GCP Big Data Case Study: Global Retail Analytics Platform

Executive Summary

Company: GlobalMart Inc.
Industry: Retail & E-commerce
Challenge: Process and analyze 50TB+ daily transaction data from 5000+ stores across 45 countries
Solution: Enterprise-grade GCP Big Data platform with robust governance and security

---

Table of Contents

---

1. Business Requirements

Stakeholders & Roles

Role	Count	Responsibilities
Data Engineers	15	Pipeline development, ETL jobs
Data Scientists	20	ML models, analytics
Business Analysts	30	Reporting, dashboards
Data Stewards	5	Data governance, quality
Platform Admins	8	Infrastructure, security
External Partners	10	Third-party data integration

Key Requirements

- **Volume**: 50TB daily ingestion, 5PB total storage

- **Velocity**: Real-time streaming + batch processing
- **Variety**: Structured (SQL), Semi-structured (JSON), Unstructured (logs, images)
- **Compliance**: GDPR, PCI-DSS, SOC 2
- **Availability**: 99.95% SLA
- **Geographic**: Multi-region deployment (US, EU, APAC)

---

2. GCP Framework & Standards

2.1 Google Cloud Architecture Framework

Google Cloud follows a comprehensive architecture framework based on these pillars:

Operational Excellence

- Infrastructure as Code (Terraform)
- CI/CD pipelines
- Automated testing and deployment
- Monitoring and alerting

Security & Compliance

- Zero Trust security model
- VPC Service Controls
- Data Loss Prevention (DLP)
- Encryption at rest and in transit

Reliability

- Multi-region deployment
- Automated backups and disaster recovery
- Health checks and auto-healing
- Capacity planning

Performance Efficiency

- Auto-scaling compute resources
- Caching strategies
- Query optimization
- CDN for static content

Cost Optimization

- Committed Use Discounts (CUDs)
- Sustained Use Discounts (SUDs)
- Preemptible VMs for batch jobs
- Resource labeling and chargeback

2.2 Resource Hierarchy & Organization Structure

Google Cloud uses a hierarchical resource organization model:

```
Organization (globalmart.com)
│
├── Folder: Production
│   ├── Host Project (hub): prd-network-host-01
│   └── Service Projects (spokes):
│       ├── prd-bigdata-analytics-01
│       ├── prd-bigdata-ml-01
│       └── prd-bigdata-streaming-01
│
├── Folder: Non-Production
│   ├── Host Project (hub): nprd-network-host-01
│   └── Service Projects (spokes):
│       ├── dev-bigdata-analytics-01
│       ├── stg-bigdata-analytics-01
│       └── qa-bigdata-analytics-01
│
└── Folder: Shared Services
    ├── shrd-security-logging-01
    ├── shrd-monitoring-ops-01
    └── shrd-identity-iam-01
```

2.3 Shared VPC Network Architecture (Hub and Spoke Model)

Google Cloud implements network isolation using **Shared VPC** with a hub-and-spoke topology:

```
Organization (globalmart.com)
│
├── Folder: Production
│   ├── Host Project (hub): prd-network-host-01
│   └── Service Projects (spokes):
│       ├── prd-bigdata-analytics-01
│       ├── prd-bigdata-ml-01
│       └── prd-bigdata-streaming-01
│
├── Folder: Non-Production
│   ├── Host Project (hub): nprd-network-host-01
│   └── Service Projects (spokes):
│       ├── dev-bigdata-analytics-01
│       ├── stg-bigdata-analytics-01
│       └── qa-bigdata-analytics-01
│
└── Folder: Shared Services
    ├── shrd-security-logging-01
    ├── shrd-monitoring-ops-01
    └── shrd-identity-iam-01
```

2.4 Google Cloud Best Practices Framework

Principle	GCP Implementation
Identity Foundation	Cloud Identity, Workforce Identity Federation, Context-Aware Access, IAM Conditions
Network Foundation	Shared VPC, Private Google Access, Private Service Connect, Cloud Interconnect
Logging & Monitoring	Cloud Logging, Cloud Monitoring, Cloud Trace, Error Reporting, Cloud Profiler
Data Governance	Data Catalog, Dataplex, Cloud DLP, Policy Tags, Column-level Security
Encryption	Cloud KMS, Customer-Managed Encryption Keys (CMEK), Cloud HSM, External Key Manager (Cloud EKM)
Compliance	Assured Workloads, Access Transparency, VPC Service Controls, Organization Policy Service

2.5 Google Cloud Design Patterns for Big Data

Medallion Architecture (Bronze-Silver-Gold)

- **Bronze (Raw)**: Immutable raw data from sources → Cloud Storage Standard
- **Silver (Processed)**: Cleaned, validated, deduplicated → BigQuery partitioned tables
- **Gold (Curated)**: Business-level aggregates → BigQuery materialized views

Streaming & Batch Lambda Architecture

- **Batch Layer**: Dataproc/Dataflow batch jobs → BigQuery tables
- **Speed Layer**: Pub/Sub → Dataflow streaming → Bigtable/BigQuery

- **Serving Layer**: BigQuery + Looker for unified queries

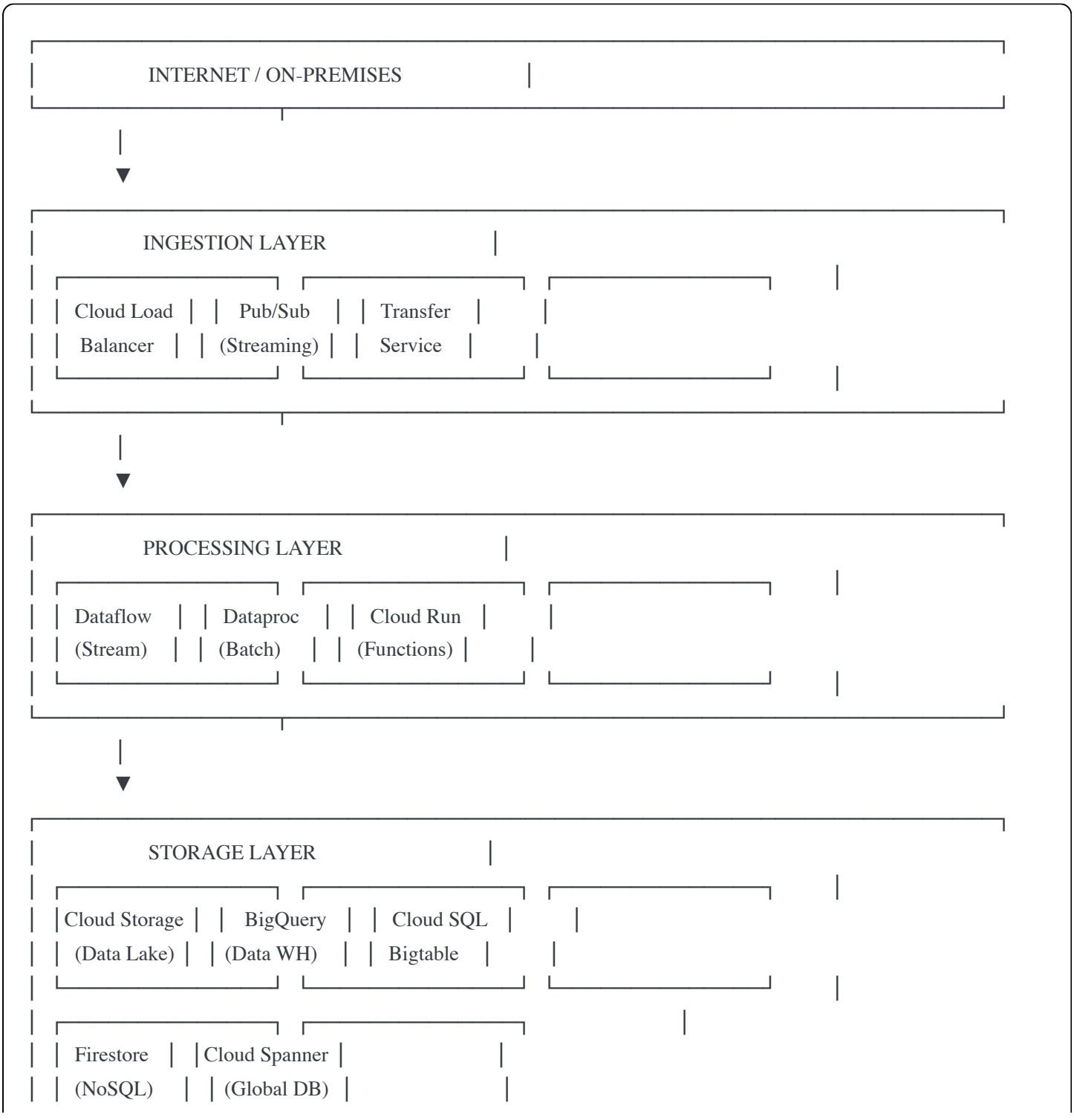
Data Mesh Principles

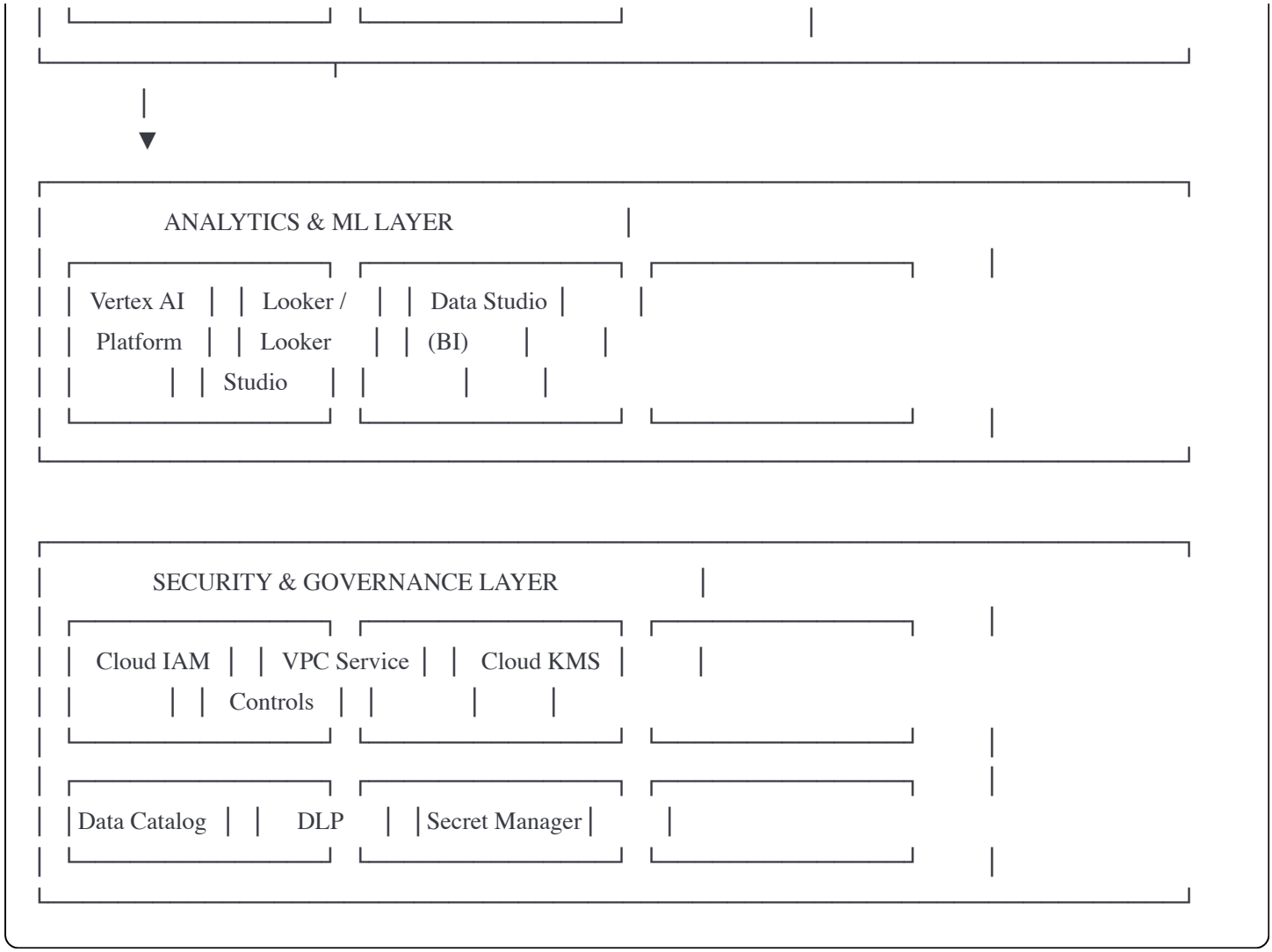
- **Domain-oriented data ownership**: Separate projects per business domain
- **Data as a product**: Published datasets in Data Catalog with SLOs
- **Self-serve data platform**: Dataplex for data lake management
- **Federated computational governance**: Policy tags and IAM at org level

