7. Deploy Angular/React/Java/Python or any other Application in Docker Container.

Step 1: Clone the Project Repository

1. **Open a Terminal**: Start a terminal session to enter the commands.
2. **Navigate to a Working Directory**:

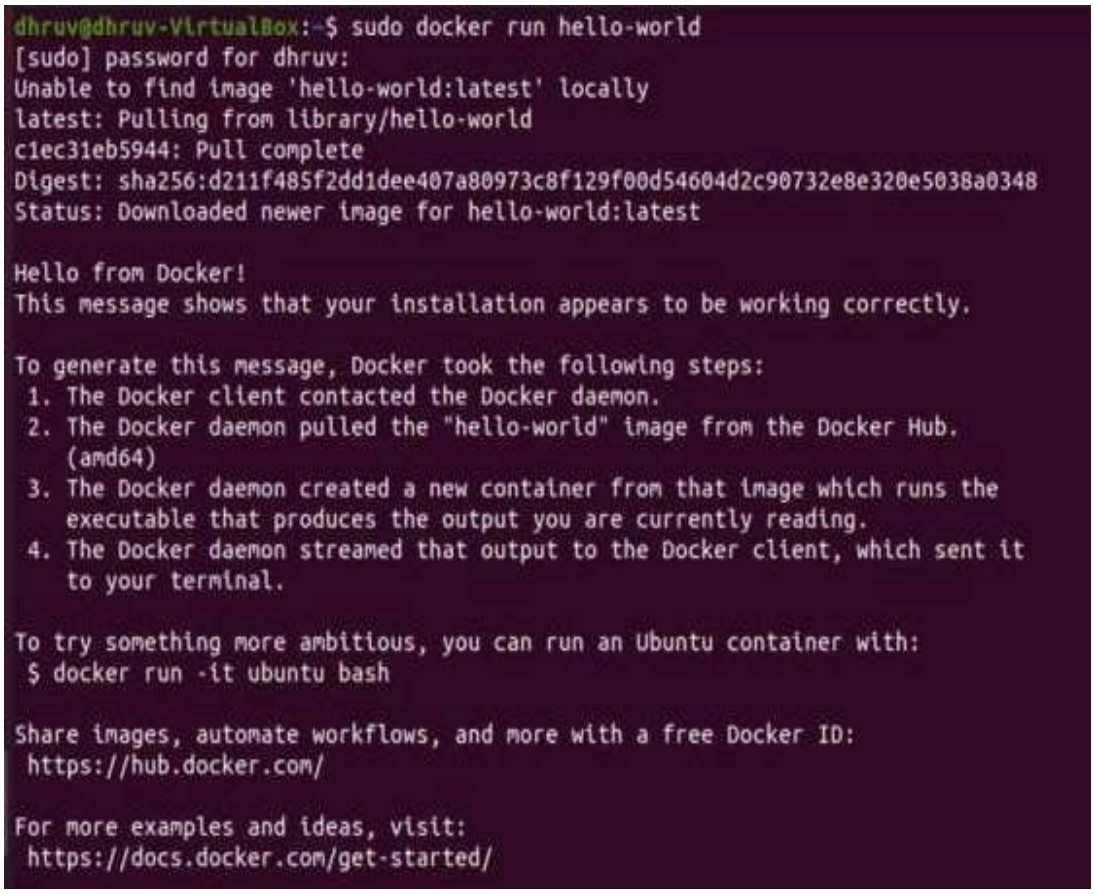
```
cd ~
mkdir my_projects
cd my_projects
```

3. **Clone the Repository**: Run the following command to clone the Git repository.

```
git clone <repository_url>
```

Example:

```
git clone https://github.com/sorani123/laptop-shopping-cart-ui.git
```



```
dhruv@dhruv-VirtualBox:~$ sudo docker run hello-world
[sudo] password for dhruv:
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

4. **Navigate into the Project Directory**:

```
cd laptop-shopping-cart-ui
```

Step 2: Create a Dockerfile

1. **Create a New File Named `Dockerfile`**:
   - In the project root directory, create a file named `Dockerfile`:

```
nano Dockerfile
```

2. **Write Dockerfile Instructions**:
   - Depending on the type of application, use one of the following configurations:
   - **For Node.js Applications (Angular/React)**:

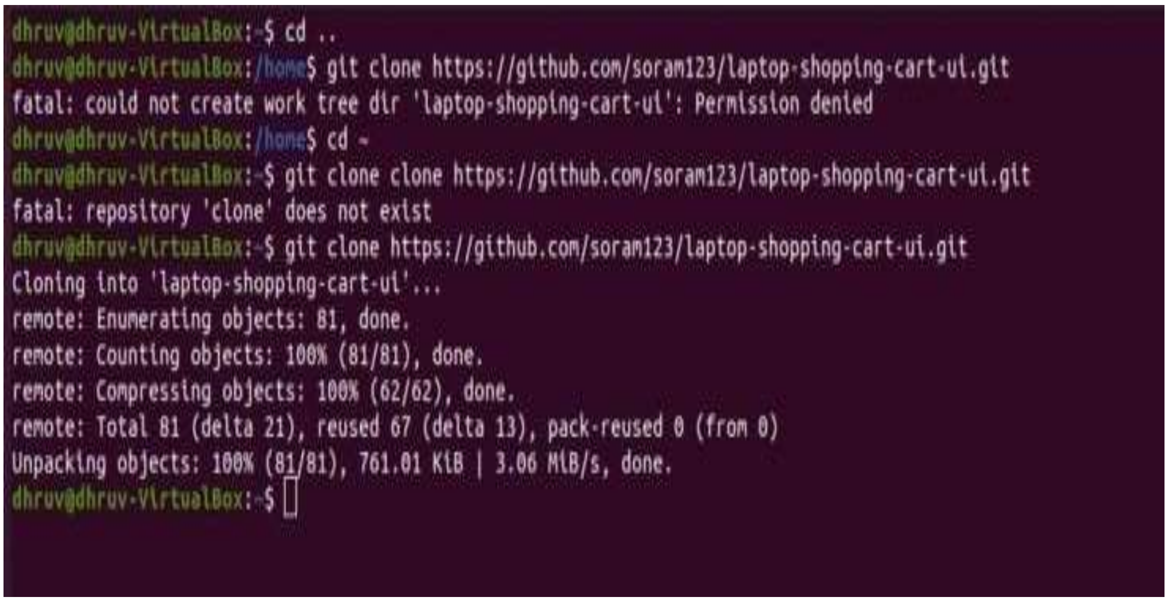
```
dockerfile

