HITACHI

Hitachi Vantara

Hitachi VSP One Hybrid Cloud Storage Architecture for Replication on Red Hat OpenShift

Reference Architecture Guide

MK-SL-389-01
November 2025

Feedback

Hitachi Vantara welcomes your feedback. Please share your thoughts by sending an email message to Docs-Feedback@hitachivantara.com. To assist the routing of this message, use the paper number in the subject and the title of this white paper in the text.

Revision history

Changes	Date
▪ Minor updates.	November 2025
▪ Initial release.	August 2025

Contents

Reference Architecture Guide....4
References....4
Executive summary....5
Overview....6
Benefits....7
Key components....7
Test environment....10
Test scenarios....10
Guidelines and recommendations....12
Validation results....12
Test 1: Prepare the environment....13
Deploy Red Hat OpenShift Cluster....13
Enable iSCSI in worker nodes of OpenShift Clusters....15
Configure storage systems....15
Install HSPC....16
Create a storage class....17
Deploy Red Hat Advanced Cluster Management....19
GitLab installation....21
Red Hat OpenShift GitOps installation....23
Test 2: Hitachi Replication Plug-in for Containers (HRPC) installation and configuration....25
Prepare manifests for HRPC installation....25
Create a namespace for HRPC installation....29
HRPC installation....29
Test 3: Data mobility of a stateful application....35
Deploy a stateful MongoDB application on the primary cluster....35
Replicate the volume (PVC) using Universal Replicator, orchestrated by HRPC....43
Test 4: Disaster Recovery solution....52
Failover operation....53
Failback operation....60
Conclusion....71

Reference Architecture Guide

Introduction

This solution emphasizes delivering the Hitachi hybrid cloud storage solution for replication use cases, disaster recovery (DR), and data mobility. This reference architecture documents how to set up a disaster recovery and container data mobility solution with Universal Replicator (UR) using Hitachi Replication Plug-in for Containers (HRPC). Additionally, the document includes test procedures to validate the resiliency of the solution, which you can leverage for your own proof-of-concept before deployment.

The environment used for validation includes two sites. The primary site consists of a Hitachi VSP One Block 24 storage system and a Red Hat OpenShift cluster on BareMetal located in a traditional, on-premises data center in Colorado. The secondary site consists of a VSP One Block 28 storage system and a Red Hat OpenShift cluster on BareMetal located in an Equinix colocation data center in California. The two sites are approximately 1,290 miles apart.

Intended audience

This document is intended for Hitachi Vantara staff and IT professionals of Hitachi Vantara customers and partners who are responsible for planning and deploying such solutions.

Value proposition

This solution ensures seamless business continuity with block-level replication that maintains operations during cluster or storage failures. Near-cloud data placement offers vendor flexibility and proximity, and eliminates cloud lock-in. This solution also connects hybrid cloud environments while ensuring guaranteed data availability across clusters, eliminating the need for dedicated data migration software.

Red Hat Advanced Cluster Management for Kubernetes, combined with Hitachi Storage Plug-in for Containers (HSPC), and Hitachi Replication Plug-in for Containers (HRPC), simplifies hybrid cloud operations with centralized cluster management and seamless stateful application deployment. It enables block-level replication directly through storage systems, eliminating the need for dedicated data movement software, while offering robust policy enforcement for Red Hat OpenShift cluster configuration and application deployment.

This solution takes advantage of Universal Replicator (UR), enabling seamless replication between storage systems across the globe in a cloud-native environment without impacting the performance of the I/O operations at the primary site.

References

- Red Hat OpenShift Documentation
- Hitachi Storage Plug-in for Containers Reference Guide

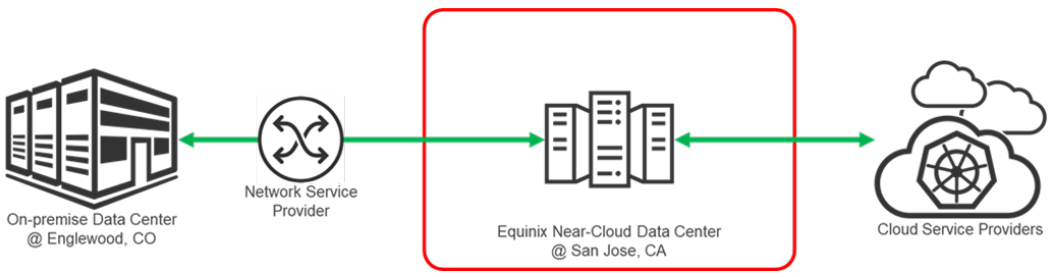
- Hitachi Replication Plug-in for Containers Configuration Guide
- Red Hat Advanced Cluster Management
- GitLab Documentation

Executive summary

- The Hitachi V**irtual** S**torage** P**latform** One family **of storage** can be deployed across a hybrid cloud landscape. This hybrid cloud storage architecture enables solutions for DevOps needs and data services that rely on replication.
- Enterprise Kubernetes distributions (such as OpenShift) are a key element in the process of modernizing customers' application landscapes. It delivers advantages in terms of agility and accelerating the pace of innovation. Containers enhance the efficiency of computing units and elevate portability. Stateless containers have less dependency in portability; however, stateful containers have complexity in portability because of dependencies with persistent volumes. There are challenges not only in deploying persistent volumes, but also in ensuring that the data is protected, available, and moveable. Hitachi Replication Plug-in for Containers (HRPC), along with Hitachi Storage Plug-in for Containers (HSPC), addresses such challenges for production-focused stateful containers.
- UR provides a solution to avoid situations where a data center is affected by a disaster that stops operations for a long period of time. UR asynchronous replication is the ability to replicate data over very long distances without impacting the performance of the I/O operations at the primary site
- Using Red Hat Advanced Cluster Management for Kubernetes, Red Hat OpenShift GitOps and HRPC, data mobility, and disaster recovery (DR) activities can be efficiently managed on multiple OpenShift clusters through a single console, making it a robust solution.
- Because the cost of owning and operating a second data center for the purpose of disaster recovery is not insignificant, leasing a small footprint in a colocation data center is a cost-attractive alternative. Equinix, Inc.™ is a leading provider of such services. The key reason for this is that the Equinix data center hosts hardware from major telecommunication companies and cloud providers. By operating in such near-cloud collocations, Equinix customers can connect to cloud providers and other businesses at high speed and low latency.
- The Equinix colocation was selected because it offered high-speed and low latency connections to the major hyperscalers, such as AWS and Azure.
- If you want to discuss hosting these types of solutions at Equinix, contact your Hitachi Vantara sales team.

Overview

- Data persistence is essential for many applications, including databases like MariaDB, PostgreSQL, MongoDB, and MySQL, as well as CI/CD pipelines. Container Storage Interface (CSI) integrations can provide persistent storage for stateful applications. Integrating Hitachi storage solutions with Red Hat OpenShift addresses the key challenges to provide robust storage support for the production deployment of stateful applications.
- Hitachi Replication Plug-in for Containers automates storage replication between two OpenShift clusters accessing storage systems located at different sites. It provides replication services for the persistent volumes on VSP storage systems and is used for disaster recovery and container data mobility use cases. After successfully deploying Hitachi Replication Plug-in for Containers, volume replication can be configured for PVCs using kubectl or equivalent tools (Red Hat Advanced Cluster Management for Kubernetes).
- Hitachi Replication Plug-in for Containers integrates with the UR program product for asynchronous replication between two storage systems. UR provides a solution to avoid situations where a data center is affected by a disaster that stops operations for a long period of time. Two storage systems are required in UR implementation. Typically, the secondary storage system is located in a second data center that is far from the first data center that contains the primary storage system. It is important to locate the two data centers far from each other to reduce the chance that a single disaster brings down both data centers' continuous server I/O when a failure prevents access to a data volume, as shown in the following illustration.
- Red Hat OpenShift is an application platform that leverages the power of Kubernetes and combines reliable and proven services to make the process of developing, modernizing, deploying, running, and managing applications more streamlined. OpenShift ensures a uniform user experience whether applications are deployed on public-cloud, on-premises, hybrid-cloud, or edge architecture.
- Red Hat Advanced Cluster Management for Kubernetes enables centralized management of multiple clusters through a single console and supports application deployment across clusters in a hybrid cloud environment. It also provides a set of policies that can be enforced on OpenShift clusters for cluster configuration and application deployment.
- To summarize, our hybrid cloud environment consists of the following two domains. The relationship between the domains is shown in the following illustration.
  - An on-premises data center, located in Englewood, Colorado.
  - A near-cloud Equinix colocation data center (named SV5), located in San Jose, California.



Benefits

- Keeps the data at the near-cloud location to ensure data availability to any cloud vendor at close proximity and avoids cloud locking.
- Closes the gap between hybrid cloud environments and guarantees data availability across different clusters.
- Allows a business to resume operations when a disaster brings down a cluster environment or the storage system.
- Hitachi VSP Block series delivers enterprise-class block storage for all mission-critical workloads, from legacy databases to modern applications. Its NVMe all-flash scalable architecture with symmetric active-active controllers ensures you can use VSP One Block to efficiently accelerate and consolidate workloads at any scale.
- The patented Hitachi Vantara Adaptive Data Reduction (ADR) increases effective capacity without slowing storage performance. VSP One Block continuously analyzes data during writes and selects and switches between real-time and deferred processing for data reduction. VSP One Block models with Compression Accelerator Modules (CAMs) offloads data reduction processing workloads from controller processors.
- VSP One Block 20 features the new patented Dynamic Drive Protection (DDP) technology that adopts an efficient data distribution method that balances workloads between the drives.

Key components

The following is a list of major components of the solution.

- Red Hat OpenShift Container Platform: Red Hat OpenShift Container Platform (OCP) provides a single platform to build, deploy, and manage applications consistently across on-premises and hybrid cloud deployments. It also provides administrator views to deploy operators, monitor container resources, manage container health, manage users, work with operators, manage pods and deployment configurations, as well as define storage resources. With OCP kubectl, a native binary of Kubernetes is complemented by the oc command which provides further support for OCP resources, such as deployment and build configurations, routes, image streams, and tags. OCP provides a GUI and a CLI interface for managing clusters.
- Red Hat Advanced Cluster Management for Kubernetes: Red Hat Advanced Cluster Management for Kubernetes provides centralized management and governance of multiple Kubernetes clusters across hybrid and multi-cloud environments.
- Hitachi Storage Plug-in for Containers (HSPC): HSPC is a software component that contains libraries, settings, and commands that you can use to create a container to run stateful applications. It enables stateful applications to persist and maintain data after the lifecycle of the container has ended. Storage Plug-in for Containers integrates Kubernetes or OpenShift with Hitachi storage systems using Container Storage Interface (CSI).

- Hitachi Replication Plug-in for Containers (HRPC): A plugin from Hitachi used to automate storage replication between two OpenShift clusters and storage systems located at different sites. Hitachi Replication Plug-in for Containers facilitates the replication of storage volumes by creating a Replication Custom Resource (CR). Hitachi Replication Plug-in for Containers starts replicating the specified PVC and triggers the creation of a PVC in the secondary site. Data in the target PVC is copied to the secondary site and is protected by UR.
- Universal Replicator: UR lives in the microcode, also known as firmware, of the storage system. It does not require any additional hardware. However, UR must be activated by a license key on the primary and secondary storage systems.
- VSP storage systems: Virtual Storage Platform One (VSP One) Block offers worry-free enterprise-class block storage with certified sustainability for mission-critical applications. A VSP One Block 24 storage system was used for persistent volume in primary site and a VSP One Block 28 storage was used for persistent volume in secondary site.
- GitLab: GitLab is a web-based Git repository that provides free open and private repositories, issue tracking capabilities, and wikis. It is more than just a Git repository manager—it's a complete DevOps platform that enables development teams to collaborate productively, automate repetitive tasks, and deliver software faster with better quality.
- Red Hat OpenShift GitOps: Red Hat OpenShift GitOps is a declarative continuous delivery platform based on Argo CD. Argo CD automates the deployment process by initiating pull requests from Git repositories and deploying applications into Kubernetes namespaces. Deployment targets can be scoped for local or remote clusters. After deploying an application, Argo CD monitors the health of the deployment and can roll back to the previous versions if necessary. Argo CD operates on declarative principles, ensuring that the deployed applications align with the application configurations stored in Git repositories, which serve as the "single source of truth" for application configurations.
- Network switches: Cisco Nexus 9000 Series switches were used to connect the two data centers.
  - 10/25Gbase-LR-S Optics: Long Range transceivers required to connect long distances
  - Single-Mode Fiber Cables: Required for long-distance communications.

Hardware and software

The following table lists the hardware specifications used in this solution.

Item	Description	Version	Function
Hitachi VSP One Block 24	(2) 12-core 2.1GHz Sapphire Rapids 256 GB cache, 1× RAID 6 (6D +2P), DDP PG 3 × 10 GbE iSCSI ports	SVOS RF 10.3.1 A3-03-01-40/01	Storage system at the primary site
Hitachi VSP One Block 28	(2) 32-core 2.3GHz Sapphire Rapids	SVOS RF 10.3.1 A3-03-01-40/01	Storage system at the secondary site

Item	Description	Version	Function
	1 TB cache, 1 × RAID 6 (6D +2P), DDP PG 3 × 10 GbE iSCSI ports		
Hitachi Advanced Server HA820 G3	(2) Intel(R) Xeon(R) Gold 6416H CPU @ 2.20GHz (1) HPE Eth 10/25Gb 2p 631FLR-SFP28 Adapters	System ROM: U30 v3.10	5-nodes Red Hat OpenShift cluster at the primary and secondary site
Cisco Nexus 93180YC-EX	(48) 1/10/25-Gbps fiber ports (6) 40/100-Gbps QSFP28 ports	9.3(13)	Network switch at the primary data center
Cisco Nexus C93180YC-FX	(48) 1/10/25-Gbps fiber ports (6) 40/100-Gbps QSFP28 ports	10.4(3)	Network switch at the secondary data center

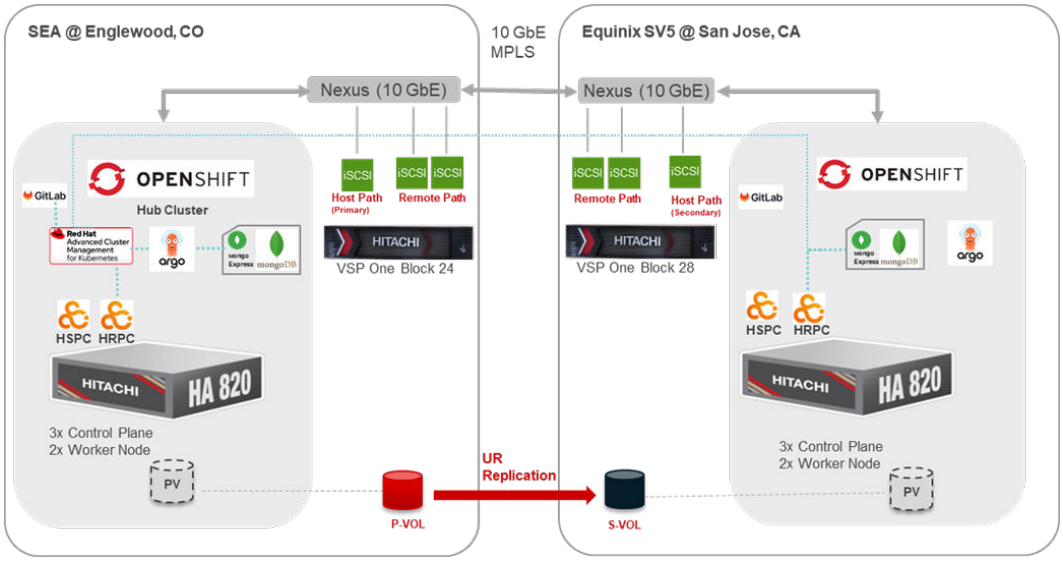
The following table lists the software specifications used in this solution.

Item	Version	Function
Red Hat OpenShift Cluster	4.16	Red Hat OpenShift Cluster deployed at the primary and secondary site
Hitachi Storage Plug-in for Containers (HSPC)	3.14.3	Hitachi Storage Plug-in for Containers is a software for creating and managing persistent volumes for Hitachi storage systems in a Kubernetes environment
Hitachi Replication Plug-in for Containers (HRPC)	1.5.0	Hitachi Replication Plug-in for Containers (HRPC) automates storage replication between two different Kubernetes clusters and storage systems located at different sites
Red Hat Advance Cluster Management for Kubernetes	2.11.3	Red Hat Advanced Cluster Management for Kubernetes provides centralized management and governance for multiple Kubernetes clusters across hybrid and multi-cloud environments
GitLab	1.6.1	A DevOps CI/CD tool deployed in both the OpenShift clusters to build and store applications. Also, acts as a container registry for HRPC
Red Hat OpenShift GitOps	1.14.1	Red Hat OpenShift GitOps is a declarative continuous delivery platform based on Argo CD
MongoDB	8.0.3	A stateful database application used to validate data consistency on both on-premises and Equinix

Item	Version	Function
Mongo Express	1.0.2	Web-based MongoDB admin interface, written with Node.js and express

Test environment

The following illustration shows the test environment used for this validation.



Test scenarios

The following table lists the test scenarios performed in this validation.

Description	Success Criteria
Prepare the environment: **1.** Deploy two Red Hat OpenShift Clusters, one in an on-premises environment in Englewood, CO and another in a near cloud data center in Equinix. **2.** Define storage, network, and iSCSI connections. **3.** Use one Dynamic Provisioning (DP) pool in both storage systems to provision persistent volume for stateful application. **4.** Enable iSCSI in the worker nodes of both clusters. **5.** Configure remote paths and journal volumes in both storage systems.	Environment is set up according to specifications.

Description	Success Criteria
6. Deploy Red Hat Advanced Cluster Management in the primary cluster. **7.** Install HSPC using OperatorHub in both clusters and configure the storage class. **8.** Install GitLab and Red Hat OpenShift GitOps in both clusters. **9.** Configure the network for REST API communication between both clusters and the storage systems.	
Configure Hitachi Replication Plugin for Containers: **1.** Download Hitachi Replication Plug-in for Containers (HRPC) installation files. **2.** Prepare HRPC image and manifest files. **3.** Install HRPC from Red Hat Advanced Cluster Management.	Verify that HRPC is deployed and running in both clusters.
Data mobility: Migrate stateful application using HRPC **1.** Deploy a stateful MongoDB application with PVC from VSP One Block 24 using Red Hat Advanced Cluster Management on the UR primary site. **2.** Insert data into the application. **3.** Replicate the PVC from the UR primary site to the secondary site using Red Hat Advanced Cluster Management. **4.** Verify that the status of the replication CR is “ready”. **5.** Change the desired pair state to “split” to make the S-VOL accessible. **6.** Verify that data inserted in the primary site is available in the secondary site by deploying the stateful MongoDB application with the replicated PVC from VSP One Block 28 using Red Hat Advanced Cluster Management.	Volume replication can be performed using HRPC from on-premises to near cloud (Equinix). Verify container data mobility to secondary site is successful using Red Hat Advanced Cluster Management.
Disaster recovery: When an outage occurs in the primary site, initiate failover operation to make the application available at the secondary site and after recovery, restore the application to the primary site and re-establish the replication. Failover operation: **1.** When an outage occurs at the primary site, change the desired pair state to “failover” from the secondary site to make the S-VOL accessible. **2.** Deploy the stateful MongoDB application with the replicated PVC from VSP One Block 28 on the secondary site, from the Red Hat OpenShift GitOps Argo CD. Verify that data inserted in the primary site is available and insert new data. Failback operation:	Validate that data can be protected during an outage across the sites using HRPC and Red Hat Advanced Cluster Management.

Description	Success Criteria
1. After recovered, clean up the primary site. **2.** Create replication from the secondary site using Red Hat Advanced Cluster Management. **3.** Perform failover from the primary site. ▪ Edit the replication CR in the primary site and change the desired pair state to “failover”. ▪ Confirm that replication CR status is “failover” and the operation value is “none”. ▪ Deploy the stateful MongoDB application in the primary site with the replicated PVC using Red Hat Advanced Cluster Management. **4.** Delete the replication CR, MongoDB application and the PVC from the secondary site. **5.** Create the replication from the primary site using Red Hat Advanced Cluster Management to resume normal business operation from the primary site.	

Guidelines and recommendations

This section describes the lessons learned from this validation, along with guidelines and recommendations.

- Establish REST API connections among clusters and storage systems and ensure communication between both the sites.
- A snapshot or clone from a PVC that is controlled by HRPC is not supported.
- NVMe over Fibre Channel is not supported for HRPC.
- For designing an optimal UR solution, see the UR best practices blog.
- Red Hat Advanced Cluster Management might display a warning because of a known issue if an application is deployed in the same namespace on two different clusters. However, the deployment succeeds.
- Red Hat OpenShift is only supported on hardware platforms running on prem such as the Hitachi Integrated Systems platform, or at Equinix that have been certified on Red Hat OpenShift.

Validation results

This section shows the steps and screenshots for each test scenario.

Test 1: Prepare the environment

The test environment consists of a primary site and a secondary site. The primary site is located in an on-premises data center in Englewood and the secondary site resides in Equinix near cloud data center. The two sites are approximately 1,290 miles apart. A Red Hat OpenShift cluster was deployed on both the sites using the “Assisted Installer” method on a bare metal environment with HA820 G3.

1. Configure physical LAN and iSCSI connections for OpenShift clusters and storage systems.
2. Provision DP pool to be used for persistent volume from VSP One Block 24 and VSP One Block 28 storage systems located at primary and secondary sites, respectively.
3. Establish REST API connections among clusters and storage systems, ensuring the following:
   a. HRPC in the primary site must communicate with the Red Hat OpenShift cluster in the secondary site and vice versa.
   b. HRPC in the primary site must communicate with the storage system in the secondary site and vice versa.
   c. Connection between primary and secondary storage system using REST API.
   d. iSCSI connection between primary and secondary storage system for data copy.
   e. Admin node must communicate with storage systems and clusters.
4. To set ports for REST API and remote copy operations, see docs.hitachivantara.com.

Deploy Red Hat OpenShift Cluster

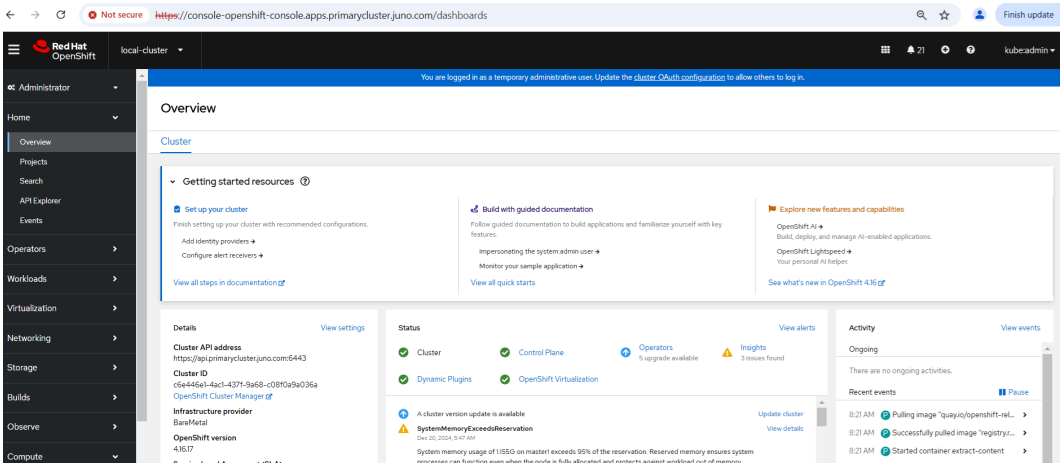
Before you begin

Both clusters were deployed on bare metal environments using the “Assisted Installer” method. For the installation process, see https://docs.openshift.com/container-platform/4.16/installing/overview/index.html.

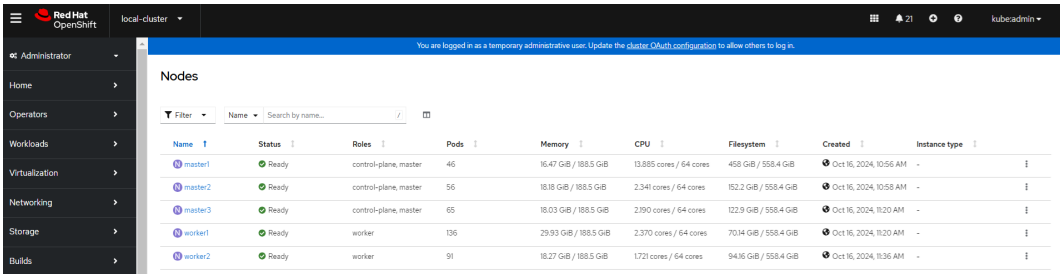
Procedure

1. When the installation completes, access the console URL.

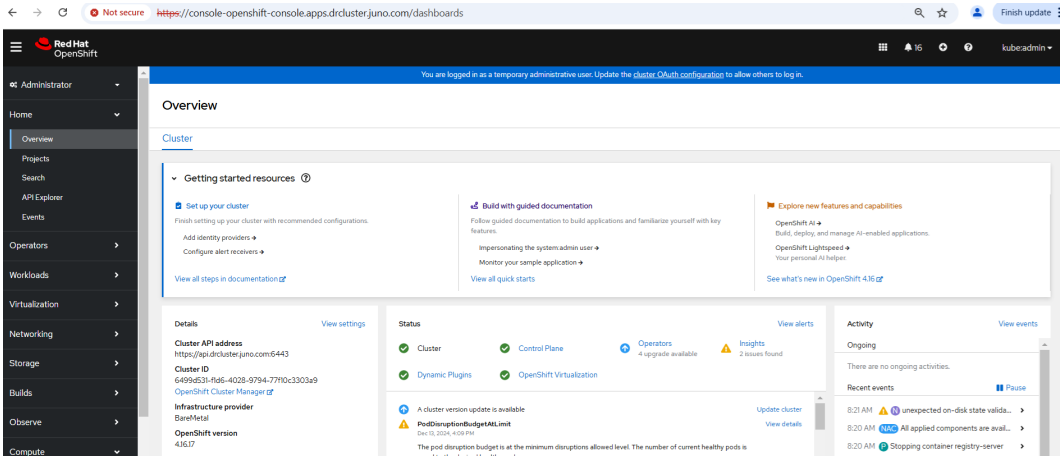
   Red Hat OpenShift Cluster on the primary site:



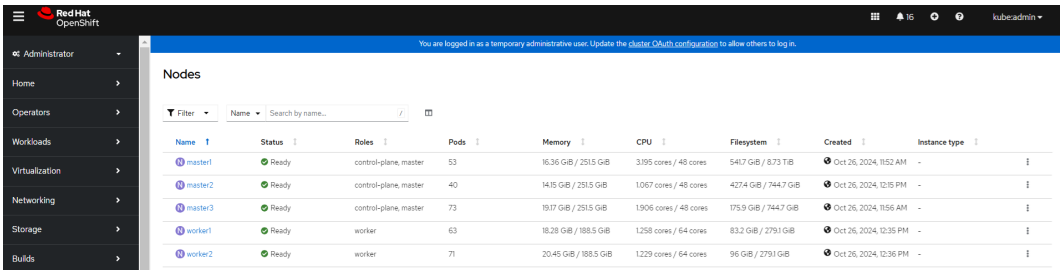
2. In the Red Hat OpenShift console, navigate to **Compute**, and then click **Nodes** to check the status of the master and worker nodes of the primary site cluster.



