RESILIENCE

REALIZED

KubeCon | CloudNativeCon

North America 2021

Disaster Recovery of Stateful Application in a Multi-Cluster Environment

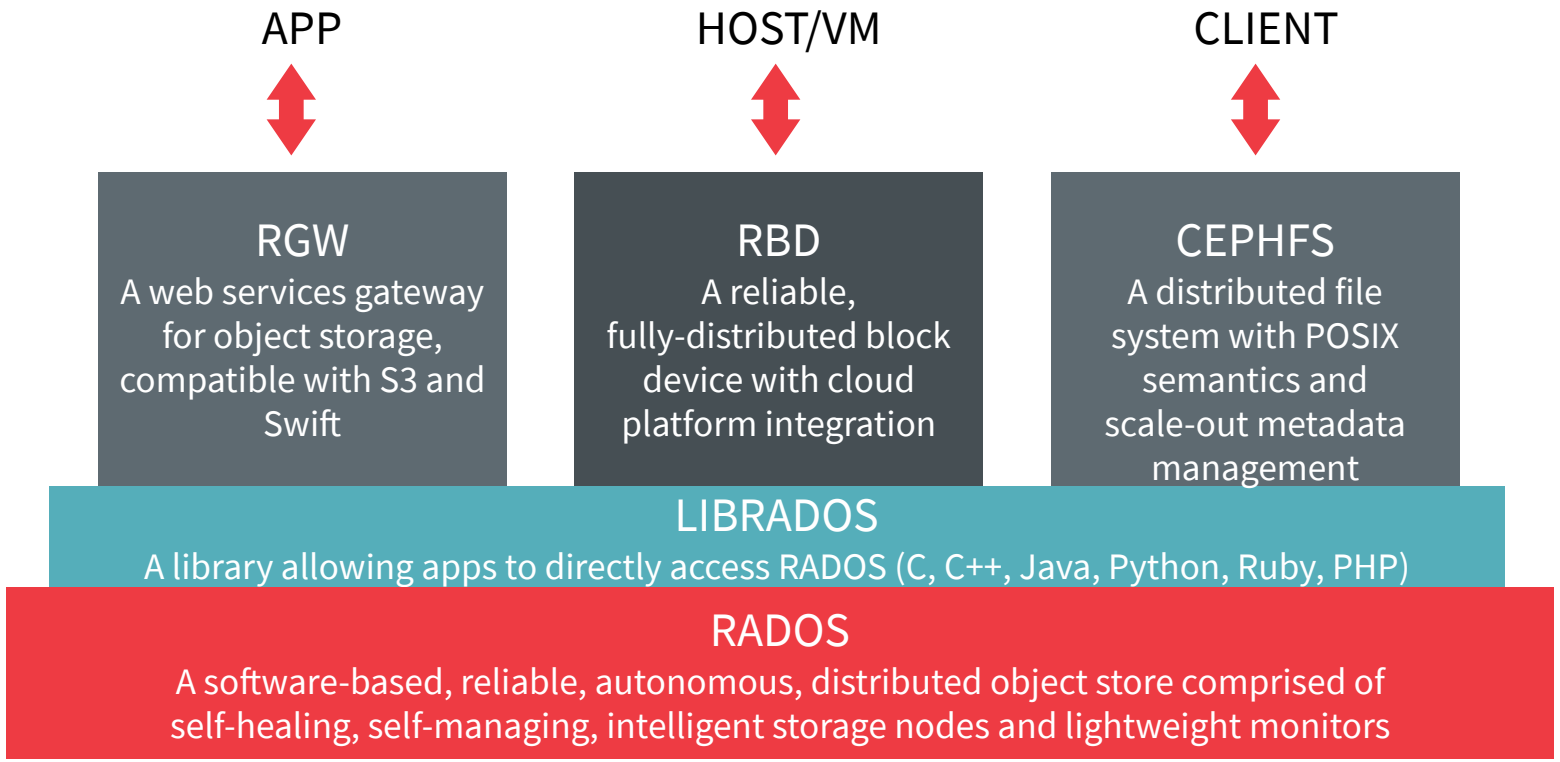
Orit Wasserman, Red Hat

Shyam Ranganathan, Red Hat

Agenda

- Disaster Recovery 101
- Storage Replication
- Multi-cluster Management
- Recovery/Relocate Orchestration
- Demo
- Future work

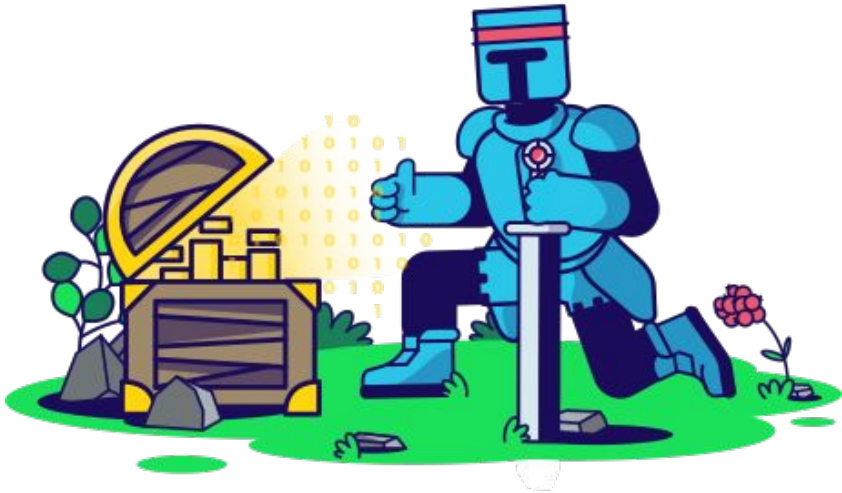
What is Ceph?



ceph

What is Rook?

- Storage made available inside your Kubernetes cluster
- Kubernetes Operators and Custom Resource Definitions (CRDs)
- Automated management
  - Deployment, configuration, upgrades
- Consume like any other K8s storage
  - Storage classes, PVCs, etc.
- Open Source (Apache 2.0)



Disaster Recovery 101

Why Disaster Recovery?

- Business Continuity in case of a full data center or region failure
- DR site should be isolated from the disaster:
  - High network latency
  - Async replication
- Use HA with Synchronous replication if the network latency allows it

Disaster Recovery and High Availability

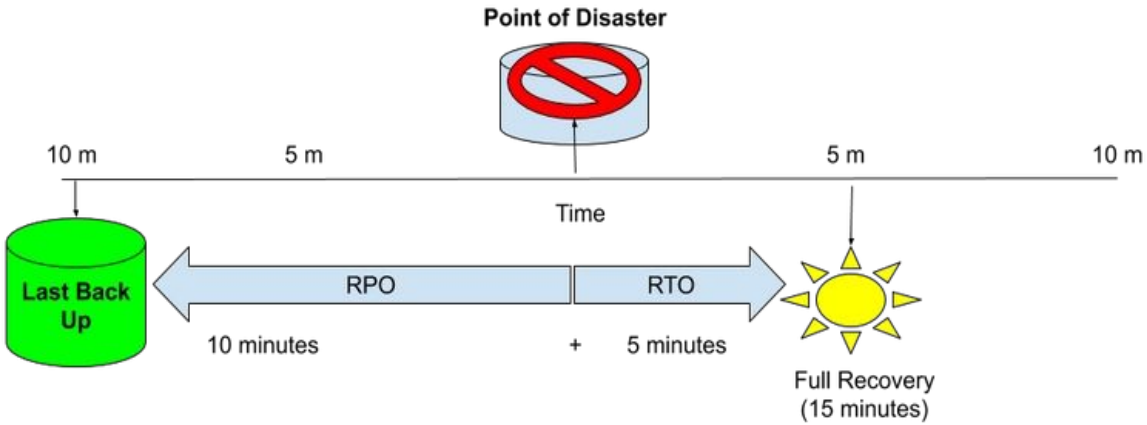
RPO and RTO

- Recovery Point Objective (RPO):

