

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

Reference Architecture Guide

© 2025 Hitachi Vantara LLC. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including copying and recording, or stored in a database or retrieval system for commercial purposes without the express written permission of Hitachi, Ltd., Hitachi Vantara, Ltd., or Hitachi Vantara LLC (collectively "Hitachi"). Licensee may make copies of the Materials provided that any such copy is: (i) created as an essential step in utilization of the Software as licensed and is used in no other manner; or (ii) used for archival purposes. Licensee may not make any other copies of the Materials. "Materials" mean text, data, photographs, graphics, audio, video and documents.

Hitachi reserves the right to make changes to this Material at any time without notice and assumes no responsibility for its use. The Materials contain the most current information available at the time of publication.

Some of the features described in the Materials might not be currently available. Refer to the most recent product announcement for information about feature and product availability, or contact Hitachi Vantara LLC at https://support.hitachivantara.com/en_us/contact-us.html.

Notice: Hitachi products and services can be ordered only under the terms and conditions of the applicable Hitachi agreements. The use of Hitachi products is governed by the terms of your agreements with Hitachi Vantara LLC.

By using this software, you agree that you are responsible for:

1. Acquiring the relevant consents as may be required under local privacy laws or otherwise from authorized employees and other individuals; and
2. Verifying that your data continues to be held, retrieved, deleted, or otherwise processed in accordance with relevant laws.

Notice on Export Controls. The technical data and technology inherent in this Document may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Reader agrees to comply strictly with all such regulations and acknowledges that Reader has the responsibility to obtain licenses to export, re-export, or import the Document and any Compliant Products.

Hitachi and Lumada are trademarks or registered trademarks of Hitachi, Ltd., in the United States and other countries.

AIX, DB2, DS6000, DS8000, Enterprise Storage Server, eServer, FICON, FlashCopy, GDPS, HyperSwap, IBM, IntelliMagic, IntelliMagic Vision, OS/390, PowerHA, PowerPC, S/390, System z9, System z10, Tivoli, z/OS, z9, z10, z13, z14, z15, z16, z17, z/VM, and z/VSE are registered trademarks or trademarks of International Business Machines Corporation.

Active Directory, ActiveX, Bing, Excel, Hyper-V, Internet Explorer, the Internet Explorer logo, Microsoft, Microsoft Edge, the Microsoft corporate logo, the Microsoft Edge logo, MS-DOS, Outlook, PowerPoint, SharePoint, Silverlight, SmartScreen, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, the Windows logo, Windows Azure, Windows PowerShell, Windows Server, the Windows start button, and Windows Vista are registered trademarks or trademarks of Microsoft Corporation. Microsoft product screen shots are reprinted with permission from Microsoft Corporation.

All other trademarks, service marks, and company names in this document or website are properties of their respective owners.

The open source content used in Hitachi Vantara products may be found within the Product documentation or you may request a copy of such information (including source code and/or modifications to the extent the license for any open source requires Hitachi make it available) by sending an email to OSS_licensing@hitachivantara.com.

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](#)

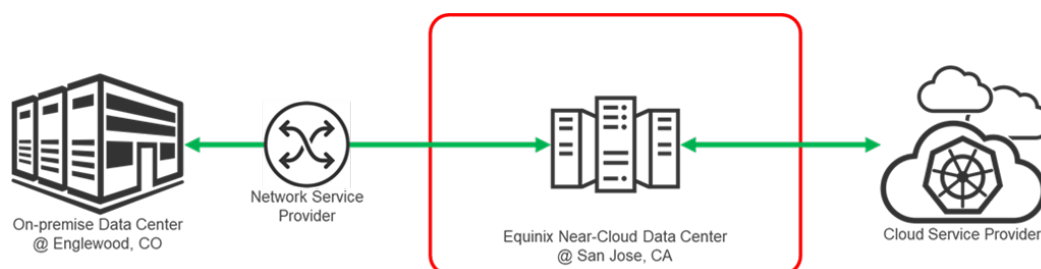
- [Hitachi Replication Plug-in for Containers Configuration Guide](#)
- [Red Hat Advanced Cluster Management](#)
- [GitLab Documentation](#)

Executive summary

- The Hitachi **Virtual Storage Platform 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

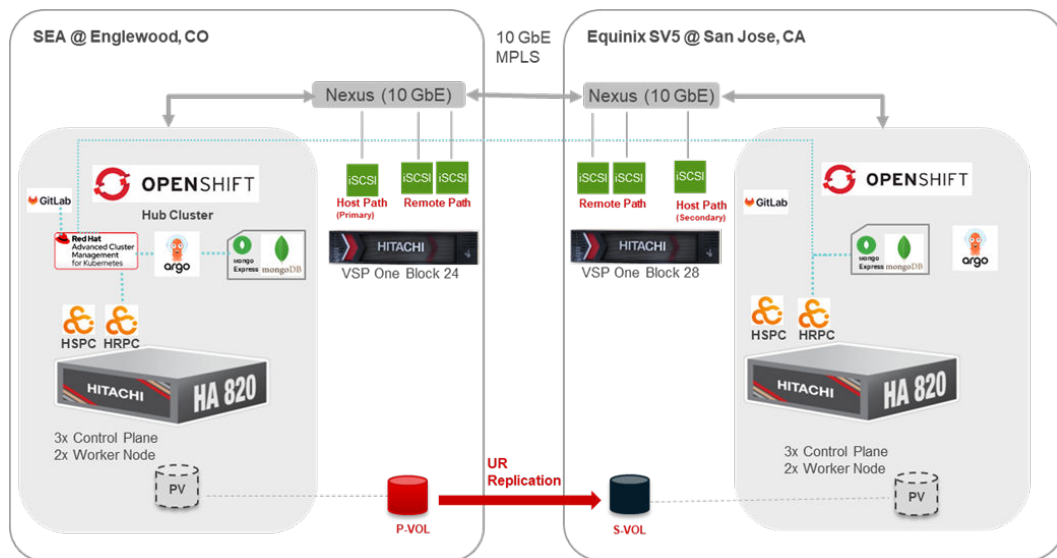
The following table lists the software specifications used in this solution.

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
<p>Prepare the environment:</p> <ol style="list-style-type: none"> 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. 	<p>Environment is set up according to specifications.</p>

Description	Success Criteria
<ol style="list-style-type: none"> 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. 	
<p>Configure Hitachi Replication Plugin for Containers:</p> <ol style="list-style-type: none"> 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. 	<p>Verify that HRPC is deployed and running in both clusters.</p>
<p>Data mobility: Migrate stateful application using HRPC</p> <ol style="list-style-type: none"> 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. 	<p>Volume replication can be performed using HRPC from on-premises to near cloud (Equinix).</p> <p>Verify container data mobility to secondary site is successful using Red Hat Advanced Cluster Management.</p>
<p>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.</p> <p>Failover operation:</p> <ol style="list-style-type: none"> 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. <p>Failback operation:</p>	<p>Validate that data can be protected during an outage across the sites using HRPC and Red Hat Advanced Cluster Management.</p>

Description	Success Criteria
<ol style="list-style-type: none"> 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. <ul style="list-style-type: none"> ▪ 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

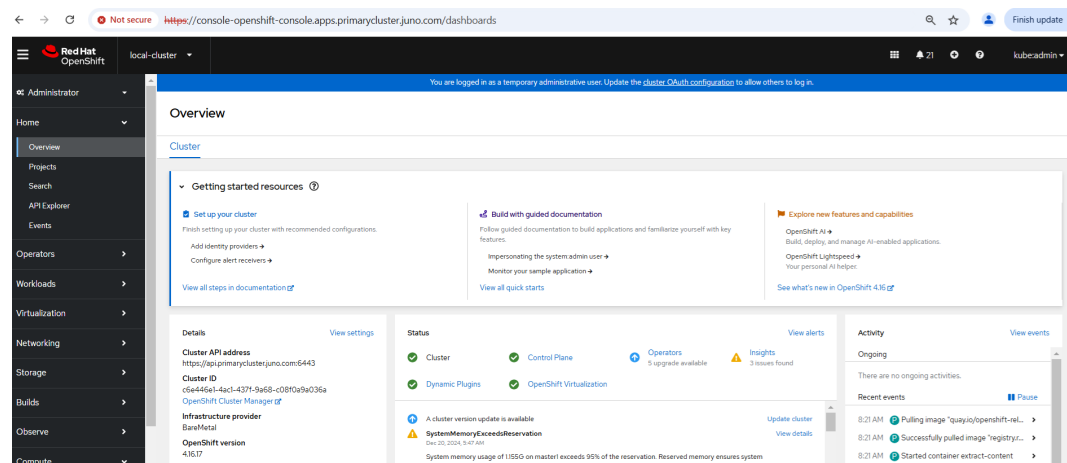
Before you begin

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:



- 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.

Name	Status	Roles	Pods	Memory	CPU	Filesystem	Created	Instance type
master1	Ready	control-plane, master	46	16.47 GB / 188.5 GB	13.885 cores / 64 cores	458 GB / 558.4 GB	Oct 16, 2024, 10:56 AM	-
master2	Ready	control-plane, master	56	18.18 GB / 188.5 GB	2.341 cores / 64 cores	152.2 GB / 558.4 GB	Oct 16, 2024, 10:58 AM	-
master3	Ready	control-plane, master	65	18.03 GB / 188.5 GB	2.290 cores / 64 cores	122.9 GB / 558.4 GB	Oct 16, 2024, 11:20 AM	-
worker1	Ready	worker	136	29.93 GB / 188.5 GB	2.370 cores / 64 cores	70.14 GB / 558.4 GB	Oct 16, 2024, 11:20 AM	-
worker2	Ready	worker	91	18.27 GB / 188.5 GB	1.721 cores / 64 cores	94.16 GB / 558.4 GB	Oct 16, 2024, 11:36 AM	-

- Red Hat OpenShift Cluster on the secondary site:

Cluster Overview

Getting started resources

- Set up your cluster
- Build with guided documentation
- Explore new features and capabilities

Details

- Cluster API address: https://api.dcluster.juno.com/6443
- Cluster ID: 6499d331-f065-4028-9794-7770c3303a9
- Infrastructure provider: BareMetal
- OpenShift version: 4.16.17

Status

- Cluster: ✔
- Control Plane: ✔
- Dynamic Plugins: ✔
- OpenShift Virtualization: ✔
- Operators: 4 upgrade available
- Insights: 2 issues found

Activity

Recent events:

- 8:21 AM: unexpected on-disk state valida...
- 8:20 AM: All applied components are avail...
- 8:20 AM: Stopping container registry-server

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

Name	Status	Roles	Pods	Memory	CPU	Filesystem	Created	Instance type
master1	Ready	control-plane, master	53	16.36 GB / 251.5 GB	3.195 cores / 48 cores	541.7 GB / 873.7 TB	Oct 26, 2024, 11:52 AM	-
master2	Ready	control-plane, master	40	14.15 GB / 251.5 GB	1.067 cores / 48 cores	427.4 GB / 744.7 GB	Oct 26, 2024, 12:15 PM	-
master3	Ready	control-plane, master	73	19.17 GB / 251.5 GB	1.906 cores / 48 cores	175.9 GB / 744.7 GB	Oct 26, 2024, 11:56 AM	-
worker1	Ready	worker	63	18.28 GB / 188.5 GB	1.258 cores / 64 cores	83.2 GB / 279.1 GB	Oct 26, 2024, 12:35 PM	-
worker2	Ready	worker	71	20.45 GB / 188.5 GB	1.229 cores / 64 cores	96 GB / 279.1 GB	Oct 26, 2024, 12:36 PM	-

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 `iscsid` 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.

MachineConfigs > MachineConfig details

MC worker-enable-iscsi

Details YAML Events

```

1  apiVersion: machineconfiguration.openshift.io/v1
2  kind: MachineConfig
3  metadata:
4    creationTimestamp: '2024-11-12T07:43:43Z'
5    generation: 1
6    labels:
7      machineconfiguration.openshift.io/role: worker
8  > managedFields: ...
30  name: worker-enable-iscsi
31  resourceVersion: '7479823'
32  uid: 360c4d93-64f5-4421-b29d-e40860ee0497
33  spec:
34    config:
35      ignition:
36        version: 3.2.0
37      systemd:
38        units:
39          - enabled: true
40            name: iscsid.service
41            state: started
42    osImageURL: ''

```

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).

The screenshot shows the 'Remote Path Group - 1' configuration page. At the top, there is a summary table with the following data:

Model	Serial Number	Protocol	Number of Remote Paths
VSP One B20, VSP E series, VSP Fx00 and VSP Gx00	810138	iSCSI	2

Below this is the 'Remote Paths' section, which includes a filter, a table of paths, and an 'Add Remote Paths' button. The table shows two paths:

	Port	Remote Port	Remote IP Address	Remote TCP Port Number	Protocol	Status
<input type="checkbox"/>	CL1-C	CL1-A	172.23.31.60	3260	iSCSI	Normal
<input type="checkbox"/>	CL2-C	CL2-A	172.23.31.61	3260	iSCSI	Normal

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

The screenshot shows the 'Remote Path Group - 1' configuration page for the secondary storage system. At the top, there is a summary table with the following data:

Model	Serial Number	Protocol	Number of Remote Paths
VSP One B20, VSP E series, VSP Fx00 and VSP Gx00	810044	iSCSI	2

Below this is the 'Remote Paths' section, which includes a filter, a table of paths, and an 'Add Remote Paths' button. The table shows two paths:

	Port	Remote Port	Remote IP Address	Remote TCP Port Number	Protocol	Status
<input type="checkbox"/>	CL1-A	CL1-C	172.23.31.56	3260	iSCSI	Normal
<input type="checkbox"/>	CL2-A	CL2-C	172.23.31.57	3260	iSCSI	Normal

- 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

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

Project: kube-system

Installed Operators > Operator details

Hitachi Storage Plug-in for Containers
1.14.3 provided by Hitachi

Details | YAML | Subscription | Events | HSPC

Provided APIs

HSPC
HSPC is the Schema for the hspcs API

[Create instance](#)

Description

About

Hitachi Storage Plug-in for Containers is a software plugin component for Kubernetes and OpenShift environments that is used to dynamically create and manage persistent block storage from Hitachi VSP One storages.

For full documentation, go to our [Product Documentation site](#) and refer to the reference guide for HSPC v3.14.3.

