

**Kindly Stop And Give Me Your Valuable
Feedback..**

Before going to next page kindly read this below paragraph.

**Please give me feedback about this type of helping material.
Is this helpful for you ?**

**Near in future I am energetic for establish the YouTube
Channel for education So I want to take your valuable
feedback for my about content
So kindly give me your opinion..**

**In Shaa Allah I Will Establish A YouTube Channel To Make
Learning Easy And Simple..For Everyone.**

Thank You

Kubernetes Lesson 5

Video 1. “Volumes”

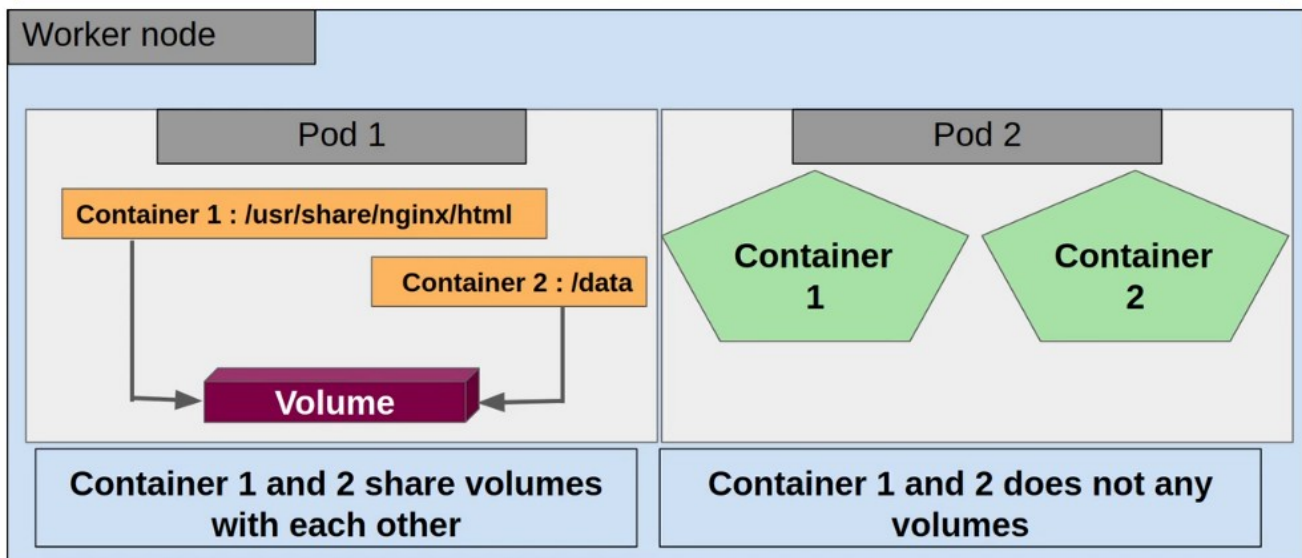
This Video Is Theory Based.

Video 2. “Volumes Flow”

This Video Is Diagram Based.



Volumes



Video 3. “Volumes Types”

Volumes

- There are many types of volumes
 - emptyDir
 - configMap , secret , downwardAPI
 - persistentVolumeClaim
 - gitRepo
 - gcePersistentDisk
 - awsElasticBlockStore
 - azureDisk

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 4. "Volumes Example Part 1"

apiVersion: v1
kind: Pod
metadata:
 name: my-pod-with-vol
spec:
 volumes:
 - name: shared-dir
 emptyDir: {}
 containers:
 - name: container-one
 image: aamirpinger/logfile_nodejs
 ports:
 - containerPort: 8080
 volumeMounts:
 - name: shared-dir
 mountPath: /data
 - name: container-two
 image: nginx
 ports:
 - containerPort: 80
 volumeMounts:
 - name: shared-dir
 mountPath: /var/c-two

volumeMounts:
- name: shared-dir

This means that the container will mount the data in this volume that is defined on the top.

Note: There are many **Volumes** can be defined in one **Pod**.

Note: If There were many volumes in this pod so we can assign one or more than one from the list of the volume to the container or the containers.

- ✓ This yaml file is consists on two containers in a one pod.
- ✓ This yaml file is example of the **Volumes**.

Remember that: There are more than one **Volumes** can be mount in one container.

mountPath: /data

This line defines the path of the folder of the container that will be use by the container for storing the data **But** this folder is now shared so the data also will be store in the

volume directory. **The Container** will feel that the data is storing in the filesystem of it's own, indeed the data is storing in the pod level means volume.


- **Kubectl get pods -w**

This Command will show the running processes of the pods for example if one one pod is initially creating the container so the when the container will come in the running state so the status will be update creating to running.

Video 5. "Volumes Example Part 2"

This Video Is Pending To Clear Due To Non Availability Of Internet.


