

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 :

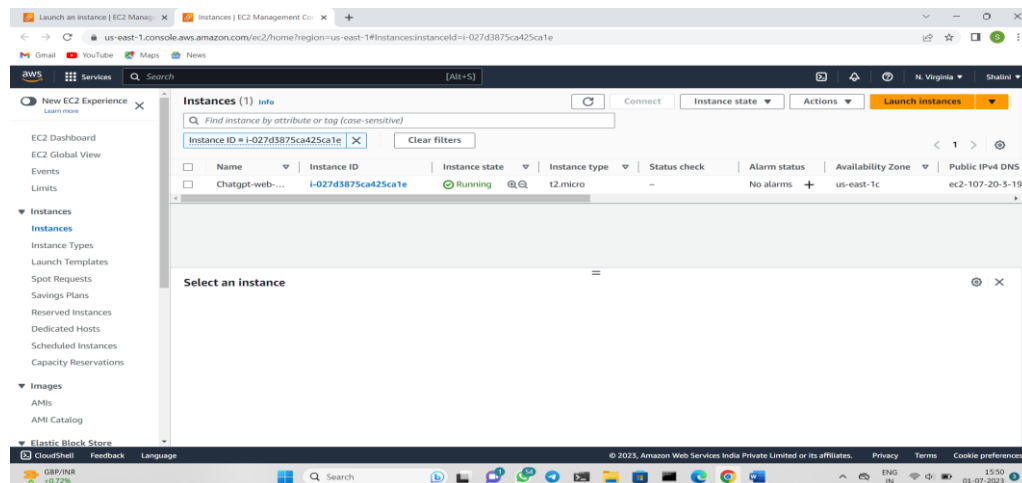
- Creating and launching ec2 instance with ami-Ubuntu.
- Installing git docker and related repos.
- Creating docker image with the help of yml scripting

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.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\kotap> cd .\Downloads\
PS C:\Users\kotap\Downloads> ssh -i "testing-key.pem" ubuntu@ec2-107-20-3-196.compute-1.amazonaws.com
The authenticity of host 'ec2-107-20-3-196.compute-1.amazonaws.com (107.20.3.196)' can't be established.
ED25519 key fingerprint is SHA256:2eHzNjGhp4inARAsrGG8jpdNDEtaf801v+5vLyxkA14.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-107-20-3-196.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Jul  1 10:22:48 UTC 2023

System load:  0.05615234375   Processes:    103
Usage of /:   20.6% of 7.57GB   Users logged in:  0
Memory usage: 24%            IPv4 address for eth0: 172.31.86.248
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

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 chmod +x /usr/local/bin/docker-compose`

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:

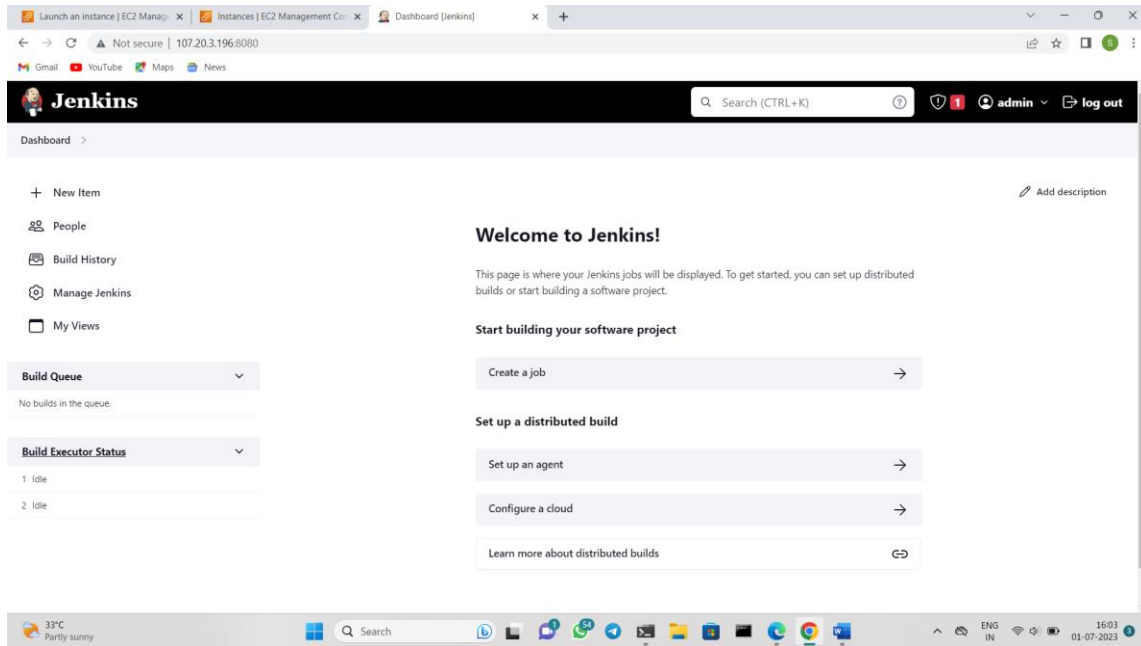
1. Go to EC2 Dashboard.
2. Go to Security Groups and select the security group associated with EC2 instance.
3. Click Add Rule, and then choose Custom TCP Rule from the Type list. Under Port Range enter 8080.

