中国电子云

CeaStor CSI Plugin 3.2.2

Guidelines for use

Classification: Public

Document Version: 01

Publish date: 2024-05-30

CLP Cloud Computing Technology Co.

preamble

summarize

This document mainly introduces the usage guide of CeaStor CSI plug-in, including the information of installation and configuration, deployment and usage, etc., to help you quickly get started with the use of CeaStor CSI plug-in.

Reader Objects

This document is intended for the following audience:

- Technical Support Engineer
- Delivery Engineer
- Maintenance Engineer

symbolic convention

警告	The notes following this sign require extra attention, and improper handling may cause personal injury.
注意	Reminds of the precautions to be taken during operation, improper operation may result in data loss or equipment damage. "Attention" does not involve bodily harm.
说明	Provide additional explanations of key information in the body of the text as necessary. "Instructions" is not a safety warning message and does not relate to personal, equipment or environmental injury information.
提示	Tips, tricks for configuring, operating or using the product.

revised record

Document version	Release time	revised description
01	2024-05-30	First official release.

Contents

1 summarize

1.1 summary

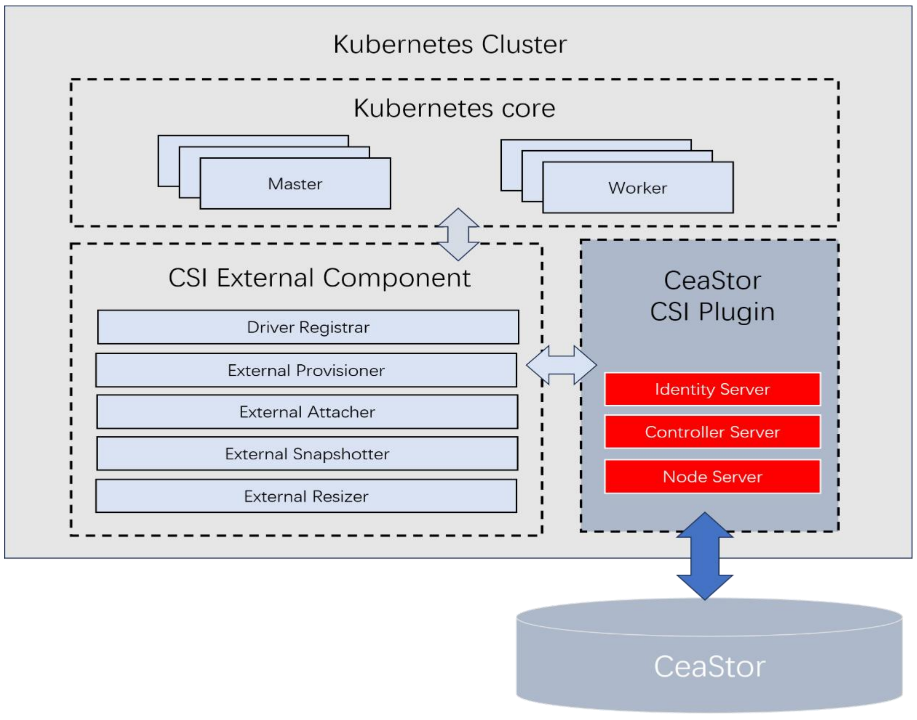
Kubernetes (or K8s), is a portable and scalable open source container orchestration management platform for managing containerized workloads and services.

CSI (Container Storage Interface) is an industry standard for exposing block storage systems to container workloads on Container Orchestration Systems (CO) such as Kubernetes.The CeaStor CSI plugin is used to provide storage services to container workloads on Kubernetes and is the required plug-in for the use of the CeaStor storage system in a Kubernetes environment.

PVC (Persistent Volume Claim) capability is provided to Kubernetes by installing the CeaStor CSI plug-in and deploying PVCs in the Kubernetes cluster to interface with CeaStor storage system resources.

Kubernetes registers to listen to Kubernetes object resources via the sidecar component and initiates calls to the CeaStor CSI plugin.The CeaStor CSI plugin implements the sidecar-initiated calls on the CeaStor storage system, e.g., initiating the Create a Persistent Volume (Persistent Volume (PV) operation on Kubernetes is implemented on CeaStor to create a CBD volume/filesystem.The schematic diagram of how the CeaStor CSI plugin works between a Kubernetes cluster and the CeaStor storage system is shown below:

图 1-1 CeaStor CSI plug-in working diagram



1.2 Feature List

The functional features supported by CeaStor CSI are listed in the following table:

表 1-1 Supported Functional Features

Feature Functions	Functional Description	Limitations on use
Creating a Volume	Dynamic Volume Creation (PVC creation)	Volume capacity 1MiiB~512TiB
	Clone rolls (cloned PVC)	Volume capacity 1MiB~512TiB, capacity greater than or equal to the source volume.
Delete Volume	Delete Volume	No subvolumes or snapshots
expansion volume	Expansion rolls (expansion PVC) If the PV is mounted, include the expansion file system.	The expansion size needs to be larger than the original volume size. ● ROX does not support online expansion. ● RWX does not support online expansion by default, you need to configure the startup item field --EnableRWXExpendOnline=true
Creating a Snapshot	Creating a Volume Snapshot	-
Deleting snapshots	Deleting a Volume Snapshot	-
Mounted PVC	Mounting PVCs for use by Pods Supported as a block device (Block mode) or as a file system (Filesystem mode) type.	● Block mode support RWO/ROX/RWX ● Filesystem mode support for RWO/ROX
Unloading PVC	Uninstall PVC from Pod	-
Volume Capacity Statistics	Volume Used Capacity	-

1.3 compatibility

Container management platforms supported by the CeaStor CSI plugin, along with operating system and multipath version information.

表 1-2 Supported Container Management Platforms

Container Management Platform	releases
Kubernetes	1.18~1.26

表 1-3 Supported operating systems and versions

Operating System Name	Operating system version	Native Multipath version
CCLinux x86_64	20.09.2	Comes with OS, supports FC/iSCSI

Operating System Name	Operating system version	Native Multipath version
CCLinux ARM		

表 1-4 Compatibility Requirements

Supported Functions	Compatibility Requirements
Mount mode file system support	Kubernetes 1.18 and above
Support for volume snapshots and cloning	Kubernetes 1.20 and above
Volume expansion support	Kubernetes 1.24 and above

1.4 Key Specifications

CeaStor CSI plug-in key specifications:

表 1-5 Description of key specifications

specifications	Specification
Volume	Subject to the volume specifications of the CeaStor storage system, a single cluster supports the creation of up to 100,000 volumes.
Snapshot	Subject to the CeaStor Storage System Volume Snapshot specification, a single volume supports the creation of up to 256 snapshots.

2 Preparation for installation

2.1 Checking the CSI dependency environment

Before installing and configuring the CeaStor CSI plug-in, check that the CSI plug-in runtime environment meets the configuration requirements.

表 2-1 Dependent environment checklist

matrix	checklist	Inspection instructions
Kubernetes Cluster Management Platform	The management platform is functioning properly.	Verify that the container platform status is healthy.
	The management platform version meets the compatibility requirements.	Satisfaction Table 1-4 Requirements.
	Host Connectivity Configuration	Confirm that network connectivity to the CeaStor storage system is normal.
Kubernetes Cluster Business Hosts	Multipath software configuration	(Optional) If multipathing needs to be enabled, make sure that all nodes that need to have the CeaStor CSI plug-in enabled have multipathing software installed.
	NVMe client software configuration	(Optional) If you choose to use the NVMe-OF protocol, make sure that the NVMe client software is installed on all nodes that need to launch the CeaStor CSI plug-in.
	iSCSI Client Software Configuration	(Optional) If you choose to use the iSCSI protocol, make sure that the iSCSI client software is installed on all nodes that need to launch the CeaStor CSI plug-in.
CeaStor Storage Systems	License information	• Confirm that the License authorization time is within the usage period. • Verify that the License authorization capacity meets the business requirements.
	System Cluster Operational Status	Verify that the cluster is running in a "healthy" state.
	System Host Service Status	Verify that the host service status is "Up and Running".
	Storage System Network Status	Verify that the host service status is "Up and Running".
	Hard disk pool status and usage	Verify that there is a hard disk pool for Block Service and that its status is "Normal".
	Access Path Status	Verify that the access path status is "Healthy".

2.2 Prepare CSI component packages

2.2.1 Understanding CSI Component Packages

Component Package Name

The CeaStor CSI component package naming convention is: CeaStor-CSI-{version number}-{version number}-{release time}.tar.gz. For example, CeaStor-CSI-3.2.1-2906-20240324003106.tar.gz.

