THE LINUX FOUNDATION

AI_dev
Open Source GenAI & ML Summit

Running and Fine tuning Open Source LLMs on Google Kubernetes Engine

Nael Fridhi, Google Cloud

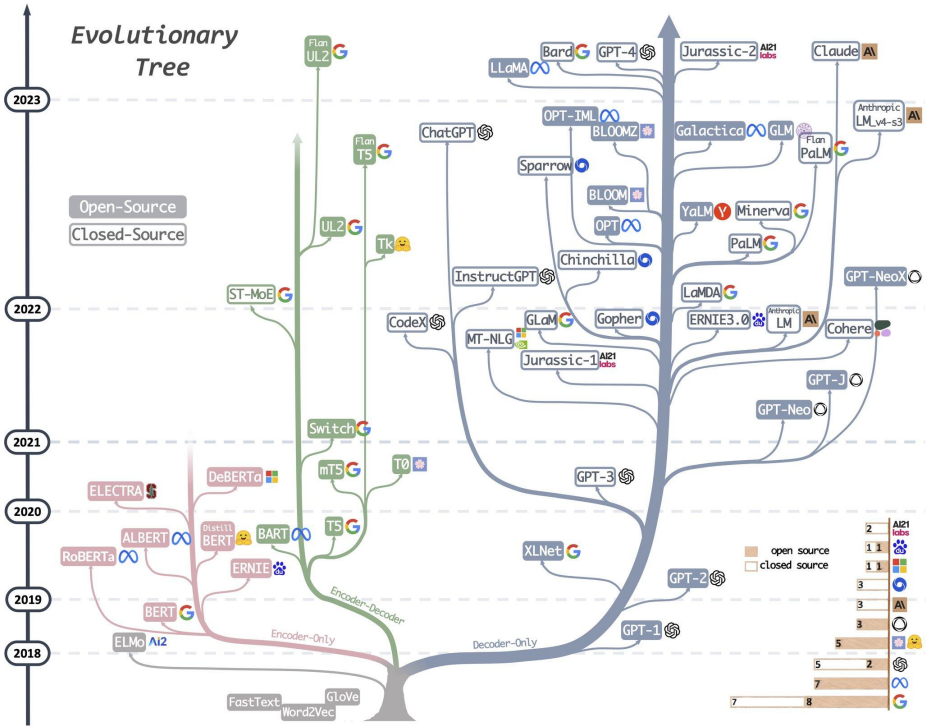
Google Cloud

Agenda:

1. Introduction
2. Why Open Source LLMs?
3. Why Kubernetes?
4. Running LLMs on GKE best practices
5. LLMs Fine-tuning
6. Fine-tuning LLMs on GKE best practices
7. Demo
8. Q & A

Introduction

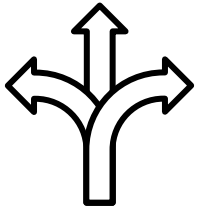
- GenAI capabilities, reasoning and performance are continually improving
- Rapid evolution of LLMs which demonstrates the power of open-source research and collaboration in driving innovation.
- Increased accessibility



https://arxiv.org/pdf/2304.13712v2.pdf

Why Open Source LLMs?

