KubeCon

CloudNativeCon

Europe 2020

Zero Database Downtime with etcd-operator

Tyler Lisowski- IBM
Kodie Glosser- IBM

Goals → Solution

1. Create an on-demand etcd cluster provisioning system
2. Automate all etcd database administration tasks
3. Eliminate etcd database downtime

etcd-operator

How are we going to meet these goals with etcd-operator?

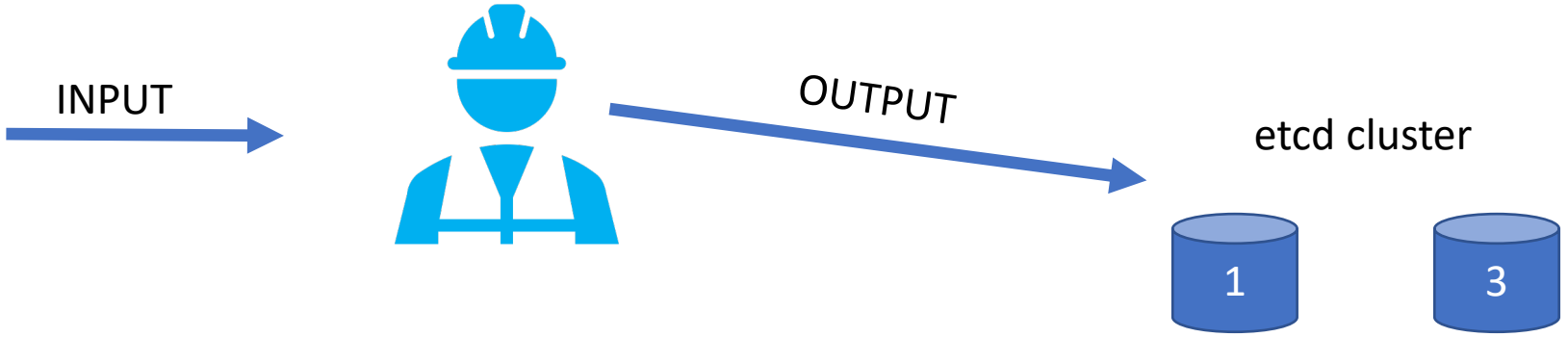
Delegate etcd cluster operational tasks to etcd-operator by defining the tasks in it's language.

Defining the database

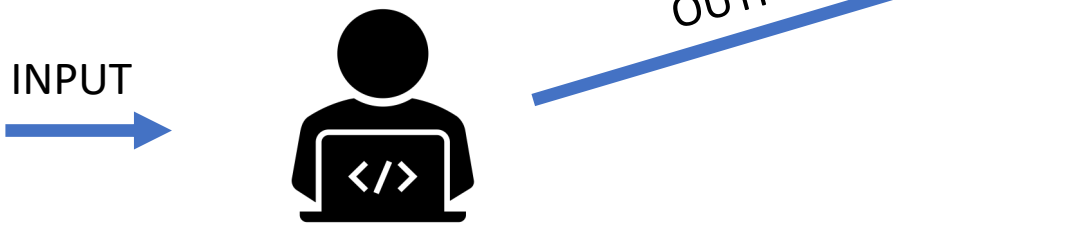
EtcdCluster Custom Resource Definition (CRD) is structured data that etcd-operator can read and update to create an etcd database.

etcd-operator's workflow

```
apiVersion: etcd.database.coreos.com/v1beta2
kind: EtcdCluster
metadata:
  name: etcd-cluster1
  labels:
    clusterID: "cluster1"
spec:
```



