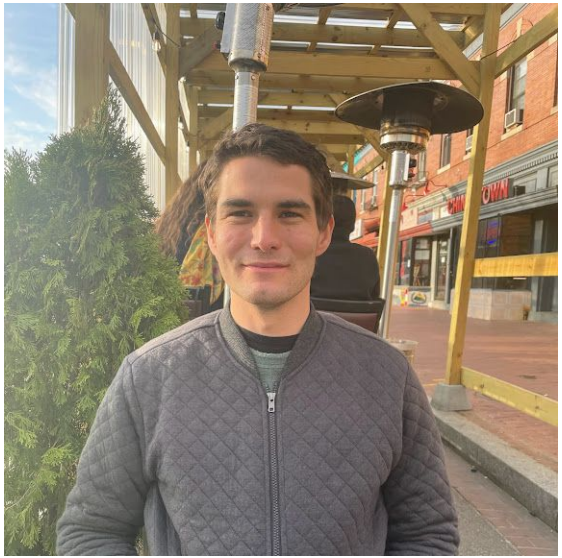
Improving Kubernetes for HPC/AI/ML Workloads

Kevin Hannon
FOSDEM 2024

Who I am

- Senior Software Engineer at Red Hat
- Red Hat Engineer focused on Kubernetes and Openshift
- Developer on Kubernetes (Job, JobSet and contributions to other CNCF batch projects)



Outline of Talk

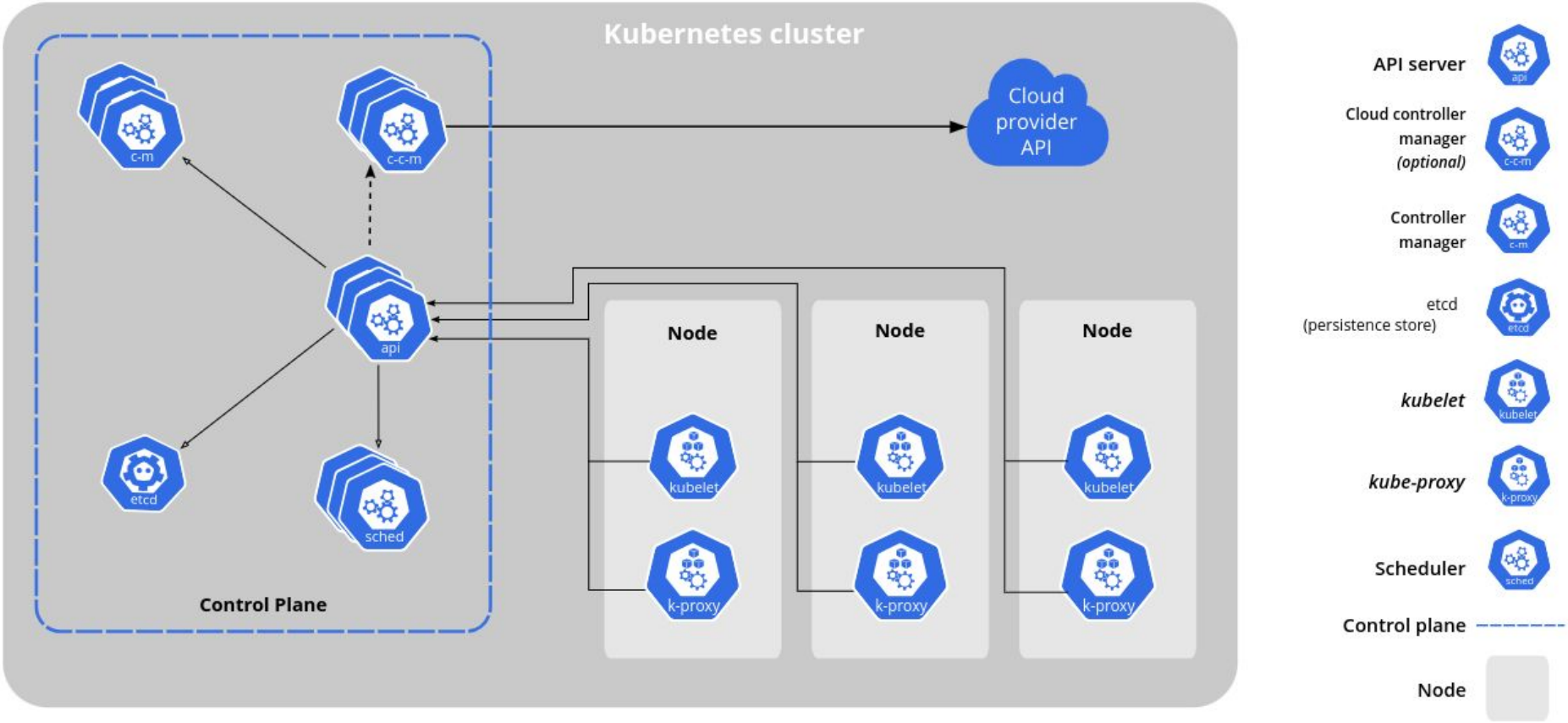
- Kubernetes
- Survey of existing tools
- Batch working group
- Job API additions
- Queueing
- JobSet

