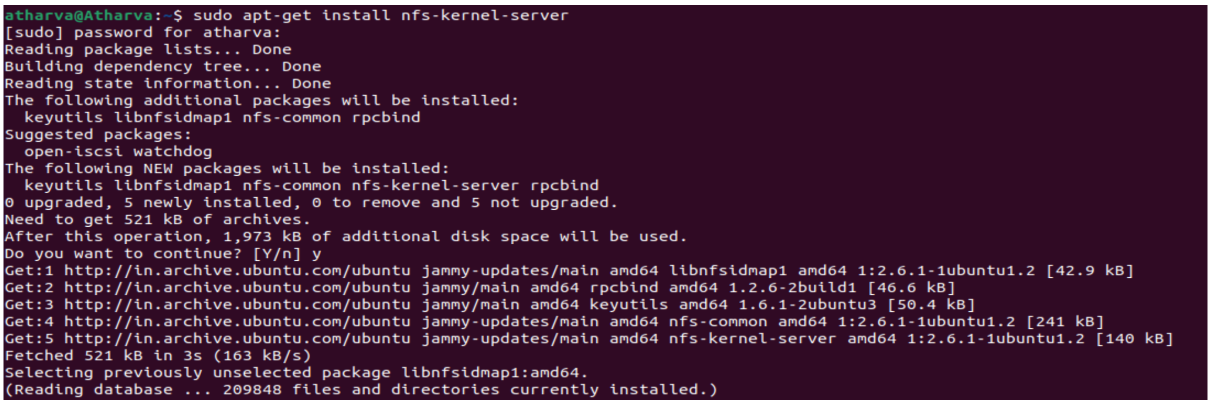
PRACTICAL 2

Aim: Configure NFS server to share directories on your network.

Step 1: Install the NFS server on your Ubuntu 16.04 using the following command

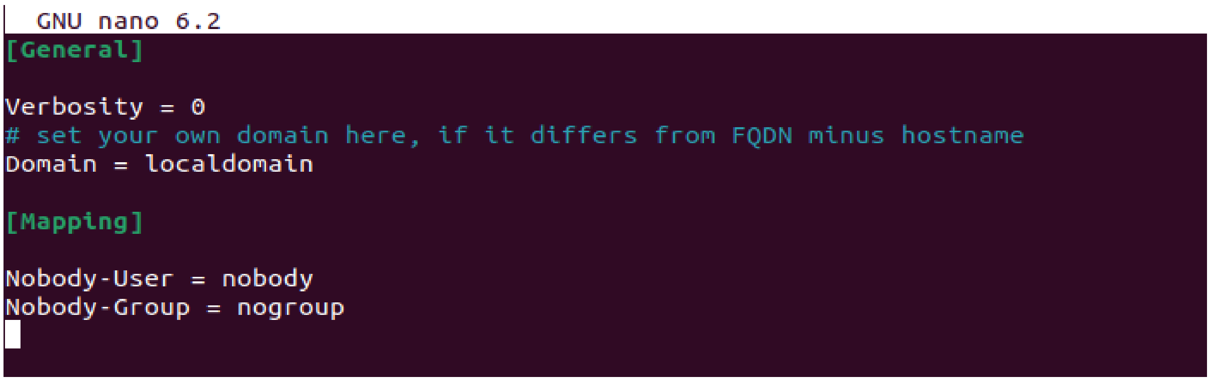
sudo apt-get install nfs-kernel-server



```
atharva@Atharva:~$ sudo apt-get install nfs-kernel-server
[sudo] password for atharva:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils libnfsidmap1 nfs-common rpcbind
Suggested packages:
  open-iscsi watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap1 nfs-common nfs-kernel-server rpcbind
0 upgraded, 5 newly installed, 0 to remove and 5 not upgraded.
Need to get 521 kB of archives.
After this operation, 1,973 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libnfsidmap1 amd64 1:2.6.1-1ubuntu1.2 [42.9 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 rpcbind amd64 1.2.6-2build1 [46.6 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 keyutils amd64 1.6.1-2ubuntu3 [50.4 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nfs-common amd64 1:2.6.1-1ubuntu1.2 [241 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nfs-kernel-server amd64 1:2.6.1-1ubuntu1.2 [140 kB]
Fetched 521 kB in 3s (163 kB/s)
Selecting previously unselected package libnfsidmap1:amd64.
(Reading database ... 209848 files and directories currently installed.)
```

Step 2: Once the installation is done we have to change the domain name of our domain which will be as same as of our root machine

