



UNIT-II

Managing Cloud and SLA



Cloud data management

- Cloud data management is the practice of storing a company's data on an **offsite server** that is typically owned and overseen by a vendor who specializes in cloud data hosting.
- Managing data in the cloud provides an automated **backup strategy**, **professional support**, and **ease of access** from any location.



benefits of cloud data management

Security:

- Modern cloud data management is often more secure than on-premises solutions.
- In fact, 94% of cloud adopters report security improvements.
- First, because cloud data management reduces the risk of data loss due to device damage or hardware failure.
- Second, because companies specializing in cloud hosting and data management employ more advanced security measures and practices than companies that invest in their on-premises data.



Scalability and savings:

- Cloud data management lets users scale services up or down as needed. More storage or compute power can be added when needed.
- Companies can then scale back after the completion of a big project, to avoid paying for services they don't need.

Governed access:

- With improved security comes greater peace of mind regarding governed data access.
- Cloud storage means team members can access the data they need from wherever they are. This access also supports a collaborative work culture, as employees can work together on a data set, easily share insights, and more.



Automated backups and disaster recovery:

- The cloud storage vendor can manage and automate data backups so that the company can focus its attention on other things, and can rest assured that its data is safe.
- Having an up-to-date backup at all times also speeds up the process of disaster recovery after emergencies.

Improved data quality:

- An integrated, well-governed cloud data management solution helps to create a **single source of truth** for every data point. Data remains **clean, consistent, and up-to-date**.



Automated updates:

- Cloud data management providers are committed to providing the best services and capabilities.
- When applications need updated, cloud providers run these updates automatically. That means your team doesn't need to pause work while they wait for it to update everyone's system.

Sustainability:

- For companies and brands committed to decreasing their environmental impact, cloud data management is a key step in the process.



Best practices for a cloud data management strategy

Start with a plan.

- Dumping data into the cloud is not “cloud data management.” Will you move all data to the cloud? Or create a hybrid environment?
- Who needs access to what data? Where should different processing tasks take place?

Maintain clean data.

- This is incredibly important, as other data management practices depend upon it.
- Keeping the data “clean” means ensuring that the data entry is **accurate** and that there are **no duplicates** or other **errors**.



Backup the data (often).

- Most cloud providers will automatically run regular backups.
- If a company is hosting its own cloud, however, make sure the IT department is running regular backups.

Don't forget about data governance.

- An existing data governance policy for on-premises data can be updated for a hybrid or cloud data management architecture.
- Moving data to the cloud, however, often means extra compliance issues need to be considered, so make sure those don't slip through the cracks.



real-world examples of successful cloud data management

Cloud data management for healthcare

- **Accolade** is a healthcare information services provider that aims to improve the **experience**, **outcomes** and **cost** of healthcare for providers and patients.
- Accolade uses a cloud data management system to merge data from diverse medical records — both from the patient and the patient's family members — in order to compile the most thorough possible portrait of the patient's medical history and potential risk factors.
- This allows the doctors and hospitals using Accolade's data services to access the most accurate and complete information available to them when making important decisions regarding their patients' health and well-being.



