Overview of OpenStack

Module I

https://www.malasuk.com/assets/eBooks/masteringopenstack.pdf

Ms. Navyamol K T

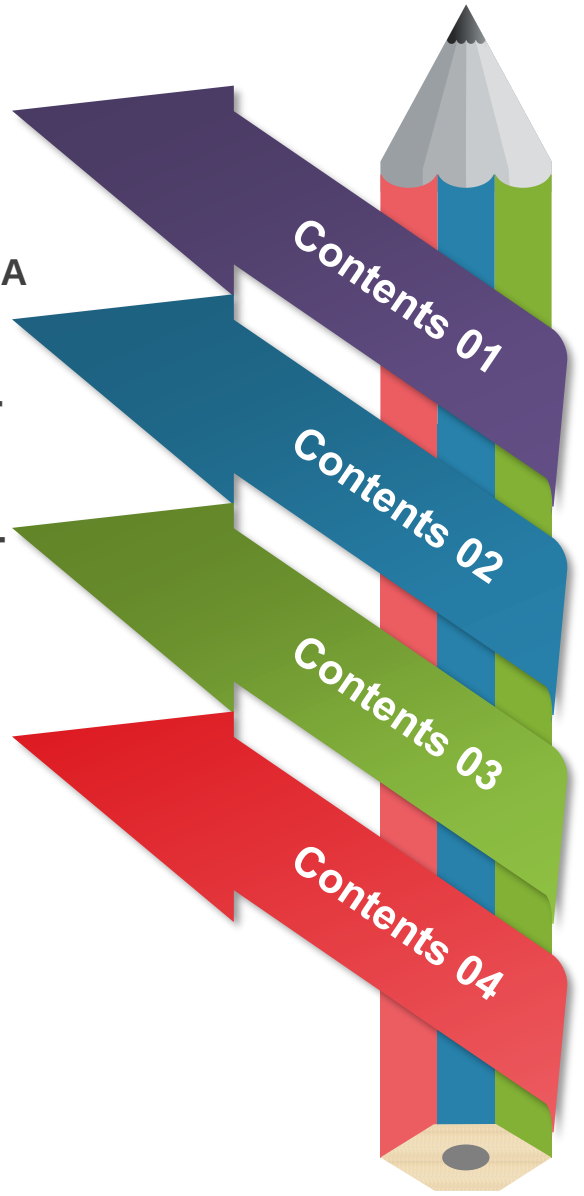
Asst. Professor

Dept. of MCA

AJCE

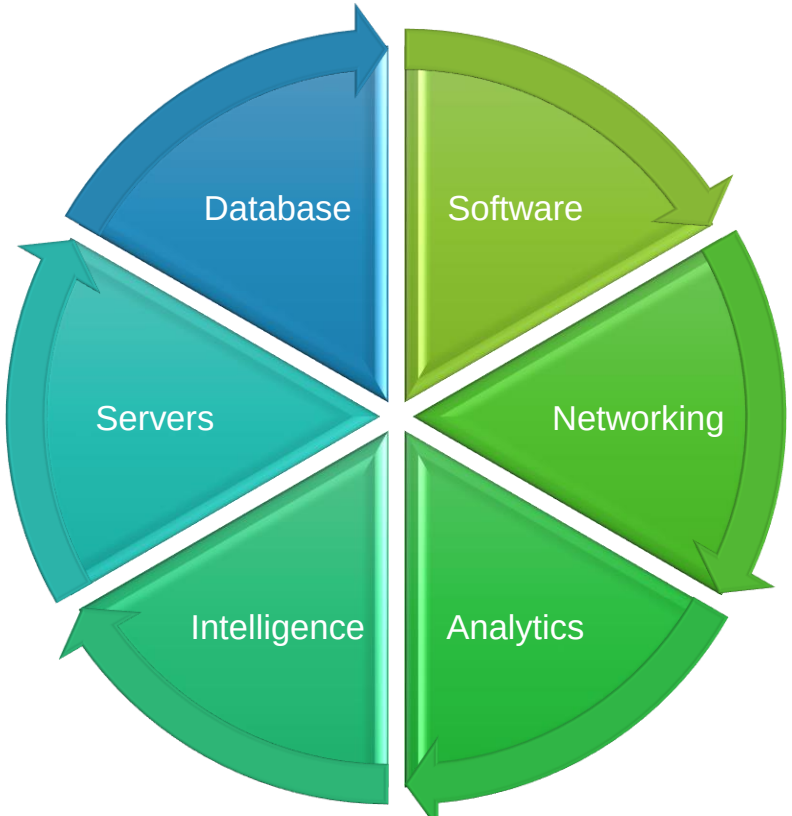
CONTENT

- INTRODUCTION TO CLOUD COMPUTING, PRIVATE CLOUD, PUBLIC CLOUD, HYBRID CLOUD ARCHITECTURE.
- CLOUD SERVICES – INFRASTRUCTURE AS A SERVICE, PLATFORM AS A SERVICE, STORAGE AS A SERVICE
- DESIGINING OPENSTACK CLOUD ARCHITECTURAL CONSIDERATION – OPENSTACK – THE NEW DATA CENTRE PARADIGM
- OPENSTACK LOGICAL ARCHITECTURE – NOVA – COMPUTE SERVICE – NEUTRON – NETWORKING SERVICES
- GATHERING THE PIECES AND BUILDING A PICTURE – A SAMPLE ARCHITECTURE SETUP.

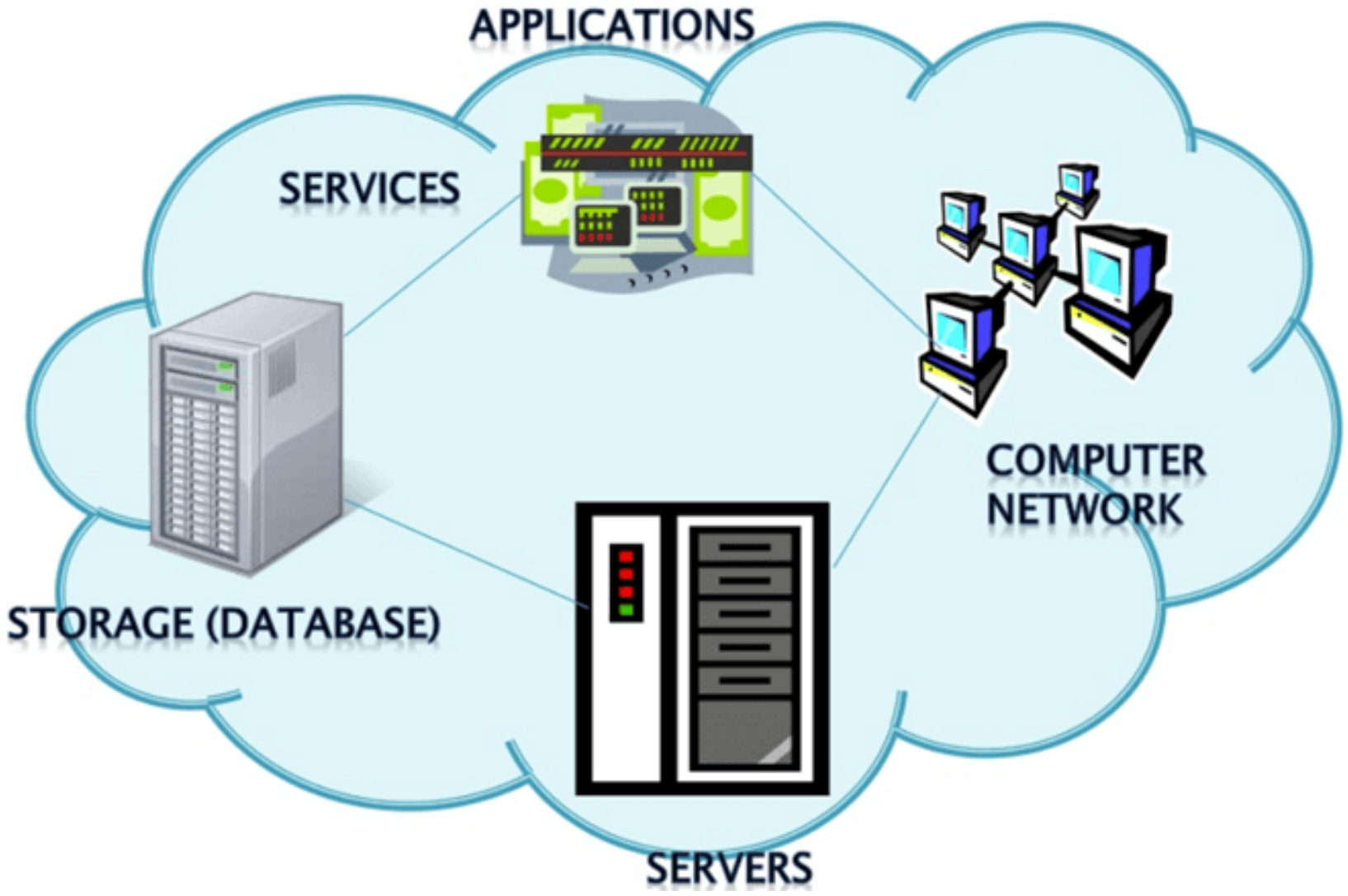


INTRODUCTION TO CLOUD COMPUTING

- Storing and manipulating programs and data over internet instead of hard disk of your computer.
- That is remote database via cloud based storage.
- Delivery of computing services over internet or cloud



INTRODUCTION TO CLOUD COMPUTING

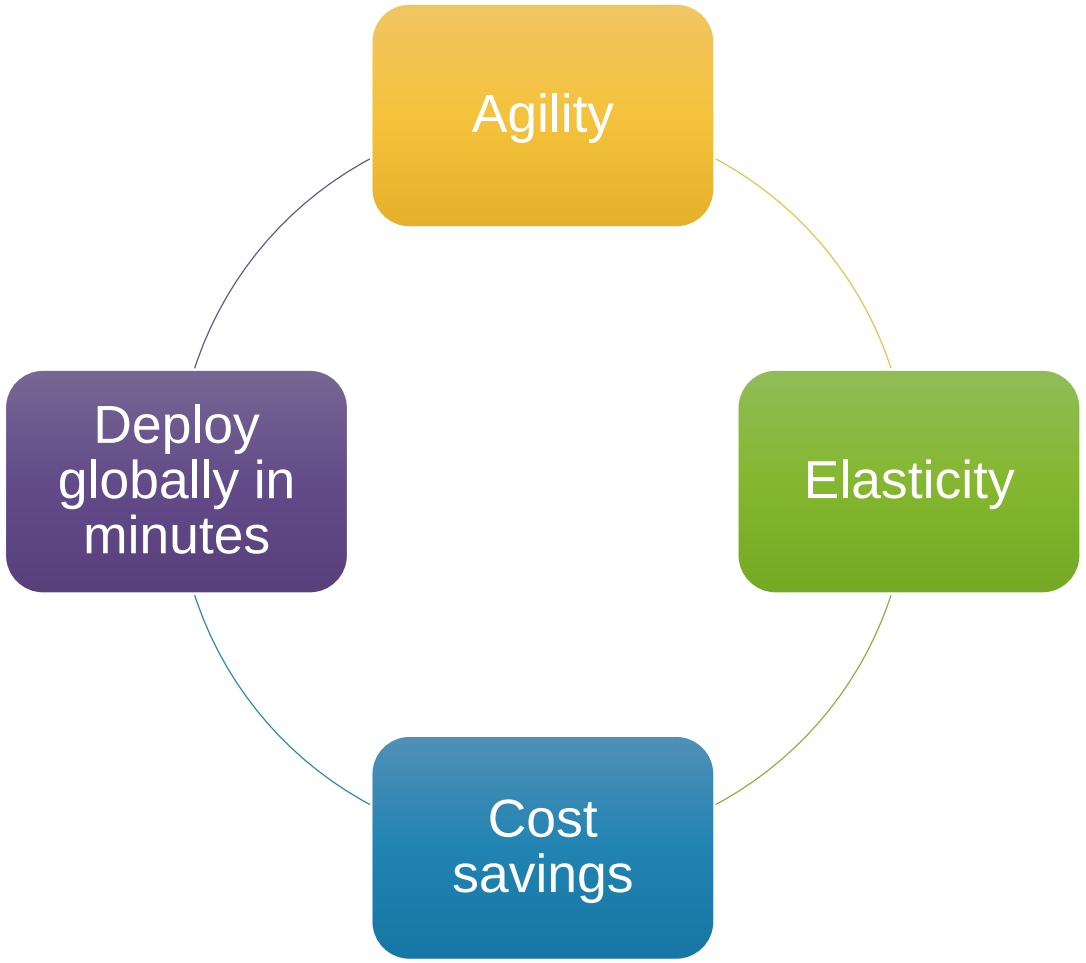


INTRODUCTION TO CLOUD COMPUTING

- Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing.
- Instead of buying, owning, and maintaining **physical data centers and servers**, you can **access technology services**, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).
- **Who is using cloud computing**
- Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications.
  - For example, **healthcare companies** are using the cloud to develop more personalized treatments for patients.
  - **Financial services** companies are using the cloud to power real-time fraud detection and prevention.
  - **Video game makers** are using the cloud to deliver online games to millions of players around the world.

INTRODUCTION TO CLOUD COMPUTING

- Benefits of Cloud Computing



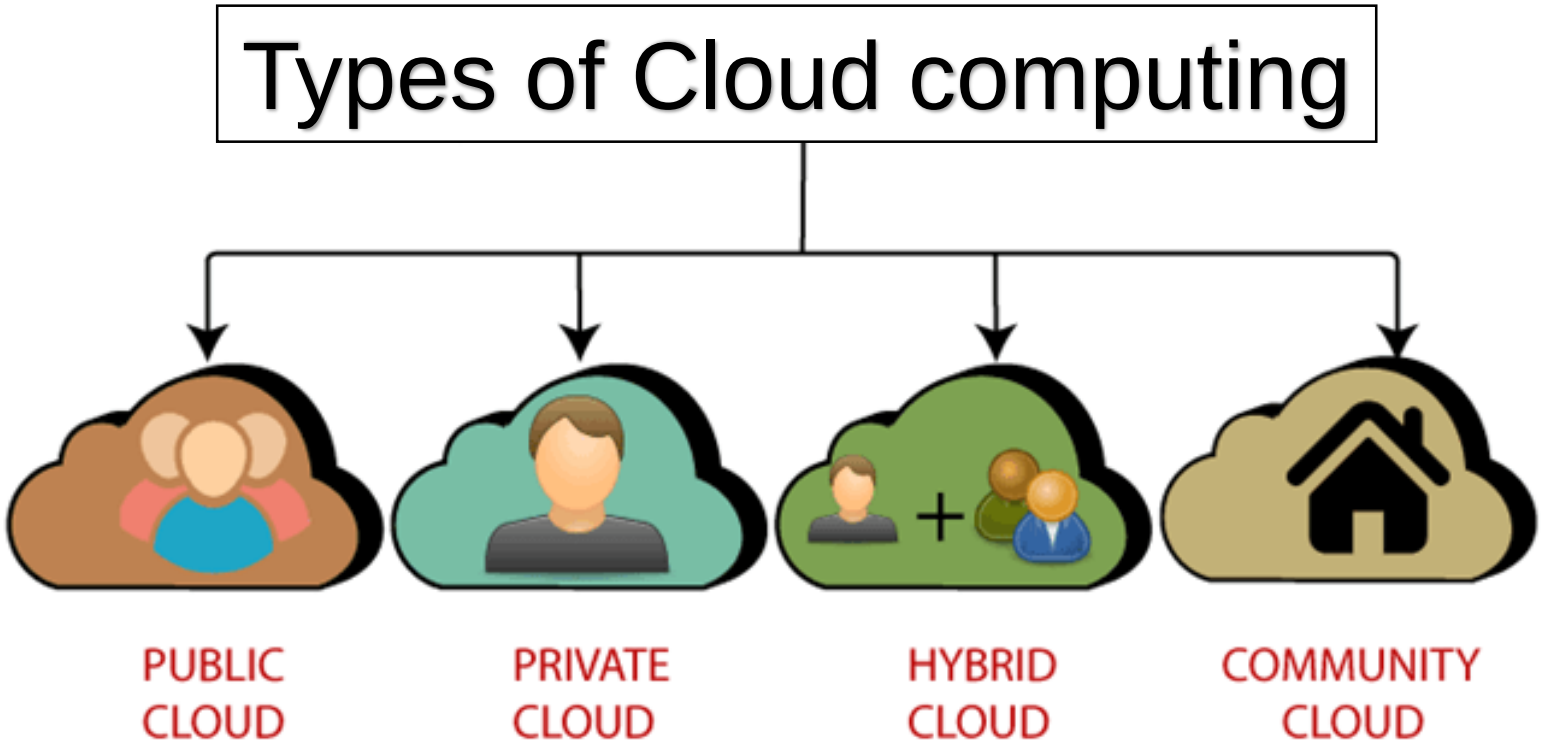
INTRODUCTION TO CLOUD COMPUTING

- Benefits of Cloud Computing
  - Agility
  - The cloud gives you easy access to a broad range of technologies so that you can innovate faster and build nearly anything that you can imagine. You can quickly spin up resources as you need them–from infrastructure services, such as compute, storage, and databases, to Internet of Things, machine learning, data lakes and analytics, and much more.
  - Elasticity
  - With cloud computing, you don't have to over-provision resources up front to handle peak levels of business activity in the future. Instead, you provision the amount of resources that you actually need. You can scale these resources up or down to instantly grow and shrink capacity as your business needs change

INTRODUCTION TO CLOUD COMPUTING

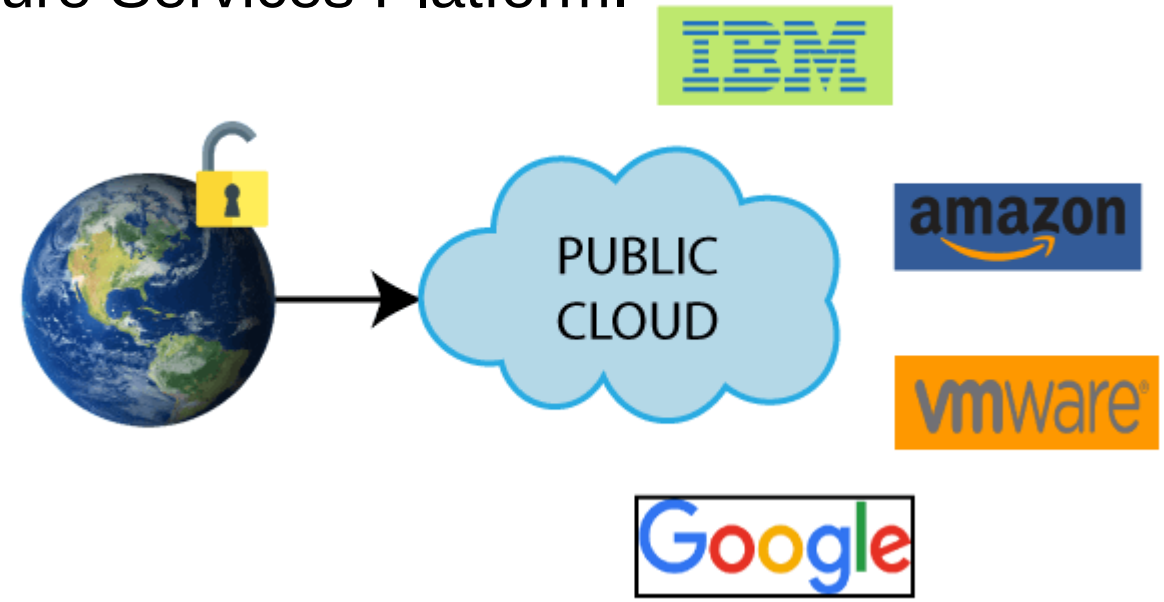
- Benefits of Cloud Computing
  - Cost savings
  - The cloud allows you to trade capital expenses (such as data centers and physical servers) for variable expenses, and only pay for IT as you consume it. Plus, the variable expenses are much lower than what you would pay to do it yourself because of the economies of scale.
  - Deploy globally in minutes
  - With the cloud, you can expand to new geographic regions and deploy globally in minutes. For example, AWS has infrastructure all over the world, so you can deploy your application in multiple physical locations with just a few clicks. Putting applications in closer proximity to end users reduces latency and improves their experience.

INTRODUCTION TO CLOUD COMPUTING



INTRODUCTION TO CLOUD COMPUTING

- Public cloud
  - Public cloud is open to all to store and access information via the Internet using the pay-per-usage method.
  - In public cloud, computing resources are managed and operated by the Cloud Service Provider (CSP).
  - Example: Microsoft Azure, GCP, Alibaba, Oracle, Amazon elastic compute cloud (EC2), IBM SmartCloud Enterprise, Microsoft, Google App Engine, Windows Azure Services Platform.



