Revolutionizing Kube Scalability Testing with KWOK

Wei Huang, Weiwei Yang, Apple

Wei Huang

Co-chair of sig-scheduling

Maintainer of scheduler-plugins

Maintainer of kwok

Weiwei Yang

ASF member

Ex-VP of Apache YuniKorn

❤️ Kubernetes & Open source

Agenda

- Kubernetes scalability testing overview
- KWOK - a bird's eye view
- How KWOK "wow!" YuniKorn's scalability testing

Agenda

- **Kubernetes scalability testing overview**
- KWOK - a bird's eye view
- How KWOK "wow!" YuniKorn's scalability testing

Kubernetes scalability testing overview

- What does scalability testing mean in Kubernetes 🤯
  - How a component responds upon the 📈 # of Kubernetes API objects
- A day-2️⃣ must
  - Performance assurance
  - Capacity planning
  - Cost efficiency
  - User experience

A (over-)simplified paradigm

workload abstraction & orchestration

- ClusterLoader2(CL2)
- scheduler-perf

metric gathering, analysis and display

- Prometheus metrics
- Grafana dashboard
- Custom dashboard

(https://perf-dash.k8s.io/)

Real Kubernetes cluster

💰 & ⏳ & ⚙️

```
59 # This is a sig-release-master-blocking job.
60 # The frequency was cut to reduce infrastructure costs.
61 - cron: '1 17 1-31/2 * *' # Run on odd days at 9:01PST (17:01 UTC)
62   name: ci-kubernetes-e2e-gce-scale-performance
63   tags:
64   - "perfDashPrefix: gce-5000Nodes"
65   - "perfDashBuildsCount: 270"
66   - "perfDashJobType: performance"
67   cluster: k8s-infra-prow-build
68   labels:
69     preset-service-account: "true"
70     preset-k8s-ssh: "true"
71     preset-e2e-scalability-common: "true"
72     preset-e2e-scalability-periodics: "true"
73     preset-e2e-scalability-periodics-master: "true"
'config/jobs/kubernetes/sig-scalability/sig-scalability-release-blocking-jobs.yaml"
```

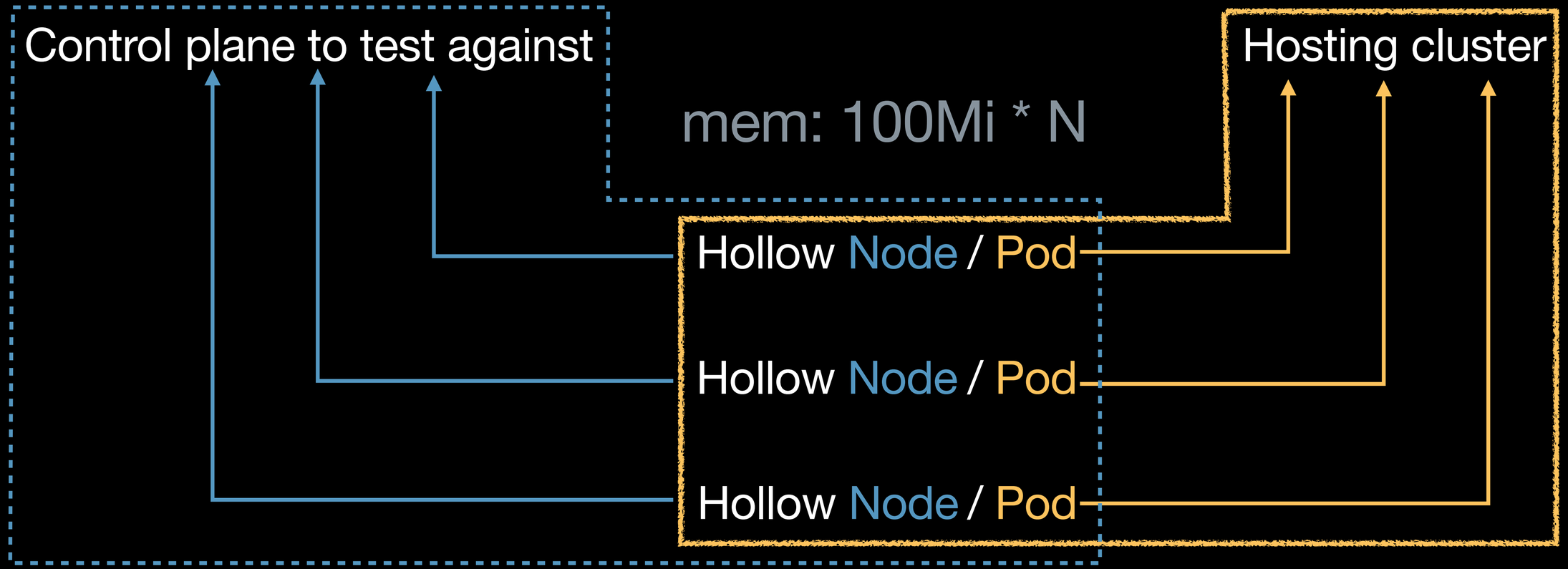
```
 97       args:
 98       - --cluster=gce-scale-cluster
 99       - --env=HEAPSTER_MACHINE_TYPE=e2-standard-32
100       # TODO(mborsz): Adjust or remove this change once we understand coredns
101       # memory usage regression.
102       - --env=KUBE_DNS_MEMORY_LIMIT=300Mi
103       - --extract=ci/latest-fast
104       - --extract-ci-bucket=k8s-release-dev
105       - --gcp-nodes=5000
106       - --gcp-project-type=scalability-scale-project
107       - --gcp-zone=us-east1-b
108       - --provider=gce
109       - --metadata-sources=cl2-metadata.json
110       - --env=CL2_LOAD_TEST_THROUGHPUT=50
111       - --env=CL2_DELETE_TEST_THROUGHPUT=50
112       - --env=CL2_RATE_LIMIT_POD_CREATION=false
113       - --env=KUBE_CONTROLLER_MANAGER_TEST_ARGS=--endpointslice-updates-batch-period=500ms --endpoint-updates-batch-period=500ms
114       # Overrides CONTROLLER_MANAGER_TEST_ARGS from preset-e2e-scalability-periodics.
115       - --env=CONTROLLER_MANAGER_TEST_ARGS=--profiling --contention-profiling --kube-api-qps=100 --kube-api-burst=100
```

