Scenario

**Your company is migrating its existing infrastructure to GCP to support a scalable, secure, and cost-effective solution. The current on-premises setup hosts a web application with the following requirements:**

1. **Traffic: The application serves 10 million users monthly, with peaks during certain times.**
2. **Components:**
   * 1. **Frontend: React-based web app.**
     2. **Backend: Java.**
     3. **Database: Relational database (currently MySQL).**
3. **Data Processing: A nightly batch job processes user data (~5TB daily) to generate analytics.**
4. **Monitoring: Application health and performance need real-time monitoring with alerts.**
5. **Security: The system must adhere to compliance standards (e.g., GDPR) and protect sensitive user data.**

Task

**Design an architecture for the migration and ongoing operation of the application on GCP. Your deliverables include:**

1. **High-Level Architecture Diagram**
   * 1. **Use appropriate GCP services to meet the requirements.**
2. **Service Selection and Justification**
   * 1. **Explain your choice of GCP services for compute, storage, networking, database, and monitoring.**
3. **Scalability and Cost Optimization**
   * 1. **Describe how your architecture handles scaling and minimizes costs.**
4. **Security and Compliance Measures**
   * 1. **Detail the steps to secure the infrastructure and ensure compliance.**
5. **Operational Strategy**
   * 1. **Include monitoring, alerting, and disaster recovery plans.**
6. **Bonus (Optional)**
   * 1. **Propose a CI/CD pipeline for the web application deployment.**
     2. **Suggest cost-saving strategies for the initial migration phase.**

Submission Format

* **Document: A 2-5 page report (PDF or Word).**
* **Diagram: Architecture diagram in PNG, PDF, or a link to a shared diagramming tool (e.g., Lucidchart, Draw.io).**
* **Code Snippets: If relevant, include any Terraform or shell scripts used for infrastructure as code (IaC).**

Evaluation Criteria

* **Technical Depth: Understanding of GCP services and their appropriate use.**
* **Scalability and Reliability: Effectiveness of the solution in handling traffic and failures.**
* **Cost Optimization: Efforts to minimize unnecessary expenses.**
* **Security: Measures for safeguarding data and ensuring compliance.**
* **Clarity and Presentation: Quality of documentation and explanation.**

**GCP Migration Architecture Report**

**Web Application Infrastructure Migration**

**Executive Summary**

This document outlines a comprehensive migration strategy for moving a high-traffic web application (10M monthly users) from on-premises infrastructure to Google Cloud Platform (GCP). The proposed architecture leverages managed GCP services to ensure scalability, security, cost-effectiveness, and compliance with standards like GDPR.

**1. High-Level Architecture Overview**

**Core Architecture Components**

**Frontend Tier:**

* **Cloud CDN** with **Cloud Load Balancer** for global content delivery
* **Cloud Storage** for static assets (React build files)
* **Firebase Hosting** as alternative for enhanced performance

**Application Tier:**

* **Google Kubernetes Engine (GKE)** for containerized Java backend services
* **Cloud Run** for serverless microservices (optional hybrid approach)
* **VPC** with private subnets for security isolation

**Data Tier:**

* **Cloud SQL for MySQL** with read replicas for primary database
* **Cloud Storage** for data lake and backup storage
* **BigQuery** for analytics and data warehousing

**Processing Tier:**

* **Dataflow** for batch processing (5TB daily analytics jobs)
* **Cloud Composer** (Apache Airflow) for workflow orchestration
* **Dataproc** for Spark-based processing if needed

**Monitoring & Security:**

* **Cloud Monitoring** and **Cloud Logging** for observability
* **Cloud Security Command Center** for security insights
* **Identity and Access Management (IAM)** for access control
* **Binary Authorization** for container security

**2. Service Selection and Justification**

**Compute Services**

**Google Kubernetes Engine (GKE)**

* **Justification**: Provides container orchestration for Java backend with auto-scaling, rolling updates, and high availability
* **Configuration**: Multi-zone cluster with node auto-provisioning
* **Benefits**: Handles traffic spikes automatically, supports blue-green deployments

**Cloud Run (Optional)**

* **Justification**: Serverless option for stateless microservices
* **Use Case**: API endpoints with variable traffic patterns
* **Benefits**: Pay-per-request pricing, automatic scaling to zero

**Database Services**

**Cloud SQL for MySQL**

* **Justification**: Managed MySQL service maintains compatibility with existing database
* **Configuration**:
  + Primary instance in main region
  + Read replicas in multiple regions for read scaling
  + Automated backups and point-in-time recovery
* **Benefits**: Reduces operational overhead, built-in high availability

**BigQuery**

* **Justification**: Serverless data warehouse for analytics processing
* **Use Case**: Store and analyze the 5TB daily processed data
* **Benefits**: Handles petabyte-scale analytics, pay-per-query pricing

