KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon | CloudNativeCon

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

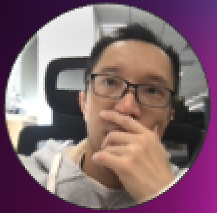
China 2024

Towards an Unified Cloud-Native Infra for AI & HPC

TACC - A Five-Year Journey

Kaiqiang Xu: HKUST

Peter Pan : DaoCloud



TACC: Turing AI Computing Cloud

About Us

Kaiqiang Xu

- **PhD Researcher in Computer Systems for Machine Learning**

- **Hong Kong University of Science and Technology**

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Peter Pan

- **R&D Engineer Lead**
- **Open Source Advocate**

- **DaoCloud**

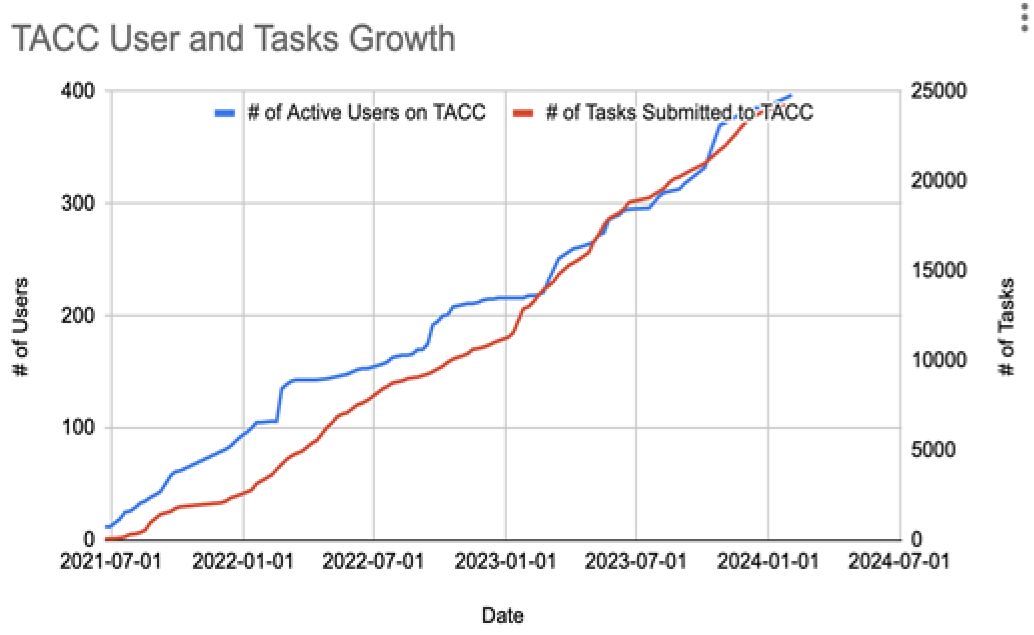
DaoCloud

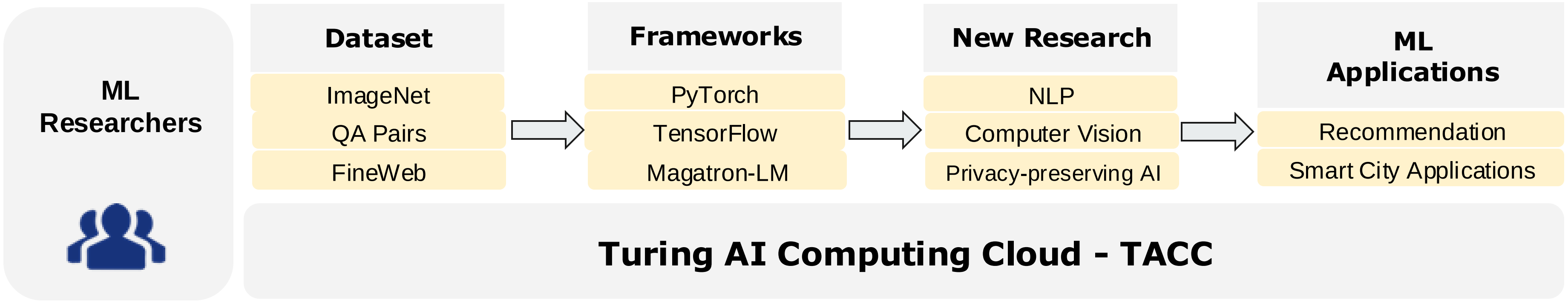
TACC: Intro

High-performance and highly scalable AI computing infrastructure

TACC is an AI computing infrastructure designed for machine learning applications, supporting and accelerating the constantly evolving research in machine learning at both the software and hardware levels.

Due to system-level optimizations specifically targeted for ML/DL programs, TACC outperforms traditional HPC computing clusters in terms of both performance and stability.


TACC User and Tasks Growth


ML Researchers

Dataset: ImageNet, QA Pairs, FineWeb

Frameworks: PyTorch, TensorFlow, Magatron-LM

New Research: NLP, Computer Vision, Privacy-preserving AI

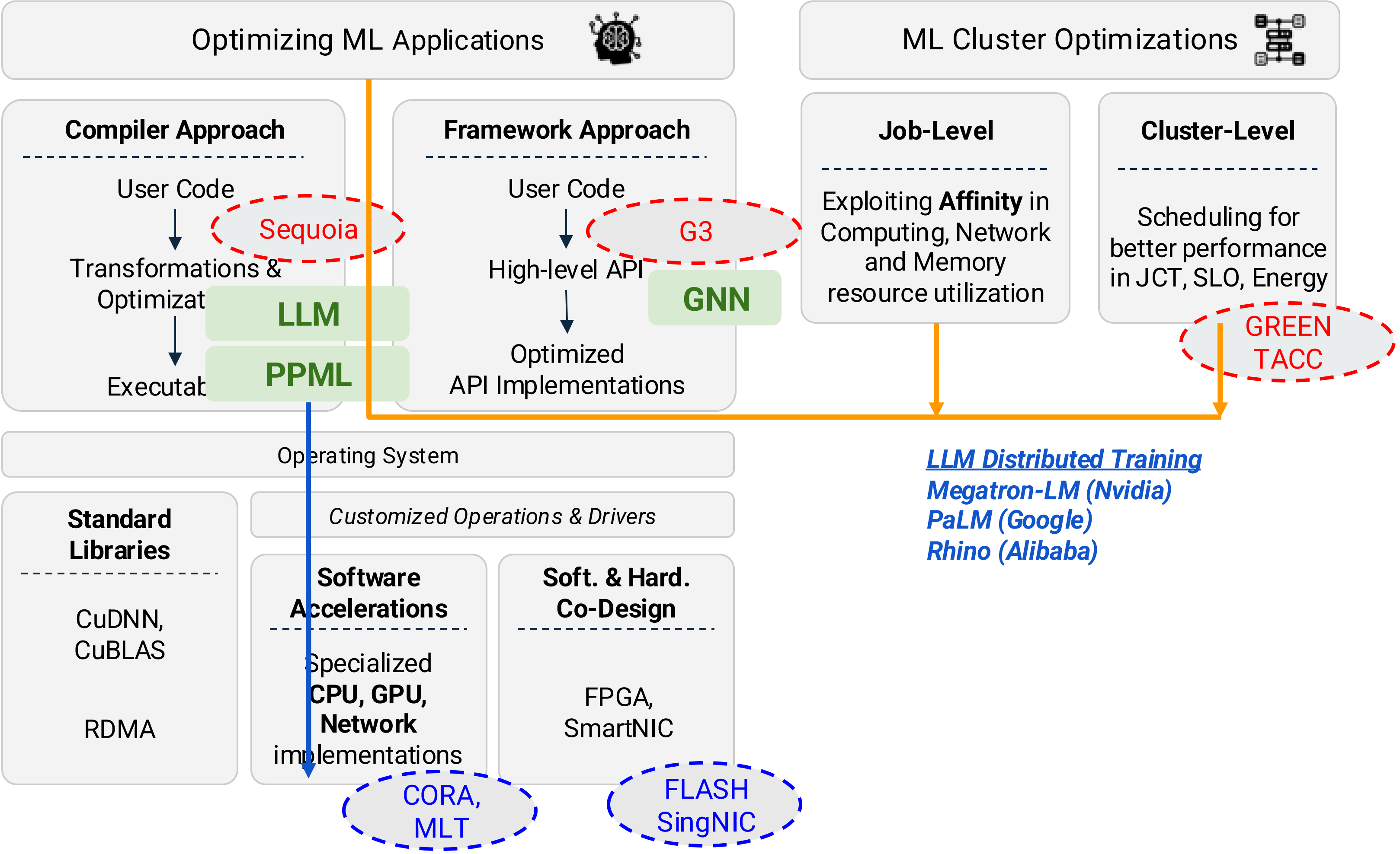
ML Applications: Recommendation, Smart City Applications

Turing AI Computing Cloud - TACC

TACC: Intro

Underpinning Research

- **ML frameworks** enhance model development through advanced parallelization and distributed training, efficiently handling complex computations and large datasets.
- **Cluster resource scheduler** optimizes cluster-wide resource allocation across AI tasks, boosting overall job throughput and other efficiency factors in AI clusters.
- **AI-centric networking** improves data flow and reduces latency by efficiently managing large model transport and using FPGAs for compute offloading



Optimizing ML Applications
ML Cluster Optimizations
Compiler Approach
User Code
Sequoia
Transformations & Optimizat
LLM
PPML
Executab
Framework Approach
User Code
G3
High-level API
GNN
Optimized API Implementations
Job-Level
Exploiting Affinity in Computing, Network and Memory resource utilization
Cluster-Level
Scheduling for better performance in JCT, SLO, Energy
GREEN TACC
Operating System
Standard Libraries
CuDNN, CuBLAS
RDMA
Customized Operations & Drivers
Software Accelerations
Specialized CPU, GPU, Network implementations
CORA, MLT
Soft. & Hard. Co-Design
FPGA, SmartNIC
FLASH SingNIC

LLM Distributed Training
Megatron-LM (Nvidia)
PaLM (Google)
Rhino (Alibaba)