Component Package Catalog Structure

The CSI component package directory is structured as follows:

```
CeaStor-CSI-3.2.1-xxxx-yyyyyMMddHHmmss
├─── csi_image_list.json
├─── images
|  ├─── amd64
|  ├─── cds-csi-nvmf_amd64
|  ├─── csi-snapshot-validation-webhook_amd64
|  ├─── csi-snapshot-controller_amd64
|  ├─── csi-node-driver-registrar_amd64
|  ├─── csi-livenessprobe_amd64
|  ├─── csi-external-snapshotter_amd64
|  ├─── csi-external-resizer_amd64
|  ├─── csi-external-provisioner_amd64
|  ├─── csi-external-attacher_amd64
|  ├─── cluster-storage-operator_amd64
|  ├─── cluster-csi-snapshot-controller-operator_amd64
|  ├─── arm64
|  ├─── cds-csi-nvmf_arm64
|  ├─── csi-snapshot-validation-webhook_arm64
|  ├─── csi-snapshot-controller_arm64
|  ├─── csi-node-driver-registrar_arm64
|  ├─── csi-livenessprobe_arm64
|  ├─── csi-external-snapshotter_arm64
|  ├─── csi-external-resizer_arm64
|  ├─── csi-external-provisioner_arm64
|  ├─── csi-external-attacher_arm64
|  ├─── cluster-storage-operator_arm64
|  ├─── cluster-csi-snapshot-controller-operator_arm64
deploy
|  ├─── kubernetes
|  ├─── configmap-csi-iscsiadm.yaml
|  ├─── configmap-csi-multipathd.yaml
|  ├─── configmap-csi-multipath.yaml
|  ├─── csi_controller_deployment.yaml
|  ├─── csi-iscsi-controller.yaml
|  ├─── csi-iscsi-driver.yaml
|  ├── csi-iscsi-node.yaml
|  ├── csi-node-rbac.yaml
|  ├── csi-nvmf-controller.yaml
|  ├── csi-nvmf-driver.yaml
|  ├── csi-nvmf-node.yaml
|  ├── operand_rbac.yaml
|  ├── serviceaccount.yaml
|  ├── snapshot.storage.k8s.io_volumesnapshotclasses.yaml
|  ├── snapshot.storage.k8s.io_volumesnapshotcontents.yaml
|  ├── snapshot.storage.k8s.io_volumesnapshots.yaml
Examples
|  ├── kubernetes
|  ├── clone.yaml
|  ├── pod-block.yaml
|  ├── pod.yaml
|  ├── pvc.yaml
|  ├── restore.yaml
|  ├── secret.yaml
|  ├── storageclass.yaml
|  ├── volumesnapshotclass.yaml
|  ├── volumesnapshot-dynamic.yaml
```

mirror image package

Under the images/amd64 and images/arm64 paths, the CeaStor CSI image packages to be loaded for the Kubernetes cluster. The nodes and descriptions of the image packages to be loaded are shown in the following table:

表 2-2 Mirror package description

Mirror package name	clarification	loading node
csi-snapshot-controller	CSI snapshot controller component image	All master nodes
csi-external-snapshotter	CSI external snapshotter component image	
csi-external-resizer	CSI external resizer component mirroring	
csi-external-provisioner	CSI external provisioner component image	
csi-external-attacher	CSI external attacher component mirroring	
csi-snapshot-validation-webhook	(Optional) CSI snapshot validation webhook image	Any master node
csi-livenessprobe	(Optional) CSI Health Screening	
cluster-storage-operator	(Optional) Define OpenShift Container Platform cluster-wide storage defaults to ensure that a default storage class exists for OpenShift Container Platform clusters.	
cluster-csi-snapshot-controller-operator	(Optional) Define CSI Snapshot Controller deployment and configuration. The CSI Snapshot Controller is responsible for monitoring the VolumeSnapshot CRD object and managing the creation and deletion lifecycle of volume	

Mirror package name	clarification	loading node
	snapshots.	
cds-csi-nvmf_	CeaStor CSI Plugin image package. Startup nodes such as master, need to add startup item configuration: --IsControllerServer = true	Requires a CSI worker or /master node
csi-node-driver-registrar	CSI driver registry component image	

yaml configuration file

Under the deploy/kubernetes path, the yaml configuration file for the CeaStor CSI component. The configuration file needs to load the nodes and descriptions as shown in the following table:

⚠ 注意

CeaStor CSI provides a pre-built yaml configuration file. If you need to modify the yaml file configuration, please contact technical support.

表 2-3 yaml configuration file description

Name of the document	clarification	loading node
configmap-csi-iscsiadm.yaml	iSCSI control command configuration. Mapping iscsiadm commands in containers.	Any master node
configmap-csi-multipathd.yaml	iSCSI multipath command configuration. Mapping multipathd commands in containers.	
configmap-csi-multipath.yaml	iSCSI multipath command configuration. Mapping the multipah command in a container.	
csi_controller_deployment.yaml	Snapshots control container configuration.	
csi-iscsi-controller.yaml	CSI controller configuration with container node-registrar, csi-provisioner, csi-attacher, csi-resizer, csi-snapshotter, and so on.	
csi-iscsi-driver.yaml	CSI drive configuration.	
csi-iscsi-node.yaml	Worker node configuration with container node-registrar, csi-iscsi-plugin.	
csi-node-rbac.yaml	Authorizes the node driver to operate the relevant APIs.	
csi-nvmf-controller.yaml	CSI controller configuration with container node-registrar, csi-provisioner, csi-attacher, csi-resizer, csi-snapshotter, and so on.	
csi-nvmf-driver.yaml	CSI drive configuration.	
csi-nvmf-node.yaml	Worker node configuration with container node-registrar, csi-nvmf-plugin.	
operand_rbac.yaml	Privilege configuration for the APIs related to authorizing snapshot driver operations.	
serviceaccount.yaml	Service account configuration.	

snapshot.storage.k8s.io_volumesnapshotclasses.yaml	K8s user-defined resource volumesnapshotclasses configuration.	
snapshot.storage.k8s.io_volumesnapshotcontents.yaml	K8s user-defined resource volumesnapshotcontents configuration.	
snapshot.storage.k8s.io_volumesnapshots.yaml	K8s user-defined resource volumesnapshots configuration.	

yaml configuration template

Under the examples/kubernetes path, configure the yaml configuration template required for the CeaStor CSI plugin installation. The configuration template needs to be loaded with the nodes and descriptions shown in the following table:

表 2-4 yaml configuration template description

file	clarification	loading node
clone.yaml	Cloning PVC templates.	Any master node
pod-block.yaml	Block mode PVC mounts the pod template.	
pod.yaml	Mount mode PVC mounts the pod template.	
pvc.yaml	PVC template.	
restore.yaml	Snapshot clone volume PVC template.	
secret.yaml	Docking storage authentication configuration templates.	
storageclass.yaml	Storage class configuration template.	
volumesnapshotclass.yaml	Snapshot class configuration template.	
volumesnapshot-dynamic.yaml	Snapshot resource creation template.	

CSI Mirror Package

The ceastor-csi-driver image package is unzipped to view the CSI components.The CSI components and descriptions included in the CSI image package are listed below:

表 2-5 CSI Mirror Package Description

CSI components	clarification
csi-external-provisioner	• When creating a PVC, the CSI Controller service is invoked to create a CBD volume as a PV on the storage and bind the PV to the PVC. • When deleting a PVC, call the CSI Controller service to unbind the PV to the PVC, and then delete the CBD volume corresponding to the PV from storage.
csi-external-attacher	When creating/deleting a Pod, it is the corresponding pv that is visible to the node that generates the volumeattach resource.
csi-external-resizer	When expanding a PVC, the CSI is called to give the PVC more storage capacity space.
csi-external-snapshotter	When creating/deleting a VolumeSnapshot, the CSI is called to complete the

CSI components	clarification
	creation and deletion of the snapshot on the storage side.
csi-snapshot-controller	When creating/deleting a VolumeSnapshot, listen for objects in the Kubernetes API for VolumeSnapshot and VolumeSnapshotContent and trigger csi-snapshotter to complete snapshot creation on the storage.
csi-node-driver-registrar	It is used to obtain CSI information and register the node with the kubelet through the kubelet's plug-in registration mechanism so that Kubernetes can sense the node's interface with the CeaStor storage system.

说明

For more CSI component descriptions, see the Kubernetes CSI Developer Documentation for more CSI component descriptions, see the Kubernetes CSI Developer Documentation.

2.2.2 Getting the CSI Component Package

Visit \\samba.cestc.cn\CeaStor_Version_Release\CeaStor3.2.1 path to get the CeaSotr CSI component package and MD5 checksum file. samba server access user name and password, please contact technical service to get.

表 2-6 Preparing the package

Package Name	clarification
CeaStor-CSI-3.2.1-{xxxx}-{yyyyMMdd}.tar.gz	CeaStor CSI component package.

2.2.3 Uploading CSI component packages

pre-conditions

An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.

procedure

(1) Log in to any master node of the Kubernetes cluster via the management IP address.

(2) Upload the CSI component package.

Upload the CSI component package to any directory of the node that requires the CSI plug-in.

(3) Upload and import the CSI component package to the node.

2.3 Preparing the configuration file

2.3.1 Preparing the CSI Configuration File

In order to support more value-added functions, before installing the CeaStor CSI plugin, you can set the parameters in yaml separately depending on the configuration protocol.

