

## DevOps Project:

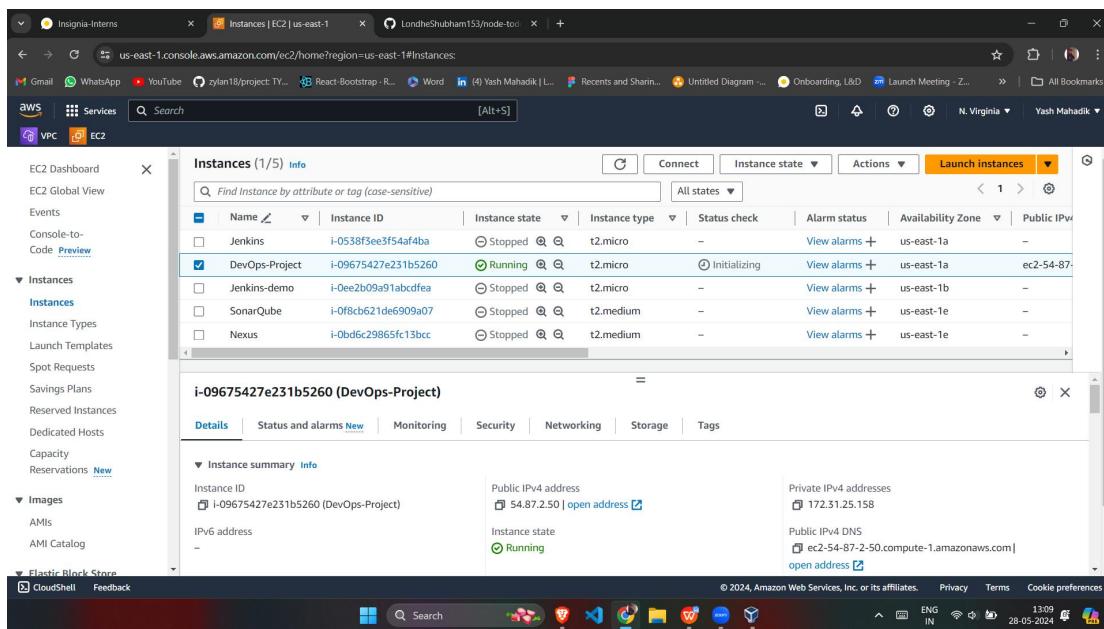
### Deploying a Node.js application with GitHub integration using Jenkins

#### Introduction:

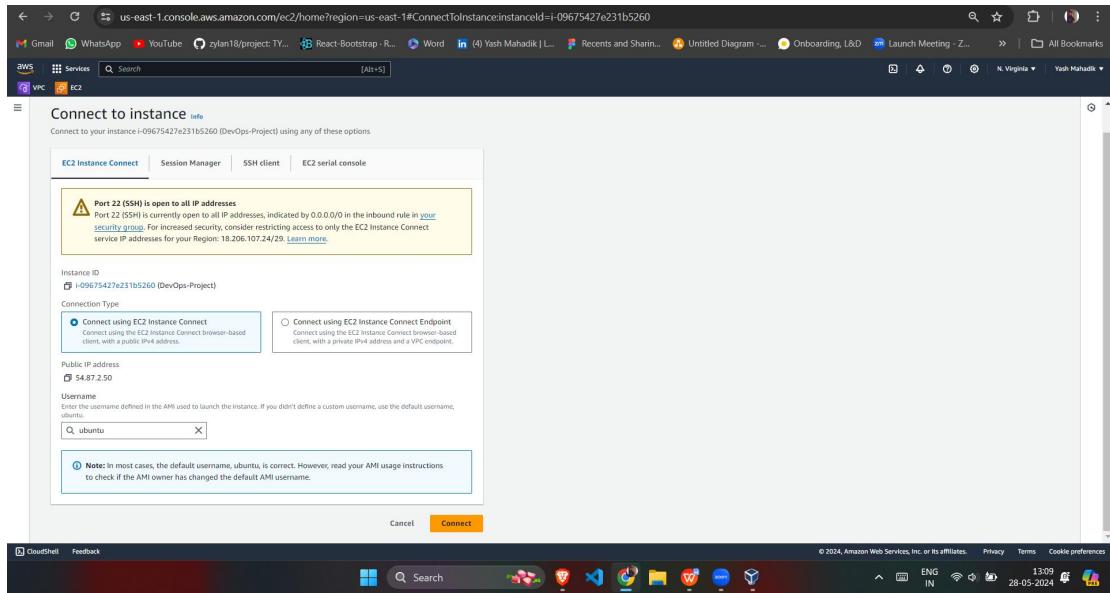
Deploying a Node.js application with GitHub integration using Jenkins involves automating the entire build and deployment pipeline to ensure a seamless and efficient workflow. This process is crucial for modern development practices, particularly in continuous integration and continuous deployment (CI/CD)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

