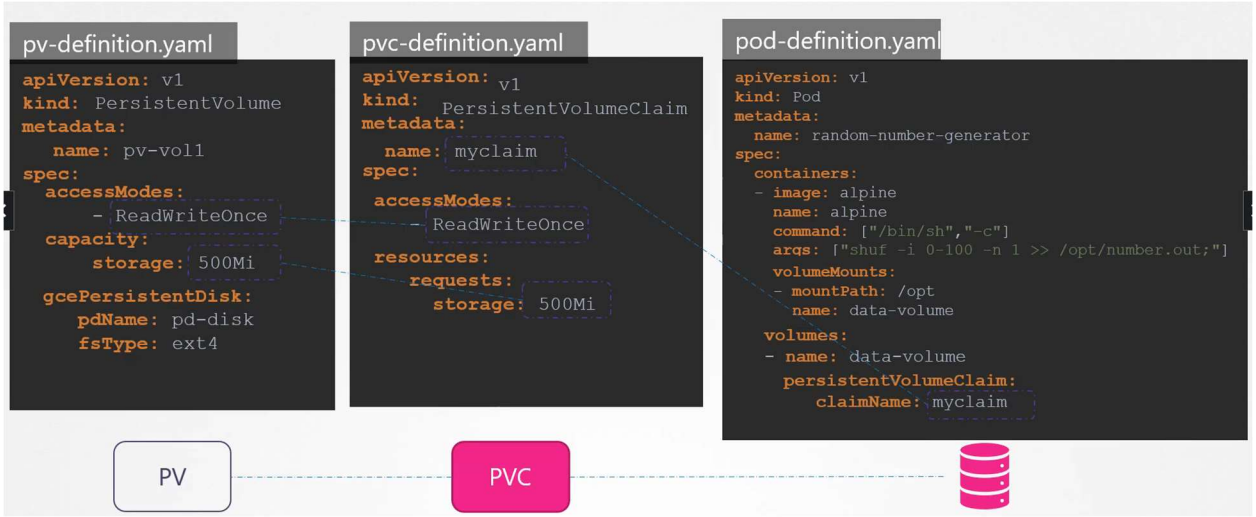
In the previous lectures, we discussed about how to create PVs, and then create PVCs to claim that storage, and then use the PVCs in the pod definition files as volumes.



In this case, we create a PV from a Google Cloud persistent disk. The problem here is that before this PV is created, you must have created the disk on Google Cloud. Every time an application requires storage, you have to first manually provision the disk on Google Cloud, and then manually create a persistent volume definition file using the same name as that of the disk that you created. That's called static provisioning volumes.

```
gcloud beta compute disks create \
    --size 1GB
    --region us-east1
    pd-disk
```

pv-definition.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-vol1
spec:
  accessModes:
      - ReadWriteOnce
  capacity:
      storage: 500Mi
  gcePersistentDisk:
     pdName: pd-disk
     fsType: ext4
```

It would've been nice if the volume gets provisioned automatically when the application requires it, and that's where storage classes come in.

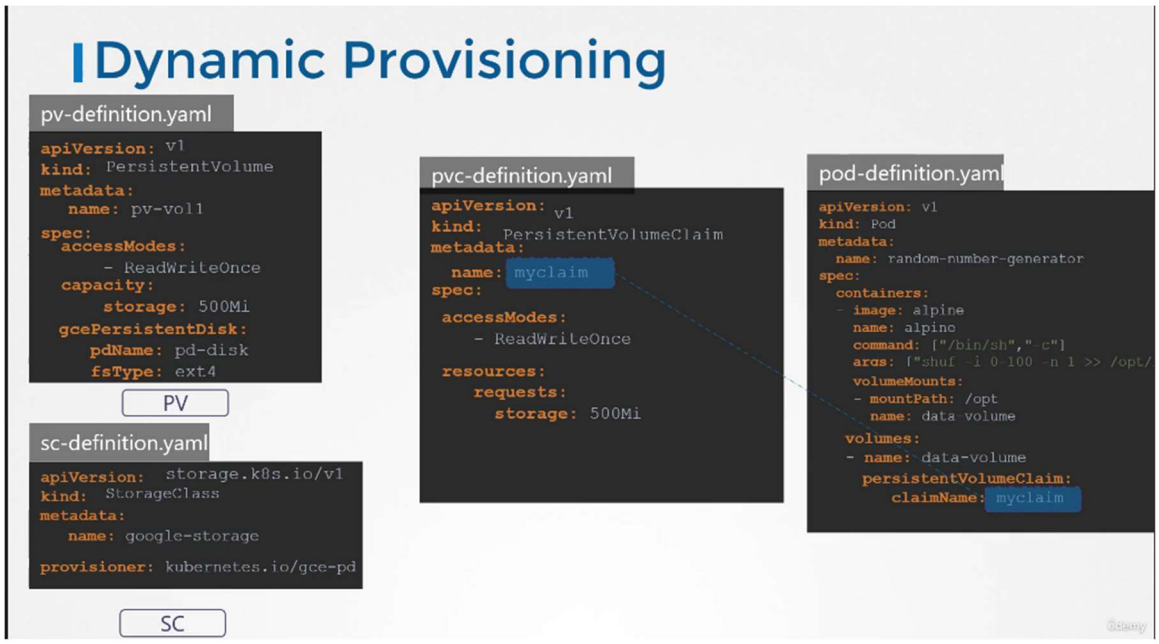
With storage classes, you can define a provisioner, such as Google Storage, that can automatically provision storage on Google Cloud and attach that to pods when a claim is made. That's called dynamic provisioning of volumes.

