# CI/CD Pipeline.

*Server 1: - Jenkins (Make sure to install Jenkins, Git, Maven, Docker container)*

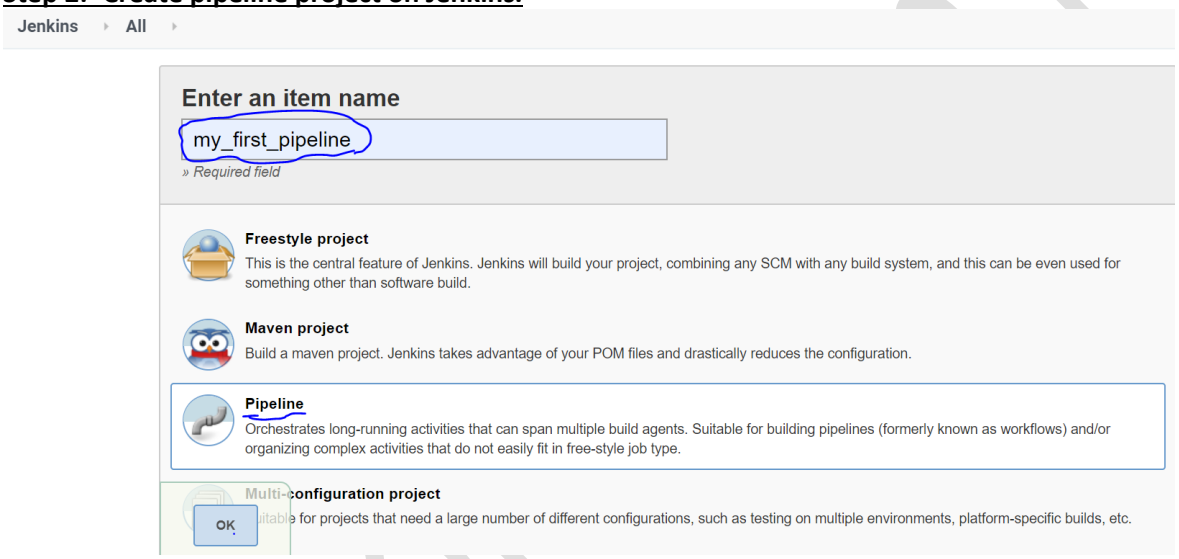
**Step1:- Fork and clone the sample repository on your GitHub.**

@Fork from this link https://github.com/jenkins-docs/simple-java-maven-app

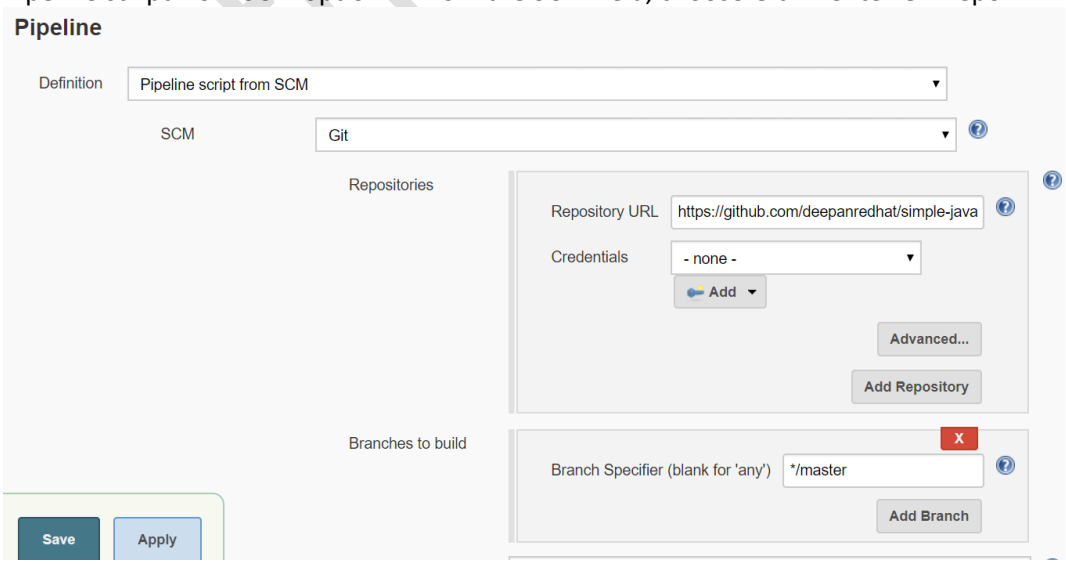
#git clone https://github.com/deepanredhat/simple-java-maven-app