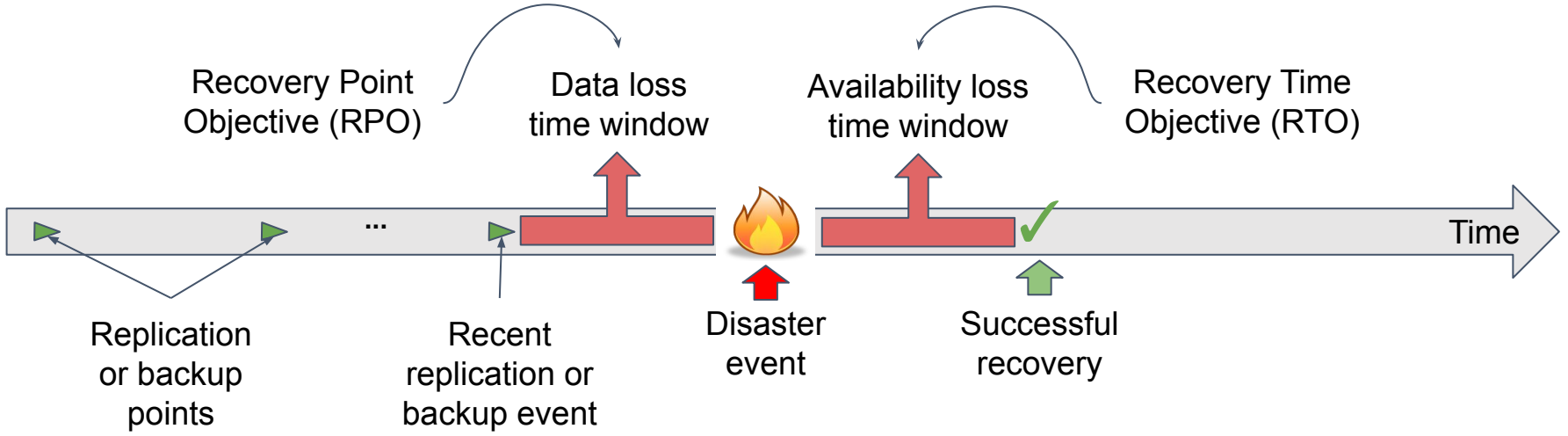
Amount of acceptable data loss defined from the point of the disaster to the last known backup or recovery point.

- Recovery Time Objective (RTO)

Amount of time that an application can be down before it significantly impacts the business.



An Ideal DR Solution



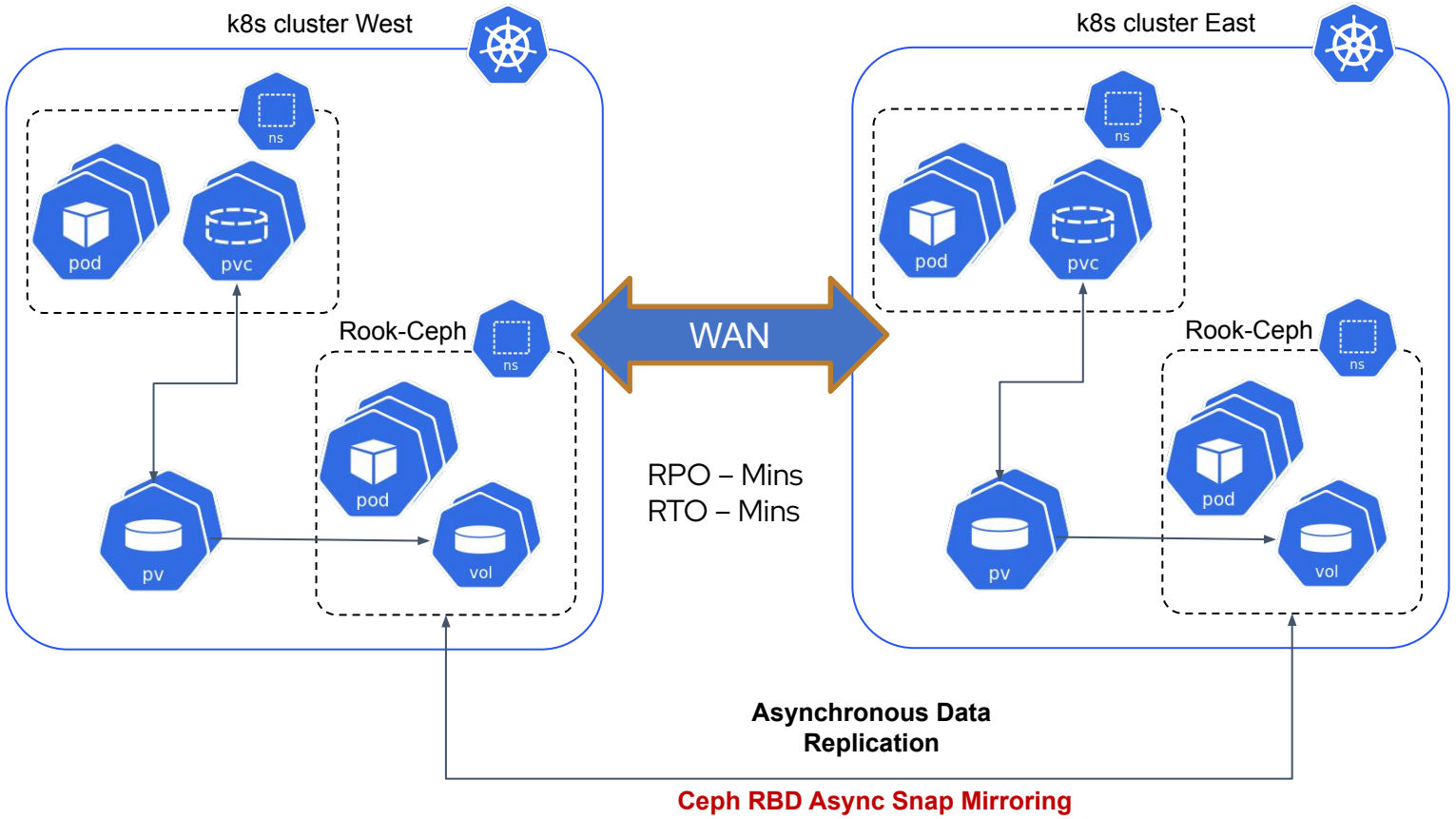
- Low RPO (minutes)
  - Admin triggered DR is usually preferred to accept data loss
- Low RTO (minutes)
- Single pane of glass to orchestrate failover and failback

