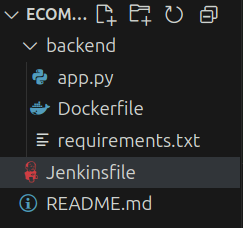
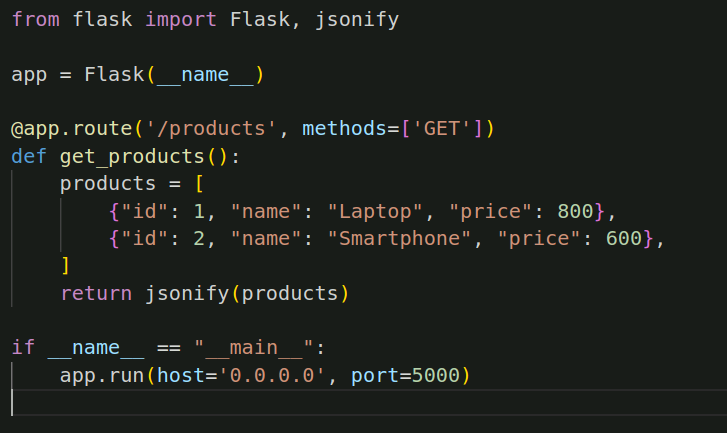
**E-commerce Web Application using Jenkins Pipeline ,docker**

