# Project: Constructing a comprehensive ChatGPT application via Jenkins pipeline

# Aim:

To create a Jenkins end-to-end declarative pipeline for a ChatGPT application which will operate with a single click to be production ready.

**Tools used:**

1. Docker
2. Dockerhub
3. Github
4. Git
5. Jenkins
6. Chatgpt application

**Steps to create the infrastructure in this pipeline :**

- Creating and launching ec2 instance with ami-Ubuntu.
- Installing git docker and related repos.
- Creating docker image with the help of yaml scripting

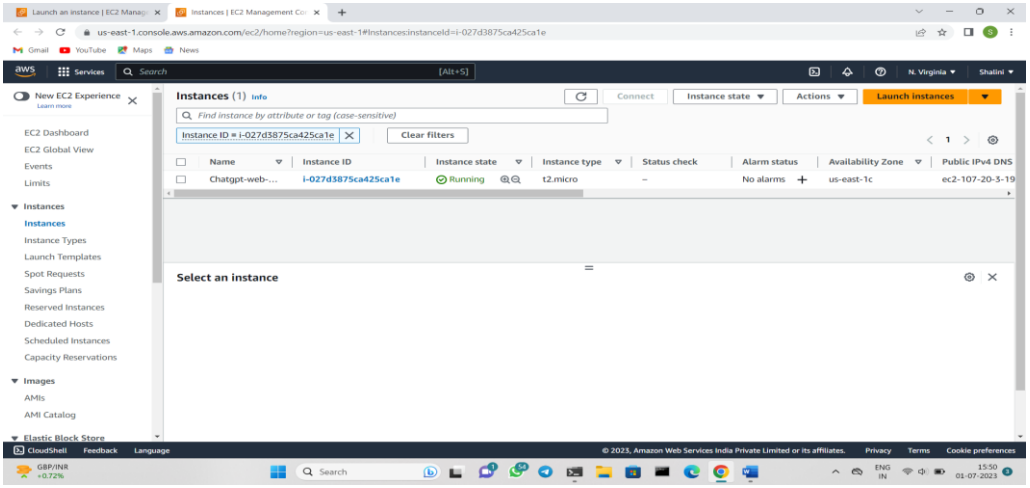
# Excution steps:

**Create an ec2 instance**

1. Launch an ec2 instance, specifying desired instance type, operating system, and other configuration options.

2. Configure security groups and network settings as per your requirements.

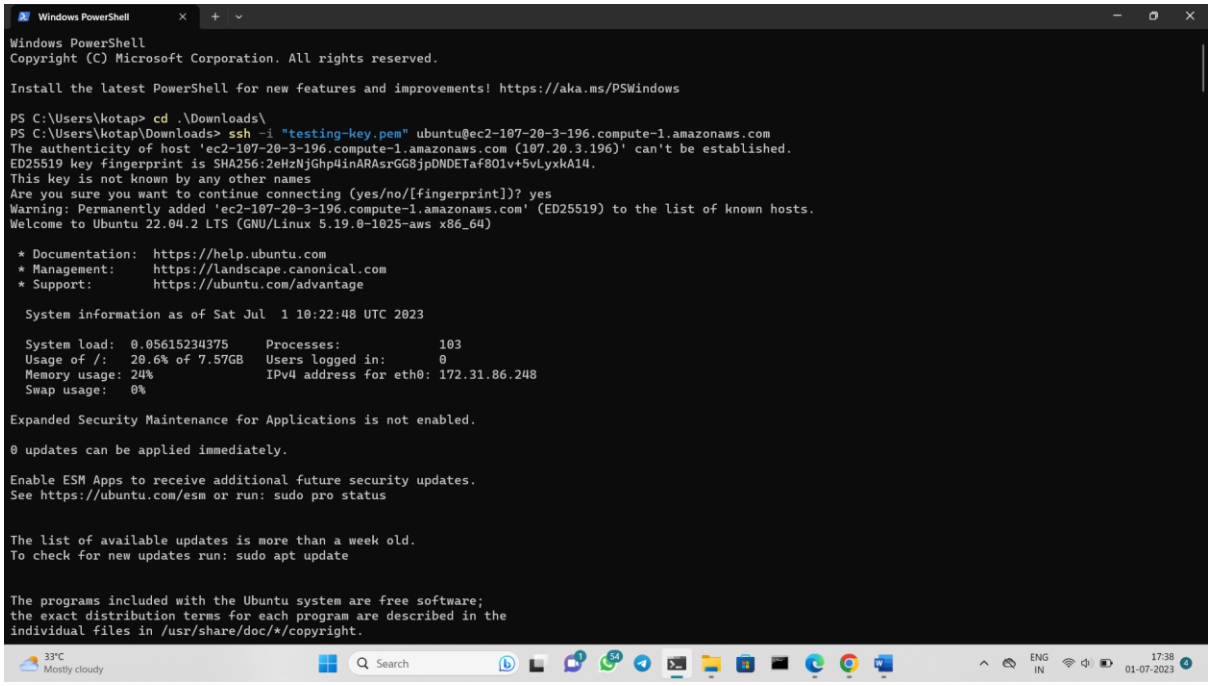
3. Review the instance details and launch the instance




