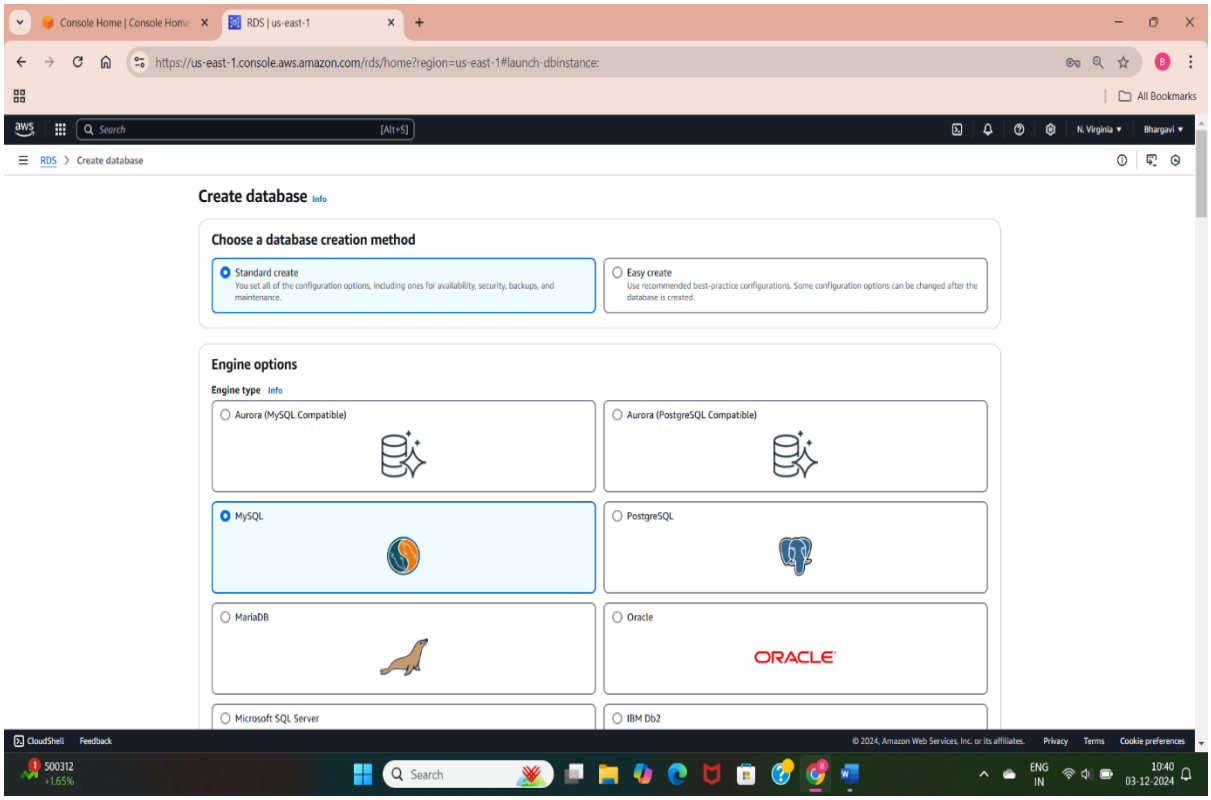
PROJECT-2

DEPLOY WORDPRESS USING DIFFERENT METHODS

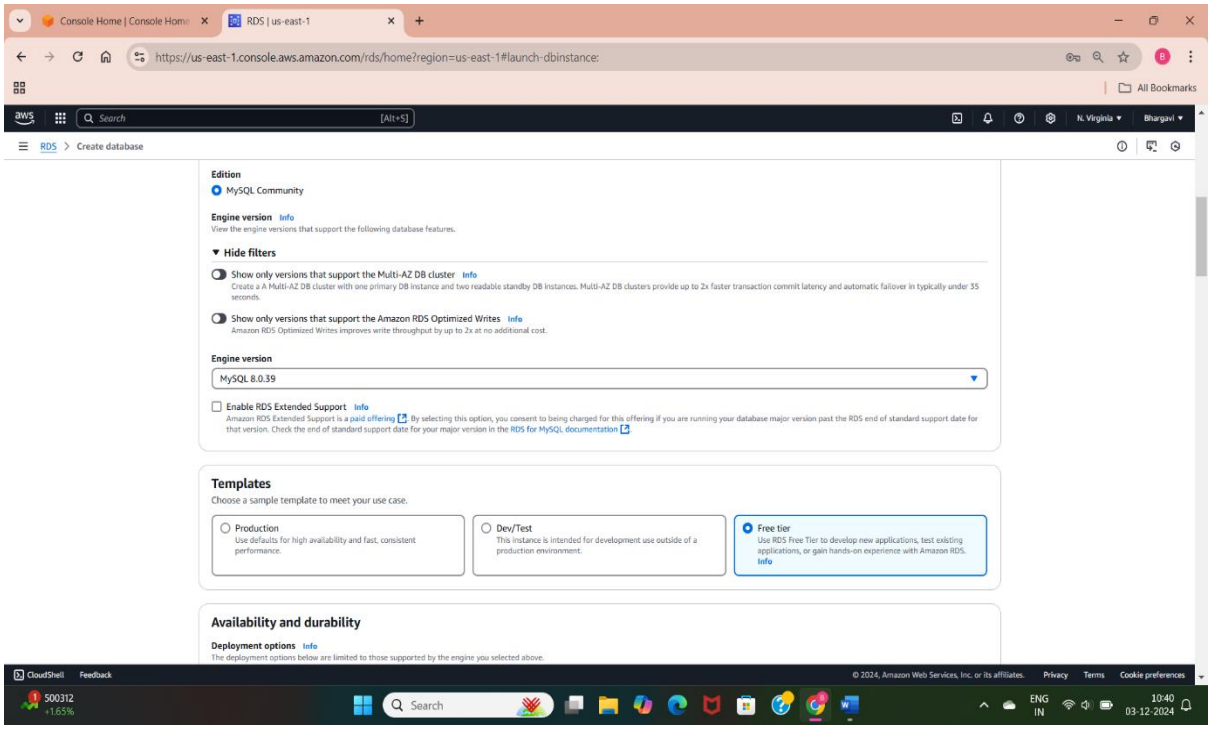
METHOD-1: CREATE MYSQL DATABASE BY USING RDS:

RDS is a fully managed service which means just we need to focus on storing and managing data not on creating and maintaining , aws will manage these tasks.

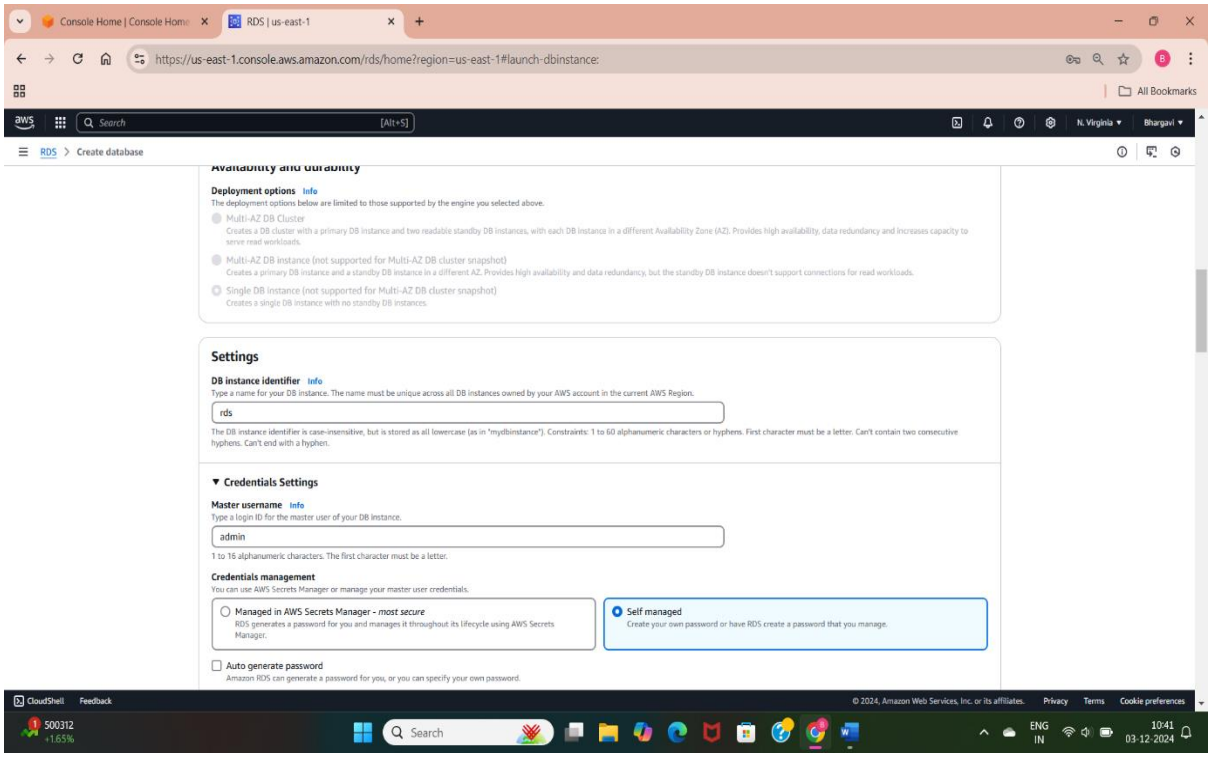
- ❖ Now first go to your aws account and login with credentials. After go to RDS service and open that service.
- ❖ Now create a mysql database by using RDS service for that go into the RDS services and click on create database.



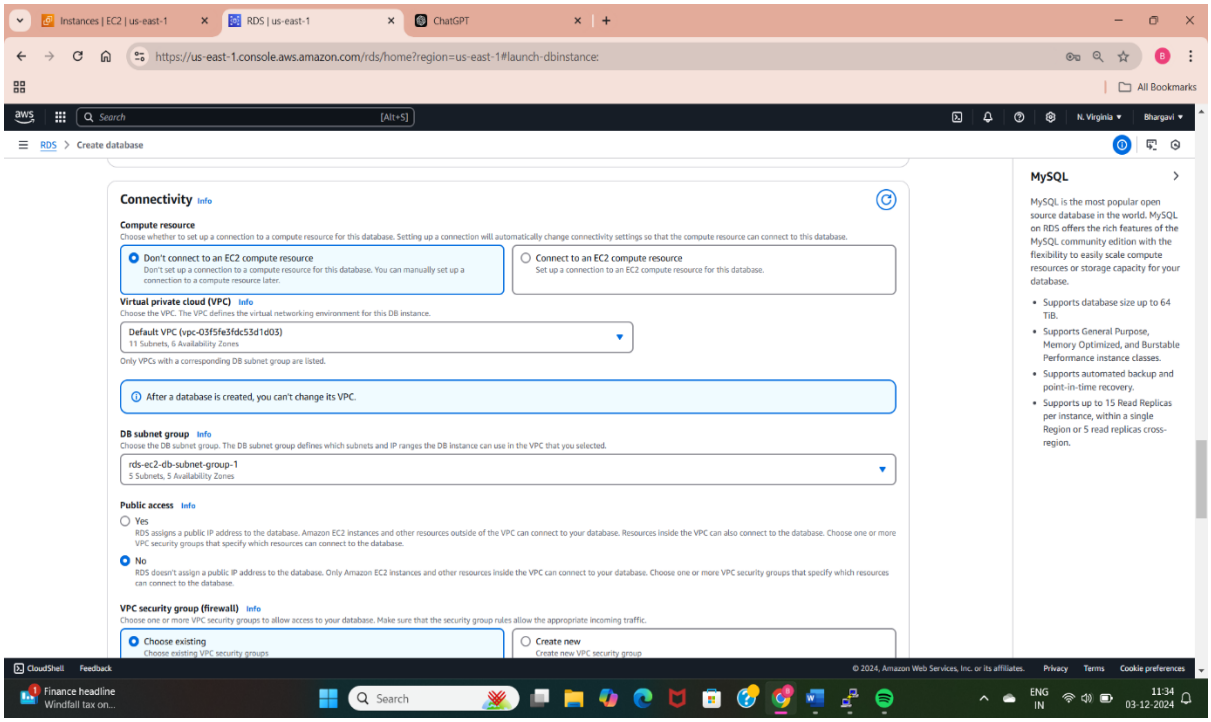
- ❖ Now select the database creation method I select here standard create method because by selecting easy create method then it disables the free tier template.
- ❖ Now select the database engine with version but I select the mysql database engine because it's a huge usage database in real time organizations.



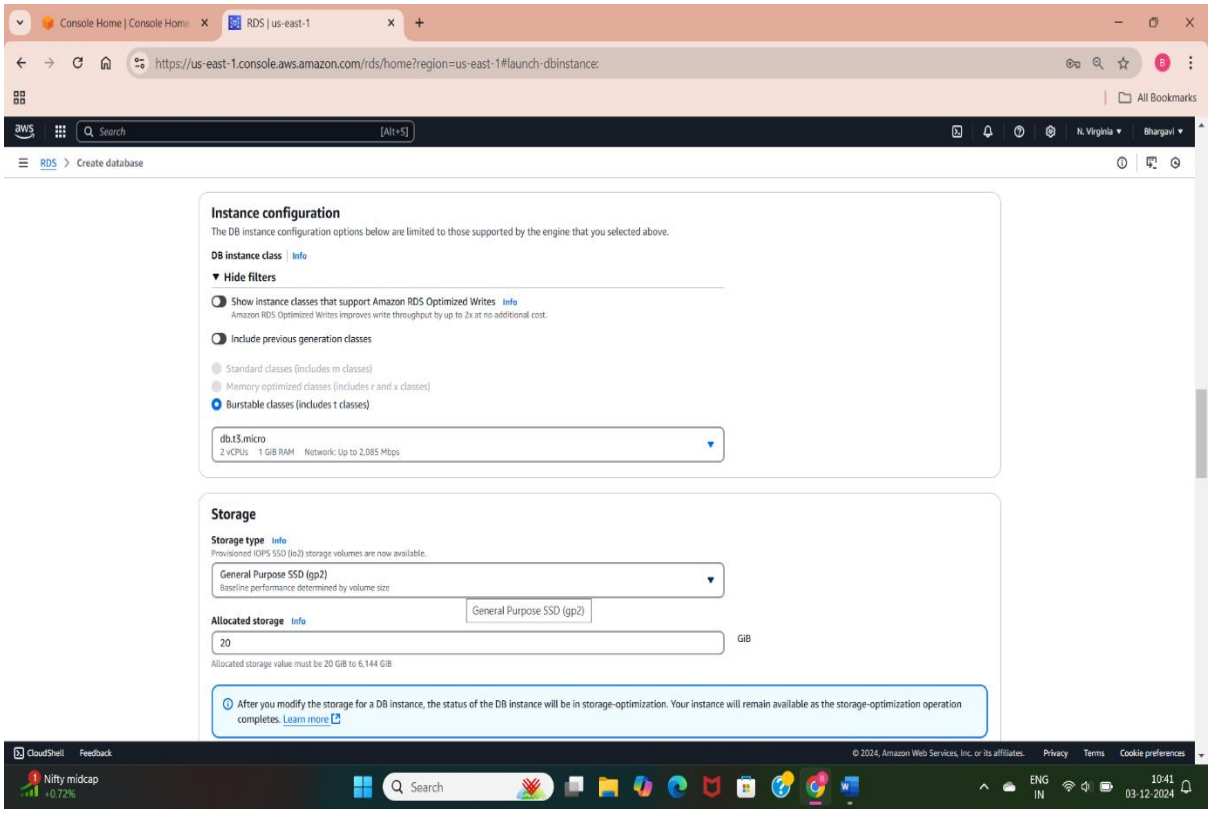
- I have selected the version “8.0.39” for MYSQL engine
- Now select the Template as free tier and by selecting this free tier it disable the availability zone selecting option.



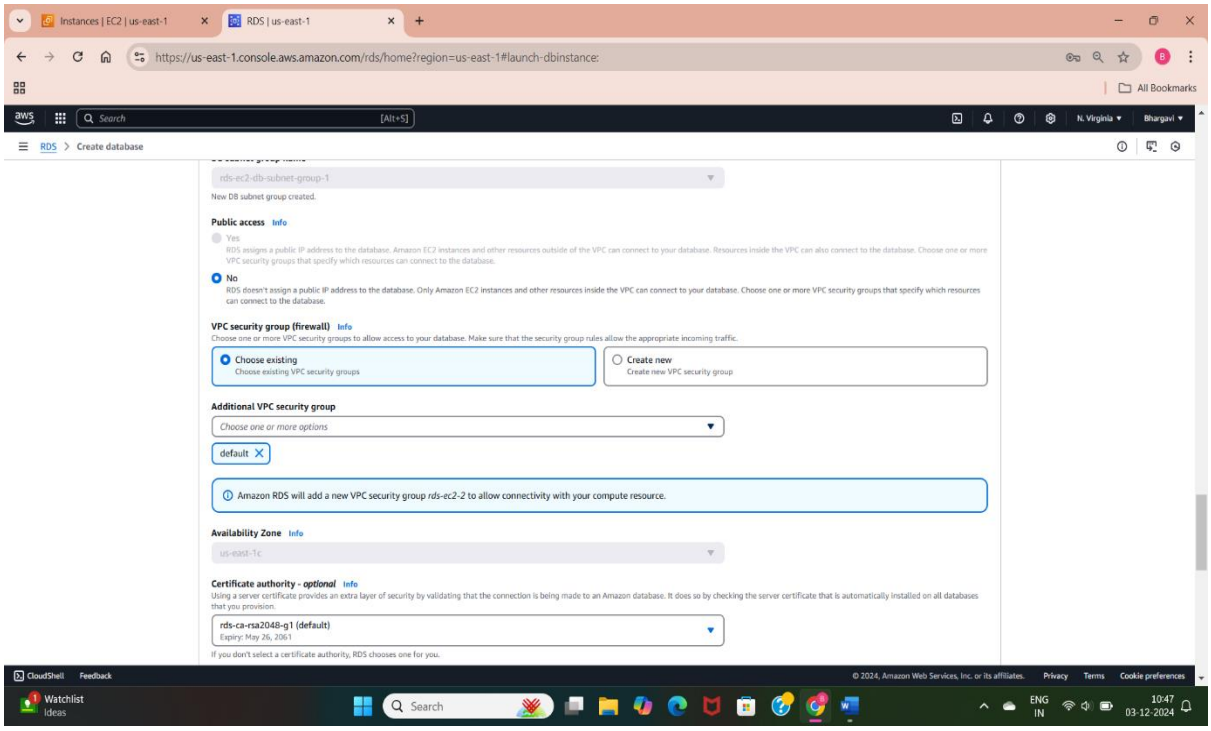
- Now give the some name to your database and give username and passwords as credentials for your database access.



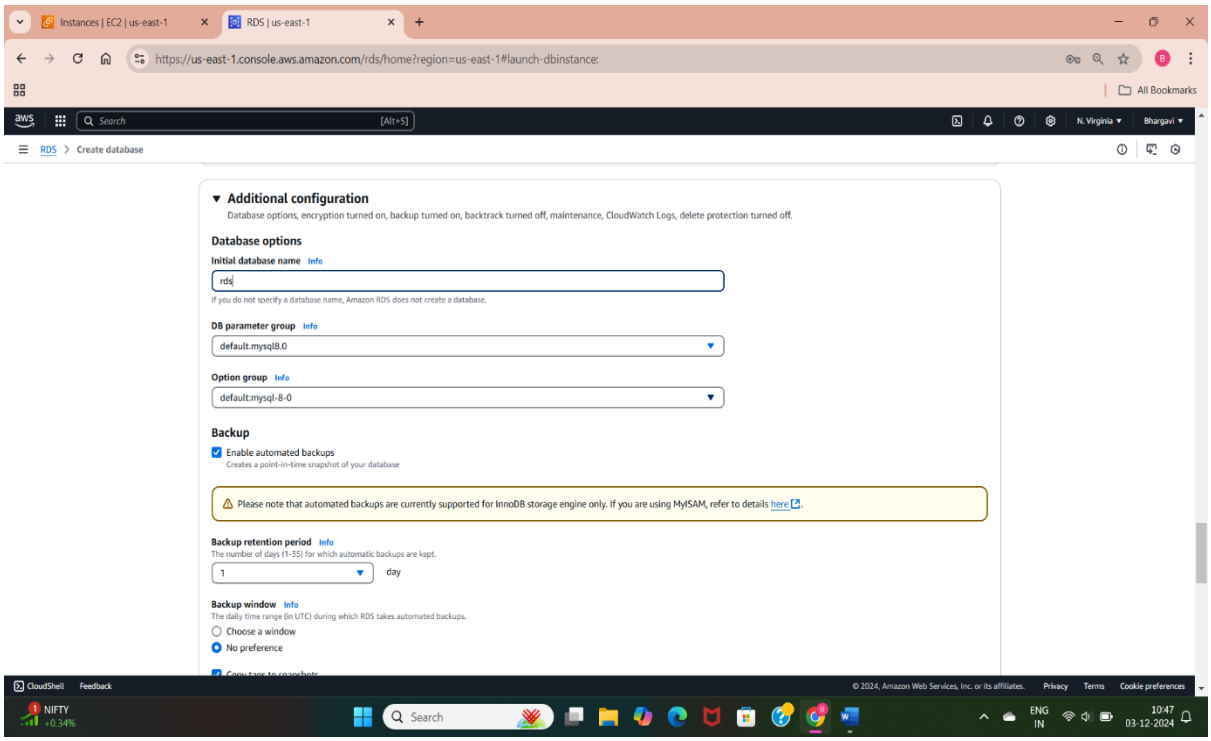
❖ I have selected the option with the Don't connect to an EC2 compute resource as connectivity, selected a default VPC for internet access.



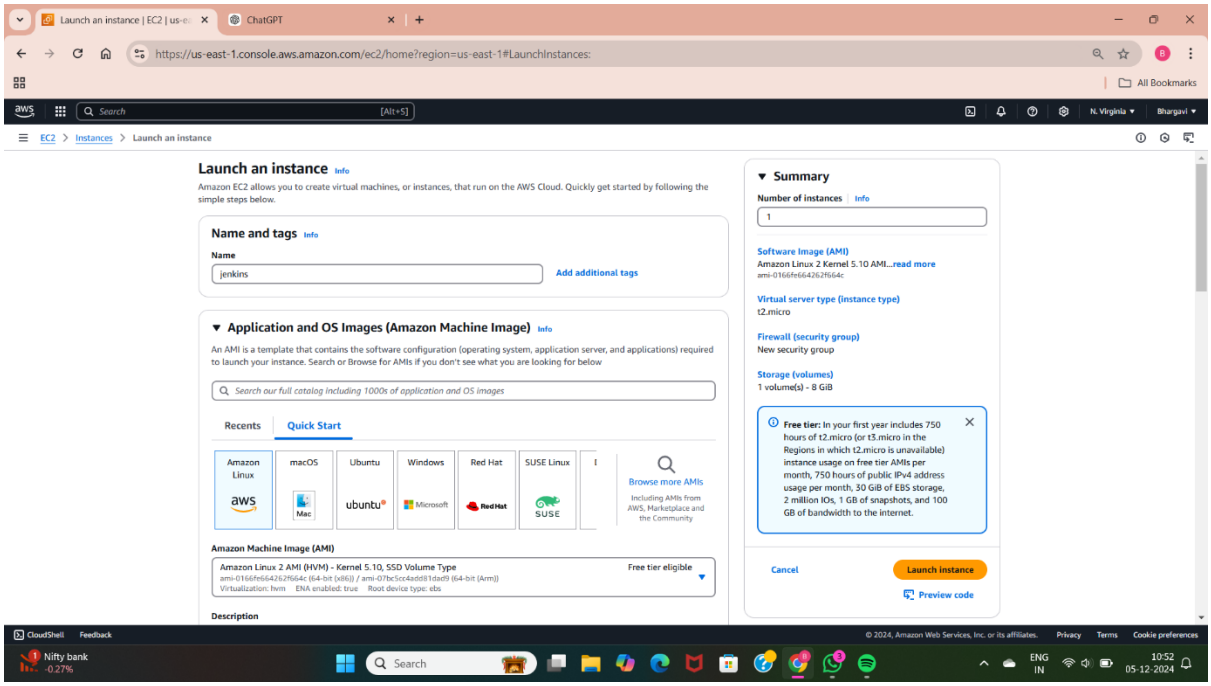
❖ Now select the storage type as General Purpose SSD(gp2) and enter the storage values as maximum(1000GB) and minimum(20GB).



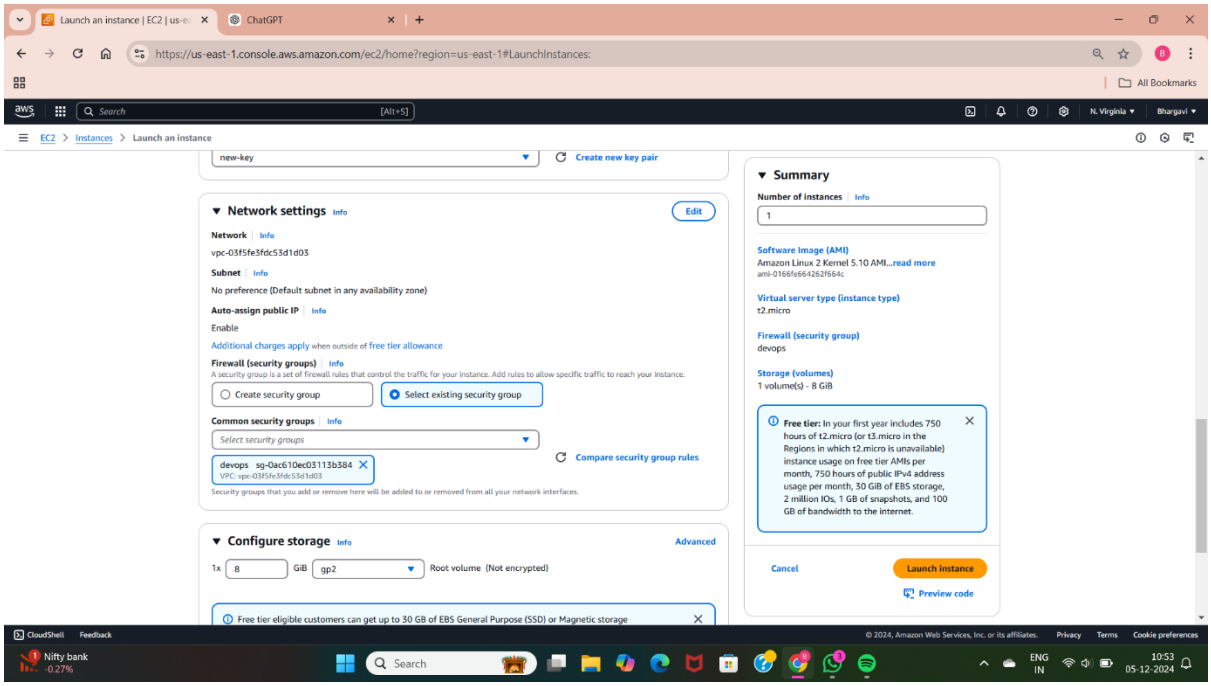
- Now select your created VPC or select default VPC and it automatically select the database subnet group.



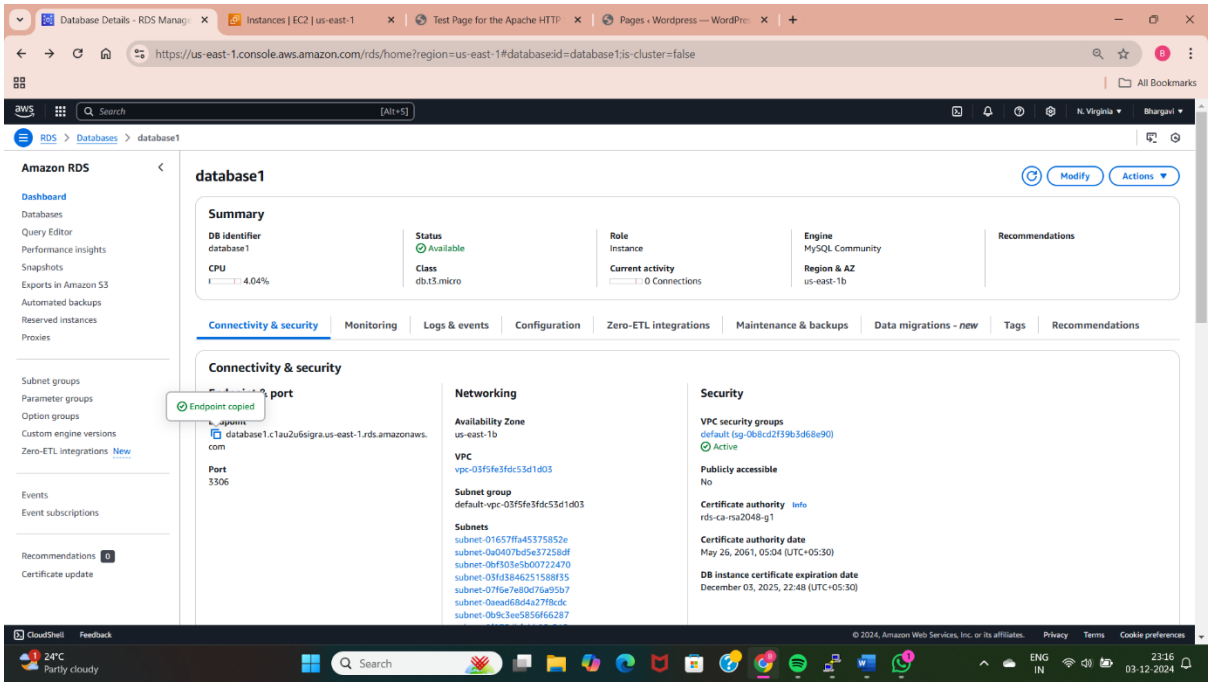
- Now give the name of the database which you give at the stage of DB instance identifier enter same name here.
- Now click on create database button and it will create the MYSQL database.



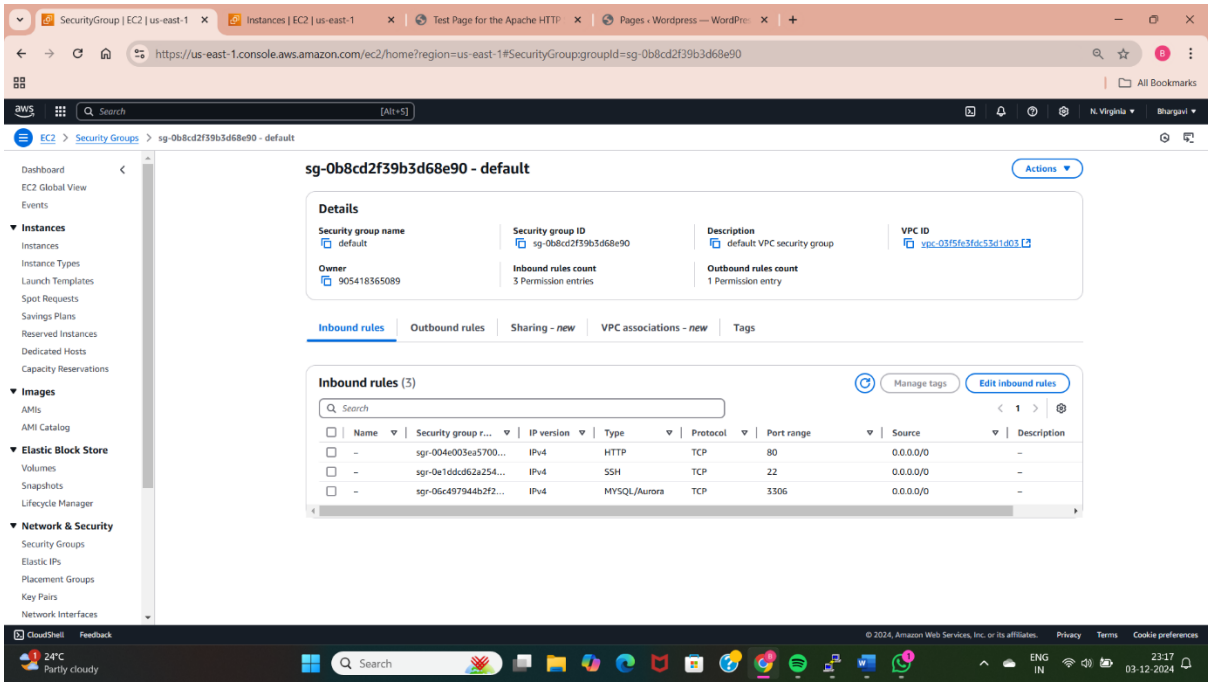
- Now create a EC2 instance by allowing the specific inbound rules with port numbers 22 for SSH, 80 for HTTP, and 3306 for MYSQL.



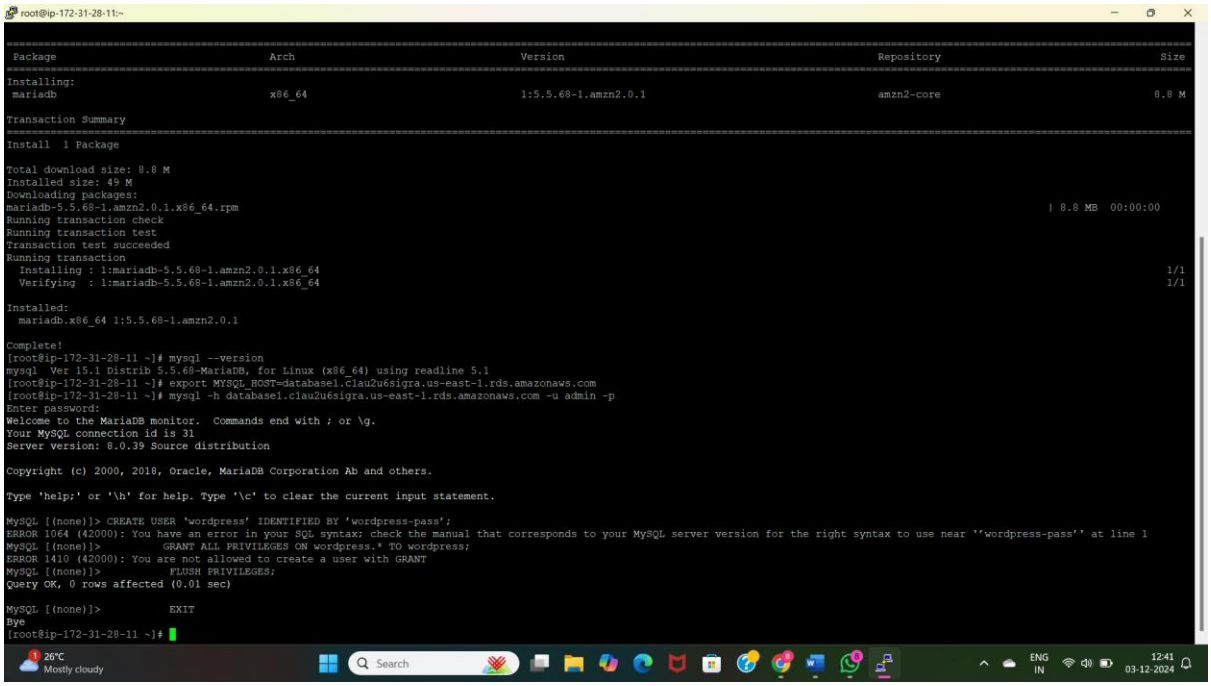
- Now create the EC2 instance by selecting EC2 services and launch the instance by selecting Amazon linux-2 version and giving the security group with SSH(22) and HTTP(80).
- Now connect the virtual server through the Git bash as shown in below.
- Now update the linux version by using command as <sudo yum -y update> and install the mysql by using the command as <sudo yum -y install mysql>.



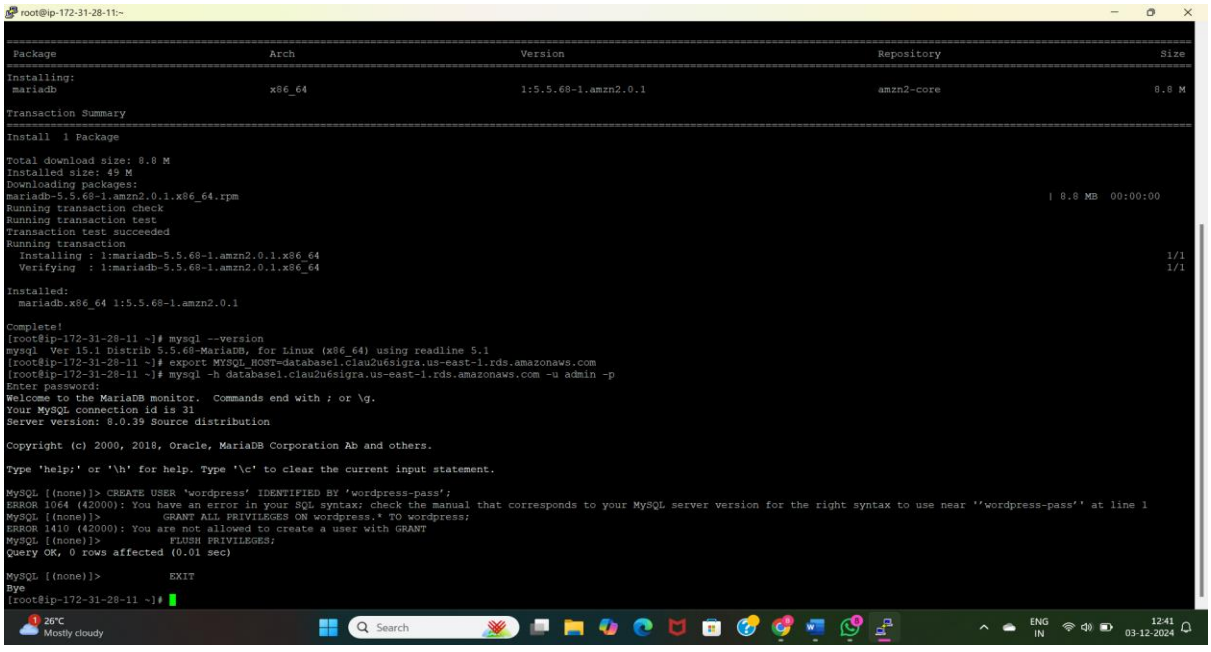
- Now to allow certain traffic from EC2 instance into RDS database for that go to RDS services and select your created database.



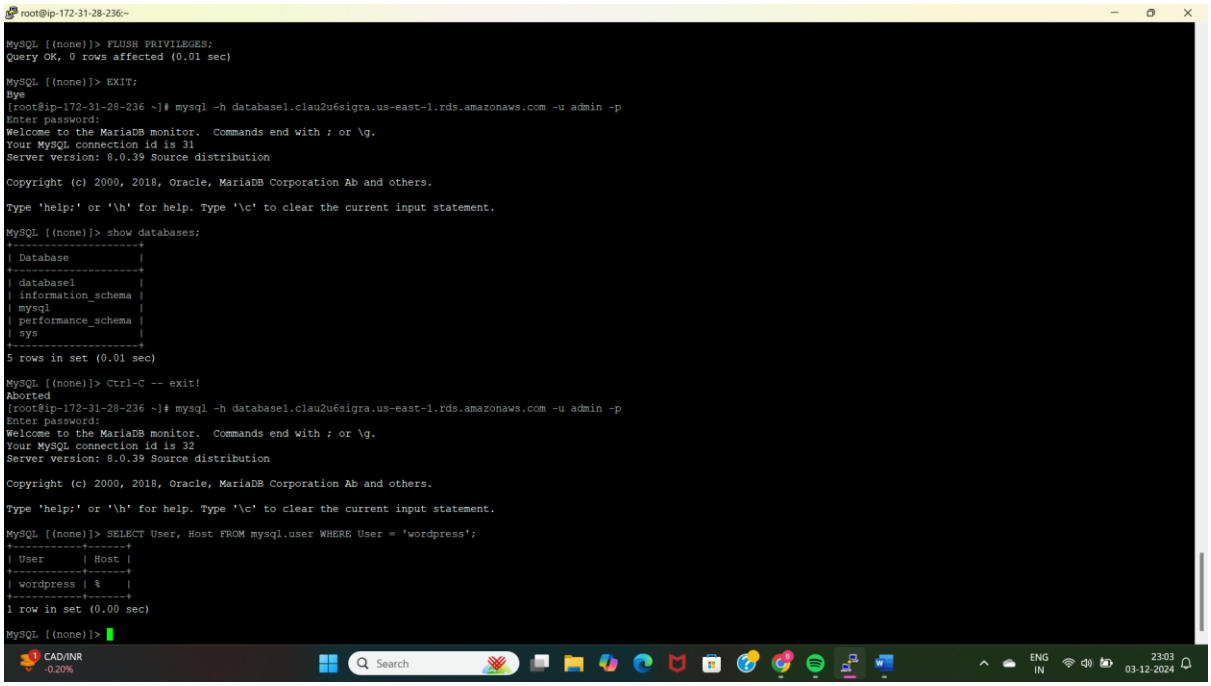
- Now go inside the created database and go to the security under this option there is a security group id click on that.
- Now go to inbound rules and click on the edit inbound rules.
- Now go to source option click on dropdown option select the EC2 instance security group id and click save rules.



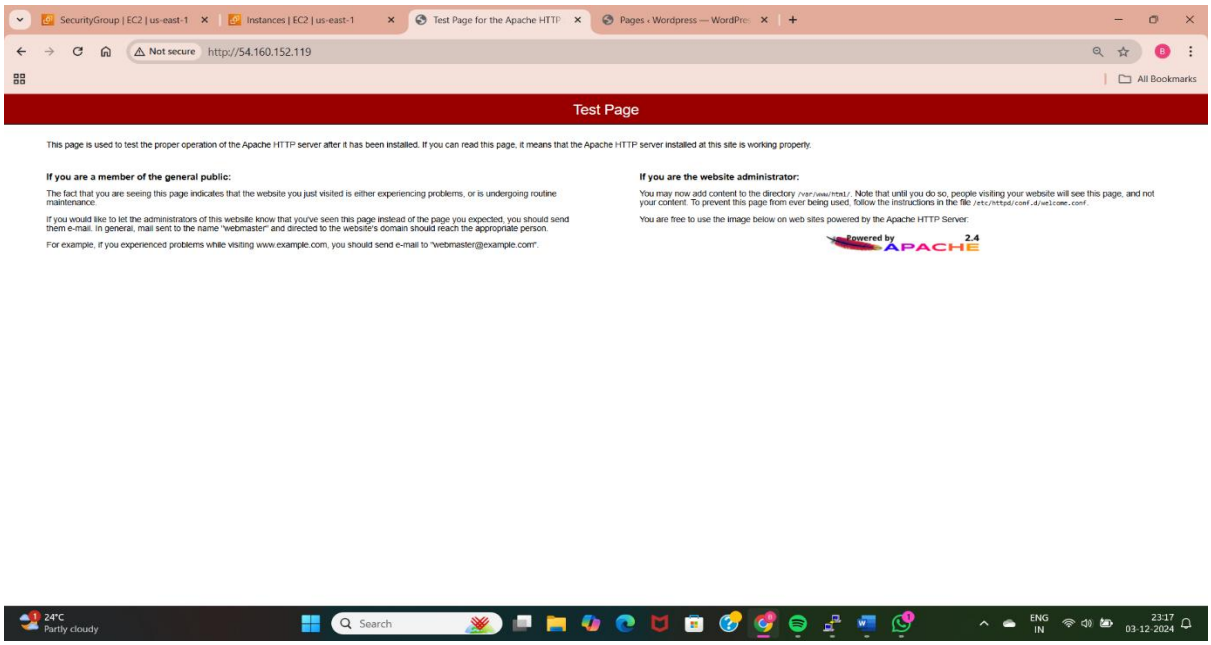
- Now access the mysql database by using the command as “export MYSQL_HOST=<endpoint address>” for that go inside the created database and go to the Endpoint and select and copy the endpoint address.
- Now give the credentials of database by giving a command as “mysql –user=<username> --password=<password> database-name”.
- There id another command to access your database as “mysql -h <endpoint address > -u <user> -p” press eneter button it asks the password enter it and press the enter button.