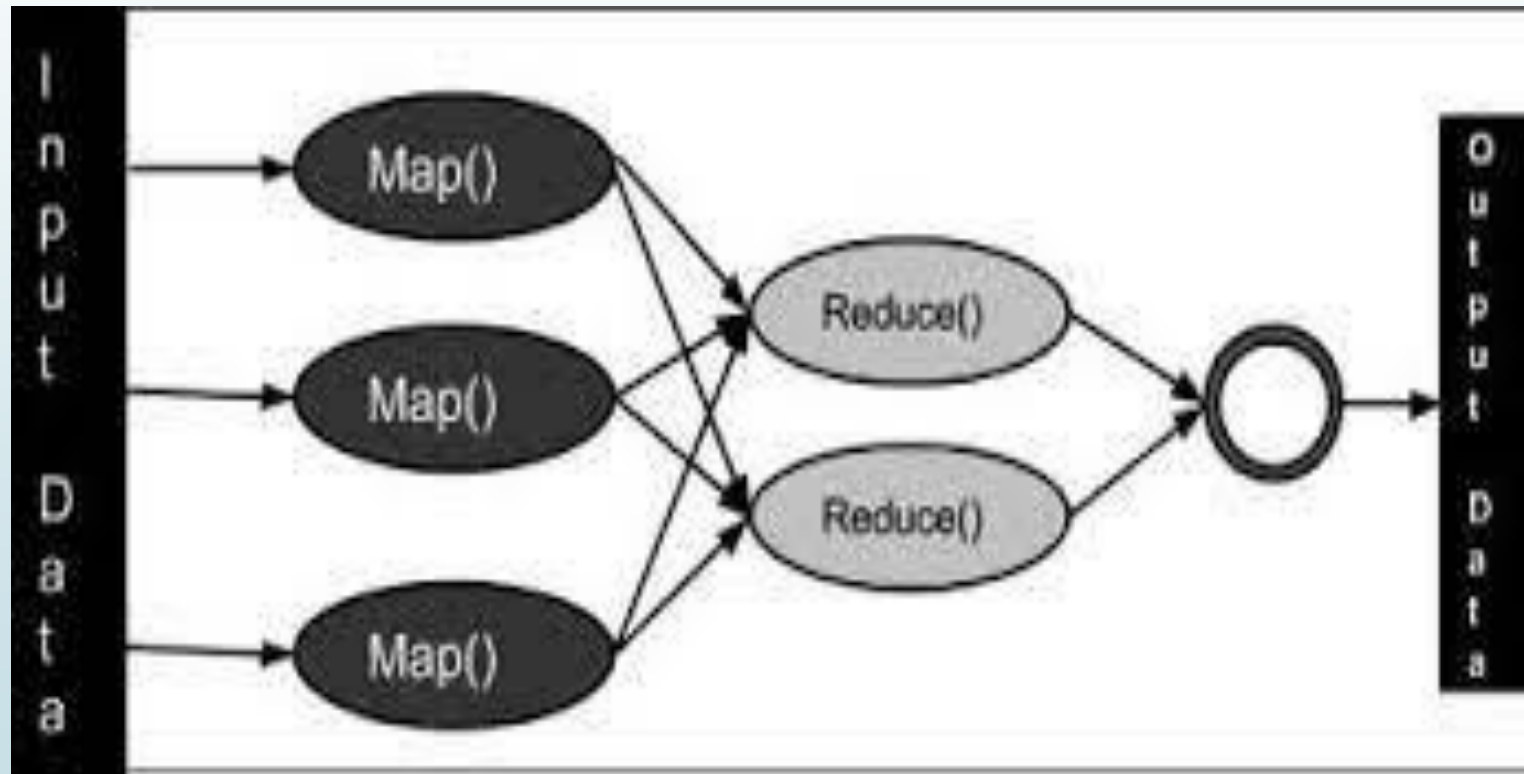
Cloud data management for finance

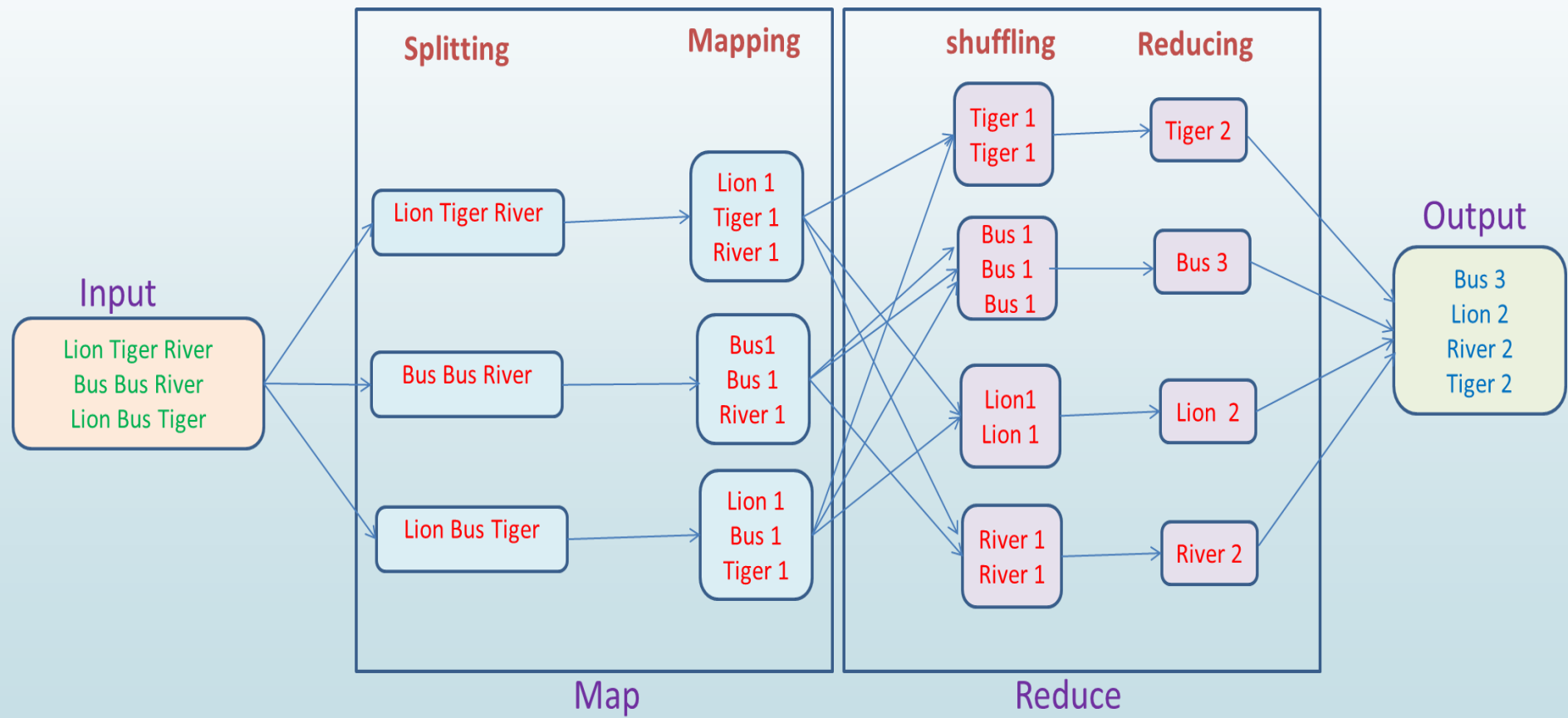
- Australian wealth-management company Class needs to be able to generate frequent, detailed.
- If the data is difficult to access, or only certain people have access to it, the reporting process bottlenecks.
- Class uses cloud data management to improve governed, self-service access to data and dramatically reduce the average amount of time required to generate reports.



Introduction to MapReduce

- **MapReduce** is a software framework and programming model used for processing huge amounts of data.
- **MapReduce** program work in two phases, namely, **Map** and **Reduce**.
- Map tasks deal with **splitting** and **mapping** of data while Reduce tasks **shuffle** and **reduce** the data.
- The programs of Map Reduce in cloud computing are **parallel** in nature, thus are very useful for performing **large-scale data analysis** using **multiple machines** in the **cluster**.







Input Splits:

- An input to a MapReduce in Big Data job is divided into fixed-size pieces called **input splits**. Input split is a chunk of the input that is consumed by a single map.

Mapping

- This is the very first phase in the execution of map-reduce program. In this phase, data in each split is passed to a mapping function to produce output values.
- For example, a job of mapping phase is to count a number of occurrences of each word from input splits.



Shuffling

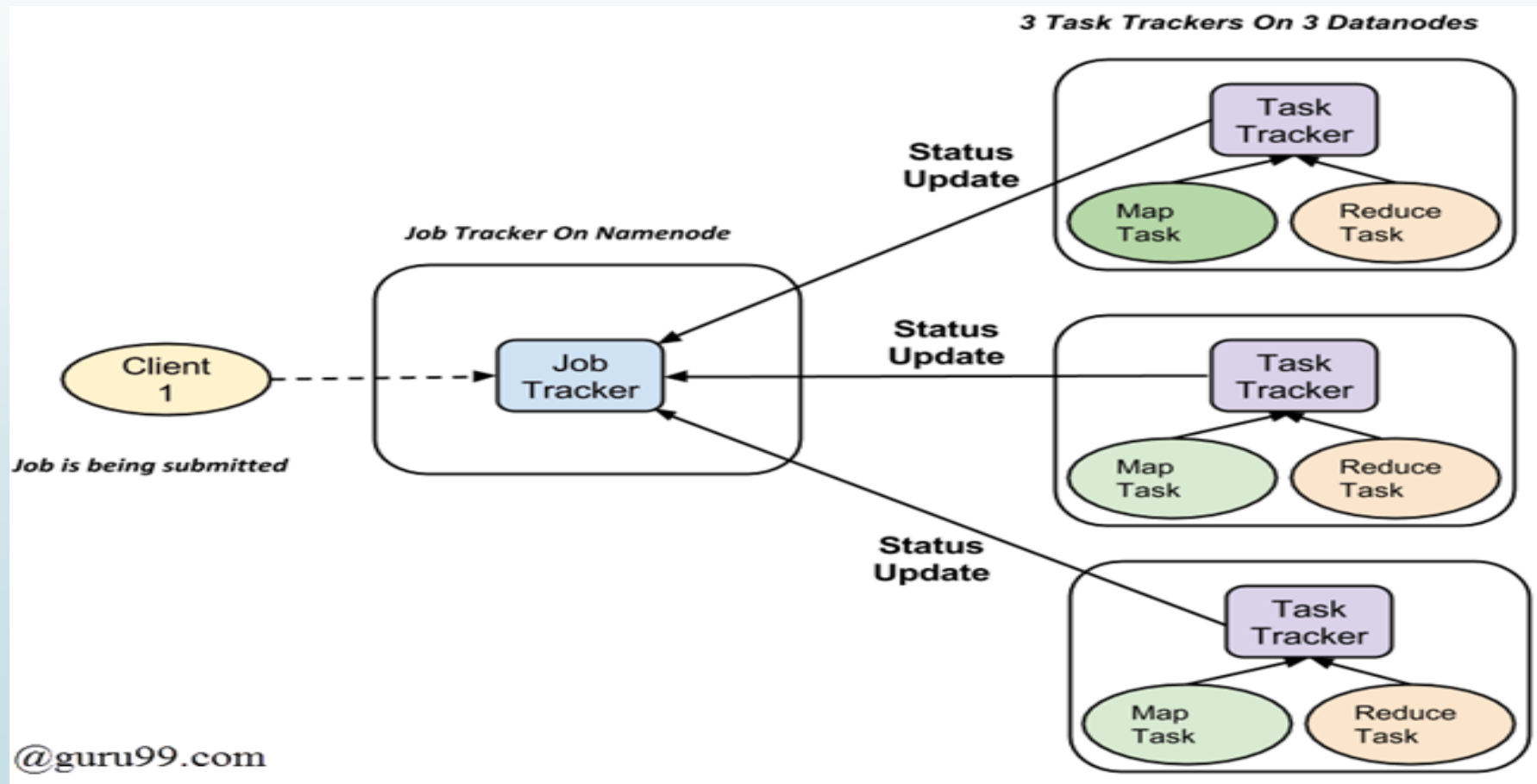
- This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubbed together along with their respective frequency.


Reducing

- In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase **summarizes** the complete dataset.

How MapReduce Organizes Work?

- There are two types of tasks:
 1. **Map tasks** (Splits & Mapping)
 2. **Reduce tasks** (Shuffling, Reducing)
- as mentioned above.
- The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called a
- **Jobtracker**: Acts like a **master** (responsible for complete execution of submitted job)
- **(Multiple) Task Trackers**: Acts like **slaves**, each of them performing the job
- For every job submitted for execution in the system, there is one **Jobtracker** that resides on **Namenode** and there are **multiple tasktrackers** which reside on **Datanode**.



- 
- A job is divided into multiple tasks which are then run onto multiple **data nodes** in a cluster.
 - It is the responsibility of job tracker to **coordinate** the activity by **scheduling** tasks to run on different data nodes.
 - Execution of individual task is then to look after by **task tracker**, which resides on every data node executing part of the job.
 - Task tracker's responsibility is to send the **progress report** to the job tracker.
 - In addition, task tracker periodically sends '**heartbeat**' signal to the Jobtracker so as to notify him of the **current state** of the system.
 - Thus job tracker keeps track of the overall progress of each job. In the event of task failure, the job tracker can reschedule it on a different task tracker.