Public cloud

- The public cloud is defined as computing services **offered by third-party providers over the public Internet**, making them available to anyone who wants to use or purchase them.
- Public cloud services may be free or offered through a variety of subscription or on-demand pricing schemes, including a pay-per-usage model.
- The main benefits of the public cloud are as follows:
- A reduced need for organizations to invest in and maintain their own on-premises IT resources;
- Scalability to meet workload and user demands;
- Fewer wasted resources because customers only pay for what they use.

PUBLIC CLOUD

- Public cloud is an alternative application development approach to traditional on-premises IT architectures.
- In the basic public cloud computing model, a third-party provider hosts scalable, on-demand IT resources and delivers them to users over a network connection, either over the public internet or a dedicated network.
- The public cloud model encompasses many different technologies, capabilities and features.

PUBLIC CLOUD

A public cloud consists of the following key characteristics:

- On-demand Computing And Self-service Provisioning;
- Resource Pooling;
- Scalability And Rapid Elasticity;
- Pay-per Use Pricing;
- Measured
- Resiliency And Availability;
- Security;
- Broad Network Access

PUBLIC CLOUD

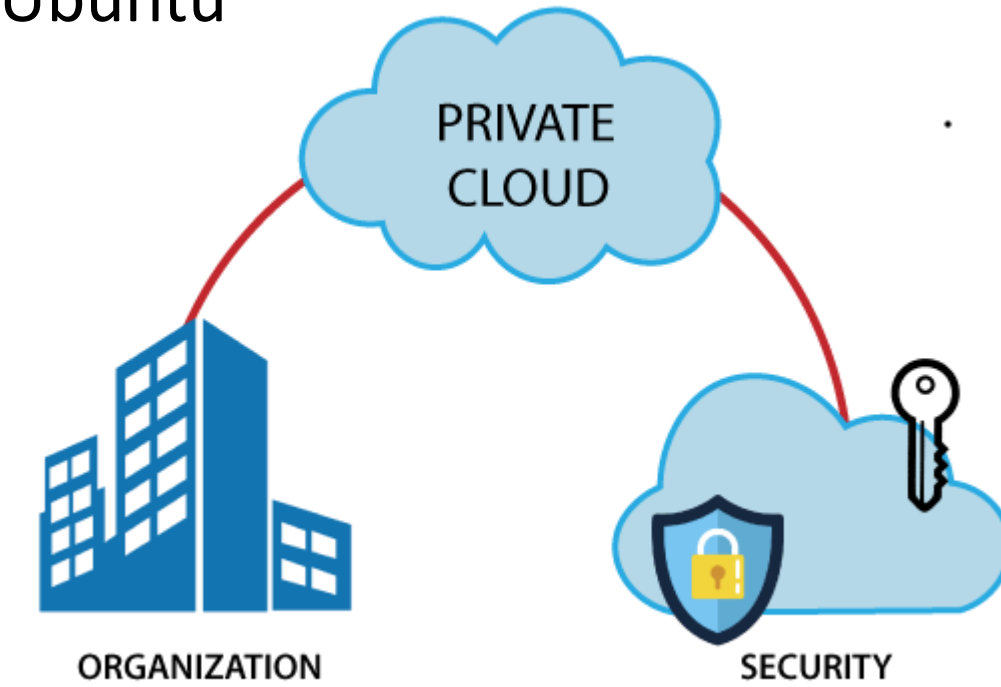
- The public cloud provider supplies the infrastructure needed to host and deploy workloads in the cloud.
- It also offers tools and services to help customers manage cloud applications, such as data storage, security and monitoring
- Unlike private clouds, public clouds can save companies from the expensive costs of having to purchase, manage and maintain on-premises hardware and application infrastructure - the cloud service provider is held responsible for all management and maintenance of the system.
- Public clouds can also be deployed faster than on-premises infrastructures and with an almost infinitely scalable platform

PUBLIC CLOUD

- When selecting a provider, organizations can opt for a large, general-use provider -- such as AWS, Microsoft Azure or Google Cloud Platform (GCP) -- or a smaller provider.
- General cloud providers offer broad availability and integration options and are desirable for multipurpose cloud needs.

PRIVATE CLOUD

- Private cloud is also known as an internal cloud or corporate cloud.
- It is used by organizations to build and manage their own data centers internally or by the third party.
- It can be deployed using Opensource tools such as Openstack and Eucalyptus.
- Example: HP Data Centers, Microsoft, Elastra, Ubuntu



PRIVATE CLOUD

- Private cloud is a cloud computing environment dedicated to a single customer.
- Private cloud (also known as an internal cloud or corporate cloud) is a cloud computing environment in which all hardware and software resources are dedicated exclusively to, and accessible only by, a single customer.
- Private cloud combines many of the benefits of cloud computing—including elasticity, scalability, and ease of service delivery—with the access control, security, and resource customization of on-premises infrastructure.

PRIVATE CLOUD

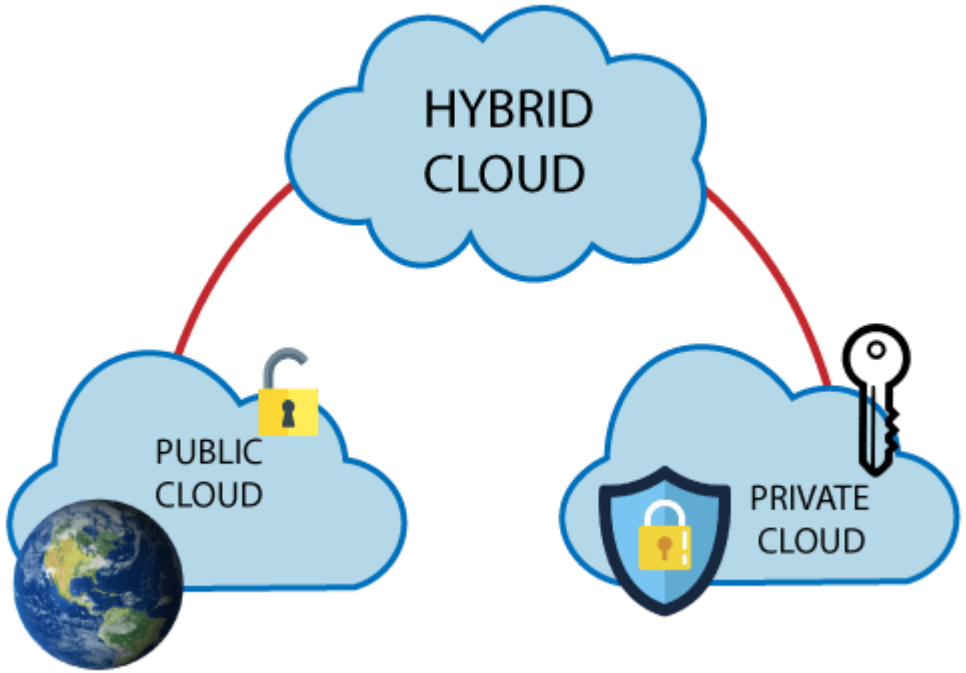
- Private cloud - These technologies include the following: –
- **Virtualization**, which enables IT resources to be abstracted from their underlying physical hardware and pooled into unbounded resource pools of computing, storage, memory, and networking capacity that can then portioned among multiple virtual machines (VMs), containers, or other virtualized IT infrastructure elements. –
- **Management software** gives administrators centralized control over the infrastructure and applications running on it. This makes it possible to optimize security, availability, and resource utilization in the private cloud environment.
- **Automation speeds tasks**—such as server provisioning and integrations—that would otherwise need to be performed manually and repeatedly. Automation reduces the need for human intervention, making self-service resource delivery possible

PRIVATE CLOUD

- **BENEFITS OF PRIVATE CLOUD**
- **Full control over hardware and software choices**. Private cloud customers are free to purchase the hardware and software they prefer, vs. the hardware and software the cloud provider offers.
- **Freedom to customize hardware and software** in any way. Private cloud customers can customize servers in any way they want and can customize software as needed with add-ons or through custom development.
- **Greater visibility into security and access control**, because all workloads run behind the customers' own firewall.
- **Fully enforced compliance with regulatory standards**. Private cloud customers aren't forced to rely on the industry and regulatory compliance offered by the cloud service provider.

HYBRID CLOUD

- Hybrid Cloud is a combination of the public cloud and the private cloud. we can say:
- Hybrid Cloud = Public Cloud + Private Cloud
- Example: Google Cloud Computing



HYBRID CLOUD

- Hybrid cloud refers to a mixed computing, storage, and services environment made up of on-premises infrastructure, private cloud services, and a public cloud—such as Amazon Web Services (AWS) or Microsoft Azure—with orchestration among the various platforms.
- Using a combination of public clouds, on-premises computing, and private clouds in your data center means that you have a hybrid cloud infrastructure
- The main aim to combine these cloud (Public and Private) is to create a unified, automated, and well-managed computing environment.
- In the Hybrid cloud, non-critical activities are performed by the public cloud and critical activities are performed by the private cloud.
- Mainly, a hybrid cloud is used in finance, healthcare, and Universities.

Advantages of Hybrid Cloud

- There are the following advantages of Hybrid Cloud
- Flexible and secure – It provides flexible resources because of the public cloud and secure resources because of the private cloud.
- Cost effective – Hybrid cloud costs less than the private cloud. It helps organizations to save costs for both infrastructure and application support. It offers the features of both the public as well as the private cloud. A hybrid cloud is capable of adapting to the demands that each company needs for space, memory, and system.

Disadvantages of Hybrid Cloud

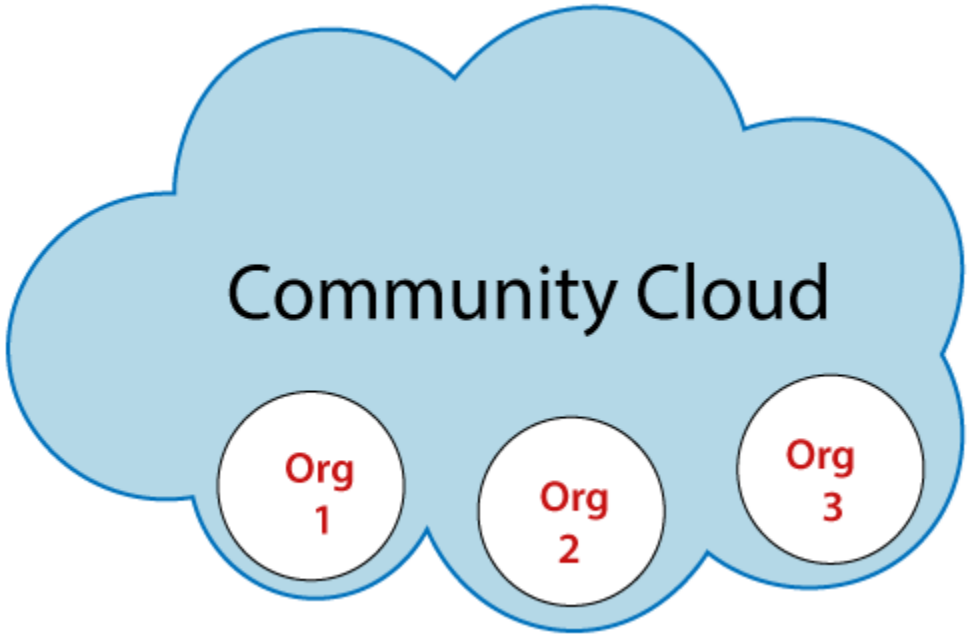
- Networking issues – In the Hybrid Cloud, networking becomes complex because of the private and the public cloud.
- Infrastructure Compatibility – Infrastructure compatibility is the major issue in a hybrid cloud. With dual-levels of infrastructure, a private cloud controls the company, and a public cloud does not, so there is a possibility that they are running in separate stacks.
- Reliability – The reliability of the services depends on cloud service providers.

Hybrid Cloud Scenario

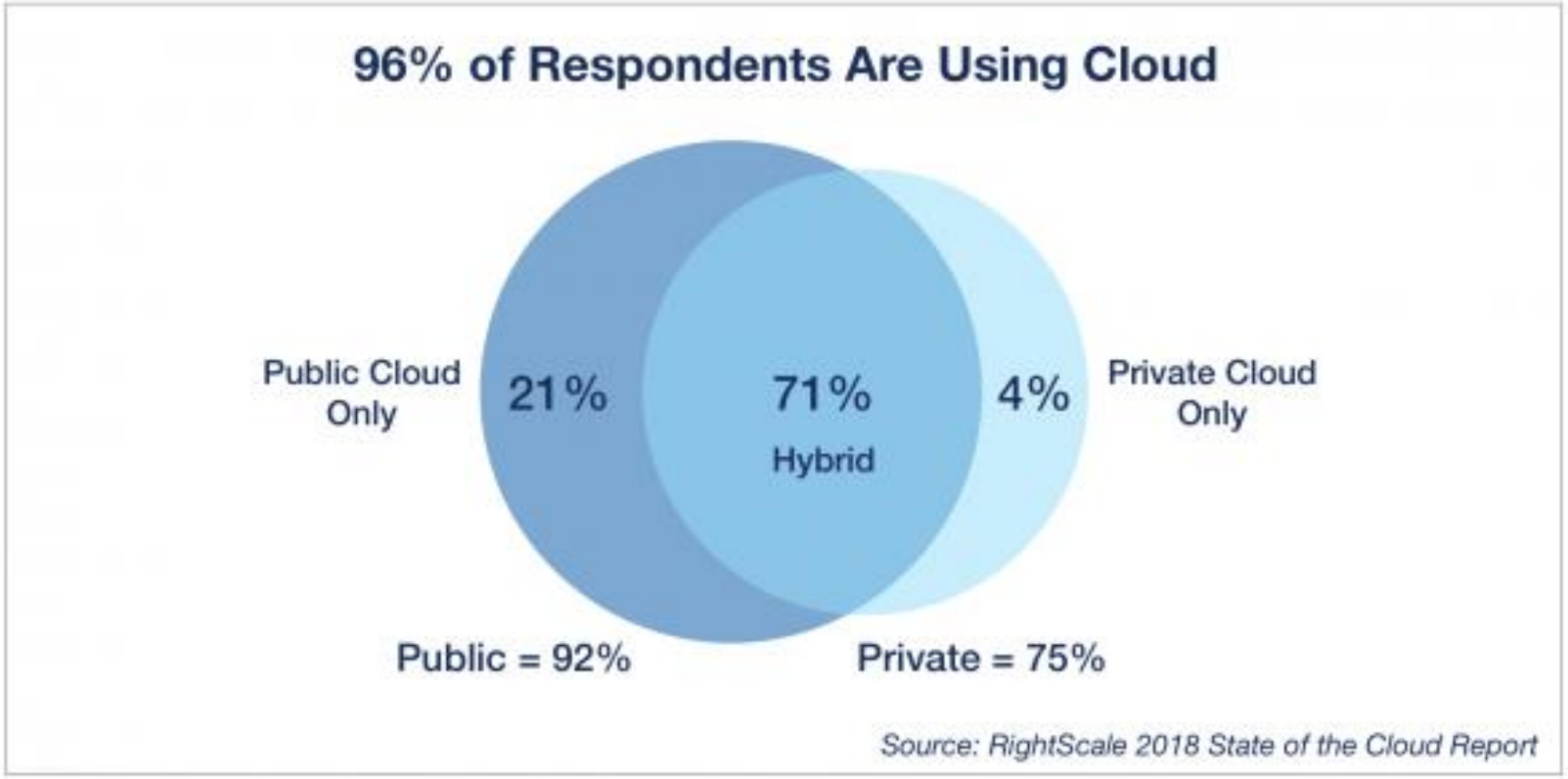
- Dynamic or frequently changing workloads.
- Separating critical workloads from less-sensitive workloads.
- Big data processing.
- Moving to the cloud incrementally, at your own pace
- Temporary processing capacity needs.
- Flexibility for the future
- Best of both worlds

INTRODUCTION TO CLOUD COMPUTING

- Community Cloud
  - Community cloud allows systems and services to be accessible by a group of several organizations to share the information between the organization and a specific community.
  - It is owned, managed, and operated by one or more organizations in the community, a third party, or a combination of them.
  - Government organizations in India



INTRODUCTION TO CLOUD COMPUTING



Cloud Computing Models

There are three main models for cloud computing. Each model represents a different part of the cloud computing stack.

Infrastructure as a Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

Platform as a Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.

INTRODUCTION TO CLOUD COMPUTING

- Three major forms of cloud computing exist.

Infrastructure as a Service (IaaS):

- It is the basic cloud service that offers networking services, load balancers, virtual machines, and firewalls services.
- It includes a method of providing everything from OS to servers and storage via IP-based networking as part of an on-demand service.
- Some common examples of IaaS are IBM cloud, AWS, and Microsoft Azure.

INTRODUCTION TO CLOUD COMPUTING

- Three major forms of cloud computing exist.

Platform as a Service (PaaS):

- A cloud computing service that offers an on-demand platform for software application development, management, testing, and distribution.
- If you use PaaS services, then you don't have to worry about setting up or maintaining the underlying server, network, storage, and database infrastructure required for the development.
- Example of PaaS is Google App Engine, Salesforce.com, etc.

INTRODUCTION TO CLOUD COMPUTING

- Three major forms of cloud computing exist.

Software as a Service (SaaS):

- It is a distribution model. Through this service, computer applications (web services) are distributed over the Internet.
- Users may use a computer or mobile device that has internet connectivity to access SaaS services.
- The most common example of a SaaS is Microsoft Office 365, which provides productivity and email services.

INTRODUCTION TO CLOUD COMPUTING

- Three major forms of cloud computing exist.

Storage as a Service (STaaS):

- Storage as a Service (STaaS) is the practice of using public cloud storage resources to store your data.
- Using STaaS is more cost efficient than building private storage infrastructure, especially when you can match data types to cloud storage offerings.

Cloud Block Storage

Cloud Object Storage

Cloud File Storage

INTRODUCTION TO CLOUD COMPUTING

Advantages of STaaS

- Key advantages to STaaS in the enterprise include the following:
- **Storage costs**. Personnel, hardware and physical storage space expenses are reduced.
- **Disaster recovery**. Having multiple copies of data stored in different locations can better enable disaster recovery measures.
- **Scalability**. With most public cloud services, users only pay for the resources that they use.
- **Syncing**. Files can be automatically synced across multiple devices.
- **Security.** Security can be both an advantage and a disadvantage, as security methods may change per vendor. Data tends to be encrypted during transmission and while at rest.