3. Red Hat OpenShift Cluster on the secondary site:

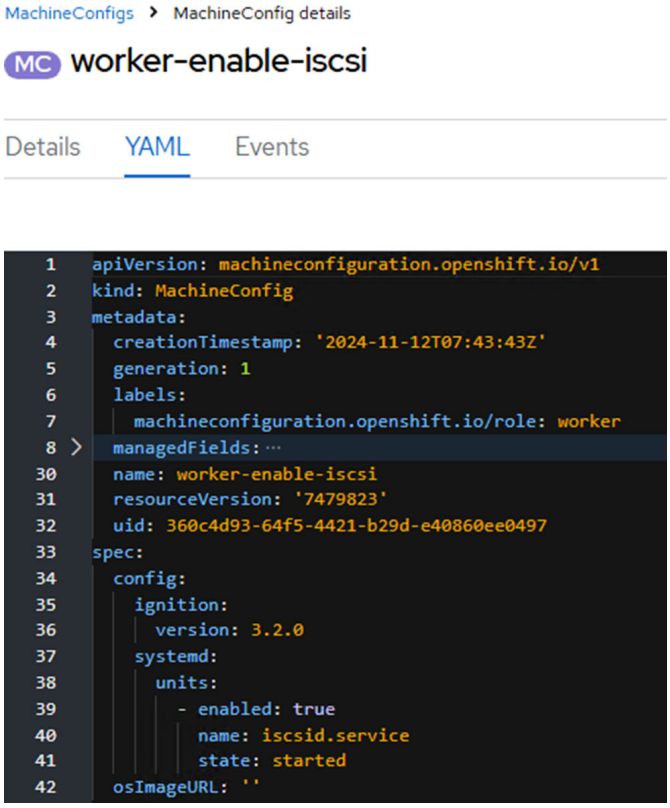


4. Check the status of the master and worker nodes of the secondary site cluster.



Enable iSCSI in worker nodes of OpenShift Clusters

To access storage using iSCSI, each of the worker nodes must have iSCSI available and running as a service. Create a MachineConfig that can enable and start the iscisd service. Log in to the OpenShift web console and navigate to Compute, click MachineConfigs, and then click Create MachineConfig. Populate the YAML file and click Create.



After a MachineConfig is created, iSCSI service will start in the worker nodes.

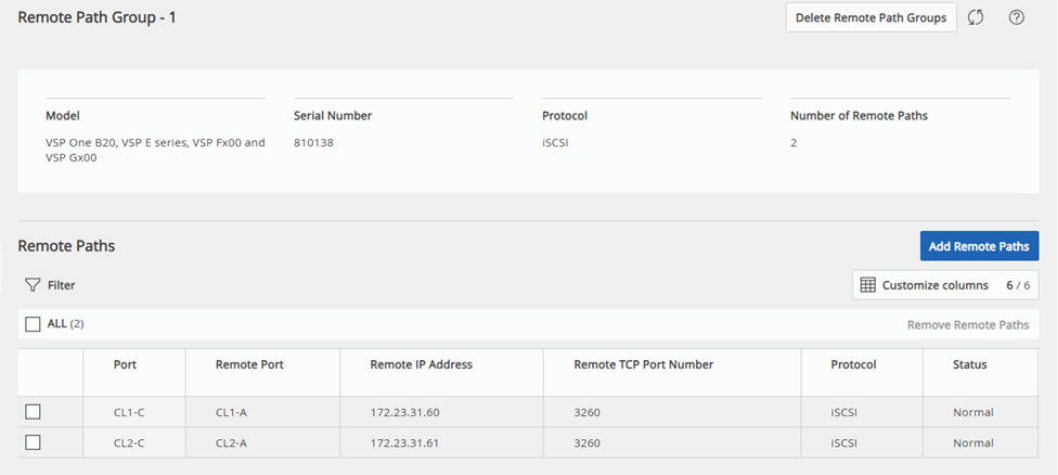
Configure storage systems

To configure the storage systems for replication, complete the following steps:

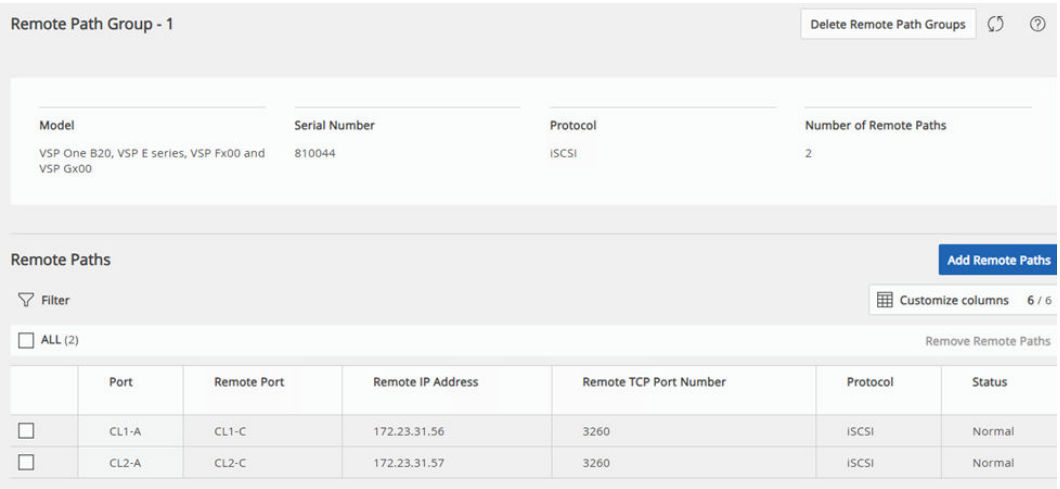
1. Configure the storage system as described in the Hitachi Storage Plug-in for Containers Quick Reference Guide at https://docs.hitachivantara.com/v/u/en-us/adapters-and-drivers/1.5.x/mk-92adptr155
2. Configure the remote paths between primary site and secondary site storage systems. For details, see the Universal Replicator User Guide at https://docs.hitachivantara.com/r/en-us/svos/10.3.x/mk-23vsp1b016.

The following screenshots show the paths between the primary and the secondary storage systems.

- Remote paths from the primary storage system (VSP One Block 24).



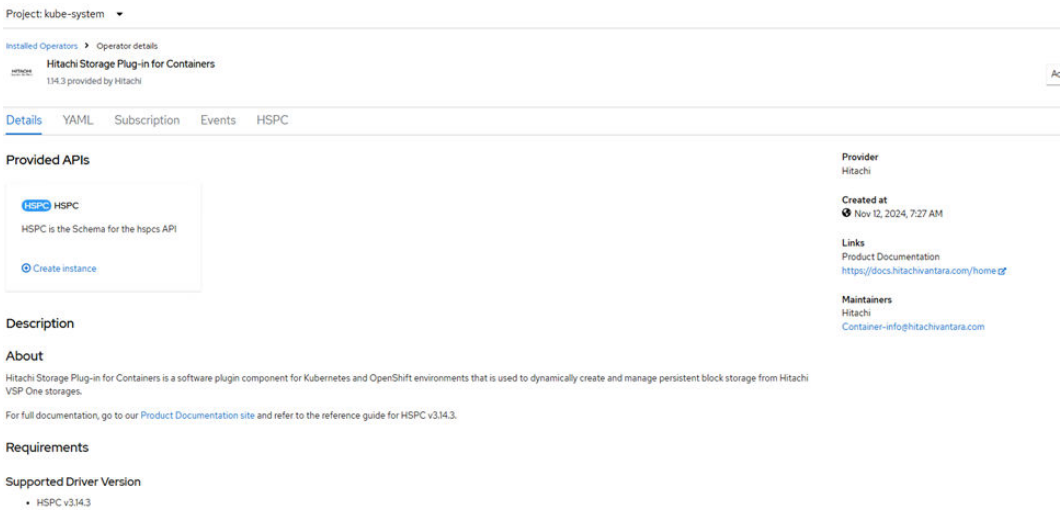
- Remote paths from the secondary storage system (VSP One Block 28).



3. Create journal volumes for UR operations on the primary and secondary storage systems. A journal volume of 200 GB is created in both the storage systems using Storage Navigator. For details, see the Universal Replicator User Guide.

Install HSPC

1. After deploying the OpenShift cluster, install the HSPC plugin using OpenShift OperatorHub. HSPC v3.14.3 is installed on both clusters.



2. Create the HSPC instance on both clusters.

Create a storage class

After installing the HSPC plugin, create a storage class on both clusters to provision persistent volumes.

1. Create a secret for HSPC on both clusters.
   a. From the Red Hat OpenShift console, navigate to Workloads, click Secret, and then click Create to open a YAML window. Enter the storage URL, username, and password in base64 format and click Create to generate a secret.

      The following screenshots show the secret YAML for the two storage systems.

VSP One Block 24

Secrets › Secret details

secret-b24

Details YAML

```yaml
kind: Secret
apiVersion: v1
metadata:
  name: secret-b24
  namespace: default
  uid: 331f51bf-175a-473d-ba36-0c919e04f015
  resourceVersion: '13529823'
  creationTimestamp: '2024-11-06T05:08:02Z'
  managedFields: ···
data:
  password: a3ViZXJuZXRlcw==
  url: aHR0cDovLzE3Mi4yMy42OC4xMzQ=
  user: a3ViZXJuZXRlcw==
type: Opaque
```

VSP One Block 28

Secrets › Secret details

secret-b28

Details YAML

```yaml
kind: Secret
apiVersion: v1
metadata:
  name: secret-b28
  namespace: default
  uid: ec0010e9-ef53-4f15-bc33-3925ec5feedd
  resourceVersion: '7477172'
  creationTimestamp: '2024-11-12T07:35:42Z'
  managedFields: ···
data:
  password: a3ViZXJuZXRlcw==
  url: aHR0cDovLzE3Mi4yMy4zMC4xMDg=
  user: a3ViZXJuZXRlcw==
type: Opaque
```

2. Create a storage class. Refer to following to create a storage class in both primary and secondary clusters. From the Red Hat OpenShift console, navigate to Storage, click StorageClasses, and then click create StorageClass.
   a. The name and “fstype” of the storage class must be same in both clusters.
   b. The storage class in the primary cluster must point to the storage in the primary site.
   c. The storage class in the secondary cluster must point to the storage in the secondary site.

VSP One Block 24

StorageClasses › StorageClass details

sc-hrpc

Details YAML

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: sc-hrpc
  uid: ee8ba470-39fe-4cf2-9541-f3ca9f0e66c4
  resourceVersion: '13561017'
  creationTimestamp: '2024-11-06T05:50:19Z'
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
    storageclass.kubernetes.io/is-default-class: 'true'
  managedFields: ···
provisioner: hspc.csi.hitachi.com
parameters:
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: default
  csi.storage.k8s.io/provisioner-secret-name: secret-b24
  csi.storage.k8s.io/node-stage-secret-name: secret-b24
  csi.storage.k8s.io/controller-expand-secret-name: secret-b24
  csi.storage.k8s.io/node-publish-secret-namespace: default
  csi.storage.k8s.io/controller-publish-secret-name: secret-b24
  csi.storage.k8s.io/controller-publish-secret-namespace: default
  poolID: '0'
  csi.storage.k8s.io/node-publish-secret-name: secret-b24
  connectionType: iscsi
  csi.storage.k8s.io/controller-expand-secret-namespace: default
  portID: CL3-C
  serialNumber: '810044'
  csi.storage.k8s.io/node-stage-secret-namespace: default
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

VSP One Block 28

StorageClasses › StorageClass details

sc-hrpc

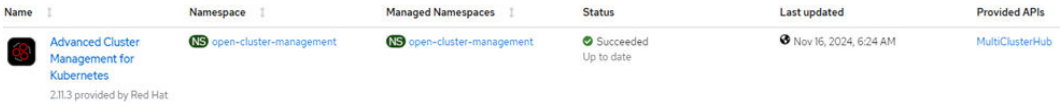
Details YAML

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: sc-hrpc
  uid: 50c63936-5c81-4e26-b4c5-cc4047cb8fa6
  resourceVersion: '7480985'
  creationTimestamp: '2024-11-12T07:39:23Z'
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
    storageclass.kubernetes.io/is-default-class: 'true'
  managedFields: ···
provisioner: hspc.csi.hitachi.com
parameters:
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: default
  csi.storage.k8s.io/provisioner-secret-name: secret-b28
  csi.storage.k8s.io/node-stage-secret-name: secret-b28
  csi.storage.k8s.io/controller-expand-secret-name: secret-b28
  csi.storage.k8s.io/node-publish-secret-namespace: default
  csi.storage.k8s.io/controller-publish-secret-name: secret-b28
  csi.storage.k8s.io/controller-publish-secret-namespace: default
  poolID: '0'
  csi.storage.k8s.io/node-publish-secret-name: secret-b28
  connectionType: iscsi
  csi.storage.k8s.io/controller-expand-secret-namespace: default
  portID: CL3-A
  serialNumber: '810138'
  csi.storage.k8s.io/node-stage-secret-namespace: default
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

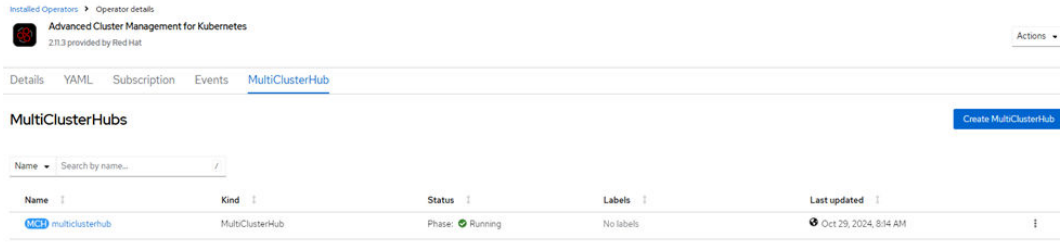
Deploy Red Hat Advanced Cluster Management

Red Hat Advanced Cluster Management v2.11.3 is installed in the primary cluster in Englewood.

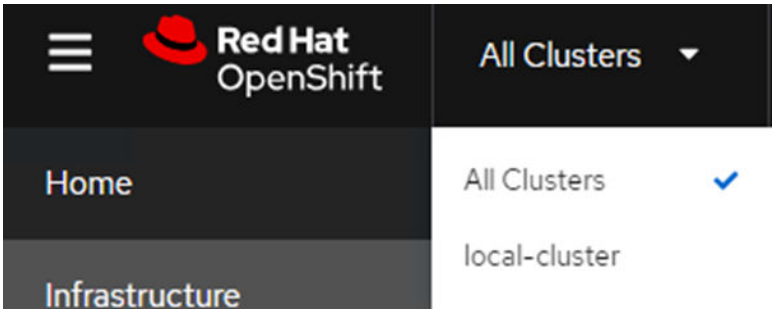
1. Log in to the Red Hat OpenShift console and select Operators and click OperatorHub. Search "Advanced Cluster Management for Kubernetes". In the Install Operator window, enter the required information and click Install.



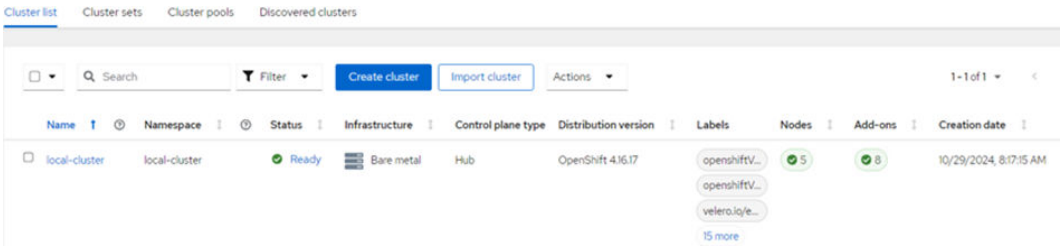
2. Wait until the installation is successful and then click Create MultiClusterHub. In the Create MultiClusterHub window, select the default name and click Create.



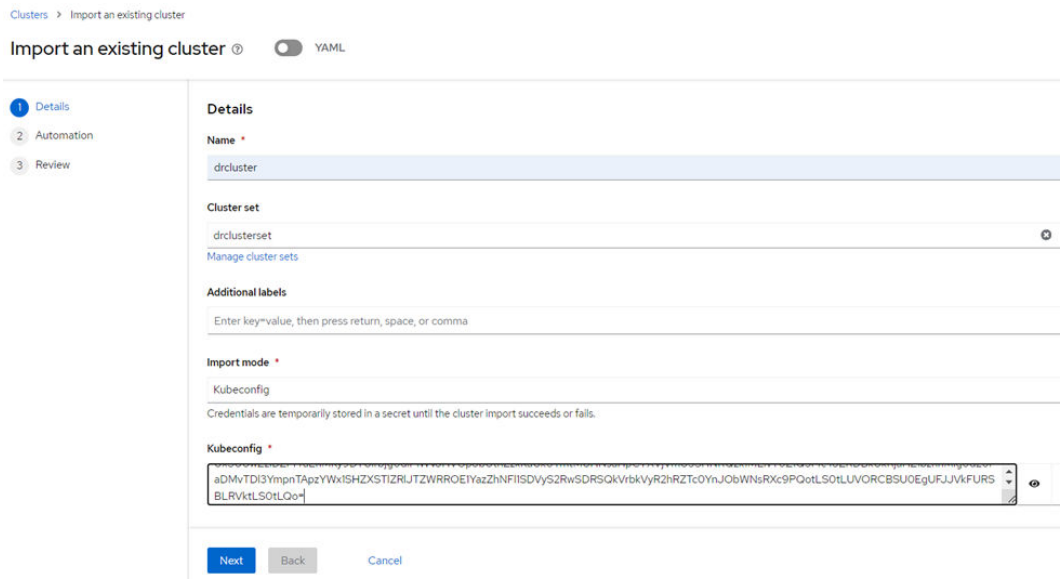
3. In the Red Hat OpenShift console, select Home, and then click All Clusters.



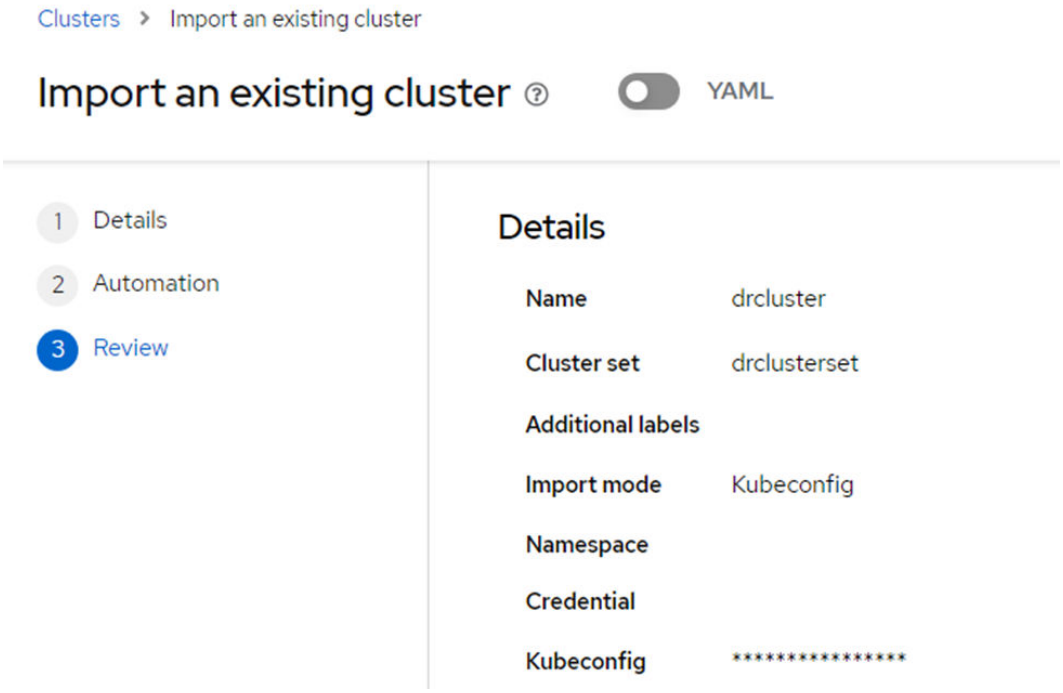
4. In the Red Hat OpenShift console, in the All Clusters menu, select Infrastructure, then select clusters to view the list of available clusters. The local-cluster shows ready and available.



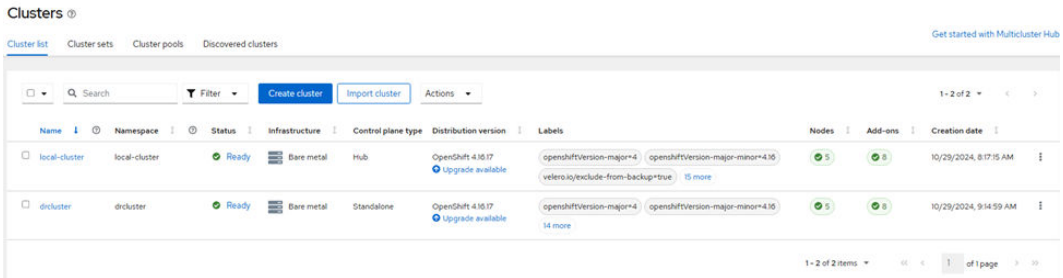
5. Before importing the secondary cluster in the MultiCluster hub, create a cluster set for the secondary cluster. In the Cluster sets tab, click Create cluster set, name it “drclusterset”.
6. In the MultiCluster hub, click Import cluster to import the secondary cluster that resides at Equinix. Set the cluster name as “drcluster”, select the cluster set created previously (“drclusterset”), use the import mode as Kubeconfig, and write the secondary kubeconfig content in the Kubeconfig field. Click Next.



7. Skip Automation, and then click Import.



8. Now both clusters are discovered in the hub cluster of Red Hat Advanced Cluster Management, where the primary cluster is “local-cluster” and the secondary cluster is “drcluster”.



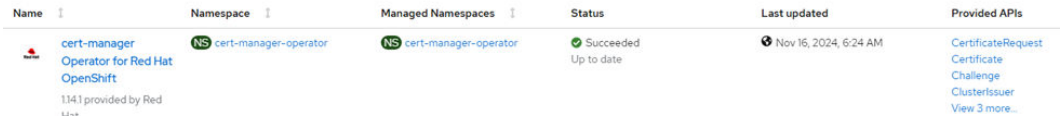
GitLab installation

GitLab is used to store manifest files, and it also act as a container registry for the HRPC image.

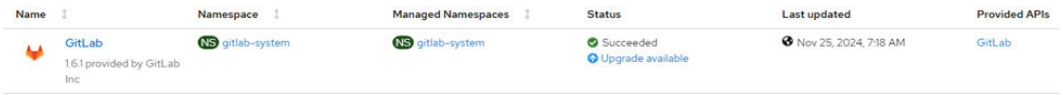
By default, “cert-manager” creates and manages TLS certificates for the GitLab-related routes.

Procedure

1. Install cert-manager from **OperatorHub** on both clusters.



2. GitLab v1.6.1 is installed on both clusters from **OperatorHub**.

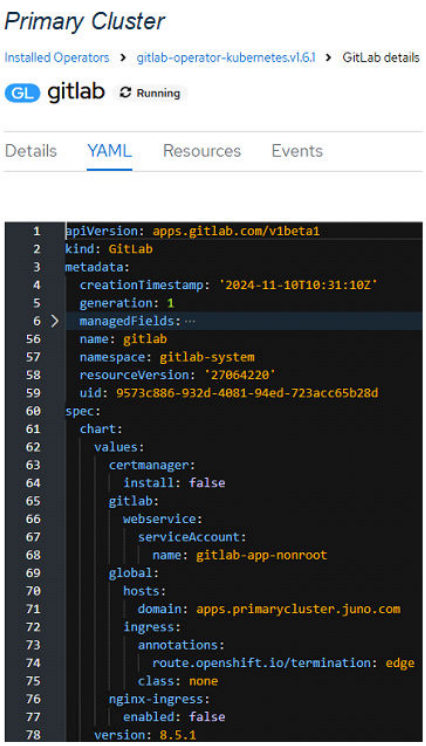


3. Add security context constraints to `gitlab-redis`, `gitlab-certmanager-issuer`, `gitlab-gitlab-runner`, `gitlab-shared-secrets`, `gitlab-prometheus-server`, and `default`.

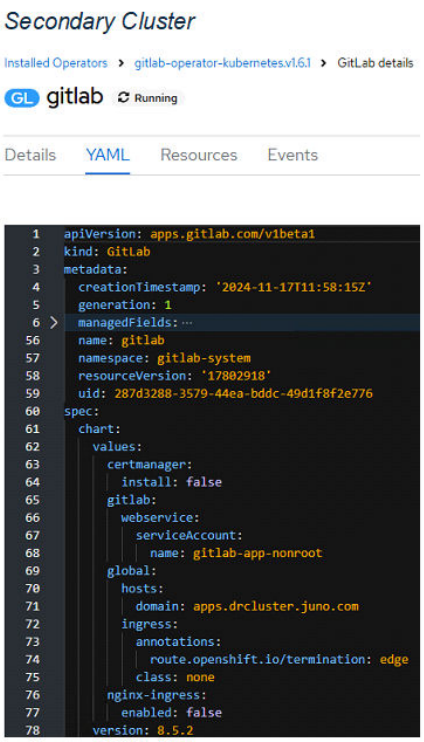
```
#oc adm policy add-scc-to-user anyuid -z gitlab-redis -n gitlab-system
#oc adm policy add-scc-to-user anyuid -z gitlab-certmanager-issuer -n #gitlab-system
#oc adm policy add-scc-to-user anyuid -z gitlab-gitlab-runner -n gitlab-system
#oc adm policy add-scc-to-user anyuid -z gitlab-shared-secrets  -n gitlab-system
#oc adm policy add-scc-to-user anyuid -z gitlab-prometheus-server -n gitlab-system
#oc adm policy add-scc-to-user anyuid -z default -n gitlab-system
```

4. Create a GitLab instance in both clusters. In the Red Hat OpenShift console, navigate to **Operators** and click **Installed Operator**. Click **GitLab**, and then click **Create GitLab**. In the YAML view, populate the content as shown and click **Create**.

Primary Cluster

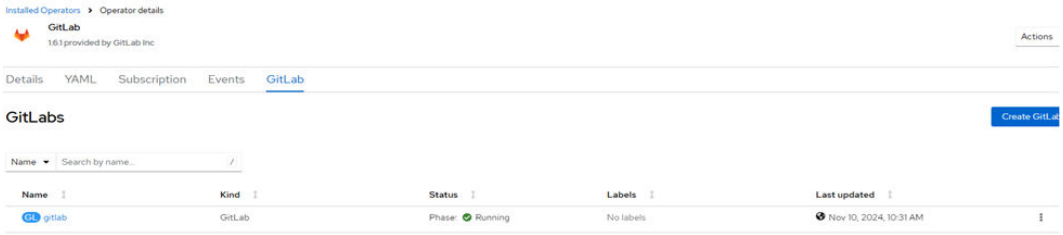


Secondary Cluster

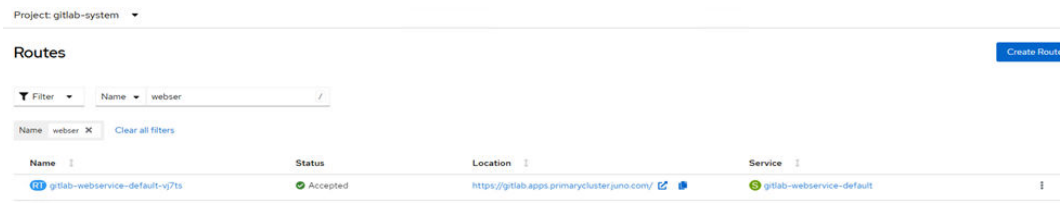


For more information see the official GitLab documentation at https://docs.gitlab.com/operator/openshift_ingress.html.

5. Verify that the GitLab instance is running on both clusters.



6. The OpenShift route is used to expose GitLab to the external network. In the Red Hat OpenShift console, select **Networking** > **Route**, and then click the route **gitlab-webserver-*** in the **gitlab-system** project. The GitLab URL is available in the **Location** field.



7. Update GitLab issuer email with any email ID.

```
#oc edit issuer gitlab-issuer -n gitlab-system
```

8. Verify that `oc get certificate,order,challenge --all-namespaces` does not throw any errors. If an error occurs, delete the certificate objects to re-issue the certificates. For more details, see https://docs.gitlab.com/charts/installation/tls.html.
9. Open the GitLab URL in the browser. The default username is `root` and you can find the password in the data field of the secret `gitlab-gitlab-initial-root-password`. Copy the password or alternatively use the following command to get the password.

```
#oc get secret gitlab-gitlab-initial-root-password  -
ojsonpath='{.data.password}' -n gitlab|  base64 --decode ; echo
```

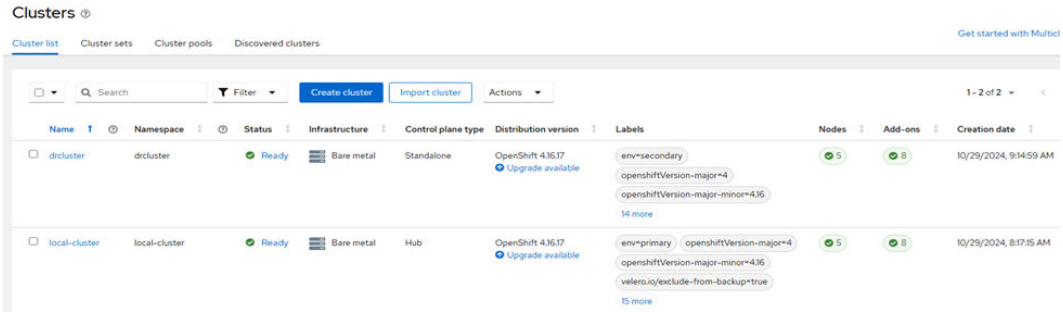
Red Hat OpenShift GitOps installation

Install Red Hat OpenShift GitOps and integrated Red Hat Advanced Cluster Management using the Argo CD.

