Storage APIs

PersistentVolume [v1]

PersistentVolumeClaim [v1]

StorageClas

PersistentVolume [v1]

Description

PersistentVolume (PV) is a storage resource provisioned by an administrator. It is analogous to a node. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources

Property	Type	Description
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	PersistentVolumeSpec is the specification of a persistent volume.
`status`	`object`	PersistentVolumeStatus is the current status of a persistent volume.

.spec

Description

PersistentVolumeSpec is the specification of a persistent volume.

Type

`object`

Property	Type	Description
`accessModes`	`array`	accessModes contains all ways the volume ca mounted. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/storage/p volumes#access-modes ↗
`awsElasticBlockStore`	`object`	Represents a Persistent Disk resource in AWS An AWS EBS disk must exist before mounting container. The disk must also be in the same zone as the kubelet. An AWS EBS disk can o mounted as read/write once. AWS EBS volum support ownership management and SELinux relabeling.
`azureDisk`	`object`	AzureDisk represents an Azure Data Disk mo the host and bind mount to the pod.
`azureFile`	`object`	AzureFile represents an Azure File Service m the host and bind mount to the pod.
`capacity`	`object`	capacity is the description of the persistent vo resources and capacity. More info: https://kubernetes.io/docs/concepts/storage/p volumes#capacity ↗
`cephfs`	`object`	Represents a Ceph Filesystem mount that las lifetime of a pod Cephfs volumes do not supp ownership management or SELinux relabeling

Property	Type	Description
`cinder`	`object`	Represents a cinder volume resource in Oper Cinder volume must exist before mounting to container. The volume must also be in the sar region as the kubelet. Cinder volumes suppor ownership management and SELinux relabeli
`claimRef`	`object`	ObjectReference contains enough informatior you inspect or modify the referred object.
`csi`	`object`	Represents storage that is managed by an ex CSI volume driver
`fc`	`object`	Represents a Fibre Channel volume. Fibre Cł volumes can only be mounted as read/write o Fibre Channel volumes support ownership management and SELinux relabeling.
`flexVolume`	`object`	FlexPersistentVolumeSource represents a ge persistent volume resource that is provisioned/attached using an exec based plu
`flocker`	`object`	Represents a Flocker volume mounted by the agent. One and only one of datasetName anc datasetUUID should be set. Flocker volumes support ownership management or SELinux relabeling.

Property	Type	Description
`gcePersistentDisk`	`object`	Represents a Persistent Disk resource in Goo Compute Engine. A GCE PD must exist before mounting to a cc The disk must also be in the same GCE proje zone as the kubelet. A GCE PD can only be m as read/write once or read-only many times. C support ownership management and SELinux relabeling.
`glusterfs`	`object`	Represents a Glusterfs mount that lasts the lif a pod. Glusterfs volumes do not support owne management or SELinux relabeling.
`hostPath`	`object`	Represents a host path mapped into a pod. H volumes do not support ownership managem SELinux relabeling.
`iscsi`	`object`	ISCSIPersistentVolumeSource represents an disk. ISCSI volumes can only be mounted as read/write once. ISCSI volumes support owne management and SELinux relabeling.
`local`	`object`	Local represents directly-attached storage wit affinity
`mountOptions`	`array`	mountOptions is the list of mount options, e.g "soft"]. Not validated - mount will simply fail if

Property	Type	Description
		invalid. More info: https://kubernetes.io/docs/concepts/storage/p volumes/#mount-options ↗
`nfs`	`object`	Represents an NFS mount that lasts the lifeti pod. NFS volumes do not support ownership management or SELinux relabeling.
`nodeAffinity`	`object`	VolumeNodeAffinity defines constraints that li nodes this volume can be accessed from.
`persistentVolumeReclaimPolicy`	`string`	persistentVolumeReclaimPolicy defines what to a persistent volume when released from its Valid options are Retain (default for manually PersistentVolumes), Delete (default for dynan provisioned PersistentVolumes), and Recycle (deprecated). Recycle must be supported by t volume plugin underlying this PersistentVolun info: https://kubernetes.io/docs/concepts/storage/p volumes#reclaiming ↗ Possible enum values: - `"Delete"` means the volume will be dele Kubernetes on release from its claim. The plugin must support Deletion. - `"Recycle"` means the volume will be rec back into the pool of unbound persistent v on release from its claim. The volume plug support Recycling.

Property	Type	Description
		- `"Retain"` means the volume will be left current phase (Released) for manual recla by the administrator. The default policy is F
`photonPersistentDisk`	`object`	Represents a Photon Controller persistent dis resource.
`portworxVolume`	`object`	PortworxVolumeSource represents a Portwor resource.
`quobyte`	`object`	Represents a Quobyte mount that lasts the lif a pod. Quobyte volumes do not support owne management or SELinux relabeling.
`rbd`	`object`	Represents a Rados Block Device mount that lifetime of a pod. RBD volumes support owne management and SELinux relabeling.
`scaleIO`	`object`	ScaleIOPersistentVolumeSource represents a persistent ScaleIO volume
`storageClassName`	`string`	storageClassName is the name of StorageCla which this persistent volume belongs. Empty means that this volume does not belong to an StorageClass.

Property	Type	Description
`storageos`	`object`	Represents a StorageOS persistent volume r
`volumeAttributesClassName`	`string`	Name of VolumeAttributesClass to which this persistent volume belongs. Empty value is no allowed. When this field is not set, it indicates volume does not belong to any VolumeAttribu This field is mutable and can be changed by t driver after a volume has been updated succe a new class. For an unbound PersistentVolum volumeAttributesClassName will be matched unbound PersistentVolumeClaims during the process. This is a beta field and requires enab VolumeAttributesClass feature (off by default)
`volumeMode`	`string`	volumeMode defines if a volume is intended t used with a formatted filesystem or to remain block state. Value of Filesystem is implied wh included in spec. Possible enum values: • `"Block"` means the volume will not be f with a filesystem and will remain a raw blo device. • `"Filesystem"` means the volume will be formatted with a filesystem.
`vsphereVolume`	`object`	Represents a vSphere volume resource.

