
JETSTACK
Presented by James Laverack, Solutions Engineer
A Kubernetes Operator for etcd
jetstack.io

Open Source Tools

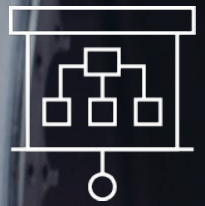
Enhancing the Kubernetes experience with open source tooling. Includes provisioning, stateful services and security related projects

Subscription

Reference architecture, online training and SLA support 24x7 for your production Kubernetes deployment

Consulting

Consulting and engineering to make the most of Kubernetes and move you to production quickly



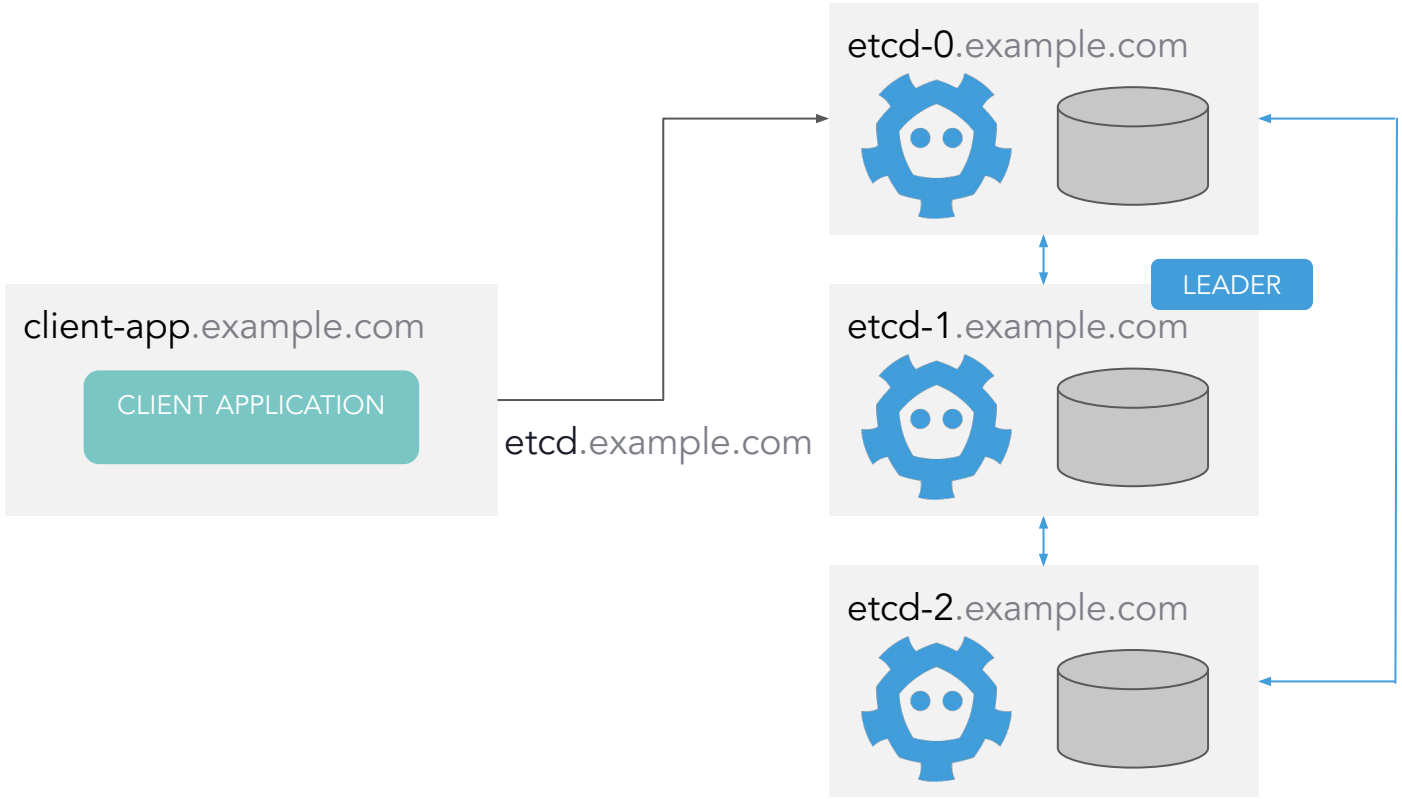
Training

Learn and engage directly alongside our team, with courses for all stages of your Kubernetes journey

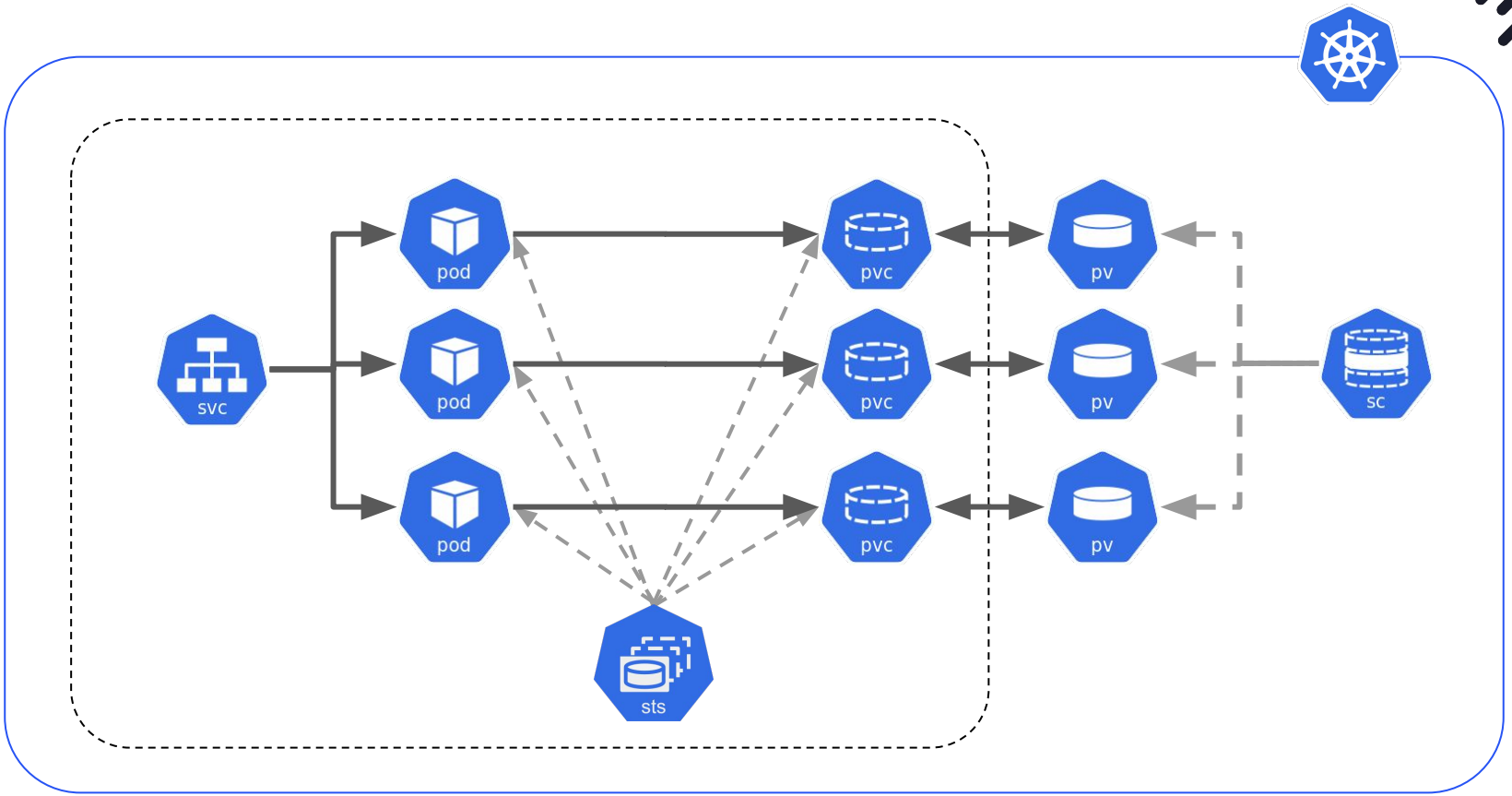
jetstack.io

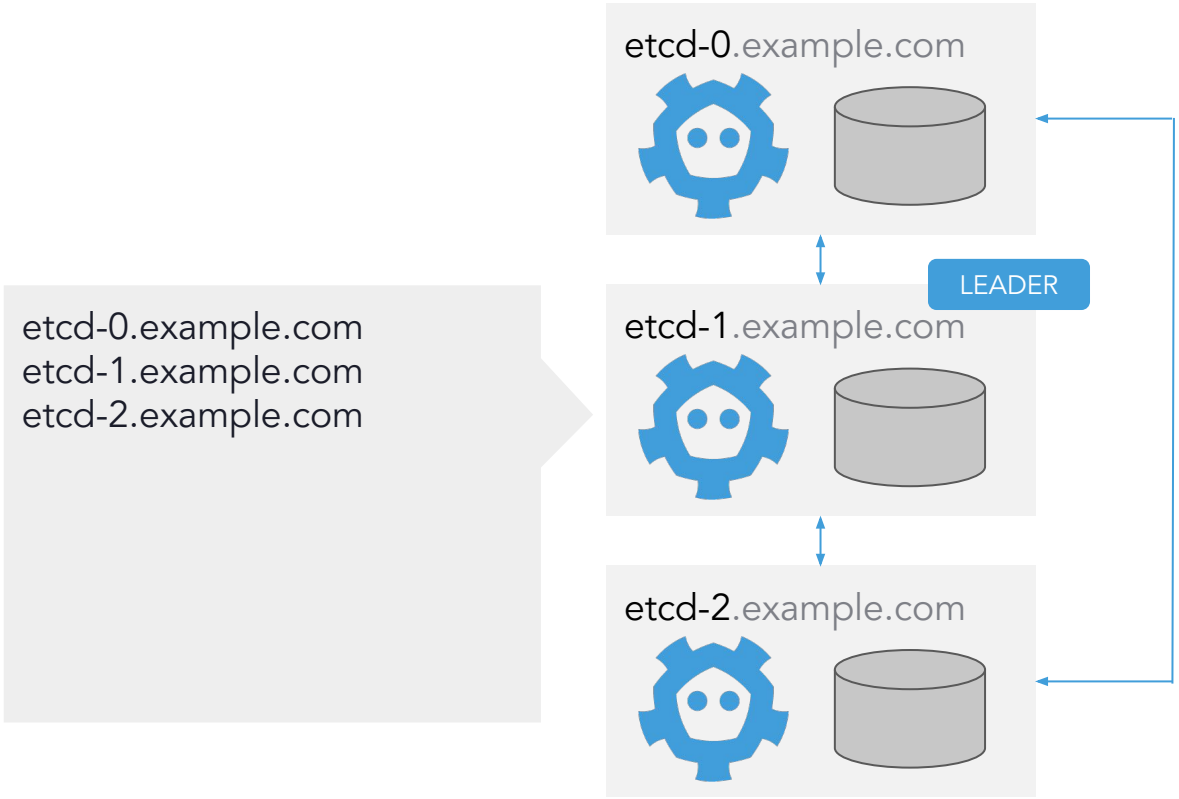
“We need to run etcd in Kubernetes.”