---

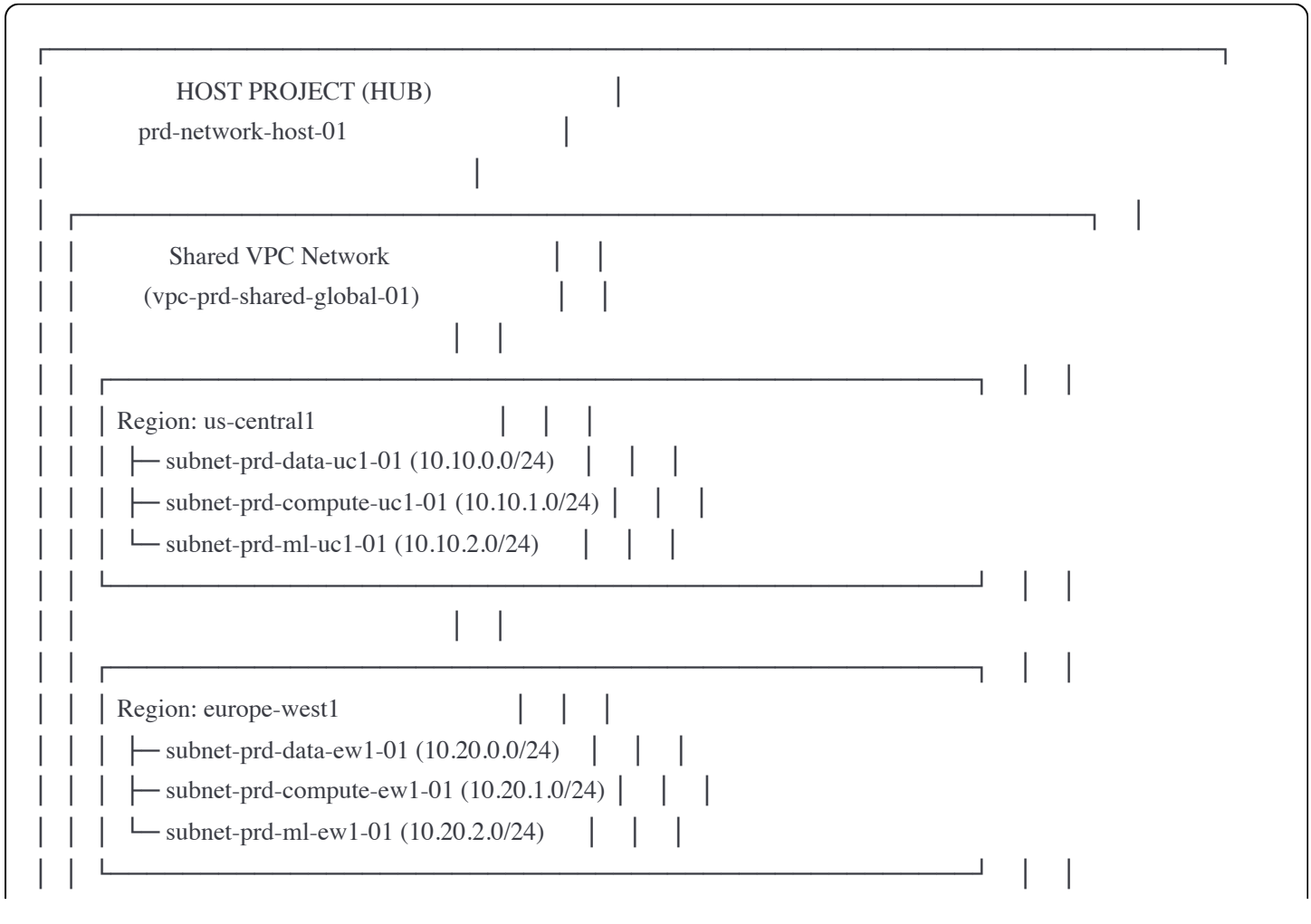
3. Architecture Overview

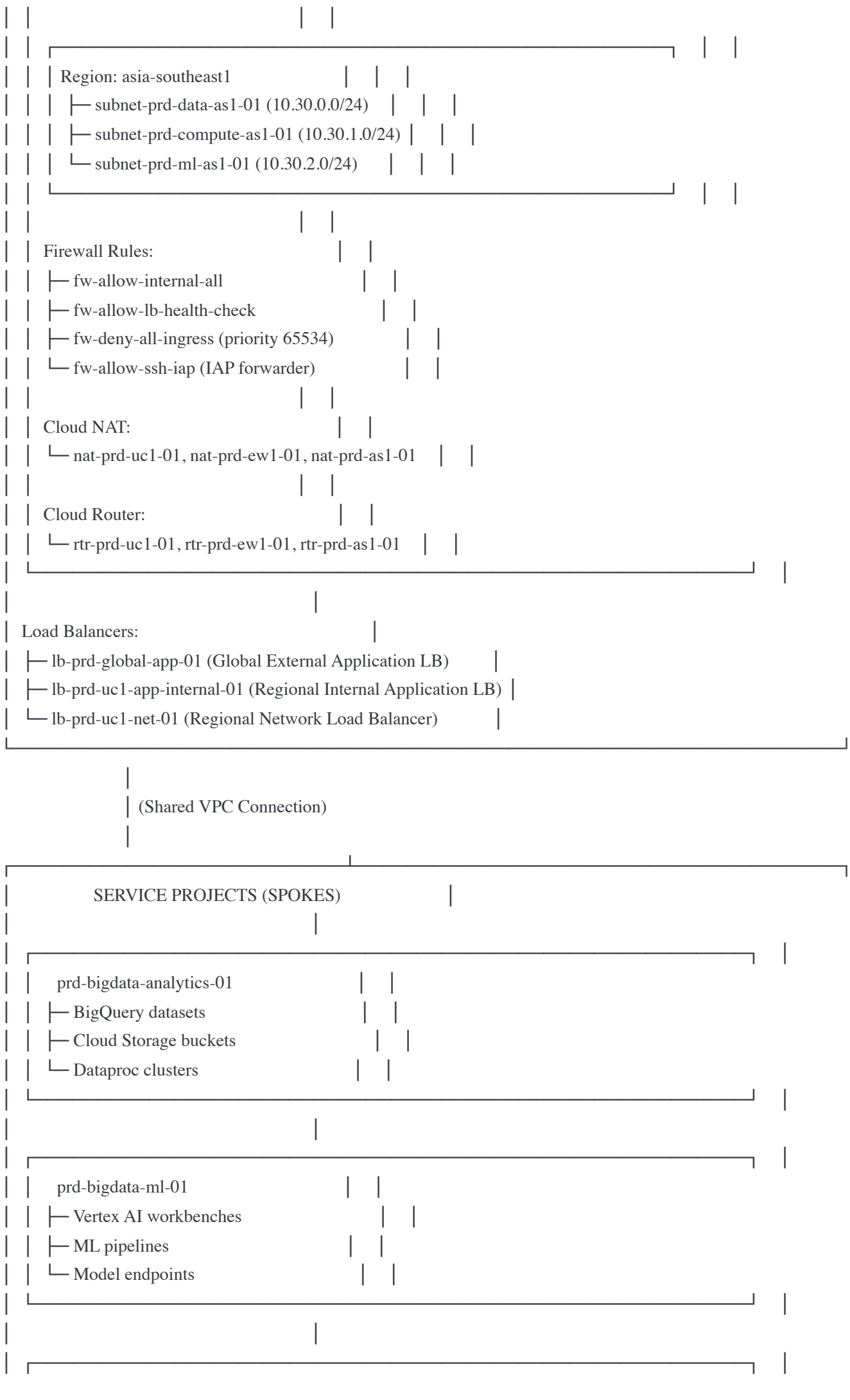
3.1 High-Level Architecture

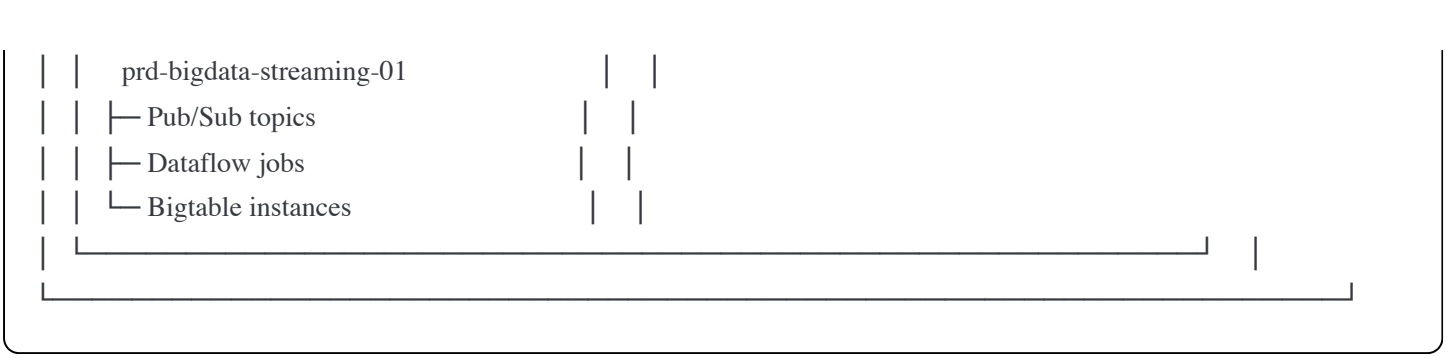




3.2 Detailed Network Architecture




Region: asia-southeast1
├─ subnet-prd-data-as1-01 (10.30.0.0/24)
├─ subnet-prd-compute-as1-01 (10.30.1.0/24)
└─ subnet-prd-ml-as1-01 (10.30.2.0/24)
Firewall Rules:
├─ fw-allow-internal-all
├─ fw-allow-lb-health-check
├─ fw-deny-all-ingress (priority 65534)
└─ fw-allow-ssh-iap (IAP forwarder)
Cloud NAT:
└─ nat-prd-uc1-01, nat-prd-ew1-01, nat-prd-as1-01
Cloud Router:
└─ rtr-prd-uc1-01, rtr-prd-ew1-01, rtr-prd-as1-01
Load Balancers:
├─ lb-prd-global-app-01 (Global External Application LB)
├─ lb-prd-uc1-app-internal-01 (Regional Internal Application LB)
└─ lb-prd-uc1-net-01 (Regional Network Load Balancer)
(Shared VPC Connection)
SERVICE PROJECTS (SPOKES)
prd-bigdata-analytics-01
├─ BigQuery datasets
├─ Cloud Storage buckets
└─ Dataproc clusters
prd-bigdata-ml-01
├─ Vertex AI workbenches
├─ ML pipelines
└─ Model endpoints



