

KumoScale CSI Storage Provisioning and Volume Management

This section explains how to provision and manage volumes using storage classes.

The following topics are covered

- [Using Storage Classes to Create a Simple Volume](#) explains how to create a storage class and provision a simple volume.
- [Storage Class Parameters](#) lists all the parameters supported in a storage class
- [Provisioning Multiple Volumes at Scale](#) explains how to scale up to provision multiple pods hosting volumes defined by your storage class.
- [Creating Multiple Replicas for a Volume](#) explains how to create a volume with more than one leg (replica) and scale up to support multiple pods.
- [Expanding Volumes](#) explains how to add replicas to a volume.
- [Deleting Volumes](#) explains how to delete a volume.
- [Cloning a Volume](#) explains how to clone a volume.
- [Extended Resources](#) explains how to create pods with requests for a specific number of extended resources.
- [Volume Migration](#) is discussed in the the [Volume Migration](#) section of the KumoScale User Guide.

TIP: You can also learn more about the general topic of volume management at the [KumoScale User Guide: Volume Management](#) page.

Using Storage Classes to Create a Simple Volume

This section describes the steps need to create a pod hosting a single volume defined by a storage class.

The KumoScale CSI driver supports the volume properties shown in the [Storage Class Parameters](#) table below. Below are the steps to creating your own storage classes and provisioning volumes starting with example files included in the KumoScale directory **Kubernetes_CSI** directory. Although we use the exact same name of the files here, you will want to make a copies to use for your own testing purposes.

- **provisioner-storage-class.yaml** – an example of a storage class definition.
- **kube1PVC.yaml** – an example of a persistent volume claim created as a KumoScale volume.
- **testServicePod.yaml** – an example of a pod with a web container (nginx) requesting a persistent volume.

1. **Define your KumoScale storage class:** Edit provisioner-storage-class.yaml and set the parameters defined in the [Storage Class Parameters](#) table as needed. For example, if the location of your KumoScale Provisioner is **kumoscale.csi.mycompany.com** and the name of your storage class is **kumoscale-simple-class**:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: kumoscale-simple-class
  namespace: kube-system
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kumoscale.csi.mycompany.com
allowVolumeExpansion: true
<...>

#The space to reserve for snapshot volume changes in (%) from from total size
reservedSpacePercentage: "20"

#specify whether snapshot volumes from this class are writable or not
writable: "true"

#protocol - volume protocol- either NVMeoF or Local , default NVMeoF
protocol: "Local"
#allowSpan - allow spanning volumes across more than one SSD default "true"
allowSpan: "true"
```

2. **Create your KumoScale storage class with:**

```
$ kubectl apply -f provisioner-storage-class.yaml
```

3. **Create a Persistent Volume Claim (PVC) using the storage class.** Using the example file **kube1PVC.yaml**, give it a name,say **kumoscale-simple-volume**, and provide the name of the storage class; in this case **kumoscale-simple-class**. See the Kubernetes site <https://kubernetes.io/docs/concepts/storage/persistent-volumes/> for more information on defining persistent volumes.

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: kumoscale-simple-volume
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: kumoscale-simple-class
```

Create the volume with

```
$ kubectl apply -f kube1PVC.yaml
```

Issue the command below to get details on **kumoscale-simple-volume**:

```
$ kubectl get pvc
```

4. **Create a pod based on the PVC.** Using the example file **testServicePod.yaml**, set **claimName** to **kumoscale-simple-volume** and give the pod a name, in this case **mypod**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mypod
<...>

  volumes:
    - name: www
      persistentVolumeClaim:
        claimName: kumoscale-simple-volume
```

Create the pod with

```
$ kubectl apply -f testServicePod.yaml
```

Issue the command below to get details on **mypod**:

```
$ kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
mypod         Bound     pvc-dd7745d8-21d4-49f2-afec-32db5501a579  10Gi       RWO             kumoscale-simple-volume 13s
```

You may use KumoScale Cluster Manager CLI commands, such as [volume-show](#), to get detailed information on volumes. For example:

```
$ ks_cluster_manager -K <Kubernetes-config-file>

CLI>volume-show --alias pvc-dd7745d8-21d4-49f2-afec-32db5501a579

Volume UUID:          b3140ca6-e765-4642-9fc4-e0b611c50580
Volume Alias:         pvc-dd7745d8-21d4-49f2-afec-32db5501a579
Capacity:             10.0 GiB
```

KumoScale CSI Storage Class Parameters

The table below lists the parameters that may be specified when defining a storage class. You may also enable topology awareness as described after the table under [Enable Topology Awareness](#).

Parameter	Description and Possible Values
name	The storage class name - A string of up to 255 characters. The default name is default.
protocol	The volume's protocol: <ul style="list-style-type: none">NVMe-oF protocol(default).local (for hosted applications). If specified only one replica is allowed, and the hostId must be specified.
provisioningType	Volume type: <ul style="list-style-type: none">thin provisioned. The storage space will be allocated only upon use (i.e., when data is actually written). <ul style="list-style-type: none">thick (default).
numReplicas	The number of replications for a volume created with this storage class. When this parameter equals: <ul style="list-style-type: none">1 (default): A non-resilient volume is created.2 or 3: A volume with 2 or 3 replicas is created accordingly. 3 is the maximum.
sameRackAllowed	Indicates whether more than one replica of a volume may be placed on the same rack. Boolean (true or false). The default value is false.
volumeBindingMode	Indicates whether to wait for the first consumer before binding the volume. This will ensure the volume is accessible to the pod since the binding will be done

Parameter	Description and Possible Values
	according to the node topology (if the node's 'zone' and 'region' labels are configured). WaitForFirstConsumer or Immediate. The default is Immediate.
csi.storage.k8s.io/fstype	Selected filesystem. The possible options are: ext2, ext3, ext4, or xfs. The default value is ext4.
maxIOPSPerGB	Sets an upper limitation on the IOPs on the volume, per volume GB. The default value is 0 – no limit.
desiredIOPSPerGB	The desired amount of IOPs for the volume, per volume GB. The default value is 0 – no specification.
maxBWPerGB	Sets an upper limitation on the bandwidth of the volume IO commands, per volume GB. In kilobytes per second (KB/s). The default value is 0 – no limit.
desiredBWPerGB	The desired amount of bandwidth for the volume IO, per volume GB. In kilobytes per second (KB/s).The default value is 0 – no specification.
blockSize	The block size of the volume and filesystem. In bytes. The default is 4096.
allowVolumeExpansion	Indicates if the volume can be expanded. Boolean (true or false). The default value is false.
reservedSpacePercentage	The space initially reserved for a snapshot volume change log, or for a thin-provisioned volume in percentage of the original volume capacity. The default value is 10. For thin-provisioned volumes the actual reserved space should be larger than 20 gigabytes ^[1] (GB).
writable	Specifies whether a snapshot volume based on this storage class is writable or read-only. Boolean (true or false). The default value is true.
maxReplicaDownTime	The maximum duration in minutes to wait when detecting degradation in the replicated volume before initiating the autonomous self-healing process documented under Failure Recovery in the User Guide . Minimum: 5. Maximum: 1440 (1 day). Default value: 0 (no automatic recovery).
allowSpan	Allow volumes to span more than one SSD. Boolean (true or false). The default value is true.
shareSSDBetweenVolumes	Specifies whether several volumes with this storage class can reside on the same volume. Boolean (true or false). The default value is true.
replicable	Indicates whether the volume can be replicated. Default is false if numReplicas =1, otherwise is true.
logSpace	Defines the required log space in case of snapshot volume as a percentage of the source volume capacity.

Enable Topology Awareness

You can set the volume's topology constraints in a storage class to match a zone or region. This is useful for for resiliency purposes or to ensure pods have access to it. To do this:

1. Set the VolumeBindingMode to **WaitForFirstConsumer** (See [Storage Class Parameters](#)).This will ensure that the Provisioner service allocation will be done when a pod that refers to it starts running for the first time.
2. If you have not set them already, as noted in [Step 1. Prepare for Installation](#), label each Kubernetes node with topology labels:

```
topology.kubernetes.io/region
topology.kubernetes.io/zone
topology.kubernetes.io/rack
```

Note - The CSI driver must be redeployed after labeling nodes.

3. Set the storage node's Region, Rack, and Zone with corresponding values.

Note - A matching storage node with enough free capacity must exist to ensure that a KumoScale software volume is allocated in the required region and zone.

Provisioning Multiple Volumes at Scale

You can provision multiple volumes in a pod by using template files with **kubectl scale**. For example, using the storage class **kumoscale-simple-class** and pod **mypod** from the example of [Using Storage Classes to Create a Simple Volume](#) we can define the file **statefulset.apps** below:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mypod
spec:
  serviceName: "nginx"
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
          ports:
            - containerPort: 80
          name: mypod
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
          livenessProbe:
            exec:
              command:
                - ls
                - /usr/share/nginx/html
            initialDelaySeconds: 5
            periodSeconds: 5
          volumes:
            - name: www
              persistentVolumeClaim:
                claimName: www
      volumeClaimTemplates:
        - metadata:
            name: www
          spec:
            accessModes: [ "ReadWriteOnce" ]
            storageClassName: "kumoscale-simple-class"
            resources:
              requests:
                storage: 10Gi
```

Then issue the command below to create 2 additional replicas

```
$ kubectl scale statefulset mypod --replicas=3
```

Confirm there are now three (3) volumes with

```
$ kubectl get pvc
```

You may also use KumoScale Cluster Manager CLI commands, such as [volume-show](#), to get information on volumes. For example

```
$ ks_cluster_manager -K <Kubernetes-config-file> volume-show
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS	NODES	STORAGECLASS	AGE
mypod-3	Bound	pvc-x-x-x	10Gi	RWQ		kumoscale-simple-class	21s
mypod-2	Bound	pvc-y-y-y	10Gi	RWQ		kumoscale-simple-class	51s
mypod-1	Bound	pvc-z-z-z	10Gi	RWQ		kumoscale-simple-class	91s

Create Multiple Replicas of a Volume

You can add replicas to a volume with the Volume Parameter **numReplicas**. For example

1. Define the storage class **kumoscale-replicated-class** using a new file **replicated-storage-class.yaml** (you can start with the example file **provisioner-storage-class.yaml**) and set **numReplicas=2**

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: kumoscale-replicated-class
  namespace: kube-system
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: kumoscale.csi.kioxia.com
allowVolumeExpansion: true
parameters:
  csi.storage.k8s.io/provisioner-secret-name: kumoscale-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: kube-system
  csi.storage.k8s.io/controller-publish-secret-name: kumoscale-provisioner
```

```
csi.storage.k8s.io/controller-publish-secret-namespace: kube-system
csi.storage.k8s.io/node-stage-secret-name: kumoscale-provisioner
csi.storage.k8s.io/node-stage-secret-namespace: kube-system
csi.storage.k8s.io/node-publish-secret-name: kumoscale-provisioner
csi.storage.k8s.io/node-publish-secret-namespace: kube-system

#csi.storage.k8s.io/fstype: "ext4"

#defines the number of replicas of a volume
numReplicas: "2"
<...>
```

2. Create the storage class **kumoscale-replicated-class**

```
$ kubectl apply -f replicated-storage-class.yaml
```

3. Create a PVC named **kumoscale-replicated-volume** with the file **replicatedPVC.yaml** (you can start with the example file kube1PVC.yaml). Use the storage class **kumoscale-replicated-class** created in step 2 above.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: kumoscale-replicated-volume
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: kumoscale-replicated-class
```

Apply the changes to create the replicas for the volume:

```
$ kubectl apply -f replicatedPVC.yaml
```

Using the KumoScale Cluster Manager volume-show command, you can see that a volume was created with the alias pvc-dd7745d8-21d4-49f2-afec-32db5501a579 and capacity of 20.0 GiB.

```
$ ks_cluster_manager -K ../config volume-show --detail --include Alias --include Capacity

Volume Alias: pvc-dd7745d8-21d4-49f2-afec-32db5501a579
Capacity: 20.0 GiB
```

You can use the Cluster Manager CLI to get details on the volume including that 2 replicas were created, each of capacity 10 GiB.

