Big Data Operations Using Kubernetes and Local Storage

Dan Norris
Senior Cloud Native Engineer @NetApp
@protochron

Agenda

- Background
- Cassandra
- Local Storage
- Cassandra + K8S
- Operations

Background

What to get out of this talk

- A high-level description to running Cassandra on Kubernetes
- Example of *operations* the setup allows you to encode

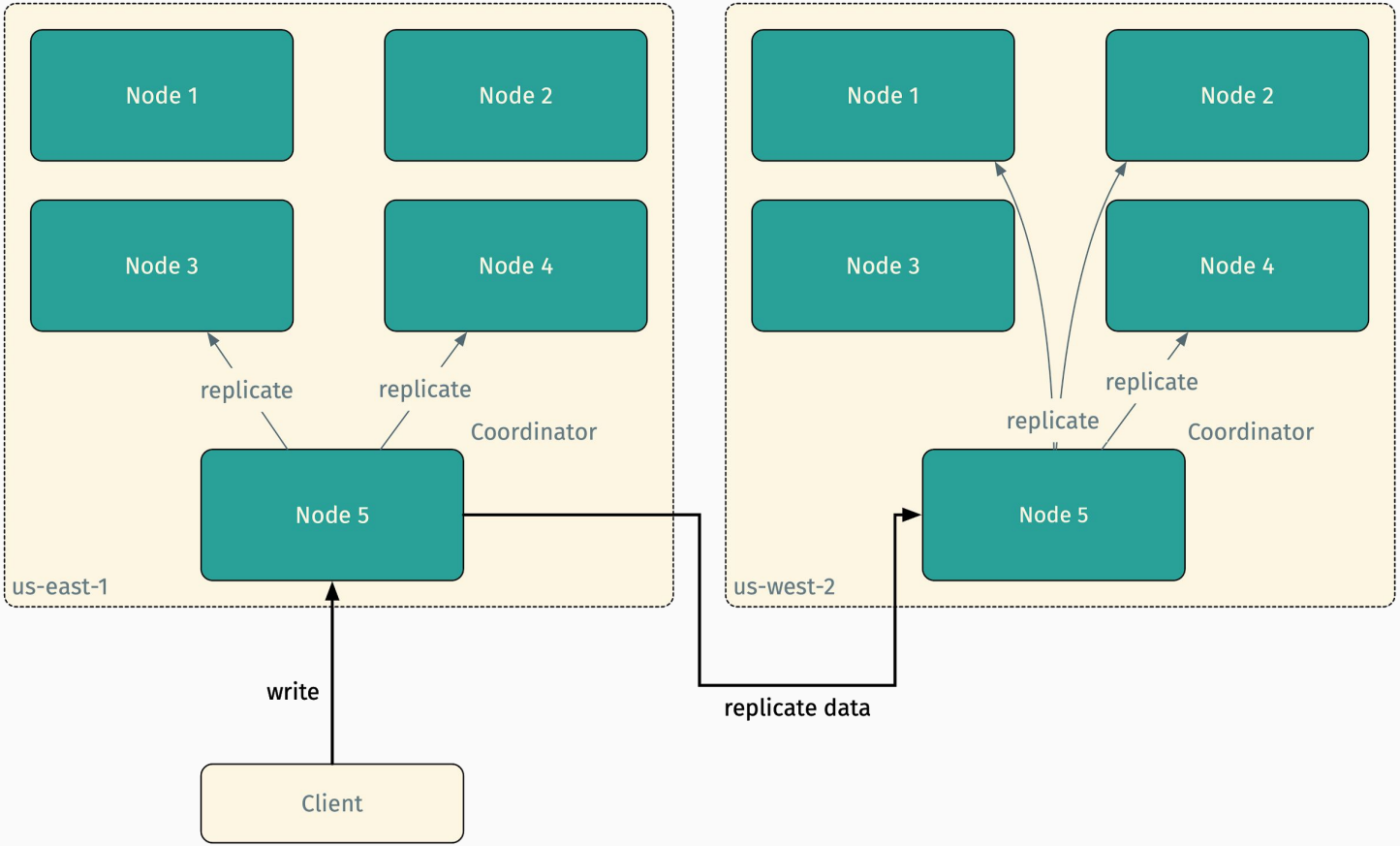
Operations

- Encode common operations using Kubernetes building blocks
  - `StatefulSets`
  - `PersistentVolumes`
  - `Jobs`


cassandra

Data Model and Replication

- Eventually consistent by design (fault tolerant)
- Replication is configurable per Datacenter/Region
  - Ex. 2 copies in us-east-1 and 3 copies in us-west-2
- Write/read consistency is tunable
  - Quorum, Local Quorum, One, etc.