---

4. Naming Conventions

4.1 Standard Naming Format

{resource-type}-{environment}-{workload}-{region}-{instance}

Examples:
- gcs-prd-rawdata-uc1-01 (Cloud Storage bucket)
- bq-prd-analytics-multi-01 (BigQuery dataset)
- vm-dev-dataproc-ew1-master-01 (Compute Engine VM)
- topic-prd-transactions-uc1-01 (Pub/Sub topic)

4.2 Component-Specific Conventions

Projects

{environment}-{domain}-{purpose}-{instance}

Examples:
- prd-bigdata-analytics-01
- dev-bigdata-ml-01
- shrd-security-logging-01

VPC Networks & Subnets

Networks: vpc-{env}-{purpose}-{region/global}-{instance}
Subnets: subnet-{env}-{purpose}-{region}-{instance}

Examples:
- vpc-prd-shared-global-01
- subnet-prd-data-uc1-01
- subnet-nprd-compute-ew1-02

Cloud Storage Buckets

gcs-{env}-{purpose}-{region}-{instance}

Storage Classes: standard, nearline, coldline, archive

Examples:
- gcs-prd-raw-landing-uc1-01 (Standard class)
- gcs-prd-processed-silver-multi-01 (Multi-region Standard)
- gcs-prd-curated-gold-multi-01 (Multi-region Standard)
- gcs-prd-logs-archive-uc1-01 (Archive class)
- gcs-prd-backup-coldline-us-01 (Coldline for infrequent access)

BigQuery

Datasets: bq_{env}_{domain}_{purpose}
Tables: {entity}_{type}
Editions: Standard, Enterprise, Enterprise Plus

Examples:
- Dataset: bq_prd_retail_transactions (Standard Edition)
  - Tables: sales_fact, customer_dim, product_dim
- Dataset: bq_prd_ml_features (Enterprise Edition for BI Engine)
  - Tables: customer_features_daily, product_embeddings
- Dataset: bq_prd_streaming_realtime (Enterprise Plus for autoscaling slots)
  - Tables: clickstream_events, inventory_updates

Pub/Sub

Topics: topic-{env}-{data-source}-{region}-{instance}
Subscriptions: sub-{env}-{consumer}-{region}-{instance}

Features: Message ordering, Dead letter queues, Schema validation, Exactly-once delivery

Examples:
- topic-prd-pos-transactions-uc1-01 (with Avro schema)
- sub-prd-realtime-analytics-uc1-01 (with message ordering)
- topic-prd-inventory-updates-multi-01 (global topic)
- sub-prd-dlq-failed-messages-uc1-01 (dead letter queue)

Cloud SQL / Cloud Spanner

Cloud SQL: sql-{env}-{purpose}-{region}-{instance}
Cloud Spanner: span-{env}-{purpose}-{config}-{instance}

Spanner Configurations:
- Regional: nam3 (Iowa), eur3 (Belgium)
- Multi-region: nam6 (US), eur6 (Europe), nam-eur-asia1 (Global)

Examples:
- sql-prd-metadata-uc1-01 (Cloud SQL PostgreSQL HA)
- span-prd-orders-nam6-01 (Multi-region US Spanner)
- span-prd-inventory-nam-eur-asia1-01 (Global multi-region)
- span-prd-catalog-eur3-01 (Regional Europe)

Bigtable

bt-{env}-{purpose}-{region}-{instance}

Instance Types: Development (1 node), Production (3+ nodes)
Storage Type: SSD, HDD

Examples:
- bt-prd-timeseries-uc1-01 (Production, SSD, multi-cluster)
- bt-prd-clickstream-as1-01 (Production, SSD, regional)
- bt-dev-testing-uc1-01 (Development, single node)

Dataproc Clusters

dpc-{env}-{purpose}-{region}-{instance}

Cluster Modes: standard (with HDFS), single-node, high-availability

Examples:
- dpc-prd-batch-etl-uc1-01 (Standard mode, ephemeral)
- dpc-dev-adhoc-query-ew1-01 (Single-node for development)
- dpc-prd-long-running-uc1-01 (HA mode with 3 masters)

Dataflow Jobs

df-{env}-{pipeline}-{region}-{instance}

Execution: Batch, Streaming, Flexible Resource Scheduling (FlexRS)

Examples:
- df-prd-stream-processor-uc1-01 (Streaming pipeline)
- df-prd-batch-aggregator-multi-01 (Batch pipeline)
- df-prd-batch-etl-flexrs-uc1-01 (FlexRS for cost optimization)

Cloud Functions / Cloud Run

cf-{env}-{function}-{region}-{instance}
cr-{env}-{service}-{region}-{instance}

Examples:
- cf-prd-data-validator-uc1-01
- cr-prd-api-gateway-uc1-01

Service Accounts

{workload}-{environment}@{project-id}.iam.gserviceaccount.com

Examples:
- dataflow-compute-prd@prd-bigdata-streaming-01.iam.gserviceaccount.com
- bigquery-loader-prd@prd-bigdata-analytics-01.iam.gserviceaccount.com
- vertex-training-prd@prd-bigdata-ml-01.iam.gserviceaccount.com

Firewall Rules

fw-{allow/deny}-{protocol/service}-{source}-{target}

Examples:
- fw-allow-https-internet-lb
- fw-allow-ssh-iap-compute
- fw-deny-all-ingress-default

Load Balancers

lb-{env}-{region/global}-{type}-{instance}

Types: app (Application LB), net (Network LB), proxy (Proxy LB)

Examples:
- lb-prd-global-app-01 (Global External Application Load Balancer)
- lb-prd-uc1-app-internal-01 (Regional Internal Application Load Balancer)
- lb-prd-uc1-net-01 (Regional Network Load Balancer)

4.3 Labels (Tags)

All resources must have these mandatory labels:

```yaml
environment: prd | nprd | dev | stg | qa
cost_center: retail | supply-chain | finance
data_classification: public | internal | confidential | restricted
owner: team-name
project_code: project-identifier
compliance: pci-dss | gdpr | sox | hipaa
backup_policy: daily | weekly | monthly | none
```

---

5. IAM & Security Design

5.1 Organization Structure

```
Organization: globalmart.com (Org ID: 123456789)
├── Admin Group: gcp-org-admins@globalmart.com
├── Billing Account: 01234-56789A-BCDEF0
└── Organization Policies:
    ├── Restrict Public IP
    ├── Restrict Resource Locations (us-central1, europe-west1, asia-southeast1)
    ├── Enforce Uniform Bucket-Level Access
    └── Require CMEK for BigQuery
```

5.2 Folder & Project Hierarchy

```
globalmart.com
│
├── Production (Folder)
│   ├── Organization Policies:
│   │   ├── Require OS Login
│   │   └── Disable Service Account Key Creation
│   │
│   ├── prd-network-host-01
```

```
│   │   └── Roles:
│   │       ├── Compute Network Admin → gcp-network-admins@globalmart.com
│   │       └── Security Admin → gcp-security-team@globalmart.com
│   │
│   ├── prd-bigdata-analytics-01
│   │   └── Roles:
│   │       ├── BigQuery Data Editor → gcp-data-engineers@globalmart.com
│   │       ├── BigQuery User → gcp-data-analysts@globalmart.com
│   │       ├── Storage Object Viewer → gcp-data-scientists@globalmart.com
│   │       └── Project Viewer → gcp-all-users@globalmart.com
│   │
│   ├── prd-bigdata-ml-01
│   │   └── Roles:
│   │       ├── Vertex AI User → gcp-data-scientists@globalmart.com
│   │       ├── ML Developer → gcp-ml-engineers@globalmart.com
│   │       └── Notebooks Runner → gcp-data-analysts@globalmart.com
│   │
│   └── prd-bigdata-streaming-01
│       └── Roles:
│           ├── Pub/Sub Editor → gcp-data-engineers@globalmart.com
│           ├── Dataflow Admin → gcp-platform-admins@globalmart.com
│           └── Bigtable User → gcp-app-developers@globalmart.com
│
├── Non-Production (Folder)
│   ├── dev-bigdata-analytics-01
│   ├── stg-bigdata-analytics-01
│   └── qa-bigdata-analytics-01
│
└── Shared-Services (Folder)
    ├── shrd-security-logging-01
    │   └── Roles:
    │       ├── Logging Admin → gcp-security-team@globalmart.com
    │       └── Log Viewer → gcp-all-users@globalmart.com
    │
    ├── shrd-monitoring-ops-01
    │   └── Roles:
    │       ├── Monitoring Admin → gcp-sre-team@globalmart.com
    │       └── Monitoring Viewer → gcp-all-users@globalmart.com
    │
    └── shrd-identity-iam-01
        └── Roles:
            └── Organization Administrator → gcp-org-admins@globalmart.com
```

5.3 User Groups & Roles

Google Groups

Group Name	Members	Purpose
gcp-org-admins@globalmart.com	3	Organization-level administration
gcp-platform-admins@globalmart.com	8	Infrastructure management
gcp-security-team@globalmart.com	5	Security operations & compliance
gcp-network-admins@globalmart.com	4	Network configuration & management
gcp-data-engineers@globalmart.com	15	ETL pipeline development
gcp-data-scientists@globalmart.com	20	ML model development
gcp-data-analysts@globalmart.com	30	Business intelligence & reporting
gcp-data-stewards@globalmart.com	5	Data governance & quality
gcp-ml-engineers@globalmart.com	10	ML ops & model deployment
gcp-app-developers@globalmart.com	12	Application development
gcp-external-partners@globalmart.com	10	Third-party integration
gcp-all-users@globalmart.com	120+	Read-only access

Custom Roles

yaml

```yaml
# Custom Role: Data Pipeline Developer
name: roles/custom.dataPipelineDeveloper
title: Data Pipeline Developer
description: Can create and manage data pipelines
stage: GA
includedPermissions:
  - dataflow.jobs.create
  - dataflow.jobs.update
  - dataflow.jobs.get
  - dataflow.jobs.list
  - dataproc.clusters.create
  - dataproc.clusters.update
  - dataproc.jobs.submit
  - bigquery.jobs.create
  - storage.objects.create
  - storage.objects.get
  - pubsub.topics.publish
  - pubsub.subscriptions.consume

# Custom Role: Data Analyst
name: roles/custom.dataAnalyst
title: Data Analyst
description: Can query and visualize data
stage: GA
includedPermissions:
  - bigquery.jobs.create
  - bigquery.datasets.get
  - bigquery.tables.get
  - bigquery.tables.list
  - bigquery.tables.getData
  - storage.objects.get
  - storage.objects.list

# Custom Role: ML Platform User
name: roles/custom.mlPlatformUser
title: ML Platform User
description: Can develop and train ML models
stage: GA
includedPermissions:
  - aiplatform.models.create
  - aiplatform.models.deploy
  - aiplatform.endpoints.predict
  - aiplatform.trainingPipelines.create
  - notebooks.instances.use
```

```
  - storage.objects.create
  - storage.objects.get
```