```
CLI>volume-show --alias pvc-dd7745d8-21d4-49f2-afec-32db5501a579

Volume UUID:          b3140ca6-e765-4642-9fc4-e0b611c50580
Volume Alias:         pvc-dd7745d8-21d4-49f2-afec-32db5501a579
Capacity:             10.0 GiB
Replicas:             2
Replicable:           Yes
Block Size:           4KB
Snapshot UUID:        None
In Use:               Used
Protocol:             NVMeoF
Span Allowed:         Yes
Provisioning Type:    Thick
Max Replica DownTime: 180 min
Max BW:               0.0 KB/sec
Desired BW:           0.0 KB/sec
Max IOPS:             0
Desired IOPS:         0
Storage Class Name:   default
Share SSD Between Volumes: Yes
LOCATION:
  Replica UUID:       ea9f51d5-d1f5-4c1d-a379-d2eb1eaa4895
  StorageNode Name:   storagenode2
  Replica State:      Available
  Current State Time: 105940 msec
LOCATION:
  Replica UUID:       a93af020-4d23-44e1-b6a9-aed3a077ecc5
  StorageNode Name:   storagenode1
  Replica State:      Available
  Current State Time: 103239 msec
```

4. We can scale this to create replicas for multiple pods (containers) using **statefulset.apps** with the **kumoscale-replicated-volume** storage class.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mypod
spec:
  serviceName: "nginx"
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
      ports:
        - containerPort: 80
```

```
name: mypod
volumeMounts:
- name: www
mountPath: /usr/share/nginx/html
livenessProbe:
exec:
command:
- ls
- /usr/share/nginx/html
initialDelaySeconds: 5
periodSeconds: 5
volumes:
- name: www
persistentVolumeClaim:
claimName: www
volumeClaimTemplates:
- metadata:
name: www
spec:
  accessModes: [ "ReadWriteOnce" ]
  storageClassName: "kumospace-replicated-volume"
resources:
requests:
storage: 10Gi
```

To create 4 pods, each with 2 volumes:

```
$ kubectl scale statefulset mypod --replicas=4
```

Confirm that four pods are running

```
$ kubectl get pvc
NAME          STATUS  VOLUME                                     CAPACITY  ACCESS  MODES  STORAGECLASS          AGE
www-mypod-0   Bound   pvc-dd7745d8-21d4-49f2-afec-32db5501a579  10Gi       RWO     kumospace-replicated-class  13m
www-mypod-1   Bound   pvc-a0e9915d-88a2-4056-961a-3dccea534c94  10Gi       RWO     kumospace-replicated-class  9m49s
www-mypod-2   Bound   pvc-1fd56d97-9606-4294-8c86-284194e44856  10Gi       RWO     kumospace-replicated-class  9m39s
www-mypod-3   Bound   pvc-0f990e90-e1e5-4964-9093-83e571054d34  10Gi       RWO     kumospace-replicated-class  9m23s
```

Confirm that each have two replicated volumes

```
$ ks_cluster_manager -K ../config volume-show --detail --include Alias --include ReplicaVolume Alias:
pvc-1fd56d97-9606-4294-8c86-284194e44856 Replicas: 2
Replicable: Yes
Max Replica DownTime: 180 min

  Replica UUID: 97541115-c86c-44cd-9f2a-a221acf8d7c8
  Replica State: Available
  Replica UUID: e8670f16-6ec4-4950-bf63-11a587e3bfe0
  Replica State: AvailableVolume Alias: pvc-0f990e90-e1e5-4964-9093-83e571054d34
Replicas: 2
Replicable: Yes
Max Replica DownTime: 180 min
  Replica UUID: 6b0266eb-92b2-42d3-a004-cf16c0efef31
  Replica State: Available
  Replica UUID: 8aab4f24-5d28-4690-83a2-26855ccb1696
  Replica State: AvailableVolume Alias: pvc-dd7745d8-21d4-49f2-afec-32db5501a579
Replicas: 2
Replicable: Yes
Max Replica DownTime: 180 min
  Replica UUID: ea9f51d5-d1f5-4c1d-a379-d2eb1eaa4895
  Replica State: Available
  Replica UUID: a93af020-4d23-44e1-b6a9-aed3a077ecc5
  Replica State: AvailableVolume Alias: pvc-a0e9915d-88a2-4056-961a-3dccea534c94
Replicas: 2
Replicable: Yes
Max Replica DownTime: 180 min
  Replica UUID: dd91db1a-7363-4727-b284-19b0d1fd71d8
  Replica State: Available
  Replica UUID: f2e1af5f-77d1-42be-a8ff-4ef815ae78fa
  Replica State: Available
```

