

TECHNOLOGY

Computing

Caltech

**Center for Technology &
Management Education**

Jenkins and Docker Angular Web Apps

TECHNOLOGY

Computing

Caltech

**Center for Technology &
Management Education**

Set Up Jenkins

You Already Know

Before we begin, let's recall what we have covered till now:



Agile



Git



SQL



Angular



HTML



CSS



JavaScript



Core java

You Already Know

Before we begin, let's recall what we have covered till now:



JDBC

JDBC



JSP

JSP



Servlets

Servlets



mongoDB®

MongoDB

Maven™

Maven

You Already Know

Before we begin, let's recall what we have covered till now:

JUnit

JUnit



Spring



Spring Boot



Webservices

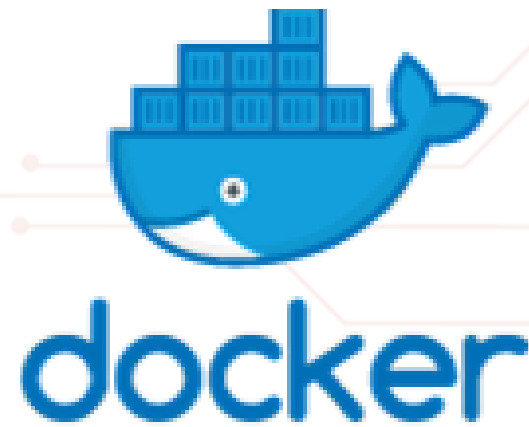


Microservices

Microservices

You Already Know

Before we begin, let's recall what we have covered till now:



Docker



Jenkins



AWS

Developed Angular Front End

- Created various components and services for the Angular eStore admin and end user.

Developed Java Back End

- Developed the Java back end for the admin and the end-user
- Implemented microservices with Spring Boot

Front-End Back-End Communication

- Used HTTP Client in Angular to communicate with the Java back end
- Implemented communication for both the admin and the end-user projects



A Day in the Life of a Full Stack Developer

As a full stack web developer, our key role is to develop both client and server software.



Angular and Node can be used to build the front end of the web page.



Spring Boot, Java, and MySQL or MongoDB can be used to build at the back end.



A Day in the Life of a Full Stack Developer

Bob needs to develop Angular web apps for the admin and the end user.
He brainstorms a bit and finds a solution.

Let me use Jenkins, Docker, and AWS to build CI/CD Pipelines, containerize the apps, and finally host them to AWS EC2 instance.



In this lesson, we will create Jenkins Pipeline for CI/CD for Java back end. Moving ahead, we will create a Dockerfile to build images, run them as containers in Docker, and help Bob complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

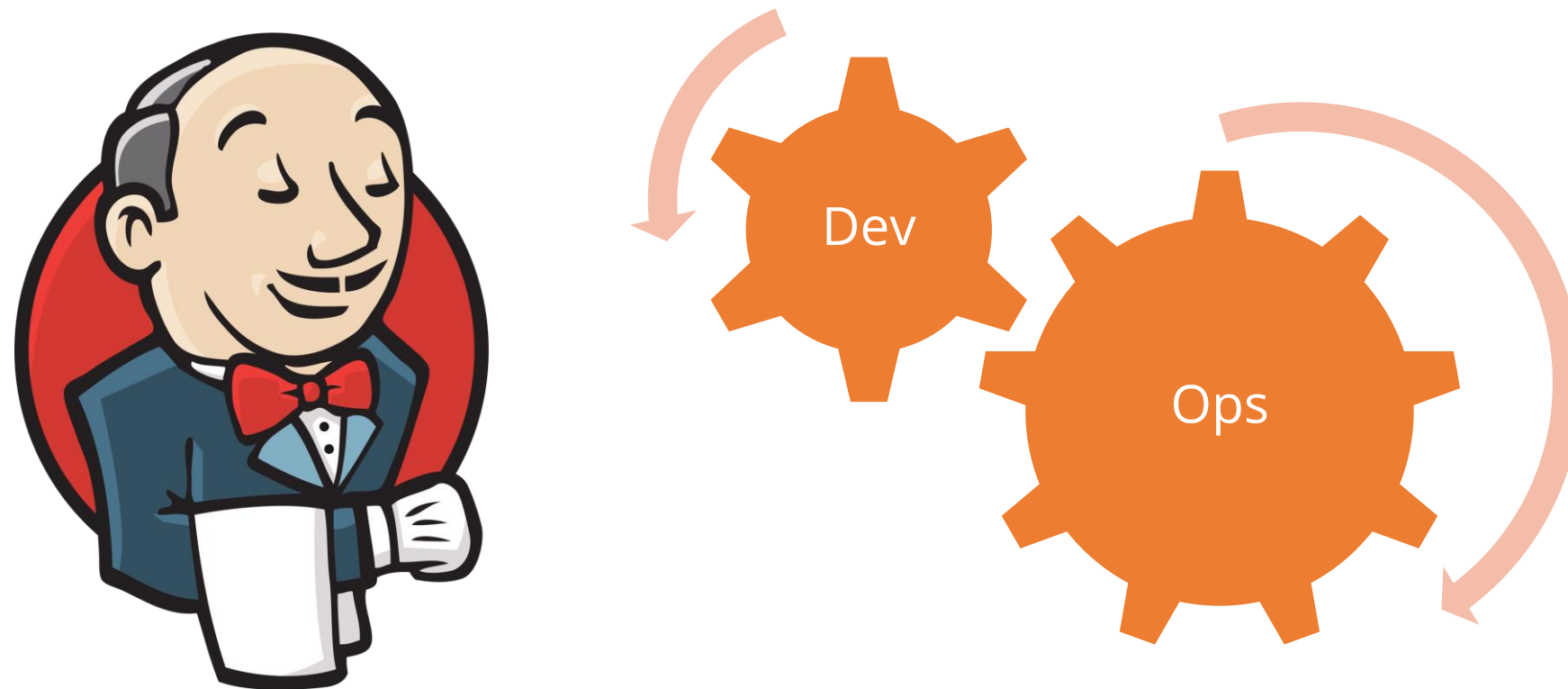
- 🕒 Create Jenkins Pipeline
- 🕒 Implement CI/CD in Jenkins with Git
- 🕒 Create a Dockerfile for building images
- 🕒 Integrate Docker in Jenkins to build and release the images as containers



Set Up Jenkins for the Angular Web-Admin Project

Jenkins

Jenkins is an automation build server used for tasks such as; building, testing, and deploying software.



Jenkins plays an important role in developing continuous integration and continuous delivery pipelines.

Setting Up Jenkins Environment

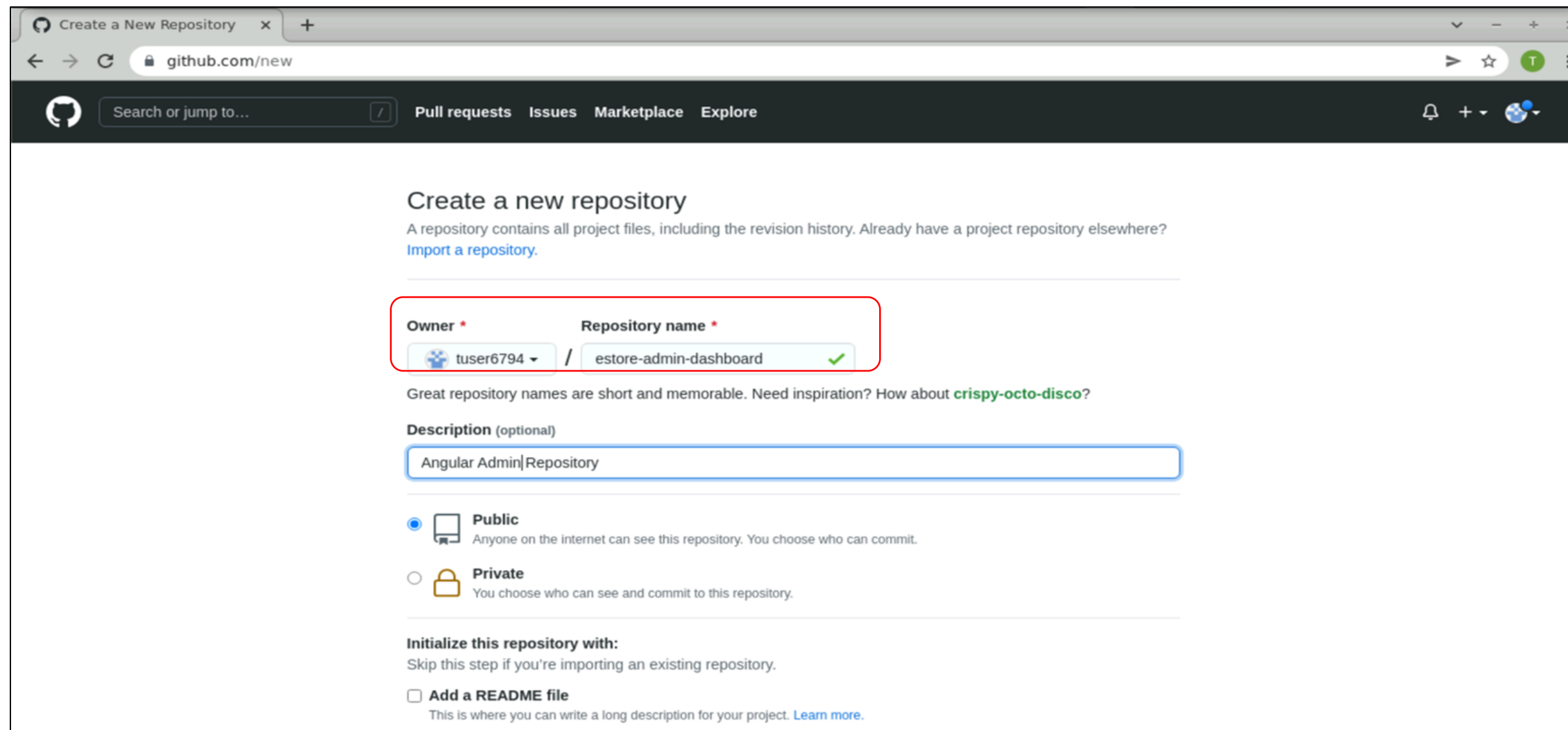
Jenkins environment is needed to create the NodeJS project using Jenkinsfile for the build.



