

# Activity-1

## A) Configure and deploy below application on linux server (3 tier application)

Frontend server( apache web server) -> backend application (Node app) - database (mongo)

Create .env file in you project to pass environment =>

MONGODB\_URL=mongodb://localhost:27017/demo

PORT=3000

Application source - <https://github.com/BL-AniketChile/NodeJs-API>

Ans.

First we need to update and upgrade the apt to the latest version so that we can install the packages

```
ubuntu@ip-172-31-12-62:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [991 kB]

ubuntu@ip-172-31-12-62:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  apport apport-core-dump-handler cloud-init landscape-common libnss-systemd libpam-systemd
  libudev1 linux-base pci.ids plymouth plymouth-theme-ubuntu-text python3-apport python3-pro
  rsyslog snapd software-properties-common sosreport systemd systemd-dev systemd-resolved sy
27 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 41.6 MB of archives.
After this operation, 579 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

We need to install prerequisites:

## Apache HTTP Server

To install the apache server we need to use the command `sudo apt install apache2`

Then we need to start the apache2 service and enable it to work it on this system

```
ubuntu@ip-172-31-12-62:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ip-172-31-12-62:~$ sudo systemctl start apache2
ubuntu@ip-172-31-12-62:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
```

## Node.js and npm

First we have to install the NodeSource APT Repository

```
ubuntu@ip-172-31-12-62:~$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
2025-04-11 04:52:20 - Installing pre-requisites
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
```

Here we use curl command to download the script from NodeSource that sets up the Node.js v18 repository.

The sudo command pipes that script directly into bash to run it as root.

-E preserves your environment variables (like proxy settings).

The script:

Adds the NodeSource repo to /etc/apt/sources.list.d/.

Adds the GPG key used to verify Node.js packages.

Runs apt update.

Now we can safely install nodejs to the system using apt

```
ubuntu@ip-172-31-12-62:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 29.7 MB of archives.
```

We also need to install npm(Node packet manager) to the system

To verify that the services have installed we can use the commands

```
ubuntu@ip-172-31-12-62:~$ node -v
v18.20.8
ubuntu@ip-172-31-12-62:~$ npm -v
10.8.2
```

We need to install process manager for node.js(Pm2)

```
ubuntu@ip-172-31-12-62:~$ sudo npm install -g pm2
added 134 packages in 8s

13 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases
npm notice To update run: npm install -g npm@11.3.0
npm notice
ubuntu@ip-172-31-12-62:~$ sudo apt install mongodb
Reading package lists... Done
Building dependency tree... Done
```

## Git

We use the command `sudo apt install git` to install git in the server

```
ubuntu@ip-172-31-12-62:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

## MongoDB

To install mongodb we need to first import a public key with the following command:

```
curl -fsSL https://www.mongodb.org/static/pgp/server-8.0.asc | \sudo gpg --dearmor -o /usr/share/keyrings/mongodb-server-8.0.gpg
```

Here curl is downloading MongoDB's GPG key file

This key is used to verify the authenticity of the packages during installation

Flags used are

- f: Fails silently on errors
- s: No progress bar
- S: Show errors
- L: Follow Redirects

Here we have used a pipeline ( | ) to convert the ASCII key to binary format

gpg = GNU Privacy Guard (used for encryption/signing).

--dearmor = converts the ASCII-armored key to binary format

-o /usr/share/keyrings/mongodb-server-8.0.gpg = Save the binary version to that file.

We use sudo because /usr/share/keyrings requires root privileges

```
ubuntu@ip-172-31-12-62:~$ curl -fsSL https://www.mongodb.org/static/pgp/server-8.0.asc | \sudo gpg --dearmor -o /usr/share/keyrings/mongodb-server-8.0.gpg
```

This is done because:

When you add MongoDB's repository to your system (/etc/apt/sources.list.d/), Ubuntu needs to **verify the authenticity of packages**.

