

IBM block storage CSI driver
Version 1.0.0

User Guide



Note

Before using this document and the product it supports, read the information in [“Notices” on page 29.](#)

Edition notice

Publication number: SC27-9590-00. This publication applies to version 1.0.0 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 2019.**

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

Contents

Figures.....	V
About this guide.....	vii
Who should use this guide.....	vii
Conventions used in this guide.....	vii
Related information and publications.....	vii
Getting information, help, and service.....	vii
IBM Publications Center.....	viii
Feedback.....	viii
Chapter 1. Introduction.....	1
Chapter 2. Installation.....	3
Compatibility and requirements.....	3
Installing the operator and driver.....	4
Installing from the OpenShift web console.....	4
Installing from GitHub through command line terminal.....	6
Uninstalling.....	7
Uninstalling from the OpenShift web console.....	7
Uninstalling from a command line terminal.....	8
Chapter 3. CSI driver configuration.....	9
Creating an array secret.....	9
Creating storage classes.....	9
Advanced configuration.....	10
Importing an existing volume.....	10
Chapter 4. Using IBM block storage CSI driver.....	13
Sample configurations for running a stateful container.....	13
Running a stateful container on IBM FlashSystem A9000R.....	13
Running a stateful container on Storwize systems.....	17
Recovering a crashed Kubernetes node.....	22
Chapter 5. Troubleshooting.....	25
Notices.....	29
Trademarks.....	30

Figures

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

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 Storwize® V5000 on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STHGJ\)](https://ibm.com/support/knowledgecenter/STHGJ)
- [IBM 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 Spectrum Accelerate on IBM Knowledge Center \(ibm.com/support/knowledgecenter/STZSWD\)](https://ibm.com/support/knowledgecenter/STZSWD)
- [IBM FlashSystem® 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)
- [Persistent volumes on Kubernetes \(kubernetes.io/docs/concepts/storage/volumes\)](https://kubernetes.io/docs/concepts/storage/volumes)
- [IBM Cloud Private \(ibm.com/cloud/private\)](https://ibm.com/cloud/private)

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.

- [IBM website \(ibm.com®\)](https://ibm.com)

- [IBM Support Portal website \(ibm.com/support/entry/portal/support?brandind=Hardware~System_Storage\)](http://ibm.com/support/entry/portal/support?brandind=Hardware~System_Storage)
- [IBM Directory of Worldwide Contacts website \(ibm.com/planetwide\)](http://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 \(ibm.com/shop/publications/order\)](http://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 ibmkc@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](https://ibm.com/support/knowledgecenter/STXKQY) (ibm.com/support/knowledgecenter/STXKQY).