The image shows the Jenkins web interface. At the top, there is a black header bar with the Jenkins logo (a cartoon man) and the word "Jenkins" in white. To the right of the logo is a red square with the number "1". Further right is a search bar with the word "search" and a magnifying glass icon. Below the header bar is a light green bar with the text "Jenkins" and "All" separated by a right-pointing arrow. The main content area is a light gray box. Inside this box, there is a section titled "Enter an item name" in bold black text. Below this title is a white text input field. Underneath the input field, there is a small gray text label that says "Required field". Below the input field is a large white rectangular area with a blue border. On the left side of this area is a small icon of a blue sphere on a yellow box. At the bottom left of the gray box, there is a light green rounded rectangle containing a blue button with the text "OK" in white.

Create Git Repository on GitHub for the Web-Admin Project

Create a new repository on your GitHub account which will be used by Jenkins to sync the code in SCM.



Create a New Repository x +

github.com/new

Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * **Repository name ***

tuser6794 / estore-admin-dashboard ✓

Great repository names are short and memorable. Need inspiration? How about [crispy-octo-disco?](#)

Description (optional)

Angular Admin|Repository

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

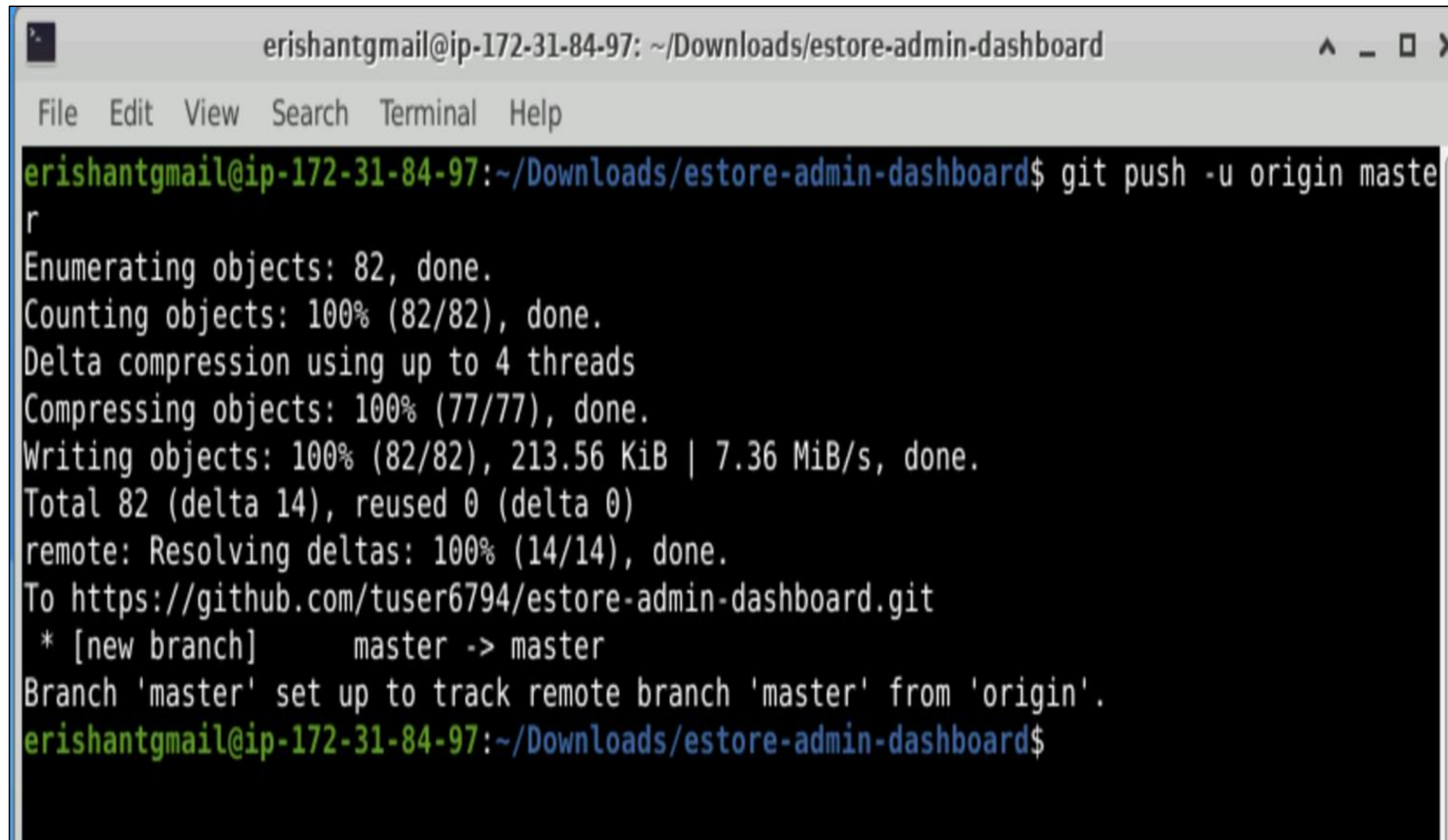
☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Sync the Project on GitHub for the Web-Admin Project

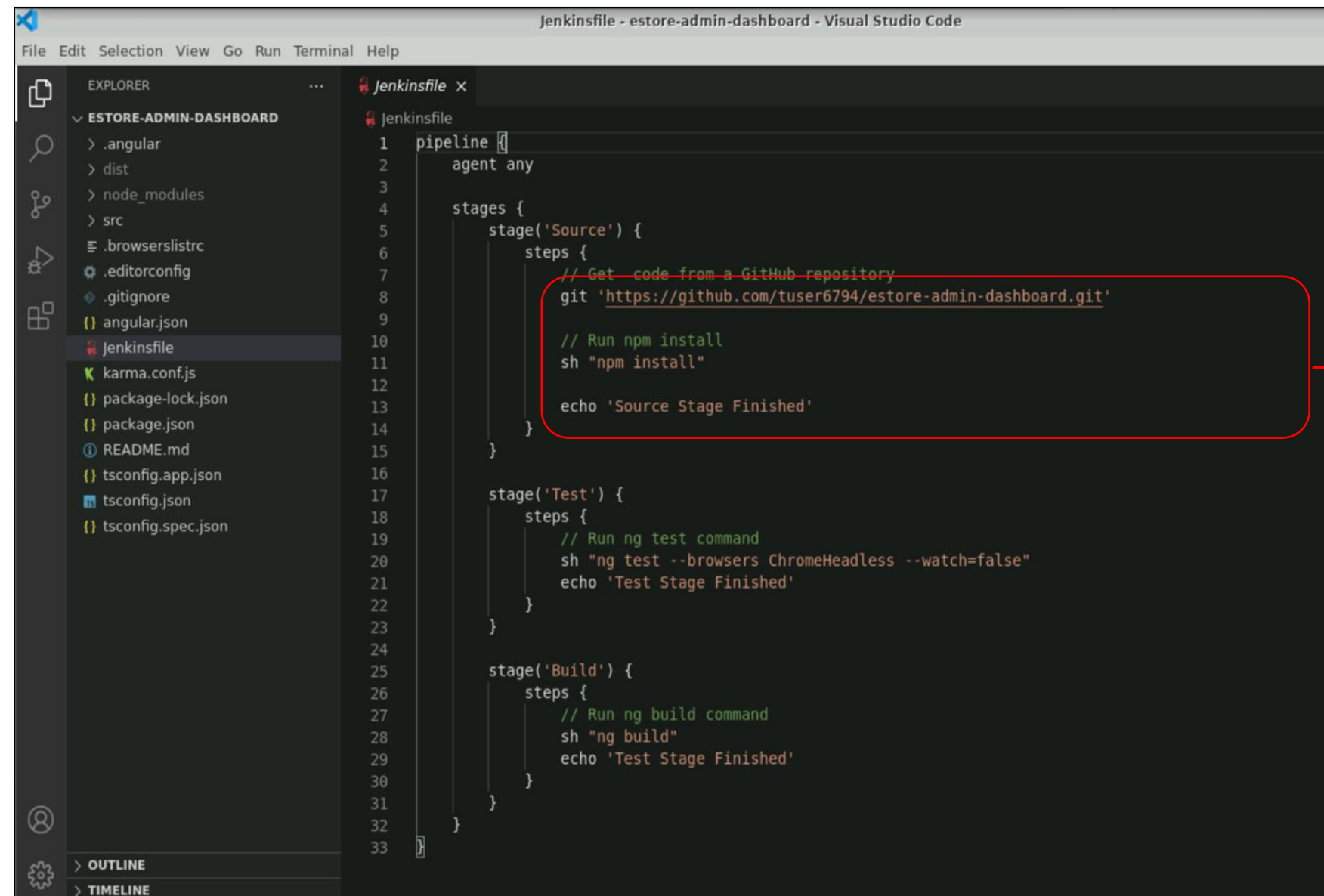
Push the code on GitHub for the web-admin project.

