KCP:
Cloud Native API Control Planes

Marvin Beckers, Kubermatic

Marvin Beckers

Team Lead @ Kubermatic
& KCP Maintainer

`github.com/embik`

`linkedin.com/in/marvinbeckers`

`hachyderm.io/@embik`

Agenda

1. Kubernetes as an API Layer
2. Lightweight Clusters to the Rescue
3. What is KCP?
   a. Workspaces
4. The API Marketplace
   a. Create APIs with APIExports
   b. Enable APIs with APIBindings
5. Wrapping Up

Kubernetes as an API Layer

```
/apis/<group>/<version>/[namespaces/<namespace>/]<resourcetype>[/<name>]
```

APIs in Kubernetes are **grouped**.

Each API group is also **versioned**.

Resources are optionally **namespaced**.

Resources have a specific **resource type** that defines their schema.

Resources are uniquely **named**.

The Kubernetes API is pretty awesome!

(that’s it. That’s the ~~tweet~~ ~~post~~ slide)

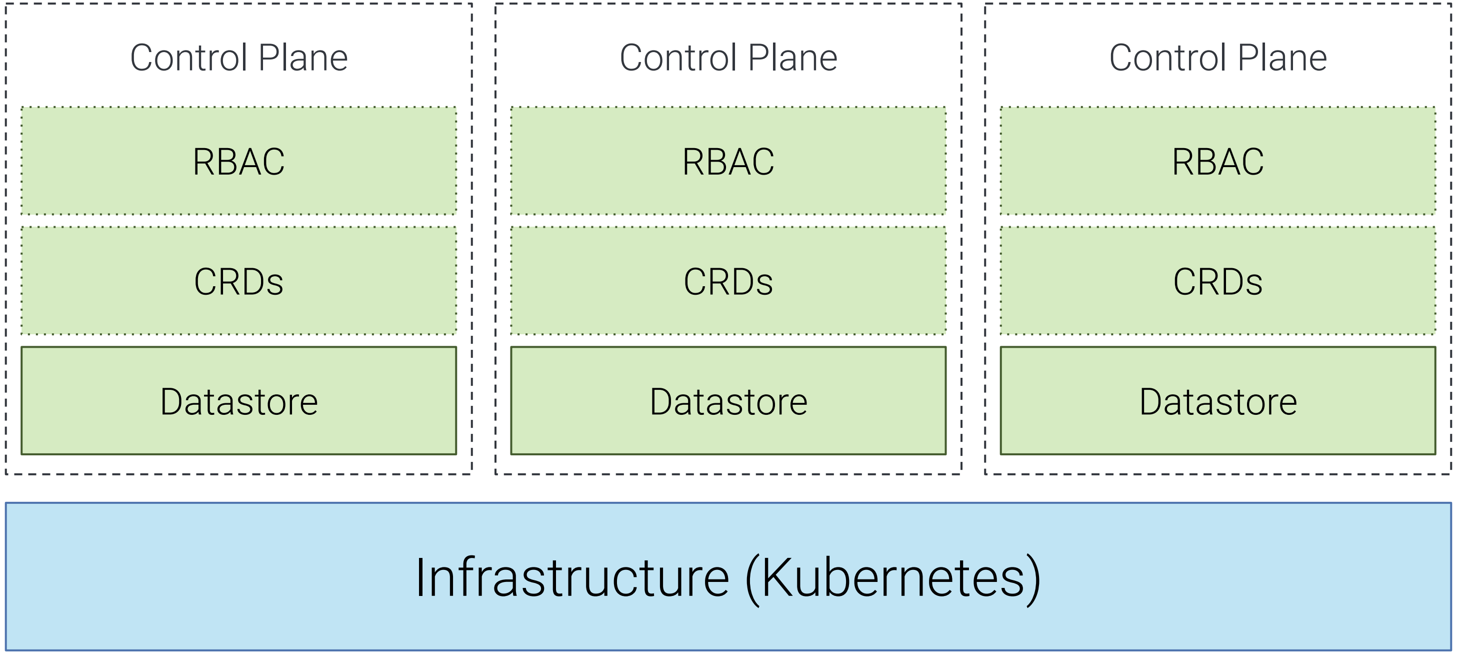
But …

- CRDs are cluster-scoped, so everyone shares them.
- Kubernetes clusters are local, not meant to scale across regions.
- You don’t need the workload orchestration part for APIs.

