
Google Events + Experiences
Proprietary & Confidential
Proprietary & Confidential
Google I/O Extended

Google I/O Extended

Simplifying LLM Serving Pipelines with GKE Inference Gateway

Efficient, Scalable, and Secure LLM Inference with GKE

Google I/O Extended

Ananda Dwi Rahmawati

- Cloud & DevOps Engineer, Singapore
- Google Developer Expert Cloud - Modern Architecture
- Master of Computer Science - University of Texas at Austin
- https://linktr.ee/misskecupbung

I/O Extended
Jogjakarta

"By leveraging GKE Inference Gateway, organizations can **automate** traffic management, scale LLM inference workloads **efficiently**, and enforce **robust security**—simplifying the operational complexity of production AI pipelines."

Introduction - The Challenge of AI Inference

AI models are increasingly **complex**, demanding **significant** computational resources.

- Inference Latency
- Operational Costs
- Scalability
- Complexity
- Resource Underutilization
- Model Lifecycle Management
- Dependency Management

The Challenges

- Organizations deploy Large Language Models (LLMs) for applications like conversational AI, semantic search, and content generation. Production LLM serving introduces challenges:
    - Orchestrating **complex** deployment pipelines for model hosting and inference
    - **Dynamically scaling** compute resources (CPU, GPU, TPU) for bursty traffic
    - Enforcing **secure** authentication, authorization, and network isolation
    - **Optimizing** resource allocation to control costs and maintain performance
    - Integrating monitoring, logging, and tracing for **observability**
    - A managed, cloud-native solution is needed to abstract these complexities, enabling efficient, secure, scalable LLM inference in production.