Human Engineer workflow



```yaml
apiVersion: etcd.database.coreos.com/v1beta2
kind: EtcdCluster
metadata:
  name: etcd-cluster1
  labels:
    clusterID: "cluster1"
spec:
  size: 3
  version: "3.4.2"
  repository: "registry.ng.bluemix.net/armada-master/etcd"
  pod:
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
                - key: etcd_cluster
                  operator: In
                  values: ["etcd-cluster1"]
            topologyKey: "failure-domain.beta.kubernetes.io/zone"
    etcdEnv:
      - name: ETCD_ELECTION_TIMEOUT
        value: "15000"
      - name: ETCD_HEARTBEAT_INTERVAL
        value: "100"
      - name: ETCD_SNAPSHOT_COUNT
        value: "10000"
      - name: ETCD_MAX_SNAPSHOTS
        value: "5"
      - name: ETCD_AUTO_COMPACTION_RETENTION
        value: "1"
      - name: ETCD_QUOTA_BACKEND_BYTES
        value: "4294967296"
      - name: ETCD_CIPHER_SUITES
        value: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,..."
  TLS:
    static:
      member:
        peerSecret: etcd-cluster1-peer-tls
        serverSecret: etcd-cluster1-server-tls
      operatorSecret: etcd-cluster1-client-tls
```

An In-Depth Look

Configuration categories

- 3 node multi availability zone etcd cluster
- Version control system with zero downtime upgrades
- etcd instance configuration

