Creating Storage Classes

In this lesson, we will create the Storage Class defined in the previous lesson and verify it.

WE'LL COVER THE FOLLOWING

Creating the Class

Let's create our StorageClass.

```
kubectl create -f pv/sc.yml
```

The **output** shows that the `storageclass "fast"` was `created`.

Verification

We'll list the StorageClassses in our cluster.

```
kubectl get sc
```

The **output** is as follows.

```
NAME          PROVISIONER           AGE
default       kubernetes.io/aws-ebs 58m
fast          kubernetes.io/aws-ebs 19s
gp2 (default) kubernetes.io/aws-ebs 58m
```

We can see that, this time, we have a new StorageClass

We can see that, this time, we have a new StorageClass.

Creating the Deployment

Let's take a look at yet another Jenkins definition.

```
cat pv/jenkins-sc.yml
```

The **output**, limited to the relevant parts, is as follows.

```
...
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: jenkins
  namespace: jenkins
spec:
  storageClassName: fast
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 4Gi
...
```

The only difference, when compared with the previous definition, is that we are now using the newly created StorageClass named `fast`.

Finally, we'll confirm that the new StorageClass works by deploying the new `jenkins` definition.

```
kubectl apply \
    -f pv/jenkins-sc.yml \
    --record
```

The **output** is as follows.

```
namespace "jenkins" configured
ingress "jenkins" configured
service "jenkins" configured
persistentvolumeclaim "jenkins" created
deployment "jenkins" created
```

Verification

As the final verification, we'll list the EBS volumes and confirm that a new one

As the final verification, we'll list the EBS volumes and confirm that a new one was created based on the new class.

```
aws ec2 describe-volumes \
    --filters 'Name=tag-key,Values="kubernetes.io/created-for/pvc/name"'
```

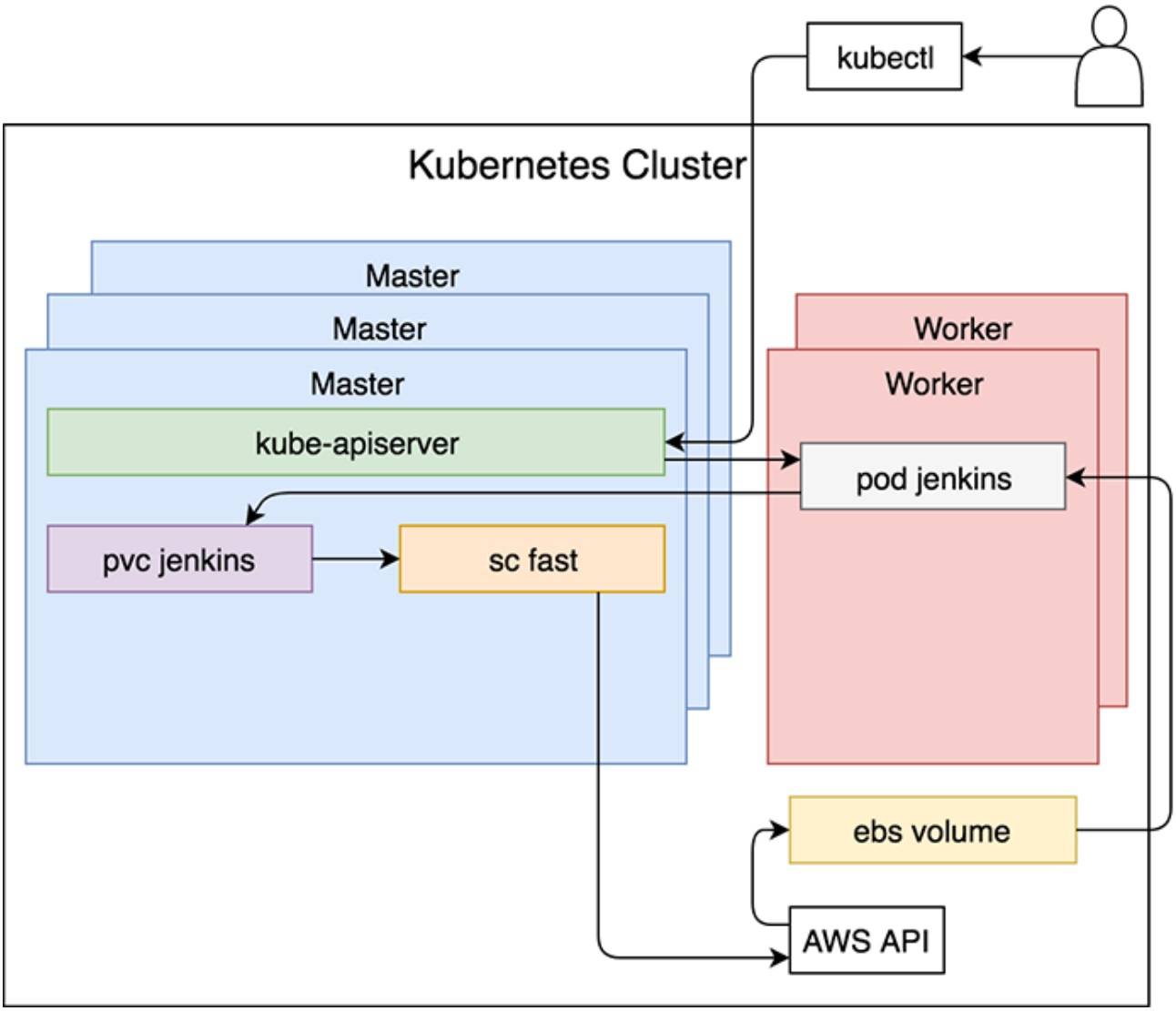
The **output**, limited to the relevant parts, is as follows.

```
{
  "Volumes": [
    {
      ...
      "VolumeType": "io1",
      "VolumeId": "vol-0e0af4f2a7a54354d",
      "State": "in-use",
      ...
   }
  ]
}
```

We can see that the type of the newly created EBS volume is `io1` and that it is `in-use`.

Sequential Breakdown of the Process

A simplified version of the flow of events initiated with the creation of the `jenkins` Deployment is as follows.



The sequence of events initiated with a request to create a Jenkins Pod with the PersistentVolumeClaim using a custom StorageClass

1. We created the `jenkins` Deployment, which created a ReplicaSet, which, in turn, created a Pod.
2. The Pod requested persistent storage through the PersistentVolumeClaim.
3. The PersistentVolumeClaim requested PersistentStorage with the StorageClass named `fast`.
4. StorageClass `fast` is defined to create a new EBS volume, so it requested one from the AWS API.
5. AWS API created a new EBS volume.
6. EBS volume was mounted to the `jenkins` Pod.

We're finished exploring persistent volumes. You should be equipped with the

We're finished exploring persistent volumes. You should be equipped with the knowledge how to persist your stateful applications, and the only pending action is to remove the volumes and the cluster.

In the next lesson, we will test our understanding regarding the Persisting State of an application with the help of a quick quiz.