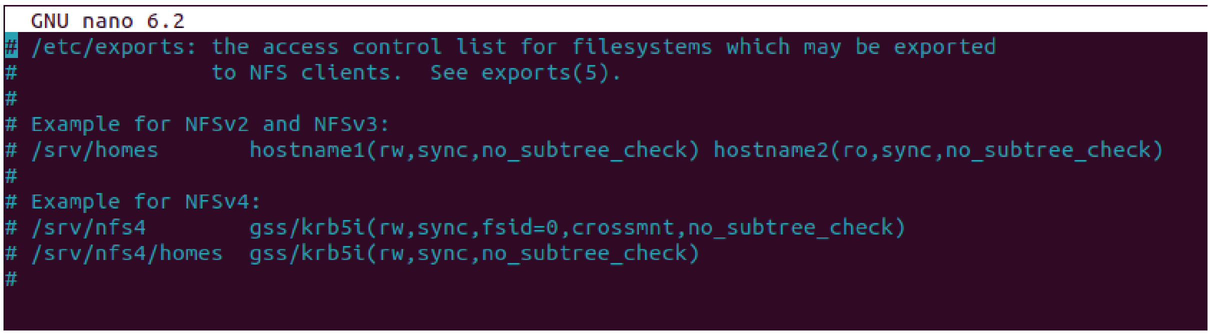
sudo nano /etc/idmapd.conf



```
  GNU nano 6.2
[General]

Verbosity = 0
# set your own domain here, if it differs from FQDN minus hostname
Domain = localdomain

[Mapping]

Nobody-User = nobody
Nobody-Group = nogroup
```

Step 3: Now open the exports file to check if it's there or not. It is basically a list for file systems which might be exported via our NFS server

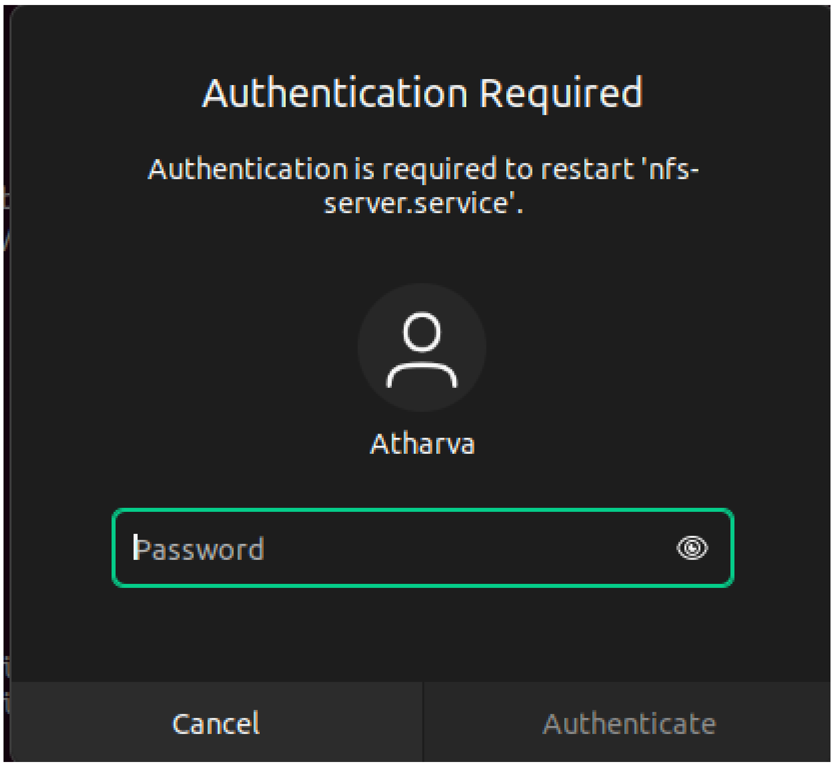
sudo nano /etc/exports



```
  GNU nano 6.2
# /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)
#
```

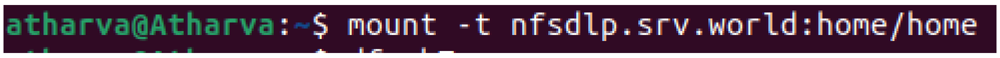
<u>Step 4:</u> Now restart the NFS server you've installed in your Ubuntu at the start of the practical using the following command

systemctl restart nfs-server



<u>Step 5:</u> In this step we will mount our nfs server into our home directory using the following command

