Overview of OpenStack

Module I

Mastering OpenStack

Ms. Navyamol K T

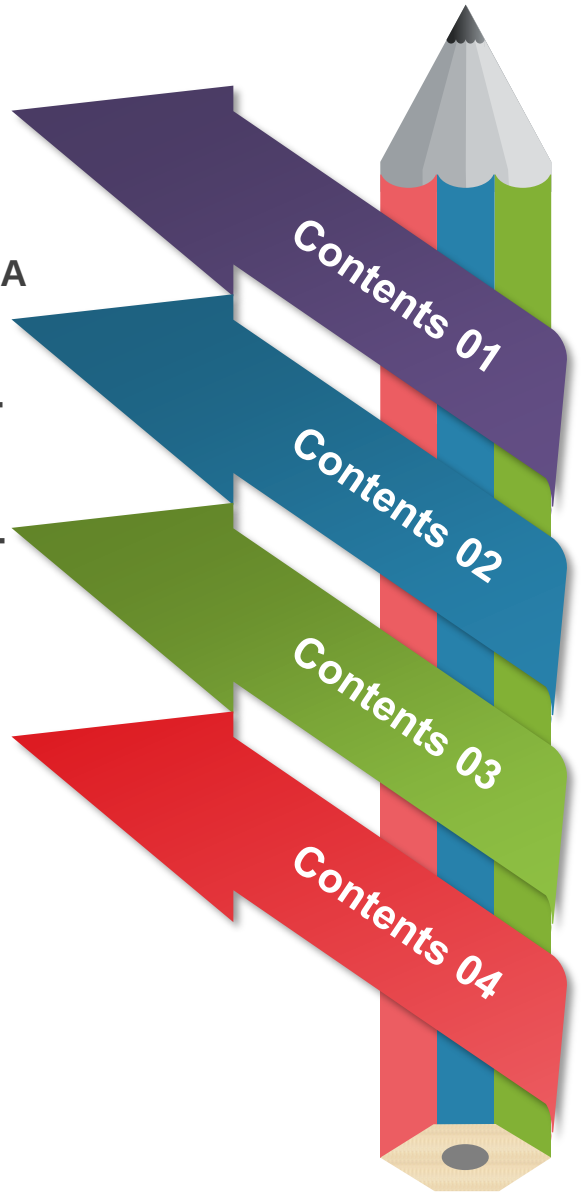
Asst. Professor

Dept. of MCA

AJCE

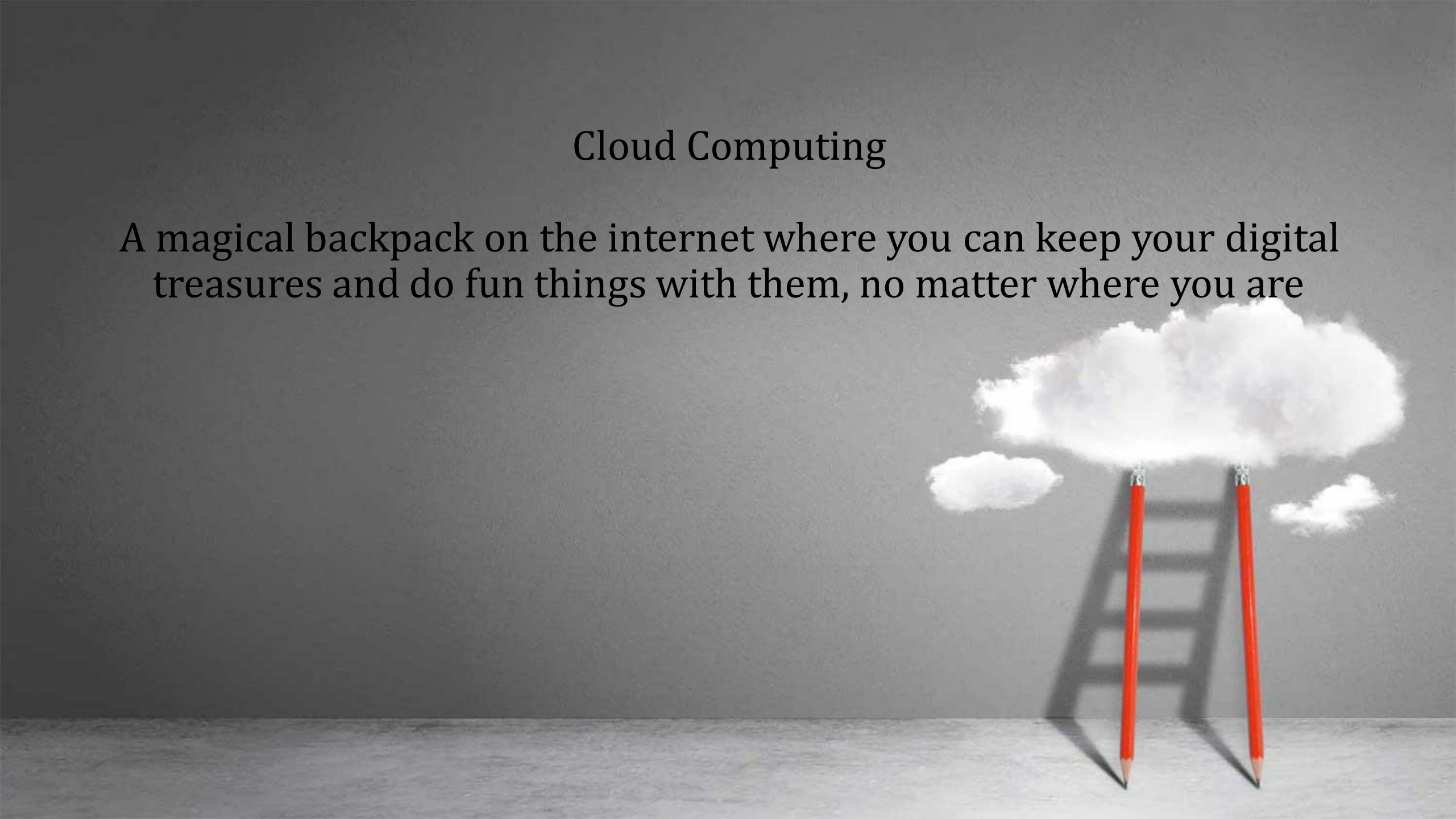
CONTENT

- 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.



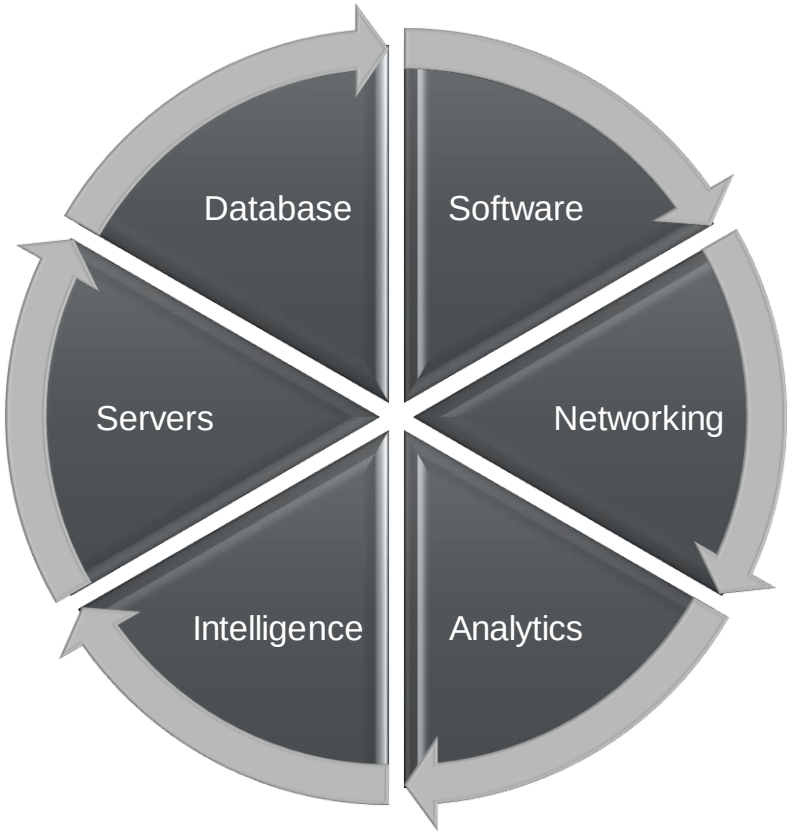
Cloud Computing

A magical backpack on the internet where you can keep your digital treasures and do fun things with them, no matter where you are

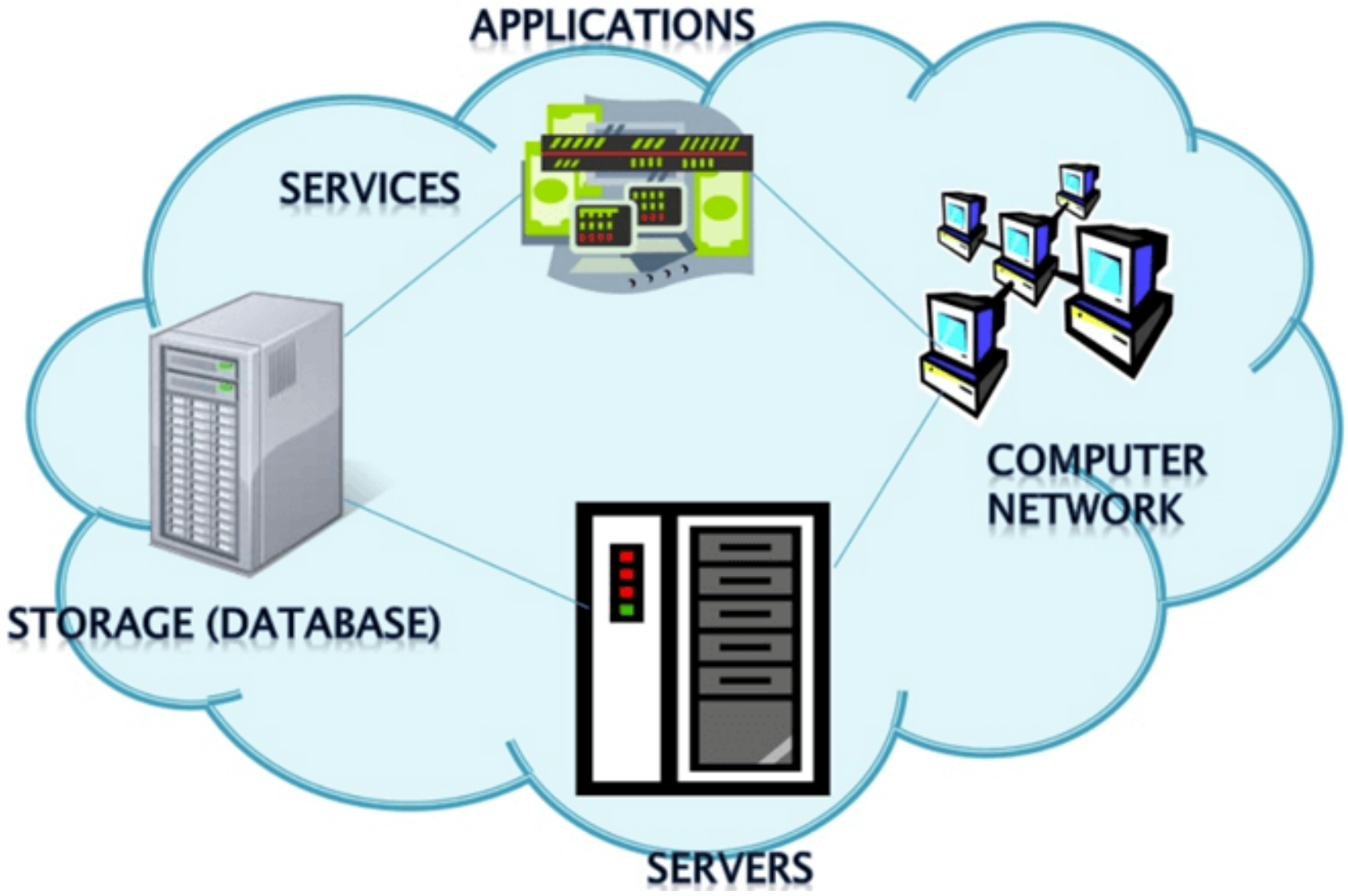


INTRODUCTION TO CLOUD COMPUTING

- 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

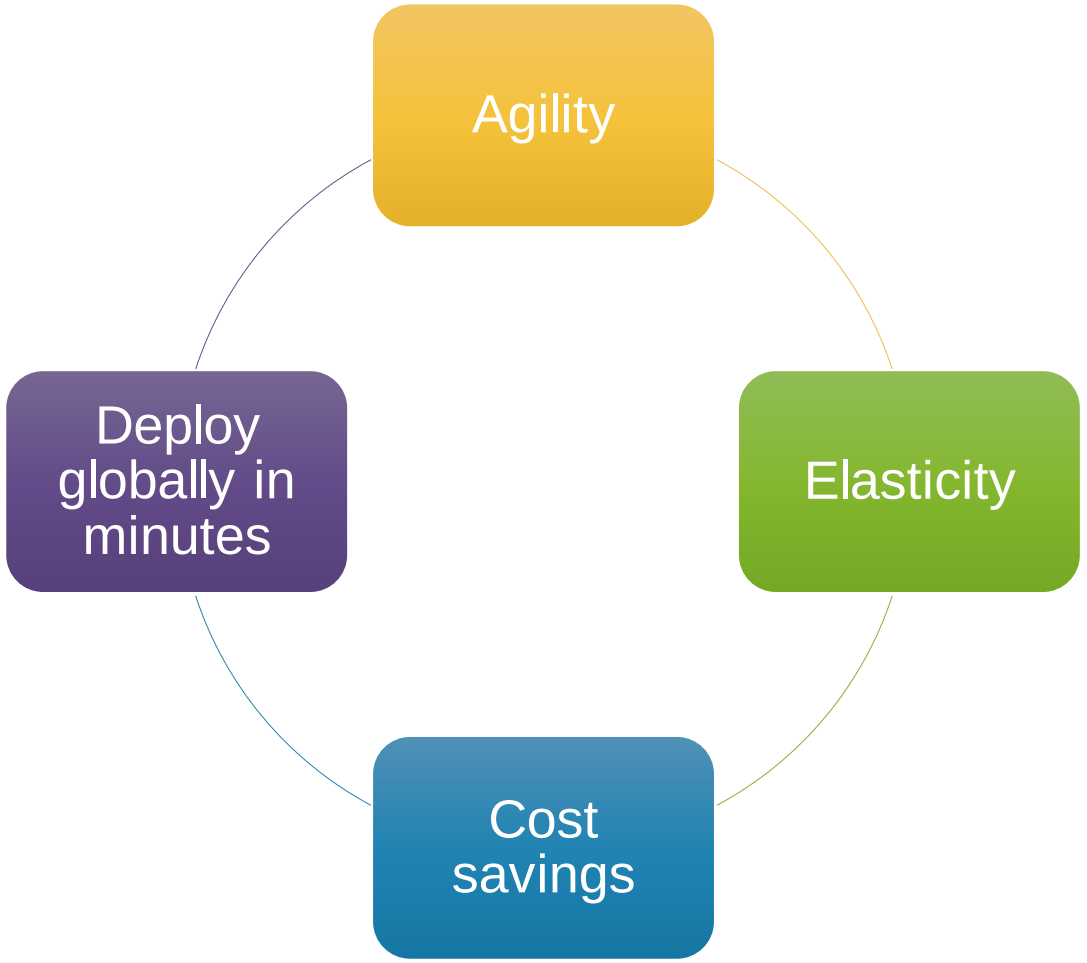
- Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing.
- Instead of buying, owning, and maintaining **physical data centers and servers**, you can **access technology services**, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

INTRODUCTION TO CLOUD COMPUTING

- **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



Benefits of Cloud Computing

- Cost-Efficiency:
- Example: Imagine you run a small business and need computer servers to host your website. Instead of buying expensive servers and paying for their maintenance, you can use cloud services like Amazon Web Services (AWS) or Google Cloud. You only pay for what you use, which can be much cheaper in the long run.
- Scalability:
- Example: If you're a game developer and suddenly your game becomes super popular, you'll need more computing power to handle all the players. Cloud platforms allow you to easily scale up your servers to meet the increased demand. When the buzz dies down, you can scale them back down to save money.
- Flexibility and Accessibility:
- Example: With cloud storage services like Dropbox or Google Drive, you can access your documents, photos, and videos from any device with an internet connection. It's like having your important files in your virtual backpack that you can open from anywhere.
- Reliability and Redundancy:
- Example: Cloud providers store your data in multiple data centers. If one center has a problem, your data is still safe in another. This is like having a backup of your favorite game so you don't lose your progress if your computer crashes.

Benefits of Cloud Computing

- Automatic Updates and Maintenance:
- Example: Cloud service providers handle software updates and maintenance for you. For instance, when you use Gmail or Microsoft 365, you always have the latest features and security patches without needing to do anything yourself.
- Collaboration:
- Example: Suppose you and your classmates are working on a group project. With cloud-based tools like Google Docs, you can all edit the same document at the same time from different devices. No need to email files back and forth.
- Security:
- Example: Cloud providers invest heavily in security. They use advanced encryption and have teams of experts to protect your data. This is like having a superhero squad guarding your digital treasures.

Benefits of Cloud Computing

- Environmental Benefits:
- Example: When you use cloud services, you share resources with others. This means fewer physical servers need to be produced and maintained, which reduces the carbon footprint. It's like helping the planet while you work and play online!
- Global Reach:
- Example: If you're a small online store, you can use a cloud-based e-commerce platform like Shopify to sell your products to customers worldwide. You don't need to worry about setting up physical stores in different countries.
- Backup and Disaster Recovery:
- Example: Imagine you accidentally delete your important school project. If it's saved on a cloud service like iCloud or OneDrive, you can easily recover it from the backup, just like a magician pulling a rabbit out of a hat.

