Caltech | Center for Technology & Management Education

Jenkins and Docker
Java Back End

Powerd by simplilearn

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

JSP

Servlets

MongoDB

Maven

You Already Know

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

JUnit

Spring

Spring Boot

Webservices

Microservices

You Already Know

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

Docker

Jenkins

AWS

Recap

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 end user
- Implemented microservices with Spring Boot

Front End Back End Communication

- Used HTTP Client in Angular to communicate with Java back end
- Implemented communication for both admin and 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/MongoDB can be used to build at the back end.

A Day in the Life of a Full Stack Developer

Bob needs to develop a Java back end. 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 Dockerfile to build images and run them as containers in Docker and help Bob complete his task effectively and quickly.

Learning Objectives

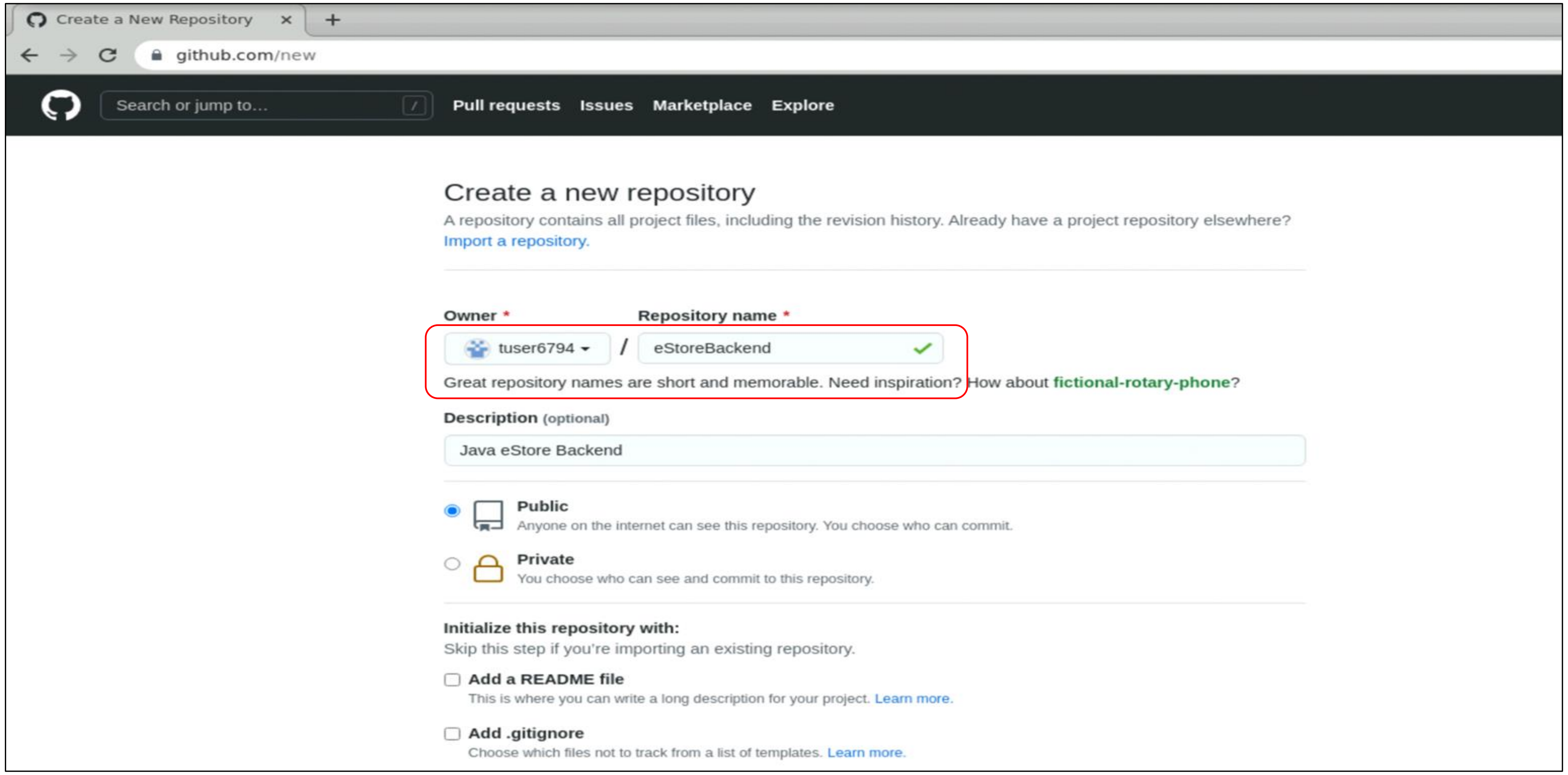
By the end of this lesson, you will be able to:

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

Set Up Jenkins for the Java Backend Project

Create Git Repository on GitHub for the Java Backend 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

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 * tuser6794 / Repository name * eStoreBackend

Great repository names are short and memorable. Need inspiration? How about fictional-rotary-phone?

Description (optional): Java eStore Backend

[✓ 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:

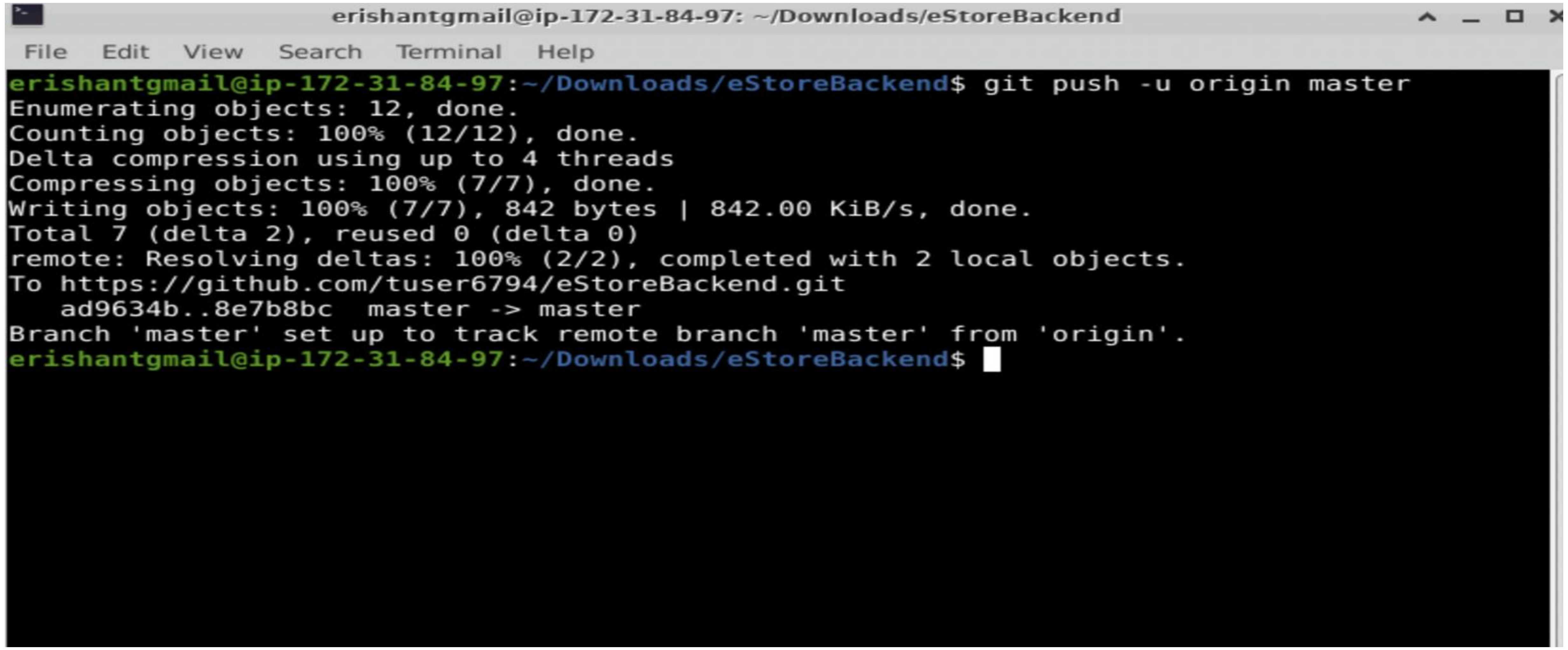
Skip this step if you're importing an existing repository.

