INFINIDAT

WHITE PAPER

Configuring Red Hat OpenShift Virtualization with Infinidat InfiniBox® Storage

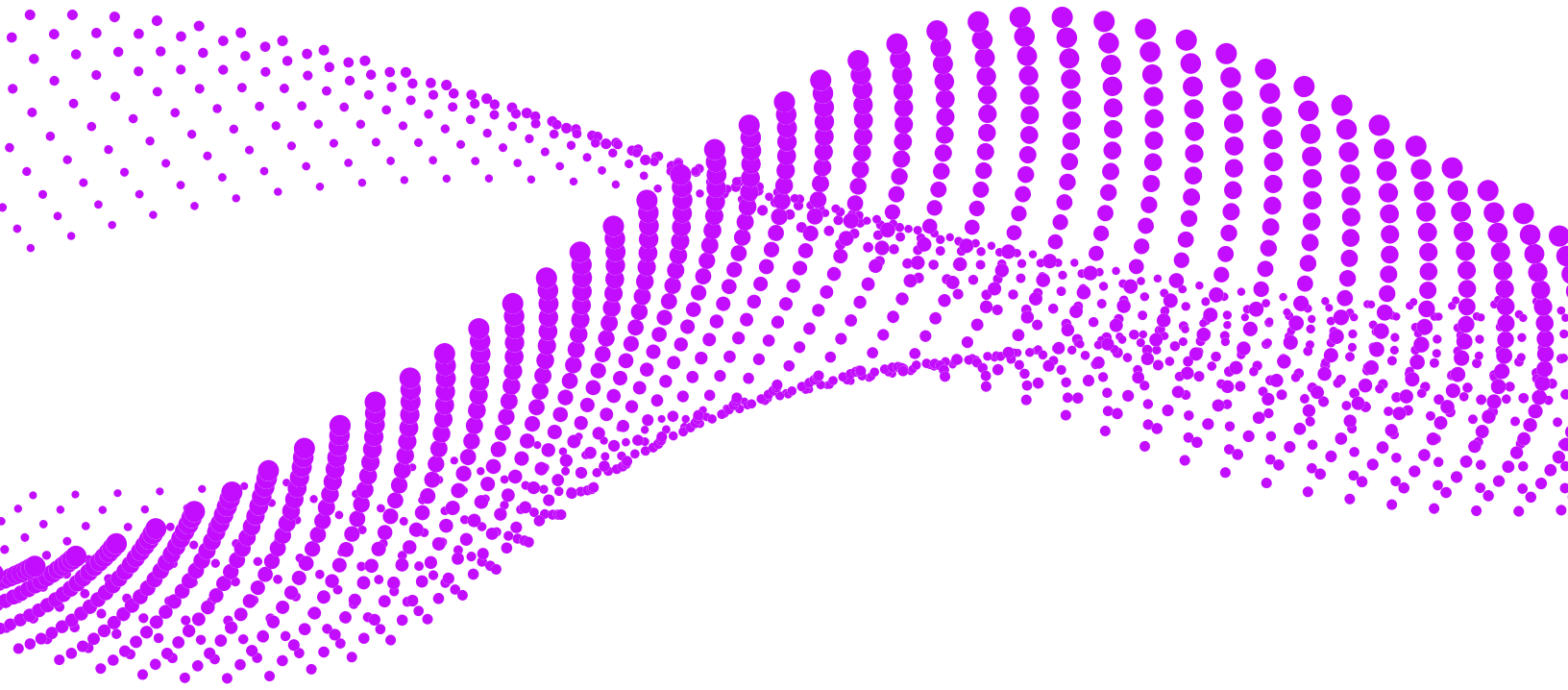


Table of Contents

Abstract

As organizations adopt a unified platform to orchestrate container and virtual machine based workloads, the need for robust and high-performance storage solutions becomes more critical. Red Hat OpenShift Virtualization offers the ability to run both containerized and traditional virtual machine (VM) workloads on a unified platform. When paired with Infinidat storage, known for its Enterprise class scalability, reliability, and performance, organizations and enterprises can unlock significant advantages in managing modern, hybrid workloads and Virtual Machines.