- nvmf protocol: set csi-nvmf-controller.yaml or csi-nvmf-node.yaml neutral parameter.
- iscsi protocol: set csi-iscsi-controller.yaml or csi-iscsi-node.yaml neutral parameter.

⚠ 注意

Do not modify other parameters. If you need to modify the yaml file configuration, please contact technical support.

表 2-7 CSI Plugin Startup Parameter Description

parameters	Parameter description
LogLevel	Log Level. Fillable from 1 to 5, default value is 4.
EnableRWXExpendOnline	Whether to allow RWX-hosted PVCs to support online expansion. The default value is false.

2.3.2 Preparing a PVC Configuration File

2.3.2.1 General configuration parameters

summary

PVC resource yaml configuration file preparation and parameter description.

Parameter description

The generic configuration parameters in the yaml file are described below:

表 2-1 Description of public parameters

parameters	Parameter description
apiVersion	The version of the Kubernetes API used by the YAML description file. The default value is **storage.k8s.io/v1**.
kind	Kubernetes object resource type.
metadata.name	(Required) The name of the customized resource object.
metadata.namespace	(Required) A customized resource object namespace.
spec	Pod Specs.

2.3.2.2 secret.yaml

summary

Configure the Secret object through the secret.yaml file to hold sensitive data, i.e., information such as administrative IP, username, and password for logging into the CeaStor storage system.

Parameter description

The description of the parameter configuration in the secret.yaml file is as follows:

表 2-2 Secret Configuration Parameter Description

parameters	Parameter description
endpoint	(Required) Management IP of the storage system.
userName	Access account for the CeaStor storage system.
userkey	The account password for the CeaStor storage system.

Configuration example

A sample configuration of the secret.yaml file is shown below:

```
apiVersion: v1
kind: Secret
metadata.
  name: csi-cbd-secret-user
  namespace: product-storage
stringData.
  # endpoint: https://${vip}
  endpoint: https://xxx.xxx.xxx.xxx
  # userName: <ID>
  userName: admin
  # userkey: <PASSWORD>
  userkey: admin@123
```

2.3.2.3 storageclass.yaml

summary

The StorageClass resource object is configured through the storageclass.yaml file and is used to define the storage resource as some kind of class (Class), which is used to mark the characteristics and performance of the storage resource.

Parameter description

The parameter configuration in the storageclass.yaml file is described below:

表 2-3 StorageClass Configuration Parameter Description

parameters	Parameter description
provisioner	(Required) Storage Resource Provider, Backend Storage Driver. The default value is **cds.csi.nvmf.com**.
PoolUuid	(Required) The UUID of the hard disk pool for the CeaStor storage system.
TargetName	The name of the access path. ● Can be left empty, defaults to InnerCSITarget or InnerCSITarget2. ● You can set the name of an existing access path to the CeaStor storage system. 说明 If Protocol takes the value iscsi, it must be set to a path that is already accessible by the CeaStor storage system. If you set an access path to a CeaStor storage system that already exists, make sure that the access path is associated with at least two gateway nodes. The system supports automatic association of clients and storage volumes, and the set access path does not need to be associated with a mapping relationship.
Protocol	Access path protocol type. ● Can be left empty, defaults to nvmeof. ● Optional values nvmeof or iscsi.
RedundancyMode	Storage volume redundancy mode. ● Can be left blank, default value is **RP_3GX** (3 copies). ● Redundancy policies can be configured, see To obtain the redundancy policies supported by the disk pool for the redundancy policies supported by the hard disk pool.
CapacityMode	Storage volume capacity type. ● Can be left empty and defaults to Thin. ● Optional values Thin (thin configuration) or Thick (thick configuration).
isFlatten	Whether the clone volume splits or not. ● Can be set to null or a value other than NO_Flatten to take effect with the global default configuration or the isFlatten configuration in pvc.yaml. ● Optional value NO_Flatten (no splitting). 说明 The global default configuration is volume splitting. Configuration priority order pvc.yaml > storageclasss.yaml > global.
fsType	File system type. ● Can be left empty and takes the value ext4 by default. ● Optional values ext4 or xfs.
QosPolicyName	(Optional) QoS flow-limiting policy name. Configurable as an existing QoS policy for the CeaStor storage system.
WriteType	(Optional) Classify volume types based on the frequency of data overwrite writes. The storage engine optimizes data placement, garbage

parameters	Parameter description
	collection, and other mechanisms by identifying the type of writes, which helps improve block storage performance. The optional values are as follows: • High Used for high-frequency overwriting of volumes with written data. • Standard A volume used for non-high-frequency write data.
csi.storage.k8s.io/provisioner-secret-name	(Required) The name of the sercet used by the provisioner container.
csi.storage.k8s.io/provisioner-secret-namespace	(Required) The sercet namespace used by the provider container. The default value is **default**.
csi.storage.k8s.io/controller-expand-secret-name	(Required) The name of the sercet used by the controller container.
csi.storage.k8s.io/controller-expand-secret-namespace	(Required) The serquet namespace used by the controller container. The default value is **default**.
csi.storage.k8s.io/node-stage-secret-name	(Required) The name of the sercet used by the node-stage container.
csi.storage.k8s.io/node-stage-secret-namespace	(Required) The sercet namespace used by the node-stage container. The default value is **default**.
csi.storage.k8s.io/node-publish-secret-name	(Required) Name of the sercet used by publish in the mount PVC process.
csi.storage.k8s.io/node-publish-secret-namespace	(Required) Mount the sercet namespace used by publish in the PVC process. The default value is **default**.
csi.storage.k8s.io/controller-publish-secret-name	(Required) Name of the sercet used by publish in the mount PVC process.
csi.storage.k8s.io/controller-publish-secret-namespace	(Required) Mount the serquet namespace used by publish in the PVC process. The default value is **default**.
reclaimPolicy	(Required) The recycling strategy. The default value is is **Delete**. The optional values are as follows: • **Retain** indicates that a persistent volume retains its volume and data contents after it is released from the persistent volume statement. • **Delete** indicates the complete deletion of the underlying storage resource.
allowVolumeExpansion	Whether to run volume expansion. • The default value is true. • Optional values true or false.
volumeBindingMode	The time at which the PV is dynamically assigned and bound. The optional values are as follows: • **Immediate** indicates that dynamic allocation and binding of PVs is completed as soon as the PVC is created. • **WaitForFirstConsumer** indicates that the allocation and binding of the PV will be delayed until the Pod using this PVC is created.

Configuration example

A sample storageclass.yaml file configuration is shown below:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata.
  name: cds-csi-nvmf-sc-user
Provider: cds.csi.nvmf.com
parameters.
  # The following example customizes the access path and client group, or the built-in
access path and built-in client group if not set.
  TargetName: "tgt1"
  Protocal: "nvmeof"
  RedundancyMode: "RP_3GX"
  CapacityMode: "Thin"
  PoolUuid: "ed01c454-6ffb-4dcf-926a-a8644f766e1c"
  fsType: ""
  isFlatten: "false"
  csi.storage.k8s.io/provisioner-secret-name: csi-cbd-secret-user
  csi.storage.k8s.io/provisioner-secret-namespace: product-storage
  csi.storage.k8s.io/controller-expand-secret-name: csi-cbd-secret-user
  csi.storage.k8s.io/controller-expand-secret-namespace: product-storage
  csi.storage.k8s.io/node-stage-secret-name: csi-cbd-secret-user
  csi.storage.k8s.io/node-stage-secret-namespace: product-storage
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

2.3.2.4 pvc.yaml

summary

PVC objects are configured through the pvc.yaml file, i.e. dynamic volume creation. Dynamic volume creation does not require the creation of PVs in advance, it automatically creates the resources needed for PVCs on the storage backend according to StorageClass, and creates PVs at the same time as PVCs are created.