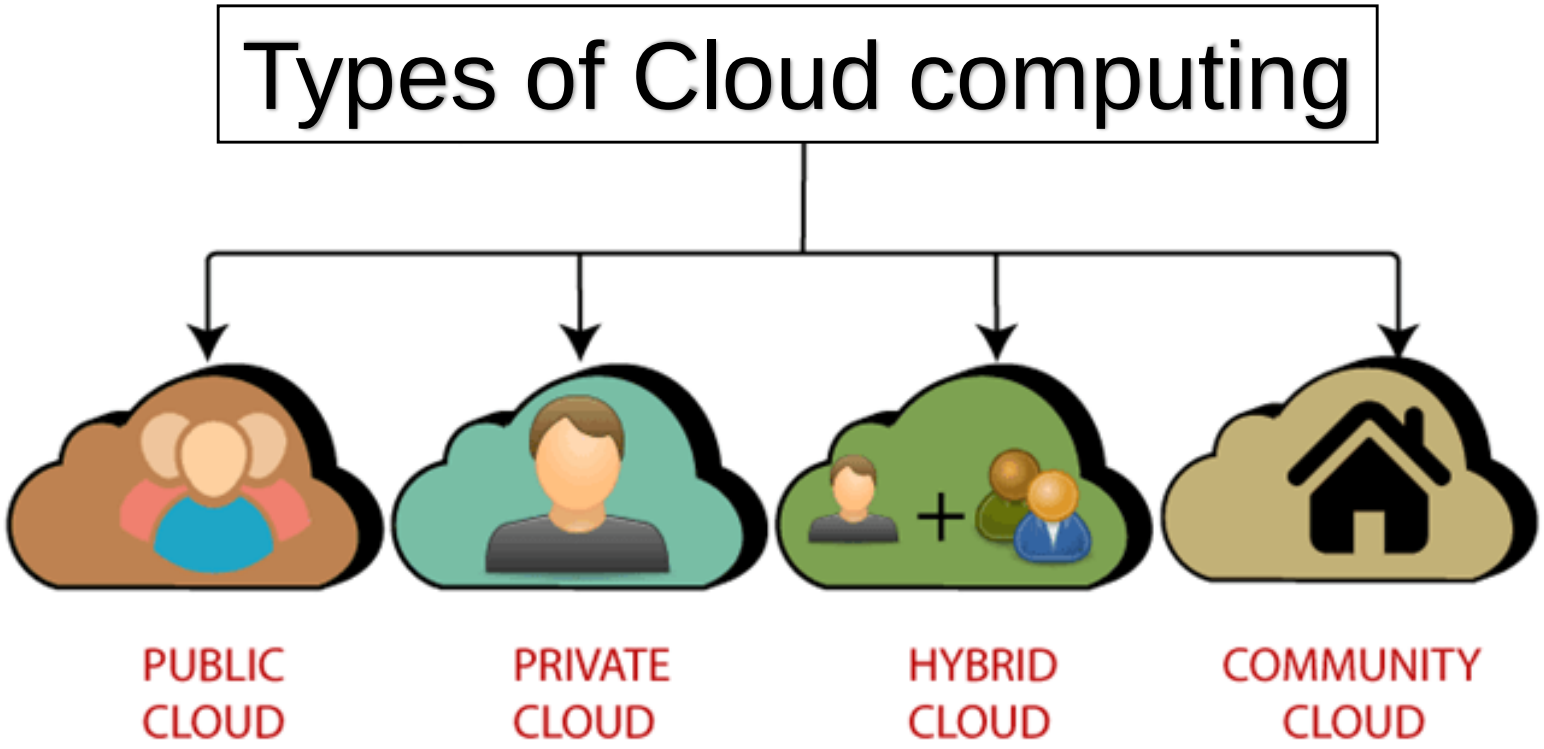
INTRODUCTION TO CLOUD COMPUTING

- 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

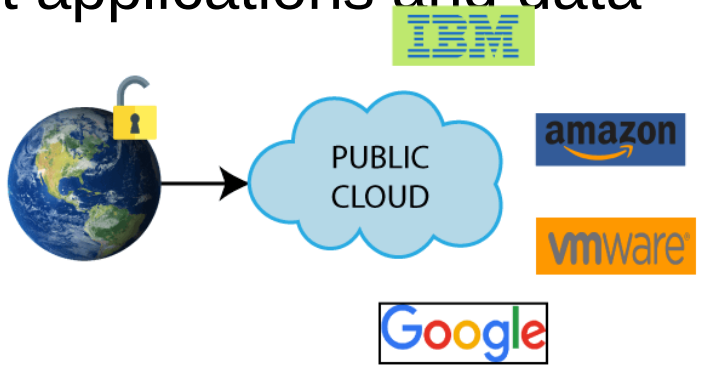
- Benefits of Cloud Computing
  - Cost savings
  - The cloud allows you to trade capital expenses (such as data centers and physical servers) for variable expenses, and only pay for IT as you consume it. Plus, the variable expenses are much lower than what you would pay to do it yourself because of the 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.
  - *Amazon Web Services (AWS)*: AWS is one of the largest and most well-known public cloud providers. Organizations can use AWS services like Amazon EC2 (virtual servers) and Amazon S3 (object storage) to host applications and data in a public cloud environment.



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

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

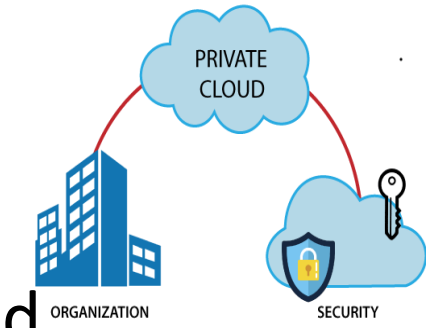
PUBLIC CLOUD

- 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
- **Example**:
- *VMware vSphere/ESXi*: Many enterprises use VMware's virtualization technology to create private cloud environments within their data centers. They can allocate resources dynamically and manage their infrastructure efficiently.

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

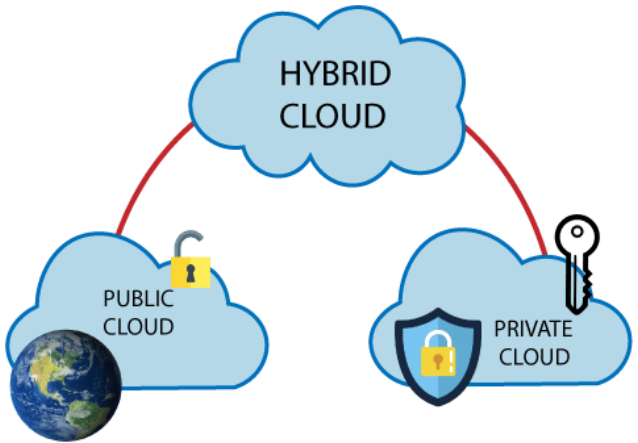
- Private cloud - These technologies include the following: –
- **Virtualization**, which enables IT resources to be abstracted from their underlying physical hardware and pooled into unbounded resource pools of computing, storage, memory, and networking capacity that can then portioned among multiple 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
- *Microsoft Azure Hybrid Cloud*: Microsoft Azure offers a hybrid cloud solution that integrates on-premises infrastructure with Azure public cloud services. This allows organizations to run applications in their data centers while utilizing Azure for additional computing power or specialized services as needed.



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

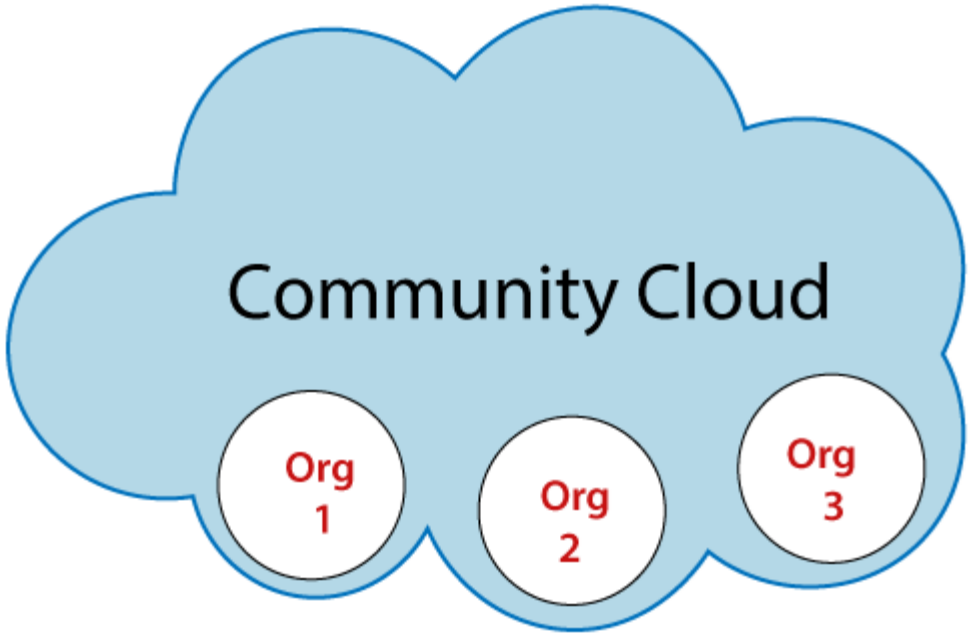
- Networking issues – In the Hybrid Cloud, networking becomes complex because of the private and the public cloud.
- Infrastructure Compatibility – Infrastructure compatibility is the major issue in a hybrid cloud. With dual-levels of infrastructure, a private cloud controls the company, and a public 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

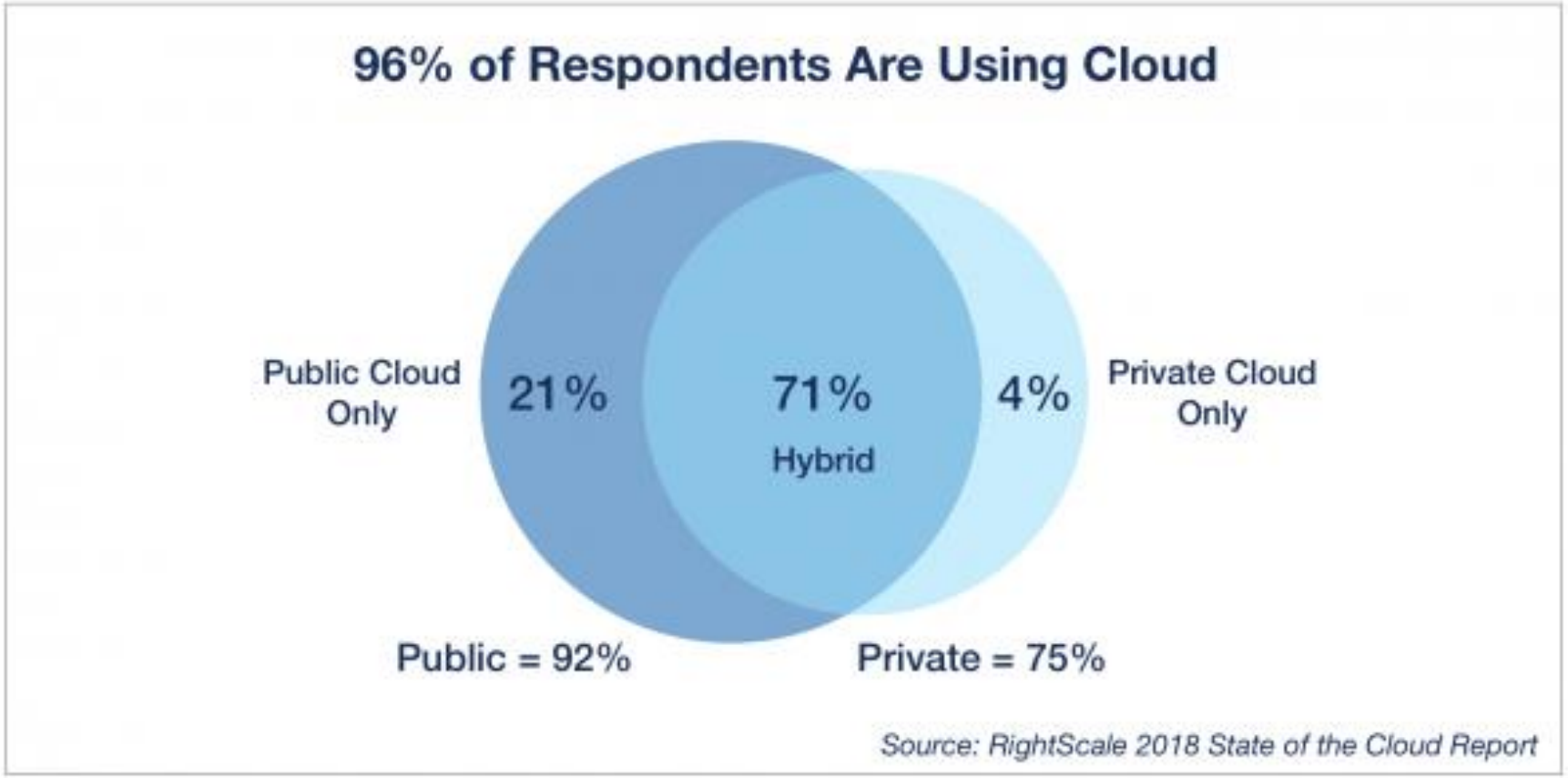
- Dynamic or frequently changing workloads.
- Separating critical workloads from less-sensitive workloads.
- Big data processing.
- Moving to the cloud incrementally, at your own pace
- Temporary processing capacity needs.
- Flexibility for the future
- Best of both worlds

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):

- What it is: Think of IaaS as renting a virtual computer or server in the cloud. It's like having a powerful computer that you can use for anything you want.
- Example: Imagine you need a computer to run a special program, but you don't want to buy a new one. With IaaS, you can rent a virtual computer in the cloud from a service like Amazon Web Services (AWS) or Microsoft Azure. You can install your software, use it, and when you're done, you return it, like borrowing a book from a library.

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):

- What it is: PaaS is like having a ready-made kitchen to cook in. You don't need to worry about buying or setting up the stove, fridge, or utensils; you just start cooking your recipe.
- Example: Let's say you want to build a website or an app. With PaaS, you use a platform like Heroku or Google App Engine. They provide the tools and environment you need to create your website or app. You focus on your code and content, while PaaS takes care of the technical kitchen stuff.

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):

- What it is: SaaS is like using an app or software on your phone or computer, but it's not actually installed on your device. Instead, it runs on the internet, and you access it through a web browser.
- Example: Think about using Gmail or Microsoft Office 365. You don't have to download or install them on your computer. You simply log in through a web browser, and you can use your email or create documents right there. It's like using a magic portal to access your favorite tools and apps from anywhere.

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.

IaaS is like renting a virtual computer.

PaaS is like using a ready-made kitchen to cook your software.

SaaS is like using apps and software through a magical internet portal.

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.
- Think of Storage as a Service like renting a big digital closet in the cloud. Instead of keeping all your important things like photos, videos, and documents on your own computer, you can store them in this virtual closet on the internet. This closet is super secure and can be as big as you need it to be.

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.

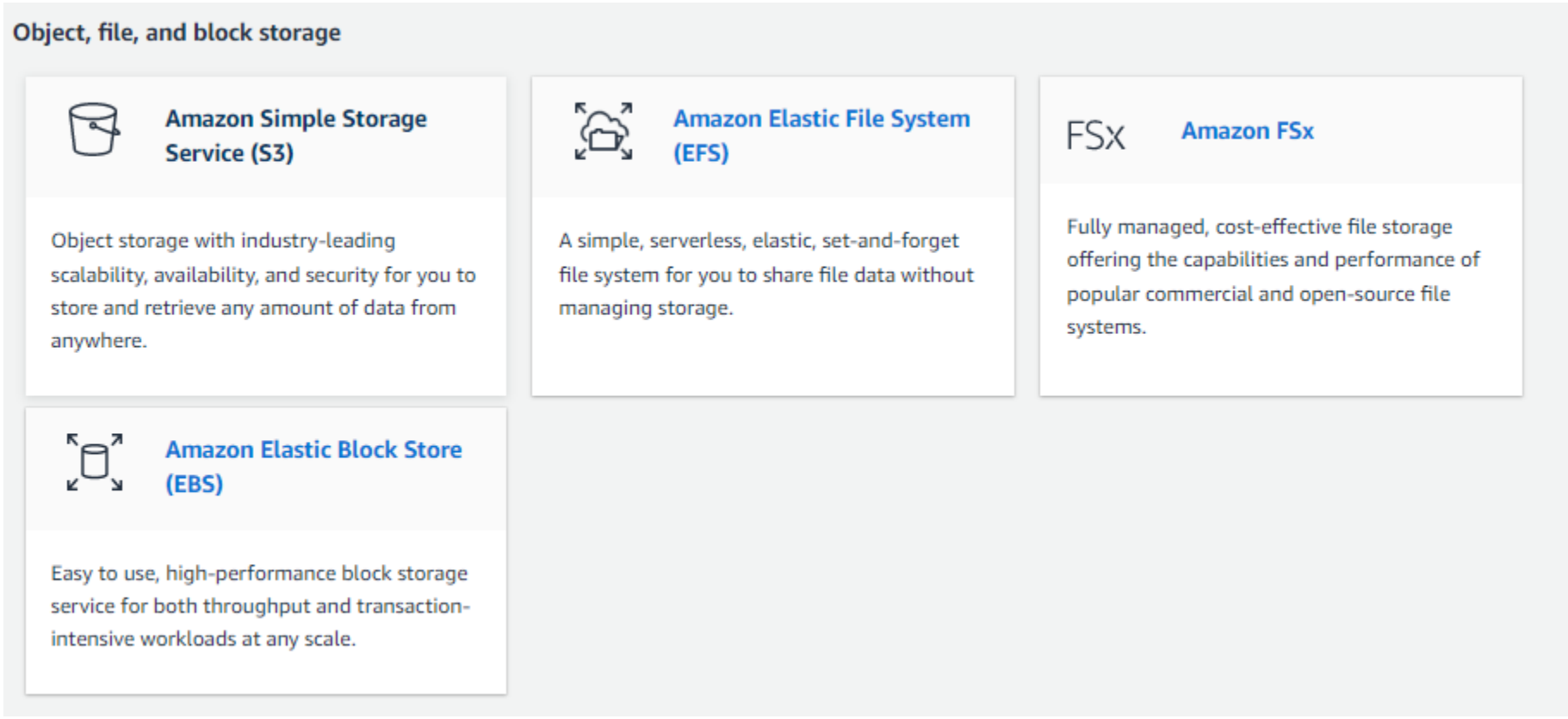
Cloud Block Storage

Cloud Object Storage

Cloud File Storage

INTRODUCTION TO CLOUD COMPUTING

- 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.

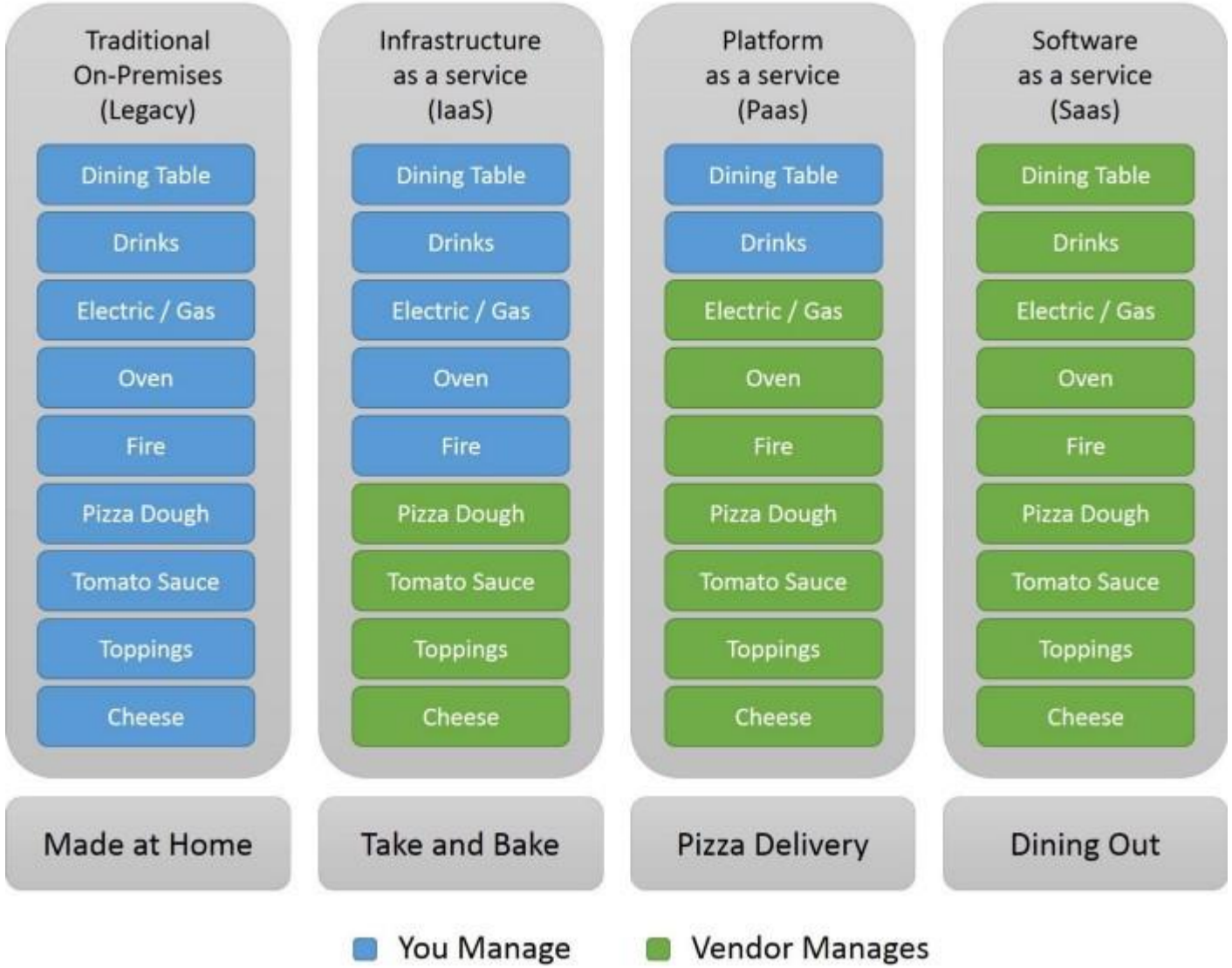
Difference Between IAAS, PAAS and SAAS

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.