4. Finally, Instance launched a Chatgpt-web-app instance

**To setup Chatgpt application:**

1. by using ssh command we are accessing the command line interface in terminal.




**Installing GIT, Docker, and related repos:**

we have to install the docker and docker-compose in our instance by using below commands

1. To install docker in ubuntu the command is
   - sudo apt install docker.io

2. To start the docker service the command
   - sudo systemctl start docker

**Install Docker Compose**

1. Download the latest version of Docker Compose (Incommand to download the current stable release of Docker Compose by using below command.

   - **sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose**

2. Apply excutable permissions to the binary.

   - **sudo usermod -a -G ec2-user**

3. Create a symbolic link.

   - **ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose**

4. to check the installed docker-compose verion.

   - **docker-compose –version**

## Configure Jenkins:

**Steps to Download and Install Jenkins:**

1. Run the following command to update all software packages on ec2 instance.
   - **Sudo apt update**
2. Add the Jenkins repo using the following command:

   - curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null

3. Import a key file from Jenkins-CI to enable installation from the package:

   - echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

```
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

4. Install java

```
➢ sudo apt install openjdk-11-jre
```

5. Install Jenkins:

```
➢ sudo apt-get install jenkins
```

6. Start Jenkins as a service:

➢ sudo systemctl start Jenkins

**Modify EC2 Security Group:**

1. Go to EC2 Dashboard.

2. Go to Security Groups and select the security group associated with EC2 instance.

3. Click Add Rule, and then choose Custom TCP Rule from the Type list. Under Port Range enter 8080.

**Configure Jenkins:**

1. Connect to http://:8080 from your browser. You will be able to access Jenkins through its management interface.

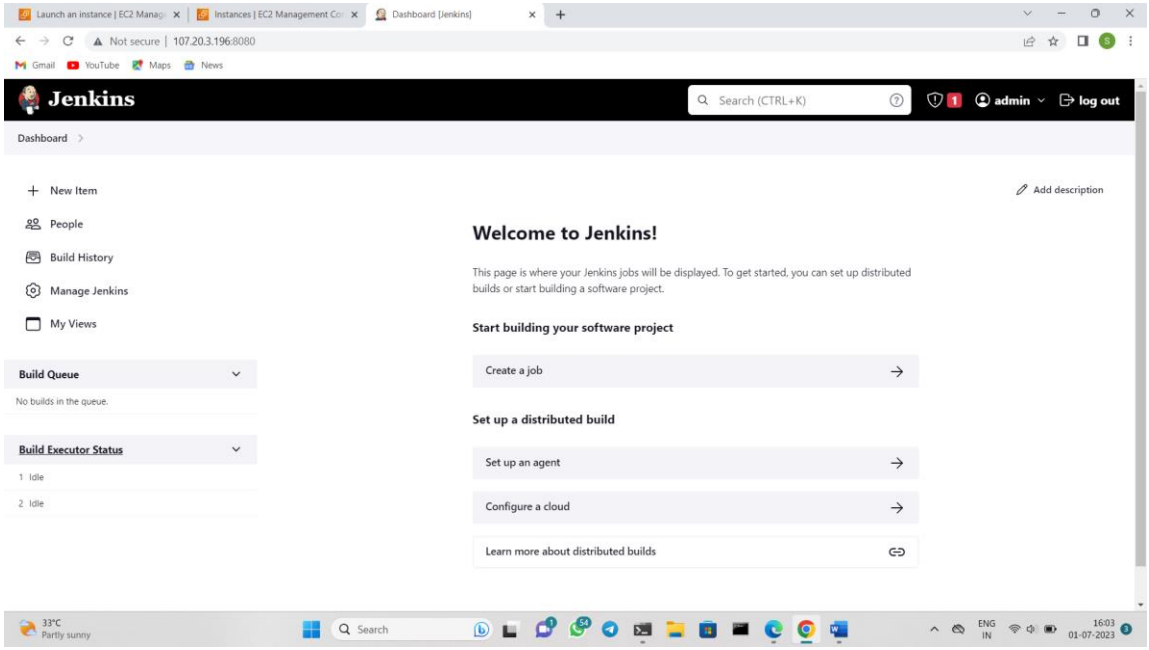
2. Enter the password found in /var/lib/jenkins/secrets/initialAdminPassword. Use the following command to display this password:
   ➢ **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

3. The Jenkins installation script directs you to the Customize Jenkins page. Click Install suggested plugins.

4. Create a new Admin User and complete the setup.

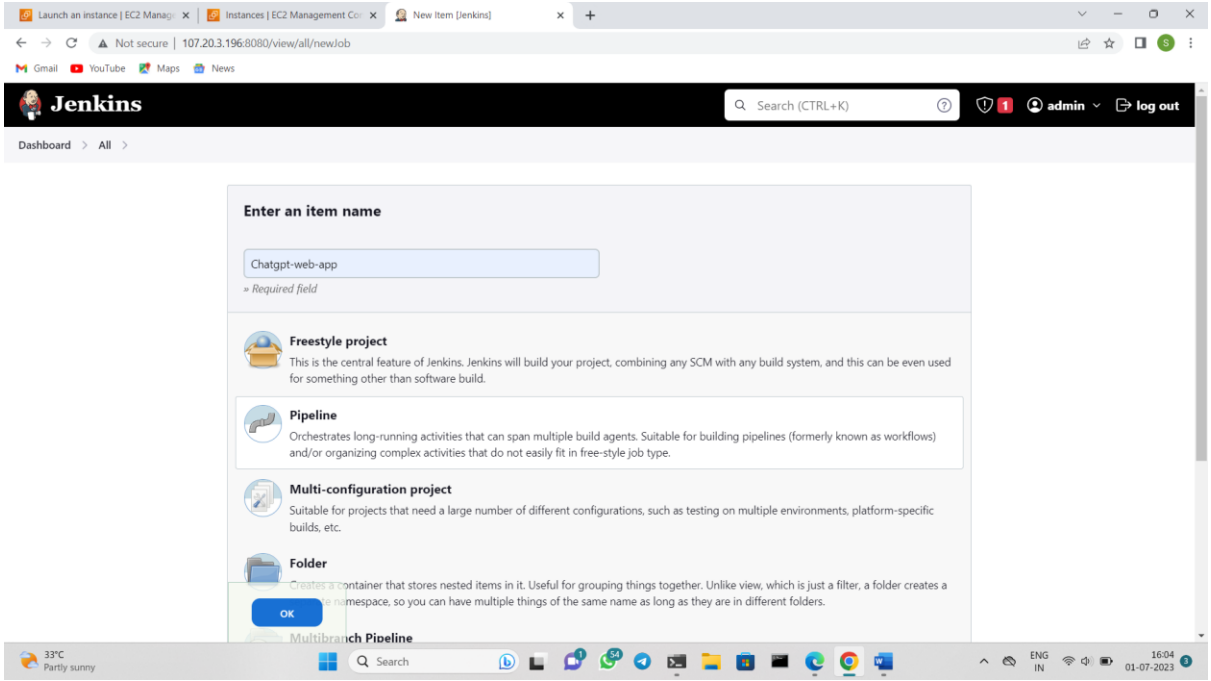
5. Click Start Using Jenkins. Jenkins Build Server is ready to be used on the AWS EC2 server.

6. Now you will be able to see the Jenkins Dashboard.




**Create a Pipeline in Jenkins by navigating through the new item.**

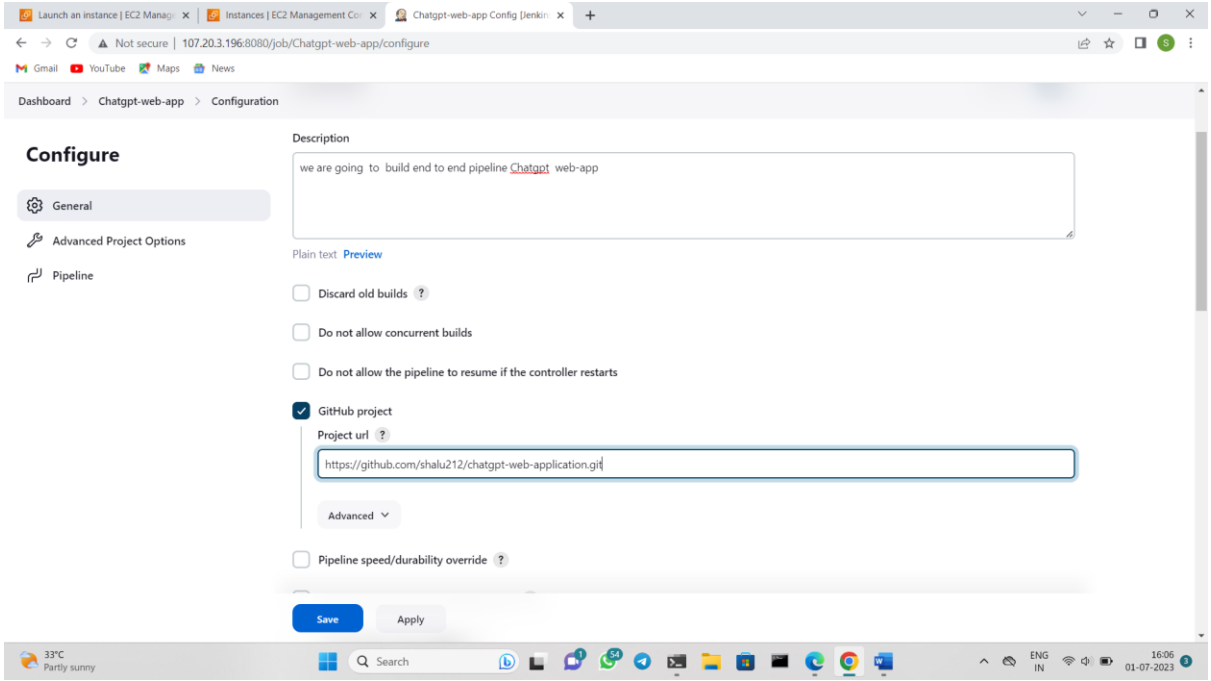
1. Click on new item
2. Enter job name
3. Select job type as pipeline
4. Click ok




5. Set up the configuration of the pipeline. Provide the necessary description for the pipeline.