Parameter description

The parameter configuration in the pvc.yaml file is described below:

表 2-4 PVC Configuration Parameter Description

parameters	Parameter description
accessModes	The PVC access mode, i.e., the user application's access rights to the storage resource, can be selected with the following values: • ReadWriteOnce (RWO) indicates read and write permissions and is only allowed to be mounted to a single node. • ReadOnlyMany (ROX) indicates read-only permissions, allowing to be

parameters	Parameter description
	mounted to multiple nodes. • ReadWriteMany (RWX) denotes read and write permissions that allow being mounted to multiple nodes. Only Block volume mode supports RWX mode.
annotations	PVC annotation to configure advanced characteristics of the volume. Configurable parameter entries include capacityMode, redundancyMode, qosIOType, qosMaxIOPS, qosMaxBPS, isFlatten, writeType . 说明 If the configuration conflicts with the configuration in storageclass.yaml, the configuration in pvc.yaml has higher priority.
annotations.capacityMode	Volume capacity type. • Can be left empty and defaults to Thin. • Optional values Thin (thin configuration) or Thick (thick configuration).
annotations.redundancyMode	Volume Redundancy Policy. • Can be left blank, default value is RP_3GX (3 copies). • Redundancy policies can be configured, see To obtain the redundancy policies supported by the disk pool for the redundancy policies supported by the hard disk pool.
annotations.qosIOType	Volume QoS flow limiting type. Optional rw, w, r. • rw indicates read/write current limiting. • w denotes read current limit. • r denotes write current limit.
annotations.qosMaxIOPS	Stream-limited maximum IOPS. A value of 0 means no limit. 说明 Effective only if qosIOType is also configured.
annotations.qosMaxBPS	Limit the maximum bandwidth in bps. A value of 0 means no limit. 说明 Effective only if qosIOType is also configured.
annotations.isFlatten	Whether the clone volume splits or not. • Can be left empty or take a value other than NO_Flatten to take effect with the global default configuration or the isFlatten configuration in storageclasss.yaml. • Optional value NO_Flatten (no splitting). 说明 The global default configuration is volume splitting. Configuration priority order pvc.yaml > storageclasss.yaml > global.
annotations.writeType	(Optional) Classify volume types based on the frequency of data overwrite writes. The storage engine optimizes data placement, garbage collection, and other mechanisms by identifying the type of writes, which helps improve block storage performance. The optional values are as follows: • High Used for high-frequency overwriting of volumes with written data.

parameters	Parameter description
	• Standard A volume used for non-high-frequency write data.
storageClassName	StorageClass resource object name.
volumeMode	Volume mode, two volume modes are supported. • Can be left empty, the default value is Filesystem. • Filesystem indicates that the volume will be mounted to a directory by the Pod. If the storage for the volume comes from a device that is currently empty, Kuberneretes creates the filesystem on the device before mounting the volume for the first time. • Block indicates that the volume will be used as a raw block device.
resources.requests.storage	The capacity of the persistent volume requested by the PVC. Example values 3Gi.

Configuration example

A sample pvc.yaml file configuration is shown below:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata.
  name: pvc-test
  namespace: product-storage
  annotations.
      capacityMode: Thin
      redundancyMode: RP_3GX
      qosIOType: rw
      qosMaxIOPS: 0
      qosMaxBPS: 0
      isFlatten.
spec.
  accessModes.
  - ReadOnlyMany
storageClassName: cds-csi-nvmf-sc
  resources.
      requests.
                storage: 1Gi
```

2.4 Preparation of relevant tools

You can refer to the following table to prepare the required tools and software.

表 2-5 Preparation of software tools

name (of a thing)	clarification
SSH Connection Tool	Used to log in to the node by SSH to execute configuration commands.
WinSCP	A file transfer tool for use between Windows and Linux systems. You can find it on the WinSCP website to download the corresponding version of the tool.

name (of a thing)	clarification
	It is recommended that you use version 5.7.5 or higher and choose the SCP protocol when transferring files.

3 Installing CSI

3.1 Install and configure CSI

summary

In order to achieve the purpose of the CeaStor storage system to provide persistent volume storage capability to Kubernetes, you need to install the CeaStor CSI image on the master and worker nodes of the Kubernetes cluster respectively, and you need to deploy the CeaStor on-premises yaml file.

pre-conditions

- Kubernetes has been confirmed to be version 1.24 and above.
- An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.
- The CeaStor CSI plug-in component package has been obtained, please contact technical support to obtain it.
- Nodes for which CSI is required for Kubernetes clusters have been planned and assigned master or worker roles. For details, see Assigning 8K8s Cluster for details.

procedure

(1) Log in (you can use a tool such as kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) Upload the CSI component package to.

Upload the CSI component package to any directory on the master node.

(3) Upload and import the CSI component package to the node.

a. In the CSI component package path, go to the images/xxx directory (xxx indicates the image type).

b. Load the CSI component tarball. For detailed instructions, please refer to Load CSI image package for detailed instructions.

(4) Create the namespace.

Execute the **kubectl create namespace** <*ns*> command to create namespaces named ccos-cluster-storage-operator and product-storage.

说明

ccos-cluster-storage-operator and product-storage specify namespaces for CeaStor CSI that must be created and are not supported for modification. If you have questions, please contact technical support.

```
$ kubectl create namespace ccos-cluster-storage-operator
$ kubectl create namespace product-storage
```

(5) Deploy the on-premises yaml file.

a. In the CSI component package path, go to the deploy/kubernetes directory.

b. Execute the **kubectl apply -f** <*yaml_name*>**.yaml** command and, depending on the configured protocol type, deploy in order 2.2.1 The yaml configuration file is pre-populated with yaml files.

说明

CeaStor CSI provides preconfigured yaml file, please do not modify it. If you need to modify the yaml file configuration, please contact technical support.

Take the example deployment of the csi-nvmf-driver.yaml file as follows:

```
$ kubectl apply -f csi-nvmf-driver.yaml
csidriver.storage.k8s.io/cds.csi.nvmf.com created
```

c. Execute the command to query the yaml file deployment result. For a detailed example of a successful deployment, see To query the CSI on-premises yaml file deployment result for a detailed example of a successful deployment, see

(6) Check that the CSI service is started.

a. Execute the **kubectl get csidriver** command to view the CSI Driver. the CSI Driver is deployed successfully if the following message is displayed.

```
$ kubectl get csidriver
NAME ATTACHREQUIRED PODINFOONMOUNT STORAGECAPACITY TOKENREQUESTS
REQUIRESREPUBLISH MODES AGE
cds.csi.nvmf.com true true false <unset> false Persistent 5d23h
```

b. Execute the **kubectl get pod -A | grep csi** command to check the pod status. If the status of all pods shows Running, the CSI service has started successfully.

```
$ kubectl get pod -A | grep csi
NAMESPACE NAME READY STATUS RESTARTS AGE
ccos-cluster-storage-operator csi-snapshot-controller-684774fcbb-58hn2 1/1
Running 0 172m
product-storage cds-csi-nvmf-lqlxg 6/6 Running 0 174m
product-storage csi-nvmf-node-w47fc 2/2 Running 0 170m
product-storage csi-nvmf-node-wz64s 2/2 Terminating 0 3d3h
```

Among them, csi-snapshot-controller-xxxx contains 1 container, cds-csi-nvmf-xxxx contains 6 containers, csi-nvmf-node-xxx contains 2 containers, the containers and their descriptions under each Pod are as follows:

表 3-2 Description of CSI Service Pods and their containers

Pod	Container	descriptive
csi-snapshot-controller-xxxx	csi-snapshot-controller	Responsible for snapshot control, deployed on master.
cds-csi-nvmf-xxxx cds-csi-iscsi-xxxx	node-registrar	Plugin Registration. All nodes started by the plugin need to be started.
	csi-provisioner	Responsible for managing the creation and deletion of PVs, deployed on master.
	csi-attacher	Responsible for posting PVs out to be deployed on the master.
	csi-resizer	Responsible for managing the expansion of PVs, deployed on the master.
	csi-snapshotter	Responsible for managing snapshot creation of PVs, deployed on master.
	csi-nvmf-plugin	CSI Driver. All nodes that are to support CSI mounts need to be started.
csi-nvmf-node-xxx csi-iscsi-node-xxx	node-registrar	Used to get CSI information, nodes can be registered into kubelet through the kubelet plugin registration mechanism. Any node started by the plugin needs to be started.
	csi-nvmf-plugin	Plugin driver for backend NVMe or iSCSI disks. All nodes that are to support CSI mounts need to be booted.

3.2 Configure and deploy PVCs

brief introduction

After successfully installing the CeaStor CSI plug-in in a Kubernetes cluster, you dock the CeaStor storage system by configuring and deploying a resource file and dynamically create PVs. during the processing of storage operations, you need to persist the data in the containers to the hard disk so that you can continue to use the persistent upgrades when the containers are rebuilt or assigned to a new node.

- Configure the Secret object through the secret.yaml file to hold sensitive data, i.e., information such as administrative IP, username, and password for logging into the CeaStor storage system.
- The StorageClass resource object is configured through the storageclass.yaml file and is used to define the storage resource as some kind of class (Class), which is used to mark the characteristics and performance of the storage resource.
- PVC objects are configured through the pvc.yaml file, i.e. dynamic volume creation. Dynamic volume creation does not require the creation of PVs in advance, it automatically creates the resources needed for PVCs on the storage backend according to StorageClass, and creates PVs at the same time as PVCs are created.

pre-conditions

- The CSI plugin has been installed in the Kubernetes cluster.
- Configuration of connectivity between the CeaStor storage system and the Kubernetes cluster has been completed.
- The Kubectl tool has been installed in the Kubernetes cluster.
- An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.

procedure

(1) Log in (you can use a tool such as Kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) Customize the creation of PVC configuration yaml files.

In the example/kubernetes directory of the CSI component package, execute the **vi** *<yaml_name>***.yaml** command to set up the <*yaml_name*>.yaml file.

References Prepare the yaml configuration files , edit the secret.yaml, storageclass.yaml, and pvc.yaml files in turn.

(3) Deploy and query the Secret object.

a. Execute the **kubectl apply -f secret.yaml** command to create the Secret object.

