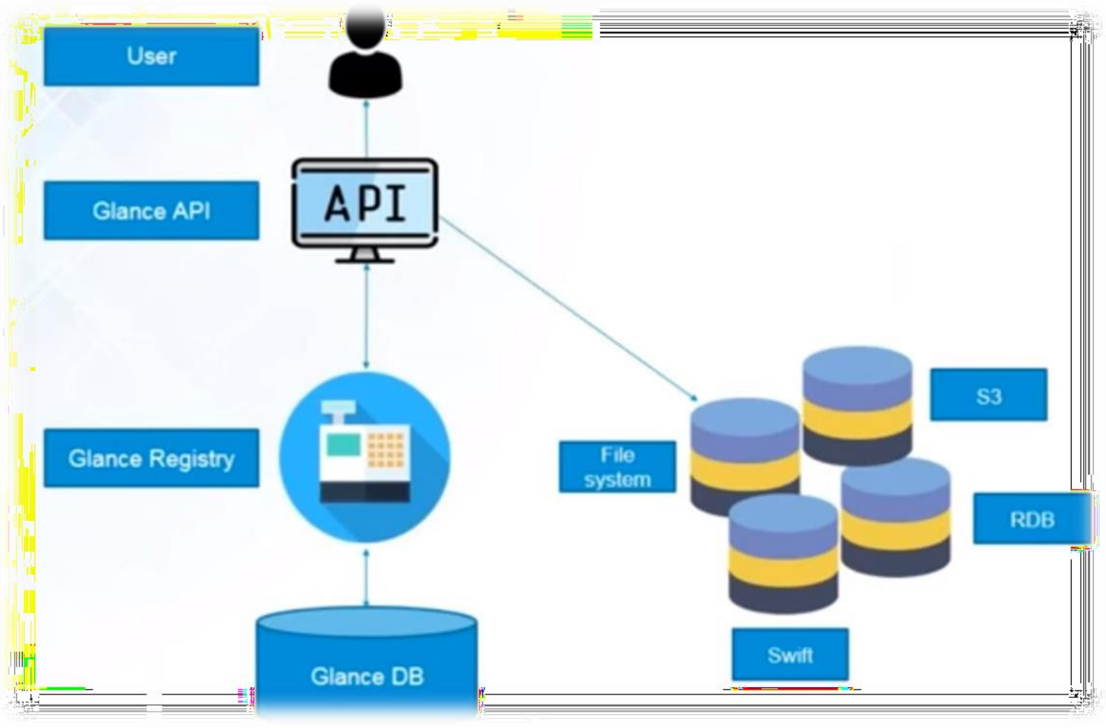
iii **Policy Service:**
- It provides rule based authorization user and it is associated with rule management interface.
- There are some policy mentioned in rule management so this policy service will check whether user authorized to do that or not.

iv **Assignment Service:**
- It provides data about roles.
- Roles are the levels of Authorization that end-user can operate.

GLANCE:

- The Glance project provide a service where users can upload and discover data assets that are meant to be used with other services.
- This currently includes image and metadata definitions.
- Glance image service includes discovering, registering and retrieving virtual machine image.
- Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.
- The glance is a repository for images , the glance store all the virtual images that you have added into openstack and from there on these images which stores on glance you can lunch the instances from there.
- And these instances you can used set up your develop environment of your application and for testing environment.

GLANCE ARCHITETURE:



- There are three pieces to Glance architecture: *glance-api*, *glance-registry*, and the image store.

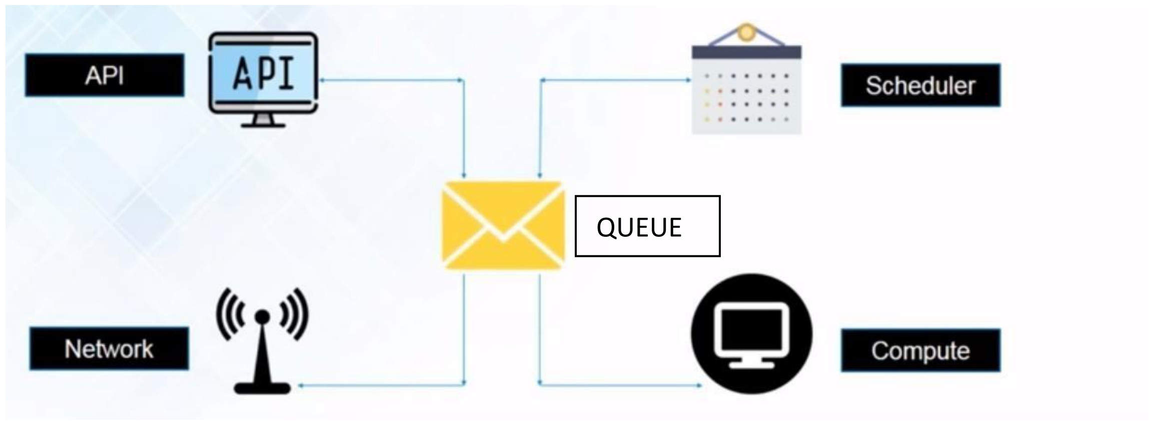
- As you can probably guess, *glance-api* accepts API calls for image retrieval, storage and discovery.
- The *glance-registry* stores process and retrieves metadata about images metadata includes the items such as size and type.
- The image store can be a number of different object stores, including Swift. Figure illustrates Glance’s logical architecture.
- *glance-api* is similar in functionality to *nova-api*, in that it accepts incoming API requests and then communicates with the other components (*glance-registry* and the image store) to facilitate querying, retrieving, uploading, or deleting images. By default, *glance-api* listens on port 9292.
- In openstack every service communicate with APIs and here glance also uses API to connect with user and other services as well.
- In above figure user using glance and it connect through Glance API and it will goes to the Glance Registry which is connected to the glance database.
- The Glance database stores all the metadata and definitions.
- And API also connected with the different file system, object storage service, swift,s3, RDB.
- These storage system actually stores the image in glance.
- So the glance registry contains each entry of images and other images that have been added into the glance architecture.
- The metadata is stored into the glance database.
- The images is stored in the swift,s3,RDB, file system and from there on if you want to lunch the images you can lunch it using glance.

NOVA(Compute Node)

- Openstack Nova is a component within the openstack open source cloud computing platform to provide on-demand access to compute resources by provisioning and managing large networks of virtual machines(VMs).
- Nova works with VMWare, Xen,KVM,Hype-V and other virtualization technology.
- Nova is a openstack project design to provide power to massive, scalable, on demand self service access to compute resources.
- It is fault tolerance , recoverable, and provide the API compatibility with system like AWS EC2.
- It is build on massaging architecture and all of its component typically run on several server.

- The nova is a compute zone so this is where your instances get launch and this is where all the computing and processing takes place.
- In NOVA All the components communicate through massage queue.

ARCHITECTURE



1. NOVA API

- The nova-api daemon is the heart of Nova. You may see it illustrated on many pictures of Nova as API and "Cloud Controller."
- Its primary purpose is to accept and fulfill incoming API requests.
- To accept and fulfill API requests, *nova-api* provides an endpoint for all API queries (accepting requests using either the OpenStack API or the Amazon EC2 API), initiates most of the orchestration activities (such as running an instance), and also enforces some policy (mostly quota checks).
- For some requests, it will fulfill the entire request itself by querying the database and then returning the answer.
- For more complicated requests, it will pass messages to other daemons through a combination of writing information to the database and adding messages to the queue.
- By default, *nova-api* listens on port 8773 for the EC2 API and 8774 for the OpenStack API.

2. Scheduler

- The nova-scheduler process is conceptually the simplest piece of code in Nova: it takes a virtual machine instance request from the queue and determines where it should run (specifically, which compute server host it should run on).
- In practice, however, this will grow to be the most complex piece, as it needs to factor in the current state of the entire cloud infrastructure and apply complicated algorithms to ensure efficient usage.
- To that end, nova-scheduler implements a pluggable architecture that lets you choose (or write) your own algorithm for scheduling.

Table 4-1 details the current scheduler choices.

Scheduler	**Notes**
Simple	Attempts to find least loaded host.
Chance	Choses random available host from service table. This is the default scheduler.
Zone	Picks random host from within an availability zone.

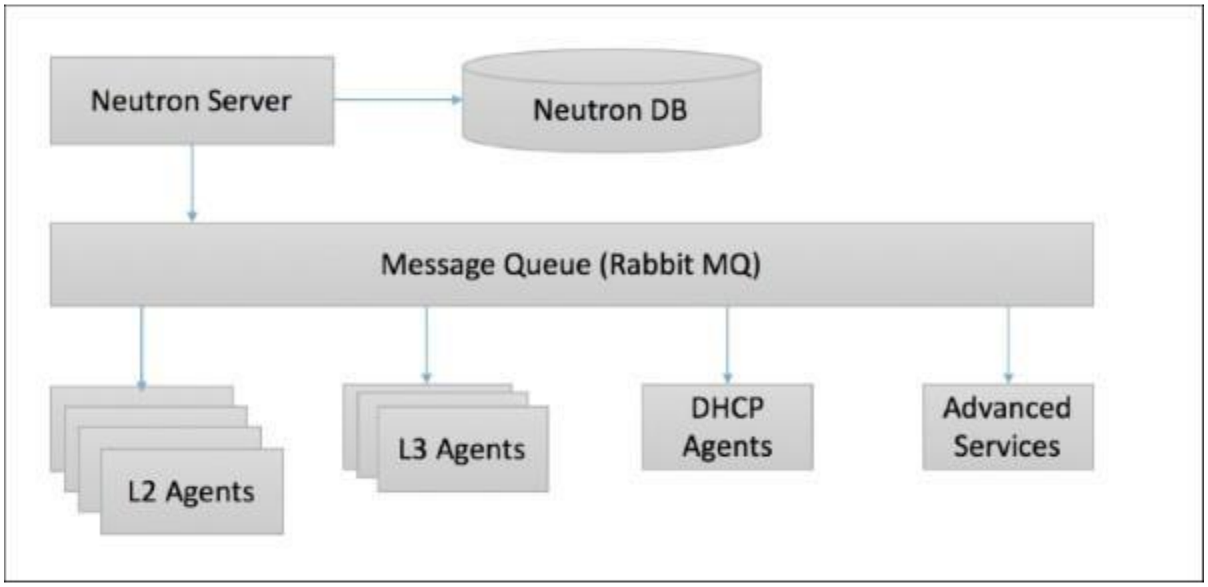
3. **Compute Worker**
   - The nova-compute process is primarily a worker daemon that creates and terminates virtual machine instances.
   - The process by which it does so is fairly complex, but the basics are simple: accept actions from the queue and then perform one or a series of virtual machine API calls to carry them out while updating state in the database.
   - An example of this would be nova-compute accepting a message from the queue to create a new instance and then using the libvirt library to start a new KVM instance.
   - There are a variety of ways that nova-compute manages virtual machines. The most common is through a software package called libvirt.
   - This is a toolkit (API, daemon, and utilities) created by Red Hat to interact with the capabilities of a wide range of Linux virtualization technologies.
   - While libvirt may be the most common, nova-compute also uses the Xen API, vSphere API, Windows Management Interface, and others to support other virtualization technologies.
4. **Queue**