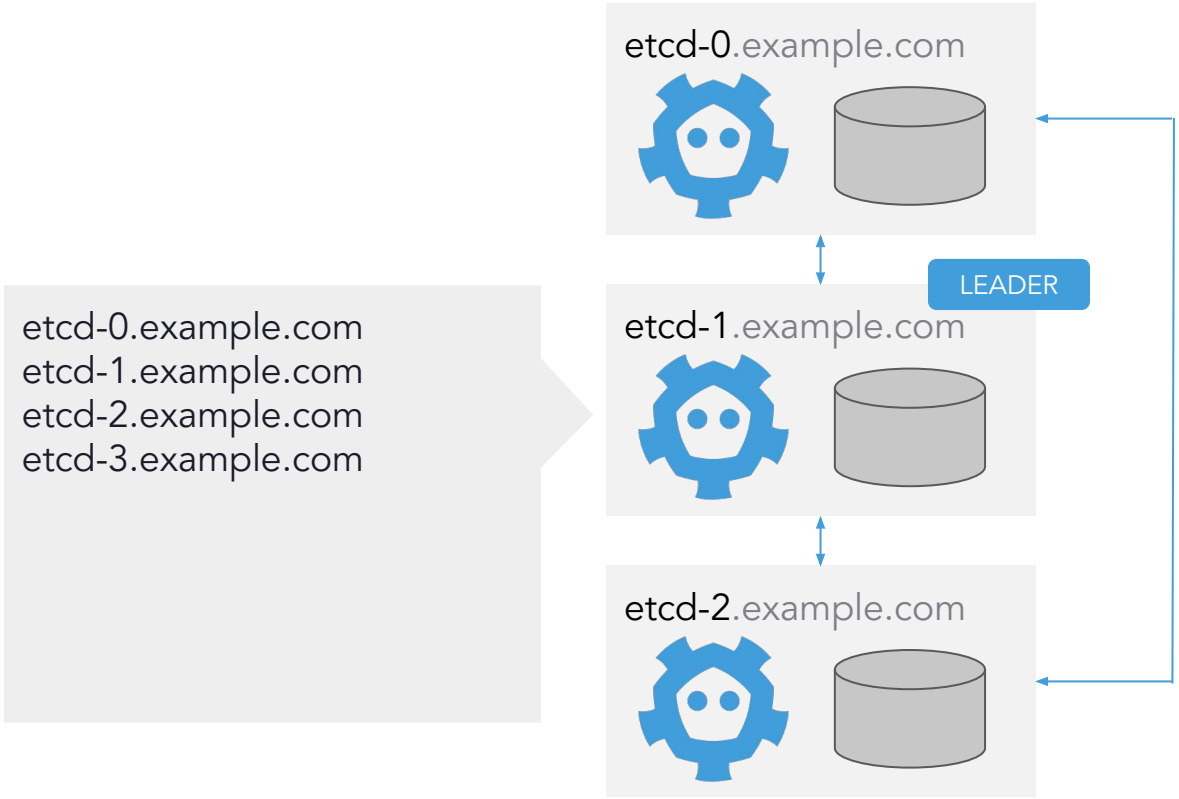
etcd

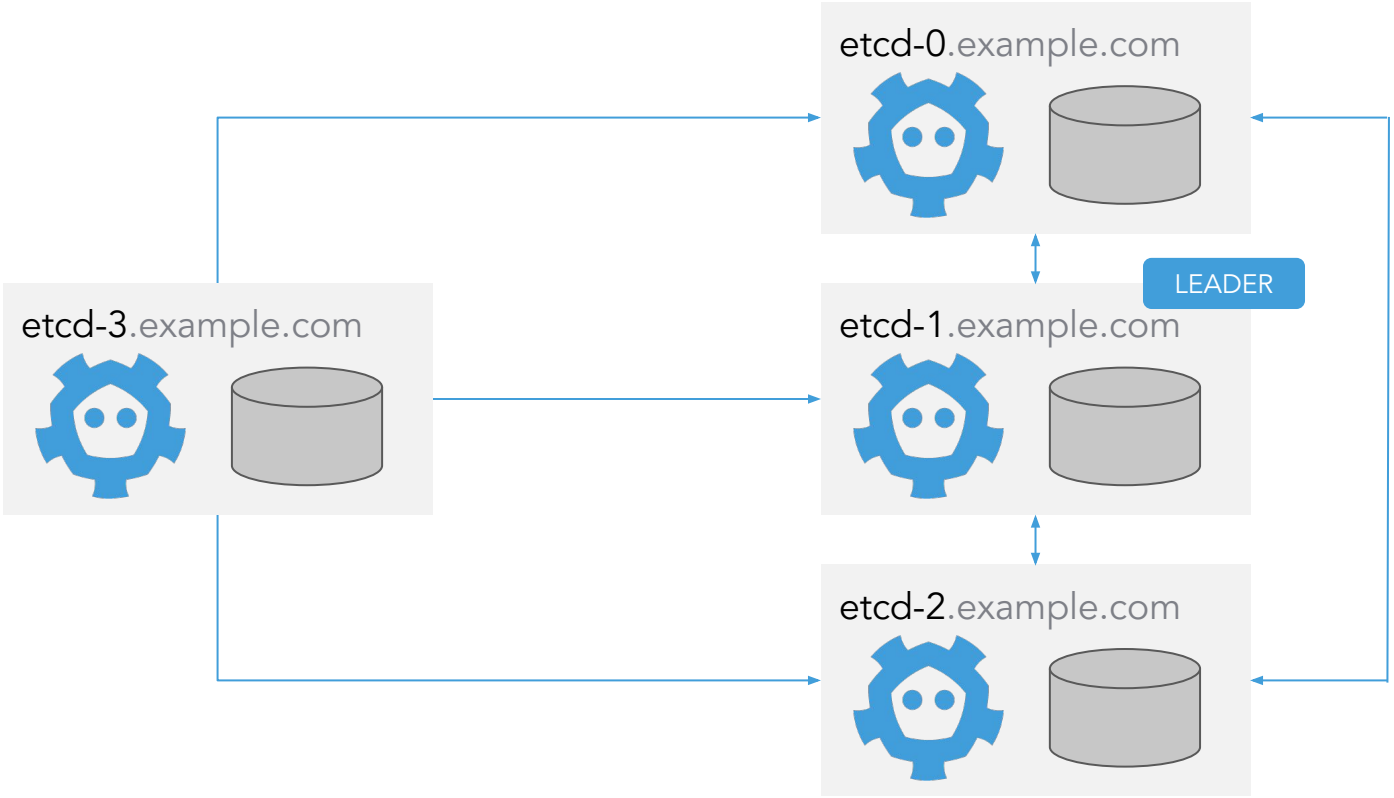

etcd-0.example.com
LEADER
client-app.example.com
CLIENT APPLICATION
etcd.example.com
etcd-1.example.com
etcd-2.example.com

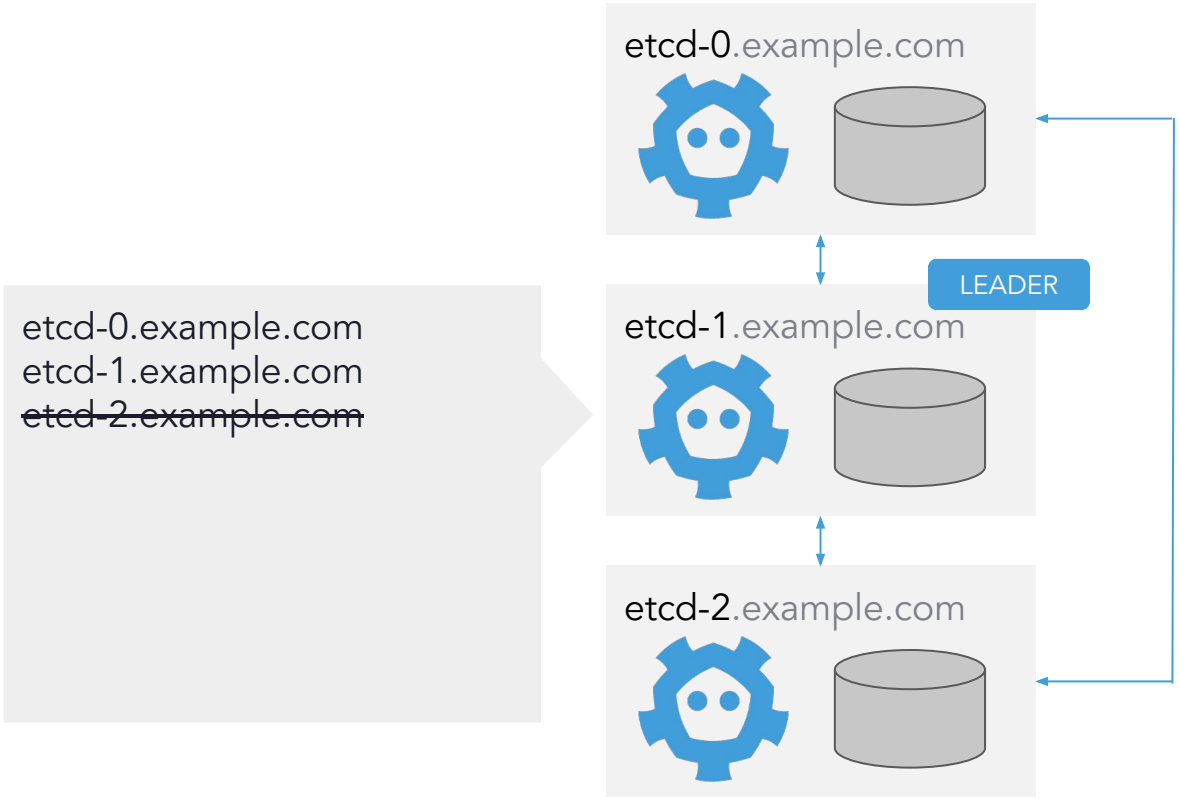
“What about a stateful set?”

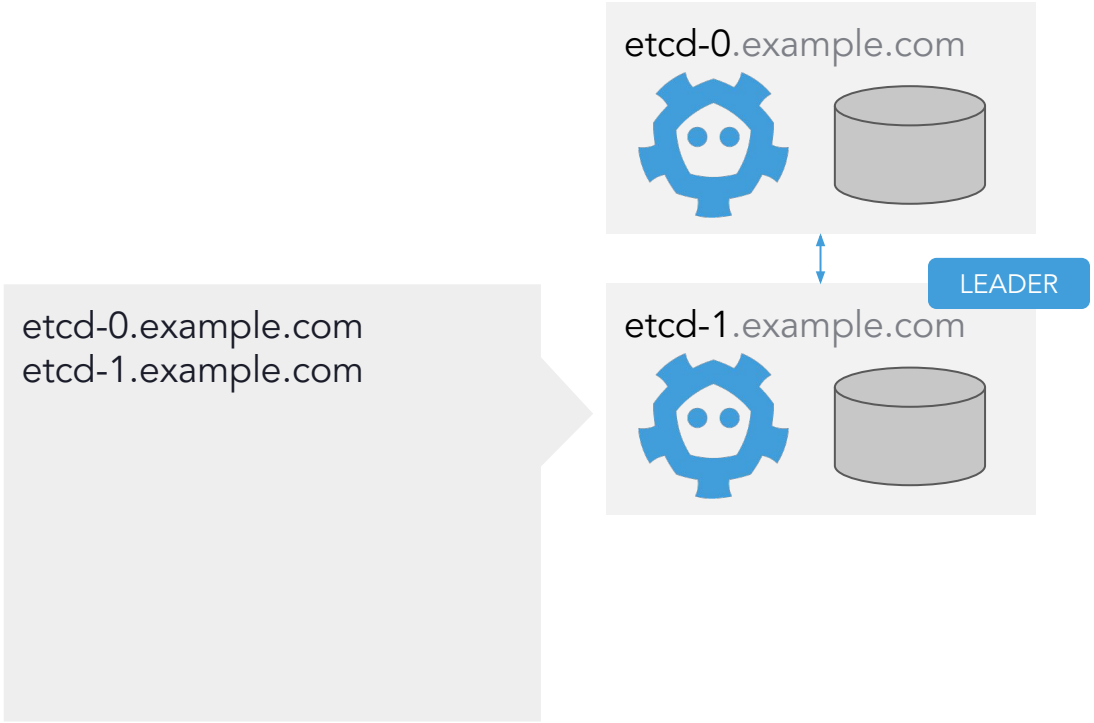


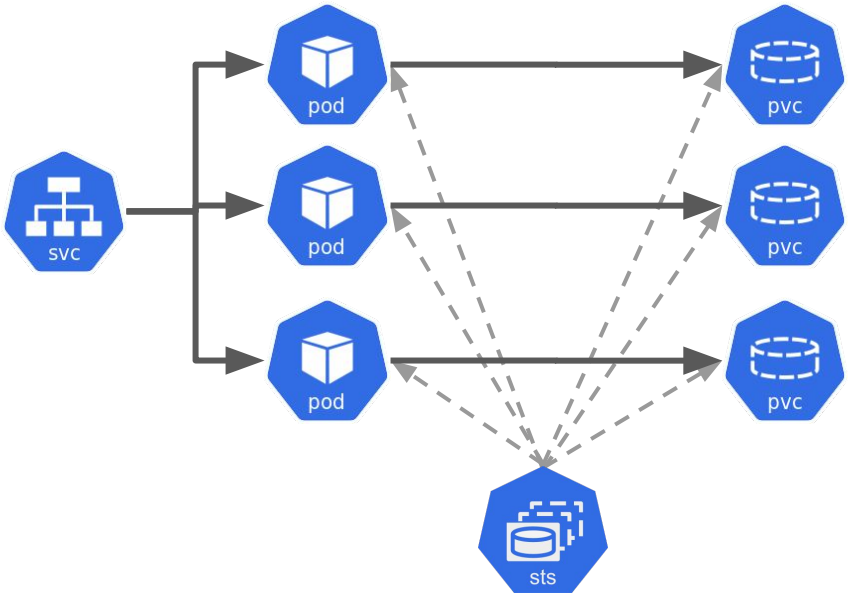

etcd-0.example.com
etcd-1.example.com
etcd-2.example.com
LEADER
etcd-0.example.com
etcd-1.example.com
etcd-2.example.com


etcd-0.example.com
etcd-1.example.com
etcd-2.example.com
etcd-3.example.com
etcd-0.example.com
LEADER
etcd-1.example.com
etcd-2.example.com


etcd-0.example.com
LEADER
etcd-3.example.com
etcd-1.example.com
etcd-2.example.com


etcd-0.example.com
etcd-1.example.com
~~etcd-2.example.com~~
etcd-0.example.com
LEADER
etcd-1.example.com
etcd-2.example.com


etcd-0.example.com
etcd-1.example.com
etcd-0.example.com
LEADER
etcd-1.example.com

