Basic Terminologies

Cloud: The term cloud refers to a Network or internet. Cloud can provide services over public and private networks.

Computing: It is the process of using computer technology to complete a given goal oriented task. It includes :

- Designing
- Developing hardware and software systems
- Processing
- Structuring
- Managing various kinds of information
- Doing scientific research on and with the computer

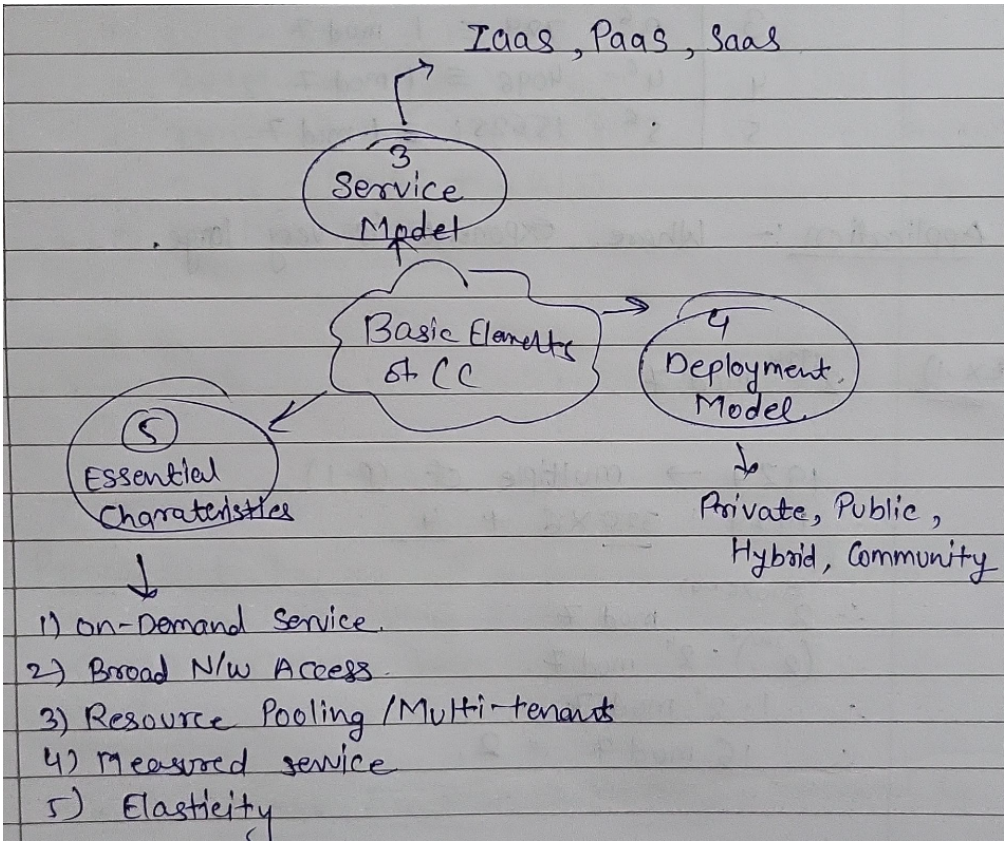
Cloud Computing: It refers to manipulating, configuring and accessing the hardware and software resources over the internet. It offers online data storage, infrastructure and application. It offers platform independency.

Bandwidth: Bandwidth in cloud computing refers to the maximum amount of data that can be transmitted over a network connection in a given amount of time, usually measured in bits per second (bps). It plays a critical role in determining the speed and performance of cloud services, affecting data transfer rates between users and cloud resources.

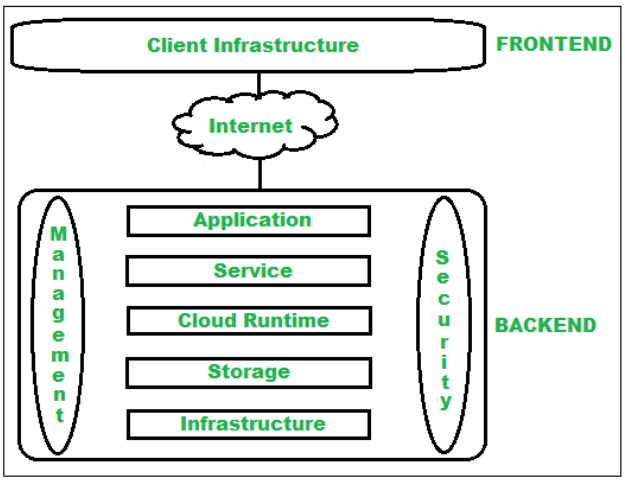
Cluster: Combination of Homogenous Nodes. A cluster composed of homogeneous nodes consists of multiple computers or servers that are identical in hardware and software configuration. This uniformity allows for efficient resource management, simplified configuration, and optimized performance across the cluster.

Grid: Combination of Heterogenous Nodes. Grid computing involves the use of a network of heterogeneous nodes—computers or servers that may differ in hardware, operating systems, and configurations. These nodes collaborate to perform complex computations, share resources, and process large-scale tasks, effectively creating a virtual supercomputer.

Basic Elements of Cloud Computing



Cloud Architecture:



Front-End

Frontend of the cloud architecture refers to the client side of cloud computing system. Means it contains all the user interfaces and applications which are used by the client to access the cloud computing services/resources. For example, use of a web browser to access the cloud platform.

Client Infrastructure

Client Infrastructure is a part of the frontend component. It contains the applications and user interfaces which are required to access the cloud platform. In other words, it provides a GUI(Graphical User Interface) to interact with the cloud.

Back-End

Backend refers to the cloud itself which is used by the service provider. It contains the resources as well as manages the resources and provides security mechanisms. Along with this, it includes huge storage, virtual applications, virtual machines, traffic control mechanisms, deployment models, etc.

Application

Application is a part of backend component that refers to a software or platform to which client accesses. Means it provides the service in backend as per the client requirement.

Service

Service in backend refers to the major three types of cloud based services like SaaS, PaaS and IaaS. Also manages which type of service the user accesses.

Types of Services

1. ***Software as a Service (SaaS)***:
   CRM, Legal, Social, Collaboration, Financial, Sales, CMS, ERP, Security, Backup and Recovery
   Is a way of delivering services and applications over the Internet. Instead of installing and maintaining software, we simply access it via the Internet, freeing ourselves from the complex software and hardware management. It removes the need to install and run applications on our own computers or in the data centers eliminating the expenses of hardware as well as software maintenance. SaaS provides a complete software solution that you purchase on a **pay-as-you-go** basis from a cloud service provider. Most SaaS applications can be run directly from a web browser without any downloads or installations required. The SaaS applications are sometimes called **Web-based software, on-demand software, or hosted software.**

   Advantages:
   Cost Effective: Pay only for what you use.
   Reduced Time: Directly install.
   Accessibility: Access Apps from anywhere.
   Automatic Updates: Users rely on the SaaS Providers to automatically perform the updates.
   Scalability: It allows the users to access the services and features on-demand
   Disadvantages:
   Limited Customization
   Depends on Internet
   Security Concerns
   Limited Control Over the Data
2. ***Platform as a Service (PaaS)***:
   Integration, Development, BI and Analytics, Data, General
   Is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet. PaaS services are hosted in the cloud and accessed by users simply via their web browser. A PaaS provider hosts the hardware and software on its own infrastructure. As a result, PaaS frees users from having to install in-house hardware and software to develop or run a new application.

Thus, the development and deployment of the application take place **independent of the hardware**. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Advantages:

Cost Effective

Simple and Convenient to use

Effectively managing Life Cycle: It is designed to support the complete web application lifecycle- building, testing, deploying, managing, and updating.

Efficiency: It allows for higher-level programming with reduced complexity thus, the overall development of the application can be more effective.

Disadvantages:

Limited Customization

Depends on Internet

Security Concerns

Limited Control Over the Data

3. ***Infrastructure as a Service (IaaS)***:

   Storage, Compute, Network, Cloud Management

   Infrastructure as a service (IaaS) is a service model that delivers computer infrastructure on an outsourced basis to support various operations. Typically IaaS is a service where infrastructure is provided as outsourcing to enterprises such as networking equipment, devices, database, and web servers. It is also known as **Hardware as a Service (HaaS).** IaaS customers pay on a per-user basis, typically by the hour, week, or month. Some providers also charge customers based on the amount of virtual machine space they use. It simply provides the underlying operating systems, security, networking, and servers for developing such applications, and services, and deploying development tools, databases, etc.

   Advantages:

   Cost-Effective: Eliminates capital expense and reduces ongoing cost and IaaS customers pay on a per-user basis, typically by the hour, week, or month.

   Website Hosting

   Limited access

   Disadvantages:

   Limited Control over Infrastructure

   Security Concerns

   Limited Access

4. ***Everything as a Service (XaaS)***:

   It is also known as Anything as a Service. Most of the cloud service providers nowadays offer everything as a service that is a compilation of all of the above services including

some additional services. It is a cloud computing model that encompasses a wide variety of services delivered over the internet. It signifies the shift from traditional on-premises solutions to cloud-based services, allowing users to access resources on a subscription or pay-as-you-go basis.

Runtime Cloud

Runtime cloud in backend provides the execution and Runtime platform/environment to the Virtual machine.

Storage

Storage in backend provides flexible and scalable storage service and management of stored data.

Infrastructure

Cloud Infrastructure in backend refers to the hardware and software components of cloud like it includes servers, storage, network devices, virtualization software etc.

Four Layers of Cloud Architecture in Cloud Computing are:

1. Physical Layer
2. Infrastructure Layer
3. Platform Layer
4. Application Layer

Four Types of Cloud Architecture in Cloud Computing are:

1. Private Cloud
2. Public Cloud
3. Hybrid Cloud
4. Multi Cloud

Cloud Deployment Model

Private Cloud

1. Dedicated to only one owner.
2. Cloud user owns information.
3. High Security.
4. Highly skilled people are required.
5. Capital Required.

Public Cloud

1. Available for all.
2. Offered over the internet to multiple organizations.
3. Resources are shared among all users.

Community Cloud

1. Private Cloud for a set of users with specific demands.
2. Several Stakeholders
3. Collaborative environment shared by multiple organizations with common interests, goals, or regulatory requirements.

Hybrid Cloud

1. Combination of two clouds.
2. Usually Private for sensitive data applications.
3. Allowing data and applications to be shared.

Types of Computing

Centralized Computing

Centralized computing refers to a system where all processing and data storage is handled by a single, central device or system. This central device is responsible for processing all requests and managing all data, and all other devices in the system are connected to it and rely on it for their computing needs.
One example of a centralized computing system is a traditional mainframe system, where a central mainframe computer handles all processing and data storage for the system. In this type of system, users access the mainframe through terminals or other devices that are connected to it.

Characteristics: Vertical Scaling

Decentralized Computing

Distributed computing refers to a model in which components of a software system are shared among multiple computers connected by a network. These systems work together to achieve a common goal, appearing as a single cohesive unit to the end-user.

Characteristics: Horizontal Scaling
Examples: Google MapReduce, Apache Hadoop, Distributed Databases as Amazon DynamoDB

Cluster Computing

A computer cluster is a set of computers that work together so that they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software. The newest manifestation of cluster computing is cloud computing.
Cluster computing is a type of computing where a group of interconnected computers (called nodes) work together as a single system to perform complex computational tasks. These computers, often physically close and connected via a high-speed local network, work in parallel to increase processing power, reliability, and availability.
Types:
High Performance Cluster (HPC)
Load Balancing Cluster
High Availability

Grid Computing

Grid computing is a computing infrastructure that combines computer resources spread over different geographical locations to achieve a common goal. All unused resources on multiple computers are pooled together and made available for a single task. Organizations use grid computing to perform large tasks or solve complex problems that are difficult to do on a single computer. For example, meteorologists use grid computing for weather modeling. Weather modeling is a computation-intensive problem that requires complex data management and analysis. Processing massive amounts of weather data on a single computer is slow and time consuming. That’s why meteorologists run the analysis over geographically dispersed grid computing infrastructure and combine the results.