- The queue provides a central hub for passing messages between daemons. This is currently implemented with RabbitMQ today, but theoretically could be any AMPQ message queue supported by the Python ampqlib and carrot libraries.
- Nova creates several types of message queues to facilitate communication between the various daemons.
- These include: topics queues, fanout queues, and host queues.
- Topics queues allow messages to be broadcast to the number of particular class of worker daemons.
- For example, Nova uses these to pass messages to all (or any) of the compute or volume daemons.
- This allows Nova to use the first available worker to process the message. Host queues allow Nova to send messages to specific services on specific hosts.
- For example, Nova often needs to send a message to a specific host's compute worker to take action on a particular instance.
- Fanout queues are only currently used for the advertising of the service capabilities to nova-scheduler workers.

5. **Database**
   - The database stores most of the configuration and run-time state for a cloud infrastructure.
   - This includes the instance types that are available for use, instances in use, networks available, and projects.

NEUTRON

- The architecture of Neutron is simple, but it is with the agents and plugins where the real magic happens!
- Neutron is a networking project focused on delivering Networking as-a-service (NaaS) in a virtual compute environment.
- Neutron has replaced the original networking application program interface (API) called Quantum in openstack.
- Neutron is designed by address deficiencies in backed-in networking technology found in cloud environment.
- Neutron architecture has been presented in the following diagram:



Let's discuss the role of the different components in a little detail.

a. The Neutron server

❖ The function of this component is to be the face of the entire Neutron environment to the outside world. It essentially is made up of three modules:

- **REST service**: The REST service accepts API requests from the other components and exposes all the internal working details in terms of networks, subnets, ports, and so on.
- **RPC service**: The RPC service communicates with the messaging bus and its function is to enable a bidirectional agent communication.
- **Plugin**: A plugin is best described as a collection of Python modules that implement a standard interface, which accepts and receives some standard API calls and then connects with devices downstream. They can be simple plugins or can implement drivers for multiple classes of devices.

❖ The plugins are further divided into core plugins, which implement the core Neutron API, which is Layer 2 networking (switching) and IP address management. If any plugin provides additional network services, we call it the service plugin -- for example, Load Balancing as a Service (LBaaS), Firewall as a Service (FWaaS), and so on.

❖ As an example, Modular Layer 2 (ML2) is a plugin framework that implements drivers and can perform the same function across ML2 networking technologies commonly used in datacenters. We will use ML2 in our installation to work with Open vSwitch (OVS).

b. L2 agent

- ❖ The L2 agent runs on the hypervisor (compute nodes), and its function is simply to wire new devices, which means it provides connections to new servers in appropriate network segments and also provides notifications when a device is attached or removed. In our install, we will use the OVS agent.

c. L3 agent

- ❖ The L3 agents run on the network node and are responsible for static routing, IP forwarding, and other L3 features, such as DHCP.

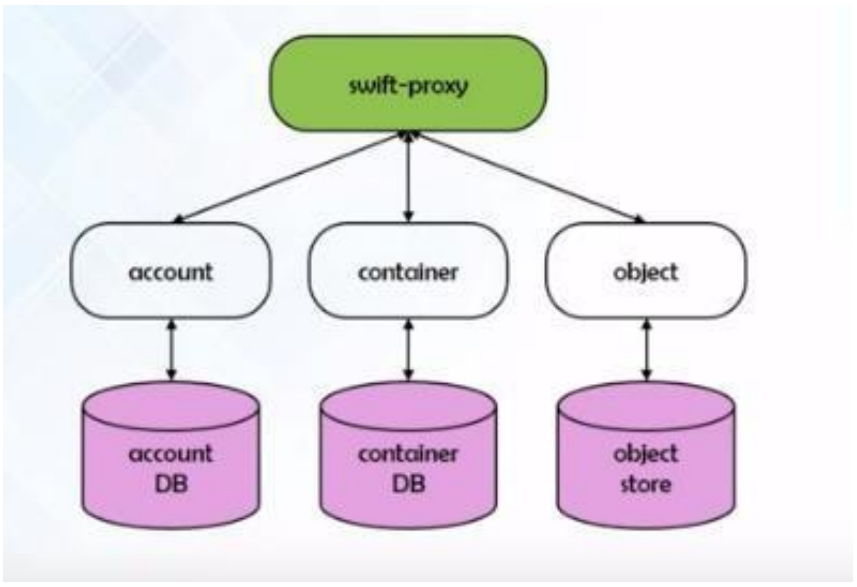
d. DHCP agent (*neutron-dhcp-agent*)

- ❖ Provides DHCP services to tenant networks.
- ❖ This agent is the same across all plug-ins and is responsible for maintaining DHCP configuration.
- ❖ The *neutron-dhcp-agent* requires message queue access. *Optional depending on plug-in.*

SWIFT

- It is an object storage in openstack you can store any kind of file, image ,data in swift. In swift anything you store is convertd into object and then it is stored inside the database.
- Swift is highly available , distributed and consistent object store.
- Organization can use swift to store lots of data efficiently, safely and cheaply.
- Swift powers storage clouds at Comcat, Time Warner, Globo and Wikipedia as well as public cloud like Rackspace, NIT OVH, and IBM SoftLayer.
- It is very easy to use data from the swift.

Architecture



i. **Proxy**

- Handle all of the incoming API requests.
- The Proxy server processes are the public face of Swift as they are the only ones that communicate with external clients.
- As a result they are the first and last to handle an API request.
- All requests to and responses from the proxy use standard HTTP verbs and response codes.
- Proxy servers use a shared-nothing architecture and can be scaled as needed based on projected workloads.
- A minimum of two proxy servers should be deployed for redundancy.
- Should one proxy server fail, the others will take over.
- For example, if a valid request is sent to Swift then the proxy server will verify the request, determine the correct storage nodes responsible for the data (based on a hash of the object name) and send the request to those servers concurrently.
- If one of the primary storage nodes is unavailable, the proxy will choose an appropriate hand-off node to send the request to.
- The nodes will return a response and the proxy will in turn return the first response (and data if it was requested) to the requester.
- The proxy server process is looking up multiple locations because Swift provides data durability by writing multiple–typically three complete copies of the data and storing them in the system.

ii. Account

- The account server process handles requests regarding metadata for the individual accounts or the list of the containers within each account.
- This information is stored by the account server process in SQLite databases on disk.

iii. Container

- The container server process handles requests regarding container metadata or the list of objects within each container.
- It’s important to note that the list of objects doesn’t contain information about the location of the object, simply that it belong to a specific container.
- Like accounts, the container information is stored as SQLite databases.

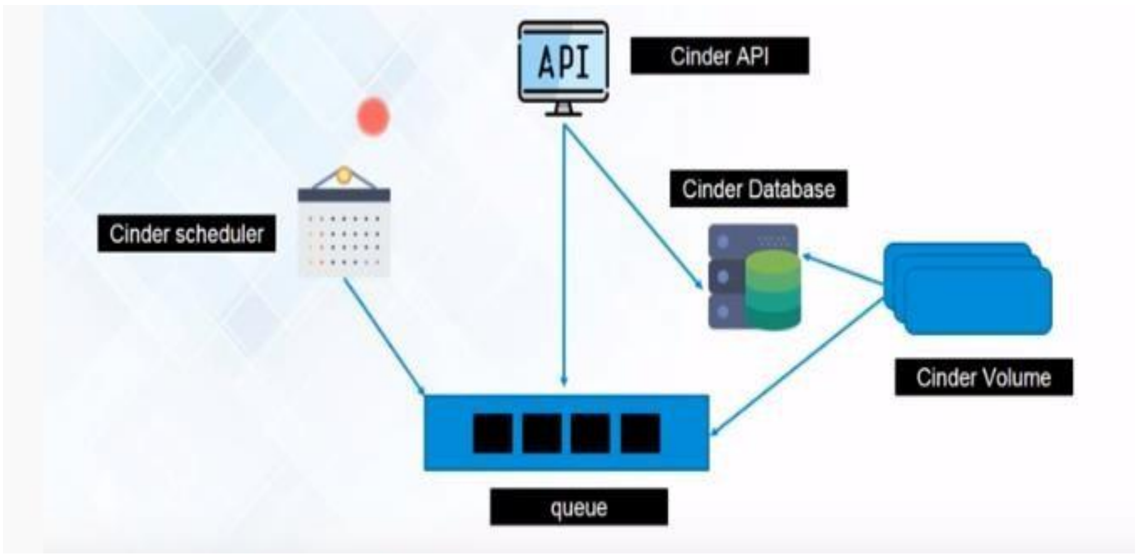
iv. Object

- The object server process is responsible for the actual storage of objects on the drives of its node.
- Objects are stored as binary files on the drive using a path that is made up in part of its associated partition (which we will discuss shortly) and the operation's timestamp.
- The timestamp is important as it allows the object server to store multiple versions of an object.
- The object’s metadata (standard and custom) is stored in the file’s extended attributes (xattrs) which means the data and metadata are stored together and copied as a single unit.

CINDER

- The Block Storage (or volume) service provides persistent block storage management for virtual hard drives.
- Block Storage allows the user to create and delete block devices, and to manage the attachment of block devices to servers.
- The actual attachment and detachment of devices is handled through integration with the Compute service. Both regions and zones can be used to handle distributed block storage hosts.

