GCP Project Demonstration: Resume Portfolio Hosting on Google Cloud Platform

1. Project Summary

This project demonstrates hosting a personal resume portfolio web application using Google Cloud Platform (GCP).
The application includes a static HTML/CSS/JS frontend hosted on Google Cloud Storage and a backend API hosted on Google Compute Engine.
This prototype showcases my familiarity with GCP services and how they can be used to deploy scalable web solutions.

2. GCP Services Used

 Google Cloud Storage: Used to host static website content (HTML, CSS, JS) and store documents such as resumes and certificates in a structured folder system.

 Google Compute Engine: Used to run a basic Node.js backend server and act as a connection point for cloud-based operations.

 Google Cloud IAM: Used to manage access permissions and securely control who can access storage buckets and compute resources.

 Google Cloud Console: Used to monitor resource usage, manage configurations, and debug issues across GCP services.

3. Architecture Diagram (Description)

The architecture is straightforward:

- A static website is hosted in a public GCP Storage bucket.
- The backend API is deployed on a virtual machine using Compute Engine.
- The frontend communicates with the backend via HTTP requests.
- I used **Google Cloud Storage** buckets not only for the static website but also to **store and organize documents** like resumes, certificates, and project folders.
- The **connection between the VM and Storage Bucket** was established through GCP's internal access settings.

- I accessed the **VM via SSH terminal**, transferred folders into it, and then connected the VM as a **remote host in VS Code** to edit and manage files seamlessly.

4. Steps to Deploy

1. Create a Google Cloud Project.
2. Enable billing and required APIs (Compute Engine, Cloud Storage).
3. Create and configure a Cloud Storage bucket with public access to host static files.
4. Upload HTML/CSS/JS files to the bucket.
5. Create a Compute Engine VM instance.
6. SSH into the VM and install Node.js.
7. Upload or clone the backend server code.
8. Run the server and allow required firewall rules for port 3000.
9. Test connectivity between frontend and backend.

5. Outcome

Successfully hosted a personal resume web app using GCP. The app runs a frontend hosted on Cloud Storage and a backend API on Compute Engine, demonstrating basic understanding of cloud deployment, permissions, and hosting architecture.

6. Notes

This project was implemented as a self-learning initiative during my internship period to understand cloud deployment concepts using Google Cloud Platform. Since this was not a client project, I am unable to provide organization-level deployment or repo access.

Understanding the Working Principles of Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, YouTube, and Google Maps. It provides a vast array of services, enabling individuals and organizations to build, deploy, and scale applications and services with high reliability and efficiency.

The core working principle of GCP revolves around **providing scalable, secure, and globally distributed infrastructure and services as a utility**. Instead of managing

physical servers, developers and businesses consume computing, storage, networking, and specialized services on demand, paying only for what they use.

Here’s a breakdown of its key components and how they interact:

1. Core Service Categories & Their Functionality:

- **Compute Services (The "Engines"):** These services provide the processing power for applications.
    - **Compute Engine (IaaS - Infrastructure as a Service):** Offers virtual machines (VMs) that give users granular control over the operating system, machine type, and software stack. This is like renting a server in Google's data center.
    - **App Engine (PaaS - Platform as a Service):** A fully managed platform for building and deploying web applications and mobile backends. Users deploy code, and GCP handles the underlying infrastructure, scaling, and maintenance.
    - **Cloud Run (Serverless Container Platform):** Allows deployment of stateless containers that are automatically scaled up or down to zero, ideal for microservices and web hooks.
    - **Cloud Functions (FaaS - Function as a Service):** An event-driven serverless compute platform. Developers write small, single-purpose functions that execute in response to events (e.g., file uploads, database changes, HTTP requests) without managing servers.
    - **Google Kubernetes Engine (GKE):** A managed service for deploying, managing, and scaling containerized applications using Kubernetes. It automates container orchestration, making it ideal for microservices architectures.
- **Storage Services (The "Data Houses"):** These store various types of data securely and scalably.
    - **Cloud Storage:** Object storage for unstructured data (files, images, videos, backups). Highly scalable and accessible globally.
    - **Cloud SQL:** Managed relational database service for MySQL, PostgreSQL, and SQL Server. GCP handles patching, backups, and replication.

    - **Cloud Spanner:** A globally distributed, strong consistent relational database service for mission-critical applications.
    - **BigQuery:** A fully managed, serverless data warehouse for large-scale data analytics, allowing petabyte-scale queries.
    - **Firestore/Cloud Datastore:** NoSQL document databases for flexible, scalable application data.
- **Networking Services (The "Connectors"):** Facilitate secure and efficient communication.
    - **Virtual Private Cloud (VPC):** Provides a global private network for your GCP resources, enabling secure and isolated communication.
    - **Cloud Load Balancing:** Distributes incoming traffic across multiple instances to ensure high availability and scalability.
    - **Cloud CDN:** Content Delivery Network to cache web content closer to users, reducing latency.
- **Big Data & Analytics Services (The "Insights Generators"):** For processing and analyzing massive datasets.
    - **Cloud Dataflow:** A fully managed service for executing Apache Beam pipelines for batch and stream data processing.
    - **Cloud Pub/Sub:** A real-time messaging service for ingesting and delivering data streams.
- **Machine Learning (The "Intelligent Services"):** Pre-trained APIs and custom model development.
    - **AI Platform:** For building, deploying, and managing ML models.
    - **Pre-trained APIs:** Services like Cloud Vision API, Natural Language API, Translation API for common AI tasks.

2. How Services Integrate and Work Together:

The power of GCP lies in its **seamless integration**. Applications are typically built by combining multiple services in a modular fashion:

- A web application might be deployed on **App Engine** or **Cloud Run**, storing its data in **Cloud SQL** (for structured data) and static assets in **Cloud Storage**.

- User actions or data updates can trigger **Cloud Functions** via **Cloud Pub/Sub** to perform background processing or send notifications.
- Log data from all services is automatically streamed to **Cloud Logging** and can be analyzed in **BigQuery** for operational insights.
- Frontend applications hosted on **Cloud Storage** can communicate with backend **Cloud Functions** or **GKE**-hosted microservices via **API Gateway**.
- Data pipelines can ingest real-time data via **Cloud Pub/Sub**, process it with **Cloud Dataflow**, and store it in **BigQuery** for analytics or **Cloud Storage** for data lakes.

3. Global Infrastructure and Reliability:

GCP operates on a global network of data centers, organized into **regions** (geographical areas) and **zones** (isolated locations within a region). This architecture allows for:

- **High Availability:** Deploying resources across multiple zones within a region ensures applications remain available even if one zone experiences an outage.
- **Disaster Recovery:** Multi-region deployments provide resilience against regional failures.
- **Low Latency:** Users can access services from regions closest to them.

4. Security and Management:

GCP is built with security at its core.

- **Identity and Access Management (IAM):** Granular control over who can do what on your GCP resources.
- **Cloud Monitoring:** Provides visibility into application performance and health.
- **Cloud Logging:** Centralized logging for all GCP resources.
- **Billing:** Pay-as-you-go model with detailed usage reporting.