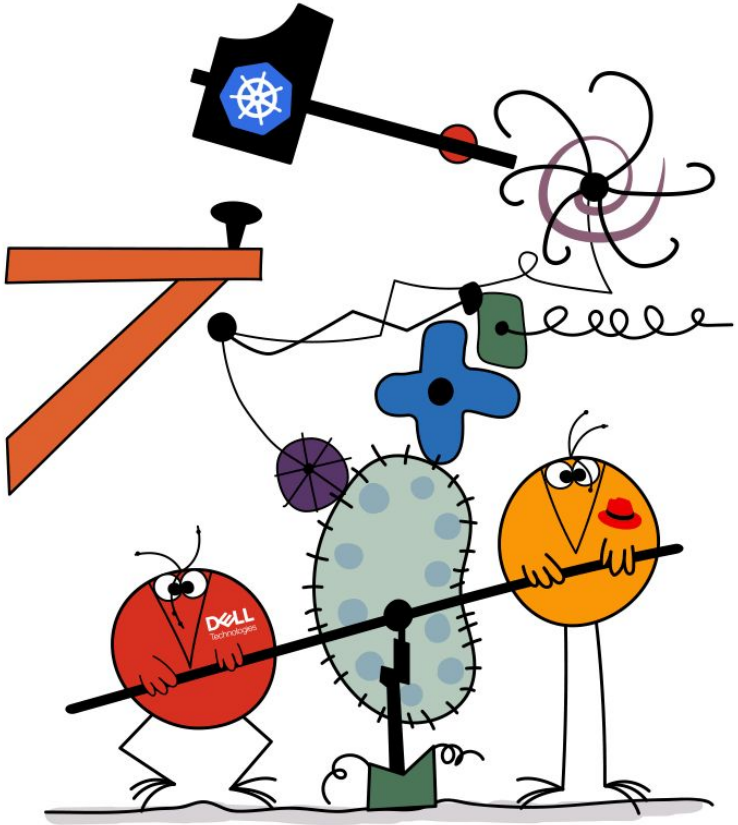
KubeCon | CloudNativeCon

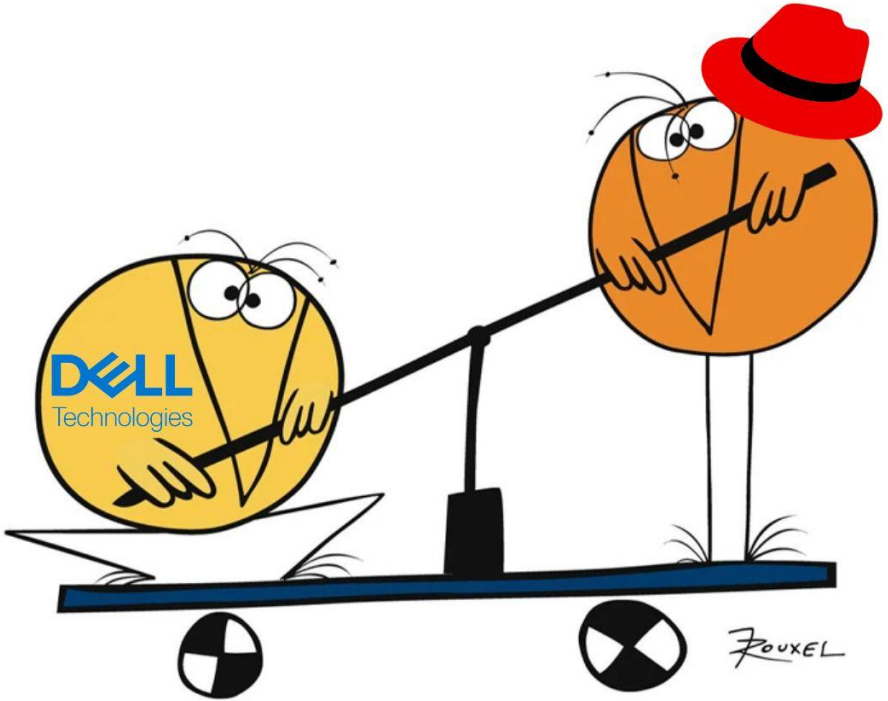
North America 2024

Kubernetes on Multisites

A story about Stateful App, Hybrid clouds, and High Availability



Why make it *simple* when you can make it **complex** ?



Florian Coulombel
Product Management

Jan Šafránek
Software Engineer

Goal of this session

Make k8s on multi-sites concepts
simple when it can be *complex*

Why Kubernetes on multi-sites anyways ?

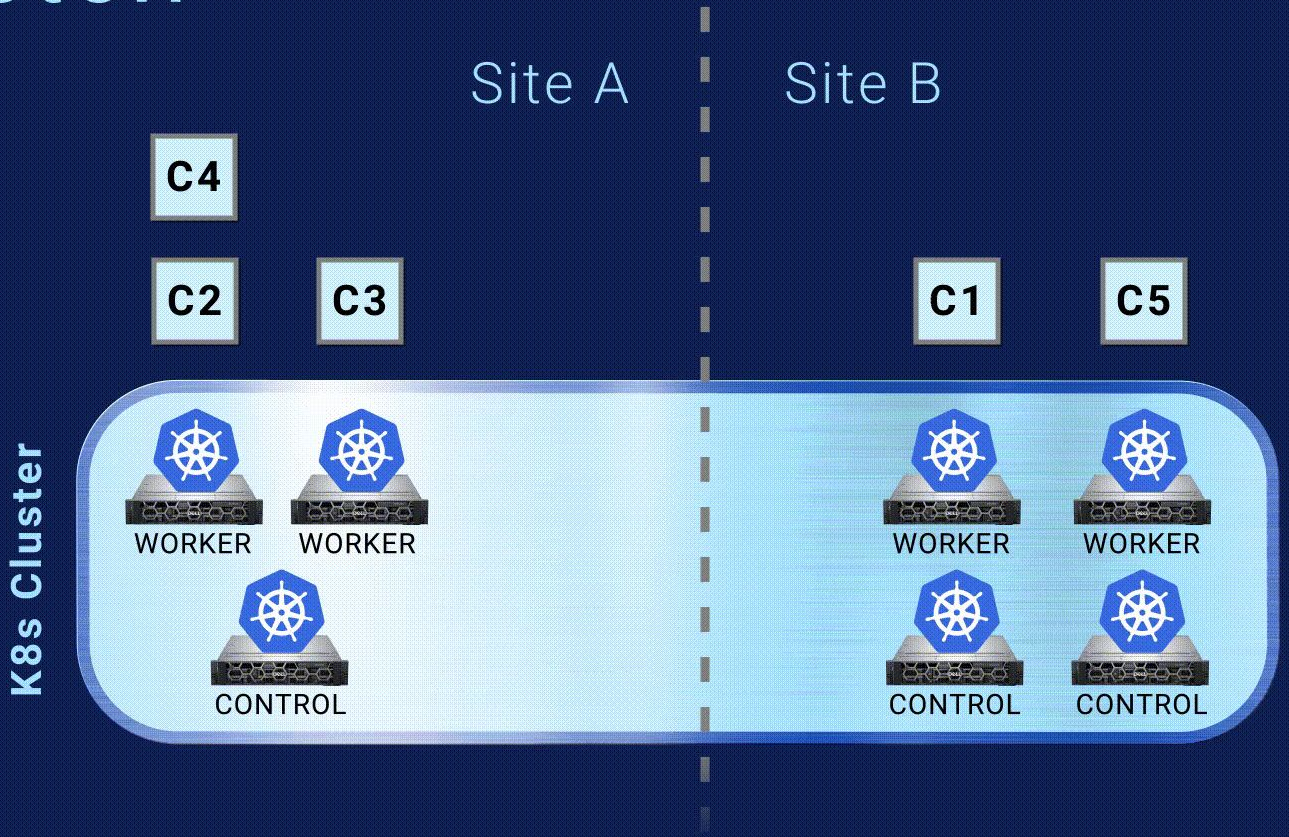
To run highly available & resilient workloads

