KubeCon

CloudNativeCon

North America 2024

Deep dive into Generic Control Planes and kcp

Stefan Schimanski, Upbound, Senior Principal Engineer

Mangirdas Judeikis, CAST AI, Staff Engineer

Agenda

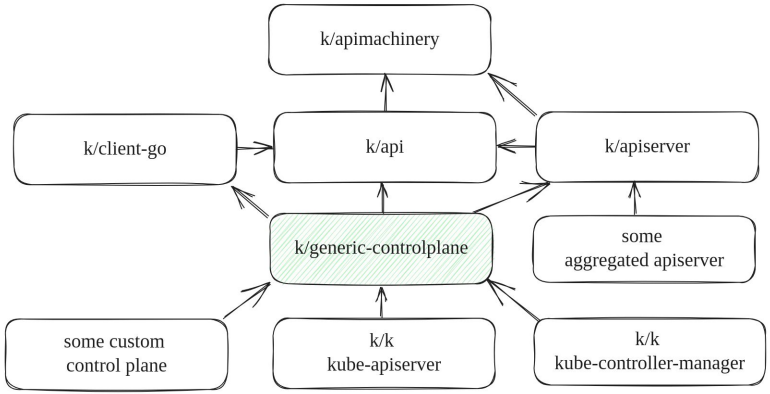
1. generic control plane
2. kcp
3. kcp component deep device & multi-regionality

Generic Control Plane – KEP 4080, where we are

1. make kube-apiserver composable, heavy lifting ✅ since 1.31

`k/k/pkg/controlplane/apiserver`	"generic ~~controlplane~~apiserver"
`k/k/pkg/controlplane`	composable kube-apiserver plumbing
`k/k/cmd/kube-apiserver`	kube-apiserver command consuming both

2. create staging repo k8s.io/generic-control-plane ❌ WIP



https://github.com/kubernetes/enhancements/tree/master/keps/sig-api-machinery/4080-generic-controlplane

Generic Control Plane

tl/dr: it’s not (yet) pretty, but most is in place – if you know where it is 💪

If you don’t, you are in the right talk 🎯

Generic Control Plane


```
mjudeikis@unknown2:/go/src/github.com/kcp-dev/generic-controlplane$ k api-resources
NAME              SHORTNAMES   APIVERSION   NAMESPACED   KIND
configmaps        cm           v1           true         ConfigMap
events            ev           v1           true         Event
namespaces        ns           v1           false        Namespace
resourcequotas    quota        v1           true         ResourceQuota
secrets                        v1           true         Secret
serviceaccounts   sa           v1           true         ServiceAccount
mjudeikis@unknown2:/go/src/github.com/kcp-dev/generic-controlplane$
```

https://github.com/kcp-dev/generic-controlplane

Generic Control Plane


```
mjudeikis@unknown2:/go/src/github.com/kcp-dev/generic-controlplane$ k api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
configmaps                        cm           v1                                     true         ConfigMap
events                            ev           v1                                     true         Event
namespaces                        ns           v1                                     false        Namespace
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingadmissionpolicies                    admissionregistration.k8s.io/v1        false        ValidatingAdmissionPolicy
validatingadmissionpolicybindings              admissionregistration.k8s.io/v1        false        ValidatingAdmissionPolicyBinding
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
selfsubjectreviews                             authentication.k8s.io/v1               false        SelfSubjectReview
tokenreviews                                   authentication.k8s.io/v1               false        TokenReview
localsubjectaccessreviews                      authorization.k8s.io/v1                true         LocalSubjectAccessReview
selfsubjectaccessreviews                       authorization.k8s.io/v1                false        SelfSubjectAccessReview
selfsubjectrulesreviews                        authorization.k8s.io/v1                false        SelfSubjectRulesReview
subjectaccessreviews                           authorization.k8s.io/v1                false        SubjectAccessReview
certificatesigningrequests        csr          certificates.k8s.io/v1                 false        CertificateSigningRequest
leases                                         coordination.k8s.io/v1                 true         Lease
events                            ev           events.k8s.io/v1                       true         Event
flowschemas                                    flowcontrol.apiserver.k8s.io/v1        false        FlowSchema
prioritylevelconfigurations                    flowcontrol.apiserver.k8s.io/v1        false        PriorityLevelConfiguration
clusterrolebindings                            rbac.authorization.k8s.io/v1           false        ClusterRoleBinding
clusterroles                                   rbac.authorization.k8s.io/v1           false        ClusterRole
rolebindings                                   rbac.authorization.k8s.io/v1           true         RoleBinding
roles                                          rbac.authorization.k8s.io/v1           true         Role
```

https://github.com/kcp-dev/generic-controlplane

Generic Control Plane – What is it?

1. A new code-path in github.com/kubernetes/kubernetes, where you can:

   initialize a control plane with specific apis only,
   including some from kube-apiserver.

kubernetes / pkg / controlplane / apiserver / Add file

pohly DRA admin access: add feature gate 9a7e4cc · 3 days ago History

Name	Last commit message	Last commit date
..		
admission	kube-apiserver: split admission initializers into ...	7 months ago
options	DRA admin access: add feature gate	3 days ago
samples	generic-controlplane: add generic-controlplan...	4 months ago
aggregator.go	CLE storage and type registration changes	4 months ago
apiextensions.go	apiserver: Add API emulation versioning.	5 months ago
apis.go	apiserver: Add API emulation versioning.	5 months ago
completion.go	controlplane/apiserver: move peer proxy code ...	7 months ago
config.go	cluster trust CM: wire in the new RequestHead...	2 months ago
config_test.go	add DefaultComponentGlobalsRegistry flags i...	5 months ago
peer.go	apiserver: Add API emulation versioning.	5 months ago
server.go	fix flake in TestLeaseCandidateCleanup	4 months ago

2. An optional wrapper / builder making it easily consumable: github.com/kcp-dev/generic-controlplane

M

Show me the code

https://github.com/kcp-dev/generic-controlplane

Show me the code

First some API server internals

https://github.com/kcp-dev/generic-controlplane

The Options-Config-Server Pattern

Options ➜ CompletedOptions ➜ Config ➜ CompletedConfig ➜ Server ➜ Prepared ➜ Run

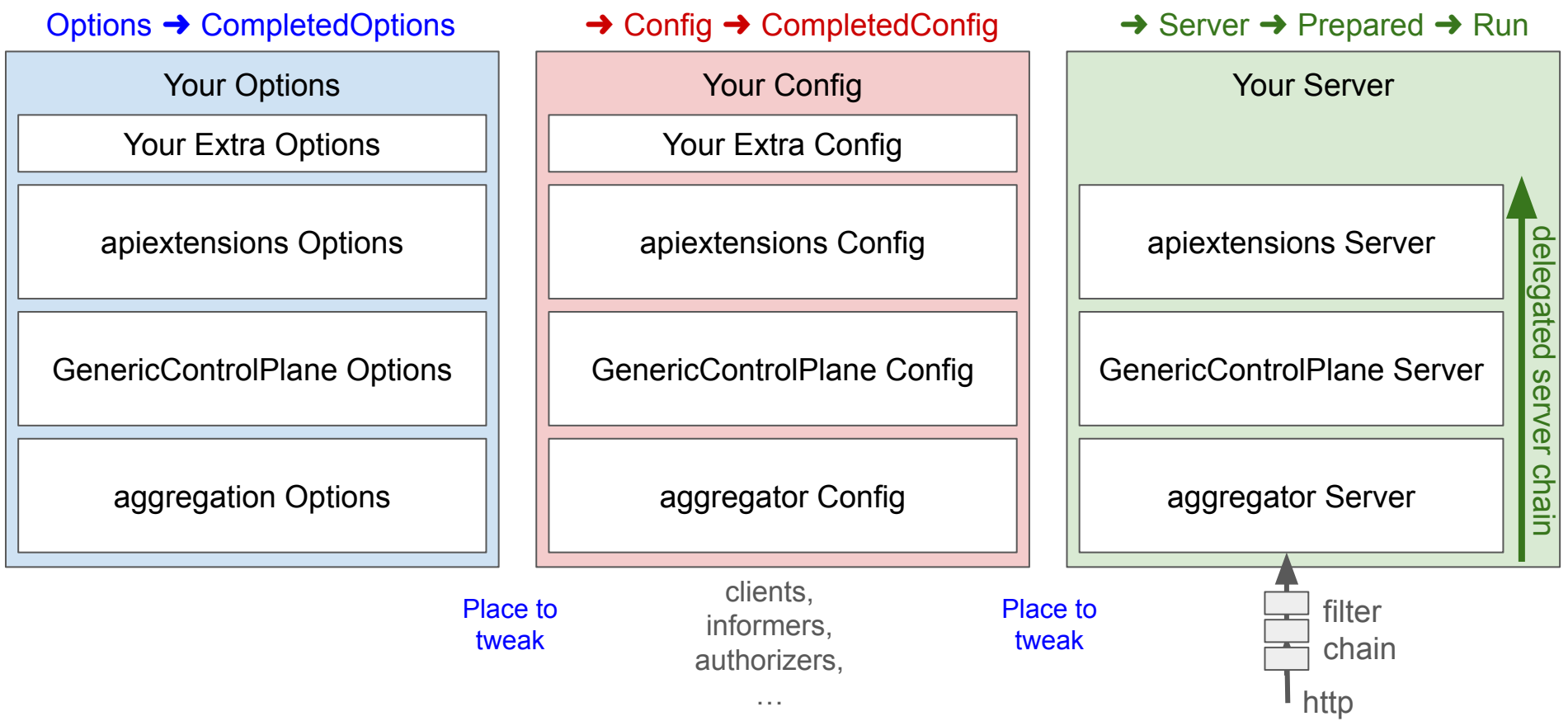
Everywhere in Kubernetes.
It is how most binaries and in particular kube-apiserver are built.
Pattern to tweak and compose.

```
o := NewOptions()
c := NewConfig(o.Complete())
s := NewServer(c.Complete())
s.Prepare().Run()
```

< Here you can tweak o
< Here you can tweak c
< Here you can tweak s

M

The Options-Config-Server Pattern



M

The Options-Config-Server Pattern

Options ➜ CompletedOptions ➜ Config ➜ CompletedConfig ➜ Server ➜ Prepared ➜ Run

Before 1.31, nearly everywhere in Kubernetes.

"Only" exception: kube-apiserver itself. It was a giant 💩, impossible to tweak or compose.

- "Generic controlplane" effort is about this for kube-apiserver:

```
o := NewOptions()
c := NewConfig(o.Complete())
s := NewServer(c.Complete())
s.Prepare().Run()
```

<Here you can tweak o

<Here you can tweak c

<Here you can tweak s

Show me the code


```go
// CreateServerChain creates the apiservers connected via delegation.
func CreateServerChain(config CompletedConfig) (*aggregatorapiserver.APIAggregator, error) {
    notFoundHandler := notfoundhandler.New(config.KubeAPIs.ControlPlane.Generic.Serializer, genericapifilters.NoMuxAndDiscoveryIncompleteKey)
    apiExtensionsServer, err := config.ApiExtensions.New(genericapiserver.NewEmptyDelegateWithCustomHandler(notFoundHandler))
    if err != nil {
        return nil, err
    }
    crdAPIEnabled := config.ApiExtensions.GenericConfig.MergedResourceConfig.ResourceEnabled(apiextensionsv1.SchemeGroupVersion.WithResource(

    kubeAPIServer, err := config.KubeAPIs.New(apiExtensionsServer.GenericAPIServer)
    if err != nil {
        return nil, err
    }

    // aggregator comes last in the chain
    aggregatorServer, err := controlplaneapiserver.CreateAggregatorServer(
        config.Aggregator,
        kubeAPIServer.ControlPlane.GenericAPIServer,
        apiExtensionsServer.Informers.Apiextensions().V1().CustomResourceDefinitions(),
        crdAPIEnabled,
        apiVersionPriorities,
    )
    if err != nil {
        // we don't need special handling for innerStopCh because the aggregator server doesn't create any go routines
        return nil, err
    }

    return aggregatorServer, nil
}
```

https://github.com/kcp-dev/generic-controlplane

The Options-Config-Server Pattern

Options ➜ CompletedOptions ➜ Config ➜ CompletedConfig ➜ Server ➜ Prepared ➜ Run

Your Options

Your Extra Options

GenericControlPlane Options

apiextensions Options

aggregation Options

Your Config

Your Extra Config

Gene

a

Your Server

apiextensions Server

GenericControlPlane Server

aggregator Server

delegated server chain

Place to tweak

clients, informers, authorizers, …

Place to tweak

filter chain

http

Flags.

No go routines. All static.

Running server.

M

Show me the code


```go
// CreateServerChain creates the apiservers connected via delegation.
func CreateServerChain(config options.CompletedConfig) (*aggregatorapiserver.APIAggregator, error) {
    notFoundHandler := notfoundhandler.New(config.ControlPlane.Generic.Serializer, genericapifilters.NoMuxAndDiscoveryIncompleteKey)
    apiExtensionsServer, err := config.APIExtensions.New(genericapiserver.NewEmptyDelegateWithCustomHandler(notFoundHandler))
    if err != nil {
        return nil, err
    }

    controlPlaneAPIServer, err := config.ControlPlane.New("generic-controlplane", apiExtensionsServer.GenericAPIServer)
    if err != nil {
        return nil, err
    }

    aggregatorServer, err := controlplaneapiserver.CreateAggregatorServer(
        config.Aggregator,
        controlPlaneAPIServer.GenericAPIServer,
        apiExtensionsServer.Informers.Apiextensions().V1().CustomResourceDefinitions(),
        false,
        controlplaneapiserver.DefaultGenericAPIServicePriorities())

    return aggregatorServer, nil
}
```

CRDs

kube APIs

configmaps, secrets, RBAC, …

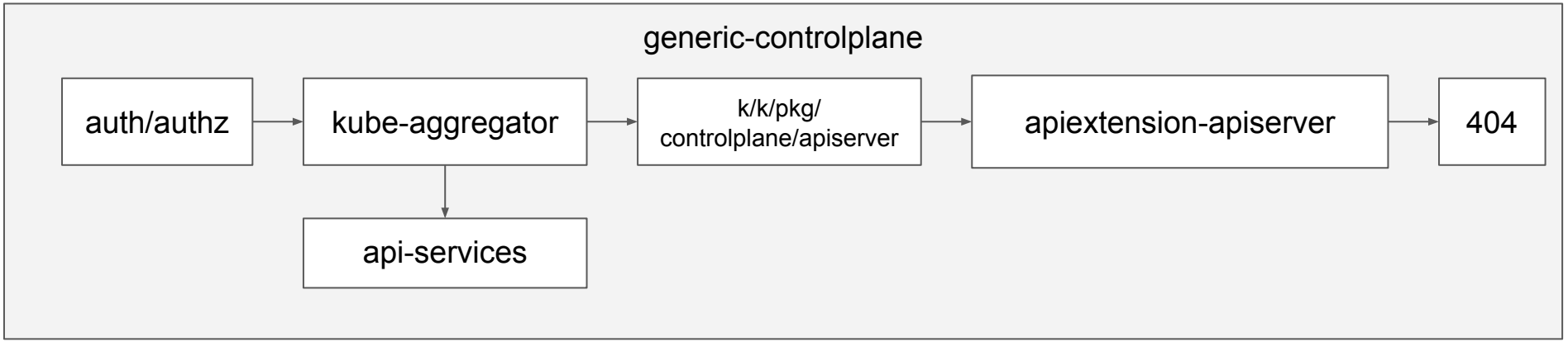
APIServices

aggregated apiservers

OpenAPI and discovery aggregation

https://github.com/kcp-dev/generic-controlplane

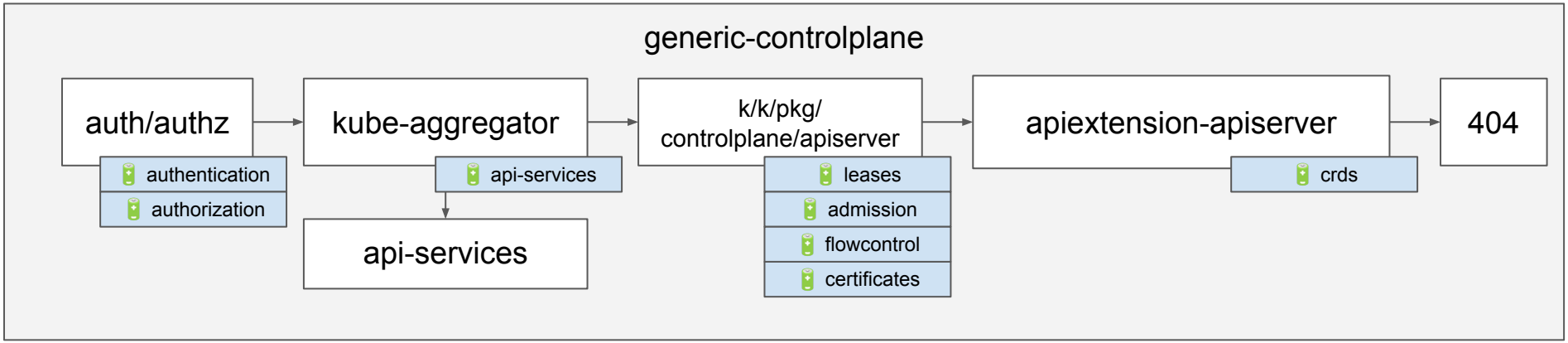
Show me the code



https://github.com/kcp-dev/generic-controlplane

prototype

Show me the code



```
const (
    // BatteryAll is the name of the all batteries.
    BatteryAll Battery = "all"
    // BatteryLeases is the name of the lease battery.
    BatteryLeases Battery = "leases"
    // BatteryAuthentication is the name of the authentication battery.
    BatteryAuthentication Battery = "authentication"
    // BatteryAuthorization is the name of the authorization battery.
    BatteryAuthorization Battery = "authorization"
    // BatteryAdmission is the name of the admission battery.
    BatteryAdmission Battery = "admission"
    // BatteryFlowControl is the name of the flow control battery.
    BatteryFlowControl Battery = "flowcontrol"      You, last week • add
    // BatteryCRDs is the name of the CRD battery.
    BatteryCRDs Battery = "crds"
    // BatteryCertificates is the name of the certificates battery.
    BatteryCertificates Battery = "certificates"
    // BatteryAPIServices is the name of the API services battery.
    BatteryAPIServices Battery = "apiservices"
)
```

