Gyuho Lee leegyuho@amazon.com

Sam Batschelet sbatsche@redhat.com

KubeCon | CloudNativeCon

North America 2018

etcd, What's next for cluster management?

Agenda

Challenges in etcd cluster management

Can we do better? Yes!

etcd is distributed

- Distributed (typically 3 or 5 nodes)
- **Consistent** + **Partition Tolerant** + (Highly) **Available**, *in CAP theorem*
- Strong(Sequential) Consistency (NOT eventual consistency)
- Consensus over Raft

Challenge #1

etcd Membership Reconfiguration

Disruptive membership reconfiguration

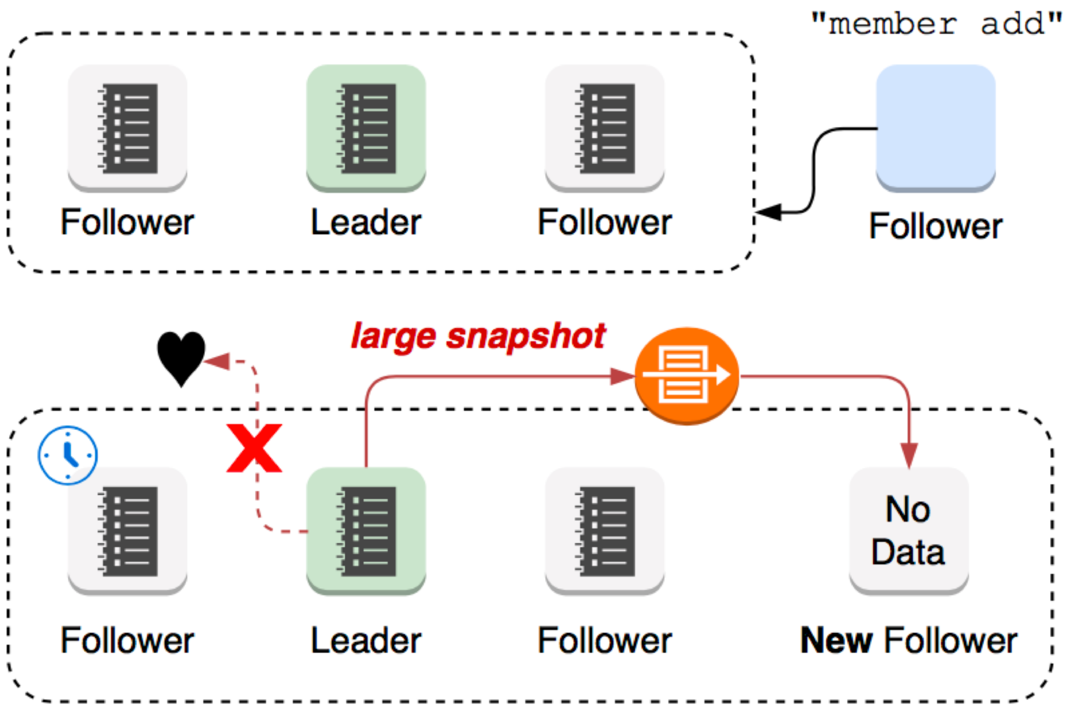


Figure 1. A new member joins with empty data, requesting more data from leader. Then leader becomes overloaded sending large snapshots. Which may block heartbeat sends. Then follower may election-timeout and start a new election.

Cluster with new member is more vulnerable to leadership election

Network Partition Will Happen!

Quorum size = (cluster size / 2) + 1

Leadership election WILL NOT happen!

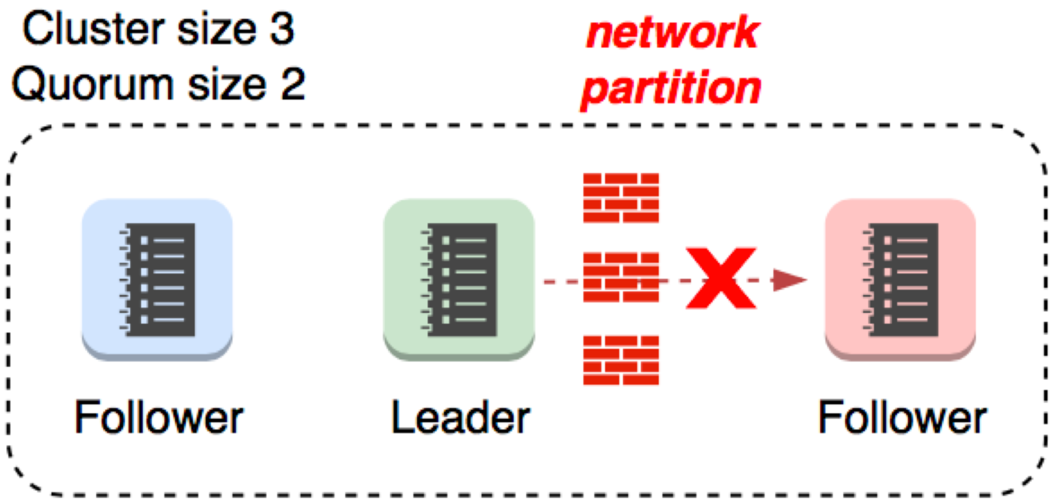


Figure 2. In 3-node cluster, a **follower gets isolated**. In this case, leader only requires 1 active follower (total 2 active nodes including leader). **No leadership election happens even with the network partition**, since the leader **still has 2 active nodes** and the size of quorum 2 is the minimum number of nodes required for cluster operation.

Network Partition Will Happen!

Cluster size 3, Quorum size 2

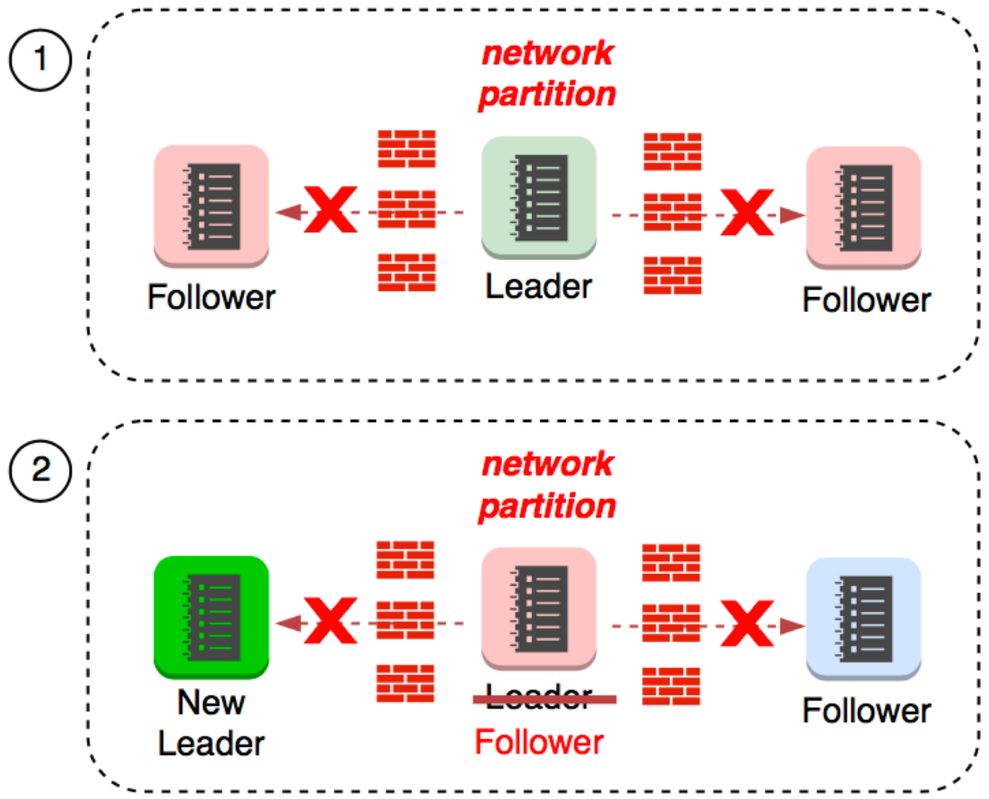


Figure 3. In 3-node cluster, the **leader node gets isolated**. In this case, leader requires at least 1 active follower (total 2 active nodes including leader). Leader had no active follower within its partition (lost quorum). Then, leader election will happen to elect a new leader.

What if leader gets isolated?

Leadership election WILL happen!

Network Partition + Membership Reconfiguration

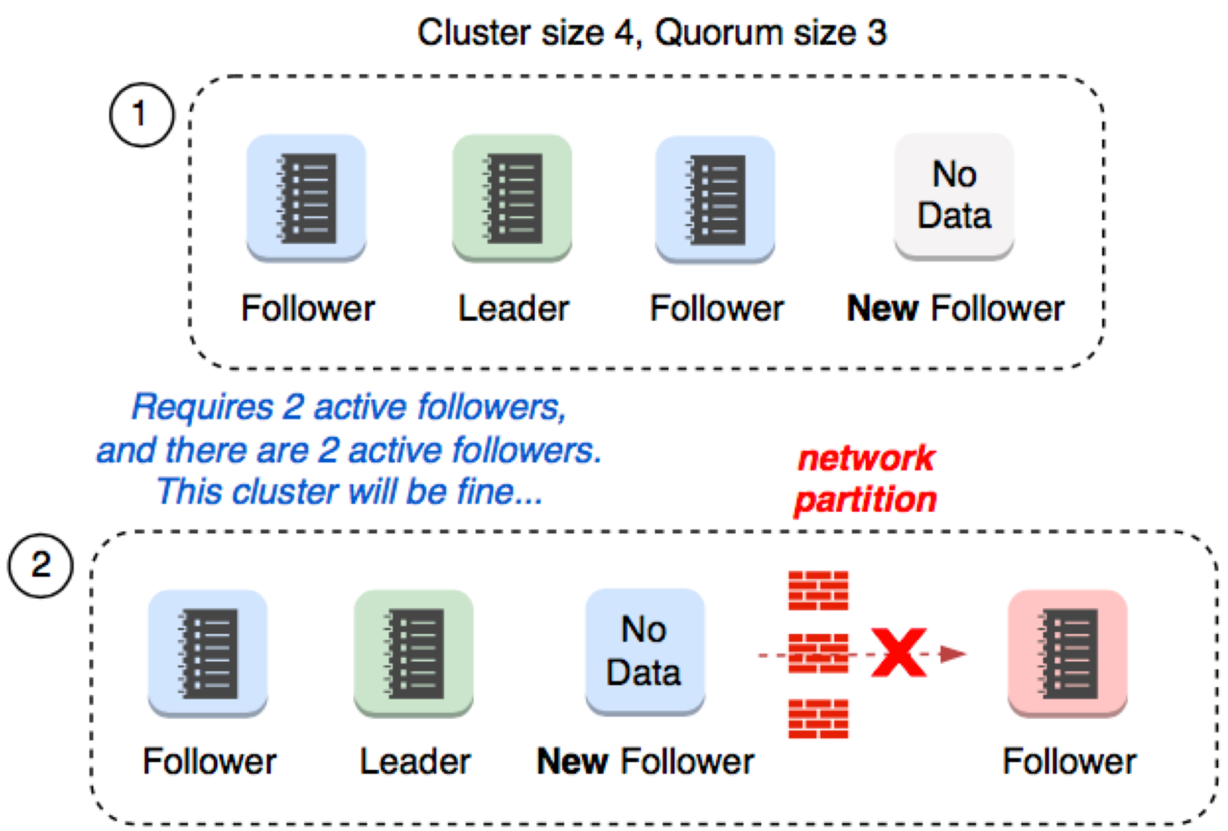

Cluster size 4, Quorum size 3
1
Follower
Leader
Follower
No Data
New Follower
Requires 2 active followers, and there are 2 active followers. This cluster will be fine...
network partition
2
Follower
Leader
No Data
New Follower
Follower

Figure 4. What if a new node has been added, and then network partition happens? When a new node joins 3-node cluster, quorum size increases to 3. And if the **new node happens to be in the same partition as leader's**, leader **still maintains the active quorum**, so cluster will continue to work under the network partition.

