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

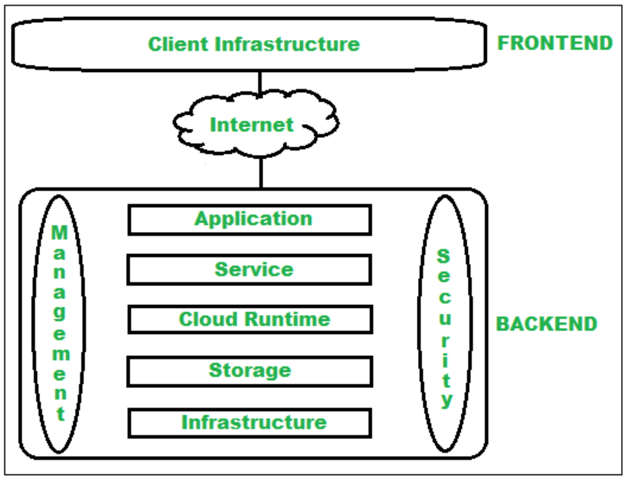
Cloud Computing: It refers to manipulating, configuring and accessing the hardware and software resources over the internet. It offers online data storage, infrastructure and application. It offers platform independency.

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.

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 powerful system.	Variable; depends on the number and type of nodes used.	Moderate; can use commodity hardware but requires good interconnect.	Low to moderate; uses existing resources but needs complex management.

Aspect	Centralized Computing	Distributed Computing	Cluster Computing	Grid Computing
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.

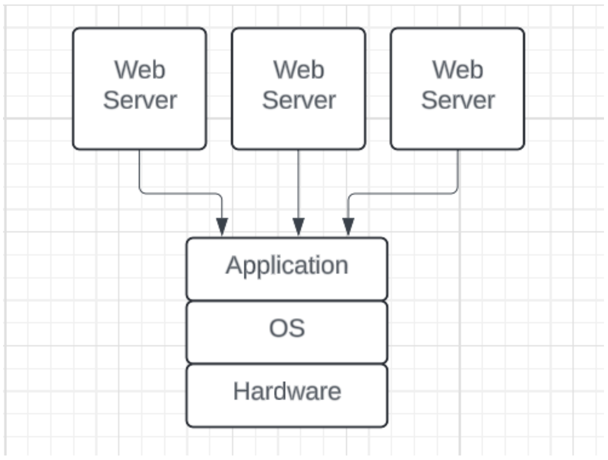
Components:

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

Types of Hosting

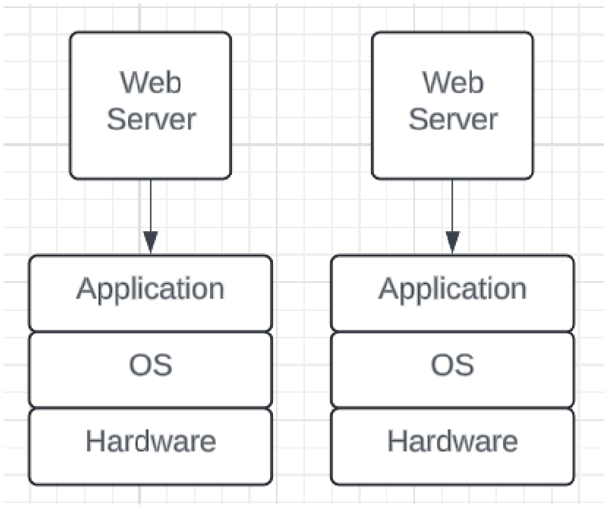
Shared Hosting

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.



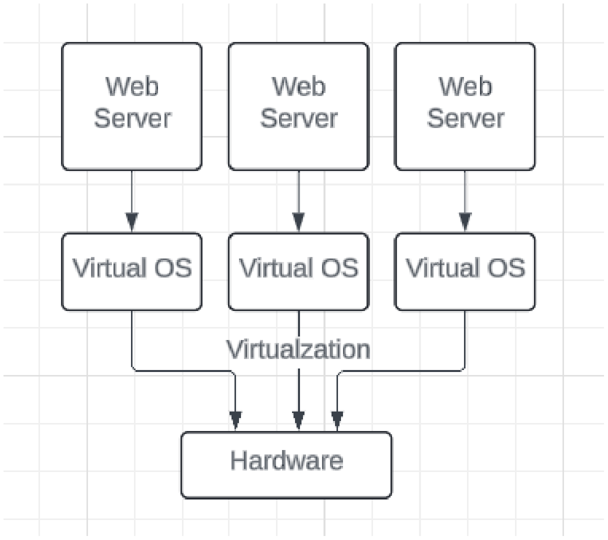
Dedicated Hosting

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.