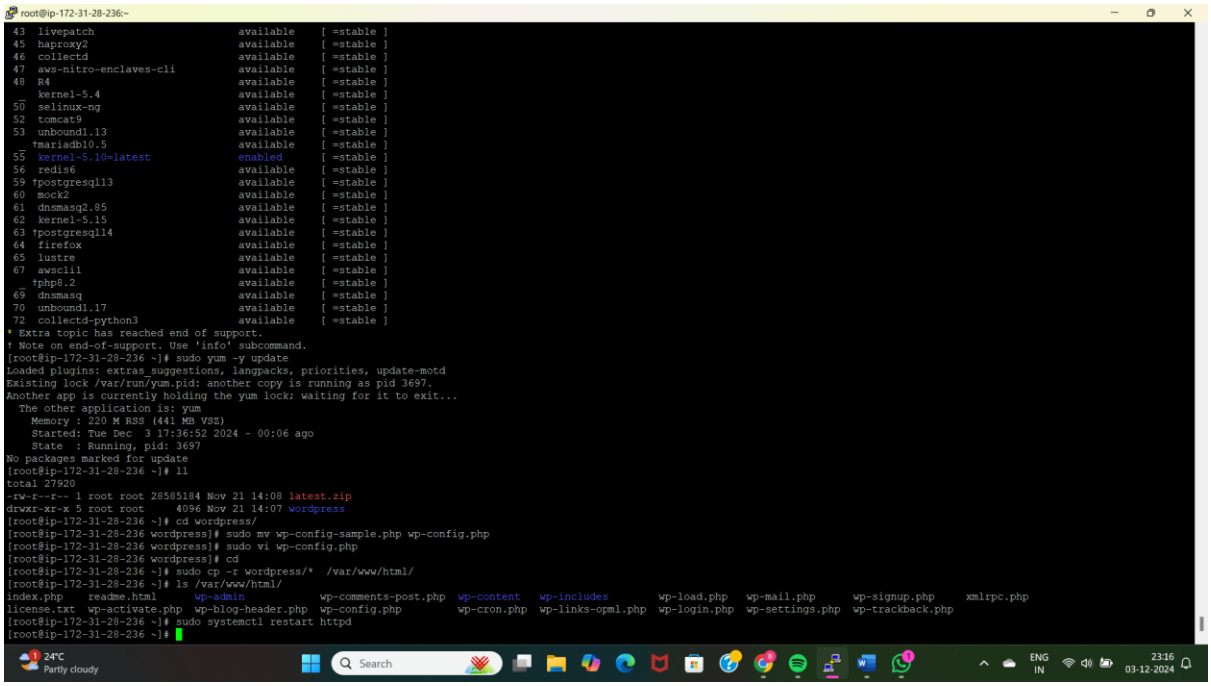
- Now create a database user for your wordpress application and give it permission to access the “wordpress” database. By using this commands as
  CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';
  GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress';
  FLUSH PRIVILEGES;
  EXIT;



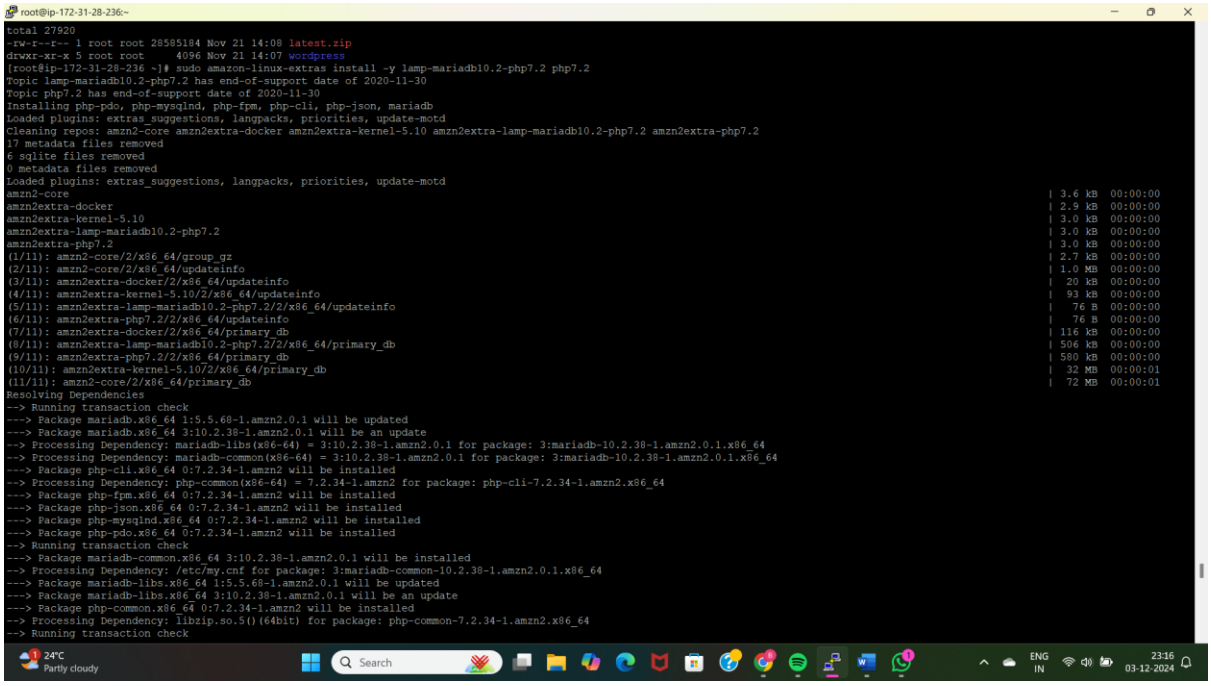
- To host wordpress application we need a apache web server httpd install that by giving a command as “sudo yum -y install httpd” and start and enable the httpd service by giving the commands as
- “sudo service httpd start” (or) “sudo systemctl start httpd”.
- “sudo chkconfig httpd on” (or) “sudo systemctl enable httpd”.



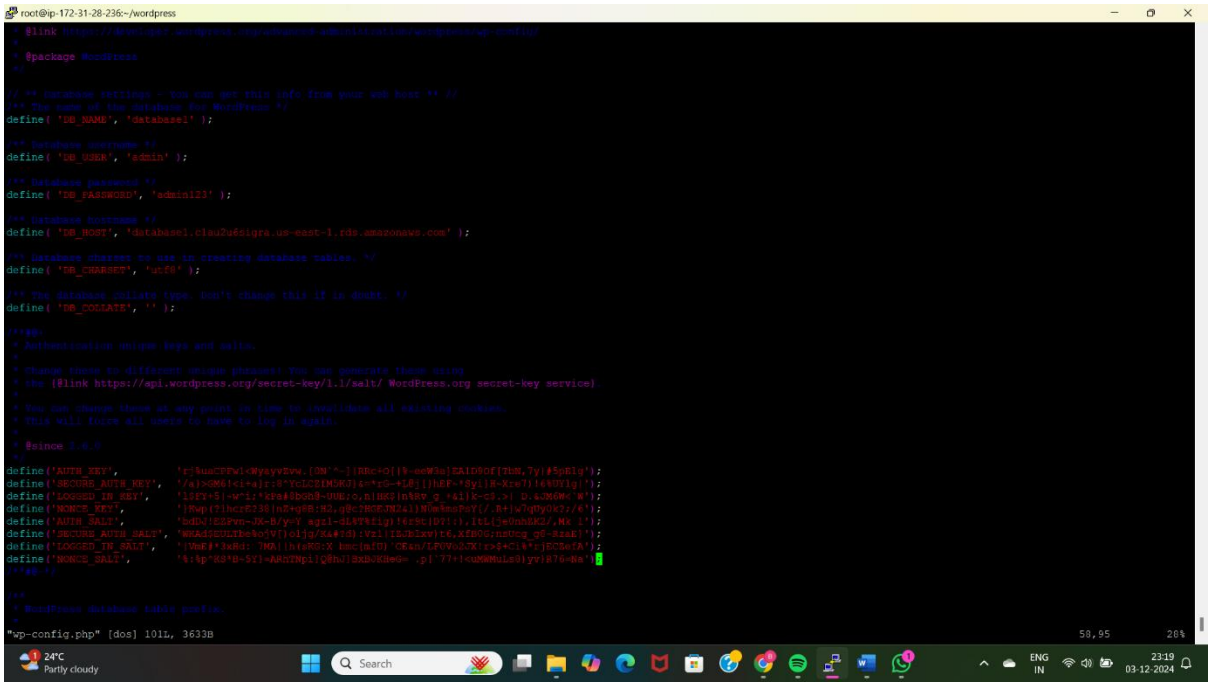
- Now go to EC2 instance and copy public ip and paste it on google browse it and check the official page of httpd is displaying or not.



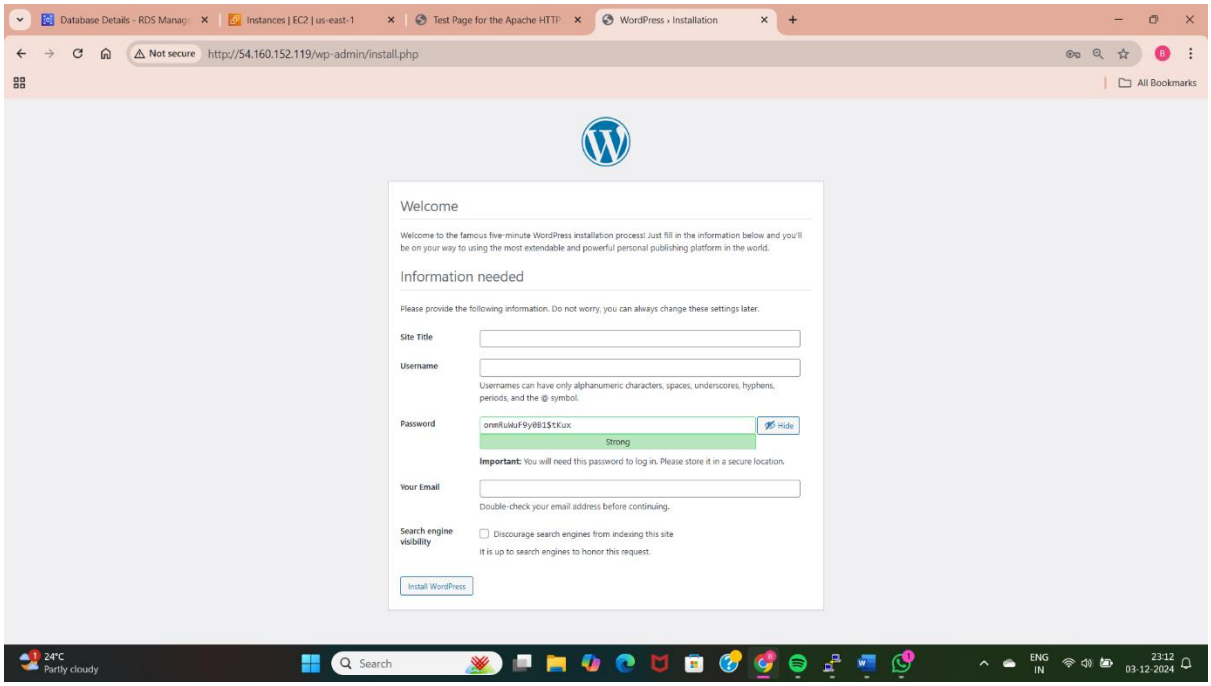
- Now go to browser and search as download word press and click on proper link and select and copy the address link of word press download file and paste that along with wget command in git bash as “wget <address link of word press download file?”
- It gives the zip file to unzip that file by using a command as “unzip <zipfile>”



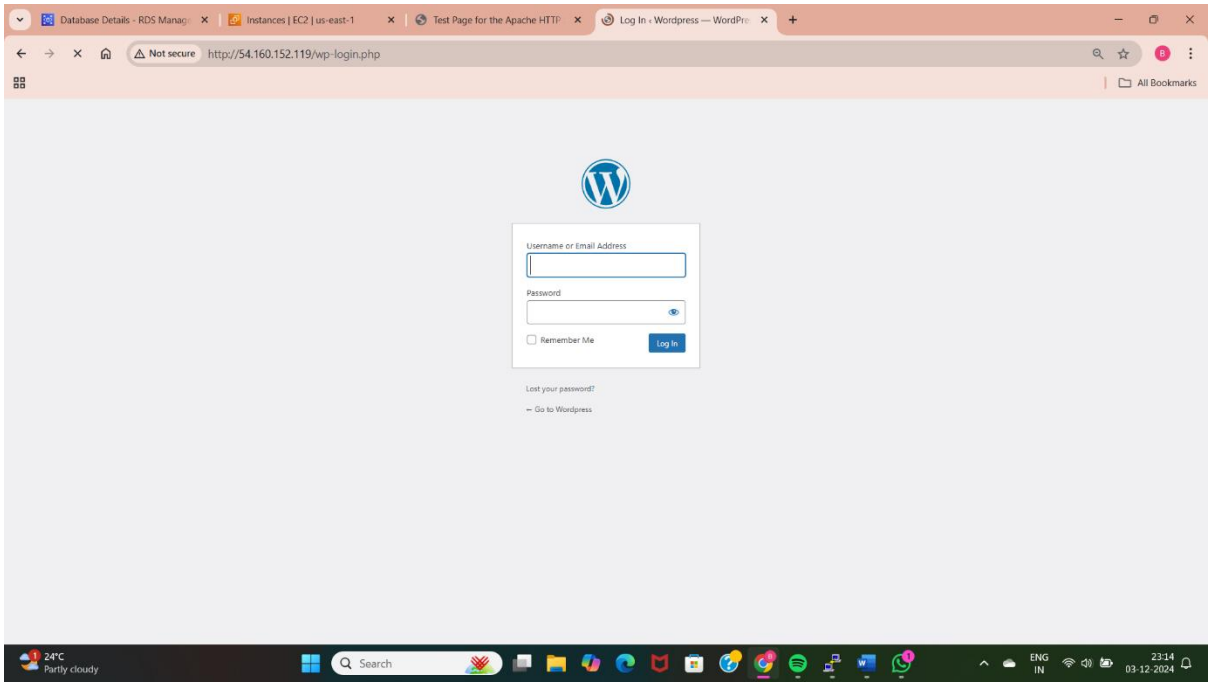
- To run wordpress web application, you have to install run time of word press web application as php language with the following command <sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2> otherwise update your EC2 instance with the following command as “<sudo yum -y update and sudo yum -y upgrade>.
- Now go inside the unzip directory by using command as “cd <unzip directory> “ and change the word press configuration file by giving command as “sudo mv wp-config-sample.php wp-config.php”.



- Now do some configurations in word press configuration file by giving database name, username, password and host name for that execute a command as “sudo vi wp-config.php”
- Search in google “Wordpress secret key generator”. And use that password.
- Now copy this wordpress directory to the document root directory to host your web application of wordpress by giving a command as
  “sudo cp -r <unzip file or wordpress directory>/* /var/www/html/”
  “ls /var/www/html/”



- Restart the httpd by giving a commands “systemctl restart httpd”.

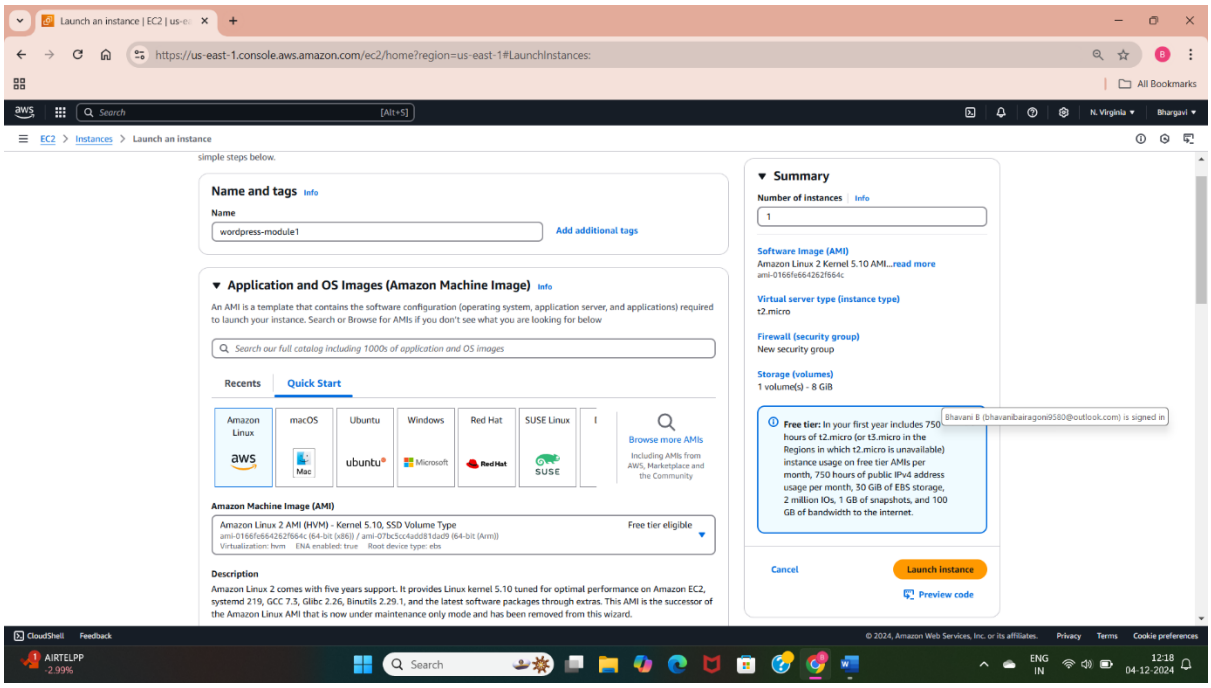


- ❖ Now go to EC2 instance and copy public ip and paste it on google browse it and check the official page of wordpress is displays.

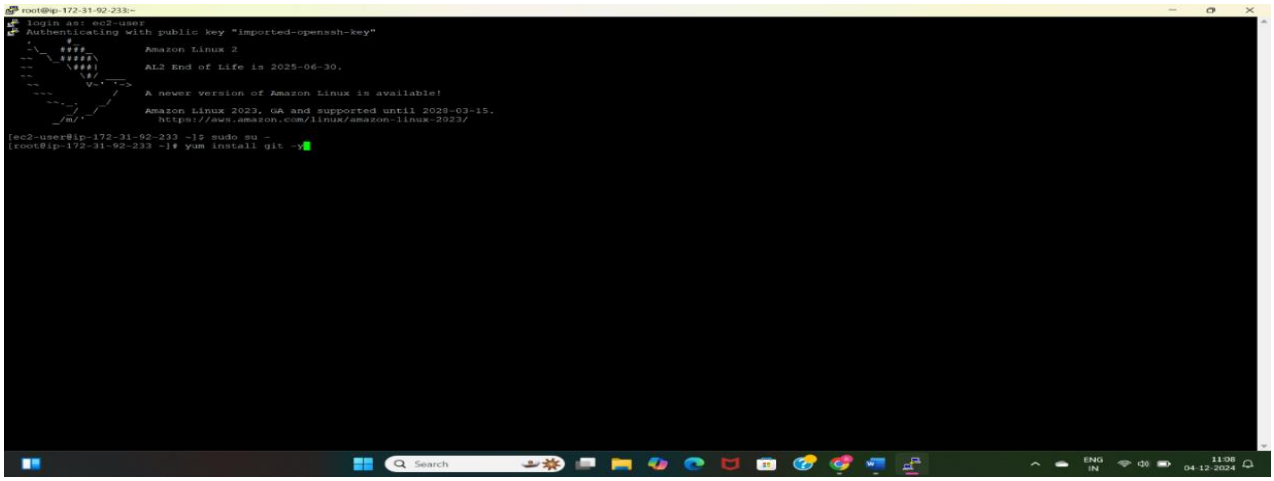
METHOD-2: CREATING AND LAUNCH AN AMAZON EC2 INSTANCE:

Amazon EC2 is virtual server which provides compute optimization for our applications by providing CPU, storage and RAM.

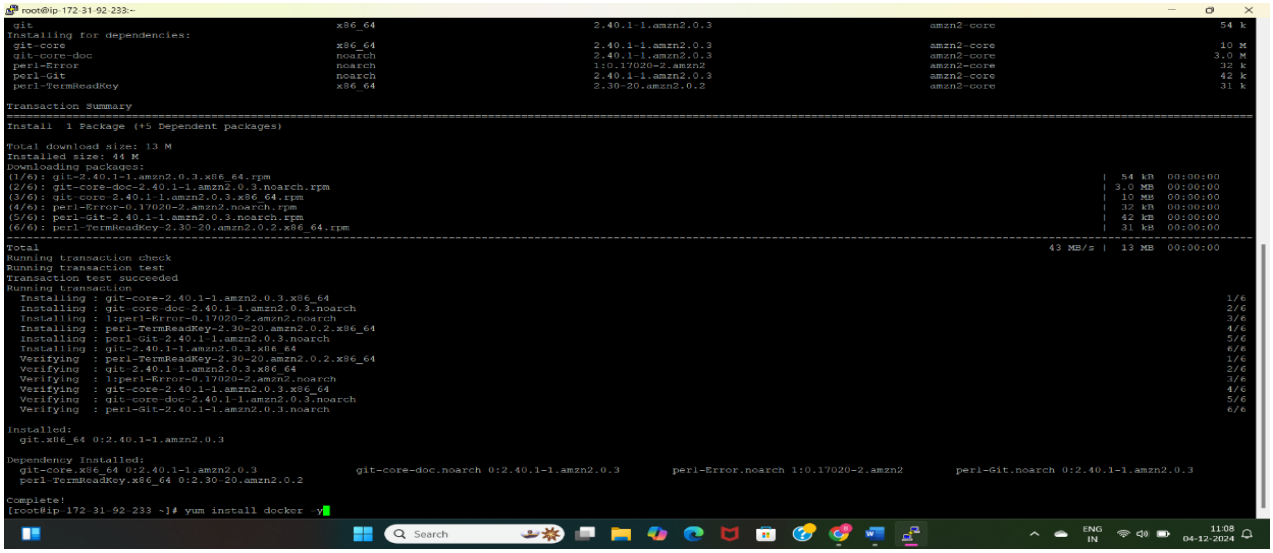
- ➲ First login to the AWS account with credentials.



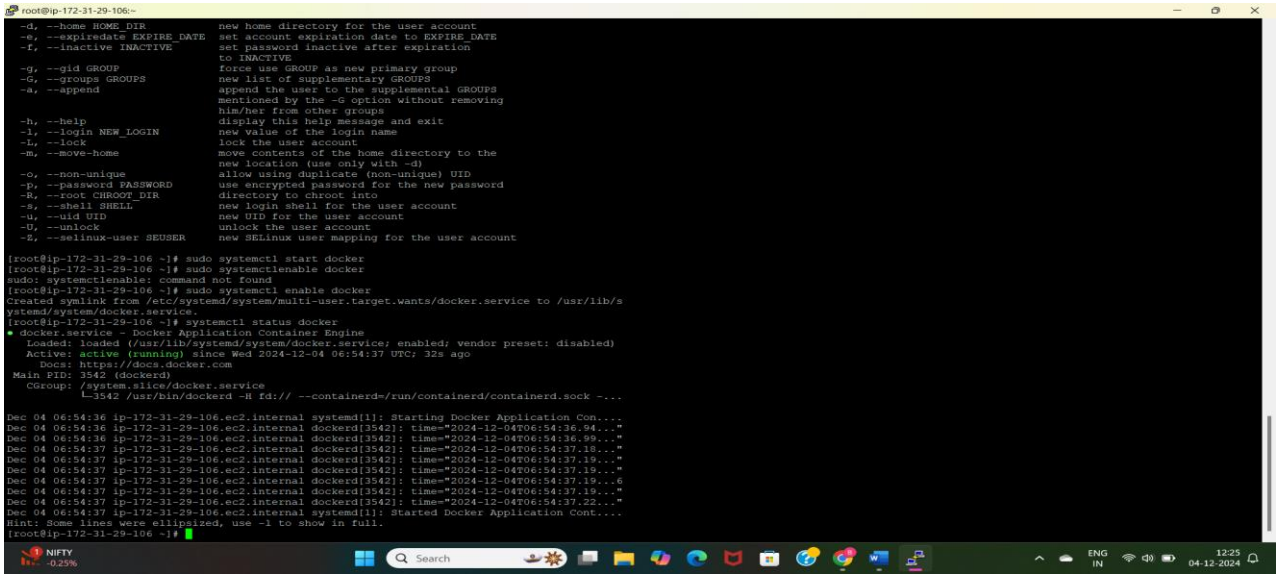
- ➲ Now give the name for EC2 instance.
- ➲ Select your AMI linux or ununtu 22.04 ltd free tier 22,80



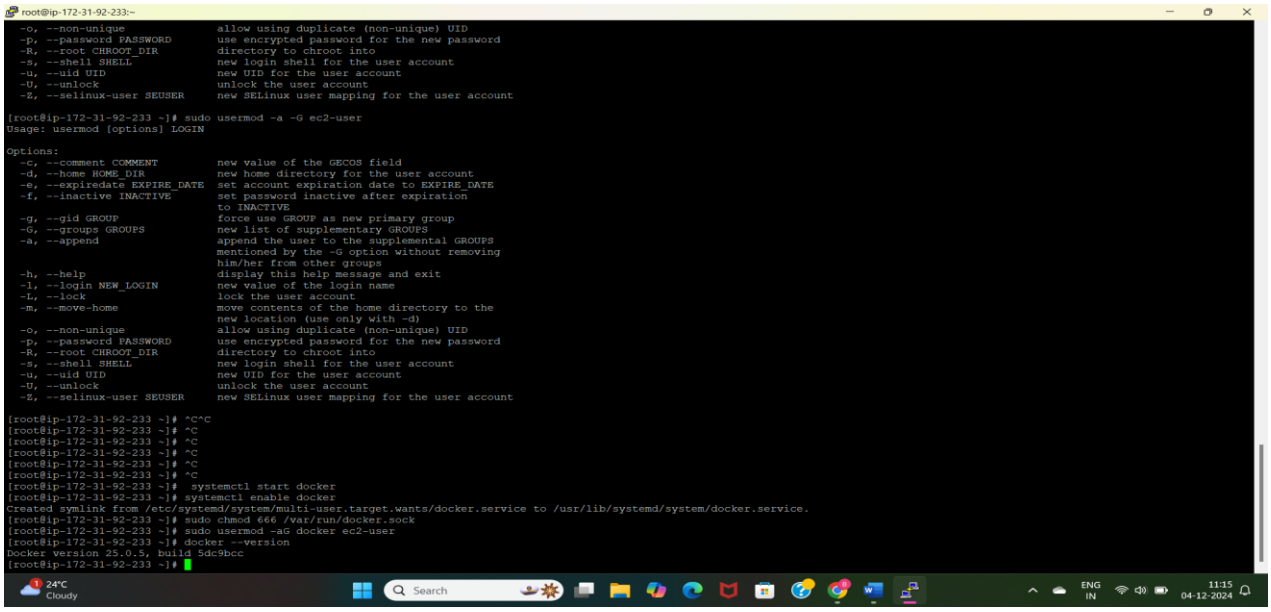
➲ Launch instance with any one of the terminal.



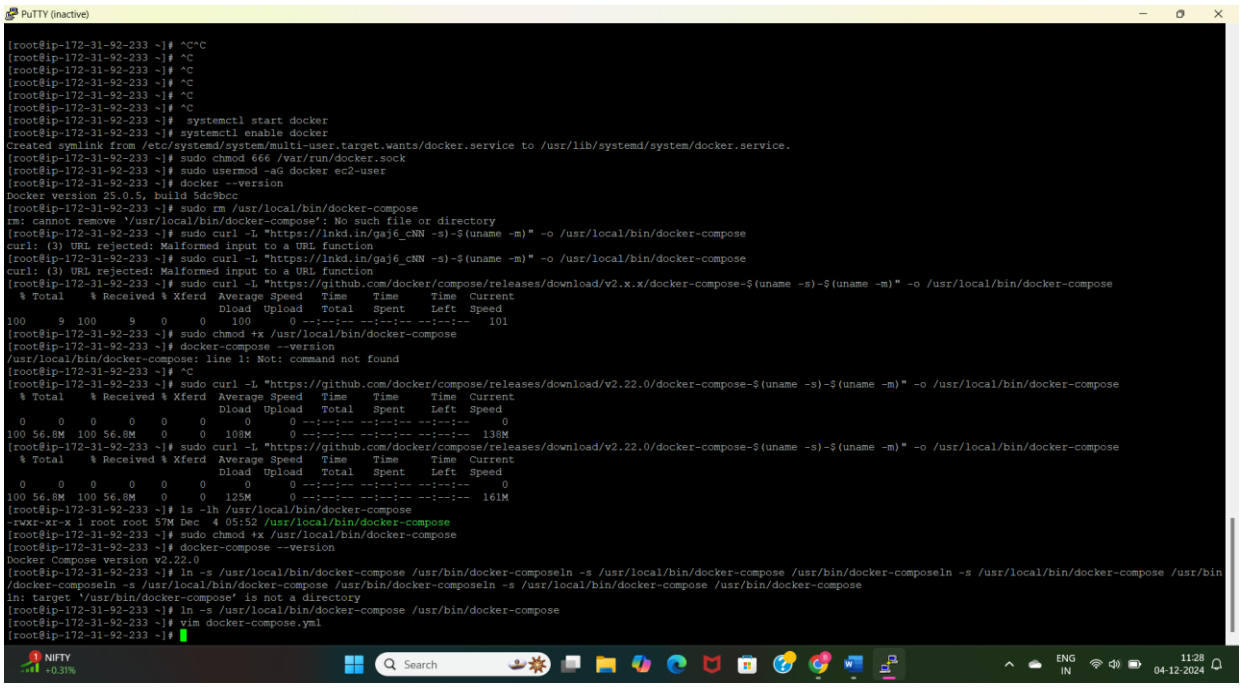
➲ Install git, docker and setup docker environment.



➲ And provide permissions for admin socket file.

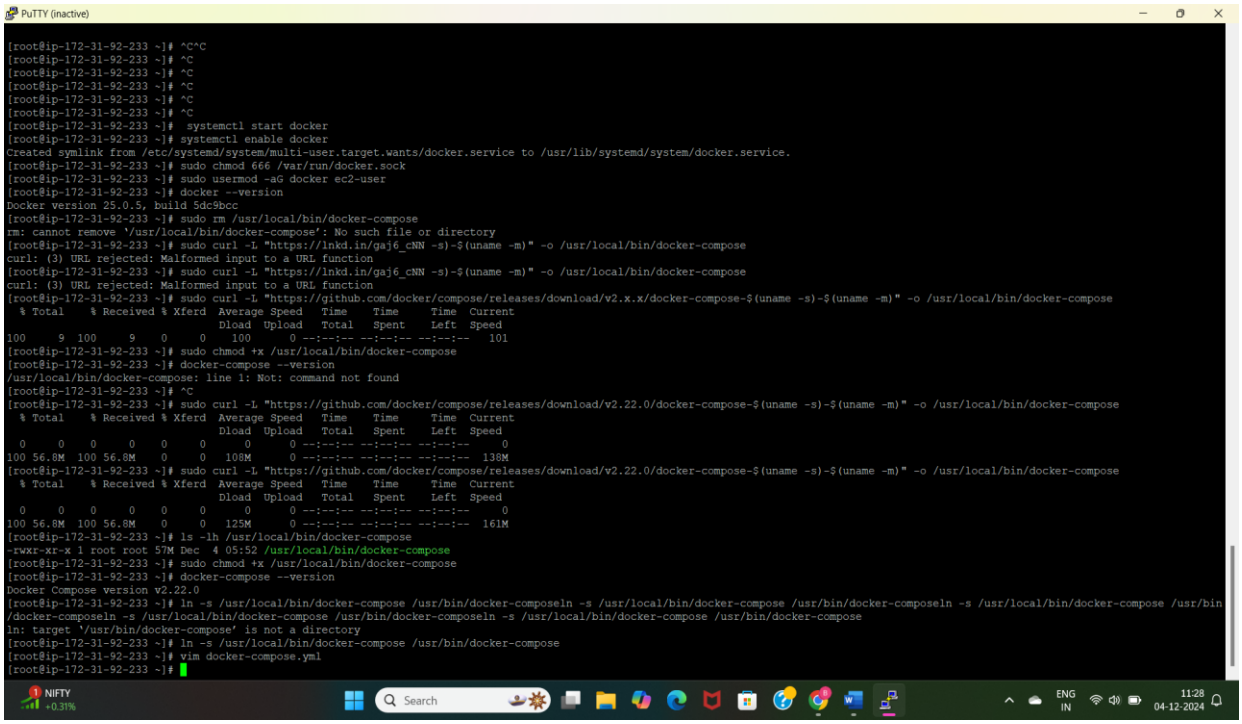


➲ Now give permissions to add limited linux user account to docker group by using a command as "<sudo usermod -aG username(ec2-user)> or <sudo usermod -a -G username(ec2-user)> another command is <sudo chmod 666 /var/run/docker.sock>

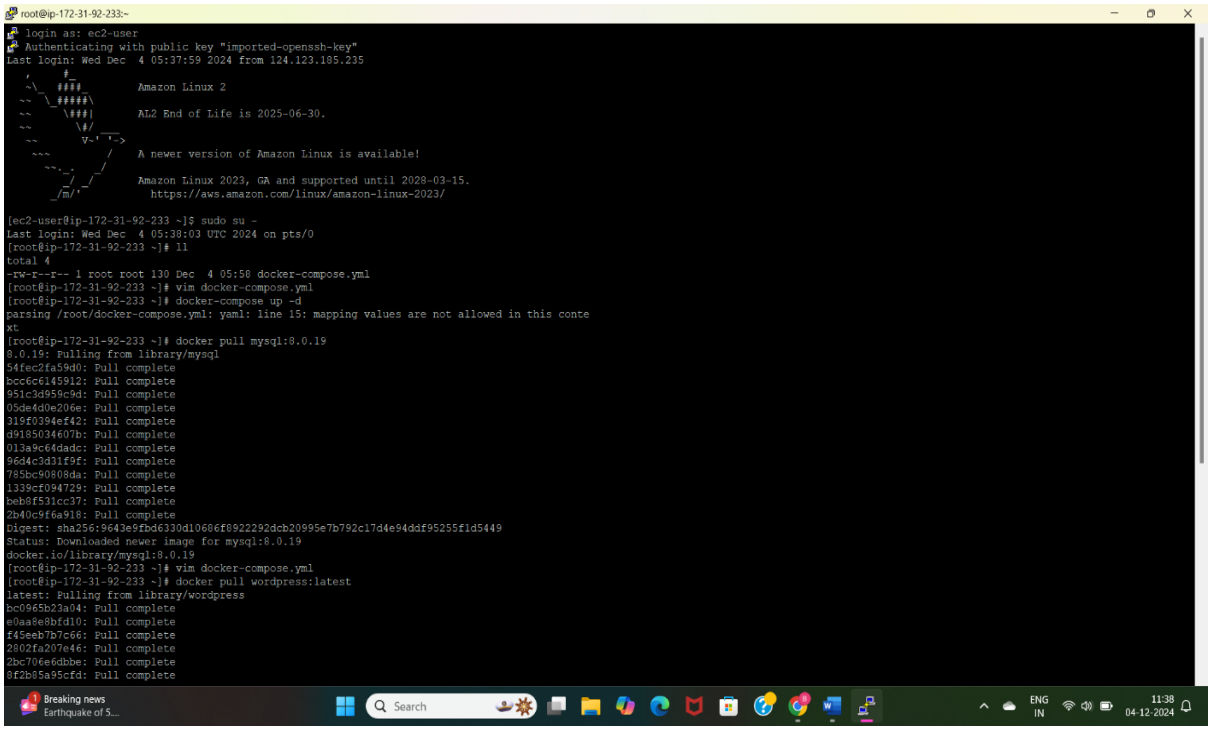


➲ Install docker compose.(go to browser and use docker compose installation) and we get the following commands

I. sudo curl -L "https://github.com/docker/compose/releases/download/v2.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
II. sudo curl -L "https://github.com/docker/compose/releases/download/v2.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
III. ls -lh /usr/local/bin/docker-compose
IV. sudo chmod +x /usr/local/bin/docker-compose
V. docker-compose –version

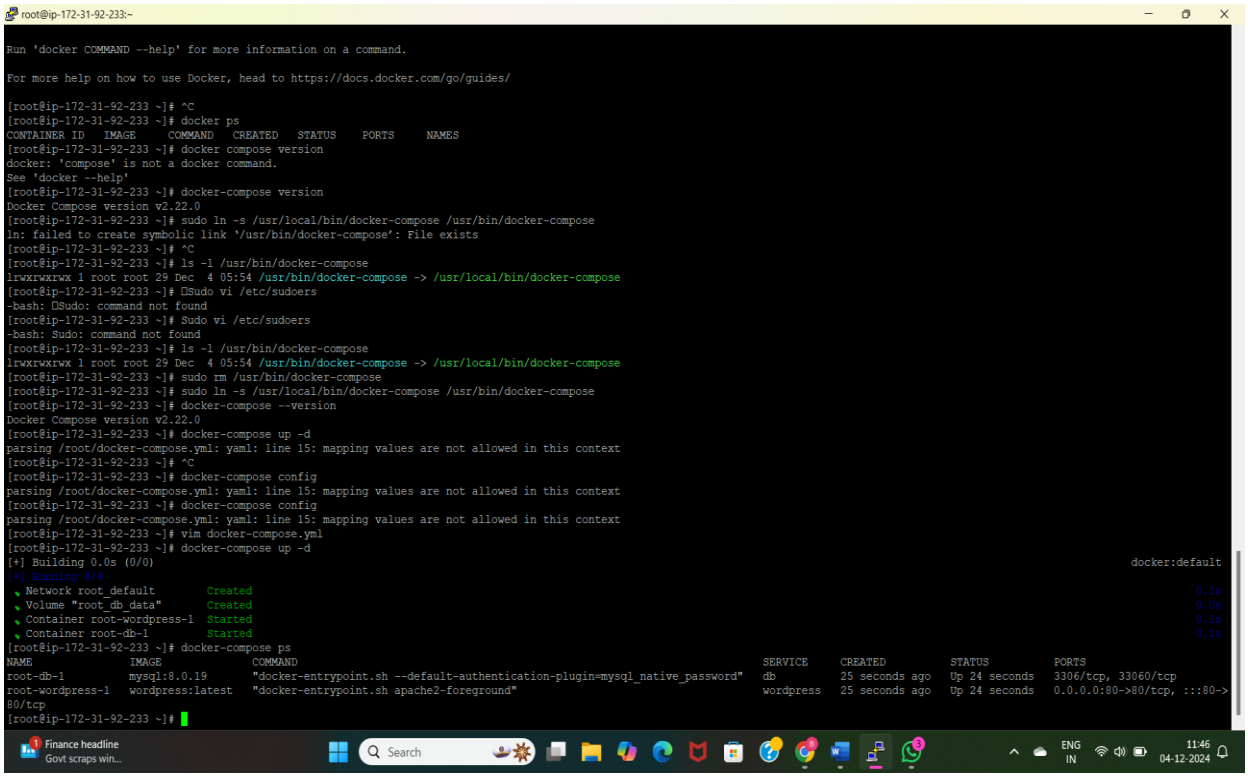


- Apply executable permissions to the binary by using command as <Sudo chmod +x /usr/local/bin/docker-compose>
- Now create the symbolic link by using command as <ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose>
- Create a docker-compose.yml file for wordpress application.