Example: Rook+Ceph Regional DR

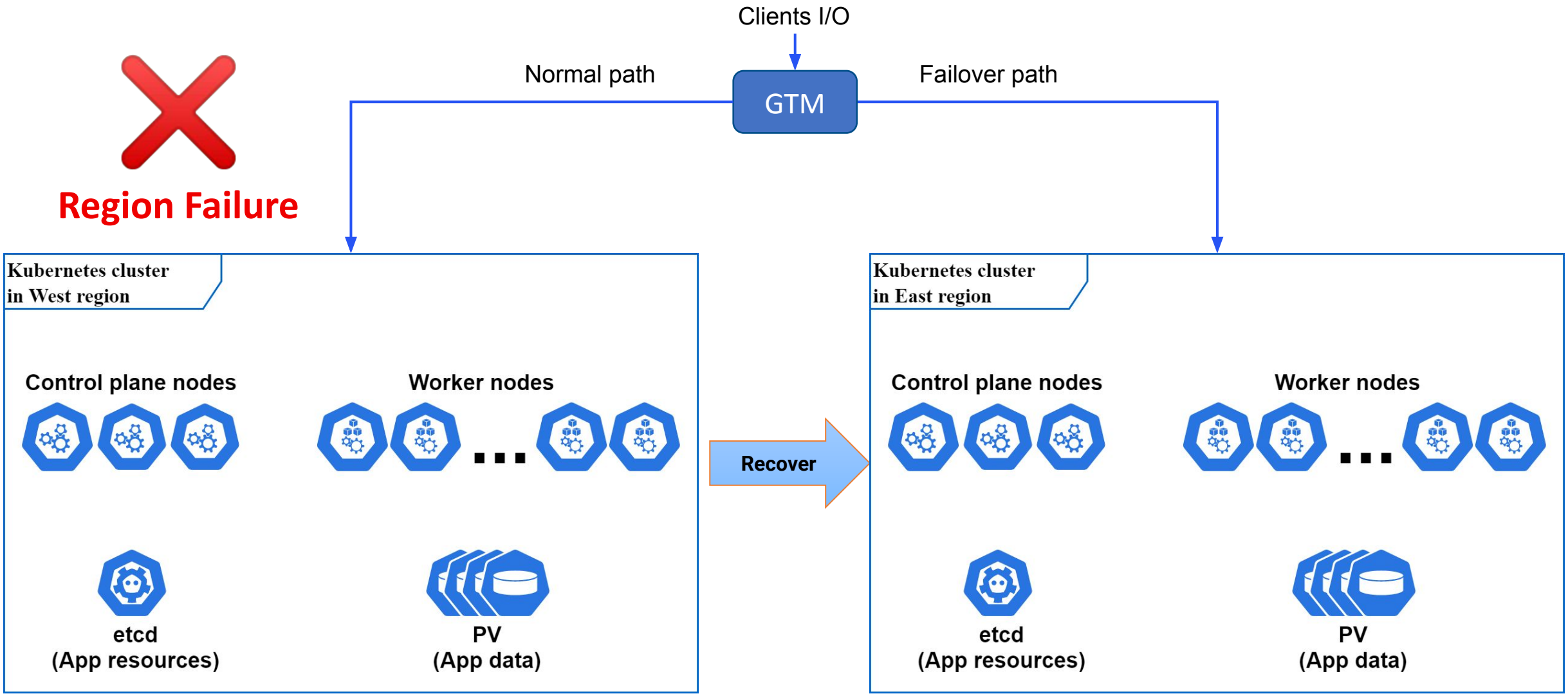
- Standby DR site:

  Application/Namespace is active on primary site and standby at DR site.
- Two way:

  Each K8s cluster can be active and standby in the same time
- Asynchronous Persistent Volume Replication
  - Based on Ceph RBD snapshot based mirroring

