What you need to know before using Local PersistentVolumes

Sébastien Guilloux

KubeCon + CloudNativeCon North America 2021

Sébastien Guilloux (@_sebgl)

Software Engineer @Elastic

github.com/elastic/cloud-on-k8s
K8s operator for the Elastic stack

elastic

Agenda

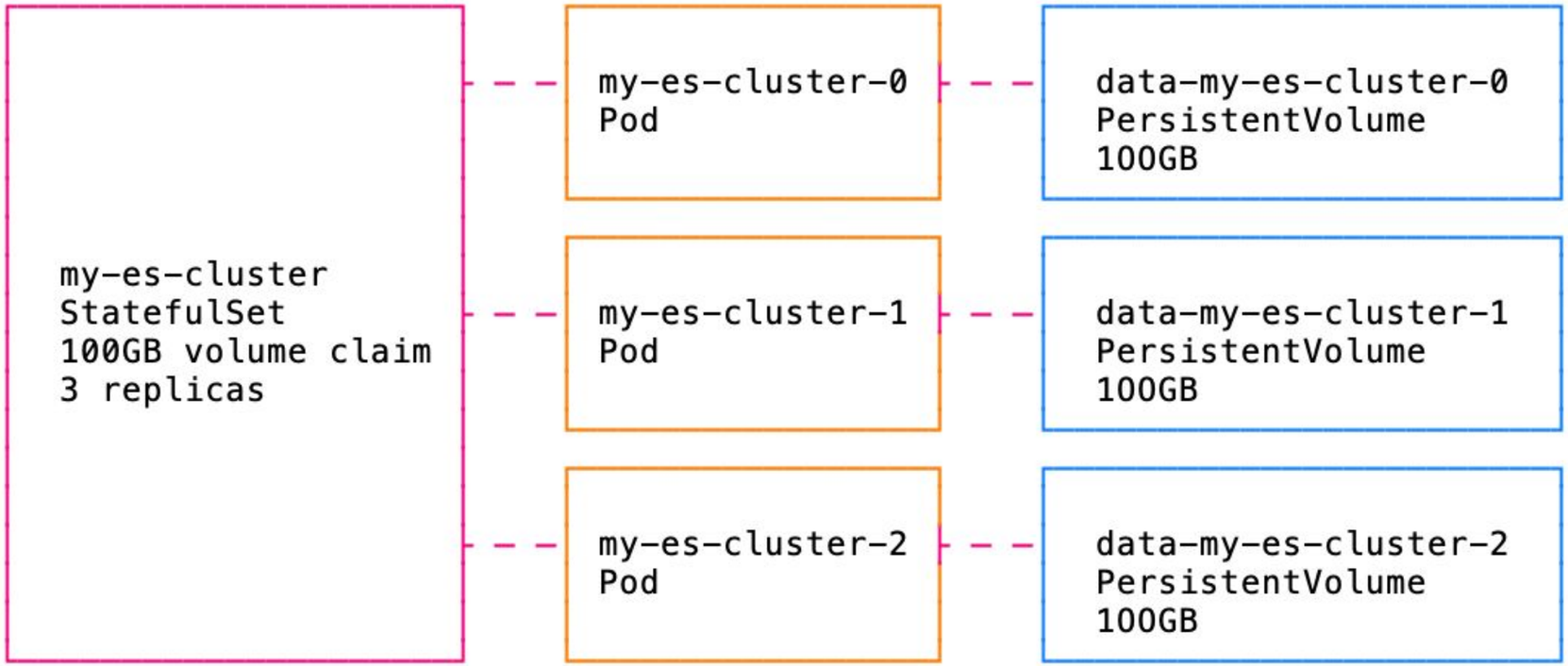
How things work

Provisioning local volumes

Operational gotchas

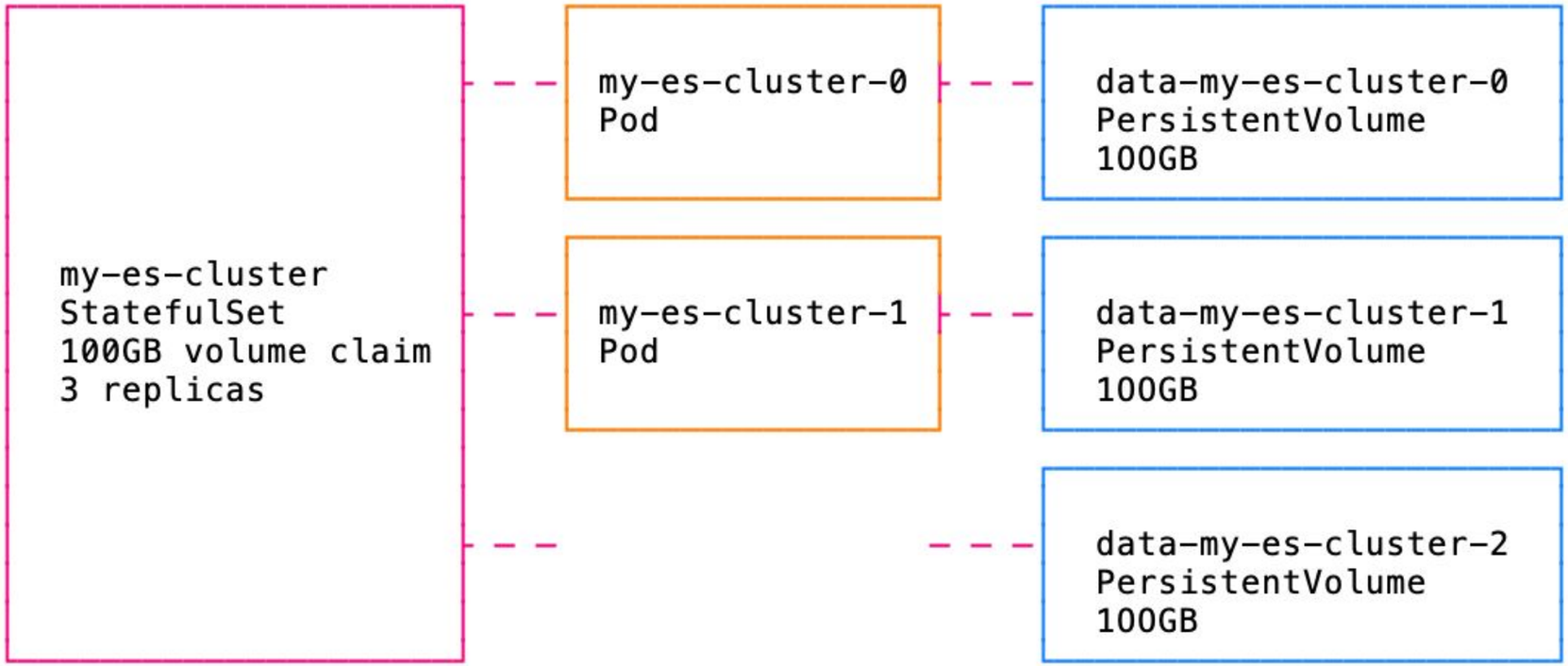
How things work

PersistentVolumes?