This is OK

Network Partition + Membership Reconfiguration

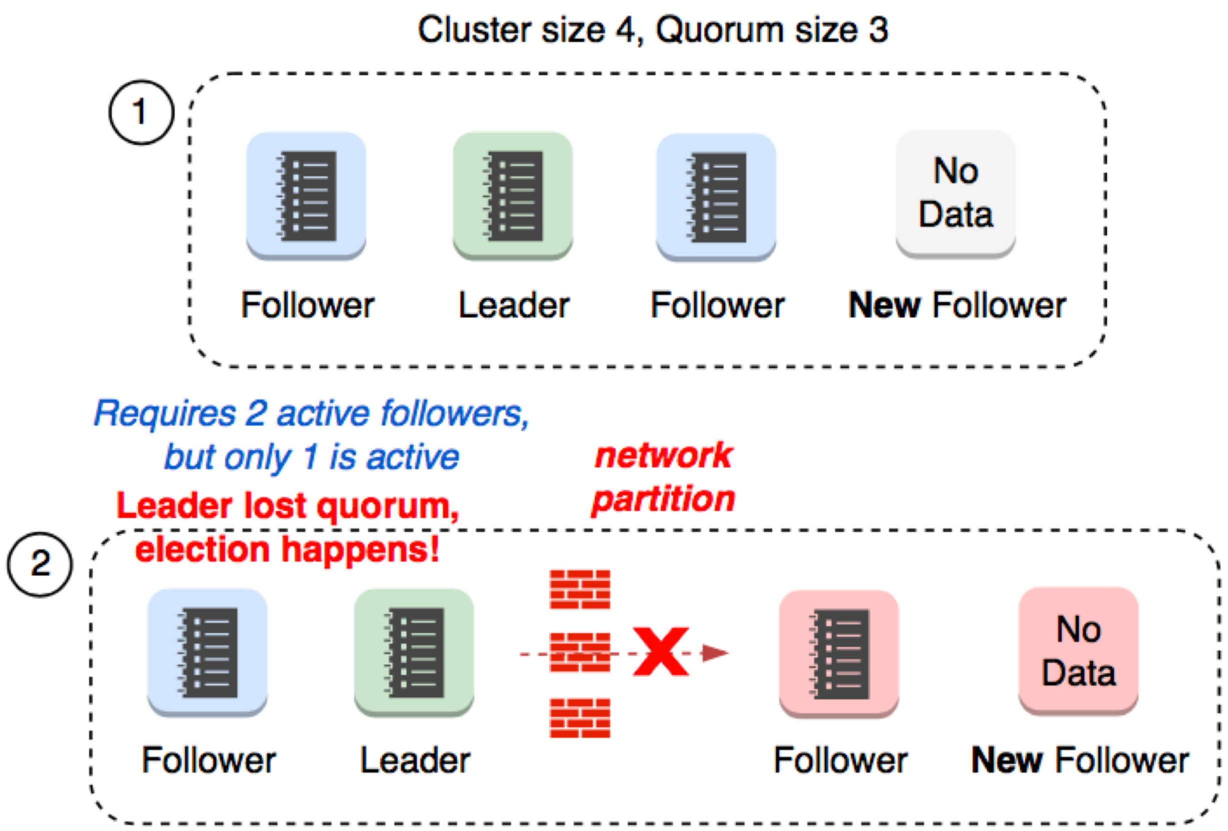

Cluster size 4, Quorum size 3
1
No Data
Follower
Leader
Follower
New Follower
Requires 2 active followers, but only 1 is active
Leader lost quorum, election happens!
network partition
2
No Data
Follower
Leader
Follower
New Follower

Figure 5. What if a new node has been added, and then network partition happens? When a new node joins 3-node cluster, quorum size increases to 3. In this case, **leader requires at least 2 active followers**, but due to network partition, there are **only 1 active follower**. Then, leader reverts back to follower.

This is NOT OK

Network Partition + Membership Reconfiguration

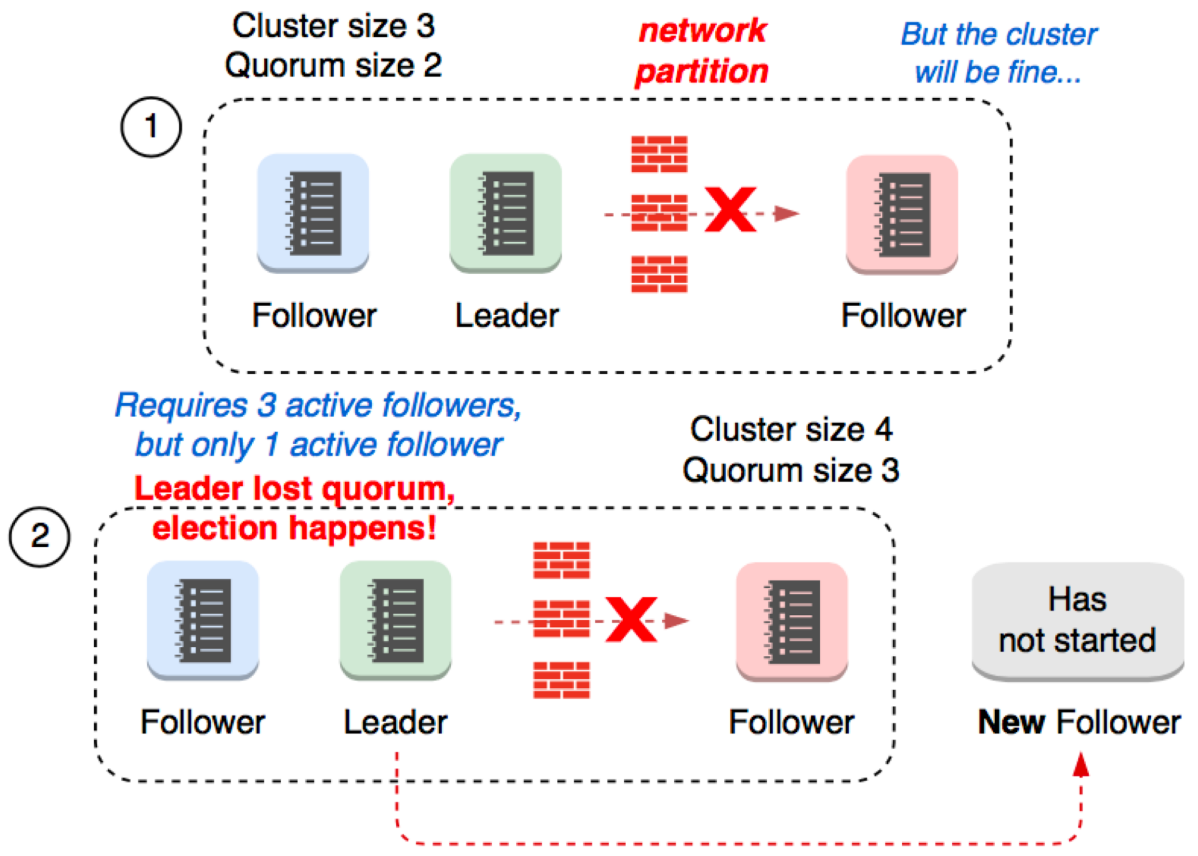


Figure 6. What if network partition has happened, and then a new node is added? When a new node is added to 3-node cluster (now 4-node cluster), quorum size changes from 2 to 3. **Since the new node is not started yet, the cluster now has only 2 active nodes and loses quorum, triggering a leadership election.**

This is NOT OK

Disruptive membership reconfiguration

Quorum size = (cluster size / 2) + 1

Member add operation is 2-step process

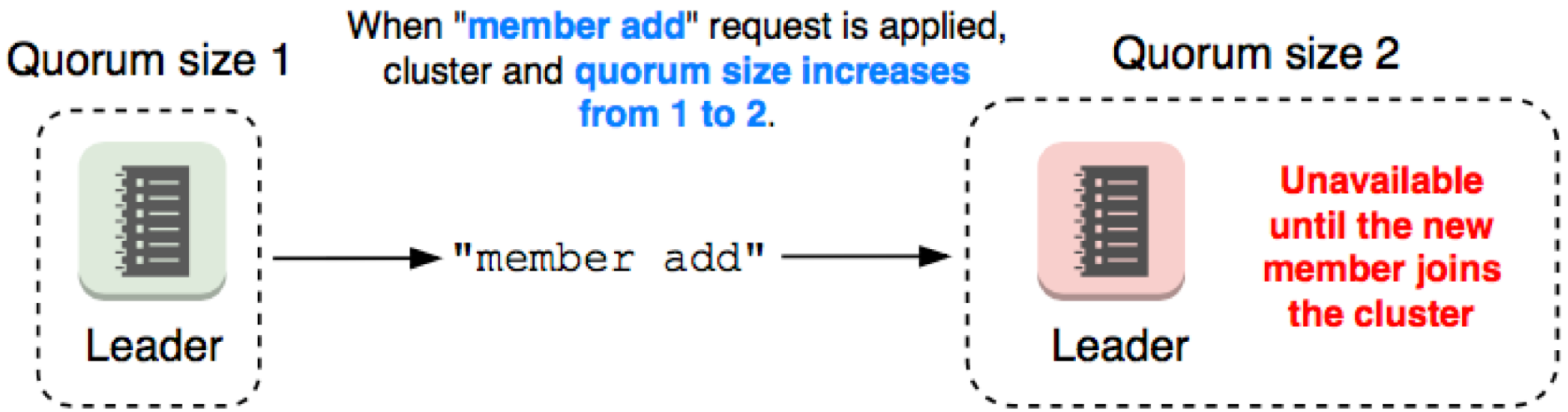

Quorum size 1
When "member add" request is applied, cluster and quorum size increases from 1 to 2.
Quorum size 2
Leader
"member add"
Leader
Unavailable until the new member joins the cluster

Figure 7. "`member add`" command to a single node cluster **increases the quorum size to 2,** causing an **immediate leader election**, because from previous leader's viewpoint, quorum is not active.

Disruptive membership reconfiguration

Quorum size = (cluster size / 2) + 1

Member add operation can increase the quorum size

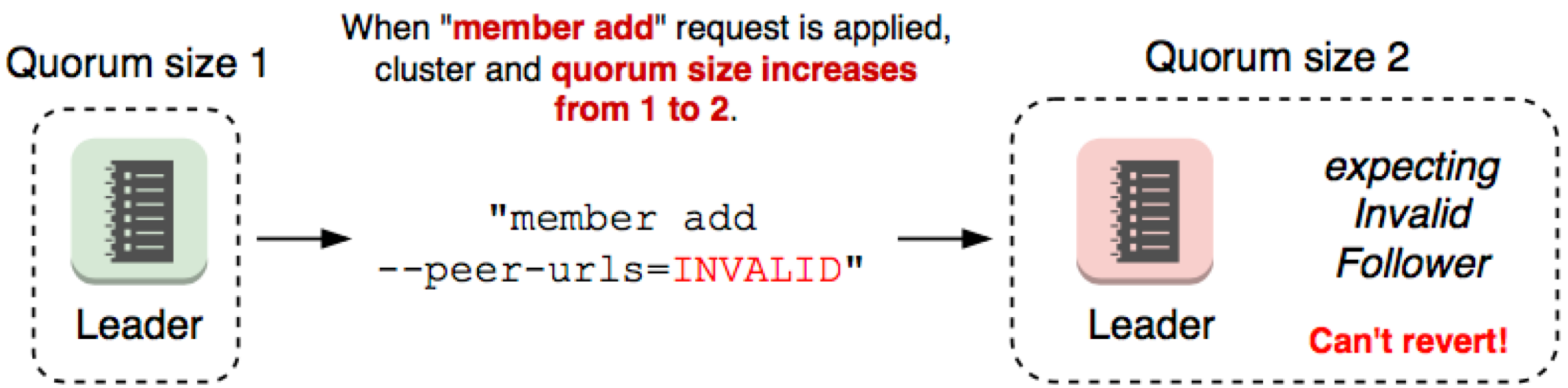

Quorum size 1
Leader
When "member add" request is applied, cluster and quorum size increases from 1 to 2.
"member add --peer-urls=INVALID"
Quorum size 2
Leader
expecting Invalid Follower
Can't revert!

Figure 8. Imagine "member add" command was **misconfigured with a wrong URL**. The request is still applied to the 1-node cluster. And quorum size becomes 2. Then, **leader loses quorum**. Now, the whole cluster is inoperable.

Disruptive membership reconfiguration

Quorum size = (cluster size / 2) + 1

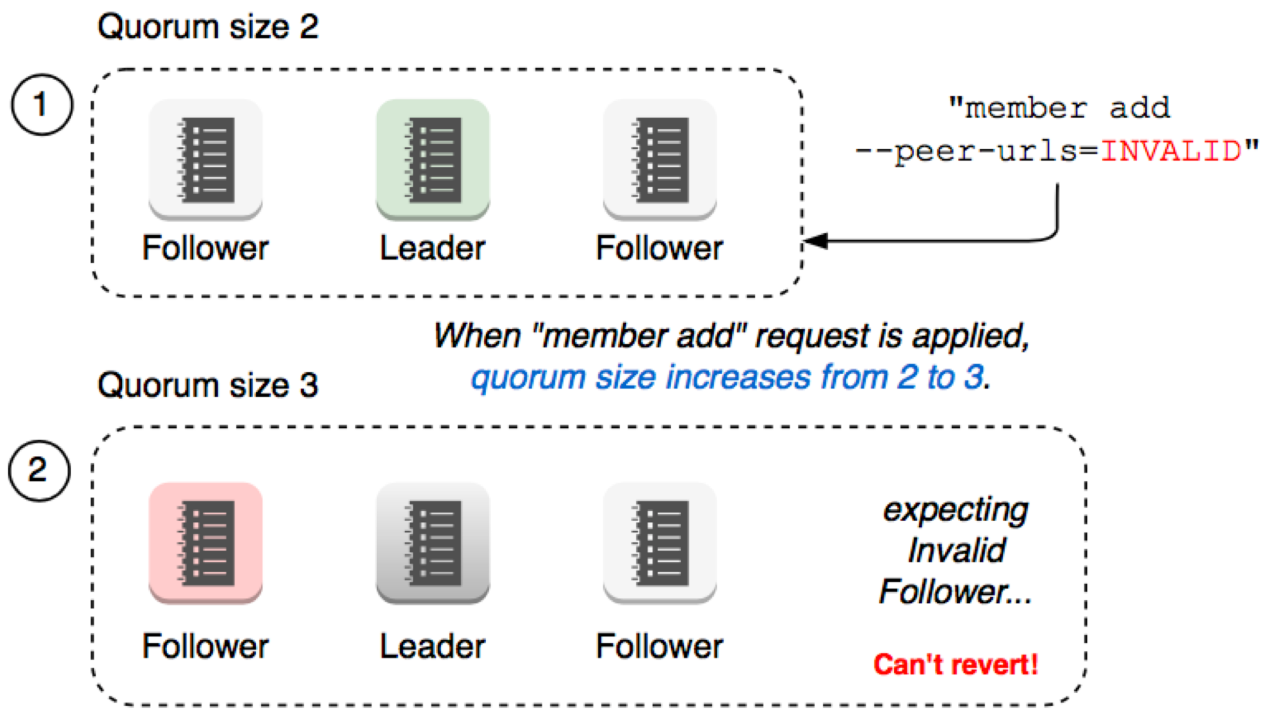


Figure 9. When a wrong member entry is applied to 3-node cluster, quorum size becomes 3. Which requires 3 votes (or 3 active nodes) to commit new entries. With one node being misconfigured but still counted in quorum, even one node failure makes the whole cluster unavailable.