- Block storage is appropriate for performance-sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block-level storage.
- Additionally, snapshots can be taken to either restore data or to create new block storage volumes (snapshots are dependent upon driver support).
- Basic operations include:
  - ❖ Create, list, and delete volumes.
  - ❖ Create, list, and delete snapshots.
  - ❖ Attach and detach volumes to running virtual machines.



i. Cinder API:

- Responds to and handles requests, and places them in the message queue.
- When an incoming request is received, the API service verifies identity requirements are met and translates the request into a message denoting the required block storage actions.
- The message is then sent to the message broker for processing by the other Block Storage services.

ii. Cinder Volume:

- Carves out storage for virtual machines on demand.
- The volume service manages the interaction with the block storage devices.
- As requests come in from the scheduler, the volume service creates, modifies, and removes volumes as required.
- A number of drivers are included for interaction with storage providers.

iii. Cinder-scheduler:

- Assigns tasks to the queue and determines the provisioning volume server.

- The scheduler service reads requests from the message queue and determines on which block storage host the request must be actioned.
- The scheduler then communicates with the volume service on the selected host to process the request.

iv. **Database:**
   - Provides state information.

v. **Queue:**
   - Provides the AMQP message queue.
   - RabbitMQ (also used by other services) handles the OpenStack transaction management, including queuing, distribution, security, management, clustering, and federation.
   - Messaging becomes especially important when an OpenStack deployment is scaled and its services are running on multiple machines.

Openstack Test-drive:

- Openstack is technically just an API specification for managing cloud server and overall cloud infrastructures.
- Different organization have created software packages that implement Openstack . to use openstack you need to acquire such software.
- Fortunately , there are several free and open source solutions.
- Since openstack is for managing the cloud infrastructure to get a minimal setup, you need two machines: One will be the infrastructure you are managing, and one will be the manager. But if you are really strapped for hardware, you can fit both on a single machine.

Openstack Software and API:

- Openstack community has created a base set of software that you can use to get started tying out. This software called as DevStack, is primarily intended for testing and development purposes, and not for production. But it includes everything you need to get going, including management tools.
- The management tools are where things become a bit fuzzy between openstack being “just an API” and a set of software makes use of an API. Anyone can technically build a set of software that matches the openstack specification. That software can be either on the managed side or the manager side.
- The manages side would implement the API allowing any openstack-complaint management tools to manage it. The manager side would be a tool that can manage

any Openstack – compliant platform. The managed side is where Openstack mostly lives with its various APIs.

- There are several API’s but here are a couple:
    - Compute: Compute is the main API for allocating and de-allocating servers. The code name for this API is NOVA. Openstack also allows you to create and manage images that represent backup of disk drives. This portion of openstack called Glance.
    - These images are often going to contain operating system such as Linux. The idea here is that you can choose an image that you will use to create a new server.
    - The image might contain for example, an Ubuntu 18.04 server that is already configured with the software you need. Then you would use the compute server to launch a couple of server using that image.
    - Because each server starts from the same image, they will be identical and already configured with the software you placed on the image.

Up and Running:

- Knowing all this lets give it a shot. The steps here are small, but you’ll want to keep in mind how these steps would scale to larger situation and the decision you would need to make. One important first decision in what service you want to use.
- Openstack provide a whole range of service beyond the compute and image APIs. As mention earlier. The second decision is how many hardware servers i.e bare metal servers you want to use as well as how many virtual machine you want to allow each bare metal server to run. And finally you all want to put together a plan where users have limits or quotas on the amount of virtual machine and drive space they can use.
- For example you might install Ubuntu 18.04 in vmware, but to make this practice session as simple as possible, if you want you can install a desktop version of Ubuntu instead of the server version and then run the Horizon console right on that same machine. As an alternative you can instead create a new server on a cloud hosting service and install Ubuntu on it.
- Next, you’ll need to install git. You don’t need to know how to actually use git; it’s just used here to get the latest version of the DevStack software. Now create a directory to house your OpenStack files. Switch to that directory and paste the following command into the console:

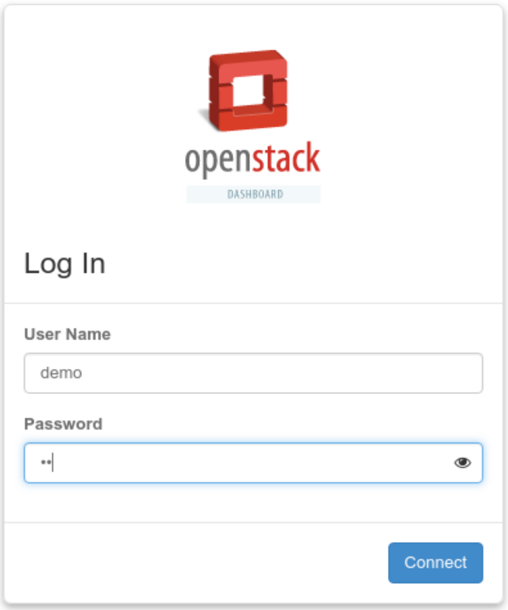
```
git clone https://git.openstack.org/openstack-dev/devstack
```

- This will create a subdirectory called devstack. Switch to the new devstack, and then switch to the samples directory under it, like so:

```
cd devstack/samples
```

- This directory contains two sample configuration files. (Check out OpenStack's online documentation page about these configuration files.) Copy these up to the parent devstack directory:
  ```
  cp local* ..
  ```
- Now move back up to the parent devstack directory:
  ```
  cd ..
  ```
- Next, you need to make a quick modification to the local.conf file that you just copied over. Specifically you need to add your machine's IP address within the local network. (It will likely start with a 10.) Open up local.conf using your favorite editor and uncomment (i.e. remove the #) the line that looks like this:
  ```
  #HOST_IP=w.x.y.z
  ```
- and replace w.x.y.z with your IP address. Here's what mine looks like:
  ```
  HOST_IP=10.128.56.9
  ```
- (If you're installing OpenStack, you probably know how to find your ipaddress. I used the ifconfig program.)
- Now run the setup program by typing:
  ```
  ./stack.sh
  ```
- If you watch, you'll see several apt-get installations taking place followed by overall configurations. This process will take several minutes to complete. At times it will pause for a moment; just be patient and wait. Eventually you'll see a message that the software is configured, and you'll be shown a URL and two usernames (admin and demo) and a password (nomoresecrete).
- Note, however, that when I first tried this, I didn't see that friendly message, unfortunately. Instead, I saw this message:
- "Could not determine a suitable URL for the plugin."
- Thankfully, somebody posted a message online after which somebody else provided a solution. If you encounter this problem, here's what you need to do. Open the stack.sh file and search for the text OS_USER_DOMAIN_ID. You'll find this line:
  ```
  export OS_USER_DOMAIN_ID=default
  ```
- and then comment it out by putting a # in front of it:
  ```
  #export OS_USER_DOMAIN_ID=default
  ```
- Then a few lines down you'll find this line:
  ```
  export OS_PROJECT_DOMAIN_ID=default
  ```
- which you'll similarly comment out:
  ```
  #export OS_PROJECT_DOMAIN_ID=default
  ```
- Now you can try again. (I encourage you at this point to read the aforementioned post and learn more about why this occurred.) Then to start over you'll need to run the unstack script:
  ```
  ./unstack.sh
  ```
- And then run stack.sh again:
  ```
  ./stack.sh
  ```

- Finally, when this is all done, you’ll be able to log into the Horizon dashboard from the web browser using the URL printed out at the end of the script. It should just be the address of your virtual machine followed by dashboard, and really you should be able to get to it just using localhost:
  http://localhost/dashboard
- Also, depending on how you’ve set up your virtual machine, you can log in externally from your host machine. Otherwise, log into the desktop environment on the virtual machine and launch the browser.
- You’ll see the login page like so:



demo

FOSS CLOUD

- The FOSS-Cloud is a Software, which enables you, to build your own Private- or your Public-Cloud!
- The FOSS-Cloud environment (software and hardware) is an integrated and redundant server infrastructure to provide cloud-Services, Windows or Linux based SaaS-, Terminal Server-, Virtual Desktop Infrastructure (VDI) or virtual server-environmens.
- It makes virtual machines available, which can be accessed from internally as well as from the Internet.
- FOSS-Cloud covers all of the aspects of an Open Source IT environment. The FOSS-Cloud is the most advanced Open Source Cloud.

Features

- Full integration into existing Windows and Linux environments
- Cloud for Server- and Desktop Virtualization
- Powerful Vitalizing for Windows and Linux 32/64bit
- Published Desktop
- Persistent Virtual Machines including session transfer to other devices
- Dynamic Desktop with Golden Image to serve user groups

- Application Streaming
- Published application support with RDS
- Video streaming (M-Jpeg)
- High resolution Display (up to 4 Displays, 32 bit)
-
- Pools of Network- and Hardware-Ressources or Virtual Machines
- VDI Access through Windows and Linux, PXE Boot and Handhelds
- Bi-directional audio and Video
- Smartcard authentication
- USB redirection
- Web-based management console
- Multi-tenancy

Advantages:

a Quality :

- In general, open source software gets closest to what users want because those users can have a hand in making it so.
- It's not a matter of the vendor giving users what it thinks they want–users and developers make what they want

b Customizability:

- Since the code is open, it's simply a matter of modifying it to add the functionality they want

c Freedom:

- users are in control to make their own decisions and to do what they want with the software.

d Flexibility.

- Open source software, is typically much less resource-intensive, meaning that you can run it well even on older hardware.
- It's up to you–not some vendor–to decide when it's time to upgrade.

e Interoperability:

- Open source software is much better at adhering to open standards than proprietary software is.
- If you value interoperability with other businesses, computers and users, and don't want to be limited by proprietary data formats, open source software is definitely the way to go.

f Auditability:

- With closed source software, you have nothing but the vendor's claims telling you that they're keeping the software secure and adhering to standards, for example.
- It's basically a leap of faith. The visibility of the code behind open source software, however, means you can see for yourself and be confident.

g Support Options :

- Open source software is generally free, and so is a world of support through the vibrant communities surrounding each piece of software.