Cost-effectiveness



Flexibility & customization

Community-driven

Privacy & Compliance

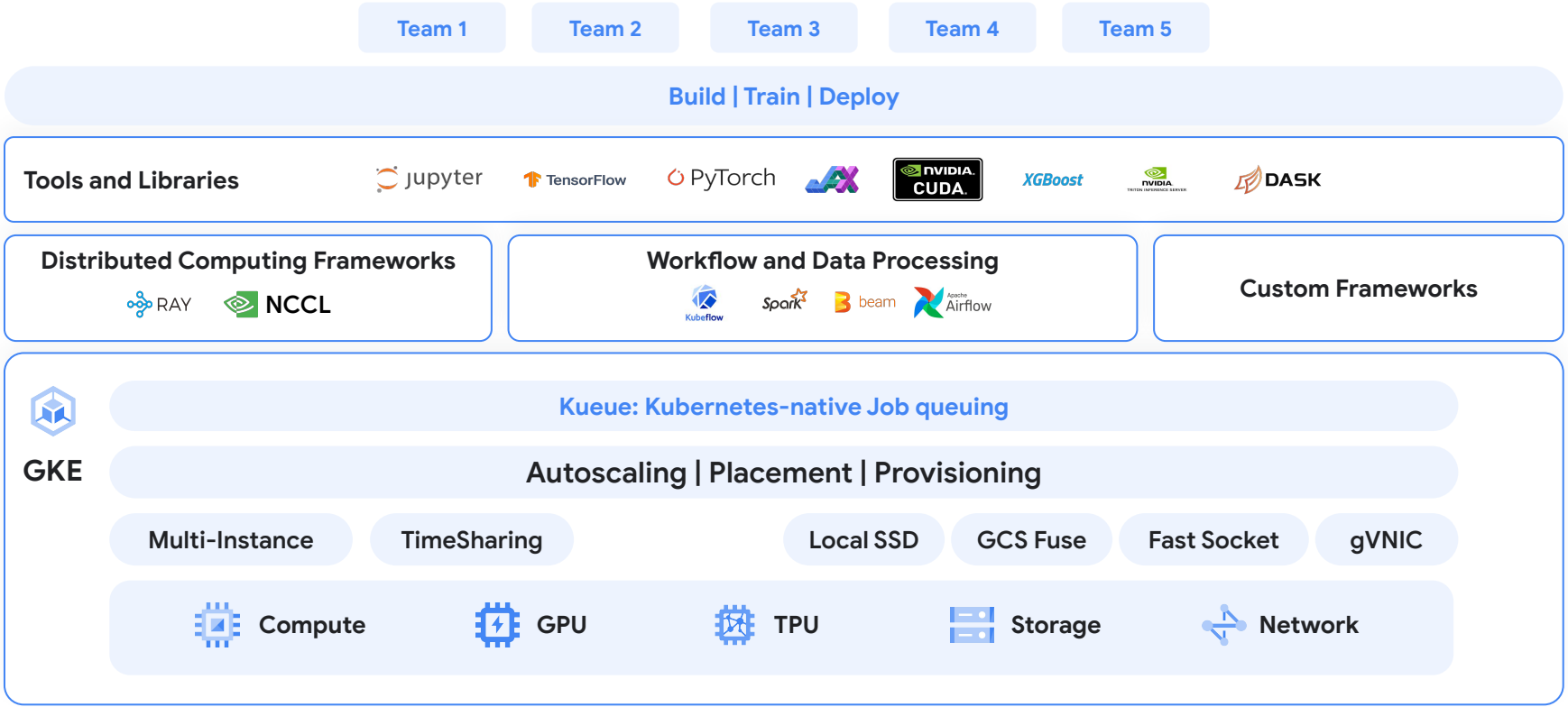
Why Kubernetes?

Flexibility Choice of **frameworks** and **ecosystem tools** that are container **portable**

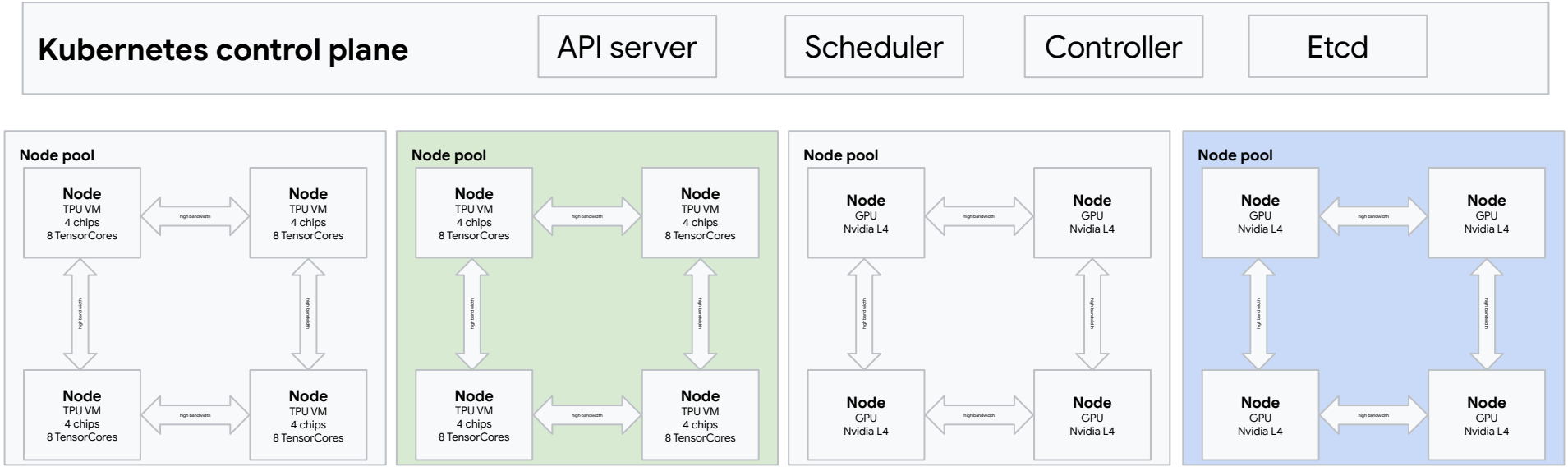
Performance Orchestrate **AI models** at massive scale across **specialized compute**