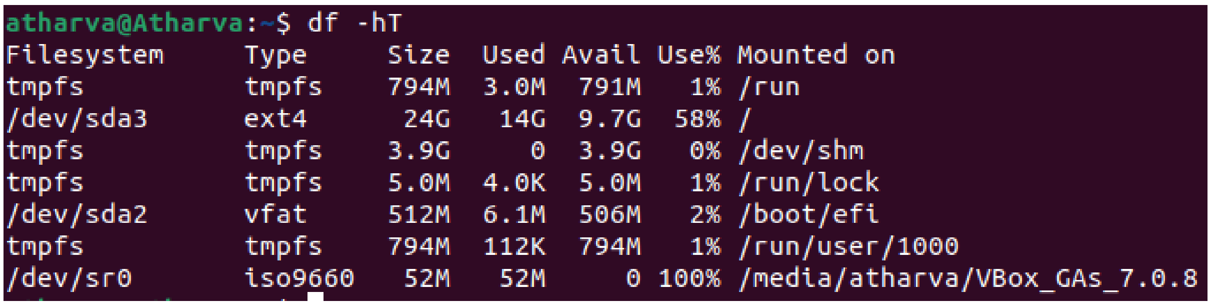
mount -t nfsdlp.srv.world:/home/home



```
atharva@Atharva:~$ mount -t nfsdlp.srv.world:home/home
```

<u>Step 6:</u> In this step we will display the information mounted by our NFS server, the 'df' command is used to display the information about total space and available space on a file system.

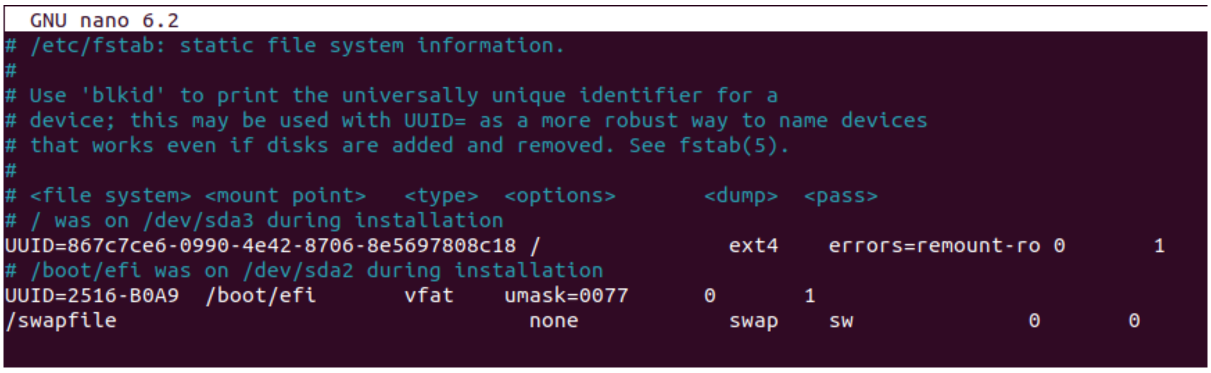
df -hT



```
atharva@Atharva:~$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
tmpfs          tmpfs     794M  3.0M  791M   1% /run
/dev/sda3      ext4       24G   14G  9.7G  58% /
tmpfs          tmpfs     3.9G     0  3.9G   0% /dev/shm
tmpfs          tmpfs     5.0M  4.0K  5.0M   1% /run/lock
/dev/sda2      vfat      512M  6.1M  506M   2% /boot/efi
tmpfs          tmpfs     794M  112K  794M   1% /run/user/1000
/dev/sr0       iso9660    52M   52M     0 100% /media/atharva/VBox_GAs_7.0.8
```

Step 7: In this step we will configure the mounting on fstab to mount it when the system boot

