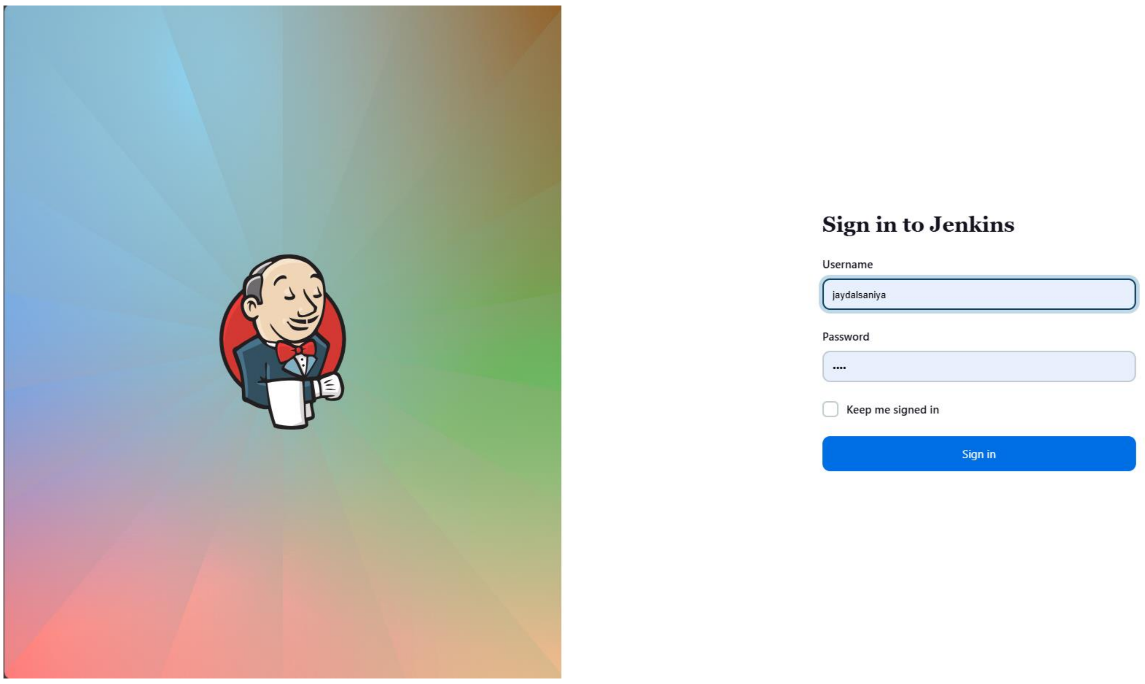
4. Building a CI/CD Pipeline with Jenkins.

Step 1: Login to Jenkins

1. **Open Jenkins**: Open your browser and go to your Jenkins dashboard by typing:

   http://localhost:8000/

2. **Login**: Enter your Jenkins username and password.