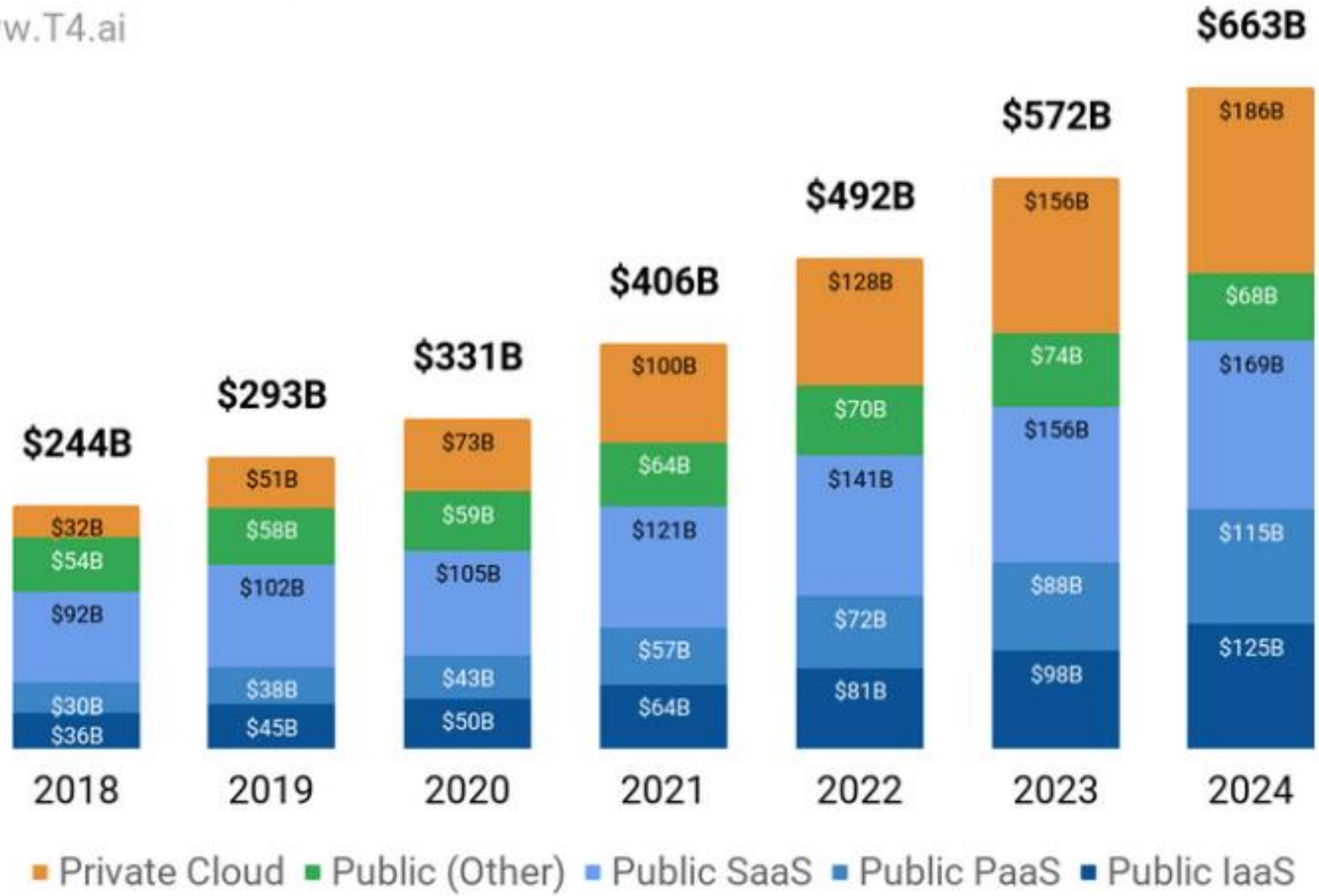
Pizza as a Service



Click on the image

Cloud Computing Market Size, 2018-2024

www.T4.ai



INTRODUCTION TO OPENSTACK

OPENSTACK

- **What is OpenStack?**

- OpenStack is like a super-manager for computer, storage, and network resources in big data centers.

- **How It Works:**
- It's free and open-source software that acts like a control system, making it easier to manage all these resources.

- **In Simple Terms:**
- Imagine it as the ultimate remote control for everything in a giant computer hub, making cloud computing much easier.

OPENSTACK LOGICAL ARCHITECTURE

9 Components

INTRODUCING THE OPENSTACK LOGICAL ARCHITECTURE

- **Getting Started:**
- Before diving into OpenStack's structure, it's important to understand the basic concepts of its core components.

- **Understanding the Basics:**
- We'll break down and explain the fundamental concepts of each component to grasp how they function.

- **Working Together:**
- OpenStack services collaborate to create a seamless cloud experience. Despite their varied purposes, they share a common design theme, ensuring a unified approach.

INTRODUCING THE OPENSTACK LOGICAL ARCHITECTURE

Developed in Python:

- OpenStack services are created using Python, allowing for quick development and updates.

REST APIs:

- All OpenStack services have REST APIs. These are like special phone lines that allow services and users to talk to each other.

Modular Implementation:

- OpenStack services are built in different parts (components). These parts communicate through a message system, making sure they work together smoothly. It's like team members passing notes to coordinate tasks.

OpenStack Architecture

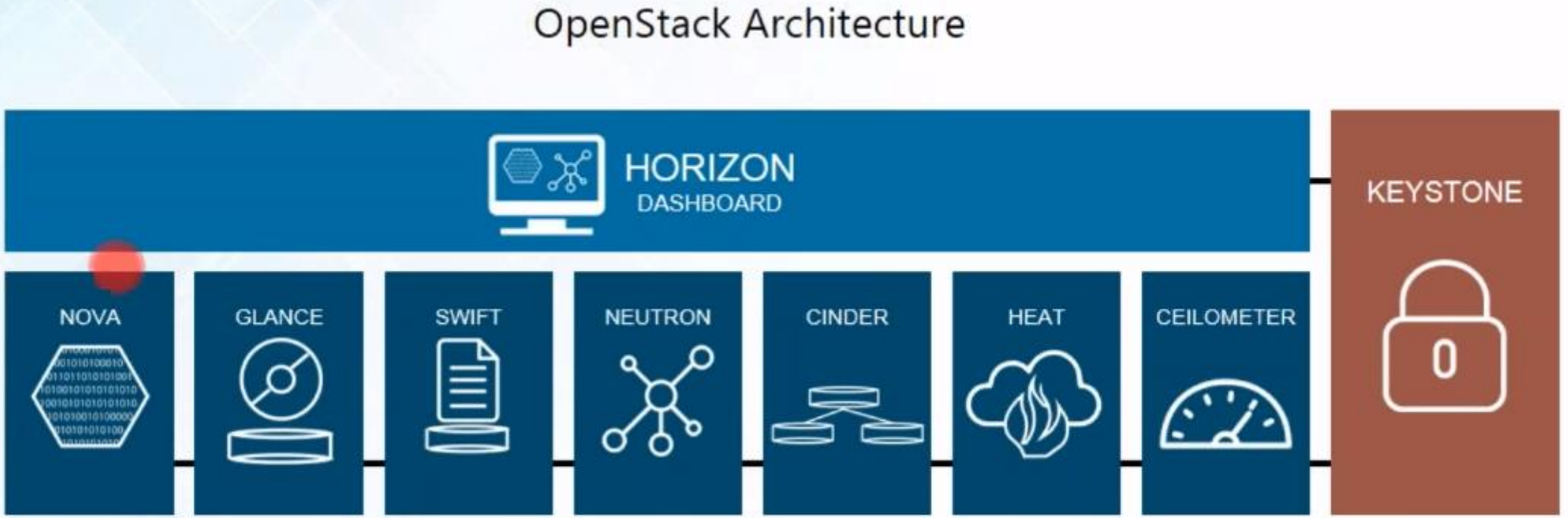


Figure: *OpenStack Architecture*

Horizon (Dashboard):

Horizon is like the control center of the cloud. It gives you a way to see what's happening in your cloud and make changes, like turning on or off a light switch in your room.

Keystone (Identity):
Keystone is like your secret key to the cloud.
It makes sure only the right people can use the cloud.
It's like having a special key to open your diary.

Nova (Compute):

Think of Nova as the brain of the cloud.

It manages the computers in the cloud.

These computers are called virtual machines (VMs).

You can create, start, stop, and delete these VMs

Glance (Image Service):

Glance is like a magical camera that takes pictures of your VMs.

You can make copies of these pictures and use them to create new VMs

Swift (Object Storage):

Swift is like a treasure chest that can hold lots of things, like photos and videos.

It keeps your treasures safe and easy to find, just like a treasure map.

Neutron (Networking):
Imagine Neutron as the roads and highways of the cloud.
It helps VMs talk to each other and connect to the internet.
It's like building roads and bridges so that people and cars can go where they need to.

Cinder (Block Storage):

Cinder is like a magical box that stores your important stuff, like pictures and documents.

You can make this box bigger or smaller whenever you want

Heat (Orchestration):

Heat is like a conductor in an orchestra.

It helps you create and manage many VMs at once.

You can tell it to play your favorite song, and it will set up all the instruments for you.

Ceilometer (Telemetry):

Ceilometer is like a weather station for your cloud. It watches what's happening and tells you how hot or cold things are, just like a thermometer tells you the temperature outside.


Imagine you're building a virtual amusement park in the cloud called **'CloudLand.'**
How does each of these OpenStack components play a role in creating and managing **CloudLand?**

Keystone - identity management

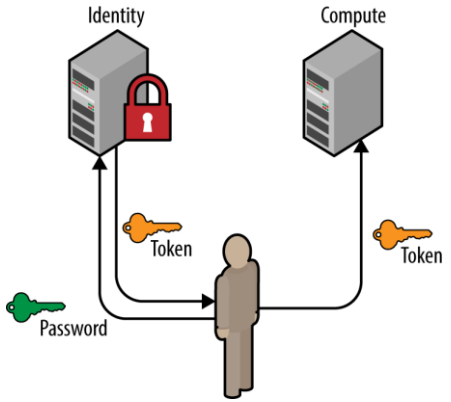
- Keystone presents the simplest service in the OpenStack composition.
- It is the core component and provides an identity service comprising authentication and authorization of tenants in OpenStack.
- Authentication: Keystone checks your ID (username and password) to confirm your identity before you enter the club. It makes sure you're a valid user of the OpenStack cloud.
- Authorization: Once inside, Keystone assigns you a role, like VIP or regular guest, to determine what you're allowed to do in the club. This controls your access to different parts of the OpenStack services.
- Service Directory: Keystone keeps a list of all the services available in the club, like the dance floor (Nova), the bar (Neutron), and the private lounge (Cinder). It helps you find these services so you can use them.
- Token: After confirming your identity, Keystone gives you a special wristband (token) that you need to show to the other services to prove you're allowed in. This wristband has an expiration time.

Keystone - Services

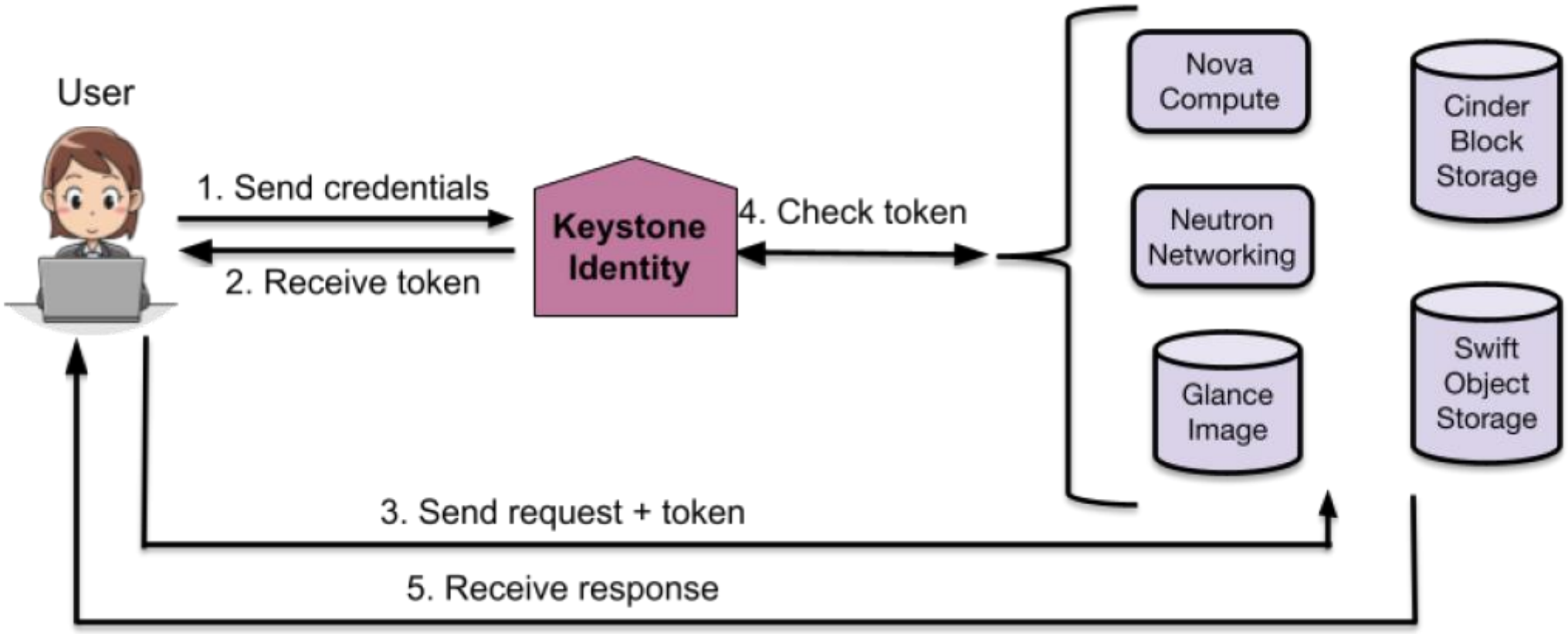
- **User Management**: Keystone manages the guest list, adding and removing people (users) and organizing them into groups (like organizing guests into tables or sections).
- **Security**: Keystone ensures that only authorized people get into the club and that they can only access what they're allowed to. It's like the club's security guard.
- **Communication:** Keystone helps different parts of the club (services) talk to each other by providing addresses (endpoints). It makes sure everyone knows where to find each other.
- **Backend Flexibility**: Keystone can use different methods to check your ID, like checking a list at the door (local database) or calling a central registry (LDAP).
- **Token Control:** If someone loses their wristband (token) or if they misbehave, Keystone can kick them out of the club by invalidating their wristband (token).
- **API**: Keystone has a menu (API) that you can use to talk to the bouncer and request access to the club or ask about the services.

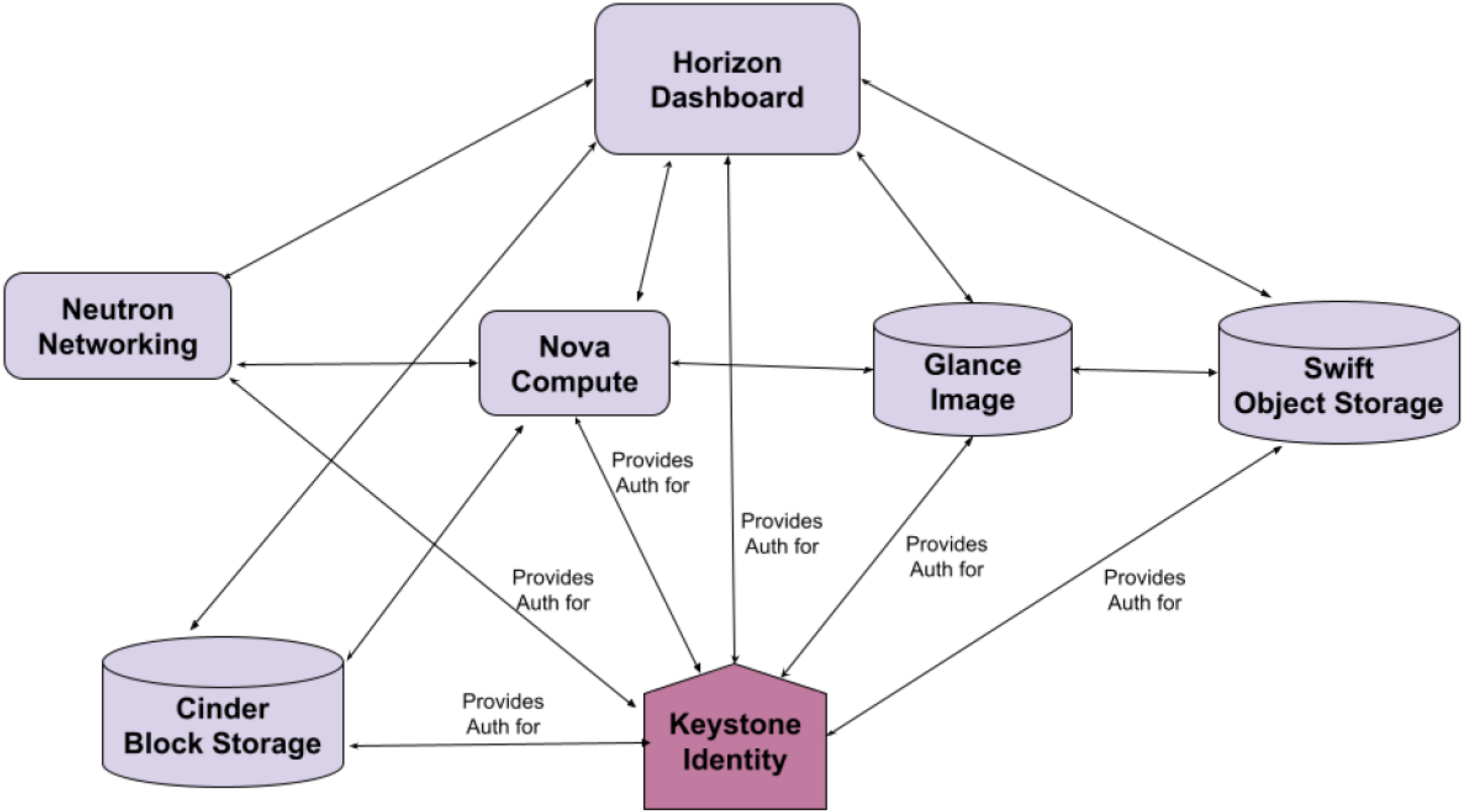
Keystone - identity management

- 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



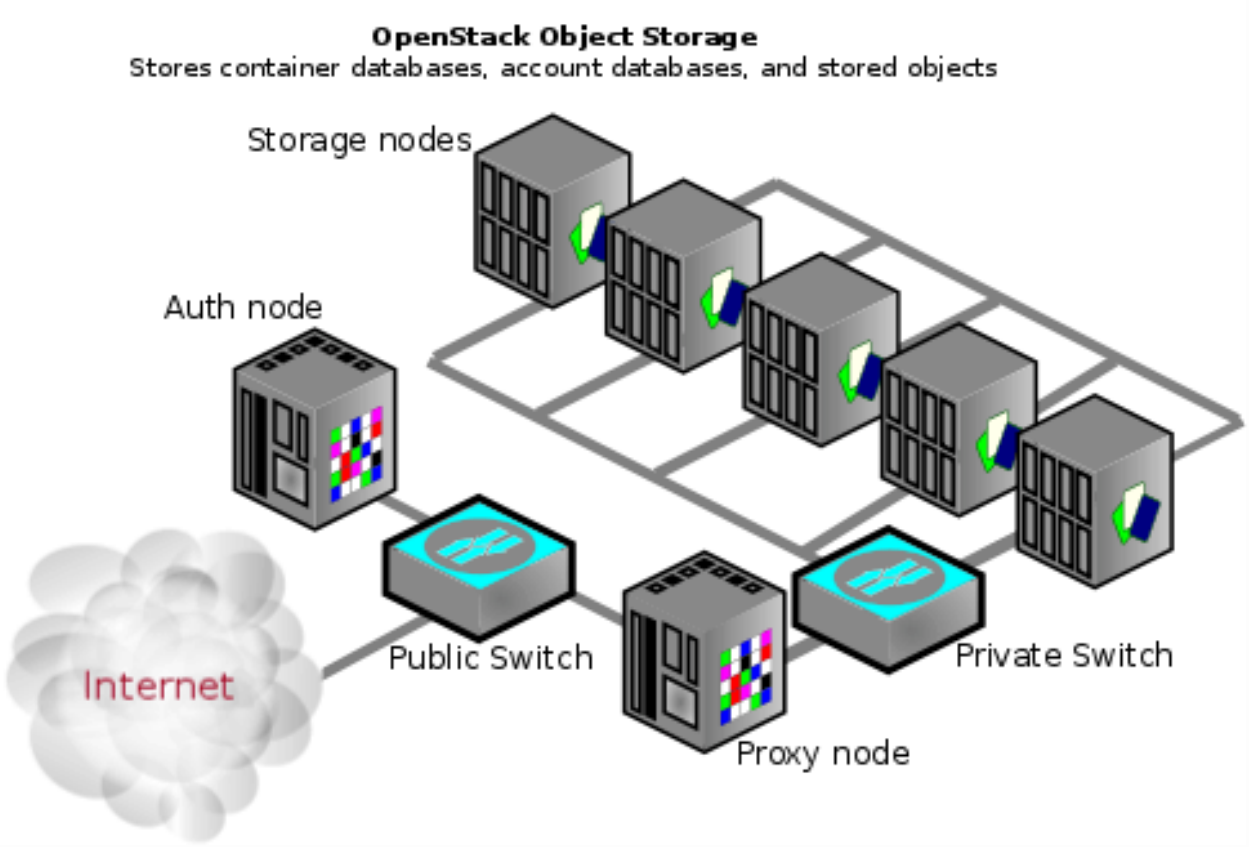

Horizon Dashboard
Neutron Networking
Nova Compute
Glance Image
Swift Object Storage
Cinder Block Storage
Keystone Identity
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for
Provides Auth for

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: –
    a. It has no central brain, and indicates no Single Point Of Failure (SPOF)
    b. It is curative, and indicates auto-recovery in the case of failure.
    c. 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.
    d. It has inexpensive hardware that can be used for redundant storage clusters

Swift - object storage

- 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.



