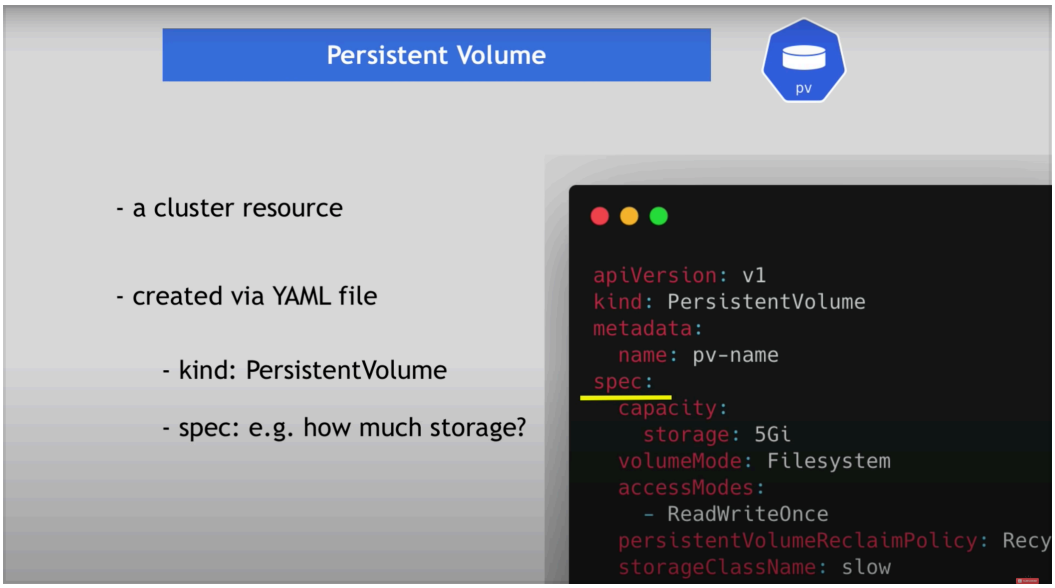
Kubernetes Volumes

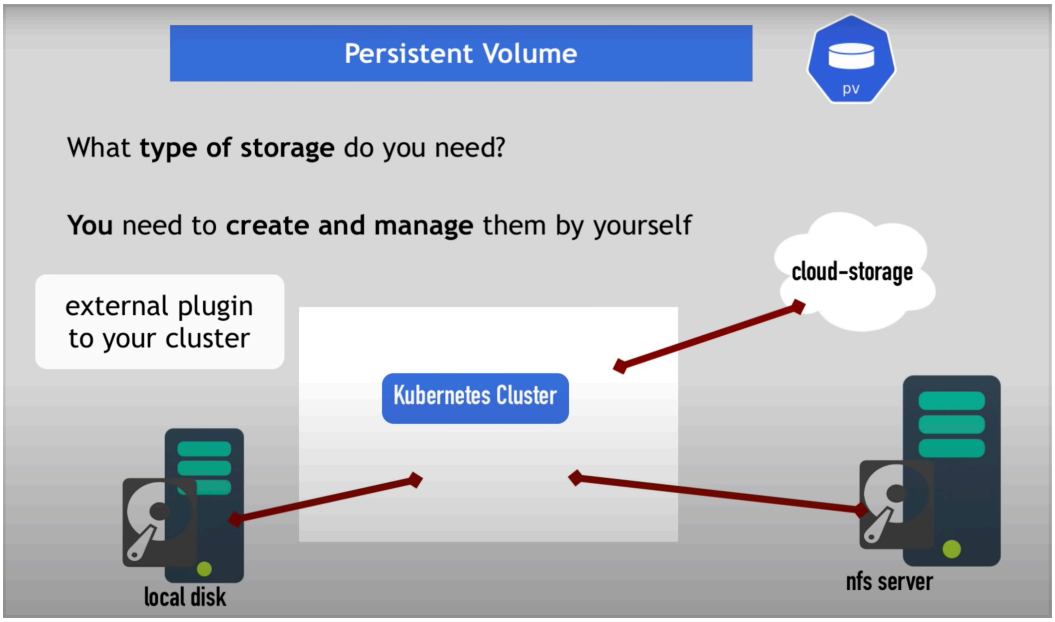
- The need for Volumes
  - When the pod restarts, could be because of any cases, all the data will be gone.
  - In k8s there is no data persistence out of the box, you need to configure it for each application that needs saving data b/n pod restarts.
  - So to persist data there is a component or concept called Volumes

  - Storage Requirements
    - How to persist data in K8s using volumes
    - Basically you need a storage that doesn't depend on the pod life cycle, so it will still be there when pod dies and new one gets created, so the new pod can pick up where the old pod left off.so, it will read the existing data from that storage to get upto date data.
    - However you don't know on which node the new pod restarts,
      - so storage must be available on all nodes, not just one specific one.\
    - Needs an highly available storage as it needs to survive even if cluster crashes

  - Another use case for persistent storage
    - Which is not for db but it for a directory, like you have an appn that writes/reads files from pre config directory, could be session files, config files etc

  - So, you can configure any of this types of storage in k8s with "Persistent Volume"
  - Think of it like a cluster resource like CPU, RAM etc that is used to store data.
  - Gets created with YAML file





- Can configure multiple storage for you cluster =, like one application uses local storage, one uses cloud and one uses nfs server and one could use all of these storage
- Volumes should be created before as PV are resources that need to be there Before

Local vs. Remote Volume Types

Each volume type has it's own use case!

