Cloud Engineer Exam

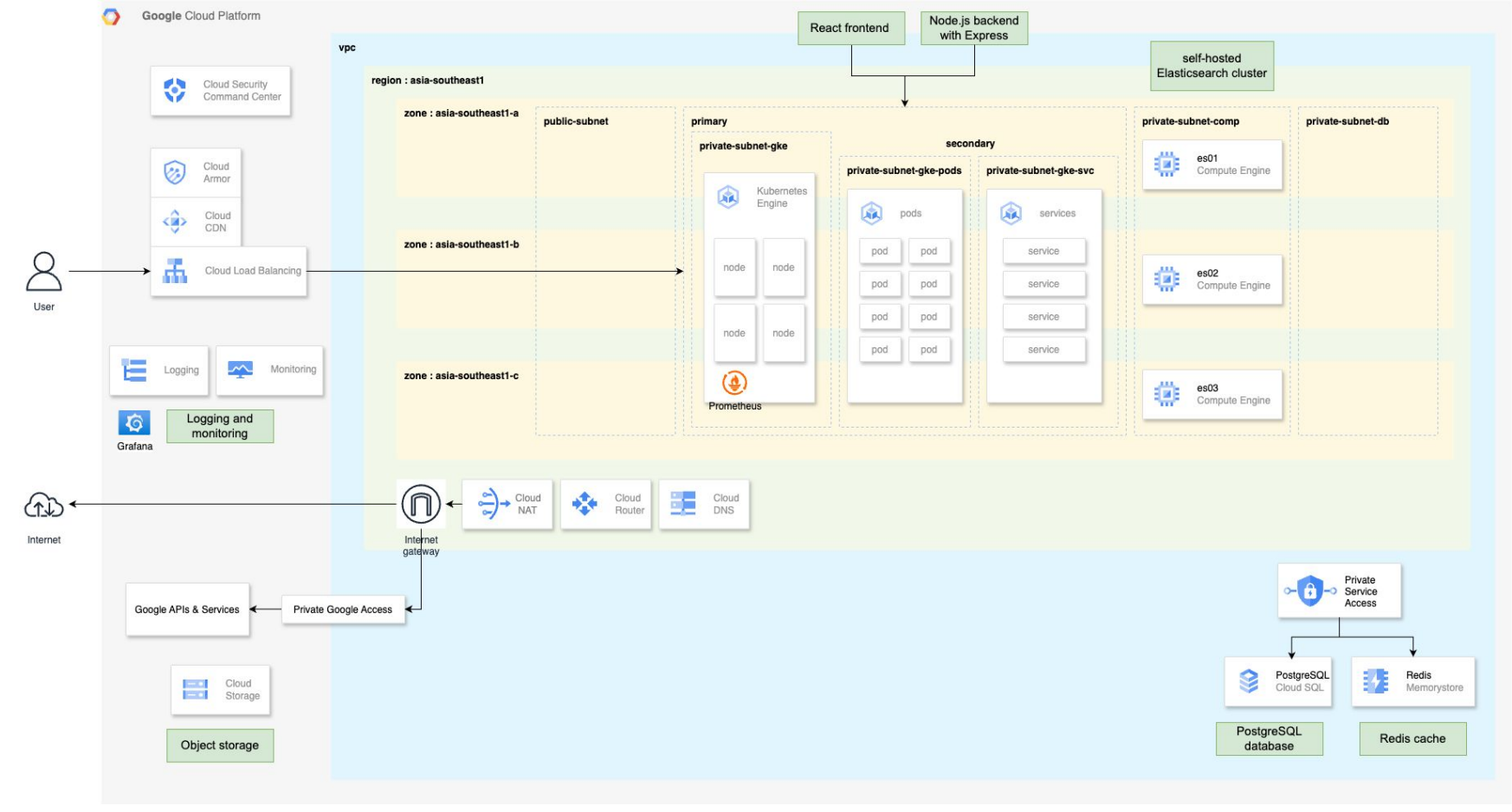
Part 1: Architecture Design

A company is building a new web application that serves users globally. The application consists of the following components:

- A React frontend
- A Node.js backend with Express
- A PostgreSQL database
- A Redis cache
- A self-hosted Elasticsearch cluster
- Object storage for user uploads
- Logging and monitoring requirements

The company wants to host the application on a cloud provider (One of the following: AWS, GCP, or Azure). The architecture should be highly available, scalable, and cost effective.