TACC: Intro

Tutorial

- https://github.com/turingaicloud/quickstart

Core Code

- https://github.com/turingaicloud/tcloud-sdk

TACC: Streamline Scientists‘ daily

Interface

- Scientist are familiar with slurm-like CLI
- Strightforward CLI & Config (tuxiv.config)

```
# tcloud submit                # submit job to remote cluster

# tcloud ps                    # show job status

# tcloud upload                # upload to remote working dir
```

tuxiv.config

```
entrypoint:
 - python mnist.py ..
environment:
    channels:
     - nvidia
    dependencies:
     - pytorch=1.9.0
     - torchvision=0.10.0
     - cudatoolkit=11.1.74
datasets:
   -  $name
job:
    name: test
    general:
     - nodes=2
     - cpus-per-task=10
     - gres=gpu:2
```

entrypoint

Python packages

dataset source

Slurm Params

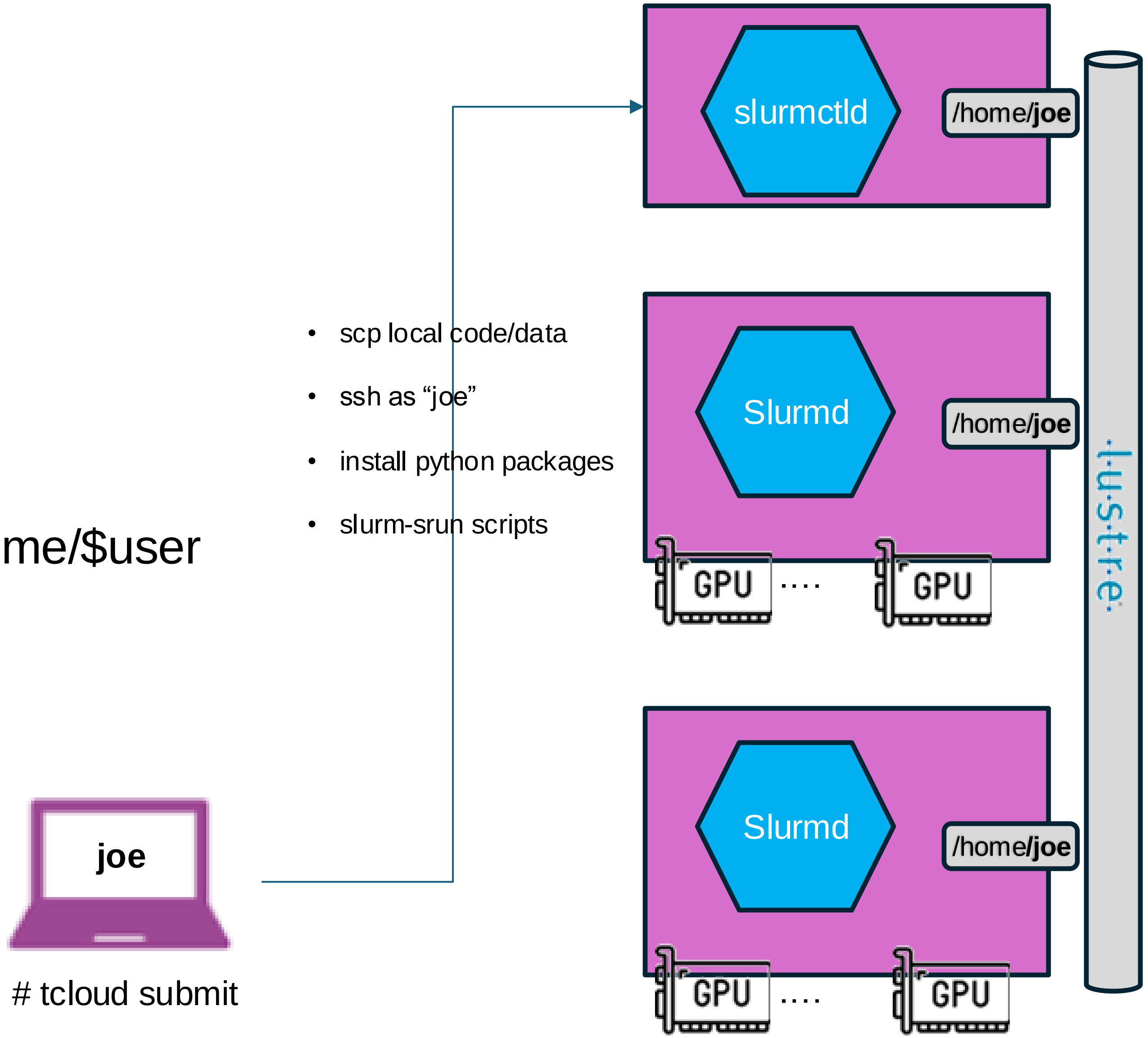
TACC: Streamline Scientists‘ daily

Remote Execution

- SCP to copy local code to user folder
- SSH to run script remotely
- Slurm agent in docker

Tenant & User

- Re-use Linux user system in servers
- Mount distributed storage to each /home/$user



- scp local code/data
- ssh as “joe”
- install python packages
- slurm-srun scripts

# tcloud submit

TACC: Streamline Scientists‘ daily

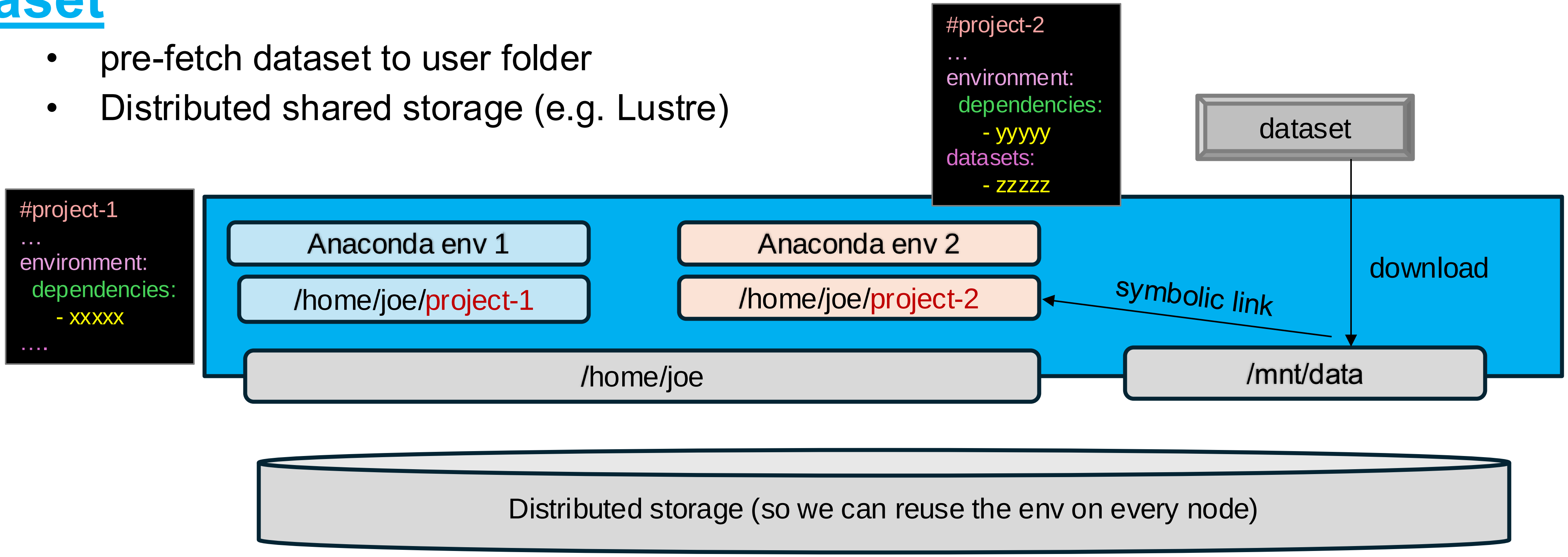
Python Env

⚠ “venv + pip install” every time is time-consuming !

- Individual Conda Env for each user
- Persistent & re-use env and packages in user folder

Dataset

- pre-fetch dataset to user folder
- Distributed shared storage (e.g. Lustre)

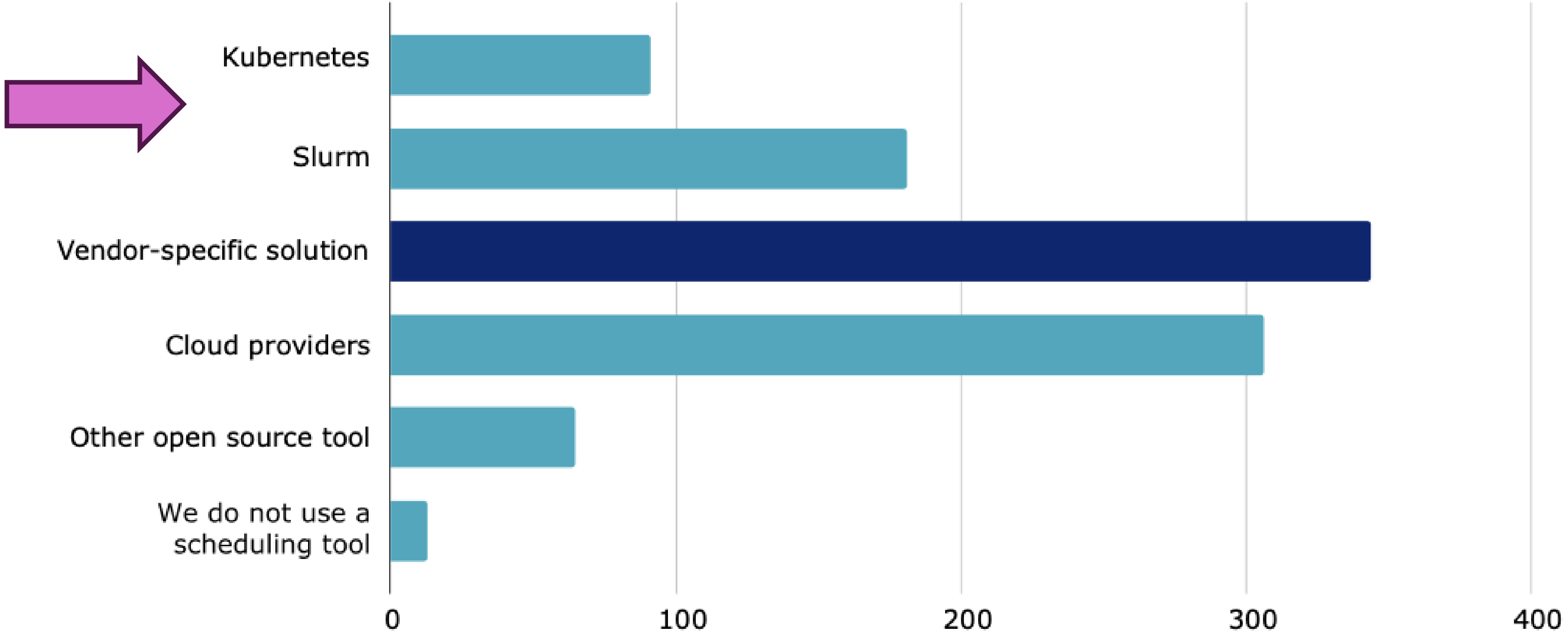


```
#project-1
…
environment:
  dependencies:
    - xxxxx
….
```

```
#project-2
…
environment:
  dependencies:
    - yyyyy
datasets:
    - zzzzz
```

Kubernetes - Slurm

Survey:
Which resource-scheduling/job-management tool do you use as part of your AI/ML tech stack?



https://ai-infrastructure.org/wp-content/uploads/2024/03/The-State-of-AI-Infrastructure-at-Scale-2024.pdf

