IBM block storage CSI driver
1.6.0

User Guide

SC27-9590-10

Note

Before using this document and the product it supports, read the information in #unique_2.

Edition notice

Publication number: SC27-9590-10. This publication applies to version 1.6.0 of IBM block storage CSI driver and to all subsequent releases and modifications until otherwise indicated in a newer publication.

Contents

Figures

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

IBM resources

- IBM block storage CSI driver documentation (ibm.com/docs/en/stg-block-csi-driver)
- IBM SAN Volume Controller documentation (ibm.com/docs/en/sanvolumecontroller)
- IBM Spectrum Scale documentation (ibm.com/docs/en/spectrum-scale)
- IBM FlashSystem® 5xxx family, Storwize® V5100 and V5000E documentation (ibm.com/docs/en/flashsystem-5x00)
- IBM FlashSystem 7200 and Storwize® V7000 documentation (ibm.com/docs/en/flashsystem-7x00)
- IBM Spectrum Virtualize as Software Only documentation (ibm.com/docs/en/spectrumvirtualsoftw)
- IBM FlashSystem® 9200 and 9100 documentation (ibm.com/docs/en/flashsystem-9x00)
- IBM FlashSystem A9000 documentation (ibm.com/docs/en/flashsystem-a9000)
- IBM FlashSystem A9000R documentation (ibm.com/docs/en/flashsystem-a9000r)
- IBM DS8880 documentation (ibm.com/docs/en/ds8880)
- IBM DS8900 documentation (ibm.com/docs/en/ds8900)
- IBM Spectrum® Access for IBM Cloud® Private Blueprint (ibm.com/downloads/cas/KK5PGD8E)

  Used as the FlexVolume driver-based solution for OpenShift® 3.11, using IBM Storage Enabler for Containers (ibm.com/docs/en/stgenablercontainers)
- IBM Storage for Red Hat® OpenShift Blueprint (http://www.redbooks.ibm.com/abstracts/redp5565.html?Open)
- Using the IBM Block Storage CSI driver in a Red Hat OpenShift environment (http://www.redbooks.ibm.com/abstracts/redp5613.html?Open)

External resources

- Persistent volumes on Kubernetes (kubernetes.io/docs/concepts/storage/volumes)
- Kubernetes Documentation (kubernetes.io/docs/home/)
- Kubernetes Blog (kubernetes.io//blog)

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 (ibm.com[®])
- IBM Support Portal website (ibm.com/support/entry/portal/support?brandind=Hardware~System_Storage)
- IBM Directory of Worldwide Contacts website (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) 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 ibmdocs@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. Overview

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 specific Spectrum Scale and Spectrum Scale CSI driver product information, see IBM Spectrum Scale documentation.

For details about volume provisioning with Kubernetes, refer to Persistent volumes on Kubernetes.

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

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
- Installing the operator and driver
- Upgrading the CSI driver

For information about uninstallation, see Uninstalling.

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.

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

For IBM Cloud® Satellite users, see cloud.ibm.com/docs/satellite for full system requirements.

Important: When using Satellite, complete the following checks, configurations, and the installation process before assigning the hosts to your locations. In addition, **do not** create a Kubernetes cluster. This is done through Satellite.

- 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 (including FlashSystem 5xxx, 7200, 9100, 9200, 9200R)**

    Port 22
  - **DS8000® Family systems**

    Port 8452
- Be sure that multipathing is installed and running.

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

1. **For RHEL OS users:** Ensure iSCSI connectivity. If using RHCOS or if the packages are already installed, skip this step and continue to step 2.

2. Configure Linux® multipath devices on the host.

   Important: Be sure to configure each worker with storage connectivity according to your storage system instructions. For more information, find your storage system documentation in IBM Documentation.

   Additional configuration steps for OpenShift® Container Platform users (RHEL and RHCOS). Other users can continue to step 3.

   Download and save the following yaml file:

   ```
   curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/master/deploy/99-ibm-
   attach.yaml > 99-ibm-attach.yaml
   ```

   This 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.

   Apply the yaml file.

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

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.

   Install both the Snapshot CRDs and the Common Snapshot Controller once per cluster.

