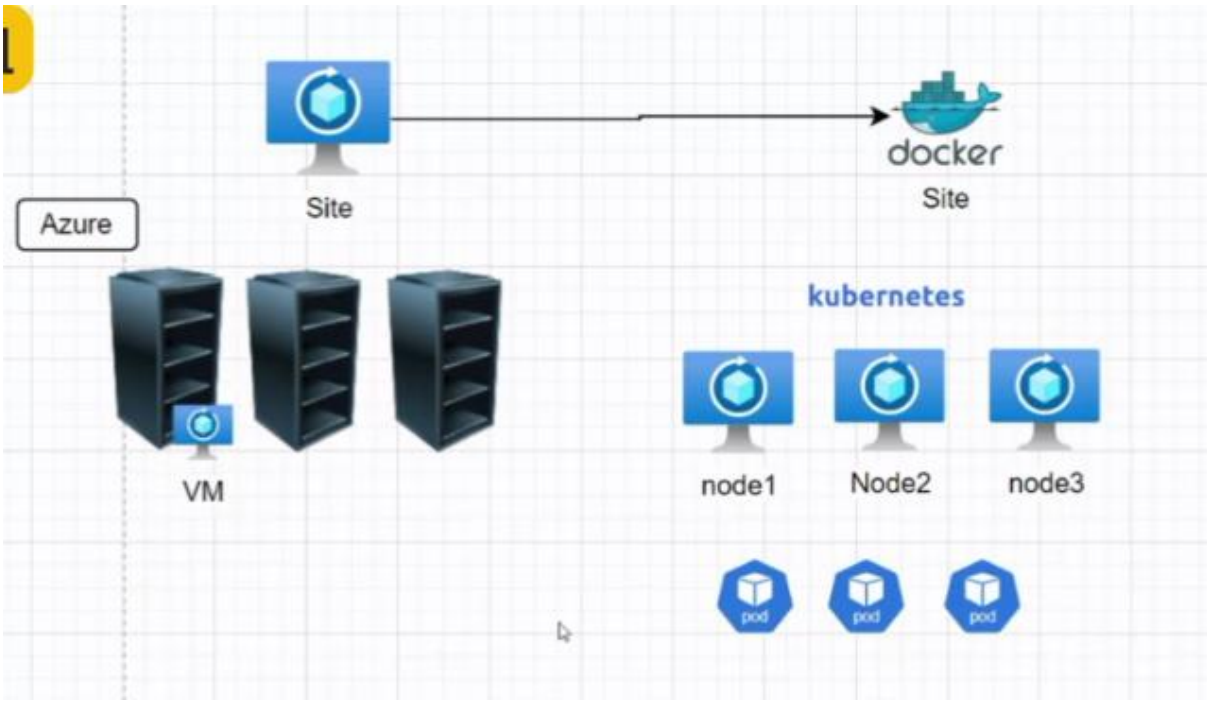
27 oct

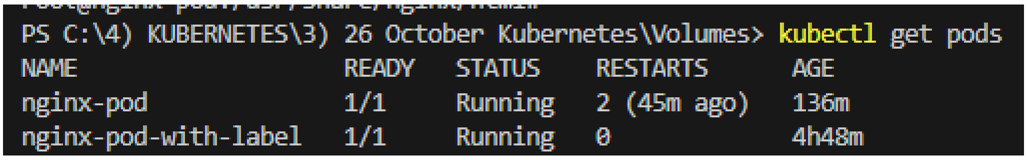
AGENDA – PV AND PVC



1) We are continuing from last class only

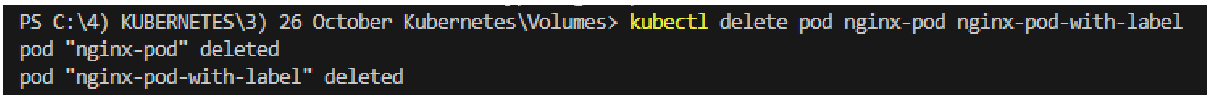
2) Now delete all pods in cluster

kubectl get pods



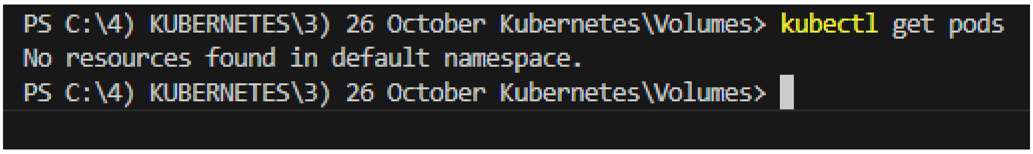
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME                   READY   STATUS    RESTARTS      AGE
nginx-pod              1/1     Running   2 (45m ago)   136m
nginx-pod-with-label   1/1     Running   0             4h48m
```

kubectl delete pod nginx-pod nginx-pod-with-label



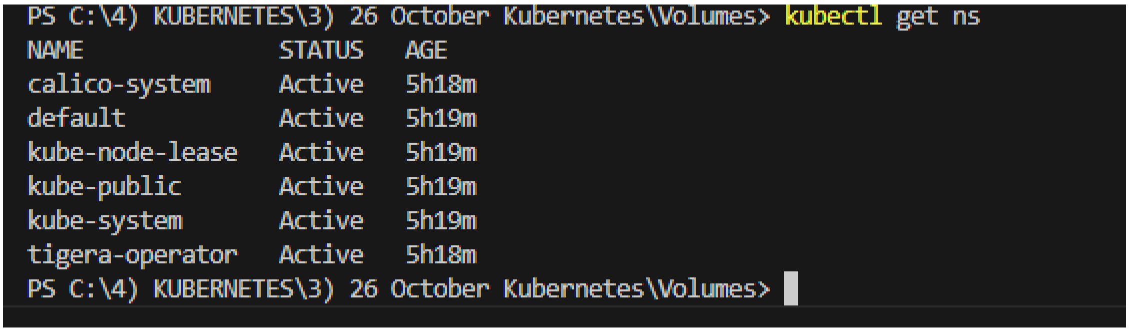
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete pod nginx-pod nginx-pod-with-label
pod "nginx-pod" deleted
pod "nginx-pod-with-label" deleted
```

kubectl get pods



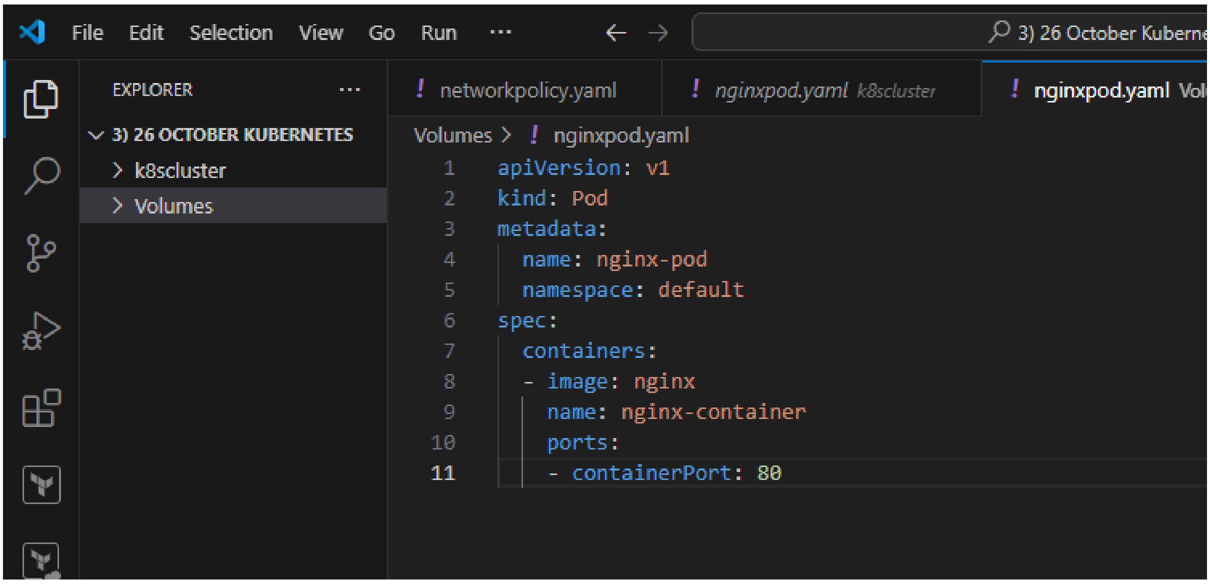
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
No resources found in default namespace.
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

3) **kubectl get ns** = Other namespaces ang things are running



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get ns
NAME               STATUS   AGE
calico-system      Active   5h18m
default            Active   5h19m
kube-node-lease    Active   5h19m
kube-public        Active   5h19m
kube-system        Active   5h19m
tigera-operator    Active   5h18m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

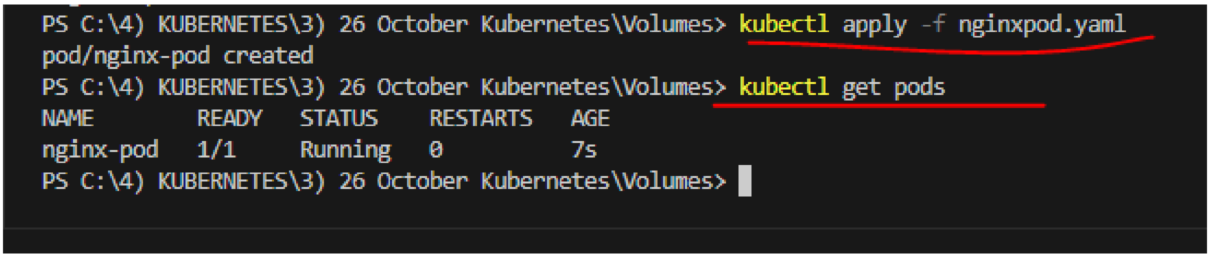
4) Create nginx pod using above file



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
spec:
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
```

5) **kubectl apply -f nginxpod.yaml =** create nginx pod

kubectl get pods

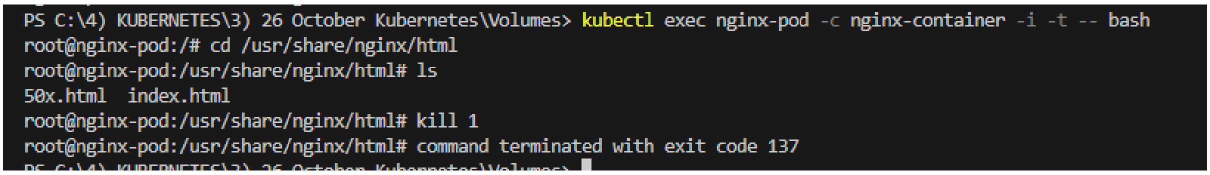


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f nginxpod.yaml
pod/nginx-pod created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          7s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

6) **kubectl exec nginx-pod -c nginx-container -i -t – bash**

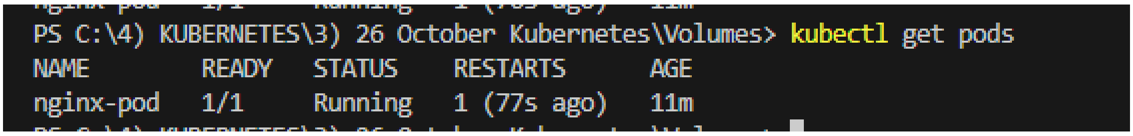
cd /usr/share/nginx/html

kill 1 = container maar diya



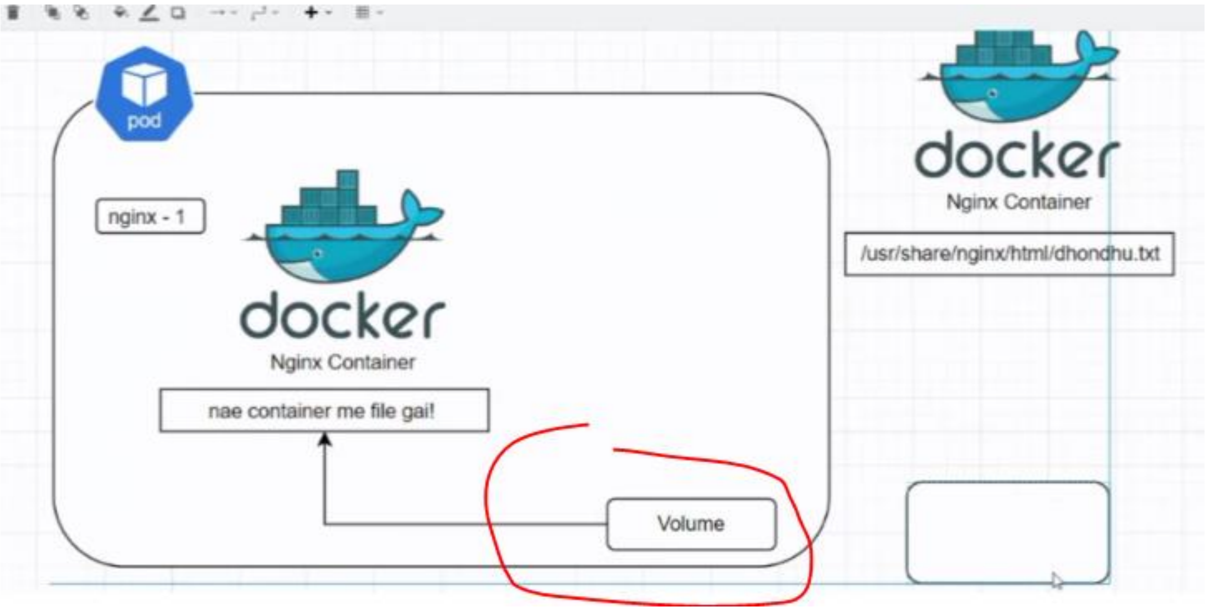
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c nginx-container -i -t -- bash
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
50x.html  index.html
root@nginx-pod:/usr/share/nginx/html# kill 1
root@nginx-pod:/usr/share/nginx/html# command terminated with exit code 137
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

7) **kubectl get pods**

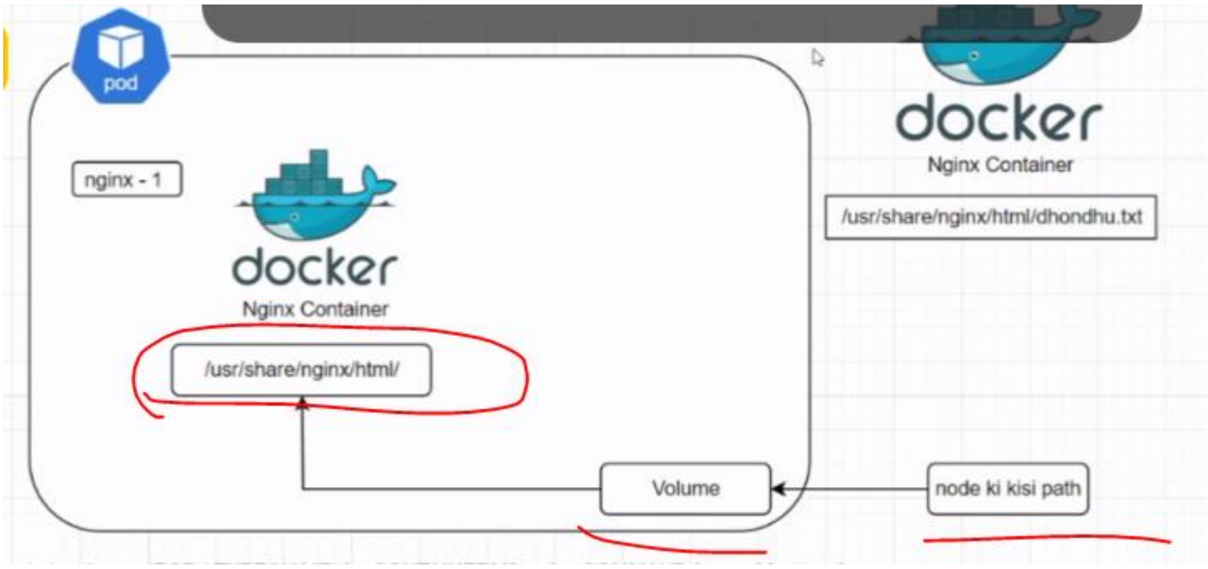


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS      AGE
nginx-pod   1/1     Running   1 (77s ago)   11m
```

8) Now as container dead then our data gets lost so for that we will mount volume in our container so that if old container dies and new container is created by pod then that volume or data is mounted on new container as well and our data is not lost. Here volume is just like a folder