Comparison of Centralized, Distributed, Cluster, and Grid Computing

Aspect	Centralized Computing	Distributed Computing	Cluster Computing	Grid Computing
Architecture	Single central server handles all processing and data storage.	Multiple interconnected nodes share resources and tasks.	Group of closely connected nodes working as a single system.	Geographically dispersed nodes collaborating over a network.
Control	Centralized control.	Can have decentralized or coordinated control.	Centralized or coordinated control for task distribution.	Decentralized control with dynamic resource allocation.
Resource Location	Resources are located in one central location.	Resources are distributed across multiple locations.	Resources are in close proximity (e.g., same data center).	Resources are spread across multiple geographical locations.
Scalability	Limited; adding resources involves significant changes.	High; nodes can be added or removed easily.	Moderate; adding nodes requires network and software support.	Very high; nodes can join or leave dynamically.
Fault Tolerance	Low; single point of failure.	Moderate; some nodes can fail without disrupting the system.	High; redundancy ensures continued operation if a node fails.	Very high; redundancy and geographical distribution improve fault tolerance.
Typical Use Cases	Legacy systems, small-scale applications, and personal computing.	Web services, distributed databases, cloud computing.	HPC (High-Performance Computing), load balancing, and HA (High Availability).	Large-scale scientific research, big data analysis, and collaborative projects.
Cost	High cost due to reliance on a single	Variable; depends on the number and type of nodes used.	Moderate; can use commodity hardware but	Low to moderate; uses existing resources but

Aspect	Centralized Computing	Distributed Computing	Cluster Computing	Grid Computing
	powerful system.		requires good interconnect.	needs complex management.
Examples	Mainframes, traditional file servers.	Google MapReduce, Apache Hadoop, microservices.	Beowulf clusters, Apache Hadoop clusters.	SETI@home, CERN's Worldwide LHC Computing Grid.

Hosting

Hosting refers to the service of providing storage space, computing resources, and network connectivity to make websites, applications, or data accessible over the internet. It involves storing website files on a server that can be accessed by users via their web browsers.

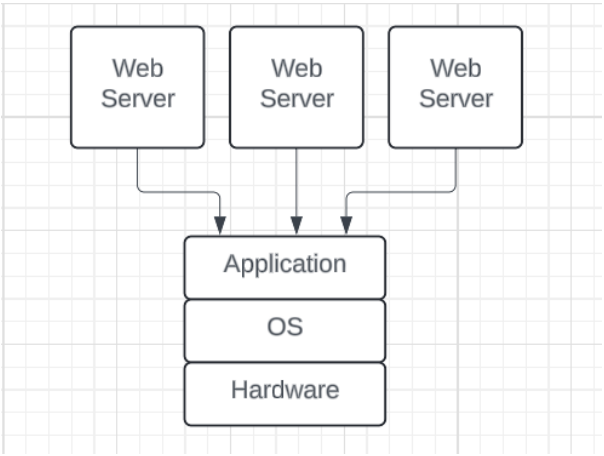
Components:

1. Server
2. Storage
3. Domain Name
4. Bandwidth
5. Control Panel

Types of Hosting

Shared Hosting

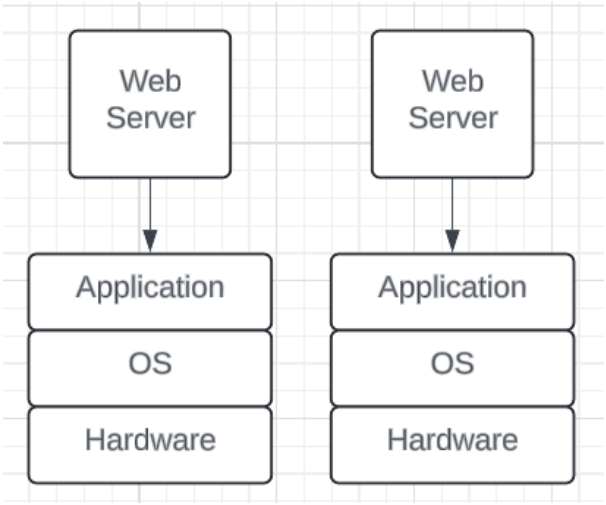
Shared hosting is a type of web hosting service where multiple websites are hosted on a single physical server, sharing its resources such as CPU, RAM, storage, and bandwidth. It is the most basic and cost-effective form of hosting, ideal for small websites, blogs, and personal projects.



Dedicated Hosting

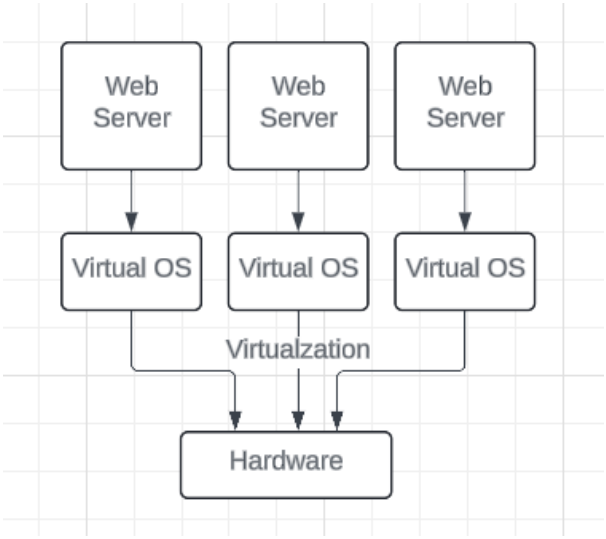
Dedicated server hosting is a type of web hosting where an entire physical server is dedicated to a single user or organization. This means that all the server's resources—such as CPU, RAM, storage, and bandwidth—are exclusively available for the user’s websites, applications,

or services.



Virtual Private Server (VPS) Hosting

Virtual Private Server (VPS) hosting is a type of web hosting that combines the benefits of shared hosting and dedicated hosting. It involves partitioning a physical server into multiple virtual servers, each running its own operating system and offering dedicated resources to users. VPS hosting provides a balance between cost and performance, offering dedicated resources and increased control compared to shared hosting. It is suitable for medium-sized businesses, e-commerce sites, and applications that require a more robust and flexible hosting solution.



Aspect	Shared Hosting	VPS Hosting	Dedicated Hosting	Cloud Hosting
Definition	Multiple websites share a single server's resources.	Virtualized servers provide dedicated resources on a single physical server.	An entire physical server is dedicated to one user.	Uses a network of virtual servers in the cloud for hosting.
Resource Allocation	Resources are shared among all users on the server.	Dedicated resources for each VPS user, isolated from others.	All resources of the server are allocated to one user.	Resources are distributed across multiple servers, scalable on demand.
Control Level	Limited control over server settings and configurations.	Greater control with root access to the virtual server.	Full control over server configuration and management.	Configurable but managed through a cloud provider's interface.
Performance	Performance can be affected by other sites on the server.	More stable performance with dedicated resources.	High performance with no competition for resources.	High performance with scalability based on demand.
Security	Lower security; vulnerabilities of one site can affect others.	Better security than shared hosting; isolated environment.	Enhanced security with complete control over security measures.	Improved security with redundancy and isolation across servers.
Cost	Most affordable option.	More expensive than shared but less than dedicated.	Highest cost among the options.	Variable cost, pay-as-you-go based on resource usage.
Ease of Use	User-friendly, ideal for beginners.	Requires some technical knowledge to manage effectively.	Requires significant technical expertise to manage.	Can be complex; some providers offer managed services.
Scalability	Limited scalability; may require	Moderate scalability; can upgrade to	Limited scalability; requires a new	Highly scalable; resources can

Aspect	Shared Hosting	VPS Hosting	Dedicated Hosting	Cloud Hosting
	migration for growth.	higher resource plans.	server for upgrades.	be adjusted dynamically.
Use Cases	Personal websites, blogs, and small businesses.	Medium-sized websites, e-commerce, and applications.	Large websites, high-traffic applications, and game servers.	Dynamic applications, high-traffic sites, and global reach.

Conclusion:
Each hosting type has its strengths and weaknesses, making them suitable for different needs. Shared hosting is great for beginners and small projects, VPS hosting offers a balance of cost and control, dedicated hosting provides maximum resources and security for large projects, and cloud hosting offers flexibility and scalability for varying workloads. Choosing the right hosting option depends on the specific requirements of your website or application.

Virtualization is used to create a virtual version of an underlying service With the help of Virtualization, multiple operating systems and applications can run on the same machine and its same hardware at the same time, increasing the utilization and flexibility of hardware.
Virtualization allows sharing of a single physical instance of a resource or an application among multiple customers and organizations at one time. It does this by assigning a logical name to physical storage and providing a pointer to that physical resource on demand.

Benefits of Virtualization:

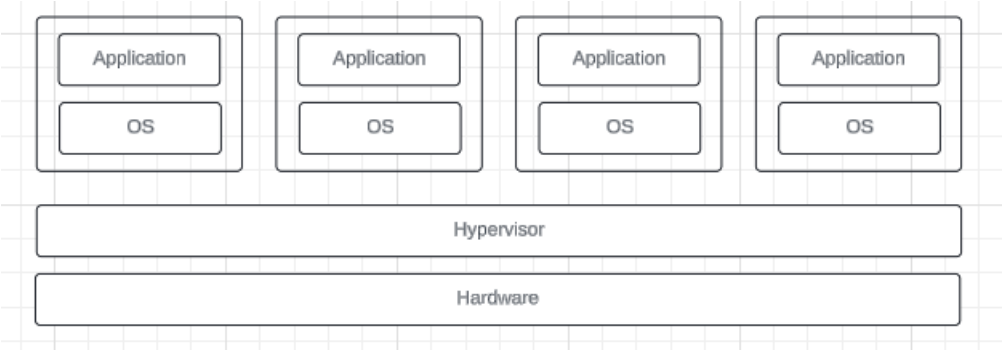
- More flexible and efficient allocation of resources.
- Enhance development productivity.
- It lowers the cost of IT infrastructure.
- Remote access and rapid scalability.
- High availability and disaster recovery.
- Pay peruse of the IT infrastructure on demand.
- Enables running multiple operating systems.

Hypervisor

A hypervisor is **a software that you can use to run multiple virtual machines on a single physical machine**. Every virtual machine has its own operating system and applications. The hypervisor allocates the underlying physical computing resources such as CPU and memory to individual virtual machines as required.
System administrators install the hypervisor software on physical servers. The hypervisor loads the virtual machine images to create multiple virtual operating systems. The physical machine is known as a *host*, and the virtual operating systems are *guests*.

Resource Allocation: The hypervisor ensures that each virtual machine receives the allocated resources as configured. It does so by acting as an intermediary between guest machines and the underlying physical hardware. The hypervisor relays requests for processing power, memory, storage, and other resources to the host machine in several ways, including API calls. An API is a software communication method that allows different applications to exchange data.

