

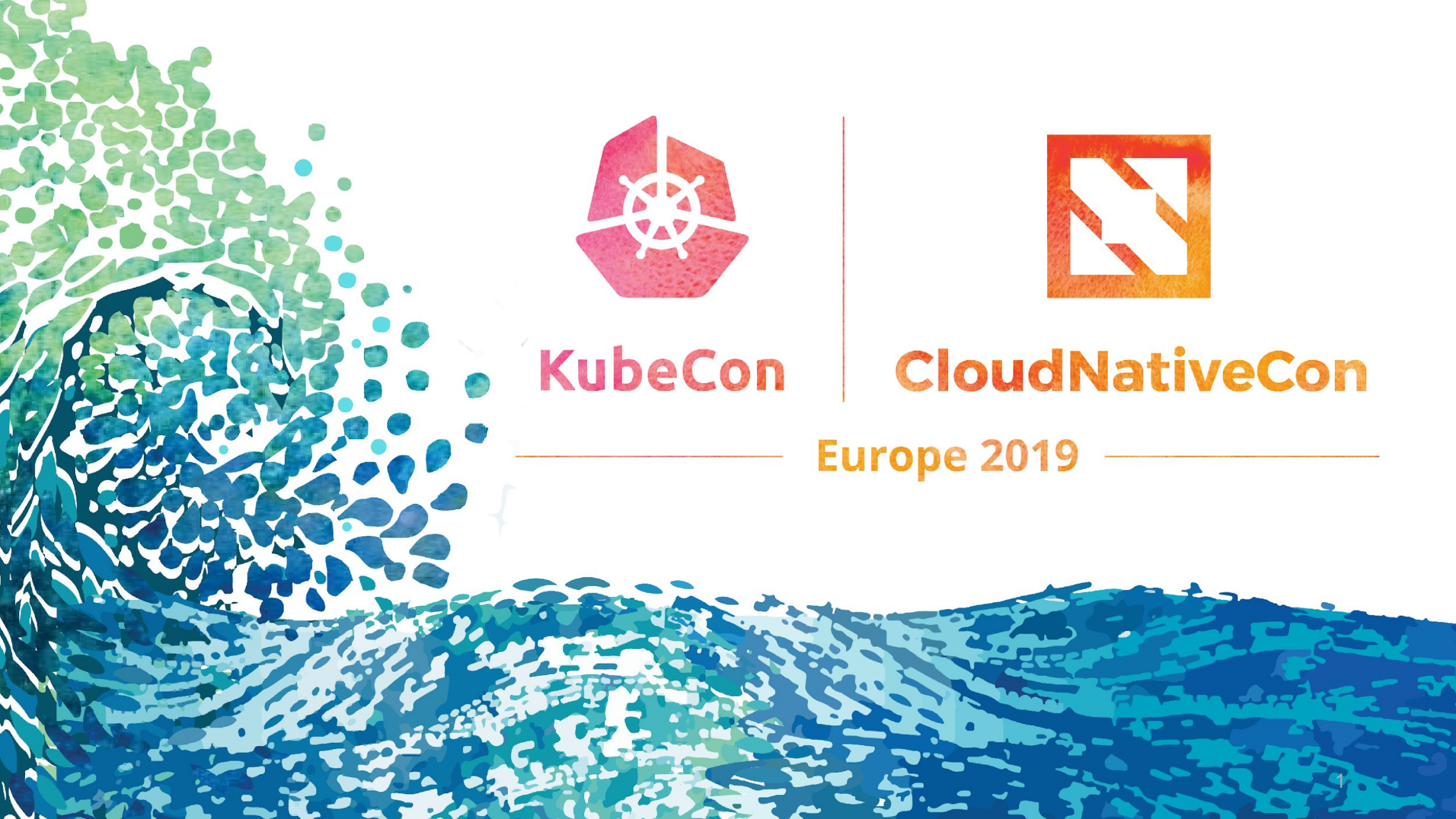


KubeCon



CloudNativeCon

Europe 2019





KubeCon



CloudNativeCon

Europe 2019

The Multicluster Toolbox

Adrien Trouillaud, Founder & CEO, Admiralty

GitHub/Twitter: @adrienjt @admiraltyio

“The Multiclustur Toolbox” will be a live demonstration.
The following slides contain supporting material, but do NOT summarize the talk.
For the full content, please attend the session, or watch the recording later.
Note: this document is a DRAFT. The final version will be uploaded later.

Can't I just use kubefed? (formerly known as Federation v2)

Maybe.