   The instructions and relevant yaml files to enable volume snapshots can be found at: https://github.com/kubernetes-csi/external-snapshotter#usage

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.

The CSI operator and driver can be downloaded in one of the following ways:

- With the OpenShift web console (see Installing the driver with the OpenShift web console).
- With GitHub (see Installing the driver with GitHub).
- With OperatorHub.io (see Installing the driver with OperatorHub.io).
- With the IBM Cloud® Satellite web console. For more information, see cloud.ibm.com/docs/satellite.

Installing the driver with 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.

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

1. From Red Hat OpenShift Container Platform **Home** > **Projects**, click **Create Project**. In the **Create Project** dialog box, enter a Project name (also referred to as namespace). Click **Create** to save.
2. From **Operators** > **OperatorHub**. Select the namespace from `Projects:<namespace>`, as defined in step 1.
3. Search for IBM block storage CSI driver.
4. Select the **Operator for IBM block storage CSI driver** and click **Install**. The **Operator Installation** form appears.
5. Set the **Installation Mode** to the project namespace selected previously, in step 2, under **A specific namespace on the cluster**.
6. Set the **Approval Strategy** to either **Automatic** or **Manual** as per your preference.

   Note: The general recommendation is to select **Automatic** option.
7. Click **Install**.
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 *Succeeded*.

   Note: While waiting for the **Status** to change from *Up to date* to *Succeeded*, you can check the pod progress and readiness status from **Workloads** > **Pods**.
9. After the operator installation progress is complete, click 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 file 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 with GitHub

The operator for IBM® block storage CSI driver can be installed directly with GitHub. Installing the CSI (Container Storage Interface) driver is part of the operator installation process.

Use the following steps to install the operator and driver, with GitHub (github.com/IBM/ibm-block-csi-operator).

Note: Before you begin, you may need to create a user-defined namespace. Create the project namespace, using the `kubectl create ns <namespace>` command.

1. Install the operator.

   a. Download the manifest from GitHub.

   ```
   curl https://raw.githubusercontent.com/IBM/ibm-block-csi-operator/v1.6.0/deploy/installer/
   generated/ibm-block-csi-operator.yaml > ibm-block-csi-operator.yaml
   ```

   b. **Optional:** Update the image fields in the ibm-block-csi-operator.yaml.

   c. Install the operator, using a user-defined namespace.

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

   d. Verify that the operator is running. (Make sure that 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
   ```

2. 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/v1.6.0/deploy/crds/
   csi.ibm.com_v1_ibmblockcsi_cr.yaml > csi.ibm.com_v1_ibmblockcsi_cr.yaml
   ```

   b. **Optional:** Update the image repository field, tag field, or both in the csi.ibm.com_v1_ibmblockcsi_cr.yaml.

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

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

   d. Verify that the driver is running:

   ```
   $ kubectl get pods -n <namespace> -l csi
   NAME                                    READY   STATUS  RESTARTS AGE
   ibm-block-csi-controller-0              6/6     Running 0        9m36s
   ibm-block-csi-node-jvmvh                3/3     Running 0        9m36s
   ibm-block-csi-node-tsppw                3/3     Running 0        9m36s
   ibm-block-csi-operator-5bb7996b86-xntss 1/1     Running 0        10m
   ```

Installing the driver with OperatorHub.io

When using OperatorHub.io, the operator for IBM® block storage CSI driver can be installed directly from the OperatorHub.io website. Installing the CSI (Container Storage Interface) driver is part of the operator installation process.

To install the CSI driver from OperatorHub.io, go to https://operatorhub.io/operator/ibm-block-csi-operator-community and follow the installation instructions, once clicking the **Install** button.

Note: To ensure that the operator installs the driver, be sure to apply the yaml that is located as part of the ibm-block-csi-operator-community page mentioned above.

Uninstalling

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

The CSI operator and driver can be uninstalled in one of the following ways:

Note: Be sure to use the corresponding uninstall method to operator and driver installation originally used.

- With the OpenShift web console (see Uninstalling the driver with the OpenShift web console).
- With GitHub (see Uninstalling the driver with GitHub).
- With OperatorHub.io (see Uninstalling the driver with OperatorHub.io).

- With the IBM Cloud® Satellite web console.