This .gpg file lets APT trust the packages from that repository, preventing man-in-the-middle attacks or installing tampered packages.

```
ubuntu@ip-172-31-12-62:~$ echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-8.0.list
deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse
```

Now we will create a list file for our version of ubuntu

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg ] https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-8.0.list
```

Here

**deb** = Tells apt this is a Debian-style package repo.

**[ arch=amd64,arm64 ... ]** = Only use this repo for 64-bit systems (x86\_64 and ARM).

**signed-by=/usr/share/keyrings/mongodb-server-8.0.gpg** = Trust only if packages are signed by this key (added earlier with `gpg --dearmor`).

<https://repo.mongodb.org/apt/ubuntu> = MongoDB's official APT repo.

**noble/mongodb-org/8.0** = Distribution version (noble for Ubuntu 24.04), and MongoDB version 8.0.  
multiverse — Section of the repo where this package is found.

Pipes (|) the output of the echo command into:

**sudo tee** — A safe way to **write with elevated privileges** (since `echo > file` doesn't work with `sudo` alone).

**/etc/apt/sources.list.d/mongodb-org-8.0.list** — The file where Ubuntu stores external repository sources.  
So, the `sudo` command **adds the MongoDB APT repository** to your system securely, using the trusted key from the earlier step.

Here we have set the server to know:

Where to find MongoDB 8.0 packages (the URL).

What key to trust them with (the .gpg file).

What architectures to accept (your CPU type).

We need to update the apt again to install the services to the system

Now we can safely install the mongodb service using `sudo apt install mongodb-org`

We can verify the services of mongodb is working or not

Now we have to clone the github repository to work on in on our system to serve as the backend

It will require sudo privileges to clone it to the folder

Now we need to install the dotenv package which loads the environment variables from .env files to process.env making it easy to manage and access configuration settings, especially sensitive ones like API keys and database credentials, without hardcoding them into your application.

```
ubuntu@ip-172-31-12-62:~/var/www/NodeJs-API$ npm install dotenv
npm warn old lockfile
npm warn old lockfile The package-lock.json file was created with an old version of npm,
npm warn old lockfile so supplemental metadata must be fetched from the registry.
npm warn old lockfile
npm warn old lockfile This is a one-time fix-up, please be patient...
npm warn old lockfile
npm error code EACCES
```

We will need to edit the `.js` file to require the `dotenv` file to convert the environment variable from `.env` files to `process.env` files.

```
require('dotenv').config();
```

We will create the environment variables in the .env file where we will connect the MongoDB Database to the application

## Database(MongoDB)

We will enable and start the mongodb service

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ sudo systemctl start mongod
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ sudo systemctl enable mongod
```

This will auto-create the demo database but we are not using the demo database

We are using our own self-created database NodeApp

So we open the mongodb shell using the **mongosh** command and give the command **use NodeApp**

This will create the database. We can put in a sample data using the command

```
db.test.insertOne({ message: "Hello from NodeAPP database!" })
```

And check it by:

## db.test.find()

Then we can just exit the database

Now we would also have to edit our .env file to reflect the change in database. So we would edit the URL to change the database in the URL to NodeApp.

## Backend (Node.js)

Now we will start the app using the pm2 service

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ pm2 start index.js --name node-api
```

[illegible]

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ pm2 start app.js --name node-api
[PM2] Starting /var/www/NodeJs-API/app.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	namespace	version	mode	pid	uptime	U	status	cpu	mem	user	watching
0	node-api	default	1.0.0	fork	17513	0s	0	online	0%	31.6mb	ubuntu	disabled

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ pm2 restart node-api --update-env
[PM2] Applying action restartProcessId on app [node-api] (ids: [ 0 ])
[PM2] [node-api] (0) ✓
```