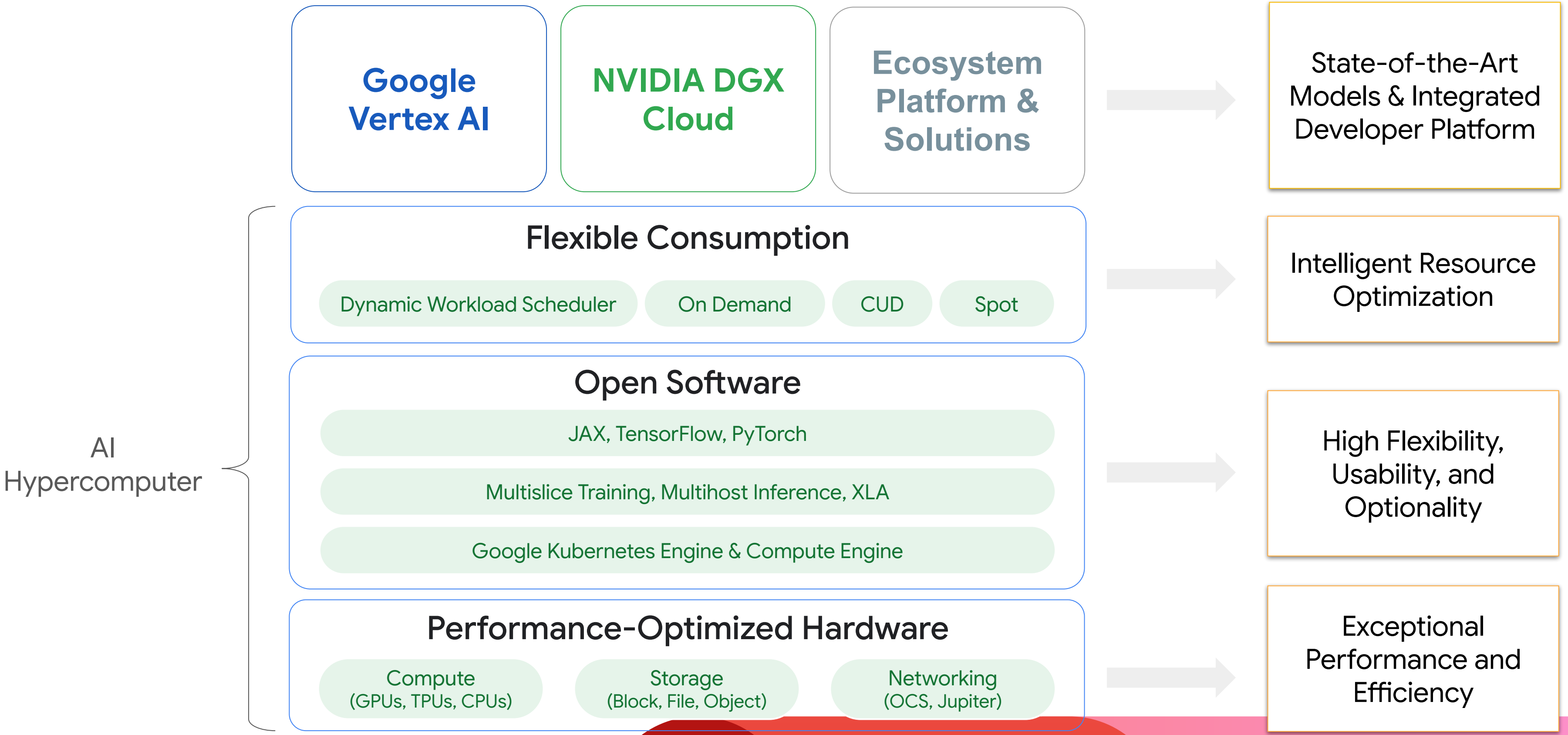
Real-Time Customer Support Chatbot

A company deploys an LLM-powered chatbot to handle customer inquiries on their website. GKE Inference Gateway enables seamless scaling and secure access, ensuring fast, reliable responses even during peak traffic.

Key Takeaways

- Understand how GKE Inference Gateway **reduces complexity** in deploying and managing LLM serving pipelines.
- Learn best practices for **scalable**, **secure**, and **cost-effective** LLM inference on Kubernetes.
- Discover how to leverage managed **traffic routing**, **autoscaling**, and **observability** features for production workloads.
- Gain practical insights from real-world deployment examples and actionable steps for your own LLM projects.

AI Hypercomputer Architecture



Google Cloud AI Hypercomputer

Flexible Consumption

- Dynamic Workload Scheduler
- On Demand
- CUD
- Spot

Open Software

- JAX, TensorFlow, PyTorch
- Multislice Training, Multihost Inference, XLA
- Google Kubernetes Engine & Compute Engine

Performance-Optimized Hardware

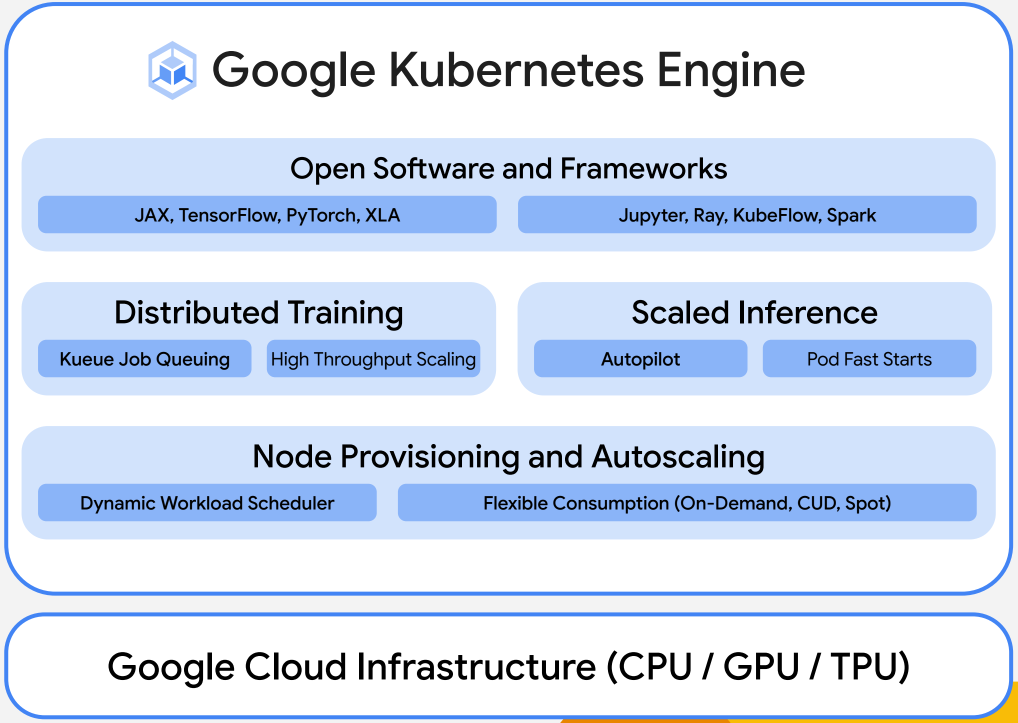
- Compute (GPUs, TPUs, CPUs)
- Storage (Block, File, Object)
- Networking (OCS, Jupiter)