Efficiency Optimize **valuable compute resources** while **reducing operational complexity**

Unified AI/ML Platform on GKE



GKE Node Auto-Provisioning to dynamically spin up new TPUs or GPUS



Example 1: If another workload needs a TPU v4 with 2x2x4 topology, GKE will spin up a second node pool.

Example 2: If another workload needs more GPU Nvidia L4 type GKE will spin up a second node pool.

Sharing GPUs with GKE

Multi-instance GPUs

- Partition one A100 or H100 GPUs into up to 7 slices
- Hardware isolation from other containers on the same physical GPU
- Predictable throughput and latency for parallel workloads

Multi-process Service (MPS)

- Up to 48 parallelized partitions
- Software based isolation works on any NVIDIA GPU
- Predictable throughput and latency for parallel workloads

GPU Time-Sharing

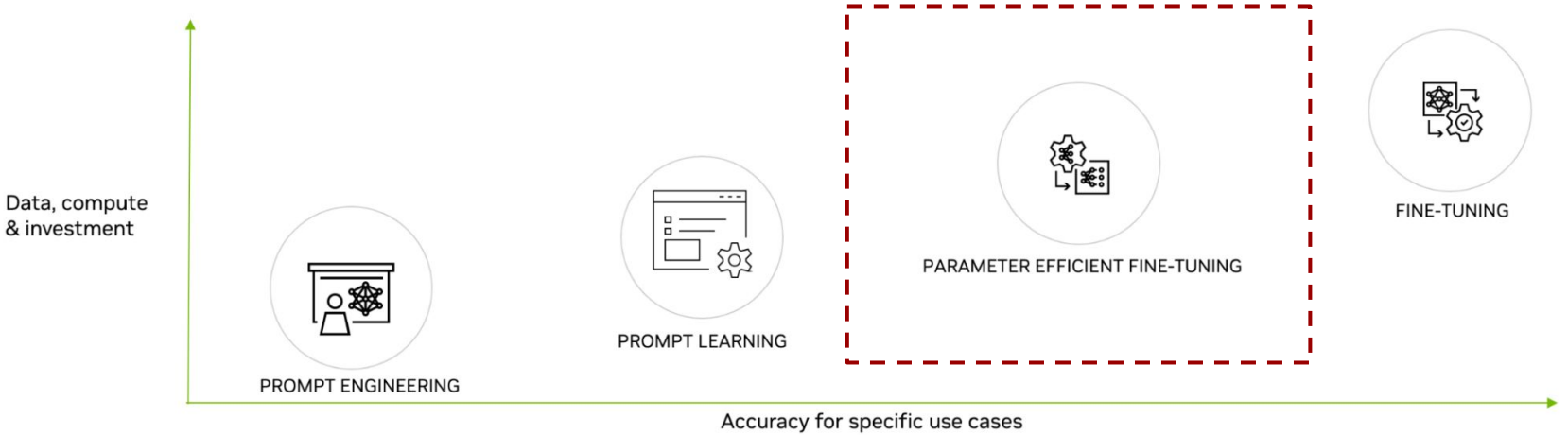
- Up to 48 fractional GPU units with time-context shifting
- Workloads with low GPU requests
- Burstable GPU workloads