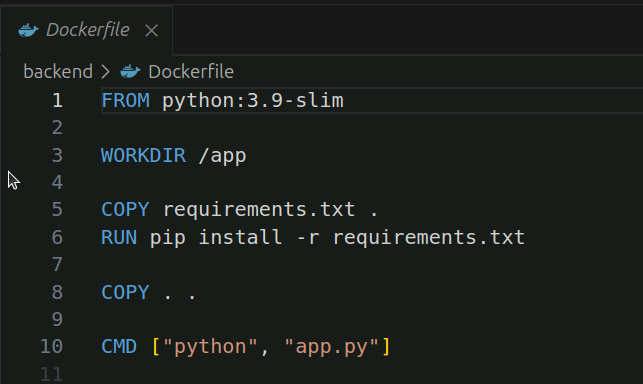
**Project structure**

****

**backend/app.py**

****

**Backend/Dockerfile**

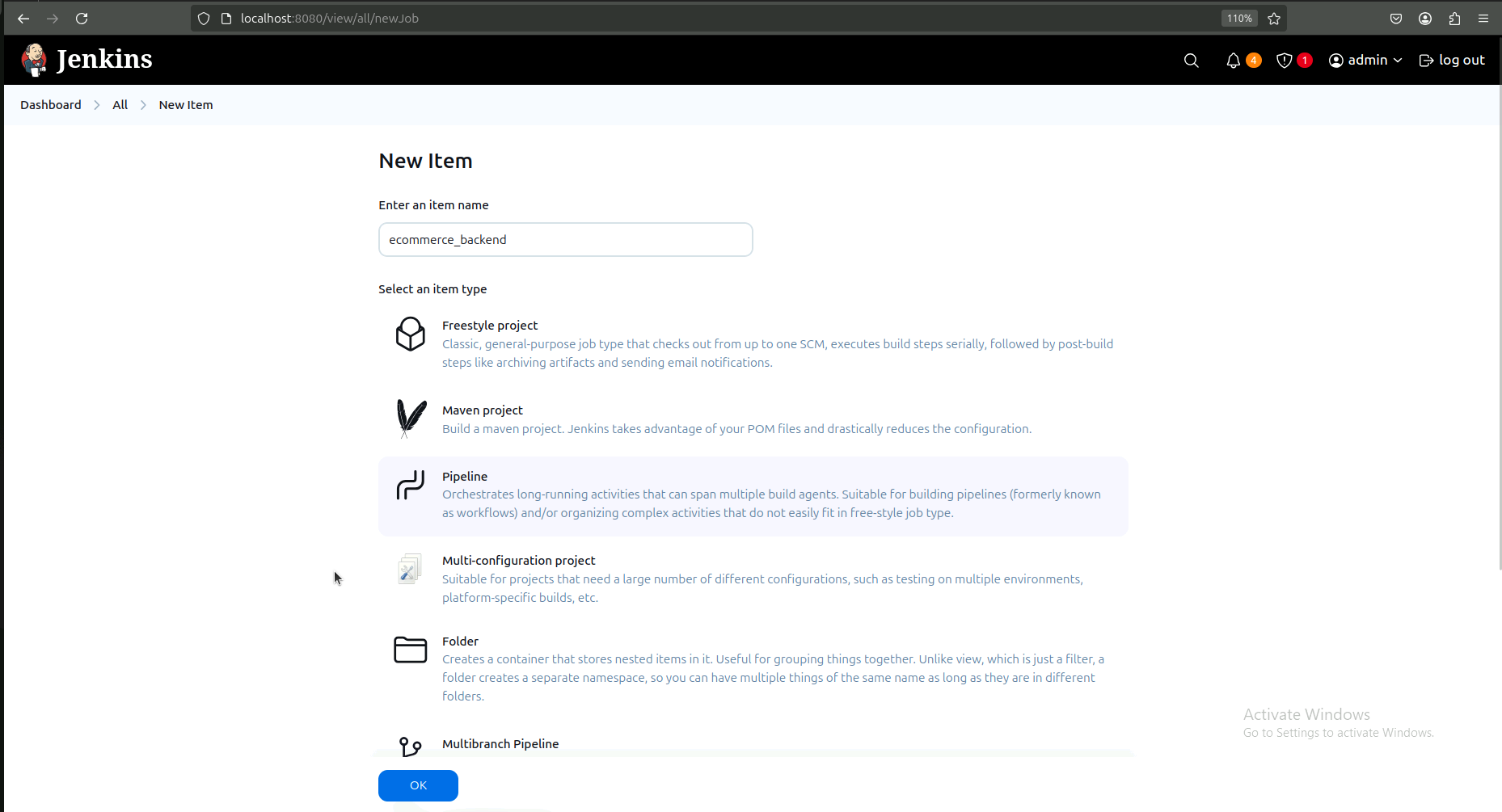
****

**Create the Jenkins code for Ecommerce website.**

**Set Up Jenkins Job**

**1. Create a Jenkins Job:**

Go to Jenkins Dashboard → New Item → Pipeline.