.spec.accessModes

Description

accessModes contains all ways the volume can be mounted. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes

Type

`array`

.spec.accessModes[]

Type

`string`

.spec.awsElasticBlockStore

Description

Represents a Persistent Disk resource in AWS. An AWS EBS disk must exist before mounting to a container. The disk must also be in the same AWS zone as the kubelet. An AWS EBS disk can only be mounted as read/write once. AWS EBS volumes support ownership management and SELinux relabeling.

Type

`object`

Required

`volumeID`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type of the volume that you want to mount. T Ensure that the filesystem type is supported by the host operating sy Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspe More info: https://kubernetes.io/docs/concepts/storage/volumes#awselasticbloc ↗
`partition`	`integer`	partition is the partition in the volume that you want to mount. If omit the default is to mount by volume name. Examples: For volume /dev you specify the partition as "1". Similarly, the volume partition for /de is "0" (or you can leave the property empty).
`readOnly`	`boolean`	readOnly value true will force the readOnly setting in VolumeMounts info: https://kubernetes.io/docs/concepts/storage/volumes#awselasticbloc ↗
`volumeID`	`string`	volumeID is unique ID of the persistent disk resource in AWS (Amaz EBS volume). More info: https://kubernetes.io/docs/concepts/storage/volumes#awselasticbloc ↗

.spec.azureDisk

Description

AzureDisk represents an Azure Data Disk mount on the host and bind mount to the pod.

Type

`object`

Required

`diskName` `diskURI`

Property	Type	Description
`cachingMode`	`string`	cachingMode is the Host Caching mode: None, Read Only, Read Write. Possible enum values: - `"None"` - `"ReadOnly"` - `"ReadWrite"`
`diskName`	`string`	diskName is the Name of the data disk in the blob storage
`diskURI`	`string`	diskURI is the URI of data disk in the blob storage
`fsType`	`string`	fsType is Filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
`kind`	`string`	kind expected values are Shared: multiple blob disks per storage account Dedicated: single blob disk per storage account Managed: azure managed data disk (only in managed availability set). defaults to shared Possible enum values: - `"Dedicated"`

Property	Type	Description
		 - `"Managed"` - `"Shared"`
`readOnly`	`boolean`	readOnly Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

.spec.azureFile

Description

AzureFile represents an Azure File Service mount on the host and bind mount to the pod.

Type

`object`

Required

`secretName` `shareName`

Property	Type	Description
`readOnly`	`boolean`	readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`secretName`	`string`	secretName is the name of secret that contains Azure Storage Account Name and Key
`secretNamespace`	`string`	secretNamespace is the namespace of the secret that contains Azure Storage Account Name and Key default is the same as the Pod

Property	Type	Description
`shareName`	`string`	shareName is the azure Share Name

.spec.capacity

Description

capacity is the description of the persistent volume's resources and capacity. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#capacity

Type

`object`

.spec.cephfs

Description

Represents a Ceph Filesystem mount that lasts the lifetime of a pod Cephfs volumes do not support ownership management or SELinux relabeling.

Type

`object`

Required

`monitors`

Property	Type	Description
`monitors`	`array`	monitors is Required: Monitors is a collection of Ceph monitors More info: https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it ↗

Property	Type	Description
`path`	`string`	path is Optional: Used as the mounted root, rather than the full Ceph tree, default is /
`readOnly`	`boolean`	readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it ↗
`secretFile`	`string`	secretFile is Optional: SecretFile is the path to key ring for User, default is /etc/ceph/user.secret More info: https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it ↗
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`user`	`string`	user is Optional: User is the rados user name, default is admin More info: https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it ↗

.spec.cephfs.monitors

Description

monitors is Required: Monitors is a collection of Ceph monitors More info: https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it

Type

`array`

.spec.cephfs.monitors[]

Type

`string`

.spec.cephfs.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.cinder

Description

Represents a cinder volume resource in Openstack. A Cinder volume must exist before mounting to a container. The volume must also be in the same region as the kubelet. Cinder volumes support ownership management and SELinux relabeling.

Type

`object`

Required

`volumeID`

Property	Type	Description
`fsType`	`string`	fsType Filesystem type to mount. Must be a filesystem type supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: https://examples.k8s.io/mysql-cinder-pd/README.md
`readOnly`	`boolean`	readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: https://examples.k8s.io/mysql-cinder-pd/README.md
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`volumeID`	`string`	volumeID used to identify the volume in cinder. More info: https://examples.k8s.io/mysql-cinder-pd/README.md

.spec.cinder.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.claimRef

Description

ObjectReference contains enough information to let you inspect or modify the referred object.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent.
`fieldPath`	`string`	If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod).

Property	Type	Description
		This syntax is chosen only to have some well-defined way of referencing a part of an object.
`kind`	`string`	Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`name`	`string`	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`namespace`	`string`	Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗
`resourceVersion`	`string`	Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗
`uid`	`string`	UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

.spec.csi

Description

Represents storage that is managed by an external CSI volume driver

Type

`object`

Required

`driver` `volumeHandle`

Property	Type	Description
`controllerExpandSecretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`controllerPublishSecretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`driver`	`string`	driver is the name of the driver to use for this volume. Required.
`fsType`	`string`	fsType to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs".
`nodeExpandSecretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Property	Type	Description
`nodePublishSecretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`nodeStageSecretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`readOnly`	`boolean`	readOnly value to pass to ControllerPublishVolumeRequest. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes of the volume to publish.
`volumeHandle`	`string`	volumeHandle is the unique volume name returned by the CSI volume plugin’s CreateVolume to refer to the volume on all subsequent calls. Required.

.spec.csi.controllerExpandSecretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.csi.controllerPublishSecretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.csi.nodeExpandSecretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.csi.nodePublishSecretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.csi.nodeStageSecretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.csi.volumeAttributes

Description

volumeAttributes of the volume to publish.

Type

`object`

.spec.fc

Description

Represents a Fibre Channel volume. Fibre Channel volumes can only be mounted as read/write once. Fibre Channel volumes support ownership management and SELinux relabeling.