CLUSTER 0

ANY NAMESPACE

```
apiVersion: types.kubefed.k8s.io/v1alpha1
kind: FederatedDeployment
metadata:
  name: test-deployment
  namespace: test-namespace
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      replicas: 3
      selector:
        matchLabels:
          app: nginx
      template:
        metadata:
          labels:
            app: nginx
        spec:
          containers:
            - image: nginx
              name: nginx
  placement:
    clusters:
      - name: cluster2
      - name: cluster1
  overrides:
    - clusterName: cluster2
      clusterOverrides:
        - path: spec.replicas
          value: 5
```

CLUSTER 1

ANY NAMESPACE (SAME)

TEST DEPLOYMENT

REPLICA SET

POD	POD	POD
-----	-----	-----

CLUSTER 2

ANY NAMESPACE (SAME)

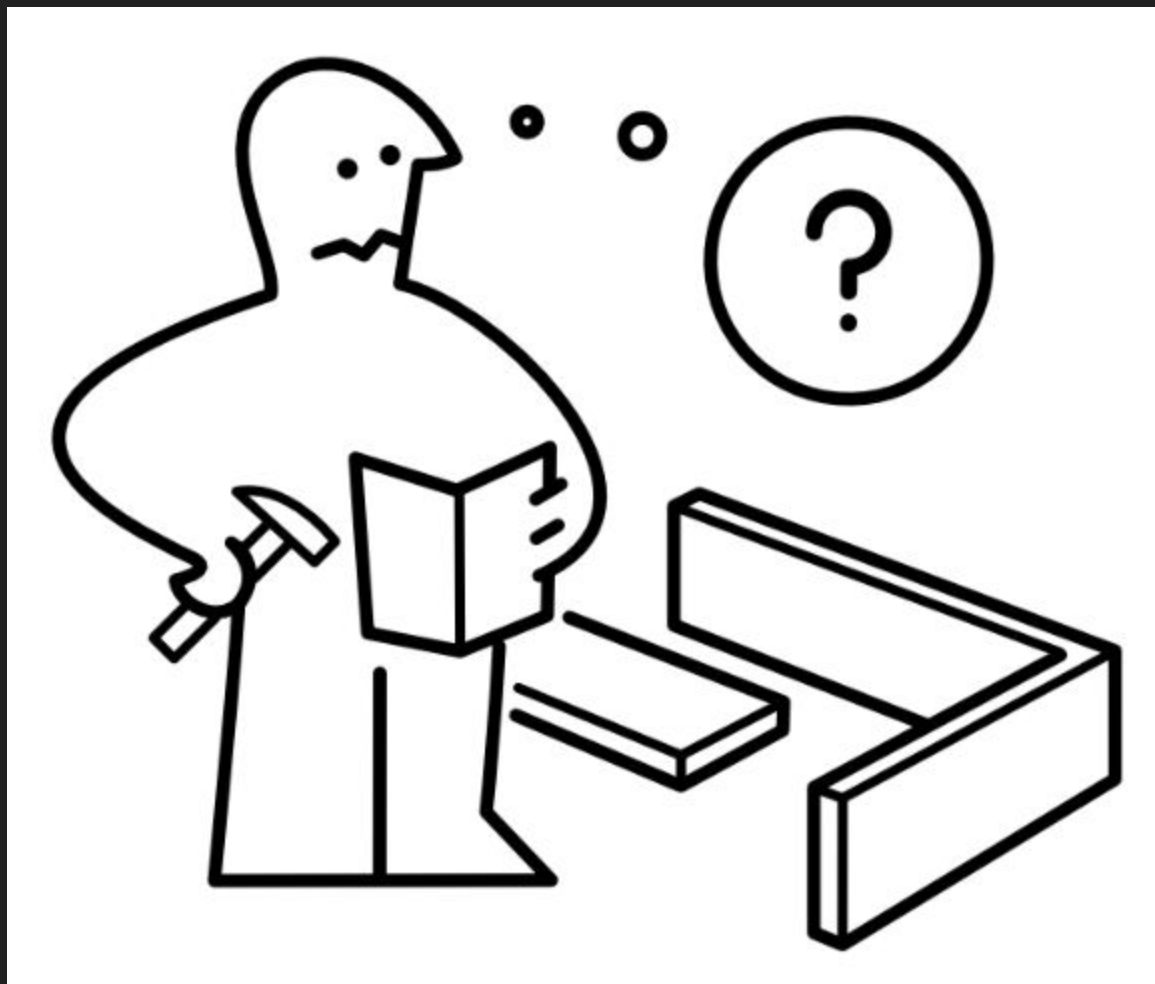
TEST DEPLOYMENT

REPLICA SET

POD	POD	
POD	POD	POD

Multicluster Application Examples

- Multi-cloud/multi-region Argo workflows
- Propagate Cilium global services across cluster mesh
- Like kubefed, but with pulling agents
- Multi-region database / storage operator
- High Throughput Computing (HTC), beyond single-cluster SLA
- Submit locally, run globally
- Customers declare X in their clusters, operated in SaaS company's cluster
- ...