5.4 Service Accounts

```yaml
# Dataflow Pipeline Service Account
name: dataflow-compute-prd
email: dataflow-compute-prd@prd-bigdata-streaming-01.iam.gserviceaccount.com
roles:
  - roles/dataflow.worker
  - roles/storage.objectAdmin (on specific buckets)
  - roles/pubsub.subscriber
  - roles/bigquery.dataEditor

# BigQuery Data Loader Service Account
name: bq-loader-prd
email: bq-loader-prd@prd-bigdata-analytics-01.iam.gserviceaccount.com
roles:
  - roles/bigquery.dataEditor
  - roles/bigquery.jobUser
  - roles/storage.objectViewer

# Vertex AI Training Service Account
name: vertex-training-prd
email: vertex-training-prd@prd-bigdata-ml-01.iam.gserviceaccount.com
roles:
  - roles/aiplatform.user
  - roles/storage.objectAdmin (scoped)
  - roles/bigquery.dataViewer

# Cloud Composer (Airflow) Service Account
name: composer-worker-prd
email: composer-worker-prd@prd-bigdata-analytics-01.iam.gserviceaccount.com
roles:
  - roles/composer.worker
  - roles/iam.serviceAccountUser (for impersonation)
  - roles/storage.objectAdmin
```

5.5 VPC Service Controls

```yaml
```

```yaml
# Access Policy
name: globalmart_access_policy
title: GlobalMart Access Policy

# Service Perimeter: Production Data
perimeter:
  name: prd_data_perimeter
  title: Production Data Perimeter
  status: enforced
  resources:
    - projects/prd-bigdata-analytics-01
    - projects/prd-bigdata-ml-01
    - projects/prd-bigdata-streaming-01
  restrictedServices:
    - bigquery.googleapis.com
    - storage.googleapis.com
    - bigtable.googleapis.com
    - pubsub.googleapis.com
  accessLevels:
    - corporate_network
    - trusted_devices
  ingressPolicies:
    - from:
        identities:
          - serviceAccount:dataflow-compute-prd@...
        sources:
          - accessLevel: corporate_network
      to:
        operations:
          - serviceName: bigquery.googleapis.com
            methodSelectors:
              - method: "*"
  egressPolicies:
    - from:
        identities:
          - serviceAccount:vertex-training-prd@...
      to:
        operations:
          - serviceName: storage.googleapis.com
            methodSelectors:
              - method: "google.storage.objects.get"
```

5.6 Security Best Practices Implementation

Encryption

yaml

```yaml
# Cloud KMS Key Ring
keyRing: kr-prd-bigdata-uc1-01
location: us-central1
keys:
  - name: key-prd-bigquery-uc1-01
    purpose: ENCRYPT_DECRYPT
    rotationPeriod: 90 days
    usedBy:
      - BigQuery datasets

  - name: key-prd-storage-uc1-01
    purpose: ENCRYPT_DECRYPT
    rotationPeriod: 90 days
    usedBy:
      - Cloud Storage buckets

  - name: key-prd-secrets-uc1-01
    purpose: ENCRYPT_DECRYPT
    rotationPeriod: 30 days
    usedBy:
      - Secret Manager secrets
```

DLP (Data Loss Prevention)

yaml

```yaml
# DLP Inspect Template
name: pii-detection-template
inspectConfig:
  infoTypes:
    - name: CREDIT_CARD_NUMBER
    - name: EMAIL_ADDRESS
    - name: PHONE_NUMBER
    - name: PERSON_NAME
    - name: US_SOCIAL_SECURITY_NUMBER
  minLikelihood: LIKELY
  limits:
    maxFindingsPerRequest: 0

# DLP Job Trigger
name: scan-prd-raw-data
inspectJob:
  storageConfig:
    cloudStorageOptions:
      fileSet:
        url: gs://gcs-prd-raw-landing-uc1-01/**
  inspectTemplateName: pii-detection-template
  actions:
    - saveFindings:
        outputConfig:
          table:
            projectId: prd-bigdata-analytics-01
            datasetId: dlp_findings
            tableId: scan_results
triggers:
  - schedule:
      recurrencePeriodDuration: 86400s  # Daily
```

---

6. Data Storage Strategy

6.1 Storage Service Selection Matrix

Data Type	Volume	Access Pattern	Latency	Service	Example
Raw transaction logs	50TB/day	Write-once, rare read	Seconds	Cloud Storage (Nearline)	POS transaction files
Processed analytics data	5PB total	Frequent queries	Seconds	BigQuery	Sales fact tables
Real-time clickstream	100M events/day	High write, scan queries	Milliseconds	Bigtable	User activity log

Data Type	Volume	Access Pattern	Latency	Service	Example
Product catalog	500GB	Frequent read/write	Milliseconds	Cloud Spanner	Global product DB
Order metadata	2TB	Transactional	Milliseconds	Cloud SQL (PostgreSQL)	Order processing
User sessions	100GB	Key-value access	Milliseconds	Firestore	Shopping cart state
ML model features	10TB	Batch read for training	Seconds	Cloud Storage (Standard)	Feature store

6.2 Cloud Storage Architecture

yaml

```yaml
# Landing Zone (Raw Data)
bucket: gcs-prd-raw-landing-uc1-01
location: us-central1
storageClass: STANDARD
lifecycleRules:
  - condition:
      age: 7
    action:
      type: SetStorageClass
      storageClass: NEARLINE
  - condition:
      age: 90
    action:
      type: SetStorageClass
      storageClass: COLDLINE
  - condition:
      age: 365
    action:
      type: Delete
encryption:
  defaultKmsKeyName: projects/prd-bigdata-analytics-01/locations/us-central1/keyRings/kr-prd-bigdata-uc1-01/cryptoKeys/k
uniformBucketLevelAccess: true
labels:
  environment: prd
  data_classification: confidential
  zone: landing

# Processing Zone (Bronze/Silver)
bucket: gcs-prd-processed-silver-multi-01
location: US (multi-region)
storageClass: STANDARD
retentionPolicy:
  retentionPeriod: 2592000  # 30 days
versioning: enabled

# Curated Zone (Gold)
bucket: gcs-prd-curated-gold-multi-01
location: US (multi-region)
storageClass: STANDARD
versioning: enabled
labels:
  data_classification: internal
  zone: curated
```

6.3 BigQuery Dataset Architecture

sql

```sql
-- Project: prd-bigdata-analytics-01

-- Dataset: Raw Layer
CREATE SCHEMA bq_prd_retail_raw
OPTIONS (
  location = 'US',
  default_table_expiration_ms = 7776000000,  -- 90 days
  labels = [('layer', 'raw'), ('environment', 'prd')]
);

-- Dataset: Processed Layer
CREATE SCHEMA bq_prd_retail_processed
OPTIONS (
  location = 'US',
  default_kms_key_name = 'projects/prd-bigdata-analytics-01/locations/us/keyRings/kr-prd-bigdata-uc1-01/cryptoKeys/key-p
  labels = [('layer', 'processed'), ('environment', 'prd')]
);

-- Dataset: Analytics Layer (Star Schema)
CREATE SCHEMA bq_prd_retail_analytics
OPTIONS (
  location = 'US',
  labels = [('layer', 'analytics'), ('environment', 'prd')]
);

-- Fact Table: Sales Transactions
CREATE TABLE bq_prd_retail_analytics.sales_fact (
  transaction_id STRING NOT NULL,
  transaction_date DATE NOT NULL,
  transaction_timestamp TIMESTAMP NOT NULL,
  store_key INT64 NOT NULL,
  product_key INT64 NOT NULL,
  customer_key INT64,
  quantity INT64 NOT NULL,
  unit_price NUMERIC(10, 2) NOT NULL,
  discount_amount NUMERIC(10, 2),
  tax_amount NUMERIC(10, 2),
  total_amount NUMERIC(10, 2) NOT NULL,
  payment_method STRING,
  -- Partitioning
  PARTITION BY transaction_date,
  -- Clustering
  CLUSTER BY store_key, product_key
)
OPTIONS (
  partition_expiration_days = 1095,  -- 3 years
```

```
  require_partition_filter = true,
  labels = [('table_type', 'fact')]
);

-- Dimension Table: Store
CREATE TABLE bq_prd_retail_analytics.store_dim (
  store_key INT64 NOT NULL,
  store_id STRING NOT NULL,
  store_name STRING,
  store_type STRING,
  region STRING,
  country STRING,
  city STRING,
  postal_code STRING,
  latitude FLOAT64,
  longitude FLOAT64,
  opened_date DATE,
  manager_name STRING,
  effective_date DATE NOT NULL,
  expiration_date DATE,
  is_current BOOL NOT NULL
)
OPTIONS (
  labels = [('table_type', 'dimension'), ('scd_type', 'type2')]
);

-- Dimension Table: Product
CREATE TABLE bq_prd_retail_analytics.product_dim (
  product_key INT64 NOT NULL,
  product_id STRING NOT NULL,
  product_name STRING,
  category STRING,
  subcategory STRING,
  brand STRING,
  supplier STRING,
  unit_cost NUMERIC(10, 2),
  effective_date DATE NOT NULL,
  expiration_date DATE,
  is_current BOOL NOT NULL
);

-- Dimension Table: Customer
CREATE TABLE bq_prd_retail_analytics.customer_dim (
  customer_key INT64 NOT NULL,
  customer_id STRING NOT NULL,
  -- PII fields (consider column-level encryption)
  first_name STRING,
```

```
  last_name STRING,
  email STRING,
  phone STRING,
  -- Demographics
  date_of_birth DATE,
  gender STRING,
  income_bracket STRING,
  -- Location
  city STRING,
  state STRING,
  country STRING,
  postal_code STRING,
  -- SCD Type 2
  effective_date DATE NOT NULL,
  expiration_date DATE,
  is_current BOOL NOT NULL
)
OPTIONS (
  labels = [('contains_pii', 'true')]
);

-- Aggregate Table: Daily Store Sales
CREATE MATERIALIZED VIEW bq_prd_retail_analytics.daily_store_sales_mv
PARTITION BY sales_date
CLUSTER BY store_key
AS
SELECT
  transaction_date AS sales_date,
  store_key,
  COUNT(DISTINCT transaction_id) AS transaction_count,
  COUNT(DISTINCT customer_key) AS customer_count,
  SUM(quantity) AS total_quantity,
  SUM(total_amount) AS total_revenue,
  AVG(total_amount) AS avg_transaction_value
FROM bq_prd_retail_analytics.sales_fact
GROUP BY sales_date, store_key;

-- Row-Level Security Policy
CREATE ROW ACCESS POLICY regional_data_policy
ON bq_prd_retail_analytics.sales_fact
GRANT TO ('group:gcp-data-analysts-us@globalmart.com')
FILTER USING (store_key IN (
  SELECT store_key
  FROM bq_prd_retail_analytics.store_dim
  WHERE country = 'US' AND is_current = TRUE
));
```