Type

`object`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
`lun`	`integer`	lun is Optional: FC target lun number
`readOnly`	`boolean`	readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`targetWWNs`	`array`	targetWWNs is Optional: FC target worldwide names (WWNs)
`wwids`	`array`	wwids Optional: FC volume world wide identifiers (wwids) Either wwids or combination of targetWWNs and lun must be set, but not both simultaneously.

.spec.fc.targetWWNs

Description

targetWWNs is Optional: FC target worldwide names (WWNs)

Type

`array`

.spec.fc.targetWWNs[]

Type

`string`

.spec.fc.wwids

Description

wwids Optional: FC volume world wide identifiers (wwids) Either wwids or combination of targetWWNs and lun must be set, but not both simultaneously.

Type

`array`

.spec.fc.wwids[]

Type

`string`

.spec.flexVolume

Description

FlexPersistentVolumeSource represents a generic persistent volume resource that is provisioned/attached using an exec based plugin.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the driver to use for this volume.
`fsType`	`string`	fsType is the Filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs",

Property	Type	Description
		"ntfs". The default filesystem depends on FlexVolume script.
`options`	`object`	options is Optional: this field holds extra command options if any.
`readOnly`	`boolean`	readOnly is Optional: defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

.spec.flexVolume.options

Description

options is Optional: this field holds extra command options if any.

Type

`object`

.spec.flexVolume.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.flocker

Description

Represents a Flocker volume mounted by the Flocker agent. One and only one of datasetName and datasetUUID should be set. Flocker volumes do not support ownership management or SELinux relabeling.

Type

`object`

Property	Type	Description
`datasetName`	`string`	datasetName is Name of the dataset stored as metadata -> name on the dataset for Flocker should be considered as deprecated
`datasetUUID`	`string`	datasetUUID is the UUID of the dataset. This is unique identifier of a Flocker dataset

.spec.gcePersistentDisk

Description

Represents a Persistent Disk resource in Google Compute Engine. A GCE PD must exist before mounting to a container. The disk must also be in the same GCE project and zone as the kubelet. A GCE PD can only be mounted as read/write once or read-only many times. GCE PDs support ownership management and SELinux relabeling.

Type

`object`

Required

`pdName`

Property	Type	Description
`fsType`	`string`	fsType is filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" unspecified. More info: https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentc ↗
`partition`	`integer`	partition is the partition in the volume that you want to mount. If omitted, the default is to mount by volume name. Examples: For volume /dev/sda1, you specify the partition as "1". Similarly, the volu partition for /dev/sda is "0" (or you can leave the property empty). Mo info: https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentc ↗
`pdName`	`string`	pdName is unique name of the PD resource in GCE. Used to identify the disk in GCE. More info: https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentc ↗

Property	Type	Description
`readOnly`	`boolean`	readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentc ↗

.spec.glusterfs

Description

Represents a Glusterfs mount that lasts the lifetime of a pod. Glusterfs volumes do not support ownership management or SELinux relabeling.

Type

`object`

Required

`endpoints` `path`

Property	Type	Description
`endpoints`	`string`	endpoints is the endpoint name that details Glusterfs topc More info: https://examples.k8s.io/volumes/glusterfs/README.md#c a-pod ↗
`endpointsNamespace`	`string`	endpointsNamespace is the namespace that contains Glu endpoint. If this field is empty, the EndpointNamespace de to the same namespace as the bound PVC. More info: https://examples.k8s.io/volumes/glusterfs/README.md#c a-pod ↗

Property	Type	Description
`path`	`string`	path is the Glusterfs volume path. More info: https://examples.k8s.io/volumes/glusterfs/README.md#c a-pod
`readOnly`	`boolean`	readOnly here will force the Glusterfs volume to be moun with read-only permissions. Defaults to false. More info: https://examples.k8s.io/volumes/glusterfs/README.md#c a-pod

.spec.hostPath

Description

Represents a host path mapped into a pod. Host path volumes do not support ownership management or SELinux relabeling.

Type

`object`

Required

`path`

Property	Type	Description
`path`	`string`	path of the directory on the host. If the path is a symlink, it will follow the link to the real path. More info: https://kubernetes.io/docs/concepts/storage/volumes#hostpath
`type`	`string`	type for HostPath Volume Defaults to "" More info: https://kubernetes.io/docs/concepts/storage/volumes#hostpath

Property	Type	Description
		Possible enum values: - `""` For backwards compatible, leave it empty if unset - `"BlockDevice"` A block device must exist at the given path - `"CharDevice"` A character device must exist at the given path - `"Directory"` A directory must exist at the given path - `"DirectoryOrCreate"` If nothing exists at the given path, an empty directory will be created there as needed with file mode 0755, having the same group and ownership with Kubelet. - `"File"` A file must exist at the given path - `"FileOrCreate"` If nothing exists at the given path, an empty file will be created there as needed with file mode 0644, having the same group and ownership with Kubelet. - `"Socket"` A UNIX socket must exist at the given path

.spec.iscsi

Description

ISCSIPersistentVolumeSource represents an ISCSI disk. ISCSI volumes can only be mounted as read/write once. ISCSI volumes support ownership management and SELinux relabeling.

Type

`object`

Required

`targetPortal` `iqn` `lun`

Property	Type	Description
`chapAuthDiscovery`	`boolean`	chapAuthDiscovery defines whether support iSCSI Discovery CHAP authentication
`chapAuthSession`	`boolean`	chapAuthSession defines whether support iSCSI Session CHAP authentication
`fsType`	`string`	fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: https://kubernetes.io/docs/concepts/storage/volumes#iscsi ↗
`initiatorName`	`string`	initiatorName is the custom iSCSI Initiator Name. If initiatorName is specified with iscsiInterface simultaneously, new iSCSI interface : will be created for the connection.
`iqn`	`string`	iqn is Target iSCSI Qualified Name.
`iscsiInterface`	`string`	iscsiInterface is the interface Name that uses an iSCSI transport. Defaults to 'default' (tcp).
`lun`	`integer`	lun is iSCSI Target Lun number.

