DevOps Project:

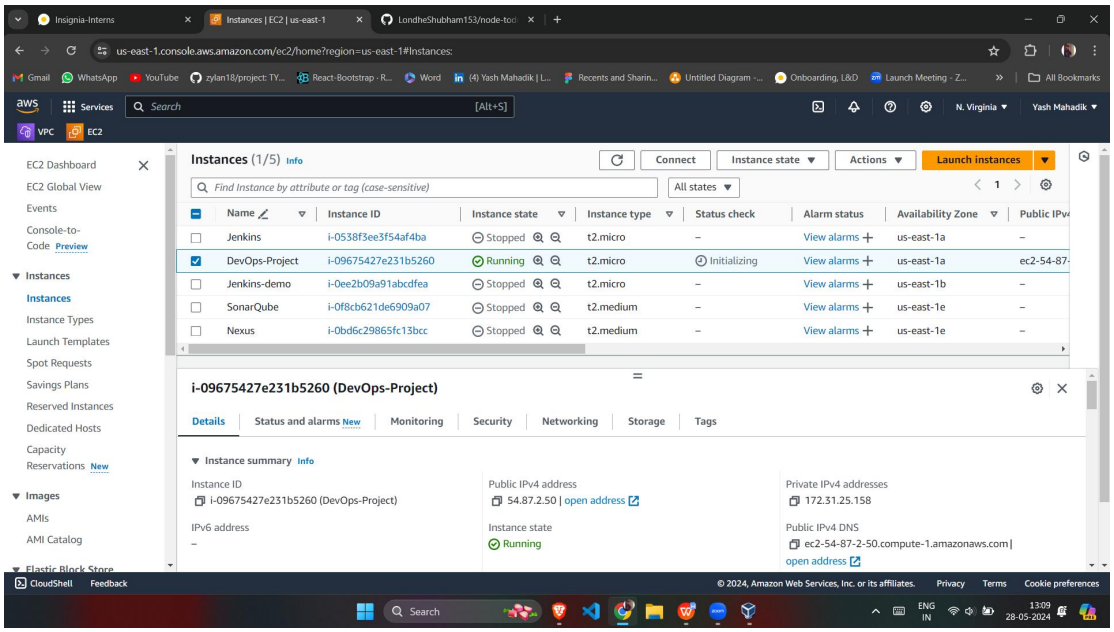
Deploying a Node.js application with GitHub integration using Jenkins

Introduction:

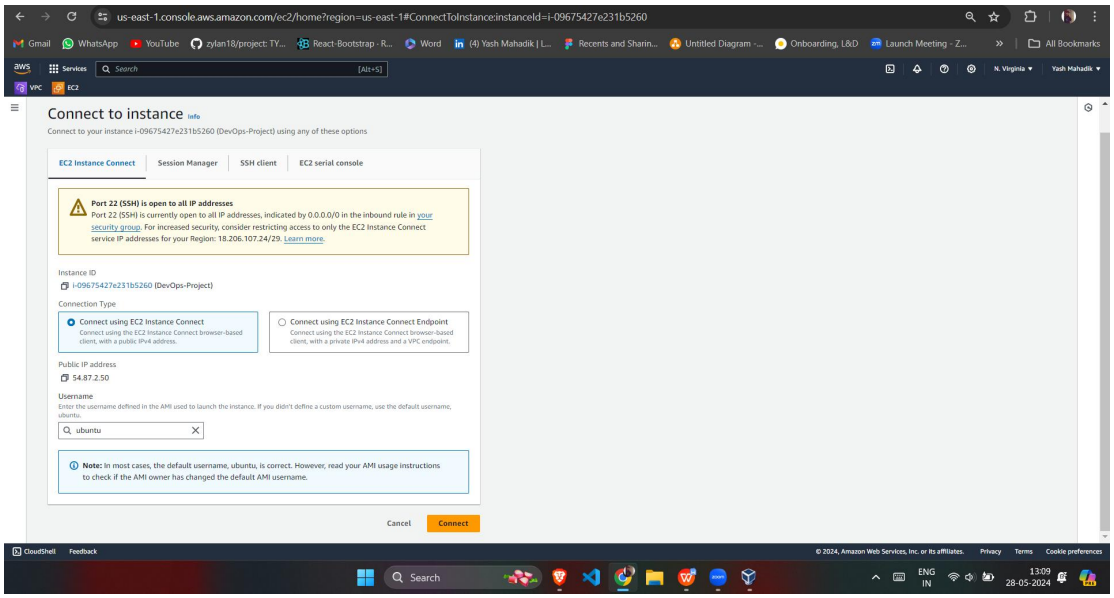
Deploying a Node.js application with GitHub integration using Jenkins involves automating the entire build and deployment pipeline to ensure a seamless and efficient workflow. This process is crucial for modern development practices, particularly in continuous integration and continuous deployment (CI/CD) environments.

A step-by-step guide

1) Login to you AWS account, Create an EC2 instance name it as DevOps -Project select Ubuntu and add key pair and launch the instance.

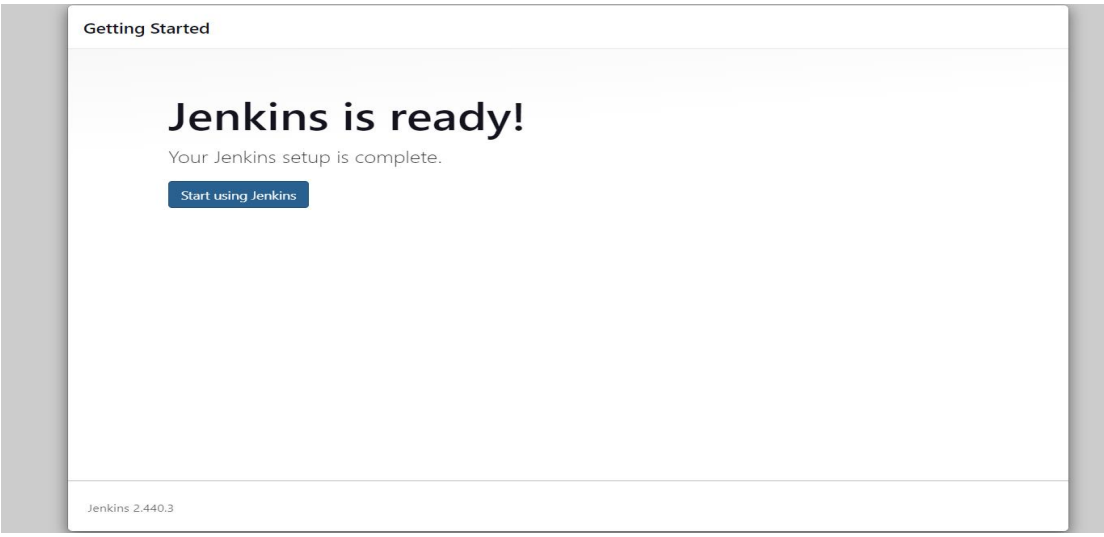


2) Connect your instance

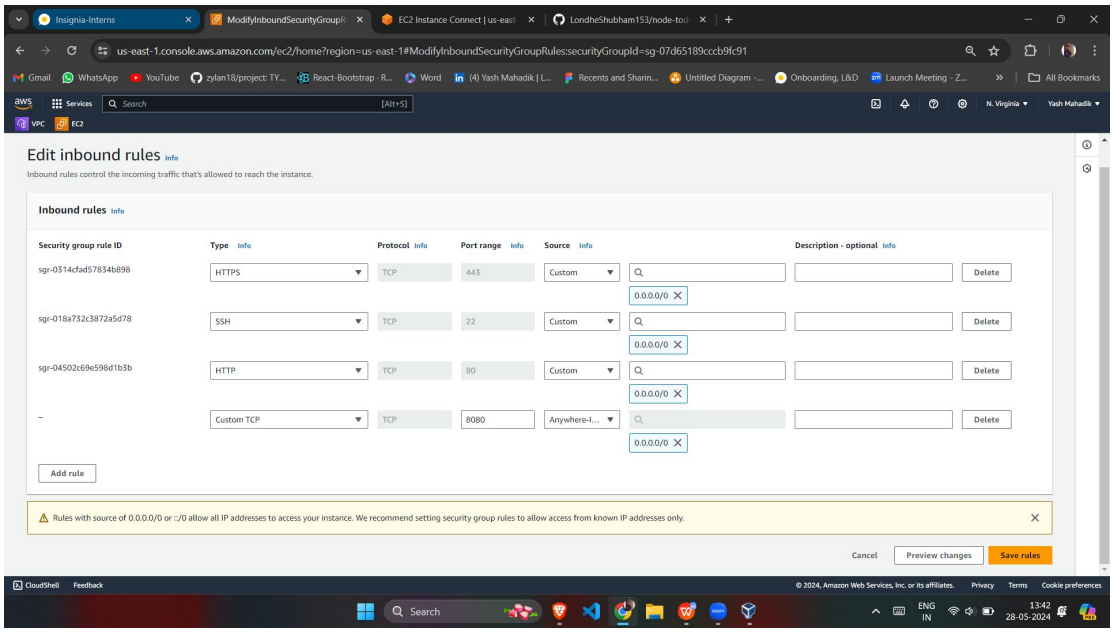


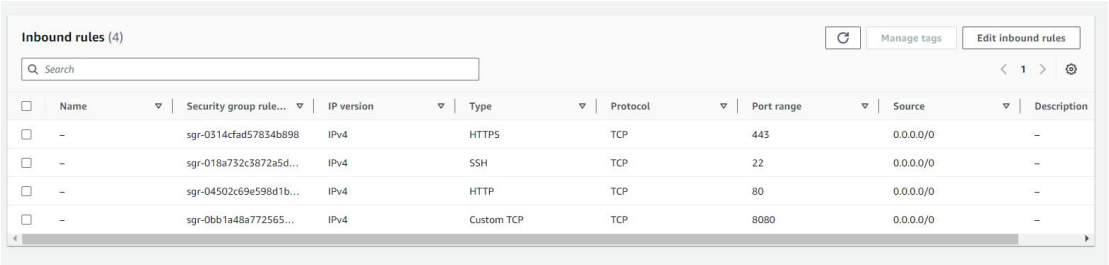
3) Need to install Jenkins using following commands

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

    sudo apt-get update
    sudo apt-get install fontconfig openjdk-17-jre
    sudo apt-get install jenkins
```

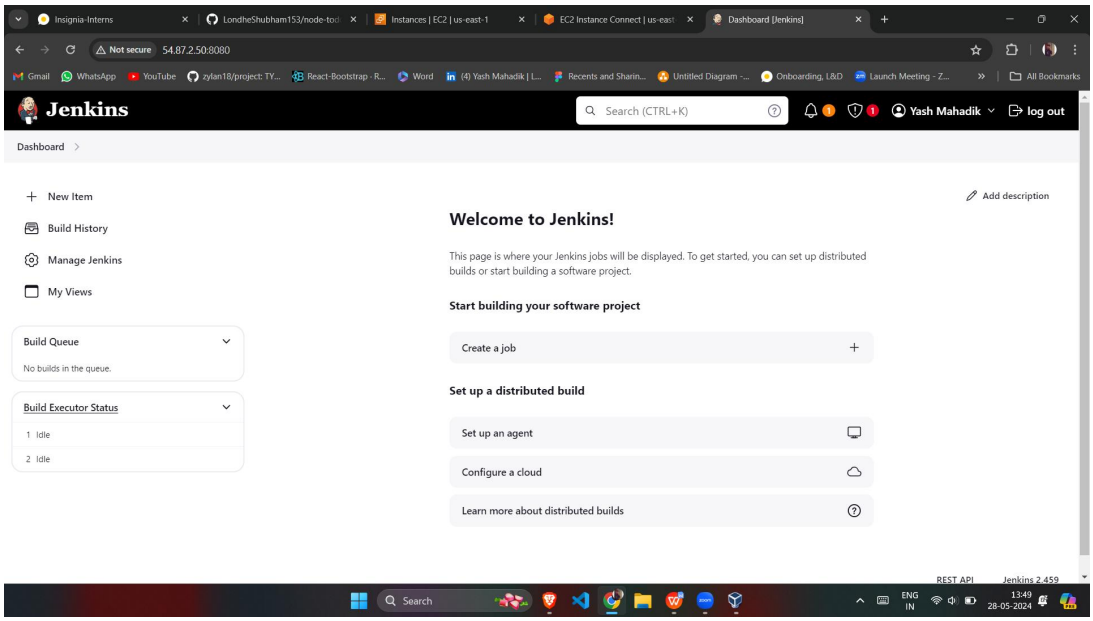


4) After installing jenkins run the jenkins using **sudo systemctl start jenkins** and to check the status use **sudo systemctl status jenkins.** As we know Jenkins run on port 8080, we need to configure security group and edit inbound rules.