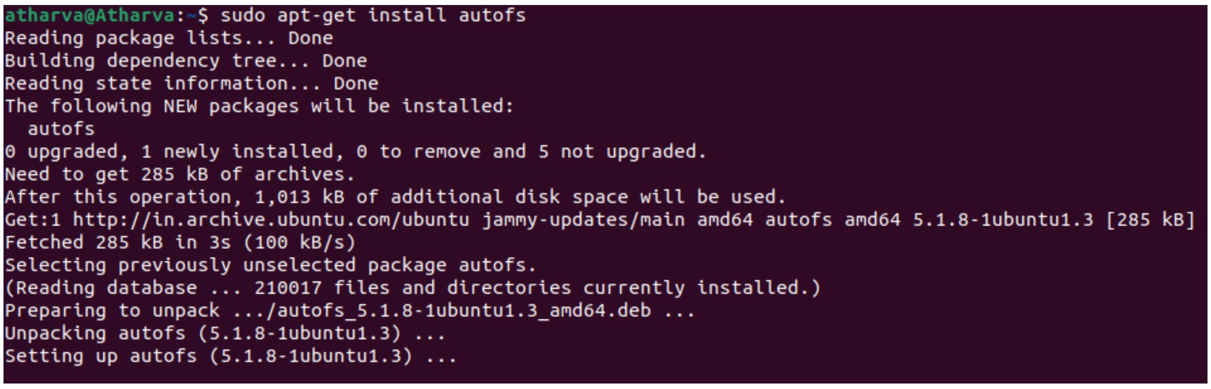
sudo nano /etc/fstab



```
  GNU nano 6.2
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda3 during installation
UUID=867c7ce6-0990-4e42-8706-8e5697808c18 /               ext4    errors=remount-ro 0       1
# /boot/efi was on /dev/sda2 during installation
UUID=2516-B0A9  /boot/efi       vfat    umask=0077      0       1
/swapfile                                 none            swap    sw              0       0
```

Step 8: Now we will install our auto mounting fstab using the following command

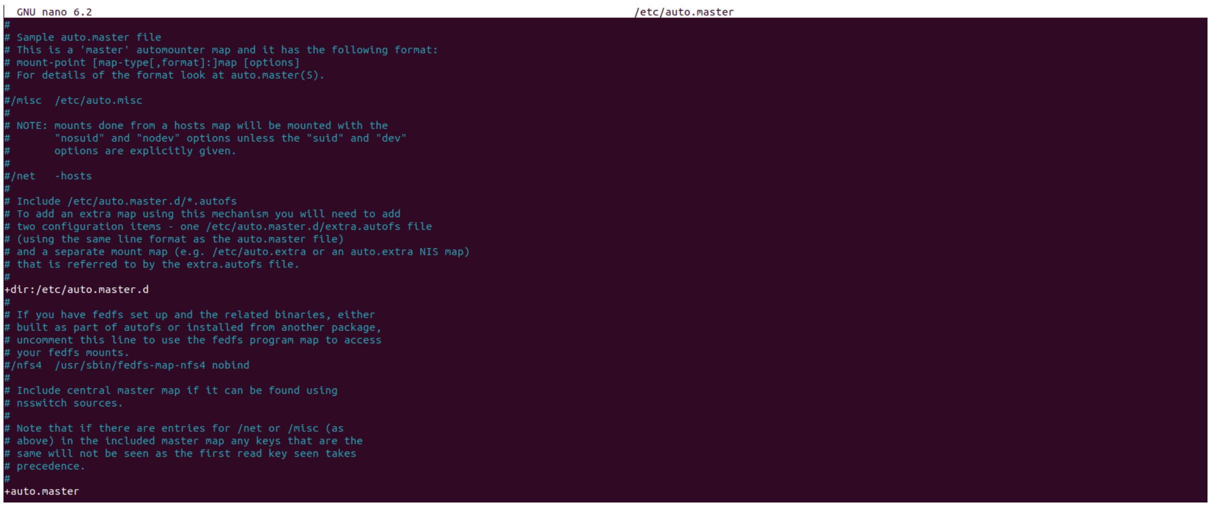
sudo apt-get install autofs



```
atharva@Atharva:~$ sudo apt-get install autofs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  autofs
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 285 kB of archives.
After this operation, 1,013 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 autofs amd64 5.1.8-1ubuntu1.3 [285 kB]
Fetched 285 kB in 3s (100 kB/s)
Selecting previously unselected package autofs.
(Reading database ... 210017 files and directories currently installed.)
Preparing to unpack .../autofs_5.1.8-1ubuntu1.3_amd64.deb ...
Unpacking autofs (5.1.8-1ubuntu1.3) ...
Setting up autofs (5.1.8-1ubuntu1.3) ...
```

Step 9: After getting done with the installation we have to open and check whether the auto.master file was created or not. This will be done by using the following command