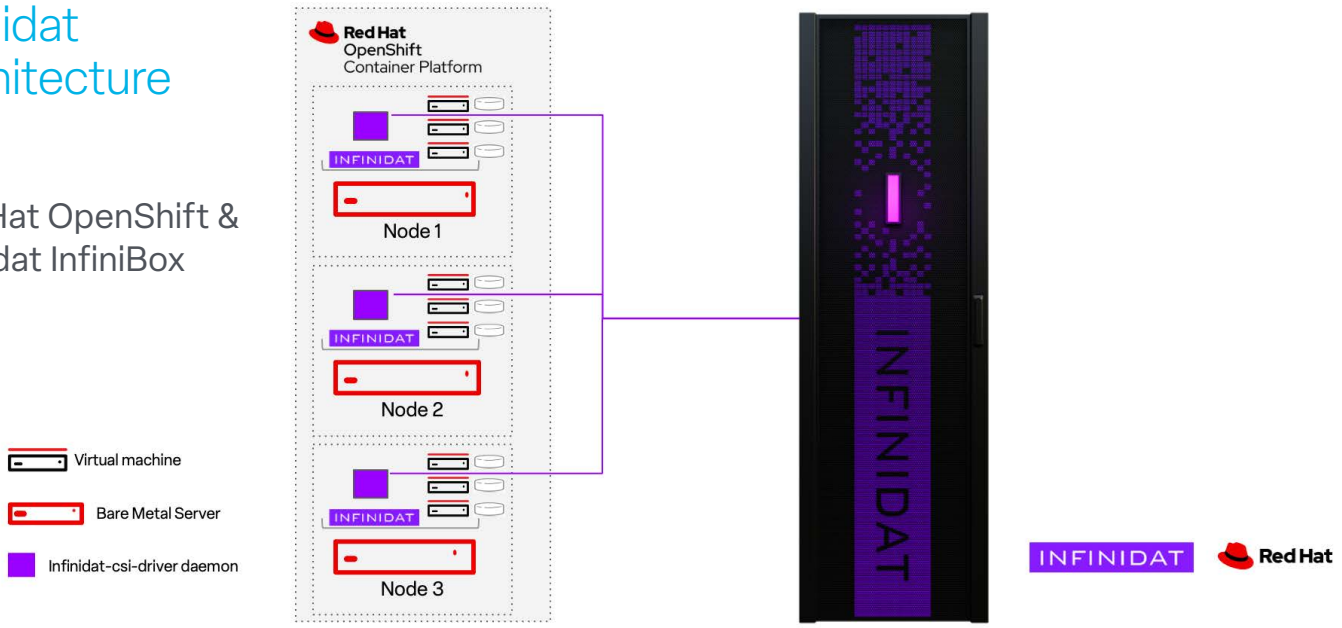
The combination of OpenShift Virtualization and Infinidat storage provides a seamless solution to scale applications and workloads without compromising performance. Infinidat's enterprise-grade storage architecture is uniquely designed to deliver low-latency, high-throughput performance capabilities that align with the demanding requirements of containerized environments, ensuring that OpenShift Virtualization can handle resource-intensive applications efficiently. The scalability of Infinidat storage and its agile performance elasticity for consolidated workloads complements the dynamic nature of containerized and Virtual Machines workloads, enabling businesses to painlessly scale their storage needs in tandem with the growth of their applications.

With this integrated solution, organizations benefit from simplified storage management, optimized performance for both virtualized and containerized workloads, and a future-proof architecture that can scale as needed. As a result, **Red Hat OpenShift Virtualization** combined with **Infinidat** storage delivers the agility and performance required to meet the evolving demands of modern, diverse enterprise workloads, while maintaining high availability and cost-efficiencies to drive down OpEx.

Infinidat enterprise storage solutions enable scaling virtual machine (VM) workloads on Red Hat OpenShift Virtualization by leveraging the certified InfiniBox Container Storage Interface (CSI) driver. This paper demonstrates the viability of deploying Infinidat Storage for an OpenShift Cluster and delivering storage for containers and mainly Virtual Machines using OpenShift Virtualization.

Infinidat Architecture

Red Hat OpenShift & Infinidat InfiniBox



INFINIDAT

Infinidat InfiniBox Container Storage Interface (CSI) Driver is a CNCF-compliant Kubernetes integration for InfiniBox storage systems, offering advanced enterprise functionality for petabyte-scale Kubernetes deployments including Red Hat OpenShift.

InfiniBox G4 and InfiniBox SSA G4 all-flash are industry leading Enterprise Storage systems that enable your organization with Enterprise class performance, capacity, and features including 100% availability, and industry leading cyber resilience and protection. Infinidat patented Neural Cache performance acceleration delivers exceptional consolidated workload performance. InfiniBox G4, the latest generation of InfiniBox storage, announced this year, delivers up to a 2x increase in overall performance. With 31% more cores per controller node, power can be reduced by 20% per core, on average.* InfiniBox and InfiniBox SSA All-Flash are ideal for diverse workload consolidation while lowering both OpEx and carbon footprints.

InfiniBox SSA G4 now includes a high-end enterprise family of solutions starting at only 14RU and lower capacities (155TB) that are ideal for all-flash use cases in smaller, remote/edge, or distributed deployments. They may be installed in the customer's own industry-standard racks.

RED HAT OPENSHIFT VIRTUALIZATION

Red Hat® OpenShift® Virtualization, an included feature of Red Hat OpenShift, provides a modern platform for organizations to run and deploy their new and existing virtual machine (VM) workloads. The solution allows for easy migration and management of traditional virtual machines onto a trusted, consistent, and comprehensive hybrid cloud application platform.

Red Hat OpenShift Virtualization adds new objects into your OpenShift Container Platform cluster by using Kubernetes custom resources to enable virtualization tasks. These tasks include:

- Creating and managing Linux and Windows virtual machines (VMs)
- Running pod and VM workloads alongside each other in a cluster
- Connecting to virtual machines through a variety of consoles and CLI tools
- Importing and cloning existing virtual machines
- Managing network interface controllers and storage disks attached to virtual machines
- Live migrating virtual machines between nodes

An enhanced web console provides a graphical portal to manage these virtualized resources alongside the OpenShift Container Platform cluster containers and infrastructure.

Red Hat OpenShift Virtualization removes workload barriers by unifying Virtual Machine (VM) deployment and management alongside containerized applications in a cloud-native manner. Users seeking virtualization alternatives demand reliable and scalable performance for VM deployments, which often include multiple PersistentVolumeClaim (PVC) attachments, including disks that are snapshot clones of a persistent image source.

Previously shared in this solution brief, InfiniBox Enterprise Storage for Red Hat OpenShift Virtualization, the Infinidat InfiniBox solution has been successfully validated to work with Red Hat OpenShift Virtualization, including scaling out multiple VM and PVC deployments in parallel.

HOW STORAGE IS PRESENTED TO VIRTUAL MACHINES IN OPENSHIFT/K8S INFINIBOX

The Kubernetes CSI model dynamically creates a 1-to-1 mapping for each PVC request a pod and/or VM makes, which includes provisioning an underlying PersistentVolume (PV) and storage volume LUN (logical unit number).

As a result direct 1:1:1 mappings of PVC:PV:LUN are dynamically created and deleted following the lifecycle of PersistentVolumeClaims. The Infinidat CSI utilizes recommended best practices for efficient device discovery (including searching by-id to allow for discovery of >255 LUNs per node) and correct device teardown order ensuring proper multipath device cleanup.