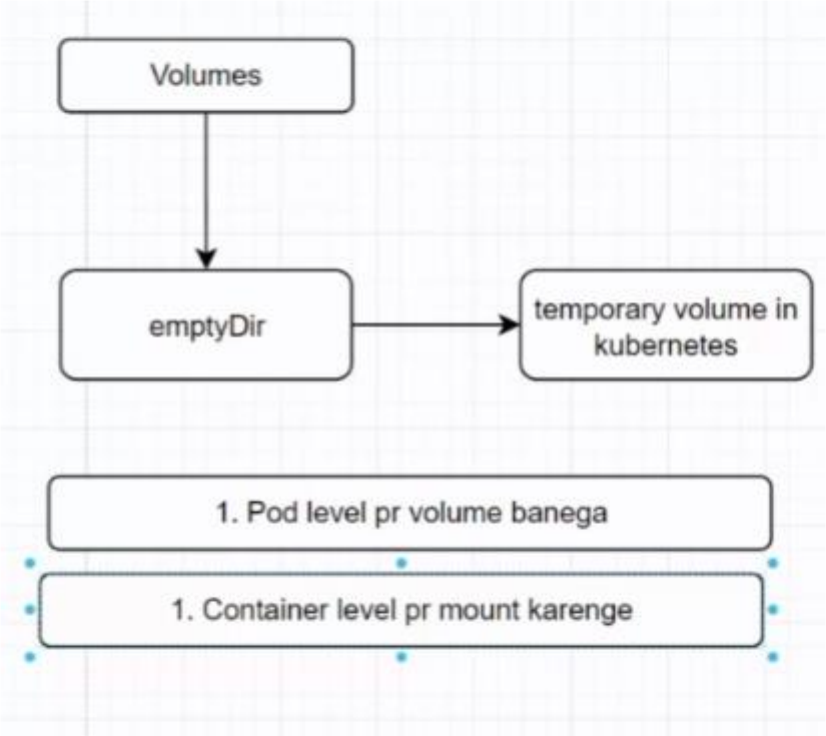
9) Our pod actually runs on node. So node ke kisi path ko container pr mount krde so our volume or data will be there or will reflect in next new container



10) **Kubectl explain pod --recursive > pod.txt** ==== creates file in left side with name pod.txt having full doc

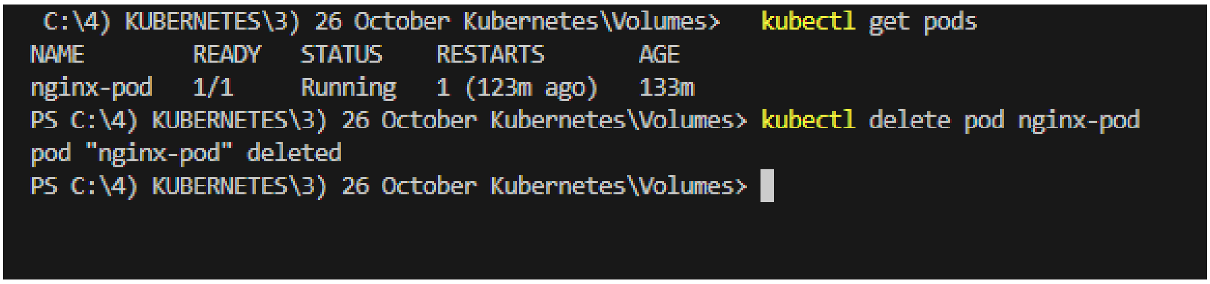
NOTE : Volume banta hai pod level pr but mount hota hai container level pr

11)


Volumes
emptyDir
temporary volume in kubernetes
1. Pod level pr volume banega
1. Container level pr mount karenge

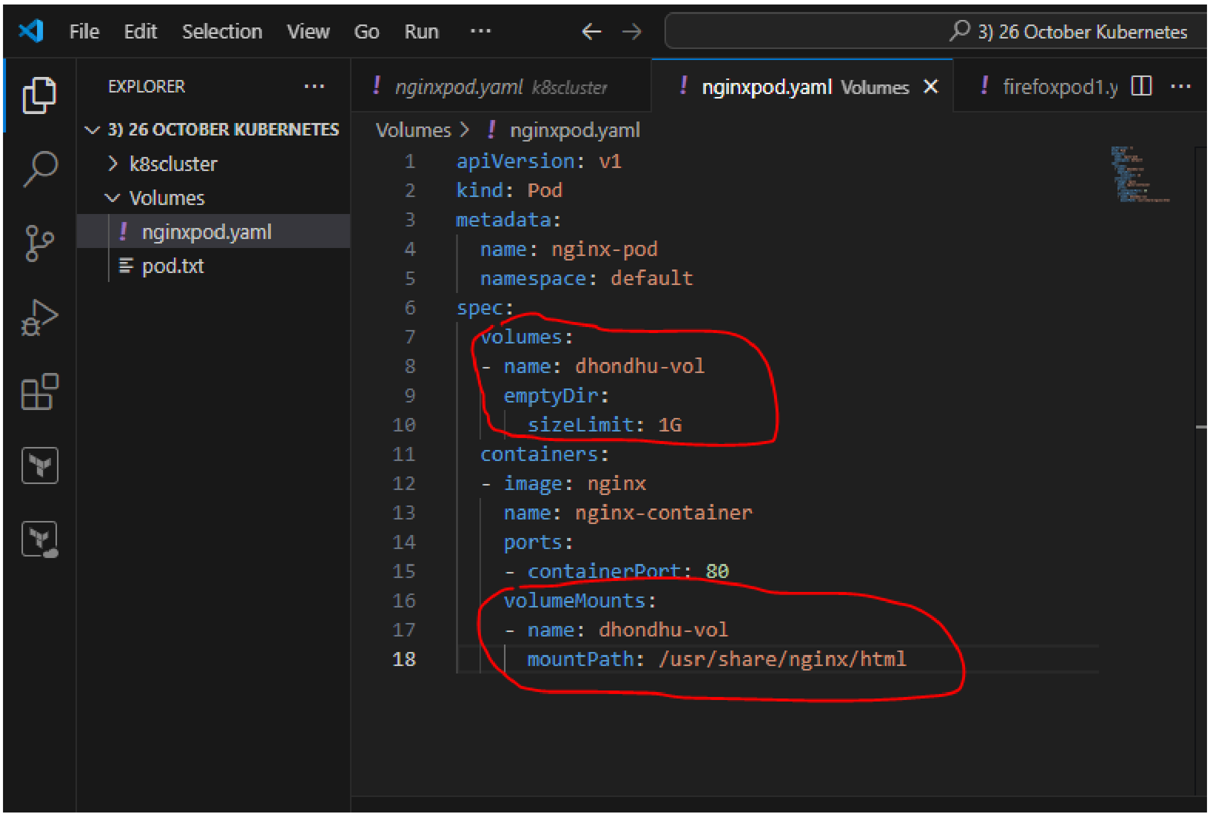
12) **kubectl get pods**

kubectl delete pod nginx-pod



```
 C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>  kubectl get pods
NAME        READY   STATUS    RESTARTS        AGE
nginx-pod   1/1     Running   1 (123m ago)    133m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete pod nginx-pod
pod "nginx-pod" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> 
```

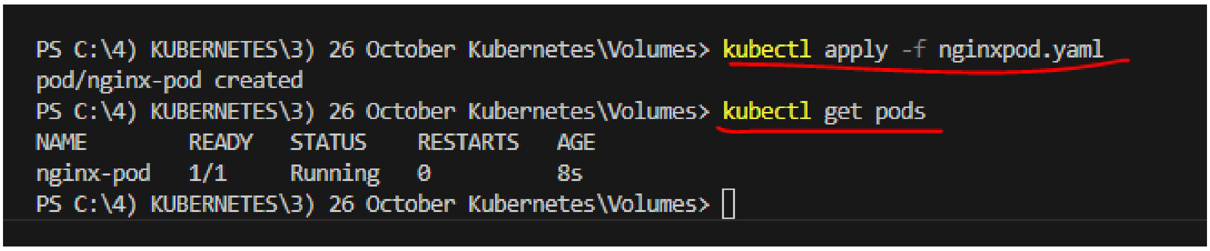
13) Now as per file created k8s will bring 1GB data to create volume



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
spec:
  volumes:
  - name: dhondhu-vol
    emptyDir:
      sizeLimit: 1G
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /usr/share/nginx/html
```

14) **kubectl apply -f nginxpod.yaml**

kubectl get pods

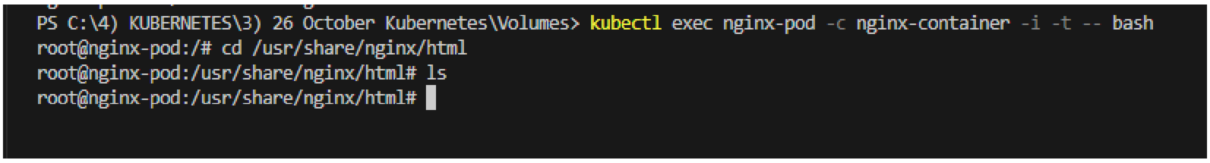


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f nginxpod.yaml
pod/nginx-pod created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          8s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

15) **kubectl exec nginx-pod -c nginx-container -i -t -- bash**

cd /usr/share/nginx/html

ls



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c nginx-container -i -t -- bash
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
root@nginx-pod:/usr/share/nginx/html#
```

So we can see after ls its not showing any file like index.html and all because it is empty volume created

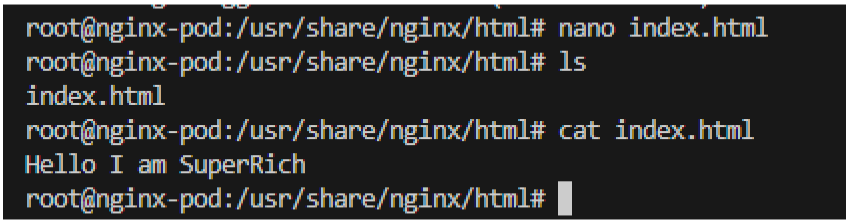
16) Now installing nano

apt update

apt install nano

17) Now making index.html file

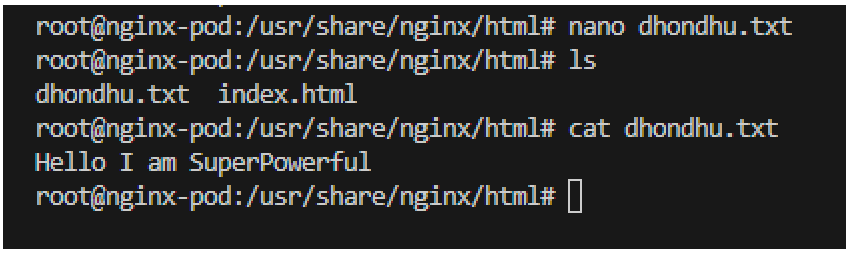
Nano index.html, Ctrl +s, Ctrl+x



```
root@nginx-pod:/usr/share/nginx/html# nano index.html
root@nginx-pod:/usr/share/nginx/html# ls
index.html
root@nginx-pod:/usr/share/nginx/html# cat index.html
Hello I am SuperRich
root@nginx-pod:/usr/share/nginx/html#
```

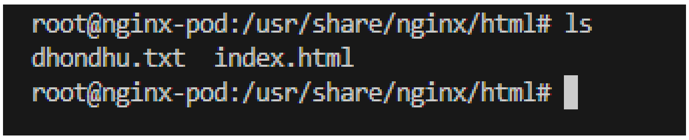
18) Now making dhondhu.txt file

nano dhondhu.txt, ctrl+s, ctrl+x



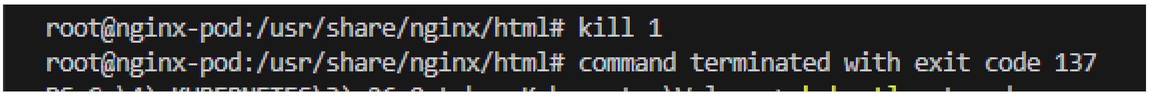
```
root@nginx-pod:/usr/share/nginx/html# nano dhondhu.txt
root@nginx-pod:/usr/share/nginx/html# ls
dhondhu.txt  index.html
root@nginx-pod:/usr/share/nginx/html# cat dhondhu.txt
Hello I am SuperPowerful
root@nginx-pod:/usr/share/nginx/html#
```

19) ls



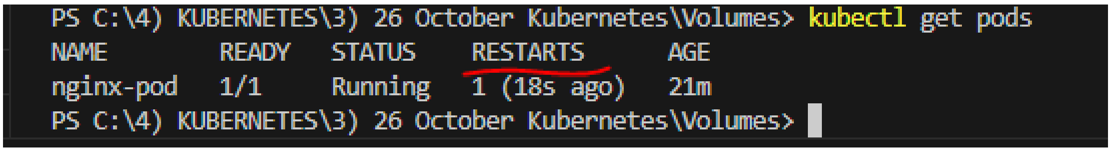
```
root@nginx-pod:/usr/share/nginx/html# ls
dhondhu.txt  index.html
root@nginx-pod:/usr/share/nginx/html#
```

20) **kill 1** = kill container



```
root@nginx-pod:/usr/share/nginx/html# kill 1
root@nginx-pod:/usr/share/nginx/html# command terminated with exit code 137
```

21) **kubectl get pods**

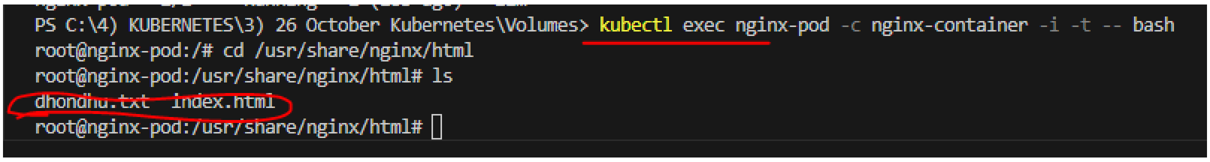


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS      AGE
nginx-pod   1/1     Running   1 (18s ago)   21m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