You do that by creating a storage class object with the API version set to storage.k8.io/v1, specify a name, and use provisioner as Kubernetes.io/gce-pd

sc-definition.yaml

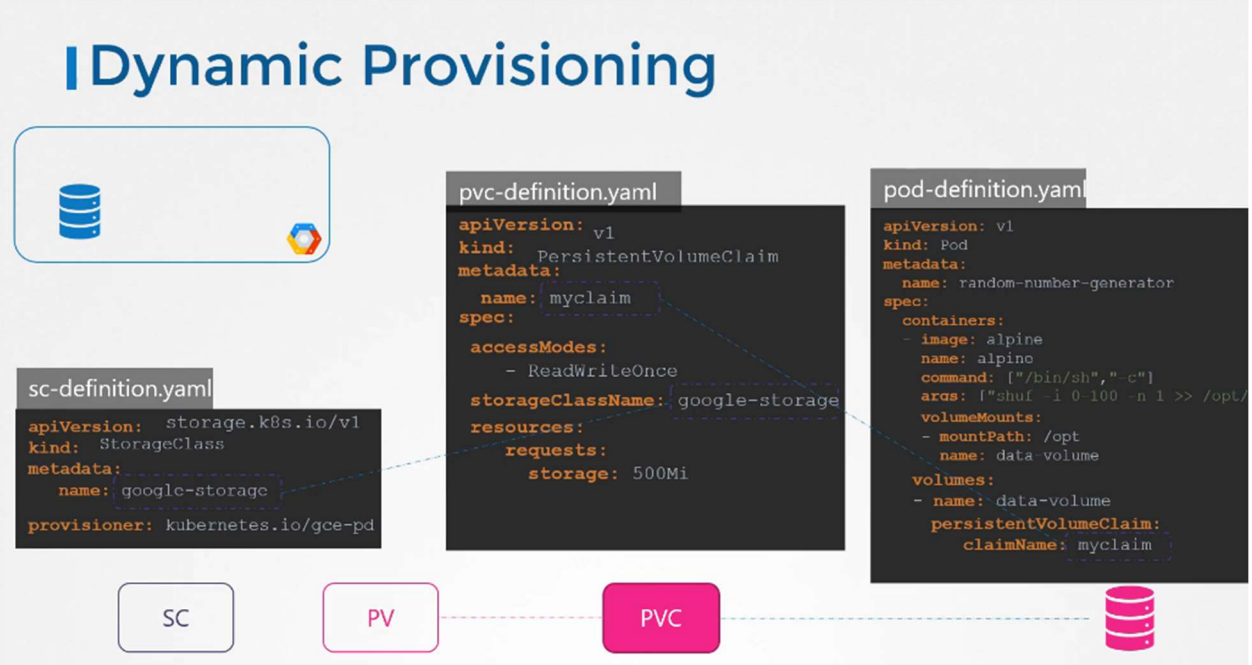
```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: google-storage
provisioner: kubernetes.io/gce-pd
```

So going back to our original state where we have a pod using a PVC for its storage, and the PVC is bound to a PV, we now have a storage class, so we no longer need the PV definition, because the PV and any associated storage is going to be created automatically when the storage class is created.

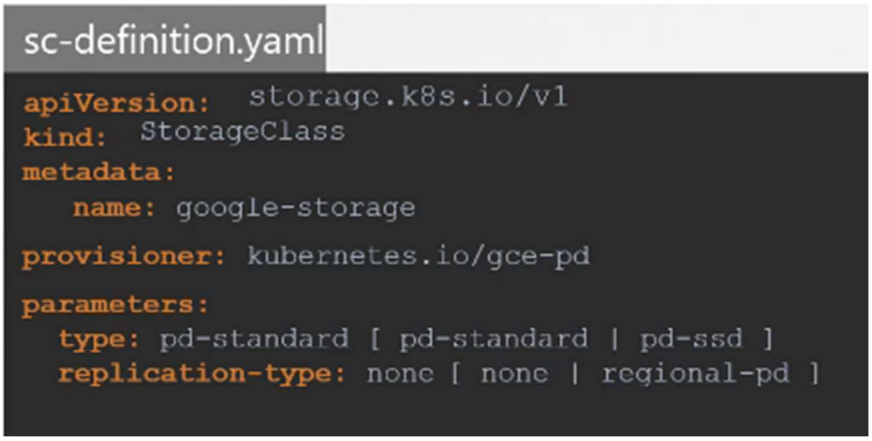


For the PVC to use the storage class we defined, we specify the storage class name in the PVC definition. That's how the PVC knows which storage class to use. Next time a PVC is created, the storage class associated with it uses the defined provisioner to provision a new disk with the

required size on GCP, and then creates a persistent volume, and then binds the PVC to that volume. So remember that it still creates a PV, it's just that you don't have to manually create PV anymore. It's created automatically by the storage class.



We used the GCE provisioner to create a volume on GCP. There are many other provisioners as well, such as, for **AWS EBS, Azure File, Azure Disk, CephFS, Portworx, ScaleIO**, and so on. With each of these provisioners, you can pass in additional parameters, such as the type of disk to provision, the replication type, et cetera. These parameters are very specific to the provisioner that you are using. For Google Persistent Disk, you can specify the type, which could be standard, or SSD. You can specify the replication mode, which could be none, or regional PD.



So you see, you can create different storage classes, each using different types of disks. For example, a silver storage class with the standard disks, a gold class with SSD drives, and a platinum

class with SSD drives and replication. And that's why it's called storage class. You can create different classes of service. Next time you create a PVC, you can simply specify the class of storage you need for your volumes.