Choosing between TGI and vLLM

Feature	Text Generation Inference	vLLM
Throughput	Lower throughput, especially for larger models	Higher throughput, optimized for speed
Memory Management	Can lead to memory bottlenecks and waste	Efficient memory use with Paged Attention
Model Compatibility	Supports various LLMs, including quantization	Supports popular LLMs, growing ecosystem
Ease of Use	Relatively easy to set up and use	Easy setup, streamlined deployment
Customization	Allows for fine-tuning	Limited options
Use Cases	Versatile, suitable for diverse application	Best for high-performance serving

```yaml
spec:
  containers:
  - name: llama-2-70b
    image: ghcr.io/huggingface/text-generation-inference:1.0.3
    resources:
      limits:
        nvidia.com/gpu: 2
    env:
    - name: MODEL_ID
      value: meta-llama/Llama-2-70b-chat-hf
    - name: NUM_SHARD
      value: "2"
    - name: PORT
      value: "8080"
    - name: QUANTIZE
      value: bitsandbytes-nf4
```

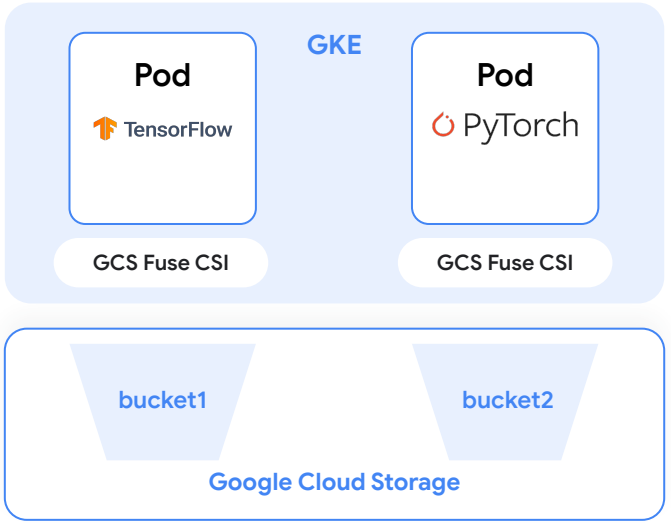
Ways To Customize LLM Models



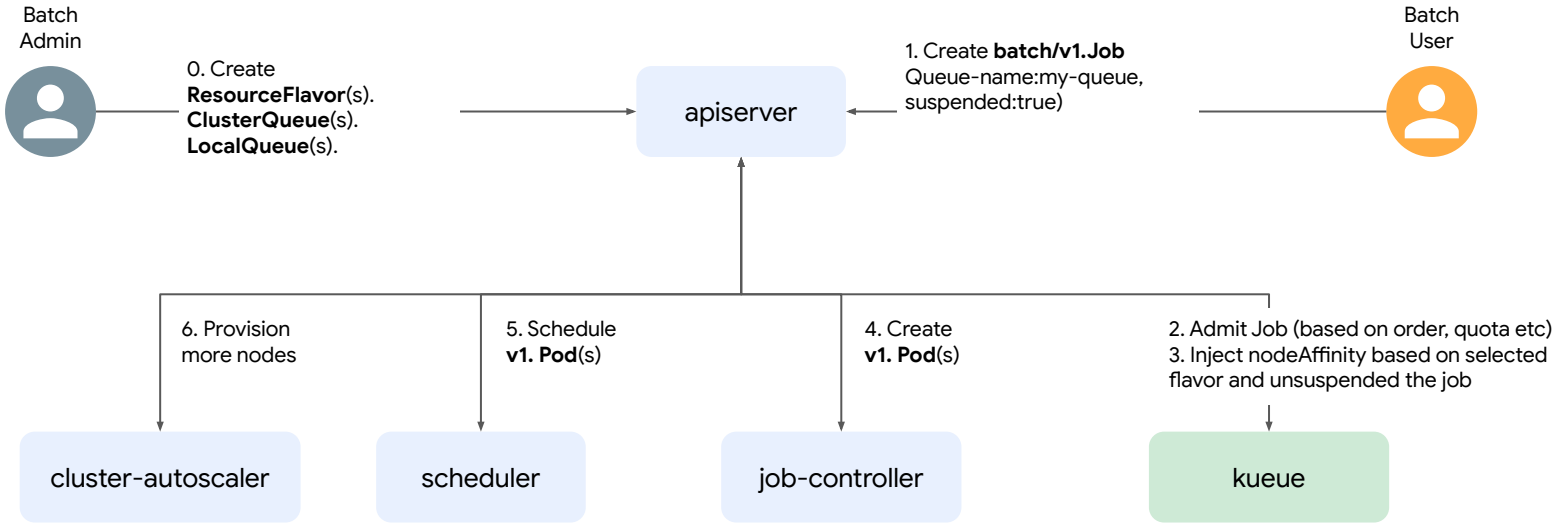
TECHNIQUES	• Few-shot learning • Chain-of-thought reasoning • System prompting	• Prompt tuning • p-tuning	• Adapters • LoRA • IA3	• SFT • RLHF
HOW	Prompt templates	Tune companion model	Add "custom" layers and parameters to LLM	Tune LLM model weights
BEST TO USE WITH	Limited prompt-completion examples & simple use cases	A few hundred examples & use cases where prompt engineering is not sufficient	Hundreds of examples for a multitude of downstream tasks	Thousands of examples & complex use cases
EXAMPLE	Extractive Q+A	Named entity extraction	Summarization, paraphrasing, etc.	Multilingual, biomedical, coding, etc.

