PodOverhead: Accounting for Greater Cluster Stability

Eric Ernst, Intel, @egernst

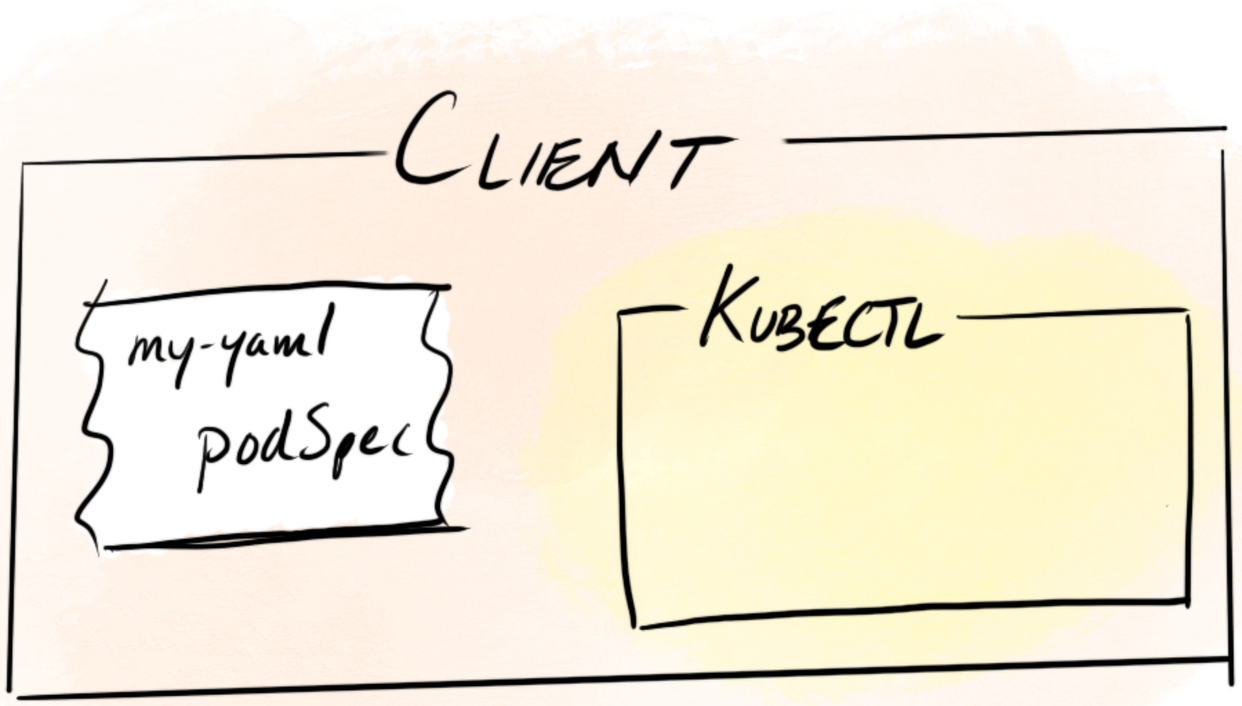
SW Engineer
kat herder
kata containers arch committee
k8s contributor

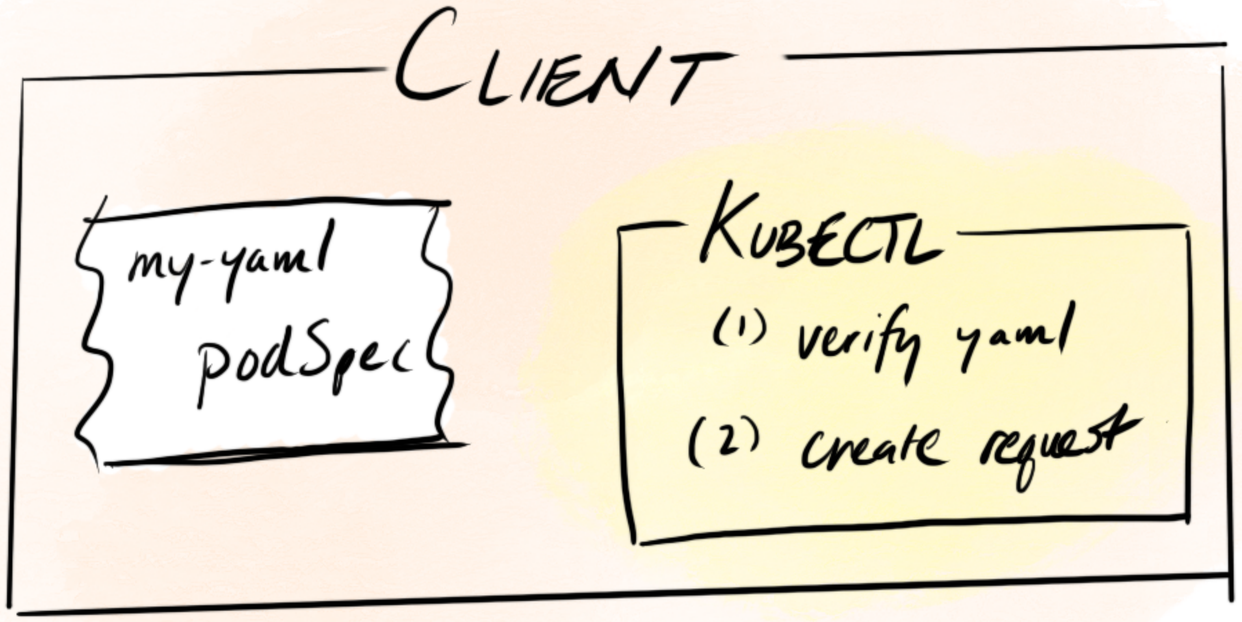
What happens when you click enter after typing kubectl apply for a pod?

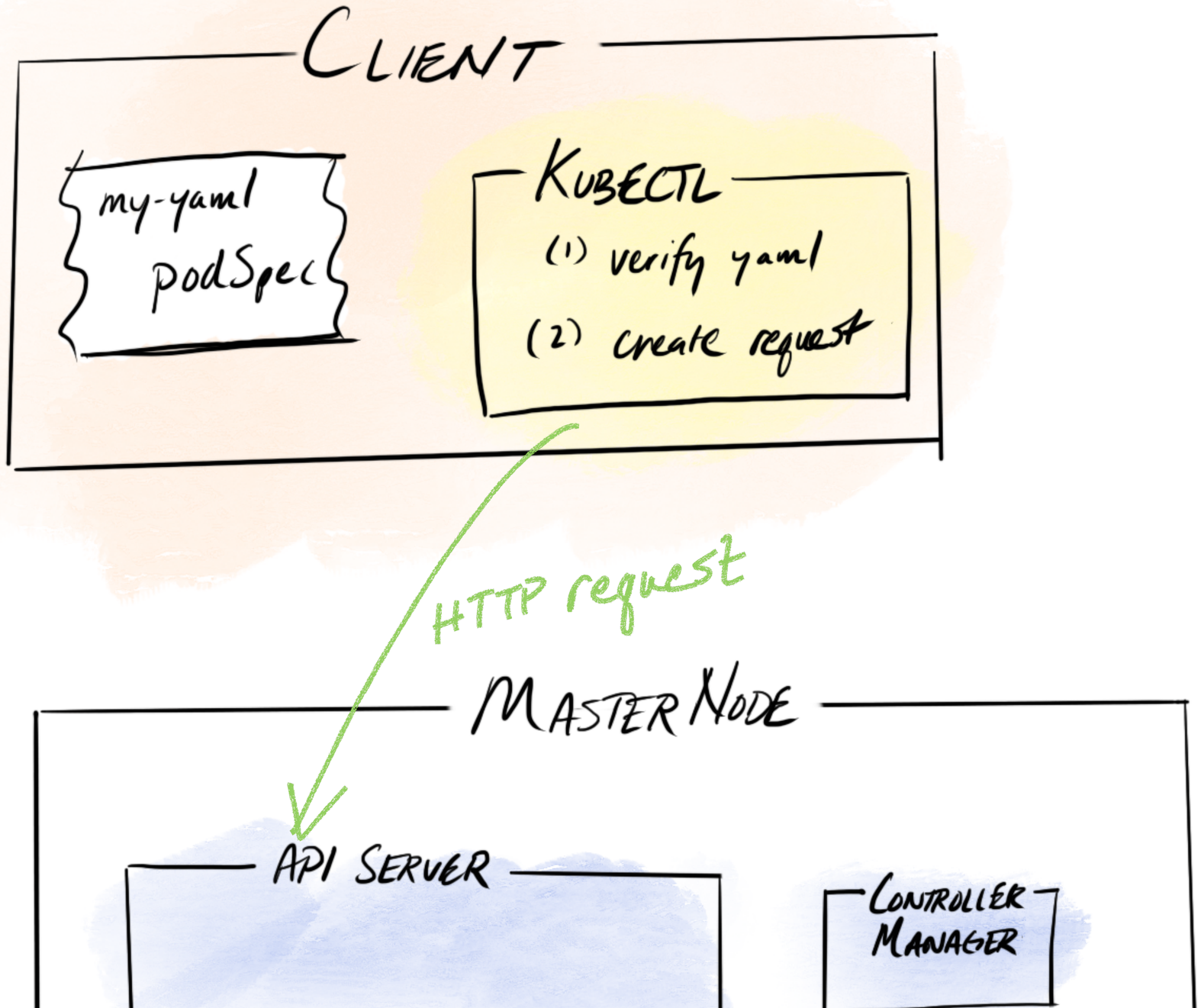
Where and what are these overheads?

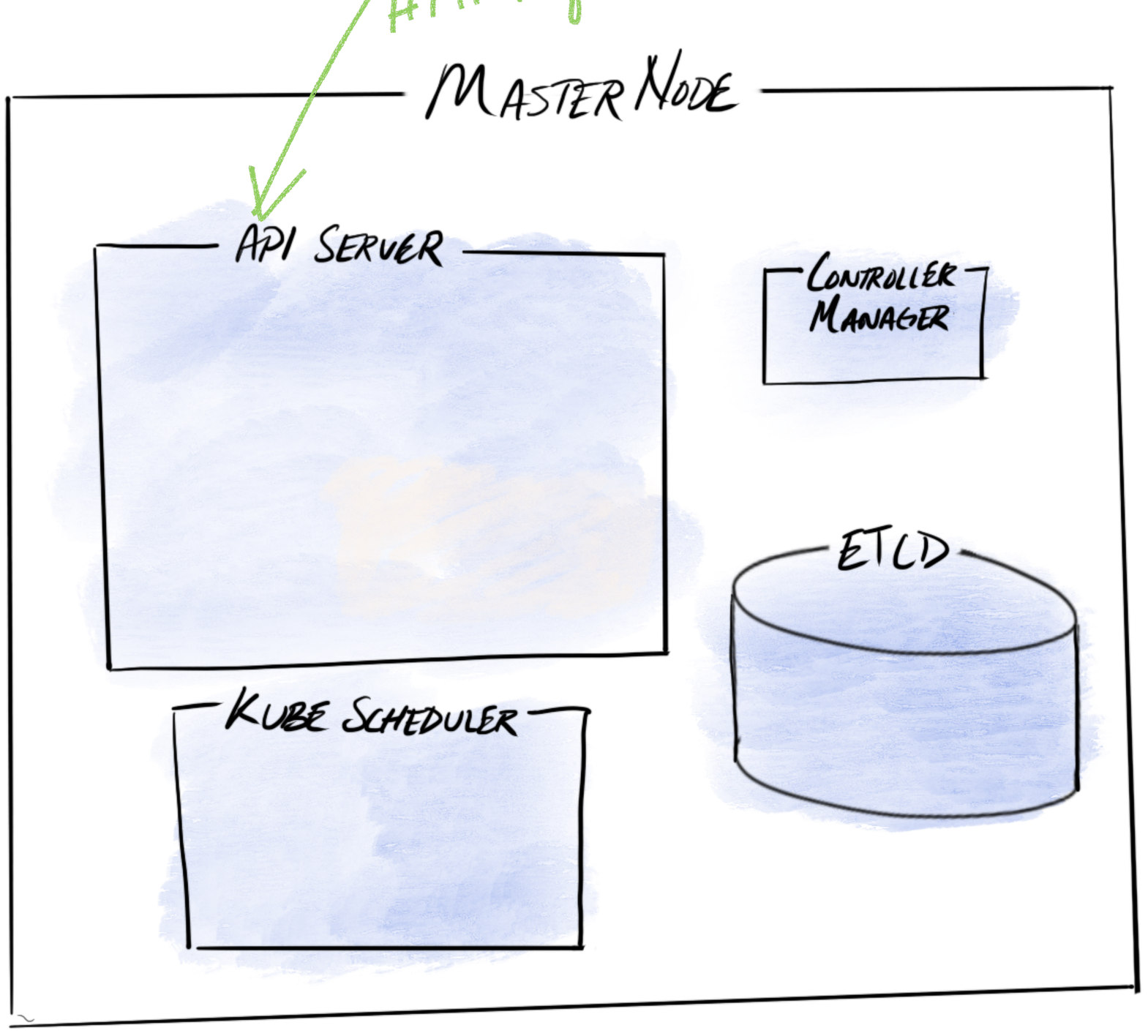
What is the PodOverhead feature?

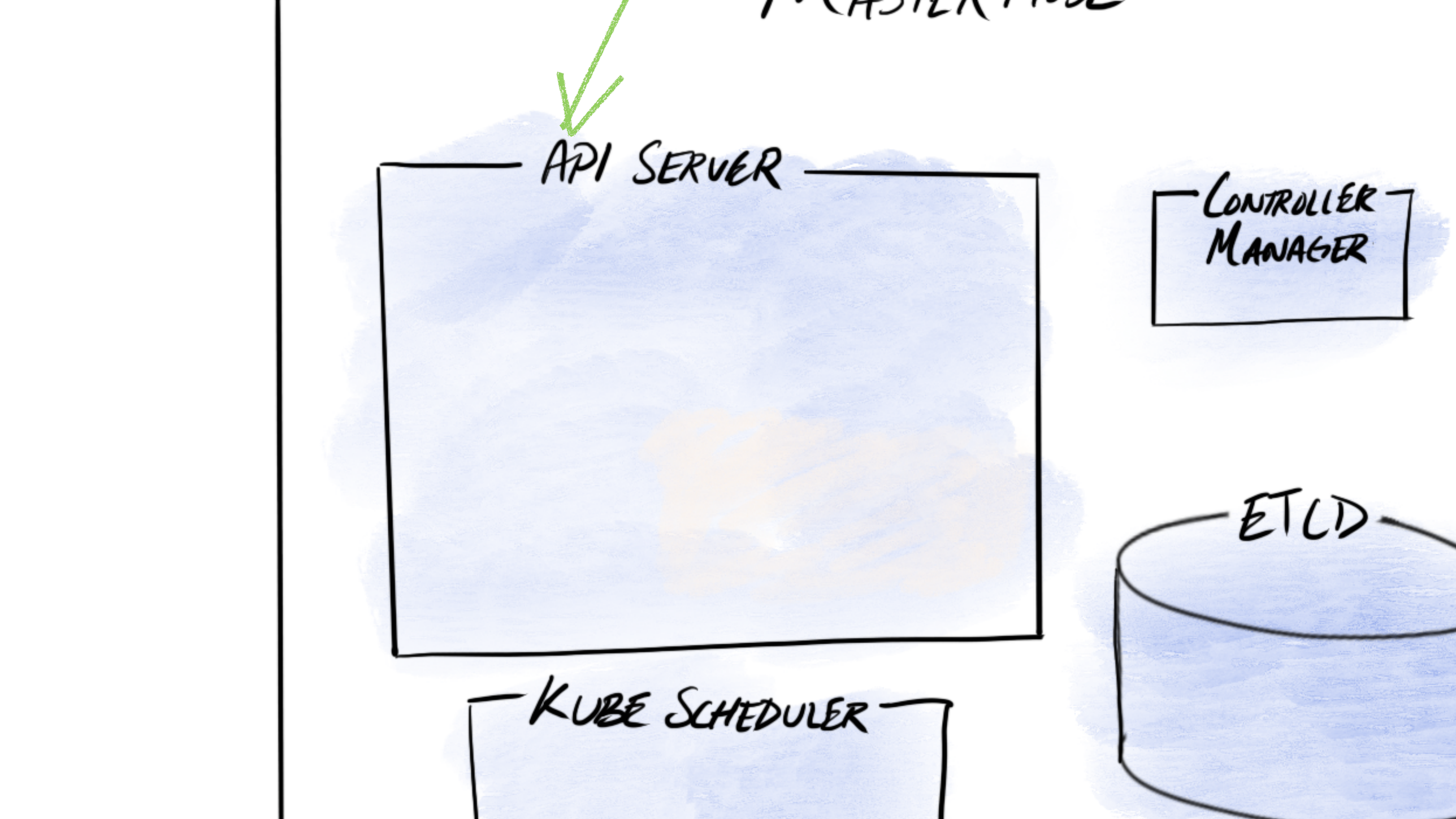
Why should you care?