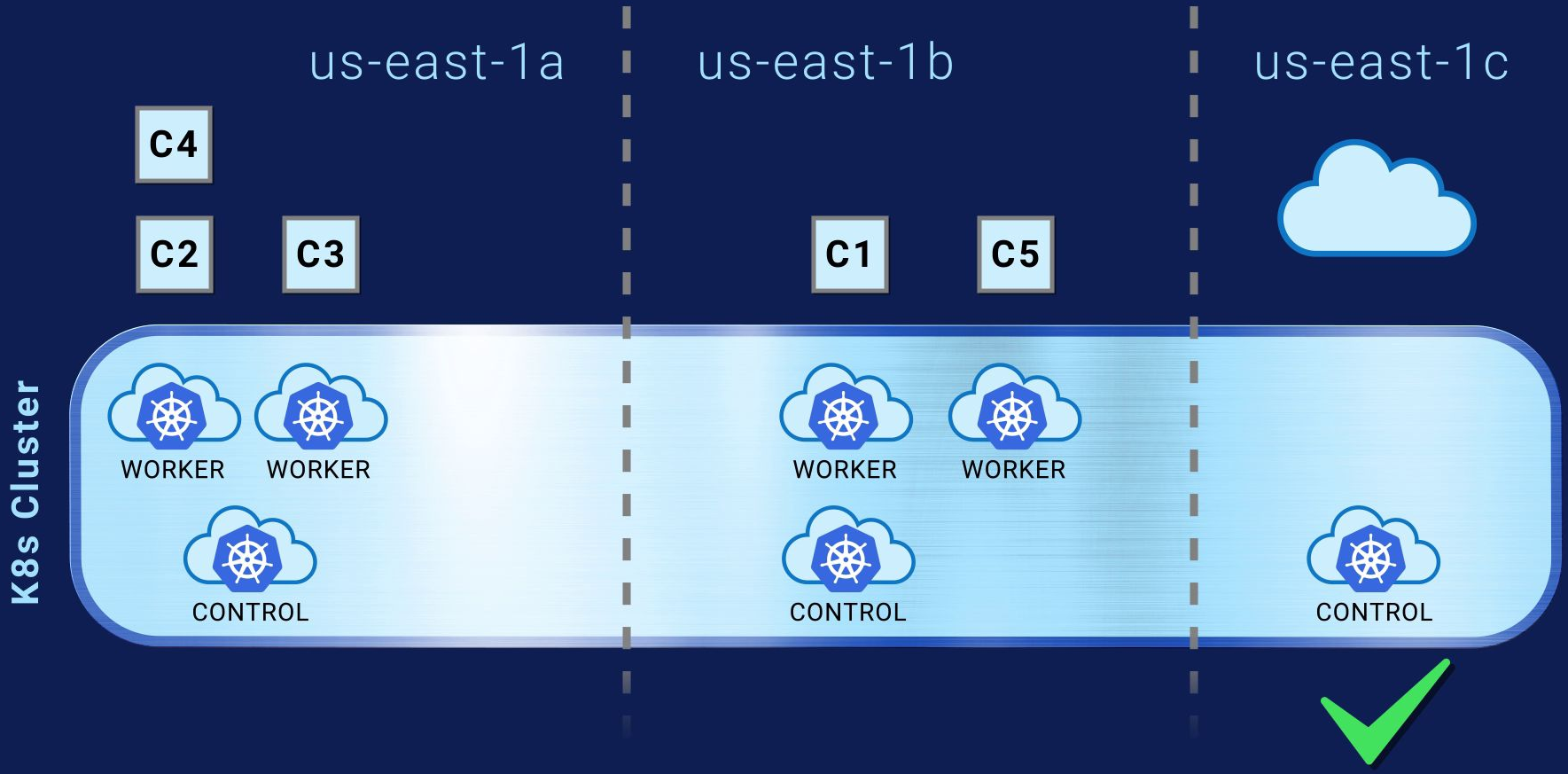
How multi-site enable HA ?

- From the (Stateful) Application PoV
- From the Kubernetes infrastructure PoV
  - Cloud & on-prem

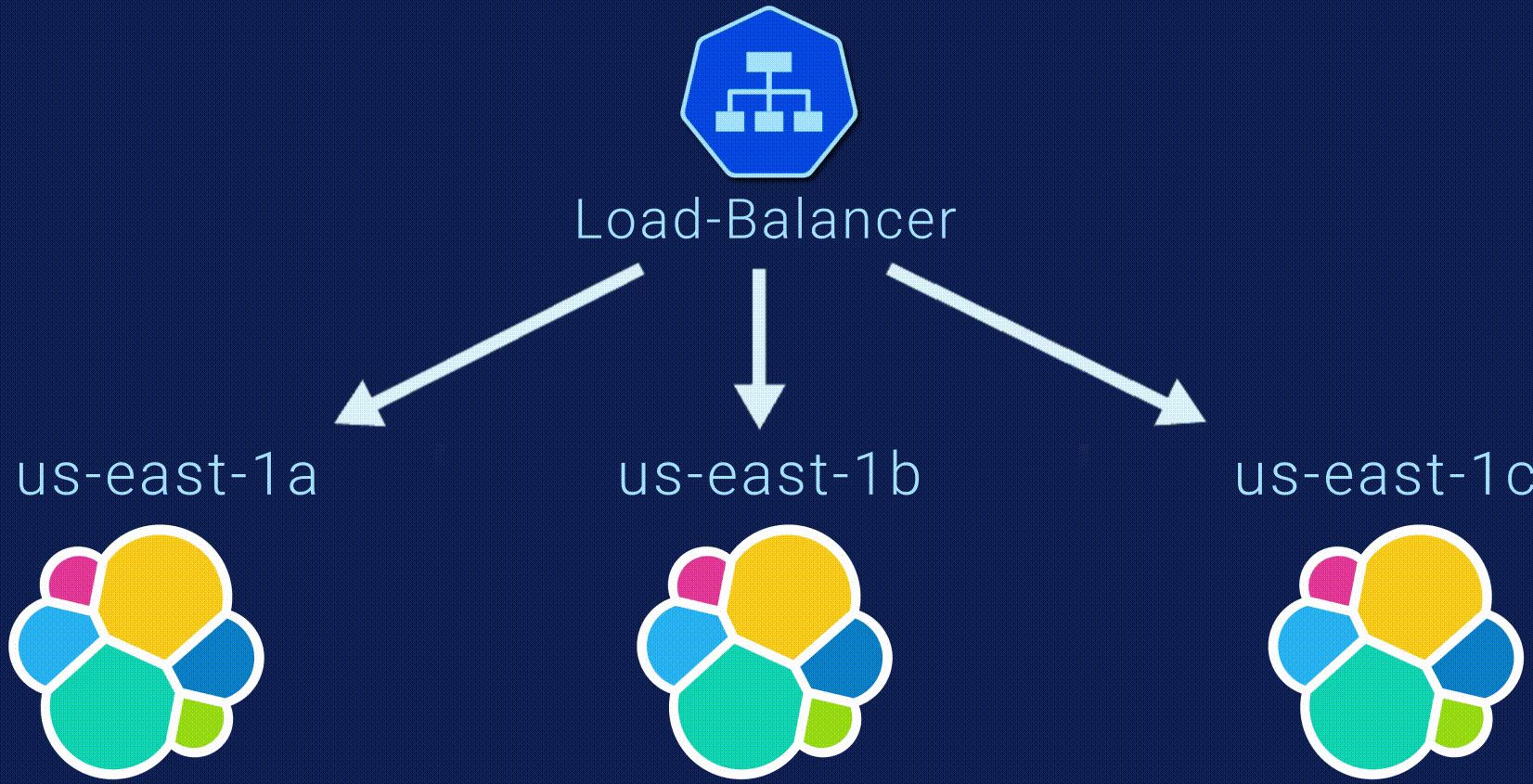
What do we mean by multi-sites ?

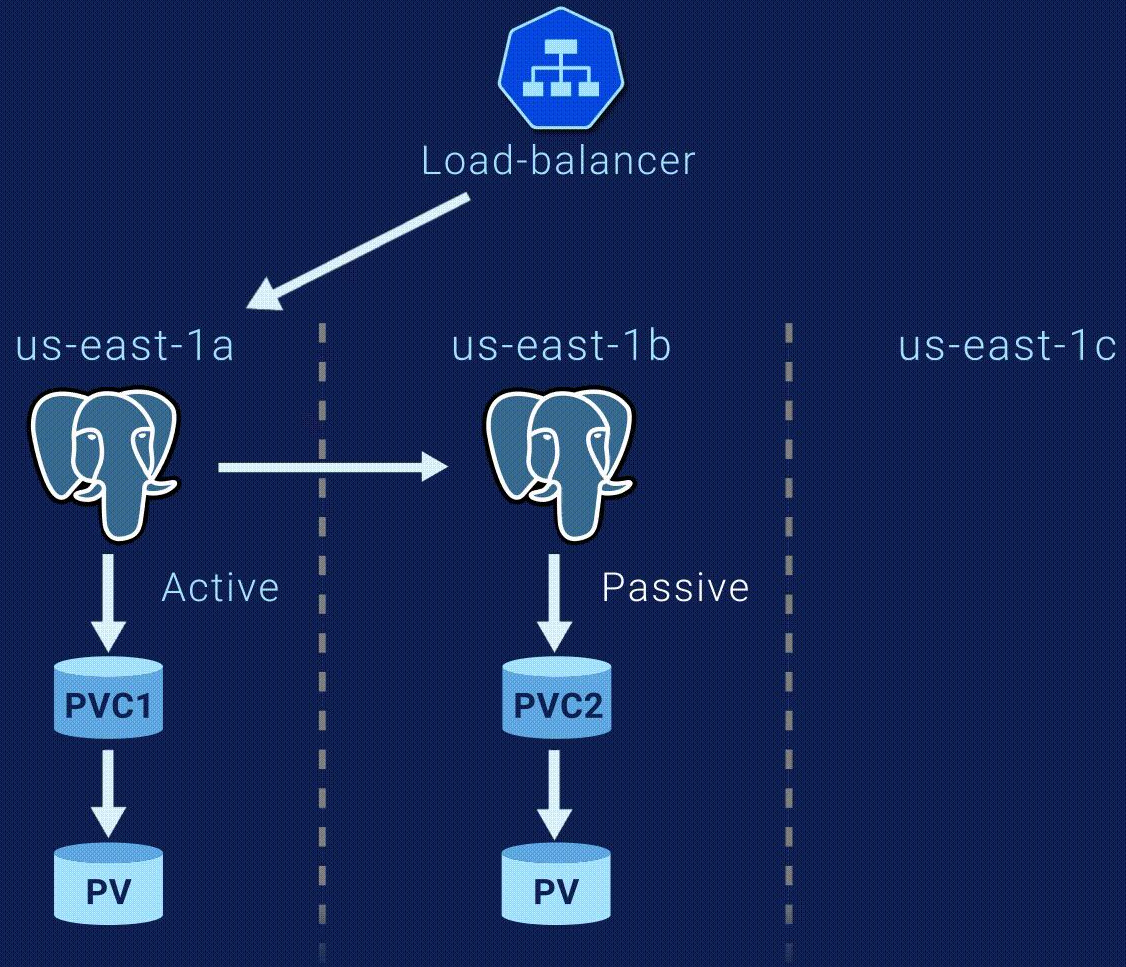
Stretch


Site A
Site B
C4
C2
C3
C1
C5
K8s Cluster
WORKER
WORKER
WORKER
WORKER
CONTROL
CONTROL
CONTROL


us-east-1a
us-east-1b
us-east-1c
C4
C2
C3
C1
C5
K8s Cluster
WORKER
WORKER
WORKER
WORKER
CONTROL
CONTROL
CONTROL

