Fine Tuning Open LLMs on Kubernetes

$ whoami

➜ Abdel Sghiouar

- Senior Cloud Developer Advocate @Google
- Co-Host of the Kubernetes Podcast from Google
- Kubernetes, Containers, Service Mesh and Security
- Moroccan Living in Sweden

What are we talking about today?

Distributed ML on Kubernetes (Fine-tuning)

Running ML model on Kubernetes (Inference)

Open Source tools for ML on Kubernetes

What the future looks like

Fine Tuning

Adapting a pre-trained model to specific tasks:

- Exp: Adapting a Language model trained on English to speak Japanese.

Requires computational power (less than training)

The computation power sometimes exceed a single computer ability to run the workload --> **Distributed**

Distributed Computing

Why use one computer to solve a problem when you can use thousands?

Distributed Computing

Python is the “lingua franca” of AI

With GenAI distributed compute is no longer optional, it is **required**

Why Distributed Computing?

Scalability

Availability

Efficiency

Flexibility

Challenges

Consistency

Fault Tolerance

Concurrency Control

Load Balancing

Security Concerns

Complexity of Management

CAP Theorem

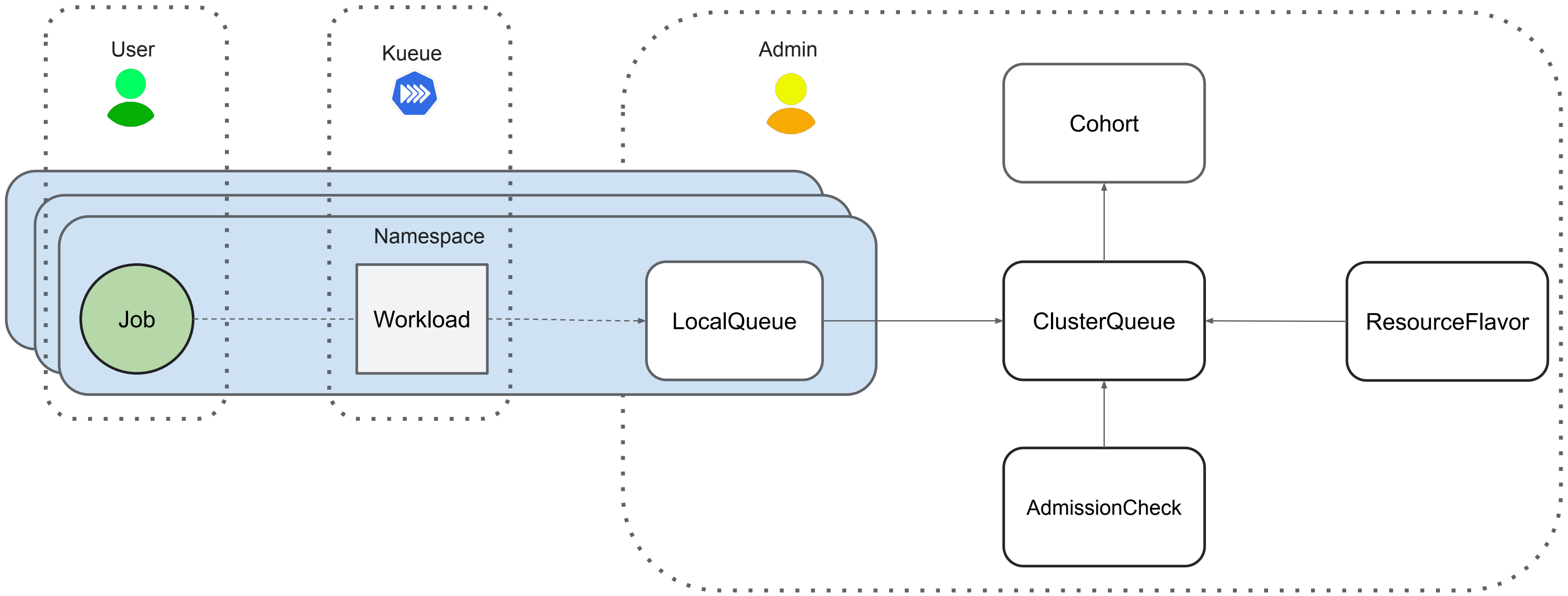
Consistency

Availability

Partition Tolerance

Demo

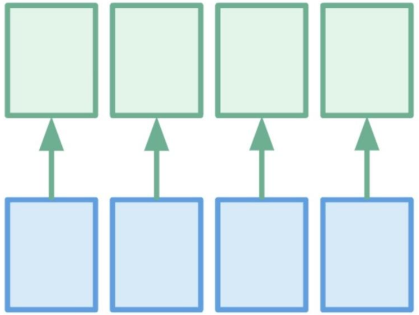
Kueue


User
Kueue
Admin
Cohort
Namespace
Job
Workload
LocalQueue
ClusterQueue
ResourceFlavor
AdmissionCheck

