
KubeCon
CloudNativeCon
North America 2024

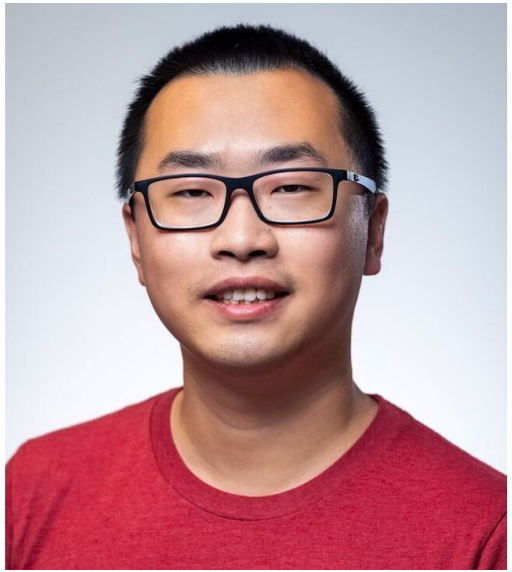
KubeCon | CloudNativeCon

North America 2024

Unlocking Potential of Large Models in Production

Yuan Tang, Principal Software Engineer at Red Hat
Adam Tetelman, Principal Product Architect at NVIDIA

Meet Your Speakers



Yuan Tang
Principal Software Engineer
Red Hat

Adam Tetelman
Principal Product Architect
NVIDIA

Session Agenda

History and Problem Statement:

Serving Large Language Models in Production

Solutions to serving LLMs:

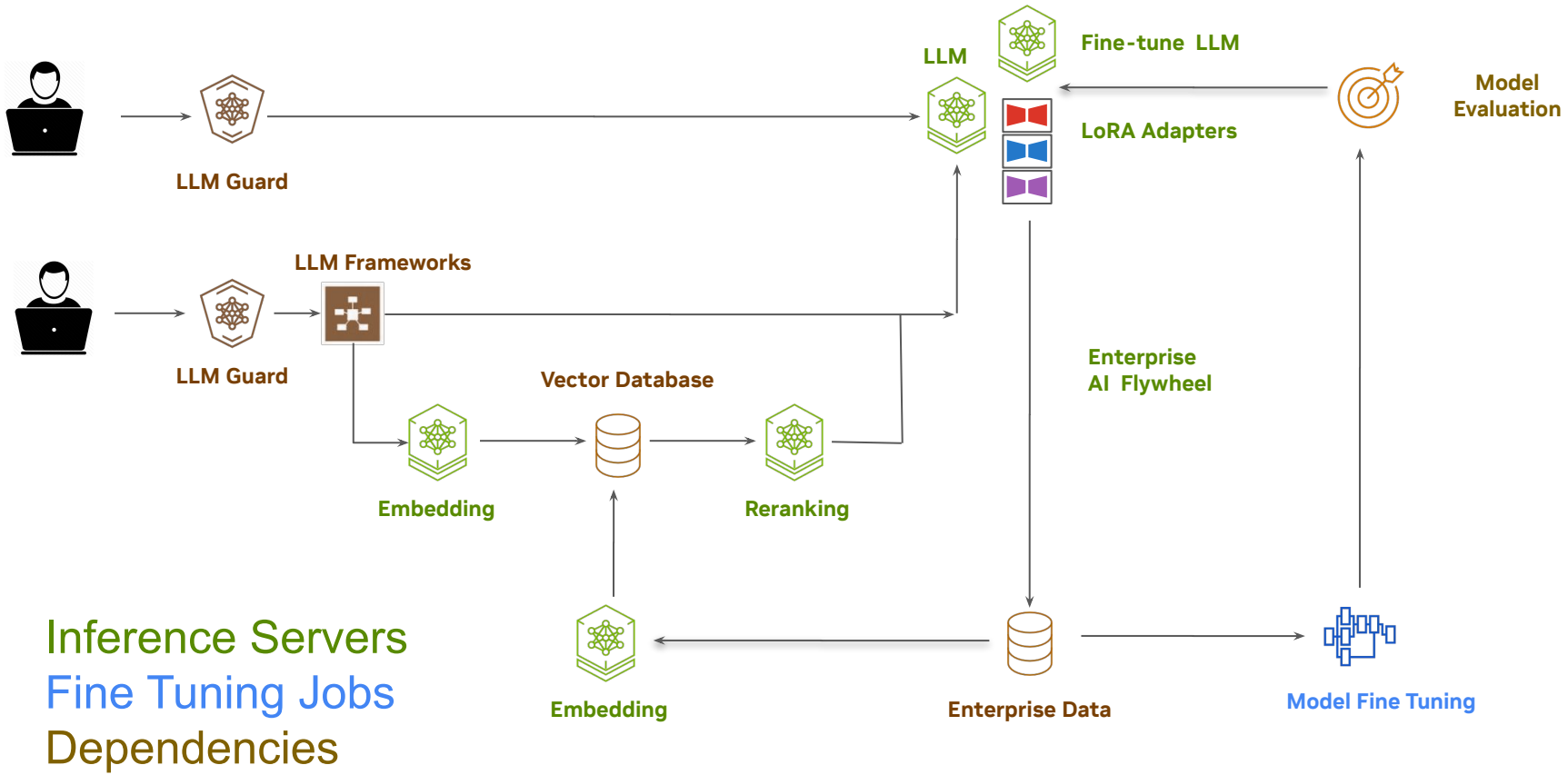
KServe Case Study

Future Problems to Solve:

Feature Gaps, Roadmaps, AI Advancements, and Community Engagement

Serving Large Language Models in Production

Inference, RAG and Fine Tuning Pipelines



Summary of Agentic RAG System

Inference

Inference Server: A server to wrap models in a callable API and provide additional operational services.

Inference Platform: A platform used to manage deployed inference servers and provide advanced operational capabilities.

RAG System

Embedding Model: Model used to convert text or objects into searchable embeddings

Reranking Model: Model used to improve the search results from a Vector DB.

Vector DB: Specialized database used to store and search embedding.

Data Connectors: APIs and integrations used to pull live data from web services or external data sources.

LLM Agent System

Agent / Chain Server: Application with memory that has access to RAG systems, LLM models, and tooling. Applies reasoning and planning to manage agentic conversations.

Large Language Model:: Foundational model that takes an input of user-generated text and system-provided context and generates a text output.

