

# Configuring NFS Storage Persistence Volume (PV, PVC, Pod Using PVC)

[Edition 3]

[Last Update 210831]

For any issues/help contact : [support@k21academy.com](mailto:support@k21academy.com)

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>2</b>	<b>Documentation.....</b>	<b>4</b>
2.1	Kubernetes Documentation .....	4
<b>3</b>	<b>Pre-Requisite .....</b>	<b>5</b>
<b>4</b>	<b>Configuring NFS Storage Persistence Volume .....</b>	<b>6</b>
<b>5</b>	<b>Test by mounting resource To Worker Nodes .....</b>	<b>7</b>
<b>6</b>	<b>Creating a Persistent NFS Volume (PV) .....</b>	<b>8</b>
<b>7</b>	<b>Creating a Persistent Volume Claim (PVC) .....</b>	<b>10</b>
<b>8</b>	<b>Creating Pod to Mount NFS volume.....</b>	<b>11</b>
<b>9</b>	<b>Deleting Resources &amp; Verify Volume Behaviour .....</b>	<b>13</b>
<b>10</b>	<b>Troubleshooting.....</b>	<b>15</b>
10.1	Access denied when mounting NFS server .....	15
<b>11</b>	<b>Summary .....</b>	<b>17</b>

## 1 INTRODUCTION

**Managing storage** is a distinct problem from managing compute instances. The PersistentVolume subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed. To do this, we introduce two new API resources: PersistentVolume and PersistentVolumeClaim.

A **PersistentVolume (PV)** is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes, but have a lifecycle independent of any individual Pod that uses the PV. This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

A **PersistentVolumeClaim (PVC)** is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., they can be mounted ReadWriteOnce, ReadOnlyMany or ReadWriteMany, see AccessModes).

This guide Covers:

- Configuring NFS storage persistence volume
- Test by mounting resource to worker nodes
- Creating a persistent NFS volume (PV)
- Creating a persistent volume claim (PVC)
- Creating pod to mount NFS volume

---

## 2 DOCUMENTATION

---

### 2.1 Kubernetes Documentation

1. Persistent Volumes

<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>

2. Persistent Volumes Claim

<https://kubernetes.io/docs/concepts/storage/persistent-volumes/#persistentvolumeclaims>

3. Persistent Volumes

<https://kubernetes.io/docs/concepts/storage/volumes/>

---

### 3 PRE-REQUISITE

Ensure that you have completed following activity guides (or you have an Ubuntu Server)

**Note:** Follow Activity Guide **AG\_Bootstrap\_Kubernetes\_Cluster\_Using\_Kubeadm\_Guide\_ed\*\*** from portal

**Note:** Follow Activity Guide **AG\_Deploy\_App\_On\_Pod\_&\_Basic\_Networking\_ed\*\*** from portal

**Note:** Follow Activity Guide **AG\_Deploying\_Scalable\_and\_Configuring\_Autoscaling\_For\_Stateless\_Application\_ed\*\*** from portal

## 4 CONFIGURING NFS STORAGE PERSISTENCE VOLUME

1. Install the software on your master node

```
$ sudo apt-get update && sudo apt-get install -y nfs-kernel-server
```

```
root@kubeadm-master:/home/ubuntu#  
root@kubeadm-master:/home/ubuntu# sudo apt-get update && sudo \  
> apt-get install -y nfs-kernel-server  
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Hit:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done
```

2. Make and populate a directory to be shared. Also give it proper permissions

```
$ sudo mkdir /opt/sfw  
$ sudo chmod 1777 /opt/sfw/  
$ echo software > /opt/sfw/hello.txt
```

```
root@kubeadm-master:/home/ubuntu# echo software > /opt/sfw/hello.txt  
root@kubeadm-master:/home/ubuntu#  
root@kubeadm-master:/home/ubuntu# sudo mkdir /opt/sfw  
root@kubeadm-master:/home/ubuntu# sudo chmod 1777 /opt/sfw/
```