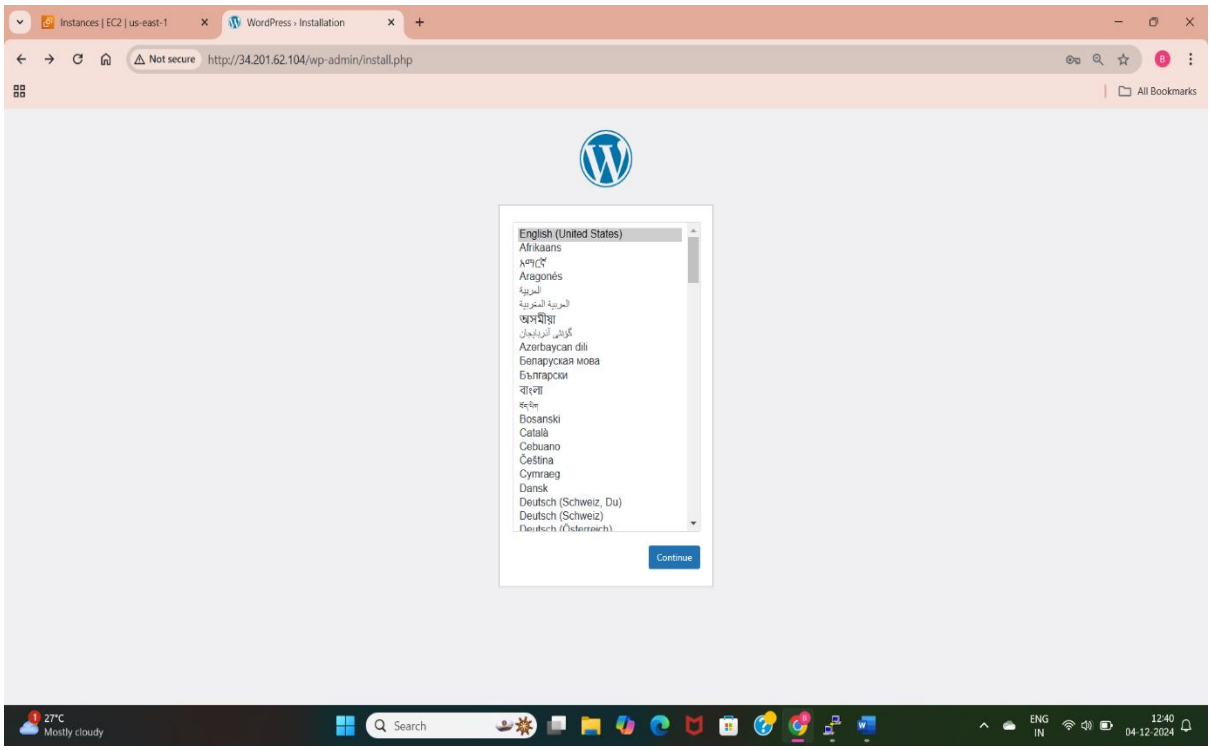


```
version : '3.3'
services:
    db:
        image: mysql: 8.0.19
        command: 'default-authentication-plugin=mysql_native_password'
        volumes:
            -db_data: /var/lib/mysql
        restart: always
        environment:
            - MYSQL_ROOT_PASSWORD=wordpress
            - MYSQL_DATABASE=databasename
            - MYSQL_USER=username
            - MYSQL_PASSWORD=password

    wordpress:

        Image: wordpress: latest

        ports:

            -80:80

        restart: always

        Environment:

            - WORDPRESS_DB_HOST=db
            - WORDPRESS_DB_USER=username
            - WORDPRESS_DB_PASSWORD=password
            - WORDPRESS_DB_NAME=db name

volumes:

    db_data:
```

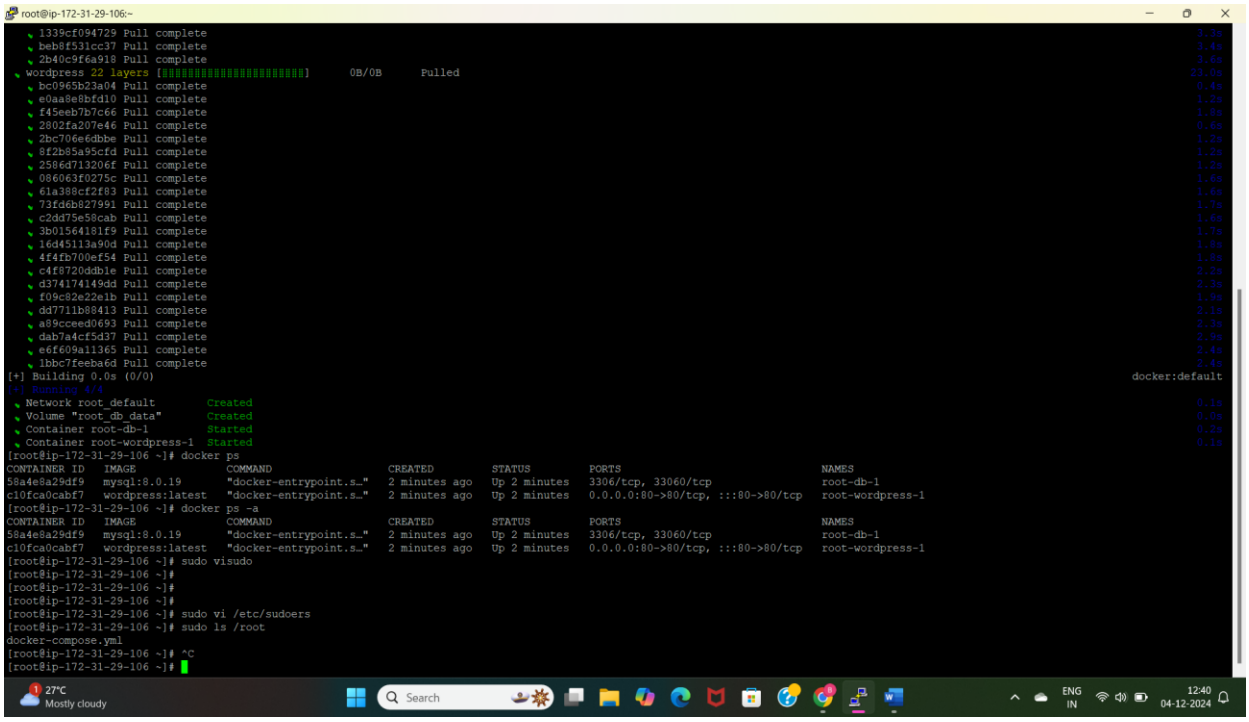
- create a file with the name of docker-compose.yml.
- To run the docker file we need to run the "docker-compose up -d" command.



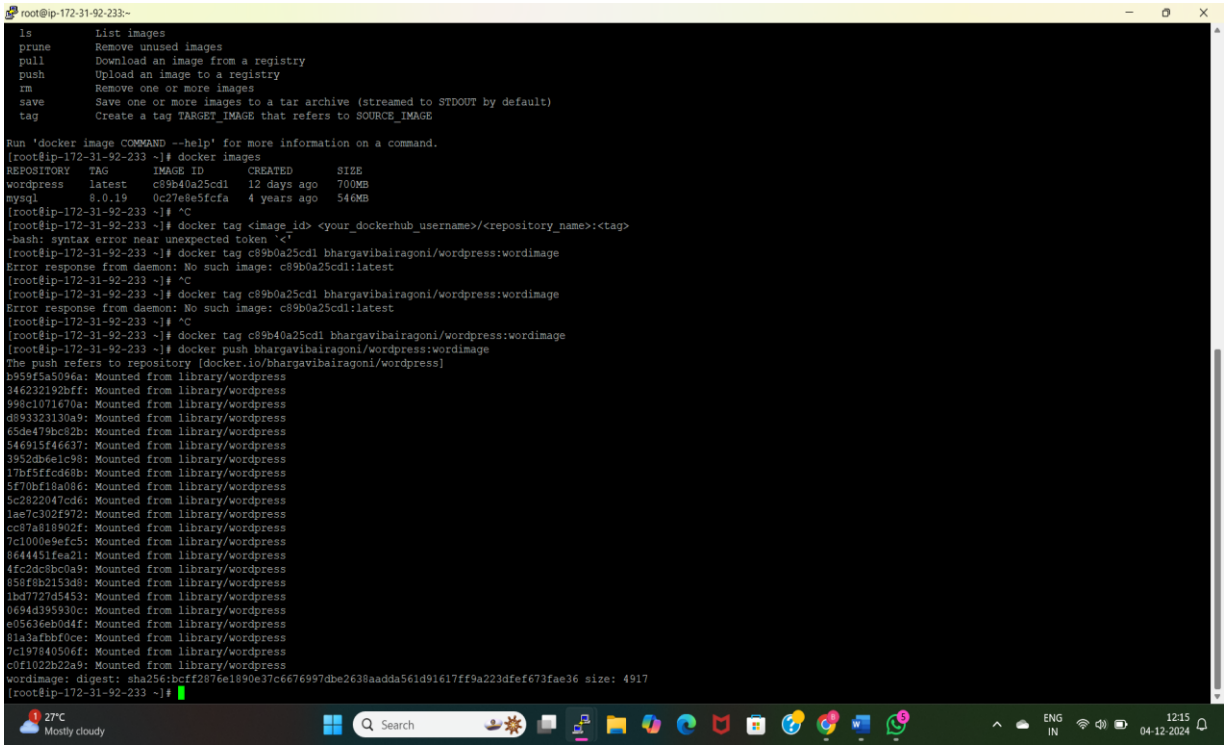
➲ Now pull the images of MYSQL and WORDPRESS we have to execute this created docker-conpose.yml file by using command as <docker-compose up -d>



➲ Now see the containers of the pulled images of MYSQL and Wordpress



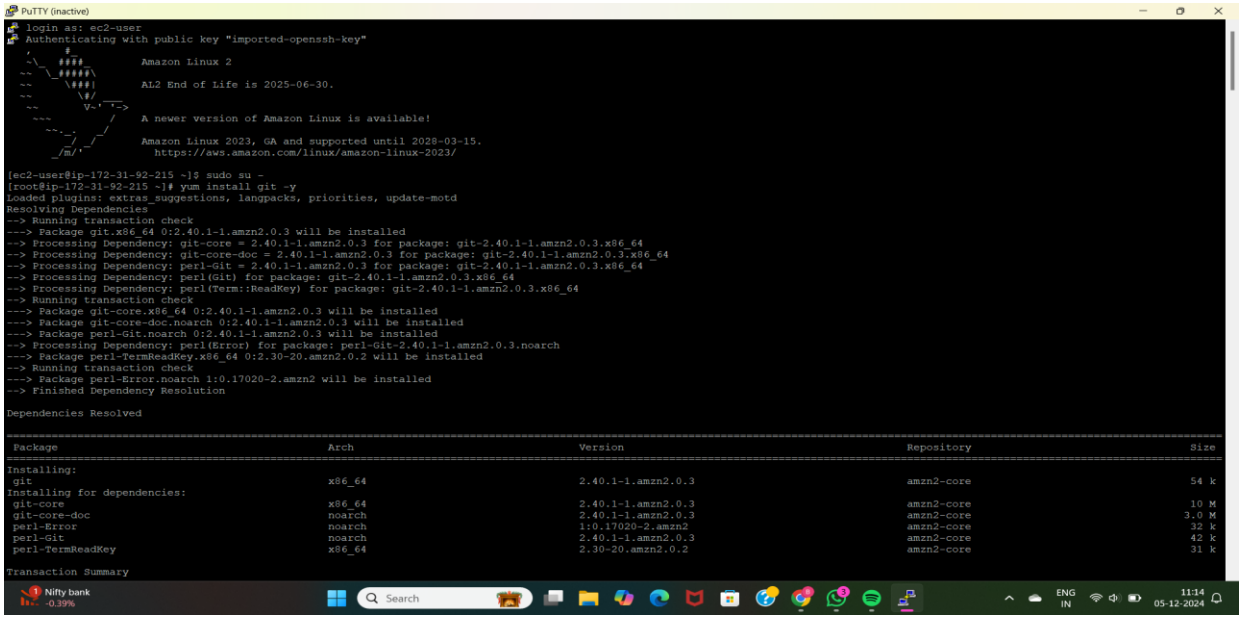
- Sudo visudo Jenkins (ALL)-ALL NOPASSWD: ALL –linux amazon
- Sudo vi /etc/sudoers



- Push the docker images to the docker hub.

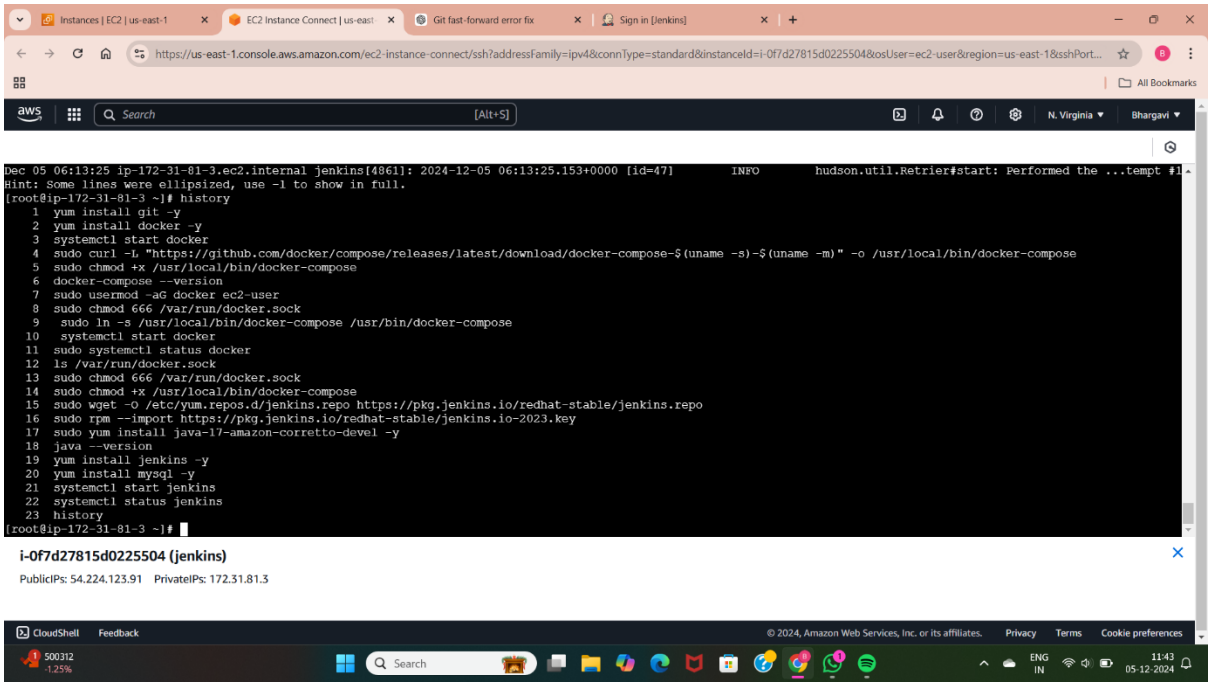
METHOD-3: DEPLOYING WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS

- To deploy wordpress application using git and jenkins first we need to install all the dependencies required for this applications.



Here I have installed the git using “yum install git -y”. Because I am using the above docker-compose file and that is stored in the github.

- Install docker using “yum install docker -y”, start the docker using “systemctl start docker”.
- Install docker-compose because the file is written in the file docker-compose.yml.

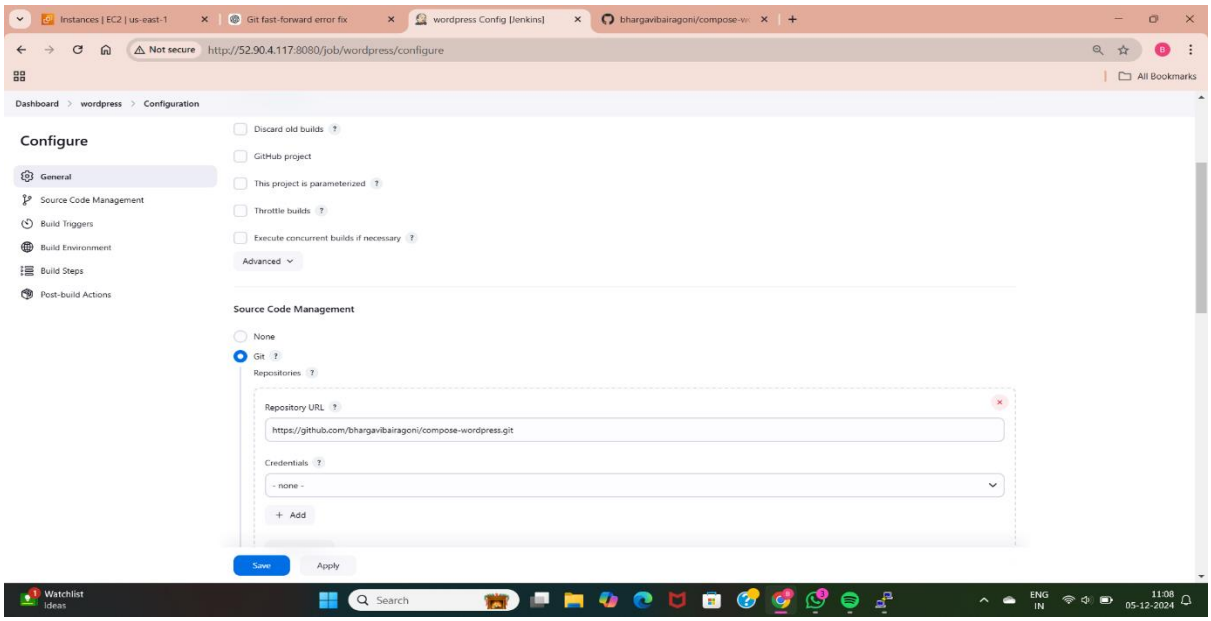


These are the commands used to install docker-compose, giving permissions to the docker-compose to execute the file. Create a user called ec2-user and assign it to the group called docker.

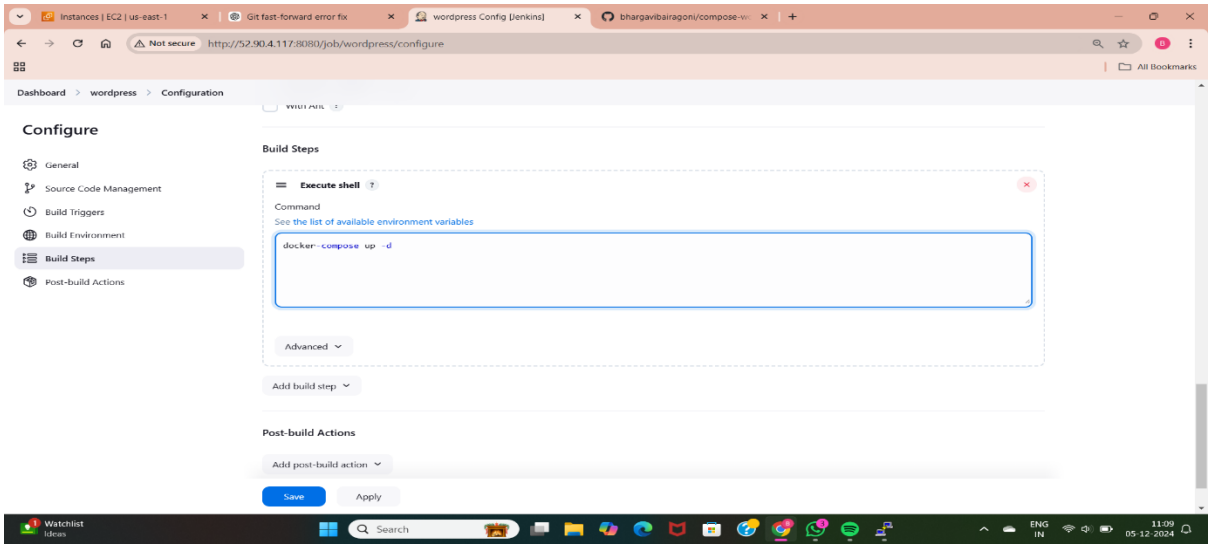
- Creating a symbolic link (ln -s) which is used as shortcut name or alias name.

- Downloading the official Jenkins repository file from the internet and saves it to the system's repository directory. This allows the system to use yum to install and update Jenkins easily.
- The command imports the Jenkins GPG key to verify the authenticity of packages from the Jenkins repository. It ensures that the packages are safe and haven't been tampered with.
- Install java 17 version as dependency for Jenkins using the command "yum install java-17-amazon-corretto-devel -y"
- Install jenkins using "yum install Jenkins -y" and start the Jenkins using "systemctl start Jenkins".
- To access the Jenkins we need to copy the instance public ip and paste it in google by specifying Jenkins port number 8080 (ip:8080)

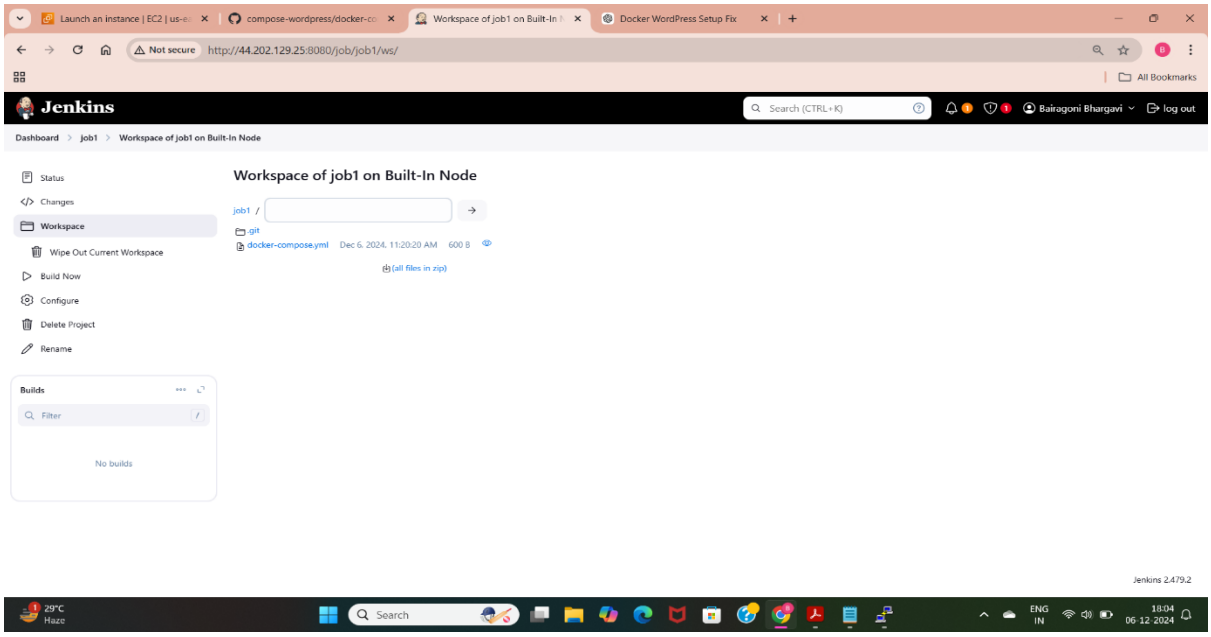
Here I have created a free-style job with the name of job1 and using that job I have deployed the wordpress application.

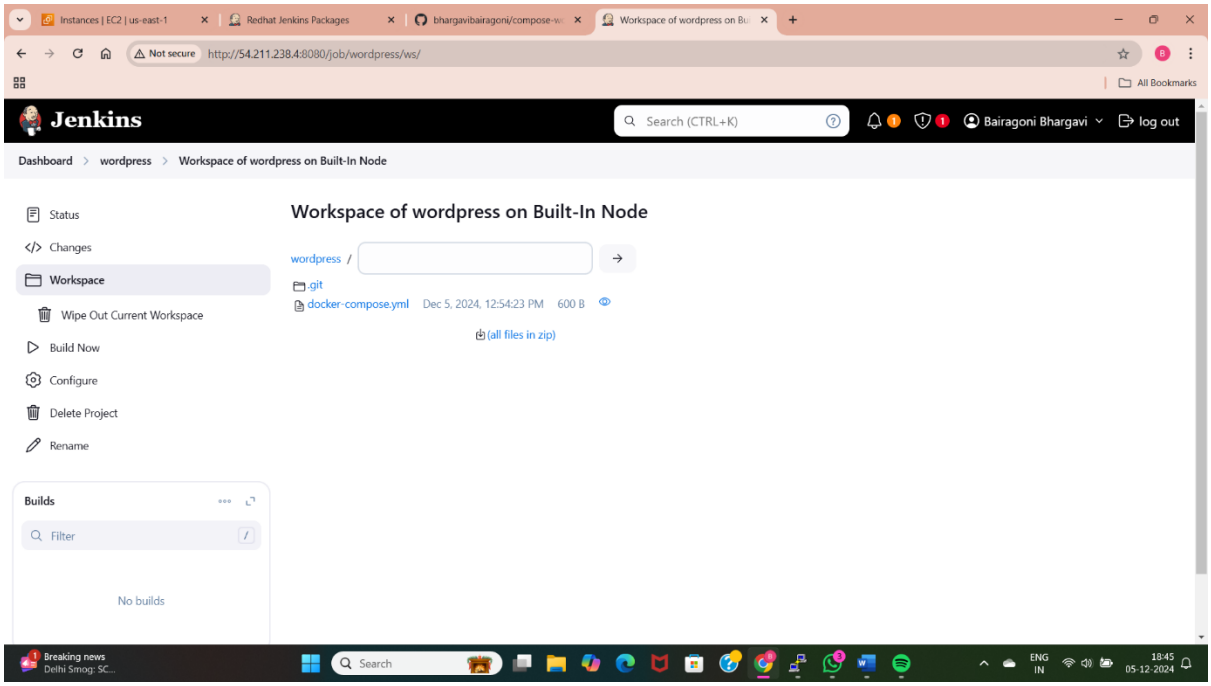


- Here am taking the compose file from github and kept in Jenkins. I have stored the compose file in public repository, so here no need to provide any credentials.

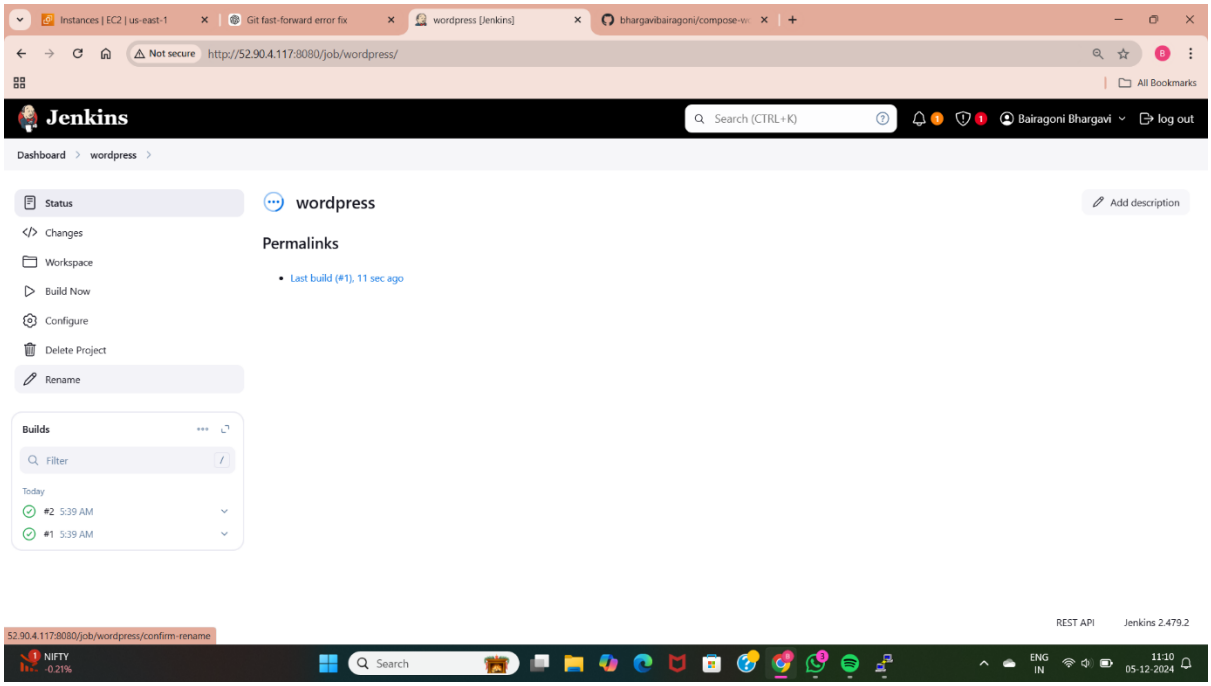


- Here am using build step called “Execute shell”. Whatever we mention in the execute shell is going to excute. Here am running the docker-compose.yml file to access the wordpress application.

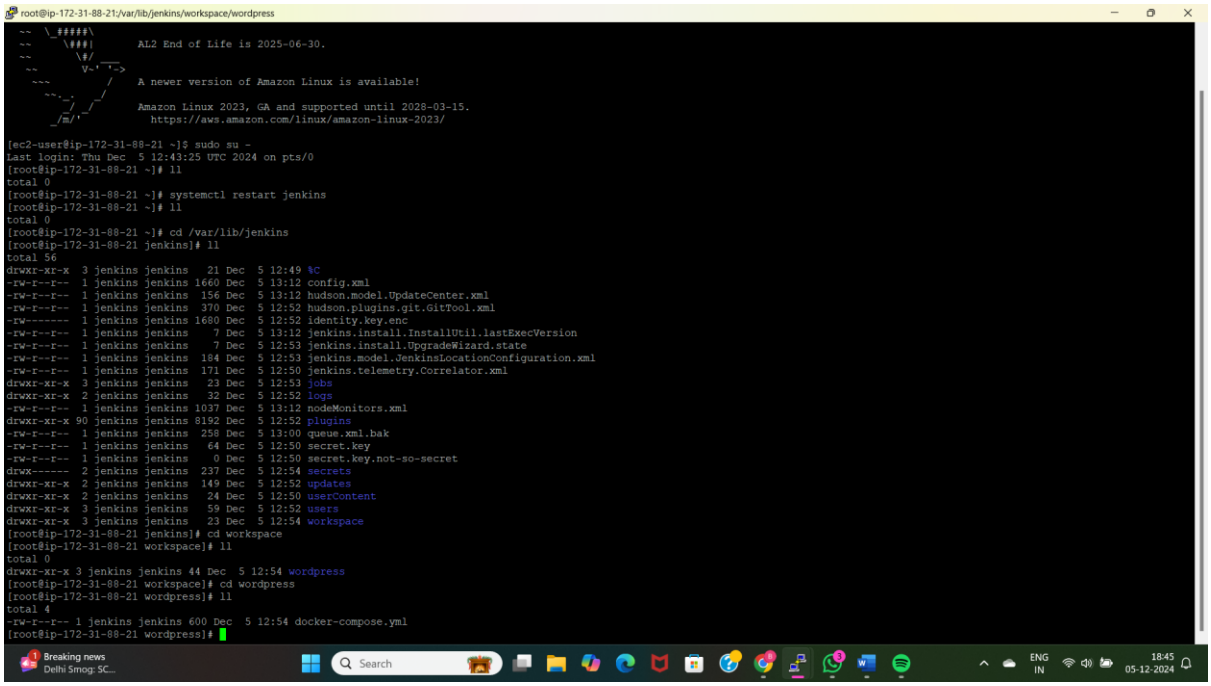




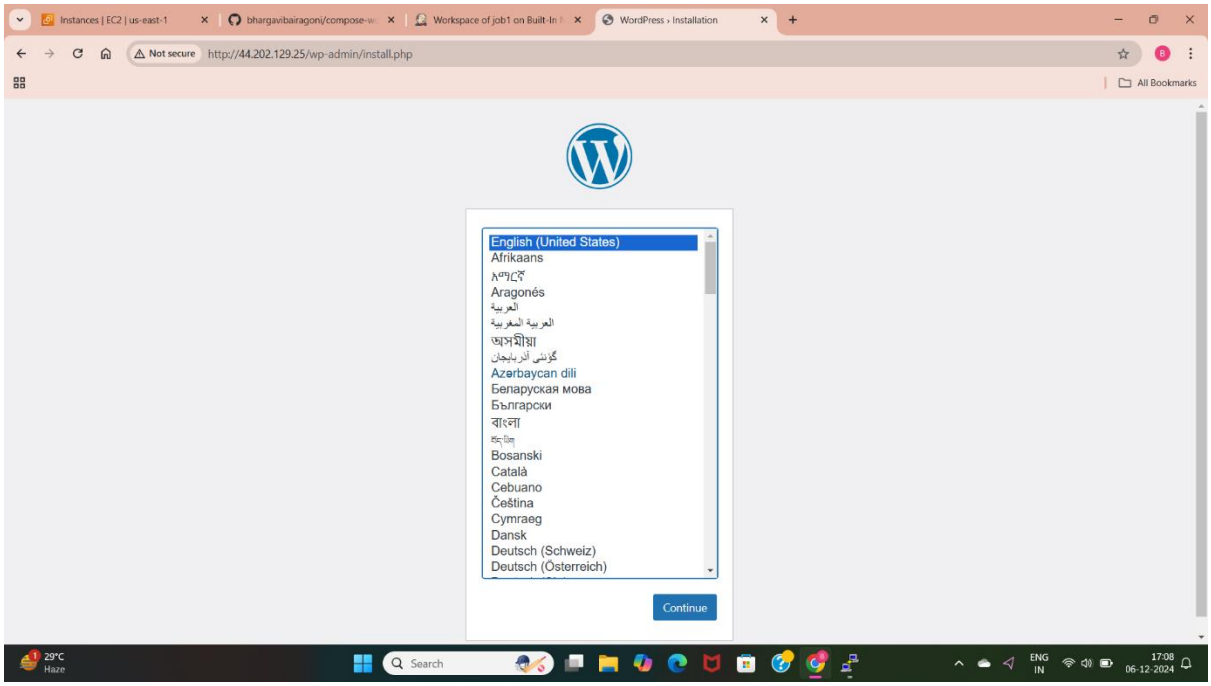
- After specifying the configurations in the Jenkins we need to build the code to access the application and we can also identify errors and correct the errors using configure option.



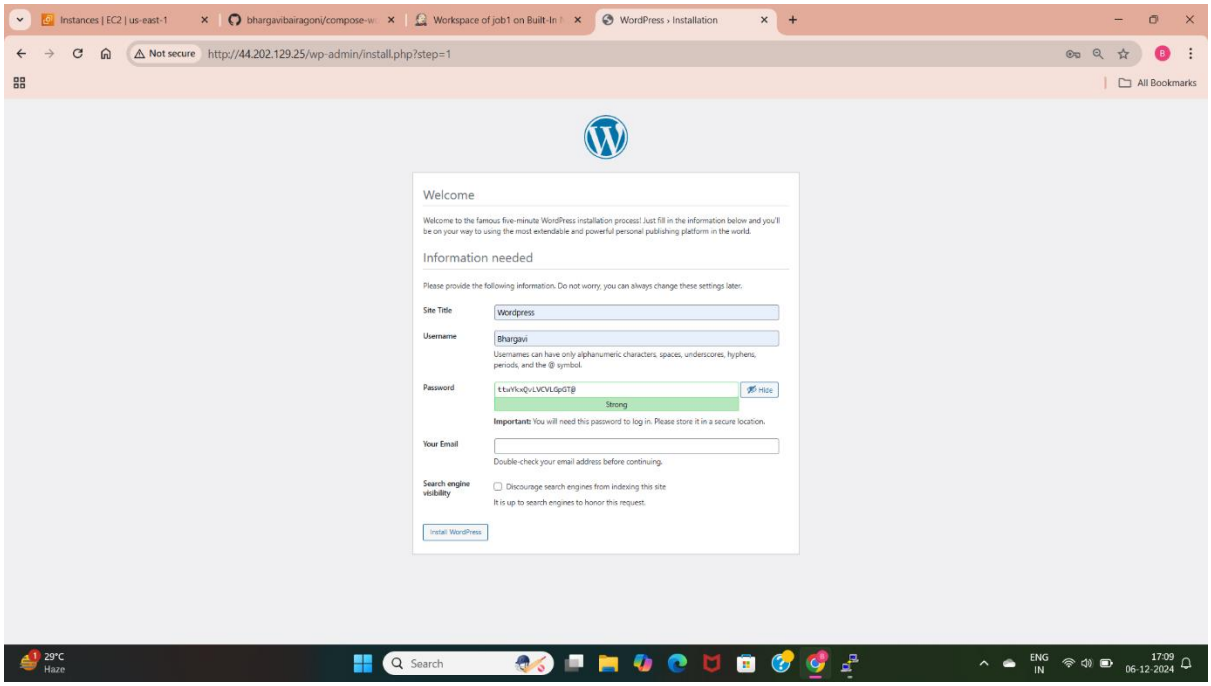
➲ Here the build is successful. We can access the application now by using the ip address of the instance.



➲ We can also see the docker-compose file through the terminal by using the Jenkins path “/var/www/Jenkins”. Inside this path we can see the “workspace” option there we can see the created jobs. By going inside the job we can see the file.



➲ Here I have deployed the wordpress application using the git and Jenkins. I am able to access the wordpress application. This is the front page of wordpress.

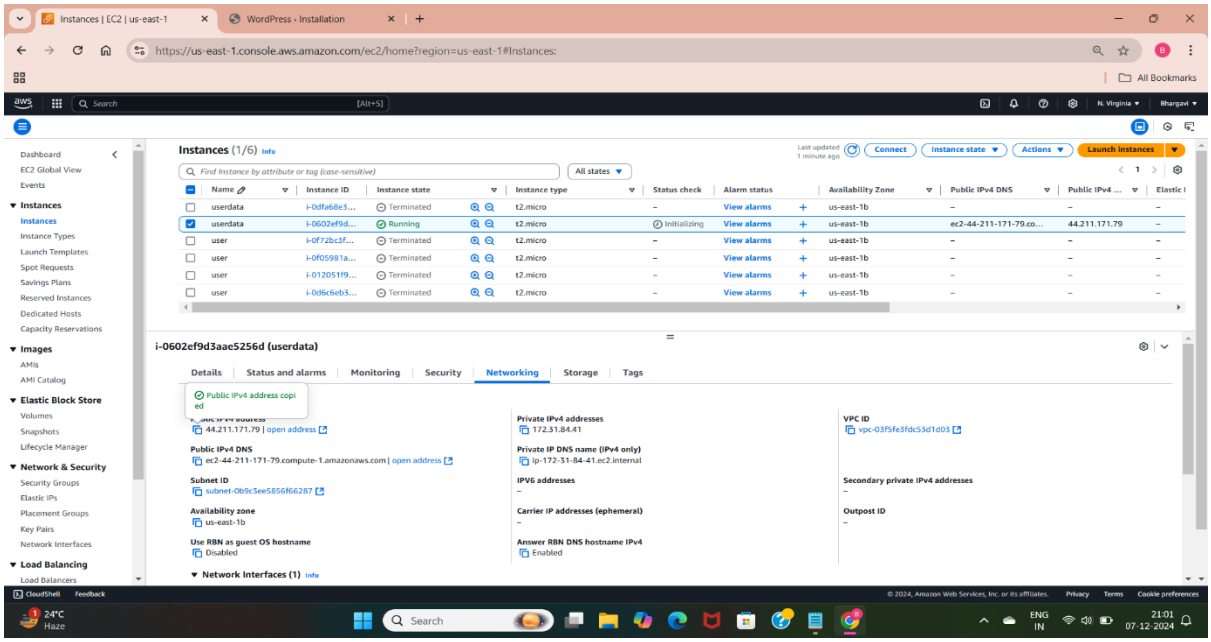


➲ We can access this application by creating the username and password and need to specify our details.

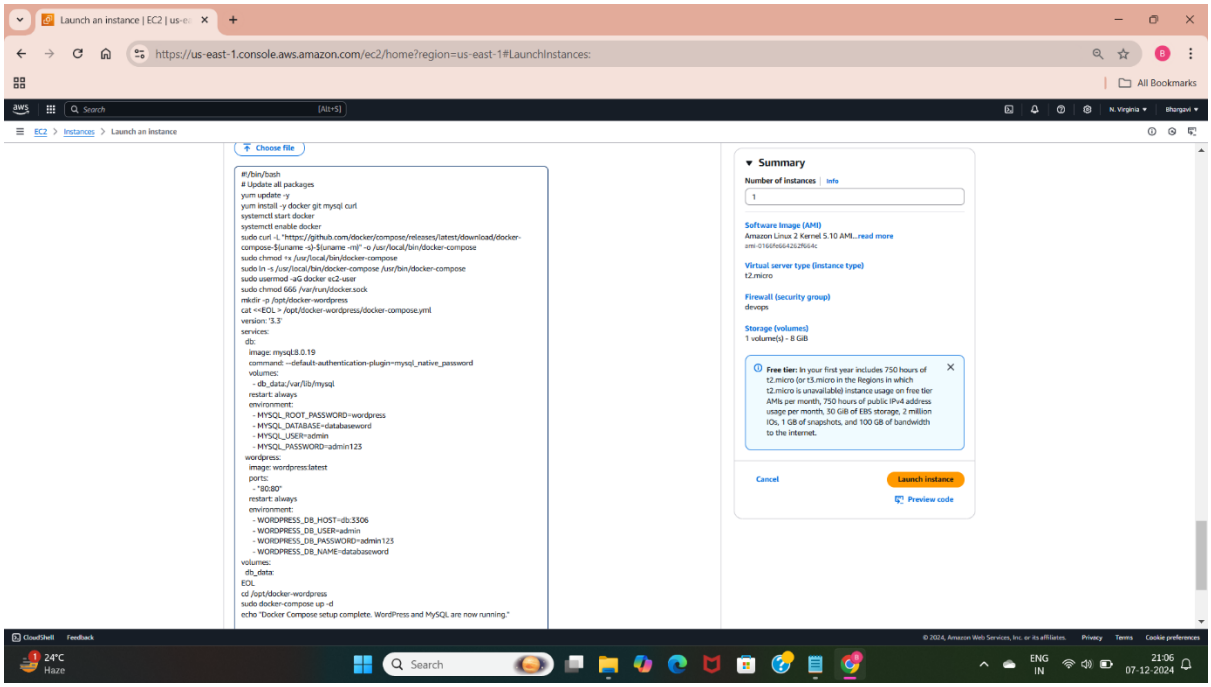
METHOD-4: DEPLOY WORDPRESS WEB APPLICATION BY USING USERDATA OF EC2 INSTANCE

Userdata is a script which is used to configure specific instructions like installing git, httpd, Jenkins and other packages which are executed while launching EC2 instance. It is also used to automate the process of deploying any applications or starting any servers.

➔ To use userdata first we need to create a EC2 instance.

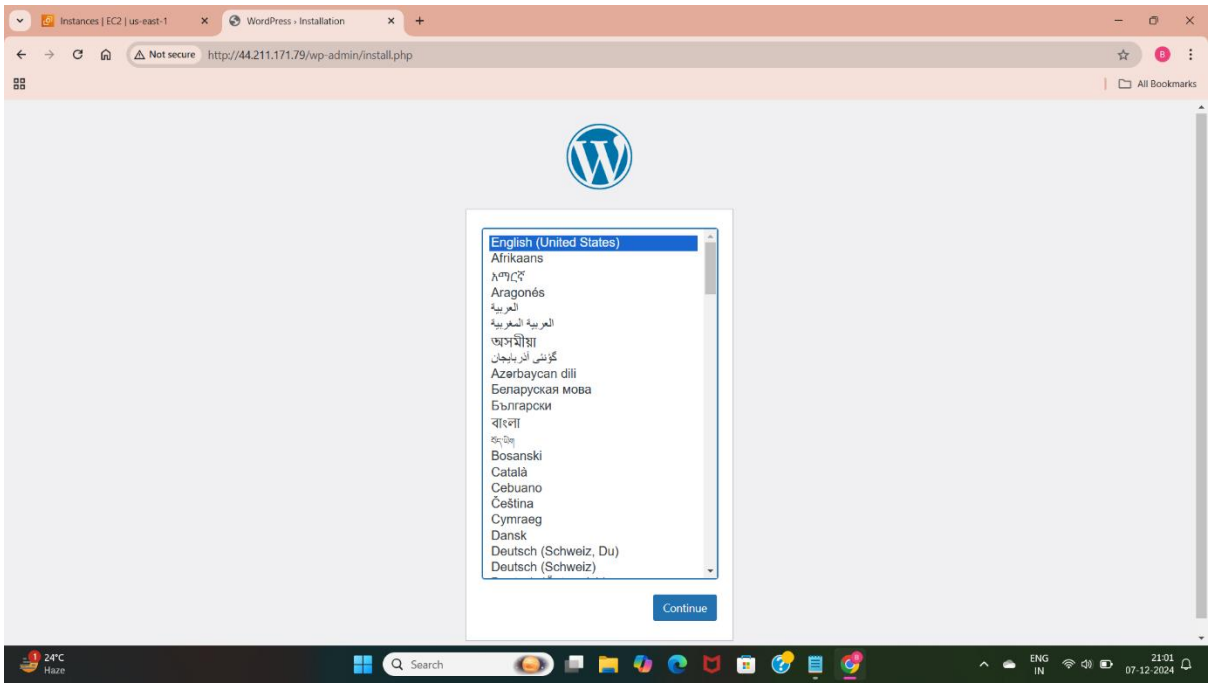


➔ I have created a EC2 instance with the name “userdata”, application os image as amazon linux with kernel 5.10, allowed the inbound ports with 22 for SSH, 80 for HTTP and 3306 for MYSQL.