#### A step-by-step guide

- 1) Login to your 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

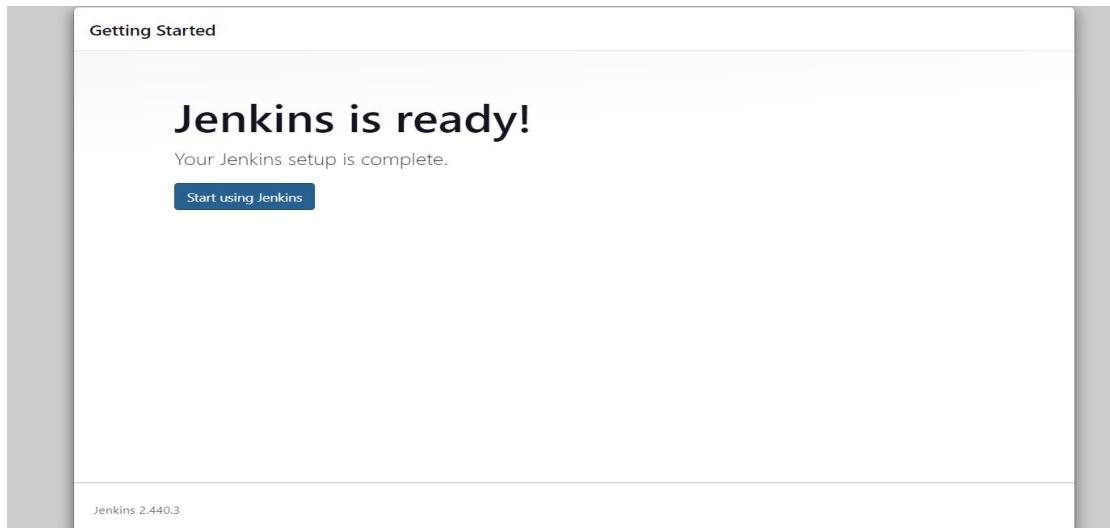


## 3) Need to install Jenkins using following commands

```
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.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0314cfad57834bb98	HTTPS	TCP	443	Custom	
sgr-018a732c3872a5d78	SSH	TCP	22	Custom	
sgr-04502c69e598d1b3b	HTTP	TCP	80	Custom	
-	Custom TCP	TCP	8080	Anywhere (0.0.0.0/0)	

**Add rule**

⚠️ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0314cfad57834bb98	IPv4	HTTPS	TCP	443	0.0.0.0/0	-
-	sgr-018a732c3872a5d78	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sgr-04502c69e598d1b3b	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-0bb1a48a772565...	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	-

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

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

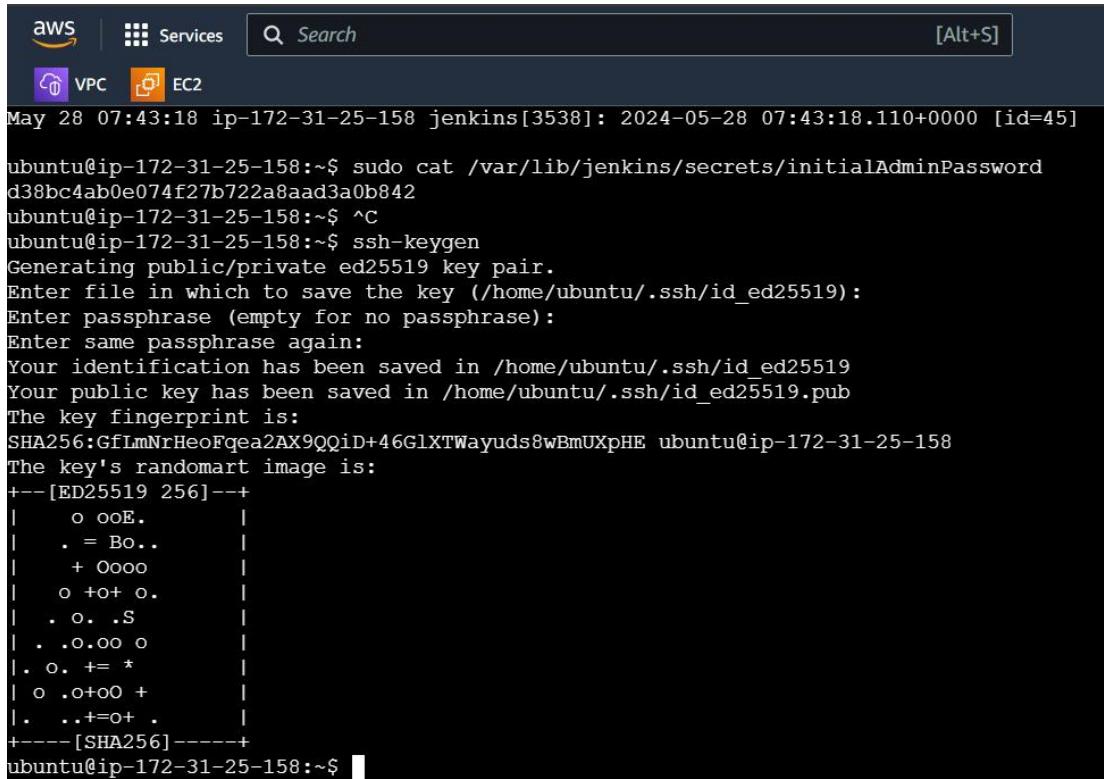
Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

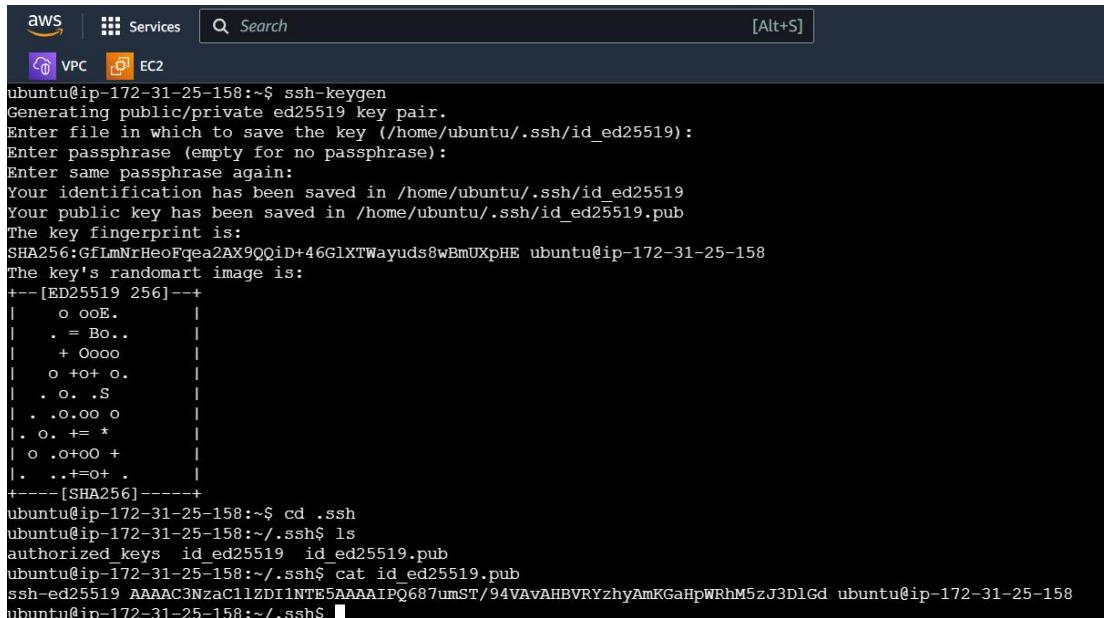
## 6) Now to Generate Access token.

**ssh-keygen**


```

aws | Services Search [Alt+S]
VPC EC2
May 28 07:43:18 ip-172-31-25-158 jenkins[3538]: 2024-05-28 07:43:18.110+0000 [id=45]
ubuntu@ip-172-31-25-158:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
d38bc4ab0e074f27b722a8aad3a0b842
ubuntu@ip-172-31-25-158:~$ ^C
ubuntu@ip-172-31-25-158:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:GfLmNrHeoFqea2AX9QQiD+46GlXTWayuds8wBmUXpHE ubuntu@ip-172-31-25-158
The key's randomart image is:
+--[ED25519 256]--+
|   o ooE.   |
| . = Bo..   |
| + Oooo   |
| o +o+ o.   |
| . o. .S   |
| ..o..o o   |
| . o. += *   |
| o .o+oO +   |
| ..+=o+ .   |
+---[SHA256]---+
ubuntu@ip-172-31-25-158:~$ 

```

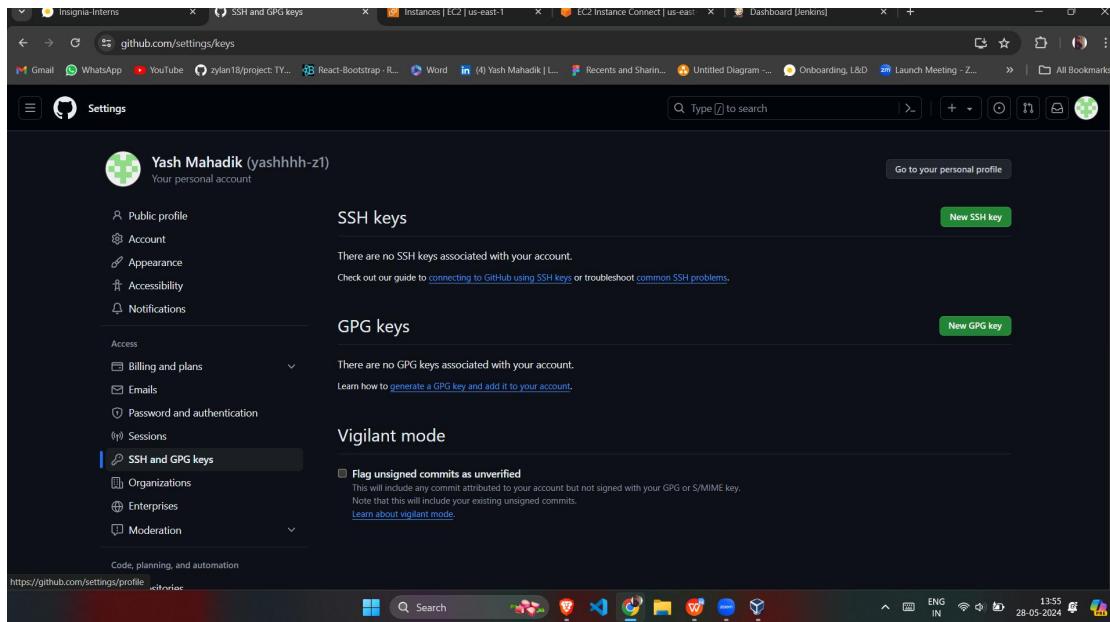
**cd .ssh****ls****cat id\_ed25519.pub // this our public key**


