Google Developer Groups

# GDG Cloud Team: Google Cloud Associate

2025/1/4

Jinsuk Park

# Index



Google Developer Groups On Campus

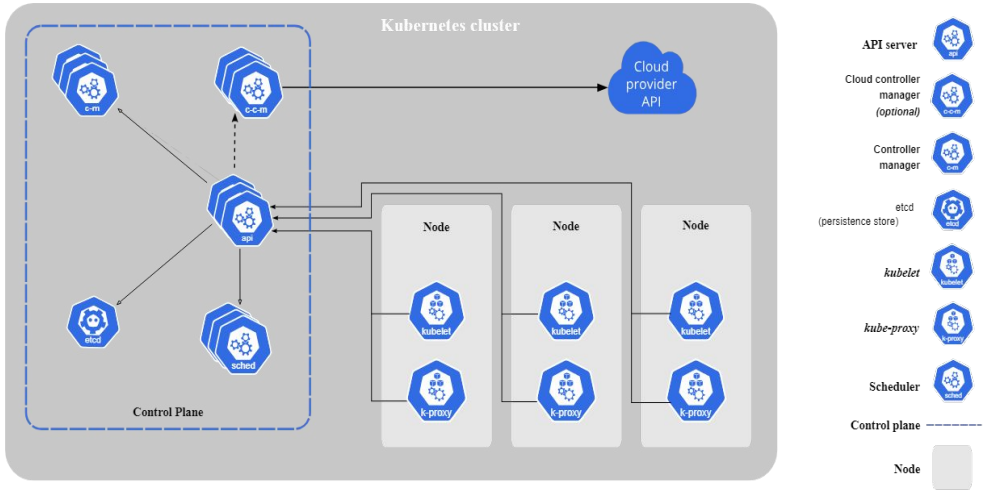
# What is Kubernetes?

An open source container orchestration solution

Key Features

- Provides Cluster Management

-> Auto scaling, Load Balancer, Self healing, Service Discovery, Zero downtime deployments



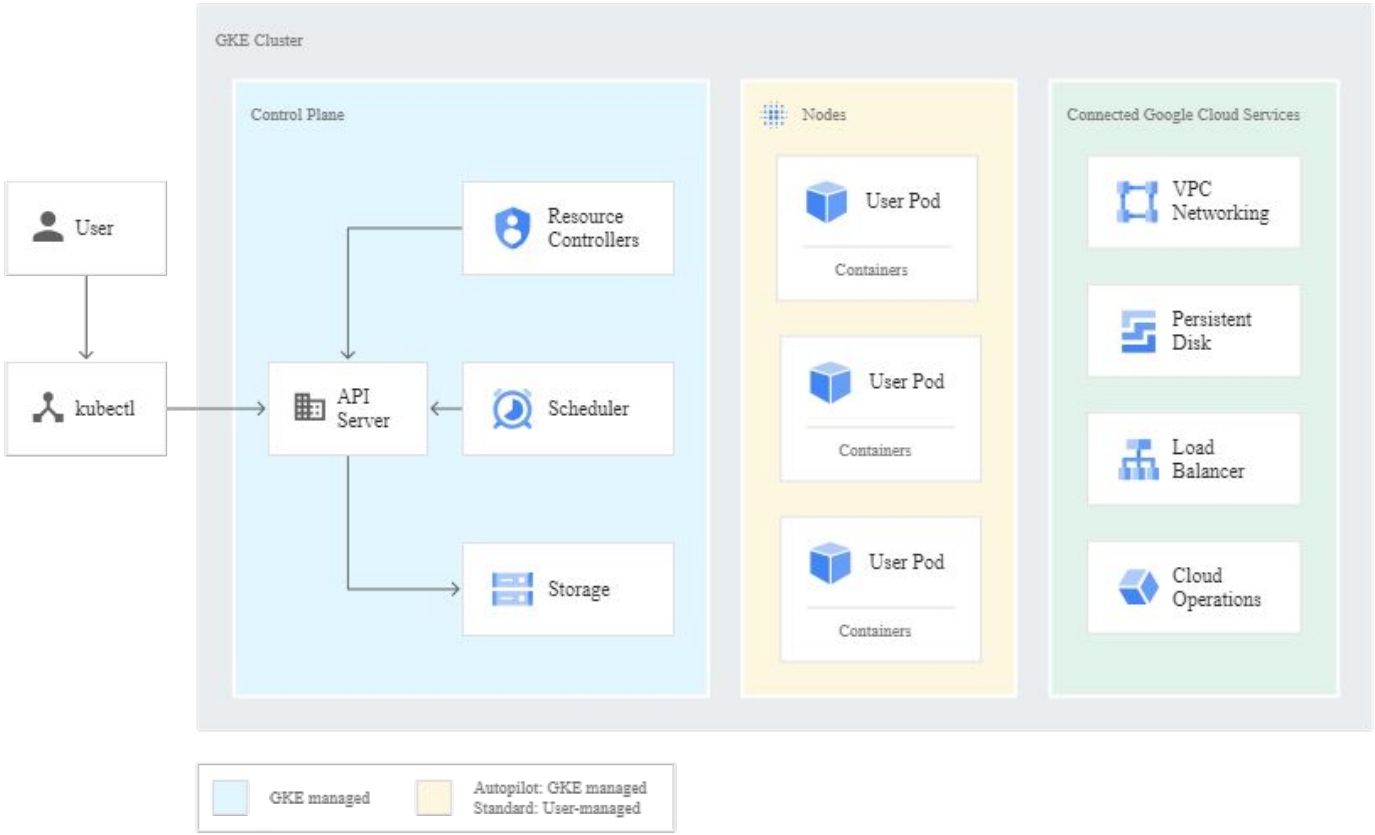
## Google Kubernetes Engine (GKE)