Node 1
Node 2
Node 3
Node 4
replicate
replicate
Coordinator
Node 5
us-east-1
Node 1
Node 2
Node 3
Node 4
replicate
replicate
Coordinator
Node 5
us-west-2
write
replicate data
Client

Problems

- Difficult to operate
- Built for a pre-container world
  - Many commands to run manually
  - Nodes are discovered by IP
    - No ip:port pairing like etcd
- Requires in-depth knowledge for tuning

Local Storage

What is Local Storage

- Local `PersistentVolumes`
  - Beta in 1.12
- Expose directories on nodes as `PersistentVolumes`
- Better abstraction than `hostPath`
  - Let scheduler worry about locality
  - Hide local paths from pod

Why Use Local Storage?

- Bare metal
- Different types of disks in different nodes
- May have custom hardware or technology in the mix
- Network storage may not be an option

Before I go any further

Local storage makes your nodes snowflakes

Snowflakes

- Something to avoid
- Goes against Kubernetes view of running applications
  - Data and node locality start to matter

You should use network storage if possible!

Using Local Storage

StorageClass

An empty provisioner indicates Local Storage

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-cassandra
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
# Supported policies: Delete, Retain
reclaimPolicy: Delete
```

StorageClass

Prevent volume binding until pods request it

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-cassandra
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
# Supported policies: Delete, Retain
reclaimPolicy: Delete
```

Example Local Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-cassandra
  local:
    path: /opt/local-storage/cassandra
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

Example Local Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-cassandra
  local:
    path: /opt/local-storage/cassandra
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

Example Local Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-cassandra
  local:
    path: /opt/local-storage/cassandra
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

Local Storage Node Affinity

- Uses `NodeAffinity` to bind pods to a node
- Forces Kubernetes to only schedule to that node
- You can also use anti-affinity on your pods
  - Ex. prevent MySQL and Cassandra from running on the same node

Example Claim

Typically consume Local Storage using a PersistentVolumeClaim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: example-cassandra-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: local-cassandra
```

External Volume Provisioner

- `DaemonSet` to create `PersistentStorage` volumes from directories/mount paths on a node
- https://github.com/kubernetes-incubator/external-storage/tree/master/local-volume
- Map `StorageClasses` to local directories and provide `PersistentVolumes`

Building Blocks

StatefulSets

2 stateful sets for Cassandra

- Seeds
- Nodes

(+ corresponding services)


Node StatefulSet (scale = 2)
Seed StatefulSet (scale = 1)
Cassandra Pod
Kubernetes node
Cassandra Pod
Kubernetes node
Cassandra Seed Pod
Kubernetes node
Kubernetes cluster

Stateful Sets

- Two stateful sets allow you to stage updates to *nodes* first and then *seeds*
- Can sync seed IPs as they change
  - Cassandra does care about seed IP addresses

Building Blocks

Local Storage

- Allocated on each node

Local Storage

- Prep nodes individually
- External volume provisoner expects *mounts*

```
sudo mkdir -p /opt/cassandra && \
sudo mkdir -p
/opt/local-storage/cassandra && \
sudo mount --bind /opt/cassandra
/opt/local-storage/cassandra
```

Configure external volume provisioner

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: local-provisioner-config
namespace: default
data:
  storageClassMap: |
    local-cassandra:
      hostDir: /opt/local-storage/cassandra
      mountDir:  /opt/local-storage/cassandra
      blockCleanerCommand:
        - "/scripts/shred.sh"
        - "2"
      volumeMode: Filesystem
      fsType: ext4
```