Introduction to OpenStack

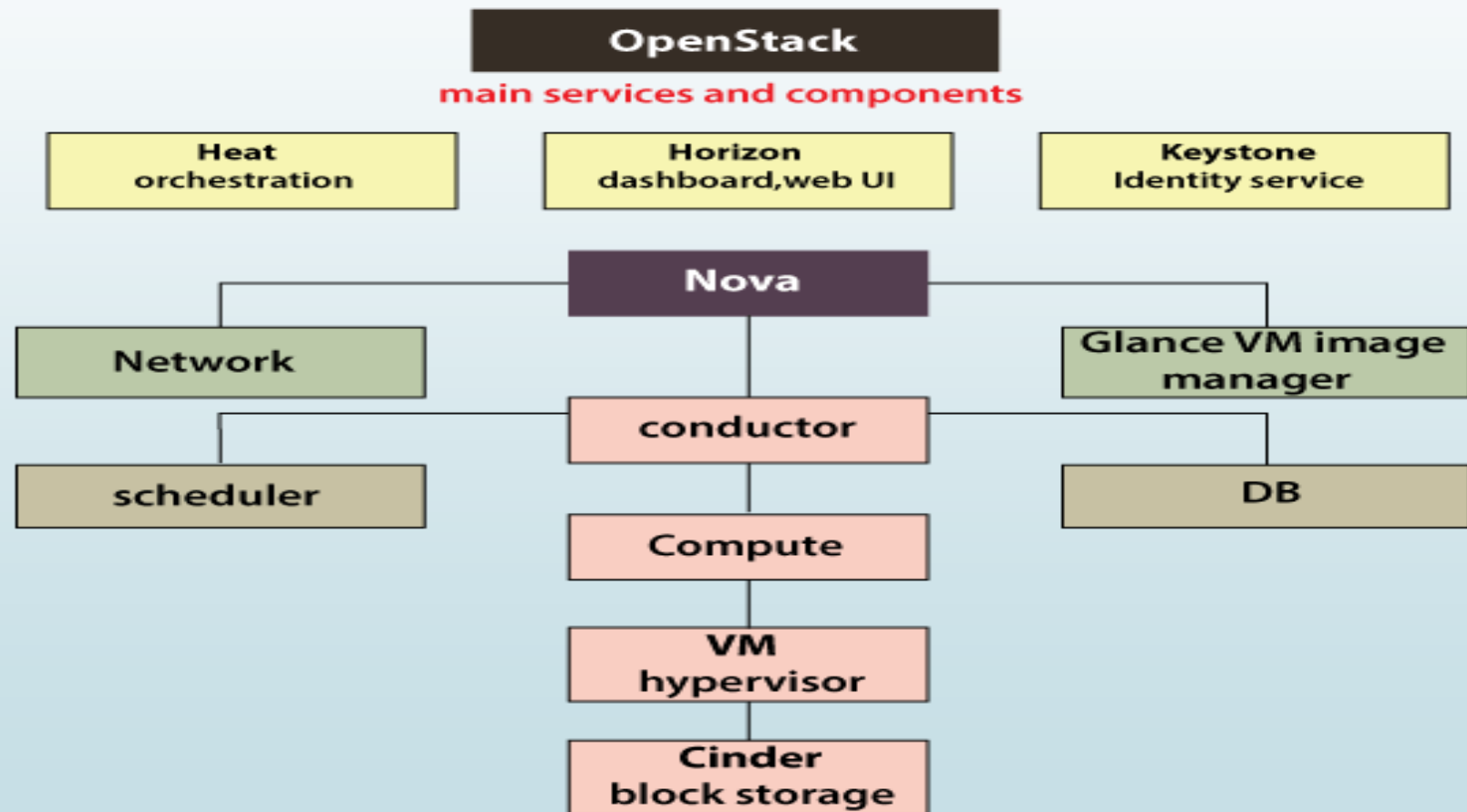
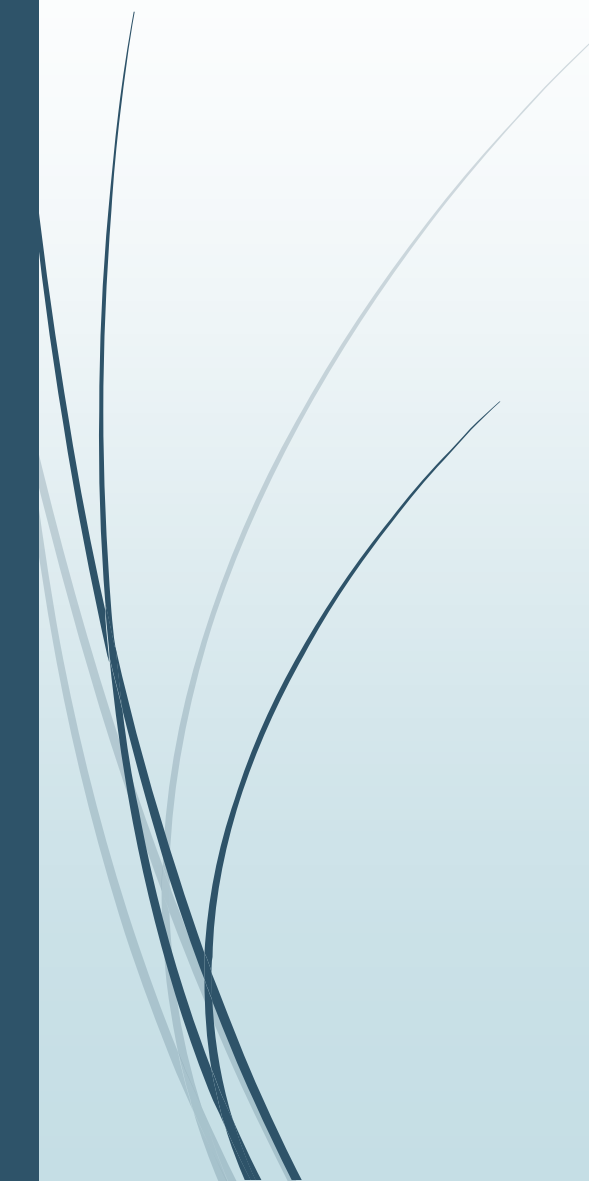
- OpenStack is a **cloud OS** that is used to control the large pools of **computing, storage, and networking resources** within a data center.
- OpenStack is an **open-source** and free **software platform**. This is essentially used and implemented as an **IaaS** for cloud computing.
- We can call the OpenStack a **software platform** that uses pooled virtual resources to **create and manage private and public cloud**.
- OpenStack offers many cloud-related services (such as networking, storage, image services, identity, etc.) by **default**.
- This can be handled by users through a web-based **dashboard**, a RESTful **API**, or **command-line** tools.
- OpenStack manages a lot of virtual machines; this permits the usage of physical resources to be reduced.

- 
- In 2010, OpenStack began as the joint project of **NASA** and **Rackspace Hosting**.
 - It was handled by the **OpenStack Foundation** which is a non-profit collective entity developed in 2012 September for promoting the OpenStack community and software.
 - 50+ enterprises have joined this project.



Basic Principles of OpenStack

- **Open Source:** Under the **Apache 2.0 license**, OpenStack is coded and published. Apache allows the community to use it for free.
- **Open Design:** For the forthcoming update, the development group holds a Design Summit every 6 months.
- **Open Development:** The developers maintain a source code repository that is freely accessible through projects like the Ubuntu Linux distribution.
- **Open Community:** OpenStack allows open and transparent documentation for the community.





Nova (Compute)

- Nova is one of the most common and **important** components of OpenStack. Compute is a **controller** that is used to handle **virtualized environments' resources**. It handles several virtual machines and other instances that perform computing tasks.
- Nova is written in **Python** language. VMware, Xen, and KVM are the hypervisor technologies that could be used, and this choice is contingent on OpenStack's version.
- Functionality :
- The **Nova-api** processes the **requests** and **responses** to and from the end-user.
- When a request is submitted, the Nova **generates** and **removes** the **instances**.
- The **Nova-scheduler schedules** nova-compute jobs.
- The **nova-network** assures **connectivity** and **routing** of the network.



Neuron (Networking)

- This component is used for **networking** in OpenStack.
- Neuron manages all the network-related queries, such as IP address management, routers, subnets, firewalls, VPNs, etc.
- It confirms that all the other components are well connected with the OpenStack.

Glance (Image)

- This component offers **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, glance allows us to use these **images** as **templates**.



Swift (Object Storage)

- To **store** and **retrieve** arbitrary data in the cloud, object storage is used.
- In Swift, it is possible to store the **files, objects, backups, images, videos, virtual machines**, and other **unstructured data**.