12 factor, treat state as a service


Load-Balancer
us-east-1a
us-east-1b
us-east-1c


Load-balancer
us-east-1a
us-east-1b
us-east-1c
Active
Passive
PVC1
PVC2
PV
PV

Stateful Apps in Kubernetes

- Deployment
  - Designed for Stateless App (random name & scheduling)
  - Can be used with Stateful in certain use-case (e.g. virt-launcher)

- StatefulSet (STS)
  - Designed for Stateful App (stable & ordered)
  - Comes with limitations (no volume expansion, deletion of PVC in beta, node ungraceful shutdown, 1 doesn't necessarily mean main instance)

- WhateverSet
  - Reimplementation of STS via a CRD to serve better the App needs (cnpg, Timescale, advanced-statefulset)

Pod topology

- Pod Topology Spread Constraints

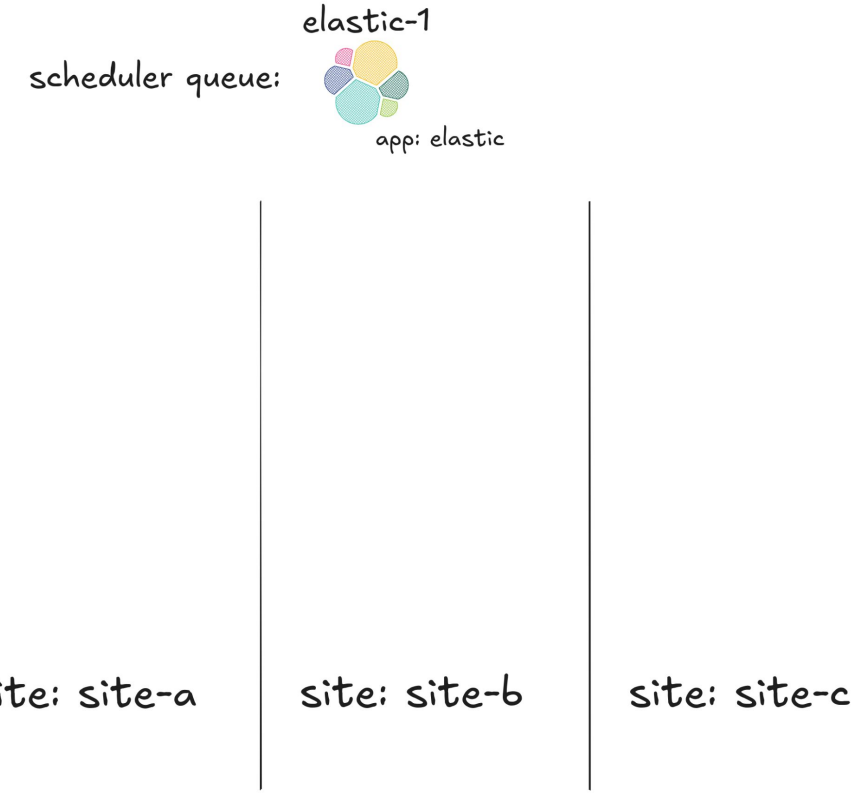
```
kind: Node
metadata:
  name: worker-a-1
  labels:
    site: site-a
...
```

```
kind: Node
metadata:
  name: worker-b-1
  labels:
    site: site-b
...
```

```
kind: Pod
metadata:
  name: elastic-1
  labels:
    app: elastic
spec:
  topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: site
    labelSelector:
      matchLabels:
        app: elastic
```

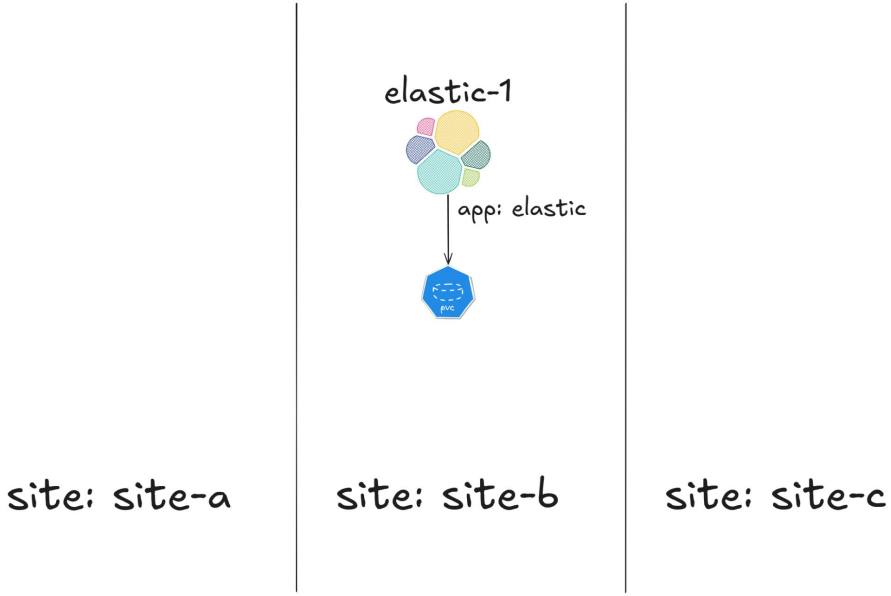
- Descheduler

Schedule on 1st node

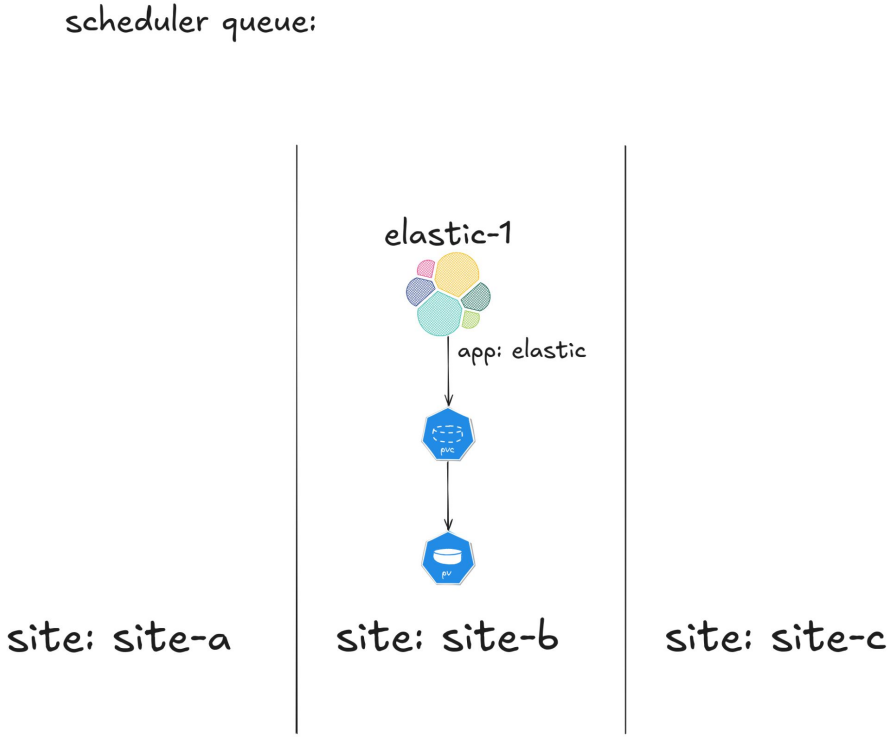


Schedule on 1st node

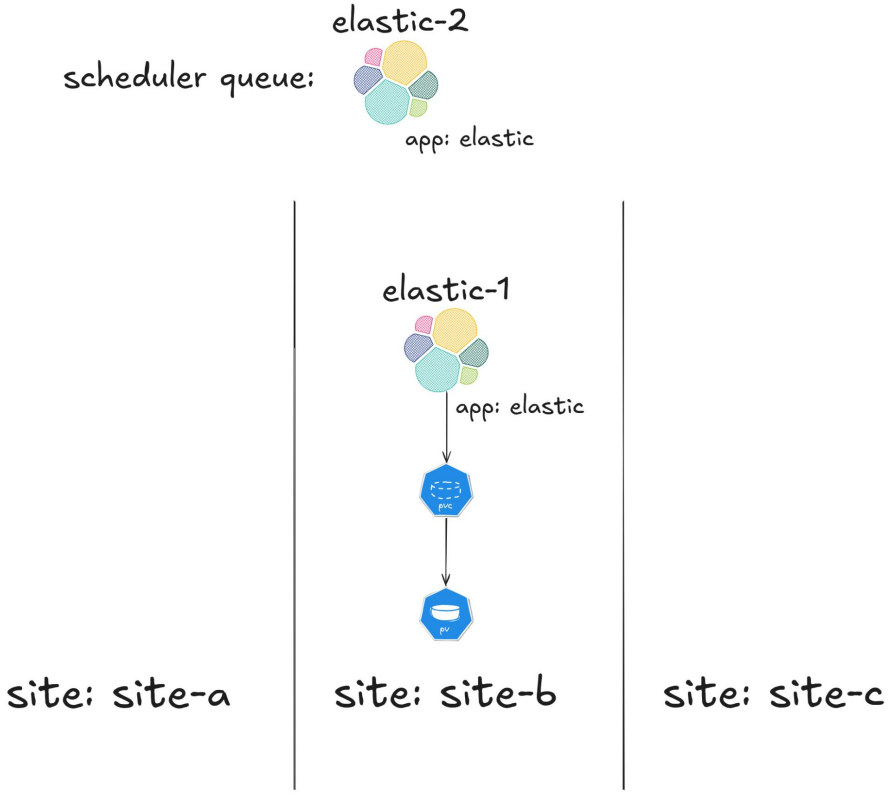
scheduler queue:



Provision PVC/PV

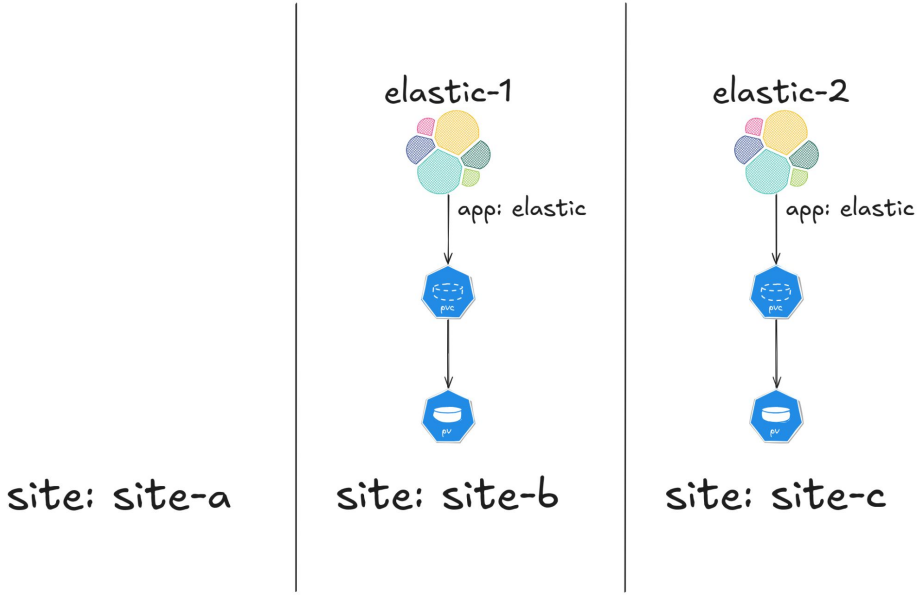


2nd Pod

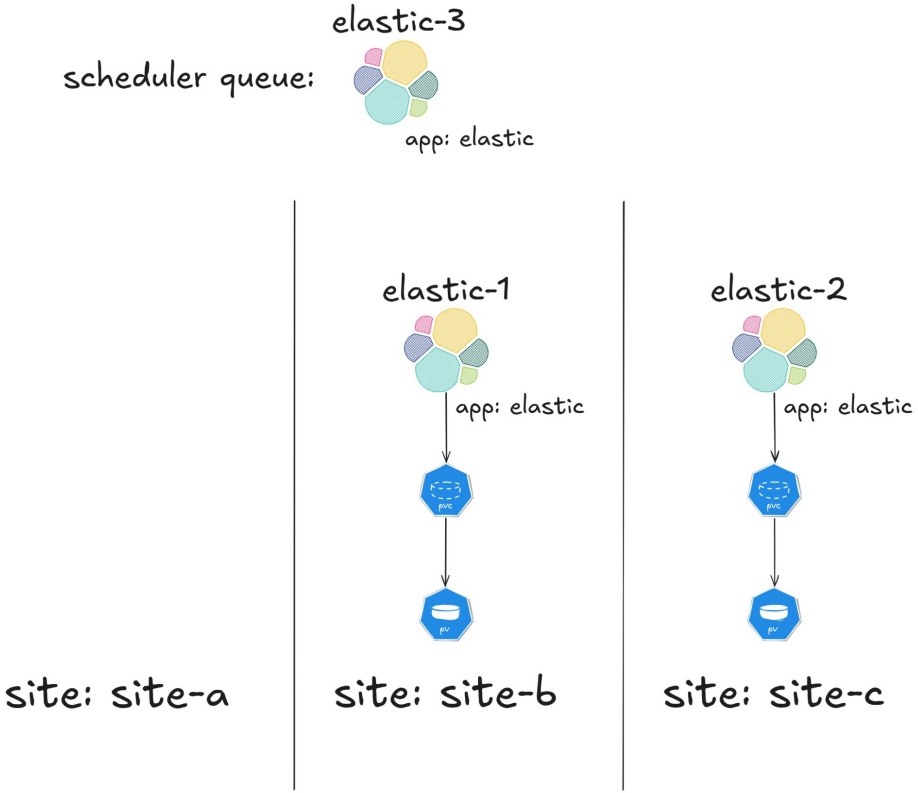

elastic-2
scheduler queue:
app: elastic
elastic-1
app: elastic
pvc
pv
site: site-a
site: site-b
site: site-c

3rd Pod

scheduler queue:



3rd Pod


elastic-3
scheduler queue:
app: elastic
elastic-1
app: elastic
pvc
pv
elastic-2
app: elastic
pvc
pv
site: site-a
site: site-b
site: site-c

3rd Pod

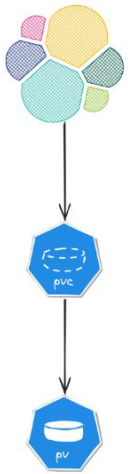
Storage topology

- PersistentVolume (PV) node affinity.

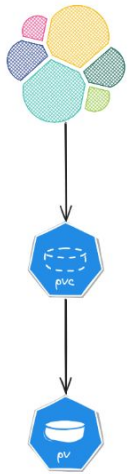
```
kind: Node
metadata:
  name: worker-a-1
  labels:
    site: site-a
...
```

```
kind: PersistentVolume
metadata:
  name: vol-42
spec:
  ...
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: site
          operator: In
          values:
          - site-a
```