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 Java Backend Project

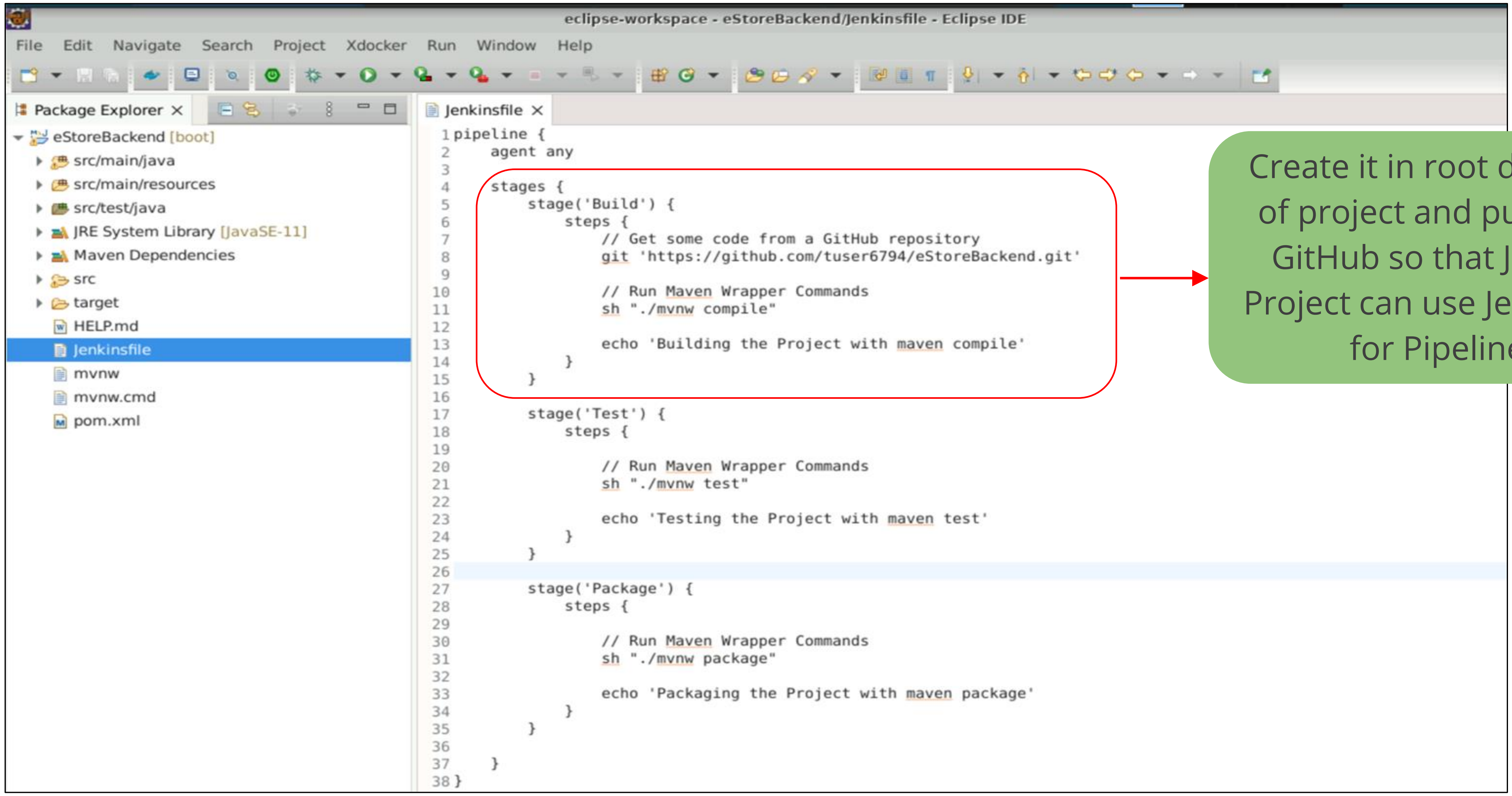
Push the code on GitHub for the Java Backend project.


erishantgmail@ip-172-31-84-97: ~/Downloads/eStoreBackend
File Edit View Search Terminal Help

```
erishantgmail@ip-172-31-84-97:~/Downloads/eStoreBackend$ git push -u origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 842 bytes | 842.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/tuser6794/eStoreBackend.git
   ad9634b..8e7b8bc  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
erishantgmail@ip-172-31-84-97:~/Downloads/eStoreBackend$
```

Configure Jenkins Pipeline Stages for the Java Backend Project

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



```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/tuser6794/eStoreBackend.git'

                // Run Maven Wrapper Commands
                sh "./mvnw compile"

                echo 'Building the Project with maven compile'
            }
        }

        stage('Test') {
            steps {

                // Run Maven Wrapper Commands
                sh "./mvnw test"

                echo 'Testing the Project with maven test'
            }
        }

        stage('Package') {
            steps {

                // Run Maven Wrapper Commands
                sh "./mvnw package"

                echo 'Packaging the Project with maven package'
            }
        }

    }
}
```

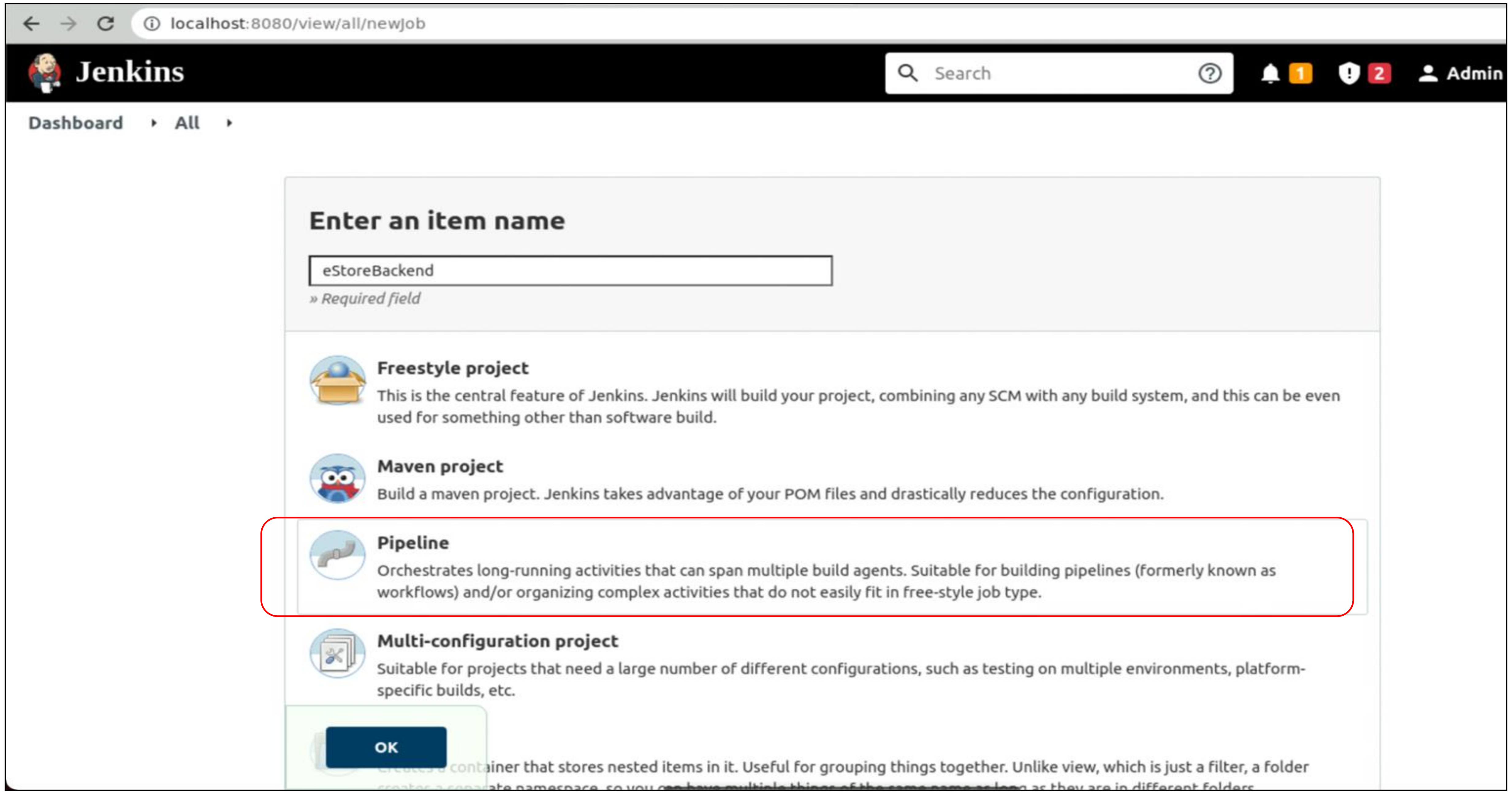
Create it in root directory of project and push it to GitHub so that Jenkins Project can use Jenkinsfile for Pipeline.

