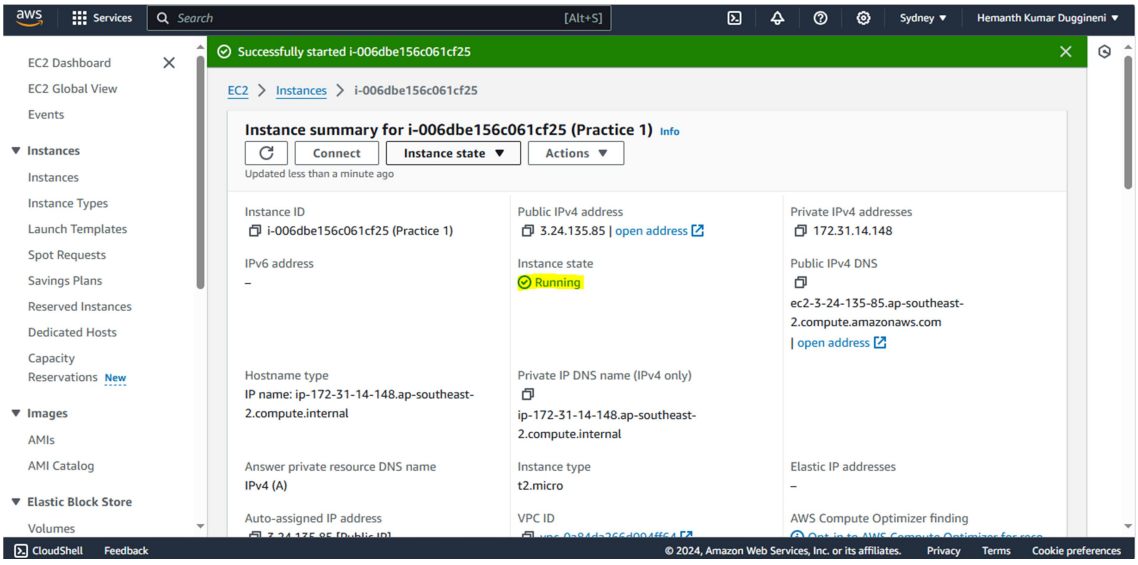
Deployment of Applications into a Server

Prerequisites:

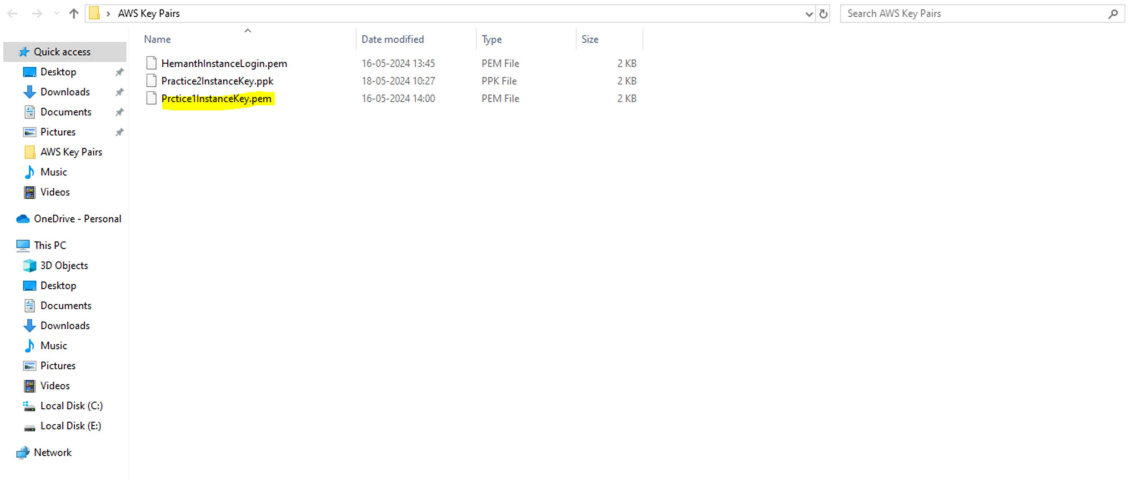
- First we have to know the Application deployment region to decrease the latency.
- Need a Server with an OS to run the application.
- We have to know the Application size and usage size to make decision on volume and size of the Server respectively.
- We have to know the Properties of the application like in which language the application has built (Node JS, Python, Java, HTML, CSS, PHP, etc…) and with which port it has been associated.
- We have to check the status of the tools in the Server to deploy the applications.

Setup of a Server with an OS before starting the Deployment of Applications:

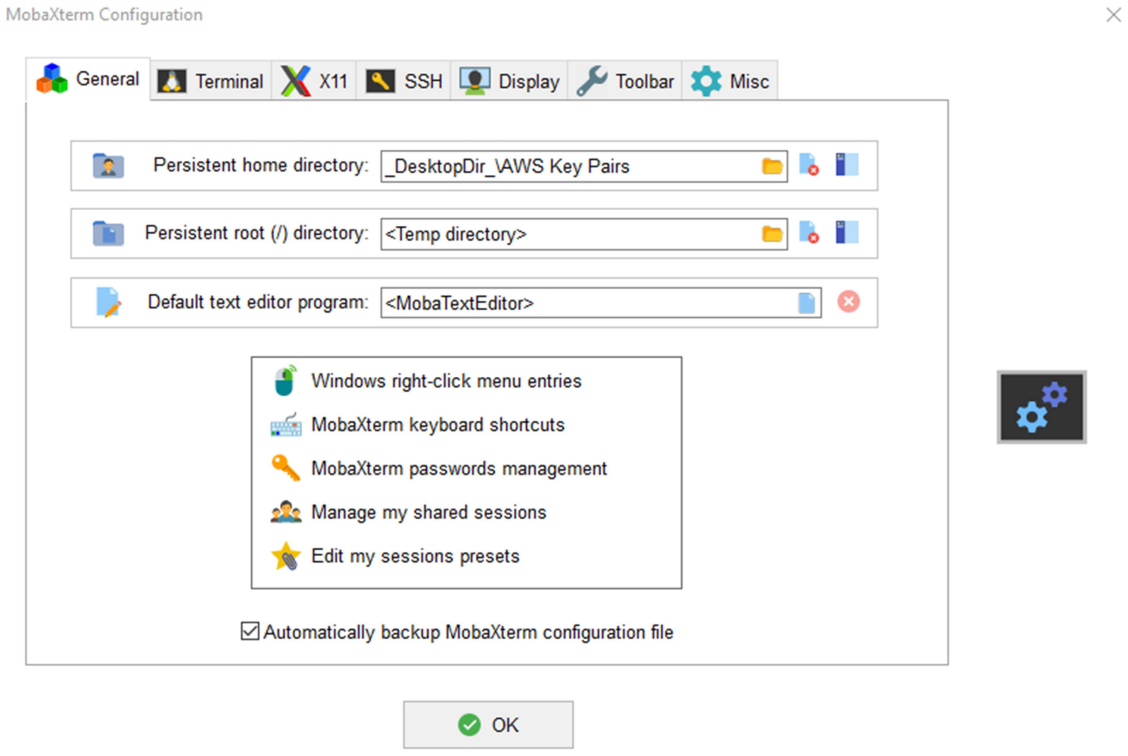
Step1: Check the status of the server which should be in running state.

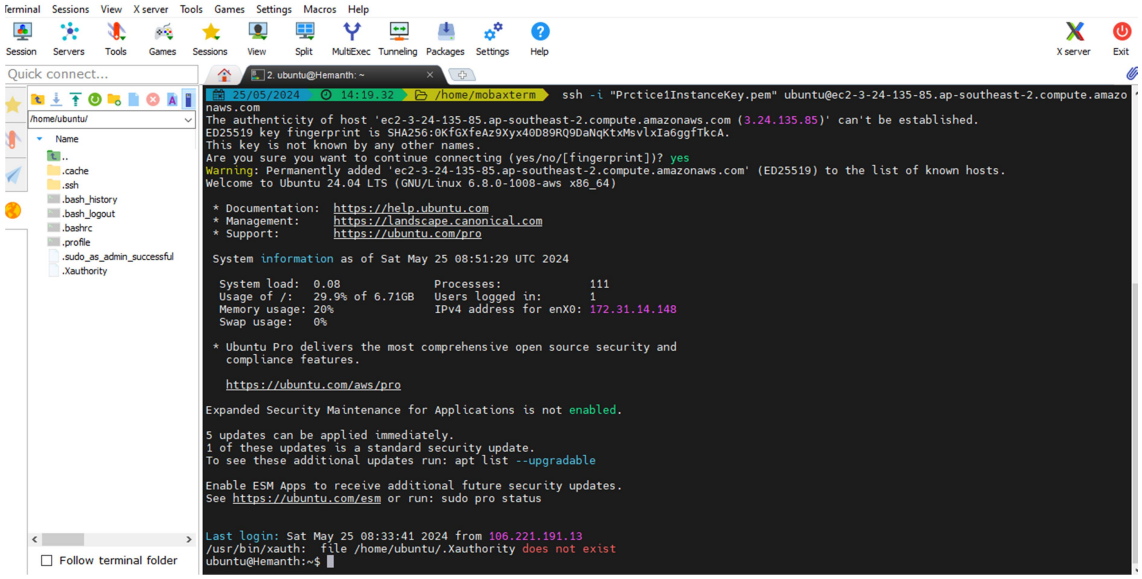


Step2: Connect through SSH client by locating Key pair.

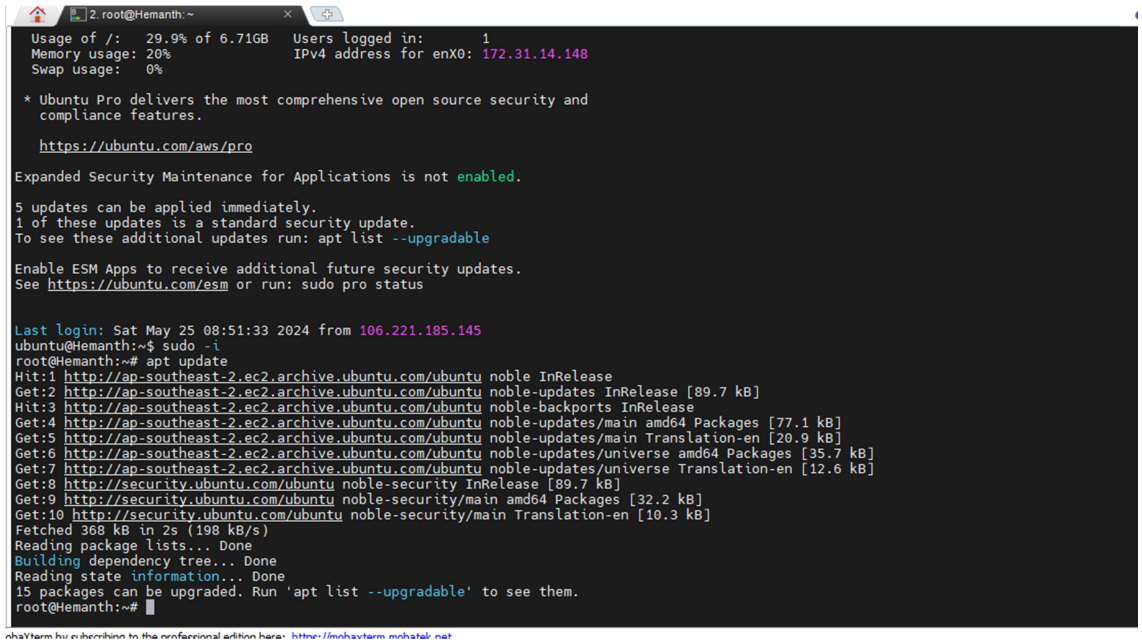


Step3: Here I am using SSH client as MobaXterm and setting my Key pair path as default location in MobaXterm.



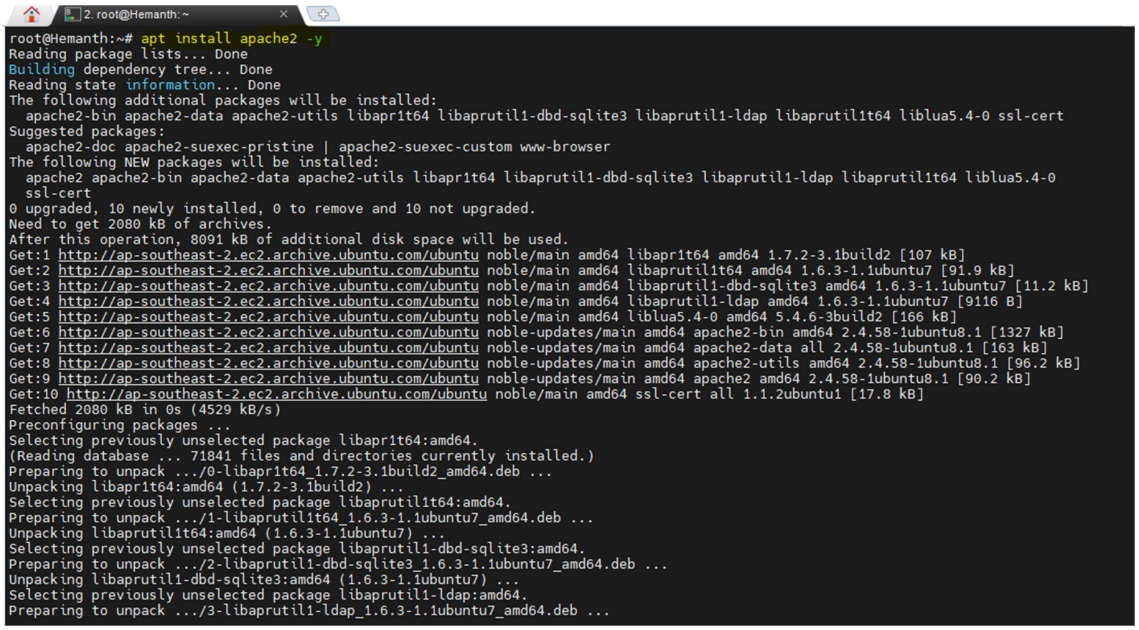


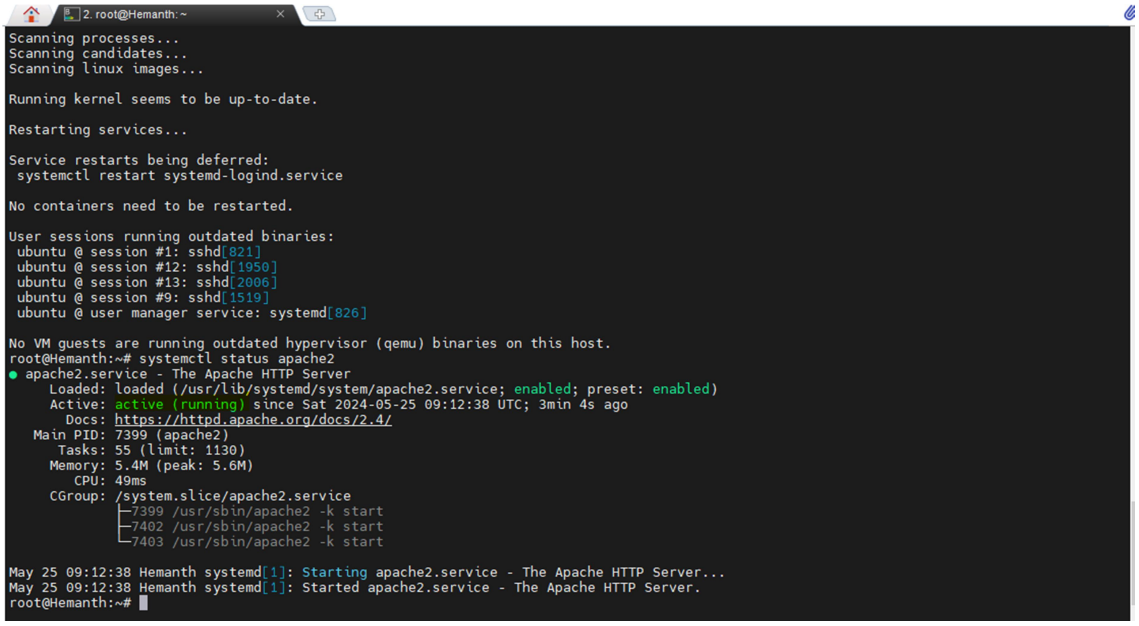
Step4: Switch to Root access and here I am using a newly created server with Ubuntu Linux, so update the server with **'apt update'** command.