# Use the official Node.js image
FROM node:16-alpine

# Set the working directory in the container
WORKDIR /app

# Copy package.json and install dependencies
COPY package*.json ./
RUN npm install

# Copy the rest of the application code
COPY . .

# Build the app
RUN npm run build

# Serve the app using an HTTP server
RUN npm install -g serve
CMD ["serve", "-s", "build", "-l", "3000"]

# Expose the port the app runs on
EXPOSE 3000
```

   - **For Java Applications (Spring Boot)**:

```
dockerfile

FROM openjdk:17-jdk-slim
COPY target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
EXPOSE 8080
```

   - **For Python Applications (Flask)**:

```
dockerfile

FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
EXPOSE 5000
```



```
dhruv@dhruv-VirtualBox:~$ cd ..
dhruv@dhruv-VirtualBox:/home$ git clone https://github.com/soram123/laptop-shopping-cart-ui.git
fatal: could not create work tree dir 'laptop-shopping-cart-ui': Permission denied
dhruv@dhruv-VirtualBox:/home$ cd ~
dhruv@dhruv-VirtualBox:~$ git clone clone https://github.com/soram123/laptop-shopping-cart-ui.git
fatal: repository 'clone' does not exist
dhruv@dhruv-VirtualBox:~$ git clone https://github.com/soram123/laptop-shopping-cart-ui.git
Cloning into 'laptop-shopping-cart-ui'...
remote: Enumerating objects: 81, done.
remote: Counting objects: 100% (81/81), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 81 (delta 21), reused 67 (delta 13), pack-reused 0 (from 0)
Unpacking objects: 100% (81/81), 761.01 KiB | 3.06 MiB/s, done.
dhruv@dhruv-VirtualBox:~$ 
```

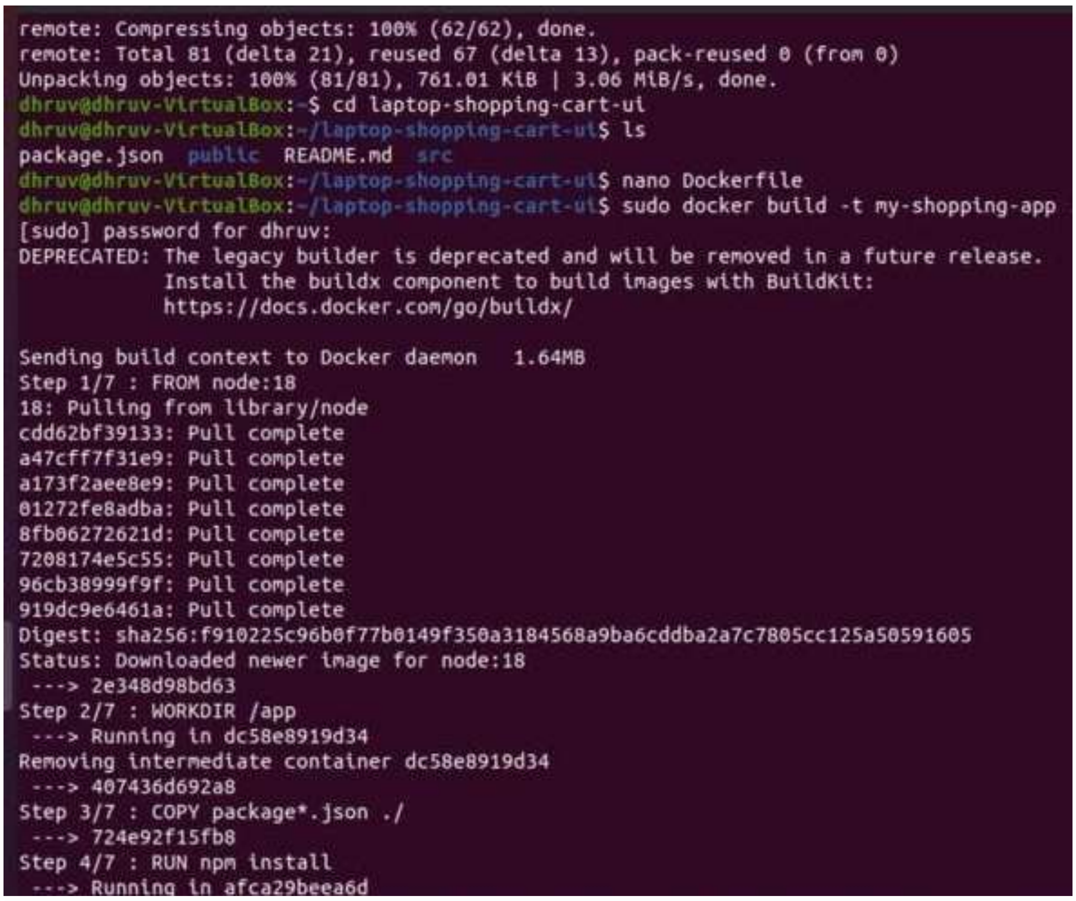
3. **Save and Exit the Dockerfile**:
   - For `nano`, press `CTRL + X`, then `Y`, and `Enter`.

Step 3: Build the Docker Image

1. **Build the Image**:

```
sudo docker build -t my-shopping-app .
```

   - `-t` is used to tag the image with a name (`my-shopping-app`).
2. **Verify Build Output**:
   - During the build, Docker will pull dependencies and prepare the environment as specified in the Dockerfile.