Scale sites



Site A

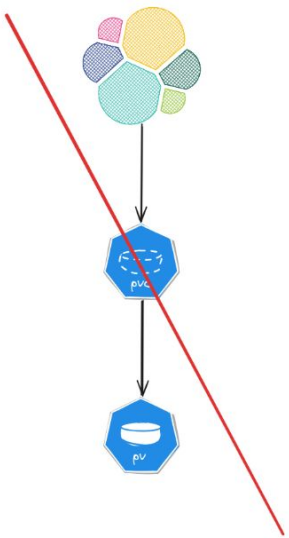


Site B

Site C

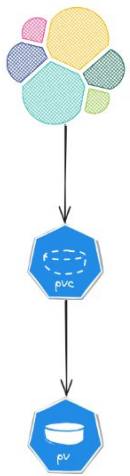
Site D

Site failure



Site A

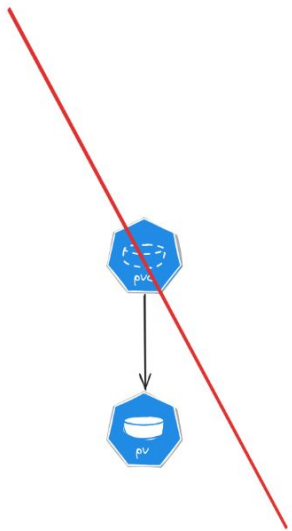
Site B



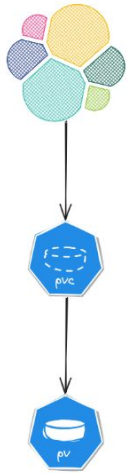
Site C

Site D

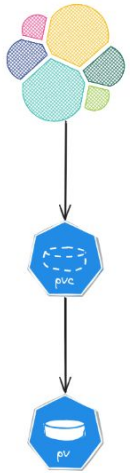
Storage topology



Site A



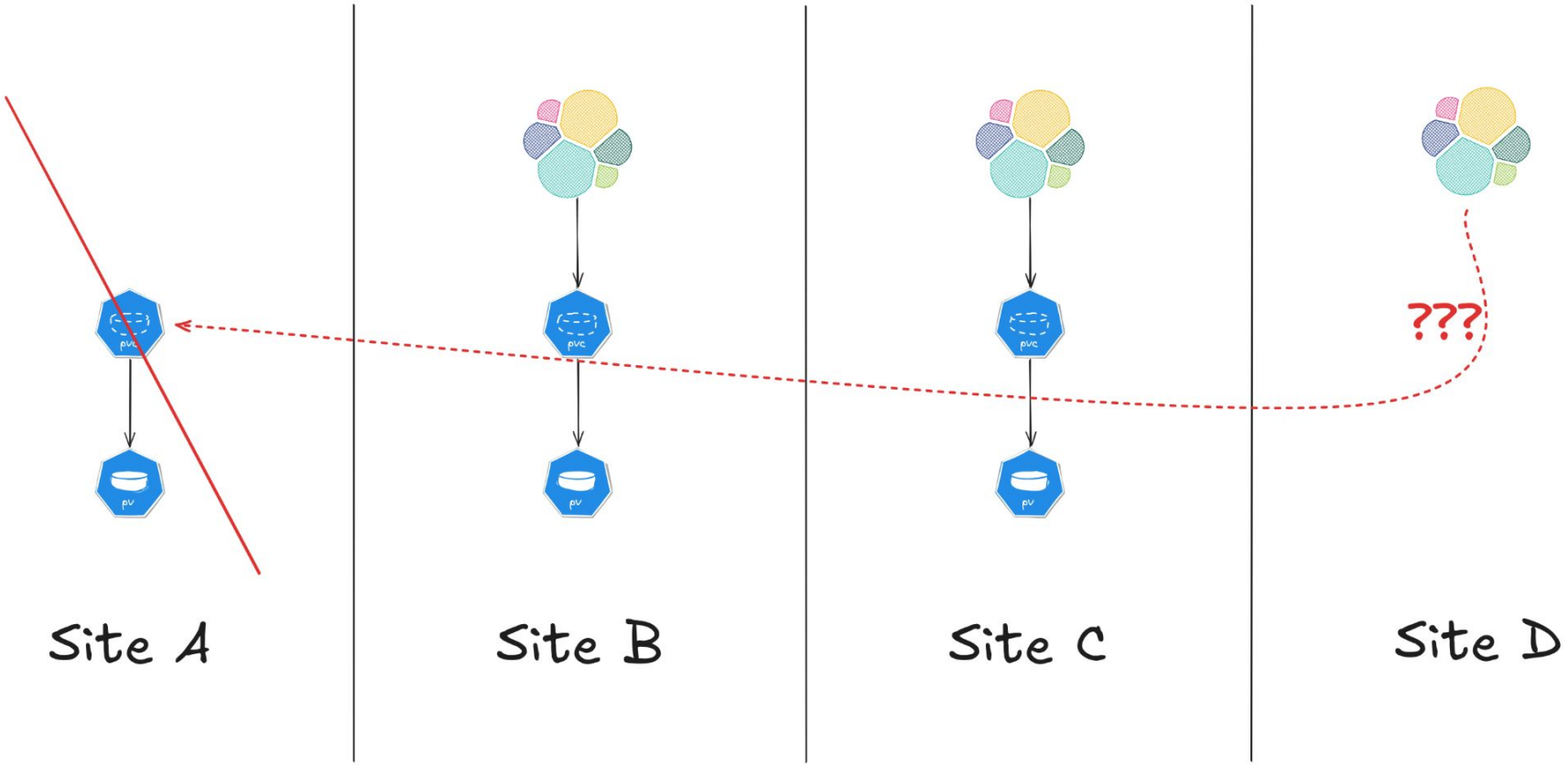
Site B

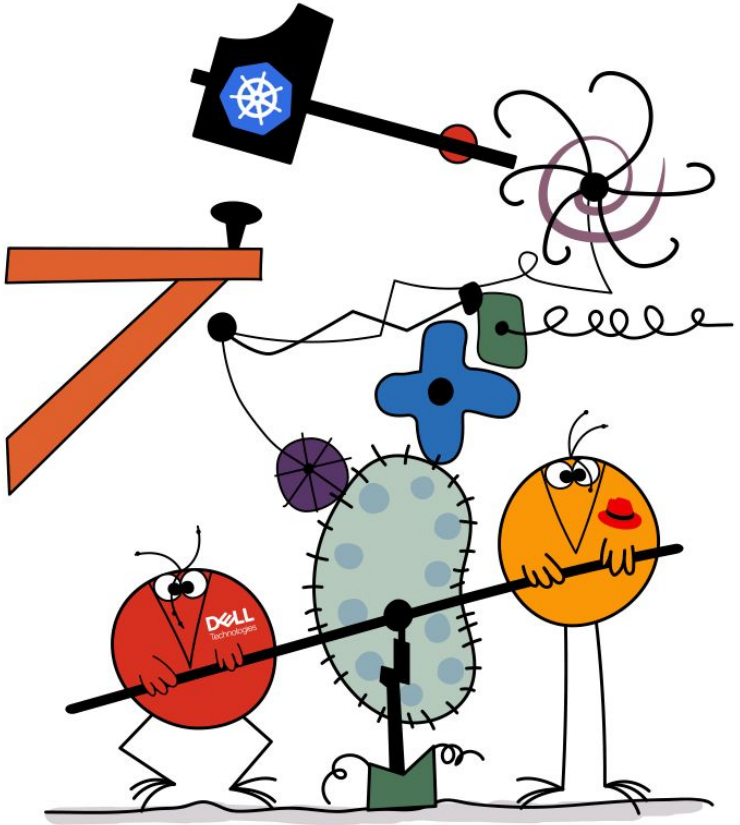


Site C

Site D

Storage topology





Is it **simple** now ?

Take away #1

3 sites with use of Kubernetes built-ins (topology, node affinity, etc.)

Take away #2

Use app HA as much as you can

Take away #3

Know your app constraints

What if my App can not carry the HA ?

~~2004~~2024 the year of Virtualization

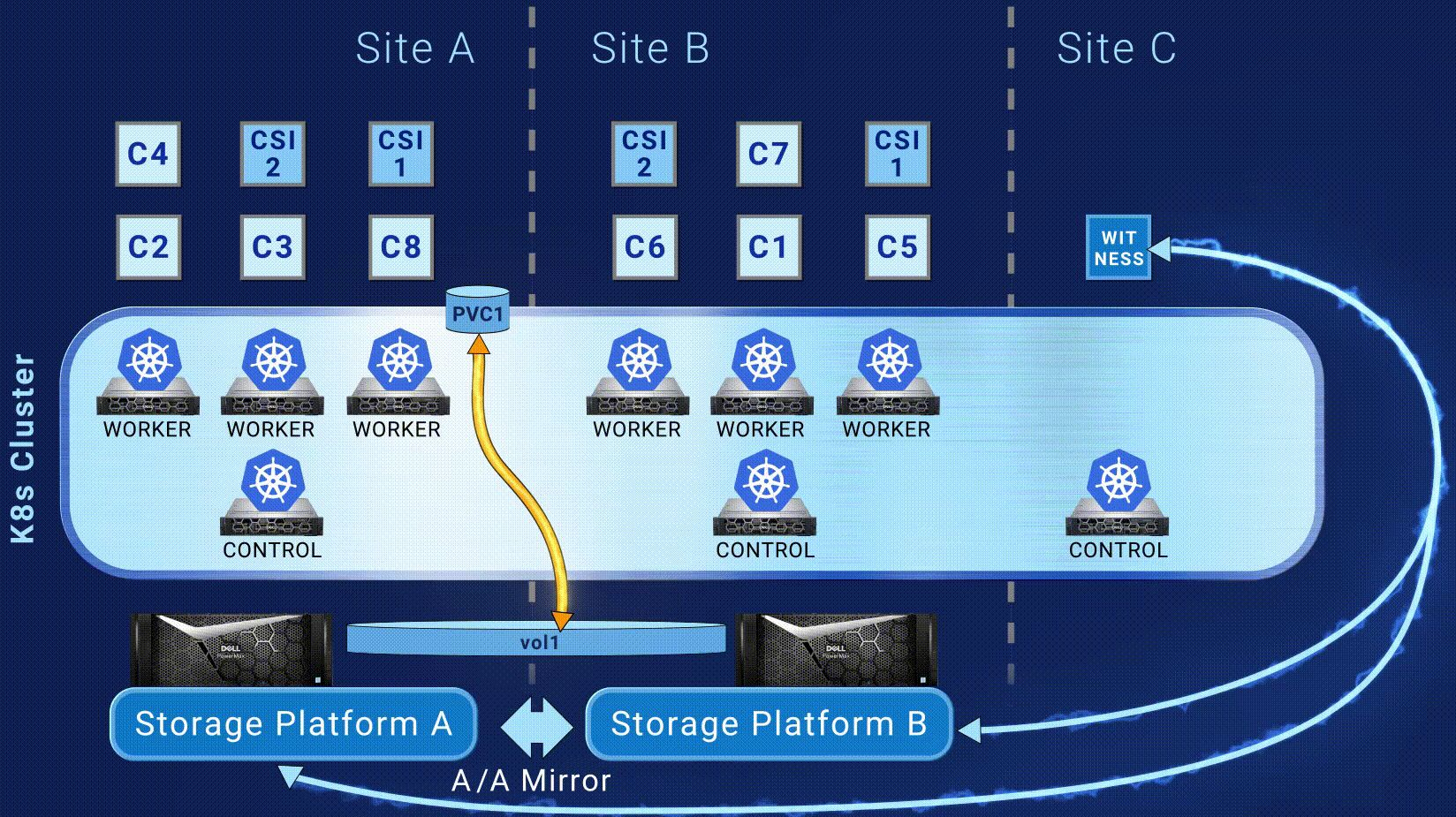
How virtualization is different from app