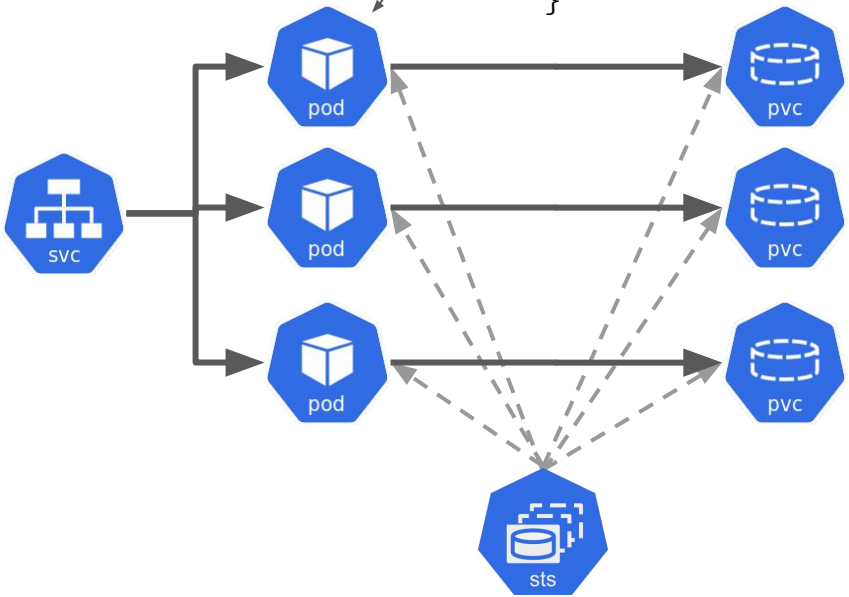


Init container:

```
if !bootstrap and firstLaunch {
  err := addPeerToEtcd(peer)
  if err != nil {
    // Comms failure?
    fail()
  }
  markAsLaunched()
}
```

Pre-stop hook:

```
err := removePeerFromEtcd(peer)
if err != nil {
  // Uhh…
  // Shut down anyway?
}
```

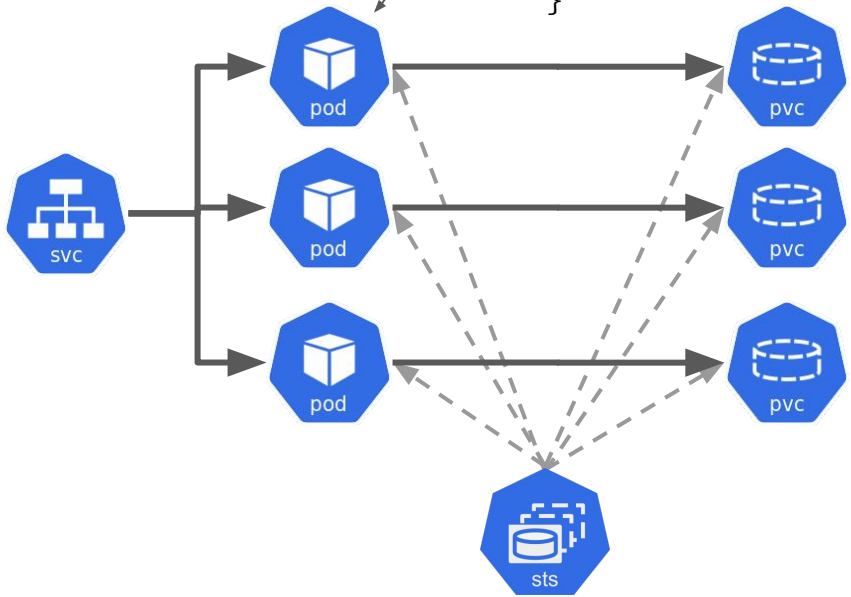


Init container:

```
if !bootstrap and firstLaunch {
  err := addPeerToEtcd(peer)
  if err != nil {
    // Comms failure?
    fail()
  }
  markAsLaunched()
}
```

Pre-stop hook:

```
err := removePeerFromEtcd(peer)
if err != nil {
  // Uhh…
  // Shut down anyway?
}
```



Only if we're really removing a peer, not if we just move the Pod…

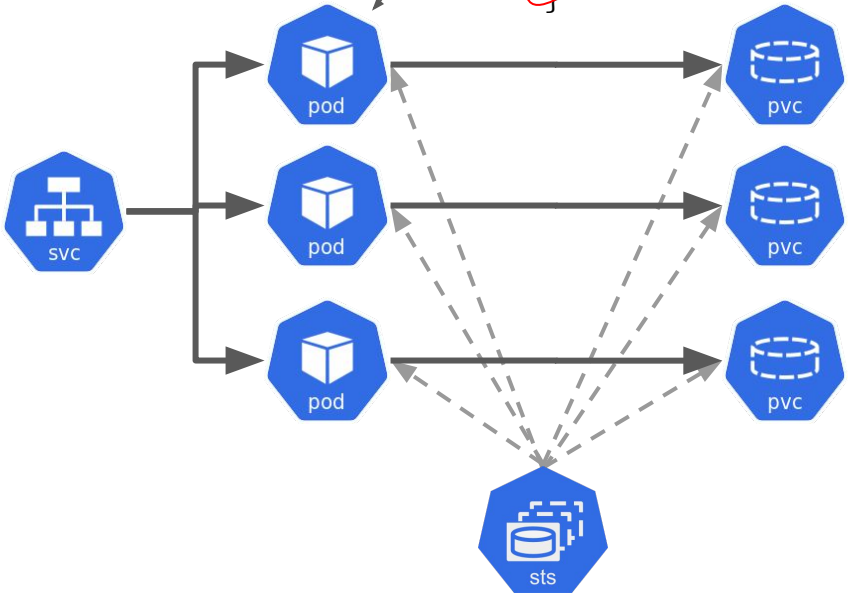
Init container:

```
if !bootstrap and firstLaunch {
  err := addPeerToEtcd(peer)
  if err != nil {
    // Comms failure?
    fail()
  }
  markAsLaunched()
}
```

Pre-stop hook:

```
err := removePeerFromEtcd(peer)
if err != nil {
  // Uhh…
  // Shut down anyway?
}
```

Only if we're really removing a peer, not if we just move the Pod…

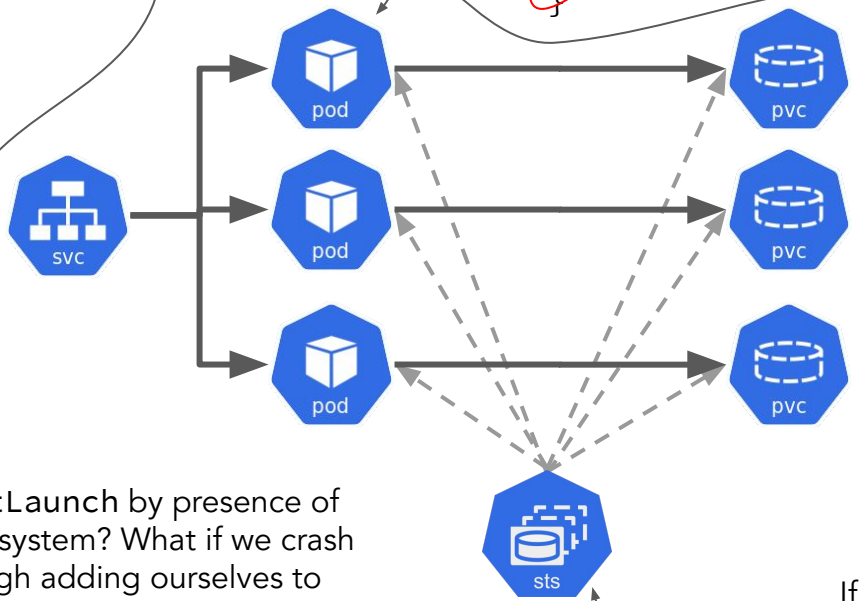


Init container:

```
if !bootstrap and firstLaunch {
  err := addPeerToEtcd(peer)
  if err != nil {
    // Comms failure?
    fail()
  }
  markAsLaunched()
}
```

Pre-stop hook:

```
err := removePeerFromEtcd(peer)
if err != nil {
  // Uhh…
  // Shut down anyway?
}
```



Detect `bootstrap` by trying to dial the cluster? What if it's just a comms failure? How do we recover from that?

Detect `firstLaunch` by presence of file on the filesystem? What if we crash halfway through adding ourselves to etcd? Can we retry safely?

Only if we're really removing a peer, not if we just move the Pod...

If we are bootstrapping, how do we determine the domain names of all the other peers? Query the STS to check the size? What if someone scales it while we're bootstrapping? Means the etcd pod needs permissions to list stateful sets...

If we scale to zero do we delete all the PVCs and all the data?

We need an Operator.

"Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components."

kubernetes.io/docs/concepts/extend-kubernetes/operator

“The Operator pattern aims to capture the key aim of a human operator [...] Human operators [...] have deep knowledge of how the system ought to behave, how to deploy it, and how to react if there are problems.”

kubernetes.io/docs/concepts/extend-kubernetes/operator

An Operator encodes knowledge.

cert-manager.io

strimzi.io

And many more

An Operator represents human operational knowledge in software, to reliably manage an application.

CoreOS

To demonstrate the Operator concept in running code, we have two concrete examples to announce as open source projects today:

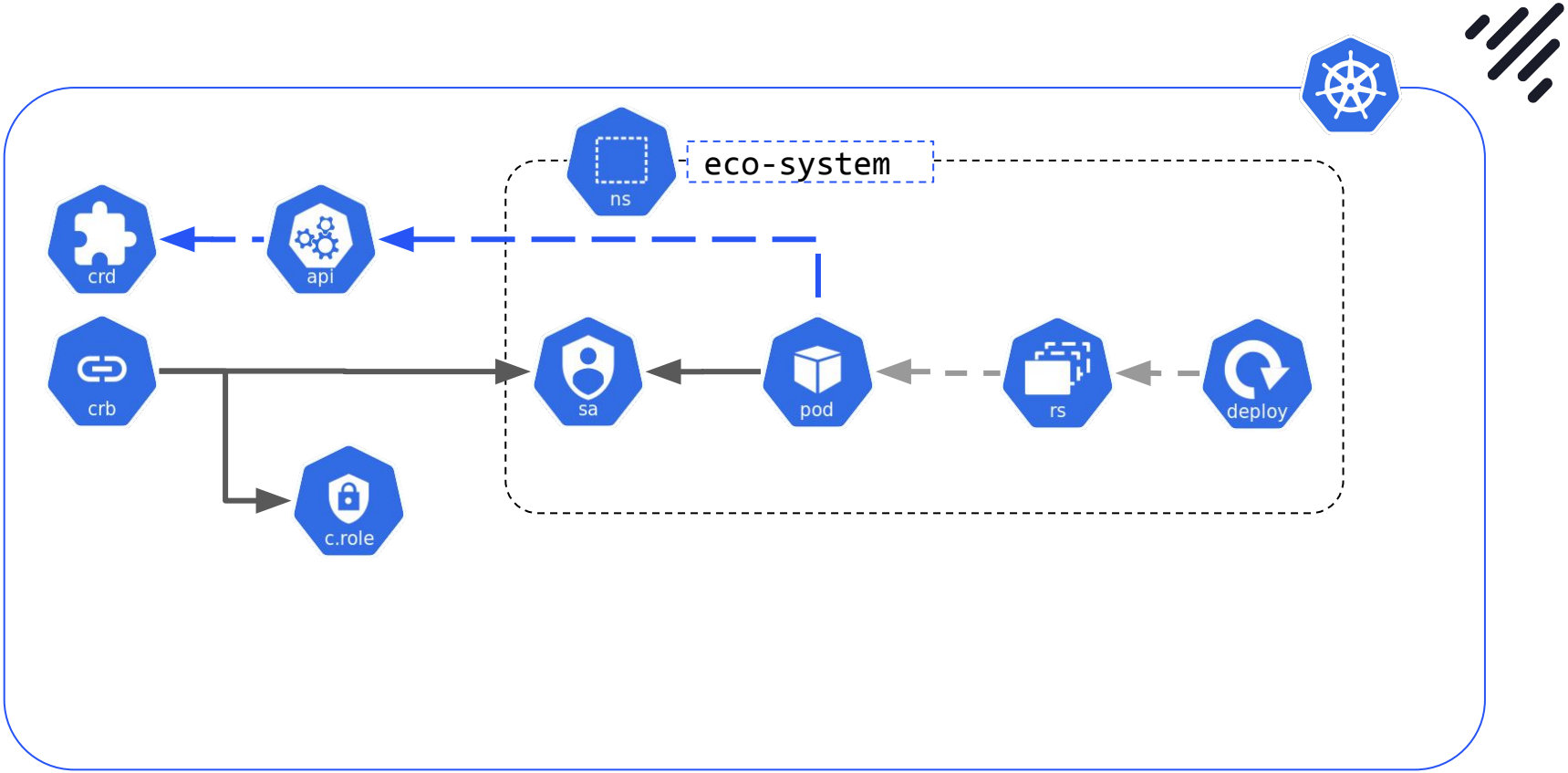
1. The *etcd Operator* creates, configures, and manages etcd clusters. etcd is a reliable, distributed key-value store introduced by CoreOS for sustaining the most critical data in a distributed system, and is the primary configuration datastore of Kubernetes itself.
2. The *Prometheus Operator* creates, configures, and manages Prometheus monitoring instances. Prometheus is a powerful monitoring, metrics, and alerting tool, and a Cloud Native Computing Foundation (CNCF) project supported by the CoreOS team.

coreos.com/blog/introducing-operators

“How do you actually build an Operator?”