3. Edit the NFS server file to share out the newly created directory. In this case we will share the directory with all. Vim the file and add the below line

```
$ sudo vim /etc/exports
```

```
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)
```

```
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients. See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)  
#  
# Example for NFSv4:  
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)  
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)  
~  
~  
~
```

4. Cause /etc/exports to be re-read

```
$ sudo exportfs -ra
```

```
root@kubeadm-master:/home/ubuntu#  
root@kubeadm-master:/home/ubuntu# sudo exportfs -ra  
root@kubeadm-master:/home/ubuntu#
```

## 5 TEST BY MOUNTING RESOURCE TO WORKER NODES

**Note:** Repeat all below steps of this task on **both the worker nodes**

1. ssh to worker node : worker1 **[Switching to the Worker Node]**

**Note:** if you are Using putty open new session for Worker nodes.

```
$ ssh root@worker1ipaddress
```

2. Install nfs software on the worker nodes

```
$ sudo apt-get -y install nfs-common
```

```
ubuntu@worker1:~$  
ubuntu@worker1:~$ sudo apt-get -y install nfs-common  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following package was automatically installed and is no longer required:  
  grub-pc-bin  
Use 'sudo apt autoremove' to remove it.  
The following additional packages will be installed:  
  libnfsidmap2 libtirpc1 rpcbind
```

3. Test by mounting the /opt/sfw path to /mnt. Use **kubeadm-master server's public IP Address**

```
$ sudo mount 104.43.166.57:/opt/sfw /mnt
```

```
$ ls -l /mnt
```

```
ubuntu@worker1:~$ sudo mount 104.43.166.57:/opt/sfw /mnt  
ubuntu@worker1:~$ ls -l /mnt  
total 4  
-rw-r--r-- 1 root root 9 Jun  7 00:42 hello.txt  
-rw-r--r-- 1 root root 0 Jun  7 00:41 hello.txt'  
ubuntu@worker1:~$
```

**Note:** Exit back to the **Master node** after performing the above Task 2 steps on **both the worker nodes**

## 6 CREATING A PERSISTENT NFS VOLUME (PV)

**Note:** In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official Kubernetes documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes
```

```
$ cd Kubernetes
```

```
root@master:~# cd Kubernetes/
root@master:~/Kubernetes# ls
Dockerfile
README.md
_pycache
adapter-configmap.yaml
adapter-pod.yaml
app.py
apple.yaml
banana.yaml
config-map.yaml
configmap-pod.yaml
counter-pod.yaml
cron.yaml
daemonset.yaml
demo-pod.yaml
docker-compose.yaml
docker-registry-secret.yaml
dockerfile-mq
elasticsearch-rbac.yaml
elasticsearch-stfullset.yaml
elasticsearch-svc.yaml
elasticsearch.yaml
example-ingress.yaml
filebeat-agent.yaml
fluentd.yaml
root@master:~/Kubernetes#
foo-allow-to-hello.yaml
guestbook-frontend-svc.yaml
guestbook-frontend.yaml
headlesservice.yaml
hello-allow-from-foo.yaml
ingress-app1.yaml
ingress-app2.yaml
ingress-route.yaml
initcontainer.yaml
job-mq.yaml
job-tmpl.yaml
job.yaml
kibana-elk.yaml
kibana.yaml
label-deployment.yaml
liveness-pod.yaml
logstash-configmap.yaml
logstash-deployment.yaml
logstash-svc.yaml
metrics-server.yaml
multi-container.yaml
multi-pod-configmap.yaml
multi-pod-nginx.yaml
multi-prod-consumer.yaml
namespace.yaml
network-policy.yaml
nfs-pv.yaml
nfs-pvc.yaml
nfspv-pod.yaml
nginx-deployment.yaml
nginx-hpa.yaml
nginx-svc.yaml
nodeaffinity-deployment.yaml
nodeaffinity1-deployment.yaml
nodeanti-affinity-deployment.yaml
nodeanti-affinity1-deployment.yaml
oke-admin-service-account.yaml
pod-dynamicpv-oci.yaml
pod-dynamicpv.yaml
podaffinity-deployment.yaml
podaffinity1-deployment.yaml
podanti-affinity-deployment.yaml
podanti-affinity1-deployment.yaml
priv-reg-pod.yaml
pvc-oci.yaml
pvc.yaml
quota-pod.yaml
quota-pod1.yaml
quota.yaml
rabbitmq-deployment.yaml
rabbitmq-service.yaml
readiness-pod.yaml
readiness-svc.yaml
redis-cm.yaml
redis-deployment.yaml
redis-master-svc.yaml
redis-master.yaml
redis-pod.yaml
redis-slave-svc.yaml
redis-slave.yaml
requirements.txt
role-dev.yaml
rolebind.yaml
script.sh
security-cxt-nonroot.yaml
security-cxt-priv.yaml
security-cxt-readonly.yaml
security-cxt-rmcap.yaml
security-cxt-time.yaml
security-cxt.yaml
statefulset1.yaml
tt-pod.yaml
tt-pod1.yaml
web.yaml
worker.py
```

**Note:** Before performing this Section make sure you have performed above two Sections.

1. View the content of nfs-pv.yaml file. Provide nfs server details in the nfs section.

**Note:** In server details write the Master Server IP Address