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.

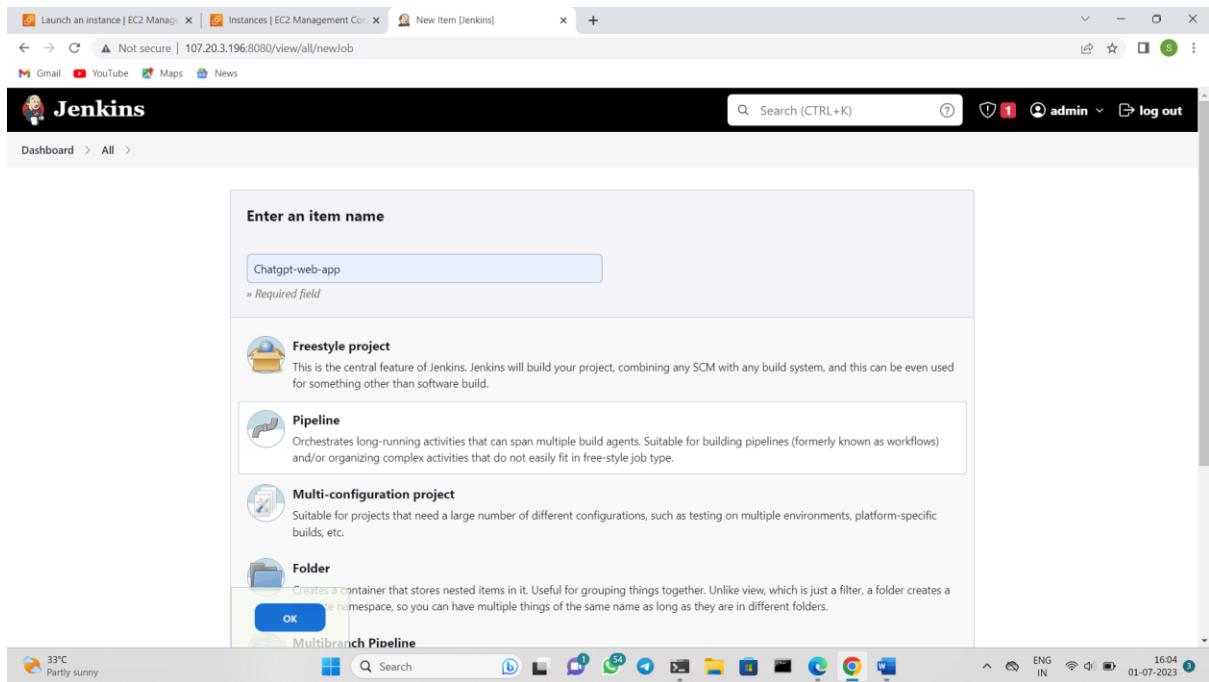
5. Click Start Using Jenkins. Jenkins Build Server is ready to be used on the AWS EC2 server.