`EtcdCluster`

```
$ kubectl api-resources
NAME          SHORTNAMES   APIGROUP                NAMESPACED   KIND
deployments   deploy       apps                    true         Deployment
replicasets   rs           apps                    true         ReplicaSet
statefulsets  sts          apps                    true         StatefulSet
pods          po                                   true         Pod
etcdclusters               etcd.improbable.io      true         EtcdCluster
< many lines omitted >
```

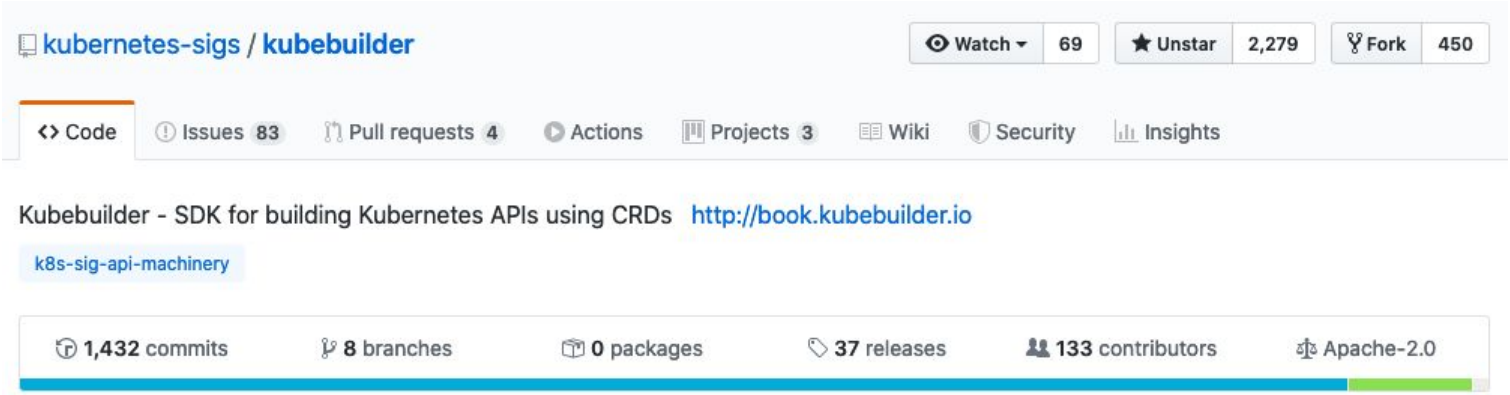


```
- apiGroups:
  - etcd.improbable.io
  resources:
  - etcdclusters
  verbs:
  - get
  - list
  - watch
```

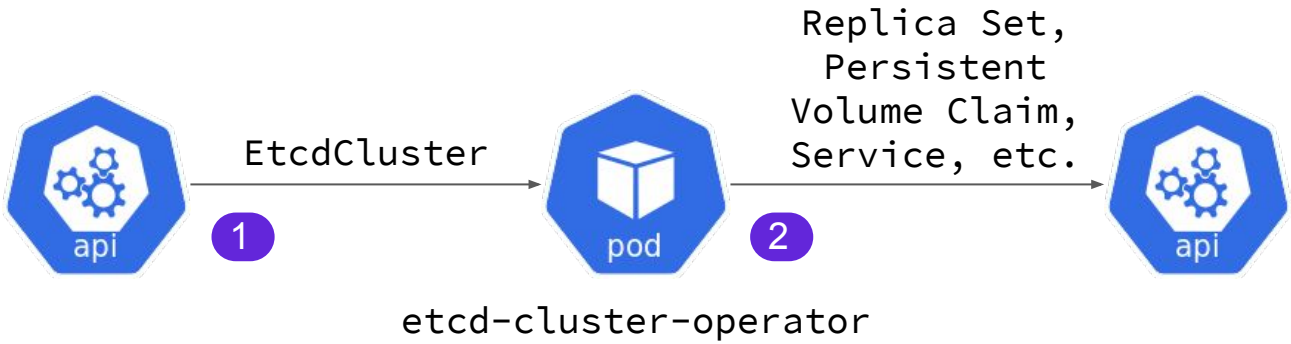
```
- apiGroups:
  - apps
  resources:
  - replicasets
  verbs:
  - create
  - get
  - list
  - watch
```

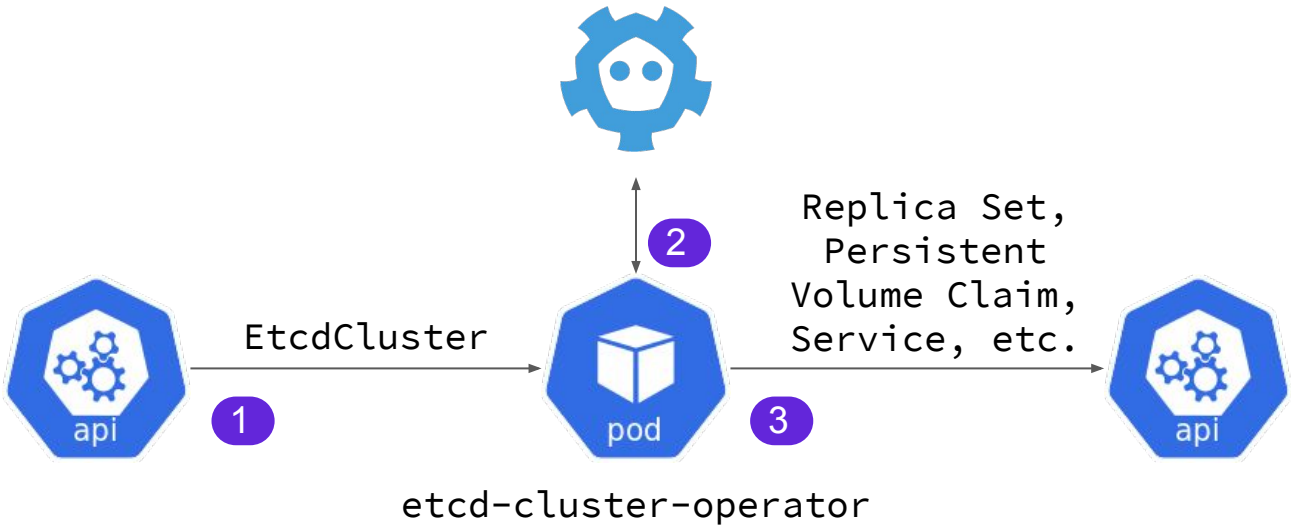
```
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - create
  - get
  - list
  - watch
```


GO


kubernetes-sigs / kubebuilder
Watch 69
Unstar 2,279
Fork 450
Code
Issues 83
Pull requests 4
Actions
Projects 3
Wiki
Security
Insights
Kubebuilder - SDK for building Kubernetes APIs using CRDs http://book.kubebuilder.io
k8s-sig-api-machinery
1,432 commits
8 branches
0 packages
37 releases
133 contributors
Apache-2.0