- Most every Linux distribution, for instance, has an online community with excellent documentation, forums, mailing lists, forges, wikis, newsgroups and even live support chat.
- For businesses that want extra assurance, there are now paid support options on most open source packages at prices that still fall far below what most proprietary vendors will charge.
- Providers of commercial support for open source software tend to be more responsive, too, since support is where their revenue is focused

h Cost :

- Between the purchase price of the software itself, the exorbitant cost of mandatory virus protection, support charges, ongoing upgrade expenses and the costs associated with being locked in, proprietary software takes more out of your business than you probably even realize

FUNCTIONALITY/Components

I. VDI (Virtual Desktop Infrastructure)
II. VSI (Virtual Server Infrastructure)
   Infrastructure as a Service (IaaS)
   Plattform as a Service (PaaS)
   Software as a Service (SaaS)
III. Storage Cloud

I. VDI (Virtual Desktop Infrastructure)

- As far as the users are concerned, VDI gives you the freedom of accessing your desktop from anywhere at any time through a VDI client software.
- VDI can be classified as persistent and non-persistent. Persistent VDI is customized for a personal user, and the user can log in to the same desktop each time.
- Non-persistent VDI consists of desktops that revert to their initial state after the user logs out.
- In this article, we will discuss the components involved in the VDI technology and it's working. Let us first discuss some of the benefits that the VDI technology entails.

Benefits of VDI

1. Access

- The most distinguishing feature of VDI is remote access.
- Traditional desktops can be viewed as connected (or, one can say 'restricted') to a single system. As soon as you are away from the system, you could not access your desktop anymore.
- With VDI, you can access your desktop from anywhere, day or night.

2. Security

- Another vital aspect of VDI is security. Conventionally, your Operating System, Applications and data are all stored on your local hardware like laptops or PCs.

- In case the computer is stolen or damaged, all the data is lost. You also have to buy a new laptop and start all over again with OS installation.
- With VDI, as remote data centers store the data with high-level redundancy, you do not need to worry about data loss. Even if you lose the device, you can access your desktop from any other device.

3. Device Portability

- VDI technology enables you to access your desktop from various devices.
- As in the case of VDI, the desktop is not bound to the hardware; it can be accessed from multiple devices.
- You can use mobile, laptops, tablets or thin clients to view your desktop.

- Easy Desktop Provisioning – Since with VDI you don't have to configure each system manually, it very easy to provision the desktops in VDI.
- The virtual desktops can be provisioned almost instantaneously as the settings have to be mirrored from a desktop image.

4. Data Center Facilities

- When you are availing VDI from a cloud service provider, the desktops are hosted on servers situated in high-performance data centers.
- You get all the facilities and features associated with the data center namely advanced security, high-end infrastructure and disaster recovery plan among others.

5. Cost Reduction

- By availing VDI services from a cloud provider, you eliminate the cost of hardware.
- You can access your desktop from any device and can use the most outdated hardware in your office.
- A thin client, mobile or tablet can also be used for the same purpose.

VDI Basic Components

- The functioning of VDI consists mainly of two parts- Hypervisor and Connection Broker.
- But before we move on to these aspects, let us first discuss in brief about virtualization.

i. Virtualization

- Virtualization is the technology that divides the system architecture into different layers.
- Before virtualization, the hardware was bound to the operating system (OS) at the time of installation.
- Thus, in the case of a hardware failure, the OS also crashed, and you would lose all the data.
- Through virtualization, the OS and the underlying hardware are separated by software called hypervisor.
- You can install multiple operating systems on a hypervisor installed server.

ii. Hypervisor

- Hypervisor is software that separates the operating system from the underlying hardware.
- The hypervisor creates a virtualized environment in which the hardware can be divided into multiple virtual machines (VMs). Each virtual machine can have its unique configuration, OS, and applications.
- In VDI, the hypervisor creates desktop instances in these VMs.
- Each desktop instance can act as a separate desktop and can be provisioned to the users.
- The High Availability (HA) function in the hypervisor also lets it connect to multiple servers.
- Hence, even if a physical server fails, your desktop instance is moved to another server almost instantaneously.

iii. Connection Broker

- Connection broker is a software program that connects the users to the desktop instances.
- It is also responsible for the authentication of the users and sending them to their desktop instances.
- The connection broker also keeps track of active and inactive desktops.
- When a user sends a request to connect to a desktop, it provides the user with an idle desktop instance.
- When a user disconnects desktop, it updates the status to inactive.

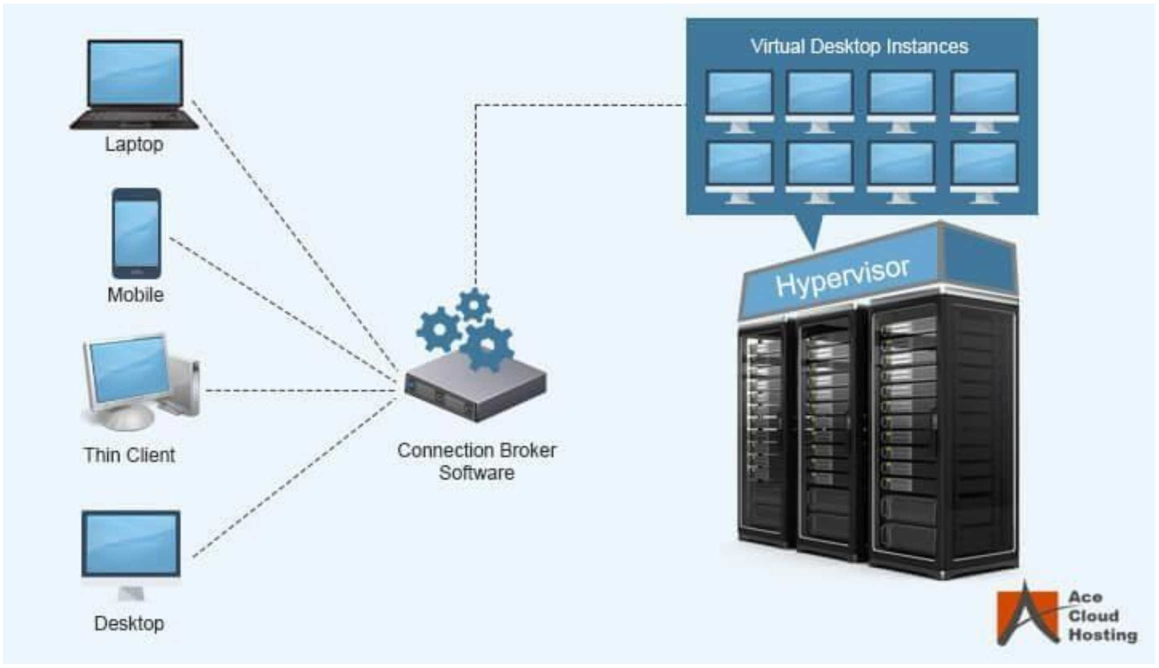
iv. Desktop Pools

- Desktops pools are a group of similar desktops that can be configured according to a specific function.
- For instance, departments like accounting and IT in an office might use desktops with different applications and configuration.
- The accounting department may need applications like Sage or QuickBooks, which may not be required for IT.
- You can create a desktop pool for these departments with the similar configuration.

v. Application Virtualization

- Application Virtualization is the technology used to create a virtualized application image and replicate it to all the virtual desktops in a desktop pool.
- It makes application deployment easy and hassle-free.
- VMware ThinApp is an example of that.
- It creates an executable file of the application by capturing pre and post images of the system before and after installation of the software.
- You can use this file in any system without going through the installation process repeatedly.

How VDI Workes



- When the user logs in to their desktop from the client software, the request is accepted by the connection broker after authentication. The connection broker then analyzes the request and sends the user to their desktop in the desktop pool.
- The hypervisor installed on the servers create multiple VMs on which the virtual desktop is hosted. The High Availability feature in hypervisors can combine the resources of various servers and the virtual desktops can be migrated to another server if needed.
- The admin can turn the virtual desktop off when the user is not using it. This way you can accommodate more users than the actual capacity of the server. For instance, on a server with 500 GB RAM, you can create 600 Virtual desktops with 1 GB RAM assuming that not all 600 users will use the desktop at the same time.
- The desktop image is mirrored from a master desktop to all the other desktops. It is called cloning. The cloning can be full or linked.
- In Linked cloning, the virtual disk of the master desktop is linked to all other desktops. It saves disk space of the server. The data of all the users is saved separately. However, the cloned desktops have to be linked to master desktops at all time.
- In Full cloning, the cloned desktops are not linked to master and function as independent desktops. Hence, all the desktops use separate disk space.
- The VDI management software like VMware View Manager can be used to create desktop pools. The admin user can manage the desktop pools, provision new desktops, create new pools and set up policies.

II. VIRTUAL SERVER INFRASTRUCTURE

- A **virtual infrastructure** is a software-based IT **infrastructure** being hosted on another physical **infrastructure** and meant to be distributed as a service as in cloud computing's **infrastructure** as a service (IaaS) delivery model.
- The main purpose of a virtual infrastructure is to bring enterprise-level technology to organizations that cannot afford the large capital required to pay for the hardware, software licenses, setup and continual maintenance of an actual data center infrastructure.
- The technology involves virtualization, which is the utilization of physical server resources to host logical or virtual servers and networking hardware in order to optimize resources and drive costs down by hosting multiple virtual servers in a single host server.
- The idea is that no single server is actually taxed enough to the point that its resource limits are reached so it would be more prudent to make use of these resources by running multiple logical servers that, together, can make use of the actual capacity of the host.
- This lean approach allows for sharing and distributing resources, which, in turn, promotes flexibility, scalability and lower total cost of ownership.
- It is divided in to three components.
  a) IAAS
  b) PAAS
  c) SAAS

Benefits of a virtual infrastructure:

- **Scalable –** Allows provisioning as many or as few logical servers as required, and users only pay for what they use.
- **Flexible –** Allows for multiple server and networking configurations as compared to a hardwired physical infrastructure, which requires more capital and effort to change.
- **Secure –** Allows more security to be layered on top of whatever security is already present in the virtual infrastructure because all traffic to the virtual infrastructure goes through the actual physical infrastructure.
- **Load balancing –** Allows software-based servers to share workloads easily and distribute them properly so that no single logical server is taxed more than the others.
- **Backup and recovery –** Promotes easier backups because everything can be saved somewhere, allowing for quick recovery in other hosts if a few hosts are down. This is almost impossible with physical servers, which have to be revived before services can resume.

