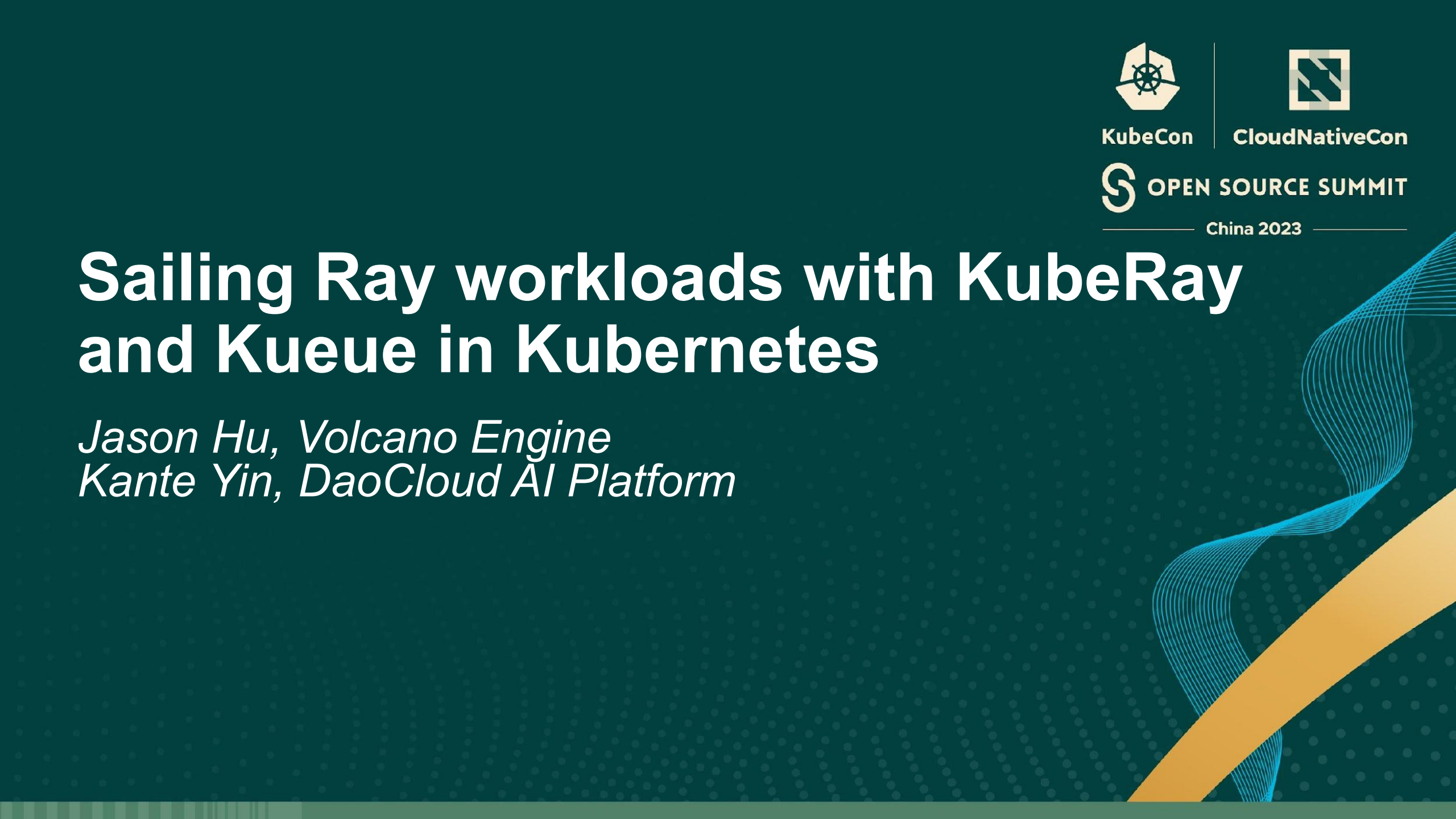

KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023



Sailing Ray workloads with KubeRay and Kueue in Kubernetes

Jason Hu, Volcano Engine
Kante Yin, DaoCloud AI Platform

A bit more about us

Jason Hu (胡元哲)
Software Engineer @Bytedance, Volcano Engine
https://github.com/Basasuya
Interested in Ray ecosystem and AI related infra

Kante Yin (殷纳)
Senior Software Engineer @DaoCloud
https://github.com/kerthcet
Kubernetes SIG-Scheduling Maintainer & Kueue Approver
I ❤️ OpenSource 🏀 & ⚽ & 🥊

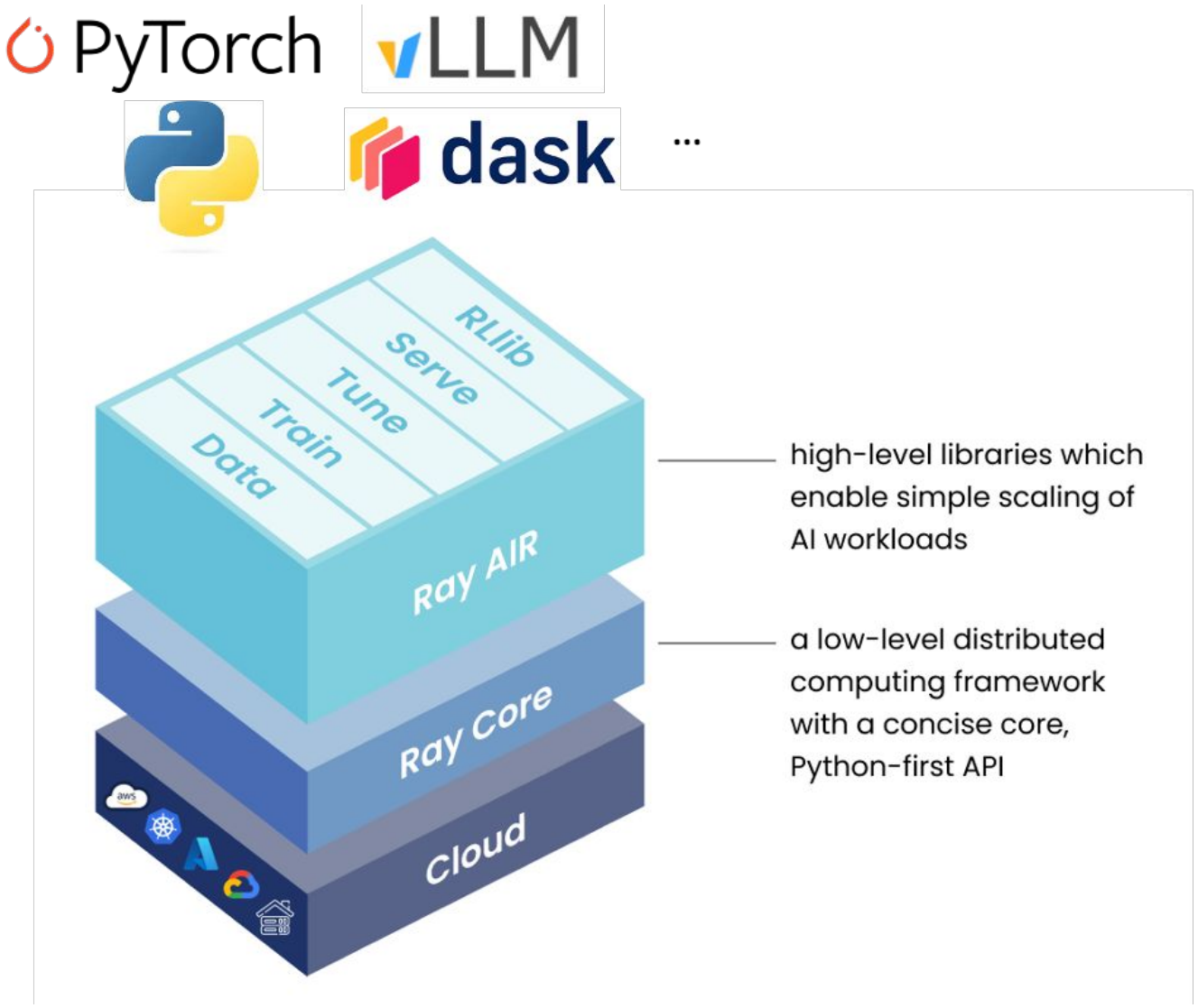
Agenda

- How Ray framework benefits AI researchers
- How to sail Ray workloads with Kuberay In ByteDance
- How Kueue helps to manage RayJobs
- Q & A