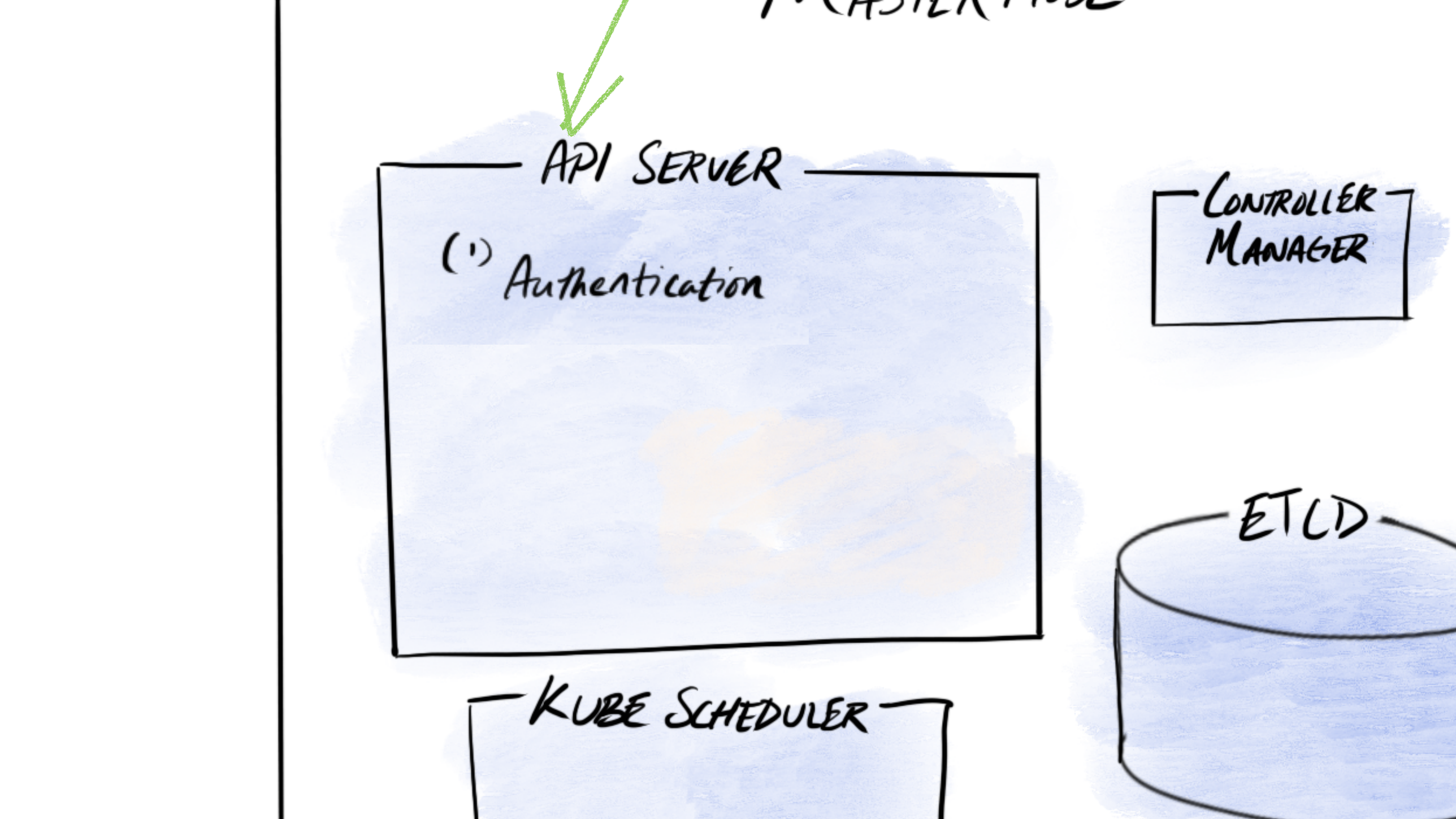

CLIENT
my-yaml
podSpec
KUBECTL

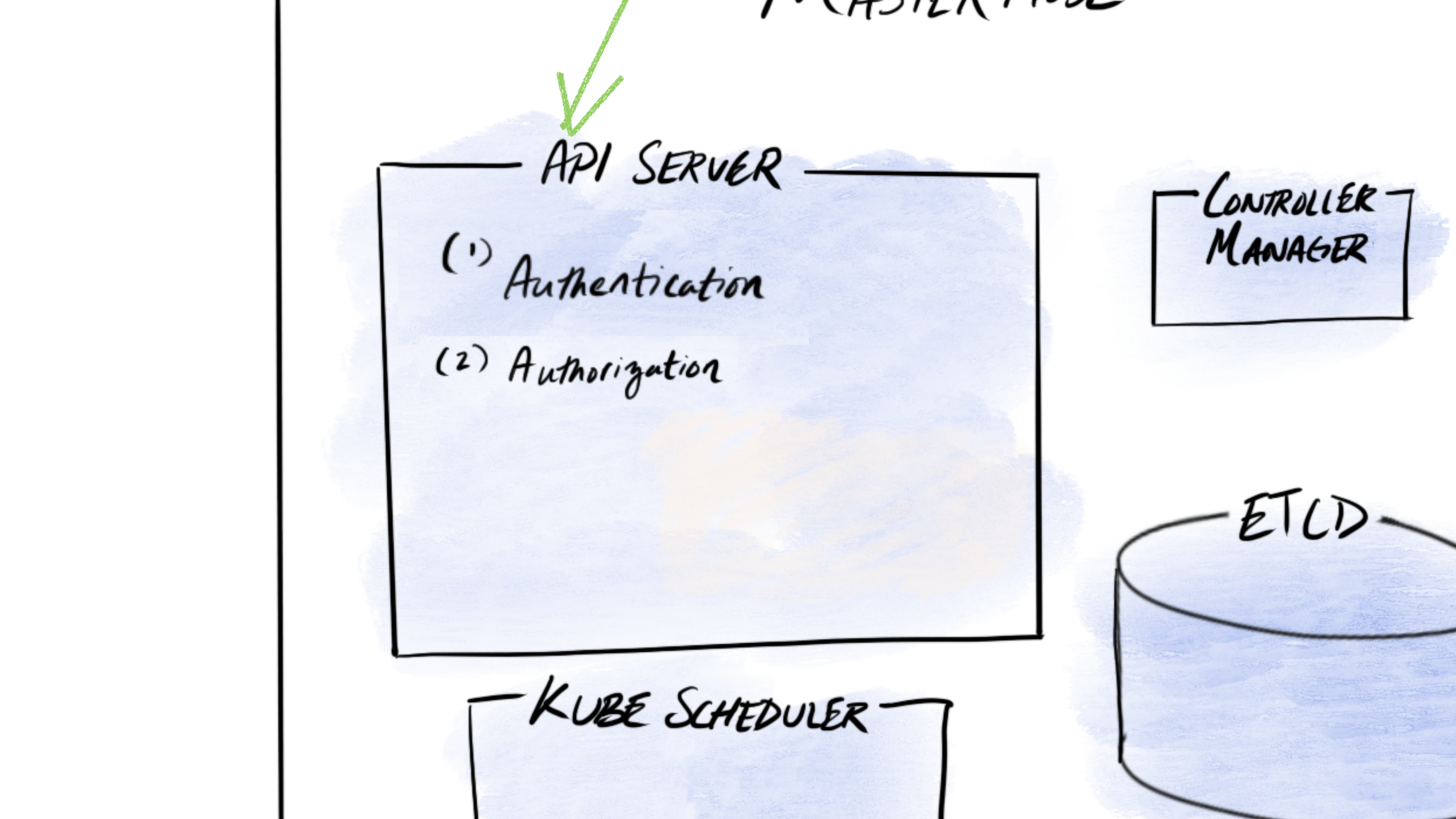

CLIENT
my-yaml
podSpec
KUBECTL
(1) verify yaml
(2) create request

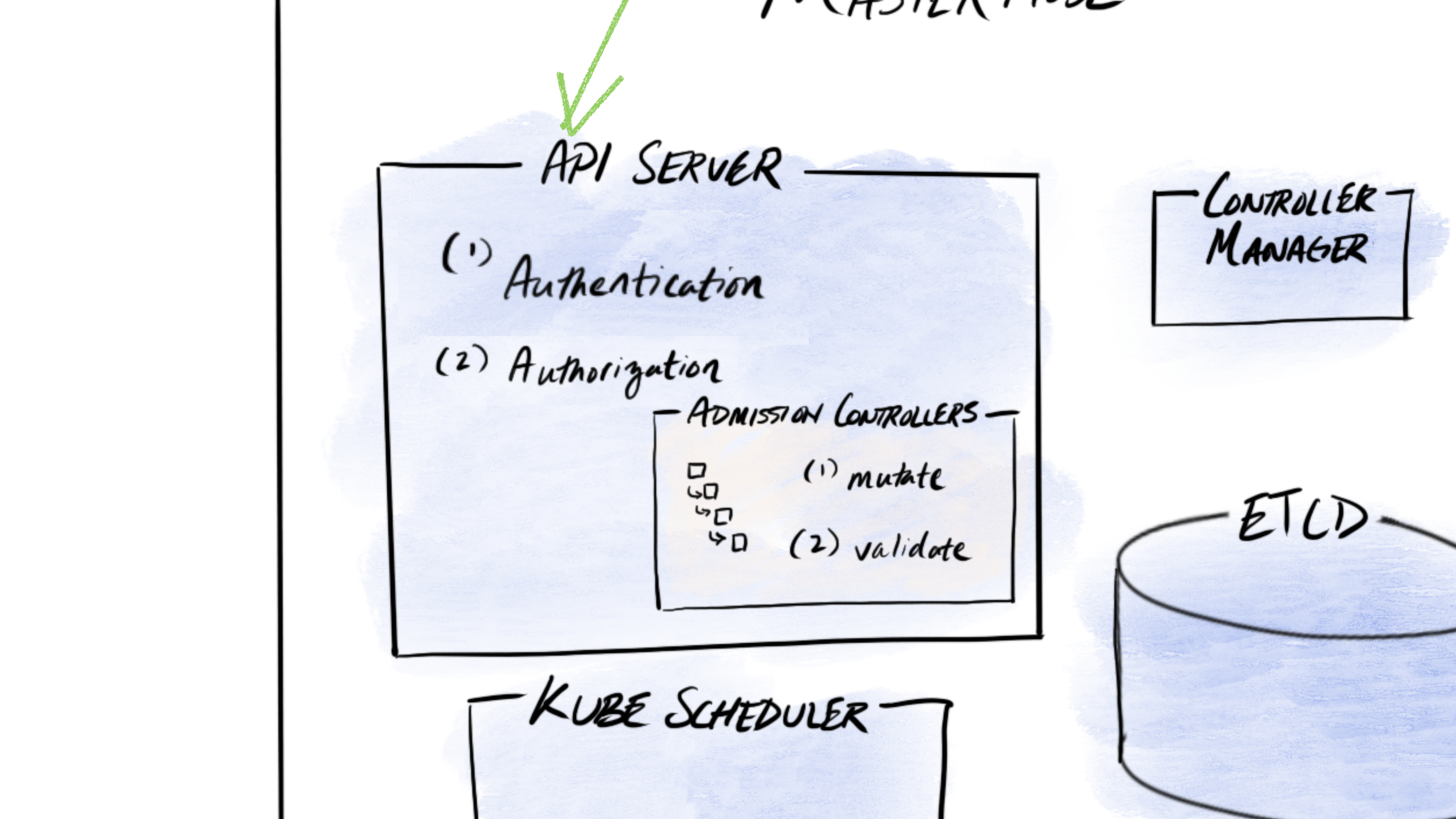

CLIENT
my-yaml
podSpec
KUBECTL
(1) verify yaml
(2) create request
HTTP request
MASTER NODE
API SERVER
CONTROLLER MANAGER

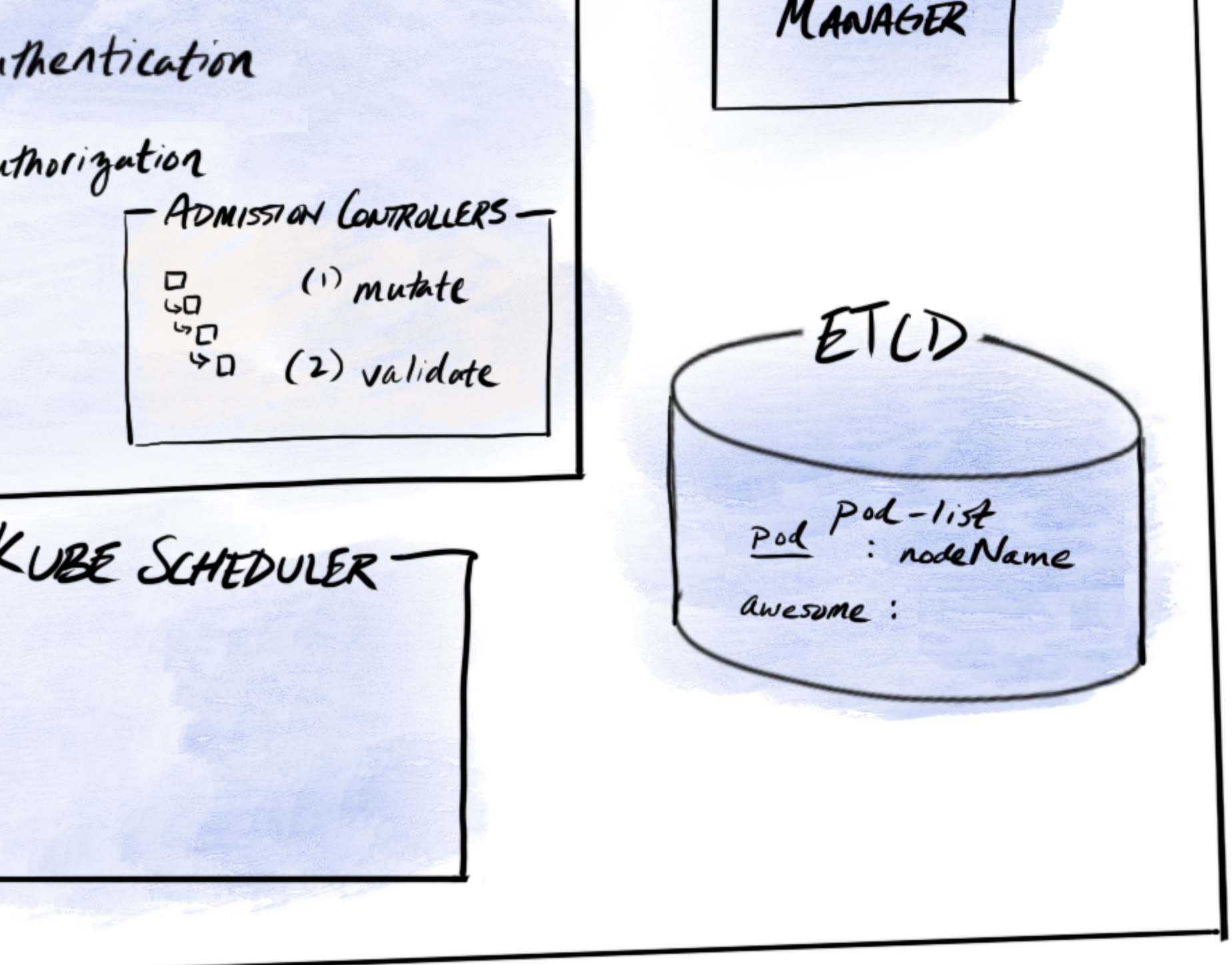

MASTER NODE
API SERVER
CONTROLLER MANAGER
ETCD
KUBE SCHEDULER


API SERVER
CONTROLLER MANAGER
ETCD
KUBE SCHEDULER


API SERVER
Authentication
CONTROLLER MANAGER
ETCD
KUBE SCHEDULER


API SERVER
(1) Authentication
(2) Authorization
CONTROLLER MANAGER
ETCD
KUBE SCHEDULER