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.



Horizon (Dashboard)

- This is the first component that the user sees in the OpenStack.
- Horizon is the web UI **(user interface)** component used to **access** the other **back-end** services.

Keystone (Identity)

- It is the **central repository** of all the users and their permissions for the OpenStack services they use.
- 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).




Heat (Orchestration)


- Heat can be expressed as a service for **orchestrating more than one fusion cloud application** with templates by **CloudFormation adaptable Query API** and **OpenStack-native REST API**.



How does OpenStack Work?

- Basically, OpenStack is a **series of commands** which is called **scripts**.
- And these scripts are packed into **packages**, which are called **projects** that rely on tasks that create **cloud environments**.
- OpenStack relies on two other forms of software in order to construct certain environments:
- **Virtualization** means a layer of virtual resources basically abstracted from the hardware.
- A **base OS** that executes commands basically provided by OpenStack Scripts.
- So, we can say all three technologies, i.e., **virtualization**, **base operating system**, and **OpenStack** must work together.

- 
- Let's discuss how OpenStack works!
 - As we know, the **Horizon** is an interface for the appliance environment. Anything that the user wants to do should use the Horizon (Dashboard). The Dashboard is a simple graphical user interface with multiple modules, where each module performs specific tasks.
 - All the actions in OpenStack work by the service API call. So, if you are performing any task, it means you are calling a service API. Each API call is first validated by **Keystone**. So, you will have to login yourself as a registered user with your login username and password before you enter the OpenStack dashboard.
 - Once you successfully log in to the OpenStack dashboard, you will get many options to create new **instances**, **Cinder**, and **configure** the network.

- 
- **Instances** are nothing but a **virtual machine** or environment. To generate a new VM, use the 'instances' option from the OpenStack dashboard. In these instances, you can configure your cloud.
 - The formation of an instance is also an API call. You can configure **network information** in the instances. You can **connect** these instances to the **cinder instance** to add more services.
 - After the successful creation of an instance, you can **configure** it, and whatever data you want to add, you can do it.
 - Even you can set up an instance to manage and store the snapshots for future reference or backup purposes.



Benefits of OpenStack

1. Open Source

- As we know, using the open-source environment, we can create a truly defined data center.
- OpenStack is the largest open-source platform. It offers the networking, computing, and storage subsystems in a single platform. Some vendors (such as RedHat) have developed and continue to support their own OpenStack distributions.
- OpenStack source code is available at **github**.
- The two main advantages of the open-source OpenStack project is :
- OpenStack can be **modified** according to your rising demand - As per your requirement, you can add the extra features in OpenStack.
- It can be used **without any limitations** - Since OpenStack is a freely available project, so there are no limitations or restrictions to use it. You can use it as per your requirement. There are no limits for what purpose you use it, where you use it, or how long you use it.



2. Scalability

- Scalability is the major key component of cloud computing. OpenStack offers better scalability for businesses. Through this feature, it allows enterprises to spin up and spin down servers on-demand.

3. Security

- One of the significant features of OpenStack is security, and this is the key reason why OpenStack is so popular in the cloud computing world.
- With OpenStack, your data is always secure - When company owners want to move their IT infrastructure to the cloud, they always fear data loss. But there is no need to think about data loss with OpenStack. It offers the best security feature.
- OpenStack provides security professionals who are responsive to OpenStack's strong security.



4. Automation

- Automation is one of the main keys selling points of OpenStack when compared to another option. The ease with which you can automate tasks makes OpenStack efficient.
- OpenStack comes with a lot of **inbuilt tools** that make cloud management much faster and easier.
- OpenStack provides its **own API** or Application Program Interface that helps other applications to have full control over the cloud. This function makes it easier to build your own apps that can communicate with OpenStack to perform tasks such as firing up VMs.



5. Easy to Access and Manage

- We can easily access and manage OpenStack, which is the biggest benefit for you. OpenStack is easy to access and manage because of the following features :
- **Command Line Tools** - We can access the OpenStack using command-line tools.
- **Dashboard** - OpenStack offers users and administrators to access and manage various aspects of OpenStack using GUI (graphical user interface) based dashboard component. It is available as a web UI.
- **APIs** - There are a lot of APIs (Application Program Interface), which is used to manage OpenStack.



6. Services

- OpenStack provides many services required for several different tasks for your public, private, and hybrid cloud.
- List of services - OpenStack offers a list of services or components such as the Nova, Cinder, Glance, Keystone, Neutron, Ceilometer, Sahara, Manila, Searchlight, Heat, Ironi, Swift, Trove, Horizon, etc.
- Each component is used for different tasks. Such as Nova provides computing services, Neutron provides networking services, Horizon provides a dashboard interface, etc.



7. Strong Community

- OpenStack has **many experts, developers, and users** who love to come together to work on the product of OpenStack and enhance the feature of OpenStack.

8. Compatibility

- Public cloud systems like AWS (Amazon Web Services) are compatible with OpenStack.



What Is Cloud Economics?

- Cloud economics is the study of the **benefits, costs, and principles** of cloud computing.
- It refers to the knowledge about the **financial aspects** of cloud computing.
- Cloud economics involves understanding:
 - The **total cost of ownership** for cloud computing
 - The **benefits of cloud computing** over on-premises models
 - **Cost optimization strategies** when operating in the cloud.



Understanding Cloud Economics: Key Areas To Consider

1. Cloud total cost of ownership (TCO)

- In cloud computing, the **total cost of ownership (TCO)** is the total cost of **adopting, operating, and provisioning** cloud infrastructure.
- TCO is helpful for understanding your **return on investment**.
- Getting an accurate TCO for cloud computing means capturing the purchase price of **on-premises vs. cloud solutions** as well as the intangible costs of either solution.
- In practice, this means:
 - Calculating the **cost** of your current **IT infrastructure**
 - Estimating the total **cost** of **cloud adoption**.
 - **Quantifying** the intangible benefits of the **cloud**.




2. CAPEX to OPEX switch

- Cloud computing uses a **different pricing model** from traditional computing and this affects how businesses account for cost.
- The move from **capital expenses (CAPEX)** to **operating expenses (OPEX)** is a key difference, and it affects how businesses gauge **profitability** in the cloud.
- In traditional IT environments, **computing costs** are **predictable** and relatively **fixed**.
- A business pays for the computing capacity it needs upfront and uses the capacity over time.
- Calculating the total cost of ownership in this setup is fairly straightforward. In contrast, **cloud providers** adopt a **pay-as-you-go** model and most services **do not require any upfront commitment**.



3. Elasticity

- With on-premises systems and traditional IT environments, there's a cost associated with expecting demand.
- Traditional IT environments are built to expect **peaks**, which means **you buy and maintain excess computing capacity in expectation of those peak days**. For most businesses, that's a significant cost for something that's rarely — if ever — used.
- Cloud computing eliminates the need for **over-provisioning** because you pay only for what you use.
- Cloud computing platforms, such as AWS, dynamically allocate resources to projects and processes, ensuring that a business has the right amount of resources it needs at any given time. T
- his **increases cost efficiency** and allows businesses to **optimize resource usage**.

- 
- This elasticity is one of the most appealing aspects of cloud computing and a major selling point when making a case for switching to the cloud.


4. On-demand pricing


- On-demand pricing is a fundamentally different economic approach to computing power.
- Outside of the cloud, you'd buy a fixed amount of computing capacity or a physical server that you own.
- But in the cloud, you switch to on-demand pricing, so your costs become elastic.
- This means cloud costs can quickly spiral out of control if you are not monitoring them regularly and making data-driven decisions.