  To uninstall from IBM Cloud® Satellite web console, **Remove** the CSI driver subscription. For more information, see cloud.ibm.com/docs/satellite.

Uninstalling the driver with 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.

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 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 with GitHub

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

Perform the following steps in order to uninstall the CSI driver and operator.

1. Delete the IBMBlockCSI custom resource.

   ```
   kubectl -n <namespace> delete -f csi.ibm.com_v1_ibmblockcsi_cr.yaml
   ```
2. Delete the operator.

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

Uninstalling the driver with OperatorHub.io

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

To uninstall the CSI driver with OperatorHub.io, use the `kubectl delete -f` command to delete the yaml files, one at a time, in the reverse order of the installation steps that are documented in https://operatorhub.io/operator/ibm-block-csi-operator-community.

Note: To see the installation steps, click **Install** on the OperatorHub.io webpage.

Upgrading the CSI driver

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

- The OpenShift web console and OperatorHub.io both automatically upgrade the the CSI (Container Storage Interface) driver when a new version is released.
  - With OpenShift web console, the **Approval Strategy** must be set to **Automatic**.

    To check if your operator is running at the latest release level, from the OpenShift web console, browse to **Operators** > **Installed Operators**. Check the status of the Operator for IBM block storage CSI driver. Ensure that the **Upgrade Status** is *Up to date*.

  Note: For more information about automatic upgrades, see https://olm.operatorframework.io/docs/concepts/crds/subscription/.
- To manually upgrade the CSI driver with the OpenShift web console, see Manual upgrade with OpenShift.
- To manually upgrade the CSI (Container Storage Interface) driver from a previous version with GitHub, perform step 1 of the installation procedure for the latest version.

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.

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), in order to use the driver and run stateful applications using IBM block storage systems, the relevant yaml files must be created.

Multiple yaml files per type can be created (with different configurations), according to your storage needs.

- Creating a Secret
- Creating a StorageClass
- Creating a PersistentVolumeClaim (PVC)
- Creating a StatefulSet
- Creating a VolumeSnapshotClass
- Creating a VolumeSnapshot
- Expanding a PersistentVolumeClaim (PVC)
- Advanced configuration

Creating a Secret

Create an array secret YAML file 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, similar to the following demo-secret.yaml:

   The `management_address` field can contain more than one address, with each value separated by a comma.

```
kind: Secret
apiVersion: v1
metadata:
  name:  demo-secret
  namespace: default
type: Opaque
stringData:
  management_address: demo-management-address  # Array management addresses
  username: demo-username                      # Array username
data:
  password: ZGVtby1wYXNzd29yZA==               # base64 array password
```

2. Apply the secret using the following command:

   `kubectl apply -f demo-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 a StorageClass

Create a storage class yaml file in order to define the storage system pool name, secret reference, `SpaceEfficiency`, and `fstype`.

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.

Create a storage class yaml file, similar to the following demo-storageclass.yaml.

Update the capabilities, pools, and array secrets, as needed.

Use the `SpaceEfficiency` parameters for each storage system, as defined in the following table. These values are not case-sensitive.

Table: *`SpaceEfficiency` 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	- thick (default value)- thin- compressed- deduplicated **Note:** If not specified, the default value is thick.
IBM® DS8000® Family	- none (default value) - thin **Note:** If not specified, the default value is none.

- The IBM DS8000 Family `pool` value is the pool ID and not the pool name as is used in other storage systems.
- Be sure that the `pool` value is the name of an existing pool on the storage system.
- The `allowVolumeExpansion` parameter is optional but is necessary for using volume expansion. The default value is *false*.

Note: Be sure to set the value to true to allow volume expansion.

- The `csi.storage.k8s.io/fstype` parameter is optional. The values that are allowed are *ext4* or *xfs*. The default value is *ext4*.
- The `volume_name_prefix` parameter is optional.