Sign in to Jenkins

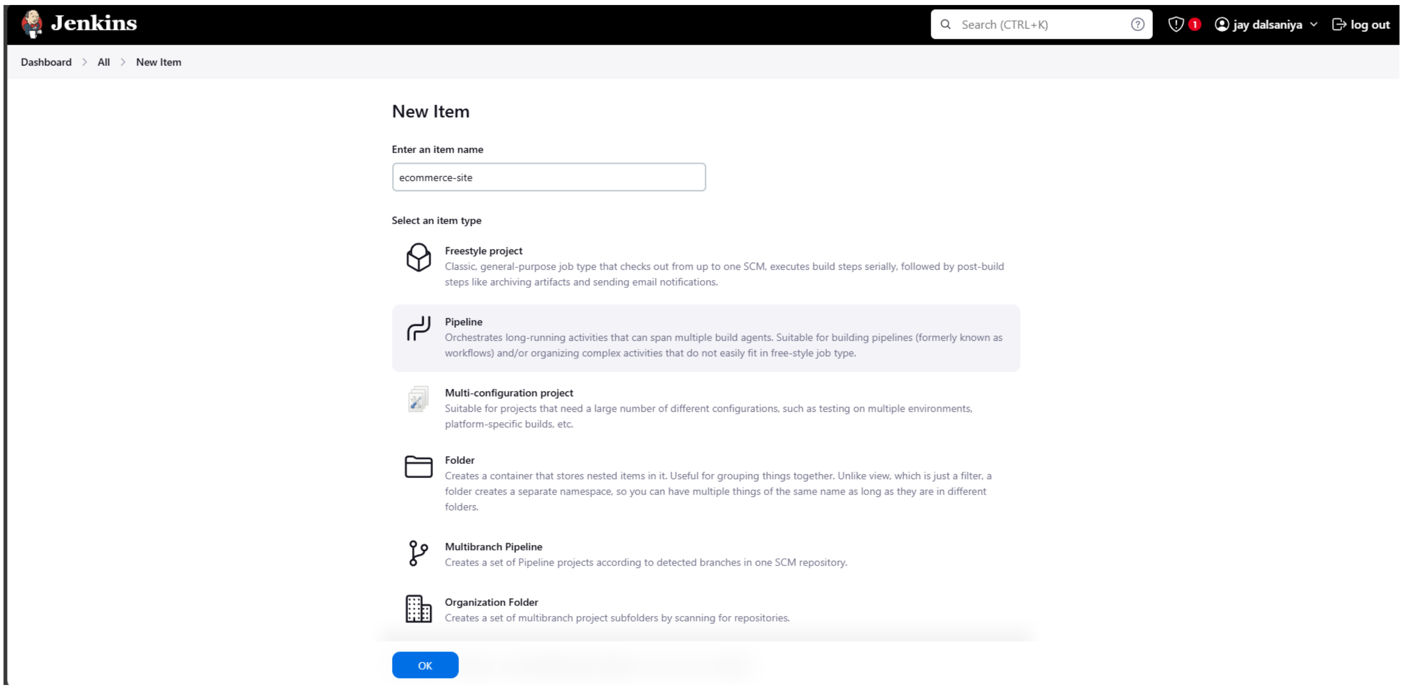
Username: jaydalsaniya

Password: ••••

Keep me signed in

Step 2: Create a New Job

1. **Create a New Job**:
   - From the Jenkins dashboard, click on **"New Item"**.
   - Enter a job name, e.g., ecommerce-site.
   - Select **"Pipeline"** as the job type and click **OK**.