TACC powered by K8S

Kubernetes v.s. Slurm

slurm workload manager

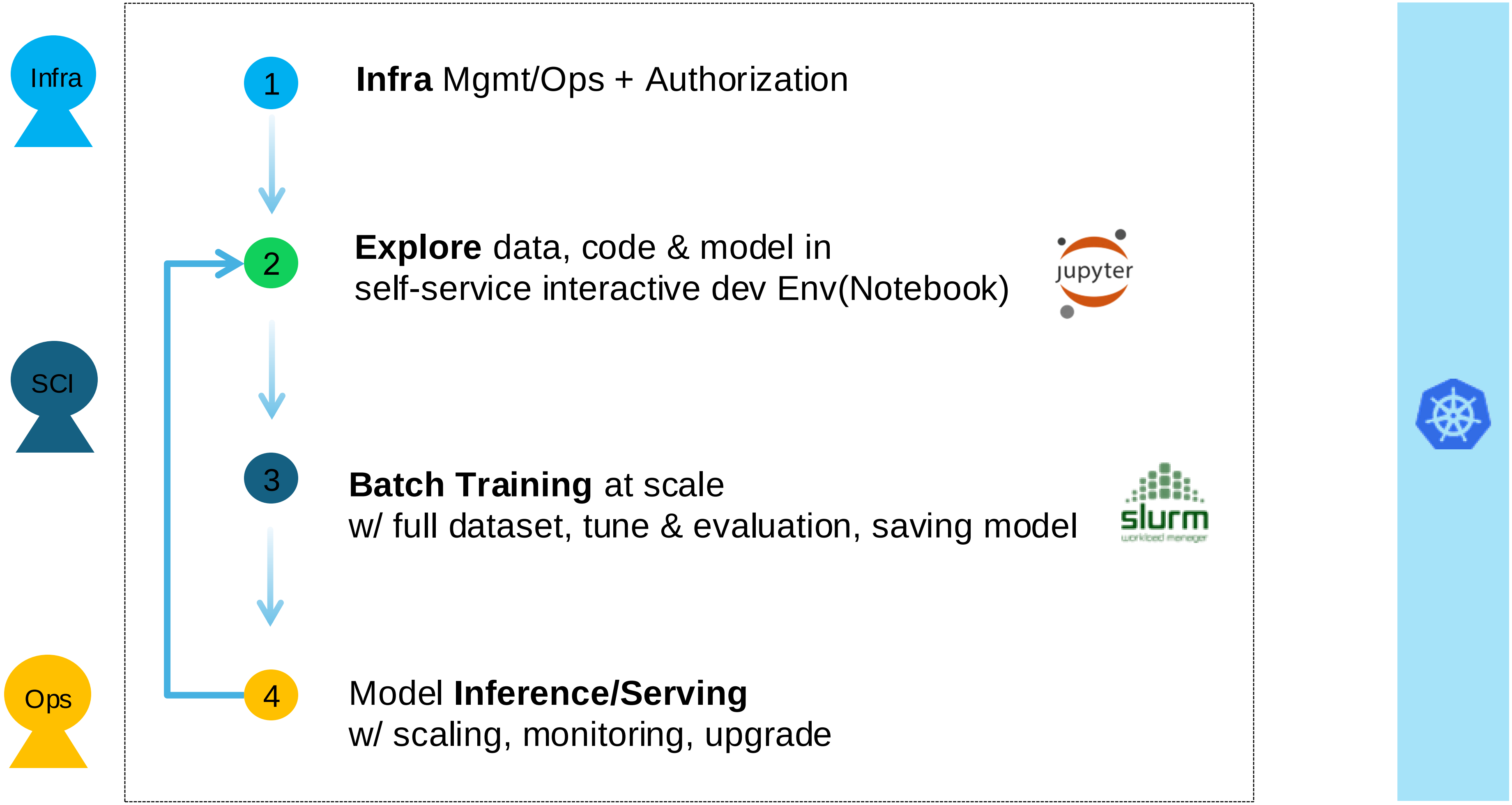
Kubernetes

- 10 years, still young
- not initially designed for ML workload, but scheduling capability keep evolving
- declarative (final-state-driven)
- docker build burden
- flexible and scalable
- both for training & serving

Slurm

- 22 years old, mature for HPC
- Born for batch workload: Queue, Job, Scheduling
- Stability issue
- Just plain shell and files
- lack of auto-scaling
- OCI image limited supported

TACC: Streamline Scientists‘ daily



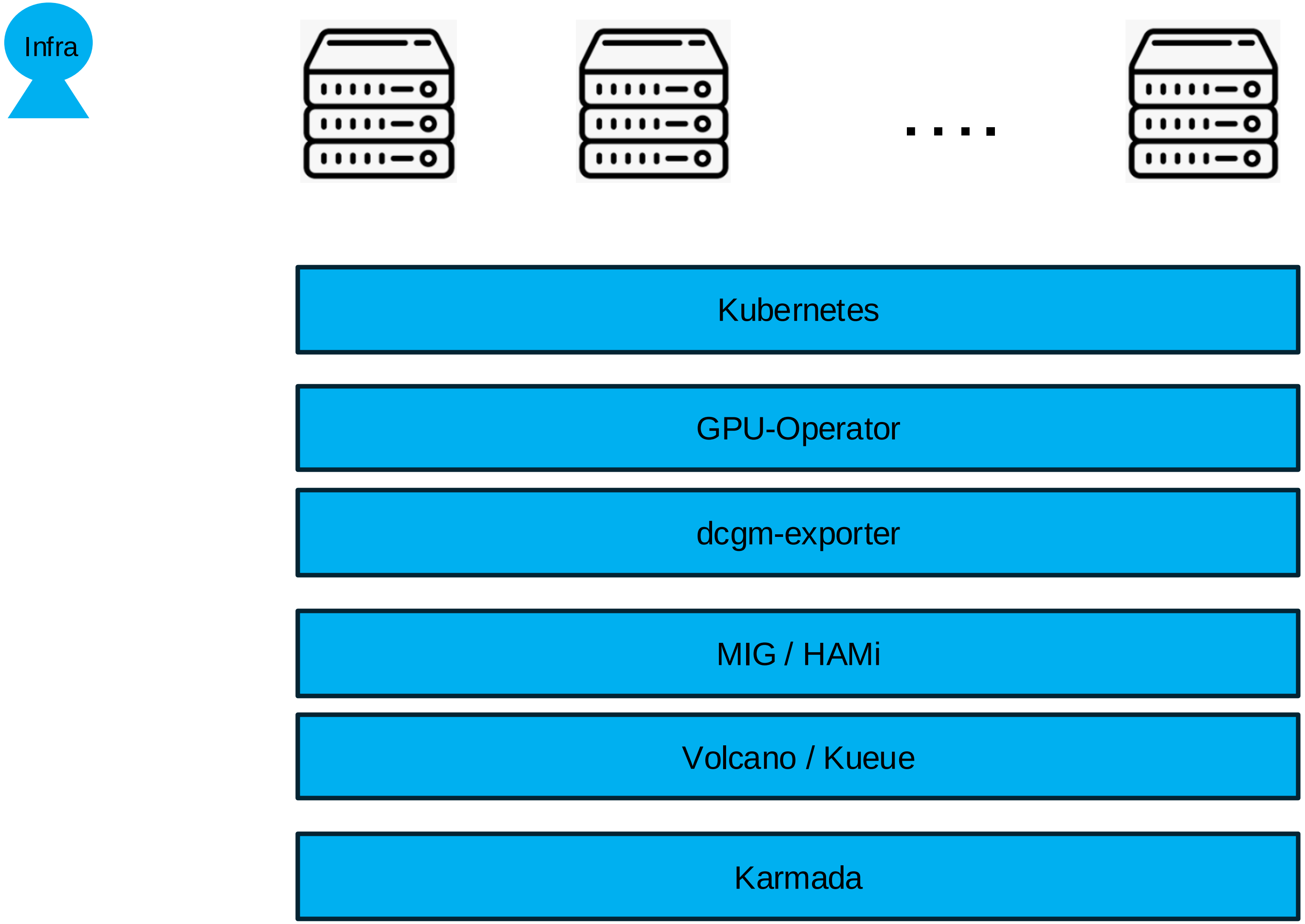
1 **Infra** Mgmt/Ops + Authorization

2 **Explore** data, code & model in
self-service interactive dev Env(Notebook)

3 **Batch Training** at scale
w/ full dataset, tune & evaluation, saving model

4 Model **Inference/Serving**
w/ scaling, monitoring, upgrade

How K8S helps Infra. Engineer



Infra

. . . .

Kubernetes — All machines in a Resource Pool

GPU-Operator — No worry about driver/plugin/runtime.. for each node

dcgm-exporter — GPU monitoring

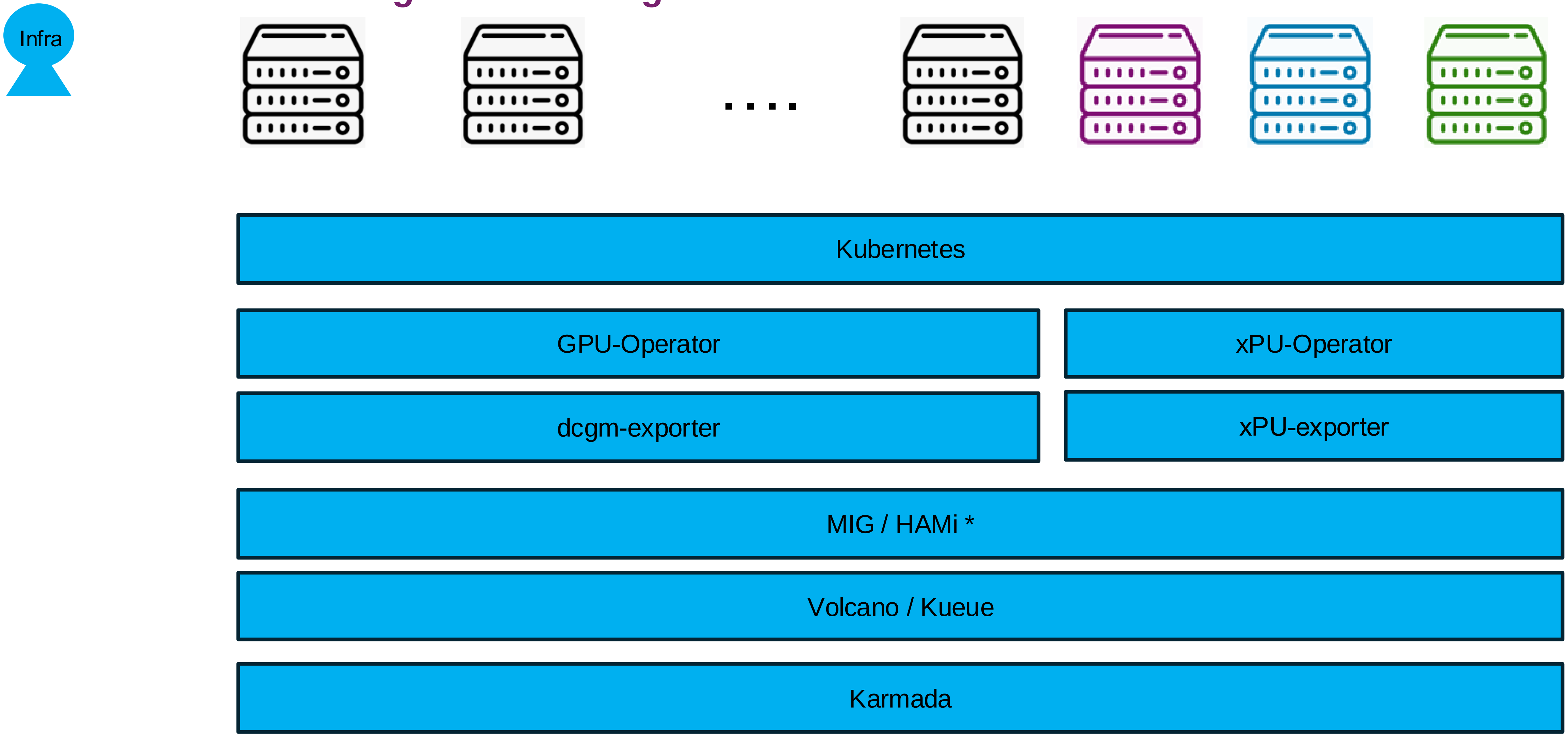
MIG / HAMi — virtual GPU (partition)