Swift - object storage

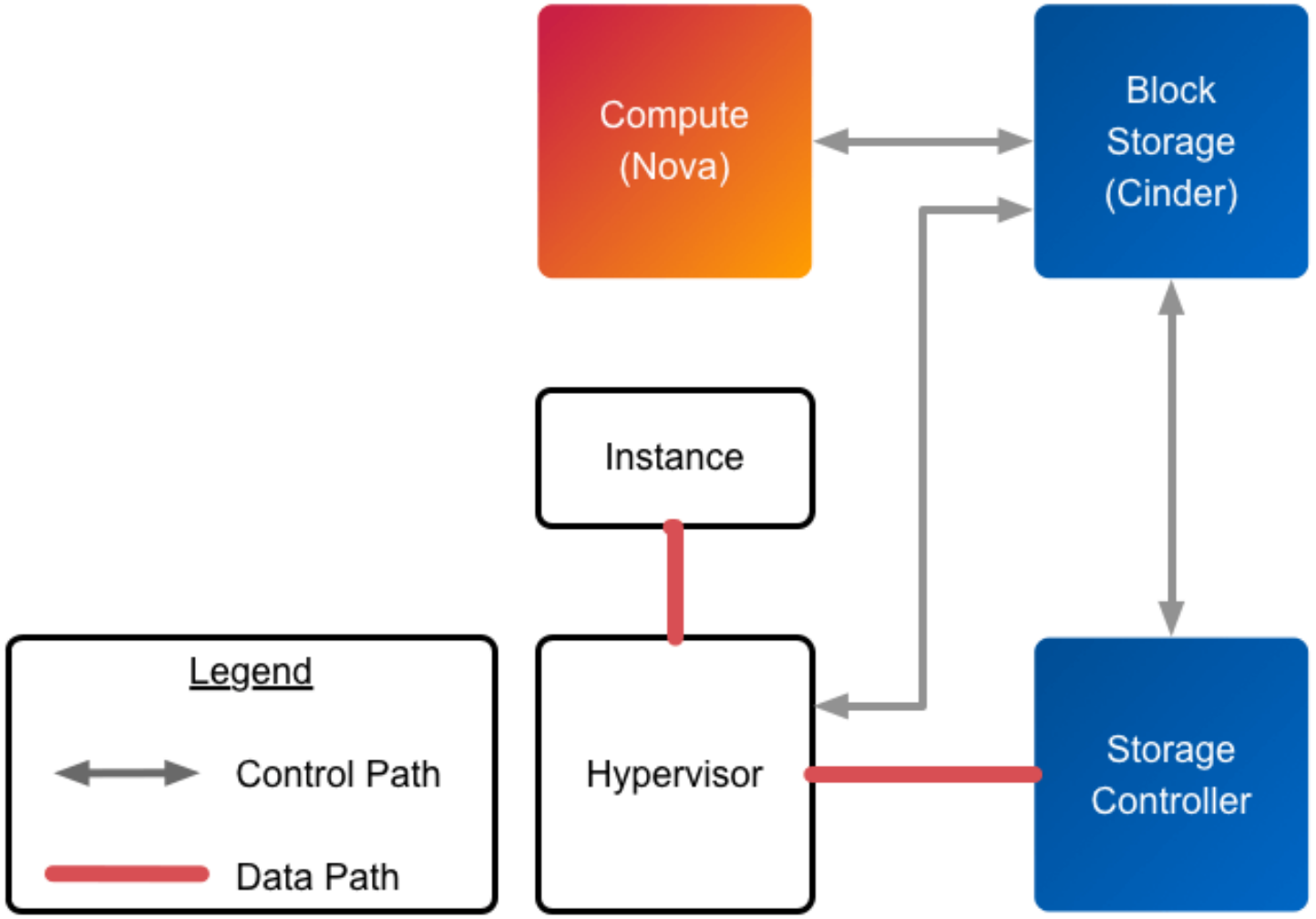
- **Storage:** Imagine a massive warehouse where you can store all your stuff, like files, photos, and videos. Swift provides this storage space.
- **Scalable**: You can add more rooms (servers) to the warehouse whenever you need to store more stuff. It grows as you do.
- **Data Replication**: Swift makes copies of your stuff and spreads them across different rooms, so even if one room has a problem, your stuff is safe in others.
- **Easy Access:** You can easily put stuff in (upload) or take stuff out (download) of the warehouse whenever you want, just like a storage unit.
- **Web Access:** You can access your stuff through a web interface, like a fancy online catalog for your warehouse.
- **Secure:** Your stuff is locked away, and only authorized people (with the right keys) can access it.
- **Data Organization**: Swift helps you organize your stuff with containers and folders, like putting items in boxes and labeling them.
- **High Availability**: Even if some parts of the warehouse are under maintenance, you can still access your stuff because there are copies in other rooms.
- **Scalability**: If you suddenly have a lot more stuff, you can quickly expand the warehouse to fit it all.
- **API:** Swift provides a menu (API) for you to tell the warehouse what you want to store, retrieve, or manage.

Cinder - block storage

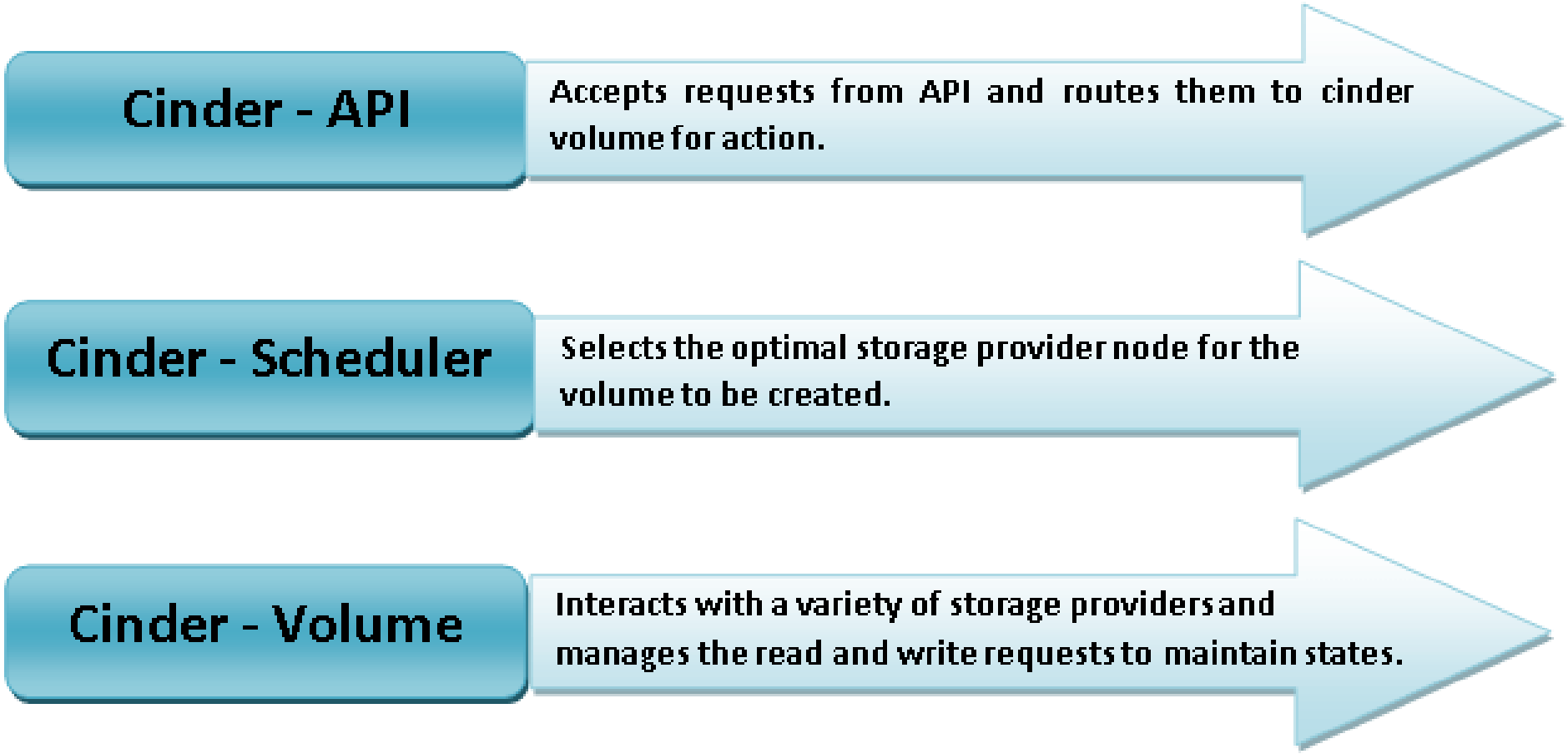
- Cinder is like a virtual hard drive provider:
- Storage: Cinder is like a service that offers virtual hard drives in the cloud. Think of it as renting a virtual storage space.
- **On-Demand Storage:** You can ask Cinder for as much storage space as you need, just like ordering the size of hard drive you want.
- **Attach and Detach:** You can "plug in" (attach) these virtual hard drives to your virtual machines when you need extra space. When you're done, you can "unplug" (detach) them.
- **Snapshots:** Cinder lets you take pictures (snapshots) of your virtual hard drives at a specific moment, so you can go back to that point if something goes wrong.

Cinder - block storage

- **Backup:** You can make copies (backups) of your virtual hard drives to keep your data safe.
- **Scalability:** If you need more storage, you can easily ask for more virtual hard drives or increase the size of the ones you have.
- *Different Types*: Cinder offers different types of virtual hard drives, like fast and expensive ones or slower and more cost-effective ones, so you can choose what's best for your needs.
- **API:** It provides a menu (API) for you to order, manage, and use these virtual hard drives.



Cinder and Nova logical architecture are:



Cinder Components are:

Manila – File share

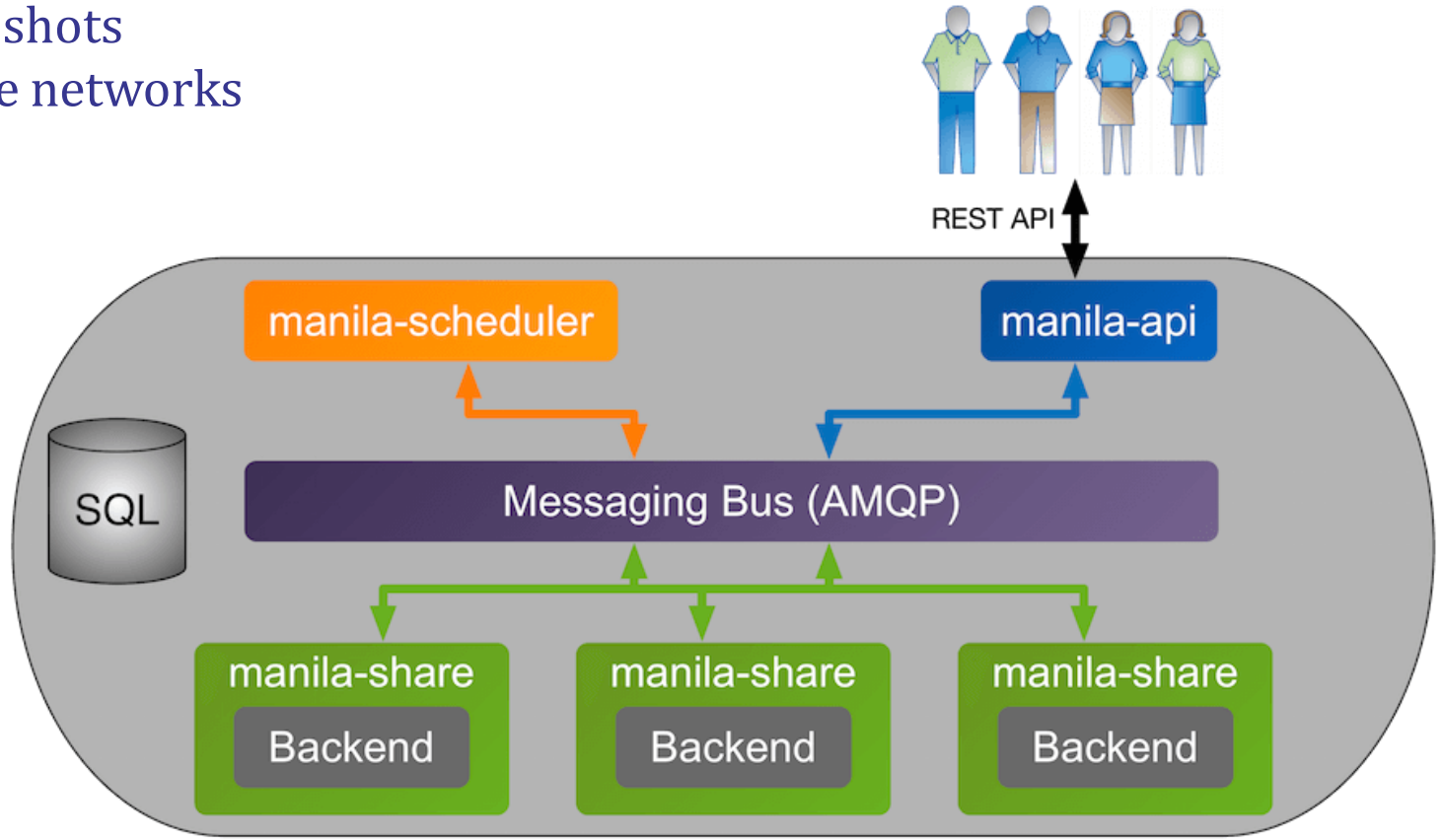
- Manila is like a file sharing service:
- **Shared Folders**: Think of Manila as a service that allows you to create shared folders in the cloud, just like folders on a network drive in an office.
- **Multiple Users**: Many people can access and use these folders at the same time, like sharing files with colleagues in different departments.
- **Protocols**: Manila supports different ways of connecting to these folders, like using Windows file sharing (SMB – Server Message Block) or Network File System (NFS), which are like different types of keys to open the folder.
- **Security:** You can control who can access and edit the files in these folders, just like giving permissions to certain people in your office.
- **Snapshots:** Just like taking a snapshot of a picture, you can take snapshots of these folders at specific moments to make sure you can always go back to a previous version.
- **Scalable:** If you need more storage for your shared folders, you can easily add more, like getting bigger cabinets for your files in a physical office.
- **API:** It provides a menu (API) for you to create, manage, and access these shared folders.

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



Storage Types

- 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

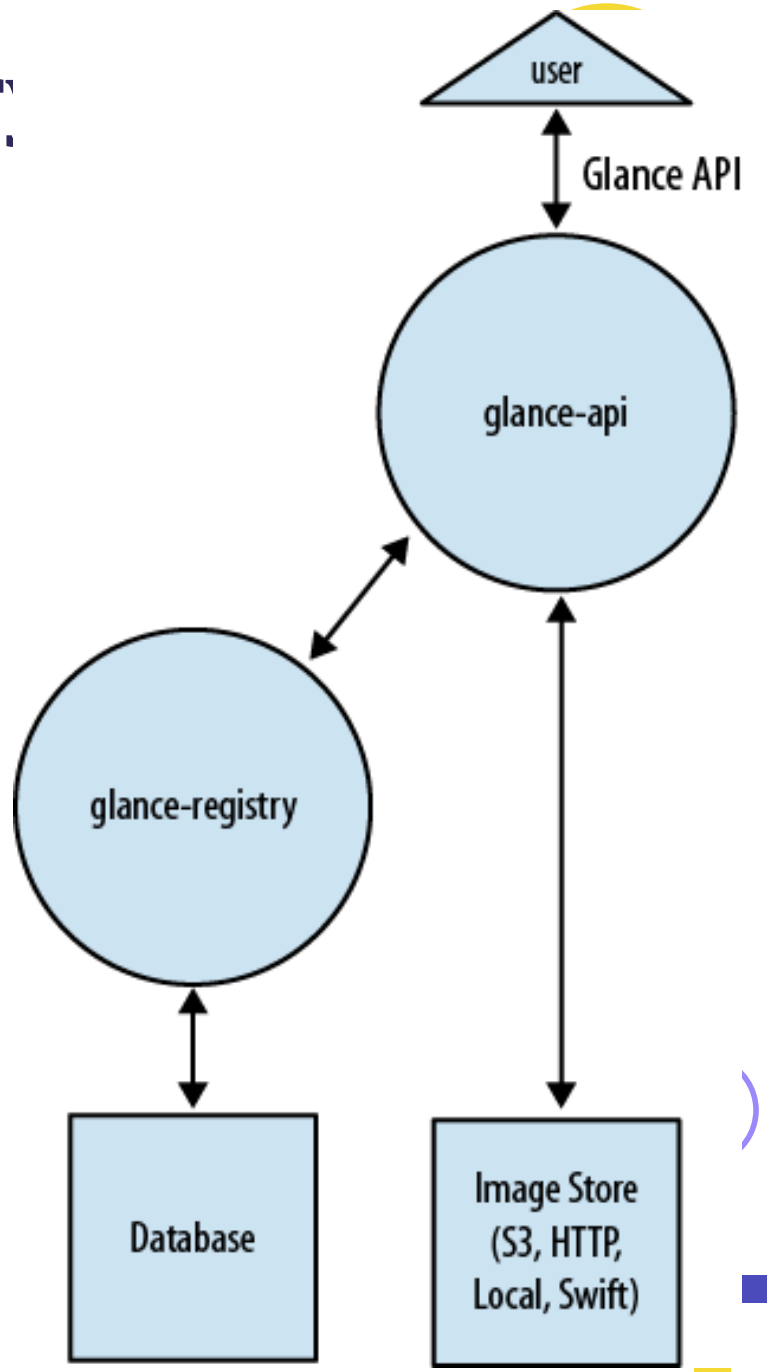
- Glance is one of the core components of OpenStack, and its primary purpose is to manage and store virtual machine (VM) images, such as operating system images and snapshots.
- Glance in OpenStack:
- **Image Repository**: Glance acts as a central repository for managing VM images, templates, and snapshots. These images serve as the building blocks for creating virtual machines in an OpenStack cloud.
- **Storage Abstraction**: It abstracts the underlying storage infrastructure, allowing you to use different storage technologies and backends, such as local file systems, Swift (OpenStack's object storage), or other storage solutions.
- **Image Formats:** Glance supports a variety of image formats, making it flexible for different virtualization technologies and hypervisors, including QEMU, KVM, Xen, and VMware.
- **Image Sharing:** Users can share and distribute images across different projects and tenants within the OpenStack environment, promoting reusability and collaboration.