```
$ kubectl apply -f secret.yaml
secret/csi-cbd-secret-user created
```

b. Execute the **kubectl get secret csi-cbd-secret-user** command to see if the Secret object was created successfully. If the following message is displayed, the secret was created successfully.

```
$ kubectl get secret csi-cbd-secret-user
NAME TYPE DATA AGE
csi-cbd-secret-user Opaque 3 46h
```

(4) Deploy and query the StorageClass object.

a. Execute the **kubectl apply -f storageclass.yaml** command to create the StorageClass object.

```
$ kubectl apply -f storageclass.yaml
storageclass.storage.k8s.io/cds-csi-nvmf-sc created
```

b. Execute the **kubectl get sc -A | grep csi** command to see if the StorageClass object was created successfully. If the following message is displayed, the object was created successfully.

```
$ kubectl get sc -A | grep csi
cds-csi-nvmf-sc-user1 cds.csi.nvmf.com Delete Immediate true 24h
```

(5) Create and query PVC objects.

a. Execute the **kubectl apply -f pvc.yaml** command to create the PVC object.

```
$ kubectl apply -f pvc.yaml
persistentvolumeclaim/pvc-test
```

b. Execute the **kubectl get pvc -A** command to see if the PVC object was created successfully. If the following message is displayed, the PVC object was created successfully.

```
$ kubectl get pvc -A
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
product-storage pvc-test Bound pvc-xxx 1Gi RWO cds-csi-nvmf-sc 47m
```

follow-up operation

Once the PVC object is successfully created, the dynamic volume creation configuration is successful, and you can use CeaStor storage resources.CeaStor CSI also supports features such as cloning PVCs, expanding PVCs, creating snapshots, and cloning volume snapshots. For detailed configuration instructions, see Using CSI .

3.3 Enabling NVMe/iSCSI Services

3.3.1 Configuring the NVMe Initiator

summary

In a Kubernetes cluster, load the NVMe initiator for all nodes that need to use CSI.

pre-conditions

- Kubernetes has been confirmed to be version 1.24 and above.
- An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.
- Already in the Kubernetes cluster Install the nvme-cli tool.

procedure

(1) Log in (you can use a tool such as kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) Execute the following command to load the kernel NVMe TCP and NVMe RDMA driver modules.

 - If it is an NVMe TCP type, execute the **modprobe nvme-tcp** command.
 - If it is an NVMe RDMA type, execute the **modprobe nvme-rdma** command.

(3) Execute the **lsmod | grep nvme** command to see if the driver is loaded successfully. An example is shown below:

```
$ lsmod | grep nvme
nvme_rdma 32768 0
nvme_tcp 32768 0
nvme_fabrics 24576 2 nvme_tcp,nvme_rdma
```

```
nvme_core 114688 3 nvme_tcp,nvme_rdma,nvme_fabrics
```

(4) Execute the following command to write the NVMe driver configuration to the system modules.conf configuration file with persistent modifications (the NVMe driver can be loaded automatically after a system reboot).

```
$ echo "nvme-tcp" >> /etc/modules-load.d/modules.conf
$ echo "nvme-rdma" >> /etc/modules-load.d/modules.conf
$ cat /etc/modules-load.d/modules.conf
nvme-tcp
nvme-rdma
```

3.3.2 Configuring the iSCSI Initiator

summary

In a Kubernetes cluster, load the iSCSI initiator for all nodes that need to use CSI.

pre-conditions

- Kubernetes has been confirmed to be version 1.24 and above.
- An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.

procedure

(1) Log in to the Linux client.

(2) Execute the **rpm -q iscsi-initiator-utils** command to query whether the iSCSI client package is installed.

```
$ rpm -q iscsi-initiator-utils
iscsi-initiator-utils-6.2.1.4-2.git2a8f9d8.cl9.x86_64
```

(3) Execute the **systemctl start iscsid.service** command to turn on the iSCSI service.

(4) Execute the **cat /etc/iscsi/initiatorname.iscsi** command to query the client IQN.

```
$ cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:a72fffd3726
```

3.4 Enable Multipath

3.4.1 Enable multipath access (NVMe-oF)

summary

If you enable multipathing in the kernel on a Kubernetes node, the node can choose the optimal path to access the storage volume, which improves I/O access rates and increases reliability.

This subsection describes how to configure NVMe-oF multipath access in a Kubernetes cluster, using the Centos 8.4 version of the operating system as an example.

pre-conditions

- NVMe-oF access type has been configured and multiple storage nodes have been configured.
- An account and password with Kubernetes cluster administrator privileges have been obtained.

procedure

(1) Log in (you can use a tool such as Kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) Execute the command **cat /sys/module/nvme_core/parameters/multipath** to check if native NVMe multipath is enabled in the kernel.

- If N is echoed back to indicate that native NVMe multipathing is disabled, execute (3) .
- If Y is displayed back, it indicates that native NVMe multipathing is enabled.

(3) Enable native NVMe multipathing.

a. Configure the **/etc/modprobe.d/nvme_core.conf** file by adding the **options nvme_core multipath=Y** field.

b. Execute the following commands in sequence to back up the initramfs file system.

```
$ cp /boot/initramfs-$(uname -r).img \
$ cp /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

c. Execute the following command to rebuild the initramfs file system.

```
$ dracut --force -verbose
```

d. Reboot the system.

(4) (Optional) On a running system, you can execute the following command to change the I/O policy in the NVMe device to distribute I/O across all available paths.

```
$ echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

The default I/O policy is "numa" mode, and the meaning of each mode is as follows:

- numa mode: IO downstreaming goes to one gateway by default, and automatically switches to other gateways when the current working gateway fails.
- round-robin mode: IO downstreaming uses polling and is evenly distributed to each gateway.

3.4.2 Enable Multipath Access (iSCSI)

summary

If you enable multipathing in the kernel on a Kubernetes node, the node can choose the optimal path to access the storage volume, which improves I/O access rates and increases reliability.

This subsection describes how to configure iSCSI multipath access in a Kubernetes cluster.

pre-conditions

- The iSCSI access type has been configured and multiple storage nodes have been configured.
- An account and password with Kubernetes cluster administrator privileges have been obtained.

procedure

(1) Log in (you can use a tool such as Kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) Check if the cluster has the multipath installer multipath installed.

```
$ rpm -qa | grep device-mapper-multipath
```

(3) Copy the file **multipath.conf** file to the /etc directory.

```
$ cp /usr/share/doc/device-mapper-multipath/multipath.conf /etc/multipath.conf
```

(4) Enable multipathing.

```
$ systemctl start multipathd
```

(5) Configure multipathing to boot with the system.

```
$ systemctl enable multipathd.service
```

(6) Execute the following commands in sequence to take effect the configuration.

```
$ multipath --F
$ systemctl restart multipath.service
$ systemctl reload multipathd.service
$ multipath -v3
```

(7) Execute the **multipath -ll** command to verify that the configuration takes effect.

4 Use of CSI

4.1 Management of PVC

4.1.1 Dynamic Volume Creation (PVC)

summary

Dynamically created volumes do not need to create PVs beforehand, they automatically create the resources needed for PVCs on the storage backend according to StorageClass, and can create PVs at the same time as PVCs.

Using PVC requires the StorageClass resource object, which defines the storage resource as a certain class (Class) and is used to mark the characteristics and performance of the storage resource. It is also necessary to configure the Secret object, which is used to save sensitive data, i.e., information such as usernames, passwords, etc., for logging in to the CeaStor back-end storage.

procedure

Dynamically creating a volume is divided into the following steps:

(1) Configuring Secret

(2) Configure StorageClass

(3) Configuring PVC

References Preparing the yaml configuration files to create and deploy the secret.yaml, storageclass.yaml, and pvc.yaml files in order.

4.1.2 Clone rolls (PVC)

summary

In Kubernetes, cloning a PVC means creating a clone volume.

Configuration

Clone the storage volume in Kubernetes by configuring and executing the clone.yaml file. the parameter configuration in the clone.yaml file is described below:

表 4-1 Description of clone.yaml file parameters

parameters	Parameter description
storageClassName	The name of the StorageClass resource object used. Same value as the storageClassName of the source PVC.
dataSource.name	(Required) The name of the source PVC being cloned.
dataSource.kind	The category to which the cloned source PVC belongs.

parameters	Parameter description
	The default value is **PersistentVolumeClaim**.
accessModes	Clone volume access mode. The optional values are as follows: • ReadWriteOnce (RWO) indicates read and write permissions and is only allowed to be mounted to a single node. • ReadOnlyMany (ROX) indicates read-only permissions, allowing to be mounted to multiple nodes. • ReadWriteMany (RWX) indicates read/write access and allows being mounted to multiple nodes. Only Block volume mode supports RWX access.
resources.requests.storage	Clone volume capacity, with an example value of 1Gi. Needs to be greater than or equal to the source PVC volume capacity.

Configuration example

A sample clone.yaml file configuration is shown below:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata.
  name: clone-example
  namespace: product-storage
spec.
  accessModes.
      - ReadOnlyMany
dataSource.
      name: pvc-example
      kind: PersistentVolumeClaim
  storageClassName: cds-csi-nvmf-sc
  resources.
      requests.
          storage: 1Gi
```