Jenkins
Dashboard > All > New Item

New Item

Enter an item name: ecommerce-site

Select an item type

Freestyle project — Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

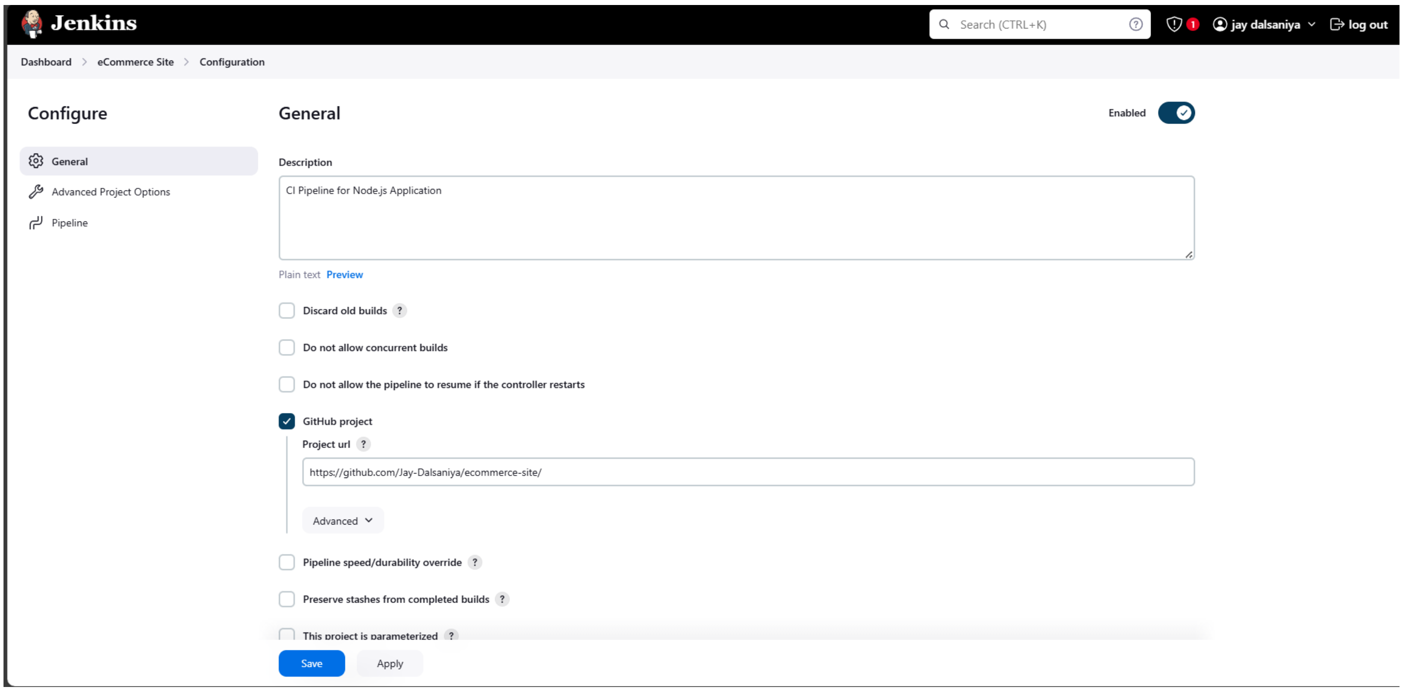
Folder — Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline — Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder — Creates a set of