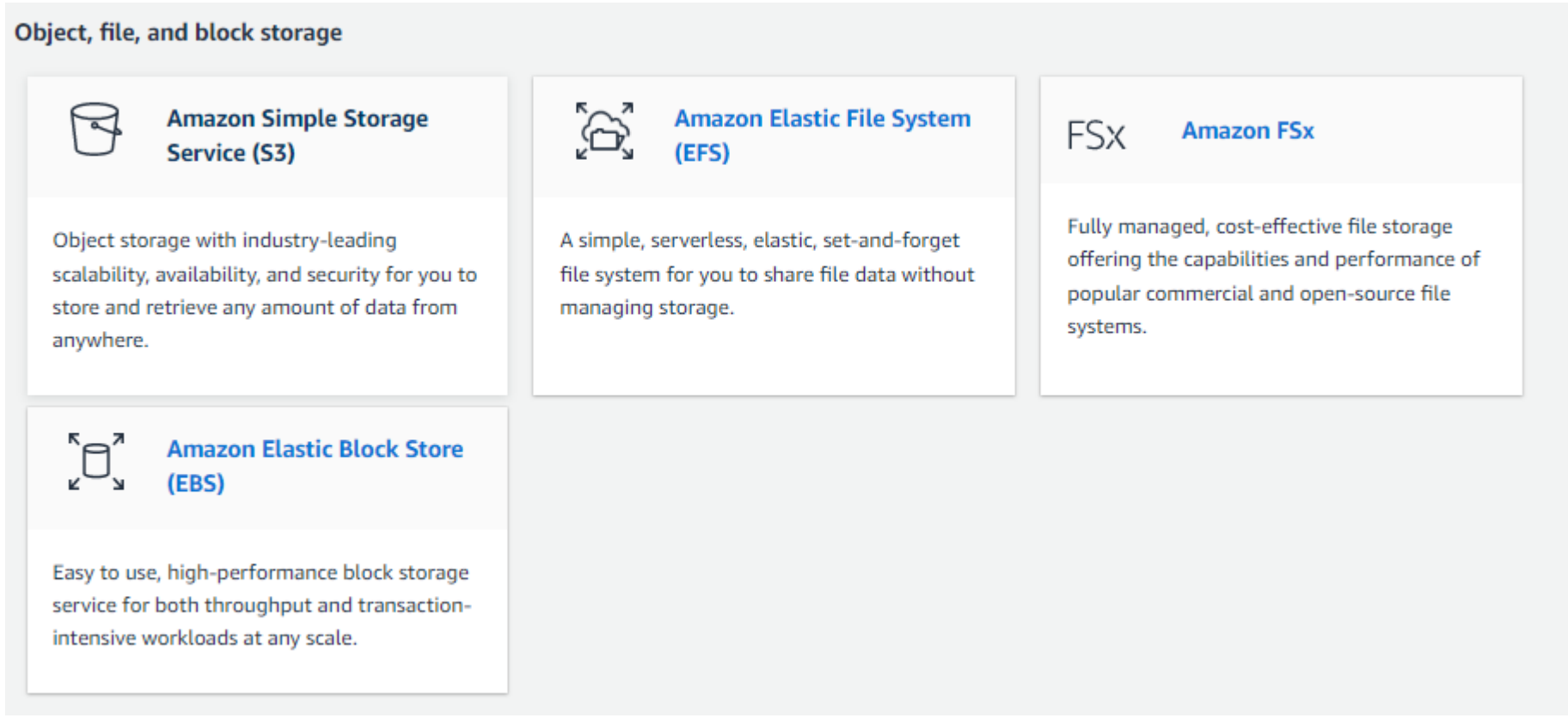
Cloud Block Storage

Cloud Object Storage

Cloud File Storage

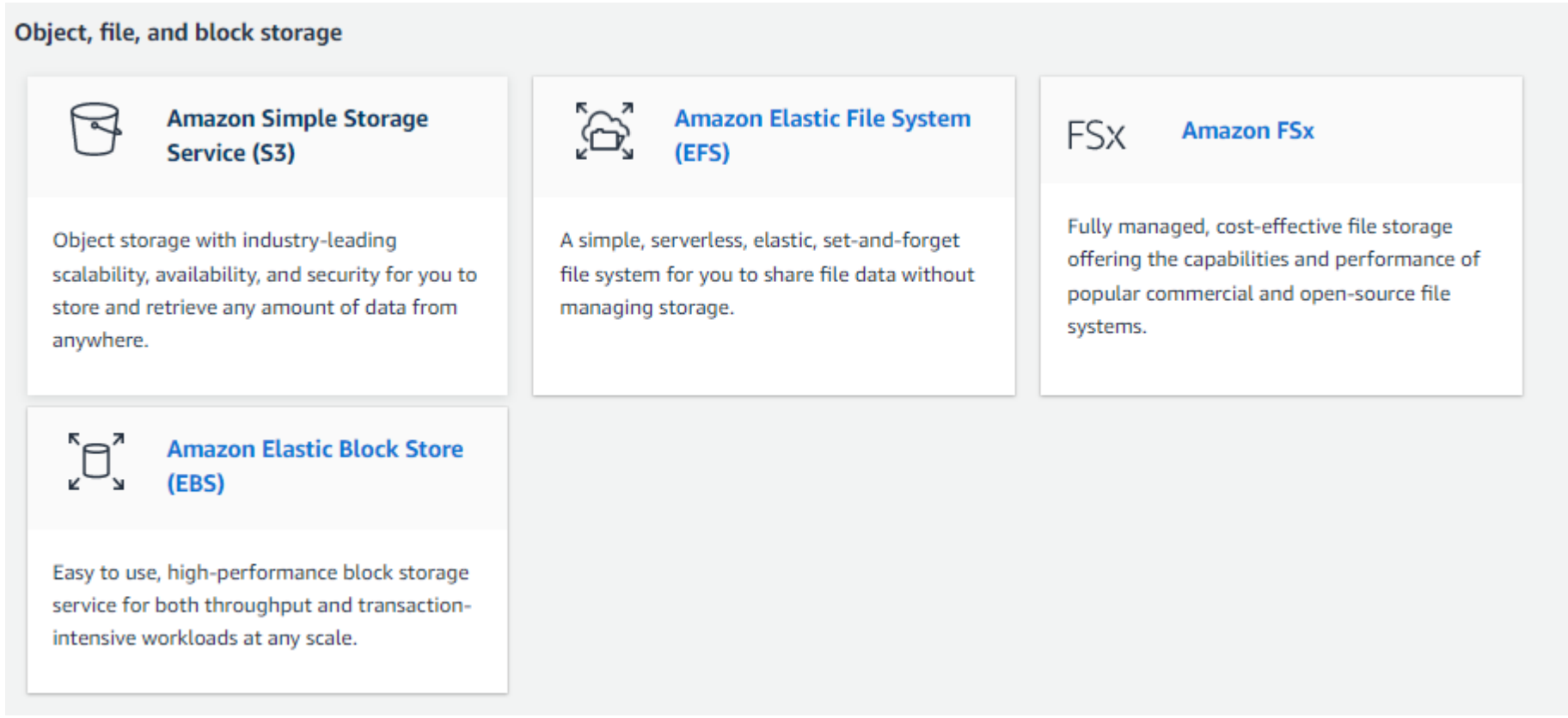
INTRODUCTION TO CLOUD COMPUTING

- STaaS in AWS



INTRODUCTION TO CLOUD COMPUTING

- Example STaaS in AWS



Difference Between IAAS, PAAS and SAAS

Parameters	IAAS	PAAS	SAAS
Full-Form	IaaS is an acronym for Infrastructure As A Service.	PaaS is an acronym for Platform As A Service.	SaaS is an acronym for Software As A Service.
Access	The IaaS service provides its users with access to various resources like virtual storage and virtual machines.	Using the PaaS services, users can get access to a runtime environment (for the development and deployment of applications and tools).	The SaaS services give access to all of their services to the end-users, where it's application hosting, storage, or any other services.
Technical Understanding	A user requires technical knowledge to make use of IaaS services.	One must acquire the basic knowledge of the concerned subjects to understand the setup of the PaaS services.	You don't need to know any technicalities to understand and use the SaaS services- the service provider can handle everything.

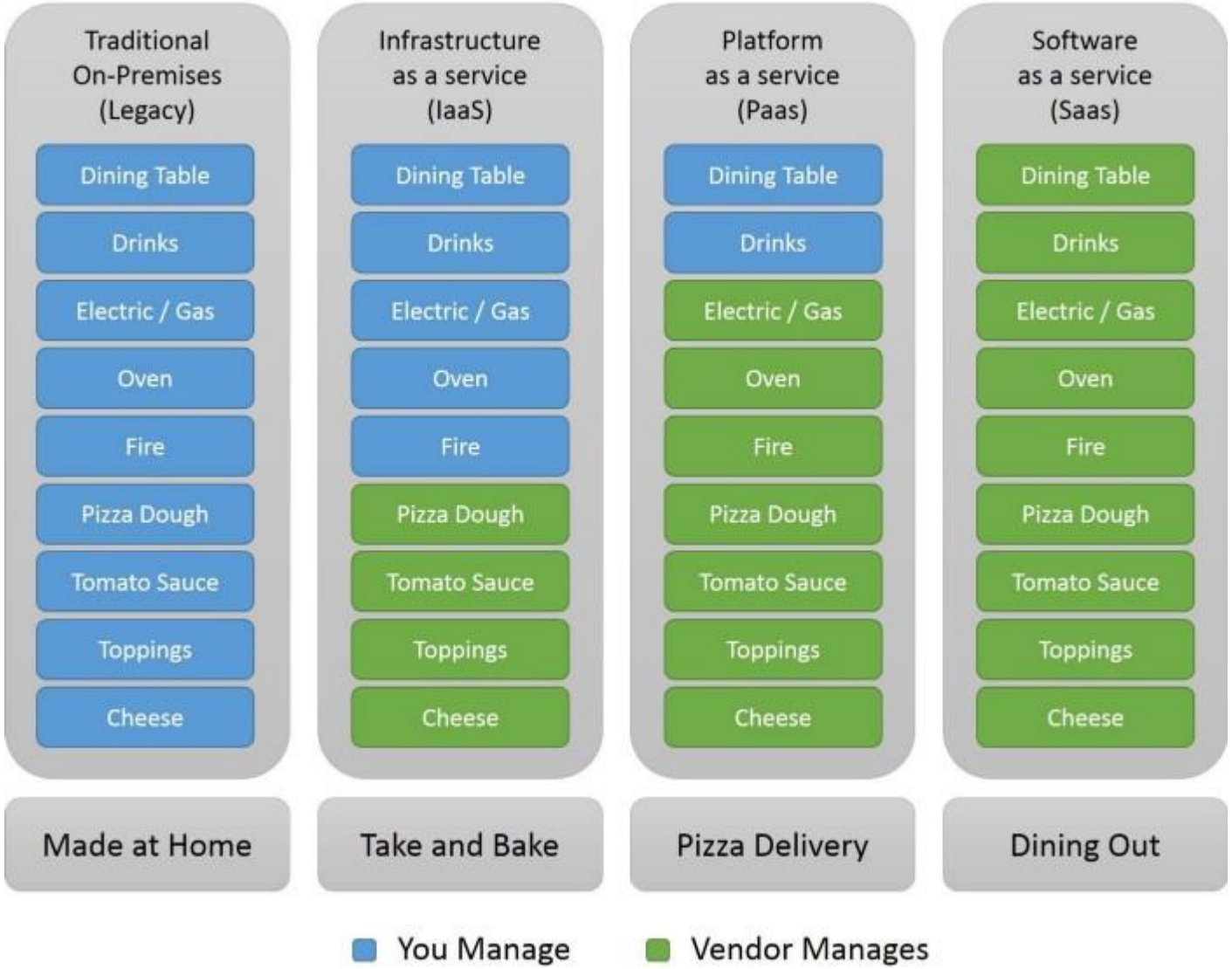
Difference Between IAAS, PAAS and SAAS

Parameters	IAAS	PAAS	SAAS
Used By	The network architects primarily use the IaaS.	Developers mainly make use of PaaS.	An end-user generally uses SaaS.
Model	The IaaS is a service model. It functions to provide various visualized computing resources all over the internet.	PaaS is a cloud computing model. It mainly delivers the tools required for developing various applications.	SaaS is a service model for cloud computing services. They mainly host various software and make them available for the clients.
Popularity	IaaS is very popular among researchers and developers.	PaaS is very common among developers who mainly focus on app and script development.	The SaaS services are very common among consumers and companies for networking, sharing emails, files, etc.

Difference Between IAAS, PAAS and SAAS

Parameters	IAAS	PAAS	SAAS
Cloud Services	VCloud Express, Sun, Amazon Web Services.	Google and Facebook (and other search engines).	Google and Facebook apps, MS Office Web.
Enterprise Services	Virtual Private Cloud by AWS.	MS Azure.	Cloud Analysis from IBM.
Outsourced form of Cloud Services	Salesforce.	Gigaspaces, Force.com.	Terremark, AWS.

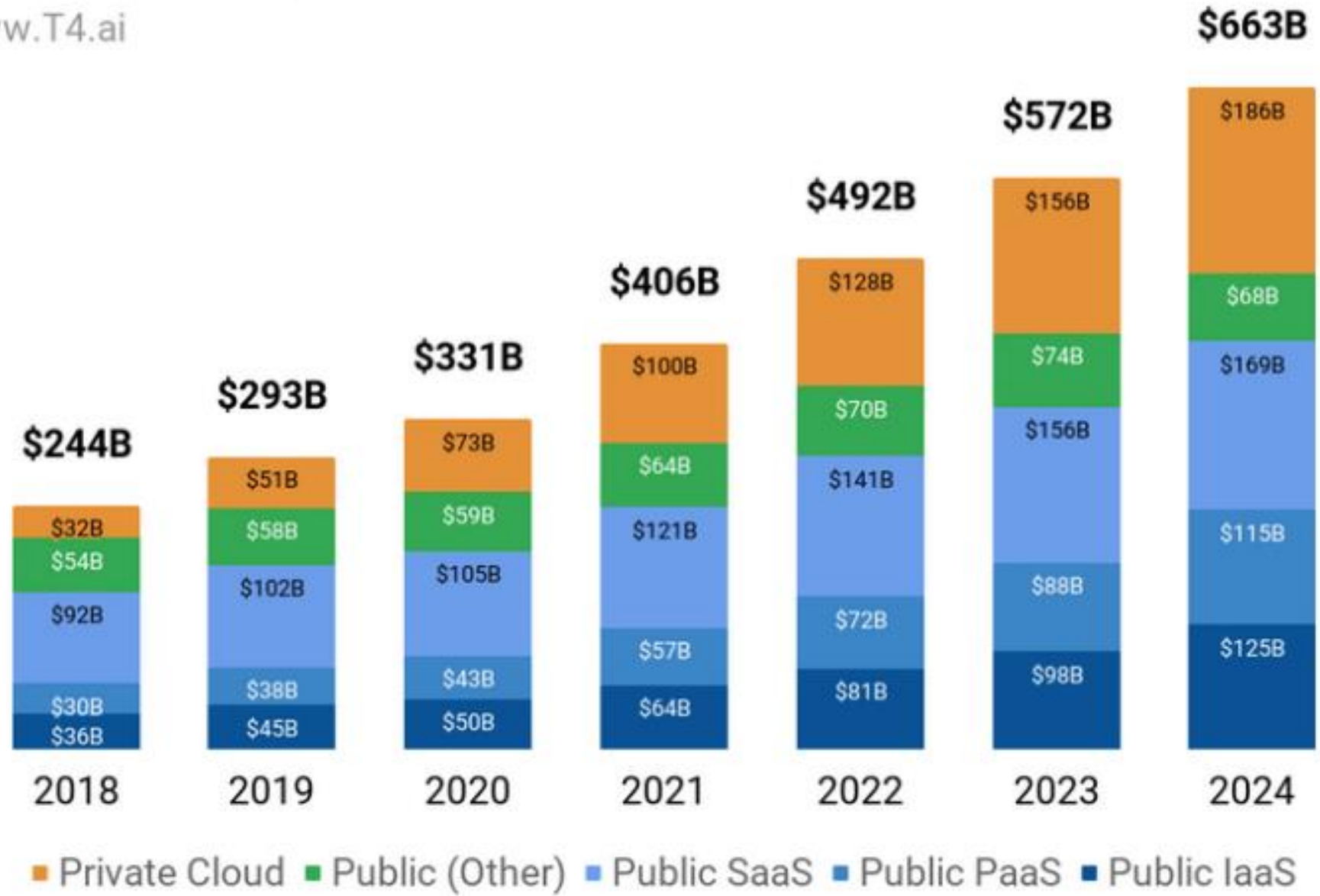
Pizza as a Service



Click on the image

Cloud Computing Market Size, 2018-2024

www.T4.ai



OPENSTACK LOGICAL ARCHITECTURE

9 Components

OpenStack Architecture

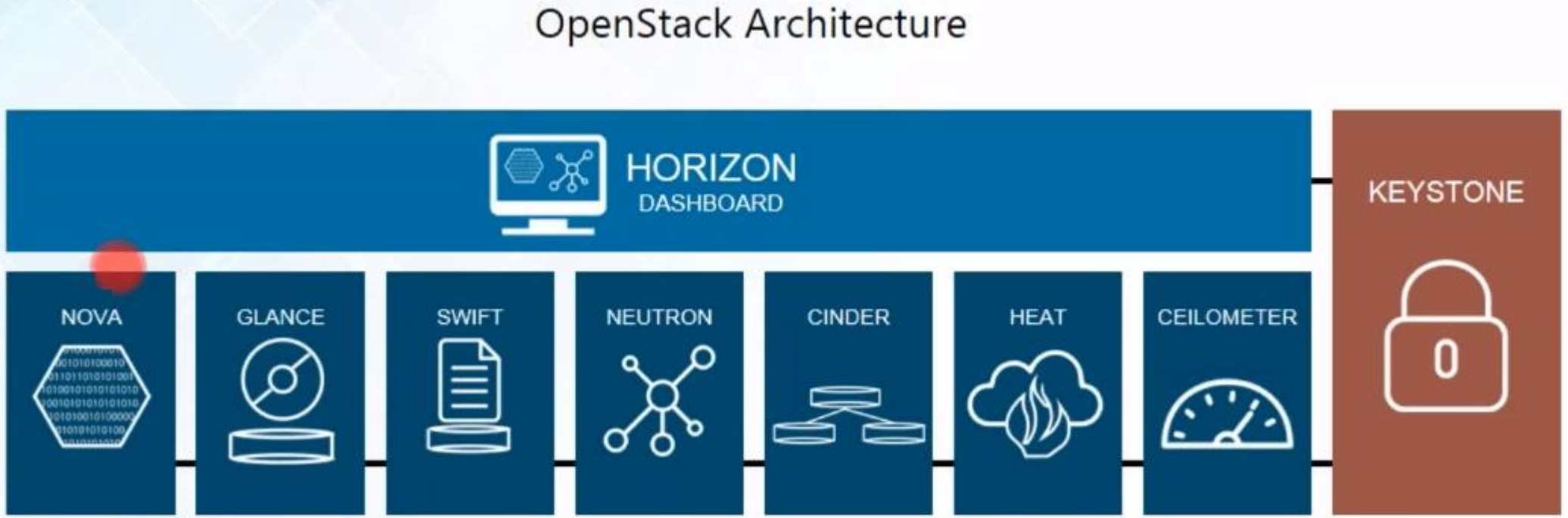
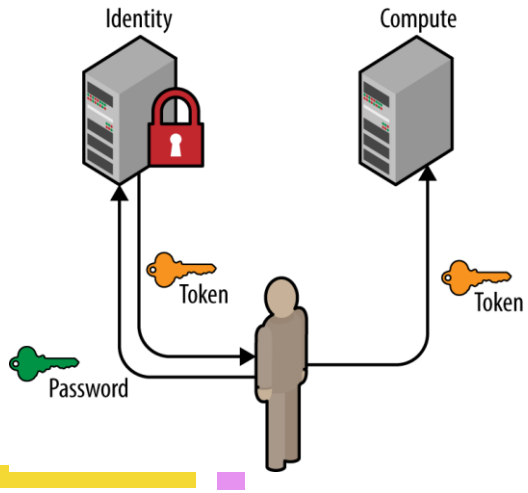


Figure: *OpenStack Architecture*

Keystone - identity management

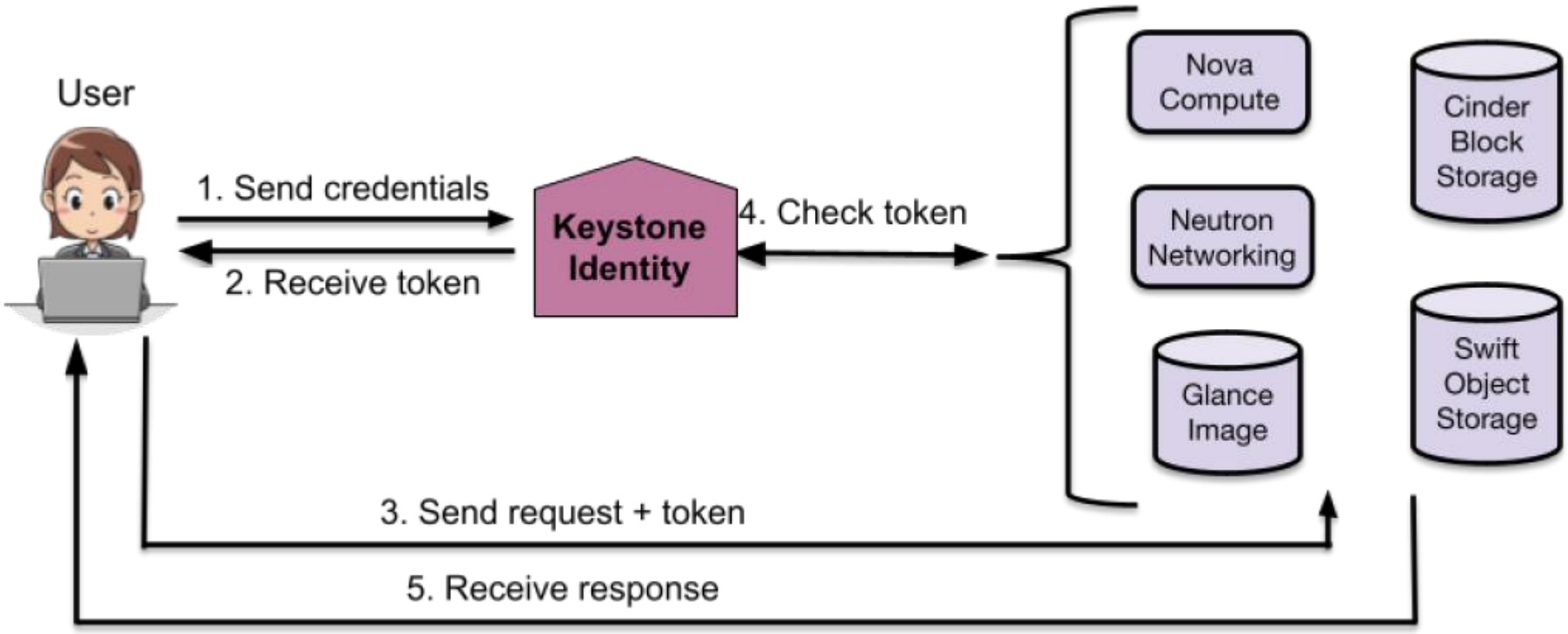
- From an architectural perspective, Keystone presents the simplest service in the OpenStack composition.
- It is the core component and provides an identity service comprising authentication and authorization of tenants in OpenStack.
- Communications between different OpenStack services are authorized by Keystone to ensure that the right user or service is able to utilize the requested OpenStack service.
- Keystone integrates with numerous authentication mechanisms such as username/password and token/authentication-based systems.