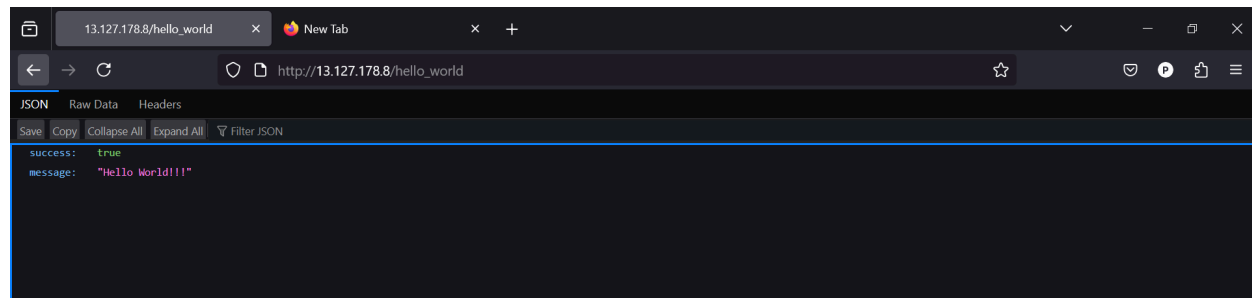
id	name	namespace	version	mode	pid	uptime	U	status	cpu	mem	user	watching
0	node-api	default	1.0.0	fork	17536	0s	1	online	0%	21.5mb	ubuntu	disabled

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ pm2 save
[PM2] Saving current process list...
[PM2] [WARN] PM2 is not managing any process, skipping save...
[PM2] [WARN] To force saving use: pm2 save --force
```

We have saved the pm2 setting. Now we can start the application by giving the command node server.js

```
ubuntu@ip-172-31-12-62:/var/www/NodeJs-API$ node server.js
(node:17715) Warning: Accessing non-existent property 'count' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:17715) Warning: Accessing non-existent property 'findOne' of module exports inside circular dependency
(node:17715) Warning: Accessing non-existent property 'remove' of module exports inside circular dependency
(node:17715) Warning: Accessing non-existent property 'updateOne' of module exports inside circular dependency
Server is running on port: 3000
MongoDB connection established successfully!!!
```

Verify the backend is working or not by putting the `http://<ip address>/hello_world` in a browser



```
13.127.178.8/hello_world x New Tab x +
http://13.127.178.8/hello_world
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
{
  "success": true,
  "message": "Hello World!!!"
}
```

## Frontend (Apache Server)

Now we can start working on the frontend of the application by using an apache server

Here we enable the proxy and proxy\_http service of apache and to save it we reload the apache service

```
ubuntu@ip-172-31-12-62:~$ sudo a2enmod proxy
Enabling module proxy.
To activate the new configuration, you need to run:
    systemctl restart apache2
ubuntu@ip-172-31-12-62:~$ sudo a2enmod proxy_http
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
To activate the new configuration, you need to run:
    systemctl restart apache2
ubuntu@ip-172-31-12-62:~$ systemctl restart apache2
```



We need to create a apache server configuration file for our application where we tell the server where to redirect the signals to

This is done in the sites-available folder in the apache directory.

```
GNU nano 7.2 /etc/apache2/sites-available/nodeapp.conf
<VirtualHost *:80>
    ServerName 13.127.178.8

    ProxyPreserveHost On
    ProxyPass / http://localhost:3000/
    ProxyPassReverse / http://localhost:3000/

    ErrorLog ${APACHE_LOG_DIR}/nodeapp_error.log
    CustomLog ${APACHE_LOG_DIR}/nodeapp_access.log combined
</VirtualHost>
```

Here we have told the server that whichever signal lands on port 80 will pass to localhost:3000

Then it will redirect the signal to the backend where the processes are done which is then sent back to apache server who will then send the data to the user device.

We will also have to add the API in the server.js file to redirect the URL from root to swagger/api as the application is using swagger api

```
require('dotenv').config();
var express = require('express'),
    app = express(),
    bodyparser = require('body-parser'),
    connectdb = require('./config/database/mongodb').connect,
    swagger_ui = require('swagger-ui-express'),
    swagger_doc = require('./app/lib/swagger-ui/api_docs')

require('./app/routes/routes')(app);