1. cross-cluster control loops
(watch and reconcile in different clusters)
2. cross-cluster garbage collection
3. declarative bootstrapping
(no kubeconfig wrangling with bash scripts, no CLI)

The Multicluster Toolbox

=

<https://github.com/admiralty.io/multicluster-controller>
(a multicluster controller-runtime)

+

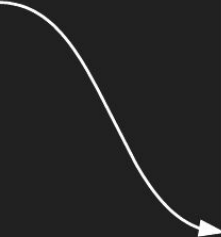
<https://github.com/admiralty.io/multicluster-service-account>
(import and automount remote service accounts)

<https://github.com/kubernetes/sample-controller>

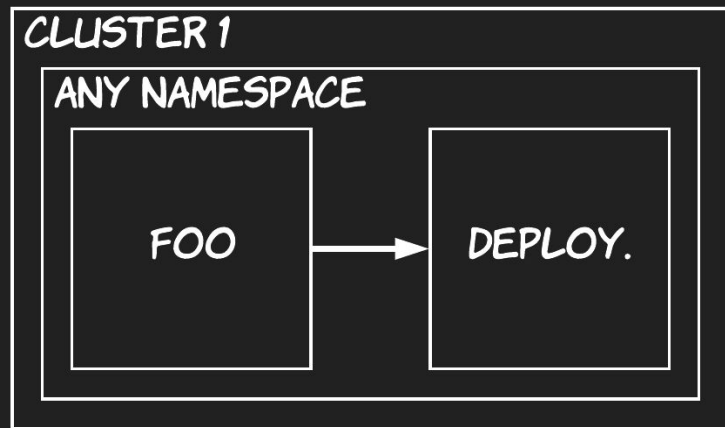
CLUSTER 1

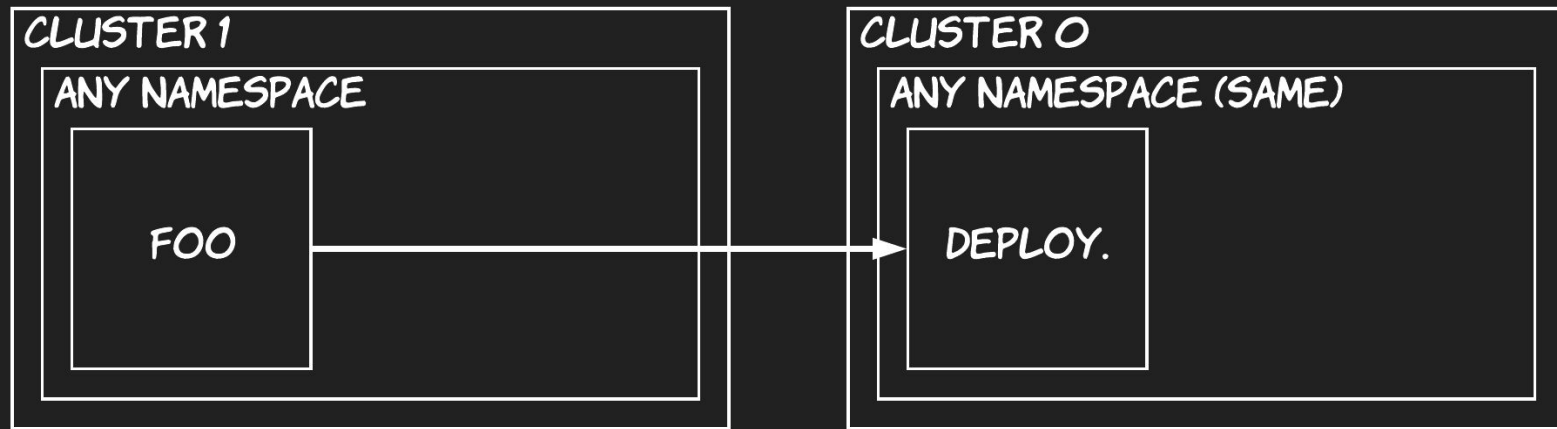
ANY NAMESPACE

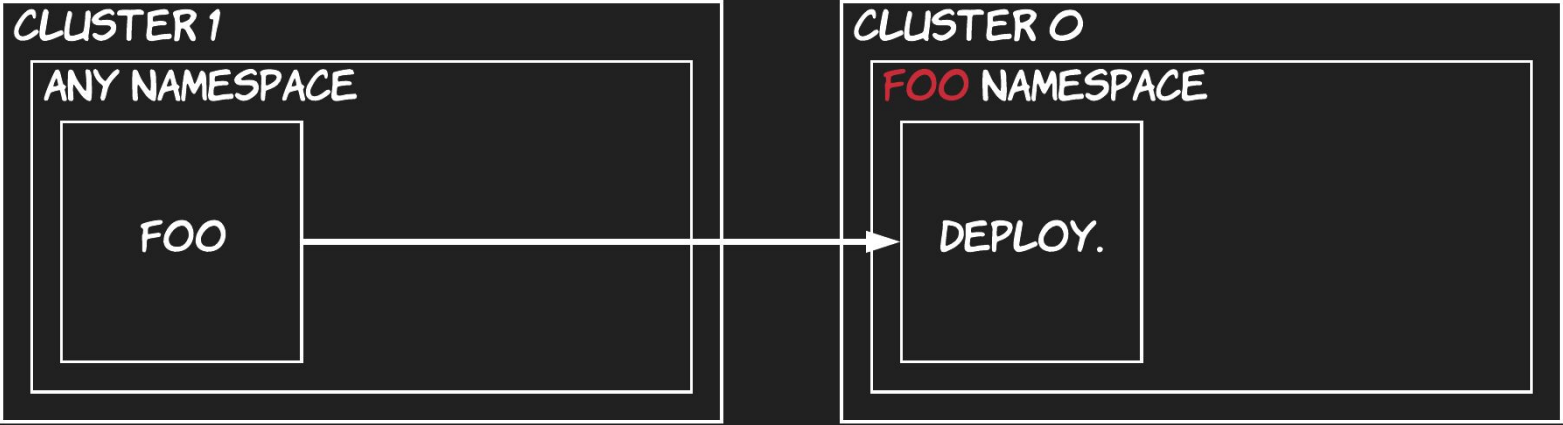
```
apiVersion: samplecontroller.k8s.io/  
v1alpha1  
kind: Foo  
metadata:  
  name: example-foo  
  namespace: default  
spec:  
  deploymentName: example-foo  
  replicas: 1
```

