Google Cloud

Professional Machine Learning Engineer

Certification exam guide

A Professional Machine Learning (ML) Engineer builds, evaluates, productionizes, and optimizes AI solutions 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 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, MLOps, and metrics interpretation. The ML Engineer is familiar with the foundational concepts of prompt and context engineering, 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, tuning, monitoring, and improving traditional and generative AI (gen AI) models, the ML Engineer designs and creates scalable, performant solutions.

*Note: The exam does not directly assess coding skills. If you have a minimum proficiency in Python and SQL, you should be able to interpret any questions with code snippets.

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

1.1 Developing ML models using BigQuery ML or AutoML on Gemini Enterprise Agent Platform. Considerations include:

- Building models in BigQuery ML or Agent Platform AutoML (e.g., classification, regression, forecasting, and clustering) based on the business problem
- Performing feature engineering or selection using BigQuery ML
- Generating predictions using BigQuery ML
- Training models using Agent Platform AutoML
- Fine-tuning Gemini models using BigQuery

1. 2 Building AI solutions using Google Cloud AI APIs or foundational models. Considerations include:

- Evaluating and selecting the appropriate model for a given task from Gemini Enterprise Agent Platform Model Garden
- Building applications using industry-specific APIs (e.g., Document AI API, Vision API, and Translate API)
- Building solutions and tuning models for specific use cases (e.g., Gemini, Imagen, Veo, and models as a service in Model Garden)
- Optimizing Gemini-based applications for cost, latency, and availability

Section 2: Collaborating within and across teams to manage data and models (~16% of the exam)

2.1 Exploring and preprocessing data for ML. Considerations include:

- Organizing and exploring different data types (e.g., tabular, text, and images) for efficient experimenting, training, and serving
- Choosing the right tool for data preprocessing based on scale and complexity (e.g., BigQuery [SQL], Dataflow, Apache Spark, and in-memory Python frameworks)
- Creating and consolidating features in Gemini Enterprise Agent Platform Feature Store
- Ensuring data privacy and handling sensitive information (e.g., personally identifiable information [PII])

2.2 Model prototyping using notebooks (e.g., Gemini Enterprise Agent Platform Workbench and Colab Enterprise). Considerations include:

- Applying collaboration and security best practices when setting up and running notebook environments
- Developing models in Agent Platform Workbench or Colab Enterprise notebooks using common frameworks (e.g., PyTorch, sklearn, and JAX)
- Using a variety of foundational and open-source models in Model Garden to create model prototypes in notebook environments

2.3 Tracking and running ML experiments. Considerations include:

- Choosing the appropriate Google Cloud environment for development and experimentation (e.g., Experiments on Gemini Enterprise Agent Platform, Gemini Enterprise Agent Platform Pipelines, and Kubeflow Pipelines) given the framework

- Evaluating predictive and gen AI solutions (e.g., model evaluation metrics and LLM-as-a-judge)
- Tracking and comparing model artifacts, versions, and lineage (e.g., Experiments on Agent Platform and Gemini Enterprise Agent Platform ML Metadata)

Section 3: Scaling prototypes into ML models (~21% of the exam)

3.1 Building models given the task considering cost, complexity, latency, and scalability. Considerations include:

- Choosing the model type (e.g., ARIMA, DNN, and LLM)
- Choosing the product (e.g., Agent Platform AutoML, BigQuery ML, and Agent Platform Pipelines)
- Choosing the deployment strategy
- Modeling techniques given interpretability requirements

3.2 Training models. Considerations include:

- Organizing training data (e.g., tabular, text, speech, images, and videos) on Google Cloud (e.g., Cloud Storage and BigQuery)
- Ingesting structured and unstructured data from various sources into training pipelines
- Model training using different software development kits (SDKs) (e.g., Agent Platform custom training, Kubeflow on Google Kubernetes Engine [GKE], Agent Platform AutoML, and Tabular Workflows) and organizing training on Google Cloud
- Troubleshooting ML model training failures
- Hyperparameter tuning
- Fine-tuning foundational models from Agent Platform and Model Garden and when tuning should be considered

3.3 Choosing appropriate hardware for training. Considerations include:

- Evaluation of compute and accelerator options (e.g., CPU, GPU, and TPU)
- Understanding the options for distributed training on GPUs and TPUs using data and model parallelism strategies

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

4.1 Serving models. Considerations include:

- Deploying models for batch and online inference using appropriate services (e.g., Agent Platform, Model Garden, Cloud Run, and GKE)
- Packaging and serving models from different frameworks (e.g., PyTorch and XGBoost) using prebuilt and custom containers
- Organizing and versioning models in Gemini Enterprise Agent Platform Model Registry
- Implementing model rollout strategies (e.g., A/B testing and canary deployments) to compare model versions
- Developing solutions for inference preprocessing and postprocessing

4.2 Scaling online model serving. Considerations include:

- Managing and serving features using Agent Platform Feature Store
- Deploying models to public and private endpoints
- Choosing appropriate hardware (e.g., CPU, GPU, TPU, and edge)
- Scaling the serving backend based on the throughput (e.g., Gemini Enterprise Agent Platform Inference and containerized serving)
- Tuning ML models for training and serving in production

Section 5: Automating and orchestrating ML pipelines (~18% of the exam)

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

- Validating data and models
- Building and orchestrating pipelines using managed or unmanaged services and from templates or custom solutions (e.g., Agent Platform Pipelines, Managed Service for Apache Airflow, and Ray on Gemini Enterprise Agent Platform)
- Ensuring consistent data preprocessing between training and serving

5.2 Automating model retraining. Considerations include:

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

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 and leaks of data or models (e.g., data exfiltration, malicious prompting, and sharing sensitive data with LLMs) using the appropriate security tool (e.g., Regex, safety filters, and Model Armor)
- Aligning with responsible AI practices (e.g., monitoring for bias)
- Model explainability on Agent Platform (e.g., Agent Platform Inference)

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

- Configuring and using Model Monitoring on Gemini Enterprise Agent Platform to establish continuous evaluation metrics for production models
- Monitoring for common issues (e.g., training-serving skew, data drift, concept drift, and feature attribution drift)
- Monitoring, testing, and evaluating gen AI solutions