```

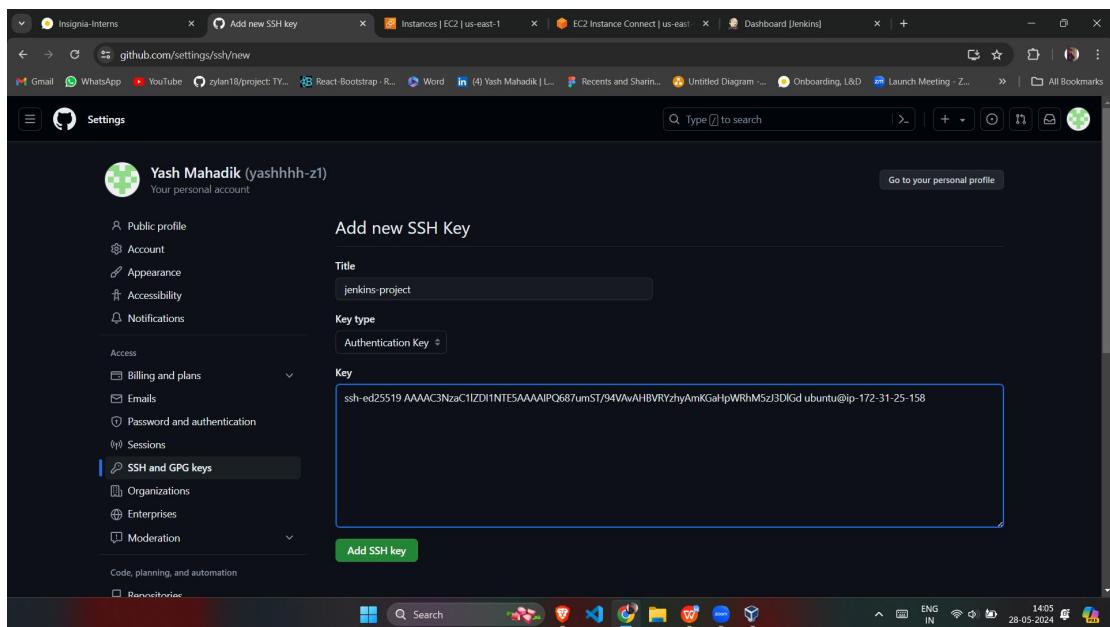
aws | Services Search [Alt+S]
VPC EC2
ubuntu@ip-172-31-25-158:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:GfLmNrHeoFqea2AX9QQiD+46GlXTWayuds8wBmUXpHE ubuntu@ip-172-31-25-158
The key's randomart image is:
+--[ED25519 256]--+
|   o ooE.   |
| . = Bo..   |
| + Oooo   |
| o +o+ o.   |
| . o. .S   |
| ..o..o o   |
| . o. += *   |
| o .o+oO +   |
| ..+=o+ .   |
+---[SHA256]---+
ubuntu@ip-172-31-25-158:~$ cd .ssh
ubuntu@ip-172-31-25-158:~/.ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@ip-172-31-25-158:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAIPO687umST/94VAvAHBVRYzhyAmKGaHpwWRhM5zJ3D1Gd  ubuntu@ip-172-31-25-158
ubuntu@ip-172-31-25-158:~/.ssh$ 

```

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.



This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

**Authentication keys**

	<b>jenkins-project</b>	SHA256: GFLmNrHeoFqea2AX9QQiD+46G1XTWayuds8wBmUXpHE	<b>Delete</b>
Added on May 28, 2024			
Never used — Read/write			