Optimizing system-level co-design streamlines the entire AI lifecycle—from training to tuning and serving

Google Kubernetes Engine

cloud native infrastructure for AI training and inference

- **Limitless Scale:** Deploy AI at industry-leading scale, supporting thousands of TPUs and nodes.
- **Cost-Efficient Performance:** Maximize price-performance with smart GPU/TPU use, job queuing, and fast provisioning.
- **Effortless Ops:** Focus on models, not infra, with GKE Autopilot's managed, optimized Kubernetes.
- **Enterprise Reliability:** Trust AI workloads to the leading Kubernetes contributor's cloud-native infra.



Why Google Kubernetes Engine for AI

1 Portability & Customizability

Choice of frameworks and ecosystem tools that are portable

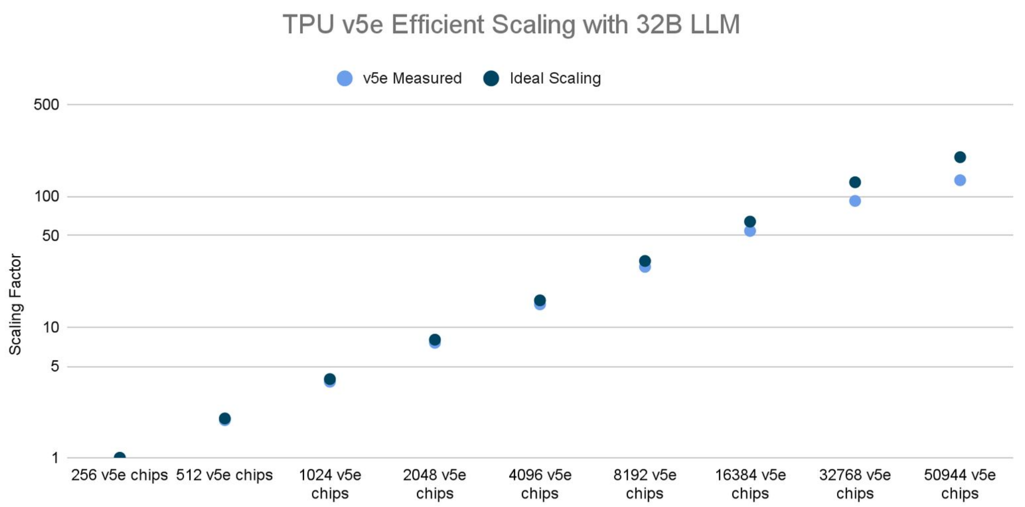
2 Performance & Scalability

Scale the platform for supercomputer scale training and inference

3 Cost-Efficiency

Increase utilization of valuable resources while reducing operational overhead

World's largest distributed training job on GKE with TPU Multislice Training



Scaled to

50,000+

TPU v5e chips

*Google Internal data for TPU v5e As of November, 2023: All numbers normalized per chip. seq-len=2048 for 32 billion parameter decoder only language model implemented using MaxText. *2*

What is GKE Inference Gateway?

