

IBM block storage CSI driver  
1.2.0

*User Guide*



**Note**

Before using this document and the product it supports, read the information in [“Notices” on page 39.](#)

**Edition notice**

Publication number: SC27-9590-04. This publication applies to version 1.2.0 of IBM block storage CSI driver and to all subsequent releases and modifications until otherwise indicated in a newer publication.

© **Copyright International Business Machines Corporation 2020.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures.....</b>	<b>v</b>
<b>Tables.....</b>	<b>vii</b>
<b>About this guide.....</b>	<b>ix</b>
Who should use this guide.....	ix
Conventions used in this guide.....	ix
Related information and publications.....	ix
Getting information, help, and service.....	x
IBM Publications Center.....	x
Feedback.....	x
<b>Chapter 1. Introduction.....</b>	<b>1</b>
<b>Chapter 2. Installation.....</b>	<b>3</b>
Compatibility and requirements.....	3
Installing the operator and driver.....	5
Installing the driver using the OpenShift web console.....	6
Installing the driver using CLIs.....	7
Uninstalling.....	8
Uninstalling the driver using the OpenShift web console.....	8
Uninstalling the driver using CLIs.....	8
Upgrading the CSI driver.....	9
Upgrading the driver using the OpenShift web console.....	9
<b>Chapter 3. CSI driver configuration.....</b>	<b>11</b>
Creating an array secret.....	11
Creating storage classes.....	12
Creating a PersistentVolumeClaim (PVC).....	13
Creating a StatefulSet.....	14
Creating volume snapshots.....	16
Advanced configuration.....	17
Importing an existing volume.....	17
<b>Chapter 4. Using IBM block storage CSI driver.....</b>	<b>21</b>
Sample configurations for running a stateful container.....	21
Running a stateful container with file system configurations.....	22
Running a stateful container with raw block volume configurations.....	26
Recovering a pod volume attachment from a crashed Kubernetes node.....	30
<b>Chapter 5. Troubleshooting.....</b>	<b>33</b>
Log collection.....	33
Detecting errors.....	33
Multipath troubleshooting.....	34
Miscellaneous troubleshooting.....	35
<b>Notices.....</b>	<b>39</b>
Trademarks.....	40



---

# Figures

1. Integration of IBM block storage systems and CSI driver in a Kubernetes environment.....	2
---	---



---

# Tables

1. SpaceEfficiency parameter definitions per storage system type..... 12





# About this guide

---

This guide describes how to install, configure, and use the IBM block storage CSI driver.

## Who should use this guide

---

This guide is intended for system administrators who are familiar with container-based application delivery, orchestration methods, and with the specific IBM storage system that is in use.

## Conventions used in this guide

---

These notices are used in this guide to highlight key information.

---

**Note:** These notices provide important tips, guidance, or advice.

---

---

**Important:** These notices provide information or advice that might help you avoid inconvenient or difficult situations.

---



---

**Attention:** These notices indicate possible damage to programs, devices, or data. An attention notice appears before the instruction or situation in which damage can occur.

---

## Related information and publications

---

You can find additional information and publications related to IBM block storage CSI driver on the following information sources.

- [IBM SAN Volume Controller on IBM Knowledge Center \(ibm.com®/support/knowledgecenter/STPVGU\)](https://ibm.com/support/knowledgecenter/STPVGU)
- [IBM Spectrum Scale on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STXKQY\)](https://ibm.com/support/knowledgecenter/STXKQY)
- [IBM FlashSystem® 5000, 5100, and Storwize® V5000E on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STHGUJ\)](https://ibm.com/support/knowledgecenter/STHGUJ)
- [IBM FlashSystem 7200 and Storwize V7000 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/ST3FR7\)](https://ibm.com/support/knowledgecenter/ST3FR7)
- [IBM Spectrum Virtualize as Software Only on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STVLF4\)](https://ibm.com/support/knowledgecenter/STVLF4)
- [IBM FlashSystem 9200 and 9100 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STSLR9\)](https://ibm.com/support/knowledgecenter/STSLR9)
- [IBM FlashSystem A9000 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STJKMM\)](https://ibm.com/support/knowledgecenter/STJKMM)
- [IBM FlashSystem A9000R on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STJKN5\)](https://ibm.com/support/knowledgecenter/STJKN5)
- [IBM DS8880 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/ST5GLJ\)](https://ibm.com/support/knowledgecenter/ST5GLJ)
- [IBM DS8900 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/SSHGBU\)](https://ibm.com/support/knowledgecenter/SSHGBU)
- [Persistent volumes on Kubernetes \(kubernetes.io/docs/concepts/storage/volumes\)](https://kubernetes.io/docs/concepts/storage/volumes/)
- [Kubernetes Documentation \(kubernetes.io/docs/home/\)](https://kubernetes.io/docs/home/)
- [Kubernetes Blog \(kubernetes.io//blog\)](https://kubernetes.io/blog/)
- [IBM Spectrum® Access for IBM Cloud® Private Blueprint \(ibm.com/downloads/cas/KK5PGD8E\)](https://ibm.com/downloads/cas/KK5PGD8E)

Used as the FlexVolume driver based solution for OpenShift® 3.11, using [IBM Storage Enabler for Containers](http://ibm.com/support/knowledgecenter/SSCKLT) (ibm.com/support/knowledgecenter/SSCKLT)

- [IBM Storage for Red Hat® OpenShift Blueprint](http://www.redbooks.ibm.com/abstracts/redp5565.html?Open) (http://www.redbooks.ibm.com/abstracts/redp5565.html?Open)

## Getting information, help, and service

---

If you need help, service, technical assistance, or want more information about IBM products, you can find various sources to assist you. You can view the following websites to get information about IBM products and services and to find the latest technical information and support.

- [#get-help\\_csi](#) Slack channel
- [IBM website](http://ibm.com) (ibm.com)
- [IBM Support Portal website](http://ibm.com/support/entry/portal/support?brandind=Hardware~System_Storage) (ibm.com/support/entry/portal/support?brandind=Hardware~System\_Storage)
- [IBM Directory of Worldwide Contacts website](http://ibm.com/planetwide) (ibm.com/planetwide)

Use the Directory of Worldwide Contacts to find the appropriate phone number for initiating voice call support. Select the Software option, when using voice response system.

When asked, provide your Internal Customer Number (ICN) and/or the serial number of the storage system that requires support. Your call will then be routed to the relevant support team, to whom you can provide the specifics of your problem.

## IBM Publications Center

---

The IBM Publications Center is a worldwide central repository for IBM product publications and marketing material.

The [IBM Publications Center website](http://ibm.com/shop/publications/order) (ibm.com/shop/publications/order) offers customized search functions to help you find the publications that you need. You can view or download publications at no charge.

## Sending comments

---

Your feedback is important in helping to provide the most accurate and highest quality information.

### Procedure

To submit any comments about this publication or any other IBM storage product documentation:

- Send your comments by email to [starpubs@us.ibm.com](mailto:starpubs@us.ibm.com). Be sure to include the following information:
  - Exact publication title and version
  - Publication form number (for example, GA32-1234-00)
  - Page, table, or illustration numbers that you are commenting on
  - A detailed description of any information that should be changed

---

# Chapter 1. Introduction

IBM block storage CSI driver is leveraged by Kubernetes persistent volumes (PVs) to dynamically provision for block storage used with stateful containers.

IBM block storage CSI driver is based on an open-source IBM project ([CSI driver](#)), included as a part of IBM storage orchestration for containers. IBM storage orchestration for containers enables enterprises to implement a modern container-driven hybrid multicloud environment that can reduce IT costs and enhance business agility, while continuing to derive value from existing systems.

By leveraging CSI (Container Storage Interface) drivers for IBM storage systems, Kubernetes persistent volumes (PVs) can be dynamically provisioned for block or file storage to be used with stateful containers, such as database applications (IBM Db2®, MongoDB, PostgreSQL, etc) running in Red Hat OpenShift Container Platform and/or Kubernetes clusters. Storage provisioning can be fully automatized with additional support of cluster orchestration systems to automatically deploy, scale, and manage containerized applications.

IBM storage orchestration for containers includes the following driver types for storage provisioning:

- The IBM block storage CSI driver, for block storage (documented here).
- The IBM Spectrum Scale CSI driver, for file storage. For more information on Spectrum Scale and the Spectrum Scale CSI driver see the [IBM Spectrum Scale knowledge center website](#) ([ibm.com/support/knowledgecenter/STXKQY](http://ibm.com/support/knowledgecenter/STXKQY)).

For details about volume provisioning with Kubernetes, refer to [Persistent volumes on Kubernetes](#) ([kubernetes.io/docs/concepts/storage/volumes](http://kubernetes.io/docs/concepts/storage/volumes)).

