

Hitachi Storage Integrations with UCP and Red Hat OpenShift

Reference Architecture Guide

© 2022 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., 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, AS/400e, DB2, Domino, DS6000, DS8000, Enterprise Storage Server, eServer, FICON, FlashCopy, GDPS, HyperSwap, IBM, Lotus, MVS, OS/390, PowerHA, PowerPC, RS/6000, S/390, System z9, System z10, Tivoli, z/OS, z9, z10, z13, z14, 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.

Copyright and license information for third-party and open source software used in Hitachi Vantara products can be found in the product documentation, at <https://www.hitachivantara.com/en-us/company/legal.html> or https://knowledge.hitachivantara.com/Documents/Open_Source_Software.

Feedback

Hitachi Vantara welcomes your feedback. Please share your thoughts by sending an email message to SolutionLab@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
Updated to support OCP 4.8.	July 6, 2022

Reference Architecture Guide

This paper demonstrates best practice operations for a reference configuration of Red Hat OpenShift Container Platform (OCP) environment deployed on a Hitachi Unified Compute Platform. It leverages the latest capabilities of data storage integrations and services to create, protect, and manage on-premises Kubernetes clusters and workloads including application services that require persistent storage. These are relevant to deployment configurations with Hitachi Unified Compute Platform (UCP) in a hybrid converged/hyperconverged configuration or Hitachi Virtual Storage Platform (VSP) with OpenShift whether the configuration is bare metal, hypervisor-based, or hybrid OpenShift. This paper covers private registry for air-gap environments, data protection of k8s, and persistent storage to cloud resources. It also covers persistent storage options that are available to you in hybrid bare metal and hypervisor-based deployments using well-known and supported storage integrations between Hitachi Vantara, Red Hat OpenShift, and VMware.

A key element in the successful deployment of a container platform is having a robust and flexible infrastructure that can meet a wide variety of requirements in a highly dynamic environment. Hitachi infrastructure with Red Hat OpenShift provides highly available and high-performance infrastructure for container applications. Some specific challenges of providing an infrastructure for a container platform are:

- Data persistence

Data is at the core of any application. Many applications require data persistence, such as MariaDB, PostgreSQL, MongoDB, and MySQL, among others. Continuous integration and continuous delivery (CI/CD) pipelines require data persistency at every level.

Using well-known and proven CSI (Container Storage Interface) storage integrations, you can provide persistent storage for stateful container applications. The Hitachi UCP solution includes a Hitachi Storage CSI driver supported with Hitachi Storage Plug-in for Containers (HSPC) and a VMware Container Native Storage (CNS) implementation supported with Hitachi Storage Provider for VMware vCenter (VASA) software. For example, using vSphere storage policies in combination with VASA, you can provide dynamic ReadWriteOnce (RWO) VMFS and vVols-based VMDK persistent volumes to container applications running within OpenShift on top of VMware clusters.

This combination of integrations can meet majority of persistent storage configurations and data services that are needed. Hitachi UCP with Red Hat OpenShift provides the infrastructure and integrations needed for your organization to successfully provide container services to your application teams.

- Backup, data protection, and replication

Backup is a critical aspect of any data center infrastructure. Red Hat OpenShift Application Data Protection (OADP) includes a built-in Velero operator to enable your organization to protect any container-related entity, including Kubernetes persistent volumes.

The integration of Hitachi Storage Plug-in for Containers (HSPC) with OpenShift brings other benefits such as snapshot and cloning and restore operations for persistent volumes, enabling rapid copy creation for immediate use in decision support, software development, and data protection operations.

Hitachi Content Platform for cloud scale (HCP for cloud scale) provides standard AWS-compliant S3 storage that can be used with Red Hat OADP/Velero implementations as target repository storage.

Hitachi Replication Plug-in for Containers (HRPC) supports any Kubernetes cluster configured with Hitachi Storage Plug-in for Containers and provides data protection, disaster recovery, and migration of persistent volumes to remote Kubernetes clusters. HRPC supports replication for both bare metal and virtual environments.

- Computing platform

With a wide range of applications that are stateful or stateless, a wide range of flexible computing platforms are necessary to match both memory and CPU requirements.

The type of computing technology is also a consideration for licensing costs. Hitachi Vantara provides different computing options from the 1U/2U dual socket Hitachi Advanced Server DS120/220 G1/G2 to the 2U quad socket Hitachi Advanced Server DS240 G1.

- Network connectivity

As with any infrastructure, a reliable network is needed to provide enough bandwidth and security for container architectures. Hitachi Unified Compute Platform uses a spine and leaf design using Cisco Nexus or Arista switches.

- Infrastructure management

Having a robust and flexible infrastructure without efficient lifecycle management decreases efficiency exponentially as the infrastructure scales.

Orchestration and automation are the key to operational efficiencies. Hitachi Unified Compute Platform (UCP) Advisor provides a single pane of glass management and lifecycle manager for converged infrastructure, with automation for compute, network, and storage infrastructure. Hitachi Ops Center is also available with Hitachi Virtual Storage Platform (VSP) for storage management.

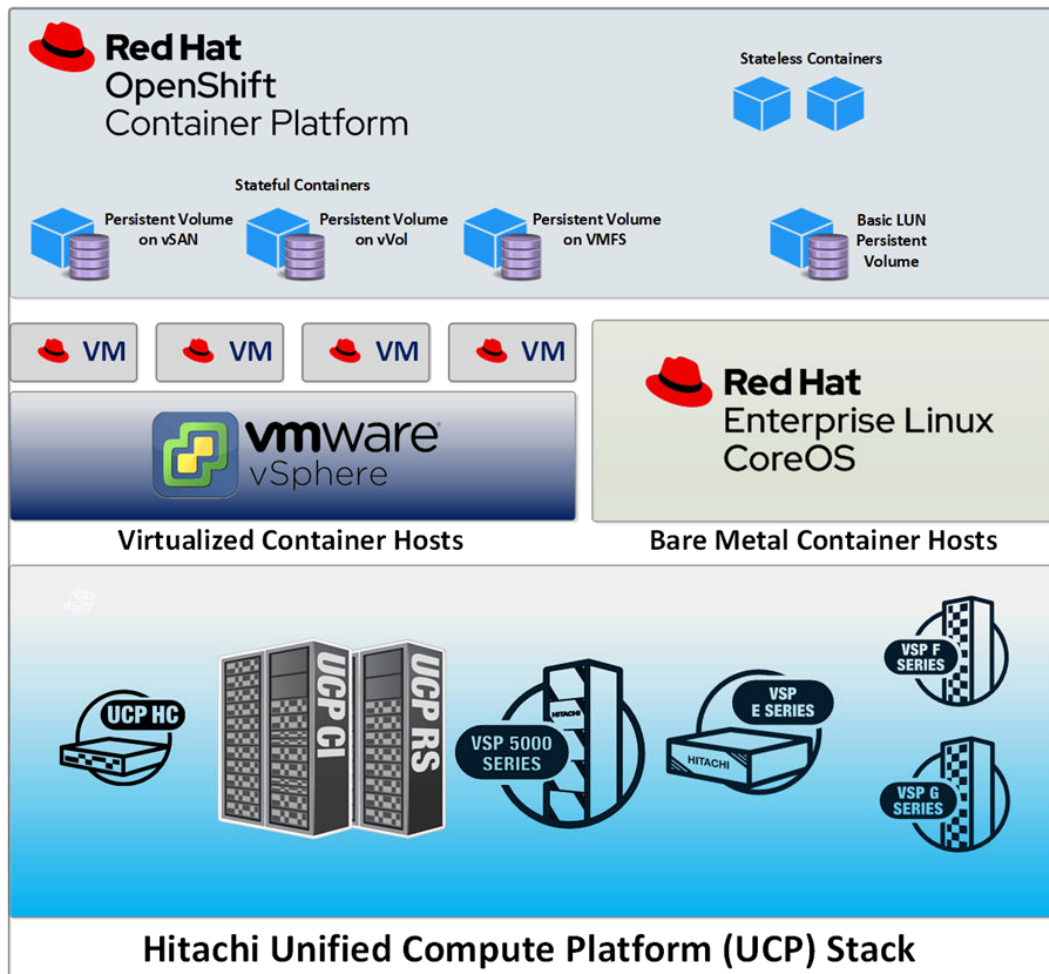
From monitoring perspective, Hitachi Storage Plug-in for Prometheus (HSPP) enables Kubernetes administrators to monitor metrics for Kubernetes resources and the Hitachi Storage resources with a single tool.

This reference architecture also provides the reference design for a build-your-own Red Hat OpenShift Container Platform environment using Hitachi Virtual Storage Platform. Although a specific converged system is used as an example, this reference design still applies to building your own container platform.

The intended audience of this document is IT administrators, system architects, consultants, and sales engineers to assist in planning, designing, and implementing Unified Compute Platform CI with OpenShift Container Platform solutions.

Solution overview

Red Hat OpenShift is a successful container orchestration platform and is one of the container orchestration solutions available with Unified Compute Platform. The following figure shows a high-level diagram of OpenShift managing containers and persistent volumes on the Unified Compute Platform stack with Hitachi Virtual Storage Platform series systems.



You can deploy OpenShift on bare metal hosts and/or virtual hosts or both. In some cases, the master nodes are virtualized while the worker nodes can be hybrid bare metal and virtual nodes. Depending on the deployment purposes, different deployments can be used. For this reference architecture, the OpenShift clusters use a hybrid deployment, combining bare metal and virtual worker nodes to show the benefits of both types of deployments.

These are the storage options and capabilities for bare metal worker nodes:

- Any storage system from the Hitachi Virtual Storage Platform family can be used. Virtual Storage Platform provides a REST API for Hitachi Storage Plug-in for Containers to provision persistent volumes. Deploy Storage Plug-in for Containers within the OpenShift Container Platform cluster in the respective cluster. Containers can access the persistent volumes through a local mount point inside the worker node. The persistent volumes are provided by Virtual Storage Platform-hosted LUNs through a block protocol to the worker nodes.
- Hitachi Storage Plug-in for Containers dynamically provisions persistent volumes for stateful containers from Hitachi storage.

These are the storage options and capabilities for virtual worker nodes:

- Any storage system from the Hitachi Virtual Storage Platform family can be used, as well as Hitachi Unified Compute Platform. Hitachi Storage Provider for VMware vCenter provides Virtual Storage Platform capabilities awareness to VMware vCenter, where it can be used with VMware Storage Policy-Based Management.
- VMware Cloud Native Storage (CNS) provides persistent storage provisioning capabilities using the VMware storage stack. Containers can access the persistent volumes through a local mount point inside the worker node virtual machines. The persistent volumes are provided by VMDKs provisioned from VMFS or vVols datastores from Virtual Storage Platform. You can also provide persistent volumes through Unified Compute Platform HC based on VMware Virtual SAN Ready Nodes (vSAN Ready Nodes).

The following persistent volume options are available for this configuration:

- Use VMware vVols to provision persistent volumes directly from Hitachi storage.
- Create persistent volumes from regular VMFS datastores.
- Create persistent volumes from VMware vSAN datastores hosted by Hitachi Unified Compute Platform HC nodes.

The solution validation of this reference architecture consists of different use cases for data protection of stateful applications, replication data services for container volumes across data centers, and monitoring and private registry, all running within OpenShift clusters on top of Hitachi UCP.

Follow the steps in Solution design and Solution Implementation and Validation to learn about the storage capabilities, data protection, and monitoring features when using Hitachi UCP with Red Hat OpenShift Container Platform.

Solution components

The following tables list the versions of hardware and software tested in this reference architecture.

Hardware components

The tested solution used specific features based on the following hardware. You can use either Hitachi Advanced Server DS120/DS220/DS225/DS240 Gen1 or Gen2 or any qualified server platform for UCP.

See the [UCP CI Interoperability Matrix](#) and [UCP Product Compatibility Guide](#) for more information.

Table 1 Hardware Components

Hardware	Description	Version	Quantity
Hitachi Advanced Server DS120 (for VMware compute cluster)	<ul style="list-style-type: none"> ▪ 2 × Intel Xeon 6240 18-core 2.60 GHz processors ▪ 16 × 16 GB DIMMs, 256 GB memory ▪ 32 GB SATADOM (boot) ▪ Emulex LPe3200 32 Gbps dual port PCIe HBA ▪ 2 × Mellanox CX4 dual port 10/25G NICs ▪ vSAN Cache Tier: 2 × Intel Optane SSD DC P4800X (375 GB, U.2) NVMe ▪ vSAN Capacity Tier: 10 × Intel SSD DC P4510 (4 TB, U.2) NVMe 	BMC: 4.68.06 BIOS: 3B19.H00	4
Hitachi Advanced Server DS120 (bare metal compute nodes)	<ul style="list-style-type: none"> ▪ 2 × Intel Xeon 4210 10-core 2.20 GHz processors ▪ 16 × 16 GB DIMM, 256 GB memory ▪ 1 x NVMe for boot ▪ Emulex LPe3200 32 Gbps dual port PCIe HBA ▪ 2 × Mellanox CX4 dual port 10/25G NICs 	BMC: 4.68.06 BIOS: 3B19.H00	2
Hitachi Virtual Storage Platform 5600 (used for replication on the primary site)	<ul style="list-style-type: none"> ▪ 2 TB cache ▪ 16 × 3.8 TB NVMe drives ▪ 4 × 32 Gbps Fibre Channel ports 	90-08-22-00/01	1
Hitachi Virtual Storage Platform 5500 (used for replication use case on the secondary site)	<ul style="list-style-type: none"> ▪ 2 TB cache ▪ 8 × 1.9 TB NVMe drives ▪ 4 × 32 Gbps Fibre Channel ports 	90-08-22-00/01	1

Hardware	Description	Version	Quantity
Hitachi Virtual Storage Platform G600	<ul style="list-style-type: none"> ▪ 86 GB cache ▪ 8 × 1.2 TB SAS 10K drives ▪ 4 × 8 Gbps Fibre Channel ports 	83-05-44-40/00	1
Cisco Nexus 9332C switch (spine)	<ul style="list-style-type: none"> ▪ 32-port 40/100 GbE ▪ 2-port 1/10 GbE 	NXOS 9.3.5	2
Cisco Nexus 93180YC-FX switch (leaf)	<ul style="list-style-type: none"> ▪ 48-port 10/25 GbE ▪ 6-port 40/100 GbE 	NXOS 9.3.5	2
Cisco Nexus 92348	<ul style="list-style-type: none"> ▪ 48-port 1 GbE ▪ 4-port 1/10/25 GbE ▪ 2-port 40/100 GbE 	NXOS 9.3.5	1
Brocade G620	<ul style="list-style-type: none"> ▪ 48-port 16/32 Gbps Fibre Channel switch 	9.0.0b	2

Software components

The following table lists the key software components.

Table 2 Software Components

Software	Version
Hitachi Storage Virtualization Operating System RF	90-05-02-00/01 83-05-33-40/00
Hitachi UCP Advisor	3.10
Hitachi Storage Provider for VMware vCenter (VASA)	3.6.2
Red Hat OpenShift Container Platform (OCP)	4.9
OpenShift API for Data Protection (OADP) operator	1.0.2
Red Hat Quay	3.6.4
Hitachi Storage Plug-in for Containers	3.9
Hitachi Replication Plug-in for Containers	1.1
Hitachi Storage Plug-in for Prometheus	1.1
Hitachi HCP for Cloud Scale	2.4.1.2
VMware vSphere	7.0 Update 2 or newer

Solution design

This section outlines the detailed solution example for the Hitachi Unified Compute Platform and Red Hat OpenShift.

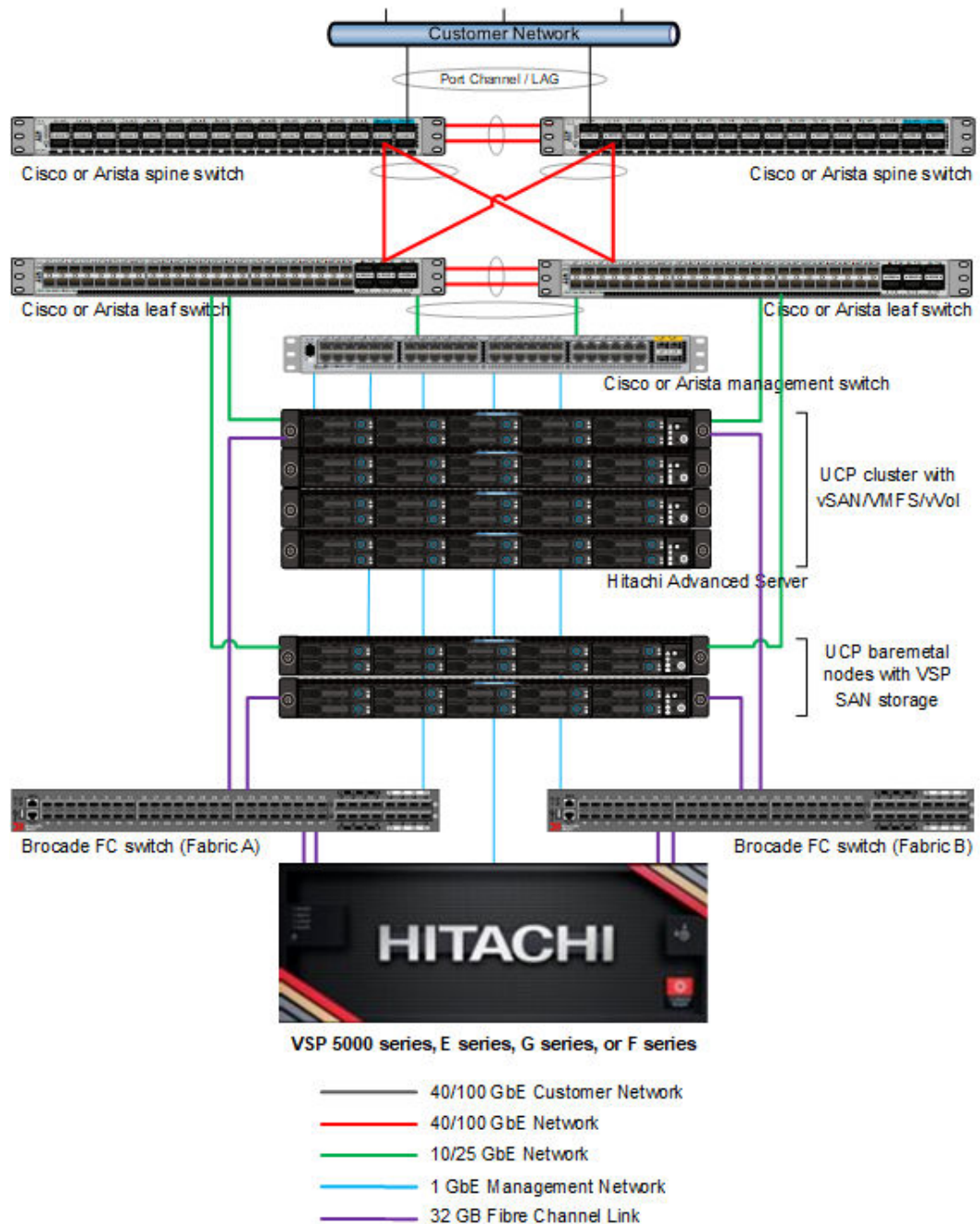
UCP infrastructure components

The following figure shows a high availability configuration of Hitachi Unified Compute Platform used to validate the Red Hat OpenShift solution. It includes the following components:

- Two Cisco 9332C or Arista 7050CX3 spine Ethernet switches.
- Two Cisco 93180YC-FX or Arista 7050SX3 leaf Ethernet switches.
- One Cisco 92348 or Arista 7010T management switch.
- Four Hitachi DS120 servers for vSAN cluster.
 - For vSAN compute nodes, leverage supported internal drives. These compute nodes are vSAN Ready Node Certified as UCP HC V120-series/V120F/V121F/V123F/V124N, UCP HC V220-series/V220F, UCP HC V225G, or UCP HC DS240, respectively. See [Hitachi Unified Compute Platform HC Series](#).
 - For vVols or VMFS compute nodes, leverage the HBA PCIe card, which is optionally configured together with the UCP HC vSAN ready nodes, or when configuring UCP Fibre Channel-only nodes in UCP RS.
- Two or more Hitachi DS120, DS220, DS225, or DS240 G1/G2 configured as bare metal worker nodes for separate OCP clusters.
- Two Hitachi VSP storage systems.

VMware vVols and storage policy-based management (SPBM)

The following diagram represents a standard architecture for Hitachi Unified Compute Platform (UCP) CI/ UCP HC/UCP RS.



The configuration with Hitachi Virtual Storage Platform is described in Unified Compute Platform CI/HC/RS in [HitachiUnified Compute Platform CI for VMware vSphere Reference Architecture Guide](#).

VMware vVols and storage policy-based management (SPBM)

Storage Provider for VMware vCenter (VASA) enables organizations to deploy Hitachi Storage infrastructure with VMware vSphere Virtual Volumes (vVols) to bring customers on a reliable enterprise journey to a software-defined, policy-controlled datacenter.

Hitachi storage policy-based management allows automated provisioning of virtual machines (VMs) and quicker adjustment to business changes. Virtual infrastructure (VI) administrators can make changes to policies to reflect changes in their business environment, dynamically matching storage-policy requirements for VMs to available storage pools and services. The vVols solution reduces the operational burden between VI administrators and storage administrators with an efficient collaboration framework leading to faster and better VM and application services provisioning.

To use VMware vVols with Hitachi storage, install VASA. See [VMware vSphere Virtual Volumes \(vVols\) with Hitachi Virtual Storage Platform Quick Start and Reference Guide](#) for details.

See [Storage Provider for VMware vCenter \(VASA\)](#) to deploy this environment.

Hitachi UCP Advisor

Hitachi UCP Advisor is used for SAN storage provisioning and SAN storage management, when vVols are not used.

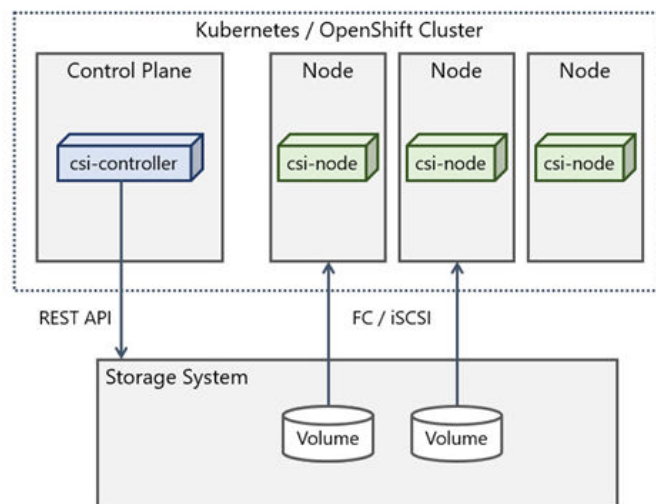
Deploy Hitachi UCP Advisor into the management cluster within your UCP infrastructure. See [Simplify Operations With Hitachi Unified Compute Platform Advisor](#) for more information.

Hitachi Storage Plug-in for Containers (HSPC)

Hitachi Storage Plug-in for Containers (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 provides persistent volumes from Hitachi Dynamic Provisioning (HDP) or Hitachi Thin Image (HTI) pools to bare metal or hybrid deployments using Fibre Channel or iSCSI protocols. iSCSI protocol is supported for both bare metal and virtual environments.

Storage Plug-in for Containers integrates Kubernetes or OpenShift with Hitachi storage systems using Container Storage Interface (CSI).

The following diagram illustrates a container environment where Storage Plug-in for Containers is deployed.



Understanding Red Hat OpenShift container platform and data protection

Red Hat OpenShift Container Platform (OCP) provides a single platform to build, deploy, and manage applications consistently across on-premises and hybrid cloud deployments. OCP provides the control plane and data plane within the same interface. OCP 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.

OCP also provides a developer view to allow users to deploy application resources from various pre-defined resources such as YAML files, Docker files, Catalogs, or GIT within user-defined namespaces. 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.

Red Hat OCP components

Within an OCP cluster there are multiple node types and roles that provide functionality to the container management platform. This section provides an overview of each node type and specific node roles within the cluster. Not all components are covered, and you are encouraged to see the Red Hat documentation for more information.

Master Nodes

Master nodes maintain the OCP cluster configuration as well as manage nodes within the cluster and schedule pods to run on worker nodes. Master nodes consist of an API server, controller manager server, certificate, and scheduler. If there is a master node outage, container applications will not be impacted and end users can continue using resources, but administrators of the cluster will not be able to make any changes to the cluster.

Worker Nodes

Worker nodes provide a runtime environment for containers and pods and are managed by the master nodes within the cluster. Worker nodes can either be virtual or physical based on the deployment type.

API Server

The API server, or kube-apiserver, provides the front end for the Kubernetes control plane by managing the interactions of cluster components via RESTful API calls. Administrators can run several instances of kube-apiserver to balance traffic among the cluster.

Scheduler

The scheduler, or kube-scheduler, ensures that container applications are scheduled to run on worker nodes within the OCP cluster. The scheduler reads data from the pod and finds a node that is a good fit based on configured policies.

Controller manager

The controller manager, or kube-controller-manager, provides the cluster with the necessary state changes to provide the most applicable state for a healthy cluster. The controller manager provides this functionality via the kube-apiserver.

Namespace (or Project)

Namespaces are intended for use in environments with many users spread across multiple teams, or projects. Namespaces provide a scope for names. Resource names need to be unique within a namespace, but not across namespaces. Namespaces cannot be nested inside one another and each Kubernetes resource can only be in one namespace.

Deployment methods and types

There are multiple deployment methods for Red Hat OCP based on the hardware configuration that supports the environment. These deployment methods include manual deployments using the User Provisioned Infrastructure (UPI) for customized deployments or fully automated deployments using the Installer Provisioned Infrastructure (IPI).

Within this guide, a hybrid environment is deployed using the UPI method. For more information about deployment methods and types, see the Red Hat OCP documentation.

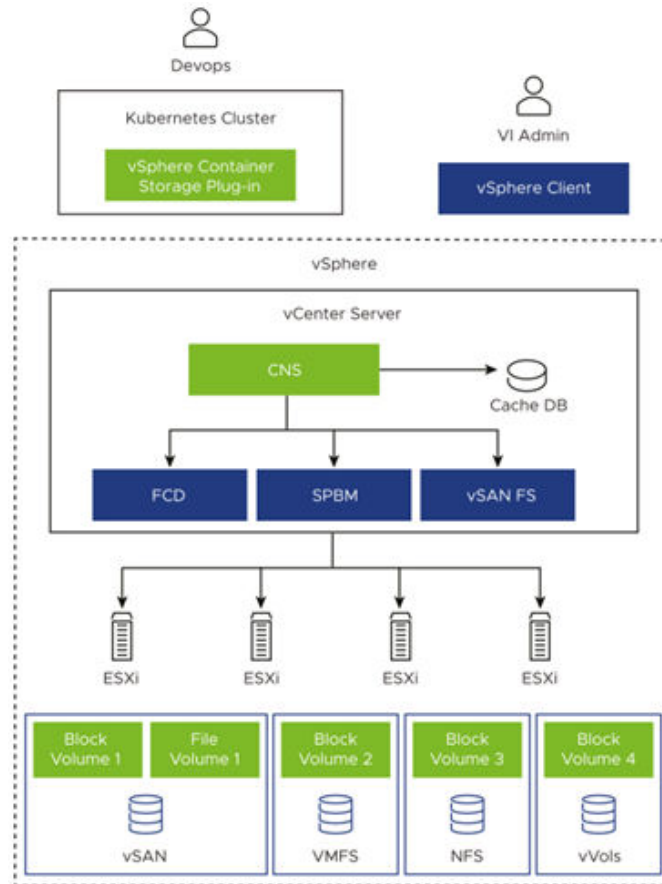
vSphere Cloud Native Storage (CNS) concepts

Cloud Native Storage (CNS) integrates vSphere and Kubernetes and offers capabilities to create and manage container volumes deployed in a vSphere environment. CNS consists of two components, a CNS component in vCenter Server and a vSphere volume driver (also called the vSphere CSI driver) in Kubernetes, called vSphere Container Storage Plug-in.

- CNS enables vSphere and vSphere storage (VMFS, vVols, and NFS), including vSAN, as a platform to run stateful applications. CNS enables access to this data path for Kubernetes and brings an understanding of Kubernetes volume and pod abstractions to vSphere. CNS uses several components to work with vSphere storage; this includes VMFS or vVols provided by the Hitachi Storage Provider for VMware vCenter. After you create PVs, you can review them and their backing virtual disks in the vSphere Client, and monitor their storage policy compliance.
- The vSphere Container Storage Plug-in has different components that provide an interface used by the Container Orchestrators such as OpenShift to manage the lifecycle of vSphere volumes. It also allows you to create, expand and delete volumes, attach and detach volumes to the cluster worker node VMs, and use bind mounts for the volumes inside the pods.

Because the vSphere CSI driver is installed in the OCP cluster, provisioning operations are similar to provisioning any Kubernetes cluster. A Persistent Volume Claim (PVC) is created that references an available StorageClass, which maps to a vSphere SPBM policy. A first class disk (FCD) is created within vSphere, and a resultant PV is presented to the OpenShift layer from the CSI driver. The FCD is then mounted to the pod when requested for use as a PV.

The following figure illustrates how CNS components, CNS in vCenter Server, and vSphere Container Storage Plug-in interact with other components in a vSphere environment (credit to VMware).



Deployment of Openshift worker nodes on VMware vSphere requires installation of the vSphere CSI driver to take advantage of the vSphere SPBM to Kubernetes StorageClass mapping.

Data protection options

Red Hat OpenShift API for Data Protection, or OADP brings an API to the OpenShift Container Platform that Red Hat partners and customers can leverage when creating a disaster recovery and data protection solution.

