

KumoScale CSI Driver Tenant Management

This section provides an overview to how KumoScale supports multiple tenants with examples using custom resource files.

- [Using Virtual Clusters to Support Multiple Tenants](#)
- [Tenant Parameters](#)
- [Creating tenants](#)
- [Adding the tenant to your Kubernetes cluster](#)
- [Modifying tenants](#)
- [Deleting tenants](#)

Using Virtual Clusters to Support Multiple Tenants

KumoScale software provides the ability to separate different applications, customers, or workloads by creating virtual clusters known as **tenants**. Each tenant is allocated a certain capacity and performance target. These constraints are validated and updated during each volume allocation, expansion, and deletion and when adding a new tenant. If a tenant reaches its capacity, a new volume will not be allocated. If a tenant reaches its performance target, a new volume may be created, but the performance Service Level Agreement (SLA) is not guaranteed.

Tenants are managed and created by the KumoScale Provisioner service and are identified by a unique value, the **tenantID**. This ID is required for all volume management and monitoring commands to maintain isolation between different customers and applications. Each tenant can only monitor and manage its own volumes and has no visibility to the other tenant volumes.

If no tenant is declared by the user, KumoScale creates a **Global Tenant** with capacity and performance equivalent to the sum of all connected KumoScale deployments. If this tenant is used, no information is needed when managing volumes.

Tenant Management using Custom Resource Definitions

KumoScale software provides a Tenant CRD file for specifying tenants, a template is provided at **KumoScale_Operator/ks-config-operator/samples/kumoscale_v1_tenant_cr.yaml**. You can create a new tenant or modify or delete an existing tenant using the **kubect**l **create**, **apply**, or **delete** commands with the CRD. The table below shows the parameters supported in the tenant CRD.

Tenant Parameters

tenant Parameter	Description	Optional/Required
name	The tenant configuration name; must comply with the Name field as defined in KumoScale Field Types .	Required
tenantId	Requested tenant ID	Optional for create operations. Required for update and delete.
capacity	In GB unless otherwise specified; for example, 100Gi	Required
totalIOPS	The number of IOPS; an integer value	Required
totalBWPerSec	Allowed bandwidth in resources in MB unless otherwise specified; for example, 100Mi	Required

To create a tenant in your cluster you will need to

- [Create the tenant](#)
- [Add the tenant to your cluster](#)

Creating a new tenant

- Make a copy of the sample tenant CR file **Kumoscale_Operator/ks-config-operator/samples/kumoscale_v1_tenant_cr.yaml**. For example `deploy/crds/myapp_tenant1_cr.yaml`.
- Edit **myapp_tenant1_cr.yaml** and specify values for tenant parameters as shown in [Tenant Parameters](#). For example, with name **mytenant** and tenantId **tenant1**.

```
apiVersion: kumoscale.kioxia.com/v1
kind: Tenant
metadata:
  name: mytenant
spec:
  #optional
  tenantId:tenant1
  budget:
    #capacity in GB
    capacity: 100Gi
```

```
totalIOPS: 10000
#in MB
totalBWPerSec: 1000Mi
```

3. Create the new tenant with

```
kubectl create -f myapp_tenant1_cr.yaml
```

A new tenant will be created if there are no identical tenants with the same ID. Capacity and performance requirements are tested only when allocating a new volume.

To validate the changes were successful for a tenant with id **tenant1**, enter one of the following commands:

```
kubectl get tenants tenant1 -o wide
kubectl describe tenants tenant1
```

The status is refreshed every 30 seconds.

Adding the Tenant to your Kubernetes Cluster

Once you have created a tenant, you need to add it to your Kubernetes cluster by following the steps below:

1. Edit the Provisioner secret. For example,for our tenant with ID **tenant1** we would specify it as the value of **tenantID** in the secret file (see the example **provisioner-secret.yaml**).

```
apiVersion: v1
kind: Secret
metadata:
  name: kumoscale-provisioner
  namespace: kube-system
type: Opaque
stringData:
  config.yaml: |-
    url: https://<ip address>:<port>
    token: kdjfs ...
    tenantID: tenant1
#  authServerTokenUrl: http://someAuthserver
#  provisionerClientID: ****
#  provisionerClientScope: ****
#  provisionerClientSecret: ***
#  storagenodeClientID: ***
#  storagenodeClientSecret: ****
#  storagenodeClientScope: ****
```

2. Apply the change with

```
kubectl apply -f provisioner-secret.yaml
```

Modifying an Existing Tenant

To modify an existing tenant, say **tenant1** above, defined in myapp_tenant1_cr.yaml.

1. Edit the CRD,

```
kubectl edit -f myapp_tenant1_cr.yaml
```

2. Change the settings as needed and save the file. Then apply the changes:

```
kubectl apply -f myapp_tenant1_cr.yaml
```

3. To validate the changes were successful, enter one of the following commands and verify the change was made:

```
kubectl get tenants tenant1 -o wide
kubectl describe tenants tenant1
```

The status is refreshed every 30 seconds.

Deleting a Tenant

A tenant can be deleted only if there are no volumes using it. To delete **tenant1**, the tenant specified above in **myapp_tenant1_cr.yaml**, enter:

```
kubectl delete -f myapp_tenant1_cr.yaml
```

To validate the changes were successful, enter one of the following commands:

```
kubectl get tenants tenant1 -o wide
kubectl describe tenants tenant1
```

You should get a message indicating **tenant1** does not exist.
