Google Compute Engine Use Case

Client Project: Deploying Scalable Web Application Infrastructure with Google Compute Engine (GCE)

GENERAL CHARACTERISTICS

Intent	To build a scalable, highly available web application infrastructure for a client using Google Compute Engine.
Scope	Deployment and management of web applications on Google Cloud Platform with components such as Virtual Machines, Load Balancers, Autoscaling, and Networking.
Level	System-level.
Client	Confidential (E-Commerce Company).
Last Update	[Today's Date]
Status	Finalized.
Stage	Execution and Monitoring.

ACTORS

Primary Actor	DevOps Engineer.
Secondary Actors	Solution Architect, Application Developer, Database Administrator, and Client's IT Operations Team.

PREREQUISITES

Static Preconditions	- Google Cloud Platform (GCP) project created for the client. - Billing account linked to the client's project. - APIs for Compute Engine, Load Balancer, and Cloud Monitoring enabled.
Dynamic Preconditions	- Cloud Storage buckets set up for static asset storage. - Firewall rules implemented for required ports. - Database instance configured (Cloud SQL).
Assumptions	- The application can be containerized or deployed on Linux VMs. - Scaling policies provided by the client.

TRIGGERS

Trigger Event	The client faced high traffic demands during promotional events, requiring a

	robust and scalable infrastructure.

EXPECTED OUTCOME

Success Postcondition	- Client's application handles traffic spikes without downtime. - Infrastructure scales dynamically to optimize costs.
Failed Postcondition	- Traffic exceeds capacity, resulting in downtime or degraded performance.

OPERATIONS AND CONCEPTS

Operations	1. Designed and implemented a VM instance template tailored for the client's web application. 2. Set up a Managed Instance Group to handle scaling and redundancy. 3. Configured an HTTP(S) Load Balancer to ensure smooth traffic distribution. 4. Implemented autoscaling policies based on CPU utilization and traffic metrics. 5. Hosted static assets on Google Cloud Storage for efficient delivery. 6. Connected VMs to a Cloud SQL instance for backend data management. 7. Enabled comprehensive monitoring with Google Cloud Monitoring and Logging.
Concepts	- Load Balancer: Ensures high availability and efficient traffic distribution. - Managed Instance Groups: Facilitates autoscaling and redundancy. - Autoscaling: Adjusts resources dynamically based on traffic. - Monitoring: Provides visibility into system performance.

MAIN SUCCESS SCENARIO

Step 1	Analyzed client requirements and designed a GCP architecture tailored for scalability.
Step 2	Created a VM instance template optimized for the client's web application.
Step 3	Deployed Managed Instance Groups to ensure redundancy and autoscaling.
Step 4	Configured a Load Balancer to manage incoming traffic.
Step 5	Implemented autoscaling to dynamically adjust resources.
Step 6	Connected the infrastructure to Cloud SQL

	for database needs.
Step 7	Enabled Cloud Monitoring and Logging for real-time performance tracking.
Step 8	Delivered a scalable and reliable infrastructure ready for high-traffic events.