With OADP, you can back up and restore applications and services in the following two ways:

1. If Hitachi Storage Plug-in for Containers (HSPC) is installed (typically for bare metal workers with Fibre Channel or iSCSI or virtual worker with iSCSI), the backup process can use CSI snapshots. Native snapshots are typically much faster than file copying since they leverage copy-on-write capabilities from Hitachi VSP storage instead of doing a full copy.

When using CSI snapshots, only the Kubernetes metadata is backed up by Velero to the S3-compatible storage, while the HSPC leverages VSP capabilities to back up volume data using snapshot technologies.

This backup process is explored in [Solution Implementation and Validation \(on page 37\)](#), Scenario 1.

2. Another option is using Restic backups on an S3-compatible object storage such as Hitachi HCP CS S3.

Restic is a file system copying technique that is used by OADP. Backups with Restic are different than the backups produced with CSI snapshots because both volume data and Kubernetes metadata are backed up by Velero and Restic to the S3 storage.

Restic is agnostic and works with any type of storage. This has been fully tested with an OpenShift cluster as described in [Solution Implementation and Validation \(on page 37\)](#), Scenario 2.

OADP alone is not a full end-to-end data protection solution, but the integration with Hitachi Storage Plug-in for Containers and Hitachi HCP CS S3 storage provide a powerful solution for data protection for container-based applications and their associated PVs and data. The OADP operator sets up and installs Velero on the OpenShift cluster.



Note: The Red Hat OADP operator version 1.0 was released in Feb 2022 and is the first generally available release that is fully supported. Earlier versions (pre-1.0) of the operator were available as a community operator with only community support.

In addition, Hitachi Replication Plug-in for Containers (HRPC) together with Hitachi VSP storage replication capabilities can be used for data protection, disaster recovery, and migration of persistent volumes to remote datacenters/Kubernetes clusters.

Volume snapshots

In OpenShift or Kubernetes, creating a Persistent VolumeClaim (PVC) initiates the creation of a PersistentVolume (PV) which contains the data. A PVC also specifies a StorageClass which provides additional attributes for backend storage.

Because this guide also covers backup with CSI snapshots, it is important to clarify some additional concepts related to snapshots. A VolumeSnapshot represents a snapshot of a volume on the storage system. In the same way how API resources PersistentVolume and PersistentVolumeClaim are used to provision volumes for users and administrators, VolumeSnapshot and VolumeSnapshotContent API resources are provided to create volume snapshots. VolumeSnapshot support is only available for CSI drivers.

- `VolumeSnapshotContent` – Represents a snapshot taken of a Volume in the cluster. Similar to PersistentVolume object, the VolumeSnapshotContent is a cluster resource that points to a real snapshot in the backend storage. VolumeSnapshotContent are not namespaced.
- `VolumeSnapshot` - Is a request for snapshot of a volume. It is similar to a PersistentVolumeClaim. Creating a VolumeSnapshot triggers a snapshot (VolumeSnapshotContent) and the objects are bound together, there is a one-to-one binding between VolumeSnapshot and VolumeSnapshotContent. VolumeSnapshot are namespaced.
- `VolumeSnapshotClass` – Allows you to define different attributes belonging to a VolumeSnapshot. This is similar to how a StorageClass is used for PVs.

This is covered in section *Backing up persistent volumes with CSI snapshot* in this guide as a requirement for CSI snapshots.

Deploy OpenShift Container Platform

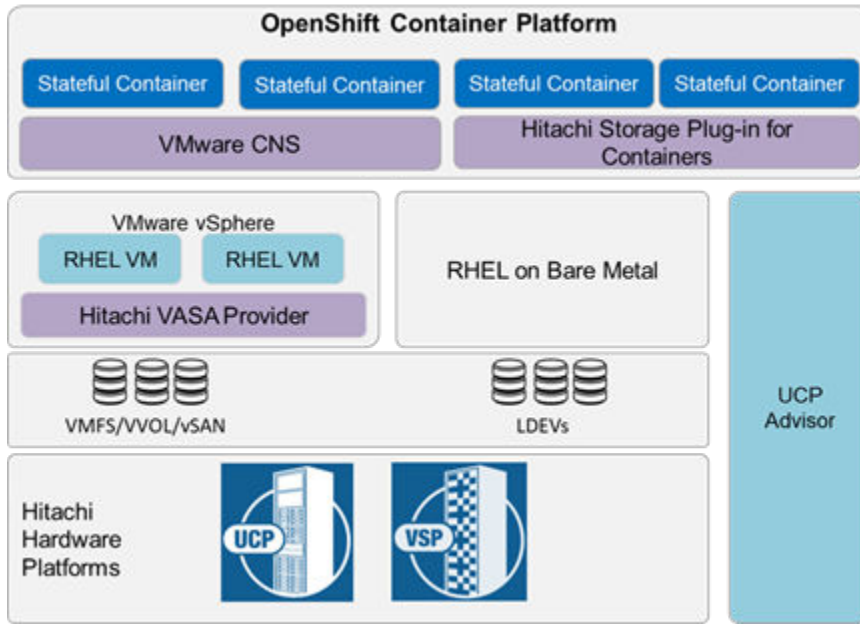
This guide does not cover the step-by-step how to implement OCP, follow Red Hat OCP documentation for the setup of the cluster.

Two OCP clusters have been configured to support the different use cases described in this guide, including replication of containerized applications between two Hitachi VSP Storage systems.

Two OCP clusters have been configured to support the different use cases described in this guide, each cluster with virtual and bare metal worker nodes and the following configuration:

- The virtual worker nodes had access to vSAN storage, as well as vVols and VMFS datastores supported by Hitachi VSP storage system and Hitachi Storage Provider for VMware vCenter (VASAProvider).
 - Details about the StorageClasses are provided in each of the use cases part of the solution implementation and validation.
- The bare metal worker node in cluster 1 (jpc2) was configured Fibre Channel connectivity to the VSP 5500 Storage System.
- The bare metal worker node in cluster 2 (jpc3) was configured Fibre Channel connectivity to the VSP 5600 Storage System.

The following figure illustrates a hybrid OpenShift Container Platform architecture on top of Hitachi UCP Platform stack for both clusters.



The tables below provide more details about the master and worker nodes for the two OCP clusters:

OCP Cluster (Primary site)

Node Name	Node Role	Device Type	OS-Image	Kubernetes Version
Router-LB	Load Balancer, Router	Existing infrastructure outside of UCP CI	CentOS	
jpc2-master-1	Master	VMs, hosted on a VMware cluster with DS120 G1, with vSAN, vVols, and VMFS datastores	Red Hat Enterprise Linux CoreOS 49.84	v1.22.5
jpc2-master-2	Master			v1.22.5
jpc2-master-3	Master			
jpc2-worker-1	Worker	Baremetal node (DS120)	Red Hat Enterprise Linux CoreOS 49.84	
jpc2-worker-2	Worker			
jpc2-worker-3	Worker			v1.22.5

OCP Cluster (Secondary site)

Node Name	Node Role	Device Type	OS-Image	Kubernetes Version
Router-LB	Load Balancer, Router	Existing infrastructure outside of UCP CI	CentOS	
jpc3-master-1	Master	VMs, hosted on a VMware cluster with DS120 G1, with vSAN, vVols, and VMFS datastores	Red Hat Enterprise Linux CoreOS 49.84	v1.22.5
jpc3-master-2	Master			v1.22.5
jpc3-master-3	Master			
jpc3-worker-1	Worker	Baremetal node (DS120)	Red Hat Enterprise Linux CoreOS 49.84	
jpc3-worker-2	Worker			
jpc3-worker-3	Worker			v1.22.5

For detailed hybrid OpenShift Container Platform installation procedures, see [Red Hat documentation](#).

Deploy and enable Hitachi Storage (VASA) Provider for VMware vCenter

To deploy the Hitachi Storage (VASA) Provider for VMware vCenter, follow these steps:

Procedure

1. Use UCP Advisor to create the necessary zone sets in the Fibre Channel fabrics.
2. Use Hitachi Storage Navigator to provision the necessary ALU targets from each storage system.
3. Register the Hitachi VSP storage systems in the VASA Provider.
4. Register VASA Provider as a Storage Provider within the vCenter associated with the cluster hosting the virtual worker nodes.
5. Create a vVol datastore or VMFS (LDEV) datastores and associated SPBM policies for the Hitachi VSP storage systems configured for use by the target OCP clusters.

For details, see [VMware vSphere Virtual Volumes \(vVols\) with Hitachi Virtual Storage Platform Quick Start and Reference Guide](#).

Install and Configure Hitachi Storage Plug-in for Containers (HSPC)

Hitachi Storage Plug-in for Containers is easily deployed to OpenShift using the Operator, which can be installed from OperatorHub. Follow the steps to:

- Install Hitachi Storage Plug-in for Containers.
- Configure Secret settings to access Hitachi VSP Storage system.
- Configure StorageClass settings.
- Configure Multipathing (FC or iSCSI).

Specific steps how to configure Persistent Volume Claims (PVC) and PODs will be covered as part of each of the use cases on the solution implementation and validation section.

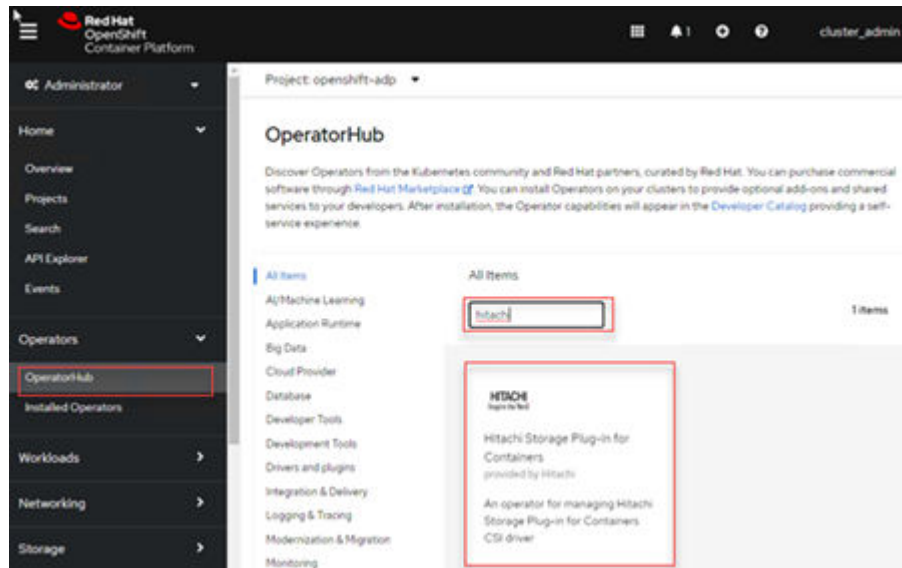


Note: If there is a previous version of Storage Plug-in for Containers, remove it before performing the installation procedure.

Install Hitachi Storage Plug-in for Containers

Procedure

1. Login to the console of your OCP cluster, select **OperatorHub** under **Operators**, then search for Hitachi.

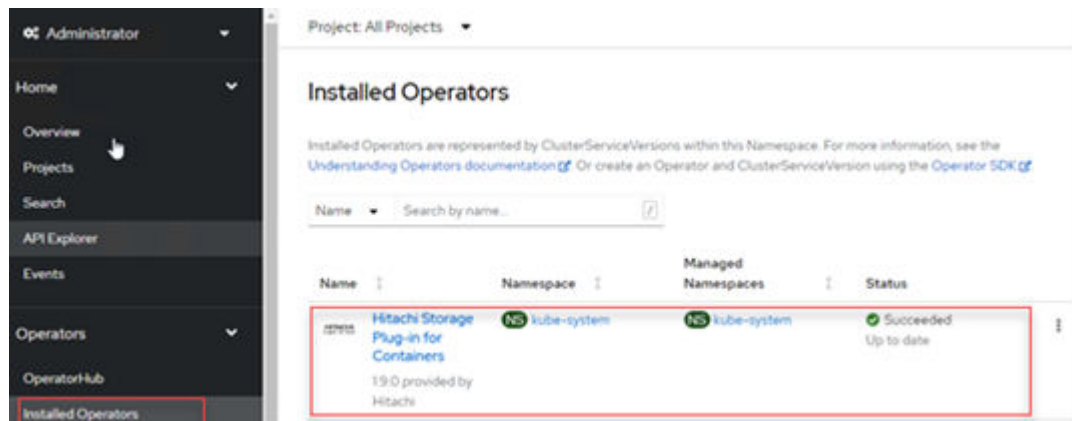


2. Click the Hitachi Storage Plug-in for Containers, and then click **Install**.



Note: Select the following settings in Operator Subscription:

- Installation Mode: A specific namespace on the cluster and <any namespace>
 - Approval Strategy: Manual and approve the Install Plan (see <https://docs.openshift.com/>).
3. Confirm the status of the Operator is **Succeeded** either using the console or `oc get pods -n <namespace>` command. On the console, click **Installed Operators** under **Operators** and you can see the status of the HSPC plug-in.



4. The next step is to create the HSPC Instance, this can be done using the Operator Details. Select the Hitachi Storage Plug-in for Containers, and then click **Create Instance** on the Operator Details. Click **Create**.
5. Confirm the status READY is true for the HSPC instance with the following command:

```
[ocpusr@dminws-c2]$oc get hspc -n <namespace>
NAME      READY   AGE
hspc     true    30s
```

Finally, verify that all the HSPC pods are in running state using the following command:

```
oc get pods -n <namespace>
```

```
[ocpusr@dminws-c2]$oc get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
hspc-csi-controller-d9b5b6b96-wp4fg	6/6	Running	0	3m16s
hspc-csi-node-drmt6	3/3	Running	0	3m17s
hspc-csi-node-sq4s1	3/3	Running	0	3m17s
hspc-csi-node-t2zh2	3/3	Running	0	3m17s
hspc-operator-controller-manager-5cd9684988-vlgw8	1/1	Running	0	3m16s

The Hitachi Storage Plug-in for Containers (HSPC) has been successfully installed. If you want to make an advanced configuration, refer to [Configuration of Storage Plug-in for Containers](#).

Configure Secret

The secret contains storage system information that enables access to the Storage Plug-in for Containers. It contains the storage URL (VSP REST API), user and password settings. Here an example of the YAML manifest file:

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-vsp-113
  namespace: test
type: Opaque
data:
  url: aHR0cHM6Ly8xNzIuMjUuNDcuMTEz
  user: b2NwdXNyMQ==
  password: SG10YWNoaTEh
```

The URL, user, and password are base64 encoded. Here an example how to get the base64 encoded of a user called “ocpusr1”, do the same for the URL and password:

```
[ocpusr@dminws-c2]$echo -n "ocpusr1" | base64
b2NwdXNyMQ==
[ocpusr@dminws-c2]$
```

One way to create a secret is using the following command:

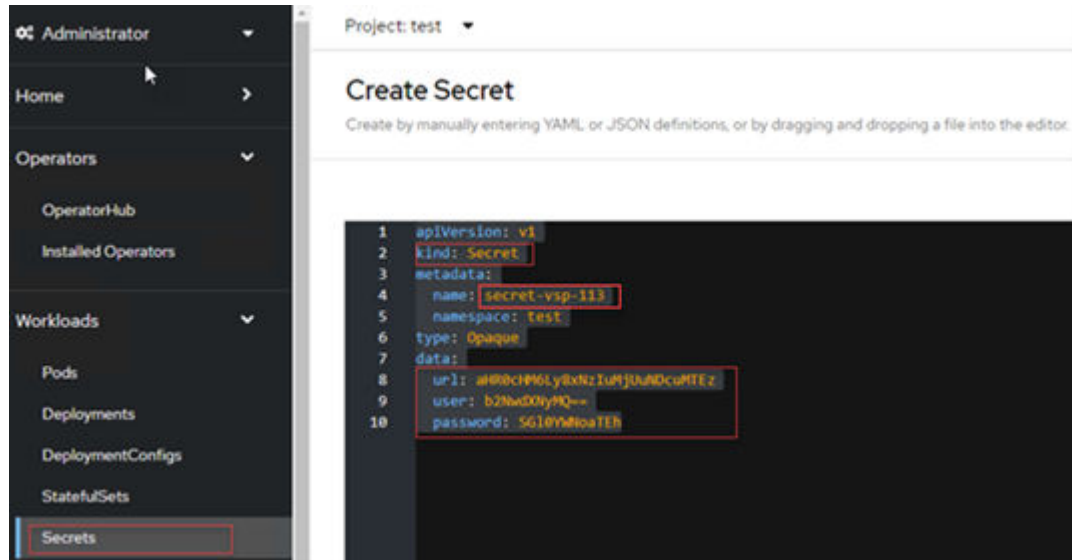
```
oc create -f <secret-manifest-file>
```

Another way to create a secret is using the OpenShift console:

Procedure

1. Login to the OCP console, select **Workloads > Secrets**.
2. Confirm the namespace in which you are creating the secret, in this example “test”.

3. Click **Create**. Select **From YAML**. Then, either copy/paste the content of the secret manifest file or just set the URL, user, and password in base64 encoded, and assign a name and corresponding namespace.



4. Click **Create**.

Configure StorageClass

The StorageClass contains storage settings that are necessary for Storage Plug-in for Containers to work with your environment. The following YAML manifest file provides information about the required parameters:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: sc-vsp-113
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
provisioner: hspc.csi.hitachi.com
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
parameters:
  serialNumber: '40016'
  poolID: '1'
  portID: 'CL1-D,CL4-D'
  connectionType: fc
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: test
  csi.storage.k8s.io/provisioner-secret-name: secret-vsp-113
  csi.storage.k8s.io/node-stage-secret-name: secret-vsp-113
  csi.storage.k8s.io/controller-expand-secret-name: secret-vsp-113
```


Here additional details about some of these parameters:

- serialNumber: VSP serial number
- provisioner: For HSPC, the default is `hspc.csi.hitachi.com`
- poolID: Pool ID on the VSP used to carve dynamically persistent volumes
- portID: VSP storage ports, use a comma separator for multipath
- connectionType: The connection type between storage and nodes, `fc` and `iscsi` are supported. If blank, `fc` is set
- fstype: Set the filesystem type as `ext4`
- secret-name: Define the VSP secret name
- secret-namespace: Enter the same namespace used for the secret

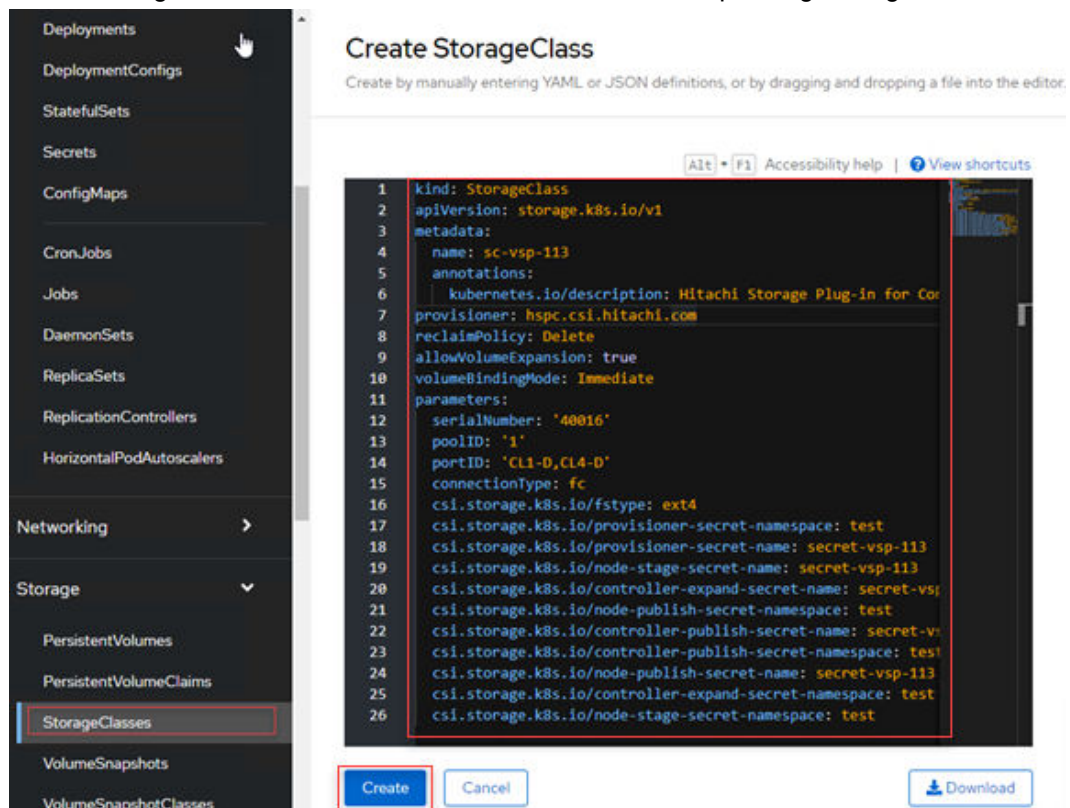
One way to create a StorageClass is using the following command:

```
oc create -f <storage-class-manifest-file>
```

Another way to create a StorageClass is using the OpenShift console:

Procedure

1. Log in to the OCP console, and then select **Storage > StorageClasses**.
2. Click **Create StorageClass**. Then, click **Edit YAML**. Then, either copy/paste the content of the StorageClass manifest file or enter each of the corresponding settings.



The screenshot shows the OpenShift console interface. On the left, a sidebar menu lists various system components, with 'StorageClasses' highlighted under the 'Storage' section. The main content area is titled 'Create StorageClass' and contains a text editor with the following YAML manifest:

```

1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: sc-vsp-113
5   annotations:
6     kubernetes.io/description: Hitachi Storage Plug-in for Co
7 provisioner: hspc.csi.hitachi.com
8 reclaimPolicy: Delete
9 allowVolumeExpansion: true
10 volumeBindingMode: Immediate
11 parameters:
12   serialNumber: '40016'
13   poolID: '1'
14   portID: 'CL1-D,CL4-D'
15   connectionType: fc
16   csi.storage.k8s.io/fstype: ext4
17   csi.storage.k8s.io/provisioner-secret-namespace: test
18   csi.storage.k8s.io/provisioner-secret-name: secret-vsp-113
19   csi.storage.k8s.io/node-stage-secret-name: secret-vsp-113
20   csi.storage.k8s.io/controller-expand-secret-name: secret-vsp-113
21   csi.storage.k8s.io/node-publish-secret-namespace: test
22   csi.storage.k8s.io/controller-publish-secret-name: secret-vsp-113
23   csi.storage.k8s.io/controller-publish-secret-namespace: test
24   csi.storage.k8s.io/node-publish-secret-name: secret-vsp-113
25   csi.storage.k8s.io/controller-expand-secret-namespace: test
26   csi.storage.k8s.io/node-stage-secret-namespace: test

```

At the bottom of the editor, there are three buttons: 'Create' (highlighted with a red box), 'Cancel', and 'Download'.

3. Click **Create**.

Configure Multipathing

For worker nodes connected to Hitachi VSP Storage via FC or iSCSI, it is recommended to enable multipathing. The requirement is to create the `multipath.conf` and ensure that the `user_friendly_names` option is set to `yes` and the `multipathd.service` is enabled. This can be done by applying an MCO (Machine Config Operator) to the OCP cluster after it has been deployed. Note, applying MachineConfig will restart the worker nodes one at a time.

Consider the following before applying the multipath configuration:

- For Fibre Channel, ensure that FC switches are configured with proper zoning for the compute worker nodes and Hitachi VSP storage systems are accessible to each other.
- For iSCSI, ensure the Hitachi VSP storage is properly configured for iSCSI and the compute worker nodes can access the iSCSI targets. Also, for iSCSI, check the [Hitachi Storage Plug-in for Containers Release Notes](#) for additional considerations regarding IQN configurations.
- RedHat CoreOS (RHCOS) already includes the `device-mapper-multipath` package which is required to support multipathing. For solutions with iSCSI, RHCOS already has the iSCSI initiator tools installed by default. There is no need to install any additional package, apply the configurations as indicated in this section.

Configure multipathing for OCP worker nodes using MachineConfig:

To enable multipath for Hitachi HSPC, apply the MachineConfig below to the cluster. This will enable `multipathd` (for FC and iSCSI) needed by the Hitachi VSP Storage and HSPC integration on each worker node. It targets the worker nodes by using the label `machineconfiguration.openshift.io/role: worker`.

The following YAML file can be used for both Fibre Channel and iSCSI configurations with multipathing. To support iSCSI, uncomment the last three lines in the file.

A MachineConfig can be created directly from the command line using the following command `oc create -f <MachineConfigFile.yaml>` or using the OpenShift console.

Procedure using the OCP console

Procedure

1. To apply a MachineConfig using the OCP console, login to the OCP web console and navigate to **Compute > Machine Configs**. Click **Create Machine Config**, then copy and paste the content of the YAML file and click **Create**.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: workers-enable-multipath-conf
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
```

```

storage:
  files:
    - path: /etc/multipath.conf
      mode: 400
      filesystem: root
      contents:
        source: data:text/plain;charset=utf-8;base64,
ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgeWVzCiAgICAgICAgZmluZm9tdWx0a
XBhdGhzIHllcwp9CgpibGFja2xpc3Qgewp9Cg==
        verification: {}
  systemd:
    units:
      - name: multipathd.service
        enabled: true
        state: started
      # Uncomment the following 3 lines if this MachineConfig will be used
with iSCSI
      #- name: iscsid.service
      # enabled: true
      # state: started
osImageURL: ""

```

2. For iSCSI without multipathing, use the following MachineConfig:

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: workers-enable-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
osImageURL: ""

```



Note: The source data string located after the line `source: data:text/plain;charset=utf-8;base64,` for *multipath.conf* is base64 encoded. If you need to update the *multipath.conf* file to suite your environment needs, you can run `echo -n "<string>" | base64 -d` to decode the contents of the config file. If you want to update it, make your changes, and then re-encode the file using base64.

3. After the MachineConfig is created, every worker node is rebooted one at a time after the configuration is applied, and it could take from 20 to 30 minutes to apply the

configuration to the worker nodes. To verify whether the machine config is applied use the `oc get mcp` command to verify that the machine config pool for workers is updated. In addition, ssh to the worker nodes to confirm that the `/etc/multipath.conf` file has been created and the `multipathd` service is running, if it is iscsi verify that the `iscsid` service is running. Here an example:

```
[ocpusr@dminws-c2]# ssh core@jpc2-worker-1
Red Hat Enterprise Linux CoreOS 49.84.202204072350-0
Part of OpenShift 4.9, RHCOS is a Kubernetes native operating system
managed by the Machine Config Operator ('clusteroperator/machine-config').