etcd-operator

I now have the necessary information to create the database. Let’s get to work…

Creating the database

etcd-operator is going to execute 4 reconciliation loops to create a cluster

EtcdCluster CRD Creation

Create initial cluster member

etcd cluster

1

Create 2nd cluster member

etcd cluster

1

2

Desired Size: 3

Create 3rd cluster member

etcd cluster

1

3

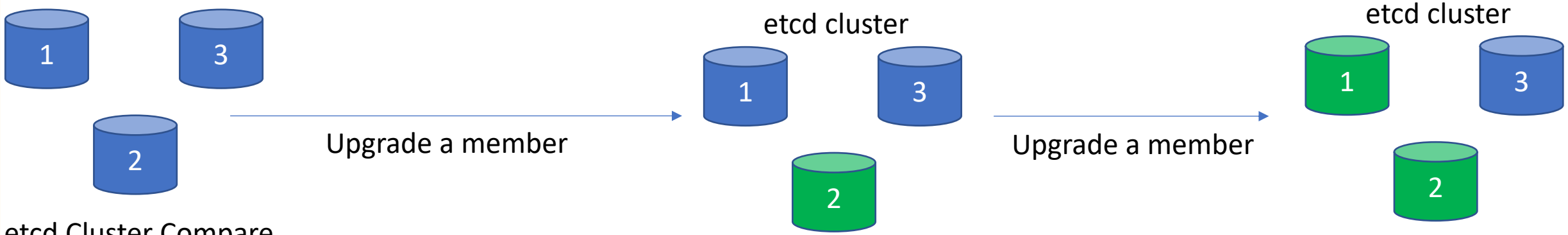
2

All members ready. Work is done time to relax

Total Create Time: <4 minutes

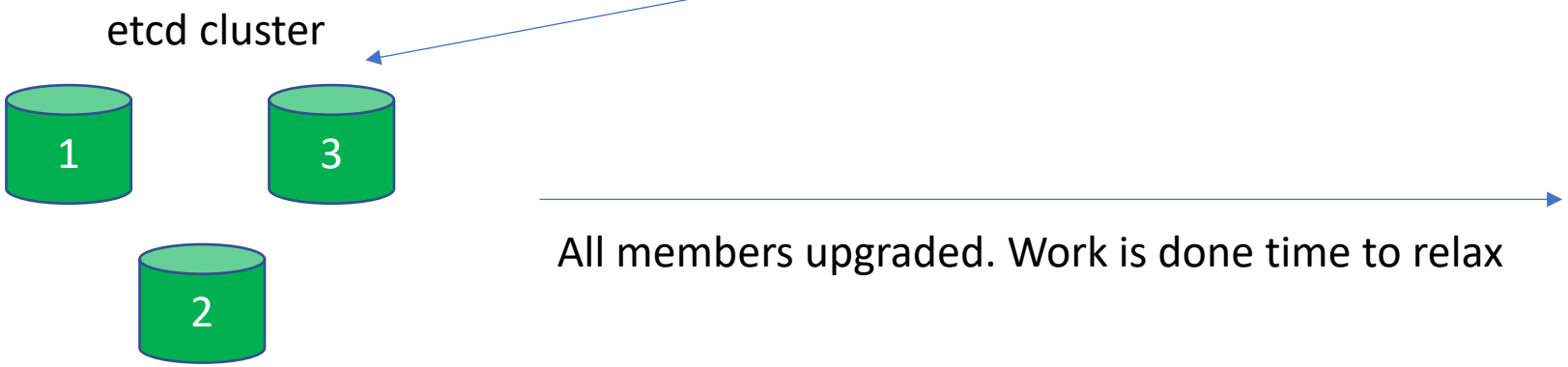