```
remote: Compressing objects: 100% (62/62), done.
remote: Total 81 (delta 21), reused 67 (delta 13), pack-reused 0 (from 0)
Unpacking objects: 100% (81/81), 761.01 KiB | 3.06 MiB/s, done.
dhruv@dhruv-VirtualBox:~$ cd laptop-shopping-cart-ui
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-ui$ ls
package.json  public  README.md  src
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-ui$ nano Dockerfile
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-ui$ sudo docker build -t my-shopping-app
[sudo] password for dhruv:
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon   1.64MB
Step 1/7 : FROM node:18
18: Pulling from library/node
cdd62bf39133: Pull complete
a47cff7f31e9: Pull complete
a173f2aee8e9: Pull complete
01272fe8adba: Pull complete
8fb06272621d: Pull complete
7208174e5c55: Pull complete
96cb38999f9f: Pull complete
919dc9e6461a: Pull complete
Digest: sha256:f910225c96b0f77b0149f350a3184568a9ba6cddba2a7c7805cc125a50591605
Status: Downloaded newer image for node:18
 ---> 2e348d98bd63
Step 2/7 : WORKDIR /app
 ---> Running in dc58e8919d34
Removing intermediate container dc58e8919d34
 ---> 407436d692a8
Step 3/7 : COPY package*.json ./
 ---> 724e92f15fb8
Step 4/7 : RUN npm install
 ---> Running in afca29beea6d
```

Step 4: Run the Docker Container

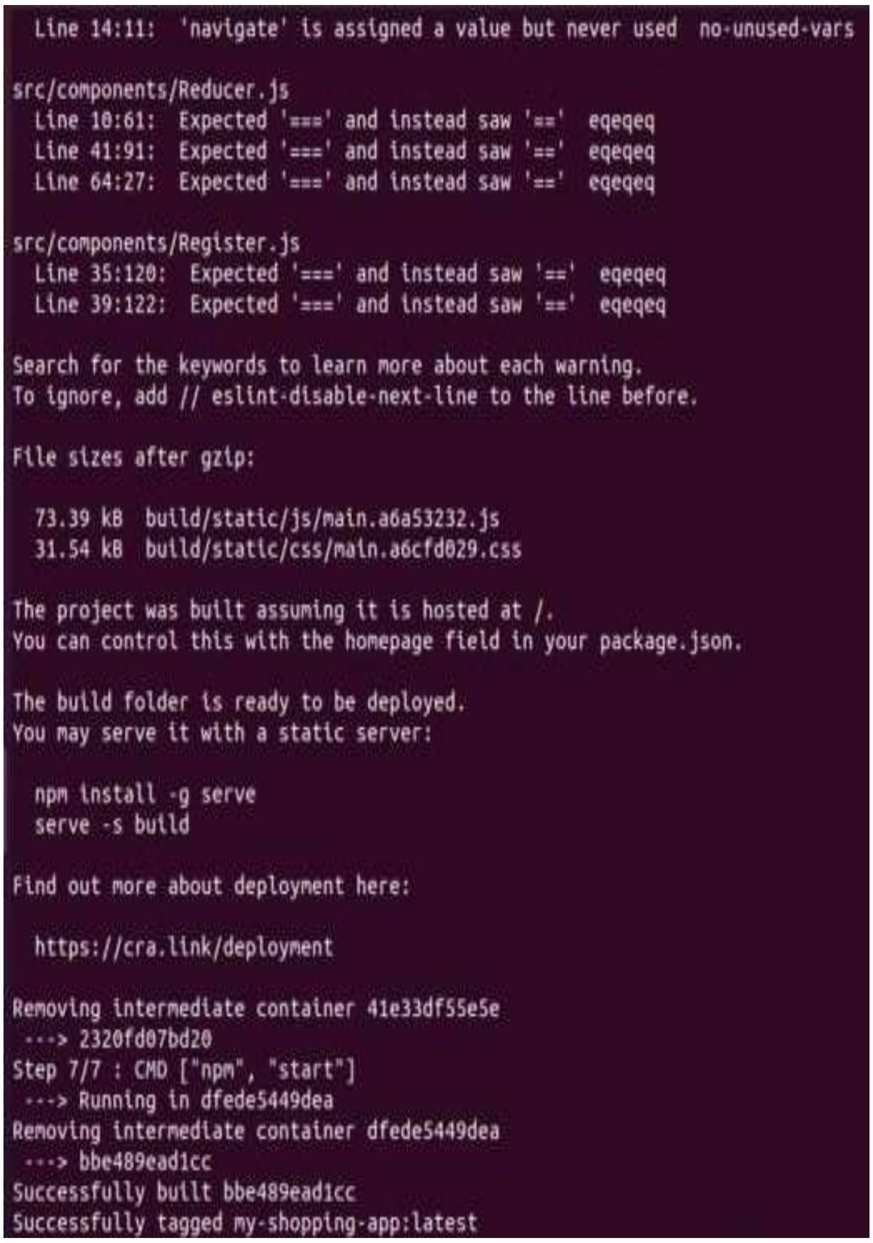
1. **Run the Container**:

   ```
   docker run -p <host_port>:<container_port> my-shopping-app
   ```

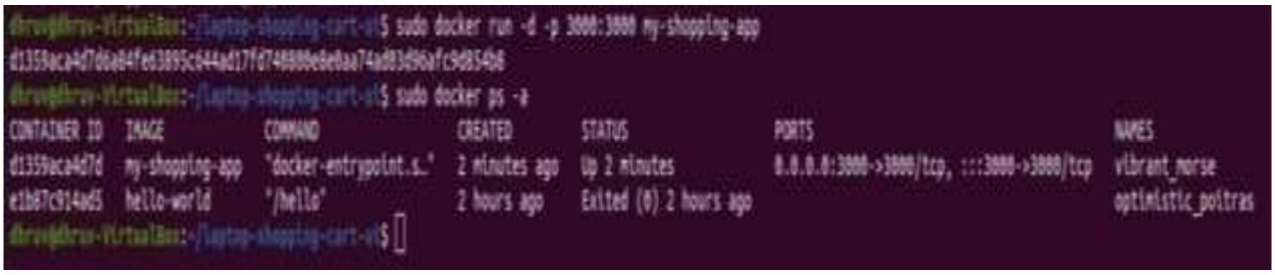
   - Replace `<host_port>` and `<container_port>` with the actual port, e.g., `3000:3000` for Angular/React.
2. **Example Command**:

   ```
   docker run -p 3000:3000 my-shopping-app
   ```

3. **Check for Application Access**:
   - Open a browser and go to `http://localhost:<host_port>`, e.g., `http://localhost:3000`.