API SERVER
(1) Authentication
(2) Authorization
ADMISSION CONTROLLERS
(1) mutate
(2) validate
CONTROLLER MANAGER
ETCD
KUBE SCHEDULER


thentication
uthorization
ADMISSION CONTROLLERS
(1) mutate
(2) validate
MANAGER
ETCD
Pod
Pod-list
: nodeName
awesome :
KUBE SCHEDULER

```
kubectl get pods --all-namespaces

NAMESPACE        NAME                          READY   STATUS    RESTARTS   AGE

default          awesome-pod                   0/1     Pending   0          100 years
```

(2) Authorization

ADMISSION CONTROLLERS

(1) mutate

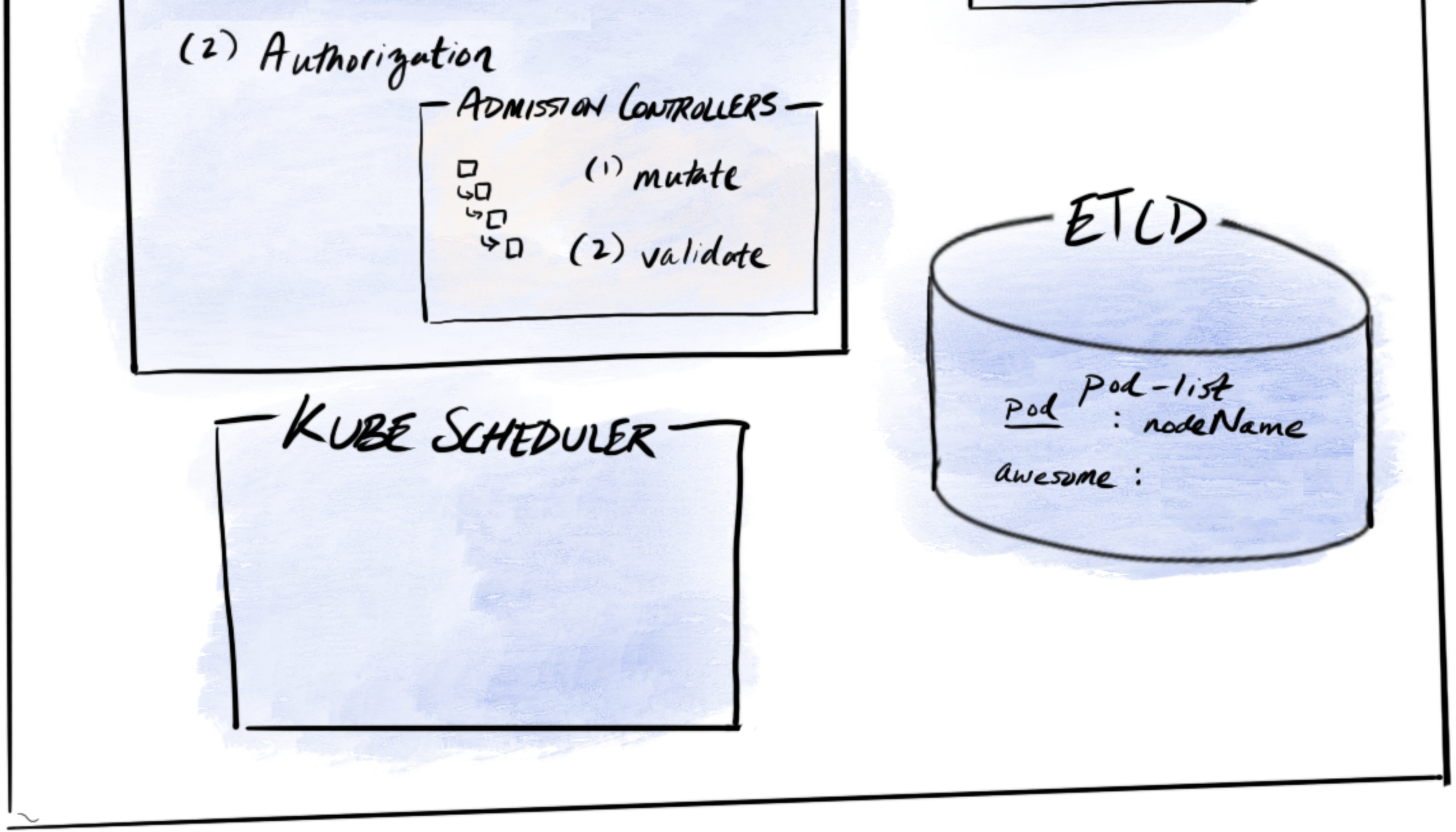
(2) validate

KUBE SCHEDULER

ETCD

Pod Pod-list : nodeName

awesome :



(2) Authorization

ADMISSION CONTROLLERS

(1) mutate

(2) validate

KUBE SCHEDULER

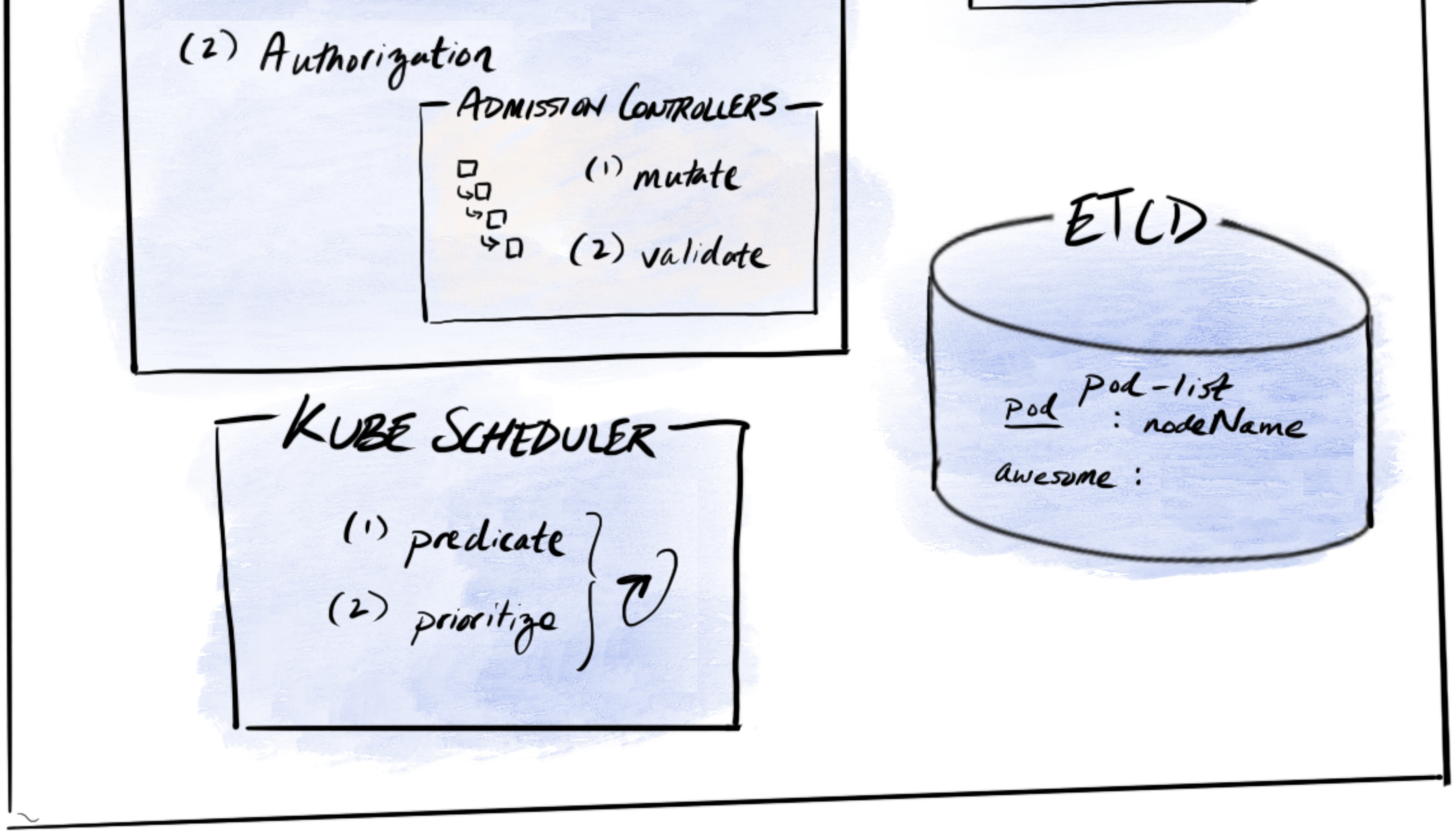
(1) predicate

(2) prioritize

ETCD

Pod Pod-list : nodeName

awesome :



(2) Authorization

ADMISSION CONTROLLERS

(1) mutate

(2) validate

KUBE SCHEDULER

(1) predicate

(2) prioritize

ETCD

Pod Pod-list : nodeName

awesome : worker

```
kubectl get pods --all-namespaces

NAMESPACE        NAME                READY   STATUS    RESTARTS   AGE

default          awesome-pod         0/1   PodScheduled      0      100 years
```

