Back to Certification

Professional Data Engineer Study Notes

Comprehensive study guide covering all exam domains for Google Cloud Professional Data Engineer certification.

Download PDF

Exam Overview

The Google Cloud Professional Data Engineer exam validates your ability to design, build, and maintain data processing systems on Google Cloud Platform. The exam covers five key domains:

- **Design data processing systems** - Architecture patterns and solution design
- **Ingest and process data** - Batch and stream processing pipelines
- **Store data** - Data warehousing, lakes, and operational databases
- **Prepare and use data for analysis** - Data transformation and ML pipelines
- **Maintain and automate data workloads** - Operations, monitoring, and optimization

Domain 1: Design Data Processing Systems

Architecture Patterns

Lambda Architecture: Combines batch and stream processing for co... data processing. Batch layer processes historical data, speed layer ... updates, serving layer merges results.

Kappa Architecture: Stream-first approach where all data flows through stream processing. Simplifies Lambda by using single processing paradigm with event replay capability.

Data Lakehouse: Combines data lake storage flexibility with data warehouse performance. BigQuery + Cloud Storage provides cost-effective analytics with ACID transactions.

Key GCP Services

- **BigQuery:** Serverless data warehouse with SQL interface, automatic scaling, ML integration
- **Dataflow:** Managed Apache Beam for unified batch/stream processing with autoscaling
- **Pub/Sub:** Global message queue for event-driven architectures with at-least-once delivery
- **Cloud Composer:** Managed Apache Airflow for workflow orchestration
- **Dataproc:** Managed Spark/Hadoop for existing workloads with cluster autoscaling

Domain 2: Ingest and Process Data

Batch Processing

Best for: Large datasets, complex transformations, cost optimization

- Use Dataflow with batch pipeline for full dataset processing
- Leverage BigQuery batch loading for cost-effective ingestion
- Consider Dataproc for Spark/Hadoop workloads requiring ecosystem tools
- Implement partitioning and bucketing for efficient processing

Stream Processing

Best for: Real-time analytics, event processing, immediate insights

- Pub/Sub for buffering and decoupling producers from consumers
- Dataflow streaming for windowing, state management, and late data handling
- BigQuery streaming inserts for real-time dashboards (quota: 100K rows/sec/table)
- Bigtable for low-latency operational queries

Change Data Capture (CDC)

- **Datastream:** Serverless CDC for replicating from Oracle, MySQL, PostgreSQL to BigQuery, Cloud Storage, or Pub/Sub
- **Debezium + Dataflow:** Custom CDC pipelines for advanced transformations
- Consider log-based CDC for minimal source impact vs query-based for simplicity

Domain 3: Store Data

BigQuery Best Practices

- **Partitioning:** Use DATE/TIMESTAMP/INTEGER partitioning for time-series data, enables partition pruning reducing scan costs
- **Clustering:** Add up to 4 clustering columns for frequently filtered fields, automatically maintained
- **Materialized Views:** Pre-compute expensive queries for fast access, automatically maintained
- **Nested/Repeated Fields:** Use STRUCT/ARRAY for denormalized storage reducing joins
- **Cost Optimization:** Use on-demand for variable workload, flat-rate for predictable, partition filters reduce scans

Cloud Storage

- **Standard:** Frequently accessed data, highest availability
- **Nearline:** Data accessed less than 1/month, 30-day minimum storage

- **Coldline:** Data accessed less than 1/quarter, 90-day minimum
- **Archive:** Long-term retention, accessed less than 1/year, 365-day minimum
- Use lifecycle policies for automatic tiering based on age or access patterns

Bigtable

- **Use Cases:** Time-series, IoT, financial data, operational workloads requiring under 10ms latency
- **Row Key Design:** Critical for performance - distribute writes, enable range scans, avoid hotspotting
- **Column Families:** Group related data, set different retention policies per family
- **Replication:** Multi-cluster for HA, read replicas for geo-proximity

Cloud Spanner

- Globally distributed relational database with strong consistency
- Use for: ACID transactions across regions, SQL interface, horizontal scalability
- Schema design: Avoid hotspots, use interleaving for parent-child relationships

Domain 4: Prepare and Use Data for Analysis

dbt (Data Build Tool)

- **Models:** SQL-based transformations organized in DAG
- **Incremental Models:** Process only new/changed data for efficiency
- **Snapshots:** Implement SCD Type 2 for historical tracking
- **Tests:** Data quality validation (not null, unique, relationships, custom)
- **Documentation:** Auto-generated with lineage visualization

ML Pipelines

- **Vertex AI Pipelines:** Managed orchestration for ML workflows using Kubeflow Pipelines
- **Feature Store:** Centralized repository for feature management, versioning, and serving
- **BigQuery ML:** In-database ML for SQL-based model training and inference
- **Dataflow ML:** Preprocessing and inference in streaming/batch pipelines

Data Quality

- **Dataplex Data Quality:** Automated validation rules and monitoring
- **Great Expectations:** Python-based data validation framework
- Implement data contracts between producers and consumers
- Monitor data freshness, completeness, accuracy, consistency

Domain 5: Maintain and Automate Data Workloads

Cloud Composer (Airflow)

- **DAGs:** Define workflows as Python code with task dependencies
- **Operators:** Use BigQueryOperator, DataflowOperator, etc. for common tasks
- **Scheduling:** Cron-based scheduling with catchup and backfill capabilities
- **Monitoring:** Track task duration, failures, and SLA misses
- **Variables/Connections:** Centralized config and secret management

Cloud Workflows

- Simpler, cheaper alternative to Composer for sequential workflows
- YAML/JSON definition with built-in retries and error handling
- Use for: API orchestration, simple ETL, event-driven workflows

Monitoring and Observability

- **Cloud Logging:** Centralized log management with log-based metrics
- **Cloud Monitoring:** Metrics, dashboards, and alerting
- **SLIs/SLOs:** Define reliability targets (latency, availability, freshness)
- **Data Catalog:** Metadata management and data lineage

Cost Optimization

- Use BigQuery slots reservations for predictable workloads
- Implement partition pruning and clustering to reduce scans
- Set table expiration for temporary data
- Use appropriate storage classes (Standard, Nearline, Coldline, Archive)
- Right-size Dataflow workers and implement autoscaling
- Monitor with billing alerts and cost attribution via labels

Security and Compliance

- **IAM:** Role-based access control with principle of least privilege
- **VPC Service Controls:** Security perimeter to prevent data exfiltration
- **CMEK:** Customer-managed encryption keys for regulatory requirements
- **DLP API:** Automated PII detection and masking
- **Audit Logs:** Admin activity, data access, and system event logs
- **Policy Tags:** Column-level access control in BigQuery

Exam Tips

- Focus on **scenario-based questions** - understand when to use each service
- Know **cost optimization** strategies for each service
- Understand **scaling patterns** - when services autoscale vs manual intervention
- Master **BigQuery** deeply - it appears in most questions

- Practice **architecture diagrams** - visualize data flow through services
- Remember **security** considerations for each design decision
- Time management: 2 hours for 80 questions = ~90 seconds per question

Additional Resources

- Official Exam Guide
- BigQuery Documentation
- Dataflow Documentation
- Data Analytics Solutions