Powerd by simplilearn

Caltech Center for Technology & Management Education

Create a Jenkins Pipeline Project for the Java Backend Project

Create a new project in Jenkins of type Pipeline.


localhost:8080/view/all/newJob

Enter an item name

eStoreBackend

» 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.

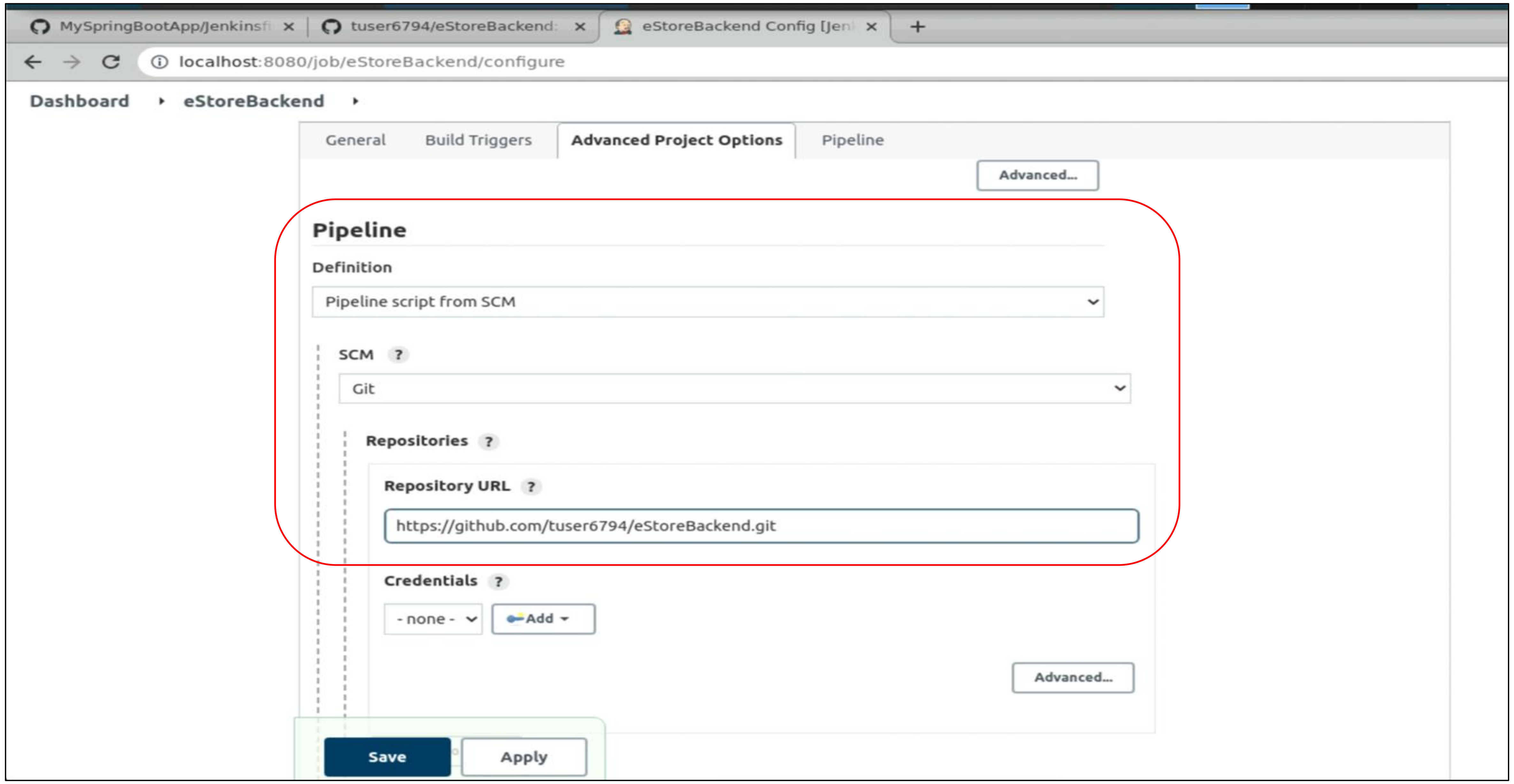
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

iner that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder

Configure Jenkins Pipeline SCM for the Java Backend Project

Configure the Jenkins Pipeline Project by passing GitHub Repository URL.