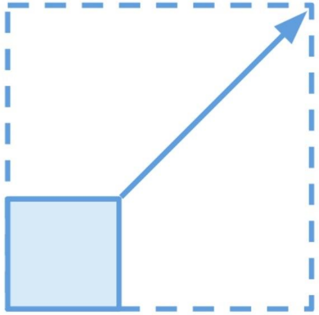
Ray

Ray - Key Characteristics

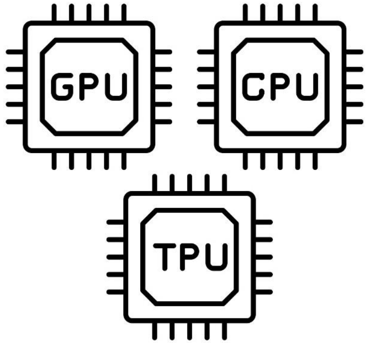
Python first approach, open source



Simple and flexible API

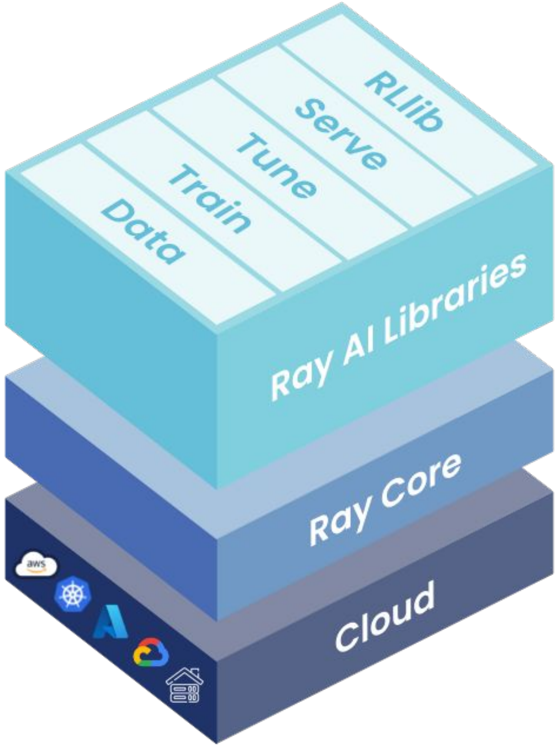


Scalability



Support for bleeding edge hardware

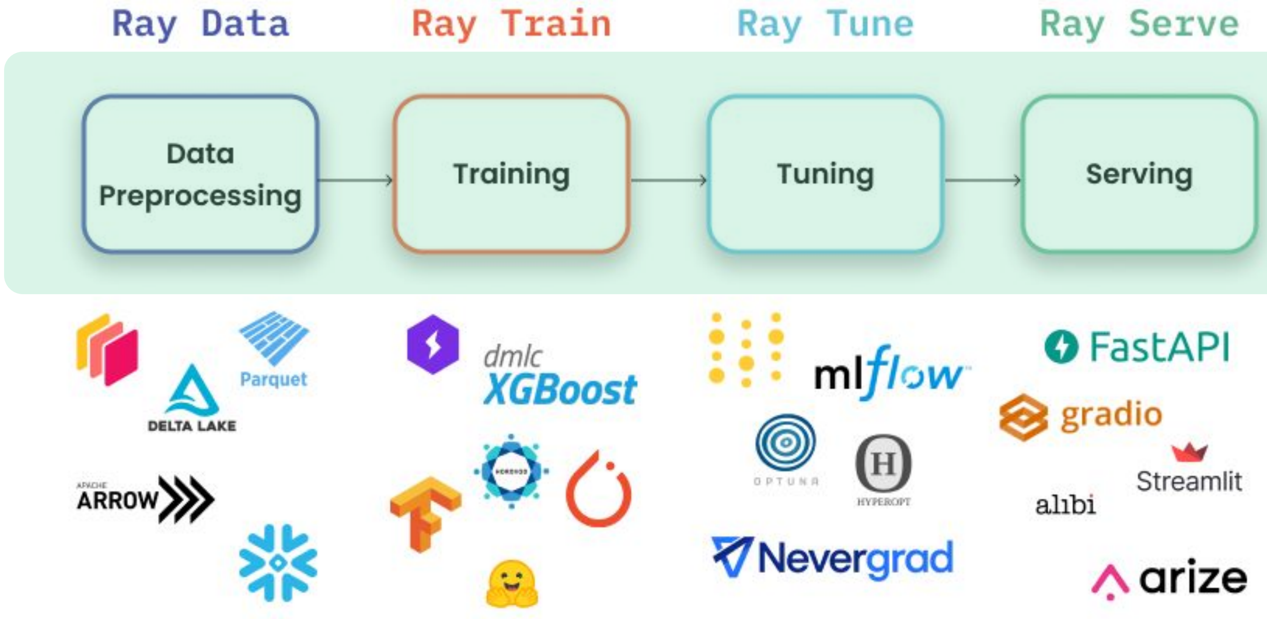
Ray - Components



high-level libraries that enable simple scaling of AI workloads

a low-level distributed computing framework with a concise core and Python-first API

