Red Hat OpenShift with Hitachi Virtual Storage Software Block

Reference Architecture Guide

MK-SL-250-01
November 2022

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 the bare metal option.	November 21, 2022
Initial release	August 8, 2022

Reference Architecture Guide

This paper demonstrates configuration of the Red Hat OpenShift Container Platform (OCP) environment deployed on Hitachi Advanced Server with Hitachi Virtual Storage Software block (VSS block). It leverages the latest capabilities of data storage integrations and services to create, protect, and manage on-premises Kubernetes clusters and distributed workloads including those application services that require persistent storage. These are relevant to deployment configurations with Hitachi Advanced Server in a hybrid converged/ hyperconverged configuration and Hitachi VSS block offering with OpenShift based on bare metal or hybrid configurations. This paper covers persistent storage options that are available in various deployments using well-known and supported storage integrations between Hitachi Vantara and Red Hat OpenShift.

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

- Data Persistence

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

  The Hitachi Virtual Storage Software solutions include a Hitachi Storage CSI driver supported with Hitachi Storage Plug-in for Containers that integrates Hitachi Virtual Storage Software block with Red Hat OpenShift Platform to provide persistent storage for stateful container applications. This combination of integrations can meet the majority of persistent storage configurations and data services required that are encountered. Hitachi Advanced Server with Red Hat OpenShift provides the infrastructure and integrations needed to successfully provide container services to your application teams.

- Compute Platform

  With a wide range of applications that are stateful or stateless, a wide range of flexible compute platforms are also necessary to match memory requirements as well as CPU requirements.

  The type of compute technology is also a consideration for licensing costs. Hitachi Vantara provides different compute options from the 1U/2U dual socket Hitachi Advanced Server DS120/DS220 to the 2U quad socket Hitachi Advanced Server DS240 or Hitachi Advanced Server HA810 and HA820 that deliver unparalleled compute density and efficiency to meet the needs of the most demanding high-performance applications in the data center.

- Network Connectivity

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

- Infrastructure Management

  Having a robust and flexible infrastructure without efficient management only increases operational inefficiencies. This solution provides the ability to increase innovation and agility.

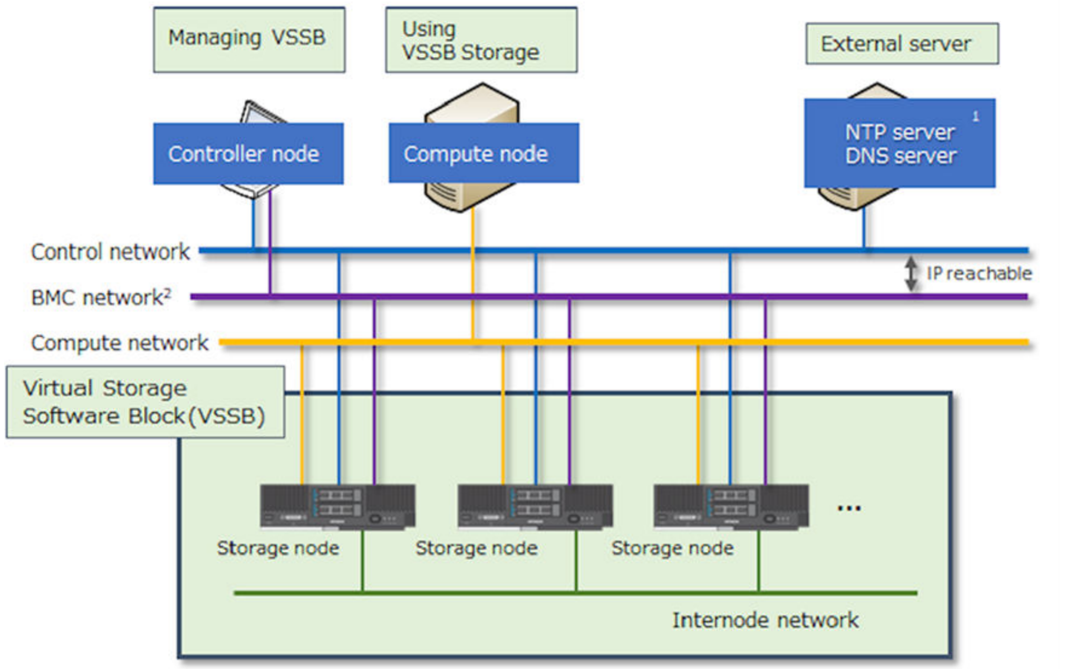
This reference architecture also provides the reference design for a build-your-own Red Hat OpenShift Container Platform environment using Hitachi Virtual Storage Software block. While this reference design is specific to Hitachi Virtual Storage Software block storage solutions, it can easily be used for building your own container platform with other Hitachi Virtual Storage Platform systems.

The intended audience of this document is IT administrators, system architects, consultants, and sales engineers who assist in planning, designing, and implementing Hitachi Virtual Storage Software block with OpenShift Container Platform solutions.

Hitachi Virtual Storage Software block overview

Hitachi Virtual Storage Software block is a storage software product that builds and sets up a virtual storage system from multiple general-purpose servers. The Virtual Storage Software block storage system is two-layer software-defined storage (SDS) in which storage nodes are separated from the compute nodes (disaggregated storage).

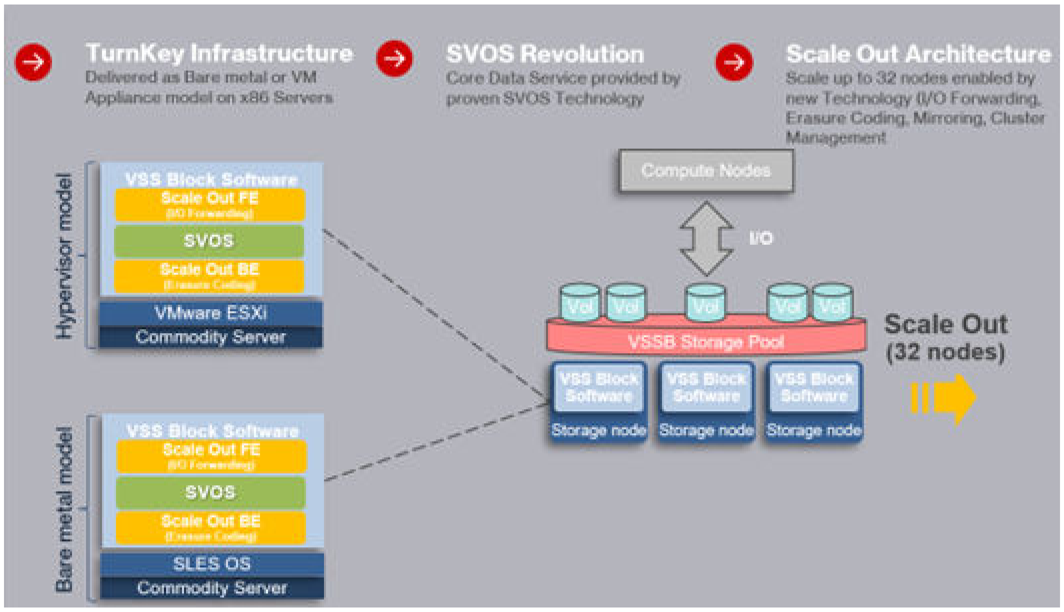
The following figure shows the configuration of Virtual Storage Software block.



1. Storage node and NTP/DNS server can be configured in different subnets. For another subnet configuration, the NTP/DNS server must be reachable via the Default Gateway.
2. The same network can be used for both the control network and BMC network.

To maximize return on investment (ROI) and deliver long-term scaling, Hitachi has redesigned Hitachi Storage Virtualization Operating System RF (SVOS RF) to run on Hitachi Advanced Server systems. By running on your server of choice, you can integrate your core enterprise storage platform with your existing hypervisor or bare metal infrastructure. This gives you the ability to run the latest distributed application alongside traditional business applications on a scale-out infrastructure, allowing you to scale performance and capacity independently to consolidate even more workloads.

The following figure shows an architectural diagram of Virtual Storage Software block.

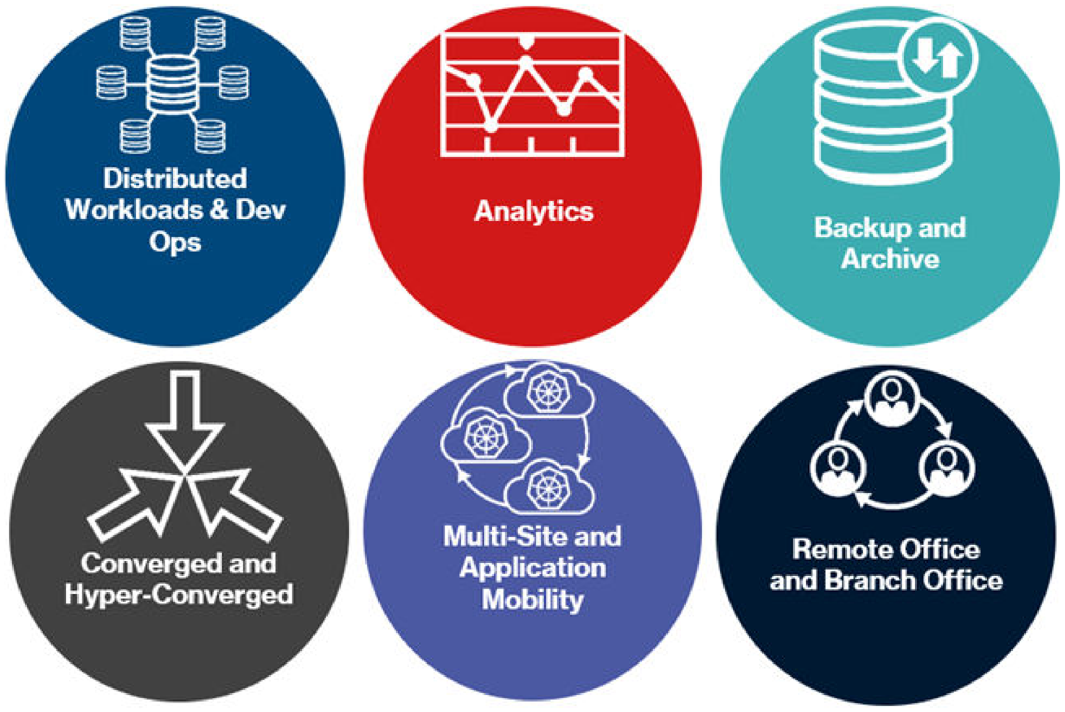


The Hitachi SDS offering is a true enterprise solution. With years of research and development, our patented Hitachi Polyphase Erasure Coding brings our legendary reliability into the SDS marketplace. By allowing data to be balanced across the entire cluster, SDS reduces the cost and complexity of management while offering the most resilient solution.

Use cases

To modernize the digital core, Virtual Storage Software block provides an SDS option along with typical Hitachi Virtual Storage Platform (VSP) systems to drive customer expectations of data mobility for multi-site environments, expanding on Hitachi enterprise storage capabilities to continue to deliver the enterprise reliability and performance you can expect from the Hitachi brand.

The following figure shows some of the use cases for this solution.


Distributed Workloads & Dev Ops
Analytics
Backup and Archive
Converged and Hyper-Converged
Multi-Site and Application Mobility
Remote Office and Branch Office

Key features

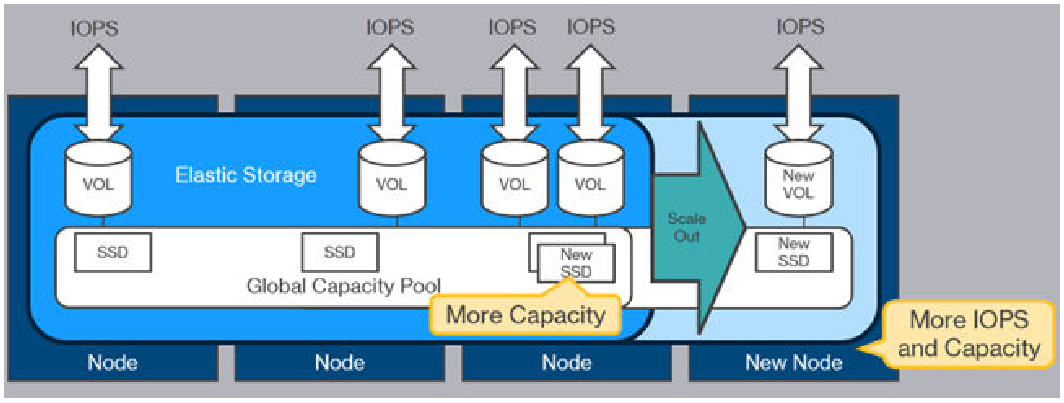
Traditional storage solutions for distributed containerized workloads such as OpenShift are too complex and have numerous limitations such as a limited number of volumes and automatic provisioning of storage to meet the dynamic needs of containers. The Hitachi SDS solution targets these limitations by abstracting storage software from its hardware and providing key features that enable a cloud-native storage solution tightly integrated into container orchestration frameworks optimized for distributed workloads.

Distributed storage

Running on x86 servers powered by Hitachi Storage Virtualization Operating System RF (SVOS RF) continues to be on the leading edge of storage virtualization, pools, tiering, deduplication and compression, and storage replication technologies. It demonstrates the ability to evaluate an underlying hardware technology and conceptualize its implementation across different industries before bringing it to market.

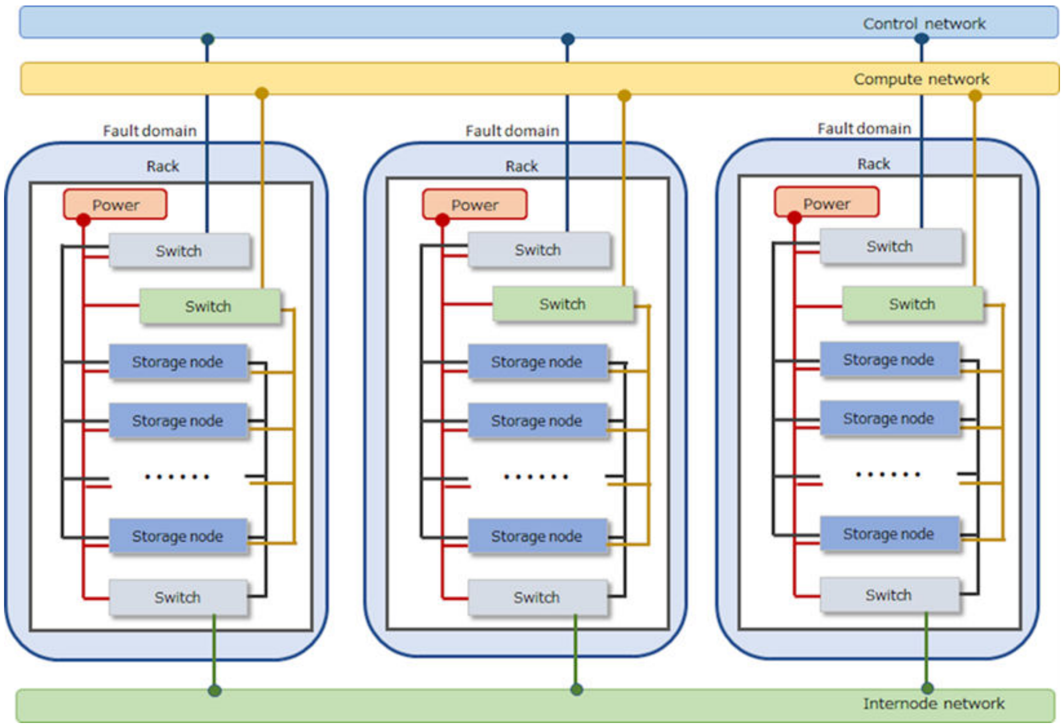
Scale-out by design

Linearly scale performance and capacity from 37 TB all the way up to 3.38 PB using a disaggregated storage cluster that uses a non-hyperconverged infrastructure (HCI). Resources can be added to an existing cluster by scaling out (adding more nodes) to increase aggregate IOPS and capacity.



Fault domains

Fault domains are groups of storage nodes that share hardware such as the same power supply and network switches. Having fault domains enables storage systems to continue operations if a failure occurs in one of the fault domains.

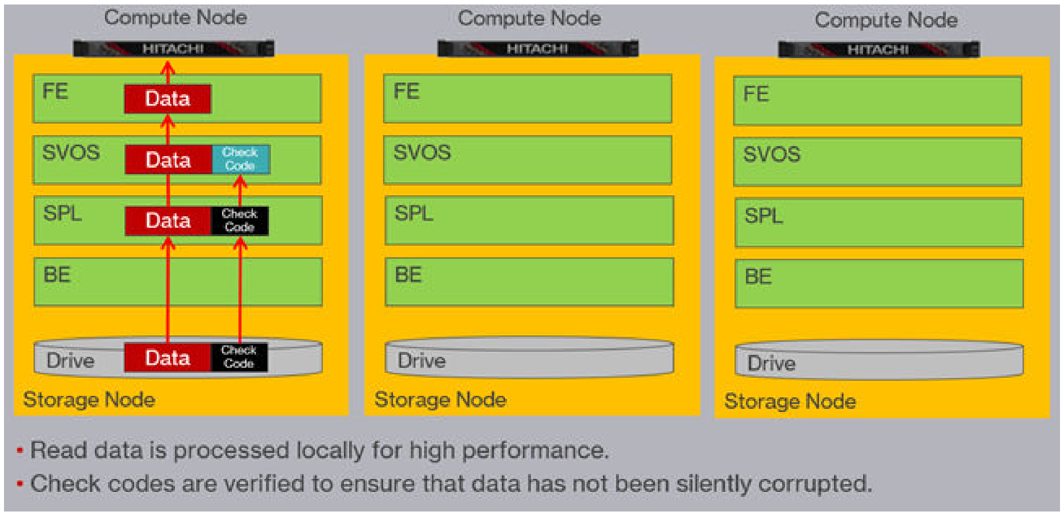


I/O flow read/write optimization

Hitachi Virtual Storage Software block is designed to take full advantage of data localization to improve performance for both reads and writes.

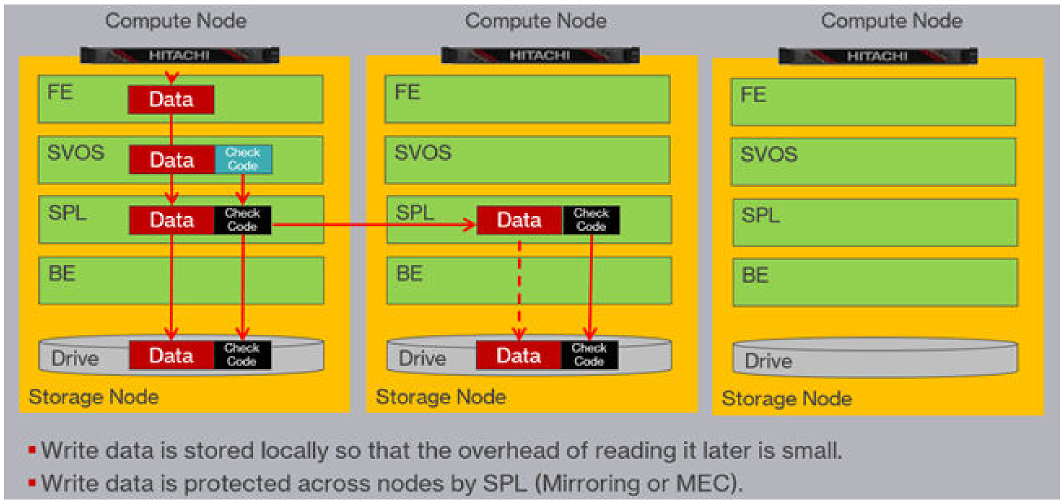
I/O read optimization

The following figure illustrates I/O read optimization.



