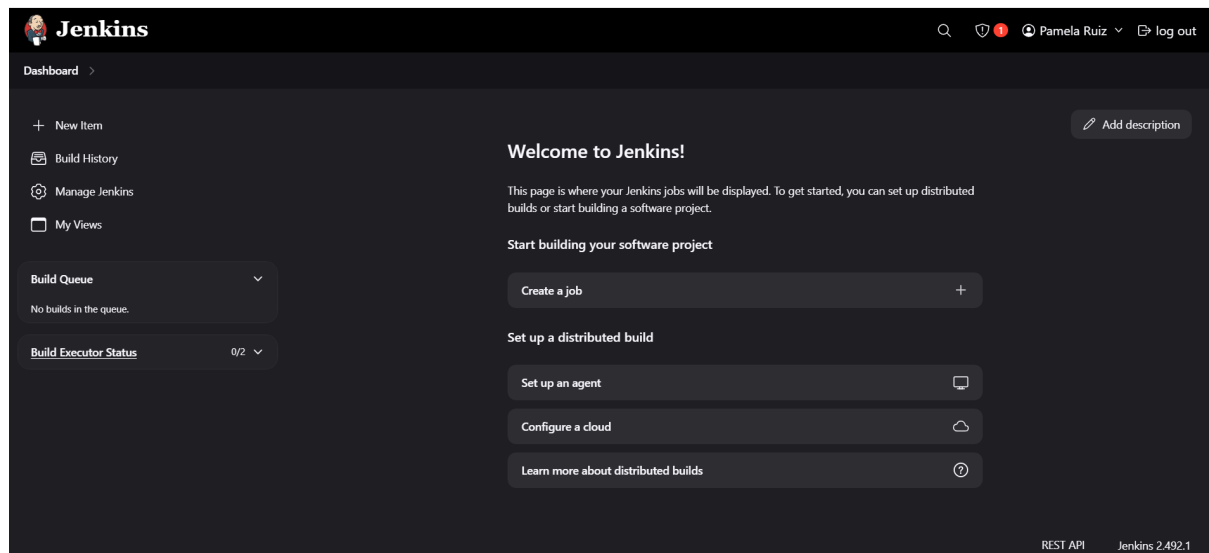


## Jenkins installation and Node.js project setup

- 1- I've downloaded the installer for windows (.msi)
- 2- I've executed the installer and follow the instructions
- 3- Jenkins is running in localhost, port 8080



4- Once Jenkins is installed, I need to set up my Node.js project. I chose to download the test project from the link provided in the description of the task.

5- The project is saved in the following link

<https://github.com/PameRuiz/DevOps/tree/main/Jenkins-CI>. Jenkinsfile has been added here.

The content of Jenkinsfile is:

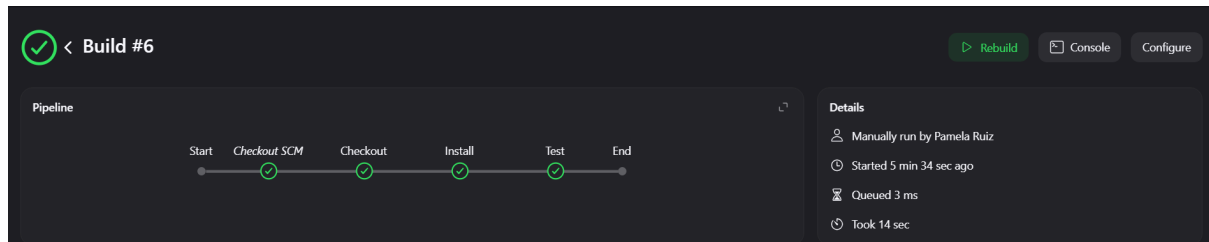
```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Install') {
      steps {
        bat 'cd Jenkins-CI & npm install'
      }
    }
    stage('Test') {
      steps {
        bat 'cd Jenkins-CI & npm test'
      }
    }
  }
}
```

This file is configured to run in any agent available. It is running on my local machine so there is only one agent for the moment.

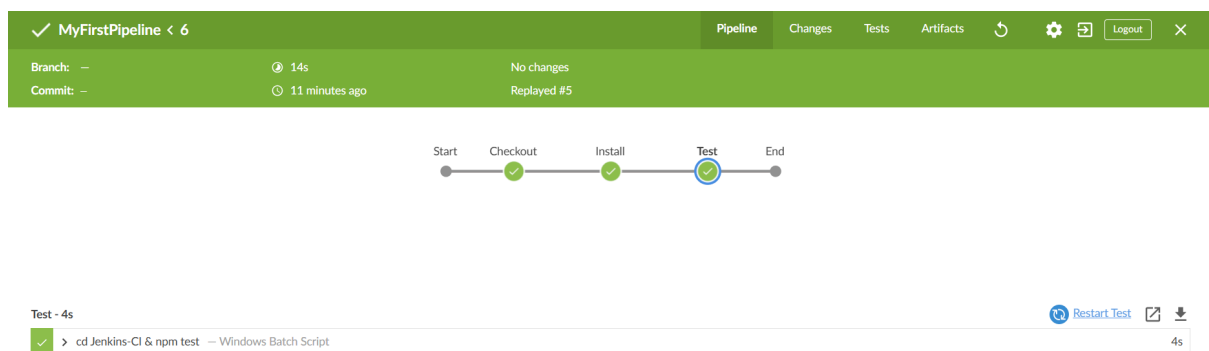
It contains 3 stages.