Ray AI Libraries

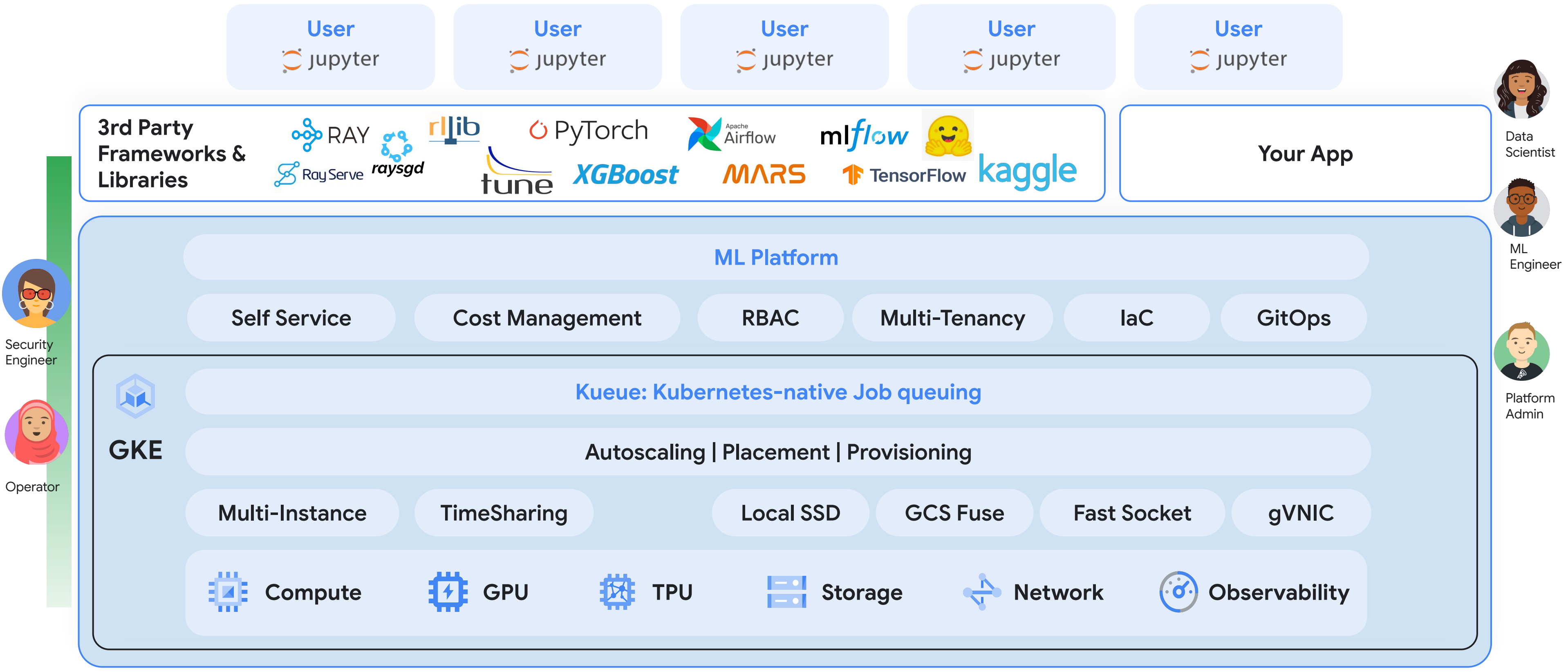


High-level libraries that make scaling easy for both data scientists and ML engineers.