6.4 Cloud Spanner Schema

sql

```sql
-- Instance: span-prd-orders-multiregional-01
-- Configuration: nam-eur-asia1 (multi-region)

-- Table: Orders (Parent)
CREATE TABLE Orders (
  order_id STRING(36) NOT NULL,
  customer_id STRING(36) NOT NULL,
  order_timestamp TIMESTAMP NOT NULL OPTIONS (allow_commit_timestamp=true),
  status STRING(20) NOT NULL,
  total_amount NUMERIC NOT NULL,
  currency STRING(3) NOT NULL,
  shipping_address_id STRING(36),
  billing_address_id STRING(36),
  created_at TIMESTAMP NOT NULL OPTIONS (allow_commit_timestamp=true),
  updated_at TIMESTAMP NOT NULL OPTIONS (allow_commit_timestamp=true)
) PRIMARY KEY (order_id);

-- Table: OrderItems (Child - Interleaved)
CREATE TABLE OrderItems (
  order_id STRING(36) NOT NULL,
  item_id INT64 NOT NULL,
  product_id STRING(36) NOT NULL,
  quantity INT64 NOT NULL,
  unit_price NUMERIC NOT NULL,
  discount_amount NUMERIC,
  tax_amount NUMERIC,
  subtotal NUMERIC NOT NULL
) PRIMARY KEY (order_id, item_id),
  INTERLEAVE IN PARENT Orders ON DELETE CASCADE;

-- Secondary Index
CREATE INDEX idx_orders_customer_timestamp
ON Orders(customer_id, order_timestamp DESC);

CREATE INDEX idx_orders_status_timestamp
ON Orders(status, order_timestamp DESC);
```

6.5 Bigtable Schema Design

yaml

```yaml
# Instance: bt-prd-timeseries-uc1-01
instance_id: bt-prd-timeseries-uc1-01
cluster_id: bt-prd-timeseries-uc1-cluster-01
zone: us-central1-a
num_nodes: 10
storage_type: SSD

# Table: user_events
table_id: user_events
column_families:
  - name: metrics
    gc_rule:
      max_age: 30d  # 30 days retention
  - name: attributes
    gc_rule:
      max_versions: 3

# Row Key Design: {user_id}#{reverse_timestamp}#{event_type}
# Example: user_12345#9223370451257000000#page_view
#
# Benefits:
# - Scans all events for a user efficiently
# - Reverse timestamp ensures latest events come first
# - Event type allows filtering within user's events

# Sample Data Structure:
# Row Key: user_12345#9223370451257000000#page_view
#   metrics:duration = 3500
#   metrics:scroll_depth = 85
#   attributes:page_url = "/products/widget"
#   attributes:referrer = "google.com"
#   attributes:device = "mobile"
```

---

7. Networking Architecture

7.1 Load Balancer Configuration

Global External Application Load Balancer

yaml

```yaml
name: lb-prd-global-app-01
loadBalancingScheme: EXTERNAL_MANAGED
protocol: HTTPS
tier: STANDARD  # or PREMIUM for global anycast

# Frontend Configuration
frontend:
  ipAddress: 34.120.50.100  # Reserved static IP
  port: 443
  sslCertificates:
    - projects/prd-network-host-01/global/sslCertificates/cert-prd-globalmart-01
  sslPolicy: tls-1-2-minimum

# Backend Services
backends:
  - name: backend-api-gateway
    protocol: HTTPS
    backends:
      - group: instance-group-api-uc1-01
        balancingMode: RATE
        maxRatePerInstance: 1000
      - group: instance-group-api-ew1-01
        balancingMode: RATE
        maxRatePerInstance: 1000
    healthCheck:
      checkIntervalSec: 10
      timeoutSec: 5
      healthyThreshold: 2
      unhealthyThreshold: 3
      port: 443
      requestPath: /health
    sessionAffinity: CLIENT_IP
    affinityCookieTtlSec: 3600

  - name: backend-analytics-ui
    protocol: HTTPS
    backends:
      - group: instance-group-looker-uc1-01
    iap:
      enabled: true
      oauth2ClientId: abc123.apps.googleusercontent.com
      oauth2ClientSecret: [SECRET]

# URL Map
urlMap:
  defaultService: backend-api-gateway
```

```yaml
  hostRules:
    - hosts:
        - api.globalmart.com
      pathMatcher: api-paths
    - hosts:
        - analytics.globalmart.com
      pathMatcher: analytics-paths
  pathMatchers:
    - name: api-paths
      defaultService: backend-api-gateway
      routeRules:
        - priority: 1
          matchRules:
            - prefixMatch: /v1/
          routeAction:
            weightedBackendServices:
              - backendService: backend-api-gateway
                weight: 100
    - name: analytics-paths
      defaultService: backend-analytics-ui

# Cloud CDN
cdnPolicy:
  cacheMode: CACHE_ALL_STATIC
  defaultTtl: 3600
  maxTtl: 86400
  clientTtl: 3600
  negativeCaching: true
```

Regional Internal Application Load Balancer

yaml

```yaml
name: lb-prd-uc1-app-internal-01
region: us-central1
loadBalancingScheme: INTERNAL_MANAGED
protocol: HTTP
network: vpc-prd-shared-global-01
subnetwork: subnet-prd-compute-uc1-01

# Backend Service
backendService:
  protocol: HTTP
  backends:
    - group: instance-group-dataproc-master-uc1-01
      balancingMode: UTILIZATION
      maxUtilization: 0.8
  healthCheck:
    checkIntervalSec: 10
    timeoutSec: 5
    port: 8088
    requestPath: /healthz
  sessionAffinity: CLIENT_IP

# Frontend
frontend:
  IPAddress: 10.10.1.100  # Internal IP from subnet
  port: 80
```

Regional Network Load Balancer (for TCP/UDP)

```yaml
```

```yaml
name: lb-prd-uc1-net-01
region: us-central1
loadBalancingScheme: EXTERNAL
protocol: TCP
network: vpc-prd-shared-global-01
subnetwork: subnet-prd-compute-uc1-01

# Backend Service
backendService:
  protocol: TCP
  backends:
    - group: instance-group-streaming-uc1-01
      balancingMode: CONNECTION
  healthCheck:
    checkIntervalSec: 10
    timeoutSec: 5
    port: 9092
  sessionAffinity: CLIENT_IP_PORT_PROTO

# Frontend
frontend:
  IPAddress: 34.120.50.200  # External static IP
  ports: [9092, 9093]  # Kafka ports
```

7.2 Private Google Access & Private Service Connect

yaml

```yaml
# Enable Private Google Access for subnets
- subnet: subnet-prd-data-uc1-01
  privateGoogleAccess: true
  purpose: Access GCS, BigQuery without public IPs

# Private Service Connect Endpoint
- name: psc-prd-bigquery-uc1-01
  network: vpc-prd-shared-global-01
  subnetwork: subnet-prd-data-uc1-01
  targetService: servicedirectory.googleapis.com
  ipAddress: 10.10.0.50
```

7.3 Cloud Interconnect / VPN

yaml

```yaml
# Dedicated Interconnect (for on-premises connectivity)
interconnect:
  name: ic-prd-onprem-uc1-01
  type: DEDICATED
  location: las-zone1-1
  linkType: LINK_TYPE_ETHERNET_10G_LR

  # VLAN Attachments
  vlanAttachments:
    - name: vlan-prd-onprem-uc1-01
      vlan: 1000
      peeringIp: 169.254.100.1/29
      customerRouterIp: 169.254.100.2/29
      cloudRouterName: rtr-prd-uc1-01
      bandwidth: 10Gbps

# Cloud VPN (backup/secondary path)
vpnGateway:
  name: vpn-gw-prd-uc1-01
  network: vpc-prd-shared-global-01
  region: us-central1

  # VPN Tunnels
  tunnels:
    - name: vpn-tunnel-prd-onprem-01
      peerIp: 203.0.113.5
      sharedSecret: [SECRET]
      ikeVersion: 2
      cloudRouter: rtr-prd-uc1-01
      bgpPeerAsn: 65001
```

7.4 DNS Configuration

yaml

```yaml
# Cloud DNS Zones
- name: dns-prd-globalmart-internal
  dnsName: globalmart.internal.
  visibility: private
  networks:
    - vpc-prd-shared-global-01

  records:
    - name: bigquery.globalmart.internal.
      type: A
      rrdatas: [10.10.0.50]  # Private Service Connect IP

    - name: dataproc-master.globalmart.internal.
      type: A
      rrdatas: [10.10.1.100]  # Internal LB IP

    - name: bigtable.globalmart.internal.
      type: CNAME
      rrdatas: [bt-prd-timeseries-uc1-01.googleapis.com]
```

---

8. Big Data Components

8.1 Pub/Sub Topics & Subscriptions

yaml

```yaml
# Topic: Real-time POS Transactions
topic:
  name: topic-prd-pos-transactions-uc1-01
  labels:
    environment: prd
    data_source: pos_systems
  messageStoragePolicy:
    allowedPersistenceRegions:
      - us-central1
  schemaSettings:
    schema: projects/prd-bigdata-streaming-01/schemas/pos-transaction-v1
    encoding: JSON

# Subscription: Dataflow Streaming Pipeline
subscription:
  name: sub-prd-dataflow-realtime-uc1-01
  topic: topic-prd-pos-transactions-uc1-01
  ackDeadlineSeconds: 60
  retainAckedMessages: false
  messageRetentionDuration: 604800s  # 7 days
  expirationPolicy:
    ttl: never
  deadLetterPolicy:
    deadLetterTopic: topic-prd-dlq-uc1-01
    maxDeliveryAttempts: 5

# Topic: Inventory Updates
topic:
  name: topic-prd-inventory-updates-multi-01
  labels:
    environment: prd
    data_source: inventory_system

# Subscription: Bigtable Writer
subscription:
  name: sub-prd-bigtable-writer-uc1-01
  topic: topic-prd-inventory-updates-multi-01
  ackDeadlineSeconds: 120
  enableExactlyOnceDelivery: true
```

