

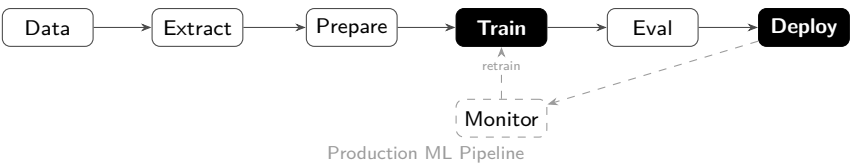


Production ML Systems

GCP Professional ML Engineer | Architecting, Adaptability, Performance & Hybrid

ML System Architecture

ML model code = ~5% of production codebase. The other 95% is data collection, feature extraction, configuration, monitoring, serving infrastructure, and process management.

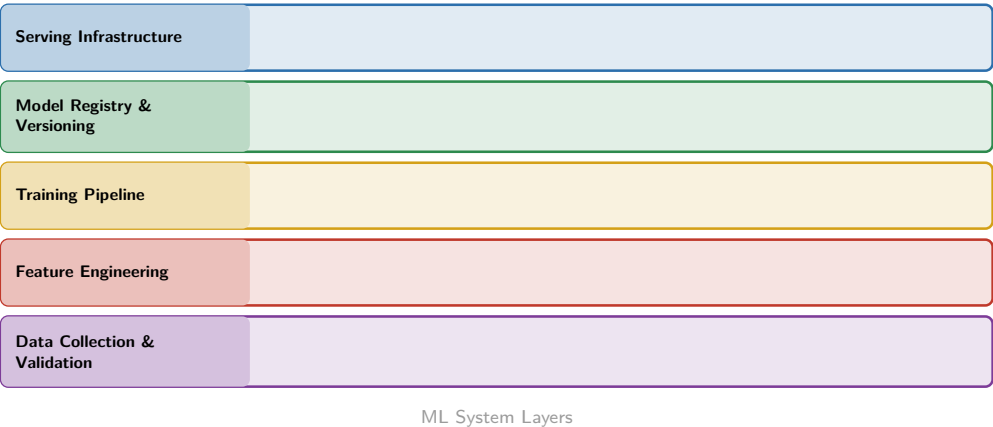


Key System Components

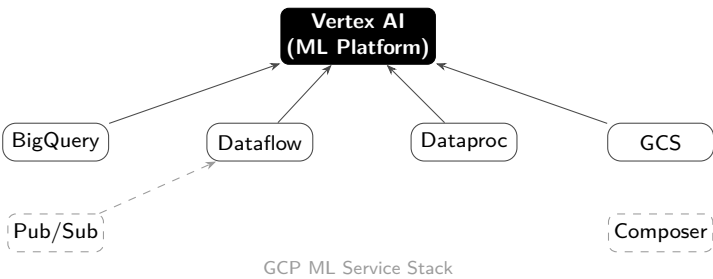
- Data collection & ingestion (Pub/Sub, Dataflow, Cloud Storage)
- Feature store for consistent train/serve features (Vertex AI Feature Store)
- Model registry for versioning (Vertex AI Model Registry)
- Prediction service with autoscaling (Vertex AI Endpoints)
- Monitoring & alerting (Vertex AI Model Monitoring, Cloud Monitoring)

🔗 ML System Design

Production ML = only ~5% ML code. The rest is data collection, feature extraction, serving infrastructure, monitoring, and configuration.