Source: https://github.com/kubernetes/test-infra/blob/master/config/jobs/kubernetes/sig-scalability/sig-scalability-release-blocking-jobs.yaml

Simulated Kubernetes cluster

- Kubemark (Kube~~rnetes bench~~mark)
- A hollow-node binary implements ***minimum*** Kubelet interface
- **N** hollow-nodes register themselves as Nodes to represent a **N-nodes** cluster

Kubemark


Control plane to test against
mem: 100Mi * N
Hosting cluster
Hollow Node / Pod
Hollow Node / Pod
Hollow Node / Pod

Agenda

- Kubernetes scalability testing overview
- **KWOK - a bird's eye view**
- How KWOK "wow!" YuniKorn's scalability testing

KWOK - a bird's eye view

- What is KWOK?
- How it is distinguished compared to other frameworks like Kubemark?
- Is it performant? How?

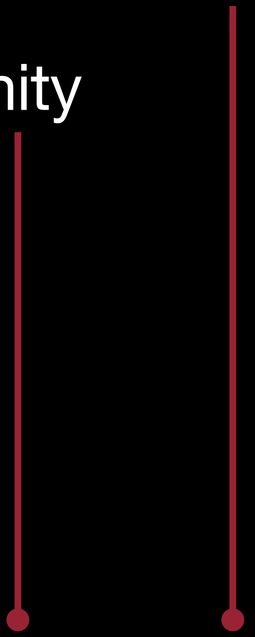
KWOK

Kubernetes WithOut Kubelet

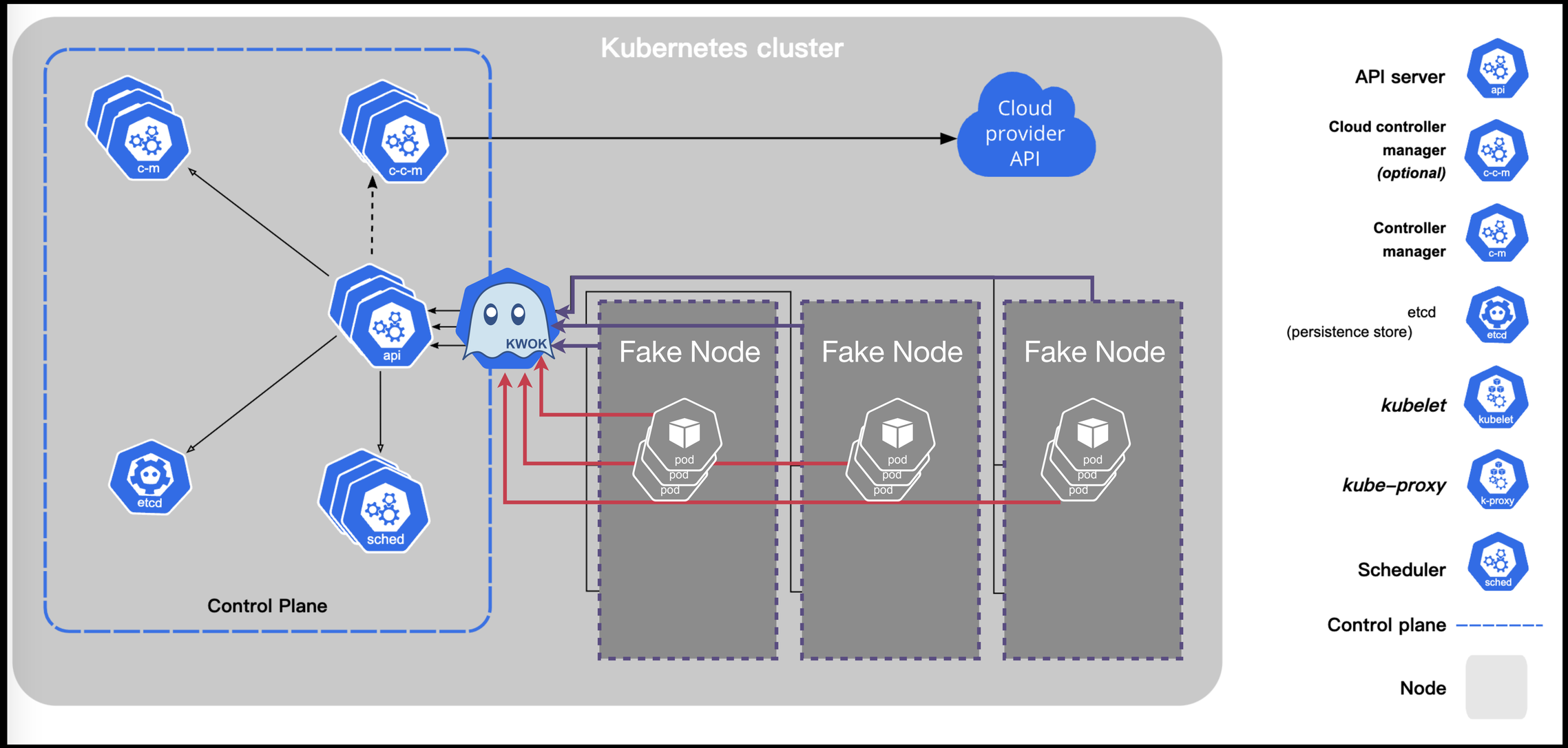
~~Kubelet,~~ what's the point?

