**Task 3**

**Documentation on designing multi cloud architecture.**

A multi-cloud architecture involves utilizing cloud services from two or more providers (e.g., AWS, Azure, GCP) to distribute workloads. This approach enhances redundancy, resilience, vendor flexibility, and optimized service usage based on provider strengths.

1. Goals and Use Cases

* High availability and failover across clouds
* Use best-of-breed services from each cloud (e.g., BigQuery on GCP, S3 on AWS)
* Data residency and compliance
* Cost optimization and vendor flexibility

2. Multi-Cloud Design Models (with GCP)

A. Split by Service

Different services are deployed to different clouds based on capabilities.

Example:

* GCP: BigQuery for analytics, Cloud Run for serverless workloads
* AWS: S3 for object storage, Lambda for some functions

B. Active-Passive

* GCP hosts the primary stack
* Secondary stack is on AWS or Azure, activated during GCP downtime

C. Active-Active

* Both GCP and another cloud provider handle traffic in parallel
* Requires global load balancing, data replication, and conflict resolution

3. Key GCP Components for Multi-Cloud

a. Networking

* Cloud Interconnect: Dedicated low-latency connections to AWS/Azure
* Cloud VPN: IPsec tunnels between GCP and other clouds
* Cloud Load Balancing + Cloud DNS: For global, geo-aware traffic routing

b. Service Integration

* Anthos: GCP's hybrid and multi-cloud platform to manage workloads across GCP, AWS, and Azure using Kubernetes (GKE).
* Apigee: API management platform that supports cross-cloud routing, security, and monitoring.

c. Identity and Access

* Cloud Identity / IAM: Centralized identity provider
* Integrate with external IdPs (e.g., Azure AD) for SSO and role-based access control
* Use Workload Identity Federation to securely access GCP services from non-GCP environments

4. Multi-Cloud Architecture Example

Scenario: Analytics on GCP, Web App on AWS

* Frontend: Hosted on AWS (CloudFront + S3)
* Backend APIs: Deployed using AWS Lambda and API Gateway
* Data Analytics: Events streamed from AWS to GCP using Pub/Sub and BigQuery
* Interconnect: AWS ↔ GCP via VPN or Interconnect
* Monitoring: Centralized in a third-party tool like Datadog or Grafana

5. Tools for Multi-Cloud Automation

* Terraform: Infrastructure as Code across GCP and other clouds
* Anthos Config Management: Manage GKE and other clusters declaratively
* Spinnaker / ArgoCD: Multi-cloud CI/CD pipelines
* Cloud Build: Can deploy to non-GCP environments using custom steps

6. Security and Governance

* Enforce consistent IAM policies across providers
* Use VPC Service Controls in GCP to protect sensitive services
* Apply encryption at rest and in transit across all environments
* Centralize logging with Cloud Logging, Cloud Audit Logs, and external log aggregators

7. Challenges and Mitigation

| Challenge | Mitigation |
| --- | --- |
| Latency | Use Interconnects and regional placement |
| Service inconsistencies | Use containers or serverless for portability |
| Monitoring complexity | Centralized observability tools |
| Cost tracking | Use tools like GCP Cost Management + third-party platforms |

8. Best Practices

* Use Anthos for consistent Kubernetes management
* Standardize DevOps and CI/CD pipelines across clouds
* Keep services loosely coupled
* Regularly test failovers and DR plans
* Monitor egress and interconnect costs

Demo Workflow

1. User accesses web app on AWS via S3-hosted frontend.
2. Frontend triggers an AWS Lambda function via API Gateway.
3. Lambda processes input and sends event data to a REST API hosted on Cloud Run (GCP).
4. GCP Cloud Run API forwards the event to Pub/Sub.
5. A Cloud Function on GCP subscribes to the Pub/Sub topic and writes data into BigQuery.
6. User can view a live dashboard (e.g., via Looker or Data Studio) visualizing data stored in BigQuery.

Interoperability Highlights

* Secure API communication across providers
* Data pipeline spans clouds, showing workload portability
* Unified observability using shared logging and monitoring tools (e.g., Datadog)
* Standardized CI/CD with Terraform and GitHub Actions deploying to both environments