22) **kubectl exec nginx-pod -c nginx-container -i -t -- bash**

cd /usr/share/nginx/html

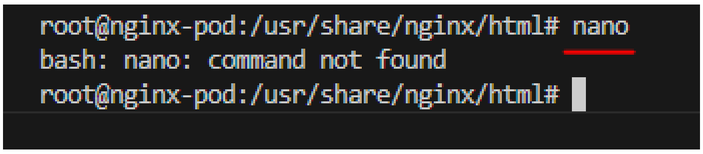
ls



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c nginx-container -i -t -- bash
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
dhondhu.txt  index.html
root@nginx-pod:/usr/share/nginx/html#
```

So this time in new created container also both files are being shown so data is not lost this time as before

23) To check it's a new container check for nano



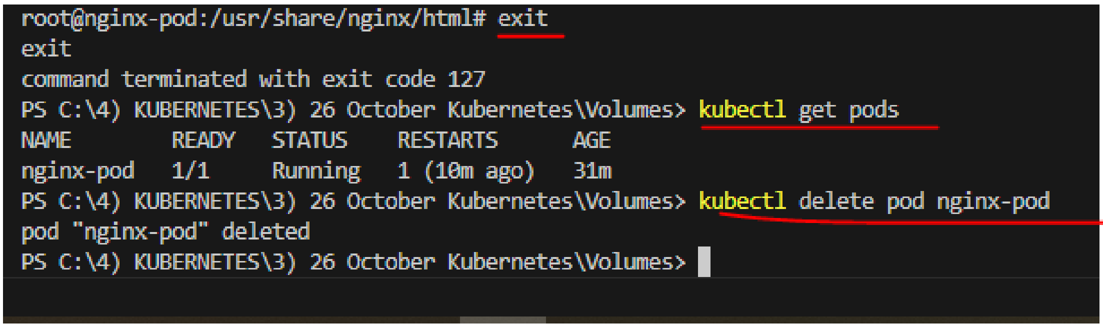
```
root@nginx-pod:/usr/share/nginx/html# nano
bash: nano: command not found
root@nginx-pod:/usr/share/nginx/html#
```

24) Now suppose if pod got died

exit

kubectl get pods

kubectl delete pod nginx-pod



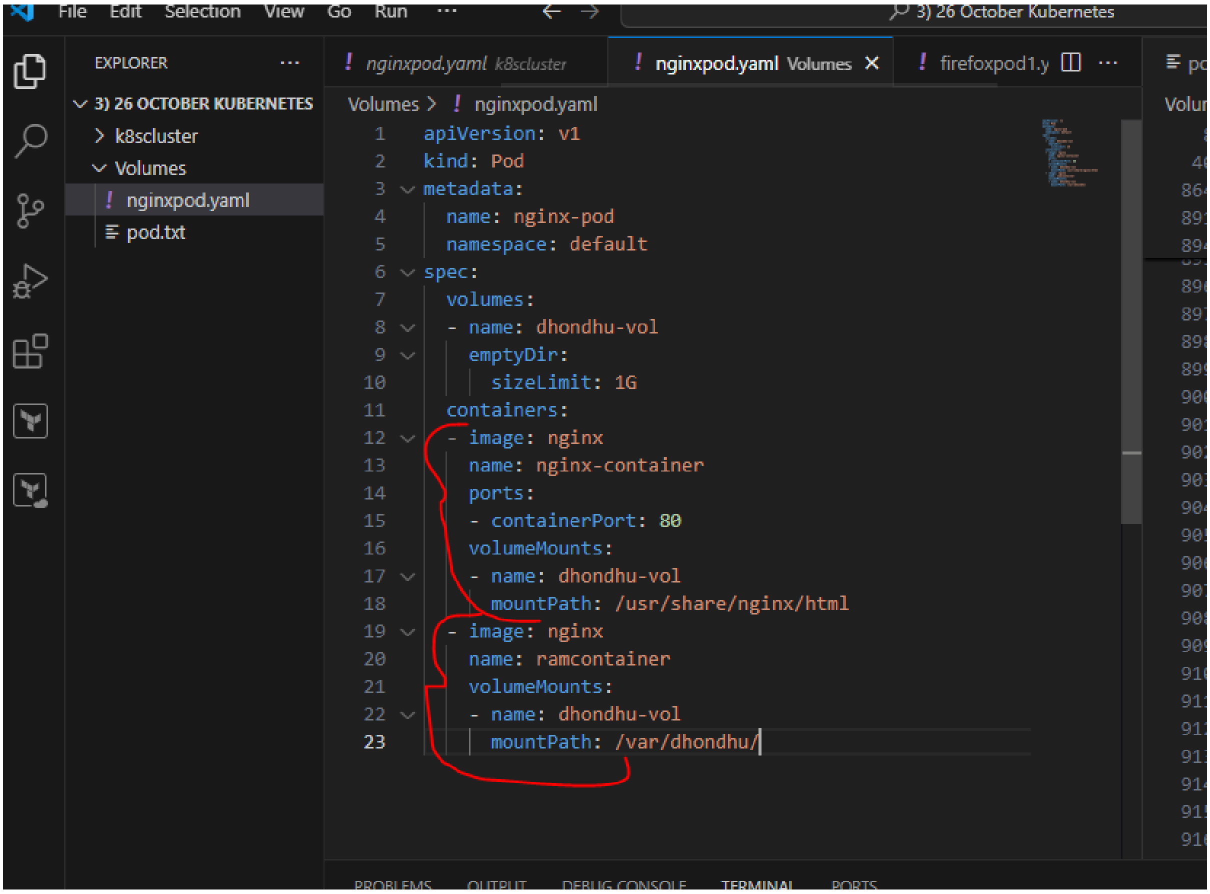
```
root@nginx-pod:/usr/share/nginx/html# exit
exit
command terminated with exit code 127
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS      AGE
nginx-pod   1/1     Running   1 (10m ago)   31m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete pod nginx-pod
pod "nginx-pod" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

25) So as pod died so our emptyDir path is also dead, So now how we will solve this pain. Basically this emptyDir is used for testing purpose not for production.

++

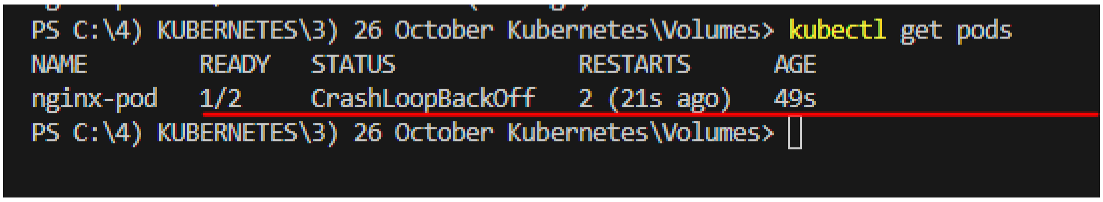
AGENDA – CREATING 2 CONTAINERS IN POD

1)



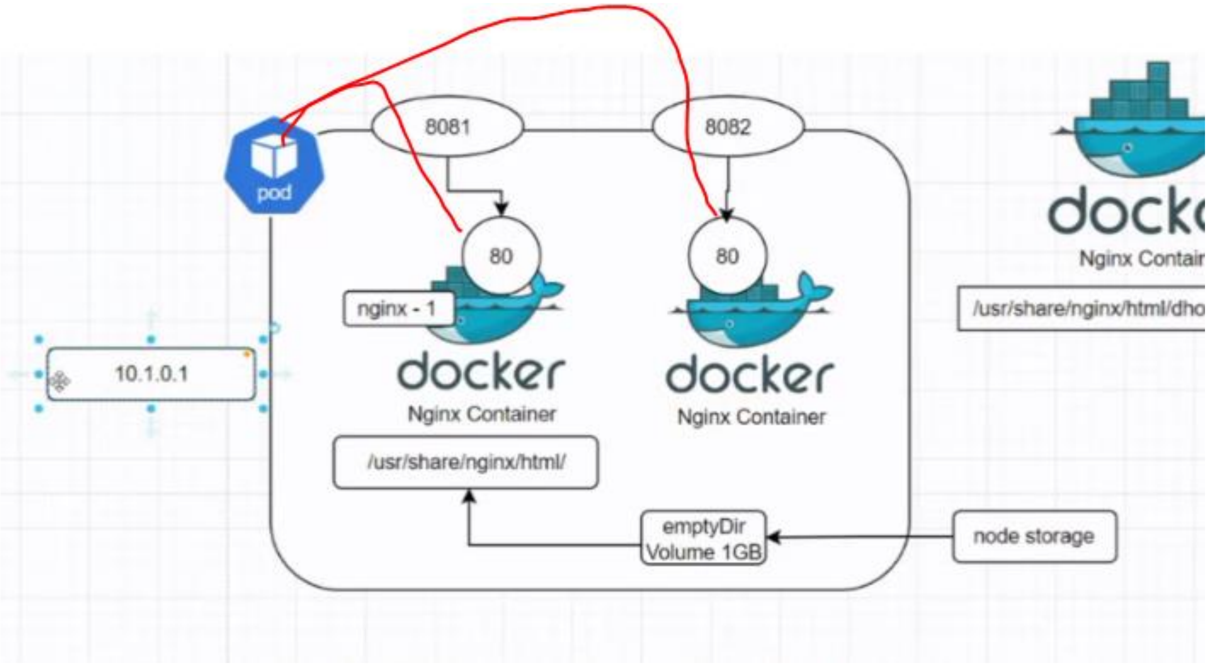
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
spec:
  volumes:
  - name: dhondhu-vol
    emptyDir:
      sizeLimit: 1G
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /usr/share/nginx/html
  - image: nginx
    name: ramcontainer
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /var/dhondhu/
```

2) Now 1 container is failing because by default ports are mapped on port 80 for nginx

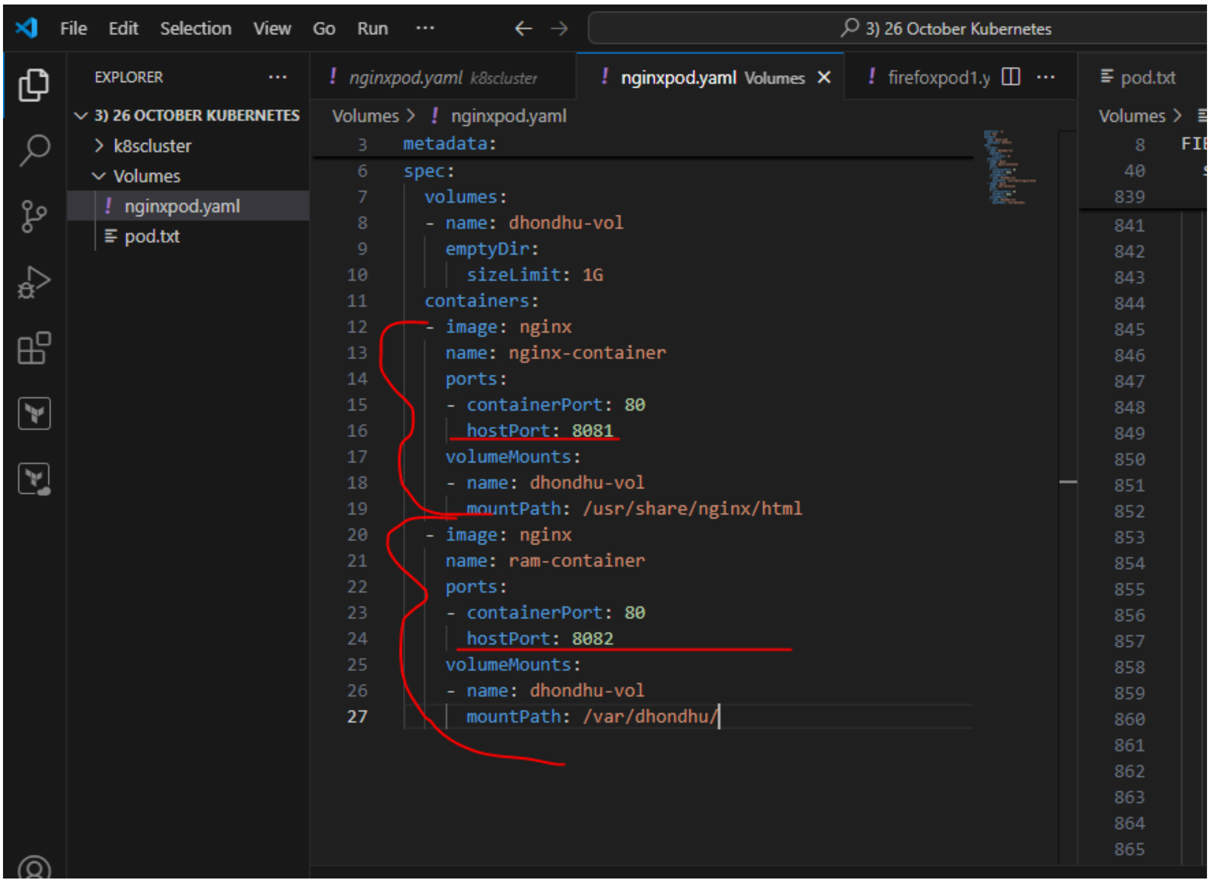


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS             RESTARTS      AGE
nginx-pod   1/2     CrashLoopBackOff   2 (21s ago)   49s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> 
```

3) So now we will change host ports in which pod is actually host.

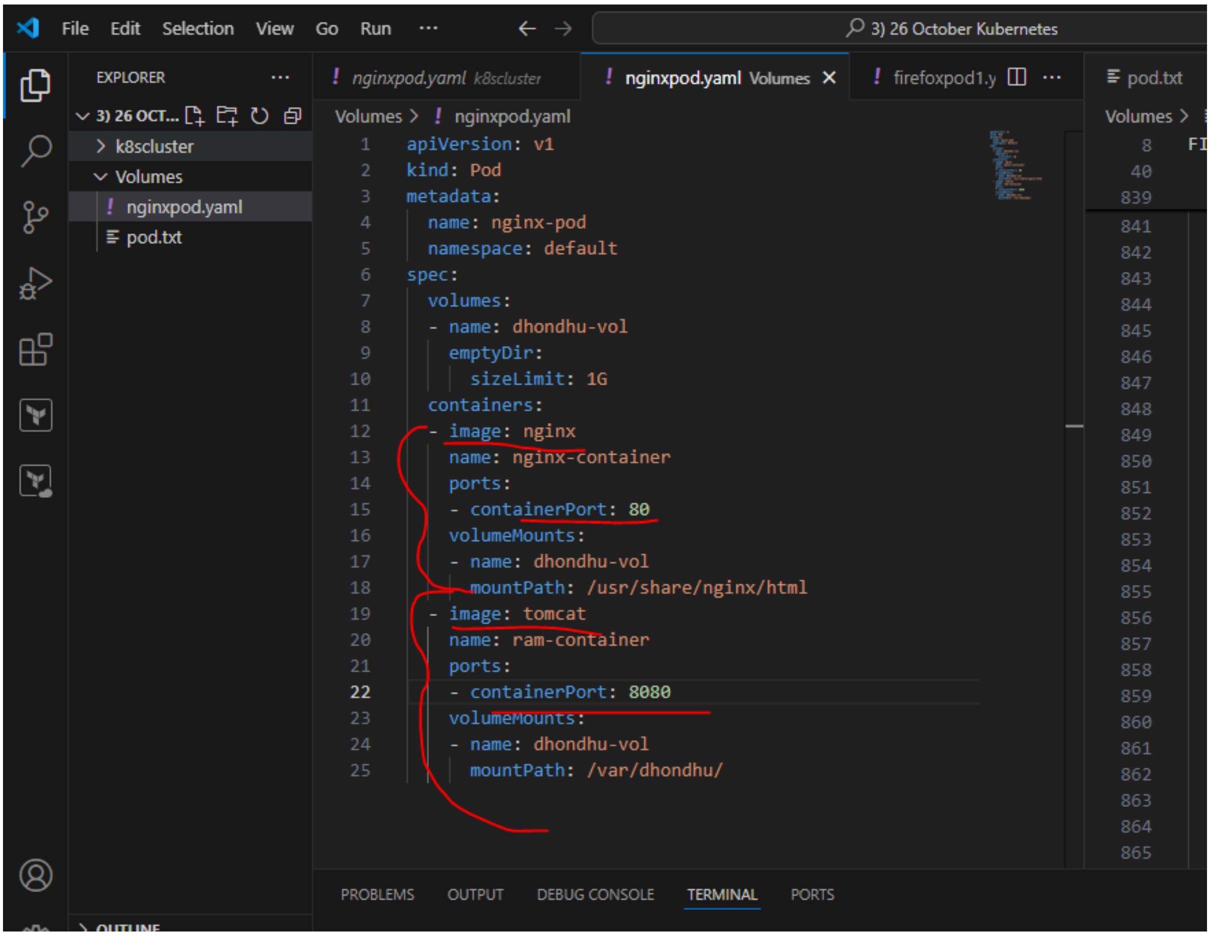


4) So mention **host ports** for both containers in yaml



```yaml
metadata:
spec:
  volumes:
  - name: dhondhu-vol
    emptyDir:
      sizeLimit: 1G
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
      hostPort: 8081
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /usr/share/nginx/html
  - image: nginx
    name: ram-container
    ports:
    - containerPort: 80
      hostPort: 8082
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /var/dhondhu/
```

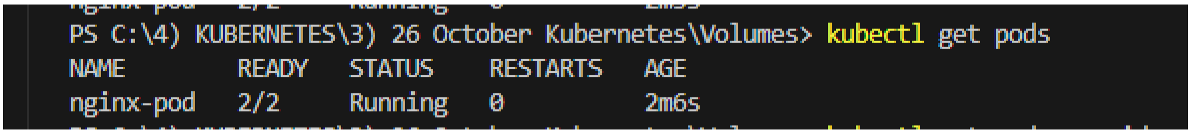
5) So our above logic failed so 2 containers on same port cannot run in a pod. So let us use tomcat image in yaml



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
spec:
  volumes:
  - name: dhondhu-vol
    emptyDir:
      sizeLimit: 1G
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /usr/share/nginx/html
  - image: tomcat
    name: ram-container
    ports:
    - containerPort: 8080
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /var/dhondhu/
```

6) kubectl apply -f nginxpod.yaml

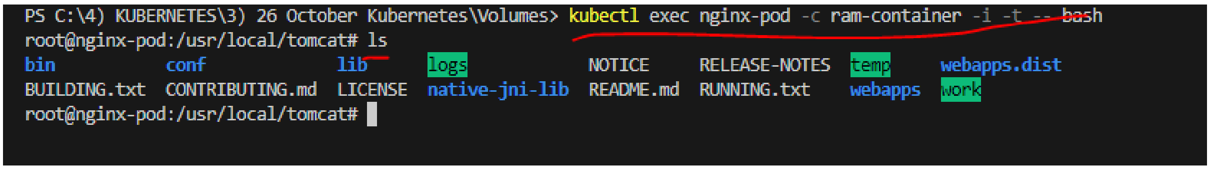
kubectl get pods



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   2/2     Running   0          2m6s
```