multibranch project subfolders by scanning for repositories.

Step 3: Configure Your Job

1. **Job Description**:
   - Scroll down to the **"General"** section.
   - Provide a description of the job (e.g., "CI Pipeline for Node.js Application").



Jenkins
Dashboard > eCommerce Site > Configuration

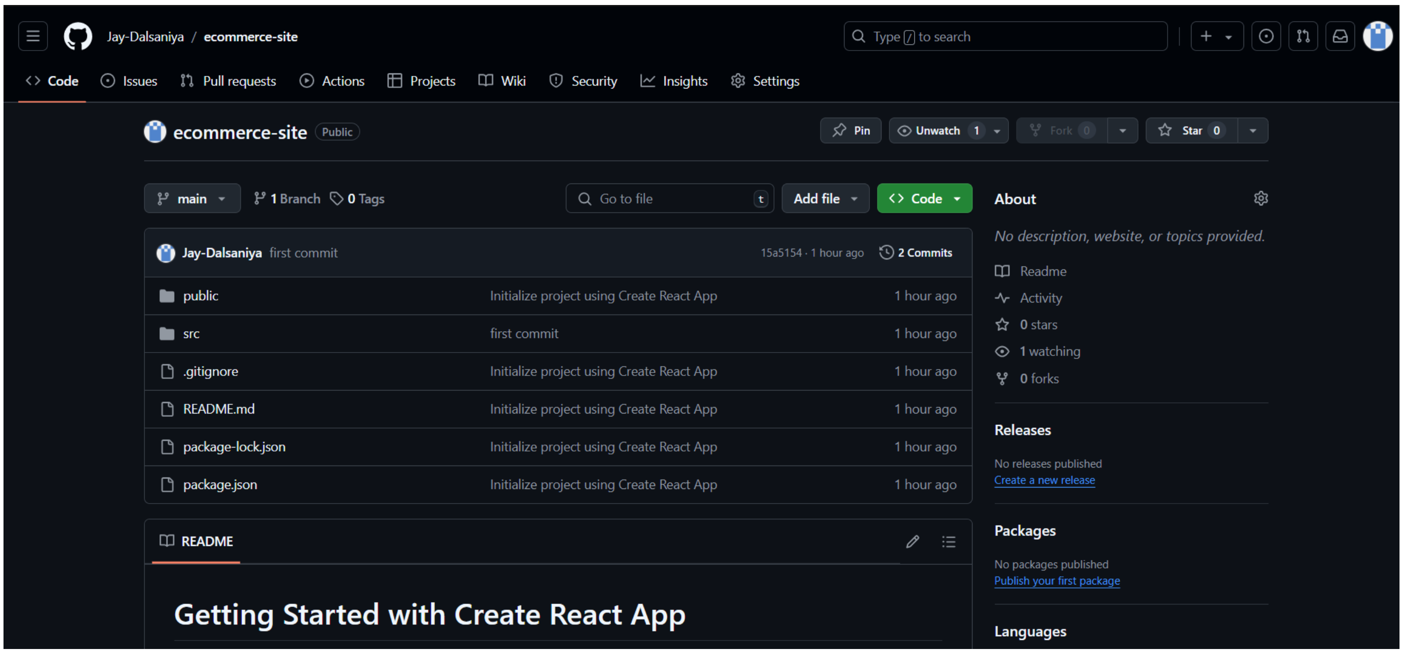
Configure — General (Enabled)