---

**Note:** For the user convenience, this guide might refer to IBM block storage CSI driver as CSI driver.

---

# Kubernetes Cluster

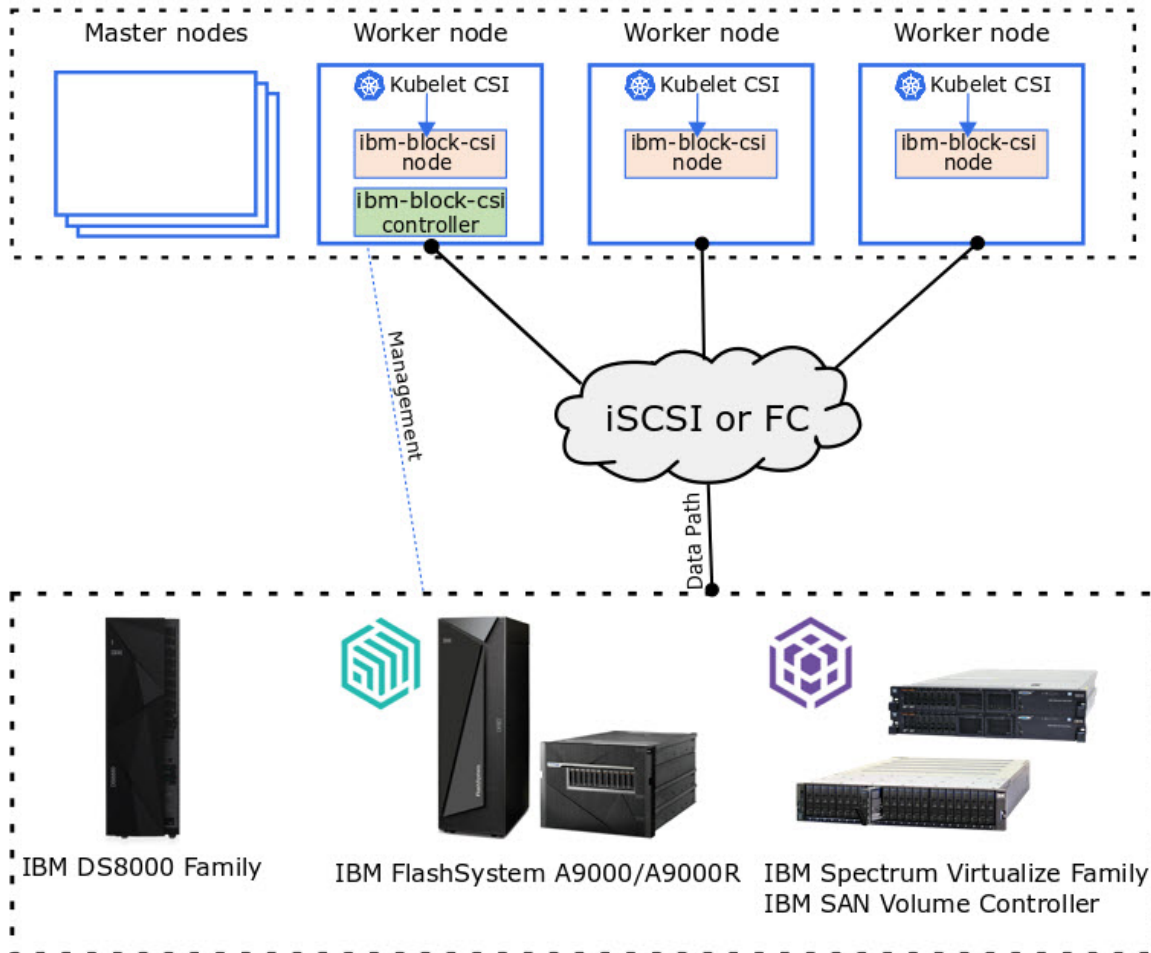


Figure 1. Integration of IBM block storage systems and CSI driver in a Kubernetes environment

---

## Chapter 2. Installation

Download and install the IBM block storage CSI driver installation package your container platform (such as Kubernetes), as described in the following sections.

- [“Compatibility and requirements” on page 3](#)
- [“Installing the operator and driver” on page 5](#)
- [“Upgrading the CSI driver” on page 9](#)

For information about uninstallation, see [“Uninstalling” on page 8](#).

---

### Compatibility and requirements

For the complete and up-to-date information about the compatibility and requirements for using the IBM block storage CSI driver, refer to its latest release notes. The release notes detail supported operating system and container platform versions, as well as microcode versions of the supported storage systems. You can find the latest release notes on [IBM block storage CSI driver Knowledge Center website](https://ibm.com/support/knowledgecenter/SSRQ8T) (ibm.com/support/knowledgecenter/SSRQ8T).

#### Before you begin

Before beginning the installation of the CSI (Container Storage Interface) driver be sure to verify that you comply with the following prerequisites.

1. The CSI driver requires the following ports to be opened on the worker nodes OS firewall:

**For all iSCSI users**

Port 3260

**FlashSystem A9000 and A9000R**

Port 7778

**IBM Spectrum Virtualize Family**

**includes IBM SAN Volume Controller and IBM FlashSystem family members built with IBM Spectrum Virtualize (FlashSystem 5010, 5030, 5100, 7200, 9100, 9200, 9200R)**

Port 22

**DS8000® Family systems**

Port 8452

2. Be sure that multipathing is installed and running.

For iSCSI single path users (RHEL only), be sure to define a virtual multipath. For example, remove **find\_multipaths yes** from the `multipath.conf` file.

#### About this task

Perform these steps for each worker node in Kubernetes cluster to prepare your environment for installing the CSI (Container Storage Interface) driver.

#### Procedure

1. Ensure iSCSI connectivity, when using RHEL OS. If using RHCOS or if the packages are already installed, continue to step [“2” on page 4](#).
  - `iscsi-initiator-utils` (if iSCSI connection is required)
  - `xfsprogs` (if XFS file system is required)

```
yum -y install iscsi-initiator-utils
yum -y install xfsprogs
```

2. Configure Linux® multipath devices on the host, using one of the following procedures.

## Configuring for OpenShift Container Platform users (RHEL and RHCOS)

The following yaml file can be used for both Fibre Channel and iSCSI configurations. To support iSCSI, uncomment the last two lines in the file.

**Important:** The `99-ibm-attach.yaml` configuration file overrides any files that already exist on your system. Only use this file if the files mentioned are not already created.

If one or more have been created, edit this yaml file, as necessary.

Save the following 99-ibm-attach.yaml file.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-ibm-attach
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
        - path: /etc/multipath.conf
          mode: 384
          filesystem: root
          contents:
            source: data:,defaults%20%7B%0A%20%20%20path_checker%20tur%0A%20%20%20path_selector%20%22round-robin%200%22%0A%20%20%20%20rr_weight%20uniform%0A%20%20%20prio%20const%0A%20%20%20%20rr_min_io_rq%201%20%20%20%20%20%20%20%20%20%20%20%20%20%20polling_interval%2030%0A%20%20%20%20path_grouping_policy%20multibus%0A%20%20%20find_multipaths%20yes%0A%20%20%20no_path_retry%20fail%0A%20%20%20user_friendly_names%20yes%0A%20%20%20failback%20immediate%0A%20%20%20checker_timeout%2010%0A%20%20%20fast_io_fail_tmo%20off%0A%20%20%20devices%20%7B%0A%20%20%20device%20%7B%0A%20%20%20%20path_checker%20tur%0A%20%20%20product%20%22FlashSystem%22%0A%20%20%20vendor%20%22IBM%22%0A%20%20%20%20rr_weight%20uniform%0A%20%20%20%20rr_min_io_rq%204%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20path_grouping_policy%20multibus%0A%20%20%20%20path_selector%20%22round-robin%200%22%0A%20%20%20%20no_path_retry%20fail%0A%20%20%20%20failback%20immediate%0A%20%20%20%20device%20%7B%0A%20%20%20%20path_checker%20tur%0A%20%20%20%20fast_io_fail_tmo%20off%0A%20%20%20%20rr_weight%20uniform%0A%20%20%20%20rr_min_io_rq%201000%20%20%20%20%20%20%20path_selector%20%22round-robin%200%22%0A%20%20%20%20no_path_retry%20fail%0A%20%20%20%20failback%20immediate%0A%20%20%20%20device%20%7B%0A%20%20%20vendor%20%22IBM%22%0A%20%20%20product%20%22145%22%0A%20%20%20%20path_checker%20tur%0A%20%20%20%20features%20%221%20queue_if_no_path%22%0A%20%20%20%20path_grouping_policy%20group_by_prio%0A%20%20%20%20path_selector%20service-time%20%22%20%23%20Used%20by%20RedHat%207.x%0A%20%20%20%20prio%20alua%0A%20%20%20%20rr_min_io_rq%201%0A%20%20%20%20rr_weight%20uniform%20%0A%20%20%20%20no_path_retry%2025%22%0A%20%20%20dev_loss_tmo%20120%0A%20%20%20%20failback%20immediate%0A%20%20%20%0A%20%7D%0A
            verification: {}
        - path: /etc/udev/rules.d/99-ibm-2145.rules
          mode: 420
          filesystem: root
          contents:
            source: data:,%23%20Set%20SCSI%20command%20timeout%20to%20120s%20%28default%203D%3D%2030%20or%2060%29%20for%20IBM%202145%20devices%0ASUBSYSTEM%3D%3D%22block%22%2C%20ACTION%3D%3D%22add%22%2C%20ENV%7BID_VENDOR%7D%3D%3D%22IBM%22%2CENV%7BID_MODEL%7D%3D%3D%22145%22%2C%20RUN%2B%3D%22/bin/sh%20-c%20%27echo%20120%20%3E/sys/block/%25k/device/timeout%27%22%0A
            verification: {}
  systemd:
    units:
      - name: multipathd.service
        enabled: true
        # Uncomment the following lines if this MachineConfig will be used with iSCSI connectivity
        # name: iscsid.service
        # enabled: true
```

Apply the yaml file.

```
oc apply -f 99-ibm-attach.yaml
```

RHEL users should verify that the `systemctl status multipathd` output indicates that the multipath status is active and error-free.

```
yum install device-mapper-multipath
sudo modprobe dm-multipath
systemctl enable multipathd
systemctl start multipathd
systemctl status multipathd
multipath -ll
```

### Configuring for Kubernetes users (RHEL)

Create and set the relevant storage system parameters in the `/etc/multipath.conf` file. You can also use the default `multipath.conf` file, located in the `/usr/share/doc/device-mapper-multipath-*` directory.

Verify that the `systemctl status multipathd` output indicates that the multipath status is active and error-free.