Procedure

1. Install Red Hat OpenShift GitOps v1.14.1 on both clusters from the OperatorHub.

2. To manage clusters with the Argo CD and integrate with Red Hat Advanced Cluster Management, a set of manifest files must be applied on the primary cluster. First label the two clusters as “env=primary” for the primary clusters and “env=secondary” for the secondary cluster as shown.



3. Apply the following manifest files on the primary cluster.

ManagedClusterbinding	

`apiVersion: cluster.open-cluster-management.io/v1beta2`
`kind: ManagedClusterSetBinding`
`metadata:`
`  name: default`
`  namespace: openshift-gitops`
	`apiVersion: cluster.open-cluster-management.io/v1beta2`
	`kind: ManagedClusterSetBinding`
	`metadata:`
	`  name: drclusterset`
	`  namespace: openshift-gitops`

spec:
  clusterSet: default
	spec:
	  clusterSet: drclusterset

Placement	

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: primarygitopsplacement
  namespace: openshift-gitops
spec:
  clusterSets:
  - default
  predicates:
  - requiredClusterSelector:
  labelSelector:
  matchExpressions:
  - key: env
  operator: In
  values:
  - primary
	apiVersion: cluster.open-cluster-management.io/v1beta1
	kind: Placement
	metadata:
	  name: secondarygitopsplacement
	  namespace: openshift-gitops
	spec:
	  clusterSets:
	  - drclusterset
	  predicates:
	  - requiredClusterSelector:
	  labelSelector:
	  matchExpressions:
	  - key: env
	  operator: In
	  values:
	  - secondary

GitOpsCluster	

apiVersion: apps.open-cluster-management.io/v1beta1
kind: GitOpsCluster
metadata:
  name: primarygitopscluster
  namespace: openshift-gitops
spec:
  argoServer:
  cluster: local-cluster
  argoNamespace: openshift-gitops
  placementRef:
  kind: Placement
  apiVersion: cluster.open-cluster-management.io/v1beta1
  name: primarygitopsplacement
	apiVersion: apps.open-cluster-management.io/v1beta1
	kind: GitOpsCluster
	metadata:
	  name: secondarygitopscluster
	  namespace: openshift-gitops
	spec:
	  argoServer:
	  cluster: drcluster
	  argoNamespace: openshift-gitops
	  placementRef:
	  kind: Placement
	  apiVersion: cluster.open-cluster-management.io/v1beta1
	  name: secondarygitopsplacement

4. Add the following security context constraints to service accounts used by the Red Hat OpenShift GitOps Argo CD.

```
#oc adm policy add-role-to-user admin system:serviceaccount:openshiftgitops:
openshift-gitops-argocd-application-controller –n <application namespace>
# oc adm policy add-scc-to-user anyuid system:serviceaccount:kube-
system:jobcontroller
```

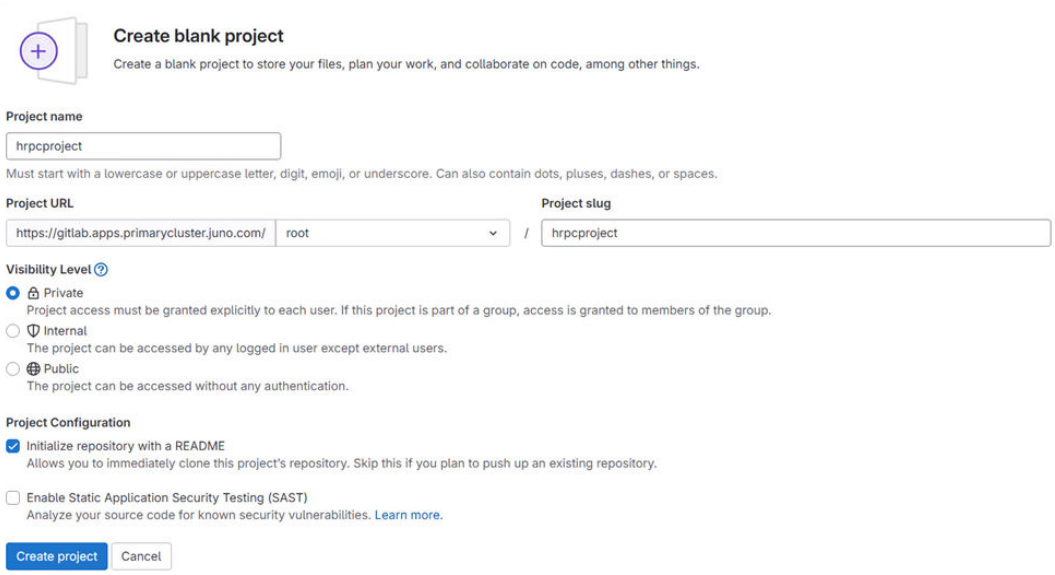
Test 2: Hitachi Replication Plug-in for Containers (HRPC) installation and configuration

Hitachi Replication Plug-in (HRPC) v1.5.0 is installed using Red Hat Advanced Cluster Management while keeping the HRPC image in the GitLab container registry and HRPC manifests in a GitLab project.

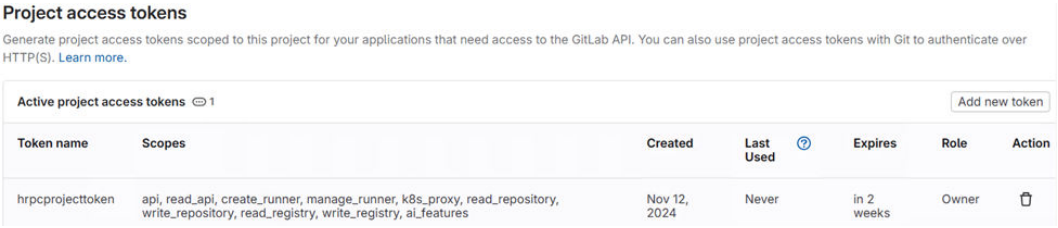
Prepare manifests for HRPC installation

Procedure

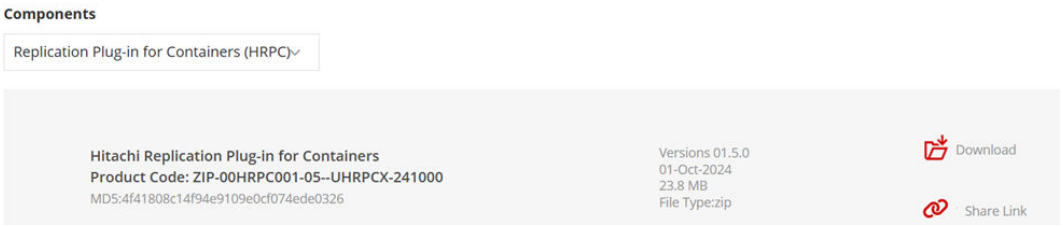
1. Create a blank project in GitLab with the name `hrpcproject`, user `root`, visibility level as `Private`, and select **Initialize repository with a README**.



2. Create an access token for this project. Select the project, click **Settings**, and then **Access tokens**. A new access token is created with all permissions. Copy the token ID. This will be required in Red Hat Advanced Cluster Management, while accessing this private project.



3. Download the HRPC plugin from https://support.hitachivantara.com. Go to **Downloads**, then **Hardware Download**, and select **Storage Plugin for Containers**. From the dropdown, select **Replication Plugin for Containers**. Then download the v1.5.0 file and copy it to the client machine, where both clusters are accessible.



This package comes with some YAML files, present in the `hrpc_1_5_0/yaml` folder and an image file located in `hrpc_1_5_0/program`.

4. Build the container image locally, then upload it to the GitLab container registry.
   a. Load the container image from the admin node.

```
[root@occlient HRPC]# podman images
REPOSITORY  TAG         IMAGE ID    CREATED     SIZE
[root@occlient HRPC]# podman load -i hrpc_1_5_0.tar
Getting image source signatures
Copying blob e16c585a5d50 done
Copying blob 10e9b9e72178 done
Copying blob 7a6477860277 done
Copying blob ff5700ec5418 done
Copying blob d52f02c6501c done
Copying blob e624a5370eca done
Copying blob 1a73b54f556b done
Copying blob 8f2b2d741d53 done
Copying blob 9959f8de3336 done
Copying blob 0ff13ec72ceb done
Copying config c3374b377f done
Writing manifest to image destination
Storing signatures
Loaded image: localhost/hitachi/hspc-replication-operator:v1.5.0
[root@occlient HRPC]# podman images
REPOSITORY                                  TAG         IMAGE ID      CREATED       SIZE
localhost/hitachi/hspc-replication-operator  v1.5.0      c3374b377ffe  2 months ago  57.1 MB
[root@occlient HRPC]#
```

   b. Log in to the GitLab container registry.

```
[root@occlient ~]# podman login registry.apps.primarycluster.juno.com --tls-verify=false
Username: root
Password:
Login Succeeded!
[root@occlient ~]#
```

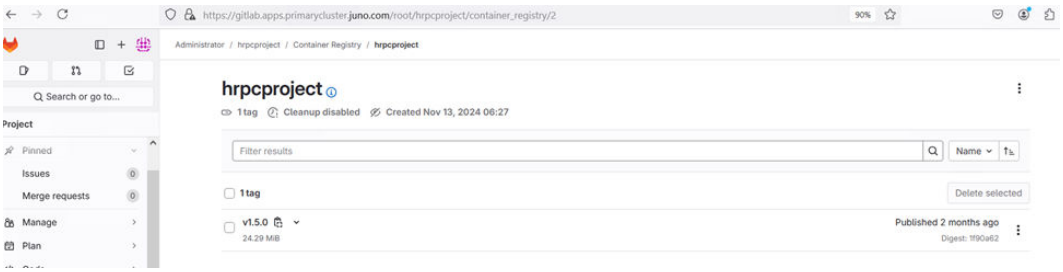
   c. Tag the image with the GitLab container registry path.

```
[root@occlient ~]# podman tag localhost/hitachi/hspc-replication-operator:v1.5.0 registry.apps.primarycluster.juno.com/root/hrpcproject:v1.5.0
[root@occlient ~]# podman images
REPOSITORY                                          TAG         IMAGE ID      CREATED       SIZE
localhost/hitachi/hspc-replication-operator         v1.5.0      c3374b377ffe  2 months ago  57.1 MB
registry.apps.primarycluster.juno.com/root/hrpcproject  v1.5.0      c3374b377ffe  2 months ago  57.1 MB
[root@occlient ~]#
```

   d. Push the image to the GitLab container registry.

```
[root@occlient ~]# podman push registry.apps.primarycluster.juno.com/root/hrpcproject:v1.5.0 --tls-verify=false
Getting image source signatures
Copying blob d8d4c1f29dbb skipped: already exists
Copying blob c4629cdd872f skipped: already exists
Copying blob c302f124a09b skipped: already exists
Copying blob c23224575d87 skipped: already exists
Copying blob 927ef23185e7 skipped: already exists
Copying blob b4aa04aa577f skipped: already exists
Copying blob c9391c2df23a skipped: already exists
Copying blob 7ce5193283fc skipped: already exists
Copying blob 7bc0e0606081 skipped: already exists
Copying blob dee140111c9c skipped: already exists
Copying config c3374b377f done
Writing manifest to image destination
Storing signatures
[root@occlient ~]#
```

   e. In the HRPC project, navigate to **Deploy**, and then **Container Registry** to verify that the HRPC image is available as v1.5.0.

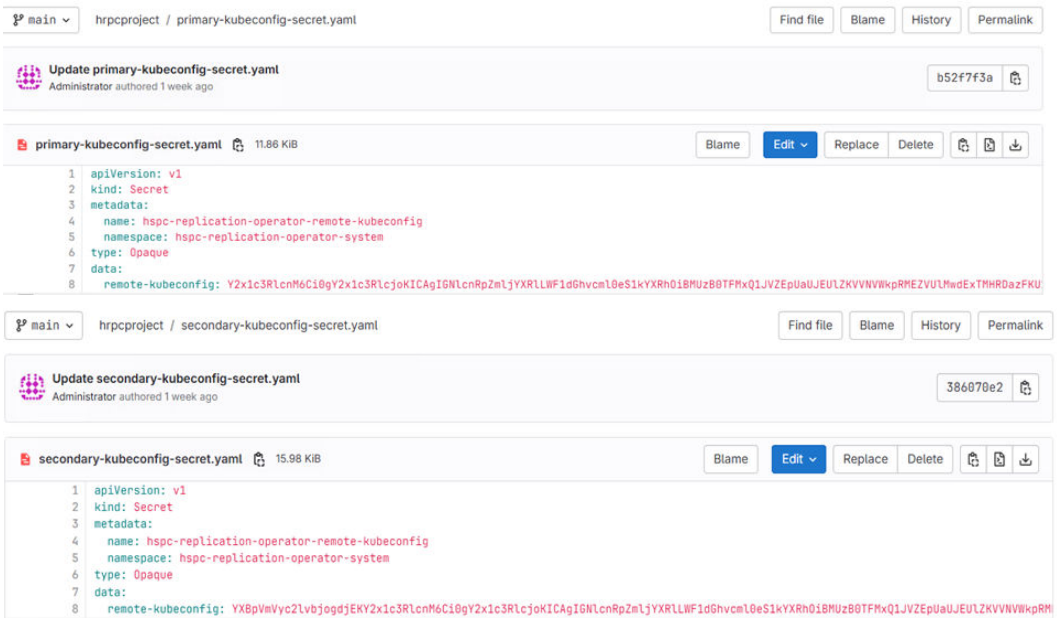


5. Prepare YAML manifest files from the sample files present in `hrpc_1_5_0/yaml`.
   a. Convert the kubeconfig primary and kubeconfig secondary in a base64-encoded format.

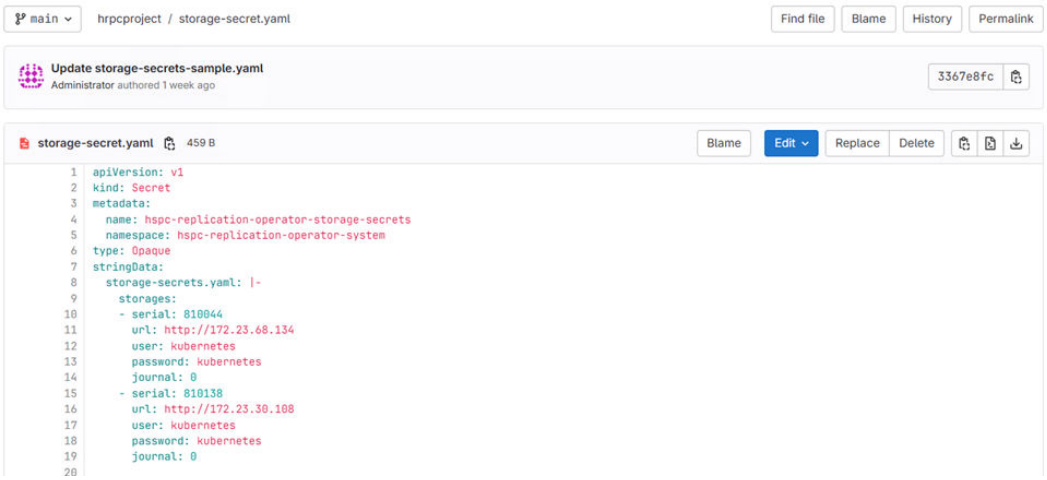
```
[root@occlient ~]# cat kubeconfig-primary|base64 -w 0
Y2x1c3RlcnM6Ci0gY2x1c3RlcjoKICAgIGNlcnRpZmljYXRlLWF1dGhvcml0eS1kYXRhOiBMUzB0TFMxQ1JVZEpUaUJEUlZKVVNVWkpRMEZVUlMwdExTMH
aMGxKUldGaVZGVldRazVoZWlkM1JGRlpTa3R2V2tsb2RtTk9RVkZGVEVKU1FYZFFhalZUVFVKQlIwRXhWVVVLUTNoTlNtSXpRbXhpYms1dllWZGFNRTFUW
ZkR2QyRllUbXhqYmxwc1kya3hjMkl5VG1oaVIyaDJZek5SZEFwaklteHVZbTFXZVUxQ05GaEVWRWt3VFZSQmVFNXFSWGROZWxFdlQxWnZXRVJVVFRCTlZF
UNRVWRCTVZWRkNrTjRUVXBpTTBKc1ltNU9iMkZYV2pCTlUyZDNTbWRaUkZaUlVVUkZlRGx5WkZkS2JFeFhSbmRoV0U1c1kyNWFiR05wTVhOaU1rNW9Za2R
```

```
[root@occlient ~]# cat kubeconfig-secondary|base64 -w 0
YXBpVmVyc2lvbjogdjEKY2x1c3RlcnM6Ci0gY2x1c3RlcjoKICAgIGNlcnRpZmljYXRlLWF1dGhvcml0eS1kYXRhOiBMUzB0TFMxQ1JVZEpUaUJEUlZKVV
FTkRRV3hIWjBGM1NVSkJaMGxKV1dZMEsxSTVUVTFOVERoM1JGRlpTa3R2V2tsb2RtTk9RVkZGVEVKU1FYZEthalZyVFVOS1IwRXhWVVVLUVhkM1ltRlhOV
RCZUU1NlNUV1BWRkY2VDBSbmVFMUNORmhFVkVrd1RWUkJlVTVxUlhoT1ZHZDNUVlp2V0FwRVZFa31UVlJCZVU1cVJYaE9WR2QzVFd4dmQwcEVSV2xOUTBG
FVubFpNbmd4WXpOU2JHTnBOWEZrVnpWMkNreHRUblpoVkVORFFWTkpkMFJSV1VwTGIxcEphSFpaVGtGUlJVSkNVVUZFWjJkRlVFRkVRME5CVVc5RFovZEZ
```

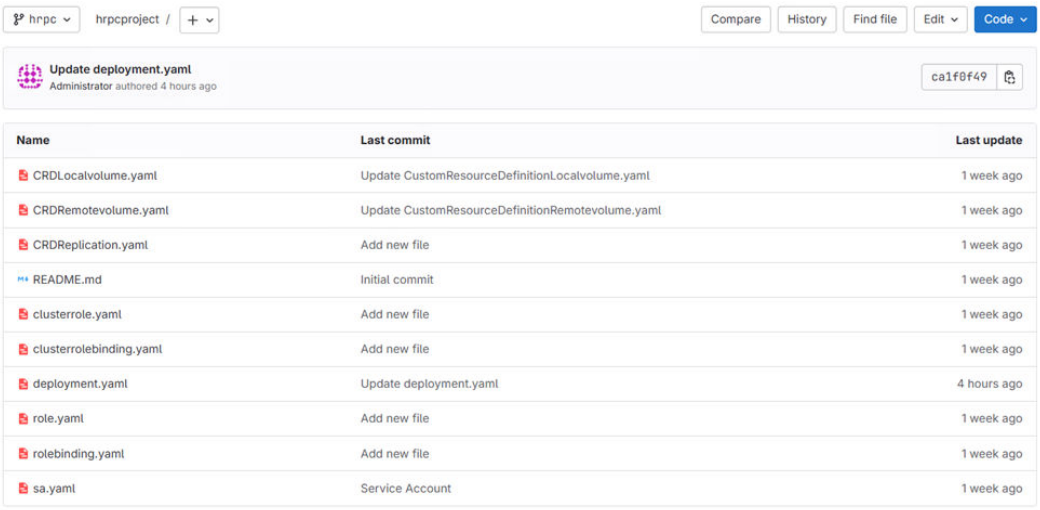
   b. Create two copies of the `remote-kubeconfig-sample.yaml` file, one for the primary and another for the secondary cluster and upload them in the `hrpcproject` main branch. The **remote-kubeconfig** field in the `primary-kubeconfig-secret.yaml` file contains the base64-encoded format of the primary kubeconfig, as derived previously. The same applies for `secondary-kubeconfig-secret.yaml`.



   c. Update the `storage-secrets-sample.yaml` file with both the storage information and upload to the `hrpcproject` main branch.



d. Create a branch `hrpc` in the `hrpcproject` project in GitLab. Create nine individual YAML files from the `hspc-replication-operator.yaml` file as shown and upload them to the `hrpc` branch. Except for `Deployment.yaml`, all files were uploaded without any modification.



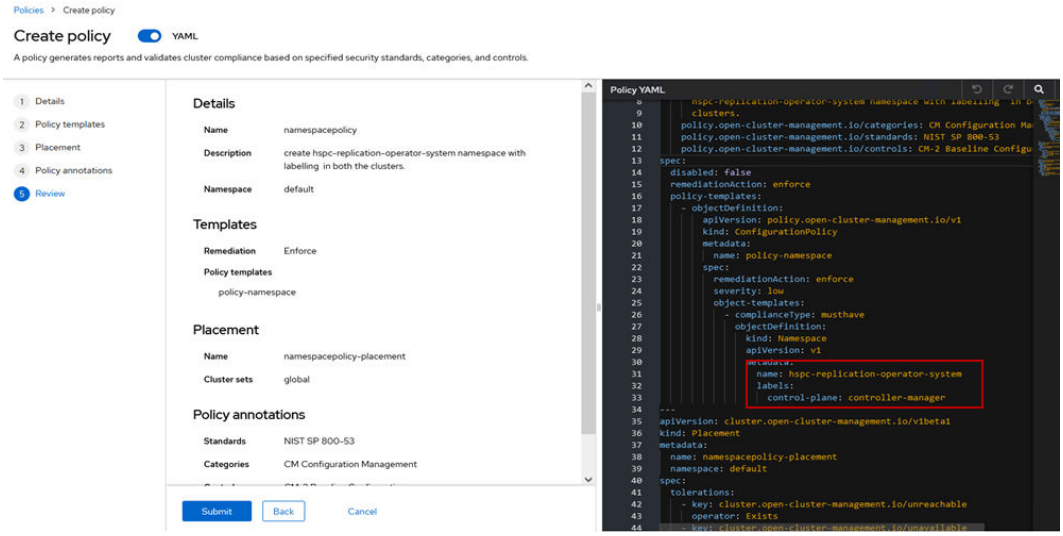
e. Update the `Deployment.yaml` file with the image location and an image pull secret created in a later step.

```
spec:
  imagePullSecrets:
  - name: registry-secret
  containers:
  - args:
    - --enable-leader-election
    command:
    - /manager
    env:
    - name: SPC_REPLICATION_STORAGE_CONFIG
      value: /spc/replication/config/storage-secrets.yaml
    - name: SPC_REPLICATION_REMOTE_KUBECONFIG
      value: /spc/replication/config/remote-kubeconfig
    image: registry.apps.primarycluster.juno.com/root/hrpcproject:v1.5.0
```

Create a namespace for HRPC installation

Procedure

1. Navigate to **Governance** in the Red Hat Advanced Cluster Management and click **Create Policy** to create the namespace `hspc-replication-operator-system` on both clusters. The policy type should be **Namespace must exist** and make sure to add the label **control-plane: controller-manager** while creating the namespace.



Result

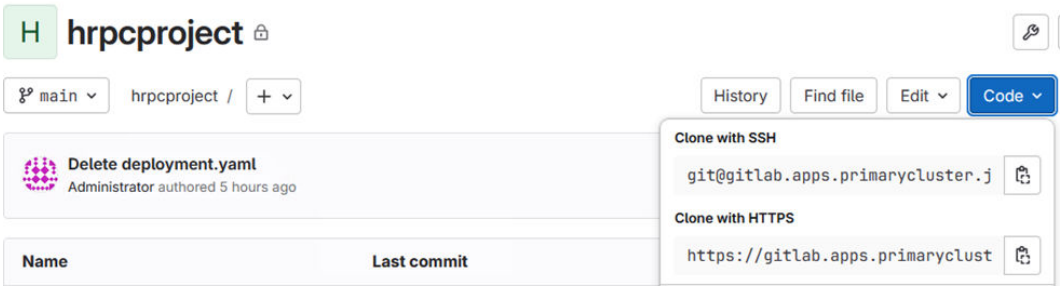
After it is submitted, the namespace `hspc-replication-operator-system` with the label `control-plane: controller-manager` is created on both clusters.

HRPC installation

Install HRPC on both clusters in the namespace “hspc-replication-operator-system” using the subscription-based application from Red Hat Advanced Cluster Management.

Procedure

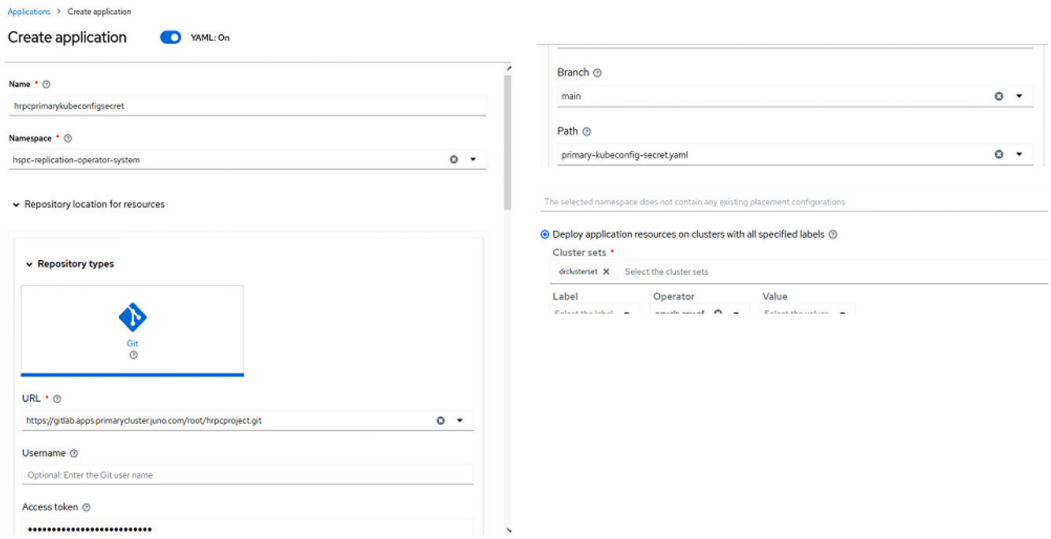
1. Copy the HTTPS URL of “hrpcproject” in GitLab.



2. In the Red Hat Advanced Cluster Management console, navigate to **Applications**, then click **Subscription** in the **Create Application** dropdown list.
3. Create four applications.
   a. Apply “primary-kubeconfig-secret.yaml” in the secondary cluster.

b. Apply “secondary-kubeconfig-secret.yaml” in the primary cluster.
c. Apply “storage-secret.yaml” in both clusters.
d. Apply all the manifest files from the “hrpc” branch of “hrpcproject” in both clusters.

4. Create a primary kubeconfig secret on the secondary cluster. Populate the following fields and click **Create**.
   a. Select the namespace as `hspc-replication-operator-system`.
   b. In the repository types, select **Git** and provide the “hrpcproject” URL.
   c. For the Access token, enter the root password of the primary GitLab.
   d. For Path, enter the file name `primary-kubeconfig-secret.yaml`.
   e. In the cluster section, select **drclusterset**.



5. Because this is the first time the GitLab project “hrpcproject” is being used, a Channel is created. When other applications use the same GitLab project, this Channel will be reused.

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  annotations:
    apps.open-cluster-management.io/reconcile-rate: medium
  name: ggitlabappsprimaryclusterjunocom-root-hrpcprojec
  namespace: ggitlabappsprimaryclusterjunocom-root-hrpcprojec-ns
spec:
  type: Git
  pathname: 'https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git'
```

6. Create a namespace on both clusters as “ggitlabappsprimaryclusterjunocom-root-hrpcprojec-ns” for this Channel.