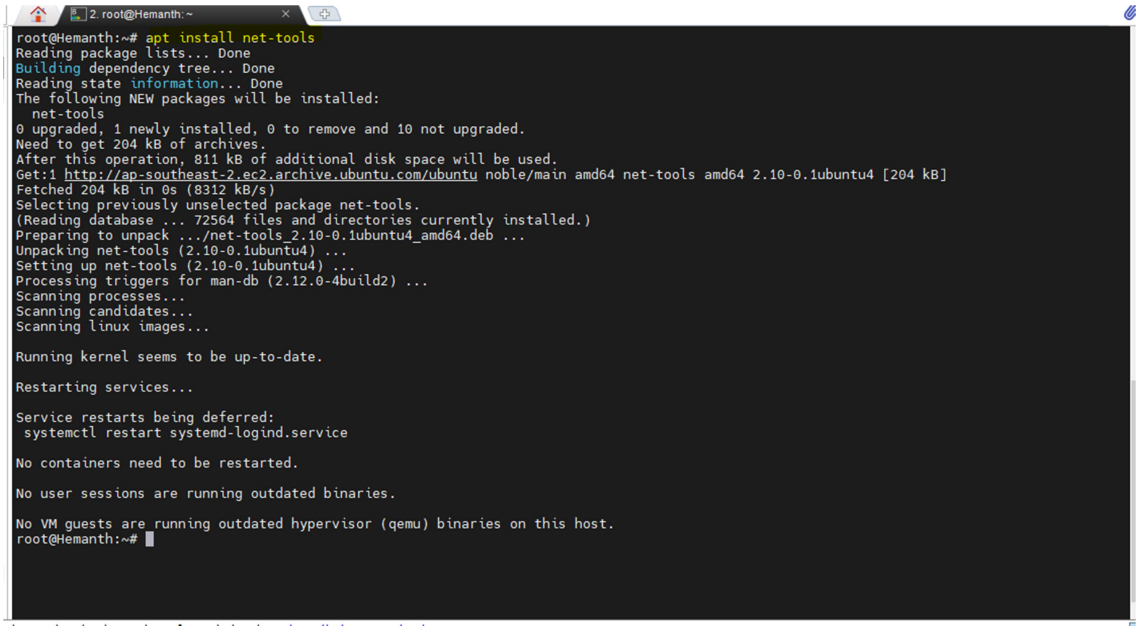
Setups in a Server before Deploying Static Applications (HTML, CSS) into a Server:

Step1: Install the Apache2 web server by using **'apt install apache2 –y'** command to create an access point of the CSS template to users and check the active status of the Apache2 by using **'systemctl status apache2'** command.

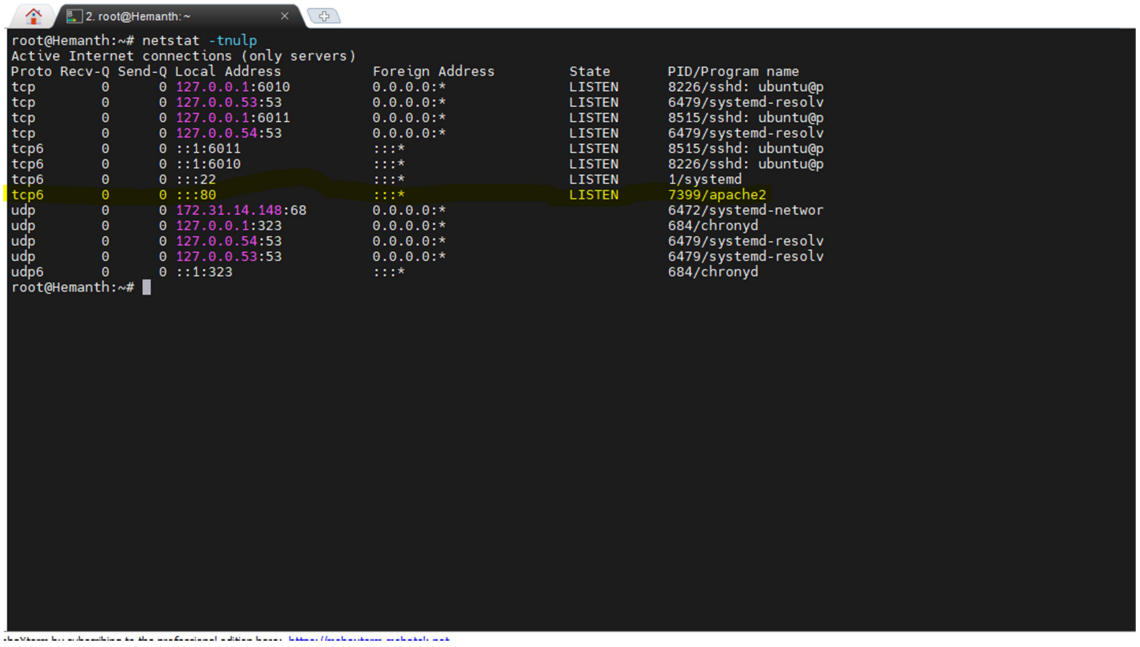




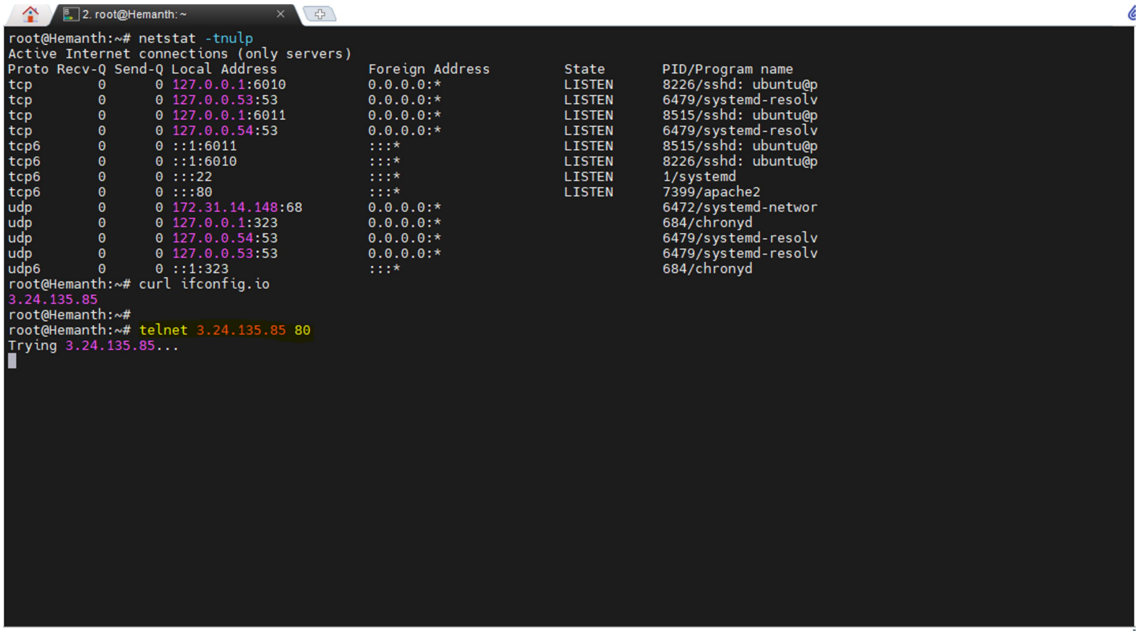
Step2: Since I am using the newly created server we have to install net-tools too with **'apt install net-tools'** command, to know the listen status of the ports.



Step3: By entering **'netstat –tnulp'** command we can see the Listen status of the ports.

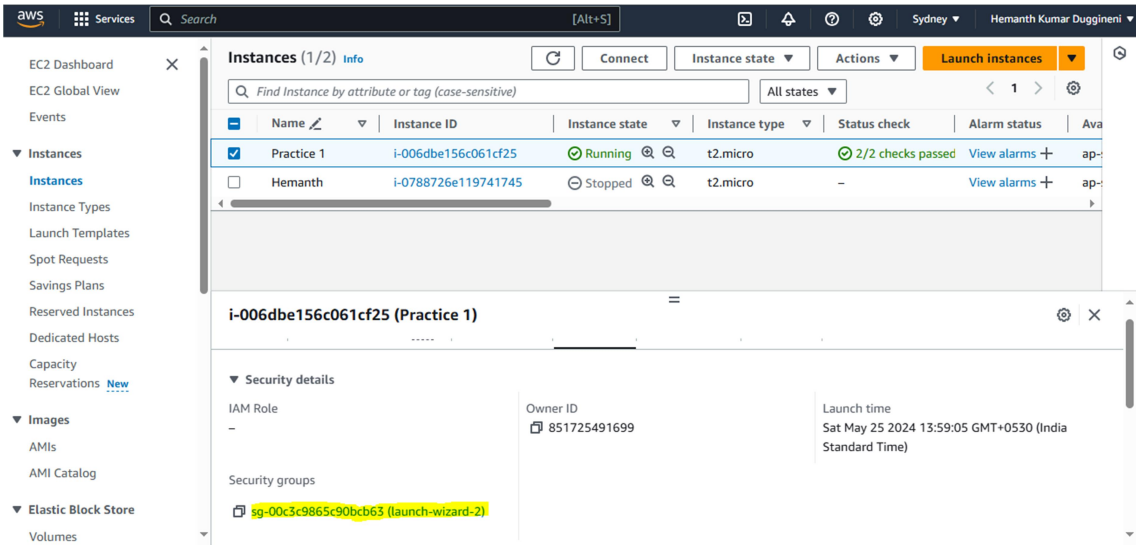


Step4: Check the port is connecting or not by entering **'telnet <Public IP> <Port>'** command.

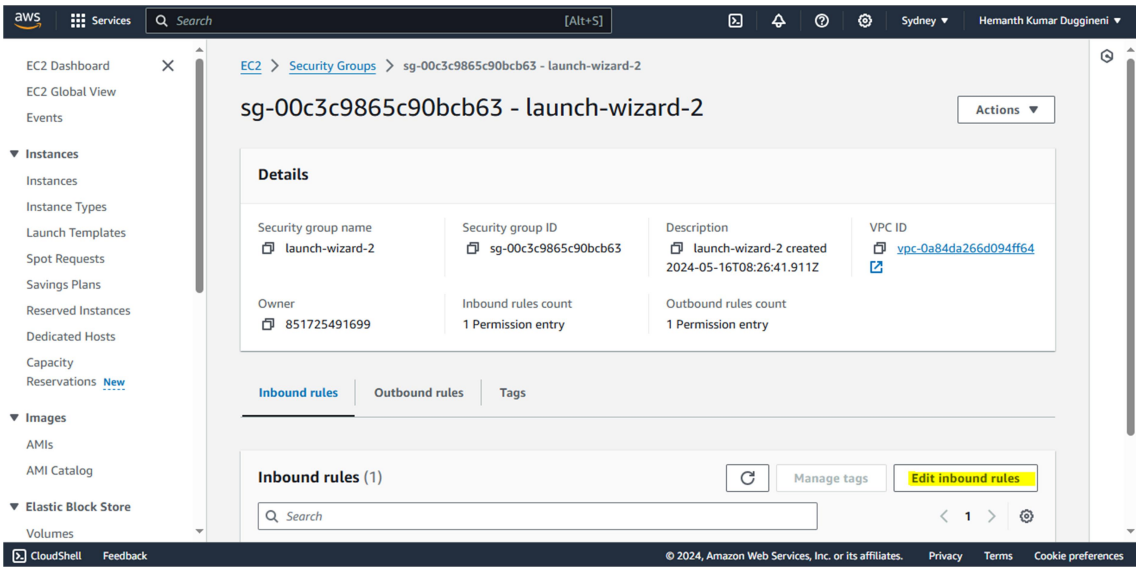


Step5: Here we can see that trying to connect but not connecting via required port because we didn't configure the port in security groups of the server in AWS console. So we have to configure required port in security groups of the server in AWS console.

Go to Instances page and select your server, under Security section click on the link under Security groups.



Under Inbound rules section click on edit inbound rules.

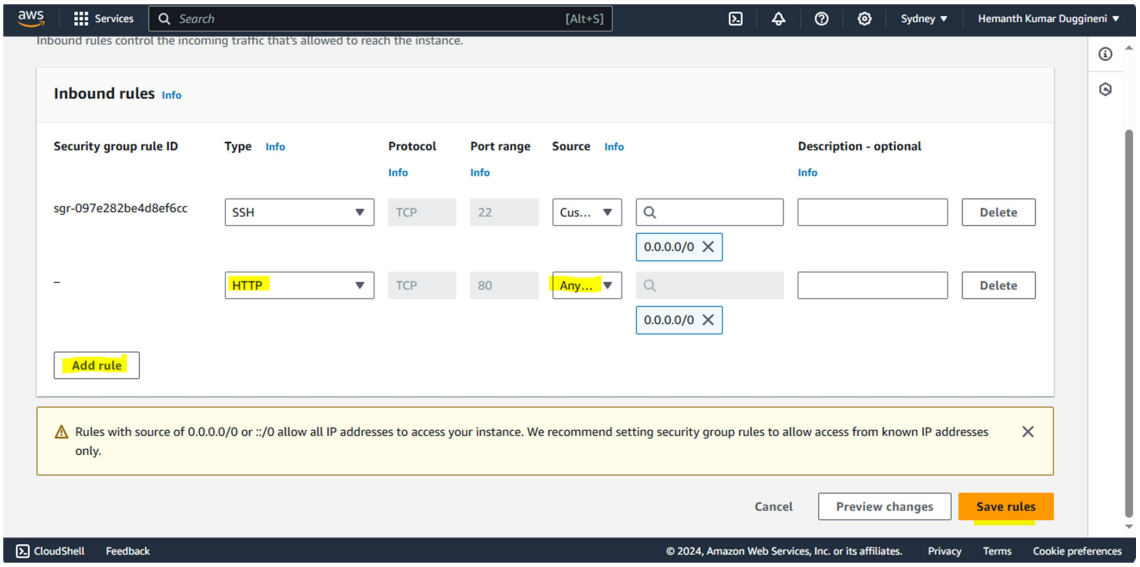


Click on Add rule and select

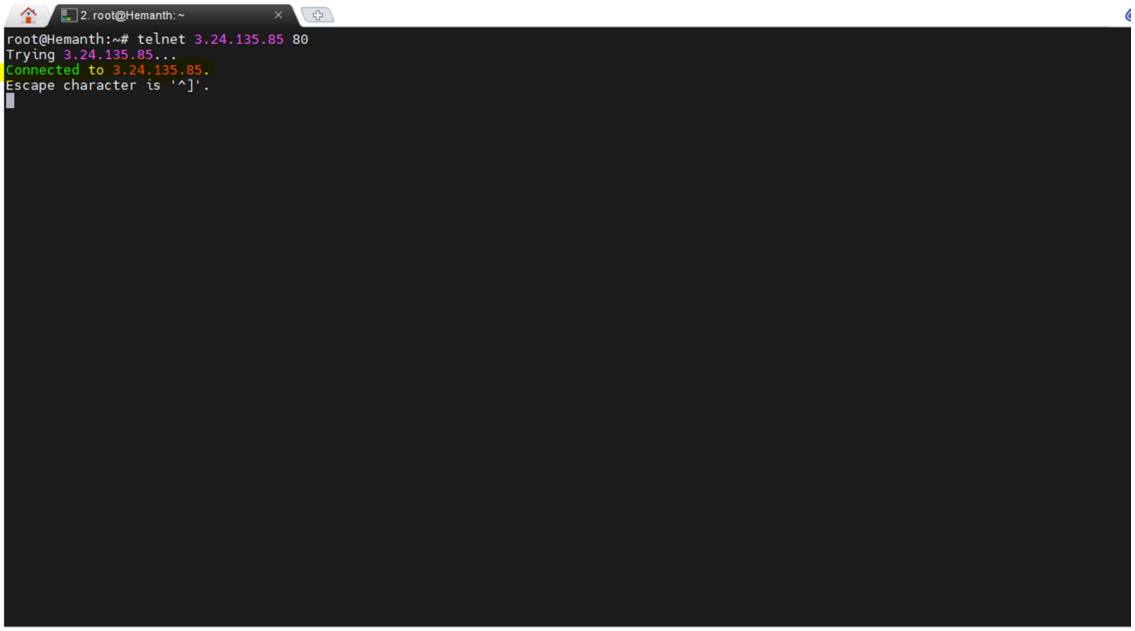
Type: HTTP (required protocol type)

Source: Anywhere-IPv4 (required access network IP addresses)

And click on save rules.