Lightweight Clusters

to the rescue?

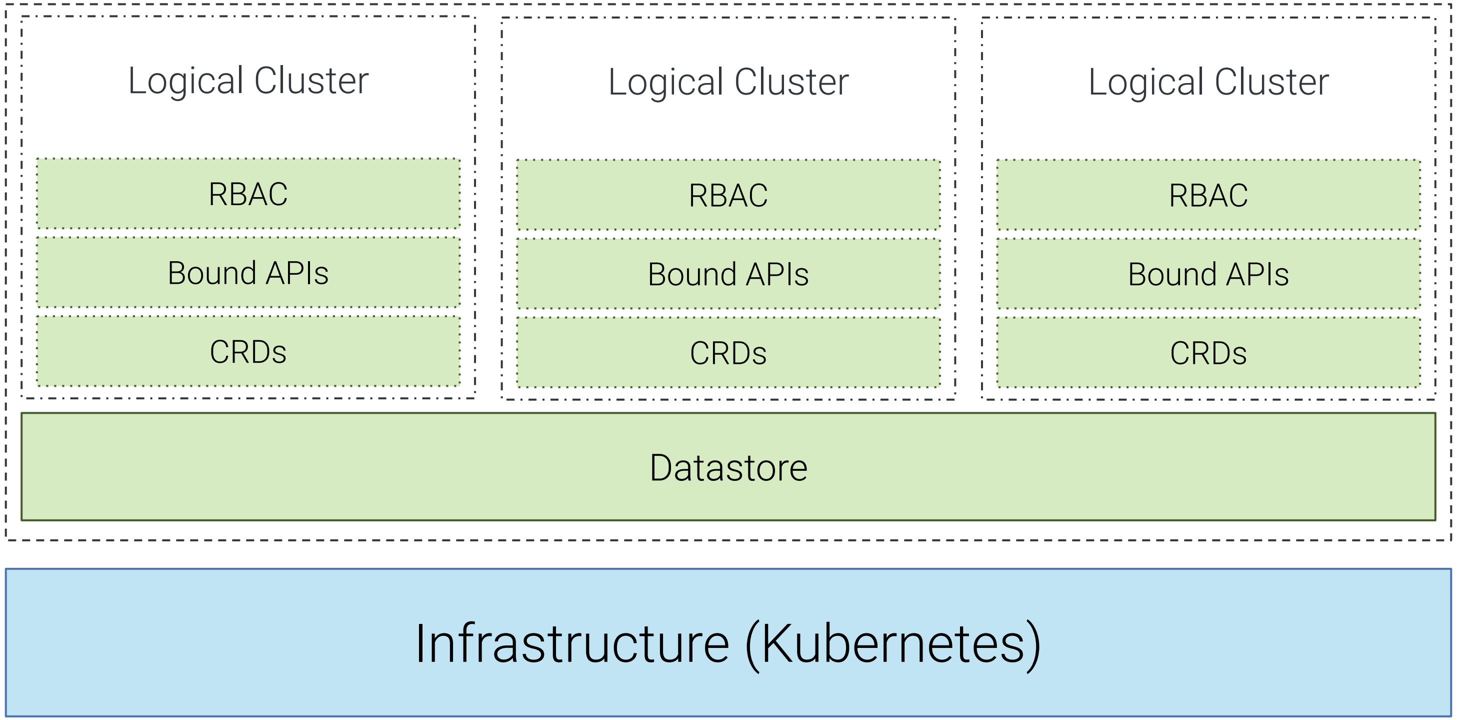
Hosted Control Planes



What if …

… we “virtualized” API servers?

“Logical” Clusters



What is KCP?

A horizontally scalable control plane for Kubernetes-style APIs.

CLOUD NATIVE
COMPUTING FOUNDATION

Sandbox project

(since end of 2023)

Workspace

A multi-tenancy **unit of isolation** in kcp.

Each workspaces has its own available **API resource types**.

API **objects** are not shared across workspaces.