Glance - Image registry

- **Versioning:** Glance supports image versioning, which enables users to update and manage image revisions while maintaining a history of changes.
- **Metadata:** It allows users to attach metadata to images, providing additional information about the image, its purpose, and its compatibility with specific virtualization environments.
- **Caching:** Glance can cache frequently used images to improve the efficiency of image retrieval and minimize the load on underlying storage systems.
- **API:** Glance provides a RESTful API that allows users and other OpenStack components (such as Nova for compute) to interact with and manage images programmatically.

Glance - Image registr

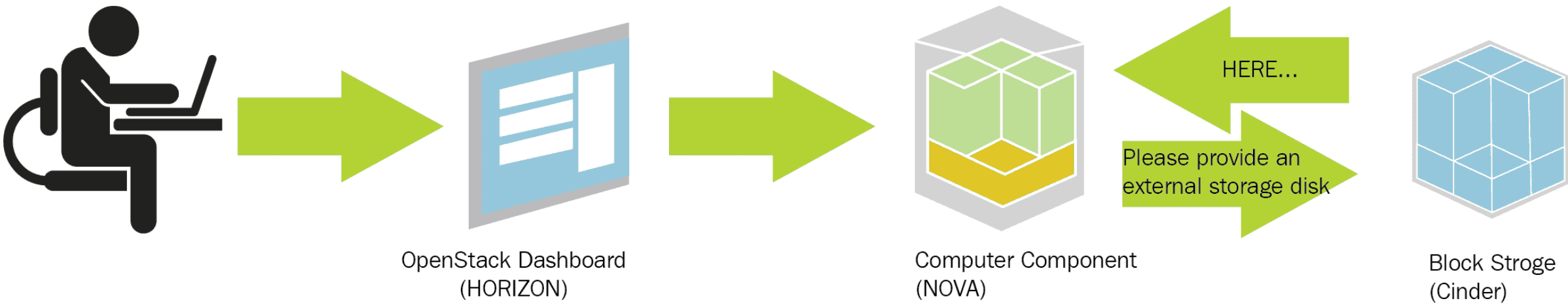
- The glance component is used to provide the image services to OpenStack. Here, image service means the images or virtual copies of hard disks.
- When we plan to deploy a new virtual machine instance, then glance allows us to use these images as templates.
- Glance contains a REST API from which you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



Difference between Cinder and Swift?

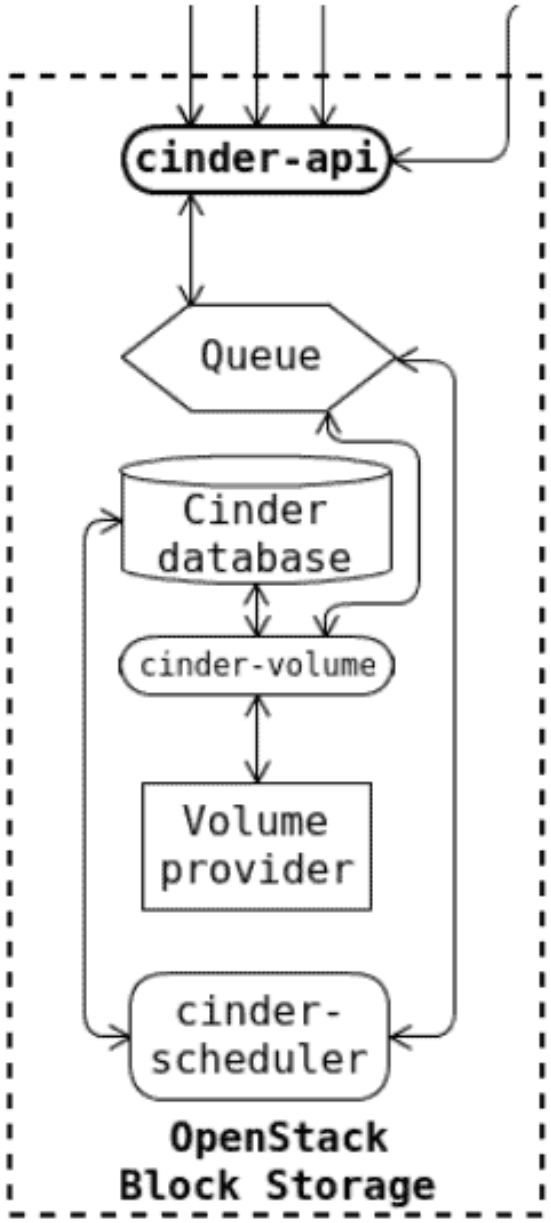
- **Cinder (Block Storage)**
- Cinder is a block storage device.
- Let's suppose it has been a year since Mr. Neo started using his workstation. His daily task saves loads and loads of data onto his hard disk. Mr. Neo noticed that he would run out of his 500 GB internal hard drive space in the next few 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 – COMPUTE SERVICE

- **Nova in OpenStack**
  Nova is a vital part of OpenStack, helping it run smoothly.
- **Complex Design**
  It's quite complex in its structure, making it one of the trickier parts of OpenStack.
- **What Nova Does**
  Nova's main job is to handle the creation and management of virtual machines. It does this in response to what OpenStack users need.
- **Dealing with User Requests**
  Whenever someone using OpenStack wants a virtual machine, Nova jumps in to make it happen.
- **Lots of Connections**
  Nova has to work closely with many other parts of OpenStack. This teamwork is essential for Nova to do its job well, especially when users want to run a virtual machine.

NOVA – Components

- **nova api**

User Interaction

- Nova-api handles user requests and commands. When users or other parts of OpenStack want to create virtual machines, they talk to nova-api.

Creating Instances

- Users use nova-api to create instances (virtual machines) through the OpenStack API or EC2 API. It's the gateway for making new virtual machines.

Orchestrating Actions

- Nova-api takes charge of important tasks like starting a virtual machine or enforcing specific rules.
- It's like the conductor, ensuring everything runs smoothly in the OpenStack environment.

NOVA – Components

- **nova-compute**
- Nova-compute is like a hardworking assistant.
- It creates and stops virtual machines (VMs) using different technologies like XenAPI, Libvirt KVM, or VMware API.
- It's the one responsible for bringing VMs to life and turning them off.
- **nova-network**
- Nova-network is in charge of network tasks.
- It takes tasks from the queue and does things like setting up connections between VMs (bridging interfaces) or changing network rules.
- Think of it as the handy technician making sure all devices can talk to each other on the network.
- Neutron is a replacement for the nova-network service.

NOVA

nova-scheduler

- Nova-scheduler is like a matchmaker.
- It decides which computer should host a new virtual machine.
- It takes requests, looks at available options, and picks the best spot for the VM to run.
- It's like finding the perfect seat in a crowded theater.

nova-conductor

- Nova-conductor is like a secure messenger.
- It helps computers (compute nodes) talk to the database without directly accessing it.
- By doing this, it enhances the security of the database.
- It's similar to passing notes between friends without revealing the secrets to others.

NOVA - Functions

Interacting with OpenStack

- Nova, the heart of OpenStack, talks to different services.

Keystone for Authentication

- Nova interacts with Keystone to confirm the identity of users.
- It's like showing an ID card to enter a secure building.

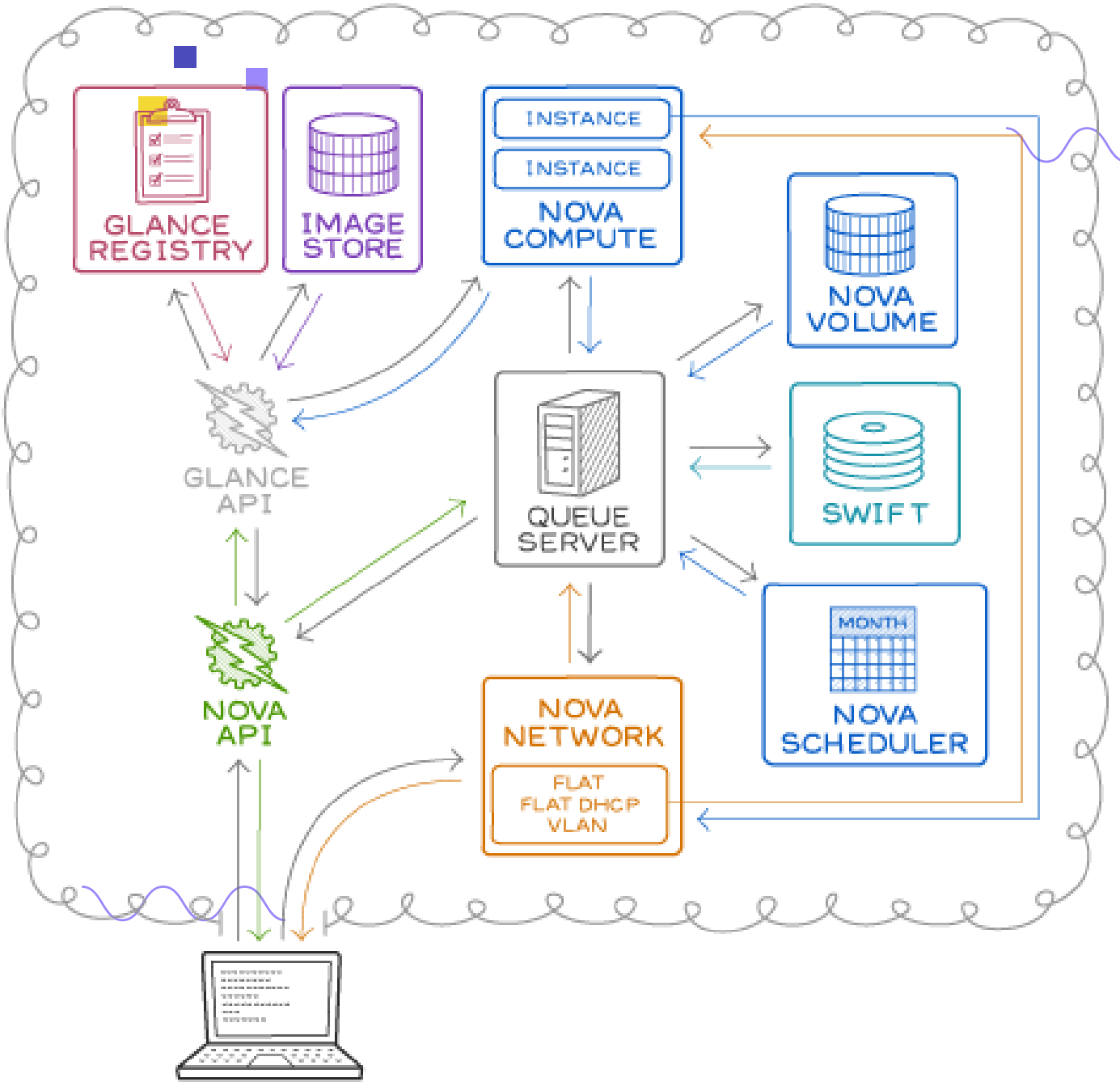
Glance for Images

- Nova works closely with Glance for images.
- It can request images, and nova-compute downloads them to create virtual machines.
- Think of it as getting the blueprint to build something.

Horizon for Web Interface

- Nova communicates with Horizon, the web interface of OpenStack.
- Users interact with Nova through this interface, making it user-friendly, like using a website to control a device.

NOVA


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

Nova –Compute Engine

- **VM Management**: Nova allows users to create, delete, start, stop, and manage virtual machines (VMs) within an OpenStack cloud.
- **Resource Scheduling**: It handles the allocation of physical compute resources (CPU, memory, and storage) to VMs, ensuring that resources are efficiently utilized across the cloud infrastructure.
- **Scaling:** Nova makes it possible to scale up or down the number of VMs based on demand. You can add more VMs when your workload increases and remove them when it decreases.
- **Isolation:** It ensures that VMs are isolated from each other, so they don't interfere with each other's operation or security. Each VM runs in its own virtual environment.

Nova –Compute Engine

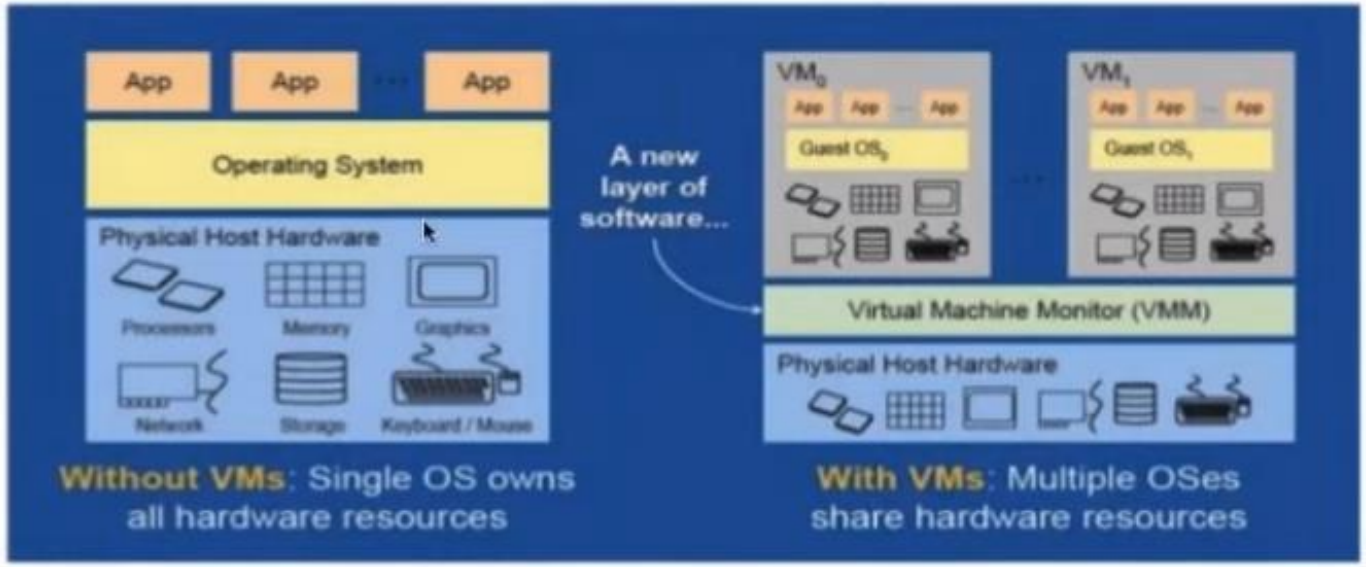
- **API Access**: Nova provides an API (Application Programming Interface) that allows users to interact with and control VMs programmatically. This means you can automate tasks like VM creation and management.
- **Hypervisor Support**: Nova supports multiple hypervisors, which are software or hardware platforms that allow VMs to run. Common hypervisors supported by Nova include KVM, VMware, and Xen.
- **Live Migration:** Nova allows for the live migration of VMs from one physical server to another without interrupting the VM's operation. This is useful for maintenance or load balancing.
- **Load Balancing:** Nova can work in conjunction with other OpenStack components, such as Neutron (for networking) and Cinder (for storage), to create comprehensive cloud solutions.
- **Security**: Nova includes security features to protect VMs and the cloud environment, such as user authentication and role-based access control.

HyperVisor

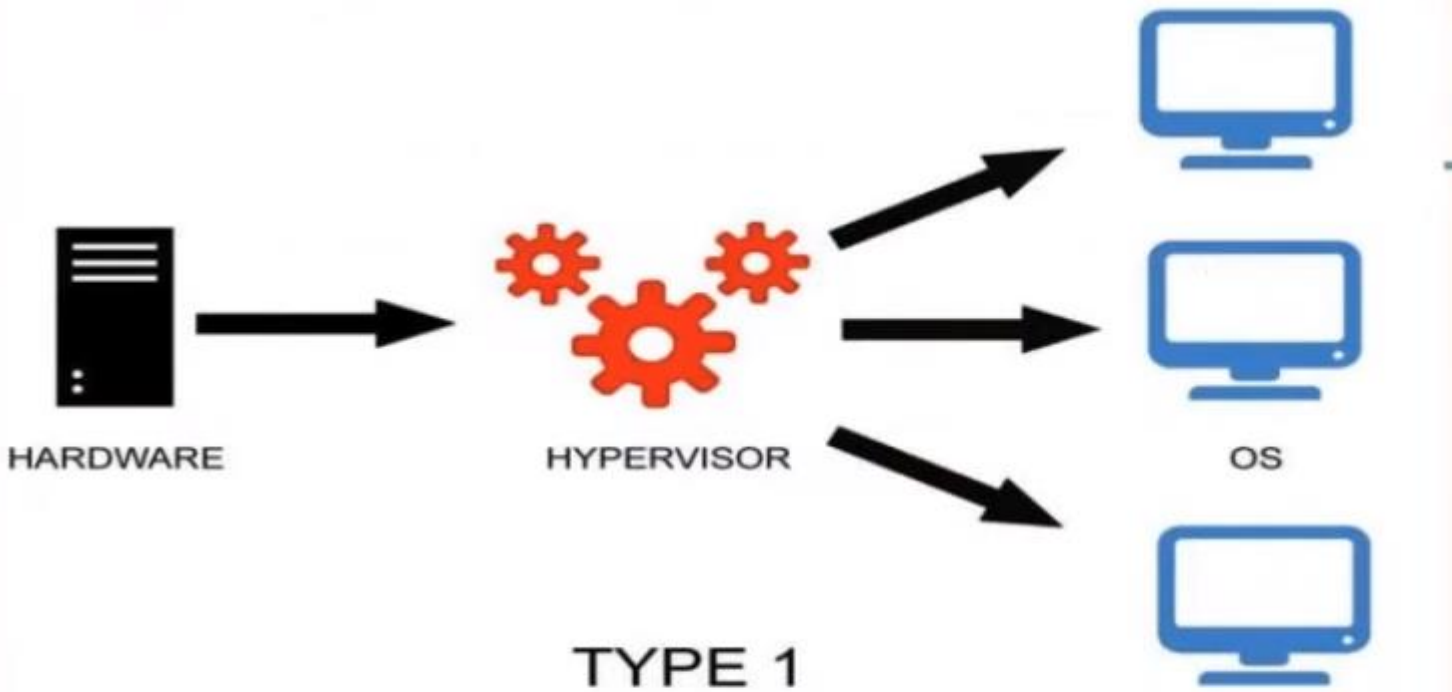
openstack™

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

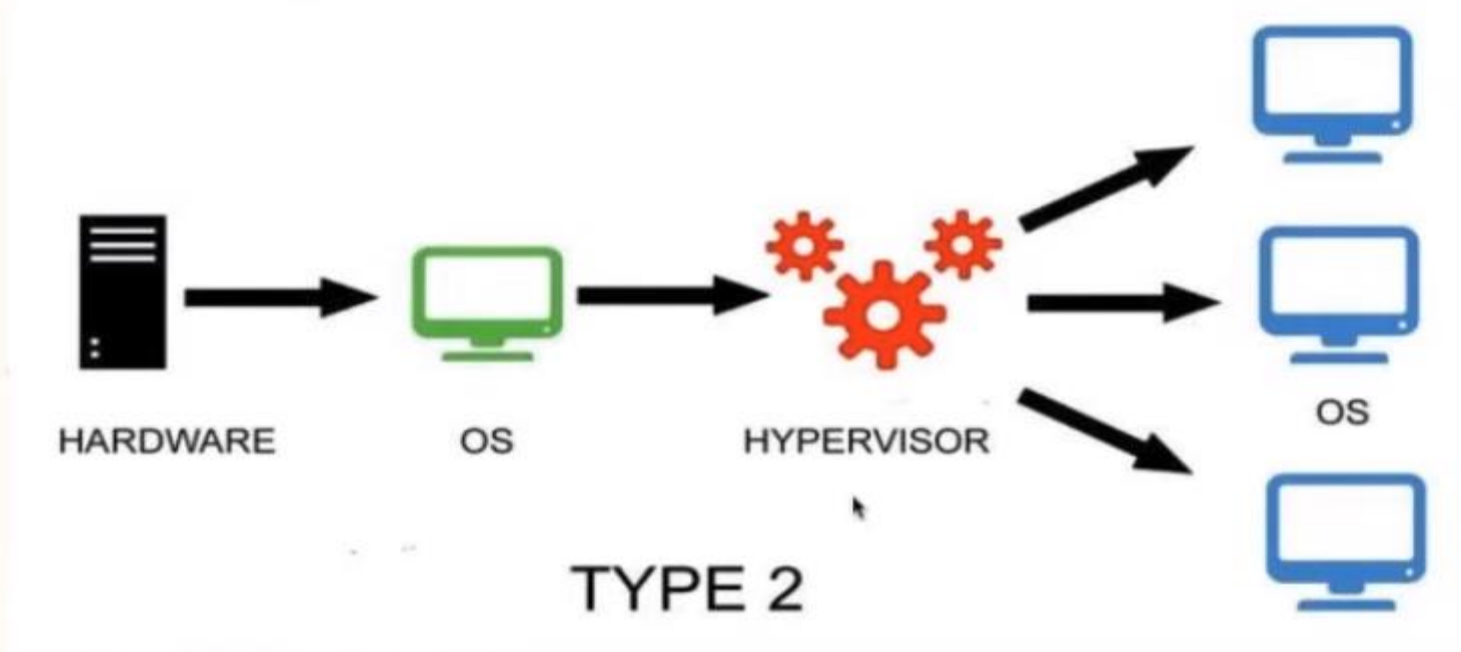


TYPE 1

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.