Description: CI Pipeline for Node.js Application

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project
  - Project url: https://github.com/Jay-Dalsaniya/ecommerce-site/
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterized

Step 4: Provide GitHub Repo Link and Description

1. **GitHub Repository**:
   - In the **"Pipeline"** section, under **"Definition"**, choose **Pipeline script**.
   - In the pipeline script area (or definition if using SCM), you can reference your GitHub repo link like so:
   - Use this link: git remote add origin `https://github.com/Jay-Dalsaniya/ecommerce-site.git`
   - This step pulls the source code from the GitHub repository.

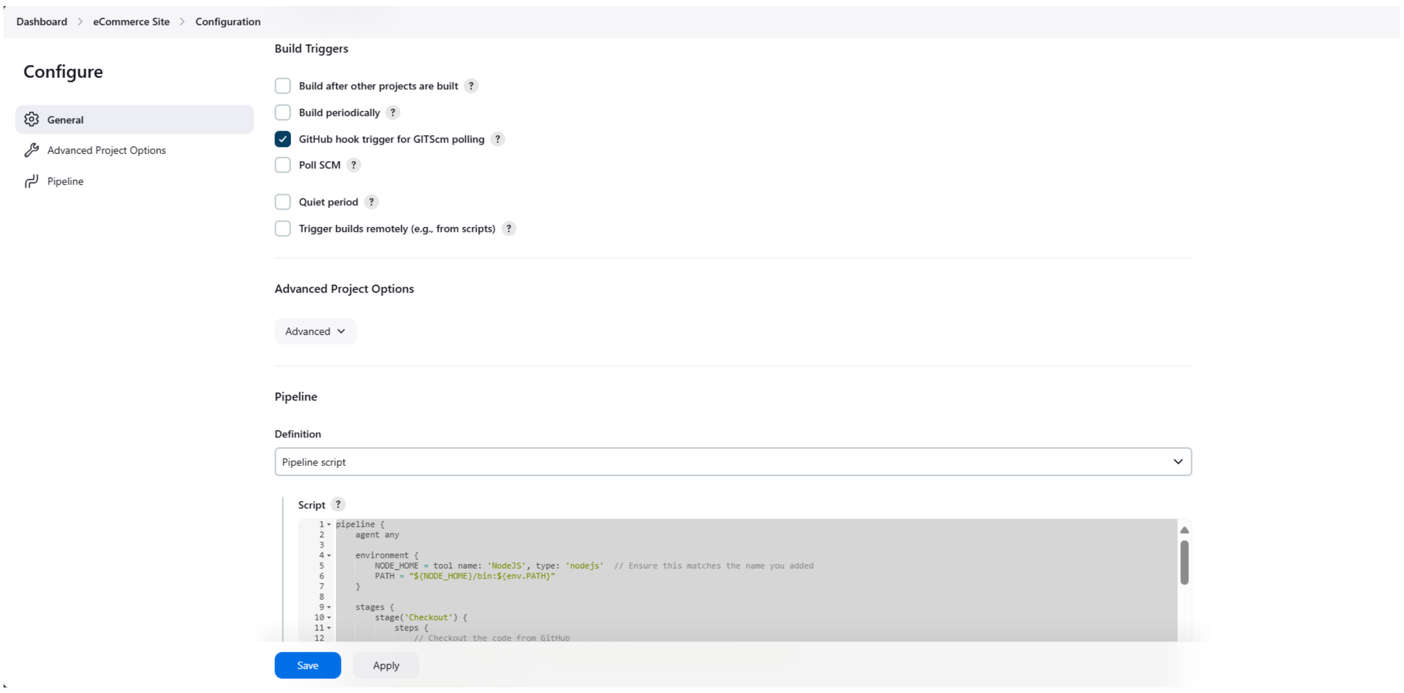


Jay-Dalsaniya first commit — 15a5154 · 1 hour ago — 2 Commits

public	Initialize project using Create React App	1 hour ago
src	first commit	1 hour ago
.gitignore	Initialize project using Create React App	1 hour ago
README.md	Initialize project using Create React App	1 hour ago
package-lock.json	Initialize project using Create React App	1 hour ago
package.json	Initialize project using Create React App	1 hour ago

Step 5: Check the GitHub Hook Trigger

1. **Trigger Build with GitHub Hook**:
   - Scroll down to **Build Triggers**.
   - Check **"GitHub hook trigger for GITScm polling"**.


Dashboard > eCommerce Site > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Build Triggers

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

Advanced Project Options

Advanced

Pipeline

Definition

Pipeline script

Script

```
pipeline {
    agent any

    environment {
        NODE_HOME = tool name: 'NodeJS', type: 'nodejs'  // Ensure this matches the name you added
        PATH = "${NODE_HOME}/bin:${env.PATH}"
    }

    stages {
        stage('Checkout') {
            steps {
                // Checkout the code from GitHub
```

Save Apply

Step 6: Write the Pipeline Script

1. **Write the Pipeline Script**:
    - Scroll down to the **Pipeline** section and enter the following script:

```
pipeline {
    agent any

    environment {
        NODE_HOME = tool name: 'NodeJS', type: 'nodejs'   // Ensure
this matches the name you added
        PATH = "${NODE_HOME}/bin:${env.PATH}"
    }

    stages {
        stage('Checkout') {
            steps {
                // Checkout the code from GitHub
                git branch: 'main', url: 'https://github.com/Jay-
Dalsaniya/ecommerce-site.git'
            }
        }

        stage('Check Node Version') {
            steps {
                script {
                    if (isUnix()) {
                        sh 'node -v'   // For Linux/Unix
                        sh 'npm -v'
                    } else {
                        bat 'node -v'   // For Windows
                        bat 'npm -v'
                    }
                }
            }
        }

        stage('Install Dependencies') {
            steps {
                script {
                    if (isUnix()) {
                        sh 'npm install'
                    } else {
                        bat 'npm install'
                    }
                }
            }
        }

        stage('Run nodejs app') {
            steps {
                script {
                    if (isUnix()) {
                        sh 'npm start'
                    } else {
                        bat 'npm start'
                    }
                }
            }
        }
    }
}
```

Note:

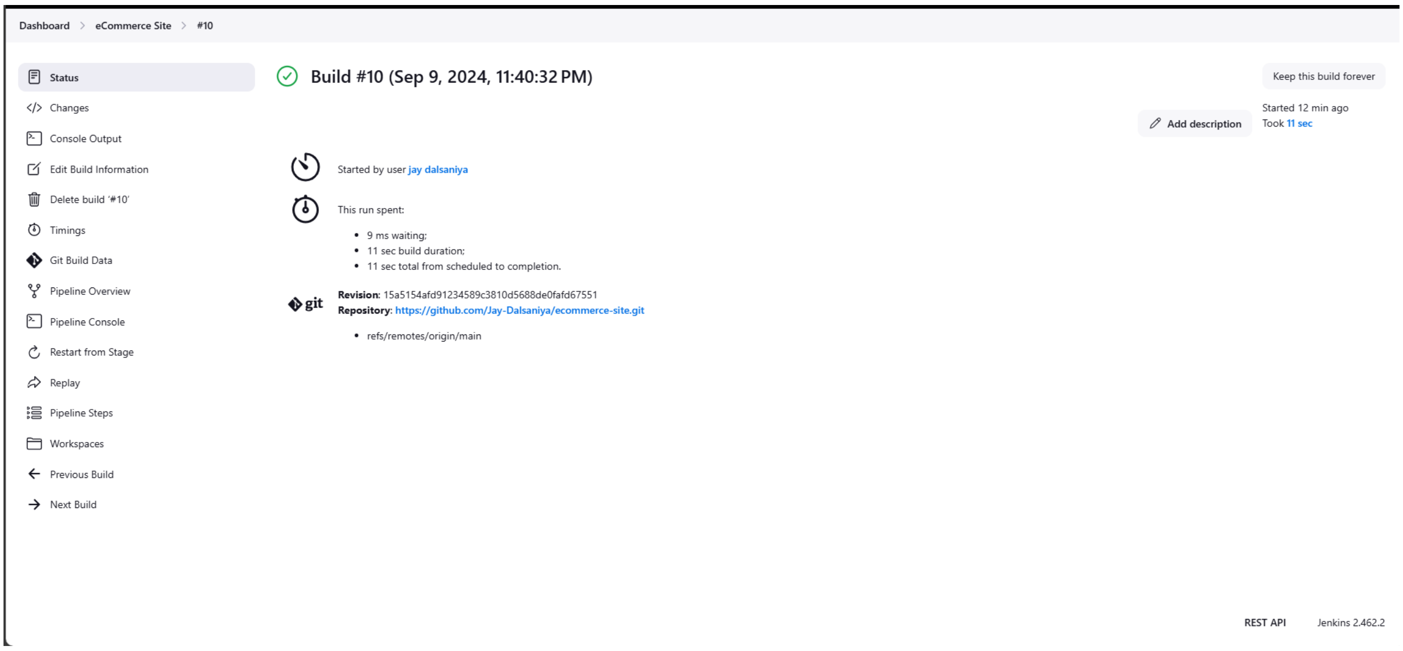
- If you're running Jenkins on **Linux/Ubuntu**, replace bat with sh:
  - sh 'node -v'
  - sh 'npm install'
  - sh 'npm start'

Explanation of the Pipeline Script:

- **agent any**: Jenkins will run this pipeline on any available agent.
- **environment**: Sets up the Node.js environment for the build.
- **Stages**:
  - **Checkout**: Fetches the code from the specified GitHub repository.
  - **Check Node Version**: Prints Node.js and npm versions to verify the environment setup.
  - **Install Dependencies**: Runs npm install to install all project dependencies.
  - **Run nodejs app**: Starts the Node.js application.

Step 7: Build the Job

1. **Save the Configuration**: After you write the pipeline script, click on **Save**.
2. **Build the Job**:
   - On the Jenkins dashboard, click on **Build Now** to run the pipeline.



Build #10 (Sep 9, 2024, 11:40:32 PM)

Started 12 min ago
Took 11 sec

Started by user jay dalsaniya

This run spent:

- 9 ms waiting;
- 11 sec build duration;
- 11 sec total from scheduled to completion.

Revision: 15a5154afd91234589c3810d5688de0fafd67551
Repository: https://github.com/Jay-Dalsaniya/ecommerce-site.git

- refs/remotes/origin/main

Step 8: Check Console Output

1. **View Build Logs**:
   - Once the build starts, click on the running build under **Build History**.
   - Click on **Console Output** to view the logs.
   - You should see the pipeline executing the steps: cloning the repository, checking Node.js and npm versions, installing dependencies, and starting the application.