Managed Inference Serving Layer

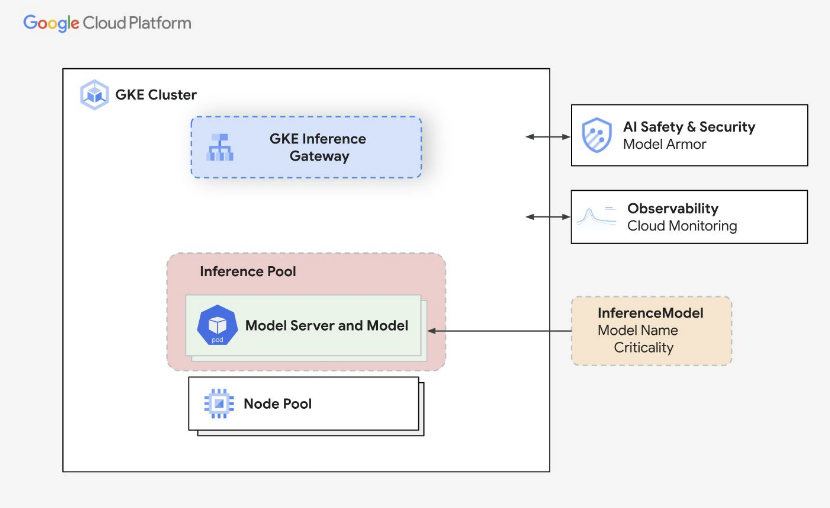
GKE Inference Gateway provides a fully managed, scalable layer for serving machine learning model inferences on Google Kubernetes Engine.

Simplified Traffic Routing

It automatically routes inference requests to the appropriate model endpoints, supporting versioning and canary deployments with minimal configuration.