```
  Line 14:11:  'navigate' is assigned a value but never used  no-unused-vars

src/components/Reducer.js
  Line 10:61:  Expected '===' and instead saw '=='  eqeqeq
  Line 41:91:  Expected '===' and instead saw '=='  eqeqeq
  Line 64:27:  Expected '===' and instead saw '=='  eqeqeq

src/components/Register.js
  Line 35:120:  Expected '===' and instead saw '=='  eqeqeq
  Line 39:122:  Expected '===' and instead saw '=='  eqeqeq

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

File sizes after gzip:

  73.39 kB  build/static/js/main.a6a53232.js
  31.54 kB  build/static/css/main.a6cfd029.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  https://cra.link/deployment

Removing intermediate container 41e33df55e5e
 ---> 2320fd07bd20
Step 7/7 : CMD ["npm", "start"]
 ---> Running in dfede5449dea
Removing intermediate container dfede5449dea
 ---> bbe489ead1cc
Successfully built bbe489ead1cc
Successfully tagged my-shopping-app:latest
```



```
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-v1$ sudo docker run -d -p 3000:3000 my-shopping-app
d1359aca4d7d6a04fe63895c644ad17fd748800e0e0aa74a083d96afc9d8540d
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-v1$ sudo docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                    PORTS                                       NAMES
d1359aca4d7d   my-shopping-app   "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes              0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   vibrant_morse
e1b07c914ad5   hello-world       "/hello"                 2 hours ago      Exited (0) 2 hours ago                                                optimistic_poitras
dhruv@dhruv-VirtualBox:~/laptop-shopping-cart-v1$ 
```

Step 5: Troubleshooting and Logs

1. **View Running Containers**:

```
docker ps
```

- This lists all active containers with their IDs, names, and other details.

Image Space: Insert a screenshot showing `docker ps` output.

2. **View Container Logs**:

```
docker logs <container_id>
```

- Replace `<container_id>` with the ID of your container.

3. **Access the Container's Shell (Optional)**:

```
docker exec -it <container_id> /bin/sh
```

- This command lets you interact with the container's filesystem for further troubleshooting.

Step 6: Stopping and Removing Containers

1. **Stop the Container**:

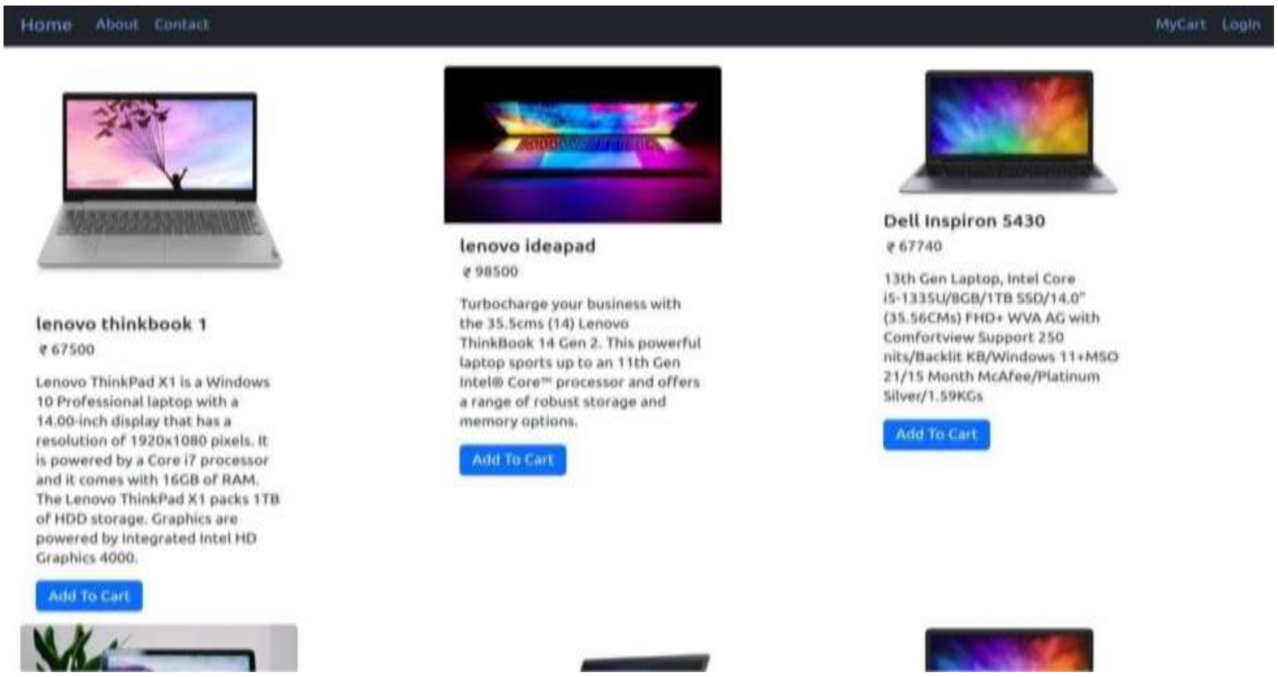
```
docker stop <container_id>
```

2. **Remove the Container**:

```
docker rm <container_id>
```

3. **Remove the Docker Image** (if no longer needed):

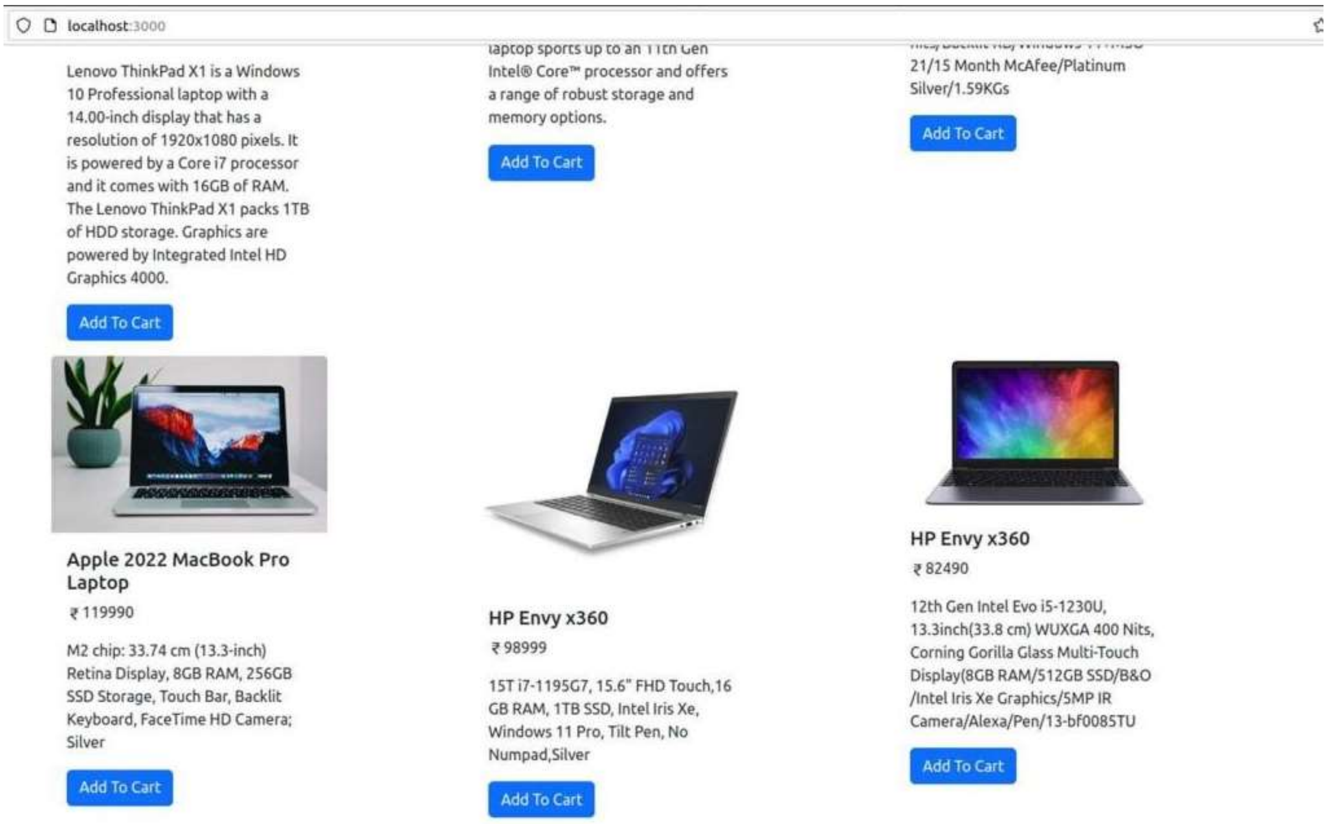
```
docker rmi my-shopping-app
```



lenovo thinkbook 1
₹ 67500
Lenovo ThinkPad X1 is a Windows 10 Professional laptop with a 14.00-inch display that has a resolution of 1920x1080 pixels. It is powered by a Core i7 processor and it comes with 16GB of RAM. The Lenovo ThinkPad X1 packs 1TB of HDD storage. Graphics are powered by Integrated Intel HD Graphics 4000.

lenovo ideapad
₹ 98500
Turbocharge your business with the 35.5cms (14) Lenovo ThinkBook 14 Gen 2. This powerful laptop sports up to an 11th Gen Intel® Core™ processor and offers a range of robust storage and memory options.

Dell Inspiron 5430
₹ 67740
13th Gen Laptop, Intel Core i5-1335U/8GB/1TB SSD/14.0" (35.56CMs) FHD+ WVA AG with Comfortview Support 250 nits/Backlit KB/Windows 11+MSO 21/15 Month McAfee/Platinum Silver/1.59KGs



Lenovo ThinkPad X1 is a Windows 10 Professional laptop with a 14.00-inch display that has a resolution of 1920x1080 pixels. It is powered by a Core i7 processor and it comes with 16GB of RAM. The Lenovo ThinkPad X1 packs 1TB of HDD storage. Graphics are powered by Integrated Intel HD Graphics 4000.

laptop sports up to an 11th Gen Intel® Core™ processor and offers a range of robust storage and memory options.

21/15 Month McAfee/Platinum Silver/1.59KGs

Apple 2022 MacBook Pro Laptop
₹ 119990
M2 chip: 33.74 cm (13.3-inch) Retina Display, 8GB RAM, 256GB SSD Storage, Touch Bar, Backlit Keyboard, FaceTime HD Camera; Silver

HP Envy x360
₹ 98999
15T i7-1195G7, 15.6" FHD Touch,16 GB RAM, 1TB SSD, Intel Iris Xe, Windows 11 Pro, Tilt Pen, No Numpad,Silver

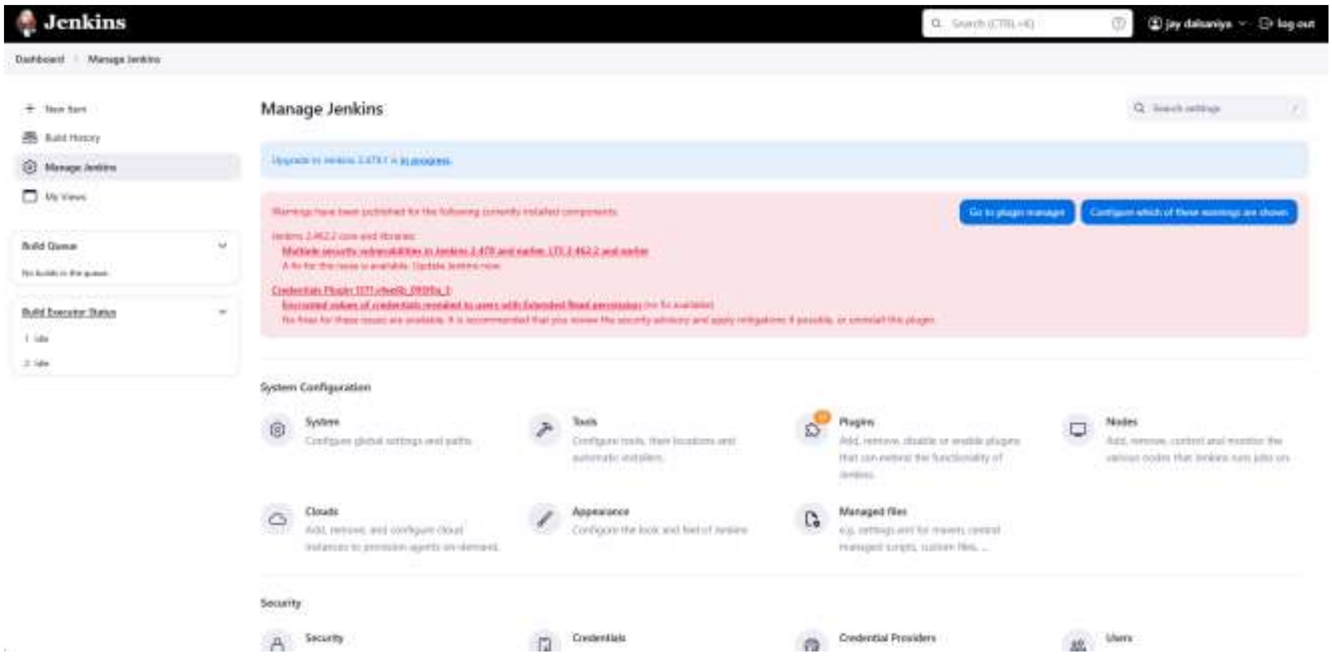
HP Envy x360
₹ 82490
12th Gen Intel Evo i5-1230U, 13.3inch(33.8 cm) WUXGA 400 Nits, Corning Gorilla Glass Multi-Touch Display(8GB RAM/512GB SSD/B&O /Intel Iris Xe Graphics/5MP IR Camera/Alexa/Pen/13-bf0085TU