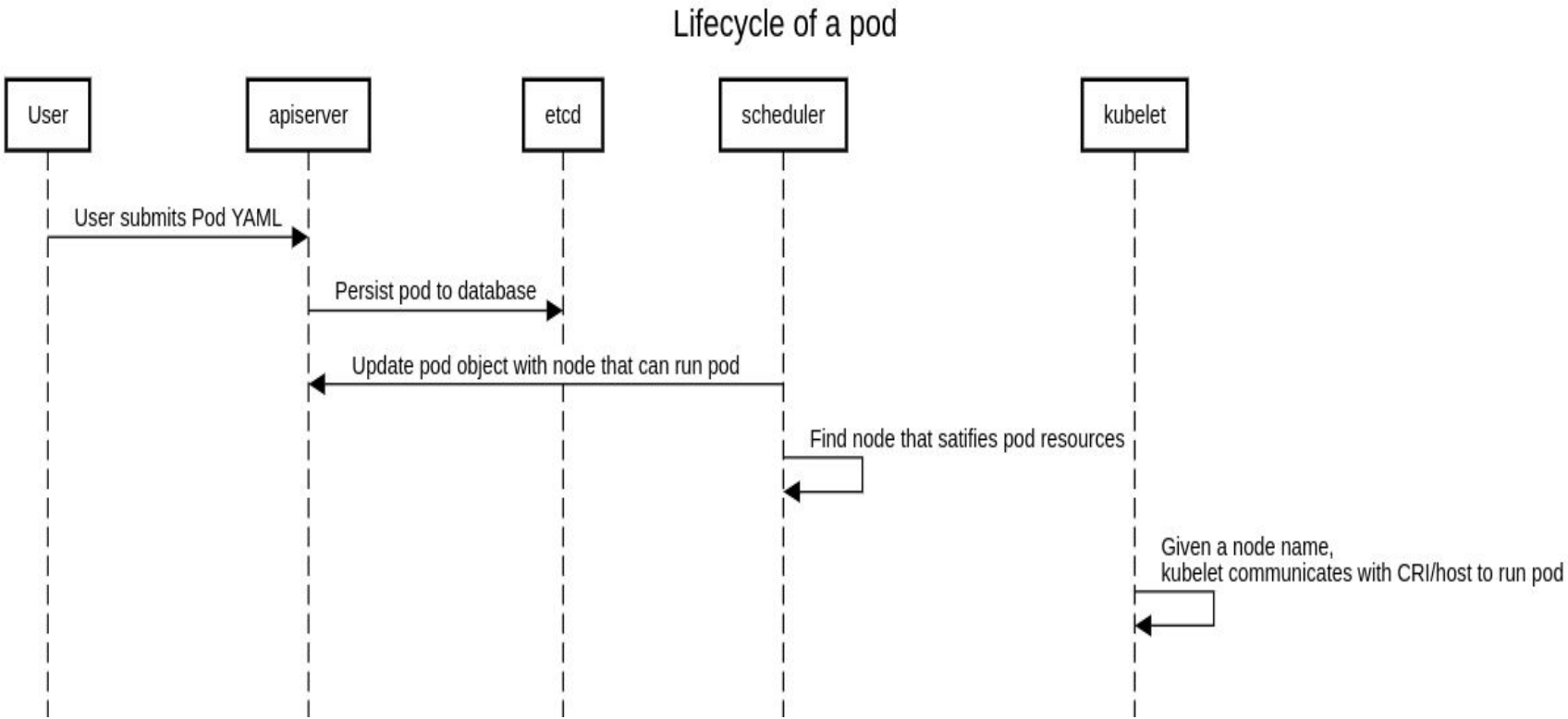
Not Covered :(

- Maximing performance
- Storage
- Networking
- Cloudfu

What is Kubernetes

- Toolkit for building distributed systems
  - What can fail will fail!
- Declarative API for workloads
- Everything starts with the YAML!
  - API first

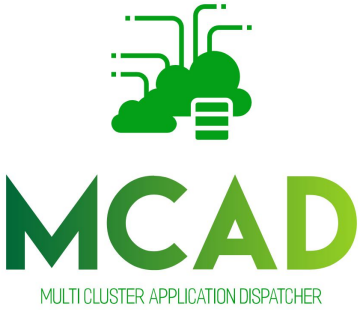

Kubernetes cluster
c-m
c-c-m
Cloud provider API
api
etcd
sched
Control Plane
Node
Node
Node
kubelet
kubelet
kubelet
k-proxy
k-proxy
k-proxy
API server
Cloud controller manager (optional)
Controller manager
etcd (persistence store)
kubelet
kube-proxy
Scheduler
Control plane
Node


Lifecycle of a pod
User
apiserver
etcd
scheduler
kubelet
User submits Pod YAML
Persist pod to database
Update pod object with node that can run pod
Find node that satifies pod resources
Given a node name,
kubelet communicates with CRI/host to run pod

Why you shouldn’t use a direct Pod

- Stateless
- No Self healing
  - Failures happen
- API difficult to change
- No queuing!

Open source tools for Batch workloads



Existing Jobs

Job	Project	Components
TensorFlowJob	Kubeflow	Pod + Service
PyTorchJob	Kubeflow	Pod + Service
MPIJobV1	Kubeflow	Pod + Service
MPIJobV2	Kubeflow	IndexedJob + Headless Service (for launcher) and Pod + Service for workers
TrainingOperator	Kubeflow	Pod + Service
PodSpec	Armada	Pod + Service + Ingress
vcJob	Volcano	Pod

Working Group Batch

Scope

Discuss and enhance the support for Batch (eg. HPC, AI/ML, data analytics, CI) workloads in core Kubernetes. We want to unify the way users deploy batch workloads to improve portability and to simplify supportability for Kubernetes providers.

In scope

- To reduce fragmentation in the k8s batch ecosystem: congregate leads and users from different external and internal projects and user groups (CNCF TAGs, k8s sub-projects focused on batch-related features such as topology-aware scheduling) in the batch ecosystem to gather requirements, validate designs and encourage reutilization of core kubernetes APIs.
- The following recommendations for enhancements:
  - Additions to the batch API group, currently including Job and CronJob resources that benefit batch use cases such as HPC, AI/ML, data analytics and CI.
  - Primitives for job-level queueing, not limited to the k8s Job resource. Long-term, this could include multi-cluster support.
  - Primitives to control and maximize utilization of resources in fixed-size clusters (on-prem) and elastic clusters (cloud).
  - Runtime and scheduling support for specialized hardware (GPUs, NUMA, RDMA, etc.)