Requirements

Supported Driver Version

- HSPC v3.14.3

Provider
Hitachi

Created at
Nov 12, 2024, 7:27 AM

Links
Product Documentation
<https://docs.hitachivantara.com/home#zf>

Maintainers
Hitachi
Container-info@hitachivantara.com

2. Create the HSPC instance on both clusters.

Installed Operators > Operator details

Hitachi Storage Plug-in for Containers
1.14.3 provided by Hitachi

Details | YAML | Subscription | Events | **HSPC**

HSPCs [Create HSPC](#)

Name

Name	Kind	Status	Labels	Last updated
HSPC hspc	HSPC	-	No labels	Nov 5, 2024, 2:57 PM

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](#)

```

1 kind: Secret
2 apiVersion: v1
3 metadata:
4   name: secret-b24
5   namespace: default
6   uid: 331f51bf-175a-473d-ba36-0c919e04f015
7   resourceVersion: '13529823'
8   creationTimestamp: '2024-11-06T05:08:02Z'
9   managedFields: ...
22 data:
23   password: a3ViZXJvZXRLcw==
24   url: aHR0cDovLzE3Mi4yMy42OC4xMzQ=
25   user: a3ViZXJvZXRLcw==
26 type: Opaque

```

VSP One Block 28

Secrets > Secret details

secret-b28

Details [YAML](#)

```

1 kind: Secret
2 apiVersion: v1
3 metadata:
4   name: secret-b28
5   namespace: default
6   uid: ec0010e9-ef53-4f15-bc33-3925ec5feedd
7   resourceVersion: '7477172'
8   creationTimestamp: '2024-11-12T07:35:42Z'
9   managedFields: ...
22 data:
23   password: a3ViZXJvZXRLcw==
24   url: aHR0cDovLzE3Mi4yMy4zMC4xMzQ=
25   user: a3ViZXJvZXRLcw==
26 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-hrpsc

Details [YAML](#)

```

1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: sc-hrpsc
5   uid: ee8ba470-39fe-4cf2-9541-f3ca9f0e66c4
6   resourceVersion: '13561017'
7   creationTimestamp: '2024-11-06T05:58:19Z'
8   annotations:
9     kubernetes.io/description: Hitachi Storage Plug-in for Containers
10    storageclass.kubernetes.io/is-default-class: 'true'
11   managedFields: ...
44 provisioner: hspc.csi.hitachi.com
45 parameters:
46   csi.storage.k8s.io/fstype: ext4
47   csi.storage.k8s.io/provisioner-secret-namespace: default
48   csi.storage.k8s.io/provisioner-secret-name: secret-b24
49   csi.storage.k8s.io/node-stage-secret-name: secret-b24
50   csi.storage.k8s.io/controller-expand-secret-name: secret-b24
51   csi.storage.k8s.io/node-publish-secret-namespace: default
52   csi.storage.k8s.io/controller-publish-secret-name: secret-b24
53   csi.storage.k8s.io/controller-publish-secret-namespace: default
54   poolID: '0'
55   csi.storage.k8s.io/node-publish-secret-name: secret-b24
56   connectionType: iscsi
57   csi.storage.k8s.io/controller-expand-secret-namespace: default
58   portID: CL3-C
59   serialNumber: '010044'
60   csi.storage.k8s.io/node-stage-secret-namespace: default
61   reclaimPolicy: Delete
62   allowVolumeExpansion: true
63   volumeBindingMode: Immediate

```

VSP One Block 28

StorageClasses > StorageClass details

sc-hrpsc

Details [YAML](#)

```


1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: sc-hrpsc
5   uid: 50c63936-5c81-4e26-b4c5-cc4047cb0fa6
6   resourceVersion: '7480905'
7   creationTimestamp: '2024-11-12T07:39:23Z'
8   annotations:
9     kubernetes.io/description: Hitachi Storage Plug-in for Containers
10    storageclass.kubernetes.io/is-default-class: 'true'
11   managedFields: ...
52 provisioner: hspc.csi.hitachi.com
53 parameters:
54   csi.storage.k8s.io/fstype: ext4
55   csi.storage.k8s.io/provisioner-secret-namespace: default
56   csi.storage.k8s.io/provisioner-secret-name: secret-b28
57   csi.storage.k8s.io/node-stage-secret-name: secret-b28
58   csi.storage.k8s.io/controller-expand-secret-name: secret-b28
59   csi.storage.k8s.io/node-publish-secret-namespace: default
60   csi.storage.k8s.io/controller-publish-secret-name: secret-b28
61   csi.storage.k8s.io/controller-publish-secret-namespace: default
62   poolID: '0'
63   csi.storage.k8s.io/node-publish-secret-name: secret-b28
64   connectionType: iscsi
65   csi.storage.k8s.io/controller-expand-secret-namespace: default
66   portID: CL3-A
67   serialNumber: '010138'
68   csi.storage.k8s.io/node-stage-secret-namespace: default
69   reclaimPolicy: Delete
70   allowVolumeExpansion: true
71   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.

Name	Namespace	Managed Namespaces	Status	Last updated	Provided APIs
 Advanced Cluster Management for Kubernetes 2.11.3 provided by Red Hat	NS open-cluster-management	NS open-cluster-management	✔ Succeeded Up to date	Nov 16, 2024, 6:24 AM	MultiClusterHub

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


Installed Operators > Operator details

 Advanced Cluster Management for Kubernetes
2.11.3 provided by Red Hat

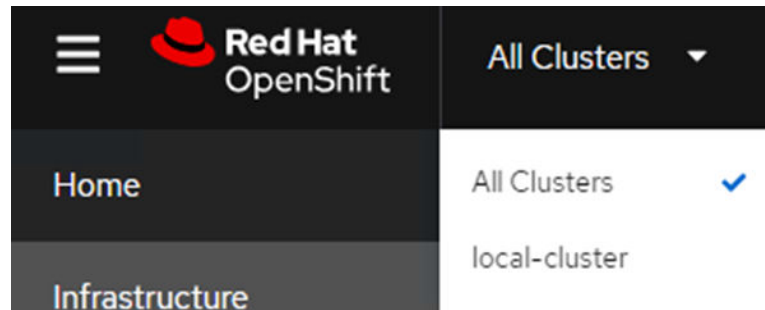
Details | YAML | Subscription | Events | MultiClusterHub | Actions

MultiClusterHubs Create MultiClusterHub

Name

Name	Kind	Status	Labels	Last updated
 multiclusterhub	MultiClusterHub	Phase: ✔ Running	No labels	Oct 29, 2024, 8:34 AM







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.

Cluster list | Cluster sets | Cluster pools | Discovered clusters

Search Create cluster Import cluster Actions 1-1 of 1

Name	Namespace	Status	Infrastructure	Control plane type	Distribution version	Labels	Nodes	Add-ons	Creation date
<input type="checkbox"/> local-cluster	local-cluster	✔ Ready	 Bare metal	Hub	OpenShift 4.16.17	 openshiftv...  openshiftv...  velero.io/e... 15 more	 5	 8	10/29/2024, 8:17:15 AM

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.

Clusters > Import an existing cluster

Import an existing cluster ? YAML

1 Details

2 Automation

3 Review

Details

Name *

drcluster

Cluster set

drclusterset

[Manage cluster sets](#)

Additional labels

Enter key=value, then press return, space, or comma

Import mode *

Kubeconfig

Credentials are temporarily stored in a secret until the cluster import succeeds or fails.

Kubeconfig *

```
aDMvTDI3YmpnTApzYWxISHZXTIZRUTZWRROEYazZnFIISDvYs2RwSDRSQkVrbkVyR2hRZTc0YmJobWnRxc9PQotLS0tLUVORCBSU0EgUUFJVVkFURS
BLRVktLS0tLGo=
```

[Next](#) [Back](#) [Cancel](#)

7. Skip Automation, and then click Import.

Clusters > Import an existing cluster

Import an existing cluster ? YAML

1 Details

2 Automation

3 Review

Details

Name drcluster

Cluster set drclusterset

Additional labels

Import mode Kubeconfig

Namespace

Credential

Kubeconfig *****

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”.

Clusters ?

Cluster list Cluster sets Cluster pools Discovered clusters Get started with Multicluster Hub

Name	Namespace	Status	Infrastructure	Control plane type	Distribution version	Labels	Nodes	Add-ons	Creation date
local-cluster	local-cluster	Ready	Bare metal	Hub	OpenShift 4.16.17 Upgrade available	openshiftVersion-major=4 openshiftVersion-minor=4.16 velero.io/exclude-from-backup=true IS more	5	2	10/29/2024, 8:17:15 AM
drcluster	drcluster	Ready	Bare metal	Standalone	OpenShift 4.16.17 Upgrade available	openshiftVersion-major=4 openshiftVersion-major-minor=4.16 IS more	5	2	10/29/2024, 9:14:59 AM

1 - 2 of 2 items 1 of 1 page






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.

Name	Namespace	Managed Namespaces	Status	Last updated	Provided APIs
 cert-manager Operator for Red Hat OpenShift 134.1 provided by Red Hat	 cert-manager-operator	 cert-manager-operator	 Succeeded Up to date	 Nov 16, 2024, 6:24 AM	CertificateRequest Certificate Challenge ClusterIssuer View 3 more...

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

Name	Namespace	Managed Namespaces	Status	Last updated	Provided APIs
 GitLab 16.1 provided by GitLab Inc	 gitlab-system	 gitlab-system	 Succeeded  Upgrade available	 Nov 25, 2024, 7:18 AM	GitLab

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

Installed Operators > gitlab-operator-kubernetes.v1.6.1 > GitLab details

gitlab Running

Details YAML Resources Events

```

1  apiVersion: apps.gitlab.com/v1beta1
2  kind: GitLab
3  metadata:
4    creationTimestamp: '2024-11-10T10:31:10Z'
5    generation: 1
6  > managedFields: ...
56 name: gitlab
57 namespace: gitlab-system
58 resourceVersion: '27064220'
59 uid: 9573c886-932d-4881-94ed-723acc65b28d
60 spec:
61   chart:
62     values:
63       certmanager:
64         install: false
65       gitlab:
66         webservice:
67           serviceAccount:
68             name: gitlab-app-nonroot
69       global:
70         hosts:
71           domain: apps.primarycluster.juno.com
72         ingress:
73           annotations:
74             route.openshift.io/termination: edge
75           class: none
76         nginx-ingress:
77           enabled: false
78         version: 8.5.1

```

Secondary Cluster

Installed Operators > gitlab-operator-kubernetes.v1.6.1 > GitLab details

gitlab Running

Details YAML Resources Events

```

1  apiVersion: apps.gitlab.com/v1beta1
2  kind: GitLab
3  metadata:
4    creationTimestamp: '2024-11-17T11:58:15Z'
5    generation: 1
6  > managedFields: ...
56 name: gitlab
57 namespace: gitlab-system
58 resourceVersion: '17802918'
59 uid: 287d3288-3579-44ea-bddc-49d1f8f2e776
60 spec:
61   chart:
62     values:
63       certmanager:
64         install: false
65       gitlab:
66         webservice:
67           serviceAccount:
68             name: gitlab-app-nonroot
69       global:
70         hosts:
71           domain: apps.drcluster.juno.com
72         ingress:
73           annotations:
74             route.openshift.io/termination: edge
75           class: none
76         nginx-ingress:
77           enabled: false
78         version: 8.5.2

```

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
```

- 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>.
- 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.

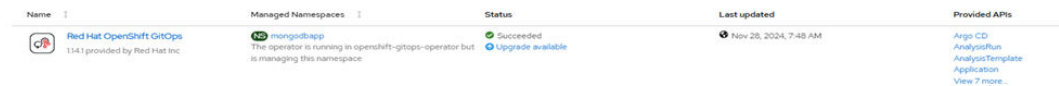
```
#oc get secret gitlab-gitlab-initial-root-password -
ojsonpath='{.data.password}' -n gitlab| base64 --decode ; echo
```

Red Hat OpenShift GitOps installation

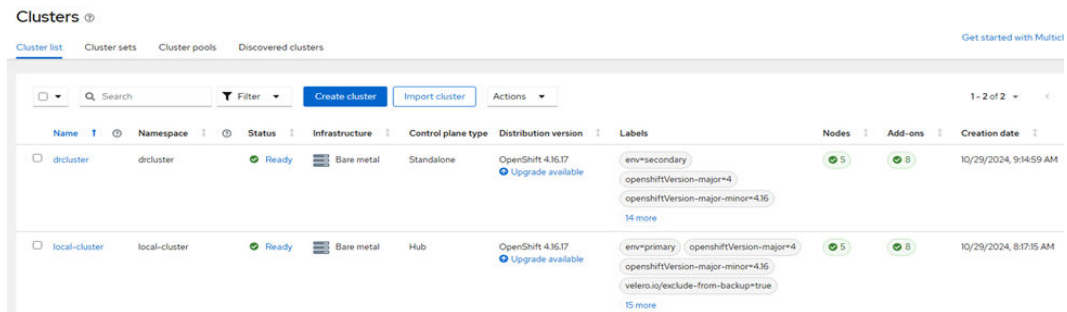
Install Red Hat OpenShift GitOps and integrated Red Hat Advanced Cluster Management using the Argo CD.

Procedure

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



- 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.



- Apply the following manifest files on the primary cluster.

ManagedClusterbinding	
<pre>apiVersion: cluster.open-cluster-management.io/v1beta2 kind: ManagedClusterSetBinding metadata: name: default namespace: openshift-gitops</pre>	<pre>apiVersion: cluster.open-cluster-management.io/v1beta2 kind: ManagedClusterSetBinding metadata: name: drclusterset namespace: openshift-gitops</pre>

spec: clusterSet: default	spec: clusterSet: drclusterset
Placement	
<pre> 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 </pre>	<pre> 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 </pre>
GitOpsCluster	
<pre> 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 </pre>	<pre> 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 </pre>

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**.

Create blank project
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
hrpcproject

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
https://gitlab.apps.primarycluster.juno.com/ root

Project slug
hrpcproject

Visibility Level

- Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.
- Internal
The project can be accessed by any logged in user except external users.
- Public
The project can be accessed without any authentication.

Project Configuration

- Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
- Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more.](#)

Create project Cancel

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.

Project access tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API. You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Active project access tokens 1 Add new token

Token name	Scopes	Created	Last Used	Expires	Role	Action
hrpcprojectoken	api_read_api, create_runner, manage_runner, k8s_proxy, read_repository, write_repository, read_registry, write_registry, ai_features	Nov 12, 2024	Never	in 2 weeks	Owner	

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.

Components

Replication Plug-in for Containers (HRPC)▼

Hitachi Replication Plug-in for Containers Product Code: ZIP-00HRPC001-05--UHRPCX-241000 MD5:4f41808c14f94e9109e0cf074ede0326	Versions 01.5.0 01-Oct-2024 23.8 MB File Type:zip	Download Share Link
---	--	--

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.


```

1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: hspc-replication-operator-storage-secrets
5   namespace: hspc-replication-operator-system
6 type: Opaque
7 stringData:
8   storage-secrets.yaml: |-
9     storages:
10    - serial: 810044
11      url: http://172.23.68.134
12      user: kubernetes
13      password: kubernetes
14      journal: 0
15    - serial: 810138
16      url: http://172.23.30.108
17      user: kubernetes
18      password: kubernetes
19      journal: 0
20

```

- 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.

Name	Last commit	Last update
CRDLocalvolume.yaml	Update CustomResourceDefinitionLocalvolume.yaml	1 week ago
CRDRemotevolume.yaml	Update CustomResourceDefinitionRemotevolume.yaml	1 week ago
CRDReplication.yaml	Add new file	1 week ago
README.md	Initial commit	1 week ago
clusterrole.yaml	Add new file	1 week ago
clusterrolebinding.yaml	Add new file	1 week ago
deployment.yaml	Update deployment.yaml	4 hours ago
role.yaml	Add new file	1 week ago
rolebinding.yaml	Add new file	1 week ago
sa.yaml	Service Account	1 week ago

- 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.

The screenshot shows the 'Create Policy' interface in the Red Hat Advanced Cluster Management console. The form is titled 'Create policy' and is set to 'YAML' mode. The 'Details' section shows the following information:

- Name:** namespacepolicy
- Description:** create hspc-replication-operator-system namespace with labelling in both the clusters.
- Namespace:** default
- Remediation:** Enforce
- Policy templates:** policy-namespace
- Placement:**
 - Name:** namespacepolicy-placement
 - Cluster sets:** global
- Policy annotations:**
 - Standards:** NIST SP 800-53
 - Categories:** CM Configuration Management

The 'Policy YAML' section shows the following code:

```

9 hspc-replication-operator-system namespace with addressing in c
10 clusters.
11 policy.open-cluster-management.io/categories: CM Configuration Ma
12 policy.open-cluster-management.io/standards: NIST SP 800-53
13 policy.open-cluster-management.io/controls: CM-2 Baseline Configu
14 spec:
15   remediationAction: enforce
16   policy-templates:
17     - objectDefinition:
18         apiVersion: policy.open-cluster-management.io/v1
19         kind: ConfigurationPolicy
20         metadata:
21           name: policy-namespace
22         spec:
23           remediationAction: enforce
24           severity: low
25         object-templates:
26           - complianceType: musthave
27             objectDefinition:
28               kind: Namespace
29               apiVersion: v1
30               metadata:
31                 name: hspc-replication-operator-system
32                 labels:
33                   control-plane: controller-manager
34
35 ---
36 apiVersion: cluster.open-cluster-management.io/v1beta1
37 kind: Placement
38 metadata:
39   name: namespacepolicy-placement
40   namespace: default
41 spec:
42   tolerations:
43     - key: cluster.open-cluster-management.io/unreachable
44       operator: Exists
  
```

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.

The screenshot shows the GitLab repository page for 'hrpcproject'. The page displays the repository name, a dropdown menu for branches (currently set to 'main'), and a 'Code' dropdown menu. The 'Code' dropdown menu is open, showing options to clone the repository with SSH or HTTPS. The SSH URL is `git@gitlab.apps.primarycluster.j` and the HTTPS URL is `https://gitlab.apps.primaryclust`.

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: aHJwY3Byb2plY3R0b2t1bG==
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.

Overview

Applications > hrpcprimarykubefconfigsecret

hrpcprimarykubefconfigsecret

Overview Topology

Details

Name hrpcprimarykubefconfigsecret

Type Subscription

Namespace hspc-replication-operator-system

Clusters 1 Remote

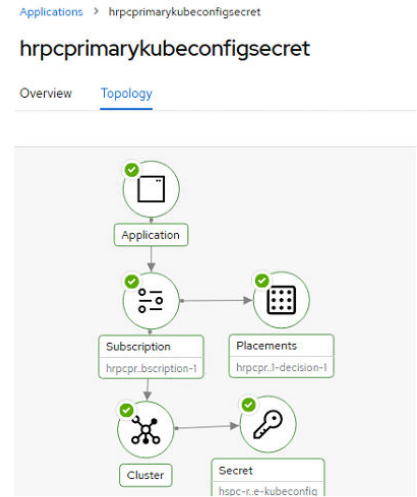
Repository Git <https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git> [Set time window](#)

Cluster resource status 1

Created 6:40 pm

Last sync requested [Sync](#)

Topology



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

Applications > Create application

Create application

YAML: On

Name secondarykubefconfigsecret

Namespace hspc-replication-operator-system

Repository location for resources

Repository types

Git

URL <https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git>

Username Optional: Enter the Git user name

Access token *****

Branch main

Path secondary-kubeconfig-secret.yaml

Select an existing placement configuration

Deploy application resources on clusters with all specified labels

Cluster sets

default x Select the cluster sets

Label	Operator	Value
Select the label	equals any of	Select the values

The application is created successfully.

Overview

Applications > secondarykubeconfigsecret

secondarykubeconfigsecret

Overview Topology

Details

Name secondarykubeconfigsecret

Type Subscription

Namespace hspc-replication-operator-system

Clusters Local

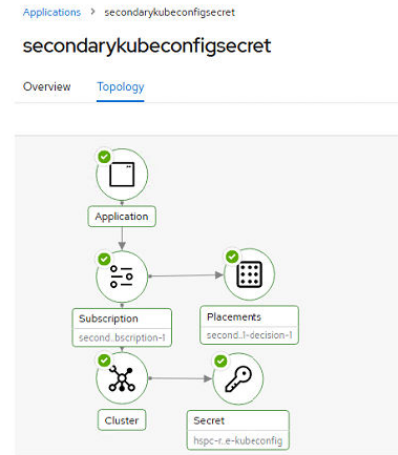
Repository Git <https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git> [Set time window](#)

Cluster resource status 1

Created Nov 30, 6:39 pm

Last sync requested - [Sync](#)

Topology



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

Applications > Create application

Create application YAML On

Name *

Namespace *

Repository location for resources

Repository types

Git

URL *

Username

Access token

Branch

Path

Select an existing placement configuration

Deploy application resources on clusters with all specified labels

Cluster sets * Select the cluster sets

Label	Operator	Value
Select the label	equals any of	Select the values

The application is created successfully.

Overview

Applications > storagesecrets

storagesecrets

Overview Topology

Details

Name storagesecrets

Type Subscription

Namespace hspc-replication-operator-system

Clusters 1 Remote, 1 Local

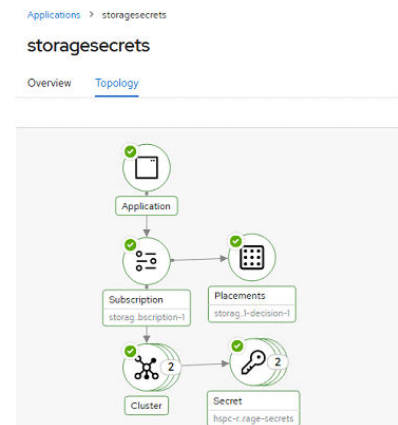
Repository Git <https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git> [Set time window](#)

Cluster resource status 1

Created Nov 30, 6:41 pm

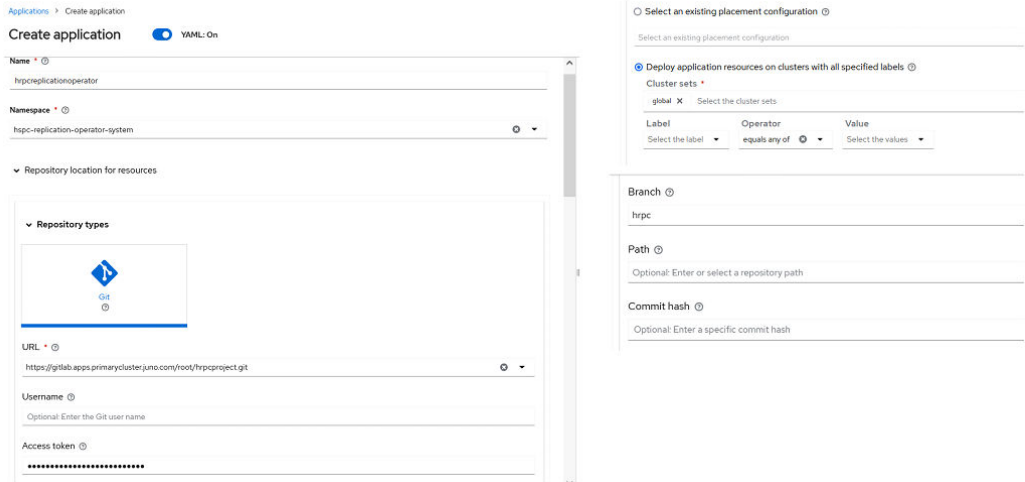
Last sync requested - [Sync](#)

Topology



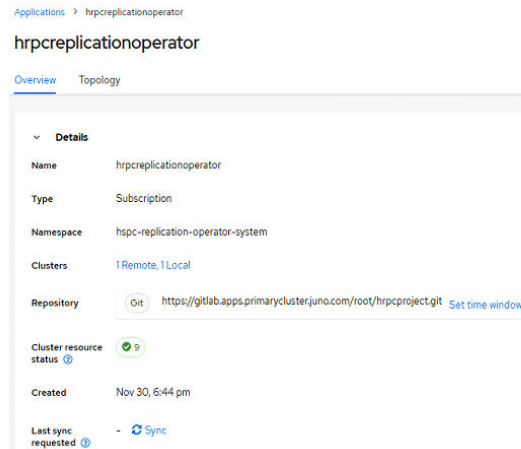
12. Create another application to deploy HSPC replication operator from the branch “hrpc” of “hrpcproject” in both clusters. Before creating the application, create a secret on both clusters to access the private registry in GitLab. This is required for the “deployment.yaml” manifest.

```
#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
```

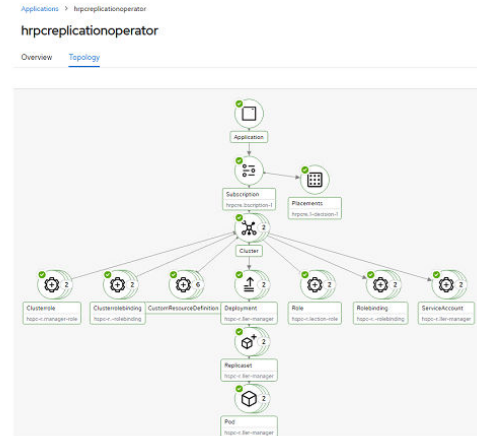


The HSPC replication operator is deployed successfully on both clusters.

Overview



Topology



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-lbvvhf 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.

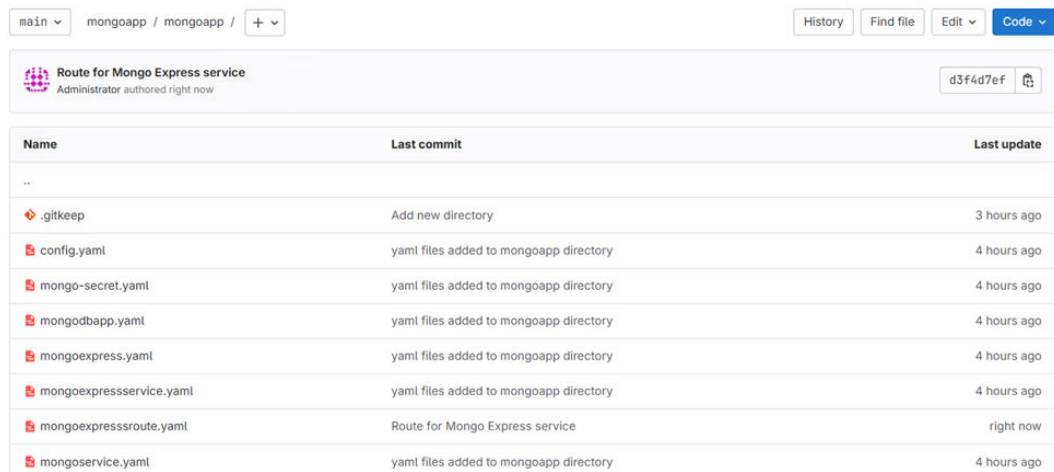
mongodbapp.yaml	mongoexpress.yaml
<pre> apiVersion: apps/v1 kind: StatefulSet metadata: name: mongodb spec: serviceName: mongodb-service replicas: 1 selector: matchLabels: app: mongodb template: metadata: labels: app: mongodb selector: mongodb spec: serviceAccountName: mongodb-sa terminationGracePeriodSeconds: 30 containers: - name: mongodb image: mongo ports: - name: mongo-port </pre>	<pre> 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: </pre>

<pre> 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 </pre>	<pre> - 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 </pre>
<p>mongoservice.yaml</p>	<p>mongoexpressservice.yaml</p>
<pre> apiVersion: v1 kind: Service metadata: name: mongodb-service spec: ports: - port: 27017 protocol: TCP targetPort: 27017 selector: app: mongodb clusterIP: None </pre>	<pre> apiVersion: v1 kind: Service metadata: name: mongo-express-service spec: selector: app: mongo-express type: ClusterIP ports: - protocol: TCP port: 8081 targetPort: 8081 </pre>
<p>mongo-secret.yaml</p>	<p>config.yaml</p>

<pre> apiVersion: v1 kind: Secret metadata: name: mongodb-secret type: Opaque data: mongo-root-username: dXNlcm5hbWU= mongo-root-password: cGFzc3dvcmQ= </pre>	<pre> apiVersion: v1 kind: ConfigMap metadata: name: mongodb-configmap data: database_url: mongodb-service </pre>
	<p>mongoexpressroute.yaml</p>
	<pre> 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 </pre>

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.

The screenshot shows the 'Create policy' interface in the Red Hat Advanced Cluster Management console. The interface is divided into several sections: Details, Templates, Placement, and Policy annotations. The 'Details' section shows the policy name 'mongoppnamespacepolicy', description 'Create a namespace mongodbapp on Primary and Secondary Cluster', namespace 'default', remediation 'Enforce', and policy templates 'policy-namespace-1'. The 'Placement' section shows the name 'mongoppnamespacepolicy-placement', cluster sets 'global', and policy annotations including standards 'NIST SP 800-53', categories 'CM Configuration Management', and controls 'CM-2 Baseline Configuration'. The 'Policy annotations' section shows the name 'mongoppnamespacepolicy-placement', cluster sets 'global', and policy annotations including standards 'NIST SP 800-53', categories 'CM Configuration Management', and controls 'CM-2 Baseline Configuration'. The 'Policy YAML' section shows the following code:

```

1 apiVersion: policy.open-cluster-management.io/v1
2 kind: Policy
3 metadata:
4   name: mongoppnamespacepolicy
5   namespace: default
6   annotations:
7     policy.open-cluster-management.io/description: Create a namespace mongodbapp on Primary and Secondary Cluster.
8     policy.open-cluster-management.io/categories: CM Configuration Management
9     policy.open-cluster-management.io/standards: NIST SP 800-53
10    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
11 spec:
12   disabled: false
13   remediationAction: enforce
14   policy-templates:
15     - objectDefinition:
16         apiVersion: policy.open-cluster-management.io/v1
17         kind: ConfigurationPolicy
18         metadata:
19           name: policy-namespace-1
20         spec:
21           remediationAction: enforce
22           severity: low
23           object-templates:
24             - complianceType: musthave
25               objectDefinition:
26                 kind: Namespace
27                 apiVersion: v1
28                 metadata:
29                   name: mongodbapp
30             -
31   apiVersions: cluster.open-cluster-management.io/v1beta1
32   kind: Placement
33   metadata:
34     name: mongoppnamespacepolicy-placement
35     namespace: default
36   spec:
37     tolerations:
38       - key: cluster.open-cluster-management.io/unreachable
39         operator: Exists
40       - key: cluster.open-cluster-management.io/unavailable
41         operator: Exists
42     clusterSets:
43       - global
44     -
45   apiVersion: policy.open-cluster-management.io/v1
46

```

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.

The screenshot shows the 'Create application' dropdown menu in the Red Hat Advanced Cluster Management console. The menu is open, showing the following options: 'Choose a type', 'Argo CD ApplicationSet - Pull model', 'Argo CD ApplicationSet - Push model', and 'Subscription'. The 'Argo CD ApplicationSet - Push model' option is selected.

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

Applications > Create application set - push model

Create application set - push model YAML

1 General

General

Name *
primarymongoapp

Argo server * ⓘ
openshift-gitops

Search for Argo server

- openshift-gitops
name: primaryargoserver; namespace: openshift-gitops
- openshift-gitops
name: primarygitopscluster; namespace: openshift-gitops
- openshift-gitops
name: secondarygitopscluster; namespace: openshift-gitops

- 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.

Applications > Create application set - push model

Create application set - push model YAML

2 Template

Type: Git | <https://gitlab.apps.primarycluster.juno.com/root/mongoapp.git>

URL * ⓘ
<https://gitlab.apps.primarycluster.juno.com/root/mongoapp.git>

Revision ⓘ
Enter or select a tracking revision

Path ⓘ
mongoapp

Destination

Remote namespace *
mongodbapp

Application set YAML