https://github.com/kcp-dev/generic-controlplane

prototype

- Pluggable DIY-Kubernetes-like API Server out of the box

M

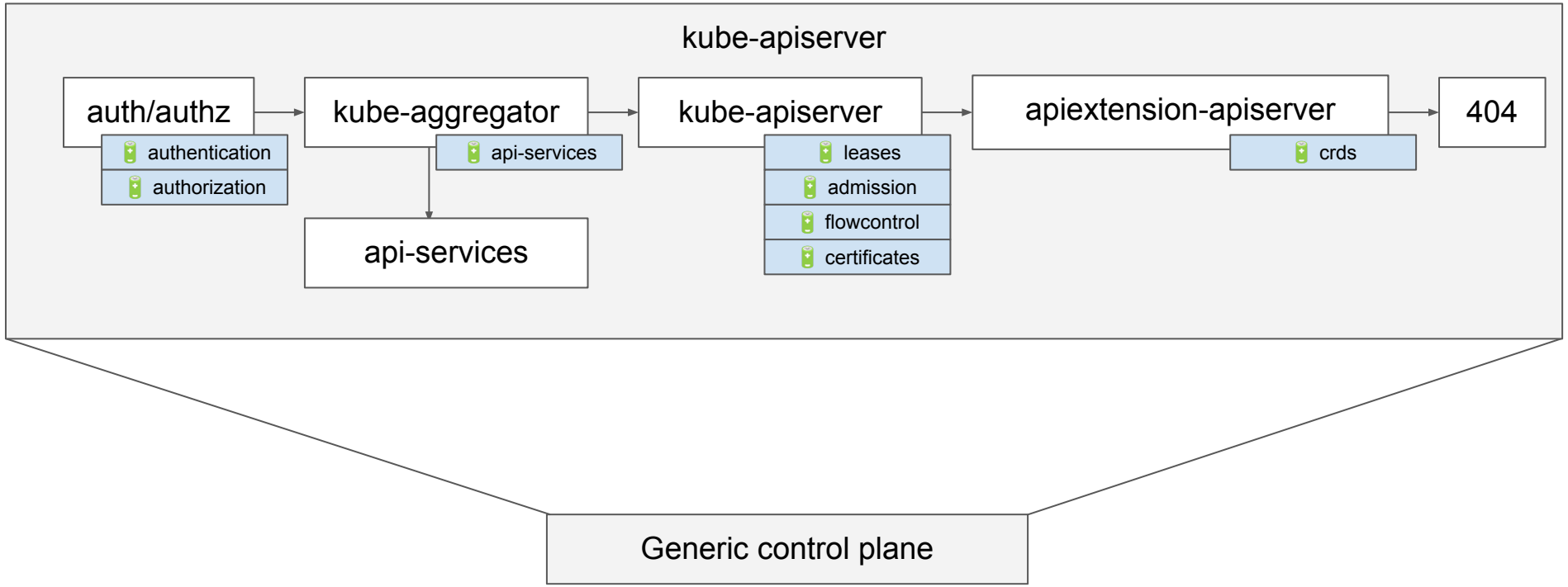
Demo! - Generic Control Plane

https://github.com/kcp-dev/generic-controlplane

But what is the difference? O_o

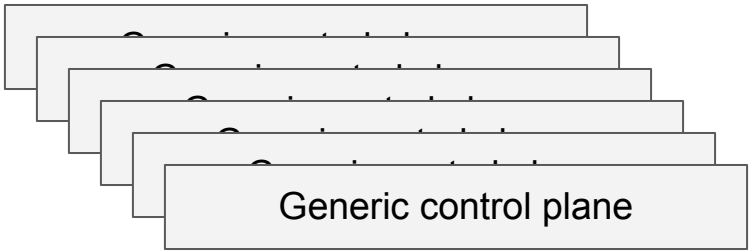
KCP > Generic Control Plane

KCP > Generic Control Plane

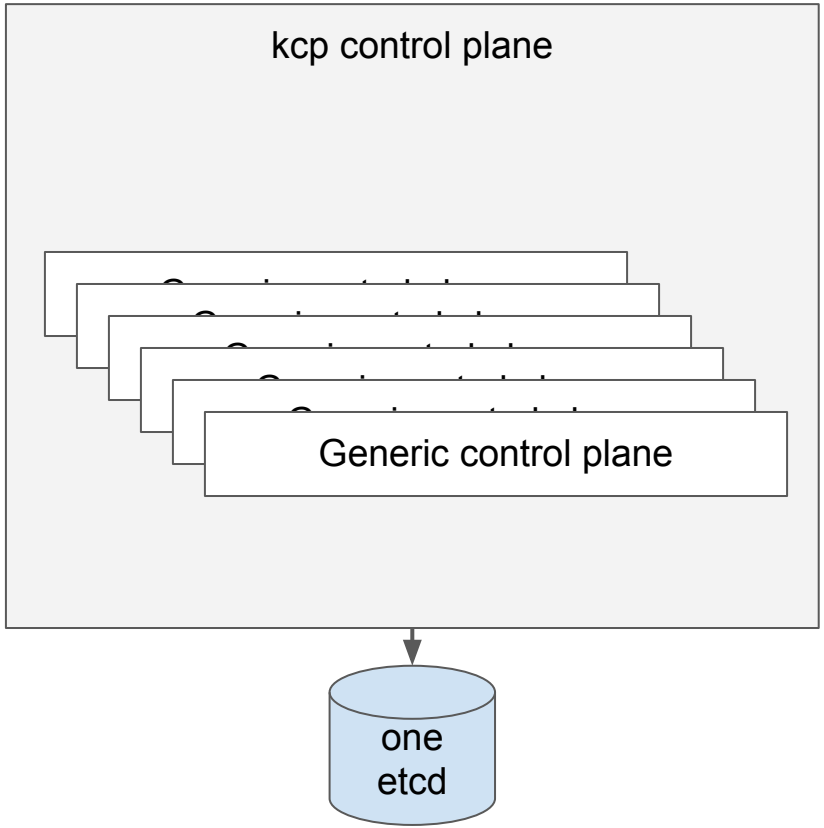


KCP > Generic Control Plane

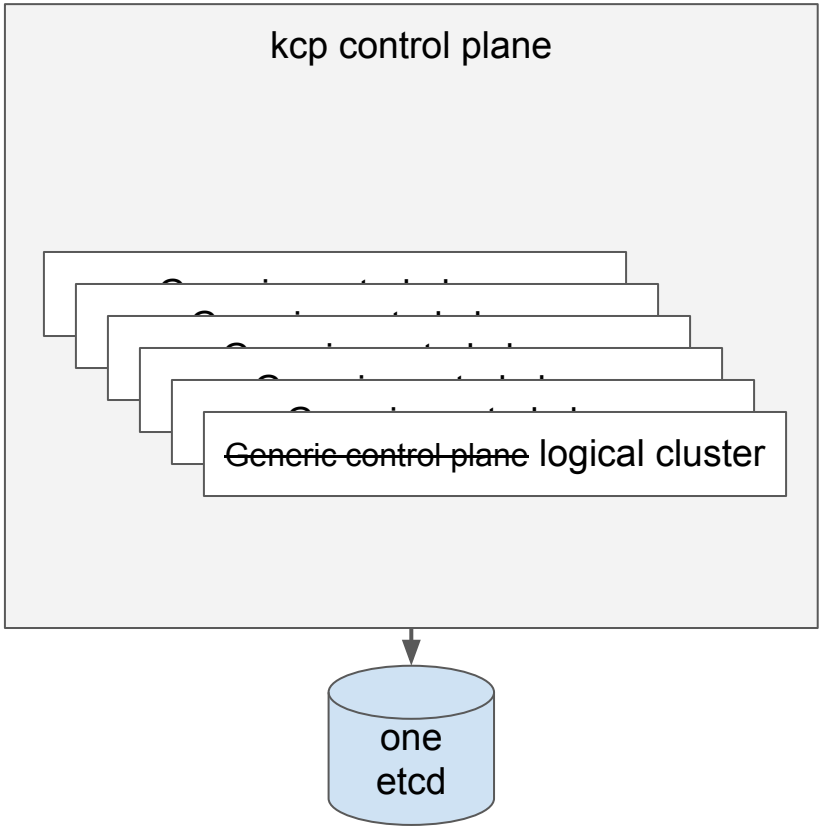
KCP > Generic Control Plane



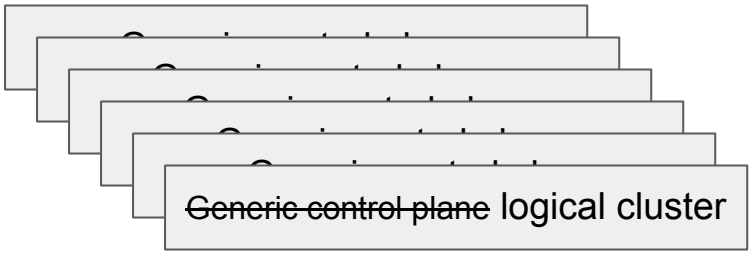
KCP > Generic Control Plane

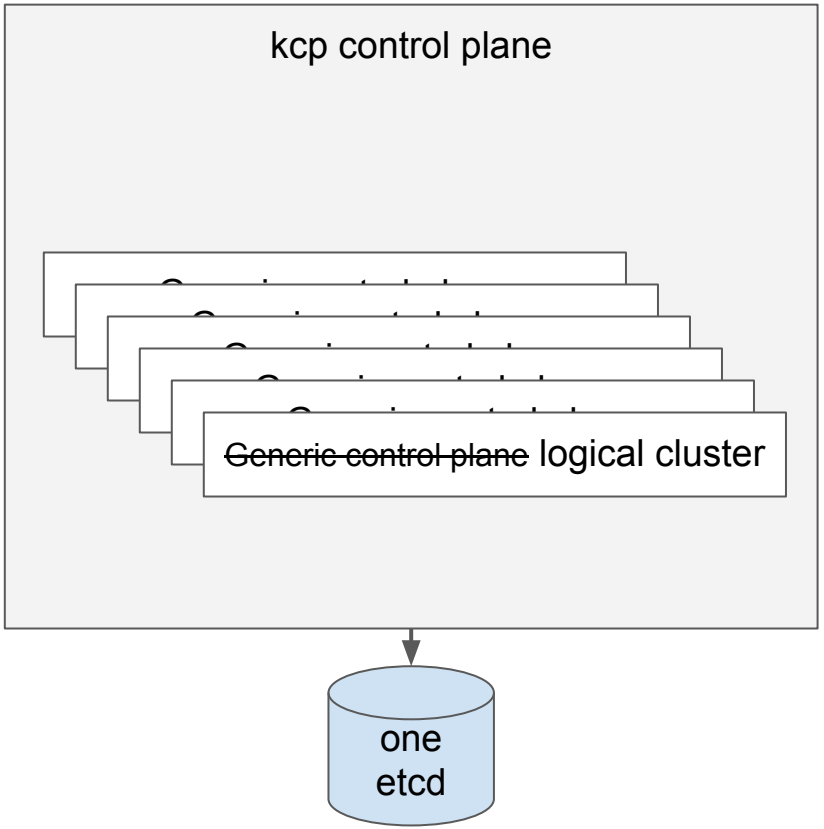


KCP > Generic Control Plane

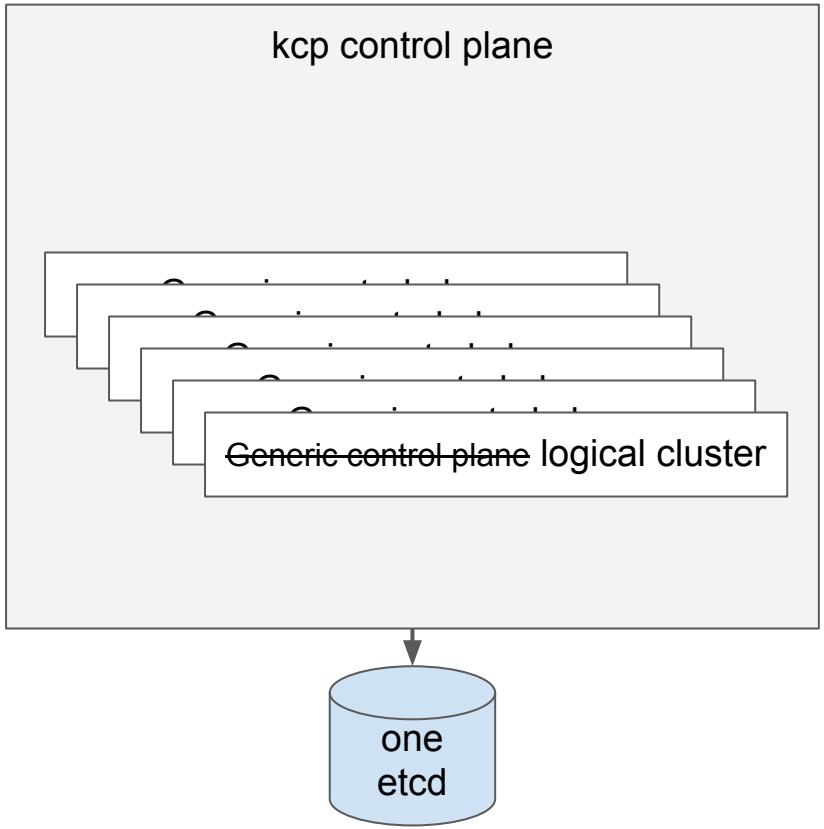


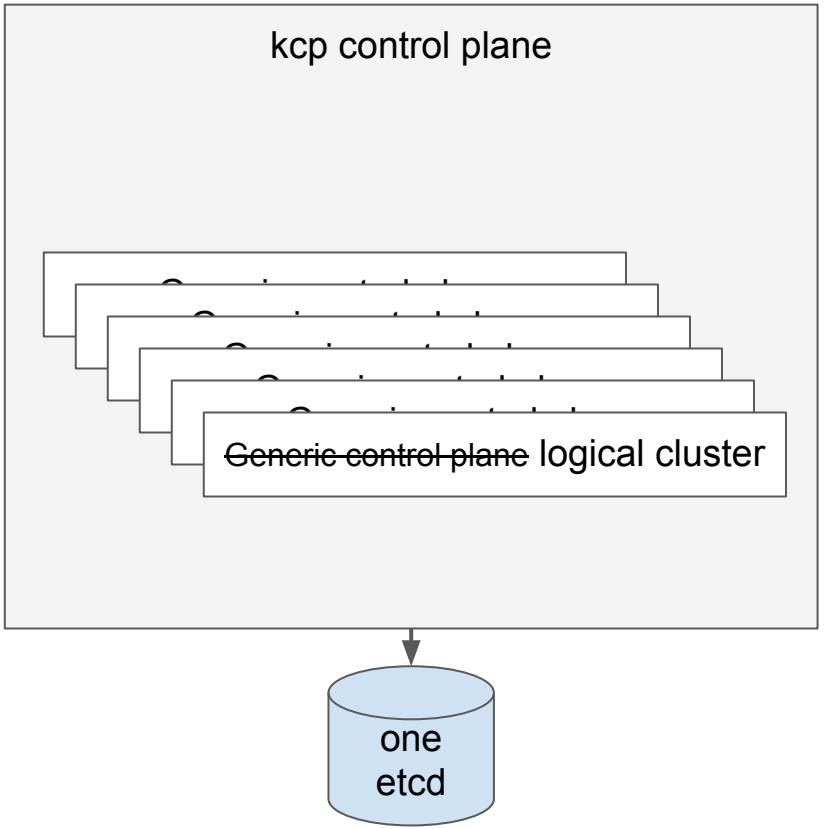
KCP > Generic Control Plane



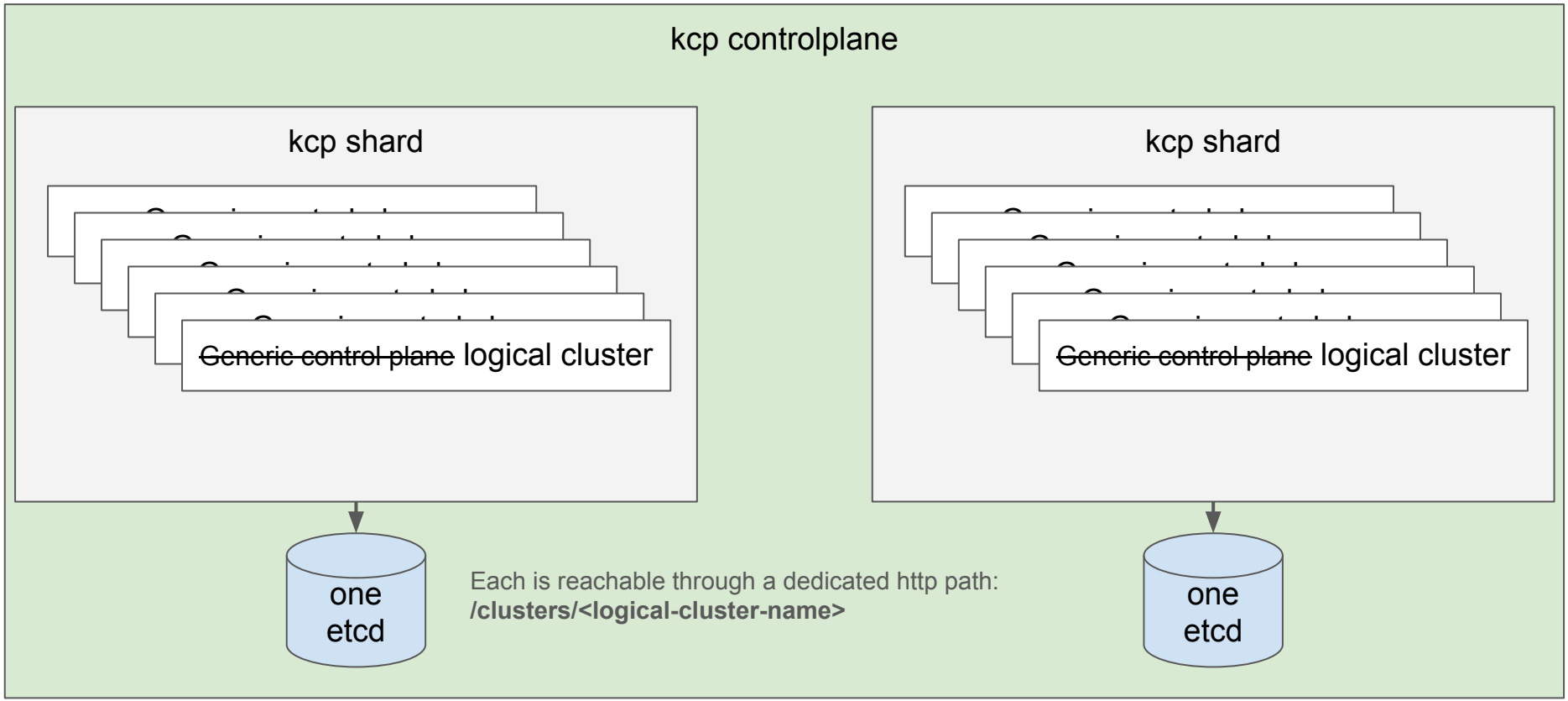


KCP > Generic Control Plane

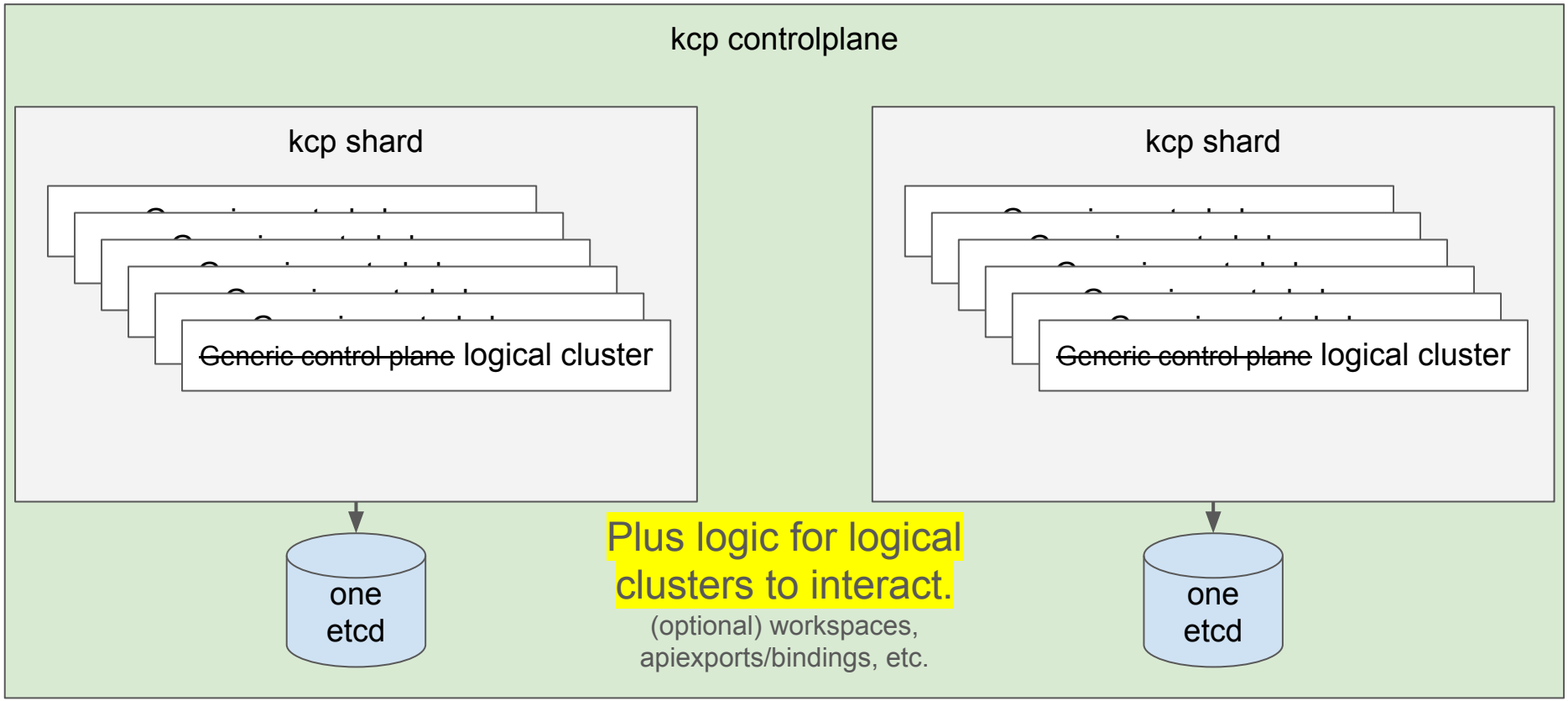




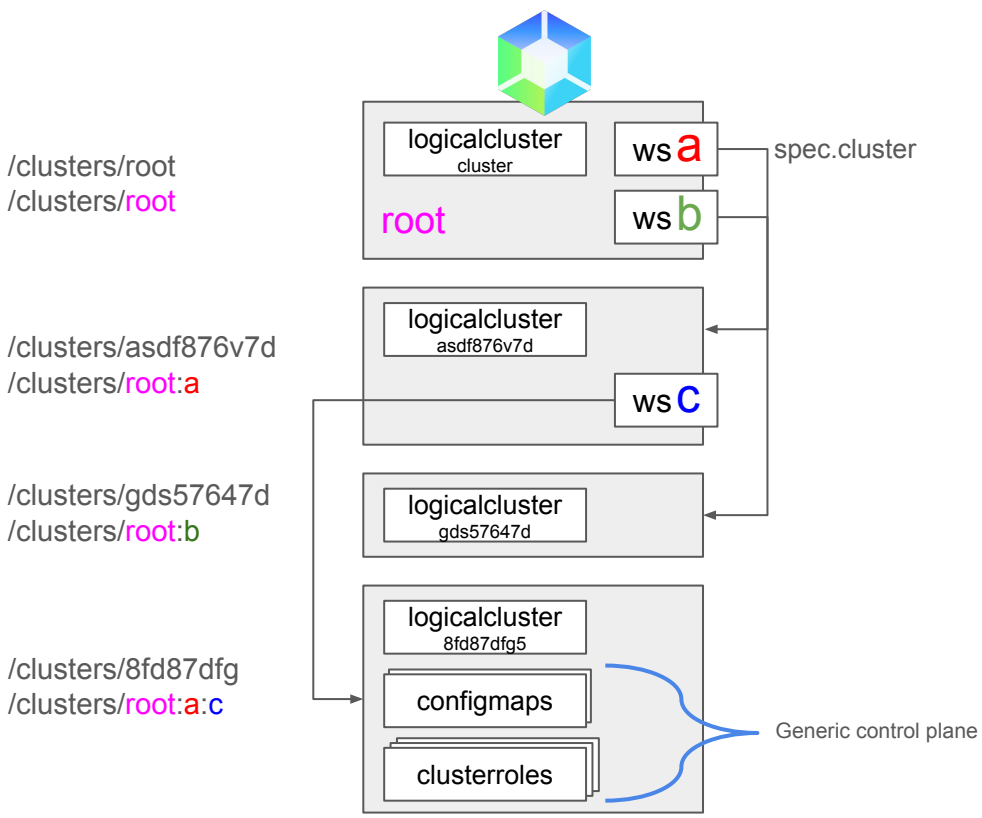
KCP > Generic Control Plane

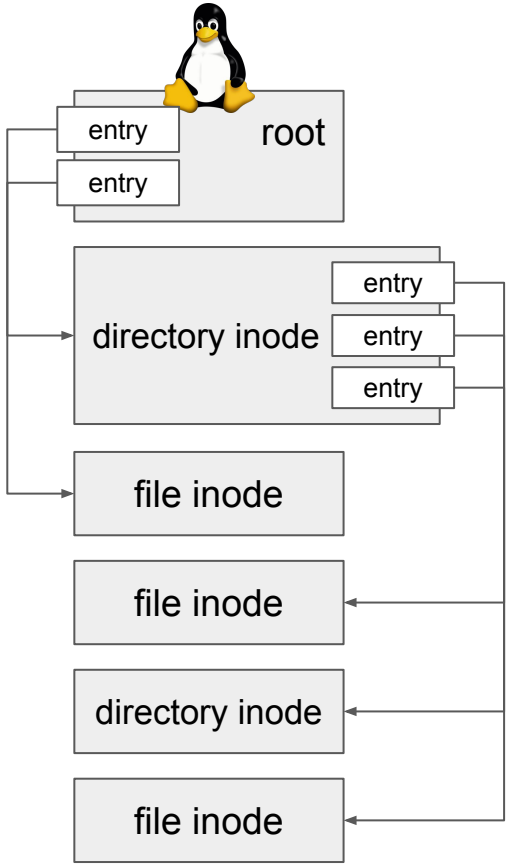


KCP > Generic Control Plane



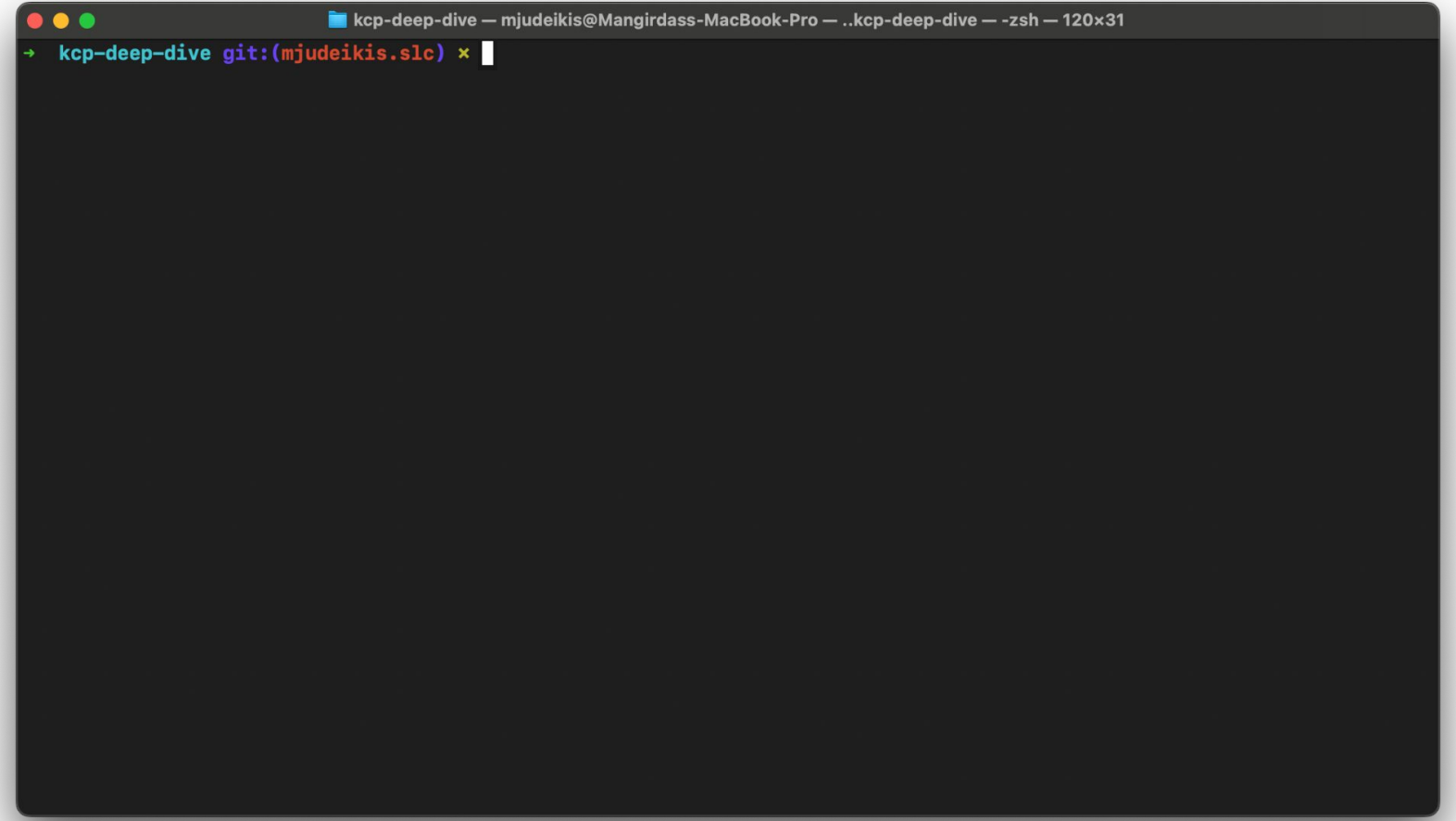
From Logical Cluster to Workspaces (optional)