GKE ML Data Layer with Google Cloud Storage (GCS) FUSE

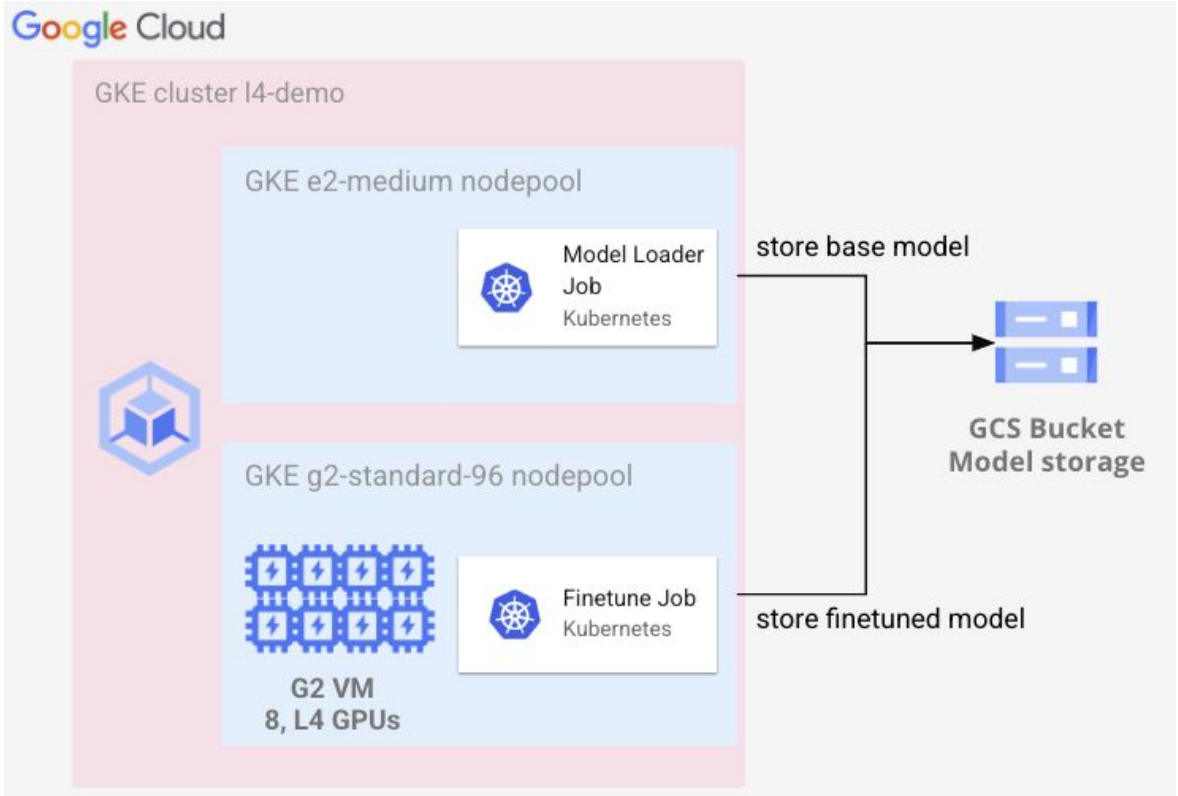
- **Portable**: Access GCS buckets as a filesystem mounted on local/cloud machines. No need to use the GCS API!
- Ideal for Machine Learning & AI workloads that value portability, simplicity, scale, and cost
- Provides **high throughput storage** for read heavy Machine Learning workloads
- Allows reading datasets that are already stored in GCS directly which **saves compute resources** from waiting for data to be fetched
- Provides the ability to write “checkpoints” and results back to GCS
- Native integration with GKE via fully managed CSI driver