Configure external volume provisioner

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: local-provisioner-config
namespace: default
data:
  storageClassMap: |
    local-cassandra:
      hostDir: /opt/local-storage/cassandra
      mountDir:  /opt/local-storage/cassandra
      blockCleanerCommand:
        - "/scripts/shred.sh"
        - "2"
      volumeMode: Filesystem
      fsType: ext4
```

Configure external volume provisioner

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: local-provisioner-config
namespace: default
data:
  storageClassMap: |
    local-cassandra:
      hostDir: /opt/local-storage/cassandra
      mountDir:  /opt/local-storage/cassandra
      blockCleanerCommand:
        - "/scripts/shred.sh"
        - "2"
      volumeMode: Filesystem
      fsType: ext4
```

Configure external volume provisioner

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: local-provisioner-config
namespace: default
data:
  storageClassMap: |
    local-cassandra:
      hostDir: /opt/local-storage/cassandra
      mountDir:  /opt/local-storage/cassandra
      blockCleanerCommand:
        - "/scripts/shred.sh"
        - "2"
      volumeMode: Filesystem
      fsType: ext4
```

Building Blocks

`ConfigMap`

- cassandra.yaml
- jvm.options
- + any other config files you need

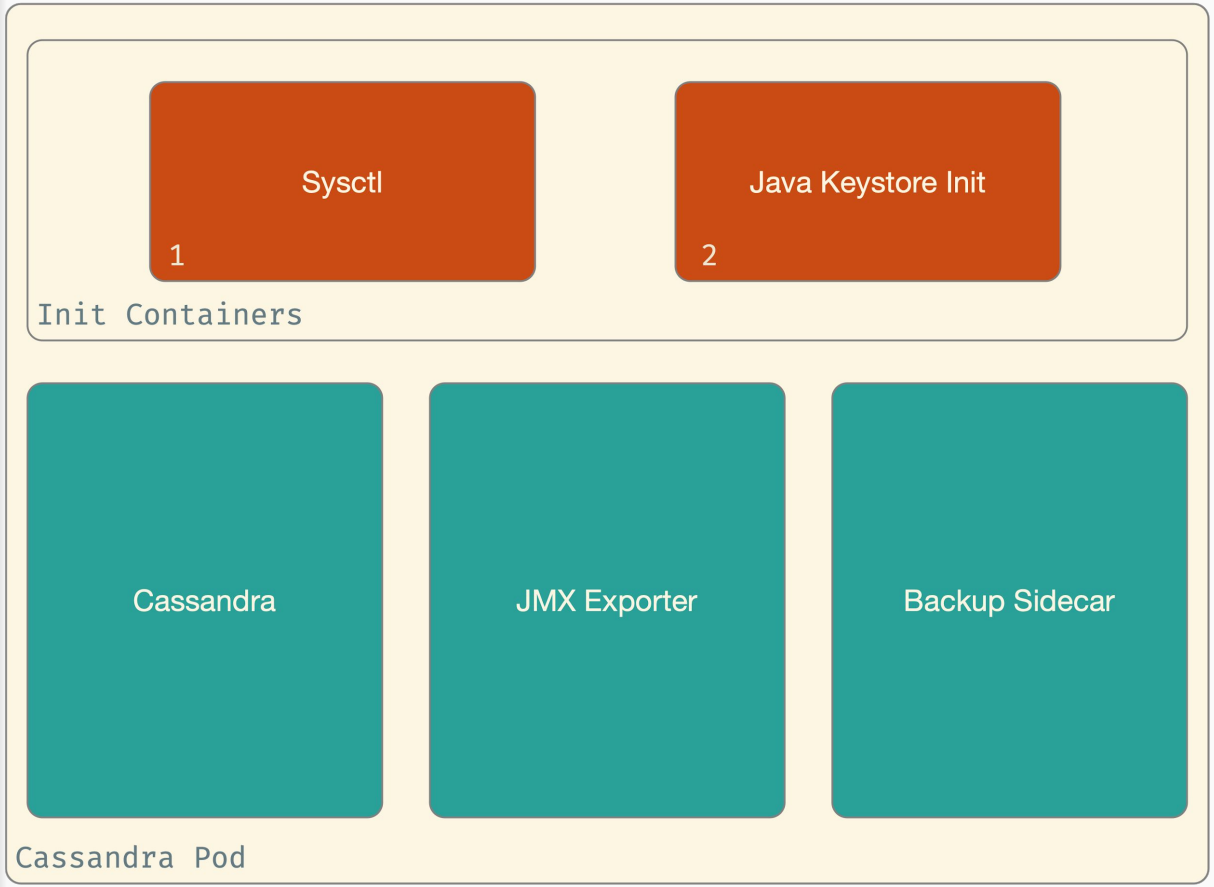
Building Blocks

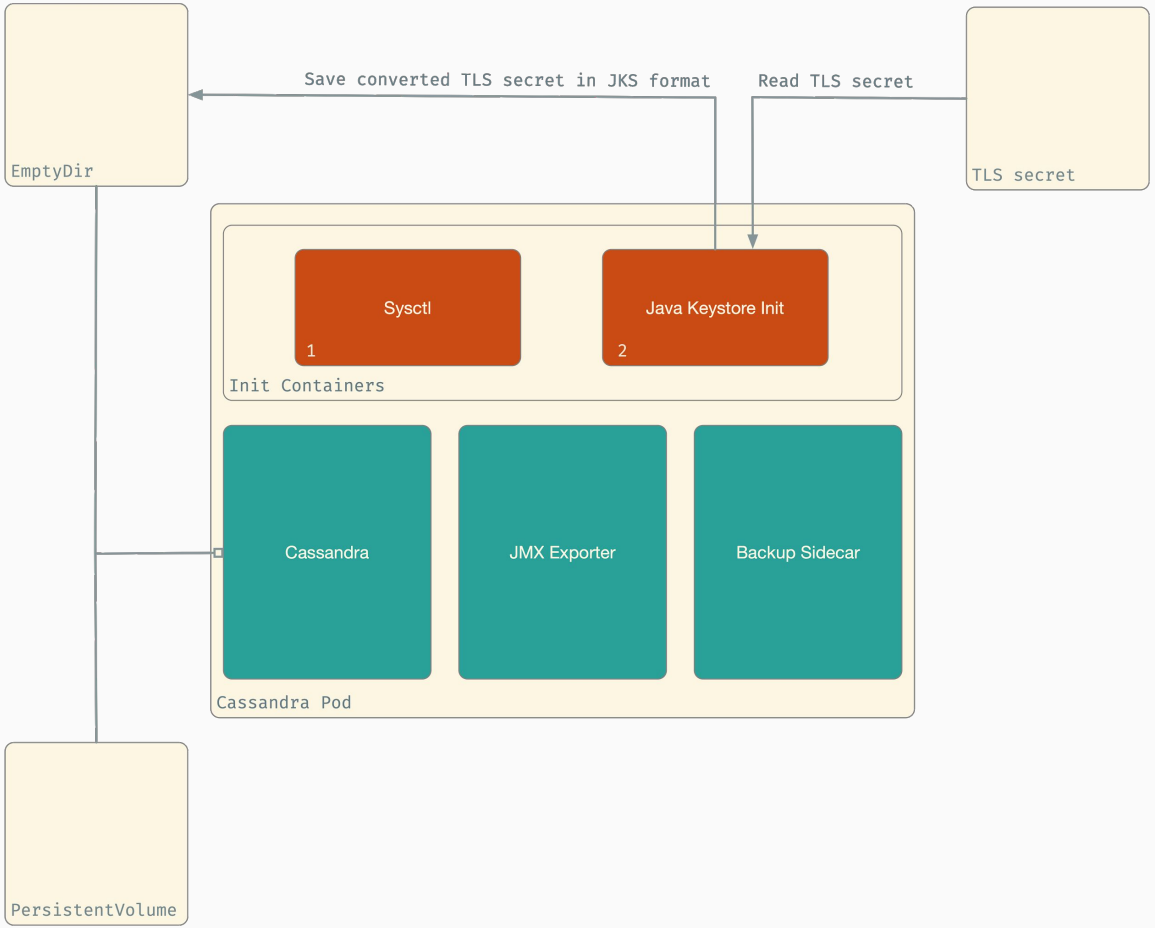
Secret

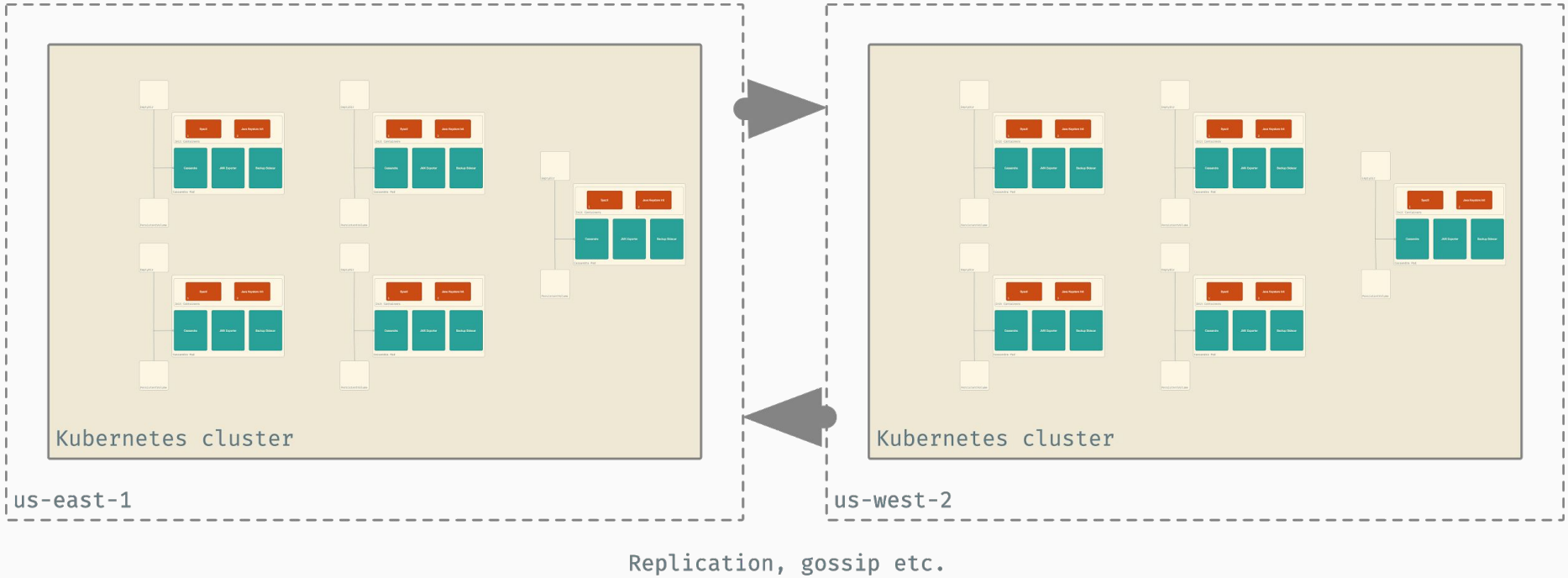