Service level agreements in Cloud computing

- A **Service Level Agreement (SLA)** is the **bond** for performance negotiated between the **cloud services provider** and the **client**.
- Earlier, in cloud computing all Service Level Agreements were **negotiated** between a client and the service consumer.
- Nowadays, with the initiation of large utility-like cloud computing providers, most Service Level Agreements are **standardized until a client becomes a large consumer of cloud services**.
- Service level agreements are also defined at **different levels** which are mentioned below:
 - Customer-based SLA
 - Service-based SLA
 - Multilevel SLA


- 
- Service Level Agreements usually specify **some parameters** which are mentioned below:
 - Availability of the Service (uptime)
 - Latency or the response time
 - Service components reliability
 - Each party accountability
 - Warranties

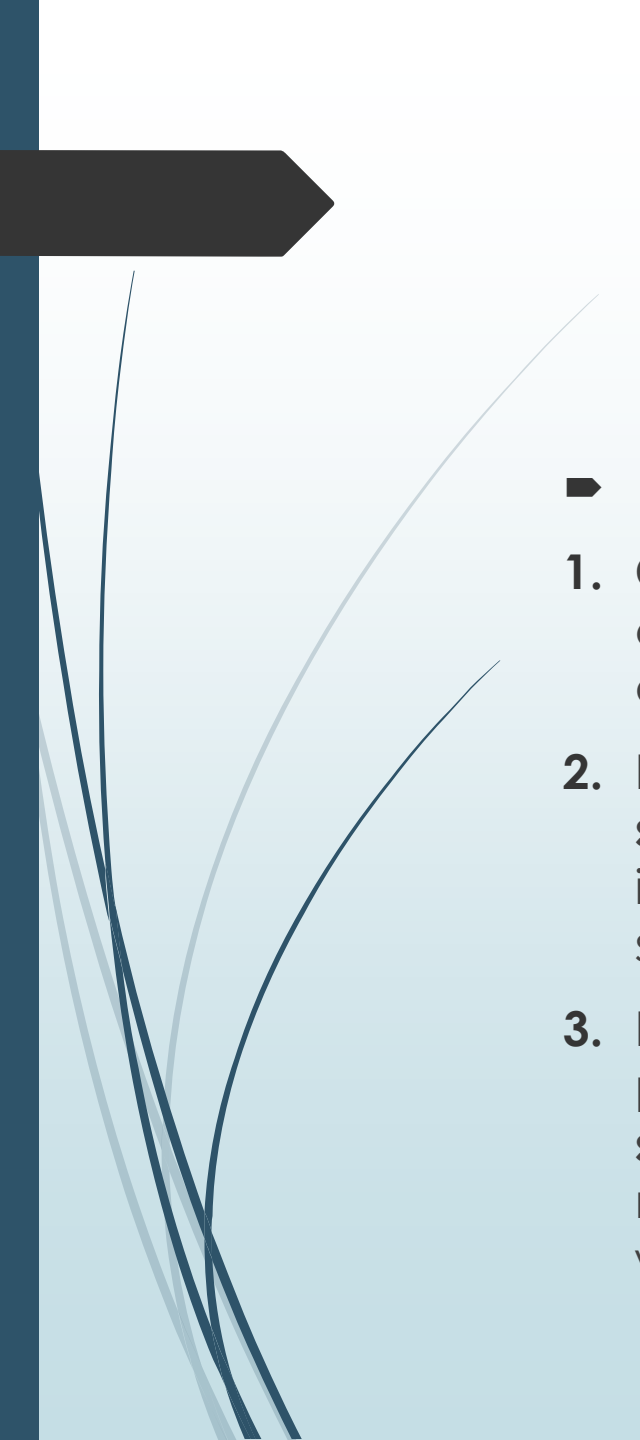
- 
- In any case, if a cloud service provider fails to meet the stated targets of minimums then the provider has to pay the **penalty** to the cloud service consumer **as per the agreement**.
 - So, Service Level Agreements are **like insurance policies** in which the corporation has to pay as per the agreements if any casualty occurs.



Resource management in cloud

- Resource management is a core function required for any cloud system.
- **Inefficient resource management** has a direct **negative effect** on performance and cost, while it can also indirectly affect system functionality, becoming too expensive or ineffective due to poor performance.
- The strategies for cloud resource management associated with the three cloud delivery models – IaaS, PaaS, SaaS – differ from one another.
- In some cases, when cloud service providers **can predict a spike**, they can **provision resources in advance** (ex. the case of seasonal web services).

- 
- However, for an unplanned spike, the situation can get more complicated. You can use Auto Scaling for unplanned spike loads, but in order to do that you need a pool of resources you can release or allocate on demand, and a monitoring system that lets you decide in real time to reallocate resources.
 - Keep in mind that Auto Scaling is supported by PaaS services, but is more difficult for IaaS due to the lack of standards.
 - Technically speaking, a cloud is a portion of cluster resources capable of growing and shrinking to accommodate the load changes.

- 
- Cloud resources are controlled on three independent levels:
 - 1. **Cluster level** – The cluster level of power management is represented by cluster resource manager, **a software complex** that manages resources and tasks in a **cluster** in order **to maintain its efficiency**.
 - 2. **Node level** – The node-level power management is done by an **operating system (OS)**, so an OS controls the high-level state of equipment. For instance, to save energy the OS can put a processor (CPU) into the sleep state or spin-down disks.
 - 3. **Hardware level** – Modern CPUs consist of many modules, which may not be permanently involved in an operation. Therefore, **unused modules can be switched off**. This is done by a special circuit responsible for internal power management of the CPU. So, all management is done on a hardware level without involving any OS.



UNIT-II

Virtualization of the resource provisioning



Virtual machine technology

- A virtual machine is a virtual representation, or emulation, of a physical computer. They are often referred to as a **guest** while the physical machine they run on is referred to as the **host**.
- Virtualization makes it possible to create multiple virtual machines, each with their own operating system (OS) and applications, on a single physical machine.
- A VM cannot interact directly with a physical computer. Instead, it needs a lightweight software layer called a **hypervisor** to coordinate between it and the underlying physical hardware.
- The hypervisor allocates physical computing resources—such as processors, memory, and storage—to each VM. It keeps each VM separate from others so they don't interfere with each other.
- While this technology can go by many names, including virtual server, virtual server instance (VSI) and virtual private server (VPS), this article will simply refer to them as virtual machines.



Key Properties of Virtual Machines

Partitioning

- Run multiple operating systems on one physical machine.
- Divide system resources between virtual machines.

Isolation

- Provide fault and security isolation at the hardware level.
- Preserve performance with advanced resource controls.



Encapsulation

- Save the entire state of a virtual machine to files.
- Move and copy virtual machines as easily as moving and copying files.

Hardware Independence

- Provision or migrate any virtual machine to any physical server.



Advantages and benefits of VMs

Resource utilization and improved ROI:

- Because multiple VMs run on a single physical computer, customers don't have to buy a new server every time they want to run another OS, and they can get more return from each piece of hardware they already own.

Scale:

- With cloud computing, it's easy to deploy multiple copies of the same virtual machine to better serve increases in load.

Portability:

- VMs can be relocated as needed among the physical computers in a network. This makes it possible to allocate workloads to servers that have spare computing power.
- VMs can even move between on-premises and cloud environments, making them useful for hybrid cloud scenarios in which you share computing resources between your data center and a cloud service provider.



Flexibility:

- Creating a VM is faster and easier than installing an OS on a physical server because you can clone a VM with the OS already installed.
- Developers and software testers can create new environments on demand to handle new tasks as they arise.

Security:

- VMs improve security in several ways when compared to operating systems running directly on hardware. A VM is a file that can be scanned for malicious software by an external program.
- You can create an entire snapshot of the VM at any point in time and then restore it to that state if it becomes infected with malware, effectively taking the VM back in time.
- The fast, easy creation of VMs also makes it possible to completely delete a compromised VM and then recreate it quickly, hastening recovery from malware infections.