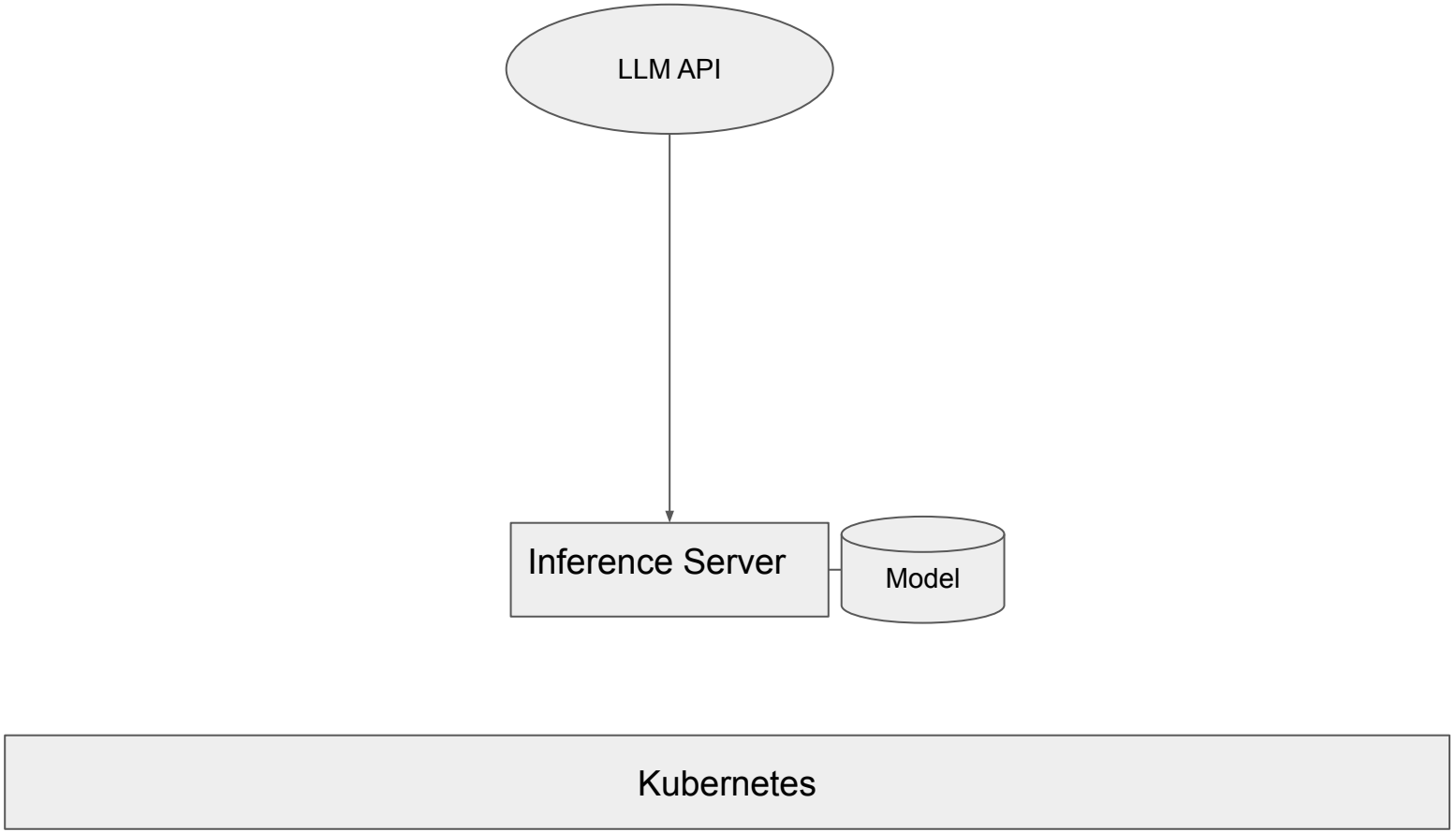
PEFT/LoRA: A form of fine-tuning that adds or overrides a small section of a foundational LLM to add specialized knowledge.

Tools: External programmatic capabilities that can be made available to an agent.

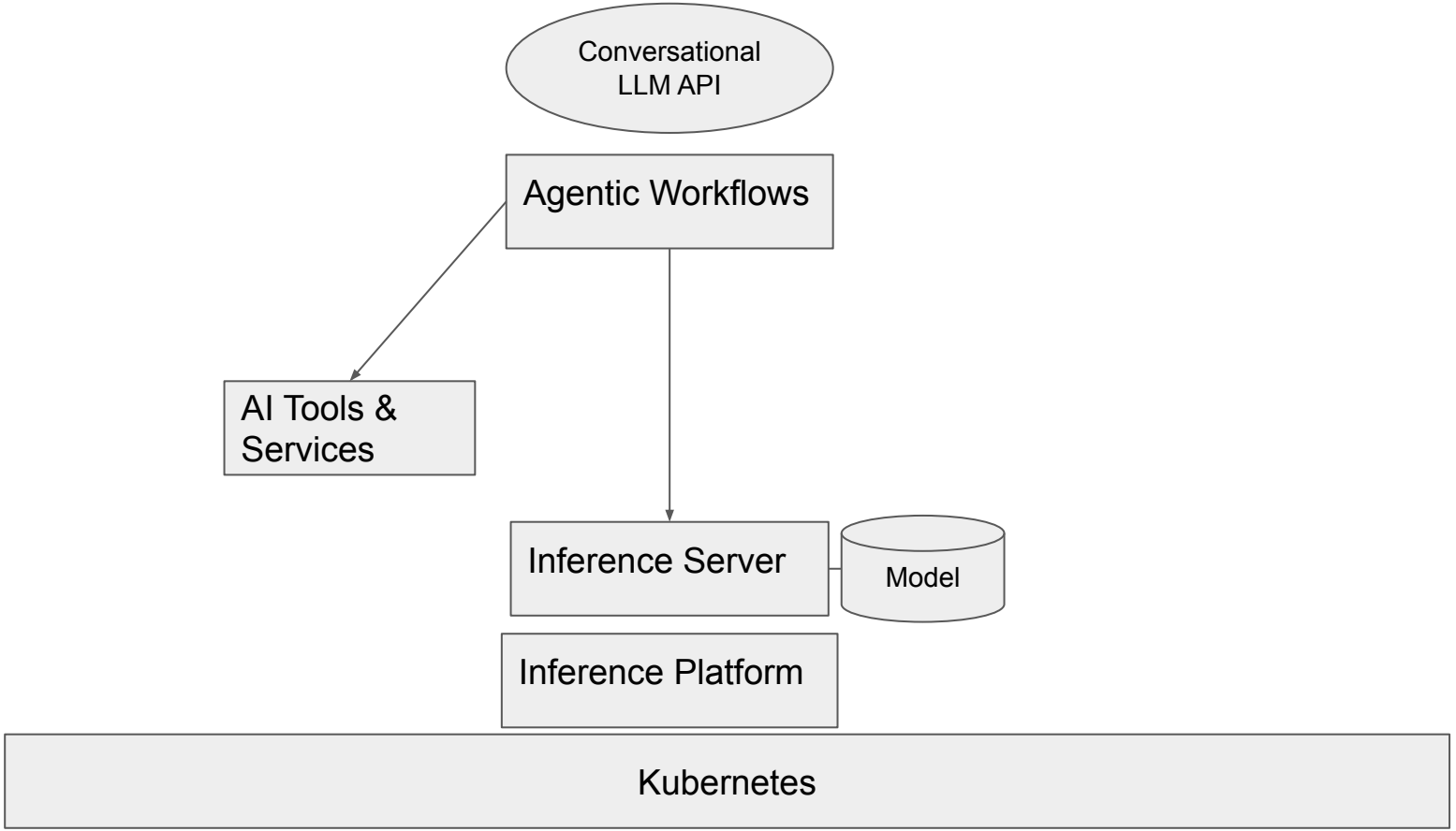
Guardrails: A lightweight system with models for detecting inappropriate content. Can be run on all user input and all LLM output.