(2) Authorization

ADMISSION CONTROLLERS

(1) mutate

(2) validate

KUBE SCHEDULER

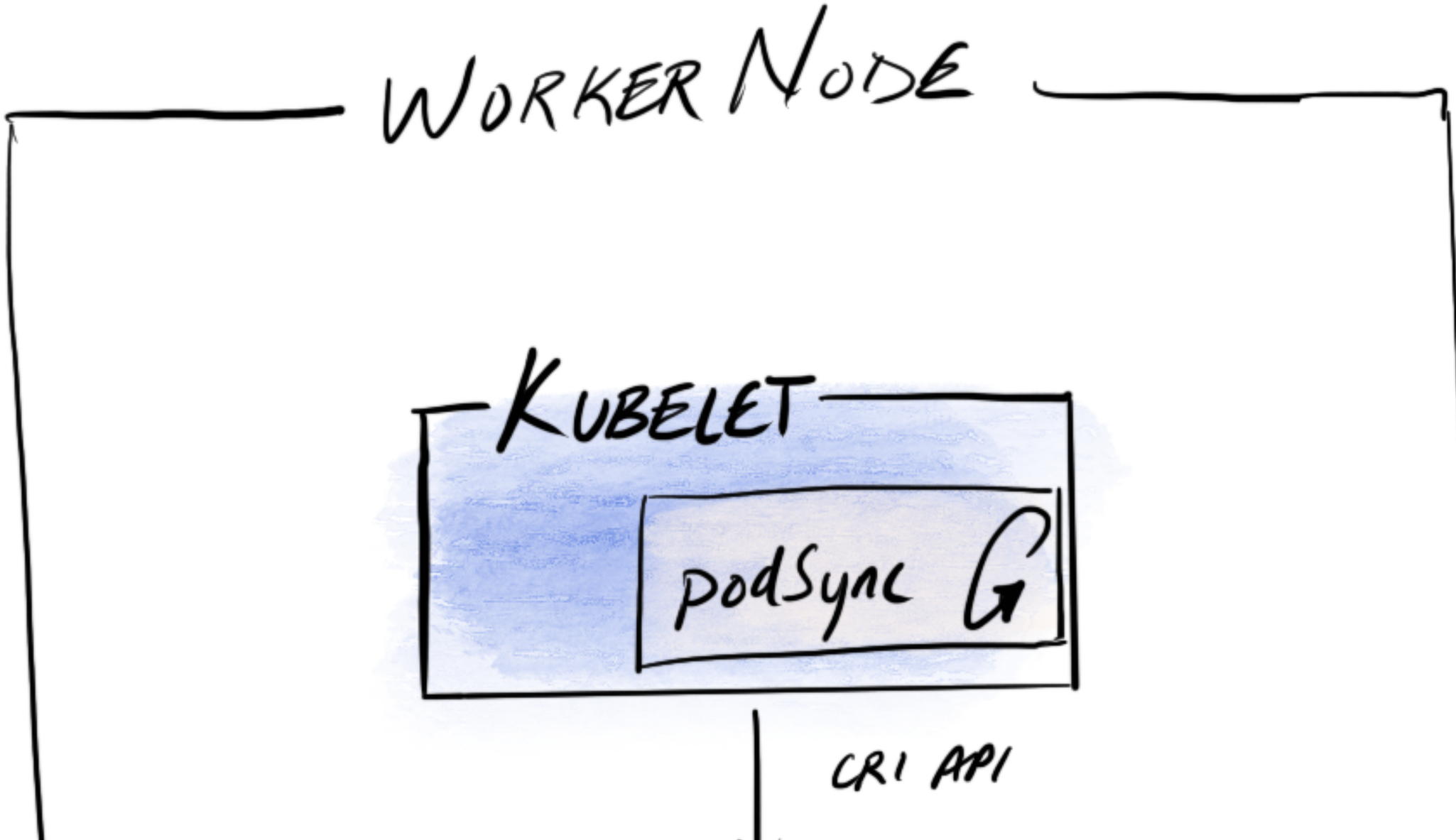
(1) predicate

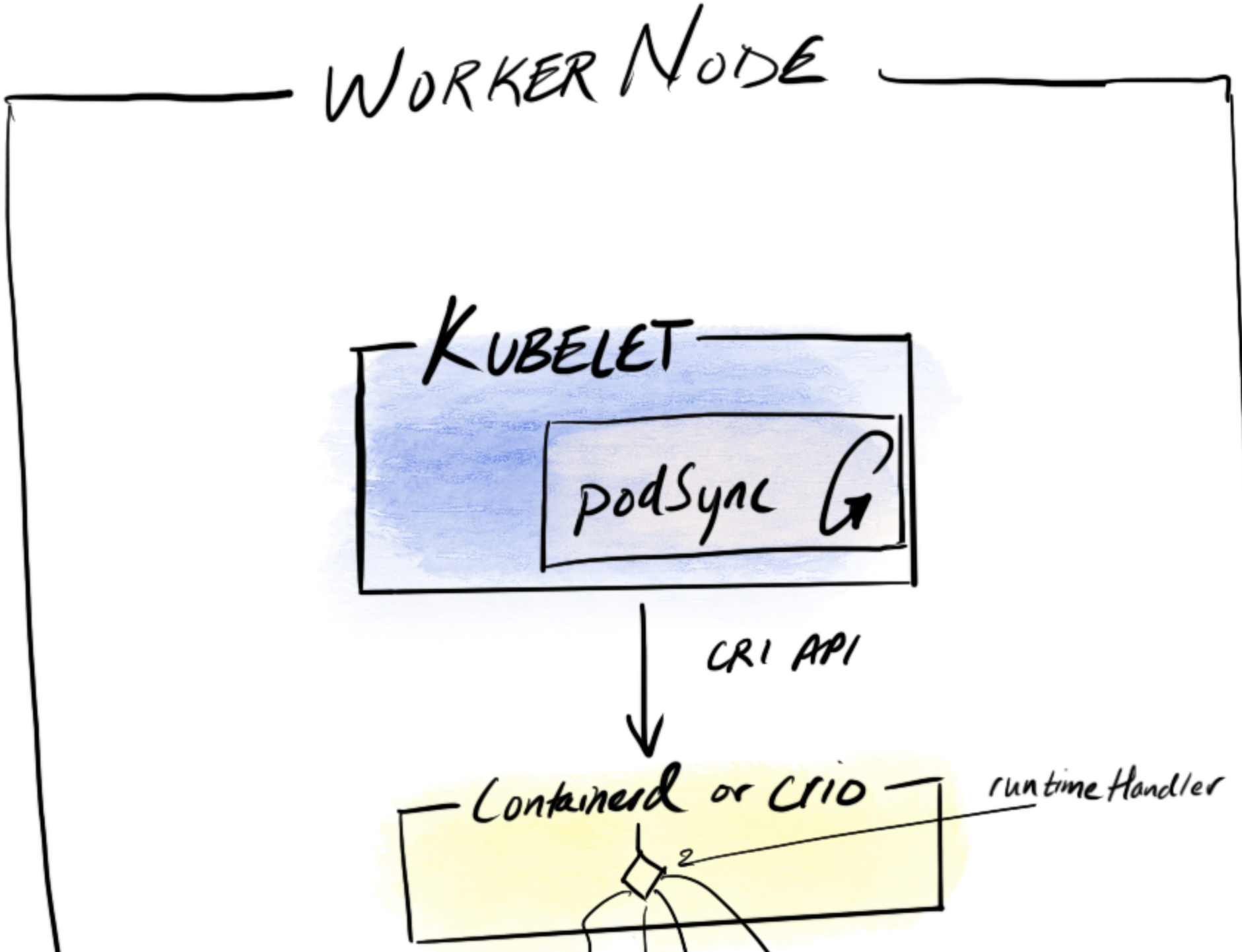
(2) prioritize

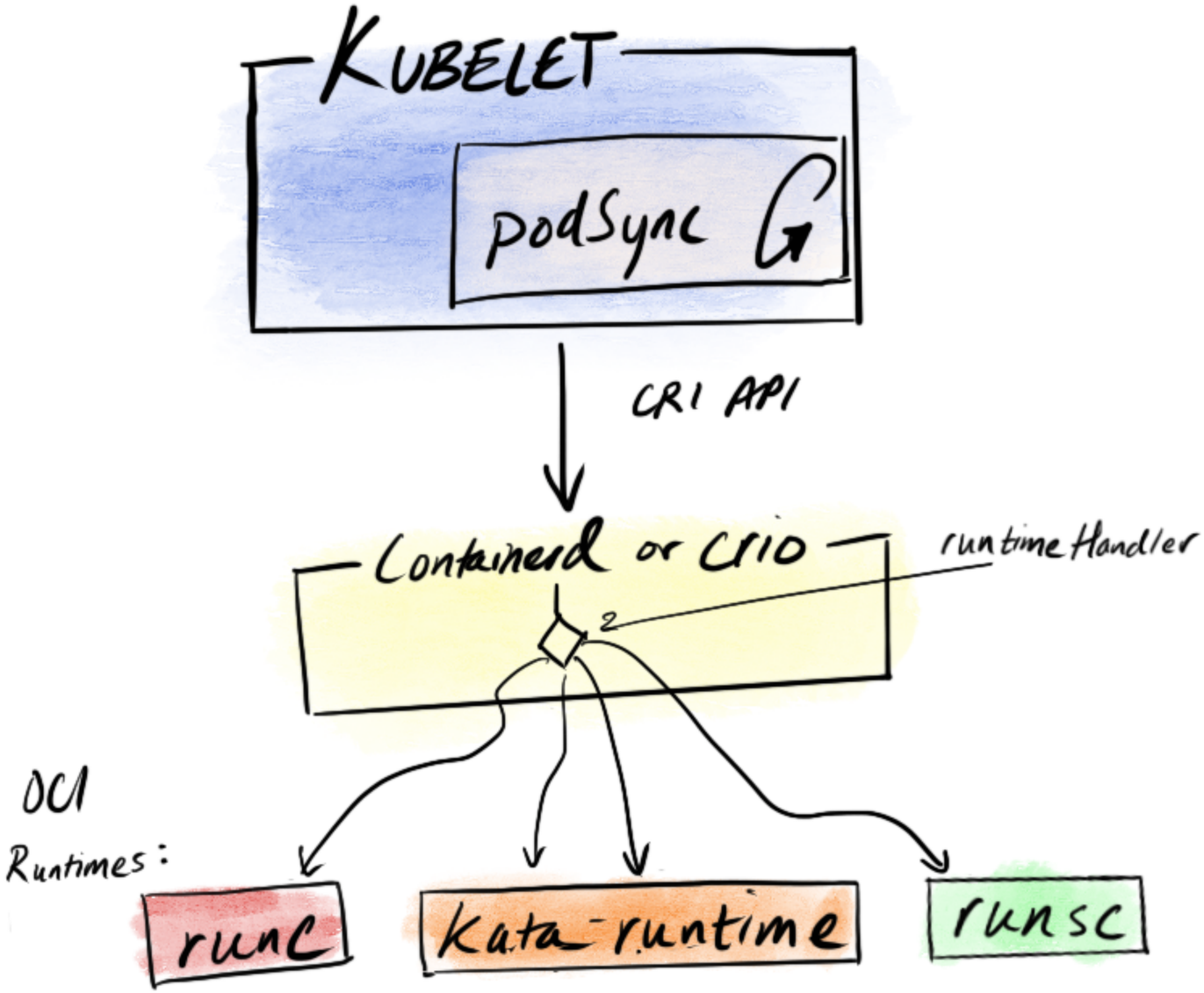
ETCD

Pod Pod-list : nodeName

awesome : worker


WORKER NODE
KUBELET
podSync
CRI API


WORKER NODE
KUBELET
podSync
CRI API
Containerd or CRIO
runtimeHandler
2



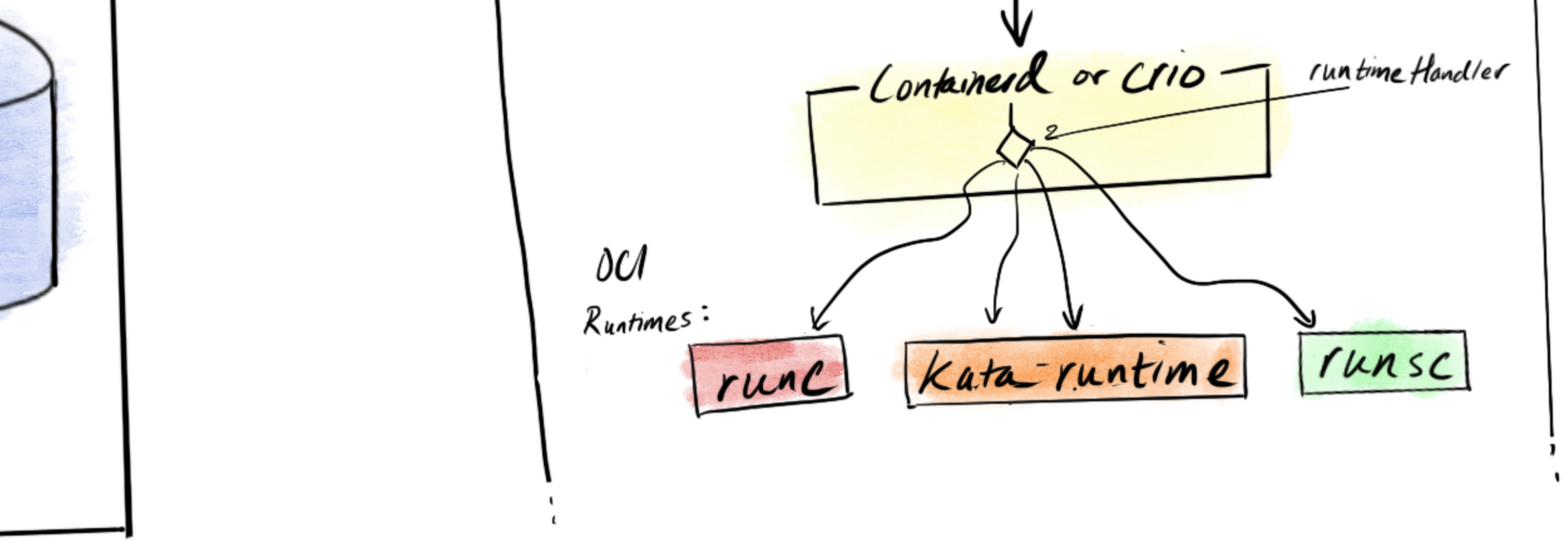
-POD SYNC-

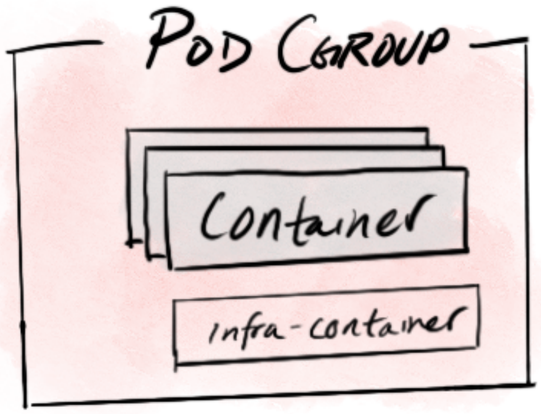
1) Create Pod cgroup

2) Ask CRI to create sandbox
   ↳ CRI calls CNI
   ↳ CRI calls OCI

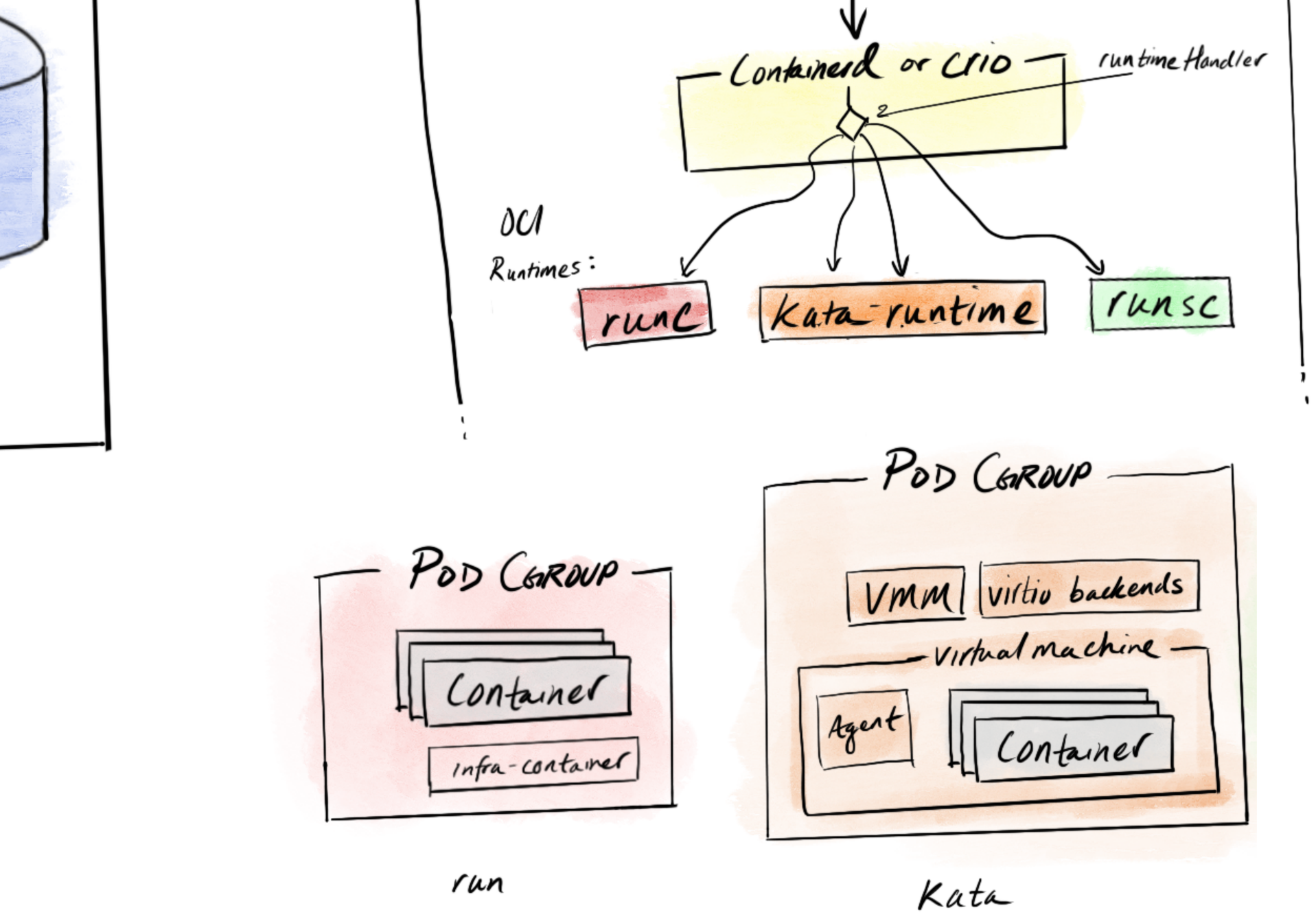
3) Ask CRI to pull image

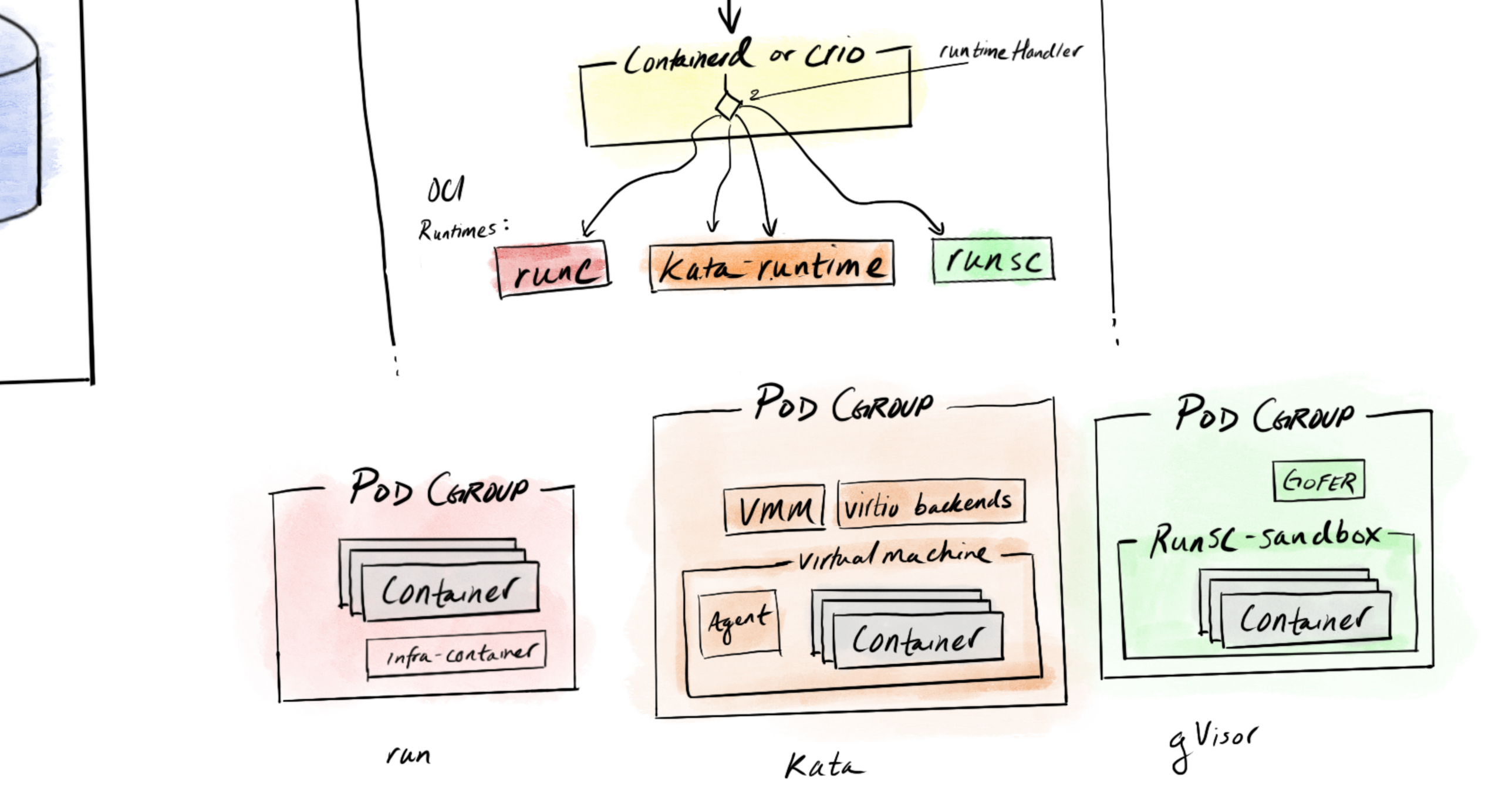
4) Ask CRI to create container
   ↳ CRI calls OCI