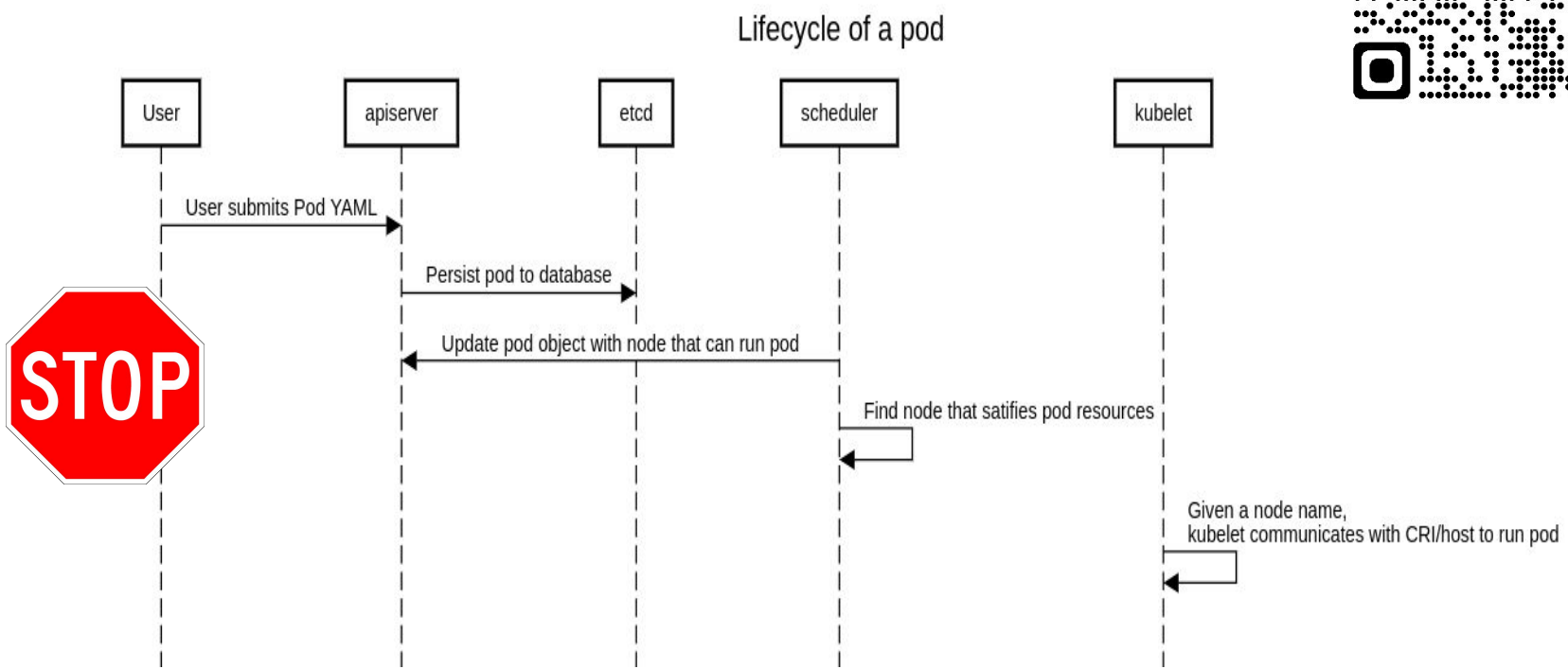
Job API

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  parallelism: 1
  completions: 1
  activeDeadlineSeconds: 1800
  backoffLimit: 6
  template:
    metadata:
      name: pi
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: OnFailure
```

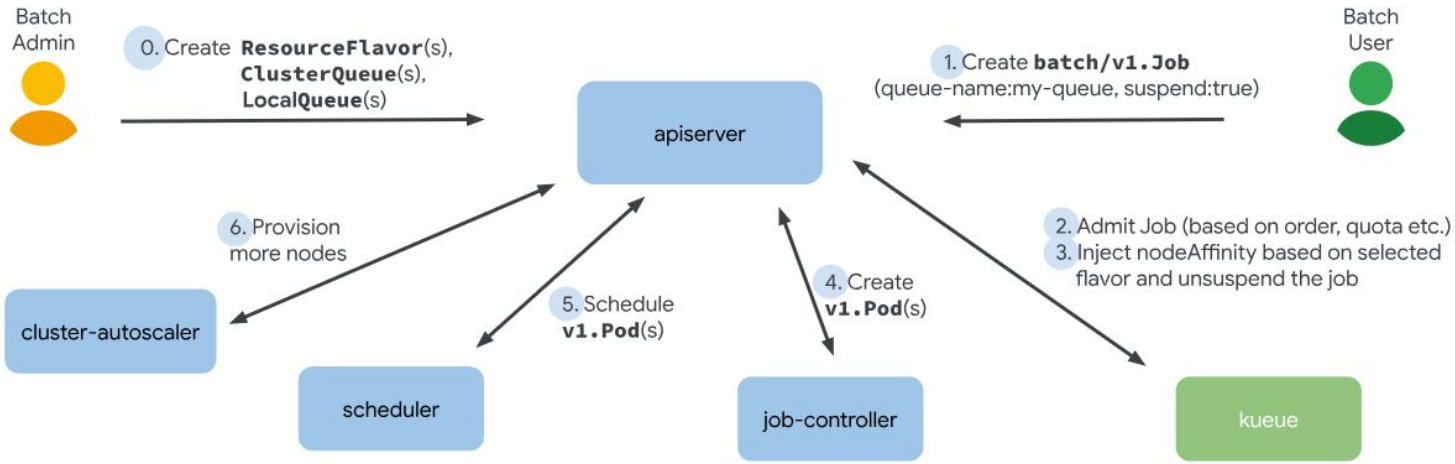
IndexedJob + Headless service

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: 'sample-job'
spec:
 completions: 3
 parallelism: 3
 completionMode: Indexed
 template:
   spec:
     restartPolicy: Never
     containers:
     - command:
       - 'bash'
       - '-c'
       - 'echo "My partition: ${JOB_COMPLETION_INDEX}"'
       image: 'docker.io/library/bash'
       name: 'sample-load'
```