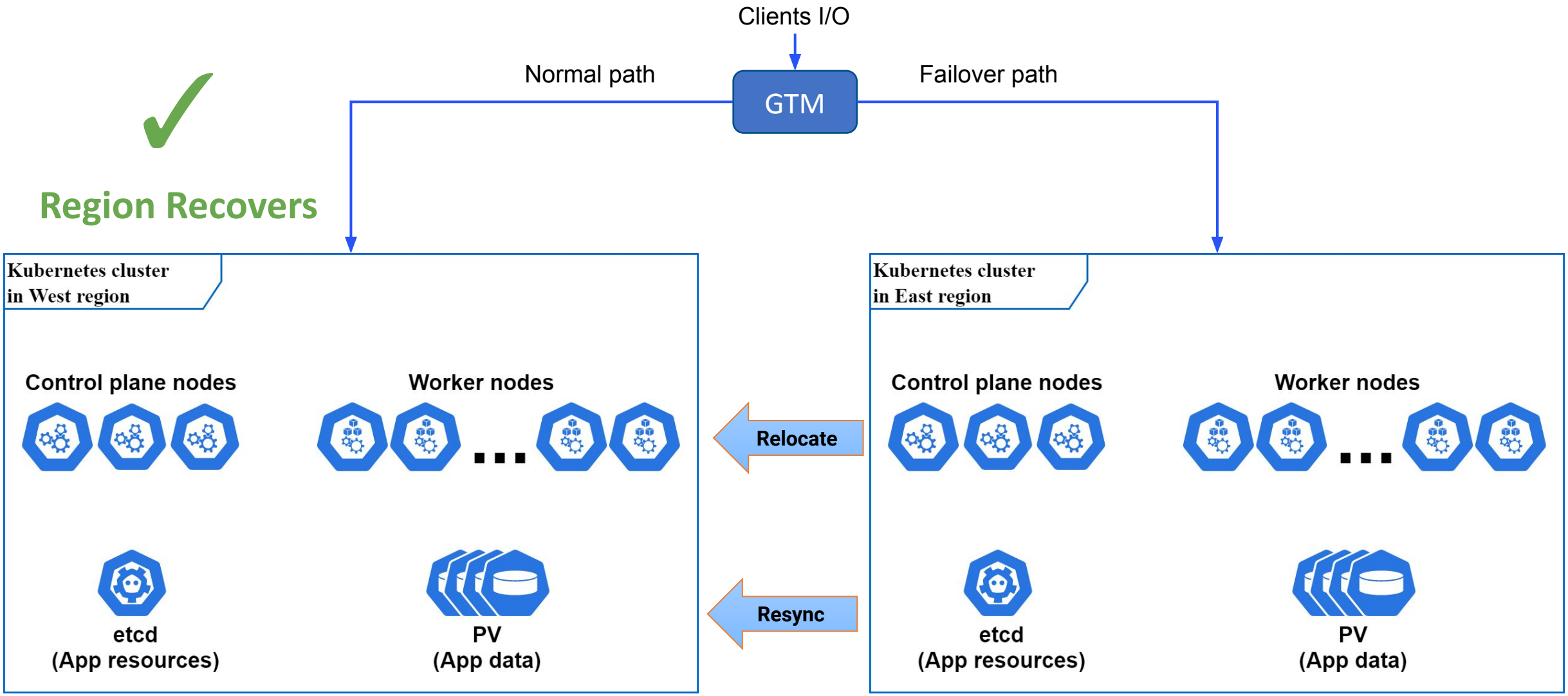


Regional DR Goal - Recover



*PV = Persistent Volume

Regional DR Goal - Relocate



*PV = Persistent Volume

Storage Replication

Backups as Replication

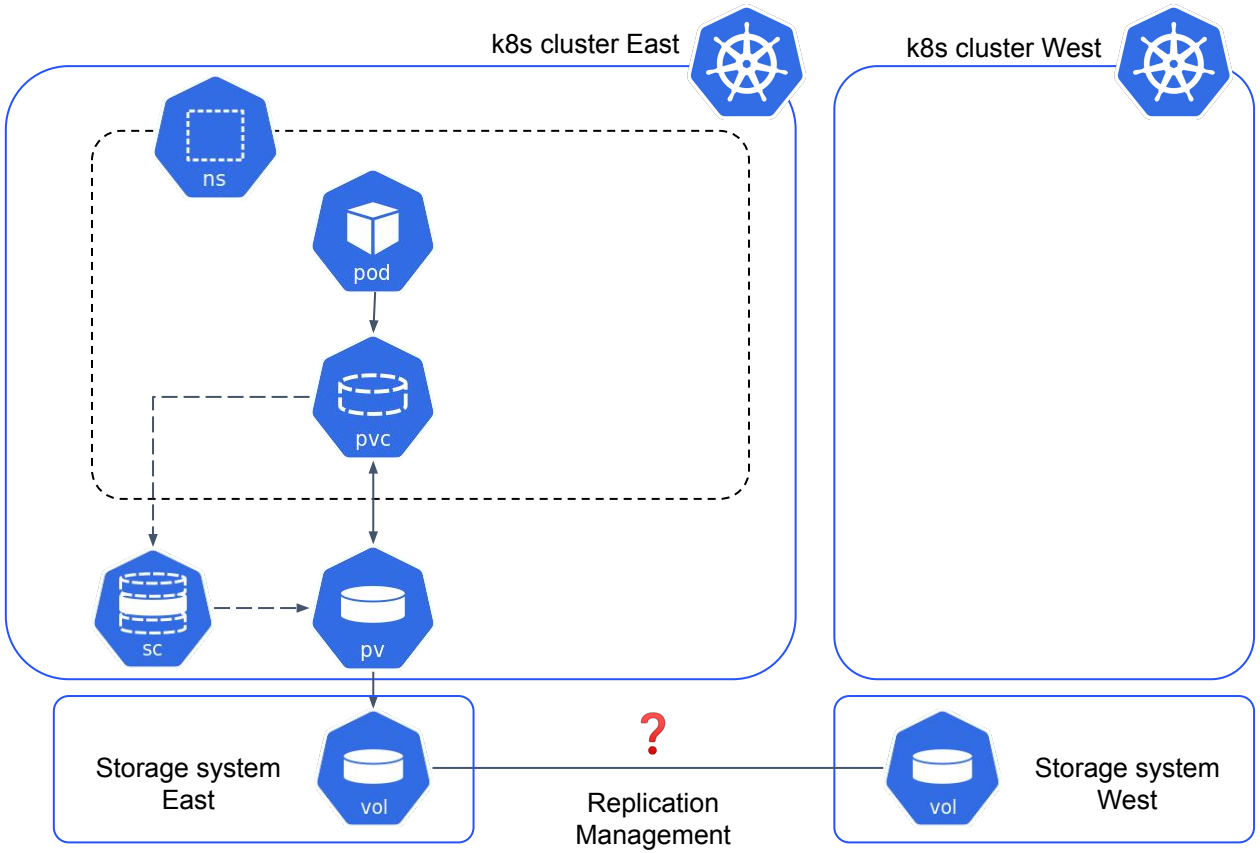
Why backups are insufficient

- High RPO and RTO for data
  - Backup frequency is typically in hours, hence RPO is higher
  - Restore before use is required, hence RTO is higher
    - NOTE: It is feasible that a live restored volume is always in place reducing RTO
- Transfer efficiency of data
  - Full data backups are less efficient, but incremental backups increase RTO
  - Entire volume data needs to be examined for detecting incremental changes to backup
    - NOTE: Enhancements like Change Block Tracking can alleviate these concerns
- K8s resources (or cluster data) backups are point-in-time
  - Potentially better served from a declarative source (gitops), providing 0 RPO

Storage System Based Replication

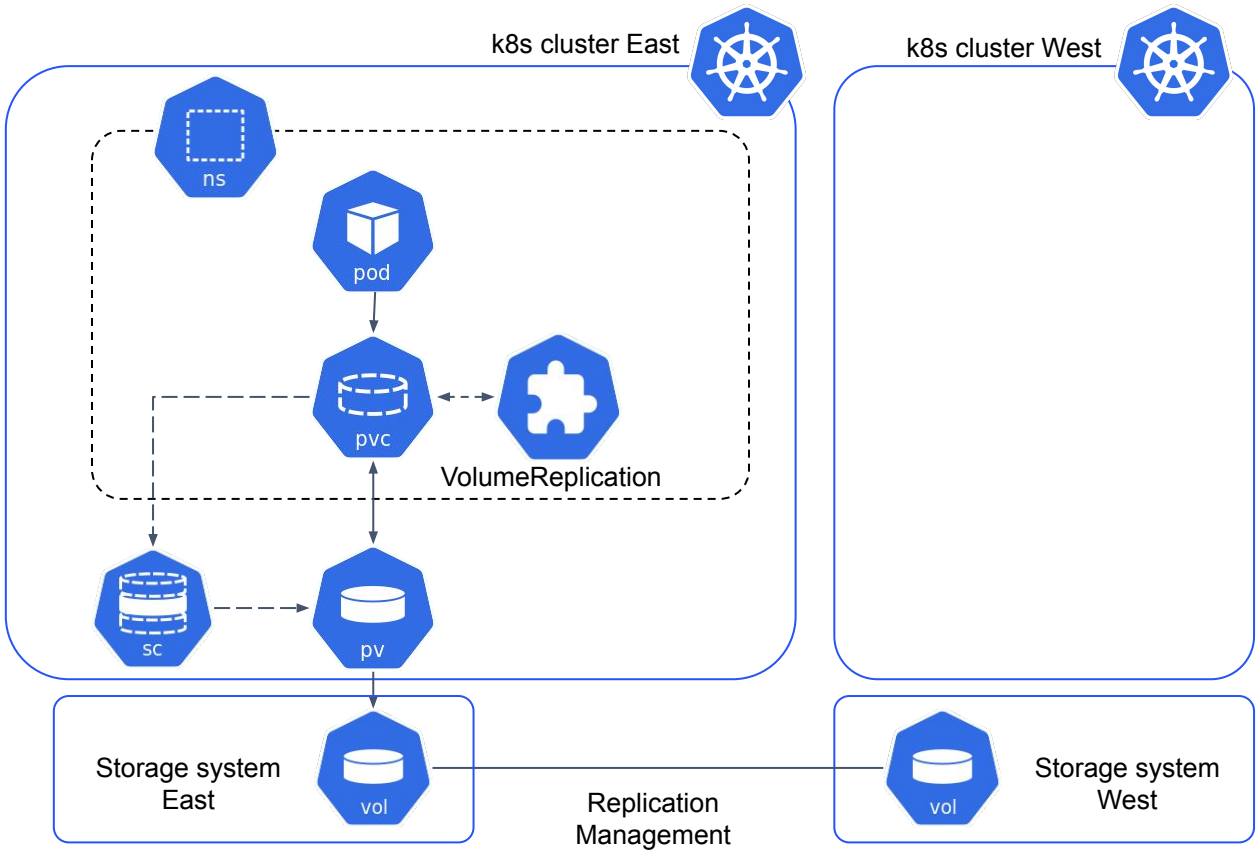
- Storage systems provide mirror/replication features that:
  - Typically leverage periodic snapshots delta transfers between storage instances, providing lower RPO
  - Separate storage and user IO pathways for better transfer efficiency
  - Replicated copies are (near) instantly available, reducing RTO for storage components
- Drawbacks
  - Are NOT storage system agnostic
- Gaps
  - Need additional APIs to manage storage assisted replication in k8s

Storage Replication Management APIs



- Storage systems setup to replicate volumes across fault domains (East-West regions)
- Missing replication management APIs ❓
  - Enable storage vendor management for per volume replication