Step6: Again check the port is connecting or not by entering **'telnet <Public IP> <Port>'** command.



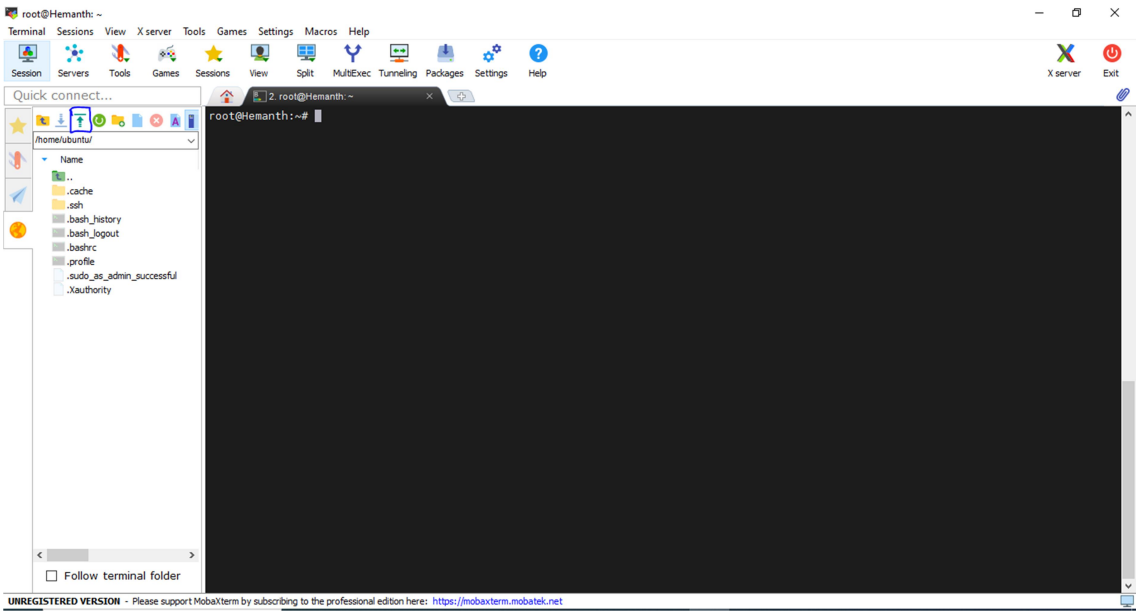
Here we can see that server is responding via required port, as we can confirm that access point has been created via web server to access the applications in the server.

Deployment of the Static Applications (HTML,CSS) into a Server:

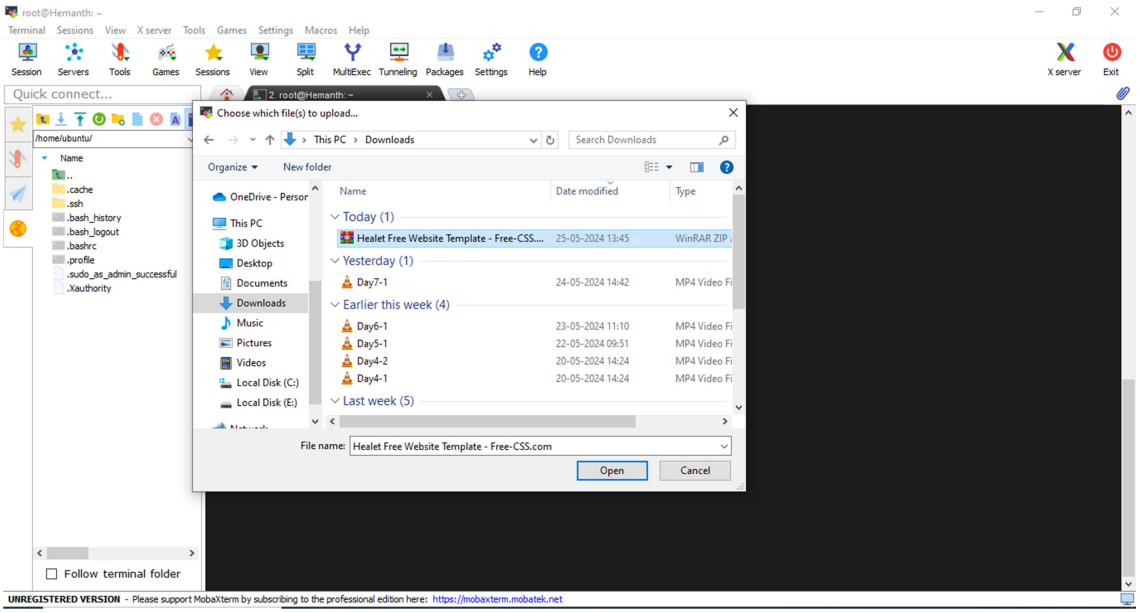
Step1: Here I am using MobaXterm to transfer our application files to our server.

Note: Before transferring the files to the server it is better to zip the files to prevent the loss of data of the application in the server.

Click on Upload button.

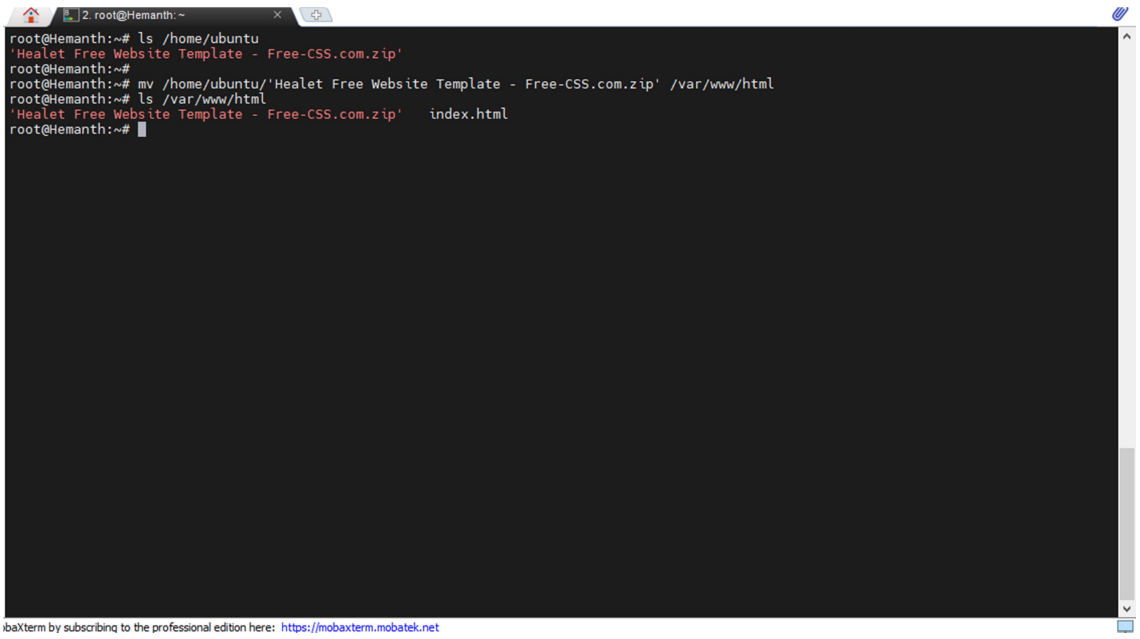


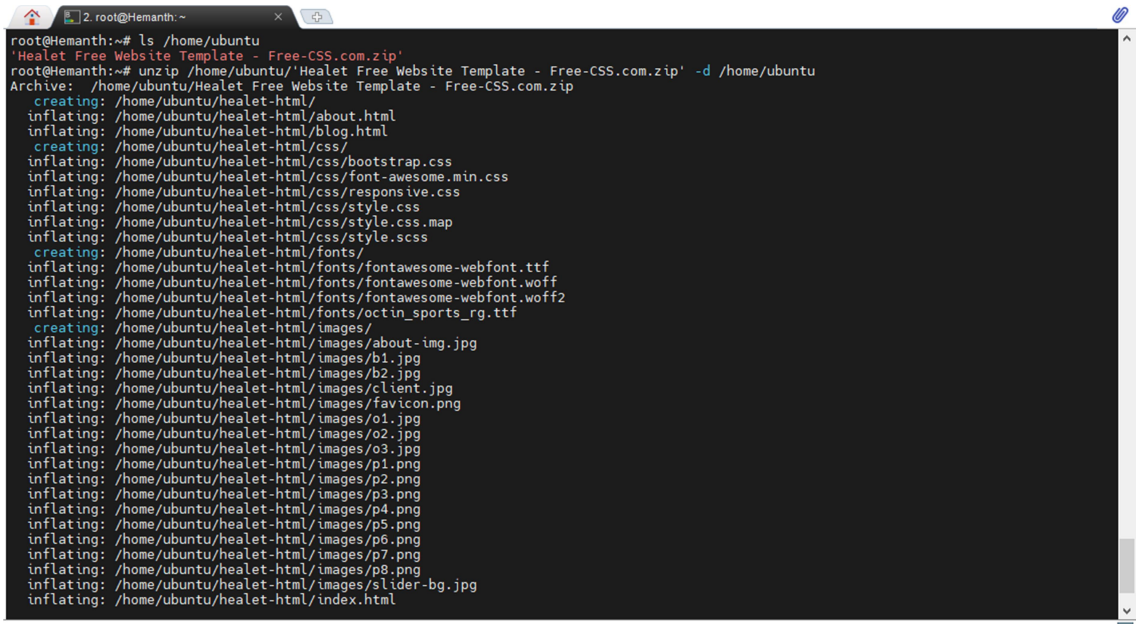
Select our Application files to upload.

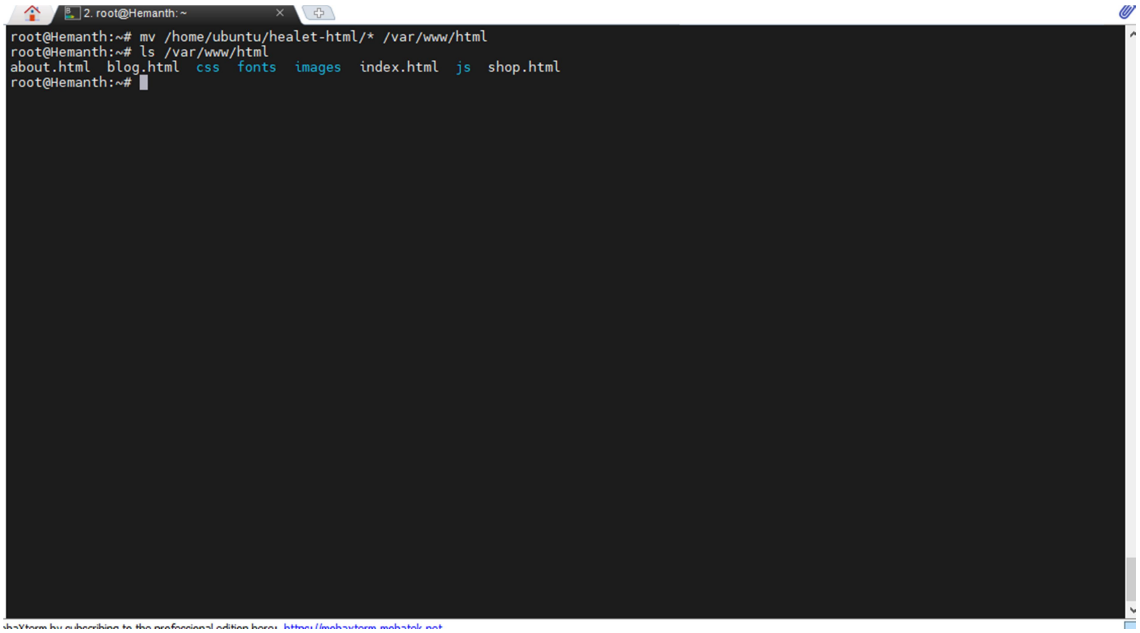


Note: We can use '**git clone <Github URL>'** command to transfer our application files from our Github account and we can proceed with the deployment process.

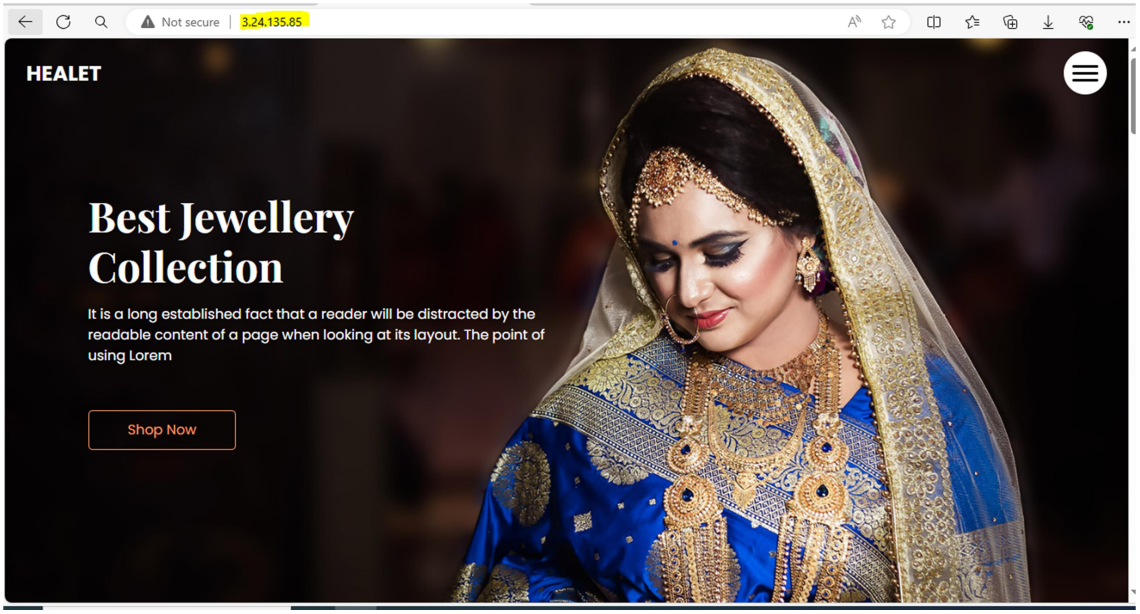
Locate the transferred files, unzip them with the command **'unzip <File path> -d <Destination path>'** and move to Web server default location (/var/www/html) with command **'mv <Source path> <Destination path>.**







Step2: Now try to open the access point (<Public IP>:<Port>) in a browser.