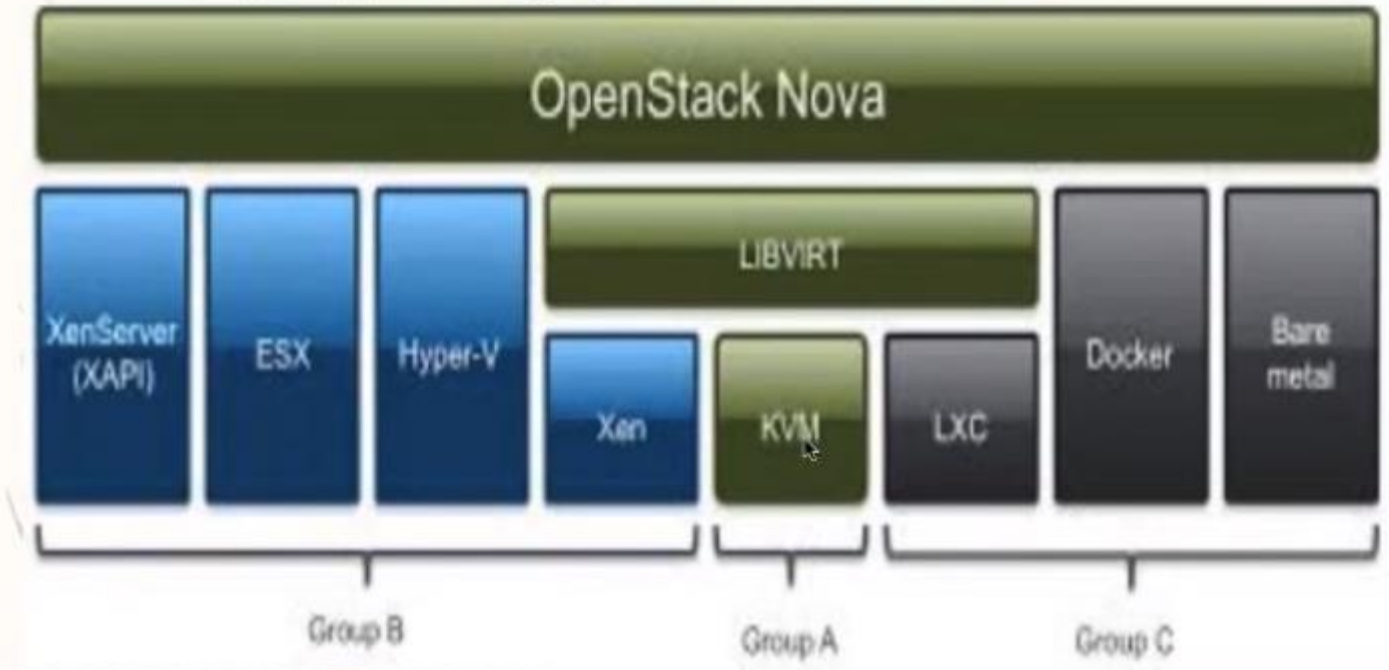
HyperVisor



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.

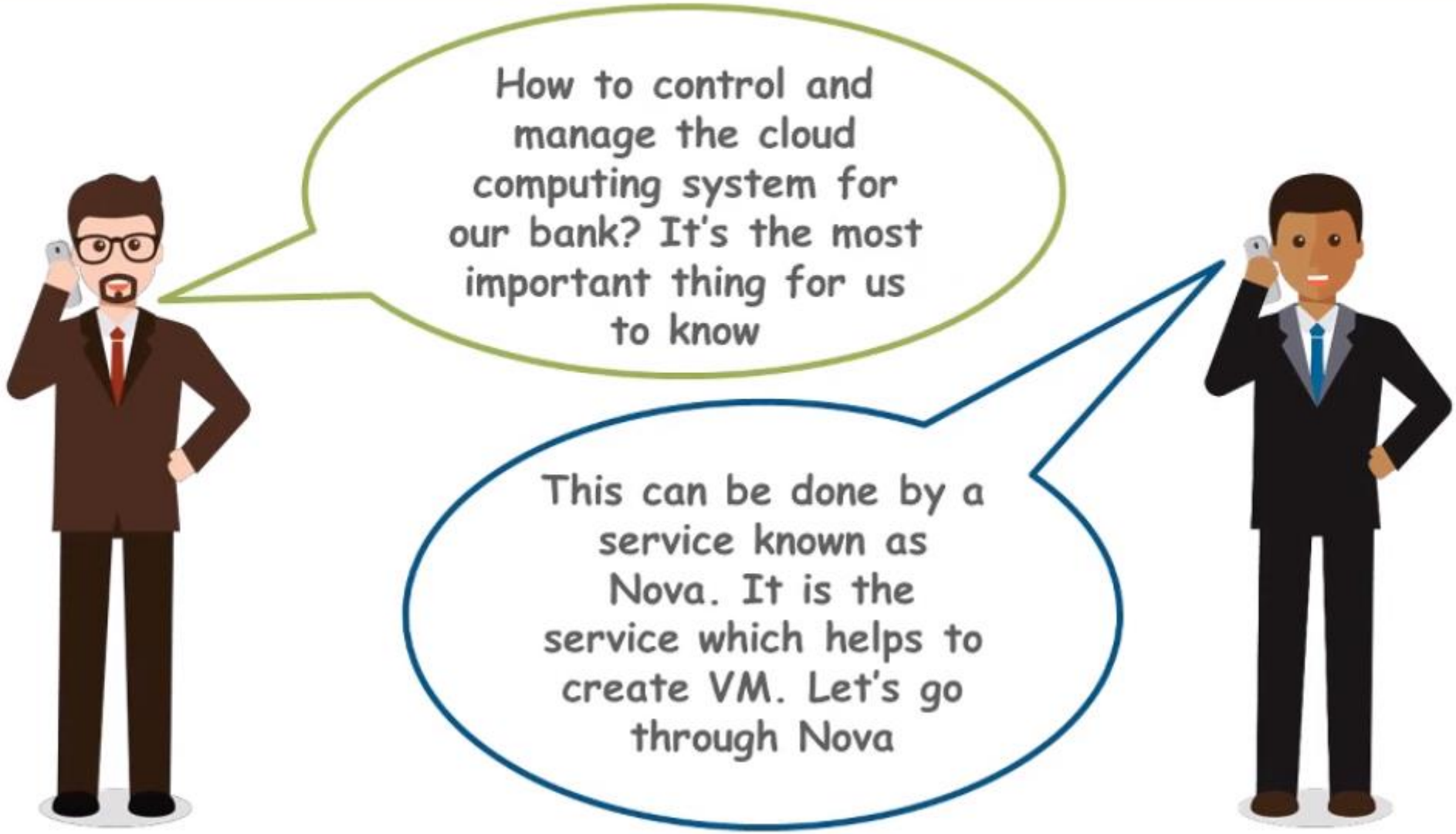
HyperVisor



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



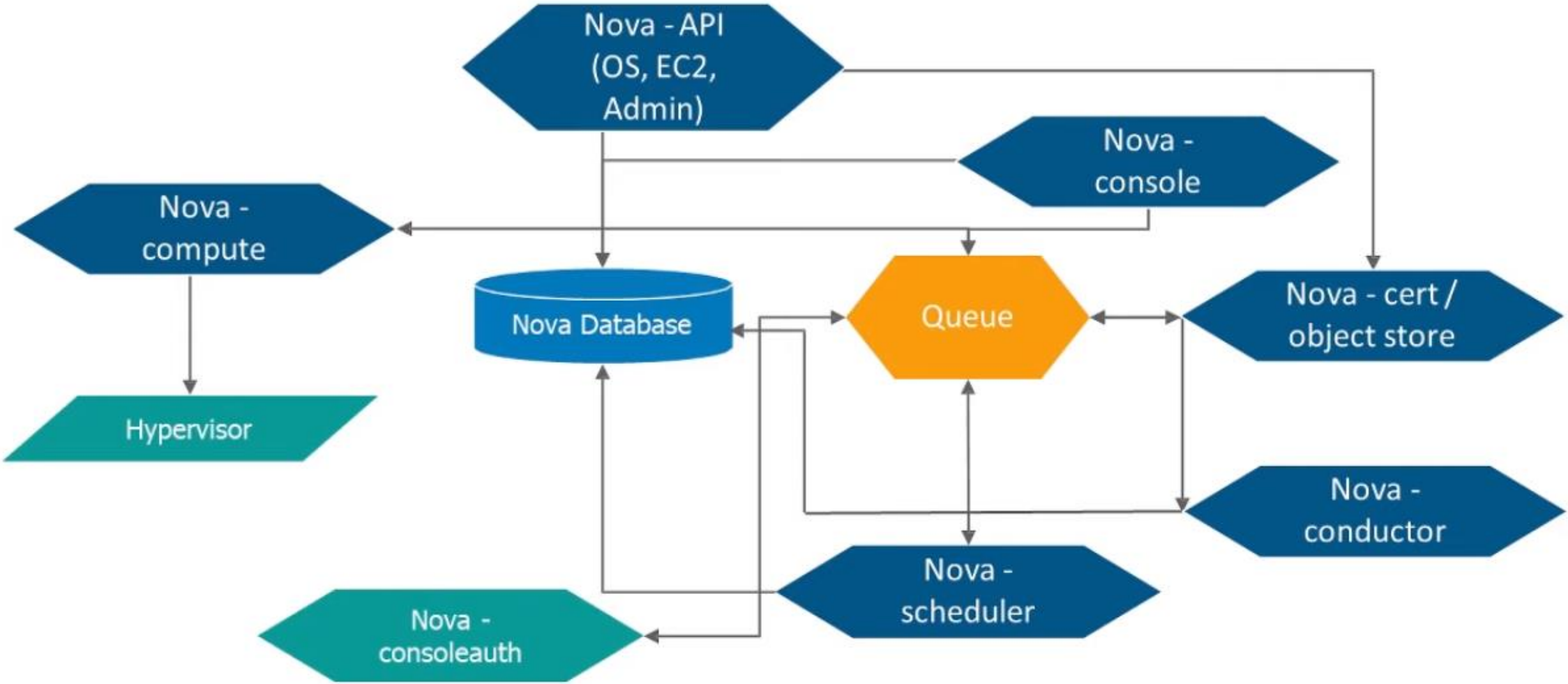
Introducing Nova

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

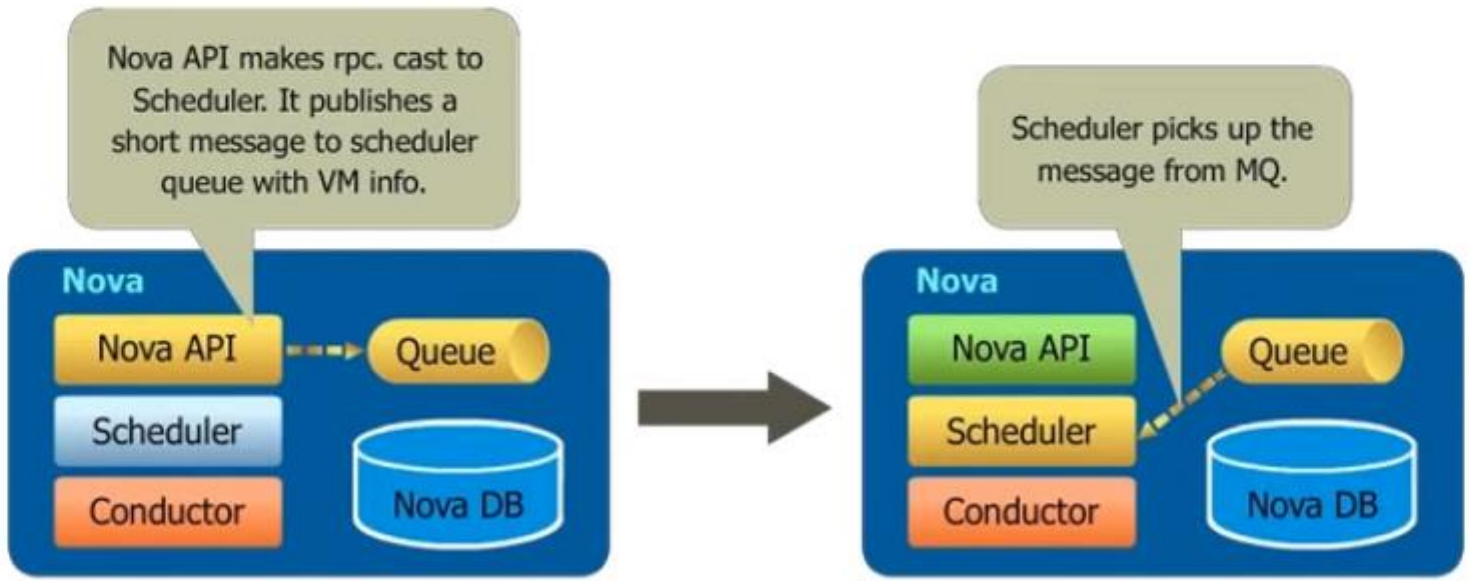
It is the work daemon which creates and terminates VMs via Hypervisor API

Nova-conductor

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

Nova-API – Description

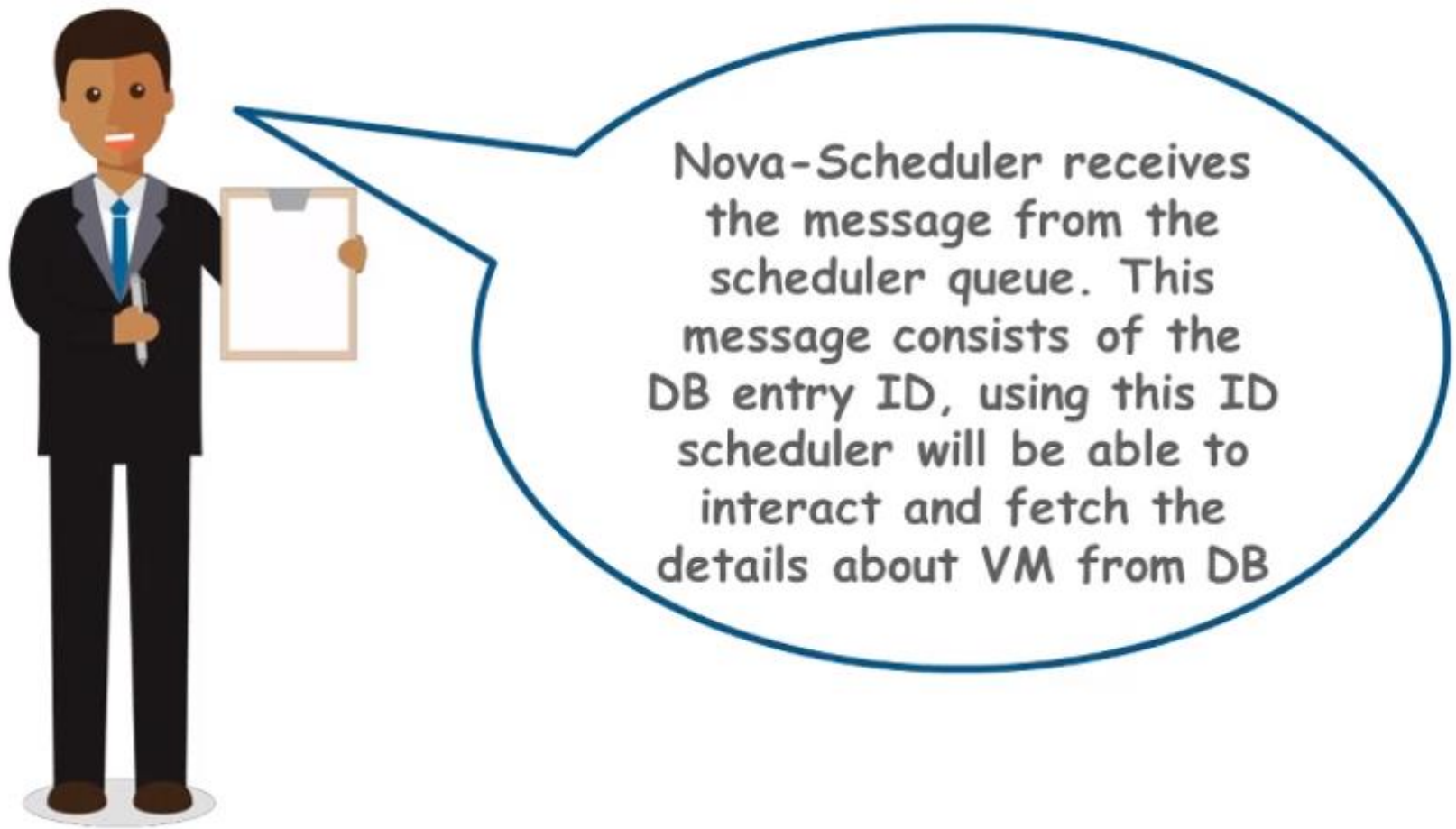
- Nova API accepts and responds to user's compute API calls
- It supports OpenStack compute API, Amazon EC2 API and Admin API, and initiates Orchestration Activities
- Once a call has been made to Nova-API, the call is sent to the queue if AUTH request is validated by Identity