Video 6. "Persistent Volumes"



Persistent Volume

- Volumes were great as they saves us from data lost incase of container restart
- Volumes hold data at a pod level but question may be asked what if for any reason kubernetes terminates the pod e.g. rescheduling the pod
- In the case of pod termination data in the volumes will be lost
- To solve this issue Kubernetes provides us option of Persistent Volume
- Persistent Volume add a volume at a cluster level instead pod level

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)




Persistent Volume

- We create a Persistent Volume resource in which we offer cluster level volume that can be used by any pod
- Any pod can use that Persistent Volume by using another resource Persistent Volume Claims
- Kubernetes Persistent Volumes remains available outside of the pod lifecycle
- That means volume will remain even after the pod is deleted
- This volume will be available to claim by another pod if required, and the data is retained

```
graph LR;
  A[POD] --> B[PERSISTENT VOLUME CLAIM];
  B --> C[PERSISTENT VOLUME];
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 7. “Persistent Volume Claim”




Persistent Volume Claim

- It is a kind of formal request from user for claiming a persistent volume
- A Persistent Volume Claim describes the amount and characteristics of the storage required by the pod
- Based on requirement from user PVC finds any matching persistent volumes and claims it
- Depending on the configuration options used for Persistent Volume resource, these PV resource can later be used/claim by other pods

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 8. “Understanding Persistent Volume Specs”



Persistent Volume Access Modes

- Type of accessModes
 - ReadWriteOnce
 - Only a single node can mount the volume for reading and writing
 - ReadOnlyMany
 - Multiple nodes can mount the volume for reading
 - ReadWriteMany
 - Multiple nodes can mount the volume for both reading and writing
- RWO , ROX , and RWX pertain to the number of worker nodes that can use the volume at the same time, not to the number of pods!

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Persistent Volume Reclaim Policy

- We have learned that depending on the configuration options used for Persistent Volume resource, these PV resource can later be used/claim by other pods
- The lifetime of a Persistent Volume is determined by its reclaim policy
- Reclaim Policy controls the action the cluster will take when a pod releases its ownership of the storage
- `persistentVolumeReclaimPolicy` tag can be used in YAML configuration file at the time of creating PV

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Persistent Volume Reclaim Policy

- Reclaim Policy can be set to
 - Delete
 - Recycle
 - Retain (Default)
- If `persistentVolumeReclaimPolicy` is **Delete**
 - PersistentVolume will be deleted when the PVC is deleted but data will persist
- If `persistentVolumeReclaimPolicy` is **Recycle**
 - Volume's contents will be deleted
 - Persistent Volume will be available to be claimed again



Persistent Volume Reclaim Policy

- If `persistentVolumeReclaimPolicy` is **Retain**
 - If `persistentVolumeReclaimPolicy` not provided, Retain is default
 - Kubernetes will retain the volume and its contents after it's released from its claim
 - To make PersistentVolume available again for claims can be done by delete and recreate the PersistentVolume resource manually
 - Underlying storage can either delete or left to be reused by the next pod

Video 9. “Persistent Volume Example Part 1”

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv

spec:

accessModes:

- ReadWriteOnce

capacity:

storage: 100M

hostPath:

path: /tmp/pvexample

persistentVolumeReclaimPolicy: Delete

✓ **accessModes:**

✓ **- ReadWriteOnce**

The Access mode is defining that how many nodes will access this **Persistent Volume** and how type of permission they have means **Read Write etc.**

✓ **capacity:**

✓ **storage: 100M**

The Capacity is show that how much storage space is fixed for storing the file on the hard disk

✓ **path: /tmp/pvexample**

The Path will define that if the persistent volume will be deleted so where the files will available for us. This is the path of the cluster means host's path.

✓ **persistentVolumeReclaimPolicy: Delete**

This the policy that will be define if the Persistent Volume Claim will delete so which type of effects will fall on the **Persistent Volume**.

Video 10. “Persistent Volume Example Part 2

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 100m

storageClassName: ""

resources:

requests:

storage: 100M

The persistent volume claim will request for needed storage space for example 100MB.

Note: we can use the less or equal to the 100MB but we can exceed the 100MB. If we will exceed the limit the kubernetes will not be able to find the resource and mount too.

storageClassName: ""

*The Storage Class Name is managed by the cluster administration and as a developer we have no need to fill out this line for the purpose of creating the **PV** or **PVC** resource.*

Video 11. "Persistent Volume Example Part 3"

apiVersion: v1

kind: Pod

metadata:

name: pod-pv

spec:

volumes:

- name: willusepvc

persistentVolumeClaim:

claimName: pvc

containers:

- name: container1

image: aamirpinger/logfile_nodejs

volumeMounts:

- name: willusepvc

mountPath: /data

volumes:

- name: willusepvc

persistentVolumeClaim:

claimName: pvc

*This means we always use volumes options in the pod and use the options for any type of volume like **EmptyDir** but here we will not use but we will use the PVC to claim/use for the PV therefore, we will not use the any other type of the volume for mounting the volume.*

mountPath: /data

This image by default stores data in the data directory at the root. Therefore, we used the data directory in the mountpath. If we will not use the data directory so the container will create a folder/directory and will save the data in it but the data will not store in the PV.

Video 12. “Persistent Volume Example Part 4”

- ***minikube ssh***

Remember: For Practice we are using minikube and using this we get the minikube cluster that cluster is in the **Virtual Box**.

By using this command we enter in the cluster that made by the minikube in the Virtual Box and finally we get the access of the **Master Nodes and Worker Nodes** and get the access of the files that the volume saves in the cluster level through the **PV**

Remaining Video Is Practical Based.

Alhamdulillah!

Kubernetes Lesson 5 Completed.

This Document Is Created By “Rizwan Sheikh”

Email: rizwansheikh7071@gmail.com

Docker ID: rizwansheikh7071

Due To Human Being The Mistake Can Be Done!

Thank You Happy Learning.