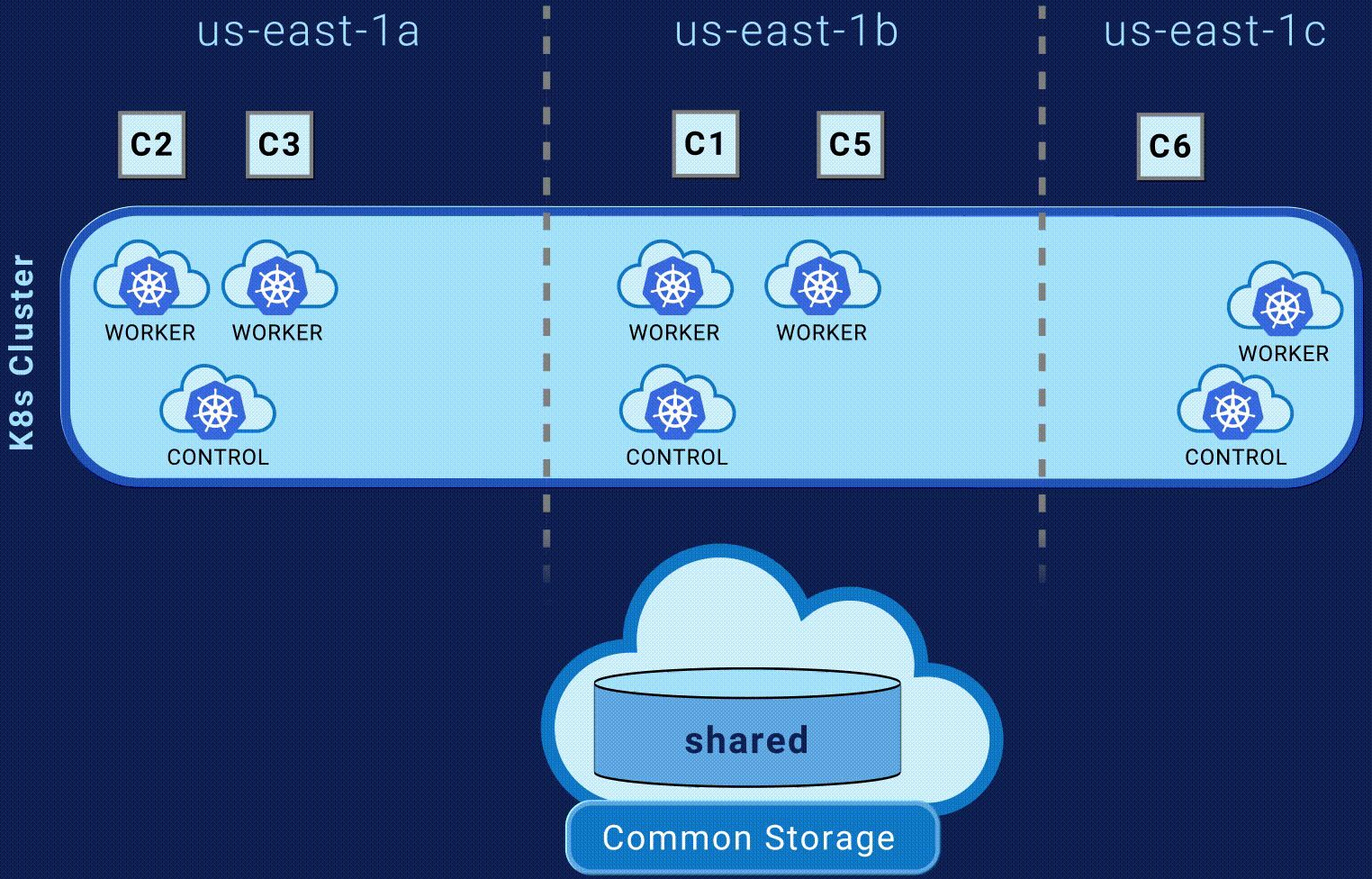
1. Virtualization will not replicate data
   a. Storage infrastructure needs to be reliable
2. Scale
   a. 1 Block device > Many VMs
   b. 1 VM > several block devices
3. Live Migration
   a. ReadWriteMany mandatory
4. Identify your constraints
   a. Recovery Point Objective (RPO)
   b. Recovery Time Objective (RTO)

Metro Disaster Recovery

RPO = 0 / RTO = 0

Stretched cluster: Metro / Active-Active replication




us-east-1a
us-east-1b
us-east-1c
C2
C3
C1
C5
C6
K8s Cluster
WORKER
WORKER
WORKER
WORKER
WORKER
CONTROL
CONTROL
CONTROL
shared
Common Storage

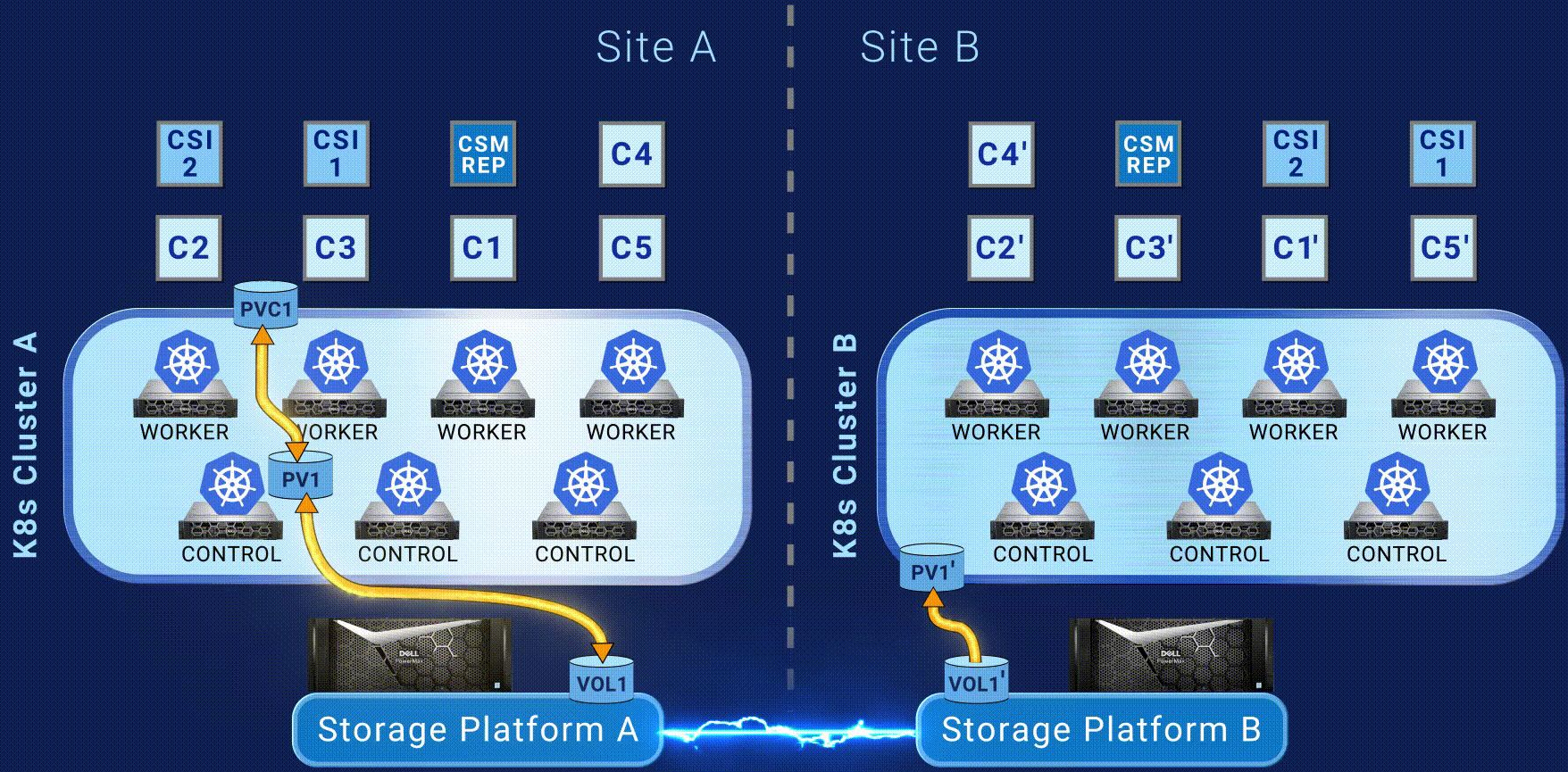
Regional Disaster Recovery

1. Synchronous replication
    a. RPO = 0
    b. RTO > 5m

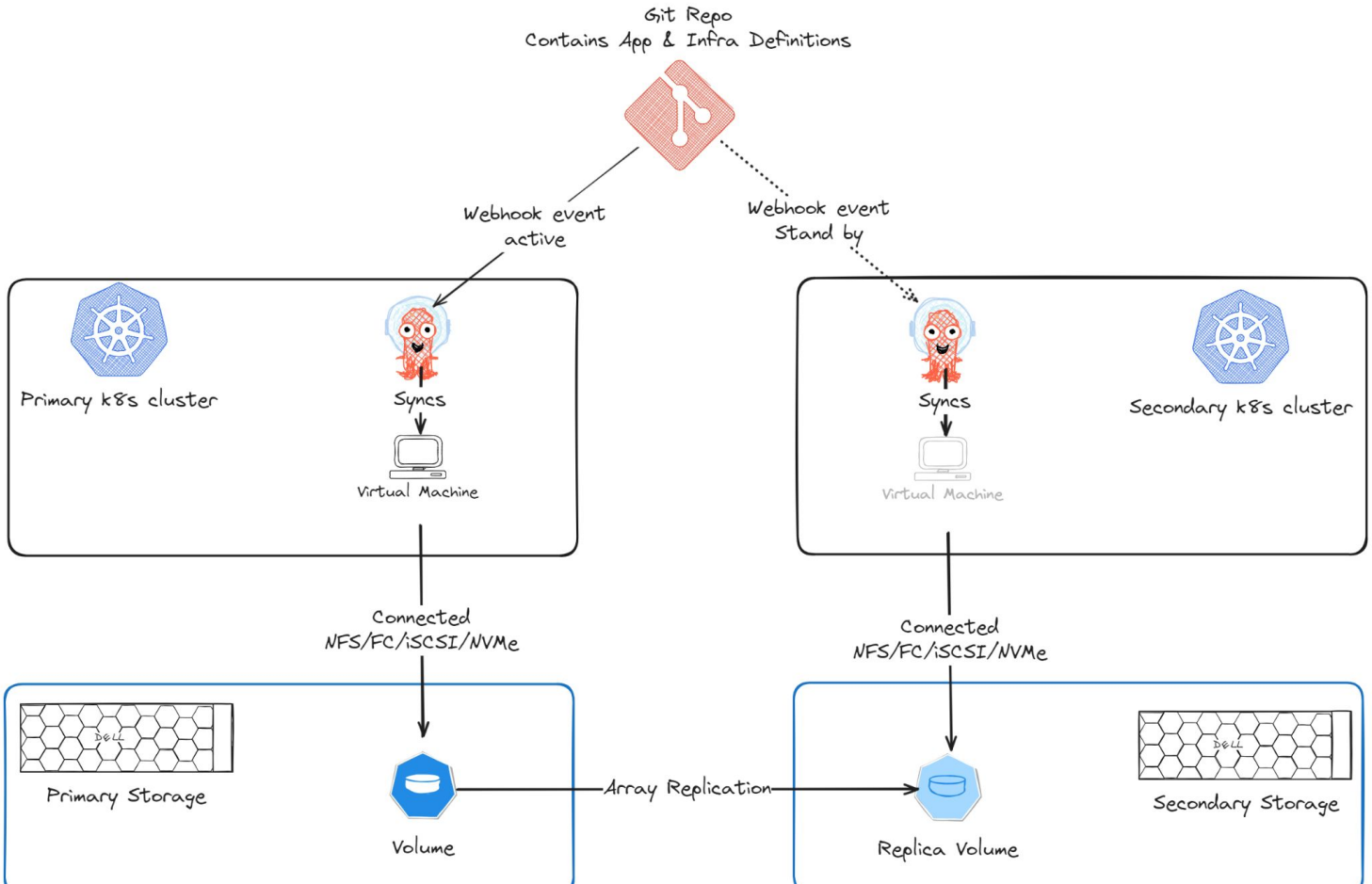
2. Asynchronous replication
    a. RPO > 15s
    b. RTO > 1h