III. Storage Cloud:

- Cloud storage allows you to save data and files in an off-site location that you access either through the public internet or a dedicated private network connection.
- Data that you transfer off-site for storage becomes the responsibility of a third-party cloud provider.
- The provider hosts, secures, manages, and maintains the servers and associated infrastructure and ensures you have access to the data whenever you need it.
- Cloud storage delivers a cost-effective, scalable alternative to storing files on on-premise hard drives or storage networks.
- Computer hard drives can only store a finite amount of data. When users run out of storage, they need to transfer files to an external storage device.
- Traditionally, organizations built and maintained storage area networks (SANs) to archive data and files.
- SANs are expensive to maintain, however, because as stored data grows, companies have to invest in adding servers and infrastructure to accommodate increased demand.
- Cloud storage services provide elasticity, which means you can scale capacity as your data volumes increase or dial down capacity if necessary.
- By storing data in a cloud, your organization save by paying for storage technology and capacity as a service, rather than investing in the capital costs of building and maintaining in-house storage networks.
- You pay for only exactly the capacity you use. While your costs might increase over time to account for higher data volumes, you don’t have to overprovision storage networks in anticipation of increased data volume.

How does it work?

- Like on-premise storage networks, cloud storage uses servers to save data; however, the data is sent to servers at an off-site location.
- Most of the servers you use are virtual machines hosted on a physical server.
- As your storage needs increase, the provider creates new virtual servers to meet demand.

- Typically, you connect to the storage cloud either through the internet or a dedicated private connection, using a web portal, website, or a mobile app.
- The server with which you connect forwards your data to a pool of servers located in one or more data centers, depending on the size of the cloud provider's operation.
- As part of the service, providers typically store the same data on multiple machines for redundancy. This way, if a server is taken down for maintenance or suffers an outage, you can still access your data.

Cloud storage is available in private, public and hybrid clouds.

- **Public storage clouds**: In this model, you connect over the internet to a storage cloud that's maintained by a cloud provider and used by other companies. Providers typically make services accessible from just about any device, including smartphones and desktops and let you scale up and down as needed.
- **Private cloud storage:** Private cloud storage setups typically replicate the cloud model, but they reside within your network, leveraging a physical server to create instances of virtual servers to increase capacity. You can choose to take full control of an on-premise private cloud or engage a cloud storage provider to build a dedicated private cloud that you can access with a private connection. Organizations that might prefer private cloud storage include banks or retail companies due to the private nature of the data they process and store.
- **Hybrid cloud storage**: This model combines elements of private and public clouds, giving organizations a choice of which data to store in which cloud. For instance, highly regulated data subject to strict archiving and replication requirements is usually more suited to a private cloud environment, whereas less sensitive data (such as email that doesn't contain business secrets) can be stored in the public cloud. Some organizations use hybrid clouds to supplement their internal storage networks with public cloud storage.

Pros and cons

As with any other cloud-based technology, cloud storage offers some distinct advantages. But it also raises some concerns for companies, primarily over security and administrative control.

Pros

The pros of cloud storage include the following:

- **Off-site management**: Your cloud provider assumes responsibility for maintaining and protecting the stored data. This frees your staff from tasks associated with storage, such as installation, administration, and maintenance. As such, your staff can focus on other priorities.
- **Quick implementation**: Using a cloud service accelerates the process of setting up and adding to your storage capabilities. With cloud storage, you can provision the service and start using it within hours or days, depending on how much capacity is involved.

- **Cost-effective**: As mentioned, you pay for the capacity you use. This allows your organization to treat cloud storage costs as an ongoing operating expense instead of a capital expense with the associated upfront investments and tax implications.
- **Scalability**: Growth constraints are one of the most severe limitations of on-premise storage. With cloud storage, you can scale up as much as you need. Capacity is virtually unlimited.
- **Business continuity**: Storing data offsite supports business continuity in the event that a natural disaster or terrorist attack cuts access to your premises.

Cons

Cloud storage cons include the following:

- **Security**: Security concerns are common with cloud-based services. Cloud storage providers try to secure their infrastructure with up-to-date technologies and practices, but occasional breaches have occurred, creating discomfort with users.
- **Administrative control**: Being able to view your data, access it, and move it is another common concern with cloud resources. Offloading maintenance and management to a third party offers advantages but also can limit your control over your data.
- **Latency**: Delays in data transmission to and from the cloud can occur as a result of traffic congestion, especially when you use shared public internet connections. However, companies can minimize latency by increasing connection bandwidth.
- **Regulatory compliance**: Certain industries, such as healthcare and finance, have to comply with strict data privacy and archival regulations, which may prevent companies from using cloud storage for certain types of files, such as medical and investment records. If you can, choose a cloud storage provider that supports compliance with any industry regulations impacting your business.

Types of Cloud Storage

There are three types of cloud data storage: object storage, file storage, and block storage. Each offers their own advantages and have their own use cases:

2. Object Storage **-** Applications developed in the cloud often take advantage of object storage's vast scalability and metadata characteristics. Object storage solutions like Amazon Simple Storage Service (S3) are ideal for building modern applications from scratch that require scale and flexibility, and can also be used to import existing data stores for analytics, backup, or archive.
3. File Storage **-** Some applications need to access shared files and require a file system. This type of storage is often supported with a Network Attached Storage (NAS) server. File storage solutions like Amazon Elastic File System (EFS) are ideal for use cases like large content repositories, development environments, media stores, or user home directories.
4. **Block Storage -** Other enterprise applications like databases or ERP systems often require dedicated, low latency storage for each host. This is analogous to direct-attached storage (DAS) or a Storage Area Network (SAN). Block-based cloud storage solutions like Amazon Elastic Block Store (EBS) are provisioned with each virtual server and offer the ultra low latency required for high performance workloads.

OPENSTACK

What is Openstack?

- Openstack is a collection of open source technology delivering a massively scalable cloud operating system.
- OpenStack is a software package that provides a cloud platform for Public and Private cloud covering various use cases including Enterprise and Telecom.
- The main focus is on Infrastructure as a Service (IaaS) cloud and additional services built upon IaaS.
- It is a set of software that allows you to build your own cloud infrastructure and allows you to launch your own cloud infrastructure on which you can run your own application like development testing and deployment.
- Openstack releasing two versions every years.
- There are number of companies which contributing to the openstack: rackspace, cisco, IBM, redhat, vmware, dell, hp.
- Rackspace is the one who created the openstack in 2010 with NASA.
- Openstack is the future of cloud computing backed by some of biggest company.
- The openstack is managed by openstack foundation.

Advantages of an Openstack

VI. Strong security

✓ One of the main reasons why OpenStack is so popular in the world of cloud computing is – it has outstanding security features that keep you secure all the time.

VII. Open-source

✓ OpenStack is open-source that makes it the favourite cloud software for the developers and entrepreneurs. The source code of this project can be found at www.github.com/openstack/.

VIII. Development support

✓ OpenStack has been receiving a concrete development support from many prestigious companies and from the top developers of the IT industry for many years.

IX. An array of services for different tasks

✓ OpenStack provides you with an array of services that accomplish different necessary tasks for your private, public and hybrid cloud.

✓ OpenStack using a dashboard, Mistral for managing workflows, Sahara for providing Hadoop clusters, and Zaqar is for messaging on the cloud.

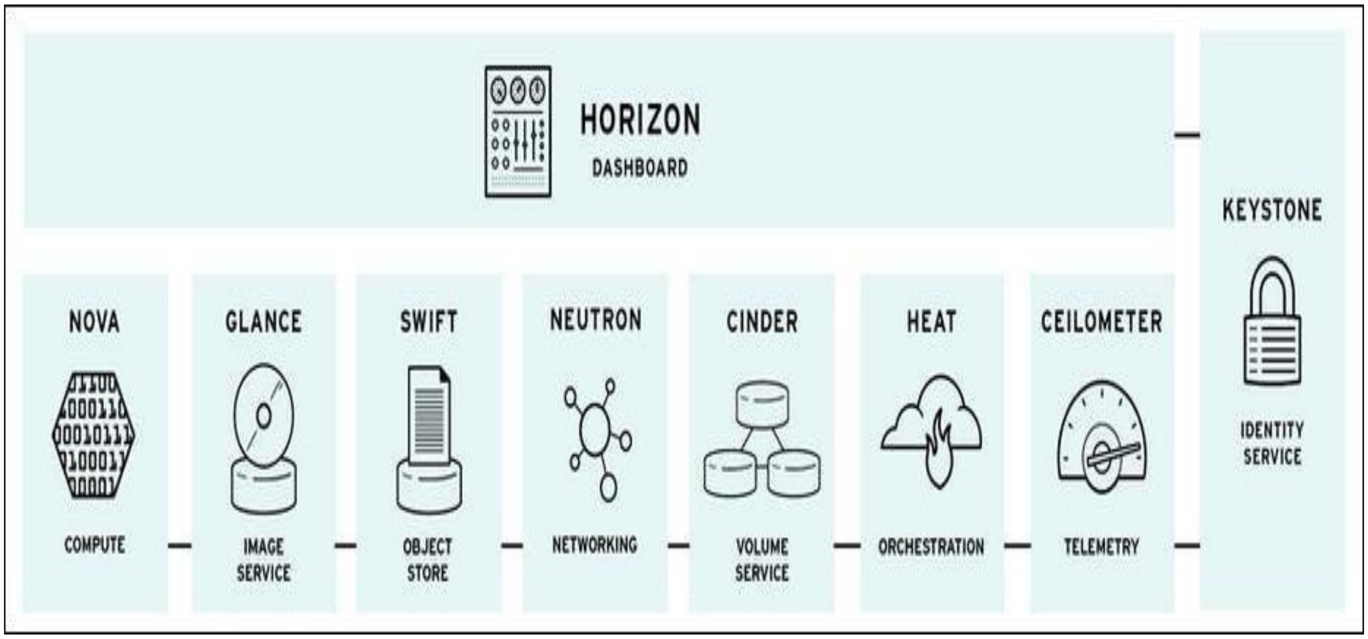
X. Easy to access and manage OpenStack

✓ OpenStack can be accessed and managed in several ways, which is a great benefit for you.

  - GUI-based dashboard
  - Command line tools
  - APIs for programmers.