I/O write optimization

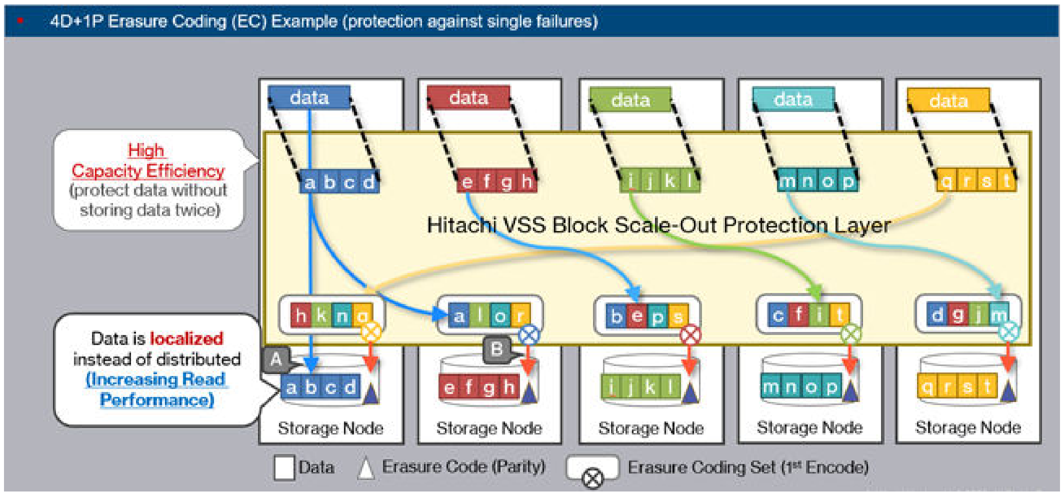
The following figure illustrates I/O write optimization.



Hitachi Polyphase Erasure Coding

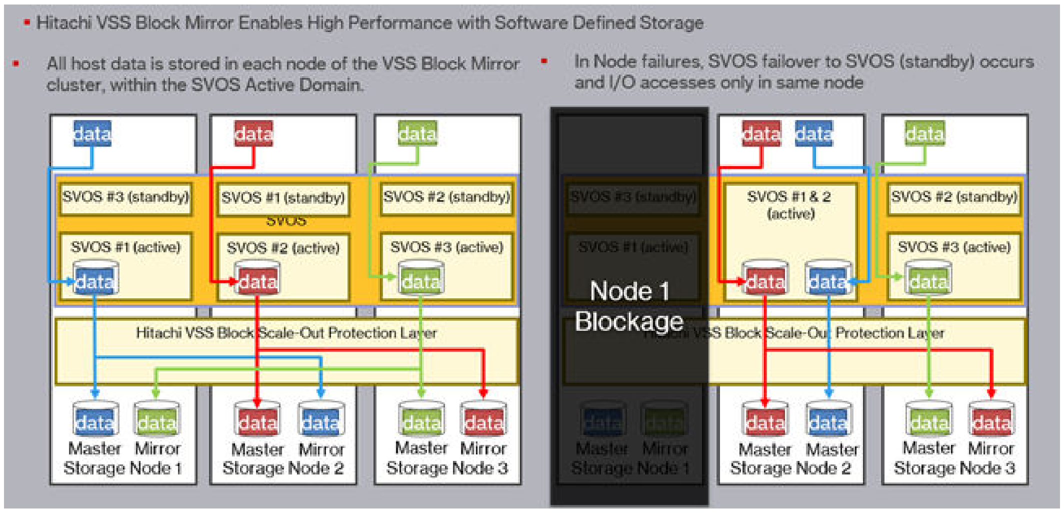
Hitachi Polyphase Erasure Coding (HPEC) features a different approach to adapting software data protection. Hitachi carefully evaluated erasure coding through years of research and developed a unique patented approach (US# 10,185,624 B2 and 10,496,479 B2). This approach improved crucial elements of data and parity placement, which efficiently improved latency within Hitachi Polyphase Erasure Coding, accelerated read performance, and lowered storage resource overhead. These necessary enhancements provide a place where your data can be synthesized that so you can create additional value while increasing the reliability of your Hitachi-based software-defined storage solution.

HPEC can be configured in a 4D+1P RAID configuration, which protects against single node failures and 4D+2P, which protects against double node failures.


4D+1P Erasure Coding (EC) Example (protection against single failures)
High Capacity Efficiency (protect data without storing data twice)
Data is localized instead of distributed (Increasing Read Performance)
Hitachi VSS Block Scale-Out Protection Layer
data
a b c d
e f g h
i j k l
m n o p
q r s t
h k n q
a l o r
b e p s
c f i t
d g j m
A
B
Storage Node
Data
Erasure Code (Parity)
Erasure Coding Set (1st Encode)

Mirroring

Mirroring mode is available where all data is stored multiple times resulting in low-capacity efficiency but high read and write performance, which is ideal when performance matters.

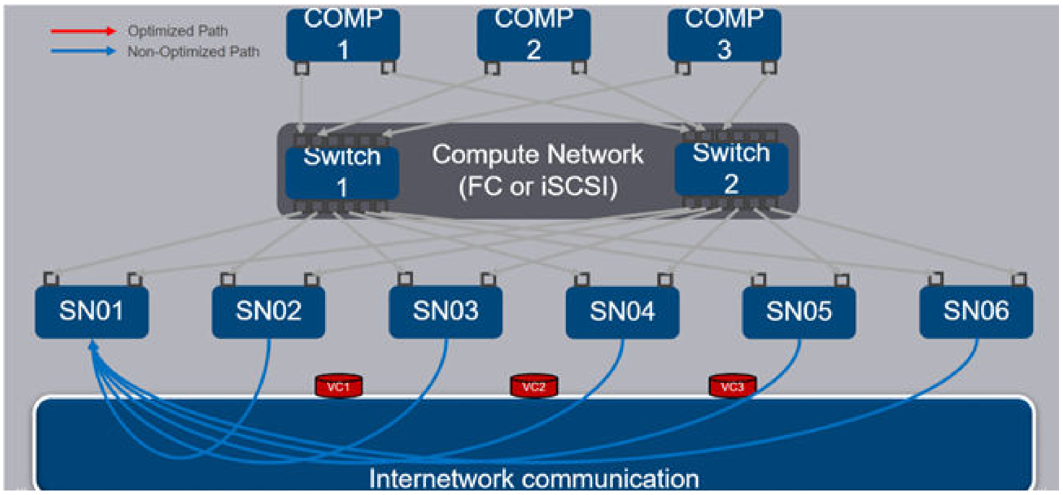

Hitachi VSS Block Mirror Enables High Performance with Software Defined Storage
All host data is stored in each node of the VSS Block Mirror cluster, within the SVOS Active Domain.
In Node failures, SVOS failover to SVOS (standby) occurs and I/O accesses only in same node
data
SVOS #3 (standby)
SVOS #1 (standby)
SVOS #2 (standby)
SVOS #1 (active)
SVOS #2 (active)
SVOS #3 (active)
SVOS #1 & 2 (active)
Hitachi VSS Block Scale-Out Protection Layer
Node 1 Blockage
Master
Mirror
Storage Node 1
Storage Node 2
Storage Node 3

Standard front-end protocols and multipathing

Proprietary host drivers are not required because standard 32 Gbps Fibre Channel and 10/25 Gbps iSCSI protocols are natively supported. OS Native Multipath (Device Mapper, MPIO) is supported.

Hitachi Polyphase Asymmetric Logical Unit Access (ALUA)

Using Pseudo Active/Active configuration, I/O can be sent to ports on any storage processor that defines the ownership of the LUN and therefore its optimized path. Non-owner processors can send and receive I/O to a given LUN as well; however, traffic must first be sent via the optimized path. In addition, true Pseudo Active/Active I/O can also be sent via a back-end channel, which in this case can be the Hitachi Virtual Storage Software block internode network.



Compute host I/O forwarding

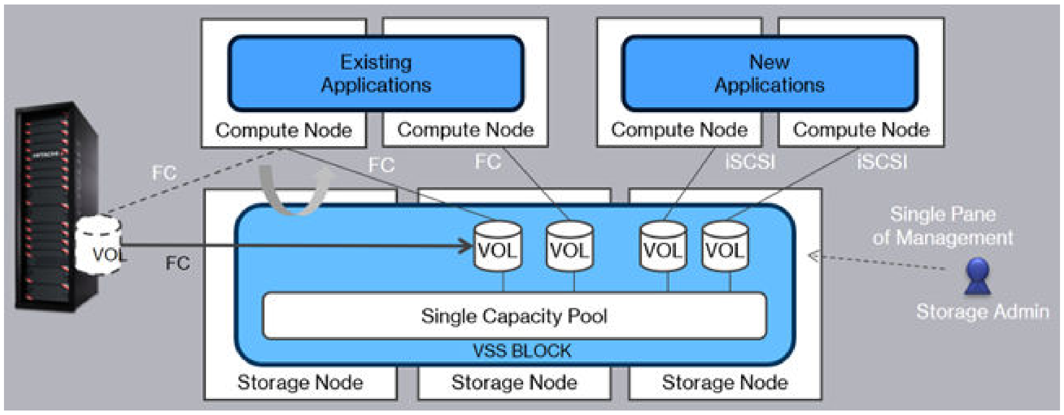
I/O forwarding is available so compute hosts can perform I/O operations to non-priority paths in case of failures.

Reserve rebuild capacity

With a fixed rebuild policy, capacity can be specified so rebuilds are possible and the capacity is not used for user data such as with a variable policy. Also, the number of drives for rebuild capacity can be specified. For example, in a mirrored configuration with three nodes, with four disks per node and the fixed rebuild policy set to two drives, two drive failures can be sustained before the node is lost.

Seamless data migration across Hitachi storage systems

With Hitachi Virtual Storage Software block, minimally disruptive migration from existing Hitachi storage systems using Fibre Channel is used to copy data and offload it to the storage layer so compute workloads are unimpeded. Single data paths can be cut over ensuring I/O continues even during migrations.



Auto-recovery

By default, storage nodes perform self-diagnosis and auto-recovery from a server failure caused by software components (such as firmware or drivers) or by a temporary failure of the internode network.

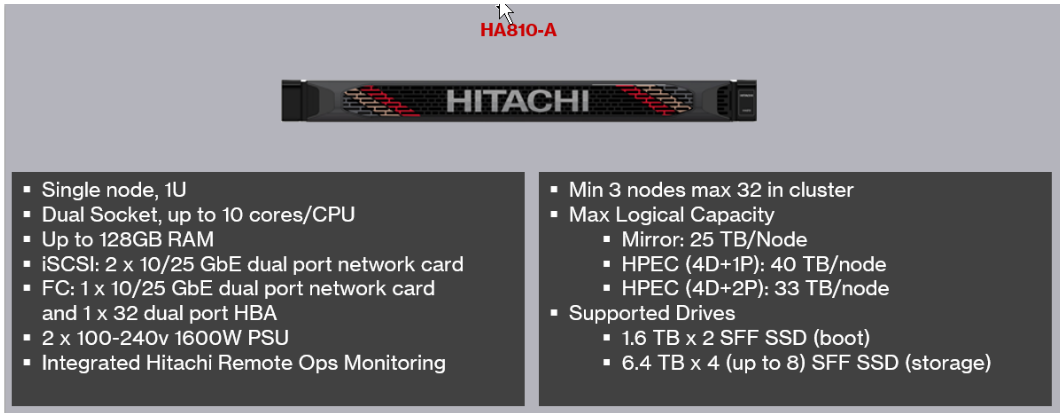
Storage-ready nodes

Virtual Storage Software block ready nodes are a quick, simple to deploy, and reliable way of building a distributed storage platform. They combine preconfigured and validated Hitachi HA servers powered by Intel® Xeon® processors, and they include the software-defined storage ability of Virtual Storage Software block. Remove the guesswork of building an SDS infrastructure and accelerate your deployment of new applications.

This document shows some of the storage ready nodes. See the *VSSB Hardware Compatibility Reference Guide* for the full details of all the ready nodes and VSSB versions supported across the hypervisor and bare metal models.

Hitachi Advanced Server HA810

Purpose-built for better performance, high density and power efficiency, this general-purpose dual-processor server delivers a balance of compute and storage capacity with the flexibility to power a wide range of solutions and applications.

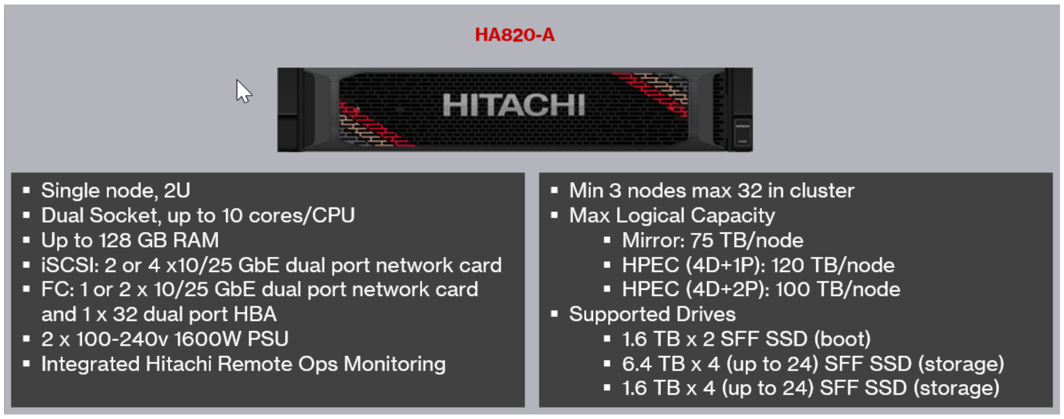


- Single node, 1U
- Dual Socket, up to 10 cores/CPU
- Up to 128GB RAM
- iSCSI: 2 x 10/25 GbE dual port network card
- FC: 1 x 10/25 GbE dual port network card and 1 x 32 dual port HBA
- 2 x 100-240v 1600W PSU
- Integrated Hitachi Remote Ops Monitoring

- Min 3 nodes max 32 in cluster
- Max Logical Capacity
  - Mirror: 25 TB/Node
  - HPEC (4D+1P): 40 TB/node
  - HPEC (4D+2P): 33 TB/node
- Supported Drives
  - 1.6 TB x 2 SFF SSD (boot)
  - 6.4 TB x 4 (up to 8) SFF SSD (storage)

Hitachi Advanced Server HA820

Adaptable for diverse workloads and environments, the secure 2U HA820 delivers world-class performance with the right balance of expandability and scalability. Designed for supreme versatility and resiliency while being backed by a comprehensive warranty makes it ideal for multiple environments from containers to cloud to big data.

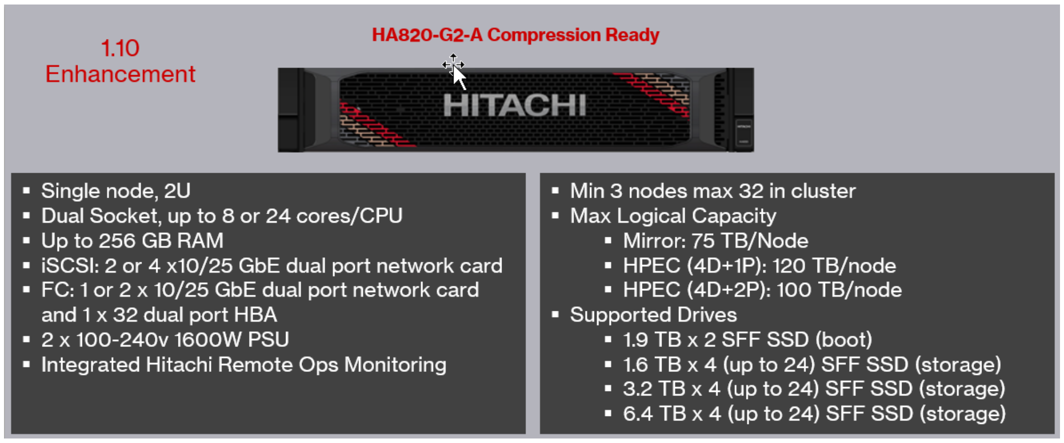


- Single node, 2U
- Dual Socket, up to 10 cores/CPU
- Up to 128 GB RAM
- iSCSI: 2 or 4 x10/25 GbE dual port network card
- FC: 1 or 2 x 10/25 GbE dual port network card and 1 x 32 dual port HBA
- 2 x 100-240v 1600W PSU
- Integrated Hitachi Remote Ops Monitoring

- Min 3 nodes max 32 in cluster
- Max Logical Capacity
  - Mirror: 75 TB/node
  - HPEC (4D+1P): 120 TB/node
  - HPEC (4D+2P): 100 TB/node
- Supported Drives
  - 1.6 TB x 2 SFF SSD (boot)
  - 6.4 TB x 4 (up to 24) SFF SSD (storage)
  - 1.6 TB x 4 (up to 24) SFF SSD (storage)

Hitachi Advanced Server HA820 G2

Hitachi Advanced Server HA820 G2 is a high-performance two-socket rackmount server designed for optimal performance and power efficiency. This allows owners to upgrade computing performance without overextending power consumption and offers non-latency support to virtualization environments that require maximum memory capacity. Hitachi Advanced Server HA820 G2 provides flexible I/O scalability for today's diverse data center application requirements.


1.10
Enhancement
HA820-G2-A Compression Ready
HITACHI

- Single node, 2U
- Dual Socket, up to 8 or 24 cores/CPU
- Up to 256 GB RAM
- iSCSI: 2 or 4 x10/25 GbE dual port network card
- FC: 1 or 2 x 10/25 GbE dual port network card and 1 x 32 dual port HBA
- 2 x 100-240v 1600W PSU
- Integrated Hitachi Remote Ops Monitoring

- Min 3 nodes max 32 in cluster
- Max Logical Capacity
  - Mirror: 75 TB/Node
  - HPEC (4D+1P): 120 TB/node
  - HPEC (4D+2P): 100 TB/node
- Supported Drives
  - 1.9 TB x 2 SFF SSD (boot)
  - 1.6 TB x 4 (up to 24) SFF SSD (storage)
  - 3.2 TB x 4 (up to 24) SFF SSD (storage)
  - 6.4 TB x 4 (up to 24) SFF SSD (storage)

Red Hat OpenShift overview

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 to define storage resources.

Scalability

Applications running on Red Hat OpenShift can scale to thousands of instances across hundreds of nodes in seconds.

Flexibility

Red Hat OpenShift simplifies deployment and management of a hybrid infrastructure, giving you the flexibility to have a self-managed or fully managed service, running on-premises or in cloud and hybrid environments.

Open-source standards

Red Hat OpenShift incorporates Open Container Initiative (OCI) containers and Cloud Native Computing Foundation-certified Kubernetes for container orchestration, in addition to other open-source technologies.

Container portability

Container images built on the OCI industry standard ensure portability between developer workstations and Red Hat OpenShift production environments.

Enhanced developer experience

Red Hat OpenShift offers a comprehensive set of developer tools, multilanguage support, command-line and integrated development environment (IDE) integrations. Features include continuous integration/continuous delivery (CI/CD) pipelines based on Tekton and third-party CI/CD solutions, service mesh, serverless capabilities, and monitoring and logging capabilities.

Automated installation and upgrades

Automated installation and over-the-air platform upgrades are supported in the cloud with Amazon Web Services, Google Cloud Platform, IBM Cloud, and Microsoft Azure, and on-premises using vSphere, Red Hat OpenStack® Platform, Red Hat Virtualization, or bare metal. Services used from the OperatorHub can be deployed fully configured and are upgradable with one click.

Automation

Streamlined and automated container and app builds, deployments, scaling, health management, and more are included.

Edge architecture support

Red Hat OpenShift enhances support for smaller footprint topologies in edge scenarios, such as 3-node clusters, single-node OpenShift, and remote worker nodes, which better map to varying physical size, connectivity, and availability requirements of different edge sites. The edge use cases are further enhanced with support for Red Hat OpenShift clusters on ARM architecture, commonly used for low-power-consumption devices.

Multicluster management

Red Hat OpenShift with Red Hat Advanced Cluster Management for Kubernetes can easily deploy apps, manage multiple clusters, and enforce policies across clusters at scale.