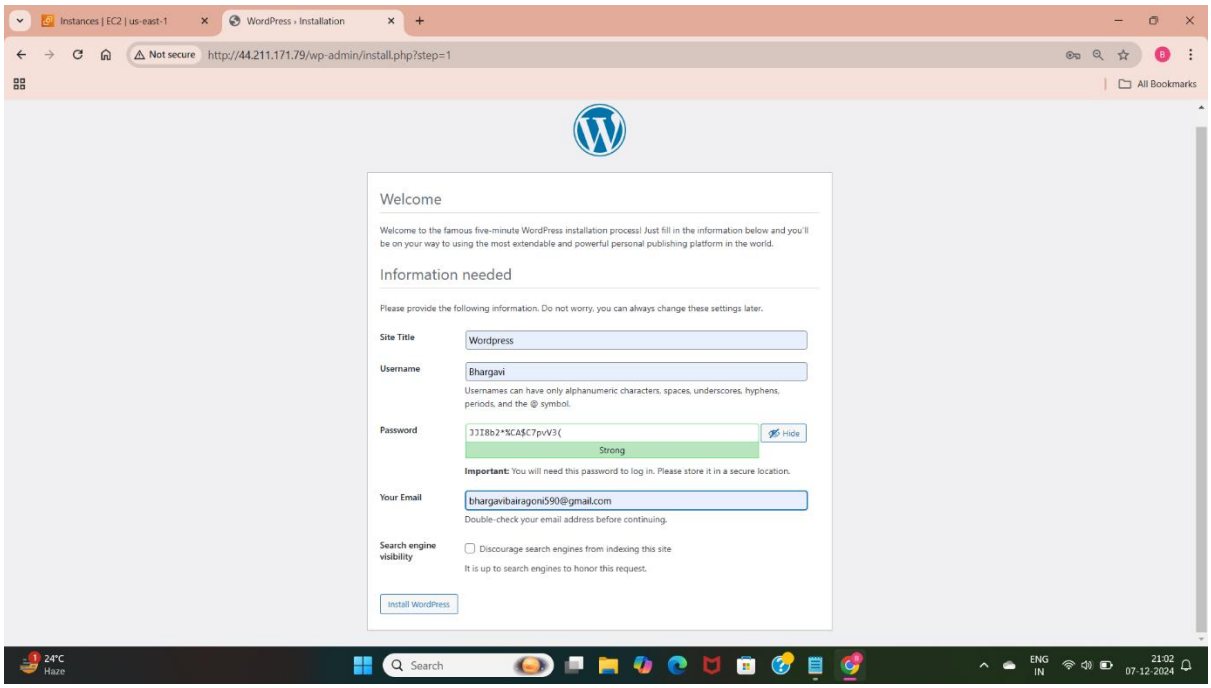


➔ This is the userdata I have used while creating EC2 instance. This script is used to deploy wordpress application using docker-compose. Here I have installed docker, mysql, docker-compose and given required permissions for the docker-compose.

➔ Created a directory for docker-compose and moved inside that directory and executed the docker-compose file.



➔ With the above specified configurations I have successfully deployed wordpress application using userdata script.

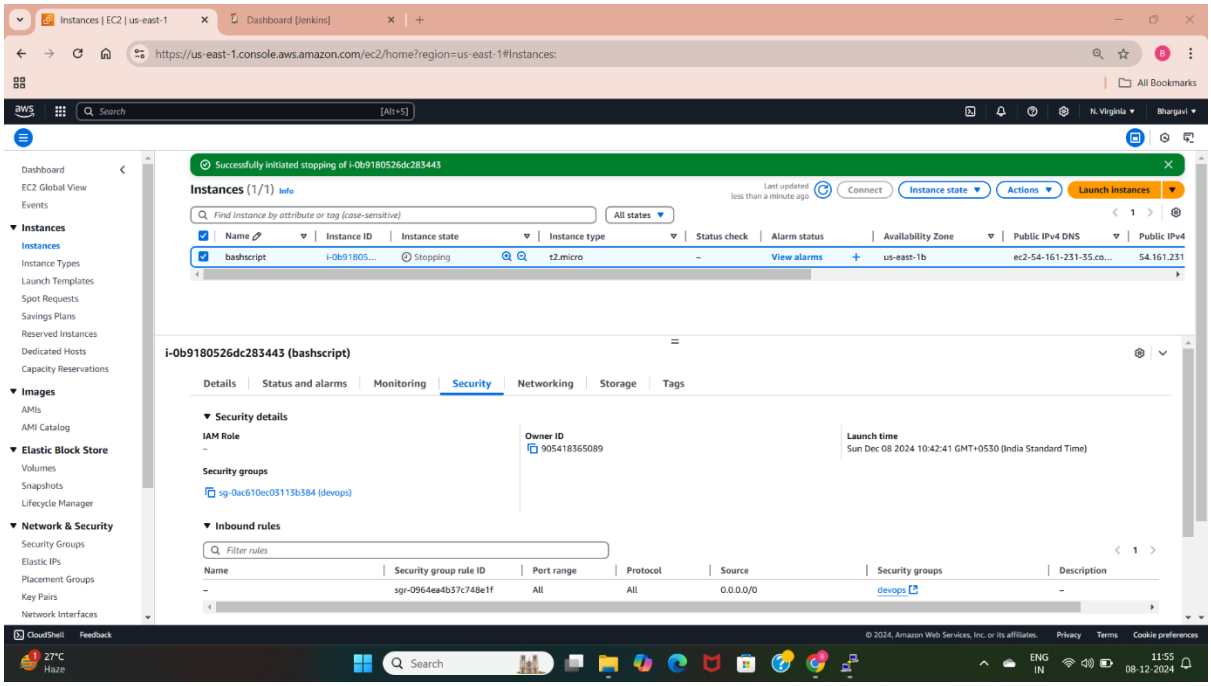


➔ This is the user login page of the wordpress application, which uses the above mentioned username and password.

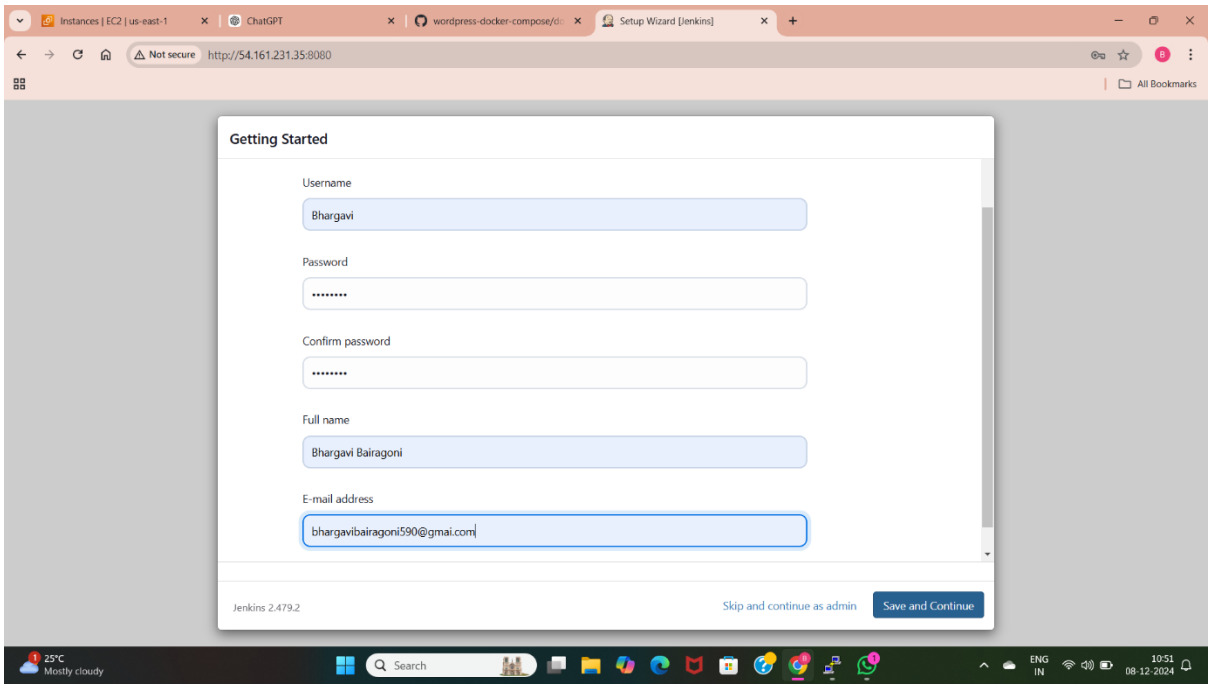
METHOD-5: DEPLOY WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS EXECUTE SHELL (BASH SCRIPT)

Jenkins is a open-source automation server used for continuous integration and continuous delivery/deployment.

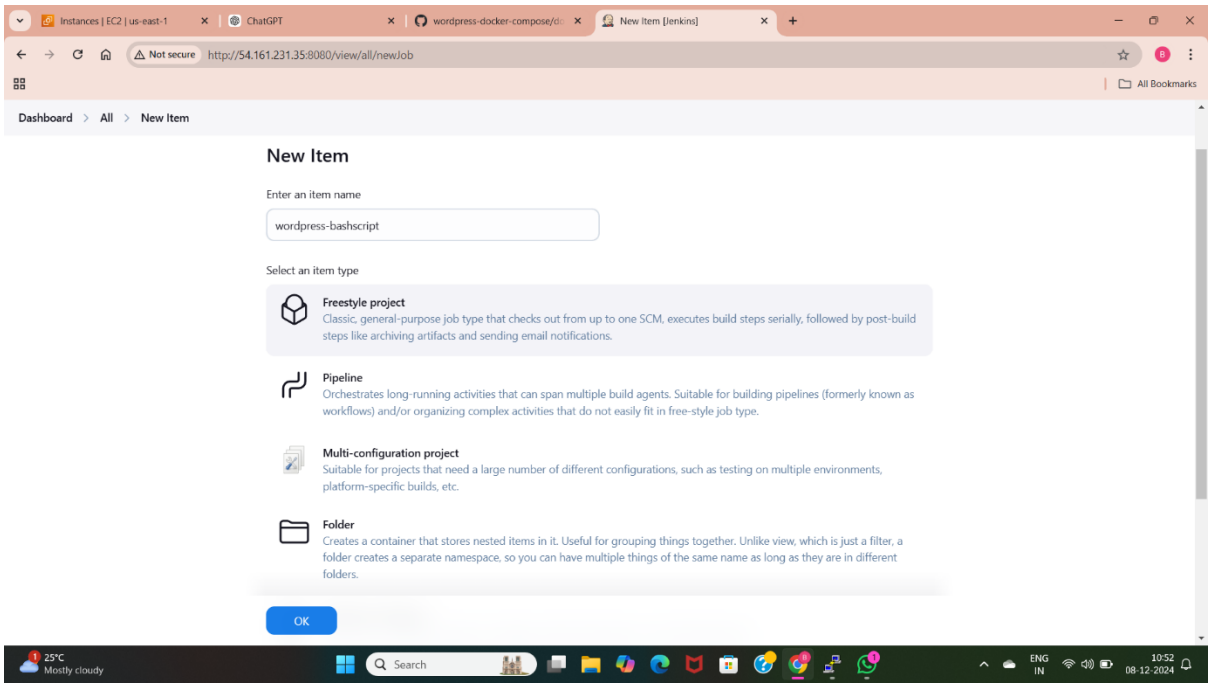
- ♣ To access Jenkins first we need to create a EC2 instance with specific configurations, connect the instance with the terminal and install repositories, tokens related to Jenkins, install Jenkins dependency i.e java17 version, install and start Jenkins.



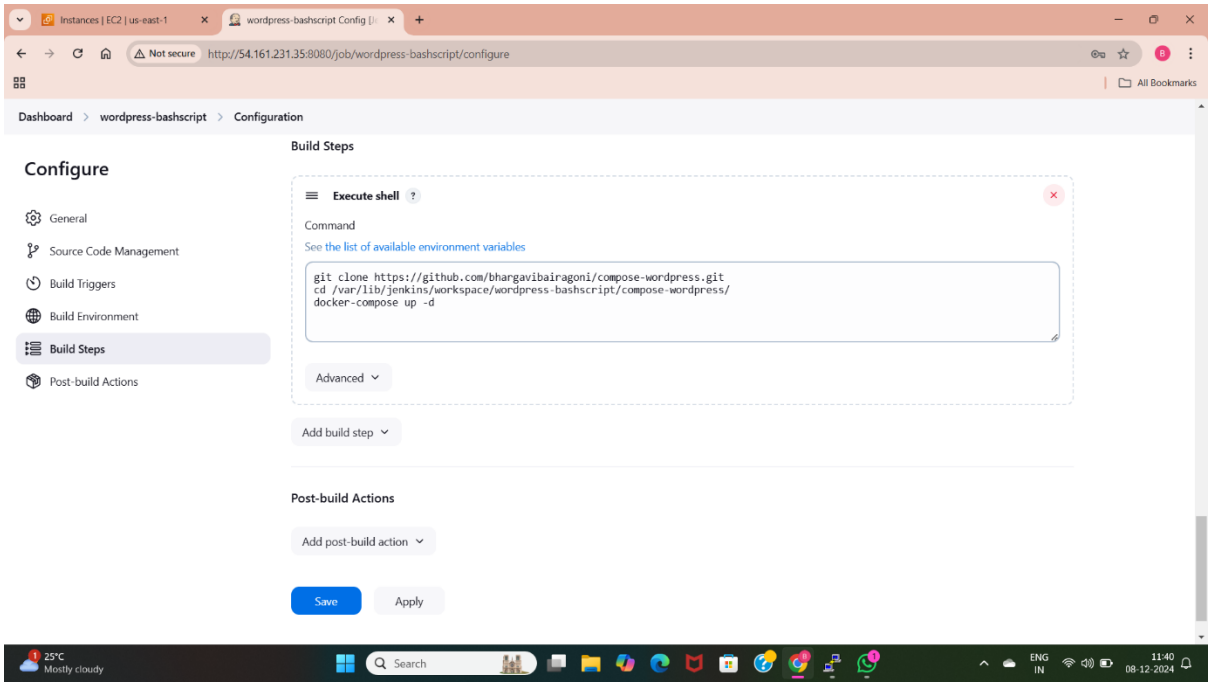
- ♣ After starting Jenkins we need to paste the ip address with Jenkins port number in the browser.
- ♣ Now we can access the Jenkins, install the necessary plugins.



- ♣ Create username and password to access the Jenkins. Here am creating Jenkins account to deploy the application.
- ♣ After entering the required details save and continue, so we can create the jobs in the Jenkins gui.
- ♣ There are two types of jobs in Jenkins
    1. Freestyle
    2. Pipeline

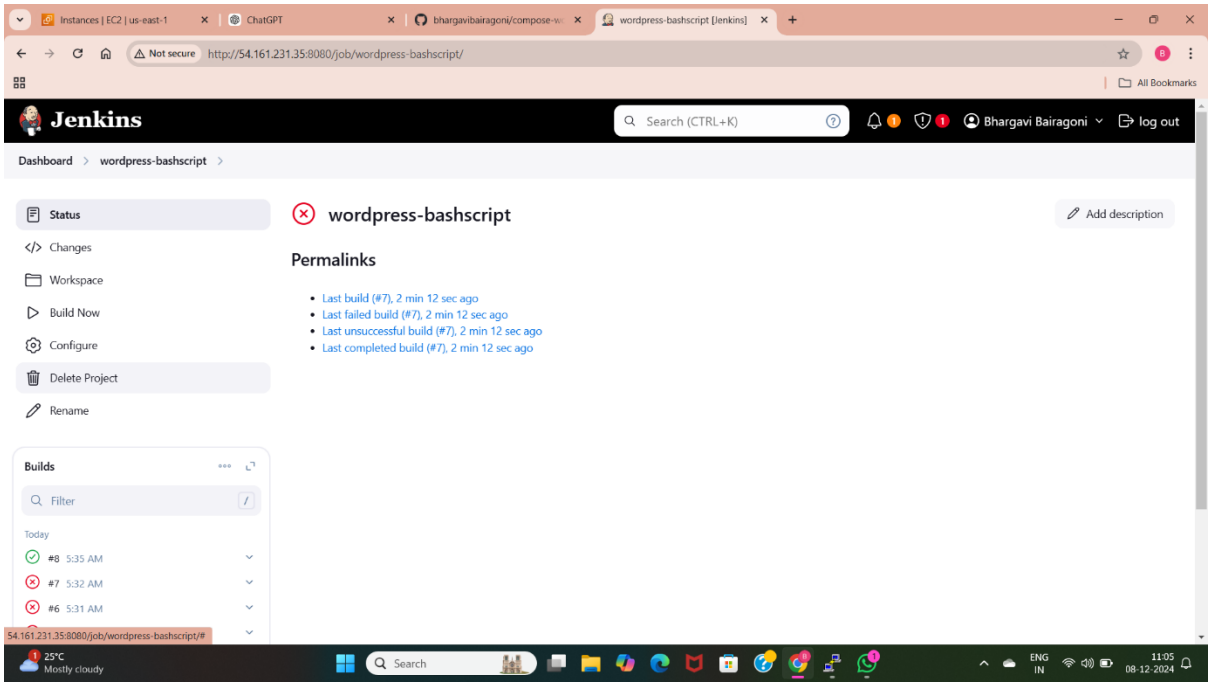


- ♣ Here am creating a Freestyle project with the name “wordpress-bashscript”. So we can deploy application in this job.

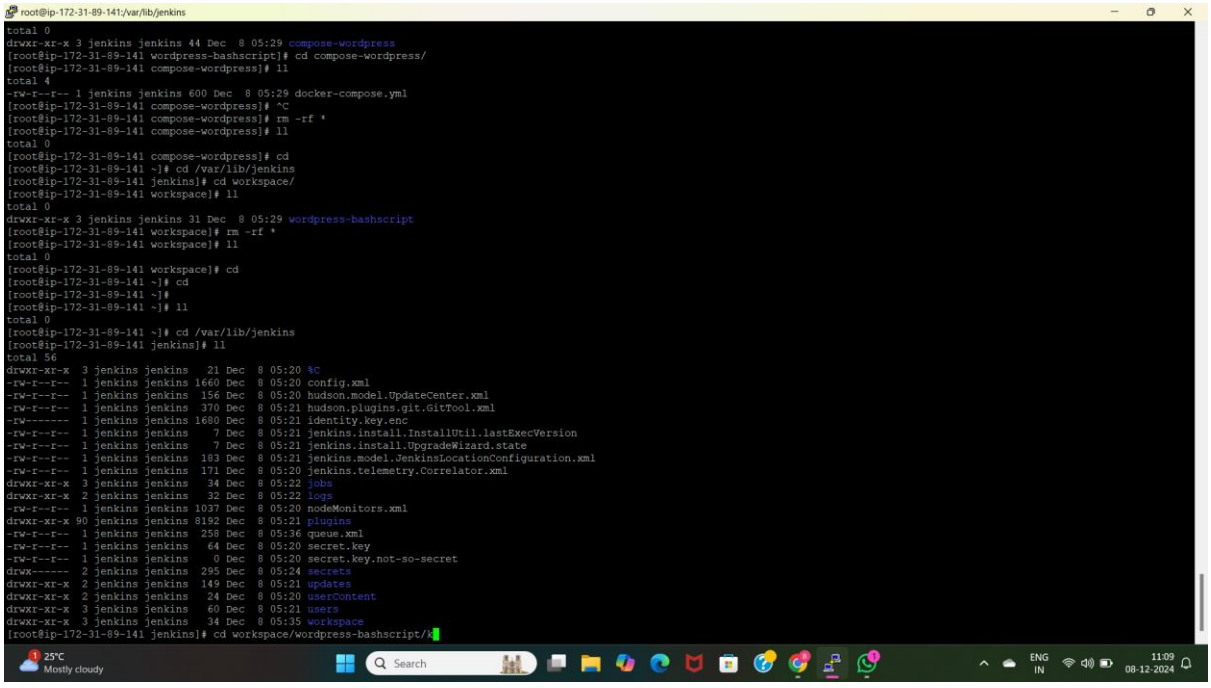


- ♣ Here I have downloaded the docker-compose.yml file from the github with the help of the specific command (git clone) and moved to that path where the compose file is present.

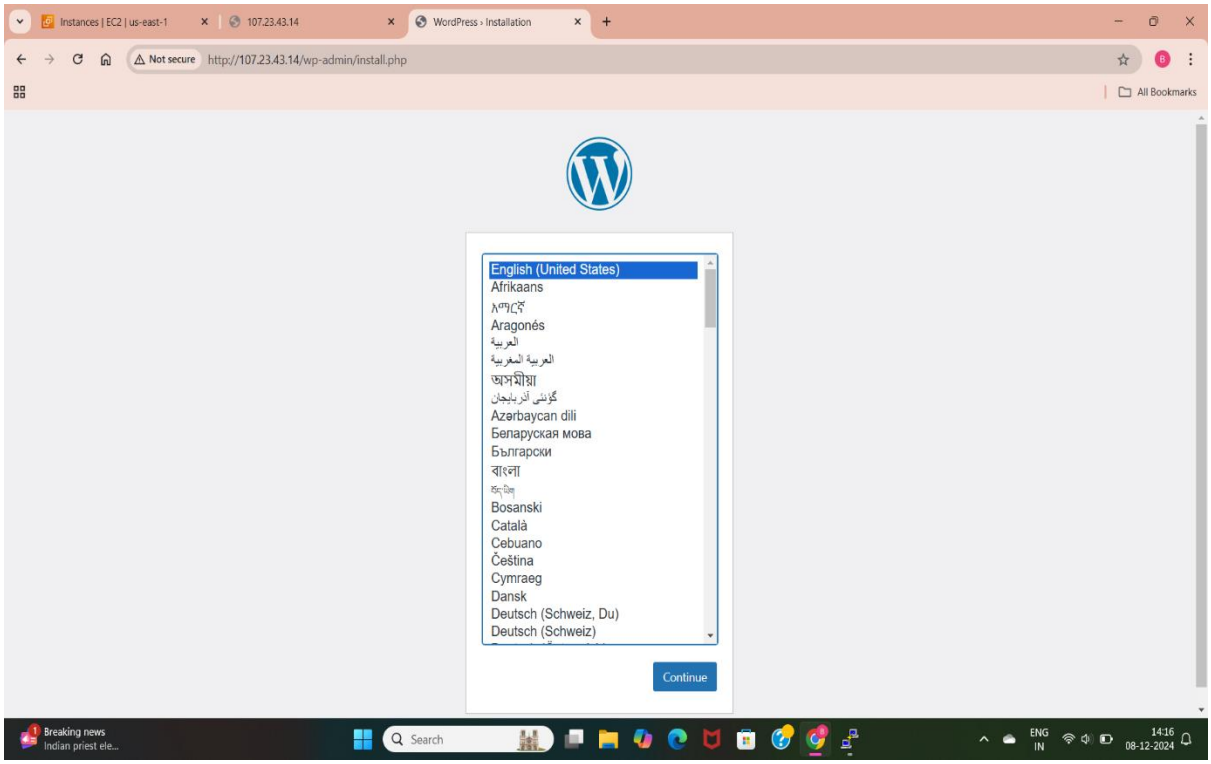
- I have executed the compose file to deploy the application using Jenkins with execute shell(bash script).



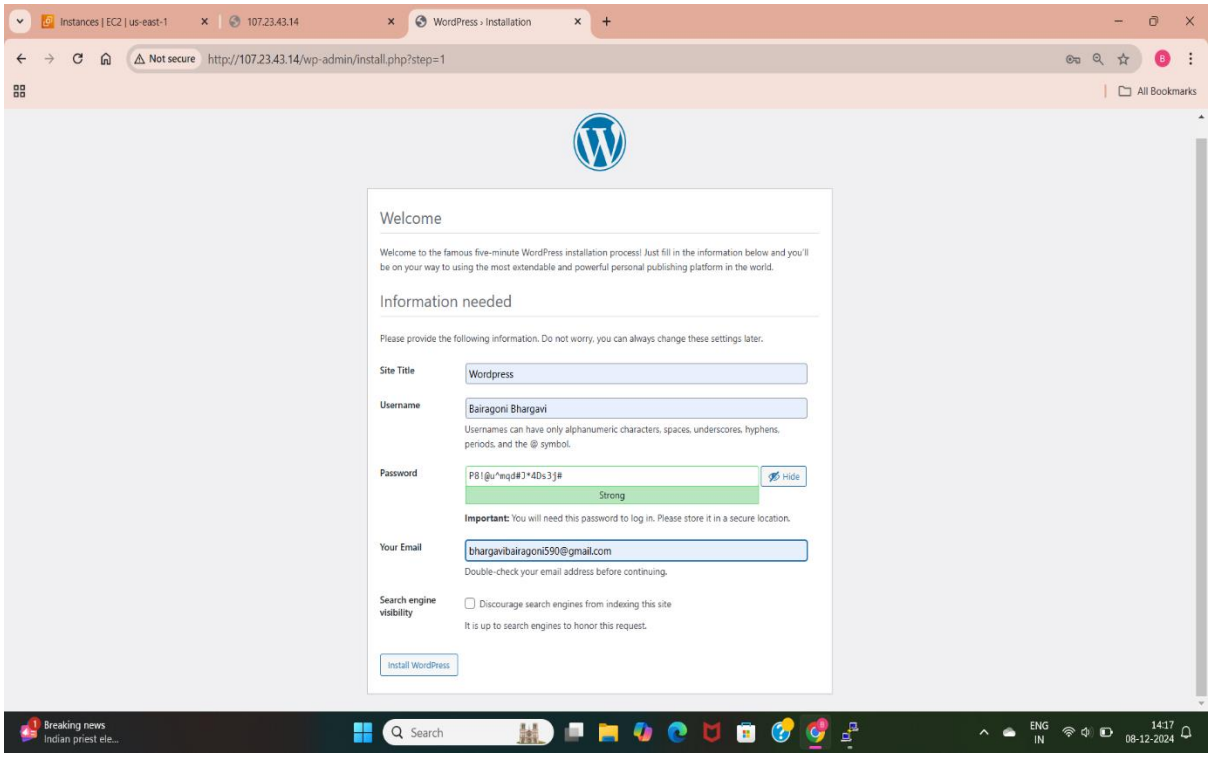
- After specifying specific file through git, am able to build the project. The build is successful after trying specific number of times.



- We can able to see the docker-compose.yml file by using the terminal by going into Jenkins path "/var/lib/Jenkins". In this path we are having a workspace which contains details of our created jobs.
- I have moved to the created job "wordpress-bashscript" and checked with the compose file. It is downloaded with the help of above command "git clone compose file url".



- After successful build step am able to access the deployed application using the public ip of the instance.

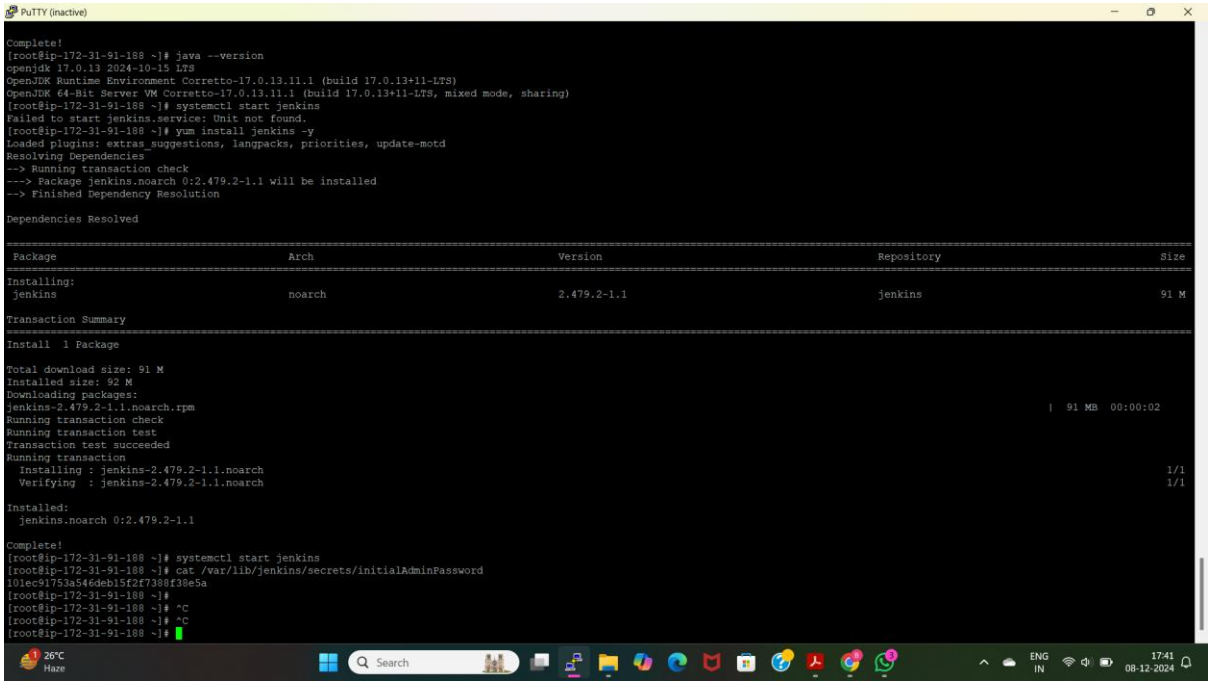


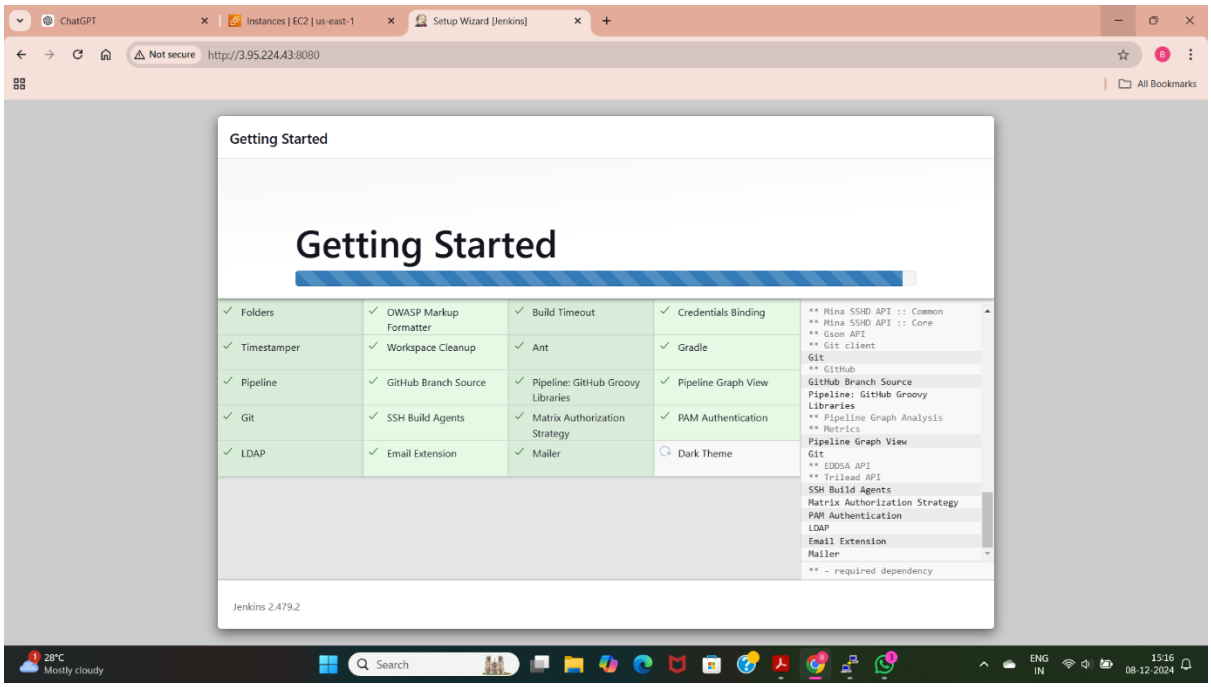
- Here am able to see the wordpress application dashboard, which is asking me to enter the details of the user like username, password and email.

METHOD-6: DEPLOY WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS EXECUTE SHELL

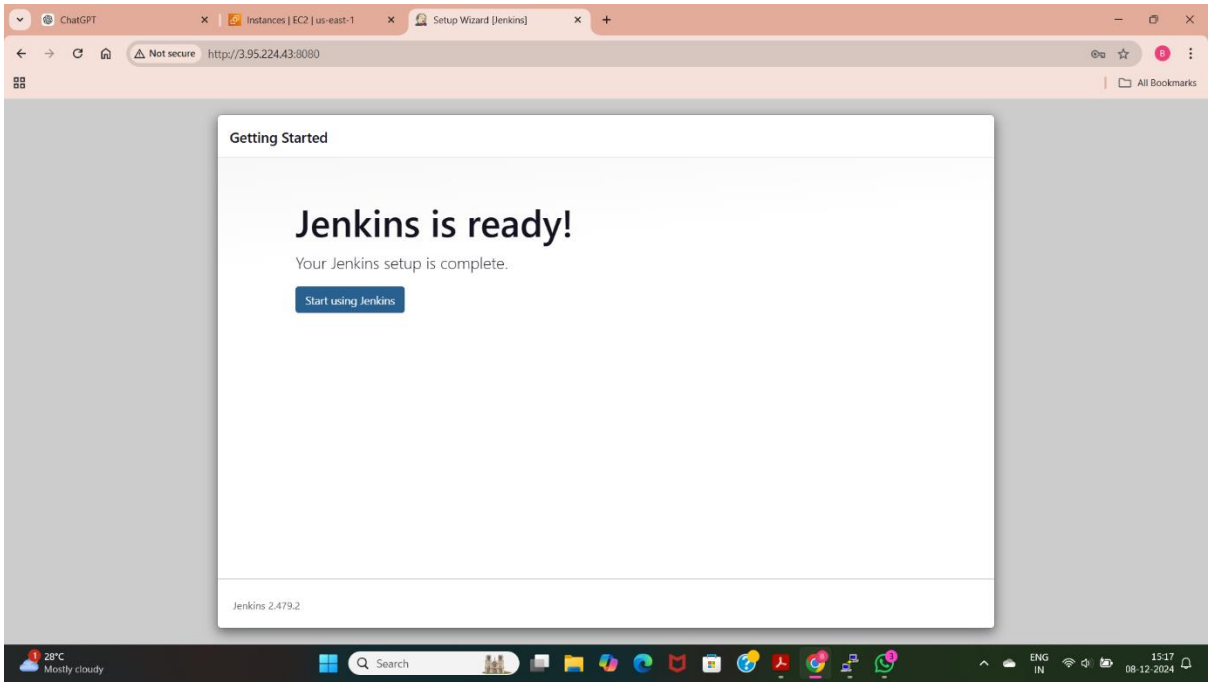
Pipeline is used to automate the process like build, test, and deploy software applications. It ensures that code changes go through various stages to verify their quality and readiness for production.

- ✓ To use pipeline in Jenkins we need to launch a EC2 instance by allowing port number 8080, install required packages.

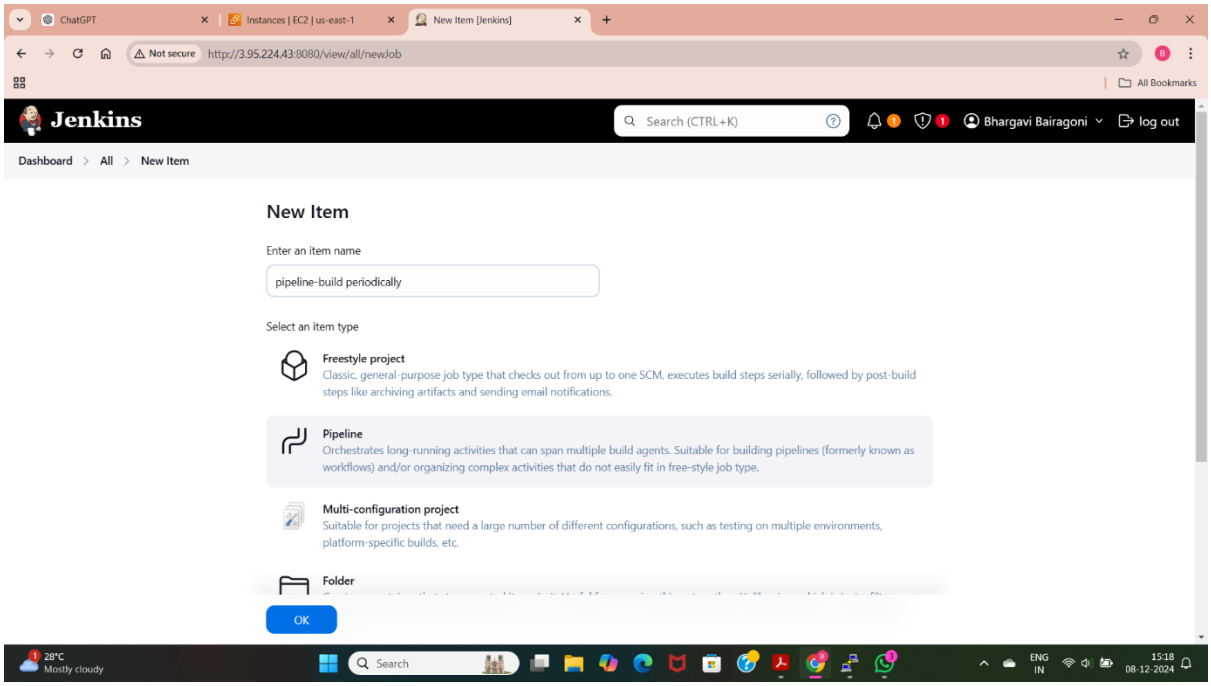




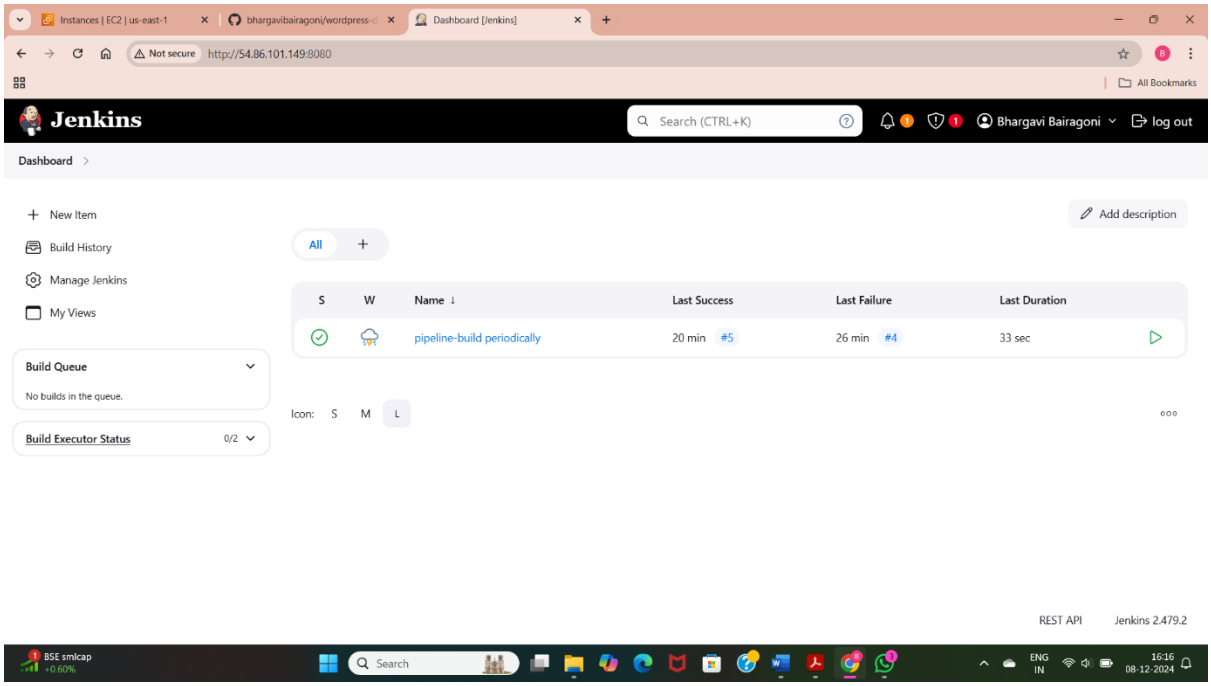
- ✓ This is the GUI of Jenkins, which is installing required plugins.



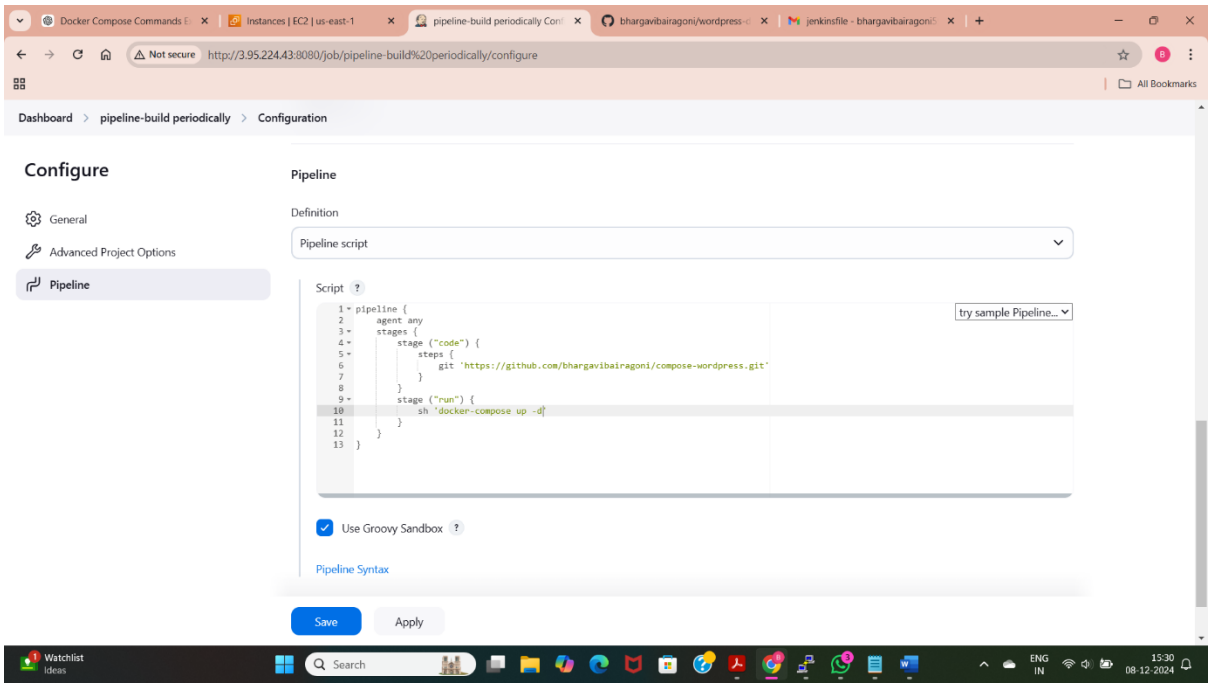
✓ Now we can able to access the Jenkins after installing required plugins. Now click on the “start using Jenkins”.



✓ Now am creating a pipeline job with the name “pipeline-build periodically”.