```
$ vim nfs-pv.yaml
```



```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvvol-1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    path: /opt/sfw
    server: kubeadm-master
    readOnly: false
~
~
~
~
```

2. Create PV from the nfs-pv.yaml file created above and verify its created as expected. It should be in Available status to bind

```
$ kubectl create -f nfs-pv.yaml
```

```
$ kubectl get pv
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nfs-pv.yaml
persistentvolume/pvvol-1 created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM  STORAGECLASS  REASON  AGE
pvvol-1       1Gi       RWX           Retain          Available             Available  14s
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

## 7 CREATING A PERSISTENT VOLUME CLAIM (PVC)

1. Create PVC from the **nfs-pvc.yaml** file created above and verify its created as expected. It should bind to the PV created in Task 3

```
$ vi nfs-pvc.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Mi
~
~
~
~
```

```
$ kubectl create -f nfs-pvc.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nfs-pvc.yaml
persistentvolumeclaim/nfs-claim created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Verify the status of PVC and PV created, see that it gets bound as expected

```
$ kubectl get pvc
```

```
$ kubectl get pv
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pvc
NAME          STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nfs-claim     Bound     pvvol-1   1Gi        RWX             default/nfs-claim 27s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM          STORAGECLASS   REASON   AGE
pvvol-1       1Gi        RWX             Retain            Bound    default/nfs-claim 28m
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

## 8 CREATING POD TO MOUNT NFS VOLUME

1. View the content of nfspv-pod.yaml file and check the volumeMount and volumes section

```
apiVersion: v1
kind: Pod
metadata:
  name: www
  labels:
    name: www
spec:
  containers:
  - name: www
    image: nginx:alpine
    ports:
      - containerPort: 80
        name: www
    volumeMounts:
      - name: nfs-vol
        mountPath: /usr/share/nginx/html
  volumes:
  - name: nfs-vol
    persistentVolumeClaim:
      claimName: nfs-claim
~
~
~
~
```

2. Create Pod resource from nfspv-pod.yaml file

```
$ kubectl create -f nfspv-pod.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# vim nfspv-pod.yaml
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nfspv-pod.yaml
pod/www created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Verify the status of the www pod and describe it to see the volumeMount and volume details

```
$ kubectl get pods -o wide
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
www      1/1     Running   0           19s   10.40.0.1    worker2   <none>             <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl describe pod www
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe pod www
Name:          www
Namespace:     default
Priority:       0
Node:          worker2/10.0.0.6
Start Time:    Sun, 07 Jun 2020 01:28:41 +0000
Labels:        name=www
Annotations:    <none>
Status:        Running
IP:            10.40.0.1
IPs:
  IP: 10.40.0.1
  Environment: <none>
  Mounts:
    /usr/share/nginx/html from nfs-vol (rw)
    /var/run/secrets/kubernetes.io/serviceaccount from default-token-t9q44 (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready            True
  ContainersReady  True
  PodScheduled     True
Volumes:
  nfs-vol:
    Type:          PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:     nfs-claim
    ReadOnly:      false
```

4. Verify the status of pod, PV and PVC created in this lab exercise and check the relation