Advanced security and compliance

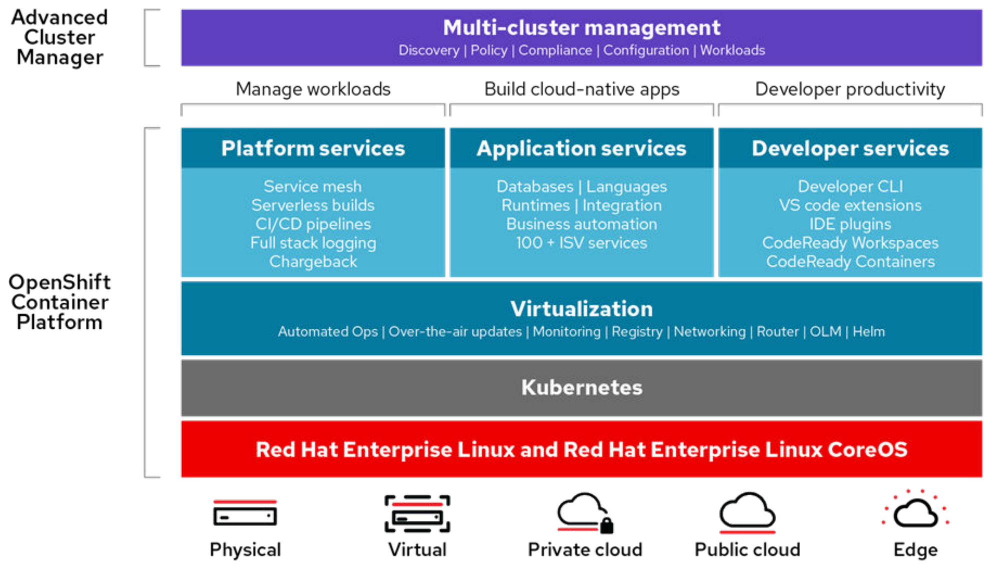
Red Hat OpenShift offers core security capabilities such as access controls, networking, and enterprise registry with a built-in scanner. Red Hat Advanced Cluster Security for Kubernetes enhances this with security capabilities such as runtime threat detection, full lifecycle vulnerability management, and risk profiling.

Persistent storage

Red Hat OpenShift supports a broad spectrum of enterprise storage solutions, including Red Hat OpenShift Data Foundation, for running both stateful and stateless apps.

Robust ecosystem

An expanding ecosystem of partners provides a wide variety of integrations. Third parties deliver additional storage and network providers, IDE, CI, integrations, independent software vendor solutions, and more.



Deployment methods

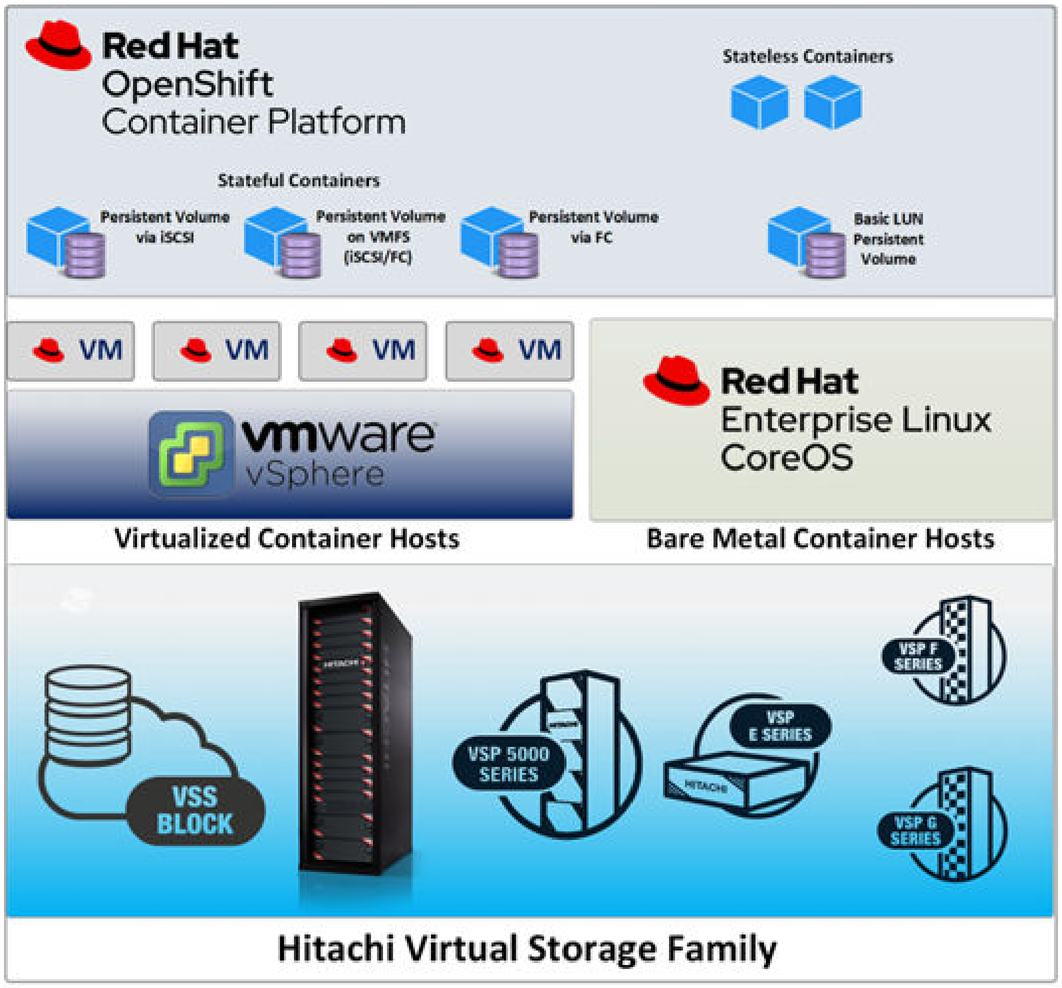
There are multiple deployment methods for Red Hat OCP based on the hardware configuration supporting the environment. These deployment methods include manual deployments using the User Provisioned Infrastructure (UPI) for customized deployments or fully automated deployments using the Installer Provisioned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

Validated OpenShift deployments

Hitachi has tested and validated various deployments of OpenShift on bare metal hosts, virtualized hosts, and mixed hybrid hosts backed with storage systems from the Hitachi Virtual Storage Platform family.

Solution overview

Red Hat OpenShift is a successful container orchestration platform that is available with Hitachi Advanced Server models. The following figure shows a high-level diagram of OpenShift managing containers and persistent volumes on the Hitachi Advanced Server stack with Hitachi Virtual Storage Platform systems.


Red Hat OpenShift Container Platform
Stateless Containers
Stateful Containers
Persistent Volume via iSCSI
Persistent Volume on VMFS (iSCSI/FC)
Persistent Volume via FC
Basic LUN Persistent Volume
VM
VM
VM
VM
vmware vSphere
Red Hat Enterprise Linux CoreOS
Virtualized Container Hosts
Bare Metal Container Hosts
VSS BLOCK
VSP 5000 SERIES
VSP E SERIES
VSP F SERIES
VSP G SERIES
Hitachi Virtual Storage Family

You can deploy OpenShift on bare metal hosts, virtual hosts, or both. In some cases, the primary nodes are virtualized while the worker nodes can be a hybrid of bare metal and virtual. Depending on the deployment purposes, different deployments can be used. For this reference architecture, the OpenShift clusters use a fully virtualized deployment where both the primary and worker nodes are virtualized.

Solution components

This section describes the key hardware and software components used for this environment.

Hardware components

The following table lists the hardware tested for this reference architecture.

The solution used specific features based on the following hardware. You can use Hitachi Advanced Server DS120/DS220/DS225/DS240/HA810/HA820, or any qualified server platform.

Hardware	Description	Version	Quantity
Hitachi Advanced Server DS120 (VMware compute nodes)	▪ 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 SSDs DC P4800X (375 GB, U.2) NVMe ▪ vSAN Capacity Tier: 10 × Intel SSDs DC P4510 (4 TB, U.2) NVMe	BMC: 4.68.06 BIOS: 3B19.H00	4
Hitachi Advanced Server DS120 (OCP bare metal node)	▪ 2 × Intel Xeon 4210 10-core 2.20 GHz processors ▪ 16 × 16 GB DIMM, 256 GB memory ▪ 1 × 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	1
Hitachi Advanced Server HA820 (Hitachi Virtual Storage Software block storage nodes – Hypervisor model)	▪ 2 × Intel Xeon Gold 6226 12-core 2.70GHz processors ▪ 12 × 16 GB DIMMs, 192 GB memory ▪ 2 × 1.6 TB SAS SSDs for boot ▪ 12 × 800 GB SAS SSDs ▪ 1 × HPE SN1600Q 32 Gb dual port Fibre Channel HBA ▪ 1 × HPE 10 Gb quad port NIC	iLO: 2.55 BIOS: U30 v2.54	3
Hitachi Advanced Server HA820 (Hitachi Virtual Storage Software	▪ 2 × Intel Xeon Gold 6226 12-core 2.70GHz processors ▪ 12 × 16 GB DIMMs, 192 GB memory ▪ 2 × 1.6 TB SAS SSDs for boot	iLO: 2.55 BIOS: U30 v2.54	3

Hardware	Description	Version	Quantity
block nodes – Bare metal model)	▪ 12 × 800 GB SAS SSDs ▪ 1 × HPE SN1600Q 32 Gb dual port Fibre Channel HBA ▪ 1 × HPE 10 Gb quad port NIC		
Cisco Nexus 9332C switch (spine)	▪ 32-port 40/100 GbE ▪ 2-port 1/10 GbE	NXOS 9.3.5	2
Cisco Nexus 93180YC-FX switch (leaf)	▪ 48-port 10/25 GbE ▪ 6-port 40/100 GbE	NXOS 9.3.5	2
Cisco Nexus 92348 switch	▪ 48-port 1 GbE ▪ 4-port 1/10/25 GbE ▪ 2-port 40/100 GbE	NXOS 9.3.5	1

Software components

The following table lists the key software components used in this solution.

Software	Version
Hitachi Storage Virtualization Operating System RF	8.x
Hitachi Virtual Storage Software block	1.11
Red Hat OpenShift Container Platform (OCP)	4.10
Hitachi Storage Plug-in for Containers	3.10
VMware vSphere	7.0 Update 2 or newer
SUSE Linux Enterprise	15

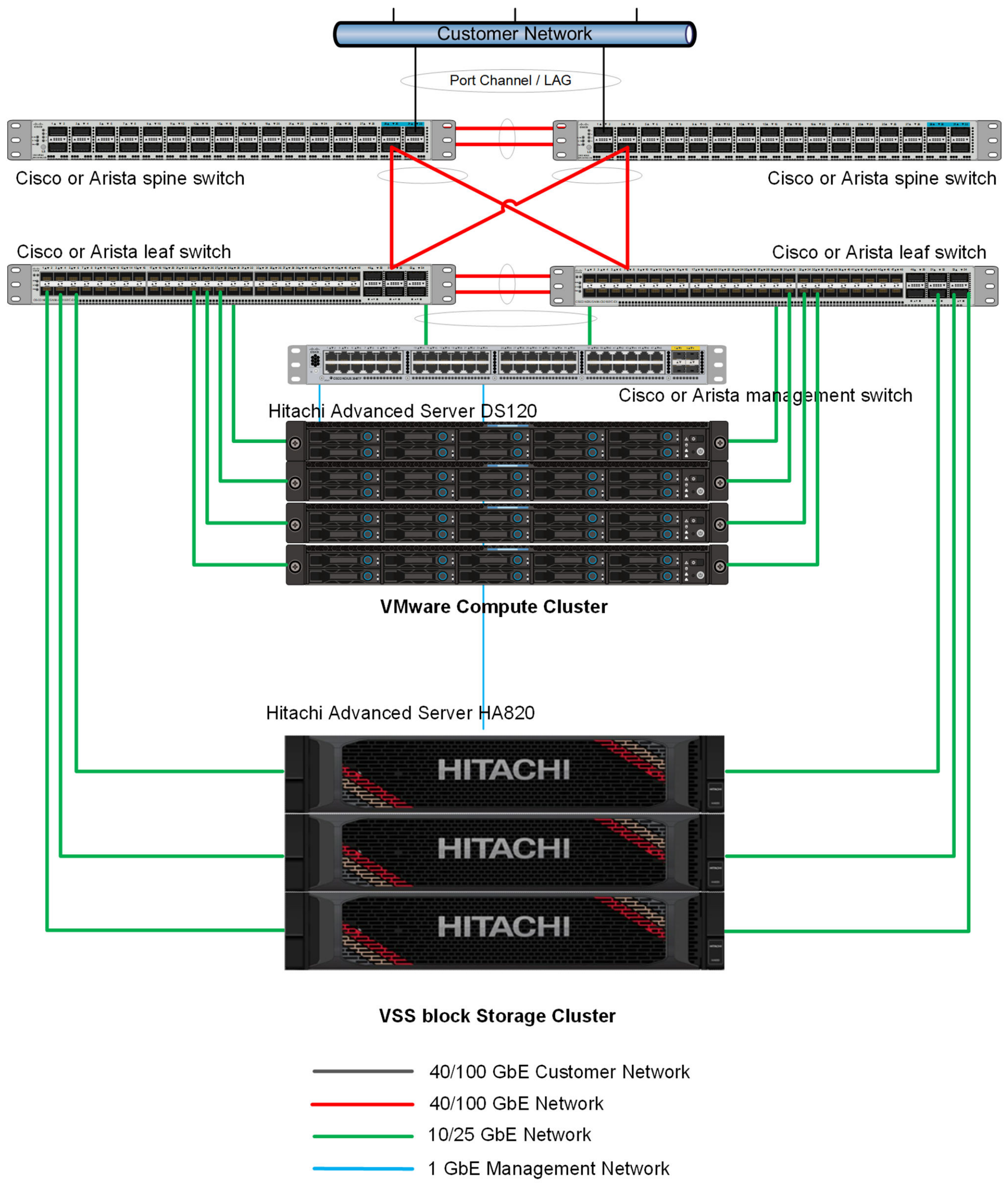
Solution design

This is a detailed solution example of Hitachi Advanced Server, Hitachi Virtual Storage Software block, and Red Hat OpenShift.

Solution infrastructure components

The following figure shows a high availability configuration of Hitachi Virtual Storage Software block 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 Advanced Server DS120 compute nodes
- Six Hitachi HA820 servers configured as Hitachi Virtual Storage Software block nodes in two separate VSSB clusters. Three servers for VSSB Hypervisor model and three servers as VSSB bare metal. For simplicity, figure below shows only three of the six servers on the VSSB cluster.

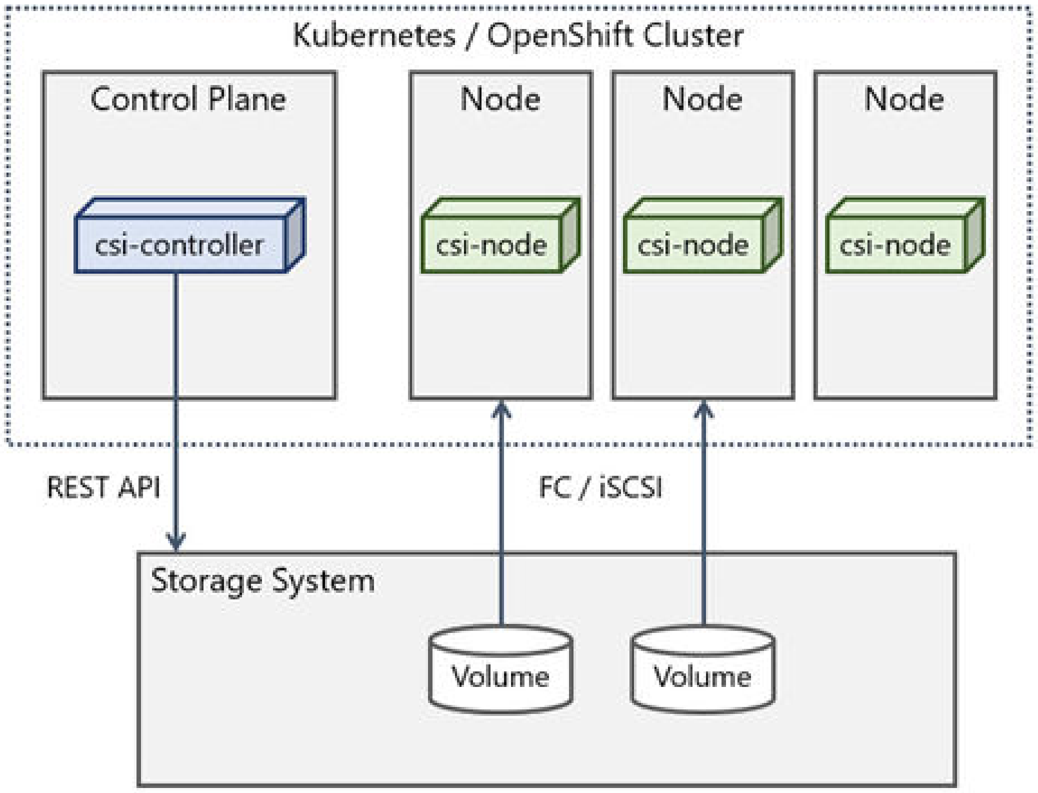

Customer Network
Port Channel / LAG
Cisco or Arista spine switch
Cisco or Arista spine switch
Cisco or Arista leaf switch
Cisco or Arista leaf switch
Cisco or Arista management switch
Hitachi Advanced Server DS120
VMware Compute Cluster
Hitachi Advanced Server HA820
HITACHI
HITACHI
HITACHI
VSS block Storage Cluster
40/100 GbE Customer Network
40/100 GbE Network
10/25 GbE Network
1 GbE Management Network

Hitachi Storage Plug-in for Containers

Hitachi Storage Plug-in for Containers is a software component that contains libraries, settings, and commands that you can use to create a container to run your 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 via the Fibre Channel protocol. Storage Plug-in for Containers can also provide virtual environments with persistent storage if iSCSI is used.

Storage Plug-in for Containers integrates Kubernetes or OpenShift with Hitachi storage systems such as Hitachi Virtual Storage Software block or Hitachi Virtual Storage Platform systems using Container Storage Interface (CSI).

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



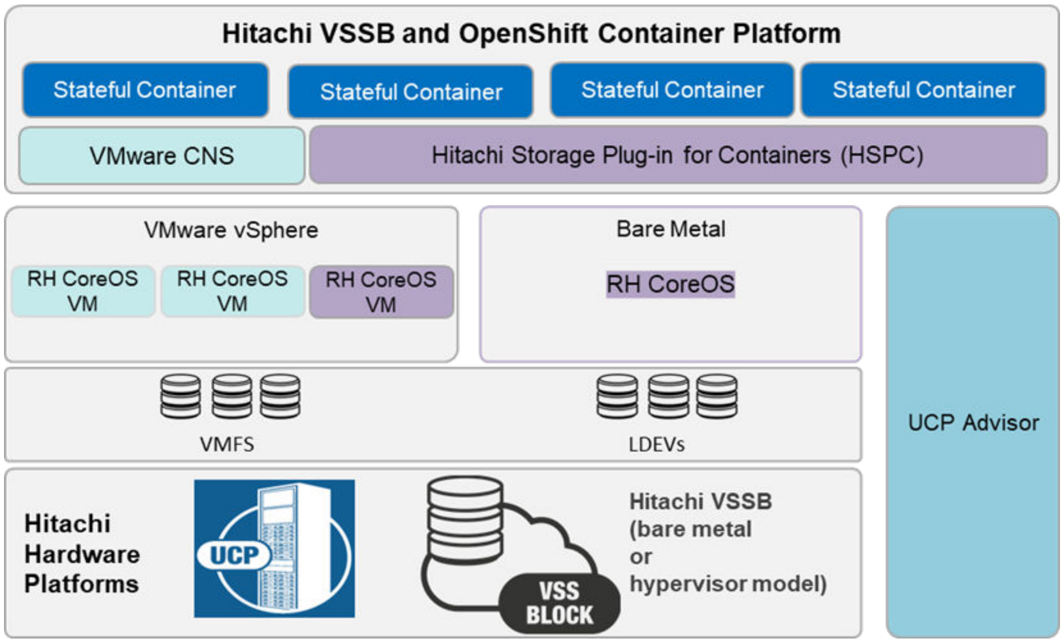
Deploy OpenShift Container Platform

One OCP cluster has been configured to support the different use cases described in this guide, including deployment of containerized applications with Persistent Volumes across two VSSB Storage systems, one VSSB bare metal, and one hypervisor model.