- TLS certs for each of the pods

Pod Anatomy

- Init Containers
  - sysctl
  - TLS Keystore init
- Containers
  - Cassandra
  - JMX Exporter
  - Backup Sidecar




EmptyDir
Save converted TLS secret in JKS format
Read TLS secret
TLS secret
Sysctl
1
Java Keystore Init
2
Init Containers
Cassandra
JMX Exporter
Backup Sidecar
Cassandra Pod
PersistentVolume



Cassandra Cluster in Two Regions

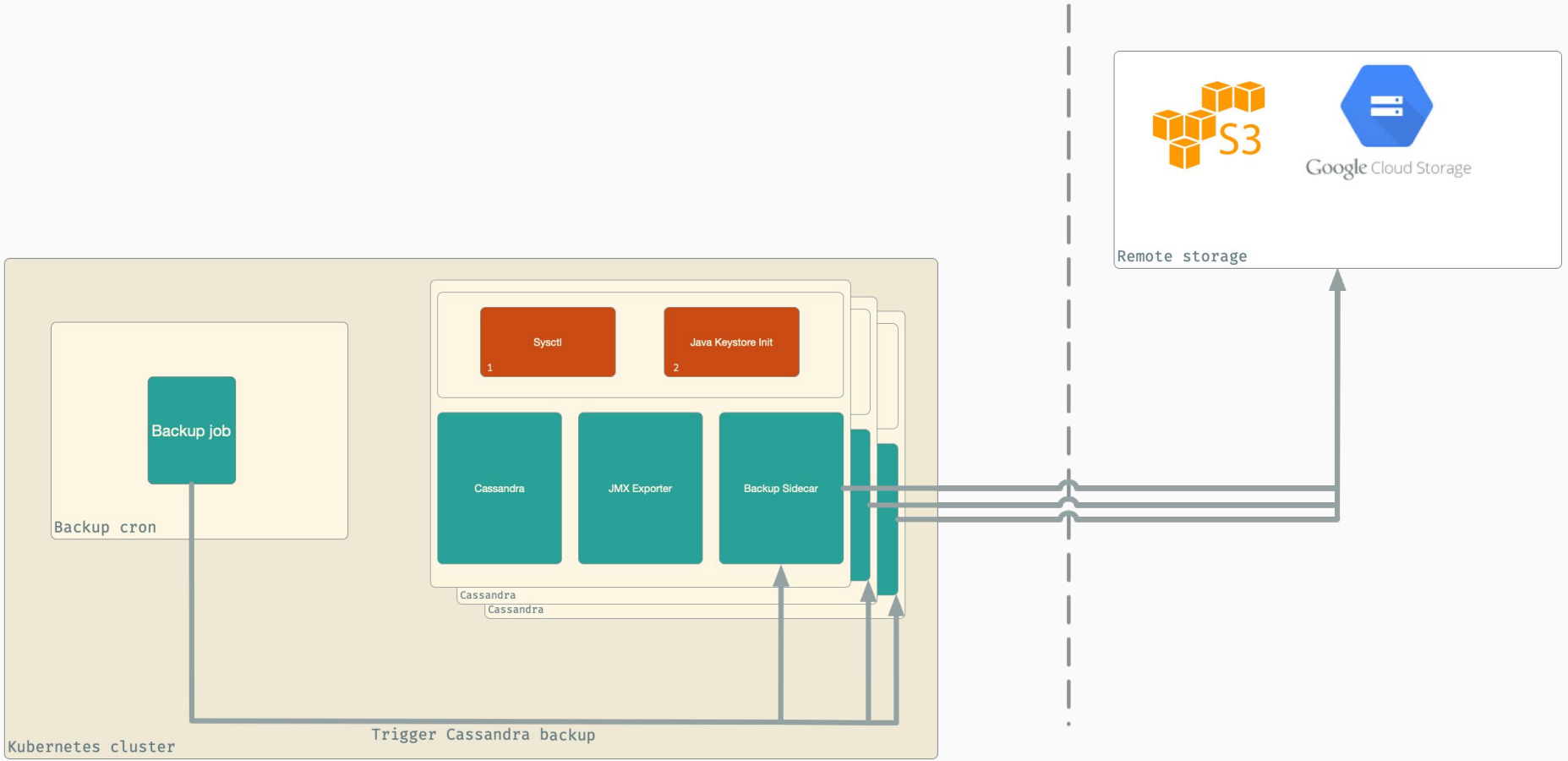
Operations

Scaling + Updates