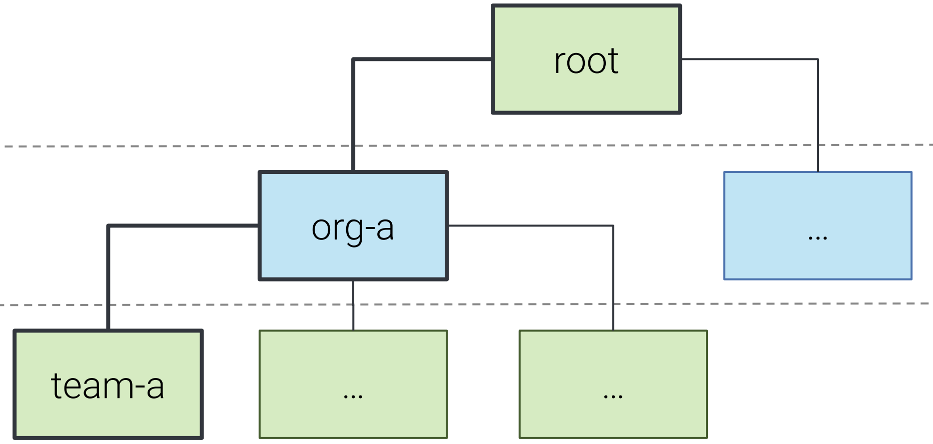
Delegation of **administrative permissions** to workspace owners.

Workspaces are **cheap**.

```
https://kcp:6443/clusters/root

https://kcp:6443/clusters/root:org-a

https://kcp:6443/clusters/root:org-a:team-a
```



Workspaces are organized in a tree (or multiple).

```
$ kubectl ws .
Current workspace is "root".

$ kubectl get ws
NAME    TYPE            REGION   PHASE   URL           AGE
org-a   organization             Ready   https://…     69d
org-b   organization             Ready   https://…     65d

$ kubectl ws org-a
Current workspace is "root:org-a" (type root:organization).

$ kubectl get ws
NAME     TYPE   REGION   PHASE   URL          AGE
team-a   team            Ready   https://…    3m23s
team-b   team            Ready   https://…    3m18s


$ kubectl ws team-a
Current workspace is "root:org-a:team-a" (type root:team).
```

And you can navigate them!

The API Marketplace

Service teams should provide services, not fiddle with their own API server implementation.

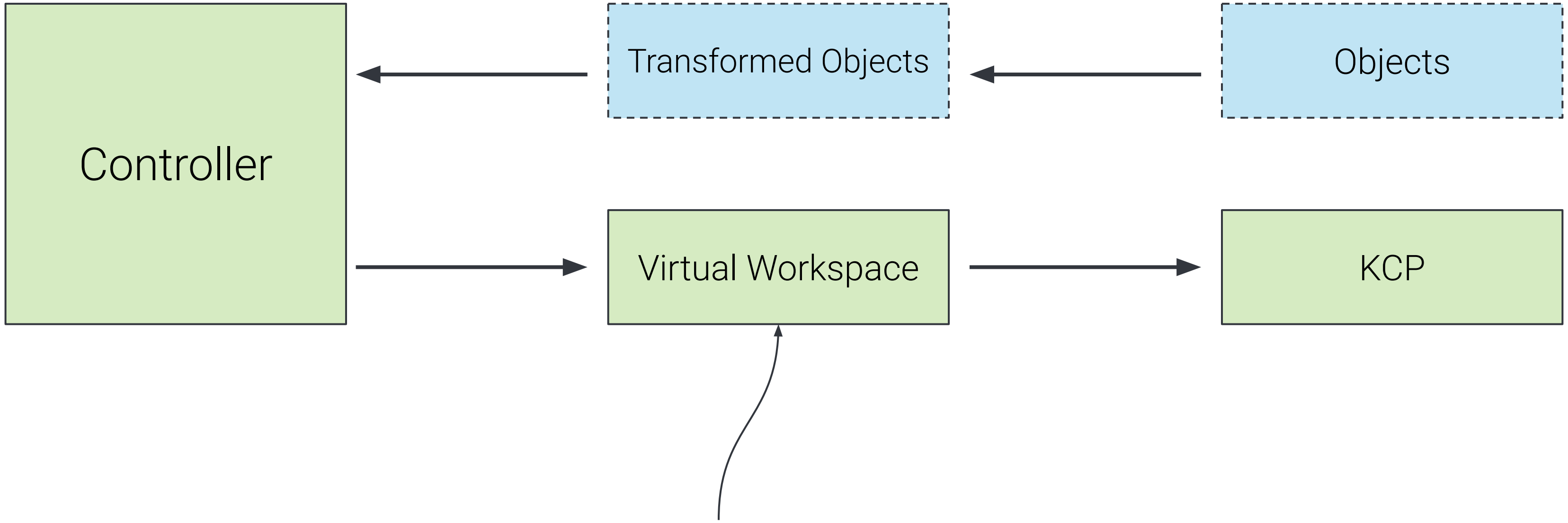
Create APIs with APIExports

APIExport

```
apiVersion: apis.kcp.io/v1alpha1
kind: APIExport
metadata:
  name: demo.embik.me
spec:
  latestResourceSchemas:
    - v1.certificates.demo.embik.me
    - v1.pizzas.demo.embik.me
```

Resource schemas define resources, just like CRDs.