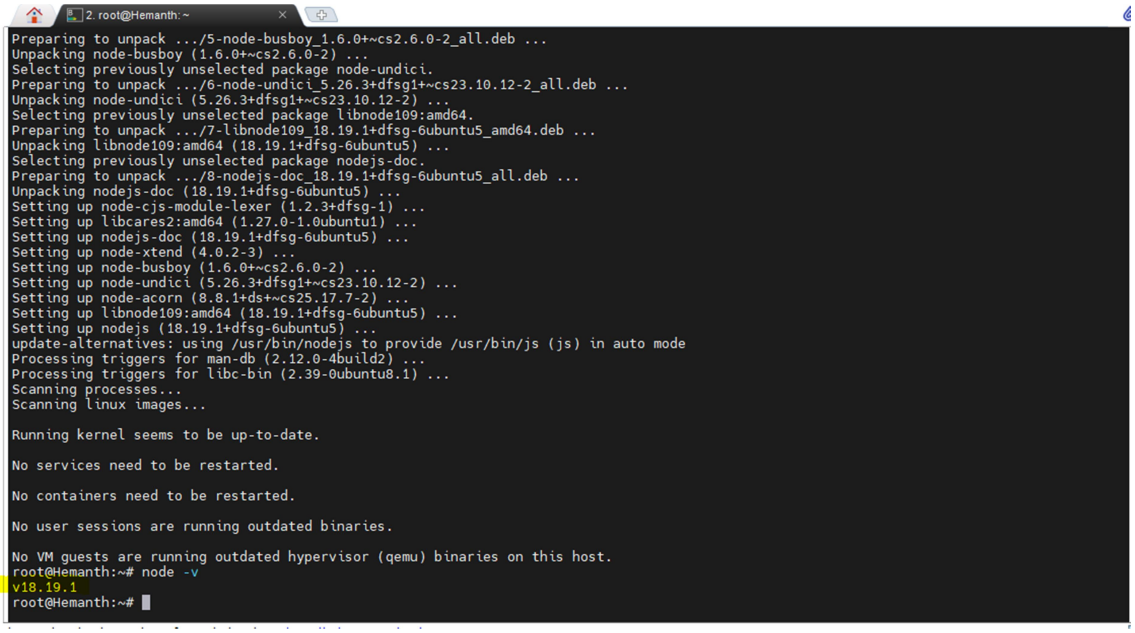

Not secure
3.24.135.85
HEALET
Best Jewellery Collection
It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem
Shop Now

Deployment of a NodeJS Application into a Server:

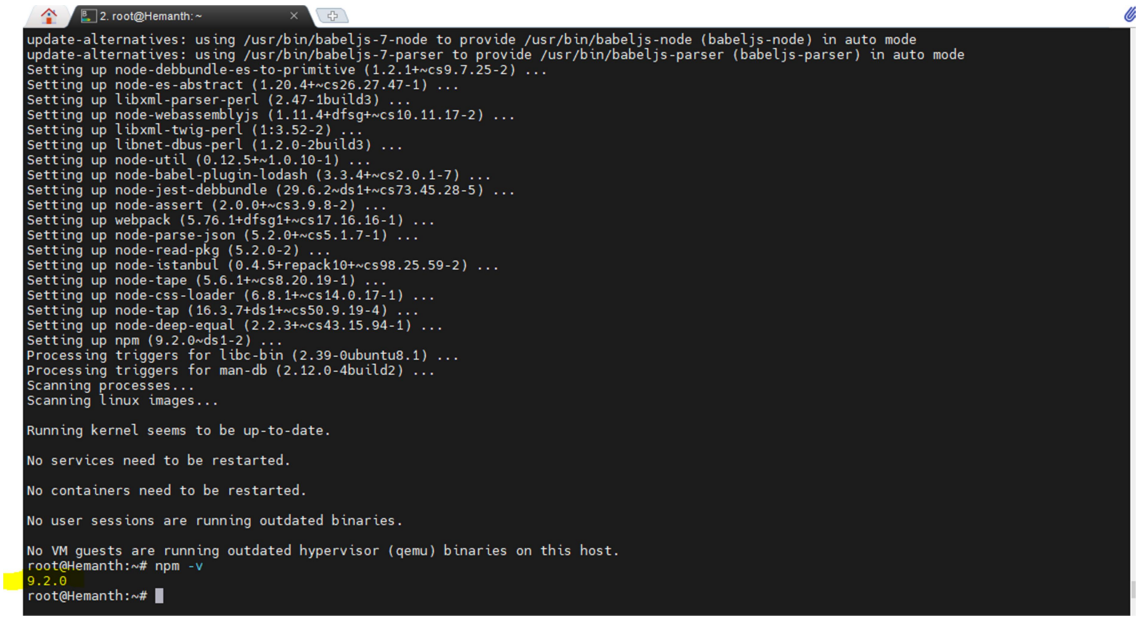
- In NodeJS applications we have Frontend and Backend applications.
- Below I will show how to deploy both applications step by step.

Setups in a Server before Deploying NodeJS Applications into a Server:

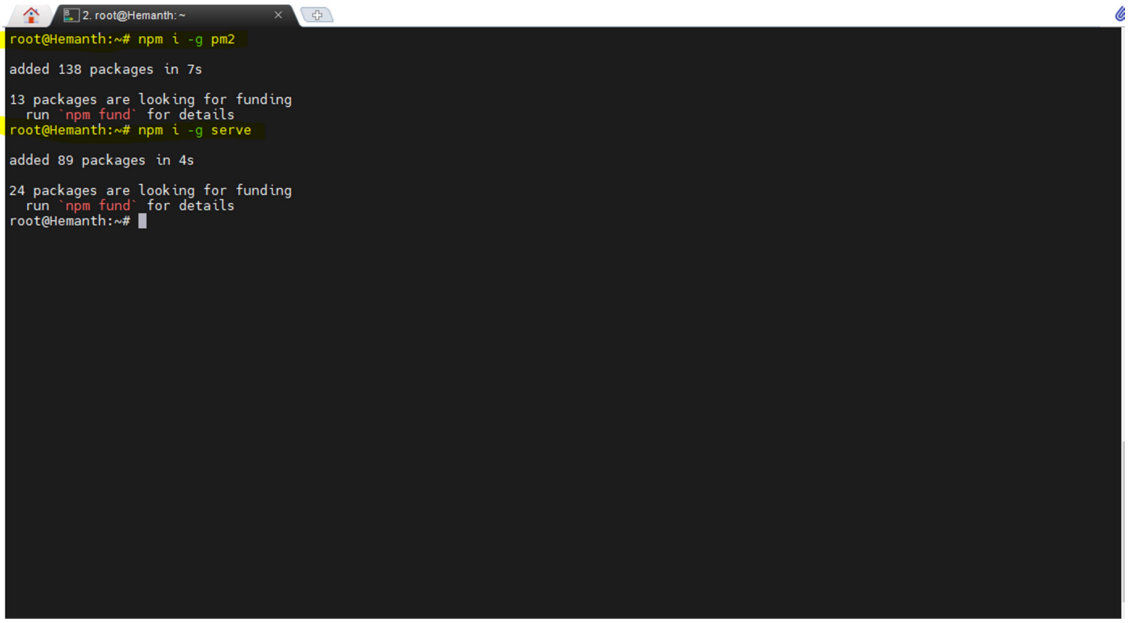
Step1: By using '**apt install nodejs -y'** command we have to install nodejs tools to resolve the compatibility setting of the nodejs applications with the server and by using **'node –v'** command we have to check the version and confirm that nodejs tools have been installed.



Step2: By using **'apt install npm –y'** command we have to install Node Package Manager Tools to resolve the compatibility issue of the applications while running and by using **'nmp –v'** command we have to check the version and confirm that npm tools have been installed.

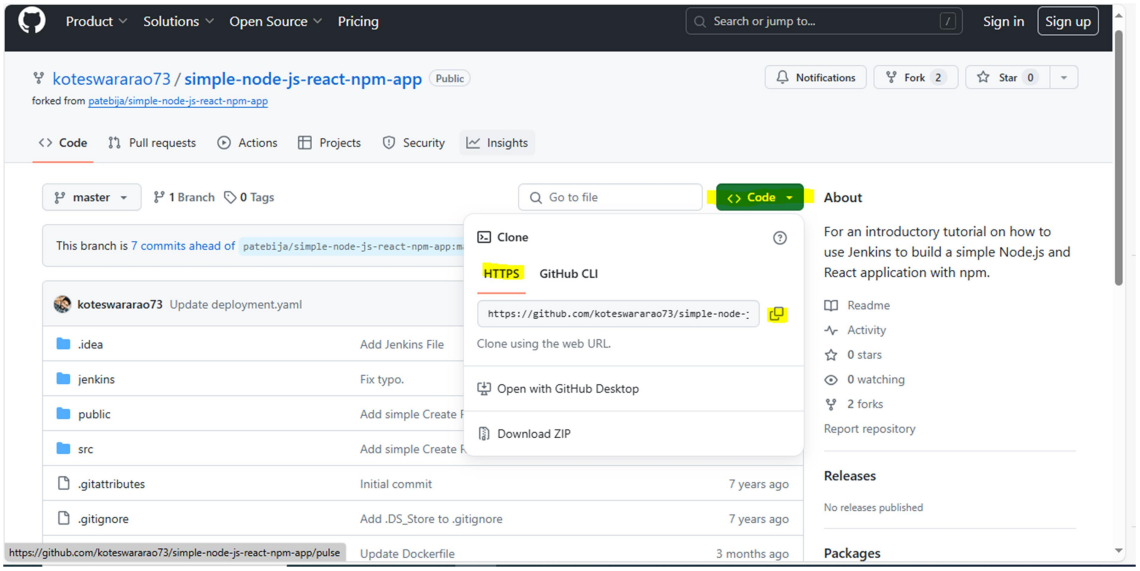


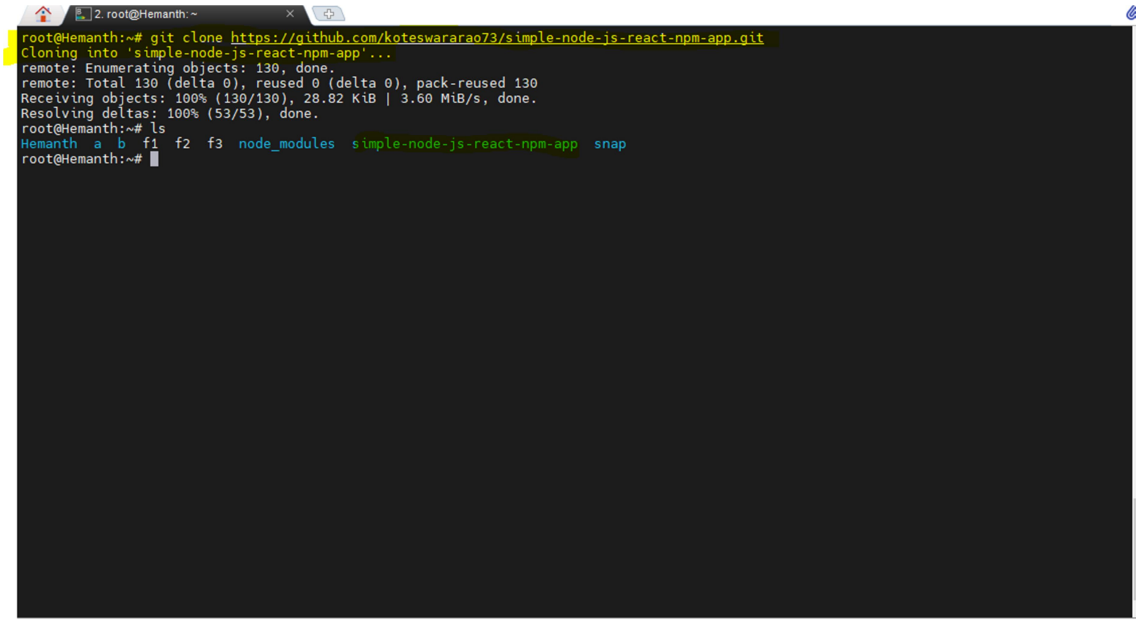
Step3: By using **'npm i pm2'** and **'npm i serve'** commands we have to install 'Package Manager Monitor' and 'Serve' tools to run the NodeJS applications.



Deployment of Frontend NodeJS Applications:

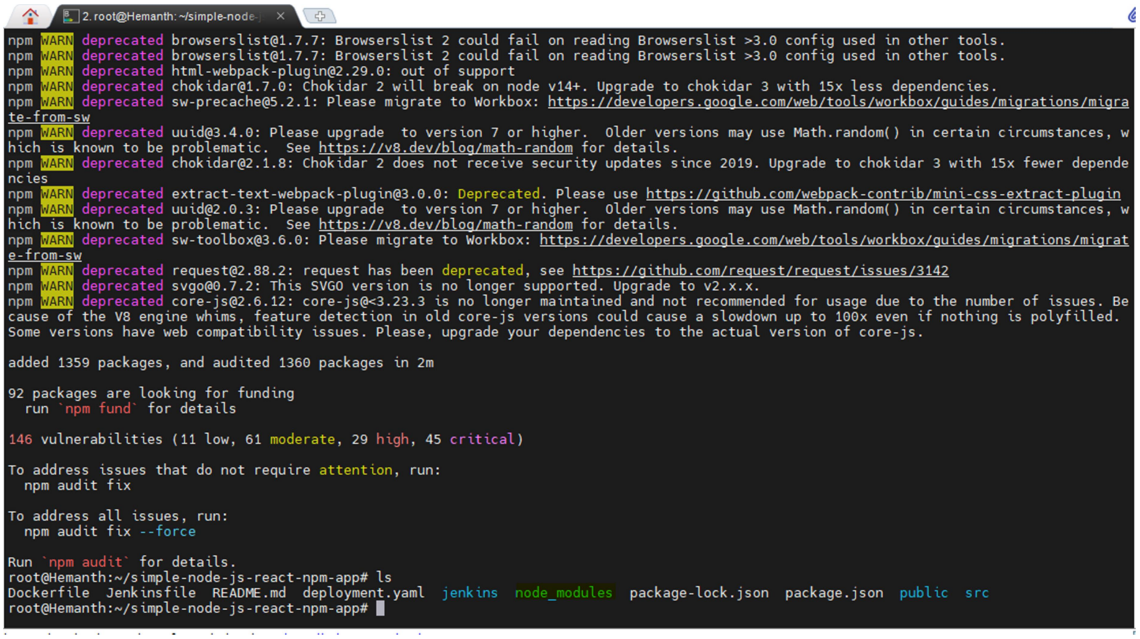
Step1: Here I am using **'git clone <Github URL>'** to transfer our files into the server from Gitbhub account and saved as the Repository name which is in the Github account.