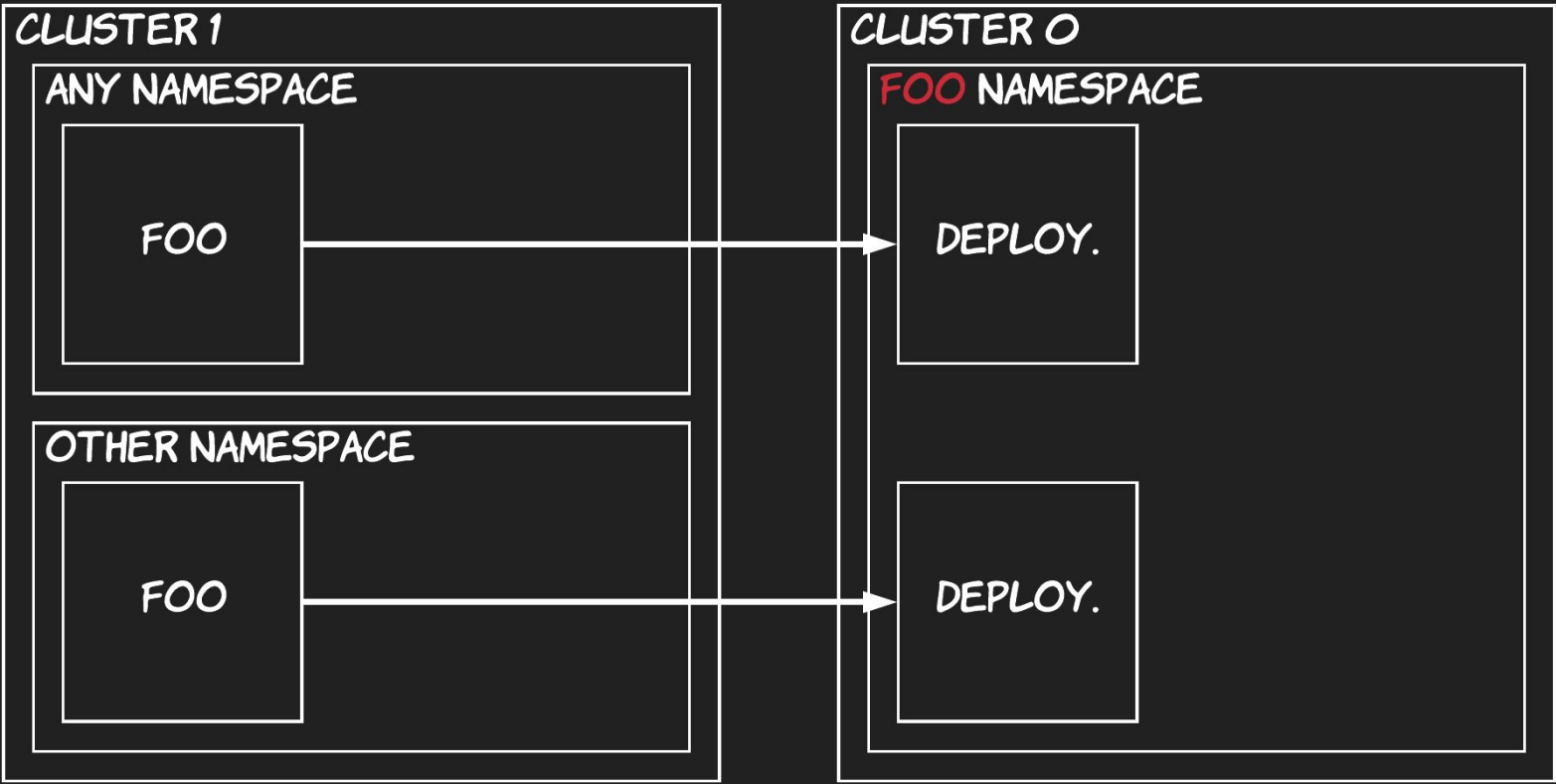


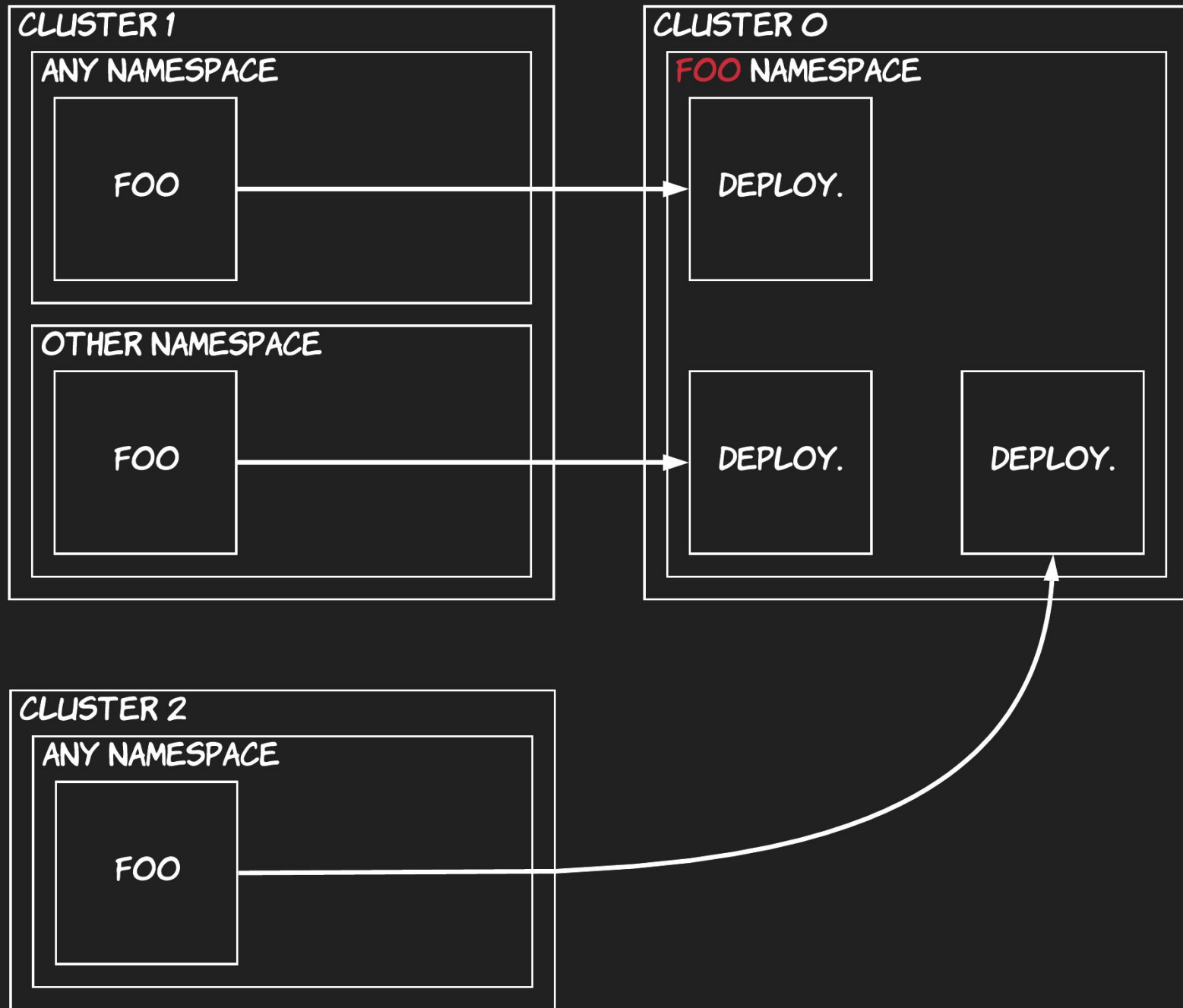
```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: example-foo  
  namespace: default  
  ownerReferences: ...  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: nginx  
      controller: example-foo  
  template:  
    metadata:  
      labels:  
        app: nginx  
        controller: example-foo  
    spec:  
      containers:  
        - image: nginx  
          name: nginx:latest
```