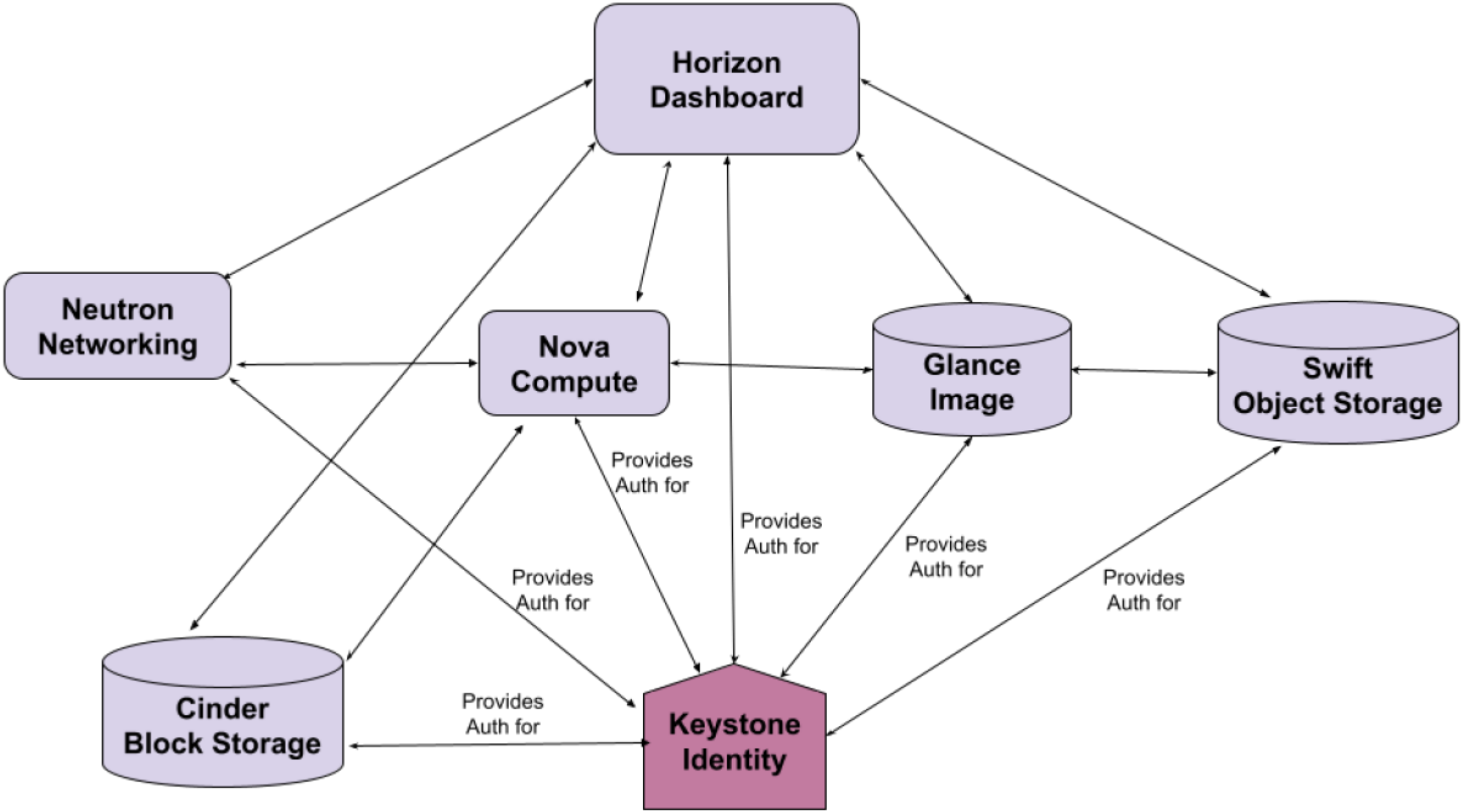


Keystone - identity management

- It is the central repository of all users and their privileges for the services they are using for OpenStack.
- This component is used to manage identity services like authorization, authentication, AWS Styles (Amazon Web Services) logins, token-based systems, and checking the other credentials (username & password).
- There are the following functions which usually perform by Keystone:
  - Monitoring users and their permissions
  - Providing a list of available resources with their API endpoints.

Keystone - identity management



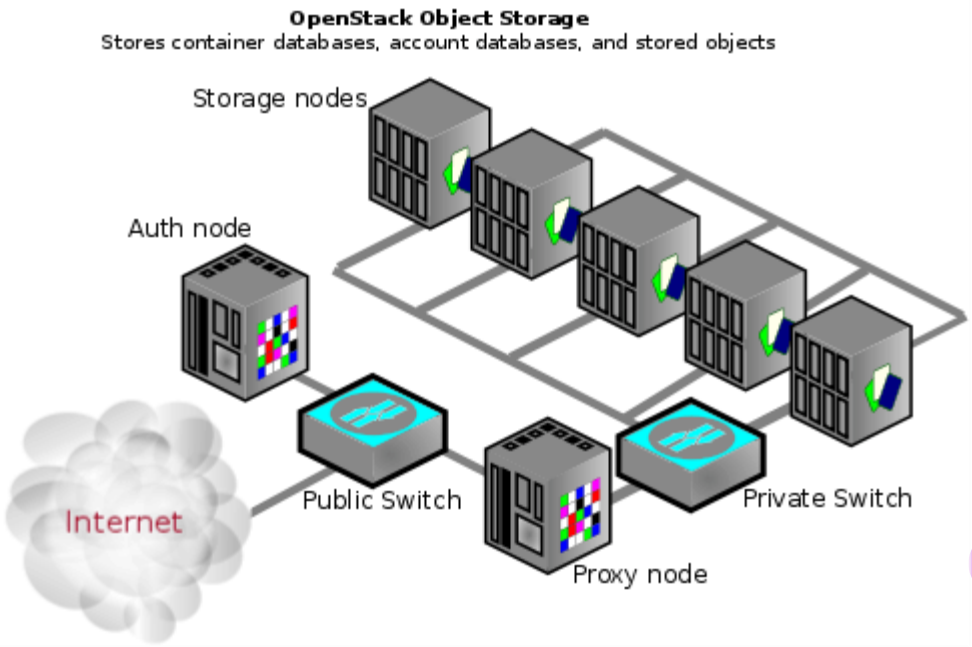

Horizon Dashboard
Neutron Networking
Nova Compute
Glance Image
Swift Object Storage
Cinder Block Storage
Keystone Identity
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for

Swift - object storage

- Swift is one of the storage services available to OpenStack users.
- It provides an object-based storage service and is accessible through REST APIs.
- Compared to traditional storage solutions, file shares, or block -based access, an Object-Storage takes the approach of dealing with stored data as objects that can be stored and retrieved from the Object-Store.
- A very high-level overview of Object Storage goes like this. To store the data, the Object-Store splits it into smaller chunks and stores it in separate containers.
- These containers are maintained in redundant copies spread across a cluster of storage nodes to provide high availability, auto-recovery, and horizontal scalability.
- Swift has a number of benefits: –
- It has no central brain, and indicates no Single Point Of Failure (SPOF)
- It is curative, and indicates auto-recovery in the case of failure.
- It is highly scalable for large petabytes of storage access by scaling horizontally It has a better performance, which is achieved by spreading the load over the storage nodes.
- It has inexpensive hardware that can be used for redundant storage clusters

Swift - object storage

- Object storage is used in order to store and recover arbitrary data in the cloud.
- In Swift, it is possible to store the files, objects, backups, images, videos, virtual machines, and other unstructured data.
- Developers may use a special identifier for referring the file and objects in place of the path, which directly points to a file and allows the OpenStack to manage where to store the files to the API.

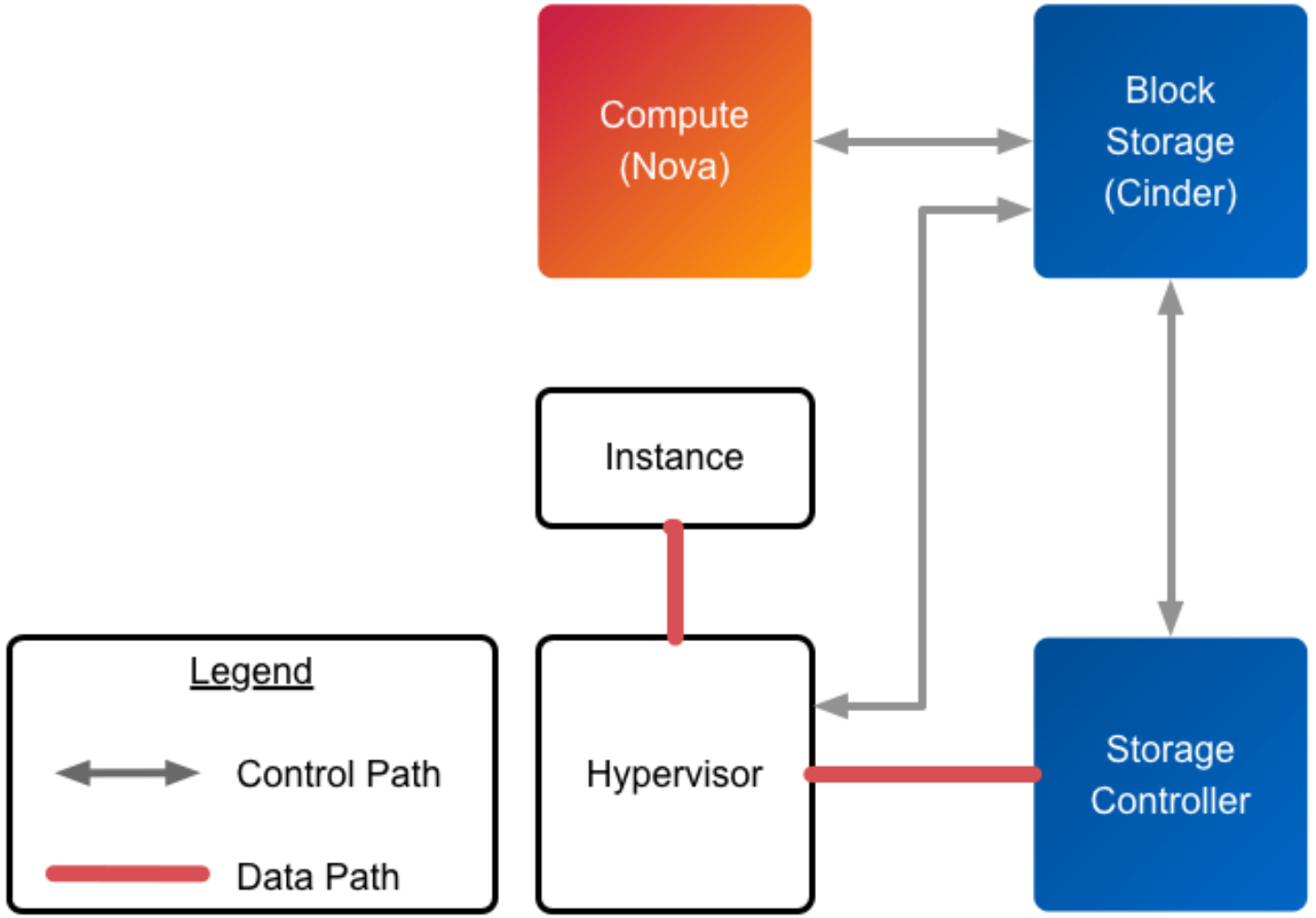


Cinder - block storage

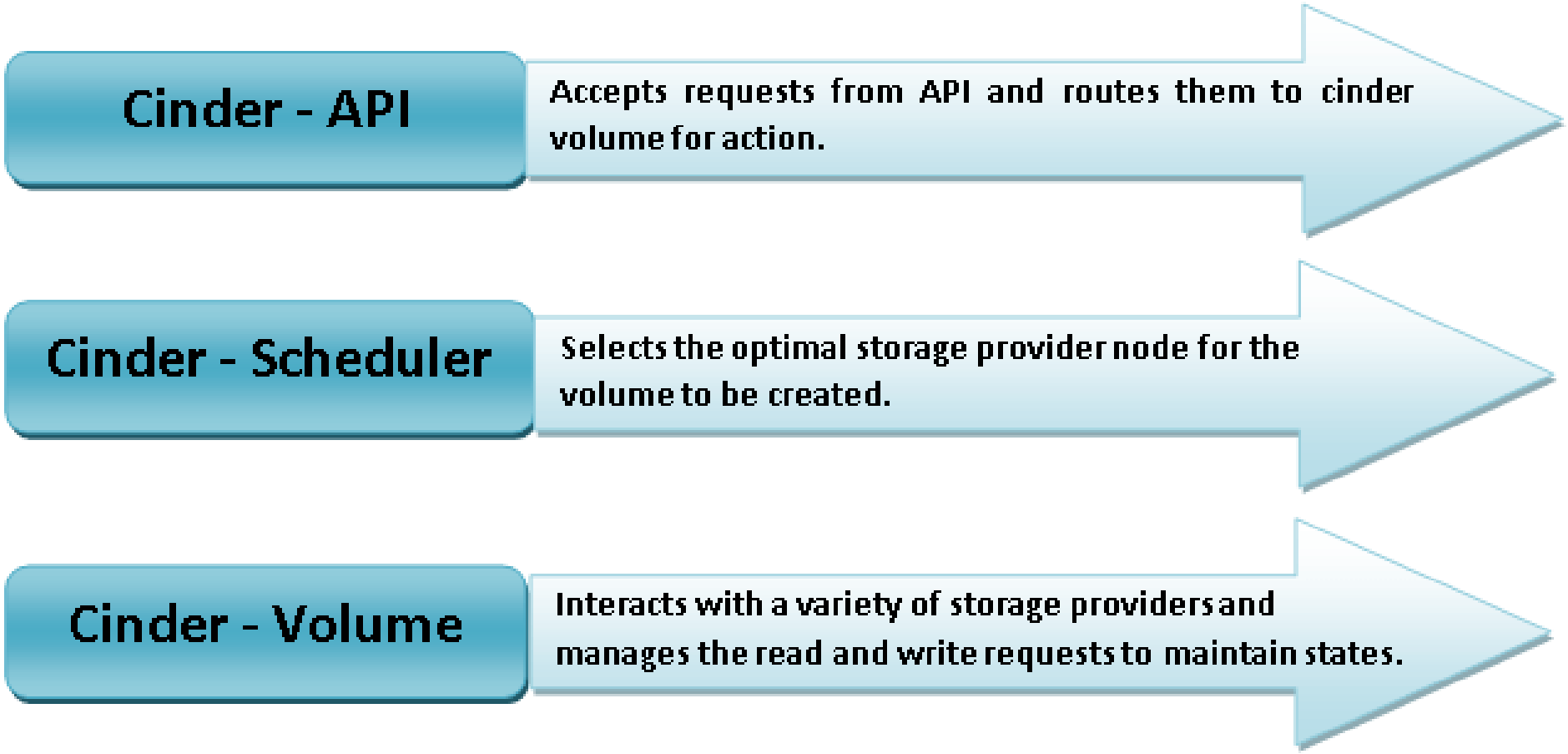
- The management of the persistent block storage is available in OpenStack by using the Cinder service.
- Its main capability is to provide block-level storage to the virtual machine.
- Cinder provides raw volumes that can be used as hard disks in virtual machines.
- Some of the features that Cinder offers are as follows:
- Volume management: This allows the creation or deletion of a volume.
- Snapshot management: This allows the creation or deletion of a snapshot of volumes.
- Attaching or detaching volumes from instances Cloning volumes
- Creating volumes from snapshots
- Copy of images to volumes and vice versa

Cinder - block storage

- This works in the traditional way of attaching and detaching an external hard drive to the OS for its local use.
- Cinder manages to add, remove, create new disk space in the server. This component provides the virtual storage for the VMs in the system. Conceptually, Cinder is similar in function to the EBS (Elastic Block Storage).
- It is usually implemented in combination with other OpenStack services (e.g., Compute, Object Storage, Image, etc.).
- Without needing to think about costly physical storage systems or servers, Cinder users are able to reduce and expand their storage space significantly.
- In addition, by allowing users to use one code for each operation, Cinder simplifies code management.
- With reliability and ease of usage, Cinder can handle all the provisioning and eliminate consumers' needs.



Cinder and Nova logical architecture are:



Cinder Components are:

Manila – File share

- It offers file-based storage to a VM. This component gives an infrastructure for managing and provisioning file shares.
- Manila uses a SQL based central database shared by all manila services in the system.
- The Manila service can operate in the configuration of a single node or multiple nodes.
- Usually, Manila is deployed with other OpenStack resources, such as Compute, Image or Object Storage.

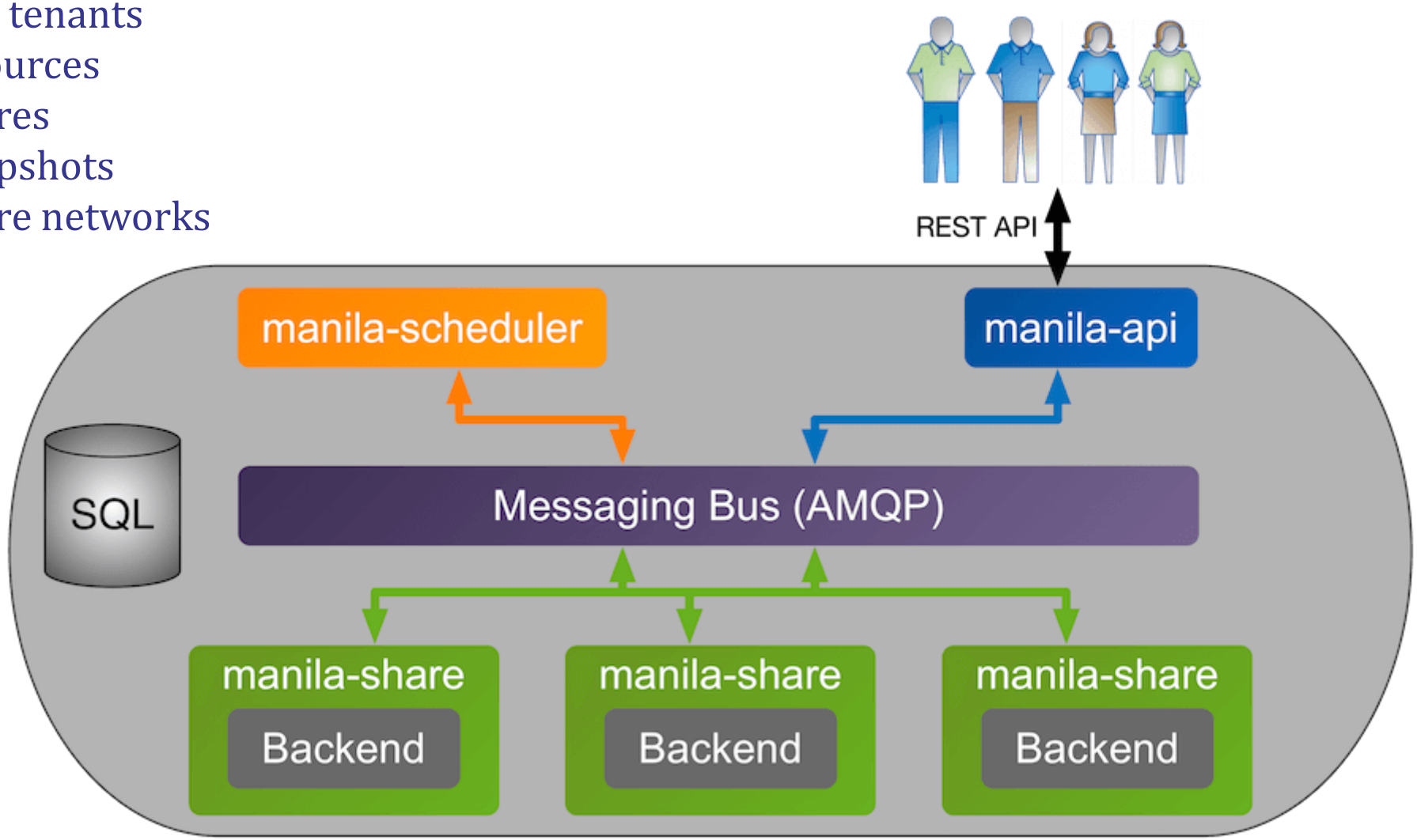
The shared file system (Manila) contains the following set of components :

•Back-end storage devices

•Users and tenants

•Basic resources

- Shares
- Snapshots
- Share networks



Manila – File share

- Since the Juno release, OpenStack has also had a file-share-based storage service called Manila.
- It provides storage as a remote file system. In operation, it resembles the Network File System (NFS) or SAMBA storage service that we are used on Linux.
- It offers file-based storage to a VM. This component gives an infrastructure for managing and provisioning file shares.
- Manila uses a SQL based central database shared by all manila services in the system.
- The Manila service can operate in the configuration of a single node or multiple nodes.
- Usually, Manila is deployed with other OpenStack resources, such as Compute, Image or Object Storage.