- Integration test framework in Kubernetes community
- `envtest` in the Kubebuilder community
- `hollow-node` in Kubemark

O(n) Fake Nodes; Memory footprint



SDK-oriented; Virtual Nodes; Pending Pods



Source: https://kubernetes.io/docs/concepts/overview/components/

Is it performant? How?

- Native Go routines vs. Informer
  - Initial impl. was a *VirtualKubelet* provider
- O(1) memory footprint by design

Components in KWOK

kwok controller - core

- Maintain heartbeats of Nodes
- Simulate lifecycle of Pods/Nodes
- Tailored design to optimize memory footprint

kwokctl - CLI tool

- Blazing fast startup speed
- Spin-up a bare-bone control plane + kwok controller
- Manage lifecycle of clusters created via kwokctl
- A series of tools to dump/restore cluster's snapshot

Demo (part 1)

Agenda

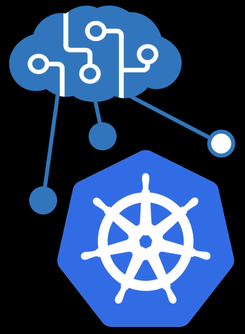
- Kubernetes scalability testing overview
- KWOK - a bird's eye view
- **How KWOK "wow!" YuniKorn's scalability testing**

What is Apache YuniKorn

Apache YuniKorn is a standalone Kubernetes scheduler that brings **multi-tenancy** readiness to the Kubernetes clusters. Widely used to schedule large scale data processing, analytics, and AIML workloads.

Scheduling capabilities

Hierarchy queues, resource fairness, job ordering (FIFO, Fair, Priority), job preemption. Gang scheduling, etc



K8s scheduler alternative

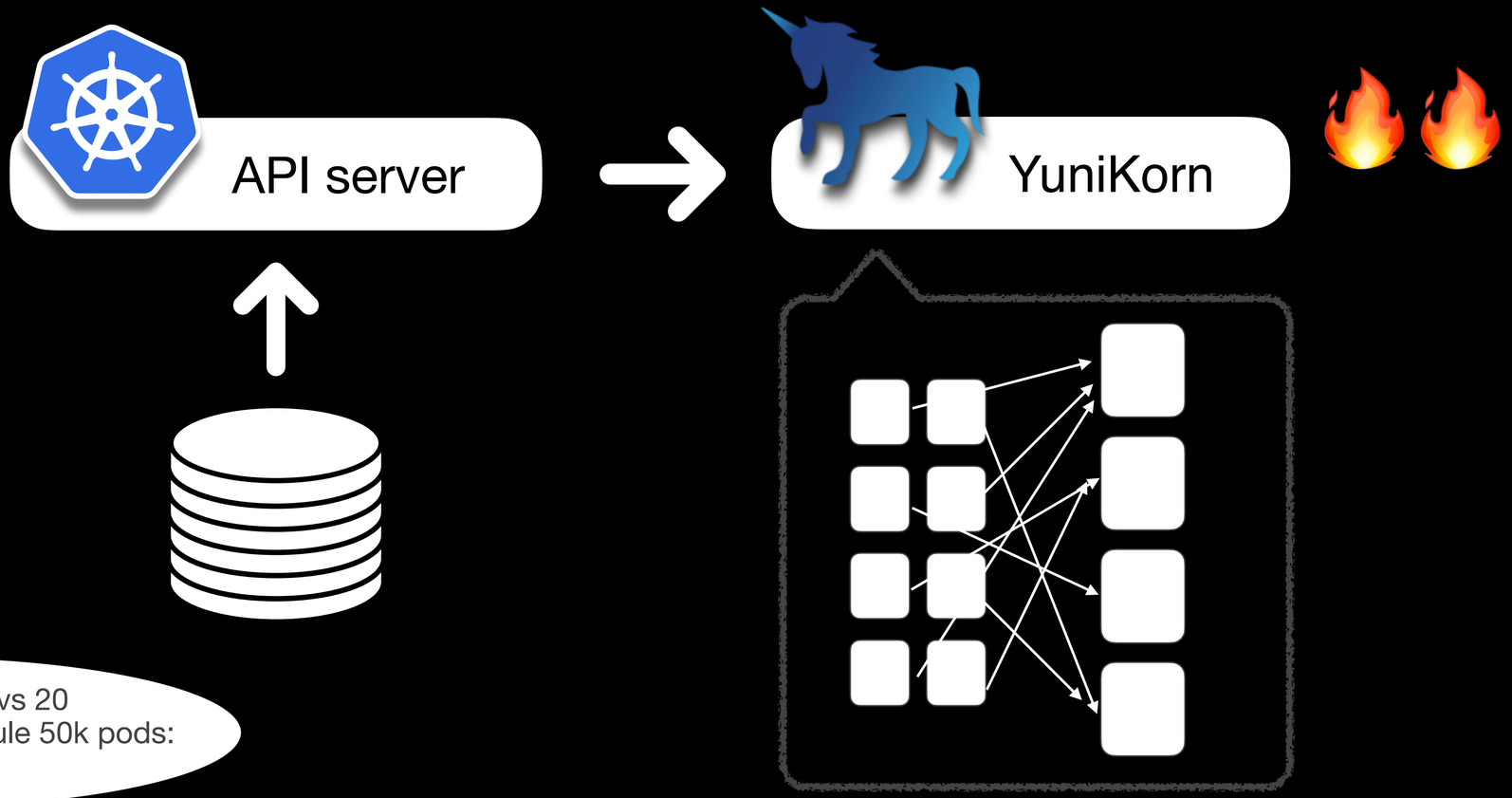
Fully compatible with Kubernetes, an alternative to the default scheduler, but more powerful. Transparent for the existing K8s applications. Foundation of a multi-tenancy cluster.