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

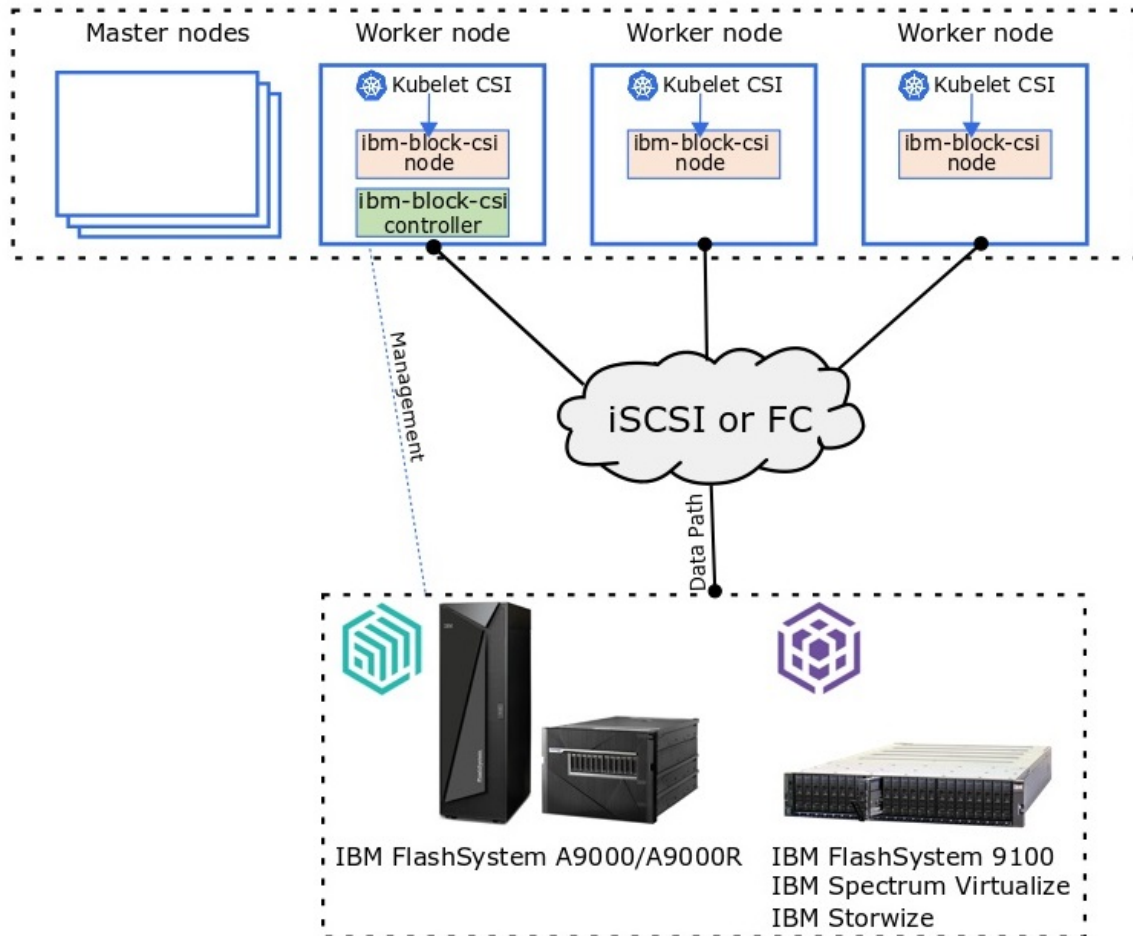


Figure 1. Integration of IBM block storage systems and CSI driver in a Kubernetes environment

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 4](#)

For information about uninstallation, see [“Uninstalling” on page 7](#).

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](http://ibm.com/support/knowledgecenter/SSRQ8T) (ibm.com/support/knowledgecenter/SSRQ8T).

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.

Note: Currently only the Red Hat Enterprise Linux (RHEL) 7.x operating system is supported.

Procedure

1. Install the following RHEL 7.x packages to ensure Fibre Channel and iSCSI connectivity. Skip this step if the packages are already installed.
 - `iscsi-initiator-utils` (if iSCSI connection is required)
 - `xfsprogs` (if a Fibre Channel, XFS file system, is required)

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

2. Configure Linux multipath devices on the host.

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
```

Important: When configuring Linux multipath devices, verify that the **`find_multipaths`** parameter in the `multipath.conf` file is disabled, by removing the **`find_multipaths yes`** string from the file.

3. Configure storage system connectivity.

- a) Define the hostname of each Kubernetes node on the relevant storage systems with the valid WWPN (for Fibre Channel) or IQN (for iSCSI) of the node.

```
root@k8s-user-v18-master:~# kubectl get nodes
NAME                STATUS    ROLES    AGE      VERSION
k8s-master          Ready    master   34d      v1.14.1
k8s-worker-node1    Ready    <none>   34d      v1.14.1
k8s-worker-node2    Ready    <none>   34d      v1.14.1
```

- b) For Fibre Channel, configure the relevant zoning from the storage to the host.
- c) For iSCSI, perform the following steps:
- 1) Make sure that the login to the iSCSI targets is permanent and remains available after a reboot of the worker node. To do this, 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}]
```

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 from the OpenShift web console”](#) on page 4.
- If Red Hat OpenShift Container Platform is not being used, follow the instructions detailed in [“Installing from GitHub through command line terminal”](#) on page 6.