A hybrid OCP cluster was configured, with virtual and bare metal worker nodes and the following configuration:

- The virtual worker nodes had access to vSAN storage, as well as VMFS datastores supported by Hitachi VSSB storage system
  - Details about the StorageClasses are provided in each of the use cases part of the solution implementation and validation.
  - The bare metal worker node (worker-4) was configured with iSCSI connectivity to both VSSB (bare metal and hypervisor) storage systems.

The following figure illustrates a hybrid OpenShift Container Platform architecture on top of Hitachi UCP Platform with Hitachi VSSB storage.



The following table provides additional details about the primary and secondary nodes for the hybrid OCP cluster.

Node Name	Node Role	Device Type	OS-Image	Kubernetes Version
Router-LB	Load Balancer, Router	Existing infrastructure outside of UCP CI	CentOS	
jpc0-master-1	Master	VMs, hosted ona VMware cluster with DS120 G1, with vSAN, and VMFS datastores	Red Hat Enterprise Linux CoreOS 410.84	v1.23.5 v1.23.5
jpc0-master-2	Master			
jpc0-master-3	Master			
jpc0-worker-1	Worker			
jpc0-worker-2	Worker			
jpc0-worker-3	Worker			
jpc0-worker-4	Worker	Bare metal node (DS120)	Red Hat Enterprise Linux CoreOS 410.84	v1.23.5

Follow the latest Red Hat OpenShift official documentation to deploy OpenShift Container Platform (OCP) based on your environment and requirements.

Install and configure Hitachi Storage Plug-in for Containers

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

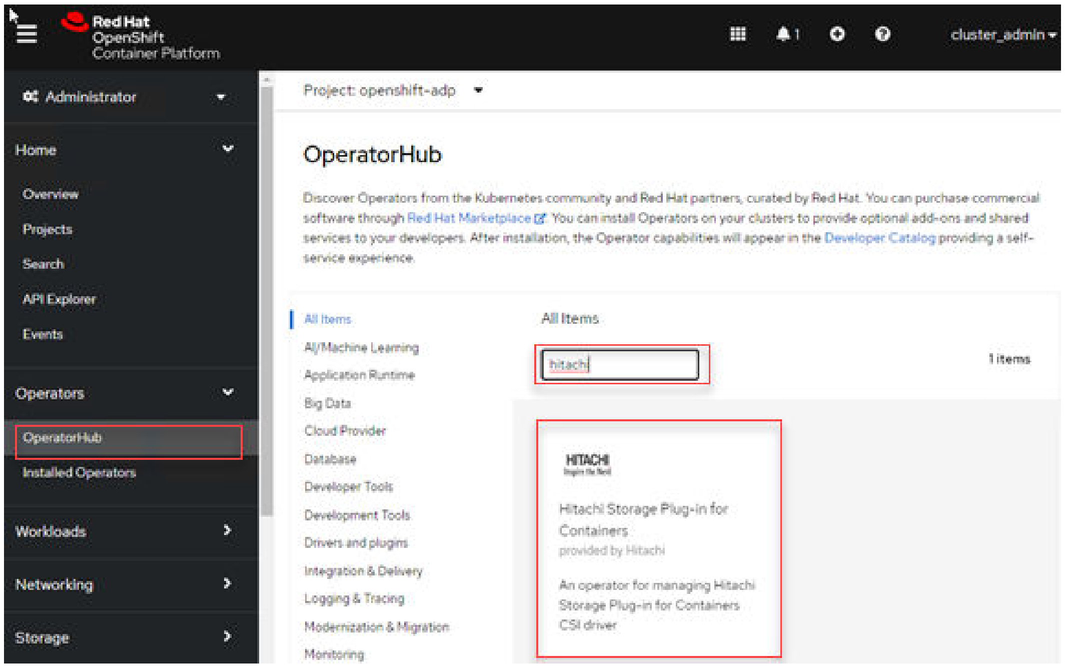
- Install Storage Plug-in for Containers
- Configure Secret settings to access Hitachi Virtual Storage Software block
- Configure StorageClass settings
- Configure Multipathing (Fibre Channel or iSCSI)

Specific steps to configure Persistent Volume Claims (PVC) and pods are covered as part of each of the use cases in Solution validation (on page 33).

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

Procedure

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



OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat Marketplace You can install Operators on your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a self-service experience.

Hitachi Storage Plug-in for Containers
provided by Hitachi

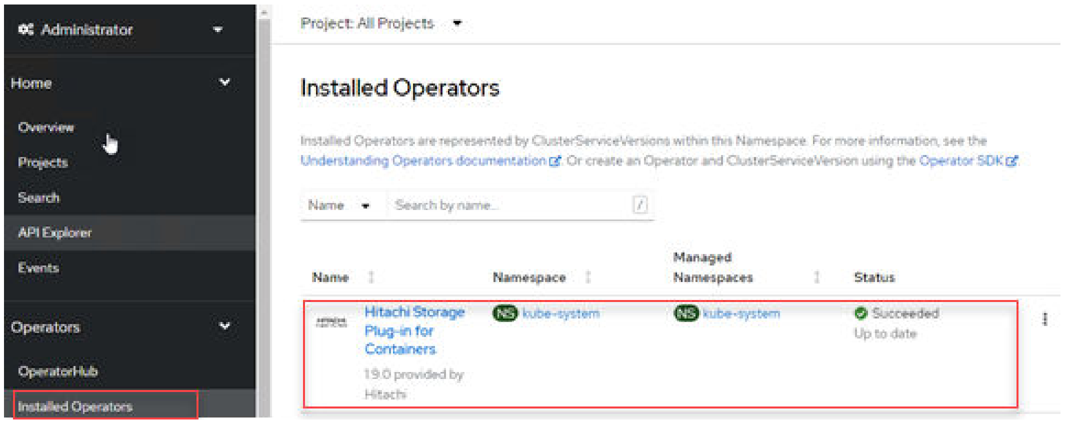
An operator for managing Hitachi Storage Plug-in for Containers CSI driver

2. Click **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 that the Operator status is **Succeeded** either using the console or the `oc get pods -n <namespace>` command. On the console, expand **Operators,** and then click **Installed Operators** to see the status of the plug-in.



Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the Understanding Operators documentation. Or create an Operator and ClusterServiceVersion using the Operator SDK.

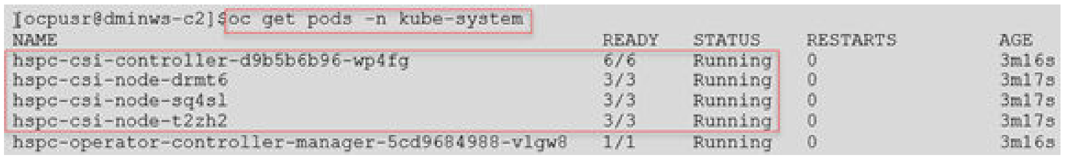
Name	Namespace	Managed Namespaces	Status
Hitachi Storage Plug-in for Containers 1.9.0 provided by Hitachi	NS kube-system	NS kube-system	Succeeded Up to date

4. To create the Storage Plug-in for Containers instance, select Hitachi Storage Plug-in for Containers, click **Create Instance** on the **Operator Details** page, and then click **Create**.
5. Verify that the status READY is true for the instance with the following command:

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

6. Verify that all the HSPC pods are in the Running status 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-sq4sl                               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
```

Result

Hitachi Storage Plug-in for Containers has been successfully installed. To make an advanced configuration, see Configuration of Storage Plug-in for Containers.

Configure secret

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

```
kind: Secret
apiVersion: v1
metadata:
  name: hspc-secret
  namespace: default
data:
  password: cGFzc3dvcmQ=
  url: aHR0cHM6Ly9sb2NhbGhvc3QK
  user:  b2NwdXNyMQ==
type: Opaque
```

The URL, user, and password are base64 encoded. The following is an example of how to get the base64 encoded information 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. Log in to the OCP console, and then select **Workloads** > **Secrets**.
2. Confirm the namespace in which you are creating the secret.
3. Click **Create** and then select **From YAML**. Then, either copy/paste the content of the secret manifest file or set the base64 encoded URL, user, and password, 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 manifest file provides information about the required parameters:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: hspc-vssb-sc
  annotations:
    kubernetes.io/description: Hitachi Storage Plug-in for Containers
provisioner: hspc.csi.hitachi.com
parameters:
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: default
  csi.storage.k8s.io/provisioner-secret-name: hspc-secret
  csi.storage.k8s.io/node-stage-secret-name: hspc-secret
  csi.storage.k8s.io/controller-expand-secret-name: hspc-secret
  csi.storage.k8s.io/node-publish-secret-namespace: default
  csi.storage.k8s.io/controller-publish-secret-name: hspc-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: default
  storageType: vssb
  csi.storage.k8s.io/node-publish-secret-name: hspc-secret
  connectionType: iscsi
  csi.storage.k8s.io/controller-expand-secret-namespace: default
  csi.storage.k8s.io/node-stage-secret-namespace: default
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

The following are additional details about some of these parameters:

- `provisioner`: For Storage Plug-in, the default is `hspc.csi.hitachi.com`.
- `storageType`: Must be set to `vssb`.
- `connectionType`: The connection type between storage and nodes, `fc` and `iscsi` are supported. If blank, `fc` is set.
- `fstype`: Set filesystem type to `ext4`.
- `secret-name`: Define the Hitachi Virtual Storage Software block secret name.
- `secret-namespace`: Enter the same namespace used for the secret.

You can create a StorageClass 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**, click **Edit YAML**, and then either copy/paste the content of the StorageClass manifest file or enter each of the corresponding settings.
3. Click **Create**.

Configure Multipathing

Multipathing should be enabled for worker nodes connected to Hitachi Virtual Storage Software block storage via Fibre Channel or iSCSI. 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 a Machine Config Operator (MCO) to the OCP cluster after it has been deployed.

Note: Applying the machine configuration will restart the worker nodes one at a time.

Consider the following before applying the multipath configuration:

- For Fibre Channel, verify that the Fibre Channel switches are configured with proper zoning for the compute worker nodes and that the Virtual Storage Platform storage systems are accessible to each other.
- For iSCSI, verify that the Hitachi Virtual Storage Software block storage is correctly configured for iSCSI and that the compute worker nodes can access the iSCSI targets. Also, for iSCSI, see 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 an additional package, just apply the configurations indicated in this section.

Configure multipathing for OCP worker nodes using MachineConfig:

To enable multipath for Hitachi Storage Plug-in, apply the following MachineConfig to the cluster. This enables multipathd (for Fibre Channel and iSCSI) needed by the Hitachi Virtual Storage Software block storage and Hitachi Storage Plug-in for Containers 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:

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

Configure multipathing using the OCP console:

Procedure

1. To apply a MachineConfig using the OCP console, log in to the OCP web console and navigate to **Compute** > **Machine Configs**. Click **Create Machine Config**, copy and paste the contents of the YAML file, and then 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,
ZGVmYXVsdHMgewogICAgdXNlcl9mcmllbmRseV9uYW1lcyB5ZXMKICAgIGZpbmRfbXVsdGlwYXRocyB5Z
XMKfQpkZXZpY2VzIHsKICAgIGRldmljZSB7CiAgICAgICAgdmVuZG9yIEhJVEFDSEkKICAgICAgICBwcm
9kdWN0IEhpLVNEUwogICAgICAgIHBhdGhfZ3JvdXBpbmdfcG9saWN5IGdyb3VwX2J5X3ByaW8KICAgICA
gICBwcmlvIGFsdWEKICAgICAgICBwYXRoX2NoZWNrZXIgcmVhZHNlY3RvcjAKICAgICAgICBub19wYXRo
X3JldHJ5IDYKICAgICAgICBkZXZfbG9zc190bW8gaW5maW5pdHkKICAgIH0KfQo=
          verification: {}
    systemd:
      units:
        - name: multipathd.service
          enabled: true
          state: started

        - name: iscsid.service
          enabled: true
          state: started
osImageURL: ""
```

For iSCSI without multipathing, use the following MachineConfig:

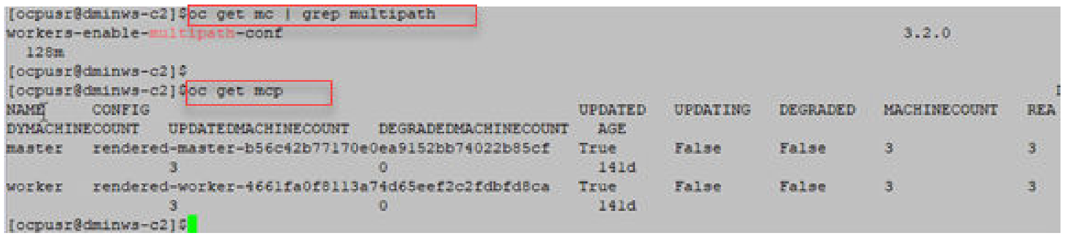
```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: workers-enable-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          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 suit your environment's needs, you can run the `echo -n “<string>” | base64 -d` command to decode the contents of the config file. To update the multipath.conf file, make your changes and then and re-encode it using base64.

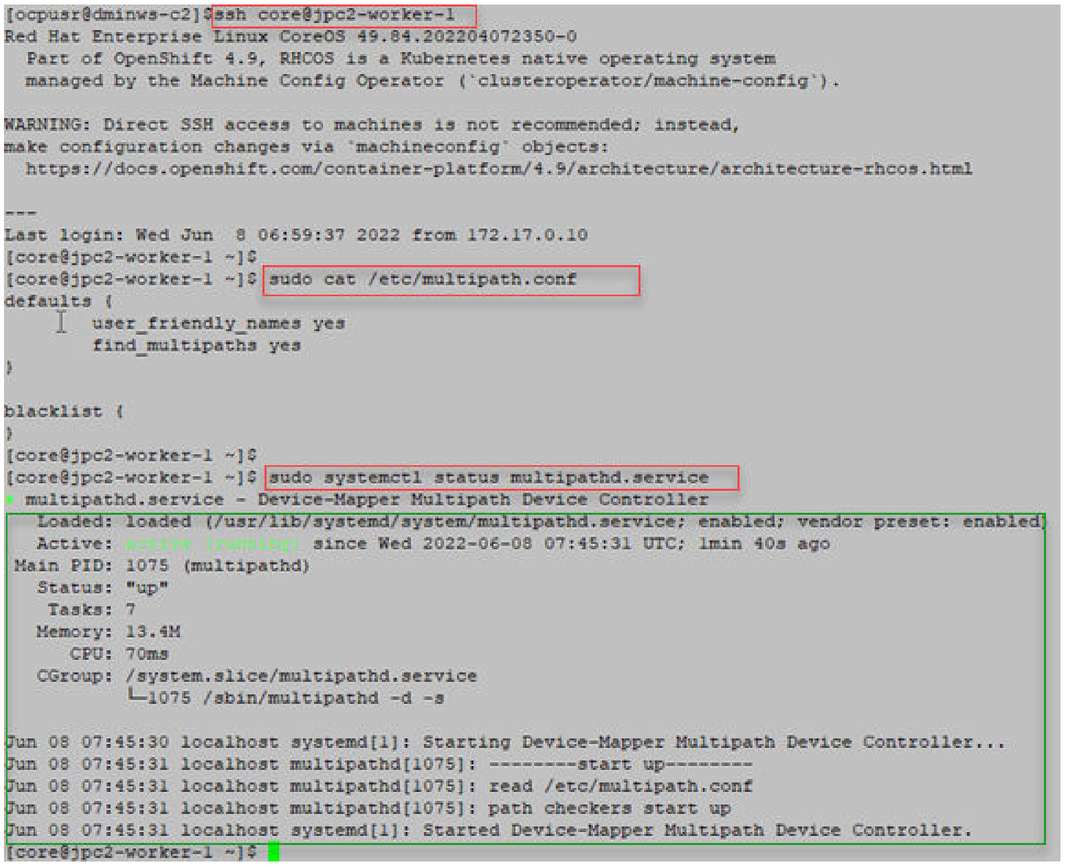
After the MachineConfig is created, every worker node is rebooted one at a time after the configuration is applied. It could take 20 - 30 minutes to apply the configuration to the worker nodes.

2. To verify that the machine config is applied, use the `oc get mcp` command and ensure that the machine config pool for workers is updated. In addition, SSH to the worker nodes to confirm that the `/etc/multipath.conf` folder has been created and that the multipathd service is running. For iSCSI verify that the iscsid service is running.

   The following are examples:



```
[ocpusr@dminws-c2]$oc get mc | grep multipath
workers-enable-multipath-conf                                                           3.2.0
  128m
[ocpusr@dminws-c2]$
[ocpusr@dminws-c2]$oc get mcp
NAME     CONFIG                                              UPDATED    UPDATING    DEGRADED    MACHINECOUNT    REA
DYMACHINECOUNT   UPDATEDMACHINECOUNT    DEGRADEDMACHINECOUNT   AGE
master   rendered-master-b56c42b77170e0ea9152bb74022b85cf    True       False       False       3               3
                 3                       0                      141d
worker   rendered-worker-4661fa0f8113a74d65eef2c2fdbfd8ca    True       False       False       3               3
                 3                       0                      141d
[ocpusr@dminws-c2]$
```