```
# oc create namespace ggitlabappsprimaryclusterjunocom-root-hrpcprojec-ns
```

7. Create a secret using access token `hrpcprojecttoken` to access the private project in GitLab, as described in step 2 of Prepare manifests for HRPC installation (on page 25).

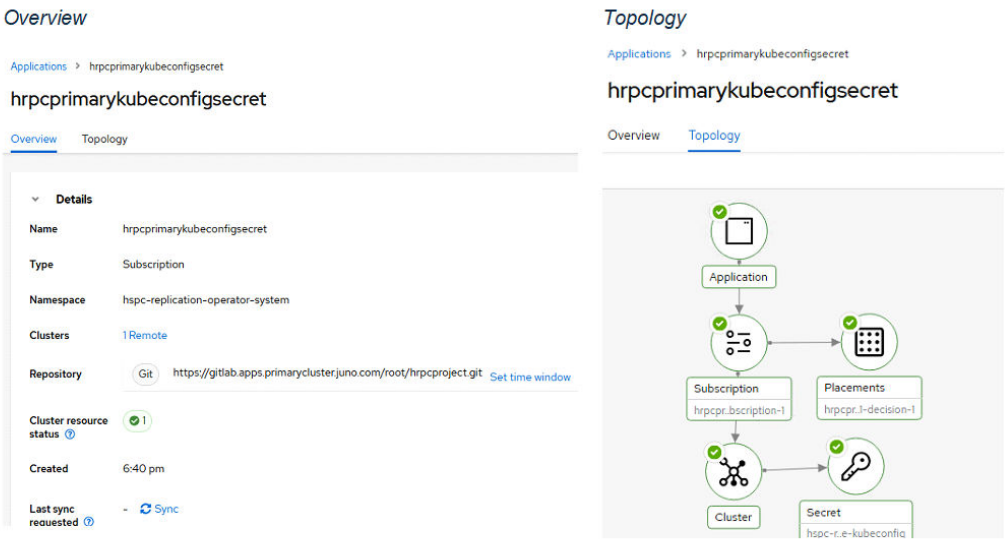
```
#cat secret_hrpcproject_channel.yaml
apiVersion: v1
data:
  accessToken: Z2xwYXQtOGRBLTNKc3ZGWWpKc2Z5elc3OUw=
  user: aHJwY3Byb2plY3R0b2tlbg==
kind: Secret
metadata:
  name: hrpcprojecttokensecret
  namespace: ggitlabappsprimaryclusterjunocom-root-hrpcprojec-ns
type: Opaque
# oc create -f secret_hrpcproject_channel.yaml
```

8. Update the Channel in the Subscription with the secret “hrpcprojecttokensecret” created previously.

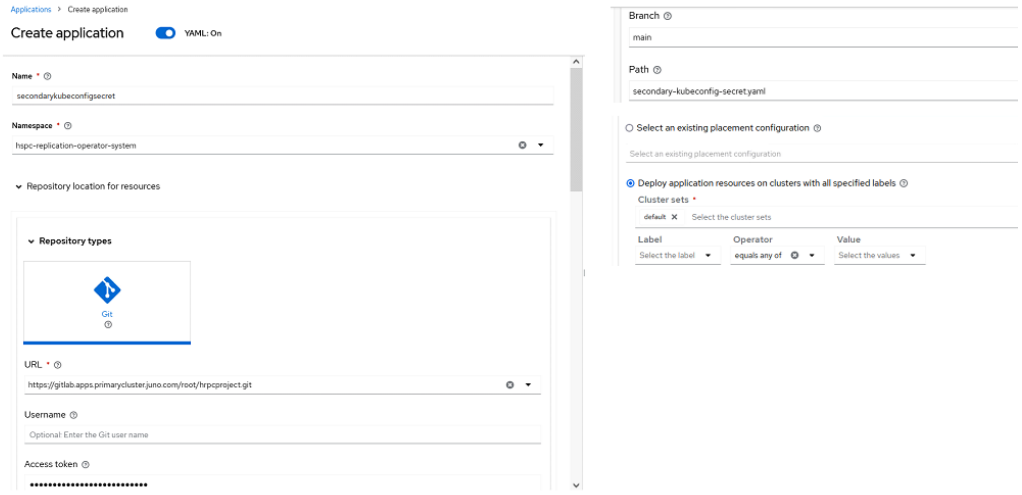
```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  annotations:
    apps.open-cluster-management.io/reconcile-rate: medium
  name: ggitlabappsprimaryclusterjunocom-root-hrpcprojec
  namespace: ggitlabappsprimaryclusterjunocom-root-hrpcprojec-ns
spec:
  insecureSkipVerify: true
  secretRef:
    name: hrpcprojecttokensecret
  type: Git
  pathname: 'https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git'
```

9. In the create application window, click **Create**.

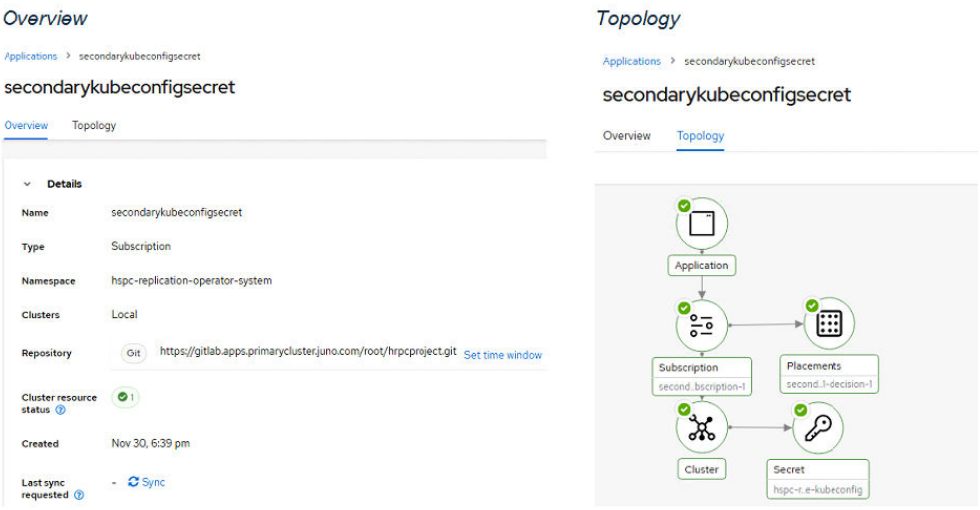
   The application is created successfully.



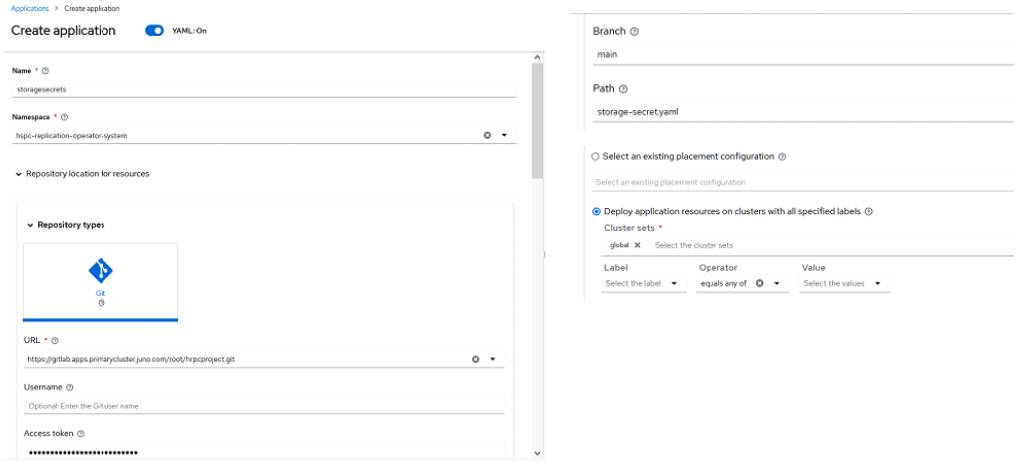
10. Similar to primary-kubeconfig, apply the “secondary-kubeconfig-secret.yaml” file on the primary cluster.



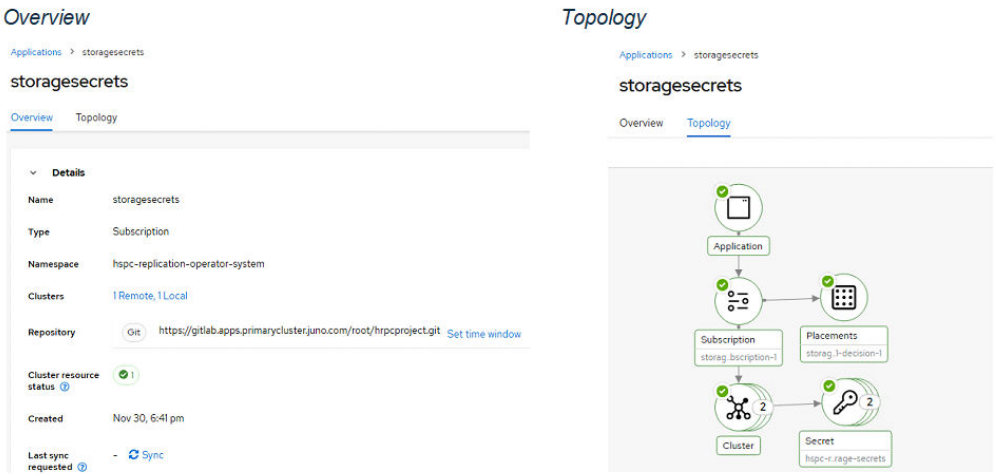
The application is created successfully.



11. Create another application to apply “storage-secret.yaml” in both clusters. For applying to both clusters, select “global” for the clusterset field.

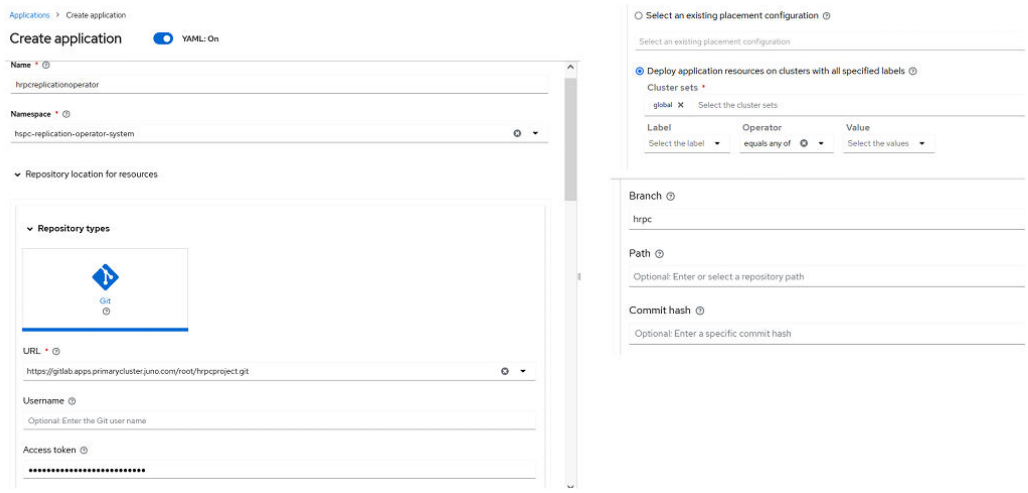


The application is created successfully.

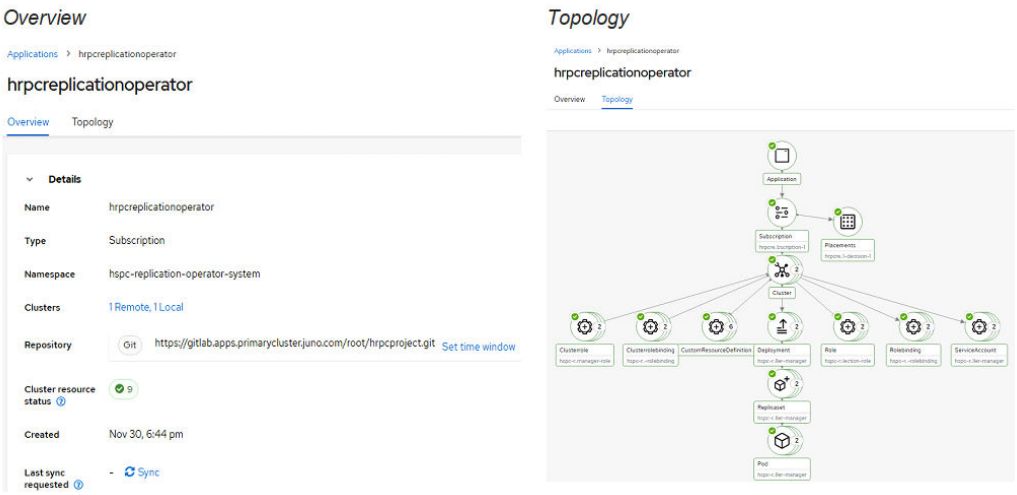


12. Create another application to deploy HSPC replication operator from the branch “hrpc” of “hrpcproject” in both clusters. Before creating the application, create a secret on both clusters to access the private registry in GitLab. This is required for the “deployment.yaml” manifest.

```
#oc create secret docker-registry registry-secret --docker-
server=registry.apps.primarycluster.juno.com --docker-username=root --docker-
password=<gitlab root password> --docker-email=gitlab_admin_7112cd@example.com  -
n hspc-replication-operator-system
```



The HSPC replication operator is deployed successfully on both clusters.



13. Verify from the CLI that the HSPC replication pod is running.

```
[root@occlient ~]# oc get pod -n hspc-replication-operator-system --kubeconfig=kubeconfig-primary
NAME                                                  READY   STATUS    RESTARTS        AGE
hspc-replication-operator-controller-manager-cbf8bd996-v6jfm   1/1     Running   16 (173m ago)   16h
[root@occlient ~]#
[root@occlient ~]# oc get pod -n hspc-replication-operator-system --kubeconfig=kubeconfig-secondary
NAME                                                  READY   STATUS    RESTARTS   AGE
hspc-replication-operator-controller-manager-cbf8bd996-lbvhf   1/1     Running   0          16h
[root@occlient ~]#
```

Test 3: Data mobility of a stateful application

This test case describes the process of migrating a stateful MongoDB application from the primary Red Hat OpenShift cluster, using a persistent volume from VSP One Block 24 storage system to the secondary Red Hat OpenShift cluster, accessing the VSP One Block 28 storage system. Both clusters are deployed on a bare-metal environment with HA820 G3 servers. The MongoDB application deployed using the Red Hat Advanced Cluster Management Argo CD ApplicationSet - Push model application and replication of a persistent volume is performed using Red Hat Advanced Cluster Management Subscription-based application.

The validation points are:

1. Deploy a stateful MongoDB application on the primary cluster and insert data.
2. Replicate the volume (PVC) using Universal Replicator, orchestrated by HRPC.
3. Rebuild the MongoDB application with the replicated volume (PVC) on the secondary cluster.
4. Validate that entries made on the primary site are available on the secondary site, and verify that new entries can also be added.

Deploy a stateful MongoDB application on the primary cluster

The example application consists of MongoDB and Mongo Express.

mongodbapp.yaml

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb
spec:
  serviceName: mongodb-service
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
        selector: mongodb
    spec:
      serviceAccountName: mongodb-sa
      terminationGracePeriodSeconds: 30
      containers:
      - name: mongodb
        image: mongo
        ports:
        - name: mongo-port
```

mongoexpress.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-express
  labels:
    app: mongo-express
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-express
  template:
    metadata:
      labels:
        app: mongo-express
    spec:
      serviceAccountName: mongodb-sa
      containers:
      - name: mongo-express
        image: mongo-express
        ports:
        -  containerPort: 8081
        env:
```

```
          containerPort: 27017
        env:
        - name:
MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-
username
        - name:
MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-
password
        volumeMounts:
        - name: mongo-data
          mountPath: /data/db
      imagePullSecrets:
      - name: regcred
  volumeClaimTemplates:
  - metadata:
      name: mongo-data
    spec:
      storageClassName: sc-hrpc
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 40Gi
```

mongoservice.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  ports:
  - port: 27017
    protocol: TCP
    targetPort: 27017
  selector:
    app: mongodb
  clusterIP: None
```

mongo-secret.yaml

```
        - name:
ME_CONFIG_MONGODB_ADMINUSERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name:
ME_CONFIG_MONGODB_ADMINPASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
        - name: ME_CONFIG_MONGODB_SERVER
          valueFrom:
            configMapKeyRef:
              name: mongodb-configmap
              key: database_url
      imagePullSecrets:
      - name: regcred
```

mongoexpressservice.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mongo-express-service
spec:
  selector:
    app: mongo-express
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
```

config.yaml

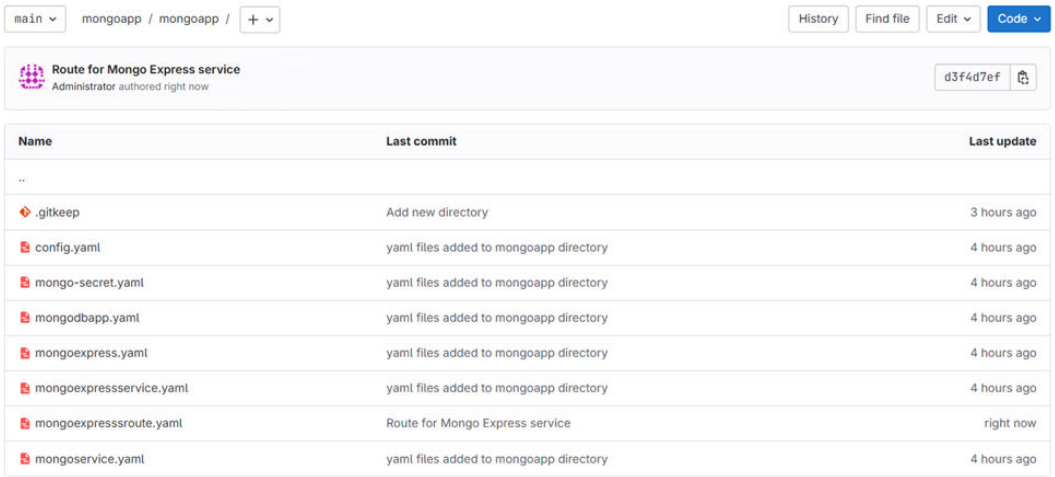
`apiVersion: v1`
`kind: Secret`
`metadata:`
`  name: mongodb-secret`
`type: Opaque`
`data:`
`  mongo-root-username: dXNlcm5hbWU=`
`  mongo-root-password: cGFzc3dvcmQ=`
	`apiVersion: v1`
	`kind: ConfigMap`
	`metadata:`
	`  name: mongodb-configmap`
	`data:`
	`  database_url: mongodb-service`

	mongoexpresssroute.yaml

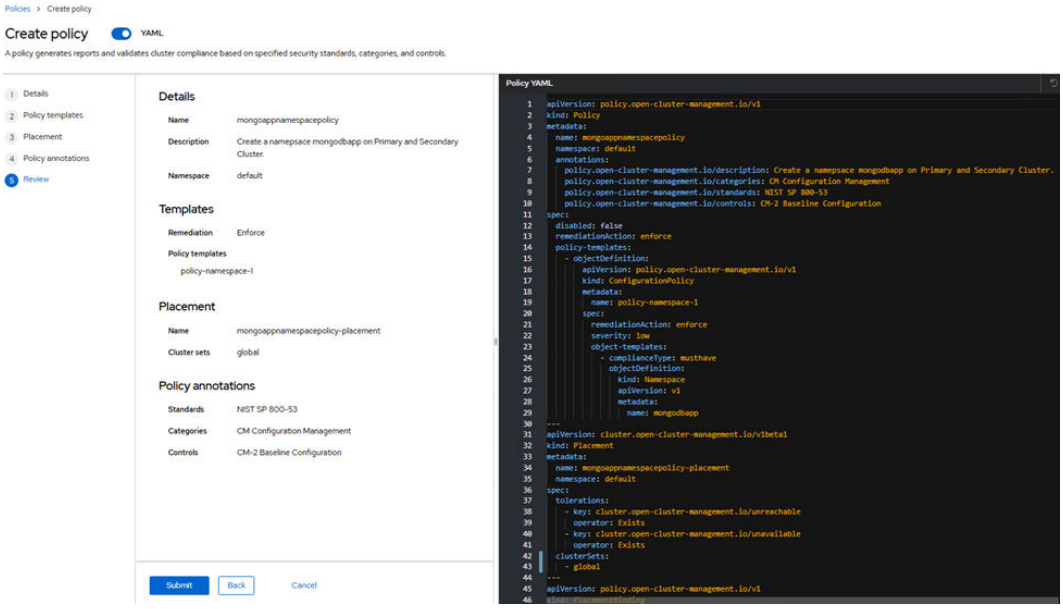
	`kind: Route`
	`apiVersion: route.openshift.io/v1`
	`metadata:`
	`  name: mongoexpressroute`
	`  namespace: mongodbapp`
	`spec:`
	`  host: mongoexpressroute-mongodbapp.apps.primarycluster.juno.com`
	`  to:`
	`  kind: Service`
	`  name: mongo-express-service`
	`  weight: 100`
	`  port:`
	`  targetPort: 8081`
	`  wildcardPolicy: None`

1. Create a project mongoapp in the GitLab of the primary cluster.

   A mongoapp directory is created and all the manifest files for the sample application are placed there.



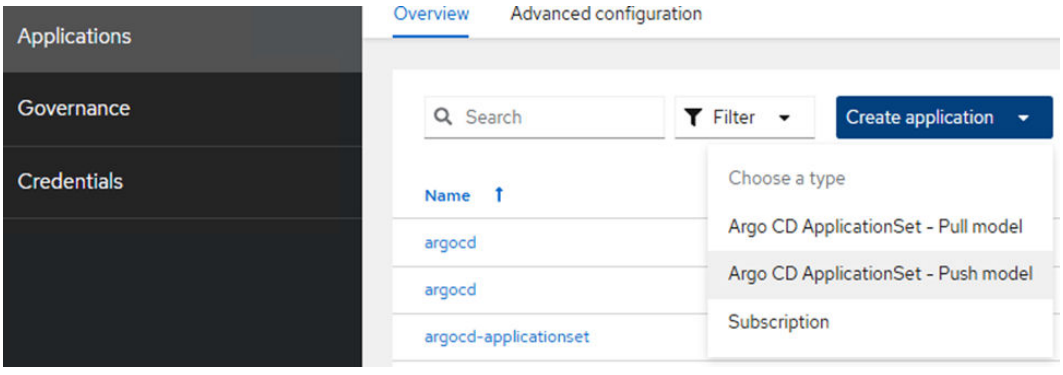
2. Create a Red Hat Advanced Cluster Management policy of type namespace must exist from the Governance section. With this policy, create a namespace mongodbapp on both clusters.



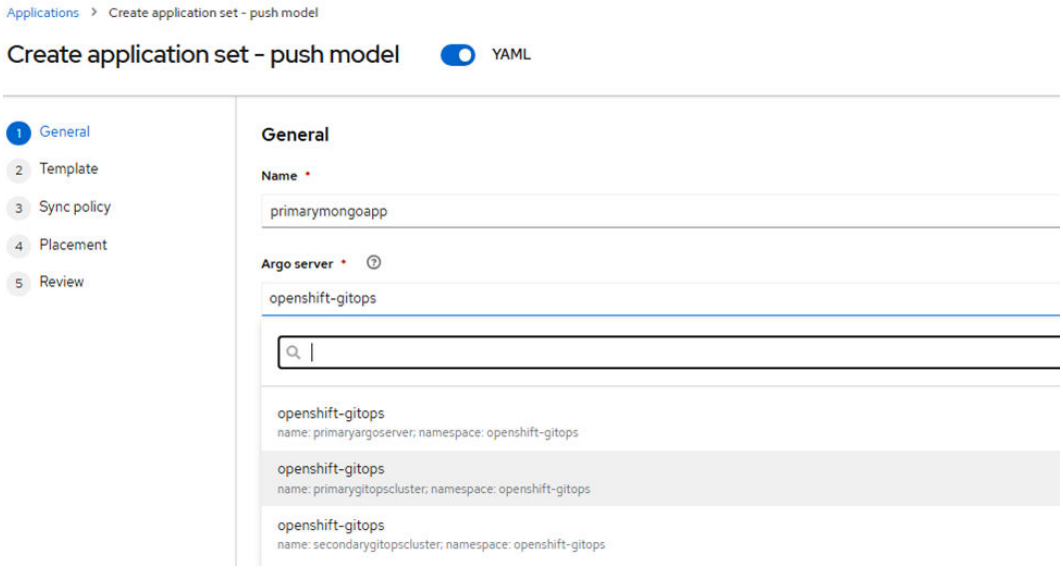
3. Create a service account mongodb-sa in the mongodbapp namespace in both clusters and add any uid security context constraints policy for that service account.