```
yum install device-mapper-multipath
sudo modprobe dm-multipath
systemctl enable multipathd
systemctl start multipathd
systemctl status multipathd
multipath -ll
```

3. If needed, enable support for volume snapshots (FlashCopy® function) on your Kubernetes cluster.

For more information and instructions, see the Kubernetes blog post, [Kubernetes 1.17 Feature: Kubernetes Volume Snapshot Moves to Beta](https://kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-cis-volume-snapshot-beta/). ([kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-cis-volume-snapshot-beta/](https://kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-cis-volume-snapshot-beta/)).

- a) Install the Snapshot CRDs, once per cluster.

The relevant yaml files to apply are located in <https://github.com/kubernetes-csi/external-snapshotter/tree/master/config/crd>.

- b) Install the Common Snapshot Controller, once per cluster.

The relevant yaml files to apply are located in <https://github.com/kubernetes-csi/external-snapshotter/tree/master/deploy/kubernetes/snapshot-controller>.

4. Configure storage system connectivity.

- a) Define the host of each Kubernetes node on the relevant storage systems with the valid WWPN (for Fibre Channel) or IQN (for iSCSI) of the node.
- b) For Fibre Channel, configure the relevant zoning from the storage to the host.

## Installing the operator and driver

Install the operator for IBM block storage CSI driver in order to deploy, install, and manage the CSI (Container Storage Interface) driver.

Use one of the following procedures to download the operator and driver:

- If using Red Hat OpenShift Container Platform, follow the instructions detailed in [“Installing the driver using the OpenShift web console”](#) on page 6.
- If Red Hat OpenShift Container Platform is not being used, follow the instructions detailed in [“Installing the driver using CLIs”](#) on page 7.

## Installing the driver using the OpenShift web console

When using the Red Hat OpenShift Container Platform, the operator for IBM block storage CSI driver can be installed directly from OpenShift web console, through the OperatorHub. Installing the CSI (Container Storage Interface) driver is part of the operator installation process.

### Before you begin

**Note:** These instructions should only be used for Red Hat OpenShift Container Platform with x86 architecture. If you are using OpenShift Container Platform with IBM Z® or PowerPC®, or if you are not using the OpenShift Container Platform, follow the instructions detailed in [“Installing the driver using CLIs”](#) on page 7.

### About this task

The Red Hat OpenShift Container Platform uses the following **SecurityContextConstraints** for the following **serviceAccounts**:

**Note:** This data is for informational purposes only.

serviceAccount	SecurityContextConstraint
ibm-block-csi-operator	restricted
ibm-block-csi-controller-sa	anyuid
ibm-block-csi-node-sa	privileged

### Procedure

1. From Red Hat OpenShift Container Platform **Home** > **Projects**, click **Create Project**. In the **Create Project** dialog box, fill in a Project name (also referred to as *namespace*). Click **Create** to save.
2. From **Operators** > **OperatorHub**. Select the namespace from **Projects: <namespace>**, defined in step “1” on page 6.
3. Search for IBM block storage CSI driver.
4. Select the **Operator for IBM block storage CSI driver** and click **Install**.  
The Operator Subscription form appears.
5. Set the **Installation Mode** to the project namespace selected above, in step “2” on page 6, under **A specific namespace on the cluster**.
6. **(Optional):** Set the **Approval Strategy** to **Automatic**.
7. Click **Subscribe**.
8. From **Operators** > **Installed Operators**, check the status of the Operator for IBM block storage CSI driver.  
Wait until the **Status** is *Up to date* and then *InstallSucceeded*.

**Note:** While waiting for the **Status** to change from *Up to date* to *InstallSucceeded*, you can check the pod progress and readiness status from **Workloads** > **Pods**.

9. Once the operator installation progress has completed, click on the installed Operator for IBM block storage CSI driver.
10. Click **Create Instance** to create the IBM block storage CSI driver (IBMBlockCSI).  
A yaml file opens in the web console. This can be left as-is, or edited as needed.
11. Update the yaml file to include your user-defined namespace
12. Click **Create**.  
Wait until the **Status** is *Running*.



## Installing the driver using CLIs

The operator for IBM block storage CSI driver can be installed directly from GitHub from a command line terminal. Installing the CSI (Container Storage Interface) driver is part of the operator installation process.

### Before you begin

---

**Note:** These instructions are for command-line terminal users only and for Red Hat OpenShift users with IBM Z or PowerPC architecture. For other Red Hat OpenShift Container Platform users, follow the instructions detailed in [“Installing the driver using the OpenShift web console” on page 6](#).

---

### Procedure

Use the following steps to install the operator and driver, using [GitHub](#) ([github.com/IBM/ibm-block-csi-operator](https://github.com/IBM/ibm-block-csi-operator)) through a command line terminal.

1. Download the manifest from GitHub.

#### For Red Hat OpenShift users with IBM Z or PowerPC architecture

```
https://github.com/IBM/ibm-block-csi-operator/releases/download/v1.2.0/ibm-block-csi-operator-non-x86.yaml > ibm-block-csi-operator.yaml
```

#### For all other users

```
curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/v1.2.0/deploy/installer/generated/ibm-block-csi-operator.yaml > ibm-block-csi-operator.yaml
```

2. **(Optional):** If required, update the image fields in the `ibm-block-csi-operator.yaml`.

---

**Note:** Updating the namespace to a user-defined namespace may be necessary in order to ensure consistency and avoid trouble installing the operator.

---

3. Use the `kubectl create ns <namespace>` command to create a project namespace.
4. Install the operator, while using a user-defined namespace.

```
$ kubectl -n <namespace> apply -f ibm-block-csi-operator.yaml
```

5. Verify the operator running. (Make sure the Status is *Running*.)

```
$ kubectl get pod -l app.kubernetes.io/name=ibm-block-csi-operator -n <namespace>
NAME                                READY   STATUS    RESTARTS   AGE
ibm-block-csi-operator-5bb7996b86-xntss 1/1     Running   0           10m
```

6. Install the IBM block storage CSI driver by creating an IBMBlockCSI custom resource.

- a) Download the manifest from GitHub.

#### For Red Hat OpenShift users with IBM Z or PowerPC architecture

```
curl -L https://github.com/IBM/ibm-block-csi-operator/releases/download/v1.2.0/csi.ibm.com_v1.2_ibmblockcsi_cr_ocp.yaml > csi.ibm.com_v1_ibmblockcsi_cr.yaml
```

#### For all other users

```
curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/v1.2.0/deploy/crds/csi.ibm.com_v1_ibmblockcsi_cr.yaml > csi.ibm.com_v1_ibmblockcsi_cr.yaml
```

- b) **(Optional):** If required, update the image repository and/or tag fields in the `csi.ibm.com_v1_ibmblockcsi_cr.yaml`.

---

**Note:** Updating the namespace to a user-defined namespace may be necessary in order to ensure consistency and avoid trouble installing the operator.

---

c) Install the `csi.ibm.com_v1_ibmblockcsi_cr.yaml`.

```
$ kubectl -n <namespace> apply -f csi.ibm.com_v1_ibmblockcsi_cr.yaml
```

## Uninstalling

---

Use this information to uninstall the IBM CSI (Container Storage Interface) operator and driver.

Use one of the following procedures to uninstall the operator and driver:

- If using Red Hat OpenShift Container Platform, follow the instructions detailed in [“Uninstalling the driver using the OpenShift web console”](#) on page 8.
- If Red Hat OpenShift Container Platform is not being used, follow the instructions detailed in [“Uninstalling the driver using CLIs”](#) on page 8.

### Uninstalling the driver using the OpenShift web console

Use this information to uninstall the IBM CSI (Container Storage Interface) operator and driver through the Red Hat OpenShift Container Platform web console.

#### About this task

---

**Note:** These instructions are for Red Hat OpenShift Container Platform users only. If you are not using the Red Hat OpenShift Container Platform, follow the instructions detailed in [“Uninstalling the driver using CLIs”](#) on page 8.

---

#### Procedure

Perform the following steps in order to uninstall the CSI driver and operator through Red Hat OpenShift Container Platform web console.

1. From the web console go to **Operators > Installed Operators**. Select the Project namespace, where installed, from **Projects: <namespace>**.
2. Select **Operator for IBM block storage CSI driver**.
3. Select **IBM block storage CSI driver**.

**Operators > Installed Operators > Operator Details.**

4. Click on the **more** menu for the **ibm-block-csi** driver and select **Delete IBMBlock CSI**.

Wait for the controller and node pods to terminate.

This deletes the CSI driver. Continue to step [“5”](#) on page 8 to delete the operator for IBM block storage CSI driver.

5. From the **Installed Operators** page, click on the **more** menu for the **Operator for IBM block storage CSI driver** and select **Uninstall Operator**.

### Uninstalling the driver using CLIs

Use this information to uninstall the IBM CSI (Container Storage Interface) operator and driver from a command line terminal.

#### About this task

---

**Note:** These instructions are for command-line terminal users only. If using the Red Hat OpenShift Container Platform, follow the instructions detailed in [“Uninstalling the driver using the OpenShift web console”](#) on page 8.

---

## Procedure

Perform the following steps in order to uninstall the CSI driver and operator from a command line terminal.

1. Delete the IBMBlockCSI custom resource.

```
$ kubectl delete -f csi.ibm.com_v1_ibmblockcsi_cr.yaml
```

2. Delete the operator.