localhost:8080/job/eStoreBackend/configure

Pipeline

Definition: Pipeline script from SCM

SCM: Git

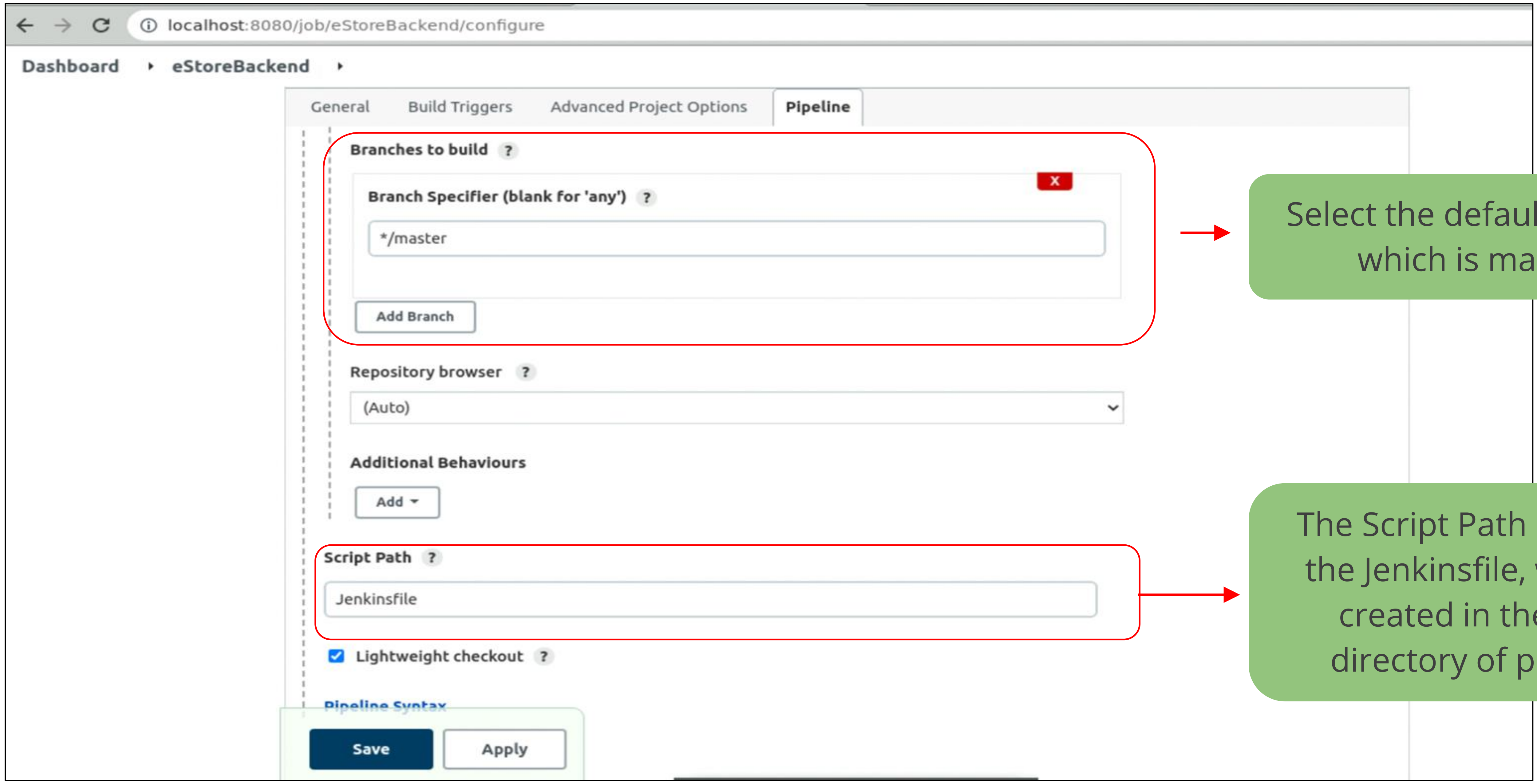
Repositories

Repository URL: https://github.com/tuser6794/eStoreBackend.git

Credentials: - none - Add

Configure Jenkinsfile in Jenkins for the Java Backend Project

From the branch specifier, select the branch.



localhost:8080/job/eStoreBackend/configure

Dashboard ▸ eStoreBackend ▸

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

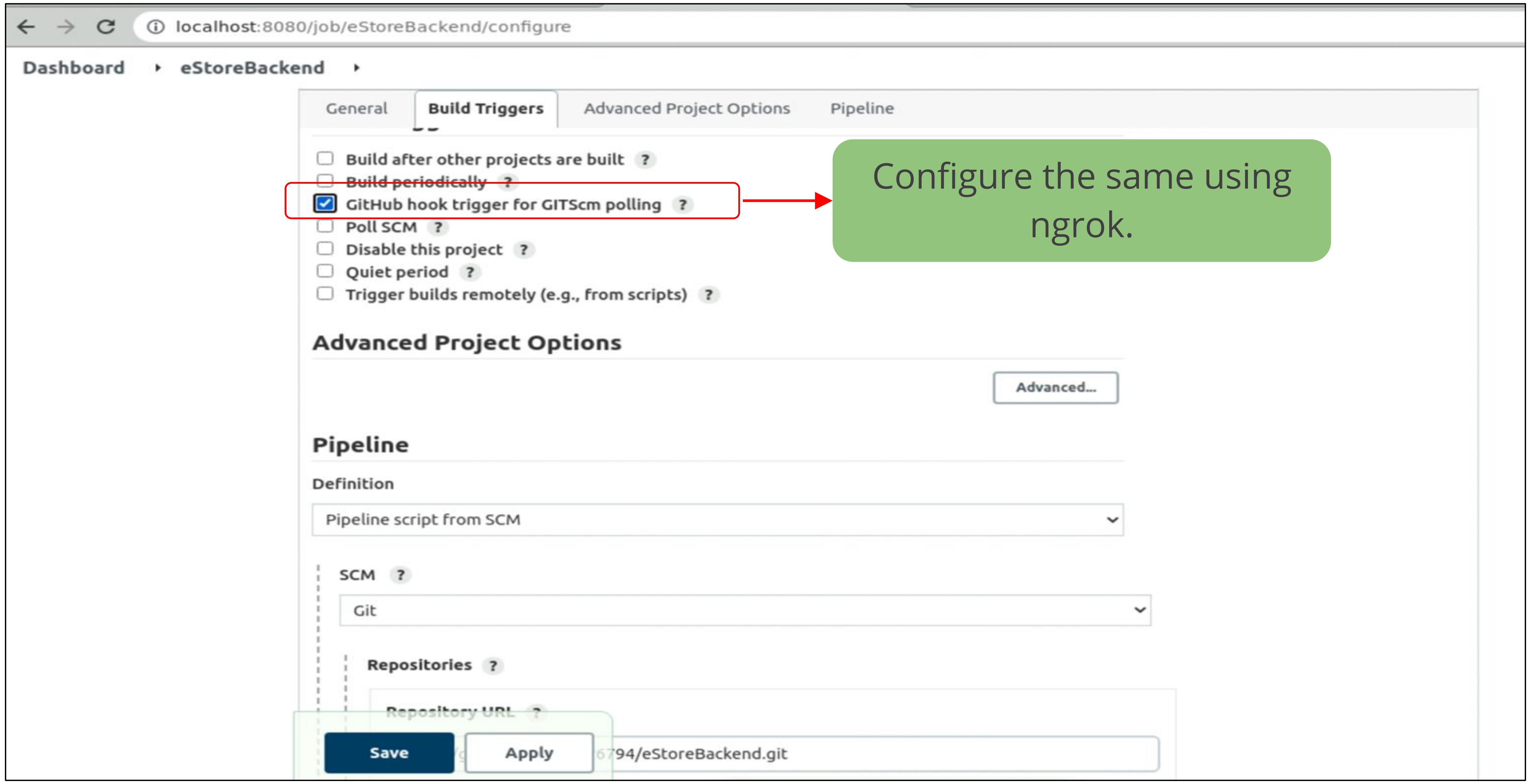
✓ Lightweight checkout

Select the default branch, which is master.