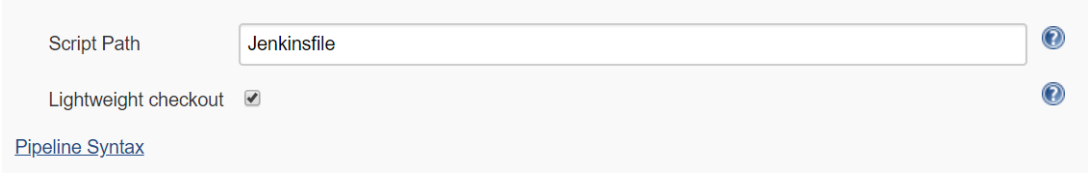
```
[root@jenkins ~]# git clone https://github.com/deepanredhat/simple-java-maven-app
Cloning into 'simple-java-maven-app'...
remote: Enumerating objects: 118, done.
remote: Total 118 (delta 0), reused 0 (delta 0), pack-reused 118
Receiving objects: 100% (118/118), 15.84 KiB | 0 bytes/s, done.
Resolving deltas: 100% (39/39), done.
```

**Step 2:- Create pipeline project on Jenkins.**




@@Click the Pipeline tab → scroll down to the Pipeline section →From the Definition field, choose the Pipeline script from SCM option→ From the SCM field, choose Git → enter GIT repo.






@@Click Save to save your new Pipeline project. You're now ready to begin creating your Jenkinsfile, which you'll be checking into your locally cloned Git repository

**Step 3:- install docker service.**

```
#yum install -y yum-utils device-mapper-persistent-data lvm2
#yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
#yum install docker-ce docker-ce-cli containerd.io -y
#systemctl start docker && systemctl enable docker && systemctl status docker
#chmod 777 /var/run/docker.sock && ls -lrt /var/run/docker.sock
```

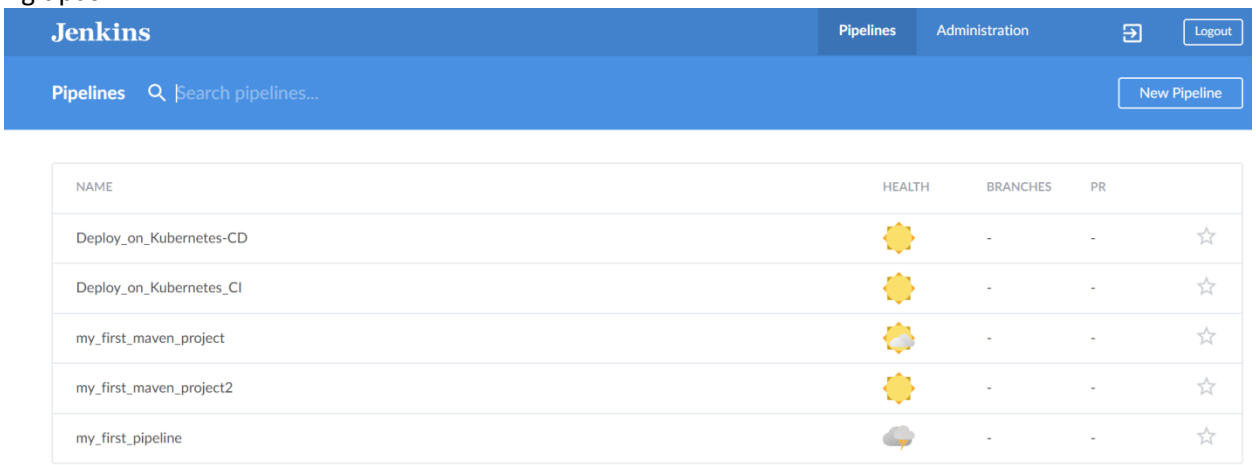
**Step4:- Create your initial Pipeline as a Jenkinsfile.**

```
#cd /root/simple-java-maven-app
#vim Jenkinsfile
#cat Jenkinsfile
pipeline {
  agent {
    docker {
      image 'maven:3-alpine'
      args '-v /root/.m2:/root/.m2'
    }
  }
  stages {
    stage('Build') {
      steps {
        sh 'mvn -B -DskipTests clean package'
      }
    }
  }
}
```