Execute and query cloned volumes

```
$ kubectl apply -f clone.yaml
persistentvolumeclaim/clone-example created
$ kubectl get pvc -A
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
product-storage clone1 Bound pvc-xxx 1Gi RWO cds-csi-nvmf-sc 48m
```

4.1.3 Delete Volume (PVC)

summary

In Kubernetes, the resources required for PVCs are automatically created on the storage backend based on the StorageClass, and PVs can be created at the same time as PVCs are created. where the reclaimPolicy in the StorageClass can be configured for Delete and Retain. The two different reclaim policies have some differences when deleting PVCs.

procedure

(1) Query the source PVC recovery policy. Take the volume with PVC name pvc-test-01/pvc-test02 as an example.

```
$ kubectl get pv | grep pvc-test
name capacit y access modes reclaim polity status claim storageclass reason age
pvc-a63fe993-1541-487e-b76b-07dbc519006c 10Mi RWO Retain Bound product-storage/pvc-test-02 sc-example02 22s
pvc-baa969c7-48e8-47ab-804b-cb7350892dea 10Mi RWO Delete Bound product-storage/pvc-test-01 sc-example 2m30s
```

(2) Delete PVC.

(3) If Reclaim Policy is Delete.

```
$ kubectl delete -f pvc-test-01.yaml
persistentvolumeclaim "pvc-test-01" deleted
```

o If Reclaim Policy is Retain.

```
$ kubectl delete -f pvc-test-02.yaml
persistentvolumeclaim "pvc-test-02" deleted

$ kubectl get pv | grep pvc-test
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-a63fe993-1541-487e-b76b-07dbc519006c 10Mi RWO Retain Released product-storage/pvc-test-02 sc-example02 8m22s

$ kubectl delete pv pvc-a63fe993-1541-487e-b76b-07dbc519006c
persistentvolume "pvc-a63fe993-1541-487e-b76b-07dbc519006c" deleted
```

4.1.4 Expansion rolls (PVC)

summary

In Kubernetes, scaling PVCs means scaling storage volumes.

caveat

- Only expansion is supported not reduction.
- ROX and RWX access modes do not support online expansion.

procedure

(1) Query the source PVC. take the volume with the expansion name pvc-test as an example.

```
$ kubectl get pvc -A | grep pvc-test
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
```

```
product-storage pvc-test Bound pvc-2171141e-9b48-45af-a31a-eb712d801bdb 2Gi RWO
cds-csi-nvmf-sc-user 48s
```

(2) Expansion of PVC. take the example of expansion from 2Gi to 3Gi.

```
$ kubectl patch pvc pvc-test -p '{"spec":{"resources":{"requests":{"storage":
"3Gi"}}}}' -n product-storage
persistentvolumeclaim/pvc-test patched
$ kubectl get pvc -A | grep pvc-test
product-storage pvc-test Bound pvc-2171141e-9b48-45af-a31a-eb712d801bdb 2Gi RWO
cds-csi-nvmf-sc-user 3m36s
$ kubectl get pv -A | grep pvc-test
pvc-2171141e-9b48-45af-a31a-eb712d801bdb 3Gi RWO Delete Bound product-storage/pvc-
test cds-csi-nvmf-sc-user 4m59s
```

(3) (Optional) Viewing PVCs after expansion does not show the expanded capacity immediately, because PVCs need to be mounted on a Pod before showing the expanded capacity. Take the example of configuring the pod.yaml file and creating a Pod to mount the PVC.

```
apiVersion: v1
kind: Pod
metadata.
  name: busybox
  namespace: product-storage
spec.
  containers.
  - name: busybox
    image: busybox
    command.
      - sleep
      - "3600."
    volumeMounts.
    - mountPath: "/busybox-v-data"
      name: busybox-v
  volumes.
  - name: busybox-v
    persistentVolumeClaim.
      claimName: pvc-test
```

After the Pod is successfully mounted, query the PVC capacity to see that the capacity has been changed to the expected 3Gi.

```
$ kubectl get po -A | grep busybox
product-storage busybox 1/1 Running 0 11m
$ kubectl get pvc -A | grep pvc-test
product-storage pvc-test Bound pvc-2171141e-9b48-45af-a31a-eb712d801bdb 3Gi RWO
cds-csi-nvmf-sc-user 3m36s
```

4.1.5 Mounted PVC

summary

After PVC configuration and deployment is complete, you need to set up pod.yaml or pod-block.yaml to set up the PVC mount path, i.e., mount volume, based on the different volume modes of volumeMode before using and expanding the PVC, respectively.

Configuration

The description of the parameter configuration in the pod.yaml file is as follows:

表 4-2 Description of pod.yaml configuration parameters

parameters	Parameter description
volumes.name	Volume Name.
volumes.persistentVolumeClaim.claimName	The name of the PVC to which the volume corresponds.
volumeMounts.name	If the volumeMode value is **Filesystem.** Configure the name of the volume to be mounted, the same as **volumes.name**.
volumeMounts.mountPath	If the volumeMode value is **Filesystem.** Configure the mount path in the Pod for the volume to be mounted.

The description of the parameter configuration in the pod-block.yaml file is as follows:

表 4-3 Description of pod-block.yaml configuration parameters

parameters	Parameter description
volumes.name	Volume Name.
volumes.persistentVolumeClaim.claimName	The name of the PVC to which the volume corresponds.
volumeDevices.name	If the volumeMode value is **Block.** Configure the block device name, the same as **volumes.name**
volumeDevices. devicePath	If the volumeMode value is **Block.** Configure the path of the block device in the Pod.

Configuration example

If the volumeMode of the PVC is the default value Filesystem, set the volume mount path when you add the PVC to the container.

A sample pod.yaml file configuration is shown below:

```
apiVersion: v1
kind: Pod
metadata.
  name: busybox
  namespace: product-storage
```

```
spec.
  containers.
  - name: busybox
    image: busybox
    command.
      - sleep
      - "3600."
    volumeMounts.
    - name: busybox-v
      mountPath: "/busybox-v-data"
  volumes.
  - name: busybox-v
    persistentVolumeClaim.
      claimName: pvc-test
```

If the volumeMode of the PVC is Block, set the block device path when the PVC is added to the container.

A sample pod-block.yaml file configuration is shown below:

```
apiVersion: v1
kind: Pod
metadata.
  name: busybox-block
  namespace: product-storage
spec.
  containers.
  - name: busybox
    image: busybox:latest
    command.
      - sleep
      - "3600."
    volumeDevices.
    - name: busybox-v
      devicePath: "/busybox-dev"
  volumes.
  - name: busybox-v
    persistentVolumeClaim.
      ClaimName: pvc-block
```

Deploy and view deployment results

```
$ kubectl apply -f pod.yaml
pod/busybox created
$ kubectl get pod -A
NAMESPACE NAME READY STATUS RESTARTS AGE
product-storage busybox 1/1 Running 0 33s
```

4.1.6 Unloading PVC

summary

In Kubernetes, unbinding a PVC from a Pod is unmounting the volume.

Configuration

- If the PVC's volumeMode is the default value of Filesystem:
  Remove the mount PVC name-related configuration from the pod.yaml configuration file, including: volumes.name, volumes.persistentVolumeClaim.claimName, volumeMounts.name, volumeMounts.mountPath.
- If the volumeMode of the PVC is Block
  Remove the mount PVC name-related configuration from the pod-block.yaml configuration file, including volumes.name, volumes.persistentVolumeClaim.claimName, volumeDevices.name, volumeDevices. devicePath.

Configuration example

If the volumeMode of the PVC is the default value Filesystem, unmount the volume mount path when the PVC is deleted from the container.

A sample pod.yaml file configuration is shown below:

```
apiVersion: apps/v1
kind: Deployment
metadata.
  name: nginx-test-nvmf2
  namespace: product-storage
  labels.
    app: nginx
spec.
  replicas: 1
  selector.
    matchLabels.
      app: nginx
  template.
    metadata.
      labels.
        app: nginx
    spec.
      containers.
        - name: nginx
          image: localhost/nginx
          # Filesystem PVC
          volumeMounts.
      volumes.
```

Implementation steps

```
$ kubectl apply -f pod.yaml
pod/nginx created
```

4.2 Managing VolumeSnapshot

4.2.1 Creating a Volume Snapshot

summary

In Kubernetes, a VolumeSnapshot is a snapshot of a volume on a storage system.

VolumeSnapshot provides a standard way for Kubernetes to replicate volumes at a specified point in time without creating new volumes, available data backups. To create a VolumeSnapshot object, you need to create a VolumeSnapshotClass object first, i.e., first by configuring and deploying the volumesnapshotclass.yaml file, and then configuring and deploying the volumesnapshot-dynamic.yaml file.