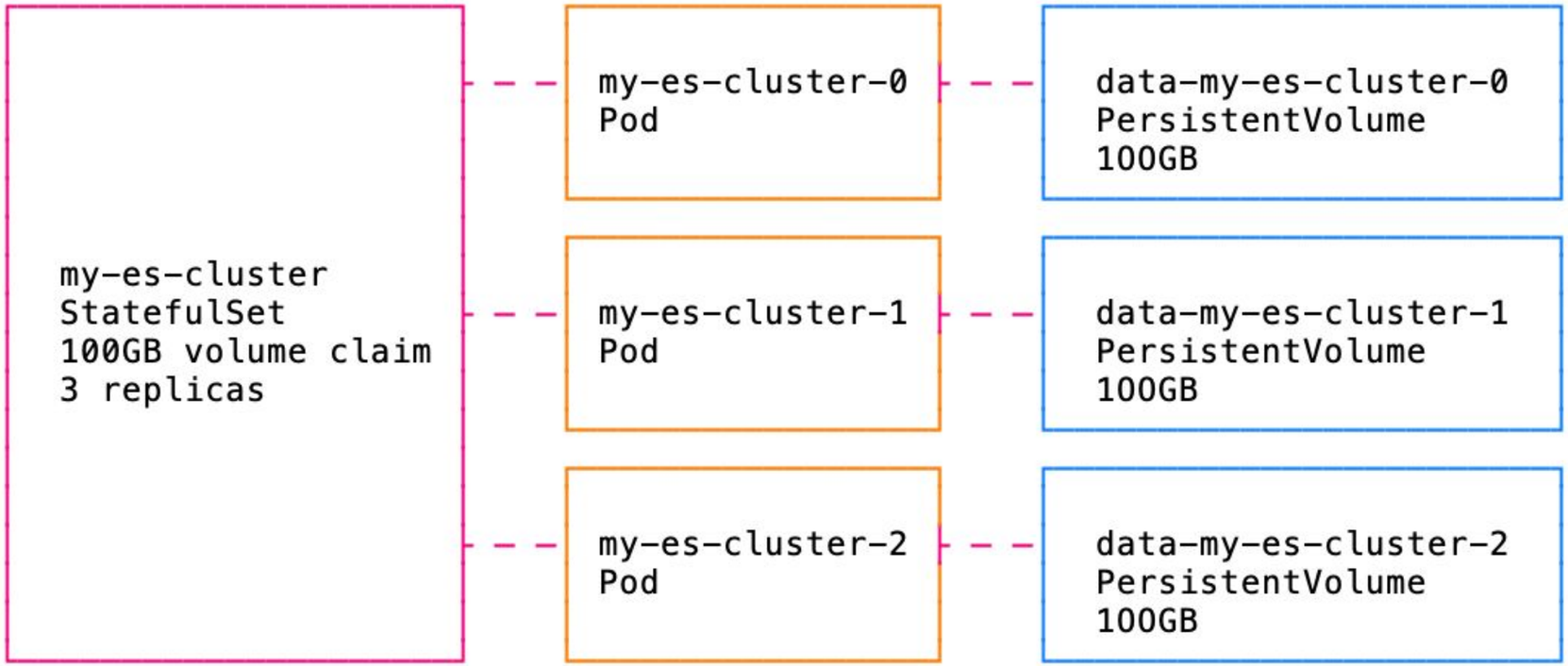
elastic

PersistentVolumes?