WARNING: Direct SSH access to machines is not recommended; instead,
make configuration changes via 'machineconfig' objects:
https://docs.openshift.com/container-platform/4.9/architecture/architecture-rhcos.html

---
Last login: Wed Jun  8 06:59:37 2022 from 172.17.0.10
[core@jpc2-worker-1 ~]$
[core@jpc2-worker-1 ~]$ sudo cat /etc/multipath.conf
defaults {
    user_friendly_names yes
    find_multipaths yes
}

blacklist {
}
[core@jpc2-worker-1 ~]$
[core@jpc2-worker-1 ~]$ sudo systemctl status multipathd.service
● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-06-08 07:45:31 UTC; 1min 40s ago
 Main PID: 1075 (multipathd)
   Status: "up"
    Tasks: 7
  Memory: 13.4M
     CPU: 70ms
  CGroup: /system.slice/multipathd.service
          └─1075 /sbin/multipathd -d -s

Jun 08 07:45:30 localhost systemd[1]: Starting Device-Mapper Multipath Device Controller...
Jun 08 07:45:31 localhost multipathd[1075]: -----start up-----
Jun 08 07:45:31 localhost multipathd[1075]: read /etc/multipath.conf
Jun 08 07:45:31 localhost multipathd[1075]: path checkers start up
Jun 08 07:45:31 localhost systemd[1]: Started Device-Mapper Multipath Device Controller.
[core@jpc2-worker-1 ~]$

[ocpusr@dminws-c2]# oc get mc | grep multipath
workers-enable-multipath-conf          3.2.0
128m
[ocpusr@dminws-c2]#
[ocpusr@dminws-c2]# oc get mcp
NAME          CONFIG          UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT  UPDATED  UPDATING  DEGRADED  MACHINECOUNT  REA
DYMACHINECOUNT  AGE
master  rendered-master-b56c42b77170e0ea9152bb74022b85cf  True  False  False  3  3
3  141d
worker  rendered-worker-4661fa0f9113a74d65eef2c2fdbfd8ca  True  False  False  3  3
3  141d
[ocpusr@dminws-c2]#
```

Label nodes in the OCP cluster

Because the OCP cluster is a hybrid environment with virtual and bare metal worker nodes, it is important for the operator to distinguish node type between physical and virtual. Assigning the right labels to the nodes allows for granular deployment of resources to the correct worker node types. This process can be done either from command line or the OpenShift console.

Follow these steps to assign labels to the worker nodes.

From the command line, use this command:

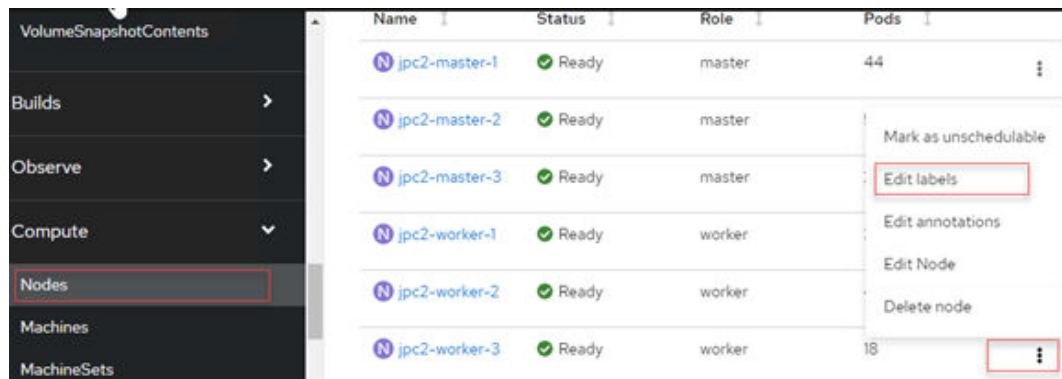
```
oc label nodes <node-name> <label>
```

Here an example where one virtual node is labeled *vm*, and one physical node is labeled *hspc-fc*:

```
[ocpusr@dminws-c2]$ oc label nodes jpc2-worker-3 nodeType=hspc-fc
node/jpc2-worker-3 labeled
[ocpusr@dminws-c2]$
[ocpusr@dminws-c2]$ oc label nodes jpc2-worker-2 nodeType=vm
node/jpc2-worker-2 labeled
[ocpusr@dminws-c2]$
```

Procedure

1. From the OCP console, select **Compute > Nodes**.
2. From the nodes list click the ellipsis icon on a worker node (physical or virtual) and select **Edit Labels**.



3. Assign the label using a key/value pair, for example: `nodeType=hspc-fc`, and then click **Save**.
4. Use the following command to verify the labels:


```
oc get nodes --show-labels
```

HSPC and VSP Host Groups

Host groups required for Storage Plug-in for Containers are automatically created by Storage Plug-in for Containers. Storage Plug-in for Containers automatically searches host groups and iSCSI targets based on the name.

If you want to use existing host groups, rename them according to the naming rule. For details, see *Host group and iSCSI target naming rules* in the [HSPC Reference Guide](#).



Note: Storage Plug-in for Containers will overwrite host mode options even if existing host groups have other host mode options.

HSPC and VSP Resource Groups

You can partition storage system resources by limiting the LDEV ID range added to the resource group for a specific Kubernetes cluster. You can also isolate impacts between Kubernetes clusters. The following requirements should be met:

- Only one resource group for one Kubernetes is supported. Virtual storage machines are not supported.
- Storage system users must have access only to the resource group that they created. The storage system user must not have access to other resource groups.
- Create a pool from pool volumes with the resource group that you have created.
- Allocate the necessary number of undefined LDEV IDs to the resource group.
- Allocate the necessary number of undefined host group IDs to the resource group for each storage system port defined in StorageClass. The number of host group IDs must be equal to the number of hosts for all ports.

For details, see *Resource Partitioning* in the [HSPC Reference Guide](#).

HSPC and VSP Resource Groups

You can partition storage system resources by limiting the LDEV ID range added to the resource group for a specific Kubernetes cluster. You can also isolate impacts between Kubernetes clusters. The following requirements should be met:

- Only one resource group for one Kubernetes is supported. Virtual storage machines are not supported.
- Storage system users must have access only to the resource group that they created. The storage system user must not have access to other resource groups.
- Create a pool from pool volumes with the resource group that you have created.
- Allocate the necessary number of undefined LDEV IDs to the resource group.
- Allocate the necessary number of undefined host group IDs to the resource group for each storage system port defined in StorageClass. The number of host group IDs must be equal to the number of hosts for all ports.

For details, see *Resource Partitioning* in the [HSPC Reference Guide](#).

Install and Configure Backup and Data Protection Operator

Complete the following procedures to configure and enable OpenShift Application Data protection (OADP)/Velero functionality with Hitachi HCP CS S3 storage.

Choose a target S3 object store for backups

While any AWS-compliant S3 target can be used as a Velero backup target, it is recommended that you use HCP for cloud scale due to the enterprise-grade compliance, security, retention, and replication features available.

It is recommended that you back up to a remote S3 target separate from your source infrastructure. This ensures maximum protection of your data in case of a site or infrastructure-specific failure.

Configure Hitachi Content Platform for cloud scale

You can obtain a 60-day trial of Hitachi Content Platform for cloud scale (HCP for cloud scale) by visiting <https://trycontent.hitachivantara.com/>. Follow the directions in your trial access email after registering to generate credentials and to create an S3 bucket to be used as a target for Velero backup data.

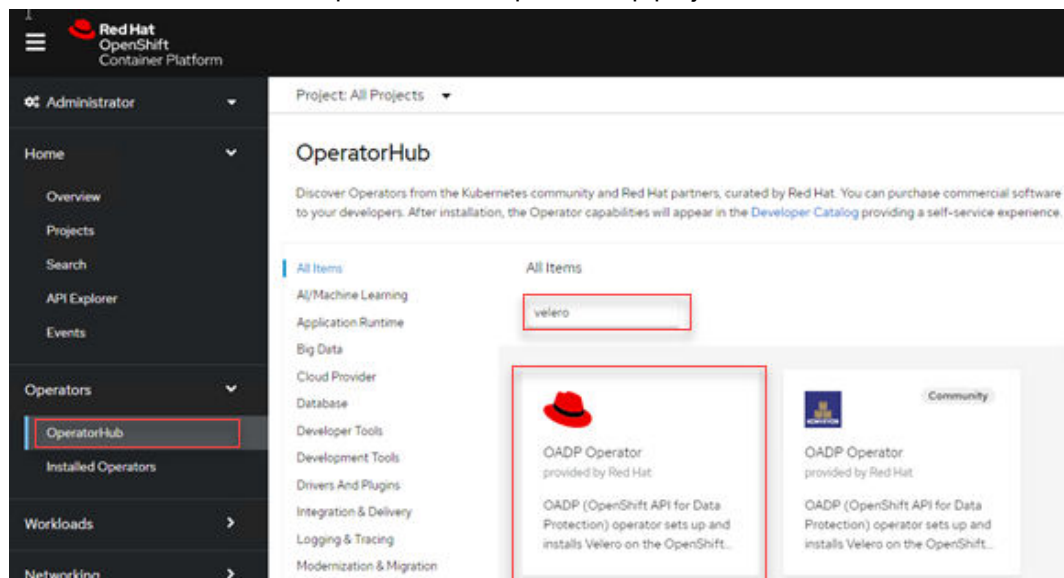
See [Hitachi Content Platform for Cloud Scale Architecture Fundamentals](#) for more information.

Install OpenShift Application Data Protection (OADP) from the OperatorHub

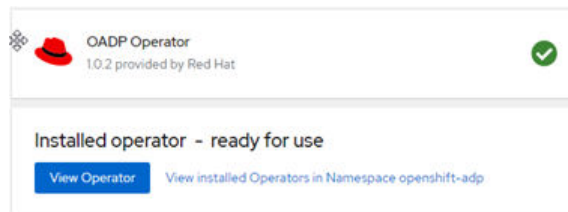
OpenShift Application Data protection (OADP) operator sets up and installs Velero on the OpenShift cluster.

Procedure

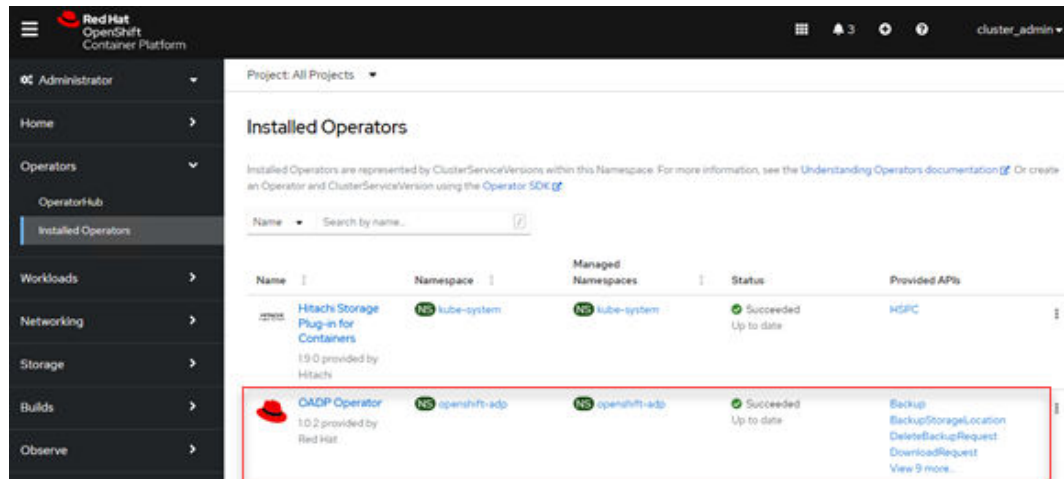
1. Log in to the OCP console, select **OperatorHub** under Operators, and then select the OADP Operator and click **Install**.
2. Click **Install** to install the Operator in the openshift-adp project.



Once the operator has been installed, it looks like this when it is ready to use:



3. The next step is to configure the Data Protection Application instance which deploys Velero and Restic pods.



4. Follow the instructions in the next section to complete the setup.

MK-SL-210 Create HCP CS/Velero credentials

Configure OpenShift Application Data Protection (OADP) with HCP CS

DataProtectionApplication instance represents configuration to install a data protection application to safely backup and restore, perform disaster recovery and migrate Kubernetes cluster resources and persistent volumes.

To perform the setup of the DataProtectionApplication instance, it is required to create the secret to access the S3 storage.

For additional and advanced setup see Red Hat OADP documentation.

Create secret for backup and snapshot locations

Installing the Data Protection Application custom resource (CR) requires creating a secret object; for this example we are using the same credentials for both backup and snapshot locations.

The following command uses the default name of the secret which is *cloud-credentials* and the credential file from previous step.

```
oc create secret generic cloud-credentials --namespace
openshift-adp --from-file cloud=velero-hcpcs-credentials
```

Create DataProtectionApplication instance

The installation of the Data Protection Application (DPA) is done by creating an instance of the DataProtectionApplication API. This can be done from the command line or by using the OpenShift console.

The following procedure uses the OpenShift console.

Procedure

1. Log in to the OCP console, click **Operators > Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.

3. Click **YAML View** and update the parameters of the `DataProtectionApplication` manifest with the following:
 - HCP CS information:
 - `S3URL` - Set the URL to the S3 endpoint URL of your HCP CS system.
 - `Region` - You can choose any region for this variable because HCP for cloud scale will accept any region name that is provided.
 - `Bucket` - This variable should be set to the S3 bucket name you configured in HCP for cloud scale.
 - `Credential` - set this to the name of the secret with the access keys for HCP CS (for example `cloud-credentials`).
 - Specify a prefix for Velero backups.
 - The snapshot location must be in the same region as the PVs.

- Enable the Container Storage Interface (CSI) in the DataProtectionApplication CR to back up persistent volumes (PVs) with CSI snapshots. In this specific case, CSI snapshots are supported with the Hitachi Storage Plugin for Containers (HSPC).

OADP Operator > Create DataProtectionApplication

Create DataProtectionApplication

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via: Form view YAML view

```

1  apiVersion: oadp.openshift.io/v1alpha1
2  kind: DataProtectionApplication
3  metadata:
4    name: velero-ocp-hcpcs
5    namespace: openshift-adp
6  spec:
7    backupLocations:
8      - velero:
9        config:
10         profile: default
11         region: us-west-1
12         insecureSkipTLSVerify: "true"
13         s3Url: "https://tryhcpforcloudscale.hitachivantara.com/"
14         s3ForcePathStyle: "true"
15       credential:
16         key: cloud
17         name: cloud-credentials
18       default: true
19       objectStorage:
20         bucket: ocp-eng-velero-target
21         prefix: velero
22       provider: aws
23     configuration:
24       restic:
25         enable: true
26       velero:
27         defaultPlugins:
28           - openshift
29           - aws
30           - csi
31           - kubvirt
32         featureFlags:
33           - EnableCSI
34     snapshotLocations:
35       - velero:
36         config:
37           profile: default
38           region: us-west-1
39         provider: aws

```

- Add the `csi` default plug-in.
- Add the `EnableCSI` feature flag.

4. Click **Create**.

5. Verify the installation of the OADP resources.

When the DataProtectionApplication instance is created, you should have Velero, Restic pods, and services running within the `openshift-adp` namespace.

- #### 6. Run the `oc get all -n openshift-adp` command and wait until all the pods are running successfully. The following figure shows both Velero and Restic pods running within the `openshift-adp` namespace.

```

oc get all -n openshift-adp

```

NAME	READY	STATUS	RESTARTS	AGE
pod/oadp-velero-ocp-hcpcs-1-aws-registry-99879dc56-gk4pr	1/1	Running	0	2m25s
pod/openshift-adp-controller-manager-5f8666d5dd-wwqfs	1/1	Running	0	3h9m
pod/restic-5b52k	1/1	Running	0	2m24s
pod/restic-bhqrm	1/1	Running	0	2m24s
pod/restic-zxhkn	1/1	Running	0	2m24s
pod/velero-9f866c45d-8qxtr	1/1	Running	0	2m24s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/oadp-velero-ocp-hcpcs-1-aws-registry-svc	ClusterIP	172.30.92.210	<none>	5000/TCP	2m25s
service/openshift-adp-controller-manager-metrics-service	ClusterIP	172.30.218.211	<none>	8443/TCP	3h9m
service/openshift-adp-velero-metrics-svc	ClusterIP	172.30.212.228	<none>	8085/TCP	2m24s

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
daemonset.apps/restic	3	3	3	3	3	<none>	2m24s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/oadp-velero-ocp-hcpcs-1-aws-registry	1/1	1	1	2m25s
deployment.apps/openshift-adp-controller-manager	1/1	1	1	3h9m
deployment.apps/velero	1/1	1	1	2m24s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/oadp-velero-ocp-hcpcs-1-aws-registry-99879dc56	1	1	1	2m25s
replicaset.apps/openshift-adp-controller-manager-5f8666d5dd	1	1	1	3h9m
replicaset.apps/velero-9f866c45d	1	1	1	2m24s

PATH	SERVICES	PORT	TERMINATION	HOST/PORT	WILDCARD
route.route.openshift.io/oadp-velero-ocp-hcpcs-1-aws-registry-route				oadp-velero-ocp-hcpcs-1-aws-registry-route	
openshift-adp.apps.jp2.ocp.hvlab.local				oadp-velero-ocp-hcpcs-1-aws-registry-svc	<all>

Velero is now enabled and installed on the RedHat OpenShift cluster.

The following is the text of the DataProtectionApplication custom resource (CR):

```

apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero-ocp-hcpcs
  namespace: openshift-adp
spec:
  backupLocations:
    - velero:
        config:
          profile: default
          region: us-west-1
          insecureSkipTLSVerify: "true"
          s3Url: "https://tryhcpforcloudscale.hitachivantara.com/"
          s3ForcePathStyle: "true"
        credential:
          key: cloud
          name: cloud-credentials
        default: true
        objectStorage:
          bucket: ocp-eng-velero-target
          prefix: velero
          provider: aws
  configuration:
    restic:
      enable: true
    velero:
      defaultPlugins:
        - openshift
        - aws
        - csi
        - kubevirt
      featureFlags:
        - EnableCSI

```

```

snapshotLocations:
  - velero:
      config:
        profile: default
        region: us-west-1
        provider: aws

```

Back up persistent volumes with CSI snapshot

To perform backup of persistent volumes with CSI snapshots, create a **VolumeSnapshotClass** custom resource (CR) to register the CSI driver. In this example we are using the Hitachi HSPC CSI driver.

The VolumeSnapshotClass CR must contain the following parameters:

- Ensure that the desired volumesnapshot class has the *velero.io/csi-volumesnapshot-class* label.
- The driver must use **hspc.csi.hitachi.com** which corresponds to Hitachi Storage Plug-in for Containers (HSPC).
- The poolID must be the same as the one specified in the StorageClass.
- The secret name and secret namespace must be the same as the ones specified in the StorageClass definition.

The YAML file below provides an example of a VolumeSnapshotClass CR using Hitachi Storage Plug-in for Containers (HSPC).

```
[ocpusr@dminws-c2]$ cat VolumeSnapshotClass_CSI_HSPC.yaml
```

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: volume-snapshot-class-csi-hspc
  labels:
    velero.io/csi-volumesnapshot-class: "true"
driver: hspc.csi.hitachi.com
deletionPolicy: Delete
parameters:
  poolID: "1"
  csi.storage.k8s.io/snapshotter-secret-name: "secret-vsp-113"
  csi.storage.k8s.io/snapshotter-secret-namespace: "test"

```

Procedure

1. Create the VolumeSnapshotClass with the following command:

```
oc apply -f VolumeSnapshotClass_CSI_HSPC.yaml
```

2. Verify the status with the following command:

```
oc get VolumeSnapshotClass
```

```
[ocpusr@dminws-c2]$ oc get VolumeSnapshotClass
NAME                                DRIVER                                DELETIONPOLICY  AGE
volume-snapshot-class-csi-hspc     hspc.csi.hitachi.com                 Delete          7d21h
```



Note: In addition to VolumeSnapshotClass, the CSI must be enabled on the DataProtectionApplication CR as indicated in the previous section.

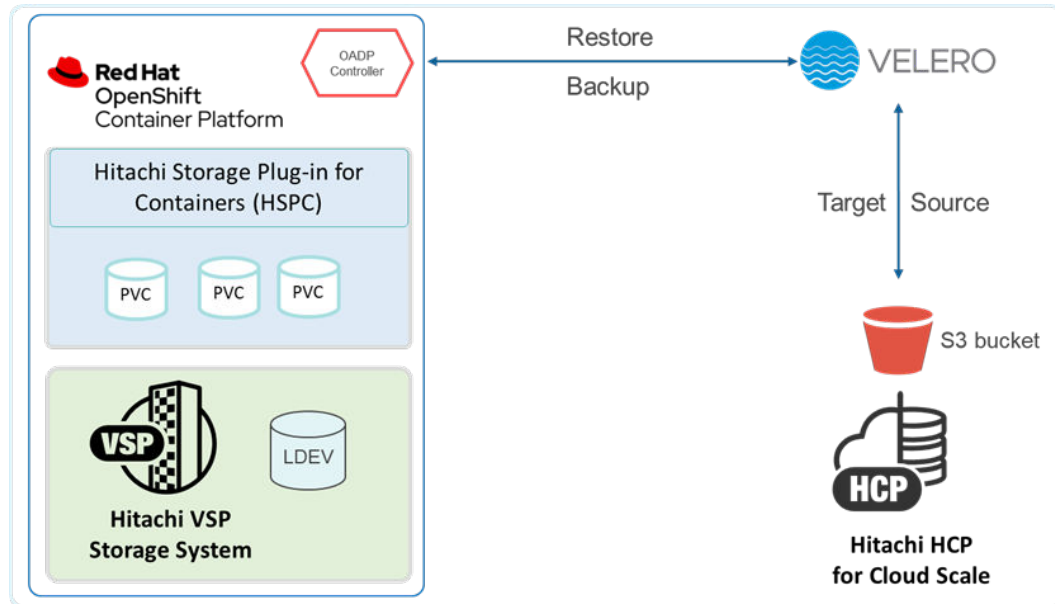
Solution Implementation and Validation

If you have followed the guidance in the Solution Design section and your infrastructure is prepared, you can try the example deployments. This reference architecture was validated by the following:

- Deploying the MySQL application with persistent volume using Helm and proving data protection functionality in OpenShift with OADP/Velero and Hitachi Content Platform for cloud scale. In this scenario, the backup/restore is performed leveraging HSPC CSI snapshots
- Deploying the Wordpress application with persistent volumes using Helm and proving data protection functionality in OpenShift with OADP/Velero and Hitachi Content Platform for cloud scale. In this scenario, the backup/restore is performed leveraging Restic, and the backup is taken in a primary OCP cluster and restored in a secondary OCP cluster. This process can be used to migrate cluster resources to other clusters
- Deploying and configuring Hitachi Replication Plug-in for Containers and validating replication of applications across datacenters.
- Installing and configuring Hitachi Storage Plug-in for Prometheus to monitor Kubernetes resources and Hitachi storage.
- Deploying a private container registry using Red Hat Quay and Hitachi HCP CS S3.

Persistent volume and data protection using Velero and Hitachi HCP CS S3 – Scenario 1

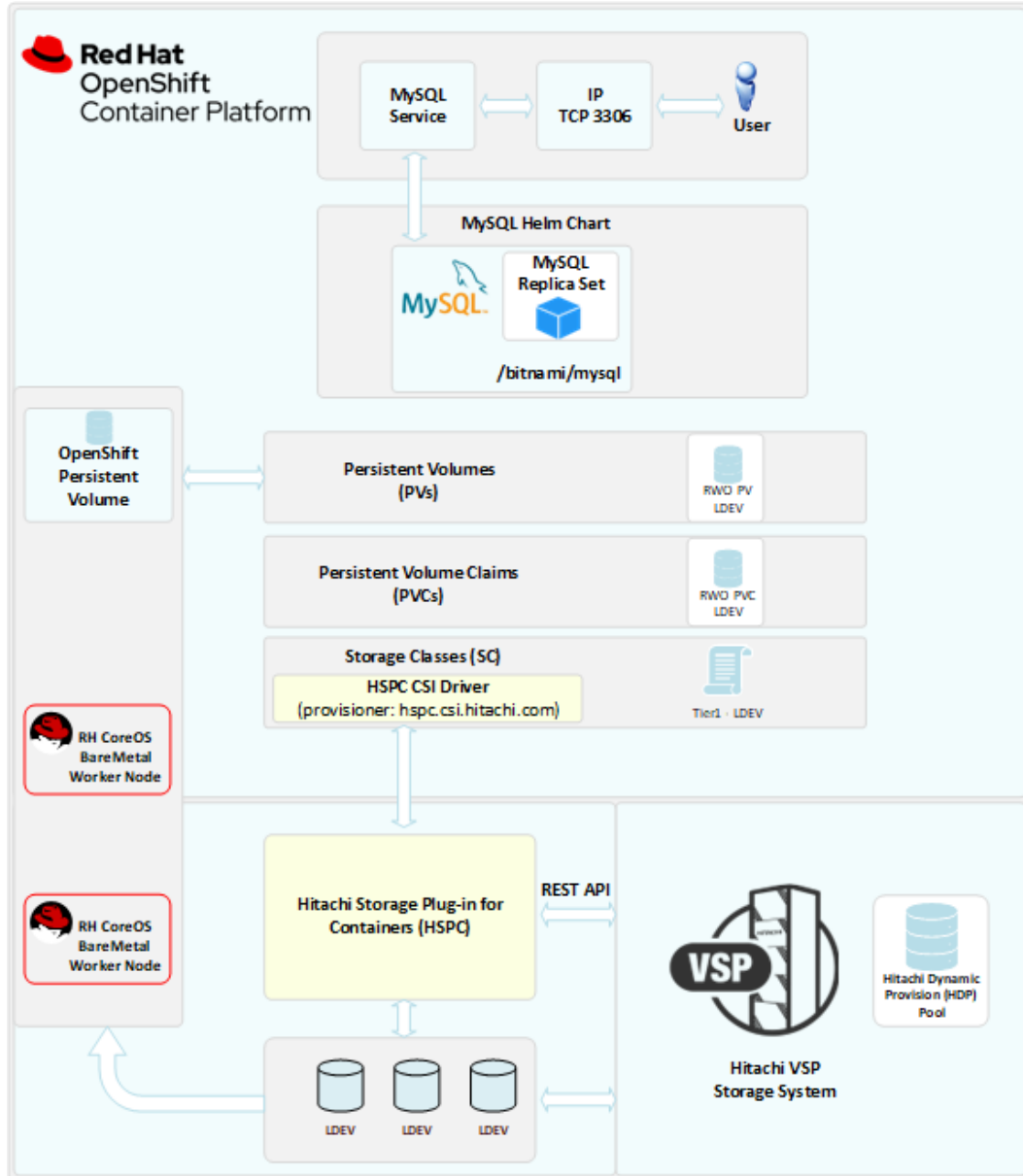
The following layout shows a high-level configuration of the setup to perform the backup and restore operation in the same OCP cluster using Velero and HSPC CSI snapshots.



Use the following procedures to:

- Deploy a MySQL database application with persistent volumes backed by Hitachi VSP storage system.
- Explore the backend storage resources mapping and allocations.
- Create a table and insert test data, back it up to HCP for cloud scale, delete it, and then restore the application from HCP for cloud scale using OADP/Velero Operator and verify the data.

The following figure shows the solution architecture with the solution to be validated.



Install and configure Helm utilities

```
[ocpusr@dminws-c2]$ oc get nodes
NAME                 STATUS    ROLES    AGE    VERSION
jpc2-master-1       Ready    master   147d   v1.22.5+a36406b
jpc2-master-2       Ready    master   147d   v1.22.5+a36406b
jpc2-master-3       Ready    master   147d   v1.22.5+a36406b
jpc2-worker-1       Ready    worker   147d   v1.22.5+a36406b
jpc2-worker-2       Ready    worker   147d   v1.22.5+a36406b
jpc2-worker-3       Ready    worker   147d   v1.22.5+a36406b
[ocpusr@dminws-c2]$
```

Helm allows you to install complex container-based applications easily with the ability to customize the deployment to your needs. On your Linux workstation, install the Helm binary by following the [Helm documentation for your distribution](#).

Add the *Bitnami* repository to your Helm configuration by running the following command:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Search for the MySQL Helm chart by running the following command:

```
helm search repo mysql
```

The following figure shows example output, showing that the Helm binary is installed properly and the *bitnami* repository has been added with a MySQL Helm chart available for use.

```
[ocpusr@dminws-c2]$ helm search repo mysql
NAME                CHART VERSION  APP VERSION  DESCRIPTION
bitnami/mysql       8.8.23         8.0.28      Chart to create a Highly
available MySQL cluster
```

Verify StorageClasses

The OCP cluster used for this test is a hybrid, containing some virtual worker nodes hosted on VMware ESXi and bare metal worker nodes. The bare metal worker nodes use FC HBAs and are connected to Hitachi VSP storage.

On this OCP cluster we have StorageClass using vSphere CSI provisioner, and other StorageClasses with Hitachi HSPC CSI provisioner. To verify the defined StorageClasses enter the command `oc get sc`.

For this example, we used the following StorageClass:

- One StorageClass `sc-vsp-113` for the primary MySQL DB instance.

The following figure shows an example listing of StorageClasses available on the OCP cluster.

```
[ocpusr@dminws-c2]# oc get sc
NAME                PROVISIONER                RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
csi-test-sc        csi.vsphere.vmware.com    Delete         Immediate          false                 1d
sc-vsp-113         hspc.csi.hitachi.com      Delete         Immediate          true                  1d
vsp-hrhc-sc        hspc.csi.hitachi.com      Delete         Immediate          true                  1d
[ocpusr@dminws-c2]#
```

Customize and deploy MySQL Helm chart with persistent storage

You can customize a Helm chart deployment by downloading the chart values to a YAML file and using that file during Helm chart installation. You can also specify the custom values for a deployment on the command line or in a script.

First, we create a namespace (or project) with the following command:

```
oc new-project demoapps
```


The following is the command and parameters used to deploy MySQL Helm chart:

```
helm install -n demoapps mysql-demo1 \
--set primary.nodeSelector.nodeType=hspc-fc \
--set global.storageClass=sc-vsp-113 \
--set primary.persistence.size=512Mi \
--set auth.rootPassword=Hitachi123,auth.database=hur_database \
--set secondary.replicaCount=0 \
--set primary.podSecurityContext.enabled=false \
--set primary.containerSecurityContext.enabled=false \
--set secondary.podSecurityContext.enabled=false \
--set secondary.containerSecurityContext.enabled=false \
bitnami/mysql
```

You must modify these values to match your environment and the StorageClass that are available in your OCP cluster. The values and their corresponding impact to the MySQL deployment are as follows:

- `nodeSelector` - Sets the node type where the pod must be deployed.
- `database` - Sets the name of the database to be created.
- `rootPassword` - Sets the password for the root user in the MySQL DB.
- `global.storageClass` - Sets the StorageClass to be used for the MySQL pod.
- `primary.persistence.size` - Sets the size of the persistent volume to be assigned to the MySQL DB.
- Set the SecurityContext parameters according to your environment.

Execute the command and Helm will begin to deploy your MySQL deployment to your OCP cluster.

The following figure shows output from Helm at the beginning of the deployment after the `install` command has been issued.

```
NAME: mysql-demo
LAST DEPLOYED: Thu Apr 21 15:16:33 2022
NAMESPACE: demoapps
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: mysql
CHART VERSION: 8.8.23
APP VERSION: 8.0.28

** Please be patient while the chart is being deployed **

Tip:

  Watch the deployment status using the command: kubectl get pods -w --namespace demoapps

Services:

  echo Primary: mysql-demo.demoapps.svc.cluster.local:3306

Execute the following to get the administrator credentials:

  echo Username: root
  MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace demoapps mysql-demo -o jsonpath="{.data.mysql-root-password}" |
  base64 --decode)

To connect to your database:

  1. Run a pod that you can use as a client:

      kubectl run mysql-demo-client --rm --tty -i --restart='Never' --image docker.io/bitnami/mysql:8.0.28-debian-10-r0
      --namespace demoapps --command -- bash

  2. To connect to primary service (read/write):

      mysql -h mysql-demo.demoapps.svc.cluster.local -uroot -p"$MYSQL_ROOT_PASSWORD"
```

You can monitor the MySQL deployment by viewing the resources in your *demoapps* namespace. Run the `oc get all -n demoapps` command to display the readiness of the pods, deployment, statefulsets, and services of the MySQL deployment.

