Tech Stack

fine-tune (SFT / RL) and serve LLM model on cloud and at scale.

Figure 1: overview of tech stack

I. Prerequisites

Docker
definition

- a system-agnostic encapsulation of code and dependencies
- image: contains library, dependence, and application code
- container: a runnable instance of image

steps

1. build image: docker build ...
2. push to registry: docker image push ...
3. pull to (K8s) application

Kubernetes (K8s)
definition

- physical orchestration of compute and memory
- hierarchy: cluster -> node -> pod -> container

steps

1. create config (yaml) to request resources
2. submit request: kubectl apply -f config.yaml

Ray (cluster)
definition

- virtual orchestration of job and process given resources

components

- dashboard: for job monitoring and logging

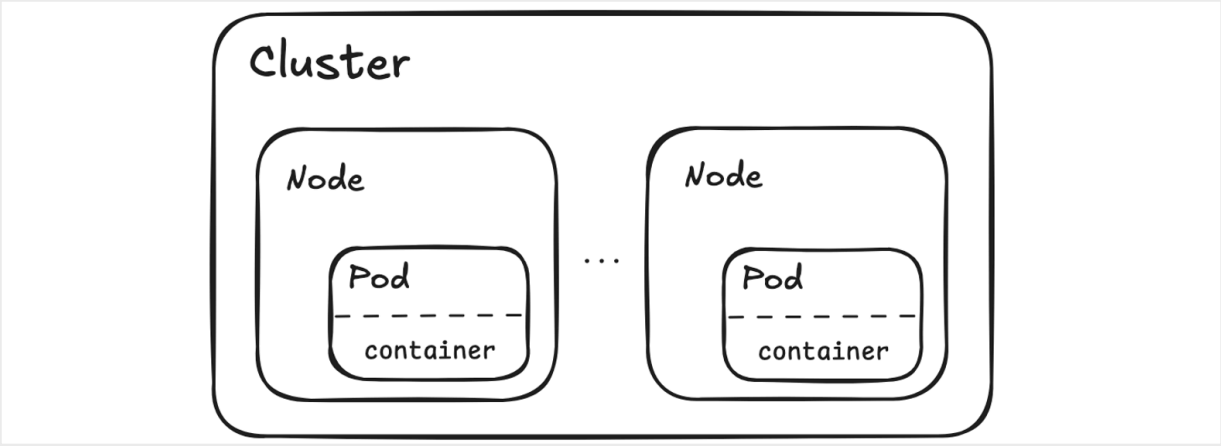

Figure 2: Hierarchy of (cloud) cluster

II. Implementation

"De-facto stack"

- docker (image): same base image and packages
- K8s:
  - RayCluster: orchestrate GPUs, storage, and protocols
  - RayJob: submit fine-tune tasks
- framework: {torch, verl, ray} for fine-tune, vllm for serving

	Docker	Kubernetes	Framework
SFT	FROM base image RUN packages ENV variables ⋮	StorageClass RayCluster Service RayJob PVC Deployment Service	torch / verl
RL (GRPO)			verl / ray
Serving			vllm

Table 1: tech stack x tasks (fine-tune, serve)

Cloud resources

The essential resources and the corresponding components on AWS.

resource	AWS
compute	EC2 instance
memory	S3 / FSx
network	Virtual Private Cloud (VPC)
K8s	Elastic K8s Services (EKS)

Table 2: AWS components

*** Reference**

- https://docker-curriculum.com
- https://kubernetes.io/docs/concepts
- https://github.com/volcengine/verl/tree/main/examples/ray