Part I

How Ray framework benefits AI researchers

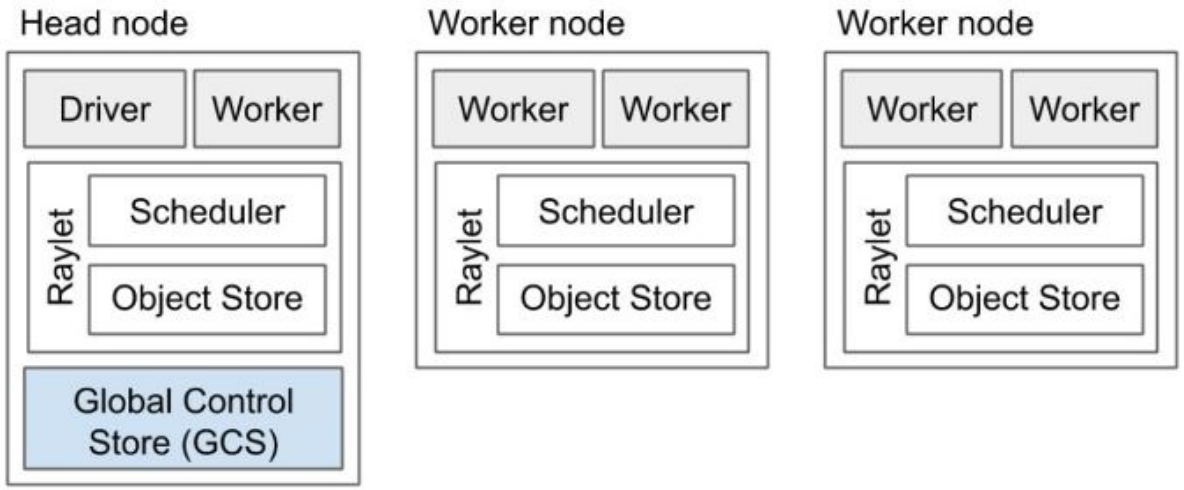
What Is Ray



- Ray is an open-source unified framework for scaling AI and Python applications.
- Ray provides the compute layer for parallel processing so that you don't need to be a distributed systems expert.

Ray Core

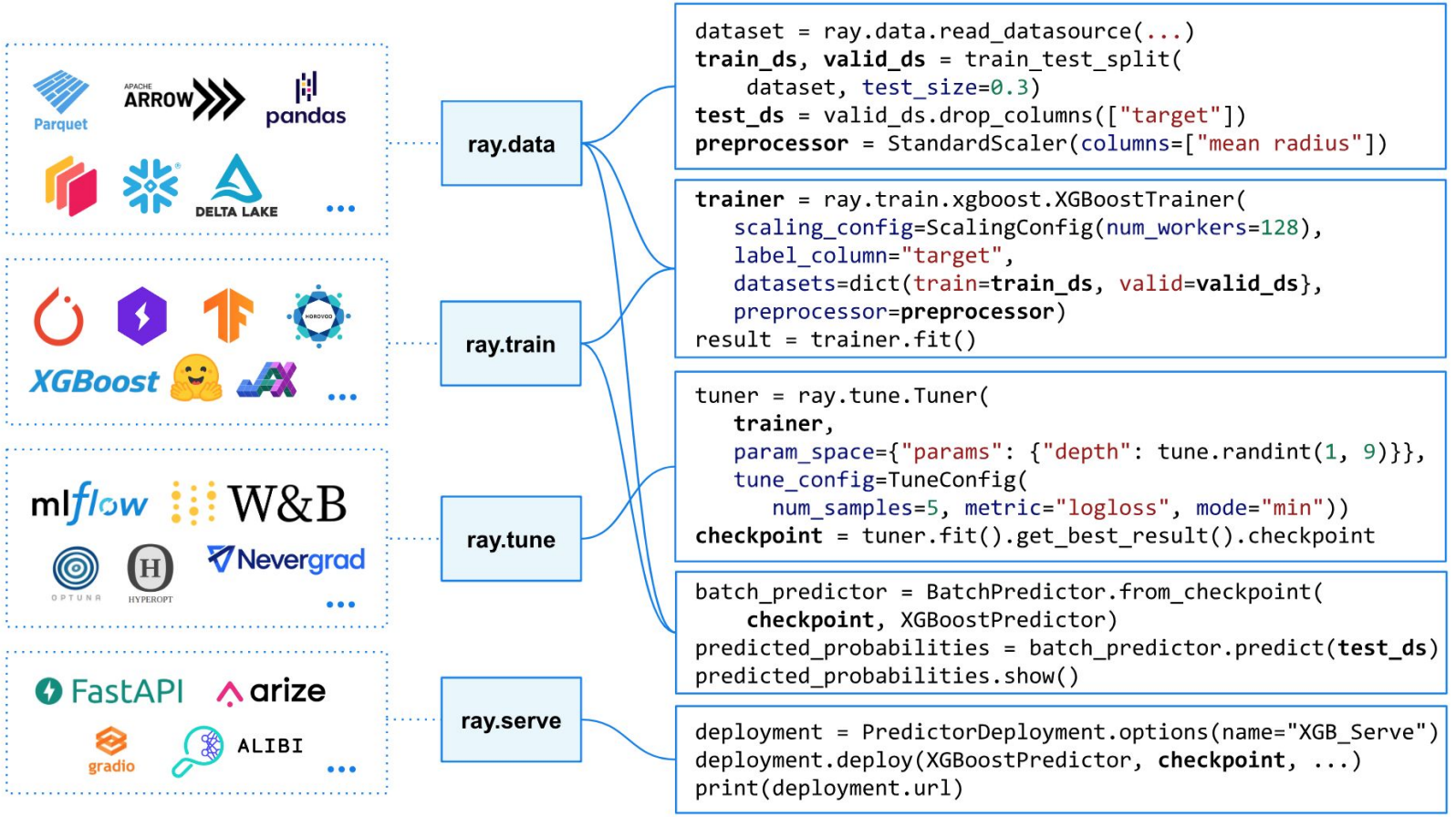

```python
import ray

@ray.remote
def square(x):
    return x * x

futures = [square.remote(i) for i in range(4)]
# Retrieve results.
print(ray.get(futures))

@ray.remote
class Counter:
    def __init__(self):
        self.i = 0

    def get(self):
        return self.i

# Create a Counter actor.
c = Counter.remote()
# Retrieve final actor state.
print(ray.get(c.get.remote()))
```



A Ray cluster.