****

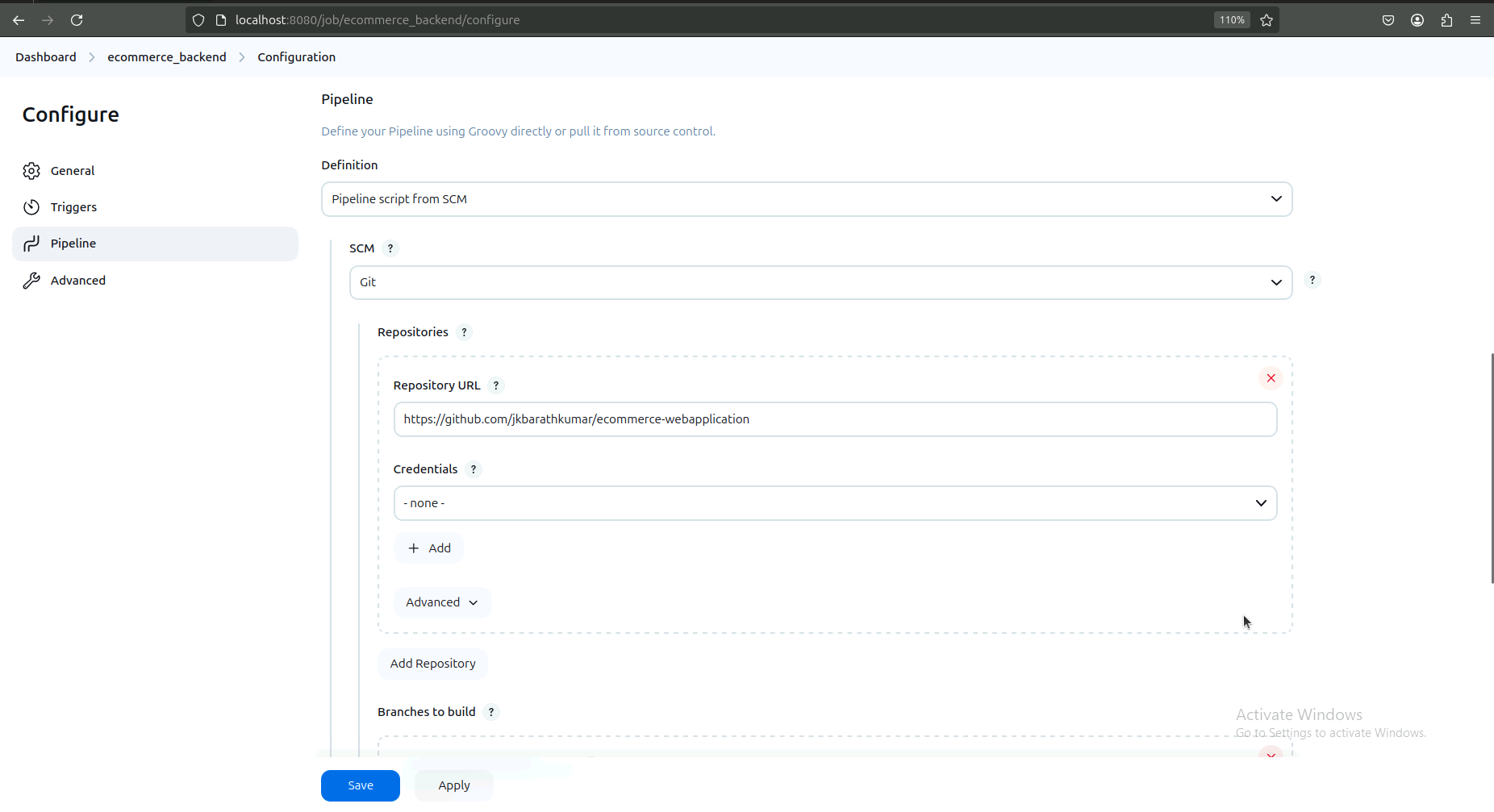
**2. Source Code Management:**

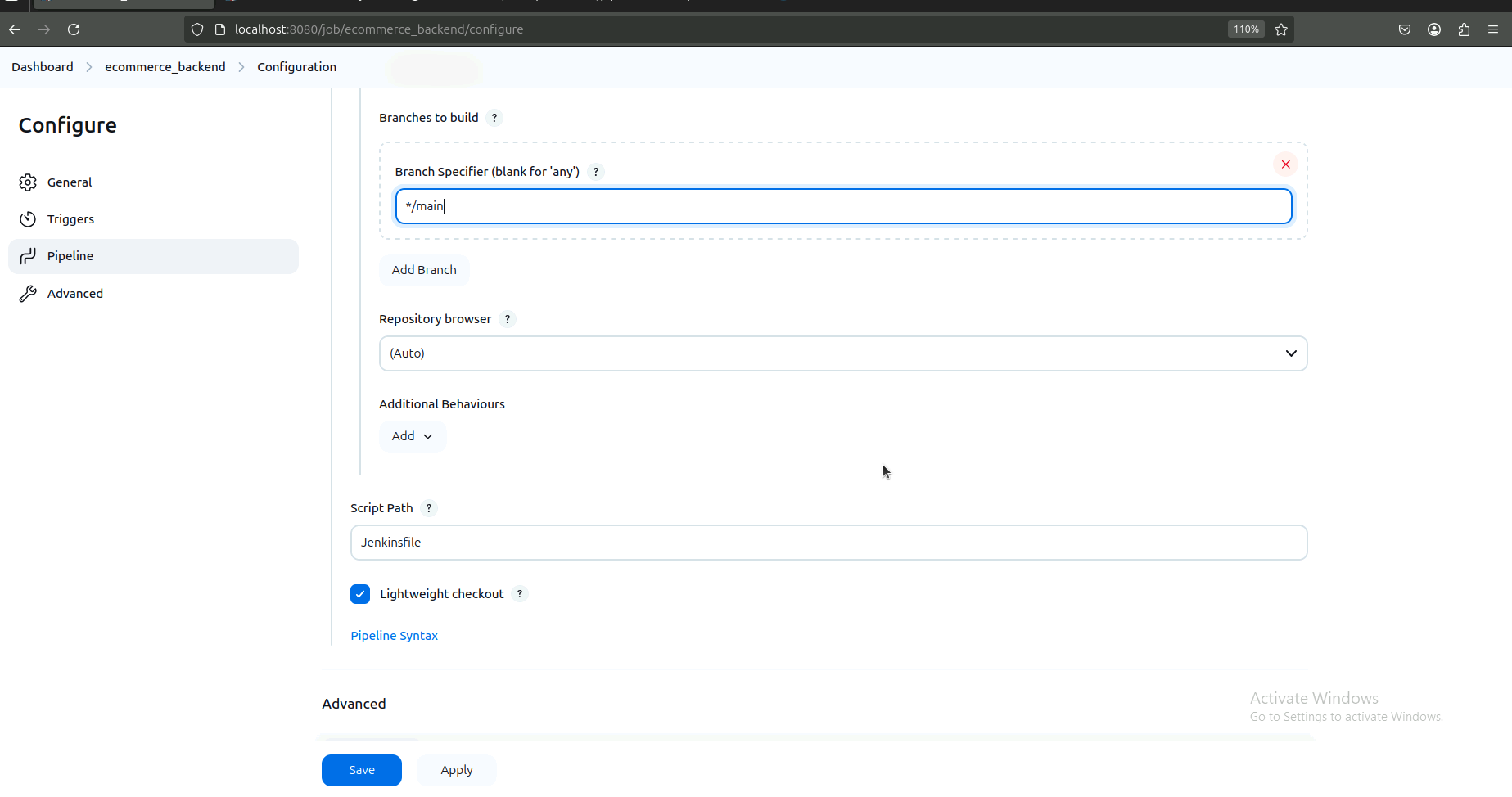
Choose Git → Provide Repository URL.

**3. Pipeline Definition:**

Select Pipeline script from SCM.

Choose Jenkinsfile from the repository.