Cloud native

Seamlessly work with cloud autoscalers, easily integrated with public cloud vendors. Support private, public and hybrid cloud use cases.

Why performance testing is important

- Stretch to the unknown
  - Build confidence for the future scale
- Increase the throughput makes huge difference
  - Cost efficiency
  - Reduce the latency

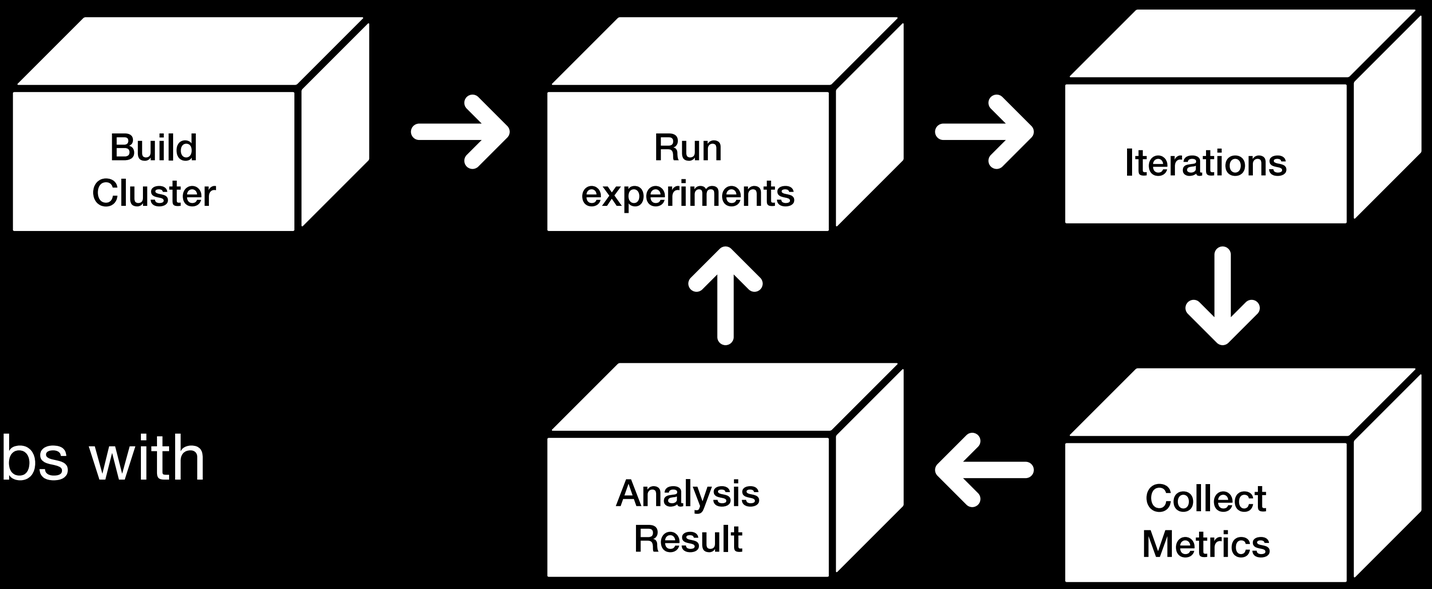


Throughput: 10 pod/s vs 20 pod/s, time spent to schedule 50k pods: 80m vs 40m.