8.2 Dataflow Pipelines

Streaming Pipeline: Real-time Transaction Processing

python

```
# Pipeline: df-prd-stream-processor-uc1-01

import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions
from apache_beam.io import ReadFromPubSub, WriteToBigQuery, WriteToBigTable

class ParseTransaction(beam.DoFn):
    def process(self, element):
        import json
        data = json.loads(element.decode('utf-8'))

        # Data validation
        if self.validate(data):
            yield {
                'transaction_id': data['transaction_id'],
                'timestamp': data['timestamp'],
                'store_id': data['store_id'],
                'amount': float(data['amount']),
                'items': data['items']
            }

    def validate(self, data):
        required_fields = ['transaction_id', 'timestamp', 'store_id', 'amount']
        return all(field in data for field in required_fields)

class EnrichWithStoreDim(beam.DoFn):
    def process(self, element):
        # Lookup store dimension from BigQuery
        from google.cloud import bigquery
        client = bigquery.Client()

        query = f"""
        SELECT store_key, region, country
        FROM `bq_prd_retail_analytics.store_dim`
        WHERE store_id = '{element['store_id']}' AND is_current = TRUE
        """
        result = client.query(query).result()
        store_info = next(result, None)

        if store_info:
            element['store_key'] = store_info.store_key
            element['region'] = store_info.region
            element['country'] = store_info.country

        yield element
```

```
pipeline_options = PipelineOptions([
    '--project=prd-bigdata-streaming-01',
    '--region=us-central1',
    '--runner=DataflowRunner',
    '--streaming',
    '--service_account_email=dataflow-compute-prd@prd-bigdata-streaming-01.iam.gserviceaccount.com',
    '--network=vpc-prd-shared-global-01',
    '--subnetwork=regions/us-central1/subnetworks/subnet-prd-compute-uc1-01',
    '--max_num_workers=50',
    '--autoscaling_algorithm=THROUGHPUT_BASED',
])

with beam.Pipeline(options=pipeline_options) as p:
    # Read from Pub/Sub
    transactions = (
        p
        | 'Read from Pub/Sub' >> ReadFromPubSub(
            subscription='projects/prd-bigdata-streaming-01/subscriptions/sub-prd-dataflow-realtime-uc1-01'
        )
        | 'Parse JSON' >> beam.ParDo(ParseTransaction())
        | 'Enrich with Store Data' >> beam.ParDo(EnrichWithStoreDim())
    )

    # Write to BigQuery
    transactions | 'Write to BigQuery' >> WriteToBigQuery(
        table='prd-bigdata-analytics-01:bq_prd_retail_raw.transactions',
        write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND,
        create_disposition=beam.io.BigQueryDisposition.CREATE_NEVER
    )

    # Write to Bigtable (for real-time dashboard)
    transactions | 'Write to Bigtable' >> WriteToBigTable(
        project_id='prd-bigdata-streaming-01',
        instance_id='bt-prd-timeseries-uc1-01',
        table_id='realtime_transactions'
    )
```

Batch Pipeline: Daily Aggregation

python

```
# Pipeline: df-prd-batch-aggregator-uc1-01

import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions

pipeline_options = PipelineOptions([
    '--project=prd-bigdata-analytics-01',
    '--region=us-central1',
    '--runner=DataflowRunner',
    '--temp_location=gs://gcs-prd-temp-dataflow-uc1-01/temp',
    '--staging_location=gs://gcs-prd-temp-dataflow-uc1-01/staging',
    '--max_num_workers=100',
])

with beam.Pipeline(options=pipeline_options) as p:
    # Read from BigQuery
    daily_sales = (
        p
        | 'Read Raw Transactions' >> beam.io.ReadFromBigQuery(
            query="""
            SELECT
                DATE(transaction_timestamp) AS sales_date,
                store_key,
                product_key,
                SUM(quantity) AS total_quantity,
                SUM(total_amount) AS total_revenue
            FROM `prd-bigdata-analytics-01.bq_prd_retail_raw.transactions`
            WHERE DATE(transaction_timestamp) = CURRENT_DATE() - 1
            GROUP BY sales_date, store_key, product_key
            """,
            use_standard_sql=True
        )
        | 'Write Aggregates' >> beam.io.WriteToBigQuery(
            table='prd-bigdata-analytics-01:bq_prd_retail_analytics.daily_product_sales',
            write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND
        )
    )
```

8.3 Dataproc Clusters

yaml

```yaml
# Ephemeral Cluster for Spark Jobs
cluster:
  name: dpc-prd-batch-etl-uc1-01
  region: us-central1

  config:
    masterConfig:
      numInstances: 1
      machineTypeUri: n1-standard-8
      diskConfig:
        bootDiskType: pd-standard
        bootDiskSizeGb: 500

    workerConfig:
      numInstances: 10
      machineTypeUri: n1-standard-16
      diskConfig:
        bootDiskType: pd-standard
        bootDiskSizeGb: 500

    secondaryWorkerConfig:
      numInstances: 20
      machineTypeUri: n1-standard-16
      isPreemptible: true

    softwareConfig:
      imageVersion: 2.1-debian11
      properties:
        spark:spark.executor.memory: 10g
        spark:spark.executor.cores: 4
        spark:spark.dynamicAllocation.enabled: true

    initializationActions:
      - executableFile: gs://gcs-prd-dataproc-init-uc1-01/install-dependencies.sh

    autoscalingConfig:
      policyUri: projects/prd-bigdata-analytics-01/regions/us-central1/autoscalingPolicies/policy-prd-spark-autoscale-01

  # Network Configuration
  gceClusterConfig:
    networkUri: vpc-prd-shared-global-01
    subnetworkUri: subnet-prd-compute-uc1-01
    internalIpOnly: true
    serviceAccount: dataproc-compute-prd@prd-bigdata-analytics-01.iam.gserviceaccount.com
    tags:
      - dataproc-cluster
```

```
    - allow-internal

  # Lifecycle
  lifecycleConfig:
    idleDeleteTtl: 3600s  # Delete after 1 hour of inactivity
```

Sample Spark Job:

```python
```

```
# Job: Customer Segmentation ETL

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum, avg, count, datediff, current_date

spark = SparkSession.builder \
    .appName("customer-segmentation-etl") \
    .getOrCreate()

# Read from BigQuery
sales_df = spark.read \
    .format("bigquery") \
    .option("table", "prd-bigdata-analytics-01:bq_prd_retail_analytics.sales_fact") \
    .option("filter", "transaction_date >= DATE_SUB(CURRENT_DATE(), 365)") \
    .load()

customer_df = spark.read \
    .format("bigquery") \
    .option("table", "prd-bigdata-analytics-01:bq_prd_retail_analytics.customer_dim") \
    .option("filter", "is_current = TRUE") \
    .load()

# Calculate RFM metrics
rfm_df = sales_df.groupBy("customer_key").agg(
    datediff(current_date(), max("transaction_date")).alias("recency"),
    count("transaction_id").alias("frequency"),
    sum("total_amount").alias("monetary_value")
)

# Join with customer dimension
result_df = rfm_df.join(customer_df, "customer_key")

# Write back to BigQuery
result_df.write \
    .format("bigquery") \
    .option("table", "prd-bigdata-analytics-01:bq_prd_retail_analytics.customer_rfm_segments") \
    .option("writeMethod", "direct") \
    .mode("overwrite") \
    .save()

spark.stop()
```

8.4 Cloud Composer (Airflow)

```
python
```

```
# DAG: daily_retail_analytics_pipeline
# File: /dags/daily_retail_analytics_pipeline.py

from airflow import DAG
from airflow.providers.google.cloud.operators.dataflow import DataflowTemplatedJobStartOperator
from airflow.providers.google.cloud.operators.bigquery import BigQueryInsertJobOperator
from airflow.providers.google.cloud.operators.dataproc import (
    DataprocCreateClusterOperator,
    DataprocSubmitJobOperator,
    DataprocDeleteClusterOperator
)
from airflow.providers.google.cloud.sensors.bigquery import BigQueryTableExistenceSensor
from airflow.utils.dates import days_ago
from datetime import timedelta

default_args = {
    'owner': 'data-engineering-team',
    'depends_on_past': False,
    'email': ['data-alerts@globalmart.com'],
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG(
    'daily_retail_analytics_pipeline',
    default_args=default_args,
    description='Daily batch processing for retail analytics',
    schedule_interval='0 2 * * *',  # 2 AM daily
    start_date=days_ago(1),
    catchup=False,
    tags=['production', 'retail', 'daily'],
)

# Task 1: Check if raw data arrived
check_raw_data = BigQueryTableExistenceSensor(
    task_id='check_raw_data',
    project_id='prd-bigdata-analytics-01',
    dataset_id='bq_prd_retail_raw',
    table_id='transactions',
    dag=dag,
)

# Task 2: Run Dataflow batch aggregation
run_dataflow_aggregation = DataflowTemplatedJobStartOperator(
```

```
    task_id='run_dataflow_aggregation',
    project_id='prd-bigdata-analytics-01',
    template='gs://dataflow-templates/latest/BigQuery_to_BigQuery',
    location='us-central1',
    parameters={
        'inputTable': 'prd-bigdata-analytics-01:bq_prd_retail_raw.transactions',
        'outputTable': 'prd-bigdata-analytics-01:bq_prd_retail_processed.transactions_aggregated',
    },
    dag=dag,
)

# Task 3: Create Dataproc cluster
create_dataproc_cluster = DataprocCreateClusterOperator(
    task_id='create_dataproc_cluster',
    project_id='prd-bigdata-analytics-01',
    region='us-central1',
    cluster_name='dpc-prd-batch-etl-{{ ds_nodash }}',
    cluster_config={
        'master_config': {'num_instances': 1, 'machine_type_uri': 'n1-standard-8'},
        'worker_config': {'num_instances': 10, 'machine_type_uri': 'n1-standard-16'},
        'gce_cluster_config': {
            'subnetwork_uri': 'subnet-prd-compute-uc1-01',
            'internal_ip_only': True,
        },
    },
    dag=dag,
)

# Task 4: Submit Spark job
submit_spark_job = DataprocSubmitJobOperator(
    task_id='submit_spark_job',
    project_id='prd-bigdata-analytics-01',
    region='us-central1',
    cluster_name='dpc-prd-batch-etl-{{ ds_nodash }}',
    job={
        'reference': {'project_id': 'prd-bigdata-analytics-01'},
        'placement': {'cluster_name': 'dpc-prd-batch-etl-{{ ds_nodash }}'},
        'pyspark_job': {
            'main_python_file_uri': 'gs://gcs-prd-dataproc-jobs-uc1-01/customer_segmentation_etl.py',
            'args': ['--execution_date={{ ds }}'],
        },
    },
    dag=dag,
)

# Task 5: Delete Dataproc cluster
delete_dataproc_cluster = DataprocDeleteClusterOperator(
```

```
    task_id='delete_dataproc_cluster',
    project_id='prd-bigdata-analytics-01',
    region='us-central1',
    cluster_name='dpc-prd-batch-etl-{{ ds_nodash }}',
    trigger_rule='all_done',
    dag=dag,
)

# Task 6: Update BigQuery materialized views
refresh_materialized_views = BigQueryInsertJobOperator(
    task_id='refresh_materialized_views',
    configuration={
        'query': {
            'query': """
                -- Refresh daily_store_sales_mv
                CALL `prd-bigdata-analytics-01.bq_prd_retail_analytics`.refresh_materialized_view(
                    'daily_store_sales_mv'
                );
            """,
            'useLegacySql': False,
        }
    },
    dag=dag,
)

# Task dependencies
check_raw_data >> run_dataflow_aggregation >> create_dataproc_cluster
create_dataproc_cluster >> submit_spark_job >> delete_dataproc_cluster
delete_dataproc_cluster >> refresh_materialized_views
```