```

1 apiVersion: argoproj.io/v1alpha1
2 kind: Applicationset
3 metadata:
4   name: primarymongoapp
5   namespace: openshift-gitops
6 spec:
7   generators:
8     - clusterDecisionResource:
9       configMapRef: ace-placement
10      labelSelector:
11        matchLabels:
12          cluster.open-cluster-management.io/placement:
13            requestAfterSeconds: 180
14      template:
15        metadata:
16          name: primarymongoapp-[[name]]
17        labels:
18          velero.io/exclude-from-backup: "true"
19      specs:
20        project: default
21        sources:
22          - repositoryType: git
23            repoURL: https://gitlab.apps.primarycluster.juno
24            path: mongoapp
25        destination:
26          namespace: mongodbapp
27          server: "{{server}}"
28        syncPolicy:
29          automated:
30            selfHeal: true
31            prune: true
32          syncOptions:
33            - CreateNamespace=true
34            - PruneLast=true
35
36 apiVersion: cluster.open-cluster-management.io/v1beta1
37 kind: Placement
38 metadata:
39   name: primarymongoapp-placement
40   namespace: openshift-gitops

```

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

Applications > Create application set - push model

Create application set - push model YAML

4 Placement

Placement

How do you want to select clusters?
 New placement Existing placement

Existing placement

9. Review the settings and click Submit.

The screenshot shows the Argo CD 'Edit application set' interface for 'primarymongoapp'. The 'General' tab is active, displaying the following configuration:

- General:** Name: primarymongoapp, Argo server: openshift-gitops, Requeue time: 180.
- Repository:** Type: Git, URL: https://gitlab.apps.primarycluster.juno.com/root/mongoapp.git
- Destination:** Remote namespace: mongodapp
- Sync policy:** Automatically sync when cluster state changes (checked), Automatically create namespace if it does not exist (checked).
- Placement:** Existing placement: primarygtopplacement

The 'YAML' tab shows the application set configuration code, including the API version, kind, metadata, spec, and template sections.

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

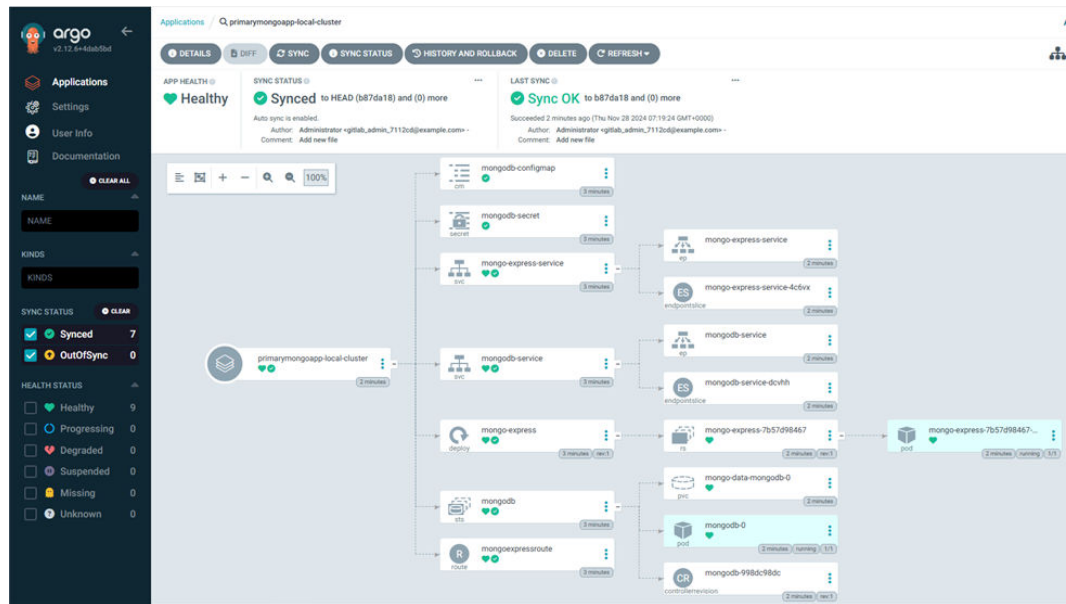
The screenshot shows two views of the 'primarymongoapp' in Argo CD:

- Overview:** Shows details for the application set, including Name, Type (Application set - Push model), Namespace (openshift-gitops), Clusters (Local), Repository (Git), and Cluster resource status (Healthy).
- Topology:** Shows a hierarchical diagram of the application set resources, including Configmap, Route, Deployment, Secret, Service, Statefulset, Replicaset, and Pod.

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

The screenshot shows the 'Details' view of the 'primarymongoapp' application set. The 'Launch Argo editor' button is visible in the top right corner. The 'Application deploy status' section shows the application is healthy and synced.

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



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

PV definition

PersistentVolumes > PersistentVolume details

[pvc-3099cbc0-876a-4fd7-ald9-17cc7b87dd76](#) Bound

Details YAML

PersistentVolume details

Name: pvc-3099cbc0-876a-4fd7-ald9-17cc7b87dd76

Status: Bound

Capacity: 40Gi

Access modes: ReadWriteOnce

Volume mode: Filesystem

StorageClass: ic-hpc

PersistentVolumeClaim: [mongo-data-mongodb-0](#)

Created at: Nov 28, 2024, 7:20 AM

Owner: No owner

Storage resources for PV

PersistentVolumes > PersistentVolume details

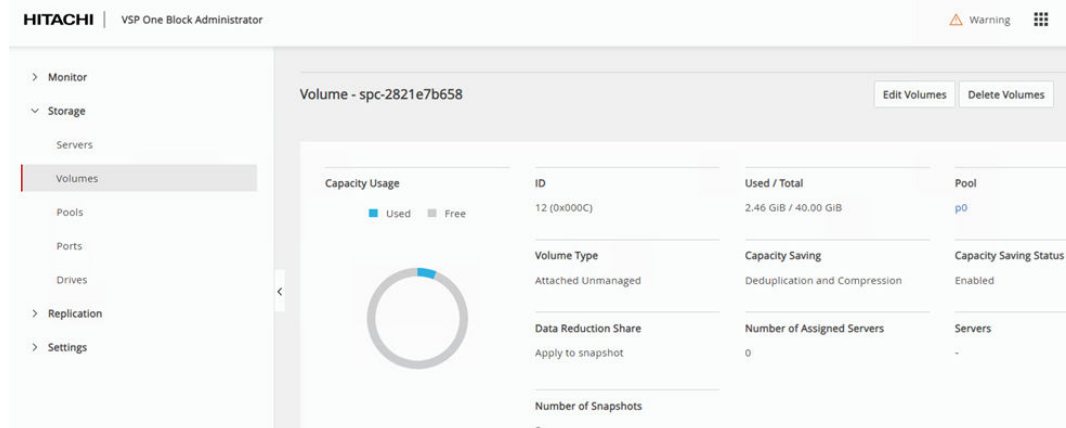
[PV pvc-3099cbc0-876a-4fd7-ald9-17cc7b87dd76](#) Bound

Details YAML

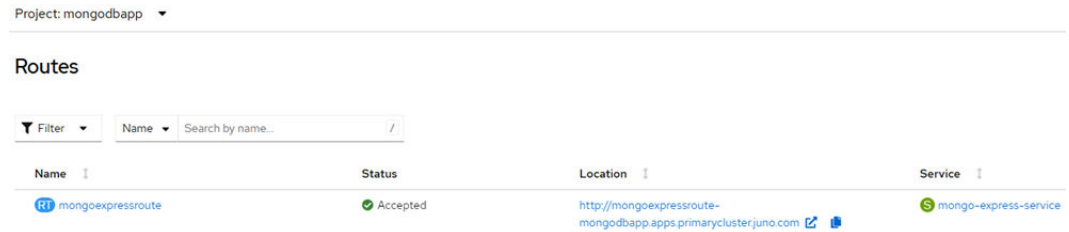
```

12  kind: PersistentVolume
13  metadata:
14    - external-attacher/hpccsi-hitachi.com
15  managedFields:
16  spec:
17    capacity:
18      storage: 40Gi
19    csi:
20      driver: hpccsi.hitachi.com
21      volumeHandle: 01--scsi--A34800010044--12--spc-2821e7b658
22    fsType: ext4
23    volumeAttributes:
24      hostNodeOption: ''
25    size: 40Gi
26    portName: ssc-2821e7b658
27    ports:
28      - CL3-C
29    idempotence: '00-00'
30    connectionType: iSCSI
31    storage.kubernetes.io/csiProvisionerIdentity: 1732760048959-3593-hpccsi
32    idempotence: '12'
    
```

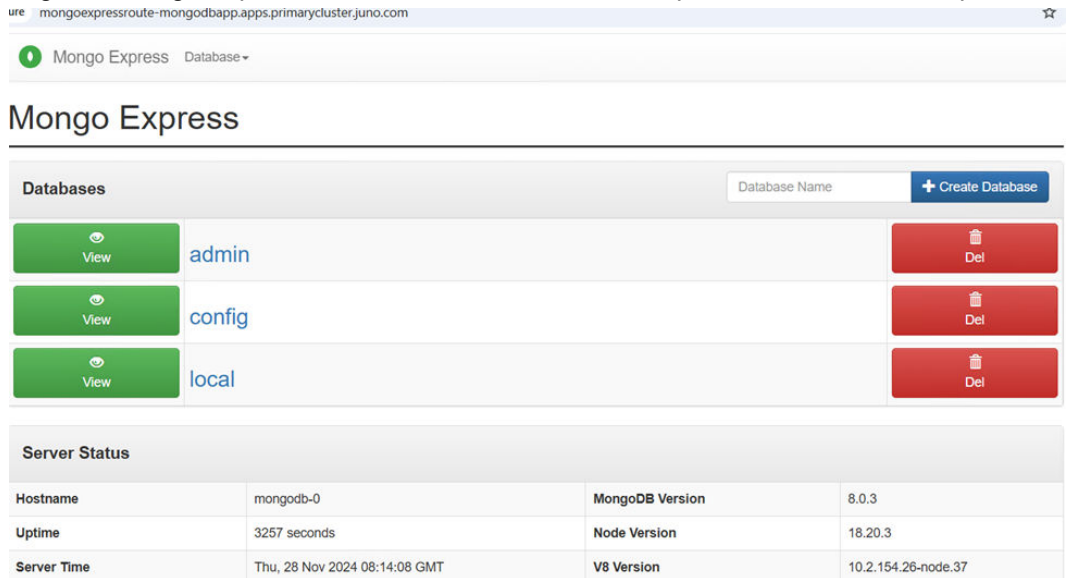
- For the PV, a DRS volume (00:0C) is created on the VSP One Block 24 storage system.



- 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.

Document1

Add Document

```

1 {
2   "_id": 1,
3   title: "Tombstone",
4   genres: [ "Western", "Drama" ],
5   runtime: 130,
6   rated: "R",
7   year: 1993,
8   directors: [ "George P. Cosmatos" ],
9   cast: [ "Kurt Russell", "Val Kilmer", "Sam Elliott" ],
10  type: "movie"
11 }
12 ]
    
```

Close Save

Document2

Add Document

```

1 {
2   "_id": 2,
3   title: "Titanic",
4   genres: [ "Western", "Romantic" ],
5   runtime: 194,
6   rated: "U",
7   year: 1997,
8   directors: [ "James Cameron" ],
9   cast: [ "Leonardo DiCaprio", "Kate Winslet", "Billy Zane" ],
10  type: "movie"
11 }
12 }
    
```

Close Save

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

Mongo Express Database: MovieDB -> Collection: MovieCollections

Viewing Collection: MovieCollections

Document added!

New Document New Index

Simple Advanced

Key Value String Find

Delete all 2 documents retrieved

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie

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 hrcproject 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.

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

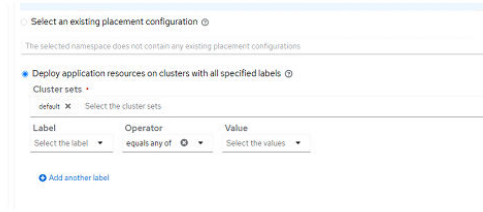
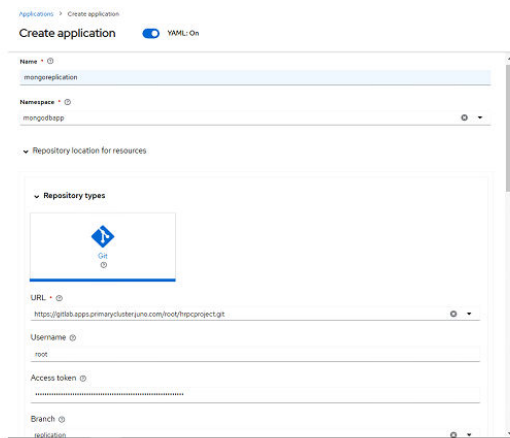
```
replication hrpcproject / hspc_v1_replication.yaml

[ci skip]
Administrator authored 1 week ago

hspc_v1_replication.yaml 179 B

1 apiVersion: hspc.hitachi.com/v1
2 kind: Replication
3 metadata:
4   name: mongodb-replication
5 spec:
6   persistentVolumeClaimName: mongo-data-mongodb-0
7   storageClassName: sc-hrpc
```

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

Applications > mongoreplication

mongoreplication

Overview Topology

Details

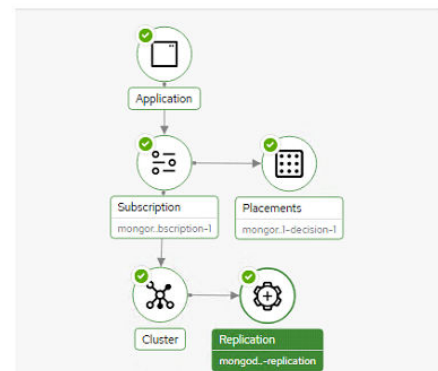
Name	mongoreplication
Type	Subscription
Namespace	mongodbapp
Clusters	Local
Repository	Git https://gitlab.apps.primarycluster.juno.com/root/hrpcproject.git
Cluster resource status	✔ 1
Created	4:20 am
Last sync requested	- Sync

Topology

Applications > mongoreplication

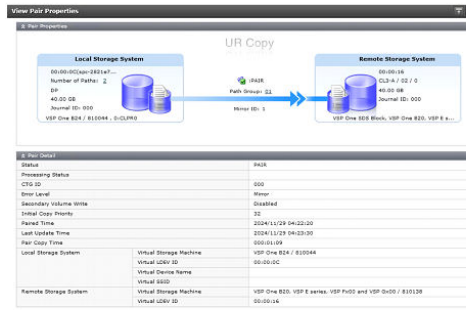
mongoreplication

Overview Topology

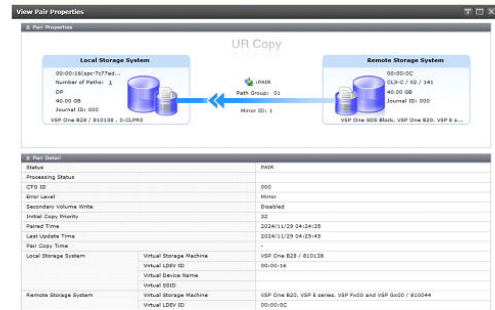


3. Verify replication status from Storage Navigator.

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

R mongodb-replication

Details [YAML](#)

```

1  apiVersion: hspc.hitachi.com/v1
2  kind: Replication
3  > metadata: ...
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: 818044
88     localStorageDeviceId: A34800818044
89     primaryVolumeId: 12
90     primaryVolumeNickname: spc-2821e7b658
91     remoteSerialNumber: 810138
92     remoteStorageDeviceId: A34800810138
93     secondaryVolumeId: 22
94     secondaryVolumeNickname: spc-7c77ed416e
95   status: Ready

```

Status from Secondary Cluster

Replications > Replication details

R mongodb-replication

Details [YAML](#)

```

66 namespace: mongodbapp
67 finalizers:
68   - hspc.csi.hitachi.com/replication-finalizer
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-3099cbc0-876a-4fd7-a1d9-17cc7b87dd76
81     replicationAttribute: secondary
82     storageClassName: sc-hrpc
83   status:
84     currentOperation:
85       operation: none
86       running: false
87     replicationPair:
88       localSerialNumber: 810138
89       localStorageDeviceId: A34800810138
90       primaryVolumeId: 12
91       primaryVolumeNickname: spc-2821e7b658
92       remoteSerialNumber: 810044
93       remoteStorageDeviceId: A34800818044
94       secondaryVolumeId: 22
95       secondaryVolumeNickname: spc-7c77ed416e
96     status: Ready

```

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

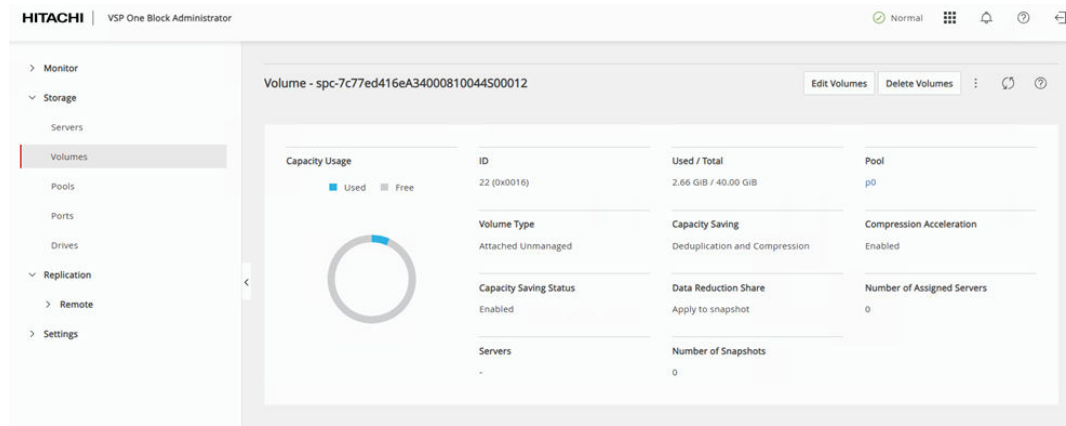
```

[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 ~]#

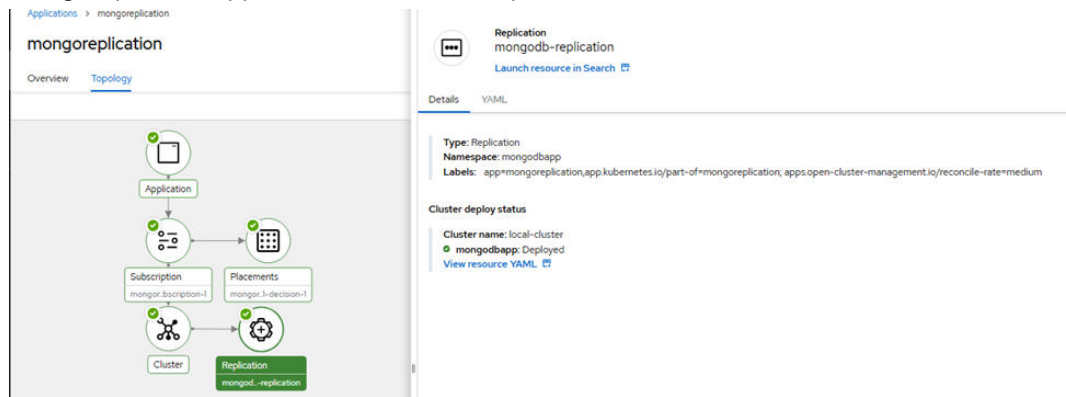
```

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.

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



- 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.



- 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

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-2821e7b658
91     remoteSerialNumber: 810138
92     remoteStorageDeviceId: A34000810138
93     secondaryVolumeId: 22
94     secondaryVolumeNickname: spc-7c77ed416e
95     status: Ready
  
```

Status after split

Search > replication details

mongodb-replication

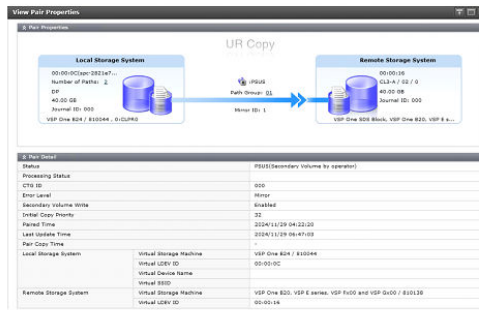
Details YAML Related resources

```

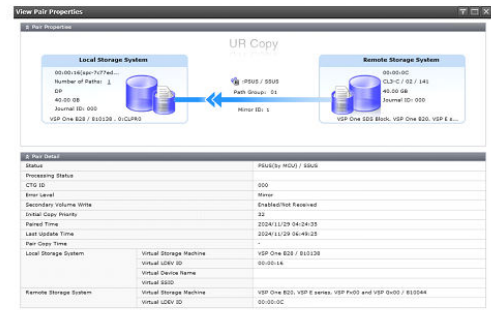
Cluster: local-cluster Namespace: mongodbapp
84 spec:
85   desiredPairState: split
86   persistentVolumeClaimName: mongo-data-mongodb-0
87   replicationAttribute: primary
88   storageClassName: sc-hrpc
89 status:
90   currentOperation:
91     operation: none
92     running: false
93   replicationPair:
94     localSerialNumber: 810044
95     localStorageDeviceId: A34000810044
96     primaryVolumeId: 12
97     primaryVolumeNickname: spc-2821e7b658
98     remoteSerialNumber: 810138
99     remoteStorageDeviceId: A34000810138
100    secondaryVolumeId: 22
101    secondaryVolumeNickname: spc-7c77ed416e
102    status: Split
  
```

- Verify the pair split status from Storage Navigator.