8. Use Jenkins to set up a distributed pipeline that will compile and test a Maven project on two different slave nodes respectively.

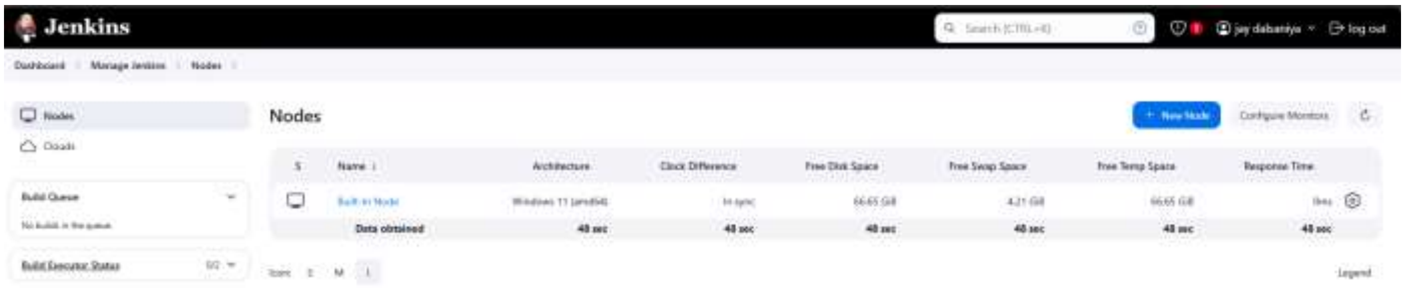
Step 1: Open Jenkins

1. Open a web browser.
2. Go to `http://localhost:8000` to access the Jenkins dashboard.

Step 2: Configure Slave Nodes



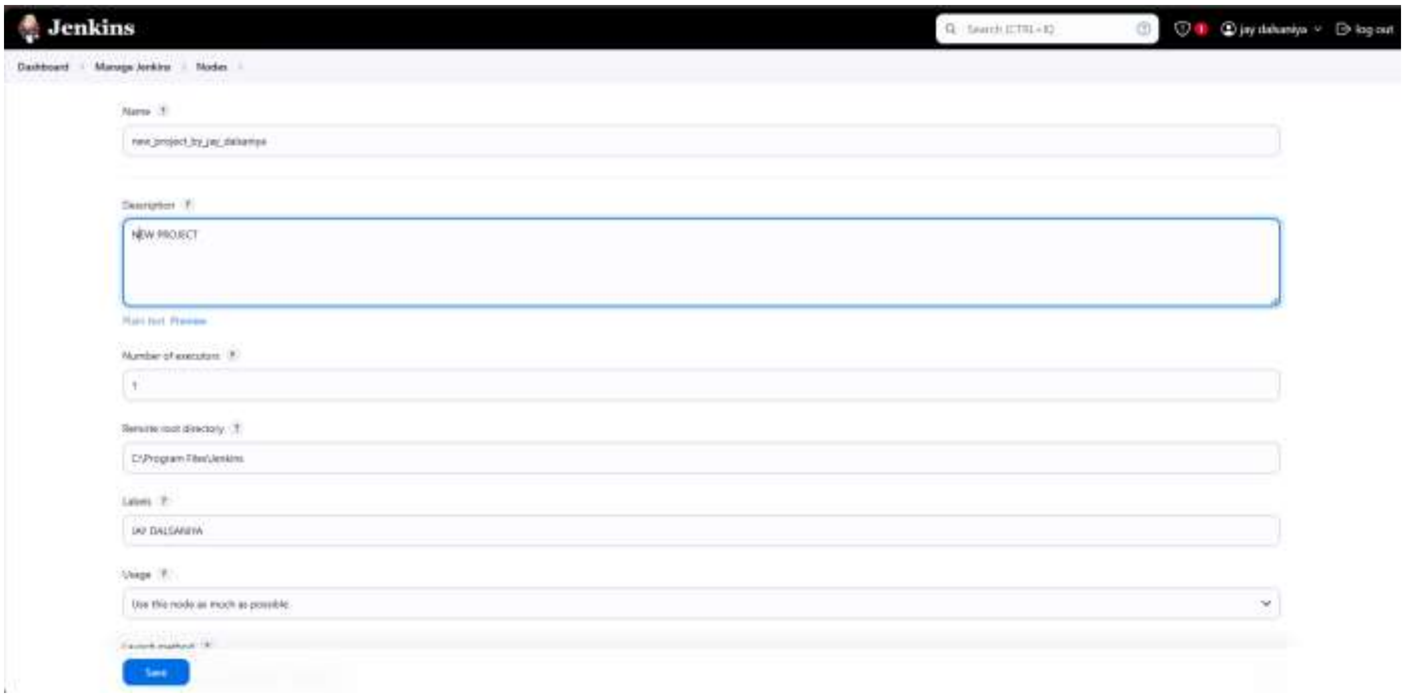
1. From the Jenkins dashboard, navigate to **Manage Jenkins**.
2. Select **Manage Nodes and Clouds**.



3. Click on **New Node** to create a new agent.
   - **Node Name**: Enter a unique name for the agent (e.g., `new_project_by_jay_dalsaniya`).
   - **Number of Executors**: Set to `1`.

- **Remote Root Directory**: Specify the directory where the agent will be located (e.g., `C:\Program Files\Jenkins\slave1`).
- **Labels**: Add a label (e.g., `JAY DALSANIYA`) to identify the node.
- **Usage**: Choose **Use this node as much as possible**.
- **Launch Method**: Set to **Launch agent by connecting it to the controller**.
- **Availability**: Choose **Keep this agent online as much as possible**.

4. Click **Save** to create the node.
5. Repeat steps 1-4 to create a second slave node, if not already configured.



Name: new_project_by_jay_dalsaniya
Description: NEW PROJECT
Number of executors: 1
Remote root directory: C:\Program Files\Jenkins
Labels: JAY DALSANIYA
Usage: Use this node as much as possible
Launch method

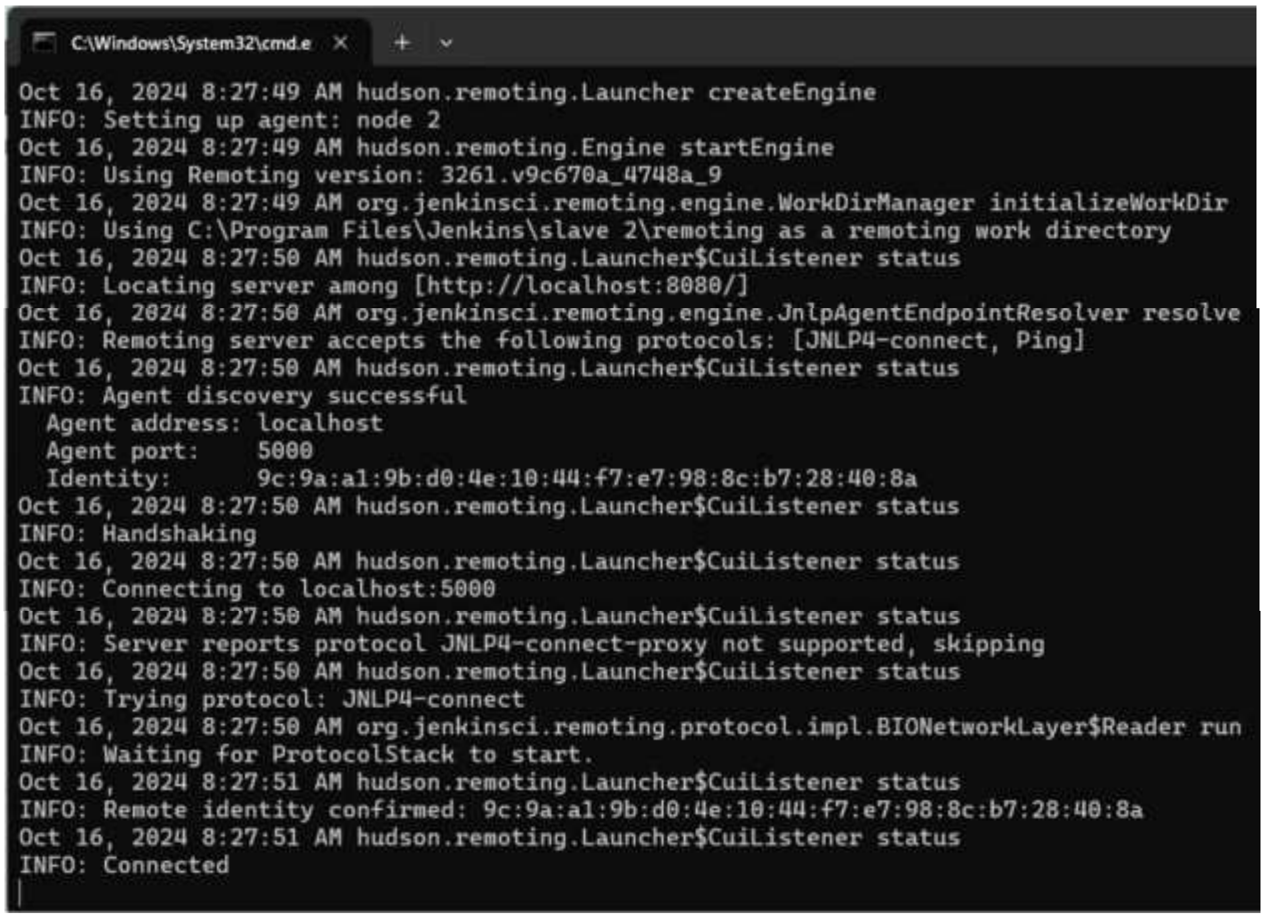
Step 3: Launch and Connect the Slave Node

1. After configuring the slave node, open a command prompt on the machine where the slave node is set up.
2. Run the command to launch the Jenkins agent. The output should indicate the connection status.

   Example output:

   ```
   INFO: Setting up agent: node 2
   INFO: Agent discovery successful
   INFO: Agent address localhost
   INFO: Handshaking
   INFO: Connected
   ```

   This indicates that the slave node has successfully connected to the Jenkins master.