Prerequisites Setup Steps

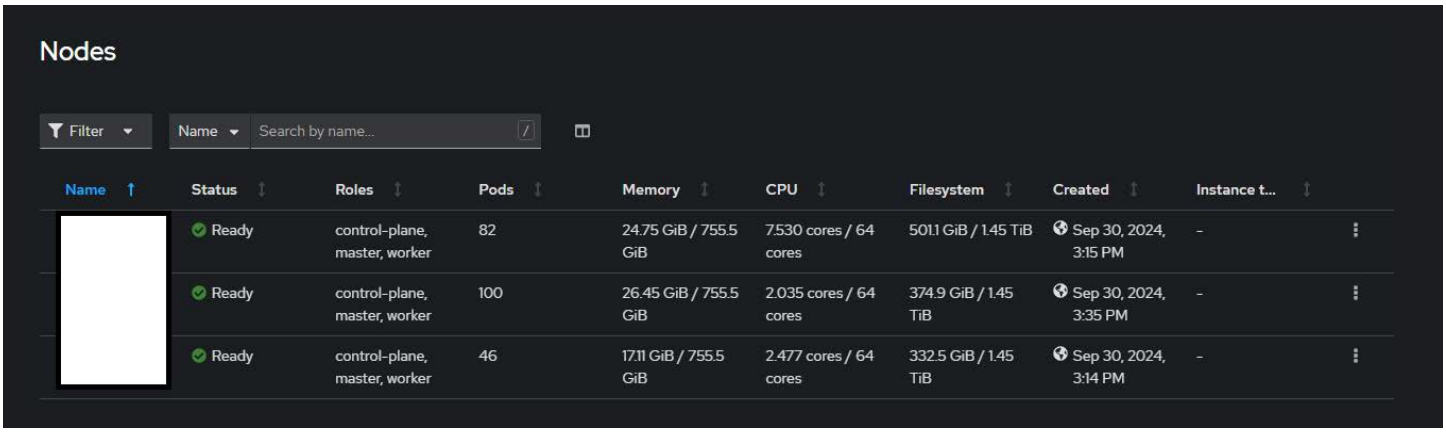
- OpenShift 4.16 or later installed.
- Node-level configuration of multipath and drivers.

Configuration

NODE LEVEL

The testing environment consisted of the following:

- InfiniBox
- FC Protocol
- 3 Node Red Hat OpenShift 4.16 compact cluster, where each node has both master and worker roles.



This small environment was used for internal lab testing, but is not recommended for production.

ENABLE MULTIPATH

Multipath should be enabled and configured on OpenShift nodes in the cluster using the Infinidat-recommended options. The following example provides a `multipath-machineconfig-master.bu` Butane config file, which should be applied as a MachineConfig object:

***Note:** Applying a MachineConfig will cause nodes to reboot

****Note:** OpenShift manages its own Operating System configuration, and it treats its server as an appliance. Any configuration applied to the host must be done through MachineConfigs, in order to deliver a fully working system, the Operating System will be versioned.

*****Note:** Consider blacklisting any devices that you do not want included in the multipath configuration, as this helps prevent unintended devices—such as local disks, USB drives, or unsupported storage—from being mistakenly managed by multipath. This ensures system stability and avoids potential conflicts or performance issues.

Multipath is leveraged when connecting to storage over FC or iSCSI.

```
Unset
variant: openshift
version: 4.16.0
metadata:
  name: 99-master-multipath-fc-udev-infinibox
  labels:
    machineconfiguration.openshift.io/role: worker
openshift:
  kernel_arguments:
    - elevator=none
storage:
  files:
    - path: /etc/modprobe.d/infinidat.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          options lpfc lpfc_hba_queue_depth=4096 lpfc_lun_queue_depth=128
lpfc_max_luns=65535
          options qla2xxx ql2xmaxqdepth=128 ql2xmaxlun=65535
          options fnic fnic_max_qdepth=128
    - path: /etc/udev/rules.d/99-infinidat-queue.rules
      mode: 0644
      overwrite: true
      contents:
        inline: |
          ACTION=="add|change", KERNEL=="sd[a-z]*", SUBSYSTEM=="block",
ENV{ID_VENDOR}=="NFINIDAT", ATTR{queue/scheduler}="none"
```

```
          ACTION=="add|change", KERNEL=="dm-*", SUBSYSTEM=="block",
ENV{DM_SERIAL}=="36742b0f*", ATTR{queue/scheduler}="none"
          ACTION=="add|change", KERNEL=="sd[a-z]*", SUBSYSTEM=="block",
ENV{ID_VENDOR}=="NFINIDAT", ATTR{queue/add_random}="0"
          ACTION=="add|change", KERNEL=="dm-*", SUBSYSTEM=="block",
ENV{DM_SERIAL}=="36742b0f*", ATTR{queue/add_random}="0"
          ACTION=="add|change", KERNEL=="sd[a-z]*", SUBSYSTEM=="block",
ENV{ID_VENDOR}=="NFINIDAT", ATTR{queue/rq_affinity}="2"
          ACTION=="add|change", KERNEL=="dm-*", SUBSYSTEM=="block",
ENV{DM_SERIAL}=="36742b0f*" , ATTR{queue/rq_affinity}="2"
    - path: /etc/multipath.conf
      mode: 0644
      overwrite: true
      contents:
        inline: |
          defaults {
            find_multipaths "yes"
          }
          devices {
            device {
              vendor "NFINIDAT"
              product "InfiniBox"
              path_grouping_policy "group_by_prio"
              path_checker "tur"
              features 0
              hardware_handler "1 alua"
              prio "alua"
              rr_weight "priorities"
              no_path_retry "queue"
              rr_min_io 1
              rr_min_io_rq 1
              flush_on_last_del "yes"
              fast_io_fail_tmo 15
              dev_loss_tmo "infinity"
              path_selector "service-time 0"
              failback "immediate"
              detect_prio "no"
              user_friendly_names "no"
            }
          }
```