Local volume types violate 2. and 3. requirement for data persistence:

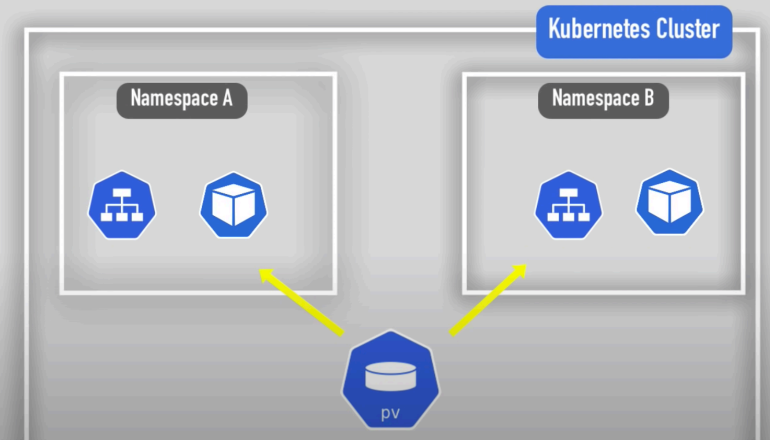
- ❌ Being tied to 1 specific node
- ❌ surviving cluster crashes

For DB persistence use remote storage!

Persistent Volumes are **NOT namespaced**

PV outside of the namespaces

Accessible to the whole cluster



Persistent Volume YAML Example

Depending on storage type,
spec attributes differ

Node Affinity:

Local storage

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/ssd1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

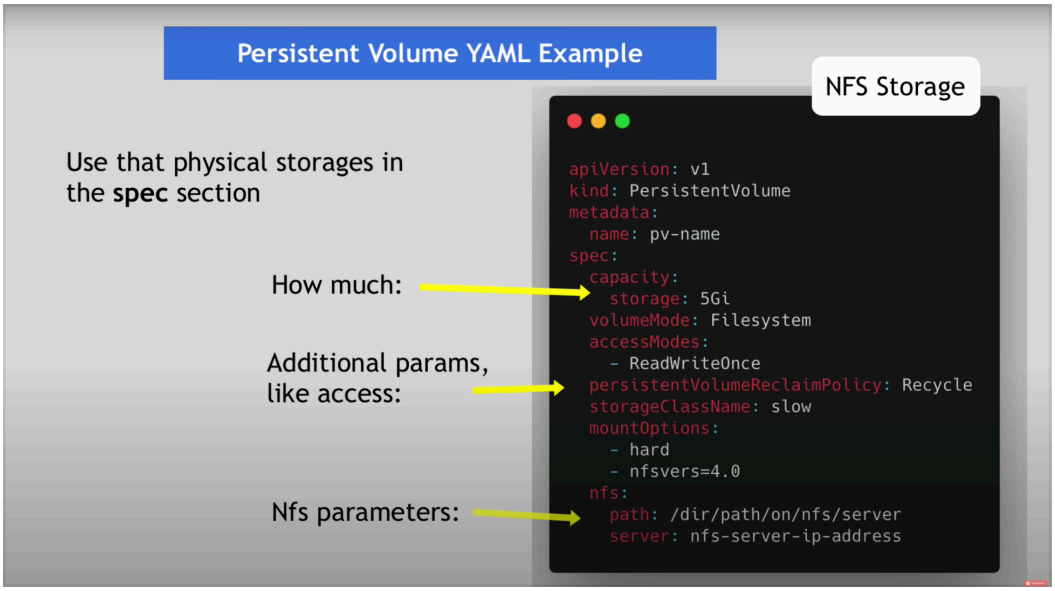
Persistent Volume YAML Example

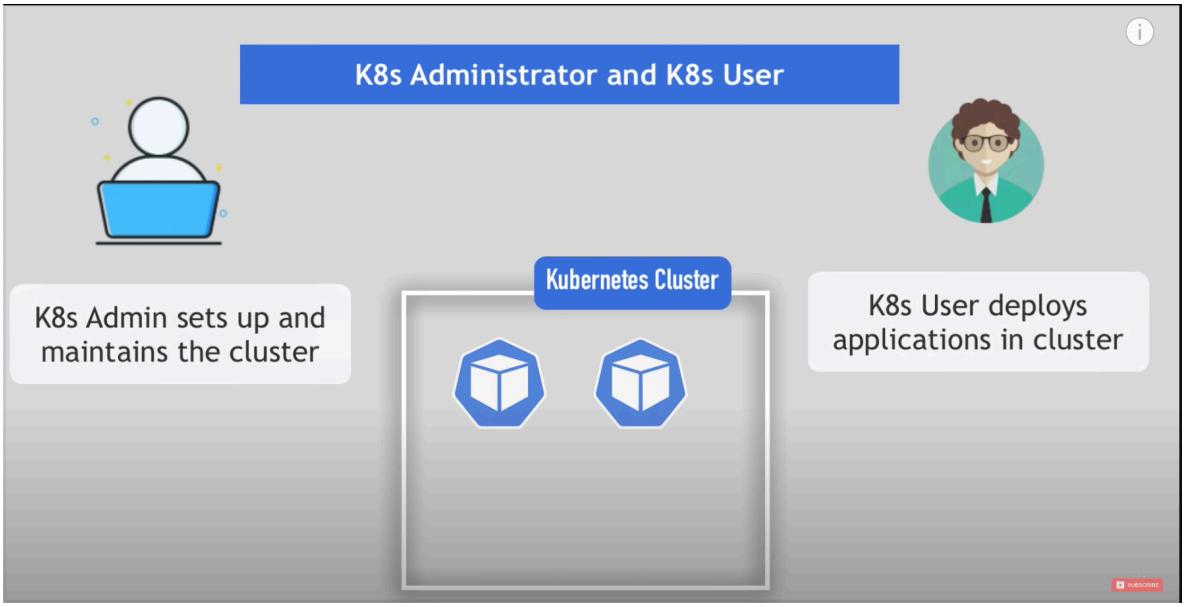
How much:

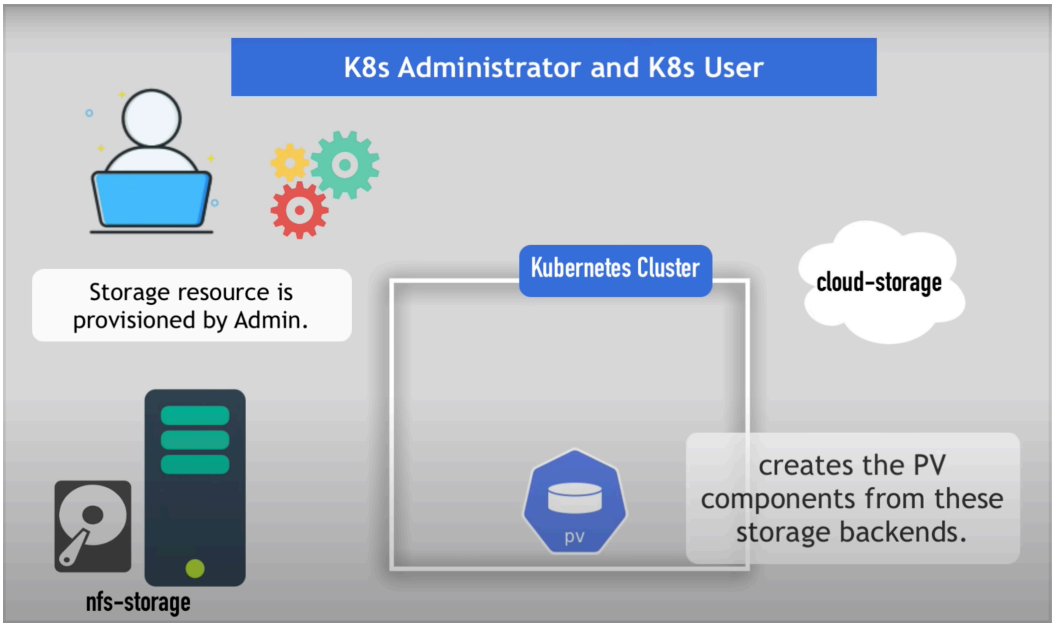
Google Cloud
parameters:

Google Cloud

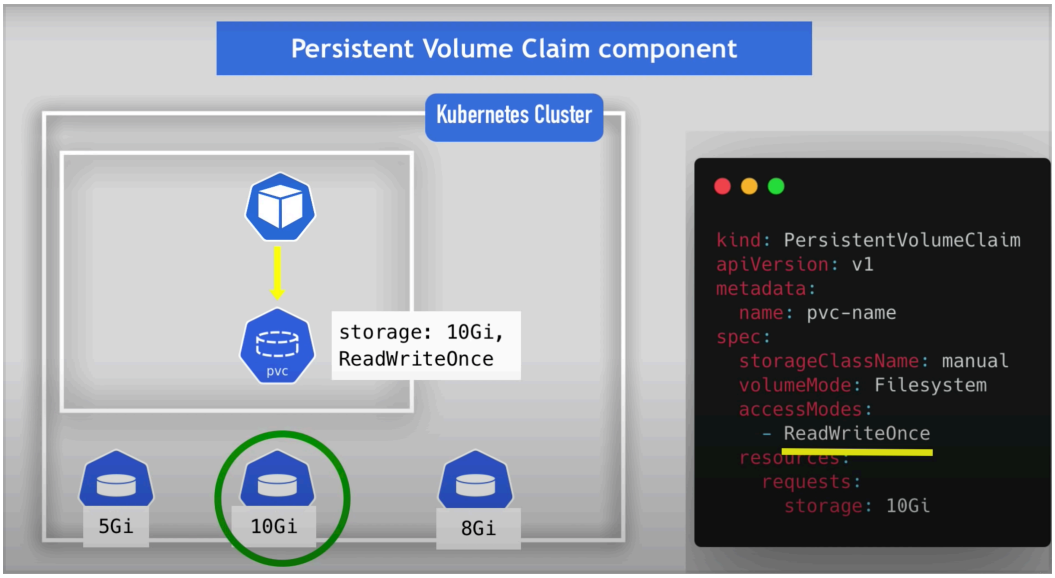
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: test-volume
  labels:
    failure-domain.beta.kubernetes.io/zone: us-central1-a__us-central1-b
spec:
  capacity:
    storage: 400Gi
  accessModes:
  - ReadWriteOnce
  gcePersistentDisk:
    pdName: my-data-disk
    fsType: ext4
```

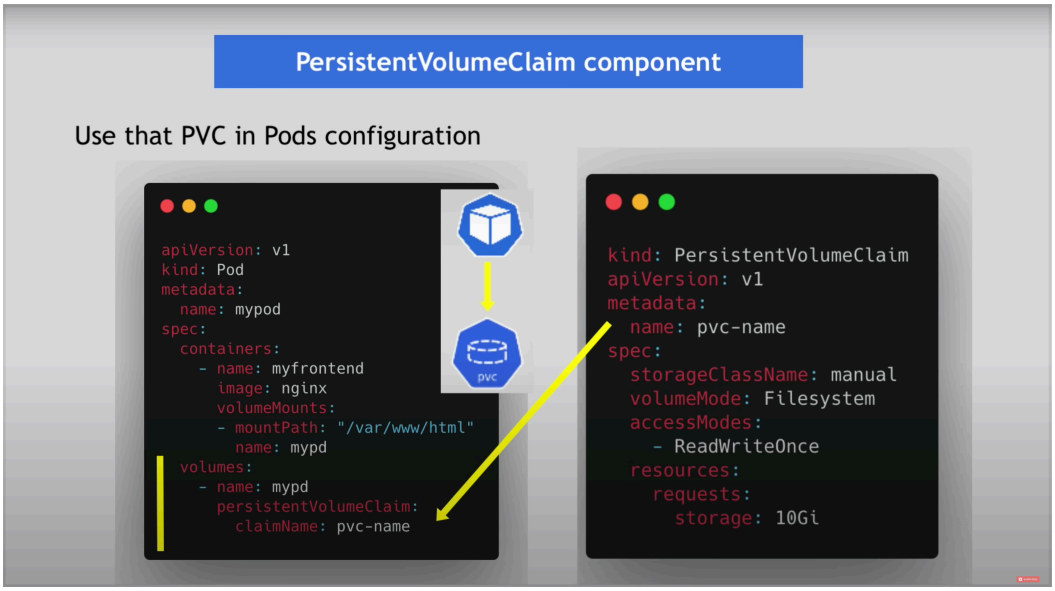

Persistent Volume YAML Example
NFS Storage
Use that physical storages in the spec section
How much:
Additional params, like access:
Nfs parameters:
apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-name
spec:
capacity:
storage: 5Gi
volumeMode: Filesystem
accessModes:
- ReadWriteOnce
persistentVolumeReclaimPolicy: Recycle
storageClassName: slow
mountOptions:
- hard
- nfsvers=4.0
nfs:
path: /dir/path/on/nfs/server