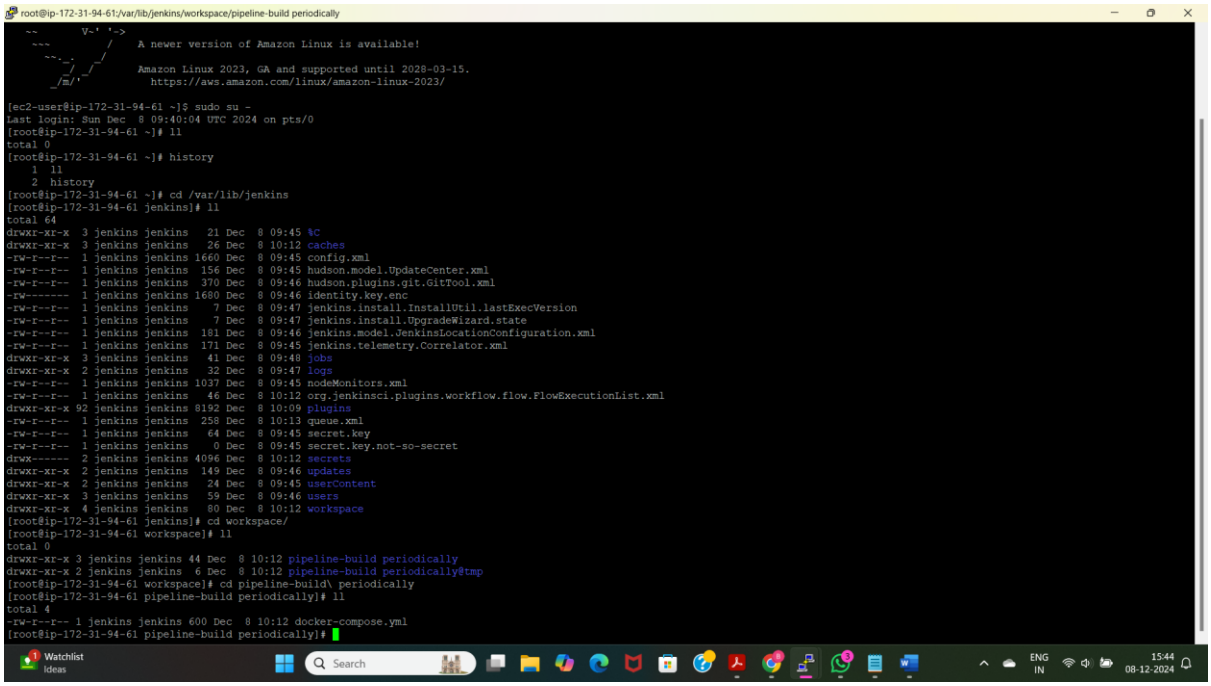


- ✓ This is the dashboard of Jenkins which shows the created jobs and we can also install the plugins, specify configurations by using Manage Jenkins. We can also check with the build history.
- ✓ By going inside the above job we can deploy our application and check with the continuous integration and continuous delivery/deployment.

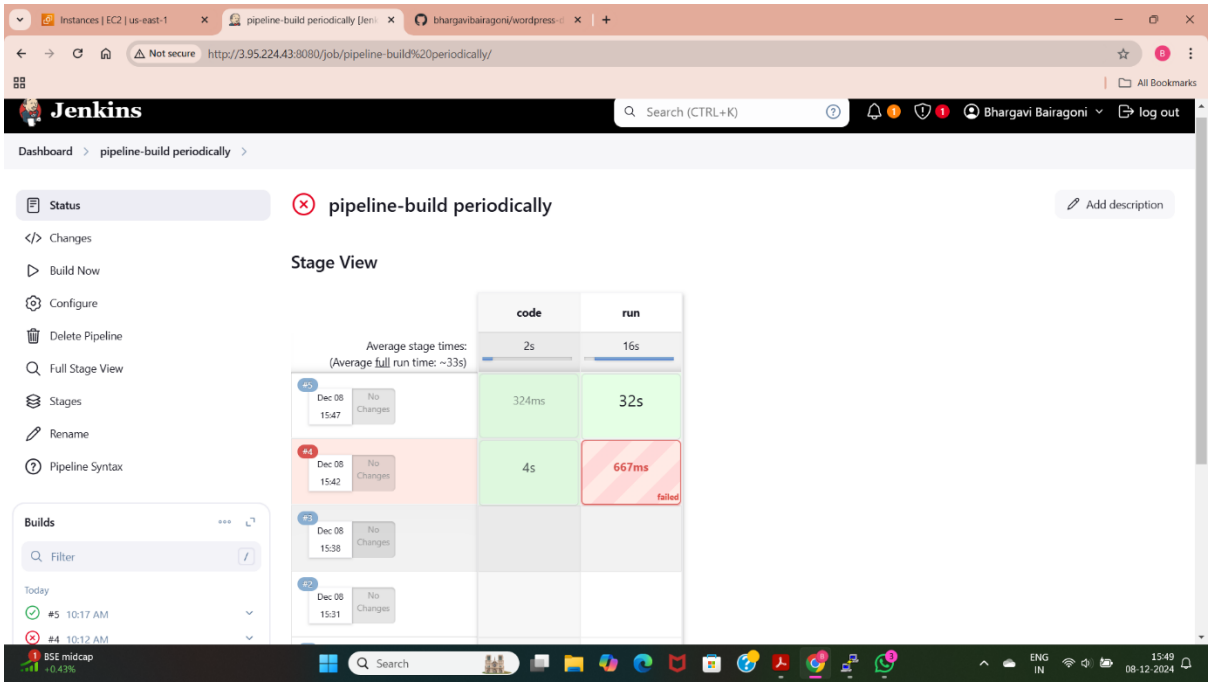


- ✓ This is the script for pipeline using some syntax, which is taking the code from github and executing compose file with the specific command.
- ✓ Jenkins is having two types of pipeline scripts
    1. Declarative pipeline
    2. Scripted pipeline
- ✓ Pipeline syntax includes pipeline
    1. agent: Defines where the pipeline runs (e.g., any, none, or a specific node).

2. stages: Contains multiple stage blocks, each representing a build phase(like code, build, test).
3. steps: Instructions for what to do in a stage (e.g., sh, git, echo).

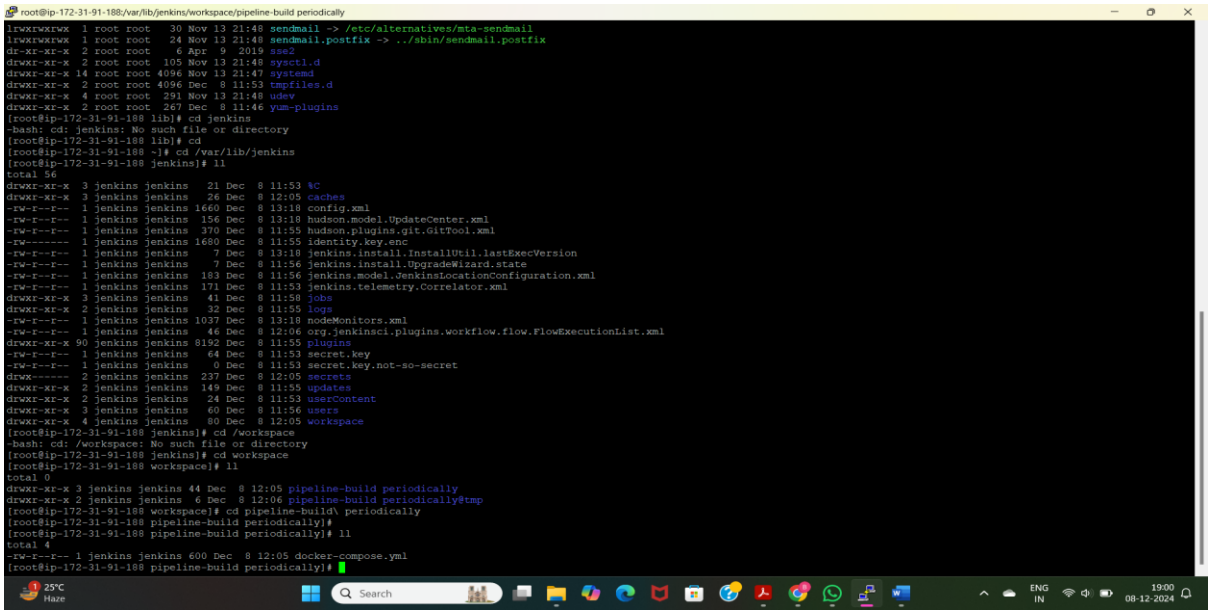


✓ This terminal shows that the compose file is stored in the Jenkins workspace. We can check with the applications, files that are imported into Jenkins pipeline or freestyle.

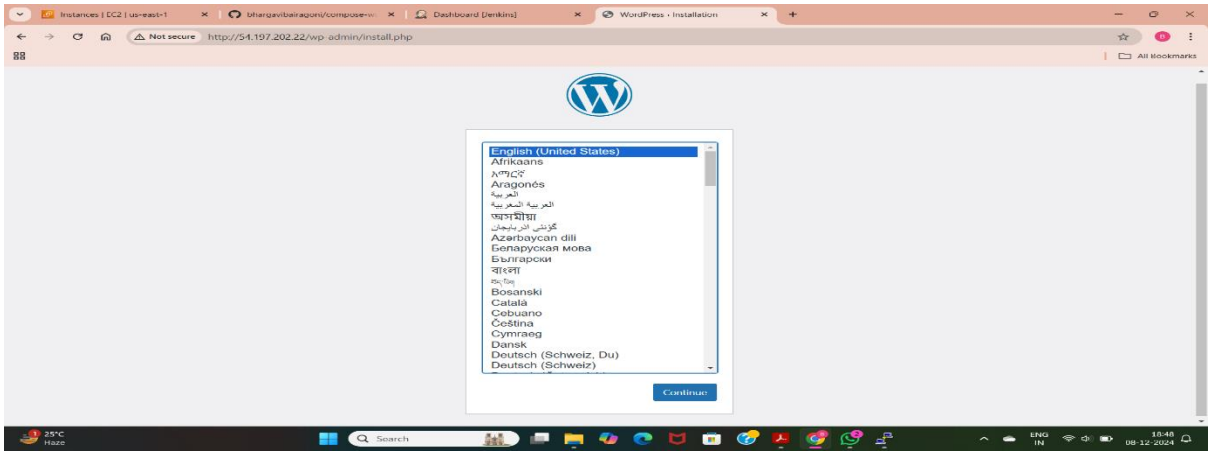


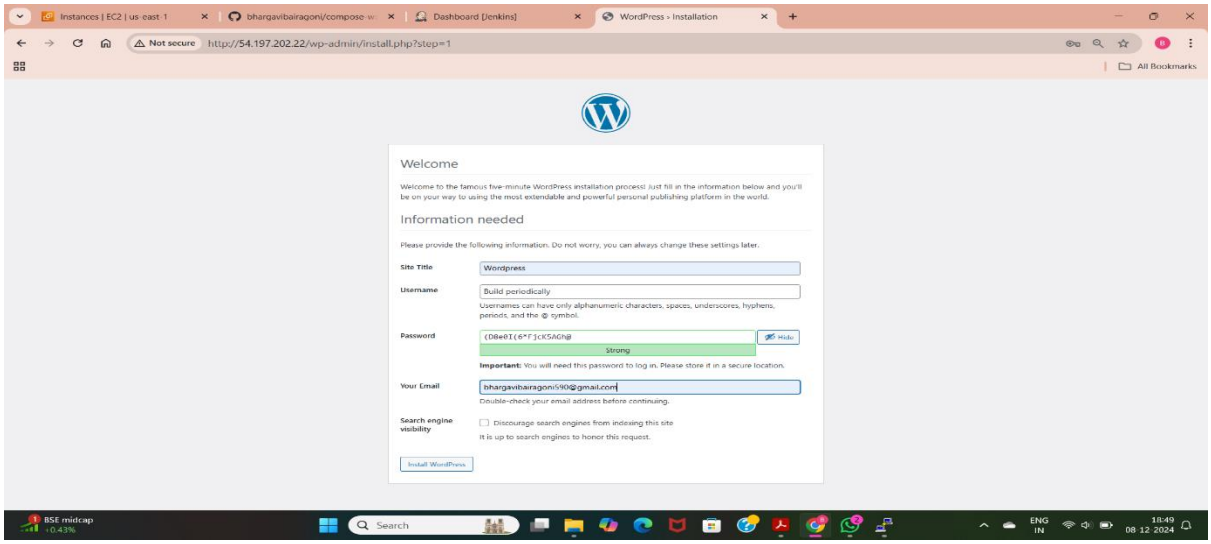
✓ I have successfully build the pipeline. Here I have used a plugin “pipeline stage view” to see the stages of the pipeline. Here am used two stages.

1. Code: Here I have taken the code from github.
2. Run: Here I have used execute command to execute the application.

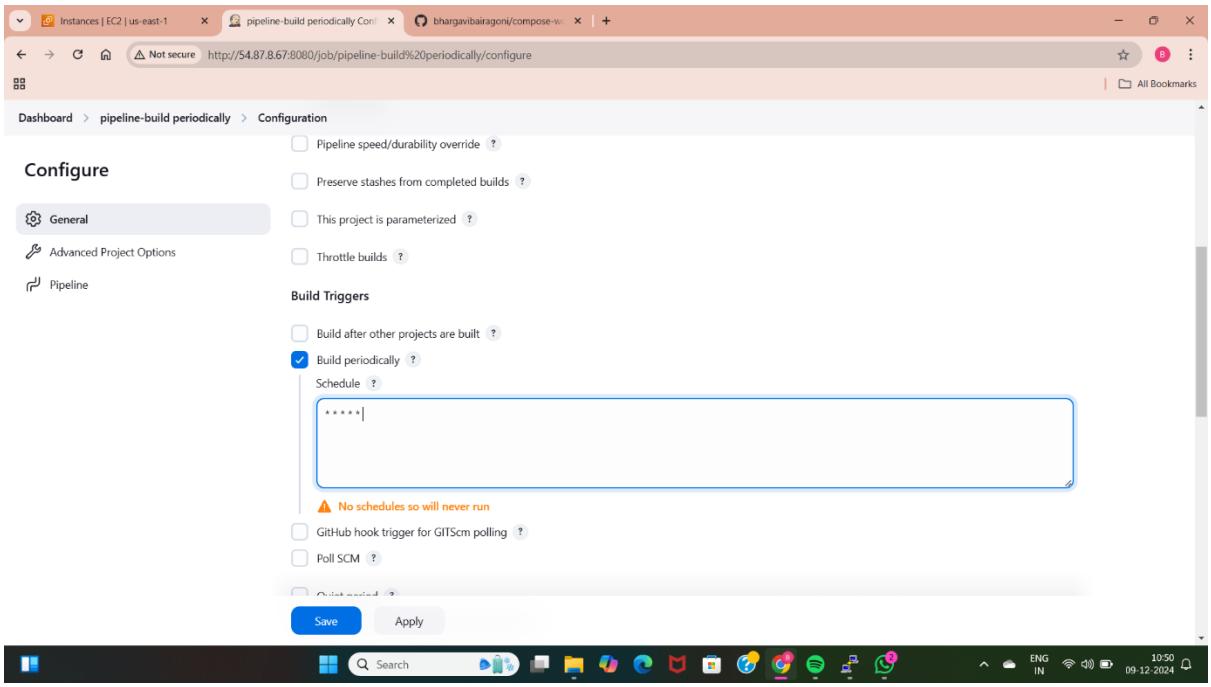


- ✓ We can see the compose file which is used in the Jenkins through the Jenkins path. Here I have used the docker-compose.yml file.

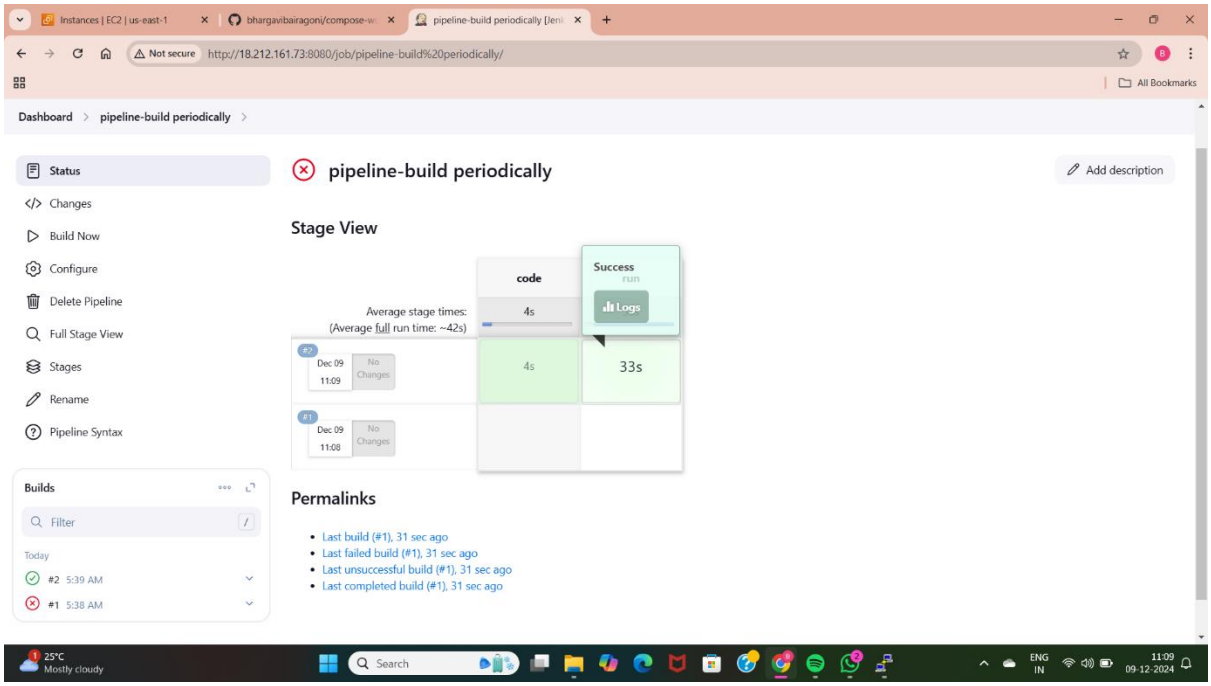




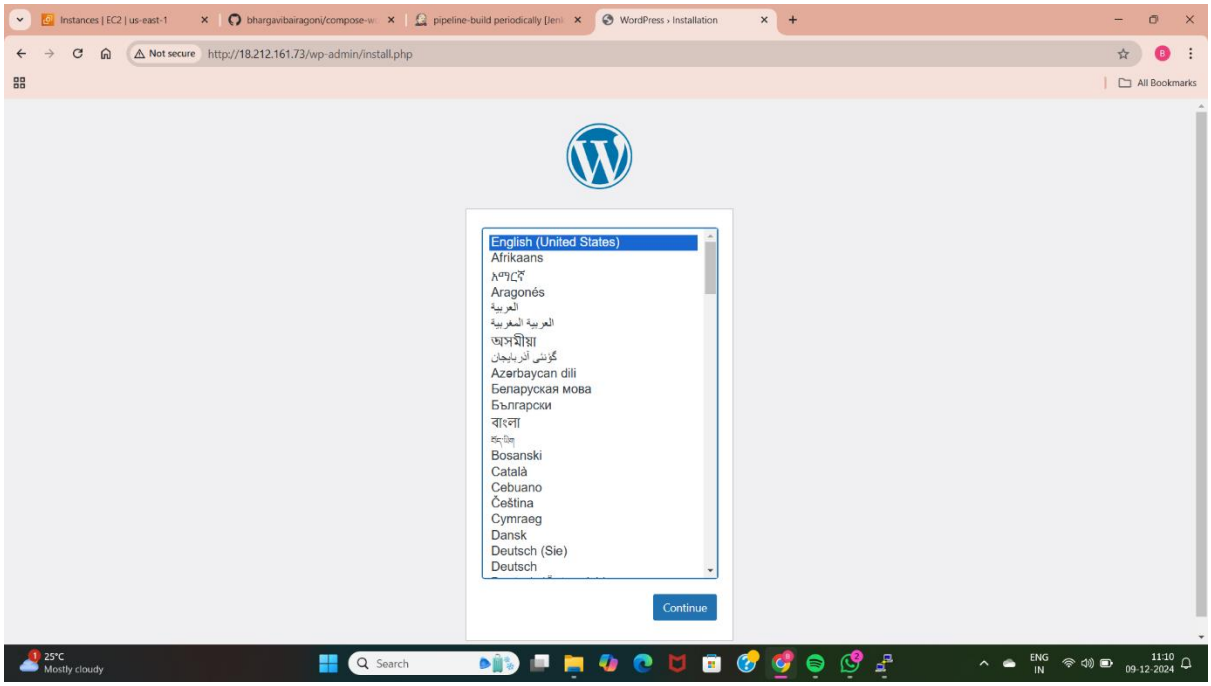
- ✓ Now by pasting the public ip address of instance am able to access the wordpress application which is deployed using build periodically.



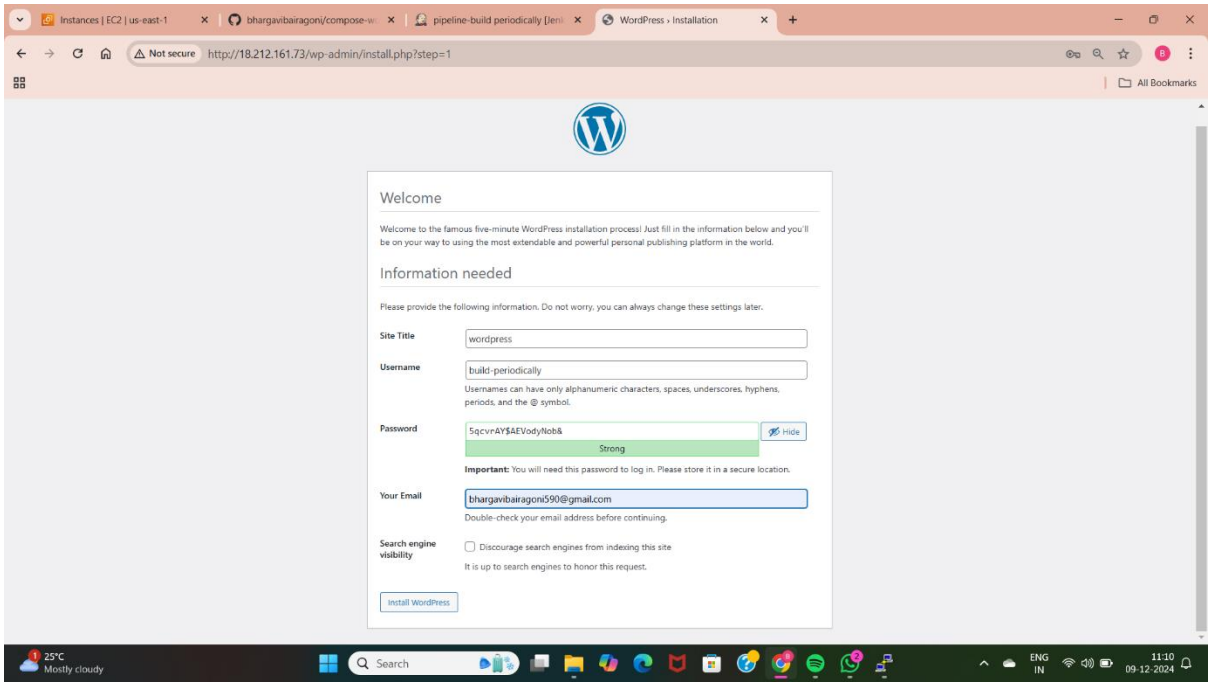
✓ Here am using build periodically method to trigger the builds, am used the crontab syntax (minute hour day month week of the day) * * * * * so that the builds take place for every minute with or without any changes in the github code.



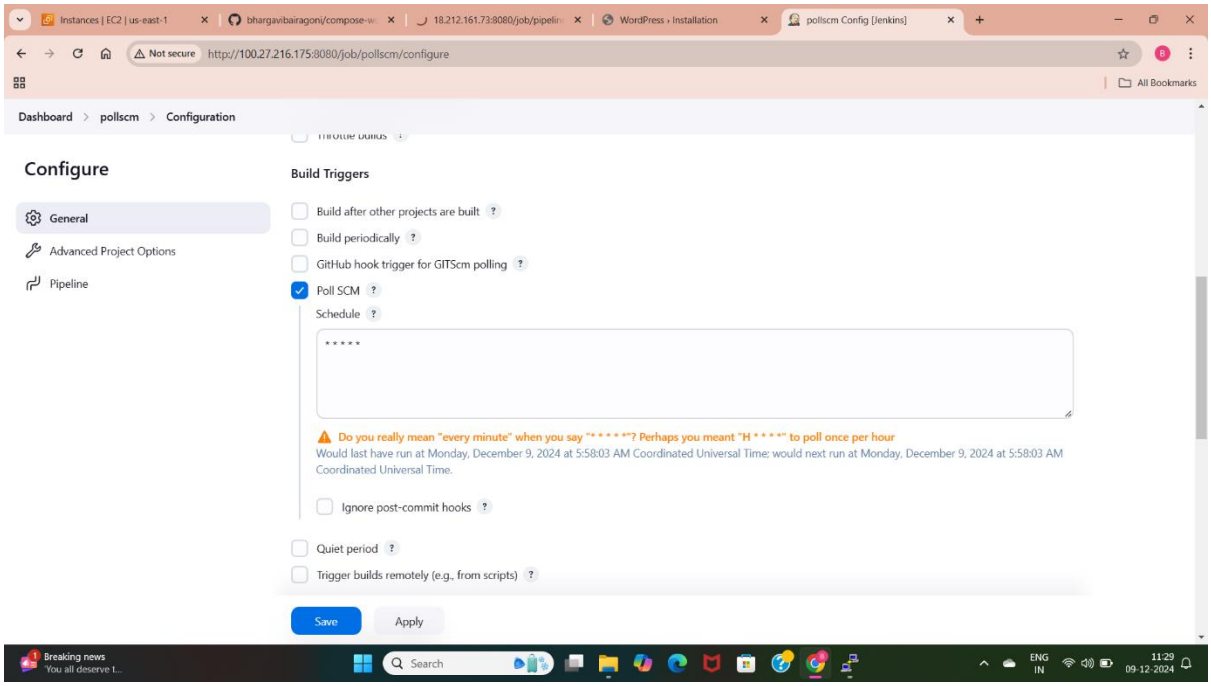
✓ After using build periodically trigger method we can able to check with the builds. Here am getting the builds for every minute.



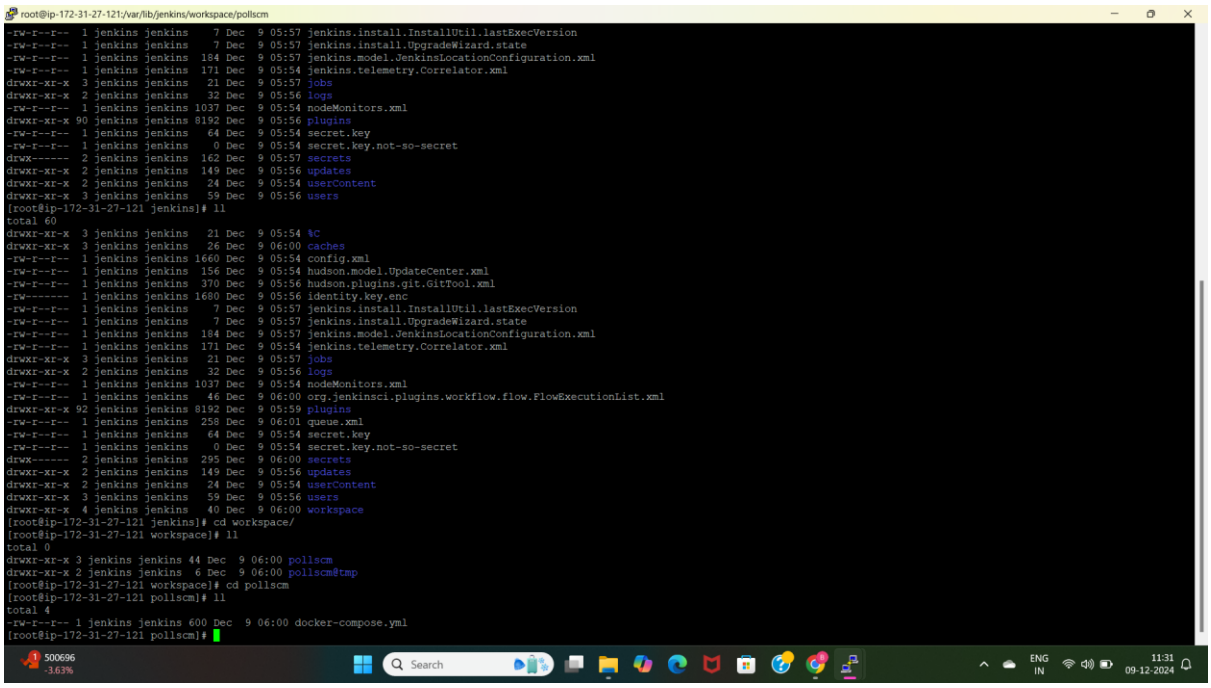
- ✓ By copying the public Ip address of the instance we can access the application, which is having the Jenkins configuration installed and set in that instance.



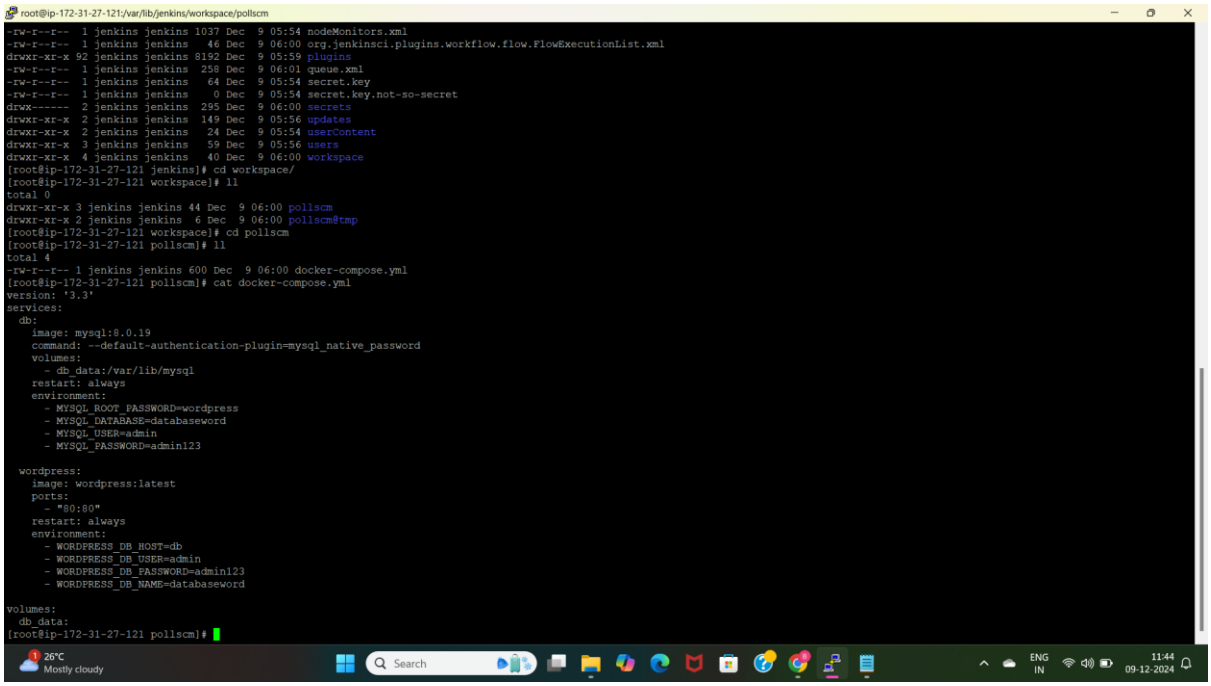
- ✓ We can able to access the wordpress application once the build is successful.
- ✓ After the build is successful am able to see the wordpress application is deployed successfully.
- ✓ Now change the build trigger method and observe the changes in the build functionality of the code.



- ✓ Here am using the build trigger method “poll scm” which is used to trigger the changes only when we update the github code.
- ✓ Here am using a crontab syntax(* * * * *) which enables the build for every minute when there are any changes in the code.



- ✓ We can also check with the Jenkins path that our github code is stored in the created job with the name “pollscm”
- ✓ We can also find the details of the builds in the workspace of the Jenkins.

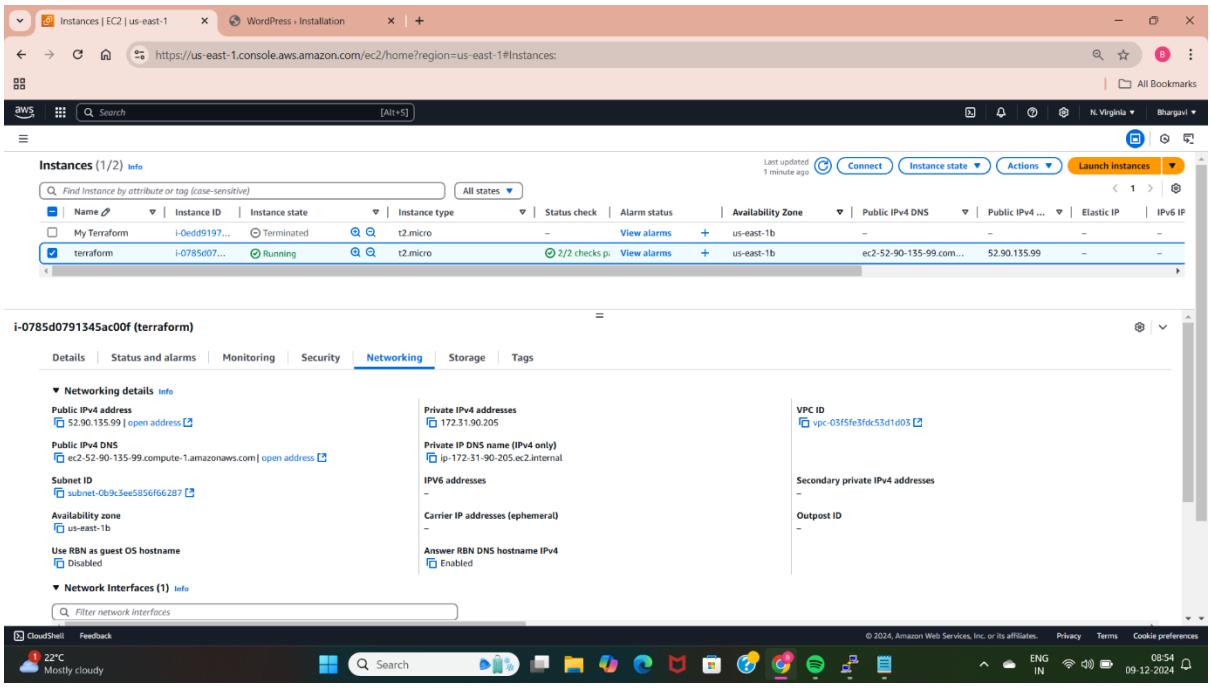


- ✓ This is the code taken from the github. After changing the code we can check withbthe change in the build.

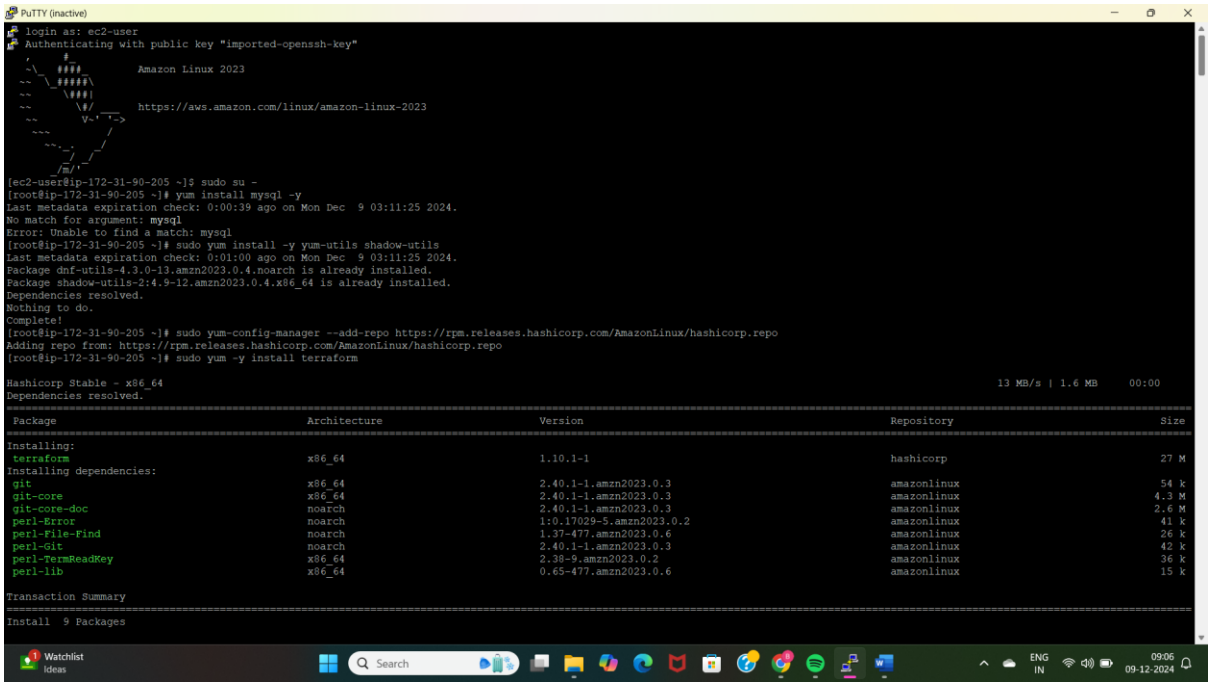
METHOD-7: DEPLOY WORDPRESS WEB APPLICATION BY USING TERRAFORM (CREATE EC2 INSTANCE ALONG WITH USERDATA .SH FILE

Terraform is an opensource infrastructure as a code(Iac) tool which is used to automate the process of creating, launching any AWS services and we can also deploy any application using the terraform.

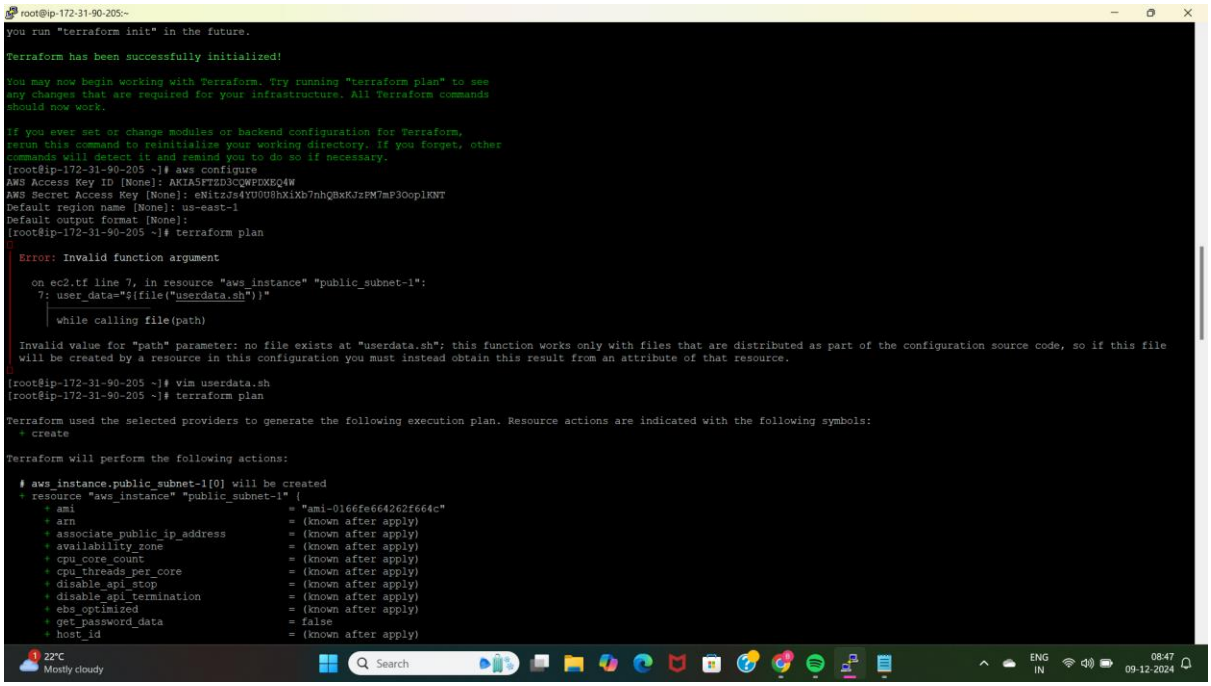
- To create any services first we need to launch a EC2 instance with basic configurations.



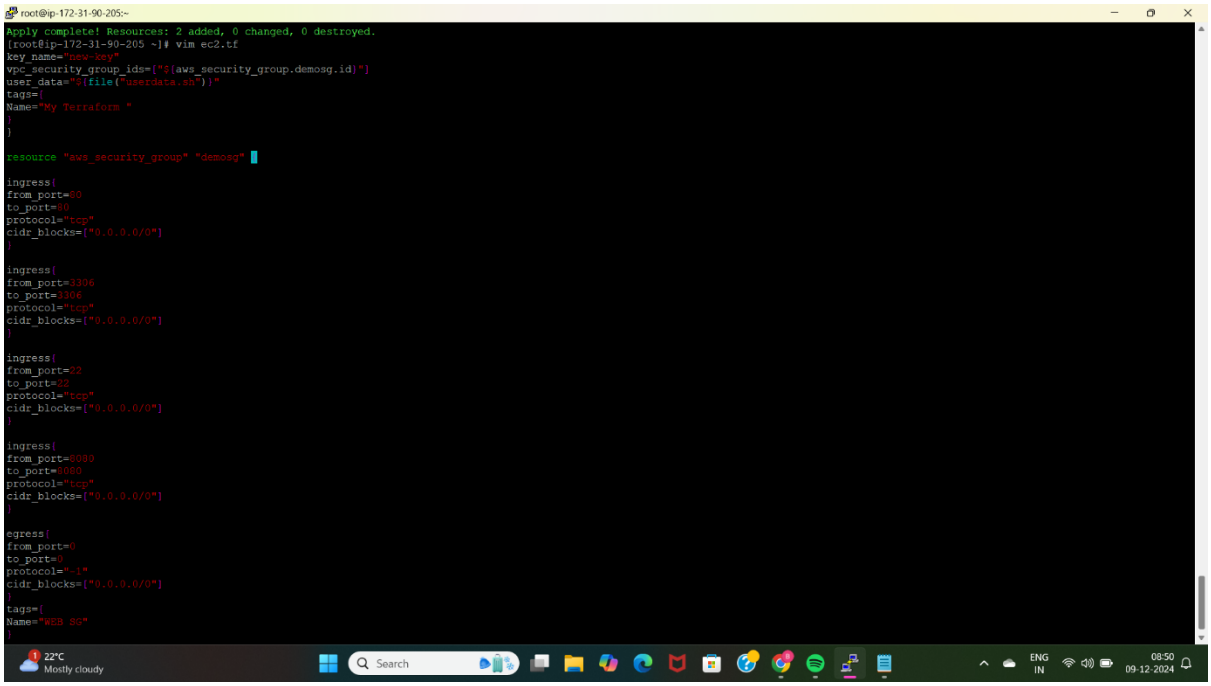
- I have launched a EC2 instance with amazon-linux application os image. We can see the details of the launched EC2 instance in this picture.