A terminal window titled "erishantgmail@ip-172-31-84-97: ~/Downloads/estore-admin-dashboard" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command "git push -u origin master" and its output: "Enumerating objects: 82, done.", "Counting objects: 100% (82/82), done.", "Delta compression using up to 4 threads", "Compressing objects: 100% (77/77), done.", "Writing objects: 100% (82/82), 213.56 KiB | 7.36 MiB/s, done.", "Total 82 (delta 14), reused 0 (delta 0)", "remote: Resolving deltas: 100% (14/14), done.", "To https://github.com/tuser6794/estore-admin-dashboard.git", "* [new branch] master -> master", and "Branch 'master' set up to track remote branch 'master' from 'origin'". The prompt returns to "erishantgmail@ip-172-31-84-97:~/Downloads/estore-admin-dashboard\$".

```
erishantgmail@ip-172-31-84-97: ~/Downloads/estore-admin-dashboard
File Edit View Search Terminal Help
erishantgmail@ip-172-31-84-97:~/Downloads/estore-admin-dashboard$ git push -u origin master
Enumerating objects: 82, done.
Counting objects: 100% (82/82), done.
Delta compression using up to 4 threads
Compressing objects: 100% (77/77), done.
Writing objects: 100% (82/82), 213.56 KiB | 7.36 MiB/s, done.
Total 82 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), done.
To https://github.com/tuser6794/estore-admin-dashboard.git
 * [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
erishantgmail@ip-172-31-84-97:~/Downloads/estore-admin-dashboard$
```

Configure Jenkins Pipeline Stages

Create a Jenkinsfile for Jenkins to build the **project** as a Pipeline in Jenkins.

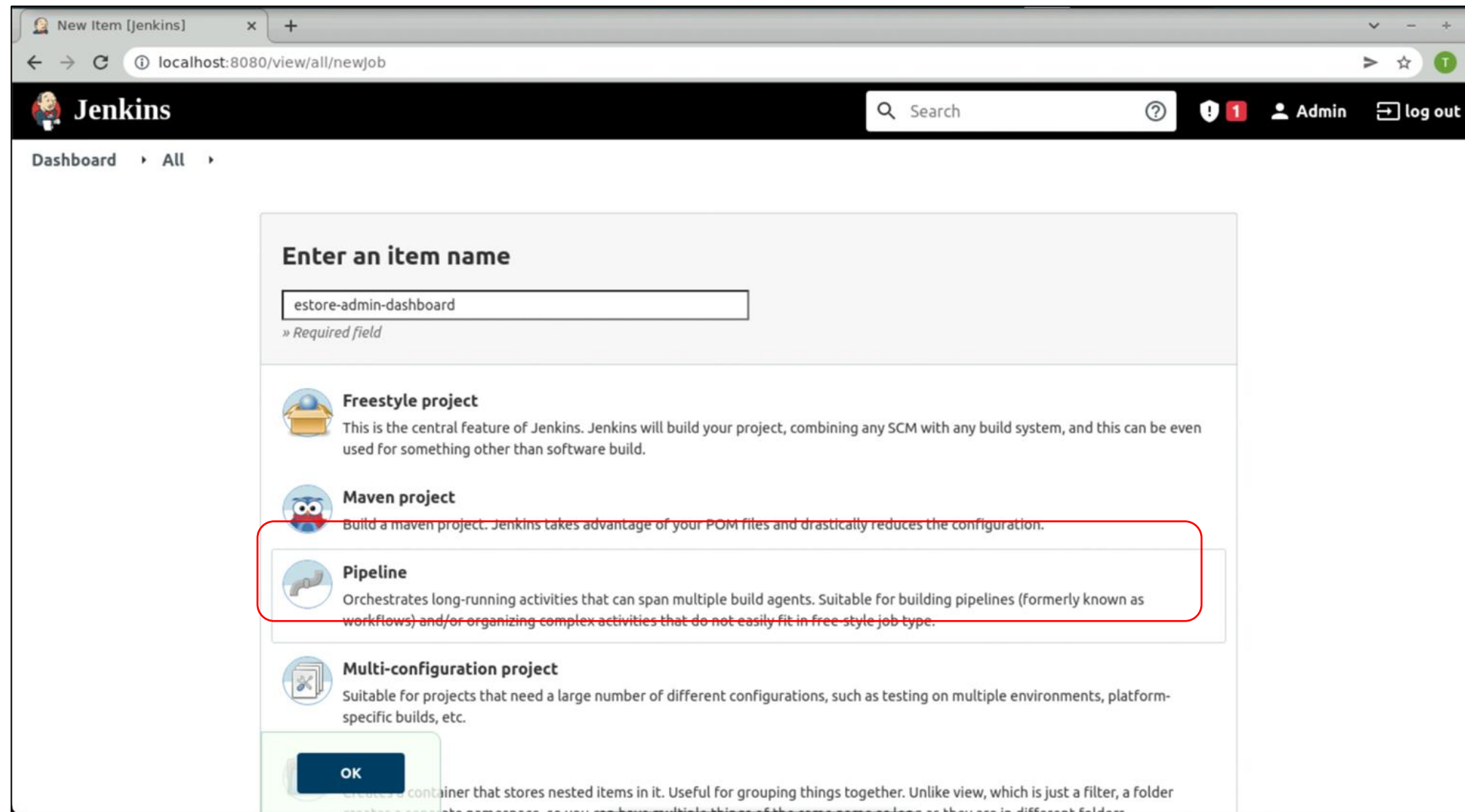


```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Source') {
6       steps {
7         // Get code from a GitHub repository
8         git 'https://github.com/tuser6794/estore-admin-dashboard.git'
9
10        // Run npm install
11        sh "npm install"
12
13        echo 'Source Stage Finished'
14      }
15    }
16
17    stage('Test') {
18      steps {
19        // Run ng test command
20        sh "ng test --browsers ChromeHeadless --watch=false"
21        echo 'Test Stage Finished'
22      }
23    }
24
25    stage('Build') {
26      steps {
27        // Run ng build command
28        sh "ng build"
29        echo 'Test Stage Finished'
30      }
31    }
32  }
33 }
```

Create Jenkinsfile in the root directory of the project and push it to GitHub so that Jenkins can use it for Pipeline.

Create a Jenkins Pipeline Project for the Web-Admin Project

Create a new project in Jenkins of type Pipeline.



Configure Jenkins Pipeline SCM for the Web-Admin Project

Configure the Jenkins Pipeline project by passing the GitHub repository URL.

The screenshot shows the Jenkins web interface for configuring a pipeline. The browser address bar indicates the URL is `localhost:8080/job/estore-admin-dashboard/configure`. The page has a breadcrumb trail: `Dashboard > estore-admin-dashboard >`. Below this, there are tabs for `General`, `Build Triggers`, `Advanced Project Options`, and `Pipeline`, with the `Pipeline` tab selected. The `Pipeline` section has a sub-header `Definition` and a dropdown menu set to `Pipeline script from SCM`. A red rounded rectangle highlights the `SCM` configuration area, which includes a dropdown menu set to `Git`, a `Repositories` section with a `Repository URL` field containing `https://github.com/tuser6794/estore-admin-dashboard.git`, and a `Credentials` section with a dropdown set to `- none -` and an `Add` button. Below the highlighted area, there is an `Advanced...` button. At the bottom of the configuration form, there are `Save` and `Apply` buttons, and a partially visible `Add Repository` button.

Configure Jenkinsfile in Jenkins for the Web-Admin Project

From the branch specifier, **select** the branch.