6. Provide the Github project URL for a better decription as shown in the below screenshot.

7. Select the Github Hook trigger to setup the automatic trigger of pipeline through the github webhook property.

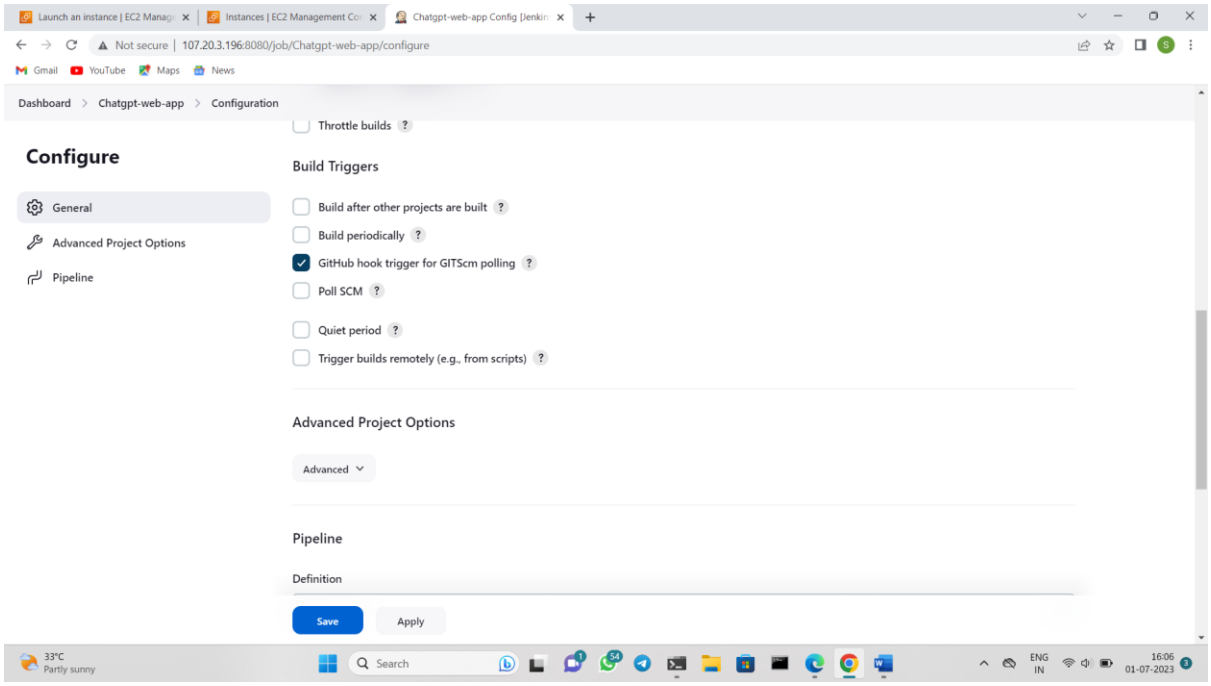



8. Navigate to the Github Project repository and go to the settings options and select the webhook option.

9. Provide the appropriate payload URL and settings to setup the webhook utility of github.

10. Now, On Jenkins lets set up the declarative pipeline. Here we will pick the pipeline script from Github without writing it in the Jenkins pipeline. This will have a huge advantage to keep track of the changes done by the users in a bigger it organizations.

11. Provide the necessary decription along with the project repository URL and secret token for connecting to github.




12.Add the Jenkinsfile to the pipeline and save the pipeline.

- Create a Jenkins in the Github repository, defining the pipeline stages, steps, and configurations.
- Include the necessary Docker-related steps, such as building the Docker image and pushing it to the DockerHub.
- Configure the pipeline to handle secrets and credentials , such as OpenAI credentials and DockerHUB credentials

https://github.com/shalu212/chatgpt-web-application/blob/master/Jenkinsfile

shalu212 Update Jenkinsfile 54a1bcf · 4 hours ago History

Code Blame 35 lines (34 loc) · 1.14 KB