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

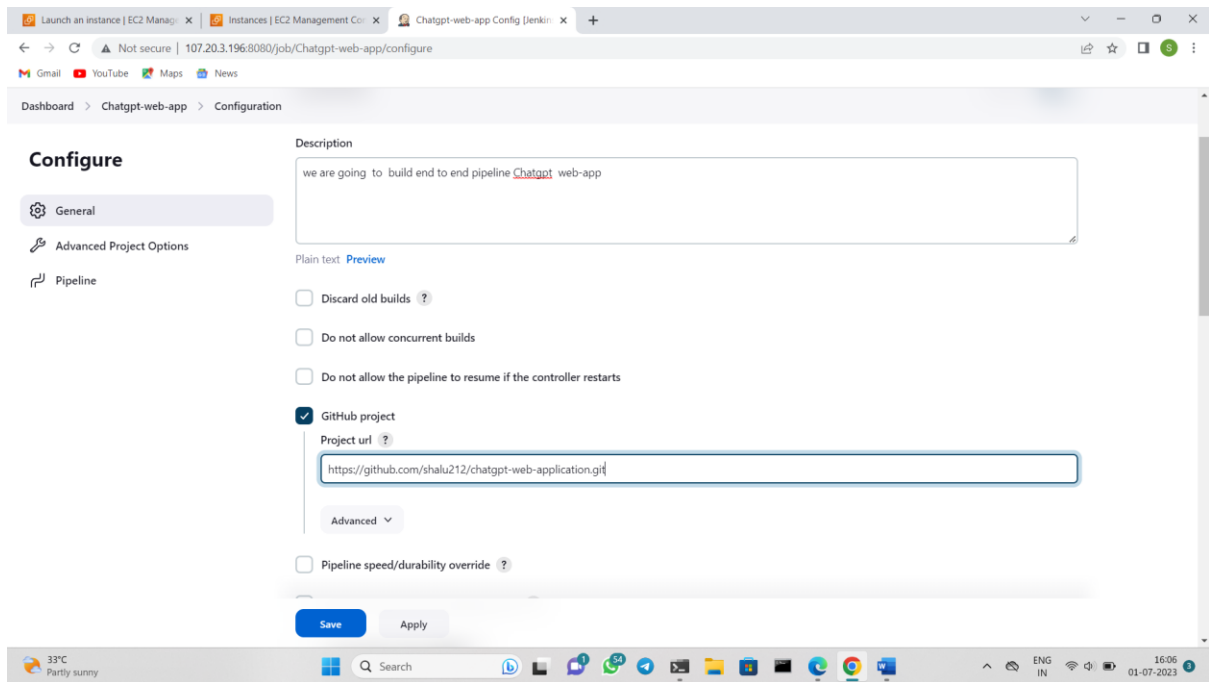


Create a Pipeline in Jenkins by navigating through the new item.

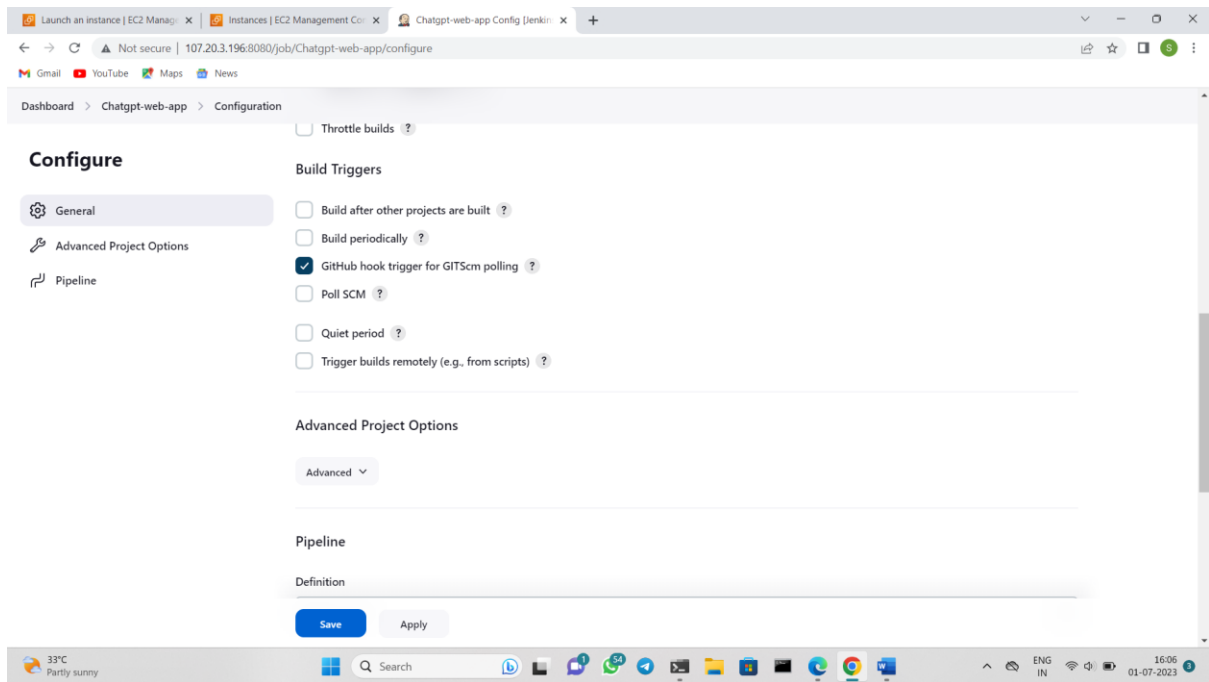
1. Click on new item
2. Enter job name
3. Select job type as pipeline
4. Click ok