8.5 Vertex AI ML Pipeline

python

```python
# ML Pipeline: Customer Churn Prediction
# File: vertex_ai_churn_pipeline.py

from google.cloud import aiplatform
from kfp.v2 import dsl
from kfp.v2.dsl import component, Dataset, Model, Metrics

PROJECT_ID = 'prd-bigdata-ml-01'
REGION = 'us-central1'
PIPELINE_ROOT = 'gs://gcs-prd-ml-pipelines-uc1-01/churn_prediction'

@component(
    base_image='gcr.io/prd-bigdata-ml-01/ml-base:latest',
    packages_to_install=['google-cloud-bigquery', 'pandas', 'scikit-learn']
)
def extract_features(
    project_id: str,
    dataset_id: str,
    output_dataset: dsl.Output[Dataset]
):
    from google.cloud import bigquery
    import pandas as pd

    client = bigquery.Client(project=project_id)

    query = f"""
    SELECT
        customer_key,
        recency,
        frequency,
        monetary_value,
        avg_transaction_value,
        days_since_first_purchase,
        product_category_diversity,
        churned  -- Target variable
    FROM `{project_id}.{dataset_id}.customer_features`
    WHERE feature_date = CURRENT_DATE() - 1
    """

    df = client.query(query).to_dataframe()
    df.to_csv(output_dataset.path, index=False)

@component(
    base_image='gcr.io/prd-bigdata-ml-01/ml-base:latest',
    packages_to_install=['scikit-learn', 'pandas', 'joblib']
)
```

```
def train_model(
    input_dataset: dsl.Input[Dataset],
    model: dsl.Output[Model],
    metrics: dsl.Output[Metrics]
):
    import pandas as pd
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
    import joblib

    df = pd.read_csv(input_dataset.path)

    X = df.drop(['customer_key', 'churned'], axis=1)
    y = df['churned']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    clf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    metrics.log_metric('accuracy', accuracy)
    metrics.log_metric('precision', precision)
    metrics.log_metric('recall', recall)
    metrics.log_metric('f1_score', f1)

    joblib.dump(clf, model.path)

@component(
    base_image='gcr.io/prd-bigdata-ml-01/ml-base:latest',
    packages_to_install=['google-cloud-aiplatform']
)
def deploy_model(
    model: dsl.Input[Model],
    project_id: str,
    region: str,
    endpoint_display_name: str
):
    from google.cloud import aiplatform
```

```
    aiplatform.init(project=project_id, location=region)

    uploaded_model = aiplatform.Model.upload(
        display_name='churn-prediction-model',
        artifact_uri=model.uri,
        serving_container_image_uri='gcr.io/cloud-aiplatform/prediction/sklearn-cpu.1-0:latest'
    )

    endpoint = aiplatform.Endpoint.create(display_name=endpoint_display_name)

    uploaded_model.deploy(
        endpoint=endpoint,
        machine_type='n1-standard-4',
        min_replica_count=2,
        max_replica_count=10,
        traffic_percentage=100
    )

@dsl.pipeline(
    name='customer-churn-prediction-pipeline',
    description='End-to-end ML pipeline for customer churn prediction',
    pipeline_root=PIPELINE_ROOT
)
def churn_prediction_pipeline(
    project_id: str = PROJECT_ID,
    dataset_id: str = 'bq_prd_ml_features',
    endpoint_display_name: str = 'churn-prediction-endpoint'
):
    extract_task = extract_features(
        project_id=project_id,
        dataset_id=dataset_id
    )

    train_task = train_model(
        input_dataset=extract_task.outputs['output_dataset']
    )

    deploy_task = deploy_model(
        model=train_task.outputs['model'],
        project_id=project_id,
        region=REGION,
        endpoint_display_name=endpoint_display_name
    )

# Compile and submit pipeline
from kfp.v2 import compiler
```

```
compiler.Compiler().compile(
    pipeline_func=churn_prediction_pipeline,
    package_path='churn_prediction_pipeline.json'
)

aiplatform.init(project=PROJECT_ID, location=REGION)

job = aiplatform.PipelineJob(
    display_name='churn-prediction-pipeline-run',
    template_path='churn_prediction_pipeline.json',
    pipeline_root=PIPELINE_ROOT,
    enable_caching=False
)

job.submit(service_account='vertex-training-prd@prd-bigdata-ml-01.iam.gserviceaccount.com')
```

---

9. Implementation Details

9.1 Data Ingestion Patterns

Batch Ingestion from On-Premises

yaml

```yaml
# Transfer Service Job
transferJob:
  name: transfer-prd-onprem-daily-sales
  description: Daily transfer of sales files from on-prem
  schedule:
    scheduleStartDate: 2025-01-01
    scheduleEndDate: 2026-01-01
    startTimeOfDay: 02:00:00
  transferSpec:
    posixDataSource:
      rootDirectory: /mnt/sales_export
      transferAgentPoolName: projects/prd-bigdata-analytics-01/transferAgentPools/pool-onprem-01
    gcsDataSink:
      bucketName: gcs-prd-raw-landing-uc1-01
      path: sales/daily/
  notificationConfig:
    pubsubTopic: projects/prd-bigdata-analytics-01/topics/topic-prd-transfer-notifications-uc1-01
    eventTypes:
      - TRANSFER_OPERATION_SUCCESS
      - TRANSFER_OPERATION_FAILED
```

Streaming Ingestion from APIs

python

```python
# Cloud Run Service: API Data Collector
# File: main.py

from flask import Flask, request, jsonify
from google.cloud import pubsub_v1
import json
import logging

app = Flask(__name__)
logging.basicConfig(level=logging.INFO)

PROJECT_ID = 'prd-bigdata-streaming-01'
TOPIC_ID = 'topic-prd-pos-transactions-uc1-01'

publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(PROJECT_ID, TOPIC_ID)

@app.route('/ingest/transaction', methods=['POST'])
def ingest_transaction():
    try:
        data = request.get_json()

        # Validate required fields
        required_fields = ['transaction_id', 'timestamp', 'store_id', 'amount']
        if not all(field in data for field in required_fields):
            return jsonify({'error': 'Missing required fields'}), 400

        # Publish to Pub/Sub
        message_json = json.dumps(data)
        message_bytes = message_json.encode('utf-8')

        future = publisher.publish(topic_path, message_bytes)
        message_id = future.result()

        logging.info(f'Published message {message_id} for transaction {data["transaction_id"]}')

        return jsonify({
            'status': 'success',
            'message_id': message_id,
            'transaction_id': data['transaction_id']
        }), 200

    except Exception as e:
        logging.error(f'Error ingesting transaction: {str(e)}')
        return jsonify({'error': str(e)}), 500
```

```
@app.route('/health', methods=['GET'])
def health():
    return jsonify({'status': 'healthy'}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

9.2 Data Quality & Validation

```sql
```

```sql
-- Data Quality Rules in BigQuery
-- Project: prd-bigdata-analytics-01
-- Dataset: bq_prd_data_quality

-- Table: Data Quality Rules
CREATE TABLE bq_prd_data_quality.dq_rules (
  rule_id STRING NOT NULL,
  rule_name STRING NOT NULL,
  table_name STRING NOT NULL,
  column_name STRING,
  rule_type STRING NOT NULL,  -- NULL_CHECK, RANGE_CHECK, PATTERN_CHECK, UNIQUENESS_CHECK
  rule_definition STRING NOT NULL,
  severity STRING NOT NULL,  -- ERROR, WARNING, INFO
  active BOOL NOT NULL DEFAULT TRUE
);

-- Insert sample rules
INSERT INTO bq_prd_data_quality.dq_rules VALUES
('DQ001', 'Transaction ID Not Null', 'sales_fact', 'transaction_id', 'NULL_CHECK',
 'transaction_id IS NOT NULL', 'ERROR', TRUE),
('DQ002', 'Transaction Amount Positive', 'sales_fact', 'total_amount', 'RANGE_CHECK',
 'total_amount > 0', 'ERROR', TRUE),
('DQ003', 'Valid Payment Method', 'sales_fact', 'payment_method', 'PATTERN_CHECK',
 'payment_method IN ("CASH", "CREDIT_CARD", "DEBIT_CARD", "DIGITAL_WALLET")', 'WARNING', TRUE);

-- Stored Procedure: Run Data Quality Checks
CREATE OR REPLACE PROCEDURE bq_prd_data_quality.run_dq_checks(
  target_table STRING,
  check_date DATE
)
BEGIN
  DECLARE rule STRUCT<rule_id STRING, rule_definition STRING, severity STRING>;
  DECLARE results_count INT64;

  FOR rule IN (
    SELECT rule_id, rule_definition, severity
    FROM bq_prd_data_quality.dq_rules
    WHERE table_name = target_table AND active = TRUE
  ) DO
    EXECUTE IMMEDIATE FORMAT("""
      INSERT INTO bq_prd_data_quality.dq_results
      SELECT
        '%s' AS rule_id,
        CURRENT_TIMESTAMP() AS check_timestamp,
        '%s' AS target_table,
        DATE('%t') AS check_date,
```

```
      COUNT(*) AS failure_count,
      '%s' AS severity
    FROM `%s`
    WHERE DATE(transaction_date) = '%t'
      AND NOT (%s)
  """, rule.rule_id, target_table, check_date, rule.severity, target_table, check_date, rule.rule_definition);
 END FOR;
END;
```

---

10. Cost Optimization

10.1 Cost Allocation & Chargeback

yaml

```yaml
# Label Strategy for Cost Attribution
standardLabels:
  environment: [prd, nprd, dev, stg, qa]
  cost_center: [retail, supply-chain, finance, marketing]
  team: [data-engineering, data-science, analytics, platform]
  project_code: [PROJ-001, PROJ-002, ...]
  workload_type: [batch, streaming, ml, analytics]

# BigQuery Cost Controls
bigQueryQuotas:
  - project: prd-bigdata-analytics-01
    quotas:
      - metric: bigquery.googleapis.com/quota/query/usage
        limit: 10000  # GB per day
        users:
          - group:gcp-data-analysts@globalmart.com

      - metric: bigquery.googleapis.com/quota/query/usage
        limit: 50000  # GB per day
        users:
          - group:gcp-data-engineers@globalmart.com

# Cost Budgets & Alerts
budgets:
  - displayName: Monthly Production Budget
    amount:
      specifiedAmount:
        currencyCode: USD
        units: 100000
    budgetFilter:
      projects:
        - projects/prd-bigdata-analytics-01
        - projects/prd-bigdata-ml-01
        - projects/prd-bigdata-streaming-01
      labels:
        environment: prd
    thresholdRules:
      - thresholdPercent: 0.5
        spendBasis: CURRENT_SPEND
      - thresholdPercent: 0.8
        spendBasis: CURRENT_SPEND
      - thresholdPercent: 1.0
        spendBasis: CURRENT_SPEND
    notificationsRule:
      pubsubTopic: projects/shrd-monitoring-ops-01/topics/topic-budget-alerts
```

```
  monitoringNotificationChannels:
    - projects/shrd-monitoring-ops-01/notificationChannels/email-finance-team
```

10.2 Resource Optimization

yaml

```yaml
# Committed Use Discounts (CUDs)
commitments:
  - type: COMPUTE
    plan: TWELVE_MONTH
    resources:
      - resourceType: VCPU
        amount: 500
      - resourceType: MEMORY
        amount: 2000  # GB
    region: us-central1

  - type: MEMORY_OPTIMIZED_COMPUTE
    plan: THIRTY_SIX_MONTH
    resources:
      - resourceType: VCPU
        amount: 200
      - resourceType: MEMORY
        amount: 1600
    region: us-central1

# BigQuery Flat-Rate Pricing
bigQueryReservations:
  - name: reservation-prd-analytics-uc1-01
    slotCapacity: 2000
    location: us-central1
    assignments:
      - project: prd-bigdata-analytics-01
        jobType: QUERY
      - project: prd-bigdata-ml-01
        jobType: QUERY