- VolumeSnapshotClass provides a way to describe the storage "class" when configuring a volume snapshot, similar to StorageClass.
- The VolumeSnapshot object is the snapshot object.

yaml configuration parameters

The parameter configuration in the volumesnapshotclass.yaml file is described below:

表 4-4 Parameter description of volumesnapshotclass.yaml file

parameters	Parameter description
annotations. snapshot.storage.kubernetes.io/is-default-class	A default VolumeSnapshotClass can be specified for VolumeSnapshots that do not request any specific class binding. The default value is true.
driver	CSI drive name. Takes the value of the **metadata.name** parameter in csi-nvmf-driver.yaml.
deletionPolicy	(Required) Type of deletion policy. Optional values Delete or Retain: • Delete indicates that the underlying storage snapshot is deleted along with the VolumeSnapshotContent object. • Retain indicates that both the underlying snapshot and the VolumeSnapshotContent object are retained.
PoolUuid	(Required) Storage system hard disk pool UUID. Takes the value of the **PoolUuid** parameter in storageclass.yaml.
csi.storage.k8s.io/snapshotter-secret-name	(Required) The name of the sercet used by the snapshotter container. The default value is **csi-cbd-secret-user**.
csi.storage.k8s.io/snapshotter-secret-namespace	(Required) The sercet namespace used by the snapshotter container. The default value is **product-storage**.

The parameter configuration in the volumesnapshot-dynamic.yaml file is described below:

表 4-5 Parameter description of volumesnapshot-dynamic.yaml file

parameters	Parameter description
volumeSnapshotClassName	(Required) The name of the VolumeSnapshotClass object used. The default value is **csi-cds-snap**.
source. persistentVolumeClaimName	(Required) Source PVC object name.

A sample configuration is shown below:

Volumesnapshotclass.yaml file

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata.
  name: snapshot1
  namespace: product-storage
spec.
  volumeSnapshotClassName: csi-cds-snap
  source.
      persistentVolumeClaimName: pvc-test
```

The volumesnapshot-dynamic.yaml file

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata.
  name: volumesnapshot-test
  namespace: product-storage
spec.
  source.
    persistentVolumeClaimName: pvc-test-01
  volumeSnapshotClassName: csi-cds-snap
```

Deploy and query volume snapshots

Deploy the VolumeSnapshotClass object.

```
$ kubectl apply -f volumesnapshotclass.yaml
volumesnapshotclass.snapshot.storage.k8s.io/csi-cds-snap created
```

Deploy and query volume snapshot objects.

```
$ kubectl apply -f volumesnapshot-dynamic.yaml
volumesnapshot.snapshot.storage.k8s.io/snapshot1 created
$ kubectl get VolumeSnapshot -n product-storage
NAME READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT RESTORESIZE SNAPSHOTCLASS
SNAPSHOTCONTENT CREATIONTIME AGE
```

```
snapshot1 true pvc1 1Gi csi-cds-snap snapcontent-4bfb12df-dd8e-4fe4-aaf3-3813d88e67f6
22d 22d 22d
```

4.2.2 snapshot of a clone volume

summary

In Kubernetes, snapshot cloning is similar to cloning PVCs and is achieved by configuring and deploying the restore.yaml file.

Configuration

The parameter configuration in the restore.yaml file is described below:

表 4-6 Description of the parameters of the restore.yaml file

parameters	Parameter description
storageClassName	The name of the StorageClass resource object used.
dataSource.name	(Required) The name of the snapshot object being cloned.
dataSource.kind	The category to which the cloned snapshot object belongs. The default value is **VolumeSnapshot**.
dataSource.apiGroup	(Required) The apiGroup to which the cloned snapshot object belongs. The fixed value is snapshot**.storage.k8s.io**.
accessModes	Clone volume snapshot access mode. The optional values are as follows: • ReadWriteOnce (RWO) indicates read and write permissions and is only allowed to be mounted to a single node. • ReadOnlyMany (ROX) indicates read-only permissions, allowing to be mounted to multiple nodes. • ReadWriteMany (RWX) indicates read/write access and allows being mounted to multiple nodes. Only Block volume mode supports RWX access.
resources.requests.storage	Clone volume snapshot capacity, with an example value of 1Gi. Needs to be greater than or equal to the source volume snapshot capacity.

Deploy and query cloned volume snapshots

```
$ kubectl apply -f clone.yaml
persistentvolumeclaim/restore-snapshot1 configured
$ kubectl get pvc -A
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
product-storage restore-snapshot1 Bound pvc-xxx 1Gi RWO cds-csi-nvmf-sc 48m
```

4.2.3 Deleting a Volume Snapshot

summary

In Kubernetes, deleting a snapshot is accomplished by creating the snapshot's volumesnapshotclass.yaml file.

Deleting a Volume Snapshot

```
$ kubectl delete -f volumesnapshot-dynamic.yaml
volumesnapshot.snapshot.storage.k8s.io "volumesnapshot-test" deleted
$ kubectl get pvc -A
NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
```

5 Managing CSI

5.1 Upgrade CSI

summary

A new version of the CeaStor CSI plug-in has been released, adding new features and fixing some issues. You will need to upgrade the CeaStor CSI image as a companion to the CeaStor storage system upgrade to use the new features.

pre-conditions

- An account and password have been obtained for the Kubernetes cluster management IP, with administrator privileges.
- The upgrade of the CeaStor storage system has been completed and a companion version of the CeaStor CS component package has been acquired.

procedure

(1) Log in to any master node of the Kubernetes cluster via the management IP address.

(2) Upload the CeaStor CSI component package to any directory on the master node.

(3) Upload and import the CeaStor CSI component package to the nodes.

   a. In the CSI component package path, go to the images/xxx directory (xxx indicates the image type).

   b. Select either the scripted or manual method to load the CSI component tarball. For detailed instructions, see Load CSI Image Package for detailed instructions, see Loading CSI Mirror Packages.

(4) Deploy the on-premises yaml file.

   a. In the CSI component package path, go to the deploy/kubernetes directory.

   b. Update the path to the image in the ***csi-nvmf-node*.yaml** file.

   c. Execute the **kubectl apply -f *csi-nvmf-node*.yaml** command to upgrade the CSI plugin.

(5) After checking the CSI Pod reboot, complete the CeaStor CSI plugin upgrade.

   Execute the **kubectl get pod -A | grep csi** command to check the pod status. If the status of all pods shows Running, the CSI service has been successfully upgraded.

```
$ kubectl get pod -A | grep csi
NAMESPACE NAME READY STATUS RESTARTS AGE
ccos-cluster-storage-operator csi-snapshot-controller-684774fcbb-58hn2 1/1 Running
0 172m
product-storage cds-csi-nvmf-lqlxg 6/6 Running 0 174m
product-storage csi-nvmf-node-w47fc 2/2 Running 0 170m
product-storage csi-nvmf-node-wz64s 2/2 Terminating 0 3d3h
```

5.2 Curing and viewing CSI logs

summary

Since container logs are lost after container restart. To avoid loss of CSI usage logs, you can configure the CSI plugin log path to solidify the CSI log storage path.

CSI log <podname>-xxx_<namespace>_<container name>-xxx.log, naming the log name based on its associated Pod, Container, and namespace, with the plugin having the same path definition as the external component log.

pre-conditions

An account and password with Kubernetes cluster administrator privileges have been obtained.

Curing the CSI Log Path

(1) Log in to any master node of the Kubernetes cluster via the management IP address.

(2) Configure the Log Path field.

a. Add the log path field to the CSI Plugin's corresponding container VolumeMounts.

```
- mountPath: /var/log/storage/csi
    name: csipluginlogpath
```

b. Add the Log Path field to the Volumes of the Pod where the CSI Plugin resides.

```
- hostPath.
    path: /var/log/storage/csi
      type: ""
  name: csipluginlogpath
```

(3) Create the path on the host where the CSI plugin resides.

```
$ sudo mkdir -p /var/log/storage/csi
```

View CSI Logs

- After the curing CSI log path configuration is complete, you can view the CSI logs at the specified path.

```
$ cd /var/log/storage/csi
$ vi <podname>-xxx_<namespace>_<container name>-xxx.log
```

- For scenarios where the log path is not solidified, CSI logs can be viewed directly based on CSI-related Pods, Containers, and namespaces. The command to view CSI is as follows:

```
$ cd /var/log/containers
$ vi <podname>-xxx_<namespace>_<container name>-xxx.log
```

5.3 Manual unloading of CSI

summary

If you no longer use the CSI service, you can manually uninstall the CeaStor CSI Driver by executing the command.

pre-conditions

- All storage resources created through the CSI service have been deleted, including storage volumes, volume snapshots.
- An account and password with Kubernetes cluster administrator privileges have been obtained.

procedure

(1) Log in (you can use a tool such as Kubectl) to any master node of the Kubernetes cluster via the management IP address.

(2) In the CSI component package path, go to the CSI preset yaml file directory.