```
[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 ~]$
```

Hitachi Storage Plug-in for Containers and Hitachi Virtual Storage Software block host groups

Host groups required for Hitachi 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.

To use existing host groups, rename them according to Host group and iSCSI target naming rules.

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

Deploy Virtual Storage Software block

Deploy Hitachi Virtual Storage Software block based on the best practices outlined in the Hitachi Virtual Storage Software block documentation.

Installation and configuration

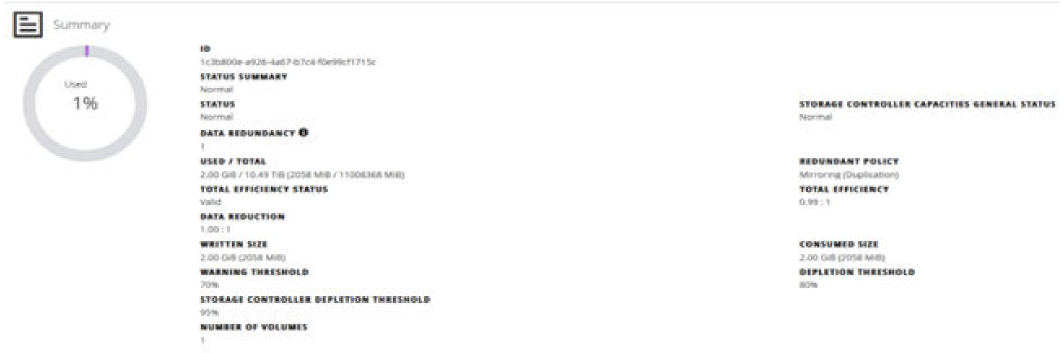
Some of the key installation and configuration options are highlighted in this section.

Redundant Policy

By default, the HPEC protection method is selected. For this solution, the Mirroring policy was chosen because it creates one copy of the data, which is ideal when performance is critical.

Redundant Type

By default, the protection type 4D+1P is selected. You can select either 4D+1P or 4D+2P depending on the minimum number of nodes in the storage cluster. Because the Redundant Policy is set to Mirroring in this solution, Duplication is selected.



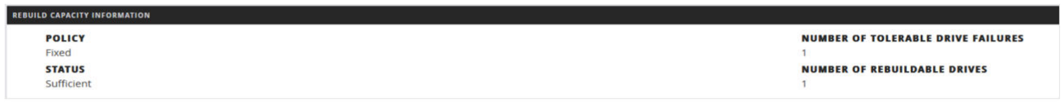
ID 1c3b800e-a926-4a67-b7c4-f0e99cf1715c	
STATUS SUMMARY Normal	
STATUS Normal	STORAGE CONTROLLER CAPACITIES GENERAL STATUS Normal
DATA REDUNDANCY 1	
USED / TOTAL 2.00 GiB / 10.49 TiB (2058 MiB / 11006368 MiB)	REDUNDANT POLICY Mirroring (Duplication)
TOTAL EFFICIENCY STATUS Valid	TOTAL EFFICIENCY 0.99 : 1
DATA REDUCTION 1.00 : 1	
WRITTEN SIZE 2.00 GiB (2058 MiB)	CONSUMED SIZE 2.00 GiB (2058 MiB)
WARNING THRESHOLD 70%	DEPLETION THRESHOLD 80%
STORAGE CONTROLLER DEPLETION THRESHOLD 95%	
NUMBER OF VOLUMES 1	

Access Protocol

All iSCSI-related options need to be configured to ensure that connectivity with compute nodes can be established.

Reserve Rebuild Capacity

By default this is set to Fixed. This enables specification of the number of disks to be reserved for rebuilds to provide additional resiliency, if required.

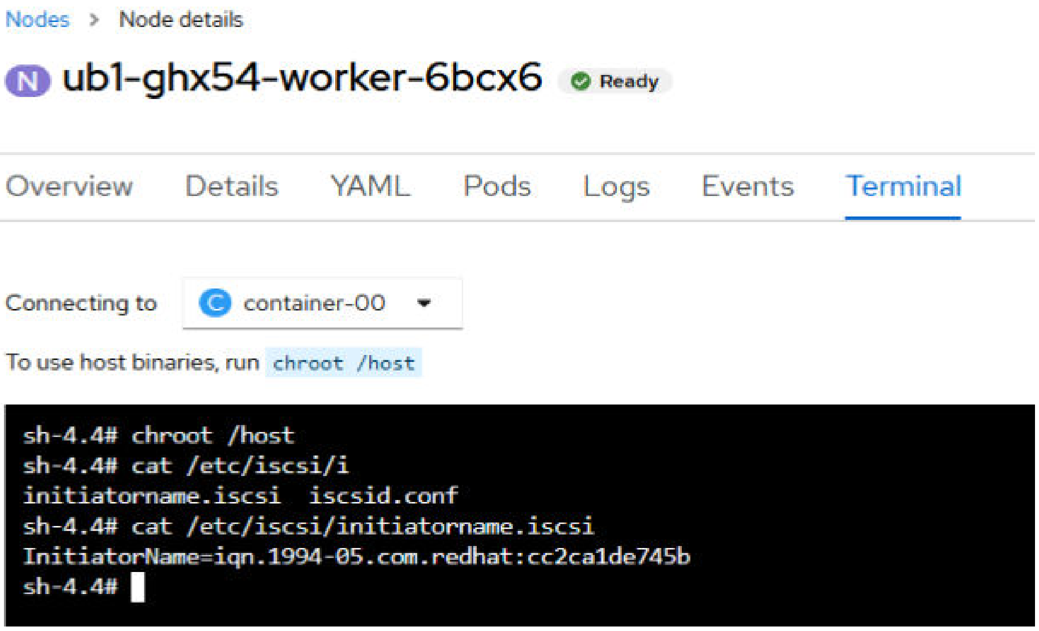


REBUILD CAPACITY INFORMATION

POLICY Fixed	NUMBER OF TOLERABLE DRIVE FAILURES 1
STATUS Sufficient	NUMBER OF REBUILDABLE DRIVES 1

Add Compute Nodes

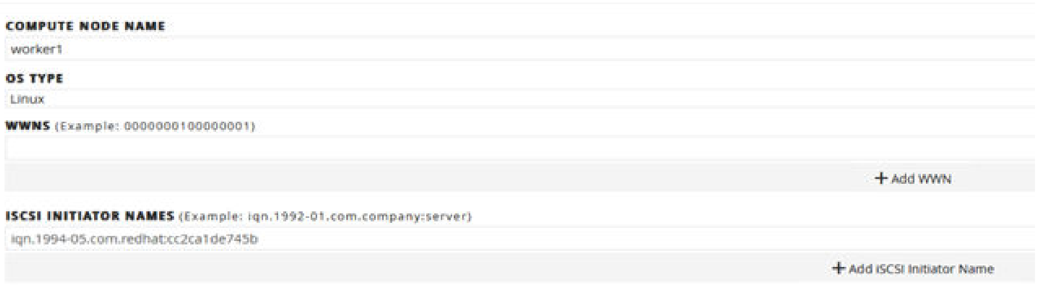
To provision persistent storage on Hitachi Virtual Storage Software block, OCP worker nodes must be added as compute nodes. After the worker nodes are configured in Hitachi Virtual Storage Software block, HSPC enables provisioning of PVs directly onto Hitachi Virtual Storage Software block. The iSCSI Initiator Name for the worker nodes must be provided, and it can be found in the `/etc/iscsi/initiatorname.iscsi` folder.



To use host binaries, run `chroot /host`

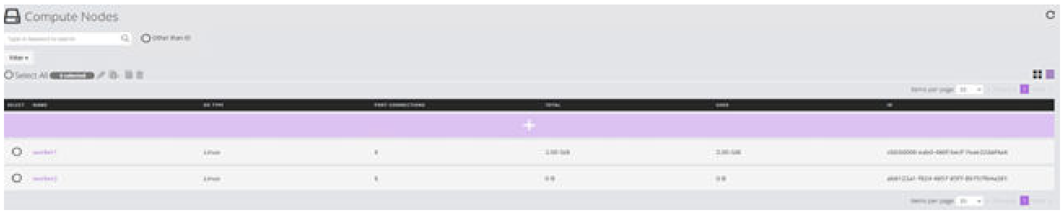
```
sh-4.4# chroot /host
sh-4.4# cat /etc/iscsi/i
initiatorname.iscsi  iscsid.conf
sh-4.4# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:cc2ca1de745b
sh-4.4#
```

Use the iSCSI Initiator Name when adding the OCP worker node as a compute node to Hitachi Virtual Storage Software block.



COMPUTE NODE NAME	worker1
OS TYPE	Linux
WWNS (Example: 0000000100000001)	
ISCSI INITIATOR NAMES (Example: iqn.1992-01.com.company:server)	iqn.1994-05.com.redhat:cc2ca1de745b

After adding all the OCP worker nodes, your compute nodes should look similar to the following.



Solution validation

The Red Hat OpenShift Container Platform cluster was deployed on a hybrid environment, with virtual and bare metal worker nodes using Hitachi Advanced Server and Hitachi Virtual Storage Software block bare metal and hypervisor models.

Hitachi Storage Plug-in for Containers was installed for persistent volume (PV) provisioning. The following container volume operations were performed to validate this solution:

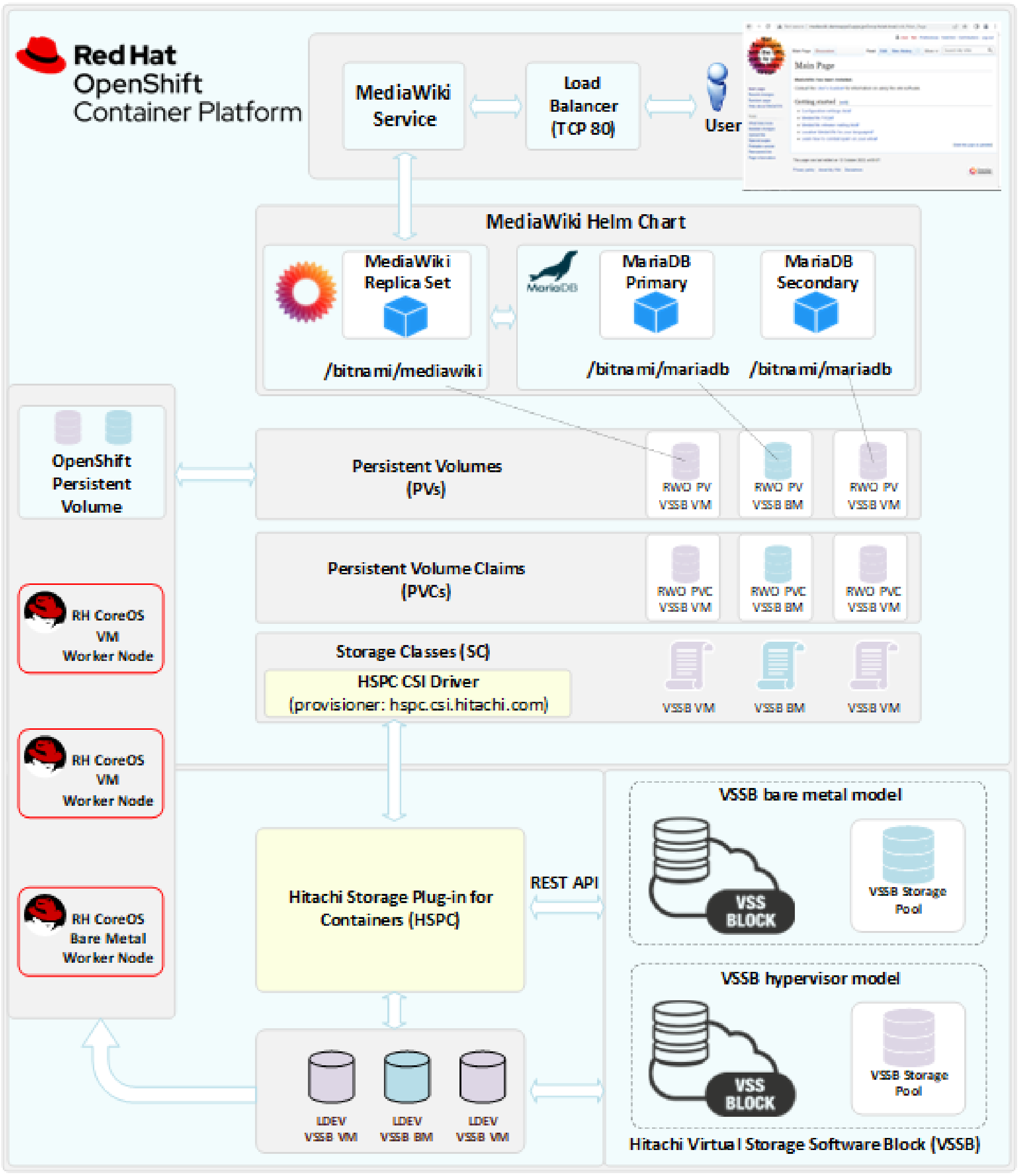
- Deploying a stateful application with high-availability backend database with PVs across two VSSB storage systems, one VSSB bare metal and the other VSSB hypervisor model. For this use case, the MediaWiki application which is an open-source wiki software was used. This application is deployed using the Bitnami Helm chart which includes MariaDB database required for the MediaWiki application. The deployment is done using a high-availability backend database (PVs replicated across two VSSB storage systems).
- Deploy a standard stateful app and perform PV expansion and high-availability tests.

Deploy a stateful application with high-availability backend database across two Hitachi Virtual Storage Software block systems

Use these procedures to do the following:

- Deploy the MediaWiki application with a high-availability backend database with persistent volumes backed by Hitachi Virtual Storage Software block storage system
- Explore the backend storage resources mapping and allocations.

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



Install and configure Helm utilities

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.

Procedure

1. Add the `Bitnami` repository to your Helm configuration by running the following command:

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

2. Search for the MediaWiki Helm chart by running the following command:

```
helm search repo mediawiki
```

3. The following shows example output, showing that the Helm binary is installed correctly and the bitnami repository has been added with a MediaWiki Helm chart available for use.

```
[ocpusr@dminws]$helm search repo mediawiki
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/mediawiki       14.3.4          1.38.2          MediaWiki is the free
and open source wiki soft...
stable/mediawiki        9.1.9           1.34.0          DEPRECATED Extremely
powerful, scalable softwar...
```

Verify StorageClasses

The OCP cluster used for this test is a hybrid, containing virtual worker nodes hosted on VMware ESXi and bare metal worker nodes. All worker nodes (virtual and bare metal) are configured with iSCSI and are connected to Hitachi Virtual Storage Software block storage.

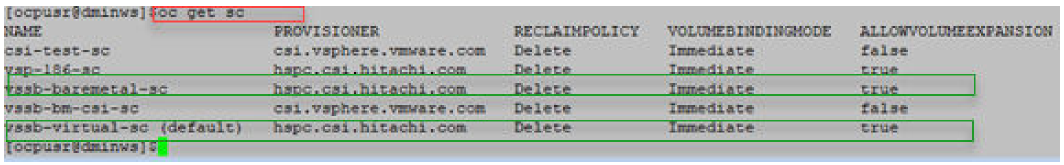
Two Hitachi Virtual Storage Software block storage systems (one bare metal and the other hypervisor-based) were configured and connected to the OCP cluster.

On this OCP cluster StorageClasses are configured with Hitachi HSPC CSI provisioner. To verify the defined StorageClasses enter the Hitachi Virtual Storage Software block `oc get sc` command.

For this example, the following StorageClasses are used:

- The StorageClass `vssb-baremetal-sc` for the primary MariaDB database instance. This storage class uses the VSSB bare metal model.
- The StorageClass `vssb-virtual-sc` for the frontend MediaWiki pods and the secondary MariaDB database instance. This storage class uses the VSSB hypervisor model.

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



```
[ocpusr@dminws]$oc get sc
NAME                     PROVISIONER               RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION
csi-test-sc              csi.vsphere.vmware.com    Delete          Immediate           false
vsp-186-sc               hspc.csi.hitachi.com      Delete          Immediate           true
vssb-baremetal-sc        hspc.csi.hitachi.com      Delete          Immediate           true
vssb-bm-csi-sc           csi.vsphere.vmware.com    Delete          Immediate           false
vssb-virtual-sc (default)  hspc.csi.hitachi.com    Delete          Immediate           true
[ocpusr@dminws]$
```

Customize and deploy MediaWiki Helm charts 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 custom values for a deployment on the command line or in a script.

Procedure

1. Create a namespace (or project) with the following command:

```
oc new-project demoapps3
```

2. The following shows the command and parameters used to deploy MediaWiki Helm chart with replication:

```
helm install -n demoapps3 mediawiki \
--set mediawikiUser=user,mediawikiPassword=Hitachi123 \
--set mariadb.mariadbRootPassword=Hitachi123 \
--set persistence.storageClass=vssb-virtual-sc \
--set persistence.size=8Gi \
--set mariadb.primary.persistence.storageClass=vssb-baremetal-sc \
--set mariadb.primary.persistence.size=8Gi \
--set mariadb.architecture=replication \
--set mariadb.secondary.persistence.storageClass=vssb-virtual-sc \
--set mariadb.secondary.persistence.size=8Gi \
bitnami/mediawiki
```

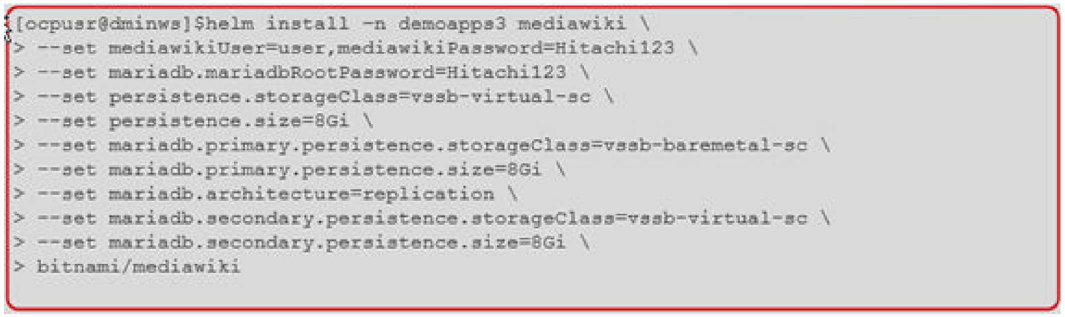
3. Modify these values to match your environment and the StorageClasses that are available in your OCP cluster.

   The values and their corresponding impact to the MediaWiki deployment are as follows:

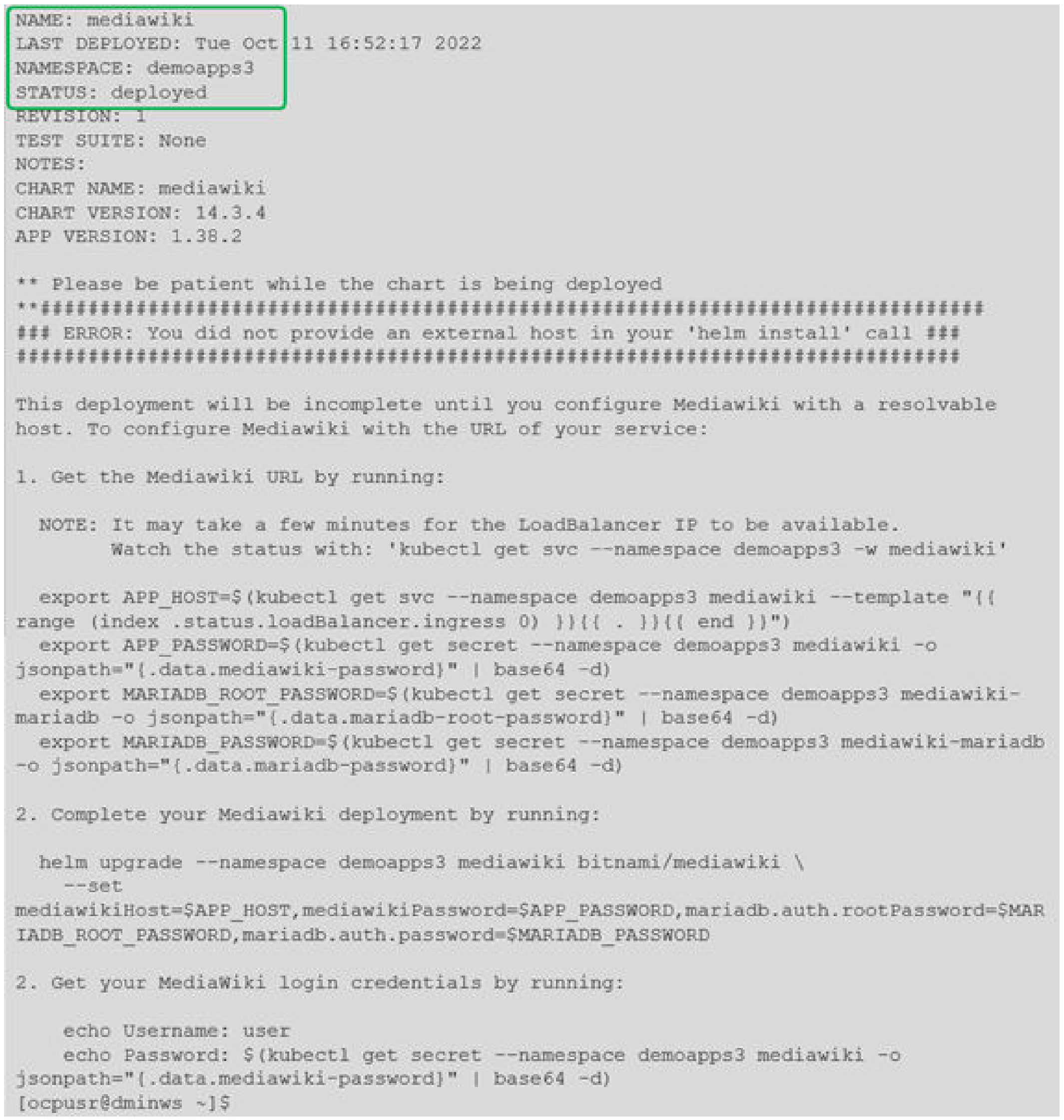
   - mediawikiUser - Sets the admin username for the MediaWiki application.
   - mediawikiPassword - Sets the password for the admin user in the MediaWiki application.
   - mariadb.mariadbRootPassword - Sets the password for the root user in the MariaDB database.
   - persistence.storageClass – Sets the StorageClass to be used for the frontend MediaWiki pod.
   - persistence.size - Sets the size of the persistent volume to be assigned to the frontend MediaWiki pod.
   - 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.architecture - Indicates whether Helm should deploy a single backend database (standalone, single pod) or a high-availability backend database (replication, two pods).
   - mariadb.secondary.persistence.storageClass – Sets the StorageClass to be used for the secondary MariaDB instance.
   - mariadb.secondary.persistence.size - Sets the size of the persistent volume to be assigned to the secondary MariaDB instance.
   - Set the SecurityContext parameters according to your environment.

4. Run the `helm install` command and Helm will begin to deploy your MediaWiki 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.

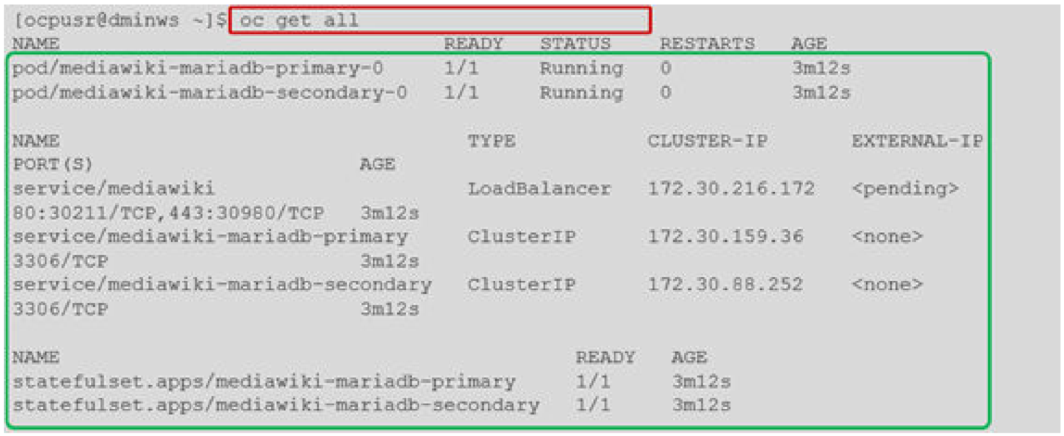