[src: https://github.com/singlecheeze/smokejumper/blob/main/fc/infinidat/99-master-multipath-fc-udev-infinibox.bu, changed role example to worker]

MachineConfig Procedure

Assumes you have a client that can connect to the cluster and has **oc** client installed

1. Create a butane file **multipath-machineconfig-master.bu**
2. **vim multipath-machineconfig-worker.bu**
3. Convert the butane file into MachineConfig YAML
   a. **$ mkdir $HOME/bin**
   b. **$ curl -o $HOME/bin/butane https://mirror.openshift.com/pub/openshift-v4/clients/butane/latest/butane-amd64**
   c. **$ chmod a+x $HOME/bin/butane**
   d. **$ butane -V**
   e. **$ butane multipath-butane-worker.bu -o ./multipath-machineconfig-worker.yaml**
4. Apply the MachineConfig to the OpenShift Cluster
   a. **$ oc create -f multipath-machineconfig-master.yaml**
5. Wait for Node reboot (OpenShift reboots nodes one by one until all the configurations are applied to desired nodes).
6. Repeat steps 3e and 4 for different configurations.

KERNEL PARAMETERS

In some cases the max number of LUNs allowed per node may need to be increased through kernel and/or module parameters. The following examples show how to increase these values using kernel arguments, they are applied as a MachineConfig which will cause nodes to reboot:

Max LUN examples:

```
variant: openshift
version: 4.16.0
metadata:
  name: 99-master-max-arguments
  labels:
    machineconfiguration.openshift.io/role: master
openshift:
  kernel_arguments:
    - scsi_mod.max_luns=65535
    - lpfc.lpfc_max_luns=65535
    - qla2xxx.ql2xmaxlun=65535
storage:
  files:
  - path: /etc/sysctl.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        fs.inotify.max_user_watches=20000
  - path: /etc/security/limits.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        *   hard    nofile  20000
```

[src: https://github.com/singlecheeze/smokejumper/blob/main/ocp/99-master-max-arguments.bu]

Infinidat Deployment

INSTALL INFINIDAT CSI DRIVER OPERATOR

You can deploy the Infinidat CSI Driver by using the OpenShift Container Platform web console:

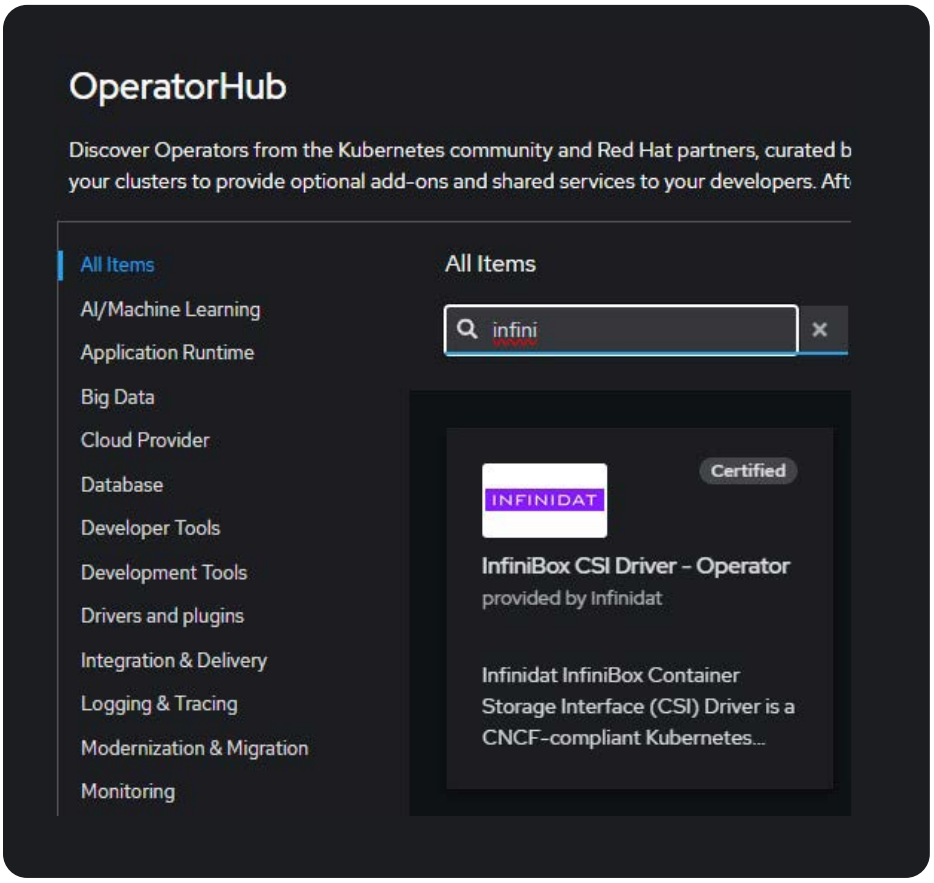
Prerequisites

- Red Hat OpenShift Container Platform 4.16 or newer installed on your nodes.
- An user with `cluster-admin` permissions, capable of logging in to the OpenShift Container Platform web console
- For Block Storage FC/ISCSI deploy Machine Config for multipathing
  - Documentation - Configuring RHEL 8 & RHEL 9 for Infinidat -Previous MachineConfig in the “Enable Multipath” section was built for the Best Practices below: https://support.infinidat.com/hc/en-us/articles/11985136447005-Setting-up-hosts-for-FC-on-RHEL-8-or-above-and-alternatives
  - Documentation - Installing the Infinidat CSI driver: https://support.infinidat.com/hc/en-us/articles/10106070174749-InfiniBox-CSI-Driver-for-Kubernetes-User-Guide

Infinidat CSI Driver Install Procedure

From the OpenShift Container Platform web console navigate to OperatorHub

1. Search Infinidat



2.