```
# oc create serviceaccount mongodb-sa -n mongodbapp
# oc adm policy add-scc-to-user anyuid -z mongodb-sa -n mongodbapp
```

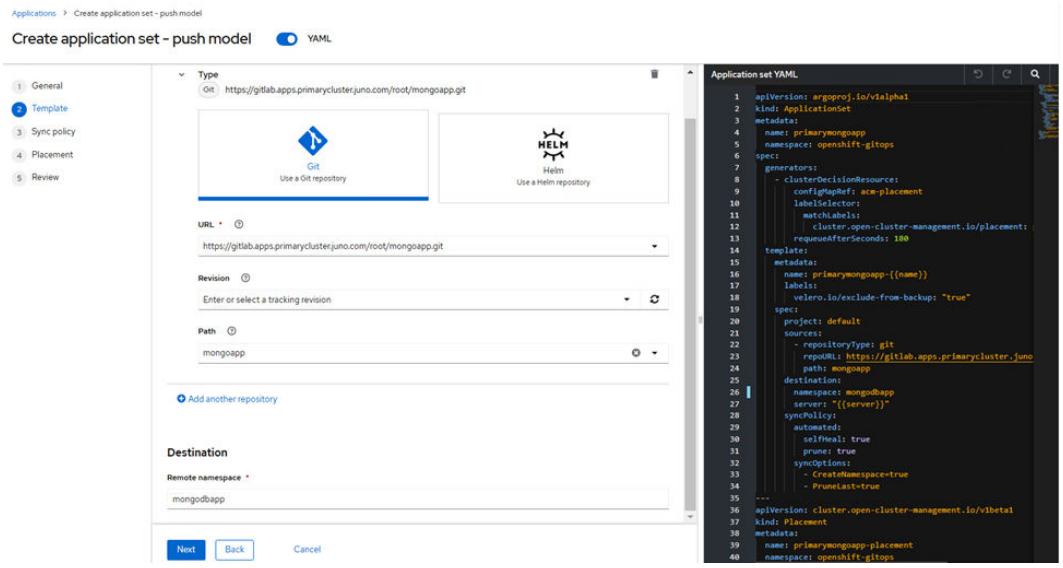
4. Deploy the MongoDB application from Red Hat Advanced Cluster Management. Select Applications, then click Create Application and select Argo CD ApplicationSet – Push model.



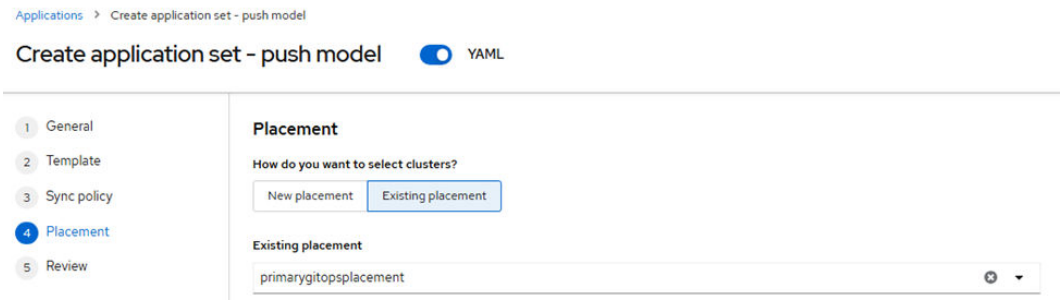
5. Provide a name primarymongoapp, and select Argo server as primarygitopscluster, and select Next.



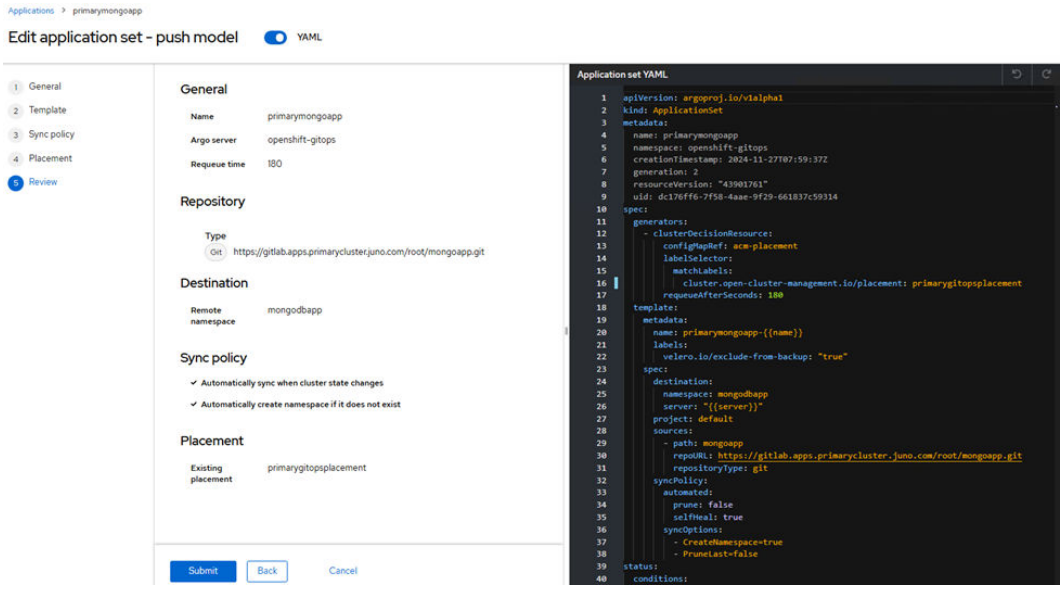
6. In the Template section, select Git, mention the GitLab URL for mongoapp, mention the directory mongoapp as Path, and mention mongodbapp as Remote namespace in the Destination. Click Next.



7. In the Sync Policy section, clear the two options with "Delete resource.." and click Next.
8. In the Placement section, select Existing placement, select primarygitopplacement, and then click Next.

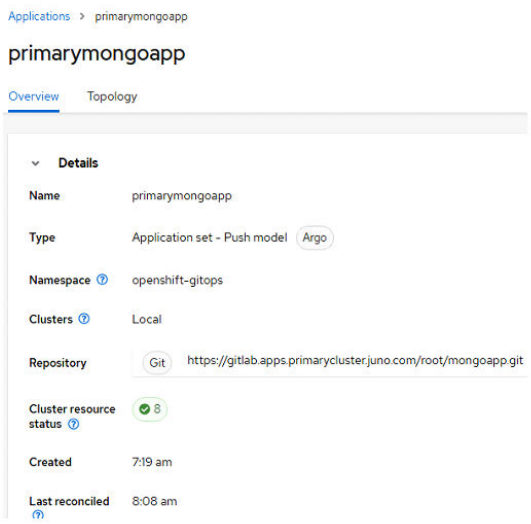


9. Review the settings and click Submit.

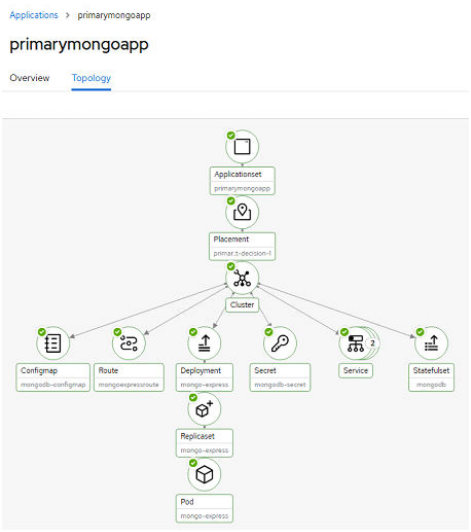


10. Check that the application deployed from both Overview and Topology as shown.

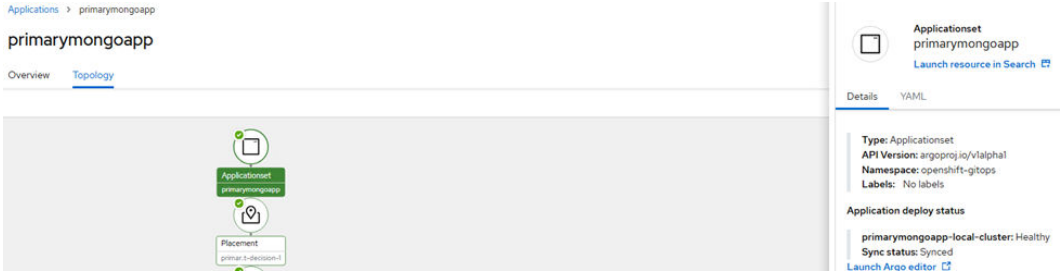
Overview



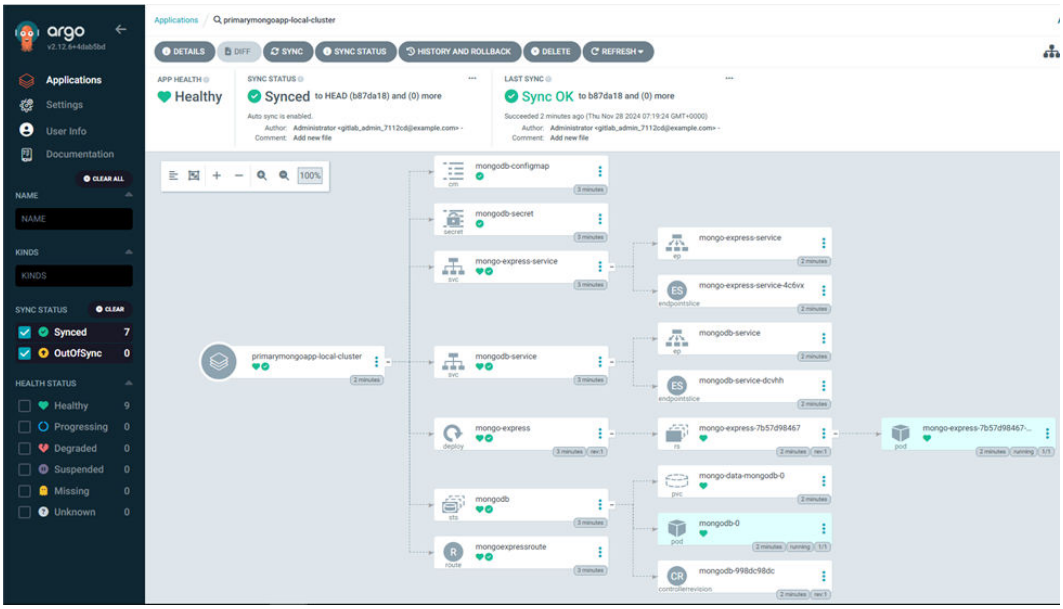
Topology



11. Click the Application set and select Launch Argo editor in the Details section. This will open the application in Argo CD.

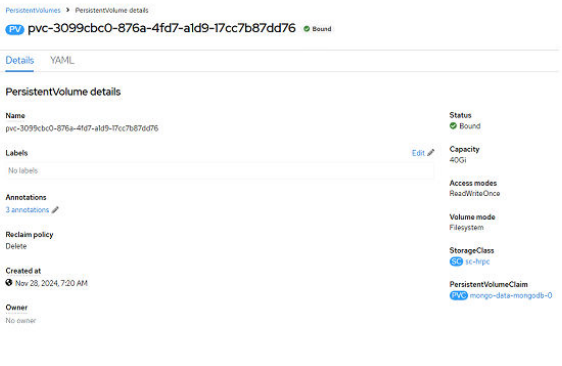


12. Verify that all the resources for the MongoDB application are created, and sync status is OK.

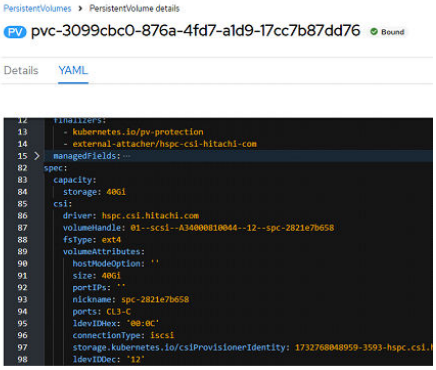


13. Verify that PVC mongo-data-mongdb-0 is created on the primary cluster. Get the information about the PV and PVC from the primary OpenShift console.

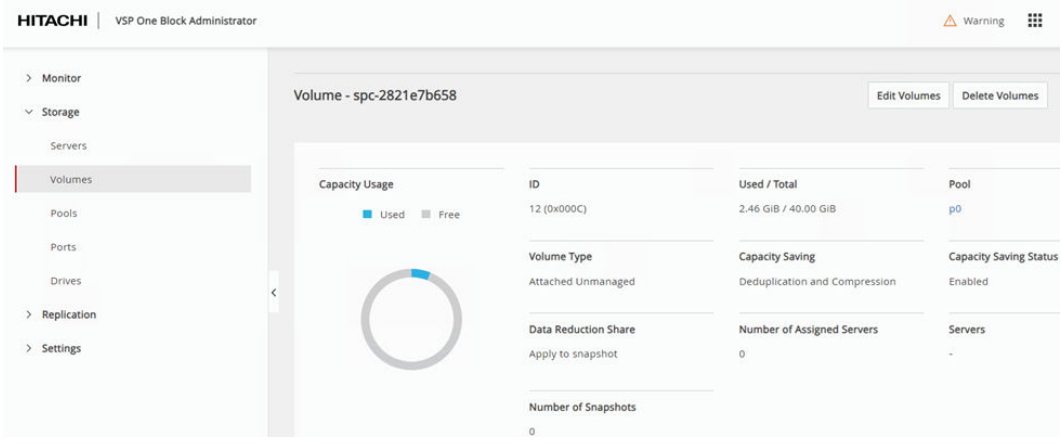
PV definition



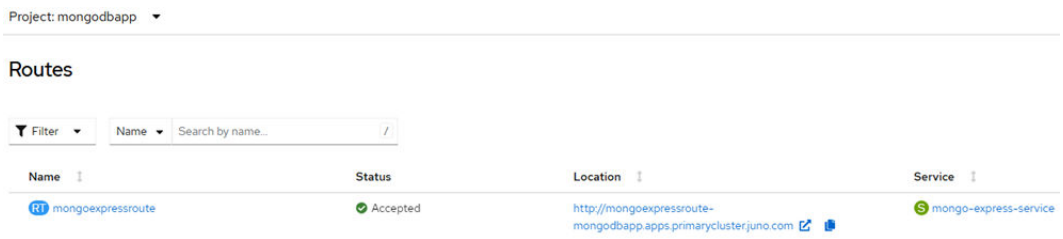
Storage resources for PV



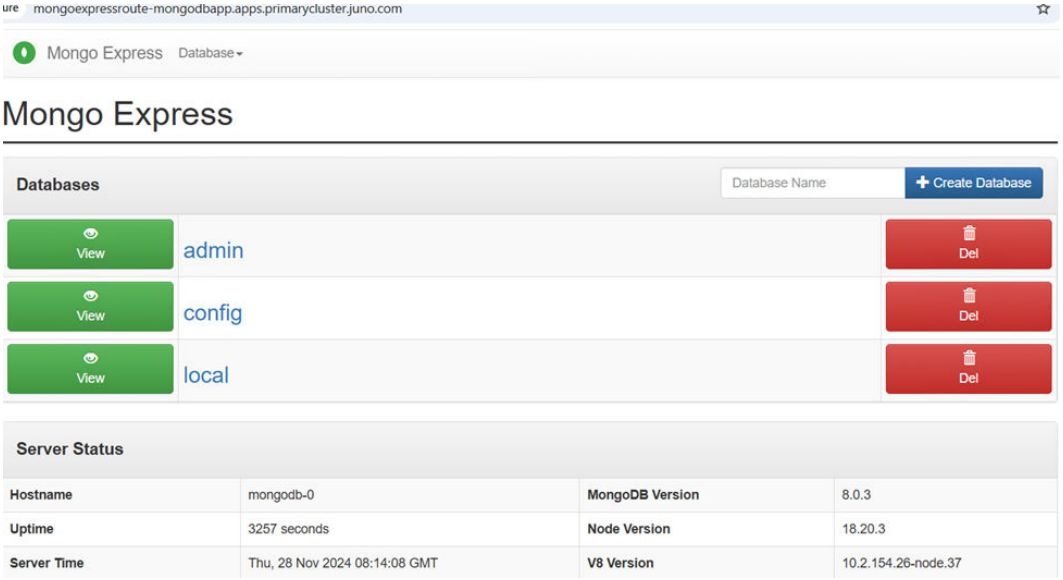
14. For the PV, a DRS volume (00:0C) is created on the VSP One Block 24 storage system.



15. The route for Mongo Express is created. Click the URL under Location.



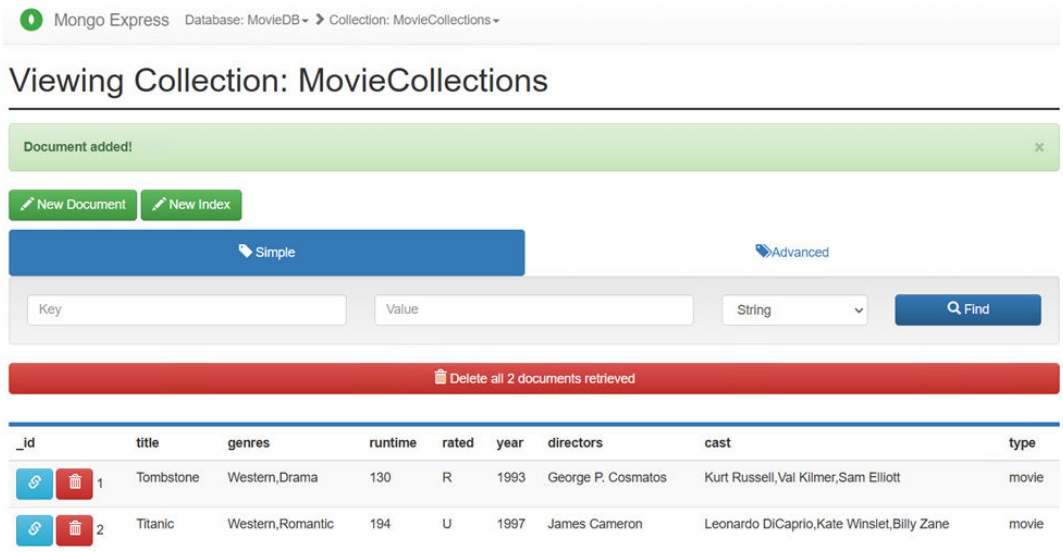
16. Log in to Mongo Express with the default username and password of admin and pass.



Create a sample database called MovieDB.

17. Click View beside MovieDB. In the collection name section, write MovieCollections, click +create collection.
18. Click New Document, and add two documents as shown.

19. After adding the two documents as described, view the collection.

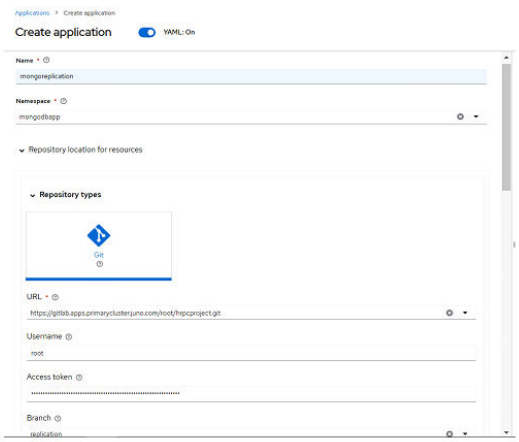


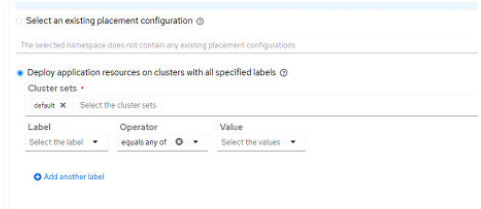
Replicate the volume (PVC) using Universal Replicator, orchestrated by HRPC

To replicate the PVC, create a manifest for the replication CR in GitLab. Then, apply the manifest on the primary cluster. Verify the status of the replication from the OpenShift console as well as from both the storage systems.

1. Create a branch in the hrpcproject project as Replication and keep the manifest for the replication CR with the PVC name. A sample manifest is available in the downloaded HRPC package.

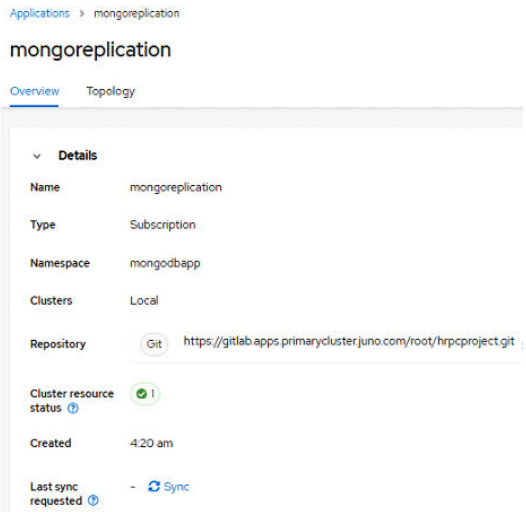
2. Create a subscription-based application to replicate the MongoDB PVC from VSP One Block 24 to VSP One Block 28. Here the application name is mongoreplication, Namespace is mongodbapp, select Git, add the URL of hrpcproject, and select Branch as Replication, and in the Cluster sets section, select default.



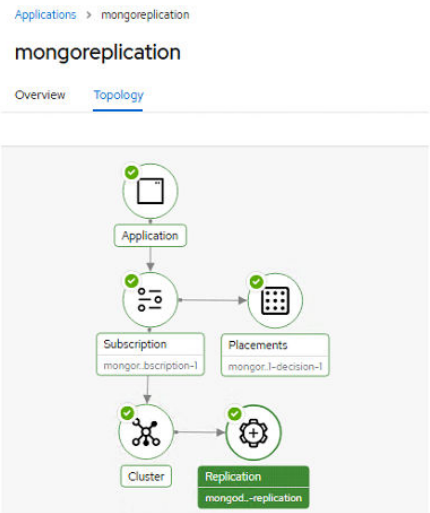


The application is created.

Overview

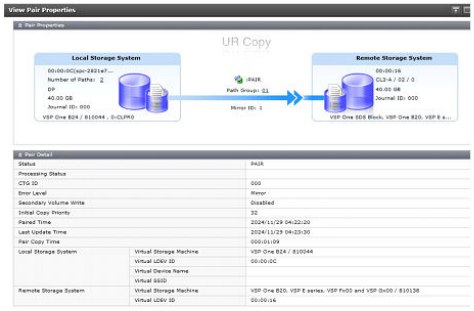


Topology

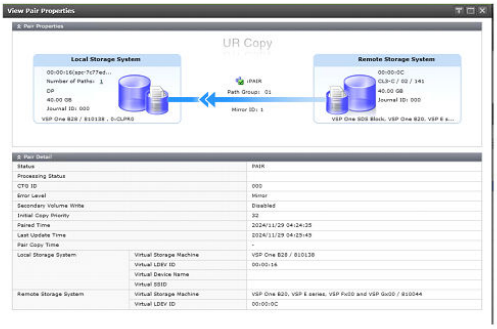


3. Verify replication status from Storage Navigator.

Status from VSP One Block 24



Status from VSP One Block 28



4. Verify the Replication CR details from both clusters.

Status from Primary Cluster

Replications › Replication details

mongodb-replication

Details YAML

```yaml
apiVersion: hspc.hitachi.com/v1
kind: Replication
metadata: ···
spec:
  desiredPairState: none
  persistentVolumeClaimName: mongo-data-mongodb-0
  replicationAttribute: primary
  storageClassName: sc-hrpc
status:
  currentOperation:
    operation: none
    running: false
  replicationPair:
    localSerialNumber: 810044
    localStorageDeviceId: A34000810044
    primaryVolumeId: 12
    primaryVolumeNickname: spc-2821e7b658
    remoteSerialNumber: 810138
    remoteStorageDeviceId: A34000810138
    secondaryVolumeId: 22
    secondaryVolumeNickname: spc-7c77ed416e
  status: Ready
```

Status from Secondary Cluster

Replications › Replication details

mongodb-replication

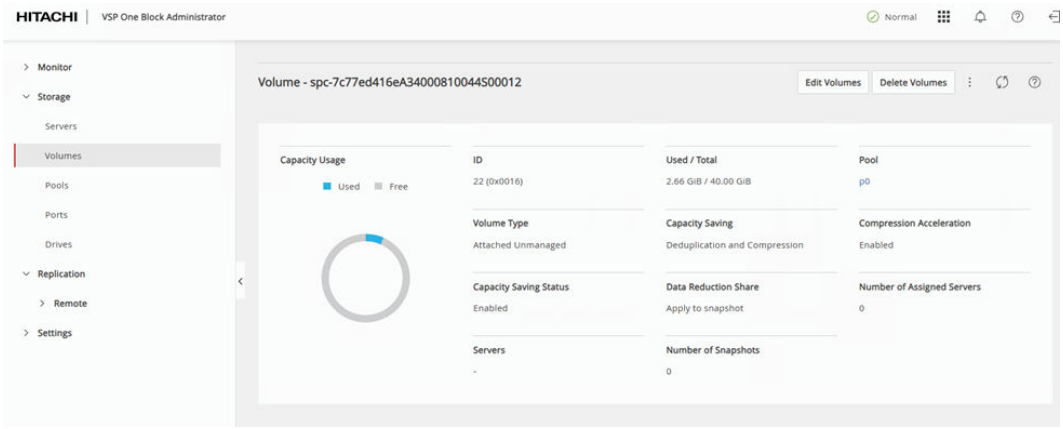
Details YAML

```yaml
  namespace: mongodbapp
  finalizers:
    - hspc.csi.hitachi.com/replication-finalizer
spec:
  desiredPairState: none
  persistentVolumeClaimName: mongo-data-mongodb-0
  persistentVolumeClaimSpec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 40Gi
    storageClassName: sc-hrpc
    volumeMode: Filesystem
    volumeName: pvc-3099cbc0-876a-4fd7-a1d9-17cc7b87dd76
  replicationAttribute: secondary
  storageClassName: sc-hrpc
status:
  currentOperation:
    operation: none
    running: false
  replicationPair:
    localSerialNumber: 810138
    localStorageDeviceId: A34000810138
    primaryVolumeId: 12
    primaryVolumeNickname: spc-2821e7b658
    remoteSerialNumber: 810044
    remoteStorageDeviceId: A34000810044
    secondaryVolumeId: 22
    secondaryVolumeNickname: spc-7c77ed416e
  status: Ready
```

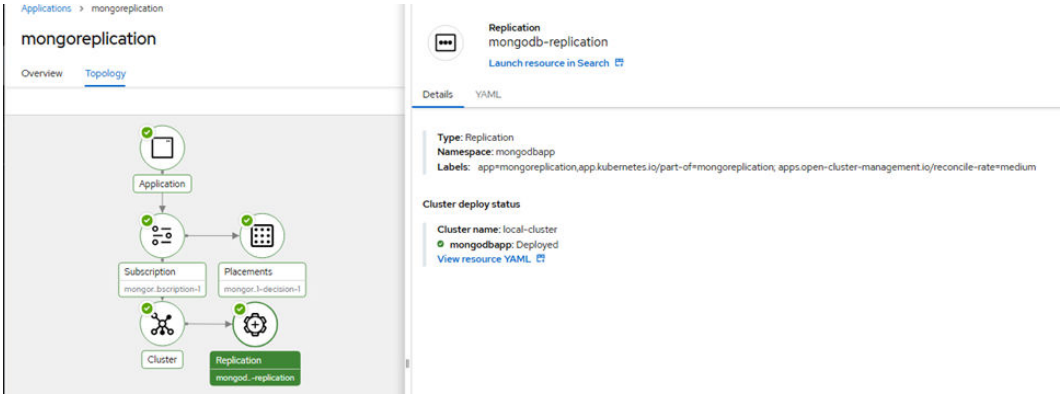
5. Verify the replication status from the OpenShift client machine.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        132m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        133m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication   true    134m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication   true    134m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication   true    134m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication   true    134m
[root@occlient ~]#
```

6. Hitachi Replication Plug-in for Containers (HRPC) in combination with Hitachi Storage Plug-in for Containers (HSPC) creates a DRS persistent volume (LDEV ID 00:16) on the secondary cluster.

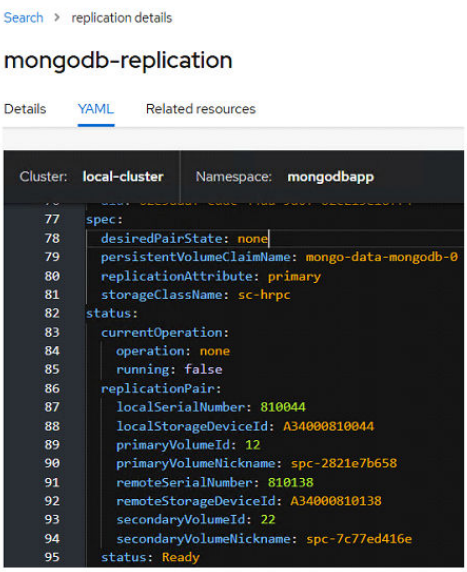


7. To access data from the secondary cluster, UR pairs need to be split. In the mongoreplicationapplication, select the Replication CR, and click View resource YAML.

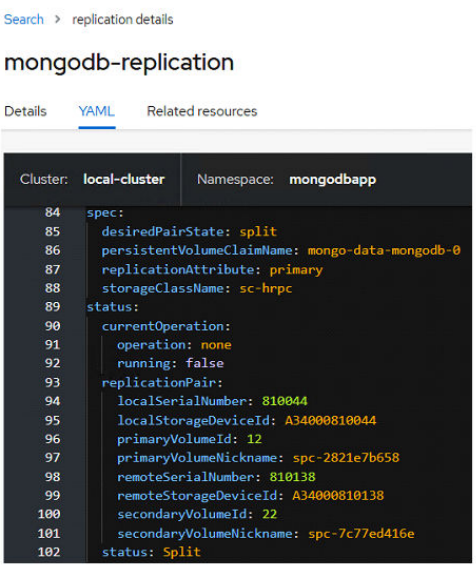


8. Change the desiredPairState, from none to split, click Save, and then Reload. The following screenshots show the status from the primary cluster.

Status in pair state

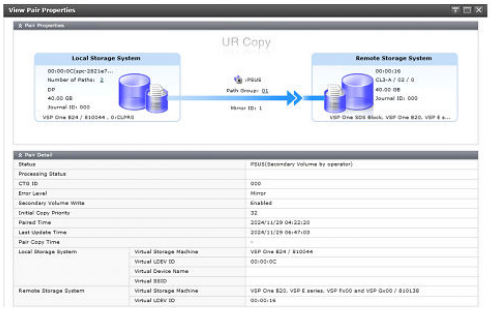


Status after split

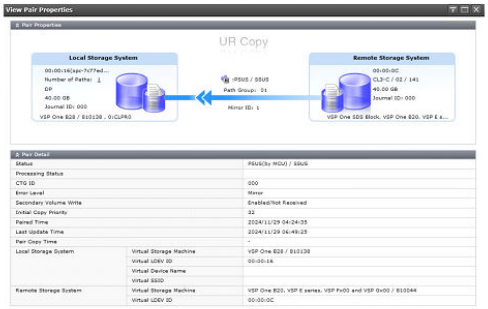


9. Verify the pair split status from Storage Navigator.

Status from VSP One Block 24

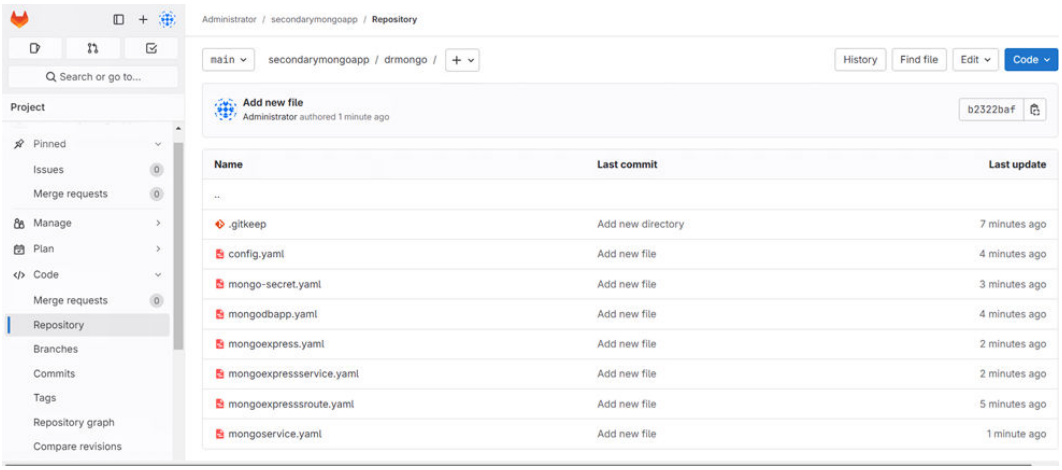


Status from VSP One Block 28



Bring up the MongoDB application on the secondary cluster

1. Copy the application codes from the primary GitLab to the secondary GitLab and make the necessary changes. You can download the codes on the client machine and push them to the secondary GitLab repository. The other option is to use Git mirroring.
2. Create a directory drmongo in the project secondarymongoapp, and keep all the manifest files required for MongoDB and Mongo Express as shown.