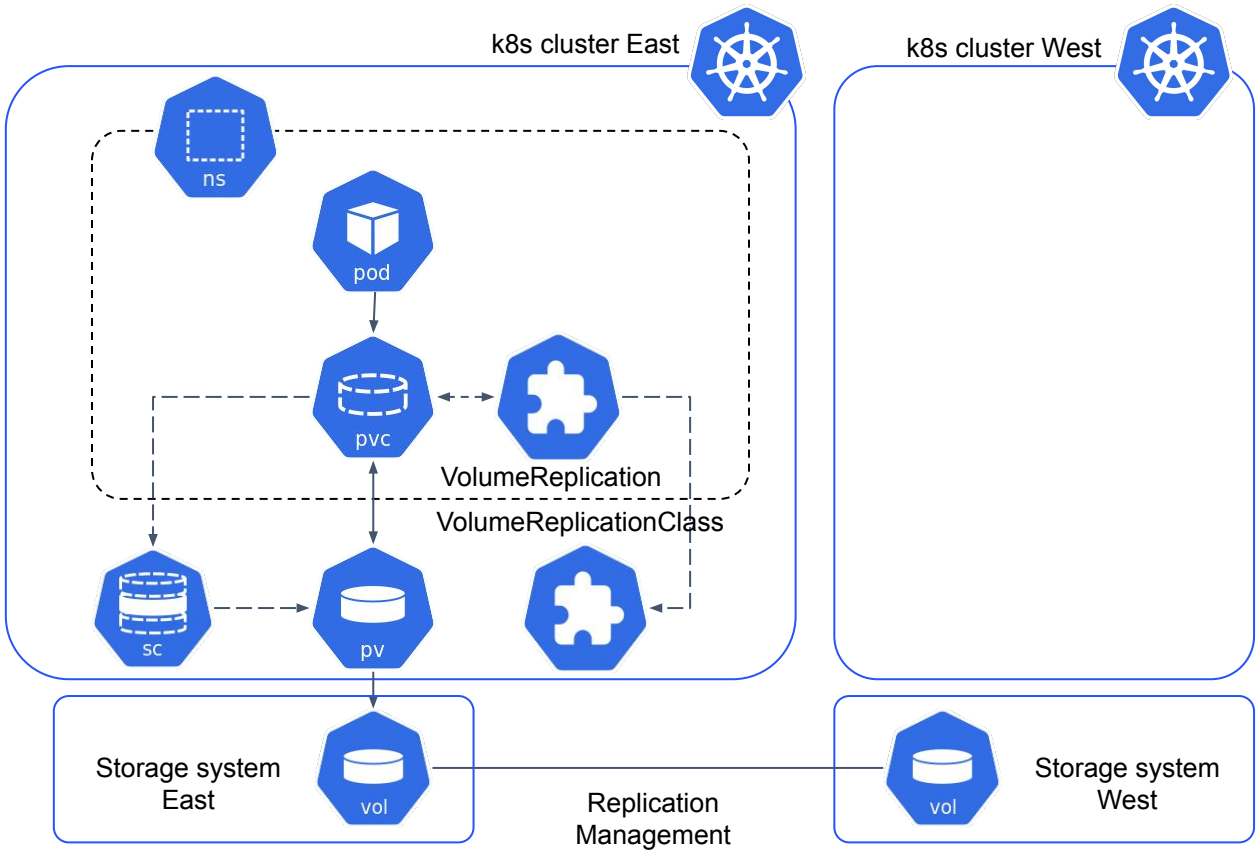
Storage Replication Management APIs



- Add CSI API extensions for VolumeReplication resource management
  - Create/Delete (Enable/Disable replication)
  - *spec.replicationState* [Primary|Secondary]
  - *spec.dataSource* points to a PVC requiring replication
  - CSI spec extensions for storage vendor specific actions
- CSI sidecar for reconciliation of VolumeReplication objects

```
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplication
metadata:
  name: volumereplication-sample
  namespace: default
spec:
  volumeReplicationClass: volumereplicationclass-sample
  replicationState: primary
  dataSource:
    kind: PersistentVolumeClaim
    name: myPersistentVolumeClaim # should be in same
namespace as VolumeReplication
```

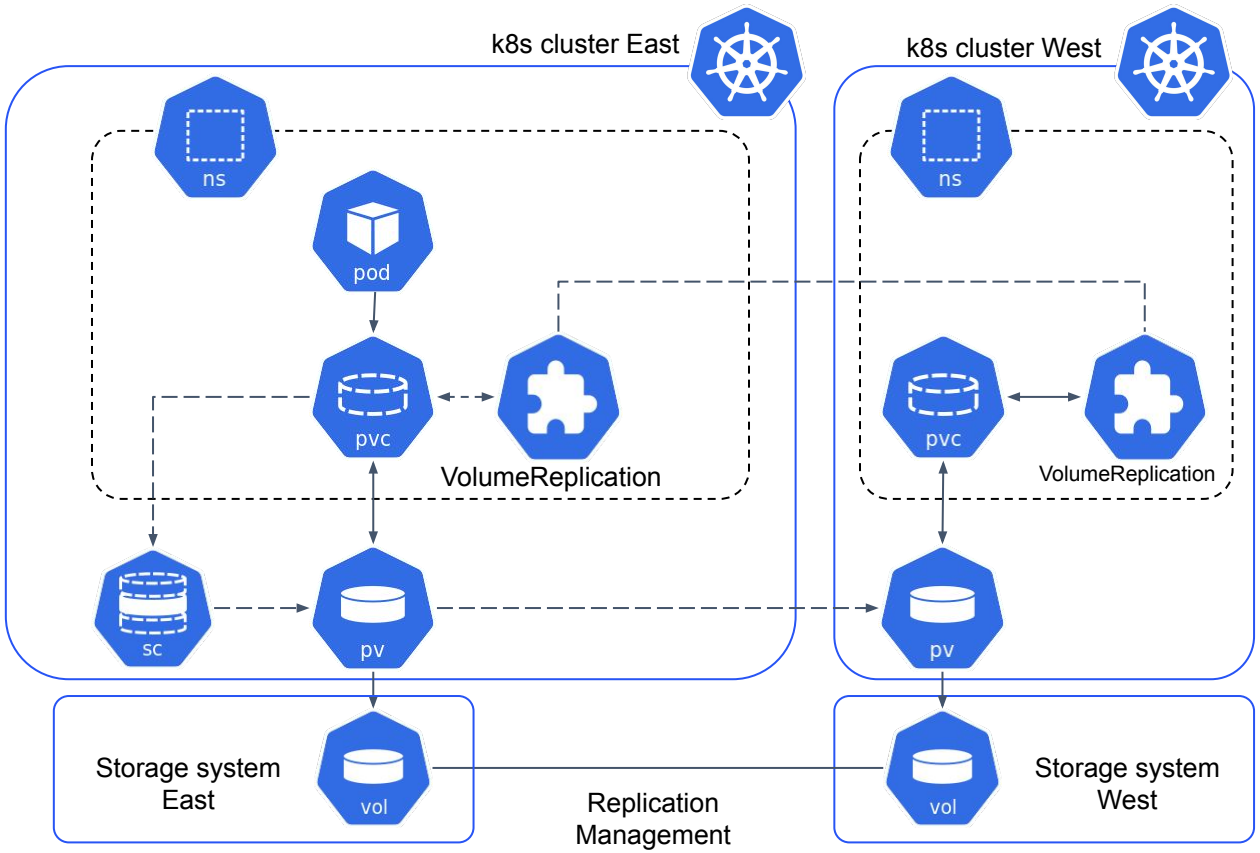
Storage Replication Management APIs



- Enhance capabilities using VolumeReplicationClass resource
  - Secrets
  - Replication schedules
  - Vendor specific parameters