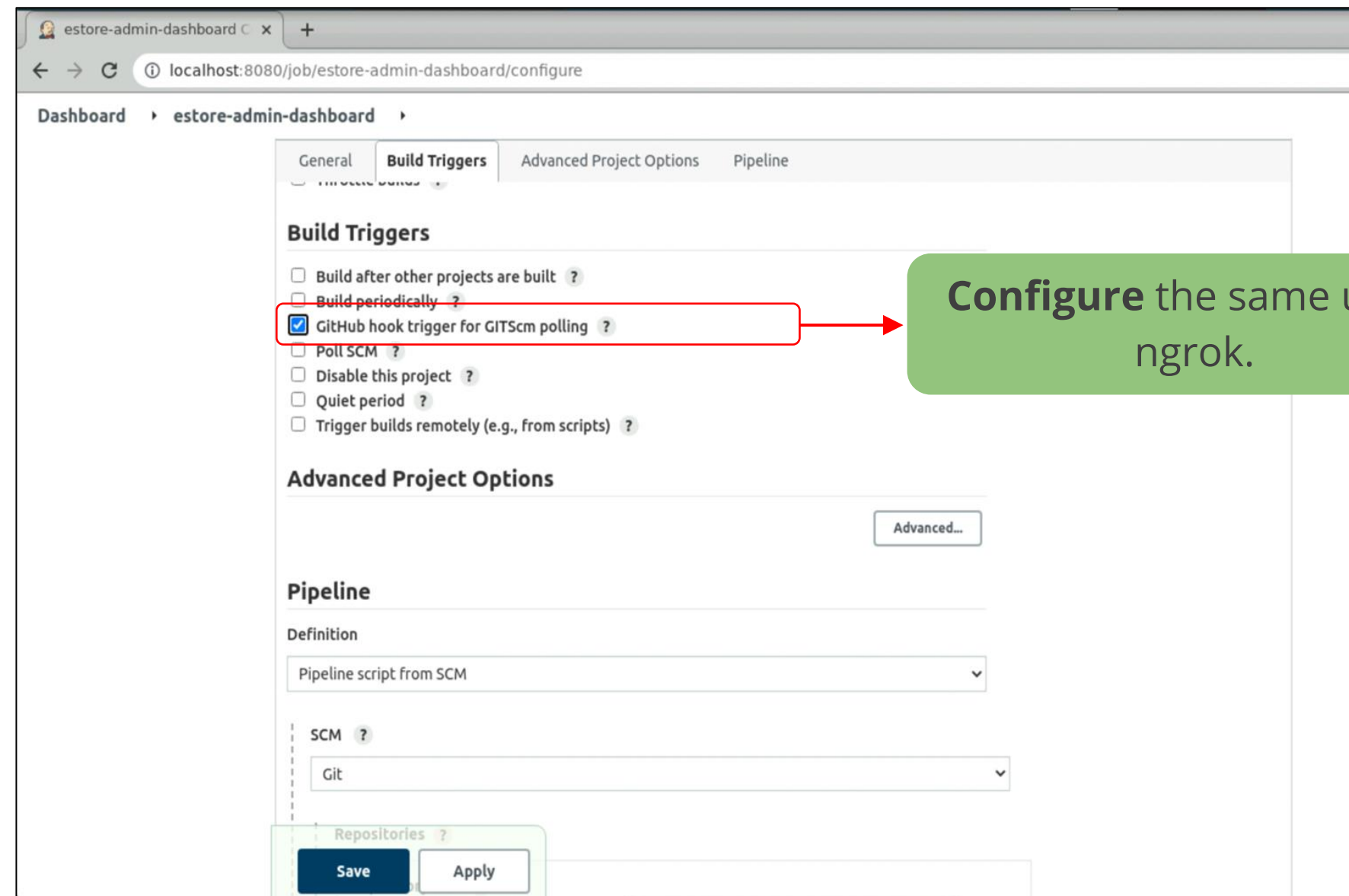
The screenshot shows the Jenkins Pipeline configuration page for a job named 'estore-admin-dashboard'. The 'Pipeline' tab is selected, and the 'Branch Specifier (blank for 'any')' field is highlighted with a red box. The field contains the text '*/master'. Below this field is an 'Add Branch' button. Further down, the 'Script Path' field is also highlighted with a red box and contains the text 'Jenkinsfile'. At the bottom of the configuration area, there is a 'Save' button and an 'Apply' button. The 'Repository browser' is set to '(Auto)' and 'Lightweight checkout' is checked.

Select the default branch, which is master.

The Script Path contains the Jenkinsfile, which is created in the root directory of the project.

Optional: GitHub Hook Trigger for the Web-Admin Project

Configure **GitHub trigger** for GITScm Polling. This option will work when Jenkins is running with a proper URL instead of the localhost.



Run the Jenkins Pipeline Project for the Web-Admin Project

Notice the stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build

Pipeline estore-admin-dashboard
Angular Admin Web Project

Build Now

Stage View

	Declarative: Checkout SCM	Source	Test	Build
Average stage times: (Average <u>full</u> run time: ~1min 19s)	427ms	26s	18s	30s
#1 Feb 26 11:43 No Changes	427ms	26s	18s	30s

Permalinks

Set Up Jenkins for the Angular End-User Project

Create Git Repository on GitHub for the End-User Project

Create a new repository on your GitHub account which will be used by Jenkins to sync the code in SCM.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

tuser6794 / estore-end-user ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-octo-broccoli?](#)

Description (optional)

Angular End User Repository

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

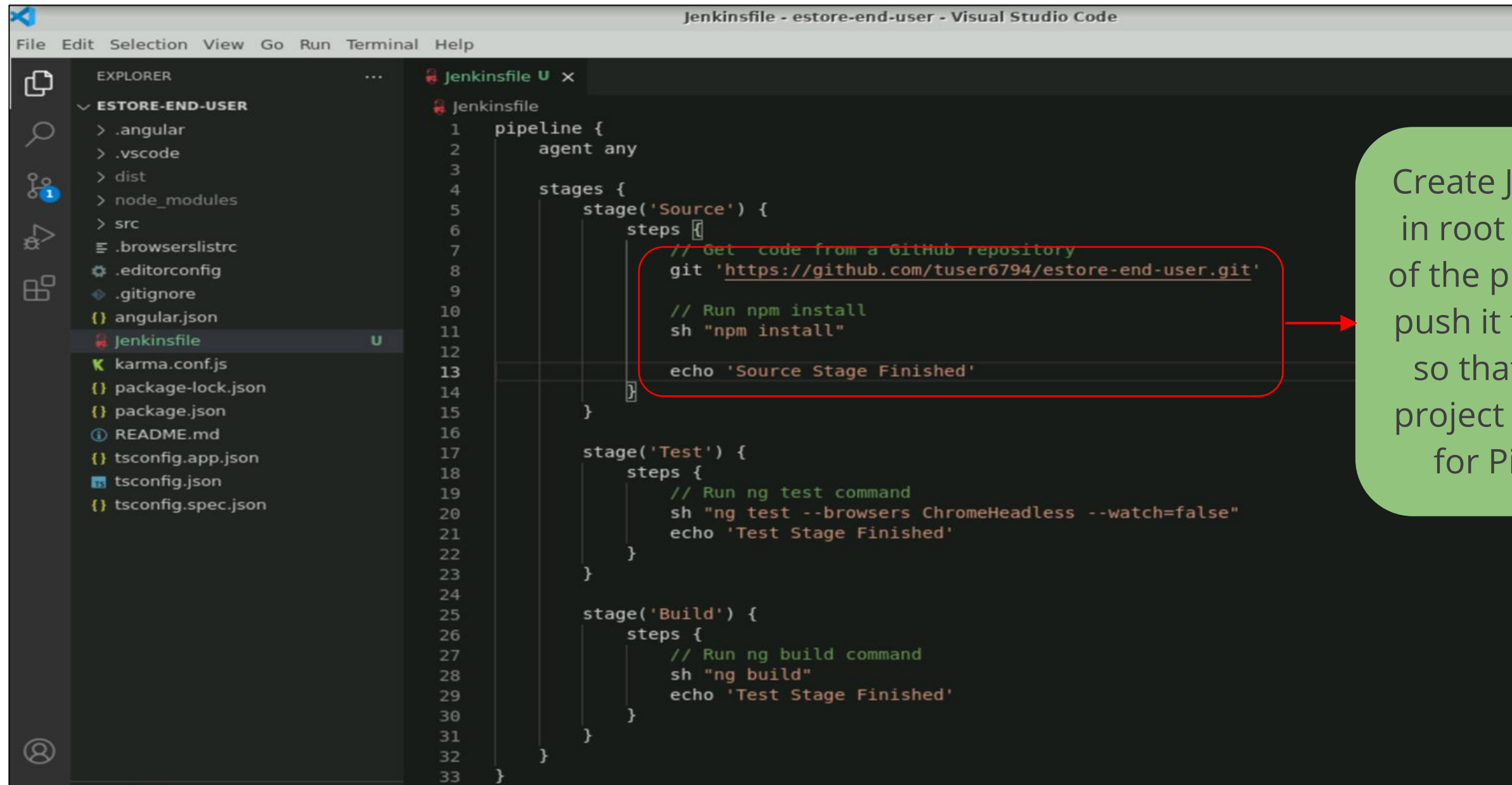
Sync the Project on GitHub for the End-User Project

Push the code on GitHub for the end-user project.

```
erishantgmail@ip-172-31-84-97: ~/Downloads/estore-end-user
File Edit View Search Terminal Help
erishantgmail@ip-172-31-84-97:~/Downloads/estore-end-user$ git push -u origin master
Enumerating objects: 87, done.
Counting objects: 100% (87/87), done.
Delta compression using up to 4 threads
Compressing objects: 100% (83/83), done.
Writing objects: 100% (87/87), 212.60 KiB | 7.59 MiB/s, done.
Total 87 (delta 15), reused 0 (delta 0)
remote: Resolving deltas: 100% (15/15), done.
To https://github.com/tuser6794/estore-end-user.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
erishantgmail@ip-172-31-84-97:~/Downloads/estore-end-user$
```

Configure Jenkins Pipeline Stages for the End-User Project

Create a Jenkinsfile for Jenkins to build project as pipeline in Jenkins.



```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Source') {
6       steps {
7         // Get code from a GitHub repository
8         git 'https://github.com/tuser6794/estore-end-user.git'
9
10        // Run npm install
11        sh "npm install"
12
13        echo 'Source Stage Finished'
14      }
15    }
16
17    stage('Test') {
18      steps {
19        // Run ng test command
20        sh "ng test --browsers ChromeHeadless --watch=false"
21        echo 'Test Stage Finished'
22      }
23    }
24
25    stage('Build') {
26      steps {
27        // Run ng build command
28        sh "ng build"
29        echo 'Test Stage Finished'
30      }
31    }
32  }
33 }
```

Create Jenkinsfile in root directory of the project and push it to GitHub so that Jenkins project can use it for Pipeline.

Create a Jenkins Pipeline Project for the End-User Project

Create a new project in Jenkins of type Pipeline.

localhost:8080/view/all/newJob

Jenkins

Search

Dashboard > All >

Enter an item name

estore-end-user

» Required field

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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.

OK

Configure Jenkins Pipeline SCM for the End-User Project

Configure the Jenkins Pipeline Project by passing the GitHub Repository URL.