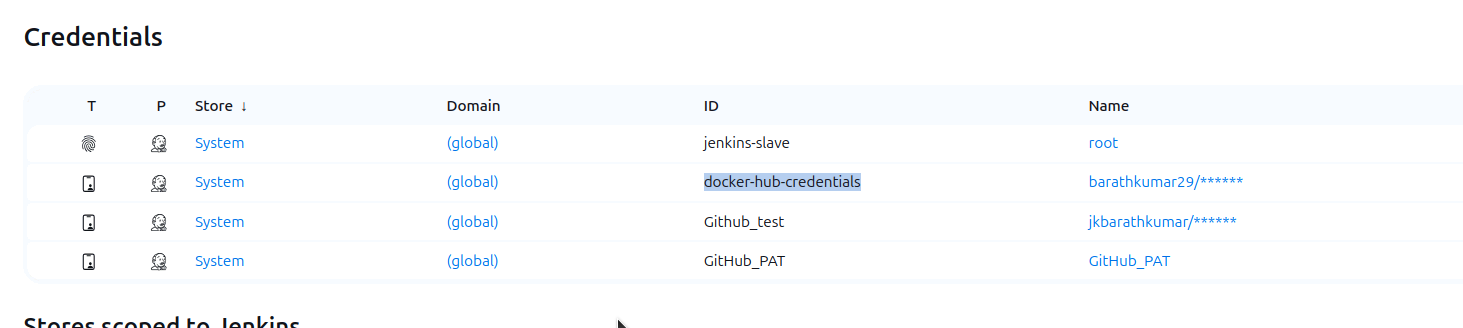
****

****

**4. Add Docker Credentials:**

Go to Jenkins → Manage Jenkins → Manage Credentials.

Add Docker Hub credentials with ID dockerhub-credentials.



**1) Write the Jenkinsfile that implements this pipeline, adhering to the specified stages and post-build conditions.Your goal is to create a Jenkins pipeline code in groove script with the following requirements:**

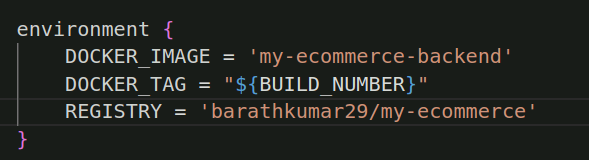
1. Environment Setup:

Define environment variables:

DOCKER\_IMAGE: The name of the Docker image should be my-ecommerce-backend.

DOCKER\_TAG: Dynamically set to the Jenkins build number.

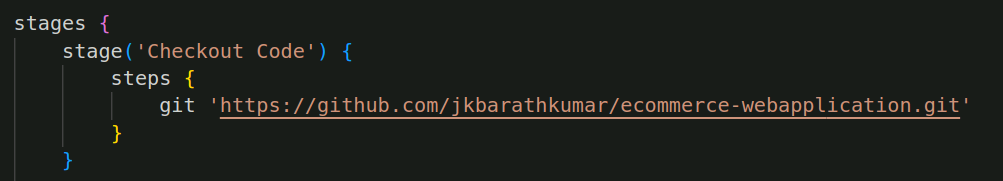
REGISTRY: The Docker Hub repository should be dockerhub\_user/my-ecommerce.

****

**2. Pipeline Stages:**

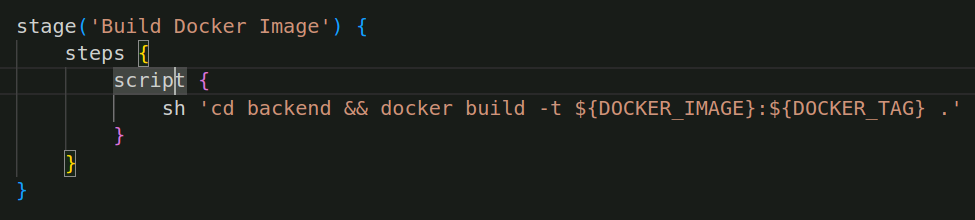
Checkout Code:

Clone the repository from GitHub: https://github.com/your-user/ecommerce-project.git.

****

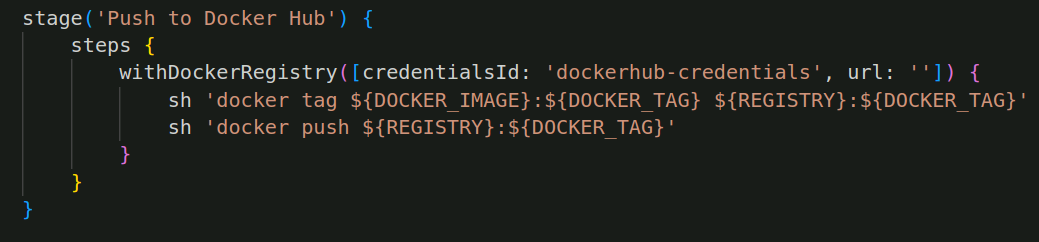
o Build Docker Image:

Inside the backend directory, build a Docker image tagged as ${DOCKER\_IMAGE}:${DOCKER\_TAG}.

****

o Push to Docker Hub:

Push the built Docker image to Docker Hub using the Jenkins credentials ID: dockerhub-credentials.