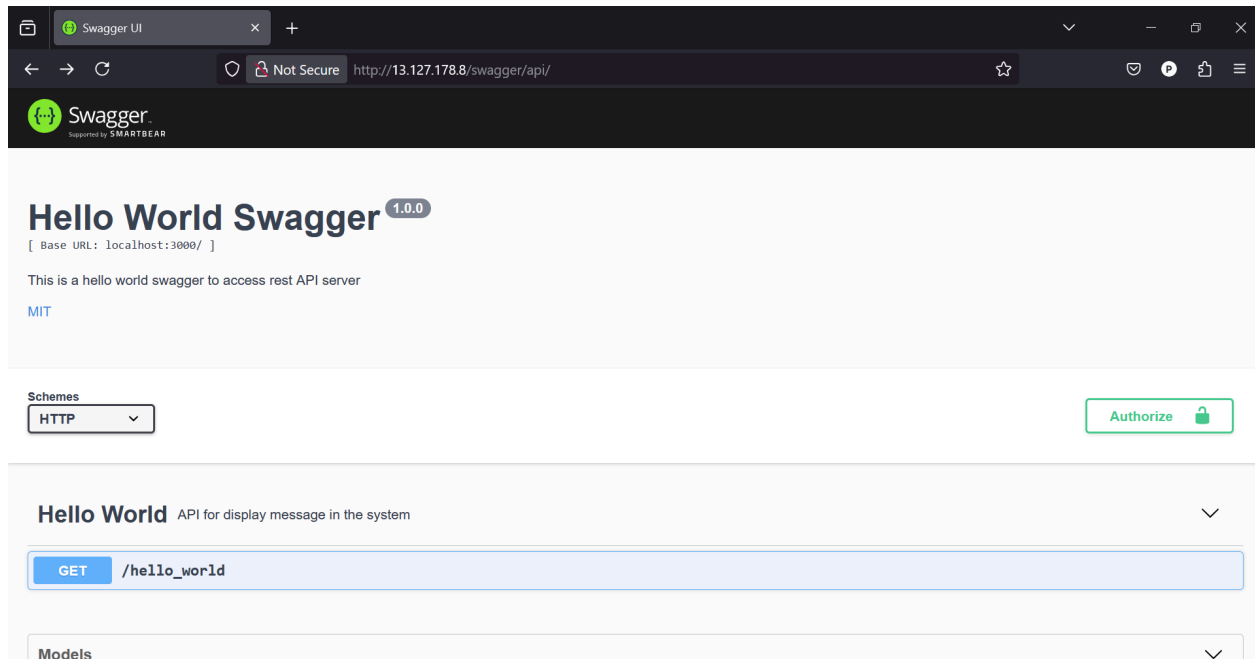
app.use(bodyparser.json());