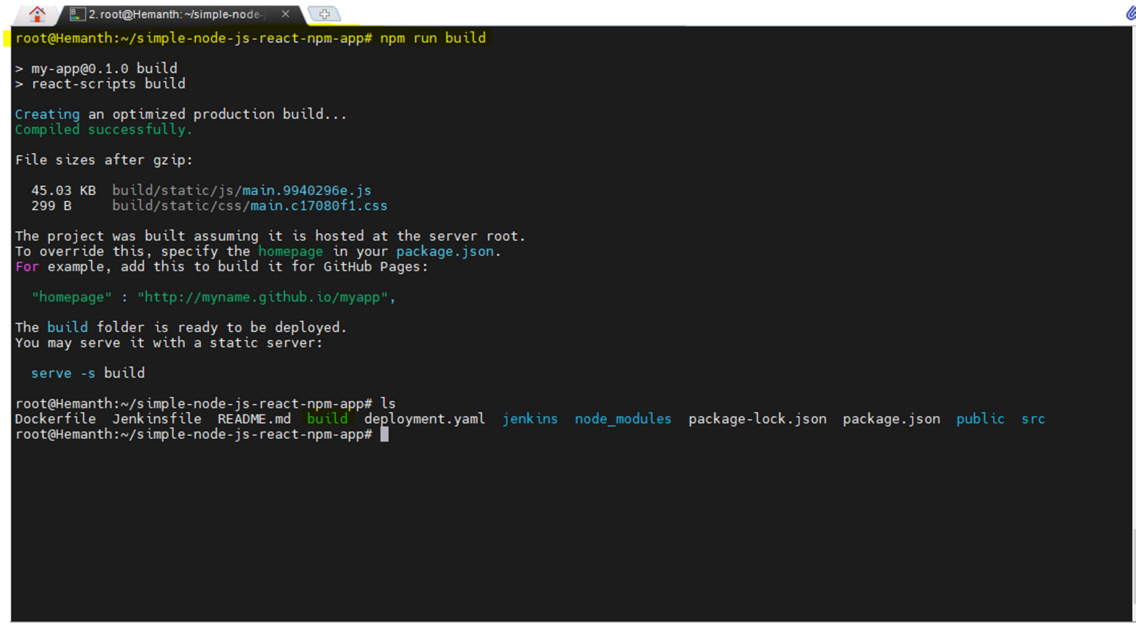


Step2: Enter into the transferred file with **'cd <Directory Name>'** command and by using **'npm i'** command install the compatibilities for the application files and the compatibilities can be stored in a directory named as 'node_modules'.

Note: Before using any npm, nodejs, pm2, serve tools your cursor location should be in the project or application directory.

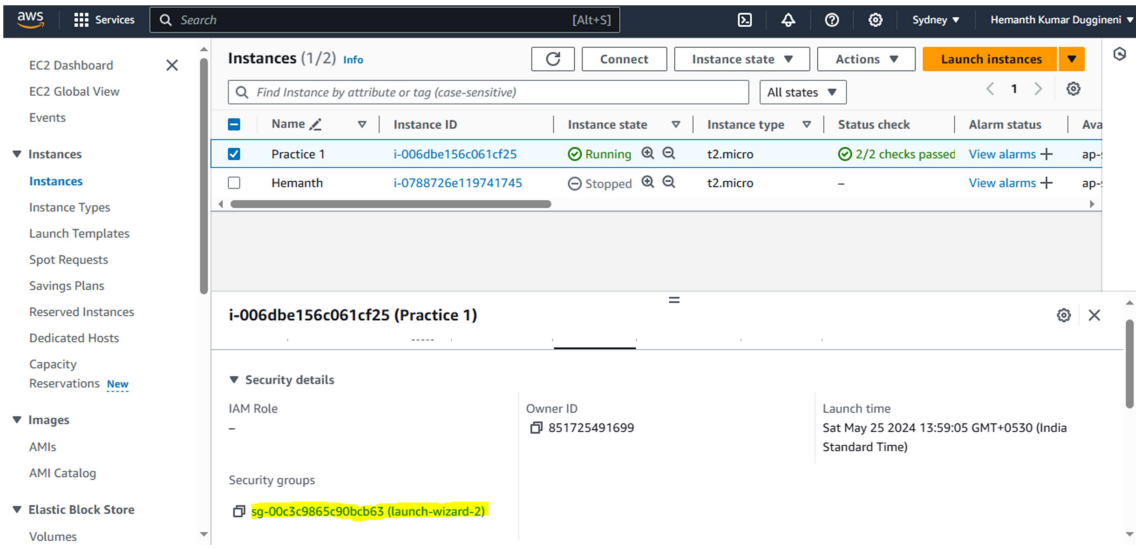


Step3: By using '**npm run build'** command we can build the application with the existing files and which can be stored in a directory named as 'build'.

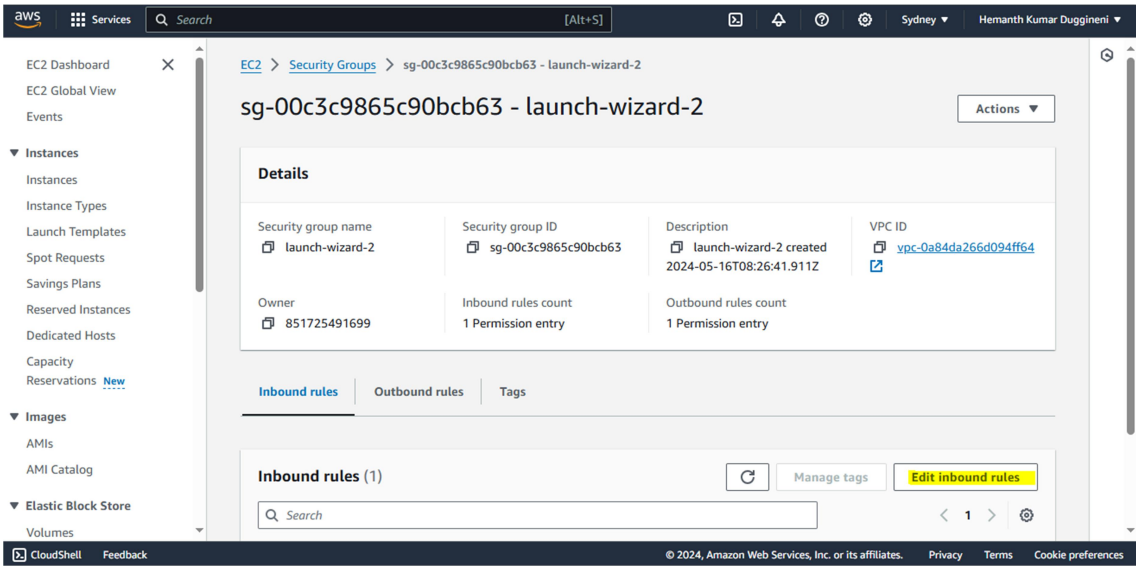


Step4: Now we have to allow the application port in Security Groups.

Go to Instances page and select your server, under Security section click on the link under Security groups.



Under Inbound rules section click on edit inbound rules.



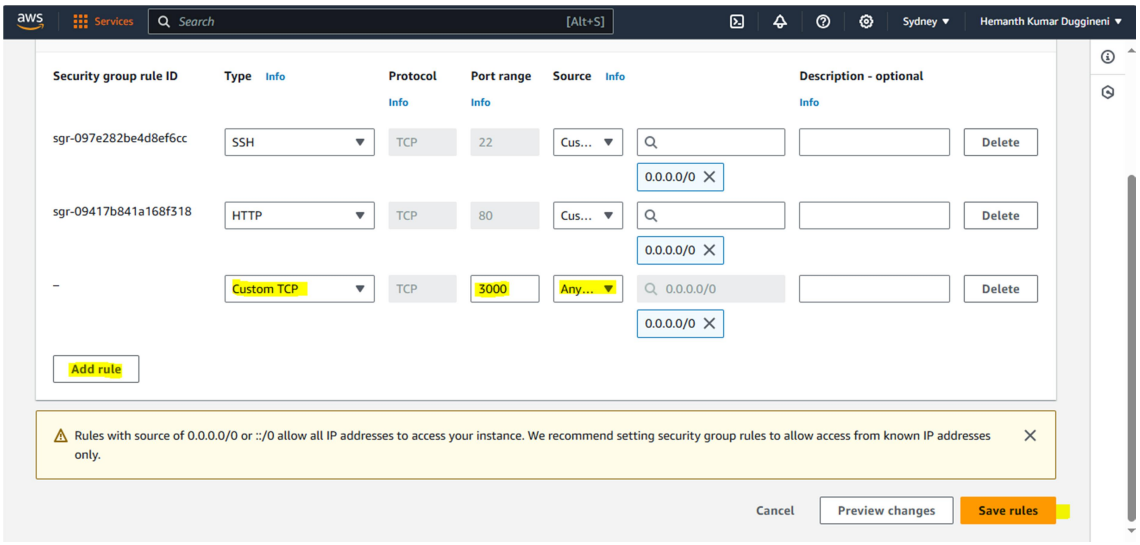
Click on Add rule and select

Type: Custom TCP (required protocol type)

Port Range: Required Port number

Source: Anywhere-IPv4 (required access network IP addresses)

And click on save rules.

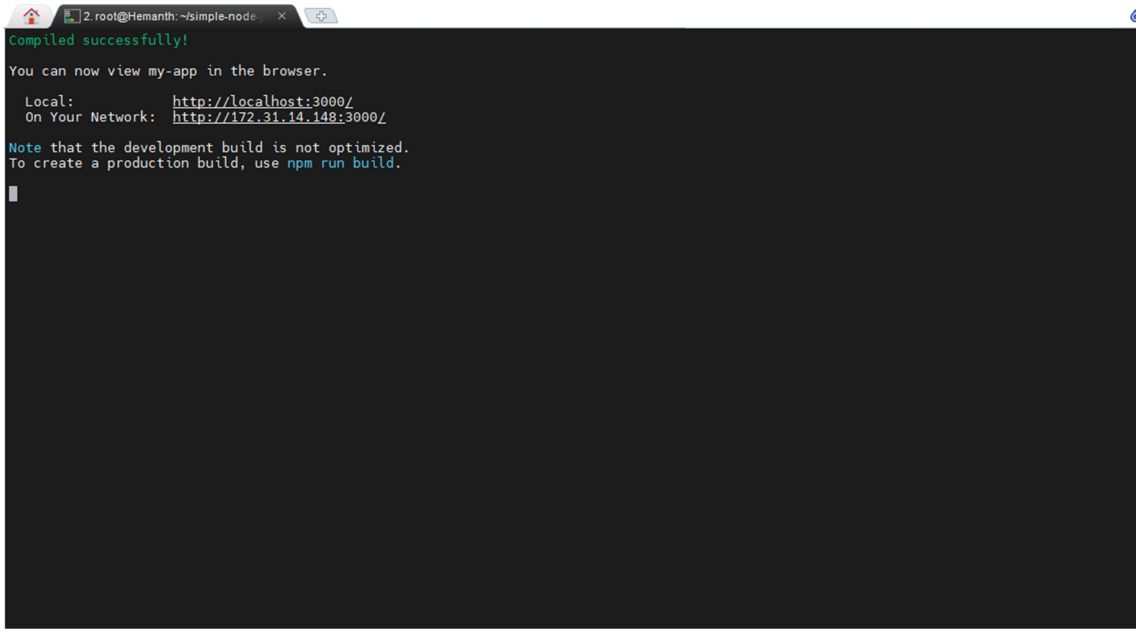


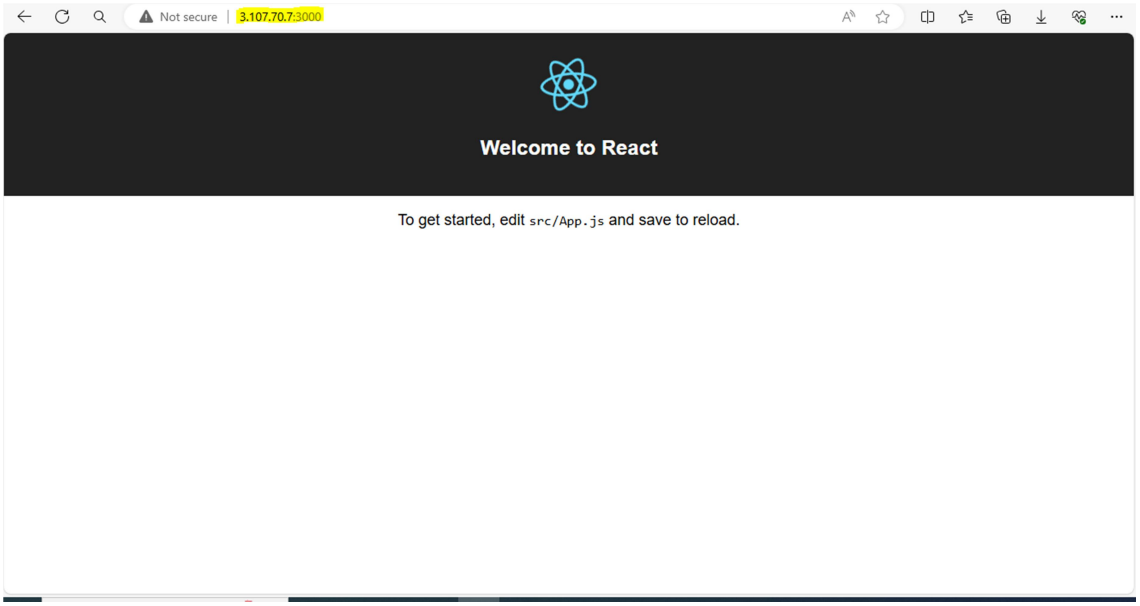
Up to here deployment of application is done.

To run the deployed application there are two ways:

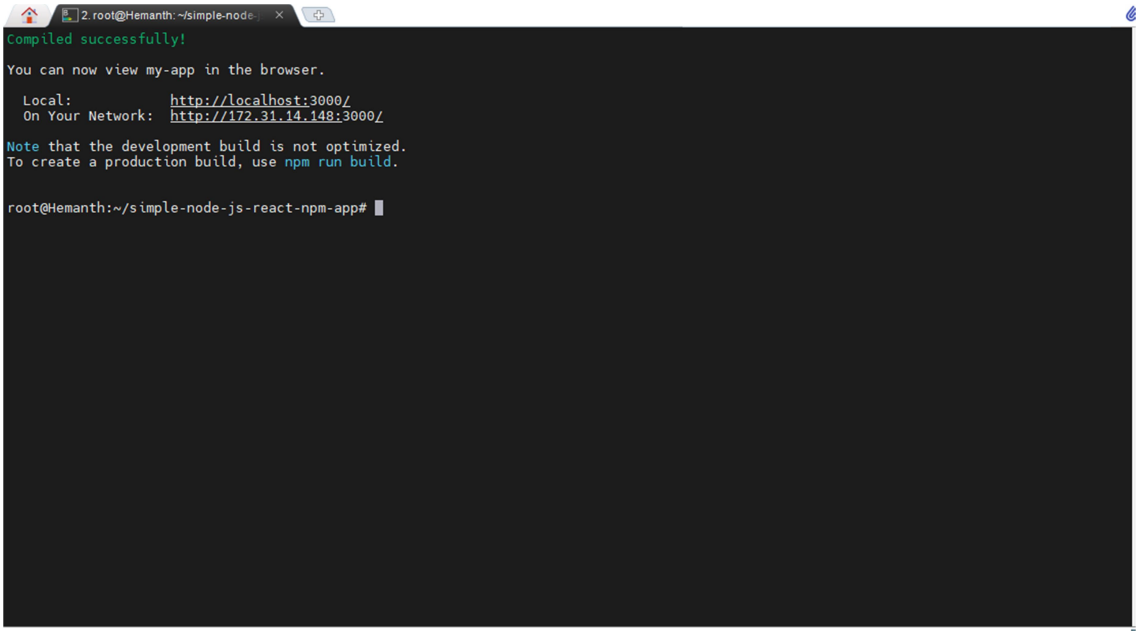
Temporary Run: By using **'npm start'** we can run our application temporarily up to we click on 'ctrl+c' in the server.