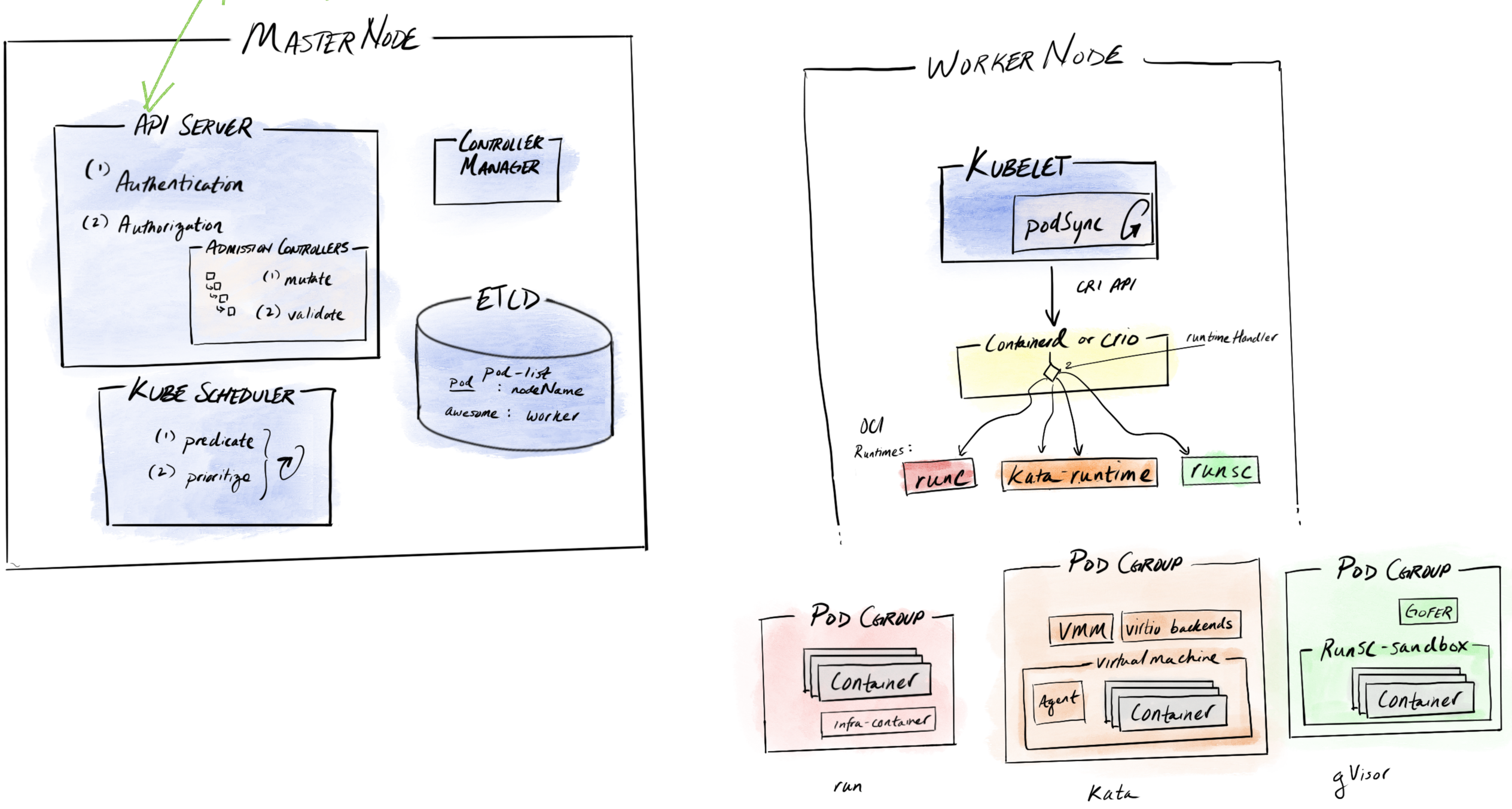

Containerd or CRIO
runtime Handler
2
OCI
Runtimes:
runc
kata-runtime
runsc


POD CGROUP
Container
infra-container

run

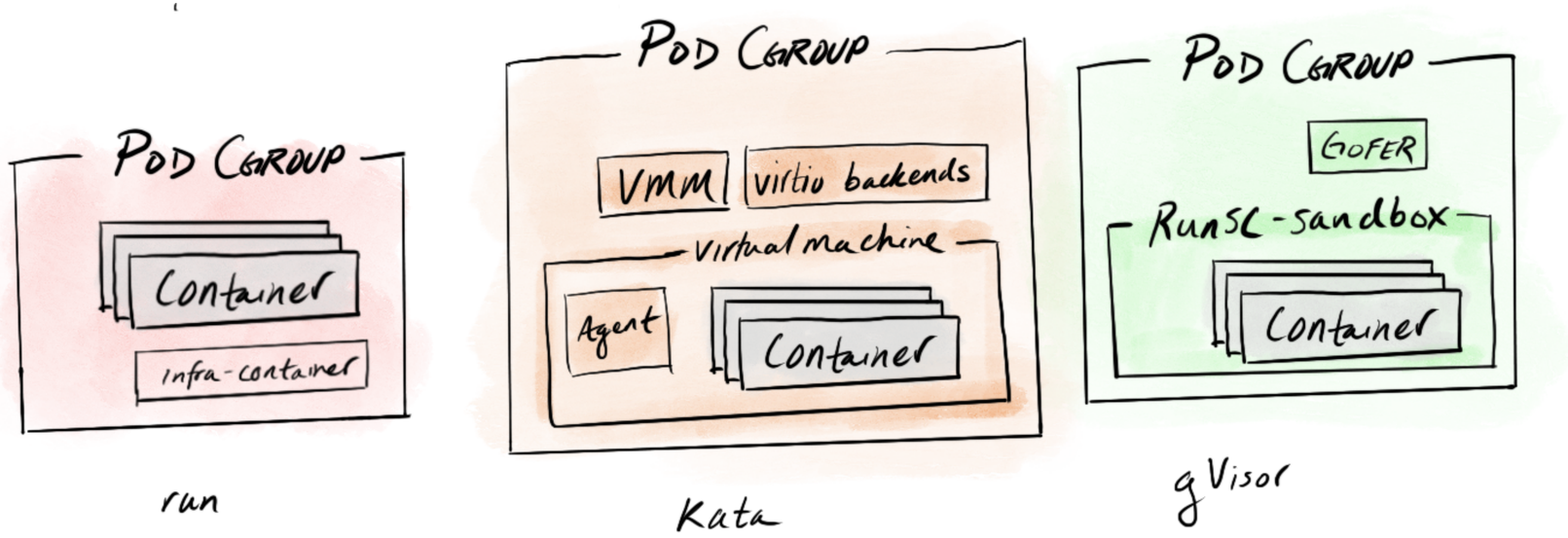

containerd or cri-o
runtimeHandler
2
OCI
Runtimes:
runc
kata-runtime
runsc
Pod Cgroup
Container
infra-container
run
Pod Cgroup
VMM
virtio backends
virtual machine
Agent
Container
Kata


containerd or cri-o
runtimeHandler
2
OCI
Runtimes:
runc
kata-runtime
runsc
Pod Cgroup
Container
infra-container
run
Pod Cgroup
VMM
virtio backends
virtual machine
Agent
Container
Kata
Pod Cgroup
Gofer
Runsc-sandbox
Container
gVisor

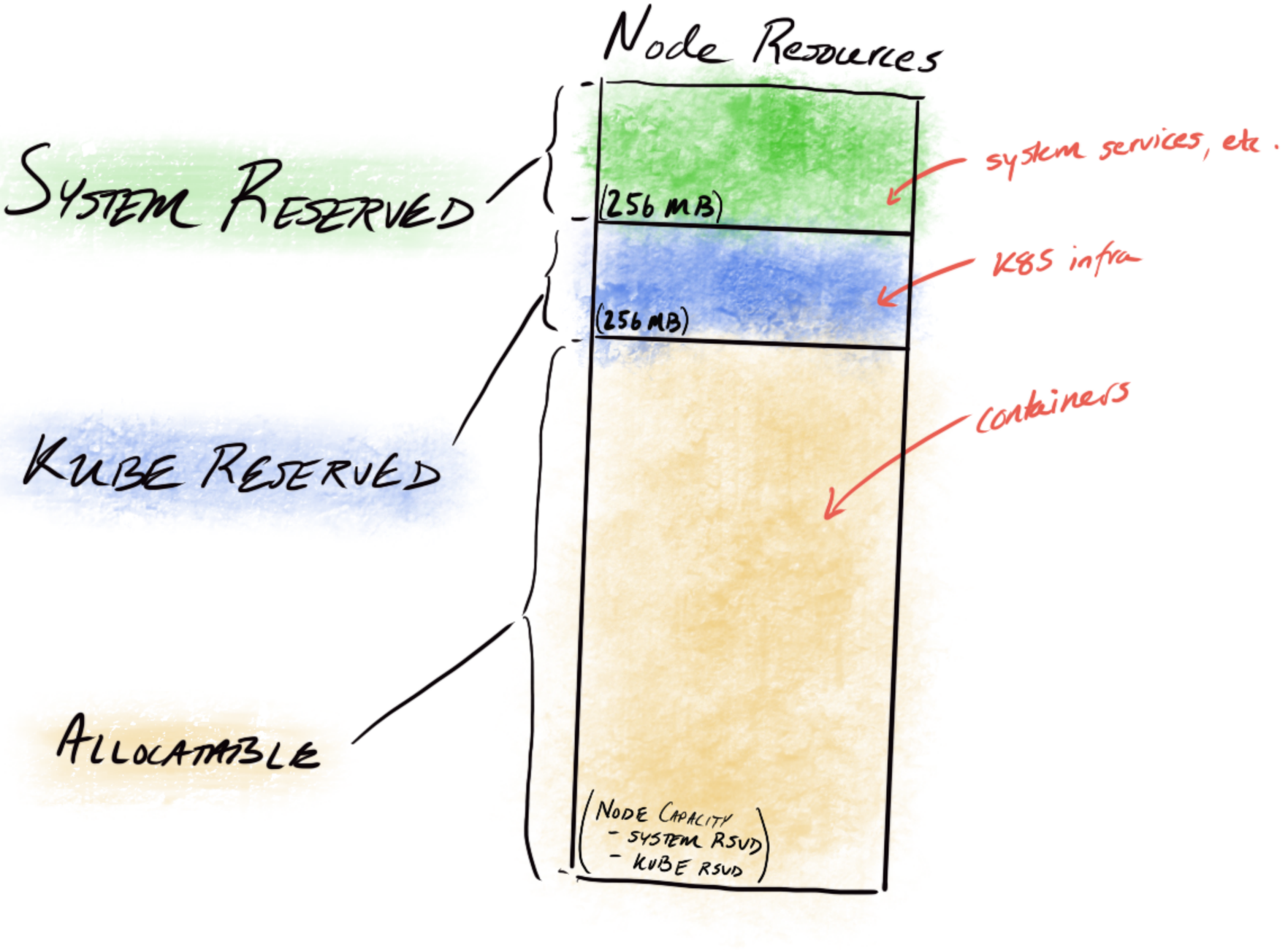

MASTER NODE
API SERVER
(1) Authentication
(2) Authorization
ADMISSION CONTROLLERS
(1) mutate
(2) validate
CONTROLLER MANAGER
ETCD
Pod
Pod-list : nodeName
awesome : worker
KUBE SCHEDULER
(1) predicate
(2) prioritize
WORKER NODE
KUBELET
podSync
CRI API
Containerd or CRIO
runtimeHandler
OCI Runtimes:
runc
kata-runtime
runsc
POD CGROUP
Container
infra-container
run
POD CGROUP
VMM
virtio backends
virtual machine
Agent
Container
Kata
POD CGROUP
GOFER
RunSC-sandbox
Container
gVisor

Pod Overhead?

Pod Overhead?



pod.resources
!=
sum(container[ ].resources)


Node Resources
System Reserved
(256 MB)
system services, etc.
Kube Reserved
(256 MB)
K8S infra
containers
Allocatable
(Node Capacity
- system RSVD
- kube RSVD)

PodOverhead

Alpha feature, introduced in 1.16, which accounts for the overheads associated with running a pod.

How it works

1. Overhead fields are added to RuntimeClass and PodSpec definitions

2. At admission time, update PodSpec to include overhead *iff* a valid overhead is specified in the specified RuntimeClass

3. Account for this overhead in remaining pod life-cycle / management Kubernetes

1. vim

```
---
kind: RuntimeClass
apiVersion: node.k8s.io/v1beta1
metadata:
  name: kata-fc
handler: kata-fc
overhead:
  podFixed:
    memory: "130Mi"
    cpu: "250m"
```

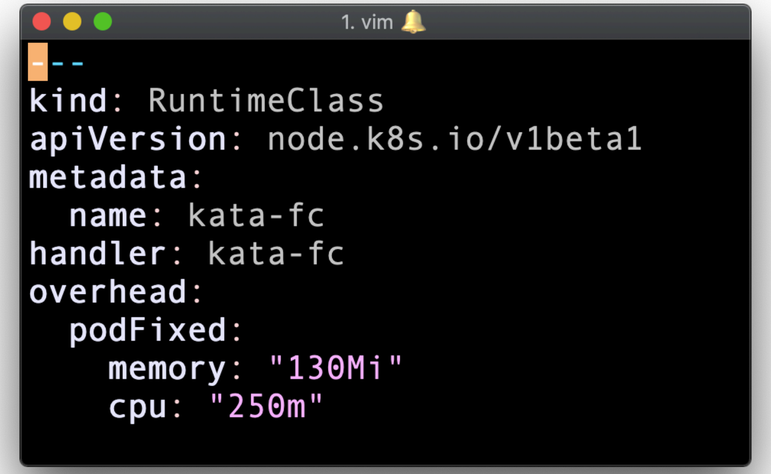
1. vim

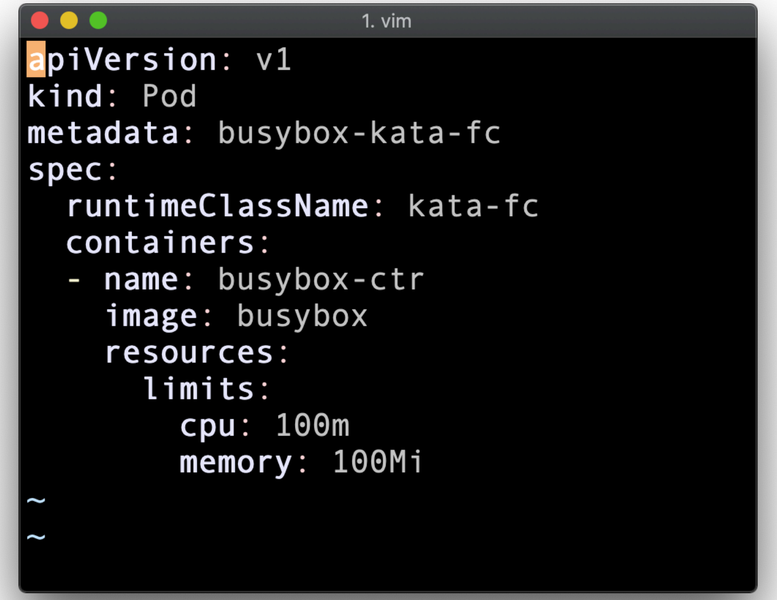
```
apiVersion: v1
kind: Pod
metadata: busybox-kata-fc
spec:
  runtimeClassName: kata-fc
  containers:
  - name: busybox-ctr
    image: busybox
    resources:
      limits:
        cpu: 100m
        memory: 100Mi
~
~
```