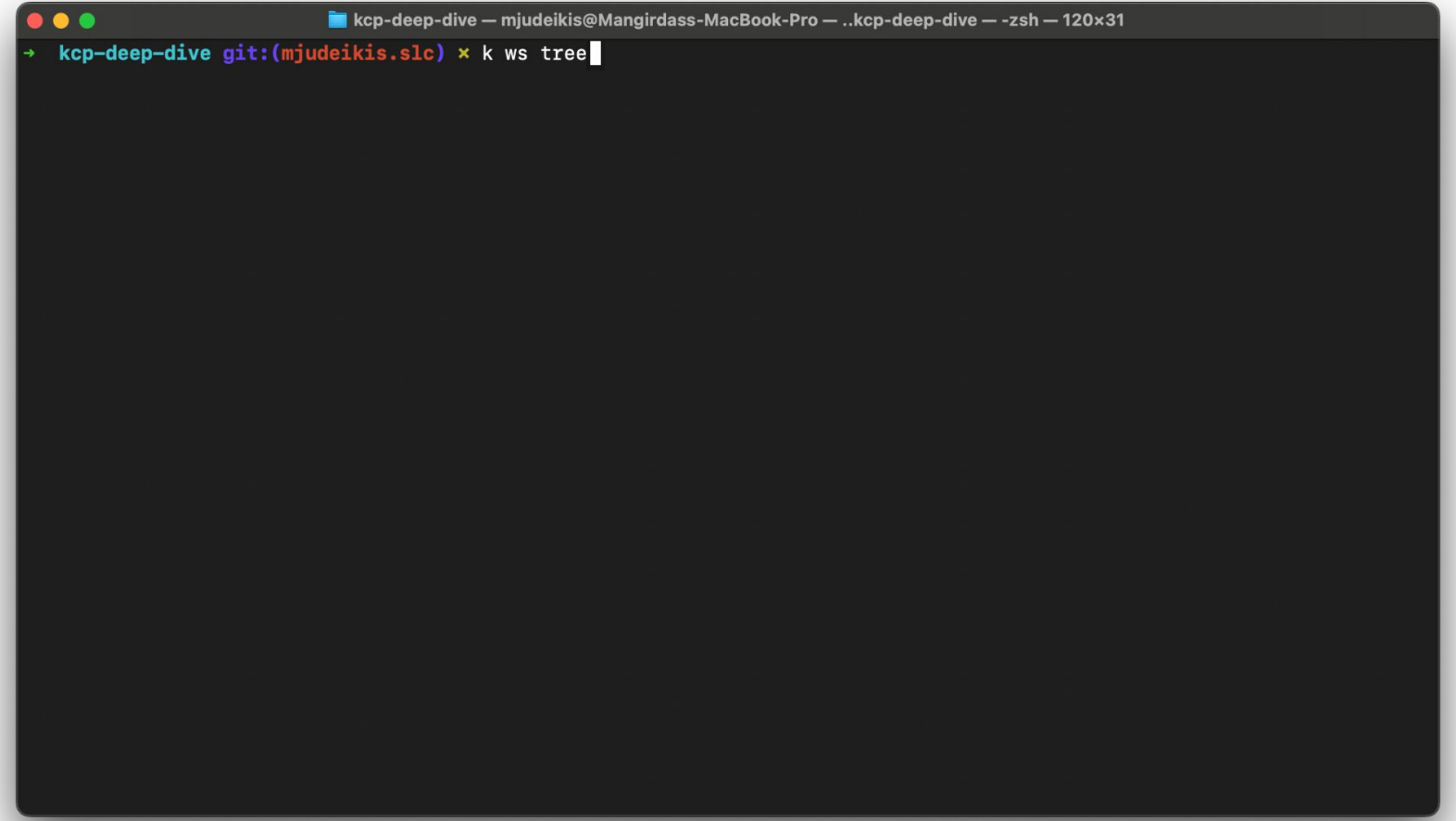


S

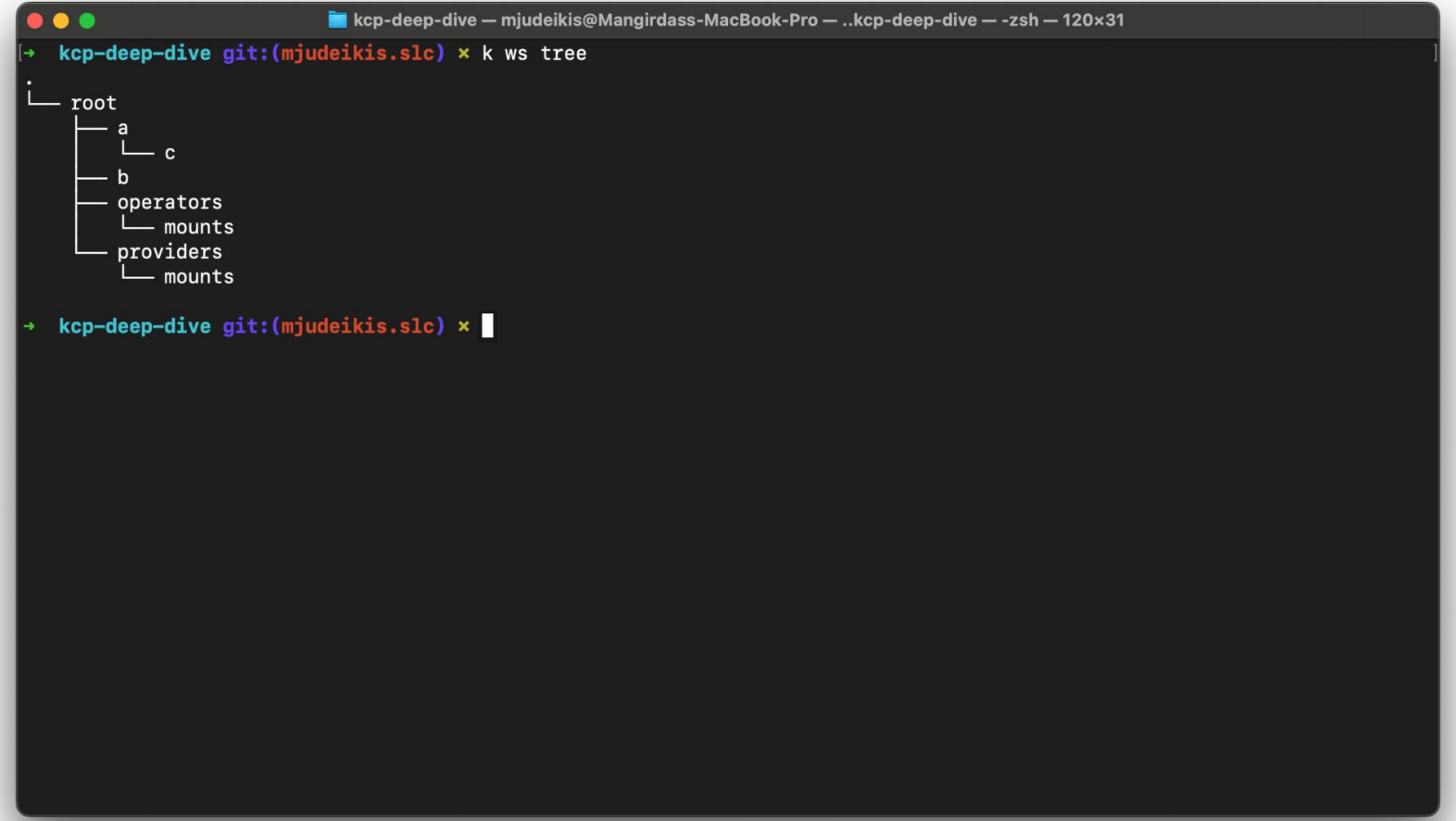
LogicalClusters



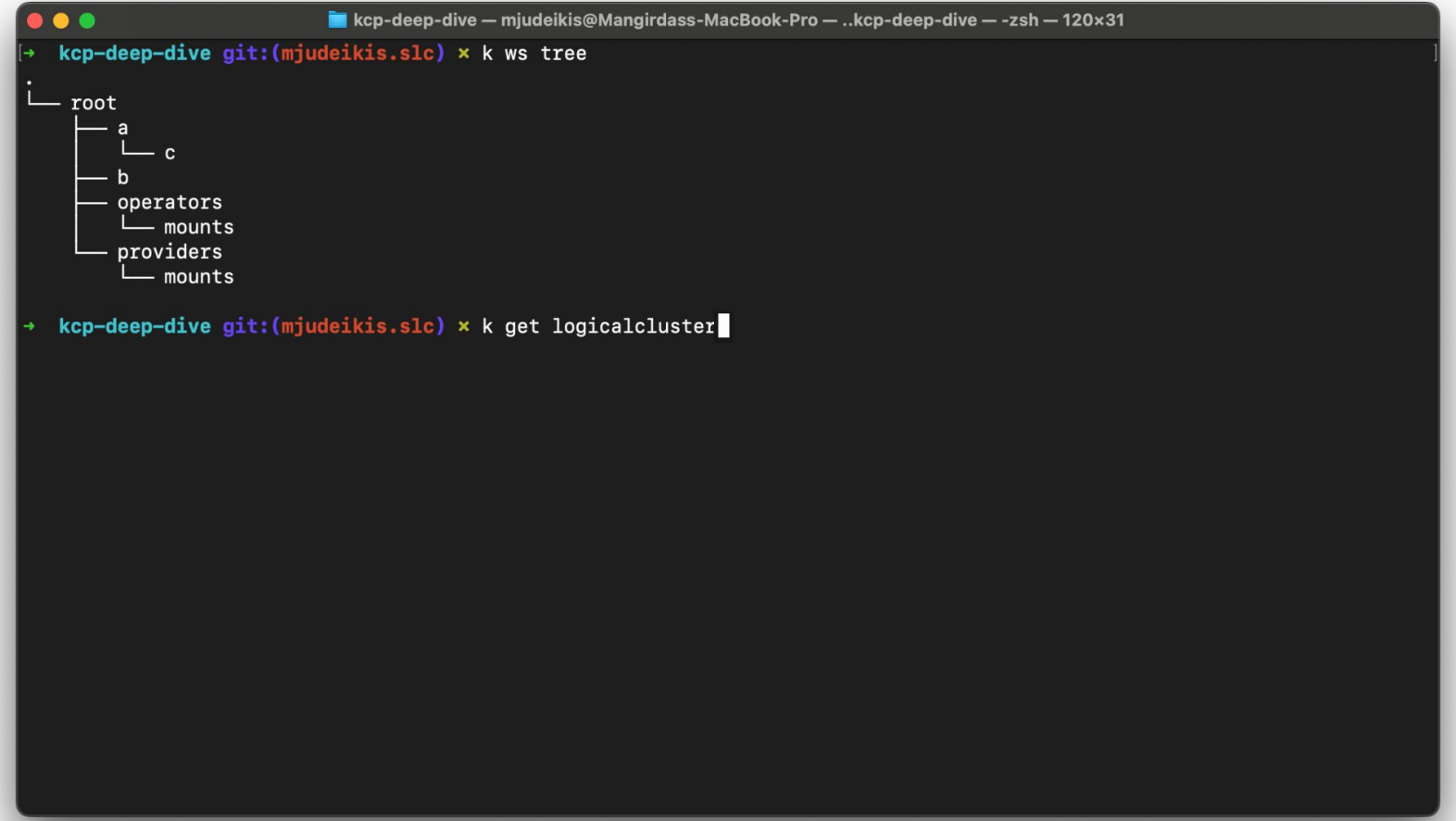
LogicalClusters



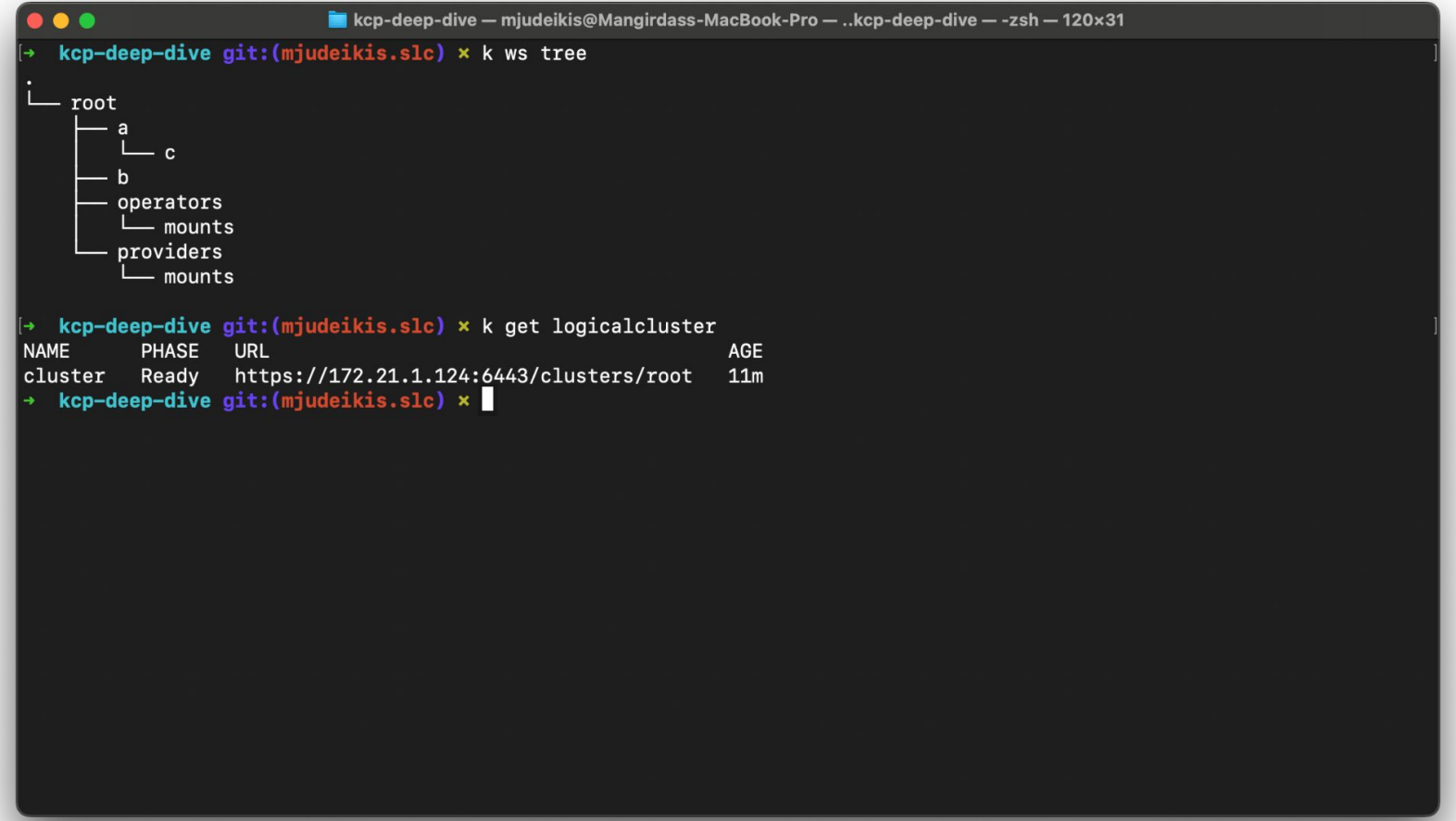
LogicalClusters



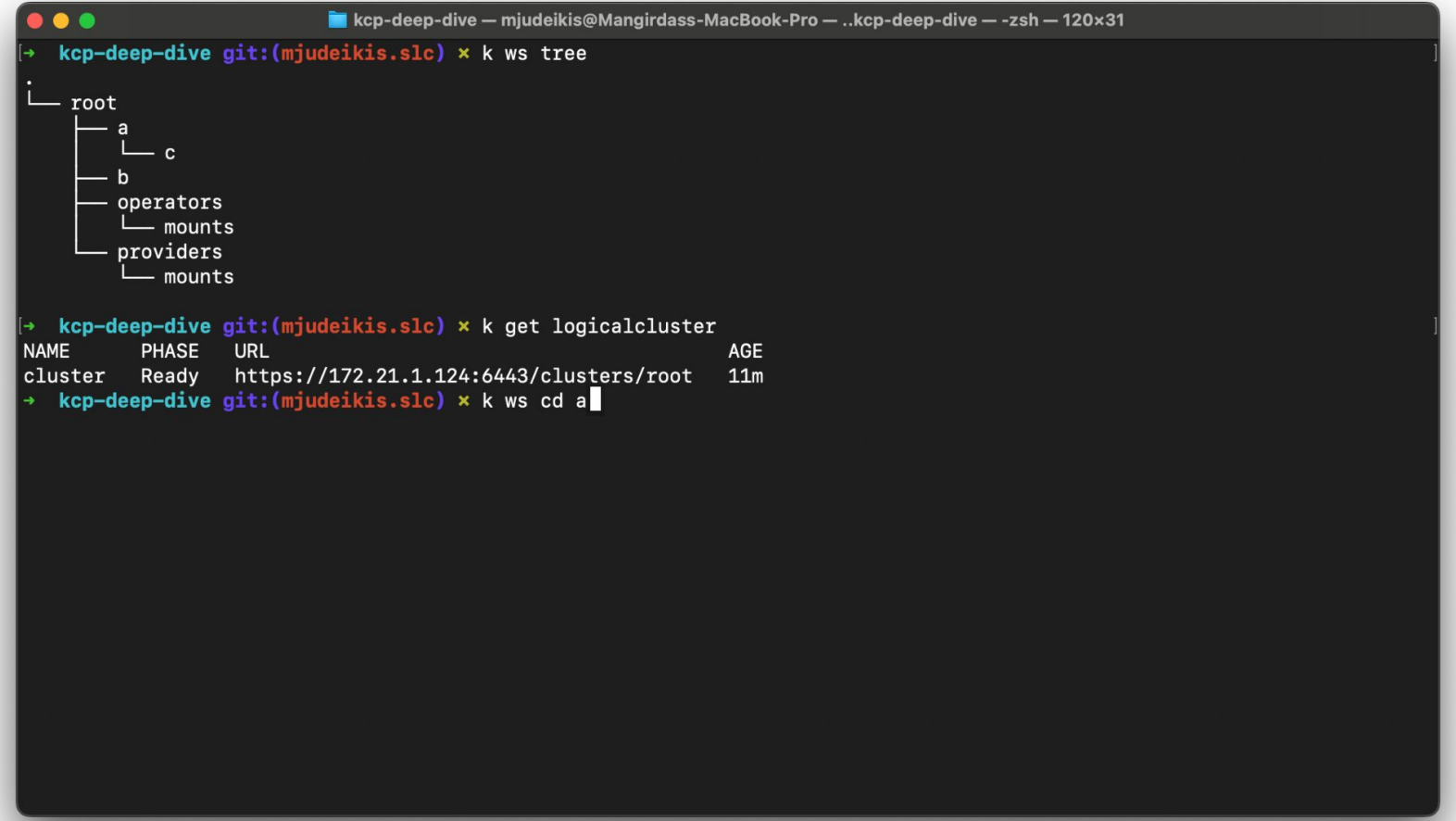
LogicalClusters



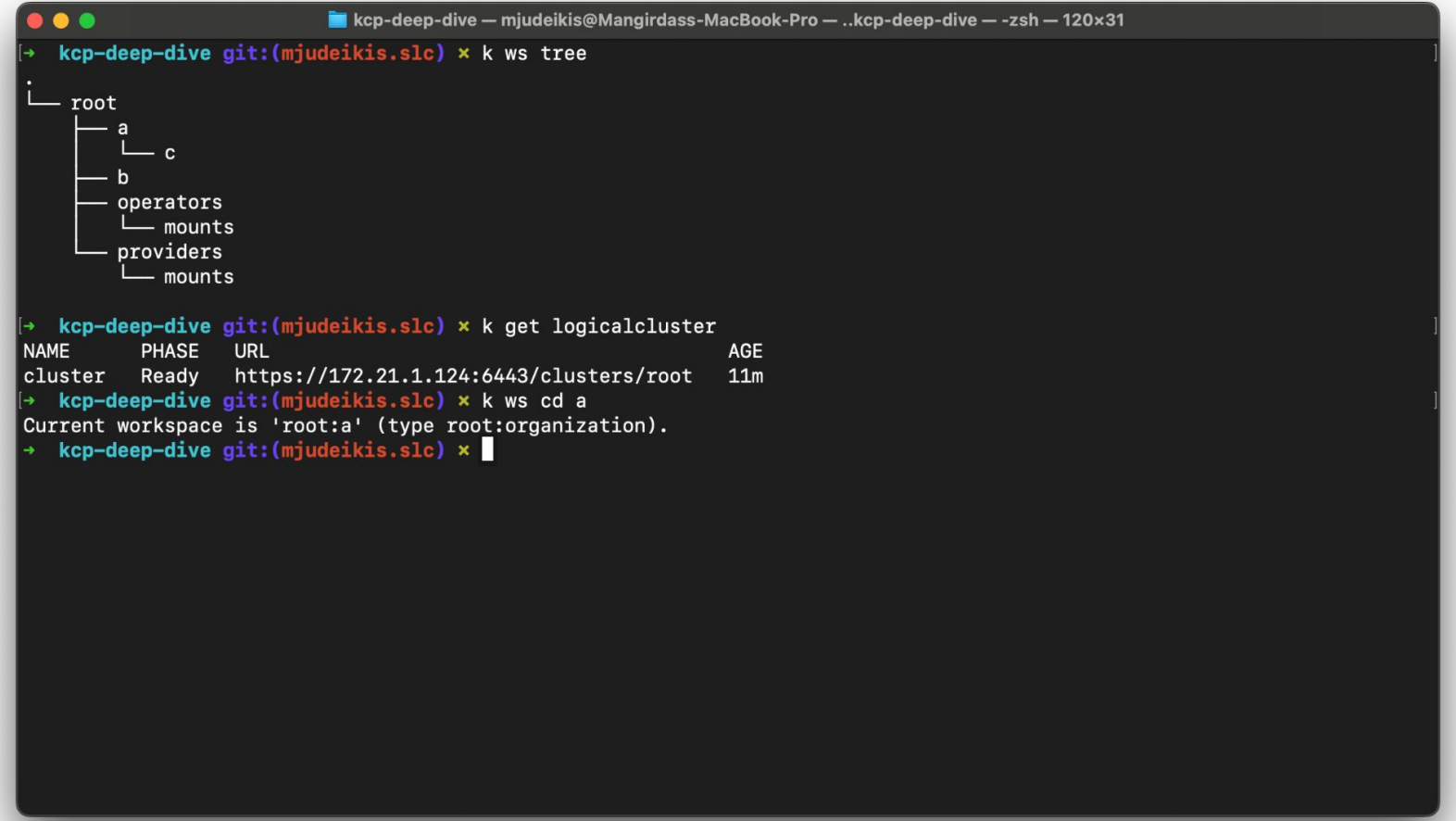
LogicalClusters



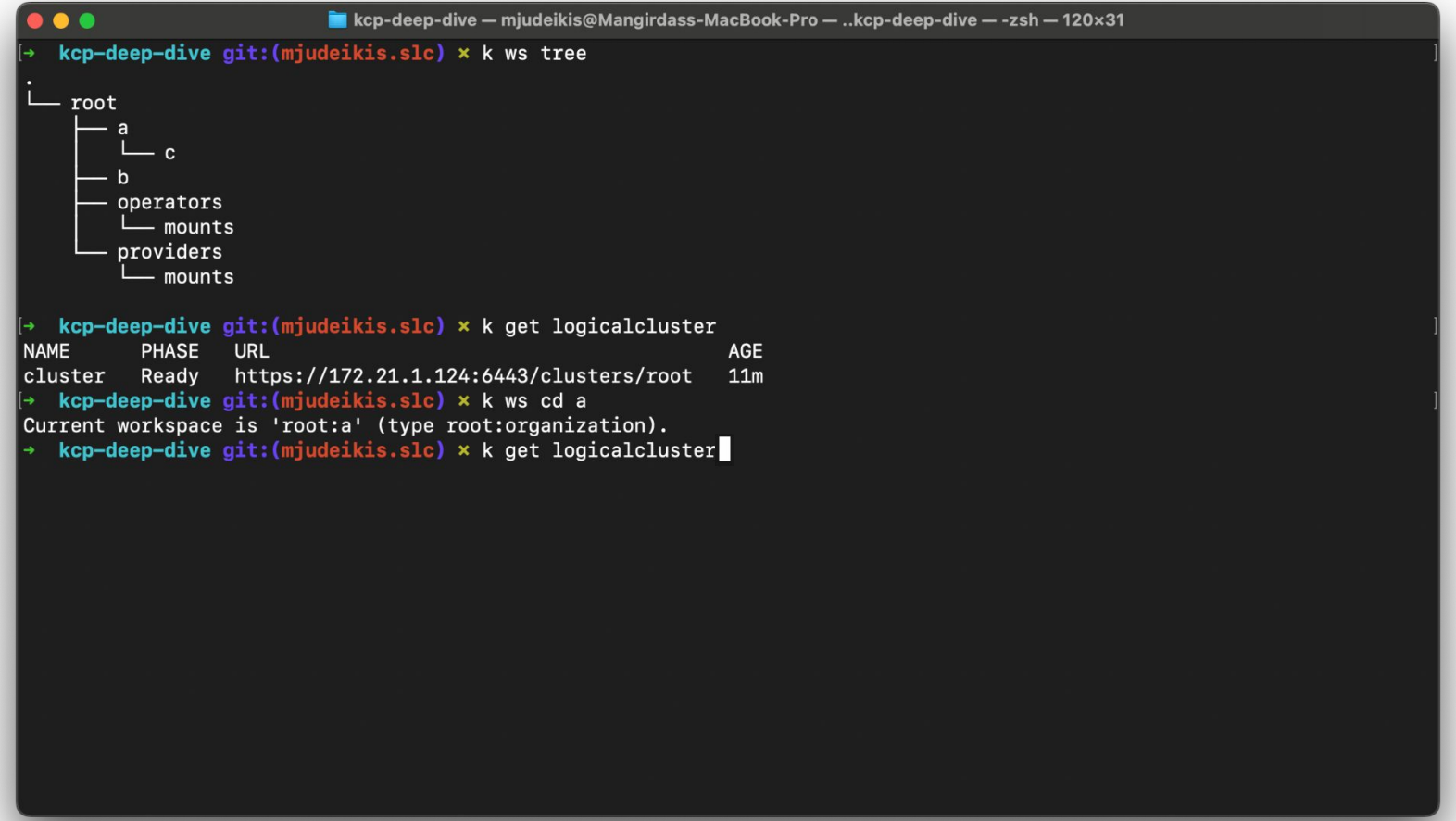
LogicalClusters