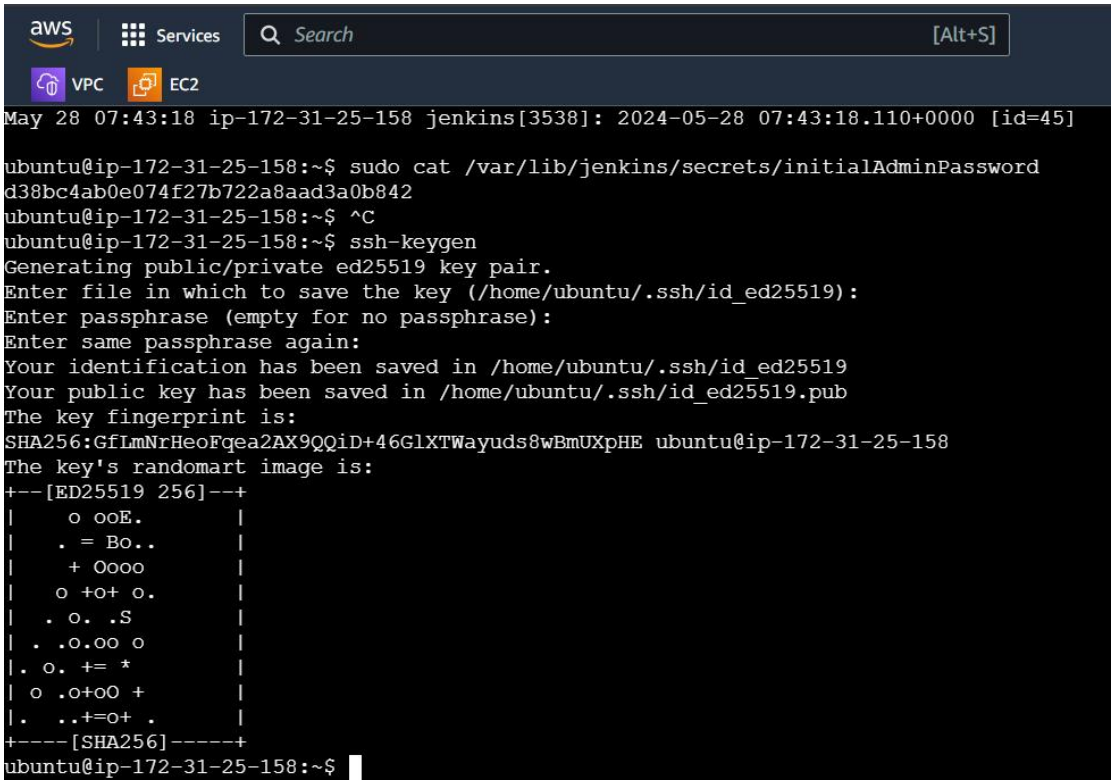


5) Jenkins is ready to use and its running on port 8080.

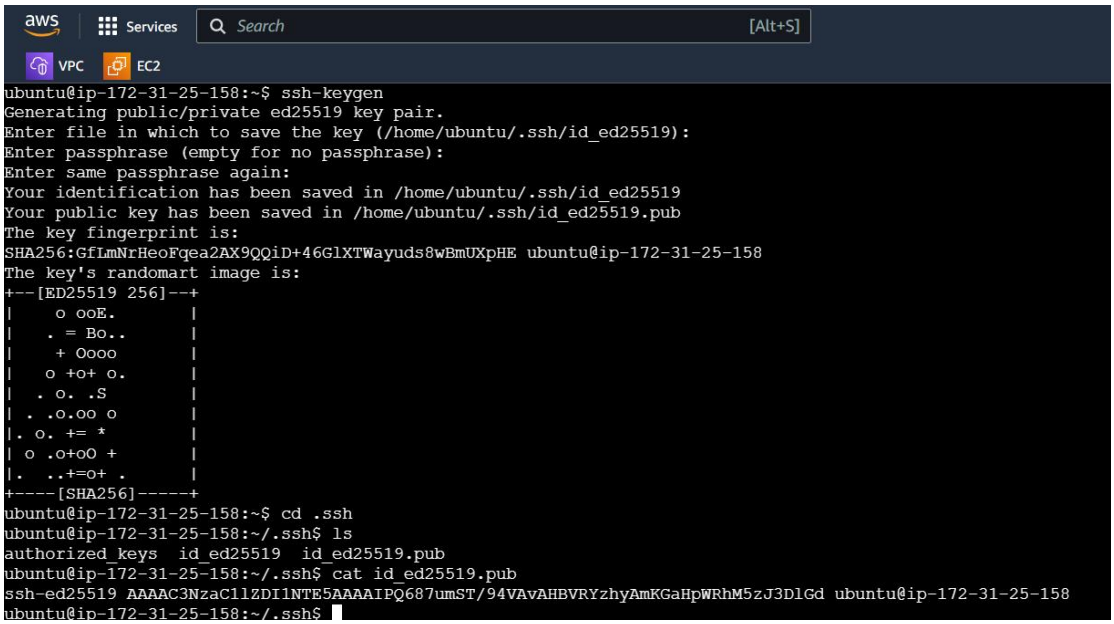


6) Now to Generate Access token.

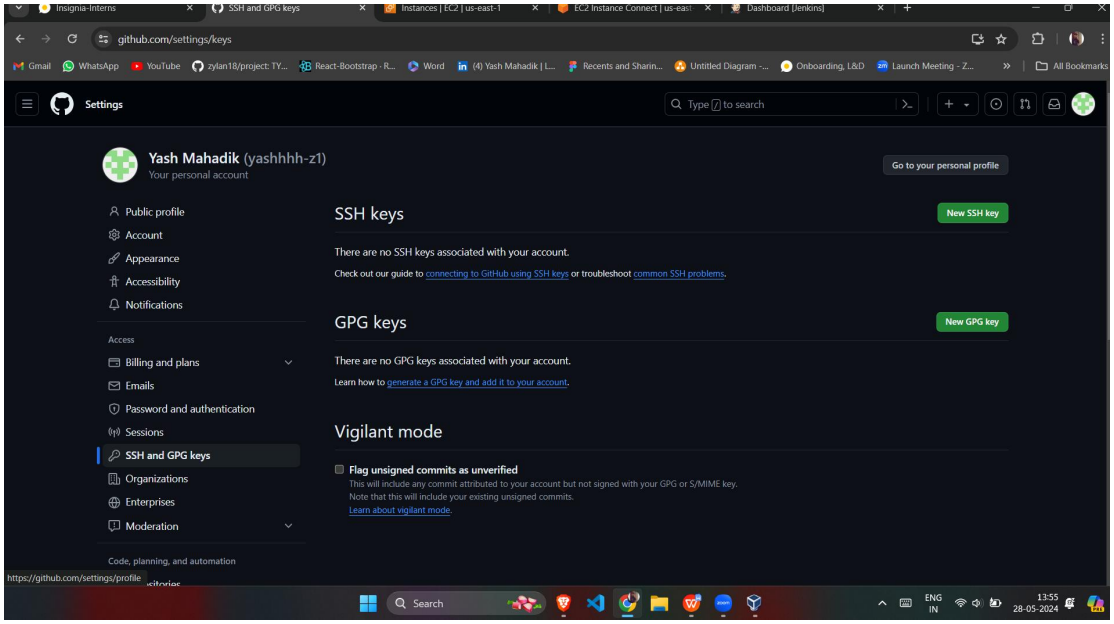
ssh-keygen



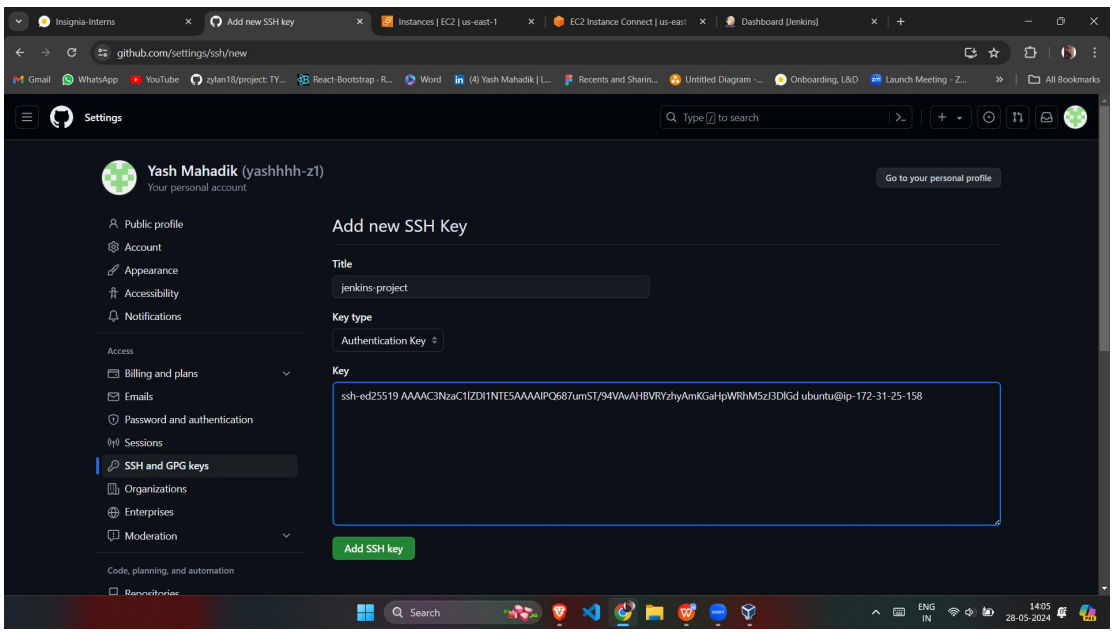
cd .ssh
ls
cat id_ed25519.pub // this our public key

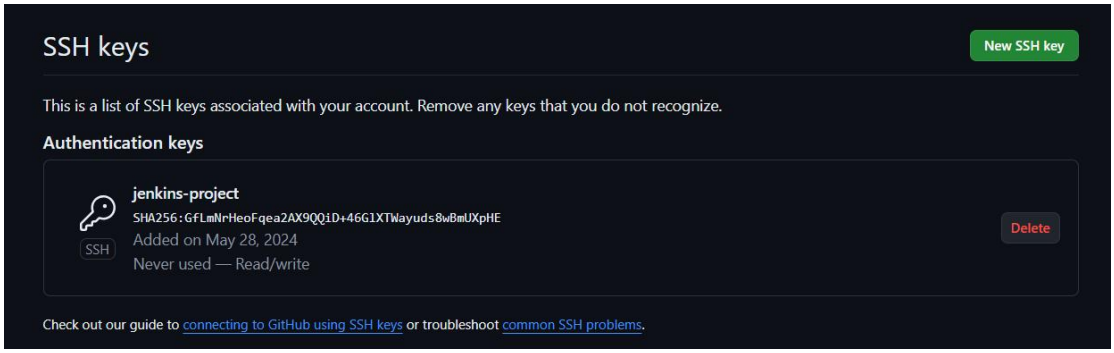


7) Setting ssh key Generate in github , Setting > ssh and gpg keys. Click on new SSH key. https://github.com/yashhhh-z1/node-todo-cicd.git

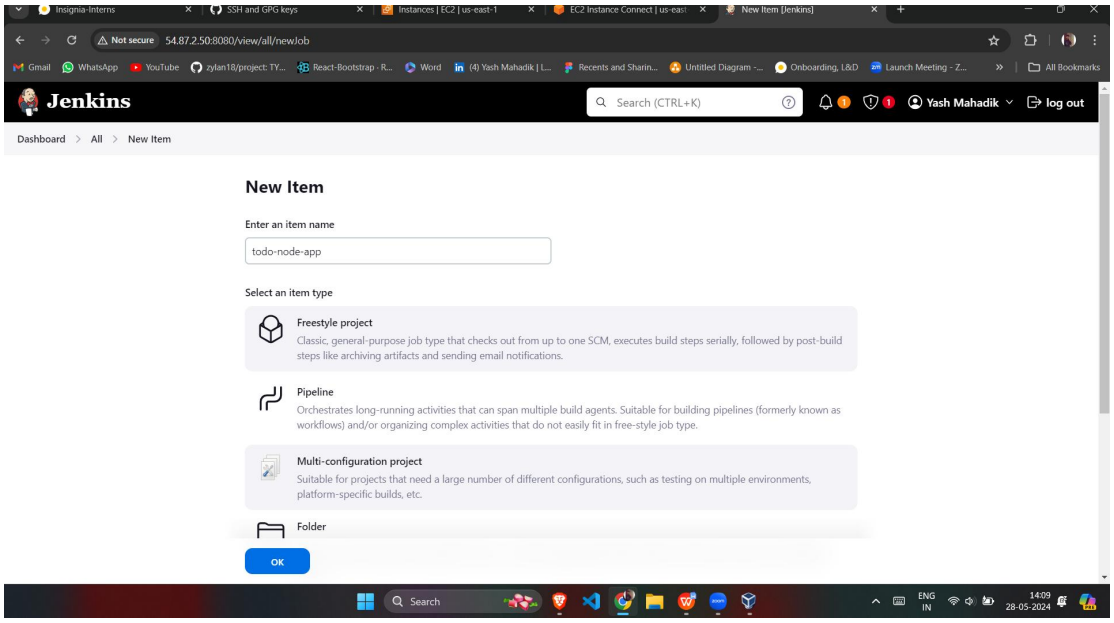