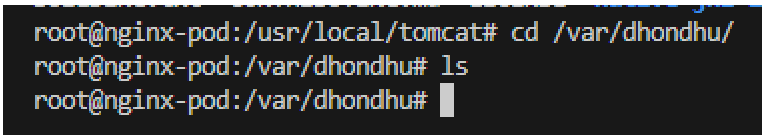
7) **kubectl exec nginx-pod -c ram-container -i -t – bash**

ls



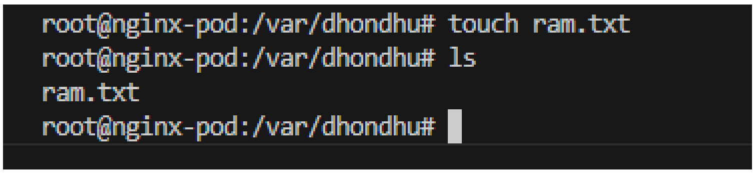
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c ram-container -i -t -- bash
root@nginx-pod:/usr/local/tomcat# ls
bin           conf             lib      logs            NOTICE     RELEASE-NOTES  temp     webapps.dist
BUILDING.txt  CONTRIBUTING.md  LICENSE  native-jni-lib  README.md  RUNNING.txt    webapps  work
root@nginx-pod:/usr/local/tomcat#
```

8) **cd /var/dhondhu/**



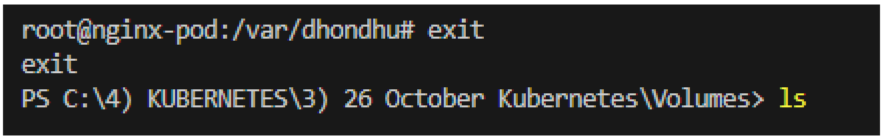
```
root@nginx-pod:/usr/local/tomcat# cd /var/dhondhu/
root@nginx-pod:/var/dhondhu# ls
root@nginx-pod:/var/dhondhu#
```

9) **touch ram.txt**



```
root@nginx-pod:/var/dhondhu# touch ram.txt
root@nginx-pod:/var/dhondhu# ls
ram.txt
root@nginx-pod:/var/dhondhu# 
```

10) **exit**

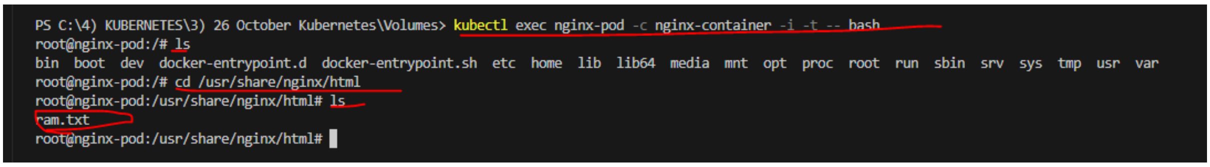


```
root@nginx-pod:/var/dhondhu# exit
exit
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> ls
```

11) **kubectl exec nginx-pod -c nginx-container -i -t – bash**

ls

cd /usr/share/nginx/html



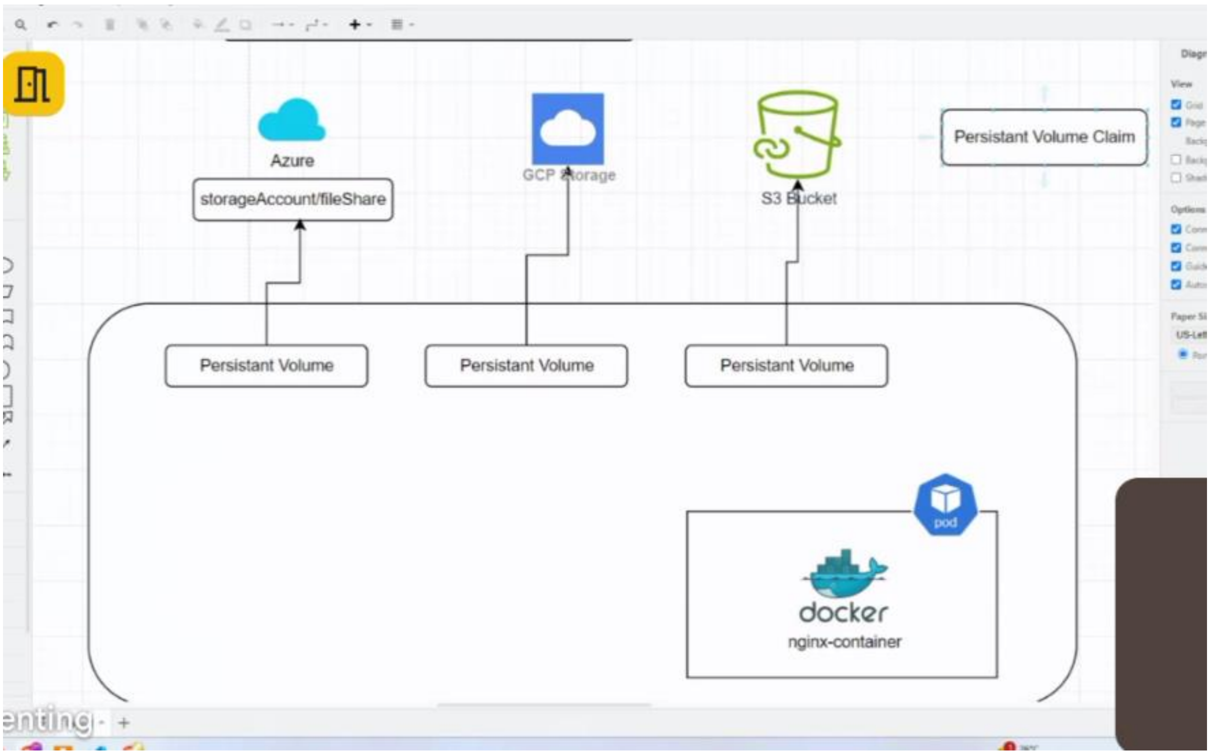
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c nginx-container -i -t -- bash
root@nginx-pod:/# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
ram.txt
root@nginx-pod:/usr/share/nginx/html# 
```

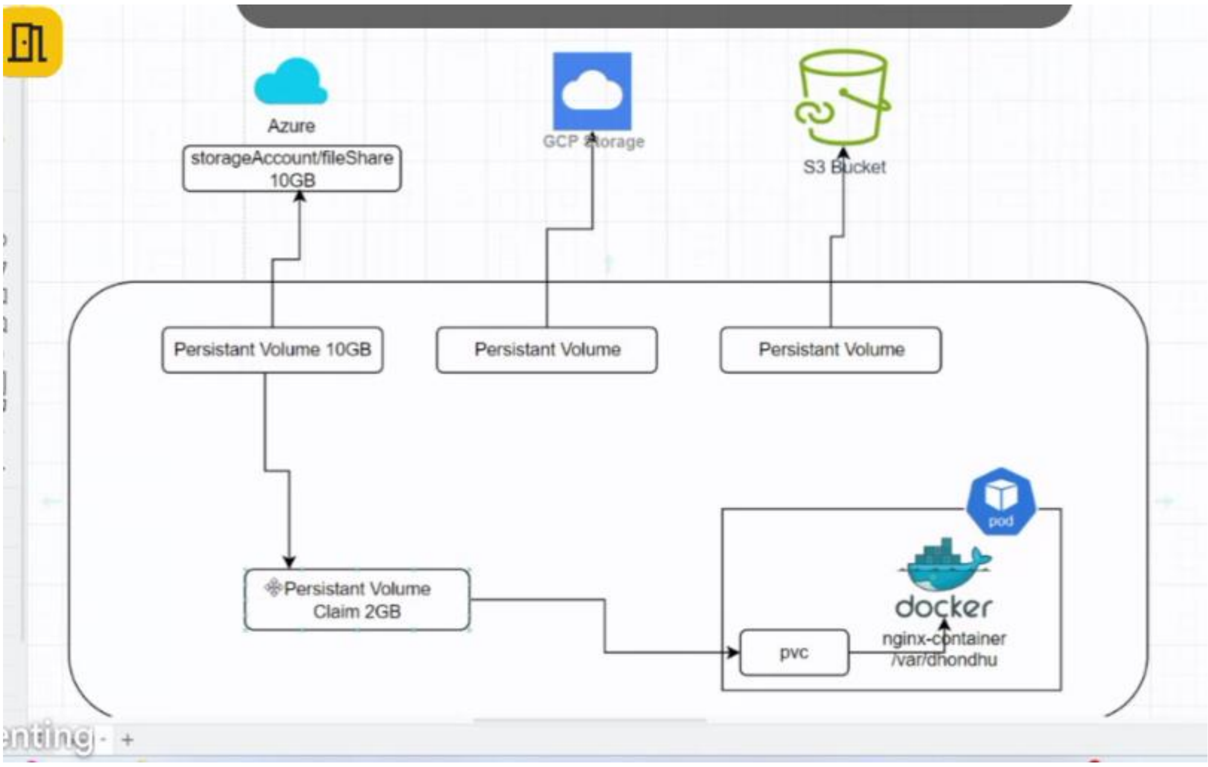
Here ram.txt file is showing so its mounted on both containers nginx as well as tomcat.

++

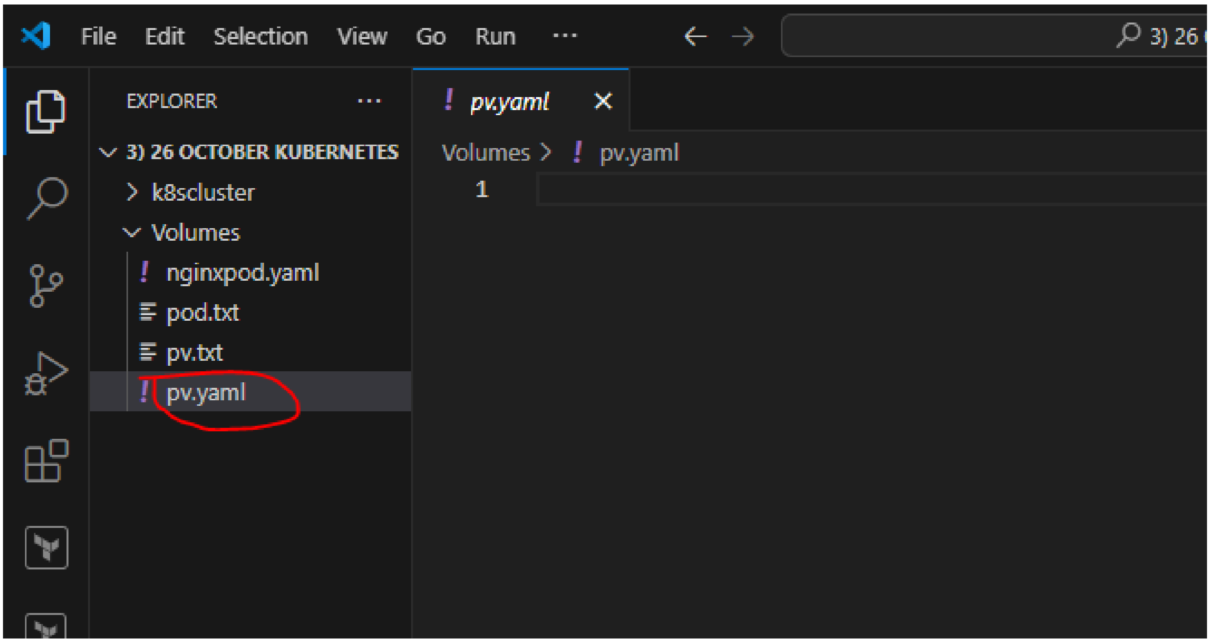
AGENDA – PV AND PVC

1)

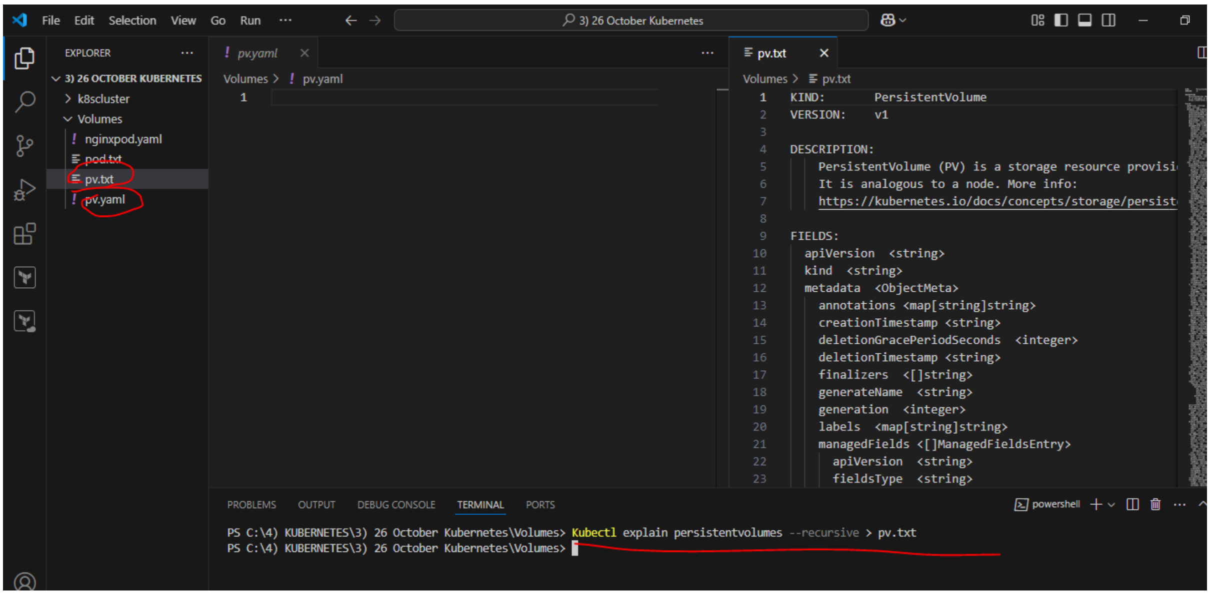




2) Creating pv YAML file= pv.yaml in vscode



3) **Kubectl explain persistentvolumes --recursive > pv.txt** = take out doc of persistentvolumes



