**1. Introduction**

Modern enterprises increasingly adopt **multi-cloud strategies** to avoid vendor lock-in, improve resilience, and take advantage of the best features offered by different providers.

This documentation explains the design and demonstration of a **multi-cloud system** where:

* **Google Cloud Platform (GCP)** hosts one service and data storage.
* **Amazon Web Services (AWS)** hosts another service and storage.
* Both clouds communicate securely, enabling a seamless workflow.

**2. High-Level Architecture**

* **Frontend / API Gateway:** Entry point that routes traffic to the correct backend service (hosted on GCP or AWS).
* **GCP:**
  + **Service A (User Service):** Handles user profiles.
  + **Google Cloud Storage (GCS):** Stores user-uploaded files (e.g., images, documents).
  + **Pub/Sub:** Publishes events like user.created.
* **AWS:**
  + **Service B (Auth Service):** Handles authentication & user audit logs.
  + **Amazon S3:** Stores audit logs and replicated data.
  + **Kinesis/SQS:** Receives events from GCP for processing.
* **Networking:** A **secure VPN tunnel** connects AWS VPC with GCP VPC.
* **Observability:** OpenTelemetry/Prometheus tracks logs and metrics across both clouds.

**3. Step-by-Step Setup**

**🔹 Step 1: Create Buckets on Both Clouds**

**GCP (Cloud Storage):**

gsutil mb -l us-central1 gs://codtech-gcs-bucket

gsutil cp sample.txt gs://codtech-gcs-bucket/

**AWS (S3):**

aws s3api create-bucket --bucket codtech-s3-bucket --region us-east-1 \

--create-bucket-configuration LocationConstraint=us-east-1

aws s3 cp sample.txt s3://codtech-s3-bucket/

**🔹 Step 2: Deploy Services**

* **GCP (Cloud Run or GKE):** Deploy a user-service API that uploads user files to **GCS**.
* **AWS (ECS/EKS):** Deploy an auth-service API that logs actions to **S3**.

**🔹 Step 3: Configure Event Sharing**

* Enable **Pub/Sub** on GCP → forward messages to AWS **SQS/Kinesis** via a connector.
* Example event: user.created → triggers AWS auth-service to store an audit entry in S3.

**🔹 Step 4: Networking & Security**

* Configure a **VPN tunnel** between AWS VPC and GCP VPC.
* Apply **IAM roles**:
  + GCP service account for bucket/object access.
  + AWS IAM role with AmazonS3FullAccess.
* Enable encryption at rest (CMEK on GCP, SSE on AWS).

**🔹 Step 5: Observability**

* Deploy **OpenTelemetry Collector** in both clusters.
* Export logs/metrics to a centralized dashboard (Grafana/Cloud Monitoring).

**4. Demo Workflow**

1. **User Request:** A client sends a request to the GCP user-service.
2. **File Upload:** File stored in **Google Cloud Storage**.
3. **Event Published:** A user.created event sent to Pub/Sub.
4. **Cross-Cloud Transfer:** Event forwarded to **AWS SQS/Kinesis**.
5. **AWS Processing:** auth-service receives event and logs audit details in **S3**.
6. **Verification:**
   * Public file accessible from GCS bucket link.
   * Audit log confirmed inside AWS S3.
   * Logs/traces visible in observability dashboard.

**5. Benefits of Multi-Cloud Setup**

* **High Availability:** Outage in one cloud does not affect the whole system.
* **Flexibility:** Best-of-breed services used from each provider.
* **Scalability:** Workload can shift between providers as required.
* **Cost Optimization:** Ability to compare and optimize provider costs.

**6. Deliverables**

* **Documentation (this file).**
* **Demo (screenshots / recording):**
  + Bucket creation in both GCP & AWS.
  + File uploads.
  + Event transfer proof.
  + Audit logs in both clouds.
* **Proof of Interoperability** → End-to-end workflow across providers.

**7. Conclusion**

This project successfully demonstrates a **multi-cloud architecture** distributed across **Google Cloud** and **AWS**. It highlights interoperability, resilience, and scalability through practical implementation of **bucket setup, file uploads, event bridging, and security**.