Volcano / Kueue — Job & Queue Scheduling

Karmada — Multiple Cloud/Cluster

How K8S helps Infra. Engineer

When introducing other Heterogeneous xPUs



Kubernetes

GPU-Operator

xPU-Operator

dcgm-exporter

xPU-exporter

MIG / HAMi *

Volcano / Kueue

Karmada

How K8S helps Scientist

SCI

- Multi-Tenant
- Dev & Train
- Scheduling
- Monitor
- Performance & Robustness

- k8s namespace for each $user
- Namespace Quota
- Queue Quota (Kueue concept)

```
# kubectl -n joe get resourcequotas  -o yaml

apiVersion: v1
kind: ResourceQuota
spec:
  hard:
    requests.nvidia.com/mig-1g.6gb: "2”
    requests.storage: 1000Gi
    limits.memory: 128Gi
```

```
# kubectl  get clusterqueue default -o yaml

apiVersion: kueue.x-k8s.io/v1beta1
kind: ClusterQueue
spec:
flavors:
    - resources:
      - name: cpu
        nominalQuota: "200"
      - name: memory
        nominalQuota: 1280Gi
      - name: nvidia.com/gpu
        nominalQuota: "10"
      - name: rdma/hca_shared_devices_a
        nominalQuota: "100"
```

How K8S helps Scientist

SCI

- Multi-Tenant
- Dev & Train
- Scheduling
- Monitor
- Performance & Robustness

code

Dataset / Modeling

Python Packages

Paints

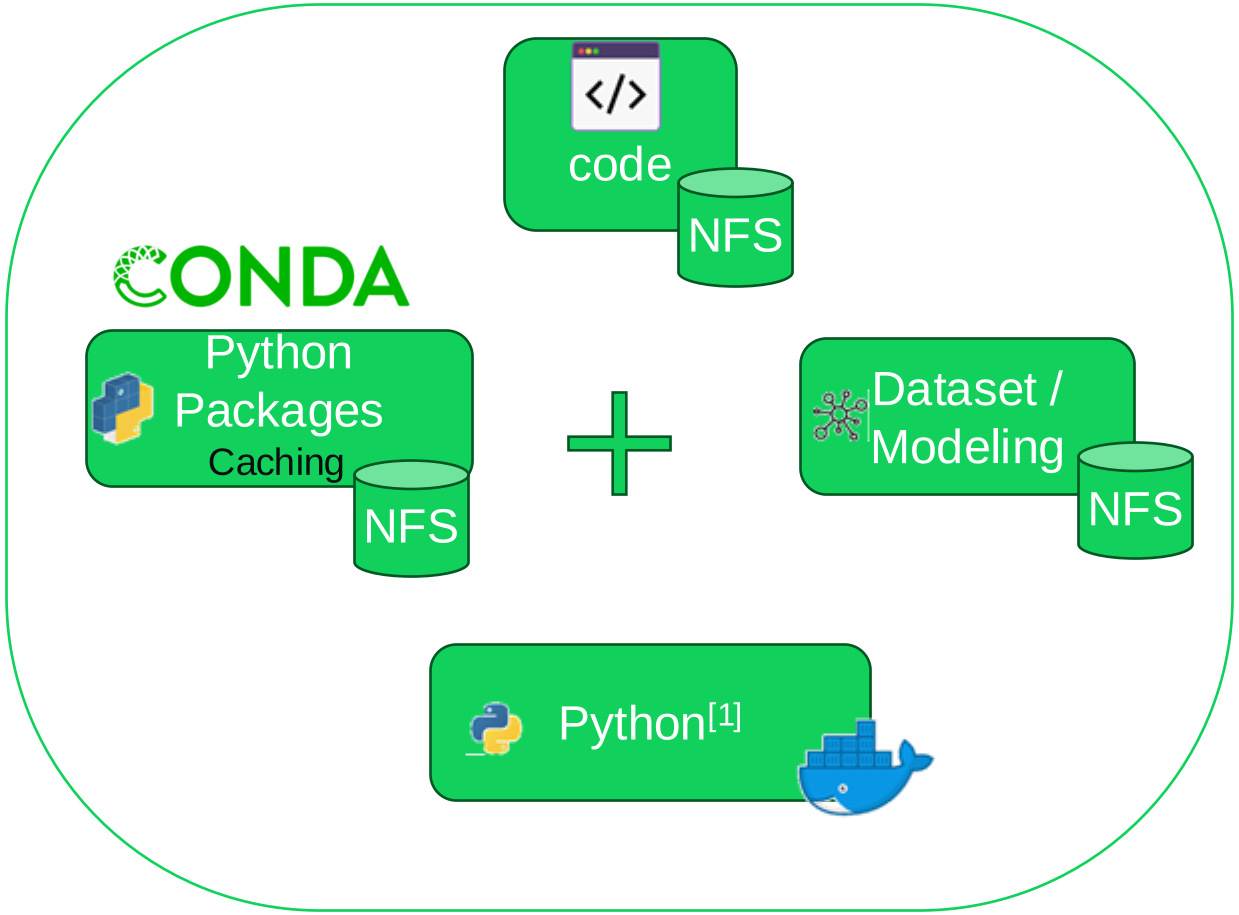
- Should learn about “docker build”
- New image when any tiny change
- Huge image size

```
FROM python

RUN pip install torch==2.4.0  torchvision==0.19.0 transformers==4.4.1 datasets==2.21

RUN  git clone $code

RUN  wget  $dataset / $model
```

How K8S helps Scientist

SCI

- Multi-Tenant
- Dev & Train
- Scheduling
- Monitor
- Performance & Robustness



- Reusable conda/pip environment
- Data Isolated by namespaced PVC
- No worry about building ”docker image”

1. **Code**: can be either managed by Git then pre-fetched by init-container, or rsync to NFS shares

2. **Data/Model**: can either download from S3 and pre-fetch by init-container

3. **Python Packages**: with conda capability, pre-install into NFS shares and can be re-used

4. **Docker image**: can be very flexible. either include all required pip packages & cuda inside, or a just a bare image all relies on conda volume.

[1] Base Image: python itself can also be hosted in conda env.
But in practice in LLM age, Pytorch + CUDA

How K8S helps Scientist

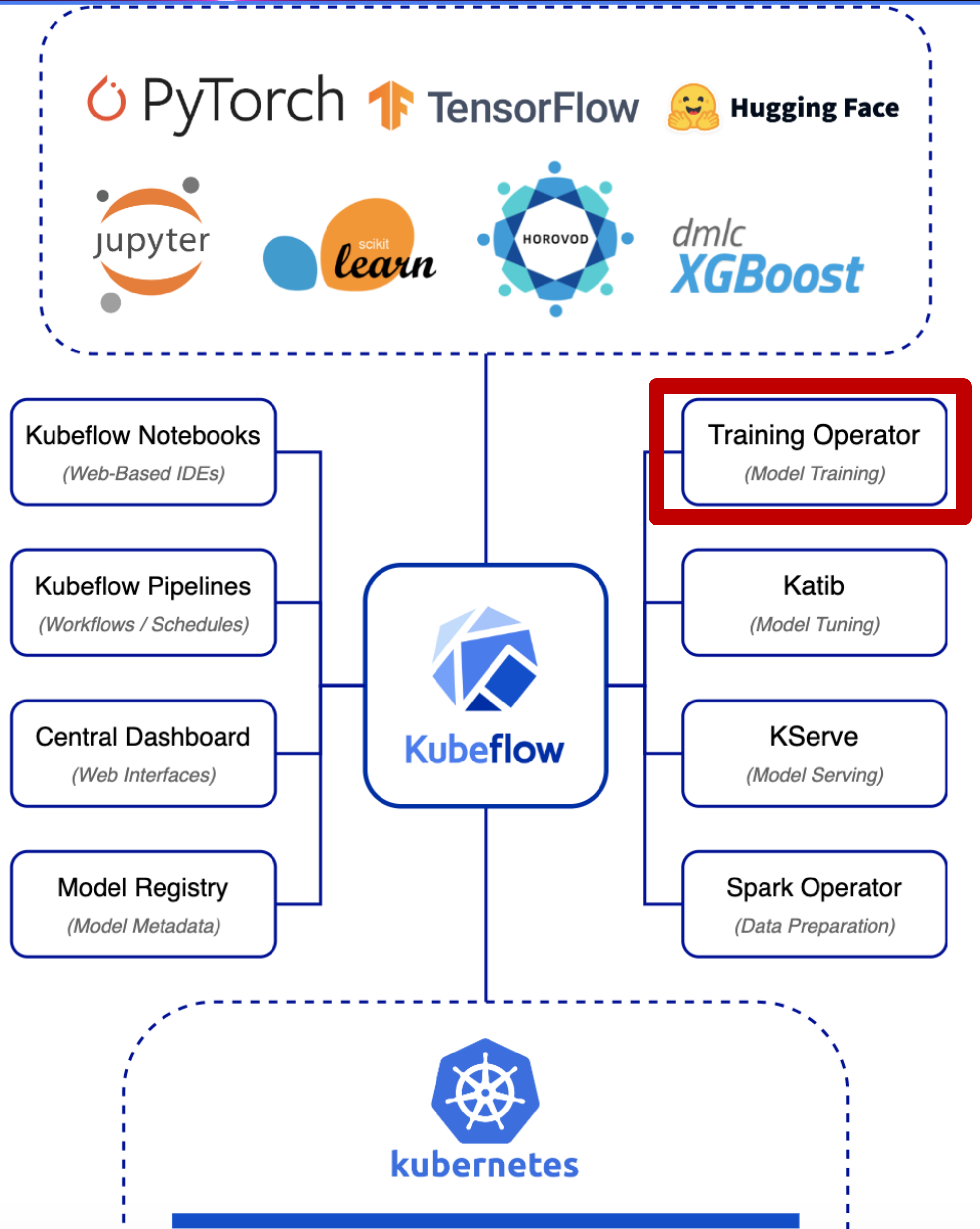
SCI

- Multi-Tenant
- Dev & Train
- Scheduling
- Monitor
- Performance & Robustness

distributed training on K8s

Kubeflow’s training-operator

Features
Distributed Training (e.g. PyTorchJob)
All-Reduce Style Training with MPI
High Performance Computing (HPC) with MPI
Job Scheduling with Volcano, Kueue
Elastic Training


PyTorch
TensorFlow
Hugging Face
jupyter
scikit learn
HOROVOD
dmlc XGBoost
Kubeflow Notebooks (Web-Based IDEs)
Kubeflow Pipelines (Workflows / Schedules)
Central Dashboard (Web Interfaces)
Model Registry (Model Metadata)
Kubeflow
Training Operator (Model Training)
Katib (Model Tuning)
KServe (Model Serving)
Spark Operator (Data Preparation)
kubernetes