COMPONENTS OF AN OPENSTACK:

- OpenStack embraces a modular architecture to provide a set of core services that facilitates scalability and elasticity as core design tenets.
- This chapter briefly reviews OpenStack components, their use cases and security considerations.



x. **Nova**
   - This is the primary computing engine behind OpenStack.
   - This allows deploying and managing virtual machines and other instances to handle computing tasks.

xi. **Swift**
   - The storage system for objects and files is referred to as Swift.
   - In the traditional storage systems, files are referred to a location on the disk drive, whereas in OpenStack Swift files are referred to by a unique identifier and the Swift is in charge where to store the files.
   - The scaling is therefore made easier because the developers don’t have the worry about the capacity on a single system behind the software.
   - This makes the system in charge of the best way to make data backup in case of network or hardware problems.

xii. **Cinder**

- This is the respective component to the traditional computer access to specific disc locations.
- It is a block storage component that enables the cloud system to access data with higher speed in situations when it is an important feature.

xiii. **Neutron**

- Neutron is the networking component of OpenStack.
- It makes all the components communicate with each other smoothly, quickly and efficiently.

xiv. **Horizon**

- In Horizon you can login with different user.
- This is the OpenStack dashboard. It's the graphical interface to OpenStack and the first component that users starting with OpenStack will see.
- There is an OpenStack API that allows developers to access all the components individually, but the dashboard is the management platform for the system administrators to have a know what is going on in the cloud.

xv. **Keystone**

- This is the identity service who actually maintain who is using what kind of things?
- This is the component that provides identity services for OpenStack.
- Basically, this is a centralized list of all the users and their permissions for the services they use in the OpenStack cloud.

xvi. **Glance**

- It is a component that provides image services or virtual copies of the hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances.

xvii. **Ceilometer**

- Ceilometer provides data measurement services, thus enabling the cloud to offer billing services to individual users of the cloud.
- It measures system usage by each user for each of the components of the cloud and makes reporting available.

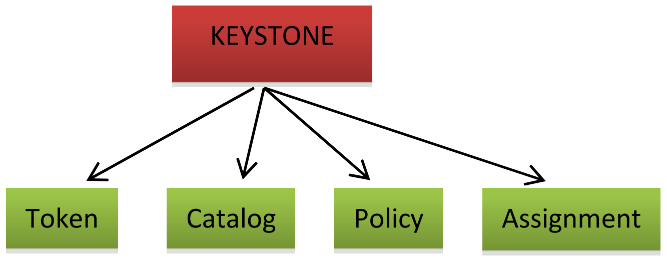
xviii. **Heat**

- Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application.

- In this way, it helps to manage the infrastructure needed for a cloud service to run.

KEYSTONE:

- This is the authentication service used for authentication and authorization of an openstack. Every service that want to avail is first get to authenticate first.
- Keystone is an openstack service that provide API client authentication , service discovery and distributed multi-tenent authentication by implementing openstack identity API.
- It keep track of who and what is using.
- Keystone have four components.
  e. Token Service
  f. Catalog Service
  g. Policy service
  h. Assignment service.



v **Token Service:**
- It validate and manages token to authenticate the request once users credential already been verified.
- It is session based management service that token been uses.

vi **Catalog Services:**
- It provides you with endpoint registry which is used for endpoint discovery.
- Openstack is connected to the every other services with endpoints.

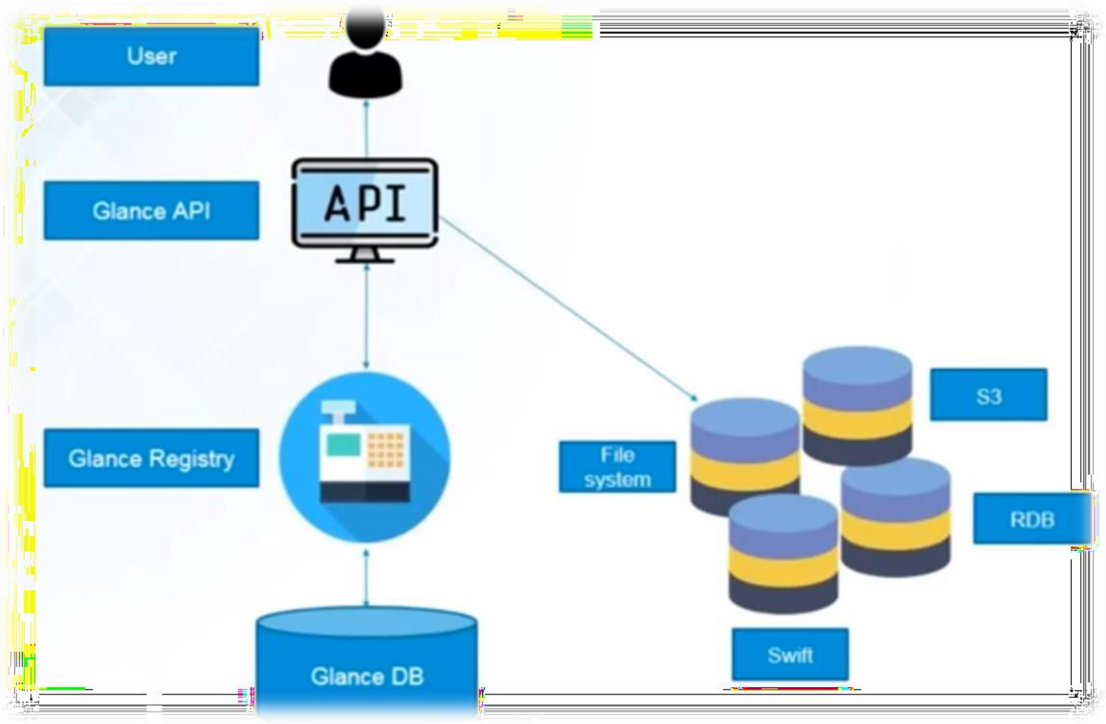
vii **Policy Service:**
- It provides rule based authorization user and it is associated with rule management interface.
- There are some policy mentioned in rule management so this policy service will check whether user authorized to do that or not.

viii **Assignment Service:**
- It provides data about roles.
- Roles are the levels of Authorization that end-user can operate.

GLANCE:

- The Glance project provide a service where users can upload and discover data assets that are meant to be used with other services.
- This currently includes image and metadata definitions.
- Glance image service includes discovering, registering and retrieving virtual machine image.
- Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.
- The glance is a repository for images , the glance store all the virtual images that you have added into openstack and from there on these images which stores on glance you can lunch the instances from there.
- And these instances you can used set up your develop environment of your application and for testing environment.

GLANCE ARCHITETURE:



- There are three pieces to Glance architecture: *glance-api*, *glance-registry*, and the image store.

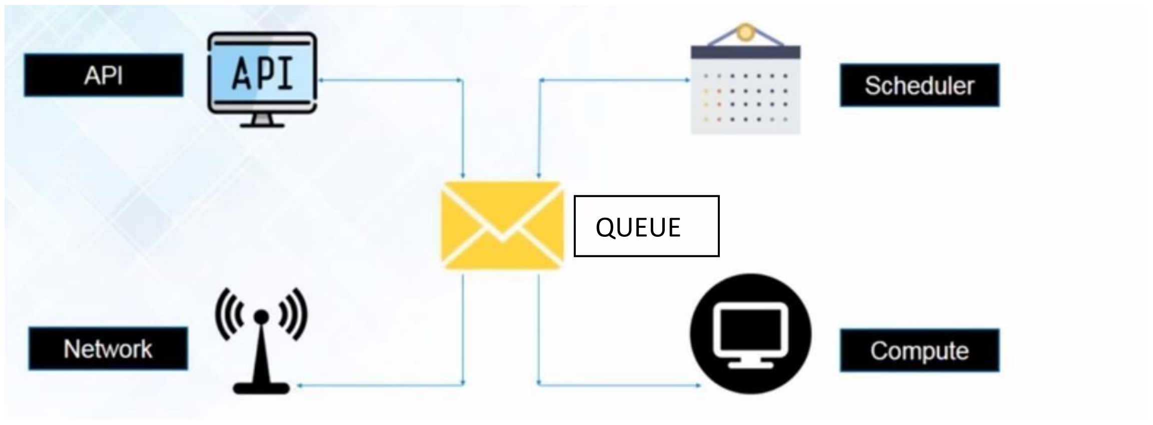
- As you can probably guess, *glance-api* accepts API calls for image retrieval, storage and discovery.
- The *glance-registry* stores process and retrieves metadata about images metadata includes the items such as size and type.
- The image store can be a number of different object stores, including Swift. Figure illustrates Glance’s logical architecture.
- *glance-api* is similar in functionality to *nova-api*, in that it accepts incoming API requests and then communicates with the other components (*glance-registry* and the image store) to facilitate querying, retrieving, uploading, or deleting images. By default, *glance-api* listens on port 9292.
- In openstack every service communicate with APIs and here glance also uses API to connect with user and other services as well.
- In above figure user using glance and it connect through Glance API and it will goes to the Glance Registry which is connected to the glance database.
- The Glance database stores all the metadata and definitions.
- And API also connected with the different file system, object storage service, swift,s3, RDB.
- These storage system actually stores the image in glance.
- So the glance registry contains each entry of images and other images that have been added into the glance architecture.
- The metadata is stored into the glance database.
- The images is stored in the swift,s3,RDB, file system and from there on if you want to lunch the images you can lunch it using glance.

NOVA(Compute Node)

- Openstack Nova is a component within the openstack open source cloud computing platform to provide on-demand access to compute resources by provisioning and managing large networks of virtual machines(VMs).
- Nova works with VMWare, Xen,KVM,Hype-V and other virtualization technology.
- Nova is a openstack project design to provide power to massive, scalable, on demand self service access to compute resources.
- It is fault tolerance , recoverable, and provide the API compatibility with system like AWS EC2.
- It is build on massaging architecture and all of its component typically run on several server.

- The nova is a compute zone so this is where your instances get launch and this is where all the computing and processing takes place.
- In NOVA All the components communicate through massage queue.

ARCHITECTURE



6. NOVA API

- The nova-api daemon is the heart of Nova. You may see it illustrated on many pictures of Nova as API and "Cloud Controller."
- Its primary purpose is to accept and fulfill incoming API requests.
- To accept and fulfill API requests, *nova-api* provides an endpoint for all API queries (accepting requests using either the OpenStack API or the Amazon EC2 API), initiates most of the orchestration activities (such as running an instance), and also enforces some policy (mostly quota checks).
- For some requests, it will fulfill the entire request itself by querying the database and then returning the answer.
- For more complicated requests, it will pass messages to other daemons through a combination of writing information to the database and adding messages to the queue.
- By default, *nova-api* listens on port 8773 for the EC2 API and 8774 for the OpenStack API.