INFINIDAT

InfiniBox CSI Driver - Operator

2.20.0 provided by Infinidat

Install

Channel

alpha

Version

2.20.0

Capability level

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

Source

Infinidev

Provider

Infinidat

Infrastructure features

Disconnected
Container Storage Interface

Repository

https://github.com/Infinidat/infinibox-csi-driver

Container image

infinidat/infinidat-csi-driver

Created at

Mar 25, 2025, 9:19 AM

Support

Infinidat

Infinidat InfiniBox Container Storage Interface (CSI) Driver is a CNCF-compliant Kubernetes integration for InfiniBox storage systems, offering advanced enterprise functionality for petabyte-scale Kubernetes deployments including Red Hat OpenShift.

Features and Benefits

- **Multi-protocol flexibility** - manage Kubernetes Persistent Volumes attached via block and file protocols, including Fibre Channel, iSCSI, and NFS, with all Kubernetes PV access modes
- **Multi-petabyte scalability** - support hundreds of thousands of PVs per InfiniBox system and control multiple InfiniBox arrays within a single Kubernetes cluster \
- **Advanced enterprise features** - manage native InfiniBox snapshots and clones, including restoring from snapshots, and import PVs created outside of InfiniBox CSI Driver

Required Parameters

- `hostname` - IP address or hostname of the InfiniBox management interface
- `username` / `password` - InfiniBox credentials
- `SecretName` - secret name, to be used in the StorageClass to define a specific InfiniBox for persistent volumes

Optional Parameters

* `inbound_user` / `inbound_secret` / `outbound_user` / `outbound_secret` - credentials for iSCSI CHAP authentication

Installation Instructions

1. Create *infinidat-csi* namespace to install operator into.
2. Delete the following *clusterrolebindings*, if they exist, before installing operator
   - infinidat-csi-operator-infinidat-csi-attacher
   - infinidat-csi-operator-infinidat-csi-controller
   - infinidat-csi-operator-infinidat-csi-driver
   - infinidat-csi-operator-infinidat-csi-node
   - infinidat-csi-operator-infinidat-csi-provisioner
   - infinidat-csi-operator-infinidat-csi-resizer
   - infinidat-csi-operator-infinidat-csi-snapshotter
3. Install Infinidat Infinibox CSI Driver - Operator (this) into the *infinidat-csi* namespace.
4. Apply the following Security Context Constraints to the created service accounts in the *infinidat-csi* namespace
   - oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-controller-manager
   - oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-infinidat-csi-driver
   - oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-infinidat-csi-node

3. Follow the steps:

a. Create **`infinidat-csi`** namespace to install operator into: the name is arbitrary

b. Delete the following **`clusterrolebindings`**, if they exist, before installing operator

i. **`Infinidat-csi-operator-infinidat-csi-attacher`**

ii. **`Infinidat-csi-operator-infinidat-csi-controller`**

iii. **`Infinidat-csi-operator-infinidat-csi-driver`**

iv. **`Infinidat-csi-operator-infinidat-csi-node`**

v. **`Infinidat-csi-operator-infinidat-csi-provisioner`**

vi. **`Infinidat-csi-operator-infinidat-csi-resizer`**

vii. **`Infinidat-csi-operator-infinidat-csi-snapshotter`**

c. Install Infinidat InfiniBox CSI Driver - Operator (this) into the **`infinidat-csi namespace`**.

d. Apply the following Security Context Constraints to the created service accounts in the infinidat-csi namespace

```
oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-controller-manager
oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-infinidat-csi-driver
oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-infinidat-csi-node
oc adm policy add-scc-to-user privileged -z infinidat-csi-operator-infinidat-csi-controller
oc adm policy add-scc-to-user anyuid -z infinidat-csi-operator-controller-manager
oc adm policy add-scc-to-user anyuid -z infinidat-csi-operator-infinidat-csi-driver
oc adm policy add-scc-to-user anyuid -z infinidat-csi-operator-infinidat-csi-node
oc adm policy add-scc-to-user anyuid -z infinidat-csi-operator-infinidat-csi-controller
oc adm policy add-scc-to-user hostnetwork -z infinidat-csi-operator-controller-manager
oc adm policy add-scc-to-user hostnetwork -z infinidat-csi-operator-infinidat-csi-driver
oc adm policy add-scc-to-user hostnetwork -z infinidat-csi-operator-infinidat-csi-node
oc adm policy add-scc-to-user hostnetwork -z infinidat-csi-operator-infinidat-csi-controller
```

e. Recreate or edit any existing storage classes, pointing to the new namespace

f. If you choose a different namespace other than **`infinidat-csi`** you will need to edit the **`clusterrolebindings`** listed above to specify your custom namespace

g. Install an instance of Infinidat Driver into the **`infinidat-csi`** namespace

STORAGECLASS

What is a StorageClass

The StorageClass resource object describes and classifies storage that can be requested, as well as provides a means for passing parameters for dynamically provisioned storage on demand. StorageClass objects can also serve as a management mechanism for controlling different levels of storage and access to the storage.

Infinidat provides example files on GitHub for each protocol https://github.com/Infinidat/infinibox-csi-driver/tree/develop/deploy/examples

For VM workloads, Block RWX mode is recommended due to Virtual Machine being live migrated.