```
$ kubectl delete -f ibm-block-csi-operator.yaml
```

## Upgrading the CSI driver

---

Use this information to upgrade the IBM block storage CSI driver.

In order to upgrade the CSI (Container Storage Interface) driver from a previous version, uninstall the existing driver and then install the newer version.

- For instructions on how to uninstall the driver, see [“Uninstalling the driver using CLIs” on page 8](#).
- For instructions on how to install the newer driver, see [“Installing the driver using CLIs” on page 7](#).

For upgrading using the OpenShift web console, see [“Upgrading the driver using the OpenShift web console” on page 9](#).

## Upgrading the driver using the OpenShift web console

When using the Red Hat OpenShift Container Platform, the operator for IBM block storage CSI driver can be upgraded directly from OpenShift web console, through the OperatorHub.

---

**Note:** These instructions are for Red Hat OpenShift Container Platform users only. If you are not using the OpenShift Container Platform, follow the instructions detailed in [“Installing the driver using CLIs” on page 7](#).

---

When the Subscription **Approval** policy for the Operator for IBM block storage CSI driver is set to **Manual**, follow the procedure detailed in [“Manual upgrade with OpenShift” on page 9](#).

### Manual upgrade with OpenShift

When using the Red Hat OpenShift Container Platform, the CSI (Container Storage Interface) driver can be manually updated through the OpenShift web console.

## Procedure

1. From Red Hat OpenShift Container Platform **Operators > Installed Operators** see the status of the **ibm-block-csi-operator**.

If the **Status** is *UpgradePending*, click on the operator.

2. From the **Subscription Overview** view, click on **1 requires approval**.

The **Review Manual Install Plan** notice appears.

3. Click on **Preview Install Plan**.

4. Review the manual install plan and click **Approve**.

5. From the **Subscription** tab, check the upgrade status and the installed version.

6. From **Operators > Installed Operators > Operator for IBM block storage CSI driver**, click **Create Instance**.

7. Check the **Subscriptions > Subscription Overview** tab see the Operator status.

Wait for the **Upgrade Status** to be **Upgrading** and **1 requires approval** appears.

8. Click **1 requires approval**.

The **Review Manual Install Plan** notice appears.

9. Click on **Preview Install Plan**.
10. Review the manual install plan and click **Approve**.
11. From the **Subscription** tab, check the upgrade status and the installed version.
12. Check the **Overview** tab and that the **Controller Image Tab** and **Node Image Tag** are showing the most up-to-date version of the driver and the **Status** is *Running*.

---

## Chapter 3. CSI driver configuration

Use this information to configure the IBM block storage CSI driver after installation.

Once the driver is installed and running (see [“Installing the operator and driver” on page 5](#)), in order to use the driver and run stateful applications using IBM block storage systems, the relevant storage classes and secrets must be created.

- [“Creating an array secret” on page 11](#)
- [“Creating storage classes” on page 12](#)
- [“Advanced configuration” on page 17](#)

---

### Creating an array secret

Create a storage system secret in order to define the storage credentials (username and password) and address.

---

**Important:** When your storage system password is changed, be sure to also change the passwords in the corresponding secrets, particularly when LDAP is used on the storage systems.

Failing to do so causes mismatched passwords across the storage systems and the secrets, causing the user to be locked out of the storage systems.

---

Use one of the following procedures to create and apply the secret:

#### Creating an array secret file

1. Create the secret file.

```
kind: Secret
apiVersion: v1
metadata:
  name: <NAME>
  namespace: <NAMESPACE>
type: Opaque
stringData:
  management_address: <ADDRESS_1,ADDRESS_2> # Array management addresses
  username: <USERNAME> # Array username
data:
  password: <PASSWORD base64> # Array password
```

2. Apply the secret using the following command:

```
$> kubectl apply -f array-secret.yaml
```

The secret/*<NAME>* created message is emitted.

#### Creating an array secret via command line

---

**Note:** This procedure is applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

---

Create the secret using the following command:

```
kubectl create secret generic <NAME> --from-literal=username=<USER> --from-
literal=password=<PASSWORD>
--from-literal=management_address=<ARRAY_MGMT> -n <namespace>
```

## Creating storage classes

Create storage classes in order to define the storage system pool name, secret reference, **SpaceEfficiency**, and **fstype**.

### Procedure

Use the following procedure to create and apply the storage classes.

#### Note:

- This procedure is applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

1. Create a storage class yaml file, `storageclass-gold.yaml`.

Update the capabilities, pools, and array secrets accordingly.

Use the **SpaceEfficiency** parameters for each storage system, as defined in [Table 1 on page 12](#). These values are not case sensitive.

Table 1. <b>SpaceEfficiency</b> parameter definitions per storage system type	
Storage system type	SpaceEfficiency parameter options
IBM FlashSystem A9000 and A9000R	Always includes deduplication and compression. No need to specify during configuration.
IBM Spectrum Virtualize Family	<ul style="list-style-type: none"><li>• <b>thin</b></li><li>• <b>compressed</b></li><li>• <b>deduplicated</b></li></ul>
IBM DS8000 Family	<ul style="list-style-type: none"><li>• <b>standard</b> (default value)</li><li>• <b>thin</b></li></ul> <b>Note:</b> If not specified, the default value is <b>standard</b> .

- The IBM DS8000 Family **pool** value is **VALUE\_POOL\_ID** (and not **VALUE\_POOL\_NAME** as indicated in the example below).
- The **csi.storage.k8s.io/fstype** parameter is optional. The values allowed are **ext4** or **xfs**. The default value is **ext4**.
- The **volume\_name\_prefix** parameter is optional.
  - For IBM DS8000 Family, the maximum prefix length is 5 characters.
  - The maximum prefix length for other systems is 20 characters.

```

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <NAME>
provisioner: block.csi.ibm.com
parameters:
  #capabilities:                                # Optional
  # SpaceEfficiency=<VALUE>
  capacity:
    pool=<POOL_NAME>

csi.storage.k8s.io/provisioner-secret-name: <ARRAY_SECRET>
csi.storage.k8s.io/provisioner-secret-namespace: <ARRAY_SECRET_NAMESPACE>
csi.storage.k8s.io/controller-publish-secret-name: <ARRAY_SECRET>
csi.storage.k8s.io/controller-publish-secret-namespace: <ARRAY_SECRET_NAMESPACE>

#csi.storage.k8s.io/fstype: <FSTYPE>           # Optional. values ext4\xfs. The default is ext4.
#volume_name_prefix: <PREFIX>                 # Optional

```

2. Apply the storage class.

```
$> kubectl apply -f storageclass-gold.yaml
```

The `storageclass.storage.k8s.io/gold` created message is emitted.

## Creating a PersistentVolumeClaim (PVC)

Use this information for PersistentVolumeClaim (PVC) configuration settings.

The IBM block storage CSI driver supports using both file system and raw block volume types.

---

**Important:** If not defined, the default type is **Filesystem**. Be sure to define the type as **Block** if this configuration is preferred.

---



---

**Note:** The examples below create the PVC with a storage size 1 Gb. This can be changed, per customer needs.

---

Use the sections below for yaml creation of PVCs with file system and raw block volume types. After yaml creation, use the **kubectl apply** command.

```
$> kubectl apply -f <filename>.yaml
```

The `persistentvolumeclaim/<filename>` created message is emitted.

### Creating PVC for volume with file system

Create a PVC `demo-pvc-file-system.yaml` file with the size of 1 Gb.

---

**Note:** `volumeMode` is an optional field. **Filesystem** is the default if the value is not added.

---

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc-file-system
spec:
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gold

```

## Creating PVC for raw block volume

Create a PVC demo-pvc-raw-block yml file with the size of 1 Gb.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc-raw-block
spec:
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gold
```

## Creating a StatefulSet

Use this information for StatefulSet configuration settings.

The IBM block storage CSI driver supports using both file system and raw block volume types.

StatefulSets can include volumes with file systems, raw block volume systems, or both.

---

**Important:** When defining the StatefulSet configuration, be sure to define volumes according to the PVC type.

---

Use the sections below for yml creation of StatefulSets with file system, raw block volume, and mixed types. After yml creation, use the **kubect1 apply** command.

```
$> kubect1 apply -f <filename>.yaml
```

The statefulset/<filename> created message is emitted.

### Creating a StatefulSet with file system volumes

Create a StatefulSet, using the demo-statefulset-file-system.yaml.

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset-file-system
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-statefulset
    spec:
      containers:
        - name: container-demo
          image: registry.access.redhat.com/ubi8/ubi:latest
          command: [ "/bin/sh", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
          volumeMounts:
            - name: demo-volume
              mountPath: "/data"
      volumes:
        - name: demo-volume
          persistentVolumeClaim:
            claimName: demo-pvc-file-system

#      nodeSelector:
#        kubernetes.io/hostname: HOSTNAME
```



## Creating a StatefulSet with raw block volume

Create a StatefulSet, using the `demo-statefulset-raw-block.yaml`.

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset-raw-block
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-statefulset
    spec:
      containers:
        - name: container-demo
          image: registry.access.redhat.com/ubi8/ubi:latest
          command: [ "/bin/sh", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
          volumeDevices:
            - name: demo-volume
              devicePath: "/dev/block"
      volumes:
        - name: demo-volume
          persistentVolumeClaim:
            claimName: demo-pvc-raw-block

#   nodeSelector:
#     kubernetes.io/hostname: HOSTNAME
```

## Creating a StatefulSet with both raw block and file system volumes

Create a StatefulSet, using the `demo-statefulset-combined.yaml`.

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset-combined
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-statefulset
    spec:
      containers:
        - name: container-demo
          image: registry.access.redhat.com/ubi8/ubi:latest
          command: [ "/bin/sh", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
          volumeMounts:
            - name: demo-volume-file-system
              mountPath: "/data"
          volumeDevices:
            - name: demo-volume-raw-block
              devicePath: "/dev/block"
      volumes:
        - name: demo-volume-file-system
          persistentVolumeClaim:
            claimName: demo-pvc-file-system
        - name: demo-volume-raw-block
          persistentVolumeClaim:
            claimName: demo-pvc-raw-block

#   nodeSelector:
#     kubernetes.io/hostname: NODESELECTOR
```

## Creating volume snapshots

Use this information for volume snapshot configuration settings.

### Note:

- IBM FlashCopy function is referred to as the more generic volume snapshots within this documentation set. Not all supported products use the *FlashCopy function* terminology.
- For volume snapshot support, the minimum orchestration platform version requirements are Red Hat OpenShift 4.4 and Kubernetes 1.17.

When creating volume snapshots, be sure to follow all of the Snapshot configurations, found in [“Compatibility and requirements”](#) on page 3, prior to Snapshot creation.

### Enabling creation and deletion of volume snapshots

In order to enable creation and deletion of volume snapshots, create a VolumeSnapshotClass yaml file.

Use the following `template-snapshotclass.yaml` to configure the file for your storage system.

When configuring the file, be sure to use the same array secret and array secret namespace as defined in [“Creating an array secret”](#) on page 11.

**Note:** When working with IBM DS8000 Family storage systems, the pool parameter is mandatory. For all other storage systems, this parameter is optional.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: <NAME>
driver: block.csi.ibm.com
deletionPolicy: <DELETION_PRIORITY> # Retain or Delete
parameters:
  csi.storage.k8s.io/snapshotter-secret-name: <ARRAY_SECRET>
  csi.storage.k8s.io/snapshotter-secret-namespace: <ARRAY_SECRET_NAMESPACE>
  pool: <POOL_ID> # Mandatory for DS8000 Family storage systems.
  snapshot_name_prefix: <NAME_PREFIX> # Optional.
```

After the yaml file is created, apply it by using the **kubectl apply -f** command.