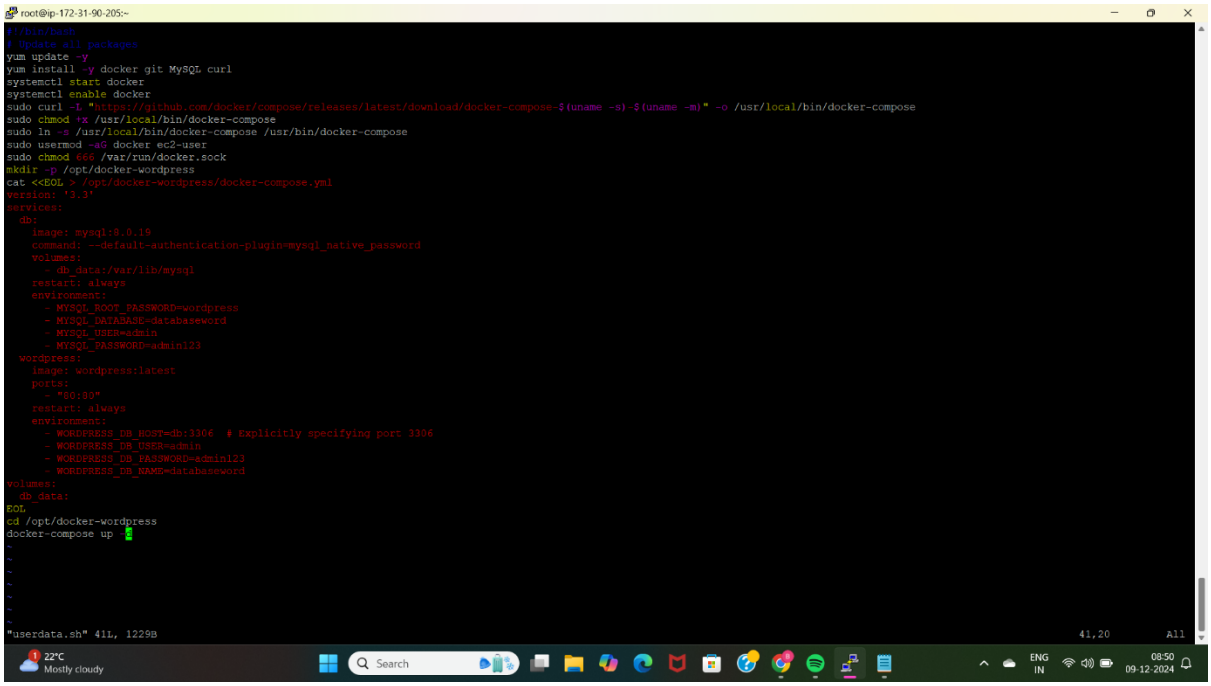
- I have connected the created EC2 instace with the terminal and installed the terraform with the above commands.



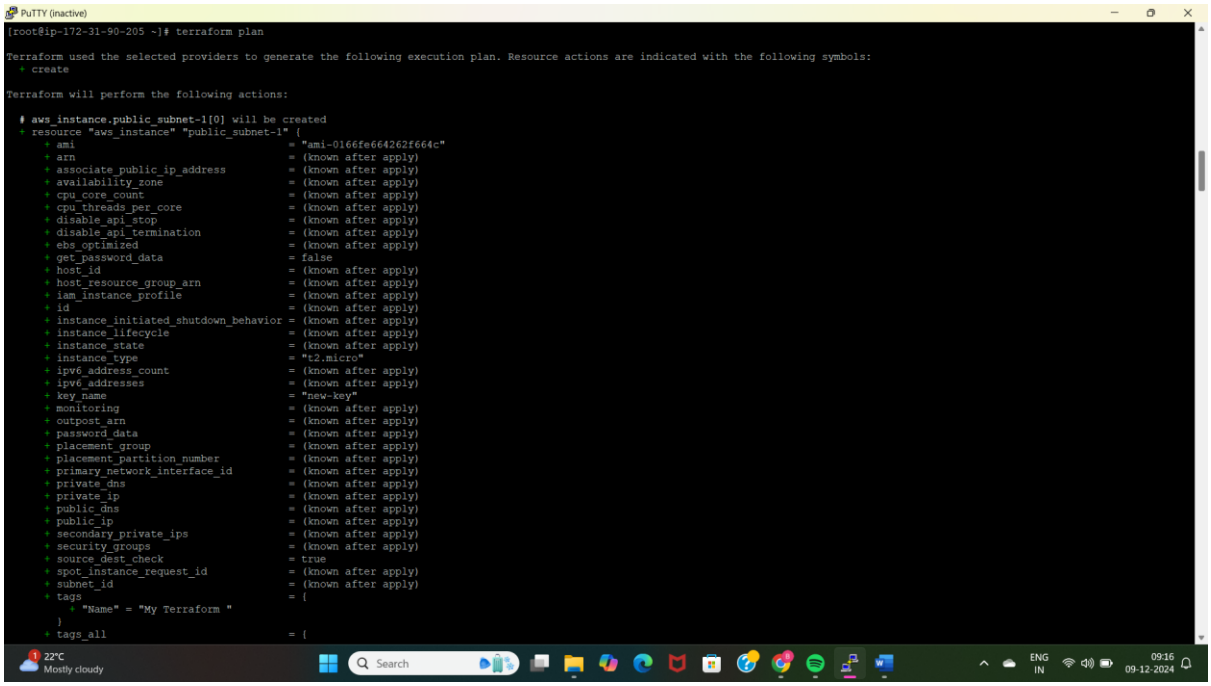
- After installing terraform, provided the authentication for using aws services with the command “aws configure” which uses the access keys and secret access keys for giving permission for providing authentication.



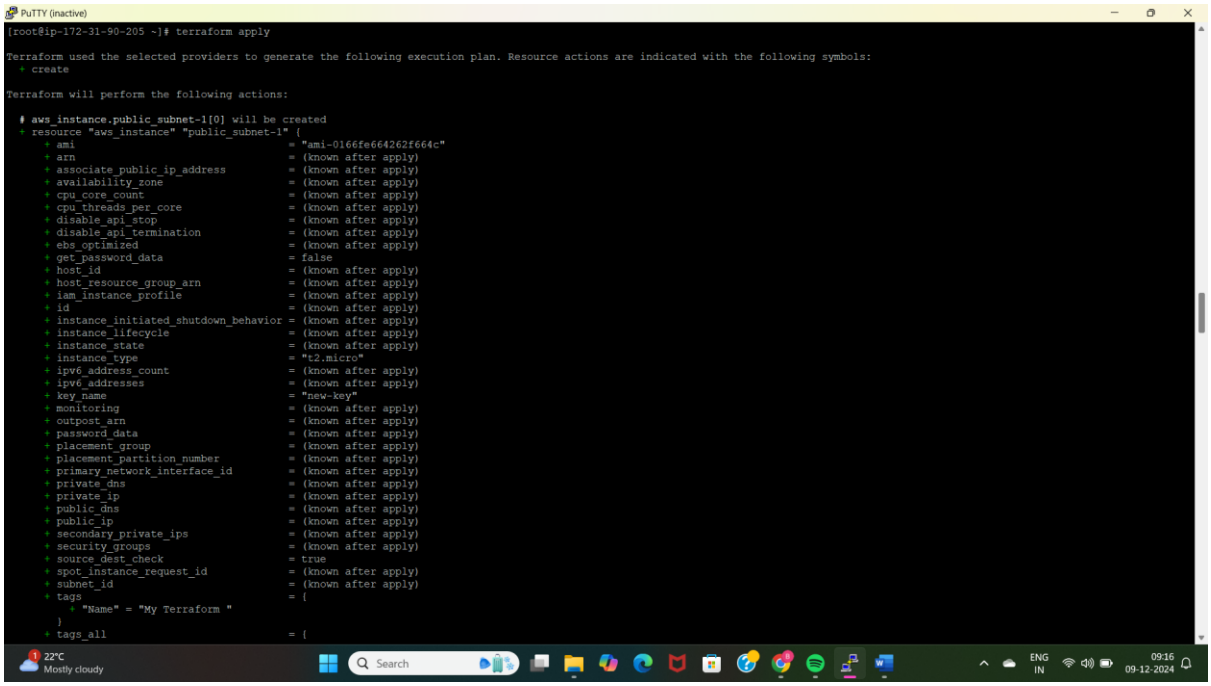
- After installing terraform I have created a ec2.tf file for creating a EC2 instance with amazon-linux 5.10 application os image AMI ID, t2.micro instance type, and provided the security with inbound ports with SSH(22),HTTP(80),MYSQL(3306) and initialized terraform.



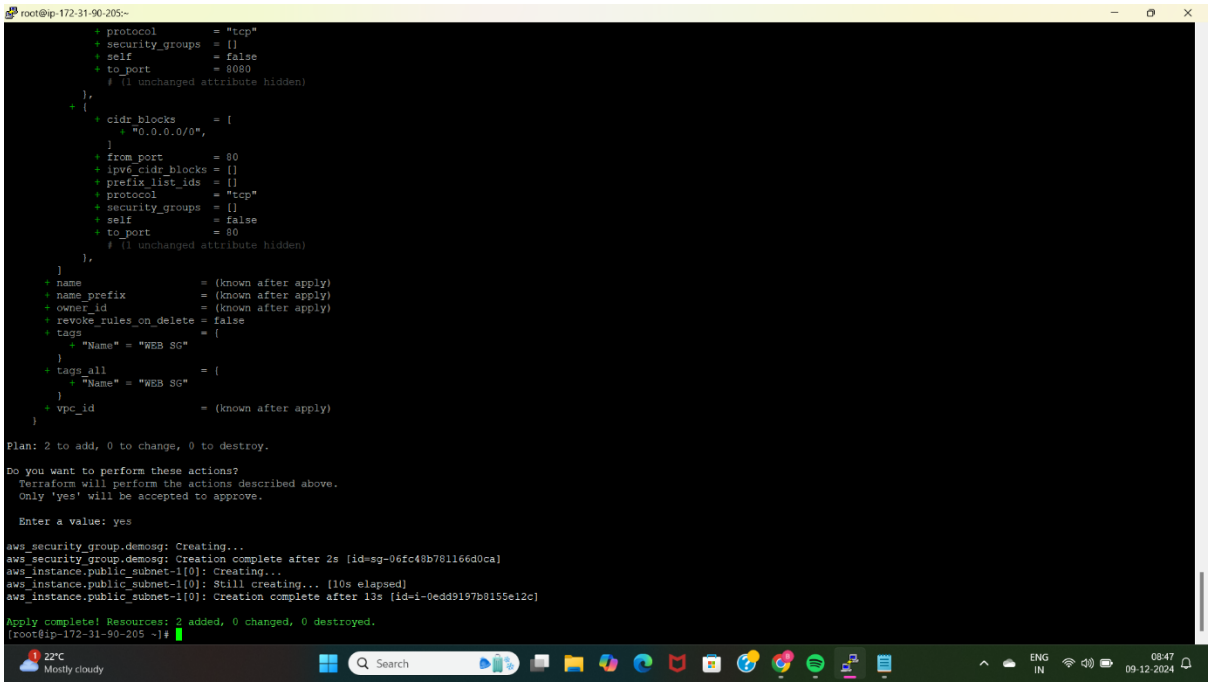
- Created a userdata.tf file by configuring the installations like mysql, docker, docker-compose.
- I have created the docker-compose.yml file inside the userdata.
- Created a directory with /opt/docker-wordpress so that I an store the docker-compose.yml file inside this directory.
- Moved inside the above created directory and performed “docker-compose up -d” to deploy wordpress application using the terraform.



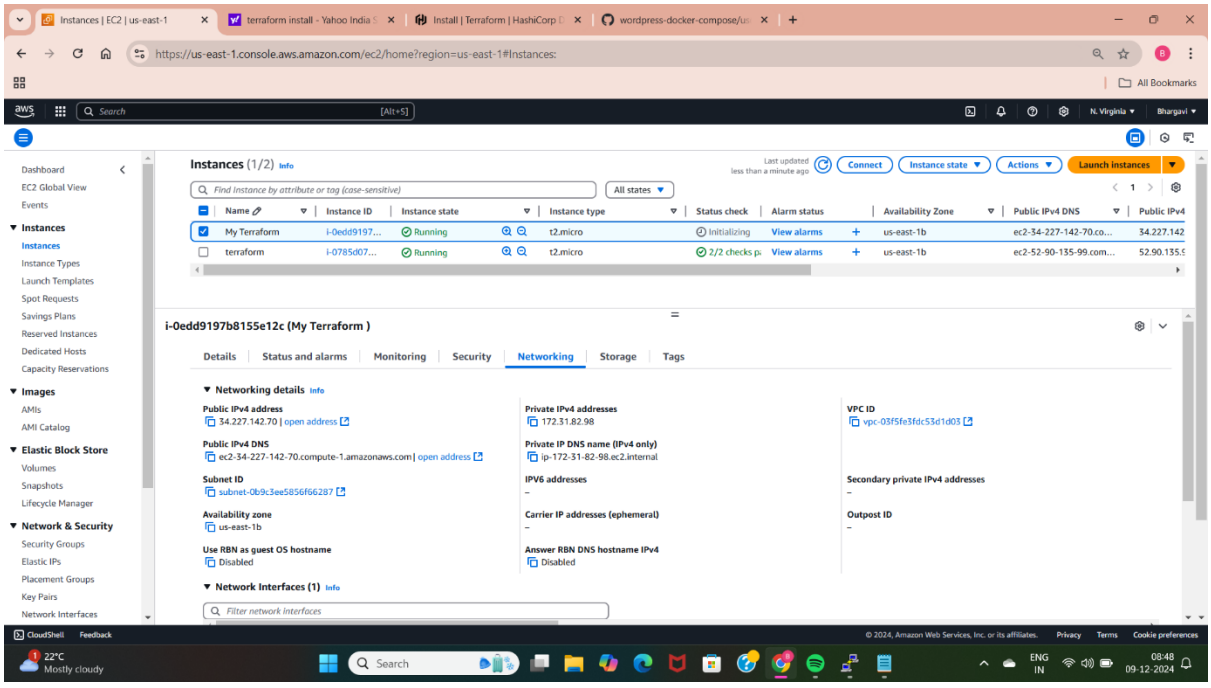
- Here I have used the command “terraform plan” to check what AWS service or resource is going to create.
- We can see the AMI id of instance, name of instance used in the code.



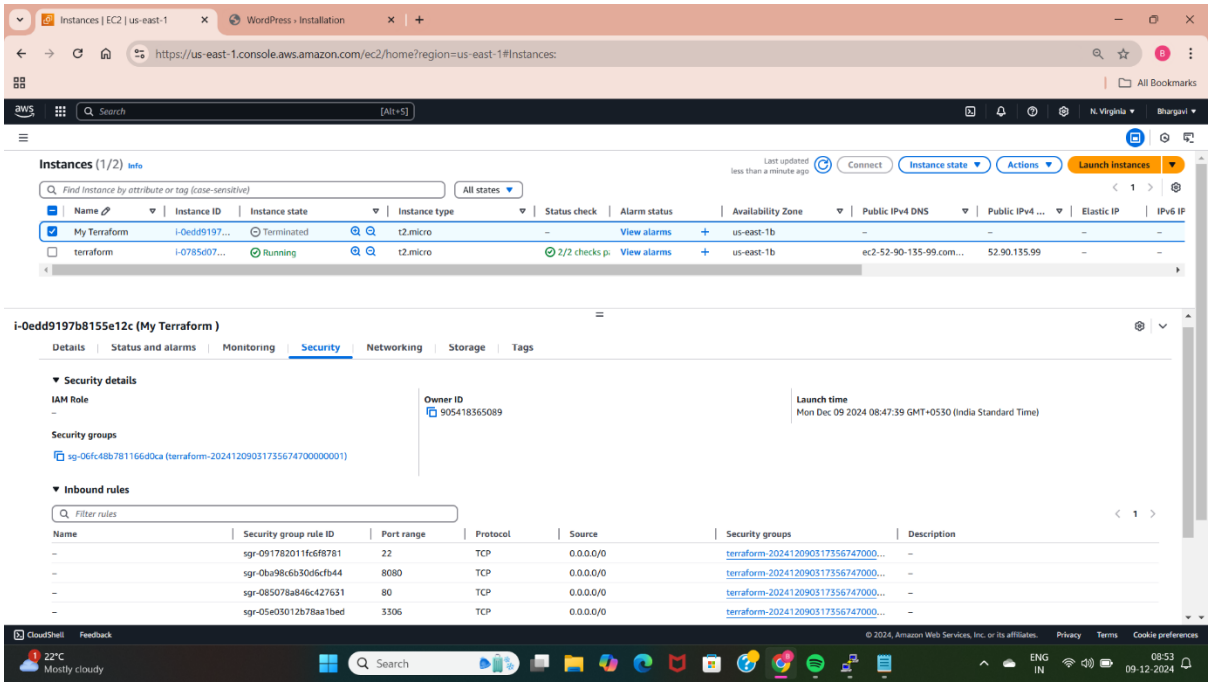
- Here I have performed the command “terraform apply” to apply the above planed resources.
- By applying this a EC2 instance with the specified security, application os image, instance type is created.



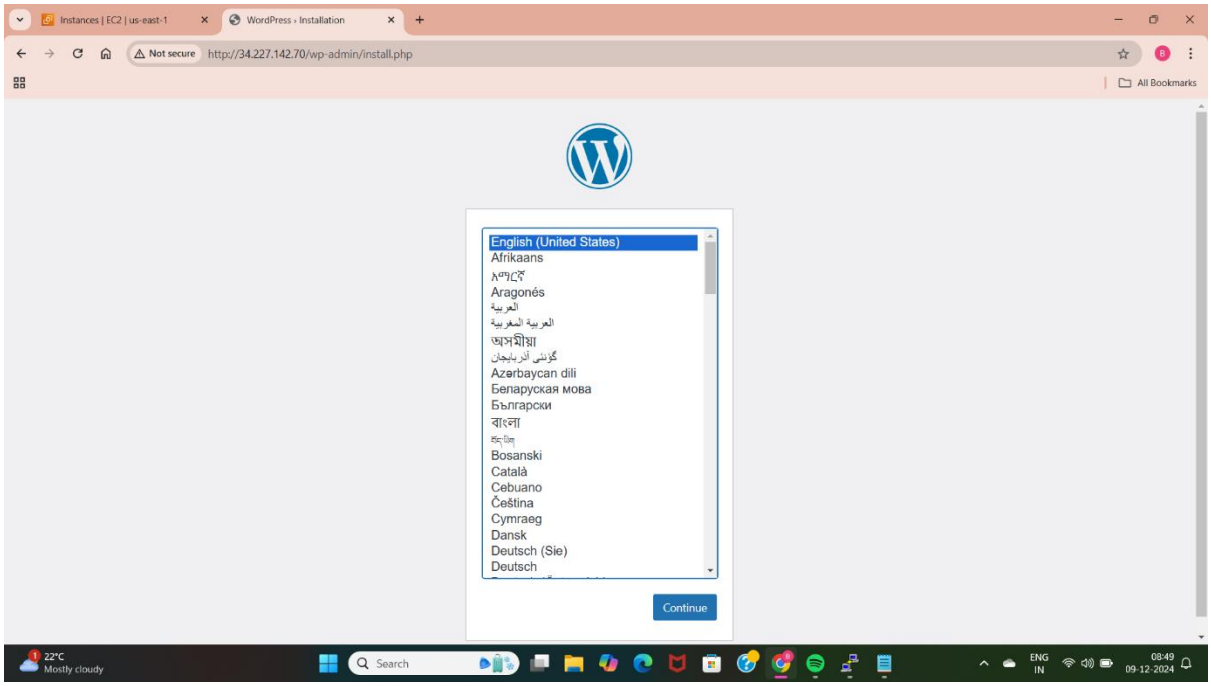
- The “terraform apply” command is successful with creating two resources one is EC2 instance with userdata.



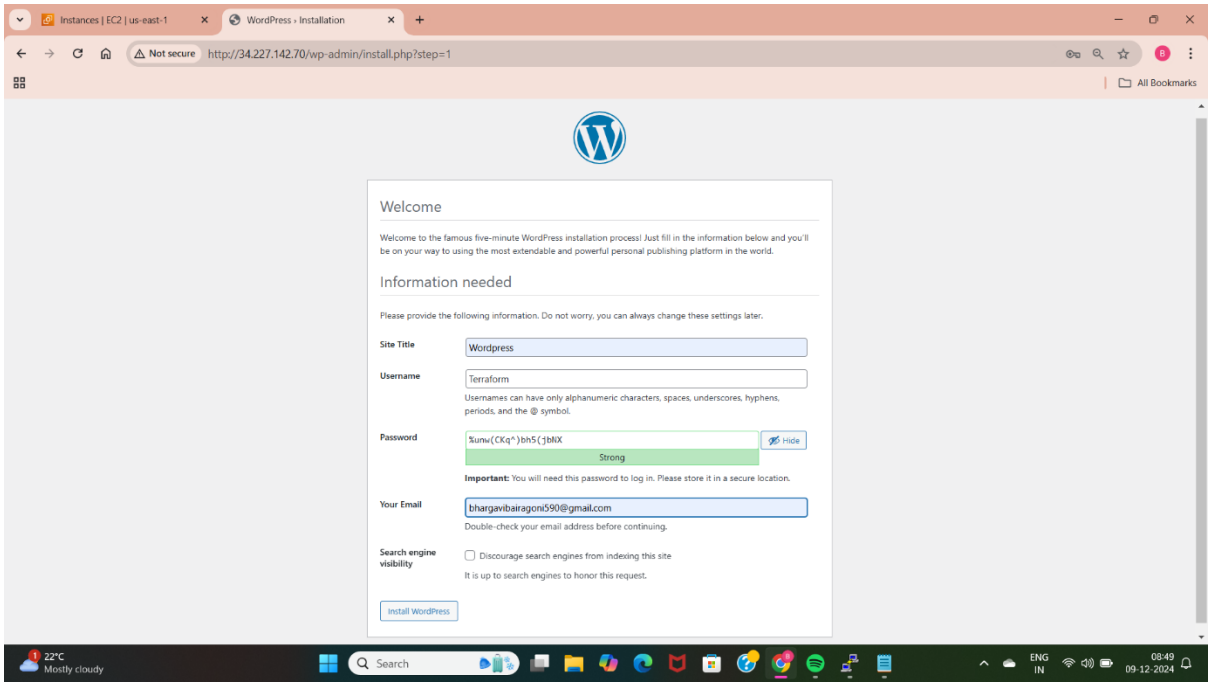
- We can check with the AWS management console whether the above terraform code is working properly or not.
- We can the instance with the name “My terraform” is created successfully.



- We can also check with the security group that is allowing above mentioned inbound rules.
- After checking with the instance and the security group copy and paste the IP address of this instance in the browser so that we can able to access the wordpress application.



- Now I am able to access the wordpress application by using the public IP address of the instance.

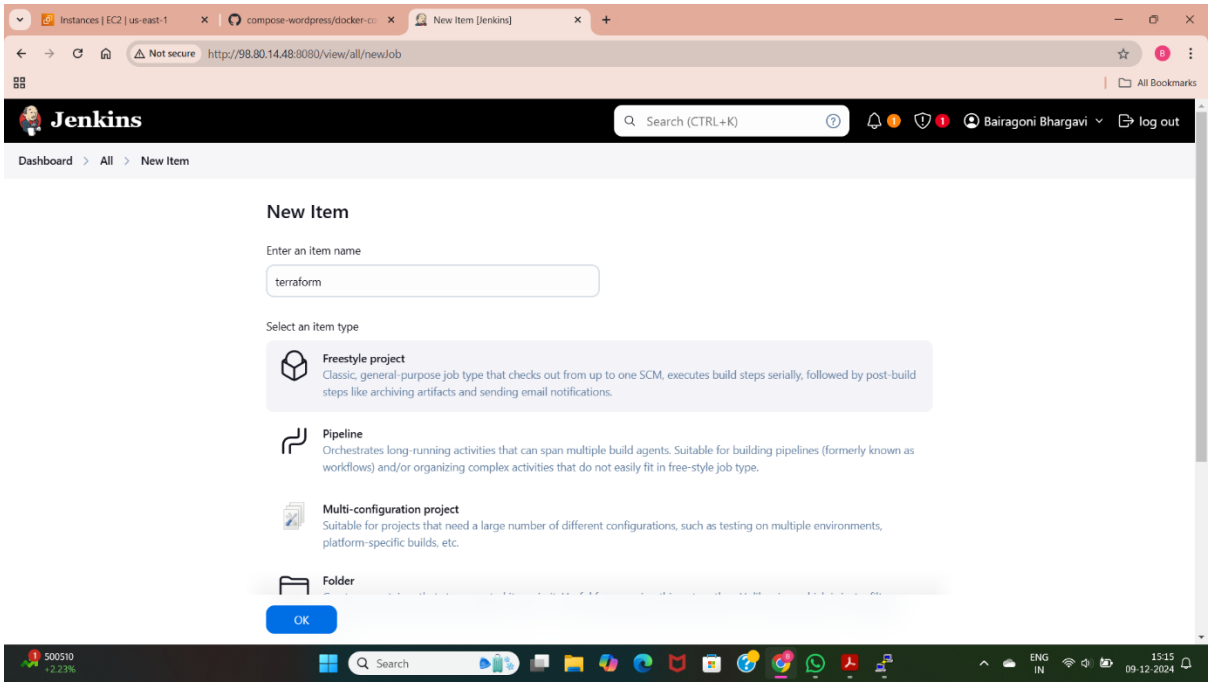


- The picture shows that the wordpress application is deployed successfully using the terraform and the userdata.

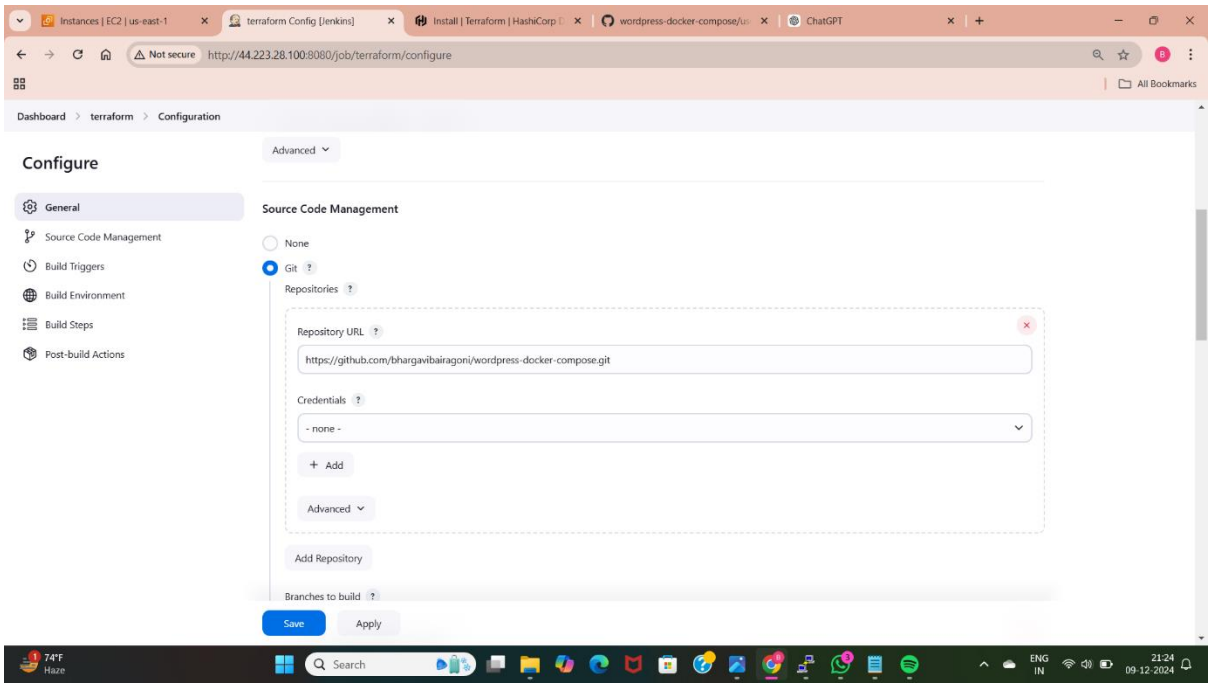
METHOD-8: DEPLOYING USING TERRAFORM AND JENKINS EXECUTE SHELL

Deploy WordPress web application by using git (clone terraform script which helps to deploy WordPress web application), jenkins (in execute shell install terraform, init, fmt, validate and apply with automatic command as terraform apply --auto-approve) and terraform.

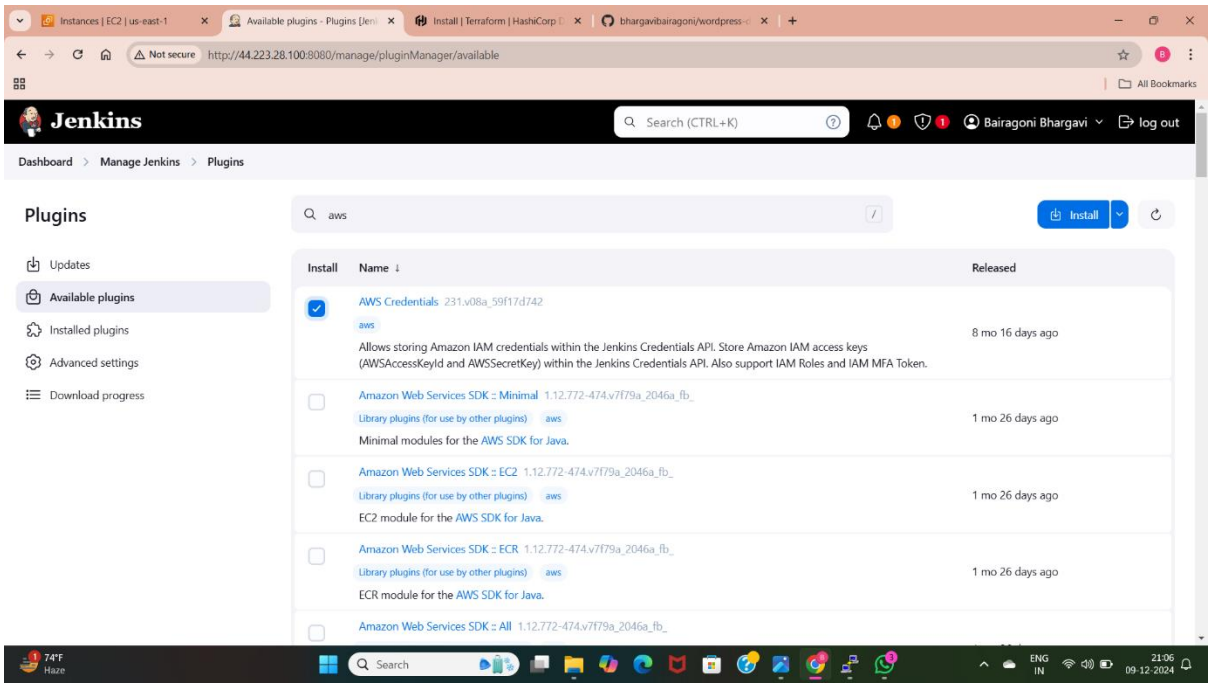
- First we need to create a instance and setup Jenkins so that we can integrate terraform with Jenkins.



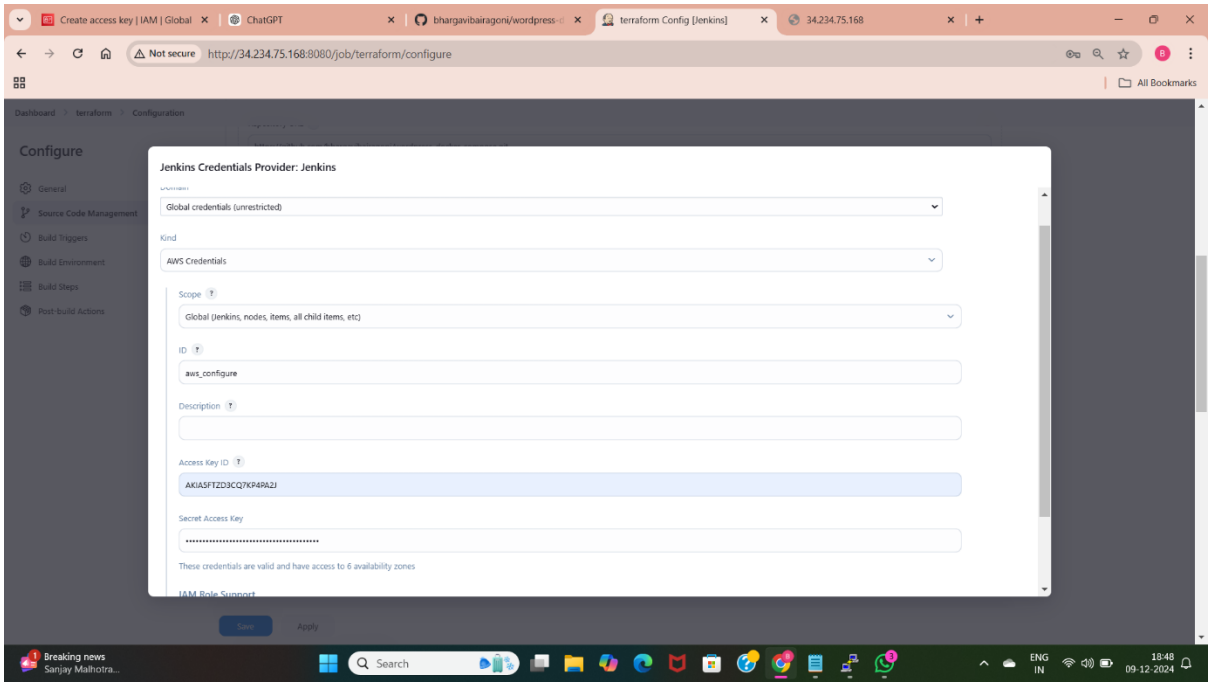
- Here I have created a Jenkins job with the name of Terraform with the freestyle project.



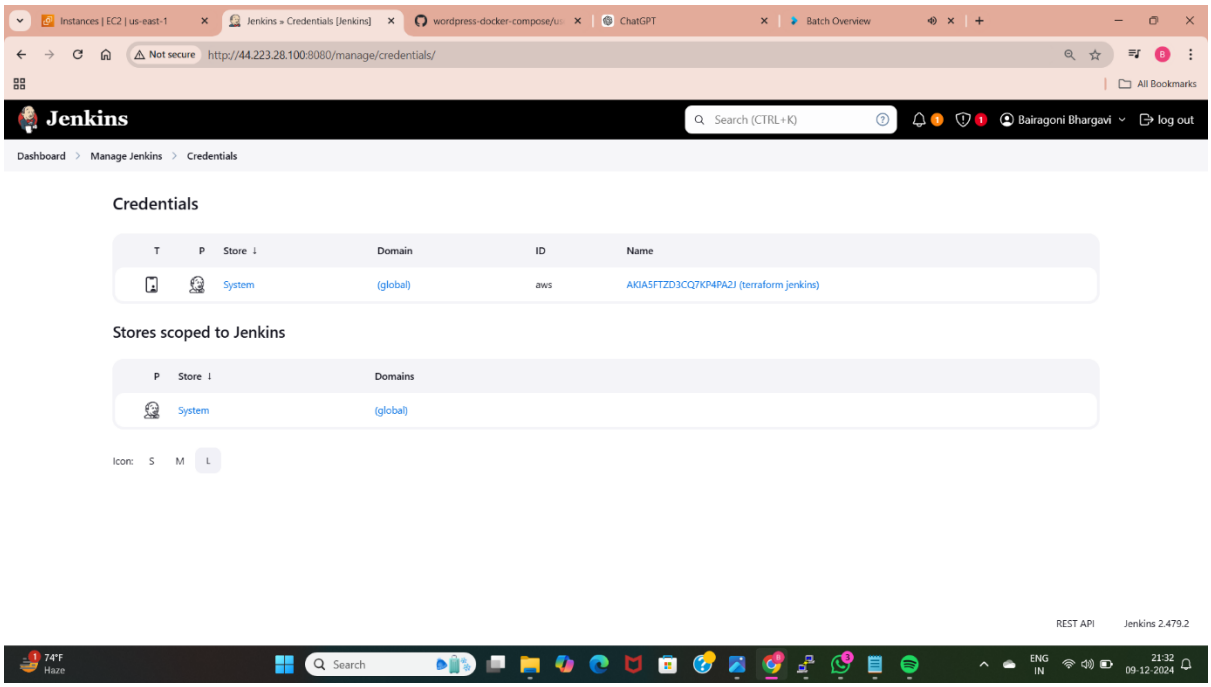
- To deploy the wordpress application using the terraform we need to take the code from the github. Here I have taken the url of the code from github using source code management.



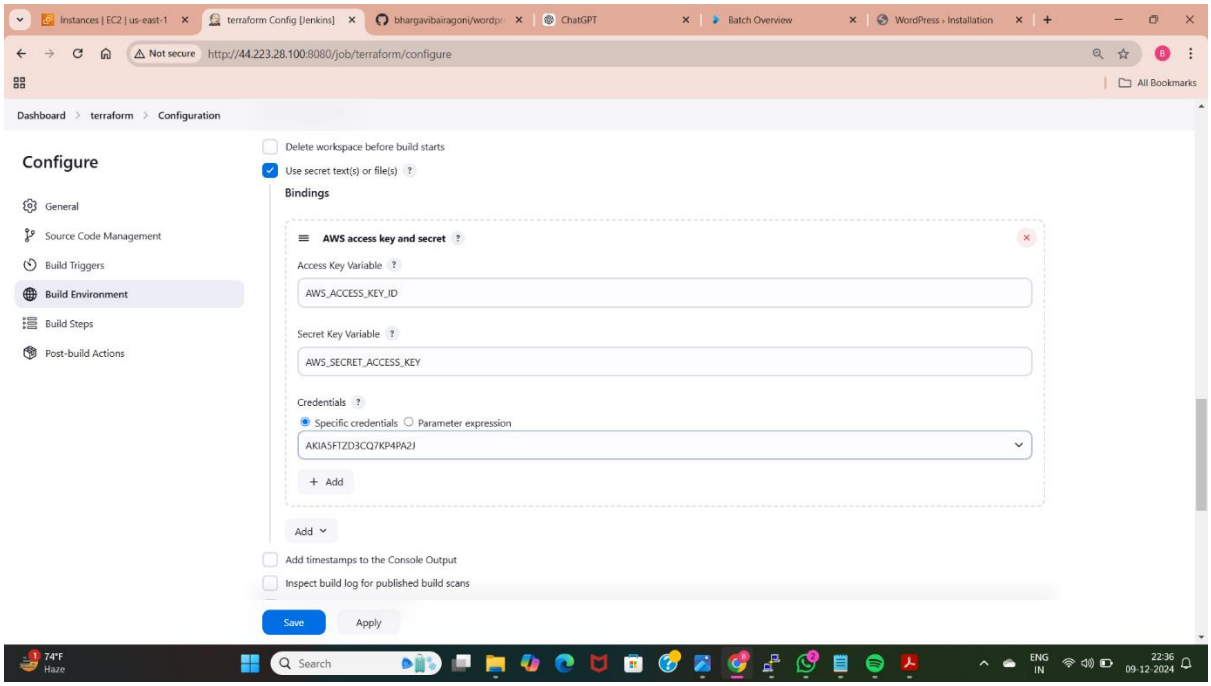
- To provide the credentials to access aws services we need to install the aws credentials plugin.
- For that we need to go to manage Jenkins, select plugins option and go to available plugins and search for aws credentials plugins and install the plugin.



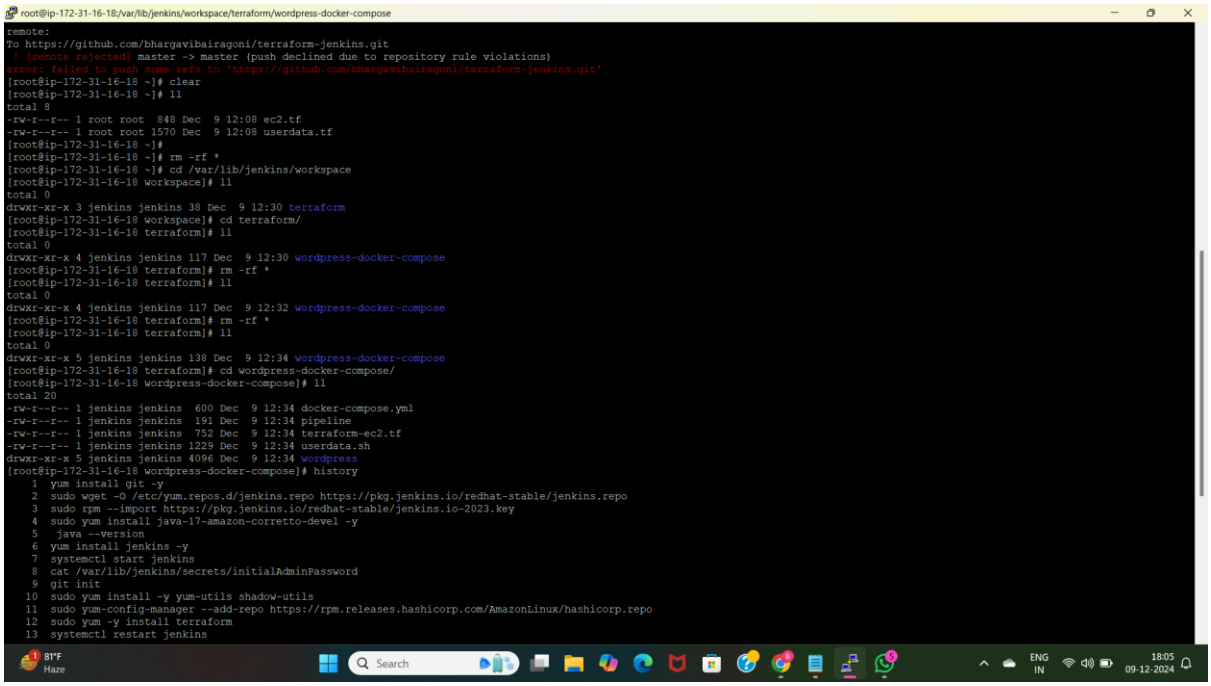
- Now by using the aws credentials plugin give the access keys and secret access keys for the Jenkins to use Aws services. For that go to management Jenkins and select credentials option , select global configurations, select add credentials and select the aws credentials plugin and provide access keys and secret access keys of aws management console.