Updating the database

etcd-operator is going to execute 4 reconciliation loops to upgrade a cluster



etcd Cluster Compare

```
<   version: "3.4.2"
---
>   version: "3.4.3"
```

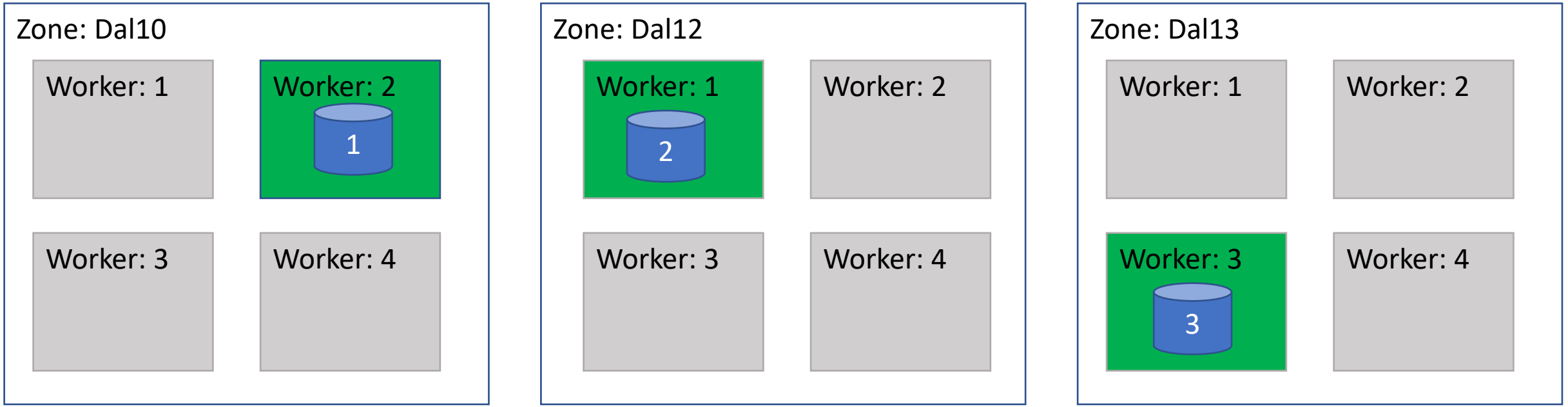


Total Upgrade Time: <4 minutes

What makes it highly available

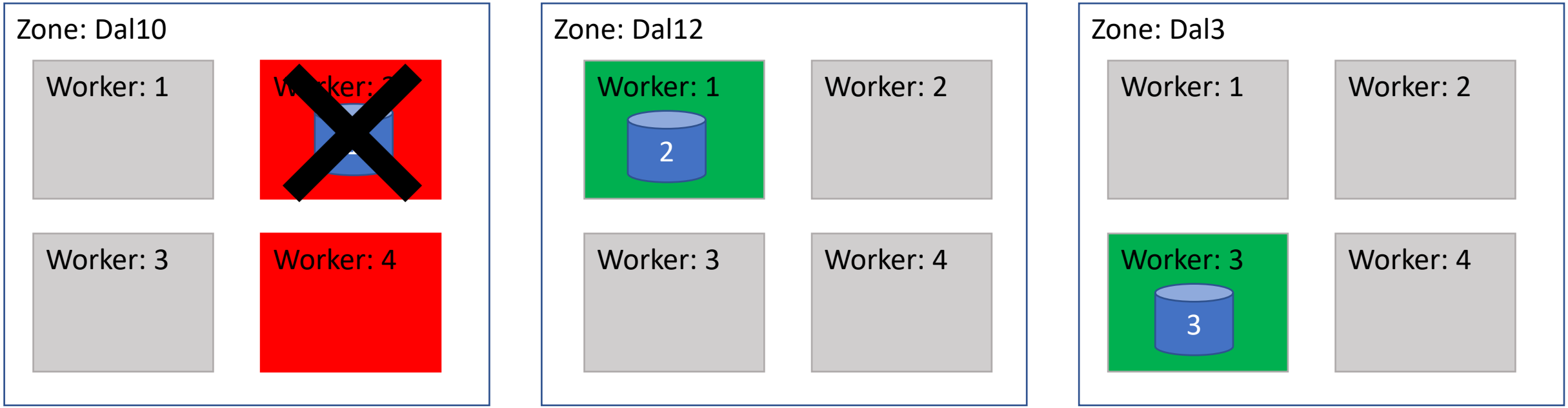

```
pod:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: etcd_cluster
                operator: In
                values: ["etcd-cluster1"]
          topologyKey: "failure-domain.beta.kubernetes.io/zone"
```