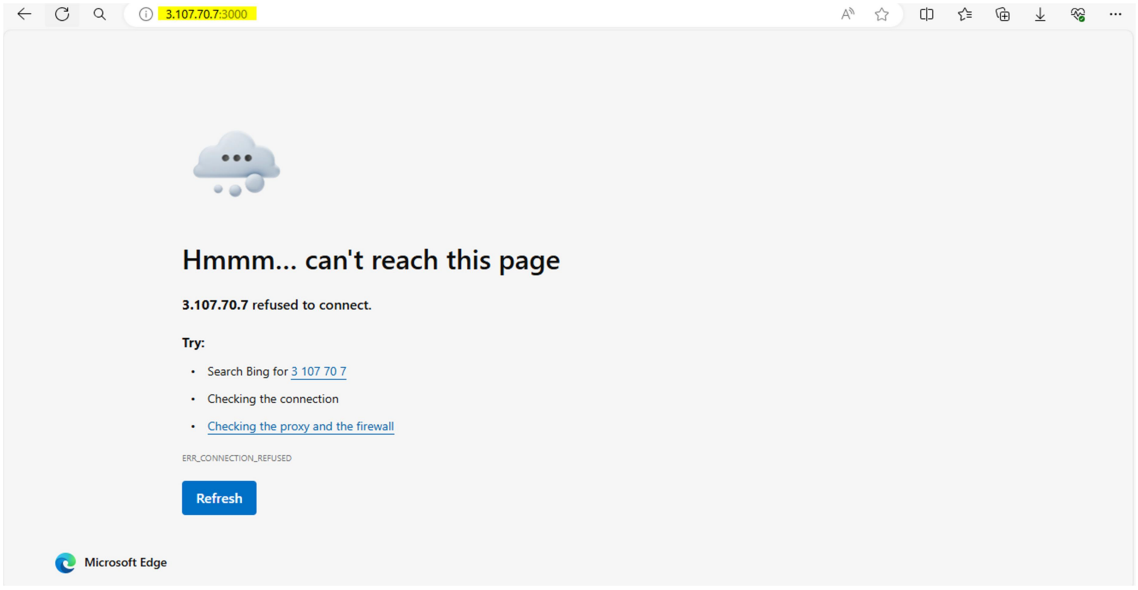
After passing the command **'npm start'** the application will start running with the access point.



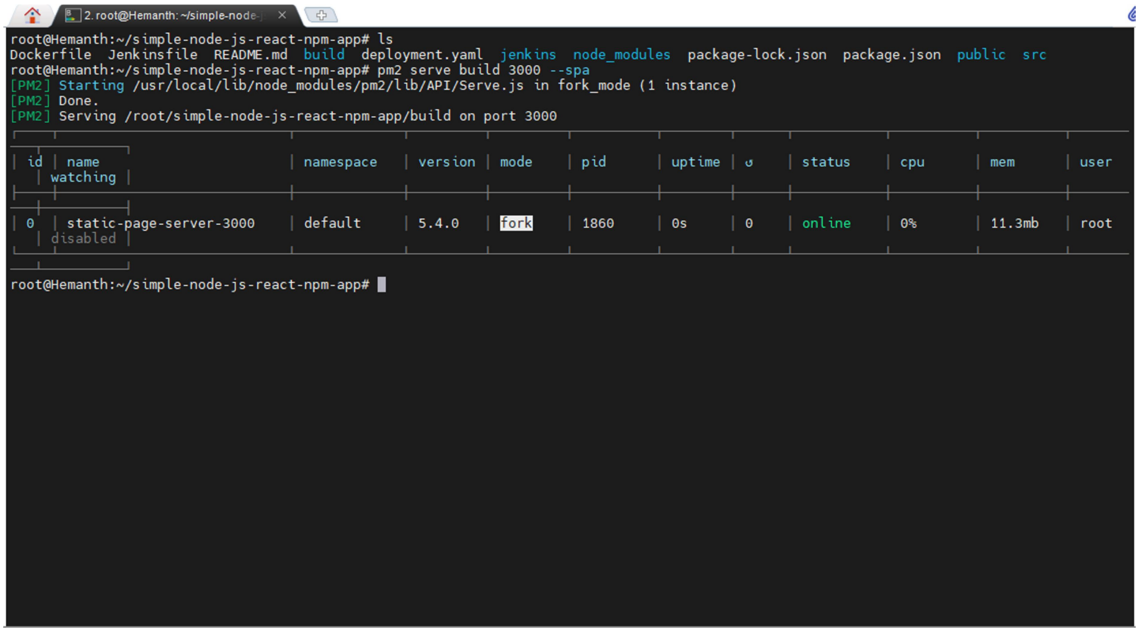


After entering ‘ctrl+c’, the application will stop running with access point.

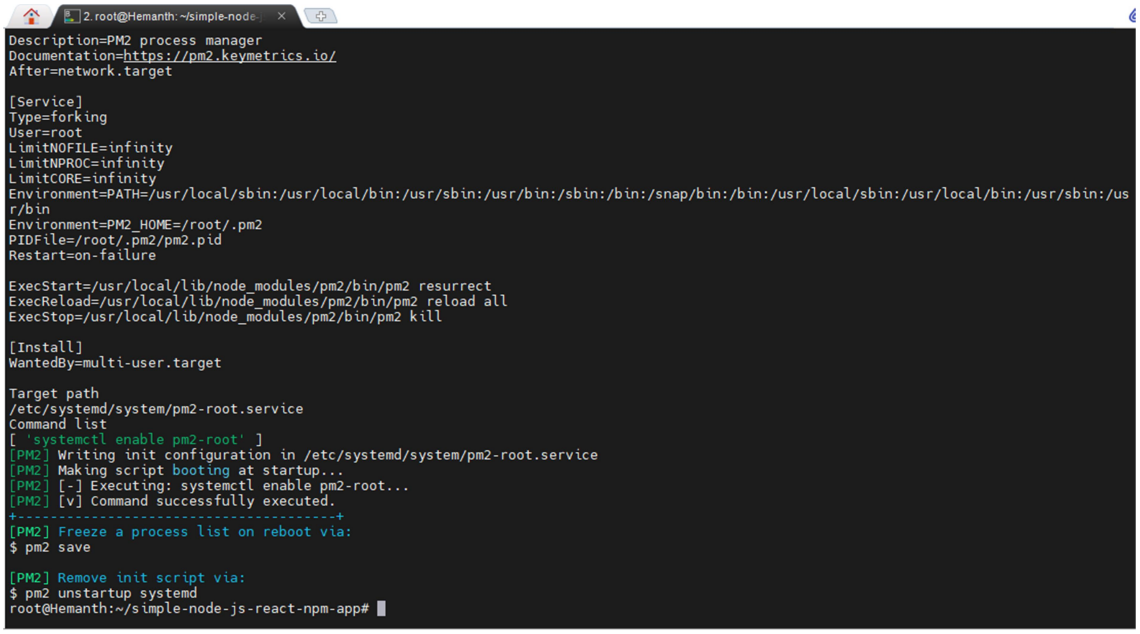




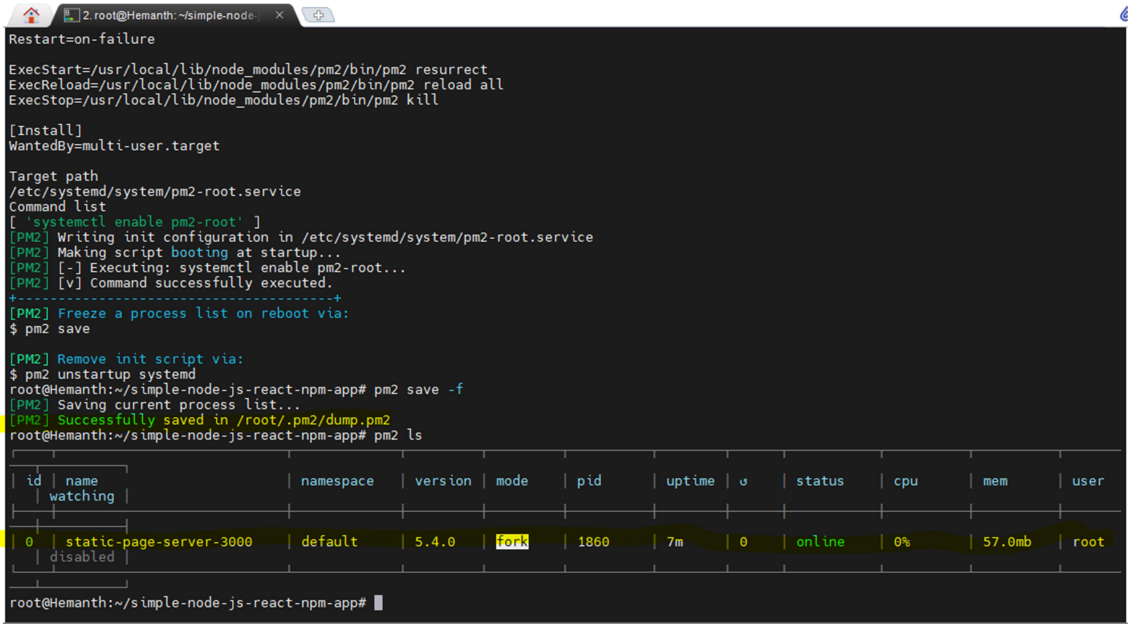
Permanent Run: By using **'pm2 serve <Application's Build Directory> <Port> --spa'** command run the application in the server.



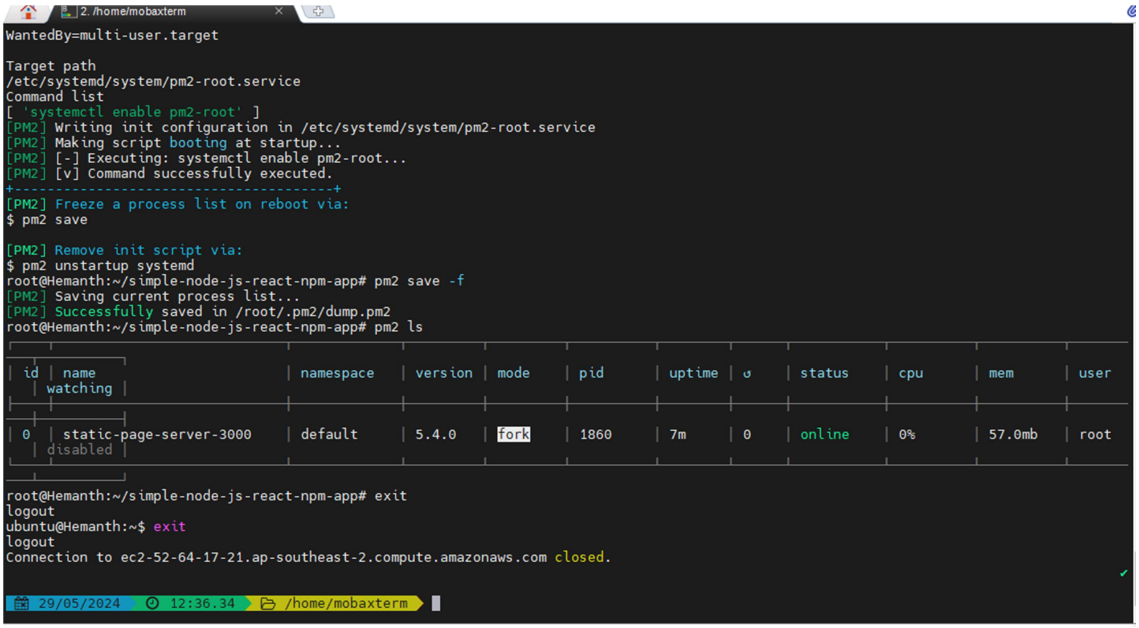
By using **'pm2 startup'** command start the application.

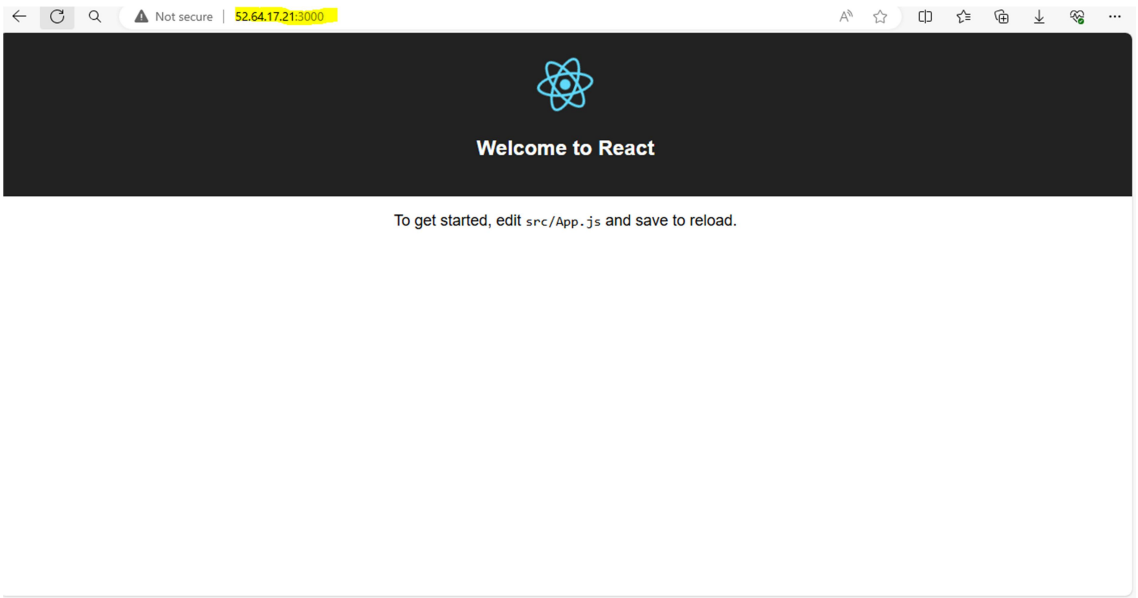


By using **'pm2 save –f'** command save the running programs forcefully to run the application even if user got logged out from the server and by using **'pm2 ls'** command we can check the application running status.



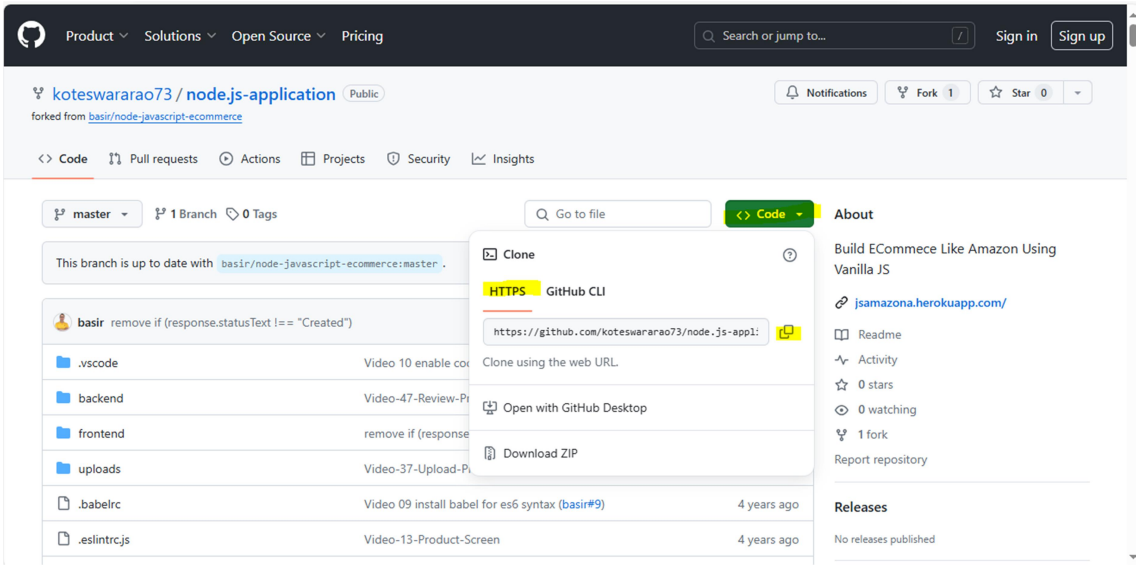
Now I am logged out from the server and even though the access point is working fine.