```
$> kubectl apply -f <filename>.yaml
```

### Creating a volume snapshot

Create a snapshot for a specific PersistentVolumeClaim (PVC), using the `template-snapshot.yaml`. For more information on PVC configuration, see [“Creating a PersistentVolumeClaim \(PVC\)”](#) on page 13.

```
apiVersion: snapshot.storage.k8s.io/v1beta1apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: <NAME>
spec:
  snapshotClassName: <SNAPSHOT_CLASS_NAME>
  source:
    persistentVolumeClaimName: <PVC_NAME>
```

After the yaml file is created, apply it by using the **kubectl apply -f** command.

```
$> kubectl apply -f <filename>.yaml
```

To verify that the VolumeSnapshot object is created, run the **kubectl describe volumesnapshot** command.

See the **Status** section of the output for the following:

- **Bound Volume Snapshot Content Name:** Indicates the volume is bound to the specified VolumeSnapshotContent.
- **Creation Time:** Indicates when the snapshot was created.
- **Ready to Use:** Indicates the volume snapshot is ready to use.
- **Restore Size:** Indicates the minimum volume size required when restoring a volume from this snapshot.

## Advanced configuration

Use advanced configuration tasks to further customize the configuration of the IBM block storage CSI driver.

- [“Importing an existing volume” on page 17](#)

### Importing an existing volume

Use this information to import volumes created externally from the IBM block storage CSI driver by using a persistent volume (PV) yaml file.

#### Before you begin

Before starting to import an existing volume, find the following information in the existing volume, in order to include the information in the persistent volume (PV) yaml file:

- **volumeHandle**
- **volumeAttributes** (optional)

Including:

- **pool\_name:** <Name of Pool where volume is located>  
(Listed as **pool\_id:** for DS8000 Family systems.)
- **storage\_type:** <SVC | A9K | DS8K>
- **volume\_name:** <Volume name>
- **array\_address:** <Array address>

To find the **volumeHandle**, use one of the following procedures:

- **Through command line (for Spectrum Virtualize Family):**

```
lsvdisk <volume name> | grep vdisk_UID
```

```
lsvdisk vol0 | grep vdisk_UID  
vdisk_UID 600507640082000B08000000000004FF
```

- **Through command line (for FlashSystem A9000 and A9000R):**

```
vol_list_extended vol=<volume_name>
```

For example, for vol1:

```
A9000>> vol_list_extended vol=vol1  
Name      WWN                      Product Serial Number  
vol1      6001738CFC9035E8000000000091F0C0  60035E80000000000091F0C0
```

- **Through the Spectrum Virtualize management GUI:**

1. Select **Volumes > Volumes** from the side bar.

The **Volumes** page appears.

2. Browse to the volume that the port is on and right-click > **Properties**.

The Properties window appears. Use the UID number.

For more information about Spectrum Virtualize see [IBM Spectrum Virtualize as Software Only on Knowledge Center \(ibm.com/support/knowledgecenter/en/STVLF4\)](https://www.ibm.com/support/knowledgecenter/en/STVLF4) or your specific Spectrum Virtualize Family product page on [Knowledge Center \(ibm.com/support/knowledgecenter/\)](https://www.ibm.com/support/knowledgecenter/).

- **Through the IBM Hyper-Scale Manager user interface for FlashSystem A9000 and A9000R storage systems:**

1. Select **Pools and Volumes Views > Volumes** from the side bar.

The **Volumes** table is displayed.

2. Select the Volume.

The **Volume Properties** form appears.

3. Use the **ID** number.

For more information about the IBM Hyper-Scale Manager see [IBM Hyper-Scale Manager on Knowledge Center \(ibm.com/support/knowledgecenter/SSUMNQ\)](https://www.ibm.com/support/knowledgecenter/SSUMNQ).

### About this task

Use this procedure to help build a PV yaml file for your volumes.

---

**Note:** These steps are setup for importing volumes from a Spectrum Virtualize Family system. Change parameters, as needed.

---

### Procedure

1. Create a persistent volume (PV) yaml file.

---

**Important:** Be sure to include the **storageClassName** and **controllerPublishSecretRef** parameters or errors will occur.

---

2. Take the **volume\_name** and other optional information (collected before the procedure) and insert it into the yaml file.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  #annotations:
  #pv.kubernetes.io/provisioned-by: block.csi.ibm.com
  name: vol1-pv
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  csi:
    controllerPublishSecretRef:
      name: svc-secret
      namespace: csi-ns
    driver: block.csi.ibm.com
    # volumeAttributes:
    #   # pool_name: ibmc-block-gold
    #   # storage_type: SVC
    #   # volume_name: vol1
    #   # array_address: baremetal10-cluster.xiv.ibm.com
    volumeHandle: SVC:600507640082000B08000000000004FF
    # persistentVolumeReclaimPolicy: Retain
    storageClassName: ibmc-block-gold
    # volumeMode: Filesystem
```

3. Create a PersistentVolumeClaim (PVC) yaml file.

---

**Note:**

- To include specific a 5Gi PV, be sure to include the **storageClassName**.
  - For more information about creating a PVC yaml file, see [“Creating a PersistentVolumeClaim \(PVC\)” on page 13](#).
- 

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  volume.beta.kubernetes.io/storage-provisioner: block.csi.ibm.com
  name: vol1-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ibmc-block-gold
  volumeName: vol1-pv
```

#### 4. Create a project namespace.

##### Using OpenShift web console

From Red Hat OpenShift Container Platform **Home > Projects**, click **Create Project**. In the **Create Project** dialog box, fill in a Project name (also referred to as *namespace*).

Click **Create** to save.

##### Using command line terminal

---

**Note:** This procedure is applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

---

Use the `kubectl create ns <namespace>` command to create a project namespace.

#### 5. Create a StatefulSet.

For more information regarding creating a StatefulSet, see [“Creating a StatefulSet” on page 14](#).

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: sanity-statefulset
spec:
  selector:
    matchLabels:
      app: sanity-statefulset
  serviceName: sanity-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: sanity-statefulset
    spec:
      containers:
        - name: container1
          image: registry.access.redhat.com/ubi8/ubi:latest
          command: [ "/bin/sh", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
          volumeMounts:
            - name: vol1
              mountPath: "/data"
      volumes:
        - name: vol1
          persistentVolumeClaim:
            claimName: vol1-pvc
```



---

## Chapter 4. Using IBM block storage CSI driver

Use this information for further usage information for the CSI (Container Storage Interface) driver.

- [“Sample configurations for running a stateful container” on page 21.](#)
- [“Recovering a pod volume attachment from a crashed Kubernetes node” on page 30.](#)

---

### Sample configurations for running a stateful container

You can use the CSI (Container Storage Interface) driver for running stateful containers with a storage volume provisioned from IBM block storage systems.

#### About this task

These examples illustrate a basic configuration required for running a stateful container with volumes provisioned on an IBM Spectrum Virtualize Family storage system.

While these examples specify the use of IBM Spectrum Virtualize products, the same configuration is used on all supported storage system types.

---

**Note:** The secret names given can be user specified. When giving secret names when managing different system storage types, be sure to give system type indicators to each name.

---

The following are examples of different types of secret names that can be given per storage type.

Storage system name	Secret name
IBM FlashSystem A9000 IBM FlashSystem A9000R	a9000-array1
IBM Spectrum Virtualize Family including IBM SAN Volume Controller and IBM FlashSystem family members built with IBM Spectrum Virtualize (FlashSystem 5010, 5030, 5100, 7200, 9100, 9200, 9200R)	storwize-array1
IBM DS8000 Family products	DS8000-array1

This example includes the following steps:

- Creating a secret for the storage system.
- Creating a storage class (gold).
- Creating a PersistentVolumeClaim (PVC) that uses the storage class gold and show some details on the created PVC and persistent volume (PV).
- Creating a StatefulSet and observing the mountpoint / multipath device that was created by the driver.
- Writing data inside the StatefulSet, and then deleting and recreating the StatefulSet, verifying that the data still exists.

---

**Note:** This procedure is applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubect1` with `oc` in all relevant commands.

---

Use the following sections for samples on running a stateful container on Statefulset volumes using file systems and raw block volumes:

- [“Running a stateful container with file system configurations” on page 22](#)

- [“Running a stateful container with raw block volume configurations” on page 26](#)

## Running a stateful container with file system configurations

Use this section as a sample of how to run a stateful container with a file system configuration.

### Before you begin

Before starting the procedure, be sure to see all information detailed in [“Running a stateful container with file system configurations” on page 22](#).

### Procedure

1. Open a command-line terminal.
2. Create an array secret.

---

**Important:** Be sure that the username and password match the same username and password used on the storage system.

---

```
$> cat demo-secret-svc-array.yaml
kind: Secret
apiVersion: v1
metadata:
  name: svc-array
  namespace: csi-ns
type: Opaque
stringData:
  management_address: <ADDRESS-1, ADDRESS-2> # Array management addresses
  username: <USERNAME> # Array username
data:
  password: <PASSWORD base64> # replace with valid password
$> kubectl create -f demo-secret-svc-array.yaml
secret/svc-array created
```

3. Create a storage class.

---

**Note:** The **SpaceEfficiency** values for Spectrum Virtualize Family are: thick, thin, compressed, or deduplicated. These values are not case specific.

For DS8000 Family systems, the default value is standard, but can be set to thin, if required. These values are not case specific. For more information, see [“Creating storage classes” on page 12](#).

This parameter is not applicable for IBM FlashSystem A9000 and A9000R systems. These systems always include deduplication and compression.