**Storage Services**

**Cloud Storage**

* **Justification**: Object storage for static assets, backups, and data lake
* **Configuration**:
  + Multi-regional buckets for static assets
  + Regional buckets for backup storage
  + Lifecycle policies for cost optimization
* **Benefits**: 99.999999999% durability, integrated with CDN

**Networking Services**

**Cloud Load Balancer**

* **Justification**: Global load balancing with SSL termination
* **Configuration**: HTTPS load balancer with backend services
* **Benefits**: Automatic traffic distribution, DDoS protection

**Cloud CDN**

* **Justification**: Content delivery network for global performance
* **Benefits**: Reduces latency, offloads traffic from origin servers

**VPC with Private Google Access**

* **Justification**: Secure network isolation with controlled internet access
* **Configuration**: Private subnets with NAT gateway for outbound traffic

**Data Processing Services**

**Dataflow**

* **Justification**: Managed Apache Beam service for batch and stream processing
* **Use Case**: Process 5TB daily user data for analytics
* **Benefits**: Auto-scaling, serverless, handles both batch and streaming

**Cloud Composer**

* **Justification**: Managed Apache Airflow for workflow orchestration
* **Use Case**: Schedule and monitor complex data pipelines
* **Benefits**: Built-in monitoring, integration with other GCP services

**Monitoring Services**

**Cloud Monitoring**

* **Justification**: Native GCP monitoring with custom metrics
* **Features**: SLA monitoring, alerting policies, dashboards

**Cloud Logging**

* **Justification**: Centralized logging with powerful search capabilities
* **Features**: Log-based metrics, export to BigQuery for analysis

**3. Scalability and Cost Optimization**

**Scalability Strategy**

**Horizontal Scaling:**

* GKE Horizontal Pod Autoscaler (HPA) based on CPU/memory metrics
* Cluster Autoscaler for node provisioning
* Cloud SQL read replicas for database read scaling

**Vertical Scaling:**

* Vertical Pod Autoscaler (VPA) for right-sizing containers
* Cloud SQL instance scaling for database performance

**Global Scaling:**

* Multi-region deployment with global load balancer
* CDN for edge caching and reduced latency

**Cost Optimization Measures**

**Compute Optimization:**

* Use Spot VMs for non-critical workloads (up to 80% savings)
* Implement node auto-scaling to match demand
* Use committed use discounts for predictable workloads

**Storage Optimization:**

* Implement lifecycle policies (move to Coldline/Archive storage)
* Use regional storage for backups instead of multi-regional
* Compress and deduplicate data before storage

**Network Optimization:**

* Use Premium Tier networking only where necessary
* Implement Cloud CDN to reduce egress costs
* Use Private Google Access to avoid NAT gateway costs for internal traffic

**Database Optimization:**

* Use read replicas instead of scaling primary instance
* Implement connection pooling to reduce instance requirements
* Schedule maintenance during low-traffic periods

**Resource Management:**

* Implement resource quotas and budgets with alerts
* Use labels and tags for cost tracking and optimization
* Regular rightsizing analysis using Recommender API

**4. Security and Compliance Measures**

**Data Protection (GDPR Compliance)**

**Data Encryption:**

* Encryption at rest using Google-managed keys
* Encryption in transit with TLS 1.3
* Customer-managed encryption keys (CMEK) for sensitive data

**Data Residency:**

* Deploy resources in EU regions for GDPR compliance
* Use regional persistent disks and Cloud SQL instances
* Configure data processing jobs to run in compliant regions

**Data Access Controls:**

* Implement principle of least privilege with IAM
* Use Cloud IAM Conditions for fine-grained access control
* Enable Cloud Audit Logs for all data access

**Network Security**

**Network Isolation:**

* Private GKE cluster with no public IP addresses
* VPC-native networking with authorized networks
* Private Google Access for accessing GCP services

**Firewall Rules:**

* Deny-all default policy with explicit allow rules
* Application-level firewall rules using GKE Network Policies
* Cloud Armor for DDoS protection and WAF capabilities

**Application Security**

**Container Security:**

* Use Binary Authorization to ensure trusted container images
* Regularly scan images with Container Analysis API
* Implement Pod Security Standards in GKE

**Identity and Access Management:**

* Service accounts with minimal required permissions
* Workload Identity for secure pod-to-GCP service communication
* Multi-factor authentication for human users

**Compliance Monitoring**

**Security Monitoring:**

* Cloud Security Command Center for security insights
* VPC Flow Logs for network traffic analysis
* Continuous compliance monitoring with Cloud Asset Inventory

**Audit and Logging:**

* Enable Cloud Audit Logs for all services
* Export logs to BigQuery for long-term retention
* Implement log retention policies per compliance requirements