Google Kubernetes Engine

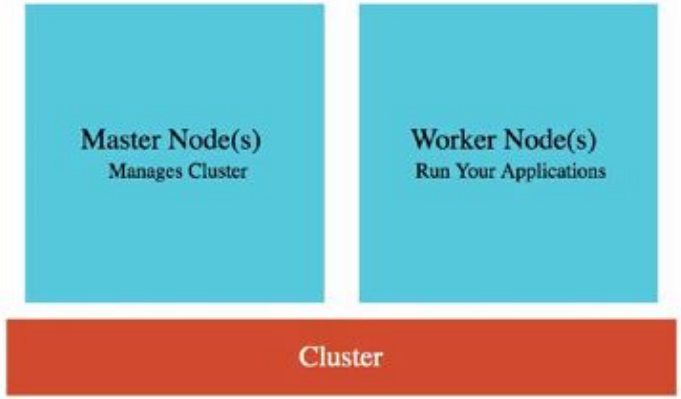
**Features:**

- **Managed** Kubernetes Service (user only focuses on app development)
- **Auto-Repair** (repair failed nodes) & **Auto-Upgrade** (use latest version)
- Pod & Cluster **Autoscaling** -> dynamically adjusts pods/nodes based on demands
- Cloud logging & Monitoring
- **Container-optimized OS** (improves security)
- Provides support for persistent disks & local SSD

# Cluster


GKE Cluster
Control Plane
Nodes
Connected Google Cloud Services
User
kubectl
API Server
Resource Controllers
Scheduler
Storage
User Pod
Containers
User Pod
Containers
User Pod
Containers
VPC Networking
Persistent Disk
Load Balancer
Cloud Operations
GKE managed
Autopilot: GKE managed
Standard: User-managed

## What is a Cluster?




A group of Compute Engine Instances:
(1) **Master Node** (2) **Worker Nodes**

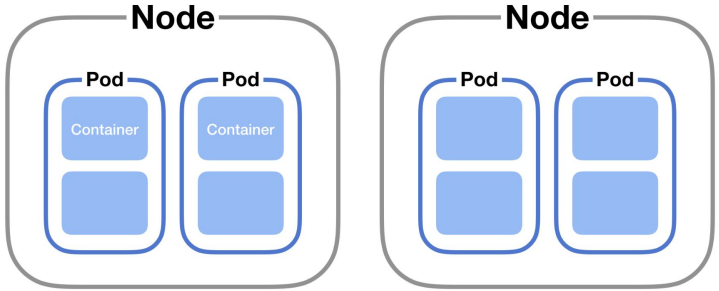
(1) **Master Node (Control Plane):**
- API Server: handles cluster communication
- Scheduler: decides which node to place pods
- Control Manager: manages deployments & replica sets
- Etcd: distributed DB storing the cluster state

(2) **Worker Nodes**
- Runs our pods
- Kubelet: manages communication with master node

## What is a Pod?

Smallest deployable unit in Kubernetes

- A pod has at least **one container**
- Each pod has an ephemeral IP address
- Containers in **SAME POD** share: network, storage, IP address, ports, volumes

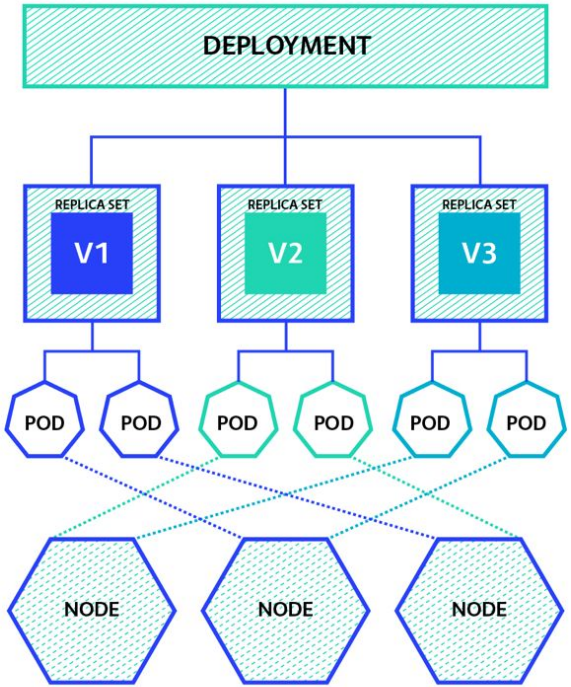


## What is a Deployment?

High-level abstractions that manages the app lifecycle

## Features

- **Rolling Updates**: gradually replacing old pods with new ones (minimize downtime)
- **Rollbacks**: easily roll back to previous versions
- **Versioning**: maintain version history
- **Self-Healing**: ensure the desired state by automatically replacing failed pods

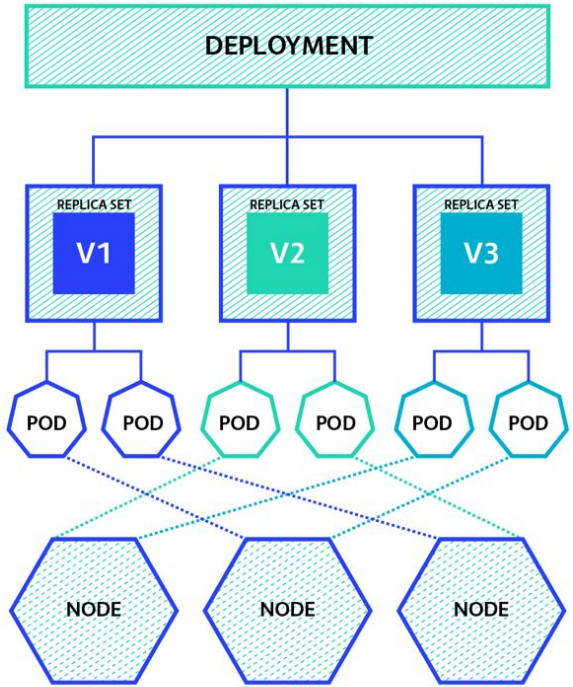


## What is a Replica Sets?

Lower-level abstractions that focus on maintaining a specified number of pod replicas

## Features

- **Pod Management**: ensure a desired number of identical pods are running at all times
- **Basic Scaling**: ReplicaSets provide a simple scaling mechanism for apps
- **Self-Healing**: If a pod fails, ReplicaSet automatically creates a new pod to maintain desired count



## Key Differences?

- **Abstraction Level:** Deployments are a higher-level resource that manages ReplicaSets. ReplicaSets are lower-level resources that directly manage pods

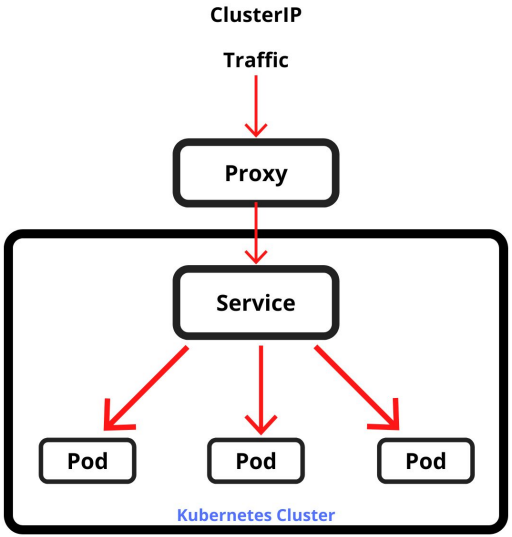
- **Update Mechanism:** Deployments offer sophisticated update strategies like rolling updates. ReplicaSets require manual updates

- **Versioning:** Deployments provide versioning and easy rollbacks, features not available in ReplicaSets

Deployments create & manage ReplicaSets which in turn create & manage pods. When we create a Deployment it automatically creates a ReplicaSet to handle the pod replicas

# What is a Service?

Essential for managing network access to pods -> Expose pods to outside world via stable IP address




## Why do we use it?

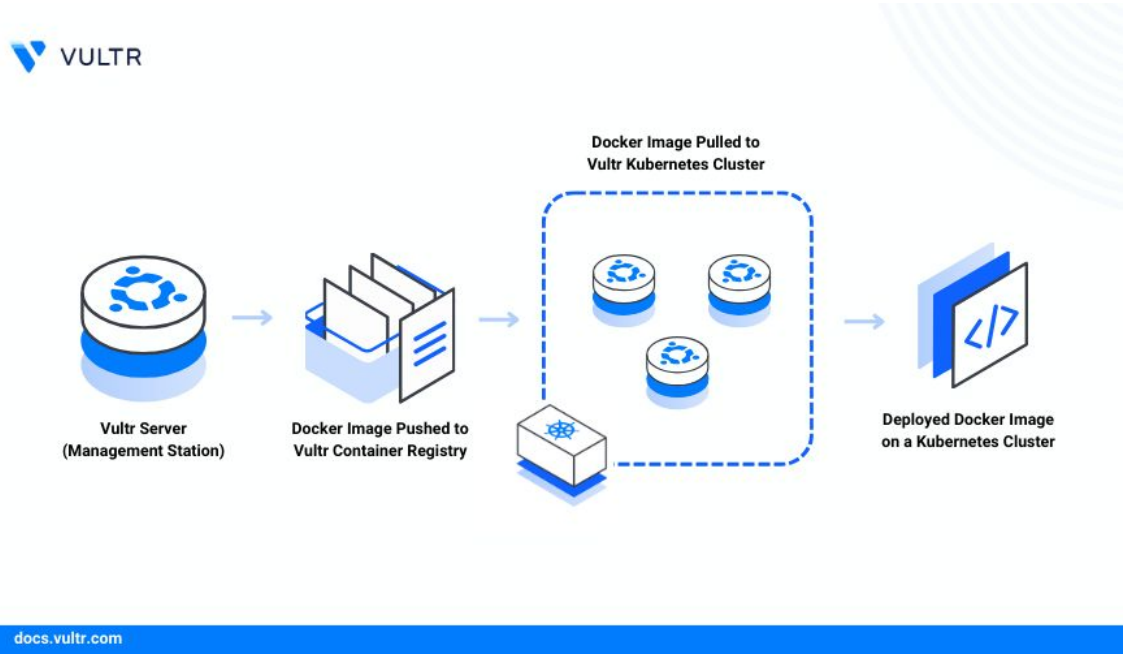
To ensure that external users are not impacted when
(1) a pod fails and replaced by ReplicaSet
(2) a new release happens and all existing pods are replaced by new ones

## Types

1. ClusterIP; cluster-internal IP
2. LoadBalancer: via GCP's load balancer (LoadBalancer for each microservice)
3. NodePort: Node's IP at a static port

# What is a Container Registry

We created docker images for our microservices.
So… where do we store them? **Container Registry!** (Docker Hub)



## Key Takeaways

- **For High Availability?** Replicate Master Nodes across multiple zones
- **(REMEMBER):** Some CPU on the nodes is reserved by Control Plane
- **Kubernetes supports stateful dpeloyments** (kafka, redis, zookeper)
- How do we run services on nodes for log collection/monitoring? DaemonSet
- Create Docker images for our microservices -> build - test - push to container repositry

# Scenarios-1

| Scenario | Solution |
| --- | --- |
| **You want to keep your costs low and optimize your GKE implementation** | Consider Preemptible VMs, Appropriate region, Committed-use discounts.<br>E2 machine types are cheaper than N1.<br>Choose right environment to fit your workload type (Use multiple node pools if needed). |
| **You want an efficient, completely auto scaling GKE solution** | Configure Horizontal Pod Autoscaler for deployments and Cluster Autoscaler for node pools |
| **You want to execute untrusted third-party code in Kubernetes Cluster** | Create a new node pool with GKE Sandbox. Deploy untrused code to Sandbox node pool. |

# Scenarios-2

| Scenario | Solution |
| --- | --- |
| **You want enable ONLY internal communication between your microservice deployments in a Kubernetes Cluster** | Create Service of type ClusterIP |
| **My pod stays pending** | Most probably Pod cannot be scheduled onto a node(insufficient resources) |
| **My pod stays waiting** | Most probably failure to pull the image |

# Quiz

You create a new Google Kubernetes Engine (GKE) cluster and want to make sure that it always runs a supported and stable version of Kubernetes. What should you do?

A. Enable the Node Auto-Repair feature for your GKE cluster.
B. Enable the Node Auto-Upgrades feature for your GKE cluster.
C. Select the latest available cluster version for your GKE cluster.
D. Select ز€Container-Optimized OS (cos)ز€ as a node image for your GKE cluster.

# Quiz

You need to select and configure compute resources for a set of batch processing jobs. These jobs take around 2 hours to complete and are run nightly. You want to minimize service costs. What should you do?

A. Select Google Kubernetes Engine. Use a single-node cluster with a small instance type.
B. Select Google Kubernetes Engine. Use a three-node cluster with micro instance types.
C. Select Compute Engine. Use preemptible VM instances of the appropriate standard machine type.
D. Select Compute Engine. Use VM instance types that support micro bursting.

# What is Google Cloud Functions?


**Imagine you want to execute some code when an event happens?**
(1) A file is uploaded in Cloud Storage
(2) An error log is written to Cloud Logging
(3) A message arrives to Cloud Pub/Sub
(4) A http/https invocation is received

A serverless computing platform that allows developers to build & deploy apps without managing infrastructure

- FaaS: Function as a Service
- Serverless: no need to manage servers
- Event-driven: triggered by specific events
- Scalable: scales based on demand (scale horizontally)

# What is Google Cloud Functions?


- Pay-as-you-go pricing
- Multiple language support: Node.js, Python, Go, Java, and .NET
- Easily connects with our GCP services
- Time-bound: 1 min ~ 60 min (not ideal for long processes)
- **Concurrency**: One function instances handles MULTIPLE request at the same time (max 1000)
- **Cold Start**: New function instances initialization from scratch can take time -> configure min number of instances

```
const escapeHtml = require('escape-html');

/**
 * HTTP Cloud Function.
 *
 * @param {Object} req Cloud Function request context.
 *                     More info: https://expressjs.com/en/api.html#req
 * @param {Object} res Cloud Function response context.
 *                     More info: https://expressjs.com/en/api.html#res
 */
exports.helloHttp = (req, res) => {
  res.send(`Hello ${escapeHtml(req.query.name || req.body.name || 'World')}!`);
};
```

```
/**
 * Background Cloud Function to be triggered by Pub/Sub.
 * This function is exported by index.js, and executed when
 * the trigger topic receives a message.
 *
 * @param {object} message The Pub/Sub message.
 * @param {object} context The event metadata.
 */
exports.helloPubSub = (message, context) => {
  const name = message.data
    ? Buffer.from(message.data, 'base64').toString()
    : 'World';

  console.log(`Hello, ${name}!`);
};
```

## What is Cloud Run?


Allows developers to run stateless containers directly on Google's infrastructure

- Fully managed & serverless
- Support for any programming language that can be containerized
- Built-in security with automatic HTTPS
- Pay-per-use pricing
- Integration with other GCP services

+ **Anthos**
- Run Kubernetes clusters anywhere (Cloud, Multi Cloud, On premise

# Encryption

## Data States

1) **Data at rest:** stored on devices/backups (hard disks, DB ..)

2) **Data in motion:** transferring across networks (data moving between on-premise-cloud storage ...)

3) **Data in use:** Active data processed in a non-persistent state - RAM

# Encryption

Encryption is essential for protecting data in all states

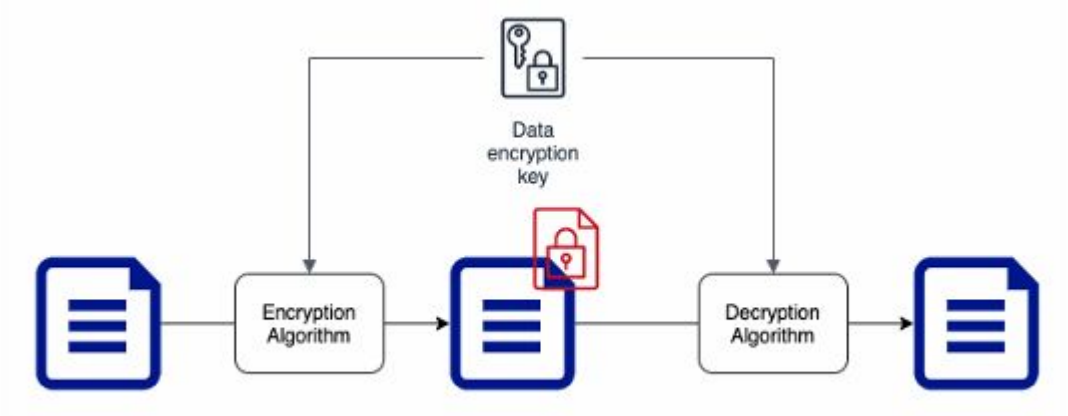
If we store data as is, what would happen if unauthorized someone gets access to it?

**First Law of Security: Defence in Depth**
-> Encrypt **ALL data states**

**Types**

1. Symmetric Key Encryption
2. Asymmetric Key Encryption
3. Cloud KMS (Key Management Service)

# 1. Symmetric Key Encryption




- **SAME key for encryption/decryption**

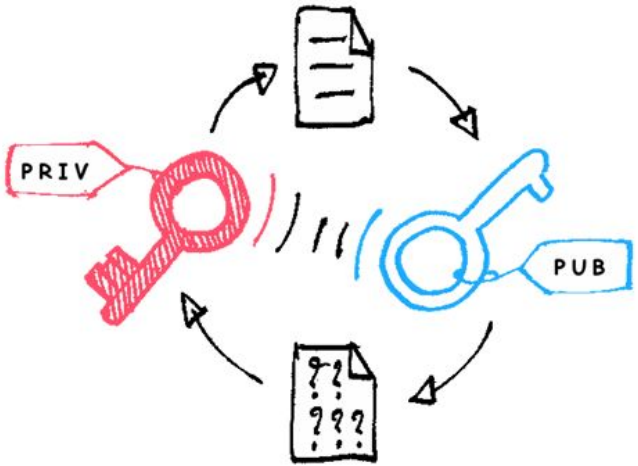
1. Choose right encryption algorithm
2. Secure encryption key
3. Share the encryption key securely