Type 1 Hypervisor

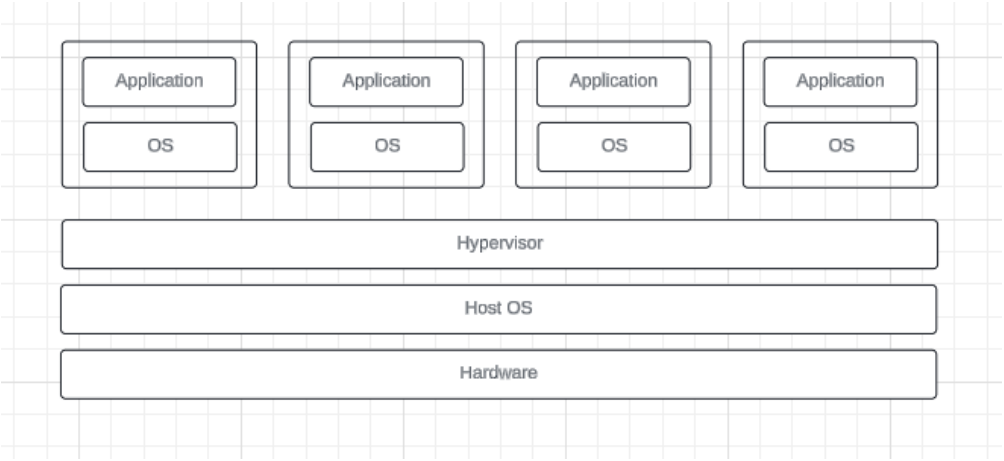


The type 1 hypervisor sits on top of the metal server and has direct access to the hardware resources. Because of this, the type 1 hypervisor is also known as a bare-metal hypervisor. The host machine does not have an operating system installed in a bare-metal hypervisor setup. Instead, the hypervisor software acts as a lightweight operating system.

Pros and Cons:

Due to its architecture, the type 1 hypervisor is very efficient. It can directly manage and allocate resources for multiple virtual machines without going through the host operating system. These types of hypervisors are also more secure, as the absence of a host operating system reduces the risks of instability.

Type 2 Hypervisor



The type 2 hypervisor is a hypervisor program installed on a host operating system. It is also known as a hosted or embedded hypervisor. Like other software applications, hosted hypervisors do not have complete control of the computer resources. Instead, the system administrator allocates the resources for the hosted hypervisor, which it distributes to the virtual machines.

Pros and Cons:

The presence of the host operating system introduces latency to the virtualized environment. When the virtual machine requests computing resources, the hypervisor cannot directly access the underlying hardware but relays the request to the host operating system. Also, the hypervisor and its hosted virtual machines are dependent on the stability of the host operating system.

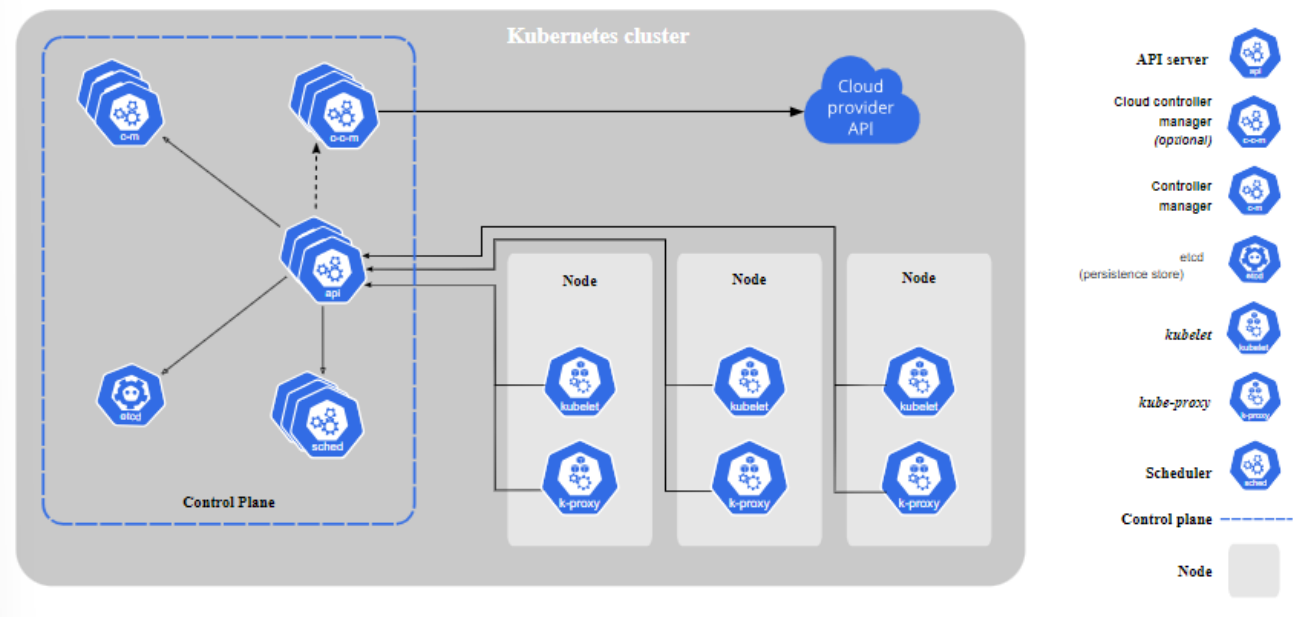
VMWare's Binary Translation for memory allocation requests

When the guest OS requests memory allocation, here's how the hypervisor handles it:

- Guest OS Request: The guest OS sends a request to allocation memory to the hypervisor.
- Hypervisor Intercept: The hypervisor intercepts the request and checks if the requested memory is available in the physical memory pool.
- Memory Pool Management: If the memory is available, the hypervisor manages the physical memory pool by allocating the required amount of memory from the free pool.
- Page Table Update: The hypervisor updates the guest OS Page Table with the new memory allocation information. This includes creating new page table entries (PTEs) for allocated memory pages.
- Memory Mapping: The hypervisor maps the allocated physical memory pages to the guest OS virtual address space. This is done by creating a mapping between the guest OS virtual addresses and the physical memory addresses.
- Guest OS Notification: Once the allocation is complete, the hypervisor notifies the guest OS that the memory allocation has been successful.

Kubernetes

Architecture



Kubernetes is a system which groups a large number of machines into a single unit that can be consumed via and API.
Types of group machines in Kubernetes:

1. ***Worker Nodes***: Run the pods and applications within them. These nodes make up the Data Plane, which runs the container images.
2. ***Control Plane Nodes***: Run the Kubernetes Control Plane, which manages the worker nodes. These nodes make up the Control Plane, which acts as the "brains" of the cluster.

Unix Philosophy of Many Components

- K8S is a collection of different applications, every application ignorant of others which work together to implement the overall system
- Even though there is a single binary (e.g. controller manager) implementing multiple functionalities.
  - Every functionality is implemented independent of others
  - They are compiled together to ease the deployment and management
  - Every components communicates with the API server rather than directly within running progress
- Advantages of Modular Approach
  - Customization of Functionality: Large pieces of functionality can be ripped out and replaced without impacting rest of the system.

- Results into increased complexity

Kube-proxy

- Every service in Kubernetes gets a virtual IP address
- Kube-proxy is responsible for implementing the Kubernetes service load-balancer networking model
- Watches the endpoint objects for all services in the K8S cluster
- Programs the network on its node so that all network requests to the virtual IP address of a service are routed to the endpoints that implement the service.

Kubelet

- Node daemon for all machines that are part of K8S cluster
- A bridge that joins available CPU, disk and memory for a node intro large K8S cluster
- Scheduling and Reporting
  - From API Server, retrieves the information about PODs schedule to run on specific node
  - Communicates the state of PODs running on the node to the API server
    - Enables the other reconciliation loops to observe the current state of PODs
  - Health Checking and Healing
    - Performs health checks and restarts the containers running on the node
    - Instead of pushing the health-state information to API server and let reconciliation loops take action to fix the health, the "local" healing is more efficient.

API Server

- Mediates all the interaction between clients and API Objects stored in the etcd.

API Management

- For every request, API server follows RESTful API pattern
- All K8s requests begin with path /api/ (core APIs e.g. pod, service) and /apis/ (batch)
- Component of Paths for namespaced resource
  - ``/api/v1/namespaces///resource-name>
  - `/apis/<api-group>/<api-version>/namespaces/<namespace-name>/<resource-type-name>/<resource-name>`

API Discovery

API discovery is an essential process within the API lifecycle, referring to the mechanisms by which available APIs and their associated endpoints, functionalities, and data structures are found and cataloged.

- Service identification: API discovery helps identify all APIs in use within an organization. This helps in avoiding redundancy, identifying potential security risks, and promoting the best use of resources.
- API documentation: Discovered APIs should be well documented. Good documentation includes information about the API's functionality, its endpoints, how to authenticate, and examples of responses. This makes it easier for developers to understand and use the API.
- Integration and interoperability: By identifying available APIs, API discovery supports better integration and operations between different systems and applications within an organization.

API Translation

- API Version Lifecycle
  - v1alpha1
  - v1beta1
  - v1

Scheduler

- Scans the API servers for unscheduled objects.
- Determines the best nodes to run the objects.
- Responsibilities:
  - Watches for newly created PODs without any nodes assigned.
  - Identifies the best node for scheduling the POD
- Scheduling Process:
  - Filtering: Shortlists the node(s) based on the criteria
  - Scoring: Ranks the feasible nodes

Scheduler

- Predicates (for Filtering Nodes)
  - Indicates whether a POD fits onto a particular node
  - Hard constraints which if violated, lead to POD not operating correctly
    - POD memory requirements
    - Node selector label query

- Priorities (for Scoring Nodes)
  - Generic interface used by scheduler to determine preference of one node over another
    - Priority Functions
      - Assumes that POD can be scheduled onto the node
      - Score relative value of scheduling a POD onto a particular node
        - Weigh nodes where image has already been pulled
        - Spreading Function: Prioritizes nodes where PODs that are members of same Kubernetes service are not present
      - Predicate values are mixed together to achieve final priority score for a node
    - Node with best score is selected for POD scheduling

Controller Manager

- Consists of reconciliation control loops for various functions.
- Ensure that desired declarative state is maintained.

Master Node Components

etcd

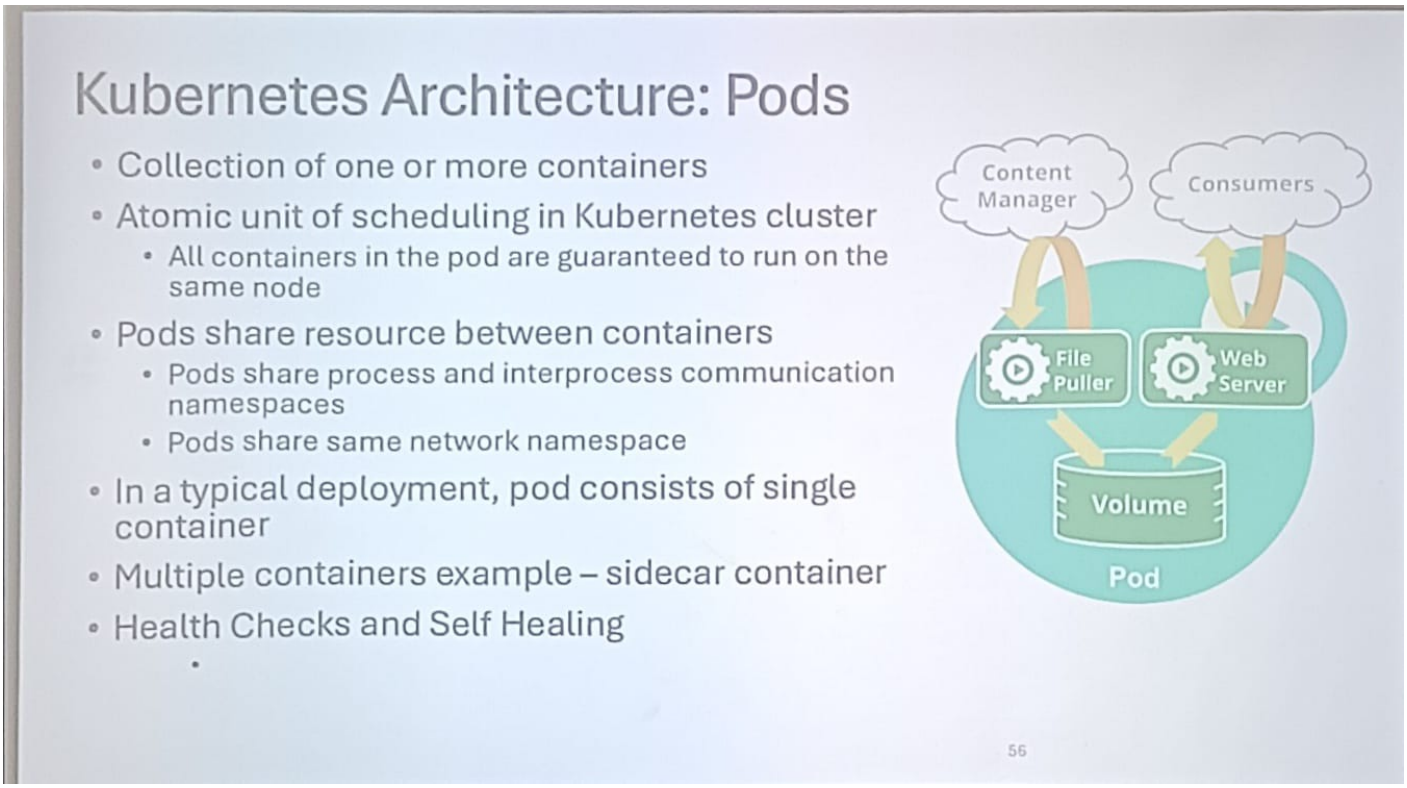
- Implements key-value store to persist Kubernetes cluster objects
- Change Notification
  - Implements "a watch" protocol. enabling the client to watch for changes in the key-value stores for an entire directory of values.
  - Eliminates the need for continuous polling.