```
pipeline {
    agent any
     environment {
                    key = credentials('openAI')
                }
    stages {
        stage('Code fetch') {
            steps {
                git url:'https://github.com/Bandank/chatgpt-web-application.git',branch:'master'
            }
        }
        stage('build and test') {
            steps {
                sh 'docker build . -t iamadminallowme/chatgpt-app:latest'
            }
        }
        stage('push') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'dockerHub', passwordVariable: 'dockerHubPassword', usernameVariable: 'dockerHubUser')]) {
                sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPassword}"
                sh 'docker push iamadminallowme/chatgpt-app:latest'
            }
          }
        }
        stage('Deploy'){
            steps {
                withCredentials([string(credentialsId: 'openAI', variable: 'OPENAI_API_KEY')]) {
                            sh 'docker-compose down'
                            sh "docker-compose up -d"
                        }
```

Not secure | 107.20.3.196:8080/job/Chatgpt-web-app/configure

Dashboard > Chatgpt-web-app > Configuration

**Configure**

General

Advanced Project Options

Pipeline

**Pipeline**

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/shalu212/chatgpt-web-application.git

Please enter Git repository.

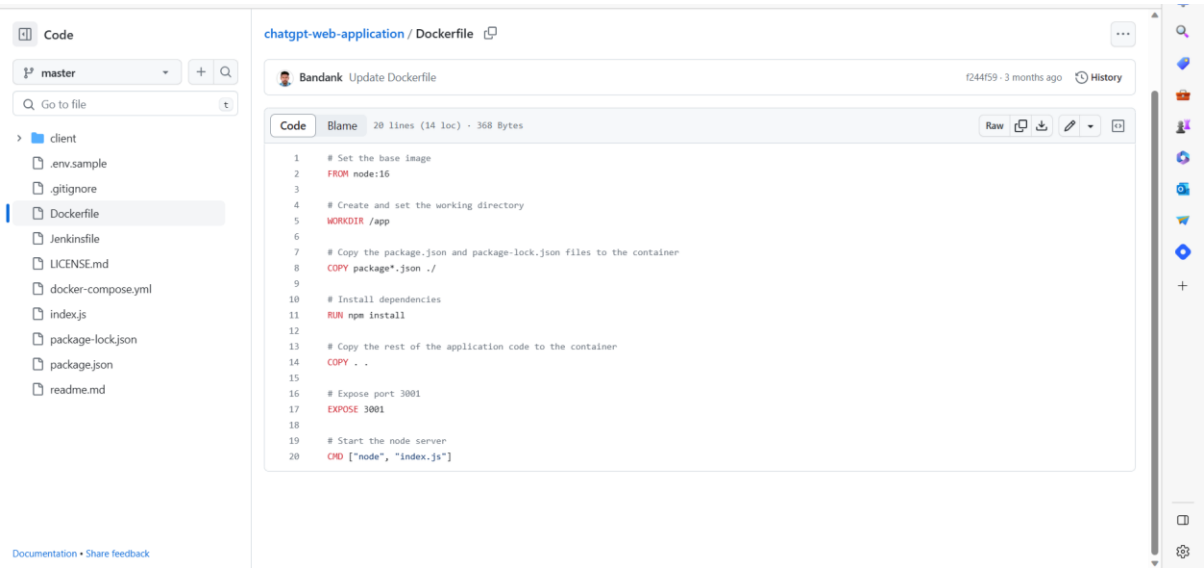
Credentials: - none -

Add

Save Apply

SHALINI.K

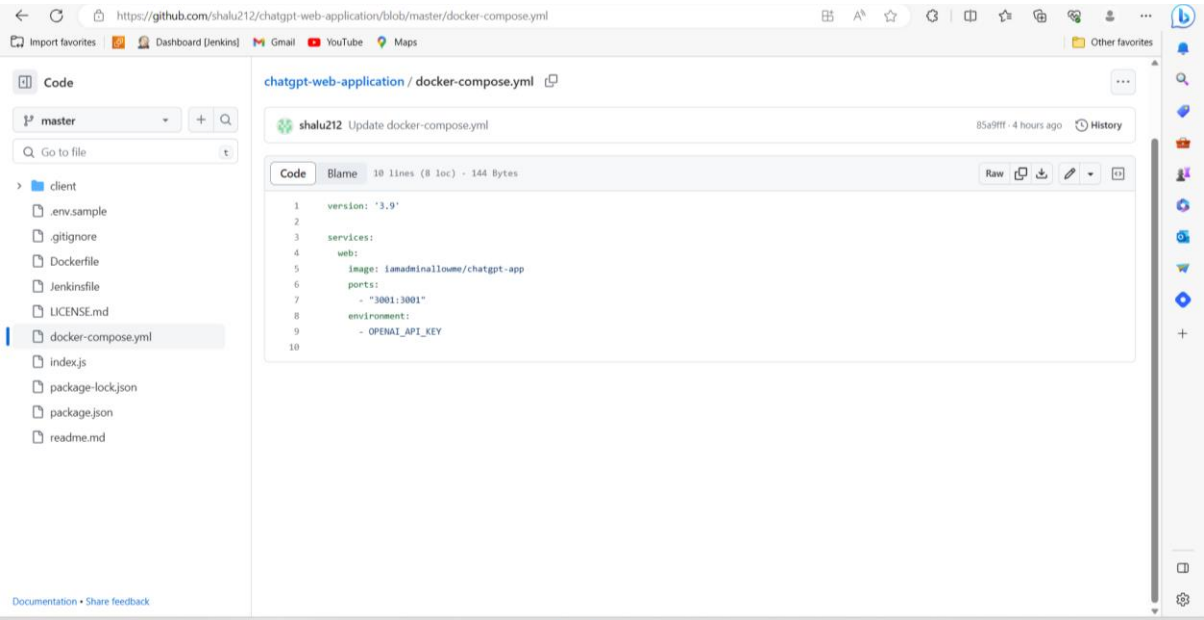
Lets create the Dockerfile for the application.



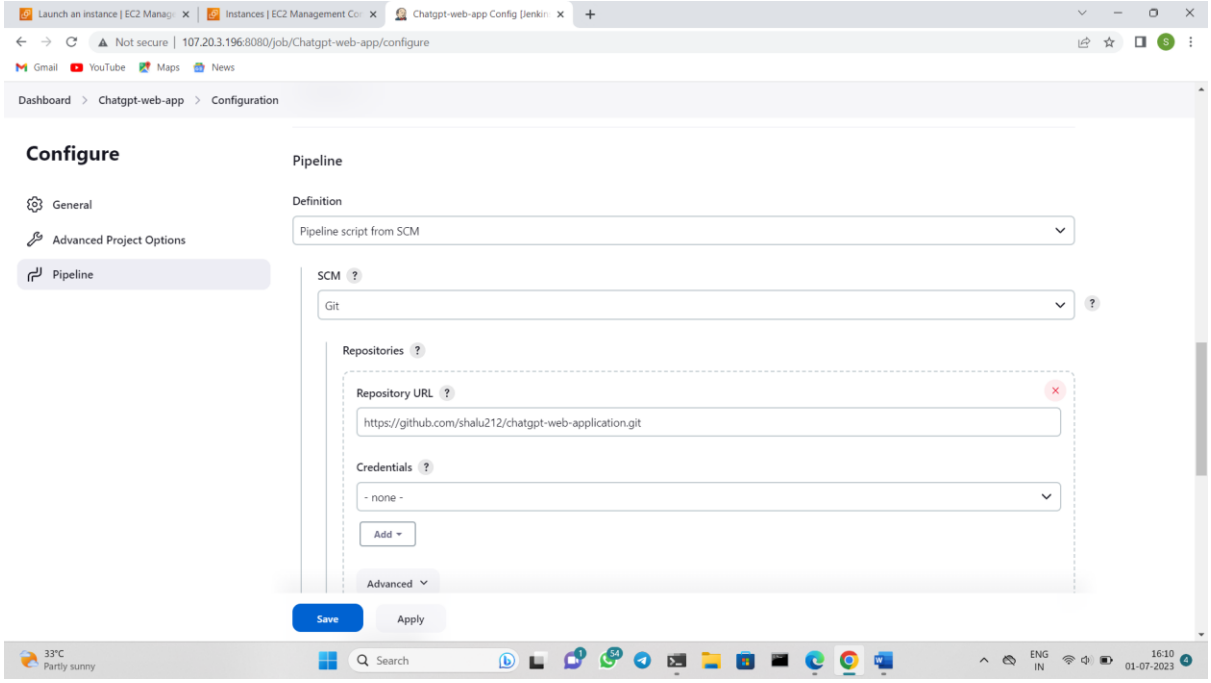

1. Create a docker-compose.yml file.

   - Define the file for your Chatgpt application, specifying the necessary dependencies and configurations.

   - Create a Docker-compose.yml file to configure the docker containers and their interconnections.

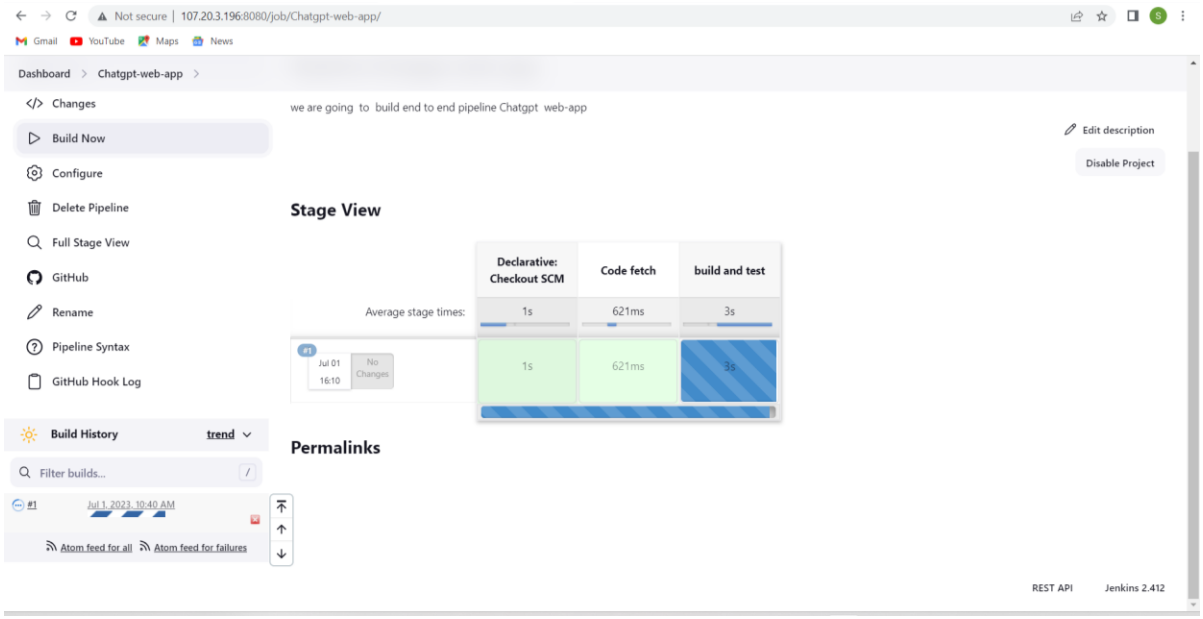