3. Backup restore
    a. RPO > 1h
    b. RTO < 1d

Severed clusters: Storage replication



ArgoCD VM failover

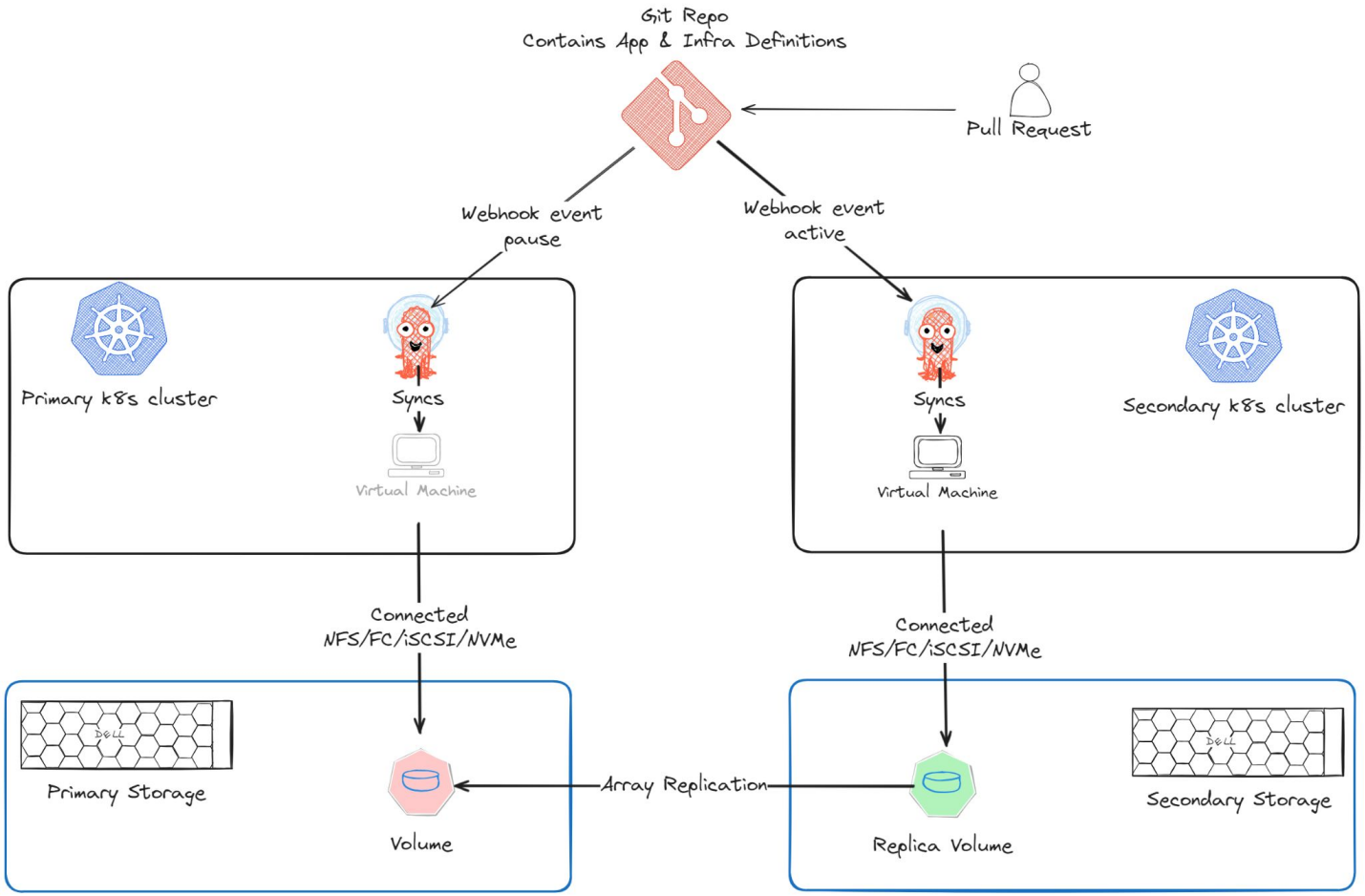


ArgoCD VM failover


```
patches:
  - patch: |
      - op: replace
        path: /spec/running
        value: true
    target:
      kind: VirtualMachine

  - patch: |
      - op: replace
        path: /spec/action
        value: FAILOVER
    target:
      kind: DellReplicationGroup
```