The following figure shows an example of the output of this command for a fully running, healthy MySQL deployment.

```
[ocpusr@dminws-c2]$helm list -n demoapps
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/ocpininstall/ocp-upi-
install/auth/kubeconfig
NAME                REVISION    UPDATED                               STATUS    CHART
APP VERSION
mysql-demo          demoapps    1                                     2022-04-21 15:16:33.06346177 -0700 PDT  deployed  mysql-8.8.23
8.0.28
[ocpusr@dminws-c2]$

[ocpusr@dminws-c2]$oc get all -n demoapps
NAME                READY    STATUS    RESTARTS    AGE
pod/mysql-demo-0    1/1     Running   0           22m

NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/mysql-demo  ClusterIP     172.30.117.191 <none>         3306/TCP    22m
service/mysql-demo-headless ClusterIP     None          <none>         3306/TCP    22m

NAME                READY    AGE
statefulset.apps/mysql-demo 1/1     22m
[ocpusr@dminws-c2]$
[ocpusr@dminws-c2]$oc get pvc -n demoapps
NAME                STATUS    VOLUME                                     CAPACITY    ACCESS MODES    STORAGECLASS    AGE
data-mysql-demo-0  Bound    pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b  512Mi       RWO             sc-vsp-113      22m
[ocpusr@dminws-c2]$
```

Verify the MySQL deployment and insert some data

In the previous step you displayed the resources in the *demoapps* namespace, which included the output of the MySQL pod. The next step is to connect to the MySQL DB and insert some data for test.

Procedure

1. Inserting test data into the MySQL database .

Once the MySQL pod is ready, a new table called `replication_cr_status` is created on the `hur_database` database. Then a few records of test data are inserted into this new table.

The following shows the commands used to connect to the MySQL DB to create a table and insert test data.

```
[ocpusr@adminws-c2]$oc exec -it mysql-demo-0 -n demoapps -- bash
1000710000@mysql-demo-0:/$
1000710000@mysql-demo-0:/$ mysql -h mysql-demo.demoapps.svc.cluster.local -uroot -p"Hitachi123"
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| hur_database |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> use hur_database;
Database changed

mysql> use hur_database;
Database changed
mysql> CREATE TABLE replication_cr_status (Status VARCHAR(32) PRIMARY KEY, Description VARCHAR(100), HUR_pair_state VARCHAR(32));
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO replication_cr_status (Status, Description, HUR_pair_state)
-> VALUES ("Pending", "Accepted CR creation command", "NA"),
-> ("WaitingPrimarySite", "Waiting for primary site to finish HUR operations", "NA"),
-> ("Copying", "Initial copy is in progress", "COPY"),
-> ("Ready", "Successfully created HUR pair and the state is PAIR", "PAIR"),
-> ("Split", "Successfully splitted HUR pair and the state is PSUS or SSUS", "PSUS or SSUS"),
-> ("Failover", "Successfully failover HUR pair", "SSWS"),
-> ("Error", "Having errors in HUR pair", "PSUE, PFUL, or PFUS"),
-> ("Unknown", "A pair does not exist or unexpected pair state", "Other than the above");
Query OK, 8 rows affected (0.01 sec)
Records: 8 Duplicates: 0 Warnings: 0
```

2. Verify the data that has been inserted into the MySQL database with the purpose to test and verify backup and restore of persistent volumes:

```
mysql> SELECT * FROM replication_cr_status;
+-----+-----+-----+
| Status | Description | HUR_pair_state |
+-----+-----+-----+
| Copying | Initial copy is in progress | COPY |
| Error | Having errors in HUR pair | PSUE, PFUL, or PFUS |
| Failover | Successfully failover HUR pair | SSWS |
| Pending | Accepted CR creation command | NA |
| Ready | Successfully created HUR pair and the state is PAIR | PAIR |
| Split | Successfully splitted HUR pair and the state is PSUS or SSUS | PSUS or SSUS |
| Unknown | A pair does not exist or unexpected pair state | Other than the above |
| WaitingPrimarySite | Waiting for primary site to finish HUR operations | NA |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

Explore backend storage resource mapping and allocations

When modifying the Helm chart values for the MySQL deployment, you provided a StorageClass that mapped back to Hitachi VSP for persistent volume allocation to the MySQL pod. You can follow the storage paths from the OCP persistent volume layer to the Hitachi VSP layer. Complete the following procedures to validate the storage path from the running pods to the allocated backend storage.

Verify the MySQL persistent volume data path

Starting at the OCP cluster layer, you can explore the PVC and corresponding PVs that were provisioned by the Hitachi HSPC CSI driver following the procedure:

Procedure

1. To list the PVCs created during the MySQL Helm chart deployment, run the following command:

```
oc get pvc -n demoapps
```

```
[ocpusr@dminws-c2]$oc get pvc -n demoapps
NAME                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-mysql-demo-0   Bound    pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b  512Mi      RWO             sc-vsp-113
[ocpusr@dminws-c2]$
```

2. Get details on the PVC by running the following command:

```
oc describe pvc data-mysql-demo-0 -n demoapps
```

The following figure shows the output of this command, including the details of the persistent volume claim and the access mode that was specified during Helm chart deployment.

```
[ocpusr@dminws-c2]$oc get pvc -n demoapps
NAME                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-mysql-demo-0   Bound    pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b  512Mi      RWO             sc-vsp-113
[ocpusr@dminws-c2]$

[ocpusr@dminws-c2]$oc describe pvc data-mysql-demo-0 -n demoapps
Name:                data-mysql-demo-0
Namespace:           demoapps
StorageClass:        sc-vsp-113
Status:              Bound
Volume:              pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b
Labels:              app.kubernetes.io/component=primary
                    app.kubernetes.io/instance=mysql-demo
                    app.kubernetes.io/name=mysql
Annotations:         pv.kubernetes.io/bind-completed: yes
                    pv.kubernetes.io/bound-by-controller: yes
                    volume.beta.kubernetes.io/storage-provisioner: hspc.csi.hitachi.com
Finalizers:          [kubernetes.io/pvc-protection]
Capacity:            512Mi
Access Modes:        RWO
VolumeMode:          Filesystem
Used By:             mysql-demo-0
Events:              
```

3. Note the volume identifier and copy it for the next step.
4. Now that you have viewed the details of the PVC for the MySQL DM, explore the associated PV to the claim by running the following command, entering your Volume identifier from the previous step as the PV ID:

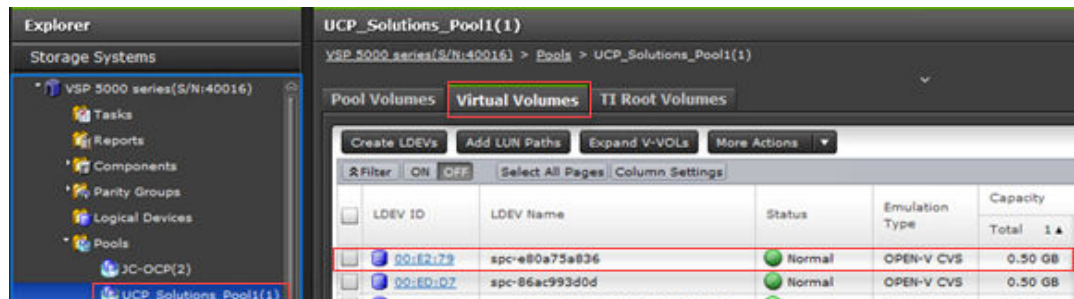
```
oc describe pv <PV ID>
```

The following figure shows the output of this command, including the details of the PV created for the PVC. On the VolumeAttributes note the volume nickname for the next step.

```

[ocpusr@minws-c2] $oc describe pv pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b
Name:          pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: hspc.csi.hitachi.com
Finalizers:    [kubernetes.io/pv-protection external-attacher/hspc-csi-hitachi-com]
StorageClass:  sc-vsp-113
Status:        Bound
Claim:         demoapps/data-mysql-demo-0
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:   Filesystem
Capacity:     512Mi
Node Affinity: <none>
Message:
Source:
  Type:        CSI (a Container Storage Interface (CSI) volume source)
  Driver:      hspc.csi.hitachi.com
  FSType:      ext4
  VolumeHandle: 60060e80089c500000509c500000e279--spc-e80a75a836
  ReadOnly:    false
  VolumeAttributes:
    autoHG=true
    connectionType=fc
    hostModeOption=91
    ldevIDDec=57977
    ldevIDHex=E2:79
    mode=normal
    nickname=spc-e80a75a836
    ports=CL1-D,CL4-D
    size=512Mi
    storage.kubernetes.io/csiProvisionerIdentity=1642637755977-8081-hspc.csi.hitachi.com
Events:        <none>
[ocpusr@minws-c2] $
  
```

5. Open the Storage Navigator and connect to the VSP storage system.
6. In the left pane, expand **Pools**, and then click on Pool #1 which is the one specified in the `sc-vsp-113` StorageClass.
7. Click **Virtual Volumes**. You will see container volumes provisioned to your cluster in the right pane.
8. Find the volume that matches the volume nickname from the previous step.



In this way, we can trace the full data path from the OCP cluster to the Hitachi VSP storage system.

Back up the MySQL application using Red Hat OADP/Velero and HSPC CSI snapshots

Follow this procedure to create and verify the status of the backup:

Procedure

1. From your Linux workstation, log in to your OCP cluster.

2. Create a backup custom resource (CR). Here an example of the backup CR used to backup MySQL DB:

```
cat demoapps-backup-csi.yaml
```

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  namespace: openshift-adp
  name: demoapps-backup-csi
  labels:
    velero.io/storage-location: default
spec:
  includedNamespaces:
  - demoapps
```

3. Issue the following command to back up the MySQL application.

```
oc apply -f demoapps-backup-csi.yaml
```

4. The following command will list all the backups.

```
[ocpusr@dminws-c2]$oc get backups -n openshift-adp
NAME                                AGE
demoapps-backup-csi                68s
```

5. When the Velero backup is initiated, you can run the `oc describe backup demoapps-backup-csi` command to show the progress of the Velero backup.

Verify the VolumeSnapshots and VolumeSnapshotContent created during the backup

```
[[ocpusr@dminws-c2]$oc describe backup demoapps-backup-csi
Name:          demoapps-backup-csi
Namespace:     openshift-adp
Labels:        velero.io/storage-location=velero-ocp-hcpcs-1
Annotations:   velero.io/source-cluster-k8s-gitversion: v1.21.6+b4b4813
               velero.io/source-cluster-k8s-major-version: 1
               velero.io/source-cluster-k8s-minor-version: 21
API Version:   velero.io/v1
Kind:          Backup
Metadata:
  Creation Timestamp: 2022-04-21T22:47:08Z
...
  Manager:          velero-server
  Operation:        Update
  Time:             2022-04-21T22:47:53Z
  Resource Version: 98000975
  UID:              974ca667-208e-4c6b-a4d6-c8244d2876f5
Spec:
  Default Volumes To Restic: false
  Included Namespaces:
    demoapps
  Storage Location: velero-ocp-hcpcs-1
  Ttl:               720h0m0s
  Volume Snapshot Locations:
    velero-ocp-hcpcs-1
Status:
  Completion Timestamp: 2022-04-21T22:47:48Z
  Expiration:          2022-05-21T22:47:08Z
  Format Version:      1.1.0
  Phase:               Completed
Progress:
  Items Backed Up:    58
  Total Items:        58
  Start Timestamp:    2022-04-21T22:47:08Z
  Version:            1
  Warnings:           1
Events:               <none>
```

Verify the VolumeSnapshots and VolumeSnapshotContent created during the backup

Because we are using Hitachi HSPC which supports CSI snapshots, the backup CR creates backup files for Kubernetes resources and internal images on the S3 object storage, and creates snapshots for the persistent volumes (PVs). The CSI snapshot controller binds VolumeSnapshot and VolumeSnapshotContent objects.

The following procedure can trace all the VolumeSnapshots and VolumeSnapshotContent back to the Hitachi VSP storage system.

Procedure

1. Run the `oc get volumesnapshots -n demoapps` command to list the volumesnapshots created during the backup process. In the output below we can see the MySQL PVC, the name of the volume snapshot, the volume snapshot class, and the associated volume snapshot content name.

```
[[ocpusr@dminws-c2]$oc get volumesnapshots -n demoapps
NAMESPACE NAME          READYTOUSE SOURCEPVC          SOURCESNAPSHOTCONTENT  RESTORESIZ
SNAPSHOTCLASS  SNAPSHOTCONTENT
demoapps [velero-data-mysql-demo-0-nlfzb] true [data-mysql-demo-0]
volume-snapshot-class-csi-hspc snapcontent-0698103a-daac-41f0-ae5a-72710372aff0 8h 512Mi
[[ocpusr@dminws-c2]$
```

Verify the VolumeSnapshots and VolumeSnapshotContent created during the backup

- The following command shows additional details of the volume snapshot. We can see the label corresponding to the name of the backup *demoapps-backup-csi* and the MySQL PVC called *data-mysql-demo-0*.

```
[ocpusr@dminws-c2]$oc describe volumesnapshot velero-data-mysql-demo-0-nlfzb -n demoapps
Name:          velero-data-mysql-demo-0-nlfzb
Namespace:    demoapps
Labels:       velero.io/backup-name=demoapps-backup-csi
Annotations:  <none>
API Version:  snapshot.storage.k8s.io/v1
Kind:         VolumeSnapshot
Metadata:
  Creation Timestamp:  2022-04-21T22:47:37Z
  Finalizers:
    snapshot.storage.kubernetes.io/volumesnapshot-as-source-protection
    snapshot.storage.kubernetes.io/volumesnapshot-bound-protection
  Generate Name:      velero-data-mysql-demo-0-
  Generation:         1
  Managed Fields:
    API Version:  snapshot.storage.k8s.io/v1beta1
    Fields Type:  FieldsV1
    ...
    Manager:      velero-plugin-for-csi
    Operation:    Update
    Time:         2022-04-21T22:47:37Z
    API Version:  snapshot.storage.k8s.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      ...
    Manager:      snapshot-controller
    Operation:    Update
    Time:         2022-04-21T22:47:46Z
  Resource Version:  98000864
  UID:              0698103a-daac-41f0-ae5a-72710372aff0
Spec:
  Source:
    Persistent Volume Claim Name:  data-mysql-demo-0
  Volume Snapshot Class Name:     volume-snapshot-class-csi-hspc
Status:
  Bound Volume Snapshot Content Name:  snapcontent-0698103a-daac-41f0-ae5a-72710372aff0
  Creation Time:                      2022-04-21T14:44:59Z
  Ready To Use:                       true
  Restore Size:                       512Mi
```

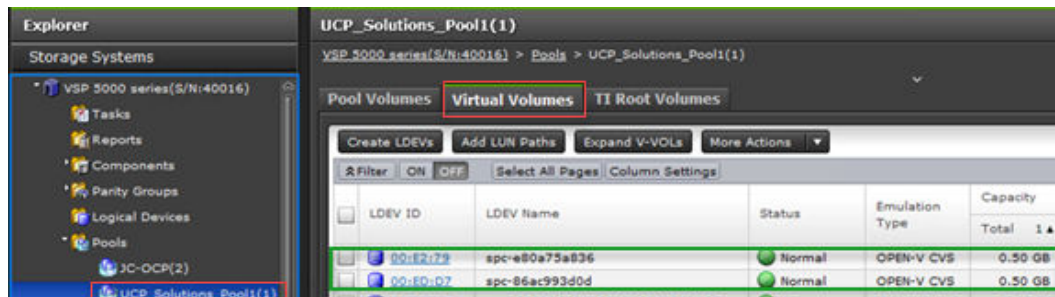
- Run the `oc get volumesnapshotcontent` command to list the volume snapshot contents. On the list we can identify the volume snapshot associated with the volume snapshot described previously. Also, we can see the Hitachi HSPC CSI driver associated with this volume.

```
[ocpusr@dminws-c2]$oc get volumesnapshotcontent
NAME                                READYTOUSE  RESTORESIZE  DELETIONPOLICY  DRIVER
VOLUMESNAPSHOTCLASS  VOLUMESNAPSHOT
snapcontent-0698103a-daac-41f0-ae5a-72710372aff0  true        536870912    Delete          hspc.csi.hitachi.com
volume-snapshot-class-csi-hspc  velero-data-mysql-demo-0-nlfzb  2m54s
```

- If we use the `describe` command, we can see more details for the volume snapshot content and trace back to the Hitachi VSP LDEV created.


```
[ocpusr@dm1nws-c2] $oc describe volumesnapshotcontent snapcontent-0698103a-daac-41f0-ae5a-72710372aff0
Name: snapcontent-0698103a-daac-41f0-ae5a-72710372aff0
Namespace:
Labels: velero.io/backup-name=demoapps-backup-csi
Annotations: snapshot.storage.kubernetes.io/deletion-secret-name: secret-vsp-113
              snapshot.storage.kubernetes.io/deletion-secret-namespace: test
API Version: snapshot.storage.k8s.io/v1
Kind: VolumeSnapshotContent
Metadata:
  Creation Timestamp: 2022-04-21T22:47:37Z
  Finalizers:
    snapshot.storage.kubernetes.io/volumesnapshotcontent-bound-protection
  Generation: 1
  Managed Fields:
    API Version: snapshot.storage.k8s.io/v1
    Fields Type: FieldsV1
    ...
    Manager: snapshot-controller
    Operation: Update
    Time: 2022-04-21T22:47:37Z
    API Version: snapshot.storage.k8s.io/v1
    Fields Type: FieldsV1
    ...
    Manager: csi-snapshotter
    Operation: Update
    Time: 2022-04-21T22:47:46Z
    API Version: snapshot.storage.k8s.io/v1beta1
    ...
    Manager: velero-plugin-for-csi
    Operation: Update
    Time: 2022-04-21T22:47:47Z
  Resource Version: 98000881
  UID: 50d16e5e-3e49-422b-9441-e99b78ea05c9
Spec:
  Deletion Policy: Delete
  Driver: hspc.csi.hitachi.com
  Source:
    Volume Handle: 60060e80089c500000509c500000e279--spc-e80a75a836
  Volume Snapshot Class Name: volume-snapshot-class-csi-hspc
  Volume Snapshot Ref:
    API Version: snapshot.storage.k8s.io/v1
    Kind: VolumeSnapshot
    Name: velero-data-mysql-demo-0-nlfzb
    Namespace: demoapps
    Resource Version: 98000710
    UID: 0698103a-daac-41f0-ae5a-72710372aff0
Status:
  Creation Time: 1650552299000000000
  Ready To Use: true
  Restore Size: 536870912
  Snapshot Handle: 60060e80089c500000509c500000edd7--spc-86ac993d0d
Events: <none>
```

5. Open Storage Navigator and connect to the VSP storage system.
6. In the left pane expand **Pools**, and then click on Pool #1 which is the one specified in the *sc-vsp-113* StorageClass.
7. Click on **Virtual Volumes**. You will see container volumes; LDEV *spc-e80a75a836* corresponds to the MySQL volume and LDEV *spc-86ac993d0d* corresponds to the snapshot.

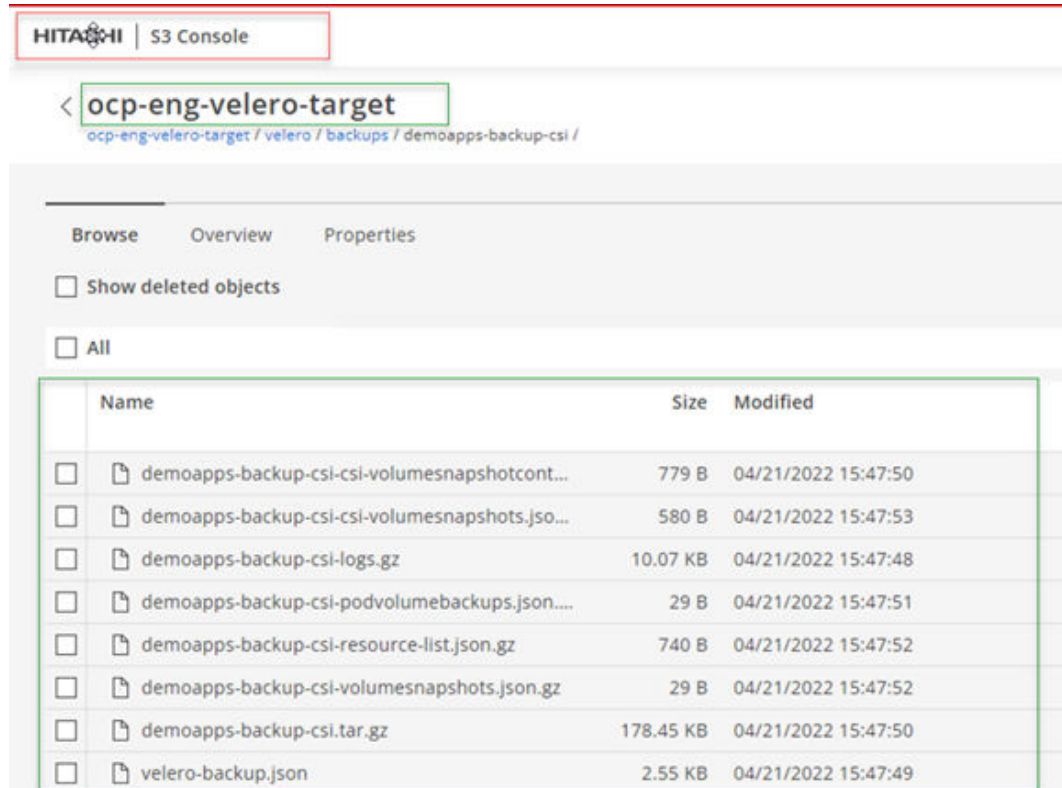


Explore the Hitachi Content Platform for cloud scale S3 object store

Follow this procedure to explore the data on the HCP CS S3 bucket.

Procedure

1. Open a web browser, navigate to your Hitachi Content Platform for cloud scale web interface, and log in.
2. Browse the contents of your bucket, locate the `ocp-eng-velero-target` folder, and open it. You will see the contents of all the Kubernetes objects that were backed up during the Velero backup, with the exception of the PVs and their data.

**Delete the MySQL application from OCP cluster**

Follow this procedure to delete MySQL application and its persistent volume.

Procedure

1. Log in to your OCP cluster from your Linux workstation.
2. Run the `helm delete mysql-demo -n demoapps` command to remove the MySQL application.

```
[ocpusr@dminws-c2]$ helm list -n demoapps
NAME      NAMESPACE   REVISION   UPDATED                               STATUS   CHART   APP VERSION
mysql-demo  demoapps     1          2022-04-21 15:16:33.06346177 -0700 PDT  deployed 8.0.28

[ocpusr@dminws-c2]$ helm delete mysql-demo -n demoapps
release "mysql-demo" uninstalled
[ocpusr@dminws-c2]$

[ocpusr@dminws-c2]$ helm list -n demoapps
NAME      NAMESPACE   REVISION   UPDATED                               STATUS   CHART   APP VERSION
[ocpusr@dminws-c2]$

[ocpusr@dminws-c2]$ oc get all -n demoapps
No resources found in demoapps namespace.
```

3. When the application is removed, delete PVCs and PVs from the cluster.

```
[ocpusr@dminws-c2]$oc get pvc -n demoapps
NAME                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-mysql-demo-0   Bound    pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b  512Mi      RWO            sc-vsp-113
[ocpusr@dminws-c2]$
[ocpusr@dminws-c2]$oc delete pvc data-mysql-demo-0 -n demoapps
persistentvolumeclaim "data-mysql-demo-0" deleted
[ocpusr@dminws-c2]$
[ocpusr@dminws-c2]$oc get pvc -n demoapps
No resources found in demoapps namespace.
[ocpusr@dminws-c2]$

[ocpusr@dminws-c2]$oc delete pv pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b
persistentvolume "pvc-d3997258-fbd7-468e-8b3f-ccc2bb94954b" deleted
```

Restore the MySQL application using the Velero operator

Follow this procedure to restore an application from backup.

Procedure

1. Run the `velero backup get` command to list the backups in the Velero database. You can see the backups of the MySQL application that you took previously.

```
[ocpusr@dminws-c2]$oc get backups
NAME                AGE
demoapps-backup-csi  53m
```

2. Create a restore custom resource (CR). The following is an example of restore CR used to restore MySQL from `demoapps-backup-csi` backup:

```
cat demoapps-restore-csi.yaml
```

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  namespace: openshift-adp
  name: demoapps-restore-csi
spec:
  backupName: demoapps-backup-csi
  includedNamespaces:
  - demoapps
```

3. Run the following command to begin the restore of the MySQL application and its associated PVs and data:

```
oc apply -f demoapps-restore-csi.yaml
```

4. The following command lists the restores:

```
[ocpusr@dminws-c2]$oc get restores
NAME                AGE
demoapps-restore-csi  5s
```

5. After the Velero restore is submitted, run the `oc describe restore demoapps-restore-csi` command to view the progress of the restore. Note that the restore will

not show as completed until the application and all of its PVs and associated data are restored.

```
[ocpusr@dmnws-c2]$oc describe restore demoapps-restore-csi
Name:          demoapps-restore-csi
Namespace:     openshift-adp
Labels:        <none>
Annotations:   <none>
API Version:   velero.io/v1
Kind:          Restore
Metadata:
  Creation Timestamp:  2022-04-21T23:41:58Z
  Generation:         6
  Managed Fields:
    API Version:  velero.io/v1
    Fields Type:  FieldsV1
    ...
    Manager:      velero-server
    Operation:    Update
    Time:         2022-04-21T23:42:03Z
  Resource Version:  98054689
  UID:              deede975-70e1-4394-a89e-fefc6def7961
Spec:
  Backup Name:  demoapps-backup-csi
  Excluded Resources:
    nodes
    events
    events.events.k8s.io
    backups.velero.io
    restores.velero.io
    resticrepositories.velero.io
  Included Namespaces:
    demoapps
Status:
  Completion Timestamp:  2022-04-21T23:42:03Z
  Phase:                 Completed
  Progress:
    Items Restored:  41
    Total Items:     41
  Start Timestamp:     2022-04-21T23:41:58Z
  Warnings:            5
  Events:              <none>
```

6. Run the following commands to view all of the MySQL components that were restored.

```
[ocpusr@minws-c2]$helm list -n demoapps
NAME                NAMESPACE    REVISION    UPDATED                               STATUS    CHART
mysql-demo          demoapps      1           2022-04-21 15:16:33.06346177 -0700 PDT  deployed  mysql-8.8.23
8.0.28
[ocpusr@minws-c2]$

[ocpusr@minws-c2]$oc get all -n demoapps
NAME                READY    STATUS    RESTARTS    AGE
pod/mysql-demo-0    1/1     Running   0            109s

NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/mysql-demo  ClusterIP   172.30.29.152 <none>         3306/TCP    109s
service/mysql-demo-headless ClusterIP    None          <none>         3306/TCP    109s

NAME                READY    AGE
statefulset.apps/mysql-demo  1/1     109s
[ocpusr@minws-c2]$

[ocpusr@minws-c2]$oc get pvc -n demoapps
NAME                STATUS    VOLUME                                     CAPACITY    ACCESS MODES    STORAGECLASS    AGE
data-mysql-demo-0  Bound    pvc-32ff78ce-37ba-4103-9255-a0eb61e92835  512Mi       RWO              sc-vsp-113      2m6s
[ocpusr@minws-c2]$

[ocpusr@minws-c2]$oc get pv
NAME                CAPACITY    ACCESS MODES    RECLAIM POLICY    STATUS    CLAIM
pvc-32ff78ce-37ba-4103-9255-a0eb61e92835  512Mi       RWO              Delete            Bound    demoapps/data-mysql-demo-0
sc-vsp-113      2m5s
```

Verify MySQL persistent data

The following shows the commands used to connect to the restore MySQL DB and verify the data that was inserted before the backup. You will see all the records created before deleting the MySQL application.