8) Give name to the key as jenkins-project, and inside authenticate key just copy paste your Public key and click on add ssh key. We have generated our access key for connecting jenkins and github.

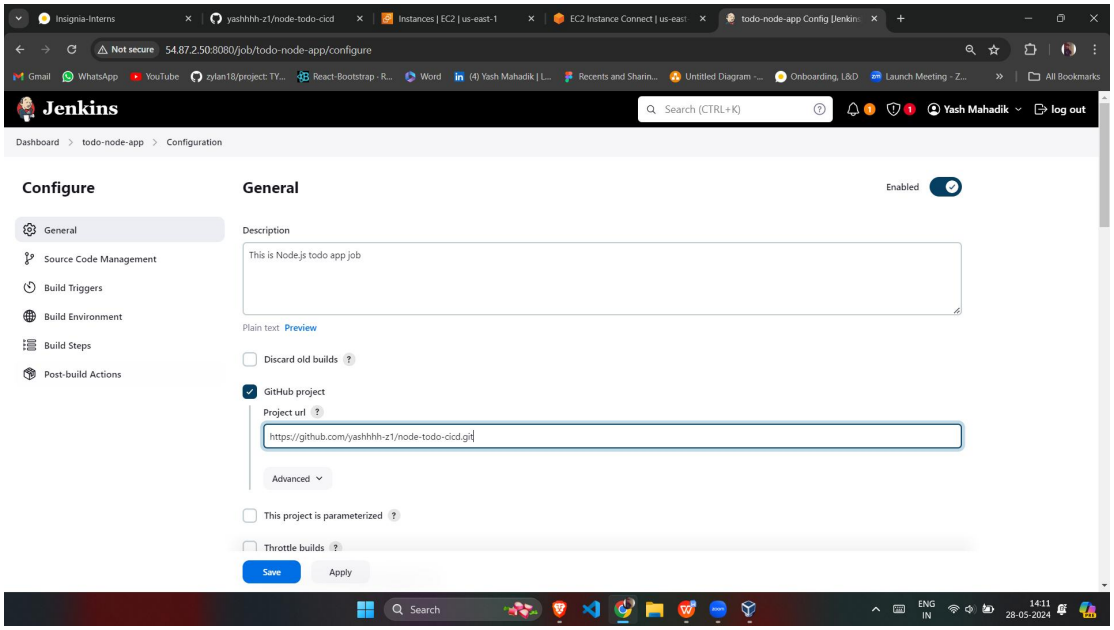




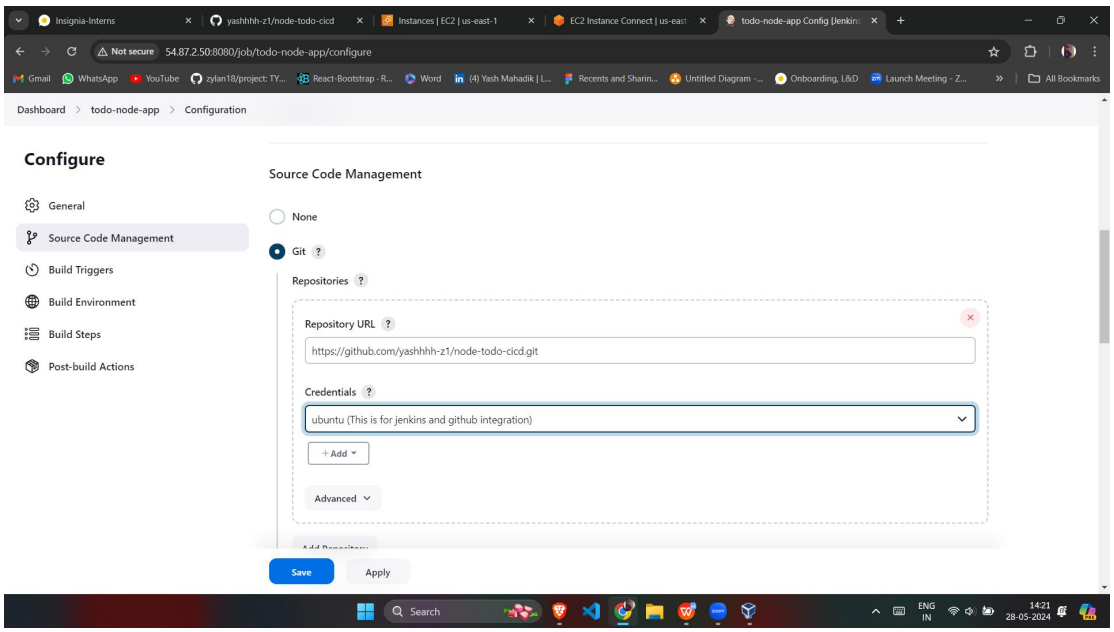
9) In Jenkins Dashboard, Go to new item and create a job by giving name as toto-node-app click on Freestyle project and click on ok.



10) As our source code is stored in github so we need give our project URL.

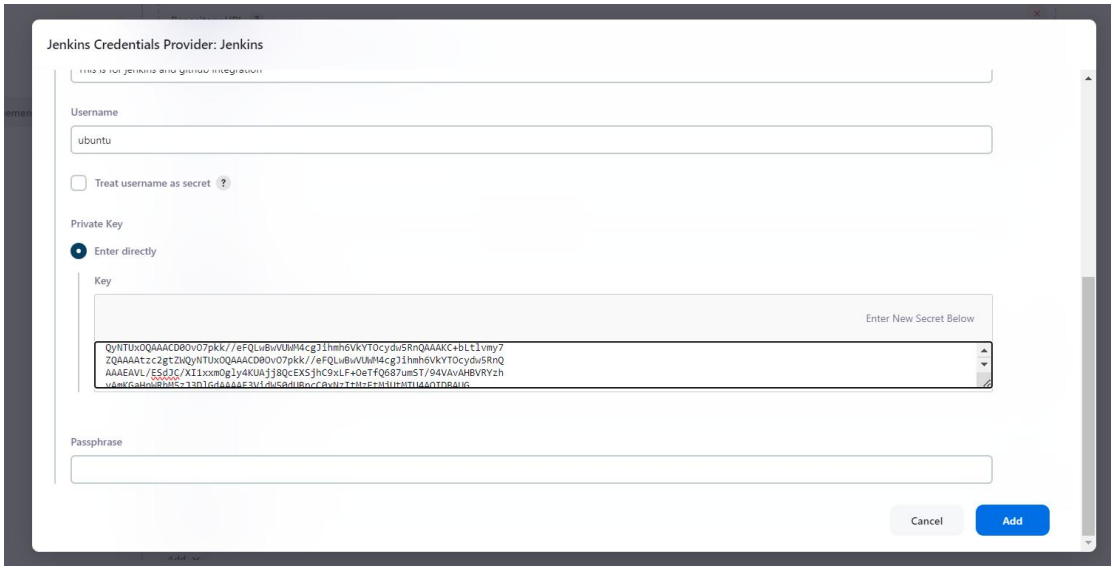


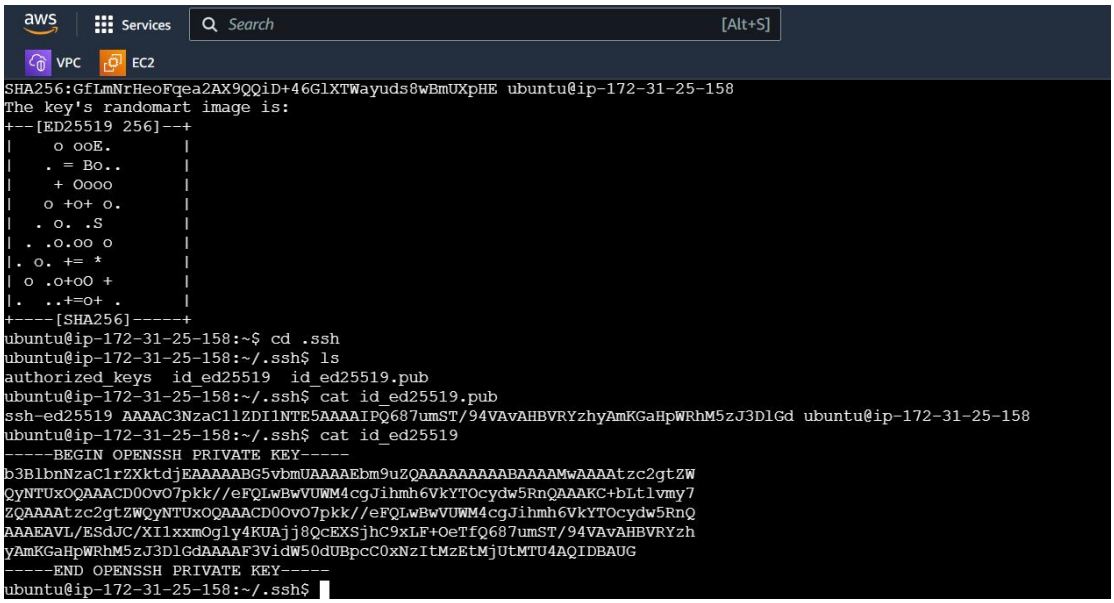
11) Under Source code Management , we will be checking out code from git , so select git provide URL and Credentials.