---

```
$> cat demo-storageclass-gold-svc.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: block.csi.ibm.com
parameters:
  SpaceEfficiency: deduplicated
  pool: gold

  csi.storage.k8s.io/provisioner-secret-name: svc-array
  csi.storage.k8s.io/provisioner-secret-namespace: csi-ns
  csi.storage.k8s.io/controller-publish-secret-name: svc-array
  csi.storage.k8s.io/controller-publish-secret-namespace: csi-ns

  csi.storage.k8s.io/fstype: xfs # Optional. values ext4\xfs. The default is ext4.
  volume_name_prefix: demo # Optional.
$> kubectl create -f demo-storageclass-gold-svc.yaml
storageclass.storage.k8s.io/gold created
```

4. Create a PVC demo-pvc-file-system.yaml with the size of 1 Gb.



---

**Note:** For more information about creating a PVC yaml file, see [“Creating a PersistentVolumeClaim \(PVC\)”](#) on page 13.

---

```
$> cat demo-pvc-file-system.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc-file-system
spec:
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gold

$> kubectl apply -f demo-pvc-file-system.yaml
persistentvolumeclaim/demo-pvc-file-system created
```

5. Display the existing PVC and the created persistent volume (PV).

---

**Note:** For more information about creating a PVC yaml file, see [“Creating a PersistentVolumeClaim \(PVC\)”](#) on page 13.

---

```
$> kubectl get pv,pvc
NAME                                     CAPACITY  ACCESS MODES
persistentvolume/pvc-828ce909-6eb2-11ea-abc8-005056a49b44  1Gi       RWO

RECLAIM POLICY  STATUS  CLAIM                                STORAGECLASS  REASON  AGE
Delete         Bound   default/demo-pvc-file-system        gold          109m

NAME                                     STATUS  VOLUME                                     CAPACITY
persistentvolumeclaim/demo-pvc-file-system  Bound   pvc-828ce909-6eb2-11ea-abc8-005056a49b44  1Gi

ACCESS MODES  STORAGECLASS  AGE
RWO           gold          78s

$> kubectl describe persistentvolume/pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Name:          pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: block.csi.ibm.com
Finalizers:    [kubernetes.io/pv-protection external-attacher/block-csi-ibm-com]
StorageClass:  gold
Status:        Bound
Claim:         default/demo-pvc-file-system
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:    Filesystem
Capacity:      1Gi
Node Affinity: <none>
Message:
Source:
  Type:        CSI (a Container Storage Interface (CSI) volume source)
  Driver:      block.csi.ibm.com
  VolumeHandle: SVC:600507607181069980000000000000543
  ReadOnly:    false
  VolumeAttributes:
    array_address=baremetal10-cluster.xiv.ibm.com
    pool_name=csi_svcPool
    storage.kubernetes.io/csiProvisionerIdentity=1585146948772-8081-
block.csi.ibm.com
    storage_type=SVC
    volume_name=demo_pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Events:        <none>
```

6. Create a StatefulSet, using the demo-statefulset-file-system.yaml.

---

**Note:** For more information about creating a StatefulSet, see [“Creating a StatefulSet”](#) on page 14.

---

```
$> kubectl create -f demo-statefulset-file-system.yaml
statefulset/demo-statefulset-file-system created
```

```
$> cat demo-statefulset-file-system.yaml
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset-file-system
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-statefulset
    spec:
      containers:
        - name: container-demo
          image: registry.access.redhat.com/ubi8/ubi:latest
          command: [ "/bin/sh", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
          volumeMounts:
            - name: demo-volume
              mountPath: "/data"
      volumes:
        - name: demo-volume
          persistentVolumeClaim:
            claimName: demo-pvc-file-system

# nodeSelector:
#   kubernetes.io/hostname: HOSTNAME
```

7. Check the newly created pod.

Display the newly created pod (make sure the pod status is *Running*).

```
$> kubectl get pod demo-statefulset-file-system-0
NAME                                READY   STATUS    RESTARTS   AGE
demo-statefulset-file-system-0      1/1     Running   0           43s
```

8. Write data to the persistent volume of the pod.

The PV should be mounted inside the pod at `/data`.

```
$> kubectl exec demo-statefulset-0 touch /data/FILE
$> kubectl exec demo-statefulset-0 ls /data/FILE
/data/FILE
```

9. Log into the worker node that has the running pod and display the newly attached volume on the node.

a) Verify which worker node is running the pod *demo-statefulset-0*.

```
$> kubectl describe pod demo-statefulset-0 | grep "^Node:"
Node: k8s-node1/hostname
```

b) Establish an SSH connection and log into the worker node.

```
$> ssh root@k8s-node1
```

c) List the multipath devices on the worker node.

```
$>[k8s-node1] multipath -ll
mpathz (828ce9096eb211eaabc8005056a49b44) dm-3 IBM ,2145 (for SVC)
size=1.0G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
   |- 37:0:0:12 sdc 8:32 active ready running
   |- 36:0:0:12 sdb 8:16 active ready running

$>[k8s-node1] ls -l /dev/mapper/mpathz
lrwxrwxrwx. 1 root root 7 Aug 12 19:29 /dev/mapper/mpathz -> ../dm-3
```

d) List the physical devices of the multipath **mpathz** and its mountpoint on the host. (This is the `/data` inside the stateful pod).

```
$>[k8s-node1] lsblk /dev/sdb /dev/sdc
NAME        MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sdb          8:16   0   1G  0 disk
└─mpathz 253:3   0   1G  0 mpath /var/lib/kubelet/pods/d67d22b8-bd10-11e9-
a1f5-005056a45d5f/volumes/kubernetes.io~csi/pvc-828ce909-6eb2-11ea-abc8-005056a49b44
sdc          8:32   0   1G  0 disk
└─mpathz 253:3   0   1G  0 mpath /var/lib/kubelet/pods/d67d22b8-bd10-11e9-
a1f5-005056a45d5f/volumes/kubernetes.io~csi/pvc-828ce909-6eb2-11ea-abc8-005056a49b44
```

e) View the PV mounted on this host.

---

**Note:** All PV mountpoints look like: `/var/lib/kubelet/pods/*/volumes/kubernetes.io~csi/pvc-*/mount`

---

```
$>[k8s-node1] df | egrep pvc
/dev/mapper/mpathz 1038336 32944 1005392 4% /var/lib/kubelet/pods/
d67d22b8-bd10-11e9-a1f5-005056a45d5f/volumes/kubernetes.io~csi/
pvc-828ce909-6eb2-11ea-abc8-005056a49b44/mount
```

f) Details about the driver internal metadata file `.stageInfo.json` is stored in the k8s PV node stage path `/var/lib/kubelet/plugins/kubernetes.io/csi/pv/<PVC-ID>/globalmount/.stageInfo.json`. The CSI driver creates the metadata file during the `NodeStage` API and is used at later stages by the **NodePublishVolume**, **NodeUnPublishVolume** and **NodeUnStage** CSI APIs later on.

```
$> cat /var/lib/kubelet/plugins/kubernetes.io/csi/pv/
pvc-828ce909-6eb2-11ea-abc8-005056a49b44/globalmount/.stageInfo.json
{"connectivity":"iscsi","mpathDevice":"dm-3","sysDevices":["sdb,sdc"]}
```

10. Delete StatefulSet and then recreate, in order to validate data (`/data/FILE`) remains in the persistent volume.

a) Delete the StatefulSet.

```
$> kubectl delete statefulset/demo-statefulset-file-system
statefulset/demo-statefulset-file-system deleted
```

b) Wait until the pod is deleted. Once deleted, the `"demo-statefulset-file-system" not found` is returned.

```
$> kubectl get statefulset/demo-statefulset-file-system
NAME                                READY STATUS RESTARTS AGE
demo-statefulset-file-system-0      0/1    Terminating 0      91m
```

c) Verify that the multipath was deleted and that the PV mountpoint no longer exists by establishing an SSH connection and logging into the worker node.

```
$> ssh root@k8s-node1

$>[k8s-node1] df | egrep pvc
$>[k8s-node1] multipath -ll
$>[k8s-node1] lsblk /dev/sdb /dev/sdc
lsblk: /dev/sdb: not a block device
lsblk: /dev/sdc: not a block device
```

d) Recreate the StatefulSet and verify that `/data/FILE` exists.

```
$> kubectl create -f demo-statefulset-file-system.yaml
statefulset/demo-statefulset-file-system created

$> kubectl exec demo-statefulset-file-system-0 ls /data/FILE
File
```

11. Delete StatefulSet and the PVC.

```
$> kubectl delete statefulset/demo-statefulset-file-system
statefulset/demo-statefulset-file-system deleted

$> kubectl get statefulset/demo-statefulset-file-system
No resources found.

$> kubectl delete pvc/demo-pvc-file-system
persistentvolumeclaim/demo-pvc-file-system deleted

$> kubectl get pv,pvc
No resources found.
```

## Running a stateful container with raw block volume configurations

Use this section as a sample of how to run a stateful container with a raw block volume configuration.

### Before you begin

Before starting the procedure, be sure to see all information detailed in [“Running a stateful container with file system configurations”](#) on page 22.

### Procedure

1. Open a command-line terminal.
2. Create an array secret.

---

**Important:** Be sure that the username and password match the same username and password used on the storage system.

---

```
$> cat demo-secret-svc-array.yaml
kind: Secret
apiVersion: v1
metadata:
  name: svc-array
  namespace: csi-ns
type: Opaque
stringData:
  management_address: <ADDRESS-1, ADDRESS-2> # Array management addresses
  username: <USERNAME> # Array username
data:
  password: <PASSWORD base64> # replace with valid password

$> kubectl create -f demo-secret-svc-array.yaml
secret/svc-array created
```

3. Create a storage class.

---

**Note:** The **SpaceEfficiency** values for Spectrum Virtualize Family are: thick, thin, compressed, or deduplicated. These values are not case specific.

For DS8000 Family systems, the default value is standard, but can be set to thin, if required. These values are not case specific. For more information, see [“Creating storage classes”](#) on page 12.

This parameter is not applicable for IBM FlashSystem A9000 and A9000R systems. These systems always include deduplication and compression.