```
[ocpusr@minws-c2]$oc exec -it mysql-demo-0 -n demoapps -- bash
1000710000@mysql-demo-0:/$ mysql -h mysql-demo.demoapps.svc.cluster.local -uroot -p"Hitachi123"
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 49
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| hur_database |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> use hur_database;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

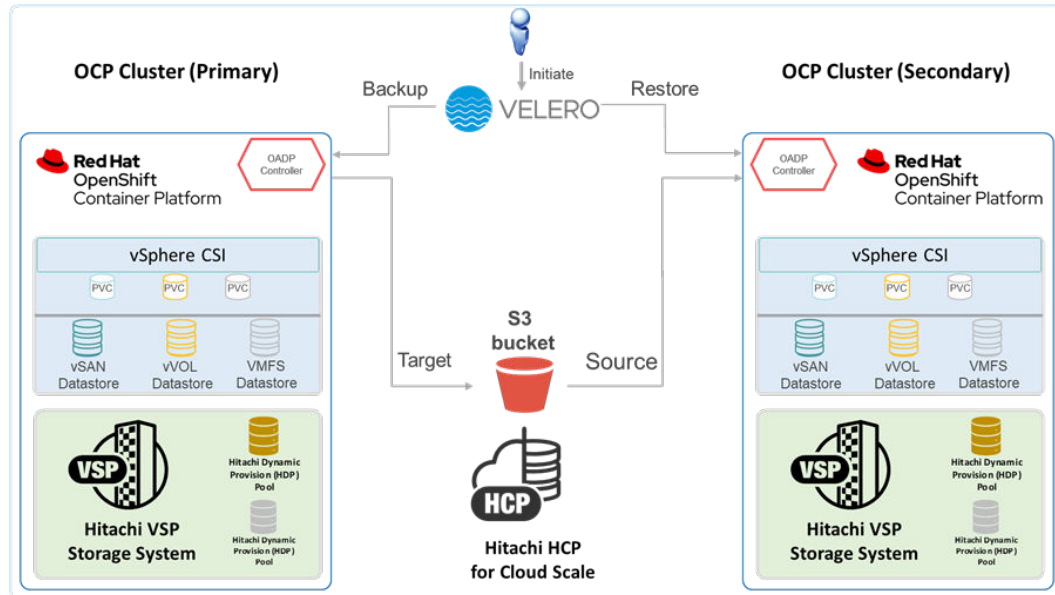
Database changed

mysql> SELECT * FROM replication_cr_status;
+-----+-----+-----+
| Status | Description | HUR_pair_state |
+-----+-----+-----+
| Copying | Initial copy is in progress | COPY |
| Error | Having errors in HUR pair | PSUE, PFUL, or PFUS |
| Failover | Successfully failover HUR pair | SSWS |
| Pending | Accepted CR creation command | NA |
| Ready | Successfully created HUR pair and the state is PAIR | PAIR |
| Split | Successfully splitted HUR pair and the state is PSUS or SSUS | PSUS or SSUS |
| Unknown | A pair does not exist or unexpected pair state | Other than the above |
| WaitingPrimarySite | Waiting for primary site to finish HUR operations | NA |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```


Persistent volume and data protection using Velero and Hitachi HCP CS S3 – Scenario 2

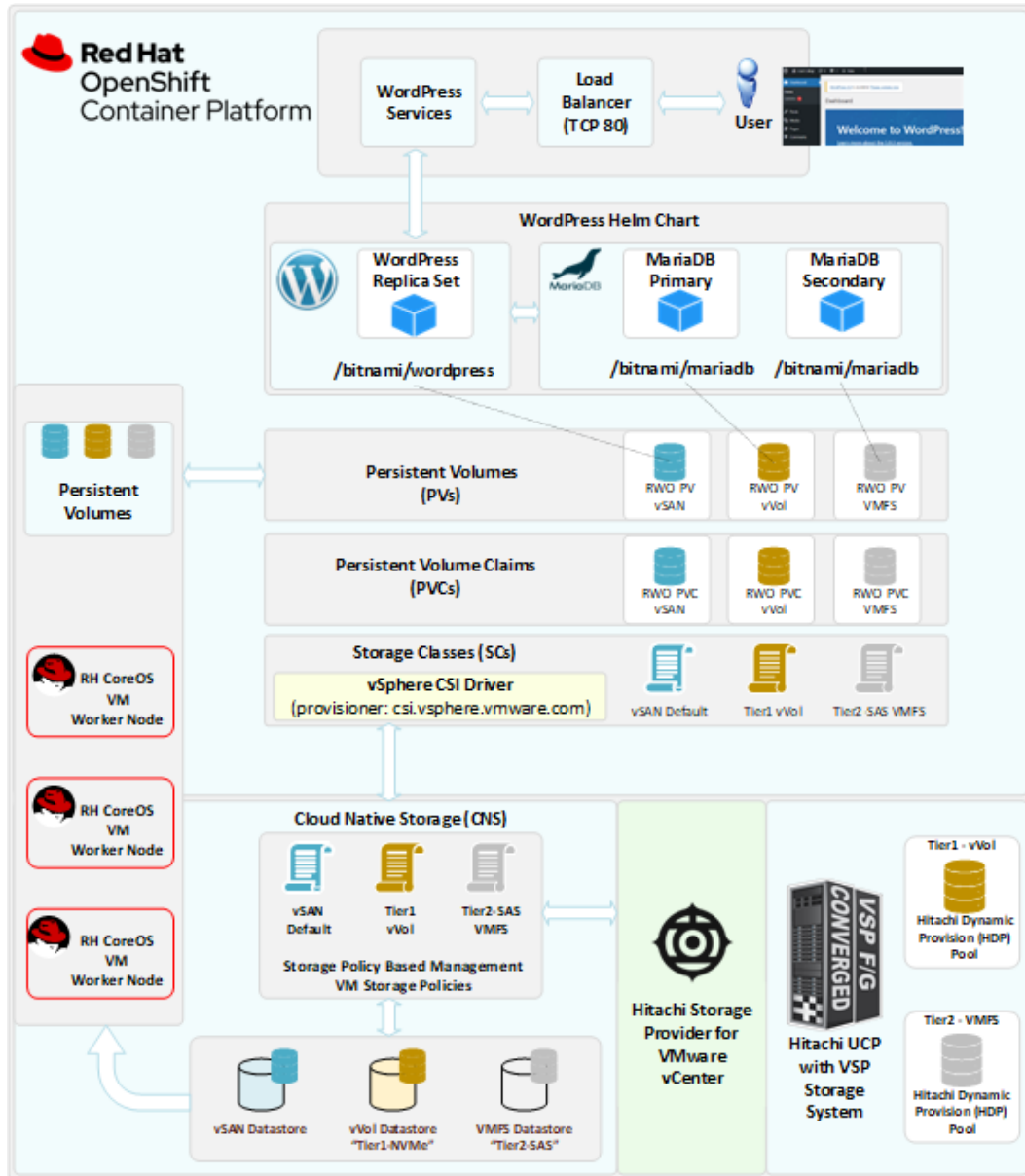
The following layout shows a high-level configuration of the setup to perform the backup and restore operations between primary and secondary OCP clusters using Velero and Restic.



Use the following procedures to:

- Deploy a Wordpress database application with persistent volumes backed by Hitachi VSP storage systems and using different SPBM policies.
- Explore the vSphere CNS layer.
- Modify the Wordpress application in the primary OCP cluster, back it up to HCP for cloud scale, delete it, and then restore the application to a secondary OCP cluster from HCP for cloud scale using OADP/Velero Operator and verify the data.

The following figure shows the solution architecture with the solution to be validated.



Install and configure Helm utilities

The installation and backup process are executed in the primary cluster called *jpc2*. The following shows the master and worker nodes corresponding to this hybrid OCP cluster (virtual and bare metal worker nodes).

```
[ocpusr@dminws-c2]$ oc get nodes
NAME                 STATUS    ROLES    AGE   VERSION
jpc2-master-1       Ready    master   147d  v1.22.5+a36406b
jpc2-master-2       Ready    master   147d  v1.22.5+a36406b
jpc2-master-3       Ready    master   147d  v1.22.5+a36406b
jpc2-worker-1       Ready    worker   147d  v1.22.5+a36406b
jpc2-worker-2       Ready    worker   147d  v1.22.5+a36406b
jpc2-worker-3       Ready, SchedulingDisabled  worker   146d  v1.22.5+a36406b
[ocpusr@dminws-c2]$
```

Helm allows you to install complex container-based applications easily with the ability to customize the deployment to your needs. On your Linux workstation, install the Helm binary by following the [Helm documentation for your distribution](#).

Add the *Bitnami* repository to your Helm configuration by running the following command:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Search for the MySQL Helm chart by running the following command:

```
helm search repo mysql
```

The following figure shows example output, showing that the Helm binary is installed properly and the *bitnami* repository has been added with a MySQL Helm chart available for use.

```
[ocpusr@dminws-c2]$ helm search repo wordpress
NAME                CHART VERSION   APP VERSION     DESCRIPTION
bitnami/wordpress   13.0.11         5.9.0           WordPress is the world's most
popular blogging ...
bitnami/wordpress-intel 0.1.4          5.9.0           WordPress for Intel is the
most popular bloggin...
```

Verify StorageClasses

The OCP cluster used for this test is a hybrid, containing some virtual worker nodes hosted on VMware ESXi and bare metal worker nodes. The virtual worker nodes are running on a VMware vSphere cluster. The VMware cluster has been configured with different SPBM policies that allow placement of the Persistent Volumes (PVs) into different types of storage (vSAN, vVols, VMFS).

On this OCP cluster we have StorageClass using vSphere CSI provisioner, and other StorageClasses with Hitachi HSPC CSI provisioner. To verify the defined StorageClasses enter the command `oc get sc`.

For this example, we used the following three StorageClasses:

- One for the frontend Wordpress pod
- One for the primary MariaDB instance
- One for the secondary MariaDB instance

The following figure shows an example listing of StorageClasses available on the OCP cluster.

```
[ocpusr@dminws-c2]$ oc get sc
NAME                PROVISIONER           RECLAIMPOLICY   VOLUMEBINDINGMODE
csi-test-sc         csi.vsphere.vmware.com Delete           Immediate
hitachi-vmfs-tier2-sc csi.vsphere.vmware.com Delete           Immediate
hitachi-vvol-tier1-sc csi.vsphere.vmware.com Delete           Immediate
sc-vsp-113          hspc.csi.hitachi.com Delete           Immediate
vsan-test-sc        csi.vsphere.vmware.com Delete           Immediate
vsp-hrpc-sc         hspc.csi.hitachi.com Delete           Immediate
[ocpusr@dminws-c2]$
```


Customize and deploy Wordpress Helm chart with persistent storage

You can customize a Helm chart deployment by downloading the chart values to a YAML file and using that file during Helm chart installation. You can also specify the custom values for a deployment on the command line or in a script.

First, we create a namespace (or project) with the following command:

```
oc new-project demovelero
```

The following is the command and parameters used to deploy Wordpress Helm chart:

```
helm install -n demovelero wordpress \
--set wordpressUsername=admin \
--set wordpressPassword=wordpress \
--set replicaCount=1 \
--set persistence.storageClass=vSAN-test-sc \
--set persistence.size=200Mi \
--set mariadb.architecture=replication \
--set mariadb.primary.persistence.storageClass=hitachi-vvol-tier1-sc \
--set mariadb.primary.persistence.size=256Mi \
--set mariadb.secondary.persistence.storageClass=hitachi-vmfs-tier2-sc \
--set mariadb.secondary.persistence.size=256Mi \
bitnami/wordpress
```

You must modify these values to match your environment and the StorageClass(es) that are available in your OCP cluster. The values and their corresponding impact to the Wordpress deployment are as follows:

- `wordpressUsername` - Sets the admin username for the Wordpress application.
- `wordpressPassword` - Sets the password for the admin user in the Wordpress application.
- `replicaCount` - Configures the number of frontend Wordpress pods.
- `persistence.storageClass` - Sets the StorageClass to be used for the frontend Wordpress pods.
- `persistence.size` - Sets the size of the persistent volume to be assigned to the frontend Wordpress pods.
- `mariadb.architecture` - Indicates whether Helm should deploy a single backend database (standalone, single pod) or a high-availability backend database (replication, two pods).
- `mariadb.primary.persistence.storageClass` - Sets the StorageClass to be used for the primary MariaDB instance.
- `mariadb.primary.persistence.size` - Sets the size of the persistent volume to be assigned to the primary MariaDB instance.
- `mariadb.secondary.persistence.storageClass` - Sets the StorageClass to be used for the secondary MariaDB instance.

Set the SecurityContext in your OCP cluster according to your environment and security requirements.

Execute the command and Helm will begin to deploy your Wordpress deployment to your OCP cluster.

The following figure shows output from Helm at the beginning of the deployment after the `install` command has been issued.

You can monitor the Wordpress deployment by viewing the resources in your `demoapps` namespace. Run the `oc get all -n demoapps` command to display the readiness of the pods, deployment, statefulsets, and services of the Wordpress deployment.

The following figure shows an example of the output of this command for a fully running, healthy Wordpress deployment.

```
[ocpusr@dminws-c2]$ oc get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/wordpress-679f4ff79b-thrh8     1/1    Running   0           70s
pod/wordpress-mariadb-primary-0    1/1    Running   0           12m
pod/wordpress-mariadb-secondary-0  1/1    Running   0           12m

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)
service/wordpress                   LoadBalancer        172.30.155.130  <pending>        80:31422/TCP,443:31943/TCP
service/wordpress-mariadb-primary   ClusterIP           172.30.152.174  <none>           3306/TCP
service/wordpress-mariadb-secondary ClusterIP           172.30.171.249  <none>           3306/TCP

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wordpress           1/1     1             1           12m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/wordpress-679f4ff79b 1         1         1       12m

NAME                                READY   AGE
statefulset.apps/wordpress-mariadb-primary 1/1     12m
statefulset.apps/wordpress-mariadb-secondary 1/1     12m
[ocpusr@dminws-c2]$
```

Verify the Wordpress deployment

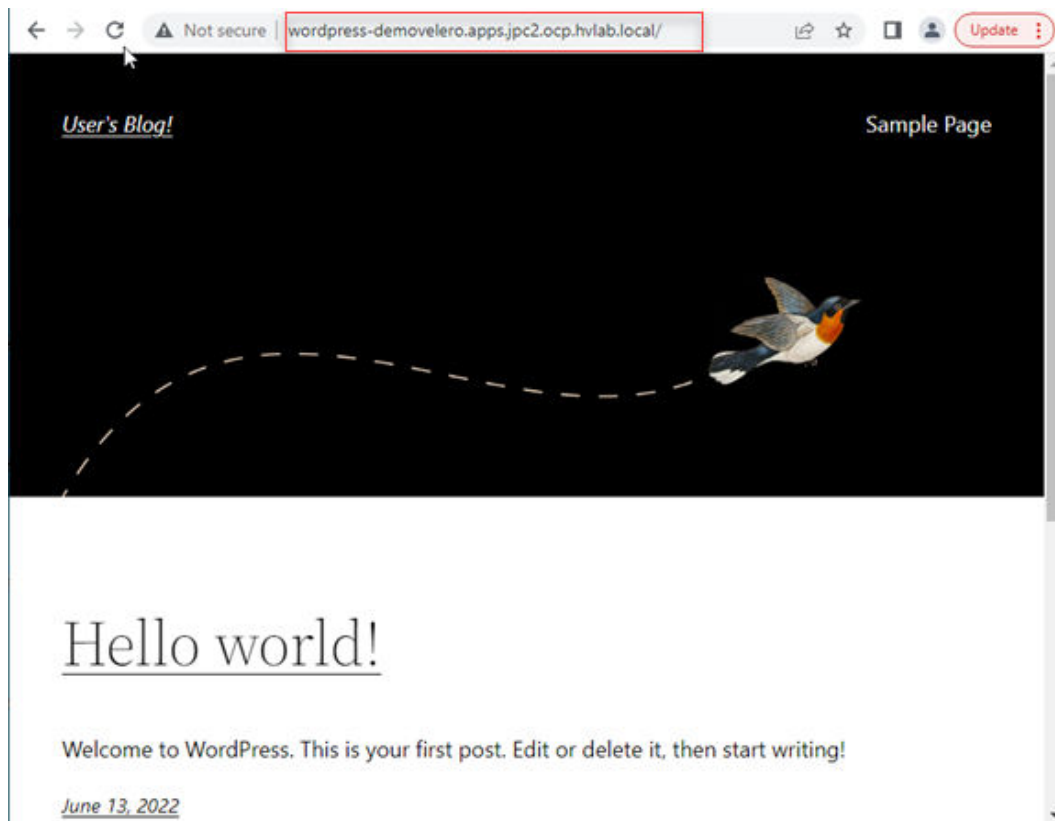
In the previous step you displayed the resources in the `demovero` namespace, which included the output of the services for Wordpress. The next step is to expose the service/wordpress using the `oc expose service/wordpress` command, log in to Wordpress and create a post that will be validated during the backup/restore process.

Procedure

1. To identify the host/port of the exposed Wordpress service, use the following command:

```
[ocpusr@dminws-c2]$ oc get routes
NAME          HOST/PORT                                     PATH   SERVICES   PORT   T
wordpress    wordpress-demovero.apps.jpc2.ocp.hvlab.local  /      wordpress  http
```

2. Open a browser and enter the address of the Wordpress service (http), and then the Wordpress interface should display. The following shows an example of the default Wordpress application user interface. We can identify the Wordpress application has been deployed in OCP cluster #2 (jpc2).



3. Create some content, for this purpose browse to the admin interface of Wordpress (/ admin), and log in using the username and password you set in the Helm installation script.
4. Click the **Create** for your first post link.
5. Enter information to create a blog post, and then publish the post.
6. Navigate back to the default URL of the Wordpress application to verify that your post was committed to the database. Here is an example of the new post:



Explore backend storage resource mapping and allocations

When modifying the Helm chart values for the Wordpress deployment, you provided three different StorageClasses that mapped back to vSphere SPBM policies for persistent volume allocation to the Wordpress and MariaDB pods.

Using the VMware vSphere Cloud Native Storage and First Class Disk (FCD) features, you can follow the storage paths from the Kubernetes persistent volume layer to the vSphere vSAN/vVol layer.

Verify the backend storage resources mapping and allocations

Starting at the OCP cluster layer, you can explore the PVC and corresponding PVs that were provisioned by the Hitachi HSPC CSI driver by following this procedure.

Procedure

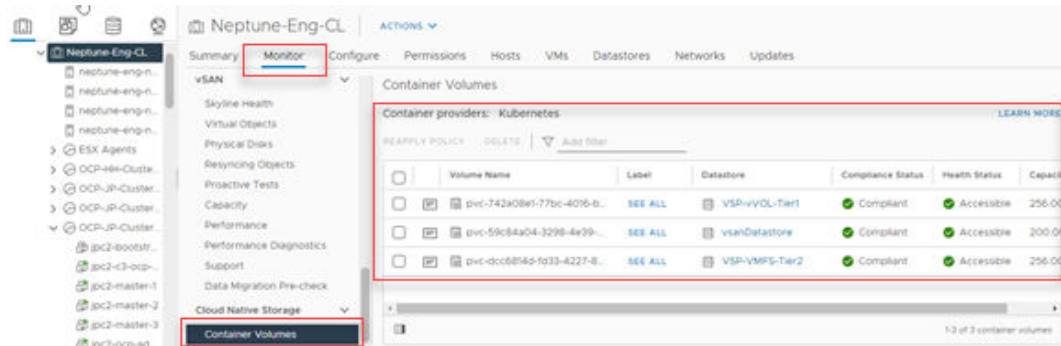
1. To list the PVCs created during the Wordpress Helm chart deployment, run the following command:

```
oc get pvc -n demoveler0
```

```
[ocpuser@mainw-02]# oc get pvc
NAME                                STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS
data-wordpress-mariadb-primary-0    Bound   pvc-742a08e1-77bc-4016-bca3-189f1224e979  256Mi     RWO            hitachi-vvol-tier1-sc
data-wordpress-mariadb-secondary-0  Bound   pvc-d0c6814d-fd33-4227-8a49-476a8a830839  256Mi     RWO            hitachi-vmfs-tier2-sc
wordpress                            Bound   pvc-59c24a04-3298-4e39-8015-5d907439cc02  200Mi     RWO            vSAN-test-sc
```

Back up the Wordpress application using Red Hat OADP/Velero and HSPC CSI snapshots

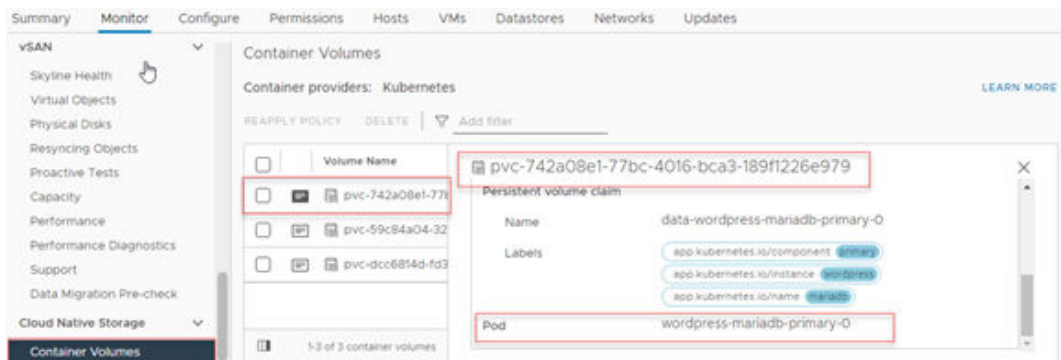
2. To observe details about the volume within vCenter, open a browser and then open a vSphere web client session to the vCenter hosting the OCP cluster `jpc2`.
3. Highlight the vSphere cluster hosting the OCP cluster VMs and navigate to its Monitor tab.
4. In the left pane expand **Cloud Native Storage**, and then click **Container Volumes**. You will see container volumes provisioned to your cluster in the right pane.



In the next step, we are going to display more details for one of the container volumes, the one assigned to the primary DB.

5. Find the volume that matches your PVC ID from the previous step, and then click on the **Details** icon.

This displays the details about the volume that are surfaced from OpenShift, including the persistent volume ID, namespace, labels, and pod allocation from within the OCP cluster.



Back up the Wordpress application using Red Hat OADP/Velero and HSPC CSI snapshots

Follow this procedure to create and verify the status of the backup:

Procedure

1. From your Linux workstation, log in to your OCP cluster.

2. Create a backup custom resource (CR). Makes sure the default `VolumesToRestic` parameter is set to `true`. The following is an example of the backup CR used to back up the Wordpress application:

```
cat demowordpress-backup-restic.yaml
```

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  namespace: openshift-adp
  name: demowordpress-backup-restic
  labels:
    velero.io/storage-location: default
spec:
  defaultVolumesToRestic: true
  includedNamespaces:
  - demovelero
```

3. Issue the following command to back up the Wordpress application:

```
oc apply -f demowordpress-backup-restic.yaml
```

4. The following command lists all the backups.

```
[ocpusr@dminws-c2]$ oc get backups -n openshift-adp
NAME                                AGE
demoapps2-backup-restic             77m
demowordpress-backup-restic         9s
```

5. When the Velero backup is initiated, you can run the `oc describe backup demovelero-backup-restic` command to show the progress of the Velero backup.

```

[cpusr@dminws-c2]$ oc describe backup demowordpress-backup-restore -n openshift-adp
Name: demowordpress-backup-restore
Namespace: openshift-adp
Labels: velero.io/storage-location=velero-ocp-hcpcs-1
Annotations: velero.io/source-cluster-k8s-gitversion: v1.22.5+a36406b
              velero.io/source-cluster-k8s-major-version: 1
              velero.io/source-cluster-k8s-minor-version: 22
API Version: velero.io/v1
Kind: Backup
Metadata:
  Creation Timestamp: 2022-06-13T23:02:22Z
  Generation: 8
  Managed Fields:
    API Version: velero.io/v1
    Fields Type: FieldsV1
    fieldsV1:
      ...
    Manager: velero-server
    Operation: Update
    Time: 2022-06-13T23:05:24Z
  Resource Version: 172635747
  UID: 12c52814-b8cf-49ab-9a82-c3c4035e6744
Spec:
  Default Volumes To Restore: true
  Included Namespaces:
    demovelero
  Storage Location: velero-ocp-hcpcs-1
  Ttl: 720h0m0s
  Volume Snapshot Locations:
    velero-ocp-hcpcs-1
Status:
  Completion Timestamp: 2022-06-13T23:05:19Z
  Expiration: 2022-07-13T23:02:22Z
  Format Version: 1.1.0
  Phase: Completed
  Progress:
    Items Backed Up: 98
    Total Items: 98
  Start Timestamp: 2022-06-13T23:02:22Z

```

Explore the Hitachi Content Platform for cloud scale S3 object store

Follow this procedure to explore the data on the HCP CS S3 bucket.

Procedure

1. Open a web browser, navigate to your Hitachi Content Platform for cloud scale web interface, and log in.
2. Browse the contents of your bucket; for this test we are using a new bucket called the *onprem-vcf-velero-target* folder and open it.

You will see the contents of all of the Kubernetes objects that were backed up during the Velero backup, with the exception of the PVs and their data.

Hitachi | S3 Console

< onprem-vcf-velero-target
 onprem-vcf-velero-target / velero / backups / demowordpress-backup-restore /

Browse Overview Properties

Show deleted objects

All

	Name	Size	Modified
<input type="checkbox"/>	demowordpress-backup-restore-csi-volumes...	29 B	06/13/2022 16:05:24
<input type="checkbox"/>	demowordpress-backup-restore-csi-volumes...	29 B	06/13/2022 16:05:23
<input type="checkbox"/>	demowordpress-backup-restore-logs.gz	10.91 KB	06/13/2022 16:05:20
<input type="checkbox"/>	demowordpress-backup-restore-podvolume...	1.33 KB	06/13/2022 16:05:22
<input type="checkbox"/>	demowordpress-backup-restore-resource-lis...	1.03 KB	06/13/2022 16:05:23
<input type="checkbox"/>	demowordpress-backup-restore-volumesna...	29 B	06/13/2022 16:05:22
<input type="checkbox"/>	demowordpress-backup-restore.tar.gz	210.92 KB	06/13/2022 16:05:21
<input type="checkbox"/>	velero-backup.json	2.58 KB	06/13/2022 16:05:20

3. Browse the contents of your bucket, locate the *onprem-vcf-velero-target* folder, and open it.
4. Navigate back to the root of your bucket and select the plugins folder.
5. Navigate to *restic/demove/velero/data*, where *demove/velero* is the name of the namespace for the Wordpress application.
6. Under data you will find multiple folders with the data.
 These folders correspond to the PVs that were backed up from the vSphere container volumes taken by the Red Hat OADP/Velero and Restic.

< onprem-vcf-velero-target
 onprem-vcf-velero-target / velero / restic / demovelero /

Browse Overview Properties

Show deleted objects

All

	Name	Size	Modified
<input type="checkbox"/>	data	--	
<input type="checkbox"/>	index	--	
<input type="checkbox"/>	keys	--	
<input type="checkbox"/>	snapshots	--	
<input type="checkbox"/>	config	155 B	06/13/2022 16:02:49



< onprem-vcf-velero-target
 onprem-vcf-velero-target / velero / restic / demovelero / data /

Browse Overview Properties

Show deleted objects

Create Directory **Upload Object**

All Delete Destroy

	Name	Size	Modified	Owner	Storage class	
<input type="checkbox"/>	00	--		--	STANDARD	⋮
<input type="checkbox"/>	04	--		--	STANDARD	⋮
<input type="checkbox"/>	07	--		--	STANDARD	⋮
<input type="checkbox"/>	09	--		--	STANDARD	⋮
<input type="checkbox"/>	0a	--		--	STANDARD	⋮
<input type="checkbox"/>	15	--		--	STANDARD	⋮
<input type="checkbox"/>	23	--		--	STANDARD	⋮
<input type="checkbox"/>	27	--		--	STANDARD	⋮
<input type="checkbox"/>	28	--		--	STANDARD	⋮
<input type="checkbox"/>	36	--		--	STANDARD	⋮

Show 10 rows

HITACHI | S3 Console


< onprem-vcf-velero-target
onprem-vcf-velero-target / velero / restic / demovelero / data / 00 /

Browse Overview Properties

Show deleted objects

All

	Name	Size	Modified
<input type="checkbox"/>	00b3d6594b4e5235a799afaedb271ccb2c1659a662...	4.02 MB	06/13/2022 16:02:54

 **Note:** Do not delete or modify these objects. If you want to do so, do it through the Velero CLI.

Delete the Wordpress application from OCP cluster

Follow this procedure to delete Wordpress application and its persistent volumes from the primary cluster.

Procedure


1. Log in to your OCP cluster from your Linux workstation.
2. Run the `helm delete wordpress -n demovelero` command to remove the Wordpress application.
3. When the application is removed, run the `oc delete project demovelero` command to remove all of the resources including PVCs and PVs from the cluster.

```
[ocpusr@dminws-c2]$ helm delete wordpress
release "wordpress" uninstalled

[ocpusr@dminws-c2]$ oc delete project demovelero
project.project.openshift.io "demovelero" deleted

[ocpusr@dminws-c2]$ oc get all -n demovelero
No resources found in demovelero namespace.

[ocpusr@dminws-c2]$ oc get pvc -n demovelero
No resources found in demovelero namespace.
```

 **Note:** While the restore process will be executed in a secondary OCP cluster, there should be no need to remove the Wordpress application from the primary OCP cluster.

Restore the Wordpress application using the Velero operator

The restore process in this demo is executed in a secondary OCP cluster called *jpc3*. This OCP cluster is a hybrid (virtual and bare metal worker nodes) similar to the primary cluster, and while the scheduling of the applications can be managed using a NodeSelector, for this exercise the bare metal node has been marked as unschedulable to make sure the application is scheduled/restored in any of the virtual worker nodes because these worker nodes are the ones that can leverage vSphere CSI.

Follow this procedure to restore an application from backup in the secondary OCP cluster.

Procedure

1. Verify the bare metal worker node has been marked as unschedulable.

```
[ocpusr@dminws-c3]$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
jpc3-master-1	Ready	master	146d	v1.22.5+a36406b
jpc3-master-2	Ready	master	146d	v1.22.5+a36406b
jpc3-master-3	Ready	master	146d	v1.22.5+a36406b
jpc3-worker-1	Ready	worker	146d	v1.22.5+a36406b
jpc3-worker-2	Ready	worker	146d	v1.22.5+a36406b
jpc3-worker-3	Ready,SchedulingDisabled	worker	146d	v1.22.5+a36406b

2. Because we are restoring the application in a new cluster, make sure that the StorageClasses have the same name as the ones used to deploy the Wordpress application in the primary cluster.

The following example shows StorageClasses with the same name as those from the primary OCP cluster:

```
[ocpusr@dminws-c3]$ oc get sc
```

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
csi-test-sc	csi.vsphere.vmware.com	Delete	Immediate
hitachi-vmfs-tier2-sc	csi.vsphere.vmware.com	Delete	Immediate
hitachi-vvol-tier1-sc	csi.vsphere.vmware.com	Delete	Immediate
sc-vsp-112 (default)	hspc.csi.hitachi.com	Delete	Immediate
sc-vsp-113	hspc.csi.hitachi.com	Delete	Immediate
vsan-test-sc	csi.vsphere.vmware.com	Delete	Immediate
vsp-hrhc-sc	hspc.csi.hitachi.com	Delete	Immediate

3. The OADP/Velero in the secondary cluster has been installed/configured with the same S3/bucket. Verify that the secondary cluster can see the backup made from the primary cluster. Run the `velero backup get` command to list the backups in the Velero database.