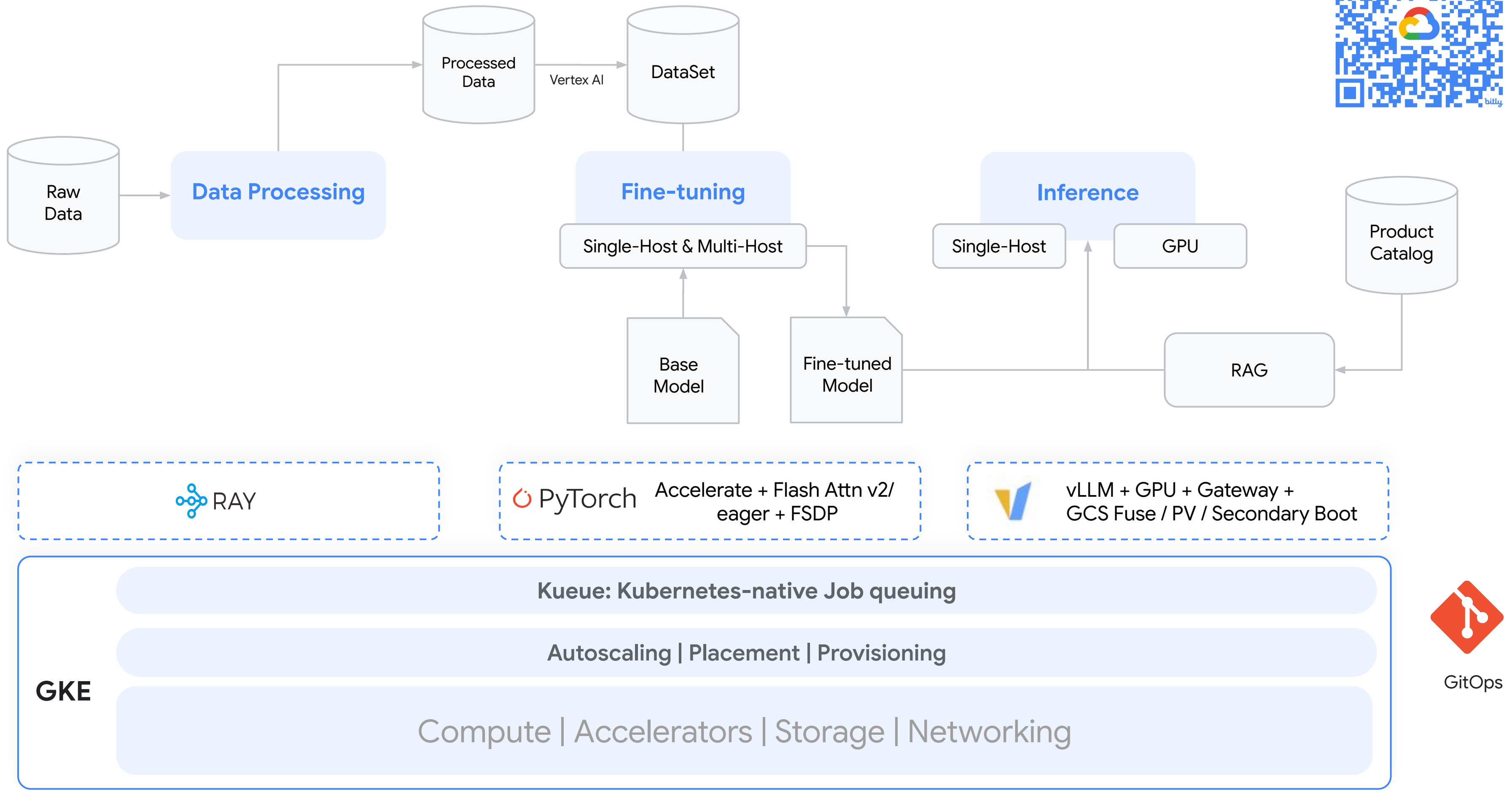
Demo

ML Platform

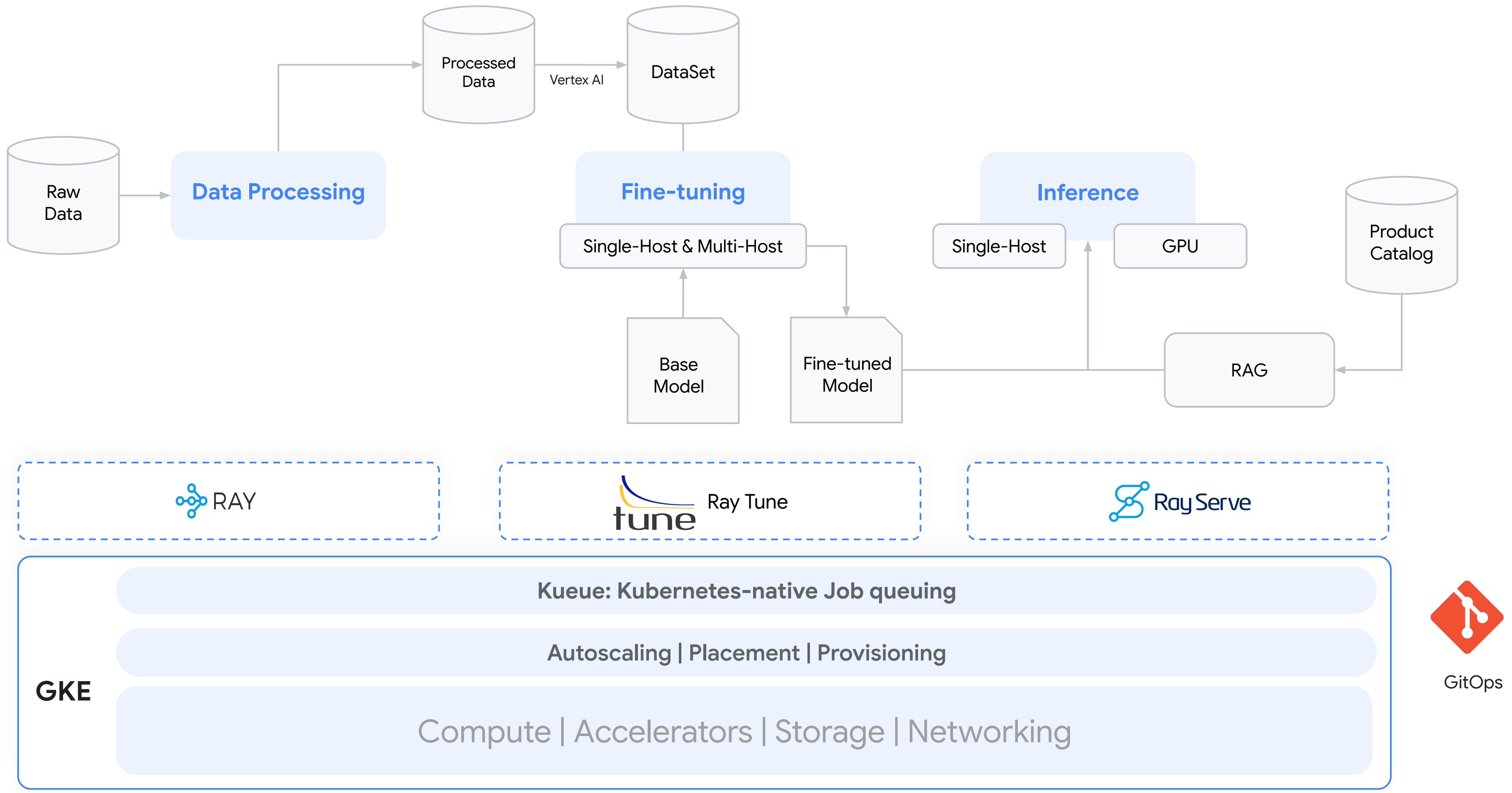
The ever growing AI/ML ecosystem



Data Preprocessing, LLM Fine-Tuning, Inference at Scale


Raw Data
Data Processing
Processed Data
Vertex AI
DataSet
Fine-tuning
Single-Host & Multi-Host
Base Model
Fine-tuned Model
Inference
Single-Host
GPU
Product Catalog
RAG
RAY
PyTorch
Accelerate + Flash Attn v2/ eager + FSDP
vLLM + GPU + Gateway + GCS Fuse / PV / Secondary Boot
Kueue: Kubernetes-native Job queuing
Autoscaling | Placement | Provisioning
GKE
Compute | Accelerators | Storage | Networking
GitOps

Data Preprocessing, LLM Fine-Tuning, Inference at Scale



Why Kubernetes?