Status from VSP One Block 24

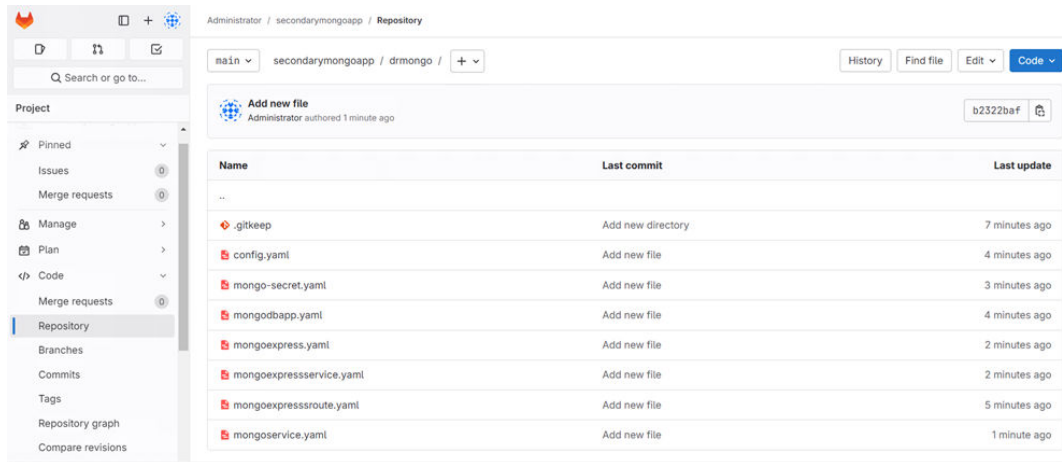


Status from VSP One Block 28



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.

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

Applications > Create application set - push model

Create application set - push model YAML

1 General
2 Template
3 Sync policy
4 Placement
5 Review

General

Name *
mongoappsecondary

Argo server *
Select the Argo server

openshift-gitops
name: secondarygitopscluster; namespace: openshift-gitops

openshift-gitops
name: primaryargoserver; namespace: openshift-gitops

openshift-gitops
name: primarygitopscluster; namespace: openshift-gitops

Add Argo Server

Application set YAML

```
1 apiVersion: argoproj.io/v1alpha1
2 kind: ApplicationSet
3 metadata:
4   name: mongoappsecondary
5   namespace: ""
6 spec:
7   generators:
8     - clusterDecisionResource:
9       configMapRef: acm:placement
10      labelSelector:
11        matchLabels:
12          cluster.open-cluster-management.io/placement: mongoappsecondary-placement
13      requestAfterSeconds: 180
14   template:
15     metadata:
16       name: mongoappsecondary-{{name}}
17     labels:
18       velero.io/exclude-from-backup: "true"
19   spec:
20     project: default
21     sources:
22     - {}
23     destination:
24       namespace: ""
25     server: "{{server}}"
26     syncPolicy:
27       automated:
28         selfFinal: true
```

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

Applications > Create application set - push model

Create application set - push model YAML

1 General
2 Template
3 Sync policy
4 Placement
5 Review

General

Name *
mongoappsecondary

Argo server *
Select the Argo server

openshift-gitops
name: secondarygitopscluster; namespace: openshift-gitops

openshift-gitops
name: primaryargoserver; namespace: openshift-gitops

openshift-gitops
name: primarygitopscluster; namespace: openshift-gitops

Add Argo Server

Application set YAML

```
1 apiVersion: argoproj.io/v1alpha1
2 kind: ApplicationSet
3 metadata:
4   name: mongoappsecondary
5   namespace: ""
6 spec:
7   generators:
8     - clusterDecisionResource:
9       configMapRef: acm:placement
10      labelSelector:
11        matchLabels:
12          cluster.open-cluster-management.io/placement: mongoappsecondary-placement
13      requestAfterSeconds: 180
14   template:
15     metadata:
16       name: mongoappsecondary-{{name}}
17     labels:
18       velero.io/exclude-from-backup: "true"
19   spec:
20     project: default
21     sources:
22     - {}
23     destination:
24       namespace: ""
25     server: "{{server}}"
26     syncPolicy:
27       automated:
28         selfFinal: true
```

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.

Applications > Create application set - push model

Create application set - push model YAML

1 General
2 Template
3 Sync policy
4 Placement
5 Review

Placement

How do you want to select clusters?

New placement Existing placement

Existing placement

secondarygitopsplacement

8. Review and click Submit.

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

Applications > Create application set - push model

Create application set - push model YAML

- 1 General
- 2 Template
- 3 Sync policy
- 4 Placement
- 5 Review

General

Name: mongoappsecondary

Argo server: openshift-gitops

Requeue time: 180

Repository

Type: Git

Git: <https://gitlab.apps.dircluster.juno.com/root/secondarymongoapp.git>

Destination

Remote namespace: mongodbbpp

Sync policy

- Automatically sync when cluster state changes
- Automatically create namespace if it does not exist

Placement

Existing placement: secondarygitopsplacement

Submit Back Cancel

Application set YAML

```
1 apiVersion: argoproj.io/v1alpha1
2 kind: ApplicationSet
3 metadata:
4   name: mongoappsecondary
5   namespace: openshift-gitops
6 spec:
7   generators:
8     - clusterDecisionResource:
9       configMapRef: acm-placement
10      labelSelector:
11        cluster: open-cluster-management.io/placement: secondarygitopsplacement
12      matchLabels:
13        requestAfterSeconds: 180
14   template:
15     metadata:
16       name: mongoappsecondary-{{name}}
17     labels:
18       velero.io/exclude-from-backup: "true"
19     spec:
20       project: default
21       source:
22         - repositoryType: git
23           repoURL: https://gitlab.apps.dircluster.juno.com/root/secondarymongoapp.git
24           path: dr/mongo
25       destination:
26         namespace: mongodbbpp
27         server: "(server)"
28       syncPolicy:
29         automated:
30           selfHeal: true
31           prune: false
32         syncOptions:
33           - CreateNamespace=true
34           - Prune=false
```

The application is created.

Overview

Applications > mongoappsecondary

mongoappsecondary

Overview Topology

Details

Name: mongoappsecondary

Type: Application set - Push model Argo

Namespace: openshift-gitops

Clusters: 1 Remote

Repository: Git <https://gitlab.apps.dircluster.juno.com/root/secondarymongoapp.git>

Cluster resource status: 8

Created: 8:43 am

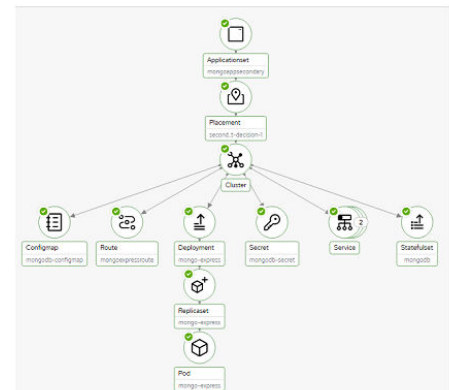
Last reconciled: 9:08 am

Topology

Applications > mongoappsecondary

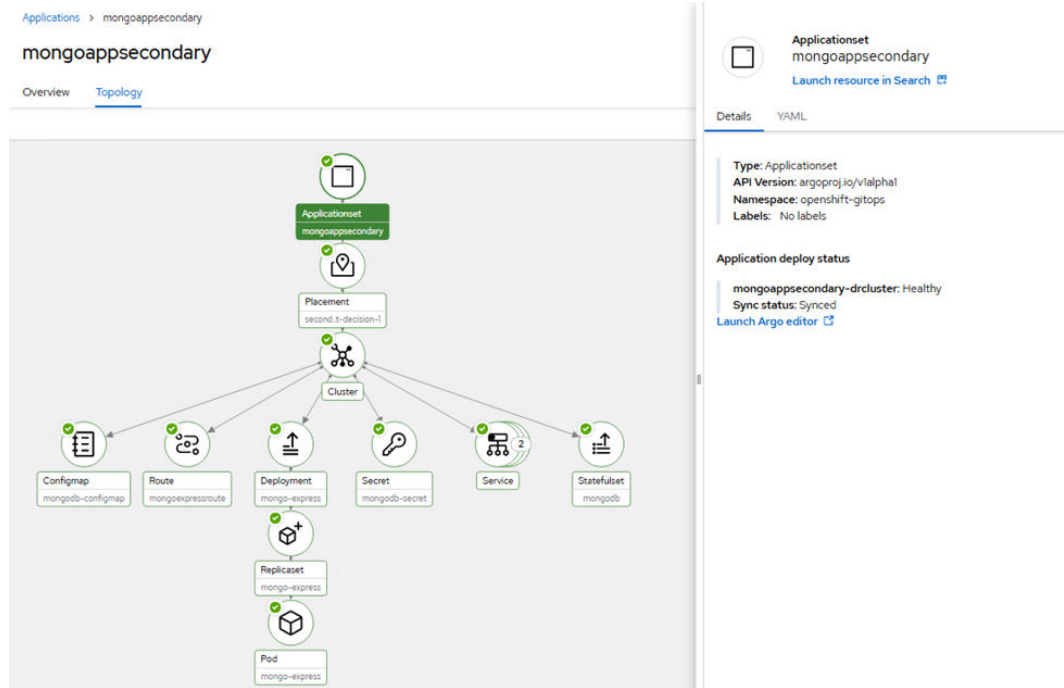
mongoappsecondary

Overview Topology

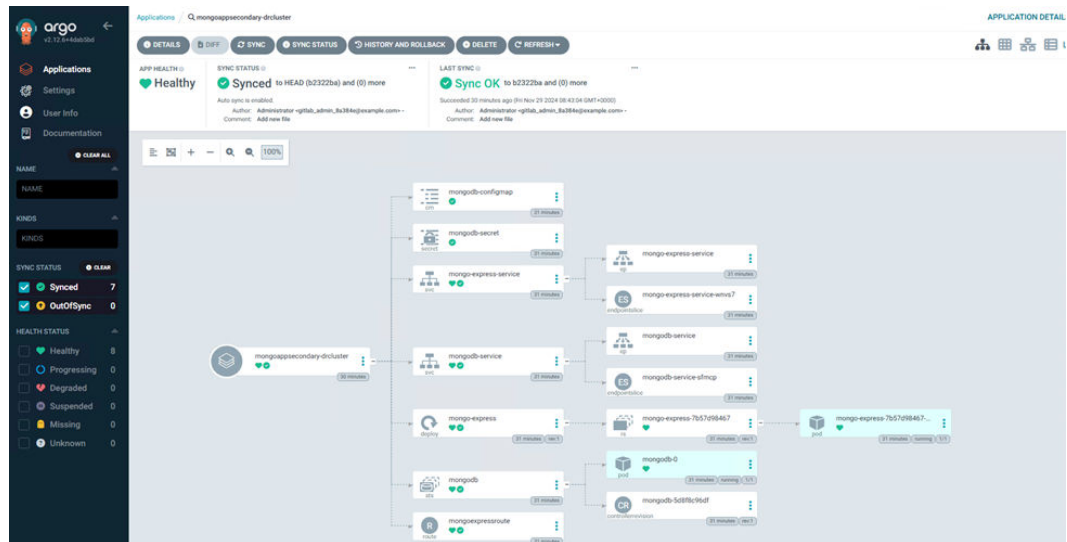


9. Click the Application set and select Launch Argo editor in the details section. This will open the application in Argo CD.

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



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.

Project: mongodbapp

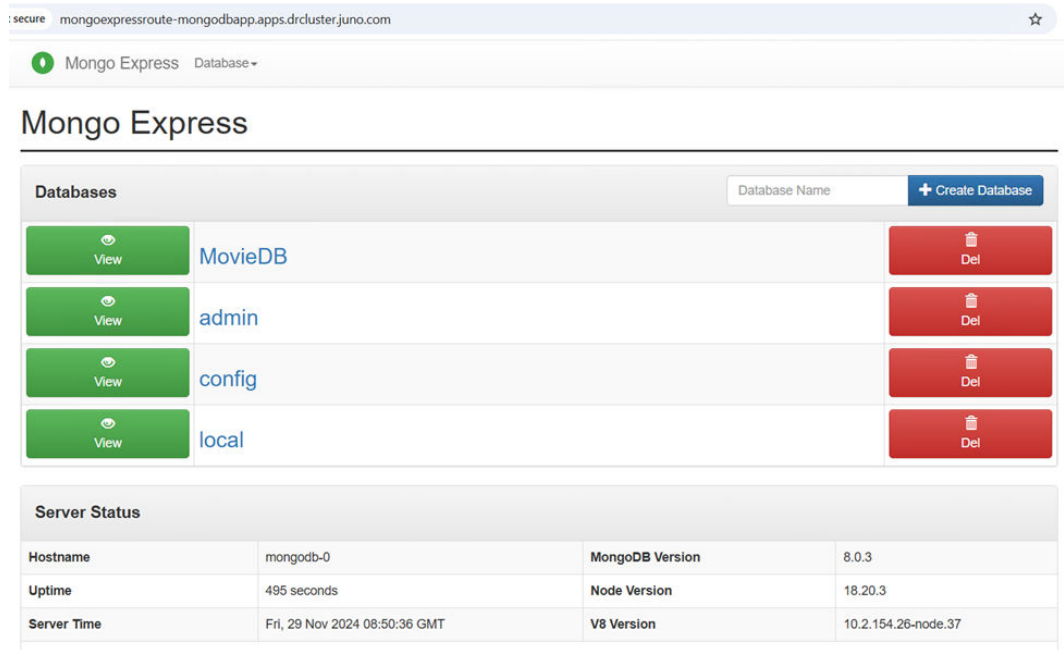
Routes

Filter: Name Search by name...