server: nfs-server-ip-address


K8s Administrator and K8s User
K8s Admin sets up and maintains the cluster
Kubernetes Cluster
K8s User deploys applications in cluster

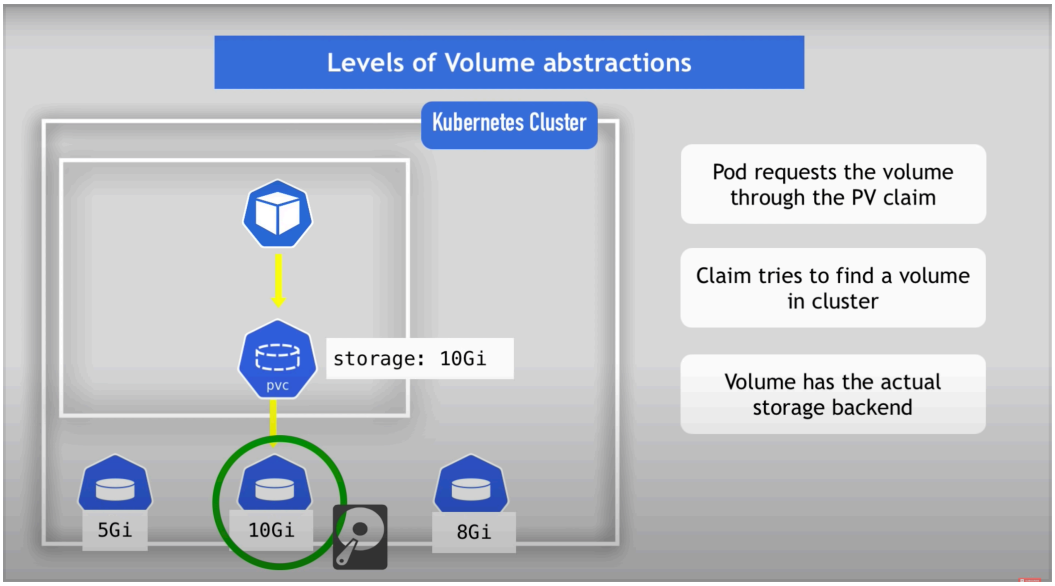

K8s Administrator and K8s User
Storage resource is provisioned by Admin.
Kubernetes Cluster
cloud-storage
pv
creates the PV components from these storage backends.
nfs-storage

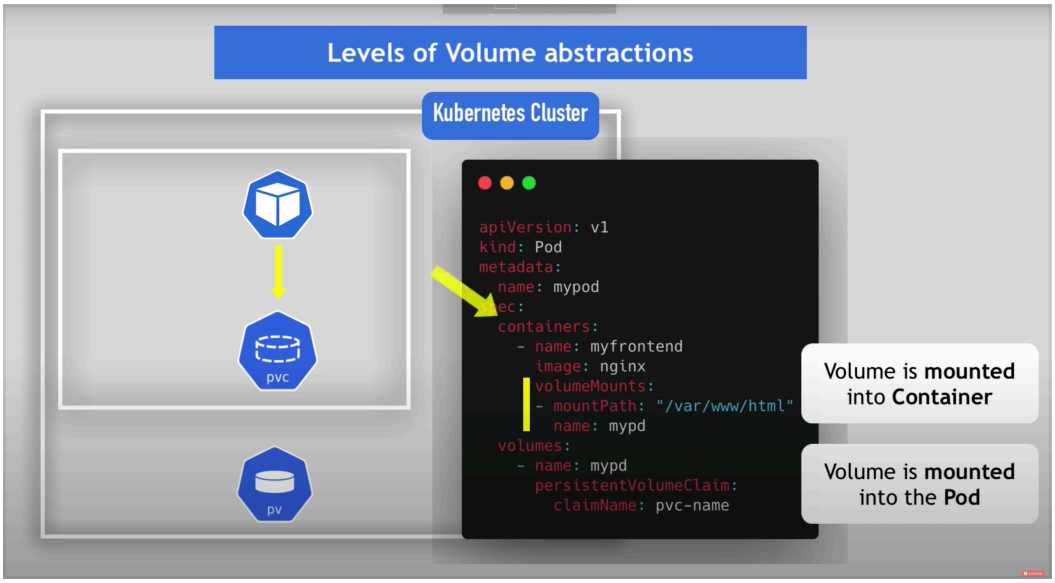
- Once the administrator creates and configure the Volume (like google cloud storage), developers has to explicitly configure the application (pods) YAML files to use those PV components in other words application has to claim that PV (PVC) using another component called PVC



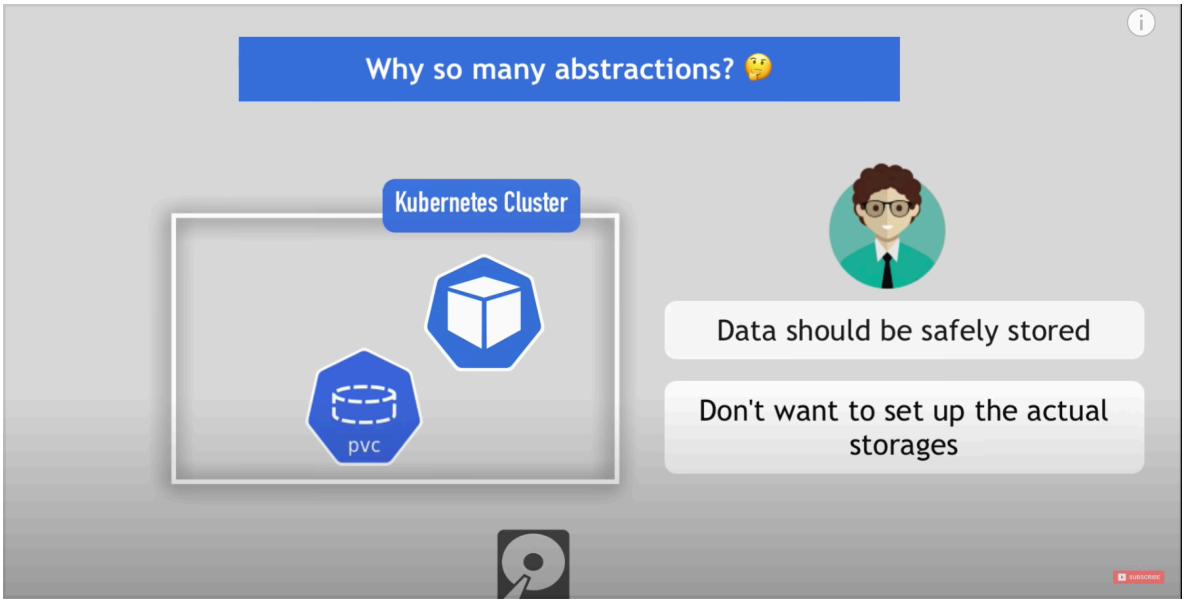


- Claims must be in the same namespace!

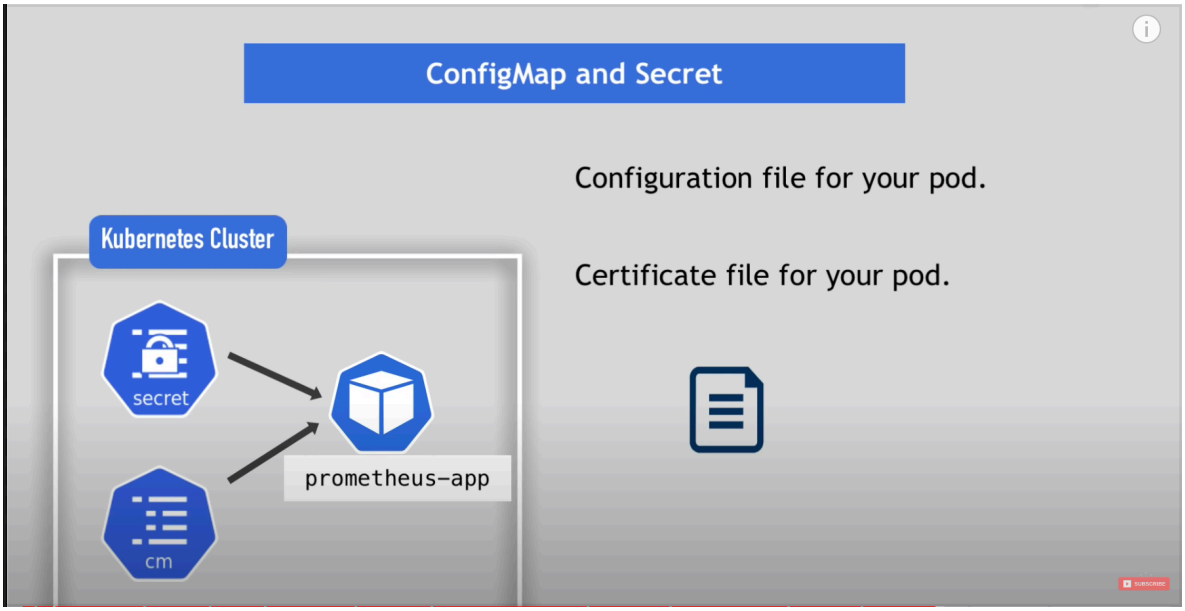