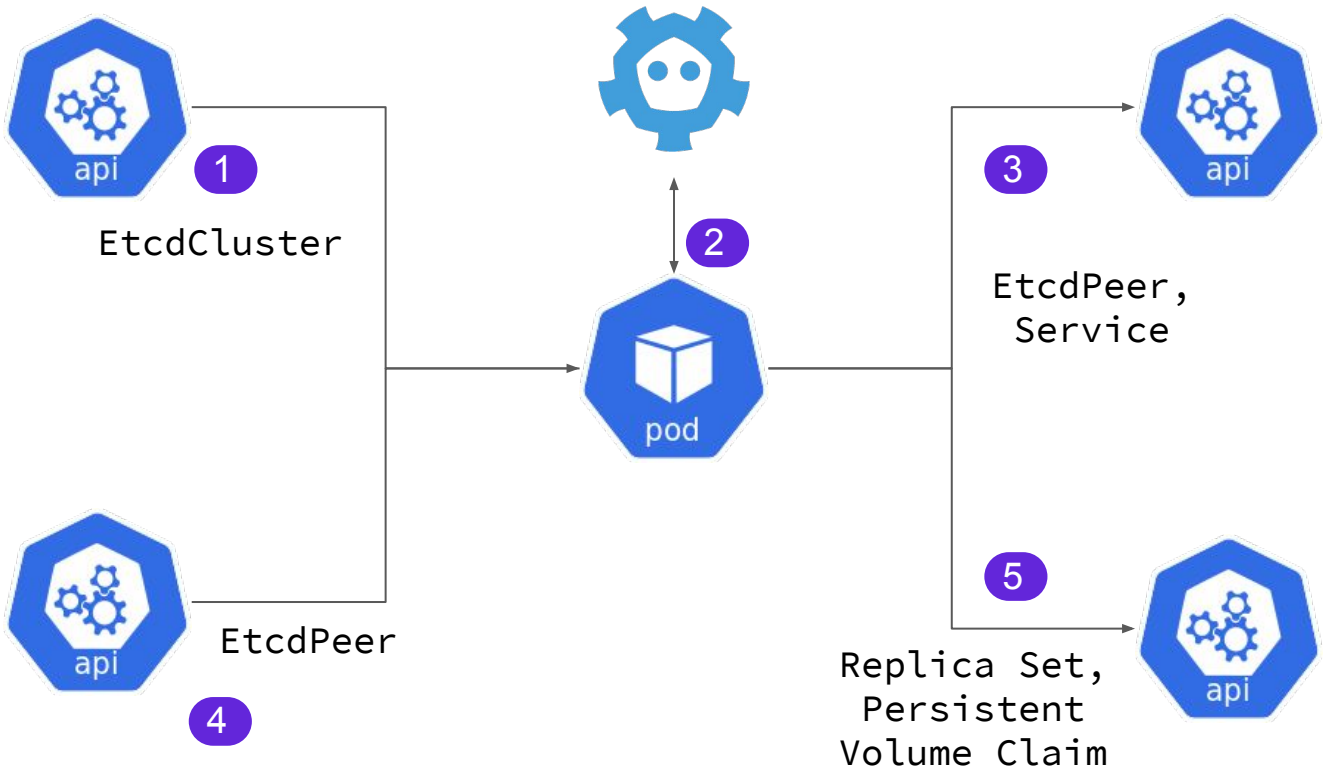
Operator logic





`EtcdCluster`

`EtcdPeer`



Design considerations

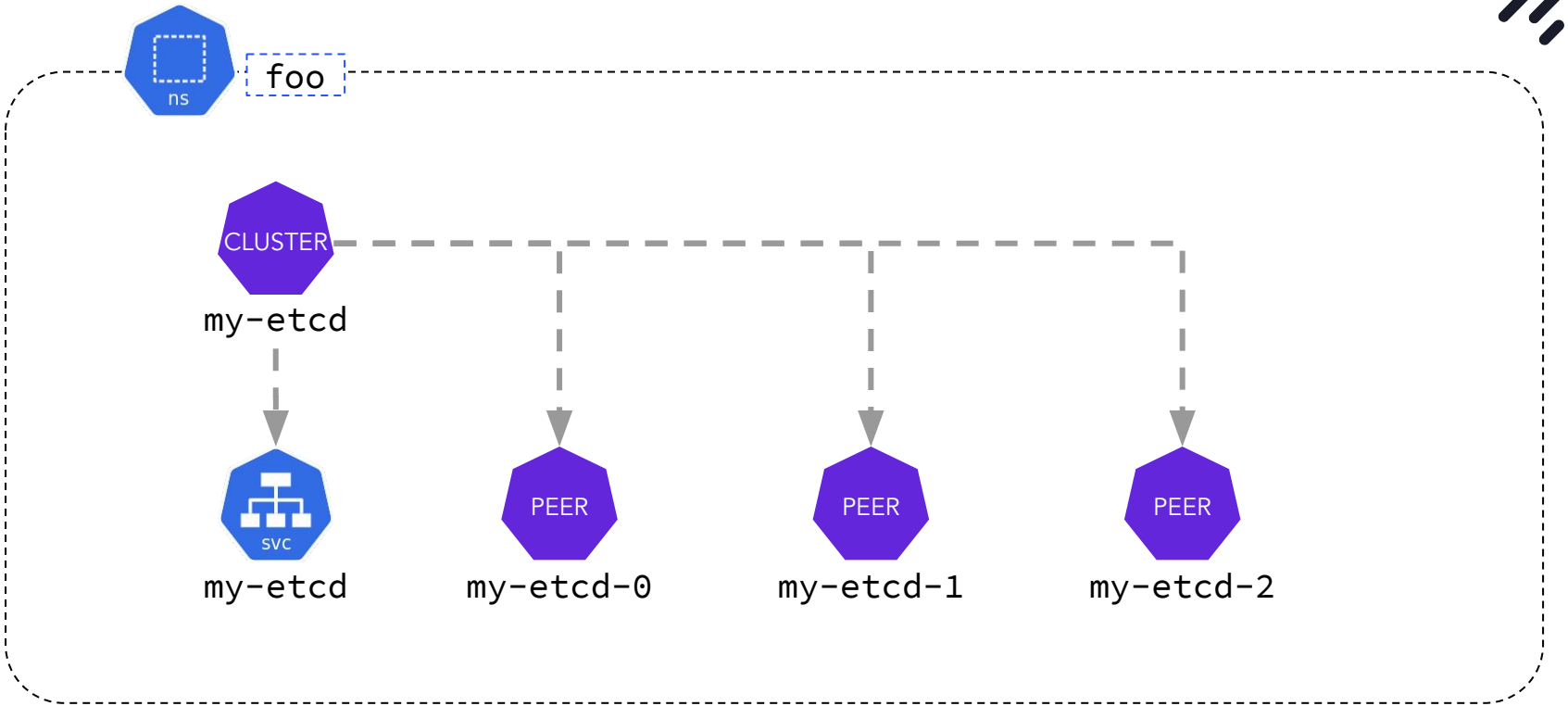
Be level-triggered.

Do one thing at a time.

The cache might lie to you.

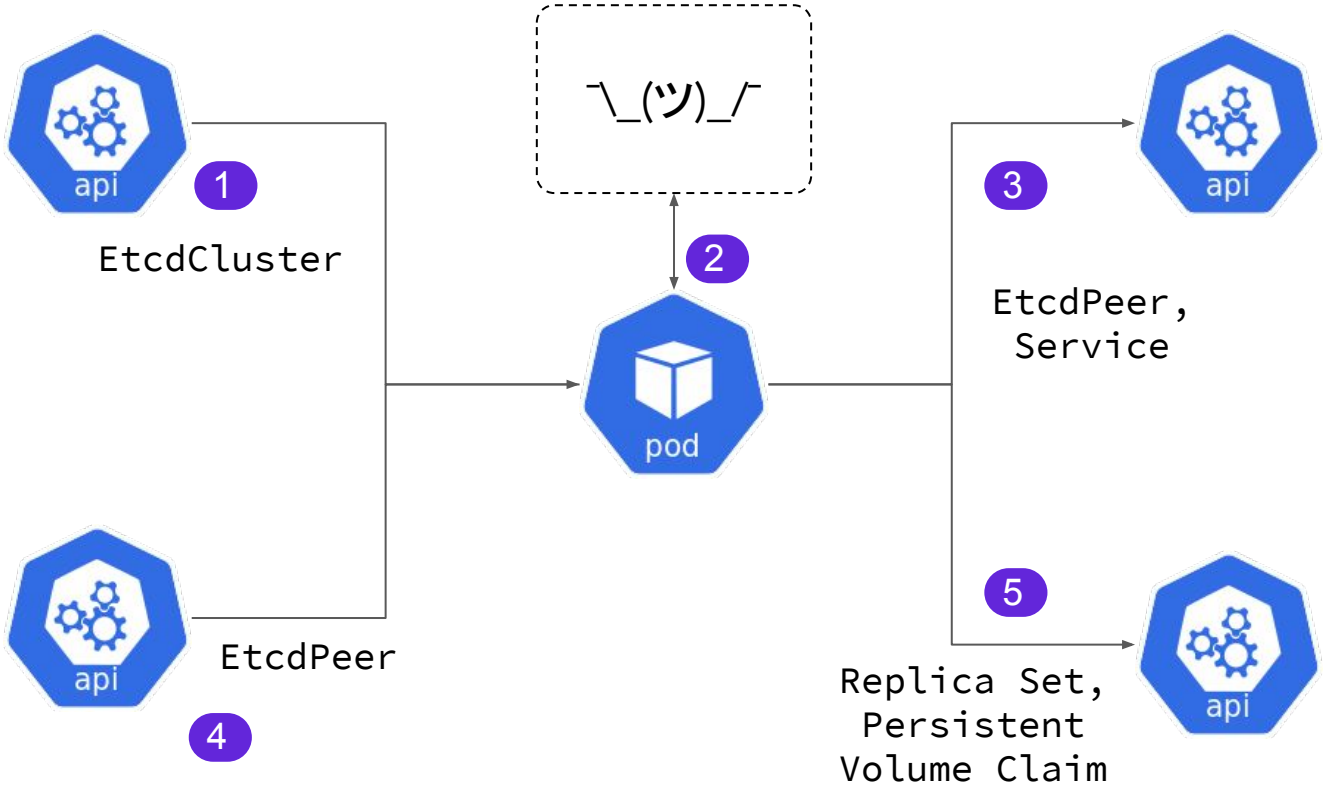
Deploying an etcd Cluster

```
apiVersion: etcd.improbable.io/v1alpha1
kind: EtcdCluster
metadata:
  name: my-etcd
  namespace: foo
spec:
  replicas: 3
```

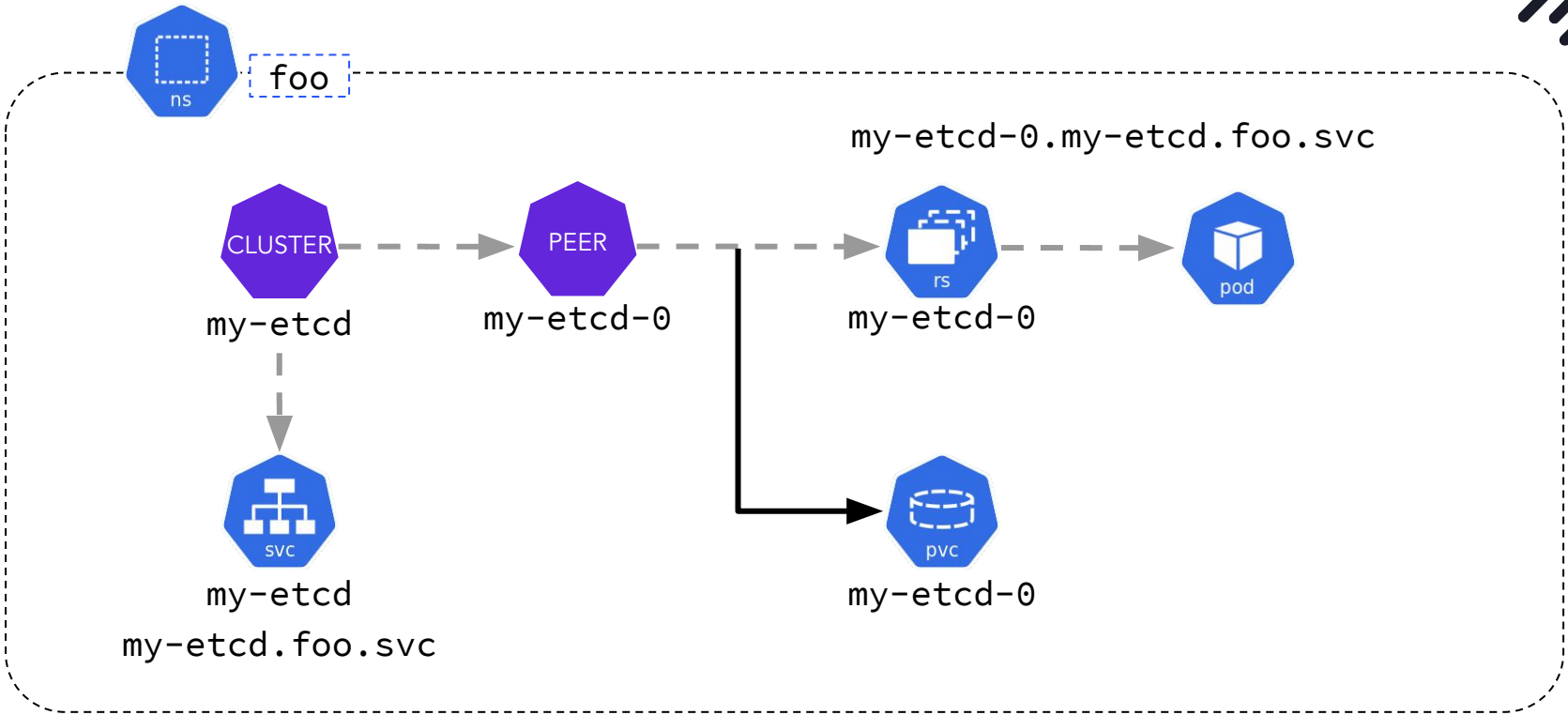


jetstack.io

Custom Resource

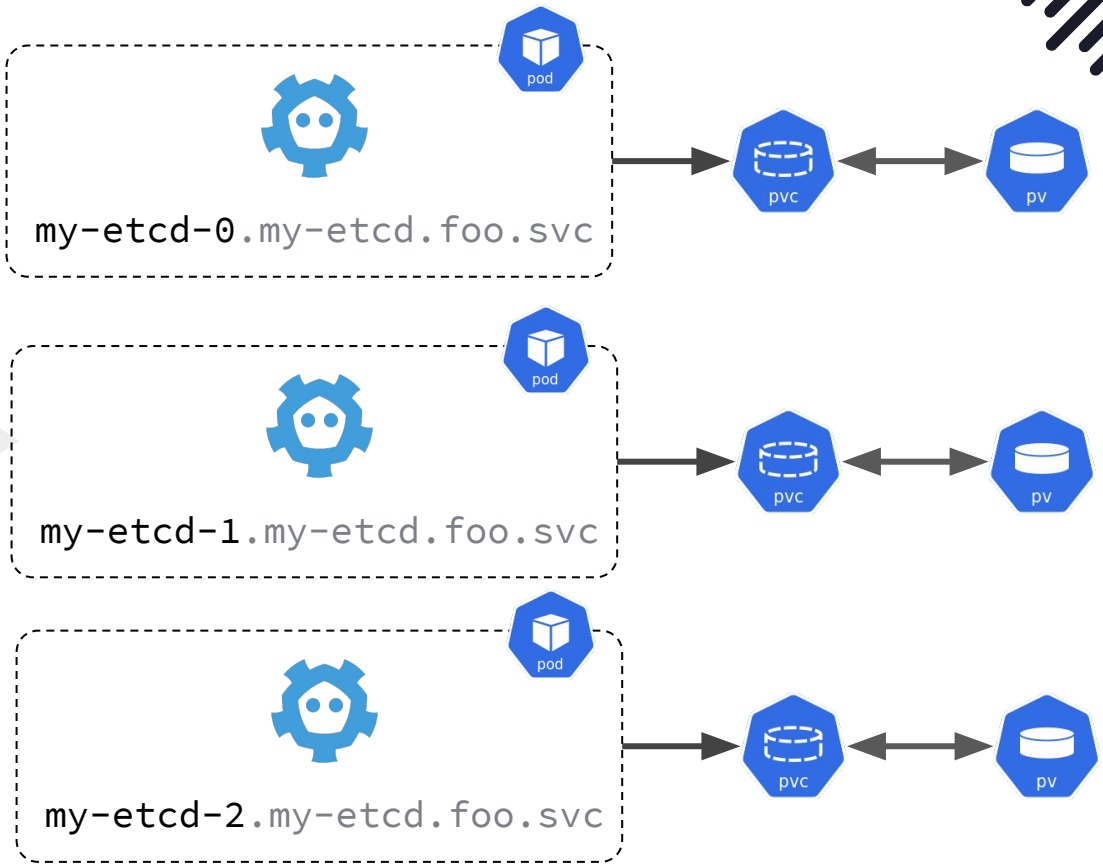


```
apiVersion: etcd.improbable.io/v1alpha1
kind: EtcdPeer
metadata:
  name: my-etcd-0
  Namespace: foo
spec:
  clusterName: my-etcd
  bootstrap:
    initialClusterState: New
    static:
      initialCluster:
      - name: my-etcd-0
        host: my-etcd-0.my-etcd.foo.svc
      - name: my-etcd-1
        host: my-etcd-1.my-etcd.foo.svc
      - name: my-etcd-2
        host: my-etcd-2.my-etcd.foo.svc
```