You can see the backups of the MySQL application that you took previously.

```
[ocpusr@dminws-c3]$ oc get backups -n openshift-adp
```

NAME	AGE
demoapps2-backup-restic	82m
demowordpress-backup-restic	4m43s

4. Verify there is no namespace with the name *demovelero*:

```
[ocpusr@dminws-c3]$ oc get project demovelero
Error from server (NotFound): namespaces "demovelero" not found
```

5. Create a restore custom resource (CR).

The following is an example of the restore CR used to restore Wordpress from demowordpress-backup-restic backup using Restic:

```
cat cat demowordpress-restore-restic.yaml
apiVersion: velero.io/v1
kind: Restore
metadata:
  namespace: openshift-adp
  name: demowordpress-restore-restic
spec:
  backupName: demowordpress-backup-restic
  includedNamespaces:
  - demovelero
```

6. Run the following command to begin the restore of the MySQL application and its associated PVs and data:

```
oc apply -f demowordpress-restore-restic.yaml
```

The following command lists the restores:

```
[ocpusr@dminws-c3]$ oc get restores -n openshift-adp
NAME                                AGE
demoapps2-restore-restic            89m
demowordpress-restore-restic        30s
```

7. After the Velero restore is submitted, run the `oc describe restore demowordpress-restore-restic` command to view the progress of the restore.

```
[ocpusr@dmnws-c3]$ oc describe restore demowordpress-restore-restic -n openshift-adp
Name: demowordpress-restore-restic
Namespace: openshift-adp
Labels: <none>
Annotations: <none>
API Version: velero.io/v1
Kind: Restore
Metadata:
  Creation Timestamp: 2022-06-13T23:20:21Z
  Generation: 7
  Managed Fields:
    API Version: velero.io/v1
    Fields Type: FieldsV1
    fieldsV1:
      ...
    Manager: velero-server
    Operation: Update
    Time: 2022-06-13T23:21:32Z
  Resource Version: 151450899
  UID: 5566f850-00c5-4947-bf23-1e2658b06083
Spec:
  Backup Name: demowordpress-backup-restic
  Excluded Resources:
    nodes
    events
    events.events.k8s.io
    backups.velero.io
    restores.velero.io
    resticrepositories.velero.io
  Included Namespaces:
    demovelero
Status:
  Completion Timestamp: 2022-06-13T23:21:32Z
  Phase: Completed
  Progress:
    Items Restored: 57
    Total Items: 57
  Start Timestamp: 2022-06-13T23:20:21Z
  Warnings: 5
  Events: <none>
```

Note that the restore will not show as complete until the application and all of its PVs and associated data are restored.

- Run the following commands to view all of the Wordpress components that were restored in the *demovelero* namespace in the secondary cluster.

```
[ocpusr@dmnws-c3]$ oc get all -n demovelero
NAME READY STATUS RESTARTS AGE
pod/wordpress-679f4ff79b-thrh8 1/1 Running 0 2m18s
pod/wordpress-mariadb-primary-0 1/1 Running 0 2m18s
pod/wordpress-mariadb-secondary-0 1/1 Running 0 2m18s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S)
service/wordpress LoadBalancer 172.30.88.157 <pending> 80:31451/TCP,443:31195/TCP
service/wordpress-mariadb-primary ClusterIP 172.30.231.223 <none> 3306/TCP
service/wordpress-mariadb-secondary ClusterIP 172.30.128.84 <none> 3306/TCP

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/wordpress 1/1 1 1 2m17s

NAME DESIRED CURRENT READY AGE
replicaset.apps/wordpress-679f4ff79b 1 1 1 2m17s

NAME READY AGE
statefulset.apps/wordpress-mariadb-primary 1/1 2m16s
statefulset.apps/wordpress-mariadb-secondary 1/1 2m16s

NAME HOST/PORT PATH SERVICES PORT
TERMINATION WILDCARD
route.route.openshift.io/wordpress wordpress-demovelero.apps.jpc3.ocp.hvlab.local wordpress http
None
```

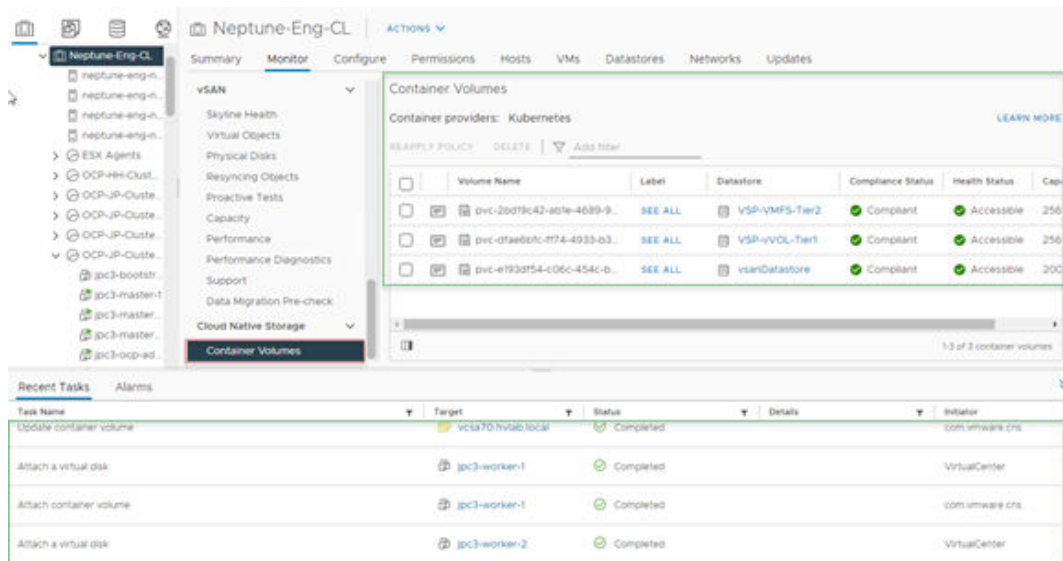
```
[ocpusr@dmnws-c3]$ oc get pvc -n demovelero
```

NAME	STORAGECLASS	AGE	STATUS	VOLUME	CAPACITY	ACCESS MODES
data-wordpress-mariadb-primary-0	hitachi-vvol-tier1-sc	2m37s	Bound	pvc-dfae6bfc-ff74-4933-b39e-1cb53c0c97e1	256Mi	RWO
data-wordpress-mariadb-secondary-0	hitachi-vmfs-tier2-sc	2m37s	Bound	pvc-2bd19c42-able-4689-962a-cfd0eb5a1e0	256Mi	RWO
wordpress	vsan-test-sc	2m37s	Bound	pvc-e193df54-c06c-454c-b311-78fb4fe90937	200Mi	RWO

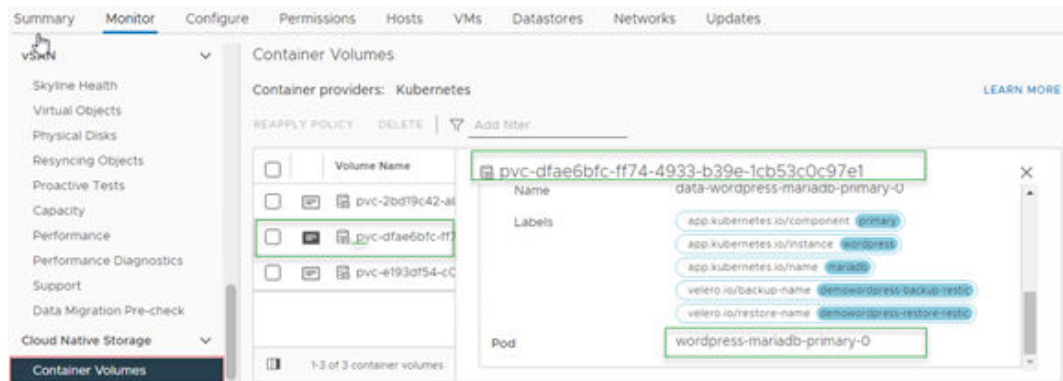

```
[ocpusr@dmnws-c3]$ oc get pv
```

NAME	STORAGECLASS	REASON	AGE	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM
pvc-2bd19c42-able-4689-962a-cfd0eb5a1e0	hitachi-vmfs-tier2-sc		2m41s	256Mi	RWO	Delete	Bound	demovelero/data-
pvc-dfae6bfc-ff74-4933-b39e-1cb53c0c97e1	hitachi-vvol-tier1-sc		2m25s	256Mi	RWO	Delete	Bound	demovelero/data-
pvc-e193df54-c06c-454c-b311-78fb4fe90937	vsan-test-sc		2m39s	200Mi	RWO	Delete	Bound	demovelero/wordpress

- To observe details about the restored container volumes within vCenter, open a browser, and then open a vSphere web client session to the vCenter hosting the secondary OCP cluster *jpc3*.
- Highlight the vSphere cluster hosting the secondary OCP cluster VMs and navigate to its **Monitor** tab.
- In the left pane expand **Cloud Native Storage**, and then click **Container Volumes**. You will see the restored container volumes provisioned to your cluster in the right pane.



The following is an example of additional details of one of the restored volumes for the primary DB.



Verify Wordpress persistent application data

Open a browser and enter the address of the Wordpress service (http) in the secondary OCP cluster, then the Wordpress interface should display. The following shows an example of the restored Wordpress application with the blog entry created before backup. The URL shows the Wordpress application has been restored and is running on the OCP cluster *jpc3*.



Disaster Recovery / Replication Services for Persistent Storage

Replication services for persistent storage on Hitachi VSP storage systems can be enabled with storage class which uses Hitachi Storage (VASA) Provider supported VMware CNS-CSI persistent storage VMDKs or Hitachi Replication Plug-in for Containers (HRPC) for Hitachi CSI-managed persistent volumes.

In this guide we are demonstrating a use case for persistent volume replication with Hitachi Replication Plug-in for Containers.

Hitachi Replication Plug-in for Containers (HRPC) provides replication data services for the persistent volumes on Hitachi's VSP storage platforms. Covering use cases such as:

- Migration – Persistent volumes can be snapshotted and cloned locally or to remote Kubernetes clusters with their own remote VSP storage system.
- Disaster Recovery – Persistent volumes can be protected against datacenter failures by having the data replicated at extensive distances using Hitachi Universal Replicator.
- Backup – Persistent volumes can be protected with point in time snapshots locally with the Hitachi CSI plugin (Hitachi Storage Plug-in for Containers), or they can be backed up to remote VSP storage using HRPC.

Hitachi Replication Plug-in for Containers (HRPC) supports any Kubernetes cluster configured with Hitachi Storage Plug-in for Containers; this guide covers the installation on a Red Hat OpenShift Container Platform configured with Hitachi VSP storage. The infrastructure for this demo is based on the Hitachi Unified Compute Platform.

For configuration details, see the [Hitachi Replication Plug-in for Containers Configuration Guide](#).

Requirements

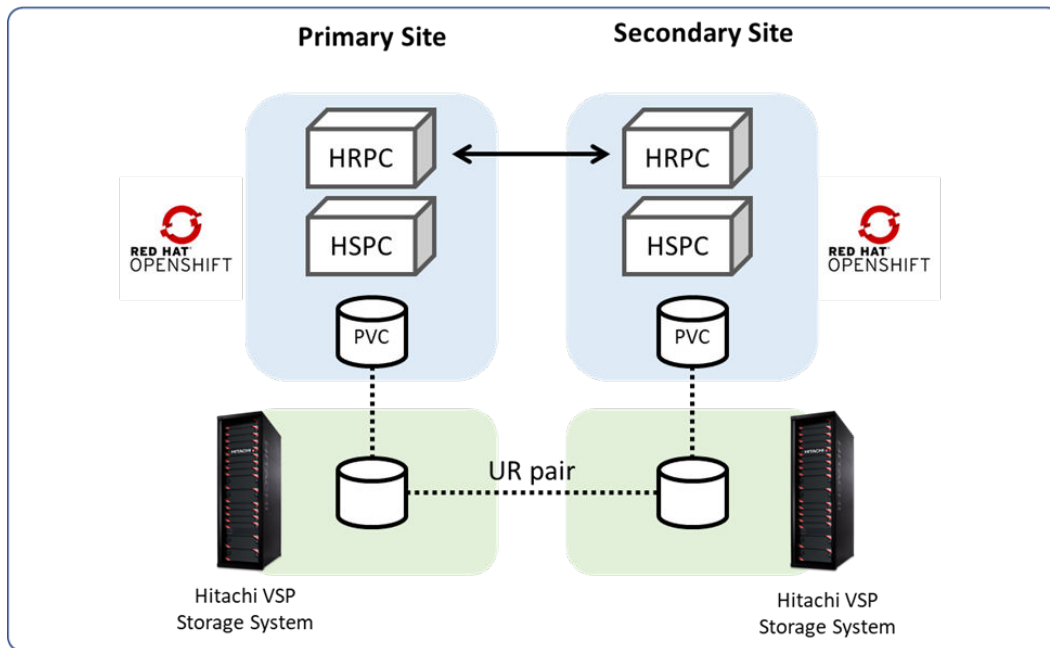
Before installation, complete the following requirements:

- Install two Kubernetes clusters, one in the primary and the other in the secondary site. A single Kubernetes cluster is not supported.
- Configure Hitachi Universal Replicator (HUR). For more details, see [Universal Replicator Overview](#).
- Install [Hitachi Storage Plug-in for Containers](#) in both clusters, either Kubernetes or Red Hat OpenShift Container.
- For inter-site connectivity:
 - Hitachi Replication Plug-in for Containers in the primary site must communicate with the Kubernetes cluster in the secondary site and vice versa.
 - Hitachi Replication Plug-in for Containers in the primary site must communicate with the storage system in the secondary site and vice versa.
 - Connection between primary and secondary storage system RESP API.
 - Fibre Channel or iSCSI connection is needed between primary and secondary storage systems for data copy.

See the [HRPC Configuration Guide](#) for more details.

Installing and Configuring Hitachi Replication Plug-in for Containers

For this demo we have configured two OpenShift Clusters, each configured with 3 masters and 3 workers, and each cluster was connected to different Hitachi VSP Storage systems as seen in the following figure.



OCP Cluster – Primary Site						OCP Cluster – Secondary Site					
[ocpinstall@admins ~]\$ KUBECONFIG=\$(KUBECONFIG 1) oc get nodes						[ocpinstall@admins ~]\$ KUBECONFIG=\$(KUBECONFIG 3) oc get nodes					
NAME	STATUS	ROLES	AGE	VERSION		NAME	STATUS	ROLES	AGE	VERSION	
3pc2-master-1	Ready	master	32d	v1.21.6+b4b4813		3pc3-master-1	Ready	master	31d	v1.21.6+b4b4813	
3pc2-master-2	Ready	master	32d	v1.21.6+b4b4813		3pc3-master-2	Ready	master	31d	v1.21.6+b4b4813	
3pc2-master-3	Ready	master	32d	v1.21.6+b4b4813		3pc3-master-3	Ready	master	31d	v1.21.6+b4b4813	
3pc2-worker-1	Ready	worker	32d	v1.21.6+b4b4813		3pc3-worker-1	Ready	worker	31d	v1.21.6+b4b4813	
3pc2-worker-2	Ready	worker	32d	v1.21.6+b4b4813		3pc3-worker-2	Ready	worker	31d	v1.21.6+b4b4813	
3pc2-worker-3	Ready	worker	32d	v1.21.6+b4b4813		3pc3-worker-3	Ready	worker	31d	v1.21.6+b4b4813	

Configure the storage systems

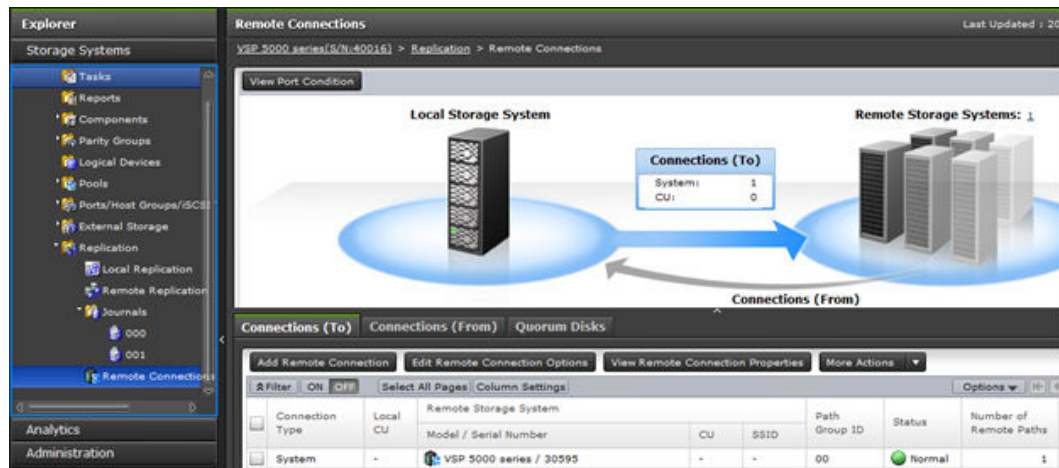
Configure the storage system for replication:

- Configure the storage system as described in the *Hitachi Storage Plug-in for Containers Quick Reference Guide*.
- Configure the remote path between primary site and secondary site storage systems. For details, see the *Hitachi Universal Replicator (HUR) User Guide*.
- Configure journal volumes. For details, see the *Hitachi Universal Replicator (HUR) User Guide*.
- Create a StorageClass in both primary and secondary sites:
 - The name and fstype of the StorageClass must be the same for both sites
 - The StorageClass in the primary site must point to the storage in the primary site.
 - The StorageClass in the secondary site must point to the storage in the secondary site.

An example of the StorageClass (for both sites) is provided in [Creating a manifest file for Replication CR \(on page 80\)](#).

- Create a namespace in both primary and secondary sites. The namespace must have the same name in both sites.

The following figure shows the remote connection configured between the two Hitachi VSP storage systems; the first VSP is connected to the primary Kubernetes cluster, and the second VSP is connected to the secondary Kubernetes cluster.



Configure Hitachi Replication Plug-in for Containers (HRPC)

The installation of HRPC requires to have a dedicated management workstation/VM that can access both the primary and secondary Kubernetes clusters.

The following tasks are just a summary of installation/configuration steps, for details follow the [Hitachi Replication Plug-in for Containers Configuration Guide](#).

Part I: Prepare manifest files and environment variables

Procedure

1. Download and extract the installation media for HRPC into the management workstation.

```
unzip hrpc_<version>.zip
```

2. Get the kubeconfig file from both the primary and secondary sites.

```
KUBECONFIG_P=/path/to/primary-kubeconfig
KUBECONFIG_S=/path/to/secondary-kubeconfig
```

The following files are created in the later steps:

```
SECRET_KUBECONFIG_P=/path/to/primary-kubeconfig-secret.yaml
SECRET_KUBECONFIG_S=/path/to/secondary-kubeconfig-secret.yaml
```

3. Configure an environment variable for the secret file of the storage system.

```
SECRET_STORAGE=/path/to/storage-secret.yaml
```

4. Copy the namespace manifest file to the management machine. This file is provided in the media kit (`hspc-replication-operator-namespace.yaml`). Do not edit it.
5. Create a Secret manifest file with the **secondary** kubeconfig information to access the secondary Kubernetes cluster from Hitachi Replication Plug-in for Containers running in the primary Kubernetes cluster. For reference, see the `remote-kubeconfig-sample.yaml` file. Here an example:

```
# base64 encoding
cat ${KUBECONFIG_S} | base64 -w 0
vi secondary-kubeconfig-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: hspc-replication-operator-remote-kubeconfig
  namespace: hspc-replication-operator-system
type: Opaque
data:
  remote-kubeconfig: <base64 encoded secondary kubeconfig>
```

6. Create a Secret manifest file the with the **primary** kubeconfig information to access the primary Kubernetes cluster from Hitachi Replication Plug-in for Containers running in the secondary Kubernetes cluster. For reference, see the `remote-kubeconfig-sample.yaml` file.

Here is an example:

```
# base64 encoding
cat ${KUBECONFIG_P} | base64 -w 0
vi primary-kubeconfig-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: hspc-replication-operator-remote-kubeconfig
  namespace: hspc-replication-operator-system
type: Opaque
data:
  remote-kubeconfig: <base64 encoded primary kubeconfig>
```

7. Create a Secret manifest file containing storage system information that enables access by Hitachi Replication Plug-in for Containers. For reference, see the `storage-secrets-sample.yaml` file. This manifest file includes information for both the primary and secondary storage systems.

Here is an example:

```
vi ${SECRET_STORAGE}
```

```
apiVersion: v1
kind: Secret
metadata:
  name: hspc-replication-operator-storage-secrets
  namespace: hspc-replication-operator-system
type: Opaque
stringData:
  storage-secrets.yaml: |-
    storages:
      - serial: 40016          #Serial number, primary storage system
        url: https://172.25.47.x #URL for the REST API server
        user: UserPriary      #User, primary storage system
        password: PasswordPrimary #Password for user
        journal: 1           #Journal ID HUR primary storage system
      - serial: 30595        #Serial number, secondary storage system
        url: https://172.25.47.y #URL for the REST API server
        user: UserSecondary   #User, secondary storage system
        password: PasswordSecondary #Password for user
        journal: 1           #Journal ID HUR secondary storage system
```

8. Modify the Hitachi Replication Plug-in for Containers manifest file (`hspc-replication-operator.yaml`) provided in the media kit based on your requirement to use your private repository.

Part II: Install the Hitachi Replication Plug-in for Containers Operator

Procedure

1. From the management workstation, login to both primary and secondary clusters:

```
KUBECONFIG=${KUBECONFIG_P} oc login -u <admin user> -p <Password>
KUBECONFIG=${KUBECONFIG_S} oc login -u <admin user> -p <Password>
```

2. Create Namespaces in the primary and secondary sites. Use the same manifest file in primary and secondary sites.

```
KUBECONFIG=${KUBECONFIG_P} oc create -f hspc-replication-operator-namespace.yaml
KUBECONFIG=${KUBECONFIG_S} oc create -f hspc-replication-operator-namespace.yaml
```

3. Create Secrets containing kubeconfig information in primary and secondary sites. Use the different manifest files in primary and secondary sites.

```
KUBECONFIG=${KUBECONFIG_P} oc create -f ${SECRET_KUBECONFIG_S}
KUBECONFIG=${KUBECONFIG_S} oc create -f ${SECRET_KUBECONFIG_P}
```

4. Create Secrets containing storage system information in primary and secondary sites. Use the same manifest file in primary and secondary sites.

```
KUBECONFIG=${KUBECONFIG_P} oc create -f ${SECRET_STORAGE}
KUBECONFIG=${KUBECONFIG_S} oc create -f ${SECRET_STORAGE}
```

5. Load the container (for example, docker load or podman for OpenShift) `hrpc_<version>.tar` and push the loaded container to your private repository.
6. Create Hitachi Replication Plug-in for Containers in primary and secondary sites. Use the same manifest file for both the primary and secondary sites.

```
KUBECONFIG=${KUBECONFIG_S} oc create -f hspc-replication-operator.yaml
KUBECONFIG=${KUBECONFIG_P} oc create -f hspc-replication-operator.yaml
```

7. Confirm that Hitachi Replication Plug-in for Containers are running in primary and secondary sites.

Check HRPC operator in primary site:

```
KUBECONFIG=${KUBECONFIG_P} oc get pods -n hspc-replication-operator-system
```

Check HRPC operator in secondary site:

```
KUBECONFIG=${KUBECONFIG_S} oc get pods -n hspc-replication-operator-system
```

```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get pods -n hspc-replication-operator-system
NAME                                READY   STATUS    RESTARTS   AGE
hspc-replication-operator-controller-manager-5b5dc456b5-zrbnm    1/1     Running   0          21h
[ocpinstall@adminws ~]$
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_S} oc get pods -n hspc-replication-operator-system
NAME                                READY   STATUS    RESTARTS   AGE
hspc-replication-operator-controller-manager-6b7bcbf5b4-54kpn    1/1     Running   0          21h
[ocpinstall@adminws ~]$
```

At this point, the Hitachi Replication Plug-in for Containers operator is ready and the next steps is to install and test with a stateful app or just create a PVC and Pod that consumes the PVC.

Installing a Stateful App for Replication Testing on the Primary Site

For this demo, we install MySQL database using Bitnami Helm chart.

Procedure

1. First, we need to add the Bitnami repository by running the following command:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Search for the MySQL Helm chart by running the following command:

```
[ocpininstall@adminws ~]$ helm search repo mysql
NAME                CHART VERSION  APP VERSION  DESCRIPTION
bitnami/mysql       8.8.23         8.0.28      Chart to create a Highly
available MySQL cluster
```

2. Next, create a namespace for the stateful app for the demo. Use the following command to create a namespace (project):

```
KUBECONFIG=${KUBECONFIG_P} oc new-project demoapps
```

3. Verify storage class.

StorageClass `vsp-hrhc-sc` was created following the requirements for Hitachi Storage Plug-in for Containers.

```
[ocpininstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get sc
```

```
[ocpininstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get sc vsp-hrhc-sc
NAME          PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
vsp-hrhc-sc  hspc.csi.hitachi.com  Delete         Immediate           true                   2d20h
[ocpininstall@adminws ~]$
```

4. Customize and deploy MySQL Helm chart with persistent storage.

The following shows an example of the installation of MySQL on the primary site using StorageClass `vsp-hrhc-sc`, database called `vsp_database`, and a Persistent Volume with 5Gi of capacity.

```
[ocpininstall@adminws ~]$ helm install mysql-hrhc-example \
--set secondary.replicaCount=0 \
--set global.storageClass=vsp-hrhc-sc \
--set primary.persistence.size=5Gi \
--set auth.rootPassword=Hitachi123,auth.database=vsp_database \
--set secondary.replicaCount=0 \
--set primary.podSecurityContext.enabled=false \
--set primary.containerSecurityContext.enabled=false \
--set secondary.podSecurityContext.enabled=false \
--set secondary
```

We can use the following commands to check the status of the MySQL pod and its corresponding Persistent Volume.

```
KUBECONFIG=${KUBECONFIG_P} oc get pods
```

```
[ocpininstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-hrpc-example-0                1/1     Running   0           22h
```

```
KUBECONFIG=${KUBECONFIG_P} oc get pvc
```

```
[ocpininstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get pvc
NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-mysql-hrpc-example-0           Bound    pvc-d2f7691e-b98f-4ee8-ae1c-238ead55a859   5Gi        RWO             vsp-hrpc-sc    22h
```

5. Insert test data in the MySQL database.

Once the MySQL pod is ready, a new table called `replication_cr_status` is created on the `vsp_database` database. Then a few records of test data are inserted into this new table. The create table and insert commands are not showed here.

```
[ocpininstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc exec -it mysql-hrpc-example-0 -- bash
1000700000@mysql-hrpc-example-0:/$
1000700000@mysql-hrpc-example-0:/$ mysql -h mysql-hrpc-example.hv-apps.svc.cluster.local -uroot -p"Hitachi123"
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| vsp_database |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

The following data has been inserted into the MySQL database with the purpose to test and verify replicated PVC to the secondary site:

```
mysql> use vsp_database;
Database changed

mysql> SELECT * FROM replication_cr_status;
+-----+-----+-----+
| Status | Description | HUR_pair_state |
+-----+-----+-----+
| Copying | Initial copy is in progress | COPY |
| Error | Having errors in HUR pair | PSUE, PFUL, or PFUS |
| Failover | Successfully failover HUR pair | SSWS |
| Pending | Accepted CR creation command | NA |
| Ready | Successfully created HUR pair and the state is PAIR | PAIR |
| Split | Successfully splitted HUR pair and the state is PSUS or SSUS | PSUS or SSUS |
| Unknown | A pair does not exist or unexpected pair state | Other than the above |
| WaitingPrimarySite | Waiting for primary site to finish HUR operations | NA |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

The next step is to replicate the Persistent Volume `data-mysql-hrpc-example-0` used by the MySQL pod to the secondary site.

Replicating Persistent Volumes

To replicate storage volumes requires to create a Replication CustomResource (CR) object. Once the Replication CR has been created, HRPC starts replicating the specified PVC and triggers the creation of a PVC in the secondary site. The data in the target PVC from the primary site is copied and protected by HUR.

Creating a manifest file for Replication CR

A Replication CR manifest file contains the name of the PVC and the StorageClass name. The manifest file below is created to replicate the PVC `data-mysql-hrpc-example-0` previously used by MySQL Pod.



Note: A StorageClass with the same name must exist on the secondary site (examples below). Also, a namespace with the same name must be created on the secondary site before creating the Replication CR.

```
cat hspc_v1_mysqldb_replication.yaml
```

```
apiVersion: hspc.hitachi.com/v1
kind: Replication
metadata:
  name: replication-mysqldb1
spec:
  persistentVolumeClaimName: data-mysql-hrpc-example-0
  storageClassName: vsp-hrpc-sc
```

Here is an example of the StorageClass CR for the VSP storage on the Primary site:

```
cat vsp-hrpc-sc.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: vsp-hrpc-sc
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
provisioner: hspc.csi.hitachi.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  serialNumber: "40016"
  poolID: "1"
  portID : CL1-D,CL4-D
  connectionType: fc
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-publish-secret-name: "secret-vsp-113"
  csi.storage.k8s.io/node-publish-secret-namespace: "test"
```



```
csi.storage.k8s.io/provisioner-secret-name: "secret-vsp-113"
csi.storage.k8s.io/provisioner-secret-namespace: "test"
csi.storage.k8s.io/controller-publish-secret-name: "secret-vsp-113"
csi.storage.k8s.io/controller-publish-secret-namespace: "test"
csi.storage.k8s.io/node-stage-secret-name: "secret-vsp-113"
csi.storage.k8s.io/node-stage-secret-namespace: "test"
csi.storage.k8s.io/controller-expand-secret-name: "secret-vsp-113"
csi.storage.k8s.io/controller-expand-secret-namespace: "test"
```

Here is an example of the StorageClass CR for the VSP storage on the Secondary site:

```
cat vsp-hrhc-sc.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: vsp-hrhc-sc
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
provisioner: hspc.csi.hitachi.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  serialNumber: "30595"
  poolID: "12"
  portID : CL1-B,CL4-B
  connectionType: fc
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-publish-secret-name: "secret-vsp-112"
  csi.storage.k8s.io/node-publish-secret-namespace: "test"
  csi.storage.k8s.io/provisioner-secret-name: "secret-vsp-112"
  csi.storage.k8s.io/provisioner-secret-namespace: "test"
  csi.storage.k8s.io/controller-publish-secret-name: "secret-vsp-112"
  csi.storage.k8s.io/controller-publish-secret-namespace: "test"
  csi.storage.k8s.io/node-stage-secret-name: "secret-vsp-112"
  csi.storage.k8s.io/node-stage-secret-namespace: "test"
  csi.storage.k8s.io/controller-expand-secret-name: "secret-vsp-112"
  csi.storage.k8s.io/controller-expand-secret-namespace: "test"
```

We can see that both StorageClasses have the same name, and each points to the respective VSP storage in each site. As we can see in these examples, the StorageClass CR is similar that the one used for non-replicated environment.