Introducing etcd Raft Learner

Raft Learner (etcd v3.4, 2019)

Learner joins as a non-voting member, does not count in quorum

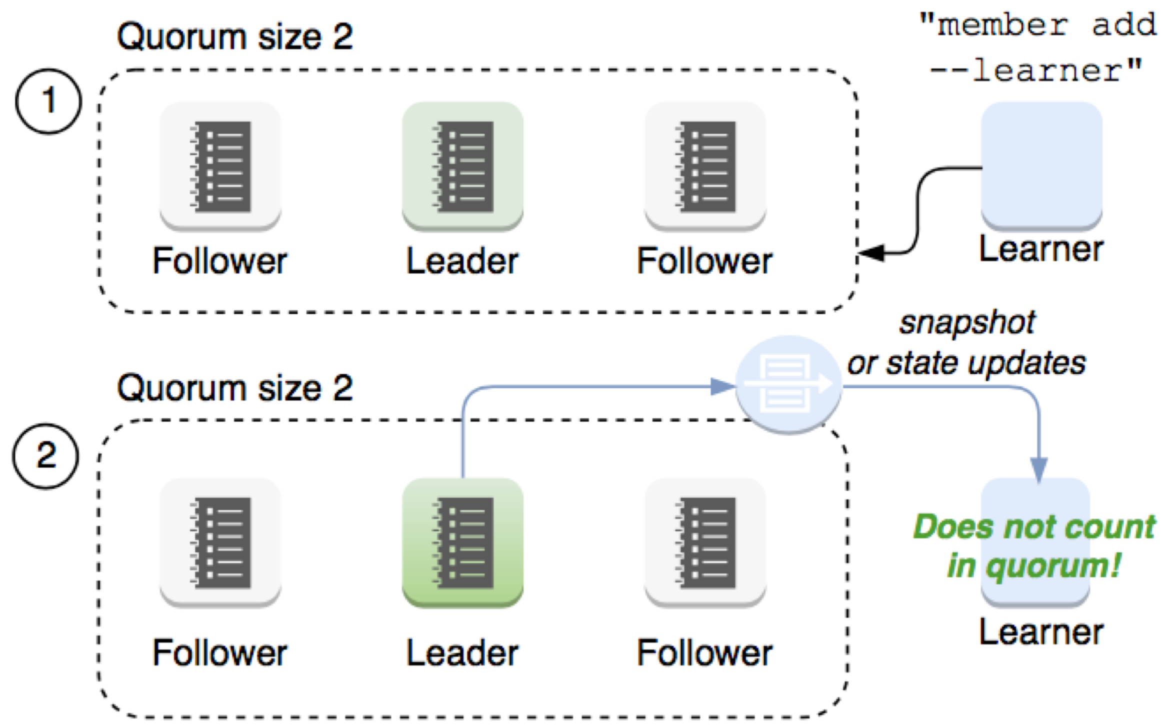


Figure 10. Add a learner node as a non-voting member. Wait until learner node catches up to leader's logs. Until then, learner node neither votes nor counts towards quorum.

Raft Learner (etcd v3.4, 2019)

Learner joins as a non-voting member, does not count in quorum

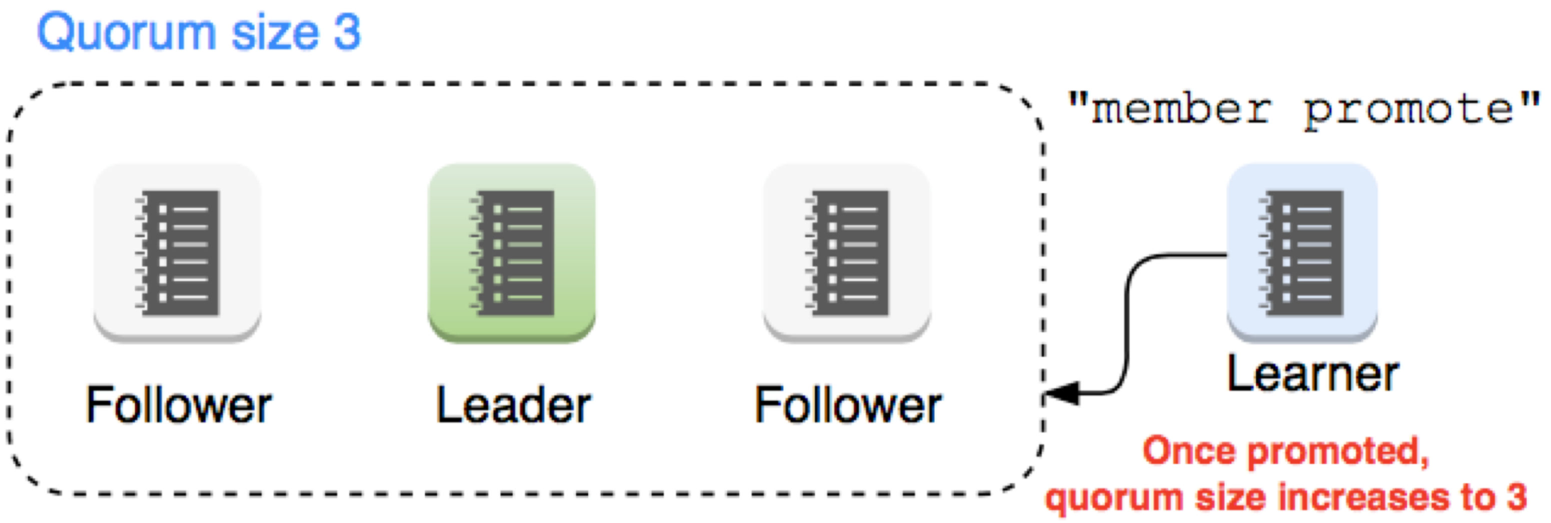


Figure 11. Once learner node has caught up to leader's log, "member promote" API can promote it to a normal voting node that counts towards quorum. In this case, it will increase the size of quorum to 3.

Raft Learner (etcd v3.4, 2019)

etcd server will validate "promote" request

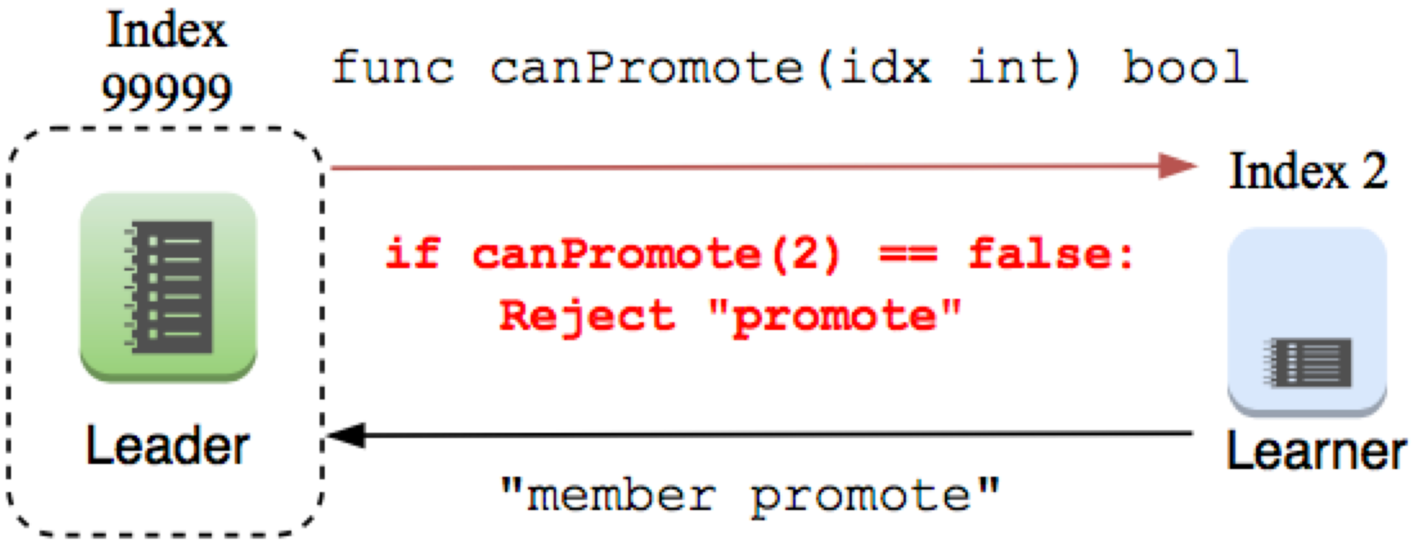


Figure 12. etcd v3.4 learner can be promoted only when it satisfies the safety requirement. Otherwise, promote request will be rejected.

Raft Learner (etcd v3.4, 2019)

Learner rejects all reads and writes (for simplest implementation possible)

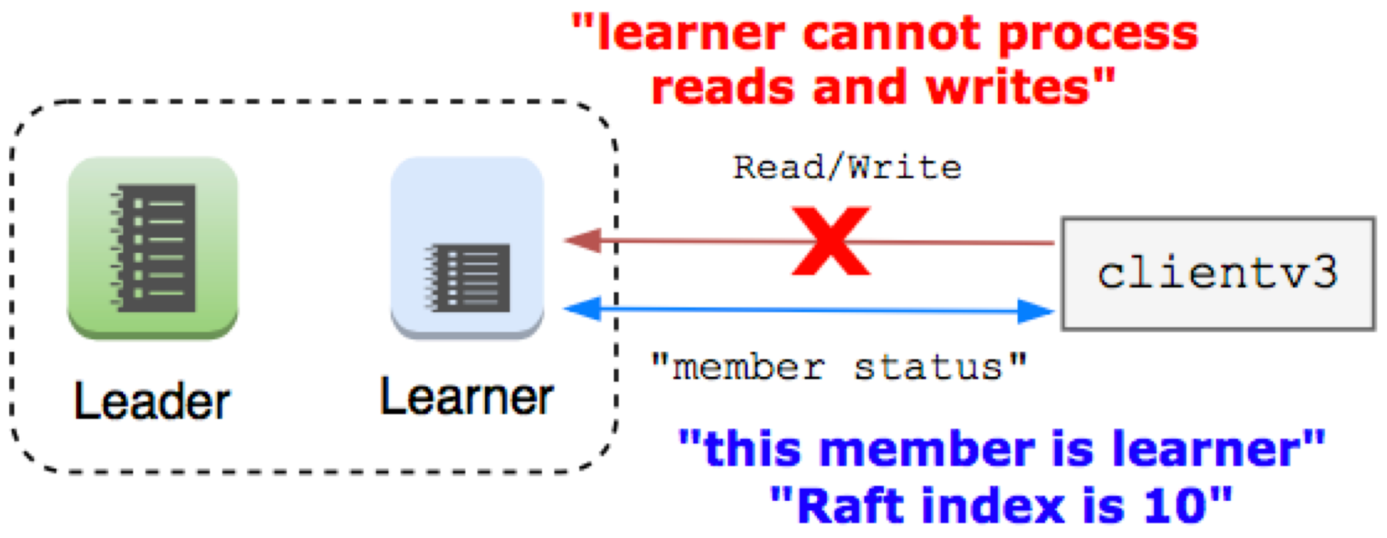


Figure 13. etcd v3.4 learner serves as a standby node only. Learner node rejects client reads and writes but allows status checks.