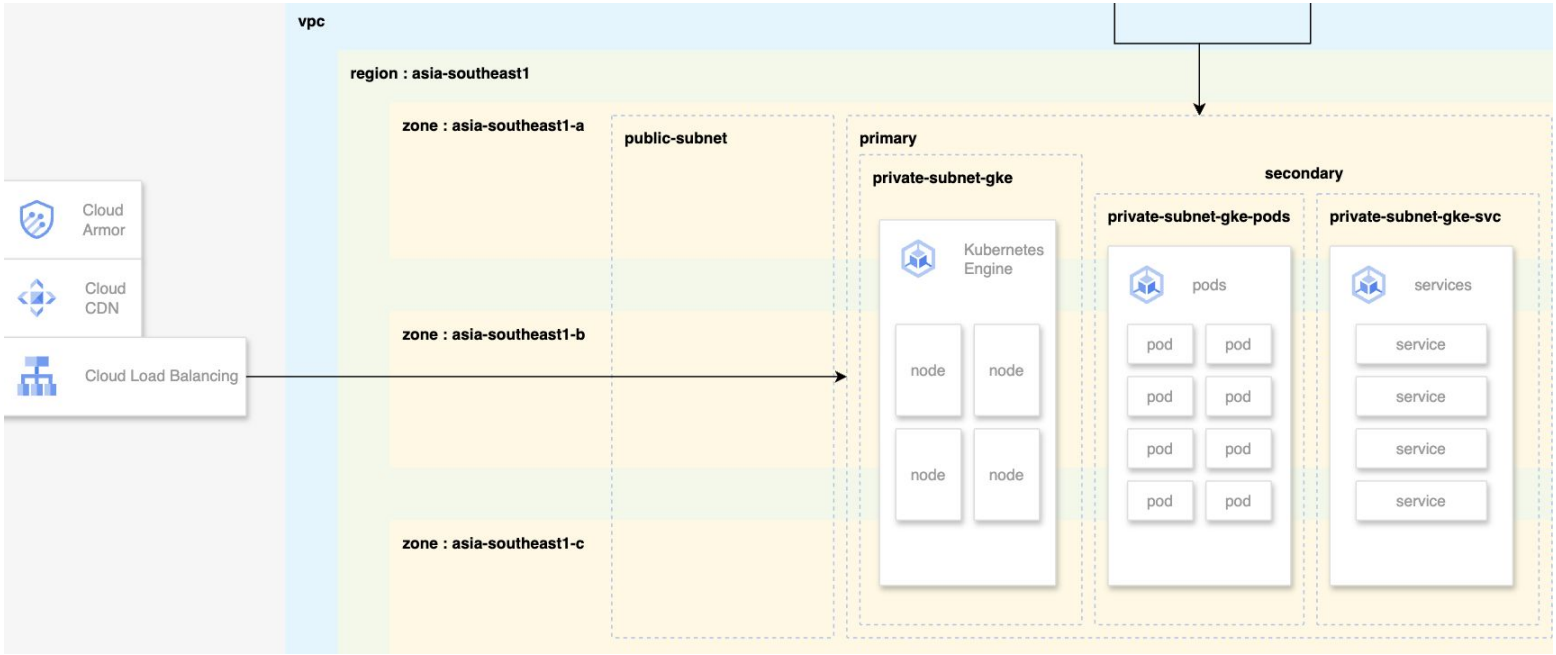
Task 1 : Draw an architecture diagram and describe how the application components will be deployed in the cloud.



Task 2 : Explain how you will ensure high availability and scalability for each component.

Frontend (React) , Backend (Node.js with Express)

- Deploy behind **Google Cloud Load Balancing** to distribute requests evenly.
- Host static files using **Cloud CDN** to scale efficiently and serve users globally with minimal latency.
- Deploy on **Google Kubernetes Engine (GKE)** with 3 zones, auto-scaling enabled based on CPU/memory utilization or request rate.
- Use **horizontal pod autoscaler (HPA)** to dynamically adjust pod instances based on traffic.