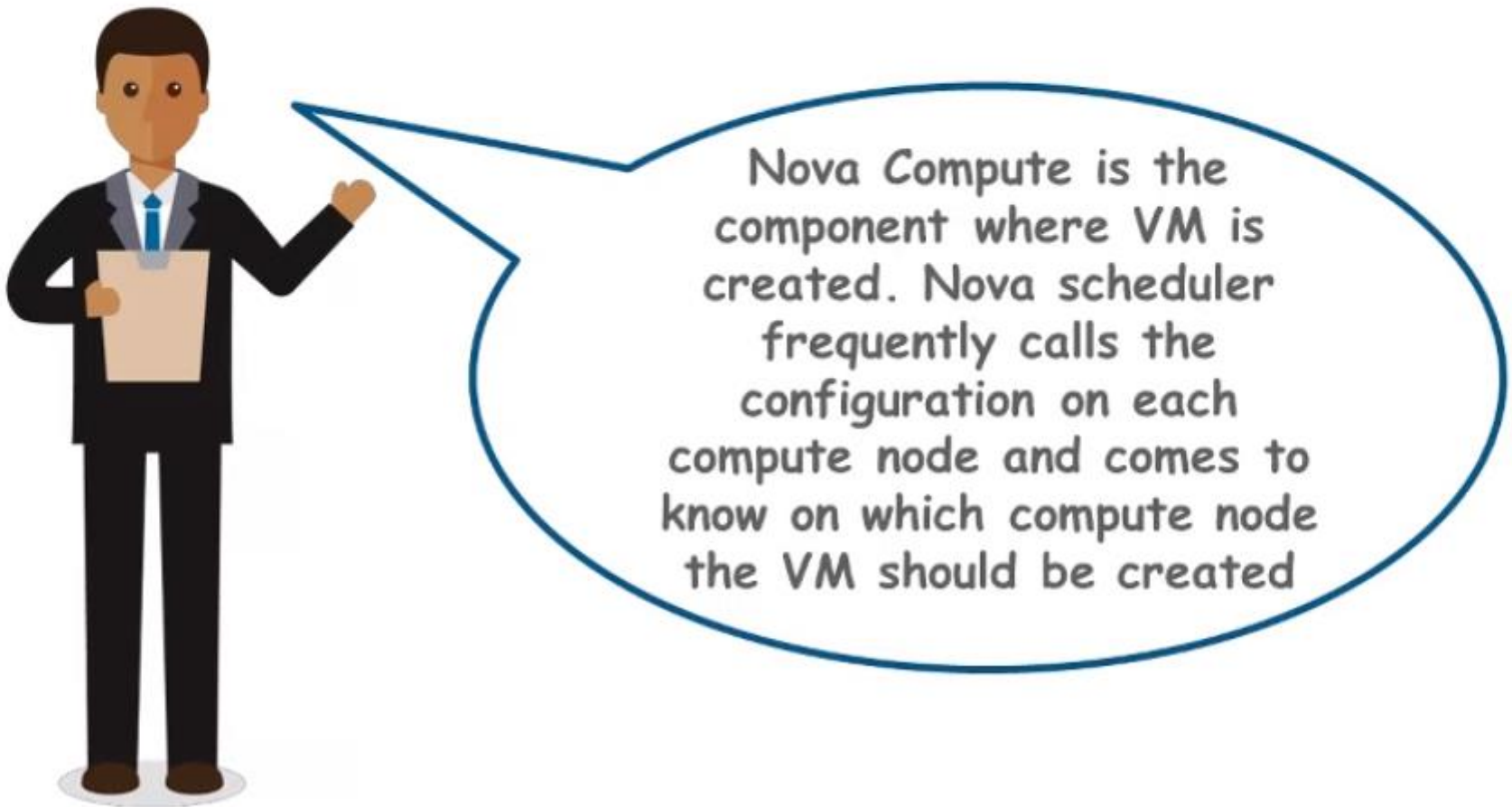
rpc.cast - don't wait for result (fire and forget)

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

Nova Scheduler

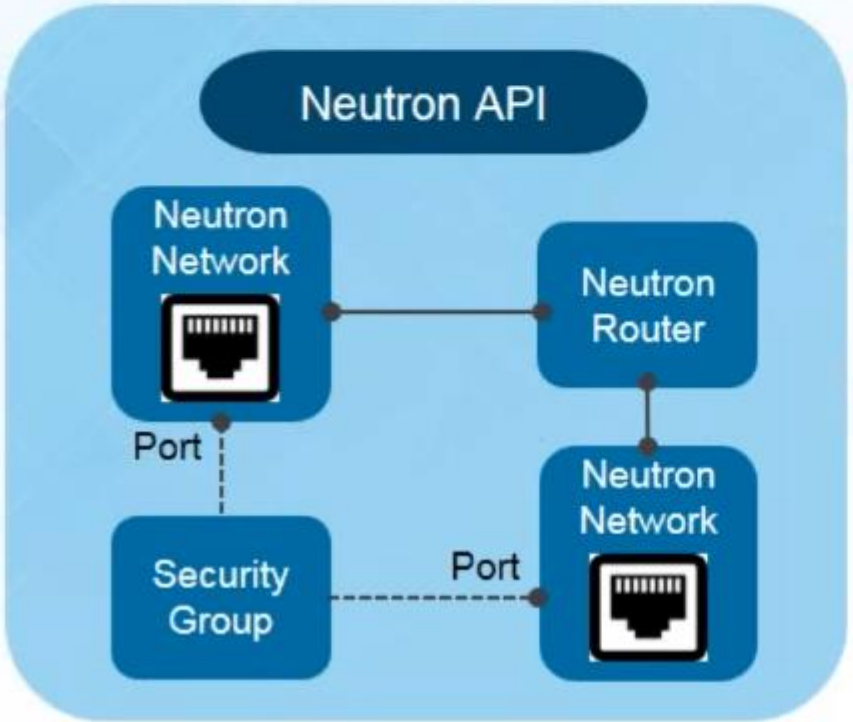


Nova Compute



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

Network as a Service (NaaS)

- Neutron offers real network services between devices managed by OpenStack, like Nova.

User-Created Networks

- Users can make their own networks and connect server devices to them. It's like setting up your own private internet connection.

Flexible Backend

- Neutron's design allows users to use vendor-supported equipment, making it adaptable to different technologies.

Additional Services

- Neutron can integrate extra network services as needed. It's like adding more features to your network setup, making it versatile and customizable.

NEUTRON- Core Network Features

- Neutron keeps growing with essential features for routers, virtual switches, and SDN networking controllers.
- Key Resources: Ports, Networks, Subnets
- **Ports:** Think of ports as virtual plug points where devices and network services connect. They include MAC and IP addresses for network interfaces.
- **Networks:** Neutron defines networks as isolated segments where devices communicate. These are like virtual switches implemented by various tools.
- **Subnets:** Subnets are blocks of IP addresses within a network. Devices' IP addresses come from these blocks, ensuring organized communication.

NEUTRON Core Network Features - Example

- Imagine you're starting a small business and you need a network to connect your computers, printers, and other devices. Neutron's core features come into play here:
- **Core Network Features:**
- **Routers:** In your office, you set up a router that connects your internal office network to the external world, allowing your employees to access the internet.
- **Virtual Switches:** Inside your office, you have multiple departments – sales, marketing, and operations. Neutron acts like virtual switches, creating isolated segments for each department. For instance, the sales team can communicate internally without directly interacting with the marketing team's network.
- **SDN Networking Controllers**: Let’s say your business expands, and you need to manage your network more efficiently. Neutron’s SDN networking controllers enable you to automate tasks, control traffic, and enhance security across your network seamlessly.

NEUTRON - Key Resources: Example

- **Ports:** Think of ports as the plug points on your office network switch. Each computer, printer, or any network device in your office has a port. These ports have unique MAC (hardware) addresses and IP addresses assigned by Neutron. When a new device is connected, Neutron ensures it gets the correct addresses, allowing it to communicate within the network.
- **Networks:** Neutron defines networks as separate, isolated areas within your office. Each department (sales, marketing, operations) represents a different network. Neutron ensures these networks are like virtual switches, enabling devices within the same department to communicate easily while keeping the networks separate from each other.
- **Subnets:** Within each department, you have different sections. Subnets are like blocks of phone numbers in these sections. Neutron ensures that devices within a section (subnet) share a common range of IP addresses. For example, all computers in the sales department have IP addresses from the same subnet, ensuring organized communication within the department.

NEUTRON - Additional resources as extensions.

- **Routers:** Routers act as bridges between different networks. They help data flow between these networks, ensuring communication.
- **Private IPs:** Neutron uses private IP addresses for two types of networks:
  a. **Tenant Networks:** These networks use private IP addresses visible only within the network. This allows devices within the network to communicate while staying isolated from other networks. These private IPs cannot be accessed from the Internet.
  b. **External Networks:**
     i. Visibility: External networks are visible and reachable from the Internet. They use specific address blocks that can be accessed globally.
- **Floating IPs:**
  a. **Definition:** Floating IPs are special IP addresses allocated on an external network.
  b. **Mapping:** Neutron connects these floating IPs to the private IP of a device (like a virtual machine).
  c. **Purpose:** Floating IPs allow instances to connect to external networks and access the Internet.
  d. **How:** Neutron achieves this connection using Network Address Translation (NAT), making sure data can flow between private and external networks.

NEUTRON - ADDITIONAL RESOURCES AS EXTENSIONS - EXAMPLE

- **Routers:**
- Example: Think of a router as a traffic cop at an intersection.
- Explanation: In our neighborhood, the router guides traffic between different streets. It helps data (cars in this analogy) move from one street to another, ensuring smooth communication between different parts of the neighborhood.
- **Private IPs:**
- Example: Imagine each house on a street has a private phone line.
- Explanation: Private IPs are like these phone lines. They allow communication between devices (houses) within the same street (network). Each house has its unique phone line (private IP), which is used for internal communication within the street (network).
- **Tenant Networks:**
- Example: Consider different neighborhoods, each with its unique set of houses and streets.
- Explanation: Tenant networks are like these neighborhoods. Houses (devices) within the same neighborhood (tenant network) can communicate using their private phone lines (private IPs). However, they are separate from houses in other neighborhoods, ensuring privacy and isolation.

NEUTRON - ADDITIONAL RESOURCES AS EXTENSIONS - EXAMPLE

- **External Networks:**
- Example: Think of a major highway connected to the neighborhood.
- Explanation: External networks are like this highway. They are visible and reachable from our neighborhood, allowing data (cars) to travel in and out. Unlike internal streets (tenant networks), the highway (external network) is public and accessible from different neighborhoods.
- **Floating IPs:**
- Example: Imagine having a temporary public phone number that can be redirected to a private phone line.
- Explanation: Floating IPs are like these temporary public phone numbers. They are allocated on the highway (external network). When someone calls the public number (floating IP), the call is redirected to a specific house's private phone line (private IP) within the neighborhood. This redirection is done by the router (traffic cop), allowing communication between the external network and a specific device in the tenant network.
- **In this analogy, routers manage traffic at intersections, private IPs are like individual phone lines within houses, tenant networks are separate neighborhoods, external networks are major highways accessible from neighborhoods, and floating IPs are temporary public phone numbers that redirect calls to specific private phone lines.**

NEUTRON ADVANCED SERVICES

- Neutron also provides advanced services to rule additional network OpenStack capabilities as follows:

Load Balancing as a Service (LBaaS):

- What: LBaaS distributes traffic across multiple computers.
- Example: Imagine a traffic cop directing cars to different lanes to avoid congestion.

Firewall as a Service (FWaaS):

- What: FWaaS secures network access at different levels.
- Example: Think of it like a security guard allowing only authorized personnel into a building.

Virtual Private Network as a Service (VPNaaS):

- What: VPNaaS creates secure tunnels between devices.
- Example: Picture a secret underground tunnel connecting two places, ensuring safe passage for important documents or people.

The Neutron architecture

Neutron Server:

- Role: Neutron server accepts requests from users.
- Analogy: Think of it as the receptionist at a hotel. Guests (requests) talk to the receptionist, who then directs them to the right rooms (actions).

Neutron Plugins:

- Role: Plugins do the actual work behind the scenes.
- Analogy: They are like the behind-the-scenes staff - cleaning rooms, setting up services, and ensuring everything works smoothly in the hotel.

Neutron Agents:

- Role: Agents run on computers and make changes based on requests.
- Analogy: Imagine them as maintenance workers. When the receptionist (server) tells them what needs to be done, they go to different rooms (nodes) and implement changes, ensuring everything is in order.

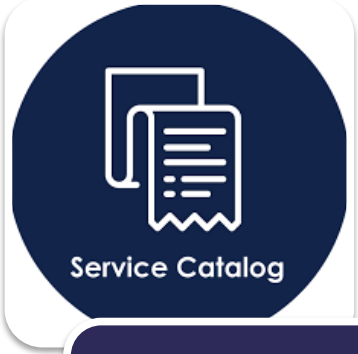
Neutron's Purpose:

- Role: Manages network connections for OpenStack instances.
- Analogy: Neutron is like a traffic manager on the roads, making sure there are no traffic jams and everyone can reach their destination smoothly. It ensures that the network doesn't slow down the whole system, giving users control over their network settings.


GATHERING THE
PIECES AND
BUILDING A PICTURE

How OpenStack Works

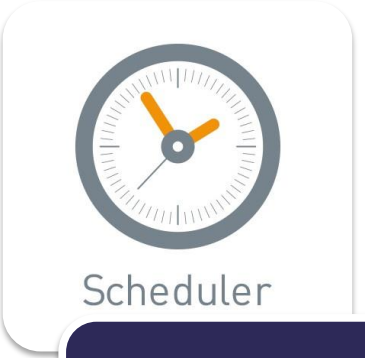
Authentication



The service catalog is then provided by Keystone

The service catalog exposes the resources available on a token request

Once authenticated, you can talk to an API node



Instance Scheduler

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 (Keystone):**
- Users log in with their username and password.
- Keystone authenticates the users, ensuring they are who they claim to be.
2. **Service Catalog (Keystone):**
- Keystone provides a catalog listing all available OpenStack services and their access points (API endpoints).
3. **Users can check the catalog using OpenStack CLI commands.**

    $ openstack catalog list
- The service catalog is a JSON structure that exposes the resources available on a token request
- This catalog acts as a menu, showing what services are available to the users.
4. **Typically, once authenticated, you can talk to an API node.**
- There are different APIs in the OpenStack ecosystem (theOpenStack API and EC2 API):

How OpenStack Works

Instance Scheduling:

- OpenStack services use schedulers (like nova-scheduler or Neutron L3 scheduler) to decide where to launch virtual machines or routers.
- Schedulers analyze the state of available resources on physical nodes. They choose the best node to run a virtual instance based on resource availability.
- If you want to launch a virtual machine, nova-scheduler helps pick the right compute node for it, ensuring efficient resource utilization.

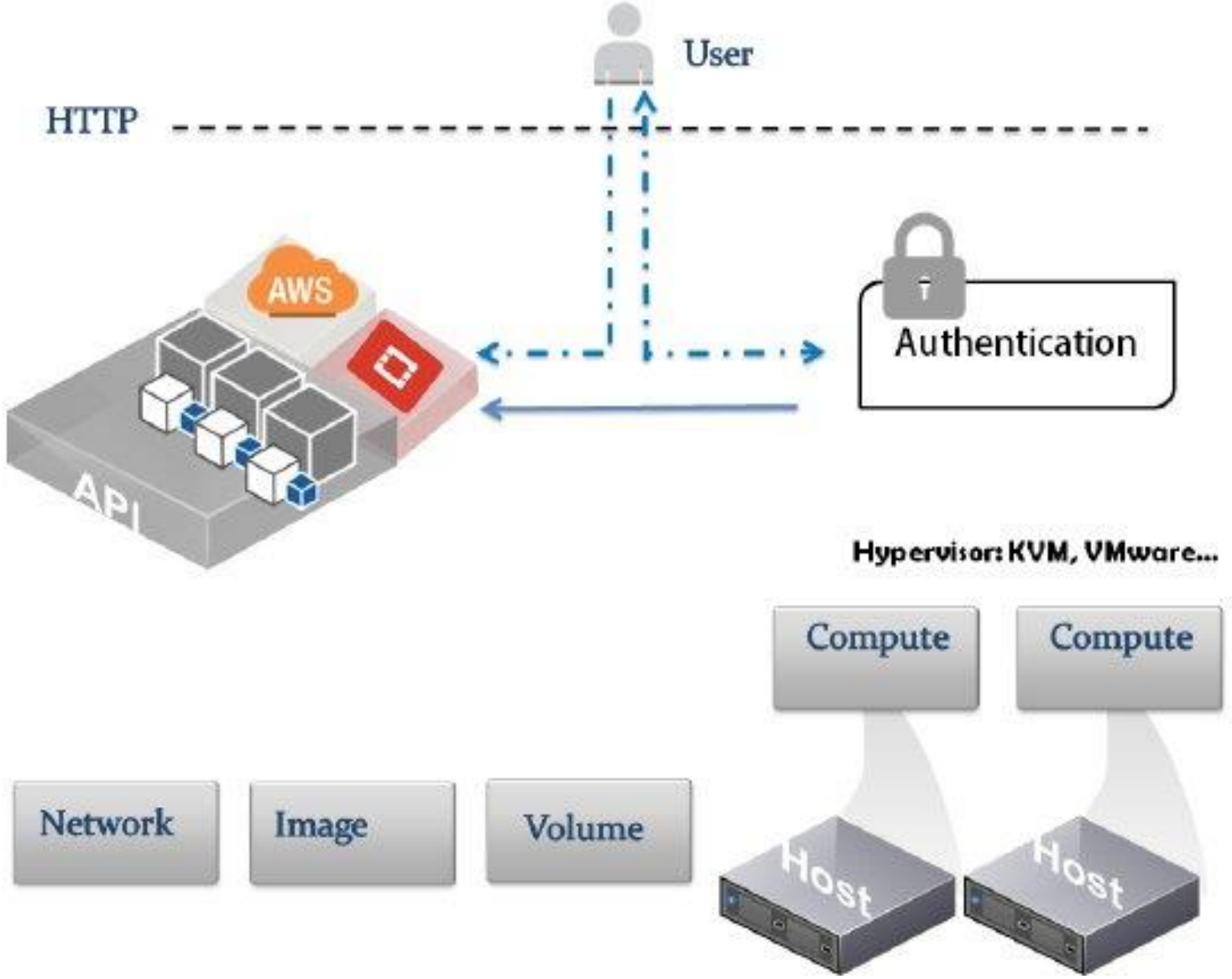
Scheduler's Role:

- The scheduler in OpenStack checks available resources on physical machines.
- It chooses the best machine to run a virtual instance (like a virtual machine or router).

Example:

- Nova-scheduler: Picks a machine for virtual machines.
- Neutron L3 scheduler: Chooses a network node for virtual routers.

High-level view of how OpenStack works:



Provisioning a VM under the hood – The process of launching a virtual machine in OpenStack:

- 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.**

OpenStack services interact with each other via a **message bus** to submit and retrieve **RPC calls.**

The information of each step of the provisioning process is **verified and passed** by different OpenStack **services** via the **message bus**.

From an architecture perspective, **sub system calls are defined and treated** in OpenStack **API endpoints** involving: Nova, Glance, Cinder, and Neutron.

On the other hand, the inter-communication of APIs within OpenStack requires an **authentication** mechanism to be trusted, which involves **Keystone**.

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 workflow based on API calls in OpenStack:

1. Authentication: Users authenticate their identity using the identity service (Keystone).

2. A token is generated for further interactions.

3. Base Image: The image service (Glance) is contacted. It lists and provides the base image needed for the virtual machine.

4. Compute Request: A request is made to the compute service (Nova) API.

5. Details about security groups and keys are processed.

6. Network Configuration: The network service (Neutron) API is called. Available networks for the virtual machine are determined.