Ray AIR

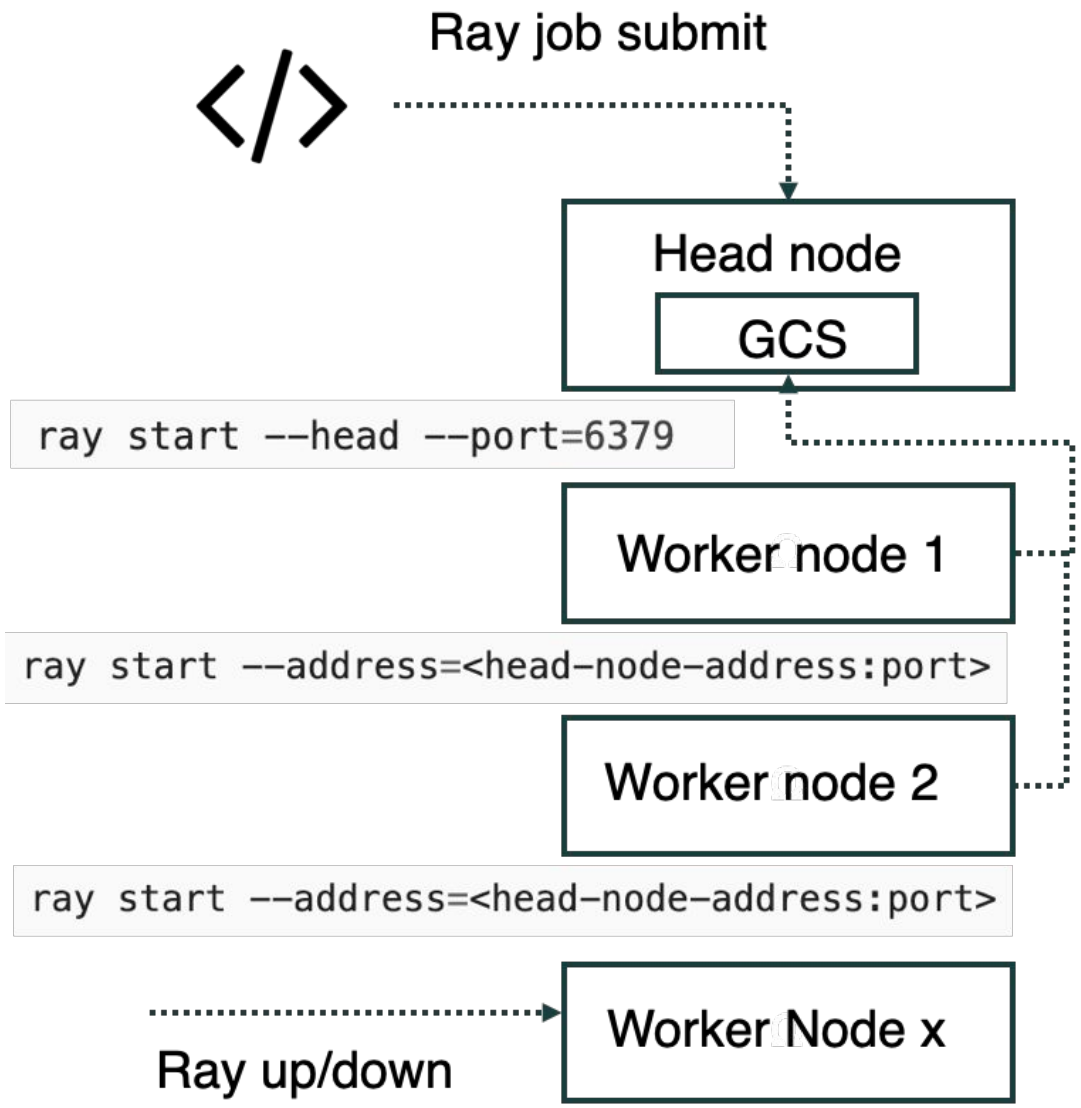
Ray provides a unified API for the ML ecosystem. This diagram shows how Ray enables an ecosystem of libraries to be run at scale in just a few lines of code.



Part II

How to sail Ray workloads with Kuberay in ByteDance

Problem using Ray w/o kuberay

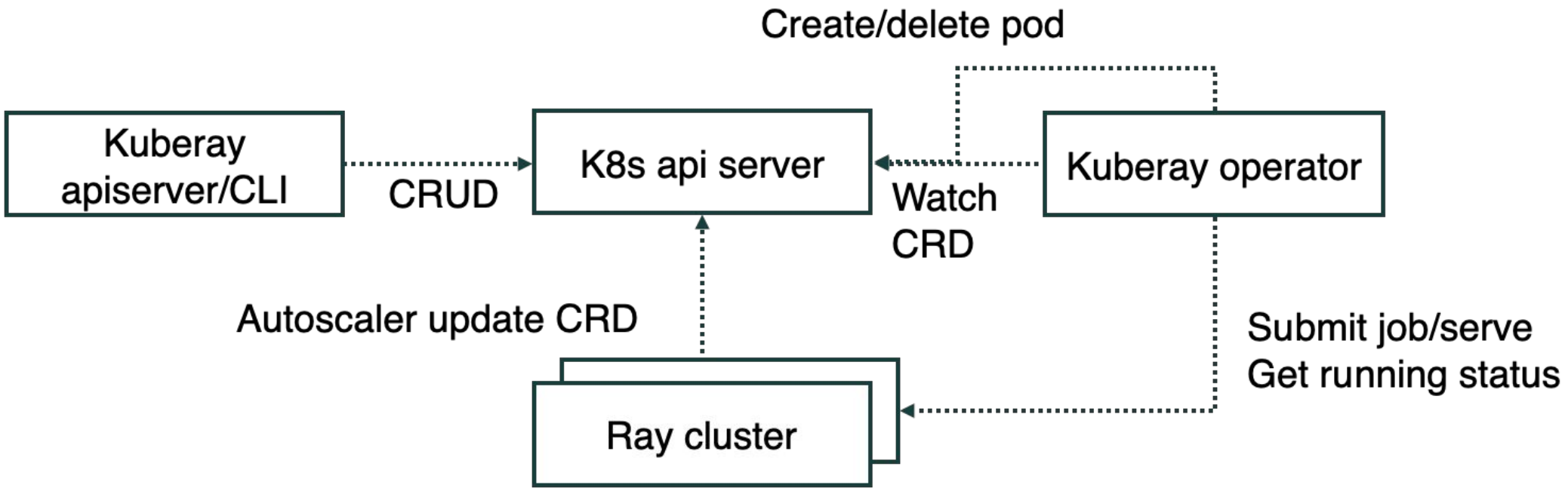


Problems

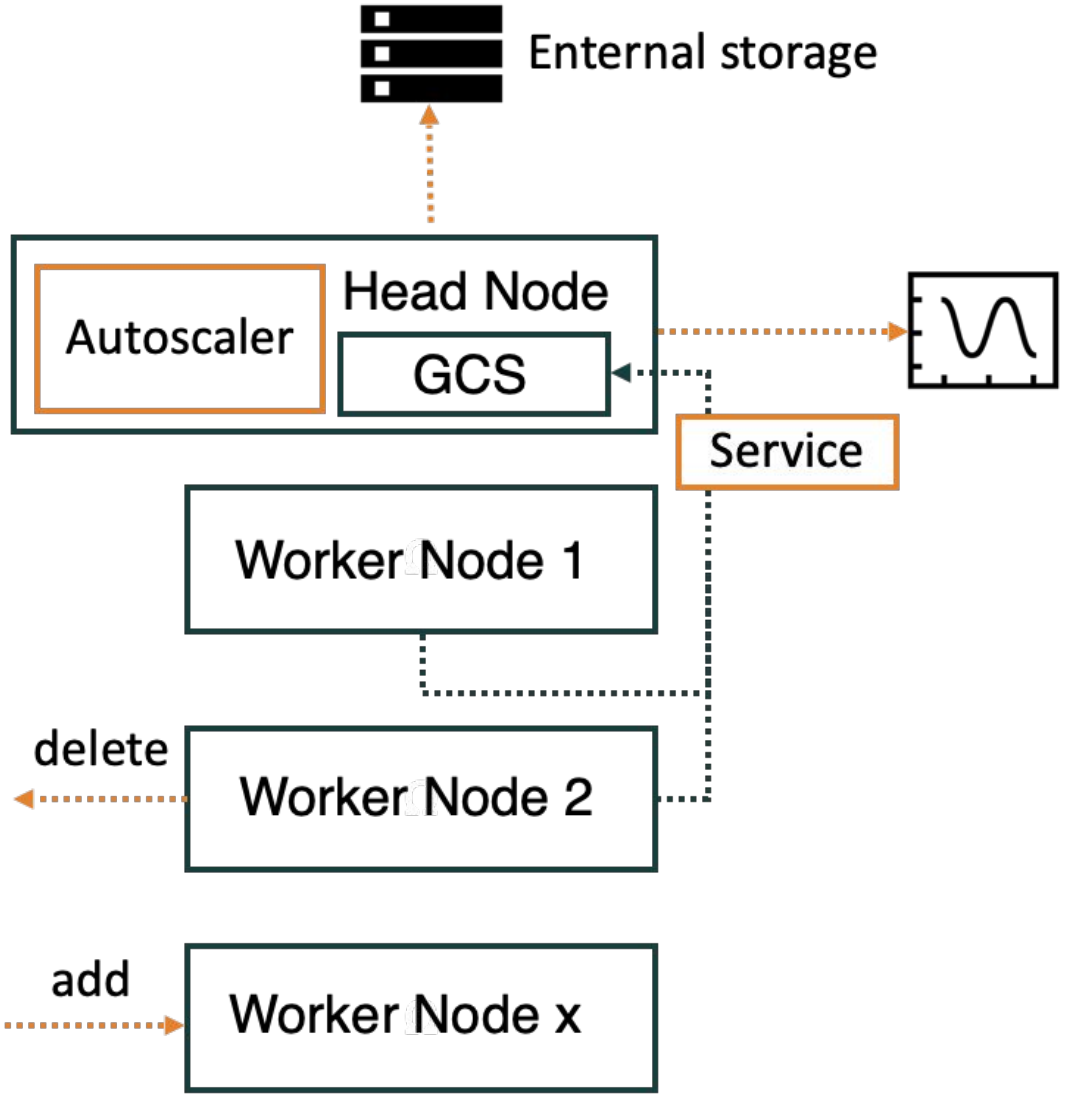
- Lack the ability to recover from failures for each worker node
- Hard to manage and submit lots of jobs with ease
- Miss observability such as metric, alarm
- Hard to achieve HPA/VPA
- …