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

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.

New Item

Enter an item name  
todo-node-app

Select an item type

- Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**

OK

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

Configure

**General**

Description  
This is Node.js todo app job

Discard old builds ?

GitHub project

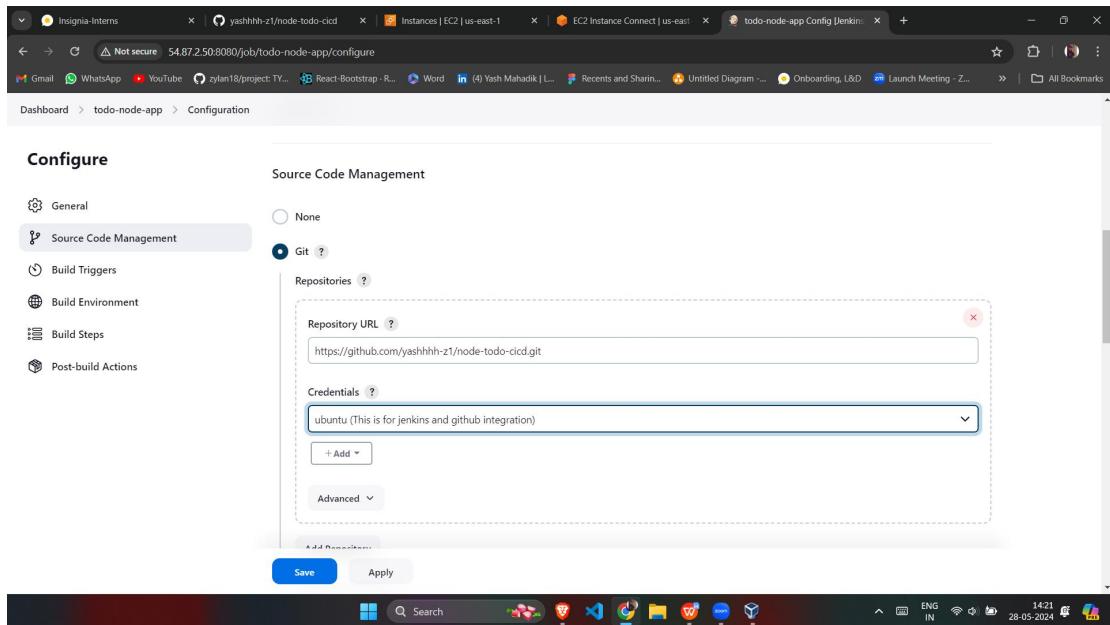
Project url ?

This project is parameterized ?

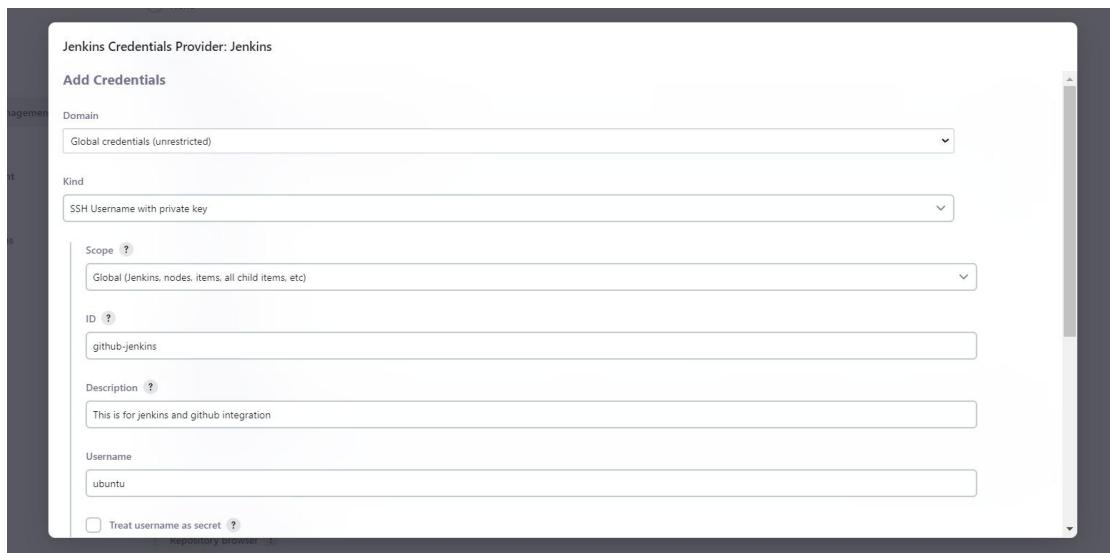
Throttle builds ?

Save Apply

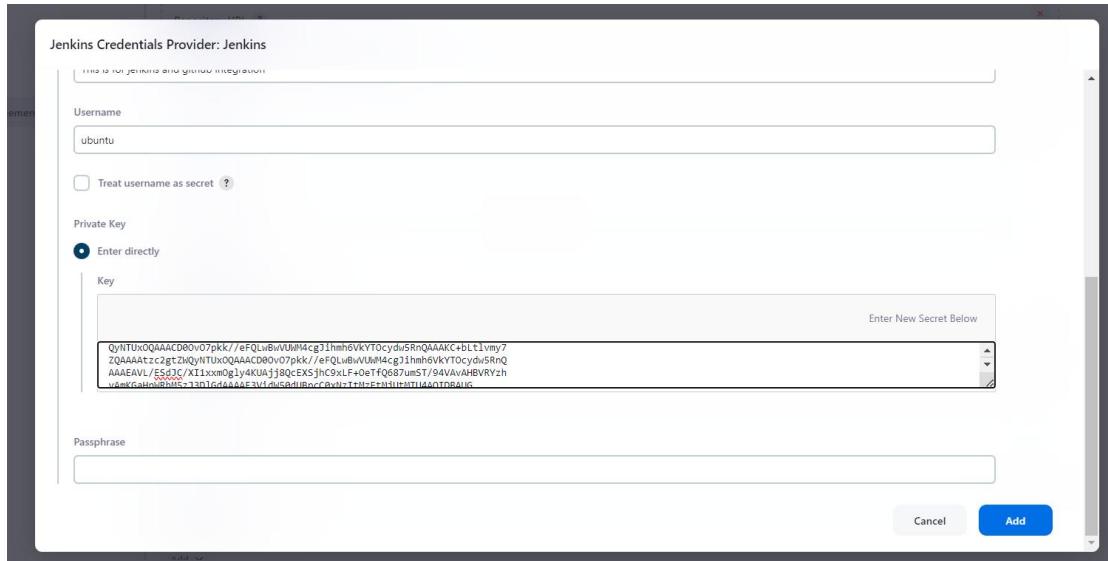
11) Under Source code Management , we will be checking out code from git , so select git provide URL and Credentials.



12) This is how we can create Credentials, the domain section will be Global and in kind section as we already have our access token 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.



aws | Services | Search [Alt+S]

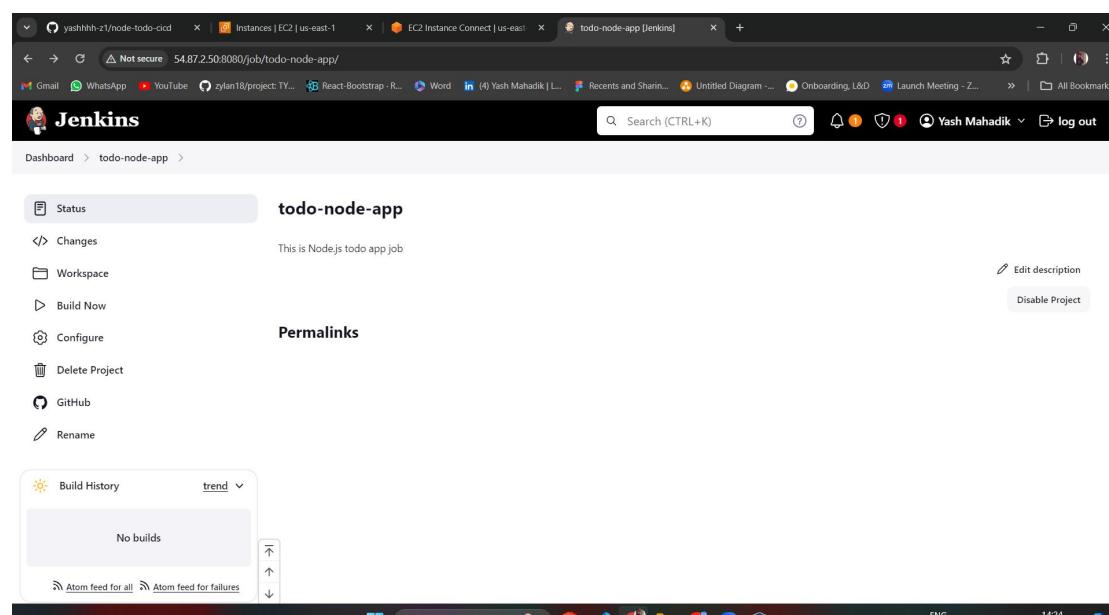
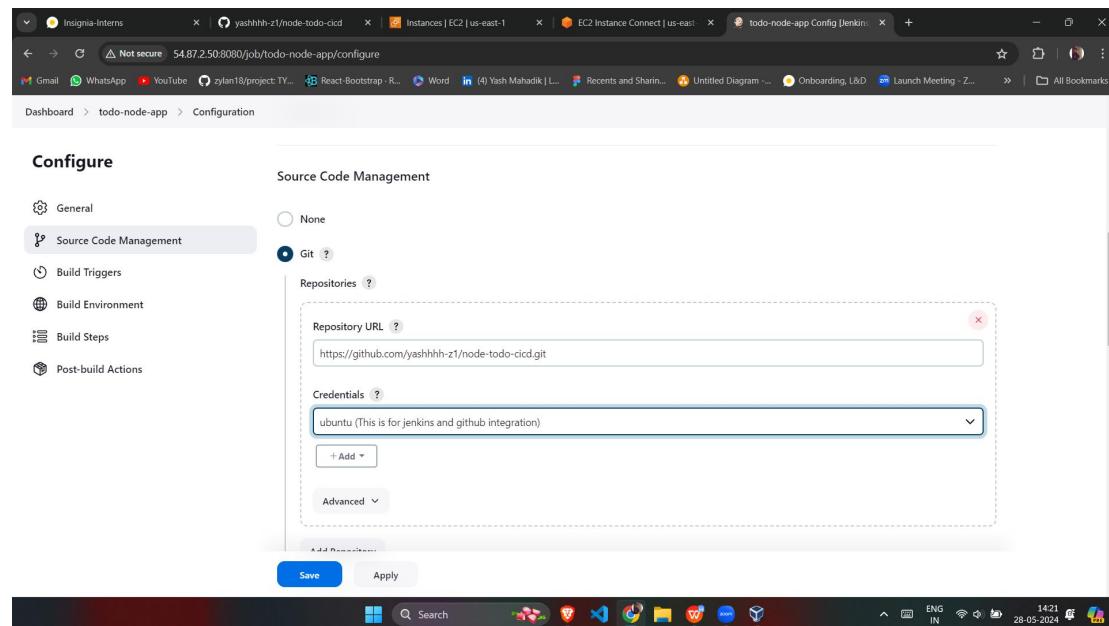
VPC EC2