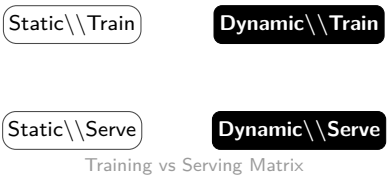
GCP Managed Services for ML



Vertex AI	Unified ML platform: AutoML, custom training, pipelines, prediction, feature store, model registry, experiments
Dataproc	Managed Spark/Hadoop clusters for large-scale batch processing
Dataflow	Managed Apache Beam for unified streaming + batch ETL pipelines
BigQuery	Serverless data warehouse; also BigQuery ML for SQL-based model training
Cloud Storage	Object store for datasets, model artifacts, unstructured data
Pub/Sub	Real-time event/message ingestion for streaming ML pipelines
Cloud Composer	Managed Apache Airflow for workflow orchestration

Static vs Dynamic Training	
✓ Pros	✗ Cons
+ Simpler infrastructure	- Concept drift risk
+ Predictable costs	- Stale predictions
+ Easier debugging	- Manual retraining
+ Good for stable data	- Poor for streaming data

Training & Serving Paradigms

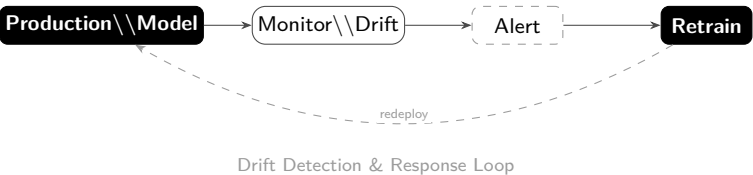


Paradigm	When to Use	GCP Tool
Static training	Stable data, infrequent updates	Vertex AI custom job
Dynamic training	Data drift, continuous new data	Vertex AI Pipelines + scheduler
Static serving	Low cardinality, high peakedness	Precompute -> Big-Query/Memorystore
Dynamic serving	High cardinality, variable queries	Vertex AI Endpoints (online)
Batch serving	Non-real-time, large volume	Vertex AI Batch Prediction

Best Practices

- Start with static training + dynamic serving as baseline
- Use peakedness vs cardinality matrix to choose serving strategy
- Batch prediction for overnight reports; online for user-facing

Designing Adaptable ML Systems



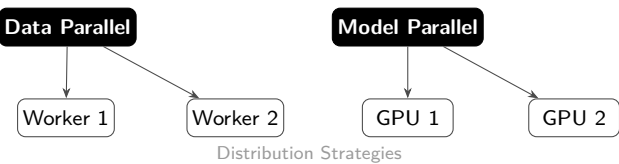
Concept Drift & Data Drift

Data drift	Input feature distribution changes (e.g. new user demographics)
Concept drift	Relationship between features and target changes (e.g. user behavior shift)
Prediction drift	Model output distribution changes without ground truth
Label drift	Target variable distribution shifts over time

Monitoring & Response

- Vertex AI Model Monitoring: auto-detect skew and drift
- Set feature-level thresholds (Jensen-Shannon divergence, L-infinity)
- Continuous evaluation: compare predictions to delayed ground truth
- Automatic retraining triggers via Cloud Functions + Vertex Pipelines
- A/B testing and canary deployments for safe model rollouts
- Feature Store prevents train/serve skew by centralizing feature logic

Designing High-Performance ML



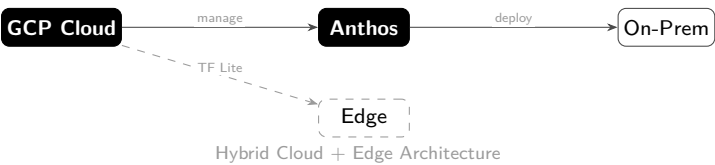
Training Performance

Data parallelism	Split data across workers, each has full model copy. Use for large datasets. MirroredStrategy (single machine), MultiWorkerMirrored (multi-node).
Model parallelism	Split model layers across GPUs. Use for models too large for single GPU memory.
GPUs/TPUs	NVIDIA T4 (inference), V100/A100 (training). TPU v3/v4 optimized for TensorFlow. Use Vertex AI for managed accelerators.
tf.data optimization	prefetch(AUTOTUNE), cache(), interleave(), map(num_parallel_calls=AUTOTUNE). Use TFRecord format.