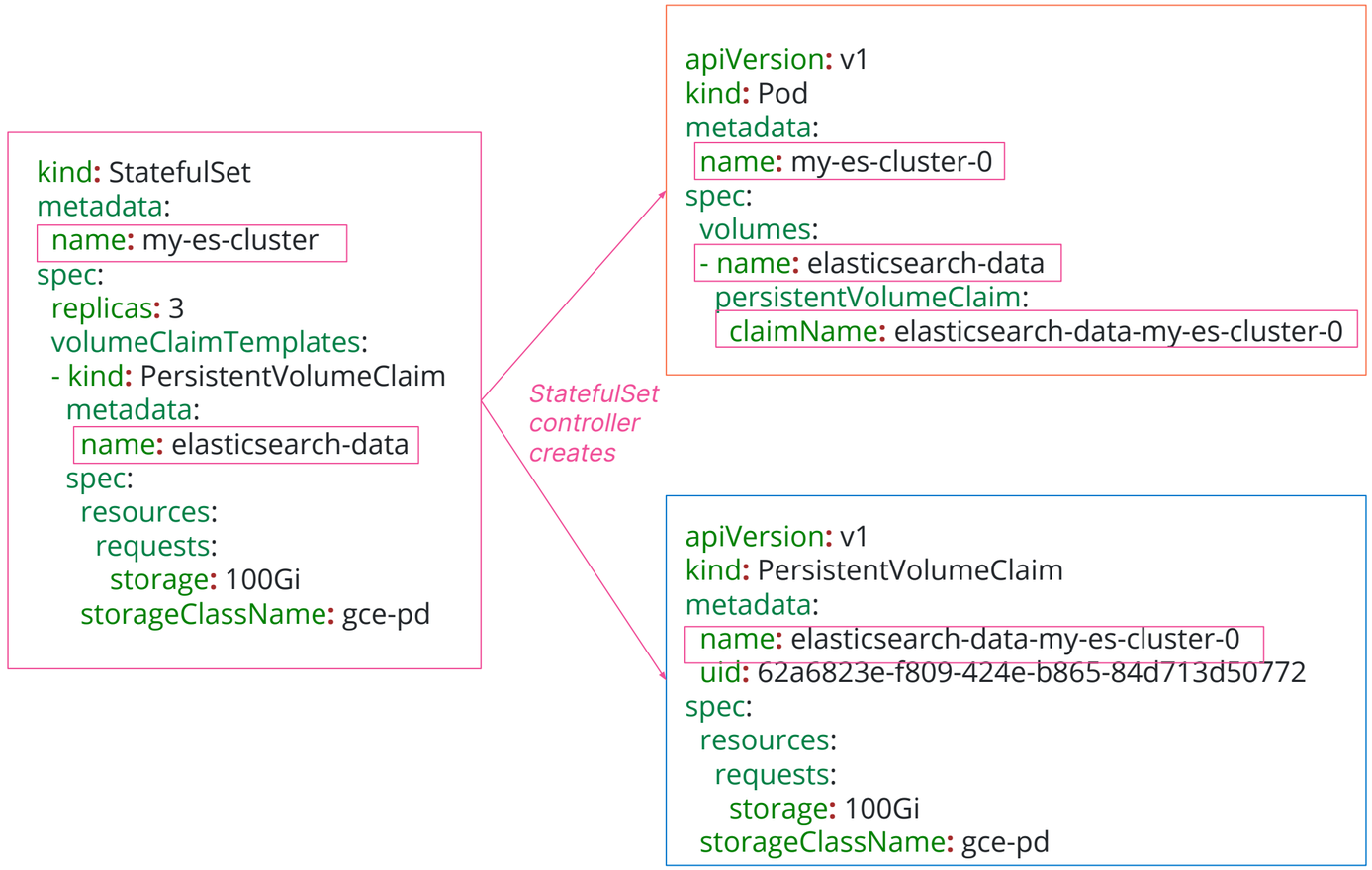
elastic

PersistentVolumes?