The Building Blocks of a GenAI Workflow

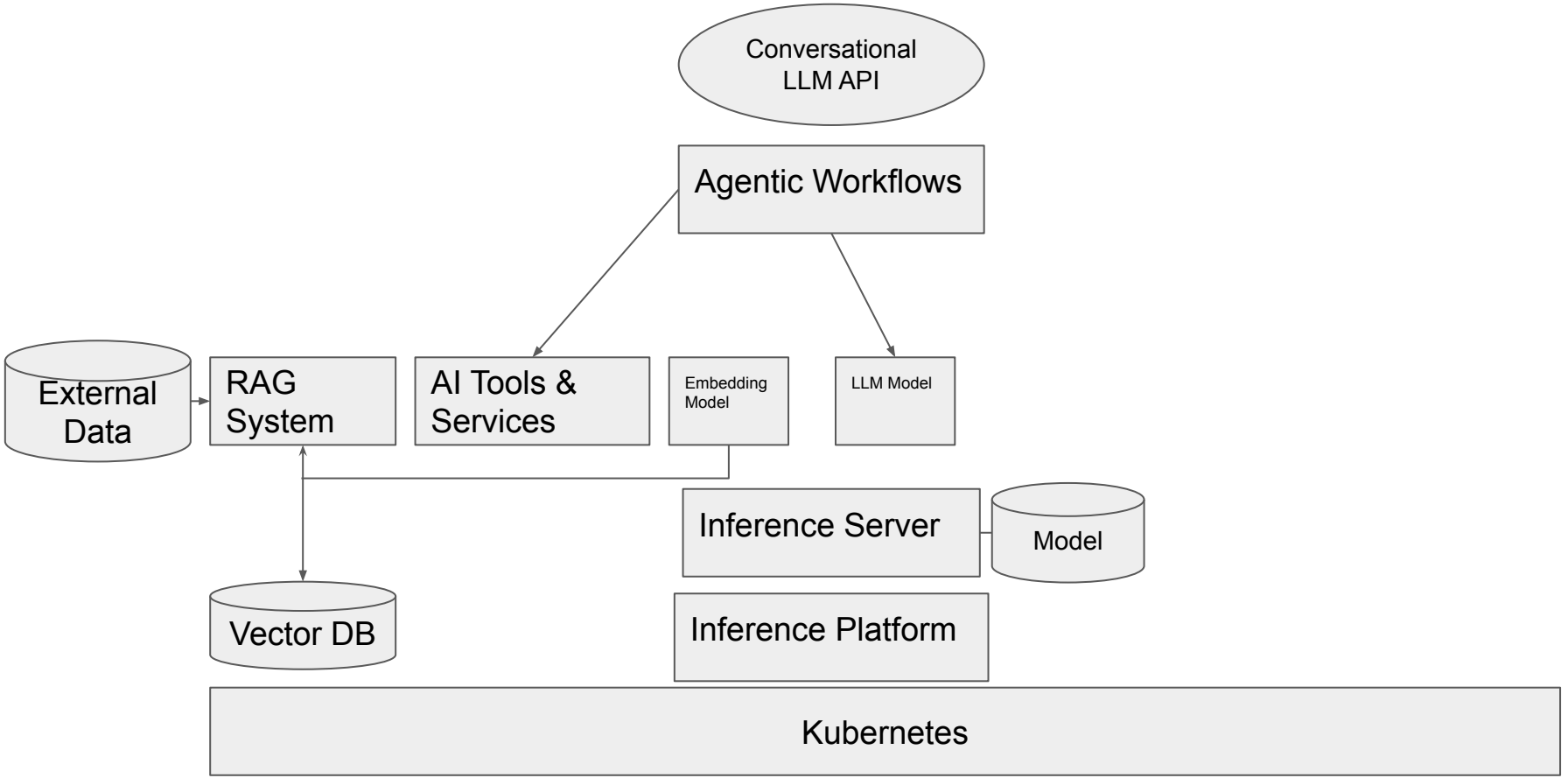
A basic developer-focused LLM MVP



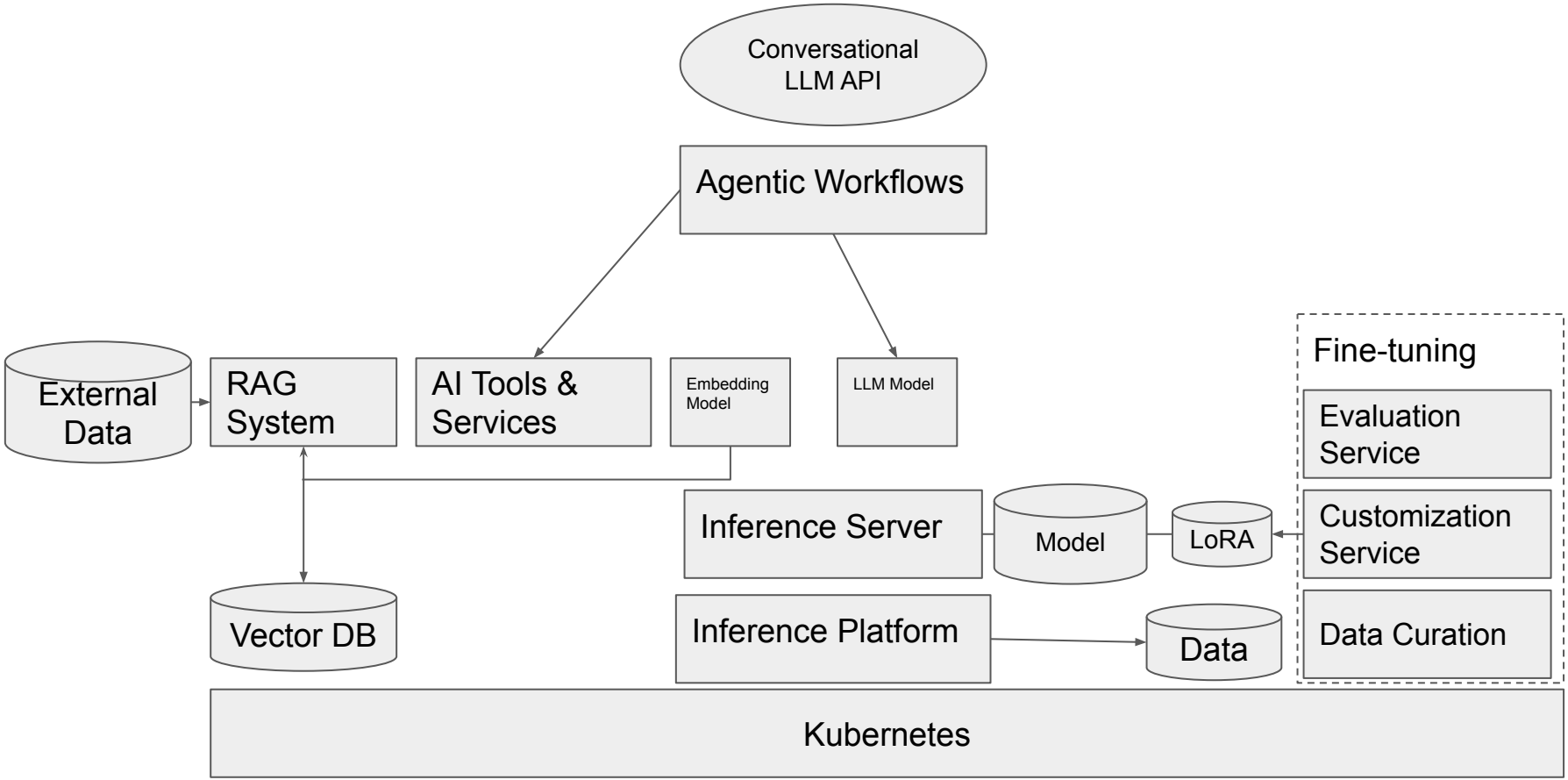
Adding workflow and agentic capabilities for real-world scenarios



