**TASK 3**

**Task: Designing a Multi-Cloud Architecture Across Two Cloud Providers**

**Step 1: Select Cloud Providers & Services**

For a two-cloud setup, you can mix and match based on strengths:

| **Service Type** | **Google Cloud** | **AWS** |
| --- | --- | --- |
| Compute | GKE (Kubernetes), Cloud Run | ECS, EKS, Lambda |
| Storage | GCS (Cloud Storage) | S3 |
| Database | Cloud Spanner, BigQuery | RDS, Dynamo DB |
| Networking | Cloud Load Balancing, Interconnect | Route 53, Elastic Load Balancer |
| Security | IAM, Secret Manager | IAM, KMS |

Example Setup: **Web App on Google Cloud, Analytics on AWS**

**Step 2: Design Multi-Cloud Networking**

**1. Secure Interconnection Between Clouds**

* **Option 1:** **Cloud VPN** (Site-to-Site IPsec VPN)
  + **Google Cloud:** Use [Cloud VPN](https://cloud.google.com/network-connectivity/docs/vpn)
  + **AWS:** Use [AWS Site-to-Site VPN](https://aws.amazon.com/vpn/)
* **Option 2:** **Cloud Interconnect & Direct Connect** (Dedicated Private Links)
  + Google Cloud Interconnect ↔ AWS Direct Connect

**2. Load Balancing Across Clouds**

Use **DNS-based Load Balancing** to distribute traffic:

* **Google Cloud:** [Cloud DNS](https://cloud.google.com/dns)
* **AWS:** [Route 53](https://aws.amazon.com/route53/)

**Step 3: Deploy Applications**

**1. Multi-Cloud Kubernetes**

Use **Google Kubernetes Engine (GKE)** and **AWS EKS**:

* Deploy the same containerized app across both clouds
* Use **Anthos** (Google) or **Kubernetes Federation** for management

**2. Microservices Across Clouds**

* Example:
  + **Google Cloud Run** for API services
  + **AWS Lambda** for serverless backend functions

**Step 4: Database & Storage Architecture**

**1. Cross-Cloud Database Synchronization**

* **Option 1:** **Use a Primary-Replica Model**
  + **Google Cloud Spanner (Primary) → AWS RDS (Replica)**
  + Sync using **Kafka, Change Data Capture (CDC), or Cloud SQL federations**
* **Option 2:** **Multi-Cloud NoSQL Database**
  + Use **MongoDB Atlas** or **CockroachDB** for real-time sync
  + Data replicated in both clouds

**2. Object Storage Replication**

* **Google Cloud Storage (GCS) ↔ AWS S3**
  + Sync files using **AWS DataSync** or **Google Storage Transfer Service**

**Step 5: Implement Multi-Cloud Security**

**1. Centralized IAM (Identity & Access Management)**

* Use **Single Sign-On (SSO)** with:
  + **Google IAM + AWS IAM Federation**
  + **Okta, Auth0, or Azure AD**

**2. Cross-Cloud Encryption & Secrets Management**

* **Google Secret Manager** ↔ **AWS Secrets Manager**
* Use **Cloud KMS** in both clouds for encryption

**Step 6: Set Up Multi-Cloud Monitoring & Logging**

**1. Unified Monitoring**

* **Google Cloud Operations Suite (Stackdriver)** + **AWS Cloud Watch**
* **Prometheus + Grafana** for cross-cloud metrics

**2. Centralized Logging**

* Use **Fluentd** or **Elasticsearch** to collect logs from both clouds
* Send logs to **Google Cloud Logging** or **AWS CloudWatch Logs**

**Step 7: Automate Deployment with Multi-Cloud CI/CD**

* **GitHub Actions / GitLab CI/CD** to deploy to both clouds
* **Terraform / Pulumi** for infrastructure-as-code
* **Kubernetes CI/CD tools** (ArgoCD, Flux) for cross-cloud deployments

**Sample Multi-Cloud Architecture (Google Cloud + AWS)**

📌 **Example: Web App on Google Cloud + AI Analytics on AWS**

User → Cloud DNS (Google) + Route 53 (AWS)

├── Google Cloud Load Balancer → Cloud Run (Web App)

├── AWS API Gateway → AWS Lambda (AI Processing)

├── Cloud VPN (Google ↔ AWS) for interconnection

├── Cloud Storage (GCS) → Replicated to AWS S3

├── Database: Cloud Spanner (Google) → AWS DynamoDB (Sync)

├── Monitoring: Prometheus + CloudWatch + Stackdriver

Screenshots