(3) Execute the **kubectl delete -f** <*yaml_name*>**.yaml** command and, depending on the configured protocol type, deploy in order 2.2.1 The yaml configuration file is pre-populated with yaml files.

5.4 Query CSI on-premises yaml file deployment results

summary

After the CSI preconfigured yaml file has been deployed, you can query whether the deployment was successful with the following command.

表 5-2 The yaml file deployment results command

yaml file	Query Deployment Results Command
csi-node-rbac.yaml	kubectl get ClusterRole -A | grep csi
operand_rbac.yaml	
serviceaccount.yaml	kubectl get ServiceAccount -A | grep csi
csi-nvmf-driver.yaml csi-iscsi-driver.yaml	kubectl get CSIDriver cds.csi. <protocol>.com
snapshot.storage.k8s.io_volumesnapshotclasses.yaml	kubectl get CustomResourceDefinition -A | grep volumesnapshot
snapshot.storage.k8s.io_volumesnapshotcontents.yaml	
snapshot.storage.k8s.io_volumesnapshots.yaml	
csi_controller_deployment.yaml	kubectl get Deployment -n <namespace>
csi-nvmf-controller.yaml csi-iscsi-controller.yaml	kubectl get DaemonSet -n <namespace>
csi-nvmf-node.yaml	

yaml file	Query Deployment Results Command
csi-iscsi-node.yaml	

kubectl get ClusterRole -A | grep csi

Execute the command to query whether csi-node-rbac.yaml, operand_rbac.yaml, and other files are installed successfully, the following result indicates that the installation is successful:

```
$ kubectl get ClusterRole -A | grep csi
NAME AGE
ccos-csi-snapshot-controller-runner 26d
csi-nvmf-cr 26d
```

kubectl get ServiceAccount -A | grep csi

Execute the command to query whether the serviceaccount.yaml file is installed successfully, the following result indicates that the installation is successful:

```
$ kubectl get ServiceAccount -A | grep csi
NAMESPACE NAME SECRETS AGE
ccos-cluster-storage-operator csi-snapshot-controller 1 26d
ccos-cluster-storage-operator csi-snapshot-controller-operator 1 26d
product-storage csi-nvmf-sa 1 26d
```

kubectl get CSIDriver cds.csi.nvmf.com

Execute the command to query whether csi-nvmf-driver.yaml and other files have been installed successfully, the following result indicates that the installation is successful:

```
$ kubectl get CSIDriver cds.csi.nvmf.com
NAME CREATED AT
cds.csi.nvmf.com 2023-03-09T06:42:22Z
```

kubectl get CustomResourceDefinition -A | grep volumesnapshot

Execute the command to query snapshot.storage.k8s.io_volumesnapshotclasses.yaml, snapshot.storage.k8s.io_volumesnapshotcontents.yaml, snapshot.storage.k8s. io_volumesnapshots.yaml and other files are deployed successfully, the following result indicates that the deployment is successful:

```
$ kubectl get CustomResourceDefinition -A | grep volumesnapshot
NAME CREATED AT
volumesnapshotclasses.snapshot.storage.k8s.io 2023-03-12T07:17:13Z
volumesnapshotcontents.snapshot.storage.k8s.io 2023-03-12T07:20:53Z
volumesnapshots.snapshot.storage.k8s.io 2023-03-12T07:21:06Z
```

kubectl get Deployment -n <namespace>

Execute the command to query csi_controller_deployment.yaml and other files whether the deployment is successful, the following results show that the deployment is successful:

```
$ kubectl get Deployment -n ccos-cluster-storage-operator
NAME READY UP-TO-DATE AVAILABLE AGE
csi-snapshot-controller 1/1 1 1 45h
$ kubectl get pod -n ccos-cluster-storage-operator
name ready status restarts age
csi-snapshot-controller-64bf858cd5-9gdp8 1/1 Running 7 (44h ago) 44h
```

kubectl get DaemonSet -n <namespace>

Execute the commands to query whether csi-nvmf-controller.yaml, csi-nvmf-node.yaml and other files are successfully deployed, and the following results show that the deployment is successful:

```
$ kubectl get DaemonSet -n product-storage
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
cds-csi-nvmf 1 1 1 1 1 1 node-role.kubernetes.io/master= 4d17h
csi-nvmf-node 2 2 2 2 2 2 node-role.kubernetes.io/worker= 4d17h
$ kubectl get pod -n product-storage -owide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
cds-csi-nvmf-6rmcw 6/6 Running 0 42h 10.255.138.129 master1 <none> <none>
csi-nvmf-node-f6qtl 2/2 Running 0 42h 10.255.138.131 worker2 <none> <none>
csi-nvmf-node-ndnfn 2/2 Running 0 42h 10.255.138.130 worker1 <none> <none>
```

6 appendice

6.1 Get the redundancy policies supported by the disk pool

pre-conditions

An account and password with CeaStor Storage System Administrator privileges have been obtained.

procedure

(1) Using an account with administrator privileges, remotely log in (you can do this via putty, xshell, etc.) to any of the CeaStor cluster nodes.

```
Connecting to 10.253.219.237:22...
Connection established.
To escape to local shell,press 'Ctrl+Alt+]'.
Activate the web console with:systemctl enable --now cockpit.socket
Last login:Sun Jan 8 11:47:30 2023 from 10.32.210.52
```

(2) Execute the **storage dmg pool redundancy-dump** <pool name> command to obtain the redundancy policies supported by the specified disk pool.

```
$ storage dmg pool redun-dump p1
Pool p1 Redun.
  RP_2
  RP_3
  EC_2P1
  EC_2P2_1
  EC_4P2_1
```

6.2 Assigning K8s cluster node roles

summary

If you need to install the CSI service using a Kubernetes cluster, tag all nodes that need to use CSI and assign the master or worker role.

operating command

(1) Execute the **kubectl label nodes** *<node_name>* **node-role.kubernetes.io/<node_role>=** command to label the nodes.

(2) Execute the **kubectl get node --show-labels** command to view the node label status.

Example of operation

Example of tagging and querying with nodes named master, node1 and node2 is as follows:

```
$ kubectl label nodes master node-role.kubernetes.io/master=
$ kubectl label nodes node1 node-role.kubernetes.io/worker=
$ kubectl label nodes node2 node-role.kubernetes.io/worker=
$ kubectl get node --show-labels
NAME STATUS ROLES AGE VERSION LABELS
master Ready control-plane,master 23d v1.24.0
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,
kubernetes.io/hostname=master,kubernetes.io/os=linux,node-role.kubernetes.io/control-
plane=,node-role.kubernetes.io/master=
node1 Ready worker 23d v1.24.0
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kuber
netes.io/ hostname=node1,kubernetes.io/os=linux,node-role.kubernetes.io/worker=
node2 Ready worker 23d v1.24.0
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kuber
netes.io/ hostname=node2,kubernetes.io/os=linux,node-role.kubernetes.io/worker=
```

6.3 Load CSI image package

summary

Support for loading CSI component packages in a manual manner after they have been uploaded to Kubernetes.

Operating Instructions

If you choose to load the CSI image package manually, you need to load it on all master nodes and worker nodes that use CSI.

Kubernetes clusters need to be loaded with nodes and CSI image packages that need to be installed, see 2.2.1 in Mirror Packages.

Use the following command to load the tarball to provide the image:

- If using podman, execute the **podman load -i** <*image_name*>**.tar** command.
- If using docker, execute the docker **load -i** <*image_name*>**.tar** command.
- If using containerd, execute the **ctr -n k8s.io images import** <*image_name*>**.tar** command.

View loading results

After all the loading is completed, execute the following commands on the master and worker nodes to view the image names respectively.

- If using podman, execute the **podman image** command.
- If you are using docker, execute the docker **images | grep v1** command.
- If using containerd, execute the **ctr -n k8s.io image list | grep csi** command.

6.4 Installation of third-party programs

6.4.1 Install nvme-cli

summary

Install the nvme-cli management tool for all nodes that require NVMe services.

procedure

(1) Use an account with administrator privileges to log in to any node.

(2) Execute the command yum install nvme-cli to install nvme-cli.

(3) Execute the nvme -version command to see if the installation was successful.

```
$nvme -version
nvme version 1.14
```

6.4.2 Install open-iscsi

summary

Install the iSCSI management tool for all nodes that need to use the iSCSI service.

procedure

(1) Use an account with administrator privileges to log in to any node.

(2) Execute the **yum install -y iscsi-initiator-utils** command to install iSCSI.

(3) Execute the **iscsiadm --version** command to see if the installation was successful.

```
$iscsiadm --version
iscsiadm version 6.2.1.4
```

6.4.3 Install DM-Multipath

summary

Install the multipathing management tool for all nodes that need to use the multipathing service.

procedure

(1) Use an account with administrator privileges to log in to any node.

(2) Execute the **yum install device-mapper-multipath** command to install the multipath software.

(3) Execute the **multipath -h** command to see if the installation was successful.

```
$ multipath -h
multipath-tools v0.8.7 (09/08, 2021)
```