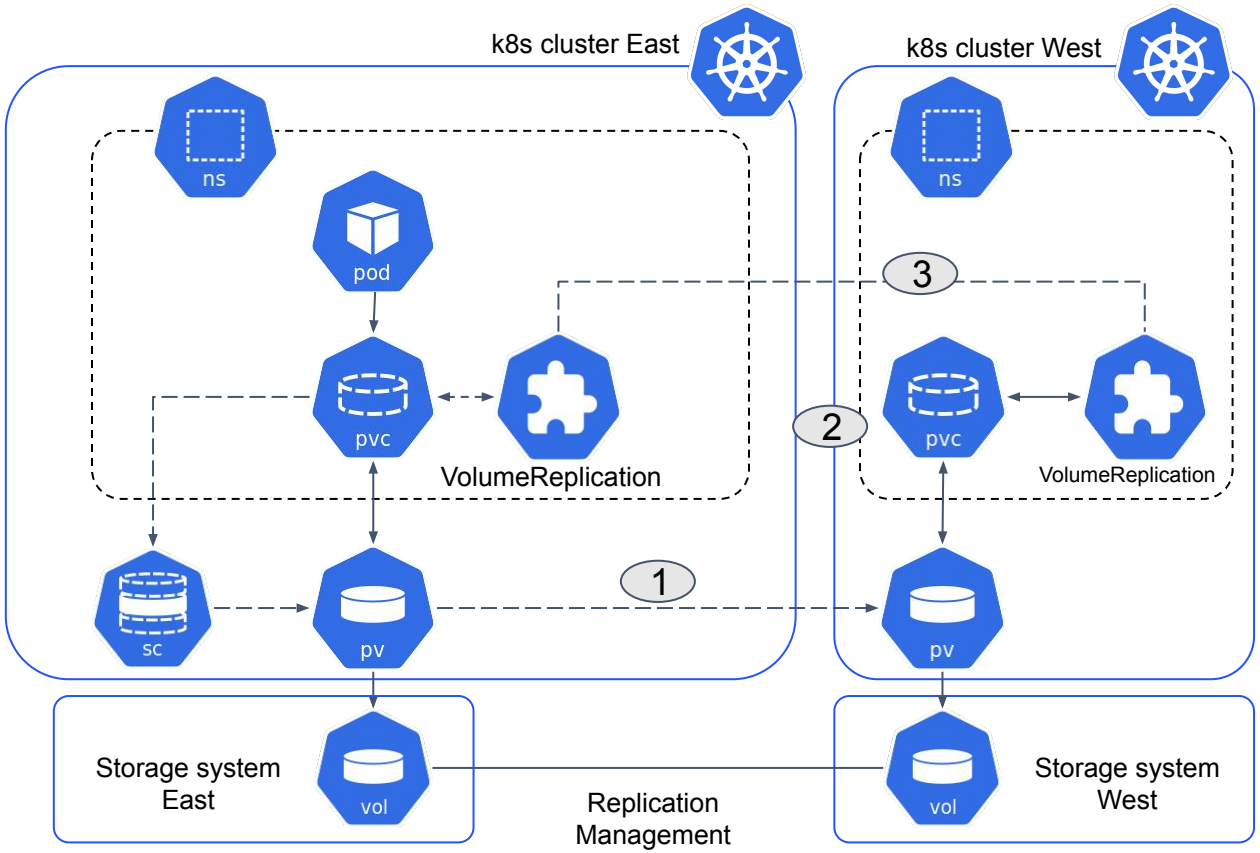
```
apiVersion:
replication.storage.openshift.io/v1alpha1
kind: VolumeReplicationClass
metadata:
  name: volumereplicationclass-sample
spec:
  provisioner: example.provisioner.io
  parameters:

replication.storage.openshift.io/replication-secret
-name: secret-name

replication.storage.openshift.io/replication-secret
-namespace: secret-namespace
```

Recovery Management



- Recover/Failover:
  - Create a new VolumeReplication CR in cluster West
  - Set it as primary
  - Use the same dataSource as replicationState
  - Changes will be detected by the sidecar
  - GRPC to pass information to the driver
- Relocate/Failback:
  - Update replicationState to *secondary* in cluster West
  - Update replication State to primary in cluster East after it is recovered.
  - Changes will be detected by the sidecar
  - GRPC to pass information to the driver

Dynamic Provisioning Conundrum



- Initial deployment
  - User created PVCs are provisioned dynamically
- Recover/Relocate:
  - PVCs need to reattach to the replicated volume
  - Dynamic provisioning, without PVC *spec.dataSource* hints would not work!?
  - Orchestration/changes required:
    - Shift to static provisioning post initial deployment
      - PV bound to PVC is recovered on peer clusters
    - Assumption is that PV *spec.csi.volumeHandle* is reusable across storage systems
    - Ordering resource creation operations as in (1) (2) and then (3)

Multi-cluster Management

Cross Cluster Actions

DR requires managing multiple k8s clusters:

- Cluster configuration
  - Cluster configuration equivalence
  - Cluster custom resources and operators equivalence
  - Storage setup (replication setup across homogeneous storage systems)
- Application Recovery/Relocation
  - User access to congruent namespaces
  - Declarative copy of application manifests
  - Rerouting inbound application traffic across clusters (GTM (re)configuration)
- Health monitoring for alerting
- Recovery/Relocation orchestration

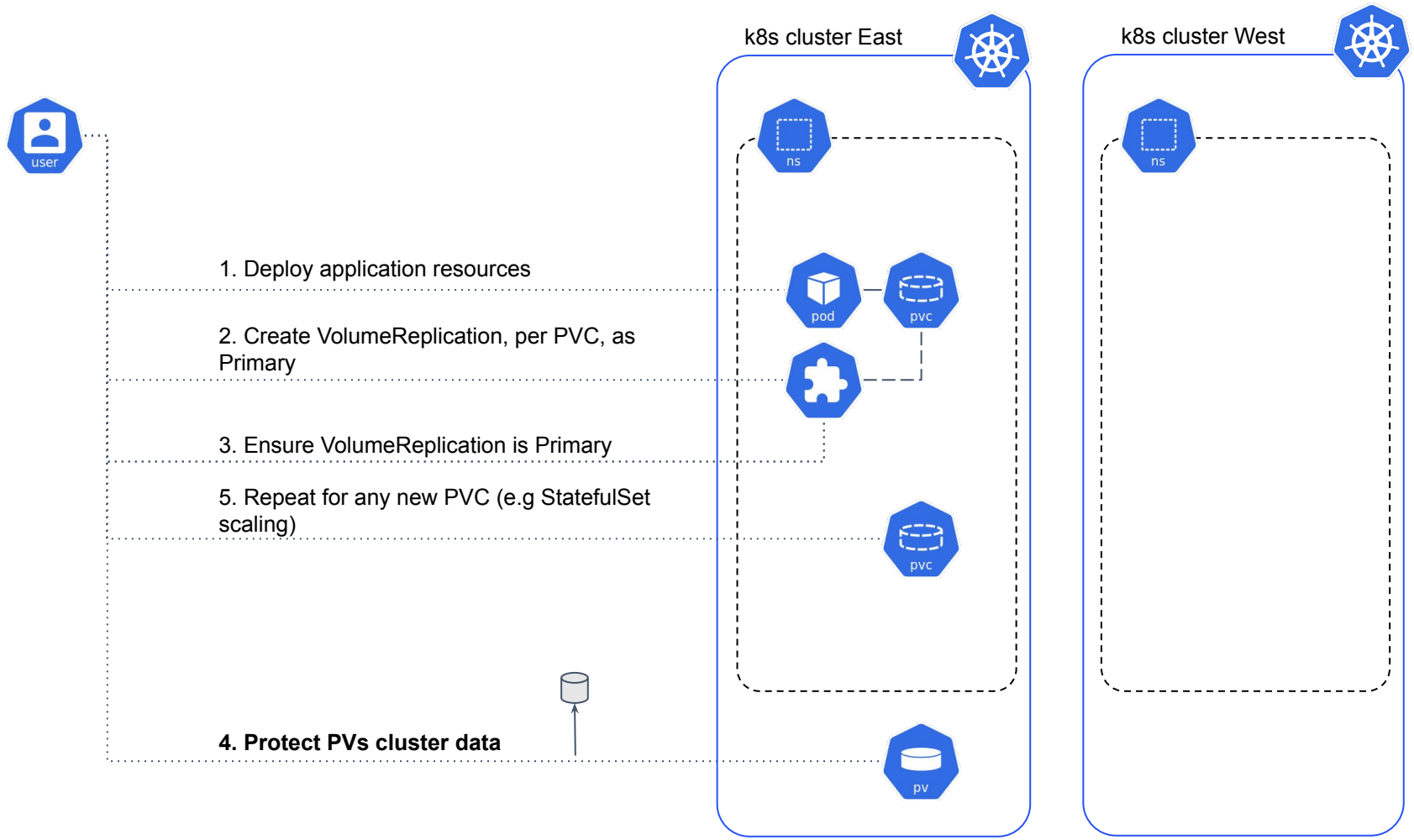
Open Cluster Management - 101

"Open Cluster Management is a community-driven project focused on multi-cluster and multicloud scenarios for Kubernetes apps."