Custom Resource

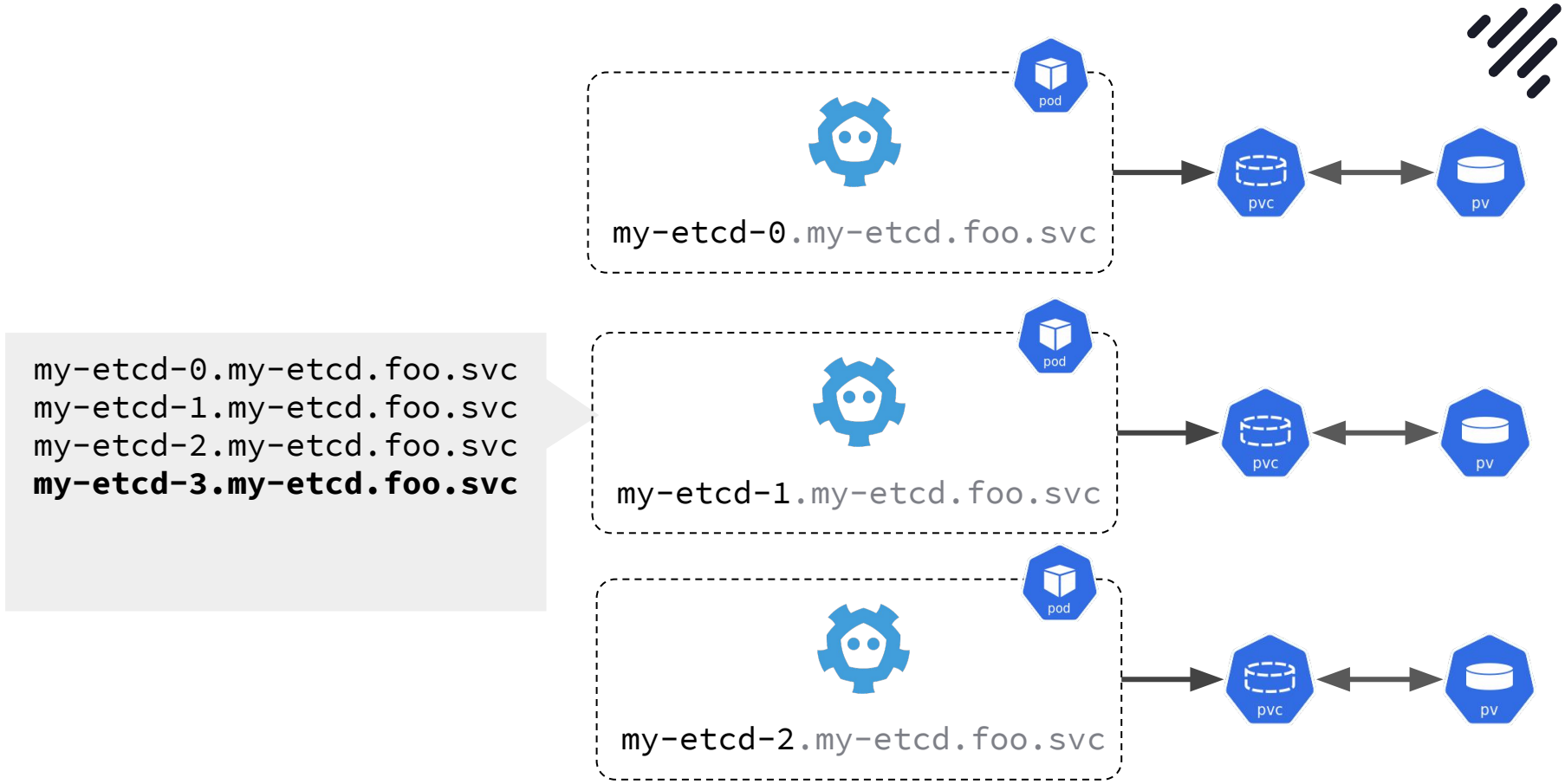
```
my-etcd-0.my-etcd.foo.svc
my-etcd-1.my-etcd.foo.svc
my-etcd-2.my-etcd.foo.svc
```

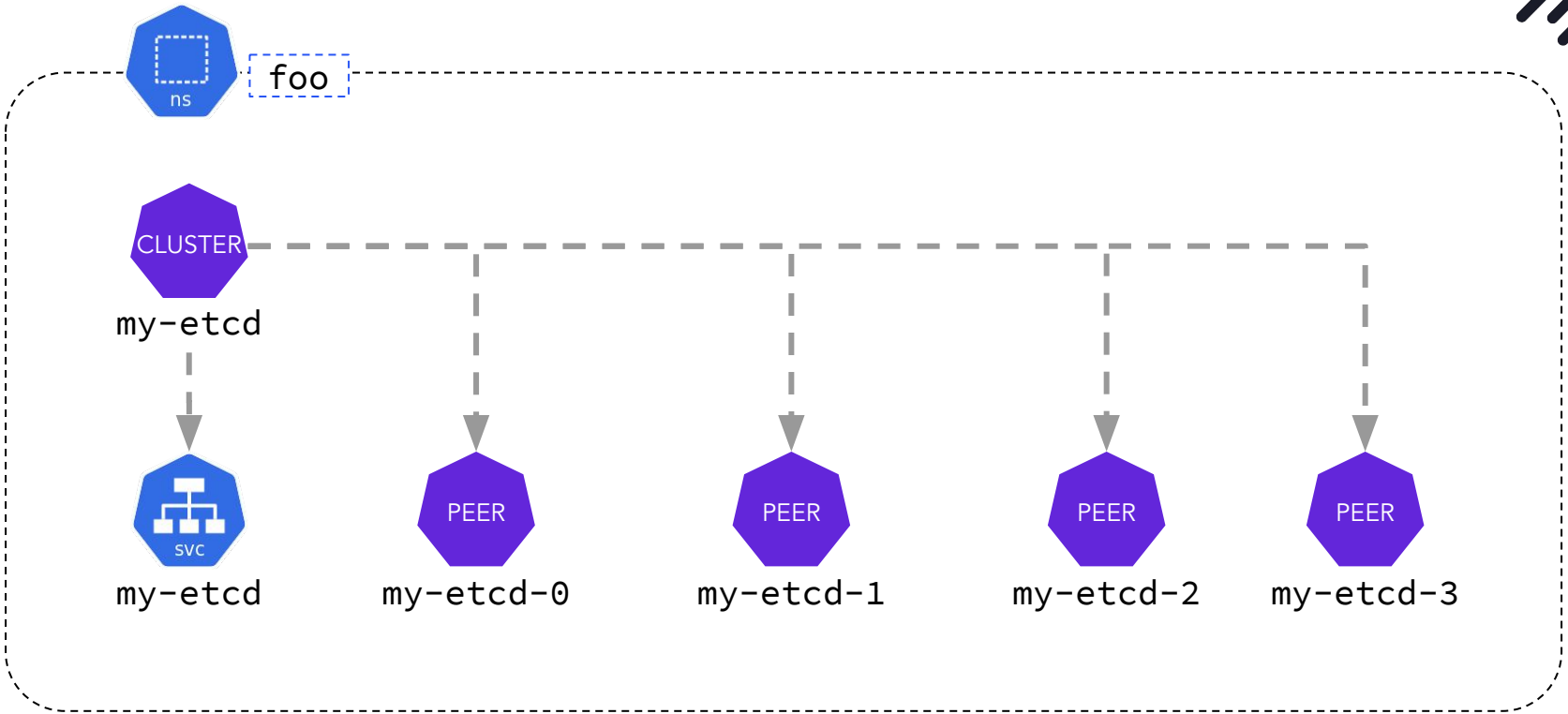


Scale up

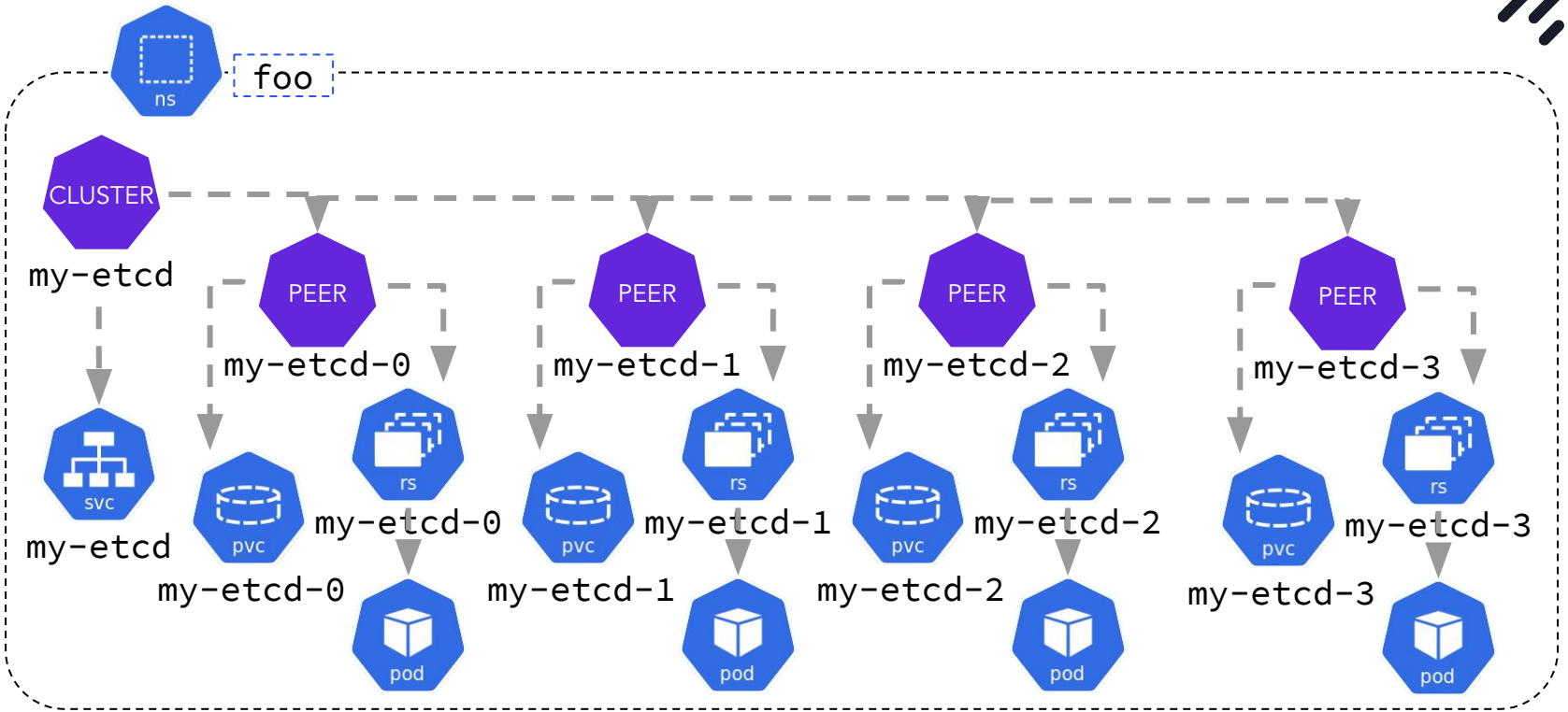
```
apiVersion: etcd.improbable.io/v1alpha1
kind: EtcdCluster
metadata:
  name: my-etcd
spec:
  replicas: 3 5
```

```
$ kubectl scale etcdcluster my-etcd --replicas 5
```




ns
foo
CLUSTER
my-etcd
svc
my-etcd
PEER
my-etcd-0
PEER
my-etcd-1
PEER
my-etcd-2
PEER
my-etcd-3

```
apiVersion: etcd.improbable.io/v1alpha1
kind: EtcdPeer
metadata:
  name: my-etcd-3
  namespace: foo
spec:
  clusterName: my-etcd
  bootstrap:
    initialClusterState: Existing
    static:
      initialCluster:
      - name: my-etcd-0
        host: my-etcd-0.my-etcd.foo.svc
      - name: my-etcd-1
        host: my-etcd-1.my-etcd.foo.svc
      - name: my-etcd-2
        host: my-etcd-2.my-etcd.foo.svc
      - name: my-etcd-3
        host: my-etcd-3.my-etcd.foo.svc
```