Name	Status	Location	Service
mongoexpressroute	Accepted	http://mongoexpressroute-mongodbapp.apps.dcluster.juno.com	mongo-express-service

12. Verify that the database MovieDB is available.

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



Mongo Express

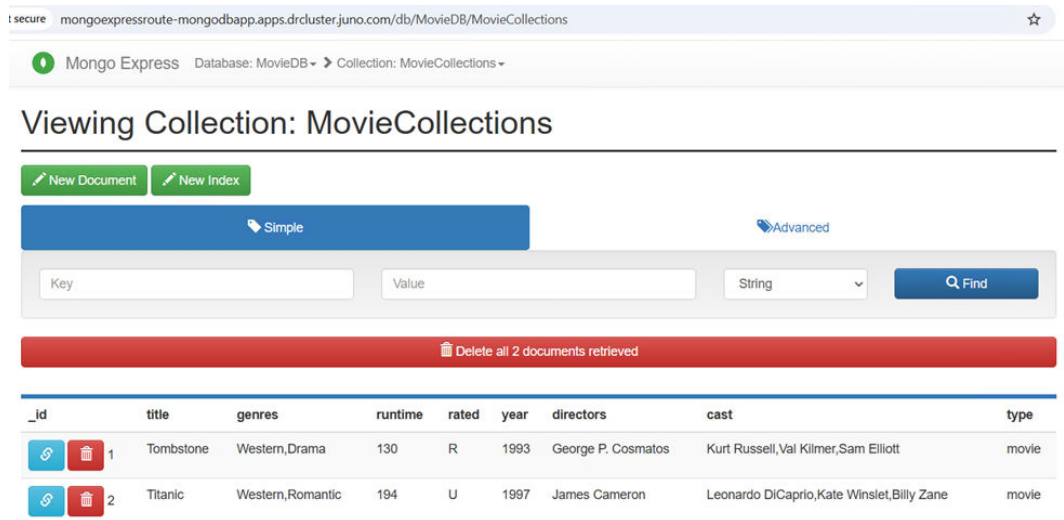
Databases

Database Name	View	Del
MovieDB	View	Del
admin	View	Del
config	View	Del
local	View	Del

Server Status

Property	Value	Property	Value
Hostname	mongodb-0	MongoDB Version	8.0.3
Uptime	495 seconds	Node Version	18.20.3
Server Time	Fri, 29 Nov 2024 08:50:36 GMT	V8 Version	10.2.154.26-node.37

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



Mongo Express Database: MovieDB - Collection: MovieCollections -

Viewing Collection: MovieCollections

New Document New Index

Simple Advanced

Key Value String Find

Delete all 2 documents retrieved

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie

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

x

Add Document

```

1 {
2   "_id": 3,
3   "title": "The Matrix",
4   "genres": [ "Western", "Sci-fi" ],
5   "runtime": 136,
6   "rated": "U",
7   "year": 1999,
8   "directors": [ "Lana Wachowskin", "Lilly Wachowski" ],
9   "cast": [ "Keanu Reeves", "Laurence Fishburne", "Carrie-Anne Moss" ]
10  "type": "movie"
11 }
12 }

```

Close
Save

15. Verify that the new document is added.

secure mongoexpressroute-mongodbbapp.apps.drcluster.juno.com/db/MovieDB/MovieCollections

● Mongo Express Database: MovieDB -> Collection: MovieCollections

Viewing Collection: MovieCollections

Document added! x

New Document
New Index

Simple
Advanced

String
Find

🗑️ Delete all 3 documents retrieved

_id	title	genres	runtime	rated	year	directors	cast	type
🔗 🗑️ 1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
🔗 🗑️ 2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie
🔗 🗑️ 3	The Matrix	Western,Sci-fi	136	U	1999	Lana Wachowskin,Lilly Wachowski	Keanu Reeves,Laurence Fishburne,Carrie-Anne Moss	movie

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.

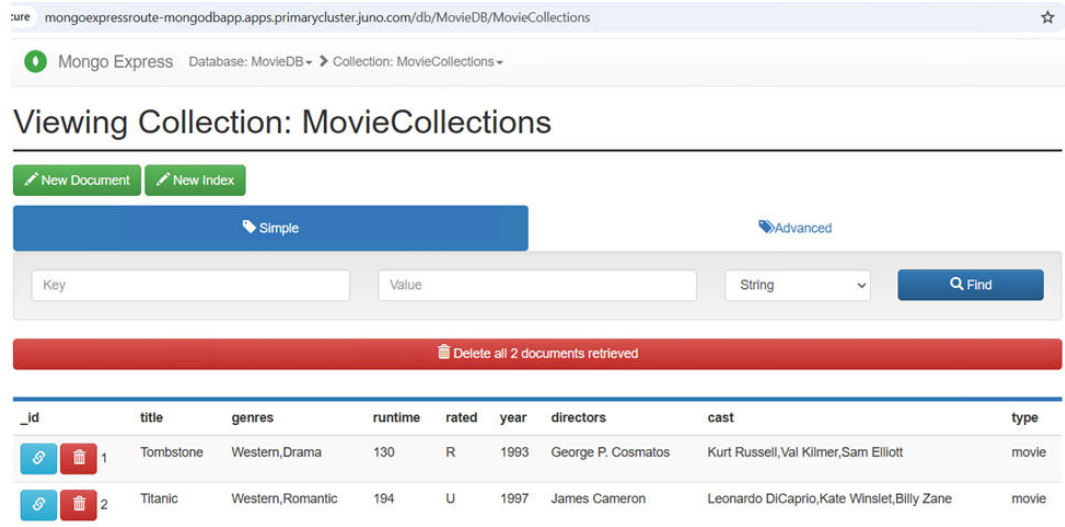
Failover operation

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.



The screenshot shows the Mongo Express interface for the `MovieDB` database and `MovieCollections` collection. It features a search bar with 'Key' and 'Value' fields, a 'String' dropdown, and a 'Find' button. A red notification bar indicates that 2 documents were deleted. Below is a table of movie entries:

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie

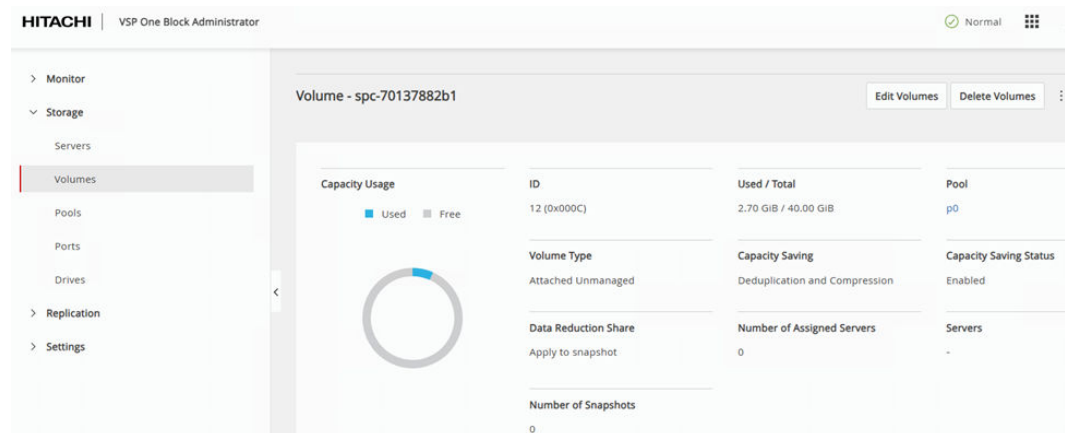
2. For the stateful MongoDB application, a PVC is created on the primary cluster.




The screenshot shows the PersistentVolumeClaims page in the Kubernetes dashboard. A single PVC is listed:

Name	Status	PersistentVolumes	Capacity	Used	StorageClass
mongo-data-mongodb-0	Bound	pvc-96a294b4-acee-48b4-91a4-a78c1eac3071	40 GiB	-	cc-hpc

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



The screenshot shows the HITACHI VSP One Block Administrator interface. The volume details for 'Volume - spc-70137882b1' are as follows:

Capacity Usage	ID	Used / Total	Pool
	12 (0x000C)	2.70 GiB / 40.00 GiB	p0
	Volume Type	Capacity Saving	Capacity Saving Status
	Attached Unmanaged	Deduplication and Compression	Enabled
	Data Reduction Share	Number of Assigned Servers	Servers
	Apply to snapshot	0	-
	Number of Snapshots	0	

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

Local Storage System		Remote Storage System	
Virtual Storage Machine	VSP One B24 / B1004	Virtual Storage Machine	VSP One B28 Block, VSP One B22, VSP One B24
Virtual LDEV ID	00-00-0C	Virtual LDEV ID	00-00-0E
Virtual Storage Name		Virtual Storage Name	
Virtual SSD		Virtual SSD	

UR Copy	
Status	PAUS
Processing Status	
CTID ID	000
Copy Level	Normal
Secondary Volume Write	Disabled
Initial Copy Priority	32
Repeat Time	2024/12/11 18:45:34
Last Update Time	2024/12/11 18:50:45
Pair Copy Time	00:00:00

Status from VSP One Block 28

Local Storage System		Remote Storage System	
Virtual Storage Machine	VSP One B28 / B1013	Virtual Storage Machine	VSP One B22 Block, VSP One B22, VSP One B24
Virtual LDEV ID	00-00-14	Virtual LDEV ID	00-00-0C
Virtual Storage Name		Virtual Storage Name	
Virtual SSD		Virtual SSD	

UR Copy	
Status	PAUS
Processing Status	
CTID ID	000
Copy Level	Normal
Secondary Volume Write	Disabled
Initial Copy Priority	32
Repeat Time	2024/12/11 18:51:06
Last Update Time	2024/12/11 18:53:07
Pair Copy Time	

6. 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-hrhc		
82	status:		
83	currentOperation:		
84	operation: none		
85	running: false		
86	replicationPair:		
87	localSerialNumber: 810044		
88	localStorageDeviceId: A34000810044		
89	primaryVolumeId: 12		
90	primaryVolumeNickname: spc-70137882b1		
91	remoteSerialNumber: 810138		
92	remoteStorageDeviceId: A34000810138		
93	secondaryVolumeId: 30		
94	secondaryVolumeNickname: spc-c420116eb7		
95	status: Ready		

Status from Secondary Cluster

Replications > Replication details

R 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-hrhc
79     volumeMode: Filesystem
80     volumeName: pvc-9ba29484-aeaa-48bf-91a4-a78c1dec3071
81   replicationAttribute: secondary
82   storageClassName: sc-hrhc
83 status:
84   currentOperation:
85     operation: none
86     running: false
87   replicationPair:
88     localSerialNumber: 810138
89     localStorageDeviceId: A34000810138
90     primaryVolumeId: 12
91     primaryVolumeNickname: spc-70137882b1
92     remoteSerialNumber: 810044
93     remoteStorageDeviceId: A34000810044
94     secondaryVolumeId: 30
95     secondaryVolumeNickname: spc-c420116eb7
96   status: Ready

```

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

```

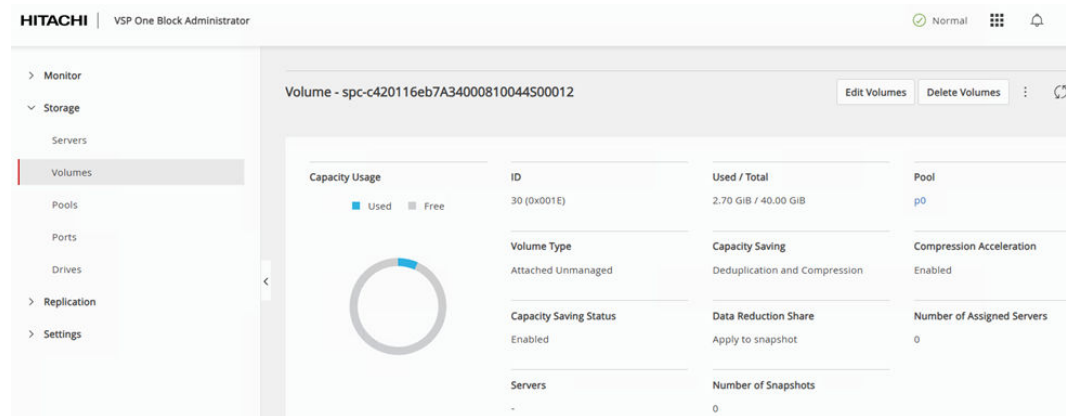
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_P} oc get replication -n mongodbapp
NAME                STATUS  DESIREDSTATE  OPERATION  AGE
mongodb-replication  Ready  none          none       128m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get replication -n mongodbapp
NAME                STATUS  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 ~]#

```

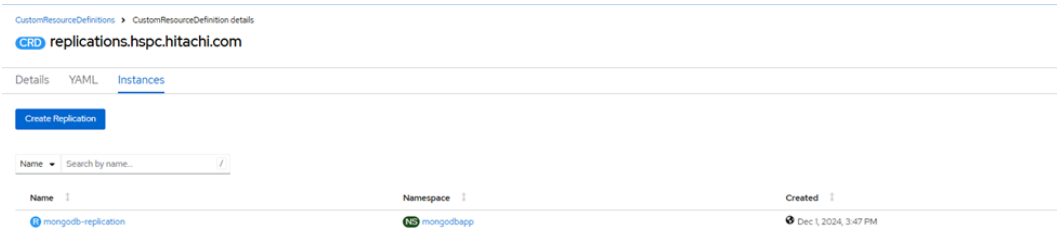
Hitachi Replication Plug-in for Containers in combination with Hitachi Storage plug-in for Containers creates a DRS persistent volume (LDEV ID 00:1E) on the secondary cluster.



- 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**.



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



- 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
R 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-hrhc
79     volumeMode: Filesystem
80     volumeName: pvc-9ba29484-aeaa-48bf-91a4-a78c1dec3071
81   replicationAttribute: secondary
82   storageClassName: sc-hrhc
83 status:
84   currentOperation:
85     operation: none
86     running: false
87   replicationPair:
88     localSerialNumber: 810138
89     localStorageDeviceId: A34000810138
90     primaryVolumeId: 12
91     primaryVolumeNickname: spc-70137882b1
92     remoteSerialNumber: 810044
93     remoteStorageDeviceId: A34000810044
94     secondaryVolumeId: 30
95     secondaryVolumeNickname: spc-c420116eb7
96   status: Ready

```

After Failover

Replications > Replication details
R mongodb-replication

Details [YAML](#)

```

11 > managedFields:---
12 > namespace: mongodbapp
13 > finalizers:
14 > - hspc.csi.hitachi.com/replication-finalizer
15 > spec:
16 > desiredPairState: failover
17 > persistentVolumeClaimName: mongo-data-mongodb-0
18 > persistentVolumeClaimSpec:
19 > accessModes:
20 > - ReadWriteOnce
21 > resources:
22 > requests:
23 > storage: 40Gi
24 > storageClassName: sc-hrhc
25 > volumeMode: Filesystem
26 > volumeName: pvc-9ba29484-aeaa-48bf-91a4-a78c1dec3071
27 > replicationAttribute: secondary
28 > storageClassName: sc-hrhc
29 > status:
30 > conditions:
31 > - lastTransitionTime: '2024-12-01T19:20:56Z'
32 > message: replication pair has failed over
33 > reason: replicationPairFailover
34 > status: 'True'
35 > type: Failover
36 > currentOperation:
37 > operation: none
38 > running: false
39 > replicationPair:
40 > localSerialNumber: 810138
41 > localStorageDeviceId: A34000810138
42 > primaryVolumeId: 12
43 > primaryVolumeNickname: spc-70137882b1
44 > remoteSerialNumber: 810044
45 > remoteStorageDeviceId: A34000810044
46 > secondaryVolumeId: 30
47 > secondaryVolumeNickname: spc-c420116eb7
48 > status: Failover

```

11. Verify the replication status from Storage Navigator.

Status from VSP One Block 24

View Pair Properties

Local Storage System: 00-00-0C (Cap=751378...), Number of Paths: 2, OP: 40-00-08, Journal ID: 000, VSP One 824 / 810044 - 01C1P00

Remote Storage System: 00-00-1E, Path Group: 01, VSP One 820 Block, VSP One 822, VSP One 824

UR Copy: Mirror, Path Group: 01, Mirror ID: 1

Pair Data	
Status	PDUG(Secondary volume by operator)
Processing Status	000
CTS ID	000
Error Level	Minor
Secondary Volume Write	Enabled
Initial Copy Priority	32
Panel Time	2024/12/01 18:03:04
Last Update Time	2024/12/01 19:21:02
Pair Copy Time	-
Local Storage System	Virtual Storage Machine: VSP One 824 / 810044
	Virtual LDEV ID: 00-00-0C
	Virtual Storage Name: VSP One 824
	Virtual SSD: 00-00-08
Remote Storage System	Virtual Storage Machine: VSP One 820, VSP One 822, VSP One 824 and VSP One 820 / 810138
	Virtual LDEV ID: 00-00-1E