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

Optional: GitHub Hook Trigger for the Java Backend Project

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



localhost:8080/job/eStoreBackend/configure

Dashboard ▸ eStoreBackend ▸

General | **Build Triggers** | Advanced Project Options | Pipeline

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

Advanced Project Options

Pipeline

Definition: Pipeline script from SCM

SCM: Git

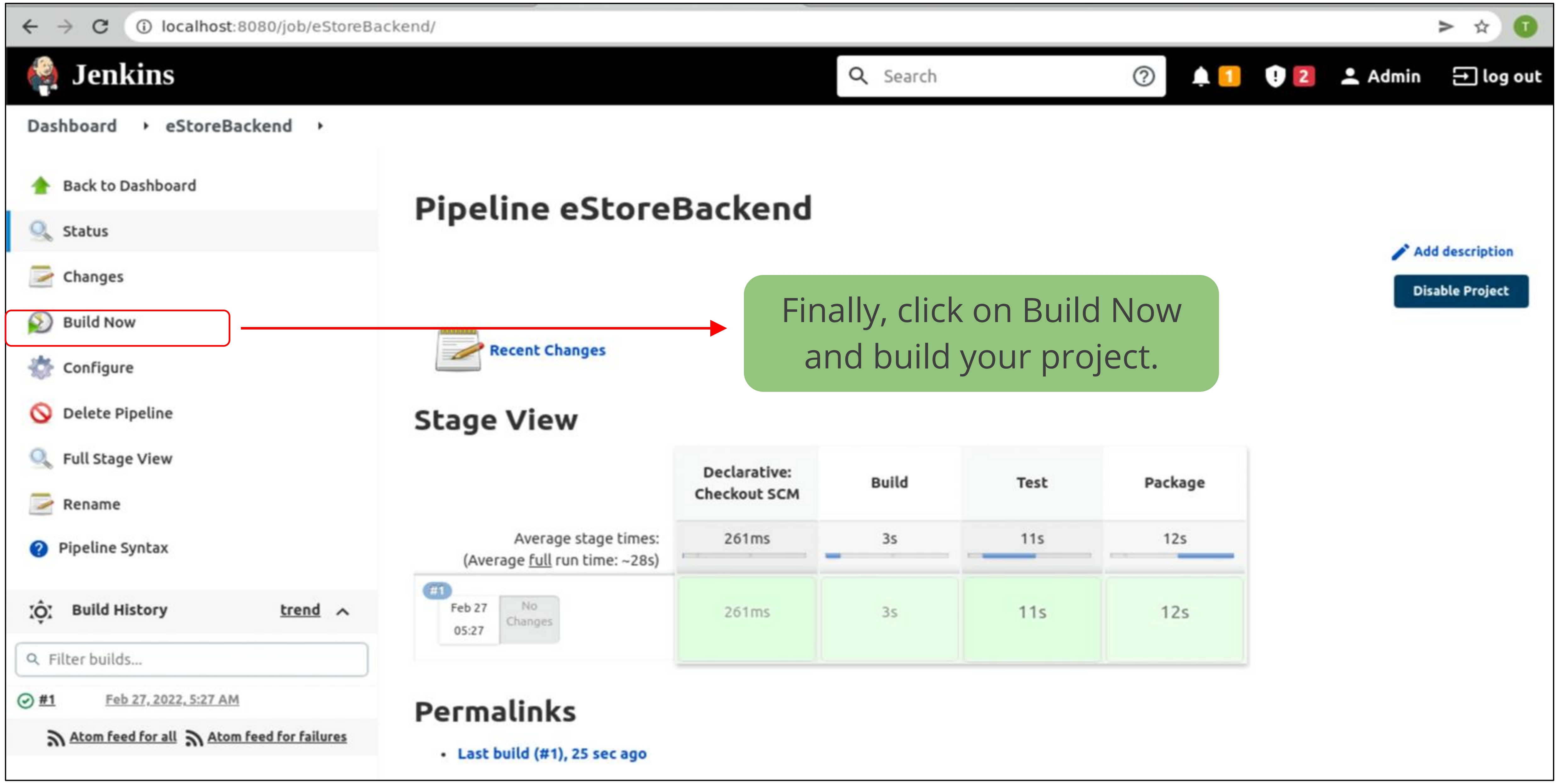
Repositories

Repository URL: …6794/eStoreBackend.git

Configure the same using ngrok.

Run the Jenkins Pipeline Project for the Java Backend Project

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



	Declarative: Checkout SCM	Build	Test	Package
Average stage times: (Average full run time: ~28s)	261ms	3s	11s	12s
#1 Feb 27 05:27 No Changes	261ms	3s	11s	12s

Finally, click on Build Now and build your project.

Powerd by simplilearn

Caltech Center for Technology & Management Education

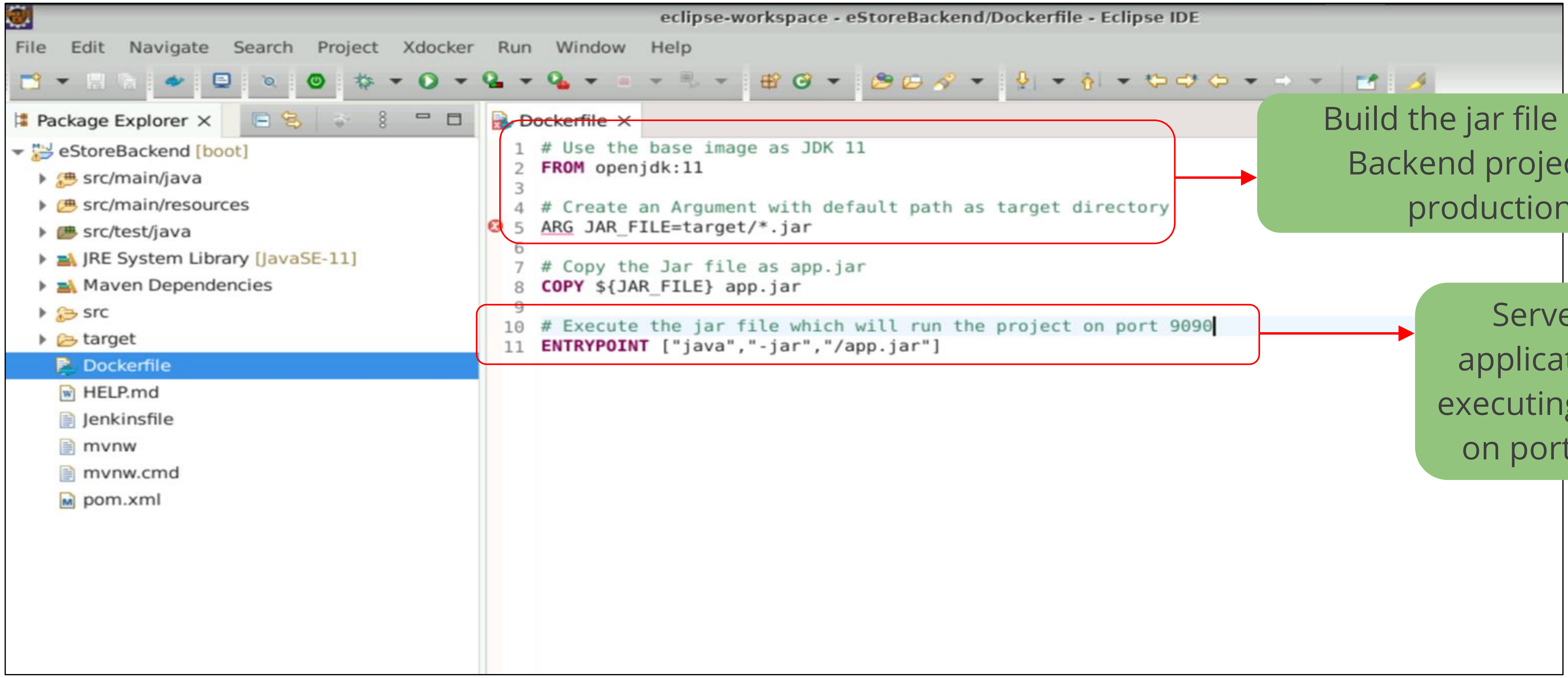
Caltech | Center for Technology & Management Education

Dockerise the Pipeline

Containerize the Java Backend Project

Configure Docker Using Dockerfile for Java Backend Project

Write the Dockerfile which will be used to **dockerize** Java for the Backend project.



```
# Use the base image as JDK 11
FROM openjdk:11

# Create an Argument with default path as target directory
ARG JAR_FILE=target/*.jar

# Copy the Jar file as app.jar
COPY ${JAR_FILE} app.jar

# Execute the jar file which will run the project on port 9090
ENTRYPOINT ["java","-jar","/app.jar"]
```

Build the jar file of Java Backend project for production.