😲

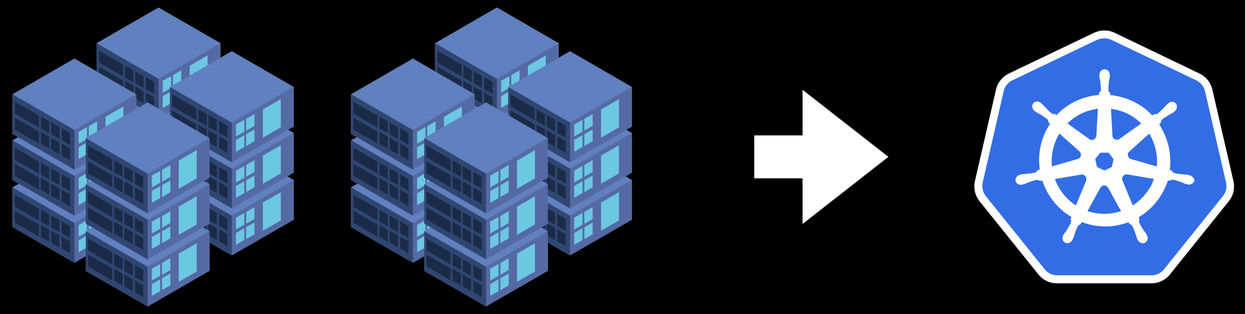
Steps: performance testing on Kubernetes

Apache YuniKorn
community approach:



- Setup cluster: **Kubemark**
- Developed a tool to launch jobs with different configs
- Developed a tool to collect metrics and draw charts

Problems with Kubemark

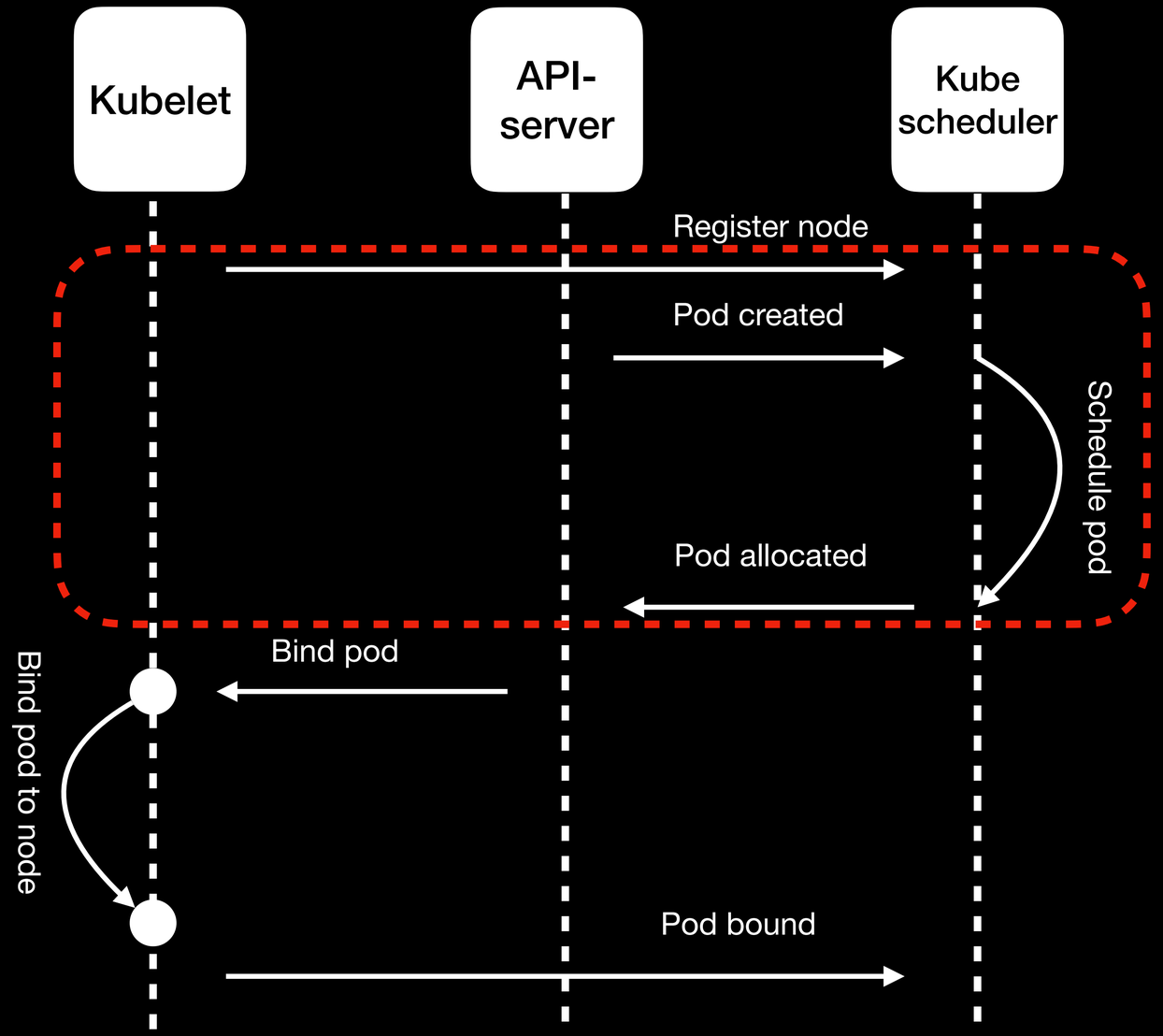


Kubemark based solution is too expensive:

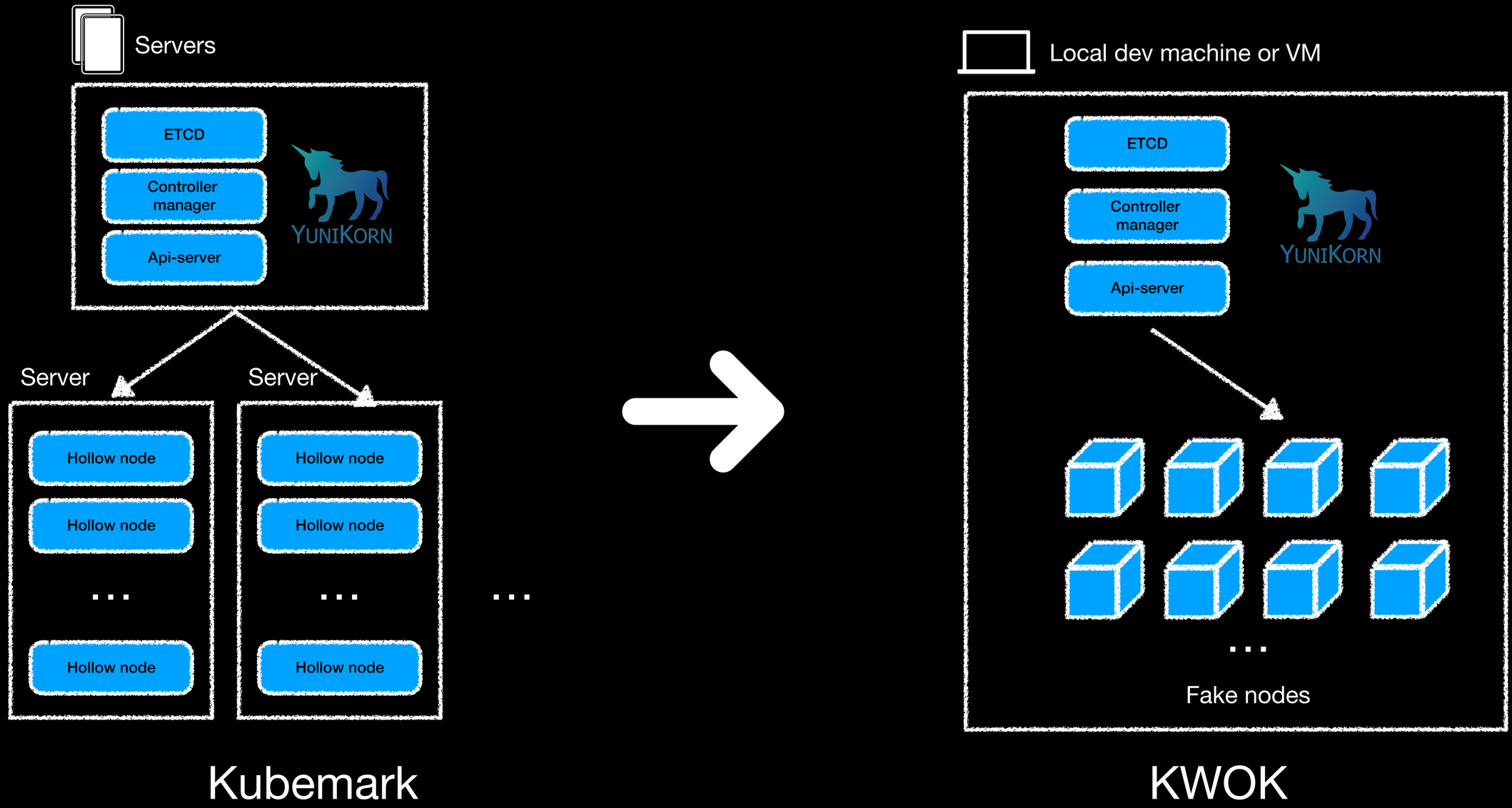
- ~20 physical servers to simulate 10k nodes, 100k pods
- Difficult to setup, need to tweak each server's system level configs
- One month turnover time
- Extremely difficult to automate

Spotlight - the scheduler performance

- How fast the scheduler can allocate pod to a node?
- Less concern about pod binding phase
- Need real control-plane, but require light-weighted "Kubelet" to scale