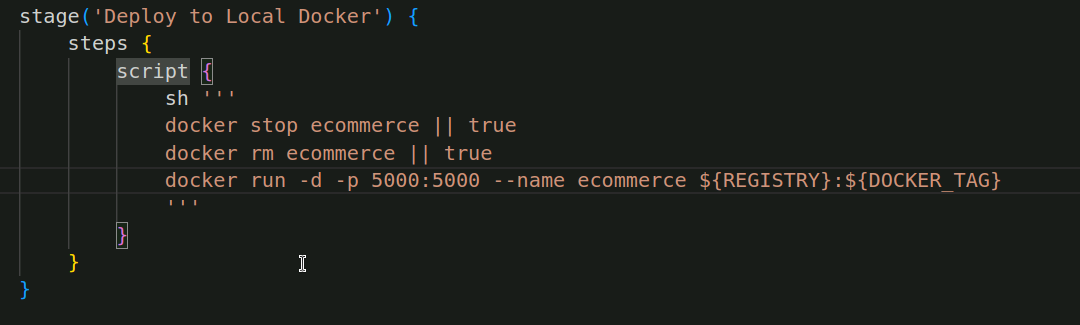
****

**o Deploy to Local Docker:**

Stop any running container named ecommerce, remove it if it exists, and run a new container.

The container should map port 5000 on the host to 5000 inside the container.

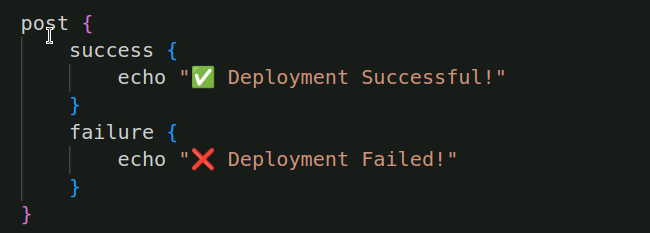
The new container should use the image ${DOCKER\_IMAGE}:${DOCKER\_TAG} and be named ecommerce.