```
Oct 16, 2024 8:27:49 AM hudson.remoting.Launcher createEngine
INFO: Setting up agent: node 2
Oct 16, 2024 8:27:49 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3261.v9c670a_4748a_9
Oct 16, 2024 8:27:49 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Program Files\Jenkins\slave 2\remoting as a remoting work directory
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Locating server among [http://localhost:8080/]
Oct 16, 2024 8:27:50 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Agent discovery successful
  Agent address: localhost
  Agent port:    5000
  Identity:      9c:9a:a1:9b:d0:4e:10:44:f7:e7:98:8c:b7:28:40:8a
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Handshaking
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Connecting to localhost:5000
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Server reports protocol JNLP4-connect-proxy not supported, skipping
Oct 16, 2024 8:27:50 AM hudson.remoting.Launcher$CuiListener status
INFO: Trying protocol: JNLP4-connect
Oct 16, 2024 8:27:50 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Oct 16, 2024 8:27:51 AM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: 9c:9a:a1:9b:d0:4e:10:44:f7:e7:98:8c:b7:28:40:8a
Oct 16, 2024 8:27:51 AM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

Step 4: Set Up a Distributed Pipeline

1. Go back to the Jenkins dashboard.
2. Select your Maven project or create a new one by clicking on **New Item**.
3. Configure the project as a **Pipeline** job.
   - **Pipeline Script**: Define a pipeline script with stages for `Build` and `Test`.
   - Assign stages to different nodes using `node('node_label')` for each slave node. Example:

```
groovy
Copy code
pipeline {
    agent none
    stages {
        stage('Build and Test') {
            agent { label 'JAY DALSANIYA' }
            steps {
                sh 'mvn clean install'
            }
        }
        stage('Windows Agent') {
            agent { label 'windows-agent1-pipeline' }
            steps {
                sh 'mvn test'
            }
```

```
                }
                stage('Another Windows Agent') {
                    agent { label 'windows-agent2-pipeline' }
                    steps {
                        sh 'mvn test'
                    }
                }
            }
        }
```

