**Final Cloud Engineering Assignment: Multi-Cloud Production-Ready Infrastructure**

**Title:** Designing & Deploying a Scalable Multi-Cloud Infrastructure  
 **Level:** Advanced (Final Project)  
 **Course:** Cloud Engineering  
  
 **Objective:** The goal of this assignment is to design, provision, and deploy a **highly available, secure, and scalable infrastructure** using Terraform, Kubernetes, and CI/CD pipelines. You will implement **multi-cloud deployment** (AWS & GCP), integrate monitoring tools, and ensure security best practices.

**Use this application for deployment in this assignment:**<https://github.com/mjhea0/flaskr-tdd>

**Project Scenario**

A **fintech startup** is launching a **secure, scalable, and high-availability application**. They require:  
 - **Multi-cloud deployment** using AWS and GCP for redundancy  
 - **Kubernetes cluster** for microservices deployment  
 - **CI/CD pipeline** for automated deployments  
 - **Centralized logging & monitoring** using Prometheus & ELK stack  
 - **Security best practices** (IAM, VPC peering, encryption, compliance)

Your role is to **design, automate, and deploy** this infrastructure.

**TASK BREAKDOWN**

**Task 1: Infrastructure as Code with Terraform**

**Objective:** Set up a multi-cloud network architecture.

**Steps:** 1️ **Provision AWS Resources:**

* **VPC with public/private subnets**
* **RDS (PostgreSQL) in a private subnet**
* **IAM roles & policies** (least privilege)
* **EKS cluster for Kubernetes**

2️ **Provision GCP Resources:**

* **VPC with peering to AWS VPC**
* **GKE (Google Kubernetes Engine) cluster**

3️ **Create Terraform Modules:**

* **VPC module** (used for both AWS & GCP)
* **Kubernetes module** (used for both EKS & GKE)
* **Security module** (IAM roles, firewall rules, encryption)

**Deliverables:**- Terraform repo (main.tf, modules/, variables.tf, outputs.tf)  
 - Screenshots of Terraform apply and cloud resources  
 - Architecture diagram

**Task 2: Deploy Microservices on Kubernetes**

**Objective:** Deploy microservices with **Kubernetes, Helm, and Ingress.**

**Steps:** 1️ **Containerize applications using Docker** 2️ **Deploy microservices using Helm charts** 3️ **Configure Ingress for traffic routing** 4️ **Implement auto-scaling (Horizontal Pod Autoscaler - HPA)** 5️ **Use service mesh (Istio or Linkerd) for traffic management**

**Deliverables:** - Helm charts repo  
 - Kubernetes YAML files (deployment.yaml, service.yaml, ingress.yaml)  
 - Screenshot of running services in kubectl get pods

**Task 3: Implement CI/CD Pipeline with GitHub Actions & ArgoCD**

**Objective:** Automate deployments with a **CI/CD pipeline.**

**Steps:** 1️ **Set up GitHub Actions:**

* Run Terraform plan & apply on main branch push
* Build & push Docker images to AWS ECR & GCP Container Registry

2️ **Set up ArgoCD for Kubernetes GitOps:**

* Deploy changes automatically to EKS & GKE
* Implement **blue-green deployment** strategy

3️ **Secure CI/CD pipeline:**

* Use **IAM roles & OIDC for GitHub Actions**
* Encrypt secrets using **AWS Secrets Manager**

**Deliverables:** - GitHub Actions YAML file  
 - ArgoCD configuration (argocd.yaml)  
 - Screenshot of a successful deployment in ArgoCD

**Task 4: Observability & Monitoring with Prometheus & ELK**

**Objective:** Implement **centralized monitoring & logging.**

**Steps:** 1️ **Deploy Prometheus & Grafana for metrics:**

* Monitor Kubernetes nodes, pods, and application metrics
* Set up alerts (CPU, memory, network)

2️ **Deploy ELK Stack for logging:**

* Collect logs from Kubernetes pods
* Set up Kibana dashboards

3️ **Configure centralized logging for AWS & GCP:**

* AWS CloudWatch for EKS logs
* GCP Logging for GKE logs

**Deliverables:** - Prometheus & Grafana config files  
 - ELK Stack config files  
 - Screenshot of Grafana dashboard with live metrics

**Task 5: Security Hardening & Compliance**

**Objective:** Ensure **best security practices** in cloud infrastructure.

**Steps:** 1️ **IAM & Least Privilege:**

* Use least-privilege IAM roles for Terraform, Kubernetes, and CI/CD
* Implement MFA for AWS & GCP accounts

2️ **Network Security:**

* **Restrict traffic using Security Groups & firewall rules**
* Implement **VPC peering & private networking**

3️ **Encryption & Secrets Management:**

* Encrypt Terraform state with AWS KMS
* Store API keys & DB credentials in AWS Secrets Manager & GCP Secret Manager

4️ **Compliance & Auditing:**

* Use **AWS Security Hub & GCP Security Command Center** for compliance checks
* Implement logging for **audit trails**

**Deliverables:** - Security policy documentation  
 - Terraform security configurations  
 - Screenshot of compliance checks