****

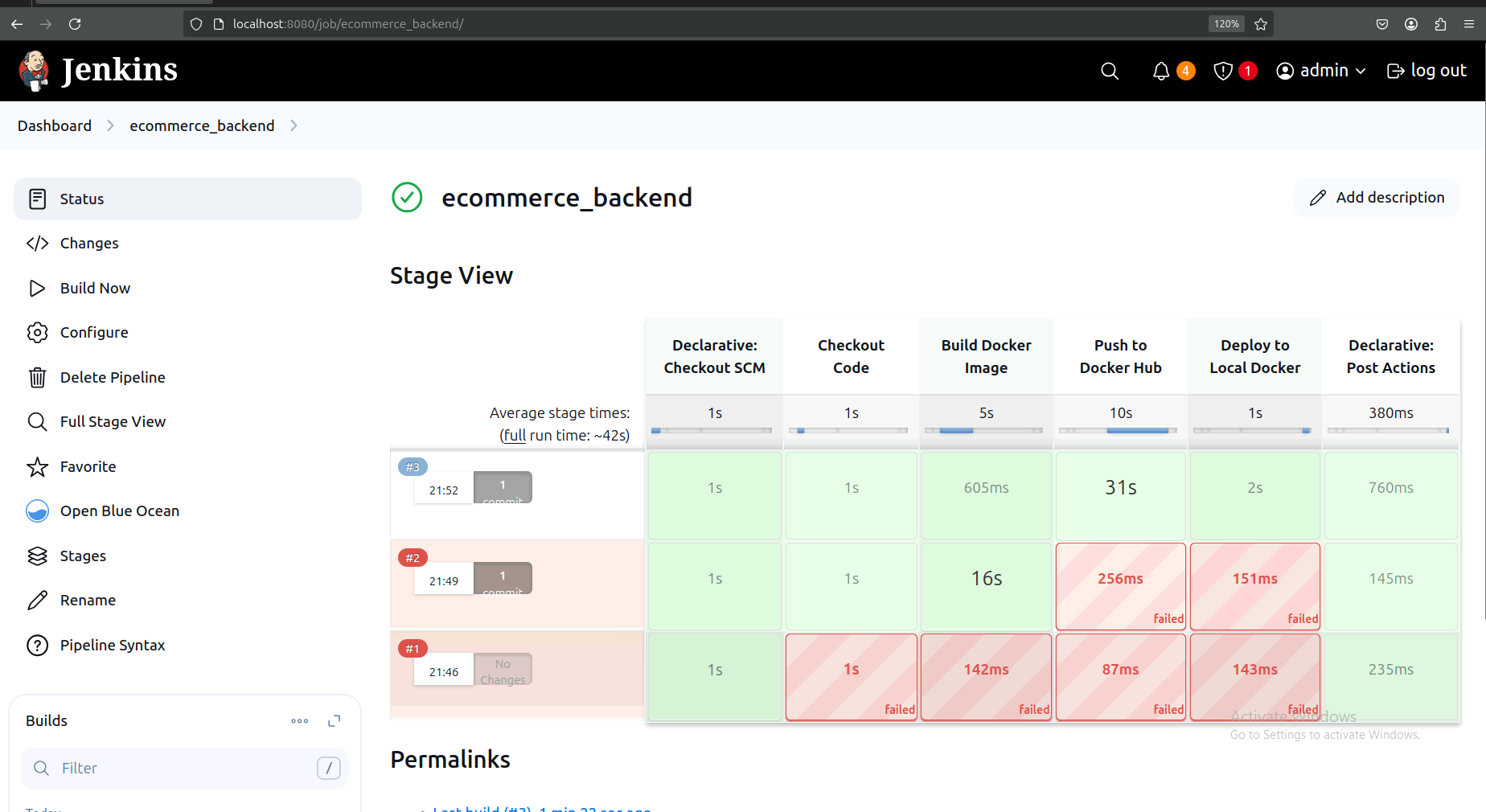
**3. Post-Build Actions:**

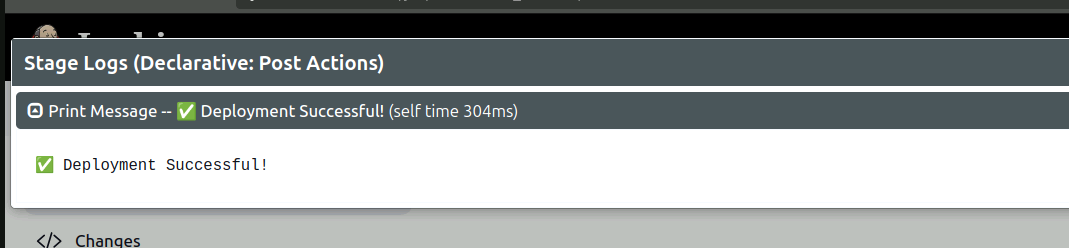
o On Success, print the message "✅ Deployment Successful!".

o On Failure, print the message "❌ Deployment Failed!"