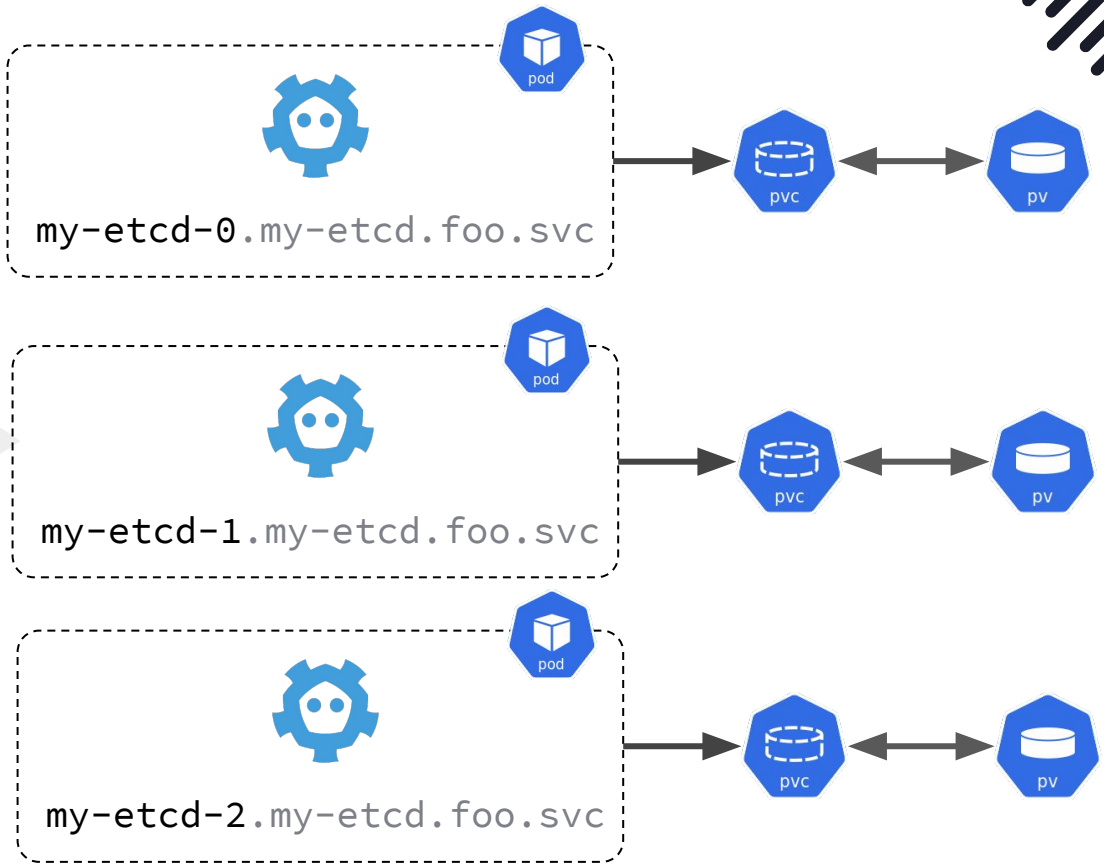
jetstack.io

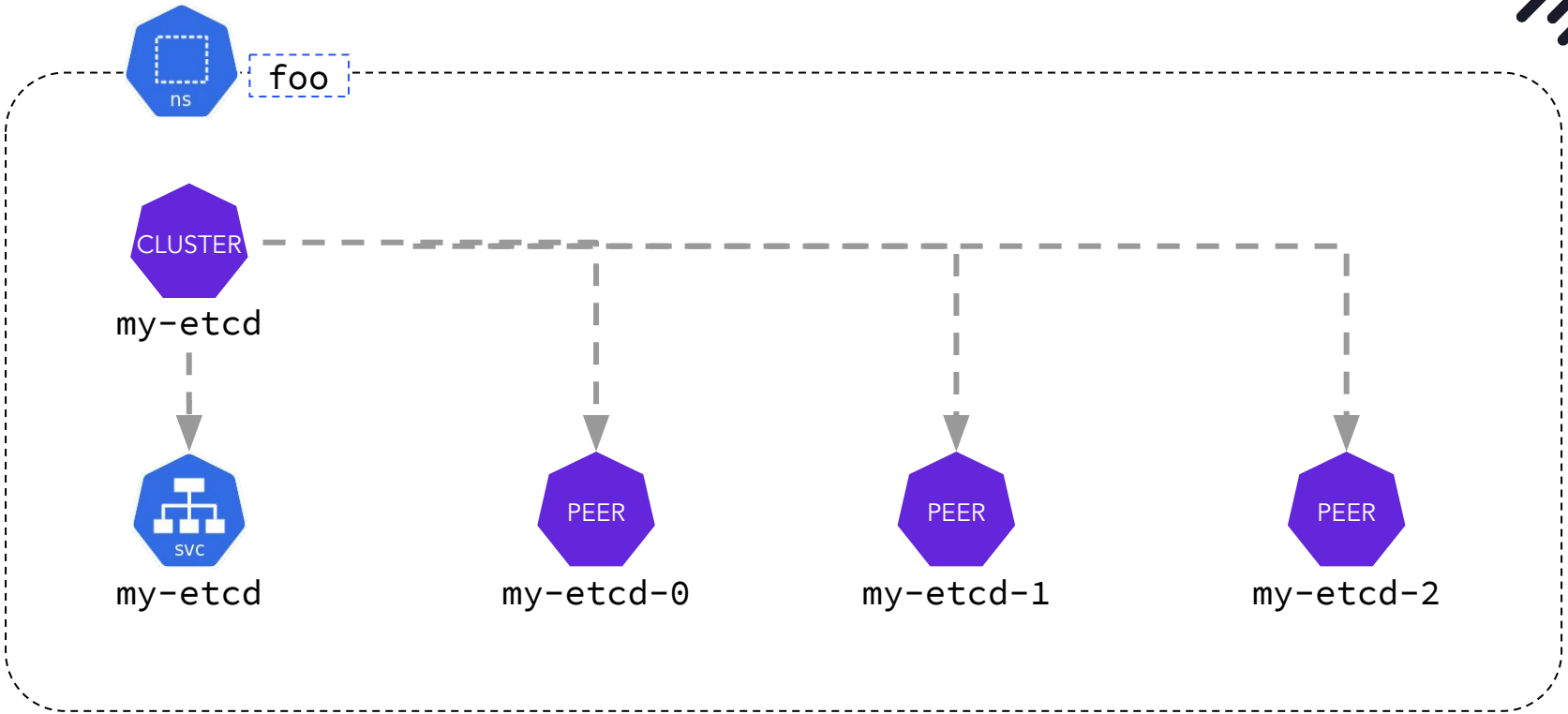
Custom Resource

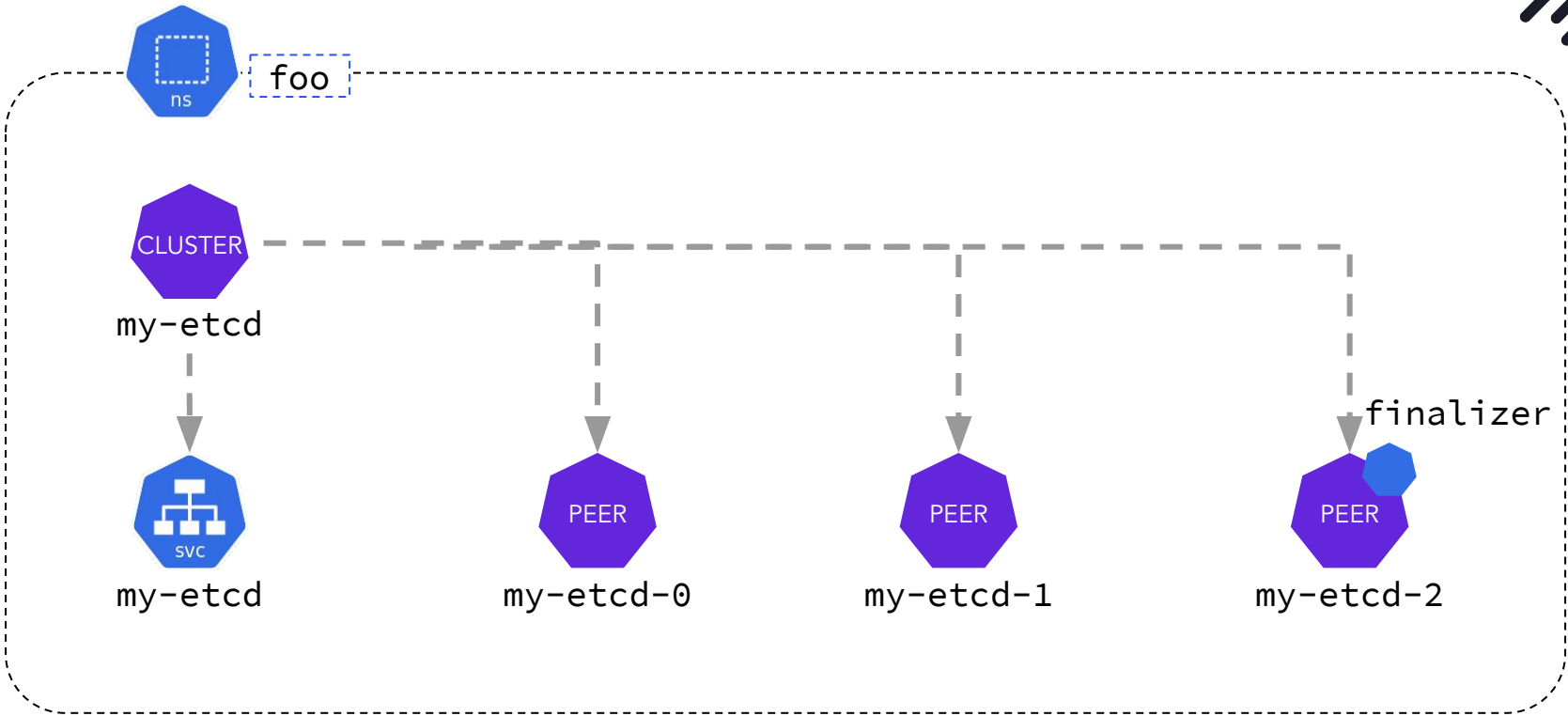
Scale down

```
$ kubectl scale etcdcluster my-etcd --replicas 1
```

```
my-etcd-0.my-etcd.foo.svc
my-etcd-1.my-etcd.foo.svc
~~my-etcd-2.my-etcd.foo.svc~~
```