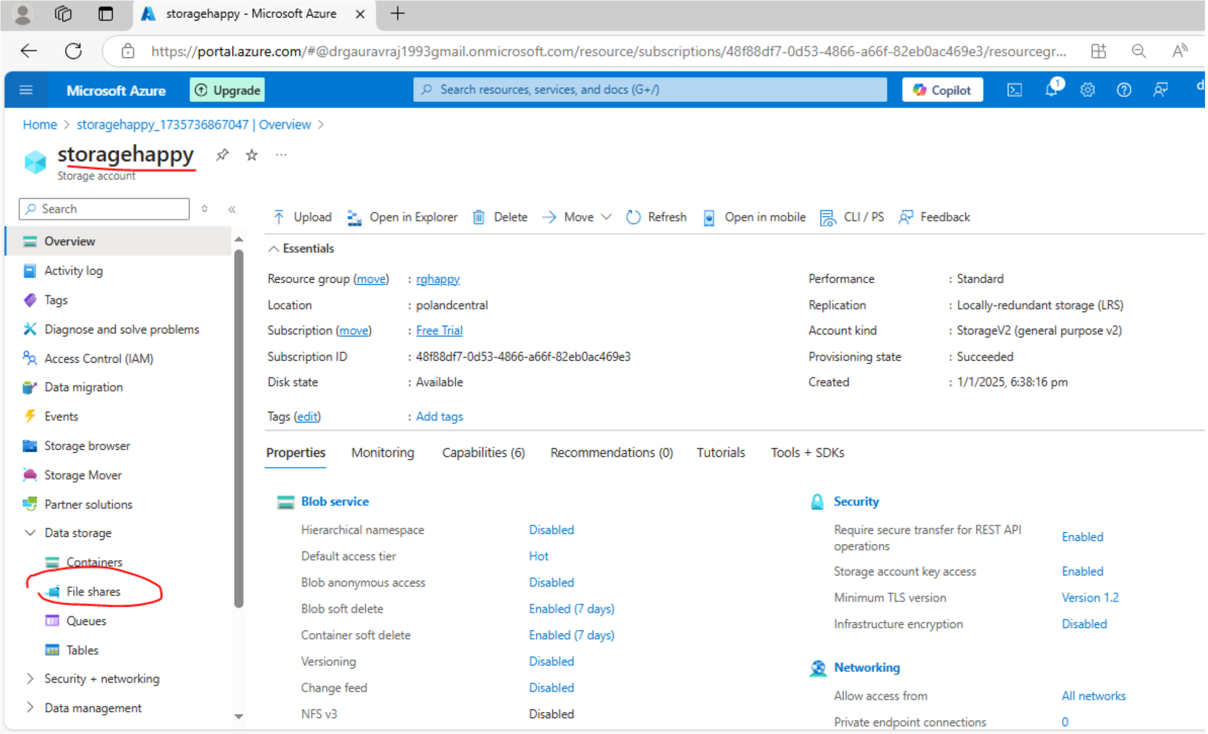
```
KIND:       PersistentVolume
VERSION:    v1

DESCRIPTION:
    PersistentVolume (PV) is a storage resource provisi
    It is analogous to a node. More info:
    https://kubernetes.io/docs/concepts/storage/persist

FIELDS:
  apiVersion  <string>
  kind  <string>
  metadata  <ObjectMeta>
    annotations <map[string]string>
    creationTimestamp <string>
    deletionGracePeriodSeconds  <integer>
    deletionTimestamp <string>
    finalizers  <[]string>
    generateName  <string>
    generation  <integer>
    labels  <map[string]string>
    managedFields <[]ManagedFieldsEntry>
      apiVersion  <string>
      fieldsType  <string>
```

```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> Kubectl explain persistentvolumes --recursive > pv.txt
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

4) Create storage account for file share



Essentials

Resource group (move) : rghappy
Location : polandcentral
Subscription (move) : Free Trial
Subscription ID : 48f88df7-0d53-4866-a66f-82eb0ac469e3
Disk state : Available
Tags (edit) : Add tags

Performance : Standard
Replication : Locally-redundant storage (LRS)
Account kind : StorageV2 (general purpose v2)
Provisioning state : Succeeded
Created : 1/1/2025, 6:38:16 pm

Blob service

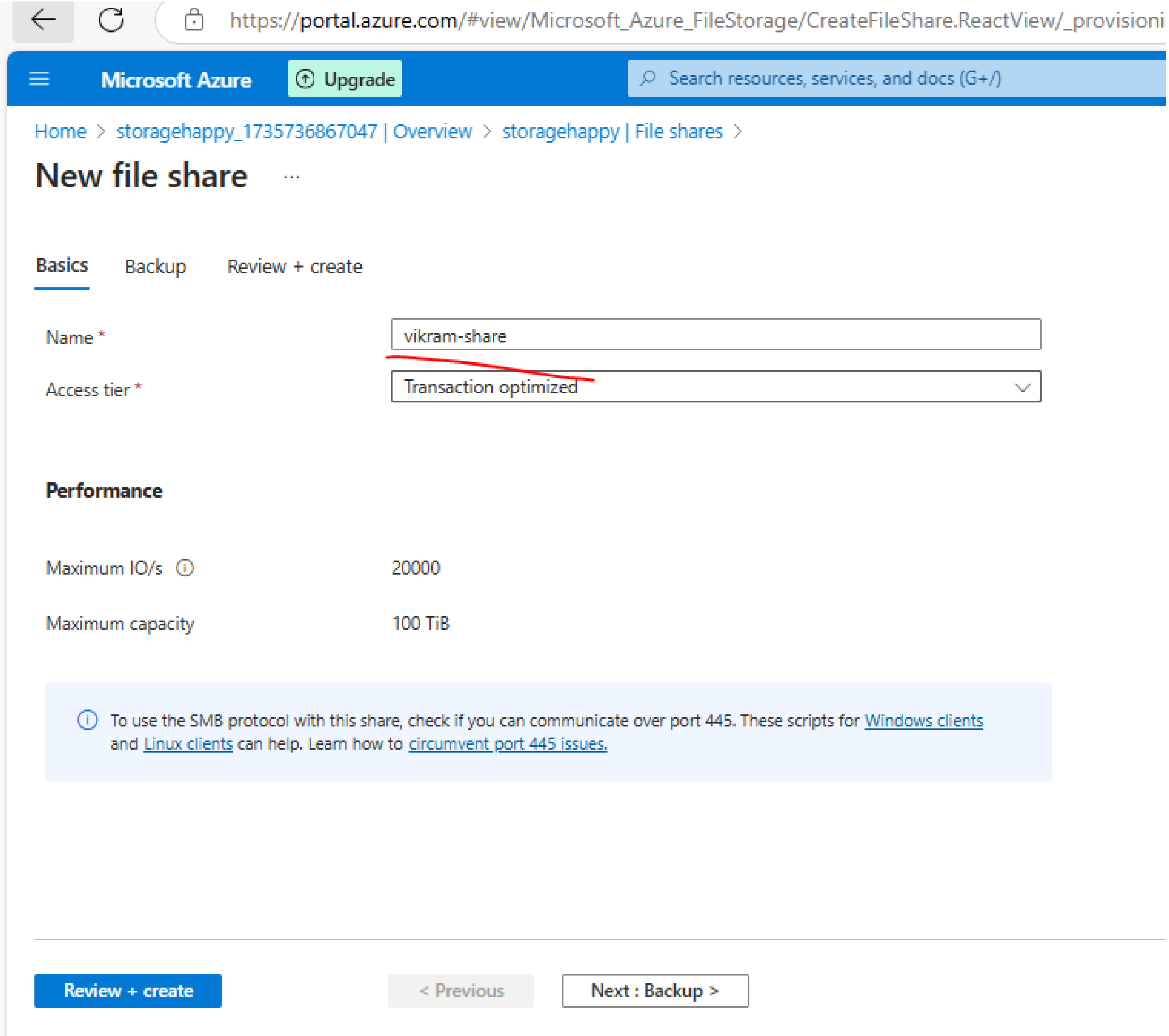
Hierarchical namespace Disabled
Default access tier Hot
Blob anonymous access Disabled
Blob soft delete Enabled (7 days)
Container soft delete Enabled (7 days)
Versioning Disabled
Change feed Disabled
NFS v3 Disabled

Security

Require secure transfer for REST API operations Enabled
Storage account key access Enabled
Minimum TLS version Version 1.2
Infrastructure encryption Disabled

Networking

Allow access from All networks
Private endpoint connections 0


https://portal.azure.com/#view/Microsoft_Azure_FileStorage/CreateFileShare.ReactView/_provisioni
Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > storagehappy_1735736867047 | Overview > storagehappy | File shares >
Basics
Backup
Review + create
Review + create
< Previous
Next : Backup >

New file share

Name * vikram-share

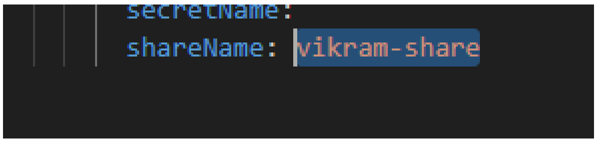
Access tier * Transaction optimized

Performance

Maximum IO/s 20000

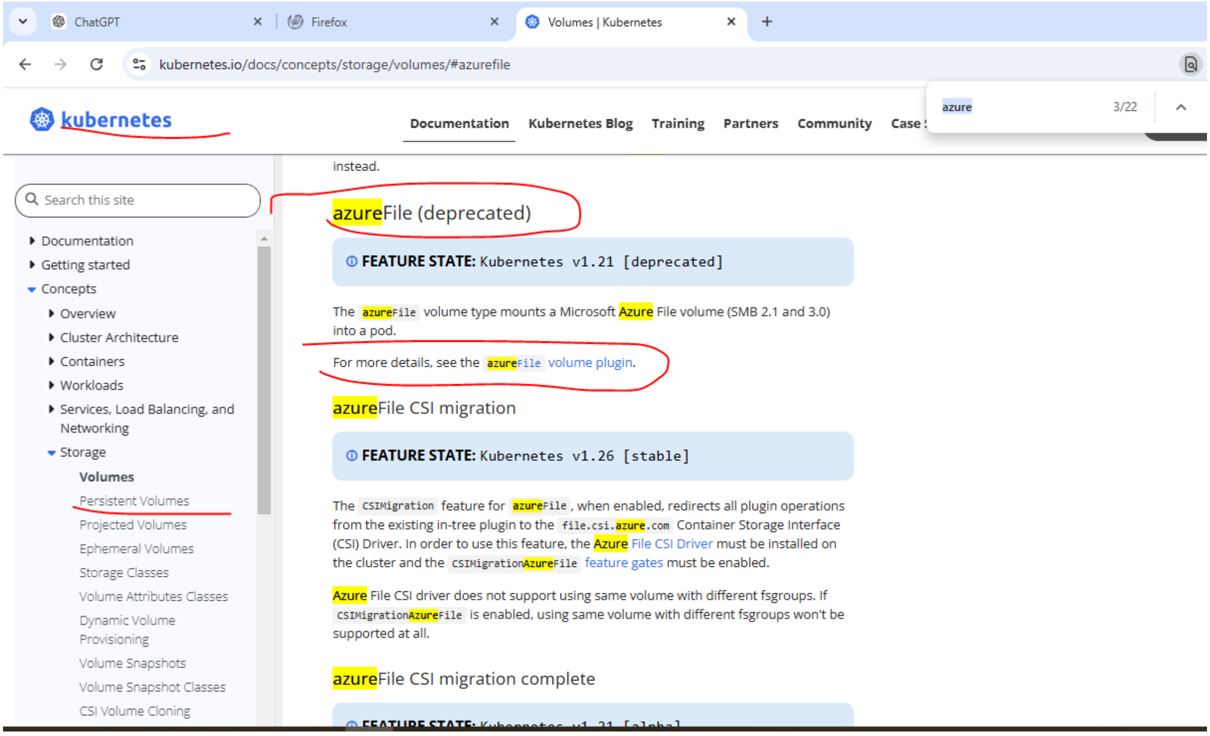
Maximum capacity 100 TiB

To use the SMB protocol with this share, check if you can communicate over port 445. These scripts for Windows clients and Linux clients can help. Learn how to circumvent port 445 issues.



```
        secretName:
        shareName: vikram-share
```

5)



instead.

azureFile (deprecated)

FEATURE STATE: `Kubernetes v1.21 [deprecated]`

The `azureFile` volume type mounts a Microsoft Azure File volume (SMB 2.1 and 3.0) into a pod.

For more details, see the `azureFile` volume plugin.

azureFile CSI migration

FEATURE STATE: `Kubernetes v1.26 [stable]`

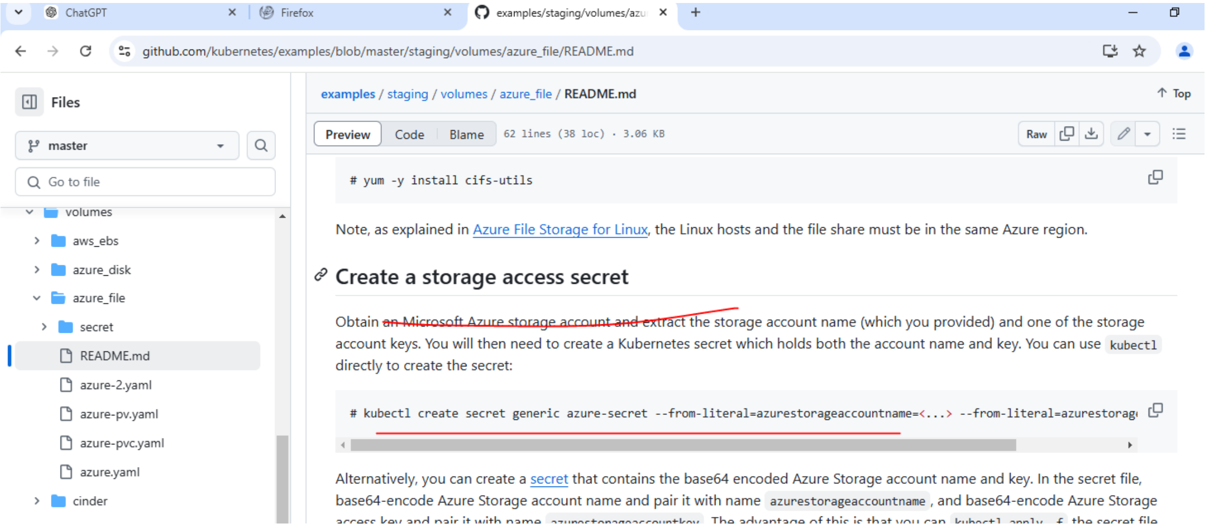
The `CSIMigration` feature for `azureFile`, when enabled, redirects all plugin operations from the existing in-tree plugin to the `file.csi.azure.com` Container Storage Interface (CSI) Driver. In order to use this feature, the Azure File CSI Driver must be installed on the cluster and the `CSIMigrationAzureFile` feature gates must be enabled.

Azure File CSI driver does not support using same volume with different fsgroups. If `CSIMigrationAzureFile` is enabled, using same volume with different fsgroups won't be supported at all.

azureFile CSI migration complete

FEATURE STATE: `Kubernetes v1.21 [alpha]`

6)


ChatGPT
Firefox
examples/staging/volumes/azu
github.com/kubernetes/examples/blob/master/staging/volumes/azure_file/README.md
Files
master
Go to file
volumes
aws_ebs
azure_disk
azure_file
secret
README.md
azure-2.yaml
azure-pv.yaml
azure-pvc.yaml
azure.yaml
cinder
examples / staging / volumes / azure_file / README.md
Top
Preview
Code
Blame
62 lines (38 loc) · 3.06 KB
Raw

```
# yum -y install cifs-utils
```

Note, as explained in Azure File Storage for Linux, the Linux hosts and the file share must be in the same Azure region.

Create a storage access secret

Obtain ~~an Microsoft Azure storage account and extract~~ the storage account name (which you provided) and one of the storage account keys. You will then need to create a Kubernetes secret which holds both the account name and key. You can use `kubectl` directly to create the secret:

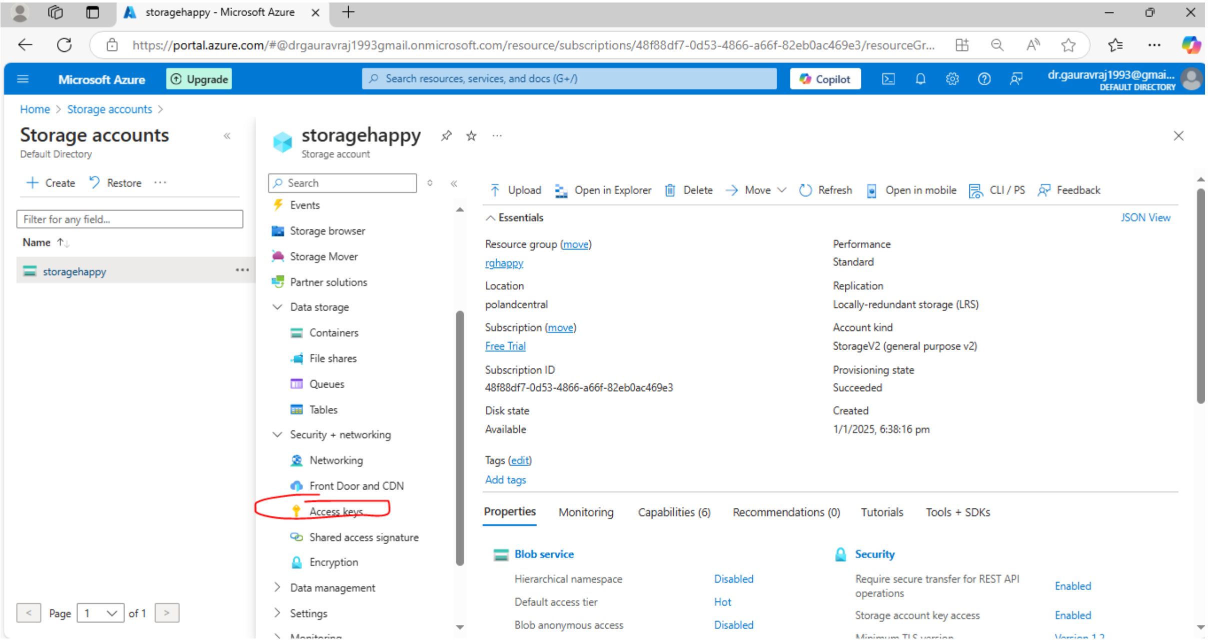
```
# kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=<...> --from-literal=azurestorag
```

Alternatively, you can create a secret that contains the base64 encoded Azure Storage account name and key. In the secret file, base64-encode Azure Storage account name and pair it with name `azurestorageaccountname`, and base64-encode Azure Storage access key and pair it with name `azurestorageaccountkey`. The advantage of this is that you can `kubectl apply -f` the secret file

kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=<...> --from-literal=azurestorageaccountkey=<...>

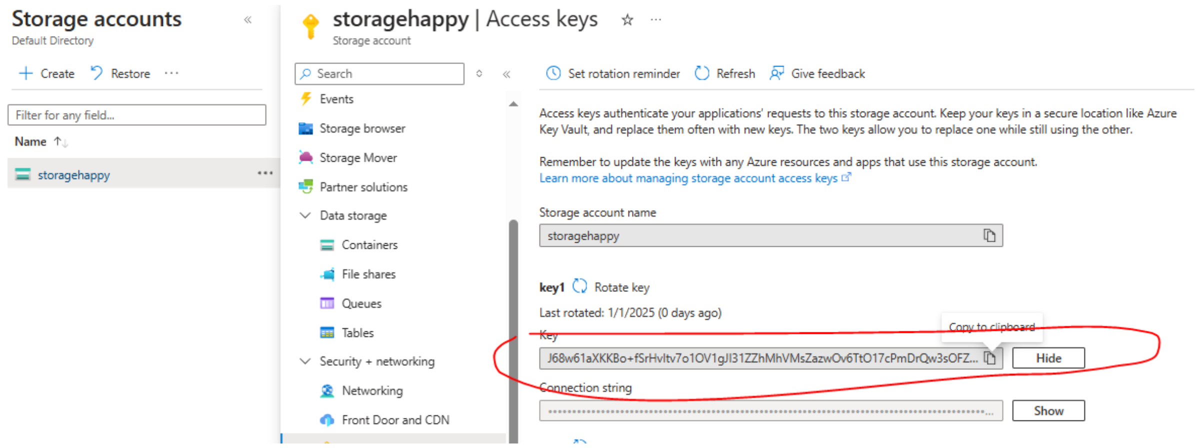
NOTE: Kuberenetes has its own secret which is same as keyvault

7) To access file share we have to firstly access storage account which we can do through its access keys



Resource group (move): rghappy
Location: polandcentral
Subscription (move): Free Trial
Subscription ID: 48f88df7-0d53-4866-a66f-82eb0ac469e3
Disk state: Available
Tags (edit): Add tags
Performance: Standard
Replication: Locally-redundant storage (LRS)
Account kind: StorageV2 (general purpose v2)
Provisioning state: Succeeded
Created: 1/1/2025, 6:38:16 pm

Hierarchical namespace: Disabled
Default access tier: Hot
Blob anonymous access: Disabled
Require secure transfer for REST API operations: Enabled
Storage account key access: Enabled



Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account.
Learn more about managing storage account access keys

Storage account name: storagehappy

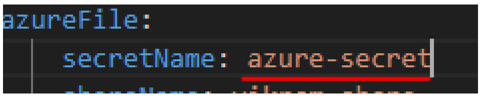
key1 Rotate key
Last rotated: 1/1/2025 (0 days ago)
Key: J68w61aXKKBo+fSrHvltv7o1OV1gJl31ZZhMhVMsZazwOv6TtO17cPmDrQw3sOFZ...
Connection string: ••

J68w61aXKKBo+fSrHvltv7o1OV1gJl31ZZhMhVMsZazwOv6TtO17cPmDrQw3sOFZvZrmKgyP7u/p+AStp3ZuOg==

8) Now putting storage account name and key in below command

# kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=<...> --from-literal=azurestorageaccountkey=<...>

# kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=storagehappy --from-literal=azurestorageaccountkey=J68w61aXKKBo+fSrHvltv7o1OV1gJl31ZZhMhVMsZazwOv6TtO17cPmDrQw3sOFZvZrmKgyP7u/p+AStp3ZuOg==

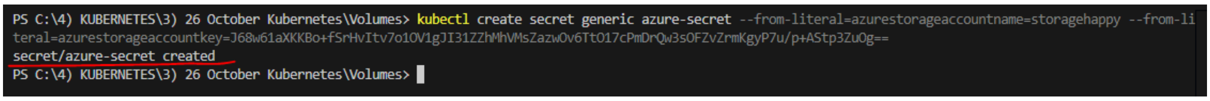