```

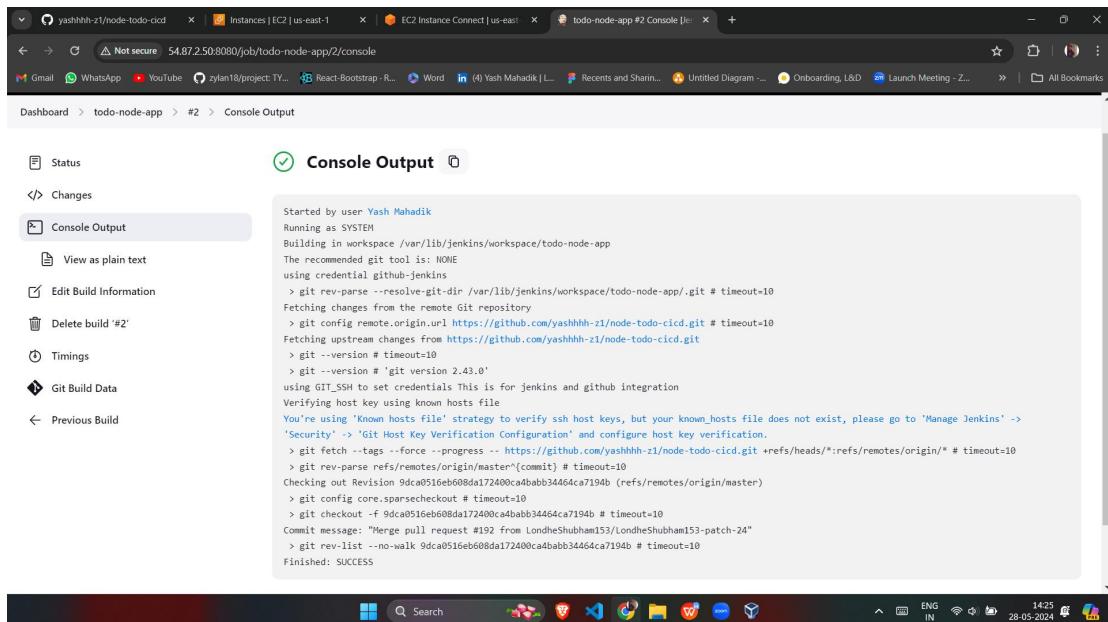
SHA256:GfImNrReoFqea2AX9QQid+46GlXTWayuds8wBmUXpHE ubuntu@ip-172-31-25-158
The key's randomart image is:
++-[ED25519 256]--+
|   o ooE.      |
|   . = Bo..     |
|   + Oooo       |
|   o +o+ o.     |
|   . o. .S      |
|   .. o.oo o    |
|.. o. += *      |
| o .o+oO +     |
| . .+=o+ .      |
+---[SHA256]-----+
ubuntu@ip-172-31-25-158:~$ cd .ssh
ubuntu@ip-172-31-25-158:~/ssh$ ls
authorized_keys id_ed25519 id_ed25519.pub
ubuntu@ip-172-31-25-158:~/ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTB5AAAAIFQ687umST/94VAvAHBVRVYzhyAmKGAhpWRhM5zJ3D1Gd ubuntu@ip-172-31-25-158
ubuntu@ip-172-31-25-158:~/ssh$ cat id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXKtDjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxQAAACD0Ov07pk//eFQlwBwUWM4cgJihmh6VKYT0cydw5RnQAAAKC+bLtlvmy7
ZQAAAAtzc2gtZWQyNTUxQAAACD0Ov07pk//eFQlwBwUWM4cgJihmh6VKYT0cydw5RnQ
AAAAEAVL/ESdJC/XI1xxm0gLy4KUajj8QcEXSjhC9xLF+OeTfQ687umST/94VAvAHBVRVYz
yAmKGAhpWRhM5zJ3D1GdAAAFA3vidW50duBpcC0xNzItMjUtMTU4AQIDBAUG
-----END OPENSSH PRIVATE KEY-----
ubuntu@ip-172-31-25-158:~/ssh$ 

```

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.



```

Started by user Yash Mahadik
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 -v --version # timeout=10
> git -v --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 9dc0516eb608da172400ca4bab34464ca7194b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 9dc0516eb608da172400ca4bab34464ca7194b # timeout=10
Commit message: "Merge pull request #192 from LondheShubham153/LondheShubham153-patch-24"
> git rev-list --no-walk 9dc0516eb608da172400ca4bab34464ca7194b # timeout=10
Finished: SUCCESS

```

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

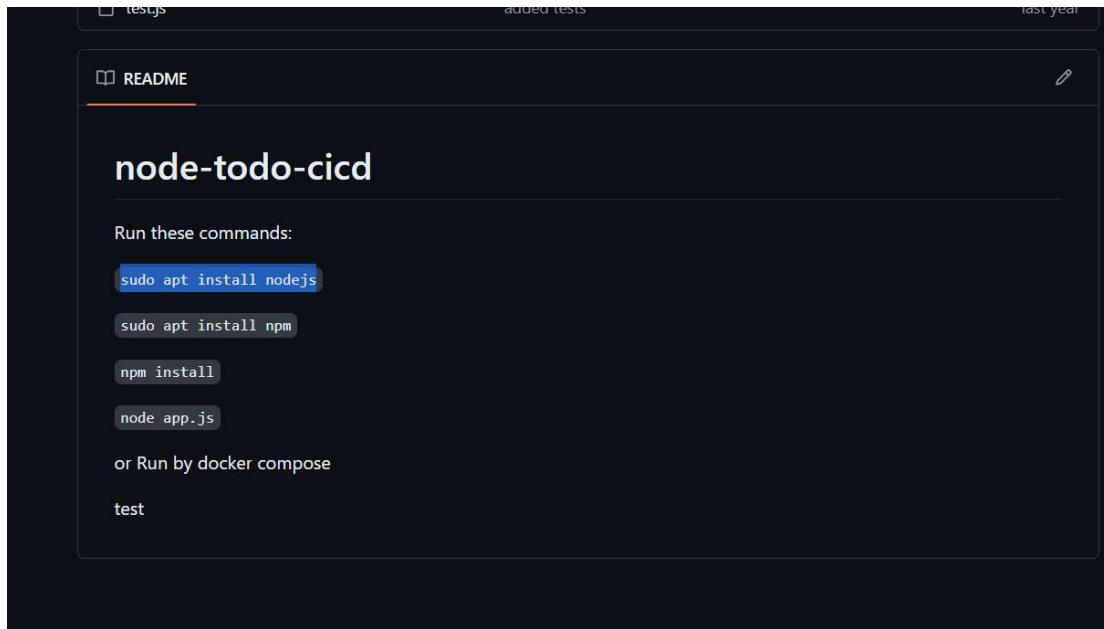


```

ubuntu@ip-172-31-25-158:~/ssh$ cd
ubuntu@ip-172-31-25-158:~$ cd /var/lib/jenkins/workspace/todo-node-app
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ ls
DevSecOps Jenkinsfile app.js           k8s      package-lock.json sonar-project.properties test.js
Dockerfile README.md docker-compose.yaml kustomize package.json      terraform          views
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ 

```

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.



```

yashhhh-zt/node-todo-cicd Instances | EC2 | us-east-1 EC2 Instance Connect | us-east-1 todo-node-app #2 Console [J] test
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-09675427e231b5260&osUser=ubuntu&sshPort=22#/
Gmail WhatsApp YouTube zylan18/project:TY... React-Bootstrap - R... Word in (4) Yash Mahadik | L... Recents and Sharin... Untitled Diagram ... Onboarding, L&D Launch Meeting - Z... All Bookmarks
AWS Services Search [Alt+S]
VPC EC2
is set in 1.x.x)
npm WARN deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZEqlau
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated debug@3.2.6: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDoS regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN deprecated superagent@3.8.3: Please upgrade to v7.0.2+ of superagent. We have fixed numerous issues with streams, form-data, attach(), filesystem errors not bubbling up (KNONOT on attach()), and all tests are now passing. See the releases tab for more information at <https://github.com/visionmedia/superagent/releases>.
added 291 packages, and audited 292 packages in 13s
44 packages are looking for funding
  run 'npm fund' for details
11 vulnerabilities (1 low, 3 moderate, 2 high, 5 critical)
To address issues that do not require attention, run:
  npm audit fix
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ node app.js
TodoList running on http://0.0.0.0:8000
i-09675427e231b5260 (DevOps-Project)
PublicIP: 54.87.2.50 PrivateIP: 172.31.25.158

```

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

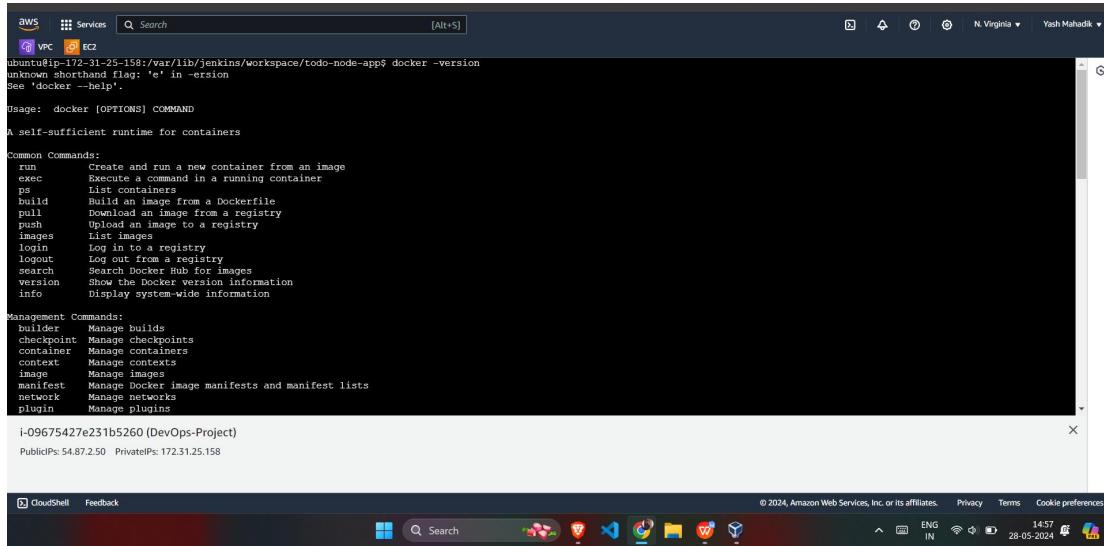
The screenshot shows the AWS CloudWatch Metrics interface. A single metric named "Lambda Function Errors" is visible, with a value of 1. The metric is plotted against time, from May 28, 2024, at 00:00:00Z to 14:35:00Z. The chart has a light blue background with a single data point at the top left.

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.

The screenshot shows a web browser window with the URL `54.87.2.50:8000/todo`. The page title is "Todo List - Made for Batch 6". The main content area displays a single todo item: "Breaking". Below the item is a text input field with the placeholder "What should I do?". At the bottom of the input field is a green "Add" button. The browser's address bar shows the URL and the page title.

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
```



```
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ docker --version
Docker version 20.10.12, build 39ab193
See 'docker --help'.
```