Property	Type	Description
`portals`	`array`	portals is the iSCSI Target Portal List. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).
`readOnly`	`boolean`	readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false.
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`targetPortal`	`string`	targetPortal is iSCSI Target Portal. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

.spec.iscsi.portals

Description

portals is the iSCSI Target Portal List. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

Type

`array`

.spec.iscsi.portals[]

Type

`string`

.spec.iscsi.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.local

Description

Local represents directly-attached storage with node affinity

Type

`object`

Required

`path`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type to mount. It applies only when the Path is a block device. Must be a filesystem type supported by the

Property	Type	Description
		host operating system. Ex. "ext4", "xfs", "ntfs". The default value is to auto-select a filesystem if unspecified.
`path`	`string`	path of the full path to the volume on the node. It can be either a directory or block device (disk, partition, ...).

.spec.mountOptions

Description

mountOptions is the list of mount options, e.g. ["ro", "soft"]. Not validated - mount will simply fail if one is invalid. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes/#mount-options

Type

`array`

.spec.mountOptions[]

Type

`string`

.spec.nfs

Description

Represents an NFS mount that lasts the lifetime of a pod. NFS volumes do not support ownership management or SELinux relabeling.

Type

`object`

Required

`server` `path`

Property	Type	Description
`path`	`string`	path that is exported by the NFS server. More info: https://kubernetes.io/docs/concepts/storage/volumes#nfs
`readOnly`	`boolean`	readOnly here will force the NFS export to be mounted with read-only permissions. Defaults to false. More info: https://kubernetes.io/docs/concepts/storage/volumes#nfs
`server`	`string`	server is the hostname or IP address of the NFS server. More info: https://kubernetes.io/docs/concepts/storage/volumes#nfs

.spec.nodeAffinity

Description

VolumeNodeAffinity defines constraints that limit what nodes this volume can be accessed from.

Type

`object`

Property	Type	Description
`required`	`object`	A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

.spec.nodeAffinity.required

Description

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

Type

`object`

Required

`nodeSelectorTerms`

Property	Type	Description
`nodeSelectorTerms`	`array`	Required. A list of node selector terms. The terms are ORed.

.spec.nodeAffinity.required.nodeSelectorTerms

Description

Required. A list of node selector terms. The terms are ORed.

Type

`array`

.spec.nodeAffinity.required.nodeSelectorTerms[]

Description

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	A list of node selector requirements by node's labels.
`matchFields`	`array`	A list of node selector requirements by node's fields.

.spec.nodeAffinity.required.nodeSelectorTerms[].matchExpressions

Description

A list of node selector requirements by node's labels.

Type

`array`

.spec.nodeAffinity.required.nodeSelectorTerms[].matchExpressions[]

Description

A node selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	The label key that the selector applies to.

Property	Type	Description
`operator`	`string`	Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt. Possible enum values: - `"DoesNotExist"` - `"Exists"` - `"Gt"` - `"In"` - `"Lt"` - `"NotIn"`
`values`	`array`	An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

.spec.nodeAffinity.required.nodeSelectorTerms[].matchExpressions[].values

Description

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

Type

`array`

.spec.nodeAffinity.required.nodeSelectorTerms[].matchExpressions[].values[]

Type

`string`

.spec.nodeAffinity.required.nodeSelectorTerms[].matchFields

Description

A list of node selector requirements by node's fields.

Type

`array`

.spec.nodeAffinity.required.nodeSelectorTerms[].matchFields[]

Description

A node selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	The label key that the selector applies to.

Property	Type	Description
`operator`	`string`	Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt. Possible enum values: - `"DoesNotExist"` - `"Exists"` - `"Gt"` - `"In"` - `"Lt"` - `"NotIn"`
`values`	`array`	An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

.spec.nodeAffinity.required.nodeSelectorTerms[].matchFields[].values

Description

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

Type

`array`

.spec.nodeAffinity.required.nodeSelectorTerms[].matchFields[].values[]

Type

`string`

.spec.photonPersistentDisk

Description

Represents a Photon Controller persistent disk resource.

Type

`object`

Required

`pdID`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
`pdID`	`string`	pdID is the ID that identifies Photon Controller persistent disk

.spec.portworxVolume

Description

PortworxVolumeSource represents a Portworx volume resource.

Type

`object`

Required

`volumeID`

Property	Type	Description
`fsType`	`string`	fSType represents the filesystem type to mount Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs". Implicitly inferred to be "ext4" if unspecified.
`readOnly`	`boolean`	readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`volumeID`	`string`	volumeID uniquely identifies a Portworx volume

.spec.quobyte

Description

Represents a Quobyte mount that lasts the lifetime of a pod. Quobyte volumes do not support ownership management or SELinux relabeling.

Type

`object`

Required

`registry` `volume`

Property	Type	Description
`group`	`string`	group to map volume access to Default is no group

Property	Type	Description
`readOnly`	`boolean`	readOnly here will force the Quobyte volume to be mounted with read-only permissions. Defaults to false.
`registry`	`string`	registry represents a single or multiple Quobyte Registry services specified as a string as host:port pair (multiple entries are separated with commas) which acts as the central registry for volumes
`tenant`	`string`	tenant owning the given Quobyte volume in the Backend Used with dynamically provisioned Quobyte volumes, value is set by the plugin
`user`	`string`	user to map volume access to Defaults to serivceaccount user
`volume`	`string`	volume is a string that references an already created Quobyte volume by name.

.spec.rbd

Description

Represents a Rados Block Device mount that lasts the lifetime of a pod. RBD volumes support ownership management and SELinux relabeling.