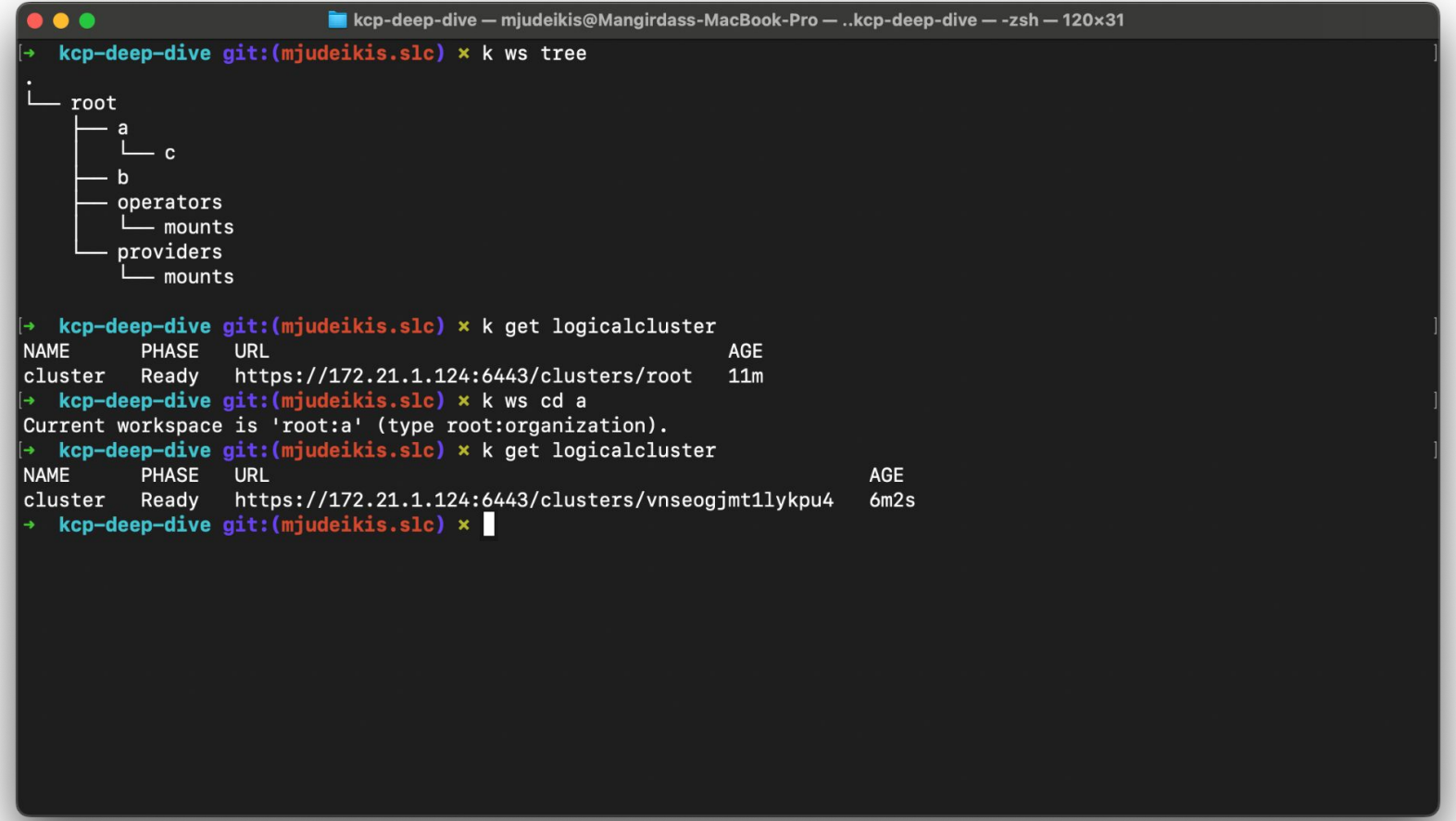
LogicalClusters



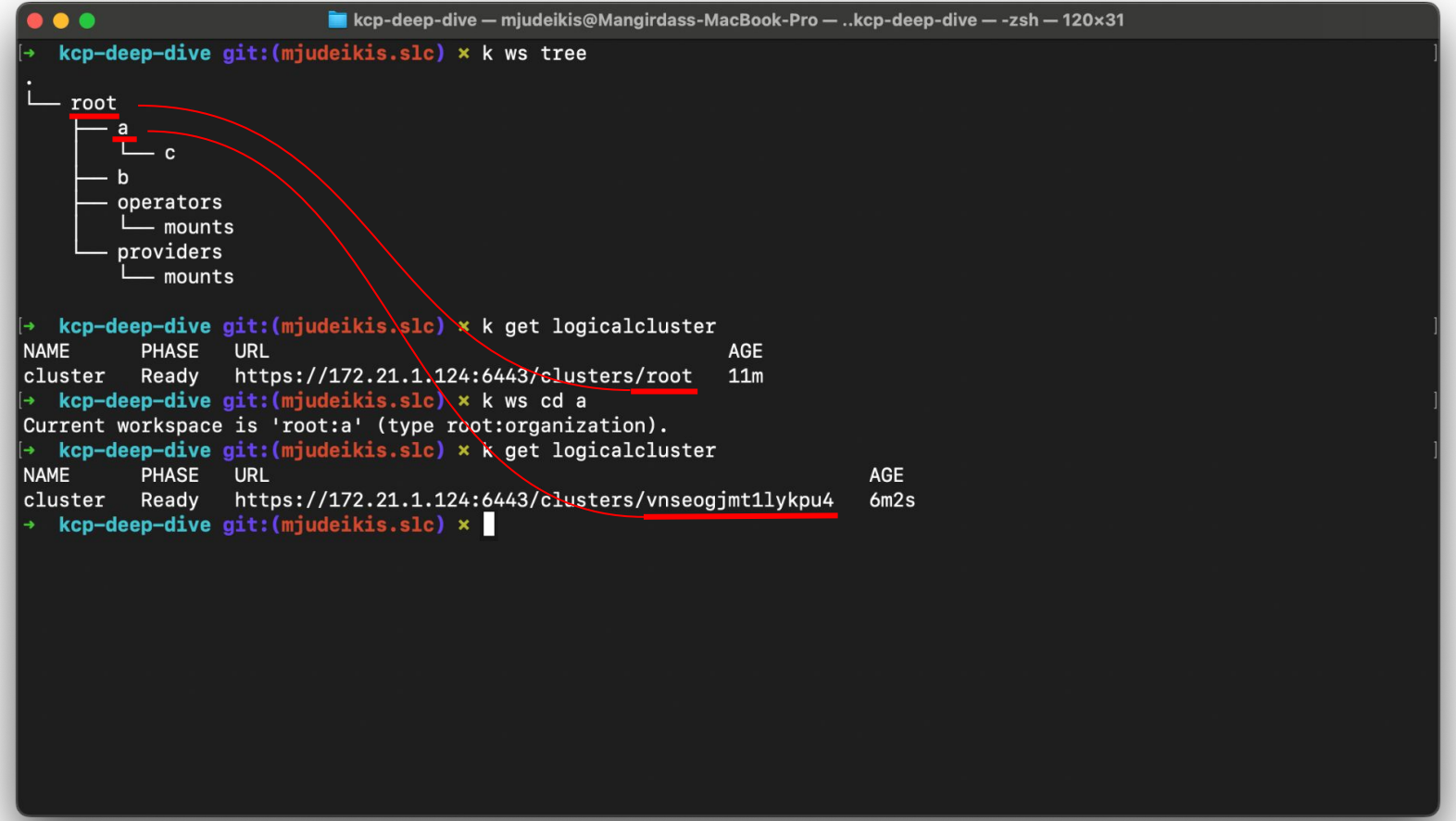
LogicalClusters



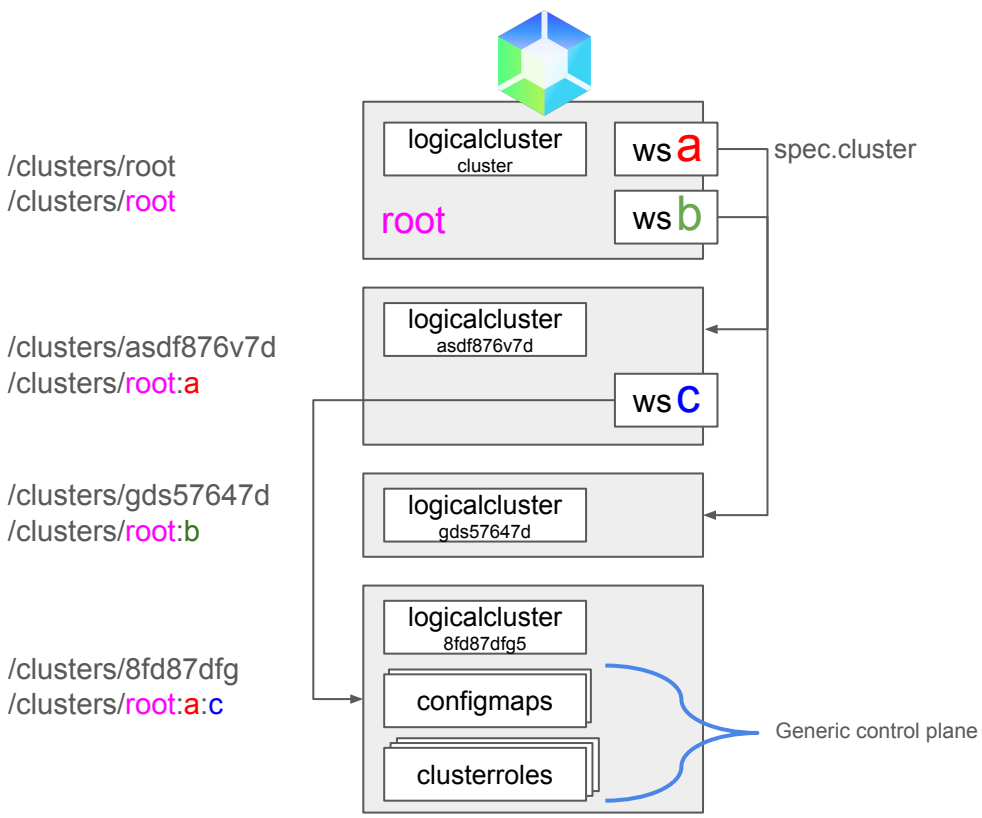
LogicalClusters

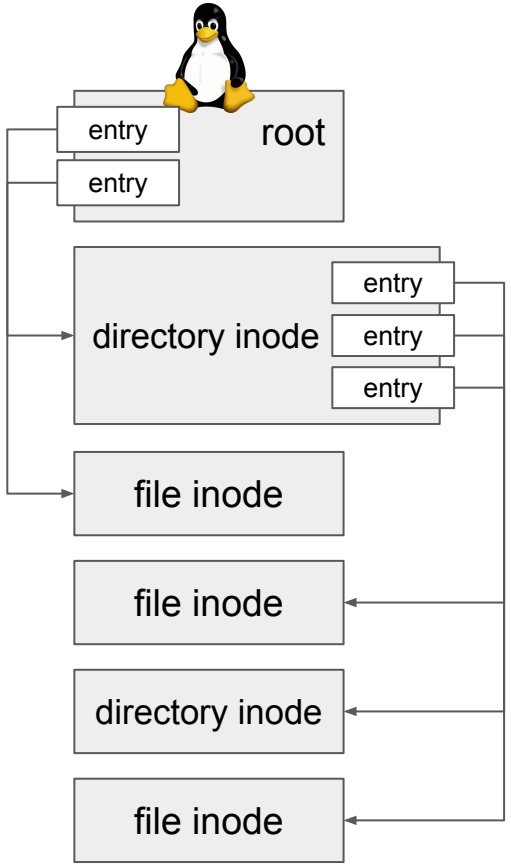


LogicalClusters



From Logical Cluster to Workspaces (optional)





M

From Logical Cluster to Workspaces (optional)

A filesystem?

/cluste
/cluste

Does it support mount?