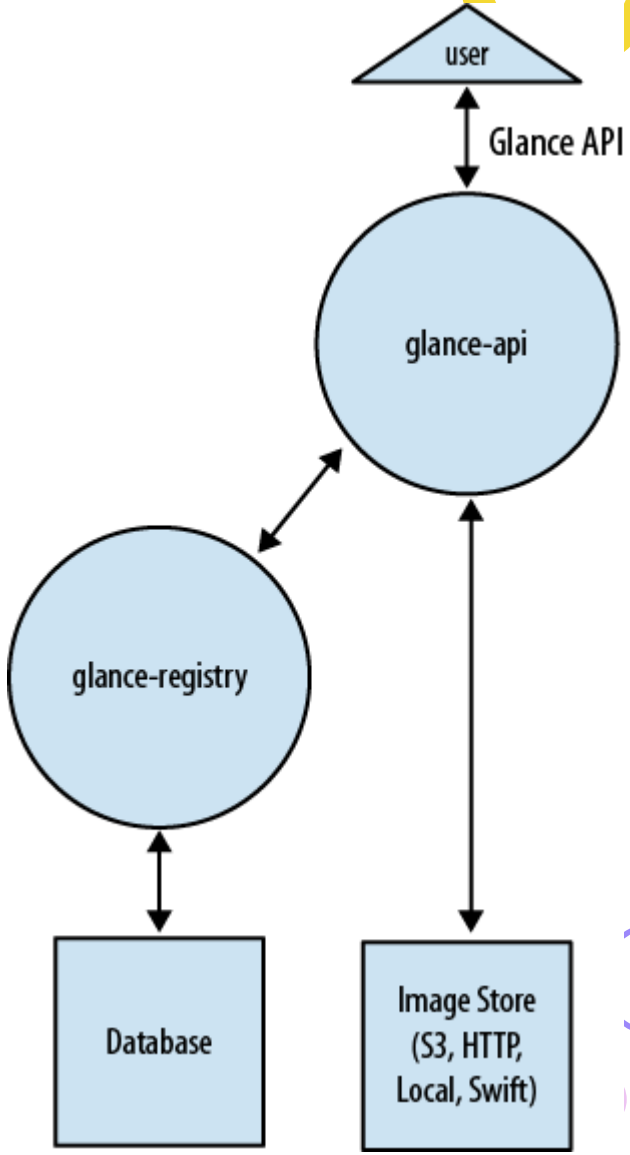
Manila – File share

- Each storage solution in OpenStack has been designed for a specific set of purposes and implemented for different targets.
- Before taking any architectural design decisions, it is crucial to understand the difference between existing storage options in OpenStack today, as outlined in the following table:

Specification	**Storage Type**		
	Swift	**Cinder**	**Manila**
Access mode	Objects through REST API	As block devices.	File-based access
Multi-access	OK	No, can only be used by one client	OK
Persistence	OK	OK	OK
Accessibility	Anywhere	Within single VM	Within multiple VMs
Performance	OK	OK	OK

Glance - Image registry

- The glance component is used to provide the image services to OpenStack. Here, image service means the images or virtual copies of hard disks.
- When we plan to deploy a new virtual machine instance, then glance allows us to use these images as templates.
- Glance contains a REST API from which you can query the metadata of a VM image and retrieve an actual image.
- It is central to IaaS (Infrastructure as a service).



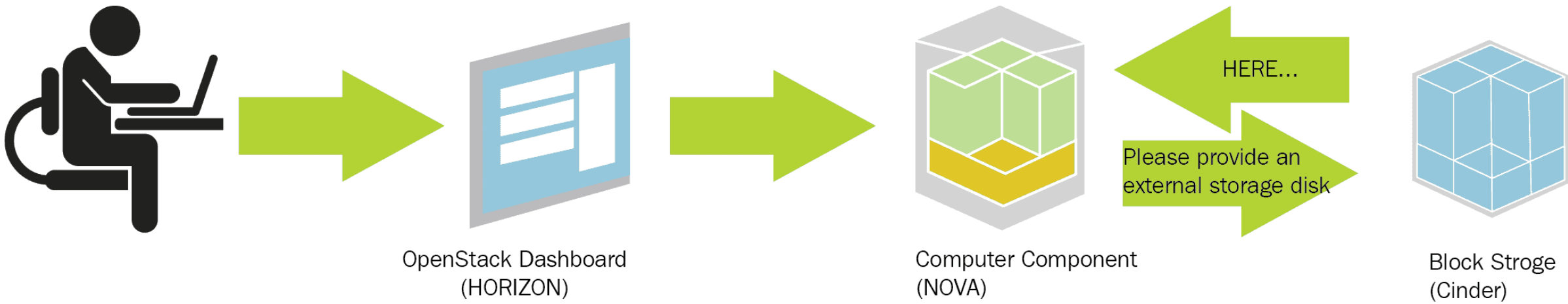
Glance - Image registry

- The OpenStack Glance service contains the following components :
- **glance-api:** For image detection, retrieval, and storage, glance-api accepts image API calls.
- **glance-registry:** It stores, processes, and retrieves image metadata. Where items like size and type are included in the metadata.
- **Database:** It is used to store image metadata, and according to your choice, you can select your database. MySQL or SQLite are used for most deployments.
- **Storage repository for image files:** Standard file systems (or any file system installed on the glance-api controller node), HTTP, Object Storage, RADOS block devices, and VMware Datastore multiple repository forms are supported.
- **Metadata definition service:** It is a normal API to meaningfully define your own custom metadata for suppliers, administrators, services, and users. Such metadata may be used for various resource types, such as images, artifacts, volumes, flavors, and aggregates.

Difference between Cinder and Swift?

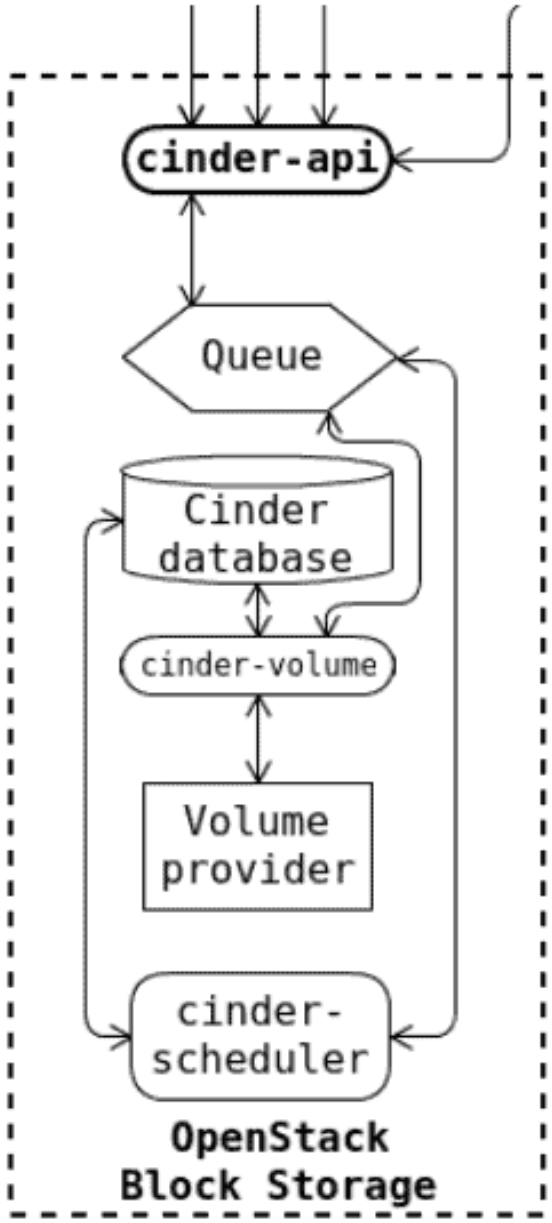
- **Cinder (Block Storage)**
- Cinder is a block storage device.
- Let's suppose it has been a year since Mr. Neo started using his workstation. His daily task saves loads and loads of data onto his hard disk. Mr. Neo noticed that he would run out of his 500 GB internal hard drive space in the next few days. So, **he decided to demand a new external hard drive** (portable HDD) from the storage team and raised the new request for an external hard drive of 1 TB in size.
- When the storage team received the request, it will **verify the token ID** of Mr. Neo with the access management team to confirm that the **user is authorized to receive an additional storage**.
- Once the authorization is verified, the storage team will **provide him with an external 1 TB hard disk.**
- We all know the advantage of using the portable hard drive. Mr. Neo has the flexibility to connect and mount the external HDD permanently and use it as a secondary internal hard drive for his workstation or optionally, he could also **mount** the external HDD temporarily to backup all necessary files and **detach** the external HDD from his workstation:

Nova requests the block storage component (Cinder) for an additional volume storage

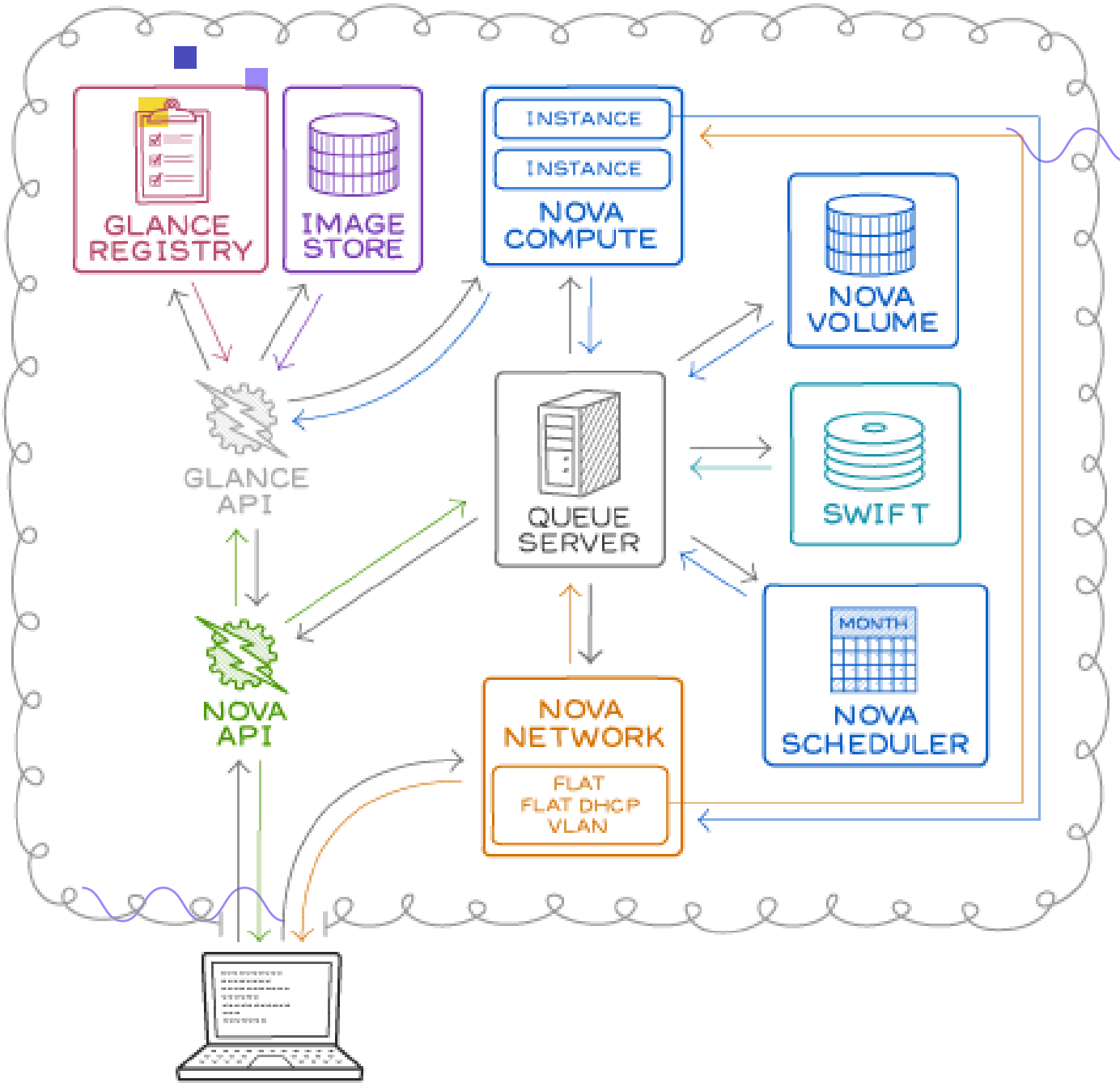


Difference between Cinder and Swift?

- Similarly, the **cinder service will do the storage team's job** in OpenStack. The Cinder block storage service will take care of providing the additional disk volume to the user. Once the new disk volume is allocated to the user tenant (project), the user has the flexibility to map (attach) the volume storage to any VMs between the same project. A Cinder volume is very similar to an external HDD; **you cannot attach the single cinder volume to two or more virtual machines at the same time**, however we could connect the single cinder volume to any virtual machine, one at a time.
- The **logical architecture diagram** shows how various processes and daemons work together to perform the block storage service (Cinder) in OpenStack, and the interconnection between them:



NOVA


GLANCE REGISTRY
IMAGE STORE
INSTANCE
INSTANCE
NOVA COMPUTE
NOVA VOLUME
GLANCE API
QUEUE SERVER
SWIFT
NOVA API
NOVA NETWORK
FLAT
FLAT DHCP
VLAN
MONTH
NOVA SCHEDULER

Nova –Compute Engine

- Nova is the original core component of OpenStack.
- From an architectural level, it is considered one of the most complicated components of OpenStack.
- Nova provides the compute service in OpenStack and manages virtual machines in response to service requests made by OpenStack users.
- What makes Nova complex is its interaction with a large number of other OpenStack services and internal components, which it must collaborate with to respond to user requests for running a VM.

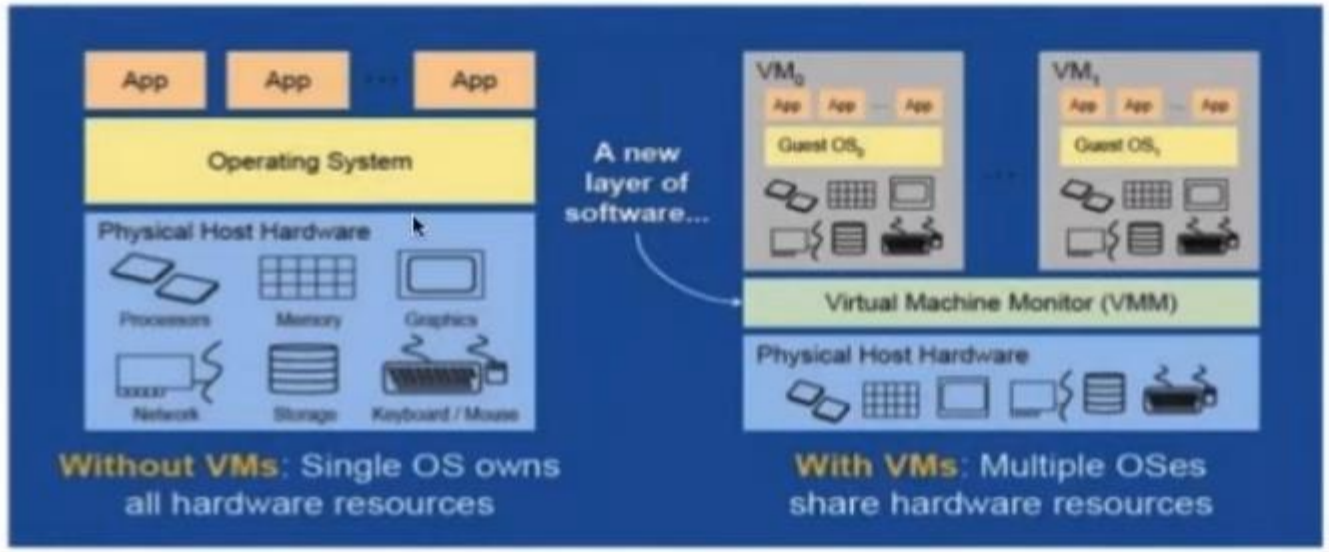
Nova –Compute Engine

- **Functionality**
- The nova-API handles the requests and responses from and to the end-user.
- The nova-compute creates and destroys the instances as and when a request is made.
- Nova-scheduler schedules the tasks to the nova-compute
- The glace registry stores the details of the image along with its metadata.
- The Image store stores the images predefined by the admin/user.
- The nova network ensures network connectivity and routing.

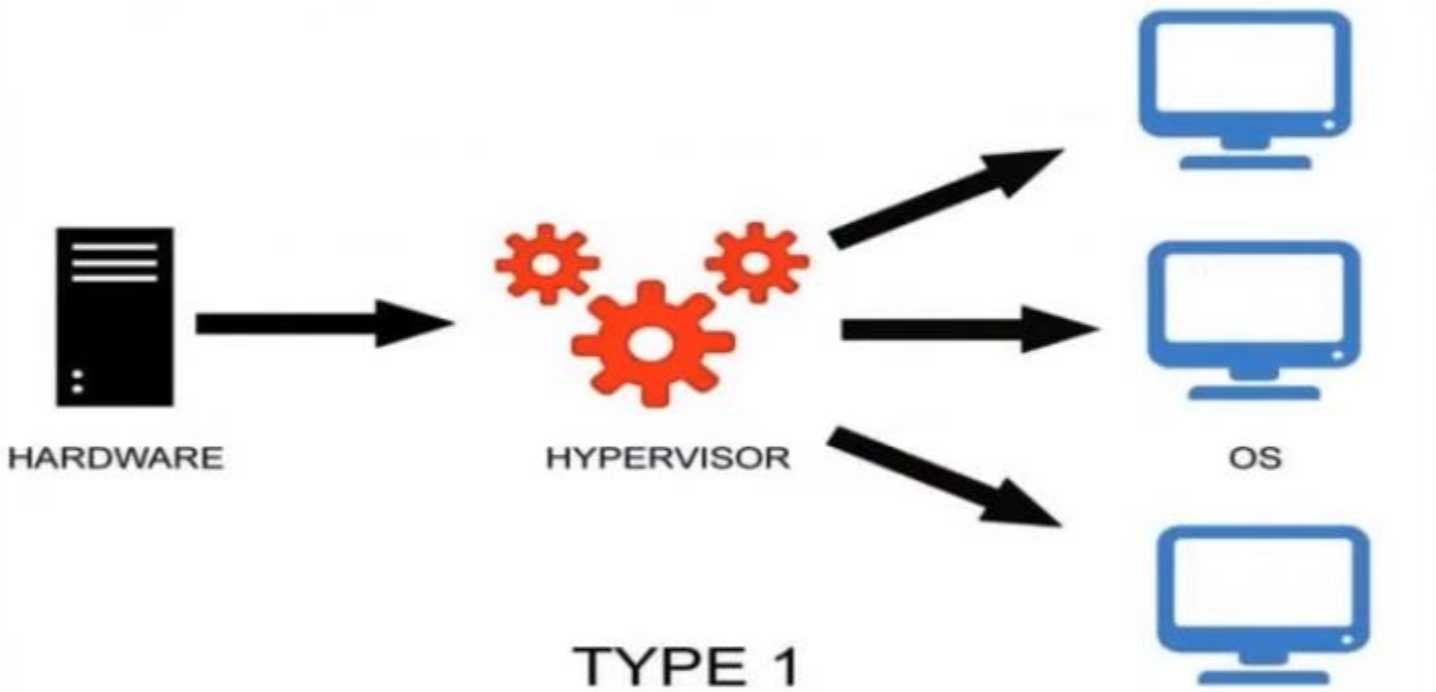
HyperVisor

What is a Hypervisor?

- A hypervisor, also called a virtual machine manager **(VMM),** is a program that allows multiple operating systems to share a single hardware host.