Services

- Each service gets three things:
  1. It's own IP Address (Virtual)
  2. DNS entry in Kubernetes cluster DNS
  3. Load balancing riles that proxy traffic to the PODs that implements the service.
- Load Balancing
  - Service load balancing is programmed into network fabric of Kubernetes cluster's DNS server.
  - Any container that tries to talk to the Service IP address is correctly balanced to corresponding pods.
  - Dynamic update of network fabric as pods are scaled out/in.

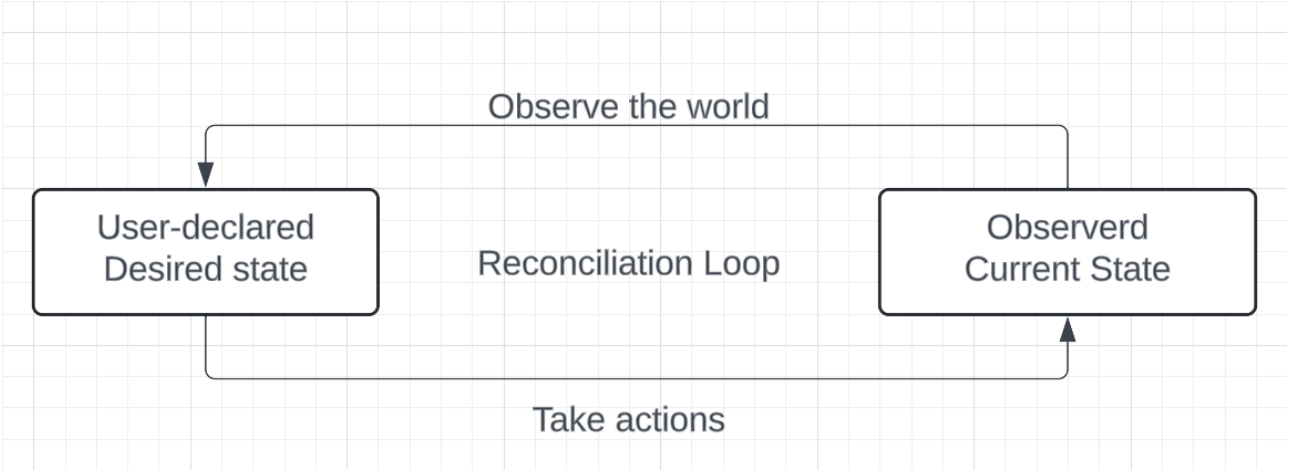
- Clients can rely on Service IP address to resolve to a Pod which implements the service.

PODs



Control Loop

1. Obtain the desired state of the world.
2. Observe the world.
3. Find the difference of the world and the desired state of the world.
4. Take Actions to make the observation of the world match the desired state.



Concepts

Configurations

1. Declarative Configuration:
- One of the primary drivers behind development of K8S.
- Specify the desired state.
- K8S understands the desired state, it can take autonomous action, independent of user interaction.
- It can implement autonomous self-correcting and self-healing behaviors.
- K8S ensures that the state above is reached.

2. Imperative Configuration:
- User takes a series of direct actions.
- Simpler to understand but needs user interaction.

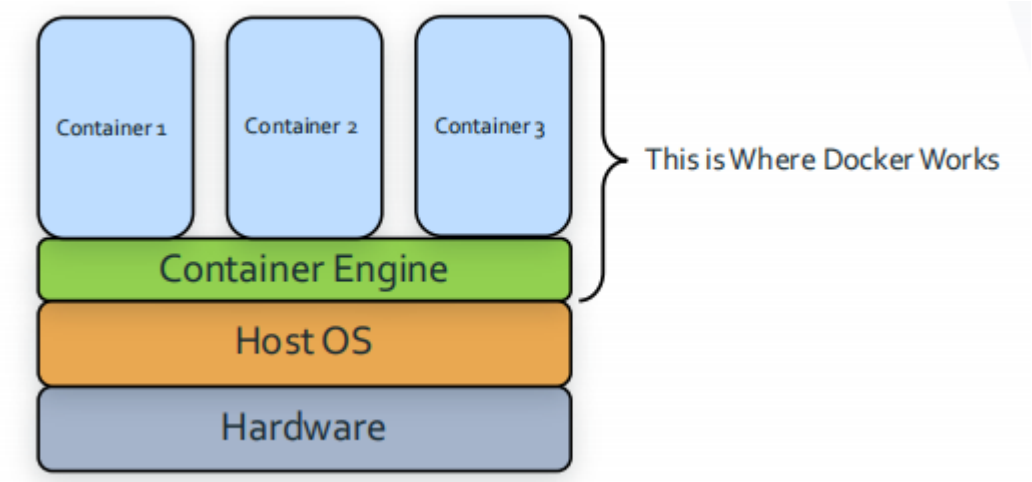
Dynamic Grouping

1. Explicit Grouping/Static Grouping:
1. Group is defined by static list
2. Explicitly refers to object names
3. Not flexible and can't respond to dynamically changing world

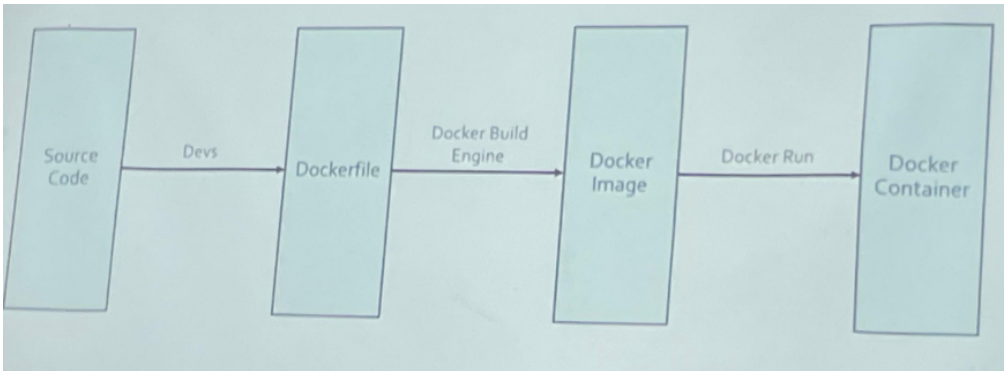
2. Dynamic/Implicit Grouping:
1. Group doesn't explicitly refer to the members
2. Achieve in K8S via labels and label queries/label selectors

Docker

Docker is a software platform that allows us to build, test, ad deploy applications on Containers quickly.



Start to finish with Docker



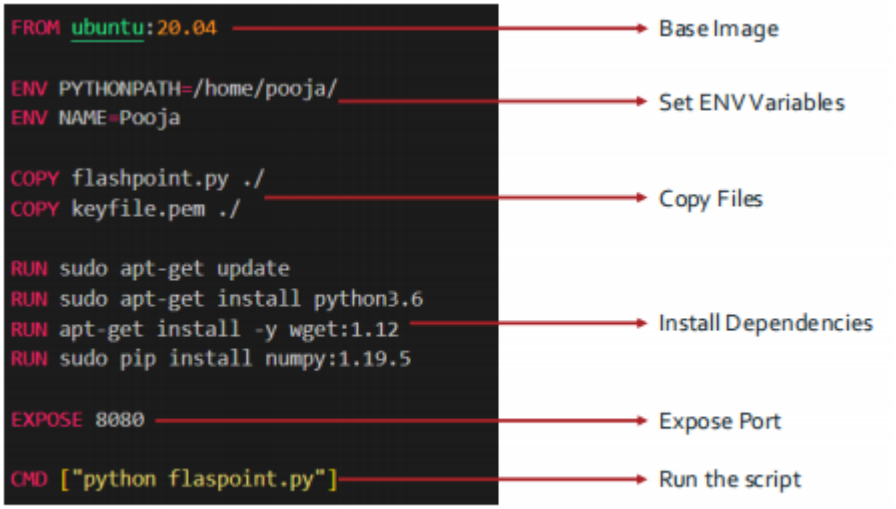
Container Registry

- A container registry is essentially acts as a place for developers to store container images and share them out via a process of uploading (pushing) to the registry and downloading (pulling) into another system.
- A Container Registry can have multiple mages. Each Image can have multiple versions. Each Image Identified by the tag (version) or Its own unique hash.
- Container Registries make efficient Use of storage space by sharing any layers that me common to more than one image.

Dockerfile

- A Dockerfile is basically a text document that contains all the commands a user would perform, from start to finish, to create the environment.

- Dockerfile is essentially the build instructions to build the Docker Image.
- The commands or Instructions in the Dockerfile are executed successively to perform actions on the base image.