5. Set up the configuration of the pipeline. Provide the necessary description for the pipeline.
6. Provide the Github project URL for a better description 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 Jenkinsfile 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

The screenshot shows a GitHub repository page for 'shalu212/chatgpt-web-application'. The 'Jenkinsfile' is selected in the file explorer on the left. The main content area displays the Jenkinsfile code, which is a YAML script for a Jenkins pipeline. The pipeline includes stages for 'Code fetch', 'build and test', and 'Deploy'. The 'Code fetch' stage uses 'git' to fetch the code from the repository. The 'build and test' stage uses 'docker build' to build the application. The 'Deploy' stage uses 'docker login' and 'docker push' to push the application to Docker Hub. The pipeline is configured to run on a 'linux/amd64' agent.

```
1 pipeline {
2   agent any
3   environment {
4     key = credentials('openAI')
5   }
6   stages {
7     stage('Code fetch') {
8       steps {
9         git url: 'https://github.com/Shandank/chatgpt-web-application.git', branch: 'master'
10      }
11    }
12    stage('build and test') {
13      steps {
14        sh 'docker build -t iamadinalloome/chatgpt-app:latest'
15      }
16    }
17    stage('push') {
18      steps {
19        withCredentials([usernamePassword(credentialsId: 'dockerHub', passwordVariable: 'dockerHubPassword', usernameVariable: 'dockerHubUser')]) {
20          sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPassword}"
21          sh "docker push iamadinalloome/chatgpt-app:latest"
22        }
23      }
24    }
25    stage('Deploy'){
26      steps {
27        withCredentials([string(credentialsId: 'openAI', variable: 'OPENAI_API_KEY')]) {
28          sh "docker-compose down"
29          sh "docker-compose up -d"
30        }
31      }
32    }
33  }
34 }
```

The screenshot shows the Jenkins 'Configure' page for the 'Chatgpt-web-app' job. The 'Pipeline' tab is selected in the left sidebar. The 'Definition' section shows 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git'. The 'Repositories' section shows a single repository with the URL 'https://github.com/shalu212/chatgpt-web-application.git'. The 'Credentials' dropdown is set to '- none -'. The 'Save' button is highlighted in blue.

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.