Creating a Replication CR object

The Replication CR object is created in the primary site, using the manifest file from previous step. This triggers the creation of an HUR pair and initial data copy. Also, this triggers the creation of a Replication CR object on the secondary site.

Verifying the status of Replication CR in primary and secondary site

Use the following command to create the Replication CR:

```
KUBECONFIG=${KUBECONFIG_P} oc create -f hspc_v1_mysqlb_replication.yaml
```

Verifying the status of Replication CR in primary and secondary site

The Replication CR status is changed to Ready when the initial replication is created, and the data protection is started. Transition to the Ready status depends on data size in target PVC and might take some time to change.

The following commands help to verify the status of the Replication CR objects in both the primary and secondary site.

```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get replication
NAME                STATUS  DESIREDSTATE  OPERATION  AGE
replication-mysqldb1  Ready  none          none       22h

[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_S} oc get replication
NAME                STATUS  DESIREDSTATE  OPERATION  AGE
replication-mysqldb1  Ready  none          none       22h
```

Also, we can see that a PVC `data-mysql-hrpc-example-0` has been automatically created on the secondary site.

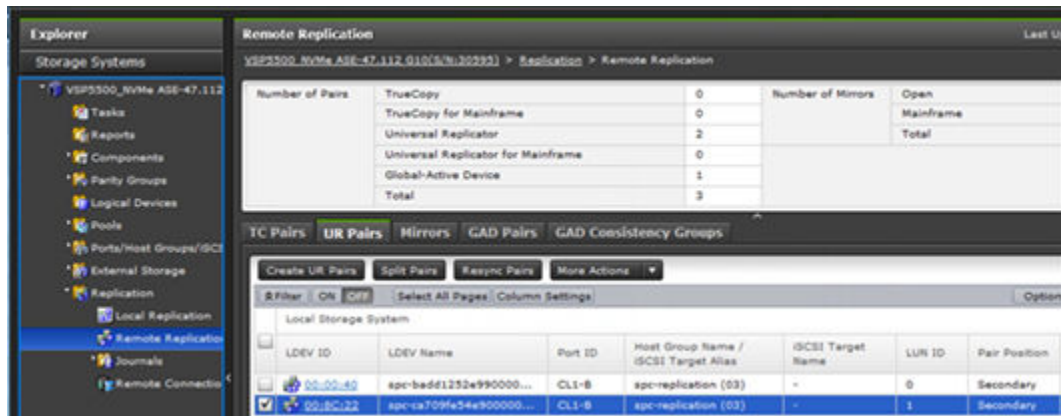
```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_S} oc get pvc
NAME                STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
data-mysql-hrpc-example-0  Bound  pvc-12890e69-4a01-4345-a45f-ae2f6f07069e  5Gi       RWO           vsp-hrpc-sc   23h
```

On the VSP Storage systems, we can verify that UR pairs have been automatically created as well:

UR pairs on primary storage system:

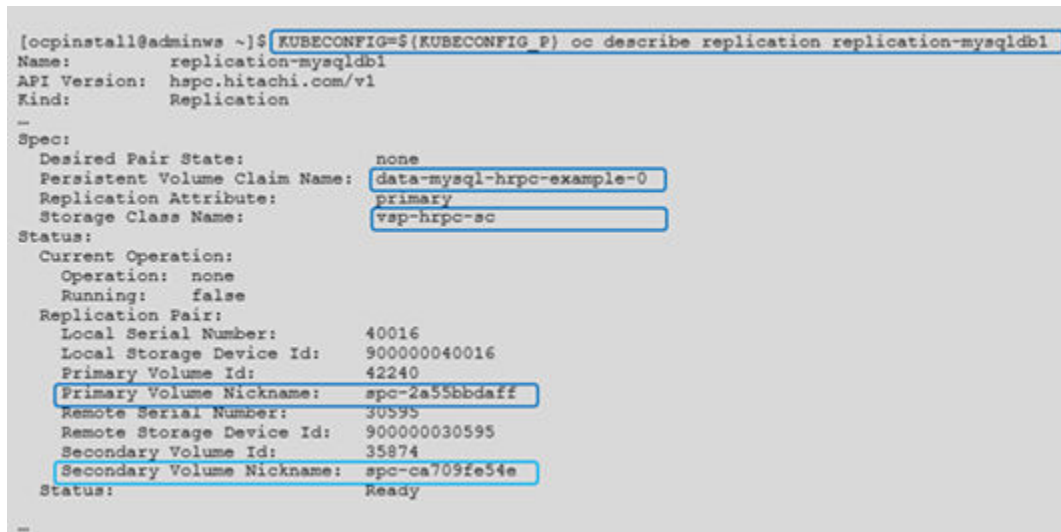
Local Storage System	LDEV ID	LDEV Name	Port ID	Host Group Name / iSCSI Target Alias	iSCSI Target Name	LUN ID	Pair Position
<input type="checkbox"/>	00_24_10	spc-580583165d9000000...	CL1-D	spc-100000109b4816...	-	19	Primary
<input checked="" type="checkbox"/>	00_43_20	spc-2a57b0a4f92000003...	CL1-D	spc-100000109b4816...	-	168	Primary

UR pairs on secondary storage system:



The following command provides more details from the Replication CR, like storage serial number, LDEV Name for both primary and secondary site, and it can easily be correlated with the LDEVs seen on the UR Pairs.

```
KUBECONFIG=${KUBECONFIG_P} oc describe replication replication-mysqldb1
```



Checking replicated data on the Secondary Site

After the Replication CR is created and it is in “Ready” status, we can perform a `split` and `resync` operation to check the data from the secondary site. The data copy process from the primary site to the secondary site is stopped during the `split` and `resync` operation.

When you `split` a pair, write-data is no longer sent to the S-VOL and the pair is no longer synchronized. Splitting a pair or mirror gives you a point-in-time copy of the P-VOL.

For more details about HUR operations, see [Universal Replicator Overview](#).

Disaster Recovery Test : Splitting the Hitachi Universal Replicator pair

To `split` the HUR pair, from the primary site, change the status of Replication CR to perform the `split` operation. This triggers HRPC to `split` the HUR pair.

First confirm the status of the Replication CR is Ready and Operation value is `none`.

```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get replication
NAME                STATUS    DESIREDSTATE  OPERATION  AGE
replication-mysqldb1 Ready     none          none       22h
```

Then use the command below to edit the Replication CR and change the `spec.desiredPairState` to `split`.

```
KUBECONFIG=${KUBECONFIG_P} oc edit replication replication-mysqldb1
```

```
apiVersion: hspc.hitachi.com/v1
kind: Replication
metadata:
  ...
  name: replication-mysqldb1
  ...
# Before
spec:
  desiredPairState: <any state>
  persistentVolumeClaimName: data-mysql-hrpc-example-0
  replicationAttribute: primary
  storageClassName: vsp-hrpc-sc
  ...
# After
spec:
  desiredPairState: split
  persistentVolumeClaimName: data-mysql-hrpc-example-0
  replicationAttribute: primary
  storageClassName: vsp-hrpc-sc
  ...
```

After the edit, make sure the Replication CR status is `split` and the operation value is `none`.

```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get replication
NAME                STATUS    DESIREDSTATE  OPERATION  AGE
replication-mysqldb1 Split     split         none       23h
```

Confirming the replicated data

To confirm the replicated data on the secondary site, we are going to deploy the same MySQL Helm chart, but this time it will be deployed with the existing Persistent Volume that was replicated from the primary site `data-mysql-hrpc-example-0`.

To accomplish this with helm chart we need to customize and indicate that MySQL needs to be installed with an existing PVC as seen below.

```
[ocpinstall@jpc3-ocp-admin-ws]$ helm install mysql-hrpc-example \
--set secondary.replicaCount=0 \
--set global.storageClass=vsp-hrpc-sc \
--set primary.persistence.size=5Gi \
--set auth.rootPassword=Hitachi123,auth.database=vsp_database \
--set secondary.replicaCount=0 \
--set primary.podSecurityContext.enabled=false \
--set primary.containerSecurityContext.enabled=false \
--set secondary.podSecurityContext.enabled=false \
--set secondary.containerSecurityContext.enabled=false \
--set primary.persistence.storageClass=vsp-hrpc-sc \
```

```
--set primary.persistence.existingClaim=data-mysql-hrpc-example-0 \
bitnami/mysql
```

We can use the following commands to check the status of the MySQL pod and its corresponding Persistent Volume on the Secondary site.

```
KUBECONFIG=${KUBECONFIG_P} oc get pods
KUBECONFIG=${KUBECONFIG_P} oc get pvc
```

Or directly from the console of the secondary cluster:

```
[ocpinstall@jpc3-ocp-admin-ws]# oc get pods
NAME                READY   STATUS    RESTARTS   AGE
mysql-hrpc-example-0 1/1     Running   0           23h

[ocpinstall@jpc3-ocp-admin-ws]# oc get pvc
NAME                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-mysql-hrpc-example-0 Bound      pvc-12890e69-4a01-4345-a45f-as2f6f07069e  5Gi        RWO            vsp-hrpc-oc    23h
```

The next step is to connect to the MySQL database and verify the same data that was created from the primary site.

```
[ocpinstall@jpc3-ocp-admin-ws]# oc exec -it mysql-hrpc-example-0 -- bash
1000710000@mysql-hrpc-example-0:/$ mysql -h mysql-hrpc-example.hv-apps.svc.cluster.local -uroot -p"Mitachi123"
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\o' to clear the current input statement.

mysql>
```

The following query confirms that the PVC/ MySQL database contains the same data that was inserted on the primary site.

```
[ocpinstall@jpc3-ocp-admin-ws]#
...

mysql> show databases;
+-----+
| Database |
+-----+
| vsp_database |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> use vsp_database;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM replication_cr_status;
+-----+-----+-----+
| Status | Description | NUR_pair_state |
+-----+-----+-----+
| Copying | Initial copy is in progress | COPY |
| Error | Having errors in NUR pair | PSUE, PFUL, or PFUS |
| Failover | Successfully failover NUR pair | SSWS |
| Pending | Accepted CR creation command | NA |
| Ready | Successfully created NUR pair and the state is PAIR | PAIR |
| Split | Successfully splitted NUR pair and the state is PSUS or SSUS | PSUS or SSUS |
| Unknown | A pair does not exist or unexpected pair state | Other than the above |
| WaitingPrimarySite | Waiting for primary site to finish NUR operations | NA |
+-----+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

Now that we have confirmed the same data on the PVC on the secondary site, we can uninstall the MySQL helm chart, the PVC will remain there since it is controlled by the Replication CR.

```
helm uninstall mysql-hrpc-example
oc get pvc
oc get pods
```

```
[ocpinstall@jpc3-ocp-admin-ws]$ helm uninstall mysql-hrpc-example
release "mysql-hrpc-example" uninstalled
[ocpinstall@jpc3-ocp-admin-ws]$ helm list
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/ocpinstall/ocp-upi-install/auth/kubeconfig
NAME      NAMESPACE      REVISION      UPDATED STATUS   CHART   APP VERSION
[ocpinstall@jpc3-ocp-admin-ws]$ oc get pvc
NAME                                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-mysql-hrpc-example-0           Bound    pvc-12890e69-4a01-4345-a45f-ae2f6f07069e   5Gi        RWO             vsp-hrpc-sc    22h
[ocpinstall@jpc3-ocp-admin-ws]$ oc get pods
No resources found in demooapp namespace.
[ocpinstall@jpc3-ocp-admin-ws]$
```

Resynchronizing HUR pair

To resync the HUR pair, from the primary site, change the status of the Replication CR to perform the resync operation. This triggers Hitachi Replication Plug-in for Containers to resync the HUR pair.

Make sure no Pod is using the PVC, otherwise the resync operation will not work.

```
KUBECONFIG=${KUBECONFIG_P} oc edit replication replication-mysqldb1
```

```
apiVersion: hspc.hitachi.com/v1
kind: Replication
metadata:
  name: replication-mysqldb1
  ...
spec:
  # Before
  desiredPairState: split
  persistentVolumeClaimName: data-mysql-hrpc-example-0
  replicationAttribute: primary
  storageClassName: vsp-hrpc-sc
  ...
  # After
  desiredPairState: pair
  persistentVolumeClaimName: data-mysql-hrpc-example-0
  replicationAttribute: primary
  storageClassName: vsp-hrpc-sc
  ...
```

After the edit, make sure the Replication CR status is Ready and the Operation value is none.

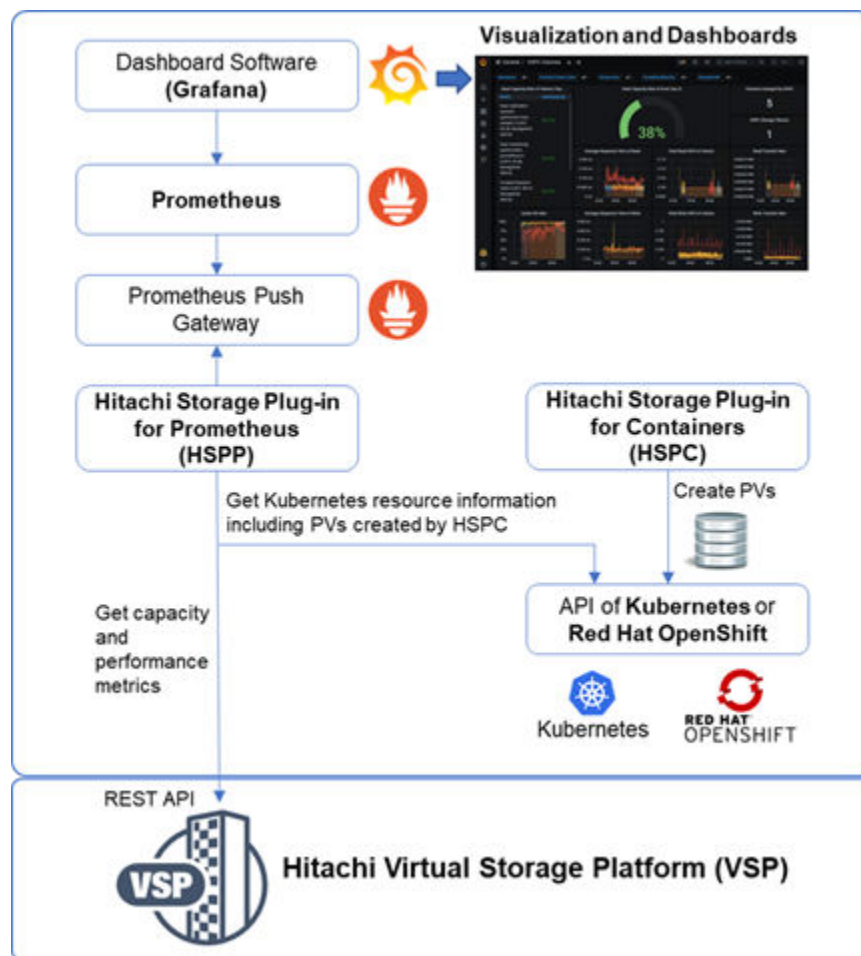
```
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_P} oc get replication
NAME                                STATUS    DESIREDSTATE  OPERATION  AGE
replication-mysqldb1               Ready    pair          none       24h
[ocpinstall@adminws ~]$ KUBECONFIG=${KUBECONFIG_S} oc get replication
NAME                                STATUS    DESIREDSTATE  OPERATION  AGE
replication-mysqldb1               Ready    pair          none       24h
```


Monitoring Kubernetes resources and Hitachi storage with Hitachi Storage Plug-in for Prometheus

Hitachi Storage Plug-in for Prometheus enables the Kubernetes administrator to monitor the metrics of Kubernetes resources and Hitachi storage system resources within a single tool. Hitachi Storage Plug-in for Prometheus uses Prometheus to collect metrics and Grafana to visualize those metrics for easy evaluation by the Kubernetes administrator. Prometheus collects storage system metrics such as capacity, IOPS, and transfer rate in five-minute intervals.

For additional details about configuration, follow the [Hitachi Storage Plug-in for Prometheus Quick Reference Guide](#).

The following diagram shows the flow of metric collection using Hitachi Storage Plug-in for Prometheus.



Requirements

- Install the Kubernetes or Red Hat OpenShift Container Platform.
- Download the Storage Plug-in for Prometheus installation media kit from the Hitachi Support Connect Portal: <https://support.hitachivantara.com/en/user/home.html>. A Hitachi login credential is required.

- Install [Hitachi Storage Plug-in for Containers](#) in Kubernetes or Red Hat OpenShift Container Platform.
- Configure StorageClass for Hitachi Storage Plug-in for Containers in Kubernetes or Red Hat OpenShift Container Platform.

Installing Hitachi Storage Plug-in for Prometheus on OpenShift Cluster

The installation below for this demo is executed on top of an OpenShift Cluster (version 4.8) configured with 3 masters and 3 workers where the worker nodes are connected to Hitachi VSP Storage.

```
oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
jpc2-master-1	Ready	master	29d	v1.21.6+b4b4813
jpc2-master-2	Ready	master	29d	v1.21.6+b4b4813
jpc2-master-3	Ready	master	29d	v1.21.6+b4b4813
jpc2-worker-1	Ready	worker	29d	v1.21.6+b4b4813
jpc2-worker-2	Ready	worker	29d	v1.21.6+b4b4813
jpc2-worker-3	Ready	worker	28d	v1.21.6+b4b4813

Procedure

1. Download and extract the installation media.

```
tar zxvf storage-exporter.tar.gz
```

2. Load the Storage Plug-in for Prometheus image into the repository.
3. Update the `exporter.yaml` with the corresponding registry hostname and port for the cluster.
4. Update the `secret-sample.yaml` using the info from the VSP storage: Serial Number, Storage System API URL, user and password.

```
apiVersion: v1
kind: Secret
metadata:
  name: storage-exporter-secret
  namespace: hspc-monitoring-system
type: Opaque
stringData:
  storage-exporter.yaml: |-
    storages:
    - serial: 40016
      url: https://172.25.47.x
      user: MaintenanceUser
      password: PasswordForUser
```

5. Create the namespace `hspc-monitoring-system` for Storage Plug-in for Prometheus:

```
oc apply -f yaml/namespace.yaml
```

6. Create security context constraints (SCC):

```
oc apply -f yaml/scc-for-openshift.yaml
```

7. Install Storage Plug-in for Prometheus and Prometheus Pushgateway.

```
oc apply -f yaml/secret-sample.yaml -f yaml/exporter.yaml
```

Verify the storage-exporter and pushgateway are running on namespace hspc-monitoring-system:

```
oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
pushgateway-77b85489b9-4vnzt	1/1	Running	0	5d
storage-exporter-77b644b8b7-pzhdj	1/1	Running	0	5d

Installing Prometheus and Grafana

After installing Hitachi Storage Plug-in for Prometheus, install and configure Prometheus and Grafana. For more information, see <https://prometheus.io/> and <https://grafana.com/>.

If you are installing Prometheus and Grafana manually, there is a couple of steps you will need to take. Follow Hitachi Storage Plug-in for Prometheus Reference Guide.

- Connect Prometheus to Pushgateway
- Import sample dashboard `json grafana/sample.json` to Grafana.

For this test and demo purpose, we are using the quick installer that comes with the Hitachi Storage Plug-in for Prometheus package.

Procedure

1. In the `grafana-prometheus-sample.yaml` file, replace `StorageClass` of with your own `StorageClass`.
2. (Optional) Modify the Grafana service.
The `grafana-prometheus-sample.yaml` file exposes Grafana as a `NodePort` with a random `nodeport`. If you want to expose Grafana in a different way, modify the `grafana-prometheus-sample.yaml` file.
3. Deploy Grafana and Prometheus.

```
oc apply -f yaml/grafana-prometheus-sample.yaml
```

Verify the Prometheus and Grafana PODs are running:

```
oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
grafana-0	1/1	Running	0	5d
prometheus-0	1/1	Running	0	5d

```
pushgateway-77b85489b9-4vnzt      1/1      Running  0          5d
storage-exporter-77b644b8b7-pzhdj  1/1      Running  0          5d
```

Make sure the four PODs are running.

4. Access Grafana.

If you use NodePort, access Grafana with <Your Node IP Address>:<Grafana Port>. You can identify <Grafana Port> by using the following command.

```
oc get svc

NAME          TYPE          CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
grafana       NodePort      172.30.219.171  <none>       3000:31929/TCP  5d
prometheus    NodePort      172.30.180.214  <none>       9090:31661/TCP  5d
pushgateway   ClusterIP     172.30.25.210   <none>       9091/TCP         5d
```

If you expose the Grafana, please get endpoint by yourself. The Grafana user/password are admin/secret.

```
oc expose svc/grafana

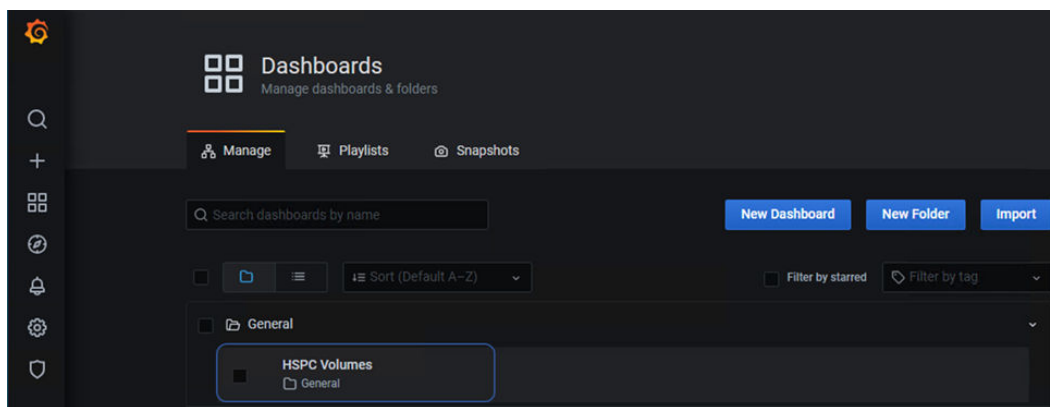
oc get routes

NAME          HOST/PORT          PATH
SERVICES      PORT              TERMINATION  WILDCARD
grafana       grafana-hspc-monitoring-system.apps.jp2.ocp.hvlab.local
grafana       None
```

Monitoring Dashboard with Hitachi Storage Plug-in for Prometheus

When using the quick installer for Grafana and Prometheus following the previous steps, there are no additional steps to configure Prometheus as a data source since it is pre-configured.

On the dashboards, the installer includes a dashboard for *HSPC Volumes*.



The HSPC Volumes Dashboard shows metrics for Persistent Volumes such as Capacity, Response Time, IOPS, Read/Write Transfer Rate, and Cache Hit Rate.

These metrics can be presented by Namespace, Persistent Volume Claims VC, Storage Class, Storage Serial Number, or Storage Pool ID.



When doing performance testing you can view specific metrics as shown.



Deploying a private container registry using Red Hat Quay and Hitachi HCP CS S3

Some environments are not allowed to have Internet connected access to public registry for images. In other cases, having a private registry is a security practice that some customers adopt.

Red Hat Quay is a distributed and highly available container image registry platform that, when integrated with Hitachi HCP CS S3, provides a secure storage, distribution, and governance of containers on any infrastructure. This is available as a standalone component or running on top of OCP cluster.

Requirements

Before starting with the deployment of Red Hat Quay Operator on the OCP cluster, consider the following:

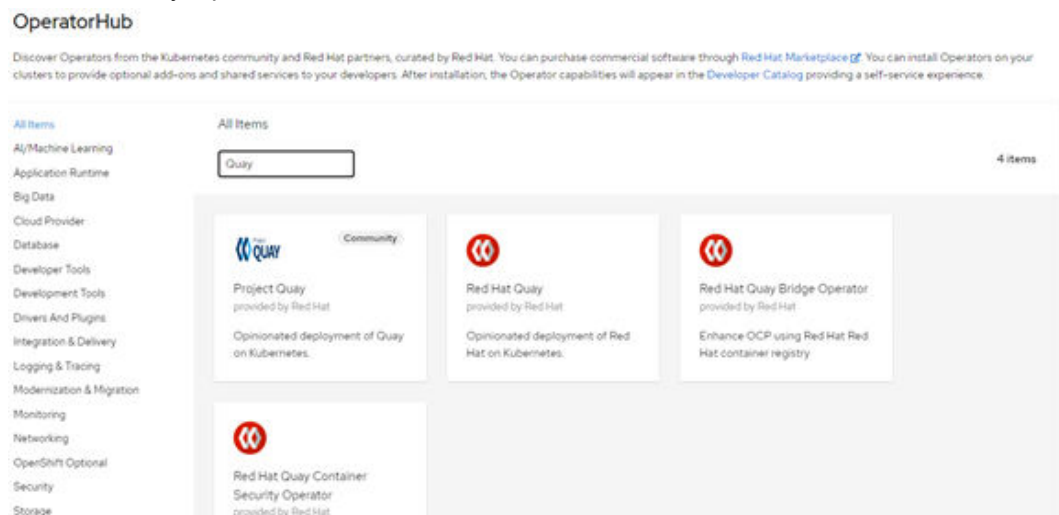
- The OCP cluster is using OpenShift 4.5 or later.
- Ensure the OCP cluster has sufficient compute resources for Quay deployment, see Red Hat Quay documentation for specific requirements.
- Ensure an Object Storage is available. For this demo we are using Hitachi HCP CS S3 storage.

Install Quay Operator from OperatorHub


Follow this procedure to install Quay Operator from OperatorHub.

Procedure

1. Log in to the OCP console, select OperatorHub under Operators, and then select the Red Hat Quay Operator.



The installation page shows features and prerequisites.



Red Hat Quay

3.6.4 provided by Red Hat

✕

Install

Latest version

3.6.4

Capability level

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

Source

Red Hat

Provider

Red Hat

Infrastructure features

Disconnected


Repository

<https://github.com/quay/quay-operator>

Container image

registry.redhat.io/quay/quay-operator-rhel8@sha256:99de3a47361e3d16479a506088a82558079afecfd6345f2a3ef25daef57d1d82

Created at

 Aug 31, 2021, 7:08 AM

The Red Hat Quay Operator deploys and manages a production-ready [Red Hat Quay](#) private container registry. This operator provides an opinionated installation and configuration of Red Hat Quay. All components required, including Clair, database, and storage, are provided in an operator-managed fashion. Each component may optionally be self-managed.

Operator Features

- Automated installation of Red Hat Quay
- Provisions instance of Redis
- Provisions PostgreSQL to support both Quay and Clair
- Installation of Clair for container scanning and integration with Quay
- Provisions and configures RHOCS for supported registry object storage
- Enables and configures Quay's registry mirroring feature

Prerequisites

By default, the Red Hat Quay operator expects RHOCS to be installed on the cluster to provide the *ObjectBucketClaim* API for object storage. For instructions installing and configuring the RHOCS Operator, see the "Enabling OpenShift Container Storage" in the [official documentation](#).