HyperVisor

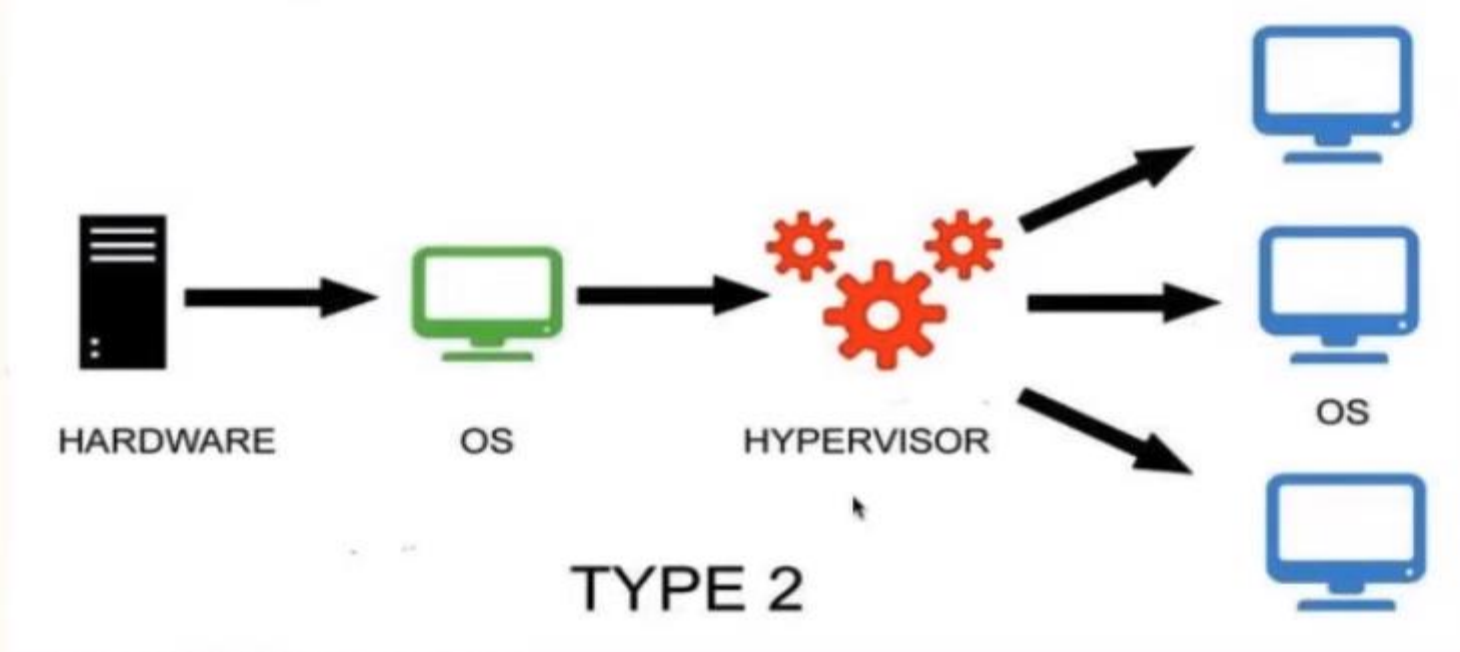


TYPE 1

Bare metal, native or type I hypervisors

This is when the hypervisors are run on the host's hardware to control it as well as manage the virtual machines on it.

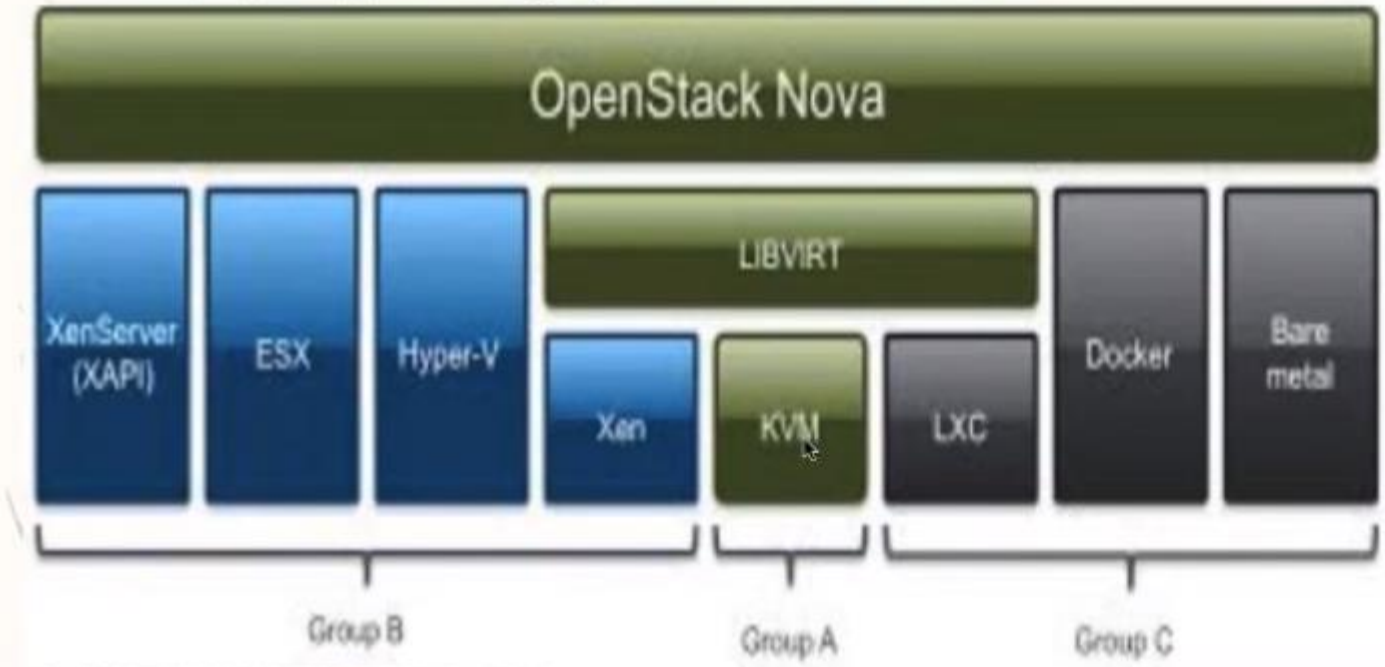
HyperVisor



Type 2 – also known as a hosted hypervisor.

This type of hypervisor is installed as a software application on an existing host operating system (OS). Example of the hosted hypervisor are Oracle VirtualBox, Microsoft Virtual PC, Vmware Server and Workstation.

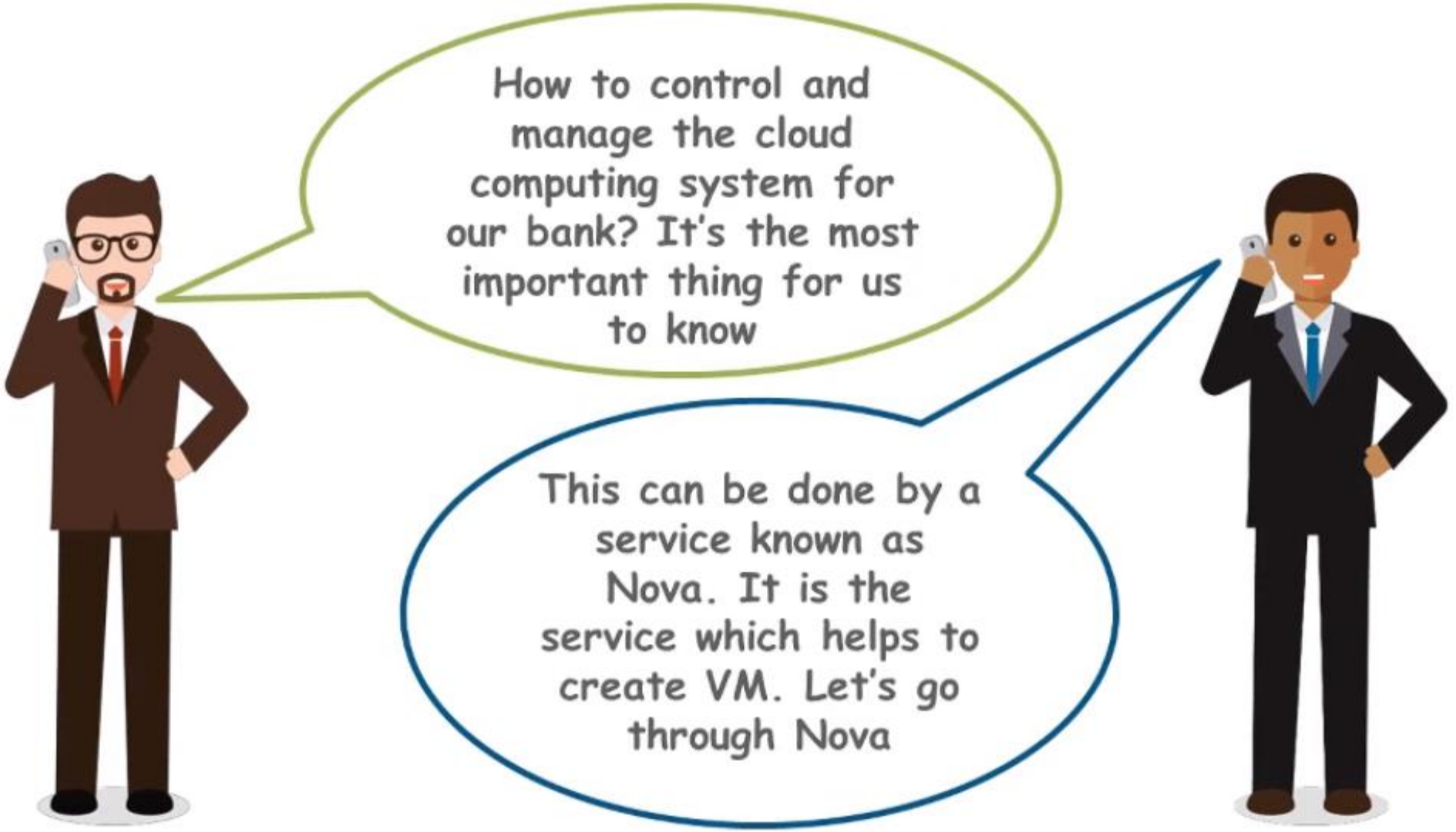
HyperVisor



The following hypervisors are supported:

- KVM - Kernel-based Virtual Machine. The virtual disk formats that it supports is inherited from QEMU since it uses a modified QEMU program to launch the virtual machine. The supported formats include raw images, the qcow2, and VMware formats.
- LXC - Linux Containers (through libvirt), used to run Linux-based virtual machines.
- QEMU - Quick EMUlator, generally only used for development purposes.
- UML - User Mode Linux, generally only used for development purposes.
- VMware vSphere 5.1.0 and newer, runs VMware-based Linux and Windows images through a connection with a vCenter server.
- Xen (using libvirt) - Xen Project Hypervisor using libvirt as management interface into nova-compute to run Linux, Windows, FreeBSD and NetBSD virtual machines.
- XenServer - XenServer, Xen Cloud Platform (XCP) and other XAPI based Xen variants runs Linux or Windows virtual machines. You must install the nova-compute service in a para-virtualized VM.

Nova as a Compute Solution in OpenStack



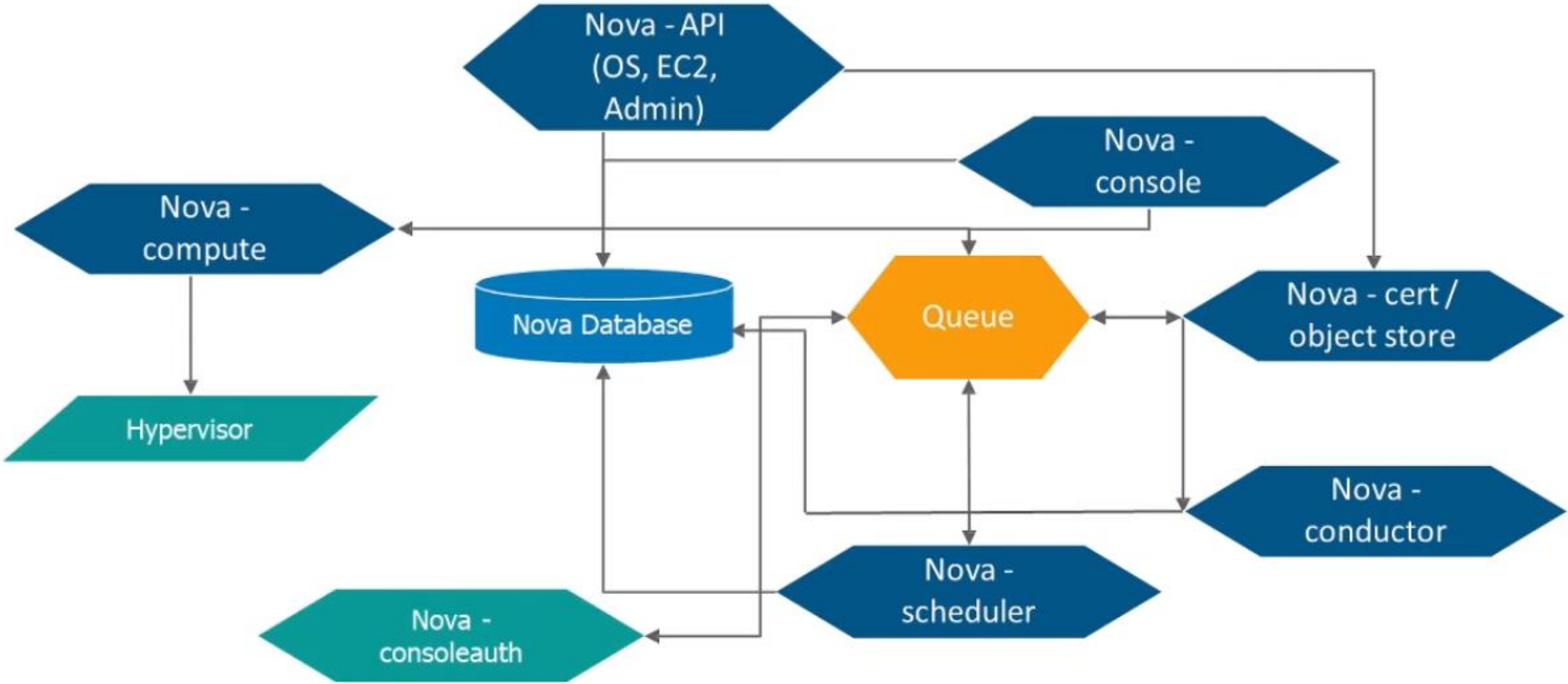
Introducing Nova

Nova, OpenStack Compute Service is used for hosting and managing cloud computing systems

Nova is built on a messaging architecture and all of its components can run on several servers

It is fault tolerant, recoverable and provides API-compatibility

Nova Architecture



Nova Components

Nova-API

Manages API HTTP requests and is used to interact with Nova

Nova-database

Nova database stores current state of all objects in the compute cluster

Nova (message) Queue

The messaging channel that passes messages between Nova components

Nova-scheduler

It's a daemon which determines on which compute host the instance should run on

Nova-compute

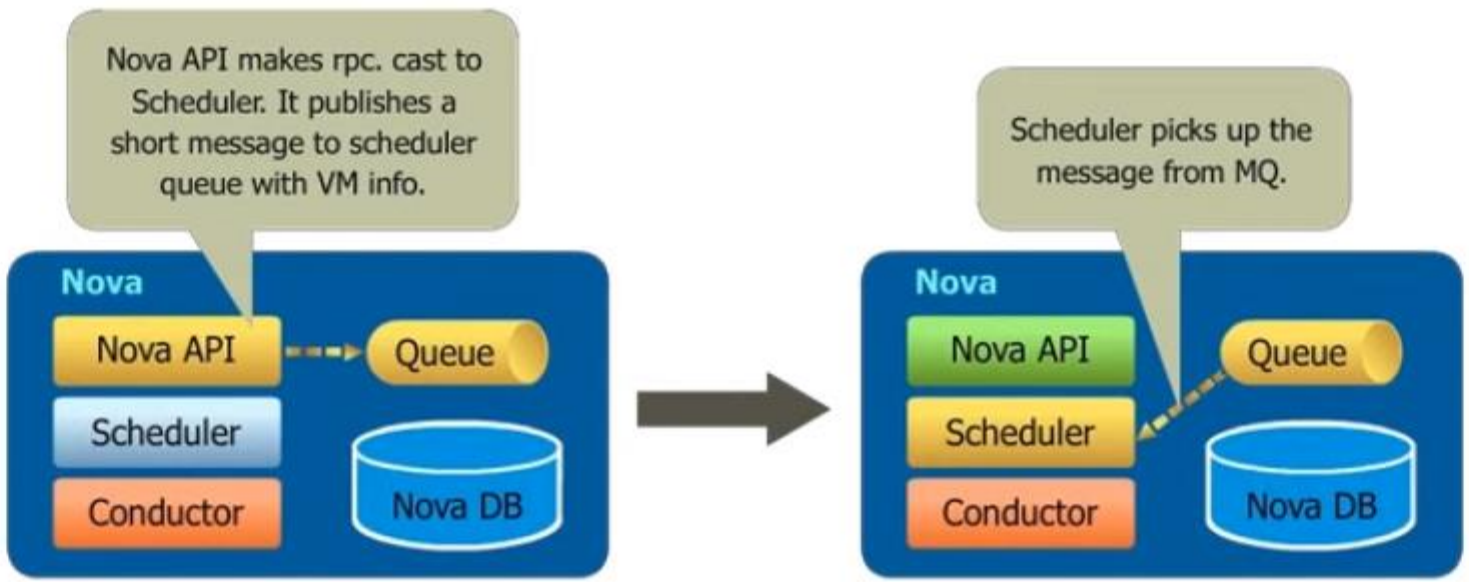
It is the work daemon which creates and terminates VMs via Hypervisor API

Nova-conductor

It is a service that conducts a no-db-compute function

Nova-API – Description

- Nova API accepts and responds to user's compute API calls
- It supports OpenStack compute API, Amazon EC2 API and Admin API, and initiates Orchestration Activities
- Once a call has been made to Nova-API, the call is sent to the queue if AUTH request is validated by Identity



rpc.cast - don't wait for result (fire and forget)

rpc.call - wait for result (when there is something to return)

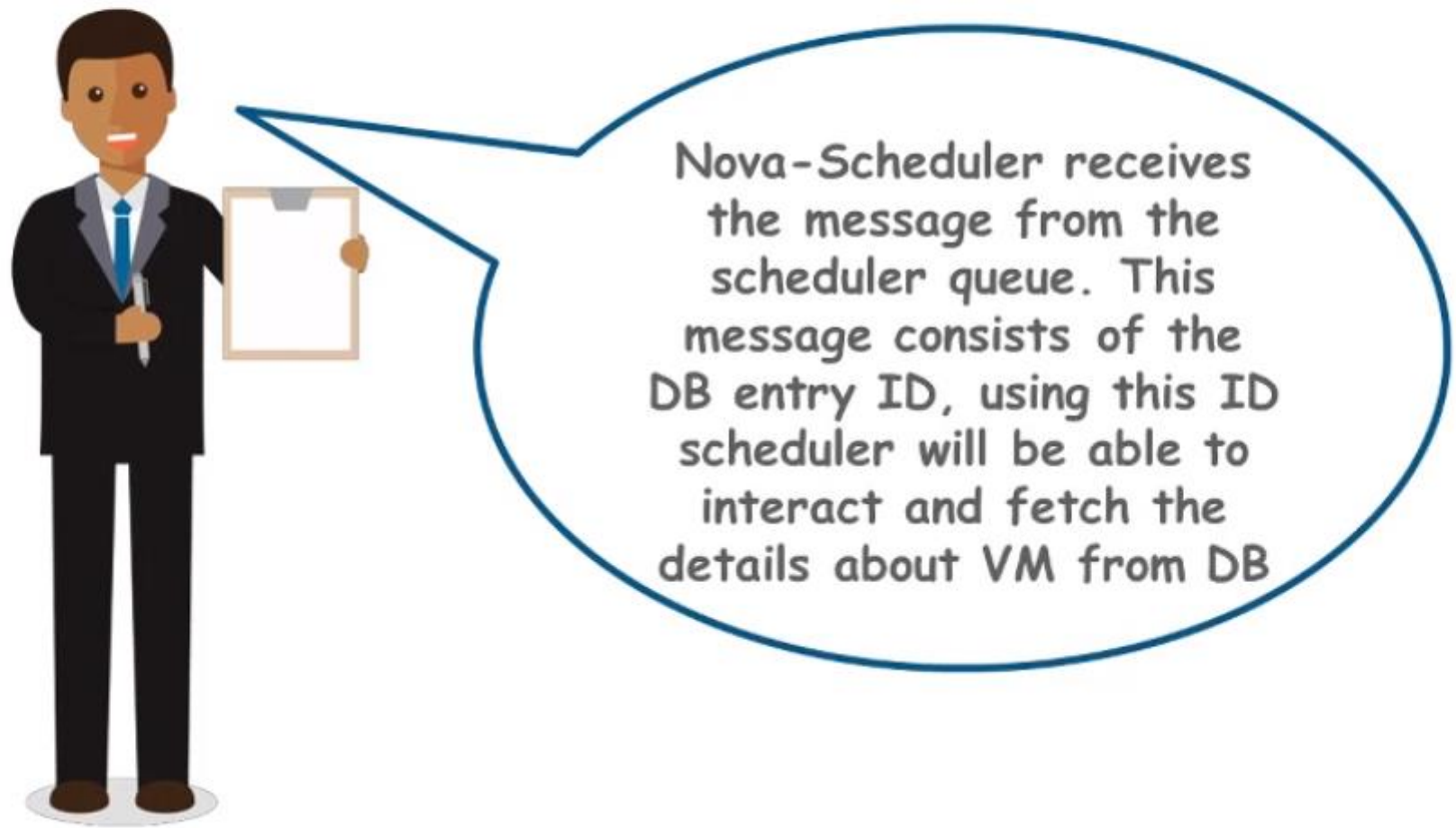
NOVA- Components

- **nova-api**
- The nova-api component accepts and responds to the end user and computes API calls.
- The end users or other components communicate with the OpenStack nova-api interface to create instances via the OpenStack API or EC2 API.
- The nova-api initiates most orchestrating activities such as the running of an instance or the enforcement of some particular policies.
- **nova-compute**
- The nova-compute component is primarily a worker daemon that creates and terminates VM instances via the hypervisor's APIs (XenAPI for XenServer, Libvirt KVM, and the VMware API for VMware).
- **nova-network**
- The nova-network component accepts networking tasks from the queue and then performs these tasks to manipulate the network (such as setting up bridging interfaces or changing IP table rules).
- Neutron is a replacement for the nova-network service.