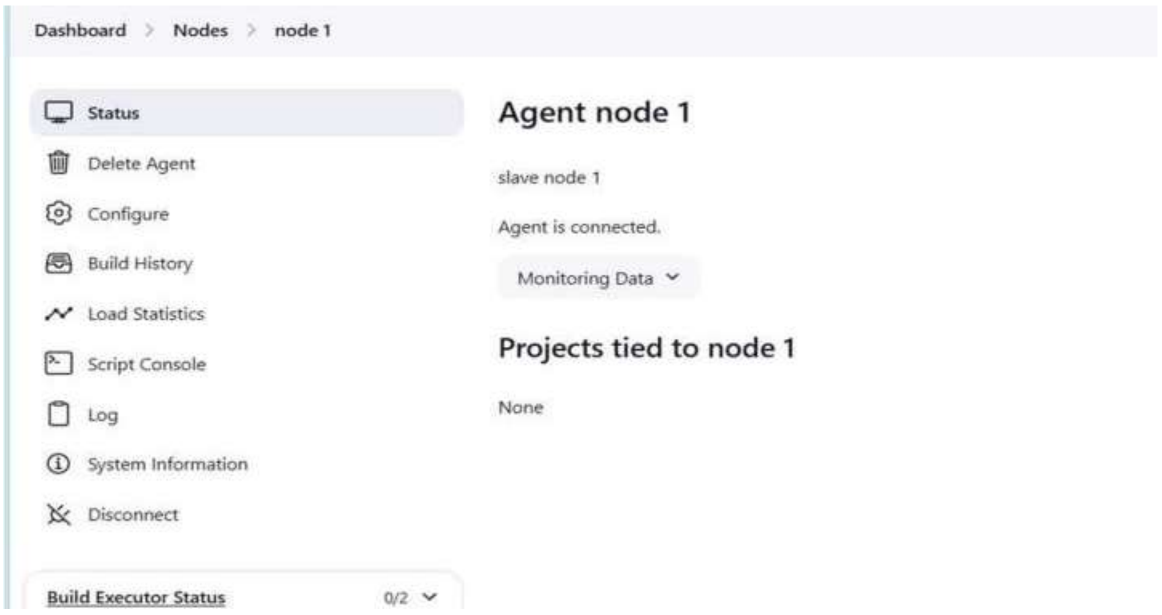
4. Click **Save** to apply your changes.

Step 5: Run and Monitor the Pipeline

1. Run the pipeline by clicking **Build Now** on the project page.
2. Go to the **Status** page to monitor the pipeline progress.
3. The **Stage View** will display each stage's execution on different nodes.

Observing Results:

- The **Stage View** shows each stage's build and test time, indicating successful distribution across slave nodes.
- Each stage executed on its designated node, ensuring the Maven project was built and tested as intended.



Dashboard > Nodes > node 1

Status
Delete Agent
Configure
Build History
Load Statistics
Script Console
Log
System Information
Disconnect

Build Executor Status 0/2

Agent node 1

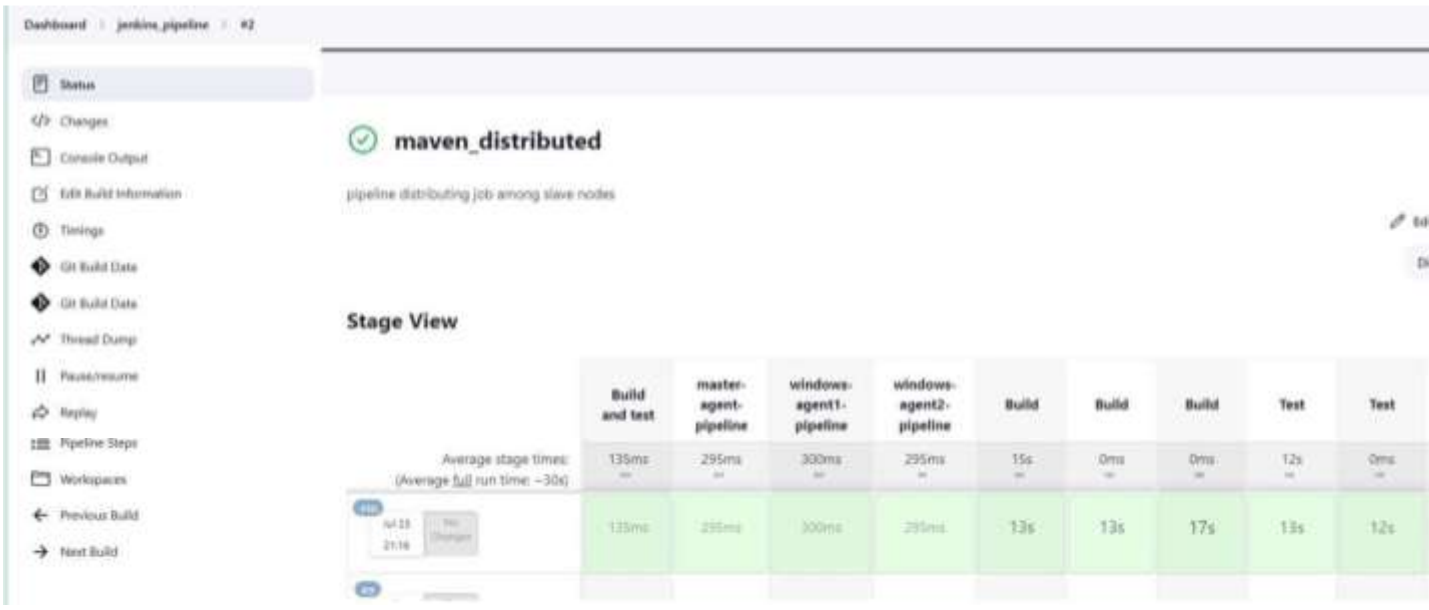
slave node 1

Agent is connected.

Monitoring Data

Projects tied to node 1

None



Dashboard > jenkins_pipeline > #2

- Status
- Changes
- Console Output
- Edit Build Information
- Timings
- Git Build Data
- Git Build Data
- Thread Dump
- Pause/resume
- Replay
- Pipeline Steps
- Workspaces
- Previous Build
- Next Build

maven_distributed

pipeline distributing job among slave nodes

Stage View

	Build and test	master-agent-pipeline	windows-agent1-pipeline	windows-agent2-pipeline	Build	Build	Build	Test	Test
Average stage times: (Average full run time: ~30s)	135ms	295ms	300ms	295ms	15s	0ms	0ms	12s	0ms
Jul 23 21:16 No Changes	135ms	295ms	300ms	295ms	13s	13s	17s	13s	12s