**INTERNSHIP TASK -3**

**Multi-Cloud Architecture Design**

## **Overview**

This document outlines a multi-cloud architecture that distributes services across two cloud providers, ensuring interoperability, high availability, and cost efficiency. The architecture leverages Google Cloud Platform (GCP) and another cloud provider (AWS/Azure) to balance workloads and optimize performance.

## **Objectives**

* Ensure seamless interoperability between cloud providers.
* Distribute workloads to optimize cost and performance.
* Enhance reliability and disaster recovery.
* Enable hybrid cloud connectivity.
* Provide security and compliance controls.

## **Architecture Design**

### **1. Cloud Providers Involved**

* **Primary Cloud:** Google Cloud Platform (GCP)
* **Secondary Cloud:** AWS/Azure (Based on specific requirements)

### **2. Core Components**

* **Compute Layer:**
  + GCP: Google Kubernetes Engine (GKE) or Compute Engine.
  + AWS/Azure: Elastic Kubernetes Service (EKS) / Virtual Machines.
* **Networking & Connectivity:**
  + VPN or Interconnect for private connectivity between clouds.
  + Load balancing and API gateway services.
* **Storage & Database:**
  + GCP: Cloud Storage, BigQuery, Firestore.
  + AWS/Azure: S3, DynamoDB, RDS.
* **Identity & Access Management:**
  + Federated authentication using IAM and Azure AD/AWS IAM.
  + Cross-cloud permissions management.
* **Monitoring & Logging:**
  + GCP Operations Suite & AWS CloudWatch/Azure Monitor.
  + Centralized logging with ELK stack or Datadog.

### **3. Interoperability Strategy**

* Use Kubernetes for containerized workloads across clouds.
* API Gateway and service mesh (Istio/Anthos) to manage microservices.
* Data synchronization via Pub/Sub or EventBridge/Event Grid.
* CI/CD pipelines with GitHub Actions or Terraform for infrastructure as code.

### **4. Security & Compliance**

* Encryption in transit and at rest across providers.
* Identity federation and Single Sign-On (SSO).
* Continuous compliance monitoring.

### **5. Disaster Recovery & High Availability**

* Multi-region deployment with failover strategies.
* Backup and restore mechanisms across clouds.

## **Implementation Steps**

1. **Set up networking and connectivity between clouds.**
2. **Deploy compute and storage resources.**
3. **Implement security measures and identity federation.**
4. **Configure monitoring, logging, and observability tools.**
5. **Test failover, DR, and workload balancing.**

## **Conclusion**

This multi-cloud design ensures resilience, interoperability, and optimized performance across GCP and another cloud provider. It allows enterprises to leverage the best of both platforms while maintaining security and cost efficiency.

**📌 Demo Plan for Interoperability**

#### **1️⃣ Setup & Deployment**

* Deploy a **microservices application** where services run across GCP (GKE) and AWS (EKS).
* Configure a **hybrid database setup**, such as **Firestore on GCP syncing with DynamoDB on AWS**.
* Set up a **multi-cloud API Gateway** (AWS API Gateway + GCP API Gateway) for cross-platform service calls.

#### **2️⃣ Interoperability Showcase**

* Demonstrate **real-time communication** between services (e.g., a web app on GCP fetching data from an AWS backend).
* Use **Pub/Sub messaging** (AWS SNS + GCP Pub/Sub) for event-driven architecture.
* Implement **monitoring & logging** (AWS CloudWatch + GCP Logging) for unified visibility.

#### **3️⃣ Testing & Validation**

* Run **latency and performance tests** between cross-cloud services.
* Failover test: If a service on one cloud goes down, ensure seamless traffic rerouting.