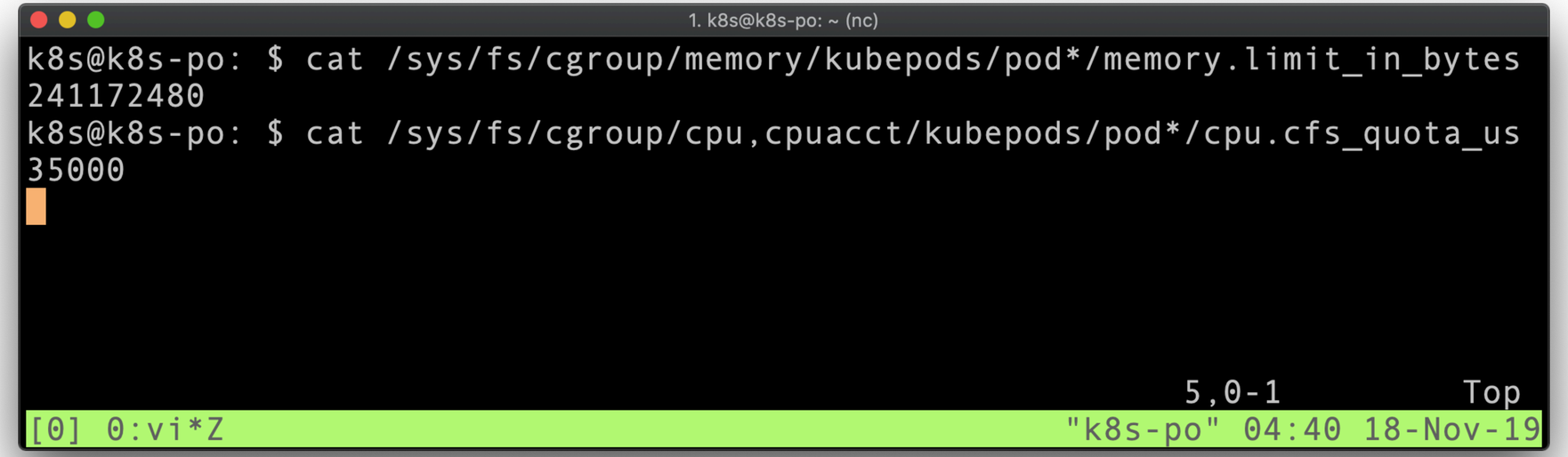
kubectl get pod busybox-kata-qemu -o yaml

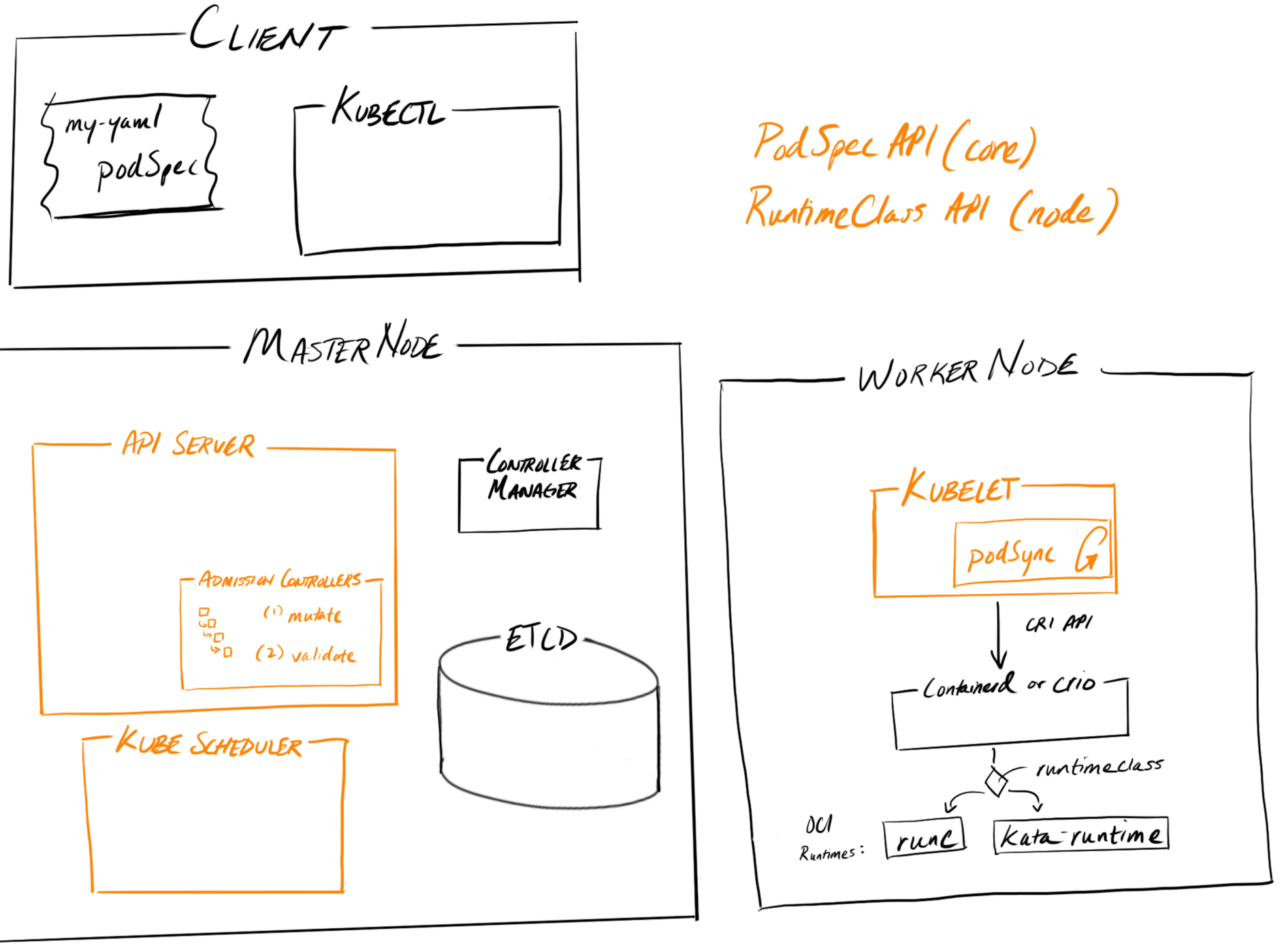
1. k8s@k8s-po: ~ (nc)

```
      limits:
        cpu: 100m
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-vw9mh
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: k8s-po
  overhead:
    cpu: 250m
    memory: 130Mi
  priority: 0
  restartPolicy: Always
  runtimeClassName: kata-fc
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
-- VISUAL --                          6          40,28          27%
[0] 0:vi*Z                                  "k8s-po" 04:26 18-Nov-19
```

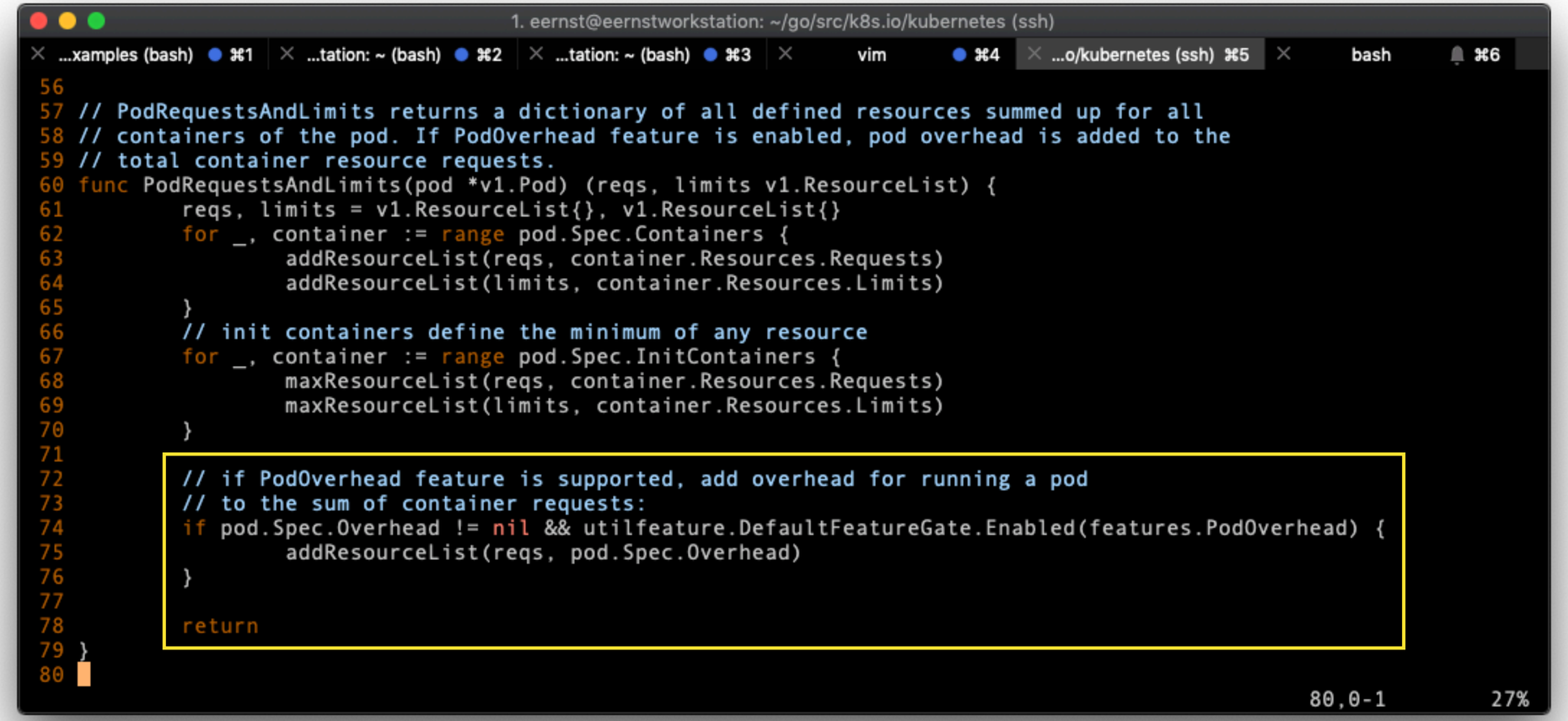

1. vim
---
kind: RuntimeClass
apiVersion: node.k8s.io/v1beta1
metadata:
  name: kata-fc
handler: kata-fc
overhead:
  podFixed:
    memory: "130Mi"
    cpu: "250m"


1. vim
apiVersion: v1
kind: Pod
metadata: busybox-kata-fc
spec:
  runtimeClassName: kata-fc
  containers:
  - name: busybox-ctr
    image: busybox
    resources:
      limits:
        cpu: 100m
        memory: 100Mi
~
~


1. k8s@k8s-po: ~ (nc)
k8s@k8s-po: $ cat /sys/fs/cgroup/memory/kubepods/pod*/memory.limit_in_bytes
241172480
k8s@k8s-po: $ cat /sys/fs/cgroup/cpu,cpuacct/kubepods/pod*/cpu.cfs_quota_us
35000
5,0-1 Top
[0] 0:vi*Z "k8s-po" 04:40 18-Nov-19


Client
my-yaml
podSpec
kubectl
PodSpec API (core)
RuntimeClass API (node)
Master Node
API Server
Admission Controllers
(1) mutate
(2) validate
Controller Manager
etcd
Kube Scheduler
Worker Node
kubelet
podSync
CRI API
Containerd or CRIO
runtimeclass
OCI Runtimes:
runc
kata-runtime

Changes Required

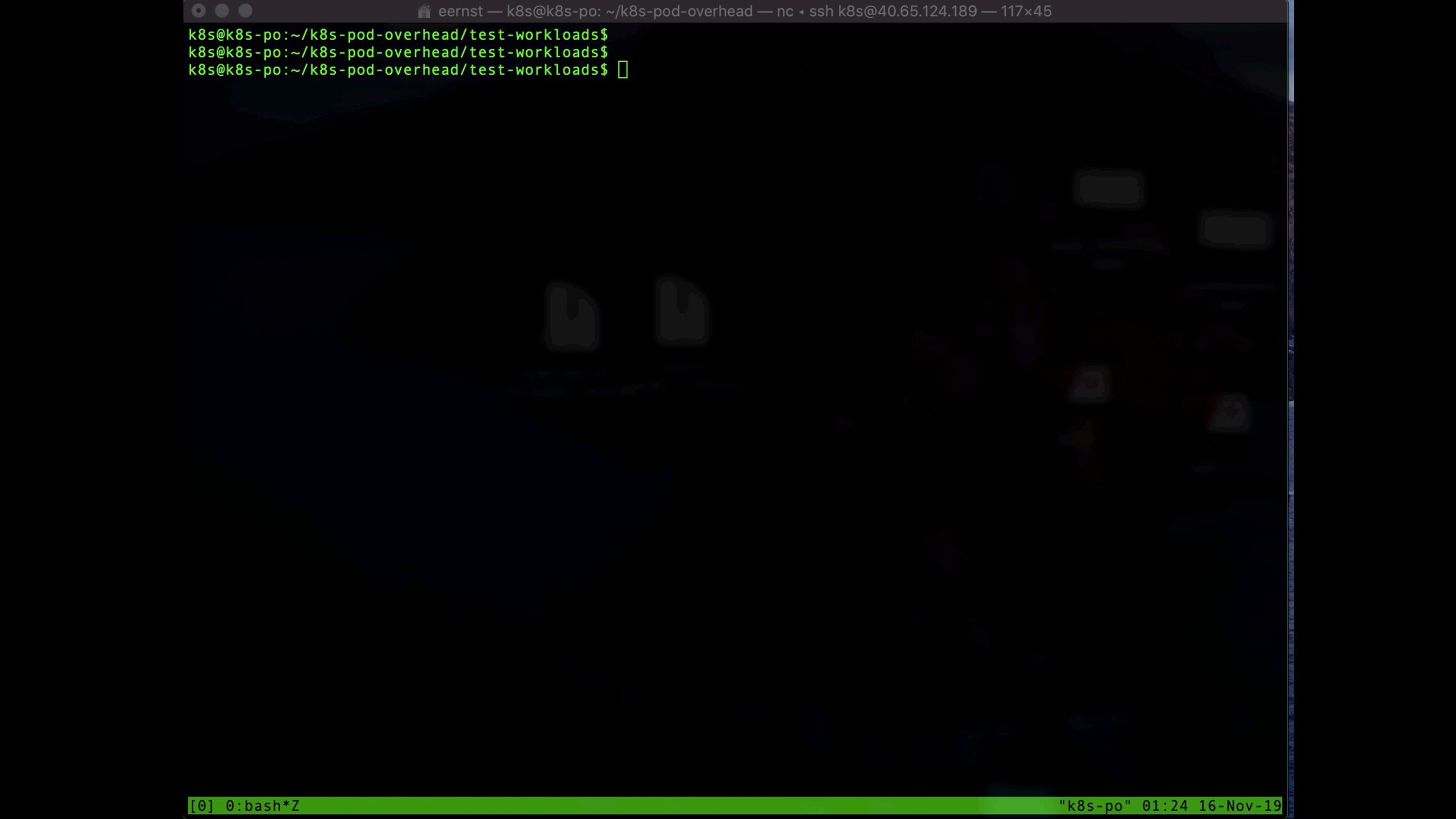


Who cares?