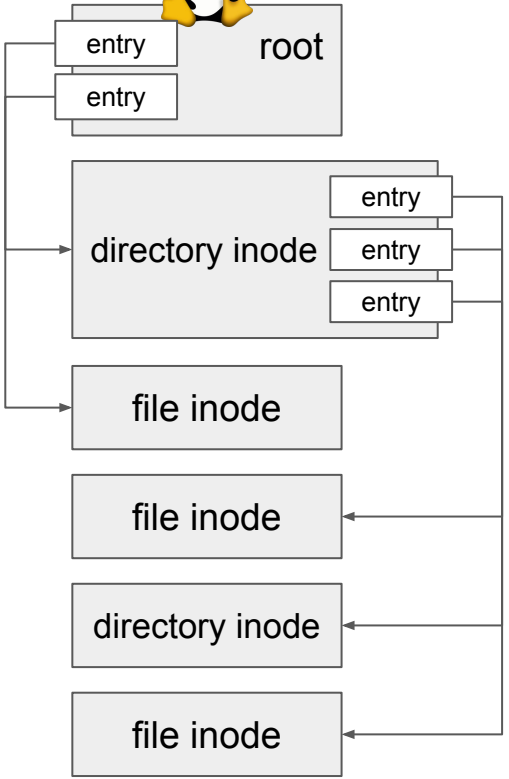
/cluste
/cluste

/cluste
/cluste

/cluste
/cluste

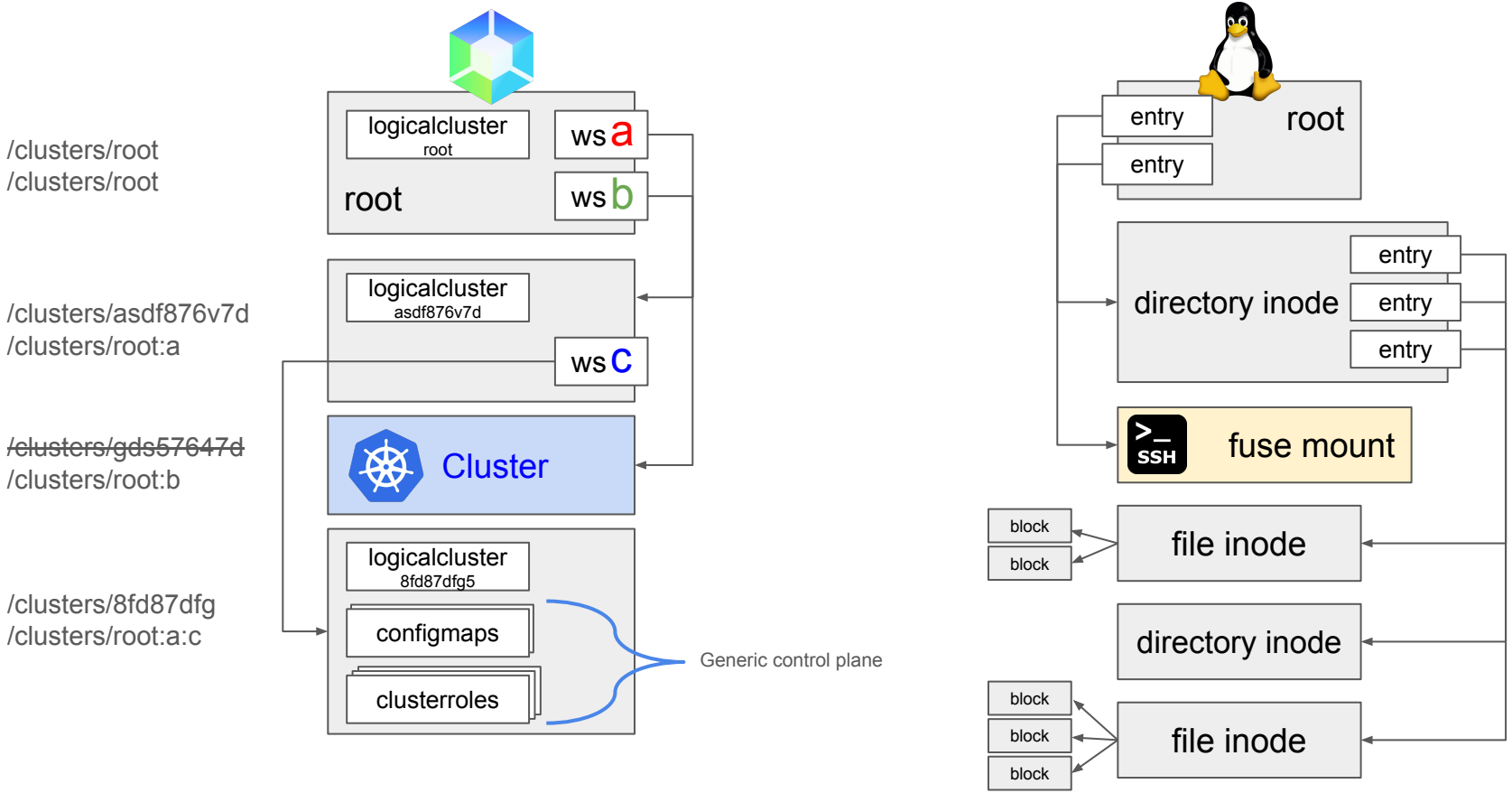
lane

clusterroles

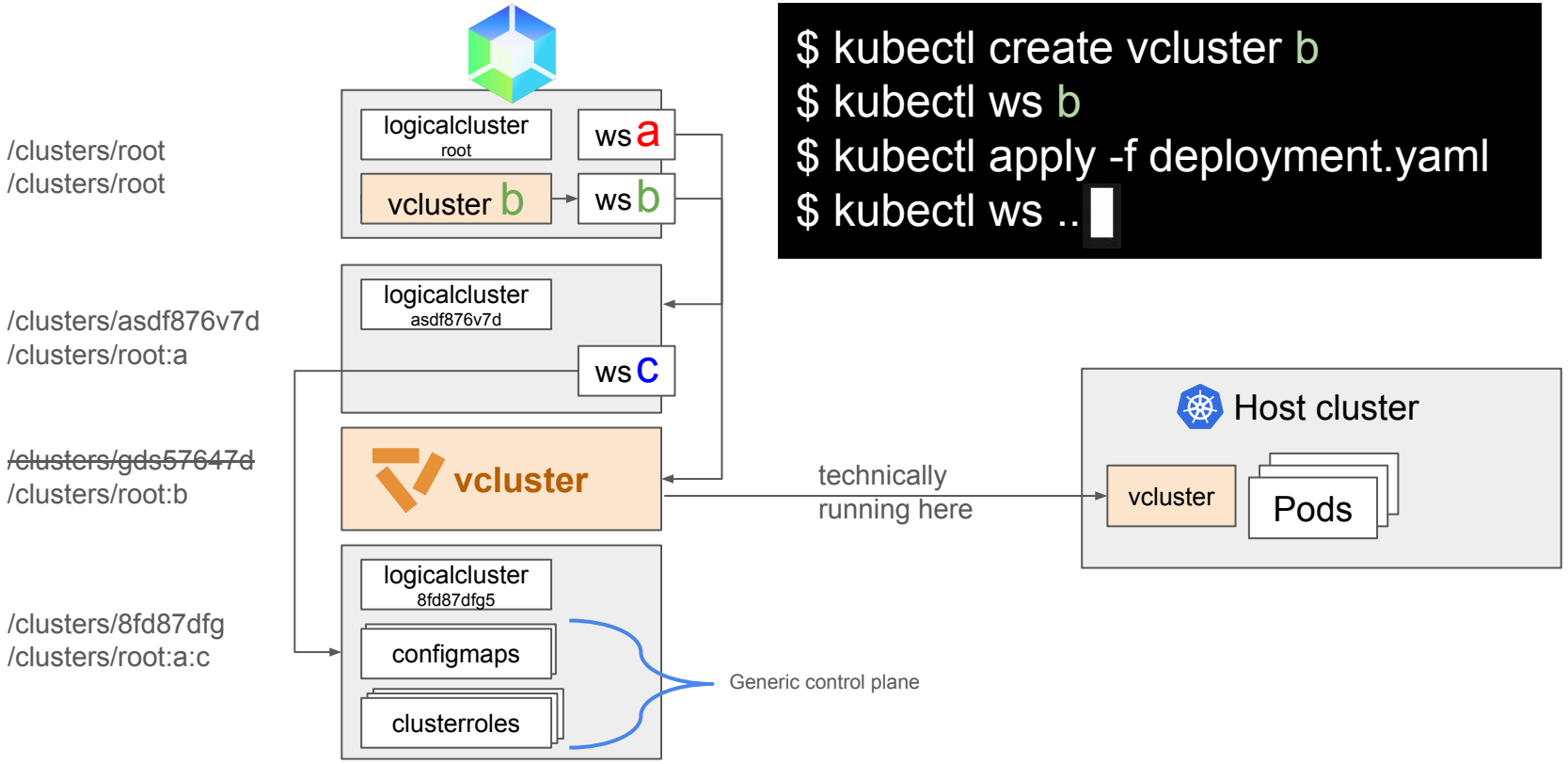


M

Mounts


/clusters/root
/clusters/root
/clusters/asdf876v7d
/clusters/root:a
/clusters/gds57647d
/clusters/root:b
/clusters/8fd87dfg
/clusters/root:a:c
logicalcluster
root
ws a
root
ws b
logicalcluster
asdf876v7d
ws c
Cluster
logicalcluster
8fd87dfg5
configmaps
clusterroles
Generic control plane
entry
entry
root
directory inode
entry
entry
entry
SSH
fuse mount
block
block
file inode
directory inode
block
block
block
file inode

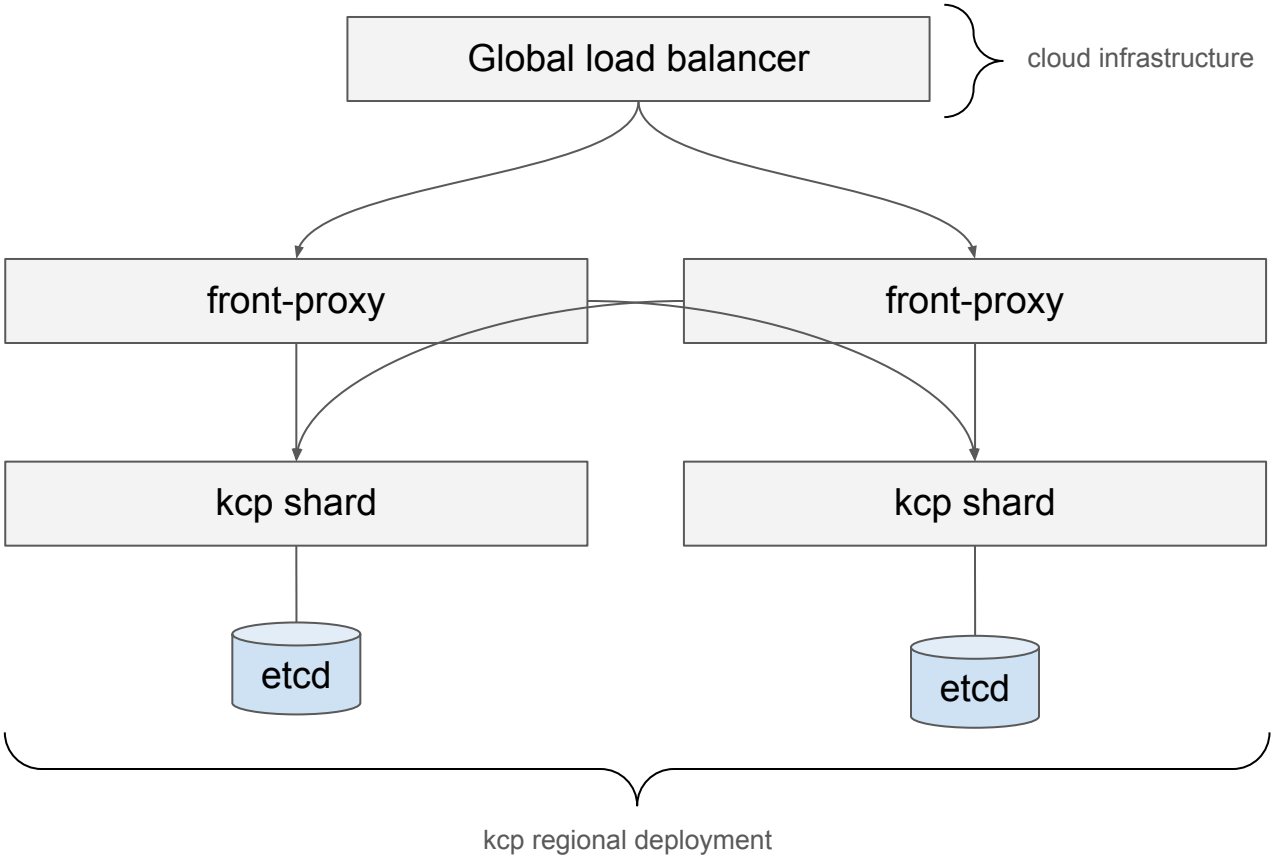
Mounts



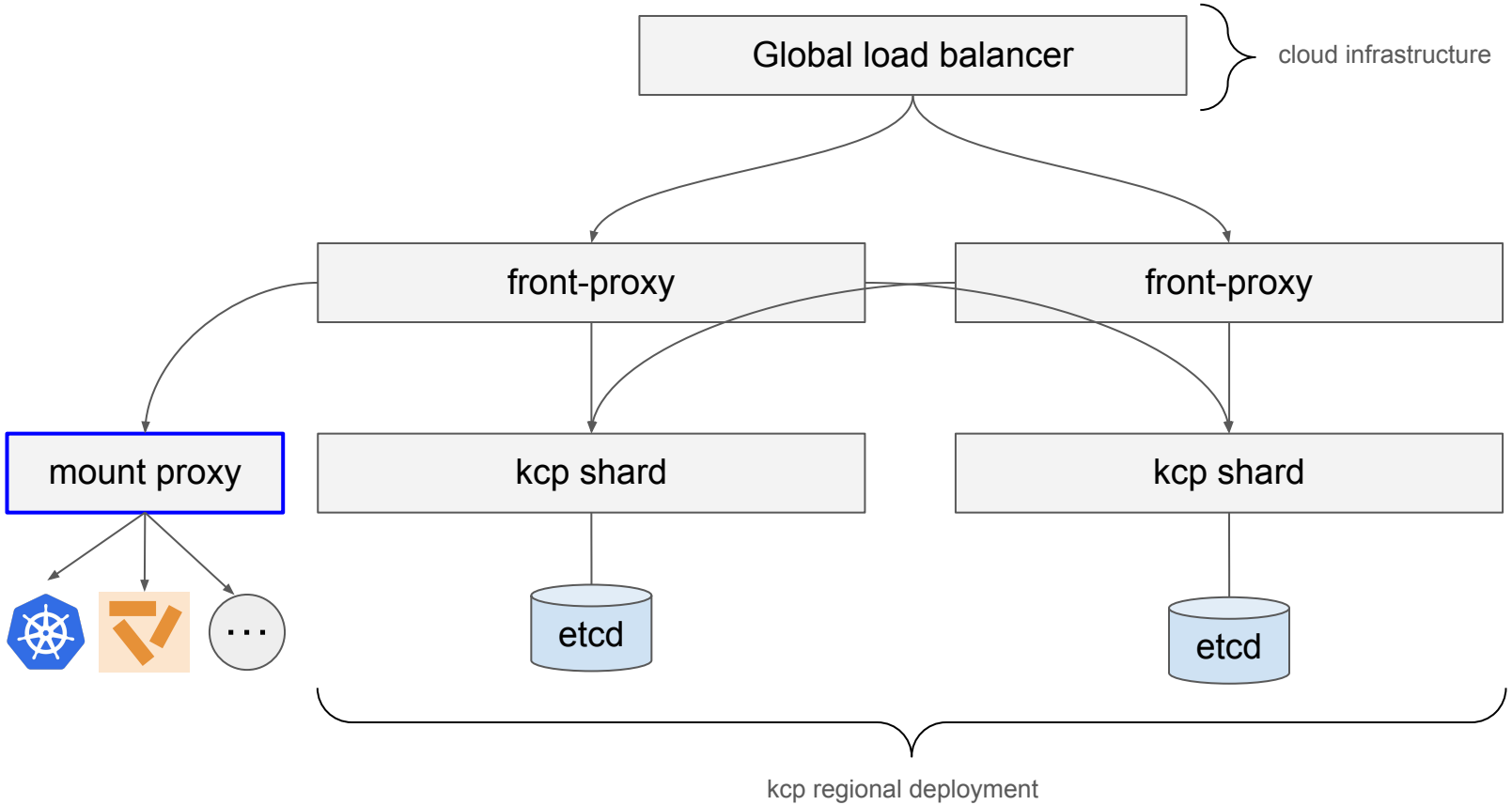

```
$ kubectl create vcluster b
$ kubectl ws b
$ kubectl apply -f deployment.yaml
$ kubectl ws ..
```