The screenshot shows the Jenkins configuration interface for a project named 'estore-end-user'. The 'Pipeline' tab is selected, and the 'Definition' is set to 'Pipeline script from SCM'. A red rounded rectangle highlights the 'SCM' section, which includes a dropdown menu set to 'Git', a 'Repositories' section with a 'Repository URL' field containing 'https://github.com/tuser6794/estore-end-user.git', and a 'Credentials' dropdown set to '- none -'. Below the highlighted section is an 'Add Repository' button. At the bottom of the configuration area are 'Save' and 'Apply' buttons. The browser address bar shows 'localhost:8080/job/estore-end-user/configure'.

Configure Jenkinsfile in Jenkins for the End-User Project

From the branch specifier, select the branch.

localhost:8080/job/estore-end-user/configure

Dashboard > estore-end-user >

General Build Triggers Advanced Project Options **Pipeline**

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

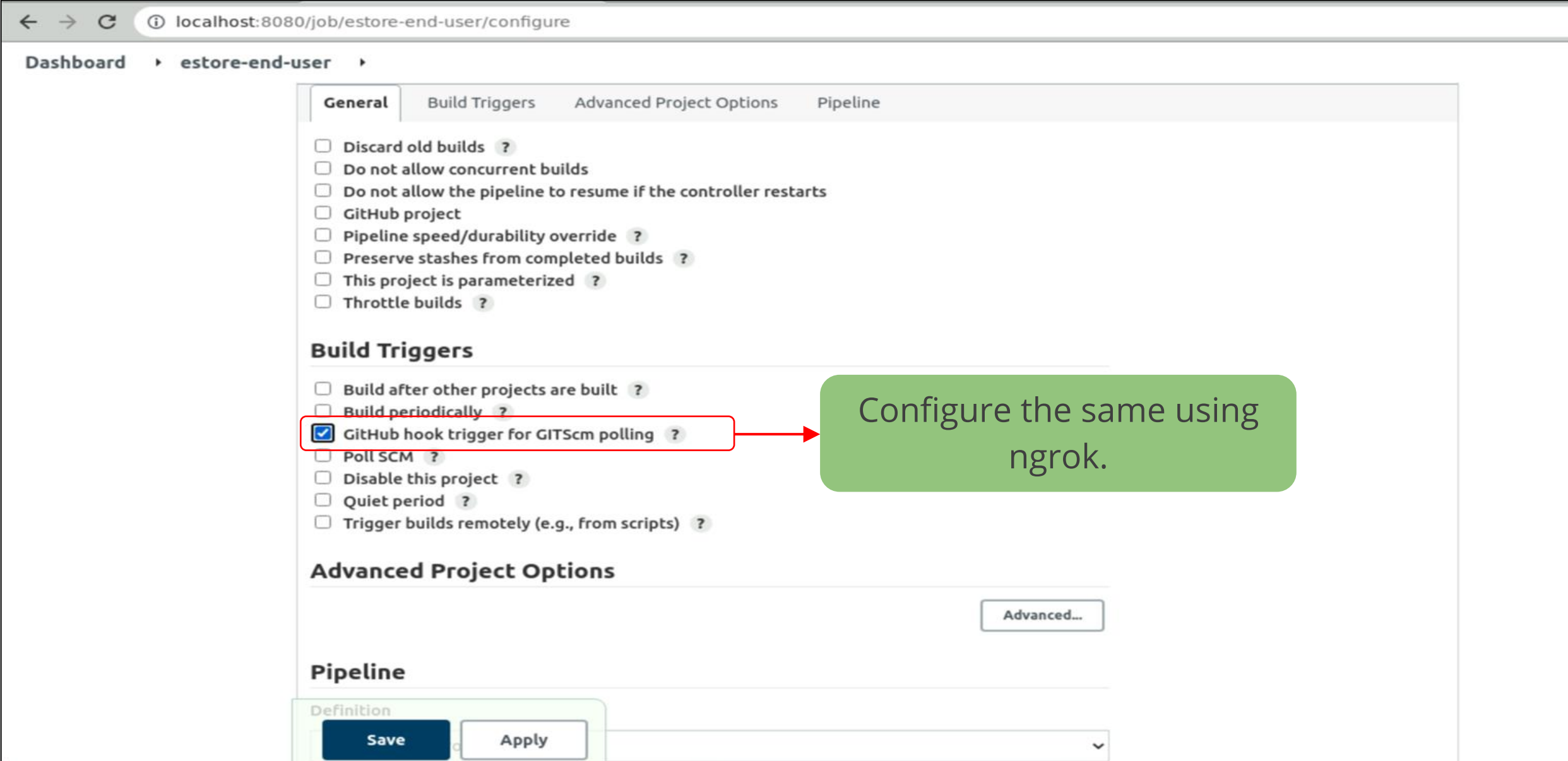
Save Apply

Select the default branch, which is master.

Script Path contains the Jenkinsfile, which is created in the root directory of the project.

Optional: GitHub Hook Trigger for the End-User Project

Configure GitHub trigger for GITScm Polling. This option will work when Jenkins is running with a proper URL instead of the localhost.



The screenshot shows the Jenkins configuration interface for a job named 'estore-end-user'. The browser address bar indicates the URL is 'localhost:8080/job/estore-end-user/configure'. The configuration page has tabs for 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. The 'Build Triggers' tab is active, and the checkbox for 'GitHub hook trigger for GITScm polling' is checked. A red box highlights this checkbox, and a red arrow points from it to a green callout box containing the text 'Configure the same using ngrok.'.

Dashboard > estore-end-user >

General Build Triggers Advanced Project Options Pipeline

- ☐ Discard old builds ?
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☐ GitHub project
- ☐ Pipeline speed/durability override ?
- ☐ Preserve stashes from completed builds ?
- ☐ This project is parameterized ?
- ☐ Throttle builds ?

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?
- ☐ Disable this project ?
- ☐ Quiet period ?
- ☐ Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced...

Pipeline

Definition

Save Apply

Run the Jenkins Pipeline Project for the End-User Project

Notice the stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build

Pipeline estore-end-user

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend ^

Filter builds...

#1 Feb 26, 2022, 12:17 PM

Atom feed for all Atom feed for failures

Stage View

Average stage times:
(Average full run time: ~1min 0s)

	Declarative: Checkout SCM	Source	Test	Build
Average	356ms	15s	13s	30s
#1	356ms	15s	13s	30s

Permalinks

Containerize the Angular Web-Admin Project

Configure Docker Using Dockerfile for the Web-Admin Project

The Dockerfile is a multi-stage docker build with two stages:

```
Dockerfile - estore-admin-dashboard - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
ESTORE-ADMIN-DASHBOARD
  > .angular
  > dist
  > node_modules
  > src
  .browserslistrc
  .editorconfig
  .gitignore
  angular.json
  Dockerfile
  Jenkinsfile
  karma.conf.js
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json

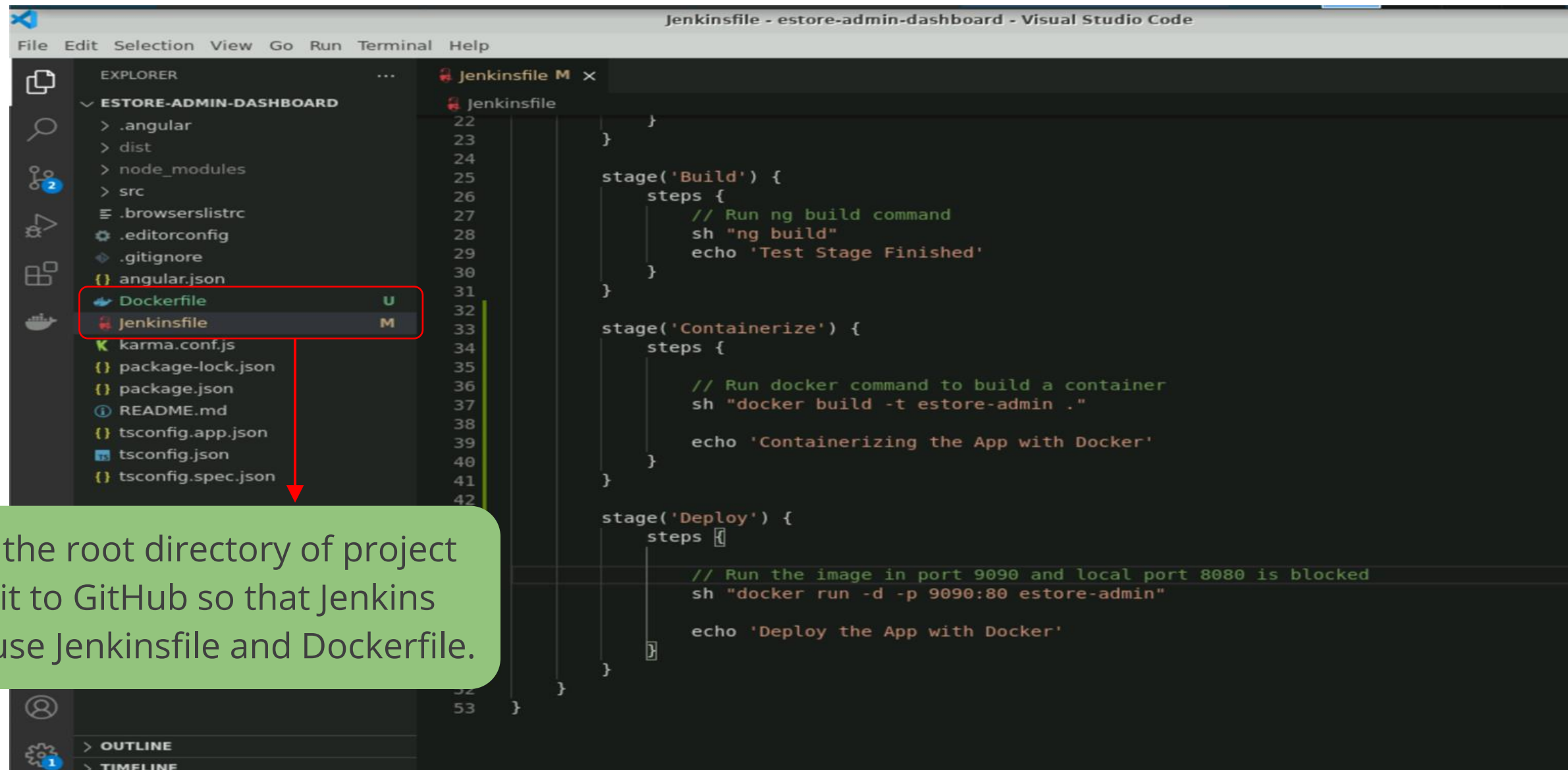
Dockerfile
1  # Use official node image as the base image
2  FROM node:latest as build
3
4  # Set the working directory as app
5  WORKDIR /usr/local/app
6
7  # Add the source code to app
8  COPY ./ /usr/local/app/
9
10 # Install the dependencies
11 RUN npm install
12
13 # Generate the build of the application
14 RUN npm run build
15
16 # Use official nginx image as the base image
17 FROM nginx:latest
18
19 # Copy the build output for nginx contents
20 COPY --from=build /usr/local/app/dist/admin-dashboard /usr/share/nginx/html
21
22 # Expose port 80
23 EXPOSE 80
```