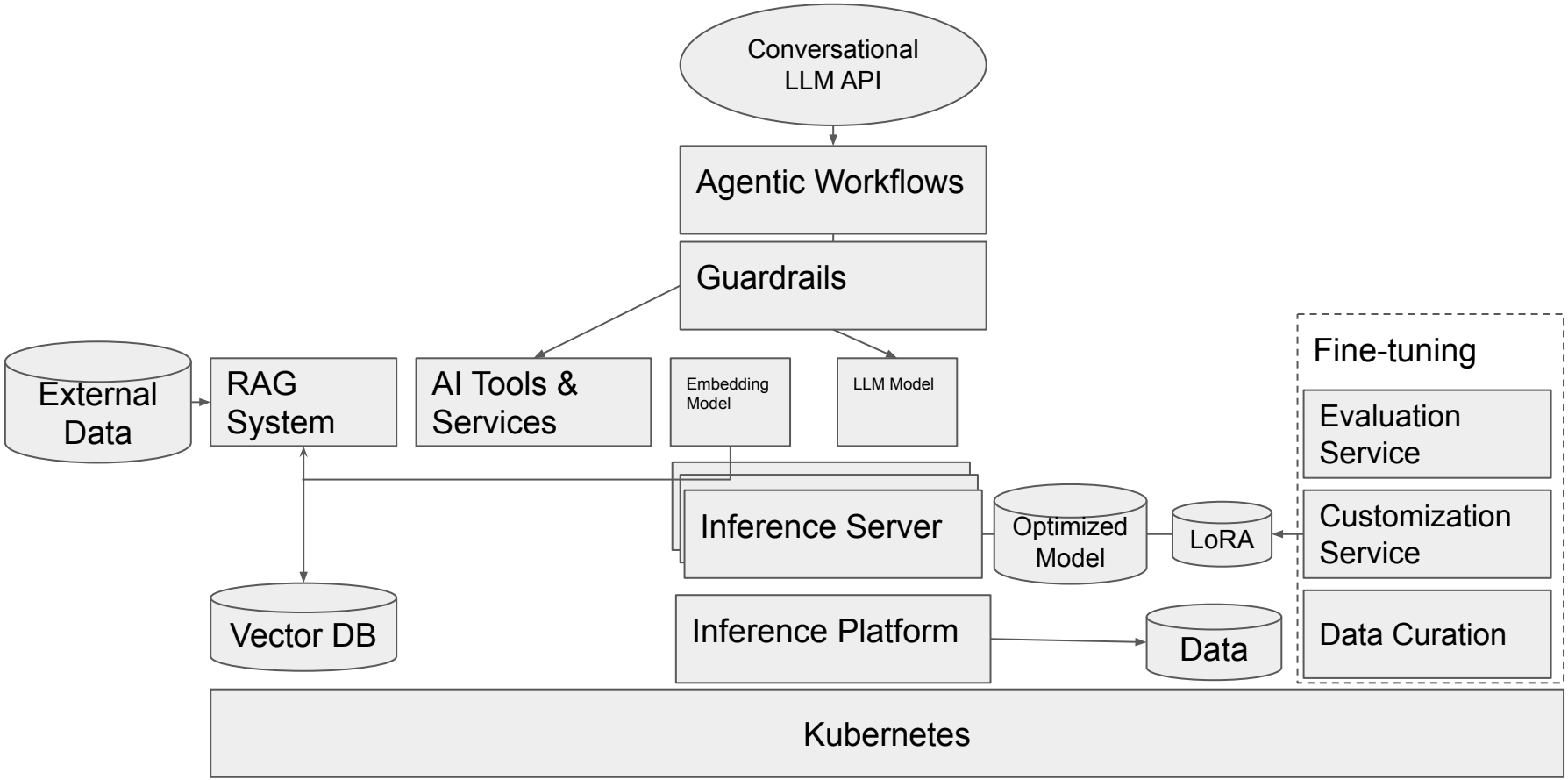
Introducing explainability with RAG



Improving accuracy with fine-tuning



Going from MVP to production



Evaluation Techniques

Training Accuracy Evaluation

Various evaluation technique for accuracy/speed

- Does the system answer predefined questions correctly? (Academic benchmarks)
  - Standard benchmarks
  - Industry benchmarks
- Does the system answer predefined questions about your use-case? (Custom benchmarks)
  - Company-specific benchmarks
- Does the system perform “better” than other LLMs?
  - Automated comparisons (LLM-as-judge)
  - Manual expert comparisons (Human-as-judge)
- How does the throughput and latency compare to similar models (performance)

Inference Evaluation

Metrics are evaluated system-wide and per-model

- How many tokens are generated? (Throughput)
  - Overall
  - Per-user
- How fast do users get a response? (Latency)
  - Average
  - Max
  - Min
  - 95%
  - Time to first token (TTFS)
- Which accelerators are being used? (Cost)

Operational Issues

Scaling

- Scaling metrics
- Startup times
- Scheduling

Monitoring

- Standard application metrics
- Infrastructure metrics
- Automated guardrails

Caching

- Models
- Containers
- Customizations
- Fine-tuning data

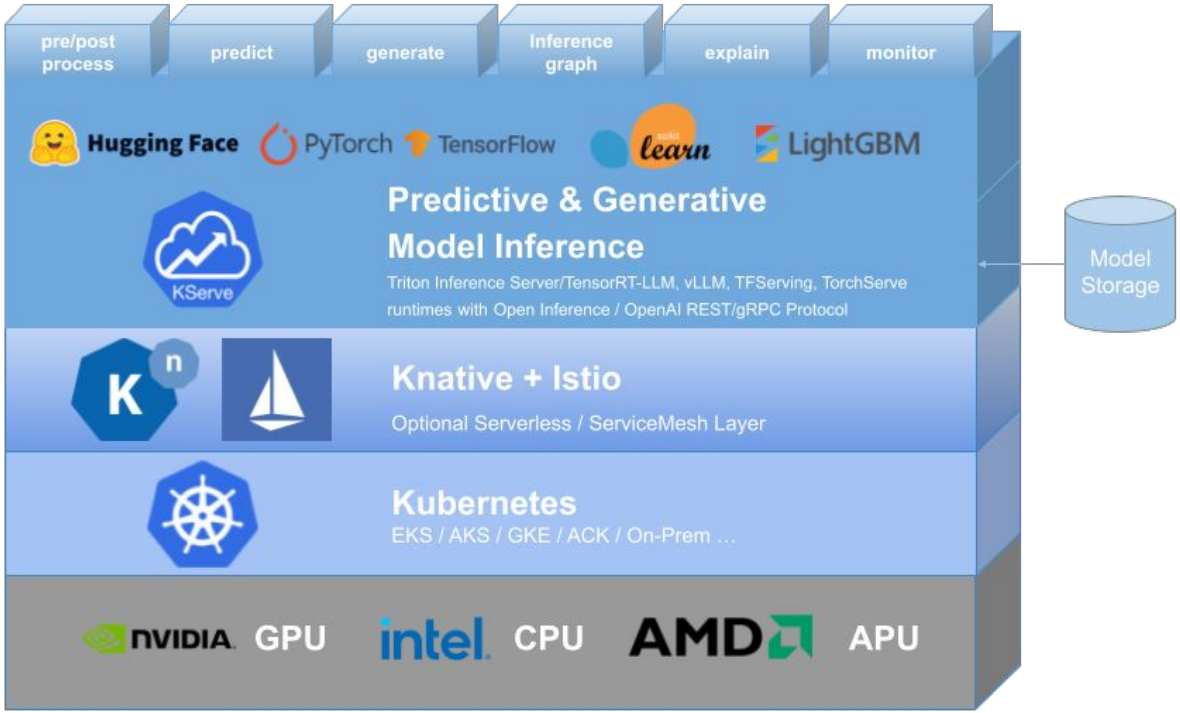
Upgrades

- A/B Testing
- Rolling Upgrades
- Hardware availability
- LLM Routing

Solutions to serving large models, KServe a case study

What is KServe?

Highly scalable, standard, cloud agnostic model inference platform on Kubernetes



Why KServe?

- Provides performant, standardized inference protocol across ML frameworks.
- Provides high scalability, density packing, and intelligent routing using ModelMesh.
- Advanced deployments with canary rollout, ensembles and transformers.
- Supports modern serverless inference workload with autoscaling including Scale to Zero on GPU.
- Simple and pluggable production serving for production ML serving including predictive/generative AI, pre/post processing, monitoring and explainability.