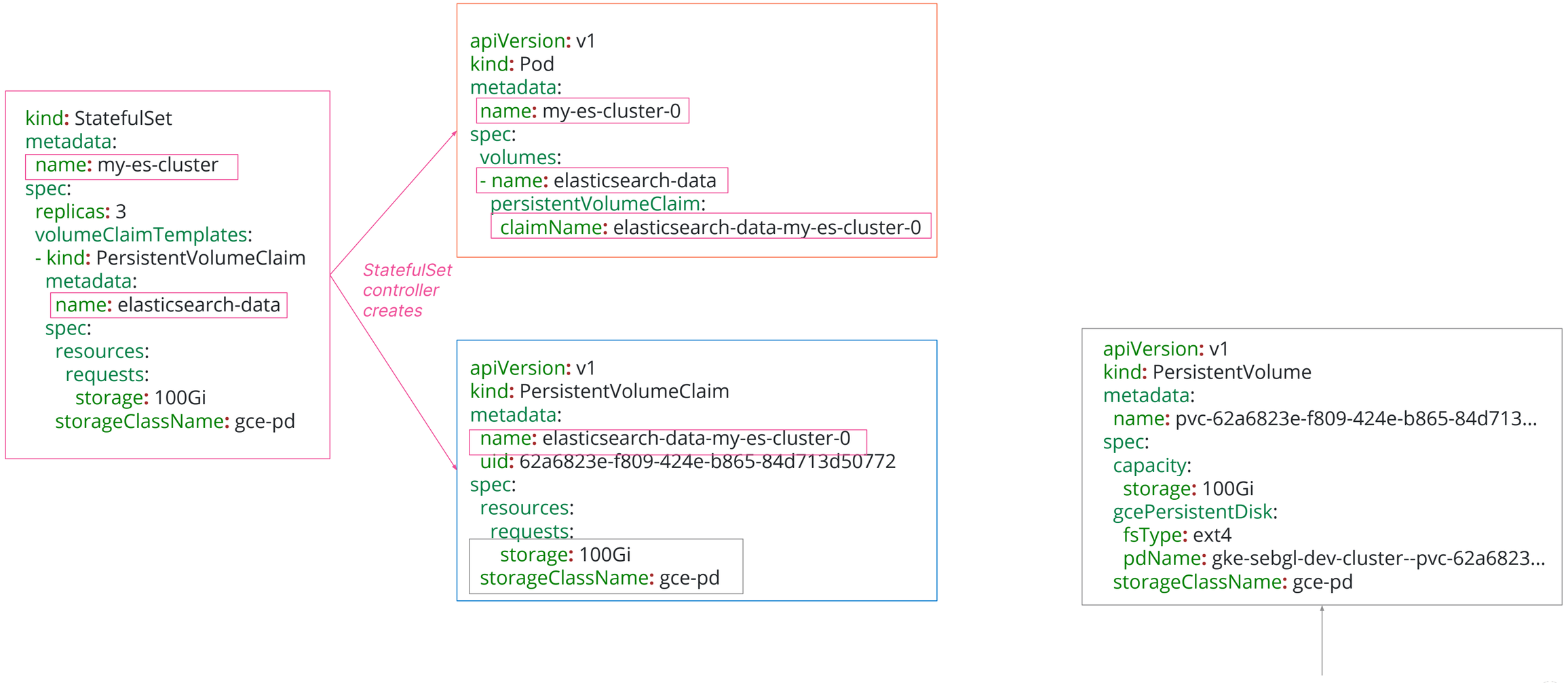
elastic

How things work



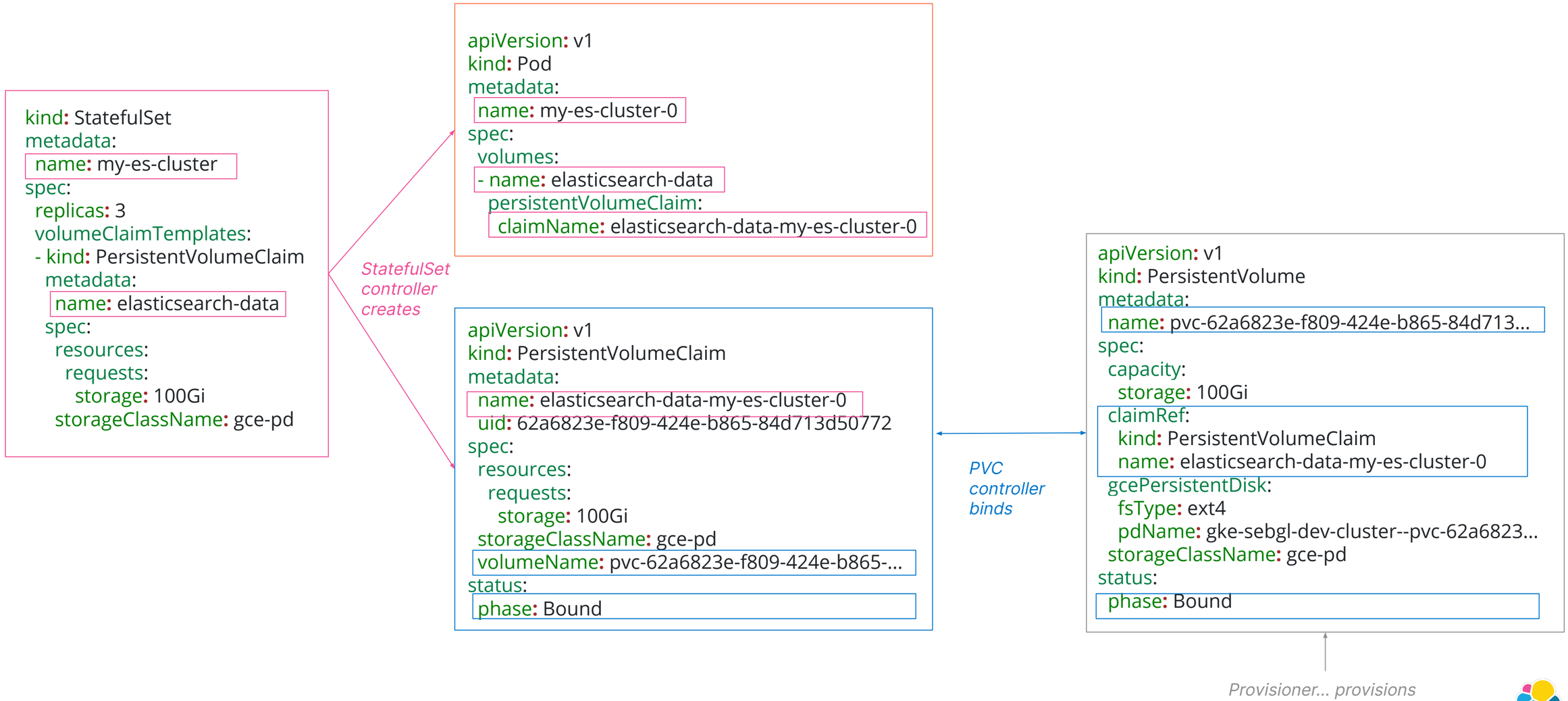
elastic

How things work



elastic

How things work



Local Volumes vs. Network-attached volumes

Local disk **performance**
(sometimes) **cheaper**
Require **provisioning**
Bound to a **single host**
Require **operational knowledge**

Network disk performance
(sometimes) more expensive
Built-in cloud provider provisioning
Can bind to any host in the region
Simpler operations

elastic

Local PersistentVolume ↔ K8s node affinity

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-2ba6de15
spec:
  capacity:
    storage: 100GB
  local:
    path: /mnt/disks/pvs/pv-1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - gke-sebgl-dev-cluster-default-pool-e73c0e56-s2b2
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  volumeMode: Filesystem
status:
  phase: Available
```

Can only be bound to
a Pod on that host

elastic

Local PersistentVolumes Provisioning

Local PV provisioning

Manual Provisioning

Create a PersistentVolume resource yourself

Static Provisioning

Run an agent to automatically create one PersistentVolume per available disk on each host

Dynamic Provisioning

Run a controller to dynamically create a PersistentVolume per PersistentVolumeClaim

Local PV provisioning

Manual Provisioning

Create a PersistentVolume resource yourself

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv-2ba6de15
spec:
  capacity:
    storage: 100GB
  local:
    path: /mnt/disks/pvs/pv-1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - gke-sebgl-dev-cluster-default-pool-e73c0e56-s2b2
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  volumeMode: Filesystem
```

kubectl apply -f my-pv.yml

elastic

Local PV provisioning

Static Provisioning

Run an agent to automatically create one PersistentVolume per available disk on each host

github.com/kubernetes-sigs/sig-storage-local-static-provisioner

README.md

Local Persistence Volume Static Provisioner

coverage 61%

The local volume static provisioner manages the PersistentVolume lifecycle for pre-allocated disks by detecting and creating PVs for each local disk on the host, and cleaning up the disks when released. It does not support dynamic provisioning.

Great when you know your volume sizes in advance, and need one volume per disk/partition

Tip: combine the static provisioner DaemonSet with an init container bash script that formats partitions as you wish