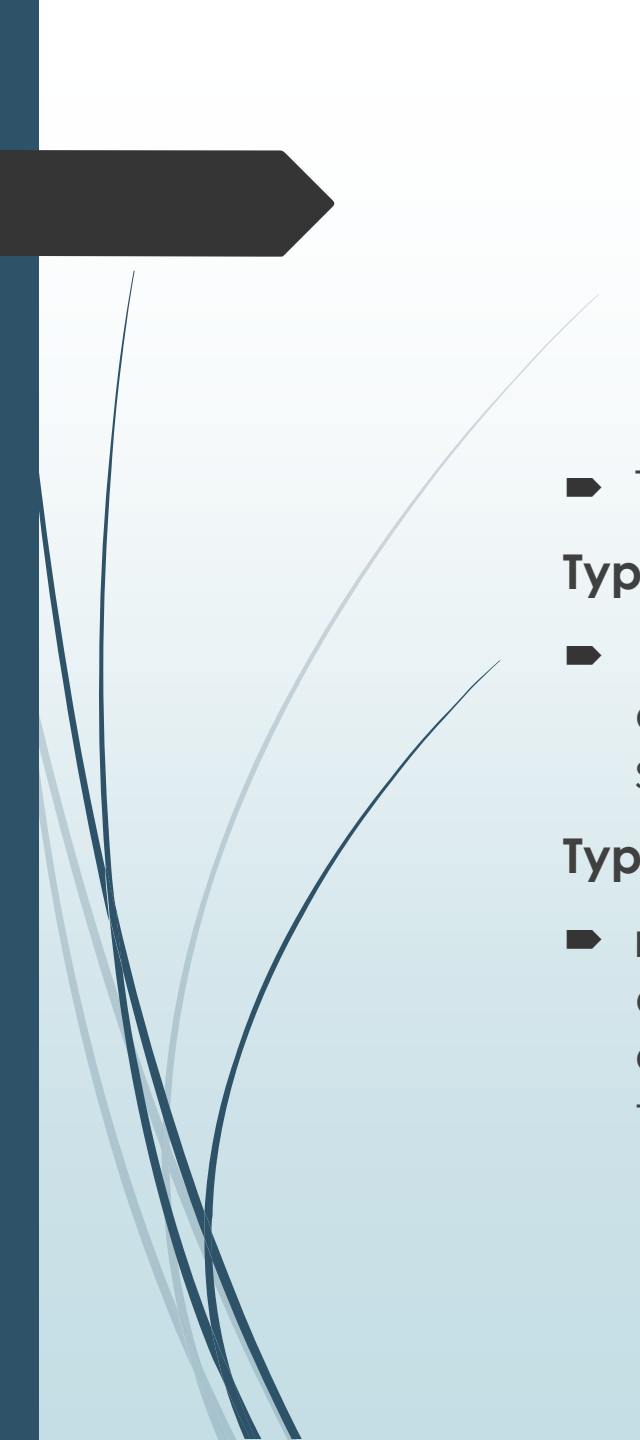
Types of VMs

- Windows virtual machines
- Android virtual machines
- Mac virtual machines
- iOS virtual machines
- Java virtual machines
- Python virtual machines
- Linux virtual machines
- VMware virtual machines
- Ubuntu virtual machines



Hypervisors(VMM)

- A hypervisor is the software layer that **coordinates** VMs.
- It serves as an **interface** between the **VM** and the underlying **physical hardware**, ensuring that each has access to the physical resources it needs to execute.
- It also ensures that the VMs don't interfere with each other by impinging on each other's memory space or compute cycles.

- 
- There are two types of hypervisors:

Type 1 or “bare-metal” hypervisors

- interact with the underlying physical resources, replacing the traditional operating system altogether. They most commonly appear in virtual server scenarios.

Type 2 hypervisors

- run as an application on an **existing OS**. Most commonly used on endpoint devices to run alternative operating systems, they carry a performance overhead because they must use the host OS to access and coordinate the underlying hardware resources.

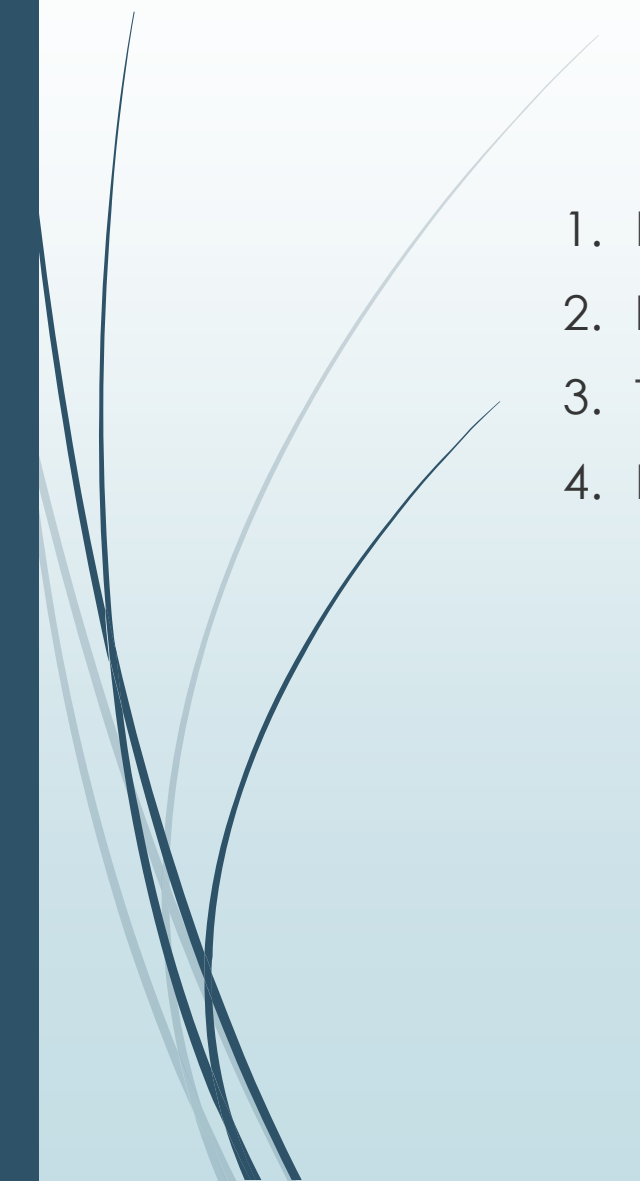


Benefits of Virtualization

1. Increases development productivity
2. Diminishes the cost of obtaining IT infrastructure
3. Provides rapid scalability and remote access.
4. More flexible.
5. Allows the user to run multiple operating systems.



Disadvantages of Virtualization

- 
1. High-cost implementation.
 2. It also poses a security risk.
 3. Time intensive.
 4. Lack of availability.



UNIT-II

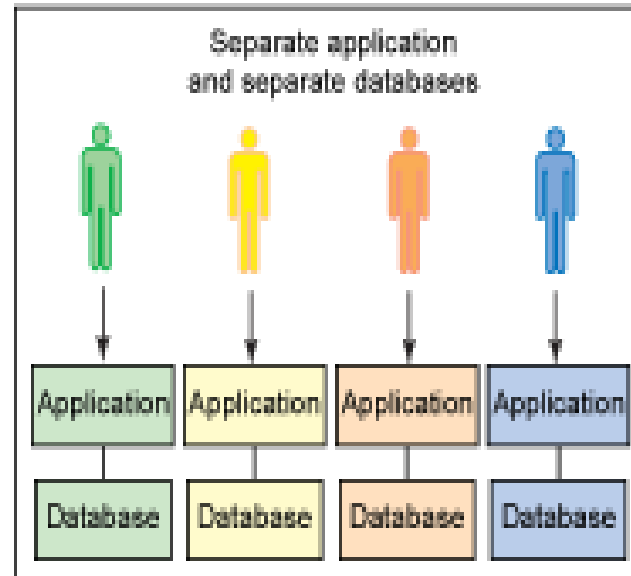
Multitenancy on offering



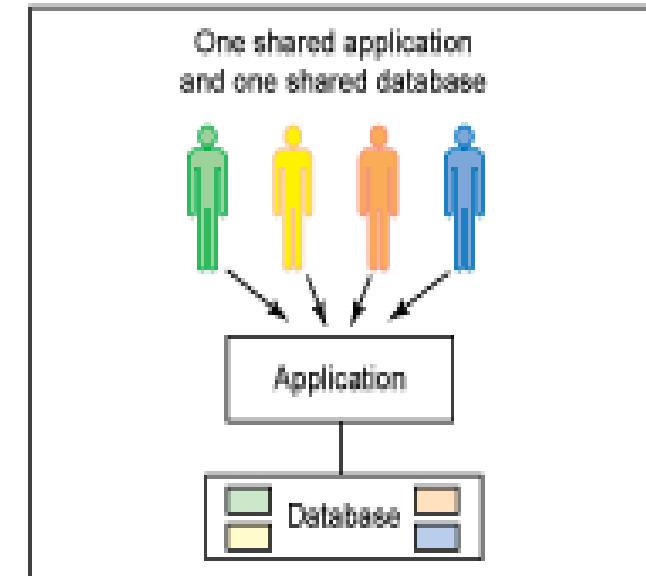
What is Multitenancy?