Usage: docker [OPTIONS] COMMAND  
A self-sufficient runtime for containers

Common Commands:

run	Create and run a new container from an image
exec	Execute a command in a running container
ps	List containers
build	Build an image from a Dockerfile
pull	Download an image from a registry
push	Upload an image to a registry
images	List images
login	Log in to a registry
logout	Log out from a registry
search	Search Docker Hub for images
version	Show the Docker version information
info	Display system-wide information

Management Commands:

builder	Manage builds
checkpoint	Manage checkpoints
container	Manage containers
context	Manage contexts
image	Manage images
manifest	Manage Docker image manifests and manifest lists
network	Manage networks
plugin	Manage plugins

i-09675427e231b5260 (DevOps-Project)  
PublicIP: 54.87.2.50 PrivateIP: 172.31.25.158

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"]
```

The screenshot shows the AWS CloudShell interface. At the top, there are navigation links for AWS, Services, and a search bar. Below the search bar, the EC2 service is selected. The main area contains a terminal window with a Dockerfile and its build logs.

```
FROM node:12.2.0-alpine
WORKDIR app
COPY . .
RUN npm install
EXPOSE 8000
CMD ["node", "app.js"]
```

Below the terminal, a message box displays the container ID and IP address:

i-09675427e231b5260 (DevOps-Project)  
PublicIPs: 54.87.2.50 PrivateIPs: 172.31.25.158

At the bottom of the screen, there are footer links for CloudShell, Feedback, and legal information.

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**

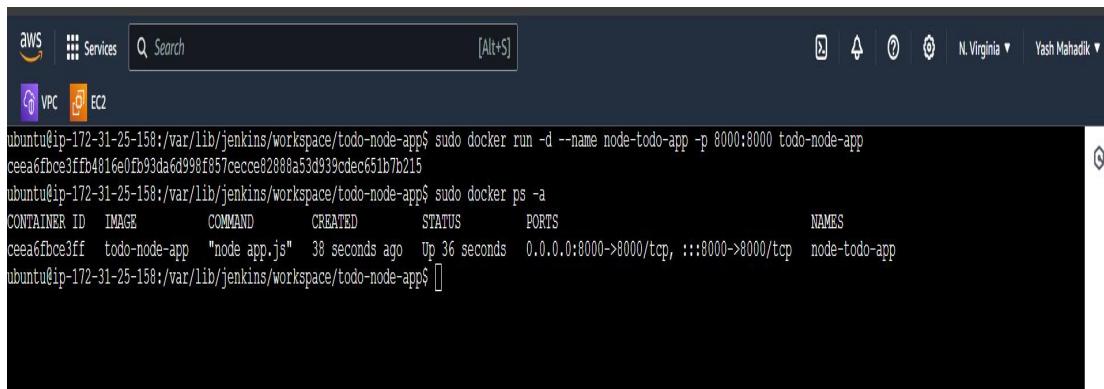
**docker build . -t todo-node-app**

The screenshot shows the AWS CloudShell interface with the terminal window displaying the execution of the `docker build` command. The logs show the creation of a container, the building of the Docker image, and the final successful build message.

```
Step 6/6 : CMD ["node","app.js"]
--> Running in 33d6ac0c1cd8
Removing intermediate container 33d6ac0c1cd8
--> fe72f06c4a90
Successfully built fe72f06c4a90
Successfully tagged node-app:latest
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ sudo usermod -a -G docker $USER
ubuntu@ip-172-31-25-158:/var/lib/jenkins/workspace/todo-node-app$ sudo docker build . -t node-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
    Install the buildx component to build images with BuildKit:
        https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 25.34MB
Step 1/6 : FROM node:12.2.0-alpine
--> f391dabf9dce
Step 2/6 : WORKDIR app
--> Using cache
--> 7e72d286ff0f8
Step 3/6 : COPY .
--> Using cache
--> a715e5996c00
Step 4/6 : RUN npm install
--> Using cache
--> 67b3ca8210b1
Step 5/6 : EXPOSE 8000
--> Using cache
--> ac77a0308e0e
Step 6/6 : CMD ["node","app.js"]
--> Using cache
--> fe72f06c4a90
Successfully built fe72f06c4a90
```

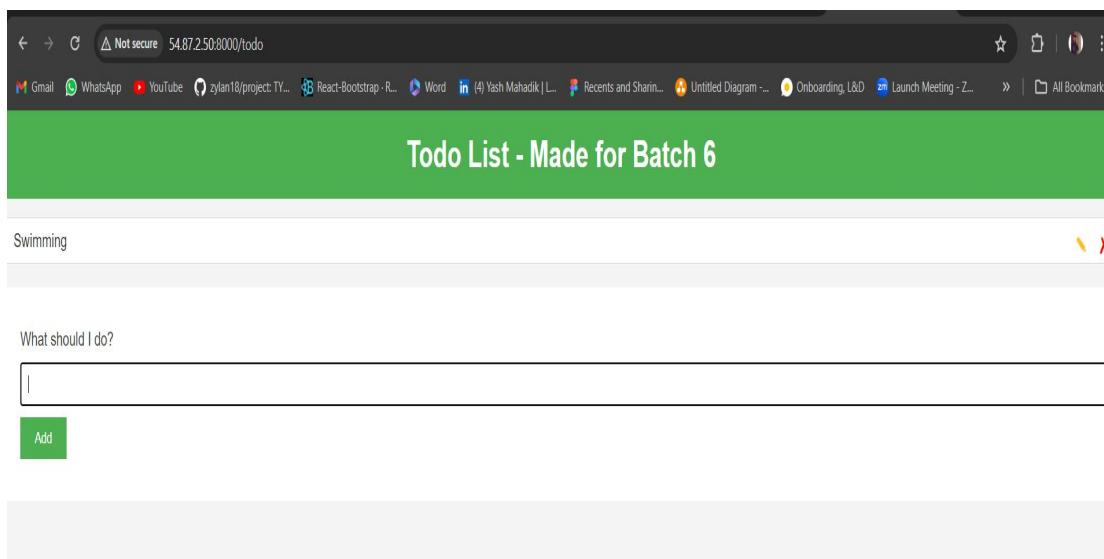
22) Our image id Build Successfully as to Run docker container use command.

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



The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `sudo docker run -d --name node-todo-app -p 8000:8000 todo-node-app`. The terminal output shows the container ID (`ceea6fbce3ff`) and its status as "Up 36 seconds". The port mapping is listed as `0.0.0.0:8000->8000/tcp, :::8000->8000/tcp`, and the name is `node-todo-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 jenkins.

## DevOps Project

The screenshot shows a configuration page for a DevOps project. On the left, a sidebar lists options: General, Source Code Management, Build Triggers, Build Environment (which is selected), Build Steps, and Post-build Actions. The main area is titled 'Configure' and contains a 'Build Steps' section. Under 'Build Steps', there is a step titled 'Execute shell'. The command entered is:

```
docker build . -t todo-node-app  
docker run -d --name node-todo-app -p 8000:8000 todo-node-app
```

At the bottom of the configuration page are 'Save' and 'Apply' buttons.

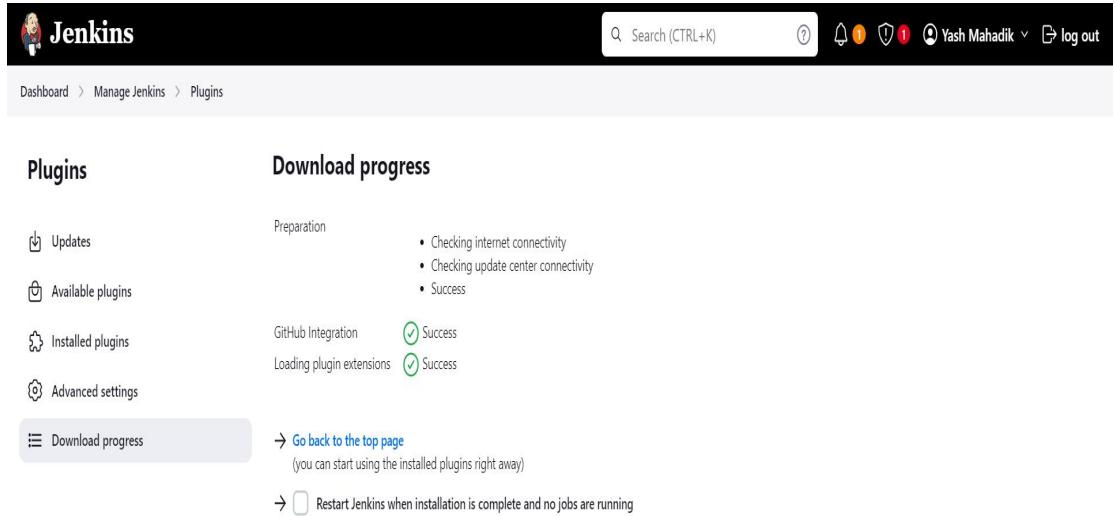
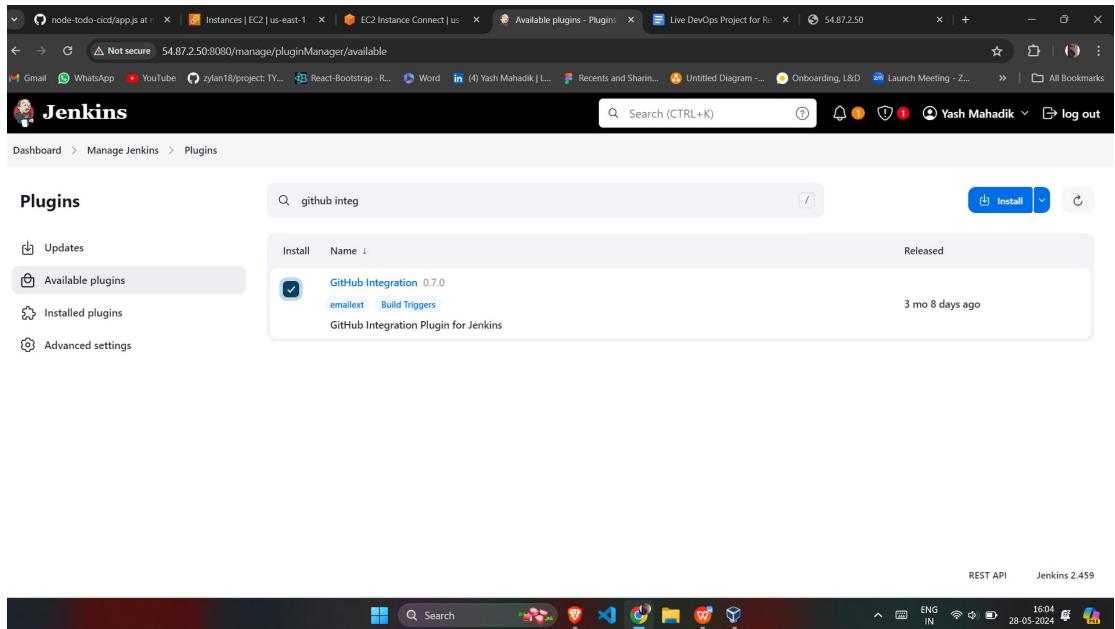
The screenshot shows a Jenkins console output for a build. The left sidebar includes links for Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Timings, Git Build Data, and Previous Build. The main area displays the build log:

```
Started by user Yash Mahadik  
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 9dc0516eb08da172400c4dbbb34464ca7194b (refs/remotes/origin/master)  
> git config core.sparsecheckout # timeout=10  
> git checkout -f 9dc0516eb08da172400c4dbbb34464ca7194b # timeout=10  
Commit message: "Merge pull request #192 from LondeShubham153/LondeShubham153-patch-24"  
> git rev-list --no-walk 9dc0516eb08da172400c4dbbb34464ca7194b # timeout=10
```

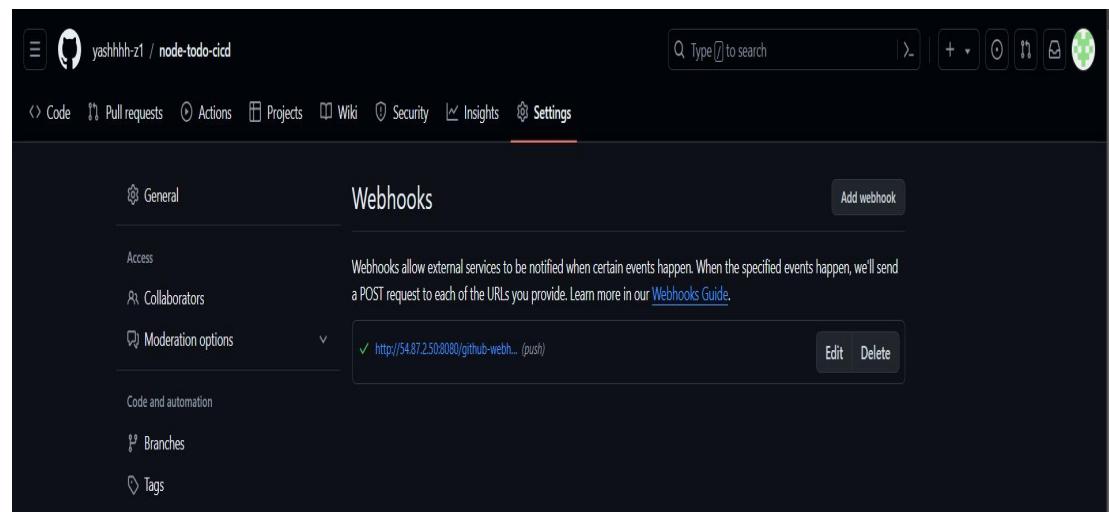
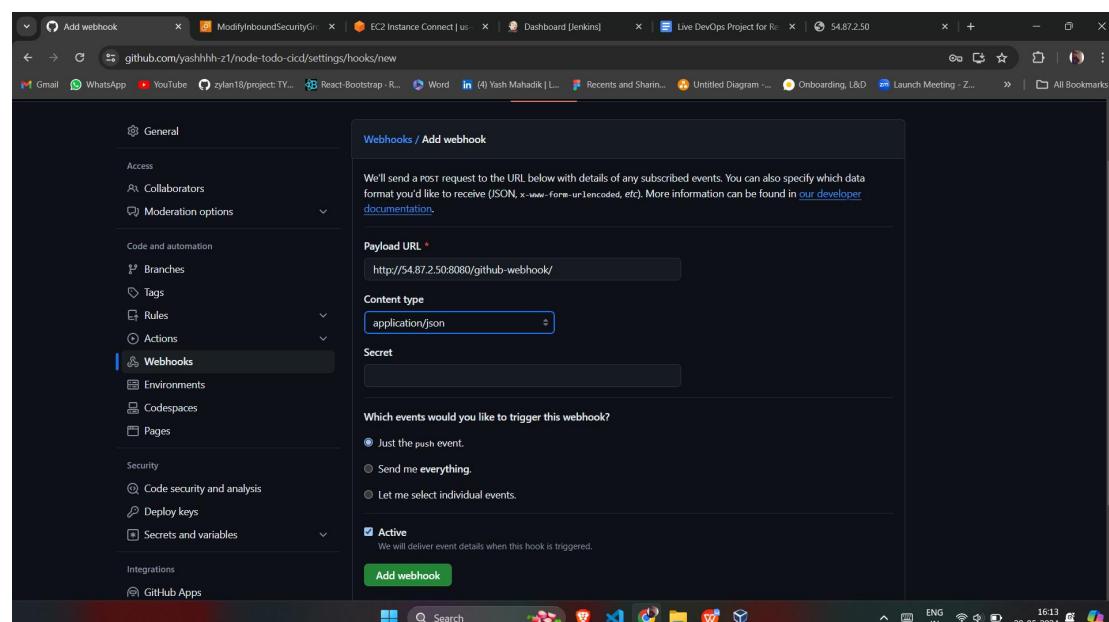
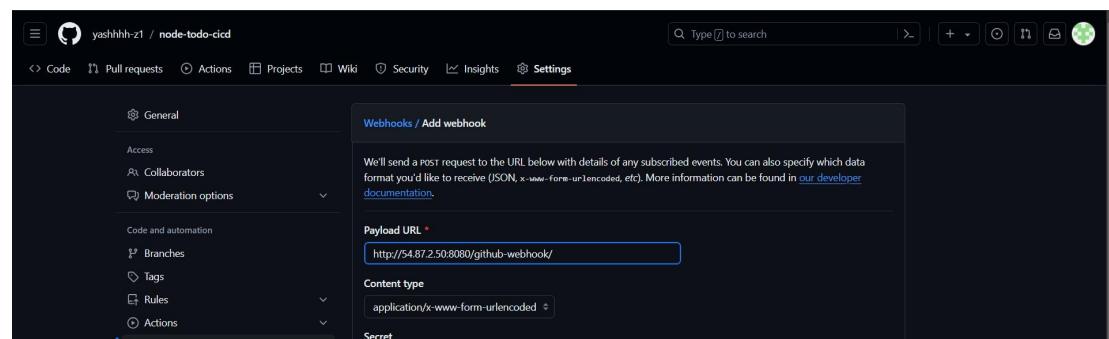
25) Our node.js app is successfully running.

The screenshot shows a browser window displaying a todo list application. The title bar says 'Todo List - Made for Batch 6'. The page content includes a header 'Riding', a search bar with placeholder 'What should I do?', and a green 'Add' button.

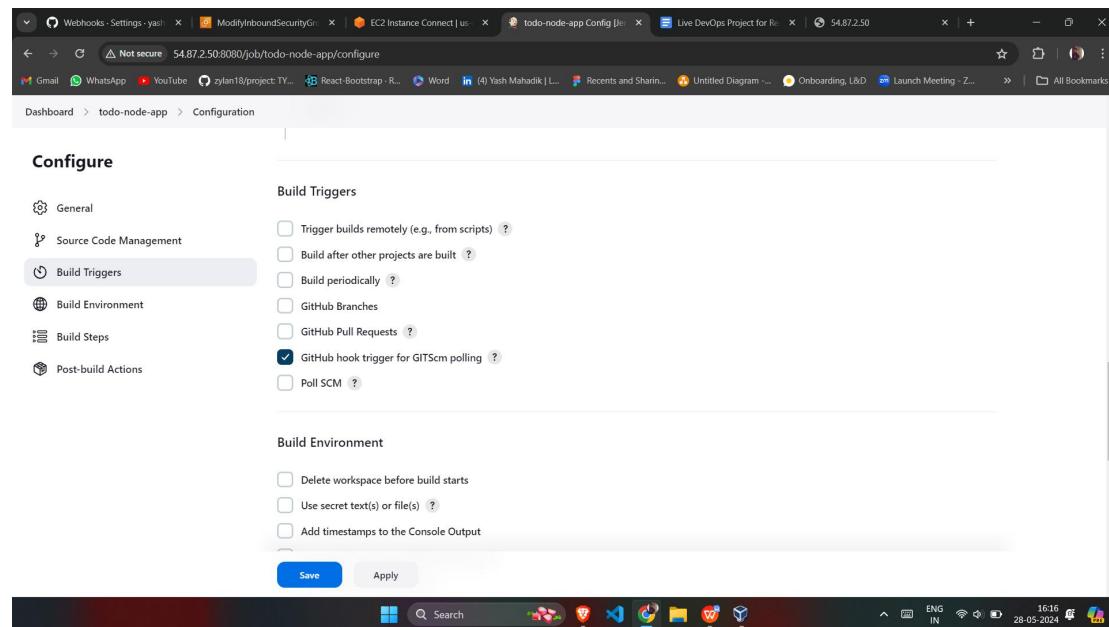
## 26) Lets automate with webhooks , First need to Install plugin in jenkins **Github Integration.**



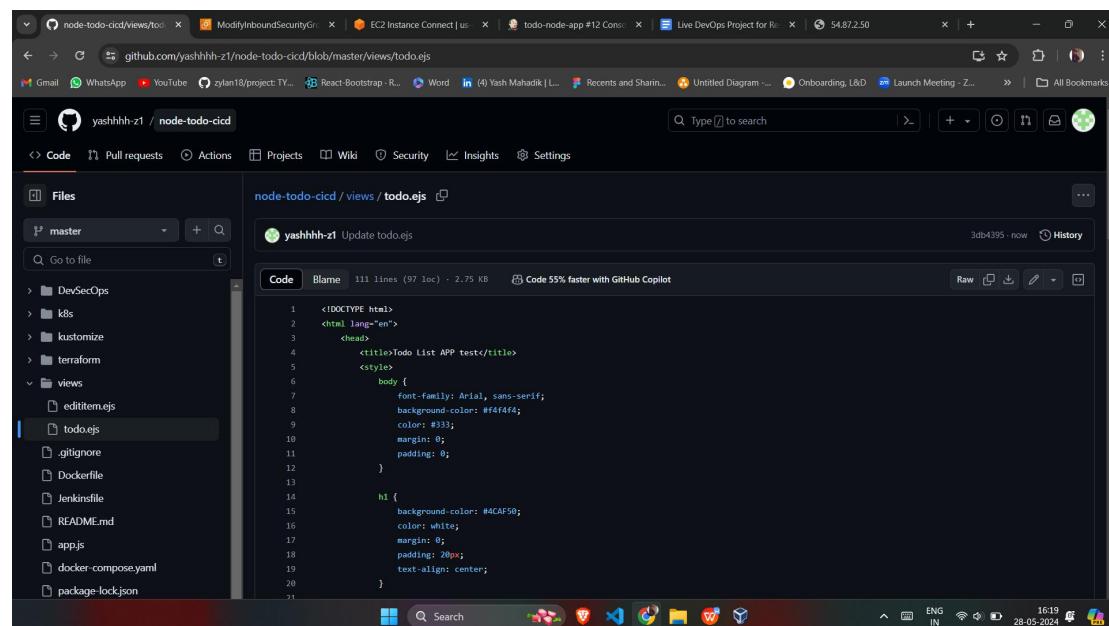
**27) Configure webhooks in github Go to Repository Setting > webhooks.**  
**Add webhook , In Payload URL, give the URL of jenkins with content type as json and add webhook.**



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.



## DevOps Project