2. The execution steps in the Jenkins file will include the secret key to connect the OPENAI credentials of ChatGPT. Also, for pushing the image to the docker hub, set up the username and password.

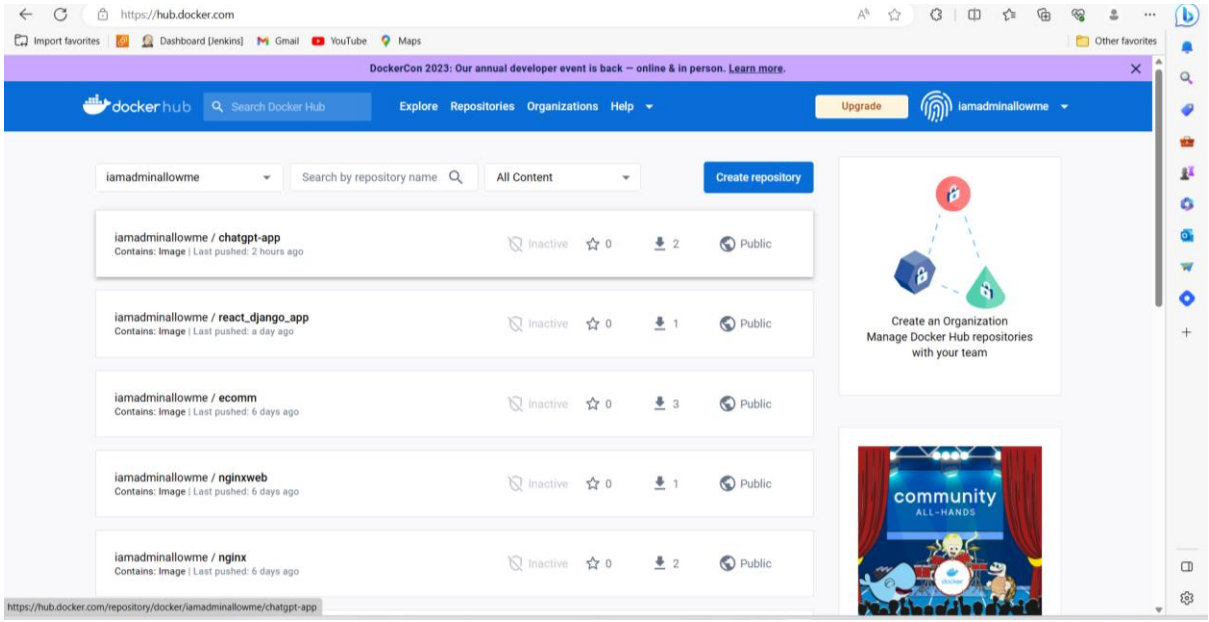



3. Save the job and click on build now to the build the job

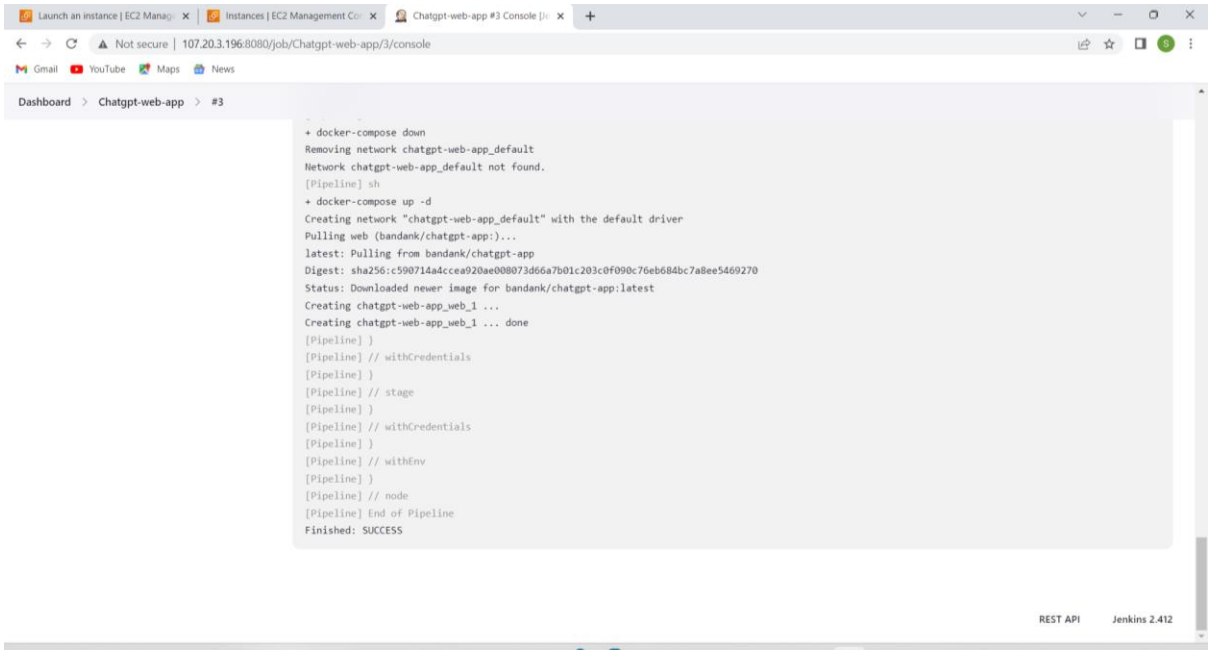
- Save the pipeline job configuration in Jenkins
- Click on “Build now” to trigger the pipeline execution.




You can check the images are pushed to the DockerHub




- Monitor the build in console output for any errors or issues.




## Project live execution

We have seen the execution of the pipeline below. Now, navigate to the agent server URL and observe the running application.