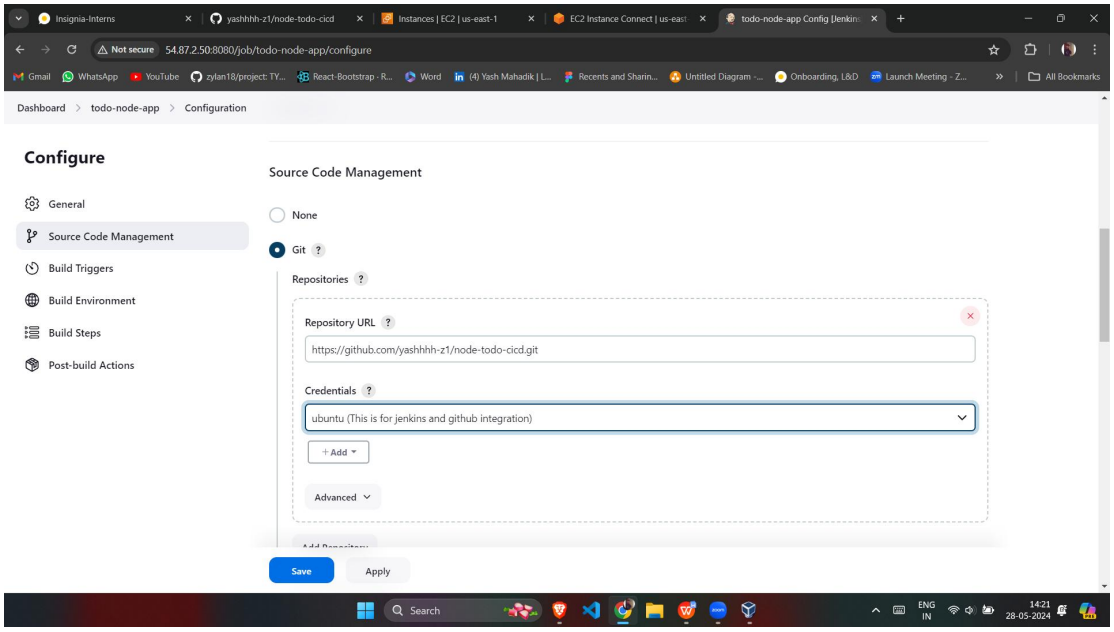
12) This is how we can create Credentials, the domain section will be Global and in kind section as we already have our access toke so we will use **SSH username with private key**, again Scope will be Global, give ID as jenkins-github , Description part is optional , username as Ubuntu.

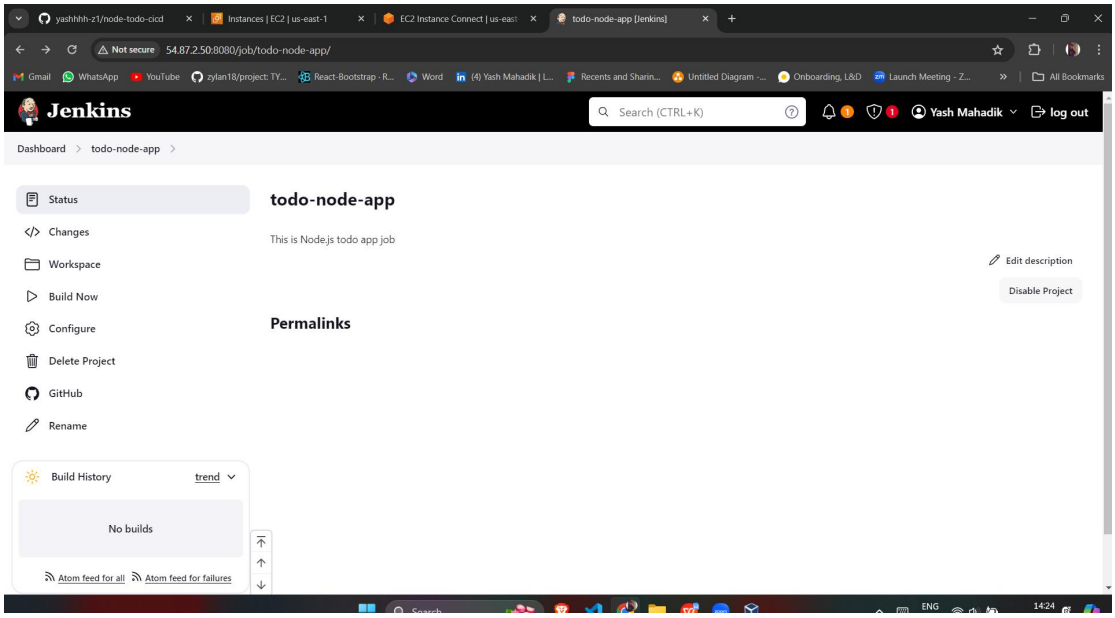
13) Under Private key copy paste the private key from terminal just like public key and click on add , your credentials are ready to add.



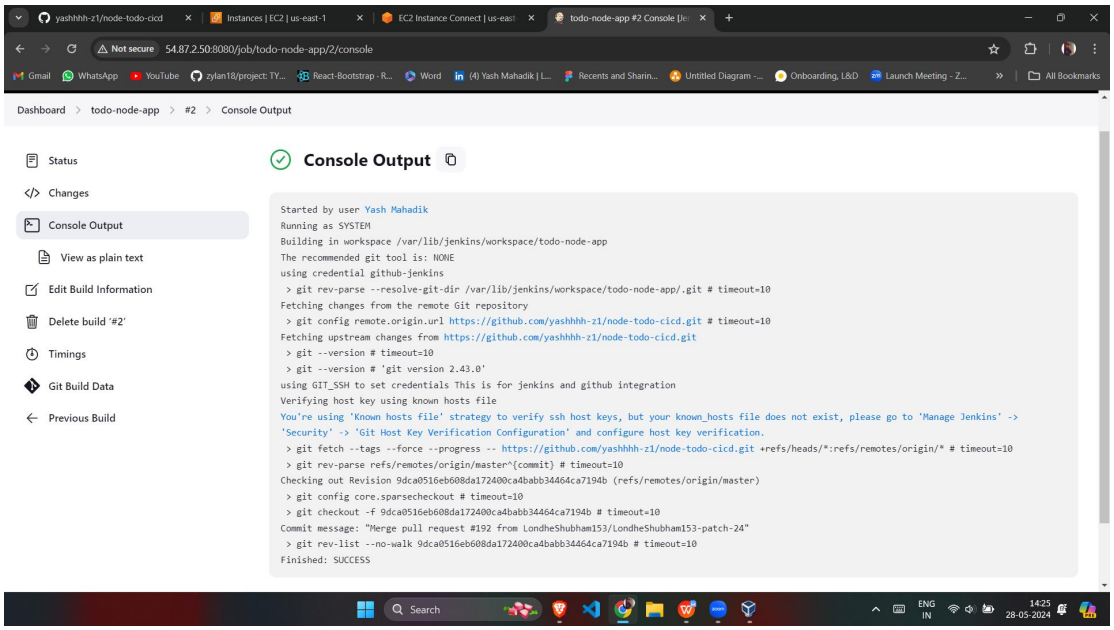


14) Click on Save and your job has been created.