What is Kuberay

- **RayCluster**: includes cluster creation/deletion, autoscaling, and ensuring fault tolerance
- **RayJob**: creates the RayCluster (or select the existed one) and submits job when cluster is ready
- **RayService**: offers zero-downtime upgrades for RayServe deployment graph and high availability
- **KubeRay APIServer/python client/CLI**: provides several tools to create,delete,update,query related CRDs without using kubeconfig



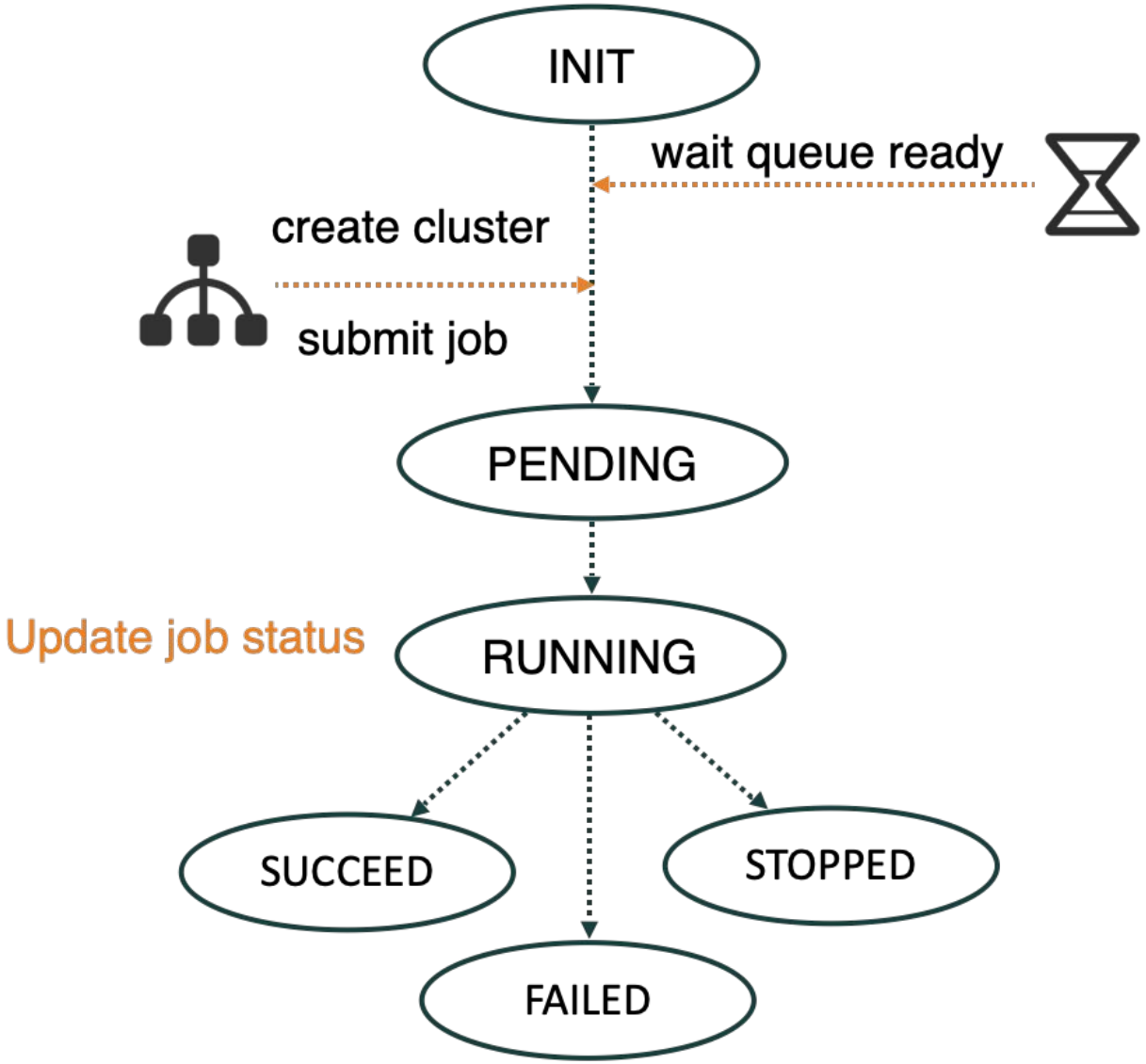
RayCluster



RayCluster provides high availability to the Ray cluster with a bunch of enhancements:

- Support pod failover and hot update
- Service discovery with head and worker nodes
- Implement head node fault-tolerance using external storage
- Observability like metric, status, endpoints, etc.
- HPA through ray autoscaler

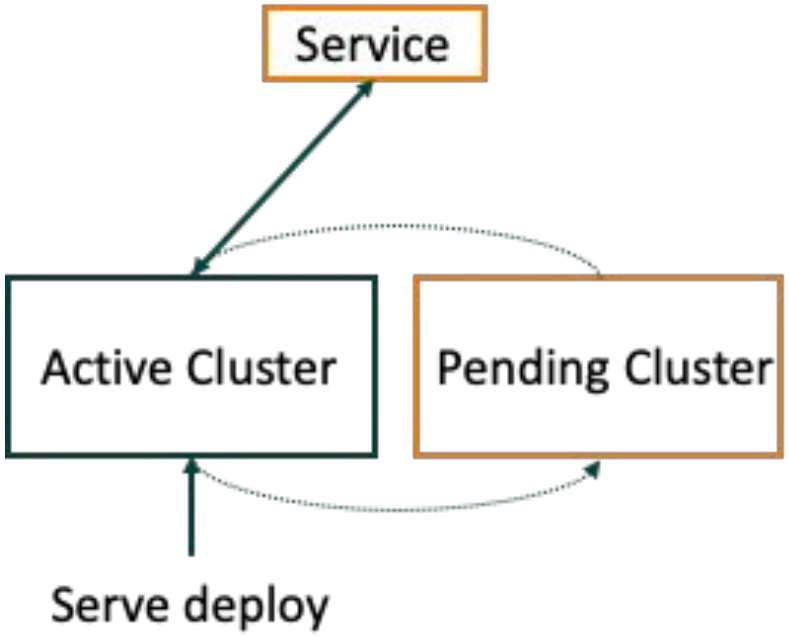
RayJob



RayJob CRD provides a solution to submit jobs in the production environment continuously and efficiently:

- Batch scheduling
- Support creating a RayCluster automatically alongside with the job lifecycle
- More complex job configurations such as timeout, wait node number, submit backpressure, etc.
- Store job information through CRD