Type

`object`

Required

`monitors` `image`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: https://kubernetes.io/docs/concepts/storage/volumes#rbd ↗
`image`	`string`	image is the rados image name. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗
`keyring`	`string`	keyring is the path to key ring for RBDUser. Default is /etc/ceph/keyring. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗
`monitors`	`array`	monitors is a collection of Ceph monitors. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗
`pool`	`string`	pool is the rados pool name. Default is rbd. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗

Property	Type	Description
`readOnly`	`boolean`	readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`user`	`string`	user is the rados user name. Default is admin. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it ↗

.spec.rbd.monitors

Description

monitors is a collection of Ceph monitors. More info: https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it

Type

`array`

.spec.rbd.monitors[]

Type

`string`

.spec.rbd.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.scaleIO

Description

ScaleIOPersistentVolumeSource represents a persistent ScaleIO volume

Type

`object`

Required

`gateway` `system` `secretRef`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Default is "xfs"

Property	Type	Description
`gateway`	`string`	gateway is the host address of the ScaleIO API Gateway.
`protectionDomain`	`string`	protectionDomain is the name of the ScaleIO Protection Domain for the configured storage.
`readOnly`	`boolean`	readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`secretRef`	`object`	SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace
`sslEnabled`	`boolean`	sslEnabled is the flag to enable/disable SSL communication with Gateway, default false
`storageMode`	`string`	storageMode indicates whether the storage for a volume should be ThickProvisioned or ThinProvisioned. Default is ThinProvisioned.
`storagePool`	`string`	storagePool is the ScaleIO Storage Pool associated with the protection domain.

Property	Type	Description
`system`	`string`	system is the name of the storage system as configured in ScaleIO.
`volumeName`	`string`	volumeName is the name of a volume already created in the ScaleIO system that is associated with this volume source.

.spec.scaleIO.secretRef

Description

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.storageos

Description

Represents a StorageOS persistent volume resource.

Type

`object`

Property	Type	Description
`fsType`	`string`	fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
`readOnly`	`boolean`	readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
`secretRef`	`object`	ObjectReference contains enough information to let you inspect or modify the referred object.
`volumeName`	`string`	volumeName is the human-readable name of the StorageOS volume. Volume names are only unique within a namespace.
`volumeNamespace`	`string`	volumeNamespace specifies the scope of the volume within StorageOS. If no namespace is specified then the Pod's namespace will be used. This allows the Kubernetes name scoping to be mirrored within StorageOS for tighter integration. Set VolumeName to any name to override the default behaviour. Set to "default" if you are not using namespaces within StorageOS. Namespaces that do not pre-exist within StorageOS will be created.

.spec.storageos.secretRef

Description

ObjectReference contains enough information to let you inspect or modify the referred object.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent.
`fieldPath`	`string`	If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.
`kind`	`string`	Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

Property	Type	Description
`name`	`string`	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`namespace`	`string`	Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗
`resourceVersion`	`string`	Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗
`uid`	`string`	UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

.spec.vsphereVolume

Description

Represents a vSphere volume resource.

Type

`object`

Required

`volumePath`

Property	Type	Description
`fsType`	`string`	fsType is filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
`storagePolicyID`	`string`	storagePolicyID is the storage Policy Based Management (SPBM) profile ID associated with the StoragePolicyName.
`storagePolicyName`	`string`	storagePolicyName is the storage Policy Based Management (SPBM) profile name.
`volumePath`	`string`	volumePath is the path that identifies vSphere volume vmdk

.status

Description

PersistentVolumeStatus is the current status of a persistent volume.

Type

`object`

Property	Type	Description
`lastPhaseTransitionTime`	`string`	Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers ar

Property	Type	Description
		provided for many of the factory methods that the tim package offers.
`message`	`string`	message is a human-readable message indicating details about why the volume is in this state.
`phase`	`string`	phase indicates if a volume is available, bound to a claim, or released by a claim. More info: [https://kubernetes.io/docs/concepts/storage/persistei volumes#phase](https://kubernetes.io/docs/concepts/storage/persistei) Possible enum values: - `"Available"` used for PersistentVolumes that are not yet bound Available volumes are held by the binder and matched to PersistentVolumeClair - `"Bound"` used for PersistentVolumes that are bound - `"Failed"` used for PersistentVolumes that faile to be correctly recycled or deleted after being released from a claim - `"Pending"` used for PersistentVolumes that are not available - `"Released"` used for PersistentVolumes where the bound PersistentVolumeClaim was deleted released volumes must be recycled before becoming available again this phase is used by tl persistent volume claim binder to signal to anothe process to reclaim the resource

Property	Type	Description
`reason`	`string`	reason is a brief CamelCase string that describes an failure and is meant for machine parsing and tidy display in the CLI.

API Endpoints

The following API endpoints are available:

- `/kubernetes/{cluster}/api/v1/persistentvolumes`
  - `DELETE`: delete collection of PersistentVolume
  - `GET`: list objects of kind PersistentVolume
  - `POST`: create a new PersistentVolume
- `/kubernetes/{cluster}/api/v1/persistentvolumes/{name}`
  - `DELETE`: delete the specified PersistentVolume
  - `GET`: read the specified PersistentVolume
  - `PATCH`: partially update the specified PersistentVolume
  - `PUT`: replace the specified PersistentVolume
- `/kubernetes/{cluster}/api/v1/persistentvolumes/{name}/status`
  - `GET`: read status of the specified PersistentVolume
  - `PATCH`: partially update status of the specified PersistentVolume
  - `PUT`: replace status of the specified PersistentVolume

/kubernetes/{cluster}/api/v1/persistentvolumes

HTTP method

`DELETE`

Description

delete collection of PersistentVolume

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind PersistentVolume

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolume` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
201 - Created	`PersistentVolume` schema
202 - Accepted	`PersistentVolume` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/api/v1/persistentvolumes/{name}

HTTP method

`DELETE`

Description

delete the specified PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified PersistentVolume

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolume` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
201 - Created	`PersistentVolume` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/api/v1/persistentvolumes/{name}/status

HTTP method

`GET`

Description

read status of the specified PersistentVolume

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing

Parameter	Type	Description
		of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified PersistentVolume

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolume` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolume` schema
201 - Created	`PersistentVolume` schema
401 - Unauthorized	Empty

Alauda Container Platform

PersistentVolumeClaim [v1]

Description

PersistentVolumeClaim is a user's request for and claim to a persistent volume

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

Property	Type	Description
`metadata`	`ObjectMeta`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	PersistentVolumeClaimSpec describes the common attributes of storage devices and allows a Source for provider-specific attributes
`status`	`object`	PersistentVolumeClaimStatus is the current status of a persistent volume claim.