```
$ kubectl get pods
```

```
$ kubectl get pv
```

```
$ kubectl get pvc
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pod
NAME    READY   STATUS    RESTARTS   AGE
www     1/1     Running   0           8m26s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                STORAGECLASS   REASON   AGE
pvvol-1  1Gi        RWX            Retain           Bound   default/nfs-claim    default        44m
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pvc
NAME    STATUS   VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nfs-claim  Bound   pvvol-1   1Gi        RWX            default        17m
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

## 9 DELETING RESOURCES & VERIFY VOLUME BEHAVIOUR

1. Delete the pod and verify the status of PV and PVC. Even after the Pod is deleted PV and PVC will be still in bound status

```
$ kubectl delete -f nfspv-pod.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f nfspv-pod.yaml
pod "www" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl get pods
```

```
$ kubectl get pv
```

```
$ kubectl get pvc
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pod
No resources found in default namespace.
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS  REASON  AGE
pvvol-1       1Gi       RWX           Retain          Bound   default/nfs-claim  45m
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pvc
NAME          STATUS  VOLUME  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
nfs-claim     Bound   pvvol-1  1Gi       RWX           17m
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Delete the PVC and verify the status of PV. After the PVC is deleted PV status is marked as Released

```
$ kubectl delete -f nfs-pvc.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f nfs-pvc.yaml
persistentvolumeclaim "nfs-claim" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl get pv
```

```
$ kubectl get pvc
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pvc
No resources found in default namespace.
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS  REASON  AGE
pvvol-1       1Gi       RWX           Retain          Released  default/nfs-claim  45m
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Delete the PV and verify all resources created in this lab exercise is cleaned-up

```
$ kubectl delete -f nfs-pv.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f nfs-pv.yaml
persistentvolume "pvvol-1" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pv
No resources found in default namespace.
root@kubeadm-master:/home/ubuntu/Kubernetes#
```



## 10 TROUBLESHOOTING

### 10.1 Access denied when mounting NFS server

**Issue:** Access denied when mounting NFS server on master.

```
xxxxxxx@worker2:/home/nckatrain# sudo mount 20.50.199.53:/opt/sfw /mnt
mount.nfs: Connection timed out
```

After security group rule addition for port 2049 on both worker as well as master node, error changed:

```
xxxxxxx@worker2:~$ sudo mount 20.50.199.53:/opt/sfw /mnt
mount.nfs: access denied by server while mounting 20.50.199.53:/opt/sfw
```

```
xxxxxxx@worker1:/home/nckatrain# sudo mount 20.50.199.53:/opt/sfw /mnt
mount.nfs: access denied by server while mounting 20.50.199.53:/opt/sfw
```

**Reason:** Not enough permissions to Workers to mount NFS on Master because while giving permission, it's becoming comment.

```
root@masternode: /etc
# cat /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv3 and NFSv4:
# /srv/homeas    hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homeas gss/krb5i(rw,sync,no_subtree_check)
# /opt/sfw/ * (rw,sync,no_root_squash,subtree_check,insecure)
```

```
root@masternode:/home/nckatrain# cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#         to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4       gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
# /opt/sfw/ *(rw,sync,no_root_squash,subtree_check,insecure)
```

**Fix:** Make sure that in **/etc/export** file there is no '#' in the starting of the below line. Otherwise it will become a comment.

```
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)
```

3. Edit the NFS server file to share out the newly created directory. In this case we will share the directory with all. Vim the file and add the below line

```
$ sudo vim /etc/exports
```

```
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)
```

```
# /etc/exports: the access control list for filesystems which may be exported
#         to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4       gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)
~
~
~
```



---

## 11 SUMMARY

In this guide we Covered:

- Configuring NFS storage persistence volume
- Test by mounting resource to worker nodes
- Creating a persistent NFS volume (PV)
- Creating a persistent volume claim (PVC)
- Creating pod to mount NFS volume