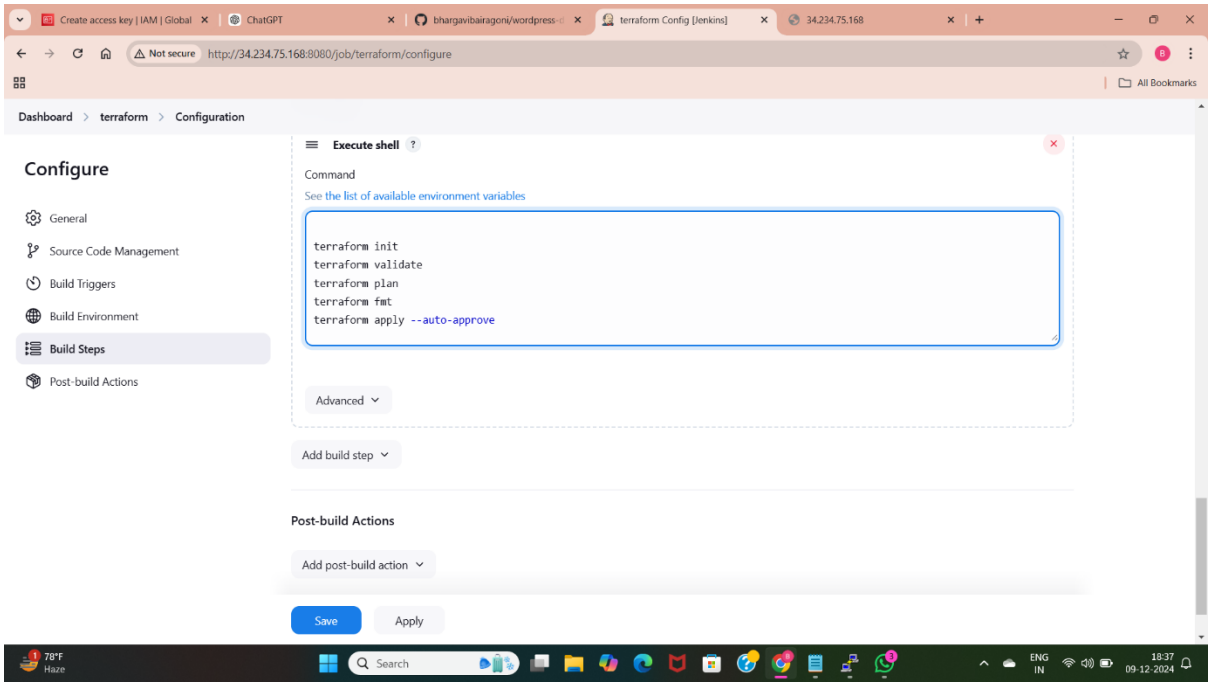
- Here I have created credentials for access keys and secret access keys.



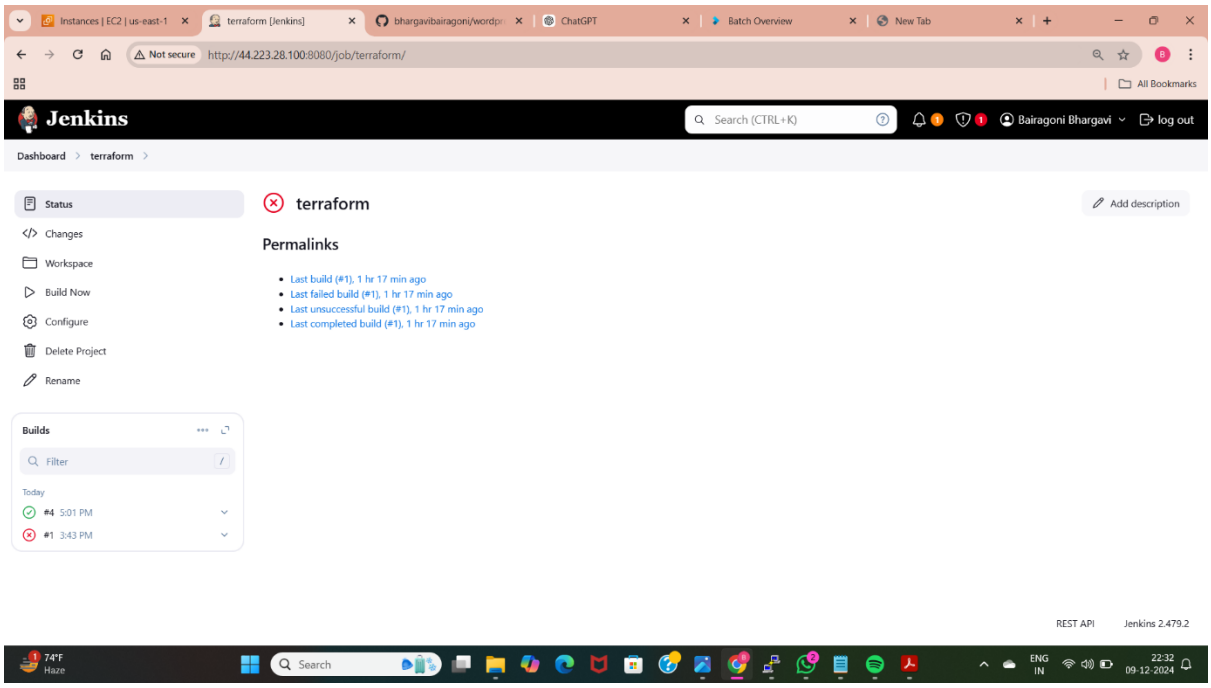
- Added credentials for Jenkins to access Aws services by using “aws configure” plugin. Added access keys and secret access key.



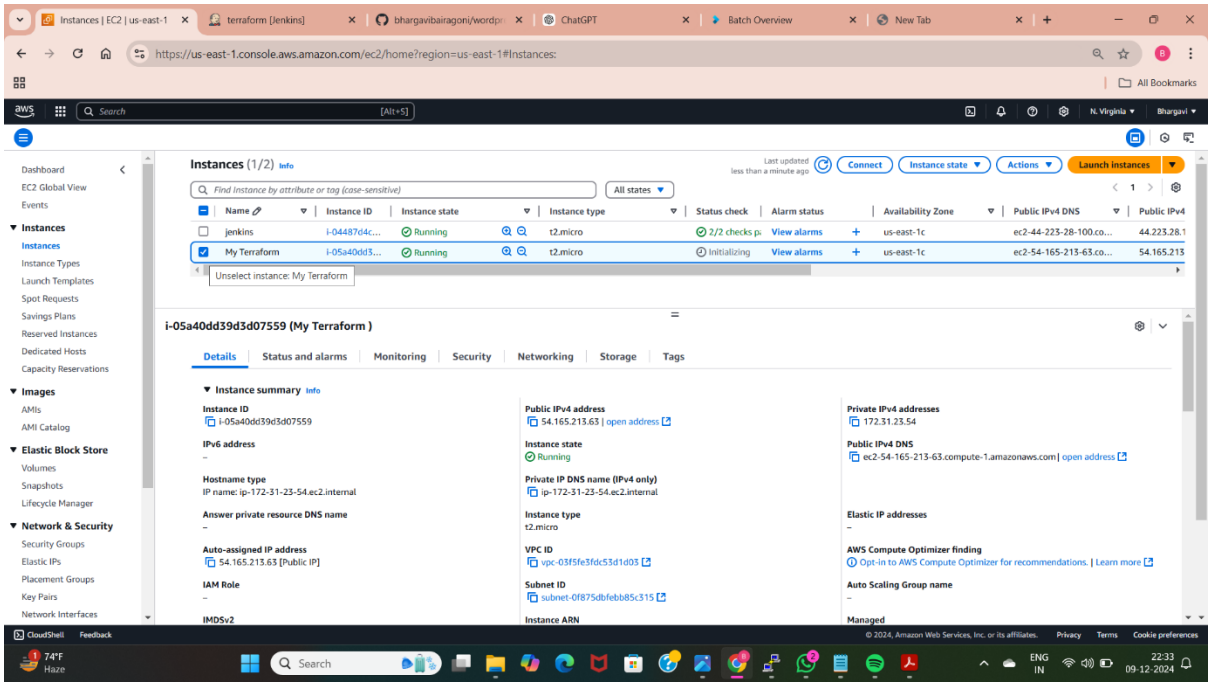
- These are the files which are pulled from the github by using the code url, and these are the commands used to install Jenkins, terraform. We can see the terraform-ec2.tf and userdata.sh files in the above picture.



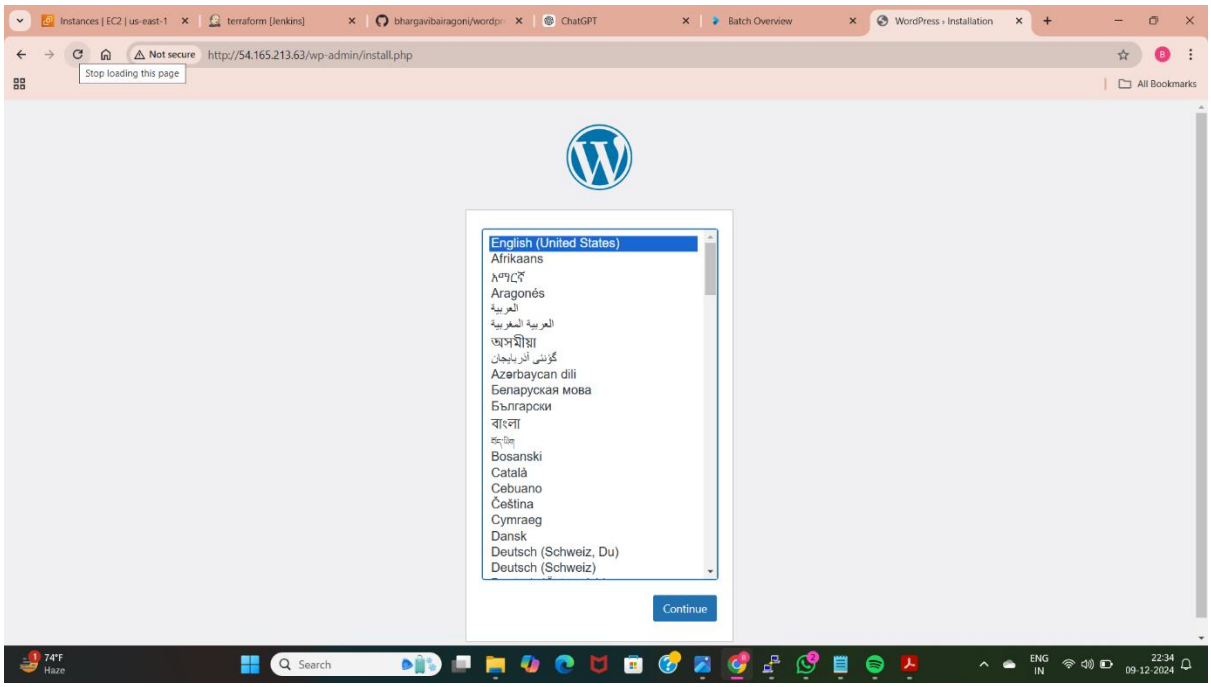
- These are commands used in the execute shell to deploy the application using the terraform.
- With the help of github url I have taken the code for creating EC2 instance and userdata.
- Terraform init is used to initialize the terraform in the created files.
- I have applied the changes by using “terraform apply” command.
- After this command a EC2 instance is created with userdata so that our application is deployed.



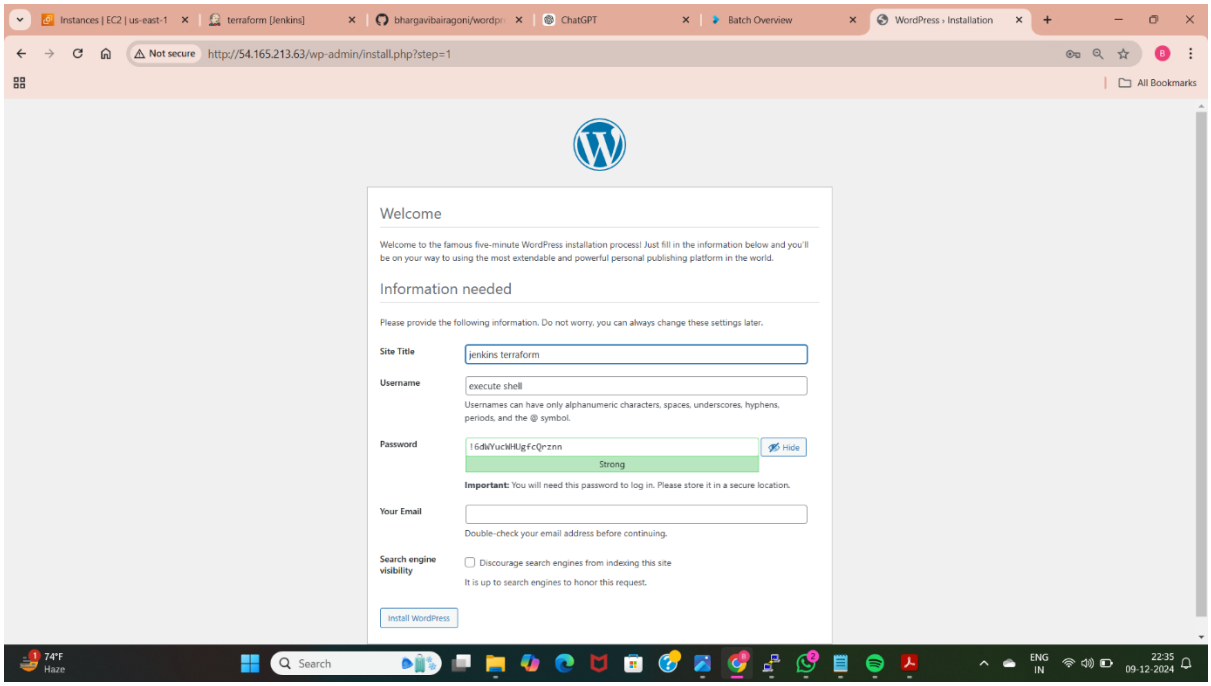
- After writing the script in the excute shell we need to build the freestyle project with the name of the terraform, so that we can able to execute the terraform files and create Ec2 instance which uses userdata script to deploy wordpress application. I have got the error while building the job and I have rectified the error and the build is success after three builds.



- After building the above execute shell we can see that the terraform code is launching a new instance with userdata which consists configurations related to deploying the wordpress application.



- After successful build we will get the new instance, we need to take the ip address of the new instance and paste in the google , so we get the above wordpress page. Select the language for accessing the wordpress, we can find the login page for wordpress.

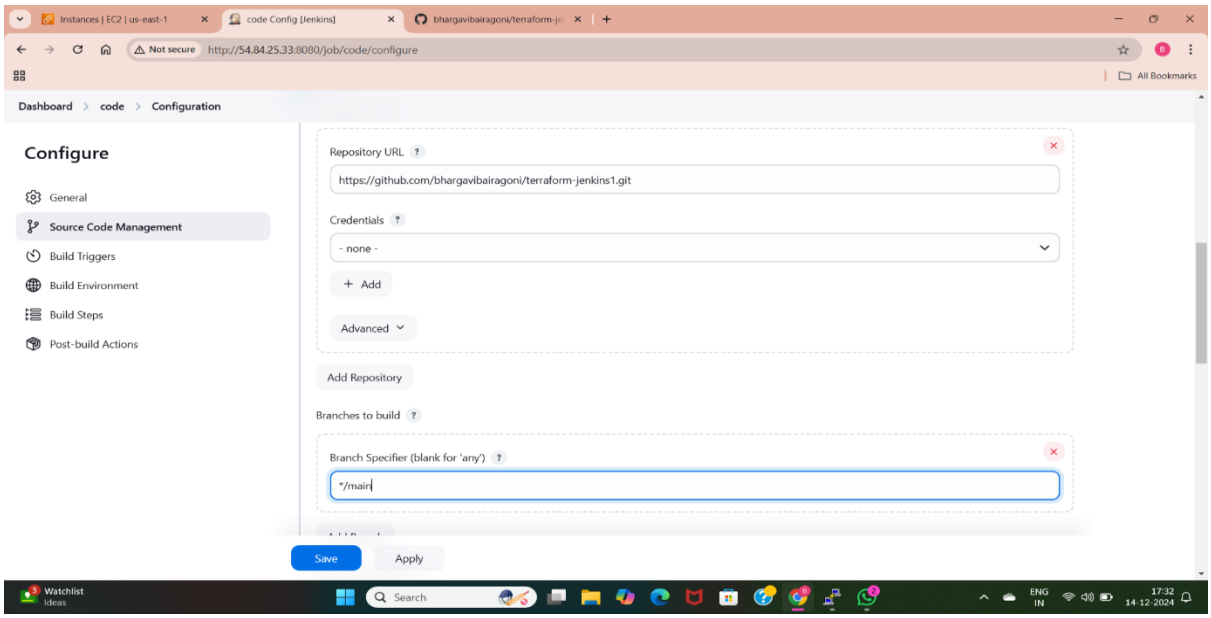


- Now I can able to access the wordpress application, provide the login details and access the wordpress application.

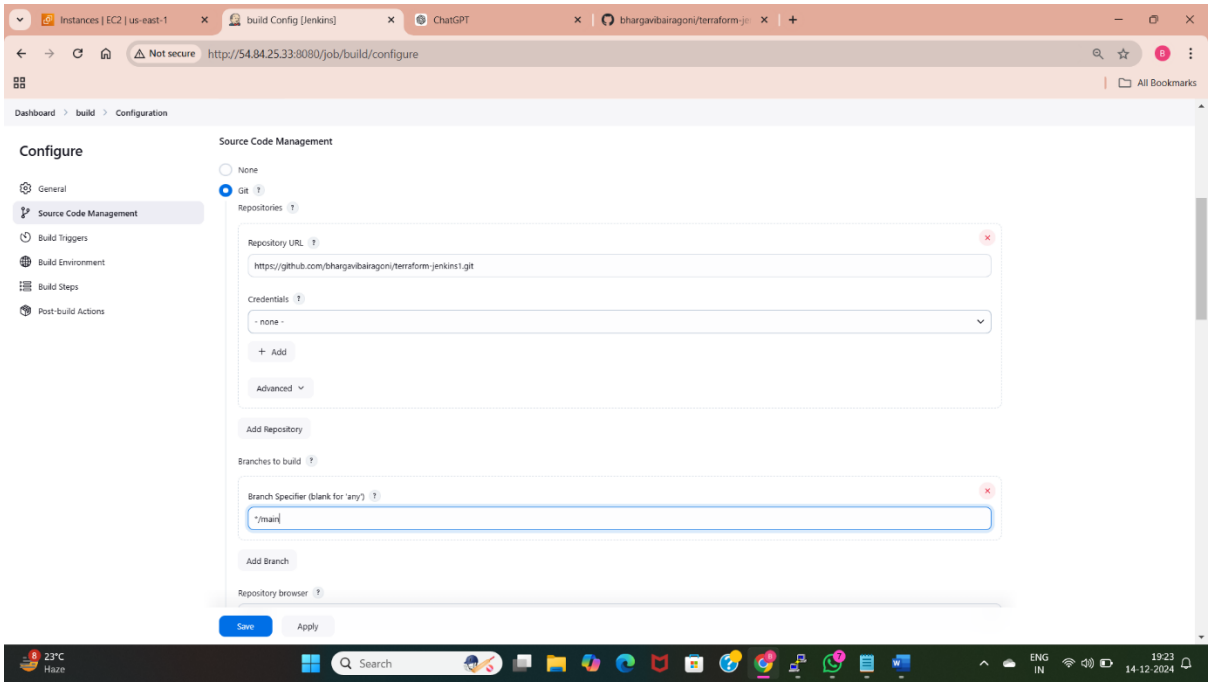
METHOD-9: DEPLOY WORDPRESS WEB APPLICATION BY USING JENKINS PIPELINE

Deploy WordPress web application by using git (clone terraform script which helps to deploy WordPress web application), jenkins (in execute shell install terraform, init, fmt, validate and apply with automatic command as terraform apply –auto-approve) and terraform and create 49enkins pipeline and add build periodically and poll scm to initial job of pipeline and check the changes happened or not which are made in github repo.

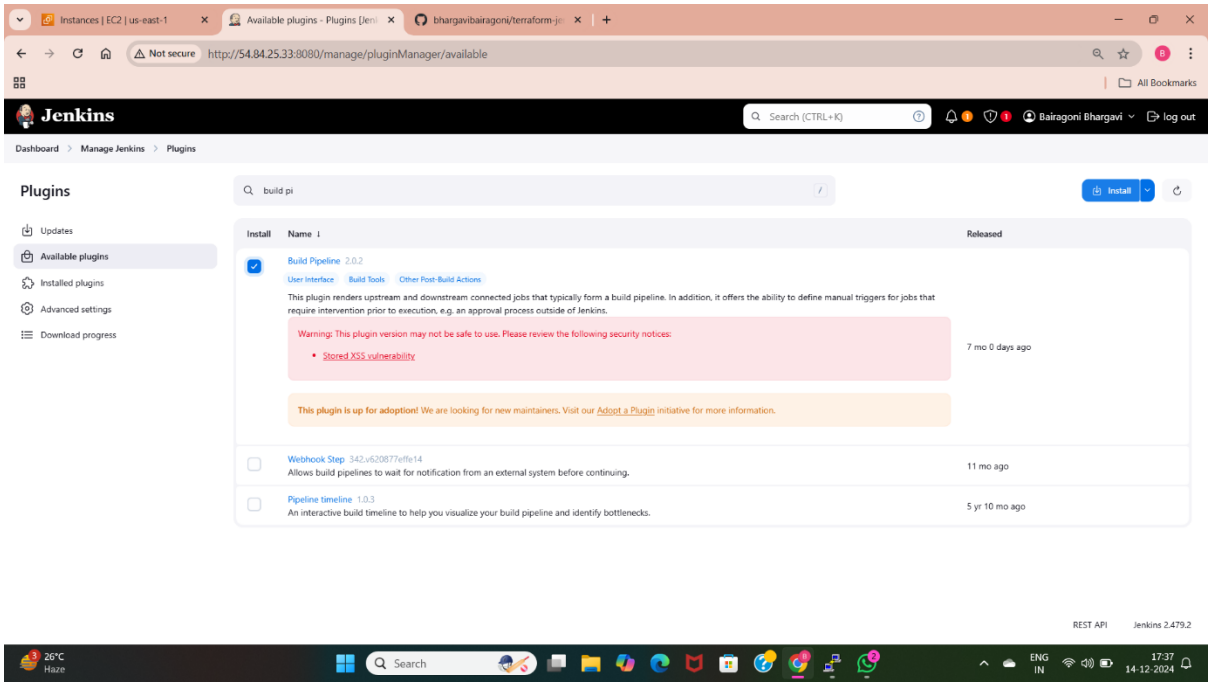
- To deploy wordpress application using pipeline first we need to create a job with freestyle project.



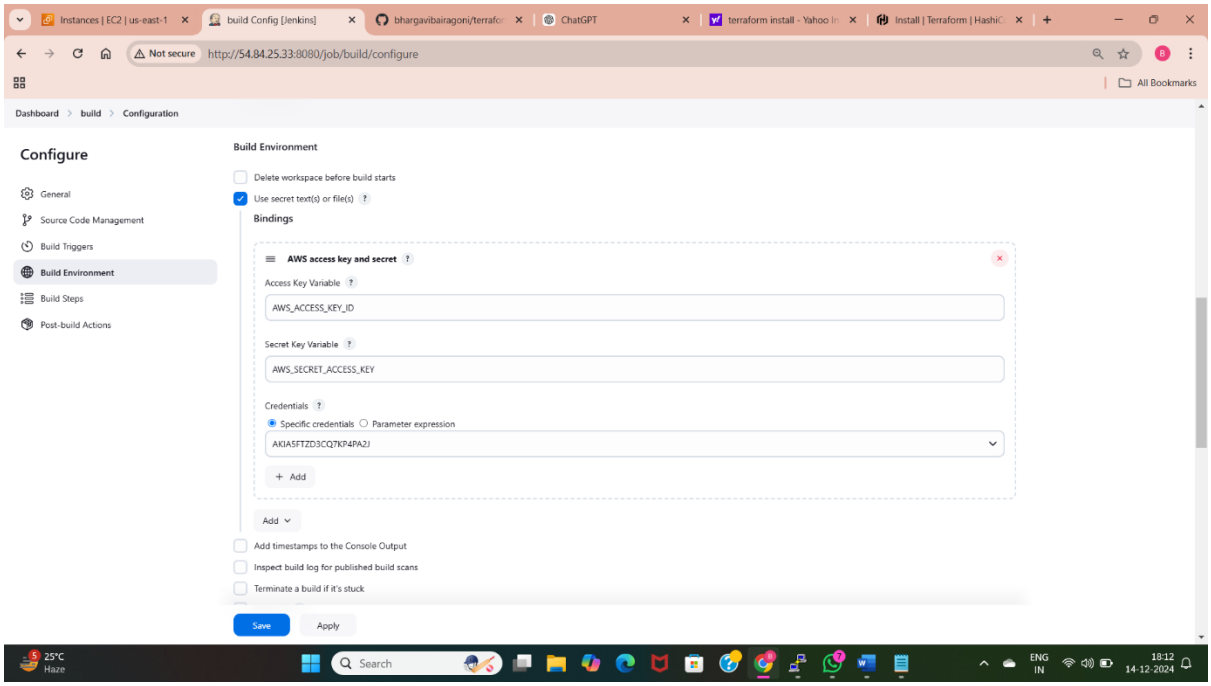
- I have created a free style job with the name “code”, I have taken the url from the github using this freestyle job.



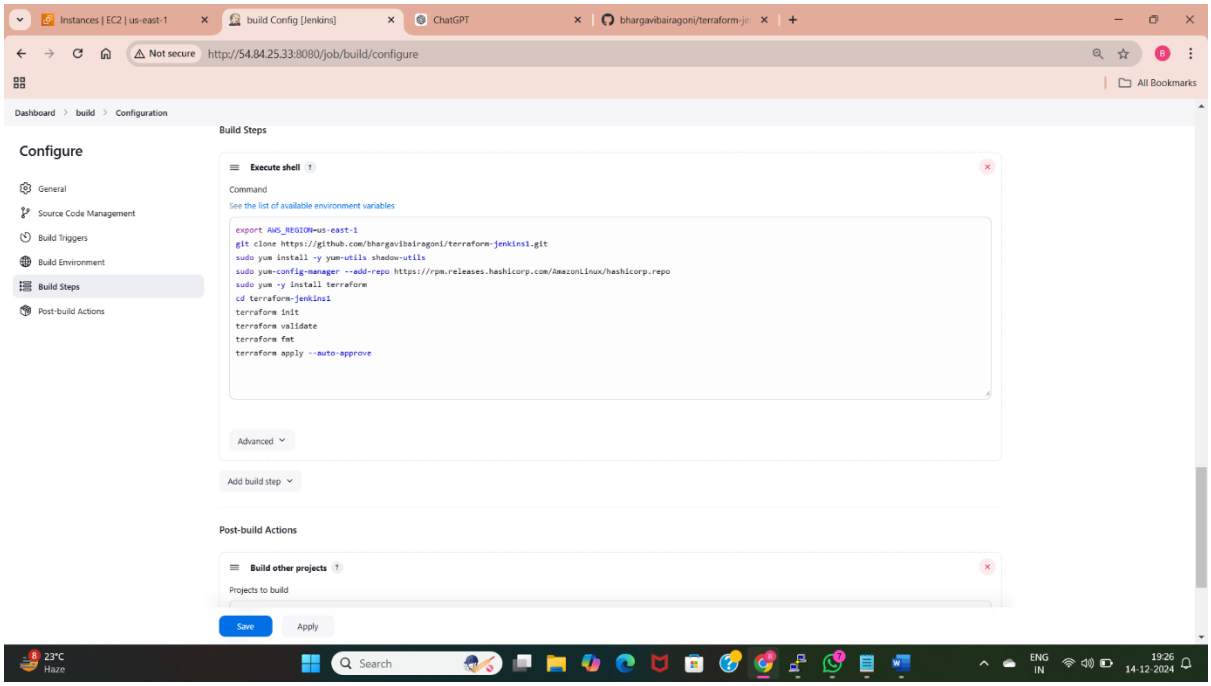
- I have created a freestyle job with the name “build”, added the code in this job using git url.



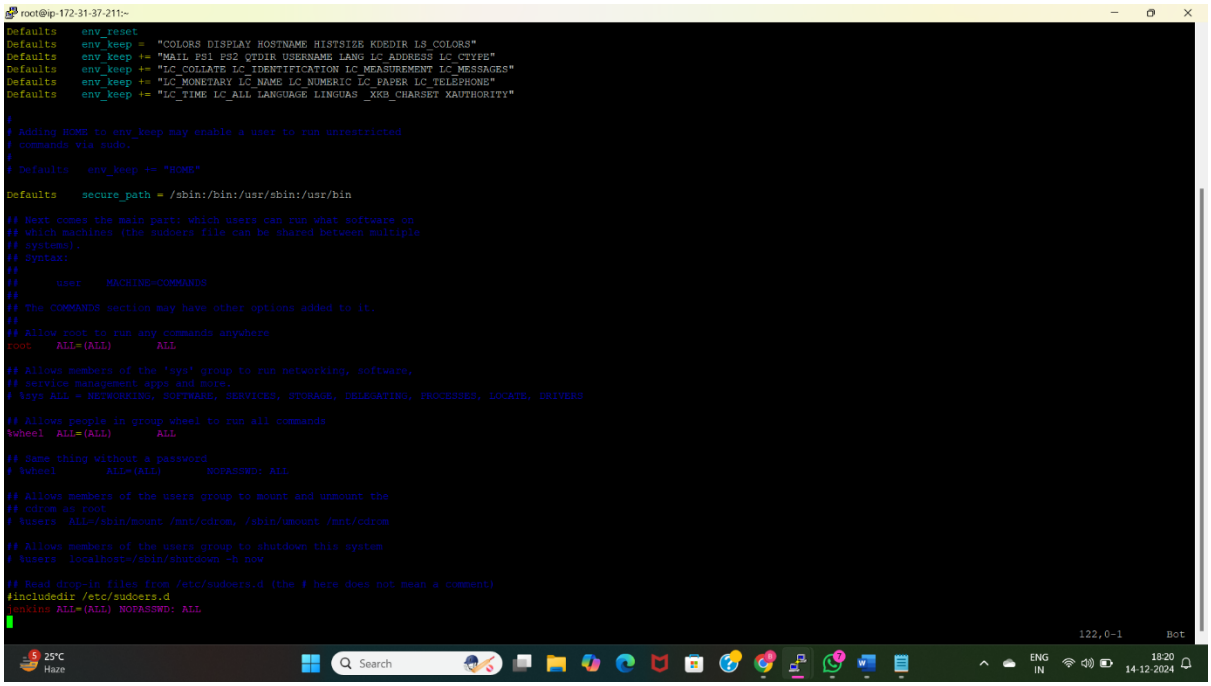
- I have used the “build pipeline” plugin to build the above jobs automatically by building only one job.



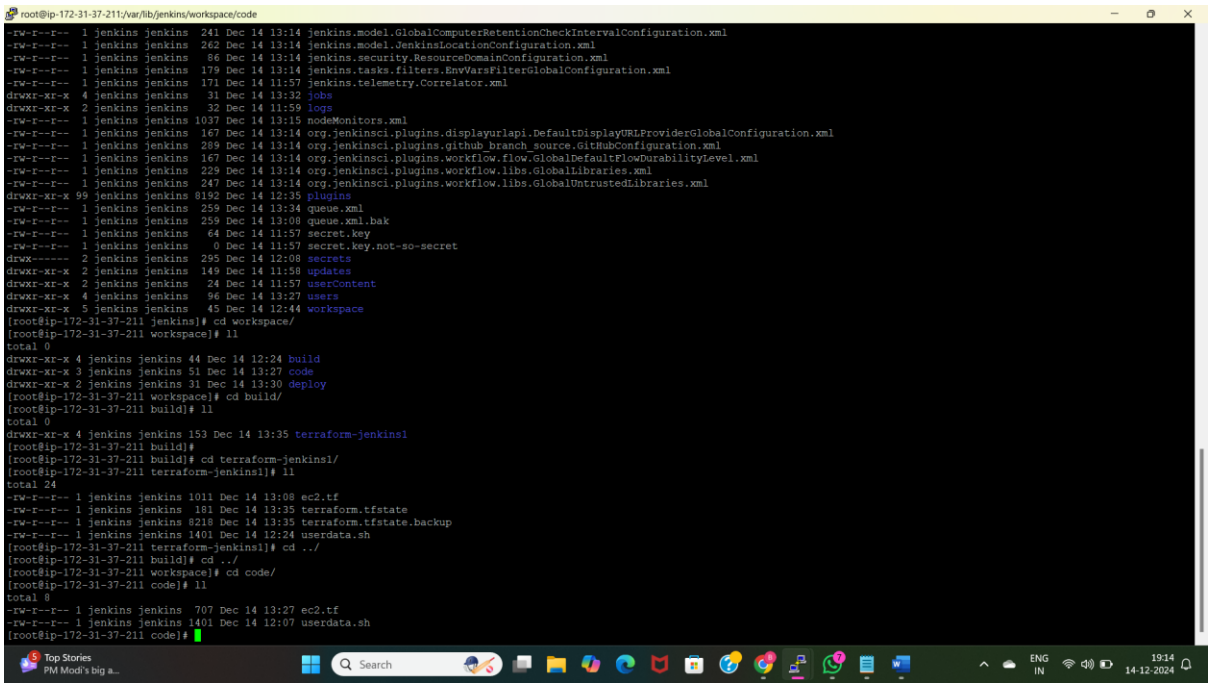
- I have assigned the access keys and secret access keys by using the credentials of aws credential plugin.
- First we need to go to manage Jenkins, move to credentials , select global credentials, select aws credentials plugin then give access keys and secret access keys.



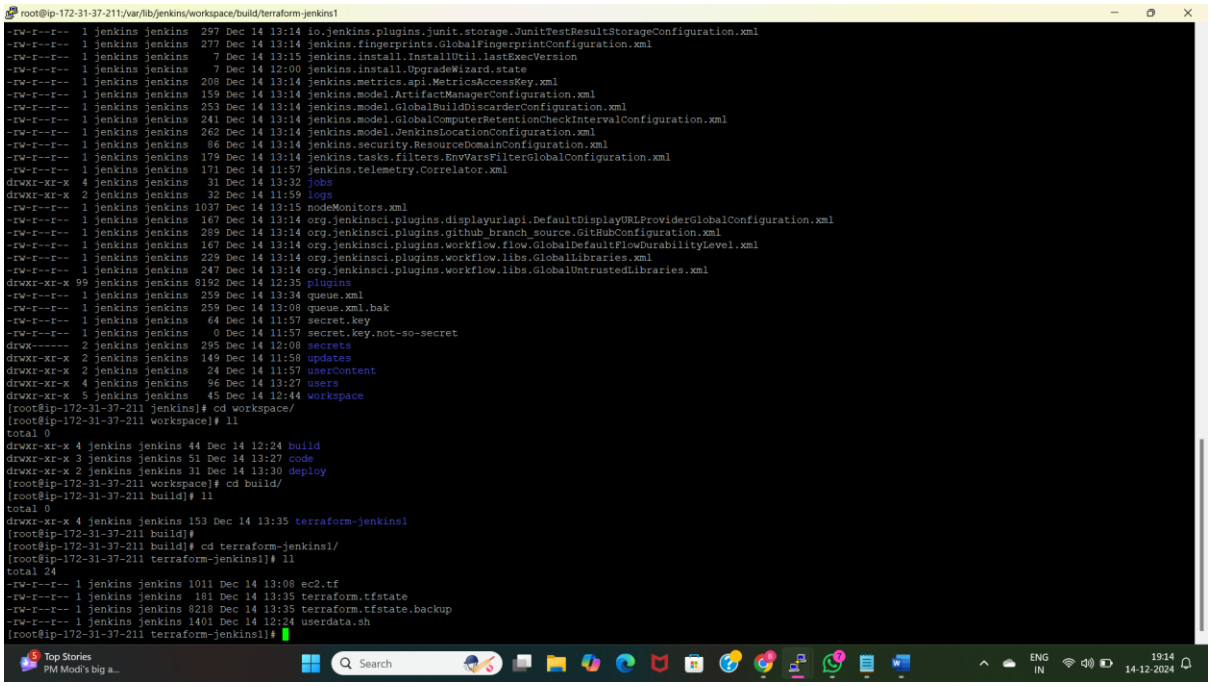
- I have used the execute shell in the build project.
- I have used the aws region us-east-1 for creating the resources in particular region.
- I have installed the terraform using the above commands.
- Moved to the directory to which this files(ec2.tf, userdata) are stored.
- I have initialized the terraform in the above files.
- I have checked the errors using the “terraform validate” command.
- I have applied the changes to launch the ec2 instance with userdata.



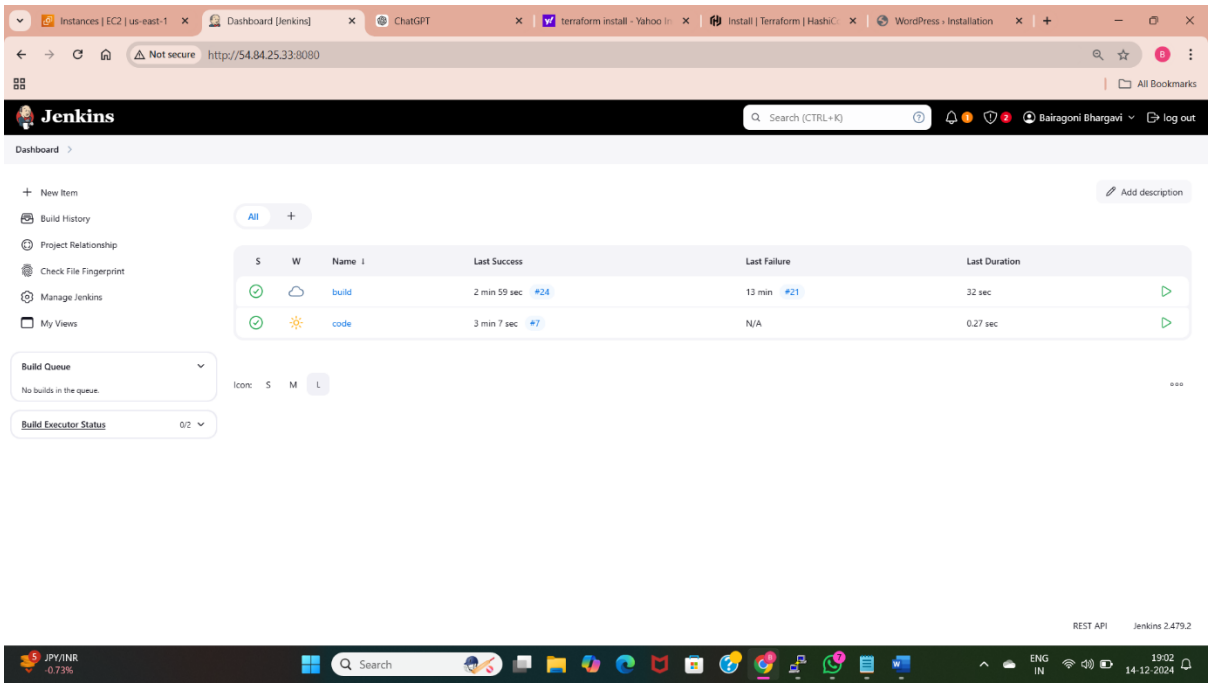
- To install terraform using sudo we need to give the sudo permissions in the Jenkins. First we need to move into the “visudo” and give permissions using “Jenkins ALL=(ALL) NOPASSWD: ALL”.



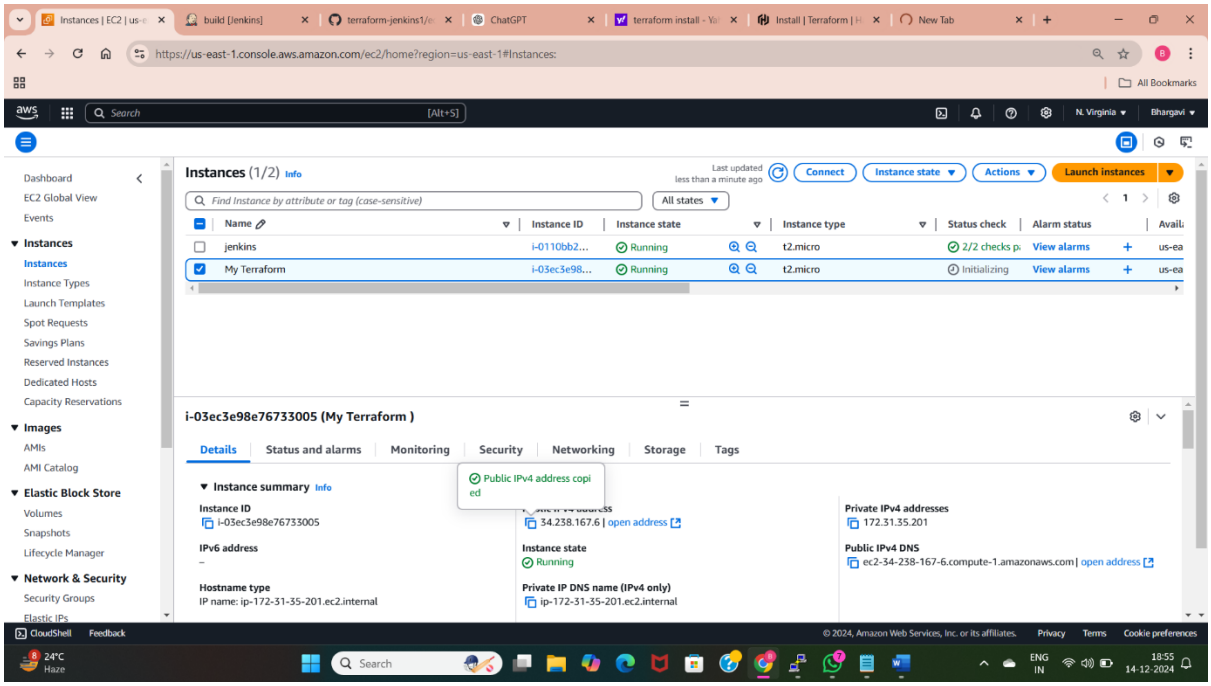
- I have checked with the terminal by moving into the Jenkins path “/var/lib/Jenkins”, there we can find details of job by using “workspace”, there we can see the created jobs and go inside every job and check whether the files are there or not.