.spec

Description

PersistentVolumeClaimSpec describes the common attributes of storage devices and allows a Source for provider-specific attributes

Type

`object`

Property	Type	Description
`accessModes`	`array`	accessModes contains the desired access modes volume should have. More info: https://kubernetes.io/docs/concepts/storage/persis volumes#access-modes-1 ↗
`dataSource`	`object`	TypedLocalObjectReference contains enough information to let you locate the typed referenced

Property	Type	Description
		object inside the same namespace.
`dataSourceRef`	`object`	TypedObjectReference contains enough informatic let you locate the typed referenced object
`resources`	`object`	VolumeResourceRequirements describes the stor resource requirements for a volume.
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label sel matches all objects. A null label selector matches r objects.
`storageClassName`	`string`	storageClassName is the name of the StorageCla required by the claim. More info: https://kubernetes.io/docs/concepts/storage/persis volumes#class-1 ↗
`volumeAttributesClassName`	`string`	volumeAttributesClassName may be used to set tl VolumeAttributesClass used by this claim. If speci the CSI driver will create or update the volume witl attributes defined in the corresponding VolumeAttributesClass. This has a different purpos than storageClassName, it can be changed after tl claim is created. An empty string value means tha VolumeAttributesClass will be applied to the claim it's not allowed to reset this field to empty string or

Property	Type	Description
		is set. If unspecified and the PersistentVolumeClai unbound, the default VolumeAttributesClass will be by the persistentvolume controller if it exists. If the resource referred to by volumeAttributesClass doe not exist, this PersistentVolumeClaim will be set to Pending state, as reflected by the modifyVolumeSt field, until such as a resource exists. More info: https://kubernetes.io/docs/concepts/storage/volum attributes-classes/ (Beta) Using this field require the VolumeAttributesClass feature gate to be enab (off by default).
`volumeMode`	`string`	volumeMode defines what type of volume is requir by the claim. Value of Filesystem is implied when r included in claim spec. Possible enum values: - `"Block"` means the volume will not be forma with a filesystem and will remain a raw block device. - `"Filesystem"` means the volume will be or is formatted with a filesystem.
`volumeName`	`string`	volumeName is the binding reference to the PersistentVolume backing this claim.

.spec.accessModes

Description

accessModes contains the desired access modes the volume should have. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

Type

`array`

.spec.accessModes[]

Type

`string`

.spec.dataSource

Description

TypedLocalObjectReference contains enough information to let you locate the typed referenced object inside the same namespace.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiGroup`	`string`	APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.
`kind`	`string`	Kind is the type of resource being referenced
`name`	`string`	Name is the name of resource being referenced

.spec.dataSourceRef

Description

TypedObjectReference contains enough information to let you locate the typed referenced object

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiGroup`	`string`	APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.
`kind`	`string`	Kind is the type of resource being referenced
`name`	`string`	Name is the name of resource being referenced
`namespace`	`string`	Namespace is the namespace of resource being referenced Note that when a namespace is specified, a gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. (Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

.spec.resources

Description

VolumeResourceRequirements describes the storage resource requirements for a volume.

Type

`object`

Property	Type	Description
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.resources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.resources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.selector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.selector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.selector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.selector.matchExpressions[].values[]

Type

`string`

.spec.selector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.status

Description

PersistentVolumeClaimStatus is the current status of a persistent volume claim.

Type

`object`

Property	Type	Description
`accessModes`	`array`	accessModes contains the actual access volume backing the PVC has. More info: https://kubernetes.io/docs/concepts/storag volumes#access-modes-1
`allocatedResourceStatuses`	`object`	allocatedResourceStatuses stores status being resized for the given PVC. Key nam standard Kubernetes label syntax. Valid va either: * Un-prefixed keys: - storage - the the volume. * Custom resources must use implementation-defined prefixed names su "example.com/my-custom-resource" Apart values - keys that are unprefixed or have kubernetes.io prefix are considered reserv hence may not be used. ClaimResourceStatus can be in any of foll states: - ControllerResizeInProgress: Stat resize controller starts resizing the volume plane. - ControllerResizeFailed: State set has failed in resize controller with a termin NodeResizePending: State set when resiz has finished resizing the volume but furthe volume is needed on the node. - NodeResizeInProgress: State set when ku resizing the volume. - NodeResizeFailed: when resizing has failed in kubelet with a error. Transient errors don't set NodeResiz example: if expanding a PVC for more cap field can be one of the following states: - pvc.status.allocatedResourceStatus['stora "ControllerResizeInProgress" - pvc.status.allocatedResourceStatus['stora

Property	Type	Description
		"ControllerResizeFailed" - pvc.status.allocatedResourceStatus['stora "NodeResizePending" - pvc.status.allocatedResourceStatus['stora "NodeResizeInProgress" - pvc.status.allocatedResourceStatus['stora "NodeResizeFailed" When this field is not means that no resize operation is in progr given PVC. A controller that receives PVC update with unknown resourceName or ClaimResourc should ignore the update for the purpose i designed. For example - a controller that c responsible for resizing capacity of the vol ignore PVC updates that change other val associated with PVC. This is an alpha field and requires enablin RecoverVolumeExpansionFailure feature.
`allocatedResources`	`object`	allocatedResources tracks the resources a a PVC including its capacity. Key names fo standard Kubernetes label syntax. Valid va either: * Un-prefixed keys: - storage - the c the volume. * Custom resources must use implementation-defined prefixed names su "example.com/my-custom-resource" Apart values - keys that are unprefixed or have kubernetes.io prefix are considered reserv hence may not be used. Capacity reported here may be larger than capacity when a volume expansion operat requested. For storage quota, the larger v

Property	Type	Description
		allocatedResources and PVC.spec.resour If allocatedResources is not set, PVC.spe alone is used for quota calculation. If a vol expansion capacity request is lowered, allocatedResources is only lowered if ther expansion operations in progress and if th volume capacity is equal or lower than the capacity. A controller that receives PVC update with unknown resourceName should ignore the the purpose it was designed. For example controller that only is responsible for resizi of the volume, should ignore PVC updates change other valid resources associated v This is an alpha field and requires enablin RecoverVolumeExpansionFailure feature.
`capacity`	`object`	capacity represents the actual resources c underlying volume.
`conditions`	`array`	conditions is the current Condition of persi volume claim. If underlying persistent volu resized then the Condition will be set to 'R