```
[ocpusr@dminws]$helm install -n demoapps3 mediawiki \
> --set mediawikiUser=user,mediawikiPassword=Hitachi123 \
> --set mariadb.mariadbRootPassword=Hitachi123 \
> --set persistence.storageClass=vssb-virtual-sc \
> --set persistence.size=8Gi \
> --set mariadb.primary.persistence.storageClass=vssb-baremetal-sc \
> --set mariadb.primary.persistence.size=8Gi \
> --set mariadb.architecture=replication \
> --set mariadb.secondary.persistence.storageClass=vssb-virtual-sc \
> --set mariadb.secondary.persistence.size=8Gi \
> bitnami/mediawiki
```



```
NAME: mediawiki
LAST DEPLOYED: Tue Oct 11 16:52:17 2022
NAMESPACE: demoapps3
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: mediawiki
CHART VERSION: 14.3.4
APP VERSION: 1.38.2

** Please be patient while the chart is being deployed
**##################################################################################
### ERROR: You did not provide an external host in your 'helm install' call ###
##################################################################################

This deployment will be incomplete until you configure Mediawiki with a resolvable
host. To configure Mediawiki with the URL of your service:

1. Get the Mediawiki URL by running:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace demoapps3 -w mediawiki'

  export APP_HOST=$(kubectl get svc --namespace demoapps3 mediawiki --template "{{
range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")
  export APP_PASSWORD=$(kubectl get secret --namespace demoapps3 mediawiki -o
jsonpath="{.data.mediawiki-password}" | base64 -d)
  export MARIADB_ROOT_PASSWORD=$(kubectl get secret --namespace demoapps3 mediawiki-
mariadb -o jsonpath="{.data.mariadb-root-password}" | base64 -d)
  export MARIADB_PASSWORD=$(kubectl get secret --namespace demoapps3 mediawiki-mariadb
-o jsonpath="{.data.mariadb-password}" | base64 -d)

2. Complete your Mediawiki deployment by running:

  helm upgrade --namespace demoapps3 mediawiki bitnami/mediawiki \
    --set
mediawikiHost=$APP_HOST,mediawikiPassword=$APP_PASSWORD,mariadb.auth.rootPassword=$MAR
IADB_ROOT_PASSWORD,mariadb.auth.password=$MARIADB_PASSWORD

2. Get your MediaWiki login credentials by running:

    echo Username: user
    echo Password: $(kubectl get secret --namespace demoapps3 mediawiki -o
jsonpath="{.data.mediawiki-password}" | base64 -d)
[ocpusr@dminws ~]$
```

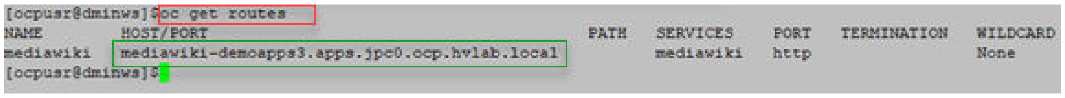
5. Monitor the deployment until the two pods are running.



```
[ocpusr@dminws ~]$ oc get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/mediawiki-mariadb-primary-0      1/1     Running   0          3m12s
pod/mediawiki-mariadb-secondary-0    1/1     Running   0          3m12s

NAME                                  TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)                       AGE
service/mediawiki                     LoadBalancer   172.30.216.172   <pending>
80:30211/TCP,443:30980/TCP    3m12s
service/mediawiki-mariadb-primary     ClusterIP      172.30.159.36    <none>
3306/TCP                      3m12s
service/mediawiki-mariadb-secondary   ClusterIP      172.30.88.252    <none>
3306/TCP                      3m12s

NAME                                         READY   AGE
statefulset.apps/mediawiki-mariadb-primary   1/1     3m12s
statefulset.apps/mediawiki-mariadb-secondary 1/1     3m12s
```

After the primary and secondary pods are in the running state, complete additional steps as indicated in the output of the `helm install` command.

6. If needed, expose the `service/mediawiki` using the `oc expose service/mediawiki` command, and then query the exposed service to get the resolvable host which will be used in the next step.

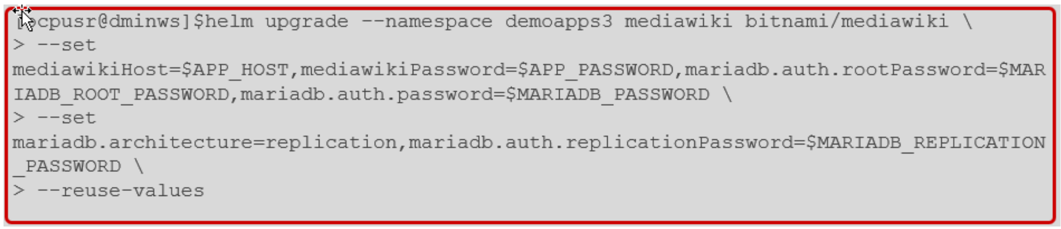


```
[ocpusr@dminws]$oc get routes
NAME        HOST/PORT                                    PATH   SERVICES    PORT   TERMINATION   WILDCARD
mediawiki   mediawiki-demoapps3.apps.jpc0.ocp.hvlab.local        mediawiki   http                 None
[ocpusr@dminws]$
```

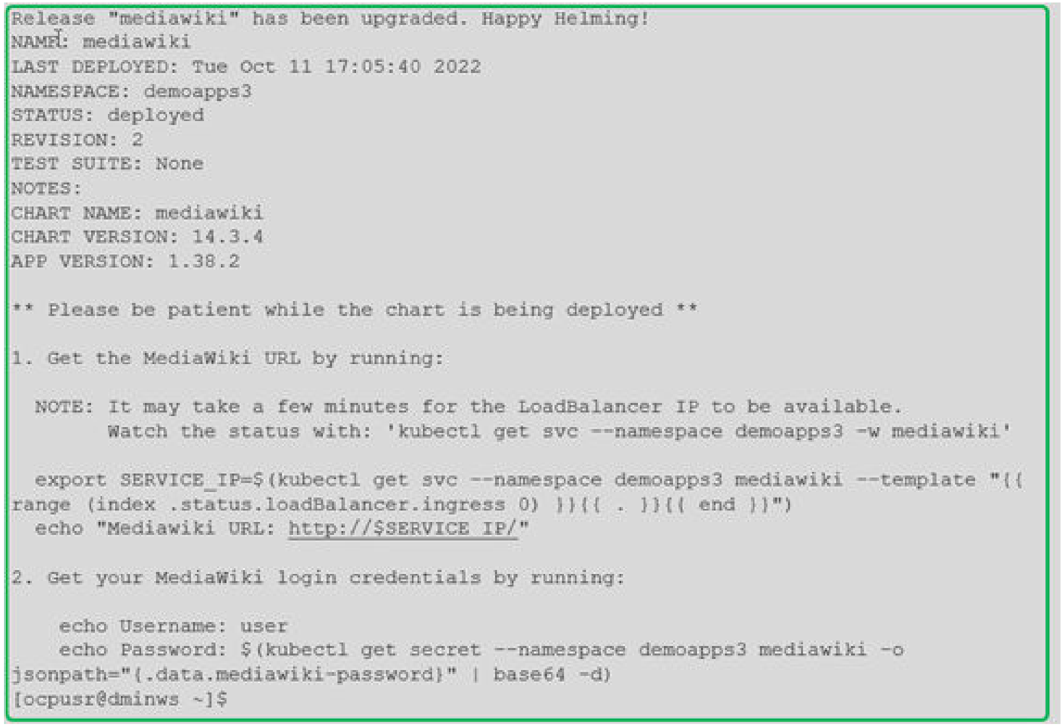
7. Run the following commands to complete the MediaWiki deployment:

```
export APP_HOST=mediawiki-demoapps3.apps.jpc0.ocp.hvlab.local
export APP_PASSWORD=$(oc get secret --namespace demoapps3 mediawiki -o
jsonpath="{.data.mediawiki-password}" | base64 -d)
export MARIADB_ROOT_PASSWORD=$(oc get secret --namespace demoapps3 mediawiki-
mariadb -o jsonpath="{.data.mariadb-root-password}" | base64 -d)
export MARIADB_PASSWORD=$(oc get secret --namespace demoapps3 mediawiki-mariadb -
o jsonpath="{.data.mariadb-password}" | base64 -d)

export MARIADB_REPLICATION_PASSWORD=$(oc get secret --namespace "demoapps3"
mediawiki-mariadb -o jsonpath="{.data.mariadb-replication-password}" | base64 -d)

helm upgrade --namespace demoapps3 mediawiki bitnami/mediawiki \
--set mediawikiHost=$APP_HOST,mediawikiPassword=$APP_PASSWORD,
mariadb.auth.rootPassword=$MARIADB_ROOT_PASSWORD,mariadb.auth.password=
$MARIADB_PASSWORD \
--set mariadb.architecture=replication,mariadb.auth.replicationPassword=
$MARIADB_REPLICATION_PASSWORD \
--reuse-values
```

The following shows output of the helm upgrade command that completes the deployment:



```
[ocpusr@dminws]$helm upgrade --namespace demoapps3 mediawiki bitnami/mediawiki \
> --set
mediawikiHost=$APP_HOST,mediawikiPassword=$APP_PASSWORD,mariadb.auth.rootPassword=$MAR
IADB_ROOT_PASSWORD,mariadb.auth.password=$MARIADB_PASSWORD \
> --set
mariadb.architecture=replication,mariadb.auth.replicationPassword=$MARIADB_REPLICATION
_PASSWORD \
> --reuse-values
```



```
Release "mediawiki" has been upgraded. Happy Helming!
NAME: mediawiki
LAST DEPLOYED: Tue Oct 11 17:05:40 2022
NAMESPACE: demoapps3
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
CHART NAME: mediawiki
CHART VERSION: 14.3.4
APP VERSION: 1.38.2

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

1. Get the MediaWiki URL by running:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace demoapps3 -w mediawiki'

  export SERVICE_IP=$(kubectl get svc --namespace demoapps3 mediawiki --template "{{
range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")
  echo "Mediawiki URL: http://$SERVICE_IP/"

2. Get your MediaWiki login credentials by running:

    echo Username: user
    echo Password: $(kubectl get secret --namespace demoapps3 mediawiki -o
jsonpath="{.data.mediawiki-password}" | base64 -d)
[ocpusr@dminws ~]$
```

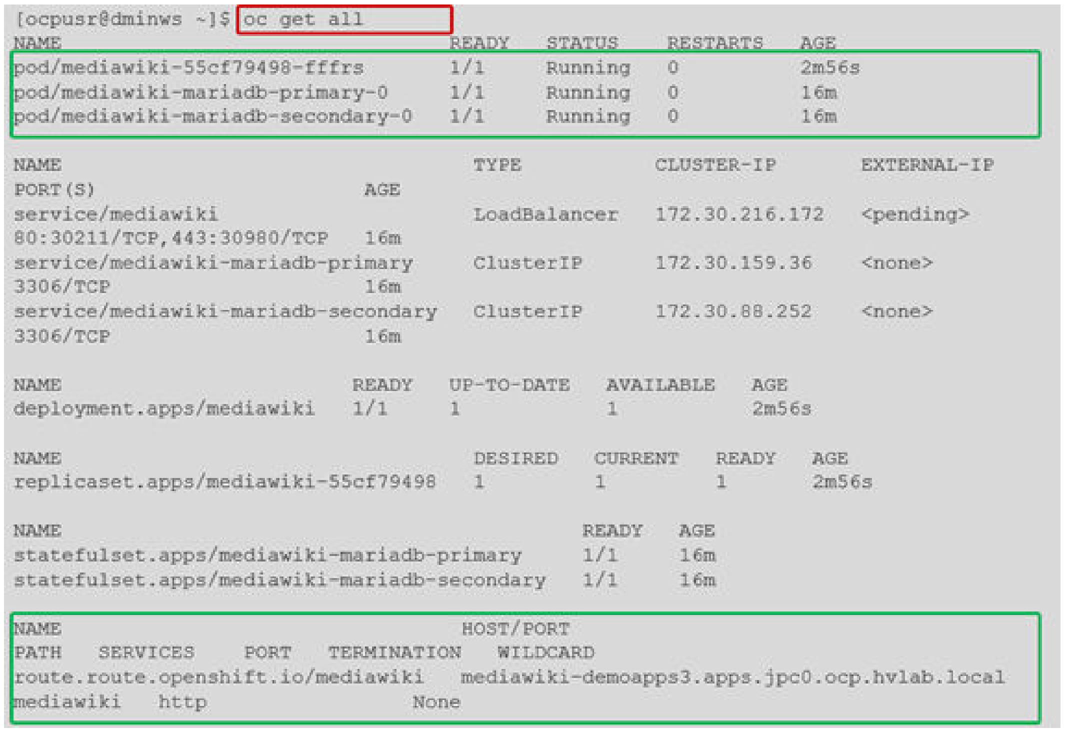
Verify the MediaWiki deployment

You can monitor the MediaWiki deployment by viewing the resources in your *demoapps3* namespace.

Procedure

1. Run the `oc get all -n demoapps3` command to display the readiness of the pods, deployment, statefulsets, and services of the MediaWiki (with MariaDB) deployment.

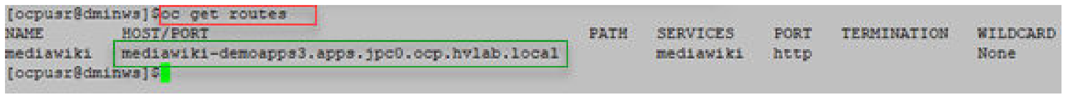
   The following shows an example of the output of this command for a fully running, healthy MediaWiki deployment.



```
[ocpusr@dminws ~]$ oc get all
NAME                                     READY   STATUS     RESTARTS    AGE
pod/mediawiki-55cf79498-fffrs            1/1     Running    0           2m56s
pod/mediawiki-mariadb-primary-0          1/1     Running    0           16m
pod/mediawiki-mariadb-secondary-0        1/1     Running    0           16m

NAME                                       TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)                            AGE
service/mediawiki                          LoadBalancer   172.30.216.172   <pending>
80:30211/TCP,443:30980/TCP         16m
service/mediawiki-mariadb-primary          ClusterIP      172.30.159.36    <none>
3306/TCP                           16m
service/mediawiki-mariadb-secondary        ClusterIP      172.30.88.252    <none>
3306/TCP                           16m

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mediawiki       1/1     1            1           2m56s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/mediawiki-55cf79498        1         1         1       2m56s

NAME                                                READY   AGE
statefulset.apps/mediawiki-mariadb-primary          1/1     16m
statefulset.apps/mediawiki-mariadb-secondary        1/1     16m

NAME                                      HOST/PORT
PATH   SERVICES    PORT   TERMINATION   WILDCARD
route.route.openshift.io/mediawiki    mediawiki-demoapps3.apps.jpc0.ocp.hvlab.local
mediawiki    http                     None
```

2. Log in to MediaWiki using the URL from the exposed service.

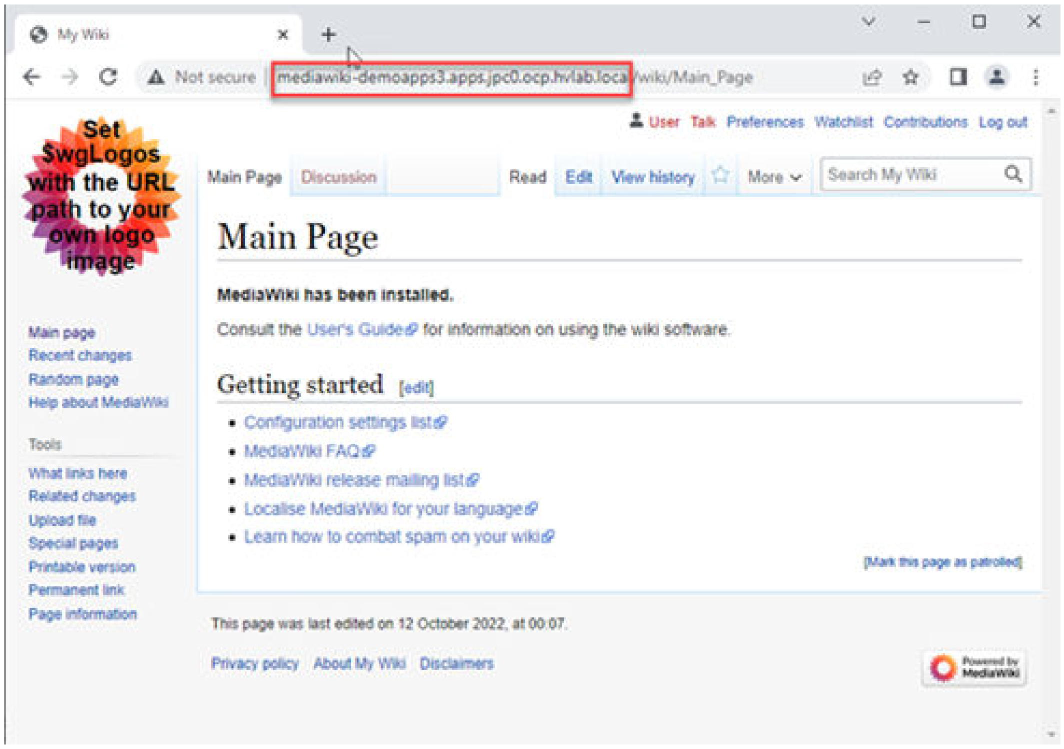
a. To identify the host/port of the exposed MediaWiki service, use the following command:



```
[ocpusr@dminws]$oc get routes
NAME        HOST/PORT                                       PATH   SERVICES    PORT   TERMINATION   WILDCARD
mediawiki   mediawiki-demoapps3.apps.jpc0.ocp.hvlab.local          mediawiki   http                 None
[ocpusr@dminws]$
```

b. Open a browser and enter the address of the MediaWiki service (http), and the MediaWiki interface should display.

c. Log in using the user and password used during the installation.



Main Page

MediaWiki has been installed.

Consult the User's Guide for information on using the wiki software.

Getting started [edit]

- Configuration settings list
- MediaWiki FAQ
- MediaWiki release mailing list
- Localise MediaWiki for your language
- Learn how to combat spam on your wiki

[Mark this page as patrolled]

This page was last edited on 12 October 2022, at 00:07.

Privacy policy About My Wiki Disclaimers

Result

At this point the MediaWiki is ready to use, and you can start creating content which will be stored in persistent storage backed by Hitachi Virtual Storage Software block.

Explore backend storage resource mapping and allocations

When modifying the Helm chart values for the MediaWiki deployment, you provided StorageClasses that map back to different Hitachi Virtual Storage Software block storage systems for persistent volume allocation to the MediaWiki and MariaDB pods.

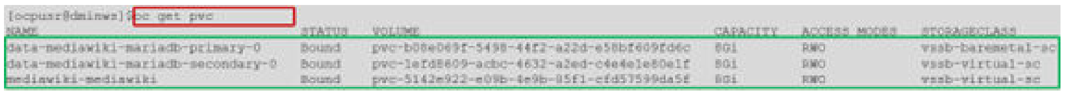
Using the Hitachi Virtual Storage Software block GUI, you can follow the storage paths from the Kubernetes persistent volume layer to the Hitachi Virtual Storage Software block layer.

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.

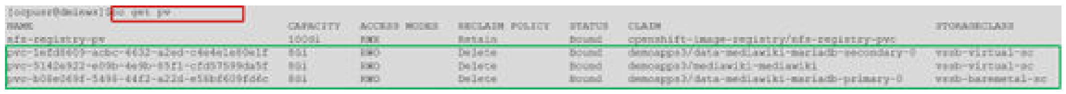
Procedure

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

```
oc get pvc
oc get pv
```



```
[ocpusr@dminws]$oc get pvc
NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-mediawiki-mariadb-primary-0     Bound    pvc-b08e069f-5498-44f2-a22d-e58bf609fd6c   8Gi        RWO            vssb-baremetal-sc
data-mediawiki-mariadb-secondary-0   Bound    pvc-1efd8609-acbc-4632-a2ed-c4e4e1e80e1f   8Gi        RWO            vssb-virtual-sc
mediawiki-mediawiki                  Bound    pvc-5142e922-e09b-4e9b-85f1-cfd57599da5f   8Gi        RWO            vssb-virtual-sc
```



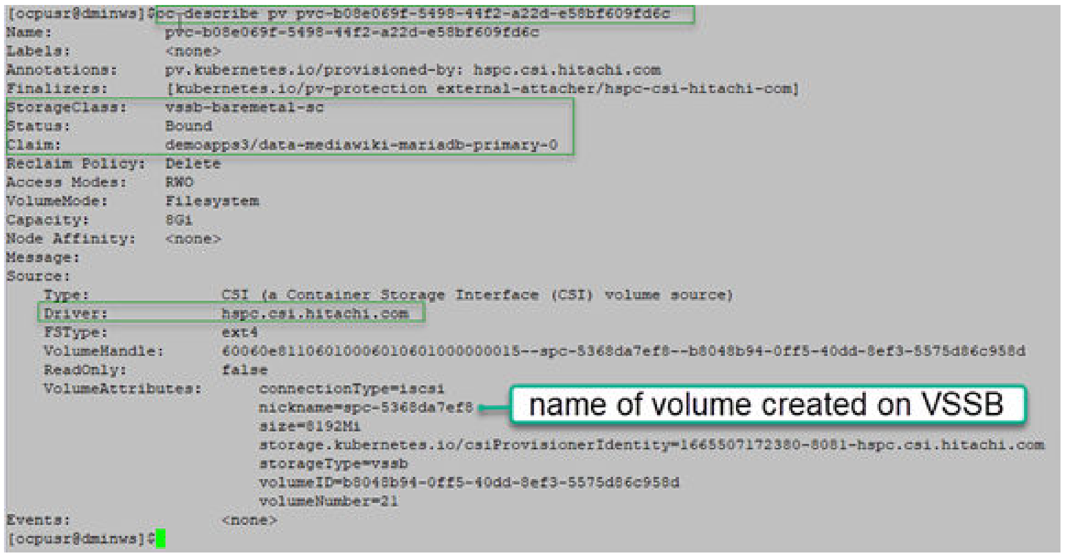
```
[ocpusr@dminws]$oc get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                          STORAGECLASS
nfs-registry-pv                            100Gi      RWX            Retain           Bound    openshift-image-registry/nfs-registry-pvc
pvc-1efd8609-acbc-4632-a2ed-c4e4e1e80e1f   8Gi        RWO            Delete           Bound    demoapps3/data-mediawiki-mariadb-secondary-0   vssb-virtual-sc
pvc-5142e922-e09b-4e9b-85f1-cfd57599da5f   8Gi        RWO            Delete           Bound    demoapps3/mediawiki-mediawiki                  vssb-virtual-sc
pvc-b08e069f-5498-44f2-a22d-e58bf609fd6c   8Gi        RWO            Delete           Bound    demoapps3/data-mediawiki-mariadb-primary-0     vssb-baremetal-sc
```

The output shows that the primary MariaDB Persistent Volume was deployed on a Hitachi Virtual Storage Software block bare metal node using the `vssb-baremetal-sc StorageClass`, and the secondary MariaDB Persistent Volume was deployed on a Hitachi Virtual Storage Software block virtual node (or hypervisor-based) using the `vssb-baremetal-sc StorageClass`. Also, the third Persistent Volume for the MediaWiki frontend pod was deployed on a Hitachi Virtual Storage Software block virtual node.

2. To observe details about the volume within Hitachi VSSB, open a browser and connect to the Hitachi Virtual Storage Software block web interface.
   a. On the Hitachi Virtual Storage Software block web interface, navigate to the **Volumes** tab.

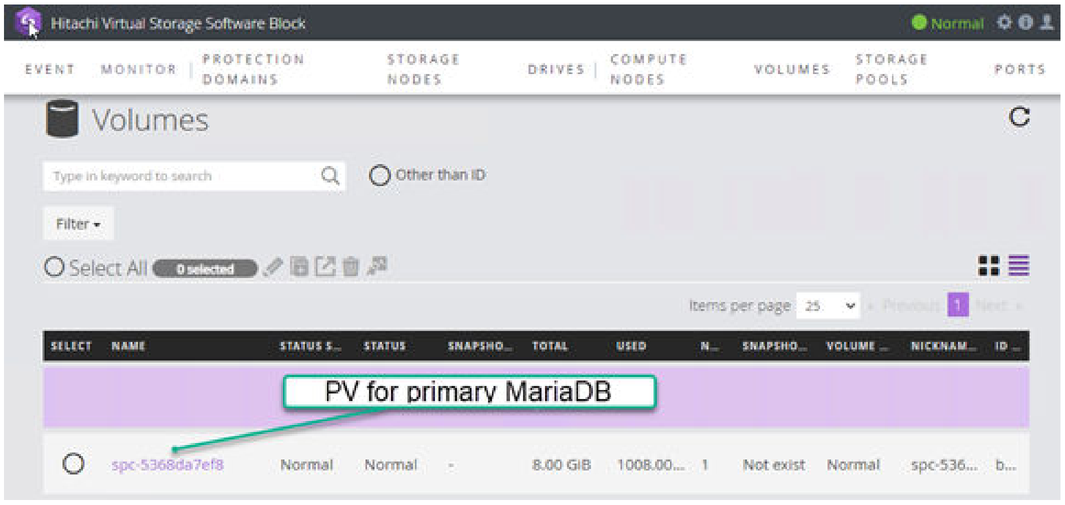
      You will see container volumes provisioned for the MediaWiki application.
   b. Use the `oc describe pv <pv-name>` command to view details about persistent volumes such as StorageClass, Claim, Driver, and details of the backend storage.

      The following shows details for the Persistent Volume created for the primary MariaDB, and under VolumeAttributes you can see the nickname of the volume which is the same of the volume as shown in the Hitachi Virtual Storage Software block (VSSB).



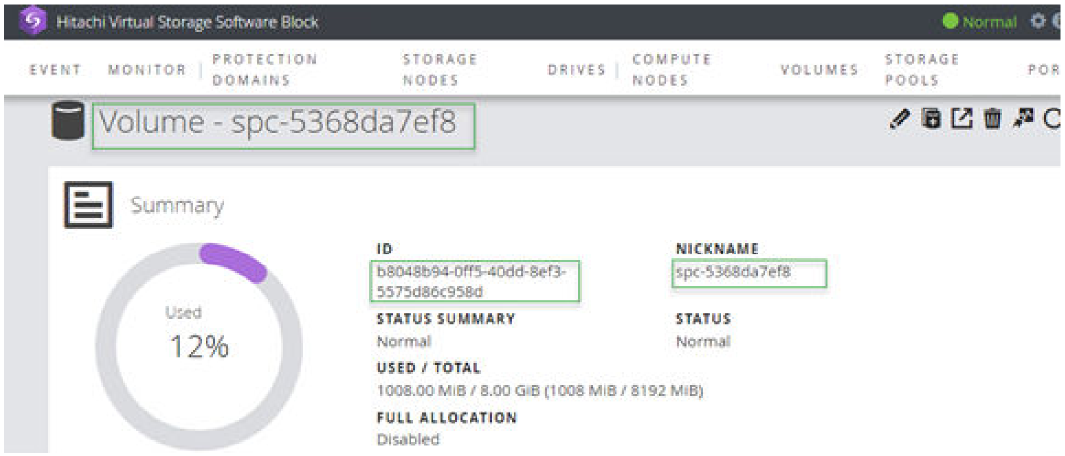
```
[ocpusr@dminws]$oc describe pv pvc-b08e069f-5498-44f2-a22d-e58bf609fd6c
Name:            pvc-b08e069f-5498-44f2-a22d-e58bf609fd6c
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: hspc.csi.hitachi.com
Finalizers:      [kubernetes.io/pv-protection external-attacher/hspc-csi-hitachi-com]
StorageClass:    vssb-baremetal-sc
Status:          Bound
Claim:           demoapps3/data-mediawiki-mariadb-primary-0
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        8Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            hspc.csi.hitachi.com
    FSType:            ext4
    VolumeHandle:      60060e8110601000601060100000015--spc-5368da7ef8--b8048b94-0ff5-40dd-8ef3-5575d86c958d
    ReadOnly:          false
    VolumeAttributes:      connectionType=iscsi
                           nickname=spc-5368da7ef8
                           size=8192Mi
                           storage.kubernetes.io/csiProvisionerIdentity=1665507172380-8081-hspc.csi.hitachi.com
                           storageType=vssb
                           volumeID=b8048b94-0ff5-40dd-8ef3-5575d86c958d
                           volumeNumber=21
Events:                <none>
[ocpusr@dminws]$
```

The following shows the LDEV/Volume created on the backend VSSB bare metal storage system, and it corresponds to the PV shown previously with the name *spc-5368da7ef8*.



SELECT	NAME	STATUS S...	STATUS	SNAPSHO...	TOTAL	USED	N...	SNAPSHO...	VOLUME ...	NICKNAM...	ID ...
○	spc-5368da7ef8	Normal	Normal	-	8.00 GiB	1008.00...	1	Not exist	Normal	spc-536...	b...

3. Click on the volume name to view additional details such as the Volume ID, volume nickname, and capacity details.



ID
b8048b94-0ff5-40dd-8ef3-5575d86c958d

NICKNAME
spc-5368da7ef8

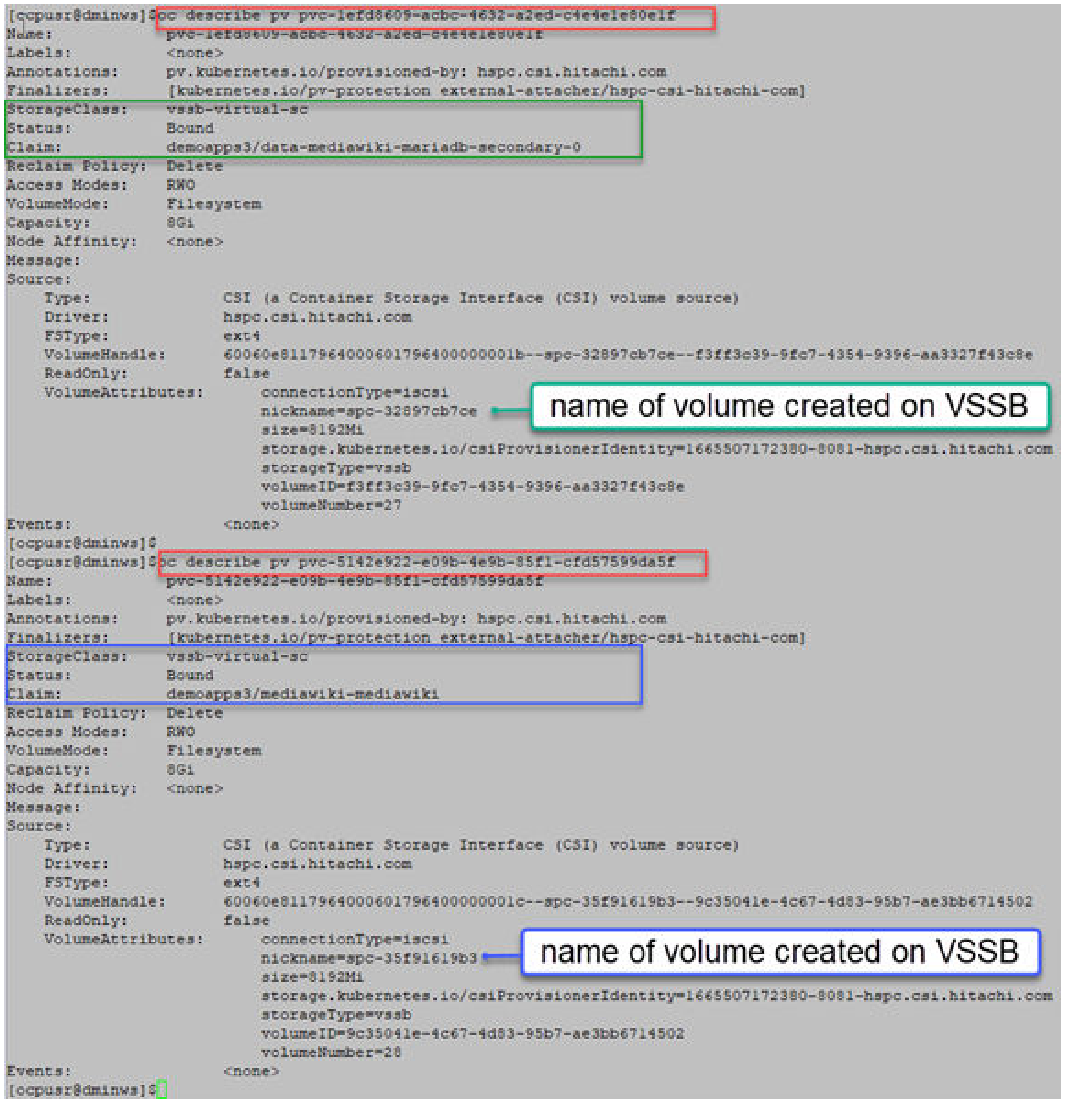
STATUS SUMMARY
Normal

STATUS
Normal

USED / TOTAL
1008.00 MiB / 8.00 GiB (1008 MiB / 8192 MiB)

FULL ALLOCATION
Disabled

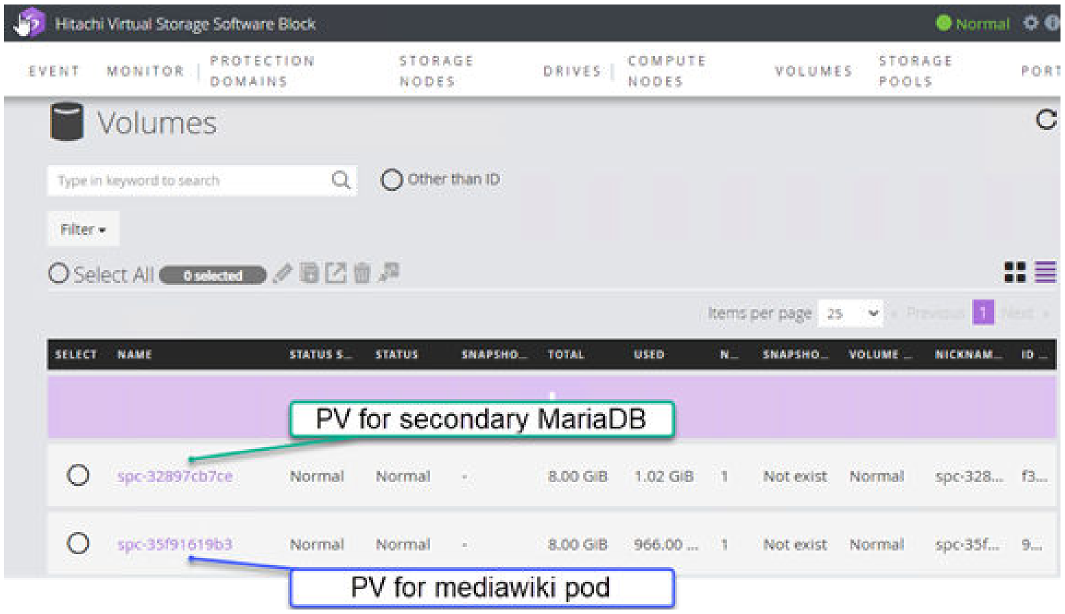
In the same way, you can correlate the other two PVCs/PVs created for the secondary MariaDB and the MediaWiki pod that were created on a different VSSB (hypervisor-based model). Under VolumeAttributes we can see the nickname of the volume which is the same name of the volume shown in the VSSB.