Comparison of Shared, VPS, Dedicated, and Cloud Hosting

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 be adjusted dynamically.

Aspect	Shared Hosting	VPS Hosting	Dedicated Hosting	Cloud Hosting
	migration for growth.	higher resource plans.	server for upgrades.	
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

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.

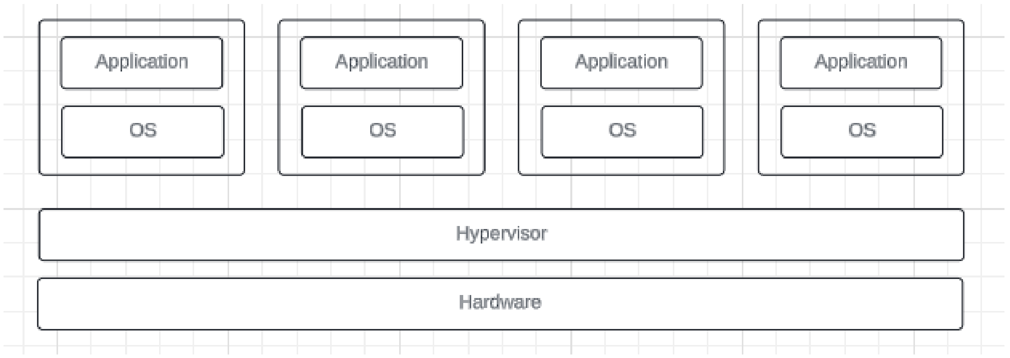
Hypervisor

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*.

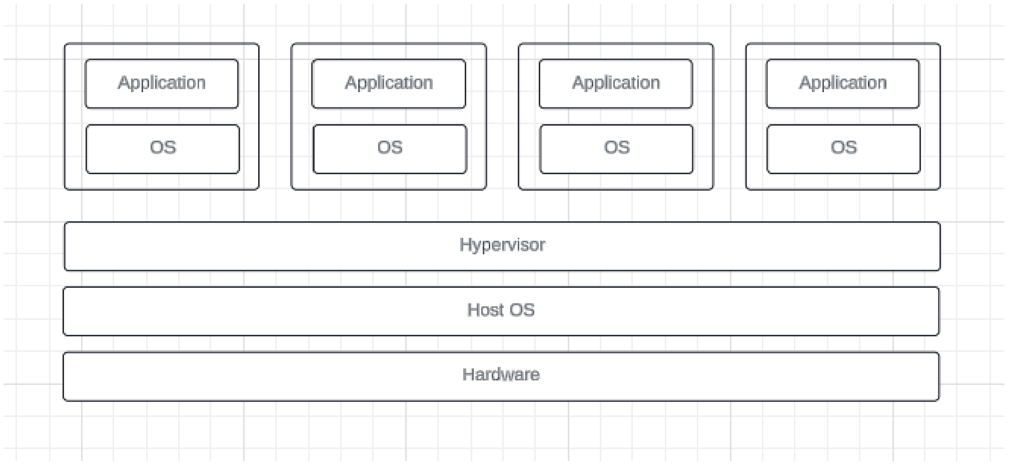
Resource Allocation: The hypervisor ensures that each virtual machine receives the allocated resources as configured. It does so by acting as an intermediary between guest machines and the underlying physical hardware. The hypervisor relays requests for processing power, memory, storage, and other resources 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.