Status from VSP One Block 28

View Pair Properties

Local Storage System: 00-00-1E (Cap=420116...), Number of Paths: 1, OP: 40-00-08, Journal ID: 000, VSP One 820 / 810138 - 01C1P00

Remote Storage System: 00-00-0C, Path Group: 01, VSP One 820 Block, VSP One 822, VSP One 824

UR Copy: Mirror, Path Group: 01, Mirror ID: 1

Pair Data	
Status	PDUG(Secondary volume by operator) / 0200
Processing Status	000
CTS ID	000
Error Level	Minor
Secondary Volume Write	Enabled/Not Respected
Initial Copy Priority	32
Panel Time	2024/12/01 18:03:04
Last Update Time	2024/12/01 19:22:04
Pair Copy Time	-
Local Storage System	Virtual Storage Machine: VSP One 820 / 810138
	Virtual LDEV ID: 00-00-1E
	Virtual Storage Name: VSP One 820
	Virtual SSD: 00-00-08
Remote Storage System	Virtual Storage Machine: VSP One 820, VSP One 822, VSP One 824 and VSP One 820 / 810044
	Virtual LDEV ID: 00-00-0C

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  Fallover  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

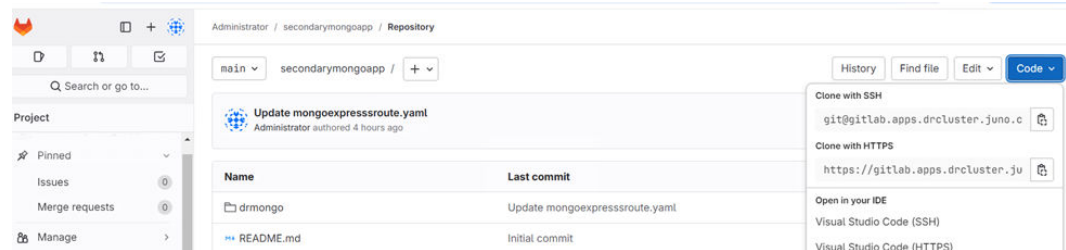
NAME                READY  AGE
remotevolume.hspc.hitachi.com/mongodb-replication  true  3h34m
[root@occlient ~]# KUBECONFIG=${KUBECONFIG_S} oc get localvolume,remotevolume -n mongodbapp
NAME                READY  AGE
localvolume.hspc.hitachi.com/mongodb-replication  true  3h34m

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

```

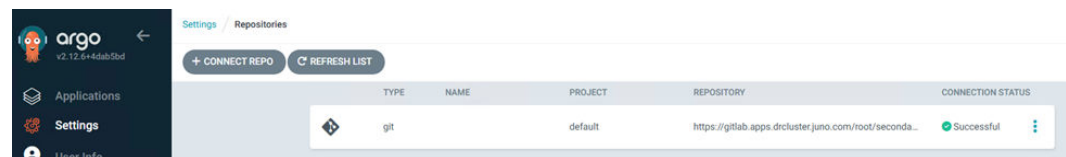
In a failover scenario assuming that the primary site is inaccessible, the secondary Red Hat OpenShift GitOps Argo CD instance is used to deploy the MongoDB application with the replicated PVC on the secondary cluster.

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

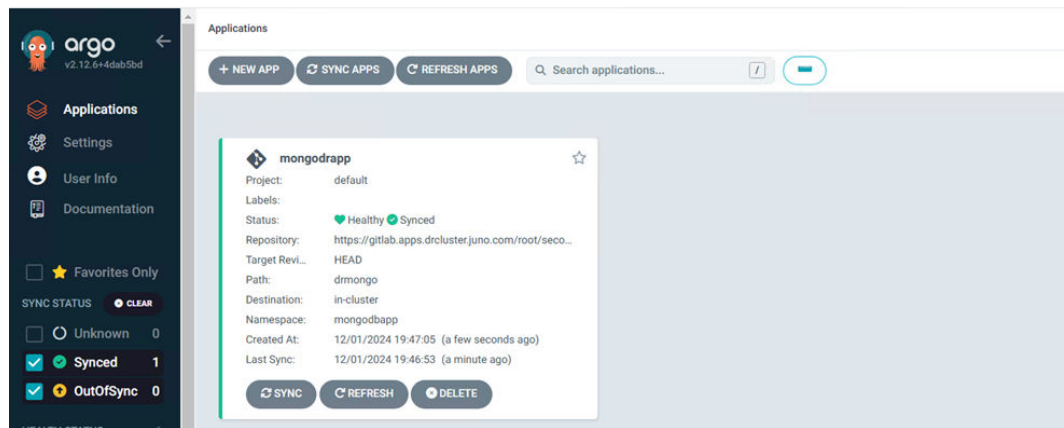


14. In the secondary Red Hat GitOps Argo CD instance, click **Settings**, then click **+CONNECT REPO**.
15. Select “VIA HTTPS”, type “git”, Project “default”, enter the copied URL in Repository URL, username as “root” and password as GitLab root password. Click on the check box Skip server verification.

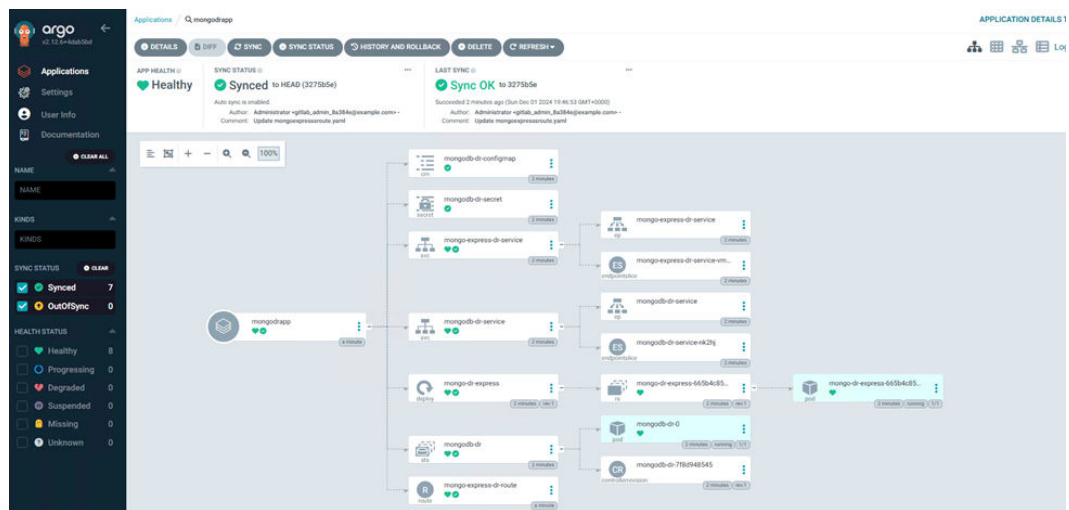
The repository is added.



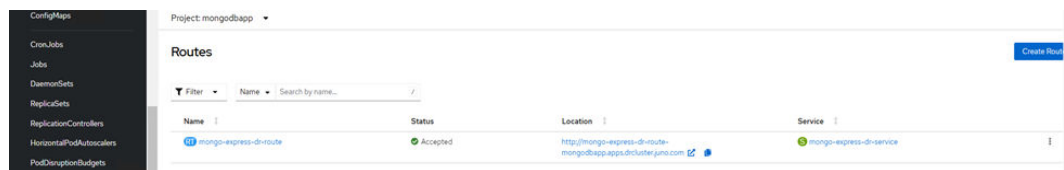
16. Change the **mongodbapp.yaml** file with the PVC name instead of **volumeClaimTemplates** as described.
17. From the Red Hat OpenShift GitOps Argo CD console, click **Applications**, then **+NEW APP**.
18. Create a new application with the following information and click **Create**.
 - a. Application 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.

Mongo Express Database: MovieDB > Collection: MovieCollections

Viewing Collection: MovieCollections

[New Document](#) [New Index](#)

Simple [Advanced](#)

Key Value String [Find](#)

Delete all 2 documents retrieved

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie

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

Add Document

```

1 {
2   "_id": 3,
3   title: "The Matrix",
4   genres: [ "Western", "Sci-fi" ],
5   runtime: 136,
6   rated: "U",
7   year: 1999,
8   directors: [ "Lana Wachowskin", "Lilly Wachowski" ],
9   cast: [ "Keanu Reeves", "Laurence Fishburne", "Carrie-Anne Moss" ]
10  type: "movie"
11 }
12 }

```

[Close](#) [Save](#)

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

Mongo Express Database: MovieDB - Collection: MovieCollections -

Document added!

New Document New Index

Simple Advanced

Key Value String Find

Delete all 3 documents retrieved

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Vai Klimer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie
3	The Matrix	Western,Sci-fi	136	U	1999	Lana Wachowskin,Lilly Wachowski	Keanu Reeves,Laurence Fishburne,Carrie-Anne Moss	movie

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

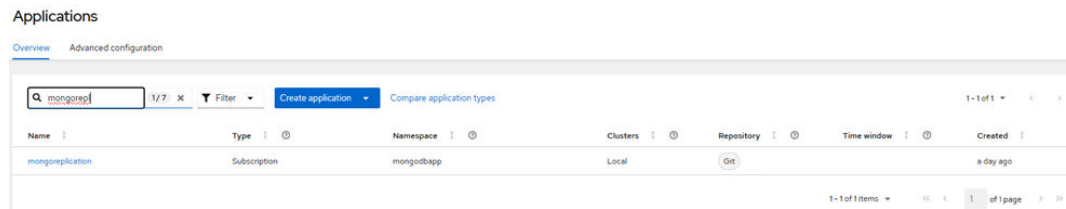
1. Clean up the primary site.
2. Create the replication CR in the secondary site.
3. Perform failover from the primary site.
 - a. Edit the replication CR in the primary site and change spec.desiredPairState to **failover**.
 - b. Confirm that replication CR status is **failover**, and the operation 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

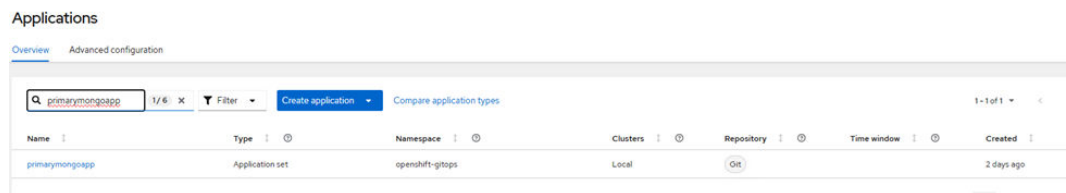
1. Delete the application **mongoreplication** created on the primary cluster from Red Hat Advanced Cluster Management by clicking the 3 dots, and then click **Delete Application**.



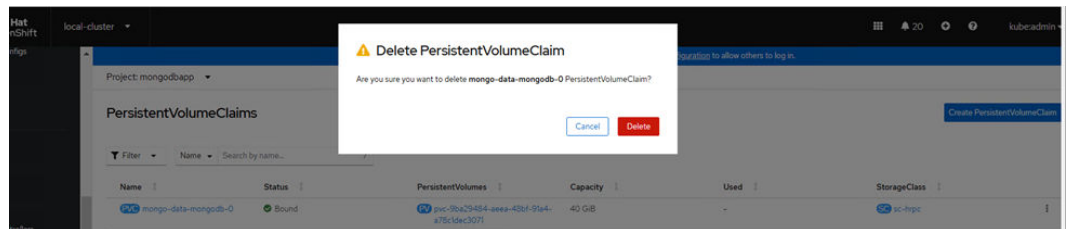
2. Confirm that all resources for **replication**, **localvolume** and **remotevolume** are deleted.

```
[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.

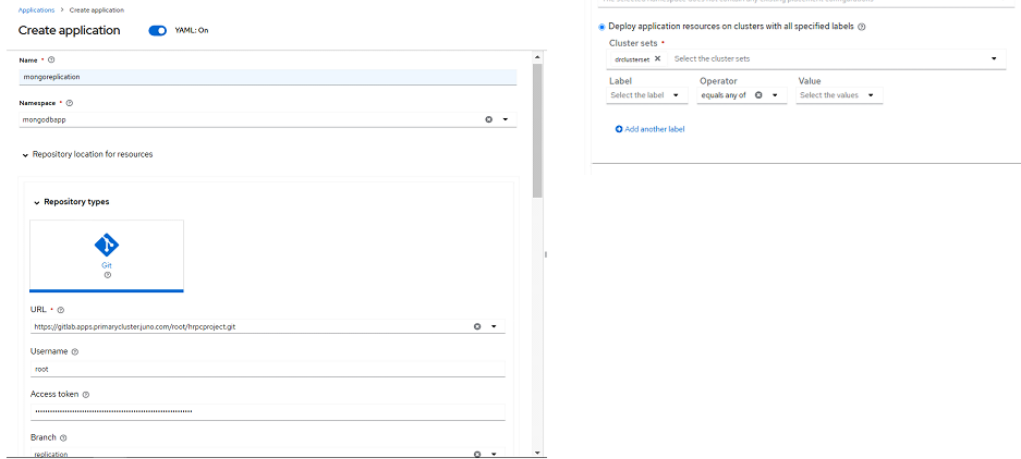


Create the replication CR in the secondary site

Procedure

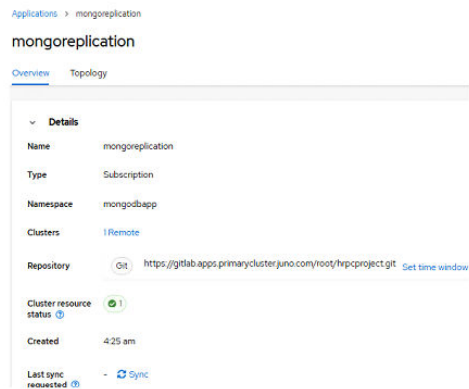
1. Create a subscription-based application to replicate the MongoDB PVC from the VSP One Block 28 to the VSP One Block 24 storage system. Here the application name is **mongoreplication**, **Namespace** is **mongodbapp**, select **Git**, add the URL of **hrpcproject**, and select **Branch** as **replication**,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

Create the replication CR in the secondary site

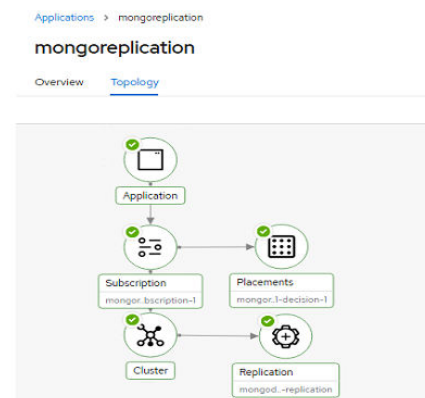


The application is created.