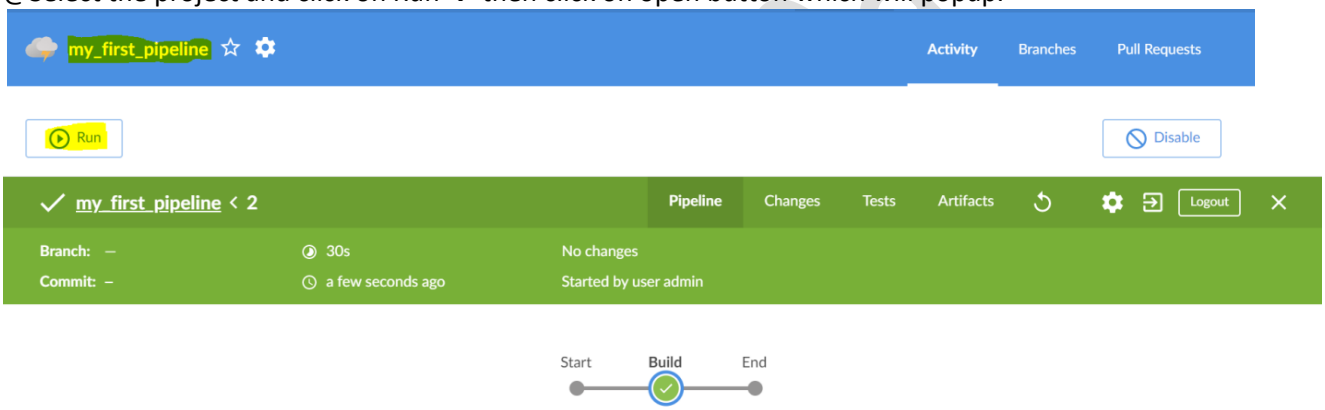
**Output:-**

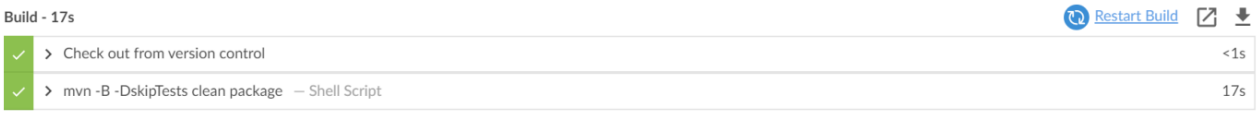
```
[root@jenkins simple-java-maven-app]# cat Jenkinsfile
pipeline {
    agent {
        docker {
            image 'maven:3-alpine'
            args '-v /root/.m2:/root/.m2'
        }
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B -DskipTests clean package'
            }
        }
    }
}
[root@jenkins simple-java-maven-app]#
```

# git add .
# git commit -m "Add initial Jenkinsfile"
# git push




@Select the project and click on Run → then click on open button which will popup.




**<u>Step 5:- Add a test stage to your Pipeline.</u>**

```
# cat Jenkinsfile
pipeline {
  agent {
    docker {
      image 'maven:3-alpine'
      args '-v /root/.m2:/root/.m2'
    }
  }
```

