Google Skills

Reviewing the best practices for architecting an inference stack on GKE

This module follows Don McCasland's video, ‘Deploying Scalable and Reliable AI Inference on Google Cloud’, to provide a deeper look into the GKE Inference reference architecture.

Where the video introduces core concepts, this document details the architectural components and strategic choices you can implement.

System-level design principles

Here we have multiple resources that can help you deploy a high-performing inference stack, but it's worth starting with the bigger picture. An inference system is only valuable if it is reliable (application uptime), elastic (automatically scalable up and down) and observable (has clearly defined, trackable metrics). Here's what to look out for:

Reliability

Deploying across multiple Google Cloud regions is a strong strategy for fault tolerance and low latency. Model files should be stored in multi-region Cloud Storage buckets to ensure fast startup times in any region.

Learn more: Read about exposing inference workloads with global load balancing and using Storage Transfer Service for replicating data across regions.

Comprehensive Observability

You can't fix what you can't see. A production-grade inference stack requires deep visibility into both the infrastructure and the model's performance. The reference architecture is designed to expose critical metrics for monitoring, exposing critical metrics for systems like Prometheus.

Learn more: Dive into how to manage and monitor your AI-optimized GKE clusters with out-of-the-box dashboards for GPU and TPU utilization, memory, and latency.

Autoscaling

The "cattle, not pets" principle is central to modern, scalable infrastructure. You should treat your servers as "cattle" that can be replaced automatically, rather than unique systems ("pets") that require manual fixing.

This architecture is defined using Infrastructure as Code (Terraform), with the option for GKE Autopilot to automatically manage and scale your infrastructure. In that scenario, if a node fails, GKE replaces it. If load increases, GKE provides more nodes to run your workloads.

Learn more: Get familiar with how GKE cluster autoscaling works to manage your infrastructure dynamically.

Key components of the GKE inference architecture

Model support

The GKE inference reference architecture is not limited to specific models. It is a flexible blueprint designed to serve a wide variety of open models. Documentation and examples illustrate how to deploy popular model families such as:

- **Google Gemma:** Google's open, cost-effective model family. Recommended for most inference use cases, delivering state-of-the-art results for a wide range of model sizes.
- **Meta Llama:** Meta open model family.
- **GPT-OSS:** OpenAI open model family.
- **Qwen:** Alibaba Cloud open model family.

Choosing and obtaining accelerators

A successful strategy involves choosing the right accelerator and planning for capacity.

1. Choosing the right processor
Read up on the available GPU platforms and TPU machine types on Google Cloud, but when selecting a VM for your workload, this options table is a good place to start. It outlines how each machine fits different workloads, with recommendations based on common optimization objectives.

2. Create a capacity strategy
Given the high demand, securing accelerators requires a deliberate strategy. Plan for a **baseline** of predictable usage and a **burst** strategy for peaks. The exact value of baseline accelerator capacity to reserve depends on your use cases and requirements. For example, you could:

- Reserve 100% of the capacity you need upfront, to give you complete assurance using Future Reservations, or
- Reserve a fraction of your capacity, instead relying on on-demand capacity and other consumption models for non time-sensitive batch inference tasks.
- For example, Dynamic Workload Scheduler Flex-start mode allows you to obtain capacity in-bulk with a single request, provisioning resources for batch inference when they become available (often off-peak). And if your jobs are fault-tolerant, you can use Spot VMs to save from 60-91%. You could also use GKE's custom ComputeClasses ('CCCs') to use multiple consumption models together by creating rules that tell the autoscaler which consumption model to choose based on set conditions, giving you complete control over how and when you use accelerator capacity.

Getting your storage in order

- **Efficient model loading:** To avoid downloading multi-gigabyte models every time a new inference server starts, this architecture uses a **Model Downloader**. This is a Kubernetes Job that pre-downloads models from repositories like Hugging Face and stores them in a Cloud Storage bucket. Inference workloads then mount this bucket using the Cloud Storage FUSE CSI driver, which allows Kubernetes pods to access model files as if they were on a local filesystem, with optimized caching for fast startup times.
- **High-throughput data access:** Traditional storage can become a bottleneck for workloads that require frequent access to massive datasets (beyond just the model weights). Managed Lustre provides a high-performance parallel file system designed for these scenarios, delivering the extreme I/O throughput needed to keep your accelerators fully utilized.

For more information on how the GKE Inference reference architecture configures the Cloud Storage FUSE CSI driver, see LLM Inference Optimization on GKE: Achieving faster Pod Startup with Google Cloud Storage

Advanced traffic routing

A standard load balancer isn't sufficient for complex AI workloads. The architecture uses the GKE Inference Gateway, an extension of the GKE Gateway that is “model-aware.” This is configured through Kubernetes custom resources where you can define routing rules, assign priorities to different models, and set policies for how traffic should be distributed based on real-time metrics.

Unlike a traditional load balancer that just sees network traffic, the Inference Gateway understands AI-specific patterns. It can perform intelligent routing based on:

- The model being requested
- The priority of the request
- The current load and queue depth of the model servers

This prevents a single, long-running request from blocking others and allows you to prioritize critical workloads. For global traffic management, GKE Gateways can be used as a first load-balancing layer to distribute traffic across regions, with GKE Inference Gateways then managing the AI-aware routing within each region.

Learn more: Explore how to customize the Inference Gateway configuration and read the comprehensive guide on how to Choose a load balancing strategy for AI/ML model inference on GKE.