Dockerfile

Docker Image

- A docker image is a read-only template containing instructions for creating a container.
- An image is like a blueprint of the container when it runs. These images contain the code or binary, runtimes, dependencies, and other filesystem objects to run an application.
- An image is composed of multiple stacked layers, with each layer changing something in the environment.
    - Layer 1: Pull base image
    - Layer 2: Set Environment Variables
    - Layer 3: Copy files
    - Layer 4: Install Dependencies

Containerization

It is the software deployment process of packaging an application into a single lightweight executable called a container. The container includes the application's code, the OS files, Libraries, and other dependencies, bundled together. This ensures that the software can run on any infrastructure.
Advantages of Containerization:

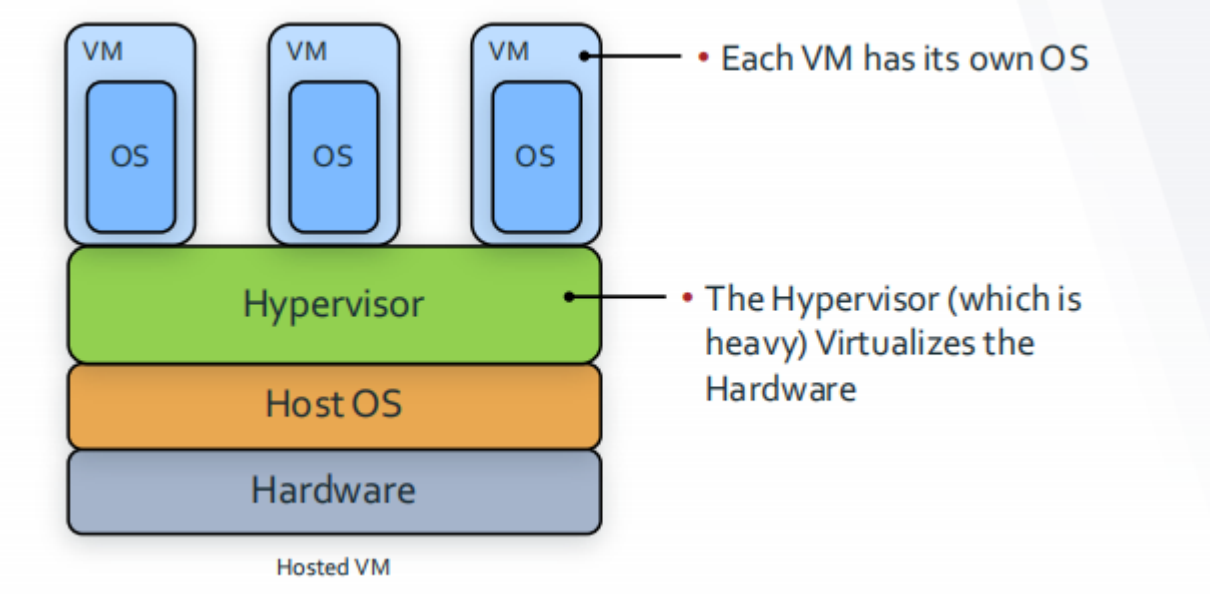
- Portability
- Speed
- Scalability
- Fault Isolation

- Ease of Management
- Developer Friendly

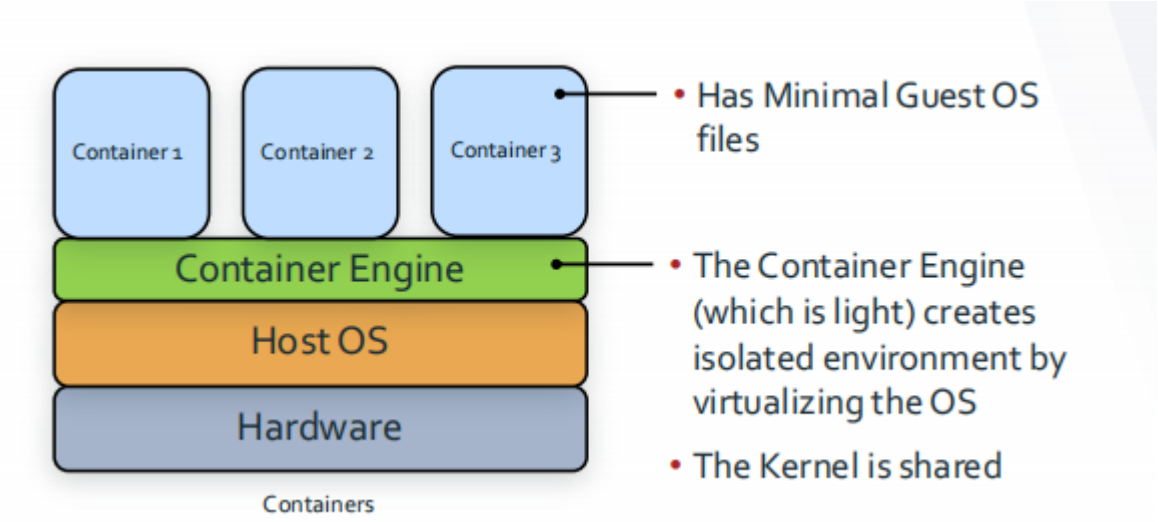
Container

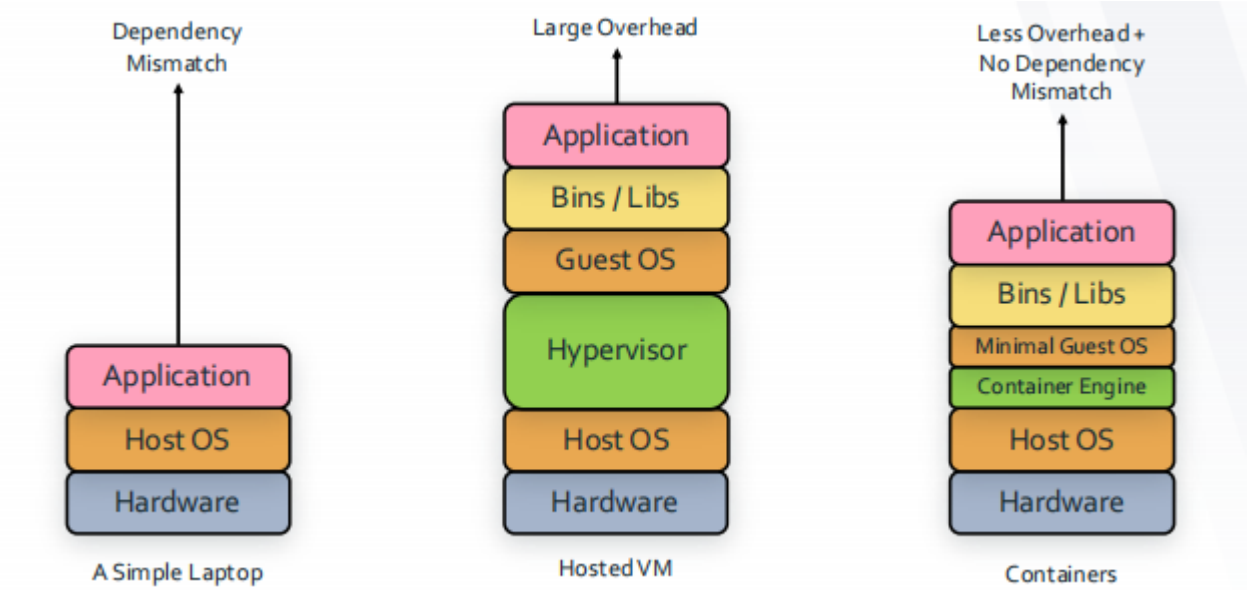
- A Container is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
- It contains less overhead compared to a VM and removes the dependency mismatch issue.
- Containers work just like a VM but contains bare minimum OS Files unlike a CM, which houses the entire OS.

Virtual Machine



Container





YAML Files

DaemonSets

- Continues running
- 1 pod / node
- Bypass Scheduler

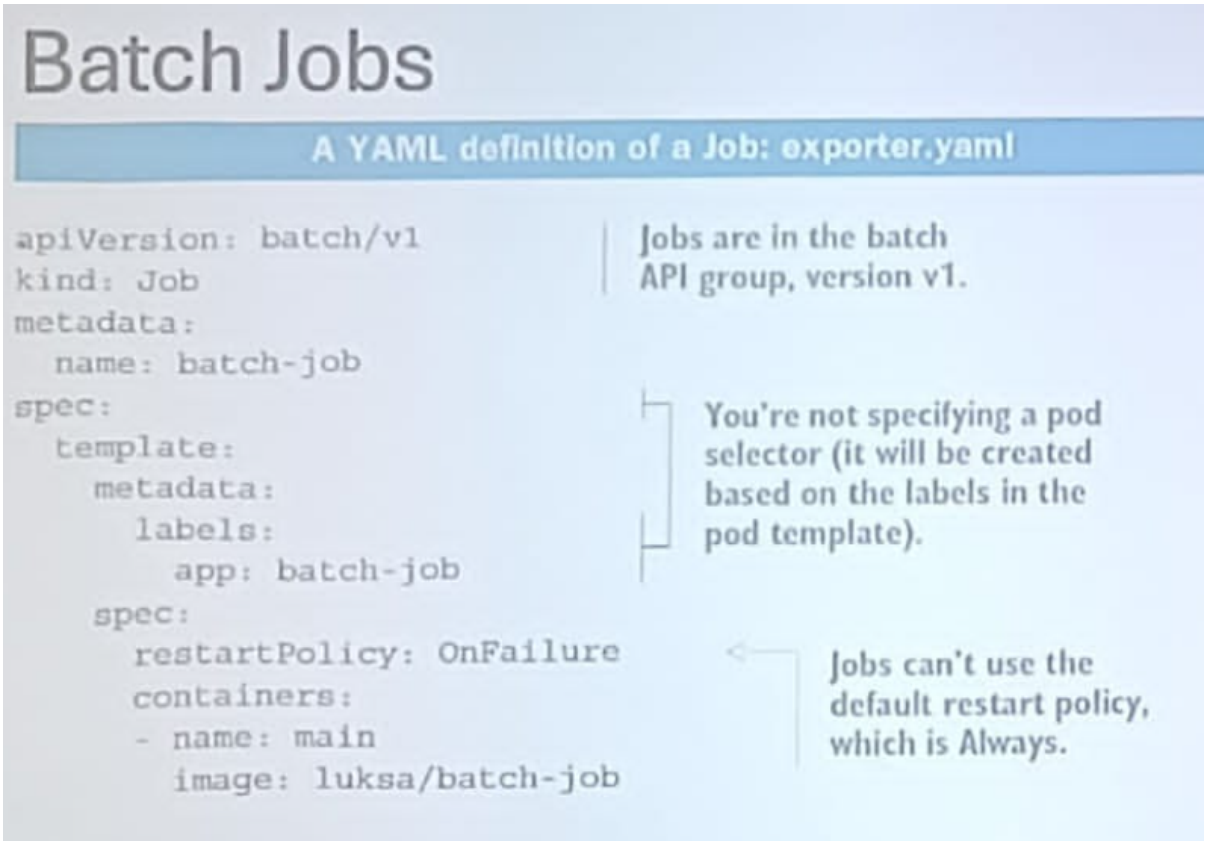
A YAML for a DaemonSet: ssd-monitor-daemonset.yaml

```
apiVersion: apps/v1beta2
kind: DaemonSet
metadata:
  name: ssd-monitor
spec:
  selector:
    matchLabels:
      app: ssd-monitor
  template:
    metadata:
      labels:
        app: ssd-monitor
    spec:
      nodeSelector:
        disk: ssd
      containers:
      - name: main
        image: luksa/ssd-monitor
```

DaemonSets are in the apps API group, version v1beta2.

The pod template includes a node selector, which selects nodes with the disk=ssd label.

Batch Jobs



CronJobs

A cron job is a scheduled task that runs automatically at a specified time or interval.

```
apiVersion: apps/v1
metadata:
        names: frontent
        labels:
                app: appname
                tier: frontend
spec:
        replicas: 3
        selector:
                matchLabels:
                        tier: frontend
        template:
                metadata:
                        labels:
                                tier: frontend
                spec:
```

```
                        containers:
                        - name: <container-name>
                        - image: <image-url>
```

Interoperability

The ability of different system / app / products working together seamlessly without special effort on that part of user.
Data interoperability: Data exchanged between different systems.

Broad Aspects of Migration into Cloud

1. Planning and strategy Development

- Assessment of the Current Environment:
  Understanding the existing IT infrastructure applications and data to determine what can be moved to the cloud and what should remain on-premises.
- Migration Goals:
  Defining clear objectives for the migration such as cost savings scalability improved performance or enhanced security.
- Cloud Services Models:
  Deciding between IaaS, PaaS or SaaS depending on organizational needs.
- Cloud Provider Selection:
  Evaluating different cloud providers based on factors like cost, services offered, compliance and support.

2. Application and Data Migration

- Rehosting (Lift and shift):
  Moving applications and the data to the cloud with minimal changes. This is the fastest method but may not fully leverage cloud capabilities.
- Replatforming:
  Making a few cloud-optimizations without changing the core architecture of applications.
- Refactoring/Re-Architecting:
  Rebuilding applications without changing the core architecture of applications.
- Data Migration:
  Transferring data to cloud, may involve large datasets. This requires careful planning to maintain data integrity and minimize downtime.

3. Security and Compliance

- Data Security:
  Ensuring data is encrypted in transit and at rest and that appropriate access controls are in place.