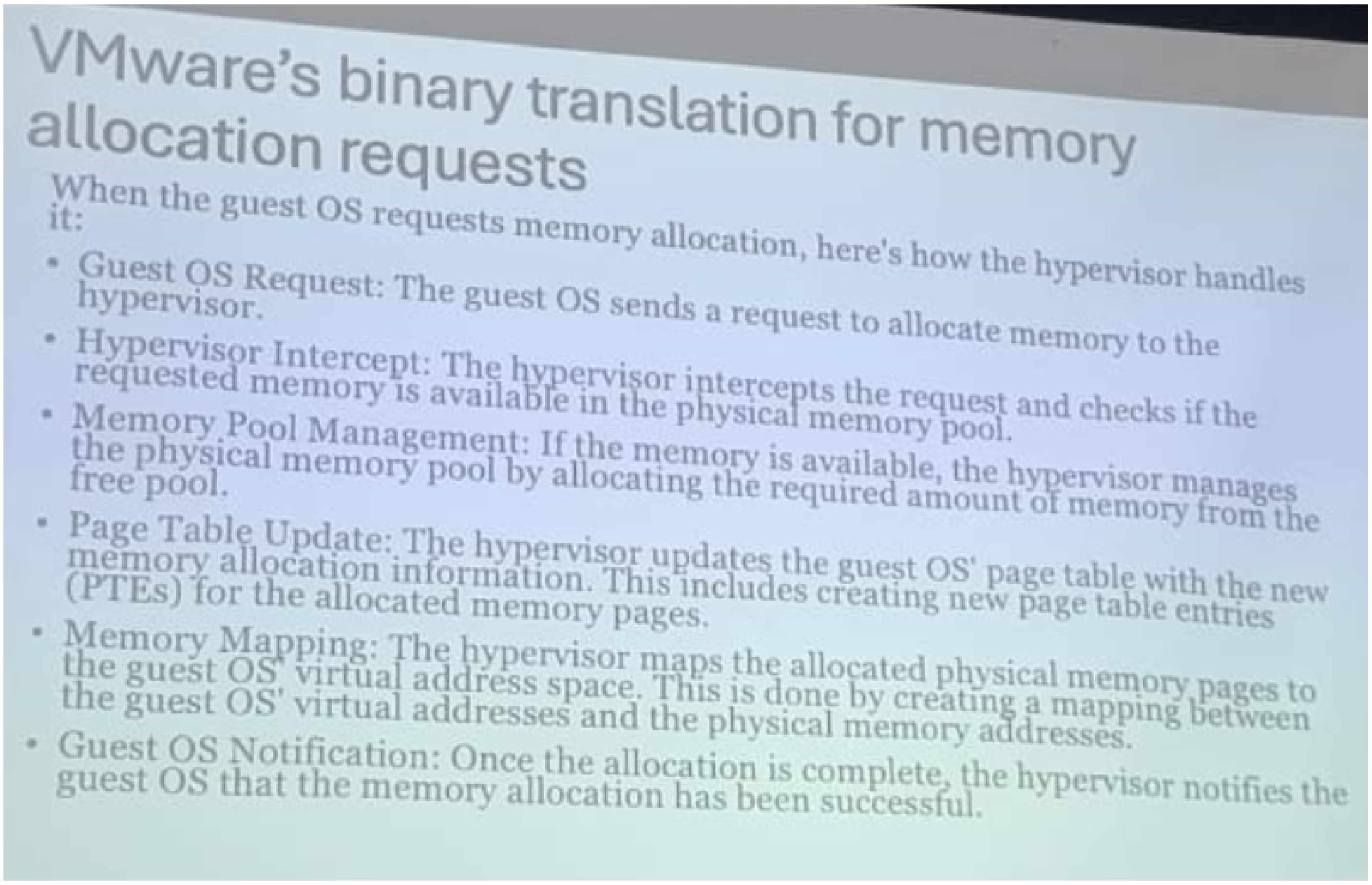
Type 2 Hypervisor



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.