Web UI

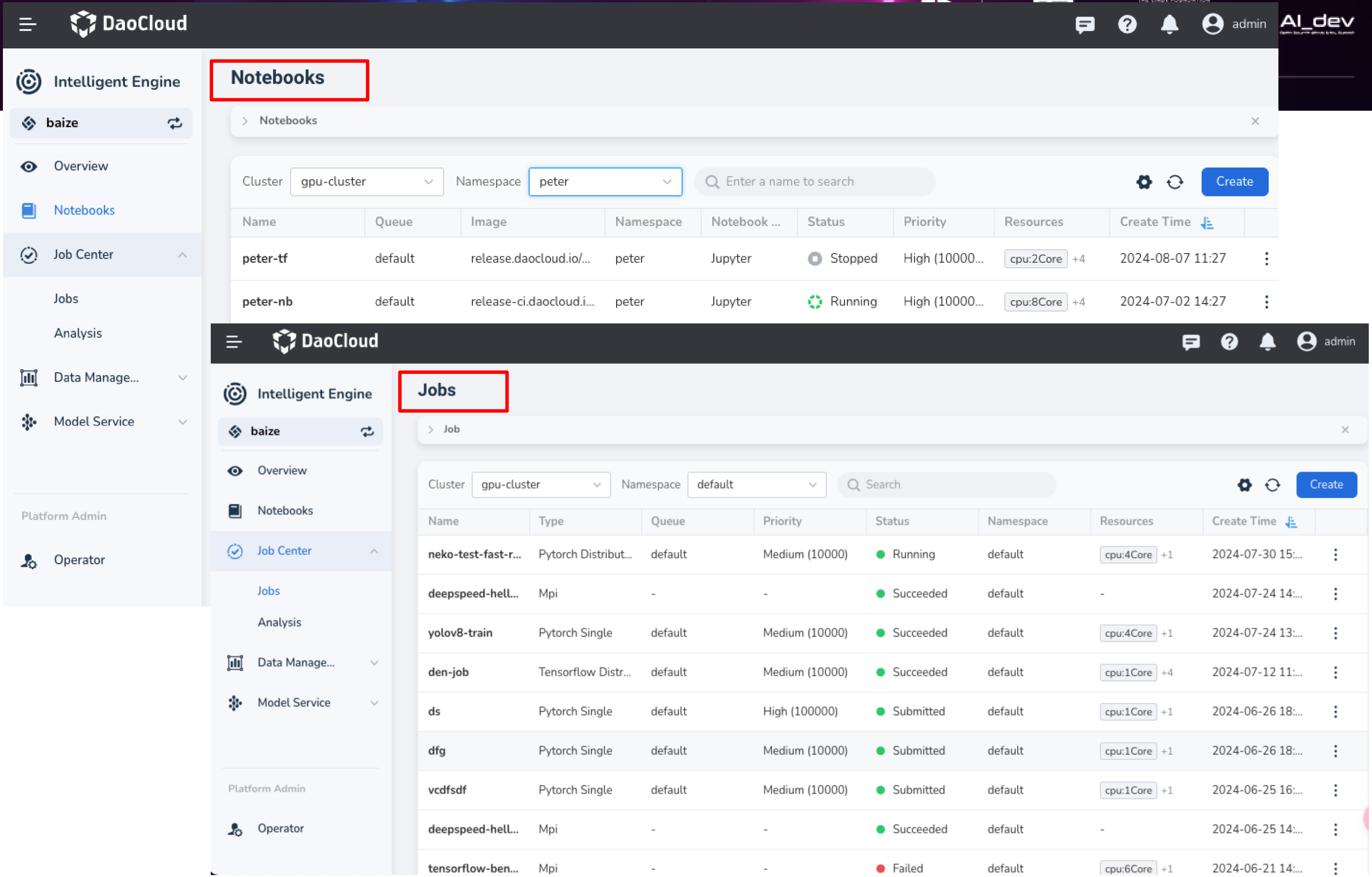
Multi-Tenant

Dev & Train

Scheduling

Monitor

Performance & Robustness



Notebooks

Notebooks

Cluster: gpu-cluster | Namespace: peter | Enter a name to search | Create

Name	Queue	Image	Namespace	Notebook ...	Status	Priority	Resources	Create Time
peter-tf	default	release.daocloud.io/...	peter	Jupyter	Stopped	High (10000...	cpu:2Core +4	2024-08-07 11:27
peter-nb	default	release-ci.daocloud.i...	peter	Jupyter	Running	High (10000...	cpu:8Core +4	2024-07-02 14:27

Jobs

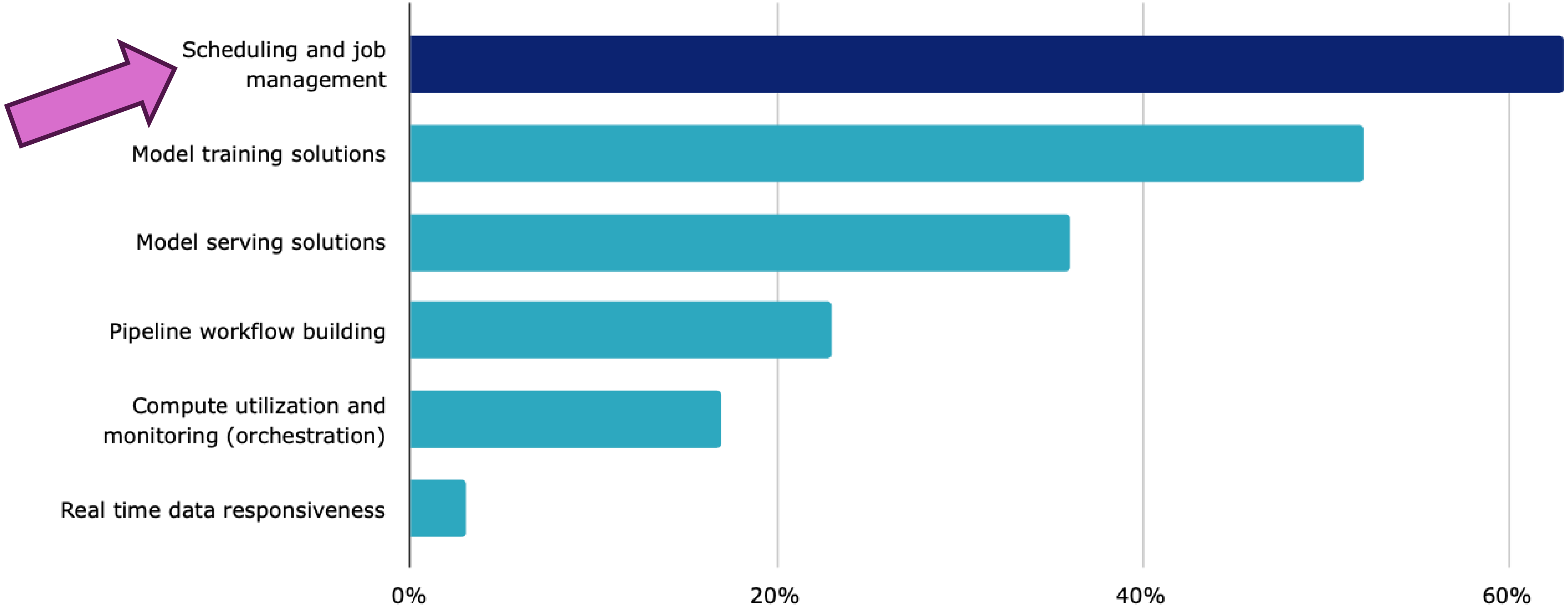
Job

Cluster: gpu-cluster | Namespace: default | Search | Create

Name	Type	Queue	Priority	Status	Namespace	Resources	Create Time
neko-test-fast-r...	Pytorch Distribut...	default	Medium (10000)	Running	default	cpu:4Core +1	2024-07-30 15:...
deepspeed-hell...	Mpi	-	-	Succeeded	default	-	2024-07-24 14:...
yolov8-train	Pytorch Single	default	Medium (10000)	Succeeded	default	cpu:4Core +1	2024-07-24 13:...
den-job	Tensorflow Distr...	default	Medium (10000)	Succeeded	default	cpu:1Core +4	2024-07-12 11:...
ds	Pytorch Single	default	High (100000)	Submitted	default	cpu:1Core +1	2024-06-26 18:...
dfg	Pytorch Single	default	Medium (10000)	Submitted	default	cpu:1Core +1	2024-06-26 18:...
vcdfsdf	Pytorch Single	default	Medium (10000)	Submitted	default	cpu:1Core +1	2024-06-25 16:...
deepspeed-hell...	Mpi	-	-	Succeeded	default	-	2024-06-25 14:...
tensorflow-ben...	Mpi	-	-	Failed	default	cpu:6Core +1	2024-06-21 14:...

Scheduling is key demand

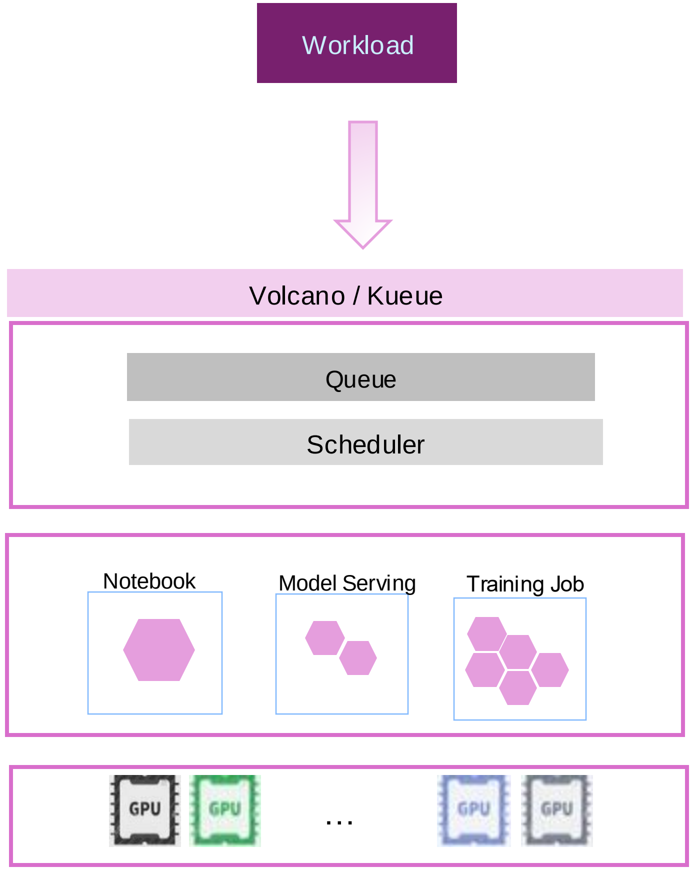
Q: What types of solutions does your organization currently lack in your AI/ML tech stack?



https://ai-infrastructure.org/wp-content/uploads/2024/03/The-State-of-AI-Infrastructure-at-Scale-2024.pdf