- Identity and Access Management (IAM):
  Implementing IAM policies to manage user access to cloud resources securely.
- Compliance:
  Ensuring that the migration complies with industry-specific regulations and standards such as GDPR, DIPAA or PCI-DSS.
- Threat Detection and Response:
  Setting up systems to detect and respond to potential security threats in the cloud.

4. Cost Management

- Cost Estimation and budgeting:
  Estimating costs for cloud services and setting a budget for the migration process.
- Cost Optimization:
  Identifying and eliminating unnecessary costs, such as over-provisioned resources or unused services.
- Billing and Monitoring:
  Setting up tools to monitor cloud usage and manage billing to avoid unexpected costs.

5. Performance and Scalability

- Load Testing and Optimization:
  Testing applications and infrastructure in the clod to ensure they meet performance requirements.
- Auto-Scaling:
  Implementing auto-scaling to handle varying workloads without manual intervention, ensuring that resources scale up or down based on demand.
- Global Reach and Latency:
  Leveraging the cloud's global infrastructure to improve application performance for users in different geographic locations.

6. Change Management and Training

- Stakeholder Communication:
  Engaging with stakeholders throughout the migration process to ensure alignment and manage expectations.
- Training:
  Providing training for IT staff and end-users on the new cloud environment, tools, and processes.
- Change Management:
  Managing the organizational and cultural changes that come with cloud adoption, including shifting to a cloud-first mindset.

7. Change Management and Training

- Risk Assessment:
  Identifying potential risks associated with cloud migration, such as data loss, downtime, or security breaches.
- Disaster Recovery and Backup:
  Setting up disaster recovery plans and ensuring regular backups to protect against data loss.
- Business Continuity Planning:
  Ensuring the critical business functions can continue operating smoothly during and after the migration.

8. Post-Migration Optimizations and Governance

- Performance Monitoring:
  Continuously monitoring the performance of cloud resources and applications to ensure they meet organizational goals.
- Ongoing Optimizations:
  Regularly reviewing cloud services and configurations to optimize performance, security and cost.
- Governance Frameworks:
  Establishing governance policies and frameworks to manage cloud resources effectively including access controls, data management, and continuous monitoring.

9. Multi-Cloud and Hybrid Cloud Strategies

- Multi-Cloud:
  Leveraging services from multiple cloud providers to avoid vendor lock-in and to optimize services based on specific needs,
- Hybrid-Cloud:
  Integrating on-premises infrastructure with public and/or private cloud environments to create a seamless hybrid cloud architecture.

10. Environmental and Sustainable Considerations

- Energy Efficiency:
  Evaluating the energy consumption of cloud services and choosing that aligns with the organization's sustainable goals.
- Green Cloud Computing:
  Partnering with cloud providers that have strong commitments to renewable energy and reducing carbon footprints.

Backing up VM

Virtual Machine Backup is the process of backing up the virtual machines (VMs) running in an enterprise environment. We usually run as guests on hypervisors that emulate a computer system, and allows multiple VMs to share a physical host hardware systems.

Terms

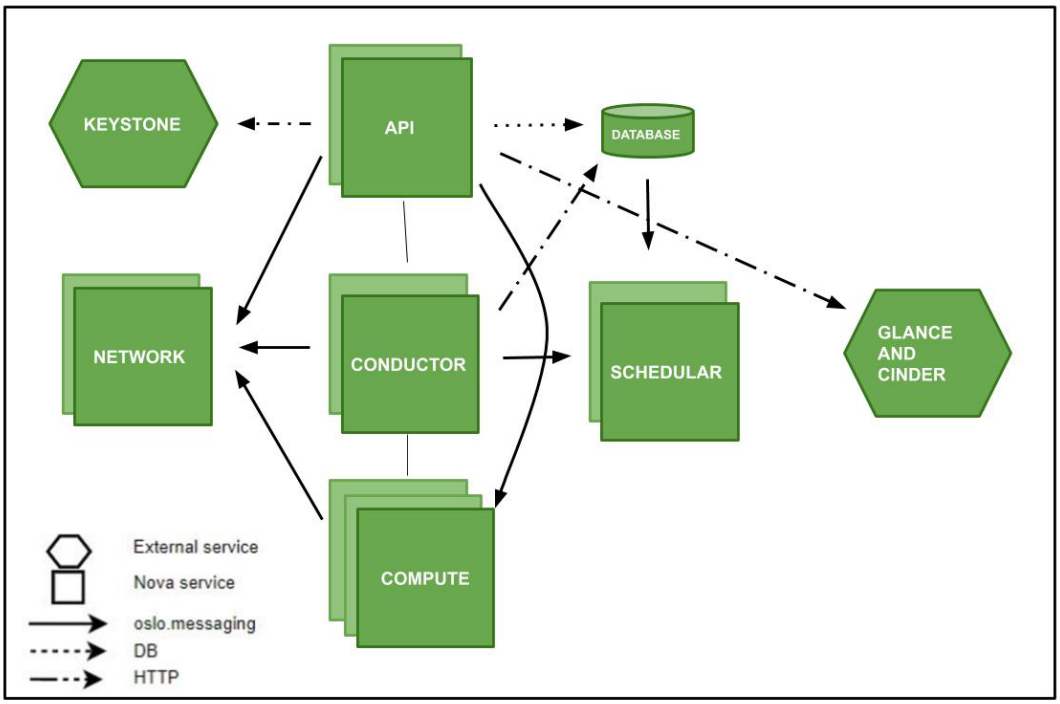
- Differential backup – Backup of only the files that have changed since the last full backup.
- File-level backup – Backup that is defined at the level of files and folders.
- Full backup – A backup of all files.
- Full VM backup – A backup of all files that comprise an entire virtual machine, including disk images, configuration files, and others.
- Image-level backup – or volume level is the backup of the entire storage volume.
- Incremental backup – Backup of only the files that have changed since the last backup, either full or incremental.
- Quiescing – A process to bring the data of the virtual machine to a state suitable for backups including, flushing of buffers from the operating systems memory cache to disk or other application specific tasks

Restoring VM

Full Virtual Machine Restore provides users to restore an entire virtual machine from a backup file to the latest state or a previous point in time if the primary virtual machine fails.
Restoring a full virtual machine backup creates a new virtual machine; the configuration information and content of the new machine is identical to what it was when the backup occurred. All virtual machine disks are restored to the specified point-in-time, as virtual disks in the newly created virtual machine

High Availability Clusters

High availability clusters are typically used for load balancing, backup, and failover purposes. To successfully configure a high availability (HA) cluster, all hosts in the cluster must have access to the same shared storage. In any case of failure, a virtual machine (VM) on one host can failover to another host, without any downtime.



The following services are required for the basic functioning of OpenStack:

1. **Keystone:** Use for the authentication purpose of the user.
2. **Glance:** Works as a storage service that provides compute image repository.
3. **Neutron:** This service is responsible for network provisioning.

Purpose

1. Manages VM instances
2. Provides compute resources in a cloud environment.
   Role in Migration:
3. Handles instance migration across compute nodes.
4. Supports live and cold migrations for high availability and resource balancing.
   Key Features:
5. Hypervisor support
6. Auto-Scaling and orchestration capabilities.
7. Integrates with other services like Neutron and Cinder for networking and storage.

Keystone

- Provide authentication and authorization services.
- Manages users, roles, and service catalogs.
- Ensures secure access to OpenStack services during migration process.

- Validates tokens for users and services involved in the migration process.
- Multi-tenancy support for project isolation.
- Centralized identity management..
- Token based and password-based authentication methods.

Neutron

- Manages networking services, including virtual networks, routers, and IP addressing.
- Ensure network connectivity when instances are migrated across nodes.
- Manages floating IPs and network interfaces during migration.
- Support VLANs, VXLANs, and SDN (Software-Defined Networking)
- Integrates with security groups and firewalls.
- Provides services like DHCP, DNS, and load balancing.

Cinder

- Provides persistent block storage for virtual machines.
- Manages the attachment/detachment of volumes during instance migration.
- Support volume migration to different storage backends if required.
- Snapshot and backup functionalities.
- Integration with various storage providers.
- Volume resizing and multi-attach support.

Swift

- Provides scalable object storage for unstructured data like images, backups, and logs.
- Used for migrating objects such as VM images to different storage locations.
- Can serve as backend for image and backup storage during migration.
- Highly available and fault-tolerant.
- Supports object replication across clusters.
- Integrates with Glance for VM image storage.

Glances

- Manages VM images for deployment in OpenStack.
- Stores and retrieves VM images required during instance.
- Ensures compatibility of images on different compute hosts.
- Supports multiple image formats (RAW, VHD, QCOW2).
- Image versioning and sharing across projects.
- Integration with storage backends like Swift and Cinder.

CI / CD

- Purpose: CI/CD automates the software development process, enabling rapid and reliable deployment of code changes.
- Continuous Integration (CI):
  - Developers regularly merge code changes into a shared repository.
  - Automated tests and builds are triggered to detect errors early.
  - Ensures code is always in a deployable state.
- Continuous Delivery (CD):
  - Automates the release process, ensuring code is ready to deploy to production at any time.
  - Involves staging environments where further tests are run.
  - Ensures fast, consistent, and reliable delivery of new features and bug fixes.
- Key Benefits:
  - Faster development cycles with minimal manual intervention.
  - Reduced risk of bugs in production due to frequent testing.
  - Improved collaboration and transparency among teams
- Popular CI/CD Tools: Jenkins, GitLab CI, CircleCI, Travis CI, AWS CodePipeline.

VM Migration

The movement of VMs from one resource to another, such as from one physical host to another physical host, or data store to data store, is known as VM migration.
Virtual Machine Migration refers to the process of relocating virtual machines from one physical server to another to achieve load balancing and power saving, often facilitated by Software-Defined Networking (SDN) technologies.
Virtual Machine (VM) migration is a promising way to ensure load balancing and power saving by VM reallocation. However, VM migration is a difficult task because it requires updates of the network state; this may lead to inconsistency and violations of service level agreement.
There are two types of VM migration:

- Cold migration occurs when the VM is shut down.
- Live migration occurs while the VM is actually running.

Considerations for VM Migration

1. Compatibility: Ensure compatibility between source and destination environments, especially when migrating between different hypervisors or cloud platforms.
2. Downtime and Performance: Plan for potential downtime, especially with cold migrations and assess the impact on performance during live migrations.
3. Data Integrity: Maintain data integrity during the migration process, ensuring that no data is lost or corrupted.
4. Network Requirements: Ensure sufficient network bandwidth and low latency to handle live migrations, particularly when dealing with large VMs or significant amounts of data.
5. Security: Implement encryption and secure transfer protocol to protect data during migration, especially when migrating VMs over public networks or to the cloud.
6. Compliance: Ensure that the migration process complies with relevant regulations and industry standards, particularly concerning data sovereignty and privacy.
7. Testing and Validation: Conduct through testing and Validation before and after migration to ensure that VMs operate correctly in the new environment.

Tools and Solutions

1. VMWare vSphere/vMotion Industry-standard tools for live migration and other VM Management tasks.
2. Microsoft Azure Migrate Comprehensive Solution for assessing migrating and optimizing workloads to Azure.
3. AWS Server Migration Service (SMS): Automates the migration of on-premises VMs to AWS.

4. Google Clod Migrate: Helps in migrating workloads from various environments to Google Cloud.
5. Red Hat Virtualization (RHV): Facilitates the migration of VMs between different Red Hat Environments or to clod platforms.
6. Third-Party Tools Solutions like CloudEndure, Zerto and Carbonite offer additional flexibility and features for VM Migration.

- Migrating virtual machines is a complex but manageable process that requires careful planning and the right techniques to ensure success.
- By understanding the different types of migration, the techniques available and the considerations involved, organizations can effectively move their workloads to new environments with minimal disruption and maximum benefit.