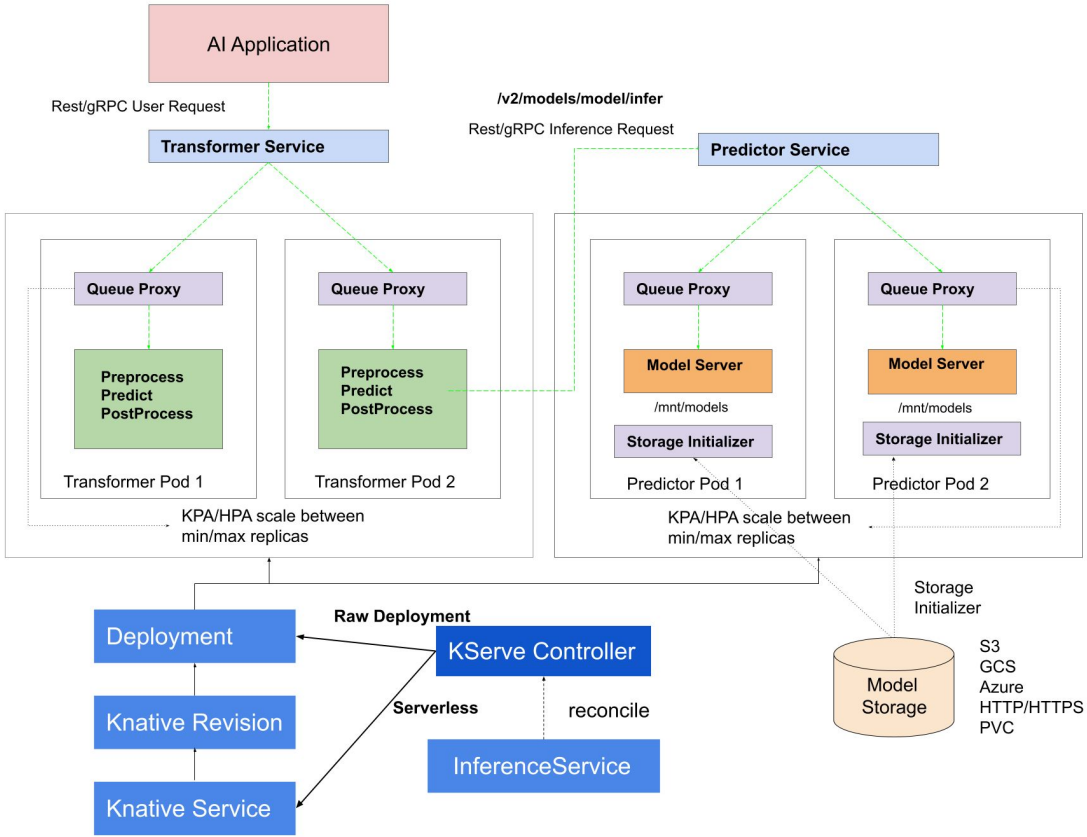
Community: Maintainers and Contributors

15 maintainers and 250+ contributors!

Community: Adopters

30+ companies varying from vendors to end users!

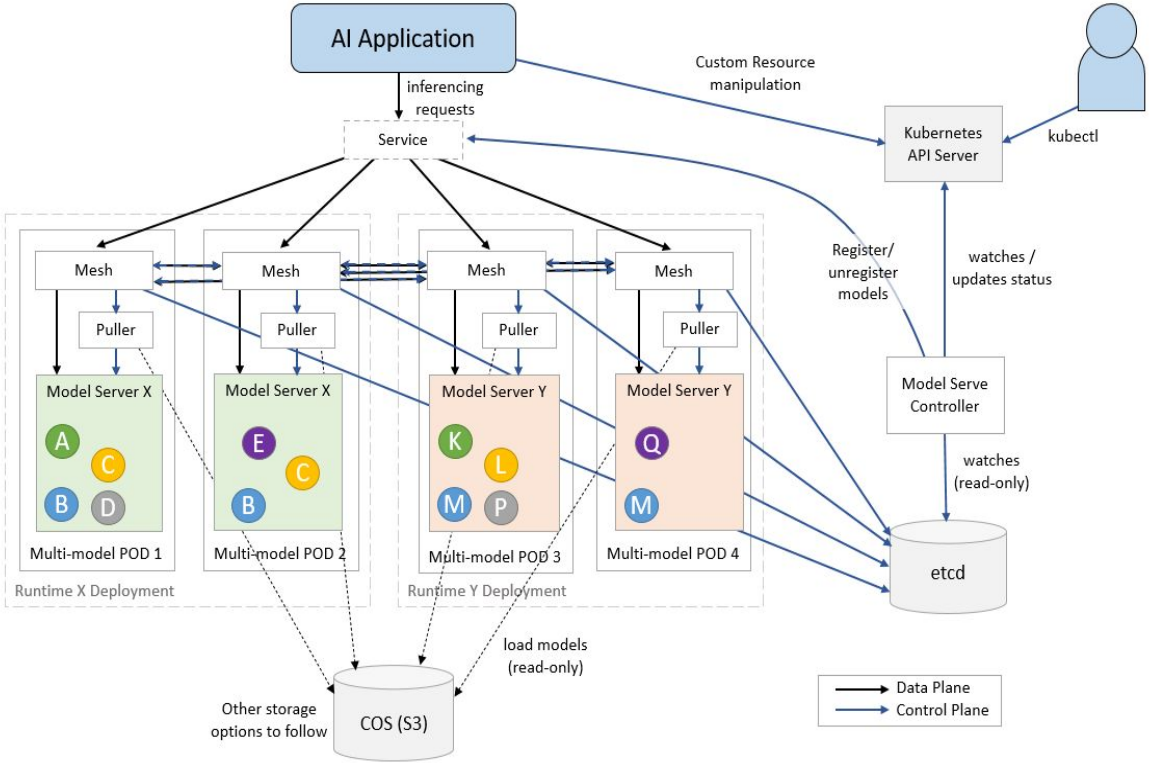
KServe Architecture: Single Model



KServe Architecture: Multi Model (ModelMesh)

Designed for high-scale, high-density and frequently-changing model use cases.

Intelligently loads and unloads models to and from memory to strike an intelligent trade-off between responsiveness to users and computational footprint.



KServe Key Features: Serving Runtimes

Pluggable, reusable, extensible runtimes

- Framework specific: PyTorch, TensorFlow, Sklearn, Paddle, XGBoost, LightGBM, PMML
- Generic/Multi-framework support: Hugging Face (vLLM and Triton backends), MLServer, ONNX
- Custom runtime support

KServe Key Features: Serving Runtimes

Hugging Face Runtime
(vLLM and Triton backends)

```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: huggingface-llama3
spec:
  predictor:
    model:
      modelFormat:
        name: huggingface
      args:
        - --model_name=llama3
        - --model_id=meta-llama/meta-llama-3-8b-instruct
      resources:
        limits:
          cpu: "6"
          memory: 24Gi
          nvidia.com/gpu: "1"
        requests:
          cpu: "6"
          memory: 24Gi
          nvidia.com/gpu: "1"
```

KServe Key Features: Model Storage

Supported storage formats

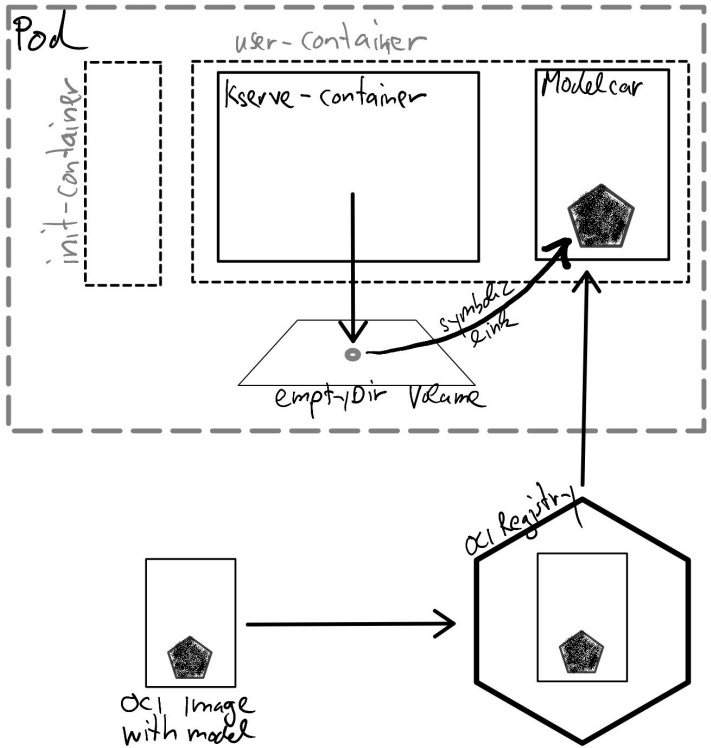
- Azure, S3, GCS
- URI
- PVC
- Storage Containers
  - e.g. custom mode-registry:// protocol with Kubeflow Model Registry
- OCI (via ModelCar)

KServe Key Features: Model Storage - ModelCars

Efficient model pulling from OCI image registry

- Reduced Startup Times: By avoiding repetitive downloads of large models, startup delays are significantly minimized.
- Lower Disk Space Usage: The feature decreases the need for duplicated local storage, conserving disk space.
- Enhanced Performance: ModelCars allows for advanced techniques like prefetching images and lazy-loading, improving efficiency.