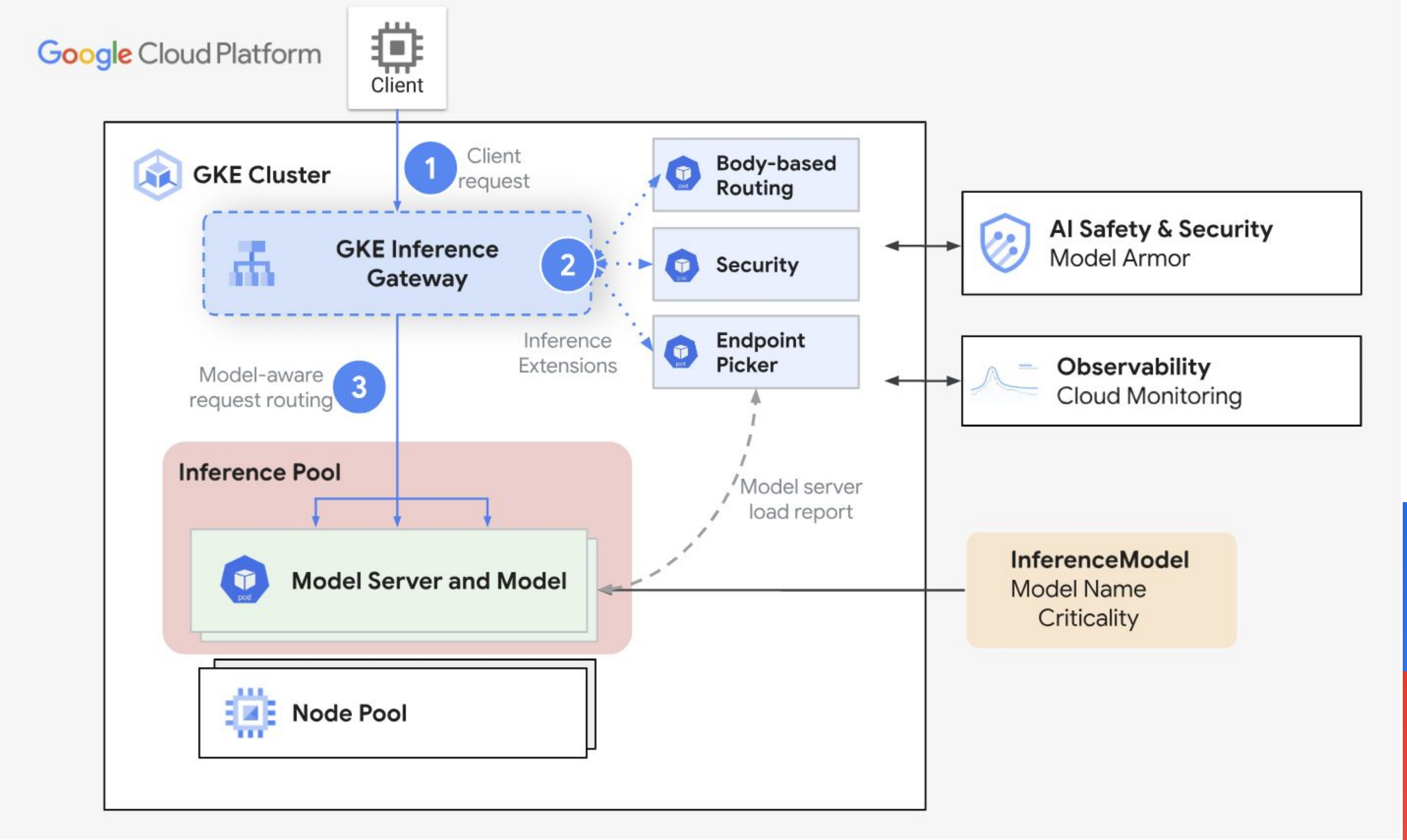
Integrated Security and Observability

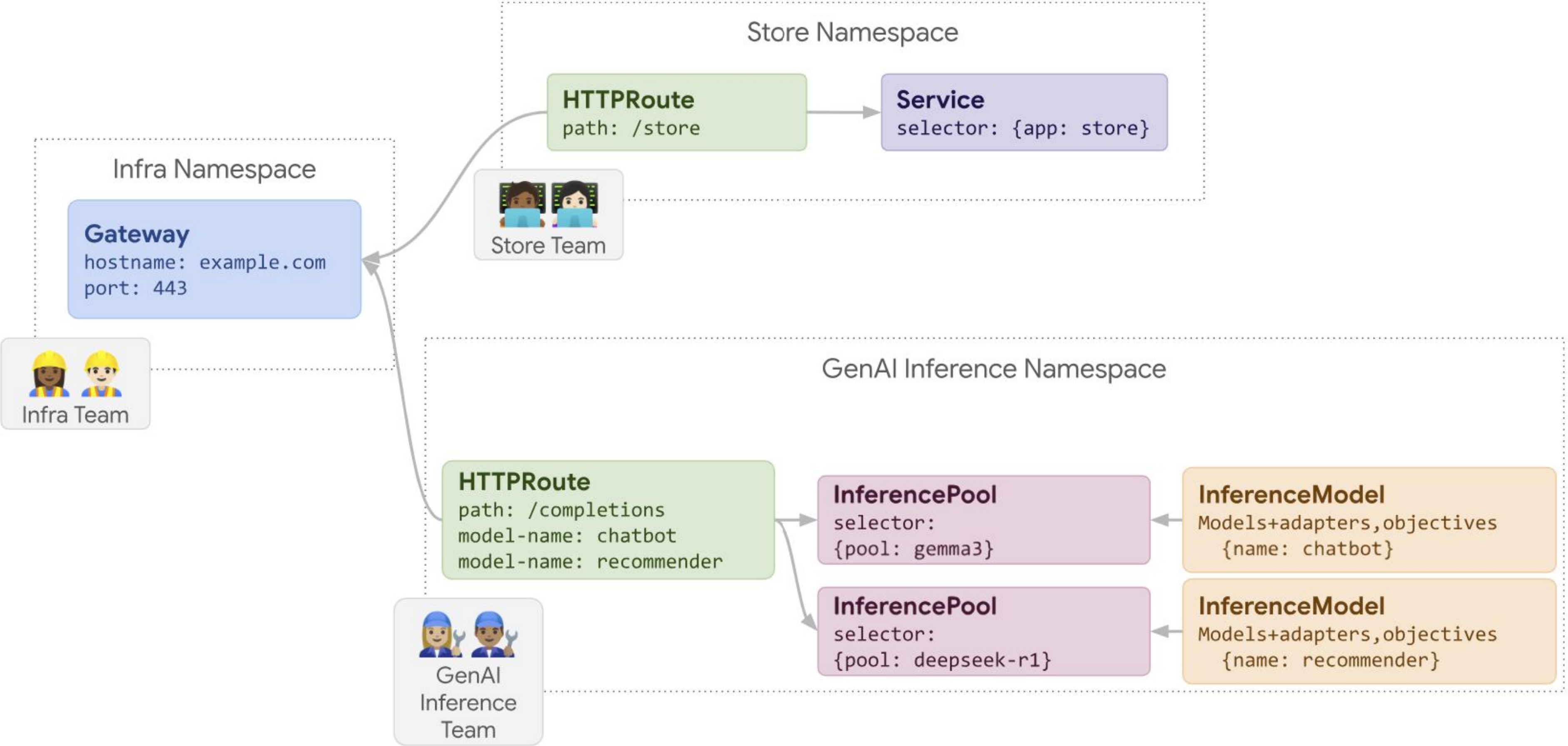
The gateway offers built-in authentication, authorization, and monitoring features, enabling secure and observable model serving out of the box.