RayService

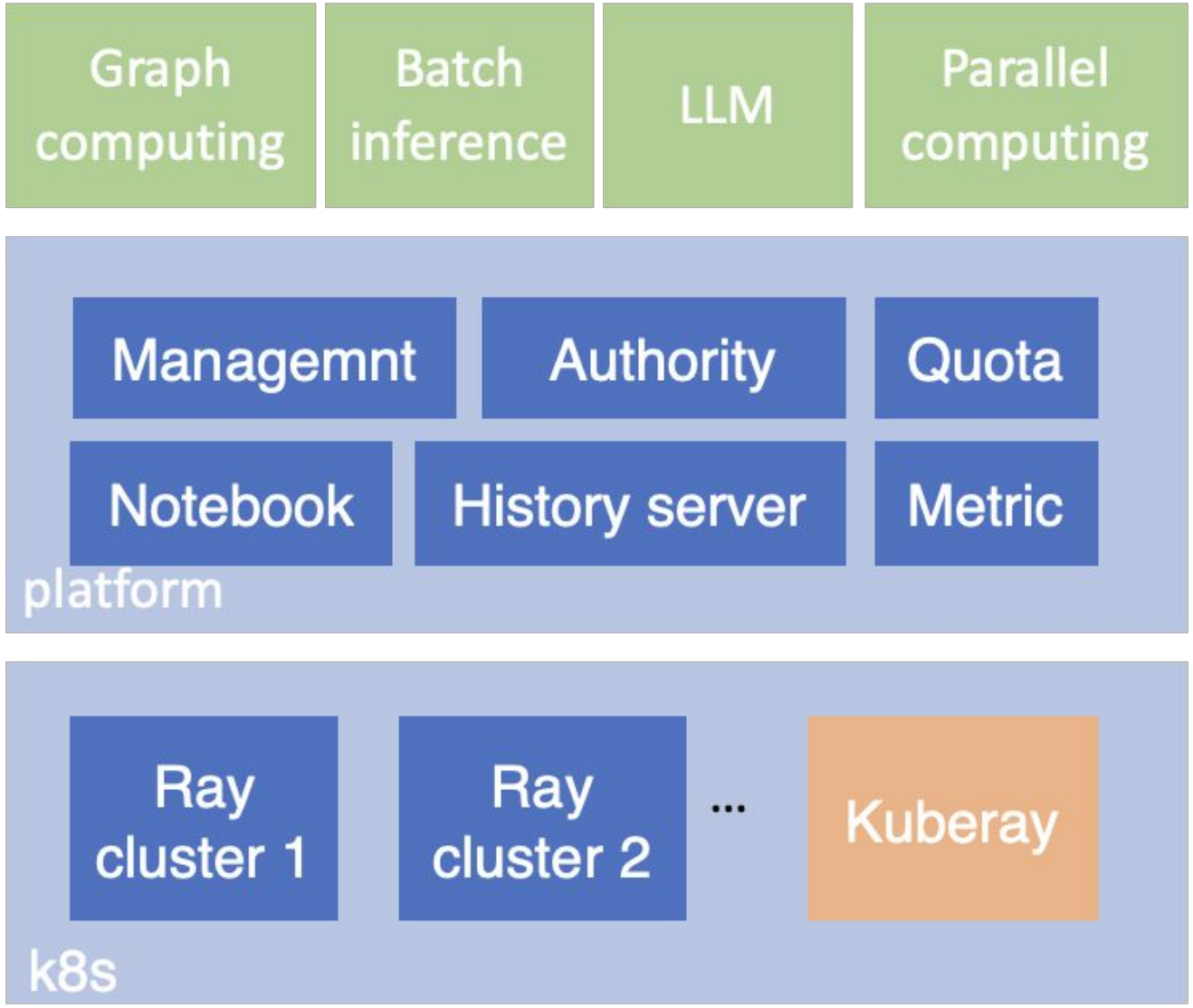



```
serveConfigV2: |
  applications:
    - name: stable_diffusion
      import_path: stable_diffusion.stable_diffusion:entrypoint
      runtime_env:
        working_dir: "https://github.com/...zip"
        pip: ["diffusers==0.12.1"]
```

RayService provides a high availability solution to deploy Ray service on cloud

- Kubernetes-native support for Ray clusters and Ray serve applications
- In-place update for Ray serve applications
- Zero downtime upgrade for Ray clusters
- Services HA

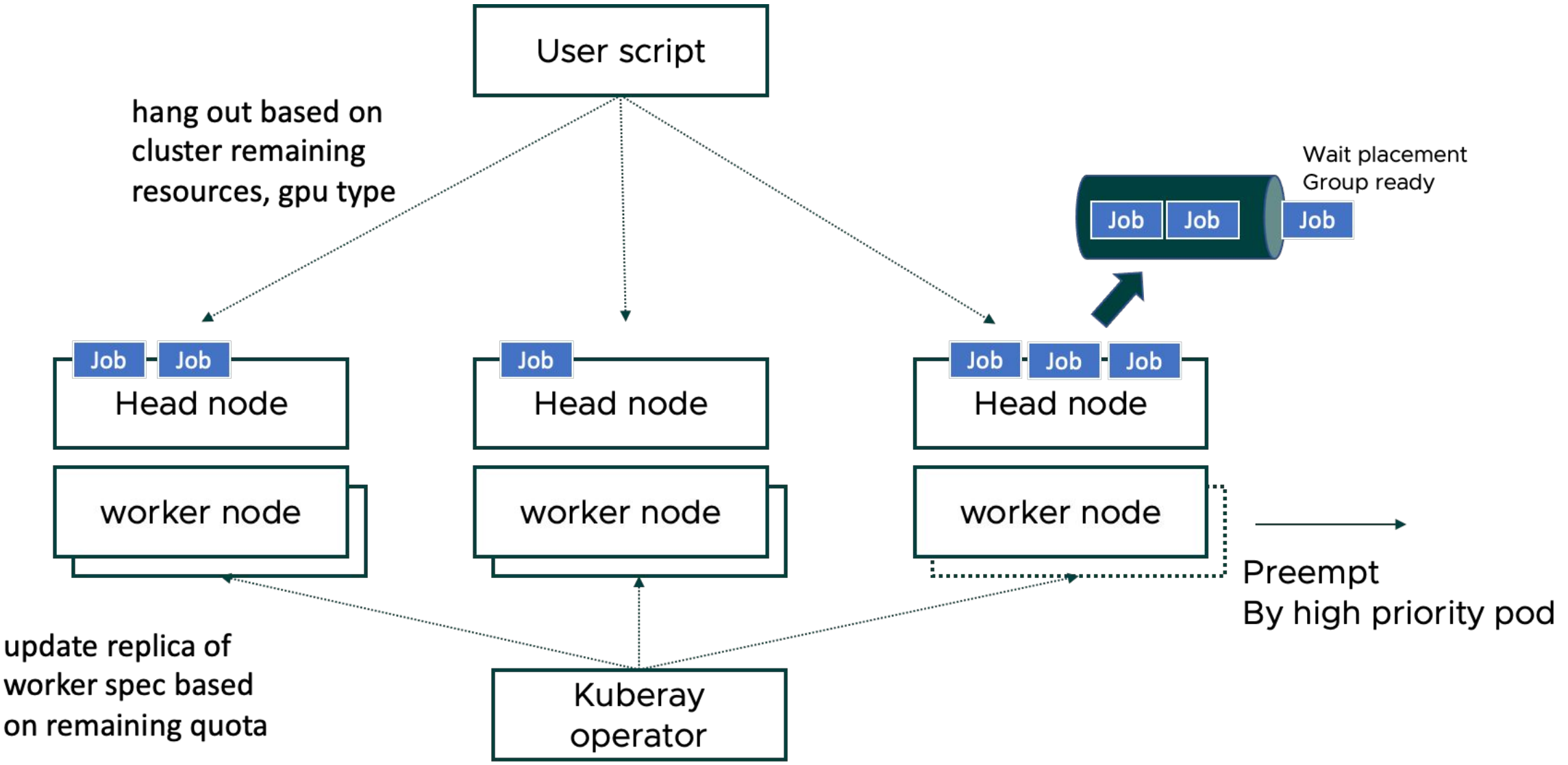
Sail Ray workloads in ByteDance



In Bytedance, we offer a complete ray ecology for many users:

- All Ray clusters are managed by kuberay
- Debug programs on long-running clusters and launch single-jobs through platform
- Provide observability, such as history server, metric, alarm, etc.