How K8S helps Scientist

- Multi-Tenant
- Code & Data
- Scheduling
- Monitor
- Performance & Robustness



Various Scheduling Policy supported

Gang

- avoid waste/ deadlock
- Big jobs starve smalls

Preempt

- Diverse SLO

Binpack

- Avoid fragments
- Small jobs starve biggers

Affinity

- inner-node performance first

How K8S helps Scientist

SCI

Multi-Tenant

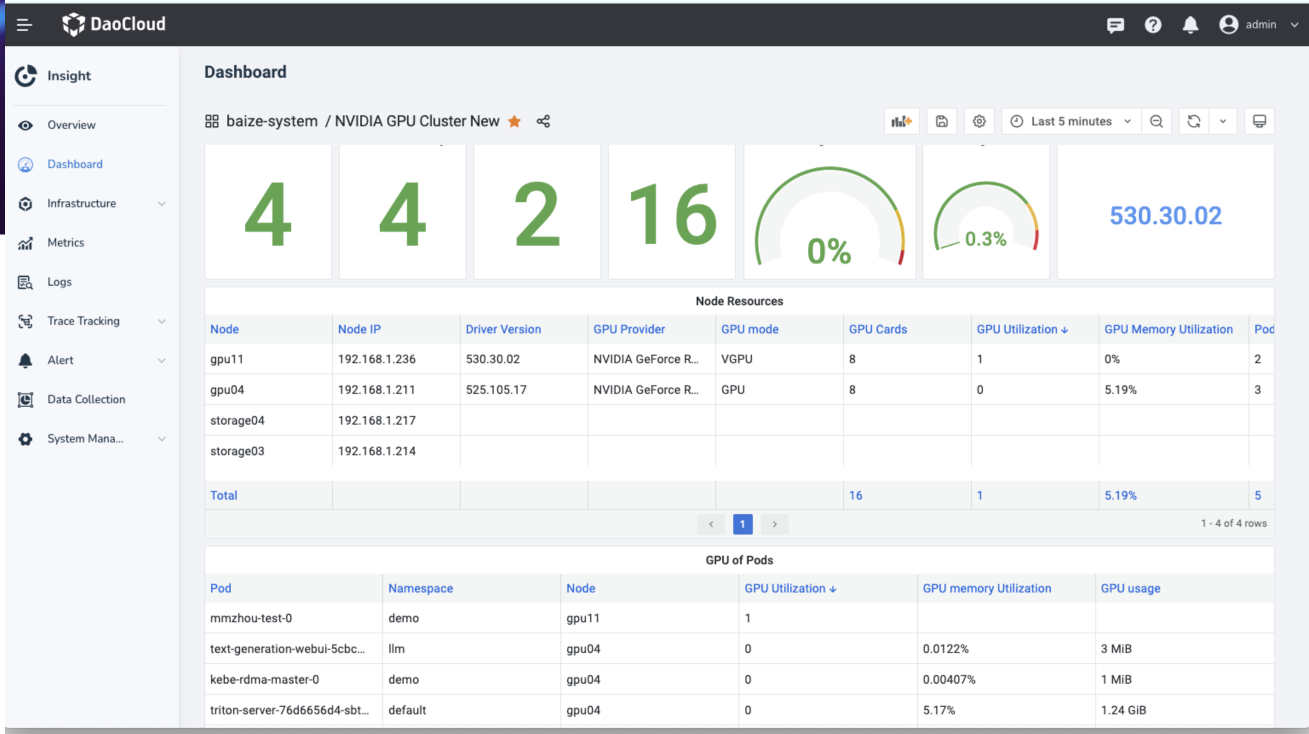
Code & Data

Scheduling

Monitor

Performance & Robustness

- GPU Inventory
- GPU Healthy
- GPU Utilization *
- Inference Queue/Latency
- Billing for tenants
- Training Failure
- Alerting



Dashboard

baize-system / NVIDIA GPU Cluster New

4 4 2 16 0% 0.3% 530.30.02

Node Resources

Node	Node IP	Driver Version	GPU Provider	GPU mode	GPU Cards	GPU Utilization ↓	GPU Memory Utilization	Pod
gpu11	192.168.1.236	530.30.02	NVIDIA GeForce R...	VGPU	8	1	0%	2
gpu04	192.168.1.211	525.105.17	NVIDIA GeForce R...	GPU	8	0	5.19%	3
storage04	192.168.1.217							
storage03	192.168.1.214							
Total					16	1	5.19%	5

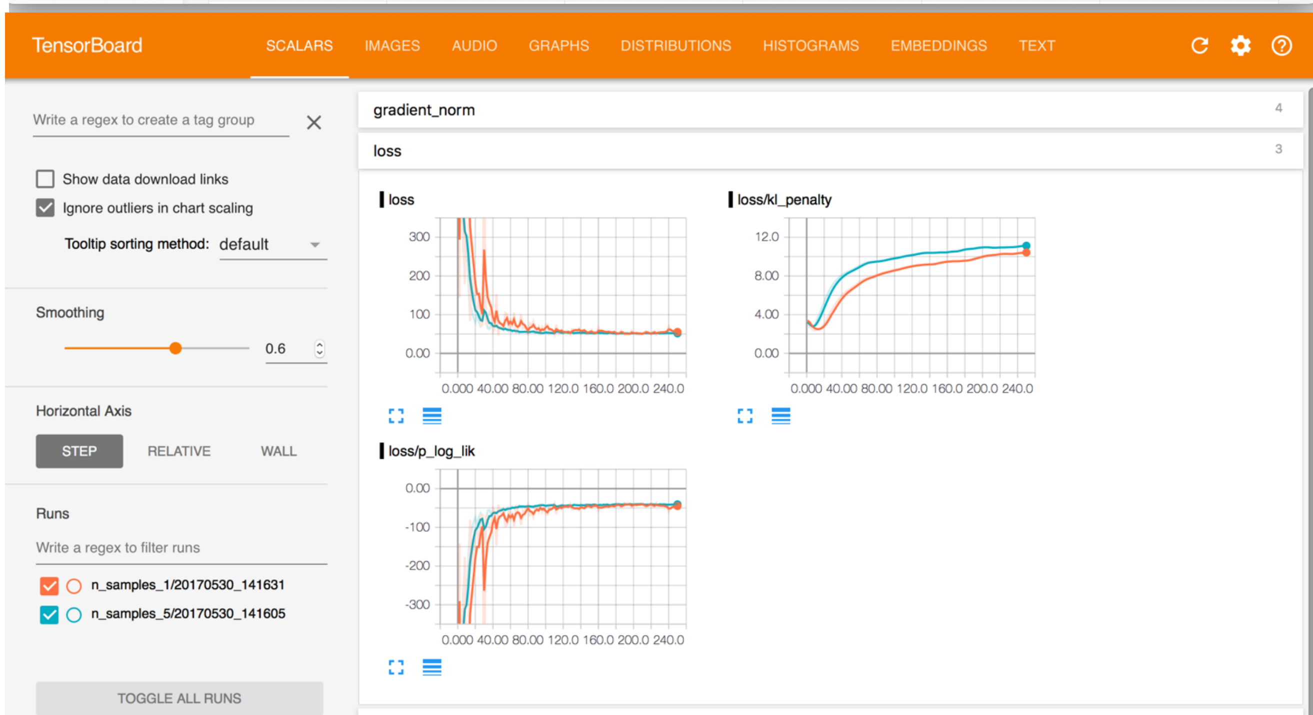
1 - 4 of 4 rows

GPU of Pods

Pod	Namespace	Node	GPU Utilization ↓	GPU memory Utilization	GPU usage
mmzhou-test-0	demo	gpu11	1		
text-generation-webui-5cbc...	llm	gpu04	0	0.0122%	3 MiB
kebe-rdma-master-0	demo	gpu04	0	0.00407%	1 MiB
triton-server-76d6656d4-sbt...	default	gpu04	0	5.17%	1.24 GiB

- TensorBoard

# arena submit xxx --tensorboard



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

Runs

☑ n_samples_1/20170530_141631

☑ n_samples_5/20170530_141605

How K8S helps Scientist

SCI

Multi-Tenant

Code & Data

Scheduling

Monitor

Performance & Robustness

Bottom necks

- Dataset loading --- caching
- Model checkpoint saving --- disk I/O bandwidth
- Network interconnect --- RDMA full utilized + best topology
- Intermedia failure --- waste of GPU time

How K8S helps Scientist

SCI

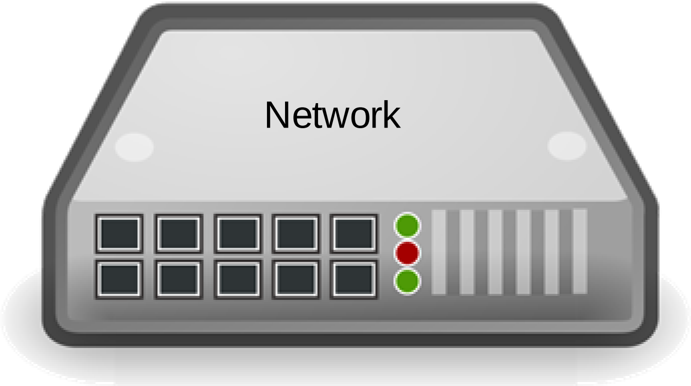
- Multi-Tenant
- Code & Data
- Scheduling
- Monitor
- Performance & Robustness

- **Distributed Storage**
  - Lusture
  - BeeGFS
  - MinIO
- **Caching**
  - JuiceFS
  - Alluxio
- **Local Storage**
  - HwameiStor

- **RoCE**
- **InfiniBand**

SpiderPool

- SRIOV-CNI
- GPU-Direct-RDMA
- GPU-NIC Topology Scheduling



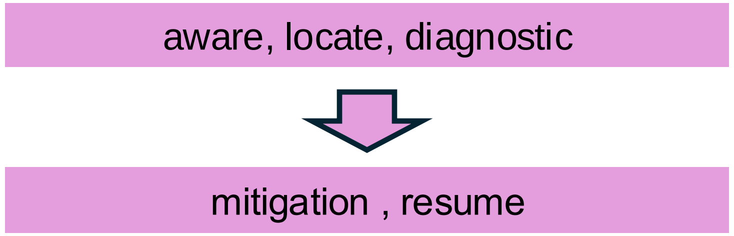
How K8S helps Scientist

SCI

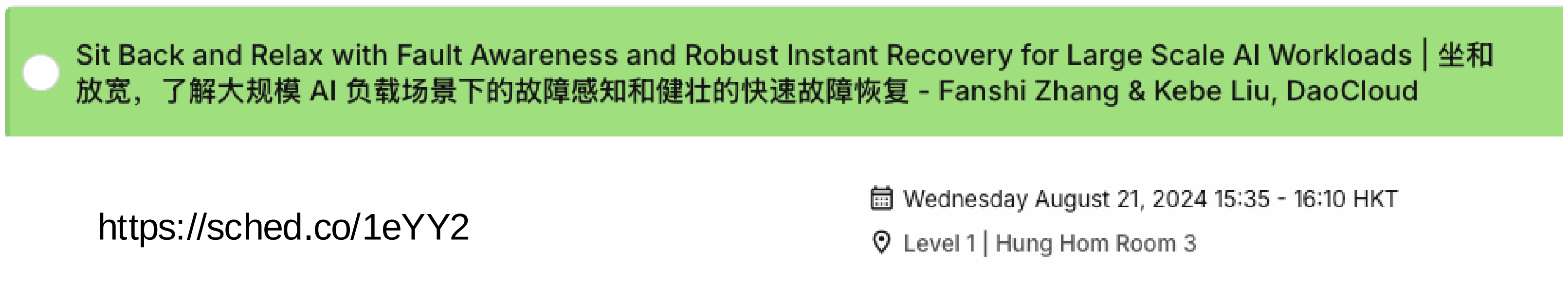
- Multi-Tenant
- Code & Data
- Scheduling
- Monitor
- Performance & Robustness

Meta:

during a 54-day Llama 3 405B training,
Average interruptions **every 3 hours**,
>50% of the 419 unexpected
were caused by issues with **GPUs** or their onboard HBM3 memory.