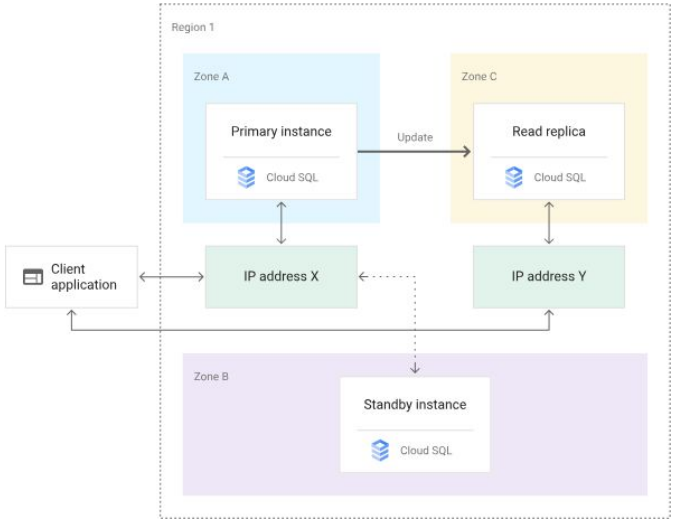
Task 2 : Explain how you will ensure high availability and scalability for each component.

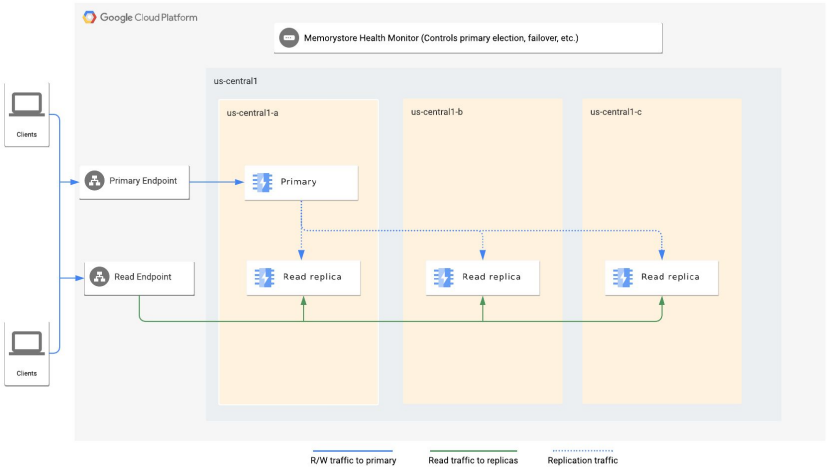
PostgreSQL Database

- Use **Google Cloud SQL** (PostgreSQL) with **high availability enabled** (automatic failover to a standby instance) across different zones.
- Enable auto-storage scaling and use read-replicas for horizontal scalability.

Redis cache

- Use **Google Memorystore for Redis** with **high-availability enabled** across multiple zones.
- with **replication** and automatic failover.





Task 2 : Explain how you will ensure high availability and scalability for each component.

Self-hosted Elasticsearch cluster

- Deploy **Compute engine** across at least **3 VMs** in separate zones.
- Configure Elasticsearch nodes as part of **Elasticsearch cluster**
- scale by adding more Elasticsearch nodes (VMs) as demand increases.

Object Storage (Google Managed)

- Use **Google Cloud Storage** (regional bucket) to store and deliver user-uploaded content.
- Scales automatically without limits.

Task 3 : Describe the security measures you will implement, including network security, data security, and access control.

Network Security:

- **VPC and Subnet Isolation:** Separate public and private subnets to isolate resources.
- **Cloud NAT:** Enable secure outbound internet access for private subnet resources without exposing external IPs.
- **Private Google Access:** Allow private VMs to securely communicate with Google APIs and services without internet exposure.
- **Private Service Access:** Establish private connectivity between the VPC and managed services (e.g., Cloud SQL, Memorystore) to eliminate public network exposure.
- **SSH with Identity-Aware Proxy (IAP):** Restrict SSH access to virtual machines via Google Cloud's IAP, removing public IP dependencies and reducing attack surfaces.

Data Security:

- **Encryption at Rest :** Encrypt all stored data in Google Cloud Storage, Cloud SQL (PostgreSQL), and Redis using Google-managed encryption keys.
- **Encryption in Transit**: Enforce SSL/TLS encryption for all data transfers

Access Control:

- **Identity and Access Management (IAM):** Service accounts for **Elasticsearch** with minimum required permissions
- **Restricted SSH Access:** Allow SSH access solely through IAP to ensure encrypted connections, auditing, and reduced external exposure.