Provisioning workflow based on API calls in OpenStack:

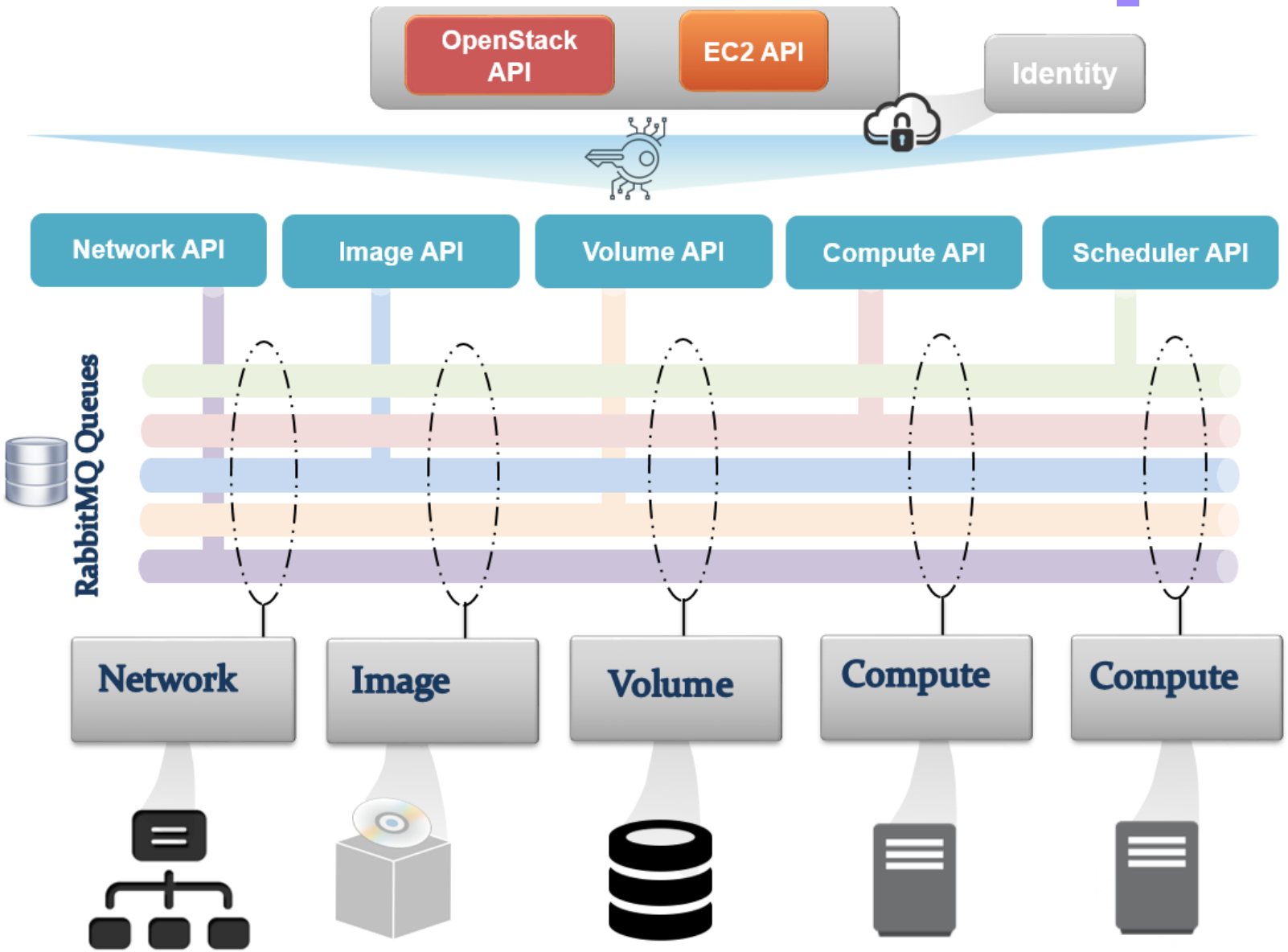
7. Scheduler Selection: The compute scheduler service chooses the appropriate hypervisor node.

8. Storage Allocation: The block storage service (Cinder) API is called. Storage volumes are allocated for the instance.

9. Instance Creation: The instance is spun up on the chosen hypervisor node via the compute service API call.

10. Network Resource Allocation: The network service API is called again to allocate necessary network resources for 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):** A UUID (Universally Unique Identifier) token is a long, unique string of characters used for authentication. Example: Like a special passcode that you get when you log in. For instance: ‘a1b2c3d4-e5f6-4g7h-8i9j-0k1l2m3n4o5p’
  - **Public Key Infrastructure (PKI):** PKI tokens use a public-private key pair for authentication, ensuring secure communication. Example: It's like having a special lock and key. The server has a unique key (private key), and the user has a corresponding unique key (public key) that fits the lock.

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**
- 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. Imagine it as a secret recipe that only the chef (server) and the person ordering the dish (user) know. It looks like this: gAAAAABgkXwUKyX2-0PQ7X23yj7gtJY_CnFz74lR6sTQDd9rW47v...

Fernet Token Example

- Example

```
80 5D 6B 3D 9B F2 38 E0 07 E4 E3 00 F0 A1 A2 08
F5 E9 A8 4C 4A 01 92 25 0F FD C6 5F 2B 53 0A 8D
CA C2 F1 49 E1 F8 D1 36 91 3E AA C5 2A 09 7A 16
A4 BC F2 2E 9F 53 F7 B0 2E D5 62 5B DC 7B D8 F1
```

- Version: 80
- Timestamp: 5D6B3D9BF238E007 (Unix timestamp: 1566982247)
- IV: E4E300F0A2A208F5E9A84C4A0192250F
- Cipher Text: FDC65F2B530A8DCA...
- HMAC: 49E1F8D136913EAA...

Note:

- IV: Initialization Vector
- HMAC: Hash-based Message Authentication Code

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**.

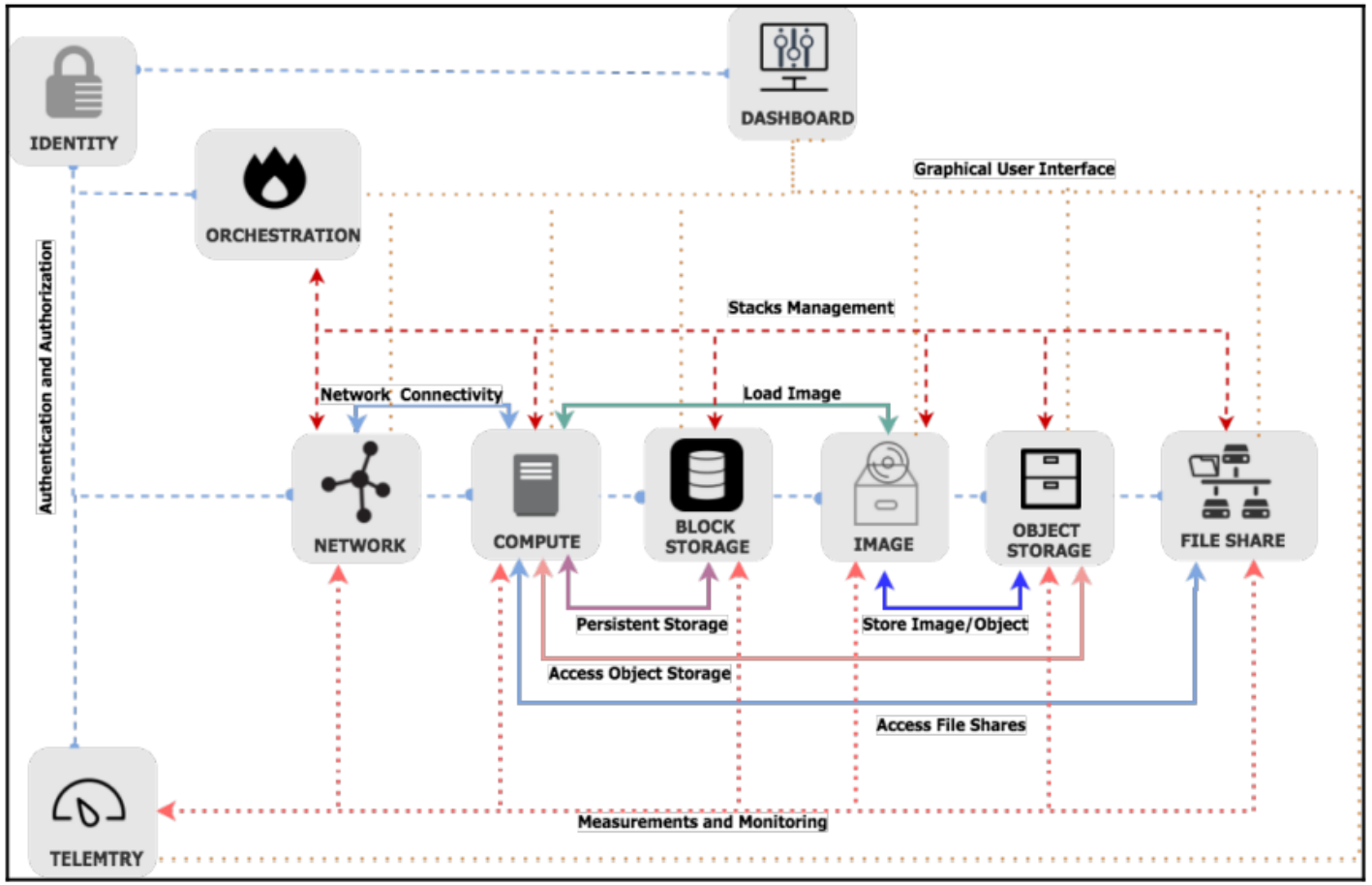
High-Level Overview

Detailed Planning

Deployment on Hardware

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.

Logical Model Design

Compute Nodes

- Physical servers with virtualization capabilities (e.g., KVM, Xen) where virtual machines run.

Storage Nodes:

- Servers with storage capacity providing block and object storage (e.g., Ceph, Swift).

Network Nodes:

- Servers managing networking components, ensuring connectivity (e.g., Neutron).

Control Nodes:

- Servers housing control services like Keystone (identity), Glance (image), Nova (compute), and Horizon (web interface).

Logical Model Design

Networking:

- Private Networks: Internal networks for communication between VMs.
- Public Networks: External networks for internet connectivity and public-facing services.

Security:

- Firewalls and Security Groups: Protect VMs from unauthorized access.
- SSL Certificates: Encrypt data in transit.

Scaling:

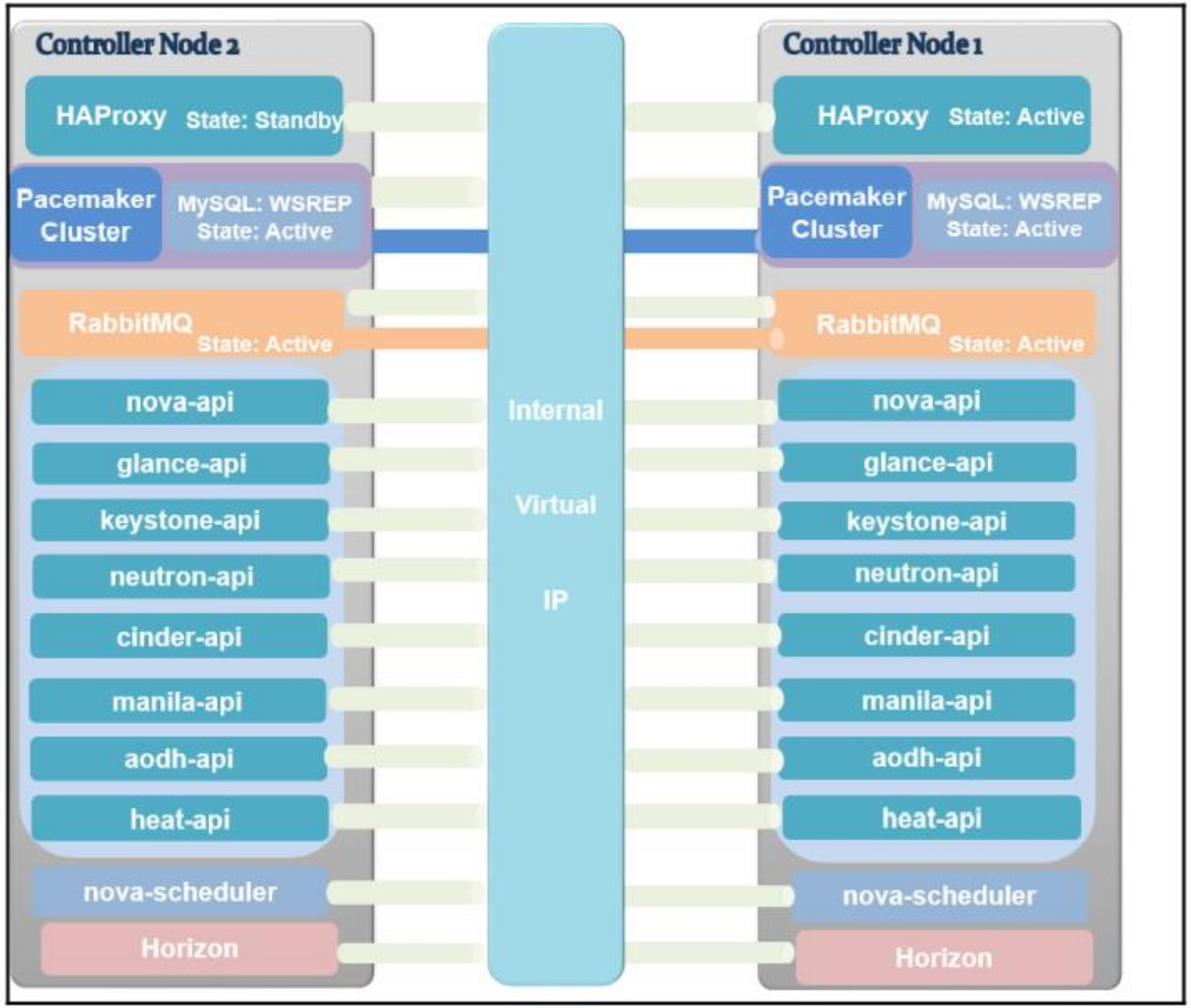
- Load Balancers: Distribute traffic across multiple compute nodes.
- Auto-Scaling: Automatically add or remove resources based on demand.

Storage:

- Block Storage: Persistent storage volumes attached to VMs.
- Object Storage: Scalable, redundant storage for objects and files.

Nodes in OpenStack

- In OpenStack, it is a more general term used to describe any physical or virtual machine that is part of the OpenStack deployment. Nodes in OpenStack can serve various purposes.
- Different types of nodes in OpenStack:
- **Compute Node (Instance Node):** Compute nodes in OpenStack are responsible for running virtual machine instances. When you launch an instance, it runs on a compute node. These nodes handle tasks such as running virtual machines, managing their life cycle, and providing access to them via the network.
- **Controller Node:** Controller nodes manage the overall OpenStack environment. They host services like the identity service (Keystone), image service (Glance), networking service (Neutron), and orchestration service (Heat). The controller node manages the core functionality and provides the API endpoints for users and other services.
- **Storage Node:** Storage nodes are responsible for storing data in an OpenStack deployment. They can use various storage technologies, including block storage (Cinder), object storage (Swift), or shared file systems (Manila). Storage nodes ensure data persistence and availability for instances and other services.
- **Network Node:** Network nodes handle networking services in OpenStack. They manage the network traffic, implement network policies, and ensure connectivity between instances and external networks. Network nodes work closely with the networking service (Neutron) to provide networking functionality.


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

- In the Logical Model Design of OpenStack, focus on:

- **High Availability**: Plan for reliability and scalability using tools like VIP, HAProxy, and Pacemaker to ensure your OpenStack setup can handle heavy traffic and remain operational even if one component fails.

- **Storage:** Consider your data needs. Ask questions like: How much data do you need to store? Will you run data-heavy applications in the future? What storage requirements do you have for backups and virtual machine snapshots? Determine if ephemeral storage (temporary storage for instances) is necessary.

Logical Model Design

Network

- In the logical model design of OpenStack, it's important to consider network features.
- OpenStack has evolved to offer flexible networking options. Instead of making random decisions, explore available network modes.
- Keep filtering until you find the right network topology for your needs.
- OpenStack's complexity ensures flexibility, allowing you to tailor the network to your requirements.

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 Design (Deployment on Hardware):

- **Server Hardware:**

Choose physical servers with suitable CPU, memory, and storage capacity.

- **Network Infrastructure:**

Implement physical switches, routers, and firewalls ensuring proper network connectivity.

- **Storage Infrastructure:**

Deploy storage systems (e.g., SAN/NAS) for backend storage solutions.

- **High Availability:**

Implement redundancy for critical components to ensure minimal downtime.

- **Data Center Layout:**

Organize physical servers, networking equipment, and storage systems within the data center for efficient cooling and maintenance.

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

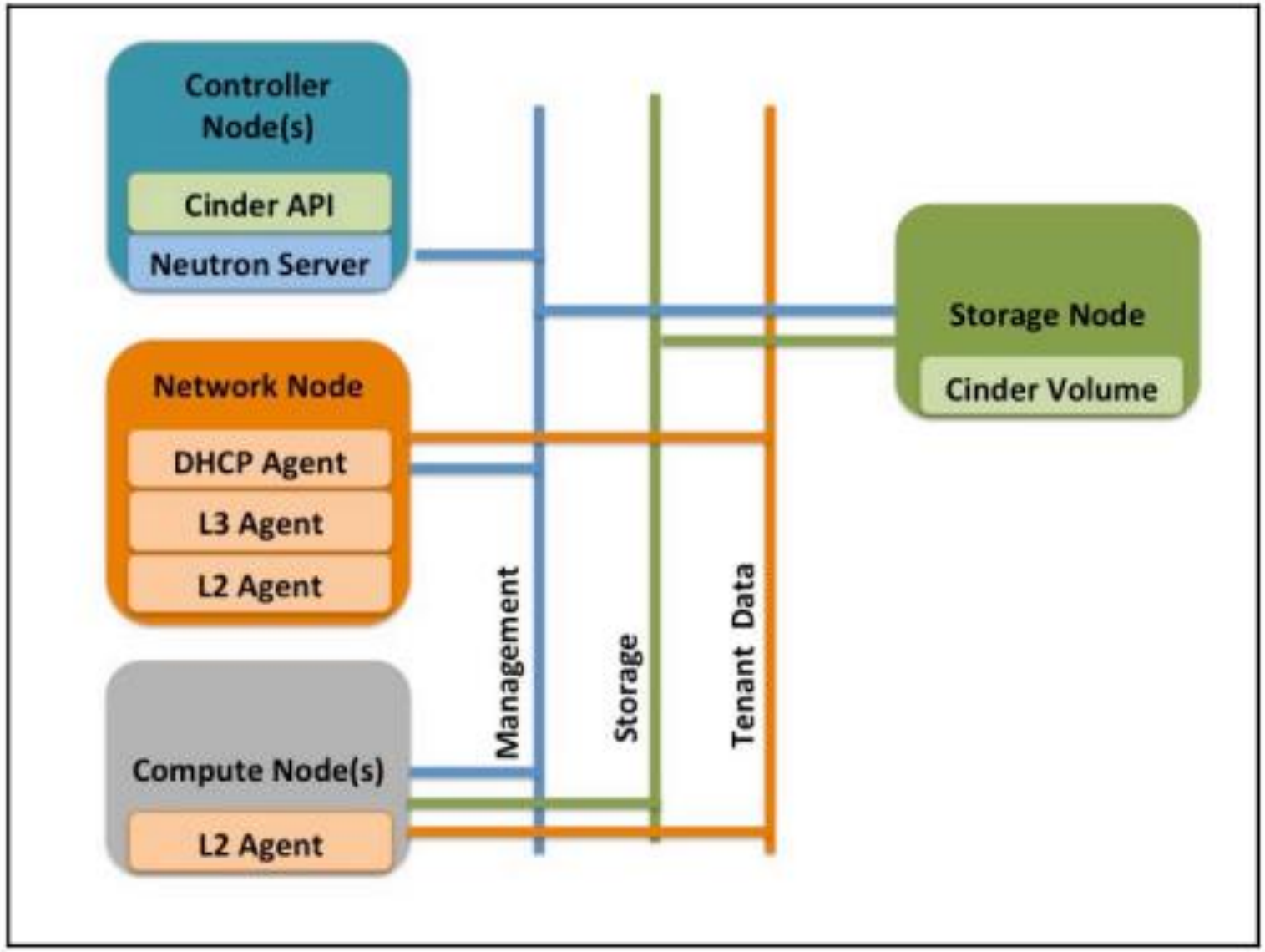
The Storage network

- 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 next step is to validate our network design in a simple diagram:



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

The physical model design

- Estimating the hardware capabilities
- Since the architecture is being designed to scale horizontally, we can add more servers to the setup. We will start by using commodity class, cost-effective hardware.
- In order to expect our infrastructure economy, it would be great to make some basic hardware calculations for the first estimation of our exact requirements.
- Considering the possibility of experiencing contentions for resources such as CPU, RAM, network, and disk, you cannot wait for a particular physical component to fail before you take corrective action, which might be more complicated.
- CPU calculations
- Memory calculations
- Network calculations
- Storage 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)

Previous Year University Question Paper 2022

Part A

- Summarize on Nova services in OpenStack. (3)
- Differentiate between private cloud and public cloud.(3)

Part B

- List and explain the different components in OpenStack Logical Architecture. (6)
- Illustrate the provisioning of VM in OpenStack using diagram. (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