app.use('/swagger/api', swagger_ui.serve, swagger_ui.setup(swagger_doc));
app.get('/', (req, res) => {
    res.redirect('/swagger/api');
});
app.listen(process.env.PORT, () => {
```

Now when we start the application again using the command **node server.js** and go to a browser and type the IP address of the instance it will instantly redirect to the url of <http://<IP Address>/swagger/api>

To give this output





To ensure that the server is always running in background and can auto-start after reboot or system startup, we use the pm2 service

PM2 is a production-grade process manager for Node.js that runs apps in the background, monitors them, restarts them on failure, and even on reboot.

We can start our application with the following command

**pm2 start server.js --name node-api**

This command run the server.js in the background. This frees our terminal so that we can give other commands in the same terminal

We can check the status using this command:

**pm2 status**

To ensure that the application run on system startup and reboot, we give the following command

**pm2 startup**

To save this configuration we need to give **pm2 save** command.

This ensures that the application starts on system reboot.

## **THEORY**

### **What is Proxy Server?**

A proxy server is an intermediate server that sits between a client and the internet. It forwards client requests to the destination server and then returns the server's response back to the client.

### **What is Reverse Proxy Server?**

A reverse proxy server acts as an intermediary between web clients (like browsers) and web servers. It intercepts client requests, forwards them to the appropriate server, and then returns the response back to the client. This provides a layer of abstraction, enabling features like load balancing, security enhancements, and performance improvements

## **What is GNU Privacy Guard?**

GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications.

## **Working**

### **Bottom to Top Approach**

#### **MongoDB Database**

It stores all your app's persistent data (users, posts, products, etc.) and runs as a service on your Linux server (port 27017 by default)

#### **Node.js Application**

It has the application source code which runs the logic. It is usually run on port 3000 and exposes the REST API or the Swagger docs

#### **Apache Server**

It receives all public traffic (on port 80/443) and also acts as a reverse proxy to your Node.js app.

It can also serve static content (like HTML, JS, CSS if needed).

It listens on port 80 and when a user opens the website/, Apache forwards the request to localhost:3000 (where your Node app is running). The response from Node is sent back through Apache to the browser.

#### **Request Flow**

1. User sends a request to your domain:  
e.g., `http://yourdomain.com/swagger/api`
2. Apache receives it on port 80, and proxies it to Node: → `http://localhost:3000/swagger/api`
3. Node.js app receives it, processes the route, and if needed: → Connects to MongoDB using `mongoose.connect(process.env.MONGODB_URL)`
4. MongoDB responds with data (e.g., list of users)
5. Node.js builds a response, sends it back to Apache
6. Apache forwards the final response to the user's browser

#### **pm2(Process Manager for Node.js)**

It keeps your app alive even after a crash or reboot and manages logs, auto-restarts, and environment variables.

## **Bottom to Top Approach**

Client Side(Top Layer)

### **User's Browser**

User opens a browser and visits the website (<http://<your-ip>>)

The browser sends a HTTP request (e.g., GET /swagger/api, or POST /user/register).

### **Apache Web Server(Reverse Proxy)**

Apache is listening on port 80. Based on the configuration (in /etc/apache2/sites-enabled/), it sees:  
apache

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

Apache forwards the request to your Node.js application running on port 3000.

Example:

Request: <http://yourdomain.com/user/register>

Apache forwards it to: <http://localhost:3000/user/register>

### **Node.js Application(Middle Layer)**

The server.js script is your app entry point.

It sets up:

- Express server
- Body-parser to parse incoming JSON
- Swagger for API documentation (available at /swagger/api)
- Routes from: ./app/routes/routes
- MongoDB database connection from: ./config/database/mongodb

### **MongoDB Database(Bottom Layer)**

The Node.js app connects to MongoDB using Mongoose or native driver.

Database runs on port 27017.

Based on your .env, it connects to:

`mongodb://localhost:27017/NodeAPP`

## **Why did we need the proxy server?**

We do not want to show the source code to the public so we use a proxy which acts as a intermediary and forwards all the requests that come to port 80 to the backend which processes the data and send it back to the proxy which sends the processes data back to the client

## B) Install and configure NFS server

i) Install NFS server component package on server X and NFS client component on client machine Y

First we need to install nfs kernel in server and nfs in common

### In server:

We need to give the command **sudo apt update** which will update all the packages to their latest version.

```
ubuntu@ip-172-31-12-62:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://deb.nodesource.com/node_18.x nodistro InRelease
Hit:5 https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [991 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1052 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Fetched 2295 kB in 2s (1445 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

Then we need to give the command **sudo apt install nfs-kernel-server**

```
ubuntu@ip-172-31-12-62:~$ sudo apt install nfs-kernel-server
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:
  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 0 not upgraded.
Need to get 569 kB of archives.
After this operation, 2022 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/mai
```

We have successfully installed NFS Server on Server X

### In Client:

We have to give the same command of **sudo apt update** to update the packages to their latest version.

```
ubuntu@ip-172-31-7-73:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
```

Now we need to install the NFS client in the client by giving the command **sudo apt install nfs-common**

```

ubuntu@ip-172-31-7-73:~$ sudo apt install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils libnfsidmap1 rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap1 nfs-common rpcbind
0 upgraded, 4 newly installed, 0 to remove and 59 not upgraded.
Need to get 400 kB of archives.
After this operation, 1416 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 keyutils amd64 1:1.3-5ubuntu1.1 [10.5 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libnfsidmap1 amd64 1:0.3-10ubuntu1 [14.1 kB]