Challenge #2

Persistent and durable etcd clusters

with etcd operator

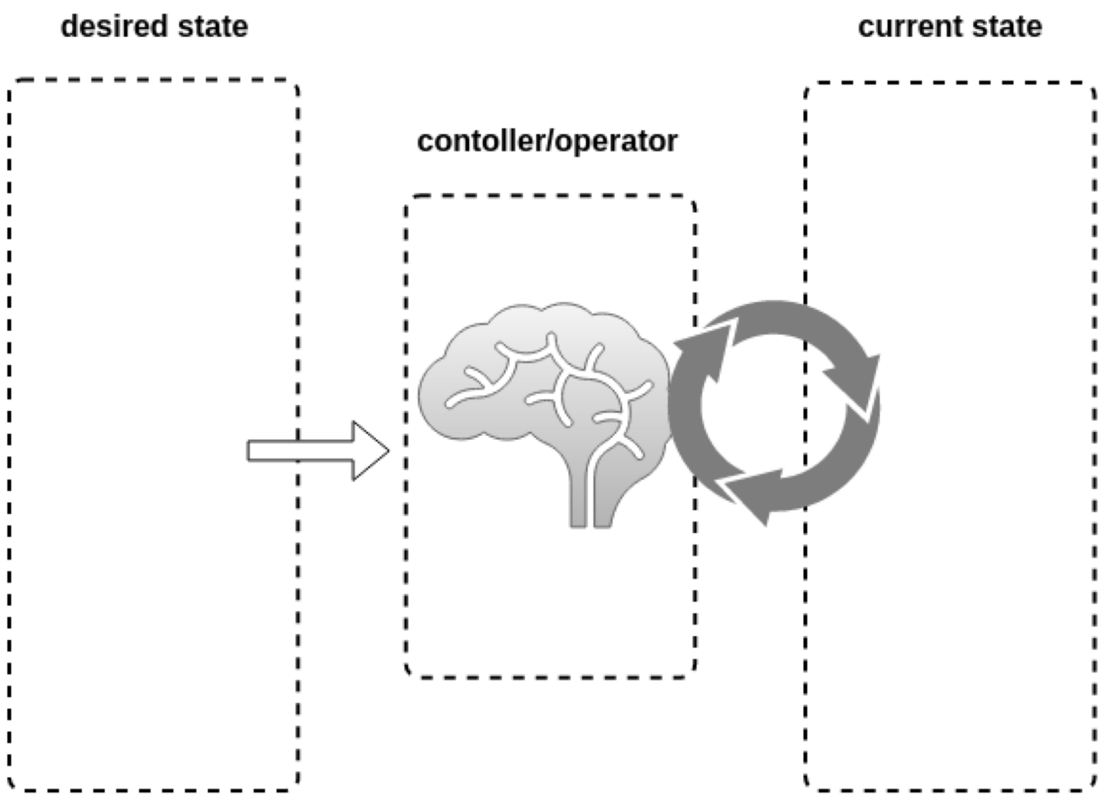
etcd operator

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

CoreOS

etcd operator

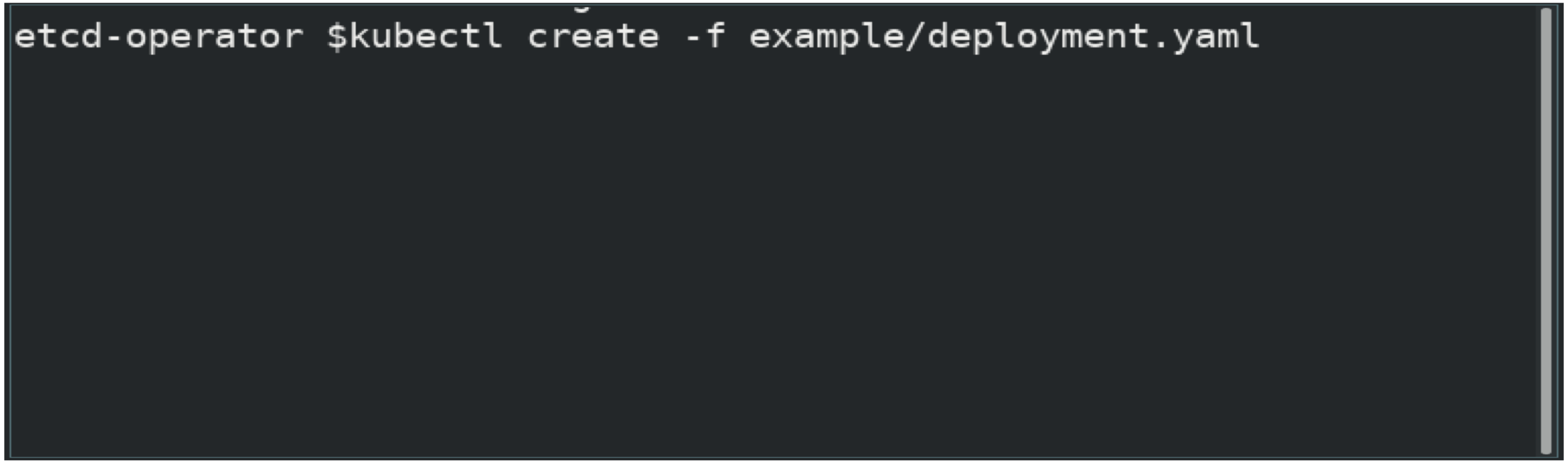
Operator Components



- Custom Resource Definition (CRD)
- Custom Controller

etcd operator

Install etcd operator

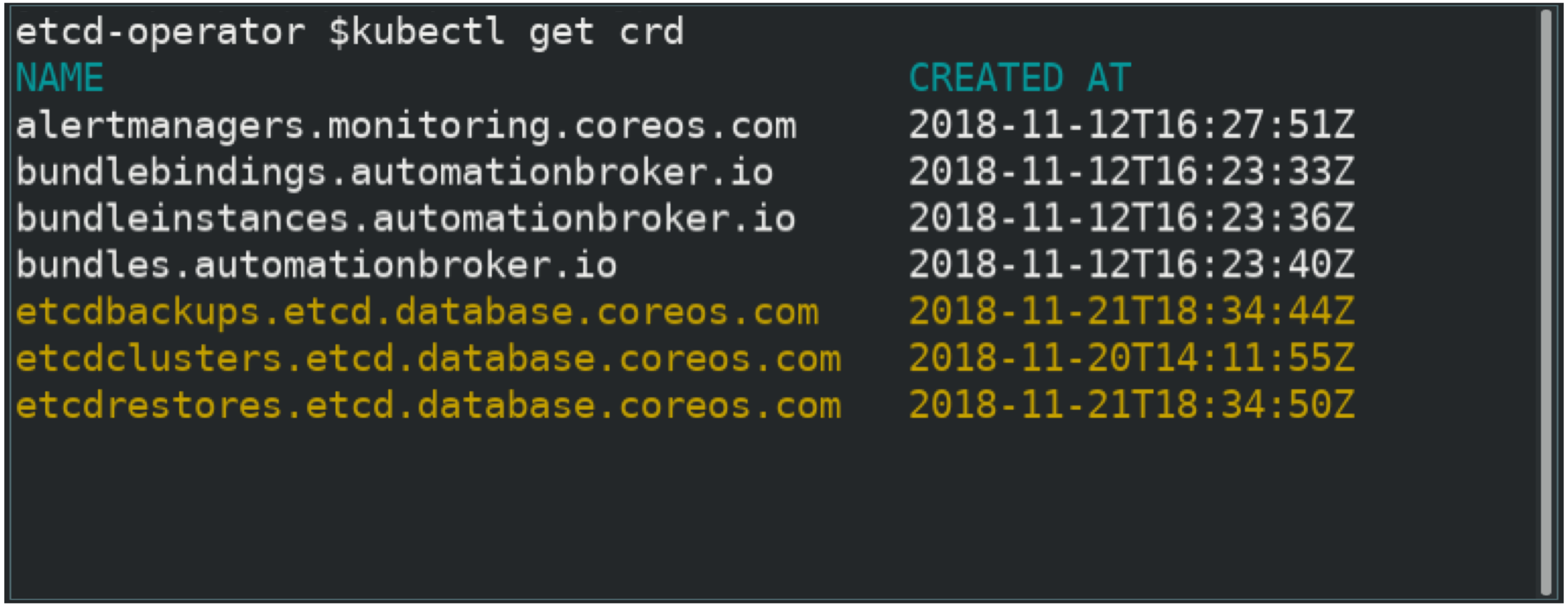


```
etcd-operator $kubectl create -f example/deployment.yaml
```

etcd operator

KubeCon | CloudNativeCon North America 2018

Deployment installs three CRDs

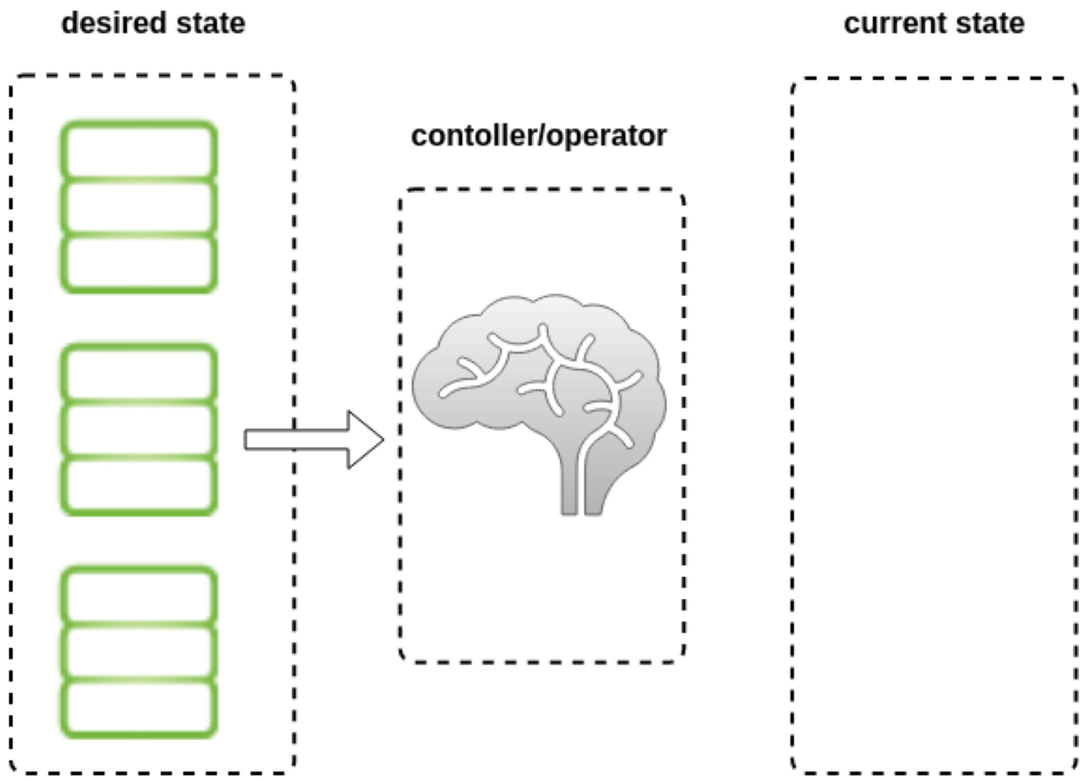


```
etcd-operator $kubectl get crd
NAME                                      CREATED AT
alertmanagers.monitoring.coreos.com       2018-11-12T16:27:51Z
bundlebindings.automationbroker.io        2018-11-12T16:23:33Z
bundleinstances.automationbroker.io       2018-11-12T16:23:36Z
bundles.automationbroker.io               2018-11-12T16:23:40Z
etcdbackups.etcd.database.coreos.com      2018-11-21T18:34:44Z
etcdclusters.etcd.database.coreos.com     2018-11-20T14:11:55Z
etcdrestores.etcd.database.coreos.com     2018-11-21T18:34:50Z
```

etcd operator

KubeCon | CloudNativeCon
North America 2018

Define desired state

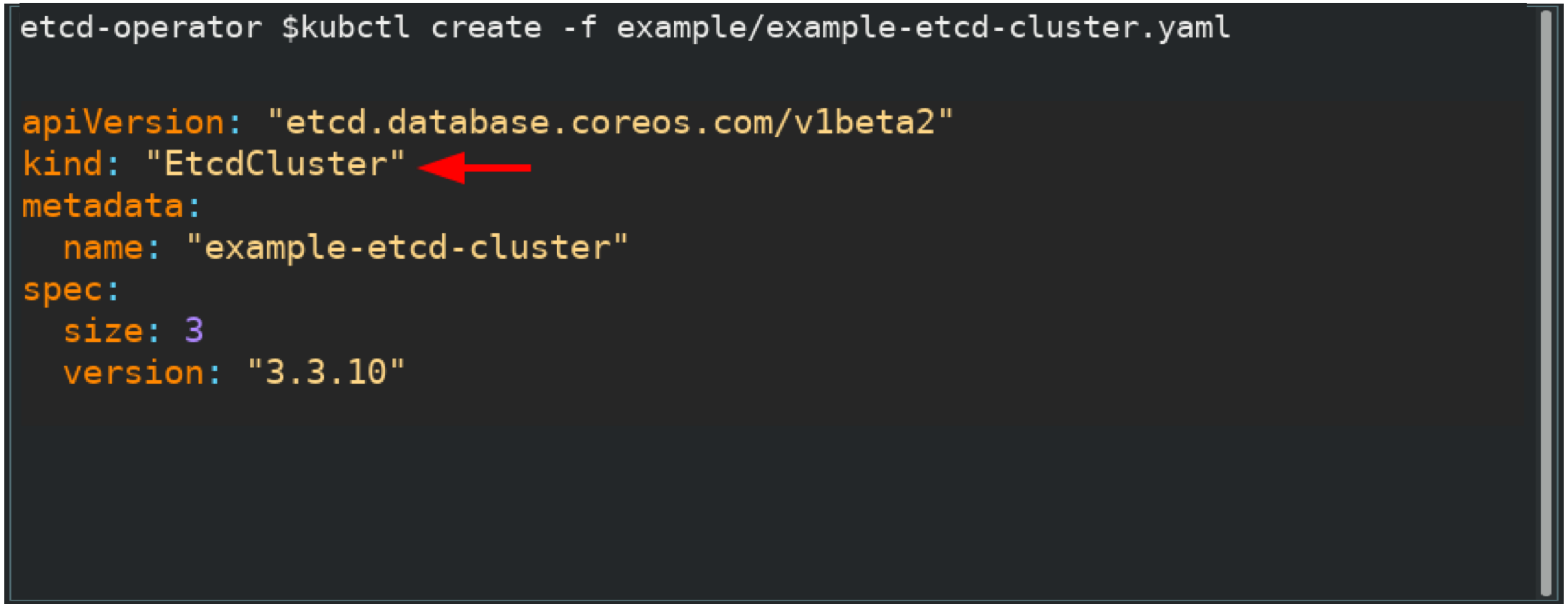


- Desired State: 3 node etcd cluster running v3.3.10
- Controller tasks
  - Define etcd configuration necessary to facilitate a 3 node cluster
  - Schedule 3 Pods on Kubernetes using v3.3.10 containers