Scenario 1: Multiple nodes go down in 1 zone



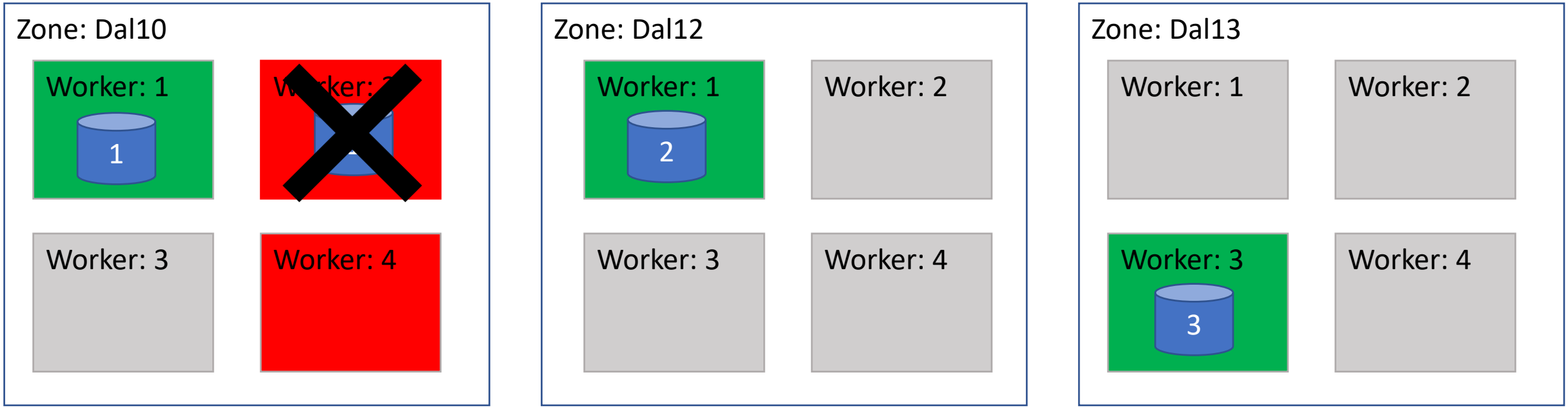
What makes it highly available

Scenario 1: Multiple nodes go down in 1 zone



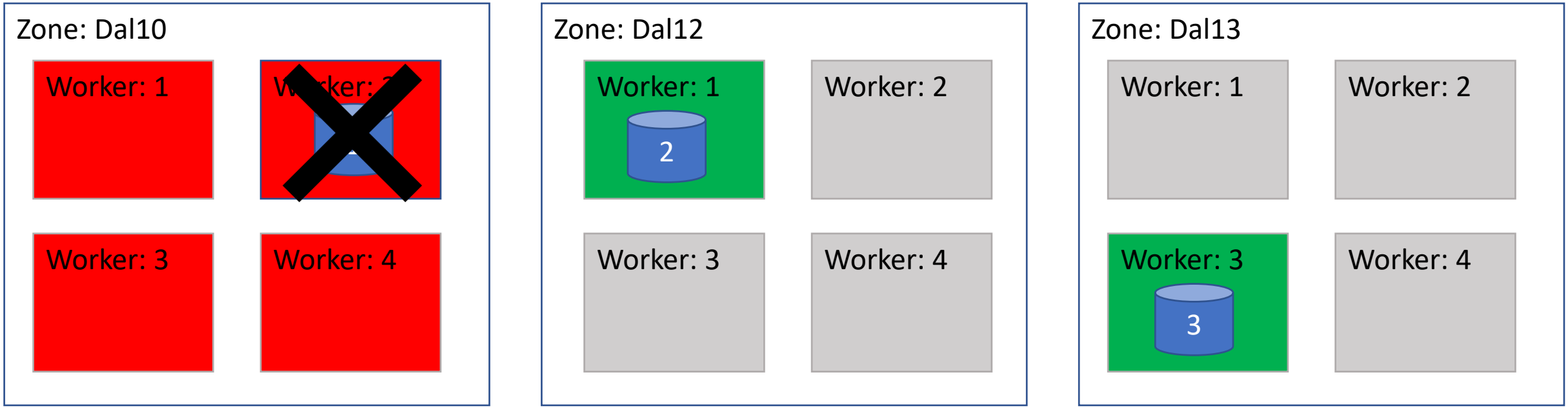
What makes it highly available

Scenario 1: Multiple nodes go down in 1 zone



What makes it highly available

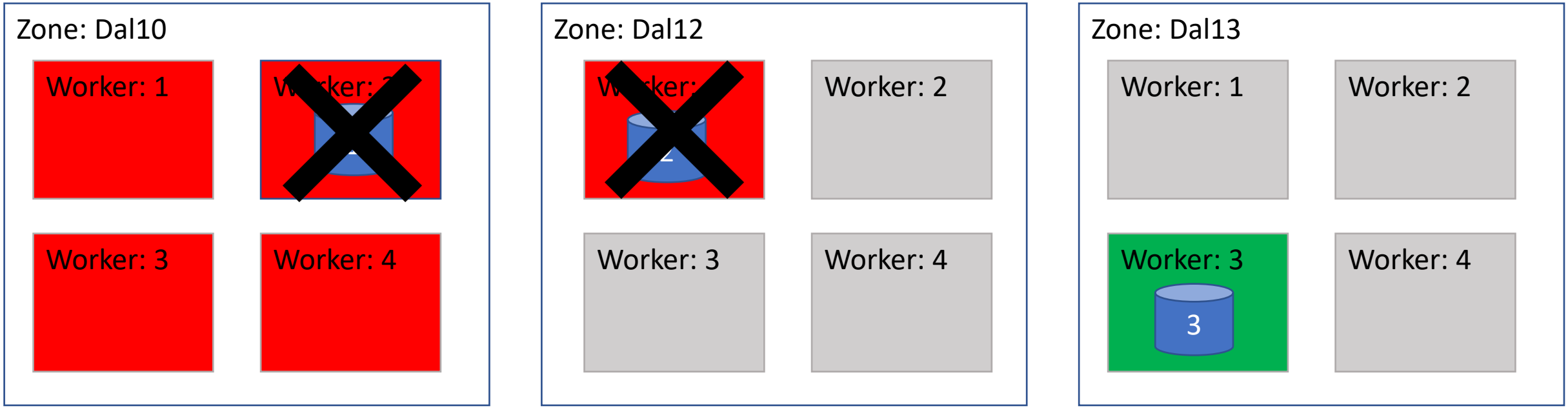
Scenario 2: All nodes go down in 1 zone



What makes it highly available

Scenario 3: Breaking quorum All nodes go down in 1 zone, lose another etcd pod in another

Quorum is the minimum number of members of an etcdcluster necessary to have it functioning. **Quorum** is achieved from having majority of pods in a running and healthy state. In our case 2/3 running pods.