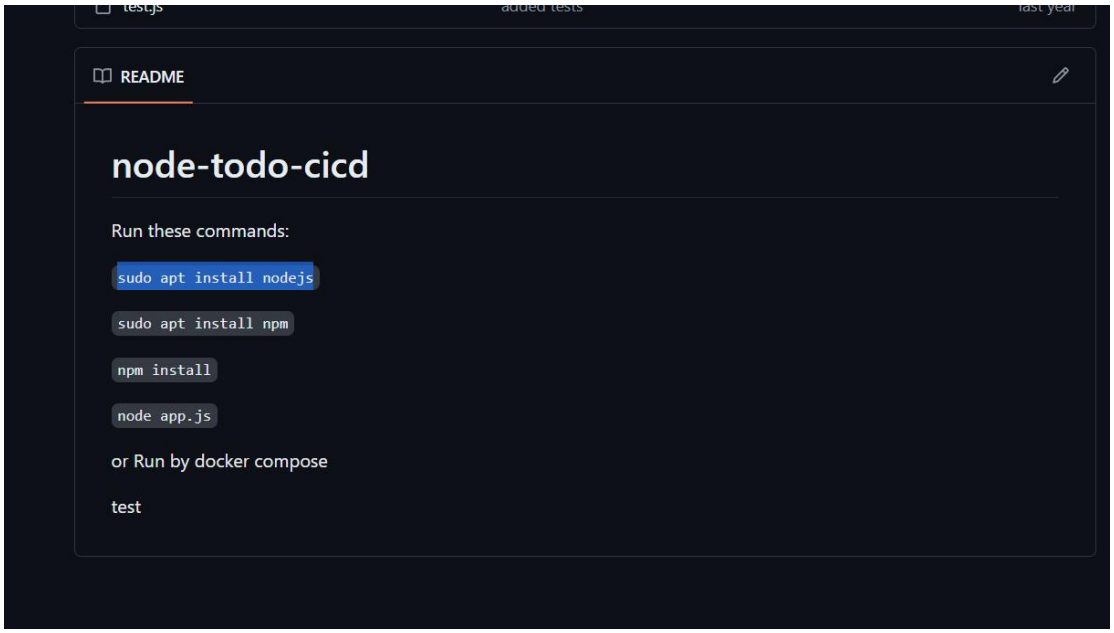


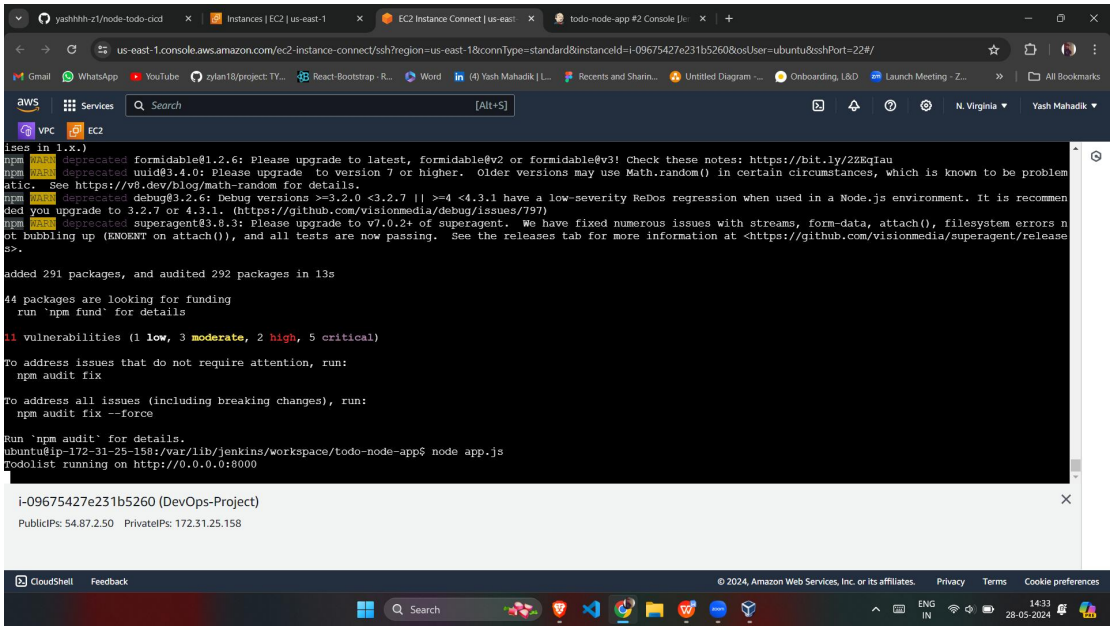
15) Click on Build Now , inside the console output you will see the job has been successfully build and Jenkins have pull all the file from github to our local machine.



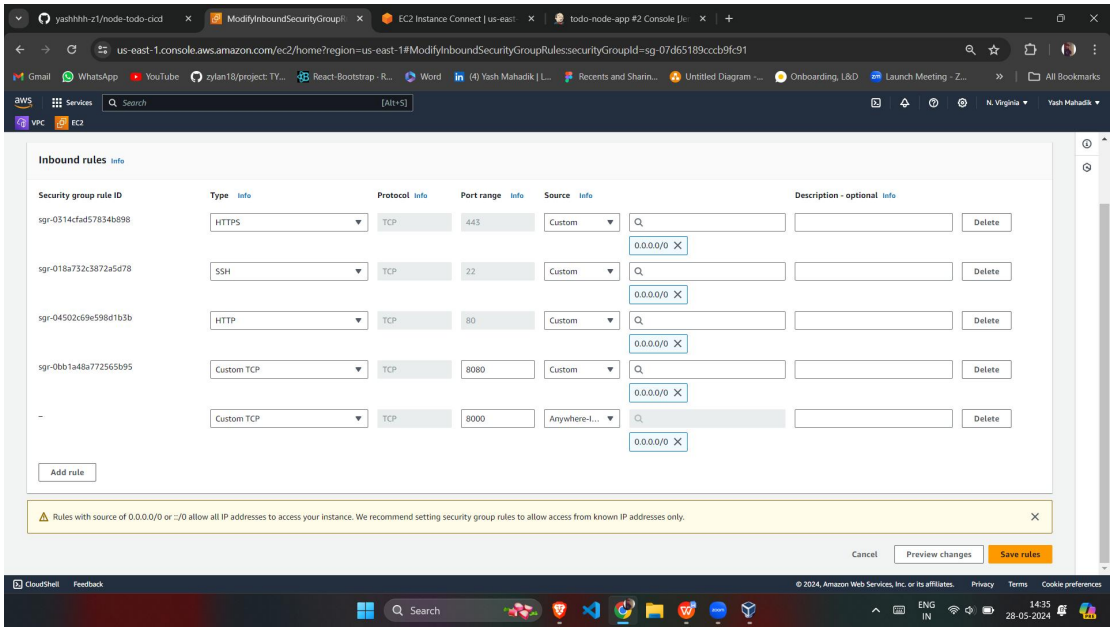
In your terminal you can check the build path, as all the file has been pull to your machine.

16) In github , we have a Readme file to run the node app, so need to install all the following in your local machine to runn the Node.js app.

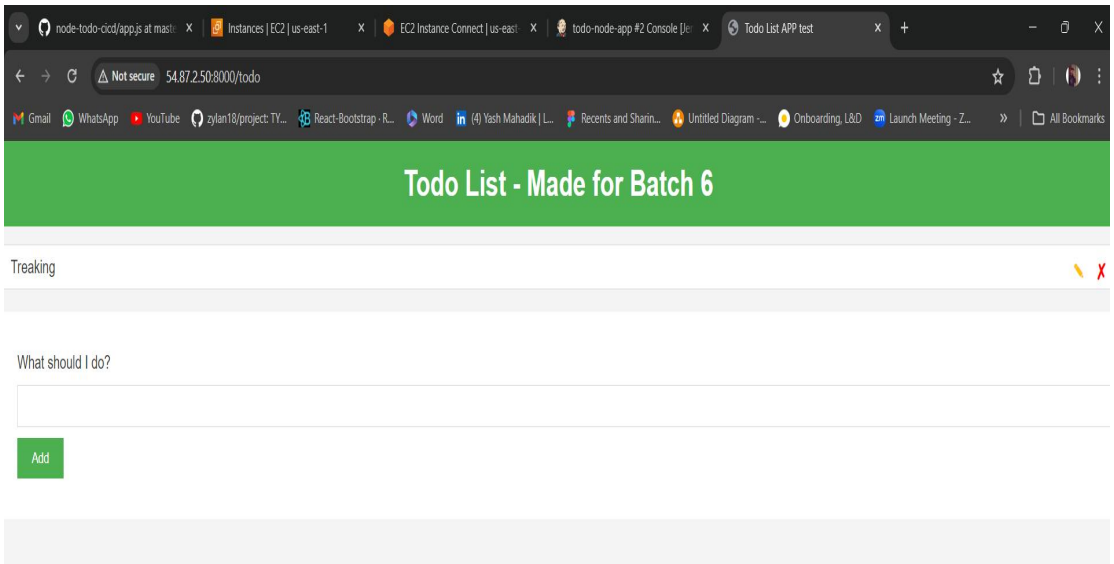




17) Once done with Installing, as our Node.js is running on port 8000, again we need to add inbound rule for port 8000.

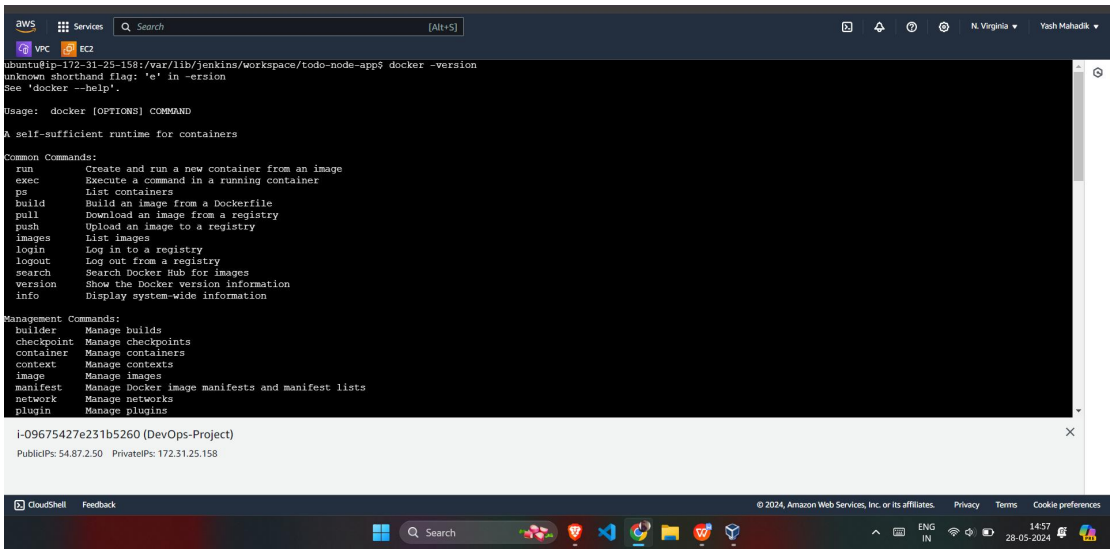


18) Boom, You can see your Node.js project has been running on port 8000 and its successfully deployed on your machine. But if we shut down our machine the port 8000 will be down too. So lets create a docker image so we can keep running port 8000 constantly.



19) Delete the old docker file and create a own docker by self , First we need to install docker using following command.

sudo apt install docker.io



20) Create new dockerfile and write down the following instruction in vim.

sudo vim dockerfile

FROM node:12.2.0-alpine

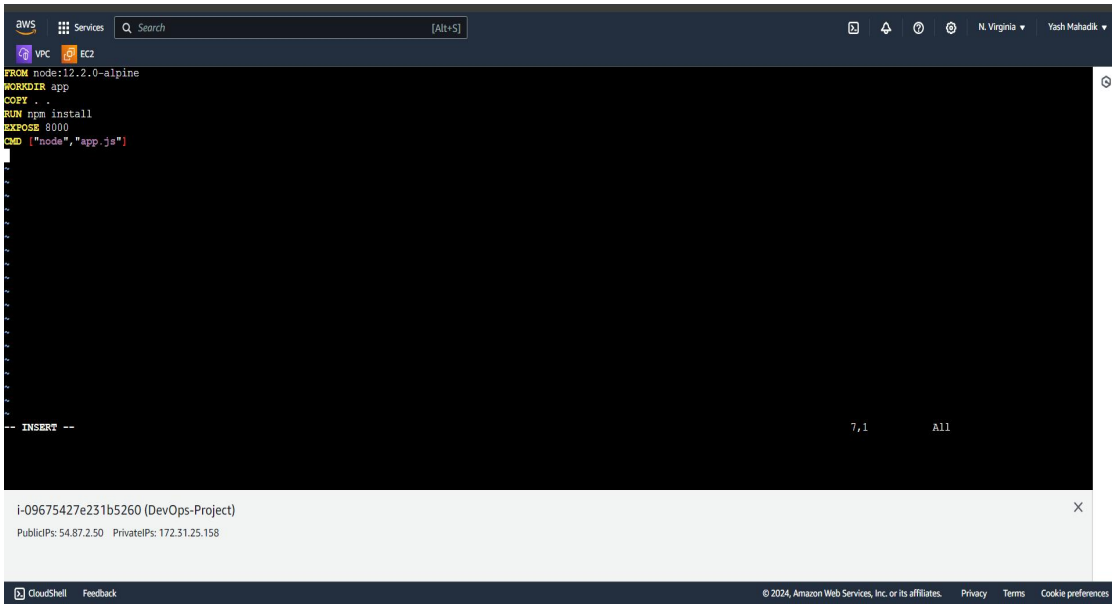
WORKDIR app

COPY . .