KServe Key Features: Model Storage - ModelCars



- OCI URI schema as the model reference.
- Replacing the init-container with a sidecar container.
- Configuring the pod with shareProcessNamespace: true to facilitate access between sidecars.
- Accessing to model data in a sidecar container without copying it into a shared volume. No init-container is needed.

KServe Key Features: Model Storage - ModelCars

Build and Push Image with Model

```
FROM busybox
RUN mkdir /models && chmod 775 /models
COPY data/ /models/
```

Specify StorageURI

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: my-inference-service
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: oci://myuser/mymodel:1.0
```

KServe Key Features: Multi-node Inference

Multi-node inference via Hugging Face Runtime (vLLM backend)

```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/deploymentMode: RawDeployment
    serving.kserve.io/autoscalerClass: external
  name: huggingface-llama3
spec:
  predictor:
    model:
      runtime: kserve-huggingfaceserver-multinode
      modelFormat:
        name: huggingface
      storageUri: pvc://llama-3-8b-pvc/hf/8b_instruction_tuned
    workerSpec:
      pipelineParallelSize: 2
      tensorParallelSize: 1
```

KServe Key Features: Autoscaling

Knative autoscaling

```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "flowers-sample"
spec:
  predictor:
    scaleTarget: 1
    scaleMetric: concurrency
    model:
      modelFormat:
        name: tensorflow
      storageUri: "gs://kfserving-examples/models/tensorflow/flowers"
```

KServe Key Features: Autoscaling

Send traffic in 30 seconds spurts maintaining 5 in-flight requests

hey -z 30s -c 5 ...

```
NAME                                                  READY   STATUS    RESTARTS   AGE
flowers-sample-default-7kqt6-deployment-75d577dcdb-sr5wd   3/3     Running   0          42s
flowers-sample-default-7kqt6-deployment-75d577dcdb-swnk5   3/3     Running   0          62s
flowers-sample-default-7kqt6-deployment-75d577dcdb-t2njf   3/3     Running   0          62s
flowers-sample-default-7kqt6-deployment-75d577dcdb-vdlp9   3/3     Running   0          64s
flowers-sample-default-7kqt6-deployment-75d577dcdb-vm58d   3/3     Running   0          42s
```

KServe Key Features: Autoscaling

Scale down to zero