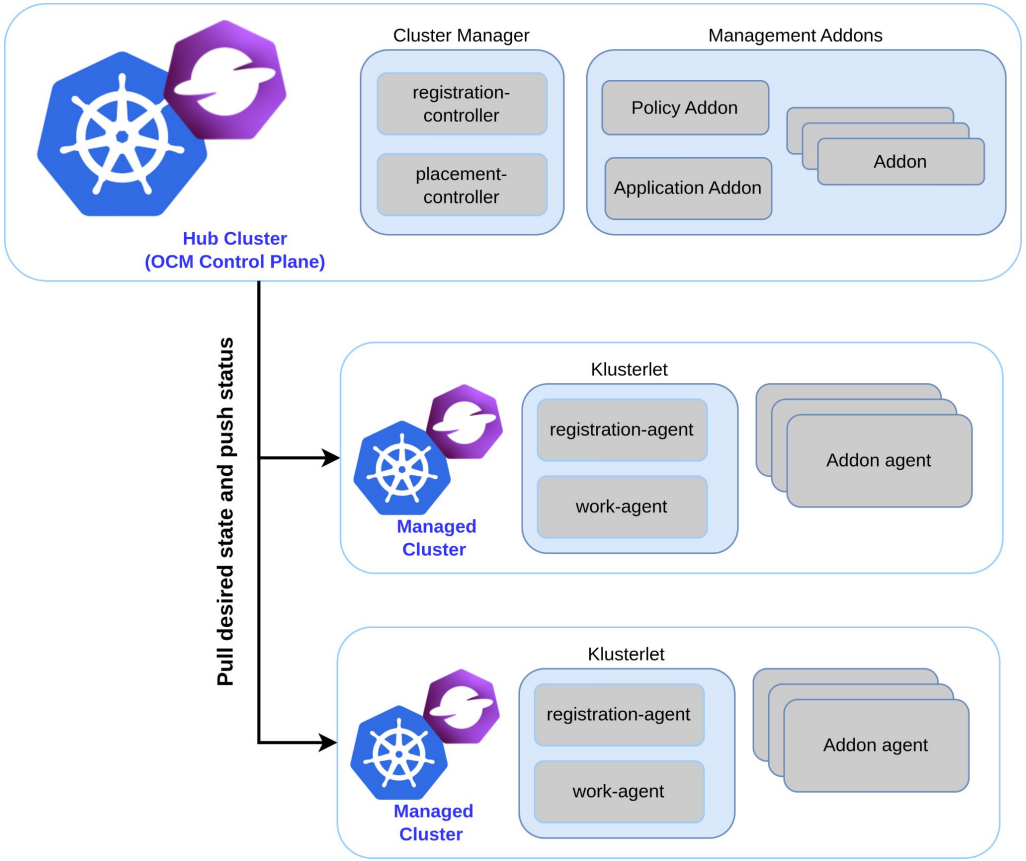
ArgoCD VM failover



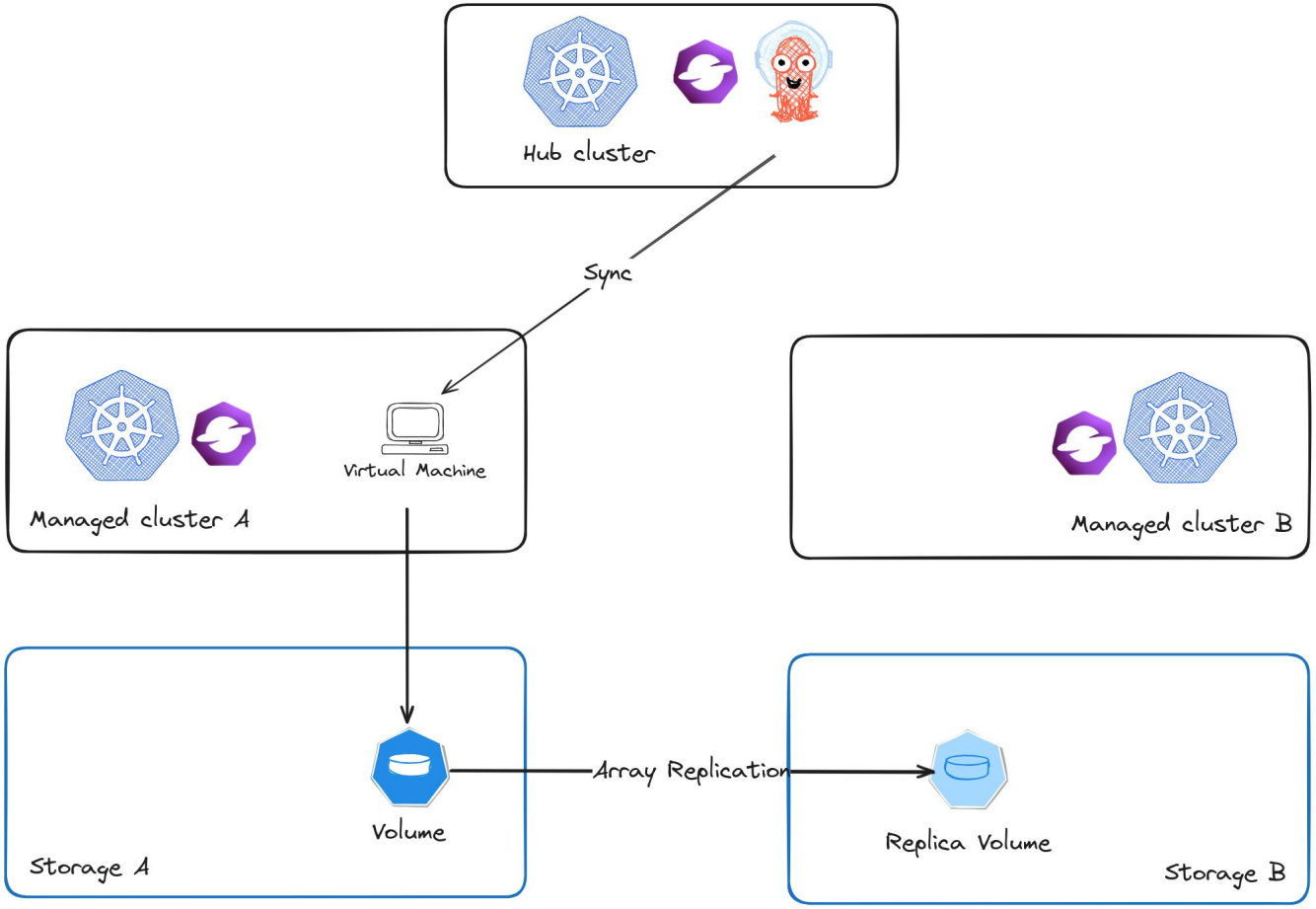
Open Cluster Management

- open-cluster-management.io
  - CNCF sandbox project
  - Integration with ArgoCD
- RamenDR
  - Open Cluster Management addon
  - Data replication

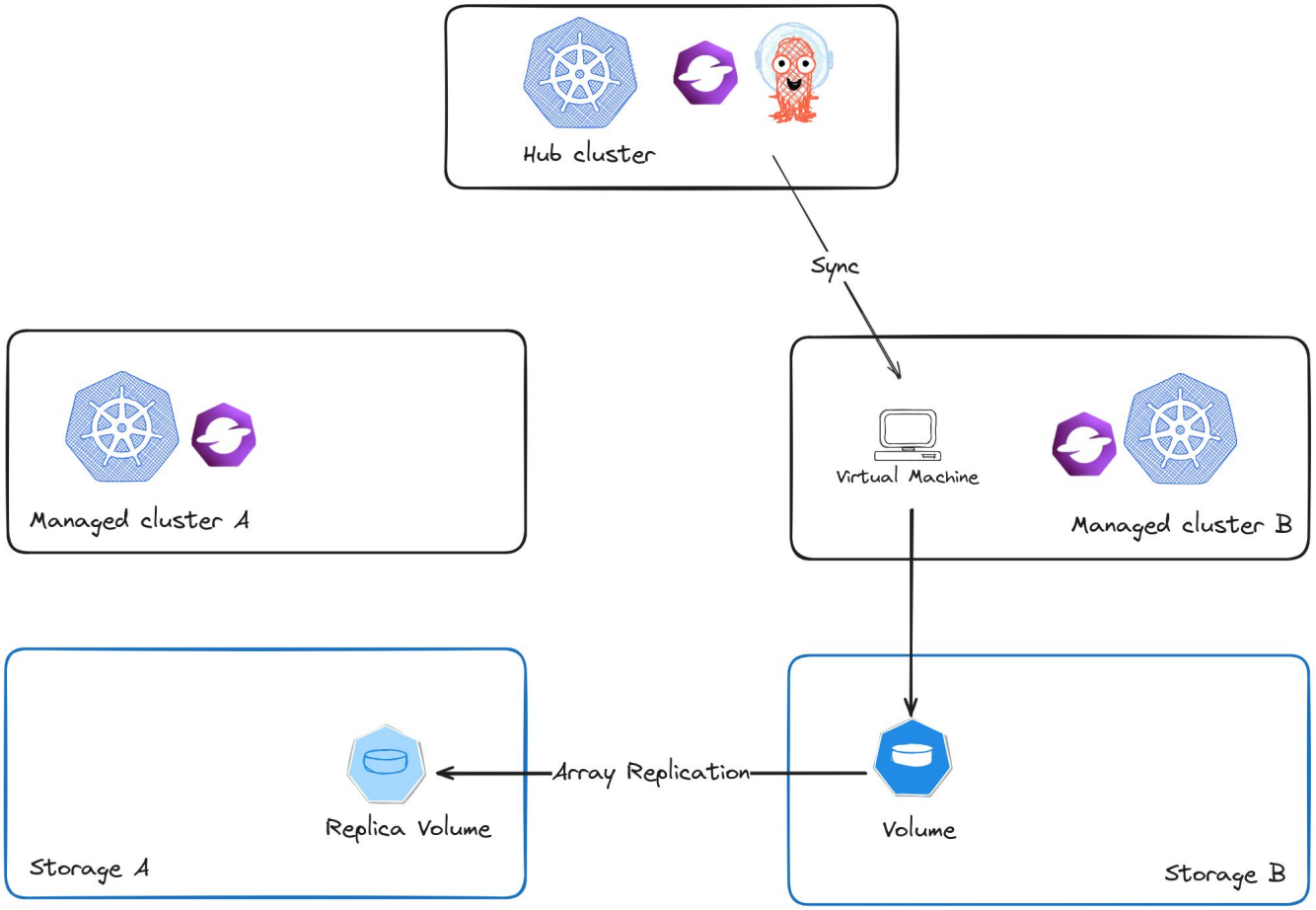
Open Cluster Management

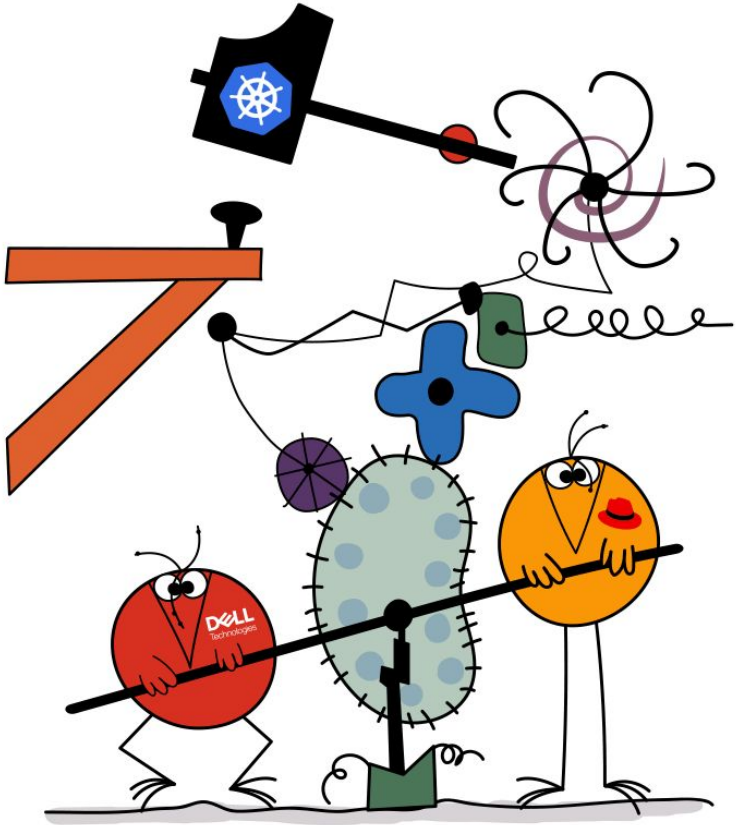


Open Cluster Management



Open Cluster Management





Is it **simple** now ?

Take away #1

Multi-sites with stretched or severed clusters

Take away #2

Use app HA as much as you can

Take away #3

No Kubernetes built-in for severed cluster
Somebody (you?) have to do the orchestration

Take away #0

If you need resilient storage do your tests

Links to other sessions on the same topic

- Replay the session
  - SIG-Multicluster : https://sched.co/1hove
  - LiStatefulSet: https://sched.co/1i7n9

- Past sessions:
  - We Tested and Compared 6 Database Operators. The Results are In! @ KubeCon EU 2024