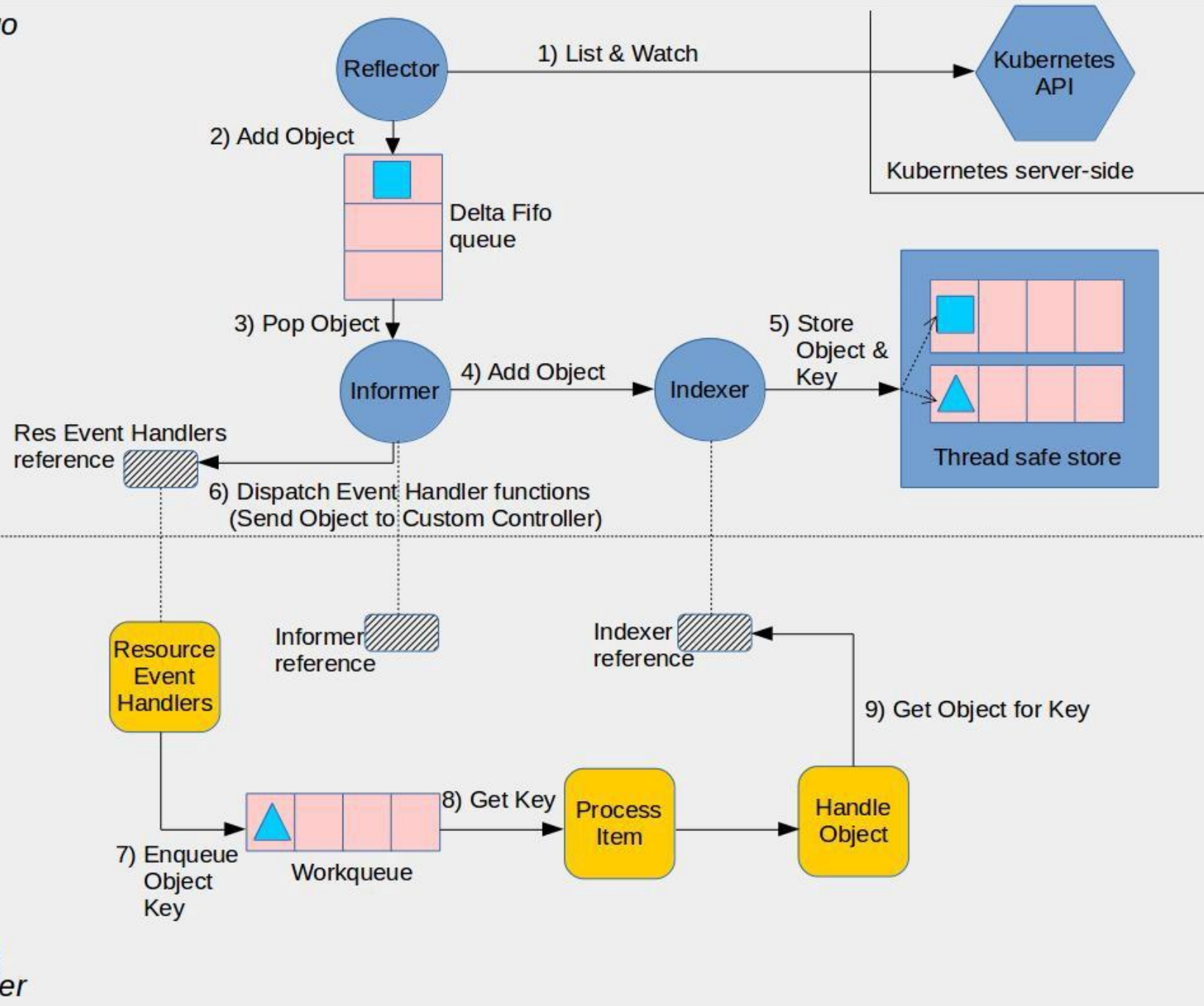




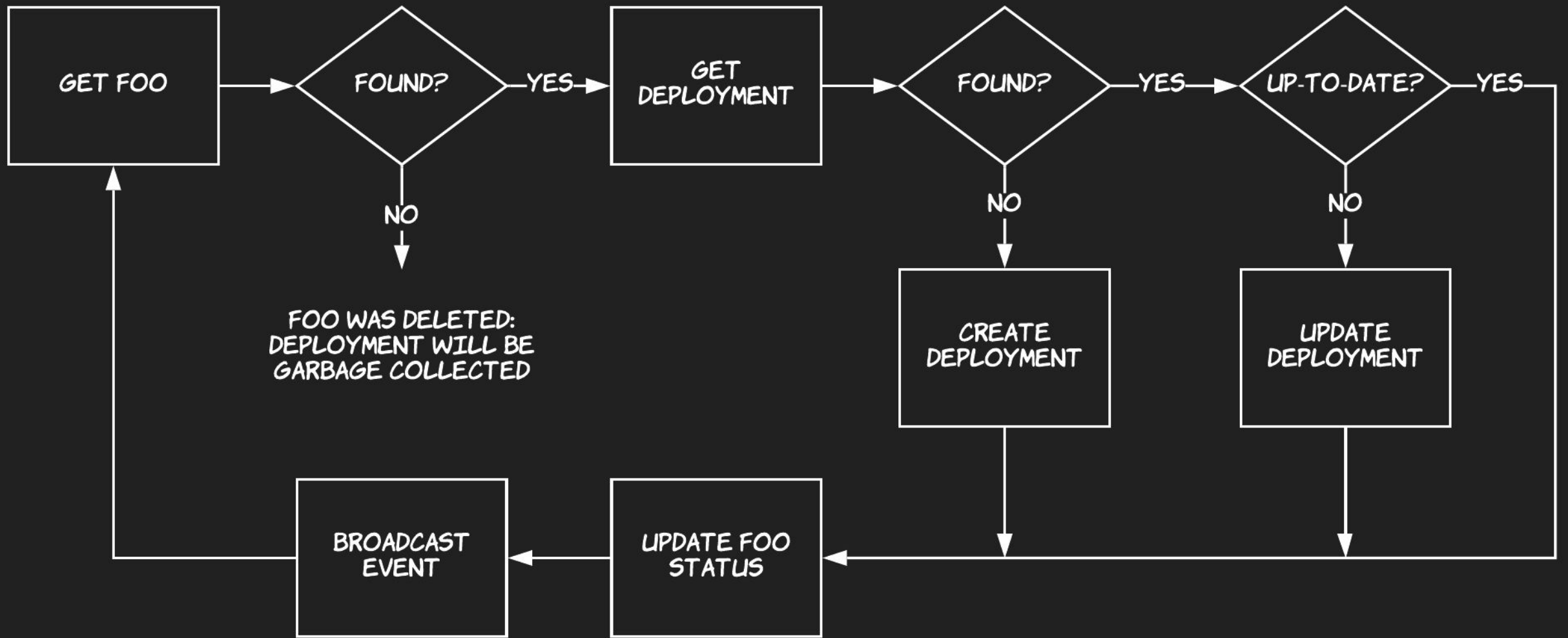




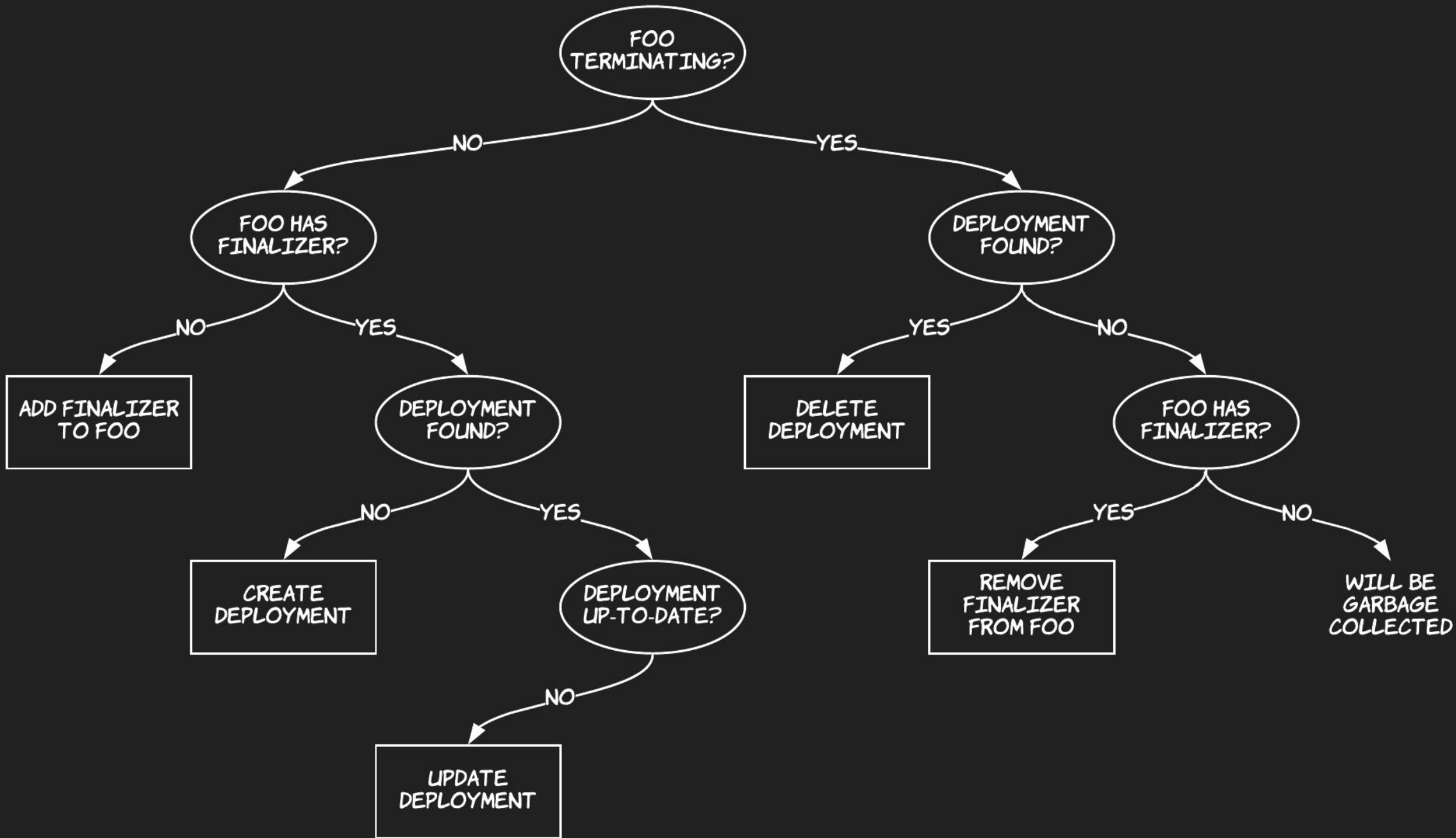
contributed by Devdatta Kulkarni
originally published in CloudARK's blog



```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
ownerReferences:
- apiVersion: samplecontroller.k8s.io/v1alpha1
  controller: true
  blockOwnerDeletion: true
  kind: Foo
  name: example-foo
  uid: d9607e19-f88f-11e6-a518-42010a800195
  ...
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
  deletionTimestamp: 2019-05-21T09:42:00Z
  ...
  finalizers:
  - foregroundDeletion
  ...
```



```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: foo-controller-remote
  namespace: foo
```



```
apiVersion: v1
kind: Secret
metadata:
  name: foo-controller-remote-token-6456p
  namespace: foo
  ...
type: kubernetes.io/service-account-token
data:
  ca.crt: ...
  namespace: ...
  token: ...
```

```
apiVersion: v1
kind: Config
currentContext: foo-controller-remote-cluster1
contexts:
- name: foo-controller-remote-cluster1
  context:
    namespace: foo
    cluster: cluster1
    user: foo-controller-remote
clusters:
- name: cluster1
  cluster:
    server: https://...
    certificate-authority-data: ...
users:
- name: foo-controller-remote
  user:
    token: ...
```