Expanding a Volume

KumoScale software supports volume online expansion for any type of volume including snapshot volumes and clones. In order to expand a volume you must have set up the plug-in as described in [Prepare the Environment for Features in Beta](#) in [Installing the KumoScale CSI Driver](#) and the PVC must be attached to a pod by following the steps below:

1. Edit the PVC using the following command

```
$ kubectl edit pvc <name of PVC>
```

For example using the www-mypod-1 pod created in [Create multiple replicas for a volume](#)

```
$ kubectl edit pvc www-mypod-1
```

2. Increase the PVC capacity (under spec: resources: requests: storage). For example, to increase to 10Gi:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: kumospace-replicated-volume
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```
storageClassName: kumospace-replicated-class
<...>
```

3. Save and exit. Allow 1-2 minutes for the actual expansion to take place; kubectl needs to detect the change and execute the expansion.

4. Verify that the volume is now 20Gi. For example:

```
$ kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
www-mypod-1   Bound    pvc-a0e9915d-88a2-4056-961a-3dccea534c94  20Gi       RWO            kumospace-replicated-class
```

Deleting a Volume

A source volume can be deleted even if it has snapshot volumes or clone volumes. Before deleting a volume, verify no pod is currently using it. Then delete the volume using kubectl delete with the name of the yaml file used to create the volume. For example, to delete the volume **kumospace-simple-volume** created by **kube1PVC.yaml**

```
$ kubectl describe pod mypod
```

Verify the volume is not used by the pod.

```
$ kubectl delete -f kube1PVC.yaml
```

You can also delete the pod and all its volumes with:

Get status on the pod:

```
$ kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www-web-0     Bound    pvc-dd7745d8-21d4-49f2-afec-32db5501a579  10Gi       RWO            kumospace-replicated  13m
www-web-1     Bound    pvc-a0e9915d-88a2-4056-961a-3dccea534c94  10Gi       RWO            kumospace-replicated  9m49
www-web-2     Bound    pvc-1fd56d97-9606-4294-8c86-284194e44856  10Gi       RWO            kumospace-replicated  9m39
www-web-3     Bound    pvc-0f990e90-e1e5-4964-9093-83e571054d34  10Gi       RWO            kumospace-replicated  9m23
```

To delete www-web-2

```
$ kubectl delete pvc www-web-2
persistentvolumeclaim "www-web-2" deleted
```

Cloning a Volume

To clone a volume, you need to create a PVC with a data source that refers to an existing PVC. For example, we can use the sample file, **CloneVolume.yaml** and define **clone-simple-volume**, a clone of the PVC **kumospace-simple-volume** defined above in Using Storage Classes to Create a Simple Volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: clone-simple-volume
spec:
  dataSource:
    name: kumospace-simple-volume
    kind: PersistentVolumeClaim
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: kumospace-simple-class
```

You can also create clones of equal or greater quantity to the original volume. This is since CSI allows allocation via a capacity range. If the value is greater than the source volume, the KumoScale Provisioner will extend the created volume to the required capacity.

Snapshot volumes and clones are independent. This means that a source volume can be deleted even if it has snapshot volumes or clone volumes.

Extended Resources

Kubernetes allows administrators to specify key, value pairs on each node as an extended resource. Administrators can then create pods with requests for a specific number of extended resources. KumoScale uses this feature to influence where pods are scheduled depending on storage nodes constraints.

For example, the pod below asks for two (2) volumes of 200 GB on two (2) separate SSDs:

```
apiVersion: v1
kind: Pod
metadata:
  name: extended-resource-demo2
spec:
  containers:
    - name: extended-resource-demo-ctr
      image: nginx
      resources:
        requests:
          kumospace.kioxia.com/availableSpaceFor2VolumesInGB: 200
        limits:
          kumospace.kioxia.com/availableSpaceFor2VolumesInGB: 200
```

Volume Migration

See the [Volume Migration](#) section of the KumoScale User Guide.

Next:[Load Testing with FIO Loadgen](#)
