GCP Professional ML Engineer Certification Study Notes

Code: PMLE

Low-Code ML (17%)

Architecting Low-Code ML Solutions

BigQuery ML, AutoML, pre-built APIs

> **Domain Weight**
>
> Architecting Low-Code ML Solutions accounts for 17% of the exam.

BigQuery ML

- CREATE MODEL directly in BigQuery using SQL — no data export required
- Supported model types: LINEAR_REG, LOGISTIC_REG, KMEANS, MATRIX_FACTORIZATION, BOOSTED_TREE_CLASSIFIER, AUTOML_CLASSIFIER, DNN_CLASSIFIER
- ML.PREDICT for batch inference on BigQuery tables
- ML.EVALUATE to compute model metrics
- Ideal when data already lives in BigQuery and team has SQL expertise

Vertex AI AutoML

API	Use Case
Cloud Vision API	Image labeling, OCR, face detection, safe search
Cloud Speech-to-Text	Audio transcription (streaming + batch)
Document AI	Structured data extraction from documents/PDFs
Natural Language API	Sentiment, entities, classification, syntax
Translation API	Text translation across 100+ languages
Vertex AI Agent Builder	RAG-based search + conversational AI grounded in your data

Scaling (17%)

Scaling Prototypes Into ML Models

Custom training, hyperparameter tuning, distributed training

> **Domain Weight**
>
> Scaling Prototypes accounts for 17% of the exam.

Vertex AI Custom Training

- Use custom containers or pre-built containers (TF, PyTorch, scikit-learn,

Distributed Training Strategies

Strategy	Framework	When to Use
MirroredStrategy	TensorFlow	Multi-GPU single machine — synchronous data parallelism
MultiWorkerMirroredStrategy	TensorFlow	Multi-node multi-GPU — synchronous distributed
ParameterServerStrategy	TensorFlow	Very large models with separate parameter servers
DDP (DistributedDataParallel)	PyTorch	Multi-node data parallelism with PyTorch
Horovod	TF / PyTorch	Ring-allreduce for distributed training at scale

Hyperparameter Tuning

- Vertex AI HyperparameterTuningJob — automated HPT built on Vizier
- Algorithms: GRID_SEARCH, RANDOM_SEARCH, BAYESIAN_OPTIMIZATION (default)
- Report metric via hypertune library or stdout (cloudml-hypertune)
- Set maxTrialCount, parallelTrialCount, maxFailedTrialCount

Serving (17%)

> Serving and Scaling Models accounts for 17% of the exam.

Online vs Batch Prediction

Mode	Latency	Use Case
Online (Vertex AI Endpoint)	Low (ms)	Real-time inference with REST/gRPC
Batch Prediction Job	High (minutes-hours)	Large-volume offline scoring
Streaming Prediction	Low	Dataflow + model for real-time stream processing

Model Optimization Techniques

- Quantization: reduce weight precision (FP32 → INT8) — reduces size ~4×, speeds up inference
- Pruning: remove unimportant weights to create sparse models
- Distillation: train smaller student model to mimic larger teacher model
- TensorRT: NVIDIA GPU optimization for production inference
- ONNX: interoperable model format for cross-framework deployment

Traffic Splitting & A/B Testing

- Vertex AI Endpoints support traffic splitting across multiple DeployedModels
- e.g., 80% to model v1, 20% to model v2 for gradual rollout
- Canary deployments: route small % to new model before full rollout

Vertex AI Pipelines, Kubeflow, CI/CD for ML

> **Domain Weight**
>
> Automating ML Pipelines accounts for 16% of the exam.

Vertex AI Pipelines

- Serverless pipeline orchestration based on Kubeflow Pipelines (KFP) SDK
- Define pipelines as Python functions decorated with @component and @pipeline
- Each component runs in a container — inputs/outputs are typed artifacts
- Google Cloud Pipeline Components: prebuilt components for AutoML, training, evaluation, deployment
- Pipeline artifacts stored in Cloud Storage; metadata in Vertex ML Metadata

CI/CD for ML

- Trigger retraining via Cloud Scheduler, Pub/Sub events, or data triggers
- Cloud Build / Cloud Deploy for containerizing and deploying model updates
- Artifact Registry: store Docker images and ML model artifacts
- Use Cloud Source Repositories or GitHub for version-controlled pipeline code

Feature Store

- Vertex AI Feature Store: centralized feature repository for online + offline serving
- Entity types map to feature groups; features are versioned and shareable
- Online serving: low-latency point-in-time feature lookup
- Offline serving: historical feature batch retrieval for training

> **Domain Weight**
>
> Monitoring ML Solutions accounts for 16% of the exam.

Types of Drift

Drift Type	Description	Detection Method
Input (feature) drift	Input distribution shifts from training baseline	Compare feature distributions: PSI, KL divergence, chi-squared
Prediction drift	Model output distribution changes over time	Compare prediction distributions over rolling windows
Concept drift	Relationship between inputs and target changes	Compare model performance (accuracy, AUC) over time with ground truth
Label skew	Training labels differ from serving-time labels	Compare training vs served label distributions

Vertex AI Model Monitoring

- Deploy ModelMonitor on a Vertex AI Endpoint — logs prediction requests/responses
- Set baseline from training dataset stored in BigQuery or Cloud Storage
- Configure skew and drift thresholds per feature
- Alerts via Cloud Monitoring on threshold violations
- Enable explanation monitoring to track feature attribution drift

- Continuous training: Vertex AI Pipelines triggered by Cloud Scheduler or Pub/Sub

Collaboration (17%)

Collaborating Within and Across Teams

Data validation, model cards, responsible AI

> **Domain Weight**
>
> Collaborating accounts for 17% of the exam.

Data Validation

- TensorFlow Data Validation (TFDV): compute statistics, infer schema, detect anomalies
- validate_statistics() to compare serving data against training schema
- Detect feature distribution anomalies and missing values in production data
- Vertex AI Dataset: managed dataset with data lineage and versioning

Responsible AI & Fairness

- Vertex Explainable AI: SHAP (Shapley values), XRAI (image), IG (integrated gradients)
- What-If Tool: inspect model predictions, test counterfactuals, analyze slices
- Model cards: document model purpose, performance, limitations, intended use, fairness
- Google's Responsible AI practices: fairness, interpretability, privacy, security

- Model Registry: centralized model versioning, aliases (champion/challenger), deployment tracking