```
[ocpusr@dminws]$oc describe pv pvc-1efd8609-acbc-4632-a2ed-c4e4e1e80e1f
Name:            pvc-1efd8609-acbc-4632-a2ed-c4e4e1e80e1f
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: hspc.csi.hitachi.com
Finalizers:      [kubernetes.io/pv-protection external-attacher/hspc-csi-hitachi-com]
StorageClass:    vssb-virtual-sc
Status:          Bound
Claim:           demoapps3/data-mediawiki-mariadb-secondary-0
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        8Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            hspc.csi.hitachi.com
    FSType:            ext4
    VolumeHandle:      60060e8117964000601796400000001b--spc-32897cb7ce--f3ff3c39-9fc7-4354-9396-aa3327f43c8e
    ReadOnly:          false
    VolumeAttributes:      connectionType=iscsi
                           nickname=spc-32897cb7ce
                           size=8192Mi
                           storage.kubernetes.io/csiProvisionerIdentity=1665507172380-8081-hspc.csi.hitachi.com
                           storageType=vssb
                           volumeID=f3ff3c39-9fc7-4354-9396-aa3327f43c8e
                           volumeNumber=27
Events:                <none>
[ocpusr@dminws]$
[ocpusr@dminws]$oc describe pv pvc-5142e922-e09b-4e9b-85f1-cfd57599da5f
Name:            pvc-5142e922-e09b-4e9b-85f1-cfd57599da5f
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: hspc.csi.hitachi.com
Finalizers:      [kubernetes.io/pv-protection external-attacher/hspc-csi-hitachi-com]
StorageClass:    vssb-virtual-sc
Status:          Bound
Claim:           demoapps3/mediawiki-mediawiki
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        8Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            hspc.csi.hitachi.com
    FSType:            ext4
    VolumeHandle:      60060e8117964000601796400000001c--spc-35f91619b3--9c35041e-4c67-4d83-95b7-ae3bb6714502
    ReadOnly:          false
    VolumeAttributes:      connectionType=iscsi
                           nickname=spc-35f91619b3
                           size=8192Mi
                           storage.kubernetes.io/csiProvisionerIdentity=1665507172380-8081-hspc.csi.hitachi.com
                           storageType=vssb
                           volumeID=9c35041e-4c67-4d83-95b7-ae3bb6714502
                           volumeNumber=28
Events:                <none>
[ocpusr@dminws]$
```

name of volume created on VSSB
name of volume created on VSSB

The following shows the LDEV/Volume created on the backend VSSB (hypervisor-based model) storage system and it corresponds to the secondary MariaDB PV with the name *spc-32897cb7ce* and the MediaWiki PV with name *spc-35f91619b3*.



SELECT	NAME	STATUS S...	STATUS	SNAPSHO...	TOTAL	USED	N...	SNAPSHO...	VOLUME ...	NICKNAM...	ID ...
○	spc-32897cb7ce	Normal	Normal	-	8.00 GiB	1.02 GiB	1	Not exist	Normal	spc-328...	f3...
○	spc-35f91619b3	Normal	Normal	-	8.00 GiB	966.00 ...	1	Not exist	Normal	spc-35f...	9...

Deploy standard stateful application and perform volume expansion tests

Use these procedures to do the following:

- Create persistent volumes
- Expand persistent volumes

Create persistent volumes

After following the instructions to deploy Red Hat OCP and installing and configuring Hitachi Storage Plug-in for Containers, you are now ready to provision persistent volumes.

Use the following YAML file to create a persistent volume claim:

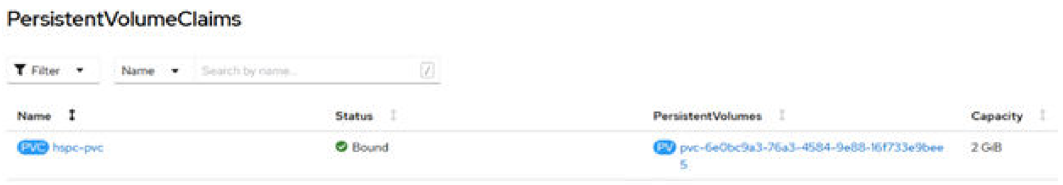
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: hspc-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  storageClassName: hspc-vssb-sc
```

Note: If the `volumeMode` parameter is skipped, by default `Filesystem` is selected. Block mode is also available for applications that require raw block storage.

Use the following YAML file to create a pod and mount the persistent storage provisioned in the previous step:

```
apiVersion: v1
kind: Pod
metadata:
  name: hspc-pod
spec:
  containers:
    - name: my-busybox
      image: busybox
      volumeMounts:
      - mountPath: "/data"
        name: hspc-volume
      command: ["sleep", "1000000"]
      imagePullPolicy: IfNotPresent
  volumes:
    - name: hspc-volume
      persistentVolumeClaim:
        claimName: hspc-pvc
```

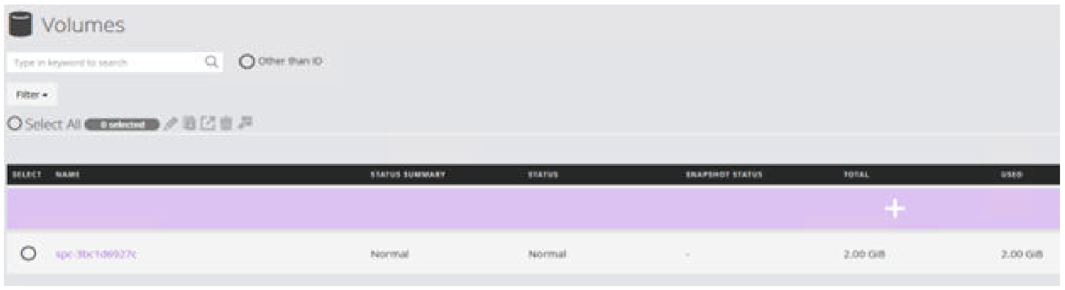
After the pod is up and running, it will mount the associated PVC and start consuming it. Verify the PVC creation in OCP and Hitachi Virtual Storage Software block.



PersistentVolumeClaims

Name	Status	PersistentVolumes	Capacity
PVC hspc-pvc	Bound	PV pvc-6e0bc9a3-76a3-4584-9e88-16f733e9bee5	2 GiB

Using the Hitachi Virtual Storage Software block GUI, you can find the volume that was created.

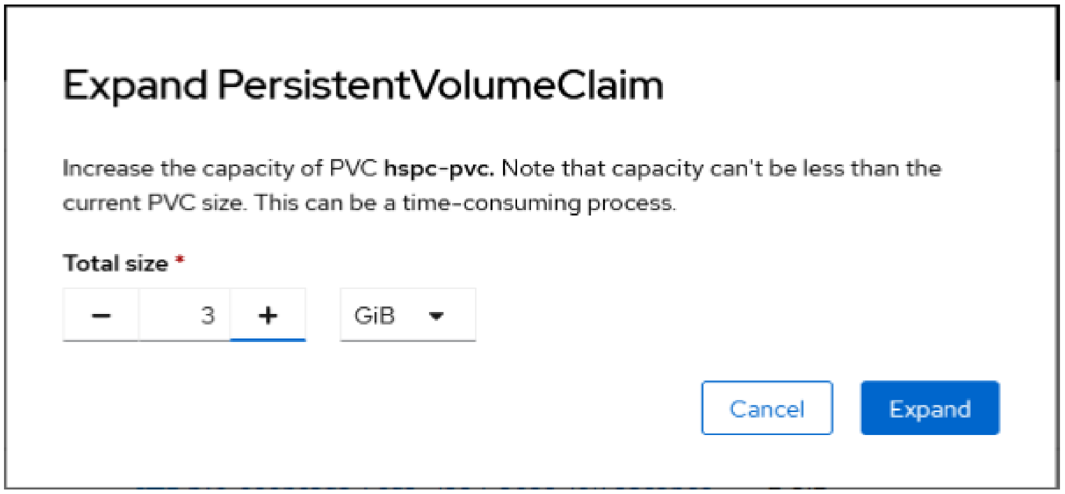


Volumes

SELECT	NAME	STATUS SUMMARY	STATUS	SNAPSHOT STATUS	TOTAL	USED
○	spc-3bc1d6927c	Normal	Normal	-	2.00 GiB	2.00 GiB

Expand persistent volumes

Expansion of the PVC by adjusting the capacity using the OCP console is quick, and it can be verified in the Hitachi Virtual Storage Software block GUI.



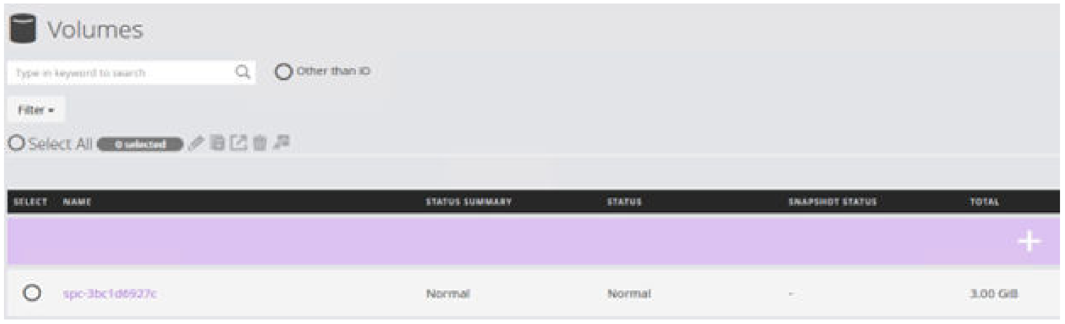
Expand PersistentVolumeClaim

Increase the capacity of PVC **hspc-pvc.** Note that capacity can't be less than the current PVC size. This can be a time-consuming process.

Total size *

3 GiB

Verify the expanded PVC in the Hitachi Virtual Storage Software block GUI.



Volumes

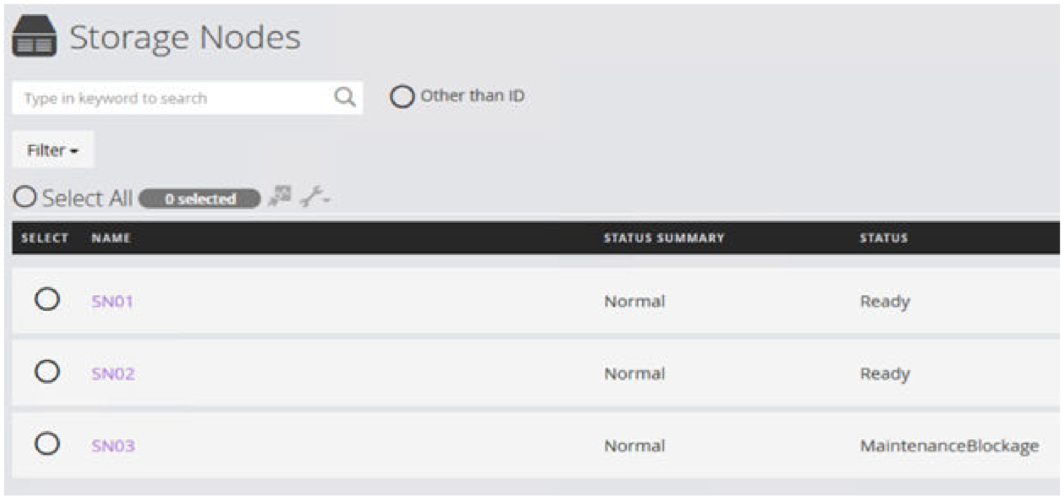
SELECT	NAME	STATUS SUMMARY	STATUS	SNAPSHOT STATUS	TOTAL
○	spc-3bc1d6927c	Normal	Normal	-	3.00 GiB

High-availability tests

This section details the high availability tests that were used for validation.

Node Failure

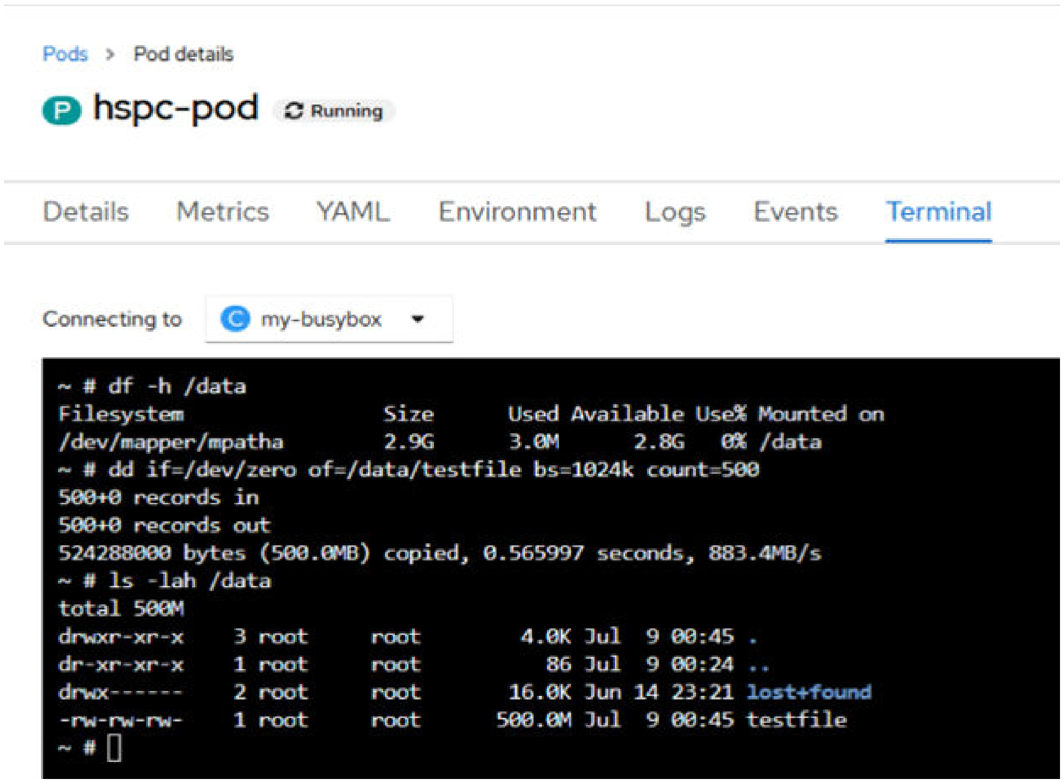
If a storage node is down because of failure or maintenance, regular operations continue because of the highly available design of Hitachi Virtual Storage Software block.



Storage Nodes

SELECT	NAME	STATUS SUMMARY	STATUS
○	SN01	Normal	Ready
○	SN02	Normal	Ready
○	SN03	Normal	MaintenanceBlockage

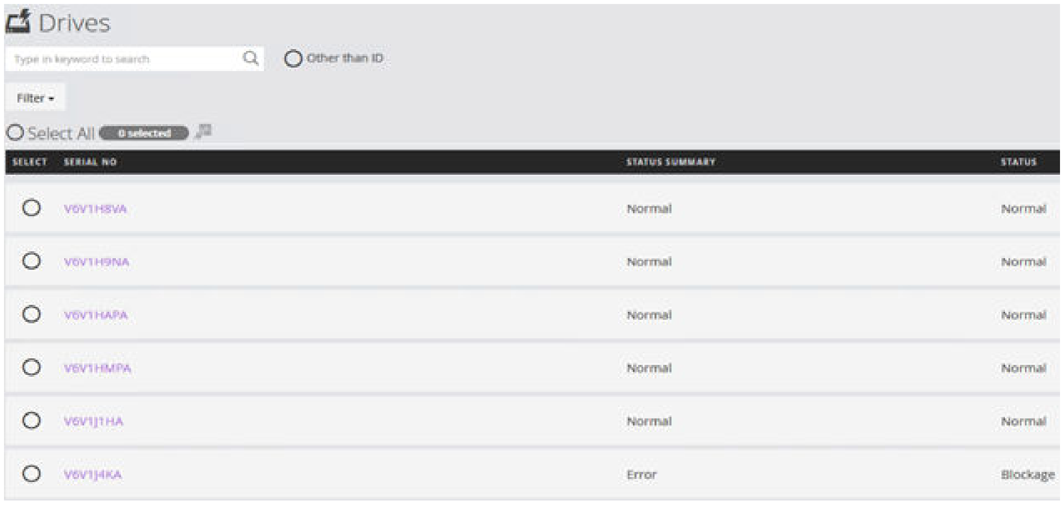
Data can still be written to volumes mounted in pods even if a node is down.



```
~ # df -h /data
Filesystem                Size      Used Available Use% Mounted on
/dev/mapper/mpatha        2.9G      3.0M      2.8G   0% /data
~ # dd if=/dev/zero of=/data/testfile bs=1024k count=500
500+0 records in
500+0 records out
524288000 bytes (500.0MB) copied, 0.565997 seconds, 883.4MB/s
~ # ls -lah /data
total 500M
drwxr-xr-x    3 root     root        4.0K Jul  9 00:45 .
dr-xr-xr-x    1 root     root          86 Jul  9 00:24 ..
drwx------    2 root     root       16.0K Jun 14 23:21 lost+found
-rw-rw-rw-    1 root     root      500.0M Jul  9 00:45 testfile
~ #
```

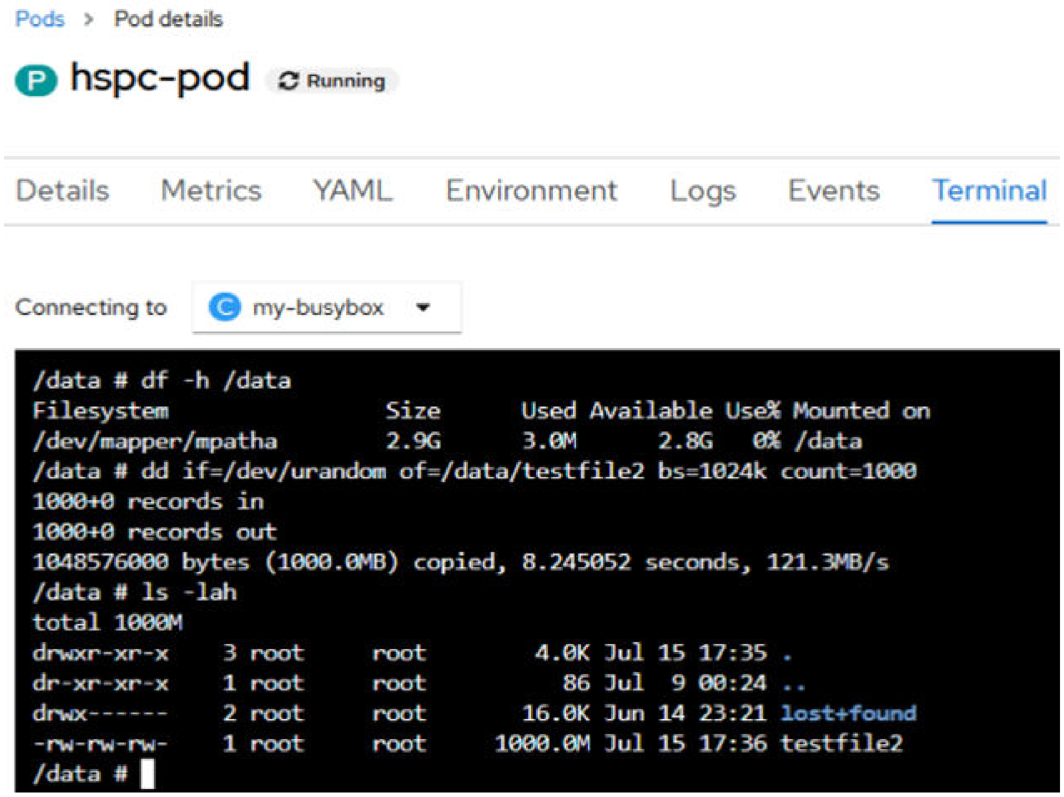
Disk Failure

If a disk fails, regular operations continue because data is redundant using Hitachi Polyphase Erasure Coding or mirrored mode protection policies.



SELECT	SERIAL NO	STATUS SUMMARY	STATUS
○	V6V1H8VA	Normal	Normal
○	V6V1H9NA	Normal	Normal
○	V6V1HAPA	Normal	Normal
○	V6V1HMPA	Normal	Normal
○	V6V1J1HA	Normal	Normal
○	V6V1J4KA	Error	Blockage

Data can still be written to volumes mounted in pods even if a disk is down.



```
/data # df -h /data
Filesystem                Size      Used Available Use% Mounted on
/dev/mapper/mpatha        2.9G      3.0M      2.8G   0% /data
/data # dd if=/dev/urandom of=/data/testfile2 bs=1024k count=1000
1000+0 records in
1000+0 records out
1048576000 bytes (1000.0MB) copied, 8.245052 seconds, 121.3MB/s
/data # ls -lah
total 1000M
drwxr-xr-x    3 root     root        4.0K Jul 15 17:35 .
dr-xr-xr-x    1 root     root          86 Jul  9 00:24 ..
drwx------    2 root     root       16.0K Jun 14 23:21 lost+found
-rw-rw-rw-    1 root     root     1000.0M Jul 15 17:36 testfile2
/data #
```

Conclusion

Hitachi Advanced Server, Hitachi Virtual Storage Software block, Hitachi Storage Plug-in for Containers, and Red Hat OpenShift Container Platform combine to create a powerful and flexible Kubernetes ecosystem.

This reference architecture highlights recommended components for operating a Red Hat OpenShift solution in a Hitachi infrastructure environment (Hitachi Advanced Server and/or Hitachi Virtual Storage Software block) while taking advantage of various Hitachi data storage integrations to achieve a highly resilient and protected platform to deliver Kubernetes clusters and containers at scale.