Mounts under the hood

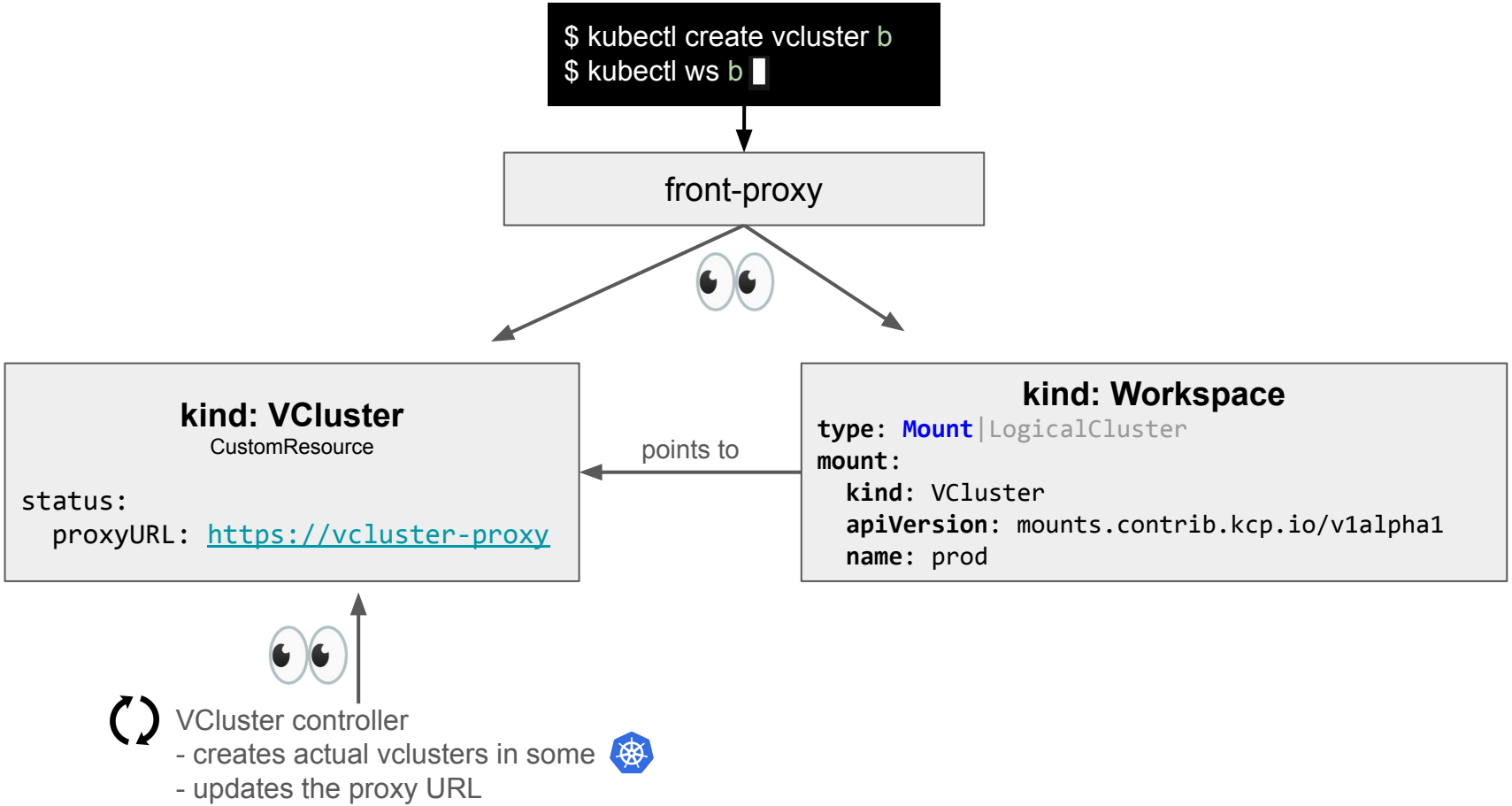
Mounts under the hood



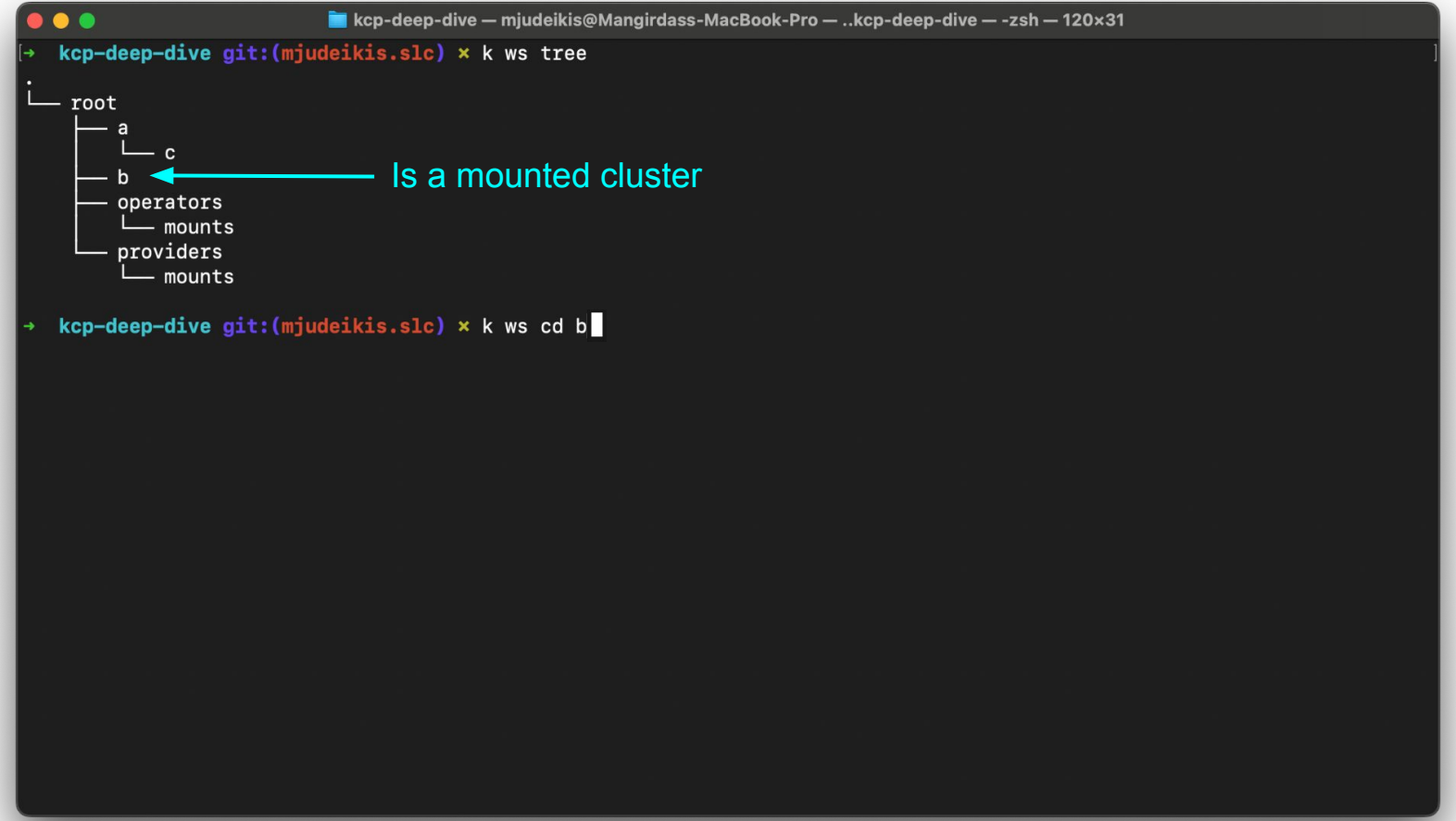
Mounts under the hood



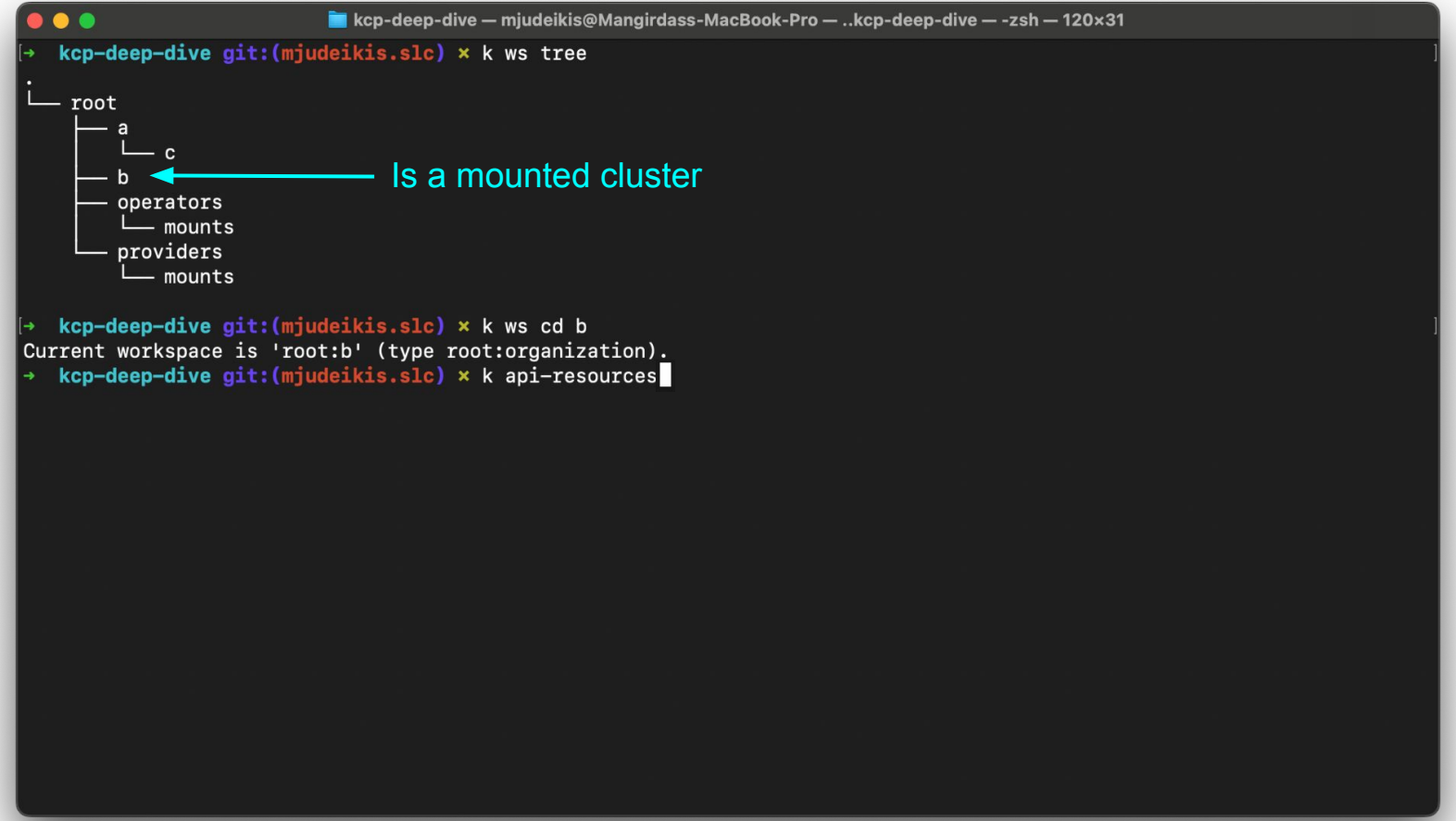
Mounts under the hood – an extension point



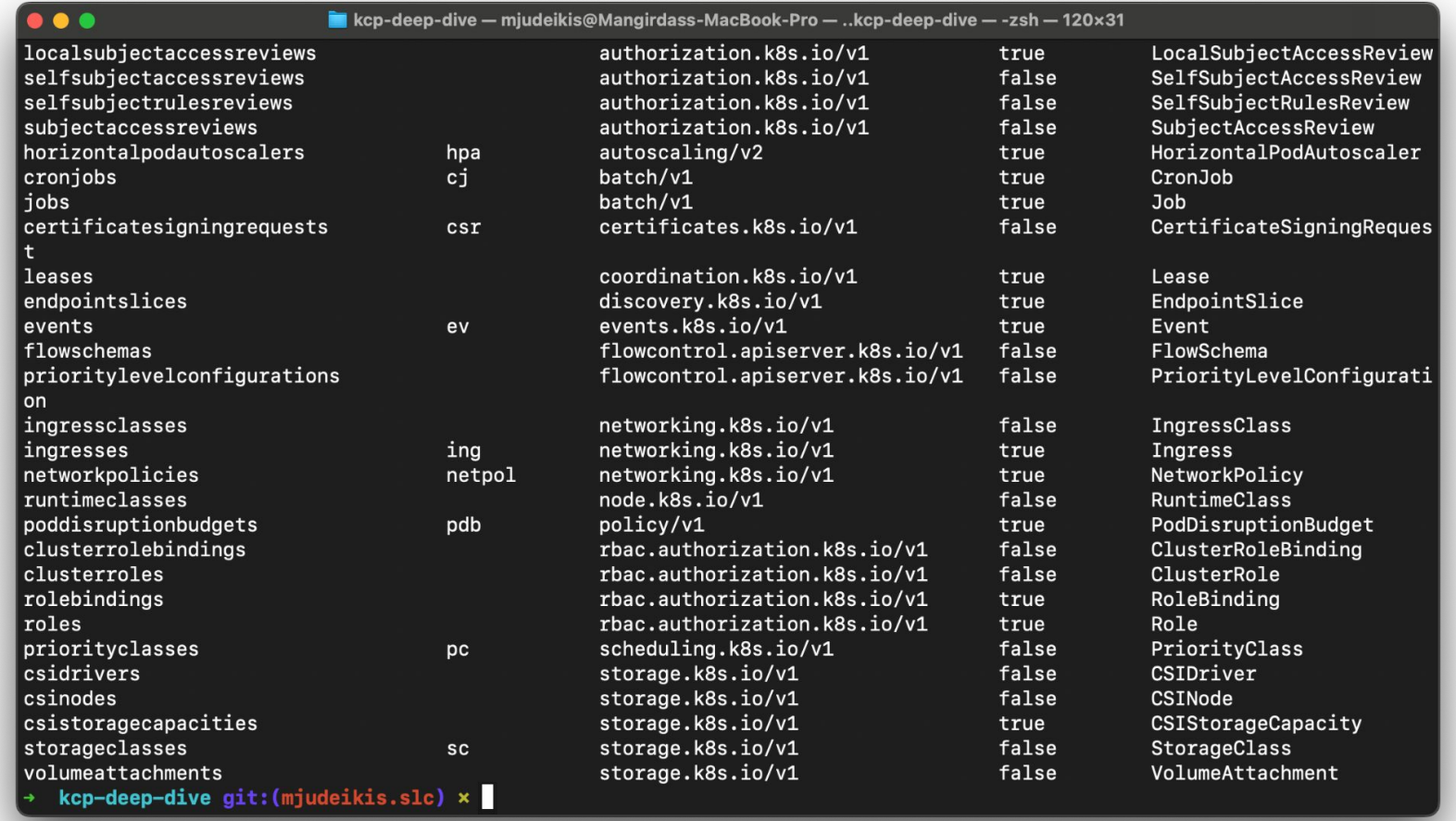
LogicalClusters



LogicalClusters



LogicalClusters



LogicalClusters

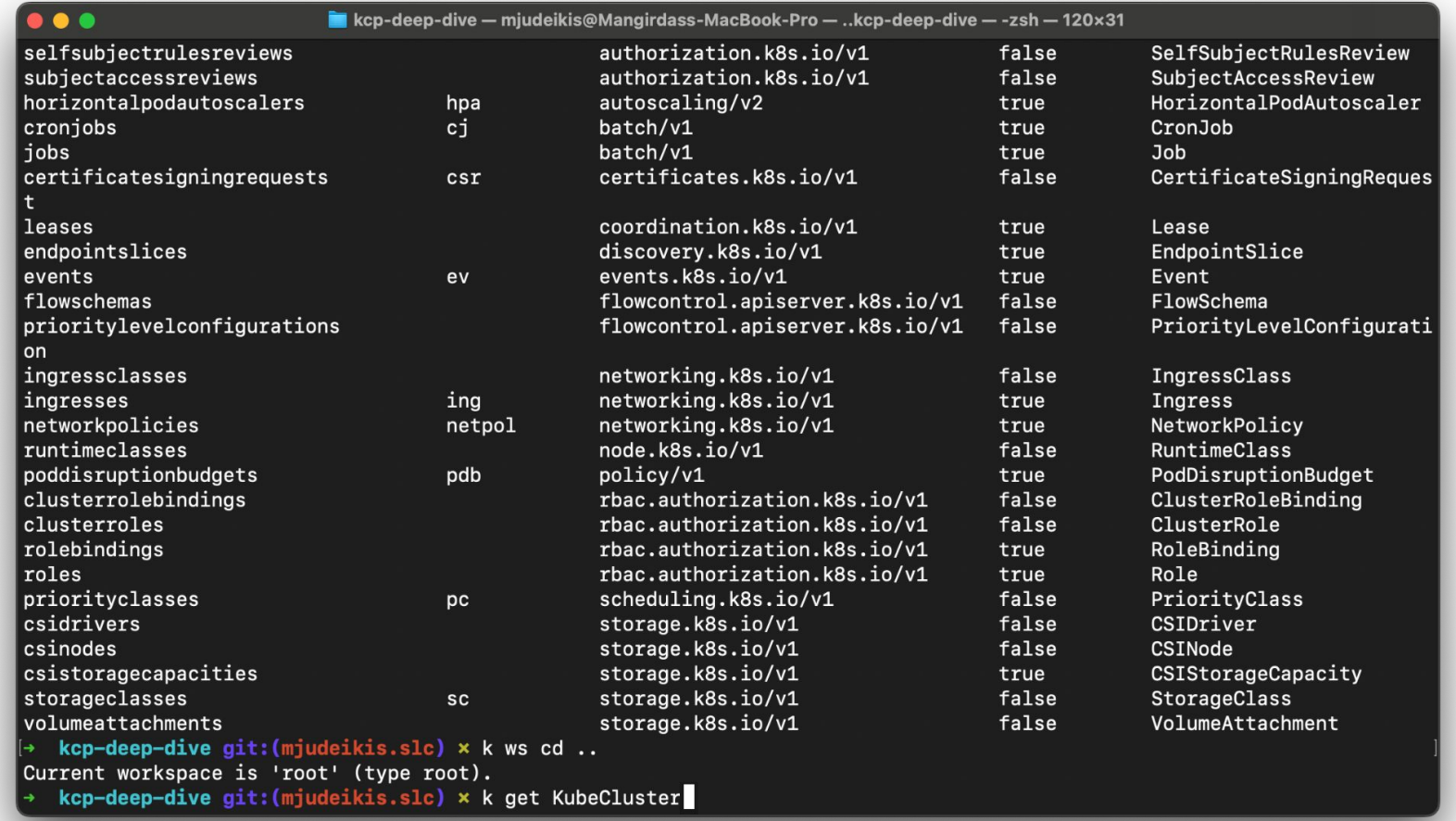
KubeCon CloudNativeCon North America 2024

kcp-deep-dive — mjudeikis@Mangirdass-MacBook-Pro — ..kcp-deep-dive — -zsh — 120×31

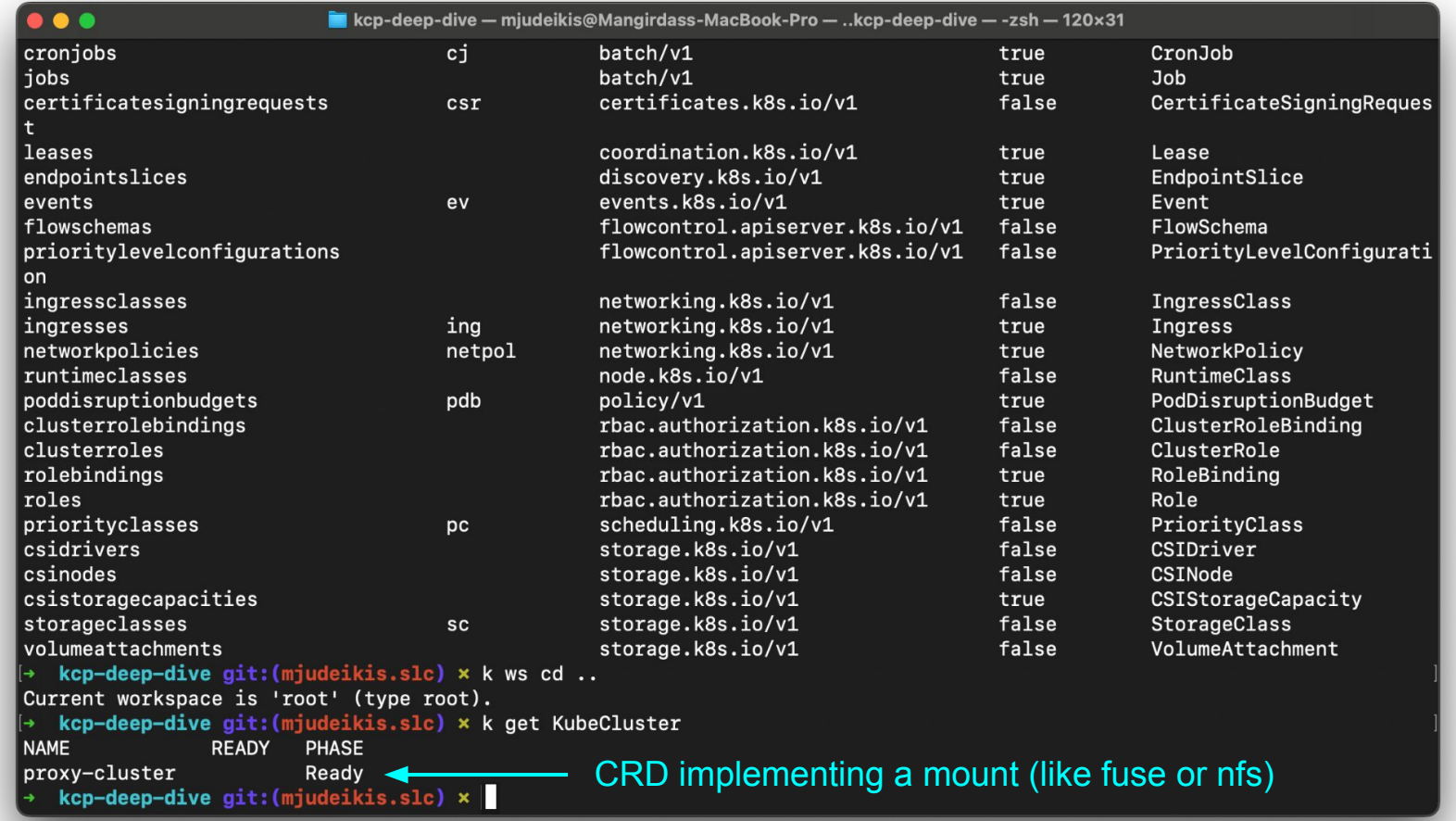
```
localsubjectaccessreviews                       authorization.k8s.io/v1              true         LocalSubjectAccessReview
selfsubjectaccessreviews                        authorization.k8s.io/v1              false        SelfSubjectAccessReview
selfsubjectrulesreviews                         authorization.k8s.io/v1              false        SelfSubjectRulesReview
subjectaccessreviews                            authorization.k8s.io/v1              false        SubjectAccessReview
horizontalpodautoscalers          hpa           autoscaling/v2                       true         HorizontalPodAutoscaler
cronjobs                          cj            batch/v1                             true         CronJob
jobs                                            batch/v1                             true         Job
certificatesigningrequests        csr           certificates.k8s.io/v1               false        CertificateSigningReques
t
leases                                          coordination.k8s.io/v1               true         Lease
endpointslices                                  discovery.k8s.io/v1                  true         EndpointSlice
events                            ev            events.k8s.io/v1                     true         Event
flowschemas                                     flowcontrol.apiserver.k8s.io/v1      false        FlowSchema
prioritylevelconfigurations                     flowcontrol.apiserver.k8s.io/v1      false        PriorityLevelConfigurati
on
ingressclasses                                  networking.k8s.io/v1                 false        IngressClass
ingresses                         ing           networking.k8s.io/v1                 true         Ingress
networkpolicies                   netpol        networking.k8s.io/v1                 true         NetworkPolicy
runtimeclasses                                  node.k8s.io/v1                       false        RuntimeClass
poddisruptionbudgets              pdb           policy/v1                            true         PodDisruptionBudget
clusterrolebindings                             rbac.authorization.k8s.io/v1         false        ClusterRoleBinding
clusterroles                                    rbac.authorization.k8s.io/v1         false        ClusterRole
rolebindings                                    rbac.authorization.k8s.io/v1         true         RoleBinding
roles                                           rbac.authorization.k8s.io/v1         true         Role
priorityclasses                   pc            scheduling.k8s.io/v1                 false        PriorityClass
csidrivers                                      storage.k8s.io/v1                    false        CSIDriver
csinodes                                        storage.k8s.io/v1                    false        CSINode
csistoragecapacities                            storage.k8s.io/v1                    true         CSIStorageCapacity
storageclasses                    sc            storage.k8s.io/v1                    false        StorageClass
volumeattachments                               storage.k8s.io/v1                    false        VolumeAttachment
→  kcp-deep-dive git:(mjudeikis.slc) ✗ k ws cd ..
```