7. Scheduler

- The nova-scheduler process is conceptually the simplest piece of code in Nova: it takes a virtual machine instance request from the queue and determines where it should run (specifically, which compute server host it should run on).
- In practice, however, this will grow to be the most complex piece, as it needs to factor in the current state of the entire cloud infrastructure and apply complicated algorithms to ensure efficient usage.
- To that end, nova-scheduler implements a pluggable architecture that lets you choose (or write) your own algorithm for scheduling.

Table 4-1 details the current scheduler choices.

Scheduler	**Notes**
Simple	Attempts to find least loaded host.
Chance	Choose random available host from service table. This is the default scheduler.
Zone	Picks random host from within an availability zone.

8. **Compute Worker**

- The nova-compute process is primarily a worker daemon that creates and terminates virtual machine instances.
- The process by which it does so is fairly complex, but the basics are simple: accept actions from the queue and then perform one or a series of virtual machine API calls to carry them out while updating state in the database.
- An example of this would be nova-compute accepting a message from the queue to create a new instance and then using the libvirt library to start a new KVM instance.
- There are a variety of ways that nova-compute manages virtual machines. The most common is through a software package called libvirt.
- This is a toolkit (API, daemon, and utilities) created by Red Hat to interact with the capabilities of a wide range of Linux virtualization technologies.
- While libvirt may be the most common, nova-compute also uses the Xen API, vSphere API, Windows Management Interface, and others to support other virtualization technologies.

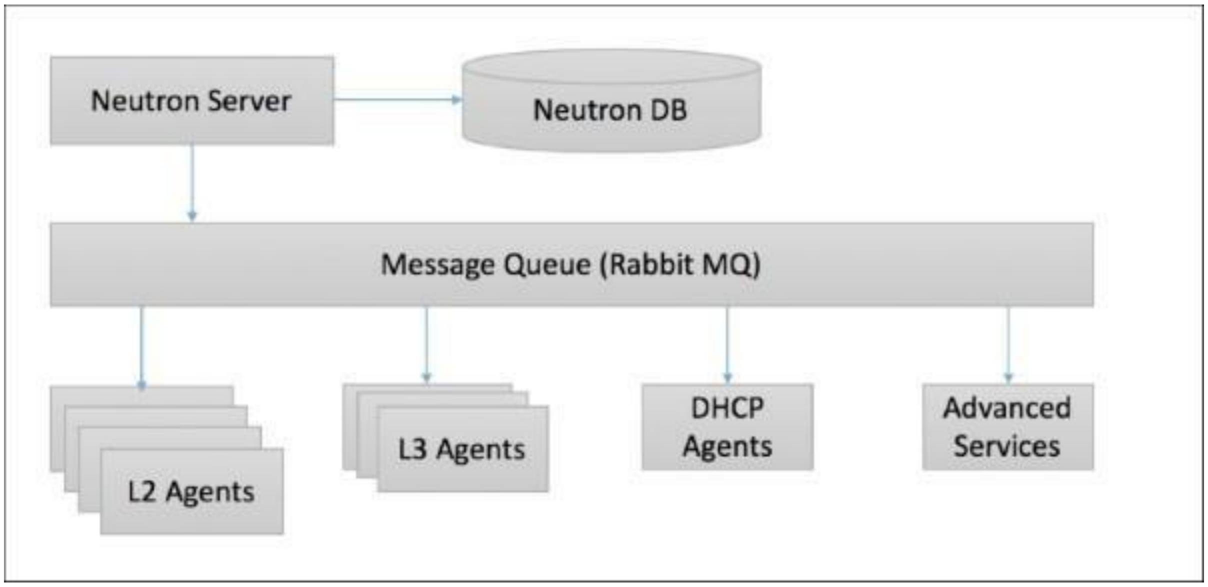
9. **Queue**

- The queue provides a central hub for passing messages between daemons. This is currently implemented with RabbitMQ today, but theoretically could be any AMPQ message queue supported by the Python ampqlib and carrot libraries.
- Nova creates several types of message queues to facilitate communication between the various daemons.
- These include: topics queues, fanout queues, and host queues.
- Topics queues allow messages to be broadcast to the number of particular class of worker daemons.
- For example, Nova uses these to pass messages to all (or any) of the compute or volume daemons.
- This allows Nova to use the first available worker to process the message. Host queues allow Nova to send messages to specific services on specific hosts.
- For example, Nova often needs to send a message to a specific host's compute worker to take action on a particular instance.
- Fanout queues are only currently used for the advertising of the service capabilities to nova-scheduler workers.

10. **Database**
    - The database stores most of the configuration and run-time state for a cloud infrastructure.
    - This includes the instance types that are available for use, instances in use, networks available, and projects.

NEUTRON

- The architecture of Neutron is simple, but it is with the agents and plugins where the real magic happens!
- Neutron is a networking project focused on delivering Networking as-a-service (NaaS) in a virtual compute environment.
- Neutron has replaced the original networking application program interface (API) called Quantum in openstack.
- Neutron is designed by address deficiencies in backed-in networking technology found in cloud environment.
- Neutron architecture has been presented in the following diagram:



Let's discuss the role of the different components in a little detail.

e. The Neutron server

❖ The function of this component is to be the face of the entire Neutron environment to the outside world. It essentially is made up of three modules:

- **REST service**: The REST service accepts API requests from the other components and exposes all the internal working details in terms of networks, subnets, ports, and so on.
- **RPC service**: The RPC service communicates with the messaging bus and its function is to enable a bidirectional agent communication.
- **Plugin**: A plugin is best described as a collection of Python modules that implement a standard interface, which accepts and receives some standard API calls and then connects with devices downstream. They can be simple plugins or can implement drivers for multiple classes of devices.

❖ The plugins are further divided into core plugins, which implement the core Neutron API, which is Layer 2 networking (switching) and IP address management. If any plugin provides additional network services, we call it the service plugin -- for example, Load Balancing as a Service (LBaaS), Firewall as a Service (FWaaS), and so on.

❖ As an example, Modular Layer 2 (ML2) is a plugin framework that implements drivers and can perform the same function across ML2 networking technologies commonly used in datacenters. We will use ML2 in our installation to work with Open vSwitch (OVS).

f. L2 agent

- ❖ The L2 agent runs on the hypervisor (compute nodes), and its function is simply to wire new devices, which means it provides connections to new servers in appropriate network segments and also provides notifications when a device is attached or removed. In our install, we will use the OVS agent.

g. L3 agent

- ❖ The L3 agents run on the network node and are responsible for static routing, IP forwarding, and other L3 features, such as DHCP.

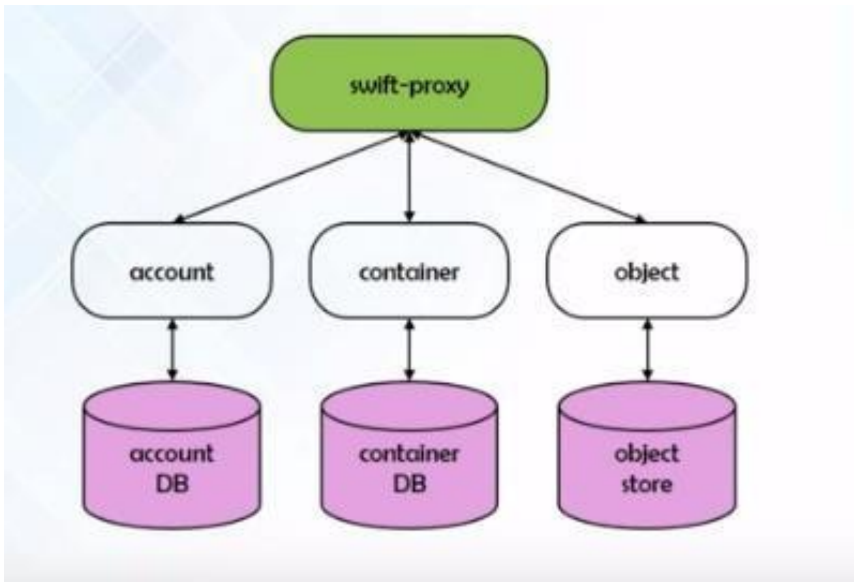
h. DHCP agent (*neutron-dhcp-agent*)

- ❖ Provides DHCP services to tenant networks.
- ❖ This agent is the same across all plug-ins and is responsible for maintaining DHCP configuration.
- ❖ The *neutron-dhcp-agent* requires message queue access. *Optional depending on plug-in.*

SWIFT

- It is an object storage in openstack you can store any kind of file, image ,data in swift. In swift anything you store is convertd into object and then it is stored inside the database.
- Swift is highly available , distributed and consistent object store.
- Organization can use swift to store lots of data efficiently, safely and cheaply.
- Swift powers storage clouds at Comcat, Time Warner, Globo and Wikipedia as well as public cloud like Rackspace, NIT OVH, and IBM SoftLayer.
- It is very easy to use data from the swift.

Architecture



v. **Proxy**

- Handle all of the incoming API requests.
- The Proxy server processes are the public face of Swift as they are the only ones that communicate with external clients.
- As a result they are the first and last to handle an API request.
- All requests to and responses from the proxy use standard HTTP verbs and response codes.
- Proxy servers use a shared-nothing architecture and can be scaled as needed based on projected workloads.
- A minimum of two proxy servers should be deployed for redundancy.
- Should one proxy server fail, the others will take over.
- For example, if a valid request is sent to Swift then the proxy server will verify the request, determine the correct storage nodes responsible for the data (based on a hash of the object name) and send the request to those servers concurrently.
- If one of the primary storage nodes is unavailable, the proxy will choose an appropriate hand-off node to send the request to.
- The nodes will return a response and the proxy will in turn return the first response (and data if it was requested) to the requester.
- The proxy server process is looking up multiple locations because Swift provides data durability by writing multiple–typically three complete copies of the data and storing them in the system.

vi. Account

- The account server process handles requests regarding metadata for the individual accounts or the list of the containers within each account.
- This information is stored by the account server process in SQLite databases on disk.

vii. Container

- The container server process handles requests regarding container metadata or the list of objects within each container.
- It’s important to note that the list of objects doesn’t contain information about the location of the object, simply that it belong to a specific container.
- Like accounts, the container information is stored as SQLite databases.