It is also important to highlight that after installing the OpenShift Virtualization add-on onto the cluster, it will automatically detect what StorageClass and providers installed on the platform. Once that is detected, it will automatically create a recommended and optimized StorageProfile for those workloads (VMs).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ibox-fc-block-rwx
provisioner: infinibox-csi-driver
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  csi.storage.k8s.io/controller-expand-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/controller-expand-secret-namespace: infinidat-csi
  csi.storage.k8s.io/controller-publish-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/controller-publish-secret-namespace: infinidat-csi
  csi.storage.k8s.io/node-publish-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/node-publish-secret-namespace: infinidat-csi
  csi.storage.k8s.io/node-stage-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/node-stage-secret-namespace: infinidat-csi
  csi.storage.k8s.io/provisioner-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/provisioner-secret-namespace: infinidat-csi
  csi.storage.k8s.io/node-expand-secret-name: $_E2E_IBOX_SECRET
  csi.storage.k8s.io/node-expand-secret-namespace: infinidat-csi
  csi.storage.k8s.io/fstype: ext4

  # Infinibox configuration
  pool_name: "csitesting"
  storage_protocol: "fc"

  # optional parameters
  #max_vols_per_host: "100"
  #provision_type: "THIN"
  #ssd_enabled: "false"
  uid: "3000" # UID of volume
  gid: "3000" # GID of volume
  #unix_permissions: "777" # optional volume mount permissions
```

[src: https://github.com/Infinidat/infinibox-csi-driver/blob/develop/deploy/examples/fc-block-rwx/storageclass.yaml]

OpenShift Virtualization will automatically receive automated boot source updates, only if a Default StorageClass is configured.

In order to achieve that, add the following annotation into the StorageClass object.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: ibox-fc-block-rwx
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
    ...
provisioner: infinibox-csi-driver
parameters:
  storage_protocol: "fc"
```

How to Setup Red Hat OpenShift Virtualization

Red Hat OpenShift Virtualization Supported Platforms

Red Hat OpenShift Virtualization Requirements

OpenShift Virtualization is an addon on top of the OpenShift Container Platform. The OpenShift Virtualization Operator can be deployed by 2 methods, using the OpenShift Console (User Interface) or declarative (yaml) via cli.

INSTALL RED HAT OPENSHIFT VIRTUALIZATION

You can deploy the OpenShift Virtualization Operator by using the OpenShift Container Platform web console.

Prerequisites

Red Hat OpenShift Container Platform 4.16 or newer installed on your nodes.

Procedure

1. Log in to the OpenShift Container Platform web console as a user with `cluster-admin` permissions.
2. From the **Administrator** perspective, click **Operators OperatorHub**.
3. In the **Filter by keyword** field, type **Virtualization**.
4. Select the **OpenShift Virtualization Operator** tile with the **Red Hat** source label.
5. Read the information about the Operator and click **Install**.
6. On the **Install Operator** page:
   a. Select **stable** from the list of available **Update Channel** options. This ensures that you install the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.
   b. For **Installed Namespace**, ensure that the **Operator recommended namespace** option is selected. This installs the Operator in the mandatory `openshift-cnv` namespace, which is automatically created if it does not exist.
   c. For **Approval Strategy**, it is highly recommended that you select **Automatic**, which is the default value, so that OpenShift Virtualization automatically updates when a new version is available in the **stable** update channel.

   d. While it is possible to select the **Manual** approval strategy, this is inadvisable because of the high risk that it presents to the supportability and functionality of your cluster. Only select **Manual** if you fully understand these risks and cannot use **Automatic**.

7. Click **Install** to make the Operator available to the `openshift-cnv` namespace.
8. When the Operator installs successfully, click **Create HyperConverged**.
9. Optional: Configure **Infra** and **Workloads** node placement options for OpenShift Virtualization components.
10. Click **Create** to launch OpenShift Virtualization.Verify that the OpenShift Virtualization is deployed by Navigating to **Workloads > Pods** to see all pods in the **Running** state.

The command line method installation is described in this section of the Red Hat OpenShift official documentation.

VM CREATION

At this point, OpenShift Virtualization is deployed and is relying on the default StorageClass from Infinidat to create Virtual Machines disks.

For a quick test, let's create a Virtual Machine from the Catalog.

Scale Test

For Red Hat OpenShift cluster deployments, it is recommended to have at least 3 nodes for control plane and additional workers, at least 2 to 3 for the data plane and customer applications.

Although since the adoption of OpenShift Virtualization is growing and bare metal servers have a lot of resources, customers can start with a 3 node cluster, where control plane and data plane can coexist. Customers can save on resources and start small and be able to scale out by adding nodes to the cluster as needed.

The testing scenario Red Hat OpenShift Virtualization was implemented in a 3 node cluster ready to scale out.

The scale test environment consisted of a 3-node Openshift cluster (two Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, 768GB of Ram 4Port FC HBA), Infinidat InfiniBox Fc connected 24 Paths).

Infinidat CSI and OpenShift Virtualization created 140VM per host in this test (limitation of cluster resources). That is 420 volumes on one host. In similar testing, we have seen greater numbers, but they are taxing the resources of the cluster.

KubeBurner was used to create VM's and namespaces. We implemented a wait between creating VM's to ease the utilization of cluster resources.

By default, Openshift has a max pod per host set at 250 pods.

This should be fine for most customers depending on their workload.

Customers can choose to change max pods per node by creating a MachineConfig. The MachineConfig Pool name being targeted in this example is master since the test environment was a compact 3 node cluster, in full cluster cases this pool name is typically `worker`.