Overview

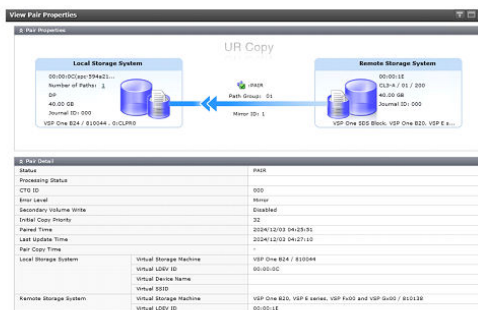


Topology

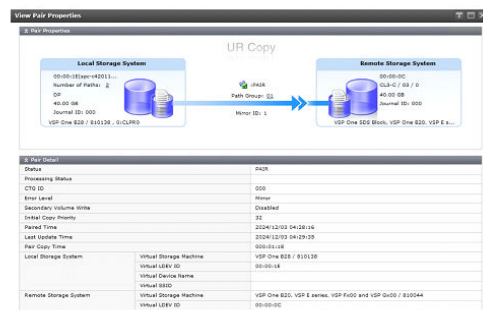


2. Verify the replication status from Storage Navigator.

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
R 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-hrhc
79     volumeMode: Filesystem
80     volumeName: pvc-292b9b0c-4d19-4810-b805-28cc20805cd4
81     replicationAttribute: secondary
82     storageClassName: sc-hrhc
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-hrhc
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

```

Replications > Replication details
R 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-hrhc
79     volumeMode: Filesystem
80     volumeName: pvc-292b9b0c-4d19-481b-b805-20cc28885cd4
81     replicationAttribute: secondary
82     storageClassName: sc-hrhc
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
    
```

After Failover

```

Replications > Replication details
R mongodb-replication

Details  YAML

11 > managedFields: --
74 namespace: mongodbapp
75 finalizers:
76   - hpc.csi.hitachi.com/replication-finalizer
77 spec:
78   desiredPairState: failover
79   persistentVolumeClaimName: mongo-data-mongodb-0
80   persistentVolumeClaimSpec:
81     accessModes:
82       - ReadWriteOnce
83     resources:
84       requests:
85         storage: 40Gi
86     storageClassName: sc-hrhc
87     volumeMode: Filesystem
88     volumeName: pvc-292b9b0c-4d19-481b-b805-20cc28885cd4
89     replicationAttribute: secondary
90     storageClassName: sc-hrhc
91   status:
92     conditions:
93       - lastTransitionTime: "2024-12-03T05:06:19Z"
94         message: replication pair has failed over
95         reason: replicationPairFailover
96         status: 'True'
97         type: Failover
98     currentOperation:
99       operation: none
100       running: false
101     replicationPair:
102       localSerialNumber: 810044
103       localStorageDeviceId: A34000810044
104       primaryVolumeId: 30
105       primaryVolumeNickname: spc-c420116eb7
106       remoteSerialNumber: 810138
107       remoteStorageDeviceId: A34000810138
108       secondaryVolumeId: 12
109       secondaryVolumeNickname: spc-594a21acc0
110   status: Failover
    
```

3. Verify the replication status from Storage Navigator.

Status from VSP One Block 24

Pair Detail		PairID(Secondary Volume to operator) / 0000
Status	Processing Status	000
CTE ID	Event Level	None
Secondary Volume Write	Enabled/Not Received	Enabled
Initial Copy Priority	30	
Period Time	2024/12/03 04:20:16	
Last Update Time	2024/12/03 05:00:46	
Pair Copy Time		
Local Storage System	Virtual Storage Machine	VSP One B24 / B1004
	Virtual LDEV ID	00-00-0C
	Virtual Service Name	
	Virtual SDD	
Remote Storage System	Virtual Storage Machine	VSP One B20 / VSP E series, VSP P100 and VSP Q100 / B10138
	Virtual LDEV ID	00-00-1E

Status from VSP One Block 28

Pair Detail		PairID(Secondary Volume to operator)
Status	Processing Status	000
CTE ID	Event Level	None
Secondary Volume Write	Enabled	
Initial Copy Priority	30	
Period Time	2024/12/03 04:20:16	
Last Update Time	2024/12/03 05:00:23	
Pair Copy Time		
Local Storage System	Virtual Storage Machine	VSP One B28 / B10138
	Virtual LDEV ID	00-00-1E
	Virtual Service Name	
	Virtual SDD	
Remote Storage System	Virtual Storage Machine	VSP One B20 / VSP E series, VSP P100 and VSP Q100 / B1004
	Virtual LDEV ID	00-00-0C

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**.

Overview

Applications > mongodbapp

mongodbapp

Overview Topology

Details

Name: mongodbapp

Type: Application set - Push model Argo

Namespace: openshift-gitops

Clusters: Local

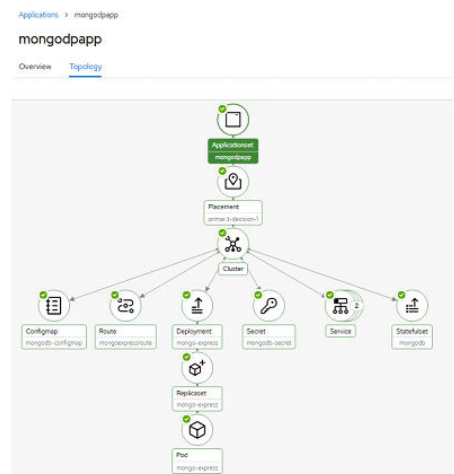
Repository: Git https://gitlab.apps.primarycluster.juno.com/root/mongoapp.git

Cluster resource status: 8

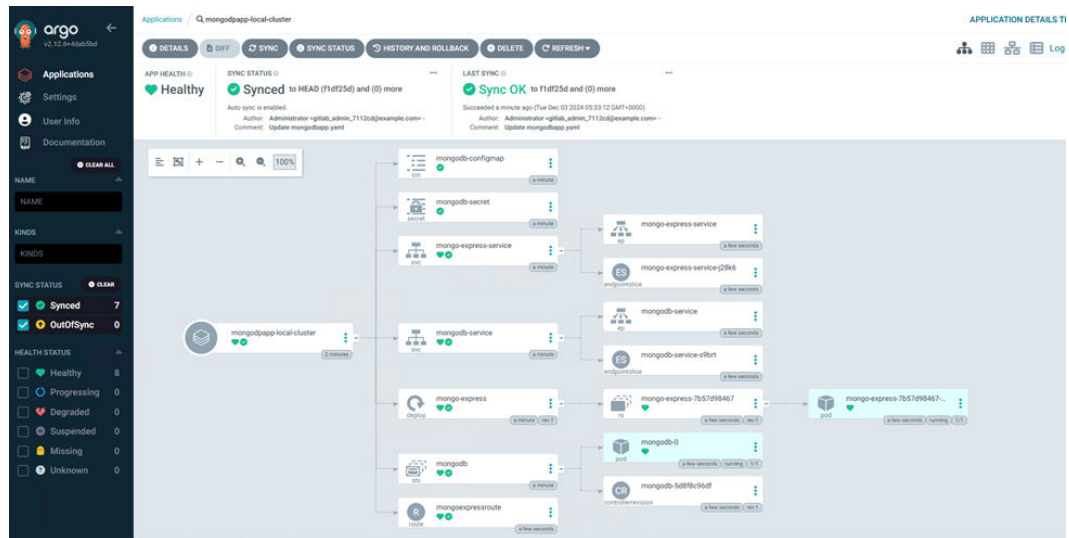
Created: 5:33 am

Last reconciled: 5:33 am

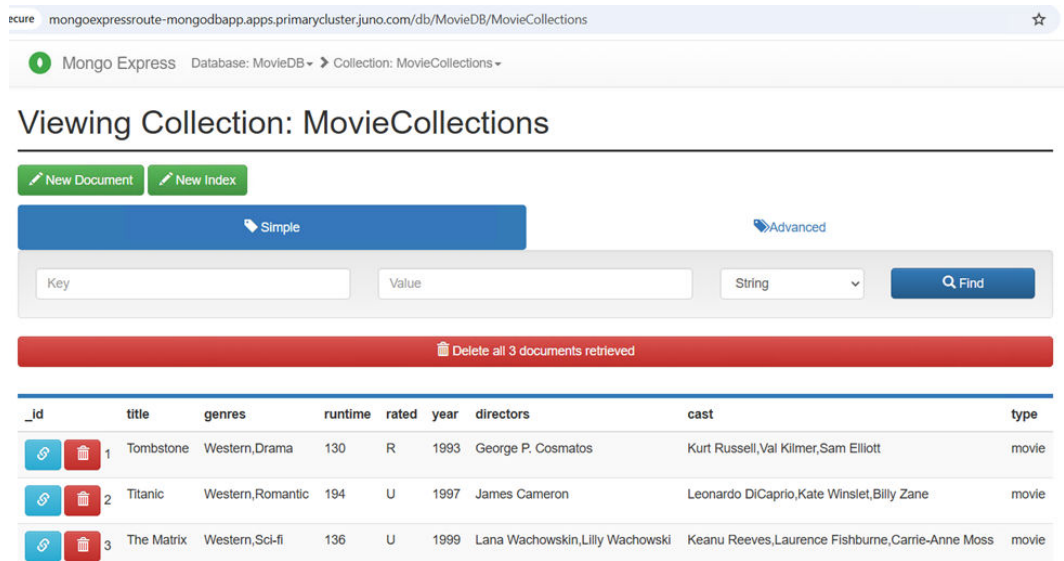
Topology



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**.

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

Add Document

```
1 {
2   "_id": 4,
3   "title": "The Godfather",
4   "genres": [ "Crime", "Drama" ],
5   "runtime": 175,
6   "rated": "U",
7   "year": 1972,
8   "directors": [ "Francis Ford Coppola" ],
9   "cast": [ "Marlon Brando", "Al Pacino", "James Caan" ],
10  "type": "movie"
11 }
```

Close Save

12. View the collection.

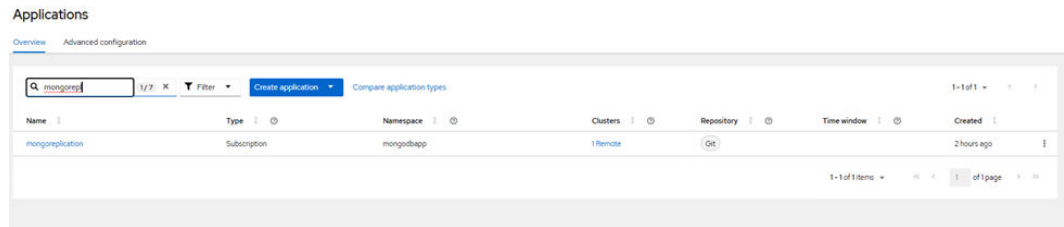
The screenshot shows the Mongo Express web interface. At the top, the URL is `mongoexpressroute-mongodbbapp.apps.primarycluster.juno.com/db/MovieDB/MovieCollections`. Below the URL bar, it says "Mongo Express Database: MovieDB -> Collection: MovieCollections". A green notification bar indicates "Document added!". There are buttons for "New Document" and "New Index". Below that, there are tabs for "Simple" and "Advanced". A search bar with "Key" and "Value" fields and a "Find" button is present. A red bar at the bottom of the search area says "Delete all 4 documents retrieved". Below this is a table with the following data:

_id	title	genres	runtime	rated	year	directors	cast	type
1	Tombstone	Western,Drama	130	R	1993	George P. Cosmatos	Kurt Russell,Val Kilmer,Sam Elliott	movie
2	Titanic	Western,Romantic	194	U	1997	James Cameron	Leonardo DiCaprio,Kate Winslet,Billy Zane	movie
3	The Matrix	Western,Sci-fi	136	U	1999	Lana Wachowski,Lilly Wachowski	Keanu Reeves,Laurence Fishburne,Carrie-Anne Moss	movie
4	The Godfather	Crime,Drama	175	U	1972	Francis Ford Coppola	Marlon Brando,Al Pacino,James Caan	movie

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

Procedure

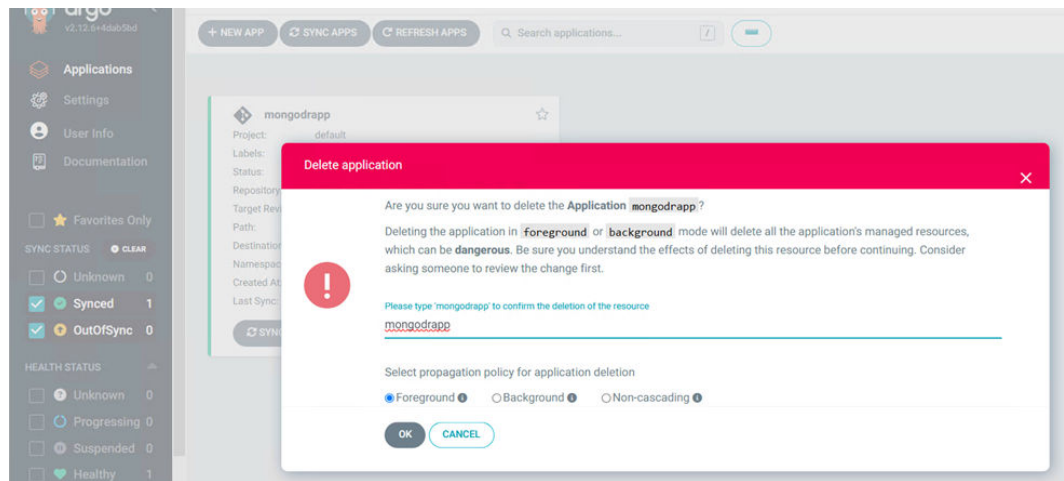
1. Delete the replication CR created previously. From Red Hat Advanced Cluster Management Applications, select the **mongoreplication**, click the 3 dots, and then click **Delete Application**.



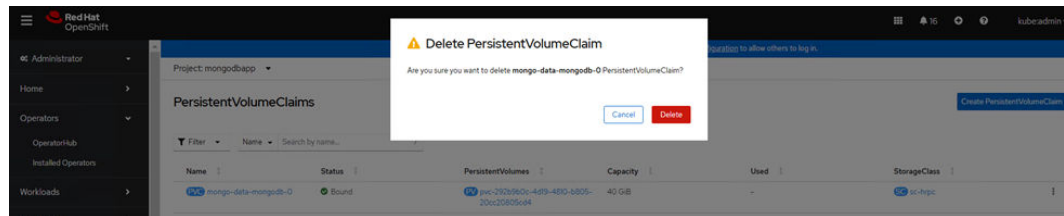
2. Confirm that all resources for **replication**, **localvolume**, and **remotevolume** are deleted.

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

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



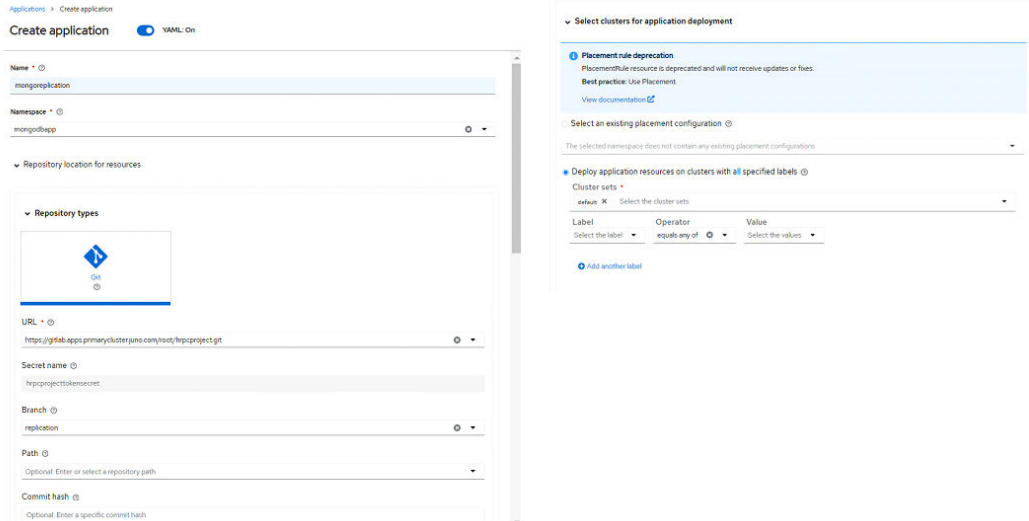
4. Delete the PVC from the secondary cluster.



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 **mongobapp**, select Git, add the URL of **hrpcproject**, select Branch as **replication**, and in the **Cluster sets** section, select **default**.



The application is created.

Overview

Applications > mongoreplication

mongoreplication

Overview Topology

Details

- Name: mongoreplication
- Type: Subscription
- Namespace: mongodbapp
- Clusters: Local
- Repository: Git <https://gitlab.apps.primarycluster.juno.com/root/hrp-project.git> [Set time window](#)
- Cluster resource status: ✔ 1
- Created: 6:36 am
- Last sync requested: [Sync](#)

Topology

Applications > mongoreplication

mongoreplication

Overview Topology



2. Verify the replication status from Storage Navigator.

Status from VSP One Block 24

Local Storage System		Remote Storage System	
Virtual Storage Machine	VSP One B24 / B10044	Virtual Storage Machine	VSP One B20, VSP E series, VSP F400 and VSP G400 / B10138
Virtual LDEV ID	00:00:0C	Virtual LDEV ID	00:00:0E

Status from VSP One Block 28

Local Storage System		Remote Storage System	
Virtual Storage Machine	VSP One B28 / B10138	Virtual Storage Machine	VSP One B20, VSP E series, VSP F400 and VSP G400 / B10138
Virtual LDEV ID	00:00:0C	Virtual LDEV ID	00:00:0E

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

R 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-0426305-1a12-49f6-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.

The screenshot shows the Hitachi VSP One Block Administrator interface. The main panel displays details for a volume named 'spc-0c4943c04cA34000810044S00012'. On the left, there is a 'Capacity Usage' section with a donut chart showing 'Used' (blue) and 'Free' (grey) space. The volume is attached to a pool named 'p0'. Other details include 'Volume Type: Attached Unmanaged', 'Capacity Saving: Enabled', 'Data Reduction Share: Apply to snapshot', and 'Number of Assigned Servers: 0'. The interface also shows 'Servers' and 'Number of Snapshots' as 0.

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