Users

Severely over-constrained: Oom…Why isn’t my workload running?

over-constrained: inconsistent, poor performance

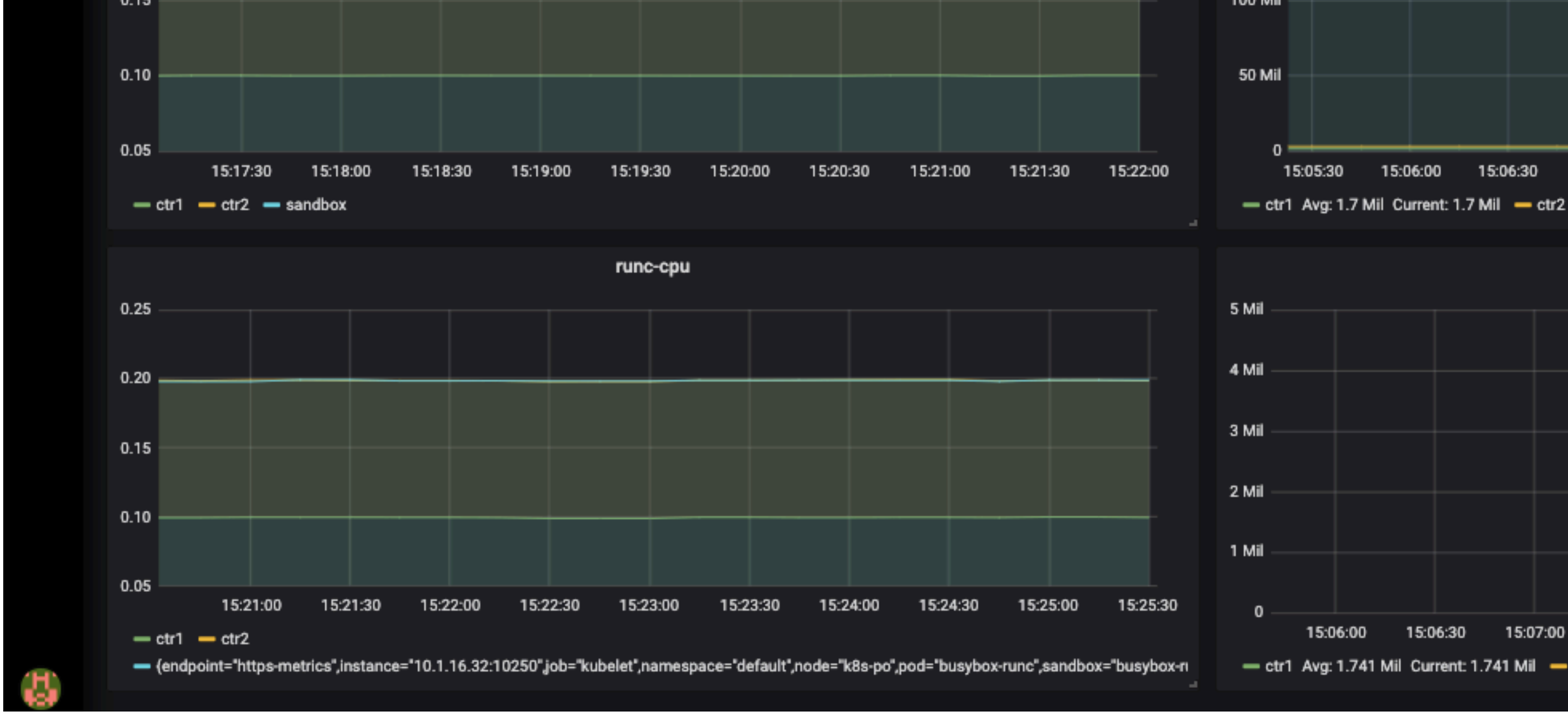

eernst — k8s@k8s-po: ~/k8s-pod-overhead — nc ◂ ssh k8s@40.65.124.189 — 117×45
k8s@k8s-po:~/k8s-pod-overhead/test-workloads$
k8s@k8s-po:~/k8s-pod-overhead/test-workloads$
k8s@k8s-po:~/k8s-pod-overhead/test-workloads$
[0] 0:bash*Z
"k8s-po" 01:24 16-Nov-19

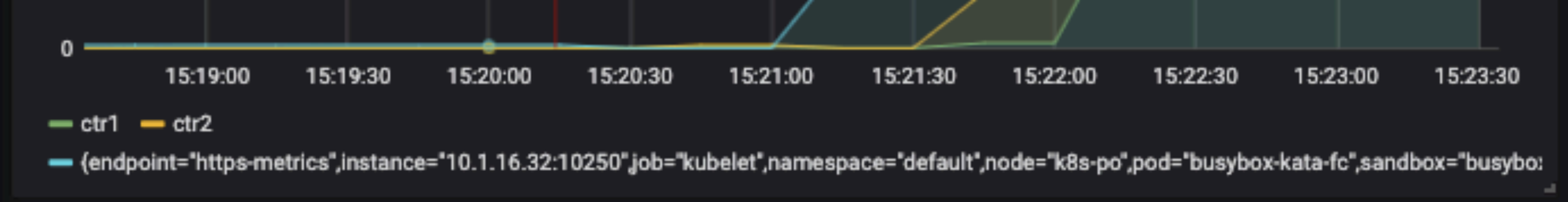
Administrators

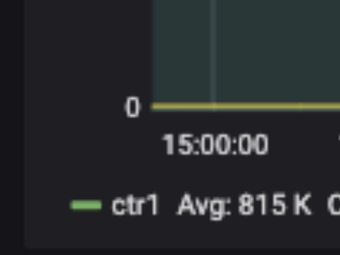
ResourceQuota doesn’t reflect reality

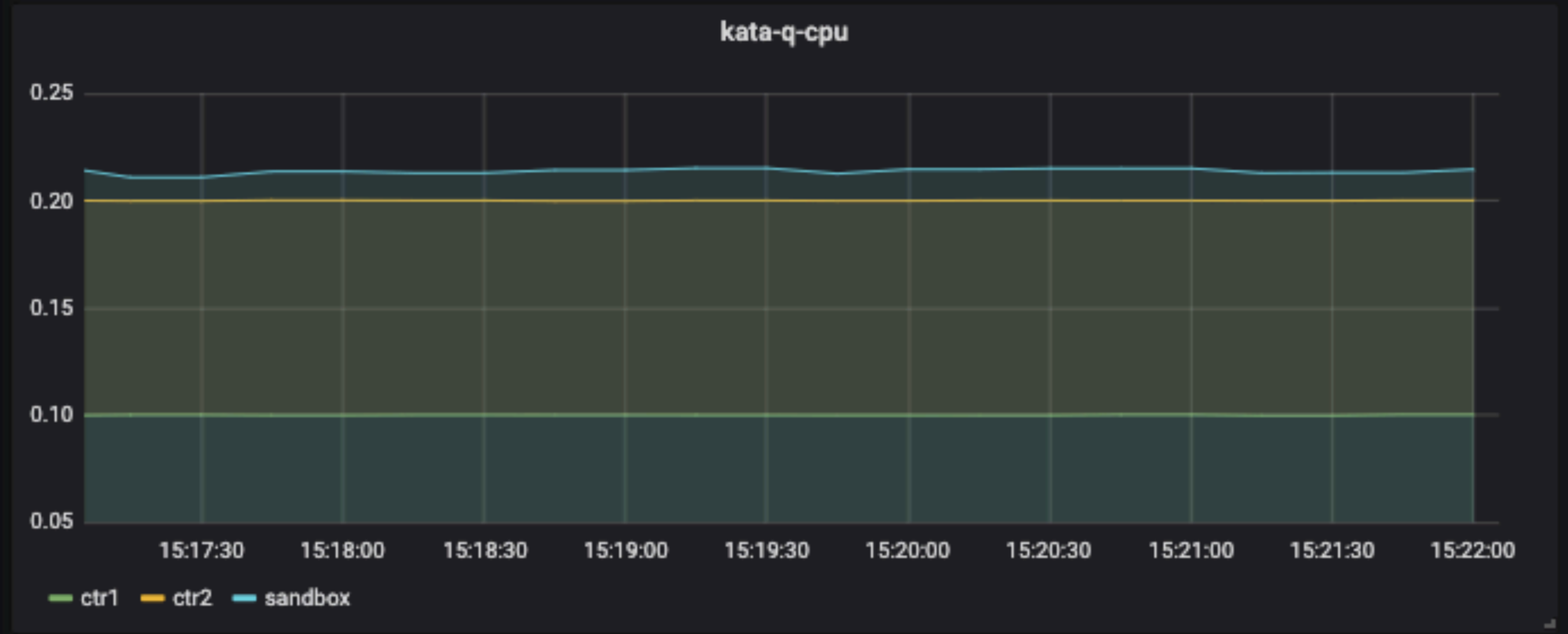

New dashboard Copy
Last 5 minutes
kata-fc-cpu
0.25
0.20
0.15
0.10
0.05
0
15:19:00
15:19:30
15:20:00
15:20:30
15:21:00
15:21:30
15:22:00
15:22:30
15:23:00
15:23:30
2019-11-11 15:20:00
ctr1: 0
ctr2: 0
{endpoint="https-metrics",instance="10.1.16.32:10250",job="kubelet",namespace="default",node="k8s-po",pod="busyb... 0.0020
ctr1
ctr2
{endpoint="https-metrics",instance="10.1.16.32:10250",job="kubelet",namespace="default",node="k8s-po",pod="busybox-kata-fc",sandbox="busybo
kata-fc-memory
150 Mil
0
15:00:00
15:00:30
15:01:00
15:01:30
15:02:00
15:02:30
15:03:00
15:03:30
15:04:00
15:04:30
ctr1 Avg: 815 K Current: 1.0 Mil
ctr2 Avg: 857 K Current: 1.1 Mil
sandbox Avg: 128.8 Mil Current: 129.9 Mil
kata-q-cpu
0.25
0.20
0.15
0.10
0.05
15:17:30
15:18:00
15:18:30
15:19:00
15:19:30
15:20:00
15:20:30
15:21:00
15:21:30
15:22:00
ctr1
ctr2
sandbox
kata-q-memory
200 Mil
150 Mil
100 Mil
50 Mil
0
15:05:30
15:06:00
15:06:30
15:07:00
15:07:30
15:08:00
15:08:30
15:09:00
15:09:30
15:10:00
ctr1 Avg: 1.7 Mil Current: 1.7 Mil
ctr2 Avg: 1.6 Mil Current: 1.6 Mil
sandbox Avg: 159.3 Mil Current: 159.4 Mil
runc-cpu
0.25
0.20
0.15
0.10
0.05
15:21:00
15:21:30
15:22:00
15:22:30
15:23:00
15:23:30
15:24:00
15:24:30
15:25:00
15:25:30
ctr1
ctr2
{endpoint="https-metrics",instance="10.1.16.32:10250",job="kubelet",namespace="default",node="k8s-po",pod="busybox-runc",sandbox="busybox-r
runc-memory
5 Mil
4 Mil
3 Mil
2 Mil
1 Mil
0
15:06:00
15:06:30
15:07:00
15:07:30
15:08:00
15:08:30
15:09:00
15:09:30
15:10:00
15:10:30
ctr1 Avg: 1.741 Mil Current: 1.741 Mil
ctr2 Avg: 1.800 Mil Current: 1.753 Mil
sandbox Avg: 3.497 Mil Current: 4.809 Mil

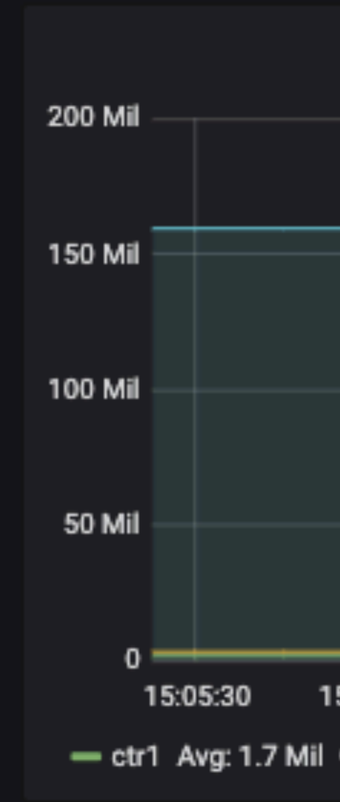

50 Mil
0
15:05:30
15:06:00
15:06:30
15:07:00
15:07:30
15:08:00
15:08:30
15:09:00
15:09:30
15:10:00
ctr1 Avg: 1.7 Mil Current: 1.7 Mil
ctr2 Avg: 1.6 Mil Current: 1.6 Mil
sandbox Avg: 159.3 Mil Current: 159.4 Mil
15:21:30
15:22:00
runc-memory
5 Mil
4 Mil
3 Mil
2 Mil
1 Mil
0
15:06:00
15:06:30
15:07:00
15:07:30
15:08:00
15:08:30
15:09:00
15:09:30
15:10:00
15:10:30
ctr1 Avg: 1.741 Mil Current: 1.741 Mil
ctr2 Avg: 1.800 Mil Current: 1.753 Mil
sandbox Avg: 3.497 Mil Current: 4.809 Mil
15:25:00
15:25:30

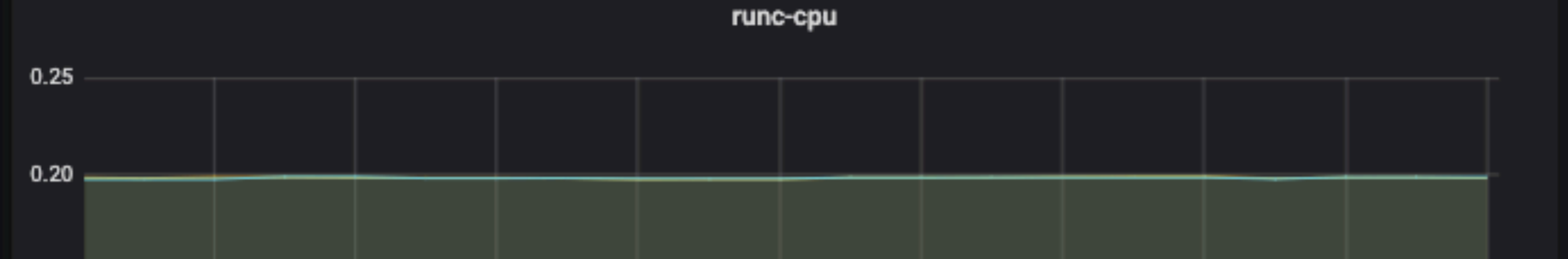

0.10
0.05
15:17:30
15:18:00
15:18:30
15:19:00
15:19:30
15:20:00
15:20:30
15:21:00
15:21:30
15:22:00
ctr1
ctr2
sandbox
50 Mil
0
15:05:30
15:06:00
15:06:30
ctr1 Avg: 1.7 Mil Current: 1.7 Mil
ctr2
runc-cpu
0.25
0.20
0.15
0.10
0.05
15:21:00
15:21:30
15:22:00
15:22:30
15:23:00
15:23:30
15:24:00
15:24:30
15:25:00
15:25:30
ctr1
ctr2
{endpoint="https-metrics",instance="10.1.16.32:10250",job="kubelet",namespace="default",node="k8s-po",pod="busybox-runc",sandbox="busybox-r
5 Mil
4 Mil
3 Mil
2 Mil
1 Mil
0
15:06:00
15:06:30
15:07:00
ctr1 Avg: 1.741 Mil Current: 1.741 Mil

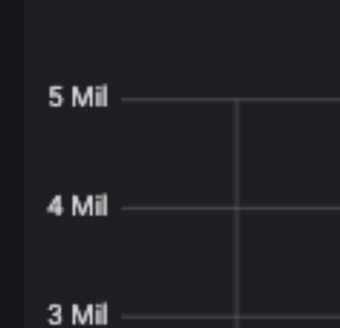

0
15:19:00
15:19:30
15:20:00
15:20:30
15:21:00
15:21:30
15:22:00
15:22:30
15:23:00
15:23:30
ctr1
ctr2
{endpoint="https-metrics",instance="10.1.16.32:10250",job="kubelet",namespace="default",node="k8s-po",pod="busybox-kata-fc",sandbox="busybo