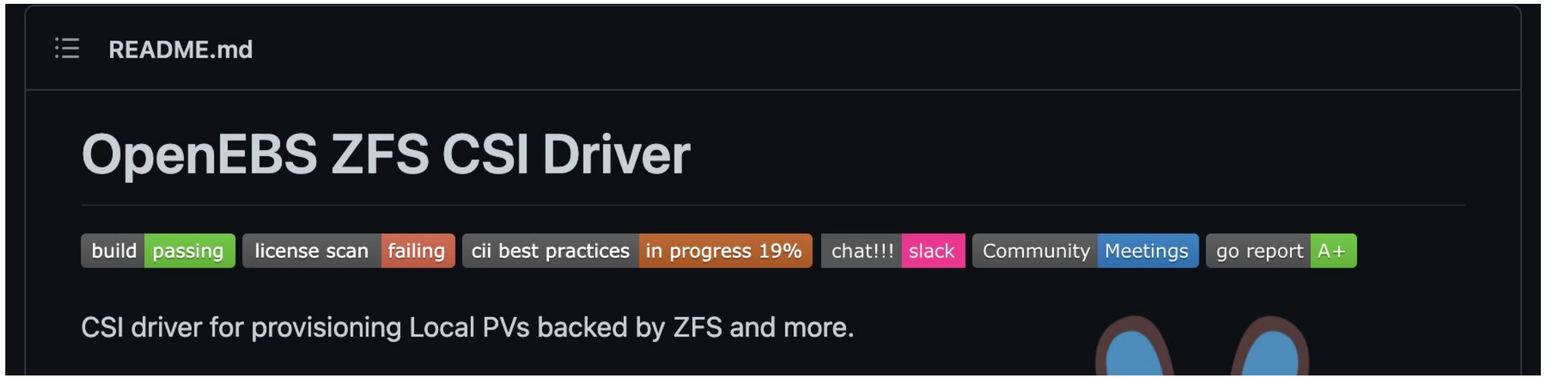
elastic

Dynamic provisioning

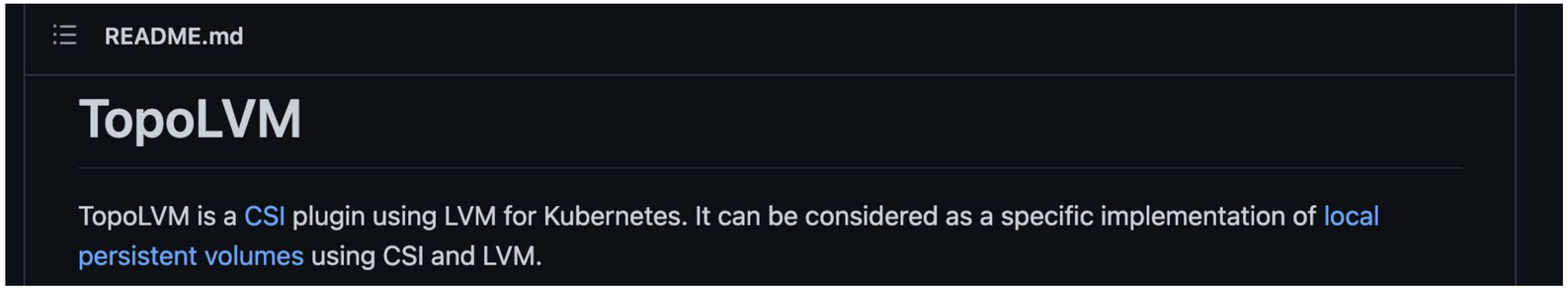
Dynamic Provisioning

Run a controller to dynamically create a PersistentVolume per PersistentVolumeClaim

github.com/openebs/zfs-localpv



github.com/topolvm/topolvm

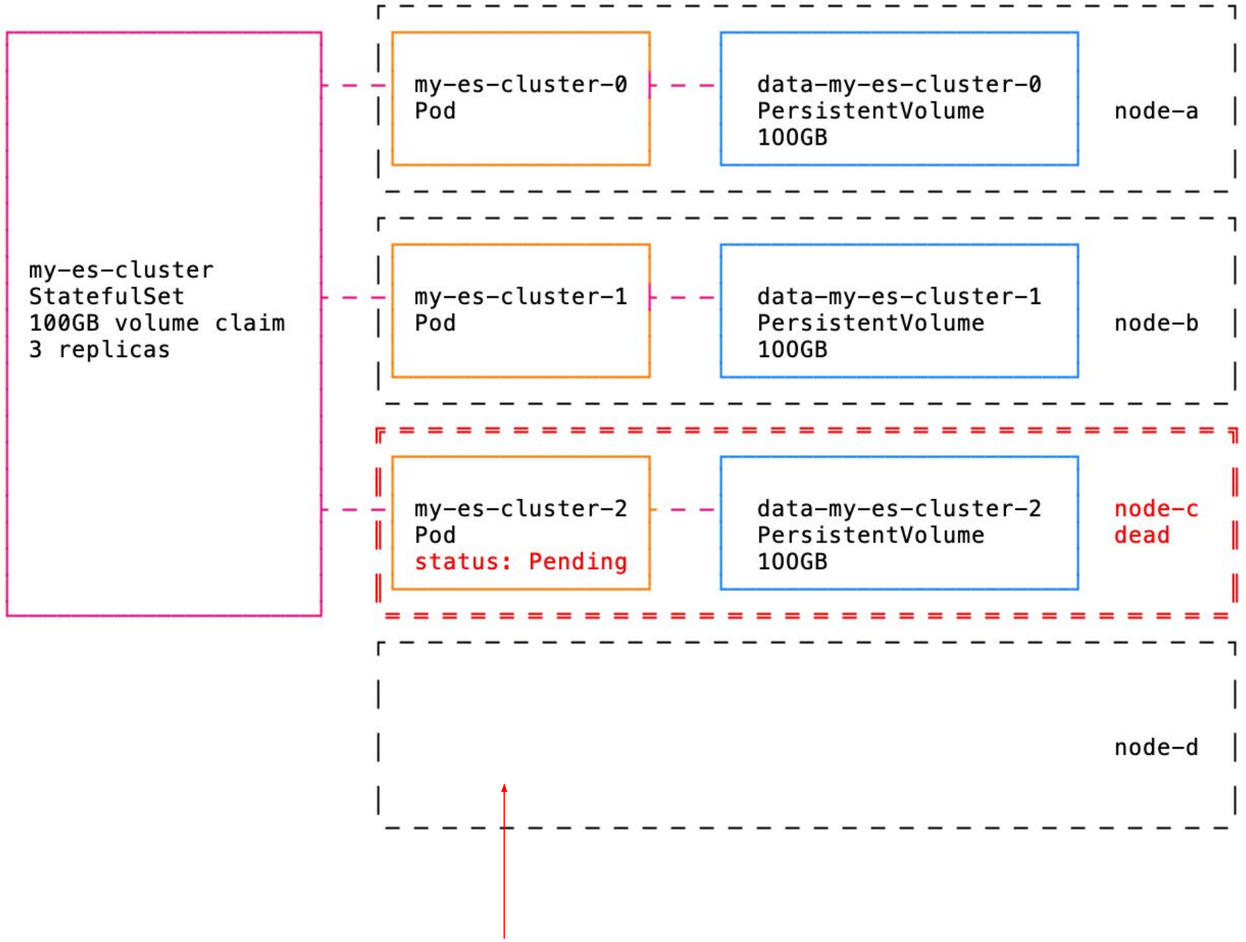


Great when you want to automatically provision a partition of the right size depending on the users needs

elastic

Operational gotchas

Host failure



I'd like a new Pod with a new empty volume instead!

elastic

Host failure

I'd like a new Pod with a new empty volume instead!

```
> kubectl delete pod my-es-cluster-2
```

Pod recreated automatically
bound to the same volume
which is bound to the same dead host
Status: Pending

elastic

Host failure

I'd like a new Pod with a new empty volume instead!

```
> kubectl delete pvc my-es-cluster-2
> kubectl delete pod my-es-cluster-2
```

Pod recreated automatically
new PVC recreated automatically
bound to an empty PV on a different host
Status: Running

Race condition in Pod + PVC deletion can require deleting the Pod again (twice).
Fixed in K8s 1.20: https://github.com/kubernetes/kubernetes/pull/93457

elastic

Host failure

I'd like a new Pod with a new empty volume instead!

```
> kubectl delete pvc my-es-cluster-2
> kubectl delete pod my-es-cluster-2
```

Can be automated:

- Watch K8s Node resources
- On any event:
  - List existing PVCs
  - Remove PVC + Pod matching PVs bound to a dead (non-existing) Node