```
azureFile:
  secretName: azure-secret
```

9) Now run above command which will create secret in k8s

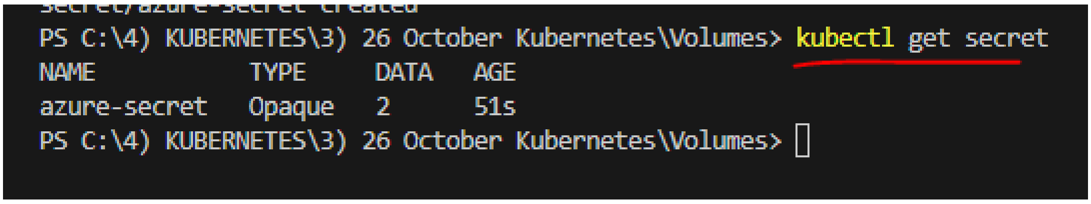
kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=storagehappy --from-

literal=azurestorageaccountkey=J68w61aXKKBo+fSrHvItv7o1OV1gJI31ZZhMhVMsZazwOv6TtO17cPmDrQw3sOFZvZrmKgyP7u/p+AStp3ZuOg==



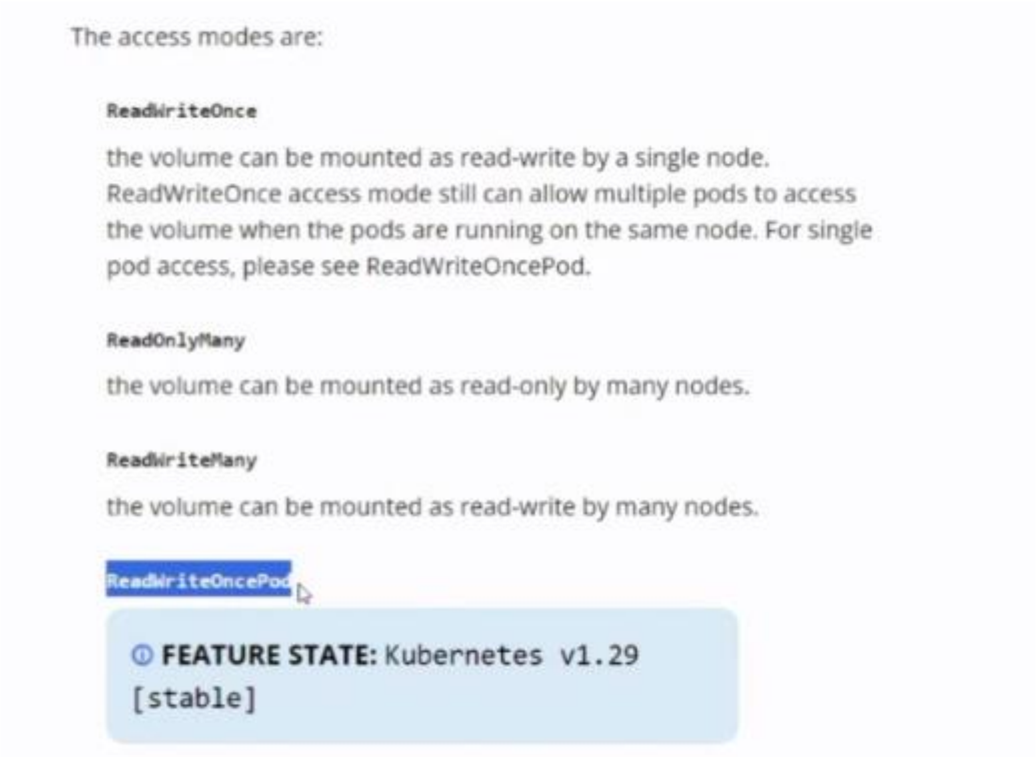
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=storagehappy --from-li
teral=azurestorageaccountkey=J68w61aXKKBo+fSrHvItv7o1OV1gJI31ZZhMhVMsZazwOv6TtO17cPmDrQw3sOFZvZrmKgyP7u/p+AStp3ZuOg==
secret/azure-secret created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

10) **kubectl get secret**



```
secret/azure-secret created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get secret
NAME           TYPE     DATA   AGE
azure-secret   Opaque   2      51s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

11)



The access modes are:

`ReadWriteOnce`

the volume can be mounted as read-write by a single node. ReadWriteOnce access mode still can allow multiple pods to access the volume when the pods are running on the same node. For single pod access, please see ReadWriteOncePod.

`ReadOnlyMany`

the volume can be mounted as read-only by many nodes.

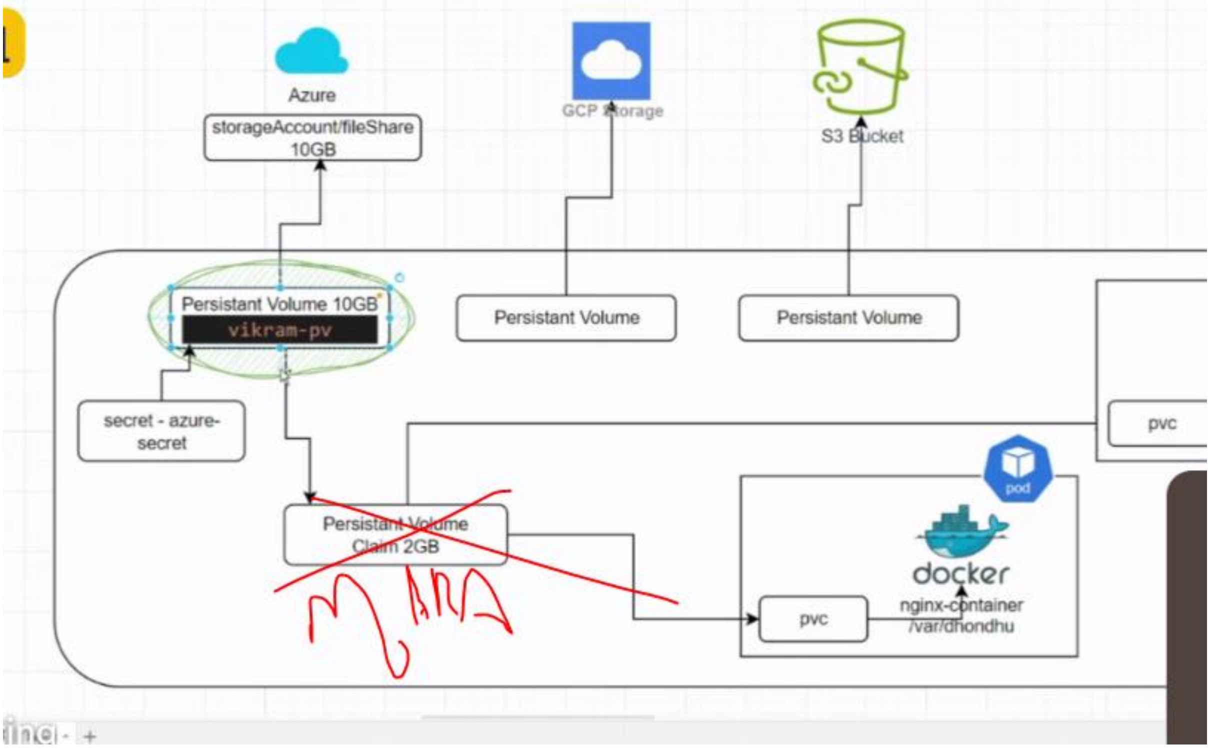
`ReadWriteMany`

the volume can be mounted as read-write by many nodes.

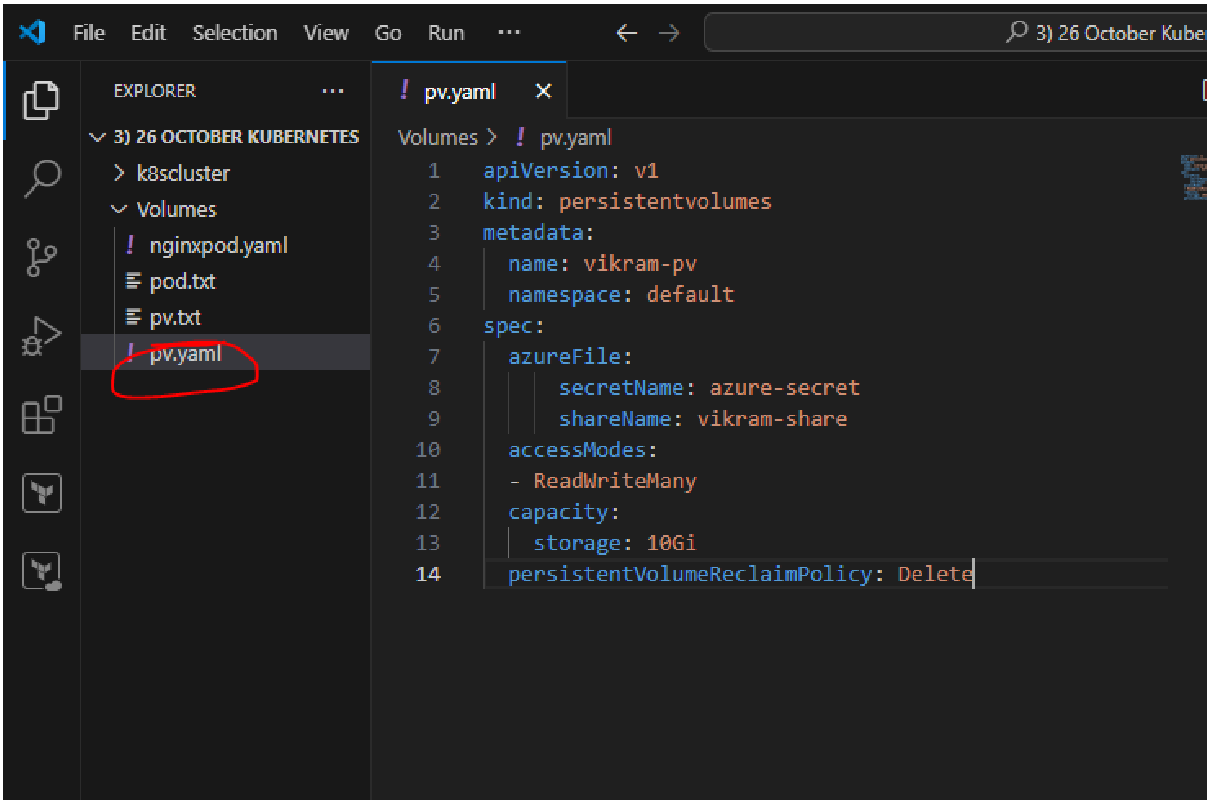
`ReadWriteOncePod`

FEATURE STATE: `Kubernetes v1.29 [stable]`

12) **persistentVolumeReclaimPolicy** = jab pvc ,marega to jo pv pr data hai uske sath kya krenge is decided by this policy. 3 steps are retain, delete, recycle

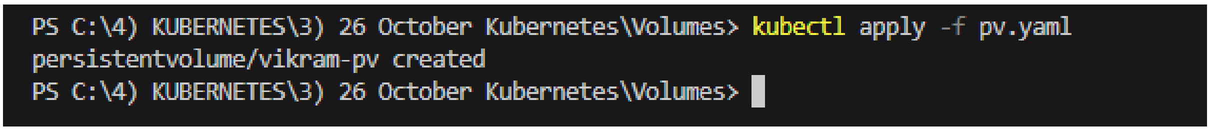


13) pv.yaml



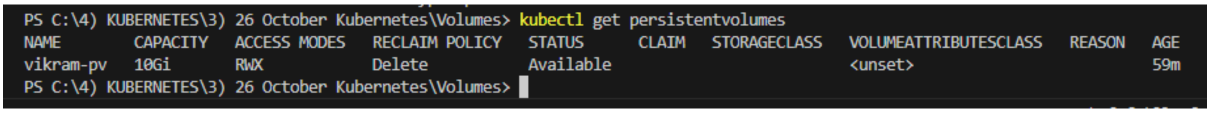
```yaml
apiVersion: v1
kind: persistentvolumes
metadata:
  name: vikram-pv
  namespace: default
spec:
  azureFile:
    secretName: azure-secret
    shareName: vikram-share
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  persistentVolumeReclaimPolicy: Delete
```

14) **kubectl apply -f pv.yaml** = creating pv



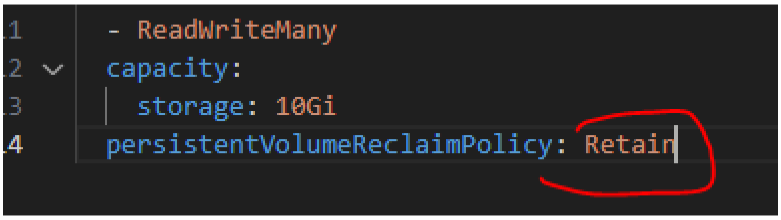
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pv.yaml
persistentvolume/vikram-pv created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

15) **kubectl get persistentvolumes**



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get persistentvolumes
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
vikram-pv   10Gi       RWX            Delete           Available                          <unset>                          59m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