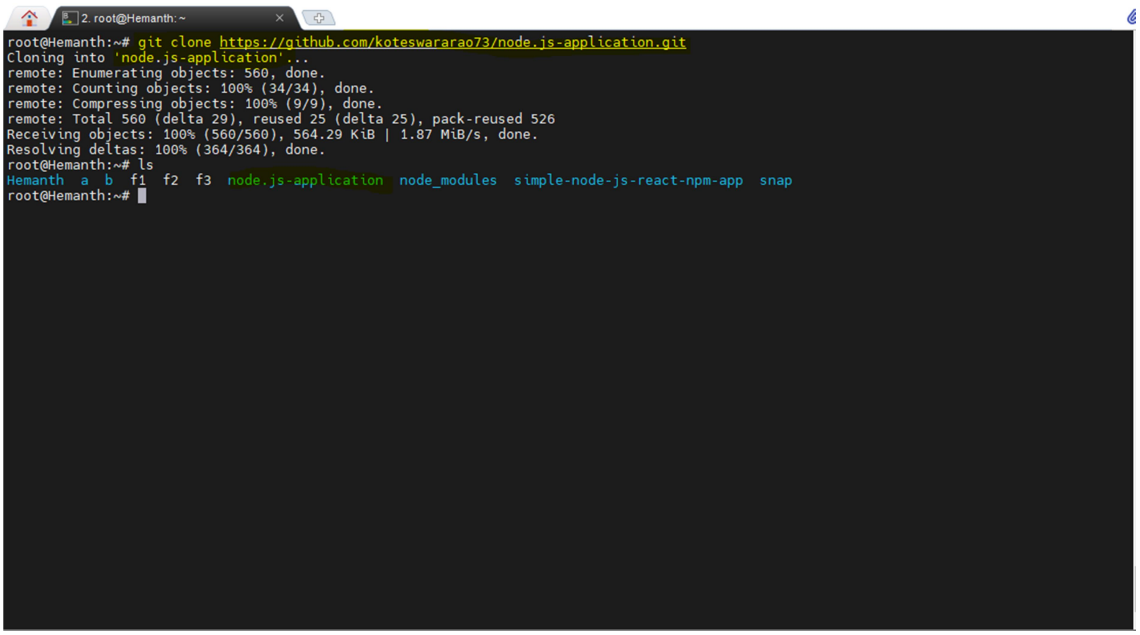




Deployment of Backend NodeJS Applications:

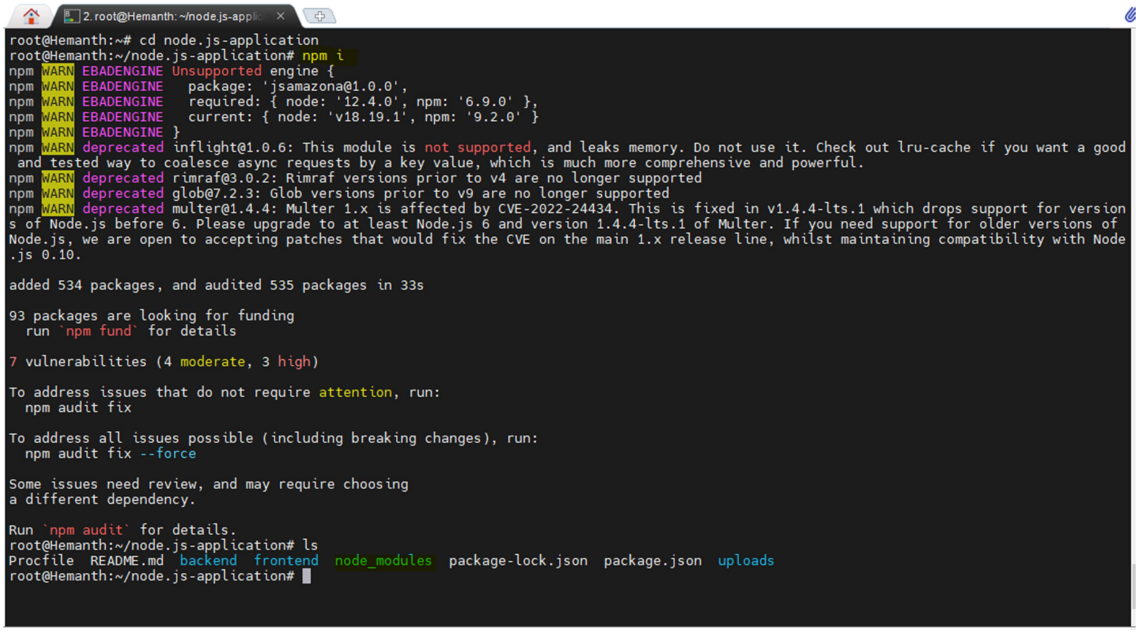
Step1: Here I am using **'git clone <Github URL>'** to transfer our files into the server from Gitbhub account and saved as the Repository name which is in the Github account.



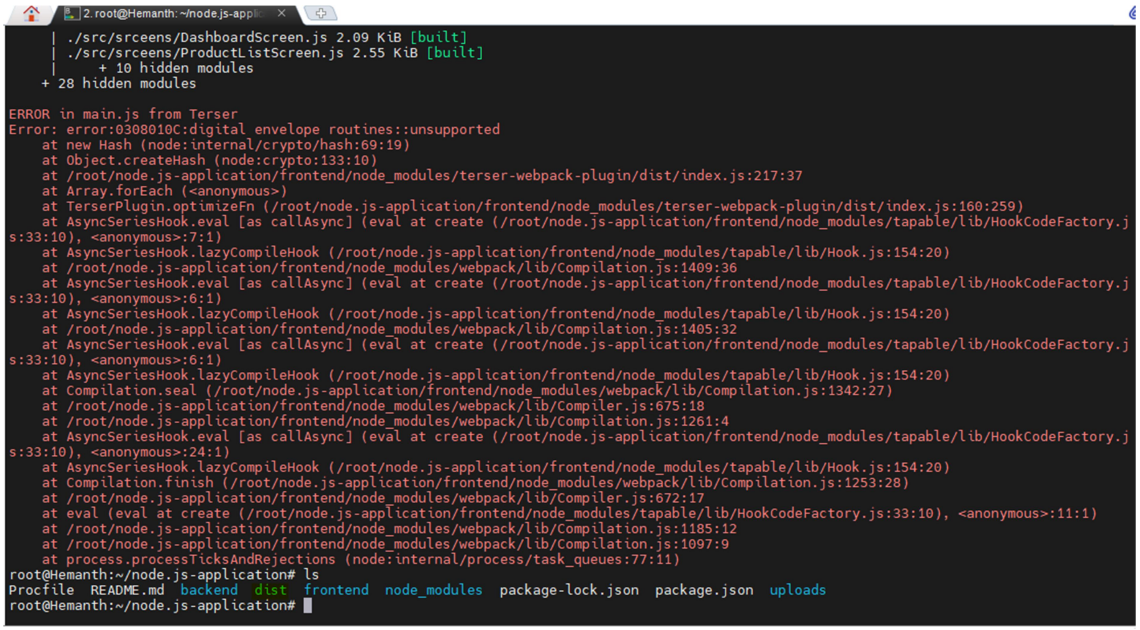


Step2: Enter into the transferred file with **'cd <Directory Name>'** command and by using **'npm i'** command install the compatibilities for the application files and the compatibilities can be stored in a directory named as 'node_modules'.

Note: Before using any npm, nodejs, pm2, serve tools your cursor location should be in the project or application directory.

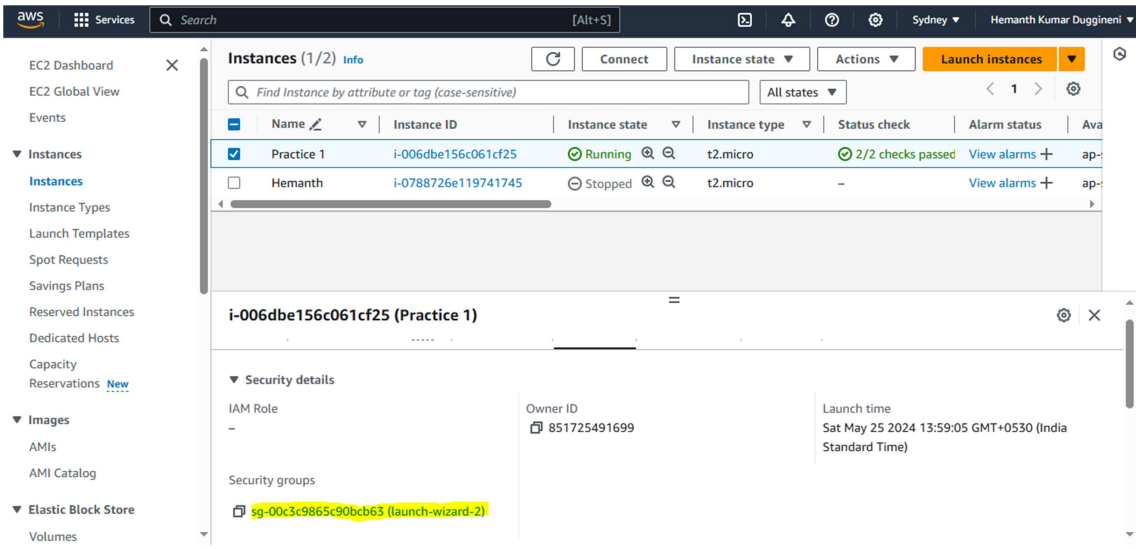


Step3: By using '**npm run build'** command we can build the application with the existing files, which can be stored in a directory named as 'dist' and by using **'cd'** command move your cursor to 'dist' directory.

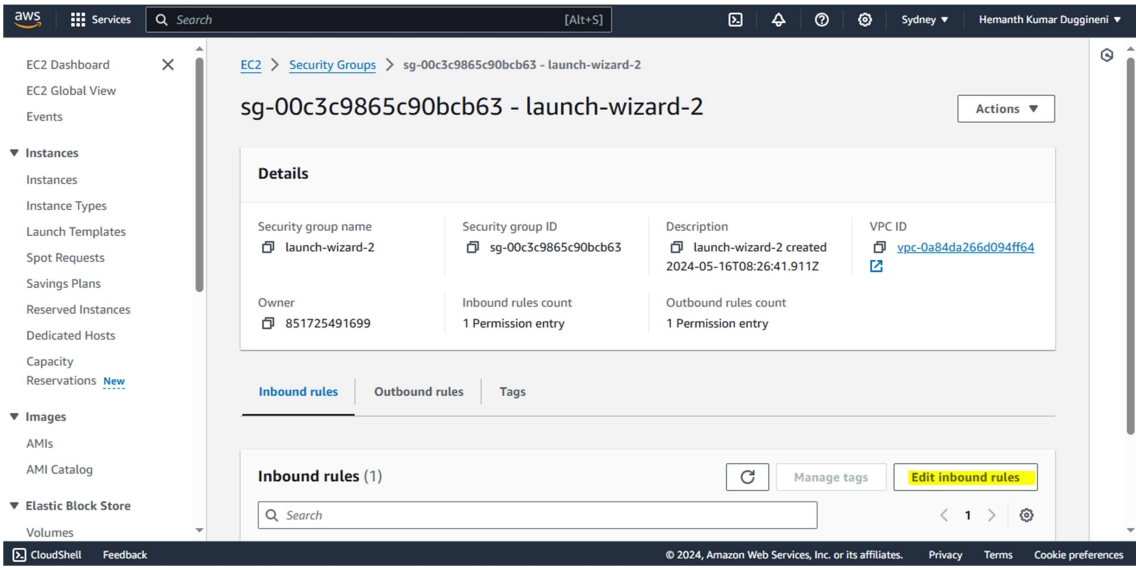


Step4: Now we have to allow the application port in Security Groups.

Go to Instances page and select your server, under Security section click on the link under Security groups.



Under Inbound rules section click on edit inbound rules.



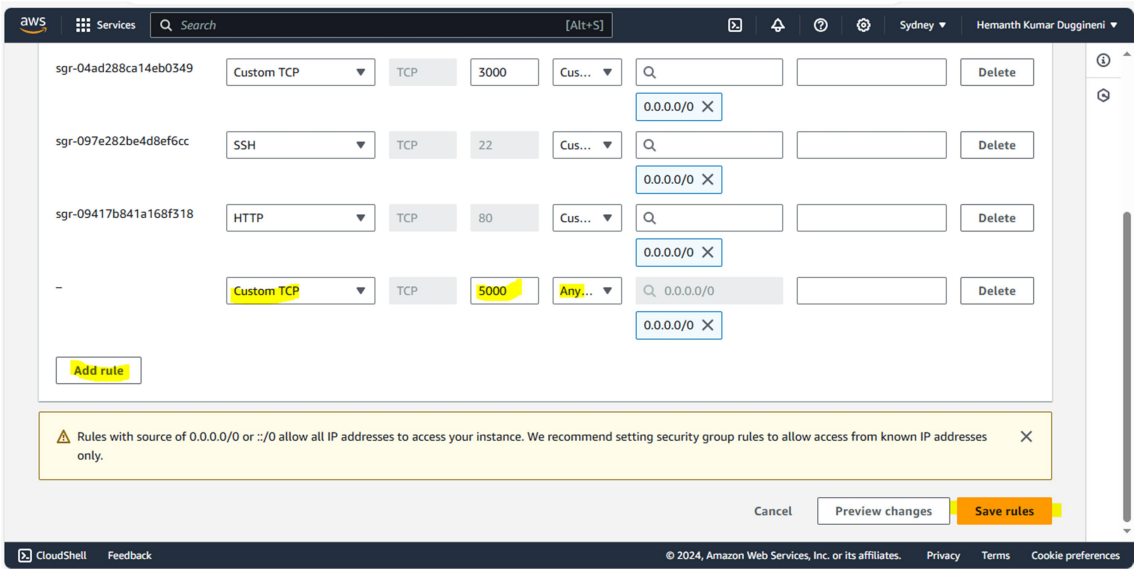
Click on Add rule and select

Type: Custom TCP (required protocol type)

Port Range: Required Port number

Source: Anywhere-IPv4 (required access network IP addresses)

And click on save rules.