3. Update the mongodbapp.yaml file with the replicated PVC name. Instead of volumeClaimTemplates as used in the primary cluster, the replicated PVC name is used in the mongodbapp.yaml file as shown.

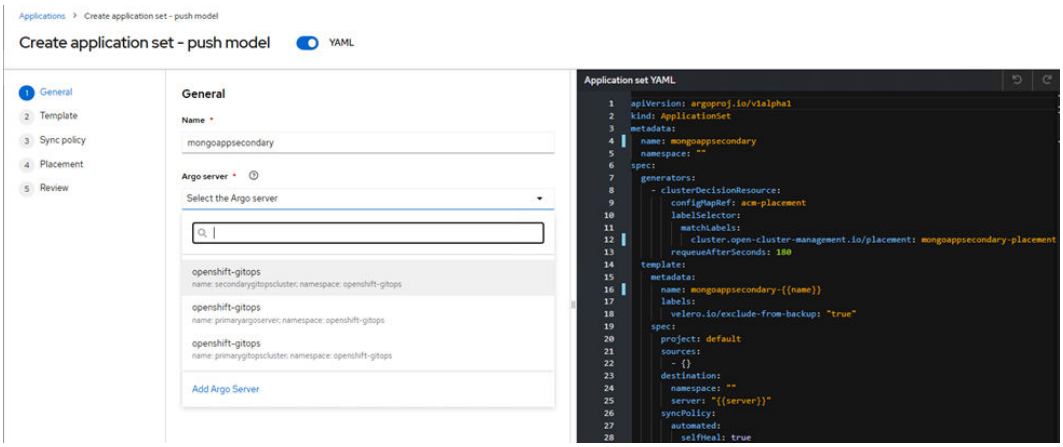
Volume definition in primary cluster

```
volumeClaimTemplates:
- metadata:
    name: mongo-data
  spec:
    storageClassName: sc-hrpc
    accessModes: [ "ReadWriteOnce" ]
    resources:
      requests:
        storage: 40Gi
```

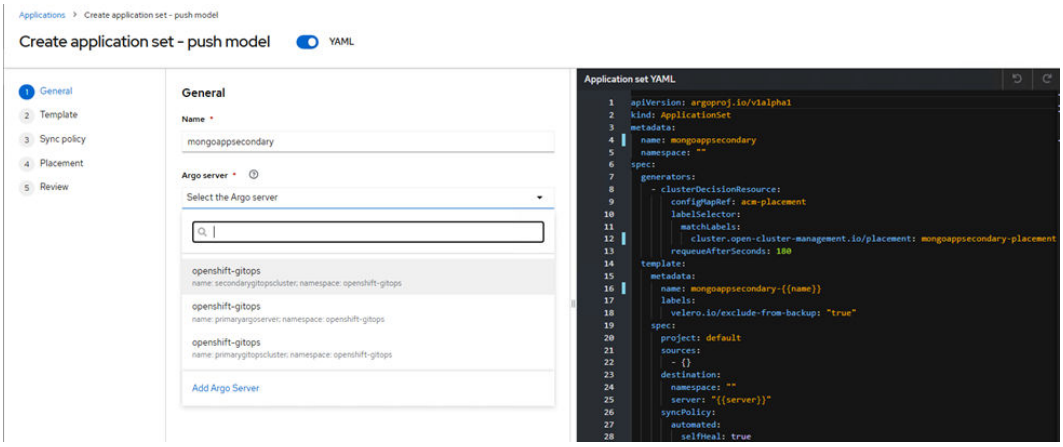
Volume definition in secondary cluster

```
volumes:
- name: mongo-data
  persistentVolumeClaim:
    claimName: mongo-data-mongodb-0
```

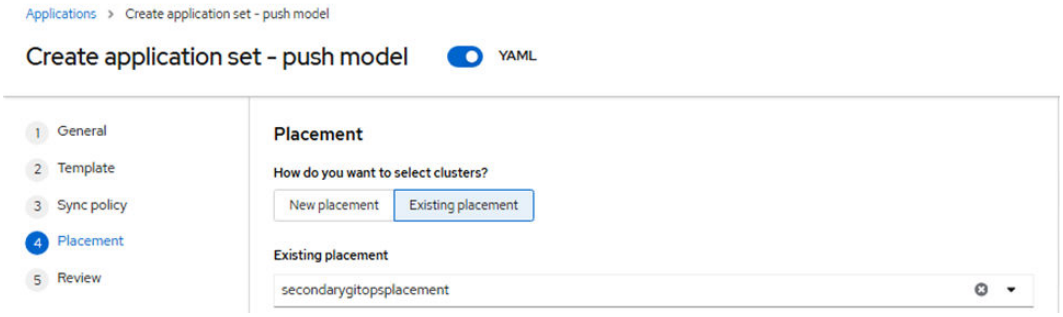
4. In the Red Hat Advanced Cluster Management console, create a new application as mongoappsecondary with Application set – push model and select the secondarygitopscluster as the Argo server. Click Next.



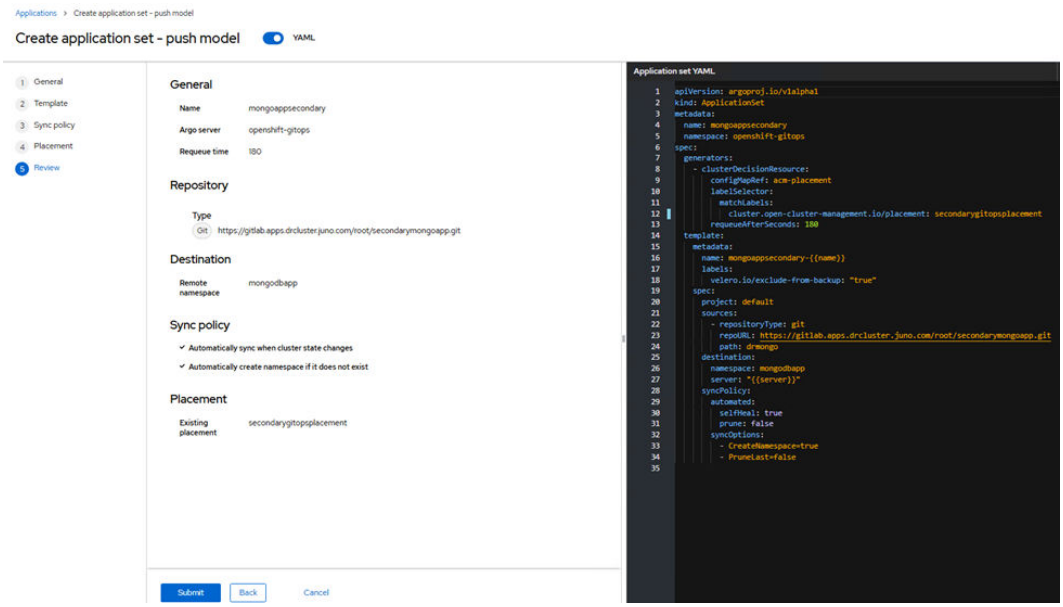
5. Select Git and mention the URL of secondarymongoapp and Path as drmongo, and then click Next.



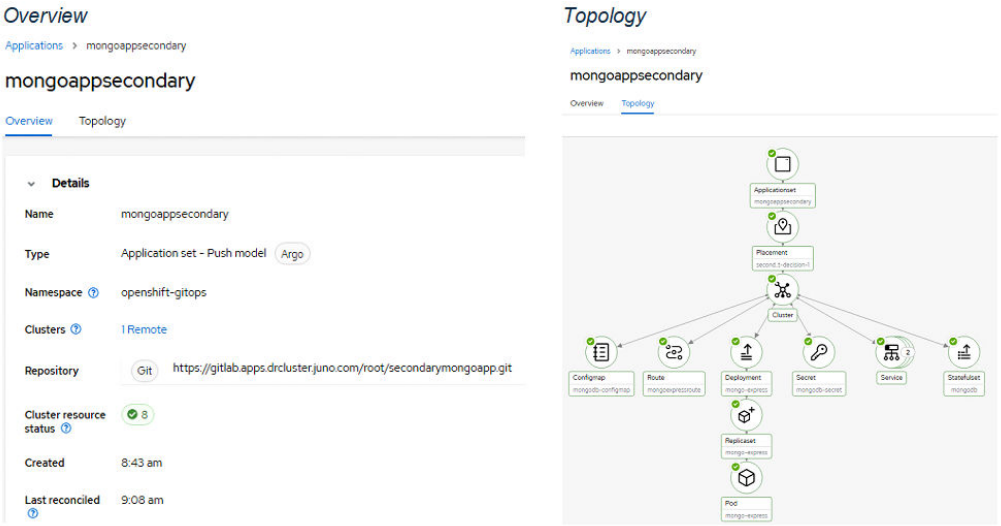
6. In the Sync Policy section, clear the two options with "Delete resource.." and click Next.
7. In the Placement section, select Existing placement and select secondarygitopplacement, and then click Next.



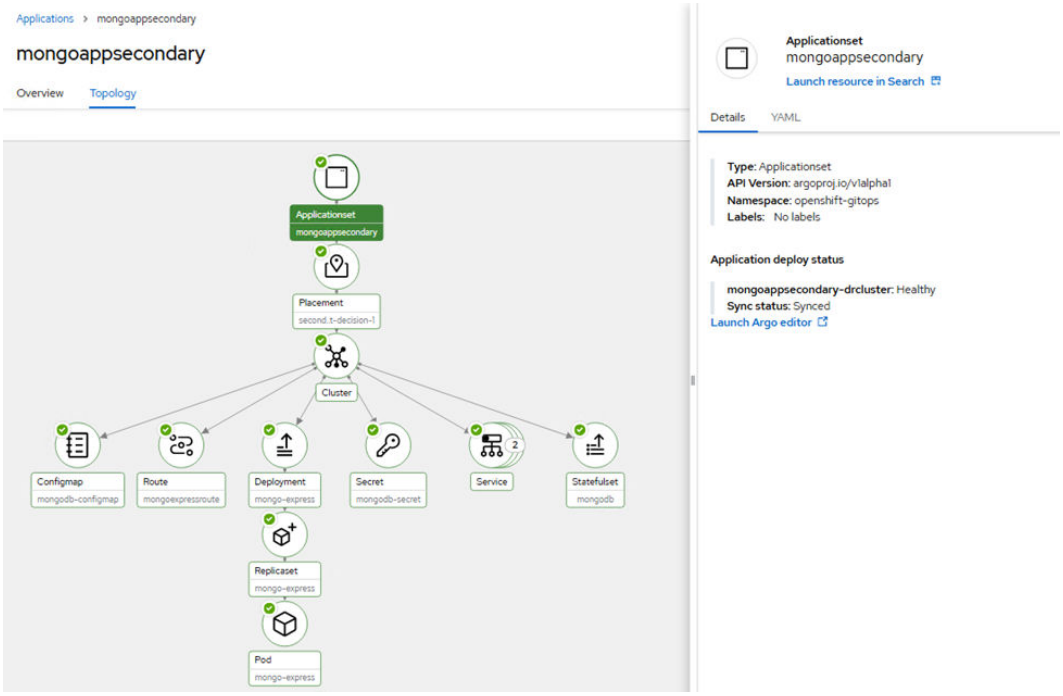
8. Review and click Submit.



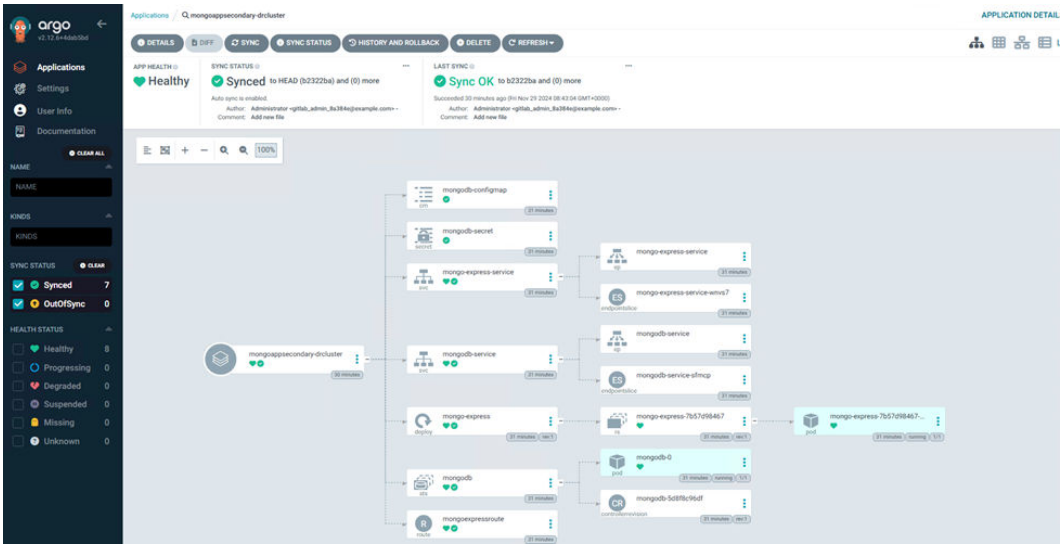
The application is created.



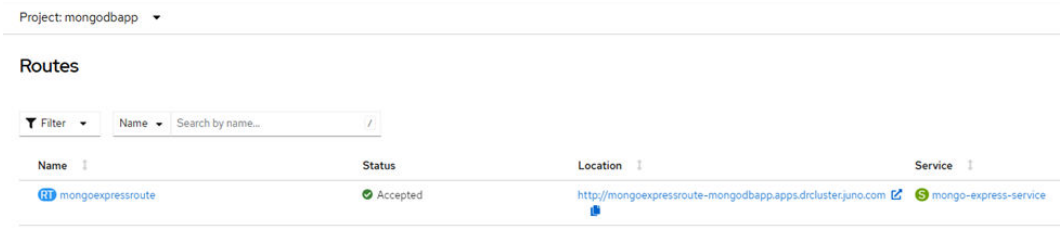
9. Click the Application set and select Launch Argo editor in the details section. This will open the application in Argo CD.



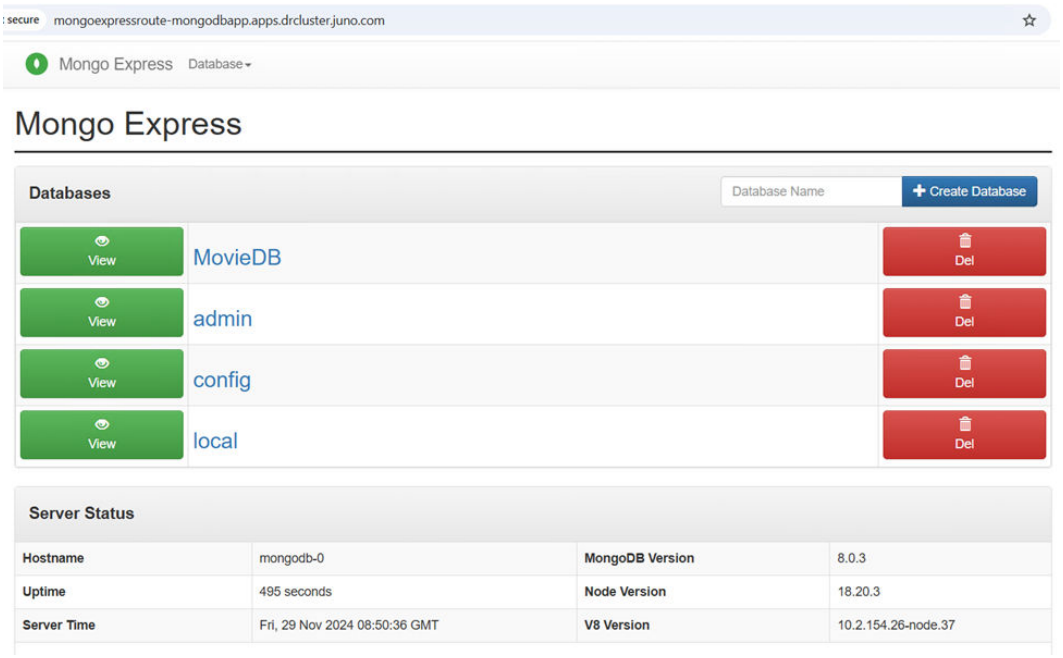
10. Verify that all the resources are created and the sync status is OK.



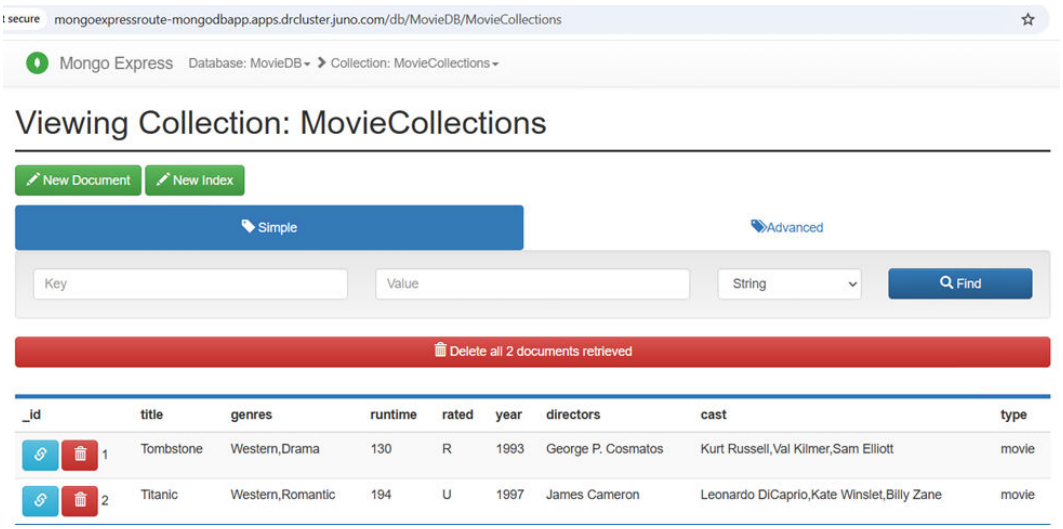
11. The route for Mongo Express is created. Click the URL under Location.



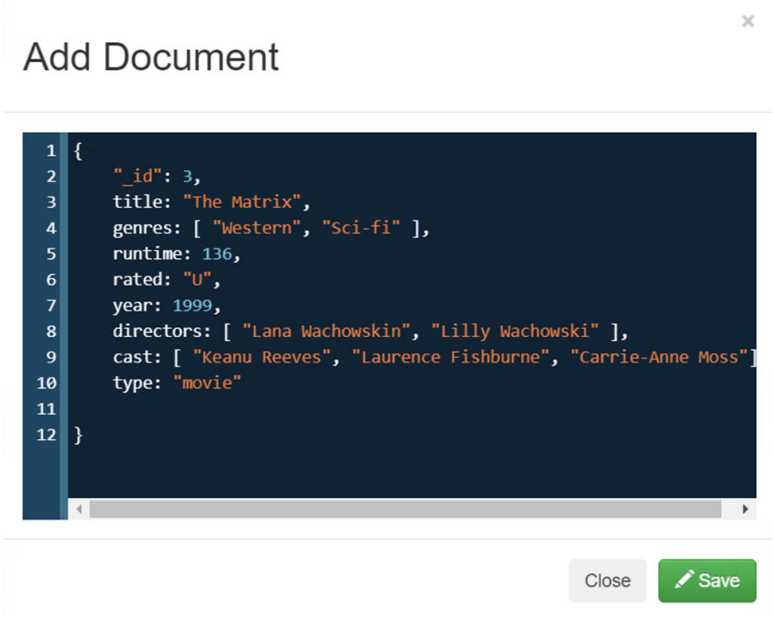
12. Verify that the database MovieDB is available.



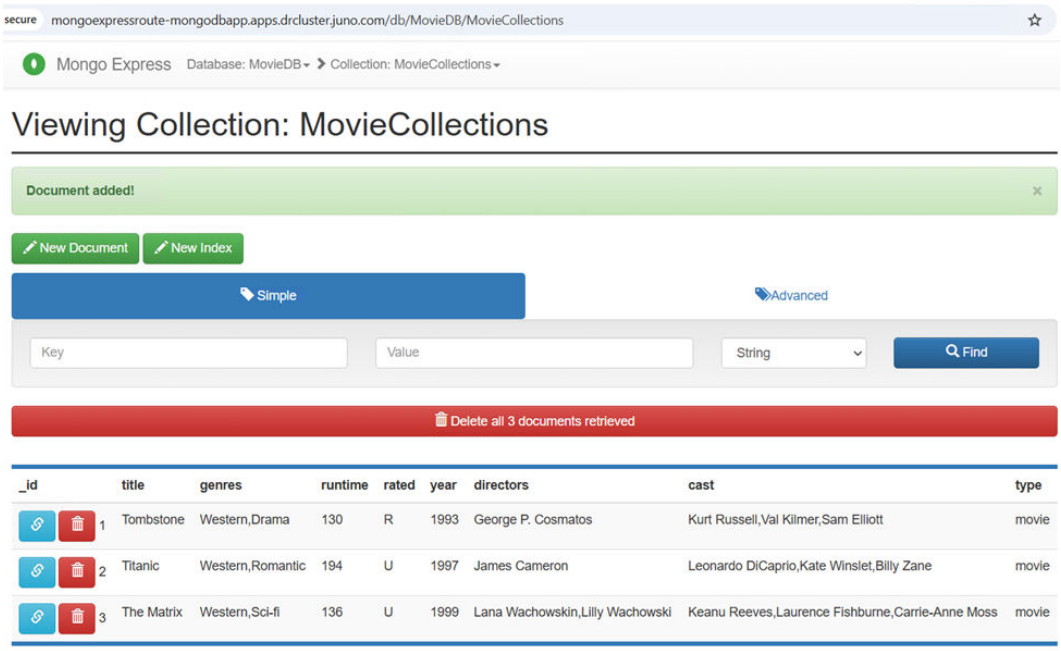
13. The two collections inserted on the primary site are also available.



14. Verify that new data can be inserted. Click New Document, and then add a document to the example, and click Save.



15. Verify that the new document is added.



Test 4: Disaster Recovery solution

This test case demonstrates how the Universal Replicator (UR) operation can be suspended in a planned outage. For example, you can perform maintenance in the primary data center and business can continue operations on the secondary storage system. It also shows a failback operation, such as after the outage. Data created on the secondary storage system is replicated back to the primary storage system and business can resume from the primary site. Red Hat Advanced Cluster Management and Red Hat OpenShift GitOps integration is used to streamline this failover and failback operation.

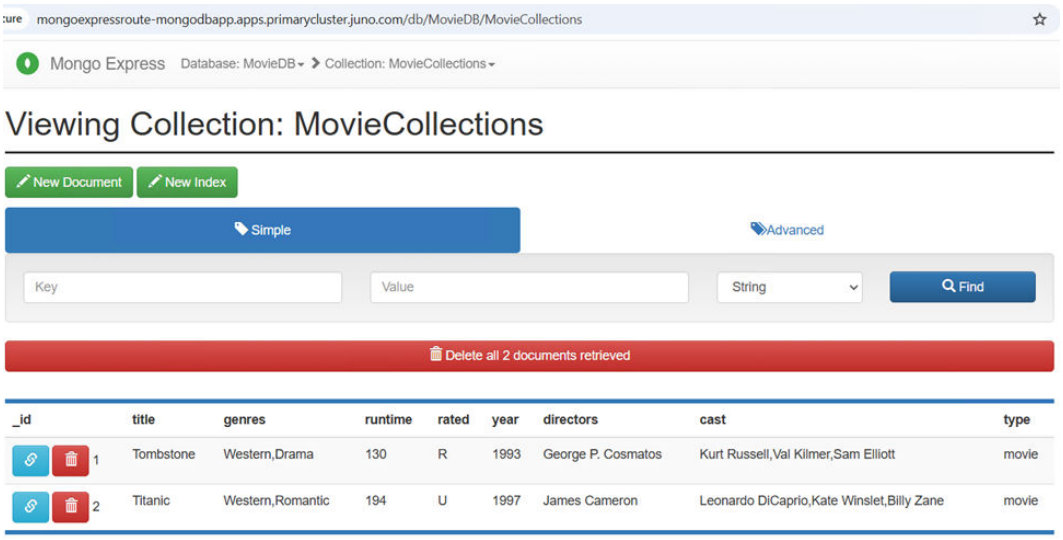
Failover operation

Before you begin

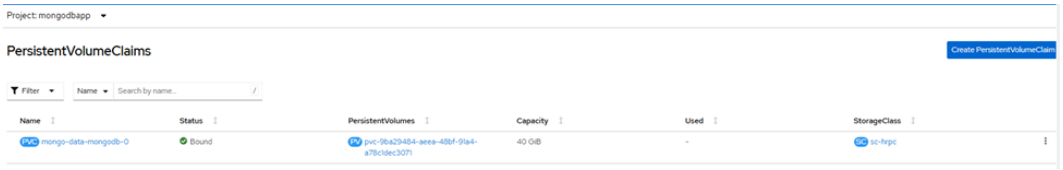
The stateful MongoDB application is deployed on the primary cluster as described previously.

Procedure

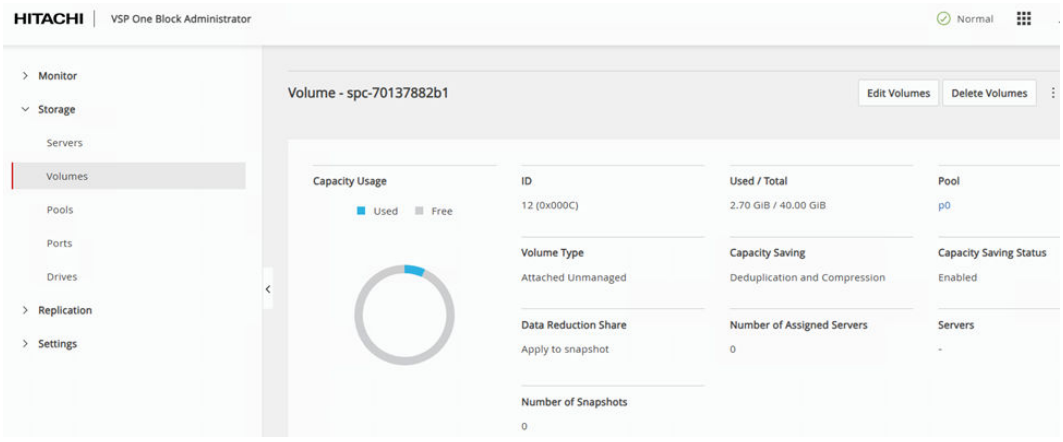
1. Create a database called *MovieDB*, a collection called *MovieCollections*, and then make some entries.



2. For the stateful MongoDB application, a PVC is created on the primary cluster.



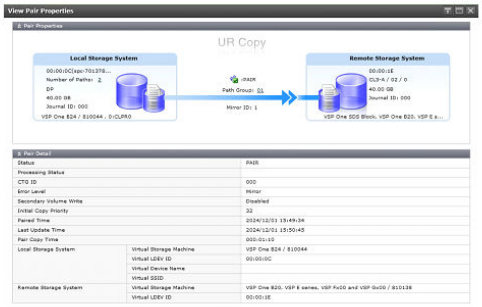
3. A corresponding DRS volume (00:0C) is created on the VSP One Block 24 storage system.



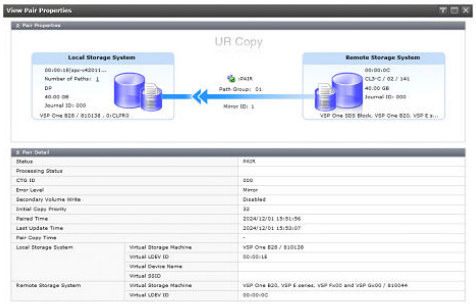
4. Create a Subscription based application to replicate the MongoDB PVC from VSP One Block 24 to VSP One Block 28 storage system as described previously.