****

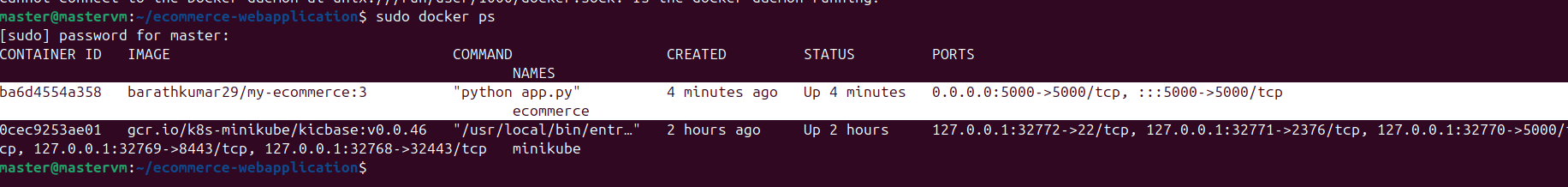
**Jenkins output:**

****

****

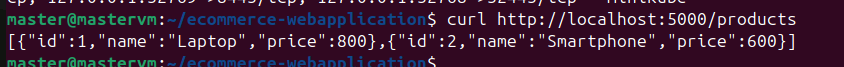
**To check the running container**

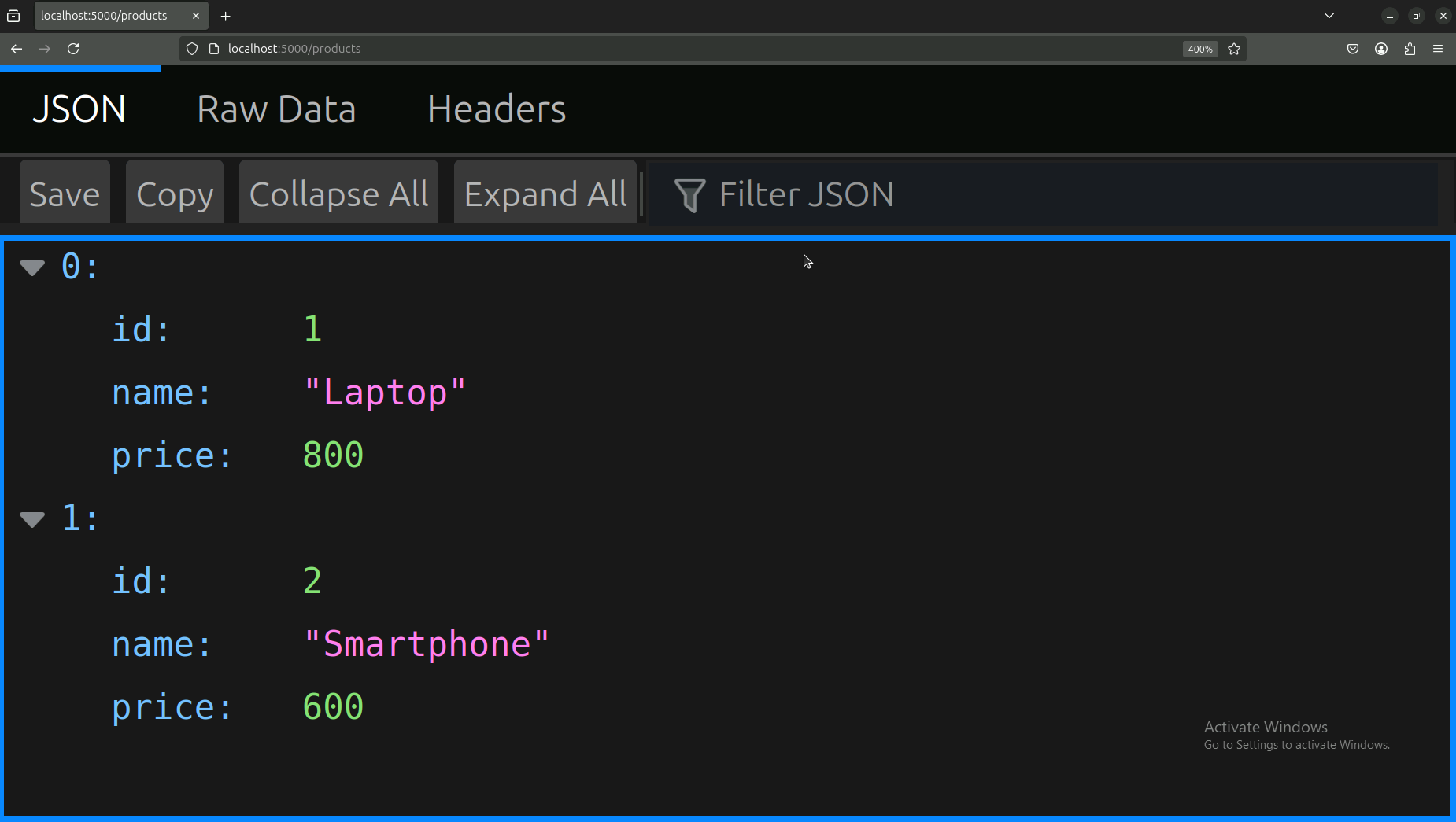
**· docker ps**



**Test the Application:**

**curl** [**http://localhost:5000/products**](http://localhost:5000/products)

****

****

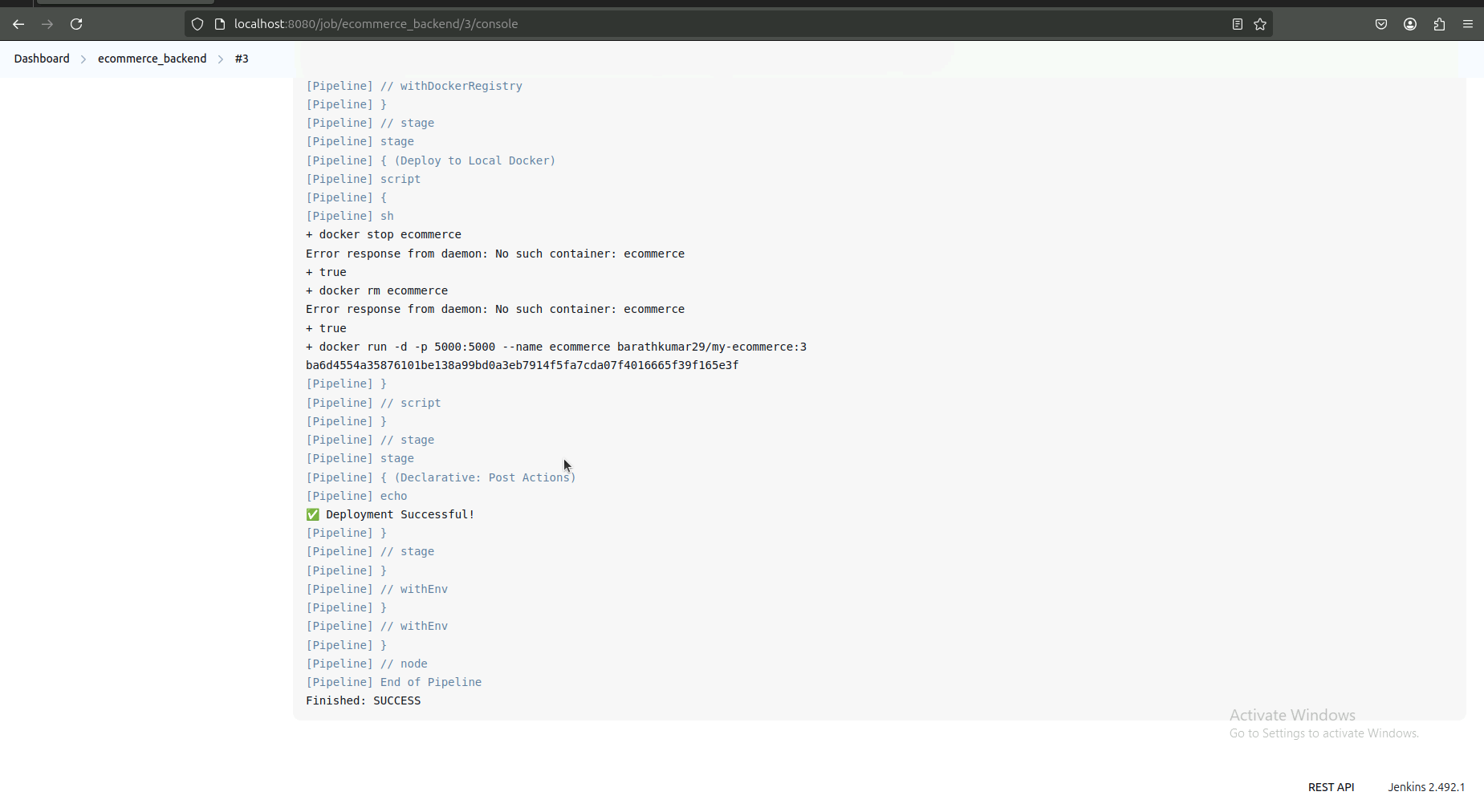
· After running the pipeline job, go to Jenkins Dashboard > Your Project > Last Build.