- Infrastructure Automation
- Scalability
- High Availability and Reliability
- Advanced Device Management
- Complex, multi-cloud deployments

KubeRay APIs

RayCluster

Manage and scale Ray clusters

Ideal for prototyping / development

RayJob

Execute a Ray job with ephemeral Ray clusters

Ideal for productionizing Ray batch workloads

RayService

Deploy a Ray Serve application with zero-downtime upgrades

Ideal for inference in production

Serve LLMs with vLLM & KubeRay

Manage, configure and scale model deployments with a single API.

See the latest guide on serving LLMs with vLLM, RayServe and KubeRay.

https://docs.ray.io/en/latest/cluster/kubernetes/examples/vllm-rayservice.html

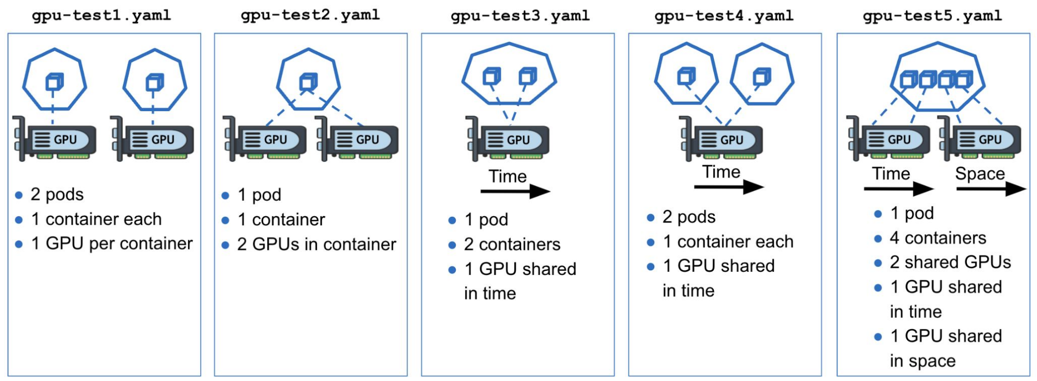
```
apiVersion: ray.io/v1
kind: RayService
metadata:
  name: llama-3-8b
spec:
  serveConfigV2: |
    applications:
    - name: llm
      route_prefix: /
      import_path:  ray-operator.config.samples.vllm.serve:model
      deployments:
      - name: VLLMDeployment
        num_replicas: 1
        ray_actor_options:
          num_cpus: 8
      runtime_env:
        working_dir:
"https://github.com/ray-project/kuberay/archive/master.zip"
        pip: ["vllm==0.5.4"]
        env_vars:
          MODEL_ID: "meta-llama/Meta-Llama-3-8B-Instruct"
          TENSOR_PARALLELISM: "2"
          PIPELINE_PARALLELISM: "1"
```

