# Google Cloud Certified Professional Machine Learning Engineer

*This version of the exam guide will go live on October 1, 2024

## Certification Exam Guide

A Professional Machine Learning Engineer builds, evaluates, productionizes, and optimizes AI solutions by using Google Cloud capabilities and knowledge of conventional ML approaches. The ML Engineer handles large, complex datasets and creates repeatable, reusable code. The ML Engineer designs and operationalizes generative AI solutions based on foundational models. The ML Engineer considers responsible AI practices, and collaborates closely with other job roles to ensure the long-term success of AI-based applications. The ML Engineer has strong programming skills and experience with data platforms and distributed data processing tools. The ML Engineer is proficient in the areas of model architecture, data and ML pipeline creation, generative AI, and metrics interpretation. The ML Engineer is familiar with foundational concepts of MLOps, application development, infrastructure management, data engineering, and data governance. The ML Engineer enables teams across the organization to use AI solutions. By training, retraining, deploying, scheduling, monitoring, and improving models, the ML Engineer designs and creates scalable, performant solutions.

### Section 1: Architecting low-code AI solutions (13% of the exam)

**1.1 Developing ML models by using BigQuery ML. Considerations include:**

- Building the appropriate BigQuery ML model (e.g., linear and binary classification, regression, time-series, matrix factorization, boosted trees, autoencoders) based on the business problem

- Feature engineering or selection by using BigQuery ML
- Generating predictions by using BigQuery ML

**1.2 Building AI solutions by using ML APIs or foundational models. Considerations include:**

- Building applications by using ML APIs from Model Garden
- Building applications by using industry-specific APIs (e.g., Document AI API, Retail API)
- Implementing retrieval augmented generation (RAG) applications by using Vertex AI Agent Builder

**1.3 Training models by using AutoML. Considerations include:**

- Preparing data for AutoML (e.g., feature selection, data labeling, Tabular Workflows on AutoML)
- Using available data (e.g., tabular, text, speech, images, videos) to train custom models
- Using AutoML for tabular data
- Creating forecasting models by using AutoML
- Configuring and debugging trained models

## Section 2: Collaborating within and across teams to manage data and models (14% of the exam)

**2.1 Exploring and preprocessing organization-wide data (e.g., Cloud Storage, BigQuery, Spanner, Cloud SQL, Apache Spark, Apache Hadoop). Considerations include:**

- Organizing different types of data (e.g., tabular, text, speech, images, videos) for efficient training
- Managing datasets in Vertex AI
- Data preprocessing (e.g., Dataflow, TensorFlow Extended [TFX], BigQuery)
- Creating and consolidating features in Vertex AI Feature Store

- Privacy implications of data usage and/or collection (e.g., handling sensitive data such as personally identifiable information [PII] and protected health information [PHI])
- Ingesting different data sources (e.g., text documents) into Vertex AI for inference

**2.2 Model prototyping by using Jupyter notebooks. Considerations include:**

- Choosing the appropriate Jupyter backend on Google Cloud (e.g., Vertex AI Workbench, Colab Enterprise, notebooks on Dataproc)
- Applying security best practices in Vertex AI Workbench
- Using Spark kernels
- Integrating code source repositories
- Developing models in Vertex AI Workbench by using common frameworks (e.g., TensorFlow, PyTorch, sklearn, Spark, JAX)
- Leveraging a variety of foundational and open-source models in Model Garden

**2.3 Tracking and running ML experiments. Considerations include:**

- Choosing the appropriate Google Cloud environment for development and experimentation (e.g., Vertex AI Experiments, Kubeflow Pipelines, Vertex AI TensorBoard with TensorFlow and PyTorch) given the framework
- Evaluating generative AI solutions

## Section 3: Scaling prototypes into ML models (18% of the exam)

**3.1 Building models. Considerations include:**

- Choosing ML framework and model architecture
- Modeling techniques given interpretability requirements

**3.2 Training models. Considerations include:**

- Organizing training data (e.g., tabular, text, speech, images, videos) on Google Cloud (e.g., Cloud Storage, BigQuery)

- Ingestion of various file types (e.g., CSV, JSON, images, Hadoop, databases) into training
- Model training by using different SDKs (e.g., Vertex AI custom training, Kubeflow on Google Kubernetes Engine, AutoML, tabular workflows)
- Using distributed training to organize reliable pipelines
- Hyperparameter tuning
- Troubleshooting ML model training failures
- Fine-tuning foundational models (e.g., Vertex AI, Model Garden)

**3.3 Choosing appropriate hardware for training. Considerations include:**

- Evaluation of compute and accelerator options (e.g., CPU, GPU, TPU, edge devices)
- Distributed training with TPUs and GPUs (e.g., Reduction Server on Vertex AI, Horovod)

## Section 4: Serving and scaling models (20% of the exam)

**4.1 Serving models. Considerations include:**

- Batch and online inference (e.g., Vertex AI, Dataflow, BigQuery ML, Dataproc)
- Using different frameworks (e.g., PyTorch, XGBoost) to serve models
- Organizing models in Model Registry
- A/B testing different versions of a model

**4.2 Scaling online model serving. Considerations include:**

- Managing and serving features by using Vertex AI Feature Store
- Deploying models to public and private endpoints
- Choosing appropriate hardware (e.g., CPU, GPU, TPU, edge)
- Scaling the serving backend based on the throughput (e.g., Vertex AI Prediction, containerized serving)

- Tuning ML models for training and serving in production (e.g., simplification techniques, optimizing the ML solution for increased performance, latency, memory, throughput)

## Section 5: Automating and orchestrating ML pipelines (22% of the exam)

**5.1 Developing end-to-end ML pipelines. Considerations include:**

- Validating data and models
- Ensuring consistent data pre-processing between training and serving
- Hosting third-party pipelines on Google Cloud (e.g., MLflow)
- Identifying components, parameters, triggers, and compute needs (e.g., Cloud Build, Cloud Run)
- Orchestration frameworks (e.g., Kubeflow Pipelines, Vertex AI Pipelines, Cloud Composer)
- Hybrid or multicloud strategies
- Designing systems with TFX components or Kubeflow DSL (e.g., Dataflow)

**5.2 Automating model retraining. Considerations include:**

- Determining an appropriate retraining policy
- Deploying models in continuous integration and continuous delivery (CI/CD) pipelines (e.g., Cloud Build, Jenkins)

**5.3 Tracking and auditing metadata. Considerations include:**

- Tracking and comparing model artifacts and versions (e.g., Vertex AI Experiments, Vertex ML Metadata)
- Hooking into model and dataset versioning
- Model and data lineage

## Section 6: Monitoring AI solutions (13% of the exam)

**6.1 Identifying risks to AI solutions. Considerations include:**

- Building secure AI systems by protecting against unintentional exploitation of data or models (e.g., hacking)
- Aligning with Google's Responsible AI practices (e.g., monitoring for bias)
- Assessing AI solution readiness (e.g., fairness, bias)
- Model explainability on Vertex AI (e.g., Vertex AI Prediction)

**6.2 Monitoring, testing, and troubleshooting AI solutions. Considerations include:**

- Establishing continuous evaluation metrics (e.g., Vertex AI Model Monitoring, Explainable AI)
- Monitoring for training-serving skew
- Monitoring for feature attribution drift
- Monitoring model performance against baselines, simpler models, and across the time dimension
- Monitoring for common training and serving errors