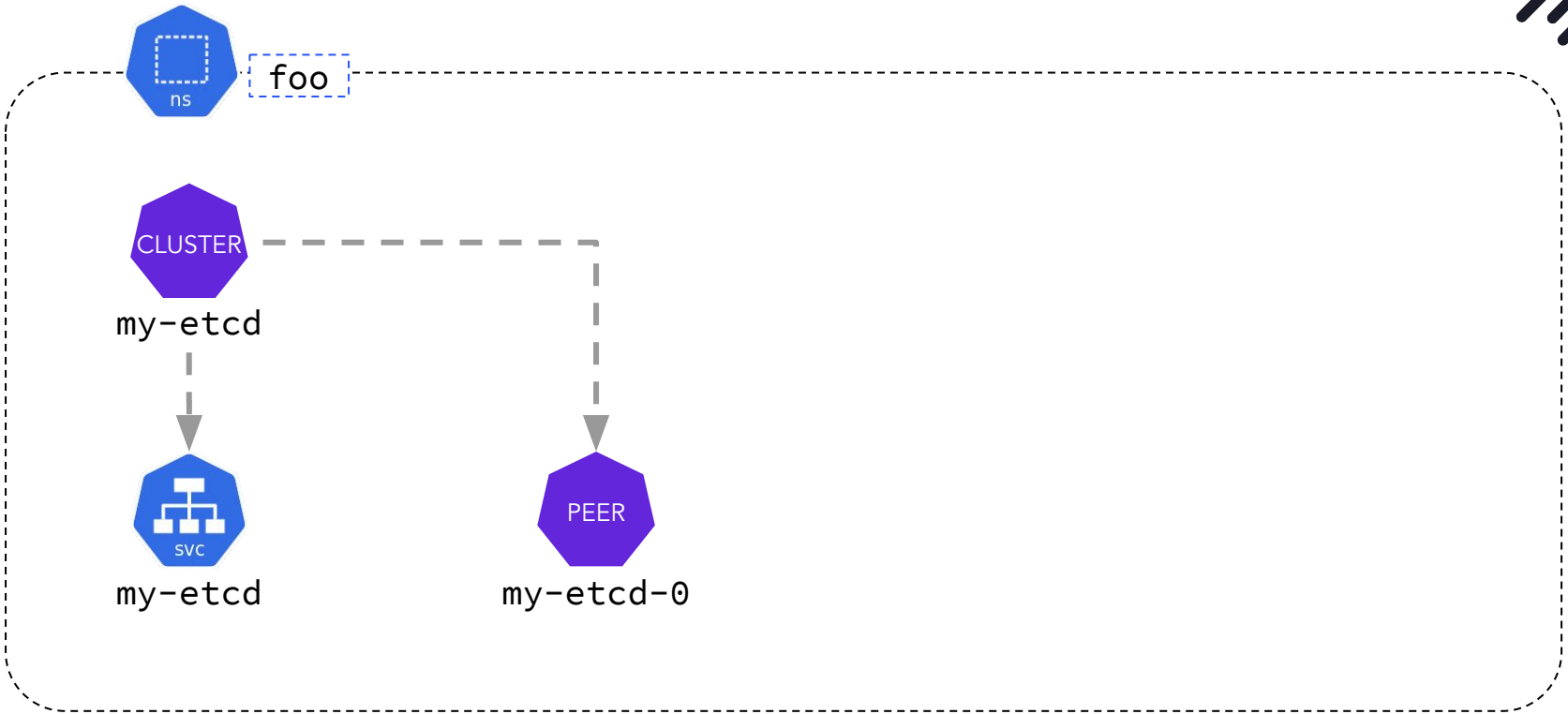



ns
foo
CLUSTER
my-etcd
svc
my-etcd
PEER
my-etcd-0
PEER
my-etcd-1
PEER
my-etcd-2



jetstack.io

Custom Resource



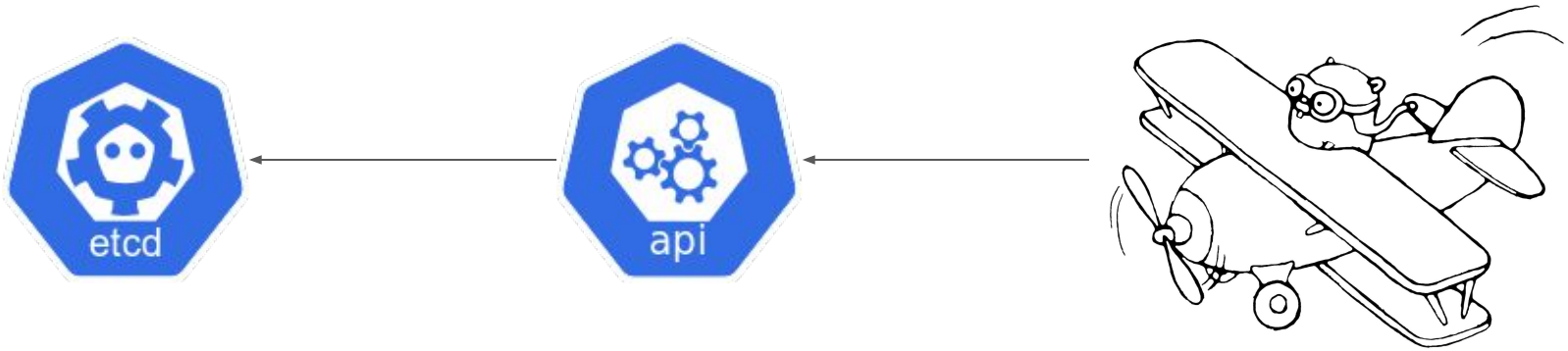
jetstack.io

Custom Resource

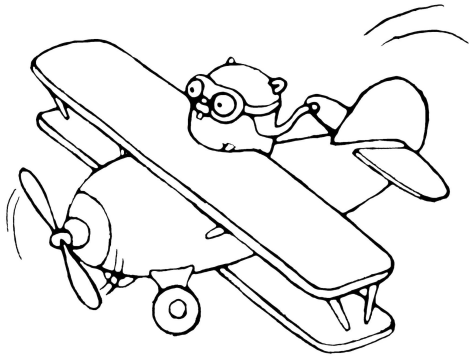
Other features

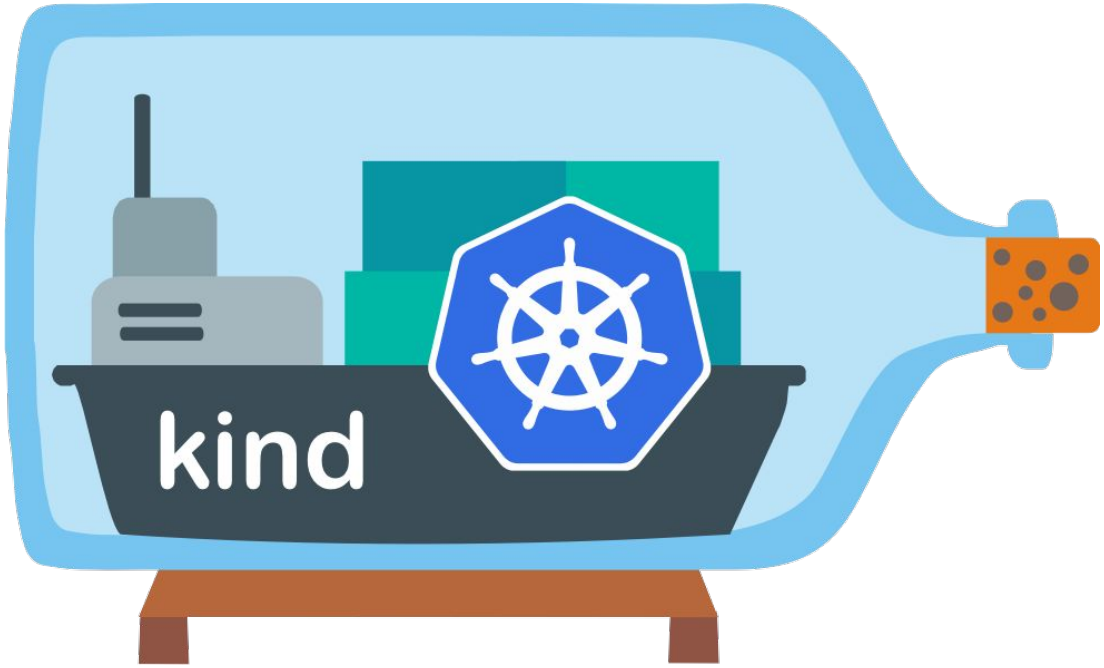
- Version upgrade
- Backup
- Restore

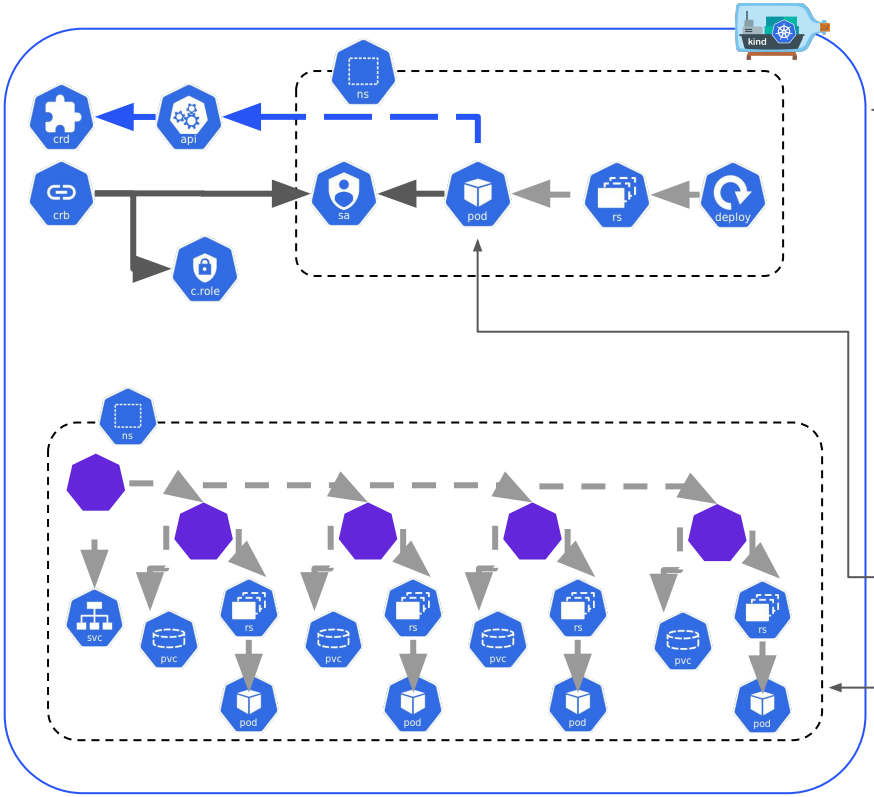
Testing



mocked!



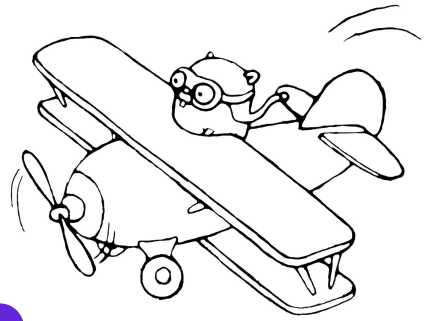

kind




```
$ kind create cluster
```

1

2



```
$ docker build .
```

3 Load images & deploy operator

4 Deploy an `EtcdCluster` and assert on behaviour

jetstack.io

Custom Resource

What did we learn?

Operators provide value for applications with complex run-books.

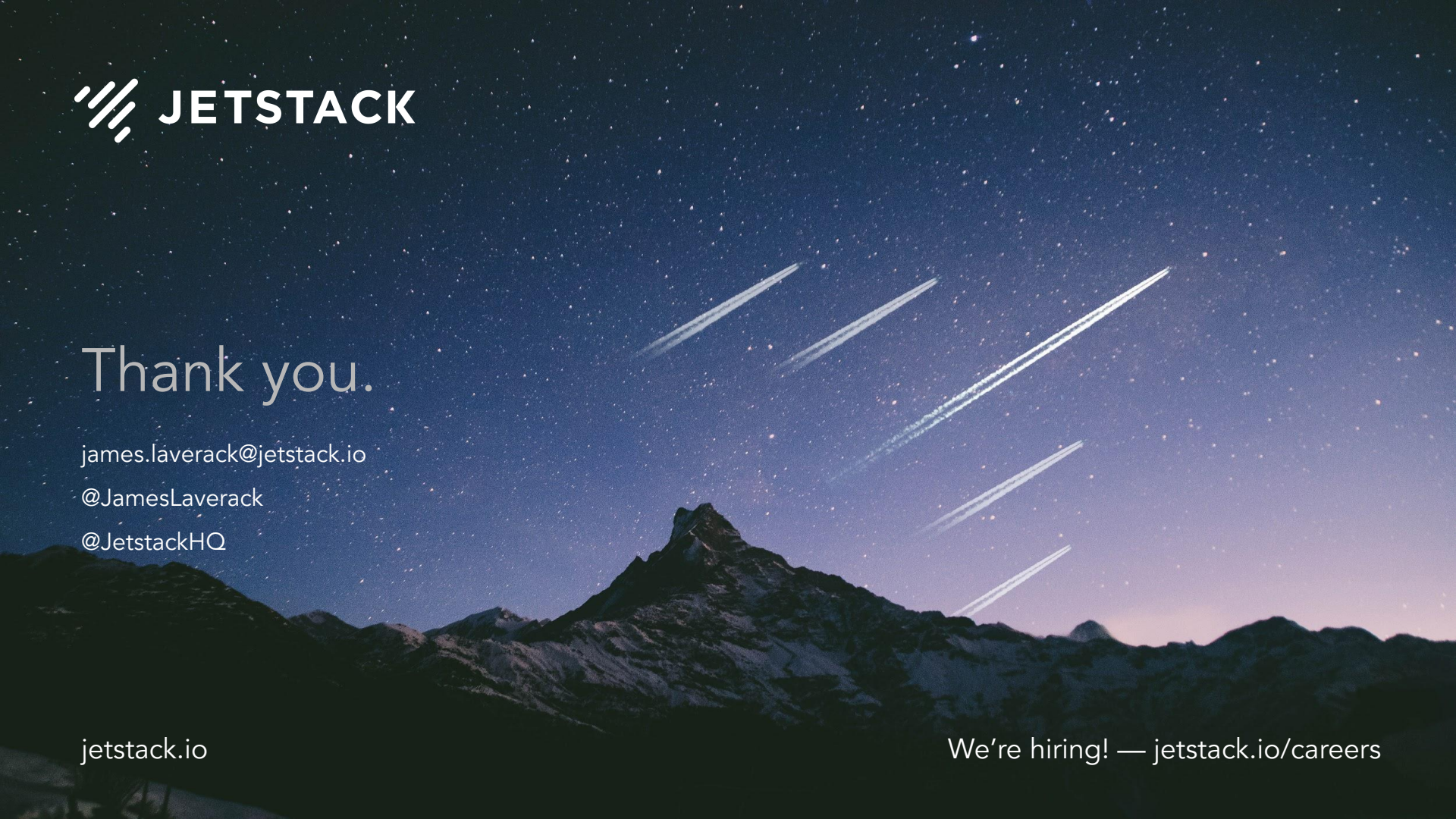
Operators expose application specific APIs that work with existing tooling.

You can build an Operator in any stack, but Go was right for us.

You can end-to-end test Operators on your laptop with kind.

github.com/improbable-eng/etcd-cluster-operator

Questions?


JETSTACK
Thank you.
james.laverack@jetstack.io
@JamesLaverack
@JetstackHQ
jetstack.io
We're hiring! — jetstack.io/careers