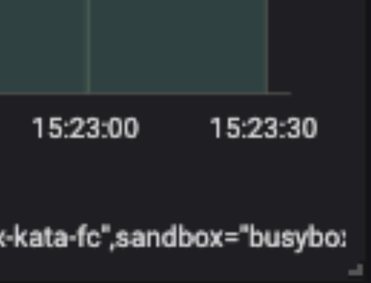

0
15:00:00
ctr1 Avg: 815 K

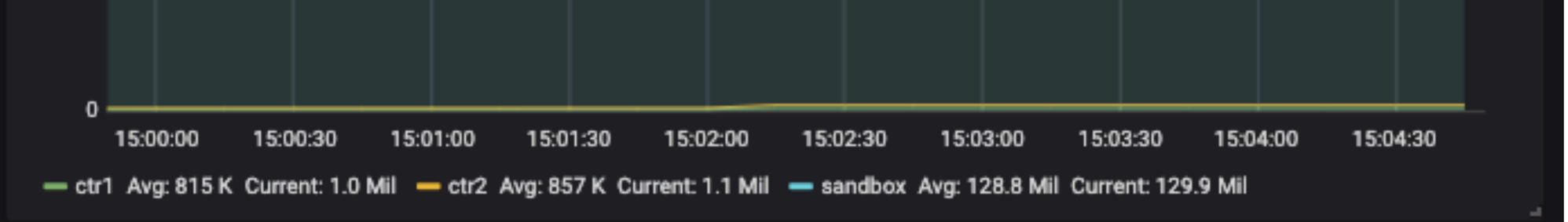

kata-q-cpu
0.25
0.20
0.15
0.10
0.05
15:17:30
15:18:00
15:18:30
15:19:00
15:19:30
15:20:00
15:20:30
15:21:00
15:21:30
15:22:00
ctr1
ctr2
sandbox

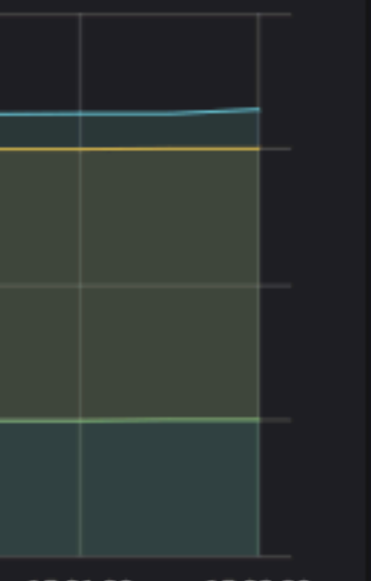

200 Mil
150 Mil
100 Mil
50 Mil
0
15:05:30
ctr1 Avg: 1.7 Mil

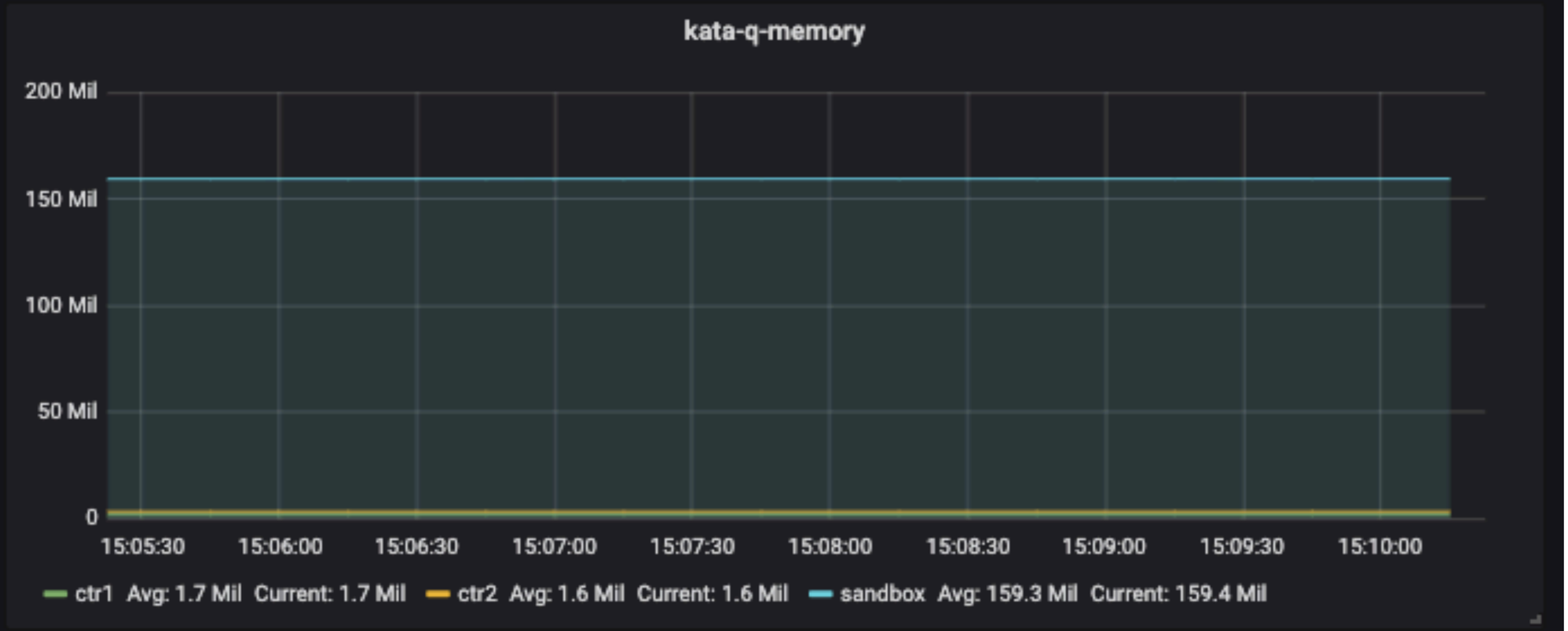

runc-cpu
0.25
0.20

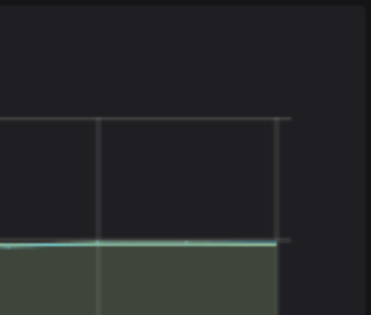

5 Mil
4 Mil
3 Mil


15:23:00
15:23:30

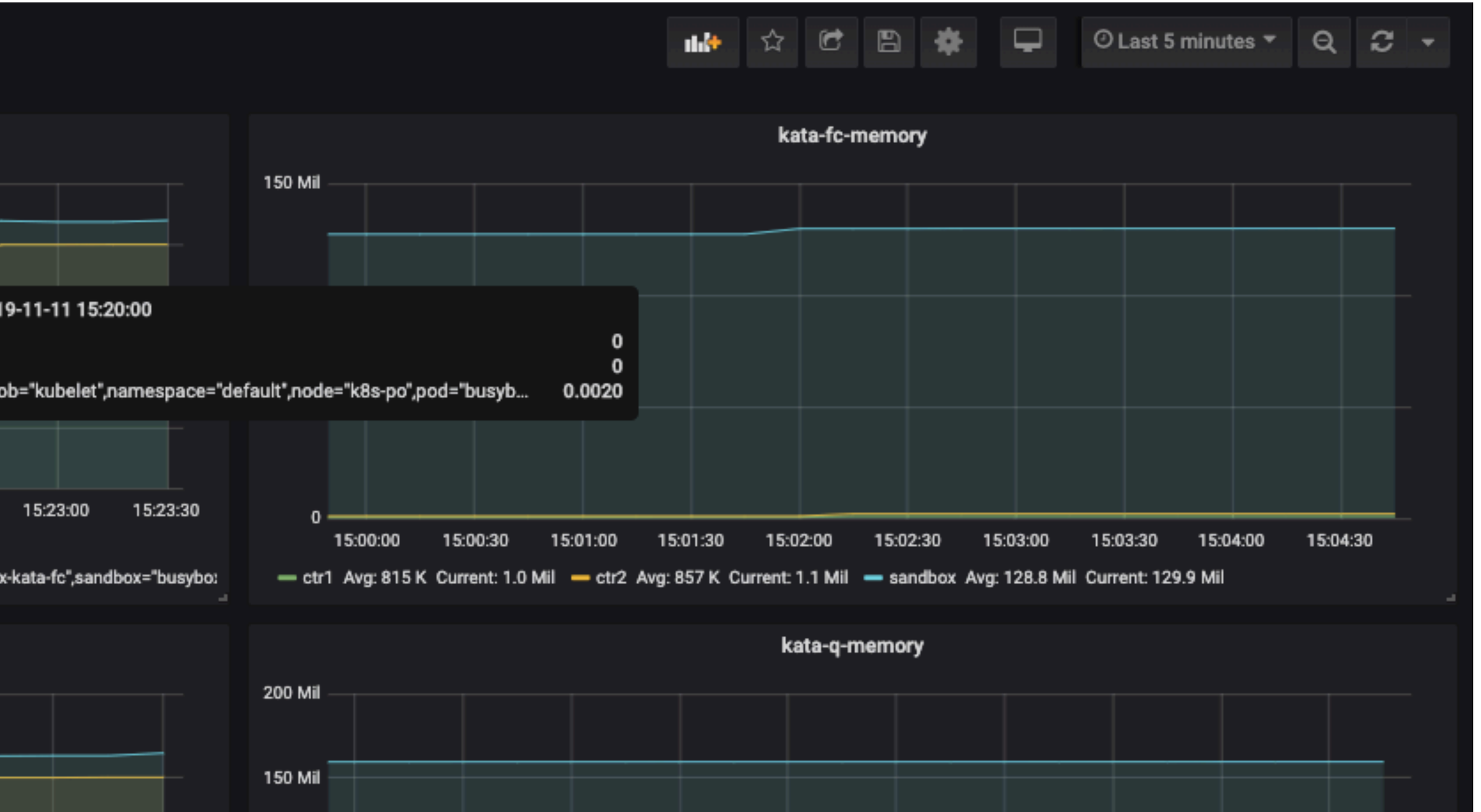

0
15:00:00
15:00:30
15:01:00
15:01:30
15:02:00
15:02:30
15:03:00
15:03:30
15:04:00
15:04:30
ctr1 Avg: 815 K Current: 1.0 Mil
ctr2 Avg: 857 K Current: 1.1 Mil
sandbox Avg: 128.8 Mil Current: 129.9 Mil


15:21:30
15:22:00


kata-q-memory
200 Mil
150 Mil
100 Mil
50 Mil
0
15:05:30
15:06:00
15:06:30
15:07:00
15:07:30
15:08:00
15:08:30
15:09:00
15:09:30
15:10:00
ctr1 Avg: 1.7 Mil Current: 1.7 Mil
ctr2 Avg: 1.6 Mil Current: 1.6 Mil
sandbox Avg: 159.3 Mil Current: 159.4 Mil




runc-memory
5 Mil
4 Mil


Last 5 minutes
kata-fc-memory
150 Mil
0
15:00:00
15:00:30
15:01:00
15:01:30
15:02:00
15:02:30
15:03:00
15:03:30
15:04:00
15:04:30
ctr1 Avg: 815 K Current: 1.0 Mil
ctr2 Avg: 857 K Current: 1.1 Mil
sandbox Avg: 128.8 Mil Current: 129.9 Mil
19-11-11 15:20:00
0
0
ob="kubelet",namespace="default",node="k8s-po",pod="busyb... 0.0020
15:23:00
15:23:30
x-kata-fc",sandbox="busybo:
kata-q-memory
200 Mil
150 Mil

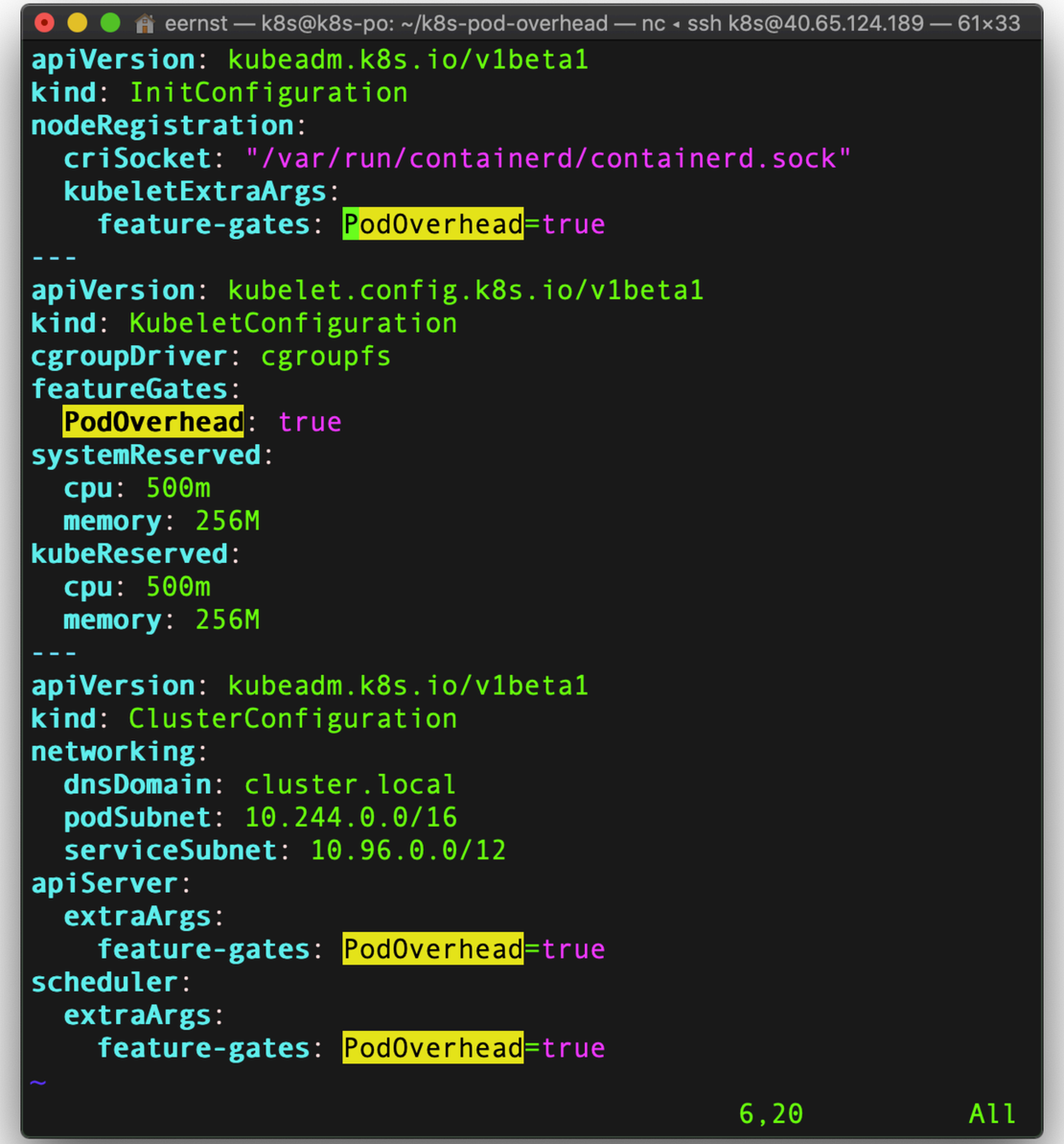
Providers

No room in the pod slice? We'll just drop xyz into system slice, and hope that's sized okay

If we don't account, who pays for it?

PodOverhead status

- Available in Kubernetes 1.16 as an alpha feature
- Overheads are static - good starting point, but not always realistic
- Expected to move to beta in 1.18 release



https://raw.githubusercontent.com/egernst/k8s-pod-overhead/master/kubeadm.yaml


KubeCon
CloudNativeCon
North America 2019
Thanks!

Learnings

- API Reviews can be painful, and are the longest pole in the tent
- Kubernetes makes use of a lot of auto-generated machinery, especially for API version conversions
- Feature wise, it is easy to get changes in Kubernetes which will improve node stability and/or more accurate accounting
- Time spent { coding << herding cats for reviews }
- Time spent coding:

  { writing unit tests and fixing > tracing code to see what happens where >> writing actual feature }