etcd operator

KubeCon | CloudNativeCon North America 2018

Create 3 node EtcdCluster

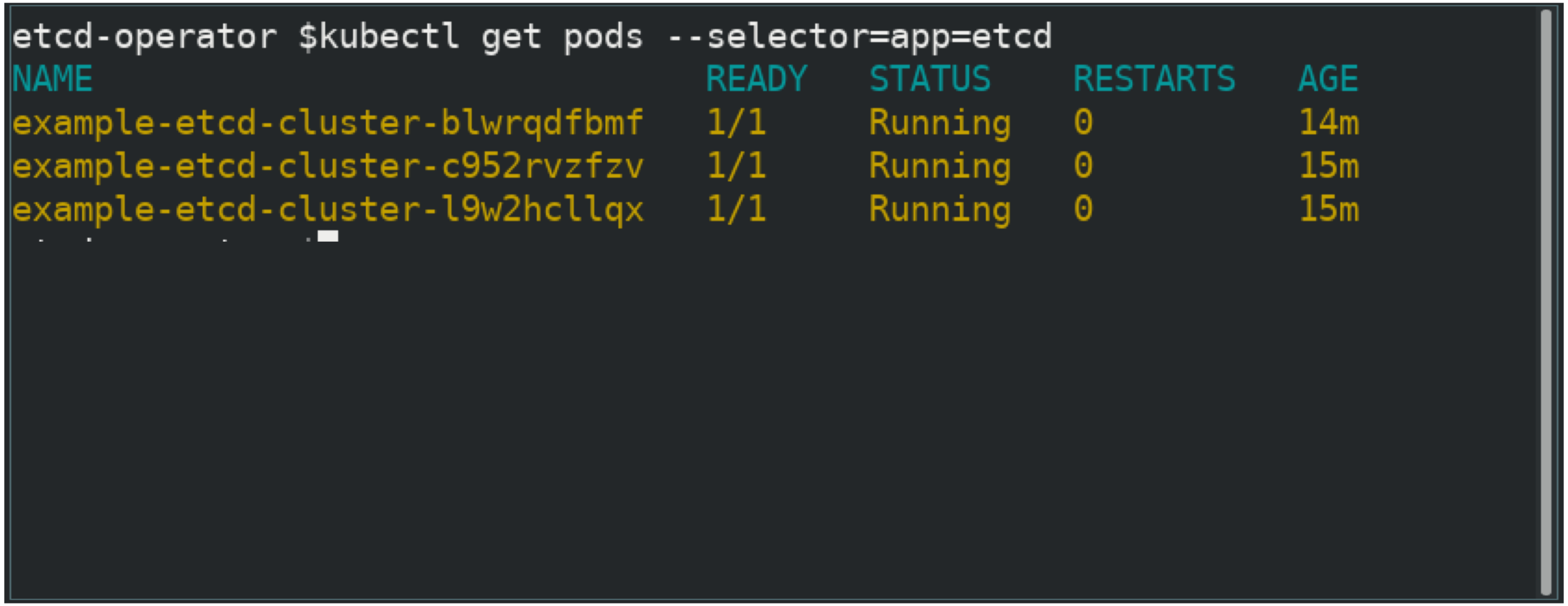


```
etcd-operator $kubctl create -f example/example-etcd-cluster.yaml

apiVersion: "etcd.database.coreos.com/v1beta2"
kind: "EtcdCluster"
metadata:
  name: "example-etcd-cluster"
spec:
  size: 3
  version: "3.3.10"
```

etcd operator

Cluster is running with 3 nodes

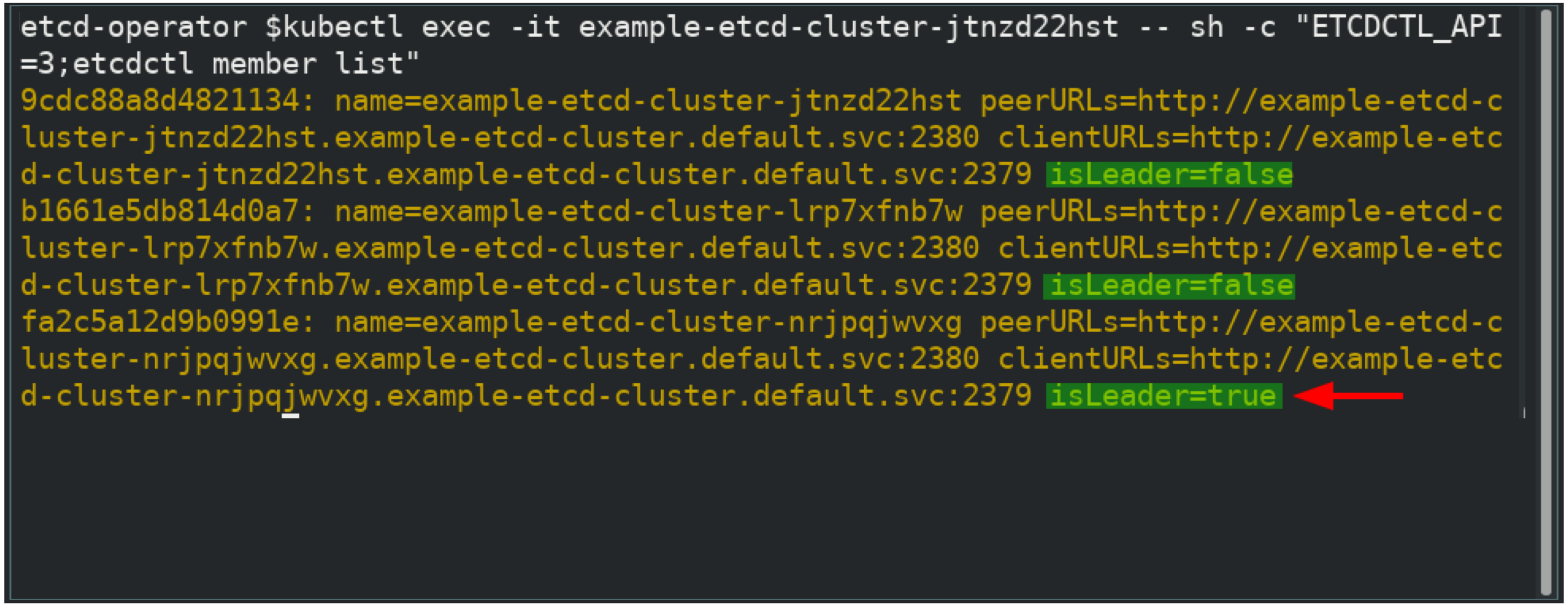


```
etcd-operator $kubectl get pods --selector=app=etcd
NAME                              READY   STATUS    RESTARTS   AGE
example-etcd-cluster-blwrqdfbmf   1/1     Running   0          14m
example-etcd-cluster-c952rvzfzv   1/1     Running   0          15m
example-etcd-cluster-l9w2hcllqx   1/1     Running   0          15m
```

etcd operator

KubeCon | CloudNativeCon
North America 2018

Cluster member list sanity check

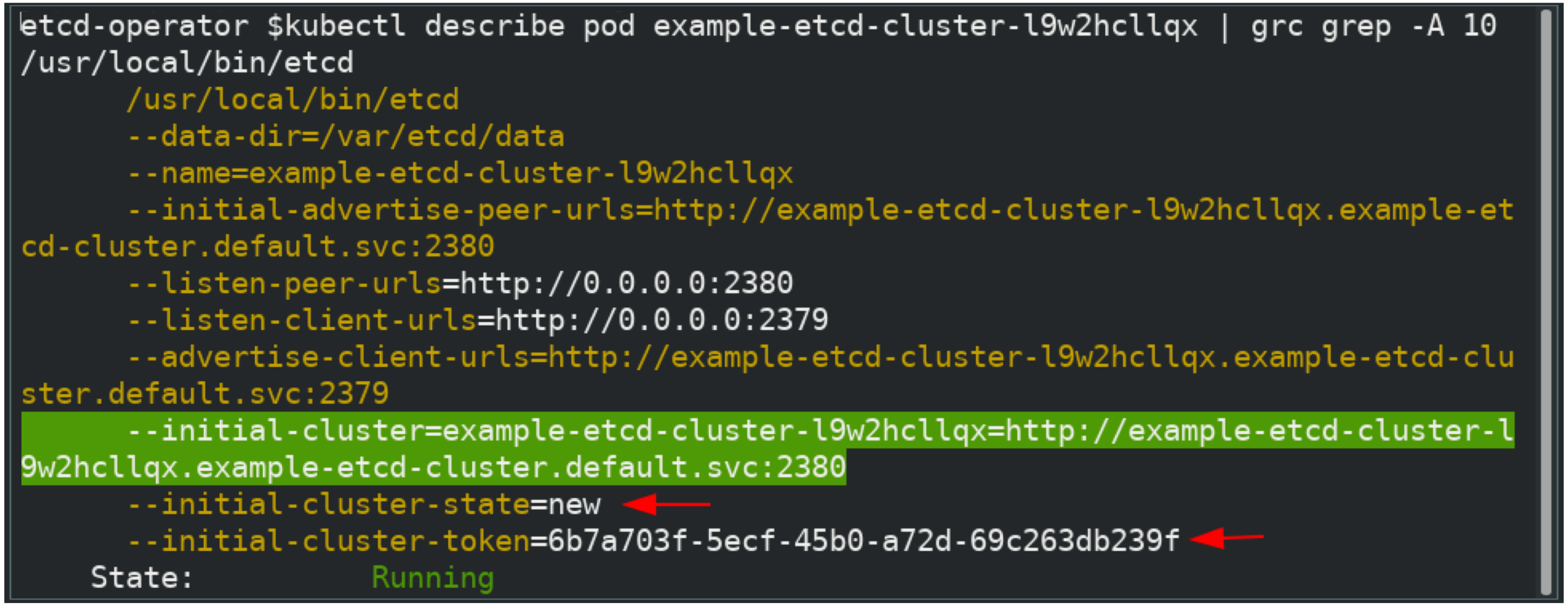


```
etcd-operator $kubectl exec -it example-etcd-cluster-jtnzd22hst -- sh -c "ETCDCTL_API
=3;etcdctl member list"
9cdc88a8d4821134: name=example-etcd-cluster-jtnzd22hst peerURLs=http://example-etcd-c
luster-jtnzd22hst.example-etcd-cluster.default.svc:2380 clientURLs=http://example-etc
d-cluster-jtnzd22hst.example-etcd-cluster.default.svc:2379 isLeader=false
b1661e5db814d0a7: name=example-etcd-cluster-lrp7xfnb7w peerURLs=http://example-etcd-c
luster-lrp7xfnb7w.example-etcd-cluster.default.svc:2380 clientURLs=http://example-etc
d-cluster-lrp7xfnb7w.example-etcd-cluster.default.svc:2379 isLeader=false
fa2c5a12d9b0991e: name=example-etcd-cluster-nrjpqjwvxg peerURLs=http://example-etcd-c
luster-nrjpqjwvxg.example-etcd-cluster.default.svc:2380 clientURLs=http://example-etc
d-cluster-nrjpqjwvxg.example-etcd-cluster.default.svc:2379 isLeader=true
```