RUN npm install

EXPOSE 8000

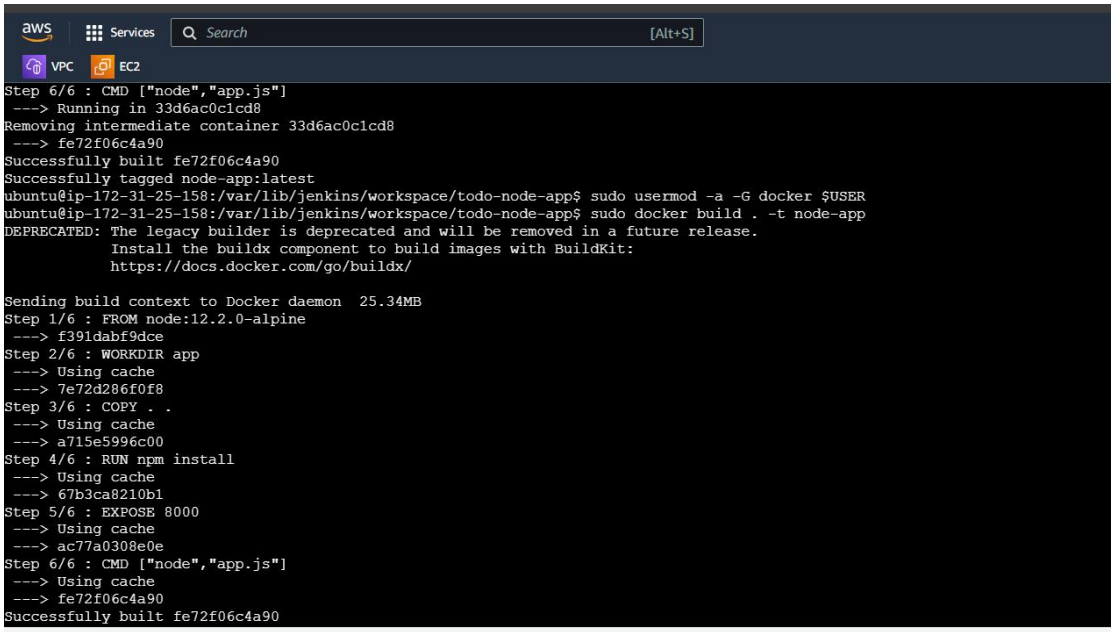
CMD ["node","app.js"]



21) Once the dockerfile is ready we need to build the image for our node.js app, using following command.

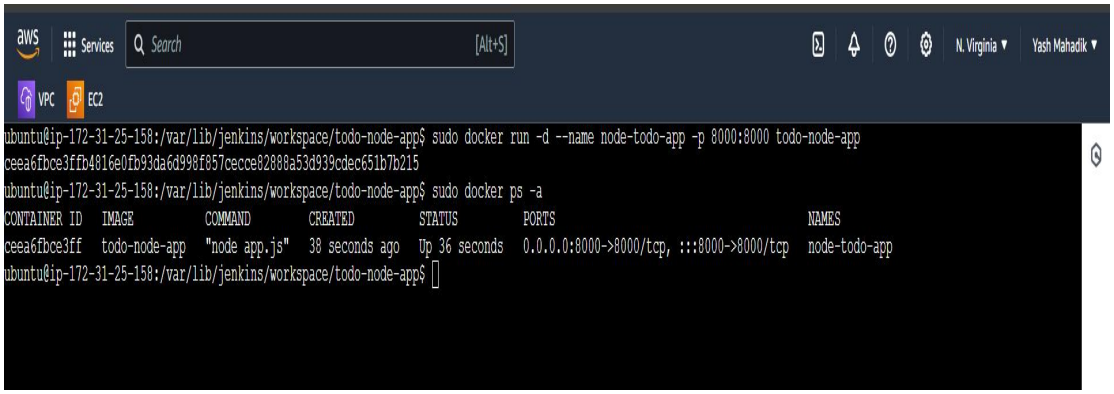
sudo usermod -a -G docker $USER // For permission

docker build . -t todo-node-app

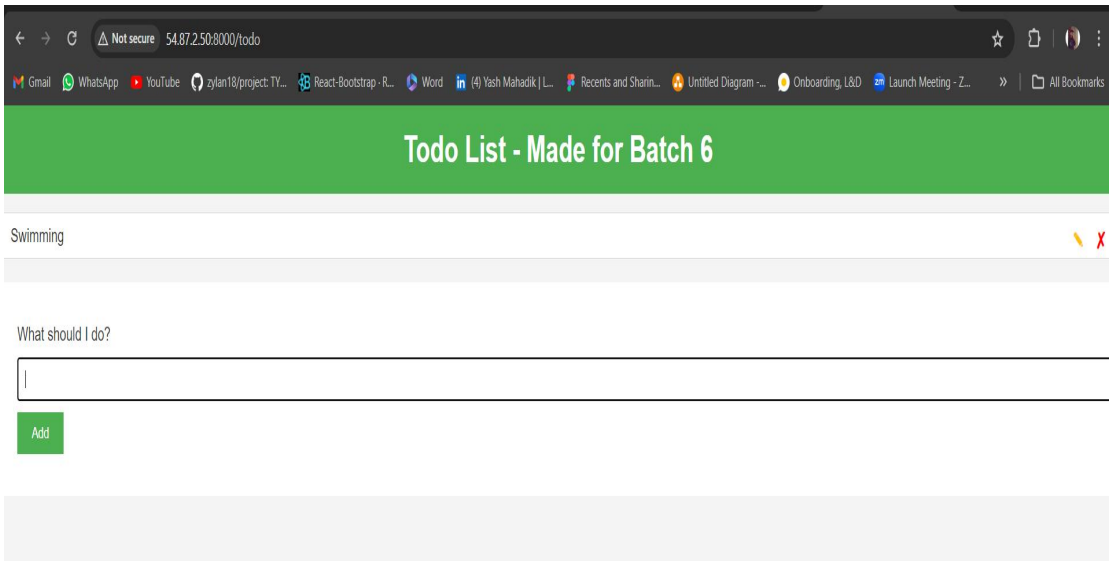


22) Our image id Build Successfully as to Run docker container use command.

sudo docker run -d --name node-todo-app -p 8000:8000 todo-node-app



23) You will see Port 8000 is running successfully and again our node.js app is deployed.

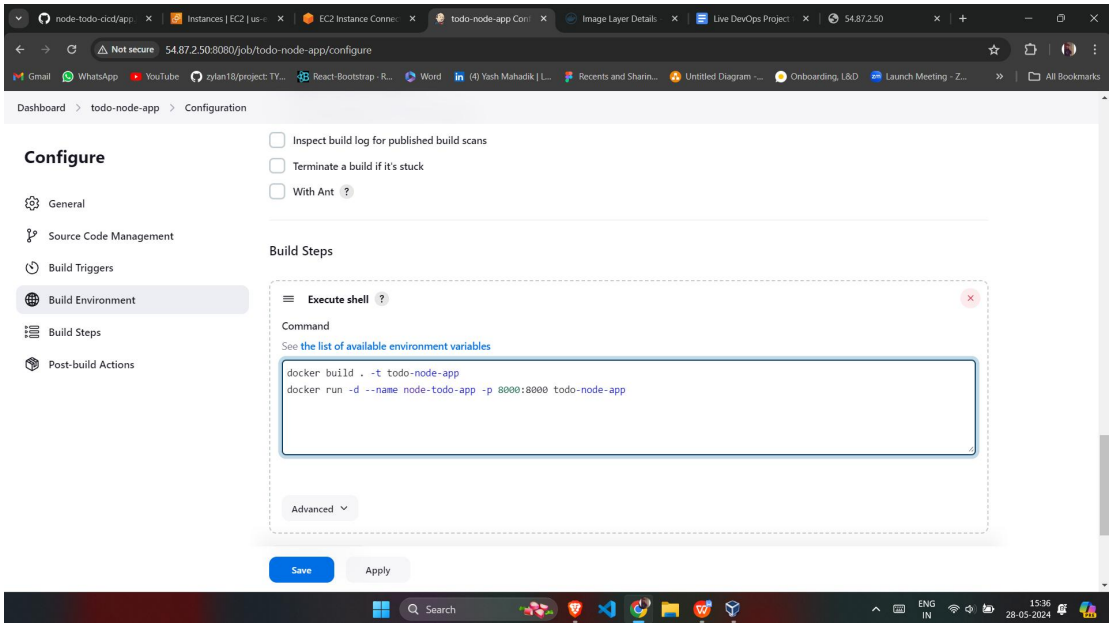


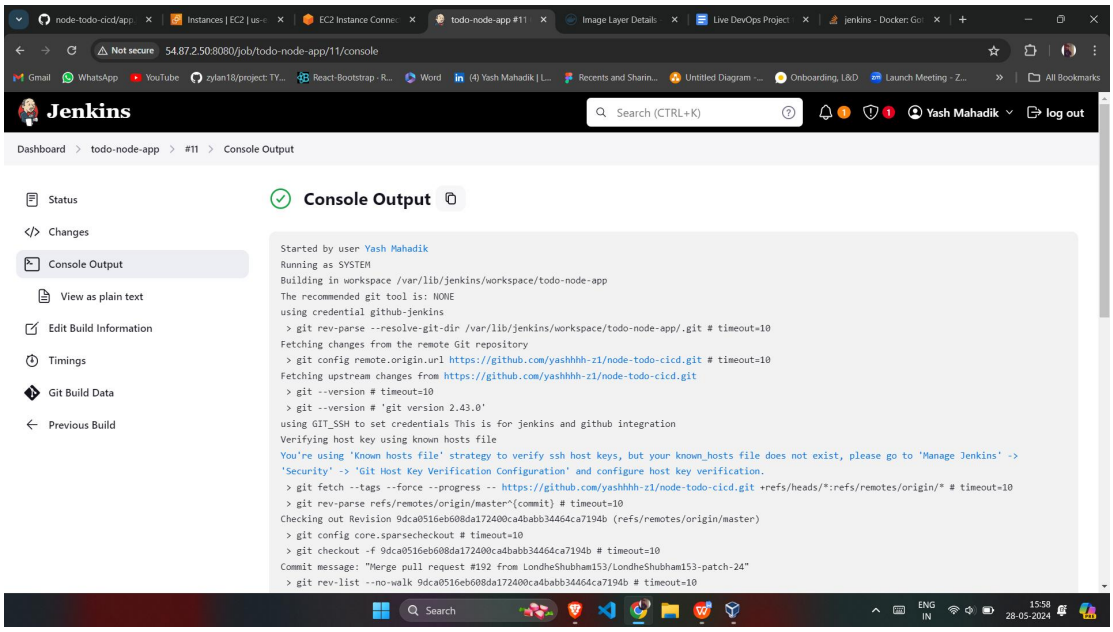
24) We did all the deployment manually , lets see how can we automate the whole deployment using Jenkins CI (Continuous integration).
