# Installing and configuring NFS Server on Ubuntu

* The first step is to install the **nfs-kernel-server** package on the server.
* Before we do this, let’s first update the system packages using the following [apt command](https://www.tecmint.com/apt-advanced-package-command-examples-in-ubuntu/).
* $ sudo systemctl status nfs-kernel-server

A screenshot of a computer

Description automatically generated

# Create an NFS Export Directory

* After the installation of nfs server we are creating a directory that will be shared among client systems.
* Referred to as the export directory and it’s in this directory that we shall later create files that will be accessible by client systems.
* Here’s we have given the read, write and execute privileges to all the contents inside the directory.

A screenshot of a computer

Description automatically generated

# Export the NFS Share Directory

* After granting access to the preferred client systems, export the NFS share directory and restart the NFS kernel server for the changes to come into effect.
* you need to allow access through the firewall otherwise, accessing and mounting the shared directory will be impossible.

A screenshot of a computer

Description automatically generated

# Install the NFS Client on the Client Systems

* We’re done installing and configuring the NFS service on the Server, let’s now install NFS on the client system.
* Next, you need to create a mount point on which you will mount the nfs share from the NFS server.
* Remaining is mounting the NFS share that is shared by the NFS server. This will enable the client system to access the shared directory.

A screenshot of a computer

Description automatically generated

# Creation of NFS storage class

* A storage class is a Kubernetes object that stores information about creating a persistent volume for your pod.
* With a storage class, you do not need to create a persistent volume separately before claiming it.

Text

Description automatically generated

# Deploy the NFS Persistent Volume

* PV is an abstraction for the physical storage device that attached to the cluster. PV is used to manage durable storage which needs actual physical storage.
* PV is independent of the lifecycle of the Pods. It means that data represented by a PV continue to exist as the cluster changes and as Pods are deleted and recreated.

Access modes: This property defines how a PV should be mounted on the host.

* ReadWriteOnce: You can mount the PV as read-write by a single node.
* ReadOnlyMany: You can mount the PV as read-only by multiple nodes.
* ReadWriteMany: You can mount the PV as read-write by multiple nodes.

A screenshot of a computer

Description automatically generated

# Deploy the NFS Persistent Volume Claim

* PVC is binding between a Pod and PV. Pod request the Volume through the PVC.
* PVC is the request to provision persistent storage with a specific type and configuration.
* PVC is similar to a Pod. Pods consume node resources and PVC consume PV resources.
* PVC must be in same namespace as the Pod. For each Pod, a PVC makes a storage consumption request within a namespace.

A screenshot of a computer

Description automatically generated

# Deploy and Create the POD

* A Pod can access storage with the help of a PVC, which will be used as a volume.
* PVC can be used in a Pod by first declaring a “volumes” property in the Pod manifest file and specifying the claim name under the declared volume type “persistentVolumeClaim” property.
* It is essential that both the PVC and the Pod using it exist in the same namespace.

A screenshot of a computer

Description automatically generated

# Login to POD

* Edit files in mounting point and ensure it reflects inside of pod

A screenshot of a computer

Description automatically generated