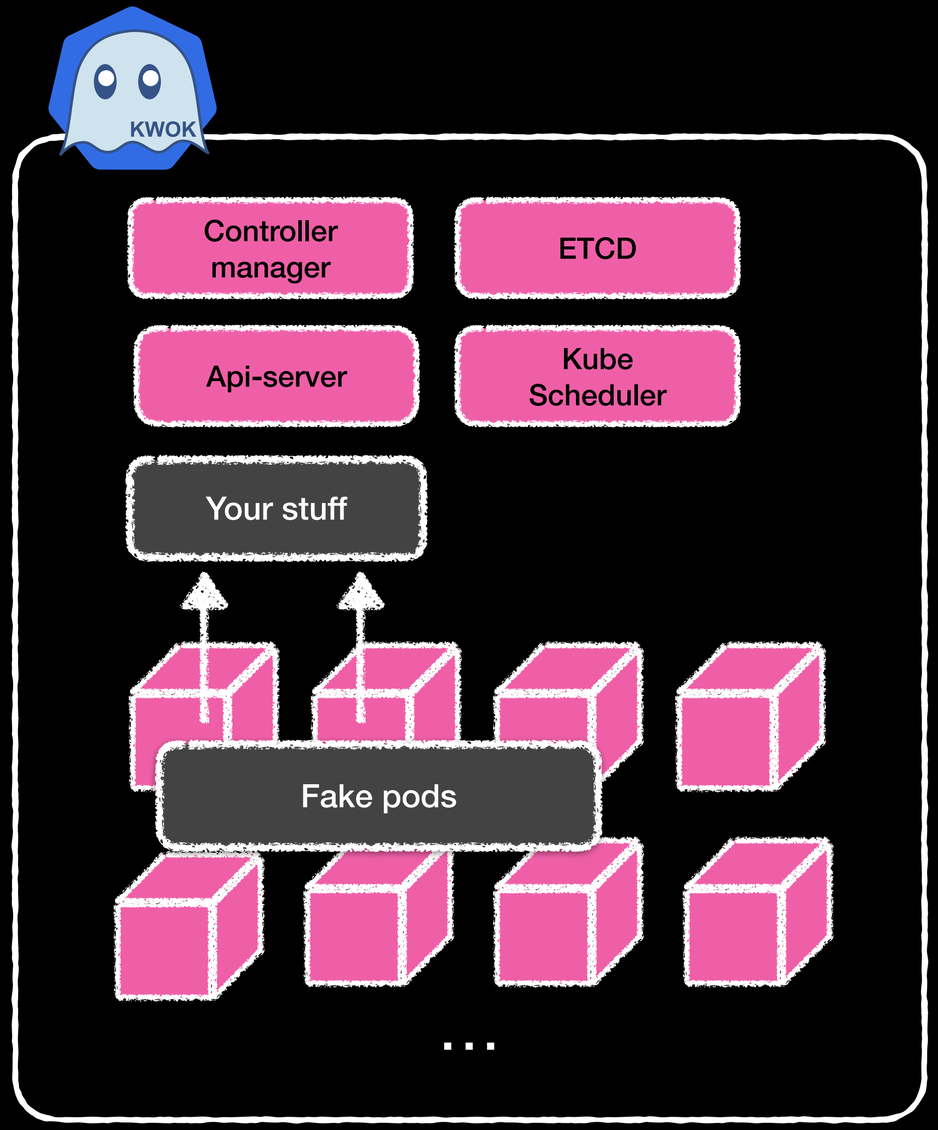


KWOK simplifies everything



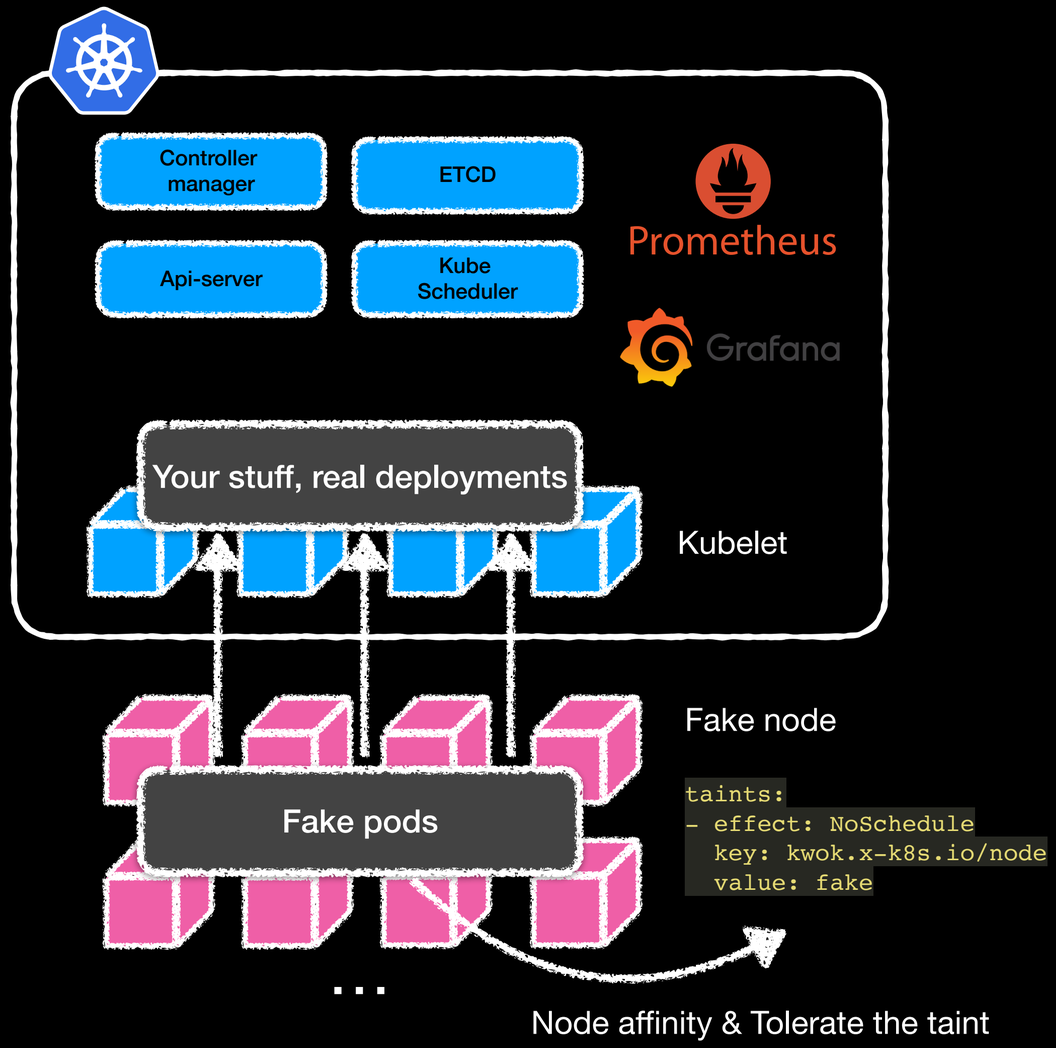
Kwok managed cluster

- Kwokctl brings up Kubernetes control-plane components locally:
  - Runtime choices: binary, container, Kind.
- Kwok simulates any number of nodes, and attach them to the cluster



Bring your own cluster to Kwok

- Real Kubernetes cluster with real Kubelet nodes
  - Local cluster: Kind/Minikube/docker-desktop…
  - Remote cluster: EKS, GKE, …
- Kwok simulates any number of nodes, and attach them to the existing cluster



Environment setup

- Setup Kubernetes (Kwok managed or Bring your own)
- Install Kwok and create nodes
- Install your app stack (YuniKorn, Prometheus, Grafana)
- 🎉 Start exploring 🙌

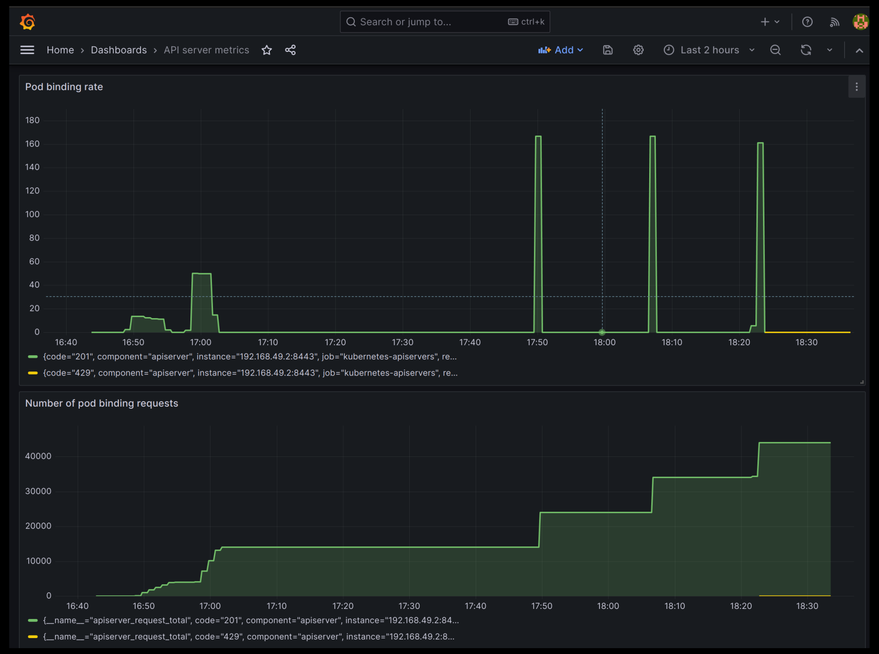
⚠️ Tips: Set bigger cpu/memory if you are using minikube; increase api-server, controller-manager QPS, reduce throttling.

🥁 Create nodes with different specs you wanted, resources, labels, annotations, etc.

Testing and metrics collecting

```
rate(apiserver_request_total{resource="pods",
subresource="binding", verb="POST"}[1m])
```

```
apiserver_request_total{resource="pods",
subresource="binding", verb="POST"}
```



https://issues.apache.org/jira/browse/YUNIKORN-1969

Evaluating YuniKorn performance using KWOK

Kwok suitable use cases

Development & local testing

Local development environment that easily reaches to hundreds of nodes,
Fast iterations before testing in real clusters.

Performance evaluation & tuning

Instantly launch large scale cluster, facilitate the local debugging
for scalability and performance issues.

Chaos engineering

Random inject failures to nodes and pods, create chaos engineering
scenarios without a real cluster.

CI/CD pipeline

Integrate with GitHub Actions to iterate performance testing,
Avoid performance regression.

Demo (part 2)

Where do I go next

- https://kwok.sigs.k8s.io/
- Kubernetes Slack Channels: #kwok, #kwok-dev
- Other KubeCon Talks
  - Best Practices: Improving Batch Scheduling Performance at Scale Using MCAD and KWOK
  - Running Large-Scale Scheduling Simulations with Virtual Kubelet

Thanks!