# Dataproc Autoscaling Policy
dataprocAutoscaling:
  - name: policy-prd-spark-autoscale-01
    workerConfig:
      minInstances: 10
      maxInstances: 100
      weight: 1
    secondaryWorkerConfig:
      minInstances: 0
      maxInstances: 200
      weight: 1
    basicAlgorithm:
      yarnConfig:
        scaleUpFactor: 0.05
        scaleDownFactor: 1.0
```

```
    scaleUpMinWorkerFraction: 0.0
    scaleDownMinWorkerFraction: 0.0
    gracefulDecommissionTimeout: 1h
```

---

11. Monitoring & Operations

11.1 Cloud Monitoring Dashboards

```yaml
```

```yaml
# Dashboard: BigData Platform Health
dashboard:
  displayName: BigData Platform - Production
  mosaicLayout:
    columns: 12
    tiles:
      # Pub/Sub Metrics
      - width: 6
        height: 4
        widget:
          title: Pub/Sub Message Throughput
          xyChart:
            dataSets:
              - timeSeriesQuery:
                  timeSeriesFilter:
                    filter: |
                      resource.type="pubsub_topic"
                      resource.label.project_id="prd-bigdata-streaming-01"
                      metric.type="pubsub.googleapis.com/topic/send_message_operation_count"
                    aggregation:
                      alignmentPeriod: 60s
                      perSeriesAligner: ALIGN_RATE

      # Dataflow Job Metrics
      - width: 6
        height: 4
        widget:
          title: Dataflow Job Status
          scorecard:
            timeSeriesQuery:
              timeSeriesFilter:
                filter: |
                  resource.type="dataflow_job"
                  resource.label.project_id="prd-bigdata-streaming-01"
                  metric.type="dataflow.googleapis.com/job/is_failed"

      # BigQuery Slot Usage
      - width: 6
        height: 4
        widget:
          title: BigQuery Slot Utilization
          xyChart:
            dataSets:
              - timeSeriesQuery:
                  timeSeriesFilter:
                    filter: |
```

```
            resource.type="bigquery_project"
            resource.label.project_id="prd-bigdata-analytics-01"
            metric.type="bigquery.googleapis.com/slots/total_allocated"

    # Bigtable Performance
    - width: 6
      height: 4
      widget:
        title: Bigtable Read/Write Latency
        xyChart:
          dataSets:
            - timeSeriesQuery:
                timeSeriesFilter:
                  filter: |
                    resource.type="bigtable_table"
                    resource.label.instance="bt-prd-timeseries-uc1-01"
                    metric.type="bigtable.googleapis.com/server/latencies"
```

11.2 Logging & Alerting

yaml

```yaml
# Log Sink: Export to BigQuery for Analysis
logSink:
  name: sink-prd-audit-logs-to-bq
  destination: bigquery.googleapis.com/projects/shrd-security-logging-01/datasets/audit_logs
  filter: |
    protoPayload.serviceName=("bigquery.googleapis.com" OR "storage.googleapis.com" OR "dataflow.googleapis.com")
    protoPayload.methodName=~".*delete.*" OR protoPayload.methodName=~".*update.*"
  includeChildren: true

# Alert Policy: High Query Cost
alertPolicy:
  displayName: BigQuery - High Daily Query Cost
  conditions:
    - displayName: Daily query cost exceeds $1000
      conditionThreshold:
        filter: |
          resource.type="bigquery_project"
          resource.label.project_id="prd-bigdata-analytics-01"
          metric.type="bigquery.googleapis.com/query/scanned_bytes_billed"
        comparison: COMPARISON_GT
        thresholdValue: 8796093022208  # 1000 USD ≈ 8TB at $5/TB
        duration: 3600s
        aggregations:
          - alignmentPeriod: 3600s
            perSeriesAligner: ALIGN_SUM
  notificationChannels:
    - projects/shrd-monitoring-ops-01/notificationChannels/email-data-eng-leads
    - projects/shrd-monitoring-ops-01/notificationChannels/slack-data-alerts
  alertStrategy:
    autoClose: 604800s  # 7 days

# Alert Policy: Dataflow Job Failure
alertPolicy:
  displayName: Dataflow - Job Failure
  conditions:
    - displayName: Dataflow job has failed
      conditionThreshold:
        filter: |
          resource.type="dataflow_job"
          resource.label.project_id="prd-bigdata-streaming-01"
          metric.type="dataflow.googleapis.com/job/is_failed"
        comparison: COMPARISON_GT
        thresholdValue: 0
        duration: 60s
```

```
  notificationChannels:
    - projects/shrd-monitoring-ops-01/notificationChannels/pagerduty-oncall
```

11.3 SLIs & SLOs

yaml

```yaml
# Service Level Indicators (SLIs)
slis:
  - name: BigQuery Query Availability
    metric: |
      good_requests / total_requests
    goodEvents: |
      metric.type="bigquery.googleapis.com/query/count"
      metric.label.status="success"
    totalEvents: |
      metric.type="bigquery.googleapis.com/query/count"

  - name: Dataflow Pipeline Latency
    metric: |
      p95_latency < 60 seconds
    goodEvents: |
      metric.type="dataflow.googleapis.com/job/element_count"
      metric.label.processing_latency < 60000  # milliseconds

# Service Level Objectives (SLOs)
slos:
  - displayName: BigQuery Availability SLO
    serviceLevelIndicator:
      requestBased:
        goodTotalRatio:
          goodServiceFilter: |
            metric.type="bigquery.googleapis.com/query/count"
            metric.label.status="success"
          totalServiceFilter: |
            metric.type="bigquery.googleapis.com/query/count"
    goal: 0.999  # 99.9% availability
    rollingPeriodDays: 30

  - displayName: Real-time Pipeline Latency SLO
    serviceLevelIndicator:
      windowsBased:
        windowPeriod: 300s  # 5 minutes
        goodBadMetricFilter: |
          metric.type="dataflow.googleapis.com/job/element_count"
          metric.label.processing_latency < 60000
    goal: 0.95  # 95% of 5-minute windows meet latency target
    rollingPeriodDays: 7
```

12. Disaster Recovery & Business Continuity

12.1 Backup Strategy

yaml

```yaml
# BigQuery Dataset Snapshots
backupSchedule:
  - dataset: bq_prd_retail_analytics
    frequency: daily
    retention: 30d
    destination: gs://gcs-prd-backups-bq-multi-01/bq_prd_retail_analytics/

# Cloud Spanner Backup
spannerBackup:
  - instance: span-prd-orders-multiregional-01
    frequency: daily
    retention: 30d
    expireTime: +30d

# Bigtable Backup
bigtableBackup:
  - instance: bt-prd-timeseries-uc1-01
    table: user_events
    frequency: daily
    retention: 7d
```

12.2 Disaster Recovery Runbook

markdown

```
# Disaster Recovery Procedures

## Scenario 1: Regional Outage (us-central1)

### Detection
- Monitor alerts for region unavailability
- Check Cloud Status Dashboard

### Response
1. Activate failover to europe-west1
2. Update DNS records to point to EU load balancer
3. Scale up compute resources in europe-west1
4. Redirect streaming pipelines to EU Pub/Sub topics

### Recovery
- RTO: 4 hours
- RPO: 15 minutes

## Scenario 2: BigQuery Dataset Corruption

### Detection
- Data quality checks fail
- User reports anomalies

### Response
1. Identify last known good snapshot
2. Restore from backup
3. Re-run ETL pipelines for affected time period

### Recovery
- RTO: 2 hours
- RPO: 24 hours (last snapshot)
```

---

13. Key Takeaways

For Data Engineers

- Implement medallion architecture (Bronze/Silver/Gold) in Cloud Storage
- Use Dataflow for both streaming and batch processing
- Leverage Dataproc with ephemeral clusters for cost optimization
- Implement comprehensive data quality checks in BigQuery

For Data Scientists

- Use Vertex AI Workbench for collaborative development

- Store ML features in BigQuery for easy access
- Deploy models using Vertex AI endpoints with auto-scaling
- Track experiments with Vertex AI Experiments

For Platform Admins

- Enforce organization policies for security compliance
- Implement Shared VPC for network isolation
- Use VPC Service Controls for data exfiltration protection
- Set up comprehensive monitoring and alerting

For Data Analysts

- Query data efficiently using BigQuery partitioned and clustered tables
- Use Looker for self-service analytics
- Request materialized views for frequently accessed aggregations
- Respect row-level security policies

Appendix

A. Terraform Example

```hcl
```

```
# terraform/projects/prd-bigdata-analytics/main.tf

terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "~> 5.0"
    }
  }
  backend "gcs" {
    bucket = "gcs-terraform-state-globalmart"
    prefix = "projects/prd-bigdata-analytics"
  }
}

provider "google" {
  project = var.project_id
  region  = var.region
}

# BigQuery Dataset
resource "google_bigquery_dataset" "retail_analytics" {
  dataset_id    = "bq_prd_retail_analytics"
  friendly_name = "Retail Analytics"
  description   = "Production dataset for retail analytics"
  location      = "US"

  default_table_expiration_ms = null

  default_encryption_configuration {
    kms_key_name = google_kms_crypto_key.bigquery_key.id
  }

  labels = {
    environment         = "prd"
    cost_center         = "retail"
    data_classification = "internal"
  }
}

# Cloud Storage Bucket
resource "google_storage_bucket" "raw_landing" {
  name          = "gcs-prd-raw-landing-uc1-01"
  location      = "US-CENTRAL1"
  storage_class = "STANDARD"
```

```
  uniform_bucket_level_access = true

  encryption {
    default_kms_key_name = google_kms_crypto_key.storage_key.id
  }

  lifecycle_rule {
    condition {
      age = 7
    }
    action {
      type          = "SetStorageClass"
      storage_class = "NEARLINE"
    }
  }

  lifecycle_rule {
    condition {
      age = 365
    }
    action {
      type = "Delete"
    }
  }

  labels = {
    environment         = "prd"
    zone                = "landing"
    data_classification = "confidential"
  }
}

# IAM Binding
resource "google_project_iam_member" "data_engineers_bq_editor" {
  project = var.project_id
  role    = "roles/bigquery.dataEditor"
  member  = "group:gcp-data-engineers@globalmart.com"
}
```

B. Useful CLI Commands

bash

```
# BigQuery
bq query --use_legacy_sql=false 'SELECT COUNT(*) FROM `prd-bigdata-analytics-01.bq_prd_retail_analytics.sales_fact`'

# Cloud Storage
gsutil -m cp -r gs://gcs-prd-raw-landing-uc1-01/sales/2025-01-01/ ./local_backup/

# Dataflow
gcloud dataflow jobs list --region=us-central1 --filter="state=RUNNING"

# Dataproc
gcloud dataproc clusters create dpc-adhoc-01 \
  --region=us-central1 \
  --subnet=subnet-prd-compute-uc1-01 \
  --no-address \
  --service-account=dataproc-compute-prd@prd-bigdata-analytics-01.iam.gserviceaccount.com

# Bigtable
cbt -project=prd-bigdata-streaming-01 -instance=bt-prd-timeseries-uc1-01 read user_events

# Pub/Sub
gcloud pubsub topics publish topic-prd-pos-transactions-uc1-01 --message='{"test": "data"}'
```

Document Version: 1.0
Last Updated: January 2025
Owner: Data Platform Team
Contact: data-platform@globalmart.com