aware, locate, diagnostic

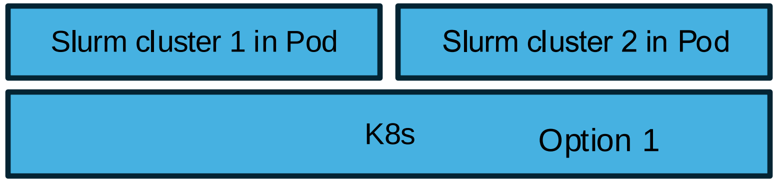
mitigation , resume


○ Sit Back and Relax with Fault Awareness and Robust Instant Recovery for Large Scale AI Workloads | 坐和放宽，了解大规模 AI 负载场景下的故障感知和健壮的快速故障恢复 - Fanshi Zhang & Kebe Liu, DaoCloud

https://sched.co/1eYY2

Wednesday August 21, 2024 15:35 - 16:10 HKT
Level 1 | Hung Hom Room 3

How to Unify K8S & Slurm



Slurm cluster 1 in Pod

Slurm cluster 2 in Pod

K8s

Option 1

Challenges of co-hosting K8S & Slurm on the same cluster

1. How to deploy slurmd(slurm agent) and container-runtime?

- Option 1: slurm cluster in pod →
- Option 2: slurm/K8s live in the same place 👍

  slurmd does not conflict with containerd or kubelet

2. How to make them aware of each other's resource usage ?

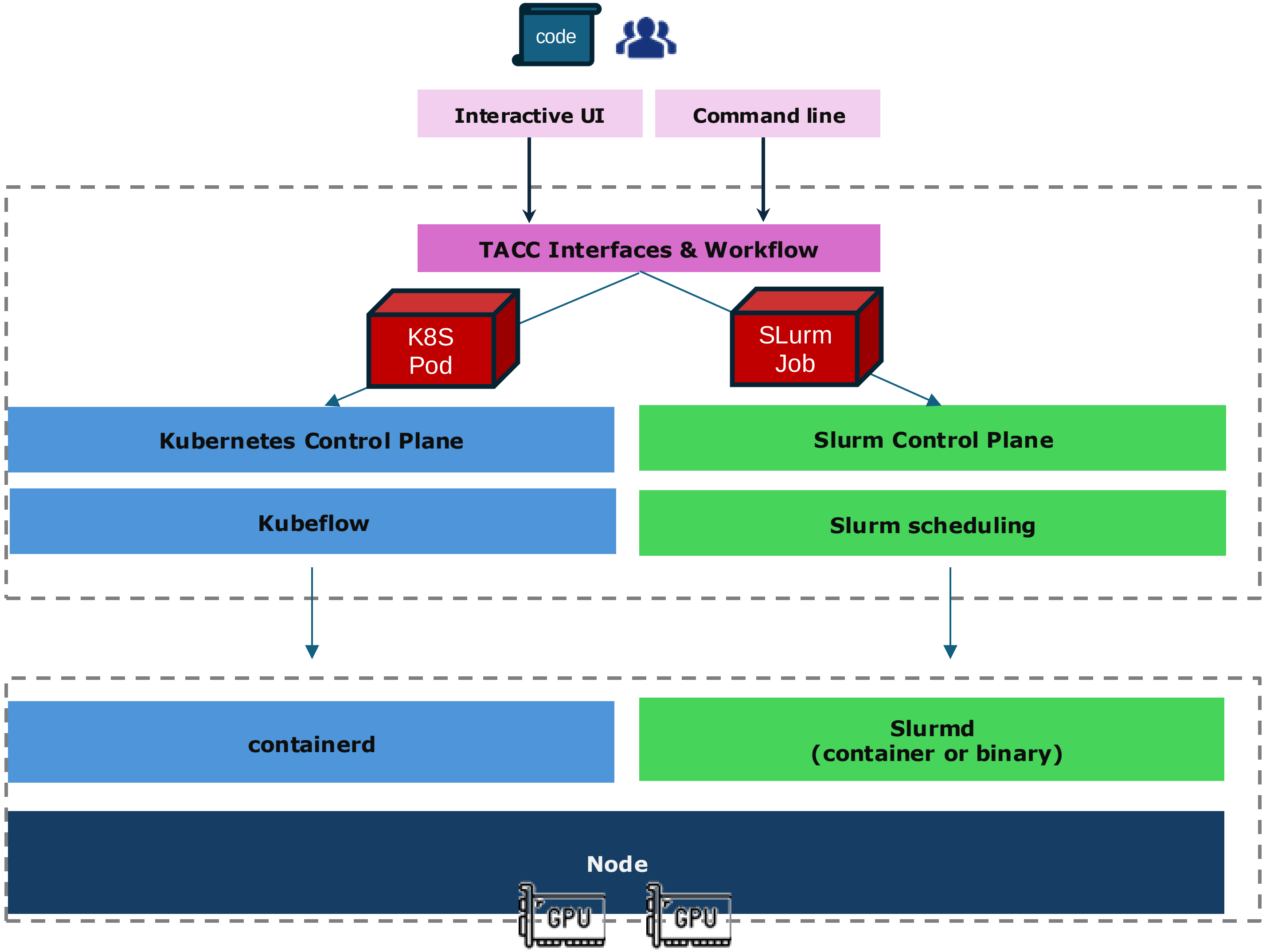
- **problem**: if a GPU been occupied by K8S Pod, Slurm isn't aware of it

PS: "SUNK" is not available yeah.. https://slurm.schedmd.com/SC23/Slurm-and-or-vs-Kubernetes.pdf

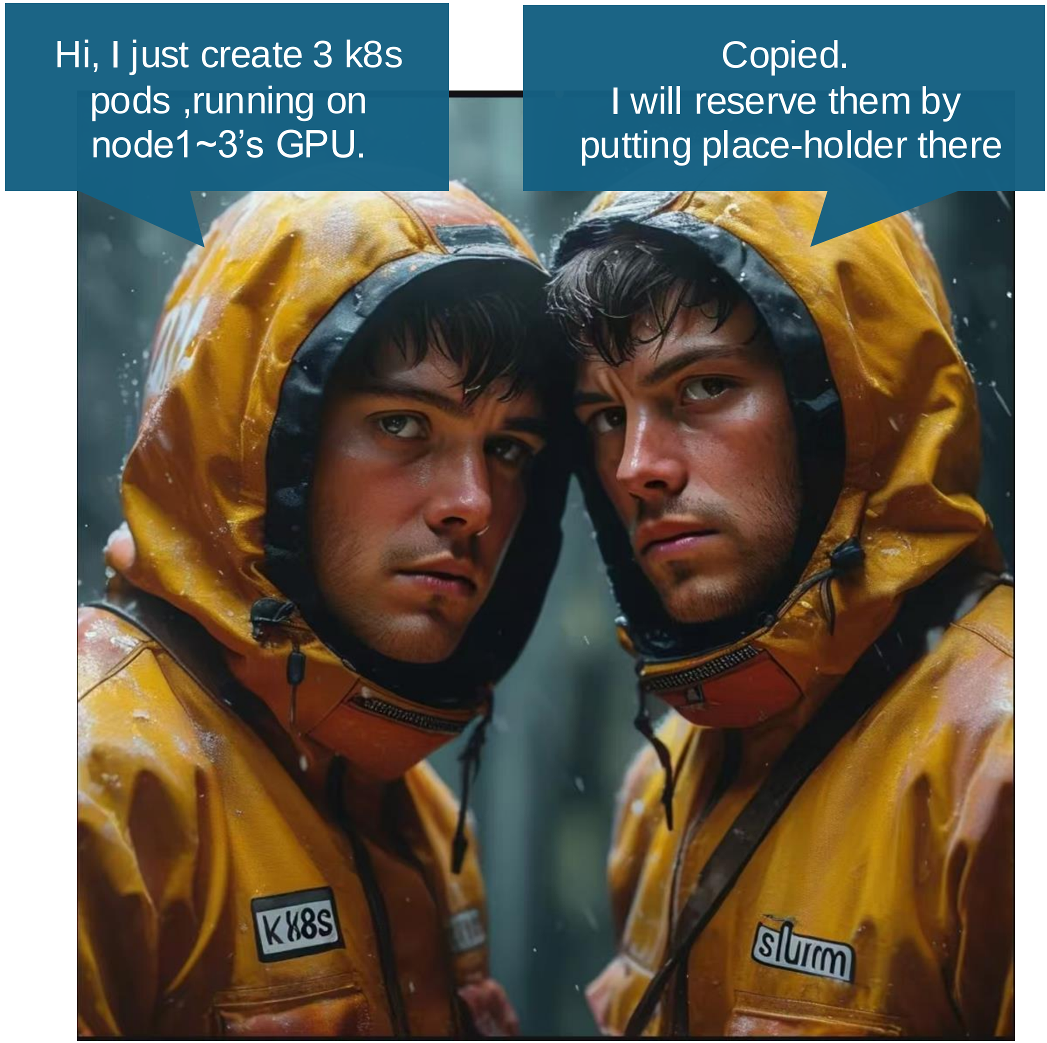
So TACC's way as below:

Co-host Deployment

every GPU machine is ready to handle either Slurm or K8S workloads

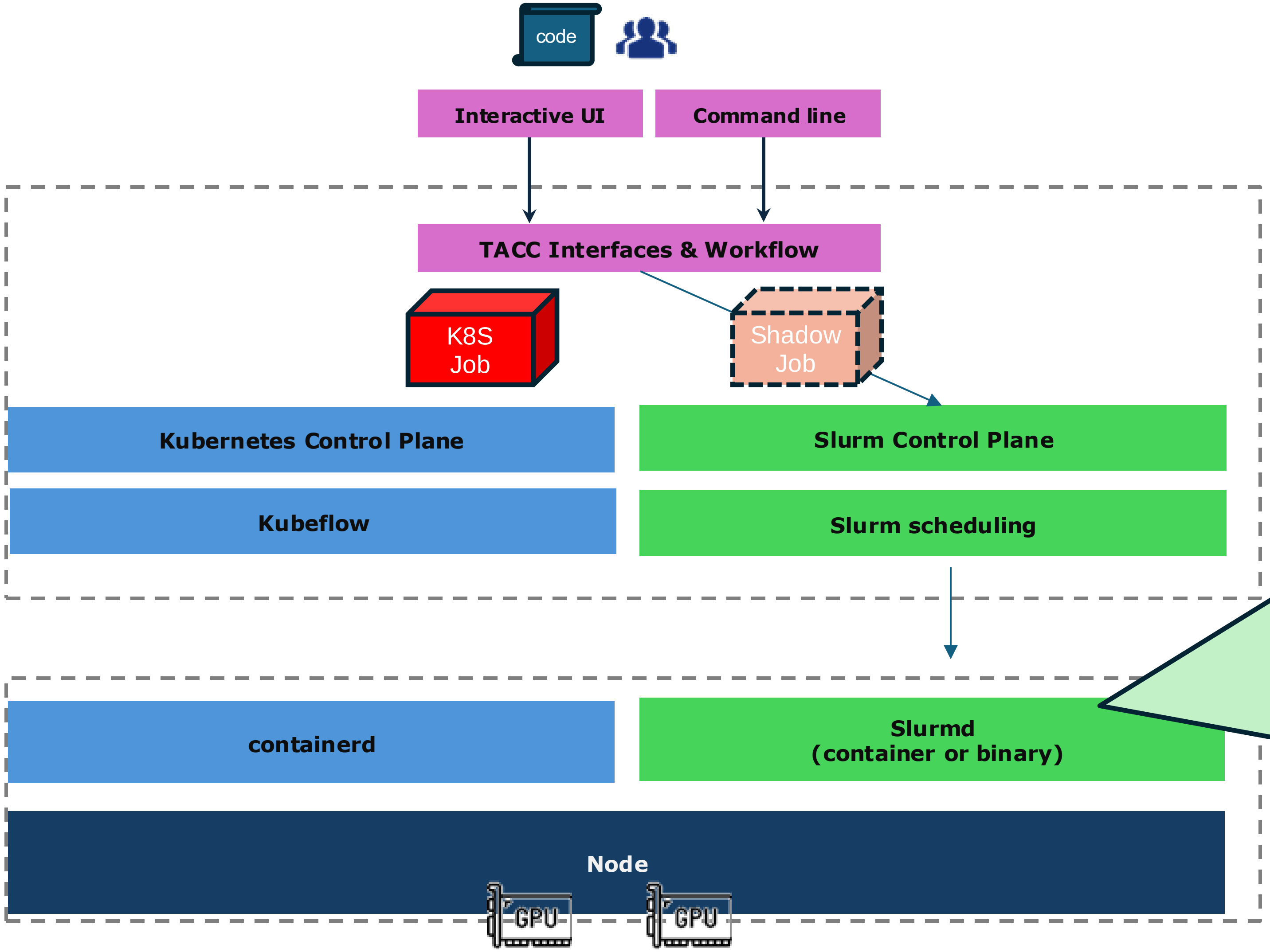


k8s will not use those resource Slurm already consumes, and vice versa.