As we already created a Jenkins job, We need to configure go to Build Steps and select Execute Shell, and just write down the commands.

docker build . -t todo-node-app

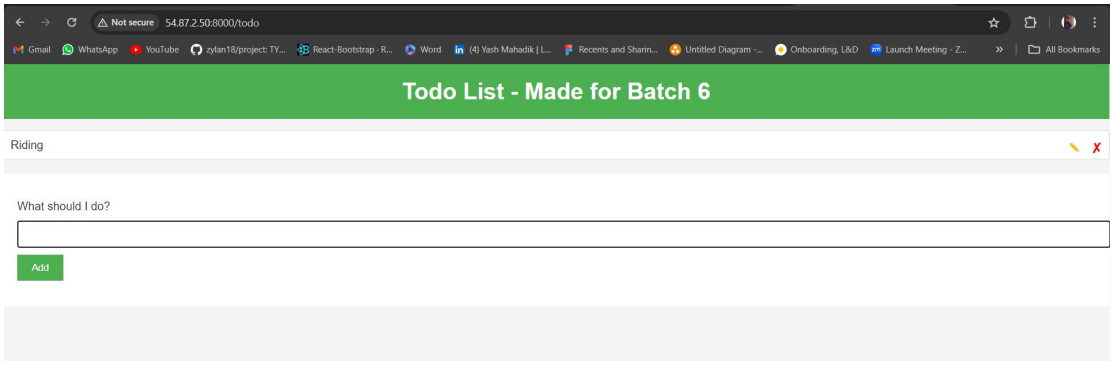
sudo docker run -d --name node-todo-app -p 8000:8000 todo-node-app

Click on Save and click on Build now, might get permission denied so use **sudo usermod -a -G docker jenkins** and do restart your jenkinks.

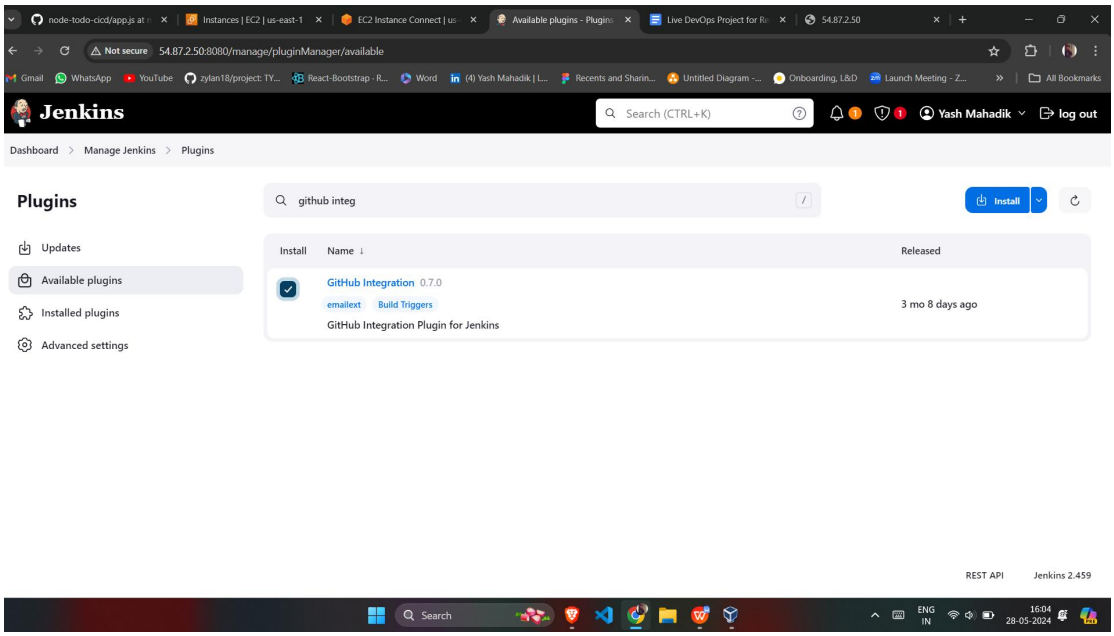


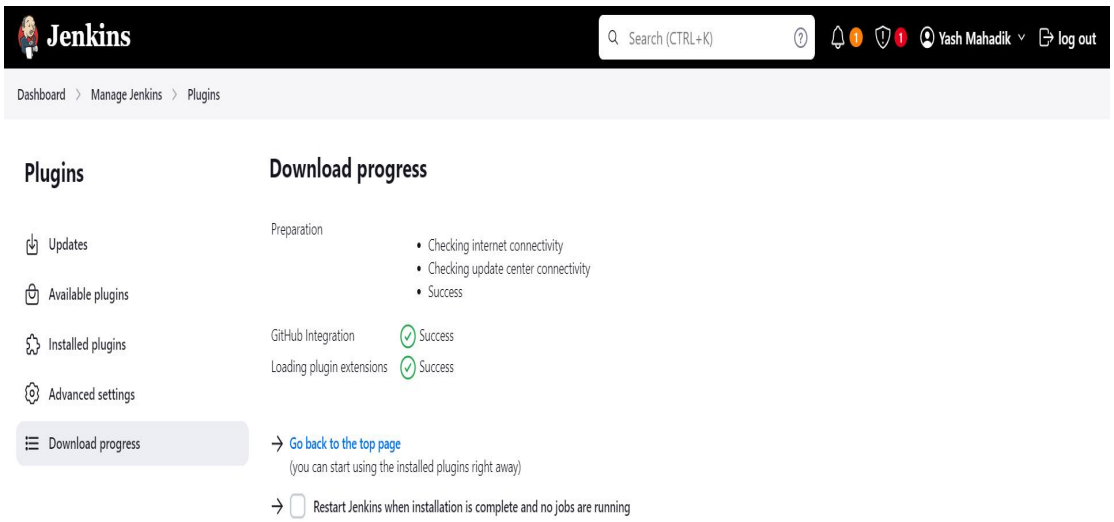


25) Our node.js app is successfully running.

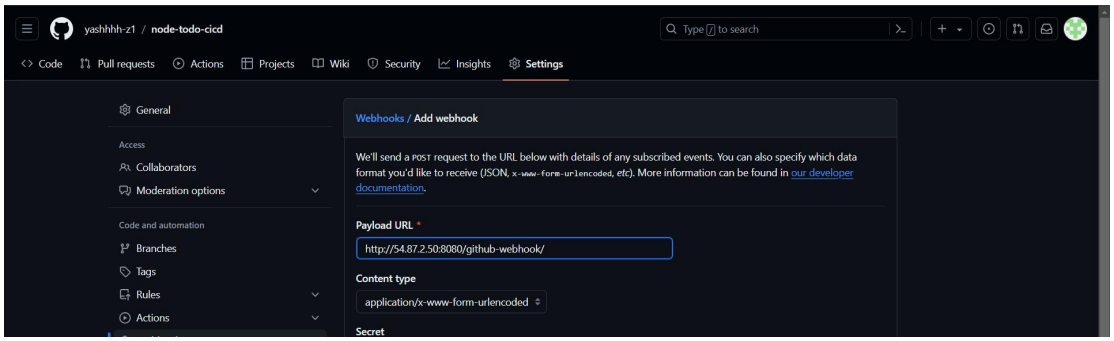


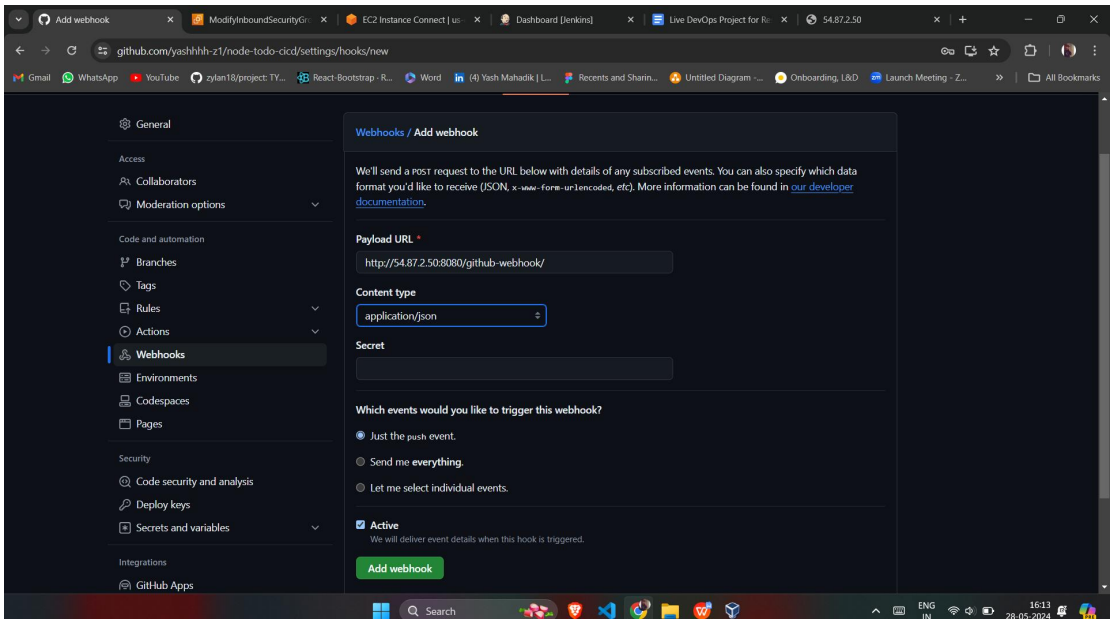
26) Lets automate with webhooks , First need to Install plugin in jenkins
Github Integration.