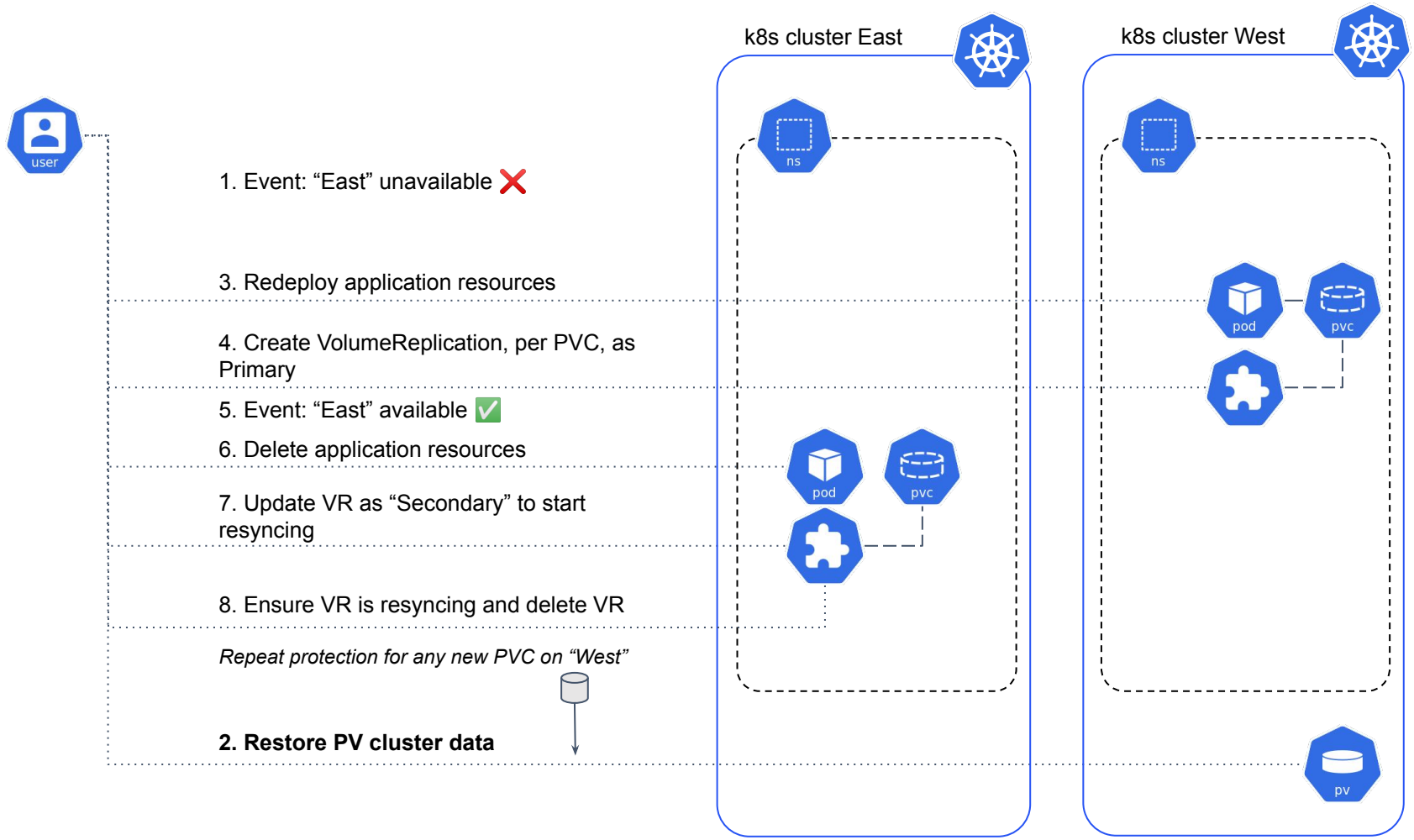
- Provides:
  - Cluster registry, Work distribution, Content placement, Vendor neutral APIs
- Leveraged for:
  - Cluster configuration
  - Application lifecycle management
    - Application manifests from a declarative OCM Channel CRD (git/helm/object store)
    - Application placement using OCM PlacementRule CRDs, which determine cluster(s) to deploy the application to
- Gaps for DR:
  - DR orchestration
  - GTM (re)configuration