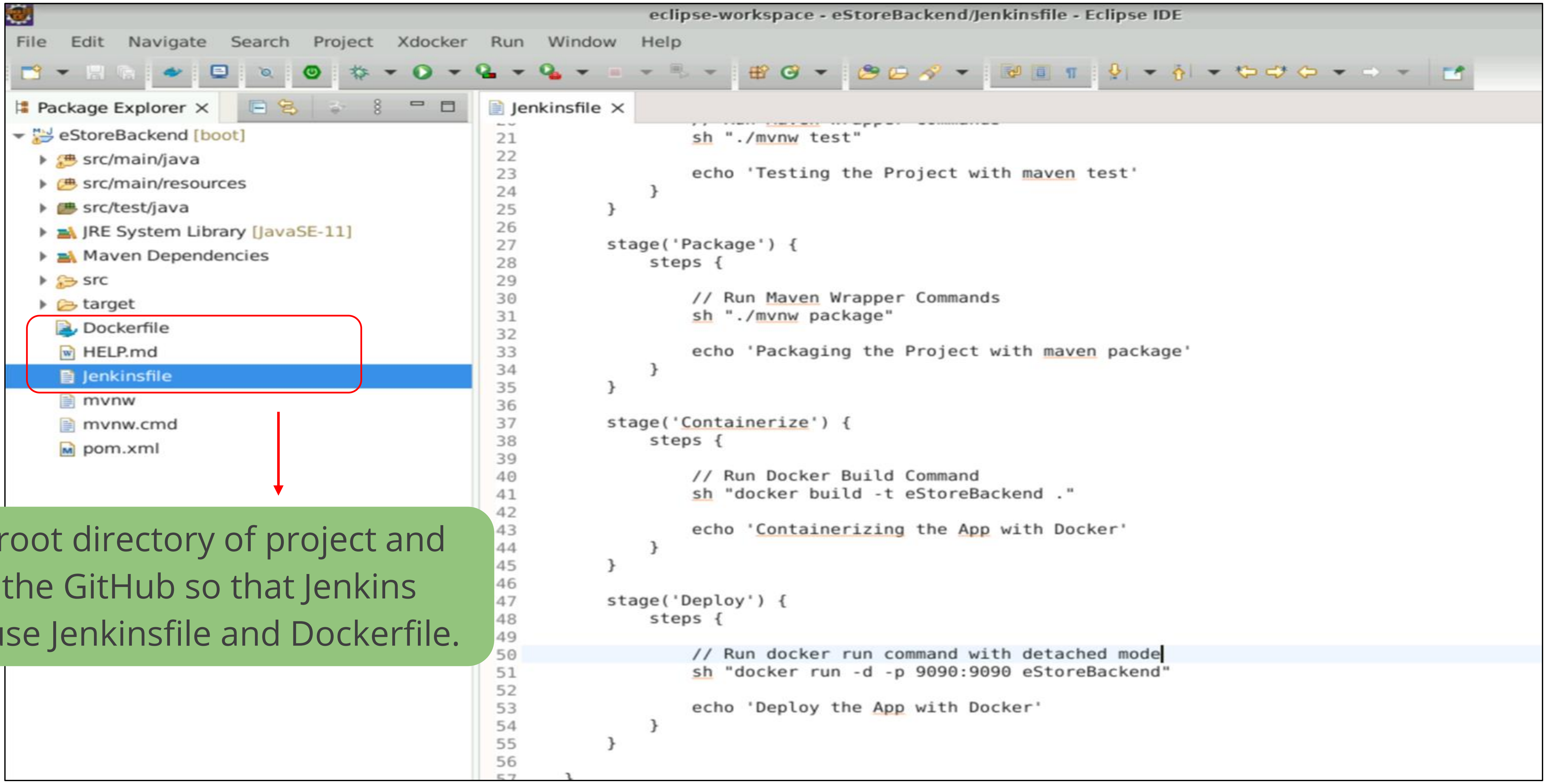
Serve the application by executing the Jar on port 9090.

Powerd by simplilearn

Caltech Center for Technology & Management Education

Configure Jenkins Pipeline Stages for Java Backend Project

Add the stages for containerizing the Java Backend that is **building and running** the container finally.



```
                sh "./mvnw test"

                echo 'Testing the Project with maven test'
            }
        }

        stage('Package') {
            steps {

                // Run Maven Wrapper Commands
                sh "./mvnw package"

                echo 'Packaging the Project with maven package'
            }
        }

        stage('Containerize') {
            steps {

                // Run Docker Build Command
                sh "docker build -t eStoreBackend ."

                echo 'Containerizing the App with Docker'
            }
        }

        stage('Deploy') {
            steps {

                // Run docker run command with detached mode
                sh "docker run -d -p 9090:9090 eStoreBackend"

                echo 'Deploy the App with Docker'
            }
        }

```

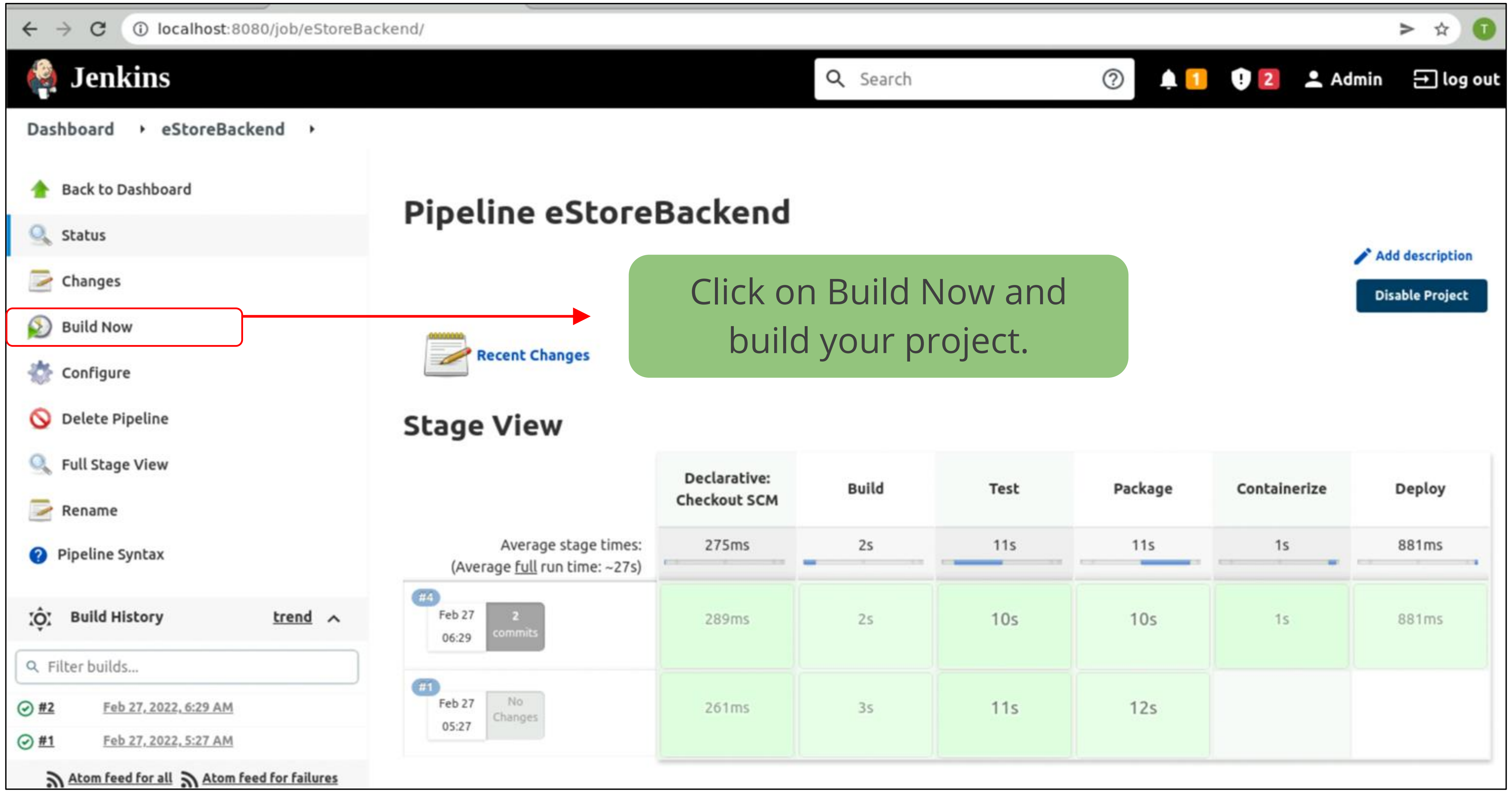
Create it in root directory of project and push it to the GitHub so that Jenkins project can use Jenkinsfile and Dockerfile.

