# Integrate Jenkins with GitHub for build and test for a node.js application

To integrate Jenkins with GitHub to build a Node.js application using Maven and test it with Selenium, you can follow these steps. This example assumes that your Node.js application is structured to support Maven (possibly using the frontend-maven-plugin for managing Node.js dependencies) and that you have Selenium tests configured.

**Prerequisites**

1. **Jenkins Installed**: Ensure Jenkins is running.
2. **GitHub Repository**: Have your Node.js application hosted on GitHub.
3. **Required Jenkins Plugins**: Install the following plugins:
   * GitHub Integration Plugin
   * Pipeline Plugin
   * NodeJS Plugin (for Node.js management)

# PHASE I

Creating a Node.js application and uploading it to a GitHub repository involves several steps. Here’s a detailed guide to help you through the process.

### Step 1: Set Up Your Node.js Application

1. **Install Node.js**: Ensure you have Node.js installed on your machine. You can download it from [nodejs.org](https://nodejs.org/).
2. **Create a Project Directory**: Open your terminal and create a new directory for your Node.js app:

bash

Copy code

mkdir my-node-app

cd my-node-app

1. **Initialize the Project**: Run the following command to create a package.json file:

bash

Copy code

npm init -y

This will create a default package.json file with basic configuration.

1. **Install Express**: For this example, we'll use Express.js as the web framework:

bash

Copy code

npm install express

1. **Create the Application**: Create a file named app.js:

bash

Copy code

touch app.js

Open app.js and add the following code:

javascript

Copy code

const express = require('express');

const app = express();

const PORT = 3000;

app.get('/', (req, res) => {

res.send('Welcome to My Node.js App!');

});

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

### Step 2: Test Your Application

1. **Run Your Application**: Start your application by running:

bash

Copy code

node app.js

1. **Access the Application**: Open your browser and go to http://localhost:3000. You should see the message "Welcome to My Node.js App!".

### Step 3: Create a GitHub Repository

1. **Log in to GitHub**: Go to [github.com](https://github.com) and log in or create a new account.
2. **Create a New Repository**:
   * Click on the "+" icon in the top right corner and select "New repository".
   * Name your repository (e.g., my-node-app).
   * Optionally add a description, choose whether to make it public or private, and click "Create repository".

### Step 4: Upload Your Application to GitHub

1. **Initialize a Git Repository**: In your project directory, initialize a git repository:

bash

Copy code

git init

1. **Add Files to the Repository**: Add all files to the repository:

bash

Copy code

git add .

1. **Commit Your Changes**: Commit the changes with a message:

bash

Copy code

git commit -m "Initial commit"

1. **Link to Your GitHub Repository**: Copy the repository URL from GitHub and link it to your local repository. Replace <your-github-username> with your actual GitHub username:

bash

Copy code

git remote add origin https://github.com/<your-github-username>/my-node-app.git

1. **Push Your Changes to GitHub**: Push your local repository to GitHub:

bash

Copy code

git push -u origin master

### Step 5: Verify Your Repository

1. **Check Your GitHub Repository**: Go back to your GitHub repository in your browser. You should see all your Node.js application files uploaded.

# PHASE II

**Steps to Integrate Jenkins with GitHub**

**1. Create a Jenkins Job**

* Go to your Jenkins dashboard.
* Click on "New Item".
* Select "Pipeline" and name your job.

**2. Configure GitHub Webhook**

* Go to your GitHub repository.
* Navigate to "Settings" > "Webhooks".
* Click on "Add webhook".
* Enter your Jenkins URL followed by /github-webhook/ (e.g., http://your-jenkins-url/github-webhook/).
* Set the content type to application/json.
* Choose the events to trigger the webhook (typically "Just the push event").
* Click "Add webhook".

**3. Set Up Jenkins Pipeline**

* In your Jenkins job configuration, scroll to the "Pipeline" section.
* Choose "Pipeline script from SCM".
* Select "Git" as the SCM.
* Enter your GitHub repository URL.
* Provide the credentials if required.
* Specify the branch to build (e.g., \*/main).

**4. Create a Jenkinsfile**

In your repository, create a Jenkinsfile at the root level. Below is an example of a Jenkinsfile that builds a Node.js application using Maven and runs Selenium tests:

groovy

Copy code

pipeline {

agent any

tools {

nodejs 'NodeJS' // Name of your NodeJS installation in Jenkins

}

stages {

stage('Checkout') {

steps {

// Checkout the code from GitHub

git url: 'https://github.com/your-repo/your-project.git', branch: 'main'

}

}

stage('Install Dependencies') {

steps {

// Install Node.js dependencies

sh 'npm install'

}

}

stage('Build with Maven') {

steps {

// Build the application using Maven

sh 'mvn clean package'

}

}

stage('Run Selenium Tests') {

steps {

// Run Selenium tests (this assumes you have a script for running your tests)

// You may need to adjust this command based on your test setup

sh 'mvn test -Dtest=YourSeleniumTestClass'

}

}

}

post {

success {

echo 'Build and tests were successful!'

}

failure {

echo 'Build or tests failed!'

}

always {

// Clean up or send notifications

echo 'Cleaning up...'

}

}

}

**Explanation of the Jenkinsfile**

1. **Checkout**: Checks out the code from your GitHub repository.
2. **Install Dependencies**: Installs Node.js dependencies using npm install.
3. **Build with Maven**: Builds the application using Maven.
4. **Run Selenium Tests**: Executes the Selenium tests defined in your project. Adjust the command as necessary based on your testing setup.

**Additional Configuration**

* **Node.js Configuration**: Ensure that you have Node.js configured in Jenkins (Manage Jenkins > Global Tool Configuration).
* **Selenium Setup**: Make sure your Selenium tests are properly configured. This may involve ensuring that the Selenium server or browser drivers are set up correctly.
* **Error Handling and Notifications**: You can enhance the post section to add email or Slack notifications based on your team's preferences.