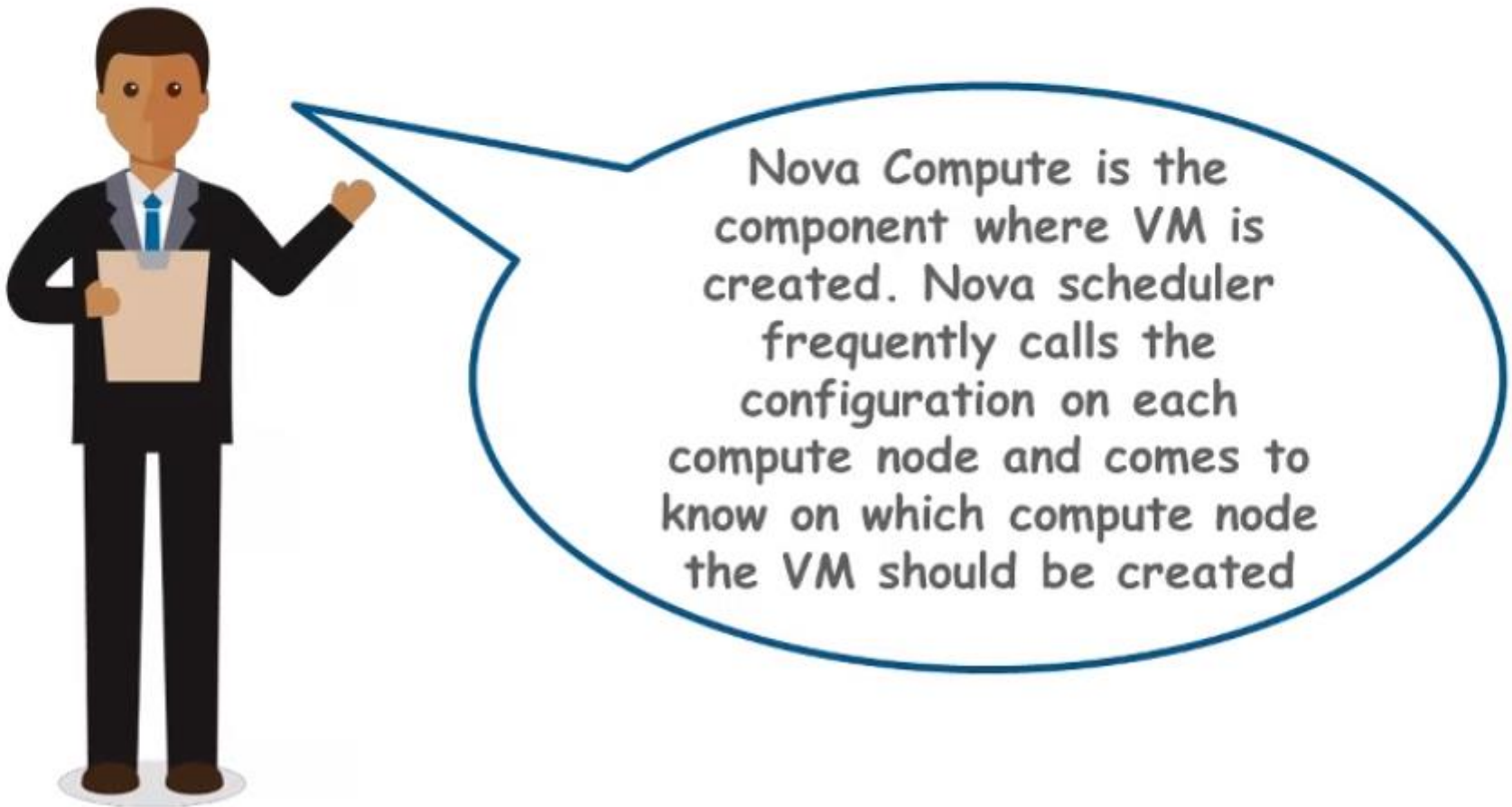
NOVA- Components

- **nova-scheduler**
- The nova-scheduler component takes a VM instance's request from the queue and determines where it should run (specifically which compute host it should run on).
- At an application architecture level, the term scheduling or scheduler invokes a systematic search for the best outfit for a given infrastructure to improve its performance.
- **nova-conductor**
- The nova-conductor service provides database access to compute nodes.
- The idea behind this service is to prevent direct database access from the compute nodes, thus enhancing database security in case one of the compute nodes gets compromised.

Nova Scheduler

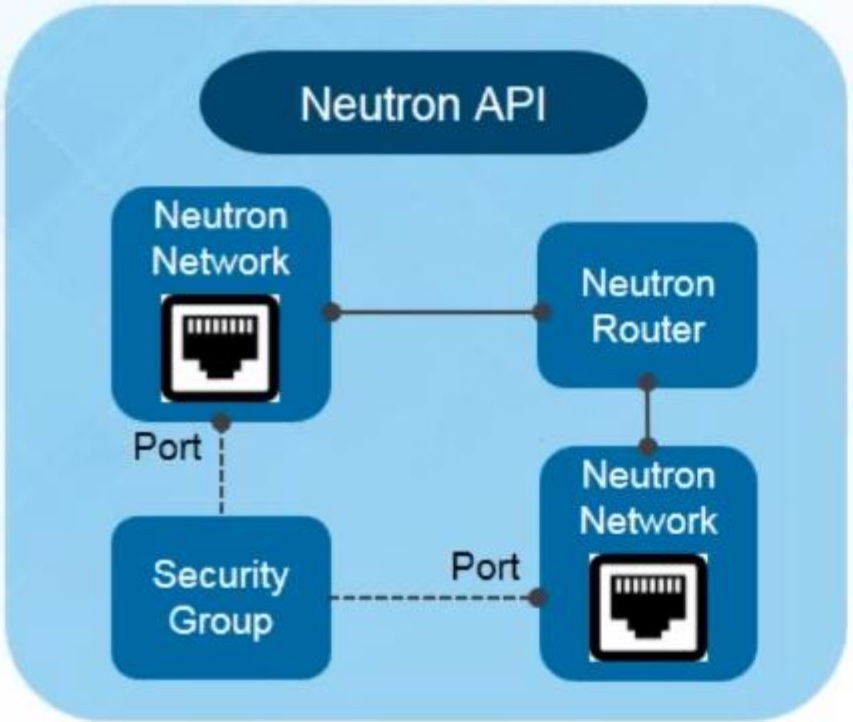


Nova Compute



OpenStack Neutron

- Neutron is a networking project focused on delivering Networking-as-a-Service (NaaS) in virtual compute environments.
- Neutron has replaced the original networking application program interface (API) called Quantum in OpenStack.



NEUTRON

- Provides Network as a Service (NaaS)
- Provides rich topologies

Characteristics of Neutron:

- It allows users to create their own networks and then attaches server interfaces to them.
- Its pluggable backend architecture lets users take advantage of vendor-supported equipment.
- It provides extensions to allow additional network services to be integrated.
- *Neutron has many core network features that are constantly growing and maturing. Some of these features are useful for routers, virtual switches, and SDN networking controllers.*

NEUTRON

Neutron introduces the following core resources:

- **Ports:** Ports in Neutron refer to the virtual switch connections.
- *These connections are where instances and network services are attached to networks. When attached to subnets, the defined MAC and IP addresses of the interfaces are plugged into them.*
- **Networks:** Neutron defines networks as isolated Layer 2 network segments.
- *Operators will see networks as logical switches that are implemented by the Linux bridging tools, Open vSwitch, or some other virtual switch software.*
- **Subnet:** Subnets in Neutron represent a block of IP addresses associated with a network. IP addresses from this block are allocated to the ports.

NEUTRON

Neutron provides additional resources as extensions.

- Routers: Routers provide gateways between various networks.
- Private IPs: Neutron defines two types of networks.
    a. They are as follows:
    b. **Tenant networks**: Tenant networks use private IP addresses. Private IP addresses are visible within the instance and this allows the tenant's instances to communicate while maintaining isolation from the other tenant's traffic. Private IP addresses are not visible to the Internet.
    c. **External networks:** External networks are visible and routable from the Internet. They must use routable subnet blocks.
    d. **Floating IPs**: A floating IP is an IP address allocated on an external network that Neutron maps to the private IP of an instance. Floating IP addresses are assigned to an instance so that they can connect to external networks and access the Internet. Neutron achieves the mapping of floating IPs to the private IP of the instance by using Network Address Translation (NAT)

NEUTRON

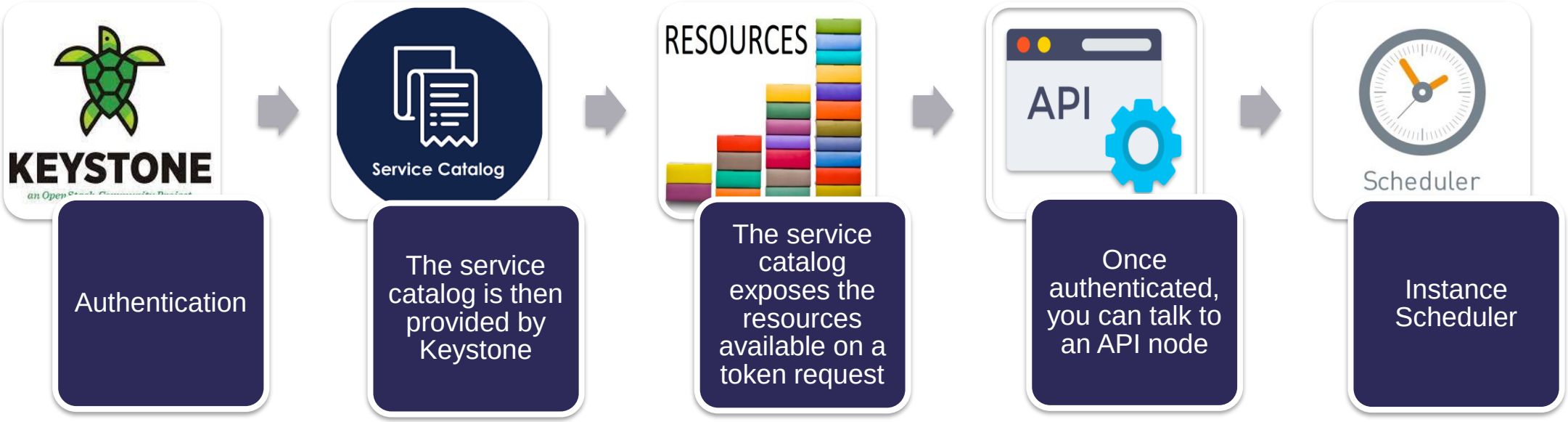
- Neutron also provides advanced services to rule additional network OpenStack capabilities as follows:
- Load Balancing as a Service (LBaaS) to distribute the traffic among multiple compute node instances.
- Firewall as a Service (FWaaS) to secure layer 3 and 4 network perimeter access.
- Virtual Private Network as a Service (VPNaaS) to build secured tunnels between instances or hosts.

The Neutron architecture

- The three main components of the Neutron architecture are:
- **Neutron server:** It accepts API requests and routes them to the appropriate Neutron plugin for action.
- **Neutron plugins:** They perform the actual work for the orchestration of backend devices such as the plugging in or unplugging ports, creating networks and subnets, or IP addressing.
- **Neutron agents**: Neutron agents run on the compute and network nodes. The agents receive commands from the plugins on the Neutron server and bring the changes into effect on the individual compute or network nodes. Different types of Neutron agents implement different functionality. (Eg: L2 and L3 agents)
- Neutron is a service that manages network connectivity between the OpenStack instances. It ensures that the network will not be turned into a bottleneck or limiting factor in a cloud deployment and gives users real self-service, even over their network configurations.


GATHERING THE
PIECES AND
BUILDING A PICTURE

How OpenStack Works



Infobox

The service catalog is a JSON structure that exposes the resources available on a token request.

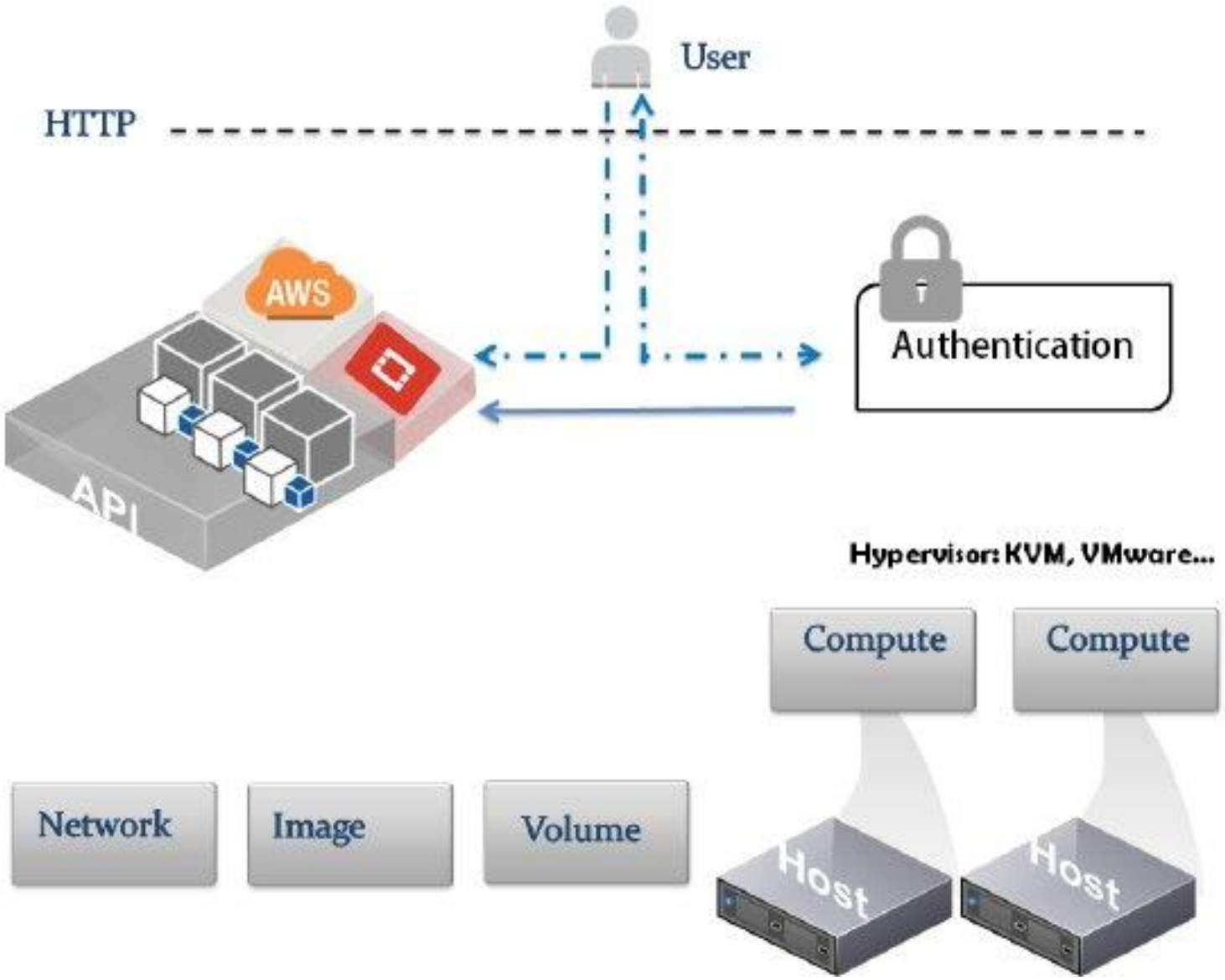
Infobox

The scheduling process in OpenStack Nova can perform different algorithms such as simple, chance, and zone. An advanced way to do this is by deploying weights and filters by ranking servers as its available resources.

How OpenStack Works

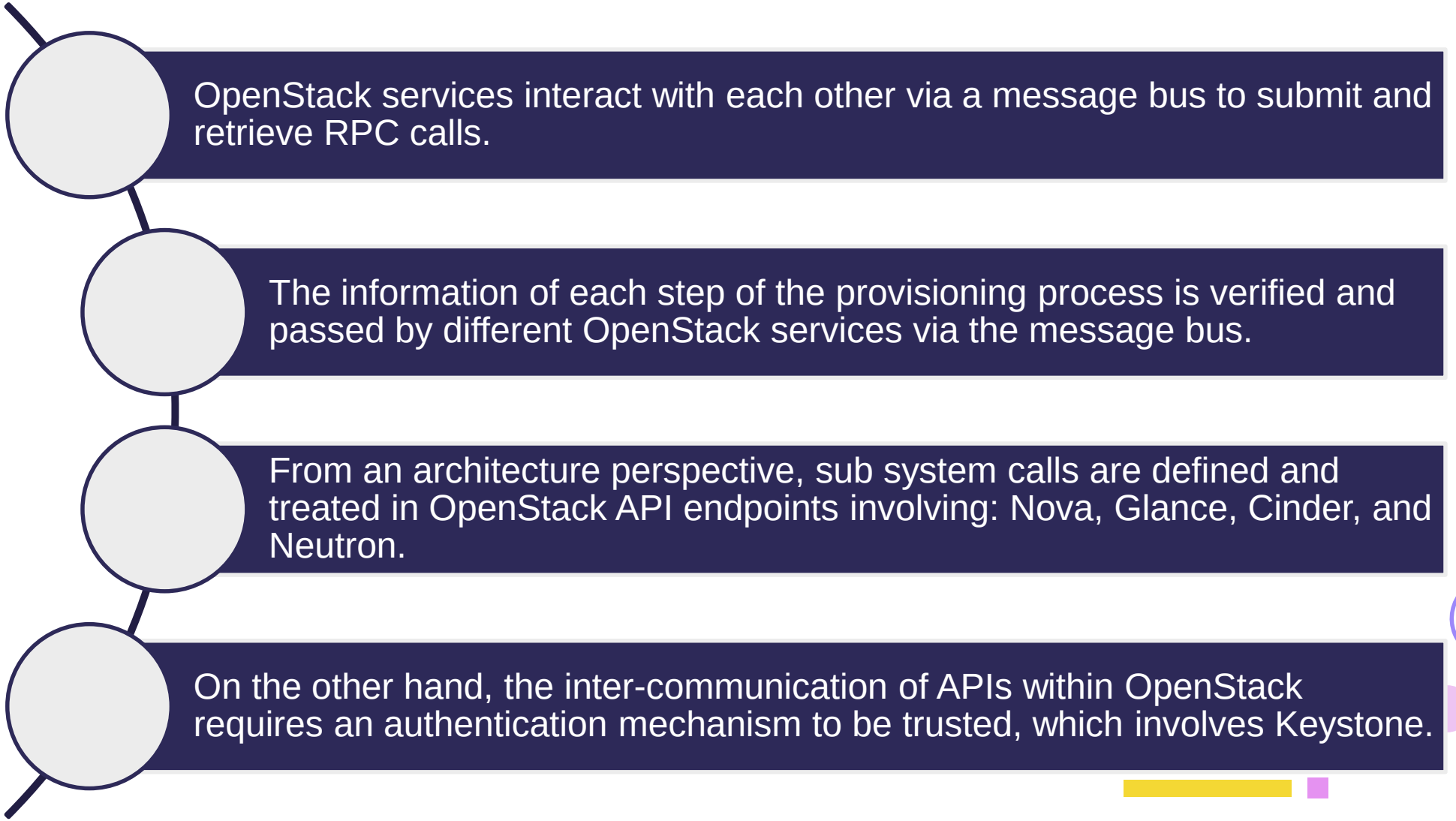
1. Authentication is the first action performed. This is where Keystone comes into the picture. Keystone authenticates the user based on credentials such as the username and password.
2. The service catalog is then provided by Keystone. This contains information about the OpenStack services and the API endpoints.
3. You can use the Openstack CLI to get the catalog:
    a. $ oqenstack catalog list
    b. The service catalog is a JSON structure that exposes the resources available on a token request
4. Typically, once authenticated, you can talk to an API node. There are different APIs in the OpenStack ecosystem (the OpenStack API and EC2 API):
5. Another element in the architecture is the instance scheduler. Schedulers are implemented by OpenStack services that are architected around worker daemons. The worker daemons manage the launching of instances on individual nodes and keep track of resources available to the physical nodes on which they run. The scheduler in an OpenStack service looks at the state of the resources on a physical node (provided by the worker daemons) and decides the best candidate node to launch a virtual instance on. An example of this architecture is nova-scheduler. This selects the compute node to run a virtual machine or Neutron L3 scheduler, which decides which L3 network node will host a virtual router.