Building the angular project into dist directory for production.

Serving the application using NGINX web server on port 80.

Configure Jenkins Pipeline Stages for Docker

Add the stages for containerizing the Angular web admin that is building and running container.



The screenshot shows the Visual Studio Code interface with the 'Jenkinsfile - estore-admin-dashboard' open. The Explorer view on the left shows the project structure, with 'Dockerfile' and 'Jenkinsfile' highlighted. A red box surrounds these files, and a red arrow points from the 'Dockerfile' to a green callout box. The main editor shows the Jenkinsfile content:

```
Jenkinsfile
22
23
24
25 stage('Build') {
26     steps {
27         // Run ng build command
28         sh "ng build"
29         echo 'Test Stage Finished'
30     }
31 }
32
33 stage('Containerize') {
34     steps {
35
36         // Run docker command to build a container
37         sh "docker build -t estore-admin ."
38
39         echo 'Containerizing the App with Docker'
40     }
41 }
42
43 stage('Deploy') {
44     steps {
45
46         // Run the image in port 9090 and local port 8080 is blocked
47         sh "docker run -d -p 9090:80 estore-admin"
48
49         echo 'Deploy the App with Docker'
50     }
51 }
52
53 }
```

Create it in the root directory of project and push it to GitHub so that Jenkins Project can use Jenkinsfile and Dockerfile.

Run the Jenkins Pipeline Project for the Web-Admin Project

Notice stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build > Containerize > Deploy

Pipeline estore-admin-dashboard
Angular Admin Web Project

Build Now

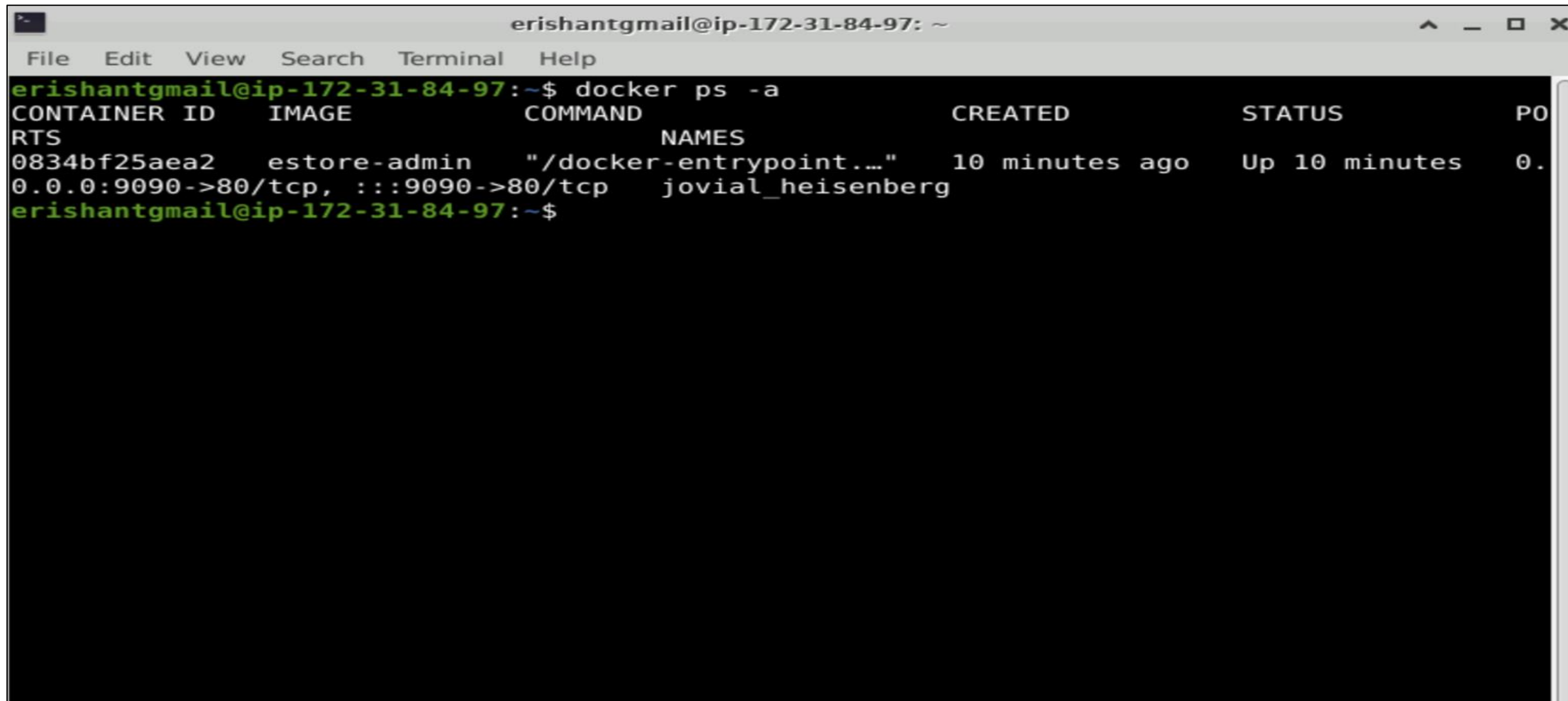
Stage View

Average stage times:
(Average full run time: ~1min 39s)

	Declarative: Checkout SCM	Source	Test	Build	Containerize	Deploy
Average	326ms	14s	17s	27s	1min 13s	565ms
#2 (Feb 26, 13:01)	225ms	3s	17s	25s	1min 13s	565ms
#1 (Feb 26, 11:43)	427ms	26s	18s	30s		

Check Docker ps -a

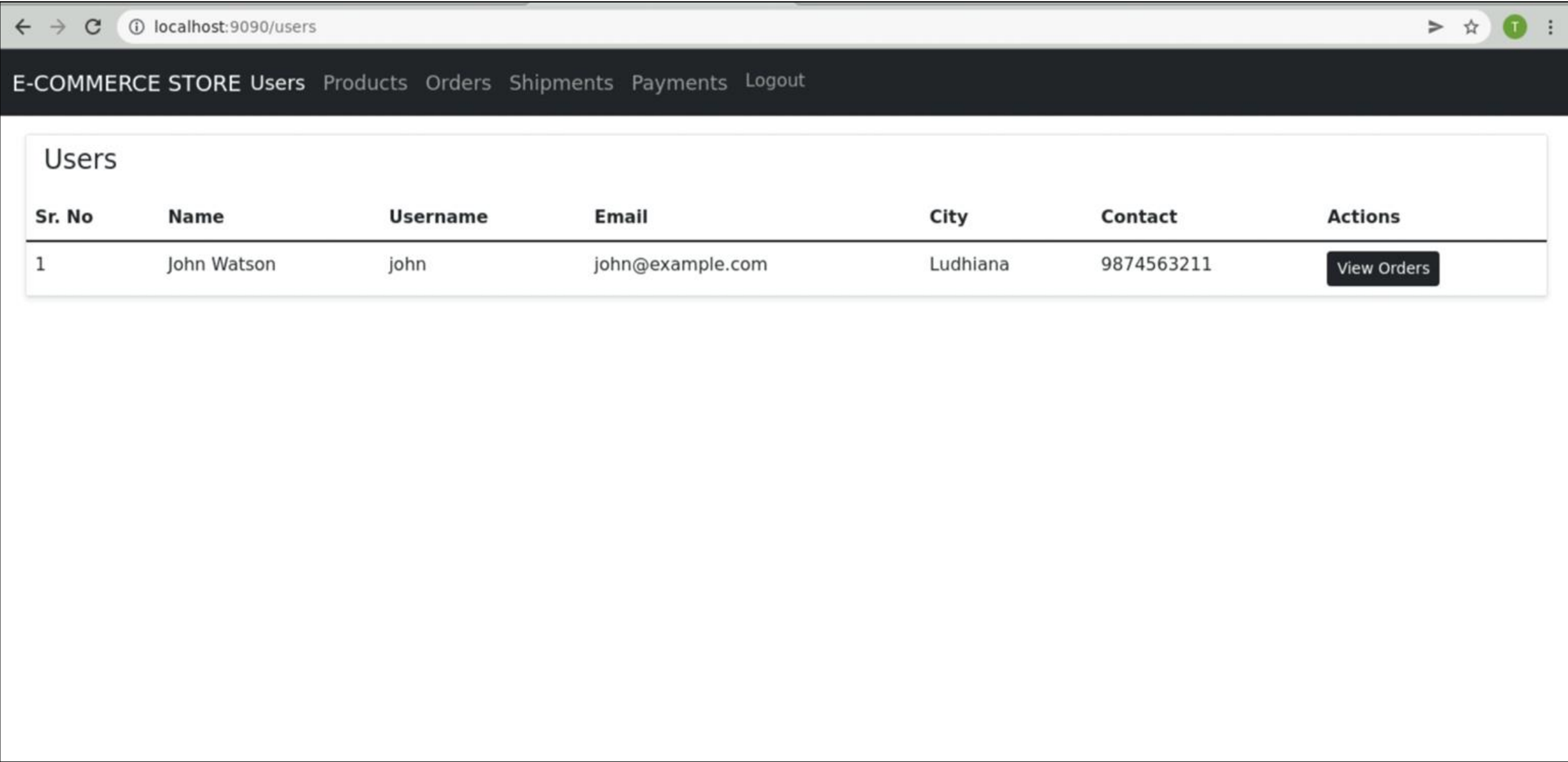
Check the status of running container by **docker ps -a**, which is a command to validate Jenkins build.



```
erishantgmail@ip-172-31-84-97: ~  
File Edit View Search Terminal Help  
erishantgmail@ip-172-31-84-97:~$ docker ps -a  
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS  
0834bf25aea2   estore-admin   "/docker-entrypoint...." 10 minutes ago Up 10 minutes 0.0.0:9090->80/tcp, :::9090->80/tcp  
erishantgmail@ip-172-31-84-97:~$
```