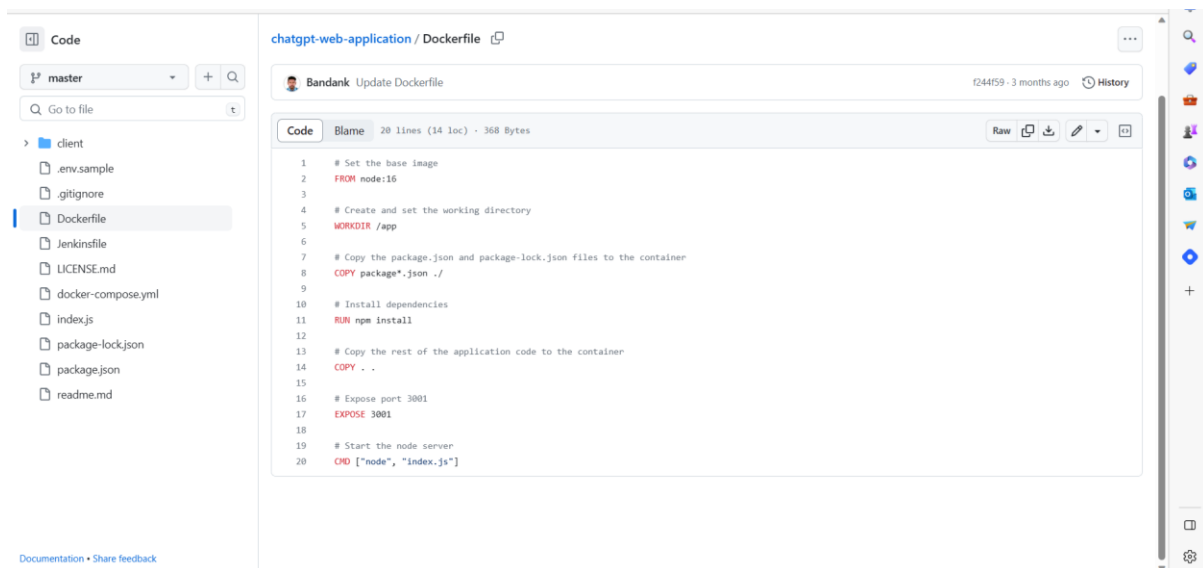
Credentials ?

- none -

Add

Save Apply

Lets create the Dockerfile for the application.

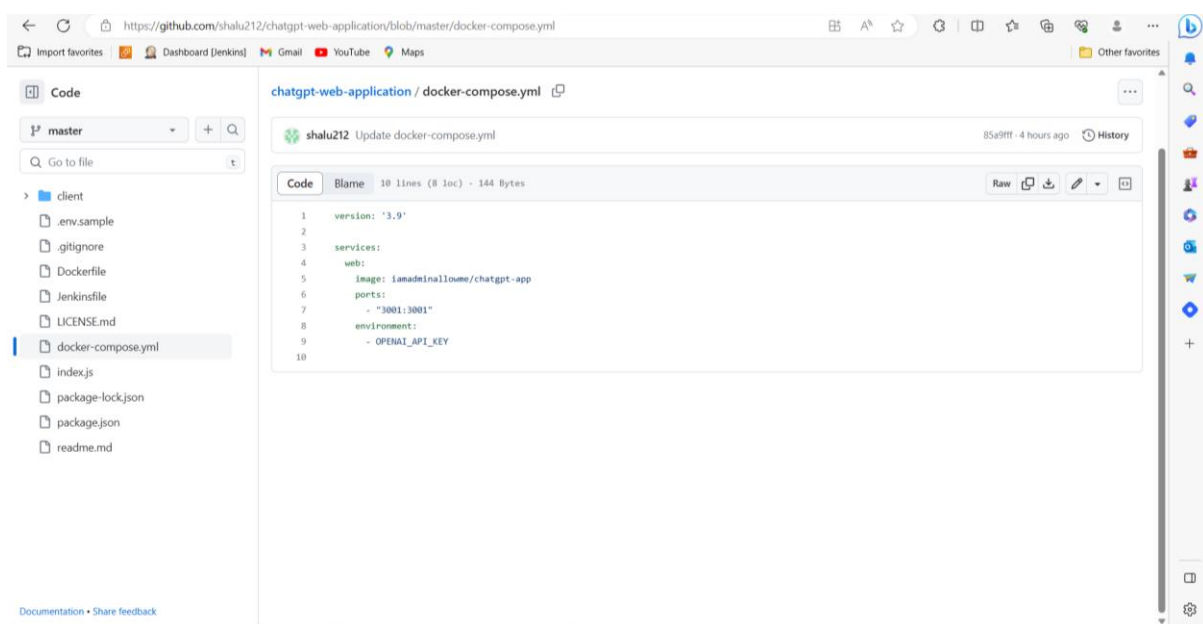


The screenshot shows a GitHub repository named 'chatgpt-web-application' with the 'Dockerfile' selected. The file is 20 lines long, 14 loc, and 368 Bytes. The code is as follows:

```
1 # Set the base image
2 FROM node:16
3
4 # Create and set the working directory
5 WORKDIR /app
6
7 # Copy the package.json and package-lock.json files to the container
8 COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the application code to the container
14 COPY . .
15
16 # Expose port 3001
17 EXPOSE 3001
18
19 # Start the node server
20 CMD ["node", "index.js"]
```