```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "flowers-sample"
spec:
  predictor:
    minReplicas: 0
    model:
      modelFormat:
        name: tensorflow
      storageUri: "gs://kfserving-examples/models/tensorflow/flowers"
```

KServe Key Features: Canary Rollout Strategy

KServe Key Features: Monitoring

Prometheus Metrics

- Prometheus latency histograms are emitted for each of the steps (pre/postprocessing, explain, predict).
- The latencies of each step are logged per request.
- Each model server may implement its own set of metrics.
- Knative/Queue proxy emits metrics by default

KServe Key Features: Monitoring

Grafana Dashboards

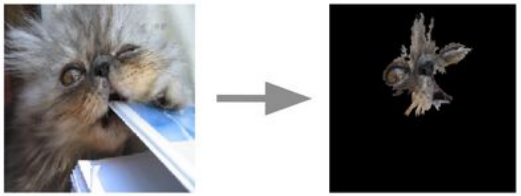
- Knative HTTP Dashboard
- KServe Dashboards for Runtimes
  - ModelServer Latency Dashboard
  - TorchServe Latency Dashboard
  - Triton Latency Dashboard
  - ...

KServe Key Features: Explainer/TrustyAI

Pluggable explainer runtimes

“Why did my model produce this inference result?”

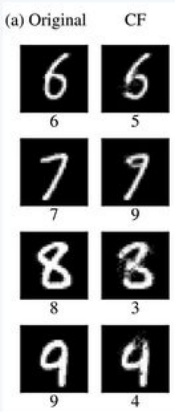
Anchor explanations for images



Integrated Gradients for text

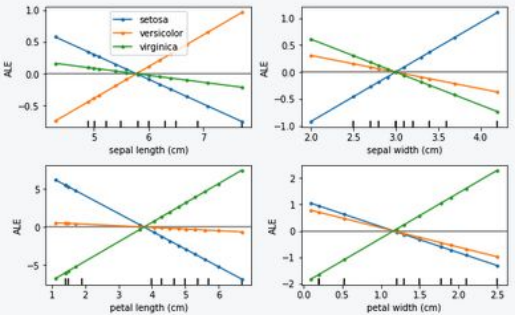
a powerful study of loneliness sexual UNK and desperation be patient UNK up the atmosphere and pay attention to the wonderfully written script br br i praise robert altman this is one of his many films that deals with unconventional fascinating subject matter this film is disturbing but it's sincere and it's sure to UNK a strong emotional response from the viewer if you want to see an unusual film some might even say bizarre this is worth the time br br unfortunately it's very difficult to find in video stores you may have to buy it off the internet

Counterfactual examples



(b)	Original	CF
Workclass	Private	State-gov
Education	High school	Bachelors
Marital Status	Married	Married
Occupation	Blue-Collar	Blue-Collar
Relationship	Husband	Husband
Race	White	White
Sex	Male	Male
Country	United-States	United-States
Age	46	46
Capital Gain	0	0
Capital Loss	0	0
Hours p/w	40	40
Prediction	$\leq \$50k/y$	$> \$50k/y$

Accumulated Local Effects



Future problems to solve, gaps, and community engagement

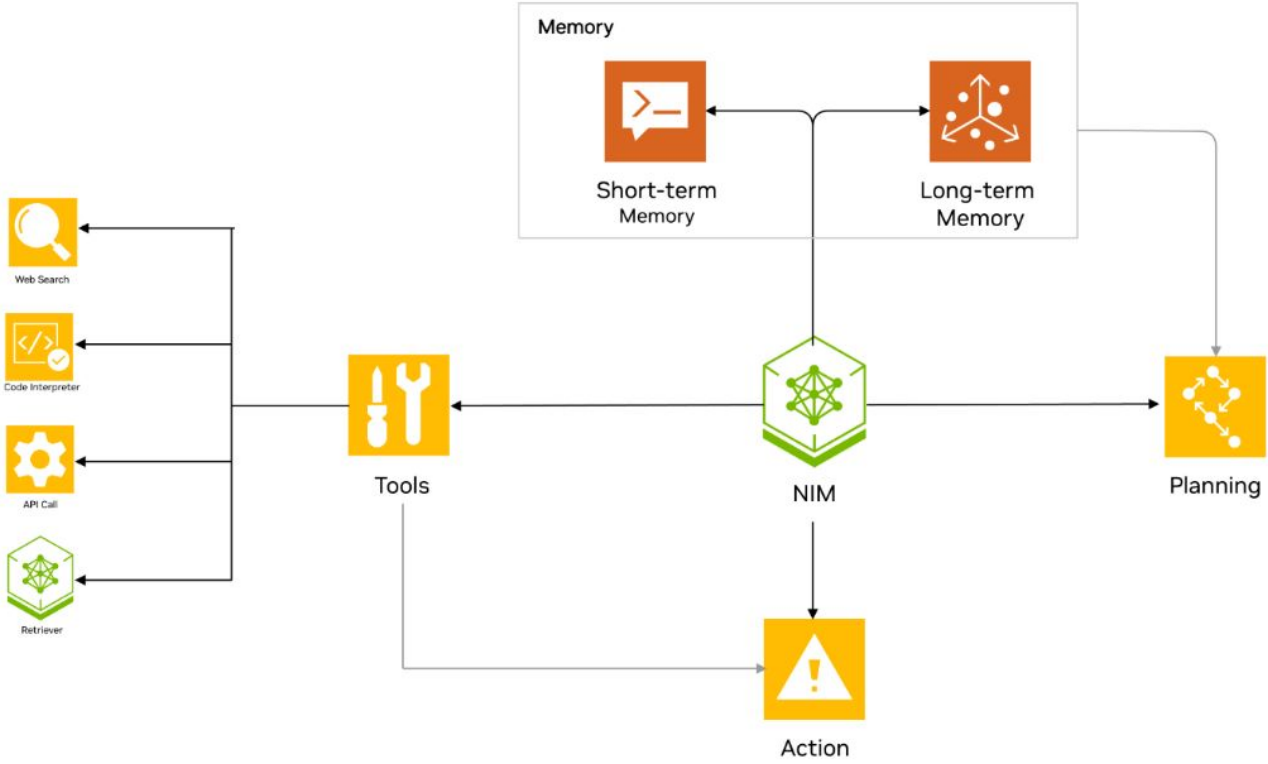
Agents and Tools

Reasoning and Planning:
Decomposing complex tasks into manageable subgoals through reasoning

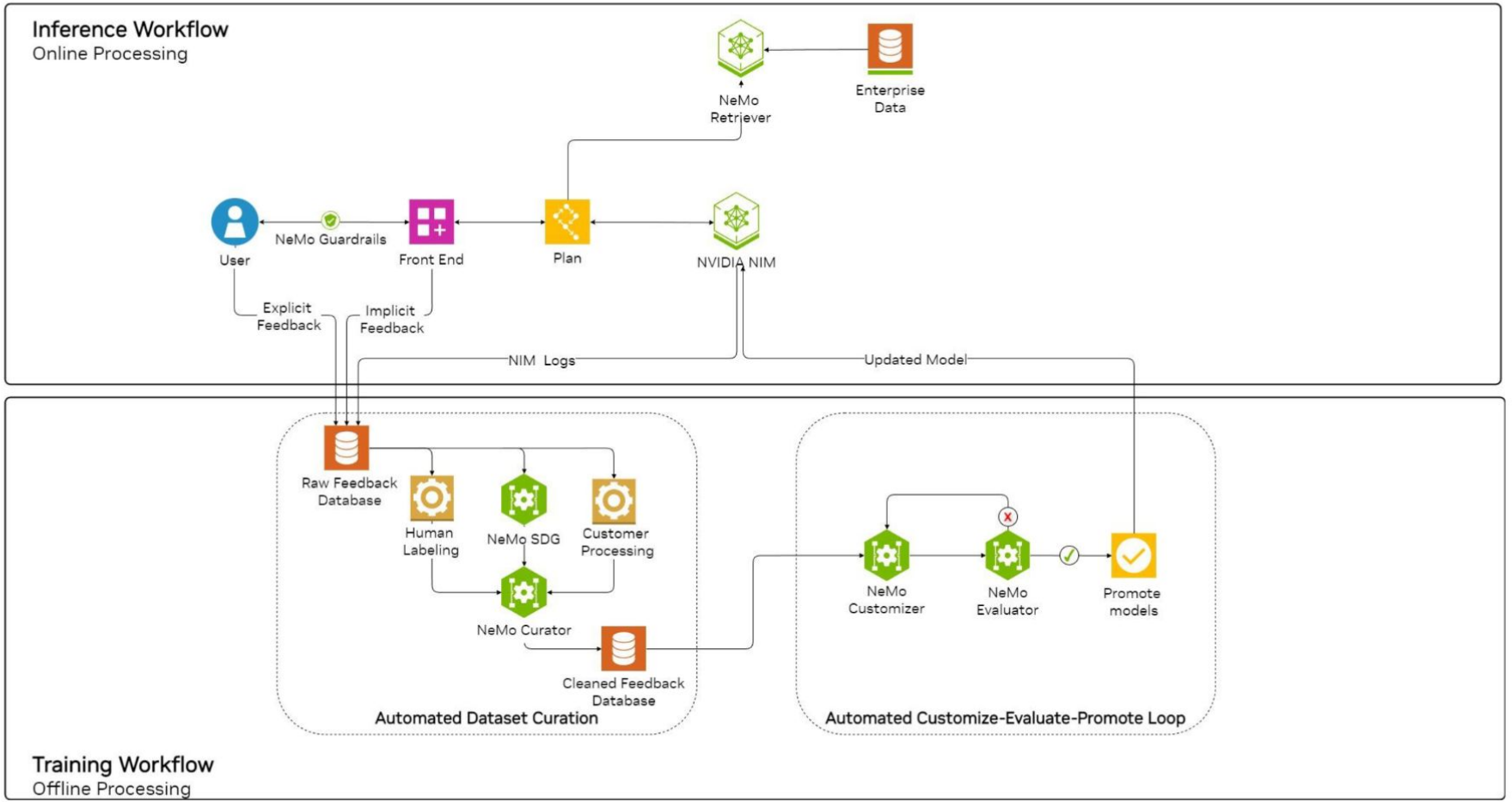
Memory:
Short-term memory in an LLM-powered agent acts as a record of actions and thoughts during a single query

Long-term memory:
Logs interactions between the user and agent over extended periods

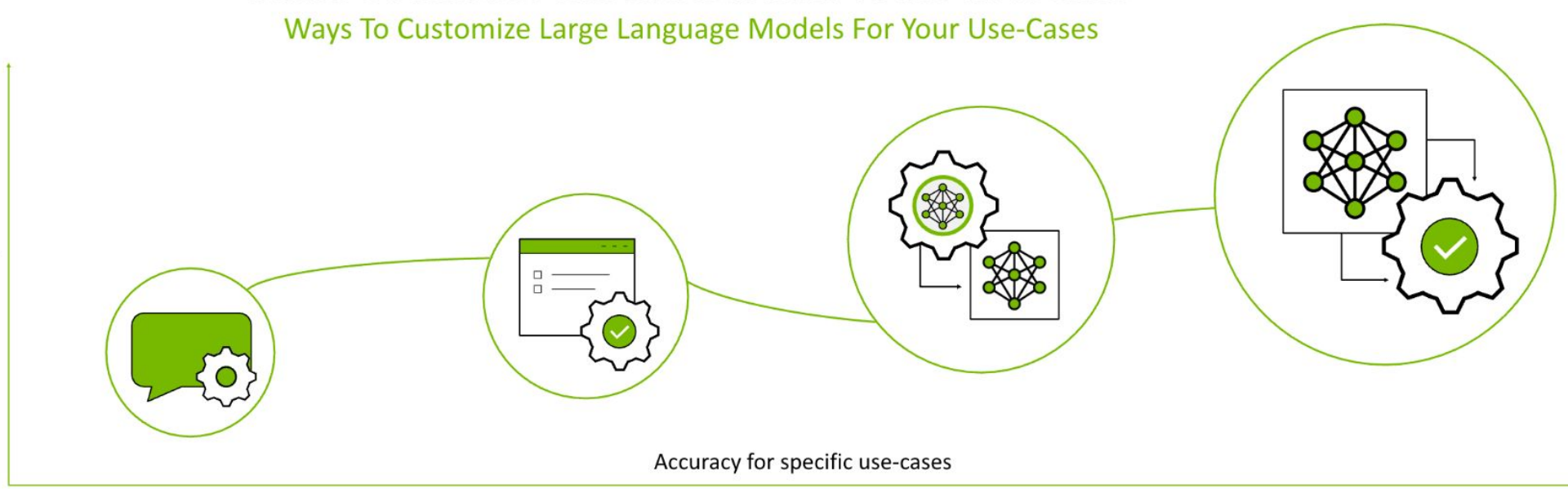
Tools:
Defined executable workflows that agents use to perform tasks



End to End Inference & Training Workflow



Training Techniques



PROMPT ENGINEERING	PROMPT LEARNING	PARAMETER EFFICIENT FINE-TUNING	FINE TUNING
. Few-shot learning . Chain-of-thought reasoning . System prompting	. Prompt tuning . P-tuning	. Adapters . LoRA . IA3	. SFT . RLHF

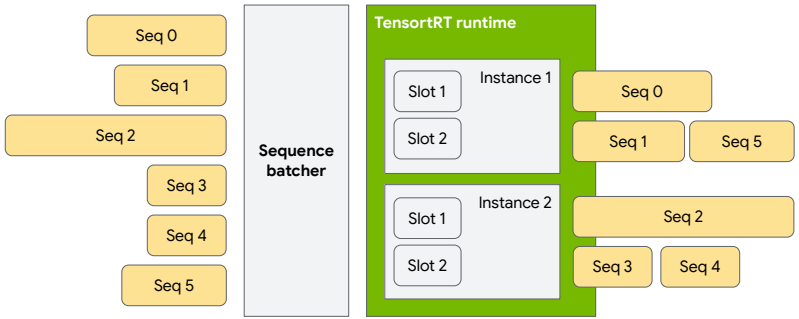