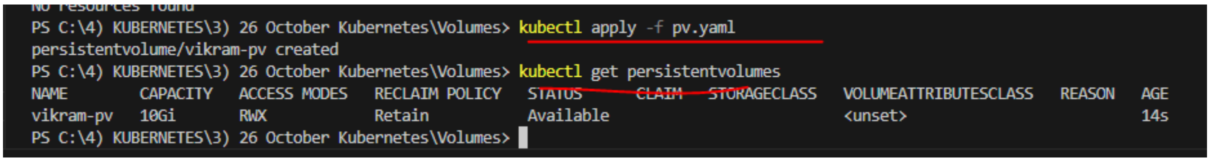
16)



```
  - ReadWriteMany
capacity:
  storage: 10Gi
persistentVolumeReclaimPolicy: Retain
```

kubectl apply -f pv.yaml

kubectl get persistentvolumes

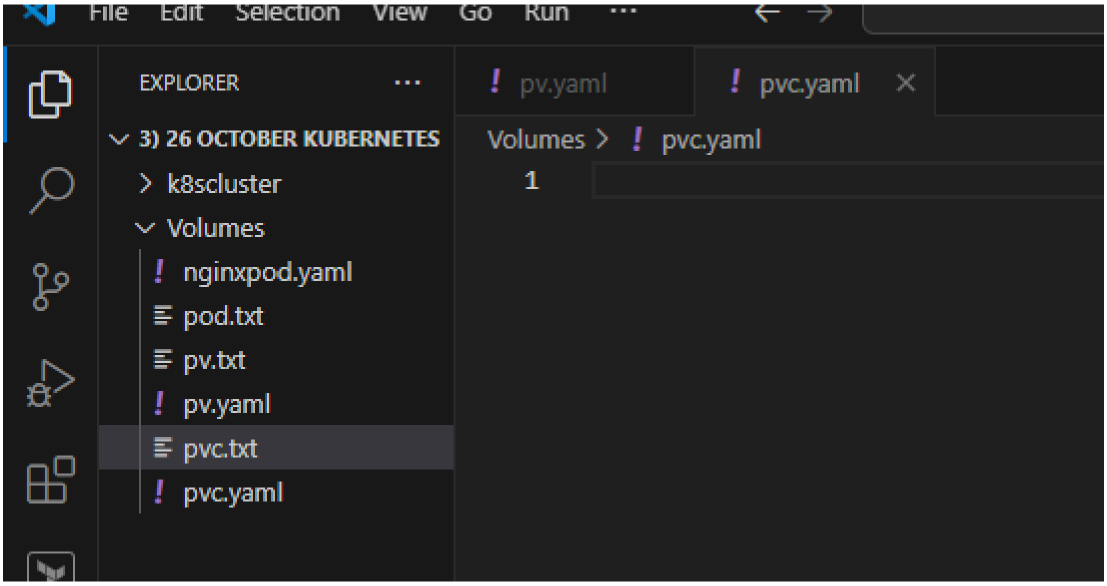


```
No resources found
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pv.yaml
persistentvolume/vikram-pv created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get persistentvolumes
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
vikram-pv   10Gi       RWX            Retain           Available                          <unset>                          14s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

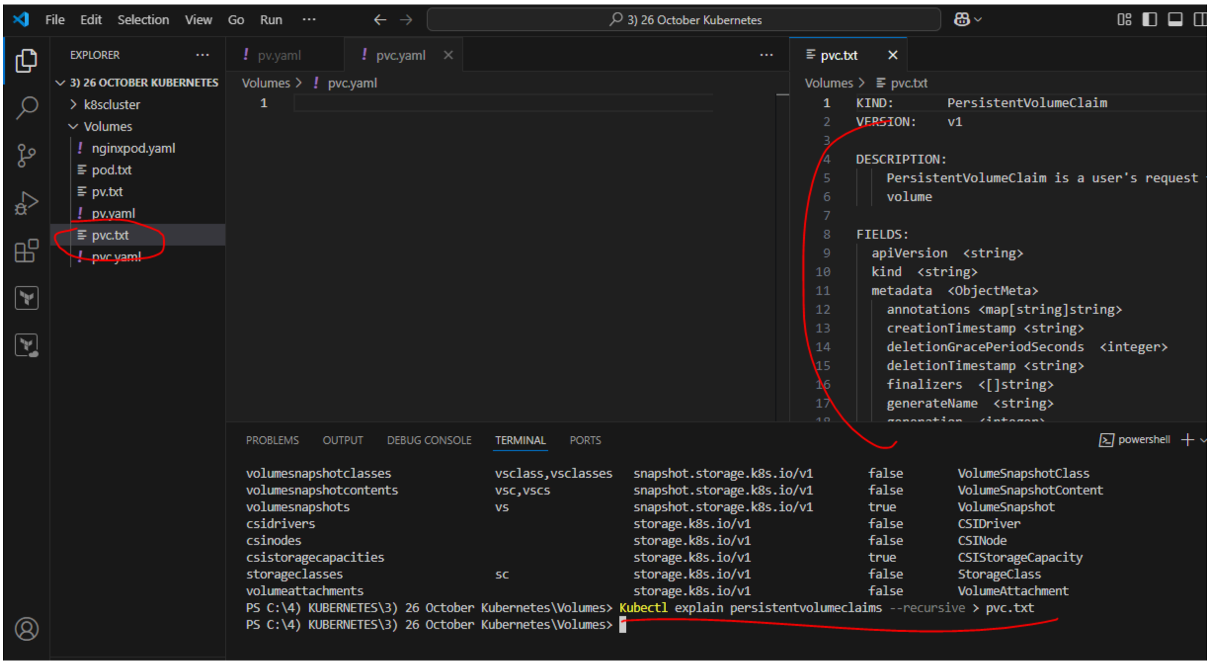
++

AGENDA – CREATING PVC YAML

1) pvc.yaml



2) **Kubectl explain persistentvolumeclaims --recursive > pvc.txt**



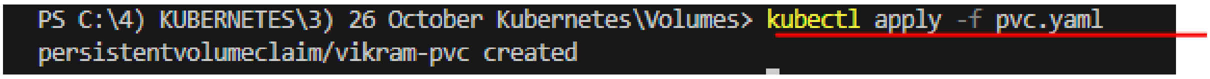
```
KIND:       PersistentVolumeClaim
VERSION:    v1

DESCRIPTION:
    PersistentVolumeClaim is a user's request
    volume

FIELDS:
  apiVersion  <string>
  kind  <string>
  metadata  <ObjectMeta>
    annotations <map[string]string>
    creationTimestamp <string>
    deletionGracePeriodSeconds  <integer>
    deletionTimestamp <string>
    finalizers  <[]string>
    generateName  <string>
```

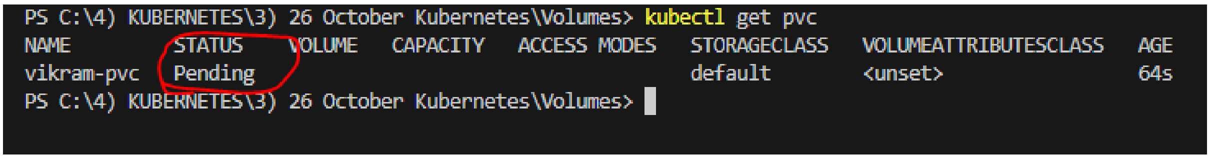
```
volumesnapshotclasses          vsclass,vsclasses   snapshot.storage.k8s.io/v1      false       VolumeSnapshotClass
volumesnapshotcontents         vsc,vscs            snapshot.storage.k8s.io/v1      false       VolumeSnapshotContent
volumesnapshots                vs                  snapshot.storage.k8s.io/v1      true        VolumeSnapshot
csidrivers                                         storage.k8s.io/v1               false       CSIDriver
csinodes                                           storage.k8s.io/v1               false       CSINode
csistoragecapacities                               storage.k8s.io/v1               true        CSIStorageCapacity
storageclasses                 sc                  storage.k8s.io/v1               false       StorageClass
volumeattachments                                  storage.k8s.io/v1               false       VolumeAttachment
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> Kubectl explain persistentvolumeclaims --recursive > pvc.txt
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

3) **kubectl apply -f pvc.yaml** = creating pvc



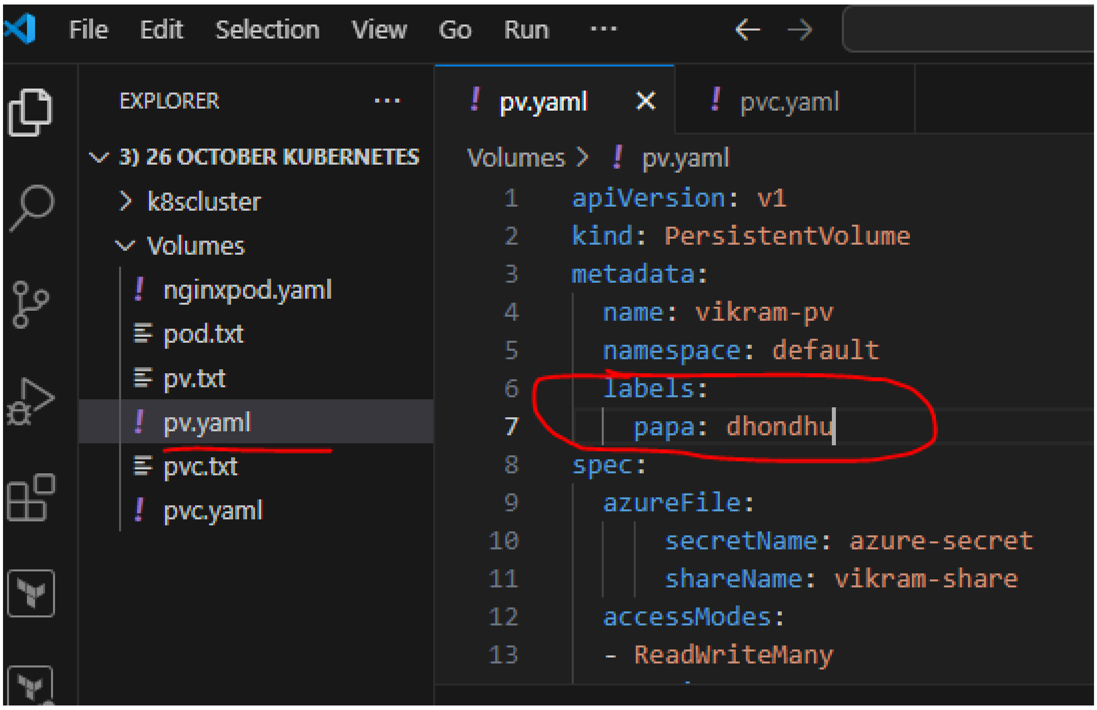
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pvc.yaml
persistentvolumeclaim/vikram-pvc created
```

4) **kubectl get pvc**



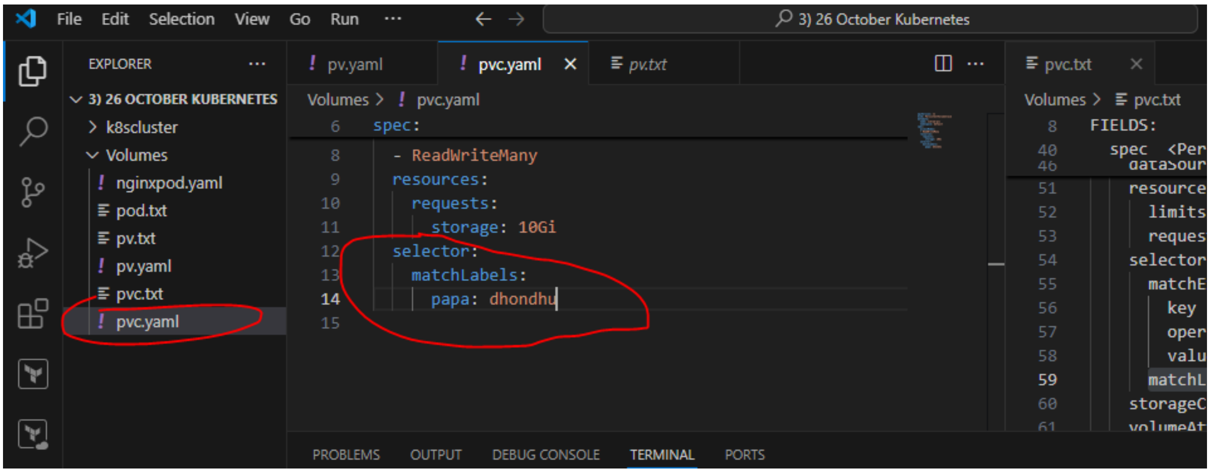
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pvc
NAME         STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
vikram-pvc   Pending                                      default        <unset>                 64s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

5) Now using labels in pv.yaml



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: vikram-pv
  namespace: default
  labels:
    papa: dhondhu
spec:
  azureFile:
    secretName: azure-secret
    shareName: vikram-share
  accessModes:
  - ReadWriteMany
```

6) Similarly put labels in pvc.yaml

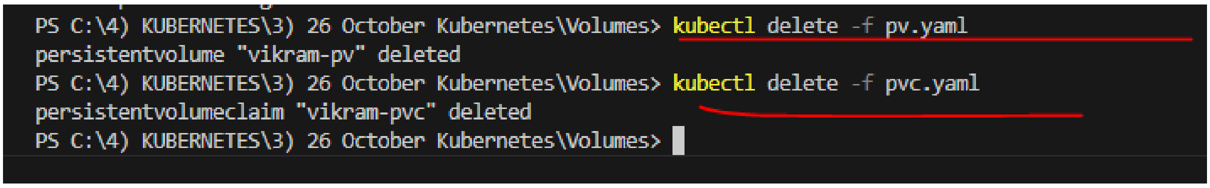


```
spec:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  selector:
    matchLabels:
      papa: dhondhu
```

```
FIELDS:
  spec  <Per
    dataSour
    resource
      limits
      reques
    selector
      matchE
        key
        oper
        valu
      matchL
    storageC
    volumeAt
```

7) So by using labels in pv and pvc yaml files we can bind them

8) Delete old pv and pvc

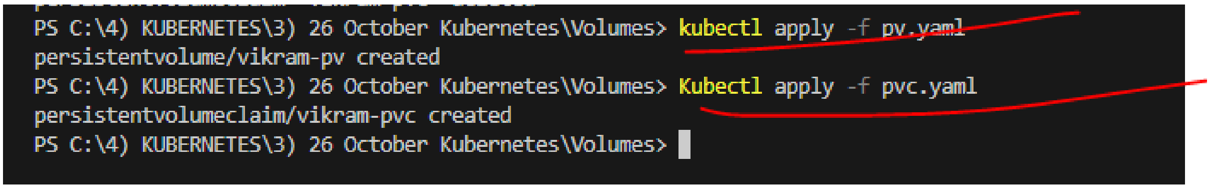


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete -f pv.yaml
persistentvolume "vikram-pv" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete -f pvc.yaml
persistentvolumeclaim "vikram-pvc" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

9) Now creating pv and pvc with labels

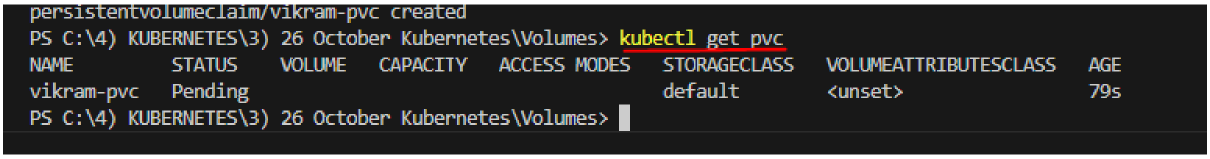
kubectl apply -f pv.yaml

Kubectl apply -f pvc.yaml



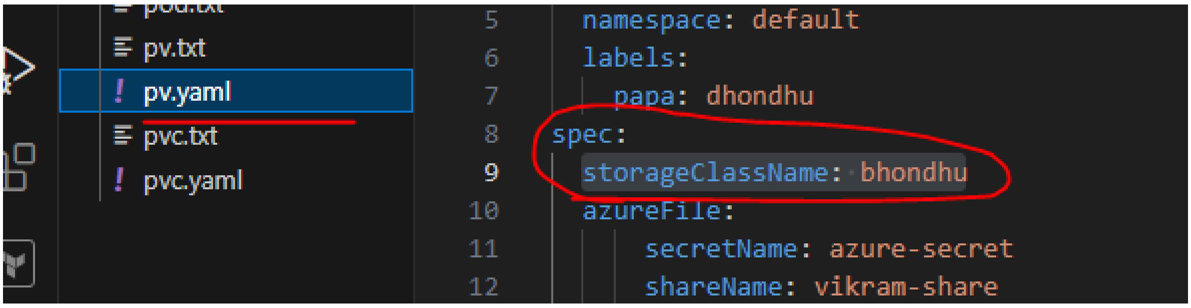
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pv.yaml
persistentvolume/vikram-pv created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> Kubectl apply -f pvc.yaml
persistentvolumeclaim/vikram-pvc created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

10) **kubectl get pvc** = still showing pending. So still its not binded so for this we actually have storage class label in k8s