**5. Operational Strategy**

**Monitoring and Alerting**

**Application Monitoring:**

* Custom metrics for business KPIs (user registrations, transactions)
* SLA monitoring with error budgets
* Distributed tracing with Cloud Trace

**Infrastructure Monitoring:**

* Resource utilization alerts (CPU, memory, disk)
* Service health checks and uptime monitoring
* Database performance monitoring

**Alerting Policies:**

* Tiered alerting (warning, critical, emergency)
* Integration with PagerDuty or similar for incident management
* Notification channels (email, SMS, Slack)

**Disaster Recovery Plan**

**Backup Strategy:**

* Automated Cloud SQL backups with 7-day retention
* Cross-region backup replication for critical data
* Application data backed up to Cloud Storage

**Recovery Procedures:**

* RTO (Recovery Time Objective): 4 hours
* RPO (Recovery Point Objective): 1 hour
* Multi-region deployment for high availability
* Documented runbooks for common failure scenarios

**Business Continuity:**

* Blue-green deployment strategy for zero-downtime updates
* Database failover procedures with read replica promotion
* CDN and load balancer automatically route around failed regions

**Maintenance and Updates**

**Patch Management:**

* Automated OS patches for GKE nodes
* Container image vulnerability scanning
* Scheduled maintenance windows during low-traffic periods

**Capacity Planning:**

* Monthly capacity reviews using historical data
* Predictive scaling based on business growth projections
* Resource utilization trend analysis

**6. Bonus: CI/CD Pipeline and Migration Strategies**

**CI/CD Pipeline Architecture**

**Source Code Management:**

* Google Cloud Source Repositories or GitHub integration
* Branch protection policies and code review requirements

**Build Pipeline:**

* Cloud Build for containerizing applications
* Automated testing in build pipeline
* Security scanning of container images

**Deployment Pipeline:**

* GitOps approach using Cloud Build and GKE
* Environment promotion (dev → staging → production)
* Automated rollback on deployment failures

**Pipeline Configuration:**

# cloudbuild.yaml example

steps:

- name: 'gcr.io/cloud-builders/docker'

args: ['build', '-t', 'gcr.io/$PROJECT\_ID/app:$BUILD\_ID', '.']

- name: 'gcr.io/cloud-builders/docker'

args: ['push', 'gcr.io/$PROJECT\_ID/app:$BUILD\_ID']

- name: 'gcr.io/cloud-builders/kubectl'

args: ['set', 'image', 'deployment/app', 'app=gcr.io/$PROJECT\_ID/app:$BUILD\_ID']

env:

- 'CLOUDSDK\_COMPUTE\_ZONE=us-central1-a'

- 'CLOUDSDK\_CONTAINER\_CLUSTER=production-cluster'

**Migration Phase Cost-Saving Strategies**

**Phased Migration Approach:**

1. **Phase 1**: Migrate static assets and CDN (low risk, immediate benefits)
2. **Phase 2**: Migrate database with replication to on-premises
3. **Phase 3**: Migrate application tier with parallel running
4. **Phase 4**: Migrate batch processing workloads

**Cost-Saving Tactics:**

* Use Preemptible VMs for development and testing environments
* Implement auto-shutdown policies for non-production resources
* Leverage free tier services during initial testing
* Use Cloud Migration tools for automated assessment and migration

**Migration Tools:**

* Migrate for Compute Engine for VM migration
* Database Migration Service for database migration
* Transfer Service for data migration

**Cost Estimates and ROI**

**Monthly Cost Breakdown (Estimated)**

* **Compute (GKE)**: $3,000-5,000
* **Database (Cloud SQL)**: $2,000-3,000
* **Storage (Cloud Storage)**: $500-1,000
* **Network (Load Balancer, CDN)**: $1,000-2,000
* **Data Processing (Dataflow)**: $1,500-2,500
* **Monitoring and Security**: $500-1,000
* **Total Estimated**: $8,500-14,500/month

**Expected Benefits**

* **Operational Cost Reduction**: 30-40% reduction in infrastructure management costs
* **Improved Performance**: 50% reduction in page load times with CDN
* **Enhanced Reliability**: 99.95% uptime SLA
* **Faster Development**: 60% faster deployment cycles with CI/CD

**Conclusion**

This architecture provides a robust, scalable, and secure foundation for migrating your web application to GCP. The proposed solution leverages managed services to reduce operational overhead while maintaining flexibility for future growth. The phased migration approach minimizes risk while the comprehensive monitoring and security measures ensure compliance and reliability.

The estimated cost savings, combined with improved performance and reduced operational complexity, provide a strong business case for the migration. The architecture is designed to handle current requirements while providing room for future growth and technological evolution.