Powerd by simplilearn

Caltech | Center for Technology & Management Education

Run the Jenkins Pipeline Project for Java Backend Project

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



Click on Build Now and build your project.

Pipeline eStoreBackend

Stage View

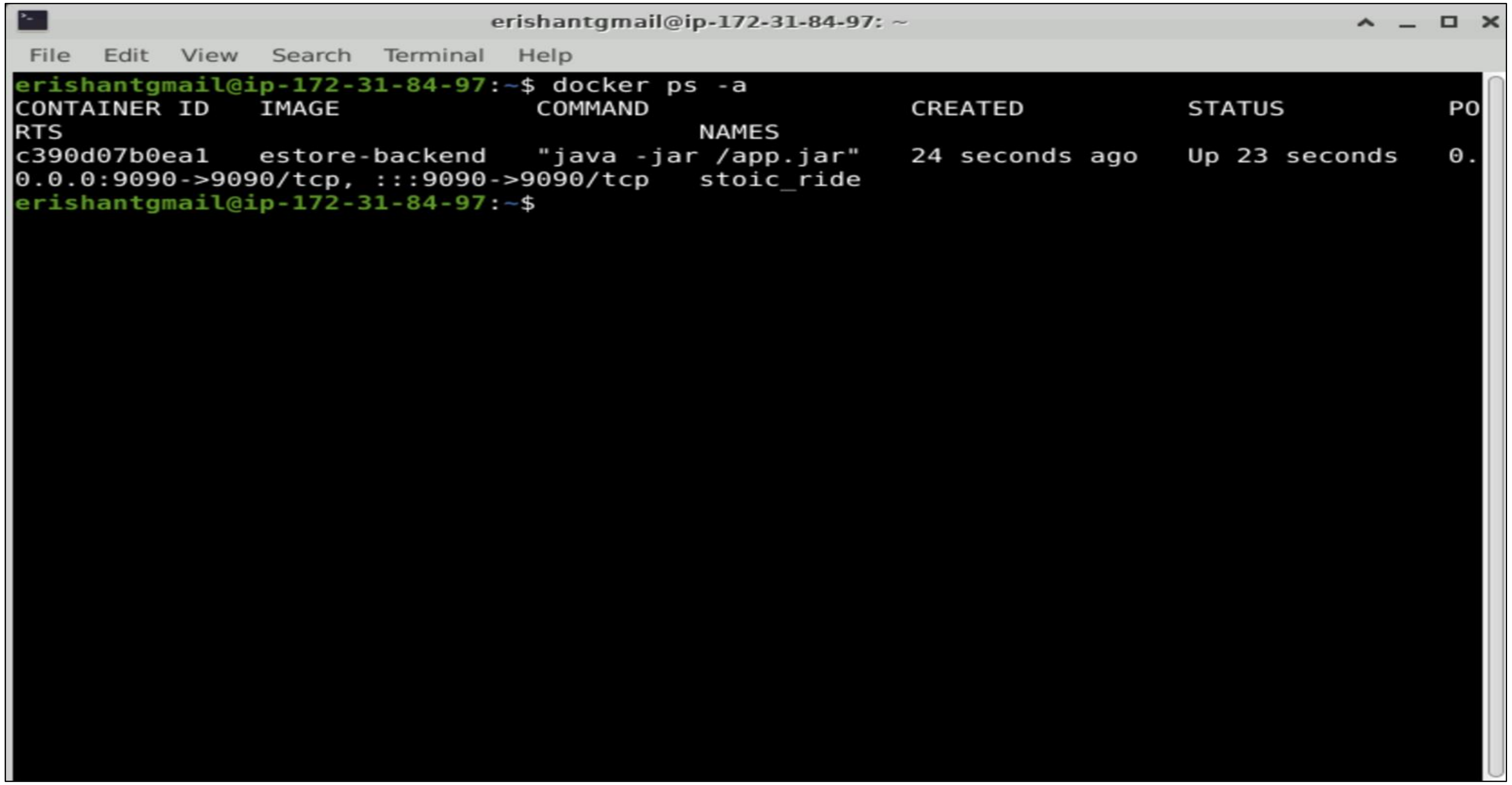
	Declarative: Checkout SCM	Build	Test	Package	Containerize	Deploy
Average stage times: (Average full run time: ~27s)	275ms	2s	11s	11s	1s	881ms
#4 Feb 27 06:29 2 commits	289ms	2s	10s	10s	1s	881ms
#1 Feb 27 05:27 No Changes	261ms	3s	11s	12s		

Build History:

- #2 Feb 27, 2022, 6:29 AM
- #1 Feb 27, 2022, 5:27 AM

Check Docker ps -a

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

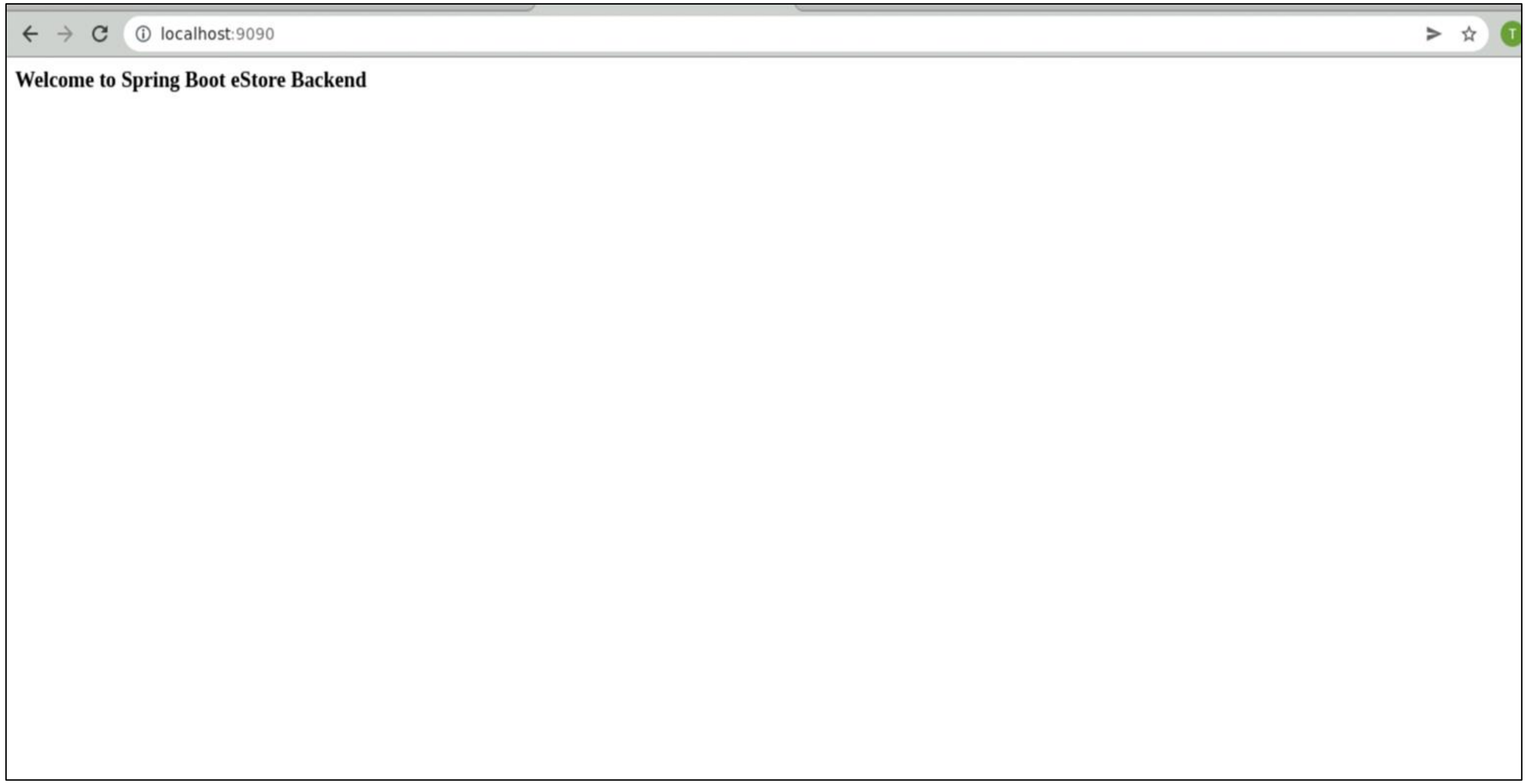

erishantgmail@ip-172-31-84-97: ~

File Edit View Search Terminal Help

```
erishantgmail@ip-172-31-84-97:~$ docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PO
RTS                                           NAMES
c390d07b0ea1   estore-backend    "java -jar /app.jar"     24 seconds ago   Up 23 seconds   0.
0.0.0:9090->9090/tcp, :::9090->9090/tcp   stoic_ride
erishantgmail@ip-172-31-84-97:~$
```

Check the Backend Project on Localhost

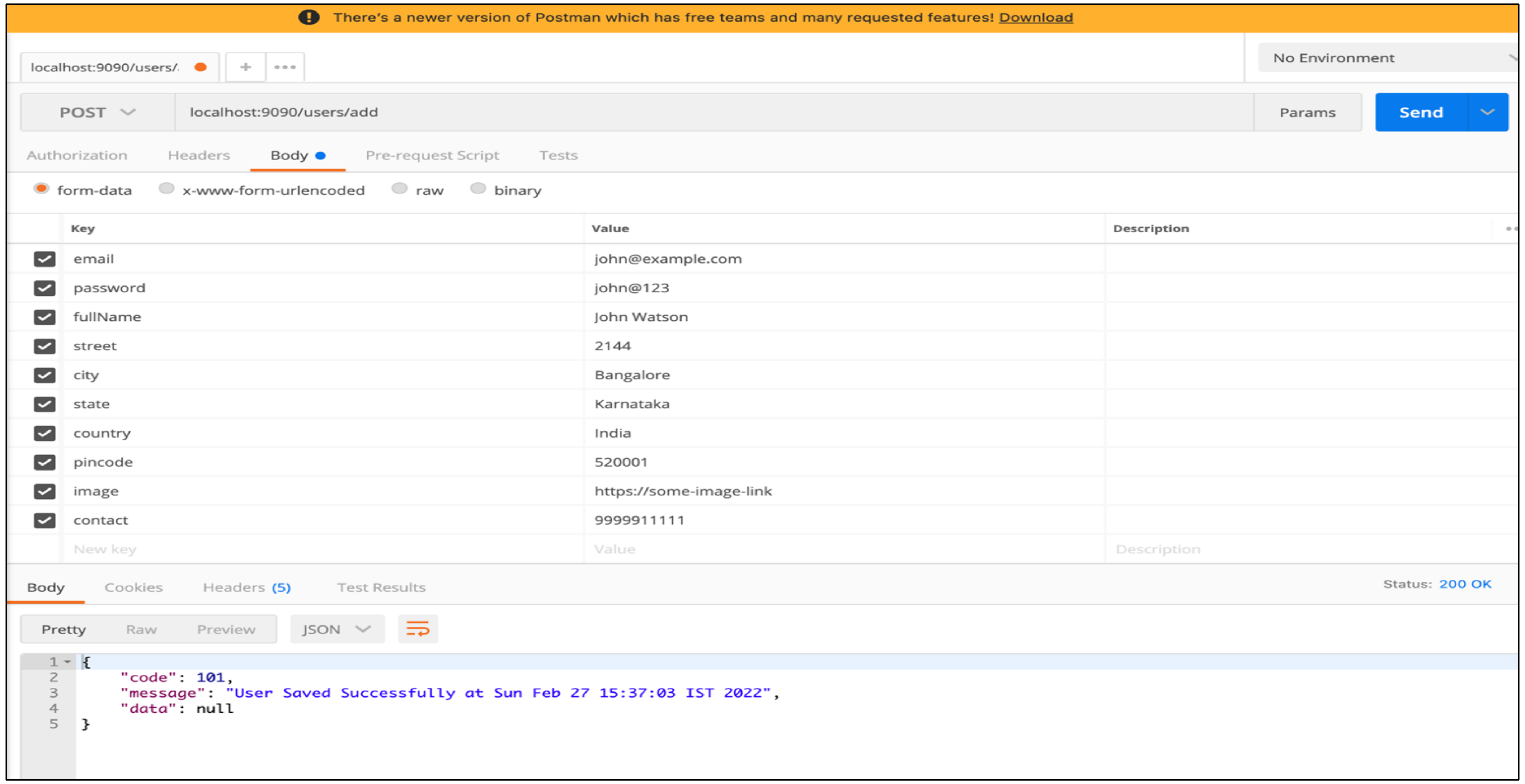
The Backend project is up and running in the docker container.


localhost:9090

Welcome to Spring Boot eStore Backend

Check Service on Postman

Validate if the project is up with services running.



There's a newer version of Postman which has free teams and many requested features! Download

localhost:9090/users/

No Environment

POST localhost:9090/users/add

[✓ form-data] x-www-form-urlencoded raw binary

	Key	Value	Description
✓	email	john@example.com	
✓	password	john@123	
✓	fullName	John Watson	
✓	street	2144	
✓	city	Bangalore	
✓	state	Karnataka	
✓	country	India	
✓	pincode	520001	
✓	image	https://some-image-link	
✓	contact	9999911111	
	New key	Value	Description

Status: 200 OK

```
{
    "code": 101,
    "message": "User Saved Successfully at Sun Feb 27 15:37:03 IST 2022",
    "data": null
}
```

Key Takeaways

- Jenkins Pipeline is built for Java Back end using Jenkinsfile.
- Java back end is dockerized using Dockerfile.
- Jenkins Pipeline is built to dockerize the Java back end.
- Service is validated using Postman.

Before the Next Class

You have successfully completed this session. For the next end-user module discussion, you should:

- Review AWS services
- Explore steps to create EC2 and S3 buckets
- Explore how to connect to EC2 instance
- Review Jenkins
- Review Docker commands

What’s Next?

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

- See how to create EC2 instance on AWS
- See how to work with storage buckets using S3
- Deploy Angular apps on EC2
- Use Jenkins and Docker as DevOps tools on EC2