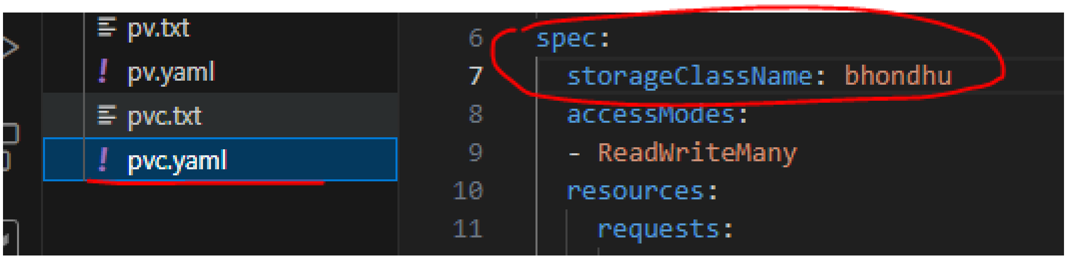


```
persistentvolumeclaim/vikram-pvc created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pvc
NAME         STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
vikram-pvc   Pending                                      default        <unset>                 79s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

11) Now add storageClassName in pv and pvc yaml

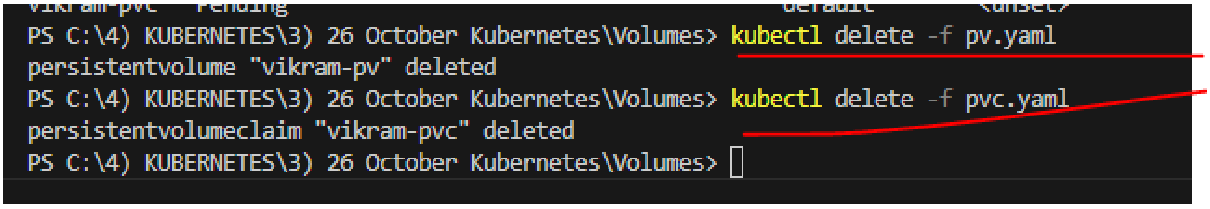


```yaml
  namespace: default
  labels:
    papa: dhondhu
spec:
  storageClassName: bhondhu
  azureFile:
    secretName: azure-secret
    shareName: vikram-share
```



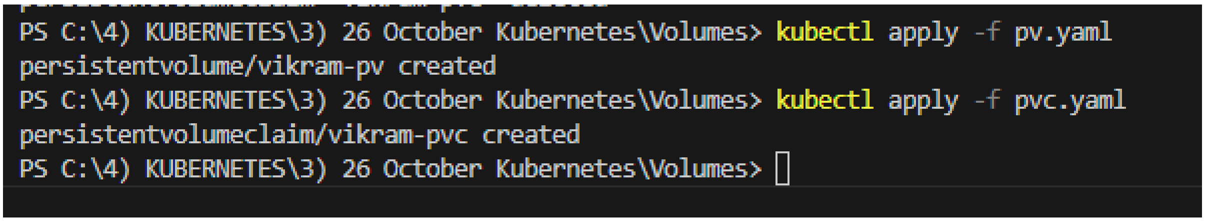
```yaml
spec:
  storageClassName: bhondhu
  accessModes:
  - ReadWriteMany
  resources:
    requests:
```

12) Again delete



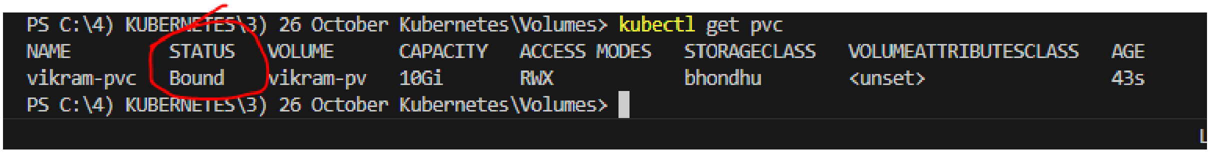
```
vikram-pvc   Pending                                    default         <unset>
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete -f pv.yaml
persistentvolume "vikram-pv" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete -f pvc.yaml
persistentvolumeclaim "vikram-pvc" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> 
```

13) Again create



```
persistentvolumeclaim "vikram-pvc" deleted
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pv.yaml
persistentvolume/vikram-pv created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pvc.yaml
persistentvolumeclaim/vikram-pvc created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> 
```

14) **kubectl get pvc**= Now we can see pv and pvc got bounded

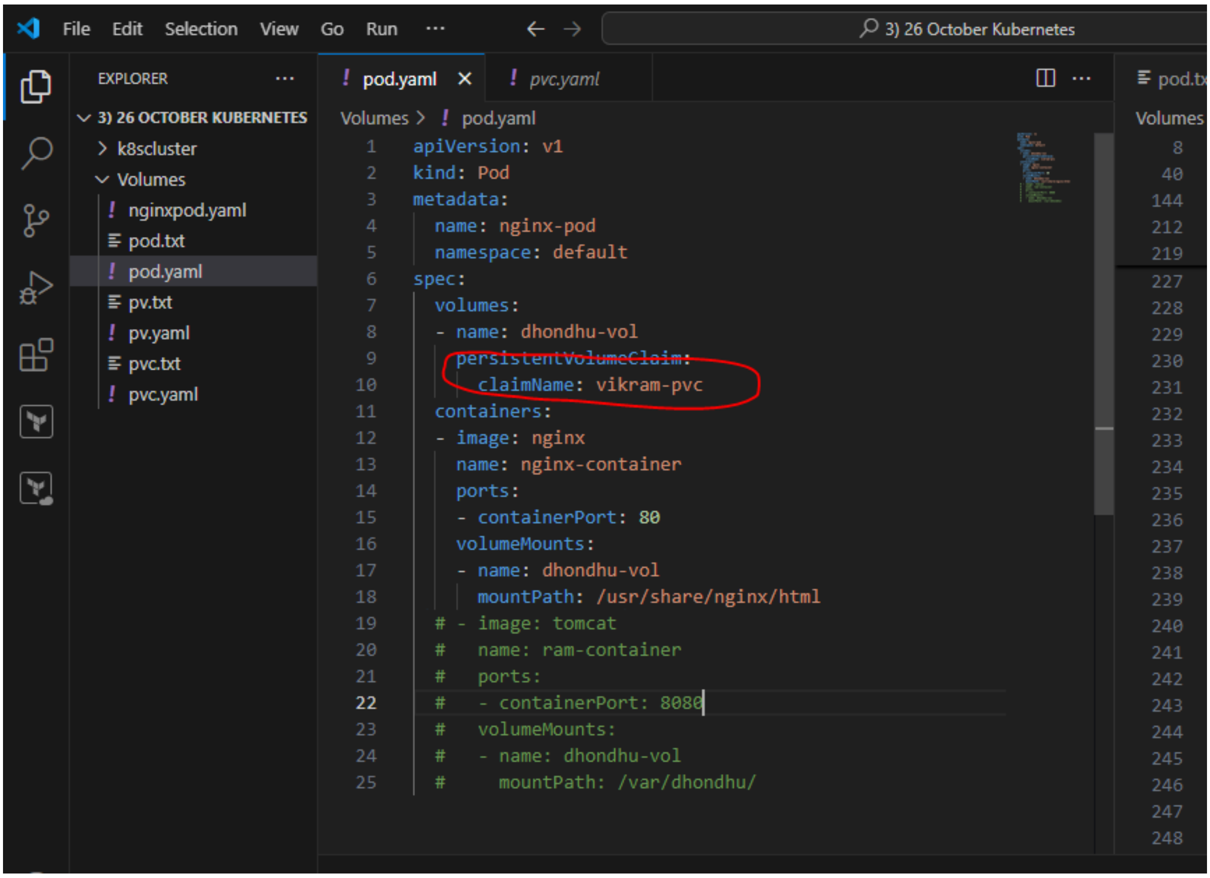


```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pvc
NAME         STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
vikram-pvc   Bound    vikram-pv   10Gi       RWX            bhondhu        <unset>                 43s
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> 
```

++

AGENDA – NOW WRITING POD

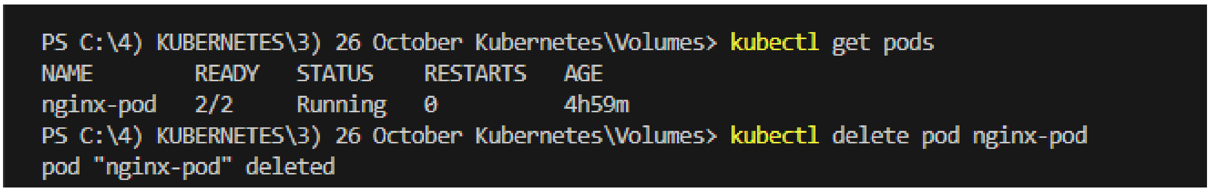
1) Create pod.yaml file = basically creating pod to connect to created and binded pv and pvc



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
spec:
  volumes:
  - name: dhondhu-vol
    persistentVolumeClaim:
      claimName: vikram-pvc
  containers:
  - image: nginx
    name: nginx-container
    ports:
    - containerPort: 80
    volumeMounts:
    - name: dhondhu-vol
      mountPath: /usr/share/nginx/html
  # - image: tomcat
  #   name: ram-container
  #   ports:
  #   - containerPort: 8080
  #   volumeMounts:
  #   - name: dhondhu-vol
  #     mountPath: /var/dhondhu/
```

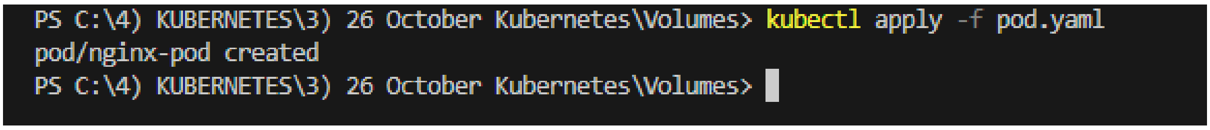
2) **kubectl get pods**

kubectl delete pod nginx-pod = delete old pods



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   2/2     Running   0          4h59m
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl delete pod nginx-pod
pod "nginx-pod" deleted
```

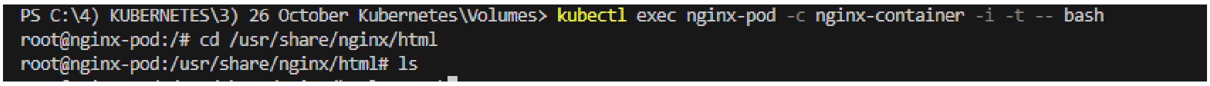
3) **kubectl apply -f pod.yaml**= create new pod



```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl apply -f pod.yaml
pod/nginx-pod created
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes>
```

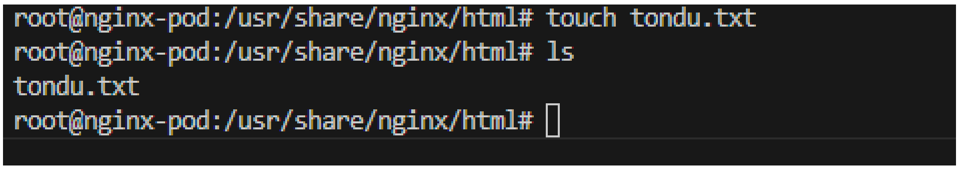
4) **kubectl exec nginx-pod -c nginx-container -i -t – bash**

cd /usr/share/nginx/html



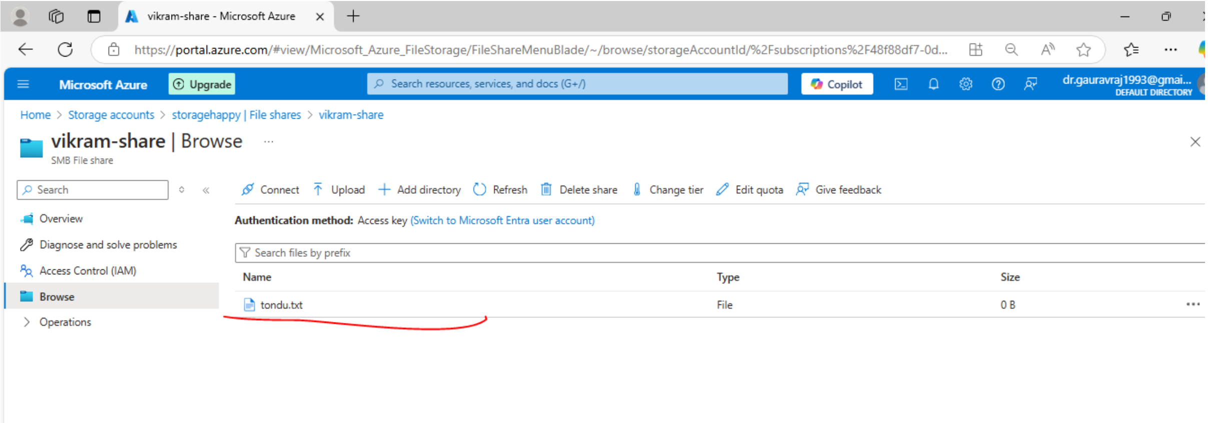
```
PS C:\4) KUBERNETES\3) 26 October Kubernetes\Volumes> kubectl exec nginx-pod -c nginx-container -i -t -- bash
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
```

5) touch tondu.txt



```
root@nginx-pod:/usr/share/nginx/html# touch tondu.txt
root@nginx-pod:/usr/share/nginx/html# ls
tondu.txt
root@nginx-pod:/usr/share/nginx/html# 
```

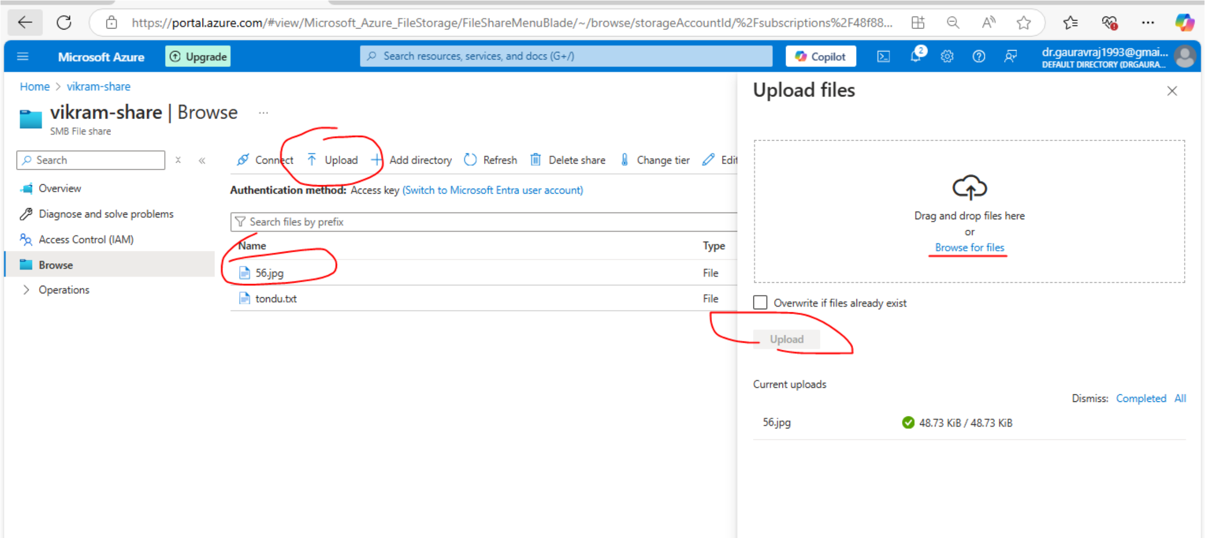
6) Now this file **tondu.txt** gone into file share of storage account


vikram-share - Microsoft Azure
https://portal.azure.com/#view/Microsoft_Azure_FileStorage/FileShareMenuBlade/~/browse/storageAccountId/%2Fsubscriptions%2F48f88df7-0d...
Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Copilot
dr.gauravraj1993@gmai...
DEFAULT DIRECTORY
Home > Storage accounts > storagehappy | File shares > vikram-share
vikram-share | Browse
SMB File share
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Browse
Operations
Connect
Upload
Add directory
Refresh
Delete share
Change tier
Edit quota
Give feedback
Search files by prefix

Authentication method: Access key (Switch to Microsoft Entra user account)

Name	Type	Size
tondu.txt	File	0 B

7) Now upload any file direct on portal and it should show in terminal



Authentication method: Access key (Switch to Microsoft Entra user account)

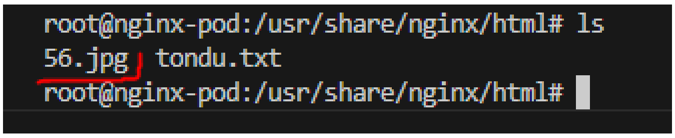
Name	Type
56.jpg	File
tondu.txt	File

□ Overwrite if files already exist

56.jpg ✓ 48.73 KiB / 48.73 KiB

8) ls

So 56.jpeg file is showing on terminal as well



```
root@nginx-pod:/usr/share/nginx/html# ls
56.jpg  tondu.txt
root@nginx-pod:/usr/share/nginx/html# 
```

9) So basically pv and pvc was binded and then used by pod to access data at cluster level by removing emptyDir