Fault Tolerance Mechanisms

Fault tolerance mechanisms are crucial for ensuring the systems continue to operate correctly even when some components fail. These mechanisms are designed to detect, isolate, and recover from failures, minimizing disruption and maintaining the availability, reliability, and integrity of a system.

Redundancy:

Hardware Redundancy:

- Definition: Involves duplicating critical components or systems to ensure that a backup is available if the primary system fails.
- Methods:
  - N+1 Redundancy: For every N critical components, there is one backup.
  - 2N Redundancy: Full duplication of systems, where each component has a backup.
  - RAID (Redundant Array of Independent Disks): A Method of storing the same data in different places to protect against disk failures.
- N+1 Redundancy provides a minimal level of resiliency by adding a single component like a UPS, HVAC system or generator to N architecture to support a failure or allow a single machine to be serviced. When one system is offline, the extra component takes over its load.
- A 2N redundancy model creates a mirror image of the original UPS, cooling system or generator to provide full tolerance. This means if four UPS Units are necessary to satisfy capacity requirements, the redundant architecture would include an additional four units of UPS, for a total of eight systems. This design also utilizes two independent distribution systems.
  This architecture allows the data center operator to take down an entire set of components

for maintenance without interrupting normal operations. Further, in the event that the primary architecture fails, the secondary architecture takes over to maintain service. The resiliency of this architecture greatly diminishes the likelihood of downtime.

- 2N+1 delivers the fully fault-tolerant 2N architecture plus an extra component for an added layer of protection. Not only can this architecture withstand multiple component failures, even in a worst case scenario when the entire primary system goes down, it can sustain N+1 redundancy.
  This level of redundancy is generally used by large companies which cannot tolerate even minor service disruptions.

Software Redundancy:

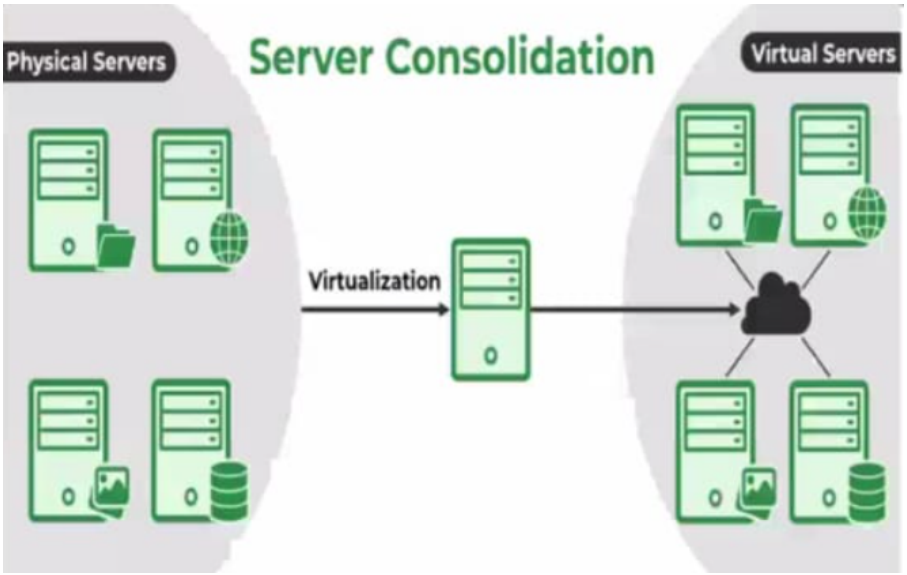
- Implementing redundant software processes or tasks to ensure continued operation if one process fails.
- Software structure and actions are modified to be able to detect a fault, isolate it and prevent the propagation of its effect throughout the system.
- Methods:
  - Primary-Secondary (Active-Passive): The secondary process takes over if the primary process fails.
  - Active-Active: Both processes run simultaneously, when load balancing between them.
- Checkpointing: This involves periodically saving the state of the software system so that if a failure occurs, the system can be restored to a previous state.
- Error Detection and Correction: Error detection and correction algorithms can be used to detect and correct errors in data transmission.
- Failure Prediction: This involves using algorithms or heuristics to predict when a failure is likely to occur so that system can take appropriate action to prevent or mitigate the failure.
- Load Balancing: Distributing workloads across multiple components so that no single component is overburdened. This can help to prevent failures and improve the overall performance of the system.
- Autonomous Systems: Autonomous systems are made to identify, diagnose and fix errors on their own without need for human assistance using automatic fault isolation, recovery, and identification procedures.
- Isolation and Restrictions: Isolation and containment is necessary to build systems so that errors in one component do not spread to the rest of the system. This can involve dividing up components and minimizing the effect of errors through the use of virtualization, microservices or containers.
- Replication: The practice of making multiple copies of essential system components or services and distributing them. Their design is fault-tolerant and able to function continuously even in the event of a failure.

- Dynamic reconfiguration: This technique allows a system to dynamically respond to faults, reallocate resources and adapt to changing conditions. By modifying the configurations of the system in real time according to the operational conditions, this technique improves system resilience.

Server Consolidation

- Server consolidation in cloud computing refers to the process of combining multiple servers into a single, more powerful server or cluster of servers.
- This can be done in order to improve the efficiency and cost-effectiveness of the cloud computing environment.
- Server consolidation is typically achieved through the use of virtualization technology, which allows multiple virtual servers to run on a single physical server.
- This allows for better utilization of resources, as well as improved scalability and flexibility.
- It also allows organizations to reduce the number of physical servers they need to maintain, which can lead to cost savings on hardware, power and cooling.
- Multiple physical servers are consolidated into a fewer number of powerful serves using virtualization.
- This process results in the creation of logical servers which are isolated from one another and have their own operating systems and applications.
- These logical servers share the same physical resources such as CPU, RAM and storage.

Architecture



- Server consolidation creates virtual servers that share the resources of the physical servers by fusing a number of physical servers into a single virtualized environment utilizing virtualization software.
- This makes it possible to use the resources more effectively and save money.
- Additionally, it makes it simple to manage existing servers, set up new ones, and scale resources up or down as necessary.
- Three Primary parts of server consolidation architecture:

- Physical Servers: The server consolidation environment's hardware consists of physical servers. These servers are usually powerful machines with high processing speeds that are built to manage massive volumes of data. They are utilized to run virtual servers and host virtualization software.
- Virtualization: A single physical server can run several virtual servers due to virtualization. Multiple virtual servers can share the resources of a single physical server with the help of the virtualization software's creation of an abstraction layer between the real hardware and virtual servers.
- Virtual Servers: Physical servers are virtualized into virtual servers. They run on the top of the physical servers and are produced and controlled by the virtualization software. Each virtual server can execute its own programs and services and is a separate instance of an operating system.

Steps in Server Consolidation

1. Assessing the current Environment: The first step in server consolidation is to assess the current environment to determine which servers are running similar workloads and which ones are underutilized or over-utilized. This can be done by analyzing the usage patterns and resource utilization of each server.
2. Identifying and Grouping Servers: Once the current environment has been assessed, the next step is to identify and group servers based on their workloads. This can help to identify servers that are running similar workloads and can be consolidated onto fewer, more powerful servers or clusters.
3. Planning the Consolidation: After identifying and grouping servers, the next step is to plan the consolidation. This involves determining the best way to consolidate the servers, such as using virtualization technology, cloud management platforms, or physical consolidation. It also involves determining the resources required to support the consolidated servers, such as CPU, RAM, and storage.
4. Testing and Validation: Before consolidating the servers, it is important to test and validate the consolidation plan to ensure that it will meet the organization's needs and that the servers will continue to function as expected.
5. Consolidating the Servers: Once the plan has been tested and validated, the servers can be consolidated. This typically involves shutting down the servers to be consolidated, migrating their workloads to the consolidated servers, and then bringing the servers back online.
6. Monitoring and Maintenance: After the servers have been consolidated, it is important to monitor the consolidated servers to ensure that they are performing as expected and to identify any potential issues. Regularly maintenance should also be performed to keep the servers running smoothly.

7. Optimizing the Consolidated Environment: To keep the consolidated environment optimal, it's important to regularly evaluate the usage patterns and resources utilization of the consolidated servers, and make adjustments as needed.

Types

Rationalized Consolidation

- It is a type of server consolidation in which multiple servers are consolidated based on their workloads.
- This process involves identifying and grouping servers based on the applications and services they are running and then consolidating them onto fewer, more powerful servers or clusters.
- The goal is rationalized consolidation is to improve the efficiency and cost-effectiveness of the cloud computing environment by consolidating servers that are running similar workloads.

Physical Consolidation

- It is a type of server consolidation in which multiple physical servers are consolidated into a single, more powerful server or cluster of servers.
- This can be done by replacing multiple older servers with newer, more powerful servers, or by adding additional resources such as memory and storage to existing servers.
- Physical consolidation can help organizations to improve the performance and efficiency of their cloud computing environment.

Logical Consolidation

- In logical server consolidation, multiple virtual servers are consolidated onto a single physical server.
- Each virtual server is isolated from the others and has its own operating systems and applications, but shares the same physical resources such as CPU, RAM, and storage.
- This allows organization to run multiple virtual servers on a single physical server, which can lead to significant cost savings and improved performance.
- Virtual servers can be easily added or removed as needed, which allows organizations to more easily adjust to changing business needs.
-

Benefits

- Cost Savings: By consolidating serves, organizations can reduce the number of physical serves they need to maintain, which can lead to cost savings on hardware, power and cooling.
- Improved Performance: Consolidating serves can also improve the performance of cloud computing environment. By using virtualization technology, multiple virtual serves can run on a single physical server, which allows for better utilization of resources. This can lead to faster processing times and better overall performance.
- Scalability and flexibility: Server consolidation can also improve the scalability and flexibility of the cloud environment. By using virtualization technology, organizations can easily add or virtual servers as needed, which allows them to more easily adjust to changing business needs.
- Management simplicity: Managing multiple servers can be complex and time-consuming. Consolidating servers can help to reduce the complexity of managing multiple servers, by providing a single point of management. This can help organizations to reduce the effort and cost associated with managing multiple servers.
- Better utilization of resources: By consolidating servers, organizations can improve the utilization of resources, which can lead to better performance and cost savings.

Cloud Provisioning

- Cloud provisioning refers to the processes involved in deploying and integrating cloud computing services into an enterprise's IT Infrastructure.
- It encompasses the establishment of policies, procedures, and objectives for souring cloud services from a provider.
- From provider's perspective, cloud provisioning includes supplying and assigning the necessary resources, such as virtual machines, storage, and network access.
- Cloud Provisioning model allows organizations to efficiently scale their IT infrastructure without requiring substantial upfront investments in hardware.
- ***Cloud Provisioning is the dynamic allocation and configuration of computing resources, facilitating the seamless deployment of applications and services in the cloud to meet evolving business needs.***
- Provisioning in cloud computing involves allocating and configuring IT resources to meet the dynamic needs of an organization. This includes configuring components like OS, middleware, and applications.
- It also ensures seamless access to necessary resources and includes security measures such as firewalls, threat detection and encryption.
- Some challenges of cloud provisioning include issues with service enforcement, service and results dependencies, cost controls, and complex management.
- Organizations can address these challenges through proper resources allocation, storage and network configuration, managing engine images, and monitoring and maintenance.

Types

Manual Provisioning

This conventional provisioning method involves hands-on allocation and configuration by IT administrators. Although it provides a high level of control, it can be time-intensive and less adaptive to dynamic workload changes. Use Cases: Well-suited for static workloads with predictable resource demands.

Automated Provisioning

Utilizing scripts or tools, automated provisioning minimizes human intervention, expediting the deployment process and enhancing responsiveness to evolving demands. Use cases: Ideal for environments characterized by varying workloads, necessitating swift and efficient resource allocation.