Virtual Workspaces for Controllers



Proxy that provides a computed view

How to Build a KCP-aware Controller

1 Use kcp-aware client and cache

```
MapperProvider: kcp.NewClusterAwareMapperProvider,
NewClient:      kcp.NewClusterAwareClient,
NewCache:       kcp.NewClusterAwareCache,
NewAPIReader:   kcp.NewClusterAwareAPIReader,
```

2 Reconcile in Virtual Workspace via **Cluster**

```
sigs.k8s.io/controller-runtime/pkg/cluster.Cluster
```

3 Reconcile with logical cluster in context

```
ctx = kontext.WithCluster(ctx, logicalcluster.Name(request.ClusterName))
```

github.com/kcp-dev/controller-runtime

Enable APIs with APIBindings

Available APIs in a Workspace

```
https://kcp:6443/clusters/root:org-a/api
```

```
$ kubectl 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
[...]
workspaces           ws           tenancy.kcp.io/v1alpha1    false        Workspace
workspacetypes                    tenancy.kcp.io/v1alpha1    false        WorkspaceType
[...]
```

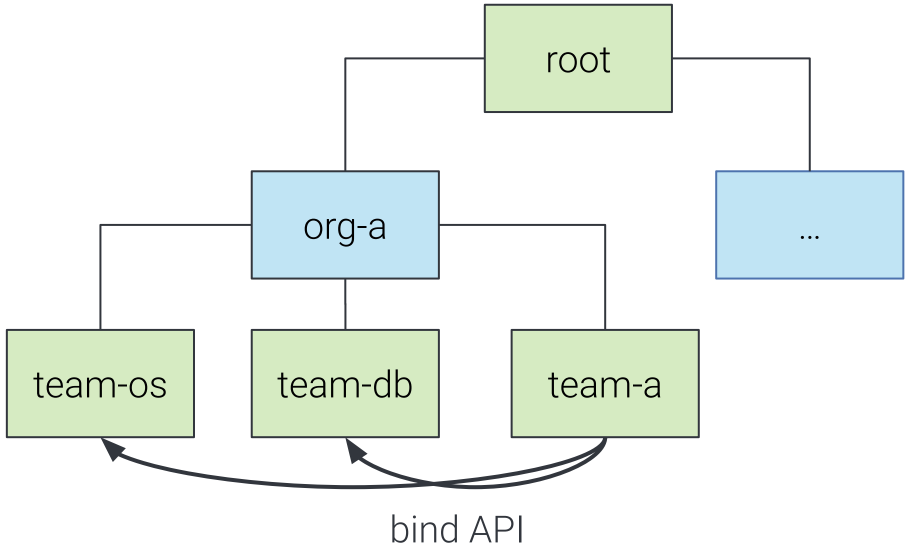
Powered by APIBindings

```
$ kubectl get apibindings
NAME                        AGE    READY
tenancy.kcp.io-3wb5h        30d    True
topology.kcp.io-cua3o       30d    True
```

```
apiVersion: apis.kcp.io/v1alpha1
kind: APIBinding
metadata:
  name: tenancy.kcp.io-3wb5h
spec:
  reference:
    export:
      name: tenancy.kcp.io
      path: root
```

This references an APIExport in a different workspace!

APIBindings across Workspaces



RBAC Extension for APIBindings

Binding to exported APIs requires RBAC permissions on the **APIExport**.

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
   name: bind-apiexport
rules:
- apiGroups:
  - apis.kcp.io
  resources:
  - apiexports
  verbs:
  - use
  resourceNames:
  - demo.embik.me
```

Wrapping Up

1. kcp is building a global control plane for API-driven platforms.
2. Workspaces allow to reconstruct organizational hierarchy.
3. Providing Kubernetes-style APIs at scale is incredibly easy.

It’s a community project! We welcome everyone to build the project’s future together.