Create a file on your admin server/vm etc.. with the following:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: set-max-pods
spec:
  machineConfigPoolSelector:
    matchLabels:
      pools.operator.machineconfiguration.openshift.io/master: ""
  kubeletConfig:
    maxPods: 500


$ oc apply -f max-pods-MachineConfig.yaml
```

In order to create a good load of VirtualMachines, `kube-burner` tool was used.

In our testing, we created 140 Virtual Machines (each VM has 3 disk/volumes) per node/host.

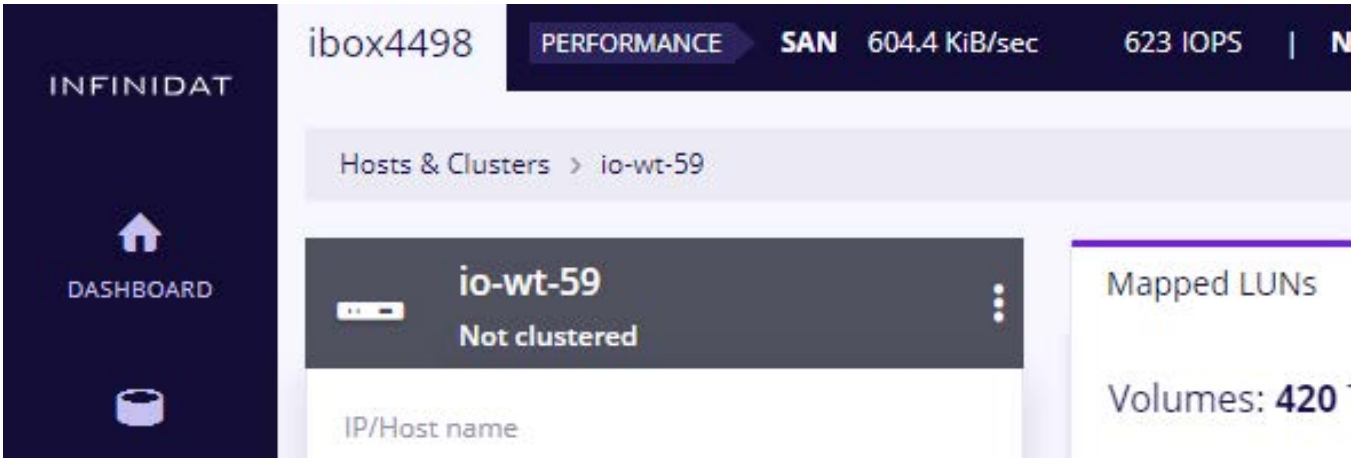
The script will create VMs from Openshift InstanceType.

On the InfiniBox a read/write snapshot is created for the OS disk and 2 new volumes for data disks. The associated volumes are presented to the worker node depending on the node selector or OpenShift will pick the least used Node.

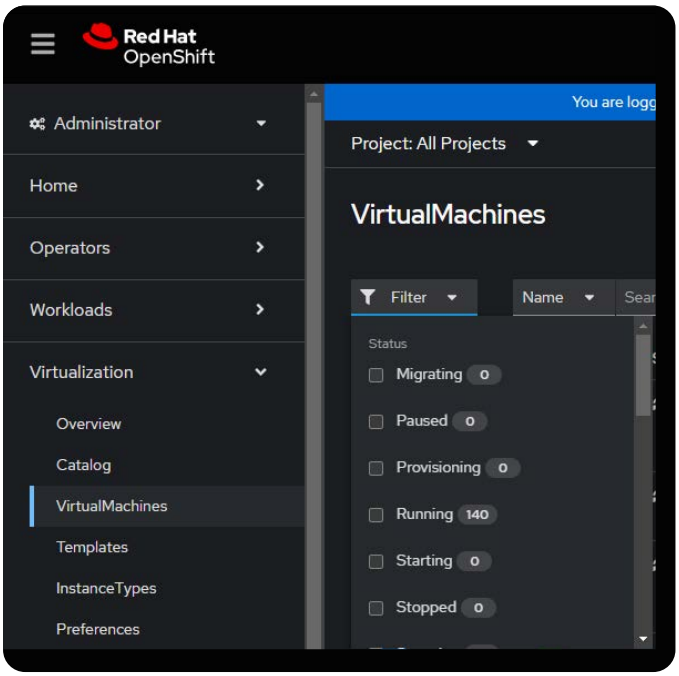
Note: Beware of the current limitation is 999 concurrent snapshots from the same InstanceType.

By using multiple templates for the same operating system customers can bypass this limitation.

Screenshot from InfiniBox:



Screenshot from OpenShift:



VM LIVE MIGRATION

Live migration is the process of moving a running virtual machine (VM) to another node in the cluster without interrupting the virtual workload. By default, live migration traffic is encrypted using Transport Layer Security (TLS).

For further details on the topic, refer to the official Red Hat OpenShift Virtualization documentation.

Note: By default OpenShift Virtualization limits live migration to 2 VMs per node in parallel: these migration limits can be configured as needed, as documented.

Results

The testing followed Red Hat's VM and storage deployment example workload scaling, as described above.

When testing high "bursts" of VM creation, it is recommended to increase the Virtualization control plane Queries Per Second (QPS) and Burst settings, as recommended by the OpenShift Virtualization - Tuning & Scaling Guide:

```
 oc patch -n openshift-cnv hco kubevirt-hyperconverged --type=json -p='[{"op": "add
"path": "/spec/tuningPolicy", "value": "highBurst"}]'
```

TESTED NUMBERS

Number of VMs	Number of Disks	Number of Paths (multipath configured)	Number of Hosts
140	420	5040	1
160	480	5760	1
480	1440	17280	3

References

https://www.infinidat.com/en/news/press-releases/infinidat-strengthens-relationship-red-hat-affirms-infiniboxr-and-csi-driver

https://www.infinidat.com/en/resource-pdfs/kubernetes-data-protection-using-infinidat-csi-driver.pdf

https://www.infinidat.com/en/resource-pdfs/infinibox-red-hat-openshift-virtualization.pdf

https://www.infinidat.com/en/resource-pdfs/infinidat-infinibox-datasheet-us.pdf

https://www.infinidat.com/en/partners/alliances/red-hat

https://catalog.redhat.com/partners/detail/infinidat

https://catalog.redhat.com/software/container-stacks/detail/5e9877cf3f398525a0ceb185

https://docs.redhat.com/en/documentation/openshift_container_platform/4.16/html/virtualization/live-migration

https://developers.redhat.com/articles/2024/09/04/use-kube-burner-measure-red-hat-openshift-vm-and-storage-deployment-scale

https://docs.redhat.com/en/documentation/openshift_container_platform/4.17/html/virtualization/installing#virt-subscribing-cli_installing-virt

https://docs.redhat.com/en/documentation/openshift_container_platform/4.17/html/virtualization/installing#virt-aws-bm_preparing-cluster-for-virt

https://docs.openshift.com/container-platform/4.17/installing/install_config/installing-customizing.html#installation-special-config-butane-create_installing-customizing

Other Whitepaper / Tech Doc References for Examples

https://www.infinidat.com/en/resource-pdfs/infinidat-storage-architecture.pdf

https://www.infinidat.com/en/resource-pdfs/esg-technical-validation-infiniguard-october-2022.pdf

Workload Reference

https://developers.redhat.com/articles/2024/09/04/use-kube-burner-measure-red-hat-openshift-vm-and-storage-deployment-scale

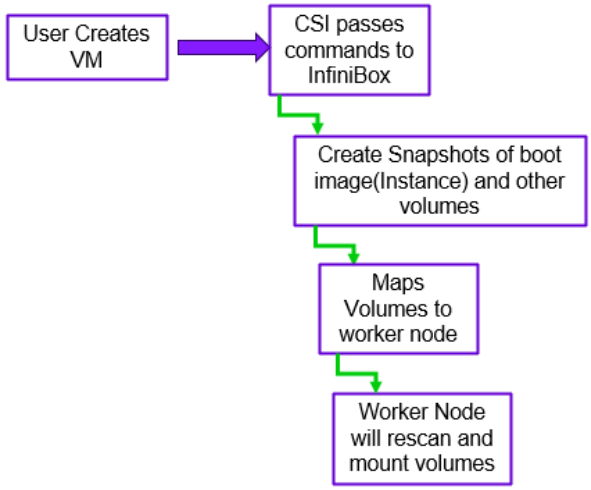
Appendix

DIAGRAM OF WORKFLOW

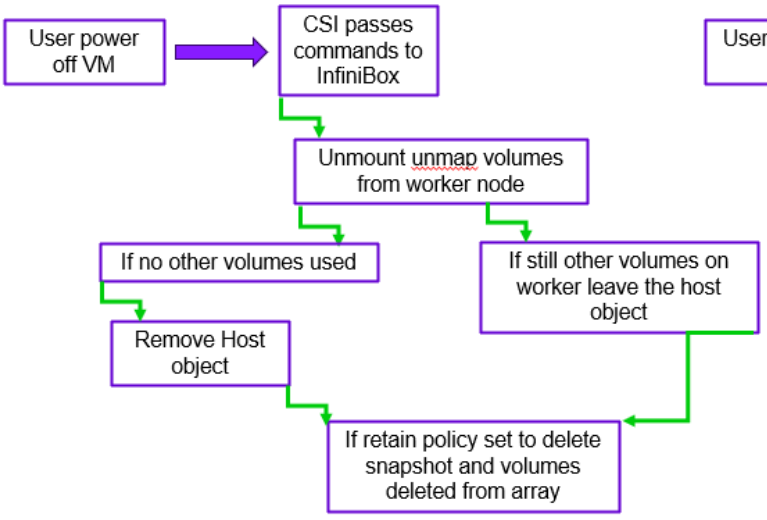
Create VM workflow, when CSI driver starts; delete VMs cleans up vs stop VM; VM migration (map to new node, unmap on orig node once migration completes)

When following Kubernetes scheduling constructs, a LUN may get mapped to one node for the data population stage, and another node when attaching to the pod or VM. For instance, when a VM is created using a DataVolume disk (which handles data population) a typical workflow has 2 scheduling phases: 1) the data population stage where a separate importer pod is scheduled and (for example, when using a url source) creates 2 PVCs for the import and population steps, then the final PVC request is Bound and 2) then the virt-launcher pod is scheduled and the volume id is attached to the pod, then the guest is launched.

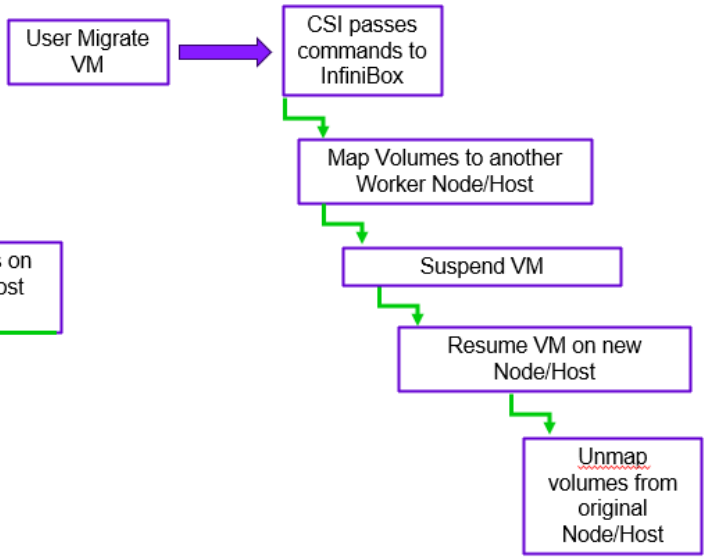
VM Creation



VM Power Off



VM Migration



sales@infinidat.com

WP-REDHATWP-2503275-US | © Infinidat 2025