Installing from 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

Important: The operator must be installed in the **kube-system** project/namespace.

Note: If using you are not using the Red Hat OpenShift Container Platform, use the instructions detailed in [“Installing from GitHub through command line terminal”](#) on page 6.

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 **Operators** > **OperatorHub**, select **Projects: kube-system**.
2. Search for IBM block storage CSI driver.
3. Select the **Operator for IBM block storage CSI driver** and click **Install**.
The Operator Subscription form appears.
4. Set the **Installation Mode** to **kube-system**, under **A specific namespace on the cluster**.
5. Click **Subscribe**.
6. 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**.

7. Once the operator installation progress has completed, click on the installed Operator for IBM block storage CSI driver.
8. Click **Create Instance** to create the IBM block storage CSI driver (IBMBlockCSI).
9. **(Optional:)** Edit the yaml file in the web console as follows:

```

apiVersion: csi.ibm.com/v1
kind: IBMBlockCSI
metadata:
  name: ibm-block-csi
spec:
  controller:
    repository: ibmcom/ibm-block-csi-driver-controller
    tag: 1.0.0
    imagePullPolicy: IfNotPresent
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
            - matchExpressions:
                - key: kubernetes.io/arch
                  operator: In
                  values:
                    - amd64
  node:
    repository: ibmcom/ibm-block-csi-driver-node
    tag: 1.0.0
    imagePullPolicy: IfNotPresent
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
            - matchExpressions:
                - key: kubernetes.io/arch
                  operator: In
                  values:
                    - amd64
  sidecars:
    - name: csi-node-driver-registrar
      repository: quay.io/k8scsi/csi-node-driver-registrar
      tag: v1.2.0
      imagePullPolicy: IfNotPresent
    - name: csi-provisioner
      repository: quay.io/k8scsi/csi-provisioner
      tag: v1.3.0
      imagePullPolicy: IfNotPresent
    - name: csi-attacher
      repository: quay.io/k8scsi/csi-attacher
      tag: v1.2.1
      imagePullPolicy: IfNotPresent
    - name: livenessprobe
      repository: quay.io/k8scsi/livenessprobe
      tag: v1.1.0
      imagePullPolicy: IfNotPresent

```

10. Click **Create**.

Wait until the **Status** is *Running*.

```

Status:
  Controller Ready:  true
  Node Ready:       true
  Phase:            Running
  Version:          1.0.0
  Events:           <none>

```

Installing from GitHub through command line terminal

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: If using the Red Hat OpenShift Container Platform, use the instructions detailed in [“Installing from the OpenShift web console”](#) on page 4.

Procedure

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

1. Download the manifest from GitHub.

```
curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/master/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`.
3. Install the operator.

```
$ kubectl -n kube-system apply -f ibm-block-csi-operator.yaml
```

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

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

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

- a) Download the manifest from GitHub.

```
curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/master/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`.
- c) Install the `csi.ibm.com_v1_ibmblockcsi_cr.yaml`.

```
$ kubectl -n kube-system 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 from the OpenShift web console”](#) on page 7.
- If Red Hat OpenShift Container Platform is not being used, follow the instructions detailed in [“Uninstalling from a command line terminal”](#) on page 8.

Uninstalling from 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: If you are not using the Red Hat OpenShift Container Platform web console, use the instructions detailed in [“Installing from GitHub through command line terminal”](#) on page 6.

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** and select **Projects: kube-system**.
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 from a command line terminal

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

About this task

Note: If you are using the Red Hat OpenShift Container Platform, use the instructions detailed in [“Uninstalling from the OpenShift web console” on page 7](#).

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
```

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 4), 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 9
- [“Creating storage classes”](#) on page 9
- [“Advanced configuration”](#) on page 10