How the Request Works?

- **Client Request**
  - Client sends a request in **OpenAI API format** to the GKE Inference Gateway.
- **Body-Based Routing Extension**
  - Extracts **model ID** from the request body.
  - Routes request via **Gateway API HTTPRoute** using this identifier.
  - Enables flexible, content-aware routing.
- **Security Extension**
  - Applies **Model Armor** or third-party security policies.
  - Performs **content filtering, threat detection, sanitization, and logging**.
  - Secures both request and response paths.
- **Endpoint Picker Extension**
  - Monitors **KV-cache**, queue lengths, and **LoRA adapter status**.
  - Selects **optimal model replica** based on real-time metrics.
  - Maximizes throughput and reduces inference latency.
- **Final Routing**
  - Request is forwarded to the **chosen model replica** in the **InferencePool**.
  - Model processes and returns the inference result.


Google Cloud Platform
Client
GKE Cluster
1
Client request
GKE Inference Gateway
2
Body-based Routing
Security
Endpoint Picker
Inference Extensions
Model-aware request routing
3
AI Safety & Security
Model Armor
Observability
Cloud Monitoring
Inference Pool
Model server load report
Model Server and Model
InferenceModel
Model Name
Criticality
Node Pool


Store Namespace
HTTPRoute
path: /store
Service
selector: {app: store}
Store Team
Infra Namespace
Gateway
hostname: example.com
port: 443
Infra Team
GenAI Inference Namespace
HTTPRoute
path: /completions
model-name: chatbot
model-name: recommender
InferencePool
selector:
{pool: gemma3}
InferencePool
selector:
{pool: deepseek-r1}
InferenceModel
Models+adapters,objectives
{name: chatbot}
InferenceModel
Models+adapters,objectives
{name: recommender}
GenAI Inference Team

To create an **InferencePool** using Helm, perform the following steps:

```
helm install vllm-llama3-8b-instruct \
  --set inferencePool.modelServers.matchLabels.app=vllm-llama3-8b-instruct \
  --set provider.name=gke \
  --version v0.3.0 \
  oci://registry.k8s.io/gateway-api-inference-extension/charts/inferencepool
```

Save the following sample manifest as inferencemodel.yaml:

```yaml
apiVersion:
inference.networking.x-k8s.io/v1alpha2
kind: InferenceModel
metadata:
  name: inferencemodel-sample
spec:
  modelName: MODEL_NAME
  criticality: VALUE
  poolRef:
    name: INFERENCE_POOL_NAME
```

Apply the sample manifest to your cluster:

```
kubectl apply -f inferencemodel.yaml
```

Create an InferenceModel that serves the food-review LoRA model on the vllm-llama3-8b-instruct **InferencePool** with **Standard** criticality, while the base model is served with a Critical priority level.

```yaml
apiVersion: inference.networking.x-k8s.io/v1alpha2
kind: InferenceModel
metadata:
  name: food-review
spec:
  modelName: food-review
  criticality: Standard
  poolRef:
    name: vllm-llama3-8b-instruct
  targetModels:
  - name: food-review
    weight: 100

---
apiVersion: inference.networking.x-k8s.io/v1alpha2
kind: InferenceModel
metadata:
  name: llama3-base-model
spec:
  modelName: meta-llama/Llama-3.1-8B-Instruct
  criticality: Critical
  poolRef:
    name: vllm-llama3-8b-instruct
```