Data is not lost while the 3rd etcd pod is running, but a manual recovery is required

Disaster Recovery – Etcd Backup


```yaml
apiVersion: "etcd.database.coreos.com/v1beta2"
kind: "EtcdBackup"
metadata:
  name: etcd-kodie-1
  namespace: kubx-etcd-01
spec:
  etcdEndpoints:
    - "http://etcd-kodie-1.kubx-etcd-01.svc.cluster.local:2379"
  storageType: S3
  backupPolicy:
    timeoutInSecond: 10
  s3:
    path: dev-south-iks-etcd-backups/cluster1/backups/backup.db
    awsSecret: cos-credentials
    endpoint: https://s3.us-south.objectstorage.softlayer.net
```

Configuration categories

- Backup location information
- Etcd instance location

Disaster Recovery – Etcd Restore

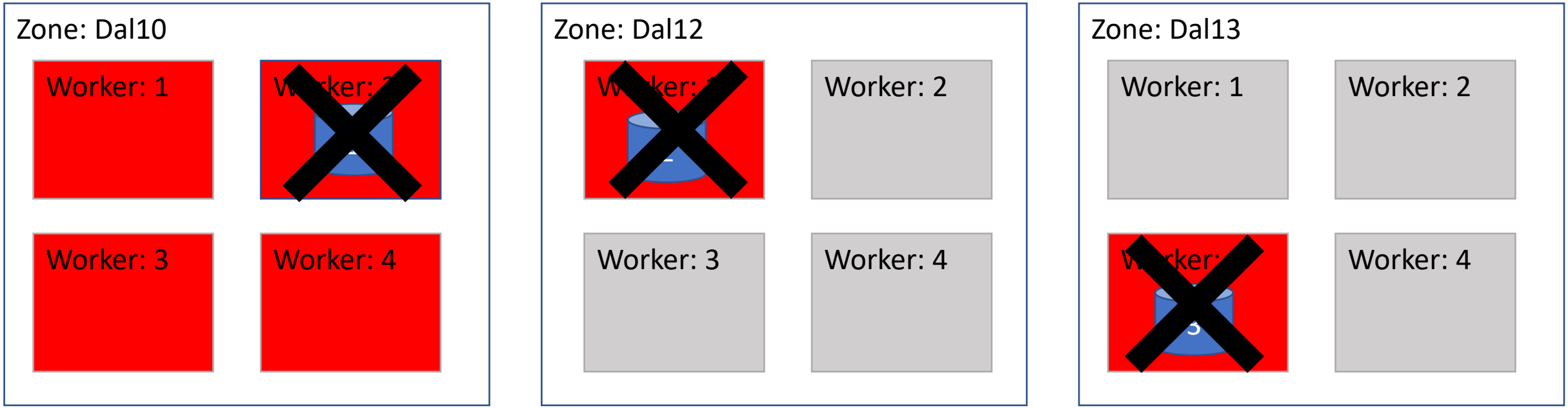

```yaml
apiVersion: "etcd.database.coreos.com/v1beta2"
kind: "EtcdRestore"
metadata:
  name: etcd-cluster1
spec:
  etcdCluster:
    name: etcd-cluster1
  backupStorageType: S3
  s3:
    path: dev-south-iks-etcd-backups/cluster1/backups/backup.db
    awsSecret: cos-credentials
    endpoint: https://s3.us-south.objectstorage.softlayer.net
```