- Multi-tenancy refers to a **software architecture** whereby a **single instance** of a software application is served to **multiple users**.
- Each user is called a **tenant**.
- Multitenancy allows **multiple instances** of the given **application** to operate in a **shared environment**.
- Thus, a single instance of software runs on a server and **accommodates numerous tenants**.
- Tenants **integrate physically**, but they're **separated logically**.
- With this configuration, a software application residing in a multi-tenant architecture will share dedicated instances of **configurations, data, user management** among other properties.

single-tenant



multitenant



- 
- Tenants have **some measure of customization** for the shared resource, like **controlling** which users can access the resources or how the application **looks and feels**.
 - They **cannot, however, customize the code**.
 - **Example.** Consider your local bank. Many customers deposit their cash in the bank, but each has their private account, where their assets can't be touched by other clients, although everyone's keeping their funds in the same bank. Fellow bank customers don't communicate with each other, can't access each other's resources, and aren't even aware of each other's existence — one bank, many customers.



Evolution Of Multitenancy

- Multi-tenancy has evolved from a combination of three types of services- **timesharing**, **hosted applications**, and **web applications**.
- Since the 1960s, companies have been utilizing **time-sharing**, which is the sharing of computing power among multiple users to reduce computing expenses.
- Fast forward to the 1990s, where application service providers hosted software applications on behalf of their customers.
- A similar idea was developed for consumer-oriented web applications which were developed to serve multiple customers with a single instance of the software.
- A multi-tenant environment represents a direct evolution from web applications.




► Multitenancy comes in three degrees:

1. **Low:** IaaS and PaaS are multi-tenant; SaaS is single-tenant.
2. **Middle:** IaaS and PaaS are multi-tenant; small SaaS clusters are multi-tenant.
3. **High:** IaaS, PaaS, and SaaS are fully multi-tenant.



Single-Tenant vs. Multi-Tenant

- **Single tenancy** means the software and supporting infrastructure serves one customer alone.
- Each customer has their separate, **independent database or software instance**, and no one shares.
- The single tenant model comes with its own advantages.
- A single tenancy offers a more **secure environment** for starters, since each tenant's resources are entirely separate from other tenants' resources and information.
- Additionally, the customer has **full customization** and functionality control.
- Finally, single tenant customers enjoy **greater reliability** because resources are always available and abundant.

- 
- However, there are drawbacks to a single tenancy.
 - No resource sharing also means **no cost-sharing**, so a single tenancy is typically more expensive.
 - Also, single tenants have no one to share regular maintenance and setup duties with, so there's **more work involved**.
 - So, single tenancy offers greater privacy and resource availability than multitenancy.
 - However, the latter drains less of the **customer's time, resources, and money**.



Benefits Of Multitenancy

Cost-effective.

- Customers end up paying only for what they need, nothing more.
- Labor and staff, on-boarding new tenants, maintenance and development, and updates — all of these get handled by the cloud host, with only a fraction of the costs passed on and shared among the tenants.

Easily scalable.

- Building off of the advantage of cost-effectiveness, this advantage means customers can add or remove resources as needed.
- This flexibility is perfect for organizations that are growing fast but unpredictably.



Secure and offers more privacy.

- While it's true that a single tenancy is more secure, multitenancy is nevertheless still good at **threat detection** and keeping tenants' **resources separate** from each other.

Better use of resources.

- On the host's side, multitenancy makes better use of infrastructure. It makes more sense to open server access to many customers instead of limiting it to a single client.

Maintenance-free for the customer.

- The host handles tasks such as updates and upgrades, maintenance, and other related tasks.
- Consider the model like renting an apartment — the landlord handles repairs, the tenant handles paying the rent.



Convenient maintenance:

- It can efficiently manage hundreds of customers from a single interface, including **configuration, asset health monitoring, user provisioning** and much more.

Fast deployment:

- Multitenancy makes it easy to add new customers by limiting the manual work needed to get the solution implemented while vastly increasing time-to-value for the end customer.

Shared analytics and intelligence:

- Multitenancy makes it easy to apply analytics, such as dashboards and reports, across customers. It can detect and respond to threats from a single pane of glass.



Privacy and security:

- It's easy for customers to access their logs. Customers can keep sensitive data confidential while still allowing efficient threat detection and response.



Drawbacks Of Multitenancy

Less flexibility

- A multi-tenant application has less flexibility than a single-tenant application when it comes to configuring low-level customizations or modifications.
- Generally, it is not that worrying but if an application requires significant customizations for each new tenant then a multi-tenancy architecture may not be the best solution.

More complex

- A multi-tenant application is more complex than an equivalent single-tenant application.
- In a single tenancy architecture, one doesn't require any code to detect which tenant a web request is intended for or to prevent the leakage of data between tenants.
- Issues such as these are usually difficult to handle in the multi-tenant application architecture.



Response time issues


- If one tenant is utilizing an excessive amount of computing power then this could slow down the performance for the other tenants within a shared environment.

security risk

- If you choose to use a multitenancy model, there's a chance, however slight, that your data might become exposed to third parties.
- This breach could happen either accidentally, like by a system malfunction or software bug that exposes the data, or deliberately, courtesy of a hacker exploiting a weakness in the architecture.
- This risk increases or decreases relative to the level of security measures the host provides.



Multitenancy Applications

- Microsoft Office 365
 - Netflix
 - Salesforce
 - WordPress
- 

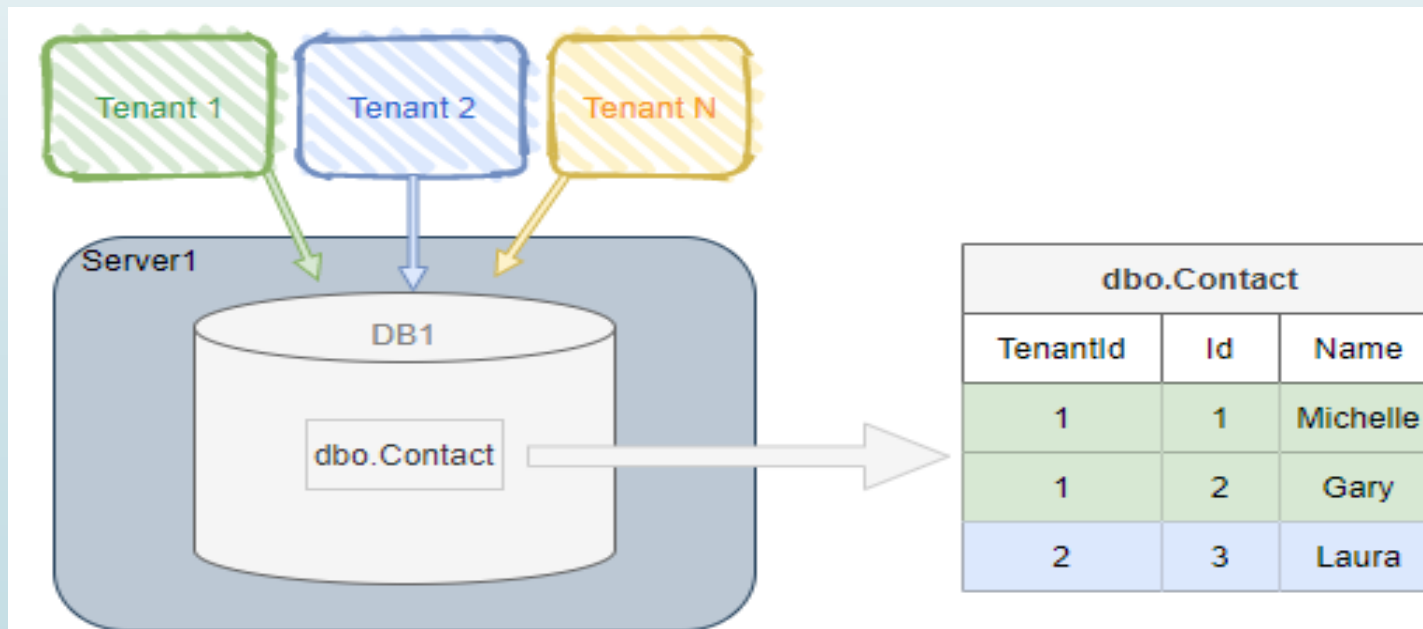


4 approaches

1. Single database, shared schema
2. Single database, separate schema
3. Database per tenant
4. Multiple databases, multiple tenants per database, shared schema

1: Single Database, Shared Schema

- One database to hold the data for all tenants
- Every tenant's data is stored in the same set of tables
- Tables that contain tenant-specific data include a column to identify which tenant each row belongs to