Check the Web-Admin Project on Localhost

The web-admin project is up and running in a docker container.



The screenshot shows a web browser window with the address bar displaying 'localhost:9090/users'. The page has a dark header with the text 'E-COMMERCE STORE' and navigation links: 'Users', 'Products', 'Orders', 'Shipments', 'Payments', and 'Logout'. The main content area is titled 'Users' and contains a table with the following data:

Sr. No	Name	Username	Email	City	Contact	Actions
1	John Watson	john	john@example.com	Ludhiana	9874563211	View Orders

TECHNOLOGY

Computing

Caltech

**Center for Technology &
Management Education**

Dockerise the Pipeline

Containerize Angular End-User Project

Configure Docker Using Dockerfile for the End-User Project

The Dockerfile is a multi-stage docker build with two stages:

The screenshot shows the Visual Studio Code editor with a Dockerfile for a project named 'estore-end-user'. The Dockerfile is a multi-stage build with two stages. The first stage builds the application, and the second stage serves it using NGINX. Callouts explain the purpose of each stage.

```
Dockerfile - estore-end-user - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
└─ ESTORE-END-USER
  ├── .angular
  ├── .vscode
  ├── dist/estore-user-app
  ├── node_modules
  ├── src
  ├── .browserslistrc
  ├── .editorconfig
  ├── .gitignore
  ├── angular.json
  ├── Dockerfile
  ├── Jenkinsfile
  ├── karma.conf.js
  ├── package-lock.json
  ├── package.json
  ├── README.md
  ├── tsconfig.app.json
  ├── tsconfig.json
  └── tsconfig.spec.json

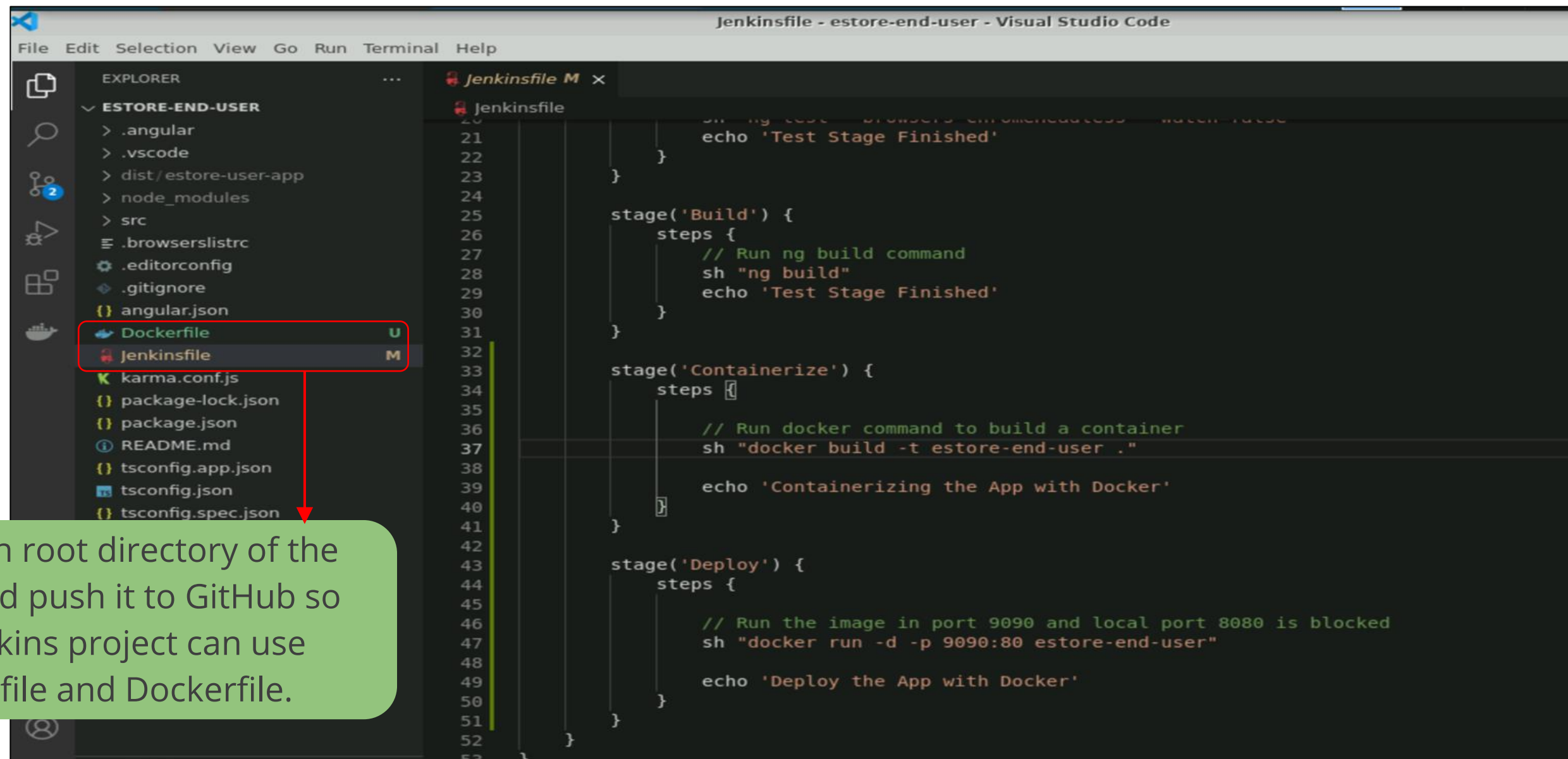
Dockerfile U x
Dockerfile > ...
1  # Use official node image as the base image
2  FROM node:latest as build
3
4  # Set the working directory as app
5  WORKDIR /usr/local/app
6
7  # Add the source code to app
8  COPY ./ /usr/local/app/
9
10 # Install the dependencies
11 RUN npm install
12
13 # Generate the build of the application
14 RUN npm run build
15
16 # Use official nginx image as the base image
17 FROM nginx:latest
18
19 # Copy the build output for nginx contents
20 COPY --from=build /usr/local/app/dist/estore-user-app /usr/share/nginx/html
21
22 # Expose port 80
23 EXPOSE 80
```

Building the angular project into dist directory for production.

Serving the application using NGINX web server on port 80.

Configure Jenkins Pipeline Stages for Docker

Add the stages for containerizing the Angular web admin that is building and running container.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Jenkinsfile editor on the right. The Explorer sidebar shows the project structure for 'ESTORE-END-USER'. The Jenkinsfile editor shows the following code:

```
20  // Run ng test command
21  sh "ng test --browser=chrome --port=9000 --watch=false"
22  echo 'Test Stage Finished'
23  }
24  }
25  stage('Build') {
26  steps {
27  // Run ng build command
28  sh "ng build"
29  echo 'Test Stage Finished'
30  }
31  }
32  stage('Containerize') {
33  steps {
34  // Run docker command to build a container
35  sh "docker build -t estore-end-user ."
36  echo 'Containerizing the App with Docker'
37  }
38  }
39  stage('Deploy') {
40  steps {
41  // Run the image in port 9090 and local port 8080 is blocked
42  sh "docker run -d -p 9090:80 estore-end-user"
43  echo 'Deploy the App with Docker'
44  }
45  }
46  }
```

A red box highlights the 'Dockerfile' and 'Jenkinsfile' files in the Explorer sidebar. A red arrow points from this box to a green callout box containing the text: 'Create it in root directory of the project and push it to GitHub so that Jenkins project can use Jenkinsfile and Dockerfile.'