Demo

DRA: Optimizing Resource Allocation

DRA enhances the Kubernetes scheduler with awareness of Ray's needs and the dynamic nature of certain workloads:

- Optimized resource utilization
- Improved cluster efficiency

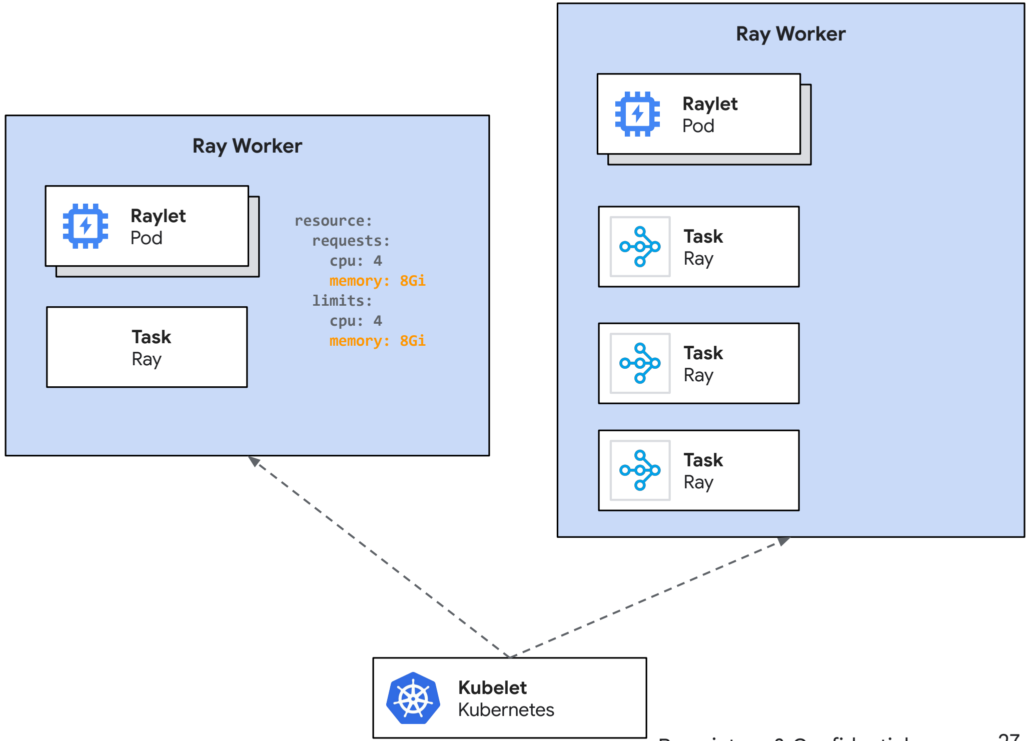


Kubernetes v1.31 introduced new Device Resource Assignment (DRA) APIs.

In-place VPA: Minimizing Disruptions

In-place Vertical Pod Autoscaling enables elastic memory consumption for Ray containers without requiring restarts.

Prevent performance degradation and risk of OOM-kill with resizable Pod memory



```
resource:
  requests:
    cpu: 4
    memory: 8Gi
  limits:
    cpu: 4
    memory: 8Gi
```

Feedback 🙏

Thank you

Google Cloud