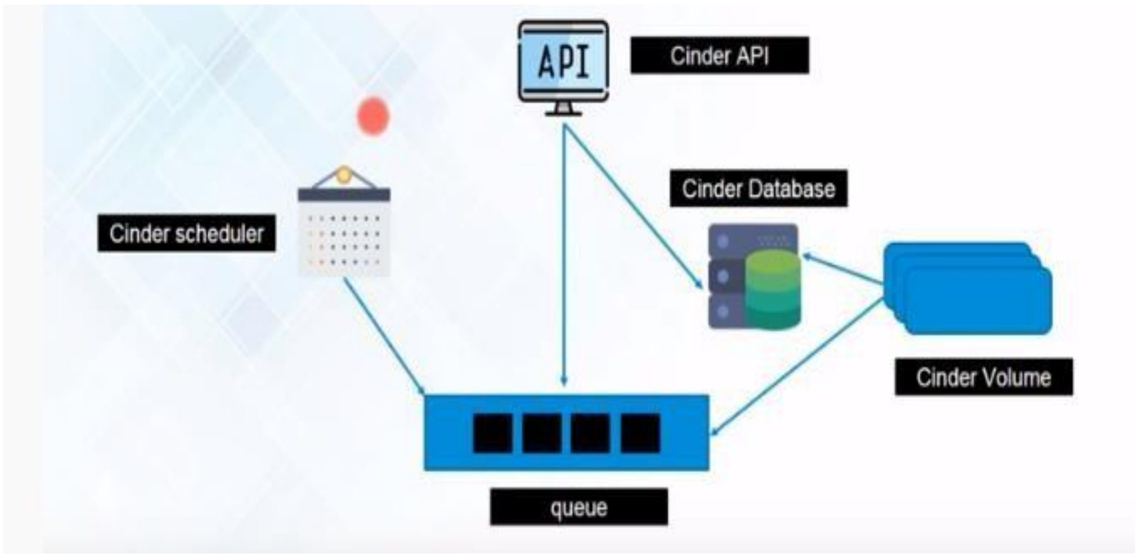
viii. Object

- The object server process is responsible for the actual storage of objects on the drives of its node.
- Objects are stored as binary files on the drive using a path that is made up in part of its associated partition (which we will discuss shortly) and the operation's timestamp.
- The timestamp is important as it allows the object server to store multiple versions of an object.
- The object’s metadata (standard and custom) is stored in the file’s extended attributes (xattrs) which means the data and metadata are stored together and copied as a single unit.

CINDER

- The Block Storage (or volume) service provides persistent block storage management for virtual hard drives.
- Block Storage allows the user to create and delete block devices, and to manage the attachment of block devices to servers.
- The actual attachment and detachment of devices is handled through integration with the Compute service. Both regions and zones can be used to handle distributed block storage hosts.

- Block storage is appropriate for performance-sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block-level storage.
- Additionally, snapshots can be taken to either restore data or to create new block storage volumes (snapshots are dependent upon driver support).
- Basic operations include:
  - ❖ Create, list, and delete volumes.
  - ❖ Create, list, and delete snapshots.
  - ❖ Attach and detach volumes to running virtual machines.



vi. Cinder API:

- Responds to and handles requests, and places them in the message queue.
- When an incoming request is received, the API service verifies identity requirements are met and translates the request into a message denoting the required block storage actions.
- The message is then sent to the message broker for processing by the other Block Storage services.

vii. Cinder Volume:

- Carves out storage for virtual machines on demand.
- The volume service manages the interaction with the block storage devices.
- As requests come in from the scheduler, the volume service creates, modifies, and removes volumes as required.
- A number of drivers are included for interaction with storage providers.

viii. Cinder-scheduler:

- Assigns tasks to the queue and determines the provisioning volume server.

- The scheduler service reads requests from the message queue and determines on which block storage host the request must be actioned.
- The scheduler then communicates with the volume service on the selected host to process the request.

ix. **Database:**

- Provides state information.

x. **Queue:**

- Provides the AMQP message queue.
- RabbitMQ (also used by other services) handles the OpenStack transaction management, including queuing, distribution, security, management, clustering, and federation.
- Messaging becomes especially important when an OpenStack deployment is scaled and its services are running on multiple machines.

Openstack Test-drive:

- Openstack is technically just an API specification for managing cloud server and overall cloud infrastructures.
- Different organization have created software packages that implement Openstack . to use openstack you need to acquire such software.
- Fortunately , there are several free and open source solutions.
- Since openstack is for managing the cloud infrastructure to get a minimal setup, you need two machines: One will be the infrastructure you are managing, and one will be the manager. But if you are really strapped for hardware, you can fit both on a single machine.

Openstack Software and API:

- Openstack community has created a base set of software that you can use to get started tying out. This software called as DevStack, is primarily intended for testing and development purposes, and not for production. But it includes everything you need to get going, including management tools.
- The management tools are where things become a bit fuzzy between openstack being “just an API” and a set of software makes use of an API. Anyone can technically build a set of software that matches the openstack specification. That software can be either on the managed side or the manager side.
- The manages side would implement the API allowing any openstack-complaint management tools to manage it. The manager side would be a tool that can manage

any Openstack – compliant platform. The managed side is where Openstack mostly lives with its various APIs.

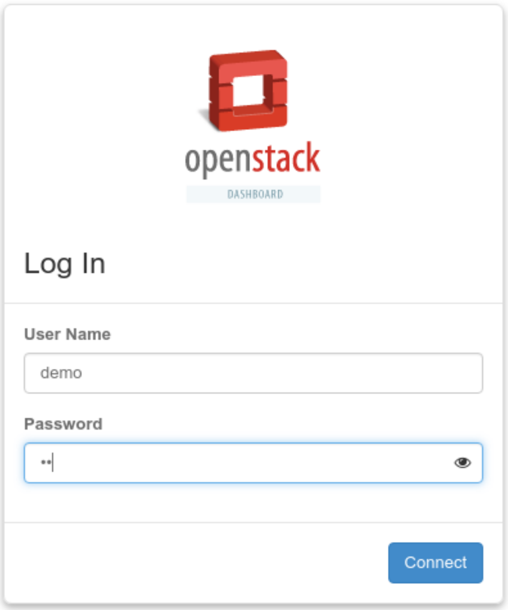
- There are several API's but here are a couple:
    - Compute: Compute is the main API for allocating and de-allocating servers. The code name for this API is NOVA. Openstack also allows you to create and manage images that represent backup of disk drives. This portion of openstack called Glance.
    - These images are often going to contain operating system such as Linux. The idea here is that you can choose an image that you will use to create a new server.
    - The image might contain for example, an Ubuntu 18.04 server that is already configured with the software you need. Then you would use the compute server to launch a couple of server using that image.
    - Because each server starts from the same image, they will be identical and already configured with the software you placed on the image.

Up and Running:

- Knowing all this lets give it a shot. The steps here are small, but you'll want to keep in mind how these steps would scale to larger situation and the decision you would need to make. One important first decision in what service you want to use.
- Openstack provide a whole range of service beyond the compute and image APIs. As mention earlier. The second decision is how many hardware servers i.e bare metal servers you want to use as well as how many virtual machine you want to allow each bare metal server to run. And finally you all want to put together a plan where users have limits or quotas on the amount of virtual machine and drive space they can use.
- For example you might install Ubuntu 18.04 in vmware, but to make this practice session as simple as possible, if you want you can install a desktop version of Ubuntu instead of the server version and then run the Horizon console right on that same machine. As an alternative you can instead create a new server on a cloud hosting service and install Ubuntu on it.
- Next, you'll need to install git. You don't need to know how to actually use git; it's just used here to get the latest version of the DevStack software. Now create a directory to house your OpenStack files. Switch to that directory and paste the following command into the console:
  ```
  git clone https://git.openstack.org/openstack-dev/devstack
  ```
- This will create a subdirectory called devstack. Switch to the new devstack, and then switch to the samples directory under it, like so:
  ```
  cd devstack/samples
  ```

- This directory contains two sample configuration files. (Check out OpenStack’s online documentation page about these configuration files.) Copy these up to the parent devstack directory:
  ```
  cp local* ..
  ```
- Now move back up to the parent devstack directory:
  ```
  cd ..
  ```
- Next, you need to make a quick modification to the local.conf file that you just copied over. Specifically you need to add your machine’s IP address within the local network. (It will likely start with a 10.) Open up local.conf using your favorite editor and uncomment (i.e. remove the #) the line that looks like this:
  ```
  #HOST_IP=w.x.y.z
  ```
- and replace w.x.y.z with your IP address. Here’s what mine looks like:
  ```
  HOST_IP=10.128.56.9
  ```
- (If you’re installing OpenStack, you probably know how to find your ipaddress. I used the ifconfig program.)
- Now run the setup program by typing:
  ```
  ./stack.sh
  ```
- If you watch, you’ll see several apt-get installations taking place followed by overall configurations. This process will take several minutes to complete. At times it will pause for a moment; just be patient and wait. Eventually you’ll see a message that the software is configured, and you’ll be shown a URL and two usernames (admin and demo) and a password (nomoresecrete).
- Note, however, that when I first tried this, I didn’t see that friendly message, unfortunately. Instead, I saw this message:
- “Could not determine a suitable URL for the plugin.”
- Thankfully, somebody posted a message online after which somebody else provided a solution. If you encounter this problem, here’s what you need to do. Open the stack.sh file and search for the text OS_USER_DOMAIN_ID. You’ll find this line:
  ```
  export OS_USER_DOMAIN_ID=default
  ```
- and then comment it out by putting a # in front of it:
  ```
  #export OS_USER_DOMAIN_ID=default
  ```
- Then a few lines down you’ll find this line:
  ```
  export OS_PROJECT_DOMAIN_ID=default
  ```
- which you’ll similarly comment out:
  ```
  #export OS_PROJECT_DOMAIN_ID=default
  ```
- Now you can try again. (I encourage you at this point to read the aforementioned post and learn more about why this occurred.) Then to start over you’ll need to run the unstack script:
  ```
  ./unstack.sh
  ```
- And then run stack.sh again:
  ```
  ./stack.sh
  ```

- Finally, when this is all done, you'll be able to log into the Horizon dashboard from the web browser using the URL printed out at the end of the script. It should just be the address of your virtual machine followed by dashboard, and really you should be able to get to it just using localhost:
  http://localhost/dashboard
- Also, depending on how you've set up your virtual machine, you can log in externally from your host machine. Otherwise, log into the desktop environment on the virtual machine and launch the browser.
- You'll see the login page like so:


openstack
DASHBOARD
Log In
User Name
demo
Password
Connect