**Storage Solutions in K8s**

* Stateful applications store data locally. In Containers the data created locally will be lost once you delete it. So, to solve this in docker we have used volumes. Volumes have a lifecycle which has no relation to container lifecycle (refer docker containers, image layers, volumes)
* IN k8s we are running docker containers, k8s is an orchestration solution.
* Let’s see what are options for storage provisioning in k8s <https://kubernetes.io/docs/concepts/storage/>
* The most widely used storage types
  + Volumes
  + Persistent Volumes
    - Storage Classes
    - Persistent Volume Claims
* Volumes <https://kubernetes.io/docs/concepts/storage/volumes/>

**Volumes**

* Volumes can be mounted to containers and they have lifetime equivalent to Pods.
* The types of Volumes <https://kubernetes.io/docs/concepts/storage/volumes/#volume-types>
* The types
  + storage on cloud
    - ebs
    - azure disk
    - efs
    - azure file
    - gcs
  + empty dir
  + hostPath
* Lets create a manifest with mysql-pod with volume

---

apiVersion: v1

kind: Pod

metadata:

name: mysql-vol

labels:

app: mysql

layer: db

spec:

containers:

- name: mysql

image: mysql:8

ports:

- containerPort: 3306

volumeMounts:

- name: test-volume

mountPath: /var/lib/mysql

volumes:

- name: test-volume

emptyDir: sizeLimit: 100Mi

**Persistent Volumes**

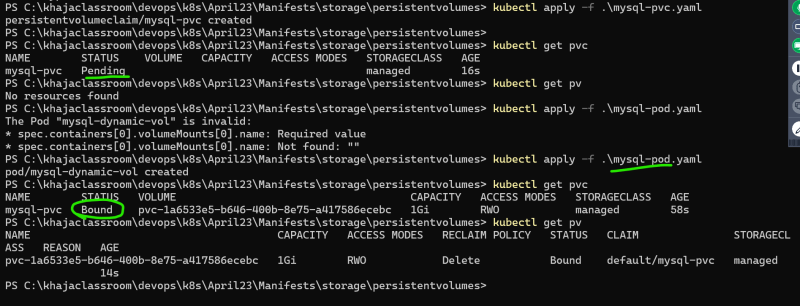
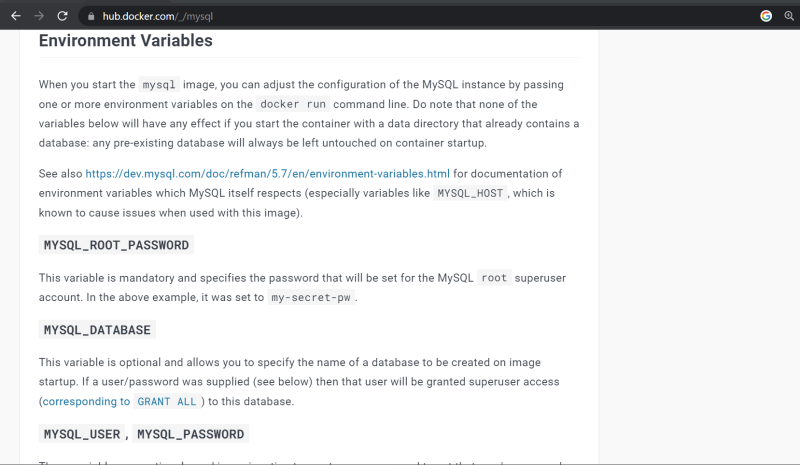
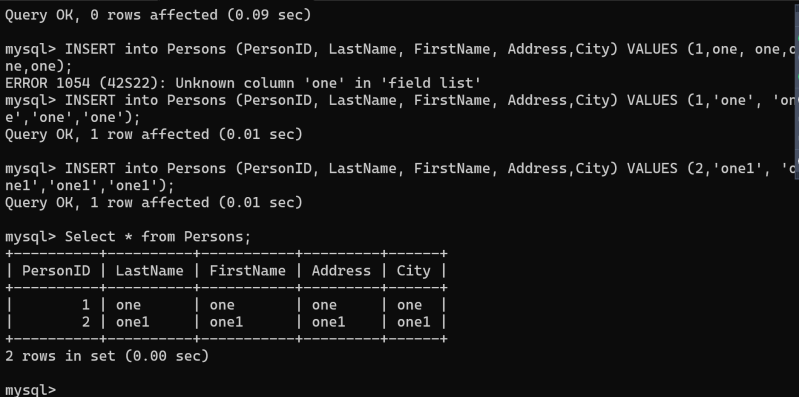
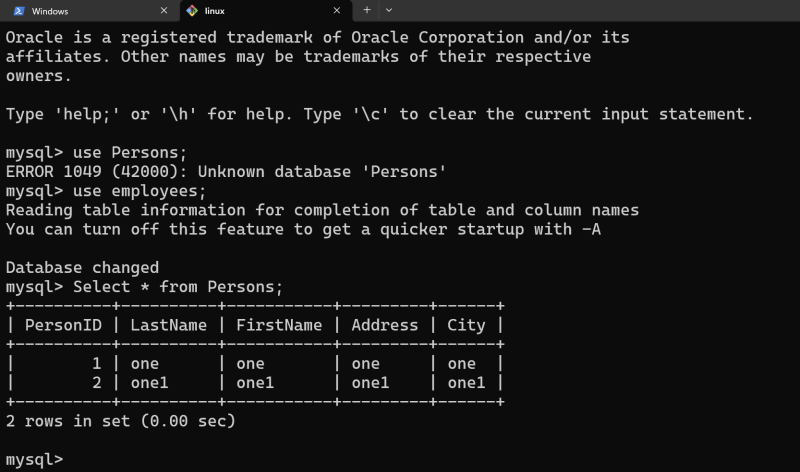
* These volumes will have a lifetime different than Pod i.e. they exist even after pod is dead.
* Types of PVC <https://kubernetes.io/docs/concepts/storage/persistent-volumes/#expanding-persistent-volumes-claims>
* Persistent Volumes in Azure <https://learn.microsoft.com/en-us/azure/aks/concepts-storage>
* Storage classes in Azure <https://learn.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes>
* Storage classes in AWS <https://docs.aws.amazon.com/eks/latest/userguide/storage-classes.html>