Creating an array secret

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

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

Creating a array secret file

```
kind: Secret
apiVersion: v1
metadata:
  name: <VALUE-1>
  namespace: kube-system
type: Opaque
stringData:
  management_address: <VALUE-2,VALUE-3> # replace with valid storage system managment address
  username: <VALUE-4> # replace with valid username
data:
  password: <VALUE-5 base64> # replace with valid password
```

Creating an array secret via command line

1. 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 kube-system
```

2. Apply the secret using the following command:

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

The secret/<name> created message is emitted.

Creating storage classes

Create k8s 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 k8s storage classes:

1. Create a storage class yaml file, `storageclass-gold.yaml`.
Update the capabilities, pools, and array secrets accordingly.

Note:

- IBM FlashSystem A9000 and A9000R always includes deduplication and compression.
 - The Storwize applicable **SpaceEfficiency** values are: thin, compressed, or deduplicated. These values are not case specific.
 - The **csi.storage.k8s.io/fstype** parameter is optional. The values allowed are **ext4** or **xfs**. The default value is **ext4**.
-

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: block.csi.ibm.com
parameters:
  SpaceEfficiency: <VALUE>
  pool: <VALUE_POOL_NAME>

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

csi.storage.k8s.io/fstype: <xfs>
```

2. Apply the storage class.

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

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

Advanced configuration

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

- [“Importing an existing volume” on page 10](#)

Importing an existing volume

Volumes not created by the IBM block storage CSI driver can be imported by using a persistent volume (PV) yaml file.

About this task

Procedure

Volumes created externally from the CSI driver can be imported using the CSI driver. Use this procedure to help build a PV yaml file for your volumes.

Note: These steps are setup for importing volumes from a Storwize system. Change parameters, as needed.

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

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

Before creating, note the following:

- The **volumeAttributes** entries are for informational purposes only.

The following examples are attributes that can be included:

- **pool_name:** <Name of Pool where volume is located>
- **storage_type:** <SVC | A9K>
- **volume_name:** <Volume name>
- To find the **volumeHandle**, use one of the following:

- **Through command line (for Storwize):**

```
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:**

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

The **Volumes** page appears.

- 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](http://ibm.com/support/knowledgecenter/en/STVLF4) on Knowledge Center (ibm.com/support/knowledgecenter/en/STVLF4) or your specific product page.

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

- Select **Systems and Domains Views > Systems Ports Connectivity** from the side bar.

The **Systems Ports** panel is displayed.

- Select the **Ports** widget.

- From the **Actions** menu on the FC port, click **View/Edit FC Port**.

Use the WWPN number.

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

```

apiVersion: v1
kind: PersistentVolume
metadata:
  #annotations:
  #pv.kubernetes.io/provisioned-by: block.csi.ibm.com
  name: vol1-pv
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 5Gi
  csi:
    controllerPublishSecretRef:
      name: flashv7k-1
      namespace: kube-system
    driver: block.csi.ibm.com
    # volumeAttributes:
    # pool_name: ibmc-block-gold
    # storage_type: SVC
    # volume_name: vol1
    volumeHandle: SVC:600507640082000B08000000000004FF
    persistentVolumeReclaimPolicy: Retain
    storageClassName: ibmc-block-gold
    # volumeMode: Filesystem

```

2. Create a PersistentVolumeClaim (PVC) yaml file.

Note: To include specific a 5Gi PV, be sure to include the **storageClassName**.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  volume.beta.kubernetes.io/storage-provisioner: block.csi.ibm.com
  name: vol1-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: ibmc-block-gold
  volumeName: vol1-pv

```

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 13.](#)
- [“Recovering a crashed Kubernetes node” on page 22.](#)

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.

Use the following sample configurations for running stateful containers in different block storage systems:

- [“Running a stateful container on IBM FlashSystem A9000R” on page 13](#) gives an example for FlashSystem A9000R systems and can also be applied to FlashSystem A9000 and FlashSystem 9100 systems.
- [“Running a stateful container on Storwize systems” on page 17](#)

Running a stateful container on IBM FlashSystem A9000R

Use this information as a sample of how to run a stateful container on an IBM FlashSystem A9000R storage service with the IBM block storage CSI driver.