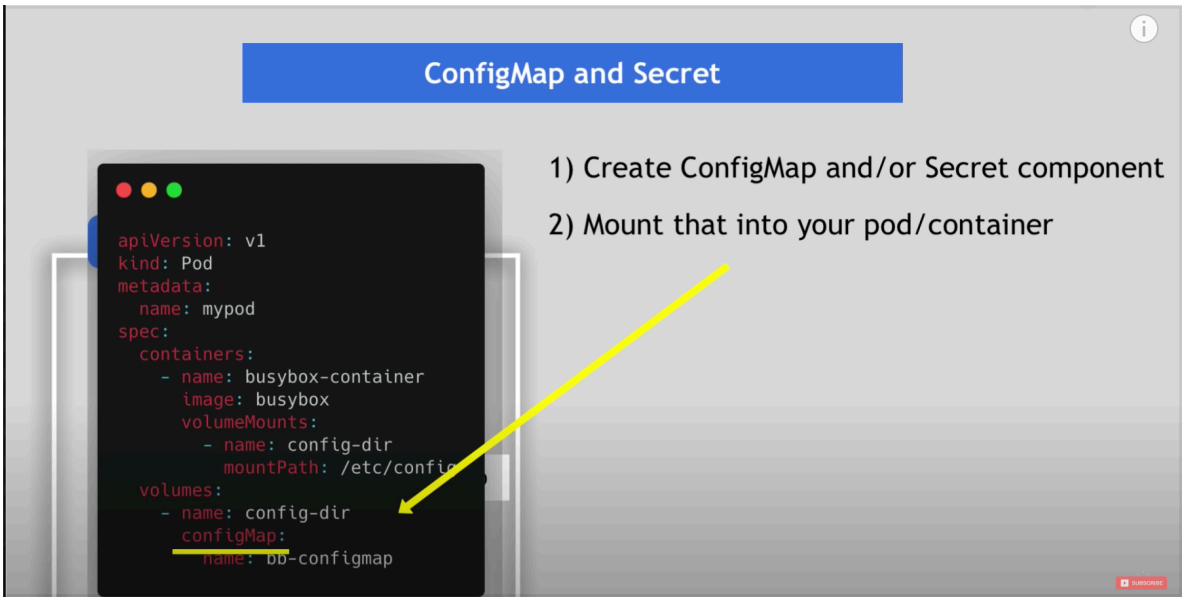


- And if you have multiple container in the Pod then you can decide which one to have and which one to not.
- You may wonder so many abstraction like admin will provision the storage and developers has to claim that etc.., it's actually has a benefit

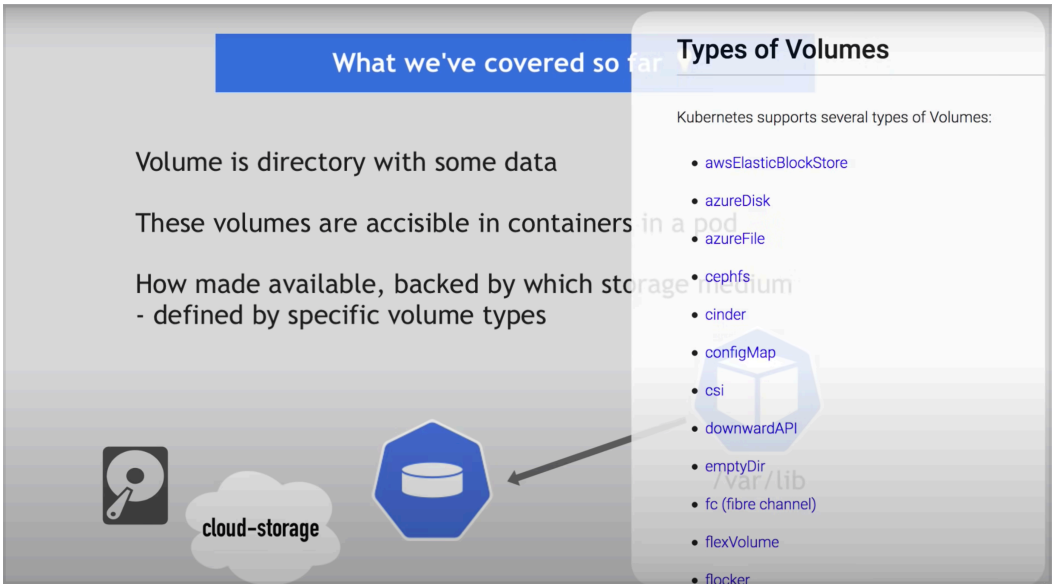


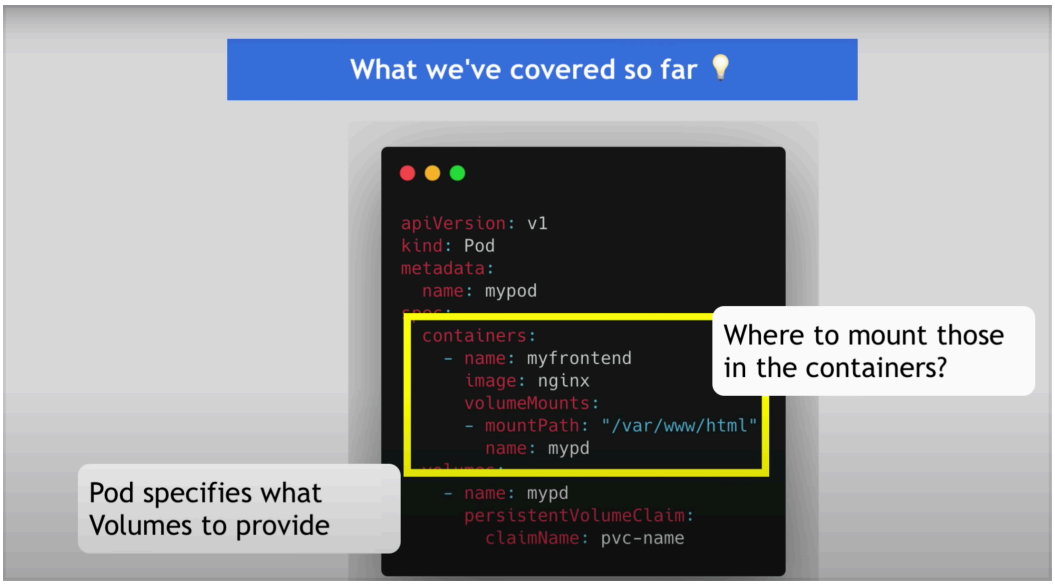
  - Easier for developers, they don't have to worry about the backend of it etc
-
- And also there are other 2 types of volumes that should be mentioned separately as they are bit different from the rest , they are ConfigMap and Secret
  - This Both are local volumes
  - Not created via PV and PVC
  - But rather by own component and managed by k8s itself

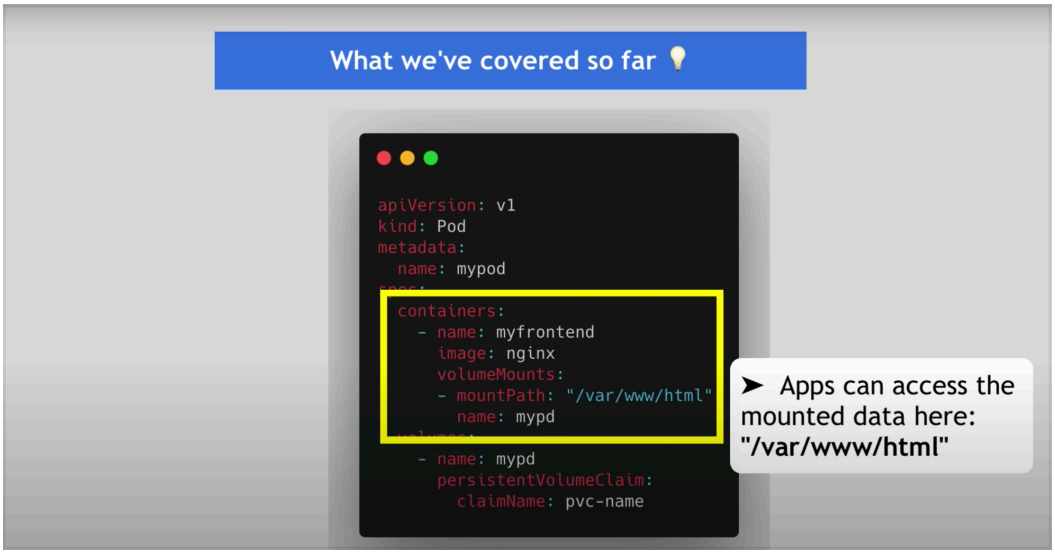


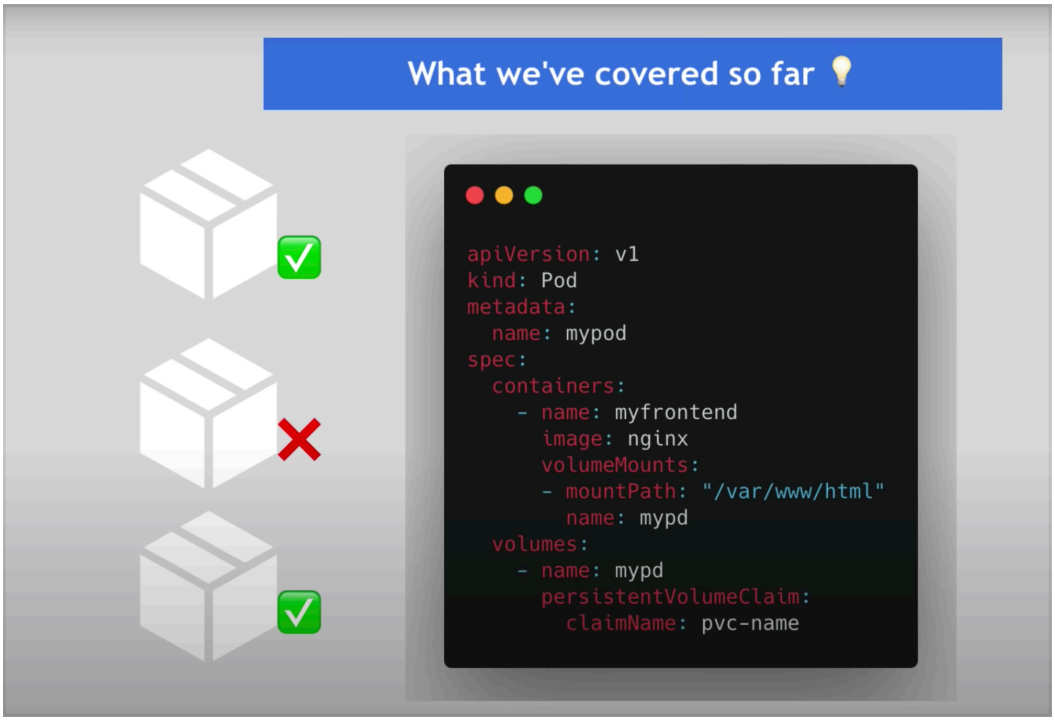