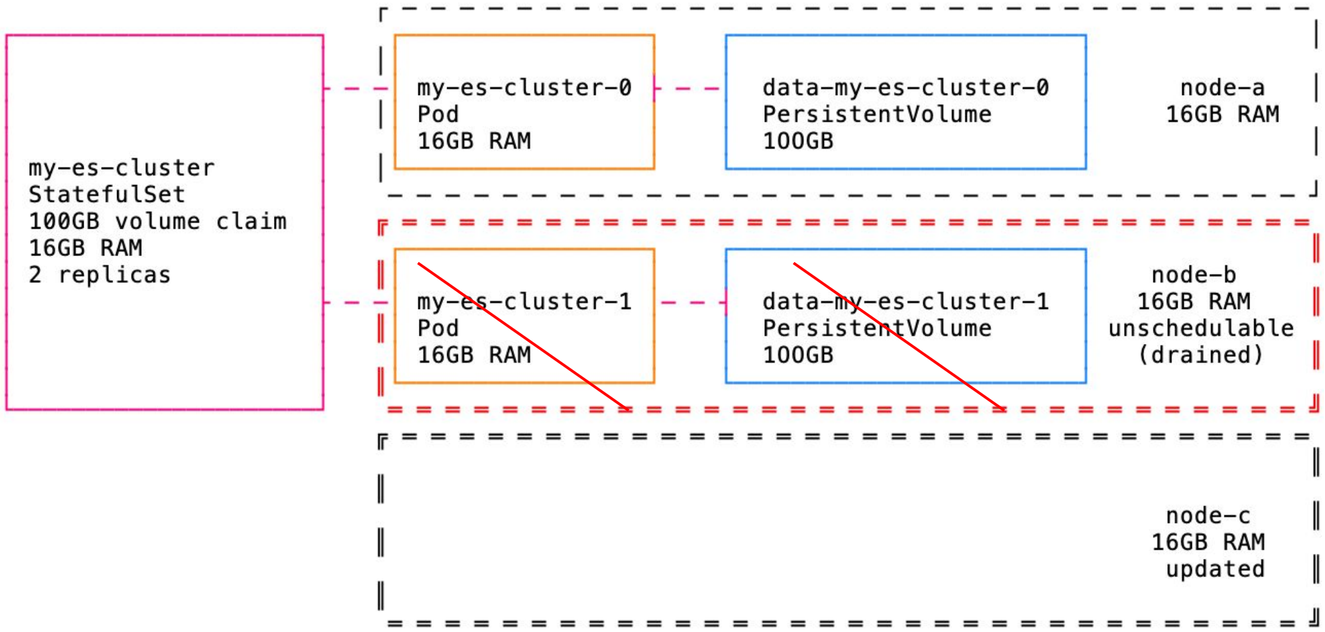
Differentiate **unhealthy/unschedulable nodes** (might come back online soon - wait for it!) from **non-existing nodes** (permanently dead).

elastic

Host decommission

Use case: Kubernetes version upgrade

- Just rolling replace all the VMs!
  - Spin up a new updated VM
  - kubectl drain <node>
  - Combined with a ***PodDisruptionBudget*** to only disrupt one Pod of the stateful workload at a time
  - Delete PVC + Pod
  - Let the Pod be recreated elsewhere (the new VM!)
  - (assuming data is replicated on other members)



elastic

Host decommission

Use case: Kubernetes version upgrade

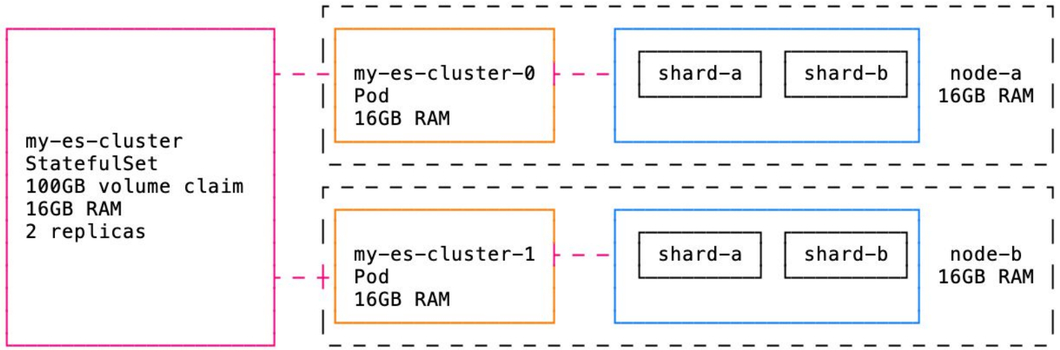
- Doesn’t always play well with **cloud providers automated upgrades**
  - Need to **delete the PVC** to let the Pod be recreated elsewhere
  - **PodDisruptionBudget** is only respected for X amount of time
    - GKE: up to 60 minutes
    - May not be enough to recover 5TB of data
    - Can lead to data loss
  - Safest option: manually create a new node pool with the new K8s version, migrate the workload there, then delete the old node pool

elastic

Host decommission

Use case: Kubernetes version upgrade

- This mechanism requires data **replication** and recovery handled **at the application level** (e.g. *Elasticsearch*)
  - Kill one node and its data, let it be recreated and recover its data



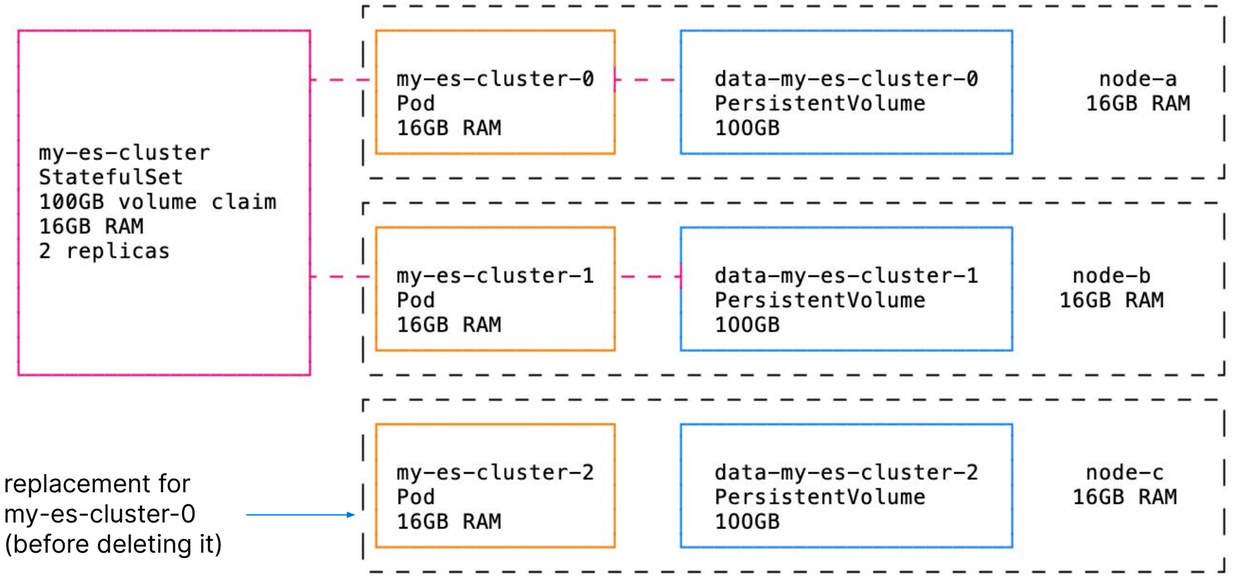
- What if another disruption (e.g. another host failure) happens concurrently?
  - Potential availability loss
  - Potential data loss