Inference Optimizations

Inference Platform Features

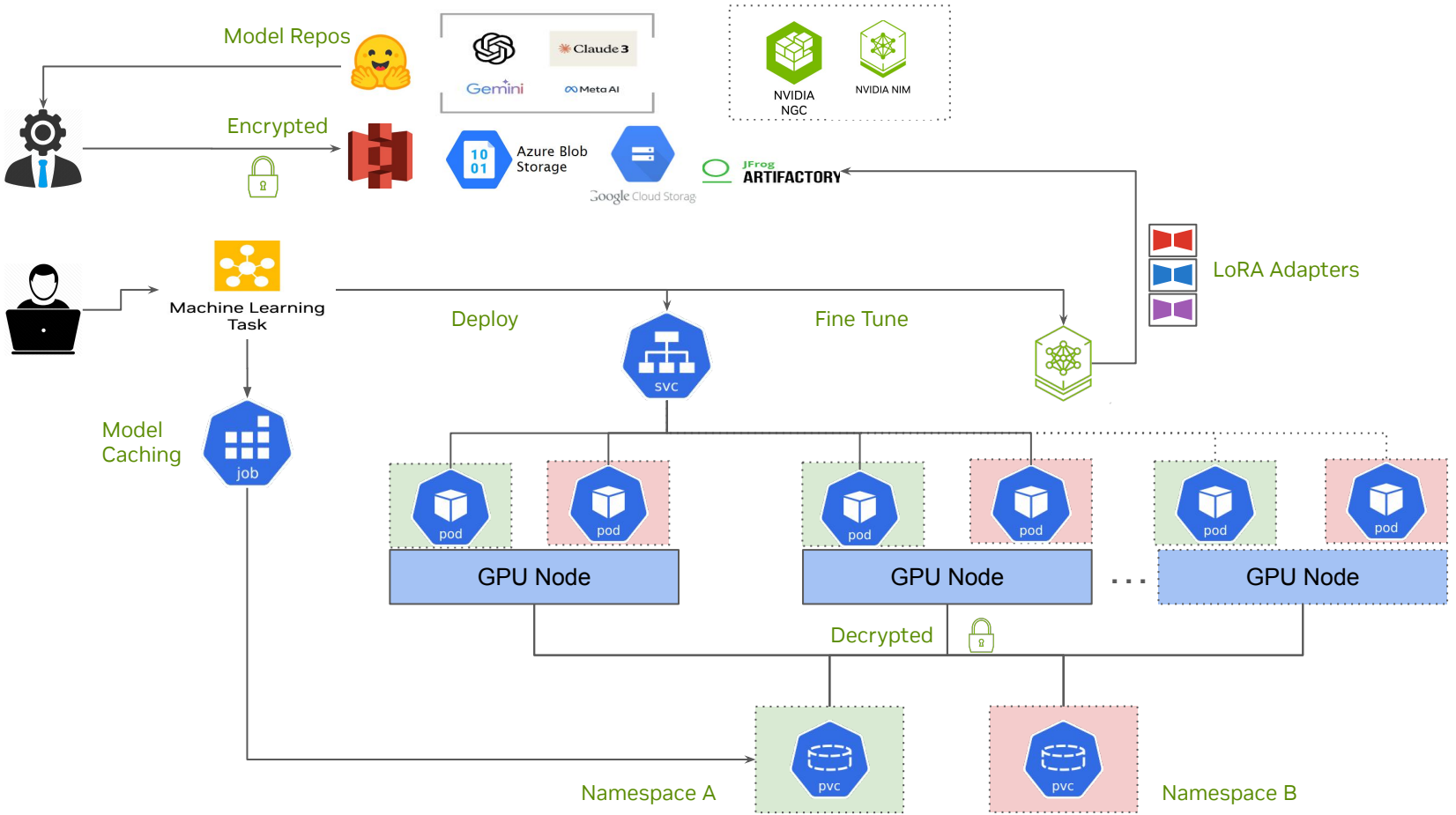
- Response Caching
- Context Caching
- Inflight Batching

Model Features

- Multi-LoRA Loading
- Just-in-time Compilation (vLLM)
- Ahead-of-time Compilation (TRT-LLM)



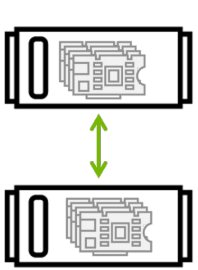
Model Cache Management



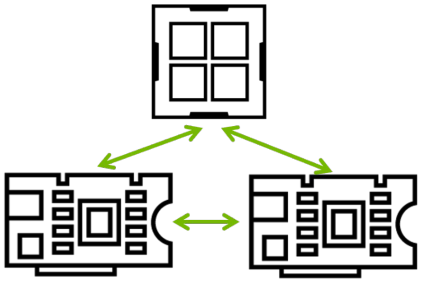
Multi-node Inference: Models too big for 1 node

Scheduling

- Avoiding deadlock (Gang Scheduling)
- Avoiding unusable GPUs (Binpacking)



Multi-Node



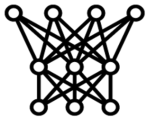
Multi-GPU

Operations

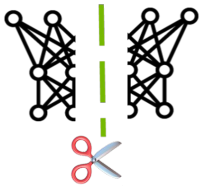
- Monitor individual and groups of Pods
- Handling scaling complexity
- Optimizing for throughput or cost

Optimizations

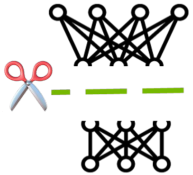
- Accelerate model loading (Caching on Shared Storage)
- Accelerate cross-node communication (Advanced Networking with RoCE / RDMA)



No Parallelism



Tensor Parallel



Pipeline Parallel

Future of KServe: GenAI Inference

Serving Runtimes

- Support LoRA adapters, speculative decoding (vLLM)
- TensorRT-LLM, TGI, and benchmarking
- Improve multi-node/multi-host inference

Autoscaling

- Support model caching with automatic PVC/PV provisioning
- Autoscaling based on custom metrics

Agent/RAG Pipeline Orchestration

- Support declarative RAG/Agent workflow using KServe Inference Graph

Future of KServe: GenAI Inference

Open Inference Protocol extension to GenAI Task APIs

- Community-maintained Open Inference Protocol repo for OpenAI schema
- Support vertical GenAI Task APIs such as embedding, Text-to-Image, Text-To-Code, Doc-To-Text

LLM Gateway

- Support multiple LLM providers.
- Support token based rate limiting.
- Support LLM router with traffic shaping, fallback, load balancing.
- LLM Gateway observability for metrics and cost reporting

Roadmap: https://github.com/kserve/kserve/blob/master/ROADMAP.md

Future of KServe: Community Collaboration

- Kubernetes WG Serving
  - Check out our talk on Friday: *WG Serving: Accelerating AI/ML Inference Workloads on Kubernetes*
- Collaboration with corporate partners and community maintainers
  - Check out our panel at 5:25pm today: *Engaging the KServe Community, The Impact of Integrating a Solutions with Standardized CNCF Projects*
- Collaboration and Integration with Ecosystem Projects
  - vLLM
  - Kubeflow
  - Envoy
  - …

WG-Serving & KServe Talks

AI Day, Sponsored Keynote: Advancing Cloud Native AI Innovation Through Open Collaboration

Optimizing Load Balancing and Autoscaling for Large Language Model (LLM) Inference on Kubernetes

Engaging the KServe Community, The Impact of Integrating a Solutions with Standardized CNCF Projects (today at 5:25pm)

Best Practices for Deploying LLM Inference, RAG and Fine Tuning Pipelines on K8s

WG Serving: Accelerating AI/ML Inference Workloads on Kubernetes

Sessions from NVIDIA

Sessions from Red Hat

Q&A