Security

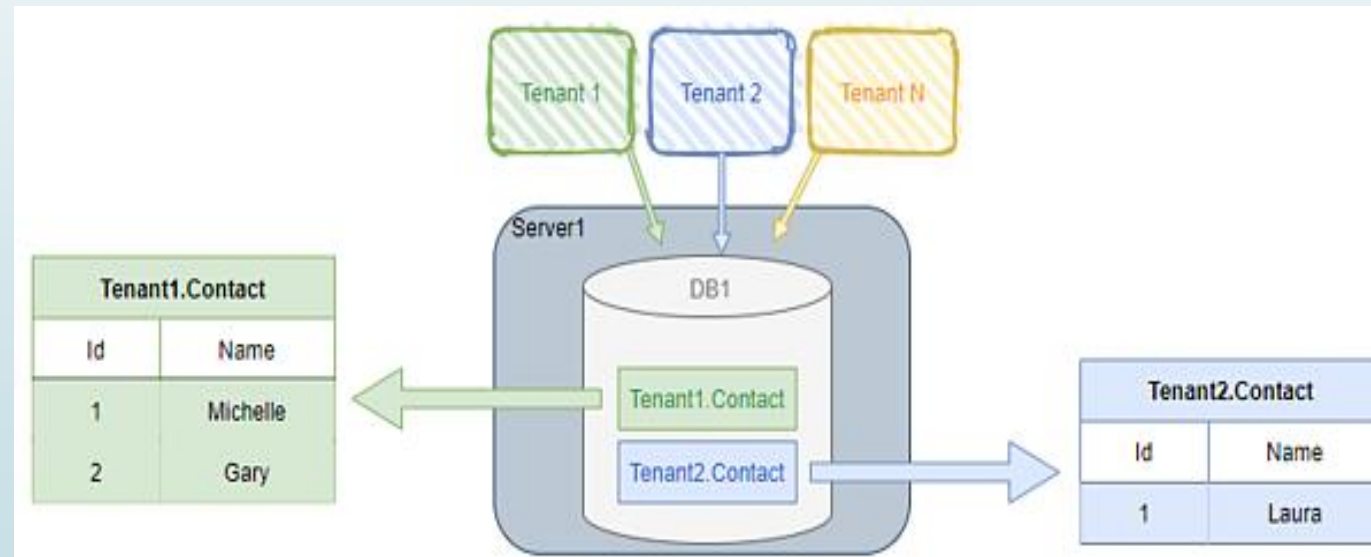
- Risk of exposing one tenant's data to another tenant or updating the wrong tenant's data (e.g., if a developer misses a WHERE clause to filter on the tenant id)
- No tenant isolation

Maintainability

- One database schema to maintain and a simple schema update rollout process—**it only needs to be applied once**
- Manage the High Availability/Disaster Recovery/maintenance operation/monitoring strategy for just one database
- Limited development/application code complexity—**single schema, single database to connect to**

2: Single Database, Separate Schema

- One database to hold the data for all tenants
- Separate tables for each tenant, each set under a tenant-specific schema





Security

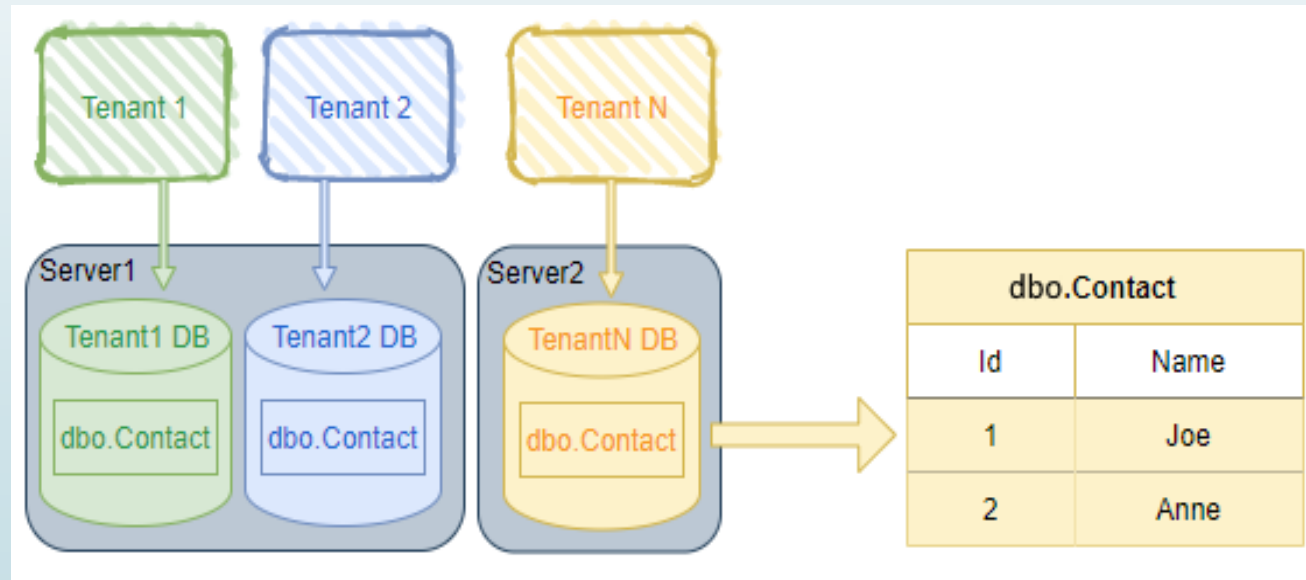
- Tenant data has some more isolation (but still within the same database)
- Risk of querying the incorrect schema
- Still limited data isolation

Maintainability

- 1 database to manage High Availability/Disaster Recovery/maintenance operation/monitoring strategy for
- Extra scope and control over some tenant-specific maintenance activities

3 : Database Per Tenant

- Each tenant has their own database





Security

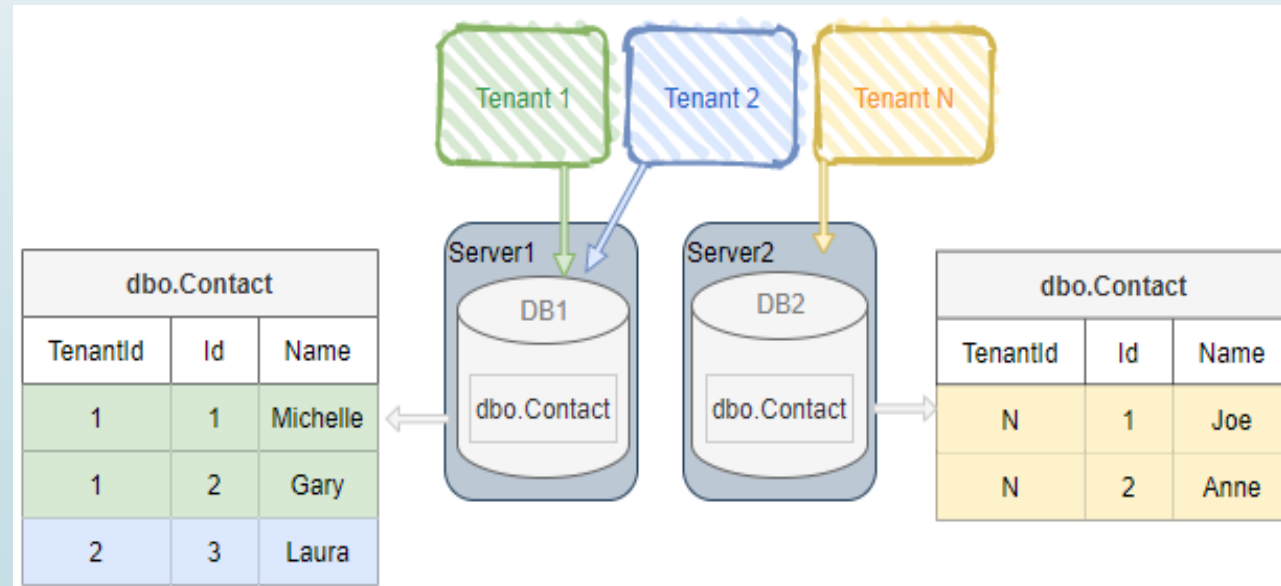
- Highest level of tenant isolation, supporting options for shared server and/or isolated servers

Maintainability

- Maintenance jobs can be managed and customized per tenant
- Can easily restore/relocate/clear down a tenant's data

4 : Multiple Databases, Multiple Tenants Per Database, Shared Schema

- Hybrid of approach #1 and approach #3
- A pool of databases exist
- Tenants share a database and schema with other tenants, but are spread over multiple databases





Security

- Some tenant isolation possible in general over approach #1

Maintainability

- More maintenance overhead than approach #1