```

This will successfully install the NFS Client on Client Y

ii) export folder /sample from NFS server X which should be available to client machine Y only. No other client should be able to access /sample

To export the folder /sample we need to first create it and give permissions to the owner, group and other for read,write and execute where only the owner has all 3 the rest do not have write permission.

```

ubuntu@ip-172-31-12-62:~$ sudo mkdir -p /sample
ubuntu@ip-172-31-12-62:~$ sudo chmod 755 /sample

```

To export the folder /sample we need to edit the /etc/exports folder and include the client's ip address and allow it to sync with the server in real-time .

```

ubuntu@ip-172-31-12-62:~$ sudo nano /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)
#
/sample 3.7.252.253(rw,sync,no_subtree_check)

```

To apply the configuration we need give the following command

```

ubuntu@ip-172-31-12-62:~$ sudo exportfs -ra

```

To verify the progress,

```

ubuntu@ip-172-31-12-62:~$ sudo exportfs -v
/sample 3.7.252.253(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_all_squash)

```

Now we change the rules of firewall to allow the signals from the client ip to come in any port through nfs

```

ubuntu@ip-172-31-12-62:~$ sudo ufw allow from 3.7.252.253 to any port nfs
Rules updated

```

iii) Mount exported nfs folder to the client machine Y on /mnt (client machine) and create folder, files inside mount point /mnt - troubleshoot if you are not able to create

folder or files inside mount point /mnt

Now to mount the exported folder in the client machine Y, we need to first create a directory /mnt

```
ubuntu@ip-172-31-7-73:~$ sudo mkdir -p /mnt
```

Now we will mount the exported folder to the client by giving this command

```
ubuntu@ip-172-31-7-73:~$ sudo mount 3.109.1.122:/sample /mnt
```

Here we have given the server's ip address and the folder to export and the directory to which the folder is to be mounted to

To make it persistent across reboots, we will edit the /etc/fstab file

```
ubuntu@ip-172-31-7-73:~$ sudo nano /etc/fstab
3.109.1.122:/sample /mnt nfs defaults 0 0
```

Now we check whether we can access the files created in the exported folder

We will use touch command and mkdir command to create file/folder in the exported folder

```
ubuntu@ip-172-31-7-73:~$ cd /mnt
ubuntu@ip-172-31-7-73:/mnt$ touch testfile
touch: cannot touch 'testfile': Permission denied
ubuntu@ip-172-31-7-73:/mnt$ sudo touch testfile
touch: cannot touch 'testfile': Permission denied
ubuntu@ip-172-31-7-73:/mnt$ mkdir testfile
mkdir: cannot create directory 'testfile': Permission denied
ubuntu@ip-172-31-7-73:/mnt$ sudo mkdir testfile
mkdir: cannot create directory 'testfile': Permission denied
```

But we are denied permissions, so we have to do troubleshooting to clear the doubt

#### **In server:**

We check the owners of the exported folder. i.e root. That is why we cannot r,w,x in the folder from the client so we change the owner of the folder to nobody and nogroup so that whoever can access the exported folder can r,w,x

```
ubuntu@ip-172-31-12-62:~$ ls -ld /sample
drwxr-xr-x 2 root root 4096 Apr 12 06:09 /sample
ubuntu@ip-172-31-12-62:~$ sudo chown nobody:nogroup /sample
ubuntu@ip-172-31-12-62:~$ sudo chown nfsnobody:nfsnobody /sample
```

Now the problem is cleared as we can successfully create files in the client side

#### **In Client:**

```
ubuntu@ip-172-31-7-73:/mnt$ sudo touch testfile
ubuntu@ip-172-31-7-73:/mnt$ mkdir testfile
mkdir: cannot create directory 'testfile': File exists
ubuntu@ip-172-31-7-73:/mnt$ sudo mkdir testfile
mkdir: cannot create directory 'testfile': File exists
```