**✅ Project Structure (Assumed)**

plaintext

CopyEdit

your-repo/

├── Jenkinsfile

├── backend/

│ └── pom.xml / build.gradle

├── frontend/

│ └── package.json

**✅ Sample Jenkinsfile (Declarative Pipeline)**

groovy

CopyEdit

pipeline {

agent any

environment {

NODE\_ENV = 'production'

}

tools {

nodejs 'NodeJS 16' // Set this in Jenkins Global Tools

maven 'Maven 3.8' // Set this in Jenkins Global Tools

}

stages {

stage('Checkout') {

steps {

checkout scm

}

}

stage('Install Frontend Dependencies') {

dir('frontend') {

steps {

sh 'npm install'

}

}

}

stage('Build Frontend') {

dir('frontend') {

steps {

sh 'npm run build'

}

}

}

stage('Unit Tests') {

parallel {

stage('React Tests') {

dir('frontend') {

steps {

sh 'npm test -- --watchAll=false'

}

}

}

stage('Spring Boot Tests') {

dir('backend') {

steps {

sh './mvnw test || mvn test' // Fallback if wrapper not present

}

}

}

}

}

stage('Build Backend') {

dir('backend') {

steps {

sh './mvnw clean package -DskipTests || mvn clean package -DskipTests'

}

}

}

stage('Archive Artifacts') {

steps {

archiveArtifacts artifacts: 'backend/target/\*.jar', fingerprint: true

archiveArtifacts artifacts: 'frontend/build/\*\*', fingerprint: true

}

}

// Optional Deployment Stage

// stage('Deploy') {

// steps {

// // your deploy steps here (e.g., SCP, Docker, K8s, etc.)

// }

// }

}

post {

success {

echo '🎉 Build completed successfully!'

}

failure {

echo '❌ Build failed.'

}

}

}

**🧠 Explanation of Each Section**

**✅ agent any**

Tells Jenkins to run the pipeline on any available agent (node/slave).

**✅ tools**

Defines versions of Node.js and Maven that Jenkins should use. You must install/configure these in **Jenkins > Global Tool Configuration**.

**✅ environment**

Used to define environment variables, e.g., NODE\_ENV.

**✅ stages**

Each stage represents a CI step:

* **Checkout:** Clones your Git repo.
* **Frontend Install/Build:** Runs npm install and npm run build.
* **Tests:** Runs React and Spring Boot tests **in parallel**.
* **Backend Build:** Compiles and packages your Spring Boot app (.jar).
* **Archive Artifacts:** Stores build results inside Jenkins.

**🛠 Requirements on Jenkins Machine**

Make sure Jenkins has:

* **Java 17** (or whatever your Spring Boot app needs)
* **Maven** or Gradle
* **Node.js + npm**
* Optional: **Git**, Docker, SSH tools for deployment

**📦 Optional: Serve Frontend With Backend**

If your Spring Boot app serves static files:

1. Add this to your pom.xml/build.gradle:
   * Copy frontend build/ to backend/src/main/resources/static/
2. Modify Jenkinsfile:

groovy

CopyEdit

stage('Copy Frontend to Backend') {

steps {

sh 'cp -r frontend/build/\* backend/src/main/resources/static/'

}

}

**🔒 Bonus: Credentials Handling (for deployments)**

Use the **Jenkins Credentials Plugin** to inject API tokens, SSH keys, passwords, etc., securely.

groovy

CopyEdit

withCredentials([usernamePassword(credentialsId: 'my-creds', usernameVariable: 'USER', passwordVariable: 'PASS')]) {

sh 'echo "Deploying with $USER"'

}

Perfect! If you want to do **full stack CI/CD (React + Spring Boot)** in Jenkins **without using a scripted pipeline (Jenkinsfile)**, you can use the **Freestyle Project UI** instead.

Here’s a full step-by-step guide using Jenkins Freestyle job — **no pipeline scripts required**.

**✅ Goal**

Use Jenkins **Freestyle Job** to:

1. Checkout code (React + Spring Boot)
2. Build frontend (React)
3. Copy frontend into Spring Boot static folder
4. Build backend (Spring Boot)
5. Deploy (Tomcat / Maven / Copy)

**✅ Prerequisites**

| **Requirement** | **Description** |
| --- | --- |
| Jenkins installed | Locally or on server |
| Node.js plugin | Installed via Jenkins Plugin Manager |
| Maven installed | Configured under **Global Tool Configuration** |
| Git installed | Jenkins agent must have Git installed |
| Tomcat/Maven Server | For deployment (optional depending on your target) |

**🛠️ Step-by-Step: Freestyle CI/CD Job**

**1️⃣ Create a Freestyle Job**

* Go to **Jenkins Dashboard → New Item**
* Enter a name like: fullstack-ci-cd
* Select **Freestyle project**
* Click **OK**

**2️⃣ Configure Source Code**

* Go to the **"Source Code Management"** section
* Select **Git**
* Enter your repo URL (e.g., GitHub)
* Add credentials if needed

**3️⃣ Configure Build Environment**

In **"Build Environment"**:

* ✅ Check **"Provide Node & npm bin/ folder to PATH"**
* Select your Node.js version (must be configured in Jenkins → Global Tools)

**4️⃣ Add Build Steps**

**➤ Build Step 1: Install and Build React Frontend**

* Click **"Add build step" → Execute shell**

bash

CopyEdit

cd frontend

npm install

npm run build

**➤ Build Step 2: Copy React Build to Spring Boot**

* Click **"Add build step" → Execute shell**

bash

CopyEdit

rm -rf backend/src/main/resources/static/\*

cp -r frontend/build/\* backend/src/main/resources/static/

**➤ Build Step 3: Build Spring Boot App**

* Click **"Add build step" → Invoke top-level Maven targets**
  + **Goals:** clean package -DskipTests
  + **POM:** backend/pom.xml

**(Optional) ➤ Build Step 4: Deploy**

Choose one:

**Option A: WAR Deploy to Tomcat (using curl)**

bash

CopyEdit

curl -u admin:admin \

-T backend/target/yourapp.war \

"http://localhost:8080/manager/text/deploy?path=/yourapp&update=true"

**Option B: Deploy to Maven Repo**

* Add another **Maven step**
  + **Goals:** deploy -DskipTests
  + **POM:** backend/pom.xml
  + Ensure settings.xml has credentials

**5️⃣ Post-build Actions (Optional)**

You can add:

* **Email Notification**
* **Archive Artifacts** → backend/target/\*.jar
* **Trigger other jobs**
* **Deploy to a remote server (via SSH plugin)**

**🧾 Summary**

| **Stage** | **Tool / Build Step** |
| --- | --- |
| Frontend Build | Execute shell → npm install && npm run build |
| Copy to Backend | Execute shell |
| Backend Build | Invoke top-level Maven targets |
| Deploy (Optional) | Execute shell with curl, or Maven deploy |

**✅ Final Notes**

* This is **100% GUI-based CI/CD**, no Jenkinsfile needed.
* Ensure proper folder structure in your Git repo:

bash

CopyEdit

/frontend/package.json

/backend/pom.xml

* Ensure Jenkins has:
  + **Node.js plugin**
  + **Maven tool configured**
  + **Permissions for Git, curl, etc.**