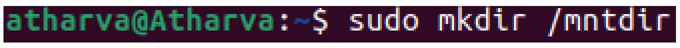
sudo nano /etc/auto.master



```
  GNU nano 6.2                                              /etc/auto.master
#
# Sample auto.master file
# This is a 'master' automounter map and it has the following format:
# mount-point [map-type[,format]:]map [options]
# For details of the format look at auto.master(5).
#
#/misc  /etc/auto.misc
#
# NOTE: mounts done from a hosts map will be mounted with the
#       "nosuid" and "nodev" options unless the "suid" and "dev"
#       options are explicitly given.
#
#/net   -hosts
#
# Include /etc/auto.master.d/*.autofs
# To add an extra map using this mechanism you will need to add
# two configuration items - one /etc/auto.master.d/extra.autofs file
# (using the same line format as the auto.master file)
# and a separate mount map (e.g. /etc/auto.extra or an auto.extra NIS map)
# that is referred to by the extra.autofs file.
#
+dir:/etc/auto.master.d
#
# If you have fedfs set up and the related binaries, either
# built as part of autofs or installed from another package,
# uncomment this line to use the fedfs program map to access
# your fedfs mounts.
#/nfs4  /usr/sbin/fedfs-map-nfs4 nobind
#
# Include central master map if it can be found using
# nsswitch sources.
#
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
```

Step 10: After checking the auto.master file we will need to create a directory(file) in which we will be adding our mounted data.

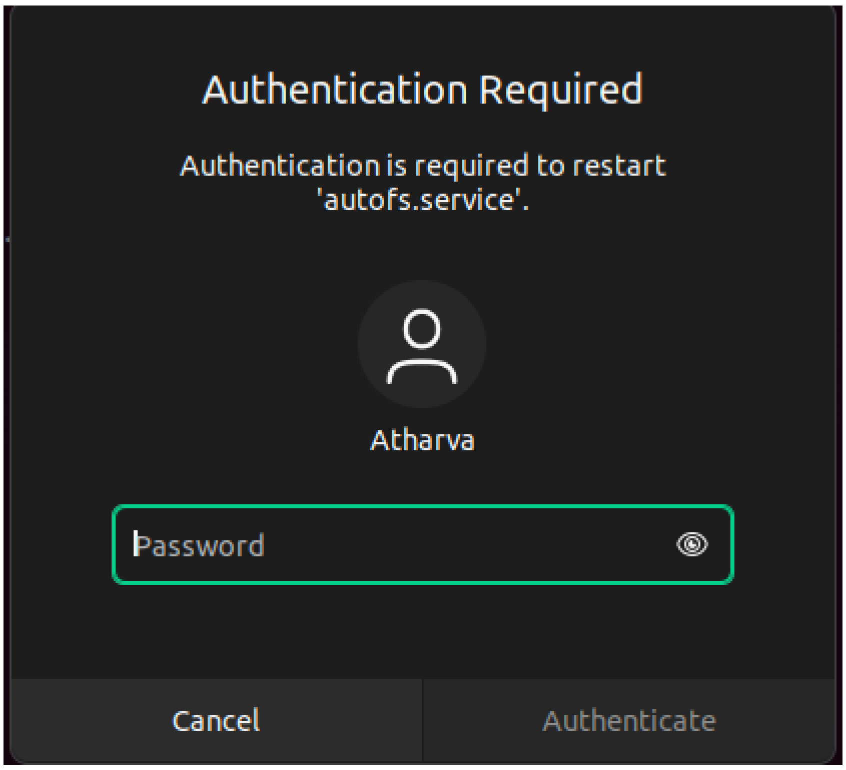
sudo mkdir /mntdir



```
atharva@Atharva:~$ sudo mkdir /mntdir
```

Step 11: Now after making the directory we will restart the auto mounting fstab, using the following command.

systemctl restart autofs



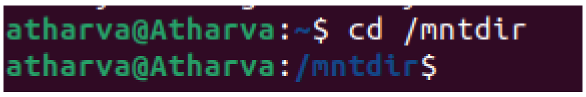
Authentication Required

Authentication is required to restart 'autofs.service'.

Atharva

Step 12: Now we have to mount the data, before doing that we will need to change the current directory we are in to the directory we made for data mounting purpose (mntdir). For changing directory we will use "cd"

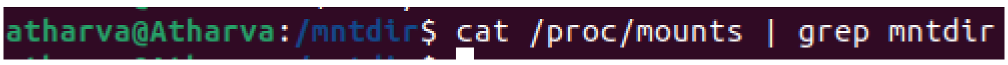
cd /mntdir



```
atharva@Atharva:~$ cd /mntdir
atharva@Atharva:/mntdir$
```

Step 13: We have to mount the data into mntdir which was mounted when we installes NFS and auto fstab in the mount directory, we will use grep command. It can be done as follow:

cat /proc/mounts | grep mntdir



```
atharva@Atharva:/mntdir$ cat /proc/mounts | grep mntdir
```