M

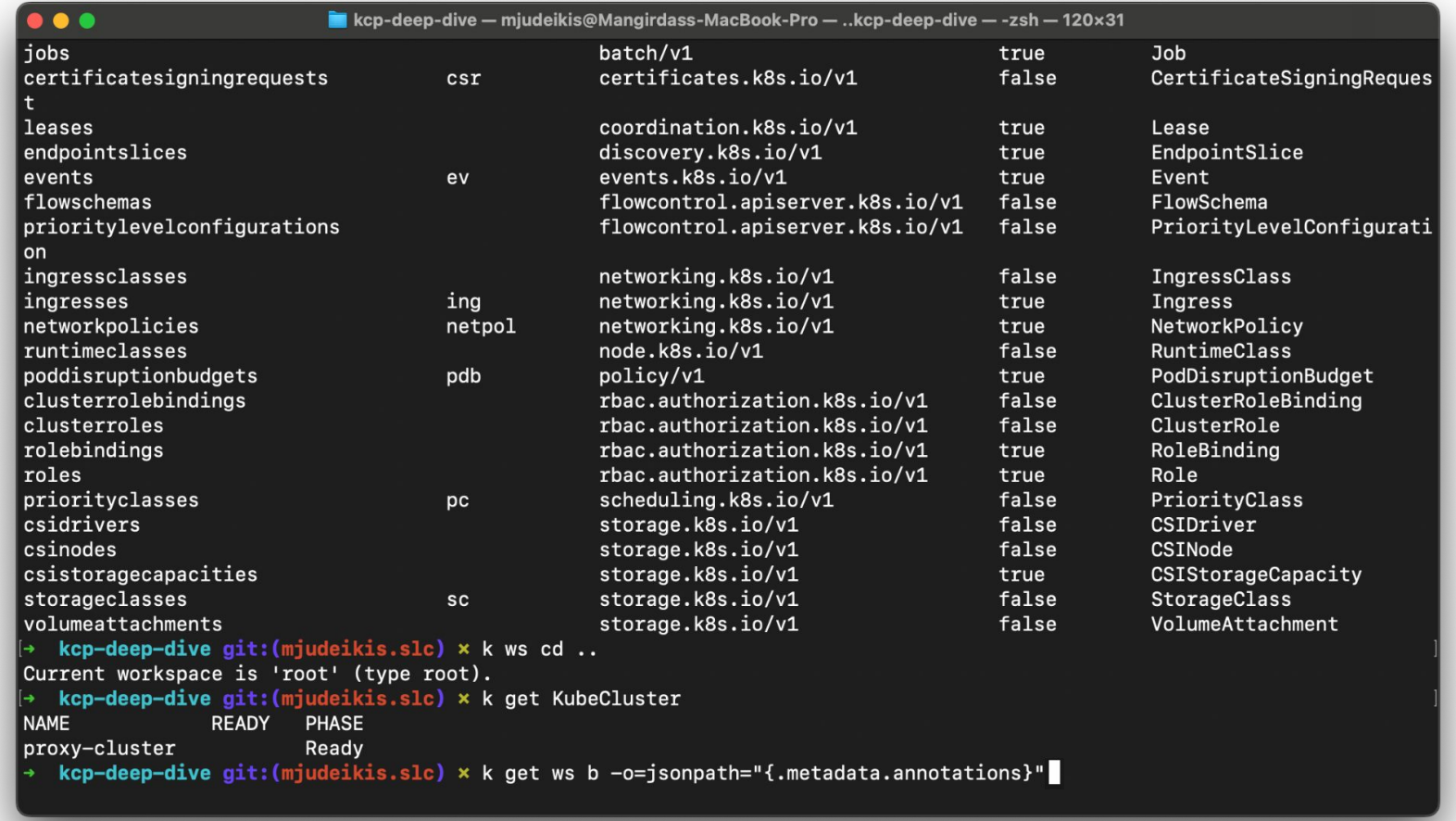
LogicalClusters



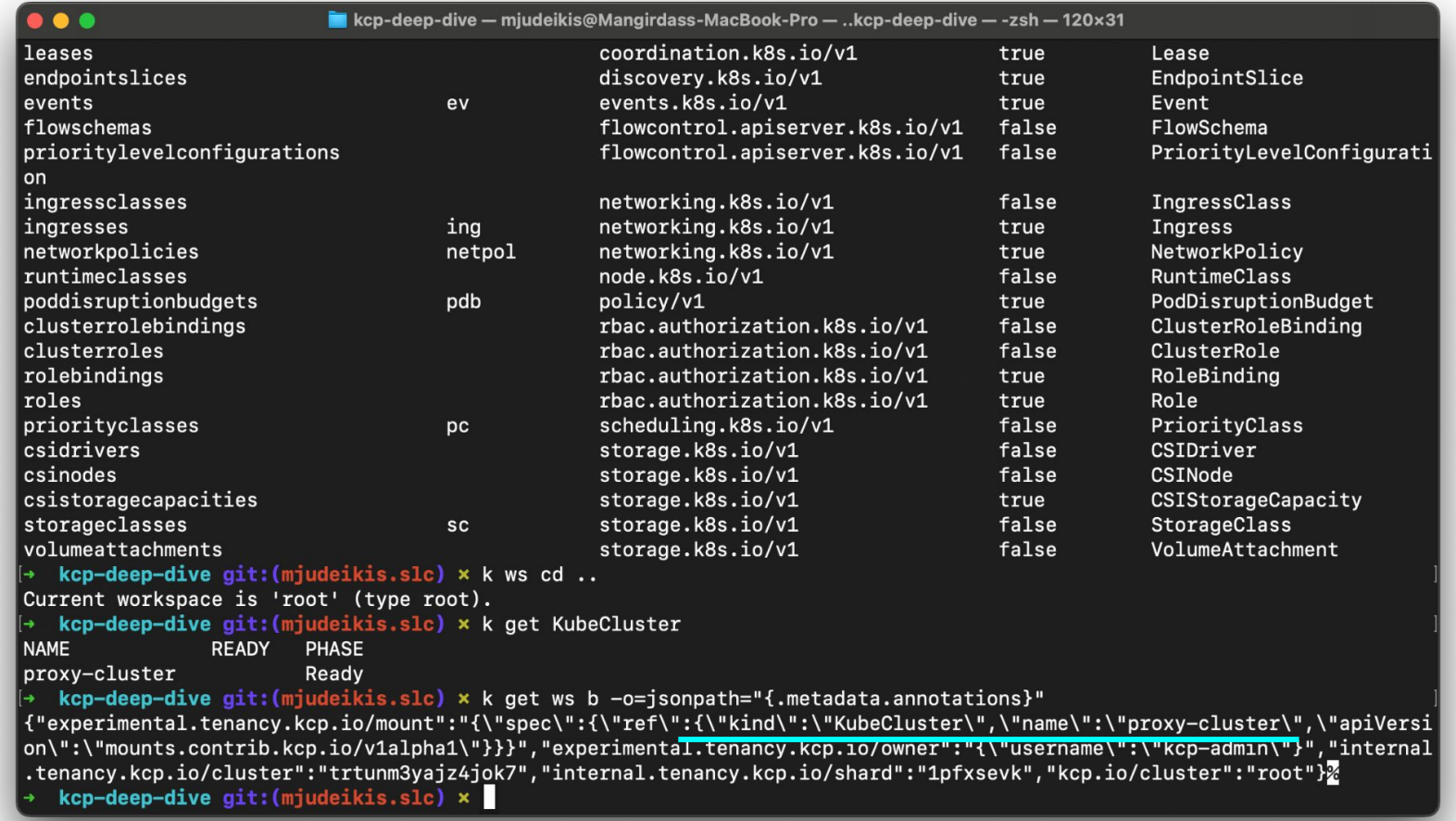
LogicalClusters



LogicalClusters

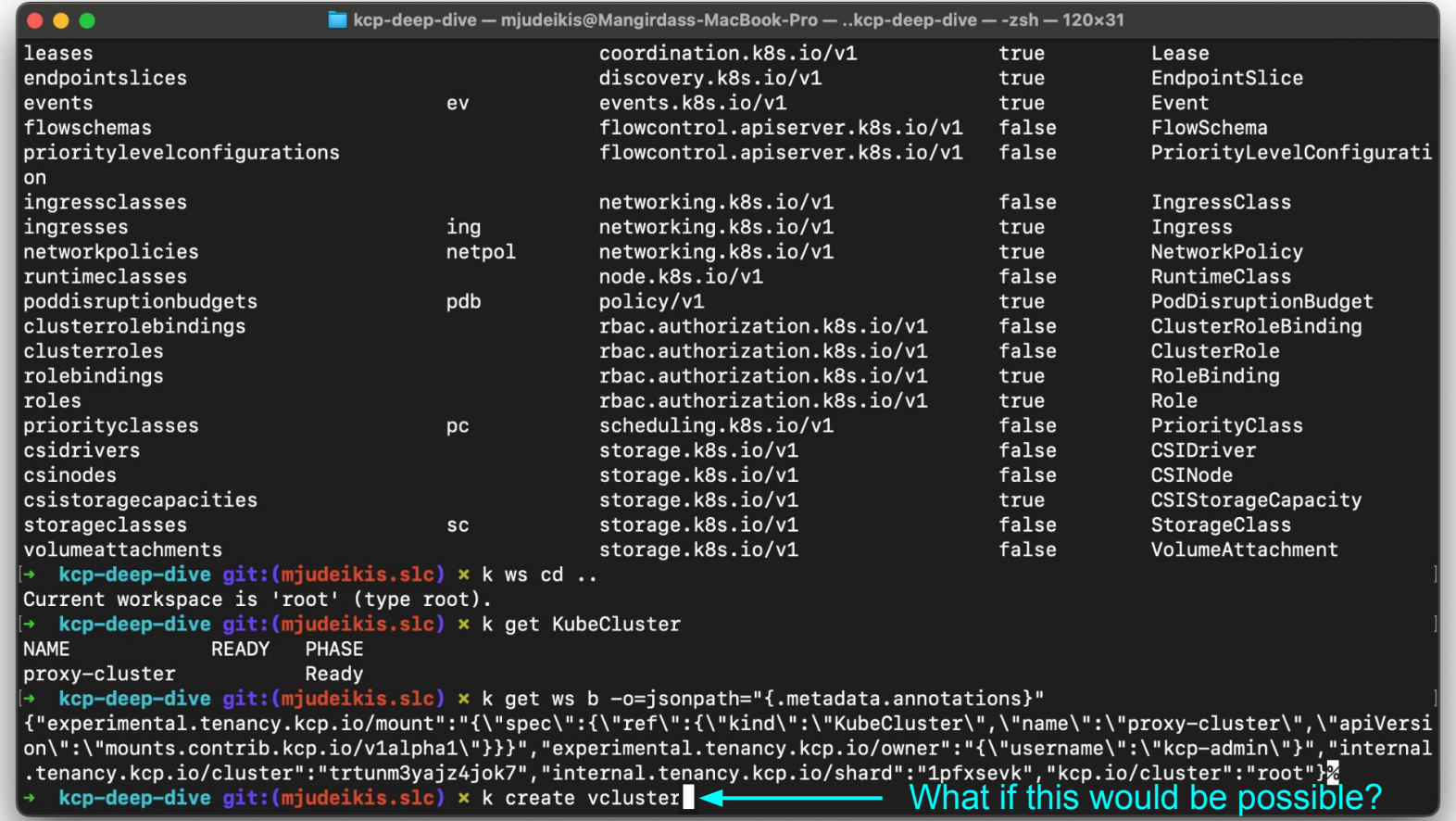


LogicalClusters



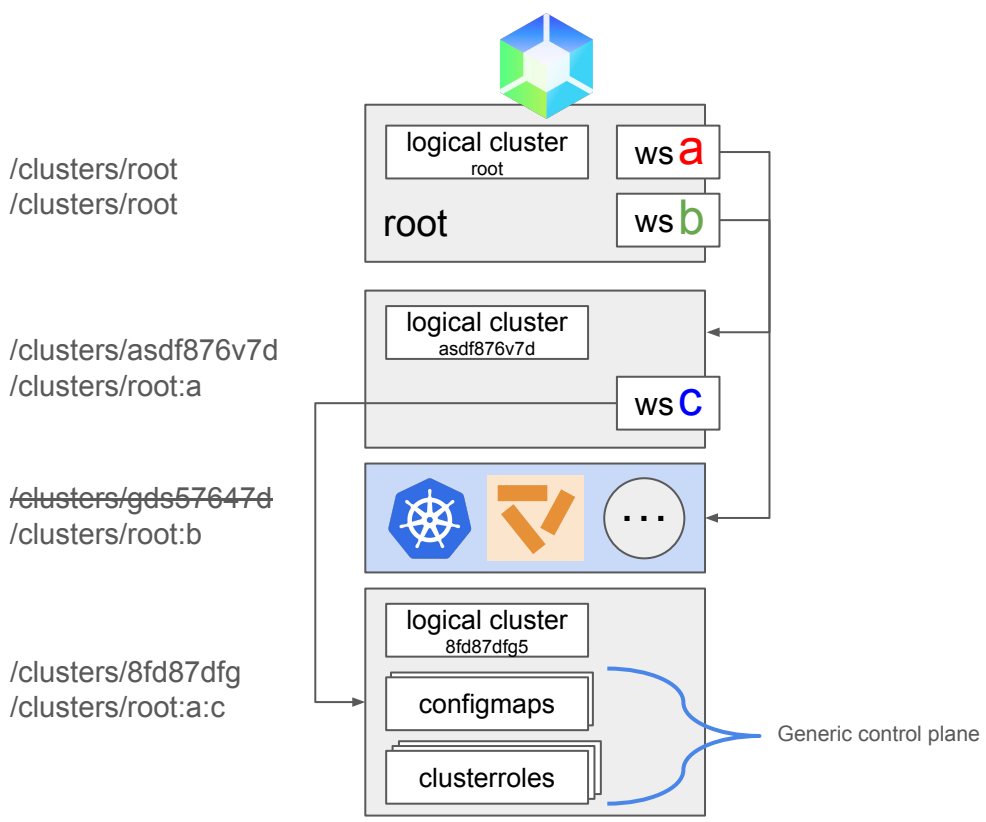
M

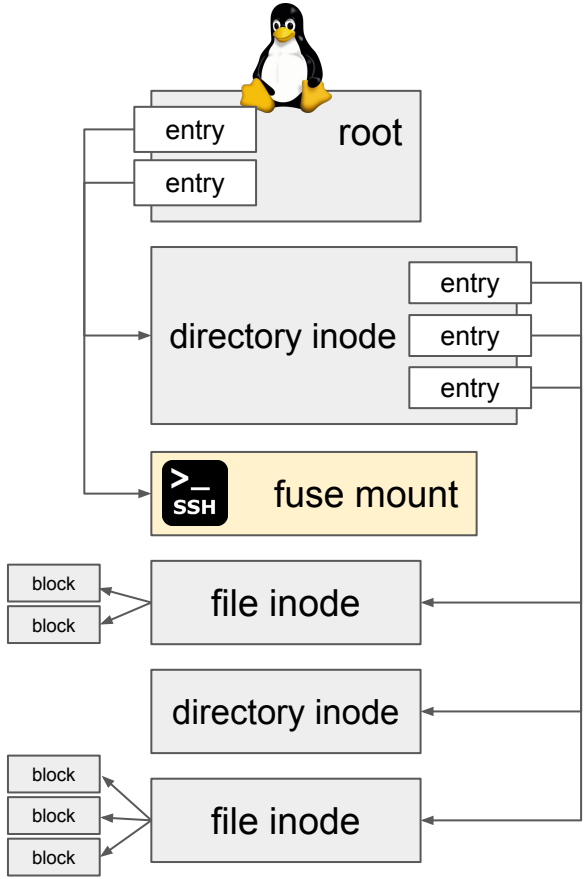
LogicalClusters



M

Recap – an extension point



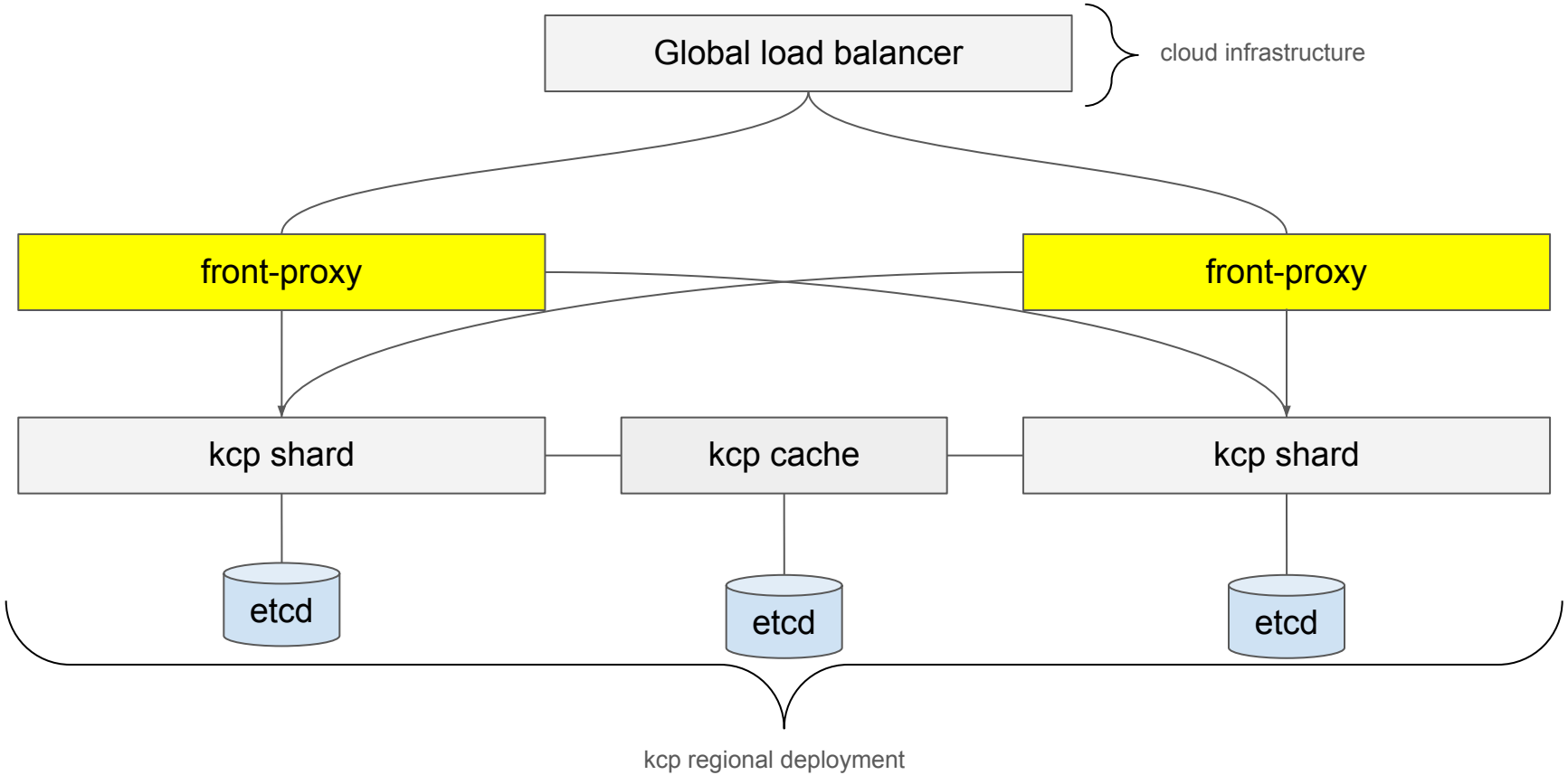


M

Deep dive into components

M

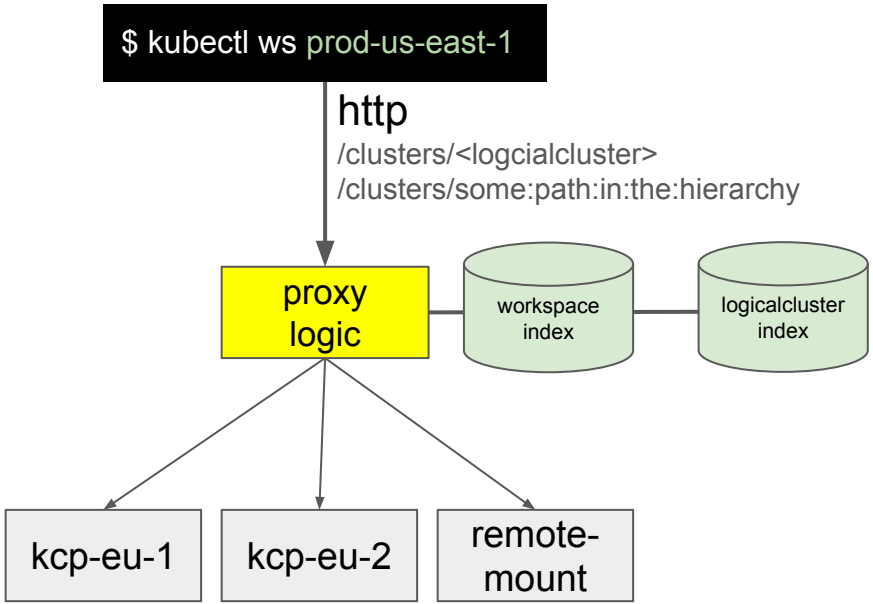
Deep dive into components: Front Proxy

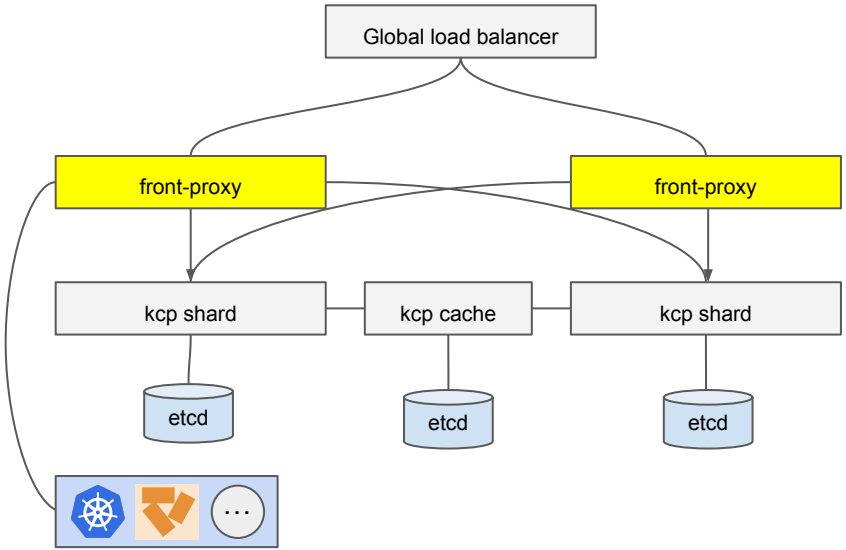


M

Deep dive into components: Front Proxy

- builds indexes:
  - logical clusters → shard | mount proxy
  - workspace paths → shard | mount proxy
- forwards requests to shards | mount proxies