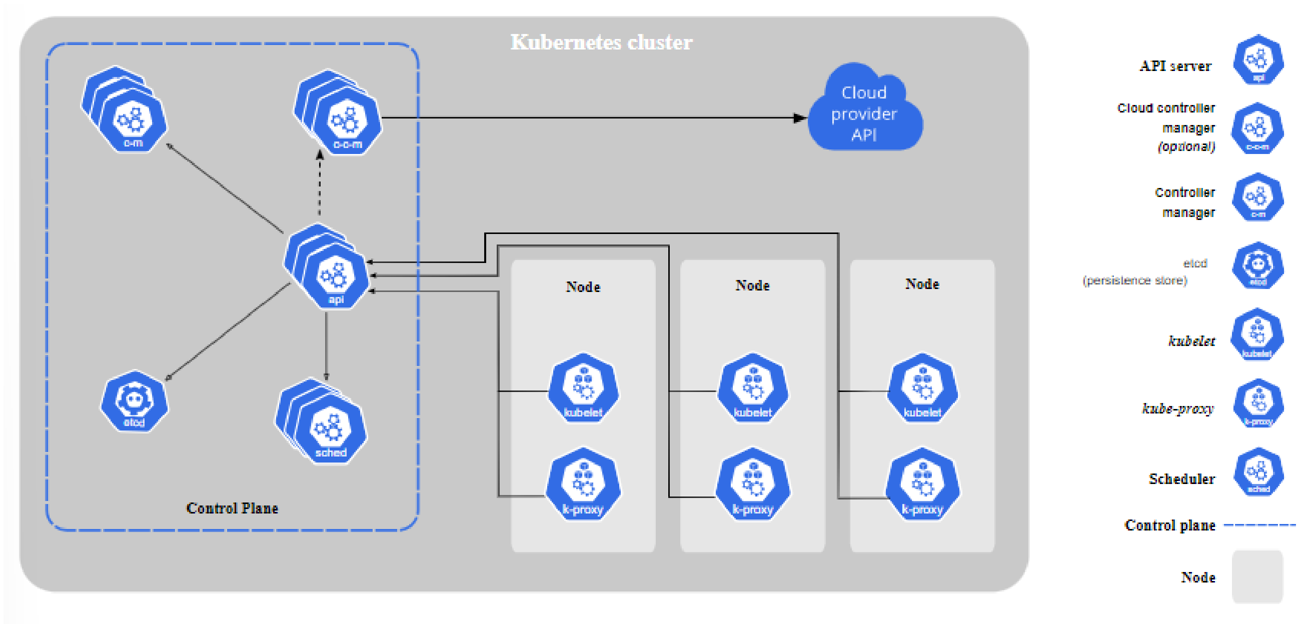
VMware's binary translation for memory allocation requests

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

- Guest OS Request: The guest OS sends a request to allocate 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 the 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.

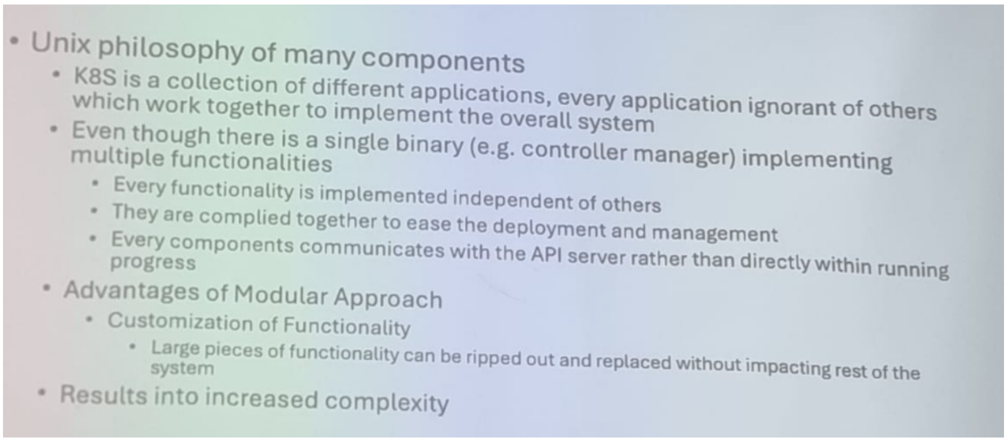
Kubernetes

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.



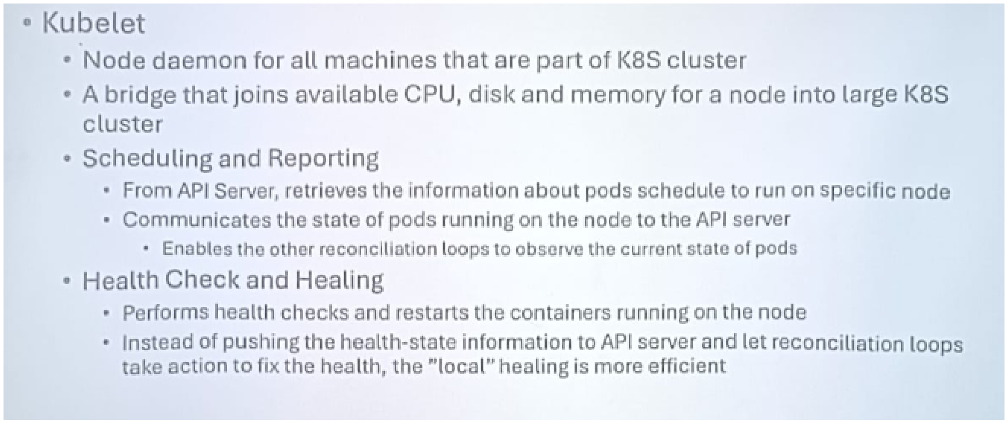
- Unix philosophy of many components
  - K8S is a collection of different applications, every application ignorant of others which work together to implement the overall system
  - Even though there is a single binary (e.g. controller manager) implementing multiple functionalities
    - Every functionality is implemented independent of others
    - They are complied 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