About this task

This example illustrates a basic configuration required for running a stateful container with volumes provisioned on an IBM FlashSystem A9000R storage service.

Note: The example uses FlashSystem A9000R as an example but the same can be used for FlashSystem A9000 and FlashSystem 9100.

- Creating a k8s secret **a9000-array1** for the storage system.
- Creating a storage class **gold**.
- Creating a PersistentVolumeClaim (PVC) **demo-pvc** that uses the storage class **gold** and show some details on the created PVC and persistent volume (PV).
- Creating a StatefulSet application **demo-statefulset** and observing the mountpoint / multipath device that was created by the driver.
- Writing data inside **demo-statefulset**, and then deleting and recreating **demo-statefulset**, verifying that the data still exists.

Procedure

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

```
$> cat demo-secret-a9000-array1.yaml
kind: Secret
apiVersion: v1
metadata:
  name: a9000-array1
  namespace: kube-system
type: Opaque
stringData:
  management_address: <VALUE-2,VALUE-3> # replace with valid storage system managment address
  username: <VALUE-4> # replace with valid username
data:
  password: <VALUE-5 base64> # replace with valid password
$> kubectl create -f demo-secret-a9000-array1.yaml
secret/a9000-array1 created
```

3. Create a storage class.

```
$> cat demo-storageclass-gold-A9000R.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: block.csi.ibm.com
parameters:
  pool: gold

  csi.storage.k8s.io/provisioner-secret-name: a9000-array1
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-publish-secret-name: a9000-array1
  csi.storage.k8s.io/controller-publish-secret-namespace: kube-system

  csi.storage.k8s.io/fstype: xfs # Optional. Values are ext4/xfs. The default is ext4.
  volume_name_prefix: demo1 # Optional.

$> kubectl create -f demo-storageclass-gold-A9000R.yaml
storageclass.storage.k8s.io/gold created
```

4. Create a PVC demo-pvc-gold with the size of 1 Gb.

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

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

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

```
$> kubectl get pv,pvc
NAME                                     CAPACITY  ACCESS MODES
persistentvolume/pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f  1Gi       RWO

RECLAIM POLICY  STATUS  CLAIM                STORAGECLASS  REASON  AGE
Delete          Bound   default/demo-pvc    gold          78s

NAME                                     STATUS  VOLUME                                     CAPACITY
persistentvolumeclaim/demo-pvc          Bound   pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f  1Gi

ACCESS MODES  STORAGECLASS  AGE
RWO           gold          78s

$> kubectl describe persistentvolume/pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Name:          pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: block.csi.ibm.com
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  gold
Status:        Bound
Claim:         default/demo-pvc
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:  A9000:6001738CFC9035EB0000000000D1F68F
  ReadOnly:      false
  VolumeAttributes:
    array_name=<IP>
    pool_name=gold
    storage.kubernetes.io/csiProvisionerIdentity=1565550204603-8081-
    block.csi.ibm.com
    storage_type=A9000
    volume_name=demo1_pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Events:         <none>
```

6. Create a StatefulSet application **demo-statefulset**, using the **demo-pvc**.

```
$> cat demo-statefulset-with-demo-pvc.yml
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-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: demo-pvc
              mountPath: "/data"
      volumes:
        - name: demo-pvc
          persistentVolumeClaim:
            claimName: demo-pvc

#nodeSelector:
#  kubernetes.io/hostname: NODESELECTOR

$> kubectl create -f demo-statefulset-with-demo-pvc.yml
statefulset/demo-statefulset created
```

7. Check the newly created pod.

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

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

- b) Check the mountpoint inside the pod.

```
$> kubectl exec demo-statefulset-0 -- bash -c "df -h /data"
Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/mpathz 1014M   33M  982M   4% /data

$> kubectl exec demo-statefulset-0 -- bash -c "mount | grep /data"
/dev/mapper/mpathz on /data type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

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
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 k8s-node1
```