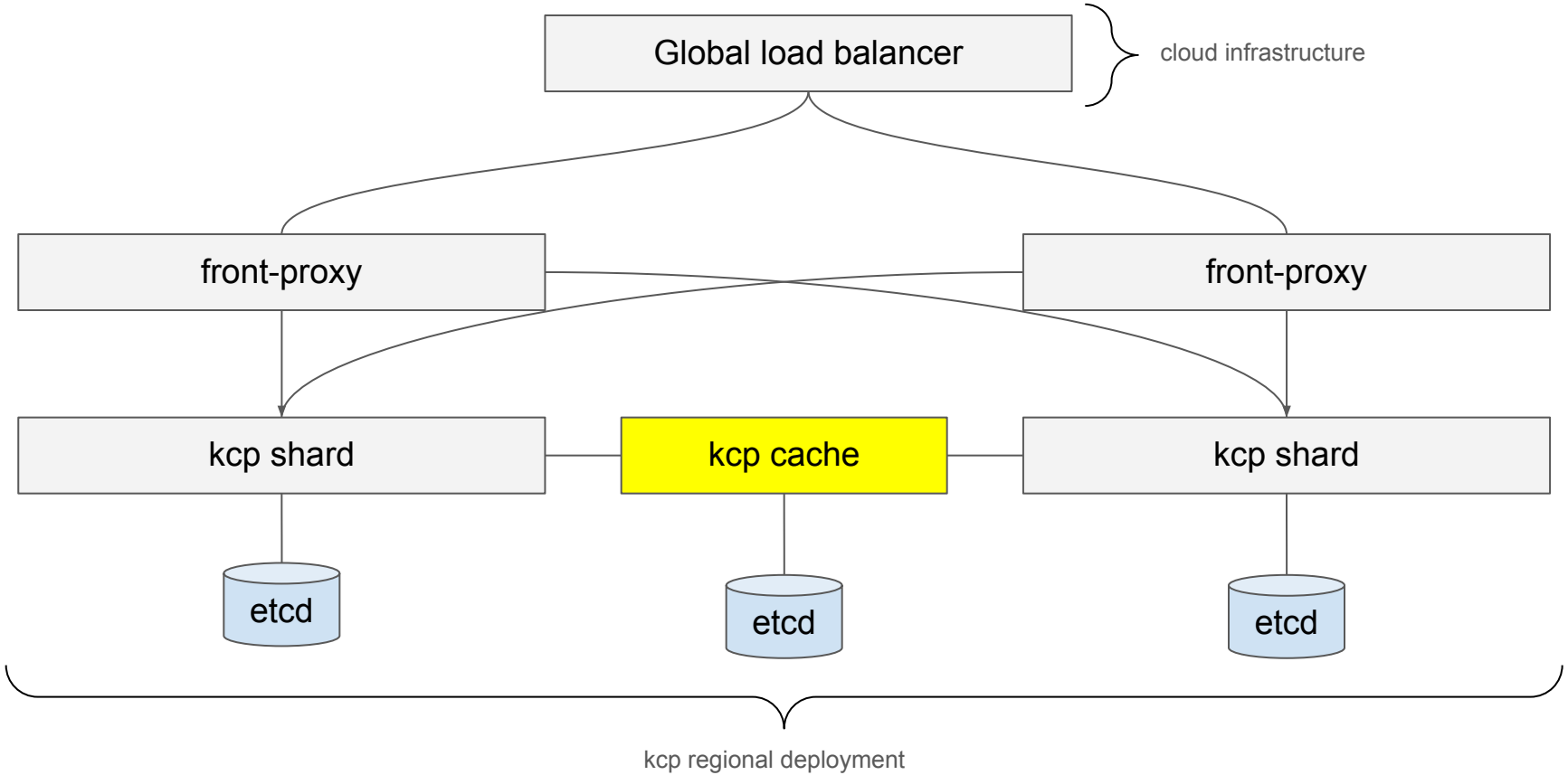




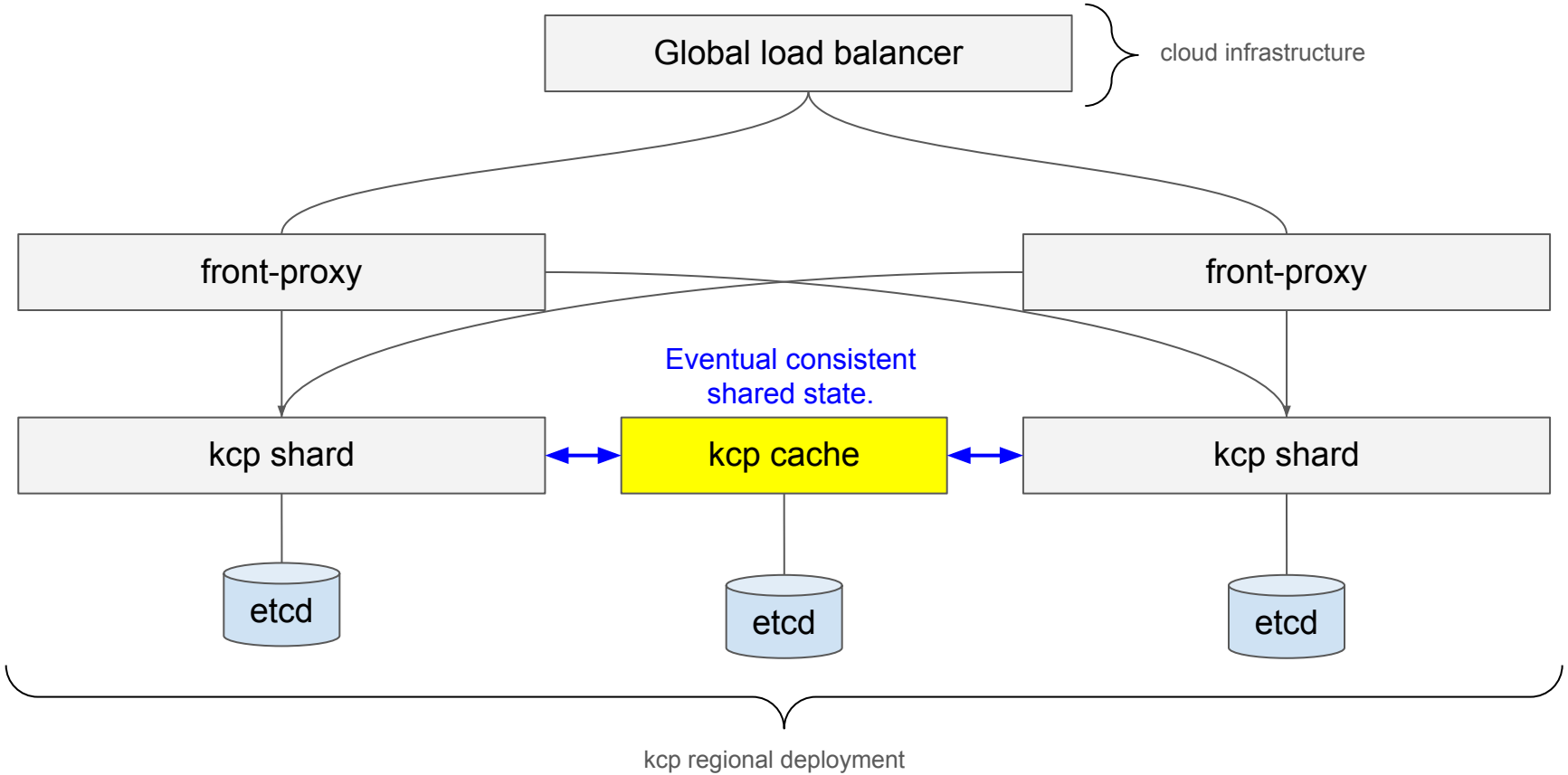
https://github.com/kcp-dev/kcp/blob/main/pkg/proxy/server.go#L50

Deep dive into components

Deep dive into components



Deep dive into components

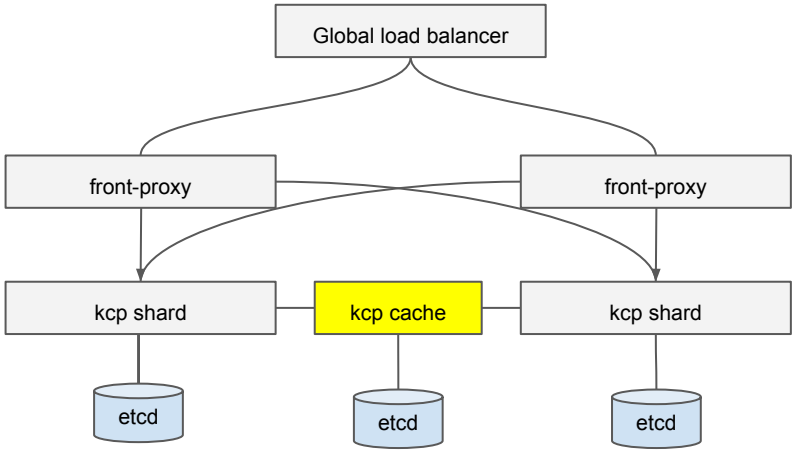


Deep dive into components: Cache Server

- some APIs (sparsely!) publish objects to the cache, visible by all shards
- e.g. APIExports, APIResourceSchemas, globally relevant RBAC
- backed by etcd for convenience to have informers (could be another store, e.g. global db, read-heavy)

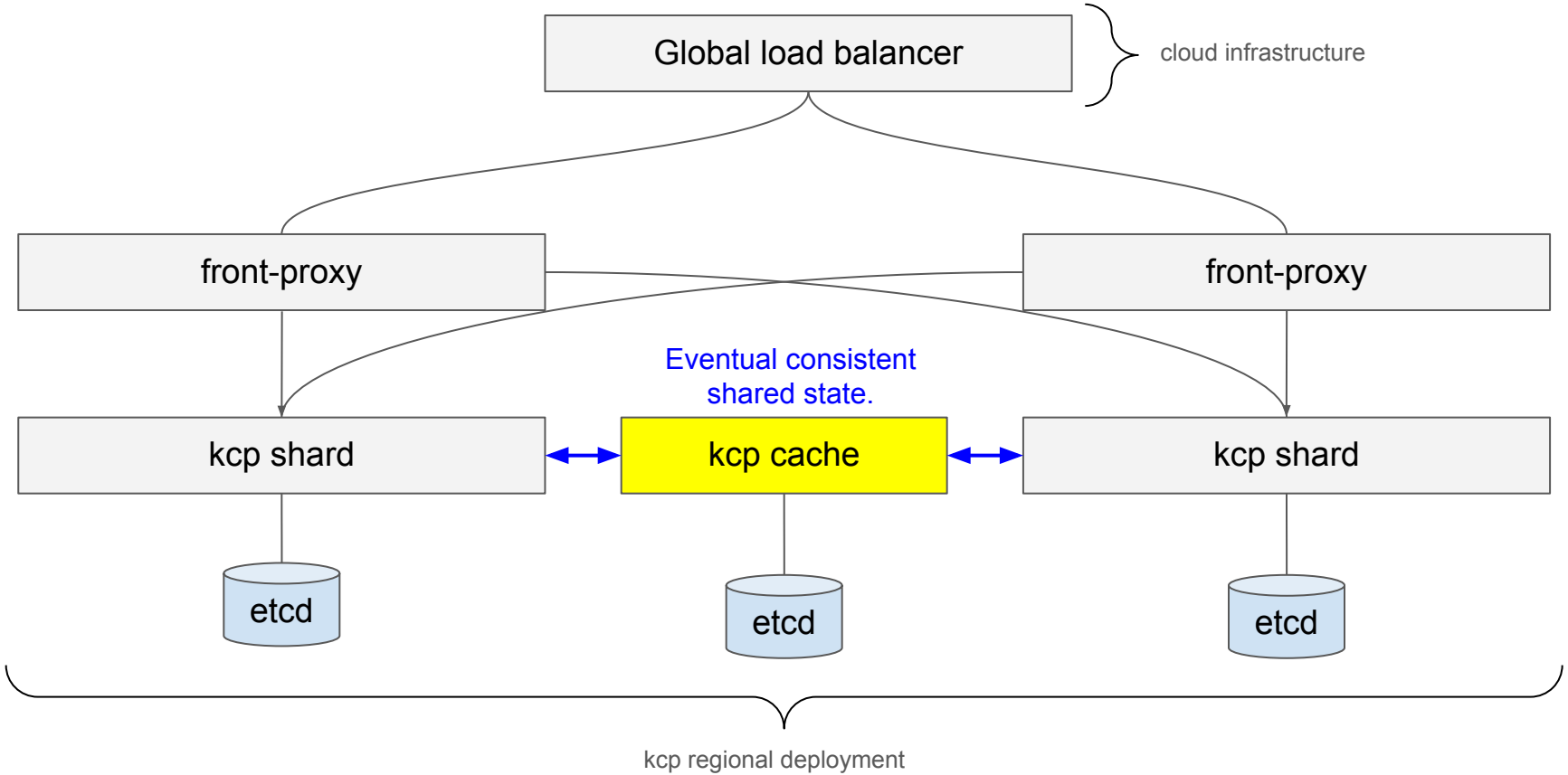
```
for _, gr := range []struct{ group, resource string }{
    {"apis.kcp.io", "apiresourceschemas"},
    {"apis.kcp.io", "apiconversions"},
    {"apis.kcp.io", "apiexports"},
    {"core.kcp.io", "logicalclusters"},
    {"core.kcp.io", "shards"},
    {"tenancy.kcp.io", "workspacetypes"},
    {"rbac.authorization.k8s.io", "roles"},
    {"rbac.authorization.k8s.io", "clusterroles"},
    {"rbac.authorization.k8s.io", "rolebindings"},
    {"rbac.authorization.k8s.io", "clusterrolebindings"},
    {"admissionregistration.k8s.io", "mutatingwebhookconfigurations"},
    {"admissionregistration.k8s.io", "validatingwebhookconfigurations"},
    {"admissionregistration.k8s.io", "validatingadmissionpolicies"},
    {"admissionregistration.k8s.io", "validatingadmissionpolicybindings"},
```

sparse, only when needed

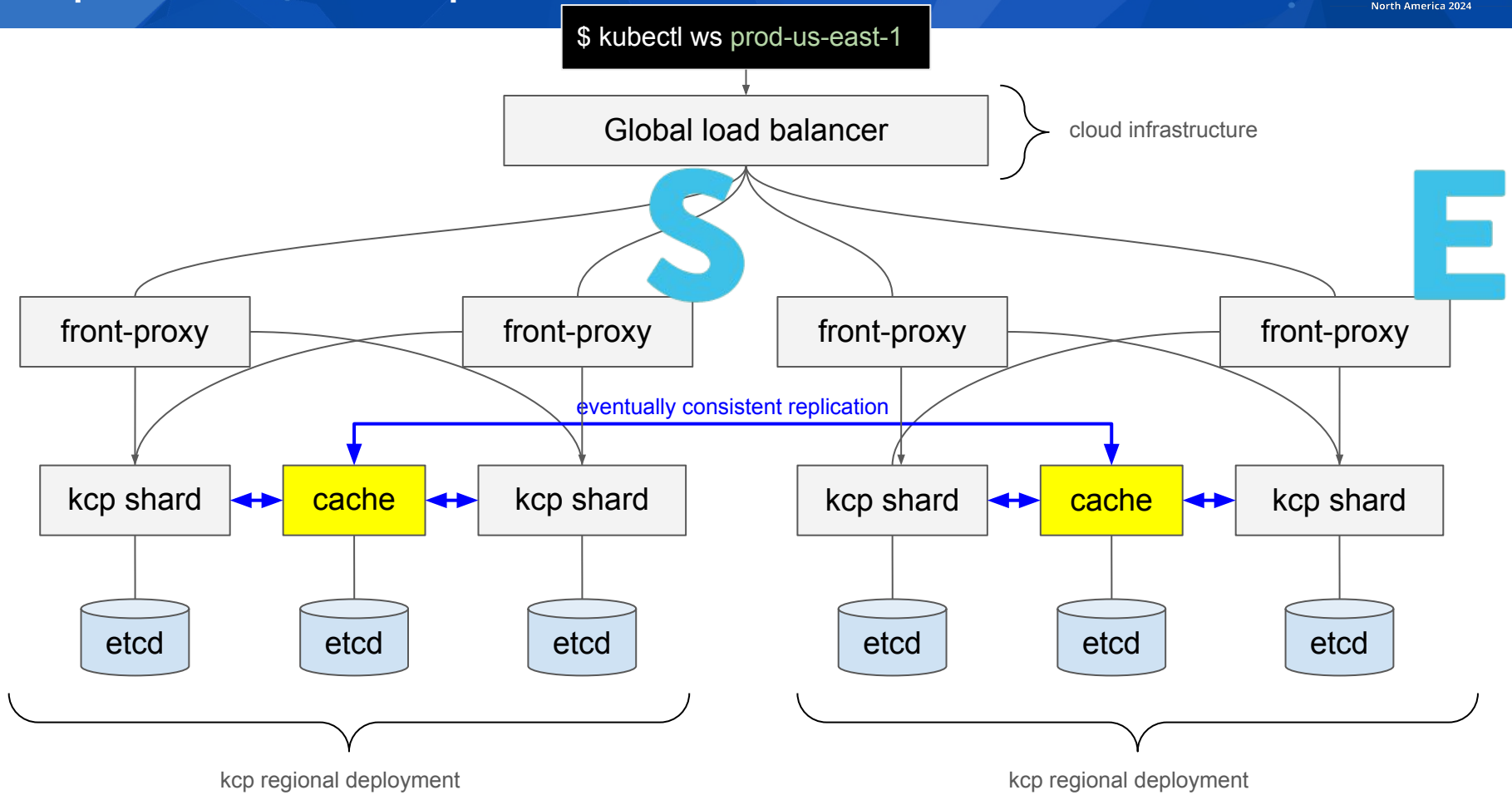


https://github.com/kcp-dev/kcp/blob/main/pkg/cache/server/bootstrap/bootstrap.go

Deep dive into components: today



Deep dive into components: outlook



Programming Model – controller-runtime

Programming against one shard with many logical clusters.

M

Programming Model – controller-runtime

KubeCon CloudNativeCon North America 2024

```
import (
      # with replace in go.mod to github.com/kcp-dev/controller-runtime

      “sigs.k8s.io/controller-runtime/pkg/kcp”
      “sigs.k8s.io/controller-runtiem/pkg/kontext”
)

mgr, _ := kcp.NewClusterAwareManager(kcpRestConfig, options)    # path / of a shard

func (r *Reconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
  log := log.FromContext(ctx).WithValues("cluster", req.ClusterName)
  ctx := kontext.WithCluster(req.ClusterName)

  var cm corev1.ConfigMap
  if err := r.Client.Get(ctx, req.NamespacedName, &cm); err != nil {
    log.Error(err, "unable to get configmap")
    return ctrl.Result{}, nil
  }
}
```

Logical cluster name

https://github.com/kcp-dev/controller-runtime/tree/kcp-0.19/examples/kcp

Programming Model – controller-runtime

Programming with objects in the cache,

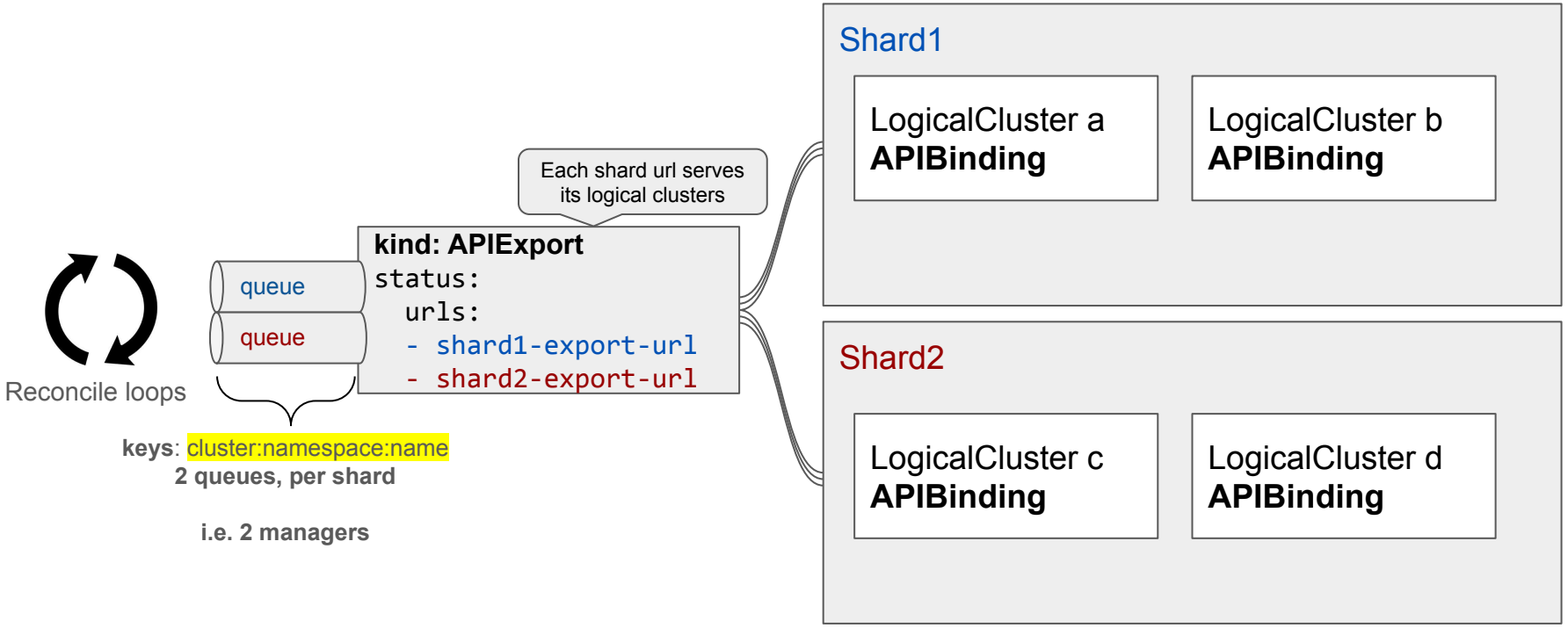
aka multi-shard awareness, potentially global and multi-region.

Programming Model – controller-runtime


```
import (
      # with replace in go.mod to github.com/kcp-dev/controller-runtime

      “sigs.k8s.io/controller-runtime/pkg/kcp”
      “sigs.k8s.io/controller-runtiem/pkg/kontext”
)

mgr, _ := kcp.NewClusterAwareManager(kcpRestConfig, options)    # path / of a shard
cache, := kcp.NewClusterAwareCache(cacheRestConfig, options)    # path /services/cache

func (r *Reconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
  ...

  cachedObj := &Widget{}
  if err := mgr.Get(kontext.WithCluster(ctx, cluster), name, cachedObj); err != nil && !kerrors.IsNotFound(err) {
    log.Error(err, "failed to get widget from local manager")
    return ctrl.Result{}, err
  } else if err := cache.Get(kontext.WithCluster(ctx, cluster), name, cachedObj); err != nil {
    log.Error(err, "failed to get widget from cache")
    return ctrl.Result{}, err
  }

  ...
}
```

https://github.com/kcp-dev/controller-runtime/tree/kcp-0.19/examples/kcp

Programming Model – multi-shard



Efforts to enable controller-runtime natively to write multi-cluster controllers (not only for kcp):
https://github.com/kubernetes-sigs/controller-runtime/pull/2726

Demo! - KCP

https://github.com/kcp-dev/kcp/pull/3190

M

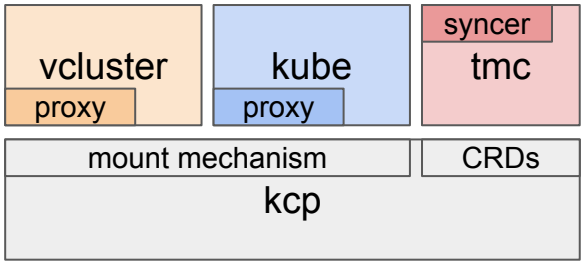
kcp as a framework

- kcp is **NOT a product**, but an **API & control plane framework**.
- like **CRDs in Kube are not opinionated** for what they are used.

⇒ kcp (core) will NOT provide compute

But you (as product developer) can add compute in different ways:

1. via **mount extension point**. Examples: **vcluster**, **kube**, **Gardener**, NaaS, you name it
2. via a **federation-like on-top API**. Example: (abandoned) transparent **multi cluster**, this is what Red Hat was working on.



Thank you

More talks in the kcp realm @ KubeCon

KubeCon CloudNativeCon North America 2024

Thursday, November 14

11:00am MST

- Kubernetes Workspaces: Enhancing Multi-Tenancy with Intelligent Apiserver Proxying - James Munnelly & Andrea Tosatto, Apple (Description: kcp)

Friday, November 15

4:55pm MST

- Best of Both Worlds: Integrating Slurm with Kubernetes in a Kubernetes Native Way - Eduardo Arango Gutierrez, NVIDIA & Angel Beltre, Sandia National Laboratories (Description: kcp)

Talk to us - look for logo!

Find us at #kcp-dev kubernetes slack

github.com/kcp-dev/kcp

@kcp @the_sttts @mangirdas

Find us at one of these booths!

or **project kiosks** after this talk!

Feedback!