· Click on Console Output.

· A successful pipeline should display messages like:

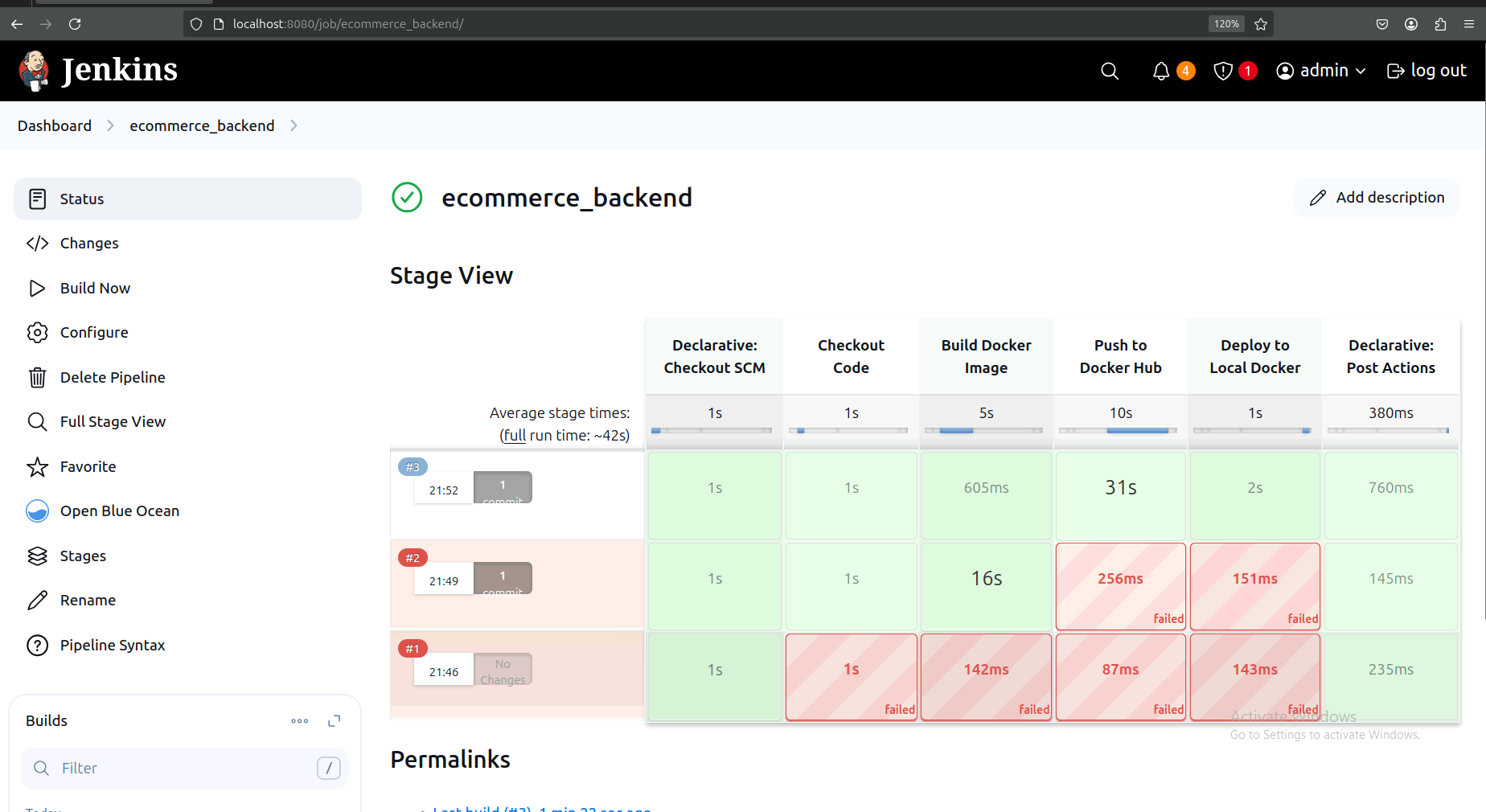
Deployment Successful!

Finished: SUCCESS

****

**Final outputs:**

Build Success:

****

Dockerhub:

