ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION)

UNDERSTANDING CLOUD COMPUTING AND GCP ECOSYSTEM

CHAPTER 1: INTRODUCTION TO CLOUD COMPUTING

1.1 What is Cloud Computing?

Cloud computing is the **on-demand availability of computing resources** such as servers, storage, databases, networking, software, and analytics over the internet. Instead of owning and maintaining physical infrastructure, users can access these resources **on a pay-as-you-go basis**.

1.2 Key Characteristics of Cloud Computing

✓ **On-Demand Self-Service** – Users can provision resources as needed.
✓ **Broad Network Access** – Services are accessible over the internet.
✓ **Resource Pooling** – Shared infrastructure for multiple users.
✓ **Rapid Elasticity** – Automatic scaling based on demand.
✓ **Measured Service** – Pay only for what you use.

1.3 Types of Cloud Computing Models

Cloud Model	Description	Example
Public Cloud	Cloud infrastructure shared among multiple users	Google Cloud, AWS, Azure
Private Cloud	Cloud resources dedicated to a single organization	VMware Private Cloud
Hybrid Cloud	Combination of public and private cloud for flexibility	Google Anthos, Azure Hybrid Cloud
Multi-Cloud	Use of multiple cloud providers for redundancy and optimization	Using both AWS and GCP

📌 **Example:**
A **startup company** uses **Google Cloud Storage** for hosting its web applications, avoiding high upfront infrastructure costs.

CHAPTER 2: INTRODUCTION TO GOOGLE CLOUD PLATFORM (GCP)

2.1 What is Google Cloud Platform (GCP)?

Google Cloud Platform (GCP) is a **suite of cloud computing services** offered by Google. It provides a **scalable, secure, and high-performance** environment for building and deploying applications, storage solutions, databases, AI/ML models, and more.

2.2 Why Choose GCP?

✓ **Global Network Infrastructure** – High-speed fiber-optic connectivity.
✓ **Advanced AI & Machine Learning** – Google's expertise in AI tools like Vertex AI.
✓ **Security & Compliance** – Meets industry standards like ISO 27001, GDPR, HIPAA.

✓ **Cost Optimization** – Pay-as-you-go pricing and sustained-use discounts.

📌 **Example:**
A **gaming company** chooses GCP's **Compute Engine and BigQuery** to handle real-time player data analytics with low latency.

CHAPTER 3: GCP GLOBAL INFRASTRUCTURE

3.1 Google Cloud Regions & Zones

GCP has **over 35 regions and 100+ zones worldwide,** allowing businesses to deploy applications close to their users for **high availability and low latency**.

✓ **Regions** – Geographical areas with multiple **zones**.
✓ **Zones** – Independent data centers within a region.

Region	Country
us-east1	USA
europe-west1	Netherlands
asia-southeast1	Singapore
southamerica-east1	Brazil

3.2 Understanding Google Cloud Interconnect & CDN

✓ **Google Cloud Interconnect** – Provides **private network connectivity** to Google Cloud.
✓ **Google Cloud CDN** – Delivers content with low latency using **edge locations**.

📌 **Example:**
A **media streaming service** uses **Google Cloud CDN** to cache and distribute video content globally, ensuring smooth playback.

CHAPTER 4: CORE GOOGLE CLOUD SERVICES

4.1 Compute Services

✓ **Compute Engine** – Virtual machines for hosting applications.
✓ **Google Kubernetes Engine (GKE)** – Managed Kubernetes for container orchestration.
✓ **Cloud Functions** – Serverless execution for event-driven applications.

📌 **Example:**
A **web development agency** uses **Compute Engine** to deploy virtual machines with automatic scaling.

4.2 Storage & Database Services

✓ **Cloud Storage** – Object storage for files, images, and backups.
✓ **Cloud SQL** – Managed relational database for MySQL, PostgreSQL, and SQL Server.
✓ **BigQuery** – Serverless data warehouse for large-scale analytics.

📌 **Example:**
A **finance company** uses **BigQuery** to analyze millions of transactions in real-time.

4.3 Networking Services

✓ **Virtual Private Cloud (VPC)** – Customizable networking for cloud resources.
✓ **Cloud Load Balancing** – Distributes traffic across multiple

instances.

✓ **Cloud DNS** – Managed domain name system (DNS) service.

📌 **Example:**

An **e-commerce company** uses **Cloud Load Balancer** to ensure website availability during high-traffic sales events.

CHAPTER 5: SECURITY & IDENTITY MANAGEMENT IN GCP

5.1 Security Features of Google Cloud

✓ **IAM (Identity and Access Management)** – Granular user permissions.

✓ **Encryption at Rest and in Transit** – Secures data across all services.

✓ **Cloud Armor** – Protects applications from DDoS attacks.

✓ **Security Command Center** – Monitors threats and compliance.

📌 **Example:**

A **healthcare provider** uses **Cloud IAM** to restrict data access only to authorized users.

CHAPTER 6: GCP AI/ML & BIG DATA SERVICES

6.1 Artificial Intelligence & Machine Learning on GCP

✓ **Vertex AI** – End-to-end ML platform for model training and deployment.

✓ **AutoML** – No-code machine learning model generation.

✓ **Cloud Vision API** – Image recognition and classification.

📌 **Example:**

A **social media company** uses **Cloud Vision API** to automatically tag images with object detection.

6.2 Big Data & Analytics on GCP

✓ **BigQuery** – Petabyte-scale analytics engine.
✓ **Cloud Dataflow** – Real-time and batch data processing.
✓ **Cloud Pub/Sub** – Event-driven messaging system.

📌 **Example:**
A **retail chain** uses **BigQuery** to analyze customer purchase behavior and optimize store layouts.

CHAPTER 7: PRICING & COST OPTIMIZATION ON GCP

7.1 Understanding GCP's Pricing Model

✓ **Pay-as-you-go** – No upfront investment, pay only for usage.
✓ **Sustained Use Discounts** – Discounts for continuous resource usage.
✓ **Preemptible VMs** – Low-cost virtual machines for short-lived workloads.

7.2 Cost Optimization Best Practices

✓ **Use Cost Management Tools** – Track spending with **Google Cloud Pricing Calculator**.
✓ **Implement Autoscaling** – Automatically adjust resources based on demand.
✓ **Optimize Storage Costs** – Use **Coldline Storage** for infrequently accessed data.

📌 **Example:**
A **startup** saves **30% on cloud costs** by using **Preemptible VMs** for development workloads.

Chapter 8: Career Opportunities in Google Cloud Computing

8.1 Job Roles & Certifications in GCP

✓ **Cloud Engineer** – Deploys and manages GCP resources.
✓ **Cloud Architect** – Designs scalable cloud solutions.
✓ **Data Engineer** – Specializes in BigQuery and AI/ML.
✓ **Security Engineer** – Focuses on cloud security best practices.

8.2 Freelancing & Startup Opportunities in GCP

✓ **Freelancing** – Offer cloud consulting, migration, and DevOps services.
✓ **Startups** – Build SaaS applications leveraging GCP's scalability.

📌 **Example:**
A **freelance cloud consultant** helps businesses migrate from on-premises servers to GCP, earning lucrative contracts.

Chapter 9: Exercise & Review Questions

Exercise:

1. **Set up a GCP free trial account** and deploy a virtual machine.
2. **Create a Cloud Storage bucket** and upload a file.
3. **Analyze a sample dataset using BigQuery.**

Review Questions:

1. What are the **key characteristics** of cloud computing?
2. How does **Google Cloud differ from AWS and Azure**?
3. What is the **difference between Compute Engine and Kubernetes Engine**?

4. What are **two cost optimization strategies** in GCP?
5. How does **Cloud IAM enhance security** in GCP?

Conclusion: Mastering GCP for Cloud Success

Google Cloud Platform offers **powerful, scalable, and cost-effective** cloud solutions for businesses and developers. Mastering GCP services and best practices enables **career growth, freelancing, and startup opportunities** in the ever-expanding cloud industry. 🚀

GOOGLE CLOUD PLATFORM (GCP) SERVICES OVERVIEW

(Compute, Storage, Networking, Big Data, AI/ML)

CHAPTER 1: INTRODUCTION TO GOOGLE CLOUD PLATFORM (GCP)

1.1 What is Google Cloud Platform?

Google Cloud Platform (GCP) is a suite of **cloud computing services** that run on the same infrastructure as Google's search engine, YouTube, and Gmail. GCP provides services in:

✓ **Compute** – Virtual Machines, Kubernetes, and Serverless Computing.
✓ **Storage** – Object Storage, Databases, and File Systems.
✓ **Networking** – Virtual Private Cloud (VPC), Load Balancers, and DNS.
✓ **Big Data & Analytics** – Data Warehousing, Streaming, and Business Intelligence.
✓ **AI/ML** – Pre-trained AI Models and Custom Machine Learning Services.

📌 **Example:**
A **global e-commerce company** uses GCP for **scalable hosting, real-time analytics, and AI-powered recommendations**.

CHAPTER 2: COMPUTE SERVICES IN GCP

2.1 Overview of Compute Services

GCP Compute Service	Use Case

Compute Engine	Virtual Machines for flexible workloads
Google Kubernetes Engine (GKE)	Managed Kubernetes clusters for containers
Cloud Run	Serverless container execution
App Engine	Fully managed platform for web apps
Cloud Functions	Event-driven, serverless compute

📌 **Example:**
A **financial services firm** runs **Compute Engine instances** for fraud detection models and **Cloud Functions** for real-time alerts.

2.2 Deploying a Virtual Machine in Compute Engine

Step 1: Create a Virtual Machine

1. Navigate to **Google Cloud Console** → Go to **Compute Engine**.
2. Click **Create Instance** → Choose Machine Type (e.g., e2-standard-2).
3. Select **Operating System** (Ubuntu, Windows, etc.).

Step 2: Configure Networking & Security

1. Attach to **Virtual Private Cloud (VPC)**.
2. Assign a **Static External IP** (optional).
3. Configure **Firewall Rules** (Allow SSH, HTTP, HTTPS).

Step 3: Deploy and Connect

- **Linux VM:**

- gcloud compute ssh my-vm-instance
- **Windows VM:** Use **RDP (Remote Desktop Protocol)**.

📌 **Example:**
A **startup deploys a Compute Engine VM** to run its **Node.js backend** with auto-scaling enabled.

2.3 Deploying Serverless Applications with Cloud Run

1. Navigate to **Cloud Run** → Click **Create Service**.
2. Select **Container Image (from Google Container Registry)**.
3. Define **Scaling Options** (Auto-scale from 0 to N instances).
4. Deploy and obtain a **public endpoint** for the application.

📌 **Example:**
A **logistics company** runs **Cloud Run services** to process shipment tracking events in real-time.

CHAPTER 3: STORAGE SERVICES IN GCP

3.1 Overview of GCP Storage Solutions

Storage Service	Use Case
Cloud Storage	Scalable object storage (images, videos, backups)
Persistent Disks	Block storage for Compute Engine VMs
Filestore	Managed file storage for shared workloads

Cloud SQL	Fully managed relational database (MySQL, PostgreSQL, SQL Server)
Bigtable	NoSQL wide-column database for high-throughput workloads
Firestore	Serverless NoSQL document database

📌 **Example:**
A **media streaming platform** stores **videos in Cloud Storage**, uses **Firestore for metadata**, and **Bigtable for user activity logs**.

3.2 Storing Data in Cloud Storage

Step 1: Create a Cloud Storage Bucket

1. Navigate to **Cloud Console** → Go to **Cloud Storage**.
2. Click **Create Bucket** → Enter a **Unique Name**.
3. Choose **Storage Class** (Standard, Nearline, Coldline, or Archive).

Step 2: Upload & Manage Files

1. Click **Upload Files** → Select files to store.
2. Set **Permissions** (Public, Private, IAM-based).
3. Enable **Object Versioning** for recovery.

Step 3: Access Files via CLI

gsutil cp local-file.txt gs://my-storage-bucket/

gsutil ls gs://my-storage-bucket/

📌 **Example:**
A **gaming company** uses **Cloud Storage** to store **game assets and logs,** ensuring fast retrieval and low latency.

CHAPTER 4: NETWORKING SERVICES IN GCP

4.1 Overview of GCP Networking Services

Networking Service	Use Case
Virtual Private Cloud (VPC)	Secure private networking
Cloud Load Balancing	Distributes traffic across instances
Cloud Interconnect	High-speed on-premises to cloud connectivity
Cloud DNS	Scalable domain name resolution
Cloud CDN	Content delivery network for faster web performance

📌 **Example:**
A **travel booking website** uses **Cloud Load Balancing** and **Cloud CDN** to **speed up content delivery** globally.

4.2 Configuring a Virtual Private Cloud (VPC)

Step 1: Create a VPC

1. Navigate to **VPC Network** → Click **Create VPC**.
2. Define **IP CIDR Range** (e.g., 10.0.0.0/16).
3. Create **Subnets** for different workloads.

Step 2: Attach Compute Resources

1. Assign **Compute Engine VMs** to the VPC.
2. Configure **Firewall Rules** for traffic filtering.
3. Set up **Private Google Access** for internal resources.

📌 **Example:**
A **banking company** isolates **customer-facing services and backend APIs** using **multiple VPC subnets**.

CHAPTER 5: BIG DATA & ANALYTICS IN GCP

5.1 Overview of GCP Big Data Services

Big Data Service	Use Case
BigQuery	Serverless data warehousing & analytics
Dataflow	Real-time stream & batch processing
Dataproc	Managed Hadoop & Spark for big data processing
Pub/Sub	Event-driven messaging for streaming

📌 **Example:**
A **retail chain** uses **BigQuery** to analyze **customer transactions** and optimize marketing strategies.

5.2 Running a Query in BigQuery

1. Navigate to **BigQuery Console** → Click **SQL Workspace**.
2. Select **Dataset & Table** → Run an SQL query:

```
SELECT product_name, SUM(sales)
FROM `retail.sales_data`
GROUP BY product_name
ORDER BY SUM(sales) DESC;
```

📌 **Example:**
A **social media company** uses **BigQuery** to analyze **billions of user interactions** daily.

CHAPTER 6: AI/ML SERVICES IN GCP

6.1 Overview of GCP AI/ML Services

AI/ML Service	Use Case
Vertex AI	Unified AI model training & deployment
Cloud AutoML	No-code AI for non-experts
Speech-to-Text	Convert speech into text
Vision AI	Image recognition & object detection
Natural Language AI	Text analysis & sentiment detection

📌 **Example:**
An **e-commerce store** uses **Vision AI** for **product image recognition** and **recommendations**.

6.2 Deploying a Machine Learning Model with Vertex AI

1. Navigate to **Vertex AI** → Click **Create Model**.
2. Upload **Training Dataset** (CSV, JSON, or BigQuery).

3. Select **AutoML or Custom Model Training**.
4. Deploy Model & Test Predictions.

📌 **Example:**
A **healthcare startup** trains **AI models in Vertex AI** to predict **disease risks based on patient history**.

CHAPTER 7: CONCLUSION & NEXT STEPS

✓ **GCP provides a complete cloud ecosystem for compute, storage, networking, big data, and AI.**
✓ **Businesses leverage GCP for scalability, analytics, and intelligent automation.**
✓ **Hands-on experience with GCP services is essential for mastering cloud computing.**

🚀 **Next Steps:**
✓ Set up a **free GCP account**.
✓ Deploy a **Compute Engine VM**.
✓ Run a **BigQuery analysis**.
✓ Train an **AI model in Vertex AI**.

👉 **Master GCP & accelerate your cloud career!**

IDENTITY AND ACCESS MANAGEMENT (IAM) IN GOOGLE CLOUD PLATFORM (GCP)

CHAPTER 1: INTRODUCTION TO IAM IN GCP

What is Identity and Access Management (IAM)?

Identity and Access Management (IAM) in **Google Cloud Platform (GCP)** is a **centralized framework** that enables organizations to **control access to cloud resources** securely. IAM helps manage **who (users, groups, service accounts)** can do **what (permissions)** on **which resources**.

Key Benefits of IAM in GCP

✓ **Granular Access Control** – Allows fine-grained permissions for users and services.
✓ **Role-Based Access Management** – Assign predefined or custom roles to users.
✓ **Security & Compliance** – Ensures **least privilege access** and regulatory compliance.
✓ **Auditing & Monitoring** – Provides logs of **who accessed what resources** and when.

📌 **Example:**
A **banking institution** uses IAM to ensure that only **database administrators** can manage **Cloud SQL instances,** while developers have **read-only** access.

CHAPTER 2: KEY COMPONENTS OF IAM IN GCP

2.1 IAM Policies

IAM policies are **rules that define permissions** for users or services. A policy consists of:

- **Principal (Who):** Users, groups, service accounts, or Google groups.
- **Role (What):** A set of permissions (e.g., **Viewer, Editor, Owner**).
- **Resource (Where):** The **Google Cloud resource** (e.g., **Compute Engine, Cloud Storage**).

IAM follows a **hierarchical model**, applying policies at different levels:

1. **Organization Level** – Applies policies across the entire organization.
2. **Folder Level** – Groups related projects for easier management.
3. **Project Level** – Grants access to a specific GCP project.
4. **Resource Level** – Provides granular access control to specific resources.

📌 **Example:**
A **tech startup** applies IAM policies at the **organization level** to ensure all **projects inherit security rules**.

2.2 IAM Roles & Permissions

IAM uses **roles** to assign permissions. There are three types of roles:

Role Type	Description	Example

Basic Roles	Broad access roles for legacy support	Owner, Editor, Viewer
Predefined Roles	Google-managed roles with specific permissions	roles/cloudsql.admin
Custom Roles	User-defined roles for specific needs	roles/custom.developer

📌 **Example:**
A **data analyst** is assigned the **BigQuery Data Viewer** role (roles/bigquery.dataViewer), allowing **read-only access to datasets**.

2.3 IAM Identities (Principals)

IAM allows **different identities** to authenticate and access GCP resources:

✓ **Google Account:** Personal or work accounts (e.g., user@gmail.com).
✓ **Google Group:** A collection of users with the same permissions.
✓ **Service Accounts:** Used by applications and workloads (e.g., CI/CD tools).
✓ **Cloud Identity / Workspace Users:** Managed enterprise users.

📌 **Example:**
A **Kubernetes cluster** uses a **service account** to authenticate with **Cloud Storage** for pulling container images.

CHAPTER 3: CONFIGURING IAM POLICIES IN GCP

3.1 Assigning IAM Roles to a User

Step 1: Open IAM in Google Cloud Console

1. Go to **Google Cloud Console → IAM & Admin → IAM**.
2. Select the **Project, Folder, or Organization**.

Step 2: Add a New IAM Member

1. Click **+ Add**.
2. Enter the **user's email or service account**.
3. Select a **role** (e.g., roles/storage.objectViewer).
4. Click **Save**.

📌 **Example:**
A **developer** is assigned the **Compute Engine Admin role,** allowing them to **create and manage virtual machines**.

3.2 Creating a Custom IAM Role

Step 1: Define the Custom Role

1. Navigate to **IAM & Admin → Roles →** Click **+ Create Role**.
2. Enter **Role Name, Description, and Permissions**.

Step 2: Assign the Custom Role

1. Go to **IAM →** Select **Project or Resource**.
2. Click **Add Member →** Assign the **Custom Role**.

📌 **Example:**
A **DevOps engineer** creates a **custom role** that allows only **restart privileges** for VMs but restricts full compute access.

CHAPTER 4: IAM BEST PRACTICES FOR SECURITY & COMPLIANCE

4.1 Implement Least Privilege Access

✓ **Avoid using roles/editor or roles/owner** for all users.
✓ **Assign roles at the lowest resource level** (not at the organization level).
✓ **Use predefined roles instead of broad basic roles**.

📌 **Example:**
A **data scientist** gets **BigQuery Data Editor access** only for a **specific dataset,** not the entire project.

4.2 Enable IAM Policy Auditing with Cloud Audit Logs

✓ **Enable Audit Logs** to track IAM activity.
✓ **Monitor role assignments** using **Cloud Logging**.
✓ **Detect policy changes** using **Security Command Center**.

📌 **Example:**
A **financial services firm** enables **Cloud Audit Logs** to track **who accessed customer records** in **Cloud Storage**.

4.3 Secure Service Accounts

✓ **Use Workload Identity Federation** instead of long-lived service account keys.
✓ **Limit service account permissions to required APIs**.
✓ **Rotate service account keys regularly**.

📌 **Example:**
A **machine learning pipeline** in **Vertex AI** uses **Workload Identity Federation** to access Google Cloud APIs securely.

CHAPTER 5: ADVANCED IAM FEATURES IN GCP

5.1 IAM Conditions for Fine-Grained Access Control

IAM Conditions allow **conditional access** based on:

✓ **Date/time** (e.g., temporary access).

✓ **Resource attributes** (e.g., only specific storage buckets).

✓ **IP addresses** (e.g., allow access only from corporate networks).

Example Condition for Temporary Access

```
"condition": {
  "title": "Temporary Access",
  "expression": "request.time < timestamp('2025-01-01T00:00:00Z')"
}
```

📌 **Example:**
A **contractor** gets **Cloud Storage read access** for **90 days,** after which the policy expires.

5.2 Using Google Cloud Organization Policies

✓ **Restrict external sharing of Cloud Storage buckets**.

✓ **Block public IP addresses for Compute Engine instances**.

✓ **Enforce encryption for all stored data**.

Step 1: Create an Organization Policy

1. Open **Organization Policies** in **Google Cloud Console**.
2. Click **+ Create Policy**.
3. Select a **constraint** (e.g., storage.uniformBucketLevelAccess).
4. Apply it to **projects or folders**.

📌 **Example:**
A **healthcare company** enforces **Google Cloud Organization Policies** to **prevent accidental exposure of sensitive patient data**.

Chapter 6: Case Study – Implementing IAM for a Multi-Cloud SaaS Platform

Problem Statement:

A **SaaS company** requires **secure, role-based access** to **Cloud Storage, BigQuery, and Compute Engine** while ensuring **compliance with GDPR and SOC 2**.

Solution Implementation:

1. **Configured IAM roles**:
    - Developers → **Compute Engine Admin**.
    - Analysts → **BigQuery Data Viewer**.
    - Service Accounts → **Least Privilege Access** for APIs.
2. **Implemented IAM Conditions**:
    - Developers **cannot modify storage buckets**.
    - External contractors get **temporary IAM access**.
3. **Enabled Cloud Audit Logs** for tracking IAM policy changes.

Results:

✓ **Reduced security risks by 80%** using least privilege access.
✓ **Improved compliance with GDPR** by **limiting data access** to authorized users.
✓ **Automated security monitoring** with **IAM policy alerts**.

CHAPTER 7: REVIEW QUESTIONS & EXERCISES

Exercise:

1. **Create a new IAM role** with limited access to BigQuery.
2. **Assign a service account** to a GCP resource.
3. **Enable IAM audit logs** and review access history.
4. **Apply IAM conditions** to allow access only during business hours.

Review Questions:

1. What is the difference between **basic, predefined, and custom IAM roles**?
2. How does **least privilege access** improve security?
3. What are **IAM Conditions**, and how are they used?
4. How do **Cloud Audit Logs help with compliance**?

CONCLUSION: STRENGTHENING SECURITY WITH IAM IN GCP

By **implementing IAM policies, enforcing least privilege access, and using auditing tools**, organizations can **secure Google Cloud resources and meet compliance requirements**. 🚀

VIRTUAL MACHINES WITH COMPUTE ENGINE

CHAPTER 1: INTRODUCTION TO VIRTUAL MACHINES & COMPUTE ENGINE

Understanding Virtual Machines (VMs) in Cloud Computing

A **Virtual Machine (VM)** is a **software-based emulation** of a physical computer that runs an operating system and applications. Cloud providers like **Google Cloud (GCP), Microsoft Azure, and AWS** offer VMs to help businesses **scale applications, run workloads, and manage infrastructure efficiently**.

What is Compute Engine?

Google Cloud's **Compute Engine** is an **Infrastructure-as-a-Service (IaaS) solution** that provides **highly scalable virtual machines** on demand. It offers **custom machine types, auto-scaling, GPU support, and persistent storage**.

Key Benefits of Compute Engine

✓ **Scalability** – Easily scale VMs up/down based on workload.
✓ **Custom Machine Types** – Configure VMs with **custom CPU, RAM, and disk sizes**.
✓ **High Performance** – Supports **GPUs & SSD storage** for faster computing.
✓ **Security & Compliance** – Provides **firewall rules, IAM roles, and encryption**.
✓ **Global Availability** – Deploy VMs across **multiple regions and zones**.

📌 **Example:**
A **gaming company** deploys Compute Engine VMs with **NVIDIA**

GPUs for **real-time graphics rendering and AI-based game physics**.

CHAPTER 2: CREATING AND MANAGING VMS IN COMPUTE ENGINE

2.1 Setting Up a Virtual Machine in Compute Engine

Step 1: Navigate to Compute Engine

1. Sign in to Google Cloud Console.
2. Select **Compute Engine** → Click **VM Instances**.
3. Click **+ Create Instance**.

Step 2: Configure VM Settings

1. **Name the VM** (e.g., my-web-server).
2. **Select a Region & Zone**:
    - Choose a region close to users for low latency.
    - Example: **us-central1-a** (Iowa).
3. **Select Machine Type**:
    - **E2 series (cost-effective)** for basic workloads.
    - **N1/N2 series (balanced CPU/RAM)** for enterprise applications.
    - **C2 series (compute-optimized)** for high-performance tasks.
4. **Select Boot Disk**:
    - **Ubuntu, Debian, Windows, or Custom Image**.
    - Select **SSD or Standard Disk**.

5. **Configure Networking & Firewall**:
    - Enable **Allow HTTP/HTTPS Traffic** (if hosting a web application).
    - Set **Internal & External IP Addresses**.

Step 3: Create and Start the VM

1. Click **Create** → Wait for the VM to start.
2. Click **SSH** to access the VM terminal.

📌 **Example:**
A **developer** sets up a **Compute Engine VM running Ubuntu** to **host a Node.js API** with **automatic scaling**.

CHAPTER 3: MANAGING & OPTIMIZING VM PERFORMANCE

3.1 VM Resource Scaling

✓ **Vertical Scaling:** Increase **CPU, RAM, or Disk size** manually.
✓ **Horizontal Scaling:** Use **Managed Instance Groups** for automatic VM scaling.

3.2 Enabling Auto-Scaling for Compute Engine VMs

1. Navigate to **Compute Engine** → Click **Instance Groups**.
2. Click **+ Create Instance Group**.
3. Select **Autoscaling Enabled** → Define **Minimum & Maximum VM Count**.
4. Set Scaling Policy (e.g., scale up at **70% CPU usage**).

3.3 Using Load Balancers for High Availability

✓ **Set up HTTP(S) Load Balancer** to distribute traffic across multiple VMs.
✓ **Enable Cloud CDN** to cache static content for faster delivery.
✓ **Use Global Load Balancing** for disaster recovery.

📌 **Example:**
A **video streaming platform** uses **Compute Engine VMs with auto-scaling & load balancing** to handle **peak user traffic** during live events.

CHAPTER 4: NETWORKING & SECURITY FOR COMPUTE ENGINE VMS

4.1 Configuring Firewall Rules

1. Navigate to **VPC Network** → Click **Firewall**.
2. Click **+ Create Rule** → Define **Allow/Deny Rules**.
3. Allow **TCP: 22 (SSH), 80 (HTTP), 443 (HTTPS)** if needed.
4. Apply rules to specific **VM instances or subnets**.

4.2 Setting Up Identity & Access Management (IAM)

✓ **Use IAM roles to control access** to VMs.
✓ Assign **Least Privilege Roles** (e.g., only admins can modify instances).
✓ Enable **Service Accounts** to manage automated VM tasks.

4.3 Encrypting VM Data with Persistent Disks

✓ **Use Customer-Managed Encryption Keys (CMEK)** for full control.
✓ **Enable disk snapshots** for disaster recovery.
✓ **Use Shielded VMs** for **protection against rootkits & unauthorized firmware changes**.

📌 **Example:**
A **banking application** encrypts customer data at rest using **Google's CMEK and automatic disk snapshots** for recovery.

Chapter 5: Backup, Snapshots & Disaster Recovery

5.1 Creating VM Snapshots

1. Navigate to **Compute Engine** → Click **Snapshots**.
2. Click **+ Create Snapshot** → Select the source disk.
3. Set **Snapshot Schedule** (daily/weekly backups).

5.2 Configuring Automatic Backups

✓ Enable **Automatic Snapshots** to recover from data loss.
✓ Store backups in **Google Cloud Storage (Coldline/Archive)**.

5.3 Disaster Recovery with Multi-Region Deployment

✓ Deploy **VMs in multiple regions** to prevent downtime.
✓ Enable **Cloud Load Balancer** to failover between regions.

📌 **Example:**
A **healthcare organization** configures **multi-region VM deployment** to ensure **zero downtime** in case of hardware failure.

Chapter 6: Cost Optimization Strategies for Compute Engine

6.1 Choosing the Right VM Type

✓ **Preemptible VMs** – Low-cost VMs for **non-critical batch jobs**.
✓ **Committed Use Discounts** – Save up to **70%** on long-term VM usage.

✓ **Sustained Use Discounts** – Automatic cost reduction for **high-usage VMs**.

6.2 Auto-Shutdown & Idle VM Management

✓ Set **auto-shutdown policies** for unused development VMs.
✓ Use **Stackdriver Monitoring** to detect **idle VMs** and optimize costs.

6.3 Using Spot VMs for Cost-Efficient Compute Workloads

✓ Spot VMs are **cheap alternatives** for running **fault-tolerant workloads**.
✓ Ideal for **Big Data, AI Training, and CI/CD Pipelines**.

📌 **Example:**
A **data science team** runs **AI training jobs** using **Spot VMs,** reducing cloud expenses by **50%**.

CHAPTER 7: CASE STUDY – SCALING AN E-COMMERCE PLATFORM WITH COMPUTE ENGINE

Problem Statement:

An **e-commerce company** experiences **high traffic spikes** during seasonal sales.

Solution Implementation:

1. **Deployed Compute Engine VMs** with **auto-scaling enabled**.
2. **Configured a Load Balancer** to handle traffic across multiple VM instances.
3. **Implemented firewall rules & IAM roles** to **secure access to VMs**.
4. **Enabled snapshots & backups** to ensure data recovery.

Results:

✓ **100% uptime** achieved during high-traffic periods.
✓ **Page load times improved by 40%** with SSD storage.
✓ **Reduced costs by 30%** using **preemptible instances** for non-critical workloads.

Chapter 8: Exercise & Review Questions

Exercise:

1. **Create a Compute Engine VM** running Ubuntu.
2. **Enable auto-scaling** for a VM instance group.
3. **Configure firewall rules** to allow only HTTP/HTTPS traffic.
4. **Create a snapshot** for VM backup.
5. **Deploy a Load Balancer** to distribute traffic across multiple VMs.

Review Questions:

1. What are the benefits of **using Compute Engine over physical servers**?
2. How do **Managed Instance Groups (MIGs) help with scaling**?
3. Why should **preemptible VMs** be used for batch workloads?
4. What is the role of **IAM roles in securing Compute Engine**?
5. How can **Load Balancers improve application performance**?

Conclusion: Optimizing Compute Engine for Cloud Workloads

Google Cloud's **Compute Engine** provides **scalable, secure, and cost-efficient virtual machines** for a wide range of applications. By leveraging **auto-scaling, IAM security, load balancing, and cost optimization,** businesses can build **high-performance cloud solutions** with minimal overhead. 🚀

INTRODUCTION TO GOOGLE KUBERNETES ENGINE (GKE)

CHAPTER 1: UNDERSTANDING GOOGLE KUBERNETES ENGINE (GKE)

1.1 What is GKE?

Google Kubernetes Engine (GKE) is a **managed Kubernetes service** that allows users to deploy, manage, and scale containerized applications using **Google Cloud Platform (GCP)**. It automates many Kubernetes operations such as **cluster provisioning, scaling, monitoring, and networking**.

1.2 Why Use GKE?

✓ **Fully Managed Kubernetes** – Google takes care of **upgrades, security patches, and scaling**.
✓ **Auto-Scaling & Auto-Repair** – Ensures high availability and self-healing of containers.
✓ **Integrated with GCP Services** – Works seamlessly with **Cloud Storage, Cloud SQL, and BigQuery**.
✓ **Multi-Cluster & Hybrid Deployments** – Supports **Anthos for on-premises & multi-cloud** deployments.
✓ **Secure by Default** – Enforces **RBAC (Role-Based Access Control), Workload Identity, and VPC-native networking**.

📌 **Example:**
A **media streaming platform** deploys its **microservices architecture on GKE**, ensuring **auto-scaling and zero downtime** during peak hours.

CHAPTER 2: GKE ARCHITECTURE & COMPONENTS

2.1 Core Components of GKE

Component	Function
Cluster	A managed Kubernetes environment that contains nodes.
Nodes	Virtual machines (VMs) that run containerized applications.
Pods	The smallest deployable units that run application containers.
Deployments	Ensures that the desired number of pod replicas run at all times.
Services	Expose applications within the cluster or to external users.
Ingress	Manages external HTTP(S) traffic to services.
Persistent Volumes (PV)	Provides storage for containers.

📌 **Example:**
A **financial services firm** runs its **trading application on GKE** with **multiple microservices communicating via Kubernetes Services**.

CHAPTER 3: SETTING UP A GKE CLUSTER

3.1 Prerequisites

✓ **GCP Account** – Sign up for **Google Cloud Free Tier**.
✓ **gcloud CLI Installed** – Install Google Cloud SDK:

curl https://sdk.cloud.google.com | bash

gcloud init

✓ **Enable Kubernetes API** in Google Cloud Console.

3.2 Creating a GKE Cluster via Cloud Console

1. Navigate to **Google Cloud Console** → Go to **Kubernetes Engine**.
2. Click **Create Cluster** → Choose **Standard or Autopilot Mode**.
3. Configure **Node Pool Settings** (e.g., e2-standard-2 machine type).
4. Click **Create** to deploy the cluster.

3.3 Creating a GKE Cluster via CLI

1. **Create a Kubernetes Cluster**:
2. gcloud container clusters create my-gke-cluster \
3. --num-nodes=3 \
4. --zone=us-central1-a
5. **Connect to the Cluster**:
6. gcloud container clusters get-credentials my-gke-cluster --zone=us-central1-a
7. **Verify the Cluster is Running**:
8. kubectl get nodes

📌 **Example:**
A **logistics startup** creates a **GKE cluster with three nodes** to run its **order processing service** efficiently.

CHAPTER 4: DEPLOYING APPLICATIONS ON GKE

4.1 Deploying a Sample Application

1. **Create a Deployment YAML file (deployment.yaml)**:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
    - name: my-app
        image: gcr.io/my-project/my-app:latest
        ports:
```

```
      - containerPort: 8080
```

21. **Apply the Deployment**:

22. kubectl apply -f deployment.yaml

23. **Verify Running Pods**:

24. kubectl get pods

📌 **Example:**

A **news website** deploys a **content management system (CMS) in GKE,** ensuring high availability using **three replicas**.

4.2 Exposing the Application with a Service

1. **Create a Service YAML file (service.yaml)**:

```
2. apiVersion: v1
3. kind: Service
4. metadata:
5.   name: my-service
6. spec:
7.   selector:
8.     app: my-app
9.   ports:
10.       - protocol: TCP
11.     port: 80
12.       targetPort: 8080
13. type: LoadBalancer
```

14. **Apply the Service Configuration**:

15. kubectl apply -f service.yaml

16. **Get the External IP**:

17. kubectl get services

📌 **Example:**
An **e-commerce platform** exposes its **customer checkout service** through a **LoadBalancer service** to ensure **scalable external access**.

CHAPTER 5: SCALING & AUTO-HEALING IN GKE

5.1 Horizontal Pod Auto-Scaling (HPA)

1. **Enable Auto-Scaling for a Deployment**:
2. kubectl autoscale deployment my-app --cpu-percent=50 --min=2 --max=5
3. **Verify Auto-Scaling Setup**:
4. kubectl get hpa

📌 **Example:**
A **food delivery app** scales **from 2 to 5 pods dynamically** during peak hours.

5.2 Auto-Repairing Nodes & Pods

- GKE automatically replaces failed nodes and pods.
- To manually restart a pod:
- kubectl delete pod my-app-12345

📌 **Example:**
A **healthcare company** ensures **zero downtime for patient appointment services** by enabling **auto-healing pods**.

CHAPTER 6: NETWORKING & SECURITY IN GKE

6.1 Configuring Ingress for HTTP(S) Traffic

1. **Create an Ingress Resource (ingress.yaml)**:
2. apiVersion: networking.k8s.io/v1
3. kind: Ingress
4. metadata:
5. name: my-ingress
6. spec:
7. rules:
8. - host: myapp.example.com
9. http:
10. paths:
11. - path: /
12. pathType: Prefix
13. backend:
14. service:
15. name: my-service
16. port:
17. number: 80

18. **Apply the Ingress Configuration**:
19. kubectl apply -f ingress.yaml

📌 **Example:**
A **travel agency** routes users to different **microservices (hotels, flights, bookings)** using a **Kubernetes Ingress Controller**.

6.2 Implementing Security Best Practices

✓ **Enable Role-Based Access Control (RBAC)**
✓ **Use Google-managed SSL certificates** for secure communication.
✓ **Implement Workload Identity for IAM authentication**.
✓ **Restrict public access to private workloads using VPC-native clusters**.

📌 **Example:**
A **fintech firm** enforces **RBAC policies** to prevent unauthorized access to **sensitive customer transactions**.

CHAPTER 7: MONITORING & LOGGING IN GKE

7.1 Enabling Cloud Logging & Monitoring

1. **View Cluster Logs in GCP Console** → Go to **Operations** → **Logging**.
2. **Use kubectl logs to check logs for a specific pod**:
3. kubectl logs my-app-12345
4. **Monitor CPU & Memory Usage**:
5. kubectl top pods

📌 **Example:**
A **stock market analytics firm** tracks **pod performance using Google Cloud Operations Suite** to optimize **real-time data processing**.

CHAPTER 8: CONCLUSION & NEXT STEPS

✓ **GKE simplifies Kubernetes management, scaling, and security.**
✓ **Ideal for microservices, AI/ML workloads, and scalable cloud-native applications.**
✓ **Hands-on experience with GKE is essential for Kubernetes career growth.**

🚀 **Next Steps:**
✓ Deploy a **GKE Cluster**.
✓ Create a **Kubernetes Deployment & Service**.
✓ Implement **auto-scaling & monitoring**.

👉 **Master GKE & accelerate your cloud-native development!**

CLOUD STORAGE & PERSISTENT DISKS IN GOOGLE CLOUD PLATFORM (GCP)

CHAPTER 1: INTRODUCTION TO CLOUD STORAGE & PERSISTENT DISKS

What is Cloud Storage & Persistent Disks in GCP?

Google Cloud provides two primary storage solutions:

- **Cloud Storage**: An **object storage** service used for storing **unstructured data**, such as files, images, and backups.
- **Persistent Disks (PDs)**: Block storage that provides **high-performance, durable storage** for **virtual machines (VMs) running on Compute Engine or Kubernetes clusters**.

Key Differences Between Cloud Storage & Persistent Disks

Feature	Cloud Storage	Persistent Disks (PDs)
Type	Object Storage	Block Storage
Use Case	File storage, backups, streaming	VM boot disks, databases, high-performance workloads
Scalability	Infinite scaling	Limited to disk size
Access	Access via API, CLI, UI	Attached to VMs
Performance	Optimized for high durability & availability	Low latency, high-speed IOPS
Persistence	Data remains independent of Compute Engine instances	Data persists even if VM is stopped, but disks must be attached

📌 **Example:**
A **video streaming service** uses **Cloud Storage** for storing and serving media files, while **Persistent Disks** store **database transactions** for real-time processing.

CHAPTER 2: CLOUD STORAGE IN GCP

2.1 Cloud Storage Basics

Google Cloud Storage provides a **highly available, durable, and scalable** solution for storing unstructured data.

2.2 Cloud Storage Key Concepts

✓ **Buckets:** Containers that store objects (files).
✓ **Objects:** Data stored in a bucket.
✓ **Storage Classes:** Determines cost, availability, and durability.
✓ **Access Control:** IAM roles and permissions for data security.
✓ **Encryption:** Data is encrypted at rest and in transit.

2.3 Cloud Storage Classes

Storage Class	Use Case	Durability	Availability
Standard	Frequently accessed data	99.999999999% (11 9s)	99.99%
Nearline	Accessed ~once a month	99.999999999%	99.95%
Coldline	Accessed ~once a year	99.999999999%	99.90%
Archive	Long-term backups	99.999999999%	99.90%

📌 **Example:**
A **data analytics company** uses **Coldline Storage** to store historical data logs, reducing costs while ensuring availability when needed.

2.4 Creating a Cloud Storage Bucket

Step 1: Create a Cloud Storage Bucket via Console

1. Open **Google Cloud Console** → Navigate to **Cloud Storage**.
2. Click **Create Bucket**.
3. Enter a **Bucket Name** (my-cloud-bucket).
4. Choose **Storage Class** (e.g., Standard, Nearline).
5. Set **Location** (e.g., Multi-Region, Regional, or Dual-Region).
6. Click **Create**.

Step 2: Upload & Access Data in Cloud Storage

- **Upload via Console**
    - Click **Upload Files** in the bucket UI.
- **Upload via gcloud CLI**
- gsutil cp local-file.txt gs://my-cloud-bucket/
- **Download from Cloud Storage**
- gsutil cp gs://my-cloud-bucket/remote-file.txt .

📌 **Example:**
A **machine learning (ML) project** stores **training datasets** in **Cloud Storage**, making them accessible to **AI models**.

2.5 Securing Cloud Storage

✓ **IAM Roles & Access Control:** Restrict bucket access using **Viewer, Editor, and Owner** roles.

✓ **Bucket-Level Permissions:** Apply access controls at the bucket level.

✓ **Object-Level ACLs:** Assign **read/write** permissions to individual files.

✓ **Encryption:** Uses **Google-managed keys, Customer-Managed Keys (CMEK), or Customer-Supplied Keys (CSEK)**.

✓ **VPC Service Controls:** Prevents **data exfiltration** from unauthorized networks.

📌 **Example:**
A **healthcare provider** enforces **bucket-level encryption and VPC service controls** to ensure **HIPAA compliance**.

CHAPTER 3: PERSISTENT DISKS IN GCP

3.1 What are Persistent Disks?

Persistent Disks (PDs) are **block storage devices** that provide **high durability and performance** for Google Compute Engine VMs.

3.2 Types of Persistent Disks

Disk Type	Use Case	Max Size	Performance
Standard PD (pd-standard)	Cost-effective, low IOPS workloads	64 TB	Medium
Balanced PD (pd-balanced)	General-purpose workloads	64 TB	High
SSD PD (pd-ssd)	High-performance applications	64 TB	Very High

Extreme PD (pd-extreme)	Enterprise-grade databases	64 TB	Ultra High

📌 **Example:**
A **financial application** uses **pd-ssd** disks to handle **millions of transactions per second**.

3.3 Creating a Persistent Disk and Attaching to a VM

Step 1: Create a Persistent Disk

1. Open **Google Cloud Console** → Go to **Compute Engine**.
2. Click **Disks** → **Create Disk**.
3. Enter **Disk Name** (my-persistent-disk).
4. Choose **Disk Type** (pd-ssd, pd-standard).
5. Set **Size** (e.g., 100GB).
6. Click **Create**.

Step 2: Attach Disk to a VM

1. Open **Compute Engine** → Click **VM Instances**.
2. Select the **VM** → Click **Edit**.
3. Click **Add Disk** → Select the newly created disk.
4. Click **Save**.

Step 3: Format and Mount the Disk

- **Check available disks**
- lsblk
- **Format the disk**

- sudo mkfs.ext4 -F /dev/sdb
- **Mount the disk**
- sudo mkdir /mnt/my-disk
- sudo mount /dev/sdb /mnt/my-disk

📌 **Example:**
A **Kubernetes cluster** attaches **Persistent Disks (PDs)** to ensure **stateful workloads** like **PostgreSQL databases** persist beyond pod restarts.

Chapter 4: Backup, Snapshot, & Disaster Recovery

4.1 Creating a Persistent Disk Snapshot

1. Navigate to **Compute Engine → Snapshots**.
2. Click **Create Snapshot**.
3. Select **Source Disk** → Choose **Storage Location**.
4. Click **Create**.

To automate backups, use **Snapshot Schedules** in Compute Engine.

📌 **Example:**
A **SaaS platform** creates **daily disk snapshots** to prevent **data loss in case of failure**.

Chapter 5: Case Study – Multi-Tier Web Application in GCP

Problem Statement:

A **multi-tier e-commerce application** requires:

- **Fast access to images and customer data**.

- **A database with high availability**.
- **Frequent backups for disaster recovery**.

Solution Implementation:

1. **Used Cloud Storage** for static files like product images.
2. **Deployed Persistent Disk (pd-ssd) with Compute Engine** for the database.
3. **Configured snapshot schedules** for automated backups.
4. **Implemented IAM policies** to restrict access to storage.

Results:

✓ **Reduced database latency by 50%** with SSD PDs.
✓ **Improved disaster recovery readiness** with automated backups.
✓ **Lowered storage costs by 30%** by using **Coldline Storage** for old data.

CHAPTER 6: EXERCISE & REVIEW QUESTIONS

Exercise:

1. **Create a Cloud Storage bucket** and upload a file using gsutil.
2. **Attach a Persistent Disk** to a Compute Engine VM.
3. **Set up an IAM policy** to restrict Cloud Storage access.
4. **Create a snapshot of a Persistent Disk**.

Review Questions:

1. What are the differences between **Cloud Storage and Persistent Disks**?
2. How does **IAM control access to Cloud Storage**?

3. What are the use cases for **pd-ssd vs. pd-balanced**?
4. Why are **Persistent Disk snapshots important** for disaster recovery?

Conclusion: Optimizing Storage in Google Cloud

By **leveraging Cloud Storage for object data and Persistent Disks for block storage**, organizations can **build scalable, secure, and high-performance applications in GCP**. 🚀

CLOUD SQL, BIGQUERY, AND NOSQL DATABASES IN GOOGLE CLOUD

CHAPTER 1: INTRODUCTION TO CLOUD DATABASES IN GOOGLE CLOUD

Understanding Cloud Databases

Google Cloud offers a variety of database solutions that cater to different application requirements, from **relational databases (Cloud SQL, Spanner)** to **NoSQL databases (Firestore, Bigtable, Datastore)** and **data analytics warehouses (BigQuery)**.

Key Database Offerings in Google Cloud

Database	Type	Use Case
Cloud SQL	Relational (SQL)	Web apps, transactional databases
BigQuery	Data Warehouse	Large-scale data analytics
Firestore	NoSQL (Document-based)	Real-time applications, mobile apps
Bigtable	NoSQL (Wide-column)	High-throughput analytics, IoT
Datastore	NoSQL (Document-based)	Scalable application backends

📌 **Example:**
An **e-commerce platform** uses **Cloud SQL for transactions, Firestore for user data,** and **BigQuery for customer analytics**.

CHAPTER 2: CLOUD SQL – MANAGED RELATIONAL DATABASES

2.1 What is Cloud SQL?

Cloud SQL is a **fully managed relational database service** that supports **MySQL, PostgreSQL, and SQL Server**. It handles **automatic backups, replication, and scaling**.

2.2 Setting Up Cloud SQL

Step 1: Create a Cloud SQL Instance

1. Open **Google Cloud Console** → Navigate to **SQL**.
2. Click **+ Create Instance** → Select **MySQL / PostgreSQL / SQL Server**.
3. Set **Instance ID, Region,** and **Machine Type**.
4. Configure **Storage** and **Enable Backups**.
5. Click **Create**.

Step 2: Connect Cloud SQL to an Application

✓ Option 1: Use Public IP

- Retrieve the **Public IP** from **Cloud SQL Overview**.
- Connect using a database client like **DBeaver or MySQL Workbench**.

✓ Option 2: Use Private IP (More Secure)

- Enable **Private IP** in **Networking Settings**.
- Connect using **Cloud SQL Proxy**.

Step 3: Run SQL Queries

```
CREATE TABLE customers (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255),
```

```
    email VARCHAR(255) UNIQUE
);
```

```
INSERT INTO customers (name, email) VALUES ('Alice',
'alice@example.com');
```

📌 **Example:**
A **banking application** uses **Cloud SQL (PostgreSQL)** to **store customer transactions securely**.

2.3 Scaling & High Availability in Cloud SQL

✓ **Read Replicas** – Improve performance for read-heavy workloads.
✓ **Failover Replicas** – Ensure high availability during failures.
✓ **Point-in-Time Recovery (PITR)** – Restore database to a previous state.

CHAPTER 3: BIGQUERY – SERVERLESS DATA WAREHOUSE

3.1 What is BigQuery?

BigQuery is a **fully managed data warehouse** designed for **fast SQL analytics on large datasets**. It supports **petabyte-scale queries with built-in machine learning (BigQuery ML)**.

3.2 Setting Up BigQuery

Step 1: Load Data into BigQuery

1. Open **BigQuery Console** → Click **+ Create Dataset**.
2. Click **+ Create Table** → Choose **Source (Cloud Storage, CSV, JSON, Parquet, etc.)**.

3. Define **Schema (Auto-detect or Manual)** → Click **Create**.

Step 2: Run SQL Queries on BigQuery

```
SELECT customer_id, SUM(total_amount) as revenue
FROM `my_project.ecommerce.transactions`
WHERE transaction_date >= '2023-01-01'
GROUP BY customer_id
ORDER BY revenue DESC
LIMIT 10;
```

📌 **Example:**
An **online retail company** uses **BigQuery** to analyze **sales trends across multiple regions** in real-time.

3.3 BigQuery Best Practices for Performance & Cost Optimization

✓ **Partitioned & Clustered Tables** – Optimize query performance.
✓ **Use Approximate Aggregation Functions** – Reduce cost.
✓ **Enable Streaming Inserts** – Load data in real-time.
✓ **Schedule Queries** – Automate analytics tasks.

📌 **Example:**
A **marketing agency** uses **scheduled BigQuery queries** to generate **daily campaign performance reports**.

CHAPTER 4: NOSQL DATABASES – FIRESTORE, BIGTABLE, & DATASTORE

4.1 Firestore – NoSQL Document Database

Firestore is a **real-time NoSQL database** ideal for **web, mobile, and IoT applications**. It offers:

✓ **Flexible Schema** – Stores data in **JSON-like documents**.
✓ **Offline Sync** – Works even when the user is offline.
✓ **Auto-scaling** – Handles millions of reads/writes per second.

Step 1: Store & Retrieve Firestore Data in Python

```
from google.cloud import firestore


db = firestore.Client()

doc_ref = db.collection('users').document('alice')


# Store Data

doc_ref.set({'name': 'Alice', 'email': 'alice@example.com'})


# Retrieve Data

doc = doc_ref.get()

print(f"User Data: {doc.to_dict()}")
```

📌 **Example:**
A **chat application** stores **user messages** in **Firestore for real-time updates**.

4.2 Bigtable – NoSQL Wide-Column Store for High Throughput

Bigtable is a **NoSQL database optimized for high-throughput applications**, such as:

✓ **IoT Data Processing**

✓ **Time-Series Data Storage**

✓ **Fraud Detection & Recommendation Systems**

Step 1: Write & Read Data in Bigtable (Python Example)

```
from google.cloud import bigtable

client = bigtable.Client()
instance = client.instance('my-bigtable-instance')
table = instance.table('sensor-data')

# Write Data
row_key = b'sensor123'
row = table.row(row_key)
row.set_cell('temperature', 'value', 22.5)
row.commit()

# Read Data
row_data = table.read_row(row_key)
print(f"Sensor Value:
{row_data.cells['temperature']['value'][0].value}")
```

📌 **Example:**
A **smart home system** uses **Bigtable** to store **sensor temperature data for millions of devices**.

4.3 Datastore – NoSQL Key-Value Database

Datastore is a **NoSQL database for application backends,** offering:

✓ Schema-less, Key-Value Storage
✓ Automatic Indexing
✓ Strong & Eventual Consistency Options

Step 1: Store & Query Data in Datastore

```
from google.cloud import datastore


client = datastore.Client()

task_key = client.key('Task', 'task123')


# Store Data

task = datastore.Entity(task_key)

task.update({'title': 'Deploy Cloud App', 'status': 'Pending'})

client.put(task)


# Query Data

query = client.query(kind='Task')

tasks = list(query.fetch())

print(f"Task: {tasks[0]['title']} - Status: {tasks[0]['status']}")
```

📌 **Example:**
A **to-do list app** stores **tasks** in **Datastore** for fast retrieval and indexing.

Chapter 5: Choosing the Right Database for Your Use Case

Use Case	Recommended Database
Transactional applications (OLTP)	Cloud SQL (MySQL/PostgreSQL)
Real-time analytics	BigQuery
Mobile apps & chat applications	Firestore
IoT & Time-Series Data	Bigtable
Scalable application backends	Datastore

📌 **Example:**
A **logistics company** uses **Cloud SQL for order management, BigQuery for delivery analytics,** and **Firestore for real-time fleet tracking**.

Chapter 6: Review Questions & Exercises

Exercise:

1. **Create a Cloud SQL instance** and connect using a client.
2. **Load a dataset into BigQuery** and run a SQL query.
3. **Store & retrieve data in Firestore using Python**.
4. **Use Bigtable to write & query IoT sensor data**.

Review Questions:

1. How does **Cloud SQL differ from BigQuery**?
2. What are the advantages of **Firestore over traditional relational databases**?

3. When should you use **Bigtable instead of Datastore**?
4. How can **partitioning improve BigQuery performance**?

Conclusion: Optimizing Cloud Databases for Scalability & Performance

Google Cloud provides **SQL & NoSQL solutions** tailored for **transactional, analytical, and real-time applications**. Choosing the right **database service** ensures **scalability, security, and efficiency** in cloud applications. 🚀

ASSIGNMENT

SET UP A GCP ACCOUNT AND CONFIGURE IAM ROLES

SOLUTION: SET UP A GCP ACCOUNT AND CONFIGURE IAM ROLES

This step-by-step guide will help you create a **Google Cloud Platform (GCP) account**, set up a **Google Cloud project**, and configure **Identity and Access Management (IAM) roles** to manage permissions efficiently.

Step 1: Create a Google Cloud Platform (GCP) Account

1.1 Sign Up for GCP

1. **Go to Google Cloud Console**: https://console.cloud.google.com/
2. Click **Get Started for Free**.
3. Sign in using your **Google account**.
4. **Enter billing details** (Google provides a **$300 free credit** for new users).

📌 **Example:**
A startup wants to **deploy its application on GCP** and signs up to take advantage of the **free trial credits**.

Step 2: Create a New GCP Project

2.1 Steps to Create a GCP Project

1. Open **Google Cloud Console**.
2. Click the **Project Selector** (top navigation bar).
3. Click **New Project** → Enter:

- **Project Name**: MyFirstGCPProject
- **Billing Account**: Select an existing or new billing account.
- **Organization**: Select an organization (if applicable).

4. Click **Create** and wait for deployment.

📌 **Example:**
A company creates a **GCP project named EcommerceApp** to host its **shopping website** using **Google App Engine**.

Step 3: Configure IAM (Identity and Access Management) Roles

3.1 Understanding IAM Roles in GCP

Role Type	Description
Owner	Full control over resources, billing, and permissions
Editor	Can create, update, and delete resources (except IAM & billing)
Viewer	Read-only access to resources
Custom Roles	Define permissions tailored to specific needs

📌 **Example:**
A **DevOps team** assigns the **Editor role** to developers, so they can manage resources but **not alter IAM settings**.

Step 4: Assign IAM Roles to Users

4.1 Steps to Assign IAM Roles

1. **Go to IAM & Admin** in GCP Console.
2. Click **IAM** → **Select Project**.
3. Click **+ Add** to assign a new user.
4. Enter **User Email** (e.g., developer@company.com).
5. Click **Select a Role** → Choose **Editor** (or another role).
6. Click **Save** to apply changes.

📌 **Example:**
A **cloud administrator** assigns the **Editor role** to developers and the **Viewer role** to the finance team for **cost monitoring**.

Step 5: Create Custom IAM Roles (Optional)

5.1 Steps to Create a Custom Role

1. Open **IAM & Admin** → Click **Roles**.
2. Click **Create Role** → Enter:
   - **Title**: GCP DevOps Engineer
   - **Description**: Custom role with deployment permissions
3. Click **Add Permissions** → Select:
   - compute.instances.create (Create VM instances)
   - compute.instances.start (Start instances)
   - compute.instances.stop (Stop instances)
4. Click **Create** and assign the role to users.

📌 **Example:**
A **startup** creates a **custom DevOps role** to allow **developers to manage compute instances** but restrict access to **billing settings**.

Step 6: Enable Multi-Factor Authentication (MFA) for Security

6.1 Steps to Enable MFA

1. Open **Google Admin Console** (https://admin.google.com/).
2. Click **Security** → **Set up 2-Step Verification**.
3. Enforce MFA for all users.

📌 **Example:**
A **financial institution** enforces **MFA on all IAM users** to **enhance security** for sensitive cloud resources.

Step 7: Monitor & Audit IAM Permissions

7.1 Steps to Review IAM Activity Logs

1. Open **Google Cloud Console** → Click **IAM & Admin**.
2. Navigate to **Audit Logs**.
3. Filter by **IAM Activity** to track permission changes.

📌 **Example:**
A **security team** regularly checks IAM logs to **detect unauthorized role changes**.

Step 8: Exercise & Review Questions

Exercise:

1. **Create a GCP Project** and assign a role to a user.
2. **Create a Custom IAM Role** and apply it to a team member.
3. **Enable Multi-Factor Authentication** for security.

Review Questions:

1. What are the **three default IAM roles** in GCP?
2. How can you **restrict billing access** for a GCP user?
3. What is the purpose of **Custom IAM Roles**?
4. Why is **MFA important in IAM security**?
5. How can you **monitor IAM activity logs**?

Conclusion: Securely Managing IAM Roles in GCP

Setting up **IAM roles in GCP** ensures **proper access control, security, and compliance**. By **assigning appropriate roles, enabling MFA, and monitoring logs**, organizations can **protect cloud resources while optimizing productivity**. 🚀

DEPLOY A VIRTUAL MACHINE USING COMPUTE ENGINE

SOLUTION: DEPLOY A VIRTUAL MACHINE USING COMPUTE ENGINE IN GOOGLE CLOUD PLATFORM (GCP)

This guide provides a **step-by-step approach** to deploying a **Virtual Machine (VM) using Google Compute Engine (GCE)** in **Google Cloud Platform (GCP)**.

Step 1: Set Up GCP Environment

1.1 Prerequisites

✓ **Google Cloud Account** – Sign up at Google Cloud Console.

✓ **Enable Compute Engine API** – Go to **APIs & Services → Enable Compute Engine API**.

✓ **Install Google Cloud SDK (CLI) (Optional)** – For CLI-based deployment:

```
curl https://sdk.cloud.google.com | bash

gcloud init
```

Step 2: Deploy a Virtual Machine Using Google Cloud Console

2.1 Create a Virtual Machine (VM)

1. **Go to Compute Engine**
    - Open **Google Cloud Console → Navigation Menu → Compute Engine → VM Instances**.
2. **Click "Create Instance"**
    - Enter **Instance Name** (e.g., my-vm-instance).

- Choose **Region & Zone** (e.g., us-central1-a).

3. **Choose Machine Type**
   - **General-purpose**: e2-medium (2 vCPUs, 4GB RAM) for basic workloads.
   - **Memory-optimized**: m2-ultramem (For memory-intensive applications).
   - **Compute-optimized**: c2-standard (For high-performance computing).
4. **Select Boot Disk**
   - Click **Change** → Choose **Operating System**:
     - **Ubuntu 22.04 LTS**
     - **Debian 11**
     - **Windows Server 2019**
   - Choose **Disk Type**: Standard Persistent Disk (for cost-effective storage) or SSD (for performance).
5. **Configure Firewall Rules**
   - Check **Allow HTTP & HTTPS traffic** (for web applications).
   - Customize **Network & Security Settings** (e.g., assign **VPC network & Subnet**).
6. **Click "Create"**
   - The VM will be deployed in a few minutes.

📌 **Example:**
A **developer deploys an Ubuntu VM** to host a **Python Flask web application**.

Step 3: Deploy a Virtual Machine Using Google Cloud CLI

Alternatively, use the **gcloud CLI** to create a VM.

1. **Run the Following Command:**

```
gcloud compute instances create my-vm-instance \
  --zone=us-central1-a \
  --machine-type=e2-medium \
  --image-family=debian-11 \
  --image-project=debian-cloud \
  --boot-disk-size=20GB \
  --tags=http-server,https-server \
  --metadata=startup-script="echo Hello, World > /var/www/html/index.html"
```

2. **Verify the Instance is Running:**
3. gcloud compute instances list

📌 **Example:**
A **startup automates VM creation** using a **Bash script in gcloud CLI** for DevOps workflows.

Step 4: Connect to the Virtual Machine

4.1 SSH into the VM (Linux/Ubuntu Instances)

1. Open **Google Cloud Console** → **Compute Engine** → Click on the instance.
2. Click **SSH** (Opens a browser-based terminal).

3. Alternatively, connect via **gcloud CLI**:
4. gcloud compute ssh my-vm-instance --zone=us-central1-a

📌 **Example:**
A **developer SSHs into a Debian VM** to install **Docker and deploy a containerized application**.

4.2 RDP into a Windows VM

1. Go to **Compute Engine** → Click on the instance.
2. Click **Set Windows Password** → Save credentials.
3. Use **Remote Desktop Protocol (RDP)** to log in.

📌 **Example:**
A **Windows Server VM** is used for **running SQL Server for enterprise applications**.

Step 5: Install & Configure Software on VM

Once connected, install necessary applications:

✓ Install Web Server on Ubuntu VM

sudo apt update && sudo apt install apache2 -y

sudo systemctl enable apache2

sudo systemctl start apache2

✓ Install MySQL Database

sudo apt update

sudo apt install mysql-server -y

sudo systemctl enable mysql

sudo systemctl start mysql

✓ Install Node.js & Express.js for Web API

curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -

sudo apt install -y nodejs

📌 **Example:**
A **backend team installs Node.js on a GCE instance** to build an API for a **ride-sharing application**.

Step 6: Configure Networking & Firewall for External Access

6.1 Allow Traffic Using gcloud CLI

gcloud compute firewall-rules create allow-http \

--allow tcp:80 \

--target-tags=http-server

6.2 Get External IP & Test Access

gcloud compute instances describe my-vm-instance --format='get(networkInterfaces[0].accessConfigs[0].natIP)'

✓ Open a browser and access:

http://<external-ip>

📌 **Example:**
A **web development team** configures **firewall rules** to allow HTTP(S) traffic for a **React-based web app**.

Step 7: Managing VM Instances

7.1 Restart, Stop, or Delete VM

✓ **Stop the VM**:

gcloud compute instances stop my-vm-instance --zone=us-central1-a

✓ **Start the VM**:

gcloud compute instances start my-vm-instance --zone=us-central1-a

✓ **Delete the VM**:

gcloud compute instances delete my-vm-instance --zone=us-central1-a

📌 **Example:**
A **data analyst shuts down an unused VM** to reduce **cloud billing costs**.

Step 8: Monitoring & Logging with Stackdriver (Cloud Operations)

1. **Enable Cloud Monitoring & Logging**:
2. gcloud services enable monitoring.googleapis.com logging.googleapis.com
3. **View Logs in Cloud Console**:
    - Navigate to **Operations > Logging** to analyze logs.
    - Set alerts for **high CPU/memory usage**.

📌 **Example:**
An **e-commerce company** monitors **VM resource usage** to **scale resources dynamically**.

Step 9: Automating VM Deployment with Terraform

For Infrastructure as Code (IaC), deploy a VM using **Terraform**:

1. **Create a Terraform Configuration File (main.tf)**:

```
provider "google" {
  project = "my-gcp-project"
  region  = "us-central1"
}

resource "google_compute_instance" "vm_instance" {
  name         = "terraform-vm"
  machine_type = "e2-medium"
        zone         = "us-central1-a"

        boot_disk {
  initialize_params {
        image = "debian-cloud/debian-11"
  }
        }

        network_interface {
         network = "default"
```

```
    access_config {}
  }
}
```

Deploy with Terraform:

```
terraform init
terraform apply -auto-approve
```

📌 **Example:**
A **DevOps team uses Terraform** to automate VM deployments for **continuous integration environments**.

Case Study: Deploying a Scalable Web Application on GCE

Problem Statement:

A **tech startup** needs to deploy a **scalable web application** using **Compute Engine**.

Solution Implementation:

✓ **Deployed multiple VM instances** with **load balancing**.
✓ **Installed Nginx & Node.js** to serve API requests.
✓ **Configured auto-scaling** to handle high traffic.
✓ **Enabled Cloud Monitoring** for performance tracking.

Results:

✓ **Reduced downtime by 95%**.
✓ **Auto-scaled from 2 to 10 VMs** during peak traffic.
✓ **Decreased operational costs by 30%** using optimal VM sizing.

CONCLUSION

✓ **Compute Engine provides powerful, flexible, and scalable VMs for cloud workloads**.

✓ **Use CLI, Terraform, or Cloud Console for VM management**.

✓ **Monitor & optimize VMs to ensure cost efficiency and performance**.

🚀 **Next Steps:**

✓ Deploy a **Compute Engine instance**.

✓ Automate infrastructure with **Terraform**.

✓ Secure and optimize your VM for production use! 🚀