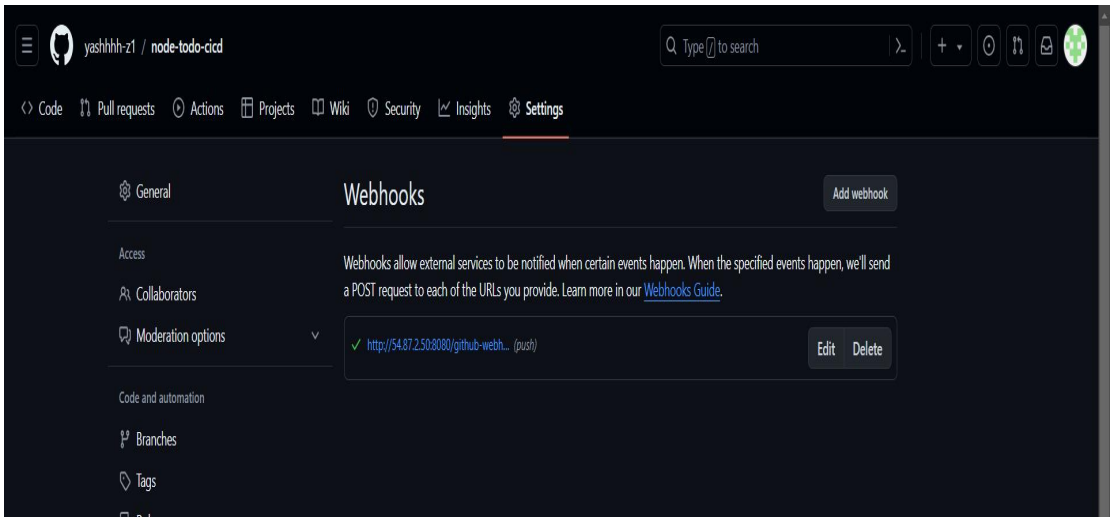




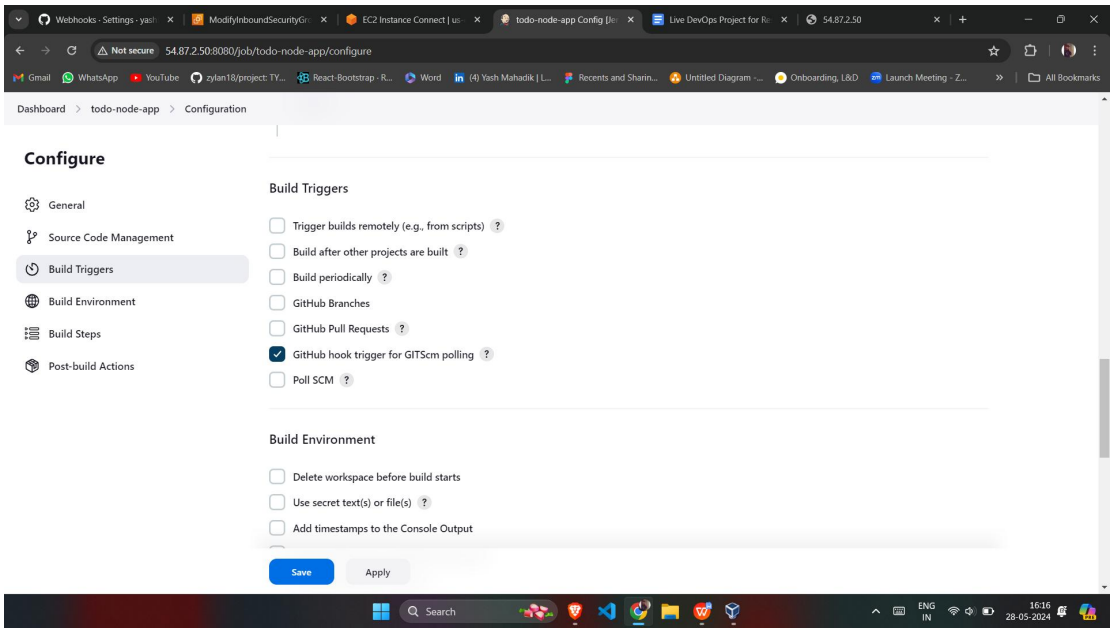
27) Configure webhooks in github Go to Repository Setting > webhooks.
Add webhook , In Payload URL, give the URL of jenkins with content type as json and add webhook.



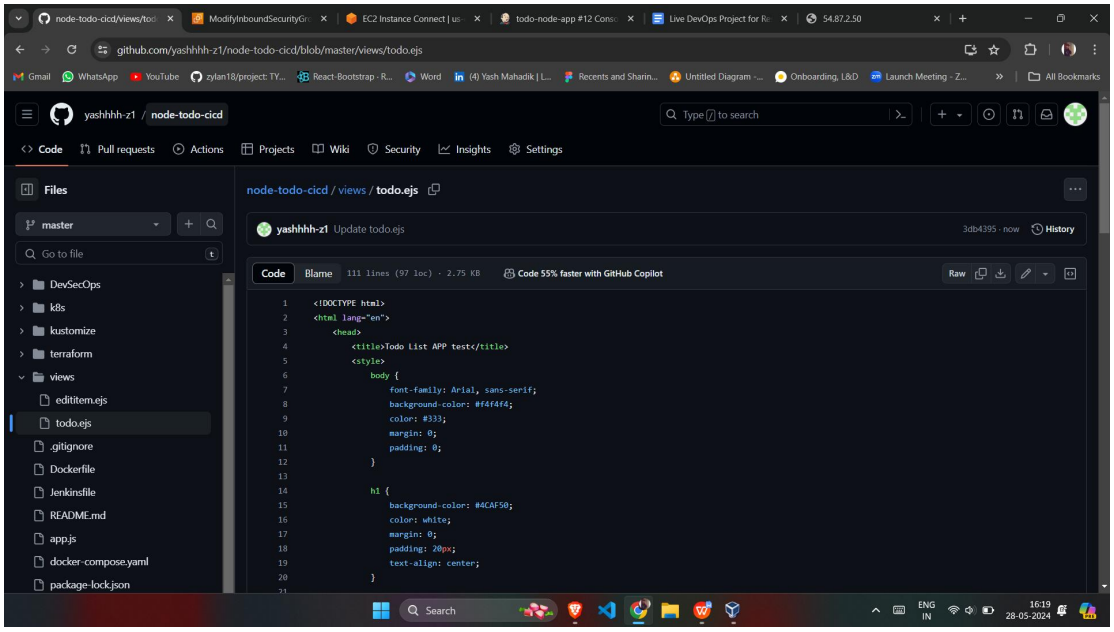


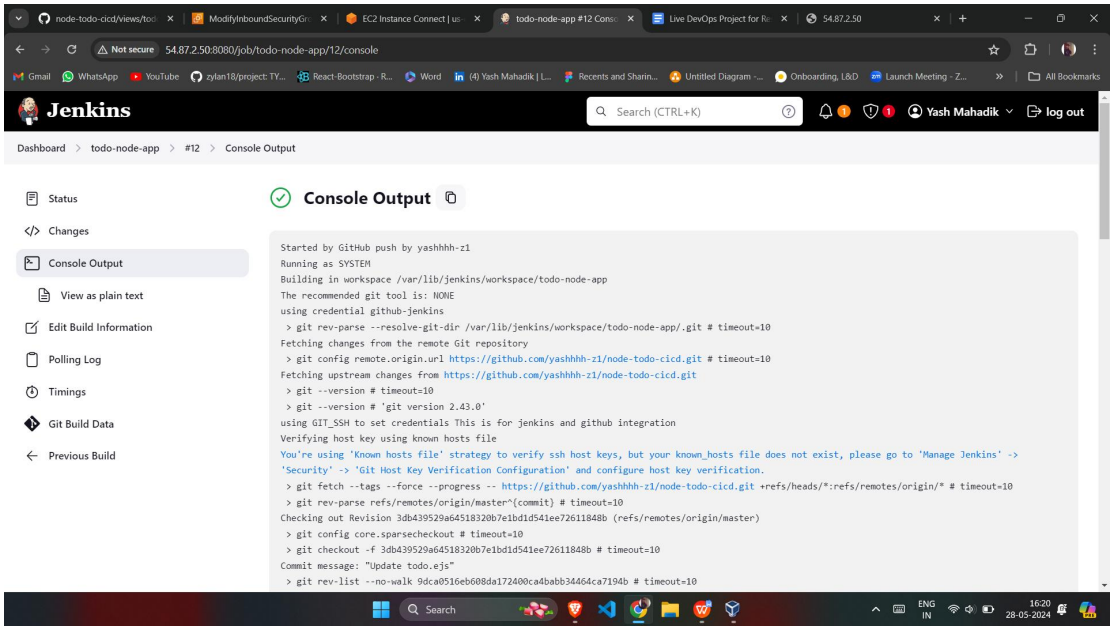


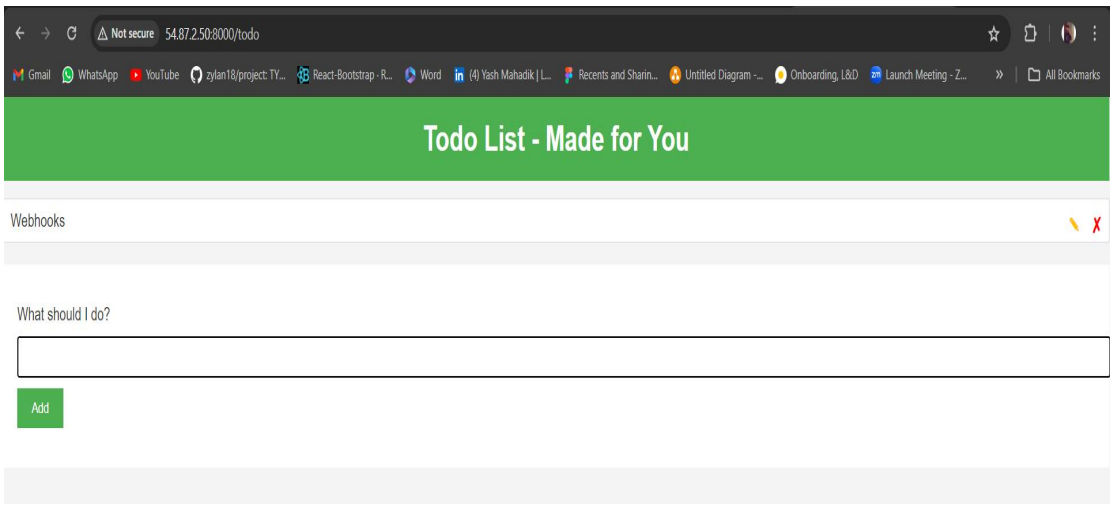
28) Go to jenkins in configue job , under Build Triggers , Check box on GitHub hook trigger for GITScm polling and Click on save.



29) Do some Changes in code and commit it so Jenkins will trigger and build automatically.




Not secure 54.87.2.50:8080/job/todo-node-app/12/console
Jenkins
Search (CTRL+K)
Yash Mahadik
log out
Dashboard
todo-node-app
#12
Console Output
Status
Changes
Console Output
View as plain text
Edit Build Information
Polling Log
Timings
Git Build Data
Previous Build
Console Output
Started by GitHub push by yashhhh-z1
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/todo-node-app
The recommended git tool is: NONE
using credential github-jenkins
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/todo-node-app/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/yashhhh-z1/node-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/yashhhh-z1/node-todo-cicd.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
using GIT_SSH to set credentials This is for jenkins and github integration
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.
> git fetch --tags --force --progress -- https://github.com/yashhhh-z1/node-todo-cicd.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 3db439529a64518320b7e1bd1d541ee72611848b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 3db439529a64518320b7e1bd1d541ee72611848b # timeout=10
Commit message: "Update todo.ejs"
> git rev-list --no-walk 9dca0516eb608da172400ca4babb34464ca7194b # timeout=10
Search
ENG
IN
16:20
28-05-2024


Not secure 54.87.2.50:8000/todo
Todo List - Made for You
Webhooks
What should I do?
Add