Hi, I just create 3 k8s pods ,running on node1~3's GPU.
Copied.
I will reserve them by putting place-holder there
K8S
slurm

Shadow Resource Place Holder

Slurm job as place holder but should not really use the GPU



Example:

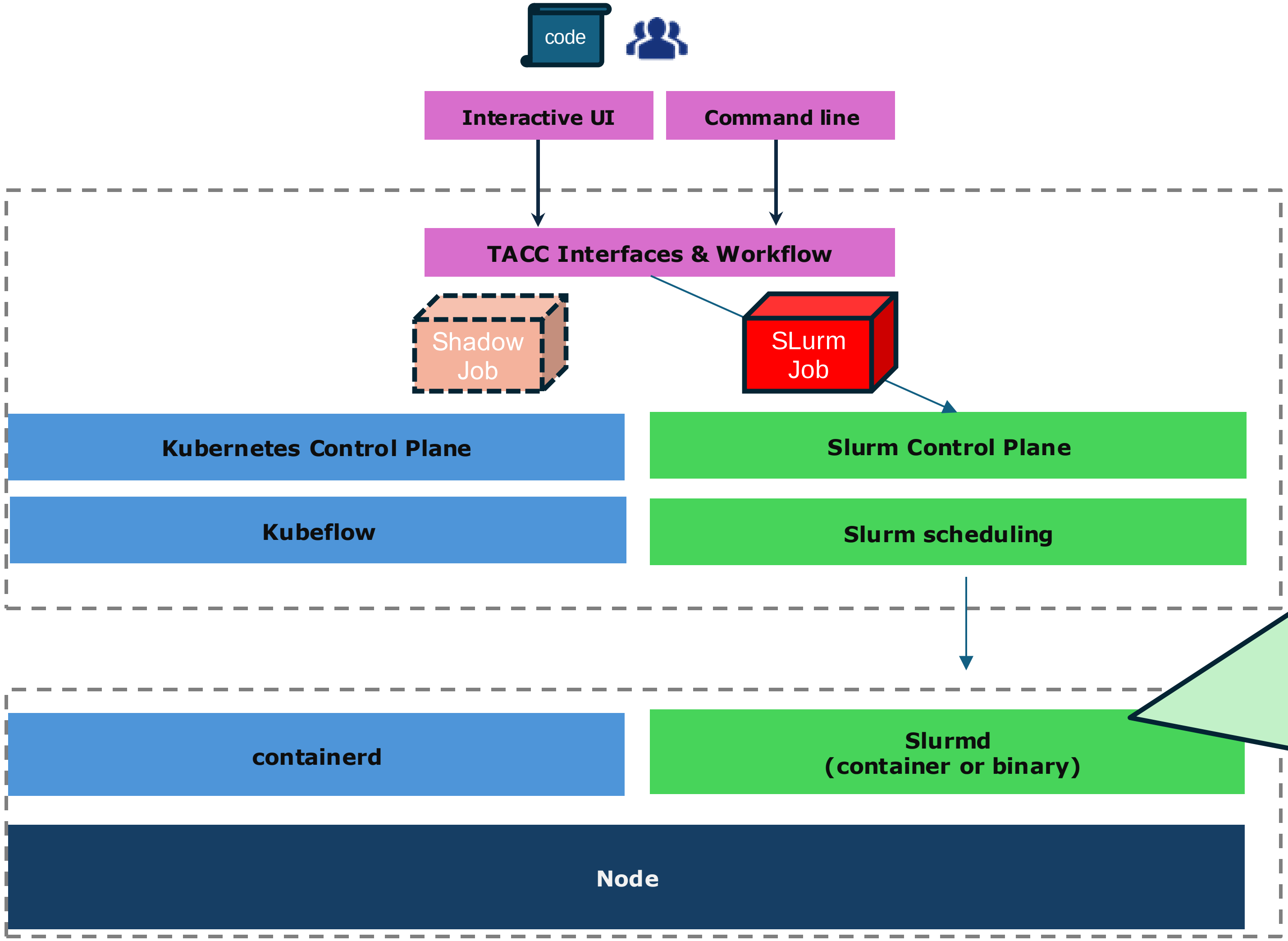
K8s Pod (actual workload)

- 2C/4G/1GPU
- 4 replicas scheduled to Node 1/3/4/5

Shadow Slurm job :

- Command:
- sleep and wait job $aboveJob (`kubectl wait `)
- -N 4 –n 4
- --cpus-per-task=2
- --mem=4G
- -gres=gpu:1
- --nodelist node[1,3,4,5]

Vice versa



Example:

<u>Slurm Job</u> (actual workload)

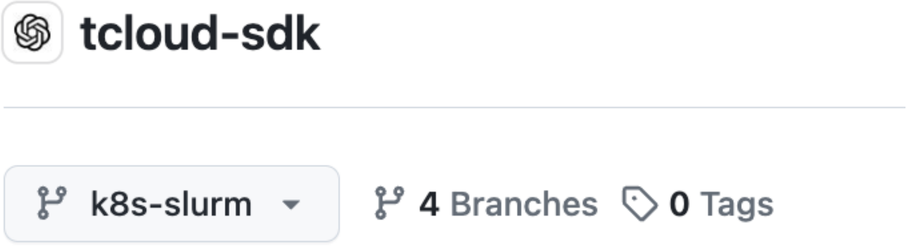
- 4 process scheduled to Node 1/3/4/5

<u>Shadow K8S Pods</u> :

- Command: `squeue` to wait job completion
- Request same resources as slurm job for each

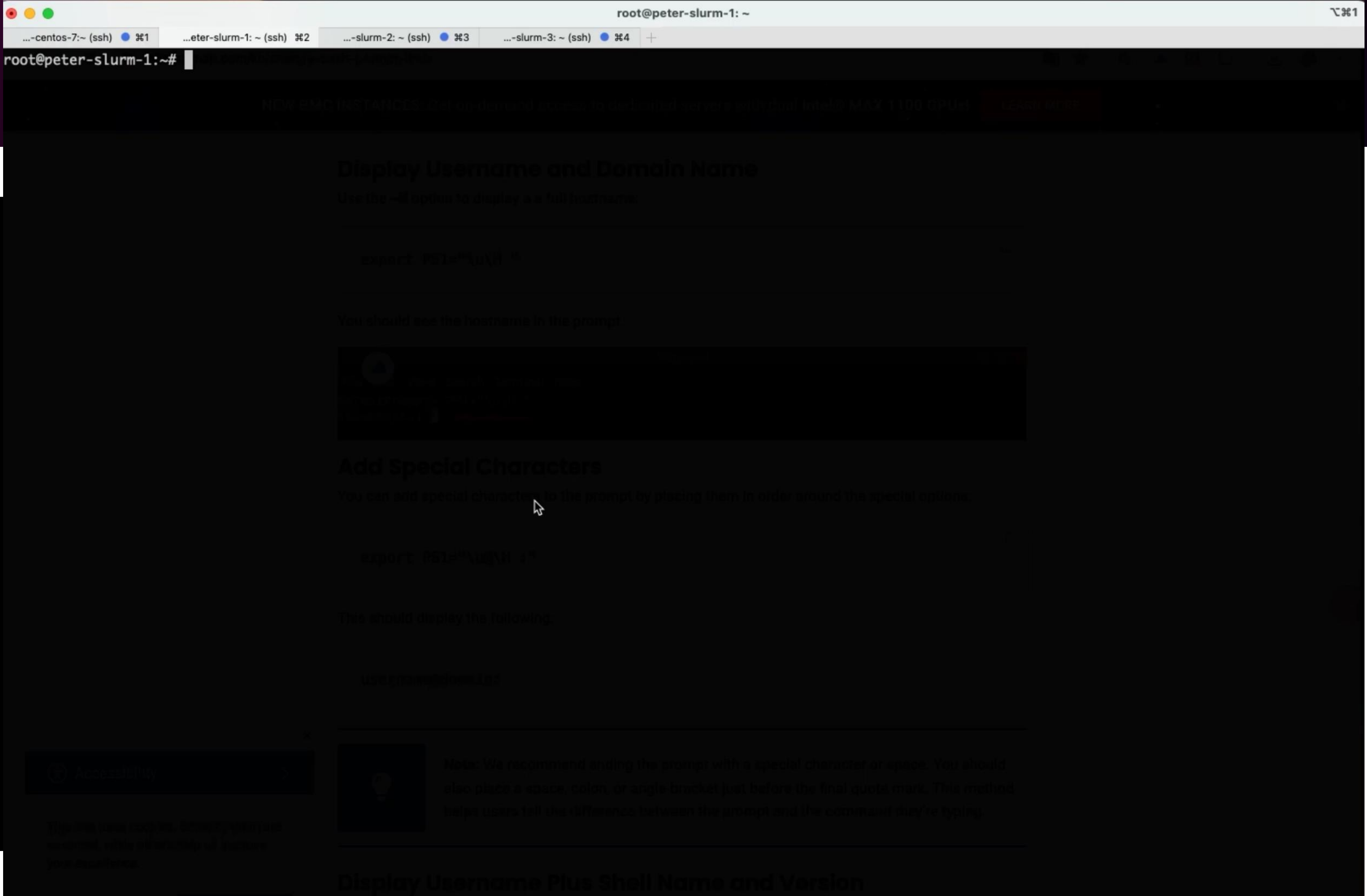
Demo Code

- https://github.com/turingaicloud/tcloud-sdk/tree/k8s-slurm



tcloud-sdk

Demo



```
root@peter-slurm-1:~#
```

Display Username and Domain Name

Use the –H option to display a a full hostname:

```
export PS1="\u\H "
```

You should see the hostname in the prompt:

Add Special Characters

You can add special characters to the prompt by placing them in order around the special options:

```
export PS1="\u@\H :"
```

This should display the following:

```
username@domain:
```

Note: We recommend ending the prompt with a special character or space. You should also place a space, colon, or angle bracket just before the final quote mark. This method helps users tell the difference between the prompt and the command they're typing.

Display Username Plus Shell Name and Version

FAQ

Q1: Race Condition?

A1: good catch. So the best way out is either to do this in each’s scheduler level, or control the entrance in a central manner . This is in future consideration.

Q2: place-holder shows node-level grain size , how about GPU-level(in one node)?

A2: Both slurm GRES-plugin and K8s device-plugin schedule those GPU on the same node in order, so it still works in GPU-in-node grant size.

Thank you

We hope **TACC**’s practice can inspire the community and industry, about how to streamline HPC and K8S workload together, and build a scientist friendly toolset and platform, with CNCF stacks.

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

DaoCloud