Note: For IBM DS8000 Family, the maximum prefix length is five characters. The maximum prefix length for other systems is 20 characters. For storage systems that use Spectrum Virtualize, the `CSI_` prefix is added as default if not specified by the user.

```
  kind: StorageClass
  apiVersion: storage.k8s.io/v1
  metadata:
    name: demo-storageclass
  provisioner: block.csi.ibm.com
  parameters:
    SpaceEfficiency: deduplicated   # Optional.
    pool: demo-pool
    csi.storage.k8s.io/provisioner-secret-name: demo-secret
    csi.storage.k8s.io/provisioner-secret-namespace: default
    csi.storage.k8s.io/controller-publish-secret-name: demo-secret
    csi.storage.k8s.io/controller-publish-secret-namespace: default
    csi.storage.k8s.io/controller-expand-secret-name: demo-secret
    csi.storage.k8s.io/controller-expand-secret-namespace: default

    csi.storage.k8s.io/fstype: xfs   # Optional. Values ext4\xfs. The default is ext4.
    volume_name_prefix: demoPVC      # Optional.
  allowVolumeExpansion: true
```

Apply the storage class.

```
kubectl apply -f demo-storageclass.yaml
```

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

Creating a PersistentVolumeClaim (PVC)

Create a PersistentVolumeClaim (PVC) yaml file for a persistent volume (PV).

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 creating yaml files for PVCs with file system and raw block volume types. After each yaml file creation, use the `kubectl apply` command.

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

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

Use the following sections, according to your PVC needs:

- Creating PVC for volume with file system
- Creating PVC for raw block volume
- Creating PVC from volume snapshot
- Creating a volume clone from an existing PVC

Creating PVC for volume with Filesystem

Create a PVC yaml file, similar to the following 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  # Optional. The default is Filesystem.
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: demo-storageclass
```

Creating PVC for raw block volume

Create a PVC yaml file, similar to the following demo-pvc-raw-block.yaml 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: demo-storageclass
```

Creating PVC from volume snapshot

To create a PVC from an existing volume snapshot, create a PVC yaml file, similar to the following demo-pvc-from-snapshot.yaml file, with the size of 1 Gb.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc-from-snapshot
spec:
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: demo-storageclass
  dataSource:
    name: demo-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Creating a volume clone from an existing PVC

Note: IBM FlashCopy® function is referred to as the more generic volume snapshots and cloning within this documentation set. Not all supported products use the FlashCopy function terminology.

To create a volume clone from an existing PVC object, create a PVC yaml file, similar to the following demo-pvc-cloned-pvc.yaml file, with the size of 1 Gb.

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

Creating a StatefulSet

Create a StatefulSet yaml file to manage stateful applications.

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 yaml creation of StatefulSets with file system, raw block volume, and mixed types. After each yaml file creation, use the `kubectl apply` command.

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

The `statefulset.apps/<filename> created` message is emitted.

Creating a StatefulSet with file system volumes

Create a StatefulSet yaml file, similar to the following demo-statefulset-file-system.yaml file.

```
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: demo-container
        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"
      volumes:
      - name: demo-volume-file-system
        persistentVolumeClaim:
          claimName: demo-pvc-file-system
```

Creating a StatefulSet with raw block volume

Create a StatefulSet yaml file, similar to the following demo-statefulset-raw-block.yaml file.

```
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: demo-container
        image: registry.access.redhat.com/ubi8/ubi:latest
        command: [ "/bin/sh", "-c", "--" ]
        args: [ "while true; do sleep 30; done;" ]
        volumeDevices:
          - name: demo-volume-raw-block
            devicePath: "/dev/block"
      volumes:
      - name: demo-volume-raw-block
        persistentVolumeClaim:
          claimName: demo-pvc-raw-block
```

Creating a StatefulSet with both raw block and file system volumes

Create a StatefulSet yaml file, similar to the following demo-statefulset-combined.yaml file.

```
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: demo-container
      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
```

Creating a VolumeSnapshotClass

Create a VolumeSnapshotClass YAML file to enable creation and deletion of volume snapshots.

Note:

- IBM® FlashCopy® function is referred to as the more generic volume snapshots and cloning 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.

In order to enable creation and deletion of volume snapshots for your storage system, create a VolumeSnapshotClass YAML file, similar to the following demo-snapshotclass.yaml.

When configuring the file, be sure to use the same array secret and array secret namespace as defined in Creating a Secret.

- The `snapshot_name_prefix` parameter is optional.