1. Create a docker-compose.yml file.

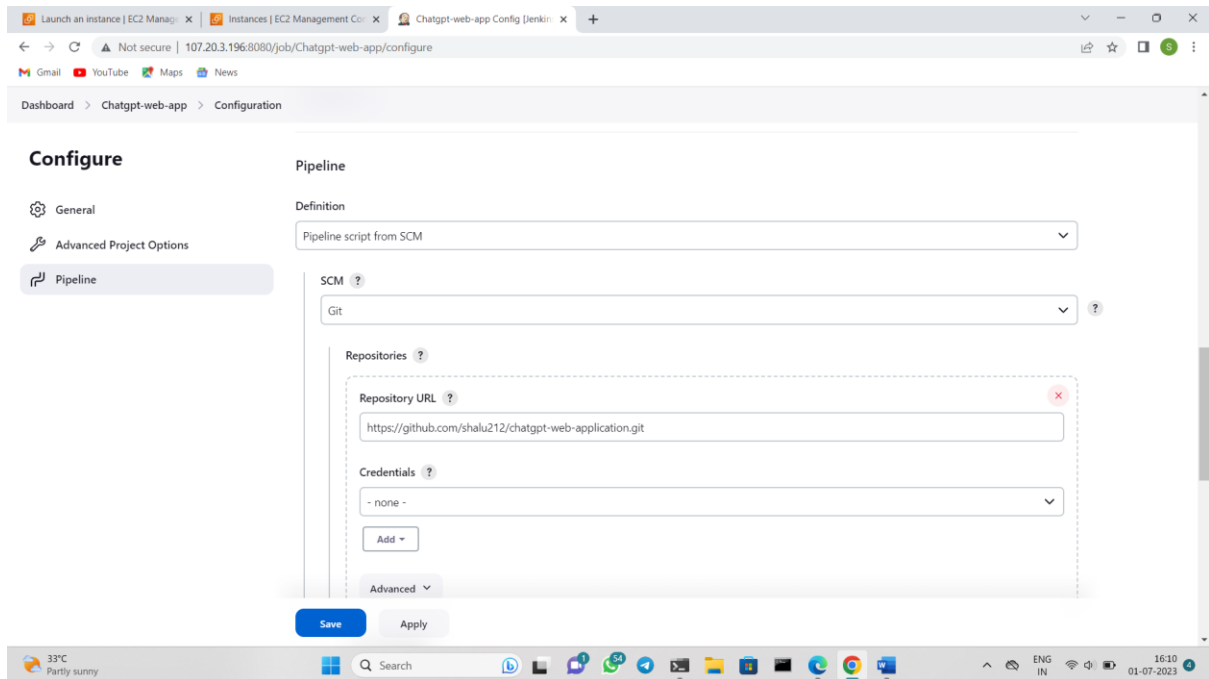
- Define the file for your Chatgpt application, specifying the necessary dependencies and configurations.
- Create a Docker-compose.yml file to configure the docker containers and their interconnections.



The screenshot shows a GitHub repository named 'chatgpt-web-application' with the 'docker-compose.yml' selected. The file is 10 lines long, 8 loc, and 144 Bytes. The code is as follows:

```
1 version: '3.9'
2
3 services:
4   web:
5     image: imadainalloume/chatgpt-app
6     ports:
7       - "3001:3001"
8     environment:
9       - OPENAI_API_KEY
```

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.