Configuration categories

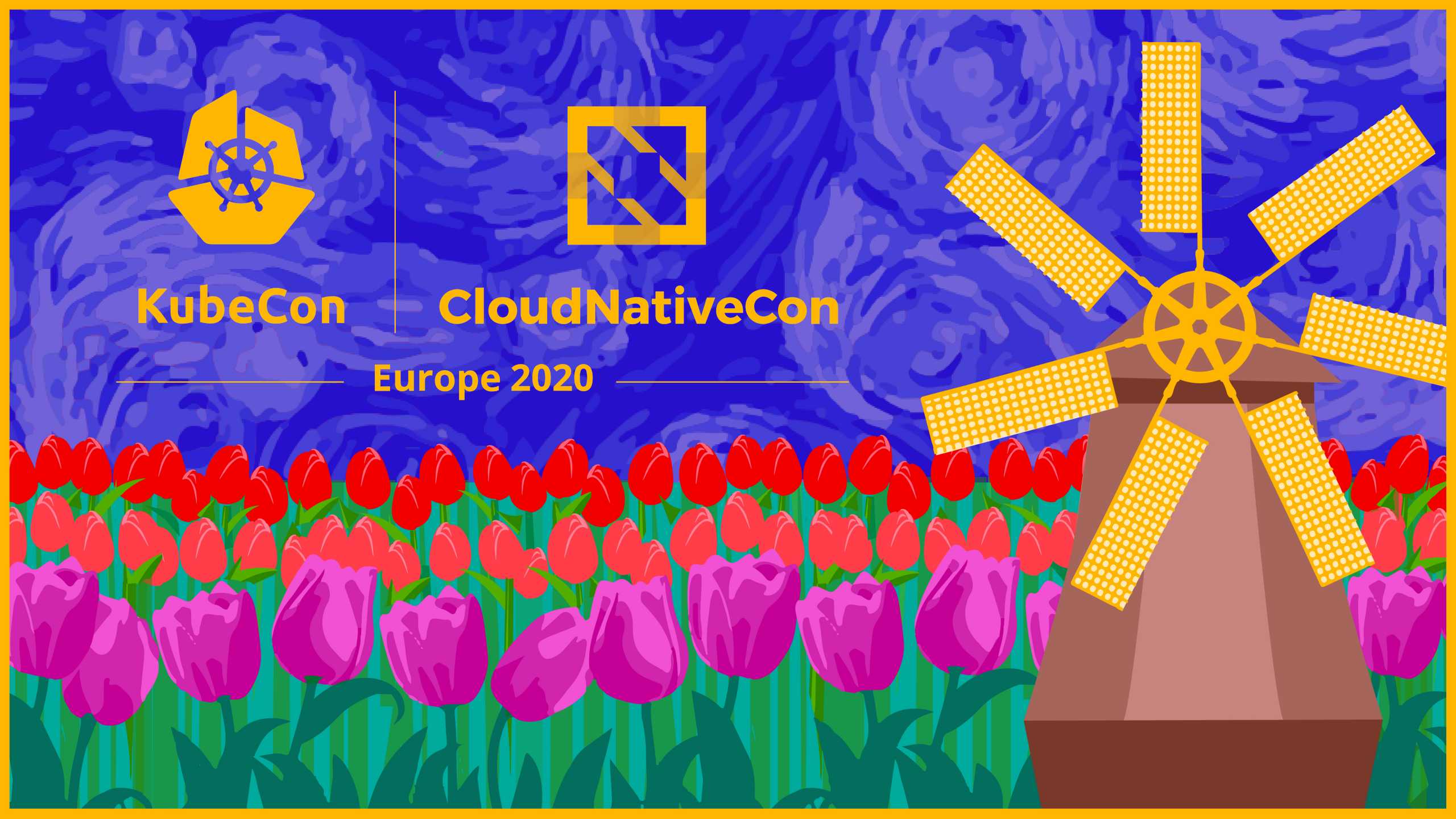
- Backup location information
- Etcd instance location

What makes it highly available

Scenario 4: Breaking quorum. All etcd pods go down



Data is now lost. A manual recovery is needed with most recent data from an external storage location


KubeCon
CloudNativeCon
Europe 2020