Develop on Long-running cluster



Submit RayJob CR through platform

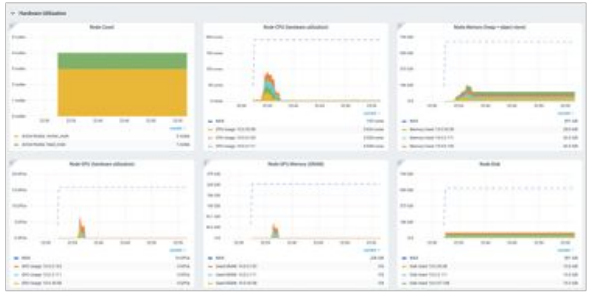
Submit single-job
to platform

Submit RayJob
To kubernetes

Kuberay
RayJob

Wait for cluster
ready
Submit Job

Develop/debug on
Long-running cluster



Watch result from

Ray History server

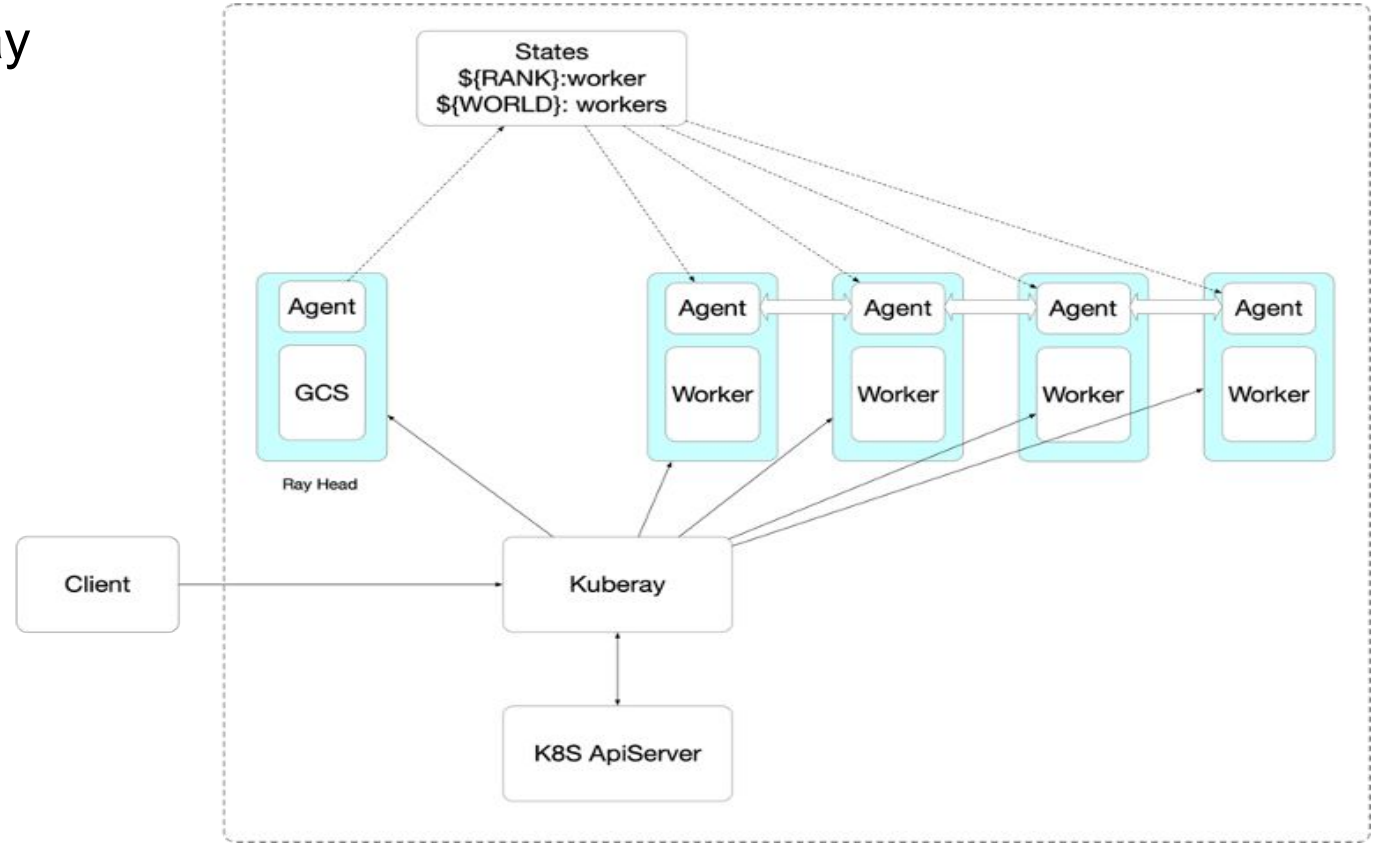
Dump event

Ray
cluster

Use Case from ByteDance

Case 1 ByteGap, Graph Analytics Platform

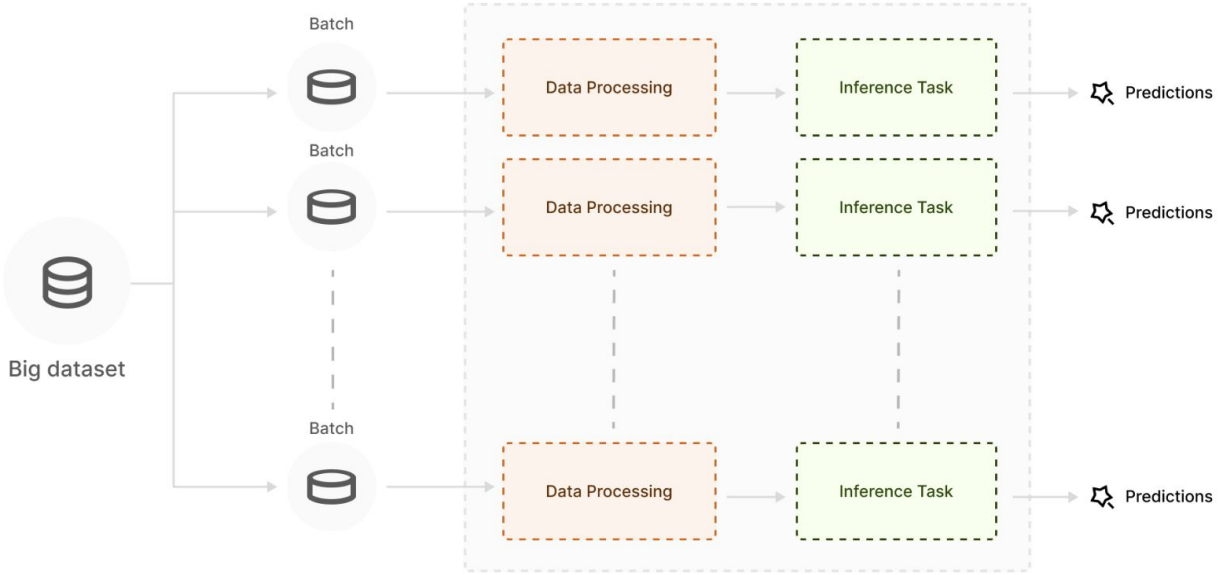
- Graph computing operators are created by ray actor
- Each actor is communicated by mpi
- Failover mechanism provided by Ray

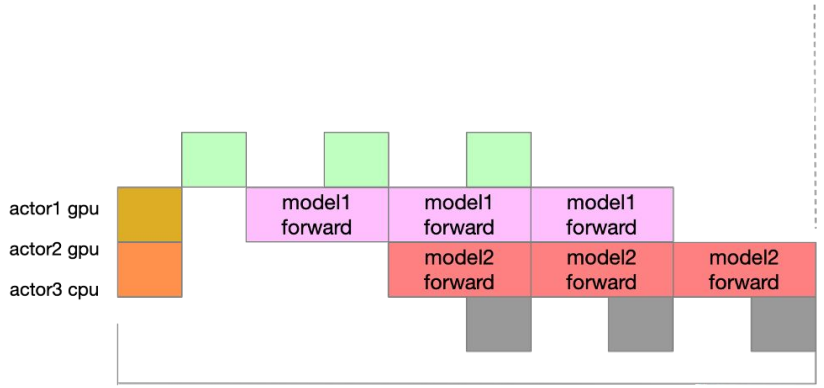


Use Case from ByteDance

Case 2: Large-scale Offline Inference

- Use offline batch inference for embedding generations
- RayData proved to be a more flexible and scalable component for large-scale model parallel inference compared to alternatives such as Spark.