- Kubelet
  - Node daemon for all machines that are part of K8S cluster
  - A bridge that joins available CPU, disk and memory for a node into 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 Check 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

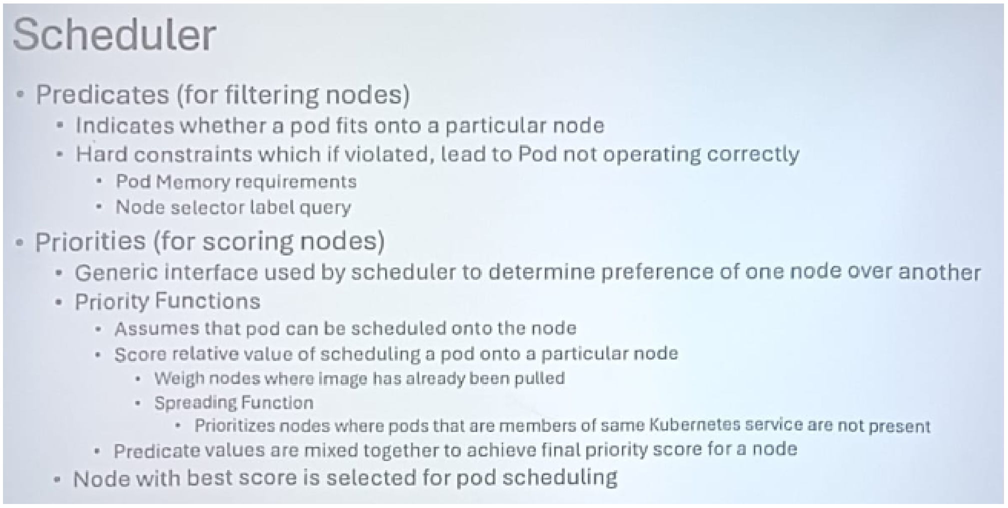
API Translation

- 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

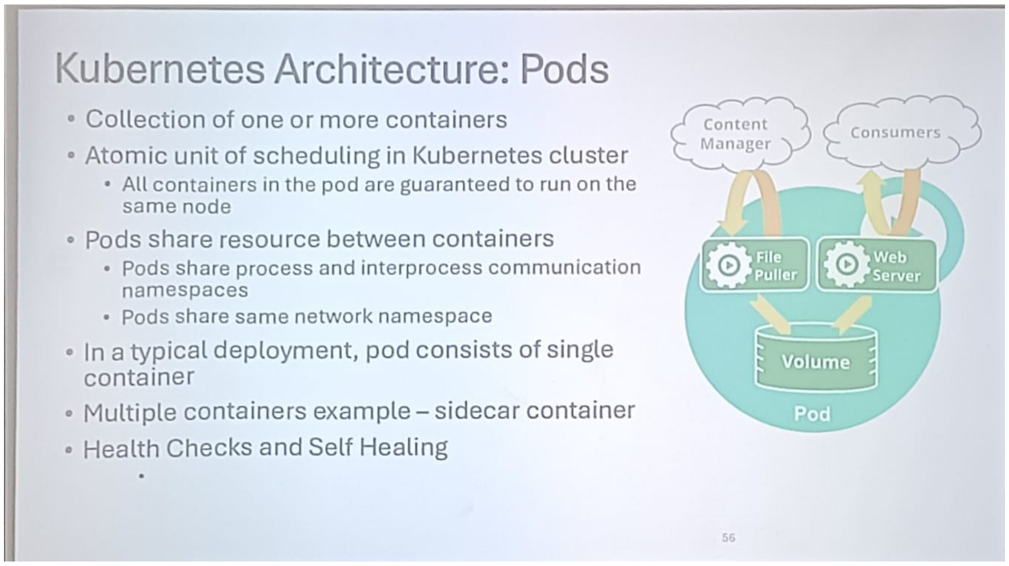
etcd

- 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



Kubernetes Architecture: Pods

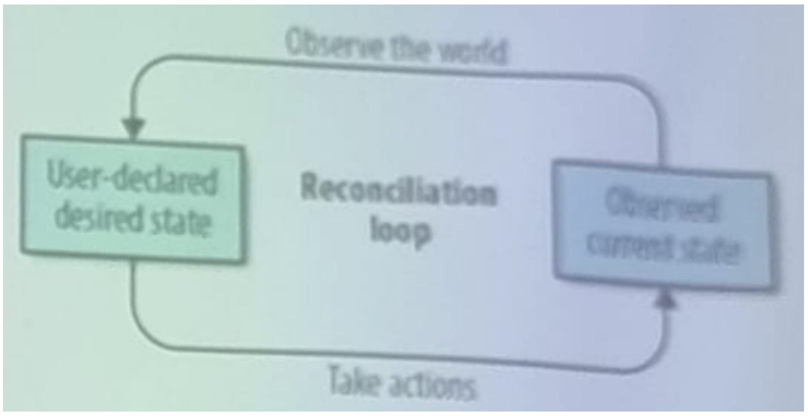
- Collection of one or more containers
- Atomic unit of scheduling in Kubernetes cluster
    - All containers in the pod are guaranteed to run on the same node
- Pods share resource between containers
    - Pods share process and interprocess communication namespaces
    - Pods share same network namespace
- In a typical deployment, pod consists of single container
- Multiple containers example – sidecar container
- Health Checks and Self Healing
    -

56

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.

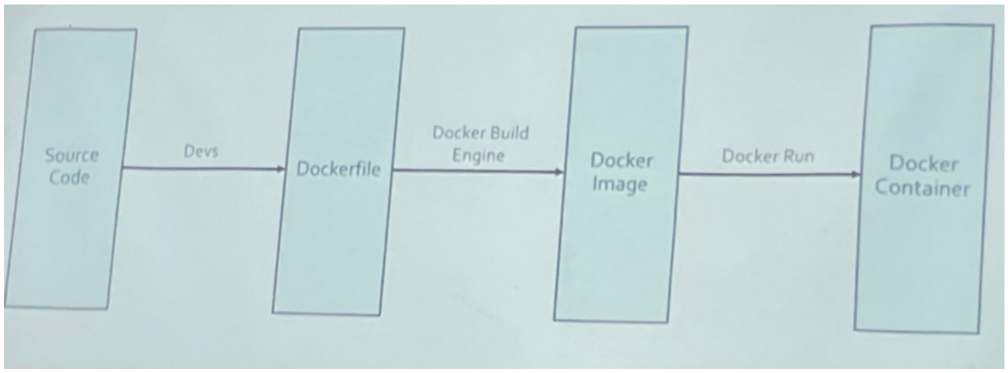
Dynamic Grouping

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 seletors

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.

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:

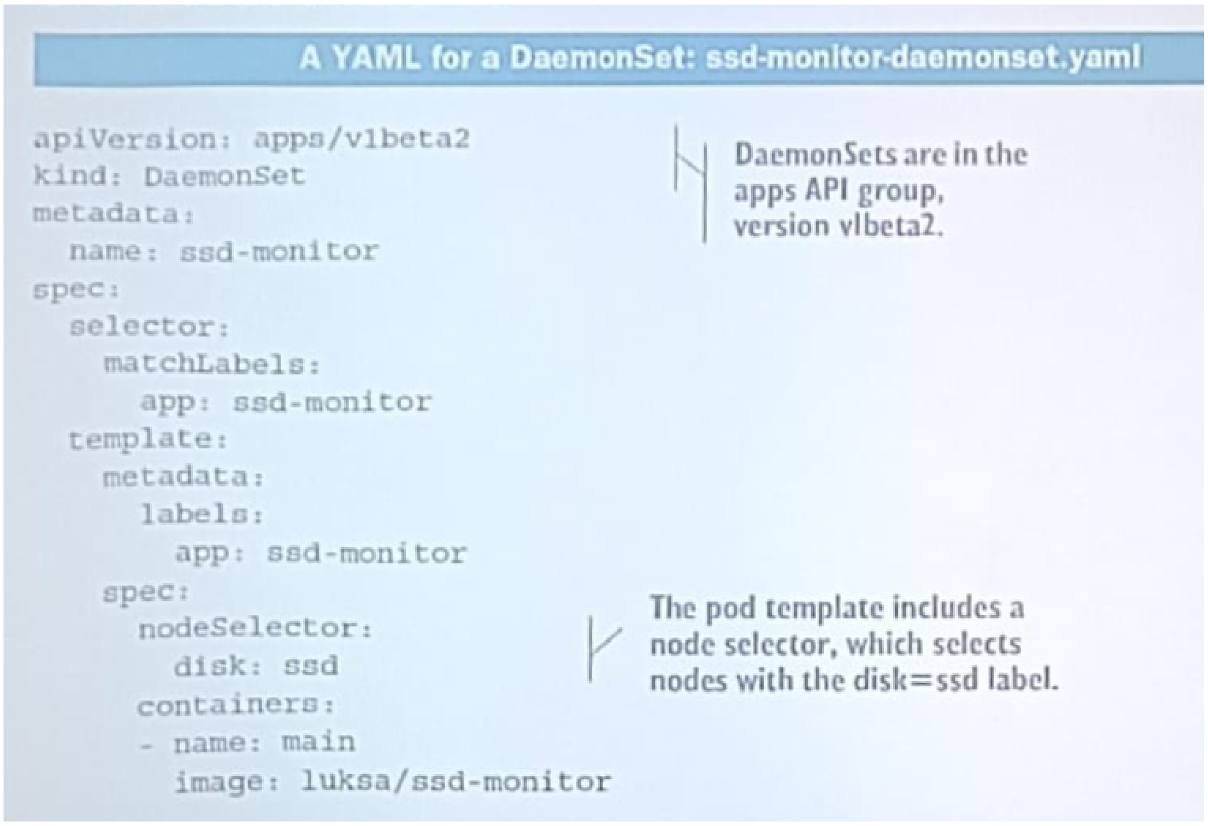
- Portability

- Speed
- Scalability
- Fault Isolation
- Ease of Management
- Developer Friendly

YAML Files

DaemonSets

- Continues running
- 1 pod / node
- Bypass Scheduler



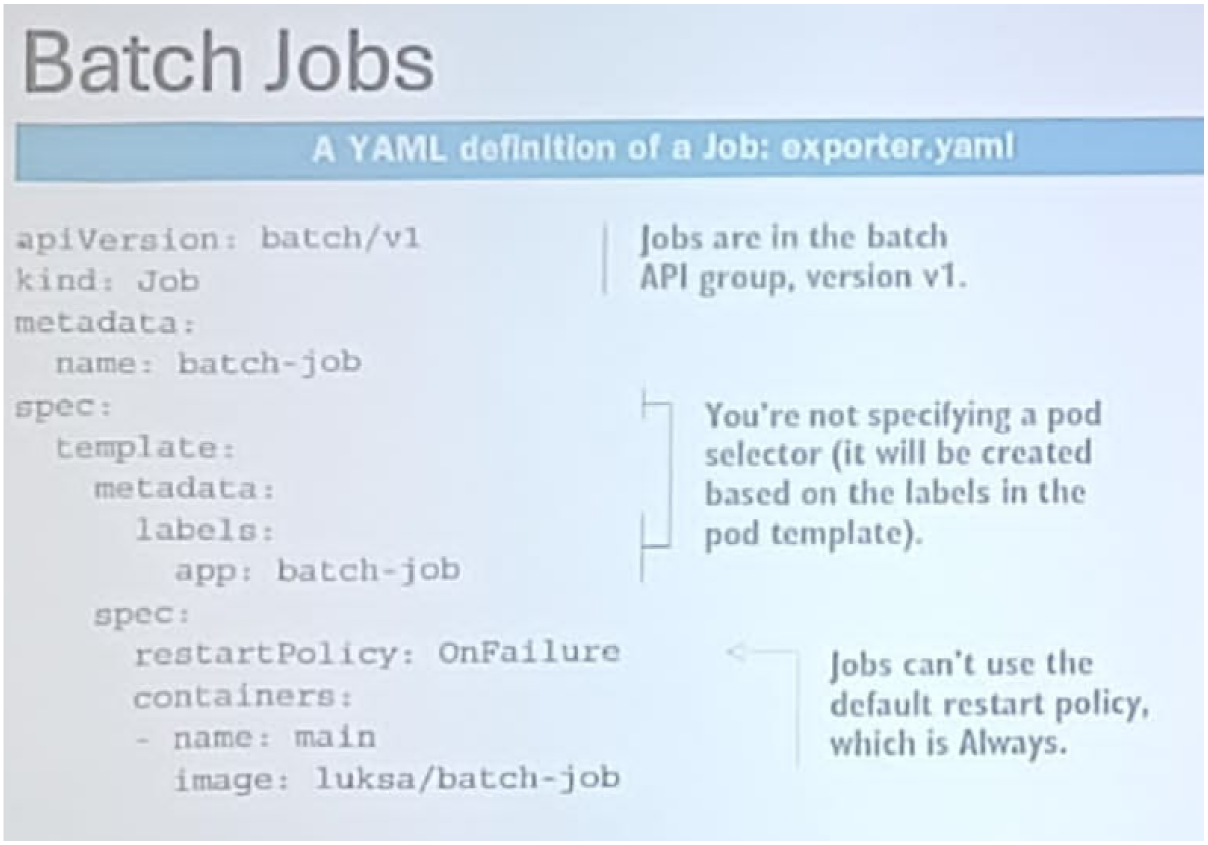
A YAML for a DaemonSet: ssd-monitor-daemonset.yaml

```
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



Batch Jobs

A YAML definition of a Job: exporter.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: batch-job
spec:
  template:
    metadata:
      labels:
        app: batch-job
    spec:
      restartPolicy: OnFailure
      containers:
      - name: main
        image: luksa/batch-job
```