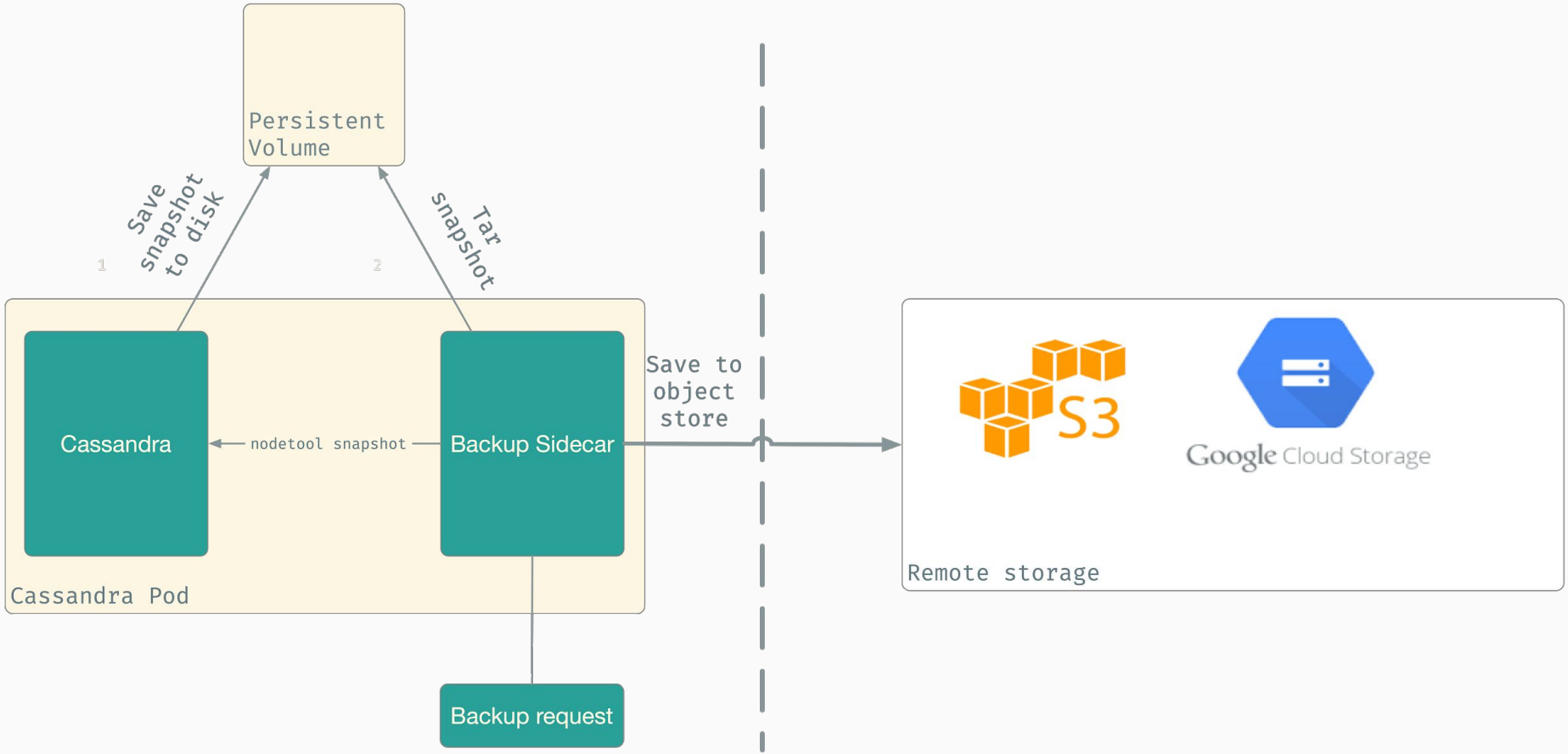
- Increase the replicas in the `StatefulSets`
- Stage updates using partitions

Use Jobs for automation

- Replace config management with Kubernetes jobs

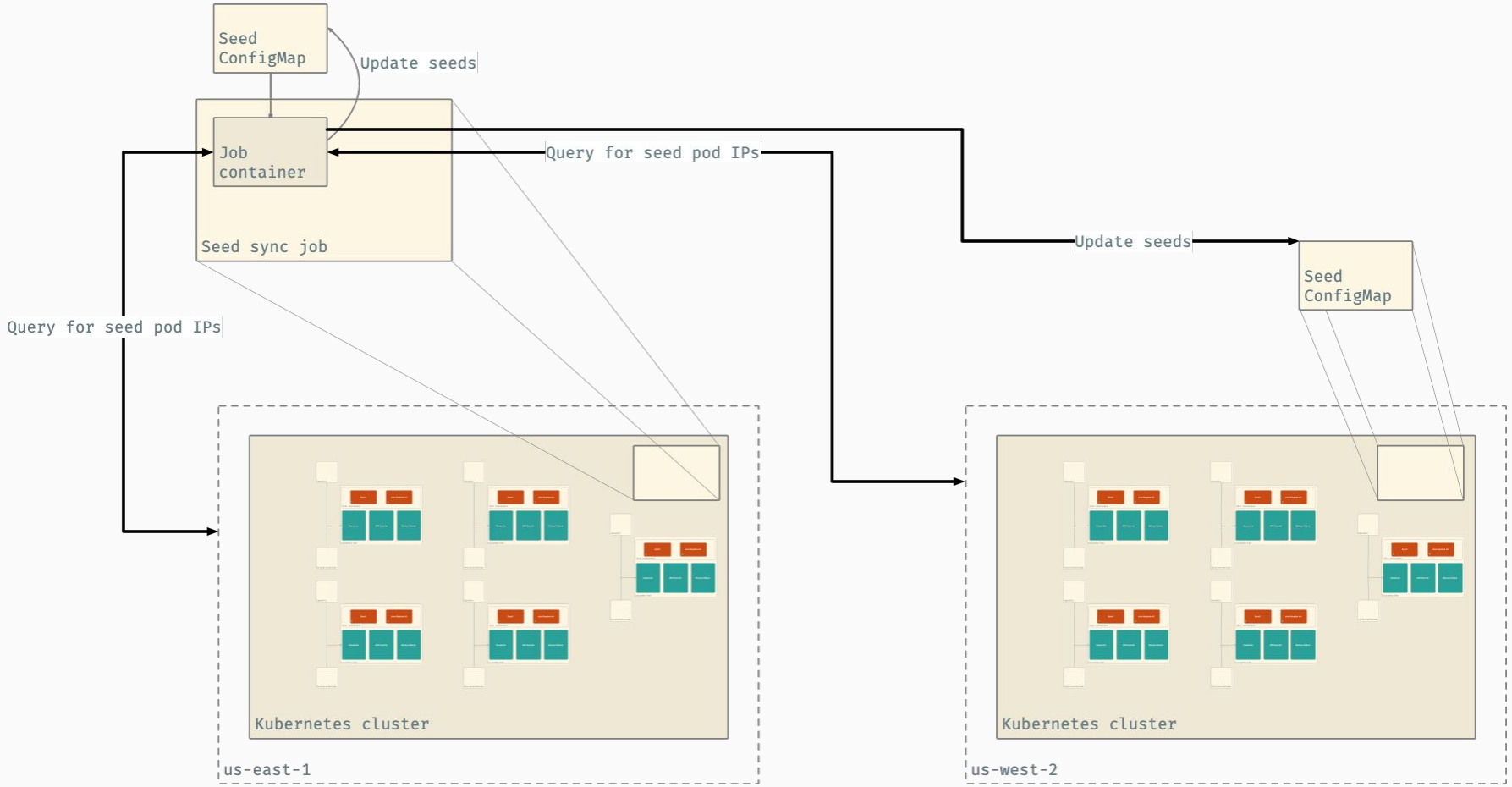

Remote storage
S3
Google Cloud Storage
Backup job
Backup cron
Sysctl
1
Java Keystore Init
2
Cassandra
JMX Exporter
Backup Sidecar
Cassandra
Cassandra
Trigger Cassandra backup
Kubernetes cluster

Backups


Persistent Volume
Save snapshot to disk
1
Tar snapshot
2
Cassandra
nodetool snapshot
Backup Sidecar
Cassandra Pod
Backup request
Save to object store
S3
Google Cloud Storage
Remote storage

Syncing seeds

- Clusters start with a predefined list of seed nodes to contact to learn the topology of the cluster
- Pod restarts change the list
- Solution: use a job
- Dynamic reloading without restarts is coming in Cassandra 4.0



Syncing Seeds

Thanks!

(also, we’re hiring -- drop by the NetApp booth for more info!)