Property	Type	Description
`currentVolumeAttributesClassName`	`string`	currentVolumeAttributesClassName is the name of the VolumeAttributesClass the P\ When unset, there is no VolumeAttributeC to this PersistentVolumeClaim This is a be requires enabling VolumeAttributesClass f by default).
`modifyVolumeStatus`	`object`	ModifyVolumeStatus represents the status ControllerModifyVolume operation
`phase`	`string`	phase represents the current phase of PersistentVolumeClaim. Possible enum values: - `"Bound"` used for PersistentVolumeC are bound - `"Lost"` used for PersistentVolumeCl lost their underlying PersistentVolume. was bound to a PersistentVolume and does not exist any longer and all data (lost. - `"Pending"` used for PersistentVolum are not yet bound

.status.accessModes

Description

accessModes contains the actual access modes the volume backing the PVC has. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

Type

`array`

.status.accessModes[]

Type

`string`

.status.allocatedResourceStatuses

Description

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used. ClaimResourceStatus can be in any of following states: - ControllerResizeInProgress: State set when resize controller starts resizing the volume in control-plane. - ControllerResizeFailed: State set when resize has failed in resize controller with a terminal error. - NodeResizePending: State set when resize controller has finished resizing the volume but further resizing of volume is needed on the node. - NodeResizeInProgress: State set when kubelet starts resizing the volume. - NodeResizeFailed: State set when resizing has failed in kubelet with a terminal error. Transient errors don't set NodeResizeFailed. For example: if expanding a PVC for more capacity - this field can be one of the following states: - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeFailed" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizePending" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeFailed" When this field is not set, it means that no resize operation is in progress for the given PVC. A controller that receives PVC update with previously unknown resourceName or ClaimResourceStatus should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC. This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

Type

`object`

.status.allocatedResources

Description

allocatedResources tracks the resources allocated to a PVC including its capacity. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used. Capacity reported here may be larger than the actual capacity when a volume expansion operation is requested. For storage quota, the larger value from allocatedResources and PVC.spec.resources is used. If allocatedResources is not set, PVC.spec.resources alone is used for quota calculation. If a volume expansion capacity request is lowered, allocatedResources is only lowered if there are no expansion operations in progress and if the actual volume capacity is equal or lower than the requested capacity. A controller that receives PVC update with previously unknown resourceName should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC. This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

Type

`object`

.status.capacity

Description

capacity represents the actual resources of the underlying volume.

Type

`object`

.status.conditions

Description

conditions is the current Condition of persistent volume claim. If underlying persistent volume is being resized then the Condition will be set to 'Resizing'.

Type

`array`

.status.conditions[]

Description

PersistentVolumeClaimCondition contains details about state of pvc

Type

`object`

Required

`type` `status`

Property	Type	Description
`lastProbeTime`	`string`	Time is a wrapper around time.Time which supports correc JSON. Wrappers are provided for many of the factory meth offers.
`lastTransitionTime`	`string`	Time is a wrapper around time.Time which supports correc JSON. Wrappers are provided for many of the factory meth offers.
`message`	`string`	message is the human-readable message indicating detail

Property	Type	Description
`reason`	`string`	reason is a unique, this should be a short, machine unders the reason for condition's last transition. If it reports "Resizi underlying persistent volume is being resized.
`status`	`string`	Status is the status of the condition. Can be True, False, U https://kubernetes.io/docs/reference/kubernetes-api/config- resources/persistent-volume-claim-v1/#:~:text=state%20of%20pvc-,conditions.status,-(string)%
`type`	`string`	Type is the type of the condition. More info: https://kubernetes.io/docs/reference/kubernetes-api/config- resources/persistent-volume-claim-v1/#:~:text=set%20to%20%27ResizeStarted%27.-,Persiste contains%20details%20about ↗

.status.modifyVolumeStatus

Description

ModifyVolumeStatus represents the status object of ControllerModifyVolume operation

Type

`object`

Required

`status`

Property	Type	Description
`status`	`string`	status is the status of the ControllerModifyVolume operation. It

Property	Type	Description
		can be in any of following states: - Pending Pending indicates that the PersistentVolumeClaim cannot be modified due to unmet requirements, such as the specified VolumeAttributesClass not existing. - InProgress InProgress indicates that the volume is being modified. - Infeasible Infeasible indicates that the request has been rejected as invalid by the CSI driver. To resolve the error, a valid VolumeAttributesClass needs to be specified. Note: New statuses can be added in the future. Consumers should check for unknown statuses and fail appropriately. Possible enum values: - `"InProgress"` InProgress indicates that the volume is being modified - `"Infeasible"` Infeasible indicates that the request has been rejected as invalid by the CSI driver. To resolve the error, a valid VolumeAttributesClass needs to be specified - `"Pending"` Pending indicates that the PersistentVolumeClaim cannot be modified due to unmet

Property	Type	Description
		requirements, such as the specified VolumeAttributesClass not existing
`targetVolumeAttributesClassName`	`string`	targetVolumeAttributesClassName is the name of the VolumeAttributesClass the PVC currently being reconciled

API Endpoints

The following API endpoints are available:

- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims`
  - `DELETE`: delete collection of PersistentVolumeClaim
  - `GET`: list objects of kind PersistentVolumeClaim
  - `POST`: create a new PersistentVolumeClaim
- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}`
  - `DELETE`: delete the specified PersistentVolumeClaim
  - `GET`: read the specified PersistentVolumeClaim
  - `PATCH`: partially update the specified PersistentVolumeClaim
  - `PUT`: replace the specified PersistentVolumeClaim
- `/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}/status`
  - `GET`: read status of the specified PersistentVolumeClaim
  - `PATCH`: partially update status of the specified PersistentVolumeClaim

- `PUT`: replace status of the specified PersistentVolumeClaim

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims

HTTP method

`DELETE`

Description

delete collection of PersistentVolumeClaim

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind PersistentVolumeClaim

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaimList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolumeClaim` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
201 - Created	`PersistentVolumeClaim` schema

HTTP code	Response body
202 - Accepted	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}

HTTP method

`DELETE`

Description

delete the specified PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified PersistentVolumeClaim

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are

Parameter	Type	Description
		present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a

Parameter	Type	Description
		BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolumeClaim` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
201 - Created	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}/status

HTTP method

`GET`

Description

read status of the specified PersistentVolumeClaim

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified PersistentVolumeClaim

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PersistentVolumeClaim` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PersistentVolumeClaim` schema
201 - Created	`PersistentVolumeClaim` schema
401 - Unauthorized	Empty

StorageClass [storage.k8s.io/v1]

Description

StorageClass describes the parameters for a class of storage for which PersistentVolumes can be dynamically provisioned. StorageClasses are non-namespaced; the name of the storage class according to etcd is in ObjectMeta.Name.