- What we have covered so far

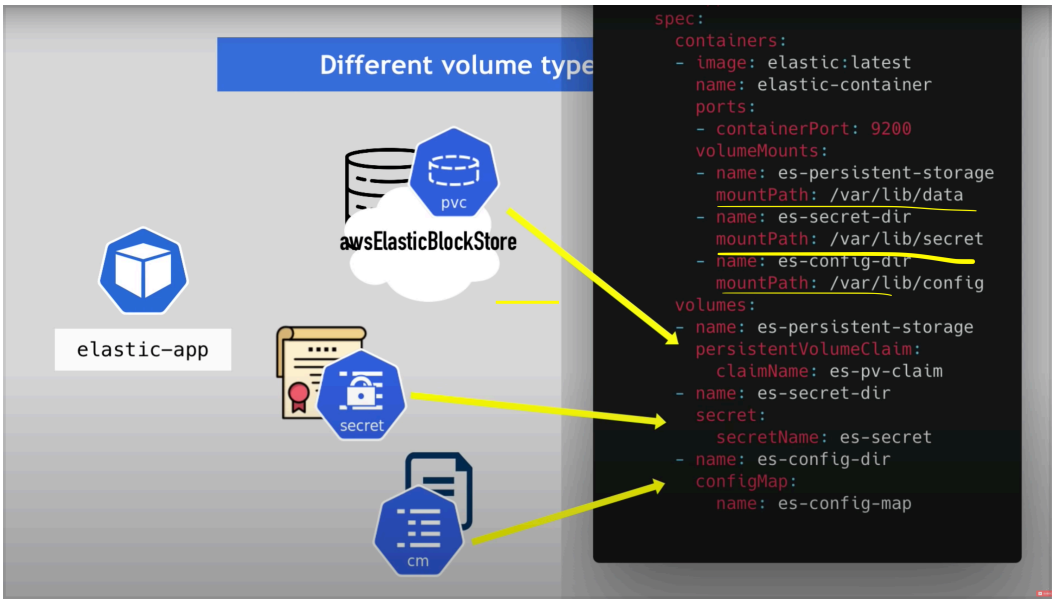




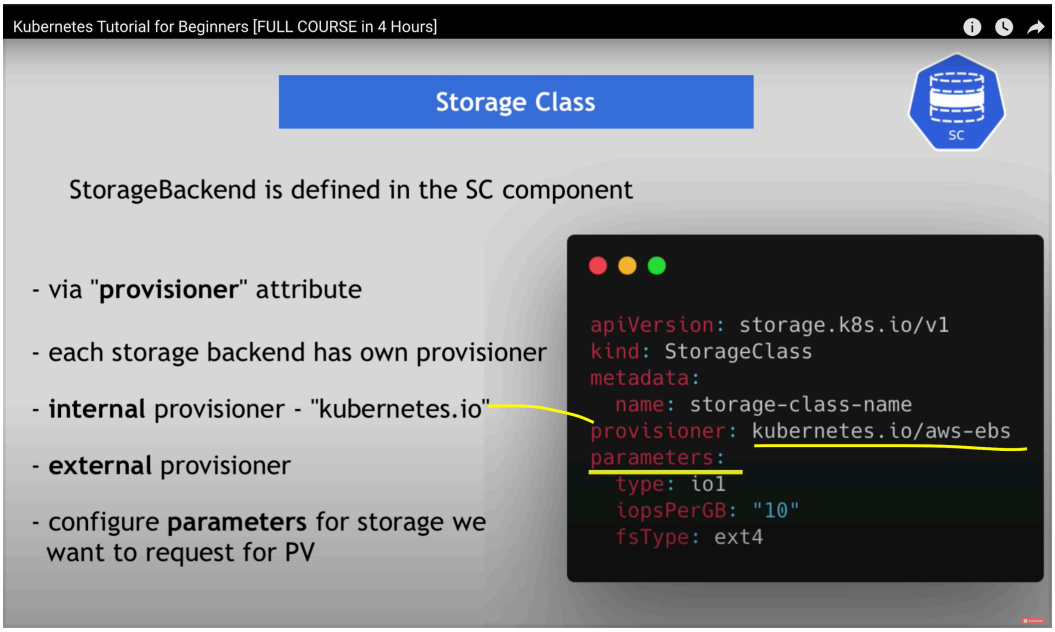




- Interesting note is
  - Pods can actually use a multiple volumes of a different types simultaneously - https://youtu.be/X48VuDVv0do?t=10422



- If we consider a cluster with 100s of applications where things get deployed daily and storage is needed for applications, so developers need to ask admins to create PV they need for apps before deploying them and admin may have to manually request cloud or storage provider and create hundreds of PV for all the apps that needs storage manually, so that very time consuming can gets messy very quickly
  - So to make this process more efficient, there is 3rd component of k8s persistence called "Storage class" which basically created or provision PVs dynamically whenever PVC claims it