etcd operator

KubeCon CloudNativeCon North America 2018

First node is a single node cluster

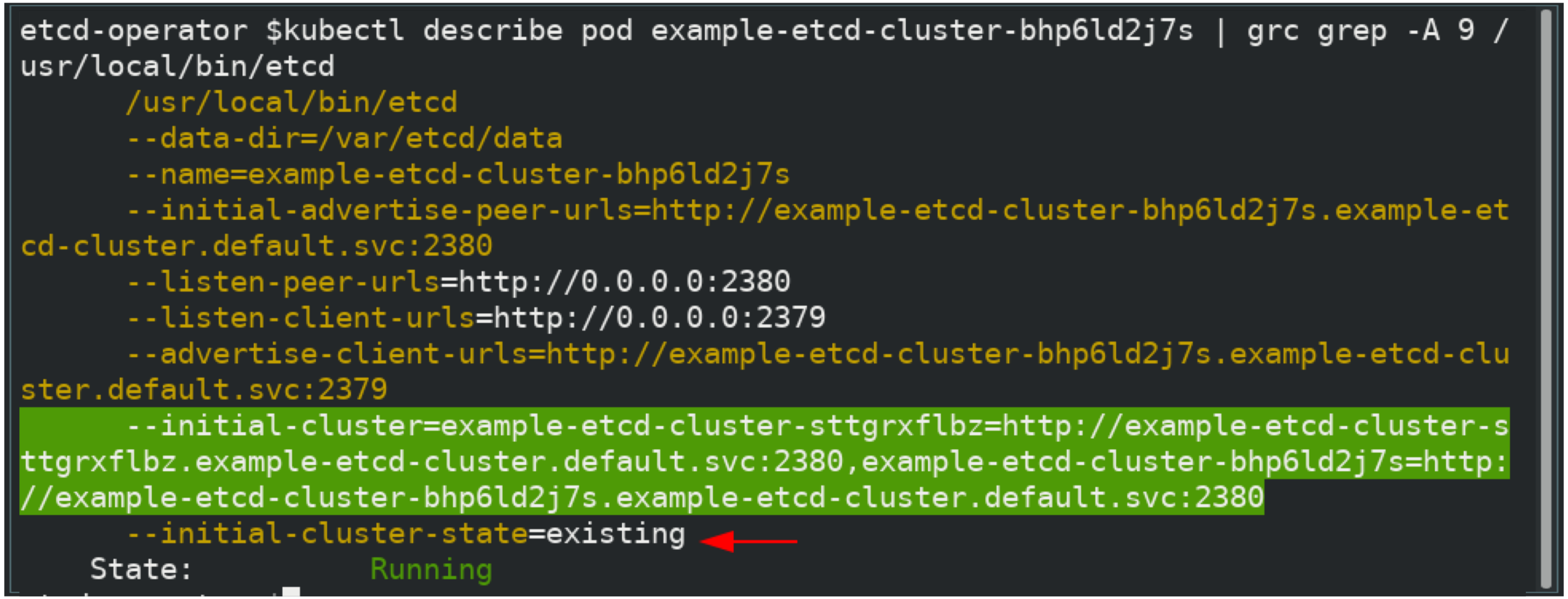


```
etcd-operator $kubectl describe pod example-etcd-cluster-l9w2hcllqx | grc grep -A 10
/usr/local/bin/etcd
      /usr/local/bin/etcd
      --data-dir=/var/etcd/data
      --name=example-etcd-cluster-l9w2hcllqx
      --initial-advertise-peer-urls=http://example-etcd-cluster-l9w2hcllqx.example-et
cd-cluster.default.svc:2380
      --listen-peer-urls=http://0.0.0.0:2380
      --listen-client-urls=http://0.0.0.0:2379
      --advertise-client-urls=http://example-etcd-cluster-l9w2hcllqx.example-etcd-clu
ster.default.svc:2379
      --initial-cluster=example-etcd-cluster-l9w2hcllqx=http://example-etcd-cluster-l
9w2hcllqx.example-etcd-cluster.default.svc:2380
      --initial-cluster-state=new
      --initial-cluster-token=6b7a703f-5ecf-45b0-a72d-69c263db239f
    State:          Running
```

etcd operator

KubeCon | CloudNativeCon
North America 2018

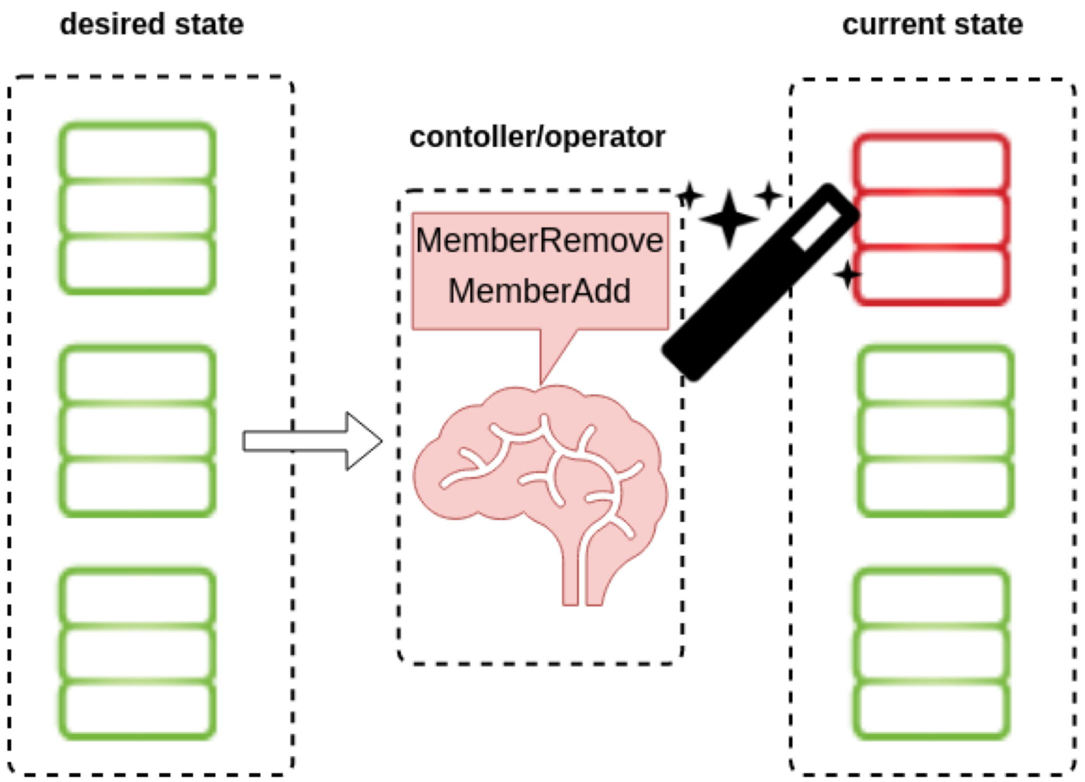
Second node joins existing cluster, smart!



```
etcd-operator $kubectl describe pod example-etcd-cluster-bhp6ld2j7s | grc grep -A 9 /
usr/local/bin/etcd
      /usr/local/bin/etcd
      --data-dir=/var/etcd/data
      --name=example-etcd-cluster-bhp6ld2j7s
      --initial-advertise-peer-urls=http://example-etcd-cluster-bhp6ld2j7s.example-et
cd-cluster.default.svc:2380
      --listen-peer-urls=http://0.0.0.0:2380
      --listen-client-urls=http://0.0.0.0:2379
      --advertise-client-urls=http://example-etcd-cluster-bhp6ld2j7s.example-etcd-clu
ster.default.svc:2379
      --initial-cluster=example-etcd-cluster-sttgrxflbz=http://example-etcd-cluster-s
ttgrxflbz.example-etcd-cluster.default.svc:2380,example-etcd-cluster-bhp6ld2j7s=http:
//example-etcd-cluster-bhp6ld2j7s.example-etcd-cluster.default.svc:2380
      --initial-cluster-state=existing
    State:          Running
```

etcd operator

Single node failure case

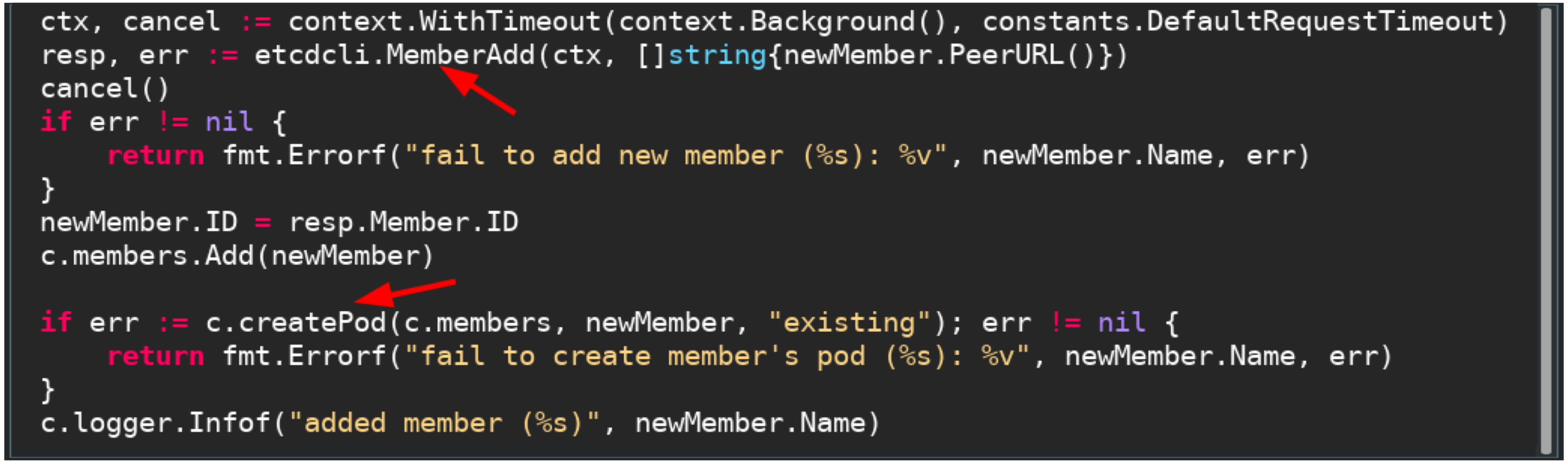


- One member fails
- Quorum is still maintained
- Controller tasks
    - Add new member with MemberAdd
    - Remove failed member with MemberRemove
    - Define proper etcd configuration
    - Schedule new pod

etcd operator

KubeCon | CloudNativeCon
North America 2018

Operator tasks, MemberAdd and createPod



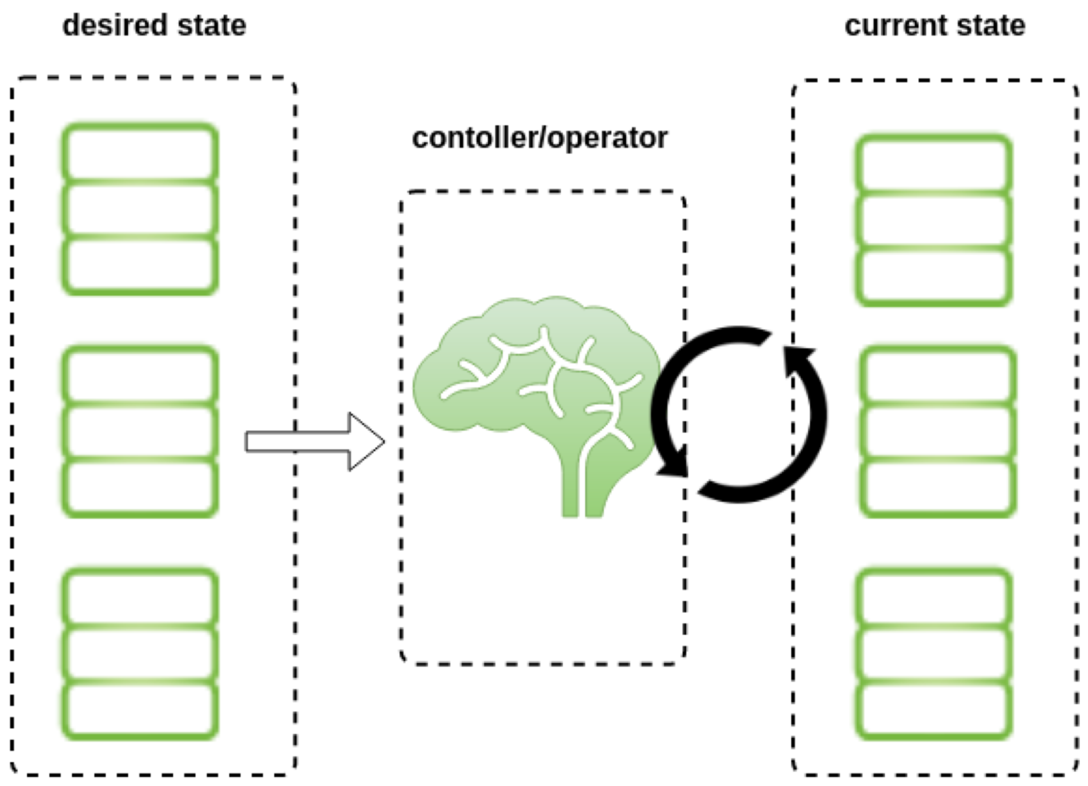
```go
ctx, cancel := context.WithTimeout(context.Background(), constants.DefaultRequestTimeout)
resp, err := etcdcli.MemberAdd(ctx, []string{newMember.PeerURL()})
cancel()
if err != nil {
    return fmt.Errorf("fail to add new member (%s): %v", newMember.Name, err)
}
newMember.ID = resp.Member.ID
c.members.Add(newMember)

if err := c.createPod(c.members, newMember, "existing"); err != nil {
    return fmt.Errorf("fail to create member's pod (%s): %v", newMember.Name, err)
}
c.logger.Infof("added member (%s)", newMember.Name)
```