Kueue: Kubernetes-native job queueing system



Llama fine-tuning on GKE Architecture



```
apiVersion: batch/v1
kind: Job
metadata:
 name: model-loader
 namespace: default
spec:
 template:
   metadata:
     annotations:
kubectl.kubernetes.io/default-container:
loader
       gke-gcsfuse/volumes: "true"
       gke-gcsfuse/memory-limit: 400Mi

gke-gcsfuse/ephemeral-storage-limit: 30Gi
   spec:
     restartPolicy: OnFailure
     containers:
     - name: loader
       image: python:3.11
       command:
       - /bin/bash
       - -c
       - |
         pip install huggingface_hub
         mkdir -p /gcs-mount/llama2-7b
         python3 - << EOF
         from huggingface_hub import
snapshot_download
model_id="meta-llama/Llama-2-7b-hf"
snapshot_download(repo_id=model_id,
local_dir="/gcs-mount/llama2-7b",
local_dir_use_symlinks=False,
revision="main",
ignore_patterns=["*.safetensors",
"model.safetensors.index.json"])
         EOF
 imagePullPolicy: IfNotPresent
       env:
       - name: HUGGING_FACE_HUB_TOKEN
         valueFrom:
           secretKeyRef:
             name: huggingface
             key: HF_TOKEN
       volumeMounts:
       - name: gcs-fuse-csi-ephemeral
         mountPath: /gcs-mount
     serviceAccountName: ai-on-gke
     volumes:
     - name: gcs-fuse-csi-ephemeral
       csi:
         driver:
gcsfuse.csi.storage.gke.io
         volumeAttributes:
           bucketName: ai-on-gke-nsandbox
           mountOptions: "implicit-dirs"
```

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: finetune-job
 namespace: default
spec:
 backoffLimit: 2
 template:
   metadata:
     annotations:
       kubectl.kubernetes.io/default-container: finetuner
       gke-gcsfuse/volumes: "true"
       gke-gcsfuse/memory-limit: 400Mi
       gke-gcsfuse/ephemeral-storage-limit: 30Gi
   spec:
     terminationGracePeriodSeconds: 60
     containers:
     - name: finetuner
       image:
us-docker.pkg.dev/google-samples/containers/gke/llama-7b-fine-tune-example
```

```yaml
       resources:
         limits:
           nvidia.com/gpu: 3
       volumeMounts:
       - name: gcs-fuse-csi-ephemeral
         mountPath: /gcs-mount
     serviceAccountName: ai-on-gke
     volumes:
     - name: gcs-fuse-csi-ephemeral
       csi:
         driver: gcsfuse.csi.storage.gke.io
         volumeAttributes:
           bucketName: ai-on-gke-nsandbox
           mountOptions: "implicit-dirs"
     nodeSelector:
       cloud.google.com/gke-accelerator: nvidia-l4
     restartPolicy: OnFailure
```

Other Fine Tuning Key Considerations

Hyperparameter Tuning:

- Experiment with learning rate, batch size, and optimization algorithms.

Regularization Techniques:

- Employ dropout, weight decay, and early stopping to prevent overfitting.

Distributed Training:

- Utilize GKE's distributed training capabilities to scale across multiple nodes and GPUs.

Monitoring and Evaluation:

- Track metrics like loss, accuracy, and perplexity to assess model progress.

Checkpoint Management:

- Use GCS storage for checkpoint management and regularly save model checkpoints to resume training if interrupted.

Demo

Q & A