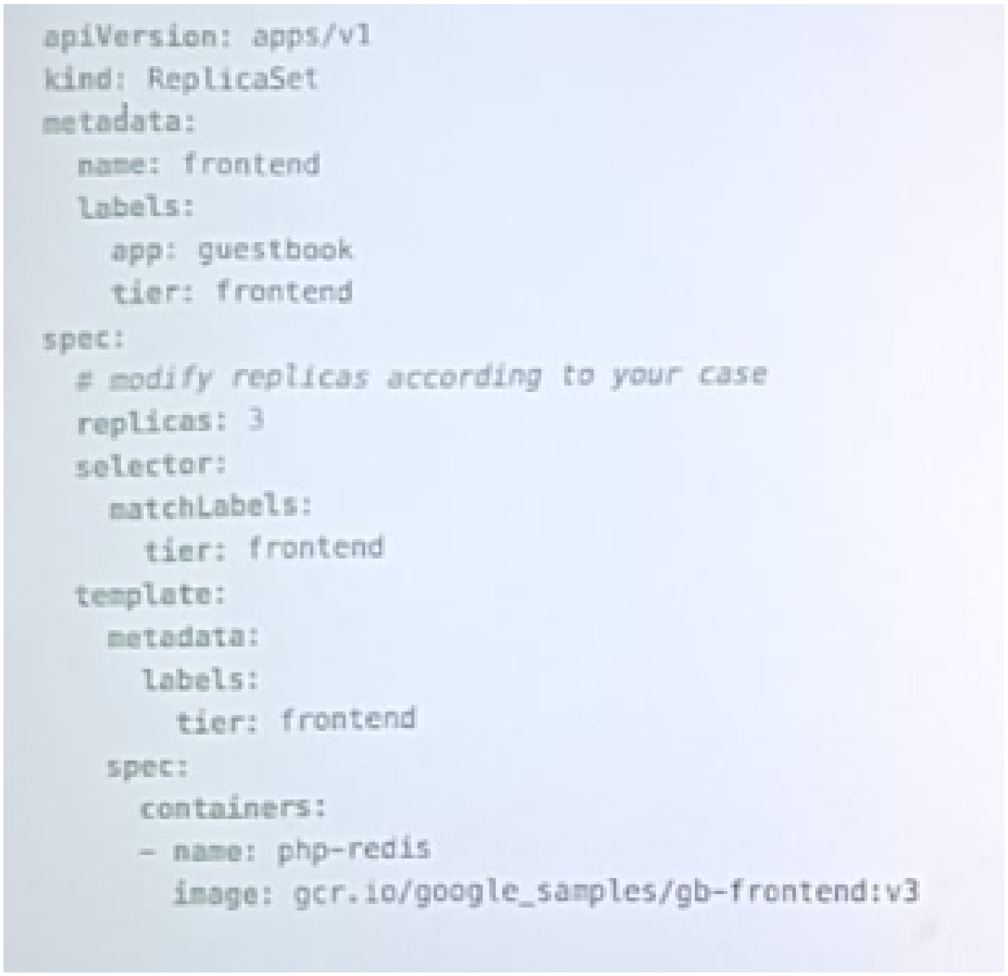
Jobs are in the batch API group, version v1.

You're not specifying a pod selector (it will be created based on the labels in the pod template).

Jobs can't use the default restart policy, which is Always.

CronJobs

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



```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

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.