Problems

Resource Is Limited

Priority Unrespected

Job Starvation

From http://www.monkeyuser.com/

Part III

How Kueue helps to manage RayJobs

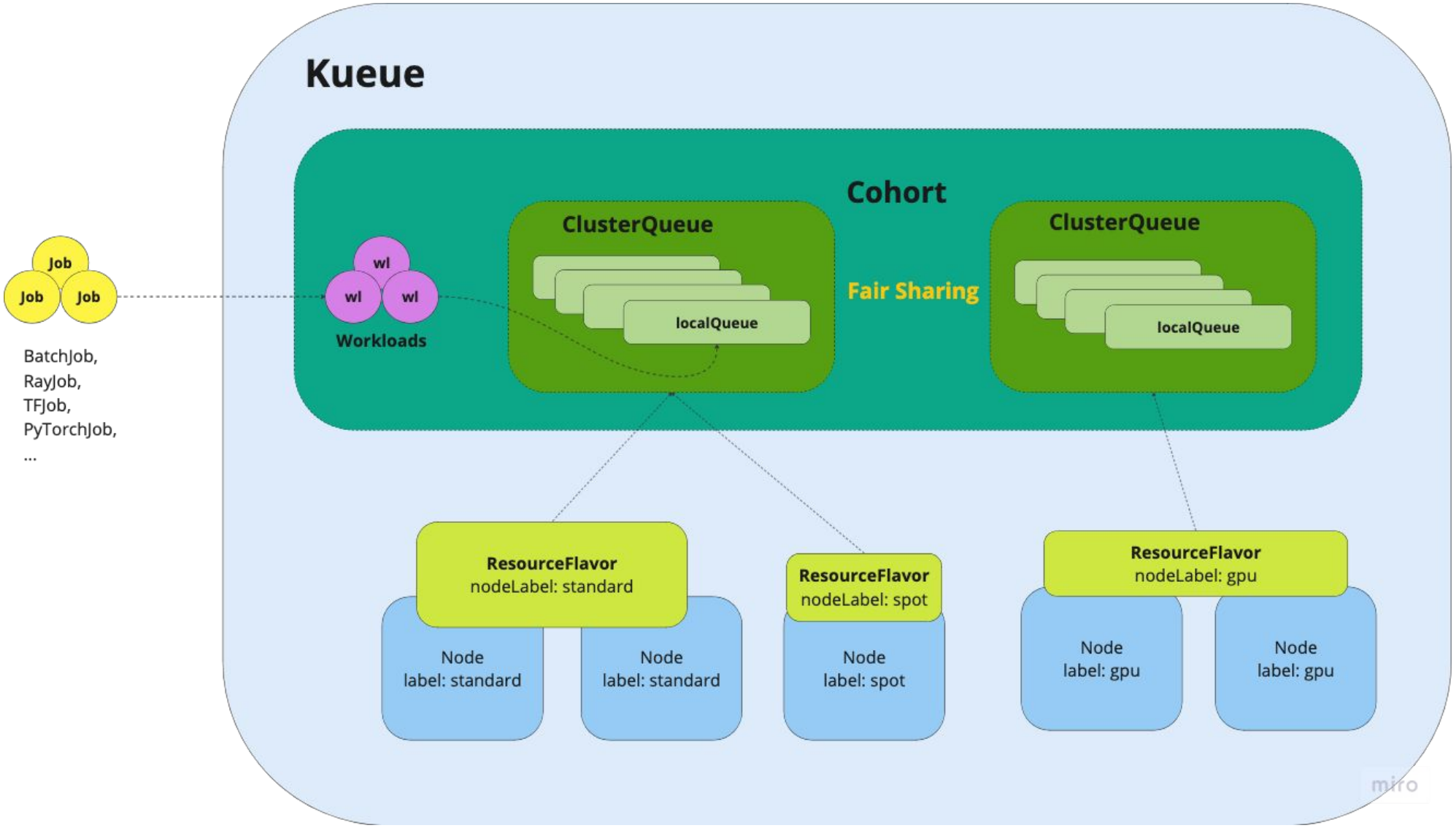
What is Kueue

A Kubernetes-Native job queueing system, offering:

- Job management with queueing policies(FIFO, BestEffort, Preemption)
- Multi-Tenant support, but no hierarchical queue support (WIP) 🤦
- Resource quota management with fair-sharing semantics
- Resource fungibility in heterogeneous clusters
- Two-Stage admission(budget, node scaling, etc.)
- …

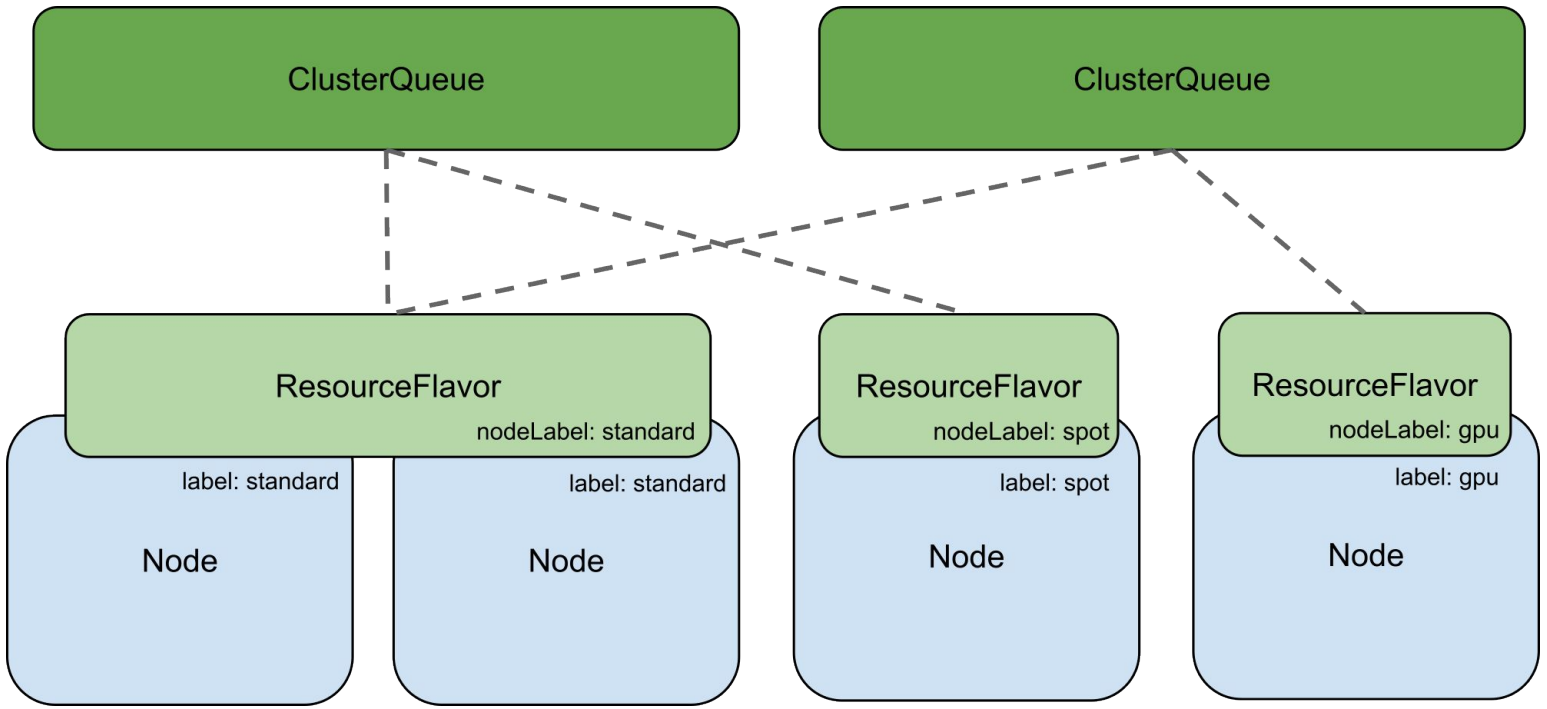
🎯 **Design principle**: compatibility and separation of concerns with standard k8s components: *kube-scheduler*, *kube-controller-manager*, *cluster-autoscaler*.