5. Verify the replication status from Storage Navigator.

Status from VSP One Block 24

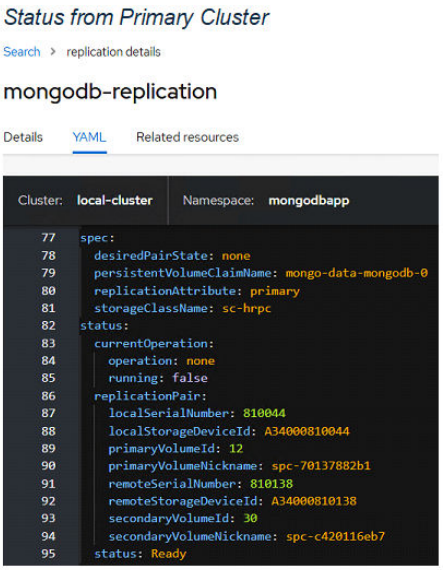


Status from VSP One Block 28

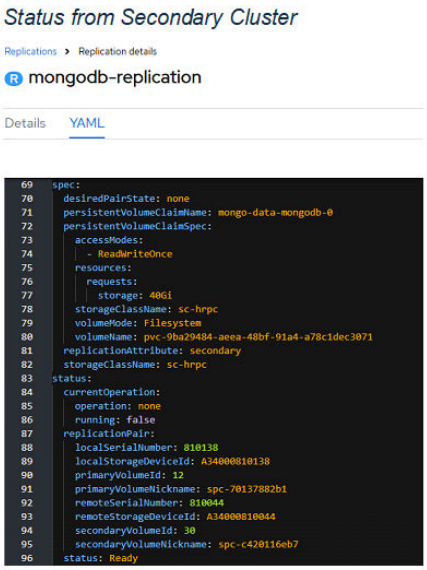


6. Verify the Replication CR details from both clusters.

Status from Primary Cluster



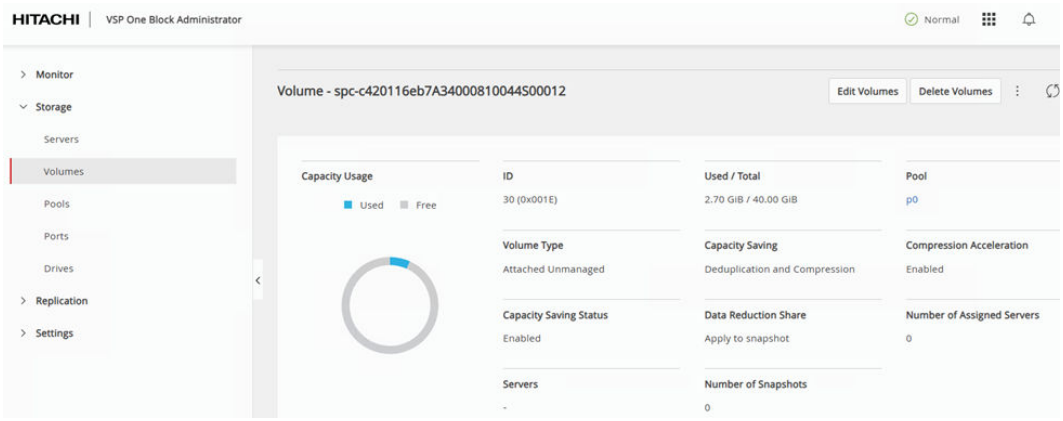
Status from Secondary Cluster



7. Verify the replication status from the OpenShift client machine.

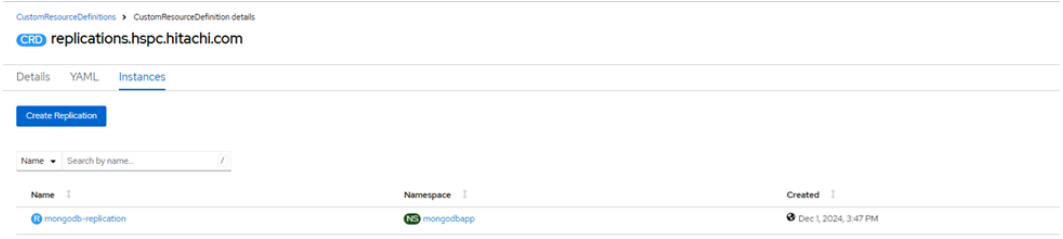
```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        128m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        127m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication        true    128m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication        true    128m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication        true    128m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication        true    128m
[root@occlient ~]#
```

Hitachi Replication Plug-in for Containers in combination with Hitachi Storage plug-in for Containers creates a DRS persistent volume (LDEV ID 00:1E) on the secondary cluster.



8. From the secondary site, change the status of the Replication CR to perform the failover operation. From the secondary cluster OpenShift console, click on **Administration**, then **CustomResourceDefinitions**, search for “replication” and then click **Replication**.

9. Click **Instances**, then **mongodb-replication** and select YAML view.



10. Edit the spec.desiredPairState from “none” to “failover” click **save**, then **reload**. This triggers Hitachi Replication Plug-in for Containers to failover the UR pair. The following screenshots show the status of the Replication CR from the secondary cluster.

Before Failover

Replications › Replication details

mongodb-replication

Details YAML

```
spec:
  desiredPairState: none
  persistentVolumeClaimName: mongo-data-mongodb-0
  persistentVolumeClaimSpec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 40Gi
    storageClassName: sc-hrpc
    volumeMode: Filesystem
    volumeName: pvc-9ba29484-aeea-48bf-91a4-a78c1dec3071
  replicationAttribute: secondary
  storageClassName: sc-hrpc
status:
  currentOperation:
    operation: none
    running: false
  replicationPair:
    localSerialNumber: 810138
    localStorageDeviceId: A34000810138
    primaryVolumeId: 12
    primaryVolumeNickname: spc-70137882b1
    remoteSerialNumber: 810044
    remoteStorageDeviceId: A34000810044
    secondaryVolumeId: 30
    secondaryVolumeNickname: spc-c420116eb7
  status: Ready
```

After Failover

Replications › Replication details

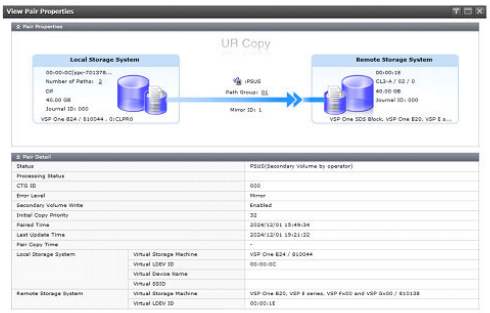
mongodb-replication

Details YAML

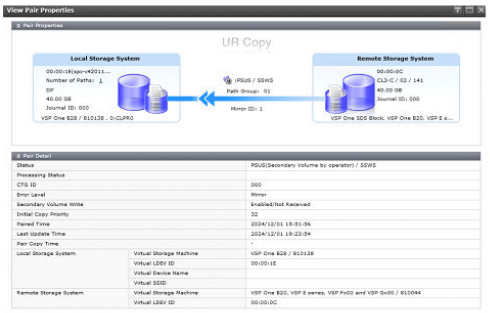
```
  managedFields: ···
  namespace: mongodbapp
  finalizers:
    - hspc.csi.hitachi.com/replication-finalizer
spec:
  desiredPairState: failover
  persistentVolumeClaimName: mongo-data-mongodb-0
  persistentVolumeClaimSpec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 40Gi
    storageClassName: sc-hrpc
    volumeMode: Filesystem
    volumeName: pvc-9ba29484-aeea-48bf-91a4-a78c1dec3071
  replicationAttribute: secondary
  storageClassName: sc-hrpc
status:
  conditions:
    - lastTransitionTime: '2024-12-01T19:20:56Z'
      message: replication pair has failed over
      reason: replicationPairFailover
      status: 'True'
      type: Failover
  currentOperation:
    operation: none
    running: false
  replicationPair:
    localSerialNumber: 810138
    localStorageDeviceId: A34000810138
    primaryVolumeId: 12
    primaryVolumeNickname: spc-70137882b1
    remoteSerialNumber: 810044
    remoteStorageDeviceId: A34000810044
    secondaryVolumeId: 30
    secondaryVolumeNickname: spc-c420116eb7
  status: Failover
```

11. Verify the replication status from Storage Navigator.

Status from VSP One Block 24



Status from VSP One Block 28

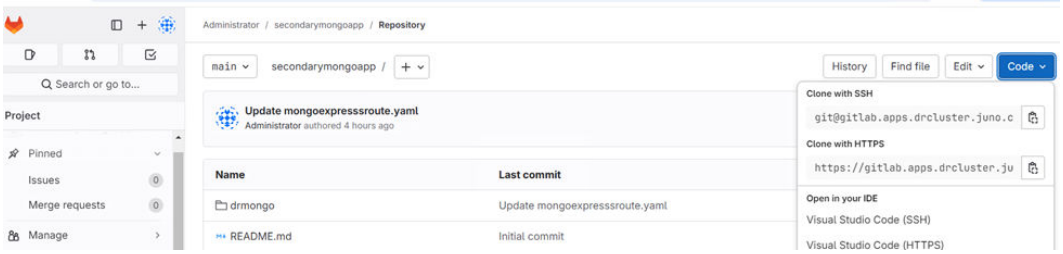


12. Verify the replication status from the OpenShift client machine.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                 STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication  Split    failover       none        3h33m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                 STATUS     DESIREDSTATE   OPERATION   AGE
mongodb-replication  Failover   failover       none        3h34m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication        true    3h34m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication        true    3h34m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                    READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication        true    3h34m

NAME                                                     READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication        true    3h34m
[root@occlient ~]#
```

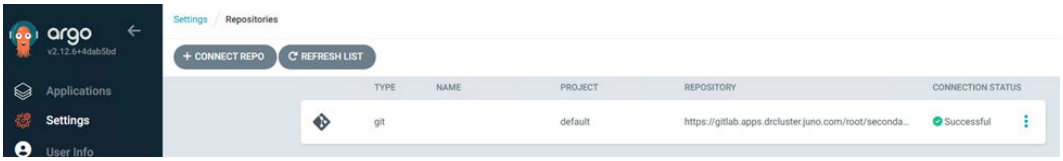
In a failover scenario assuming that the primary site is inaccessible, the secondary Red Hat OpenShift GitOps Argo CD instance is used to deploy the MongoDB application with the replicated PVC on the secondary cluster.

13. Copy the HTTPS URL of the GitLab project “secondarymongoapp” in the secondary cluster.

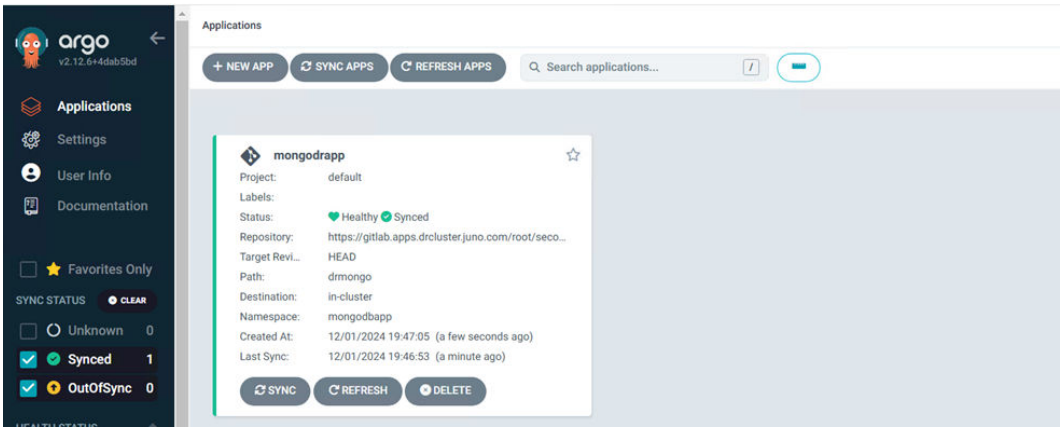


14. In the secondary Red Hat GitOps Argo CD instance, click **Settings**, then click **+CONNECT REPO**.
15. Select “VIA HTTPS”, type “git”, Project “default”, enter the copied URL in Repository URL, username as “root” and password as GitLab root password. Click on the check box Skip server verification.

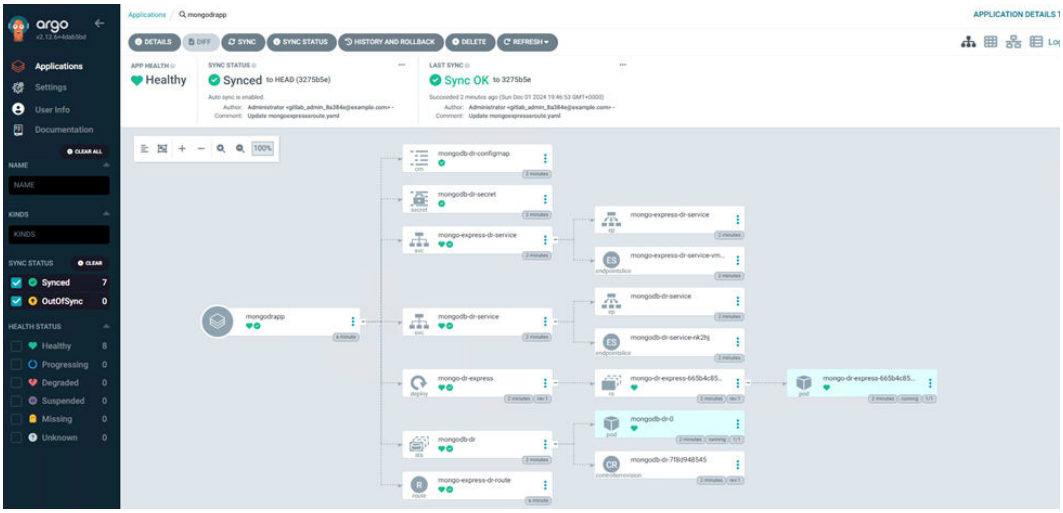
    The repository is added.



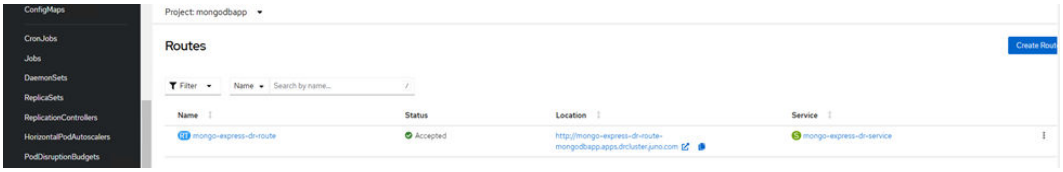
16. Change the **mongodbapp.yaml** file with the PVC name instead of **volumeClaimTemplates** as described.
17. From the Red Hat OpenShift GitOps Argo CD console, click **Applications**, then **+NEW APP**.
18. Create a new application with the following information and click **Create**.
    a. Application Name as **mongodrapp**
    b. Project name as **default**
    c. Sync Policy as **Automatic**
    d. Select the repository URL for GitLab project **secondarymongoapp**
    e. Select the Path as drmongo
    f. Select the local cluster for **Cluster URL**
    g. Select Namespace as **mongodbapp**
19. Verify that the application is created.



20. Click the application and verify all the components are ready and sync status is **OK**.

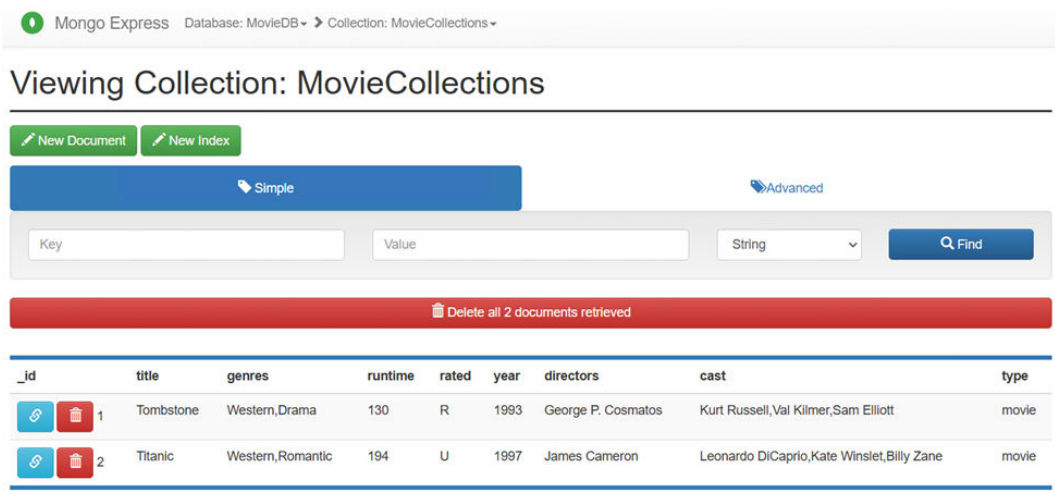


21. On the secondary cluster, click **Networking**, then **Route** and select the **mongodbapp** project.

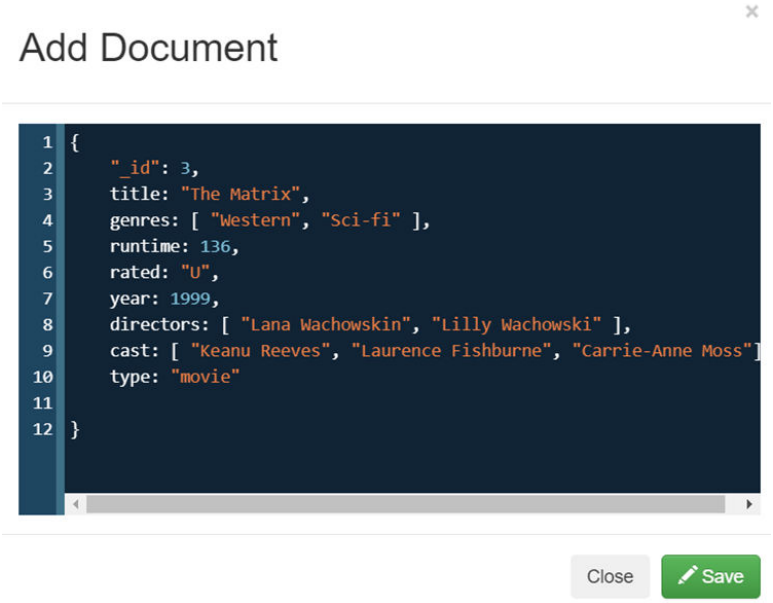


22. Click the **URL** under **Location for Mongo Express** route, log in with the default username and password as admin and pass.

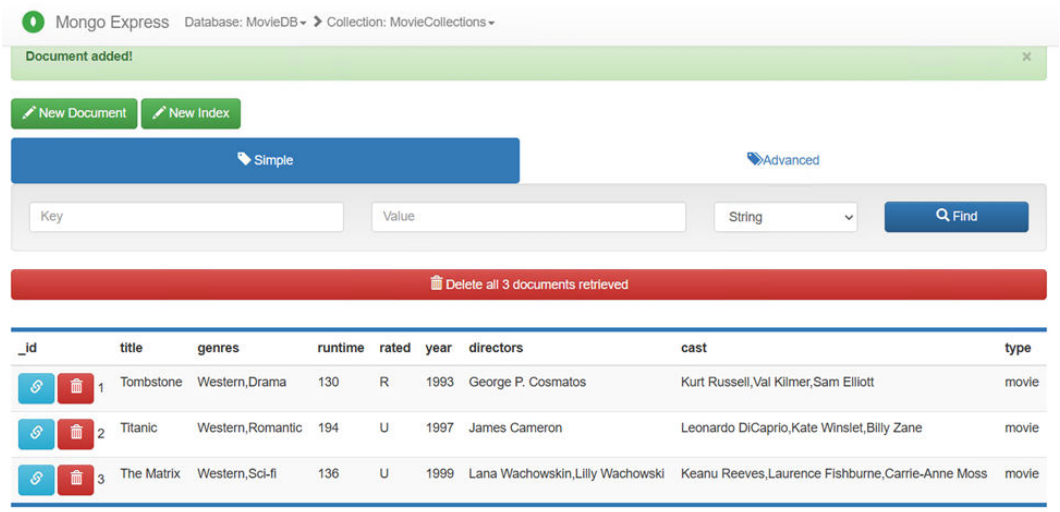
23. Verify that the MovieDB database and its contents are present.



24. Add a **New Document** to the **MovieCollections** collection.



25. Verify that the new **Document** is added.



Failback operation

After failover, business can continue on the secondary site within RPO. If you plan to recover the primary site after the outage window completes, perform failback operation.

The following are the high-level steps for the failback operation.

Procedure

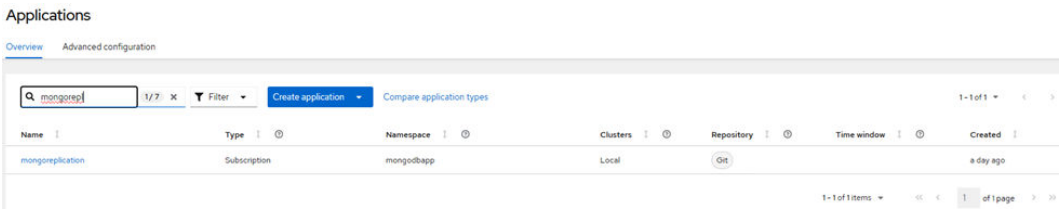
1. Clean up the primary site.
2. Create the replication CR in the secondary site.
3. Perform failover from the primary site.
   a. Edit the replication CR in the primary site and change spec.desiredPairState to **failover**.
   b. Confirm that replication CR status is **failover**, and the operation value is **none**.
   c. Deploy the stateful MongoDB application in the primary site with the replicated PVC.
4. Delete the replication CR, MongoDB application and the PVC from the secondary site.
5. Create the replication CR in the primary site.

Clean up the primary site

Clean up the primary site after recovery.

Procedure

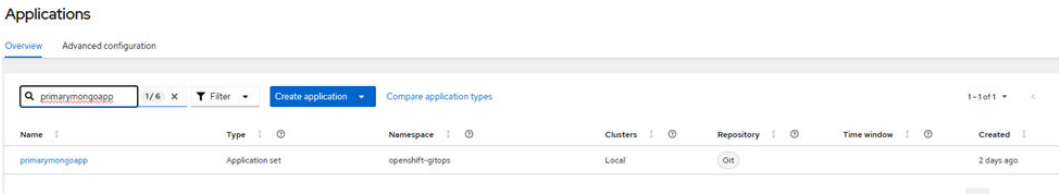
1. Delete the application **mongoreplication** created on the primary cluster from Red Hat Advanced Cluster Management by clicking the 3 dots, and then click **Delete Application**.



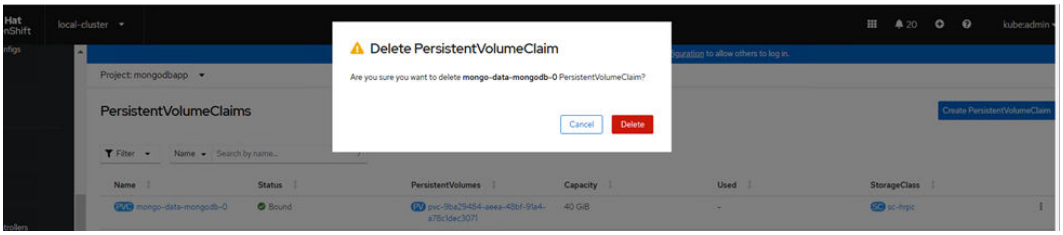
2. Confirm that all resources for **replication**, **localvolume** and **remotevolume** are deleted.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]#
```

3. Delete the application **primarymongoapp** created on the primary cluster.



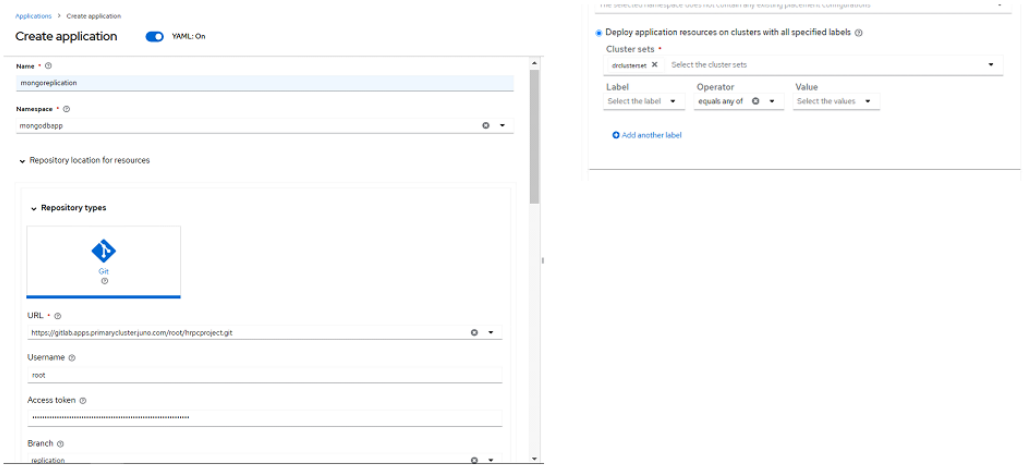
4. Remove the PVC **mongo-data-mongodb-0** from the primary cluster.



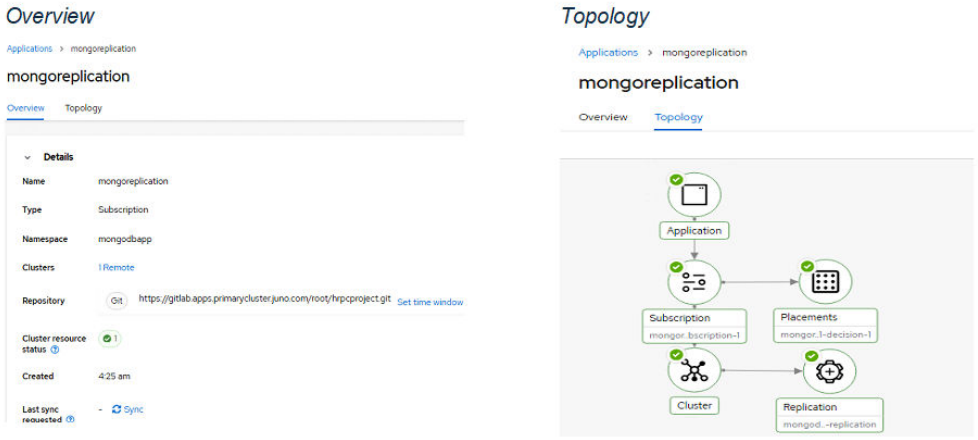
Create the replication CR in the secondary site

Procedure

1. Create a subscription-based application to replicate the MongoDB PVC from the VSP One Block 28 to the VSP One Block 24 storage system. Here the application name is **mongoreplication**, **Namespace** is **mongodbapp**, select Git, add the URL of **hrpcproject**, and select **Branch** as **replication**, and in the **Cluster sets** section, select **drclusterset**.

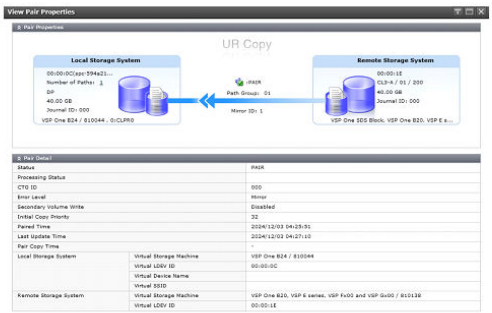


The application is created.