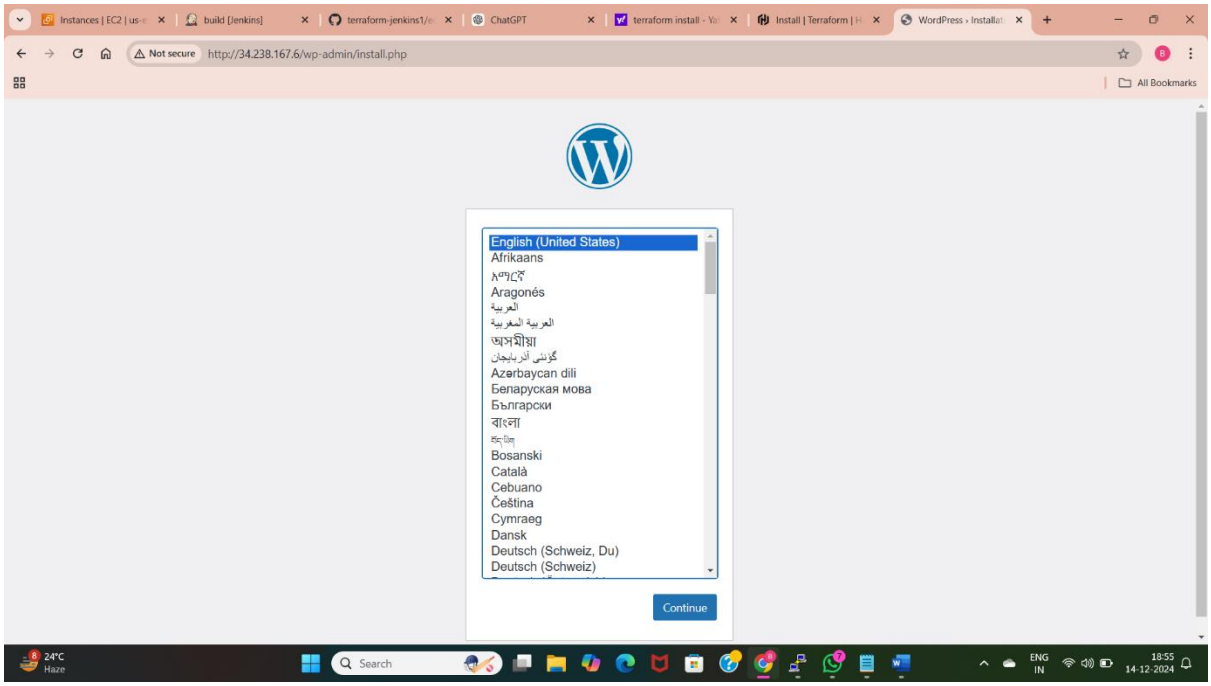
- Here I have moved into build job and there we can see “terraform-jenkins1” repository having the terraform files.



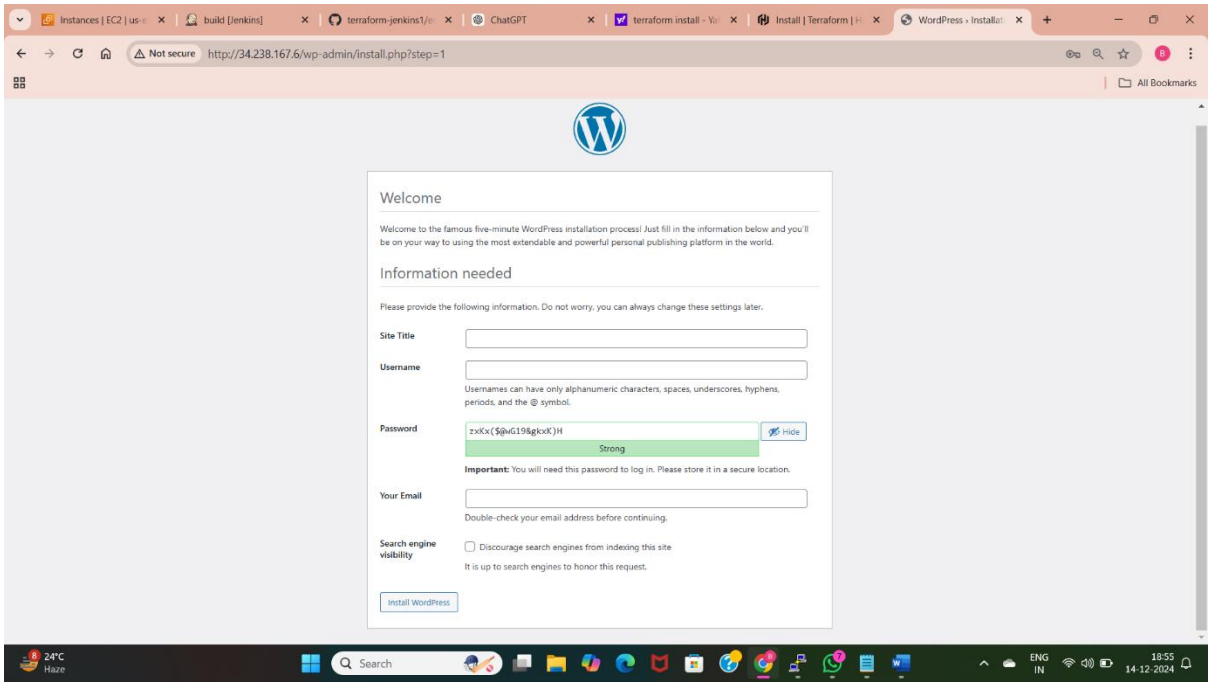
- After specifying the configurations we need to build the jobs. I have build the code first, I have used “build other projects” so the “build project” job is also build after completion of code.



- EC2 instance is created using the above configurations.



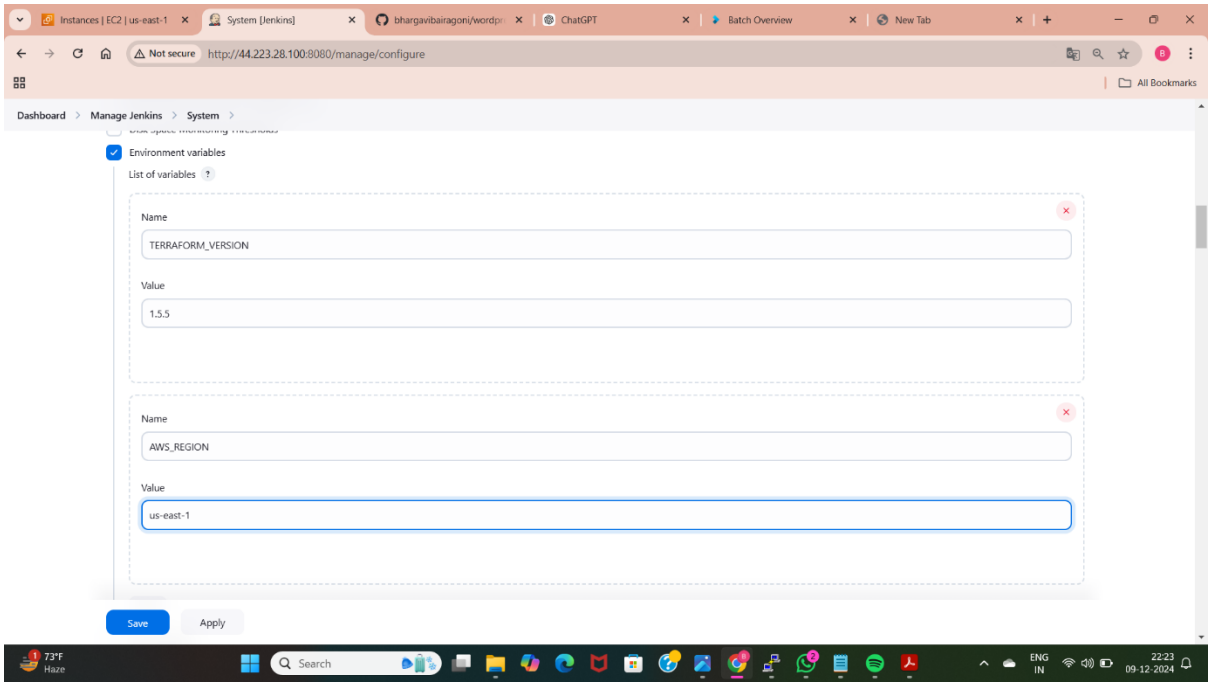
- Using the above created Public IP address we can access the deployed wordpress application.



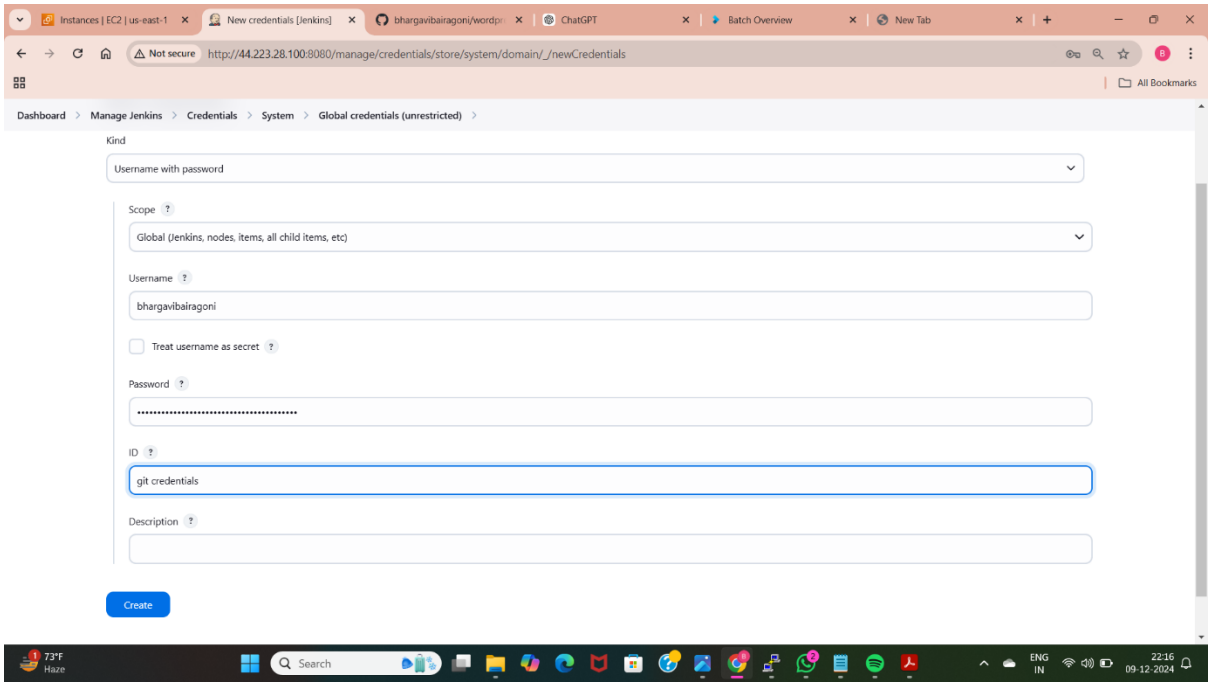
- I am able to access the wordpress application which is deployed using Jenkins freestyle pipeline.

We can also deploy this using the pipeline project.

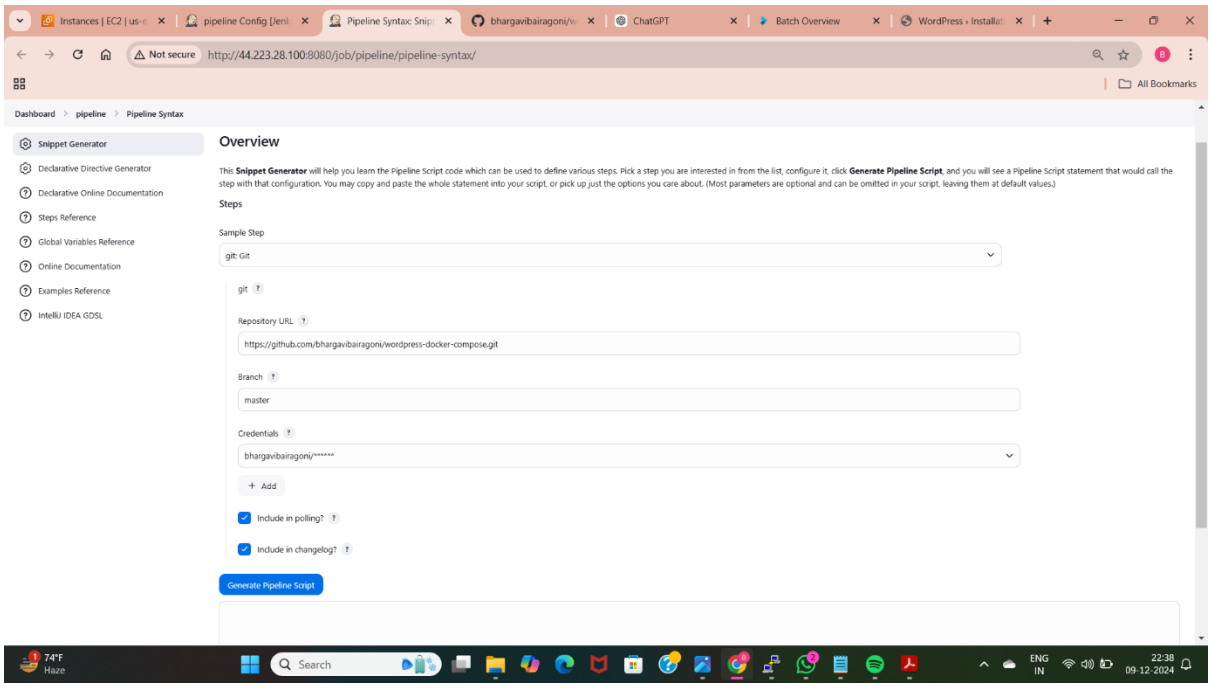
- I have created a job with the name of “pipeline” using pipeline project.



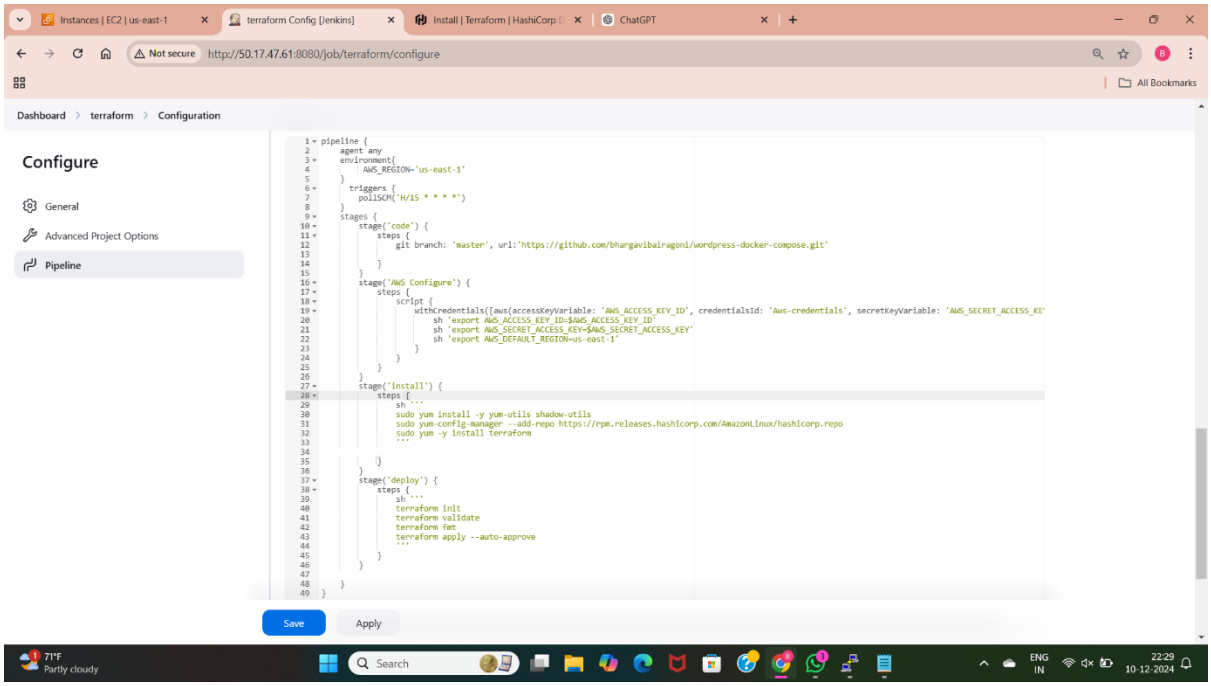
- We need to install terraform to deploy wordpress using terraform and Jenkins.
- I have given the virsion of the terraform as 1.5.5
- As we are using terraform code to deploy, I have created a code for launching EC2 instance with userdata. So we need to give Aws credentials to access Aws services using pipeline script.



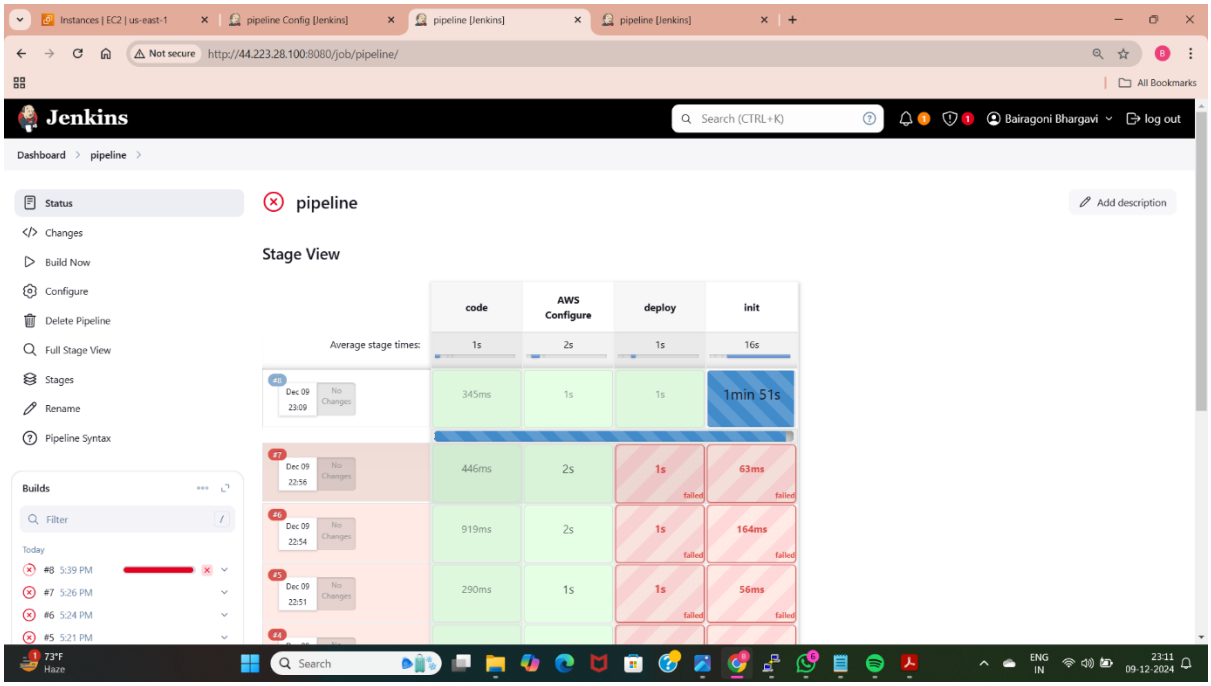
- I have provided the git credentials using username with password. I have given username as my github account username and password as personal access token, id as git-credentials.



- I have generated script for code that is taking code from github using url, here I have provided above created credentials.



- This is the pipeline script used to deploy wordpress application.
- First we need to take code from github using url and credentials if it is private repository.
- Next we need to provide aws credentials i.e access keys and secret access keys using aws credentials plugin.
- I have installed terraform using the stage “deploy”. To work with terraform we need to install terraform in our Jenkins server and newly launching instance.
- Next I have provided the terraform commands that are used to deploy wordpress application i.e first we need to initialize terraform in our created files, we need to validate the code whether the code is working or not, we need to format the code so that we will get indentation for our code. We need to apply the changes to create the resources.

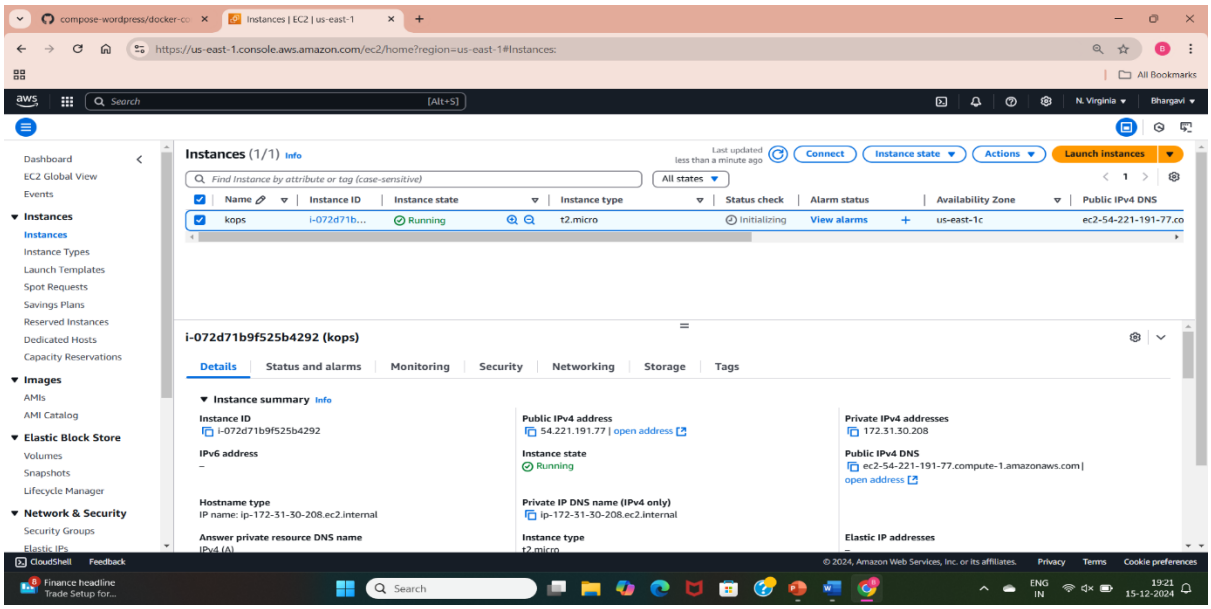


- After writing the pipeline script we need to build the pipeline so that we can access our wordpress application. Here I have used four stages in pipeline, three stages are successful and fourth stage is running.

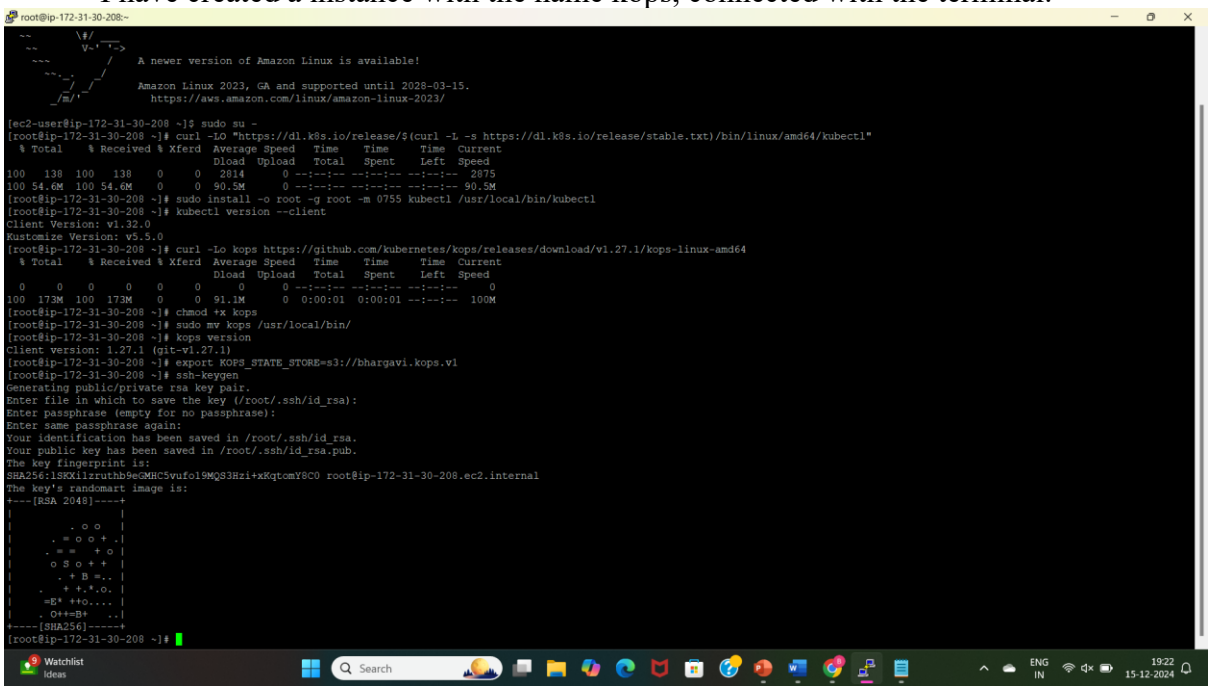
METHOD-10: DEPLOY WORDPRESS WEB APPLICATION BY USING K8'S

Kubernetes is an open-source container orchestration platform. It is used to automate the deployment, scaling, and management of containerized applications. It is also known as k8s. K8s simplifies the process of running applications in containers by providing tools and abstractions for managing infrastructure and application workloads efficiently.

✓ To setup the Kubernetes first we need to create a instance.

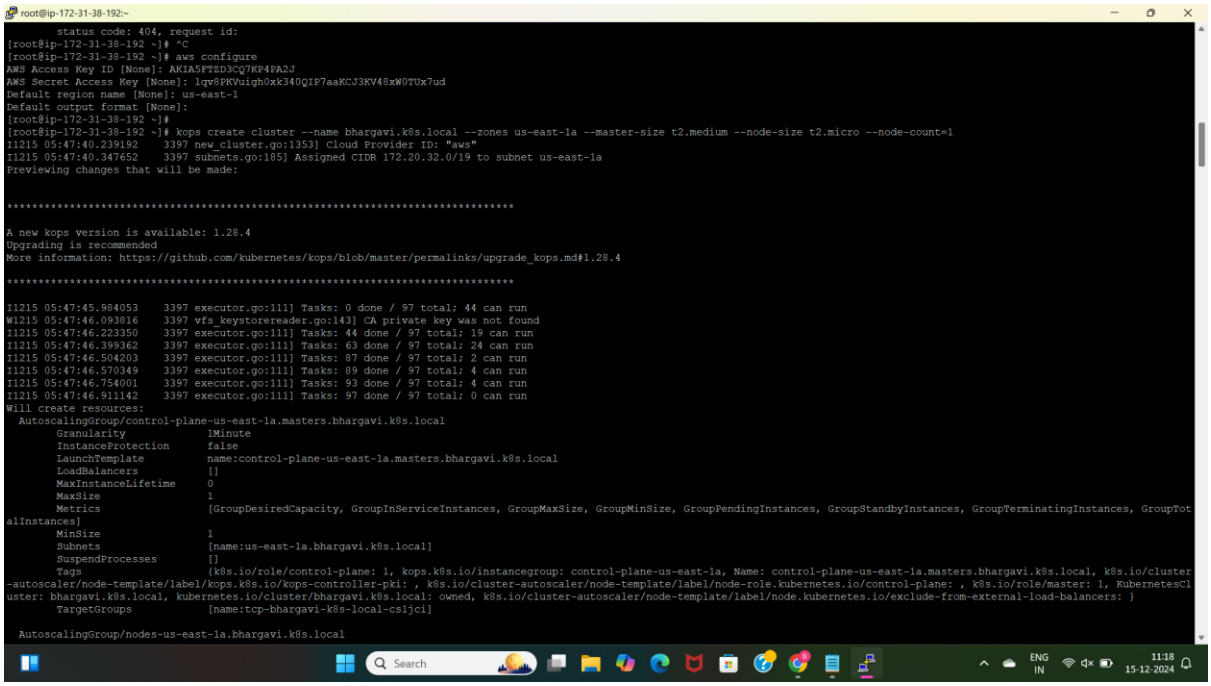


✓ I have created a instance with the name kops, connected with the terminal.

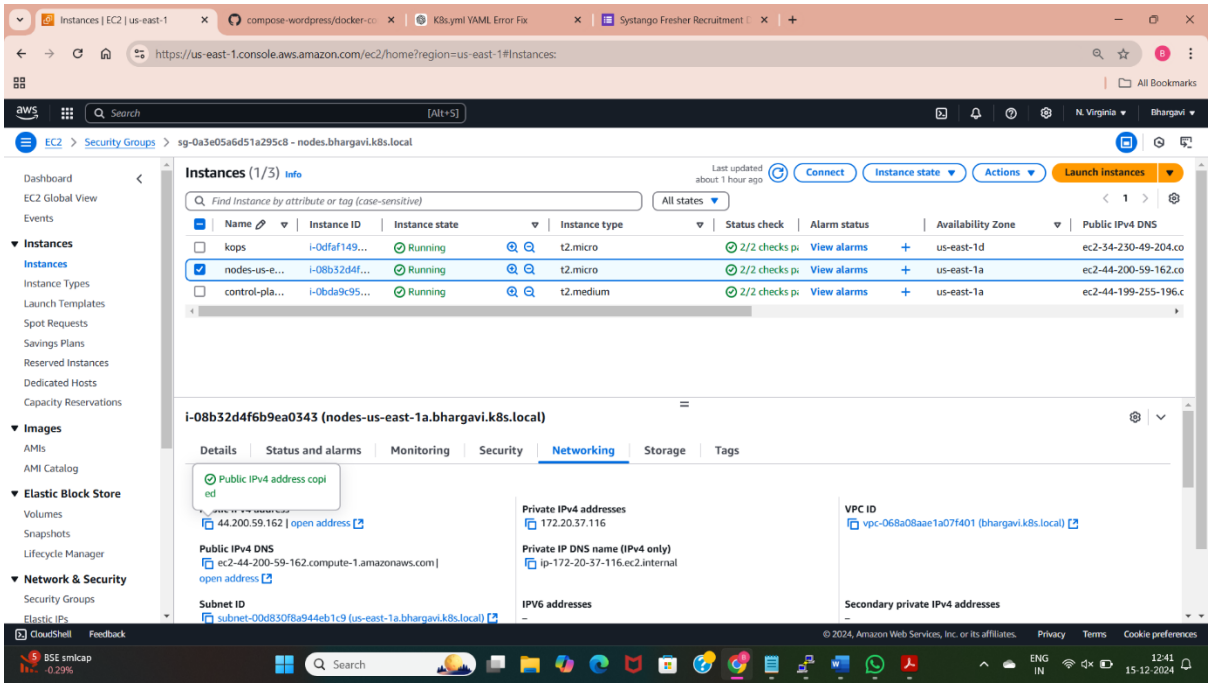


✓ Using the created instance I have installed kubectl(to interact with k8s cluster), kops(to create, destroy, and manage k8s clusters).

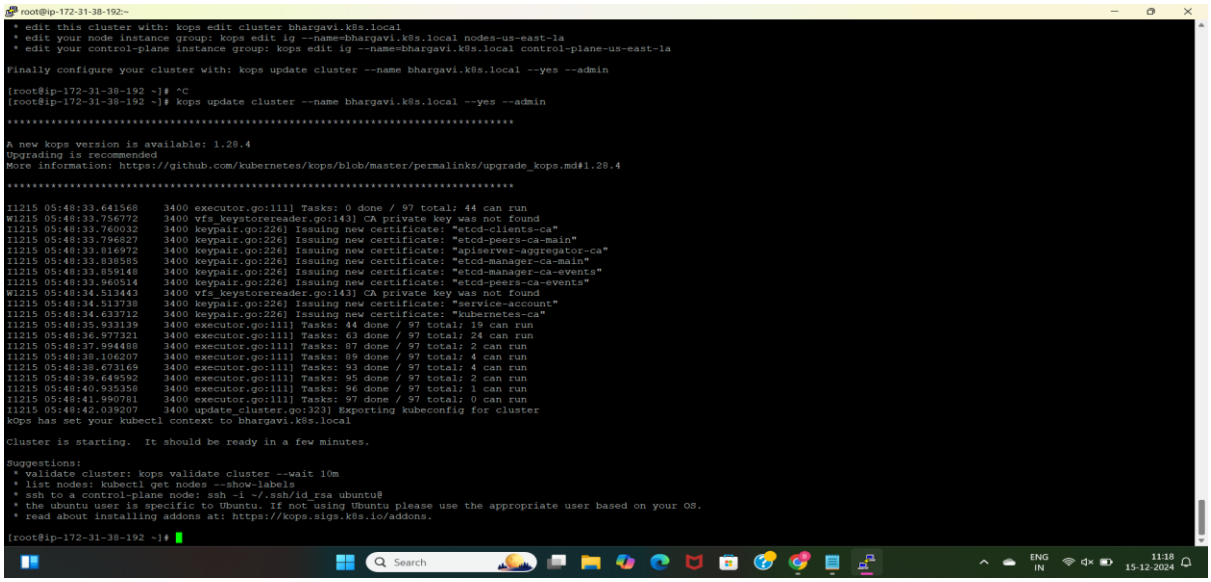
✓ We need to create a s3 bucket to store the cluster state and export into the Kubernetes so it stores the state of cluster.



- ✓ I have given the configurations to the aws by using aws configure command.
- ✓ I am creating the clusters using the kops.
- ✓ Created a master with t2.medium size and us-east-1 zone, node with t2.micro size and count is 1 means creating 1 node using the command “kops create cluster bhargavi.k8s.local –zones us-east-1a –master-size t2.medium –node-size t2.micro –node-count=1”.



- ✓ With the help of above I have created two clusters one is master with the name “control-plane” other one is node with the name“node-us-east-1”.



- ✓ We need to update the created cluster with “kops update cluster –name bhargavi.k8s.local –yes –admin”.

ec2-user@slave1:~/workspace/slave1

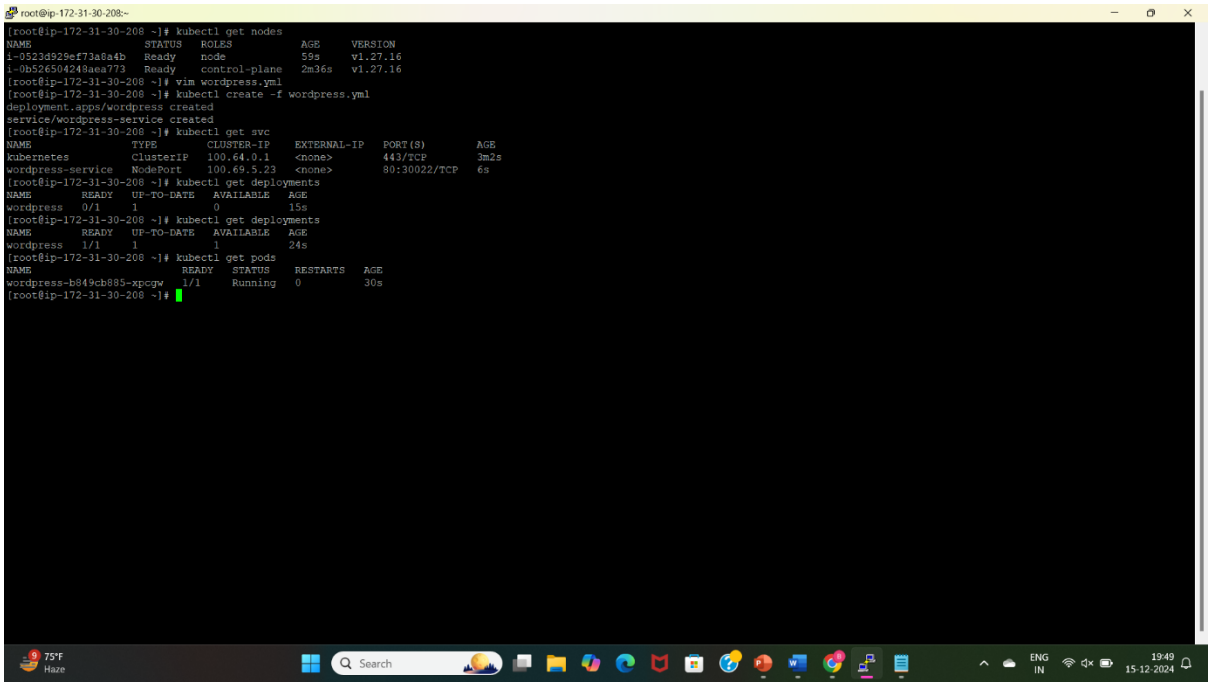
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: word-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: wordpress
  template:
    metadata:
      labels:
        tier: wordpress
    spec:
      containers:
      - name: wordpress-container
        image: bhargavibairagoni/wordpress:wordimage
        ports:
        - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: word-service
spec:
  selector:
    tier: wordpress
  ports:
  - protocol: TCP
    port: 80          # External port
    targetPort: 80    # Port inside container
  type: LoadBalancer
```

- ✓ This is the deployment and service file for wordpress. It is running on the port number 80 with two replicas.

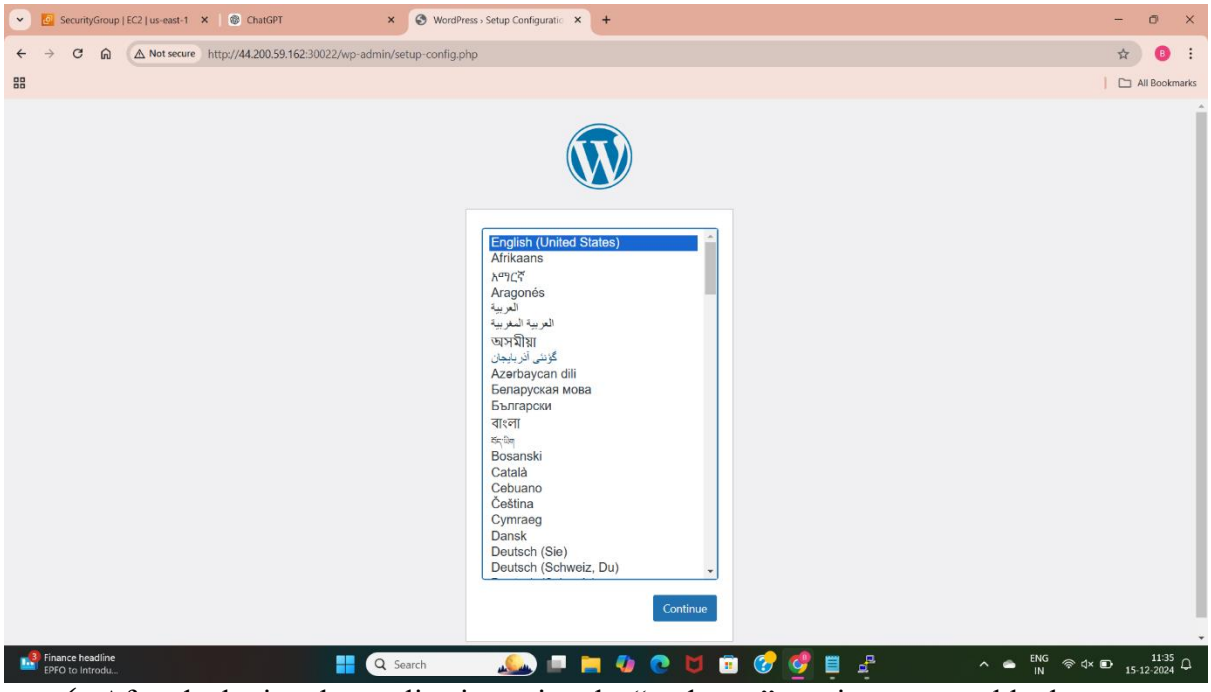
ec2-user@slave1:~/workspace/slave1

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: mysql
  template:
    metadata:
      labels:
        tier: mysql
    spec:
      containers:
      - name: mysql
        image: bhargavibairagoni/mysql:mysqlimage
        ports:
        - containerPort: 3306
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "wordpress"
        - name: MYSQL_DATABASE
          value: "databaseword"
        - name: MYSQL_USER
          value: "admin"
        - name: MYSQL_PASSWORD
          value: "ADMIN123"

---
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  selector:
    tier: mysql
  ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
  type: LoadBalancer
```

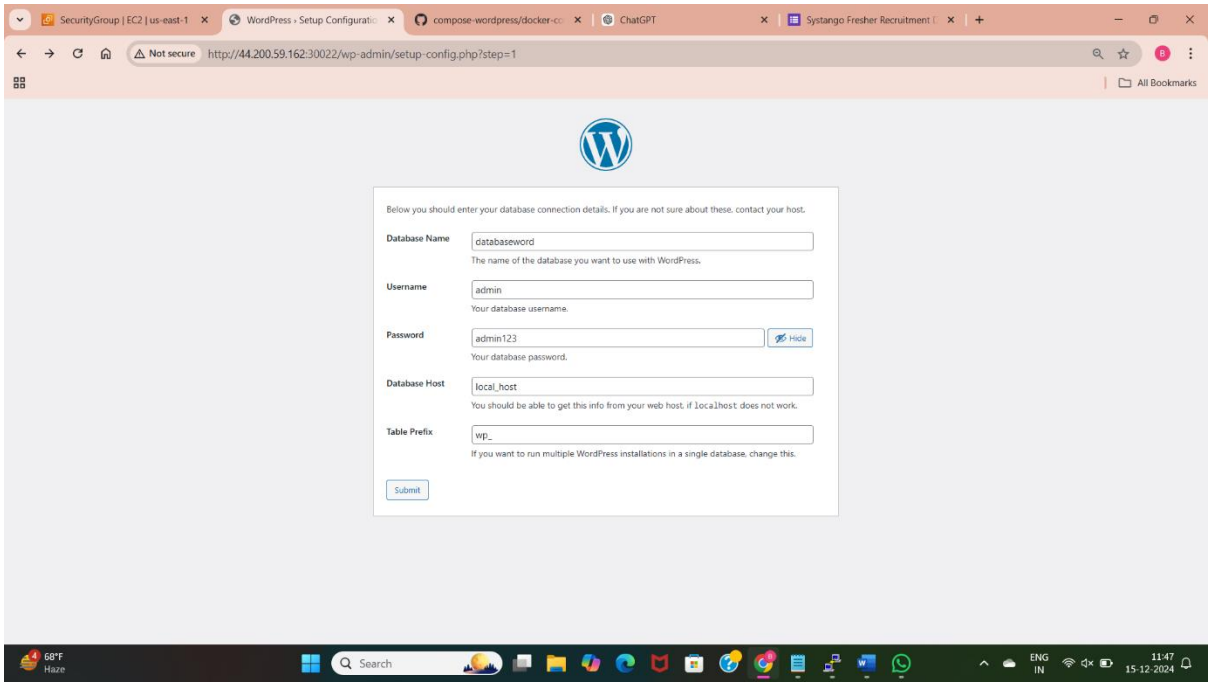
- ✓ I have created a manifest file with the deployment and service.
- ✓ The deployment file is used to create a pod with the containers.
- ✓ I have created a deployment with the name wordpress, created a container with the name wordpress and taken the image from the dockerhub "bhargavibairagoni/wordpress:wordpimage".
- ✓ As am taking the replica of the pod I haved created the template of the pod so that if new pod is created the above mentioned configurations are created inside the pod and the container.
- ✓ We need to give the port number to the container.



- ✓ The container port is used to take the traffic and the port no. 80 is listening the traffic from the users.
- ✓ I have created a service file with the name “wordpress”, I have used the nodeport service type with the target port 80, and node port 30022.
- ✓ The target port is used to route the traffic to the clusters and nodeport is used to expose the applications to external access..
- ✓ Port is used inside the Kubernetes cluster.



- ✓ After deploying the application using the “nodeport” service we can able the access the application with the port number 30022 and the ip address of the created clusters.



- ✓ Here am getting the wordpress application using the public ip address of the cluster and the port number we used in the manifest file.