Defining Storage Classes

In this lesson, we will look into defining our own Storage Class.

WE'LL COVER THE FOLLOWING

Even though kops created two StorageClasses, both are based on `gp2`. While that is the most commonly used EBS type, we might want to create volumes based on one of the other three options offered by AWS.

User-Defined Storage Classes

Let's say that we want the fastest EBS volume type for our Jenkins. That would be `io1`. Since kops did not create a StorageClass of that type, we might want to create our own.

Looking into the Definition

YAML file that creates StorageClass based on EBS `io1` is defined in `pv/sc.yml`. Let's take a quick look.

```
cat pv/sc.yml
```

The **output** is as follows.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: fast
  labels:
    type: ebs
```

```
    type: ebs
provisioner: kubernetes.io/aws-ebs
parameters:

  type: type: io1 # https://aws.amazon.com/ebs/details/ > Amazon EBS Volume Types
reclaimPolicy: Delete
```

We used `kubernetes.io/aws-ebs` as the `provisioner`. It is a mandatory field that determines the plugin that will be used for provisioning PersistentVolumes. Since we are running the cluster in AWS, `aws-ebs` is the logical choice.

List of Supported Provisioners

There are quite a few other provisioners we could choose. Some of them are specific to a hosting provider (e.g., `GCEPersistentDisk` and `AzureDisk`) while others can be used anywhere (e.g., `GlusterFS`).

The list of supported provisioners is growing. At the time of this writing, the following types are supported.

Volume Plugin	Internal Provisioner
AWSElasticBlockStore	yes
AzureFile	yes
AzureDisk	yes
CephFS	no
Cinder	yes
FC	no
FlexVolume	no
Flocker	yes
GCEPersistentDisk	yes

Glusterfs	yes
iSCSI	no
PhotonPersistentDisk	yes
Quobyte	yes
NFS	no
RBD	yes
VsphereVolume	yes
PortworxVolume	yes
ScaleIO	yes
StorageOS	yes
Local	no

The internal provisioners are those with names prefixed with `kubernetes.io` (e.g., `kubernetes.io/aws-ebs`). They are shipped with Kubernetes.

External provisioners, on the other hand, are independent programs shipped separately from Kubernetes. An example of a commonly used external provisioner is `NFS`.

The parameters depend on the StorageClass. We used the `aws-ebs` provisioner which allows us to specify the `type` parameter that defines one of the supported Amazon EBS volume types. It can be:

1. EBS Provisioned IOPS SSD (`io1`)
2. EBS General Purpose SSD (`gp2`)
3. Throughput Optimized HDD (`st1`)
4. Cold HDD (sc1)

We set it to `io1` which is the highest performance SSD volume.

> ℹ Please consult Parameters section of the *Storage Classes* documentation for more info.

Delete type Reclaim Policy

Finally, we set the `reclaimPolicy` to `Delete`. Unlike `Retain` that forces us to delete the contents of the released volume before it becomes available to new PersistentVolumeClaims, `Delete` removes both the PersistentVolume as well as the associated volume in the external architecture. The `Delete` reclaim policy works only with some of the external volumes like AWS EBS, Azure Disk, or Cinder volume.

Now that we dipped our toes into the StorageClass definition, in the next lesson, we can proceed and create it.