1. Set the environment variables needed to build the `kubeconfig` file for the `istio-multi` service account with the following commands:

```
$ export WORK_DIR=$(pwd)
$ CLUSTER_NAME=$(kubectl config view --minify=true -o "jsonpath={.clusters[].name}")
$ export KUBECFG_FILE=${WORK_DIR}/${CLUSTER_NAME}
$ SERVER=$(kubectl config view --minify=true -o "jsonpath={.clusters[].cluster.server}")
$ NAMESPACE=istio-system
$ SERVICE_ACCOUNT=istio-multi
$ SECRET_NAME=$(kubectl get sa ${SERVICE_ACCOUNT} -n ${NAMESPACE} -o jsonpath='{.secrets[].name}')
$ CA_DATA=$(kubectl get secret ${SECRET_NAME} -n ${NAMESPACE} -o "jsonpath={.data['ca.crt']}")
$ TOKEN=$(kubectl get secret ${SECRET_NAME} -n ${NAMESPACE} -o "jsonpath={.data['token']}" | base64 --decode)
```

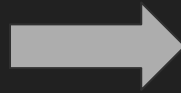
i An alternative to `base64 --decode` is `openssl enc -d -base64 -A` on many systems.

2. Create a `kubeconfig` file in the working directory for the `istio-multi` service account with the following command:

```
cat <<EOF > ${KUBECFG_FILE}
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: ${CA_DATA}
  server: ${SERVER}
  name: ${CLUSTER_NAME}
contexts:
- context:
  cluster: ${CLUSTER_NAME}
  user: ${CLUSTER_NAME}
  name: ${CLUSTER_NAME}
current-context: ${CLUSTER_NAME}
kind: Config
preferences: {}
users:
- name: ${CLUSTER_NAME}
  user:
  token: ${TOKEN}
EOF
```



```
apiVersion: multicluster.admiralty.io/v1alpha1
kind: ServiceAccountImport
metadata:
  name: foo-controller-remote-cluster1
  namespace: foo
spec:
  clusterName: cluster1
  namespace: foo
  name: foo-controller-remote
```



```
apiVersion: v1
kind: Secret
metadata:
  name: foo-controller-remote-cluster1-token-6456p
  namespace: foo
  ...
type: Opaque
data:
  config: ... # serialized kubeconfig
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: foo
  labels:
    multicluster-service-account: enabled
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: foo-controller
  namespace: foo
spec:
  selector:
    matchLabels:
      app: foo-controller
  template:
    metadata:
      labels:
        app: foo-controller
      annotations:
        multicluster.admiralty.io/service-account-import.name:
foo-controller-remote-cluster1,foo-controller-remote-cluster2
    spec:
      ...
```



References

@adrienjt

@admiraltyio

<https://admiralty.io/#blog>

<https://admiralty.io/#open-source>

multicluster-controller, multicluster-service-account, multicluster-scheduler

<https://github.com/kubernetes/sample-controller>