Suspend


Lifecycle of a pod
User
apiserver
etcd
scheduler
kubelet
User submits Pod YAML
Persist pod to database
Update pod object with node that can run pod
Find node that satifies pod resources
Given a node name,
kubelet communicates with CRI/host to run pod
STOP

High-level Kueue operation



Kueue Supported Jobs

- Design requires suspend field or pod support
- Supported jobs:
  - KubeRay
  - Kubeflow operators
  - JobSet
  - Job
  - FluxMiniCluster

JobSet

- 2023 Kubernetes Project for defining a API for multiple jobs
- Support lifecycle operators around DistributedJobs
- Use IndexedJob + HeadlessService
- Failure + Success Policies for replicated jobs

Pytorch

```yaml
# Distributed training of a ResNet18 model to do image classification
# using the CIFAR-10 dataset and PyTorch.
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: pytorch
spec:
  replicatedJobs:
  - name: workers
    template:
      spec:
        parallelism: 4
        completions: 4
        backoffLimit: 0
        template:
          spec:
            containers:
            - name: pytorch
              image: gcr.io/k8s-staging-jobset/pytorch-resnet:latest
              ports:
              - containerPort: 3389
              env:
              - name: MASTER_ADDR
                value: "pytorch-workers-0-0.pytorch"
              - name: MASTER_PORT
                value: "3389"
              # Force python to not buffer output and write directly to stdout, so we can view training logs via `kubectl logs`.
              - name: PYTHONUNBUFFERED
                value: "0"
              command:
              - bash
              - -xc
              - |
                torchrun --nproc_per_node=1 --master_addr=$MASTER_ADDR --master_port=$MASTER_PORT resnet.py --backend=gloo --num_epochs=2
```

```yaml
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: success-policy
spec:
# We want to declare our JobSet successful if workers finish.
# If workers finish we should clean up the remaining replicatedJobs.
  successPolicy:
    operator: All
    targetReplicatedJobs:
    - workers
  replicatedJobs:
  - name: leader
    replicas: 1
    template:
      spec:
        # Set backoff limit to 0 so job will immediately fail if any pod fails.
        backoffLimit: 0
        completions: 1
        parallelism: 1
        template:
          spec:
            containers:
            - name: leader
              image: bash:latest
              command:
              - bash
              - -xc
              - |
                sleep 10000
  - name: workers
    replicas: 1
    template:
      spec:
        backoffLimit: 0
        completions: 2
        parallelism: 2
        template:
          spec:
            containers:
            - name: worker
              image: bash:latest
              command:
              - bash
              - -xc
              - |
                sleep 10
```

Leader + Workers

Demo!

To install JobSet:

```
VERSION=v0.3.1
kubectl apply --server-side -f https://github.com/kubernetes-sigs/jobset/releases/download/$VERSION/manifests.yaml
```

To install Kueue:

```
VERSION=v0.5.2
kubectl apply --server-side -f https://github.com/kubernetes-sigs/kueue/releases/download/$VERSION/manifests.yaml
```

Thank you!