Overview

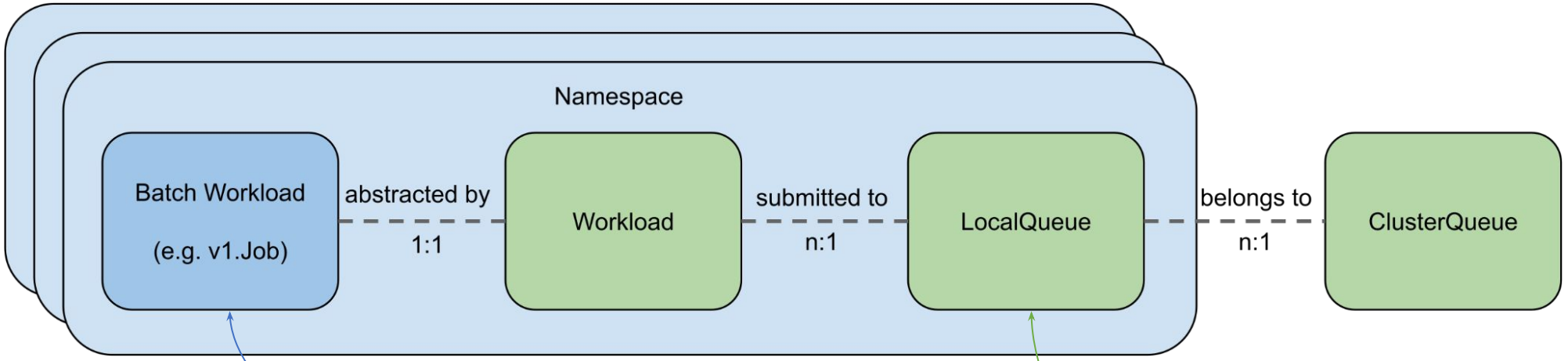

Kueue
Cohort
ClusterQueue
localQueue
Fair Sharing
ClusterQueue
localQueue
Job
Job
Job
wl
wl
wl
Workloads
BatchJob,
RayJob,
TFJob,
PyTorchJob,
...
ResourceFlavor
nodeLabel: standard
Node
label: standard
Node
label: standard
ResourceFlavor
nodeLabel: spot
Node
label: spot
ResourceFlavor
nodeLabel: gpu
Node
label: gpu
Node
label: gpu
miro

Kueue APIs - for admin


```yaml
apiVersion: kueue.x-k8s.io/v1beta1
kind: ClusterQueue
metadata:
  name: a-cluster-queue
spec:
  resourceGroups:
  - coveredResources: ["cpu", "memory"]
    flavors:
    - name: standard
      resources:
      - name: cpu
        nominalQuota: 40
        borrowingLimit: 20
      - name: memory
        nominalQuota: 128Gi
        borrowingLimit: 64Gi
    - name: spot
      resources:
      - name: cpu
        nominalQuota: 160
      - name: memory
        nominalQuota: 512Gi
```

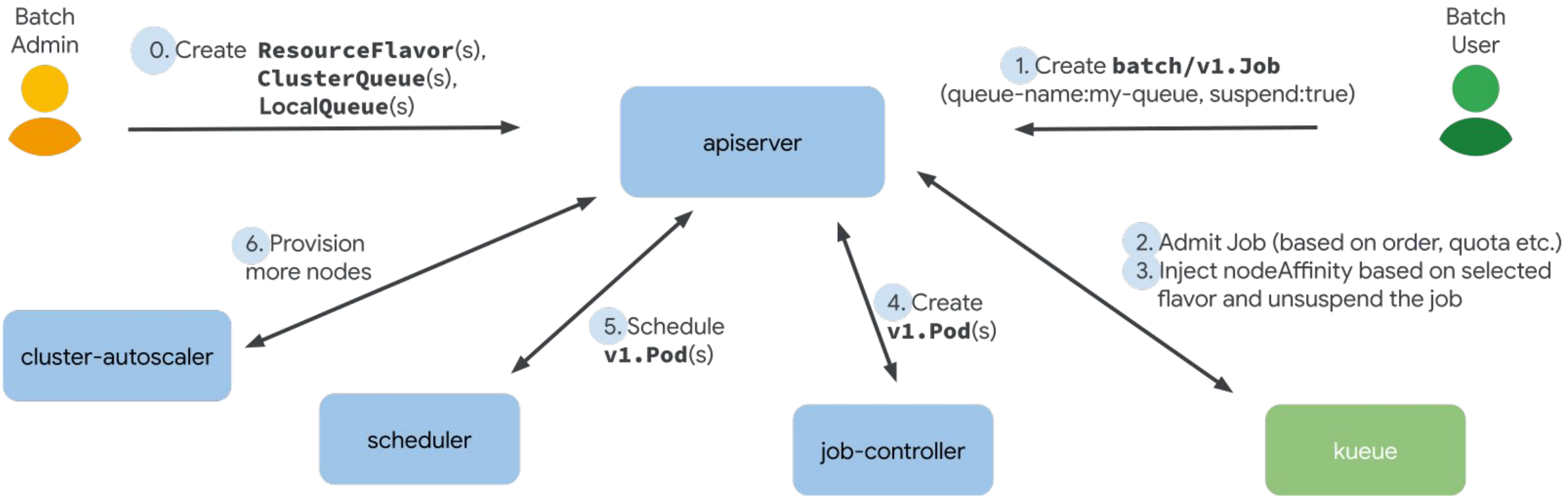


Kueue APIs - for end user




```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: sample-job
  labels:
    kueue.x-k8s.io/queue-name: user-queue
spec:
  parallelism: 3
  template:
    spec:
      containers:
      - name: job
        image: foo
        resources:
          requests:
            cpu: 1
```

How Kueue Works



Live Demo

Env Info: kuberay - 0.6.0, Kueue - main, ray - 2.5.0

Resources:

- **kubectl get priorityclasses -o custom-columns=NAME:.metadata.name,VALUE:.value**

NAME	VALUE
high-priority	1000000
low-priority	100

- **kubectl get node -o custom-columns=NAME:.metadata.name,CPU:.status.allocatable.cpu,Mem:.status.allocatable.memory**

NAME	CPU	Mem
kind-control-plane	4	16398656Ki
kind-worker	4	16398656Ki
kind-worker2	4	16398656Ki

Integrations

Integrated
WIP

Kueue

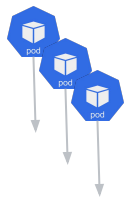
Job Framework (I/F)

BatchJob, JobSet, Pod

RayJob

Kubeflow

MPIJob, TFJob, PyTorchJob,
XGBoostJob, PaddleJob, MXJob, …



More, Flux MiniCluster etc., see *Kueue.sh*

Come on

Q & A

Thank you!

References:
https://github.com/ray-project/kuberay
https://github.com/kubernetes-sigs/kueue
https://github.com/Basasuya
https://github.com/kerthcet