In story> To save data in container we have solution called volume, but volume would be available until pod is running. To store data, we have persistent volume. In persistent volume we can do provisioning in 2 ways

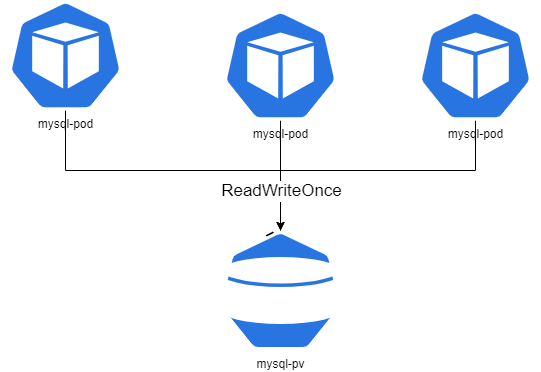
1. Static provisioning> Here we precure the storage
2. Dynamic provisioning>Here we get on demand storage

For doing dynamic provisioning we need storage class.

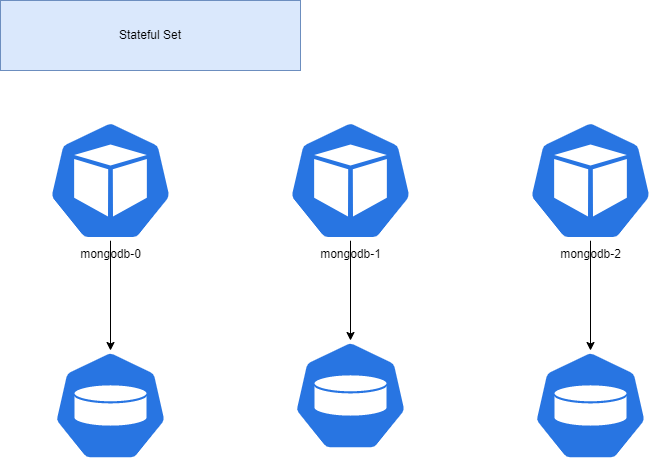
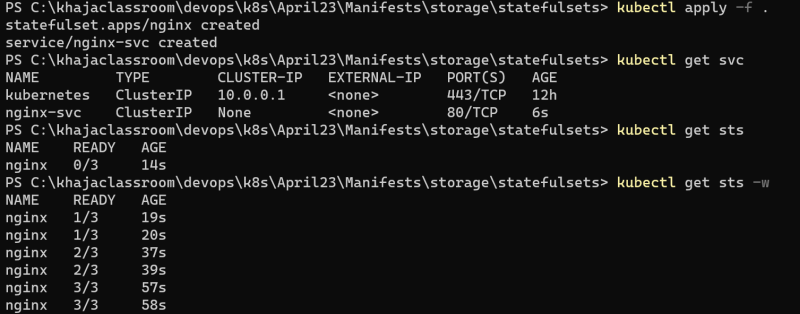
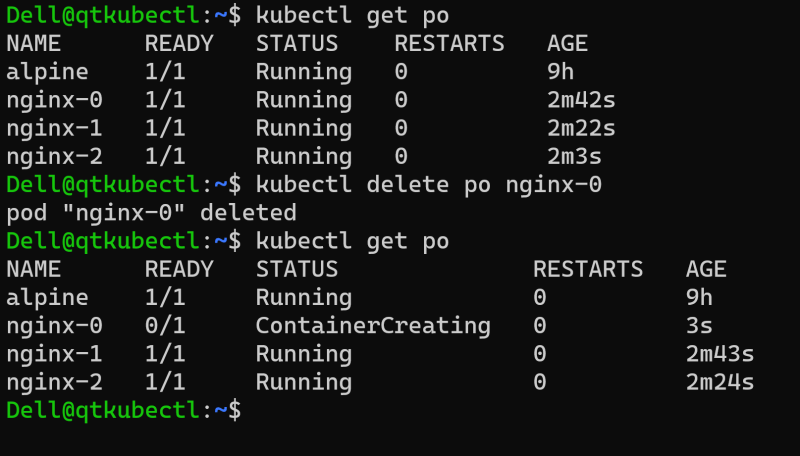
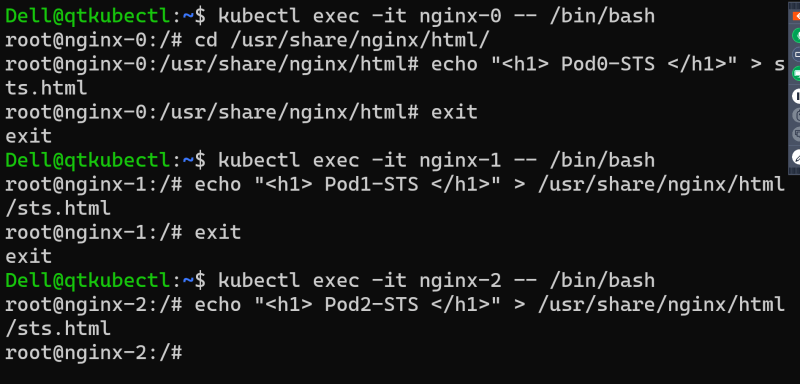
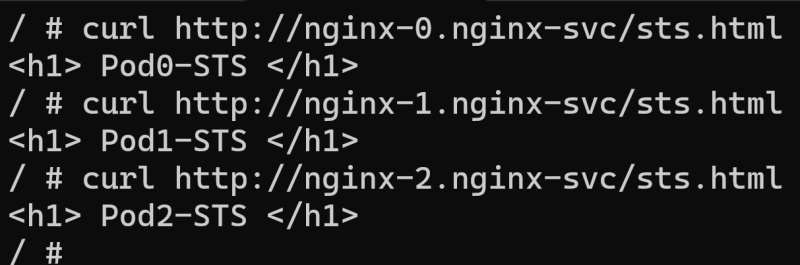
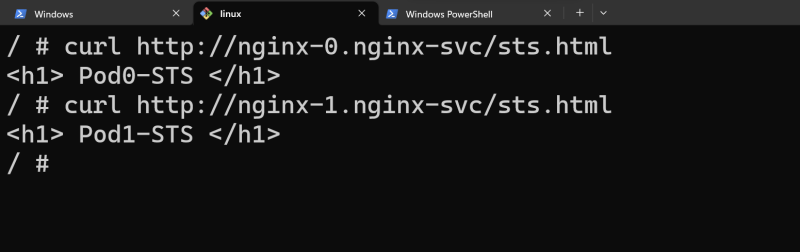
**Let’s create a MySQL-pod with dynamic volume provisioning**

* Get storage classes  
  
* Lets create a Persistent Volume Claim and a mysql pod claiming pvc  
  
* <https://github.com/asquarezone/KubernetesZone/commit/c48ea958b6d8fc4a9dab0637a92e105a9dc61b56> for the initial manifests written by us
* Now let’s modify the spec to add mysql related environment variables  
  
* <https://github.com/asquarezone/KubernetesZone/commit/d6afa41b79ff3e552549018459f380eb7f8135f7> for the mysql env values
* Insert some data into mysql  
  
* now delete pod and recreate and check for the data created  
  
* For static provisioning of volume <https://learn.microsoft.com/en-us/azure/aks/azure-csi-disk-storage-provision#statically-provision-a-volume>

If you want to scale Pods we can use

* + Replica Sets
  + Deployments  
    

**StatefulSets**

* Statefulset is like deployment with replicas. But each pod gets its own volume.
* Stateful Set is for stateful applications  
  
* When we create replicas in Stateful Set we get predictable names
* <https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/> for official docs
* We can access individual pod, by creating headless service and by using ...svc.cluster.local
* Lets experiment with stateful sets and create nginx pods
* <https://github.com/asquarezone/KubernetesZone/commit/255c9c7303ac2d66f22c79367ade1f6bcc7af877> for the manifests  
  
* Naming is predictable in stateful sets  
  
* Now create a different html page for every pod  
  
* Send a curl request from any alpine pod <pod-name>.<headless.svc-name>.<namespace>.svc.cluster.local  
    
  

Shards is the concept to share data.