Serving Performance

Autoscaling	Vertex AI Endpoints: min/max replicas, target CPU utilization, traffic-based scaling
Model optimization	Quantization (FP32->INT8), pruning (remove weights), distillation (teacher->student)
Edge deployment	TF Lite for mobile, Edge TPU for IoT, TF.js for browser inference
Caching	Memorystore/Redis for repeated predictions on same inputs

Hybrid ML Systems

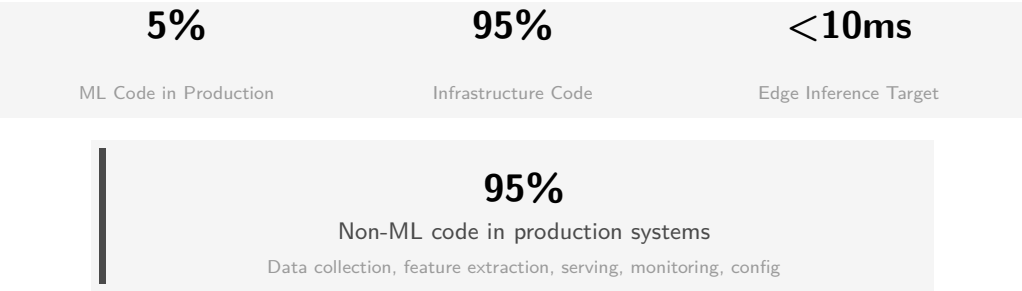


Hybrid = combining cloud and on-premise/edge for ML workloads. Use when data residency, latency, or connectivity constraints exist.

Anthos	Run containers on-prem and multi-cloud with consistent management. GKE everywhere.
Kubeflow	ML toolkit for Kubernetes; portable ML pipelines across environments
TFX	TensorFlow Extended: production ML pipelines, works on Kubeflow or Vertex
Edge TPU	Google's ASIC for running TF Lite models on edge devices (Coral)

When to Use Hybrid

- Data residency: regulations prevent data leaving specific regions
- Low latency: inference must happen on-device (<10ms)
- Intermittent connectivity: edge devices without reliable network
- Cost: reduce cloud egress by processing data locally



Official resources

⚠️ **Check the current exam guide**
Vertex AI documentation now redirects to **Gemini Enterprise Agent Platform** (checked 2026-10-05) and Google Cloud is updating its exams to the products announced at Next '26: these sheets keep the Vertex AI names; read the current exam guide before you study.

1	EXAM GUIDE FREE	Professional Machine Learning Engineer exam guide services.google.com/fh/files/misc/profession...engineer_exam_guide_english_new.pdf domains and topics	1 h	Σ 1 h
2	EXAM GUIDE FREE	Professional Machine Learning Engineer certification page cloud.google.com/learn/certification/machine-learning-engineer format, price, sample questions link, renewal	0.5 h	Σ 1.5 h
3	VIDEO FREE	Professional Machine Learning Engineer learning path skills.google/paths/17 Google Skills (ex Cloud Skills Boost)	40 h	Σ 41.5 h
4	CHEATSHEET FREE	gcloud CLI cheat sheet cloud.google.com/sdk/docs/cheatsheet the commands on one page	1 h	Σ 42.5 h
5	DOCS FREE	Gemini Enterprise Agent Platform (ex Vertex AI) documentation docs.cloud.google.com/gemini-enterprise-agent-platform training, pipelines, endpoints	4 h	Σ 46.5 h
6	DOCS FREE	Agent Platform Pipelines (ex Vertex AI Pipelines) docs.cloud.google.com/gemini-enterprise-agen...ine-learning/pipelines/introduction	2 h	Σ 48.5 h
7	WHITEPAPER FREE	MLOps: continuous delivery and automation pipelines cloud.google.com/architecture/mlops-continuo...ation-pipelines-in-machine-learning	1.5 h	Σ 50 h
Total 50 h · 8.3 weeks at 6 h/week · 50 h free				