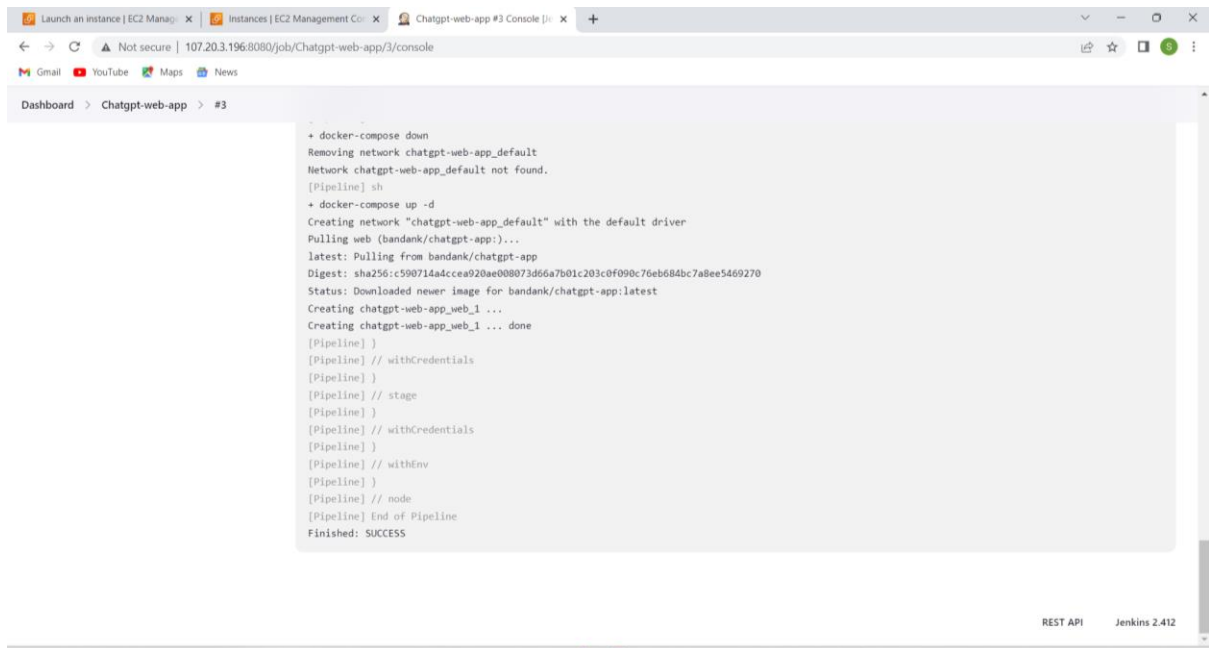
3. Save the job and click on build now to the build the job
 - Save the pipeline job configuration in Jenkins
 - Click on “Build now” to trigger the pipeline execution.

The screenshot shows the Jenkins dashboard for a project named 'Chatgpt-web-app'. The interface includes a left sidebar with navigation options: Changes, Build Now, Configure, Delete Pipeline, Full Stage View, GitHub, Rename, Pipeline Syntax, and GitHub Hook Log. The main area displays the 'Stage View' of a pipeline. The pipeline consists of three stages: 'Declarative: Checkout SCM' (1s), 'Code fetch' (621ms), and 'build and test' (3s). Below the stage view, there is a 'Permalinks' section and a 'Build History' section showing a list of builds. The bottom right corner indicates the REST API and Jenkins version 2.412.

You can check the images are pushed to the DockerHub

The screenshot shows the DockerHub repository page for the user 'iamadmnallowme'. The page displays a list of repositories: 'chatgpt-app', 'react_django_app', 'ecommm', 'nginxweb', and 'nginx'. Each repository entry shows the number of images, the last pushed time, and the repository status (Inactive). The 'chatgpt-app' repository is highlighted, showing it contains 2 images and was last pushed 2 hours ago. The page also features a 'Create repository' button and a 'community ALL-HANDS' banner.

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



```
+ docker-compose down
Removing network chatgpt-web-app_default
Network chatgpt-web-app_default not found.
[Pipeline] sh
+ docker-compose up -d
Creating network "chatgpt-web-app_default" with the default driver
Pulling web (bandank/chatgpt-app)...
latest: Pulling from bandank/chatgpt-app
Digest: sha256:c590714a4ccea920ae008073d66a7b01c203c0f090c76eb684bc7a8ee5469270
Status: Downloaded newer image for bandank/chatgpt-app:latest
Creating chatgpt-web-app_web_1 ...
Creating chatgpt-web-app_web_1 ... done
[Pipeline]
[Pipeline] // withCredentials
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withCredentials
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Project live execution

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