# 2. Asymmetric Key Encryption

- **2 keys: Public / Private Key**

1. Public Key: Shared openly - used for encryption
2. Private Key: Kept secret - used for decryption

Data encryption with public key can ONLY be decrypted with the private key



# 3. Cloud KMS (Key Management Service)


Service for managing cryptographic keys in cloud environments

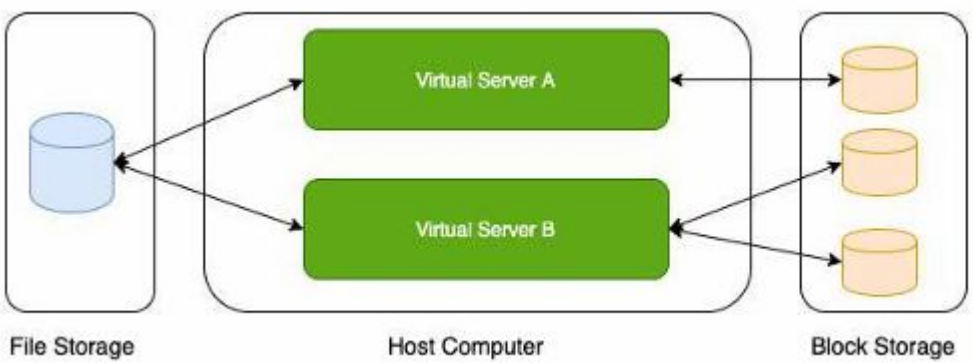
- Creation/management of symmetric/asymmetric keys
- API for encryption, decryption, and signing data

Options for key management

1. Google-managed keys
2. Customer-managed keys
3. Customer-supplied keys

Cloud KMS provides a secure & flexible way to handle encryption in cloud environments!

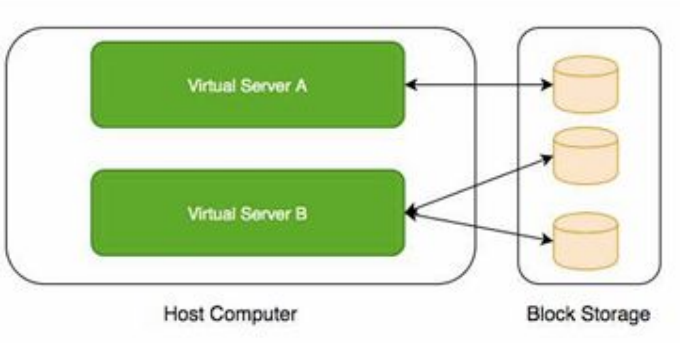
# Storage




What is the type of storage of our hard disk?
-> Block Storage

Designed for sharing files across multiple users
-> File Storage

# Block Storage




Storage used in hard disks attached to computers

**Connectivity:** one block storage device connects to 1 virtual server (read-only attachments to multiple servers allowed)

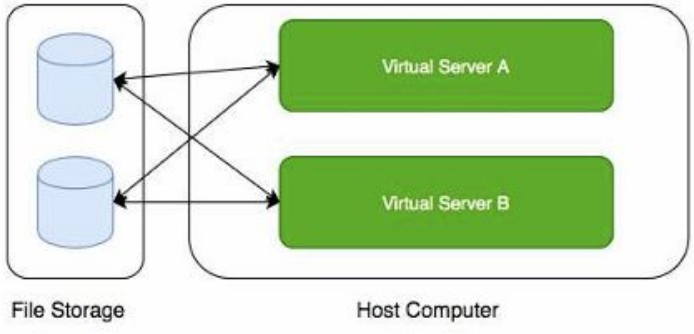
**Multiple Attachments:** Generally, 1 block device - 1 server.
BUT, a server can have multiple attachments

- **Direct-attached Storage (DAS):** similar to a hard disk (directly connected to computer)
- **Storage-Area Network (SAN):** high-speed network connecting a pool of storage devices

# File Storage

Designed for sharing files across multiple users

**Media Workflows:** supports processes like video editing that need large shared storage

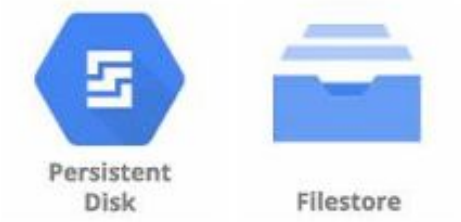



**Enterprise Usage:** Provides a quick & secure way for enterprise users to share files

**Hierarchy:** Organize data in a file and folder structure making it intuitive for users

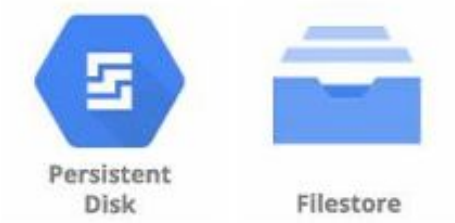
# GCP - Block & File Storage

1. **Block Storage** (attach to VM instances)
- **Persistent Disks:** Network Block Storage
- **Local SSDs:** Local block storage




2. **File Storage**
- **Filestore:** High performance file storage

# GCP Block Storage




1. **Persistent Disks**

- Network-based block storage

**Features**

- Most durable & reliable storage option
- Data persists independently of the VM instance's lifecycle
- Offers 99.99% availability, with data replications across multiple zones
- Supports features like snapshots, resizing, and automated encryption

**Snapshots**

- Schedule snapshots / Multi-regional / share across projects / **incremental** (only stores changes made since previous snapshot) **/ reduce** performance
- Creating snapshots from disk is faster than from image

# GCP Block Storage

| Scenarios | Machine image | Persistent disk snapshot | Custom image | Instance template |
|---|---|---|---|---|
| Single disk backup | Yes | Yes | Yes | No |
| Multiple disk backup | Yes | No | No | No |
| Differential backup | Yes | Yes | No | No |
| Instance cloning and replication | Yes | No | Yes | Yes |
| VM instance configuration | Yes | No | No | Yes |

# GCP Block Storage

| Scenario | Solution |
| --- | --- |
| **You want to improve performance of Persistent Disks (PD)** | Increase size of PD or Add more PDs. Increase vCPUs in your VM. |
| **You want to increase durability of Persistent Disks (PD)** | Go for Regional PDs (2X cost but replicated in 2 zones) |
| **You want to take hourly backup of Persistent Disks (PD) for disaster recovery** | Schedule hourly snapshots! |
| **You want to delete old snapshots created by scheduled snapshots** | Configure it as part of your snapshot scheduling! |

# GCP Block Storage

## 2. Local SSDs

- Physically attached to the host server of instance

**Features**

- Temporary, high-performance storage (fast performance)
- Data is ephemeral and tied to VM instance's lifecycle
- Ideal for caches, or flash-optimized DB
- Data automatically encrypted
- Choose NVMe-enabled & multi-queue SCSI images for best performance
- Larger Local SSds (more storage), More vCPUs (higher performance)
- **CANNOT** detach & attach to another VM instance

# GCP Block Storage

## Persistent Disks vs Local SSDs

| Feature | Persistent Disks | Local SSDs |
|---|---|---|
| **Attachment to VM instance** | As a network drive | Physically attached |
| **Lifecycle** | Separate from VM instance | Tied with VM instance |
| **I/O Speed** | Lower (network latency) | 10-100X of PDs |
| **Snapshots** | Supported | Not Supported |
| **Use case** | Permanent storage | Ephemeral storage |

# Cloud Filestore

Managed file storage service that provides shared cloud storage

**Features**

- NFS Protocol Support: allows easy integration with existing app and workflows
- Provisioned Capacity: users can allocate storage capacity based on their needs
- High performance: capable of delivering up to 16 GB/s throughput
- Supports HDD (general purpose) , SSD (performance-critical workloads)

Use cases: file share, media workflows

# GCP Block Storage

| Scenario | Solution |
| --- | --- |
| You want Very High IOPS but your data can be lost without a problem | Local SSDs |
| You want to create a high perfomance file sharing system in GCP which can be attached with multiple VMs | Filestore |
| You want to backup your VM configuration along with all its attached Persistent Disks | Create a Machine Image |
| You want to make it easy to launch VMs with hardened OS and customized software | Create a Custom Image |