- c) List the multipath devices on the worker node. Note the same **mpathz**, as mentioned in step “7.b” on page 16.

```
$>[k8s-node1] multipath -ll
mpathz (36001738cfc9035eb000000000d1f68f) dm-3 IBM ,2810XIV
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-a04bd32f-bd0f-11e9-a1f5
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-a04bd32f-bd0f-11e9-a1f5
```

- 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-a04bd32f-bd0f-11e9-a1f5-005056a45d5f/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-711b6fef-bcf9-11e9-a1f5-005056a45d5f/globalmount/.stageInfo.json
{"connectivity":"iscsi","mpathDevice":"dm-3","sysDevices":["sdb,sdc"]}
```

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

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

### Wait until the pod is deleted. Once deleted the '"demo-statefulset" not found' is
returned.
$> kubectl get statefulset/demo-statefulset
NAME                READY    STATUS    RESTARTS   AGE
demo-statefulset-0  0/1     Terminating    0          91m

##### Establish an SSH connection and log into the worker node in order to see that the
multipath was deleted and that the PV mountpoint no longer exists.

$> ssh 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

##### Recreate the statefulset and verify that /data/FILE exists.
$> kubectl create -f demo-statefulset-with-demo-pvc.yml
statefulset/demo-statefulset created

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

11. Delete StatefulSet and the PVC.

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

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

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

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

Running a stateful container on Storwize systems

This information provides an example of running a stateful container on Storwize systems using the IBM block storage CSI driver.

About this task

This example illustrates a basic configuration required for running a stateful container with volumes provisioned on a Storwize and Spectrum Virtualize storage services.

Note: This example can be used for any Storwize, Spectrum Virtualize, or FlashSystem as Storwize storage service.

- Creating a k8s secret **storwize-array1** for the storage system.

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

Procedure

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

```
$> cat demo-secret-storwize-array1.yaml
kind: Secret
apiVersion: v1
metadata:
  name: storwize-array1
  namespace: kube-system
type: Opaque
stringData:
  management_address: <VALUE-2,VALUE-3> # replace with valid storage system managment address
  username: <VALUE-4> # replace with valid username
data:
  password: <VALUE-5 base64> # replace with valid password

$> kubectl create -f demo-secret-storwize-array1.yaml
secret/storwize-array1 created
```

3. Create a storage class.

```
$> cat demo-storageclass-gold-storwize.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold

provisioner: block.csi.ibm.com
parameters:
  SpaceEfficiency: thick # SpaceEfficiency values are: thick, thin, deduplicated, and compressed.
  pool: gold

csi.storage.k8s.io/provisioner-secret-name: storwize-array1
csi.storage.k8s.io/provisioner-secret-namespace: kube-system
csi.storage.k8s.io/controller-publish-secret-name: storwize-array1
csi.storage.k8s.io/controller-publish-secret-namespace: kube-system

csi.storage.k8s.io/fstype: xfs # Optional. Values are ext4/xfs. The default is ext4.
volume_name_prefix: demo1 # Optional.

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

4. Create a PVC `demo-pvc-gold` with the size of 1 Gb.

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

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

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

```
$> kubectl get pv,pvc
NAME                                     CAPACITY  ACCESS MODES
persistentvolume/pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f  1Gi       RWO

RECLAIM POLICY  STATUS  CLAIM                STORAGECLASS  REASON  AGE
Delete          Bound   default/demo-pvc     gold          78s

NAME                                     STATUS  VOLUME                                     CAPACITY
persistentvolumeclaim/demo-pvc          Bound   pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f  1Gi

ACCESS MODES  STORAGECLASS  AGE
RWO           gold          78s

$> kubectl describe persistentvolume/pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Name:          pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: block.csi.ibm.com
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  gold
Status:        Bound
Claim:         default/demo-pvc
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:6001738CFC9035EB0000000000D1F68F
  ReadOnly:    false
  VolumeAttributes:
    array_name=<IP>
    pool_name=gold
    storage.kubernetes.io/csiProvisionerIdentity=1565550204603-8081-
    block.csi.ibm.com
    storage_type=SVC
    volume_name=demo1_pvc-a04bd32f-bd0f-11e9-a1f5-005056a45d5f
Events:        <none>
```