Task 4 : Propose a logging and monitoring solution, specifying which tools you will use and how logs/metrics will be collected.

1. Logging Solution

- **Application Logs**:
  - Use **Elasticsearch** for centralized log storage and searching.
  - Deploy **Fluent Bit** as a **DaemonSet** in GKE to collect logs from containers and nodes.
  - Forward logs from **Fluent Bit** to **Elasticsearch** for indexing and analysis.
- **Infrastructure Logs**:
  - Enable **Google Cloud Logging** to capture logs from **Google Cloud services**
- **Log Storage and Retention**:
  - Store short-term logs in **Google Cloud Logging**.
  - Archive long-term logs in **Cloud Storage** and Elasticsearch with lifecycle policies.

2. Monitoring Solution

- **Grafana Monitoring**:
  - Monitor CPU, memory, disk usage, and network traffic for **VM instances**, **PostgreSQL**, and **Redis**.
  - Integrate **Prometheus with Grafana** to visualize application and infrastructure metrics.
  - Set up **uptime checks** for the backend and frontend endpoints.
  - Configure **alerting policies** for high error rates, latency spikes, and service downtime.
- **Elasticsearch Monitoring**:
  - Use **Kibana** to visualize application logs.
  - Implement **Elasticsearch X-Pack Monitoring** for cluster health checks.

Task 5 : Estimate the monthly cost of running this architecture and suggest optimizations for cost savings.

Estimated Monthly Cost (USD):

- **GKE** (1 cluster, 3 nodes, n4-standard-4) : ~$615/month
- **Cloud SQL**(PostgreSQL HA, dbdb-standard-4) : ~$640/month
- **Memorystore for Redis** (Standard, 5GB, 3 read replicas) : ~$585/month
- **Cloud Storage** (standard class, ~2TB storage) : ~$55/month
- **Compute Engine** (Elasticsearch self-hosted 3 nodes, n4-standard-4) : ~$660/month
- **Networking & Load Balancing** : ~$50/month
- **Cloud Logging** : ~$230/month

Estimated Total Cost: Approximately **~$2,835/month**

Optimizations for Cost Savings:

- **Rightsize Instances:** Regularly monitor and adjust instance types to match actual resource utilization.
- **Committed-use Discounts:** Leverage committed usage discounts for predictable workloads (1 or 3-year commitments save ~20-50%).
- **Cloud Storage Lifecycle Management:** Use lifecycle rules to transition less frequently accessed data to cheaper storage tiers.
- **Optimize Log Storage:** Use Log Routing to Store Logs in Cloud Storage Instead of Cloud Logging

Cost details

Part 2: Infrastructure as Code (IaC)

Based on the architecture design in Part 1, write Infrastructure as Code (IaC) to provision the cloud infrastructure. Use Terraform or AWS CloudFormation (or another equivalent tool).

Task

1. Write a Terraform/CloudFormation script to provision the following:
    a. A PostgreSQL database with high availability
    b. A Redis cache
    c. A self-hosted Elasticsearch cluster
    d. Object storage (e.g., S3, Google Cloud Storage, or Azure Blob Storage)
2. Define IAM roles and security groups to enforce least privilege access.
3. Implement autoscaling for the backend API.
4. Use Terraform modules or reusable components to structure your IaC efficiently

Terraform : https://github.com/chin-js/cloud_engineer_exam

Part 3: Elasticsearch Configuration

The application requires a self-hosted Elasticsearch cluster to handle search queries efficiently. You need to set up and configure this cluster using automation tools like Shell Scripts, Ansible, or another configuration management tool.

Task

1. Write a script (Shell, Ansible, or another tool) to:

    a. Install and configure an Elasticsearch cluster with multiple nodes

    b. Set up cluster settings for high availability and performance

    c. Secure the cluster with authentication and encryption

2. Configure monitoring for Elasticsearch, including:

    a. Node health checks

    b. Performance metrics collection

    c. Alerting setup

Ansible : https://github.com/chin-js/cloud_engineer_exam/tree/main/ansible-elk