  - AN this way creating our provisioning Volumes in cluster may be automated
    - Also gets created with YAML



Storage Class

Another abstraction level

- abstracts underlying storage provider
- parameters for that storage

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

Storage Class usage

Requested by PersistentVolumeClaim

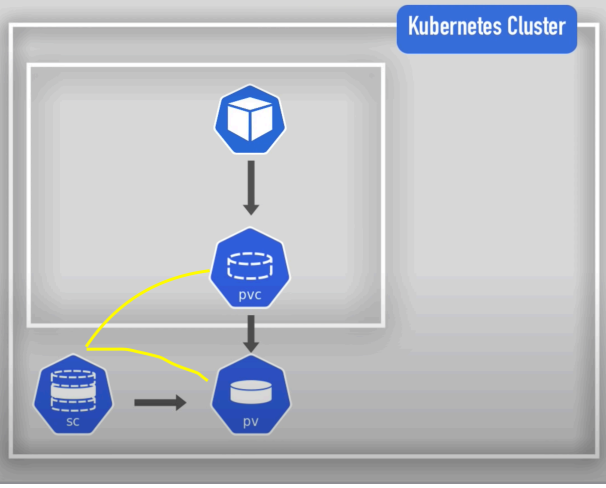
Storage Class Config

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

PVC Config

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: mypvc
spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 100Gi
    storageClassName: storage-class-name
```

Storage Class usage



1) Pod claims storage via PVC

2) PVC requests storage from SC

3) SC creates PV that meets the needs of the Claim