1. Checkout: In this stage, Jenkins makes the checkout of the project for having all the files available.
2. Install: In this stage, Jenkins installs all the node dependencies for the project
3. Test: In this stage, Jenkins runs the tests written in the project.

The result of a successful run looks like this



Once the job is running, I installed the Blue Ocean plugin so we can see the job in a better way



As shown in the image, we can observe the different stages that run during the job.

After we have implemented the pipeline, I started to add improvements

1. **Slack notification:** I needed to add a post build step that always sends a notification to Slack.
  - a. I created a new app in Slack
  - b. Copied the token for the bot
  - c. Added the @Jenkins bot to my channel called Jenkins
  - d. Added the step in the Jenkinsfile

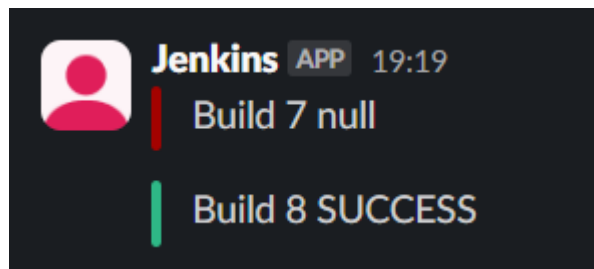
```
pipeline {
  agent any
  stages {
    stage('Checkout') {
```

```

    steps {
        checkout scm
    }
}
stage('Install') {
    steps {
        bat 'cd Jenkins-CI & npm install'
    }
}
stage('Test') {
    steps {
        bat 'cd Jenkins-CI & npm test'
    }
}
}
post {
    always {
        slackSend(
            message: "Build ${env.BUILD_NUMBER} ${currentBuild.result}",
            color: currentBuild.result == 'SUCCESS' ? 'good' : 'danger'
        )
    }
}
}
}

```

The notification in Slack looks like this



2. The second improvement was running in parallel two stages

I've created a new step for running lint.

the code in the Jenkinsfile is the following:

```

stage('Checks') {
    steps {
        parallel(
            test: {bat 'cd Jenkins-CI & npm test'},
            lint: {bat 'cd Jenkins-CI & npm run lint'}
        )
    }
}
}

```

I am running tests and lint in parallel to save time in the execution of the pipeline.

3. The last improvement was adding the retry option. I've chosen the test stage for adding the retry.

For setting this configuration we just need to add a `retry(N)` under the stage, where N is the maximum number of tries.

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Install') {
      steps {
        bat 'cd Jenkins-CI & npm install'
      }
    }
    stage('Checks') {
      steps {
        parallel(
          test: {
            retry(2) {
              bat 'cd Jenkins-CI & npm test'
            }
          },
          lint: {
            bat 'cd Jenkins-CI & npm run lint'
          }
        )
      }
    }
  }
  post {
    always {
      slackSend(
        message: "Build ${env.BUILD_NUMBER} ${currentBuild.result}",
        color: currentBuild.result == 'SUCCESS' ? 'good' : 'danger'
      )
    }
  }
}
```

4. For configuring the webhook I needed to make some changes.
  - a. Since I had my Jenkins server running locally. I had to use ngrok for making it accessible from Internet
  - b. In github I created a new webhook with the following data

**Payload URL \***

**Content type \***


- c. In Jenkins I set the configuration in the job for being triggered by changes in github


**Triggers**


Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?
- ☐ Trigger builds remotely (e.g., from scripts) ?


- d. Once everything was configured I was able to test the trigger by sending changes to the project

 **#13 (Feb 20, 2025, 7:18:18 PM)**


 Started by GitHub push by PameRuiz

 This run spent:

- 5.4 sec waiting;
- 12 sec build duration;
- 18 sec total from scheduled to completion.

 **Revision:** 66b7a05669c2ba19b7777db6baad5c874d6a8d4c  
**Repository:** <https://github.com/PameRuiz/DevOps.git>

- refs/remotes/origin/main

 **Changes**

1. fix: testing webhook ([commit: 66b7a05](#)) ([details](#) / [githubweb](#))