elastic

Host decommission

Use case: Kubernetes version upgrade

- We'd be in a better place if we could **spin up a replacing Pod** and **migrate data before** deleting the original Pod
- StatefulSets don't support this

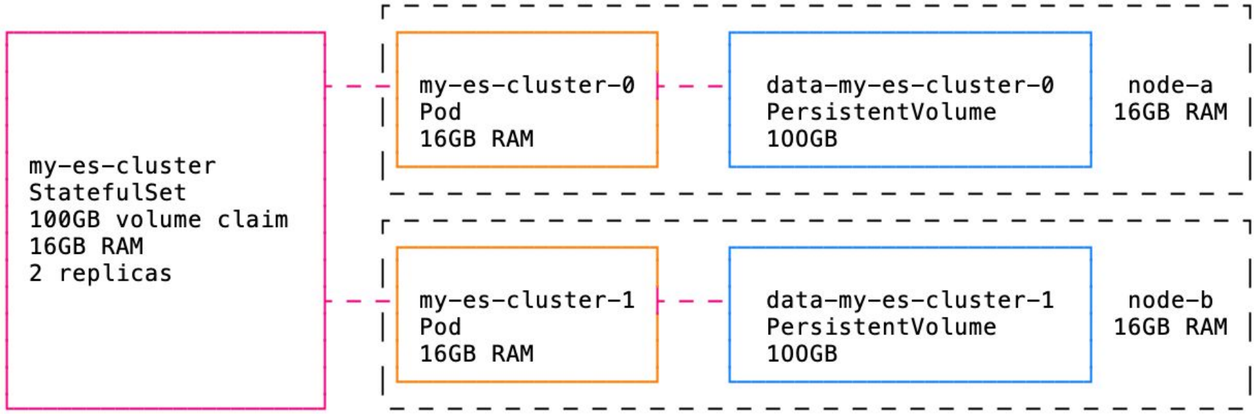


Pod with ordinal -0 must exist, will be recreated

elastic

Concurrent StatefulSet upgrade & Pod scheduling

StatefulSet rolling upgrade (spec change)



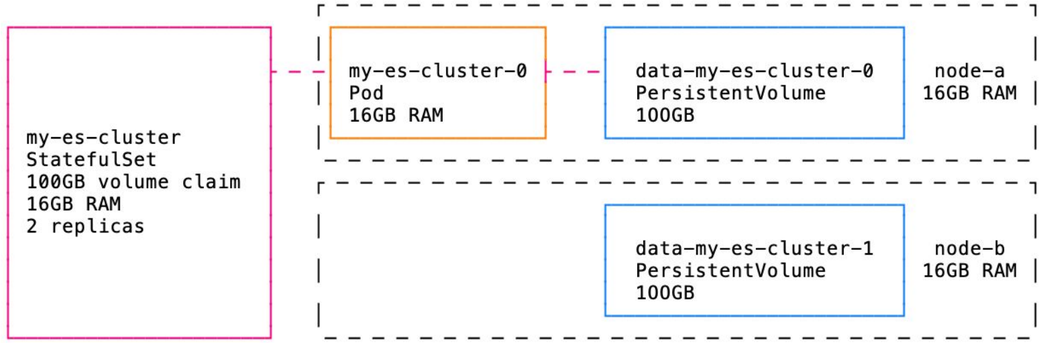
elastic

Concurrent StatefulSet upgrade & Pod scheduling

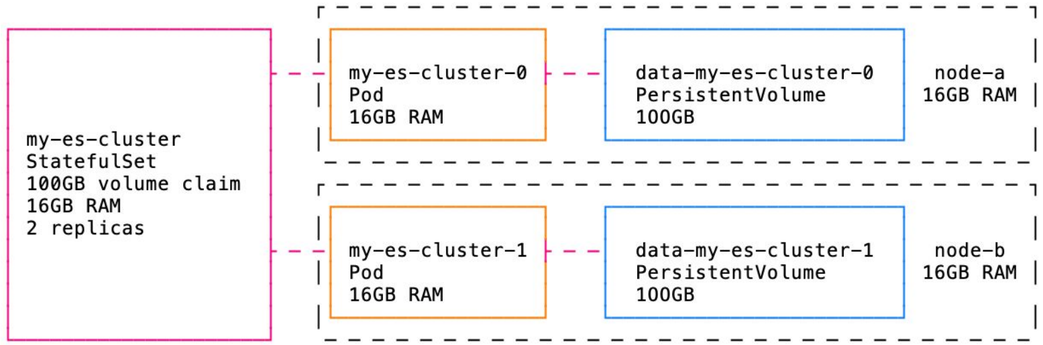
StatefulSet rolling upgrade (spec change)

For each Pod, in a rolling fashion:

1. Pod deleted



2. Pod recreated with the new spec



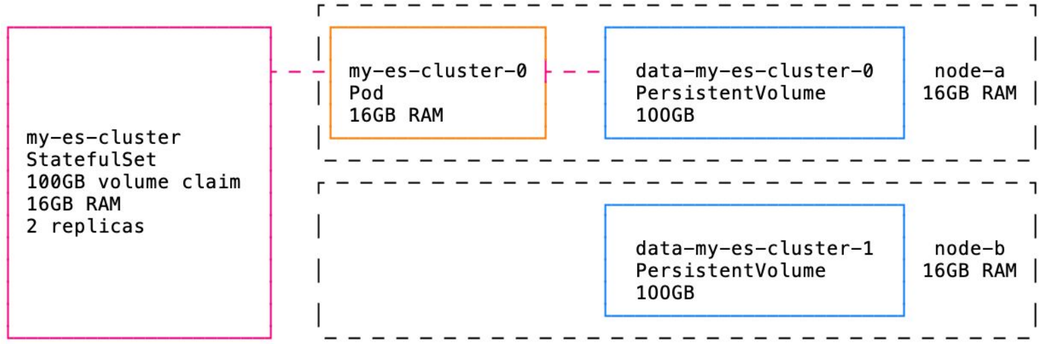
elastic

Concurrent StatefulSet upgrade & Pod scheduling

StatefulSet rolling upgrade (spec change)