Simplified Deployment


The following example provisions a fully operator-managed deployment of Red Hat Quay, including all services necessary for production:

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: my-registry
```

Documentation

See the [official documentation](#) for more complex deployment scenarios and information.

- Select Install and the operator installation page appears. Select Install one more time. After a short time, you will see the installed operator is ready for use. The operator can be seen on the Installed Operators page as well.



Red Hat Quay

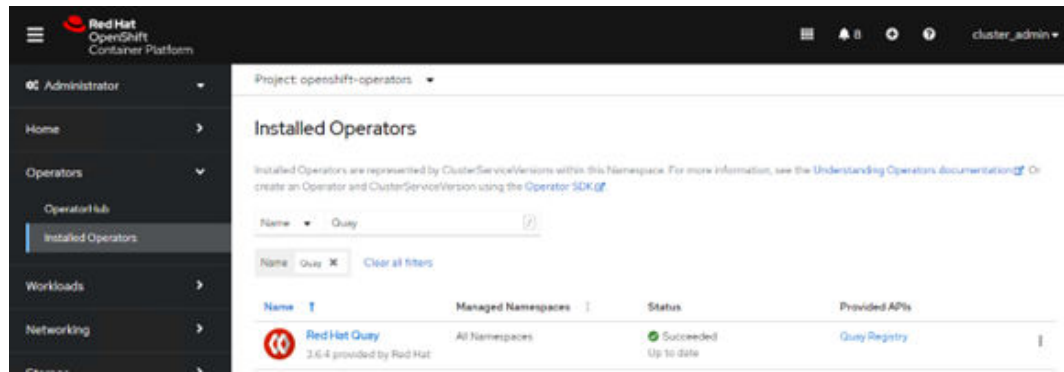
3.6.4 provided by Red Hat

✔

Installed operator - ready for use

View Operator

View installed Operators in Namespace openshift-operators



Configure Quay before deployment

The deployment for this demo uses Hitachi VSP storage for Persistent Volumes for Quay Progress DB and Hitachi HCP CS S3 storage for the image repository.

If there are several StorageClasses already defined on the OCP cluster, make sure to configure one of the storage classes as default so the PVC for Quay Postgres DB can be created there during deployment.

In this case, we are using storage class `vsp-186-sc` which is a storage class defined to be used with Hitachi Storage Plug-in for Containers and Hitachi VSP storage systems.

The following command sets storage class `vsp-186-sc` as the default:

```
oc patch storageclass vsp-186-sc -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

```
[ocpusr@dm1nws-c0]$ oc get sc
NAME          PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
csi-test-sc   csi.vsphere.vmware.com  Delete         Immediate          false                 84d
vsp-186-sc (default)  hspc.csi.hitachi.com  Delete         Immediate          true                  82d
[ocpusr@dm1nws-c0]$
```

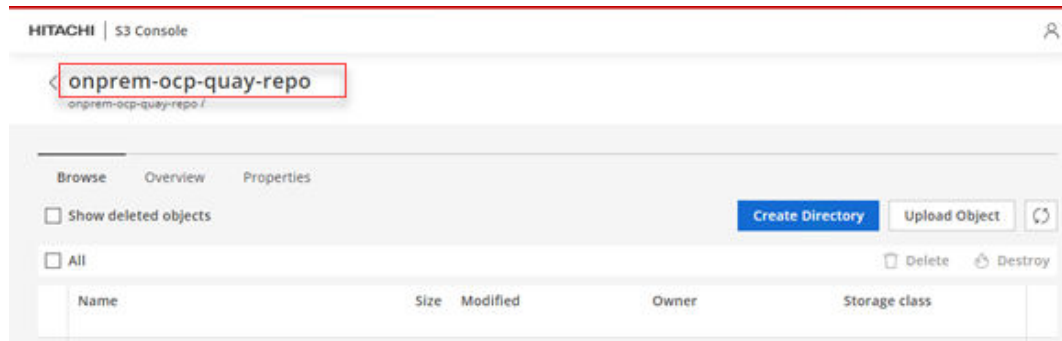
Create an S3 bucket for the repository

Follow this procedure to prepare S3 storage.

Procedure

1. Open a web browser, navigate to your Hitachi Content Platform for cloud scale web interface, and log in.
2. Browse the contents and create a bucket for the Quay repository.

The following example an `onprem-ocp-quay-repo` bucket that has been created for this purpose.



Create configuration and secret to access Hitachi HCP CS S3 store

Create a configuration file that includes the S3 storage (URL for HCP CS), access key, secret key to access Hitachi HCP CS S3, and bucket. Here is an example of the configuration file:

```
cat config_hcpcs.yaml
```

```
DISTRIBUTED_STORAGE_CONFIG:
  s3Storage:
    - S3Storage
    - host: tryhcpforcloudscale.hitachivantara.com
      s3_access_key: Hitachi_HCP_CS_access_key_here
      s3_secret_key: Hitachi_HCP_CS_secret_key_here
      s3_bucket: onprem-ocp-quay-repo
      storage_path: /
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
  - s3Storage
```

The following command creates secret `config-bundle-secret` to access S3 storage using the information from the previous configuration file.

```
oc create secret generic --from-file config.yaml=./config_hcpcs.yaml config-bundle-secret
```

Create the Quay Registry

The next step is to create the Quay Registry; this can be done from the console or using config files. For this demo we are using a minimum configuration with the configuration file `quayregistry_hcpcs.yaml` where we indicate that the object storage will be unmanaged. For production environment or if there is a need to use an existing database, follow Red Hat documentation. Make sure to indicate the name of the secret created in the previous step.

Here is an example of the configuration file used to create a Quay Registry:

```
cat quayregistry_hcpcs.yaml
```

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: openshift-operators
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: clair
      managed: false
    - kind: horizontalpodautoscaler
      managed: false
    - kind: mirror
      managed: false
    - kind: monitoring
      managed: false
```

The following command is used to provision the Quay Registry:

```
oc apply -f quayregistry_hcpcs.yaml
```

After a short time, the following pods for the Quay Registry will be deployed. Note that these are the components deployed for a minimum deployment. The number of PODs might vary depending on the type of deployment.

```
[ocpusr@adminws-c0]$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
example-registry-quay-app-795785d556-kx19d	0/1	Running	0	93s
example-registry-quay-app-795785d556-vxnfb	1/1	Running	0	93s
example-registry-quay-app-upgrade-n6dts	0/1	Completed	0	99s
example-registry-quay-config-editor-5b54c86848-4bdb6	1/1	Running	0	93s
example-registry-quay-database-68f8b7d47c-781w9	1/1	Running	0	3m16s
example-registry-quay-postgres-init-vk9h4	0/1	Completed	0	93s
example-registry-quay-redis-687b9444cd-t9k76	1/1	Running	0	3m16s
quay-operator.v3.6.4-7847768db5-2xgwh	1/1	Running	0	27h

Also, we can see that a PVC was automatically created for the Quay database:

```
[ocpusr@adminws-c0]$ oc get pvc
```

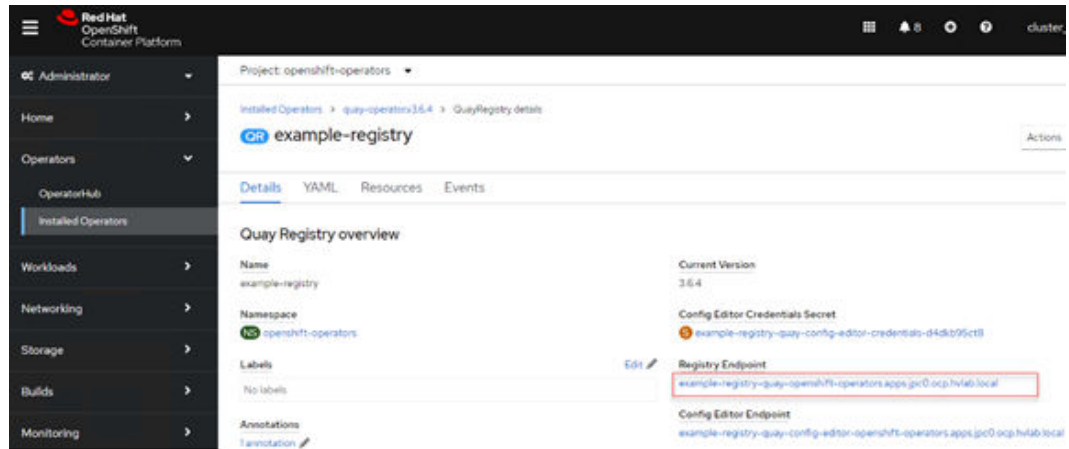
NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
example-registry-quay-database	Bound	pvc-fe773e7b-0998-4482-ad48-bf823efc6ec0	50Gi	RWO
vsp-186-sc				

Create Quay Admin User and Login to Quay EndPoint

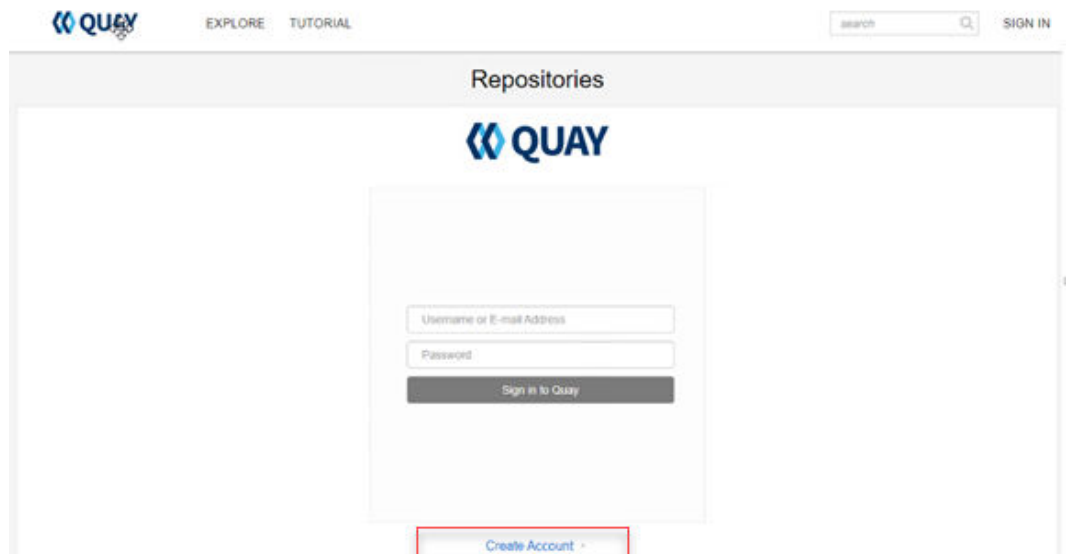
The admin user to access the Quay Registry can be created from the OCP console following this procedure.

Procedure

1. In the OCP console, navigate to **Operators > Installed Operators**, with the appropriate namespace/project.
2. Click on the newly installed Quay Registry, to view the details:



3. Navigate to the URL of the Registry EndPoint, for example in this demo:
`https://example-registry-quay-openshift-operators.apps.jpc0.ocp.hvlab.local/`
4. Select **Create Account** in the Quay registry UI to create a user account.



Test the Quay Registry

Once the admin user has been created, the next step is to test the registry.

Using Podman to push images to the Quay registry

For this test we are using Podman; as seen we are connecting to the Registry EndPoint with the admin user created in the previous step:

```
podman login --tls-verify=false example-registry-quay-openshift-operators.apps.jpc0.ocp.hvlab.local
```

```
[ocpusr@dminws-c0]$ podman login --tls-verify=false example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local
Username: quayadmin
Password:
Login Succeeded!
```

For this test we are going to use the image for Hitachi Storage Plug-in for Prometheus which has been downloaded to a local folder and Podman will be used to push the image to the Quay registry. First, we load the image, assign a tag, and then push the image to the registry:

```
podman load -i storage-exporter/program/storage-exporter.tar
```

```
[ocpusr@dminws-c0]$ podman load -i storage-exporter/program/storage-exporter.tar
Getting image source signatures
Copying blob 6366941b9a40 skipped: already exists
Copying blob e2eb06d8af82 skipped: already exists
Copying config f74f80f354 done
Writing manifest to image destination
Storing signatures
Loaded image(s): localhost/hitachi/storage-plugin-for-prometheus:v1.0.0
```

```
podman tag hitachi/storage-plugin-for-prometheus:v1.0.0 example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/quayadmin/storage-plugin-for-prometheus:v1.0.0
```

```
[ocpusr@dminws-c0]$ podman tag hitachi/storage-plugin-for-prometheus:v1.0.0 example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/quayadmin/storage-plugin-for-prometheus:v1.0.0
[ocpusr@dminws-c0]$
```

```
[ocpusr@dminws-c0]$ podman images
REPOSITORY
TAG          IMAGE ID      CREATED      SIZE
example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/quayadmin/storage-plugin-for-prometheus v1.0.0      f74f80f35454 1 months ago 51.9 MB
localhost/hitachi/storage-plugin-for-prometheus v1.0.0      f74f80f35454 1 months ago 51.9 MB
[ocpusr@dminws-c0]$
```

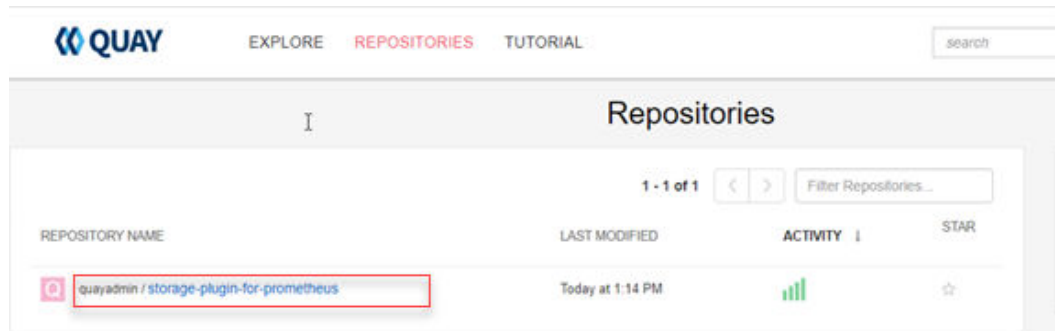
```
podman push --tls-verify=false example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/quayadmin/storage-plugin-for-prometheus:v1.0.0
```

```
[ocpusr@dminws-c0]$ podman push --tls-verify=false example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/quayadmin/storage-plugin-for-prometheus:v1.0.0
Getting image source signatures
Copying blob 6366941b9a40 done
Copying blob e2eb06d8af82 done
Copying config f74f80f354 done
Writing manifest to image destination
Storing signatures
[ocpusr@dminws-c0]$
```

Verification of images on the Quay Registry UI

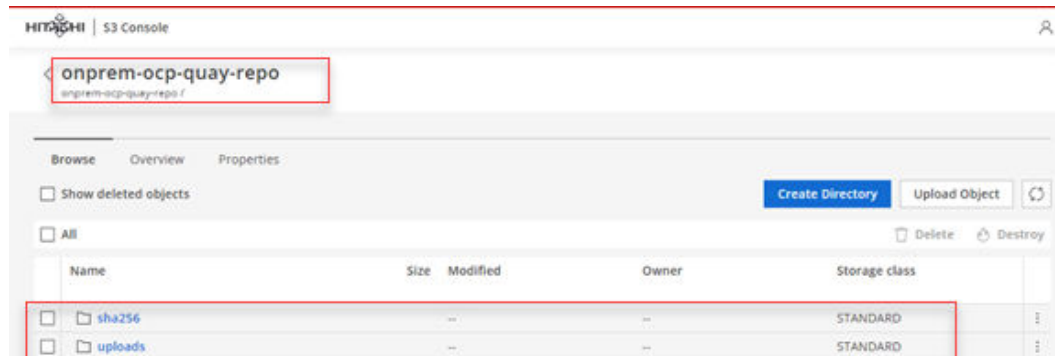
Navigate to the URL of the Registry EndPoint with the admin account and check the image uploaded with Podman. In this case we have the image for Hitachi Storage Plug-in for Prometheus.

<https://example-registry-quay-openshift-operators.apps.jpco.ocp.hvlab.local/>



Verification of images on Hitachi HCP CS S3

To verify images on the S3 store, open a web browser, navigate to your Hitachi Content Platform for cloud scale web interface, and log in. Then, browse the contents for the *onprem-ocp-quay-repo* bucket as shown.



Conclusion

Hitachi Unified Compute Platform, Hitachi Virtual Storage Platform, Hitachi Content Platform for cloud scale, Hitachi Storage Plug-ins for Containers, VMware CSI and VASA plugins, and Red Hat OpenShift Container Platform combine to create a powerful and flexible Kubernetes ecosystem.

This reference architecture highlights recommended approaches for using Red Hat OpenShift in a Hitachi infrastructure environment (UCP and/or VSP) while taking advantage of various Hitachi storage platforms and data storage integrations to achieve a highly resilient and protected platform to deliver Kubernetes clusters and containers at scale.

Product descriptions

This section provides information about the hardware and software components used in this solution for OpenShift on Hitachi Unified Compute Platform.

Hardware components

These are the hardware components available for Hitachi Unified Compute Platform.

Hitachi Advanced Server DS120

Optimized for performance, high density, and power efficiency in a dual-processor server, [Hitachi Advanced Server DS120](#) delivers a balance of compute and storage capacity. This 1U rack-mounted server has the flexibility to power a wide range of solutions and applications.

The highly-scalable memory supports up to 3 TB using 24 slots of high-speed DDR4 memory. Advanced Server DS120 is powered by the Intel Xeon Scalable processor family for complex and demanding workloads. There are flexible OCP and PCIe I/O expansion card options available. This server supports up to 12 small form factor storage devices with up to 4 NVMe drives.

This solution allows you to have a high CPU-to-storage ratio. This is ideal for balanced and compute-heavy workloads.

Multiple CPU and storage devices are available. Contact your Hitachi Vantara sales representative to get the latest list of options.

Hitachi Advanced Server DS120 G2

With support for two Intel Xeon Scalable processors in just 1U of rack space, the [Hitachi Advanced Server DS120 G2](#) delivers exceptional compute density. It provides flexible memory and storage options to meet the needs of converged and hyperconverged infrastructure solutions, as well as for dedicated application platforms such as internet of things (IoT) and data appliances.

The Intel Xeon Scalable processor family is optimized to address the growing demands on today's IT infrastructure. The server provides 32 slots for high-speed DDR4 memory, allowing up to 4 TB memory capacity with RDIMM population (128 GB × 32) or 8 TB (512 GB × 16) of Intel Optane Persistent Memory. DS120 G2 supports up to 12 hot-pluggable, front-side-accessible 2.5-inch non-volatile memory express (NVMe), serial-attached SCSI (SAS), serial-ATA (SATA) hard disk drive (HDD), or solid-state drives (SSD). The system also offers 2 onboard M.2 slots.

With these options, DS120 G2 can be flexibly configured to address both I/O performance and capacity requirements for a wide range of applications and solutions.

Hitachi Advanced Server DS220

With a combination of two Intel Xeon Scalable processors and high storage capacity in a 2U rack-space package, [Hitachi Advanced Server DS220](#) delivers the storage and I/O to meet the needs of converged solutions and high-performance applications in the data center.

The Intel Xeon Scalable processor family is optimized to address the growing demands on today's IT infrastructure. The server provides 24 slots for high-speed DDR4 memory, allowing up to 3 TB of memory per node when 128 GB DIMMs are used. This server supports up to 12 large form factor storage devices and an additional 2 small form factor storage devices.

This server has three storage configuration options:

- 12 large form factor storage devices and an additional 2 small form factor storage devices in the back of the chassis
- 16 SAS or SATA drives, 8 NVMe drives, and an additional 2 small form factor storage devices in the back of the chassis
- 24 SFF devices and an additional 2 SFF storage devices in the back of the chassis

Hitachi Advanced Server DS220 G2

With a combination of two Intel Xeon Scalable processors and high storage capacity in a 2U rack-space package, [Hitachi Advanced Server DS220 G2](#) delivers the storage and I/O to meet the needs of converged solutions and high-performance applications in the data center.

The Intel Xeon Scalable processor family is optimized to address the growing demands on today's IT infrastructure. The server provides 32 slots for high-speed DDR4 memory, allowing up to 4 TB memory capacity with RDIMM population (128 GB × 32) or 8TB (512 GB × 16) with Intel Optane Persistent Memory population.

DS220 G2 comes in three storage configurations to allow for end user flexibility. The first configuration supports 24 2.5-inch non-volatile memory express (NVMe) drives, the second supports 24 2.5-inch serial-attached SCSI (SAS), serial-ATA (SATA) and up to 8 NVMe drives, and the third supports 12 3.5-inch SAS or SATA and up to 8 NVMe drives. All the configurations support hot-pluggable, front-side-accessible drives as well as 2 optional 2.5-inch rear mounted drives. The DS220 G2 delivers high I/O performance and high capacity for demanding applications and solutions.

Hitachi Advanced Server DS225

Choose [Hitachi Advanced Server DS225](#) to ensure you have the flexibility and performance you need to support your business-critical enterprise applications.

Advanced Server DS225 delivers compute density and efficiency to meet the needs of your most demanding high-performance applications. It takes full advantage of the Intel Xeon scalable processor family with up to four dual-width 300 W graphic accelerator cards, up to 3 TB memory capacity, and additional PCIe 3.0 expansion slots in a 2U rack space package.

Front-side accessible storage bays supports up to eight hot-pluggable, serial-attached SCSI (SAS) or serial-ATA (SATA) devices. These bays also support flexible configuration, which allows Advanced Server DS225 to deliver high I/O performance and high capacity.

Hitachi Advanced Server DS240

Meet the needs of your most demanding high-performance applications with [Hitachi Advanced Server DS240](#). With up to four Intel Xeon Scalable Processors and up to 6 TB memory capacity in a 2U rack-space package, this server delivers unparalleled compute density and efficiency.

The Advanced Server DS240 architecture takes full advantage of the Intel Xeon Scalable Processor family, including the highest performance options, to address the growing demands of your IT infrastructure.

Hitachi Virtual Storage Platform 5000 series

This enterprise-class, flash array evolution storage, [Hitachi Virtual Storage Platform 5000 series](#) (VSP) has an innovative, scale-out design optimized for NVMe and storage class memory. It achieves the following:

- **Agility using NVMe:** Speed, massive scaling with no performance slowdowns, intelligent tiering, and efficiency.
- **Resilience:** Superior application availability and flash resilience. Your data is always available, mitigating business risk.
- **Storage simplified:** Do more with less, integrate AI (artificial intelligence) and ML (machine learning), simplify management, and save money and time with consolidation.

Hitachi Virtual Storage Platform E990

[Hitachi Virtual Storage Platform E990](#) supercharges business application performance with all-NVMe storage. It uses Hitachi Ops Center, so you can improve IT operations with the latest AI and ML capabilities. Advanced data reduction in Virtual Storage Platform E990 enables you to run data reduction with even the most performance hungry applications.

The all-NVMe architecture in Virtual Storage Platform E990 delivers consistent, low-microsecond latency to reduce latency costs for critical applications. This predictable performance optimizes storage resources.

With Virtual Storage Platform E990 and the rest of Hitachi midrange storage family, you have agile and automated data center technology. These systems allow you to cost-effectively meet your current digital expectations and give you the ability to address future challenges, as your application data needs and service levels evolve. With time-tested, proven availability and scalability, Hitachi Vantara delivers infrastructure solutions that help you maximize your data center advantage.

Hitachi Virtual Storage Platform F Series family

Use [Hitachi Virtual Storage Platform F series family](#) storage for a flash-powered cloud platform for your mission critical applications. This storage meets demanding performance and uptime business needs. Extremely scalable, its 4.8 million random read IOPS allows you to consolidate more applications for more cost savings.

Hitachi Virtual Storage Platform F series family delivers superior all-flash performance for business-critical applications, with continuous data availability.

Hitachi Virtual Storage Platform G series family

The [Hitachi Virtual Storage Platform G series family](#) enables the seamless automation of the data center. It has a broad range of efficiency technologies that deliver maximum value while making ongoing costs more predictable. You can focus on strategic projects and consolidating more workloads while using a wide range of media choices.

The benefits start with Hitachi Storage Virtualization Operating System RF. This includes an all new enhanced software stack that offers up to three times greater performance than our previous midrange models, even as data scales to petabytes

Hitachi Virtual Storage Platform G series offers support for containers to accelerate cloud-native application development. Provision storage in seconds, and provide persistent data availability, all the while being orchestrated by industry leading container platforms. Move these workloads into an enterprise production environment seamlessly, saving money while reducing support and management costs.

Arista Data Center switches

[Arista Networks](#) builds software-driven cloud networks for data center, cloud, and campus environments. Arista delivers efficient, reliable and high-performance Universal Cloud Network architectures, based on 10 GbE, 25 GbE, 40 GbE, 50 GbE, and 100 GbE platforms delivered with an extensible operating system - Arista EOS.

- [Arista 7050CX3-32S](#) is a 1RU sized spine switch with 32 (downlink) and 4 (uplink) 100 GbE QSFP ports for multiple-rack solutions. Each QSFP port supports a choice of five speeds, with flexible configuration between 100 GbE, 40 GbE, 4 × 10 GbE, 4 × 25 GbE, or 2 × 50 GbE modes.
- [Arista 7050SX3-48YC8](#) is a 1RU sized switch with 48 × 25 GbE SFP and 8 × 100 GbE QSFP ports. The high density SFP ports can be configured in groups of 4 to run either at 25 GbE or a mix of 10 GbE/1 GbE speeds. The QSFP ports allow 100 GbE or 40 GbE high speed network uplinks.
- [Arista 7010T](#) is a 1RU sized, 48-port 1 GbE management switch for single-rack and multiple-rack solutions.

Cisco Nexus switches

The Cisco Nexus switch product line provides a series of solutions that make it easier to connect and manage disparate data center resources with software-defined networking (SDN). Leveraging the Cisco Unified Fabric, which unifies storage, data and networking (Ethernet/IP) services, the Nexus switches create an open, programmable network foundation built to support a virtualized data center environment.

Brocade switches from Broadcom

Brocade and Hitachi Vantara have partnered to deliver storage networking and data center solutions. These solutions reduce complexity and cost, as well as enable virtualization and cloud computing to increase business agility.

[Brocade Fibre Channel switches](#) deliver industry-leading performance, simplifying scale-out network architectures. Get the high-performance, availability, and ease of management you need for a solid foundation to grow the storage network you want.

Software components

These are the software components used in this reference architecture.

Hitachi Storage Virtualization Operating System RF

[Hitachi Storage Virtualization Operating System RF](#) powers the Hitachi Virtual Storage Platform (VSP) family. It integrates storage system software to provide system element management and advanced storage system functions. Used across multiple platforms, Storage Virtualization Operating System includes storage virtualization, thin provisioning, storage service level controls, dynamic provisioning, and performance instrumentation.

Flash performance is optimized with a patented flash-aware I/O stack, which accelerates data access. Adaptive inline data reduction increases storage efficiency while enabling a balance of data efficiency and application performance. Industry-leading storage virtualization allows SVOS RF to use third-party all-flash and hybrid arrays as storage capacity, consolidating resources for a higher ROI and providing a high-speed front end to slower, less-predictable arrays.

Hitachi Unified Compute Platform Advisor

Hitachi Unified Compute Platform Advisor (UCP Advisor) is a comprehensive cloud infrastructure management and automation software that enables IT agility and simplifies day 0-N operations for edge, core, and cloud environments. The fourth-generation UCP Advisor accelerates application deployment and drastically simplifies converged and hyperconverged infrastructure deployment, configuration, life cycle management, and ongoing operations with advanced policy-based automation and orchestration for private and hybrid cloud environments.

The centralized management plane enables remote, federated management for the entire portfolio of converged, hyperconverged, and storage data center infrastructure solutions to improve operational efficiency and reduce management complexity. Its intelligent automation services accelerate infrastructure deployment and configuration, significantly minimizing deployment risk and reducing provisioning time and complexity, automating hundreds of mandatory tasks.

Hitachi Storage Provider for VMware vCenter

When you want to support policy-based automation and improve operational insight into the storage or converged platform hosting that environment, use [Hitachi Storage Provider for VMware vCenter](#). This allows a unique implementation of VMware vSphere API for Storage Awareness (VASA), supporting traditional-based datastores (VMFS and NFS) and VMware vVols-based datastores.

Hitachi Storage Provider for VMware vCenter, as part of the infrastructure, communicates with VMware vCenter to indicate storage capabilities and state information. It supports policy-based management, operations management, and resource scheduling functionality.

Hitachi Content Platform for cloud scale

Hitachi Content Platform for cloud scale (HCP for cloud scale) is a software-defined object storage solution that is based on a massively parallel microservice architecture, and is compatible with the Amazon S3 application programming interface (API). HCP for cloud scale is well suited to service applications requiring high bandwidth and compatibility with Amazon S3 APIs.

Hitachi Storage Plug-in for Containers

[Hitachi Storage Plug-in for Containers \(HSPC\)](#) provides connectivity between Docker, Kubernetes, or Kubernetes Container Storage Interface (CSI) containers and Hitachi Virtual Storage Platform 5000 series, G series, and F series systems. With the compatibility plug-in, your organization can deliver shared storage for containers that persists beyond the timeline of a single container host.

Hitachi Replication Plug-in for Containers

[Hitachi Replication Plug-in for Containers \(HRPC\)](#) automates storage replication between two different Kubernetes clusters and storage systems located at different sites. This enables your organization to take a self-service approach when creating replications using the Kubernetes command-line tool, `kubectl` or `oc`.

Hitachi Storage Plug-in for Prometheus

[Hitachi Storage Plug-in for Prometheus \(HSPP\)](#) enables Kubernetes administrators to monitor the metrics of Kubernetes resources and Hitachi storage system resources within a single tool.

Red Hat OpenShift

[Red Hat Enterprise Linux High Availability Add-On](#) allows a service to fail over from 1 node to another with no apparent interruption to cluster clients, evicting faulty nodes during transfer to prevent data corruption. This Add-On can be configured for most applications (both off-the-shelf and custom) and virtual guests, supporting up to 16 nodes. The High Availability Add-On features a cluster manager, lock management, fencing, command-line cluster configuration, and a Conga administration tool.

Red Hat Enterprise Linux

Using the stability and flexibility of [Red Hat Enterprise Linux](#), reallocate your resources towards meeting the next challenges instead of maintaining the status quo. Deliver meaningful business results by providing exceptional reliability on military-grade security. Use Enterprise Linux to tailor your infrastructure as markets shift and technologies evolve.

VMware vSphere

[VMware vSphere](#) is a virtualization platform that provides a datacenter infrastructure. It helps you get the best performance, availability, and efficiency from your infrastructure and applications. Virtualize applications with confidence using consistent management.

VMware vSphere has the following components:

- VMware vSphere ESXi

This hypervisor loads directly on a physical server. ESXi provides a robust, high-performance virtualization layer that abstracts server hardware resources and makes them shareable by multiple virtual machines.

- VMware vCenter Server

This management software provides a centralized platform for managing your VMware vSphere environments so you can automate and deliver a virtual infrastructure with confidence:

- VMware vSphere vMotion
- VMware vSphere Storage vMotion
- VMware vSphere Distributed Resource Scheduler
- VMware vSphere High Availability
- VMware vSphere Fault Tolerance

Hitachi Vantara



Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA
HitachiVantara.com | community.HitachiVantara.com

Contact Information
USA: 1-800-446-0744
Global: 1-858-547-4526
HitachiVantara.com/contact