Recovery/Relocate Orchestration

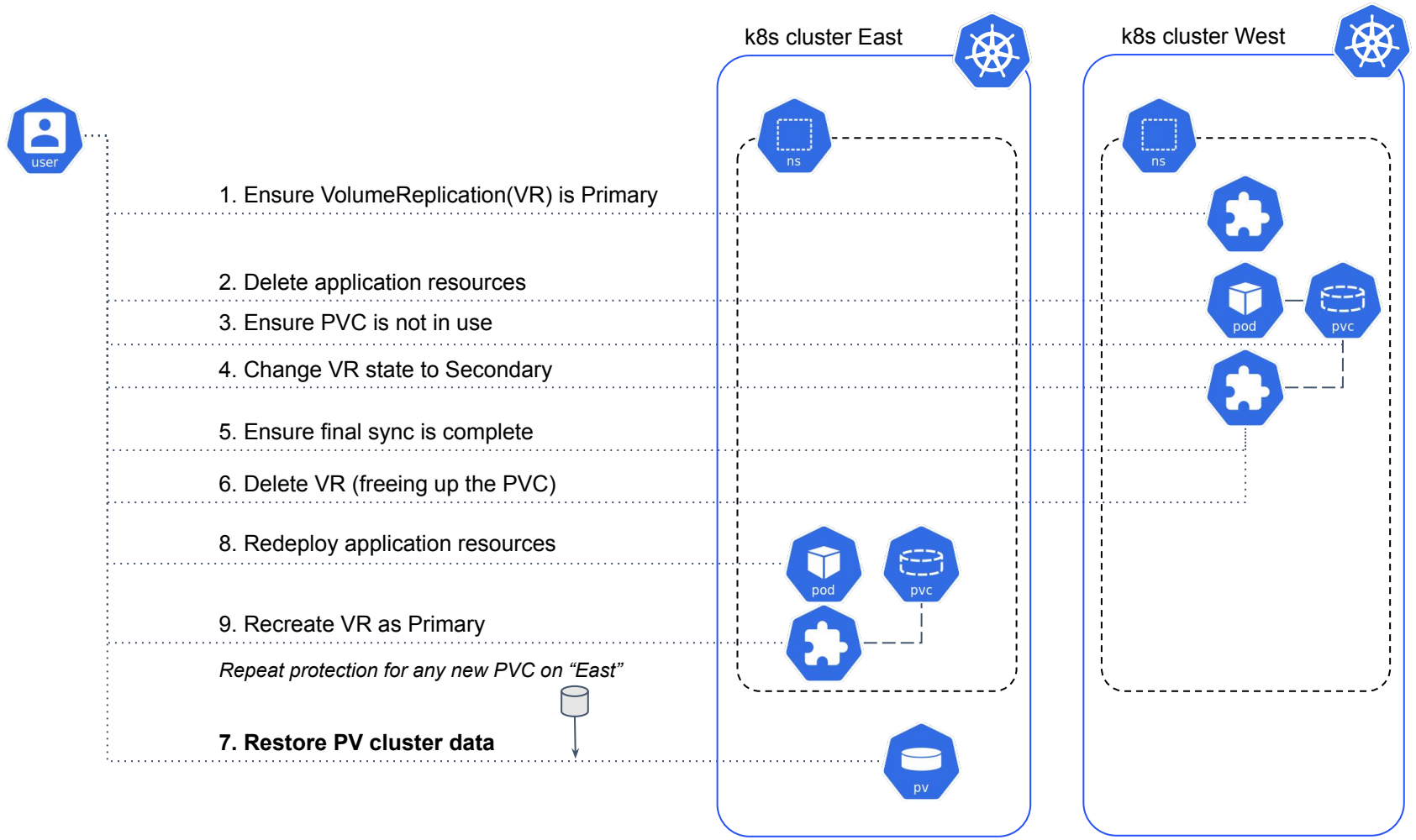
Ensuing User Complexity: Deploy



Ensuing User Complexity: Recover



Ensuing User Complexity: Relocate



DROrchestrator: Ramen

Ramen

- K8s orchestrator that provides: *"Instant Cloud-Native Workload Recovery and Relocation Across Kubernetes Clusters"*
- Orchestrates workload placement and PVC replication across k8s clusters:
  - Enhances OCM PlacementRule scheduling for DR workflows
  - Groups PVCs in an application and orchestrates their replication, leveraging VolumeReplication

Ramen: DRPolicy API

DRPolicy is a cluster scoped policy object that:

- Contains a pair of clusters that are storage DR peers in *spec.drClusterSet*
- Defines a replication schedule (RPO) using *spec.schedulingInterval*
- Optionally enables choosing a VolumeReplicationClass matching a PVCs CSI provider name, by a label selector using *spec.replicationClassSelector*

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: drpolicy-sample
spec:
  schedulingInterval: "1h"  # hourly
  replicationClassSelector:
    matchLabels:
      class: ramen
  drClusterSet:
    - name: east
      s3ProfileName: s3-profile-of-east
    - name: west
      s3ProfileName: s3-profile-of-west
```

Ramen: DRPlacementControl API

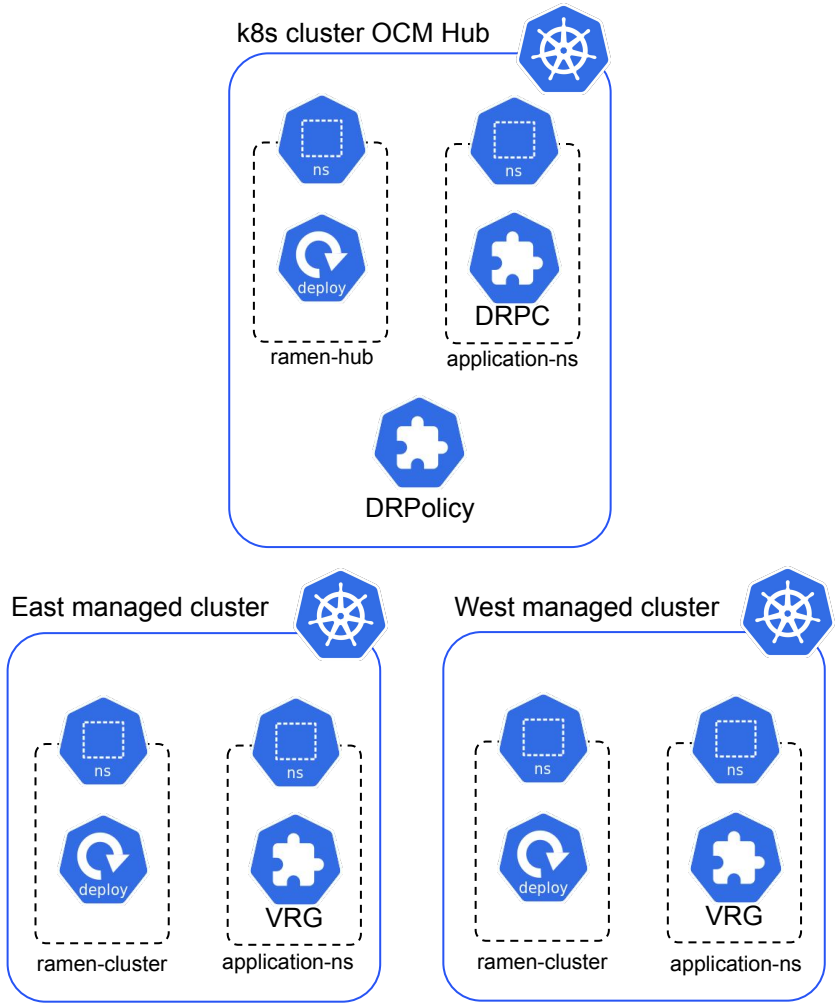
DRPlacementControl is a namespaced resource per application, that:

- Reconciles OCM PlacementRule referenced by *spec.placementRef*
- Placement and schedule for application is controlled by referenced *spec.drPolicyRef*
- Auto protects PVCs matching *spec.pvcSelector*
- Provides actions to:
  - Failover to the *spec.failoverCluster*
  - Relocate to the *spec.preferredCluster*

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPlacementControl
metadata:
  name: drplacementcontrol-sample
  namespace: application-namespace
spec:
  preferredCluster: "east" (optional)
  drPolicyRef:
    name: drpolicy-sample
  placementRef:
    kind: PlacementRule
    name: application-placement-rule
  pvcSelector:
    matchLabels:
    any-pvc-label: value
failoverCluster: [cluster-name] (optional)
action: [Failover|Relocate] (optional)
```

Ramen: Operator Deployments

- Operator at the OCM Hub (multi-cluster control plane)
  - Reconciles DRPlacementControl (DRPC)
  - Orchestrates VolumeReplicationGroup (VRG) resource on managed clusters (“east”/”west”)
- Operator at the OCM managed clusters
  - Reconciles VolumeReplicationGroup (VRG) resource
  - Ensure in-cluster VR/PVC states are orchestrated as required




KubeCon
CloudNativeCon
North America 2021
RESILIENCE
REALIZED

Demo


KubeCon
CloudNativeCon
North America 2021
RESILIENCE
REALIZED

Future Work

Future work and adaptations

- MetroDR use cases
  - Assumption is that storage replication is synchronous (no(?) VolumeReplication required)
  - Requires application orchestration for recovery and relocation on k8s cluster loss
  - Will potentially require storage fencing
- Leverage AnyVolumeDataSource feature gate, to use VolumeReplication as a data source for PVCs
- Improve replication consistency
  - Currently storage assisted replication is only crash consistent
  - Adapt to WIP application consistent snapshots when available

Future work and adaptations...

- Improve volume grouping and replication consistency across a group
  - Applications using multiple PVCs may require that these are point in time consistent
  - Adapt to WIP VolumeGroups proposal when available
- Move towards more storage agnostic data replication schemes
  - Proposals like “Change Block Tracking” can enable shorter RPOs for replication across storage vendors
  - Provide pluggability to add any replication scheme other than VolumeReplication management

Links and References

- Rook
- Ceph
- Open cluster management (OCM)
- VolumeReplication CSI extension
- Ramen orchestrator


KubeCon
CloudNativeCon
North America 2021
RESILIENCE
REALIZED

Thank you!