Dynamic Provisioning

Also known as On-demand provisioning, it stands out as most flexible and scalable cloud computing model. It empowers cloud service providers to allocate resources dynamically enabling client organizations to swiftly acquire IT resources without manual adjustments. Cloud automation and orchestration streamline this process, catering to diverse customer needs.
It also enables the balancing of computing demands in one part of the grid with the availability of resources in another part. This is done by using policies to control and balance the allocation of resources.

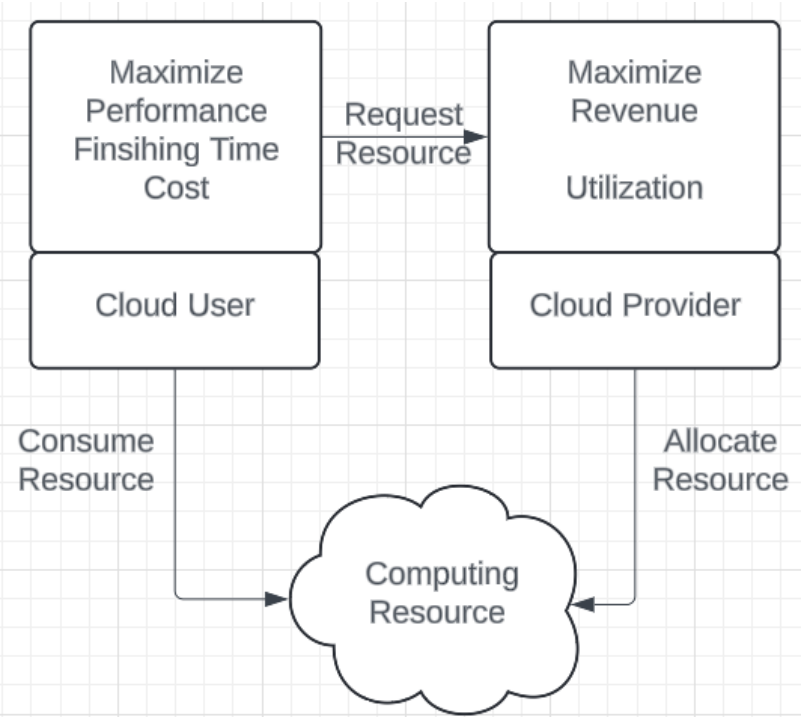
User Self-Provisioning

Termed as cloud self-service, it allows customers to directly subscribe to required resources from the cloud provider via a website. Users create an account and pay for the needed resources. Use cases: Ideal for organizations emphasizing autonomy and agility, offering a straightforward subscription process without complex procurement or onboarding procedures with cloud vendor.

Resource Management

- In cloud computing, resource management refers to process of managing the usage of resources in a cloud environment. This includes compute resources (like CPU and memory), storage resources (like disk space), and network resources (like bandwidth).
- Resource Management is crucial for optimizing the performance and cost of a cloud environment. It involves monitoring resource usage, allocating resources, and managing resource capacity.

Objectives



Cloud Vendor

- Provision resources on and on-demand basis.
- Energy conservation and proper utilization is maintained in Cloud Data Centers.

Cloud Service Provider

- To make available the best performance resources at the cheapest cost.
- QoS {Quality of service} to their cloud users.

Cloud User

- Renting resources at a low price without compromising performance.
- Cloud Providers guarantees to provide a minimum level of service to the user.

Types

Compute Resources

- Collection of Physical Machines
  - Processors, Memory, Network Interface, I/O
  - Provides the computational capacity of cloud

Network Resources

- Within a data center large number of compute resources are clustered and interconnected using high bandwidth network

Power Resource

- Data centers consume large amount of power
- Creating and reducing power consumption is the focus.

Resource Management Models

Compute Model

- Resources in the cloud is shared by all users at the same time.
- It allows the users to reserve the VM's memory to ensure that the memory size requested by the VM is always available to operate locally on clouds with a good enough level of QoS being delivered to the end user.
- Grid Strictly manages the workload of computing mode.
- Local resource manager such as Portable Batch System, Codor, And Sun Grid Engine manages the compute resources for the Grid site.

Data Model

- It is related to plotting, separating, querying, transferring, caching, and replicating data.
- Data stored at an un-trusted host: Although may not seem the best policy to store data and let others use the data without permission moving data off-premises increases the number of potential security risks.
- Data Replication over Large Areas: Making sure data is available whenever demanded is of utmost importance for cloud storage providers. Data availability is typically achieved through under-the-covers replication, i.e. data is automatically replicated without customer interference or requests.

Programming Model

- User-level programming languages are used for accessing and operating the cloud.
- In Cloud: Makes use of Web Services where users have more control over the Cloud Services. The translation of data for the receiving system and real-time data exchange between systems without middleware of all the services and applications remain a big challenge.
- In Grid: Makes use of parallel and distributed computing environment
- Challenges:
    - Multiple service providers allow to access data to clients with little authorization or authentication.
    - Diversity in resources in turn affects the performance and stability.
    - Error handling in a continuously changing business environment

Security Model

- Allows users to control the security of their own data by maintaining passwords, and receiving any news regarding suspicious activity with their data via email.
- Risks in Security Model:
    1. Privileged use access
    2. Regulatory compliance
    3. Data location
    4. Data partition
    5. Recovery
    6. Investigation support
    7. Long-term durability

Scheduling Techniques

The task scheduling system in cloud computing passes through three levels.

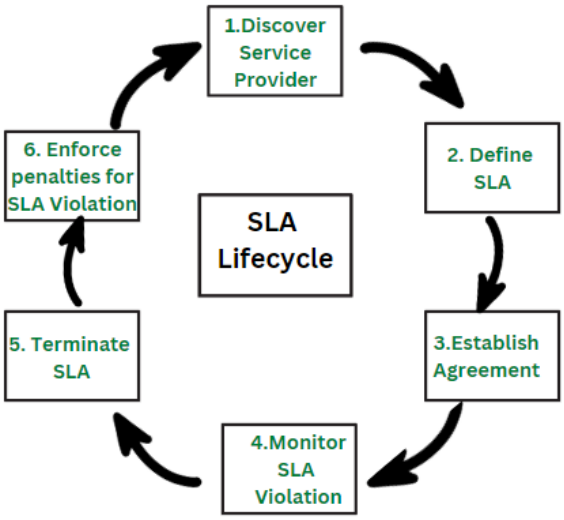
- The first task level: is a set of tasks (Cloudlets) that are sent by cloud users, which are required for execution.
- The second scheduling level: is responsible for mapping tasks to suitable resources to get the highest resource utilization with minimum make span. The make span is the overall completion time for all tasks from the beginning to the end.
- The third VMS level: is a set of (VMs) which are used to execute the tasks.

Capacity Management to meet SLA Requirements

Capacity management to meet SLA requirements refers to the practice of strategically managing the available resources within a system to ensure that the service level agreements (SLAs) set with customers are consistently met, by accurately forecasting demand, optimizing resource allocation, and taking proactive measures to avoid service disruptions and maintain desired performance levels.

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



1. **Discover service provider:** This step involves identifying a service provider that can meet the needs of the organization and has the capability to provide the required service. This can be done through research, requesting proposals, or reaching out to vendors.
2. **Define SLA:** In this step, the service level requirements are defined and agreed upon between the service provider and the organization. This includes defining the service level objectives, metrics, and targets that will be used to measure the performance of the service provider.
3. **Establish Agreement:** After the service level requirements have been defined, an agreement is established between the organization and the service provider outlining the terms and conditions of the service. This agreement should include the SLA, any penalties for non-compliance, and the process for monitoring and reporting on the service level objectives.
4. **Monitor SLA violation:** This step involves regularly monitoring the service level objectives to ensure that the service provider is meeting their commitments. If any violations are identified, they should be reported and addressed in a timely manner.
5. **Terminate SLA:** If the service provider is unable to meet the service level objectives, or if the organization is not satisfied with the service provided, the SLA can be terminated. This can be done through mutual agreement or through the enforcement of penalties for non-compliance.
6. **Enforce penalties for SLA Violation:** If the service provider is found to be in violation of the SLA, penalties can be imposed as outlined in the agreement. These penalties can include financial penalties, reduced service level objectives, or termination of the agreement.

Load Balancing

Load balancing is a method of distributing network traffic evenly across multiple servers to ensure optimal performance and resource utilization by directing incoming requests to the most available server, while "load balancing techniques" refer to the different algorithms used by a load balancer to decide which server to send a request to, such as Round Robin, Least Connections, Least Response Time, and Weighted Round Robin, each with its own strengths and weaknesses depending on the situation.
Load balancing is the method of distributing network traffic equally across a pool of resources that support an application. Modern applications must process millions of users simultaneously and return the correct text, videos, images, and other data to each user in a fast and reliable manner. To handle such high volumes of traffic, most applications have many resource servers with duplicate data between them. A load balancer is a device that sits between the user and the server group and acts as an invisible facilitator, ensuring that all resource servers are used equally.

Load Balancing Algorithms

Static load balancing

Static load balancing algorithms follow fixed rules and are independent of the current server state. The following are examples of static load balancing.

1. Round-robin method:
   Servers have IP addresses that tell the client where to send requests. The IP address is a long number that is difficult to remember. To make it easy, a Domain Name System maps website names to servers. When you enter aws.amazon.com into your browser, the request first goes to our name server, which returns our IP address to your browser.
   In the round-robin method, an authoritative name server does the load balancing instead of specialized hardware or software. The name server returns the IP addresses of different servers in the server farm turn by turn or in a round-robin fashion.
2. Weighted round-robin method:
   In weighted round-robin load balancing, you can assign different weights to each server based on their priority or capacity. Servers with higher weights will receive more incoming application traffic from the name server.
3. IP hash Method:
   In the IP hash method, the load balancer performs a mathematical computation, called hashing, on the client IP address. It converts the client IP address to a number, which is then mapped to individual servers.

Dynamic load balancing

Dynamic load balancing algorithms examine the current state of the servers before distributing traffic. The following are some examples of dynamic load balancing algorithms.

1. Least connection method:
   A connection is an open communication channel between a client and a server. When the client sends the first request to the server, they authenticate and establish an active connection between each other. In the least connection method, the load balancer checks which servers have the fewest active connections and sends traffic to those servers. This method assumes that all connections require equal processing power for all servers.
2. Weighted least connection method:
   Weighted least connection algorithms assume that some servers can handle more active connections than others. Therefore, you can assign different weights or capacities to each server, and the load balancer sends the new client requests to the server with the least connections by capacity.
3. Least response time method:
   The response time is the total time that the server takes to process the incoming requests and send a response. The least response time method combines the server response time and the active connections to determine the best server. Load balancers use this algorithm to ensure faster service for all users.
4. Resource-based method:
   In the resource-based method, load balancers distribute traffic by analyzing the current server load. Specialized software called an agent runs on each server and calculates usage of server resources, such as its computing capacity and memory. Then, the load balancer checks the agent for sufficient free resources before distributing traffic to that server.

Load Balancing Technology

Hardware Load Balancers

A hardware-based load balancer is a hardware appliance that can securely process and redirect gigabytes of traffic to hundreds of different servers. You can store it in your data centers and use virtualization to create multiple digital or virtual load balancers that you can centrally manage.

Software Load Balancers

A hardware-based load balancer is a hardware appliance that can securely process and redirect gigabytes of traffic to hundreds of different servers. You can store it in your data centers and use virtualization to create multiple digital or virtual load balancers that you can centrally manage.

Comparison of hardware balancers to software load balancers

Hardware load balancers require an initial investment, configuration, and ongoing maintenance. You might also not use them to full capacity, especially if you purchase one only to handle peak-time traffic spikes. If traffic volume increases suddenly beyond its current capacity, this will affect users until you can purchase and set up another load balancer.

In contrast, software-based load balancers are much more flexible. They can scale up or down easily and are more compatible with modern cloud computing environments. They also cost less to set up, manage, and use over time.