6. Create a StatefulSet application **demo-statefulset**, using the **demo-pvc**.

```
$> cat demo-statefulset-with-demo-pvc.yml
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: demo-statefulset
spec:
  selector:
    matchLabels:
      app: demo-statefulset
  serviceName: demo-statefulset
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-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: demo-pvc
              mountPath: "/data"
      volumes:
        - name: demo-pvc
          persistentVolumeClaim:
            claimName: demo-pvc

#nodeSelector:
# kubernetes.io/hostname: NODESELECTOR

$> kubectl create -f demo-statefulset-with-demo-pvc.yml
statefulset/demo-statefulset created
```

7. Check the newly created pod.

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

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

- b) Check the mountpoint inside the pod.

```
$> kubectl exec demo-statefulset-0 -- bash -c "df -h /data"
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/mpathz 1014M  33M  982M   4% /data

$> kubectl exec demo-statefulset-0 -- bash -c "mount | grep /data"
/dev/mapper/mpathz on /data type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

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
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 k8s-node1
```

- c) List the multipath devices on the worker node. Note the same **mpathz**, as mentioned in step [“7.b”](#) on page 20.

```
$>[k8s-node1] multipath -ll
mpathz (36001738cfc9035eb000000000d1f68f) 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-a04bd32f-bd0f-11e9-a1f5
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-a04bd32f-bd0f-11e9-a1f5
```

- 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-a04bd32f-bd0f-11e9-a1f5-005056a45d5f/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-711b6fef-bcf9-11e9-a1f5-005056a45d5f/globalmount/.stageInfo.json
{"connectivity":"iscsi","mpathDevice":"dm-3","sysDevices":["sdb,sdc"]}
```

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

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

### Wait until the pod is deleted. Once deleted the '"demo-statefulset" not found' is
returned.
$> kubectl get statefulset/demo-statefulset
NAME                READY    STATUS    RESTARTS   AGE
demo-statefulset-0  0/1     Terminating    0          91m

##### Establish an SSH connection and log into the worker node in order to see that the
multipath was deleted and that the PV mountpoint no longer exists.

$> ssh 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

##### Recreate the statefulset and verify that /data/FILE exists.
$> kubectl create -f demo-statefulset-with-demo-pvc.yml
statefulset/demo-statefulset created

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

11. Delete StatefulSet and the PVC.

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

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

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

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

Recovering 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

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 v1.14.1
k8s-node1 Ready <none> 6d v1.14.1
k8s-node3 NotReady <none> 6d v1.14.1

$> 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

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 v1.14.1
k8s-node1 Ready <none> 6d v1.14.1
k8s-node3 NotReady <none> 6d v1.14.1

$> 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.

Checking logs

You can use the CSI (Container Storage Interface) driver logs for problem identification. 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 kube-system -l csi
```

For example:

```
$> kubectl get all -n kube-system -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 kube-system 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 kube-system ibm-block-csi-node-<PODID> -c ibm-block-csi-node
```

Detecting errors

This is an overview of actions that you can take to pinpoint a potential cause for a stateful pod failure. The table at the end of the procedure describes the problems and provides possible corrective actions.

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

```
$> kubectl get all -n kube-system -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 kube-system pod/ibm-block-csi-controller-0
```

View the logs.

Miscellaneous troubleshooting

General troubleshooting

Use the following command for general troubleshooting:

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

For example:

```
$> kubectl get -n kube-system 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

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/mpathm: One or more block group descriptor checksums are invalid. FIXED.
/dev/mapper/mpathm: Group descriptor 0 checksum is 0x0000, should be 0x3baa.

/dev/mapper/mpathm: 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.

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).

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

SC27-9590-00