Save the following sample manifest as gateway.yaml:

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: GATEWAY_NAME
spec:
  gatewayClassName: GATEWAY_CLASS
  listeners:
    - protocol: HTTP
      port: 80
      name: http
```

Apply the sample manifest to your cluster:

```
kubectl apply -f gateway.yaml
```

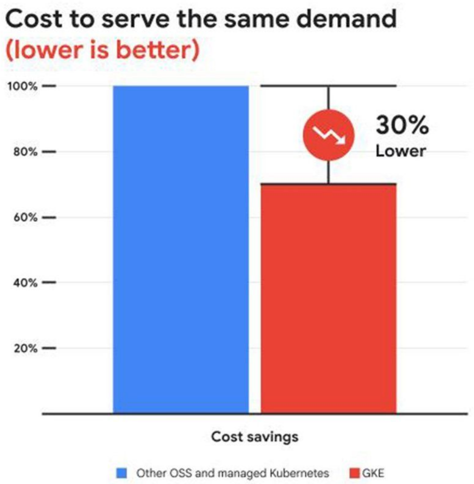
To create an HTTPRoute, save the following sample manifest as httproute.yaml:

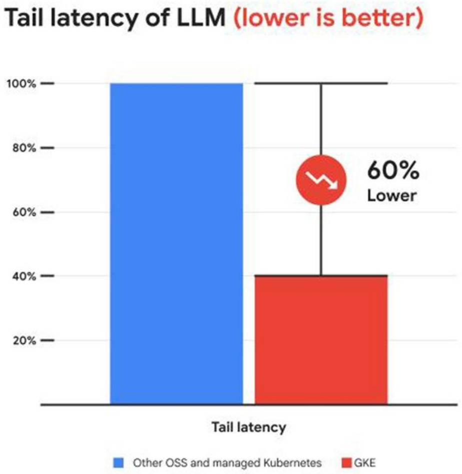
```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: HTTPROUTE_NAME
spec:
  parentRefs:
  - name: GATEWAY_NAME
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: PATH_PREFIX
    backendRefs:
    - name: INFERENCE_POOL_NAME
      group: inference.networking.x-k8s.io
      kind: InferencePool
```

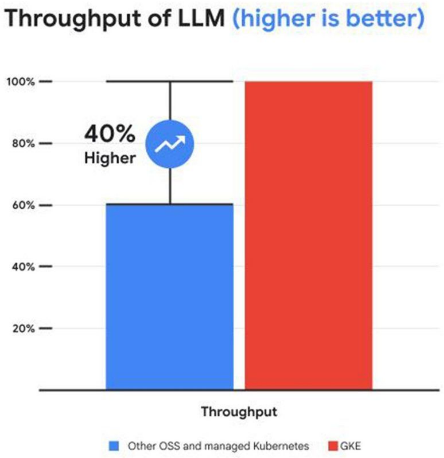
Apply the sample manifest to your cluster:

```
kubectl apply -f httproute.yaml
```

GKE Inference Performance







Solution mapping to customer journey

	01 **Evaluation**	02 **Onboarding**	03 **Production ramp** (day 2 operations)
Typical needs	Learn about all the accelerator options and choose one Procure capacity for inference POC (reservations, on-demand, spot, or DWS)	Establish performance baseline for common open model (Llama, Mixtral, ...) and model server (vLLM) via basic benchmarking Expect parity with GPUs on-prem or in other clouds	Load balance traffic price-performance at scale with real traffic Balance availability and cost efficiency Monitor performance metrics
GKE Gen AI features	**GKE Inference Quickstart** simplifies accelerator qualification from months to days	**GKE Inference Quickstart** suggests scaling thresholds vLLM/TPU stack for price-perf	**GKE Inference Gateway** for LLM aware routing Built-in vLLM observability metrics

I/O Extended
Jogjakarta

QnA

(Answers Not Guaranteed)

Extended
Jogjakarta

Thank You