---

```
$> cat demo-storageclass-gold-svc.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: block.csi.ibm.com
parameters:
  SpaceEfficiency: deduplicated
  pool: gold

  csi.storage.k8s.io/provisioner-secret-name: svc-array
  csi.storage.k8s.io/provisioner-secret-namespace: csi-ns
  csi.storage.k8s.io/controller-publish-secret-name: svc-array
  csi.storage.k8s.io/controller-publish-secret-namespace: csi-ns

  csi.storage.k8s.io/fstype: xfs      # Optional. values ext4\xfs. The default is ext4.
  volume_name_prefix: demo           # Optional.

$> kubectl create -f demo-storageclass-gold-svc.yaml
storageclass.storage.k8s.io/gold created
```

4. Create a PVC `demo-pvc-raw-block.yaml` with the size of 1 Gb.

---

**Note:** For more information about creating a PVC yaml file, see [“Creating a PersistentVolumeClaim \(PVC\)”](#) on page 13.

---

```
https://github.com/IBM/ibm-block-csi-driver/blob/develop/deploy/kubernetes/examples/demo-
pvc-raw-block.yaml$> cat demo-pvc-raw-block.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc-raw-block
spec:
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gold

$> kubectl apply -f demo-pvc-raw-block.yaml
persistentvolumeclaim/demo-pvc-raw-block created
```

5. Display the existing PVC and the created persistent volume (PV).

---

**Note:** For more information about creating a PVC yaml file, see [“Creating a PersistentVolumeClaim \(PVC\)”](#) on page 13.

---

```
$> kubectl get pv,pvc
NAME                                     CAPACITY  ACCESS MODES
persistentvolume/pvc-828ce909-6eb2-11ea-abc8-005056a49b44  1Gi       RWO

RECLAIM POLICY  STATUS  CLAIM                                STORAGECLASS  REASON  AGE
Delete          Bound   default/demo-pvc-raw-block          gold                                109m

NAME                                     STATUS  VOLUME                                     CAPACITY
persistentvolumeclaim/demo-pvc-raw-block  Bound   pvc-828ce909-6eb2-11ea-abc8-005056a49b44  1Gi

ACCESS MODES  STORAGECLASS  AGE
RWO           gold          78s

kubectl describe persistentvolume/pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Name:          pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: block.csi.ibm.com
Finalizers:    [kubernetes.io/pv-protection external-attacher/block-csi-ibm-com]
StorageClass:  gold
Status:        Bound
Claim:         default/demo-pvc-raw-block
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:    Block
Capacity:      1Gi
Node Affinity: <none>
Message:
Source:
  Type:          CSI (a Container Storage Interface (CSI) volume source)
  Driver:        block.csi.ibm.com
  VolumeHandle:  SVC:600507607181069980000000000000543
  ReadOnly:      false
  VolumeAttributes:  array_address=baremetal10-cluster.xiv.ibm.com
                    pool_name=csi_svcPool
                    storage.kubernetes.io/csiProvisionerIdentity=1585146948772-8081-
block.csi.ibm.com
                    storage_type=SVC
                    volume_name=demo_pvc-828ce909-6eb2-11ea-abc8-005056a49b44
Events:          <none>
```

6. Create a StatefulSet, using the demo-statefulset-raw-block.yaml.

---

**Note:** For more information about creating a StatefulSet, see [“Creating a StatefulSet” on page 14.](#)

---

```
$> kubectl create -f demo-statefulset-raw-block.yaml
statefulset/demo-statefulset-raw-block created
```

```
$> cat demo-statefulset-raw-block.yaml

kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset-raw-block
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-statefulset
    spec:
      containers:
      - name: container-demo
        image: registry.access.redhat.com/ubi8/ubi:latest
        command: [ "/bin/sh", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        volumeDevices:
          - name: demo-volume
            devicePath: "/dev/block"
      volumes:
      - name: demo-volume
        persistentVolumeClaim:
          claimName: demo-pvc-raw-block

#   nodeSelector:
#     kubernetes.io/hostname: HOSTNAME
```

## 7. Check the newly created pod.

Display the newly created pod (make sure the pod status is *Running*).

```
$> kubectl get pod demo-statefulset-raw-block-0
NAME                READY    STATUS    RESTARTS   AGE
demo-statefulset-raw-block-0  1/1     Running   0          43s
```

## 8. Write data to the persistent volume of the pod.

The PV should be mounted inside the pod at `/dev`.

```
$> kubectl exec podraw67b -- bash -c " echo \"test_block\" | dd conv=unblock of=/dev/block"
0+1 records in
0+1 records out
11 bytes copied, 9.3576e-05 s, 118 kB/s
$> kubectl exec podraw67b -- bash -c "od -An -c -N 10 /dev/block"
t e s t _ b l o c k
```

## 9. Delete StatefulSet and then recreate, in order to validate data (test\_block in /dev/block) remains in the persistent volume.

### a) Delete the StatefulSet.

```
$> kubectl delete statefulset/demo-statefulset-raw-block
statefulset/demo-statefulset-raw-block deleted
```

### b) Wait until the pod is deleted. Once deleted the "'demo-statefulset-raw-block" not found' is returned.

```
$> kubectl get statefulset/demo-statefulset-raw-block
Error from server (NotFound): statefulsets.apps <StatefulSet name> not found
```

### c) Recreate the StatefulSet and verify that the content written to `/dev/block` exists.

```
$> kubectl create -f demo-statefulset-raw-block.yml
statefulset/demo-statefulset-raw-block created

$> kubectl exec podraw67b -- bash -c "od -An -c -N 10 /dev/block"
t e s t _ b l o c k
```

## 10. Delete StatefulSet and the PVC.

```
$> kubectl delete statefulset/demo-statefulset-raw-block
statefulset/demo-statefulset-raw-block deleted

$> kubectl get statefulset/demo-statefulset-raw-block
No resources found.

$> kubectl delete pvc/demo-pvc-raw-block
persistentvolumeclaim/demo-pvc-raw-block deleted

$> kubectl get pv,pvc
No resources found.
```

## Recovering a pod volume attachment from a crashed Kubernetes node

This section details a manual operation required to revive Kubernetes pods that reside on a crashed node due to an existing Kubernetes limitation.

### Identifying a crashed node

**Note:** These procedures are applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

When a worker node shuts down or crashes, all pods in a StatefulSet that reside on it become unavailable. In these scenarios, the node status is *NotReady*, and the pod status appears as *Terminating*.

For example:

```
$> kubectl get nodes
NAME STATUS ROLES AGE VERSION
k8s-master Ready master 6d <your k8s version>
k8s-node1 Ready <none> 6d <your k8s version>
k8s-node3 NotReady <none> 6d <your k8s version>

$> kubectl get pods --all-namespaces -o wide | grep default
default sanity-statefulset-0 1/1 Terminating 0 19m 10.244.2.37 k8s-node3
```

### Recovering a crashed node



**Attention:** In order to avoid data loss, before continuing, verify that there are no pods connected to this volume.

Follow the following procedure to recover from a crashed node (see a full example at the end of the procedure):

1. Find for the **volumeattachment** of the created pod:

```
$> kubectl get volumeattachment
```

2. Copy the **volumeattachment** name.

3. Delete the **volumeattachment**:

```
$> kubectl delete volumeattachment <volumeattachment name>
```

4. Delete the pod:

```
$> kubectl delete pod <pod name> --grace-period=0 --force
```

5. Verify that the pod is now in a *Running* state and that the pod has moved to worker-node1.

For example:



```

$> kubectl get nodes
NAME STATUS ROLES AGE VERSION
k8s-master Ready master 6d <your k8s version>
k8s-node1 Ready <none> 6d <your k8s version>
k8s-node3 NotReady <none> 6d <your k8s version>

$> kubectl get pods --all-namespaces -o wide | grep default
default sanity-statefulset-0 1/1 Terminating 0 19m 10.244.2.37 k8s-node3

$> kubectl get volumeattachment
NAME AGE
csi-5944e1c742d25e7858a8e48311cdc6cc85218f1156dd6598d4cf824fb1412143 10m

$> kubectl delete volumeattachment
csi-5944e1c742d25e7858a8e48311cdc6cc85218f1156dd6598d4cf824fb1412143
volumeattachment.storage.k8s.io
"csi-5944e1c742d25e7858a8e48311cdc6cc85218f1156dd6598d4cf824fb1412143" deleted

$> kubectl delete pod sanity-statefulset-0 --grace-period=0 --force
warning: Immediate deletion does not wait for confirmation that the running resource has been
terminated. The resource may continue to run on the cluster indefinitely.
pod "sanity-statefulset-0" deleted

$> kubectl get pods --all-namespaces -o wide | grep default
default sanity-statefulset-0 1/1 Running 0 26s 10.244.1.210 k8s-node1

```



---

## Chapter 5. Troubleshooting

This section can help you detect and solve problems that you might encounter when using the IBM block storage CSI driver.

### Log collection

---

Use the CSI (Container Storage Interface) driver logs for problem identification.

---

**Note:** These procedures are applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

---

To collect and display logs, related to the different components of IBM block storage CSI driver, use the following Kubernetes commands:

#### Log collection for CSI pods, daemonset, and StatefulSet

```
$> kubectl get all -n <namespace> -l csi
```

For example:

```
$> kubectl get all -n <namespace> -l csi
NAME READY STATUS RESTARTS AGE
pod/ibm-block-csi-controller-0 4/4 Running 0 2h
pod/ibm-block-csi-node-nbtsg 3/3 Running 0 2h
pod/ibm-block-csi-node-wd5tm 3/3 Running 0 2h
pod/ibm-block-csi-operator-7684549698-hzmfh 1/1 Running 0 2h

NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
daemonset.apps/ibm-block-csi-node 2 2 2 2 2 <none> 2h

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
deployment.apps/ibm-block-csi-operator 1 1 1 1 2h

NAME DESIRED CURRENT READY AGE
replicaset.apps/ibm-block-csi-operator-7684549698 1 1 1 2h

NAME DESIRED CURRENT AGE
statefulset.apps/ibm-block-csi-controller 1 1 2h
```

#### Log collection for IBM block storage CSI driver controller

```
$> kubectl log -f -n <namespace> ibm-block-csi-controller-0 -c ibm-block-csi-controller
```

#### Log collection for IBM block storage CSI driver node (per worker node or PODID)

```
$> kubectl log -f -n <namespace> ibm-block-csi-node-<PODID> -c ibm-block-csi-node
```

#### Log collection for Operator for IBM block storage CSI driver

```
$> kubectl log -f -n <namespace> ibm-block-csi-operator-<PODID> -c ibm-block-csi-operator
```

---

### Detecting errors

Use this information to help pinpoint potential causes for stateful pod failure.