Type

`object`

Required

`provisioner`

Specification

Property	Type	Description
`allowVolumeExpansion`	`boolean`	allowVolumeExpansion shows whether the storage class allow volume expand.
`allowedTopologies`	`array`	allowedTopologies restrict the node topologies where volumes can be dynamically provisioned. Each volume plugin defines its own supported topology specifications. An empty TopologySelectorTerm list means there is no topology restriction. This field is only honored by

Property	Type	Description
		servers that enable the VolumeScheduling feature.
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`mountOptions`	`array`	mountOptions controls the mountOptions for dynamically provisioned PersistentVolumes of this storage class. e.g. ["ro", "soft"]. Not validated - mount of the PVs will simply fail if one is invalid.

Property	Type	Description
`parameters`	`object`	parameters holds the parameters for the provisioner that should create volumes of this storage class.
`provisioner`	`string`	provisioner indicates the type of the provisioner.
`reclaimPolicy`	`string`	reclaimPolicy controls the reclaimPolicy for dynamically provisioned PersistentVolumes of this storage class. Defaults to Delete. Possible enum values: - `"Delete"` means the volume will be deleted from Kubernetes on release from its claim. The volume plugin must support Deletion. - `"Recycle"` means the volume will be recycled back into the pool of unbound persistent volumes on release from its claim. The volume plugin must support Recycling. - `"Retain"` means the volume will be left in its current phase (Released) for manual reclamation by the administrator. The default policy is Retain.
`volumeBindingMode`	`string`	volumeBindingMode indicates how PersistentVolumeClaims should be provisioned and bound. When unset, VolumeBindingImmediate is used. This field is only honored by servers that enable the VolumeScheduling feature.

Property	Type	Description
		Possible enum values: - `"Immediate"` indicates that PersistentVolumeClaims should be immediately provisioned and bound. This is the default mode. - `"WaitForFirstConsumer"` indicates that PersistentVolumeClaims should not be provisioned and bound until the first Pod is created that references the PeristentVolumeClaim. The volume provisioning and binding will occur during Pod scheduling.

.allowedTopologies

Description

allowedTopologies restrict the node topologies where volumes can be dynamically provisioned. Each volume plugin defines its own supported topology specifications. An empty TopologySelectorTerm list means there is no topology restriction. This field is only honored by servers that enable the VolumeScheduling feature.

Type

`array`

.allowedTopologies[]

Description

A topology selector term represents the result of label queries. A null or empty topology selector term matches no objects. The requirements of them are ANDed. It provides a subset of functionality as NodeSelectorTerm. This is an alpha feature and may change in the future.

Type

`object`

Property	Type	Description
`matchLabelExpressions`	`array`	A list of topology selector requirements by labels.

.allowedTopologies[].matchLabelExpressions

Description

A list of topology selector requirements by labels.

Type

`array`

.allowedTopologies[].matchLabelExpressions[]

Description

A topology selector requirement is a selector that matches given label. This is an alpha feature and may change in the future.

Type

`object`

Required

`key` `values`

Property	Type	Description
`key`	`string`	The label key that the selector applies to.
`values`	`array`	An array of string values. One value must match the label to be selected. Each entry in Values is ORed.

.allowedTopologies[].matchLabelExpressions[].values

Description

An array of string values. One value must match the label to be selected. Each entry in Values is ORed.

Type

`array`

.allowedTopologies[].matchLabelExpressions[].values[]

Type

`string`

.mountOptions

Description

mountOptions controls the mountOptions for dynamically provisioned PersistentVolumes of this storage class. e.g. ["ro", "soft"]. Not validated - mount of the PVs will simply fail if one is invalid.

Type

`array`

.mountOptions[]

Type

`string`

.parameters

Description

parameters holds the parameters for the provisioner that should create volumes of this storage class.

Type

`object`

API Endpoints

The following API endpoints are available:

- `/kubernetes/{cluster}/apis/storage.k8s.io/v1/storageclasses`
  - `DELETE`: delete collection of StorageClass
  - `GET`: list objects of kind StorageClass
  - `POST`: create a new StorageClass
- `/kubernetes/{cluster}/apis/storage.k8s.io/v1/storageclasses/{name}`
  - `DELETE`: delete the specified StorageClass
  - `GET`: read the specified StorageClass
  - `PATCH`: partially update the specified StorageClass
  - `PUT`: replace the specified StorageClass

/kubernetes/{cluster}/apis/storage.k8s.io/v1/storageclasses

HTTP method

`DELETE`

Description

delete collection of StorageClass

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind StorageClass

HTTP responses

HTTP code	Response body
200 - OK	`StorageClassList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new StorageClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a

Parameter	Type	Description
		warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`StorageClass` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`StorageClass` schema
201 - Created	`StorageClass` schema
202 - Accepted	`StorageClass` schema
401 - Unauthorized	Empty

/kubernetes/{cluster}/apis/storage.k8s.io/v1/storageclasses/{name}

HTTP method

`DELETE`

Description

delete the specified StorageClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified StorageClass

HTTP responses

HTTP code	Response body
200 - OK	`StorageClass` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified StorageClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`StorageClass` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified StorageClass

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`StorageClass` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`StorageClass` schema
201 - Created	`StorageClass` schema
401 - Unauthorized	Empty