  Note: For IBM DS8000® Family, the maximum prefix length is five characters.The maximum prefix length for other systems is 20 characters.For storage systems using Spectrum Virtualize, the `CSI_` prefix is added as default if not specified by the user.
- The `pool` parameter is not available on IBM FlashSystem A9000 and A9000R storage systems. For these storage systems the snapshot must be created on the same pool as the source.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: demo-snapshotclass
driver: block.csi.ibm.com
deletionPolicy: Delete
parameters:
  csi.storage.k8s.io/snapshotter-secret-name: demo-secret
  csi.storage.k8s.io/snapshotter-secret-namespace: default
  snapshot_name_prefix: demoSnapshot   # Optional.
  pool: demo-pool                      # Optional. Use to create the snapshot on a different
 pool than the source.
```

After the YAML file is created, apply it by using the `kubectl apply -f` command.

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

Creating a VolumeSnapshot

Create a VolumeSnapshot yaml file for a specific PersistentVolumeClaim (PVC).

VolumeSnapshotClass needs to be present before a VolumeSnapshot can be created. For more information, see Creating a VolumeSnapshotClass.

Note:

- IBM® FlashCopy® function is referred to as the more generic volume snapshots and cloning 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 before snapshot creation.

1. Create a snapshot for a specific PersistentVolumeClaim (PVC) using the demo-snapshot.yaml.

   For more information about PVC configuration, see Creating a PersistentVolumeClaim (PVC).

   ```
   apiVersion: snapshot.storage.k8s.io/v1beta1
   kind: VolumeSnapshot
   metadata:
     name: demo-snapshot
   spec:
     volumeSnapshotClassName: demo-snapshotclass
     source:
       persistentVolumeClaimName: demo-pvc-file-system
   ```

2. After the YAML file is created, apply it by using the `kubectl apply -f` command.

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

3. Verify that the VolumeSnapshot was 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 (provisioning) a volume from this snapshot.

Expanding a PersistentVolumeClaim (PVC)

Use this information to expand existing volumes.

Important: Before expanding an existing volume, be sure that the relevant StorageClass yaml `allowVolumeExpansion` parameter is set to true. For more information, see Creating a StorageClass.

To expand an existing volume, open the relevant PersistentVolumeClaim (PVC) yaml file and increase the `storage` parameter value. For example, if the current `storage` value is set to *1Gi*, you can change it to *10Gi*, as needed. For more information about PVC configuration, see Creating a PersistentVolumeClaim (PVC).

Be sure to use the `kubectl apply` command in order to apply your changes.

Advanced configuration

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

- Importing an existing volume

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 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   60035E8000000000091F0C0
  ```

- 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 products, find your product information in IBM Documentation.

- Through the IBM Hyper-Scale Manager user interface for FlashSystem A9000 and A90000R 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, see IBM Hyper-Scale Manager documentation.

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.

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: demo-secret
         namespace: default
       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 a specific 5 Gi PV, be sure to include the `storageClassName`.
   - For more information about creating a PVC yaml file, see Creating a PersistentVolumeClaim (PVC).

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     # annotations:
       # pv.kubernetes.io/provisioned-by: 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.

   - **With OpenShift® web console**

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

   Click **Create** to save.

   - **With GitHub**

   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 about creating a StatefulSet, see Creating a StatefulSet.

```
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

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

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 (including FlashSystem 5xxx, 7200, 9100, 9200, 9200R)	storwize-array1
IBM DS8000® Family products	DS8000-array1

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 this information to run a stateful container on StatefulSet volumes using either file systems or raw block volumes.

1. Create an array secret, as described in Creating a Secret.
2. Create a storage class, as described in Creating a StorageClass.

   Remember: 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 a StorageClass.

   This parameter is not applicable for IBM FlashSystem A9000 and A9000R systems. These systems always include deduplication and compression.
3. Create a PVC with the size of 1 Gb, as described in Creating a PersistentVolumeClaim (PVC).
4. Display the existing PVC and the created persistent volume (PV).
5. Create a StatefulSet, as described in Creating a StatefulSet.

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
```

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
   ```

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).

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
```

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

Use the following command for general troubleshooting:

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

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.

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 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 (www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Red Hat®, and OpenShift® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Printed in USA

SC27-9590-10