**GKE Infrastructure Deployment with Terraform & Helm**

**Overview and Purpose of the Solution**

This project provisions a Google Kubernetes Engine (GKE) cluster using Terraform, and deploys a full monitoring stack (Prometheus & Grafana) alongside a 3-tier application (frontend, backend, database) into a Kubernetes cluster. The solution is modular, supports multiple environments, and follows infrastructure-as-code best practices.

**How to Run Terraform and Helm Commands (Step-by-Step)**

**Prerequisites:**

* Terraform >= 1.3
* GCP project & service account with required permissions
* Google Cloud SDK (gcloud)
* kubectl & helm installed

**1. Authenticate to GCP and Set Up Terraform**

export GOOGLE\_APPLICATION\_CREDENTIALS="<path-to-service-account>.json"

gcloud auth activate-service-account --key-file $GOOGLE\_APPLICATION\_CREDENTIALS

gcloud auth application-default login

**2. Initialize Terraform**

cd tf-gke-project

terraform init

**3. Plan Infrastructure**

terraform plan

**4. Apply Infrastructure**

terraform apply -auto-approve

**5. Get GKE Credentials and Verify**

gcloud container clusters get-credentials <CLUSTER\_NAME> --region <REGION> --project <PROJECT\_ID>

kubectl get nodes

**How TLS Certificates Were Created and Used**

**Generate Self-Signed TLS Certs**

mkdir certs

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout certs/tls.key -out certs/tls.crt \

-subj "/CN=frontend.services"

**Use in Terraform**

These files are base64-encoded and embedded into a Kubernetes TLS Secret via Terraform:

resource "kubernetes\_secret" "frontend\_tls" {

metadata {

name = "frontend-tls"

namespace = "services"

}

data = {

"tls.crt" = filebase64("${path.module}/../certs/tls.crt")

"tls.key" = filebase64("${path.module}/../certs/tls.key")

}

type = "kubernetes.io/tls"

}

**Exposed via Ingress**

resource "kubernetes\_ingress" "frontend" {

metadata {

name = "frontend-ingress"

namespace = "services"

annotations = {

"kubernetes.io/ingress.class" = "gce"

}

}

spec {

tls {

secret\_name = kubernetes\_secret.frontend\_tls.metadata[0].name

}

rule {

http {

path {

path = "/\*"

backend {

service\_name = kubernetes\_service.frontend.metadata[0].name

service\_port = 80

}

}

}

}

}

}

**Optional: Vault and ArgoCD Configuration**

*Not implemented in this solution.*

**Time Spent, Assumptions, and Limitations**

**Time Spent:**

* Terraform GKE setup: ~2 hours
* Monitoring stack (Helm): ~45 minutes
* 3-tier application: ~2 hours
* TLS & Ingress config: ~1 hour
* Documentation: ~30 minutes

**Assumptions:**

* One GCP project used
* Public IP access needed for frontend & Grafana
* Single region GKE cluster

**Known Limitations:**

* No autoscaling policies configured
* No ArgoCD/Vault in place (future work)
* Frontend IP restriction assumes fixed IP range

**Architecture Diagram**

(Include as an image file in your submission: architecture.png)

It should illustrate:

* GKE Cluster with two namespaces: services, monitoring
* Prometheus & Grafana in monitoring
* Frontend, Backend, DB (with PVC) in services
* TLS via Ingress
* IP-restricted LoadBalancer for frontend