pkg/cluster/reconcile.go

etcd operator

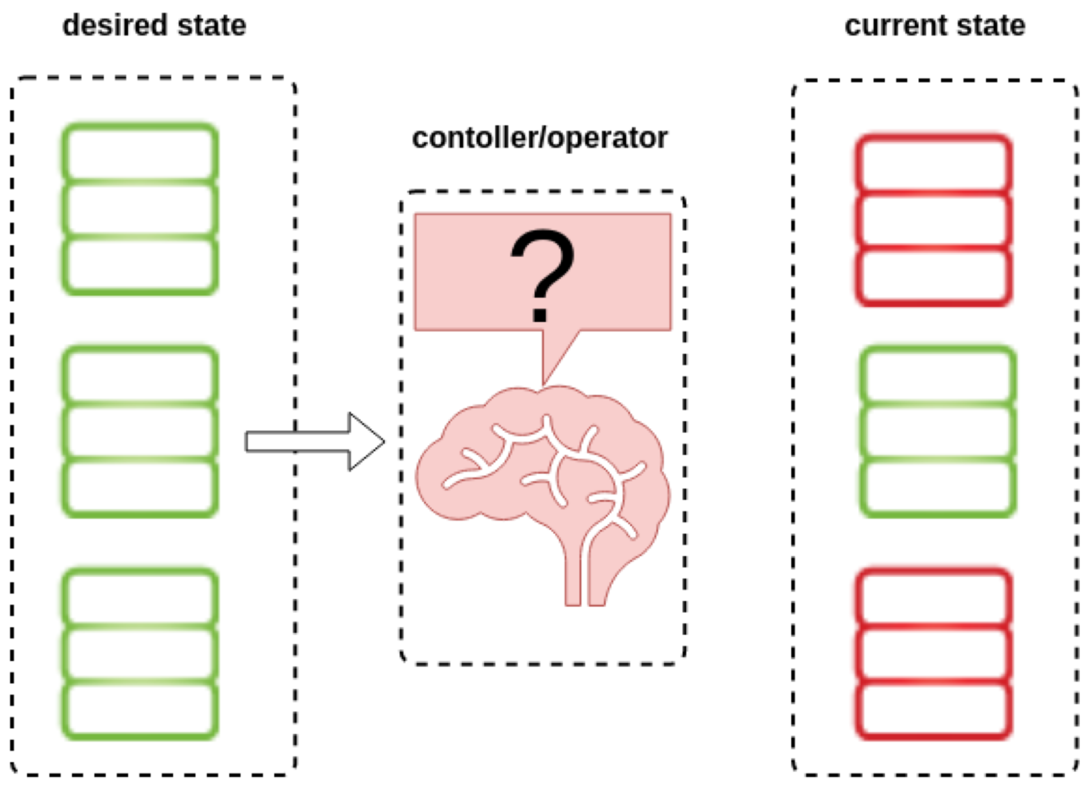
Cluster healed



- Operator worked as expected and solved cluster state.
- Eliminated an otherwise manual process.
- Success!

etcd operator

Multi node failure



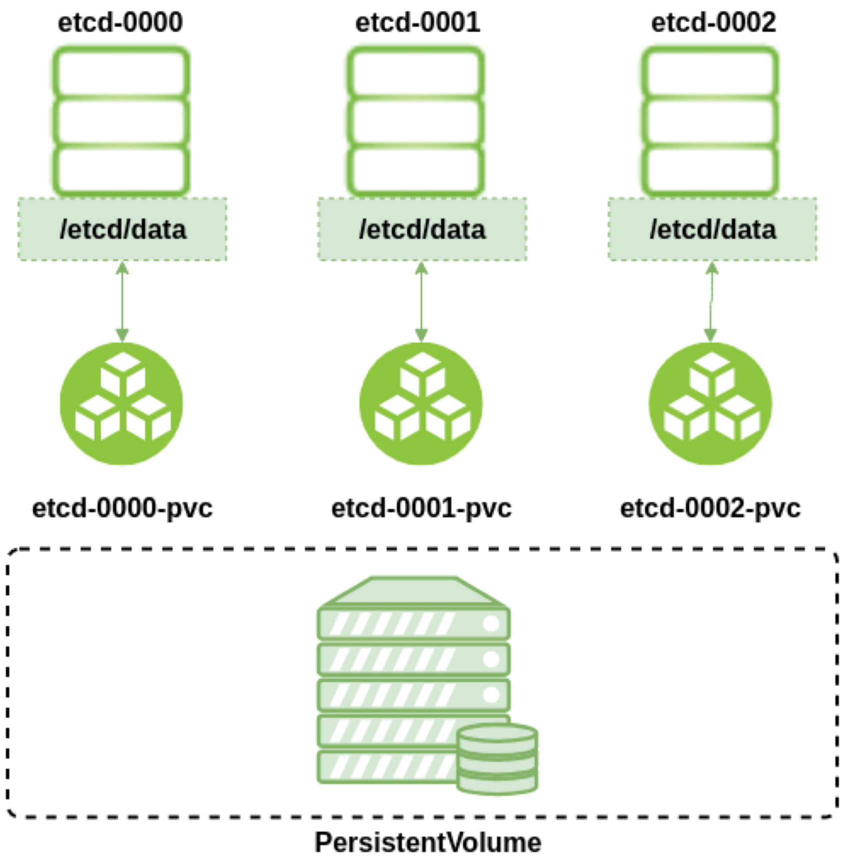
- Multiple members fail
- Quorum is lost
- Controller tasks
  - Cluster API not enough to resolve failed cluster
- Solution: Snapshot restore, configure and start new cluster
  - Possible data loss.
- Result: Failure :(

etcd operator

PersistentVolume Support

etcd operator

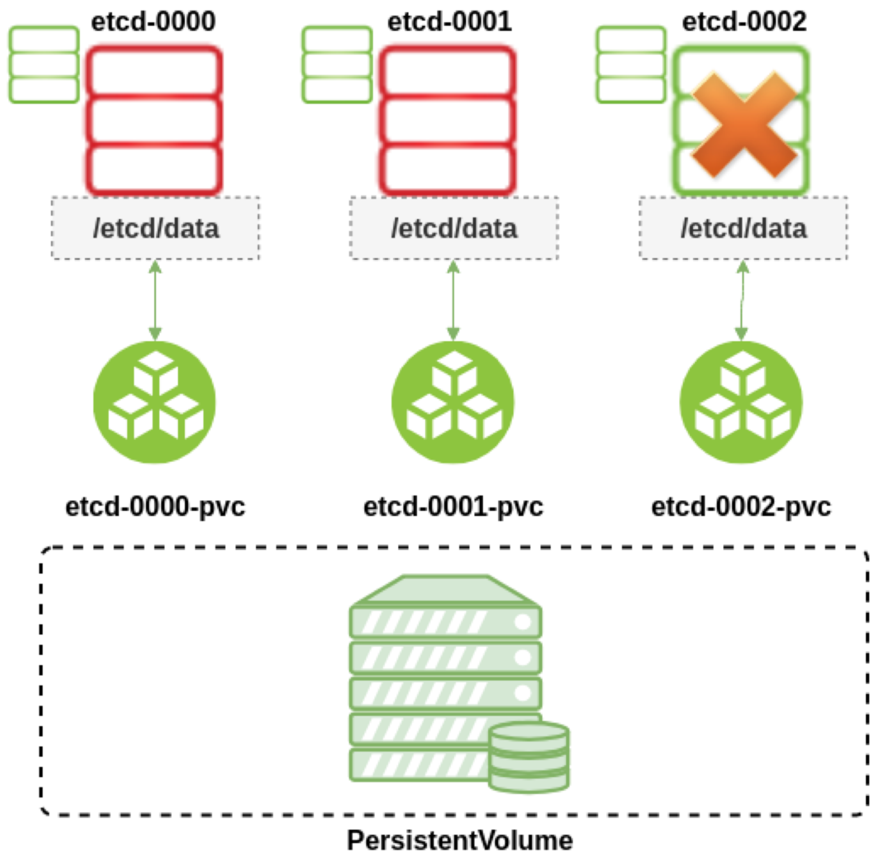
PersistentVolumes



- Initail support added added via PR #1861
- Adds persistence to data-dir outside life of pod
- Controller
  - Recycle vs RemoveMember?

etcd operator

Multi node failure and recovery



- Quorum would be lost
- Next step would involve restore from snapshot (manual).
- Controller tasks
  - Stop and remove all pods
  - Recreate all Pods using same name and reuse PVC.
- Result: Healthy cluster :)

Future Goals

- Solve PV/PVC corner cases
- Add etcd learner node support
- 1.0 release!

Challenge #3

etcd Upgrade

How does etcd upgrade work?

1. Leader fetches server versions from each peer
2. Leader picks the lowest version as cluster version
3. Leader broadcasts it to peers
4. Each peer tries to apply that cluster version
5. Fail if requested cluster version is downgrade

Upgrade must happen incrementally, one by one (rolling upgrade)

etcd maintains cluster version

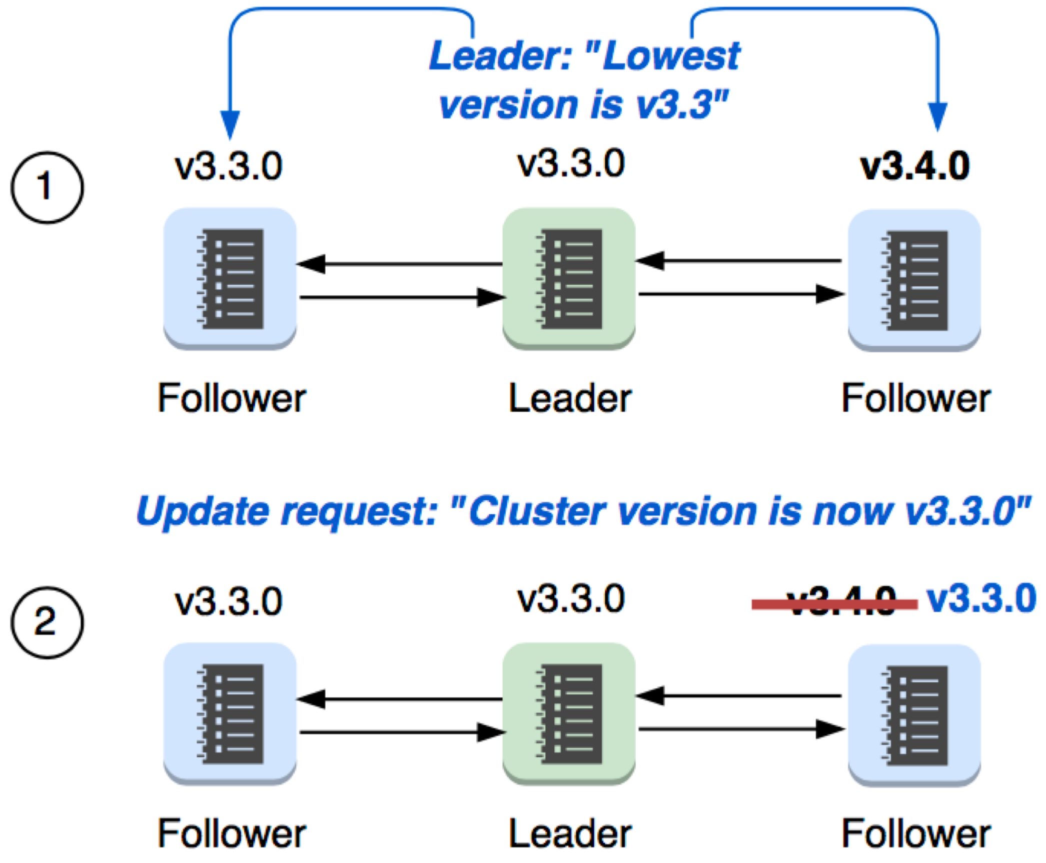


Figure 1. etcd uses cluster version to prevent downgrade. Leader fetches server versions from each peer, and picks the **lowest version as a cluster version**. And broadcasts it to peers.

etcd maintains cluster version

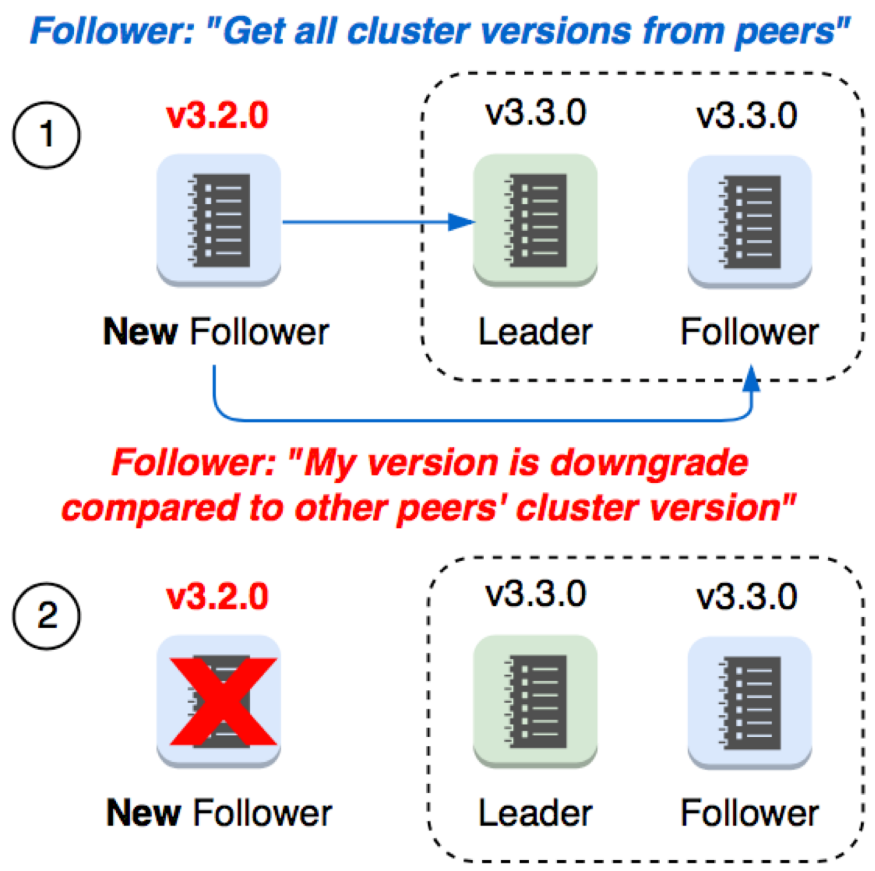

Follower: "Get all cluster versions from peers"
1
v3.2.0
v3.3.0
v3.3.0
New Follower
Leader
Follower
Follower: "My version is downgrade compared to other peers' cluster version"
2
v3.2.0
v3.3.0
v3.3.0
New Follower
Leader
Follower

Figure 2. When a **new follower joins the cluster, the new follower checks all cluster versions from its peers**. In etcd v3.3, if a follower server version is lower than peers' cluster version, the follower fails to start in order to prevent downgrade.