This is an overview of actions that you can take to pinpoint a potential cause for a stateful pod failure.

---

**Note:** This procedures is applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

---

1. Verify that the CSI driver is running. (Make sure the `csi-controller` pod status is *Running*).

```
$> kubectl get all -n <namespace> -l csi
NAME READY STATUS RESTARTS AGE
pod/ibm-block-csi-controller-0 4/4 Running 0 2h
pod/ibm-block-csi-node-nbtsg 3/3 Running 0 2h
pod/ibm-block-csi-node-wd5tm 3/3 Running 0 2h
pod/ibm-block-csi-operator-7684549698-hzmfh 1/1 Running 0 2h

NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
daemonset.apps/ibm-block-csi-node 2 2 2 2 2 <none> 2h

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
deployment.apps/ibm-block-csi-operator 1 1 1 1 2h

NAME DESIRED CURRENT READY AGE
replicaset.apps/ibm-block-csi-operator-7684549698 1 1 1 2h

NAME DESIRED CURRENT AGE
statefulset.apps/ibm-block-csi-controller 1 1 2h
```

2. If `pod/ibm-block-csi-controller-0` is not in a *Running* state, run the following command:

```
$> kubectl describe -n <namespace> pod/ibm-block-csi-controller-0
```

View the logs (see [“Log collection”](#) on page 33).

## Multipath troubleshooting

Use this information to help pinpoint potential causes for multipath failures.

### Display multipath information (FC and iSCSI)

Display multipath information, using the **sudo multipath -ll** command.

```
mpathb (36005076802838515300000000000000a6) dm-0 IBM,2145
size=1.0G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
|+- policy='service-time 0' prio=50 status=active
|  '- 3:0:0:0 sda 8:0 active ready running
|+- policy='service-time 0' prio=10 status=enabled
|  '- 2:0:0:0 sdb 8:16 active ready running
```

### Display device attachment

Display device attachment information, using the **sudo lsblk** command.

```
NAME        MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda          8:0    0  1G  0 disk
`-mpathb 253:0    0  1G  0 mpath /var/lib/kubelet/pods/c9fee230-6227-11ea-a0b6-52fdcf072182/
volumes/kubernetes.io~csi/pvc-32a7e21b-6227-11ea-a0b6-52fdcf
sdb          8:16    0  1G  0 disk
`-mpathb 253:0    0  1G  0 mpath /var/lib/kubelet/pods/c9fee230-6227-11ea-a0b6-52fdcf072182/
volumes/kubernetes.io~csi/pvc-32a7e21b-6227-11ea-a0b6-52fdcf
vda         252:0    0 31G  0 disk
|-vda1      252:1    0  1M  0 part
|-vda2      252:2    0  1G  0 part /boot
`-vda3      252:3    0 30G  0 part /sysroot
```

To display device attachment information, together with SCSI ID information, use the **sudo lsblk -S** command.

```
NAME HCTL          TYPE VENDOR  MODEL          REV TRAN
sda  3:0:0:0         disk IBM     2145           0000 iscsi
sdb  2:0:0:0         disk IBM     2145           0000 iscsi
```

### Check for multipath daemon availability (FC and iSCSI)

Check for multipath daemon availability, using the **systemctl status multipathd** command.

```
multipathd.service - Device-Mapper Multipath Device Controller
  Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-03-09 16:28:37 UTC; 22min ago
  Main PID: 1235 (multipathd)
  Status: "up"
  Tasks: 7
  Memory: 14.1M
  CPU: 131ms
  CGroup: /system.slice/multipathd.service
          └─1235 /sbin/multipathd -d -s
```

### Check for iSCSI daemon availability

Check for iSCSI daemon availability, using the **systemctl status iscsid** command.

```
iscsid.service - Open-iSCSI
  Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2020-03-09 16:28:37 UTC; 22min ago
  Docs: man:iscsid(8)
        man:iscsiadm(8)
  Main PID: 1440 (iscsid)
  Status: "Ready to process requests"
  Tasks: 1 (limit: 26213)
  Memory: 4.7M
  CPU: 27ms
  CGroup: /system.slice/iscsid.service
          └─1440 /usr/sbin/iscsid -f
```

## Miscellaneous troubleshooting

Use this information to help pinpoint potential causes for stateful pod failure.

**Note:** These procedures are applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

- [“General troubleshooting” on page 35](#)
- [“Error during pod creation” on page 36](#) (for volumes using StatefulSet only)
- [“Error during automatic iSCSI login” on page 36](#)

### General troubleshooting

Use the following command for general troubleshooting:

```
$> kubectl get -n <namespace> csidriver,sa,clusterrole,clusterrolebinding,statefulset,pod,daemonset |
grep ibm-block-csi
```

For example:

```
$> kubectl get -n csi-ns csidriver,sa,clusterrole,clusterrolebinding,statefulset,pod,daemonset |
grep ibm-block-csi
csidriver.storage.k8s.io/ibm-block-csi-driver 7d

serviceaccount/ibm-block-csi-controller-sa 1 2h
serviceaccount/ibm-block-csi-node-sa 1 2h
serviceaccount/ibm-block-csi-operator 1 2h

clusterrole.rbac.authorization.k8s.io/ibm-block-csi-external-attacher-clusterrole 2h
clusterrole.rbac.authorization.k8s.io/ibm-block-csi-external-provisioner-clusterrole 2h
clusterrole.rbac.authorization.k8s.io/ibm-block-csi-operator 2h

clusterrolebinding.rbac.authorization.k8s.io/ibm-block-csi-external-attacher-clusterrolebinding 2h
clusterrolebinding.rbac.authorization.k8s.io/ibm-block-csi-external-provisioner-clusterrolebinding 2h
clusterrolebinding.rbac.authorization.k8s.io/ibm-block-csi-operator 2h

statefulset.apps/ibm-block-csi-controller 1 1 2h
pod/ibm-block-csi-controller-0 4/4 Running 0 2h
pod/ibm-block-csi-node-nbtsg 3/3 Running 0 2h
pod/ibm-block-csi-node-wd5tm 3/3 Running 0 2h
pod/ibm-block-csi-operator-7684549698-hzmfh 1/1 Running 0 2h

daemonset.extensions/ibm-block-csi-node 2 2 2 2 2 <none> 2h
```

## Error during pod creation

---

**Note:** This troubleshooting procedure is relevant for volumes using file system types only (not for volumes using raw block volume types).

---

If the following error occurs during stateful application pod creation (the pod status is *ContainerCreating*):

```
-8e73-005056a49b44" : rpc error: code = Internal desc = 'fsck' found errors on device /dev/
dm-26 but could not correct them: fsck from util-linux 2.23.2
/dev/mapper/mpathym: One or more block group descriptor checksums are invalid. FIXED.
/dev/mapper/mpathym: Group descriptor 0 checksum is 0x0000, should be 0x3baa.

/dev/mapper/mpathym: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
(i.e., without -a or -p options)
```

1. Log in to the relevant worker node and run the **fsck** command to repair the filesystem manually.

```
fsck /dev/dm-<X>
```

The pod should come up immediately. If the pod is still in a *ContainerCreating* state, continue to the next step.

2. Run the **# multipath -ll** command to see if there are faulty multipath devices.

If there are faulty multipath devices:

- a. Restart multipath daemon, using the **systemctl restart multipathd** command.
- b. Rescan any iSCSI devices, using the **rescan-scsi-bus.sh** command.
- c. Restart the multipath daemon again, using the **systemctl restart multipathd** command.

The multipath devices should be running properly and the pod should come up immediately.

## Error during automatic iSCSI login

If an error during automatic iSCSI login occurs, perform the following steps for manual login:

---

**Note:** These procedures are applicable for both Kubernetes and Red Hat OpenShift. For Red Hat OpenShift, replace `kubectl` with `oc` in all relevant commands.

---

---

**Note:** This procedure is applicable for both RHEL and RHCOS users. When using RHCOS, use the following:

- Log into the RHCOS node with the `core` user (for example, `ssh core@worker1.apps.openshift.mycluster.net`)
- **iscsiadm** commands must start with `sudo`

- 
1. Verify that the **node.startup** in the `/etc/iscsi/iscsid.conf` file is set to *automatic*. If not, set it as required and then restart the `iscsid` service (**\$ service iscsid restart**).
  2. Discover and log into at least two iSCSI targets on the relevant storage systems.

---

**Note:** A multipath device can't be created without at least two ports.

---

```
$> iscsiadm -m discoverydb -t st -p ${STORAGE-SYSTEM-iscsi-PORT-IP1}:3260 --discover
$> iscsiadm -m node -p ${STORAGE-SYSTEM-iscsi-PORT-IP1} --login

$> iscsiadm -m discoverydb -t st -p ${STORAGE-SYSTEM-iscsi-PORT-IP2}:3260 --discover
$> iscsiadm -m node -p ${STORAGE-SYSTEM-iscsi-PORT-IP2} --login
```

3. Verify that the login was successful and display all targets that you logged into. The *portal* value must be the iSCSI target IP address.

```
$> iscsiadm -m session -rescan
Rescanning session [sid: 1, target: {storage system IQN},
portal: {storage system iSCSI port IP},{port number}]
```





## Notices

---

These legal notices pertain to the information in this IBM Storage product documentation.

This information was developed for products and services offered in the US. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
USA*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119*

Armonk, NY 10504-1785  
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

## Trademarks

---

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of the International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the [Copyright and trademark information website](http://www.ibm.com/legal/us/en/copytrade.shtml) ([www.ibm.com/legal/us/en/copytrade.shtml](http://www.ibm.com/legal/us/en/copytrade.shtml)).

VMware, the VMware logo, ESX, ESXi, vSphere, vCenter, and vCloud are trademarks or registered trademarks of VMware Corporation in the United States, other countries, or both.

Microsoft, Windows Server, Windows, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.





Printed in USA

SC27-9590-04