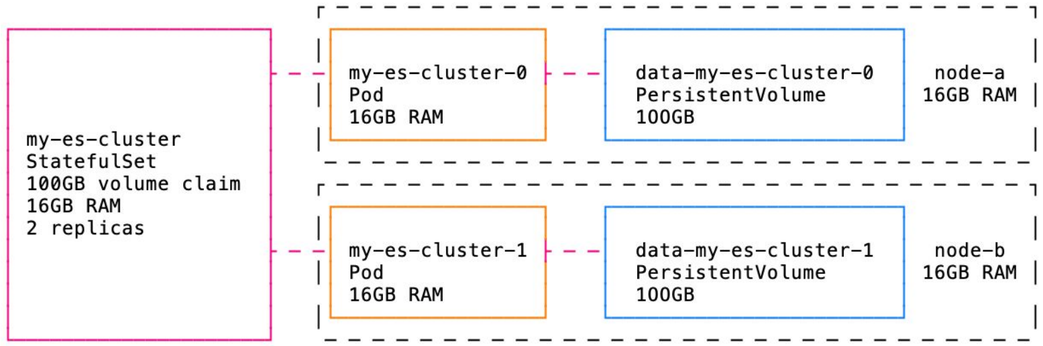
For each Pod, in a rolling fashion:

1. Pod deleted



1.5 What if another Pod gets scheduled on the host here?!

2. Pod recreated with the new spec



elastic

Concurrent StatefulSet upgrade & Pod scheduling

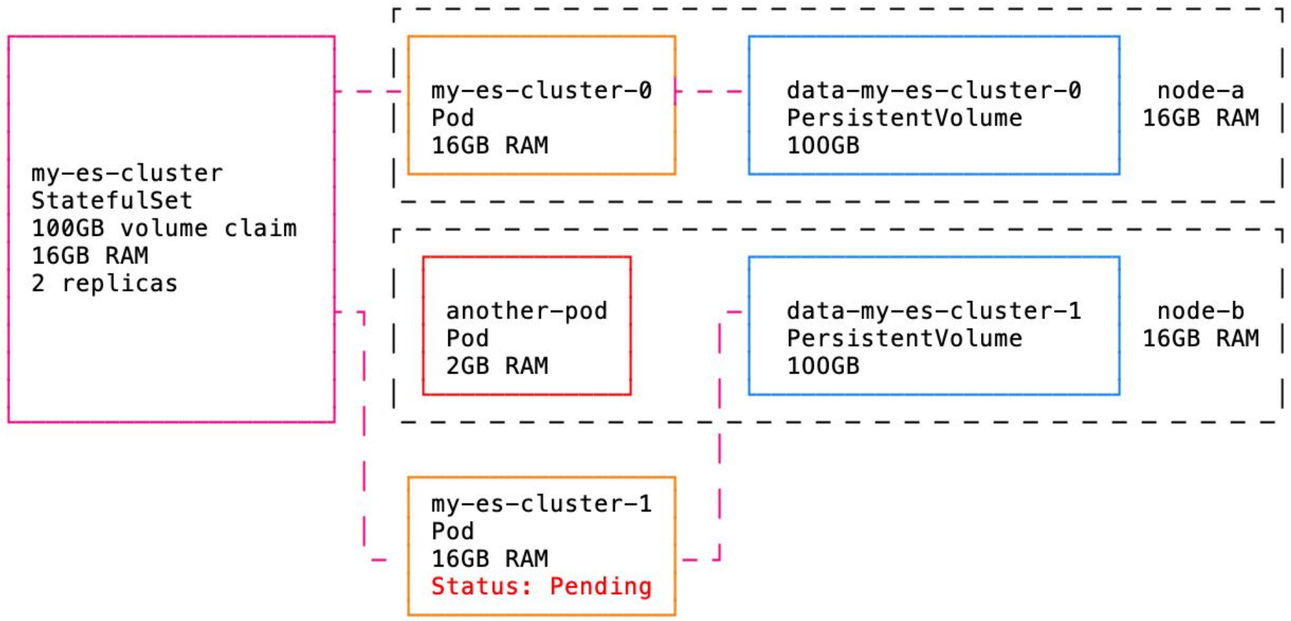
StatefulSet rolling upgrade (spec change)

For each Pod, in a rolling fashion:

1. Pod deleted

1.5 What if another Pod gets scheduled on the host here?!

2. Pod recreated with the new spec



elastic

Concurrent StatefulSet upgrade & Pod scheduling

Avoiding that situation

- Give higher priority to Pods with Local PVs (*spec.priorityClassName*)
- Setup taints and tolerations to isolate workloads with Local PVs (e.g. dedicated k8s nodes for Elasticsearch workloads)
- Use a fixed RAM to Storage capacity ratio for those Pods
  - example ratio: 10GB RAM ↔ 200GB storage
  - example machine:
    - 30GB RAM, 600GB storage
  - can’t schedule a new 10GB RAM Pod while another one is being recreated if there isn’t also 200GB storage available

Github issue (closed...):
https://github.com/kubernetes/kubernetes/issues/78638

elastic

Storage capacity awareness

No more storage

When using dynamic provisioning, Pods can be scheduled to Nodes with insufficient storage capacity.

- Storage capacity tracking in CSI drivers starting 1.21: https://github.com/kubernetes/enhancements/issues/1472
- Not a problem with dedicated Nodes or a fixed RAM - Storage ratio (no more storage ⇒ no more RAM ⇒ no scheduling)

1TB volume for my 1GB claim

When using static provisioning, the scheduler ignores available PV sizes when picking a Node. Large PVs can be consumed for small PVCs. (Though once a Node is picked, the smallest PV is favored).

- Prioritization on volume capacity, alpha in 1.21: https://github.com/kubernetes/kubernetes/issues/102902

elastic

elastic

What you need to know before using Local PersistentVolumes

Sébastien Guilloux - Kubecon NA 2021