Introducing etcd Server Downgrade

Downgrade (etcd v3.4, 2019)

(Similar to etcd rolling upgrade)

"downgrade" command to temporarily whitelist a lower version:

```
etcdctl downgrade --target-version [TARGET_VERSION]
etcdctl downgrade status
etcdctl downgrade cancel
```

See github.com/etcd-io/etcd/issues/9306 for more detail.

(Credits to github.com/wenjiaswe at Google)

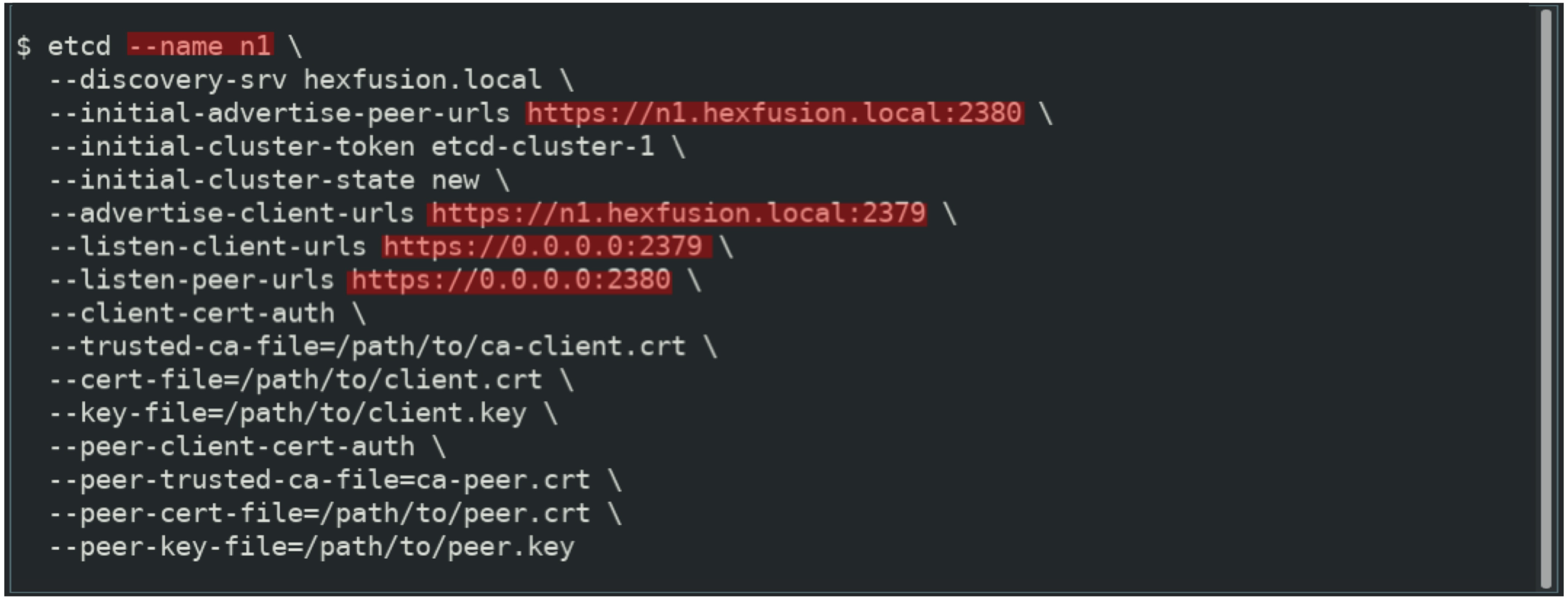
Challenge #4

Extensible Discovery

Challenge #4

KubeCon | CloudNativeCon
North America 2018

etcd SRV Discovery



```
$ etcd --name n1 \
  --discovery-srv hexfusion.local \
  --initial-advertise-peer-urls https://n1.hexfusion.local:2380 \
  --initial-cluster-token etcd-cluster-1 \
  --initial-cluster-state new \
  --advertise-client-urls https://n1.hexfusion.local:2379 \
  --listen-client-urls https://0.0.0.0:2379 \
  --listen-peer-urls https://0.0.0.0:2380 \
  --client-cert-auth \
  --trusted-ca-file=/path/to/ca-client.crt \
  --cert-file=/path/to/client.crt \
  --key-file=/path/to/client.key \
  --peer-client-cert-auth \
  --peer-trusted-ca-file=ca-peer.crt \
  --peer-cert-file=/path/to/peer.crt \
  --peer-key-file=/path/to/peer.key
```

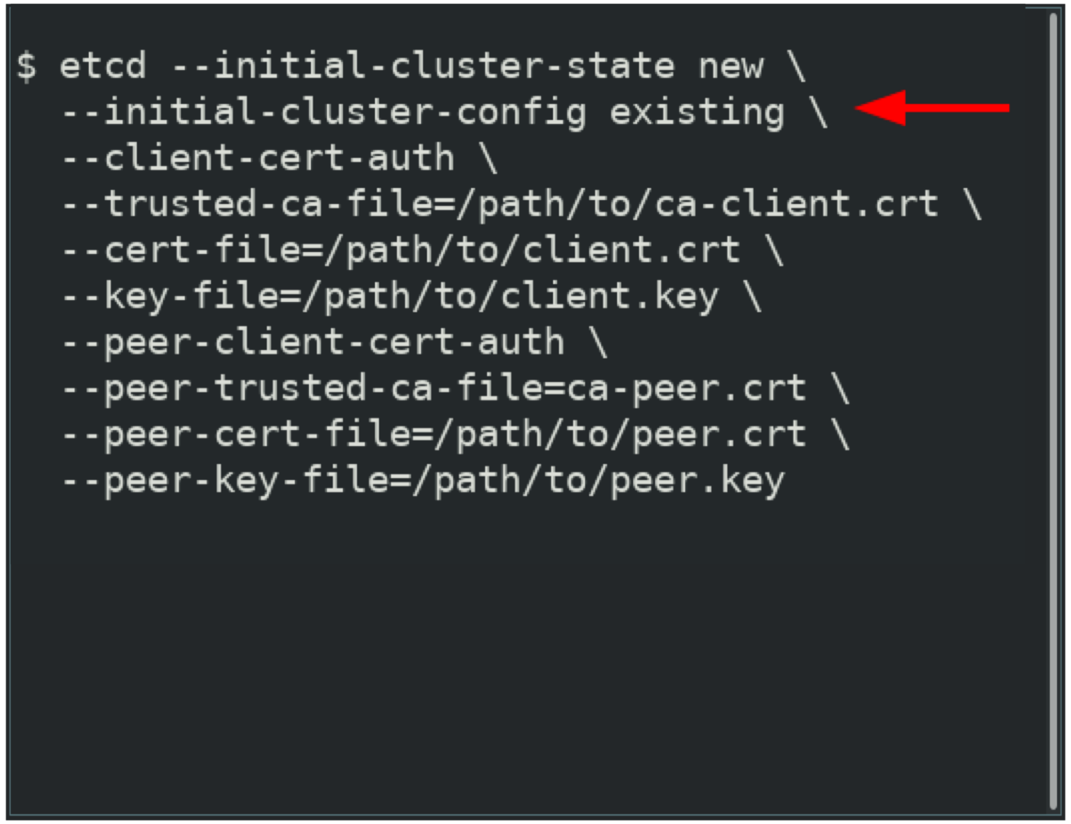
Cluster init: Proposal

Introducing clientv3 cluster init

Feature state: **initial proposal**

What if...

This was a multi node bootstrap configuration



```
$ etcd --initial-cluster-state new \
  --initial-cluster-config existing \
  --client-cert-auth \
  --trusted-ca-file=/path/to/ca-client.crt \
  --cert-file=/path/to/client.crt \
  --key-file=/path/to/client.key \
  --peer-client-cert-auth \
  --peer-trusted-ca-file=ca-peer.crt \
  --peer-cert-file=/path/to/peer.crt \
  --peer-key-file=/path/to/peer.key
```

- Static configuration
  - No server name
  - No IPs or domain names
  - Reusable
- Simplify Deployment
  - Discovery completed before etcd starts
- External discovery process
  - Client side
  - Easy to extend

Cluster init: Usage

- **Client**
  - etcdctl --cluster-init --discovery-srv=hexfusion.local
    - PeerURLs, ClientURLs and Name
    - Value persisted to store for member bucket
- **Server**
  - --initial-cluster-config exiting
    - Read values from store if they exist
    - Fall back to existing functionality

Cluster init: Layers

```
$ORIGIN hexfusion.local.
n1     IN      A      172.16.8.210

_etcd-server-etcd._tcp.hexfusion.local. 300 IN  SRV  0 0 2380 n1.hexfusion.local.
_etcd-client-etcd._tcp.hexfusion.local.   300 IN  SRV  0 0 2379 n1.hexfusion.local.
```

DNS

SRV query results

```
Member{
    ID: "a9ae0331e8a6683c",
    Name: "node1"
    peerURLs: []string{"http://n1.hexfusion.local:2380"},
    clientURLs: []string{"http://n1.hexfusion.local:2379"},
}
```

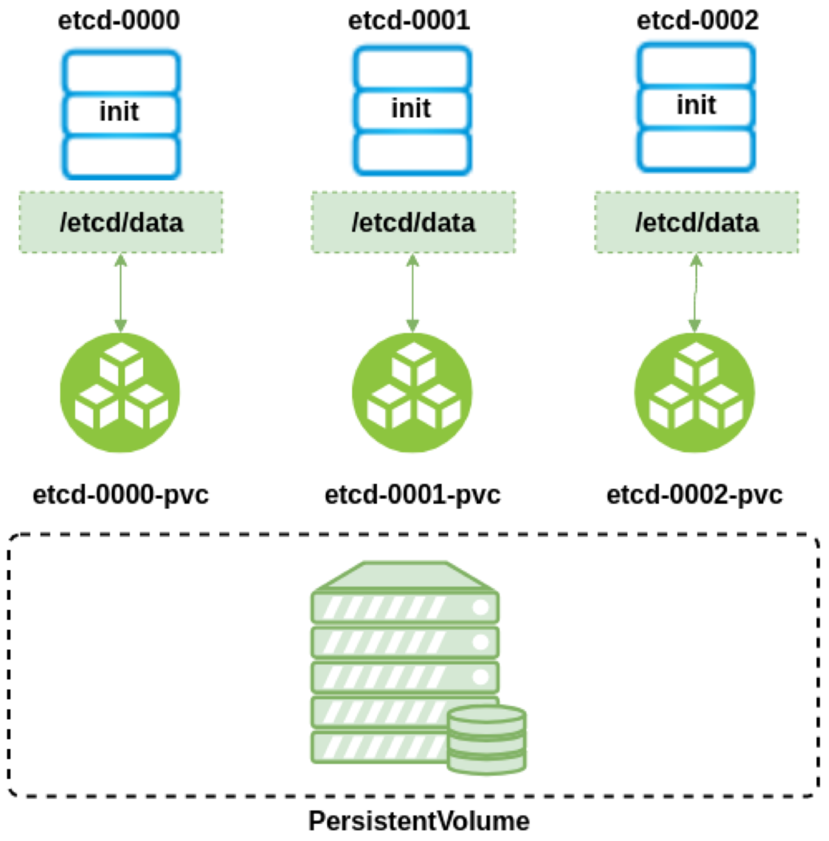
member bucket

```
key="a9ae0331e8a6683c"
value="{"id":12226713551248975932,"peerURLs":["http://n1.hexfusion.local:2380"],
"name":"n1","clientURLs":["http://n1.hexfusion.local:2379"]}"
```

Cluster init: Use case

KubeCon | CloudNativeCon
North America 2018

etcd operator PVC init container



- Bootstrap large cluster quickly
- Currently each node must start and join cluster before the next Pod starts
- With init container bootstrap N nodes all at the same time!

KubeCon | CloudNativeCon

North America 2018

Thank You!