Run the Jenkins Pipeline Project for the End-User Project

Notice stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build > Containerize > Deploy

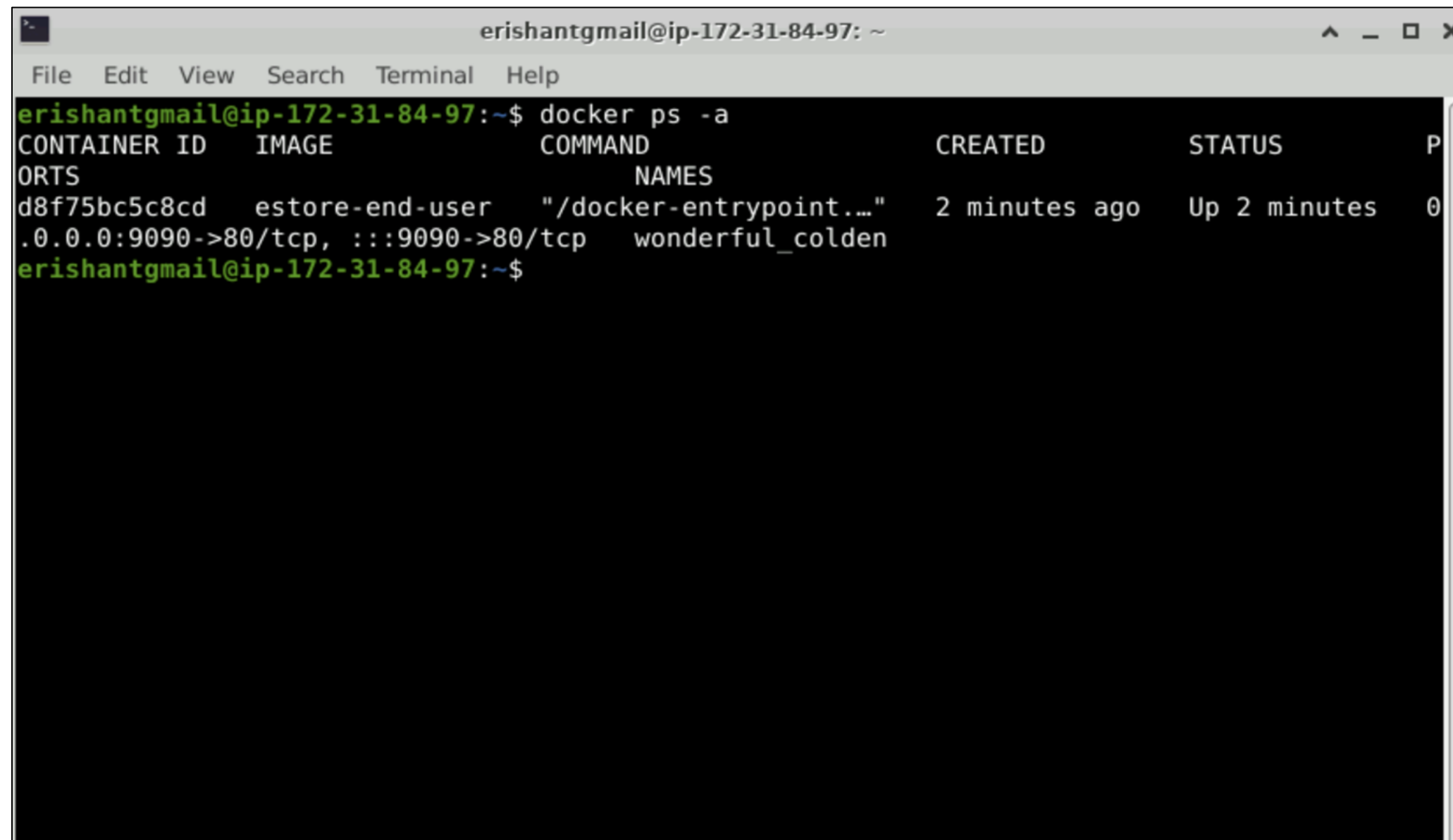
The screenshot shows the Jenkins web interface for a pipeline named 'estore-admin-dashboard'. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build Now (highlighted with a red box and an arrow pointing to a green callout), Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area displays the 'Pipeline estore-admin-dashboard' with a 'Recent Changes' section and a 'Stage View' table. The 'Stage View' table shows the following stages and their durations:

Stage	Declarative: Checkout SCM	Source	Test	Build	Containerize	Deploy
Average stage times: (Average full run time: ~1min 39s)	326ms	14s	17s	27s	1min 13s	565ms
#2 Feb 26 13:01 No Changes	225ms	3s	17s	25s	1min 13s	565ms
#1 Feb 26 11:43 No Changes	427ms	26s	18s	30s		

A green callout box with the text 'Click on Build Now and build project.' is positioned over the 'Build Now' button in the sidebar.

Check Docker ps -a

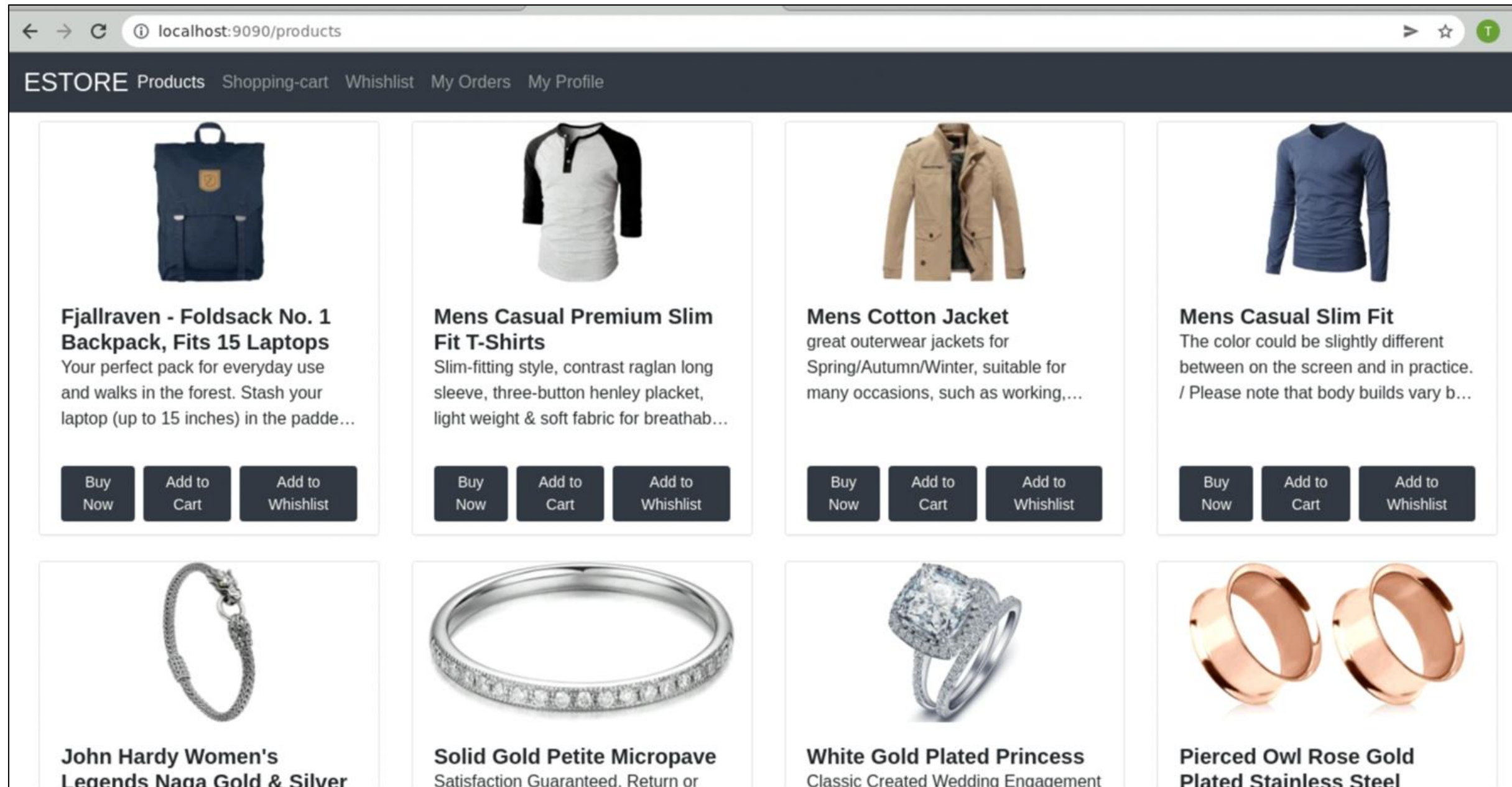
Check status of running container by **docker ps -a**, which is a command to validate Jenkins build.



```
erishantgmail@ip-172-31-84-97: ~  
File Edit View Search Terminal Help  
erishantgmail@ip-172-31-84-97:~$ docker ps -a  
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS  
d8f75bc5c8cd   estore-end-user  "/docker-entrypoint...."  2 minutes ago  Up 2 minutes  0.0.0.0:9090->80/tcp, :::9090->80/tcp  
erishantgmail@ip-172-31-84-97:~$
```

Check the End-User Web Project on Localhost

The End-user web project is up and running in a docker container.



Key Takeaways

- 🕒 Jenkins is an automation build server.
- 🕒 Jenkins Pipeline for Angular apps is set using Jenkinsfile.
- 🕒 Dockerfile is used to dockerize the Angular apps.
- 🕒 Jenkins Pipeline is built to dockerize the Angular apps.



Before the Next Class

You have successfully completed this session.

Before the next discussion, you should go through:

- AWS



What's Next?

Now we have finished our classes and design pattern for the back-end project with respect to end-user module. In our next live session, we will:

- See how to create Jenkins Pipeline for Java back end
- See how to work with Jenkinsfile for the Maven project
- Learn various commands in Dockerfile for the Java back-end project
- Explore CI/CD Pipeline with Jenkins