High-level view of how OpenStack works:



Provisioning a VM under the hood

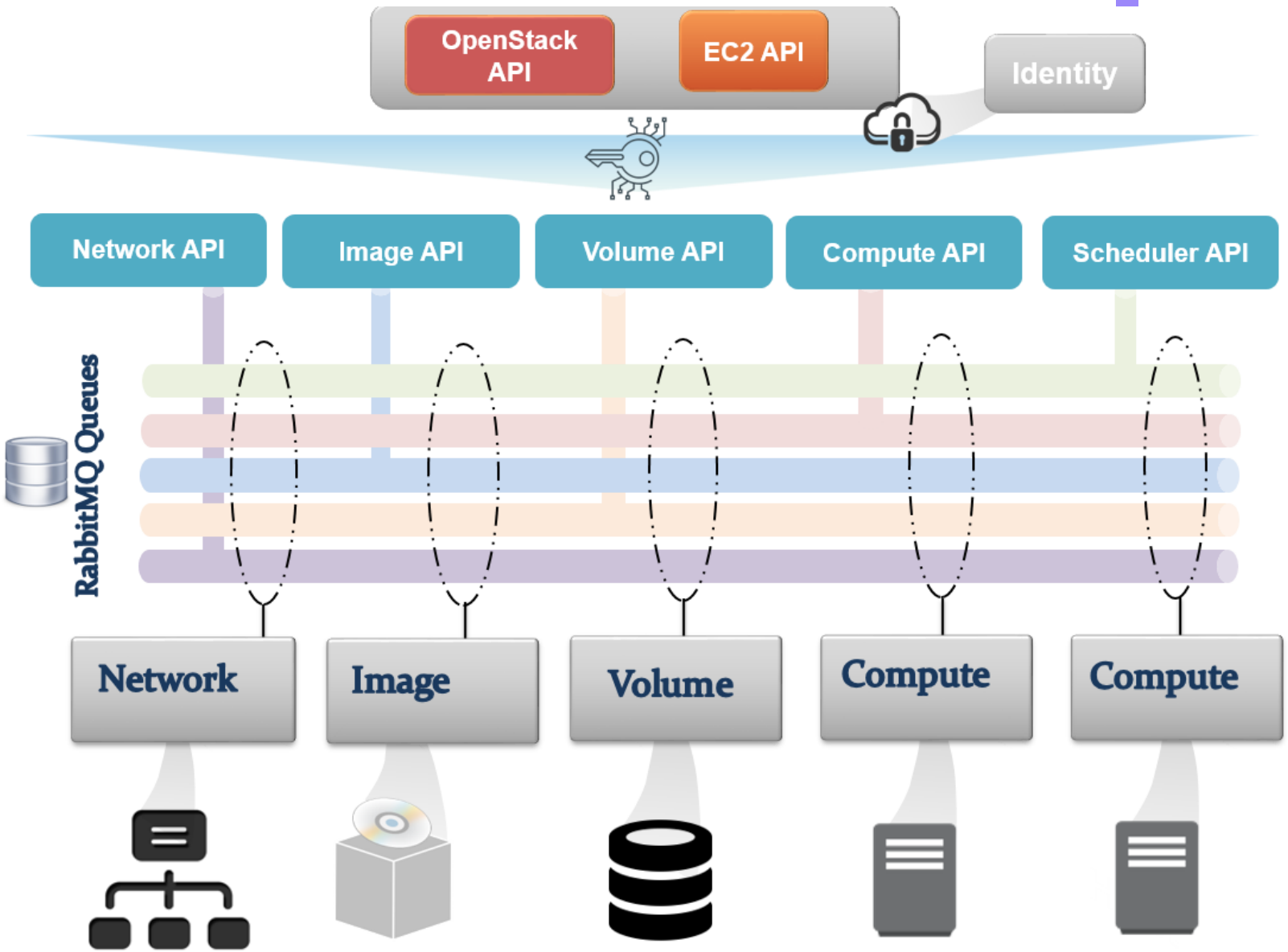
- The process of launching a virtual machine involves the interaction of the main OpenStack services that form the building blocks of an instance including compute, network, storage, and the base image.



Provisioning workflow based on API calls in OpenStack:

1. Calling the identity service for authentication
2. Generating a token to be used for subsequent calls
3. Contacting the image service to list and retrieve a base image
4. Processing the request to the compute service API
5. Processing compute service calls to determine security groups and keys
6. Calling the network service API to determine available networks
7. Choosing the hypervisor node by the compute scheduler service
8. Calling the block storage service API to allocate volume to the instance
9. Spinning up the instance in the hypervisor via the compute service API call
10. Calling the network service API to allocate network resources to the instance

Provisioning a VM under the hood



Tokens in OpenStack

- It is important to keep in mind that handling tokens in OpenStack on every API call and service request is a time limited operation. One of the major causes of a failed provisioning operation in OpenStack is the expiration of the token during subsequent API calls.
- Additionally, the management of tokens has faced a few changes within different OpenStack releases. This includes two different approaches used in OpenStack prior to the Liberty release including:

  - **Universally Unique Identifier (UUID):** Within Keystone version 2, an UUID token will be generated and passed along every API call between client services and back to Keystone for validation. This version has proven performance degradation of the identity service.

  - **Public Key Infrastructure (PKI):** Within Keystone version 3, tokens are no longer validated at each API call by Keystone. API endpoints can verify the token by checking the Keystone signature added when initially generating the token.

Tokens in OpenStack

- Starting from the Kilo release, handling tokens in Keystone has progressed by introducing more sophisticated cryptographic authentication token methods, such as **Fernet**
- **Fernet tokens (pronounced fehr:NET)** are message packed tokens that contain authentication and authorization data. Fernet tokens are signed and encrypted before being handed out to users. Most importantly, however, Fernet tokens are ephemeral.
- The new implementation will help to tackle the token performance issue noticed in **UUID and PKI tokens.**
- Fernet tokens (pronounced fehr:NET) are message packed tokens that contain authentication and authorization data. Fernet tokens are signed and encrypted before being handed out to users. Most importantly, however, Fernet tokens are ephemeral. This means they do not need to be persisted across clustered systems in order to successfully be validated..

Tokens in OpenStack

- **Benefits of Fernet tokens**
- Because Fernet tokens are ephemeral, you have the following immediate benefits:
- Tokens do not need to be replicated to other instances of Keystone in your controller cluster
- Storage is not affected, as these tokens are not stored
- The end-result offers increased performance overall. This was the design imperative of Fernet tokens, and the OpenStack community has more than delivered.

A SAMPLE ARCHITECTURE SETUP

A SAMPLE ARCHITECTURE SETUP

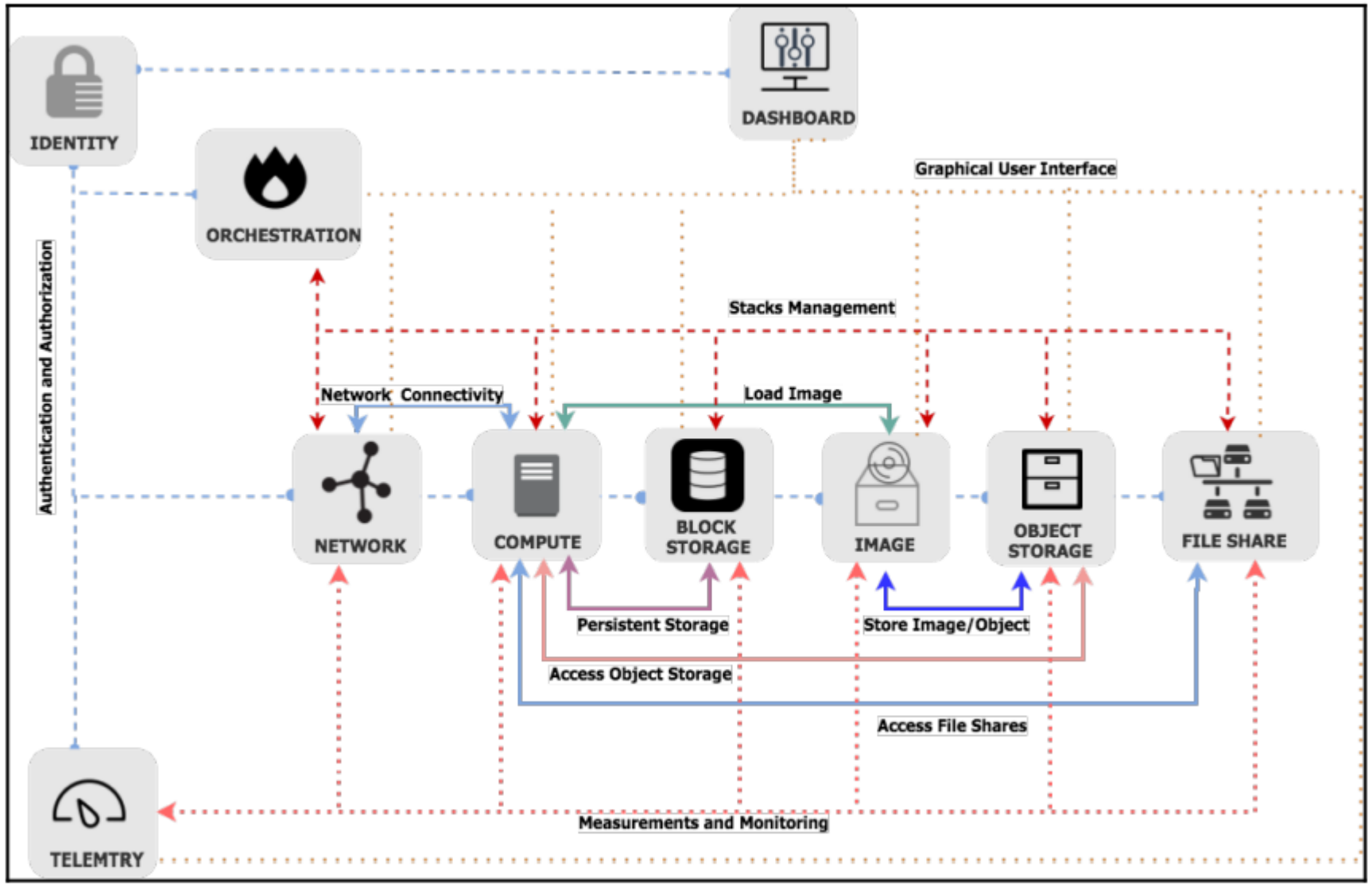
- Many enterprises have successfully designed their OpenStack environments by going through three phases of design: designing a **conceptual model**, designing a **logical model**, and finally, realizing the **physical design**.

Conceptual Model Design

- As the first conceptual phase, we will have our high-level reflection on what we will need from certain generic classes from the OpenStack architecture:

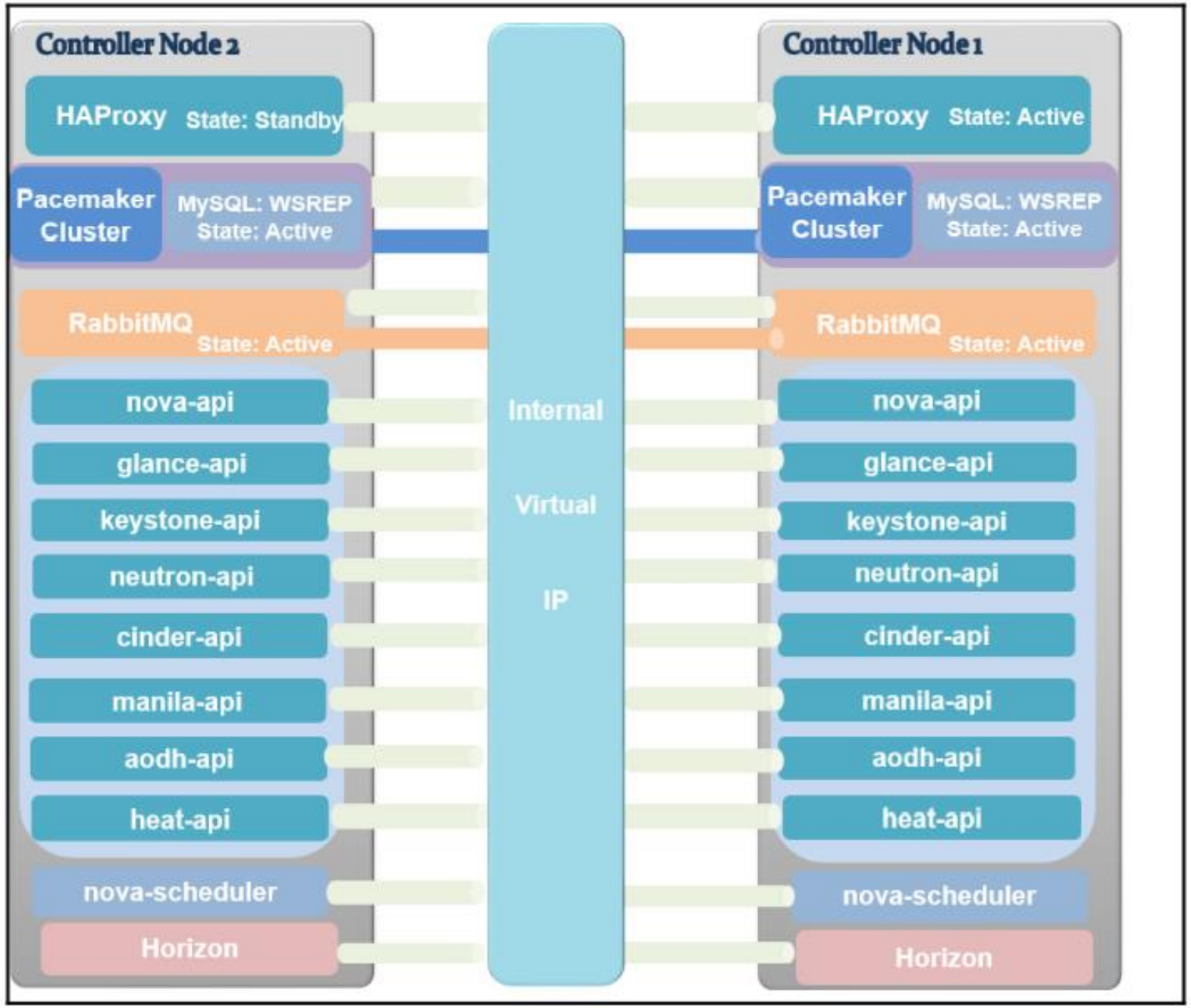
Class	Role
Compute	Stores virtual machine images Provides a user interface
Image	Stores disk files Provides a user interface
Object storage	Stores objects Provides a user interface
Block storage	Provides volumes Provides a user interface
Network	Provides network connectivity Provides a user interface
Telemetry	Provides measurements, metrics, and alerts Provides a user interface
File Share	Provides a scale-out file share system for OpenStack Provides a user interface
Identity	Provides authentication
Dashboard	Provides a graphical user interface
Orchestration	Provides orchestration engine for stack creation Provides a user interface

- Generic classes from the OpenStack architecture:



Logical Model Design

- Based on the conceptual reflection design, most probably you will have a good idea about different OpenStack core components, which will lay the formulation of the logical design.
- We will start by identifying nodes to run an OpenStack service: the cloud controller, network nodes, and the compute node.
- Thus, the physical model design will be elaborated based on the previous theoretical phases by assigning parameters and values to our design. Let's start with our first logical iteration:
- Obviously, in a highly available setup, we should achieve a degree of redundancy in each service within OpenStack.


Controller Node 2
HAProxy State: Standby
Pacemaker Cluster
MySQL: WSREP State: Active
RabbitMQ State: Active
nova-api
glance-api
keystone-api
neutron-api
cinder-api
manila-api
aodh-api
heat-api
nova-scheduler
Horizon
Internal
Virtual
IP
Controller Node 1
HAProxy State: Active
Pacemaker Cluster
MySQL: WSREP State: Active
RabbitMQ State: Active
nova-api
glance-api
keystone-api
neutron-api
cinder-api
manila-api
aodh-api
heat-api
nova-scheduler
Horizon

Logical Model Design

Things to consider in Logical Model Design are

High Availability

- The previous figure includes several essential solutions for a highly-scalable and redundant OpenStack environment such as virtual IP (VIP), HAProxy, and Pacemaker.

Storage

- You will have to ask yourself questions such as: How much data do I need to store?
- Will my future use cases result in a wide range of applications that run heavy-analysis data?
- What are my storage requirements for incrementally backing up a virtual machine's snapshots?
- Do I need ephemeral storage?.

Logical Model Design

Network

- OpenStack has moved from simplistic network features to more complicated ones, but of course the reason is that it offers more flexibility! This is why OpenStack is here. It brings as much flexibility as it can! Without taking any random network-related decisions, let's see which network modes are available. We will keep on filtering until we hit the first correct target topology

Network mode	Network Characteristics	Implementation
nova-network	Flat network design without tenant traffic isolation	nova-network Flat DHCP
	Isolated tenants traffic and predefined fixed private IP space size Limited number of tenant networks (4K VLANs limit)	nova-network VLANManager
Neutron	Isolated tenants traffic Limited number of tenant networks (4K VLANs limit)	Neutron VLAN
	Increased number of tenant networks Increased packet size Lower performance	Neutron tunneled networking (GRE, VXLAN, and so on)

Physical network layout Components

The tenant data network

- A tenant is **a group of users who share a common access with specific privileges to the software instance.**
- The main feature of a data network that it provides the physical path for the virtual networks created by the OpenStack tenants.
- It separates the tenant data traffic from the infrastructure communication path required for the communication between the OpenStack component itself.

Management and the API network

- In a smaller deployment, the traffic for management and communication between the OpenStack components can be on the same physical link.
- This physical network provides a path for communication between the various OpenStack components such as REST API access and DB traffic, as well as for managing the OpenStack nodes.
- For a production environment, the network can be further subdivided to provide better isolation of traffic and contain the load on the individual networks.

Physical network layout Components

The Storage network

- The storage network provides physical connectivity and isolation for storage-related traffic between the VMs and the storage servers.
- As the traffic load for the storage network is quite high, it is a good idea to isolate the storage network load from the management and tenant traffic

Virtual Network types

- **The external network** – The features of an external or a public network are as follows: It provides global connectivity and uses routable IP addressing
- It is used by the virtual router to perform SNAT from the VM instances and provide external access to traffic originating from the VM and going to the Internet [SNAT refers to Source Network Address Translation. It allows traffic from a private network to go out to the Internet. OpenStack supports SNAT through its Neutron APIs for routers.]
- **The tenant networks**
- The features of the tenant network are as follows:
- It provides a private network between virtual machines It uses private IP space
- It provides isolation of tenant traffic and allows multi-tenancy requirements for networking services

The physical model design

- Hardware commodity selection will accomplish the mission of our massive scalable architecture.
- Estimating the hardware capabilities
  - CPU calculations
  - Memory calculations
  - Network calculations
  - Storage calculations

✓ See Textbook for calculations

Previous Year University Question Paper 2021

Part A

- Define cloud computing and its service models. (3)
- What are the main components of Neutron architecture? Explain each.(3)

Part B

- Explain OpenStack cloud architecture and any 4 service components. (6)
- Explain Nova compute service and its basic components. (6)

Further Reading

- How OpenStack's Keystone handles authentication and authorization
- OpenStack Services
- OpenStack Tutorial – An Overview
- IaaS, PaaS, SaaS
- IaaS, PaaS, SaaS

Thank You