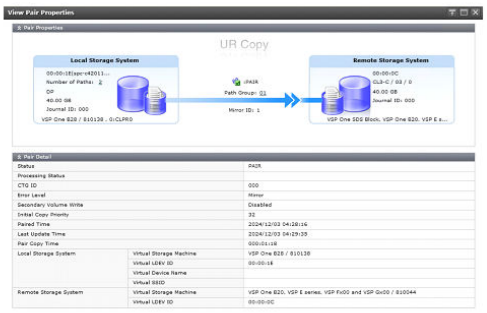


2. Verify the replication status from Storage Navigator.

Status from VSP One Block 24



Status from VSP One Block 28



3. Verify the replication CR details from both clusters.

Status from Primary Cluster

Replications › Replication details

mongodb-replication

Details YAML

```
69  spec:
70    desiredPairState: none
71    persistentVolumeClaimName: mongo-data-mongodb-0
72    persistentVolumeClaimSpec:
73      accessModes:
74        - ReadWriteOnce
75      resources:
76        requests:
77          storage: 40Gi
78      storageClassName: sc-hrpc
79      volumeMode: Filesystem
80      volumeName: pvc-292b9b0c-4d19-4810-b805-20cc20805cd4
81    replicationAttribute: secondary
82    storageClassName: sc-hrpc
83  status:
84    currentOperation:
85      operation: none
86      running: false
87    replicationPair:
88      localSerialNumber: 810044
89      localStorageDeviceId: A34000810044
90      primaryVolumeId: 30
91      primaryVolumeNickname: spc-c420116eb7
92      remoteSerialNumber: 810138
93      remoteStorageDeviceId: A34000810138
94      secondaryVolumeId: 12
95      secondaryVolumeNickname: spc-594a21acc0
96    status: Ready
```

Status from Secondary Cluster

Search › replication details

mongodb-replication

Details YAML Related resources

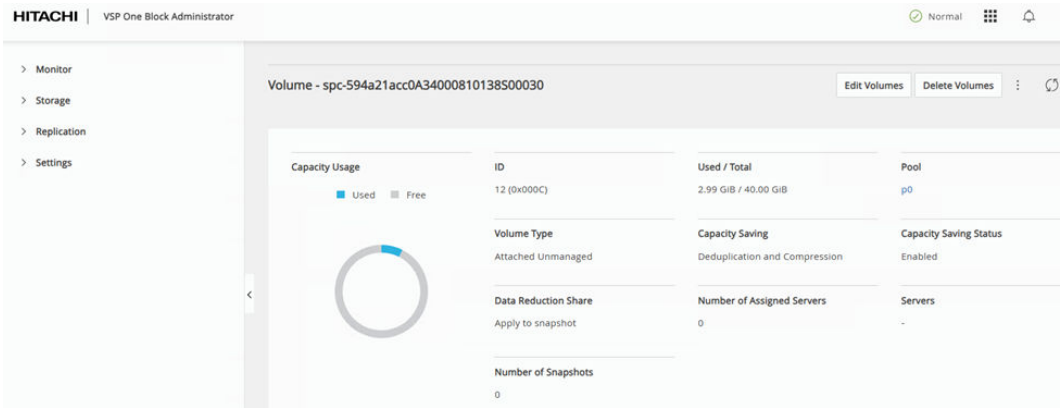
Cluster: **drcluster** Namespace: **mongodbapp**

```
77  spec:
78    desiredPairState: none
79    persistentVolumeClaimName: mongo-data-mongodb-0
80    replicationAttribute: primary
81    storageClassName: sc-hrpc
82  status:
83    currentOperation:
84      operation: none
85      running: false
86    replicationPair:
87      localSerialNumber: 810138
88      localStorageDeviceId: A34000810138
89      primaryVolumeId: 30
90      primaryVolumeNickname: spc-c420116eb7
91      remoteSerialNumber: 810044
92      remoteStorageDeviceId: A34000810044
93      secondaryVolumeId: 12
94      secondaryVolumeNickname: spc-594a21acc0
95    status: Ready
```

4. Verify the replication status from the OpenShift client machine.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        8m2s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        9m7s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                  READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication      true    8m56s

NAME                                                   READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication      true    8m33s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                  READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication      true    10m

NAME                                                   READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication      true    10m
[root@occlient ~]#
```

Hitachi Replication Plug-in for Containers (HRPC) in combination with Hitachi Storage Plug-in for Containers (HSPC) creates a DRS persistent volume (LDEV ID 00:0C) on the primary cluster.

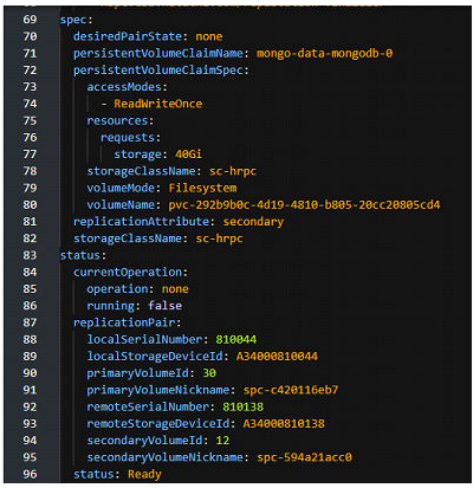


Perform failover to the primary site

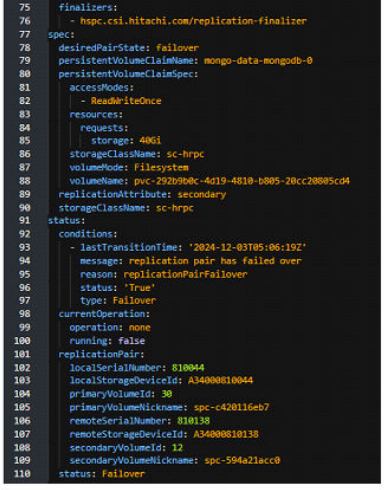
Procedure

1. From the primary site, change the status of replication CR to perform failover operation. From the primary cluster Red Hat OpenShift console, click **Administration**, then **CustomResourceDefinitions**, search for **replication**, and then click **Replication**. Click **Instances**, **mongodb-replication**, and then select **YAML** view.
2. Edit the **spec.desiredPairState** to **failover**, click save, then reload. This triggers the Hitachi Replication Plug-in for Containers to fail over the UR pair. The following screenshots show the status of the Replication CR from the primary cluster.

Before Failover

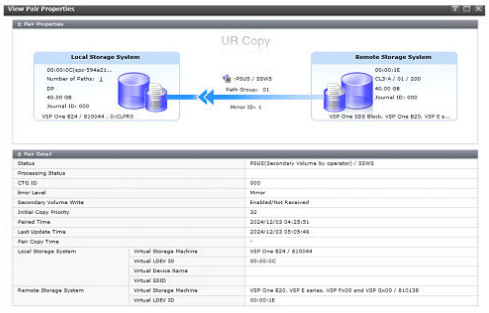


After Failover

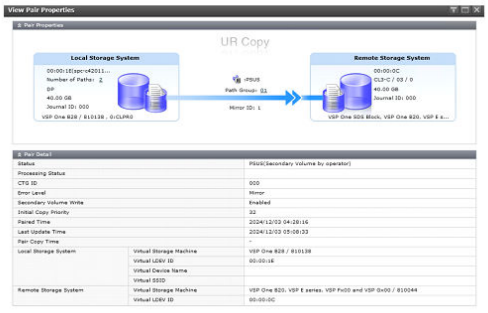


3. Verify the replication status from Storage Navigator.

Status from VSP One Block 24



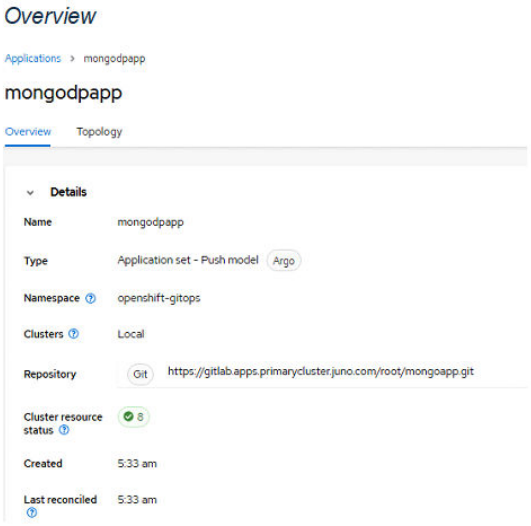
Status from VSP One Block 28

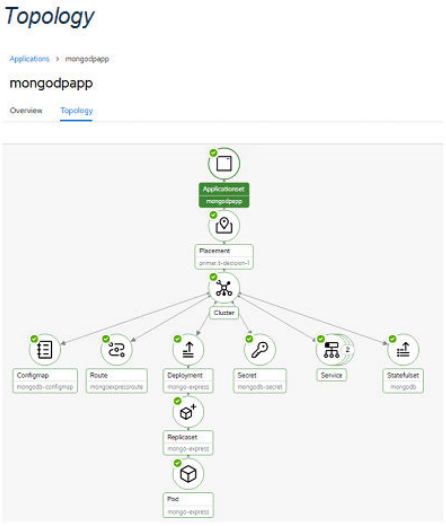


4. Verify the replication status from the OpenShift client machine.

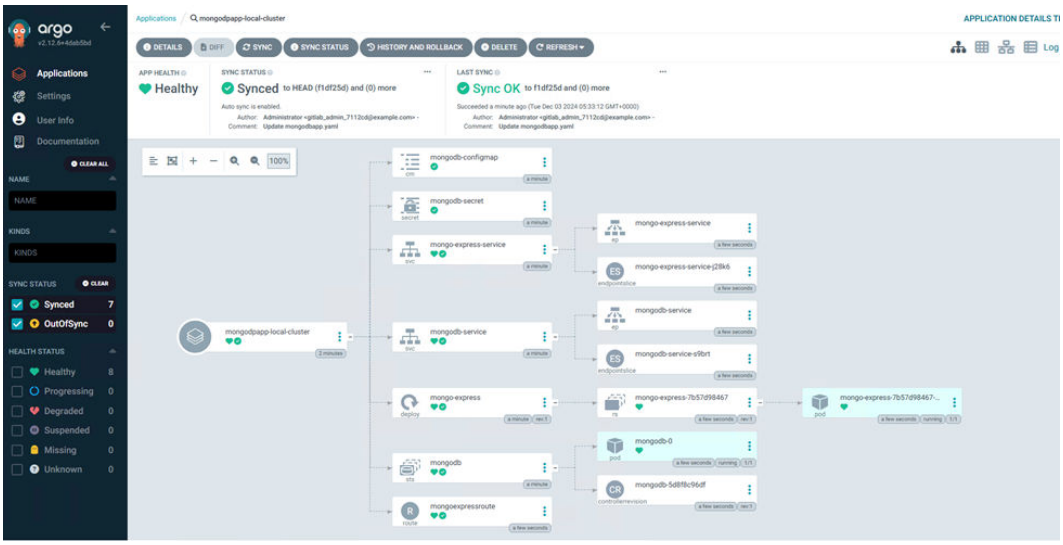
```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                  STATUS     DESIREDSTATE   OPERATION   AGE
mongodb-replication   Failover   failover       none        44m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Split    failover       none        45m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication    true    45m

NAME                                                 READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication    true    45m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication    true    46m

NAME                                                 READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication    true    45m
[root@occlient ~]#
```

5. Change the **mongodbapp.yaml** file in the primary GitLab project with the PVC name instead of **volumeClaimTemplates**.
6. Deploy the MongoDB application from Red Hat Advanced Cluster Management. Select **Applications**, click **Create Application**, and then select **Argo CD Application set – Push model**.
7. Verify that the application is deployed from both **Overview** and **Topology**.

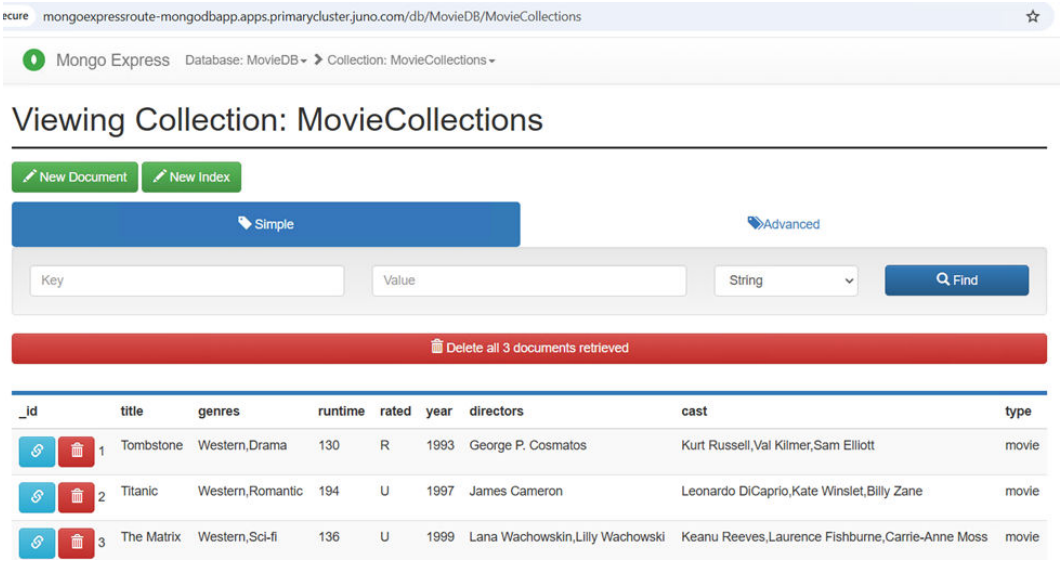




8. Verify that all the resources for the MongoDB application are created and sync status is **OK**.

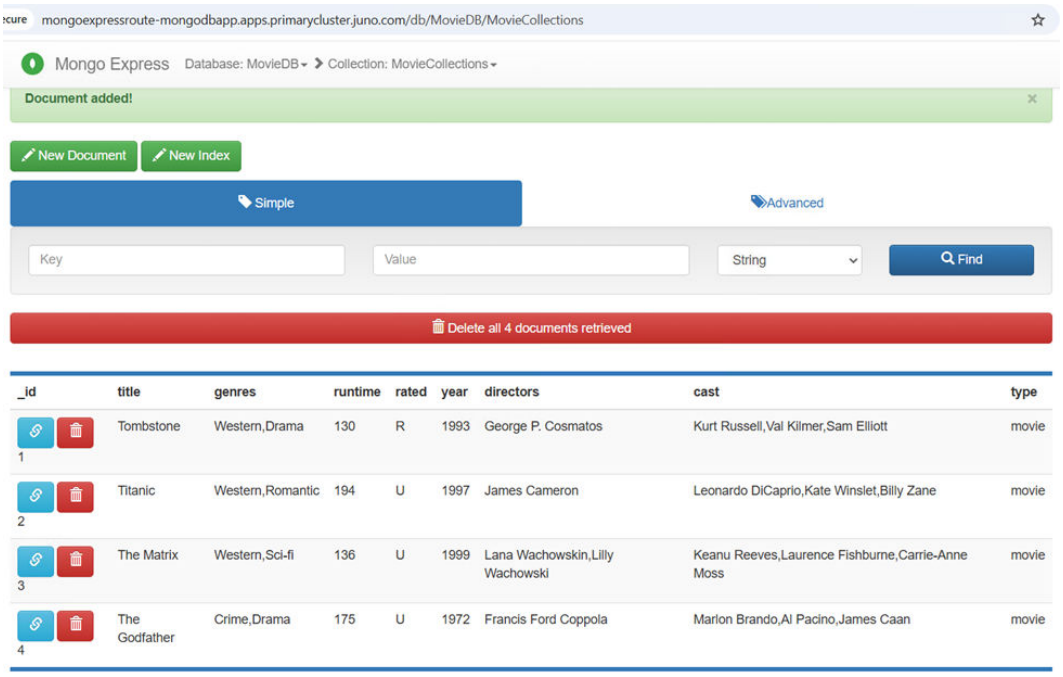


9. From the OpenShift console, click the mongo-express route and log in with default username and password as admin and pass.
10. Click MovieDB and confirm that entries made on the secondary cluster during the failover operation are available.



11. Add a new entry to **MovieCollections**.

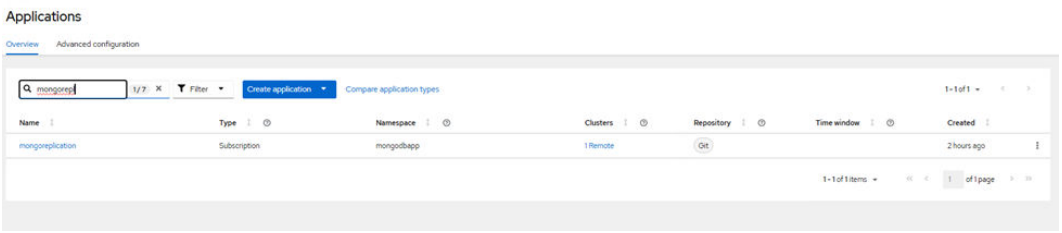
12. View the collection.



Delete the Replication CR, MongoDB application, and the PVC from the secondary site

Procedure

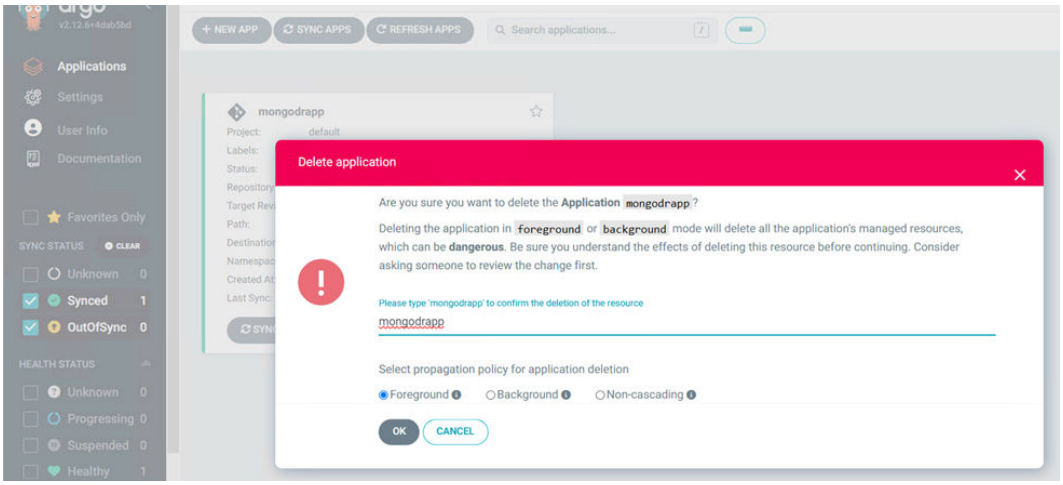
1. Delete the replication CR created previously. From Red Hat Advanced Cluster Management Applications, select the **mongoreplication**, click the 3 dots, and then click **Delete Application**.



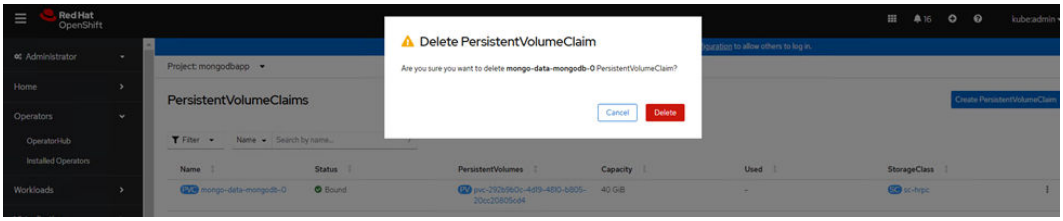
2. Confirm that all resources for **replication**, **localvolume**, and **remotevolume** are deleted.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
No resources found in mongodbapp namespace.
[root@occlient ~]#
```

3. From the Red Hat OpenShift GitOps Argo CD console in the secondary cluster, delete the **MongoDB** application created previously.



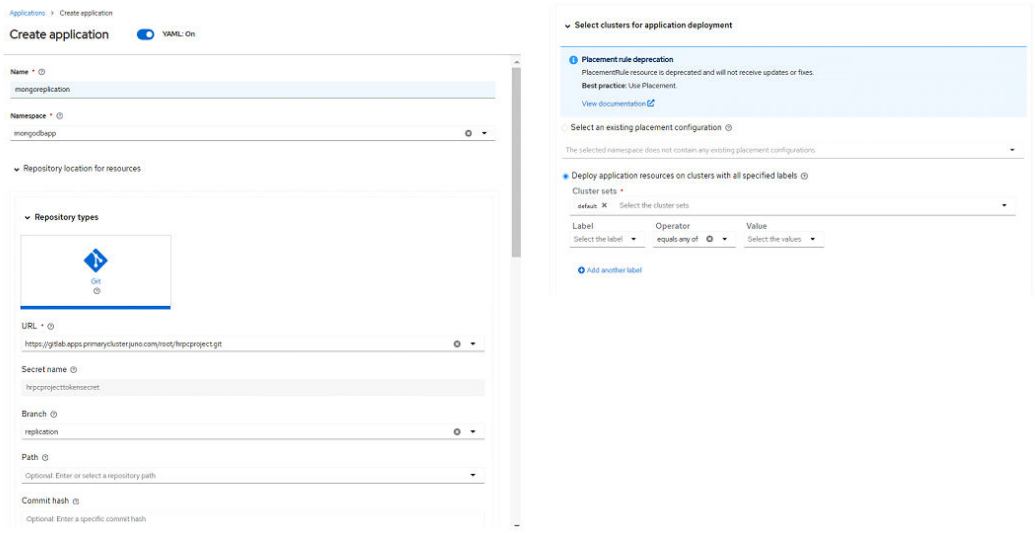
4. Delete the PVC from the secondary cluster.



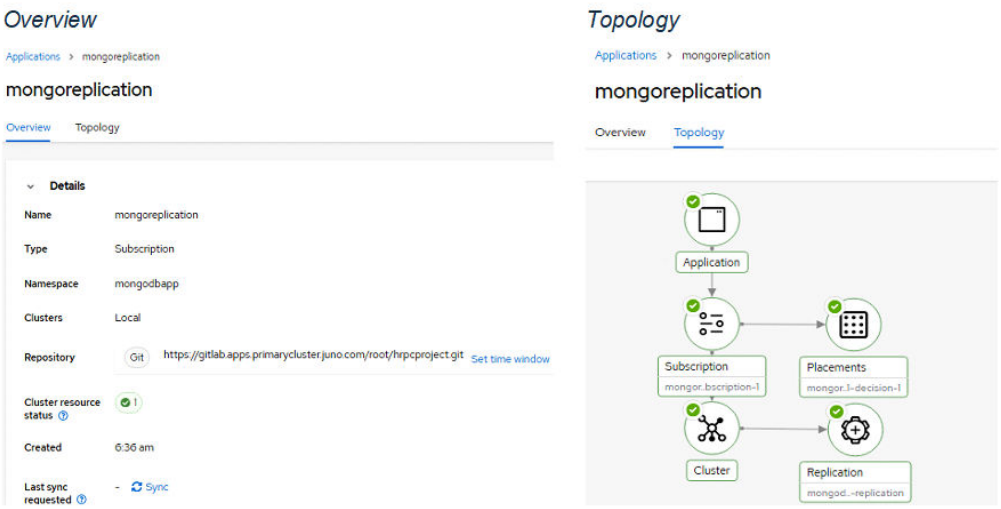
Create the Replication CR in the primary site

Procedure

1. Create a subscription-based application to replicate the MongoDB PVC from VSP One Block 24 to VSP One Block 28. Here the application name is **mongoreplication**, Namespace is **mongodbapp**, select Git, add the URL of **hrpcproject**, select Branch as **replication**, and in the **Cluster sets** section, select **default**.

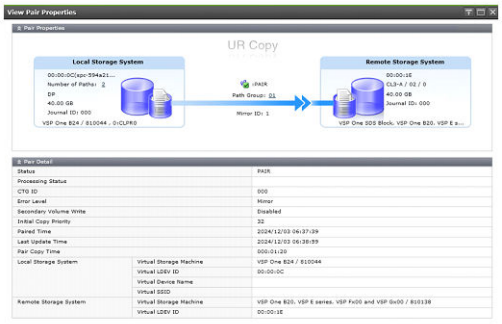


The application is created.

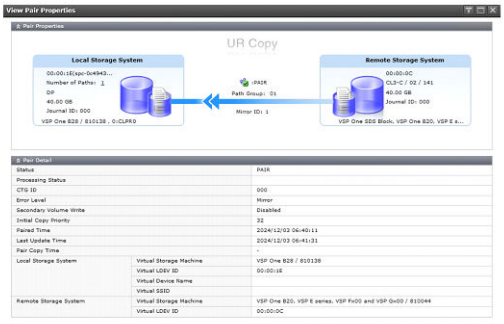


2. Verify the replication status from Storage Navigator.

Status from VSP One Block 24



Status from VSP One Block 28



3. Verify the replication CR details from both clusters.

Status from Primary Cluster

Search > replication details

mongodb-replication

Details YAML Related resources

Cluster: **local-cluster** Namespace: **mongodbapp**

```
77  spec:
78    desiredPairState: none
79    persistentVolumeClaimName: mongo-data-mongodb-0
80    replicationAttribute: primary
81    storageClassName: sc-hrpc
82  status:
83    currentOperation:
84      operation: none
85      running: false
86    replicationPair:
87      localSerialNumber: 810044
88      localStorageDeviceId: A34000810044
89      primaryVolumeId: 12
90      primaryVolumeNickname: spc-594a21acc0
91      remoteSerialNumber: 810138
92      remoteStorageDeviceId: A34000810138
93      secondaryVolumeId: 30
94      secondaryVolumeNickname: spc-0c4943c04c
95    status: Ready
```

Status from Secondary Cluster

Replications > Replication details

mongodb-replication

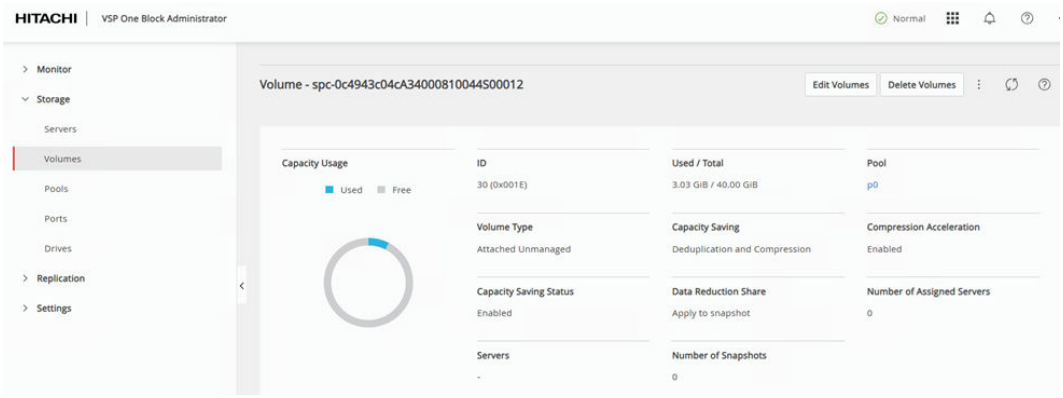
Details YAML

```
69  spec:
70    desiredPairState: none
71    persistentVolumeClaimName: mongo-data-mongodb-0
72    persistentVolumeClaimSpec:
73      accessModes:
74        - ReadWriteOnce
75      resources:
76        requests:
77          storage: 40Gi
78      storageClassName: sc-hrpc
79      volumeMode: Filesystem
80      volumeName: pvc-e4a26305-1a12-49f6-a954-d266bdaa2022
81    replicationAttribute: secondary
82    storageClassName: sc-hrpc
83  status:
84    currentOperation:
85      operation: none
86      running: false
87    replicationPair:
88      localSerialNumber: 810138
89      localStorageDeviceId: A34000810138
90      primaryVolumeId: 12
91      primaryVolumeNickname: spc-594a21acc0
92      remoteSerialNumber: 810044
93      remoteStorageDeviceId: A34000810044
94      secondaryVolumeId: 30
95      secondaryVolumeNickname: spc-0c4943c04c
96    status: Ready
```

4. Verify the replication status from the OpenShift client machine.

```
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        9m1s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                  STATUS   DESIREDSTATE   OPERATION   AGE
mongodb-replication   Ready    none           none        8m11s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get localvolume,remotevolume -n mongodbapp
NAME                                                  READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication      true    8m26s

NAME                                                   READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication      true    8m25s
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                                                  READY   AGE
localvolume.hspc.hitachi.com/mongodb-replication      true    8m54s

NAME                                                   READY   AGE
remotevolume.hspc.hitachi.com/mongodb-replication      true    8m43s
[root@occlient ~]#
```

Hitachi Replication Plug-in for Containers (HRPC) in combination with Hitachi Storage Plug-in for Containers (HSPC) creates a DRS persistent volume (LDEV ID 00:30) on the secondary cluster.



Conclusion

Red Hat Advanced Cluster Management for Kubernetes, combined with Hitachi Replication Plug-in for Containers (HRPC), simplifies container data mobility and disaster recovery for stateful containerized applications from a centralized management console. This hybrid cloud storage architecture supports asynchronous, bi-directional data replication for stateful containerized applications without any distance limitations.

Hitachi Vantara

Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA

HitachiVantara.com/contact