Step 9: Verify the App Running

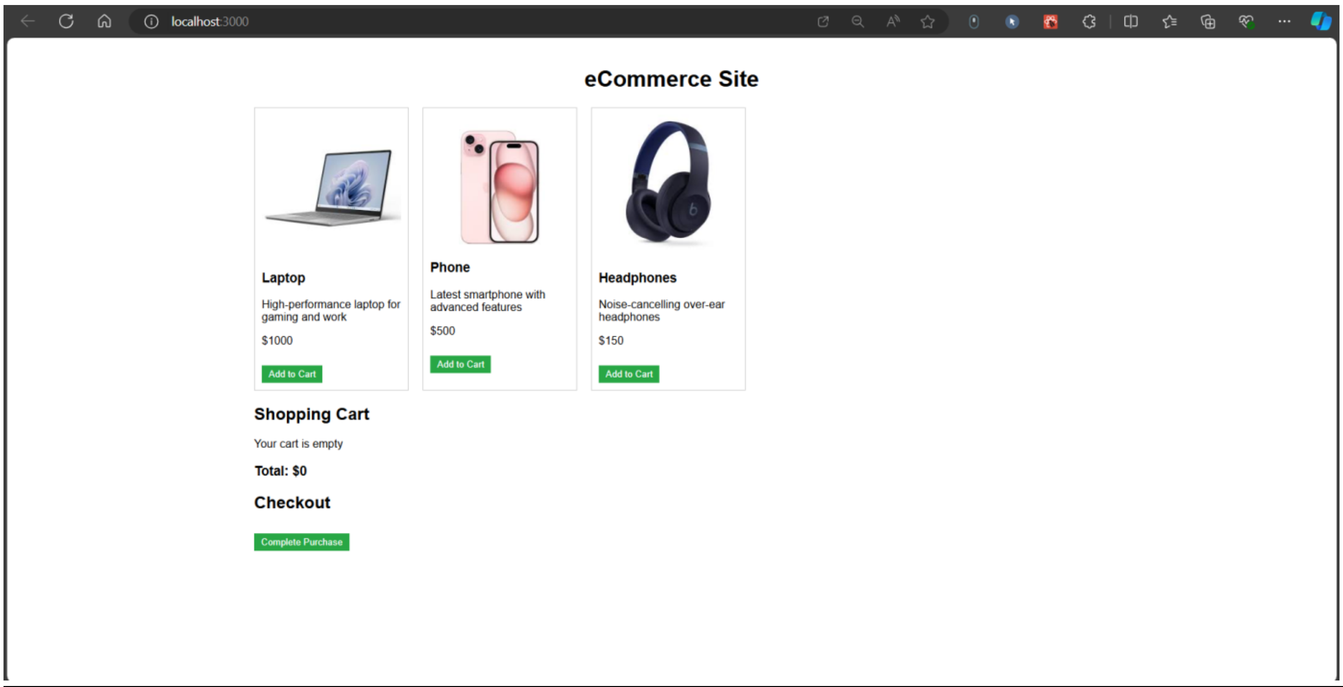
1. **Verify Success**:
   - If the pipeline runs successfully, the logs should show that the application has started without any issues.
   - The npm start command should output logs indicating that your Node.js app is running.

```
[Pipeline] { (Run nodejs app)
[Pipeline] script
[Pipeline] {
[Pipeline] isUnix
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\eCommerce Site>npm start

> ecommerce-site@0.1.0 start
> react-scripts start

(node:13284) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:13284) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

⊠[0;33mOne of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it ⊠[1mmay break at any time⊠[0;33m.

babel-preset-react-app is part of the create-react-app project, ⊠[1mwhich
is not maintianed anymore⊠[0;33m. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.⊠[0m

Compiled successfully!

You can now view ecommerce-site in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.186.36:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled ⊠[1m⊠[32msuccessfully⊠[39m⊠[22m
```

Step 10: Confirm the App is Running

If the application is running, you should be able to confirm it by:

- Visiting the localhost or domain where the app is deployed (if the app has a front-end).
- Observing successful start logs in the Jenkins **Console Output**.



eCommerce Site

Laptop
High-performance laptop for gaming and work
$1000
Add to Cart

Phone
Latest smartphone with advanced features
$500
Add to Cart

Headphones
Noise-cancelling over-ear headphones
$150
Add to Cart

Shopping Cart
Your cart is empty
Total: $0
Checkout
Complete Purchase