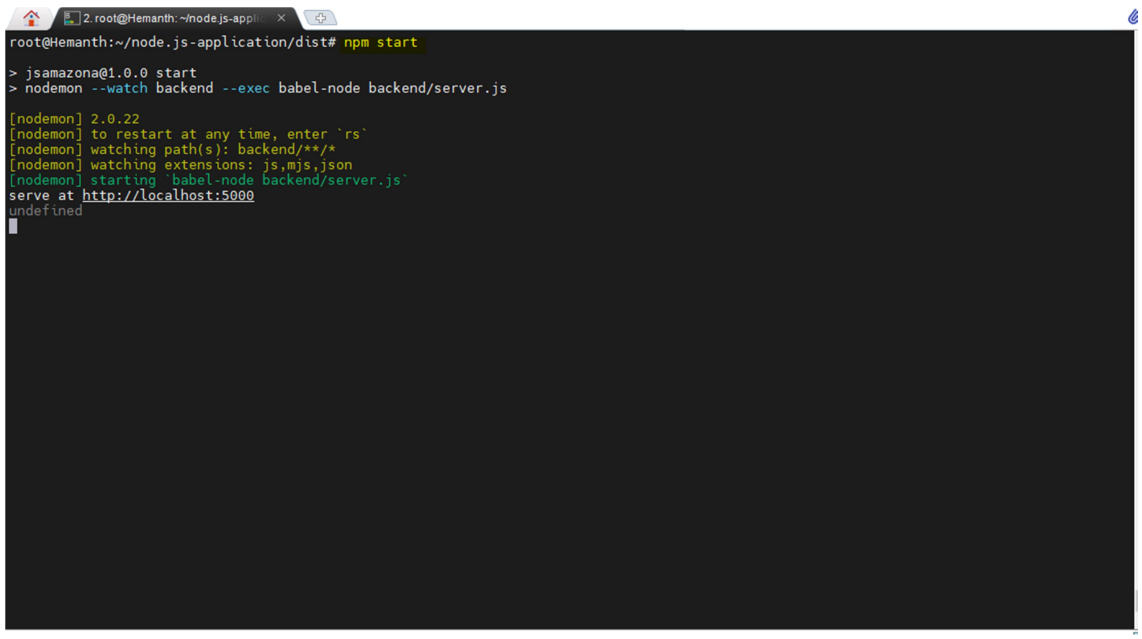


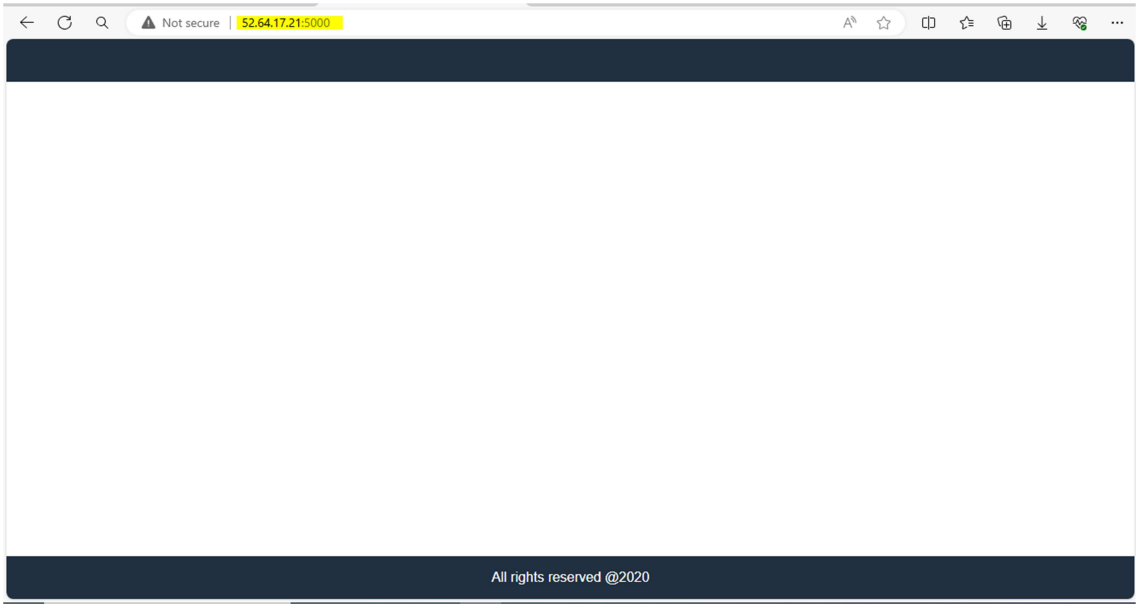
Up to here deployment of application is done.

To run the deployed application there are two ways:

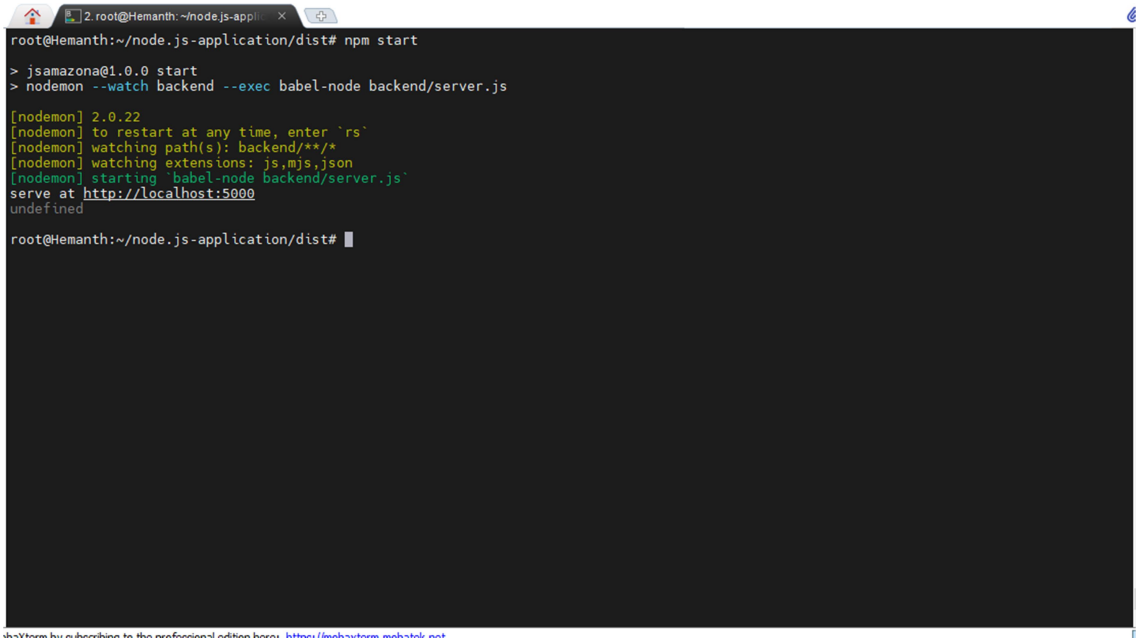
Temporary Run: By using **'npm start'** we can run our application temporarily up to we click on 'ctrl+c' in the server.

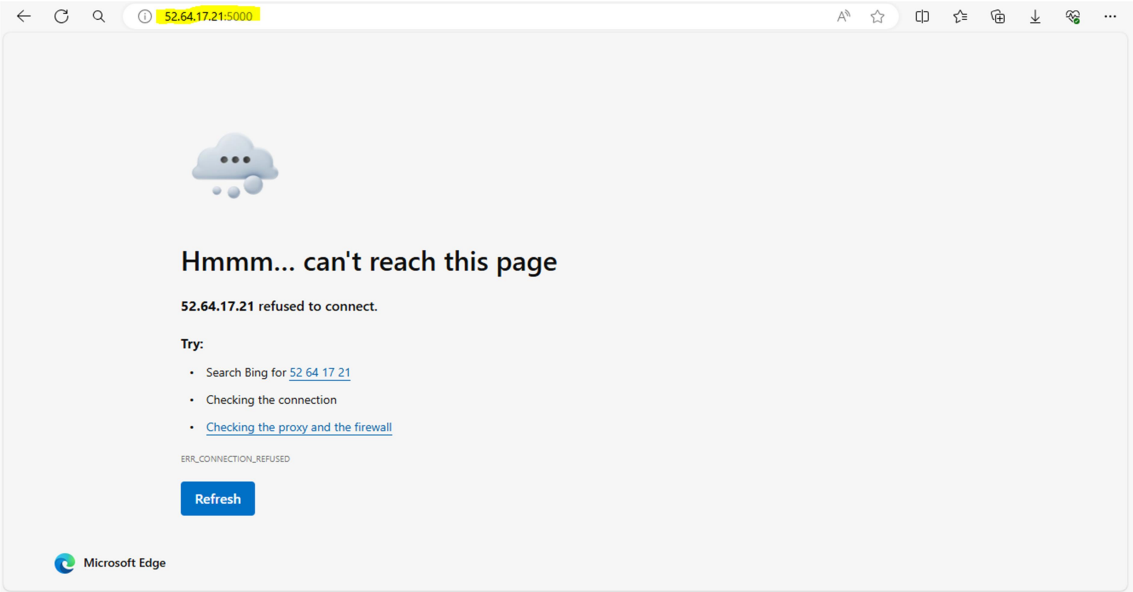
After passing the command **'npm start'** the application will start running with the access point.



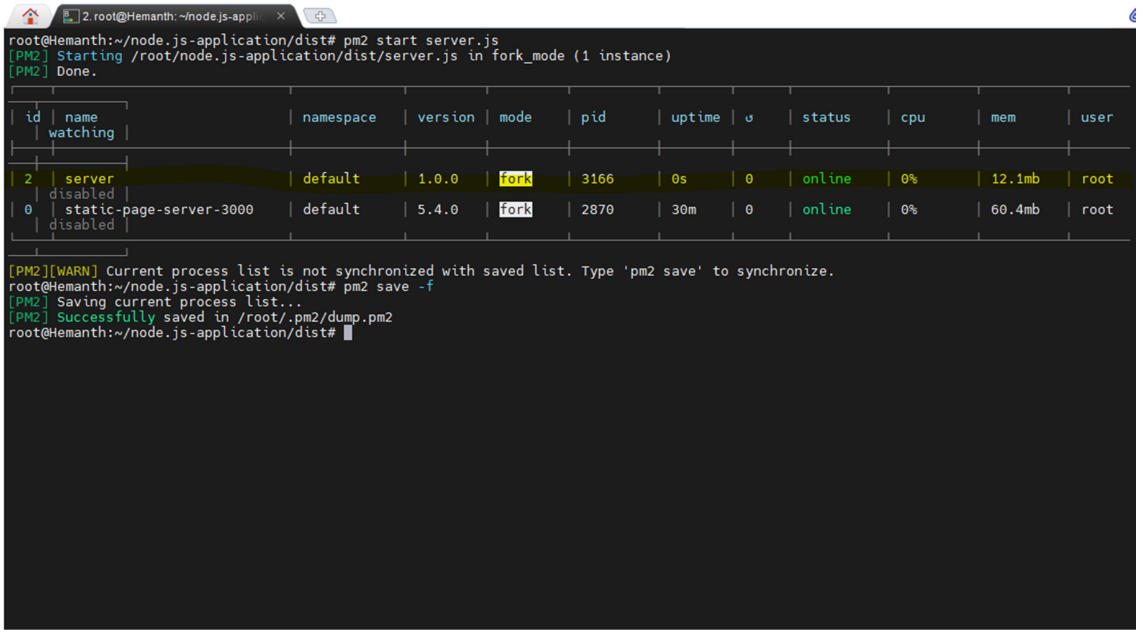


After entering ‘ctrl+c’, the application will stop running with access point.

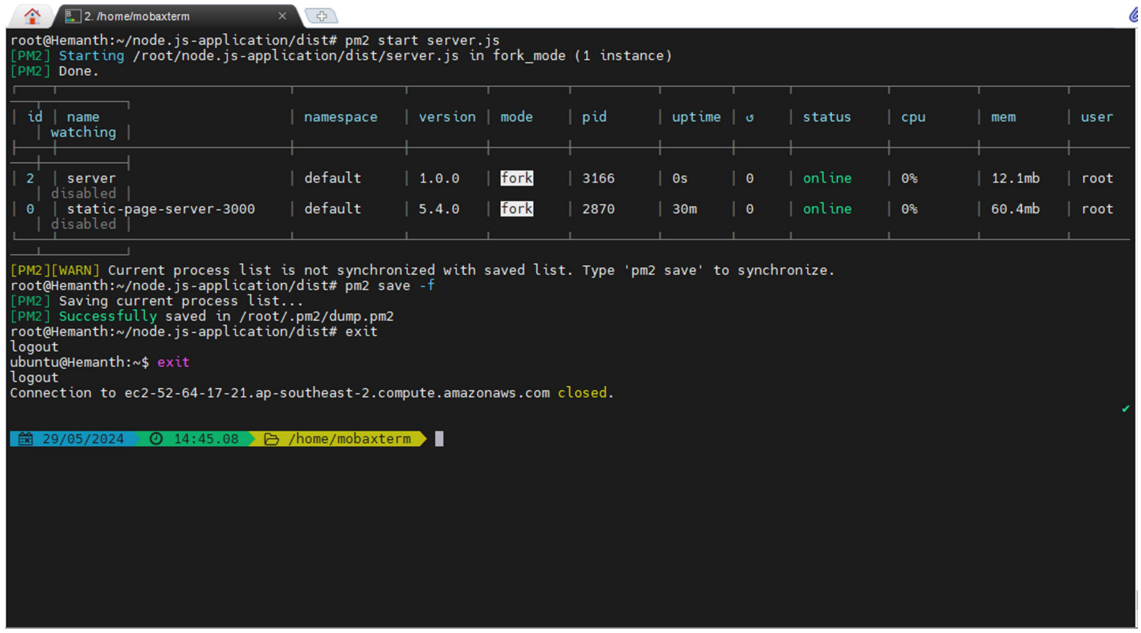


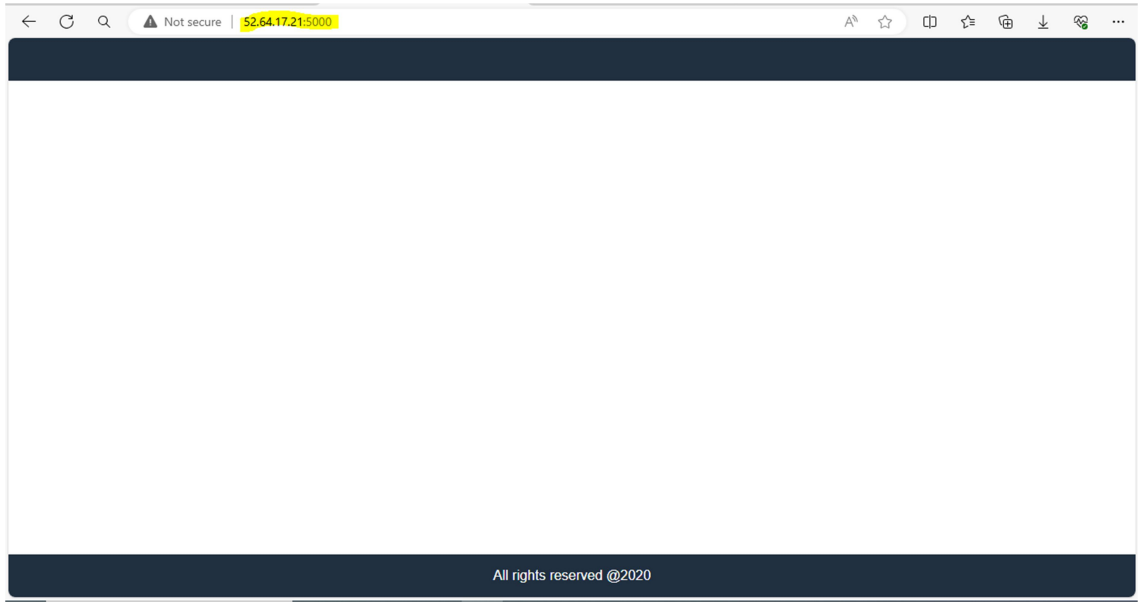


Permanent Run: By using **'pm2 start <Application's server.js file>'** command start and run the application in the server. By using **'pm2 save –f'** command save the running programs forcefully to run the application even if user got logged out from the server.



Now I am logged out from the server and even though the access point is working fine.


2. /home/mobaxterm
root@Hemanth:~/node.js-application/dist# pm2 start server.js
[PM2] Starting /root/node.js-application/dist/server.js in fork_mode (1 instance)
[PM2] Done.
id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
2	server	default	1.0.0	fork	3166	0s	0	online	0%	12.1mb	root	disabled
0	static-page-server-3000	default	5.4.0	fork	2870	30m	0	online	0%	60.4mb	root	disabled
[PM2][WARN] Current process list is not synchronized with saved list. Type 'pm2 save' to synchronize.
root@Hemanth:~/node.js-application/dist# pm2 save -f
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2
root@Hemanth:~/node.js-application/dist# exit
logout
ubuntu@Hemanth:~$ exit
logout
Connection to ec2-52-64-17-21.ap-southeast-2.compute.amazonaws.com closed.
29/05/2024 14:45.08 /home/mobaxterm


Not secure
52.64.17.21:5000
All rights reserved @2020