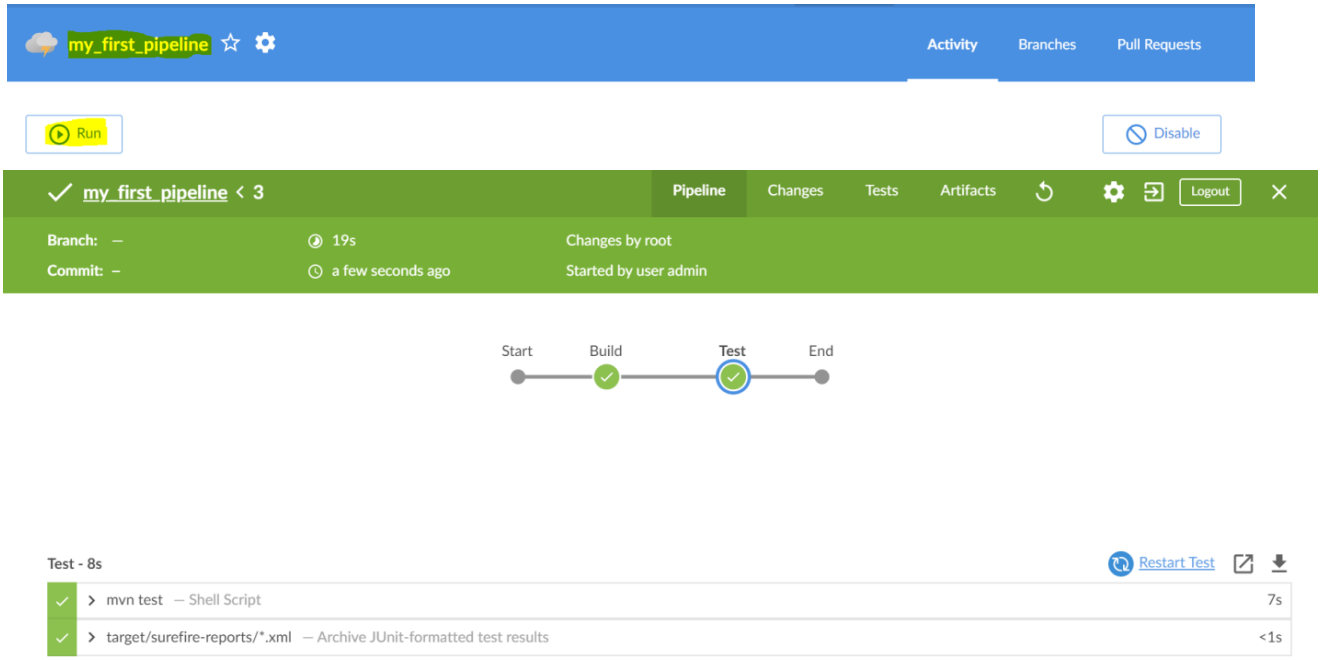
```
  stages {
    stage('Build') {
      steps {
        sh 'mvn -B -DskipTests clean package'
      }
    }
    stage('Test') {
      steps {
        sh 'mvn test'
      }
      post {
        always {
          junit 'target/surefire-reports/*.xml'
        }
      }
    }
  }
}
```

---

**Output:-**

```
[root@jenkins simple-java-maven-app]# cat Jenkinsfile
pipeline {
    agent {
        docker {
            image 'maven:3-alpine'
            args '-v /root/.m2:/root/.m2'
        }
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B -DskipTests clean package'
             }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
            post {
                always {
                    junit 'target/surefire-reports/*.xml'
                }
            }
        }
    }
}
[root@jenkins simple-java-maven-app]#
```

@ click on Run → then click on open button which will pop up on right.



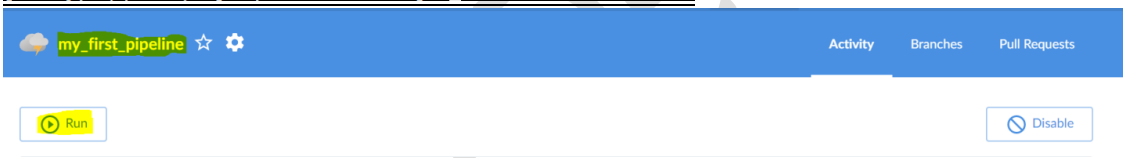

**Step 6:-Add a final deliver stage to your Pipeline.**

```
# cat Jenkinsfile
pipeline {
  agent {
    docker {
      image 'maven:3-alpine'
      args '-v /root/.m2:/root/.m2'
    }
  }
  stages {
    stage('Build') {
      steps {
        sh 'mvn -B -DskipTests clean package'
      }
    }
    stage('Test') {
      steps {
        sh 'mvn test'
      }
      post {
        always {
          junit 'target/surefire-reports/*.xml'
        }
      }
    }
    stage('Deliver') {
      steps {
        sh './jenkins/scripts/deliver.sh'
      }
```

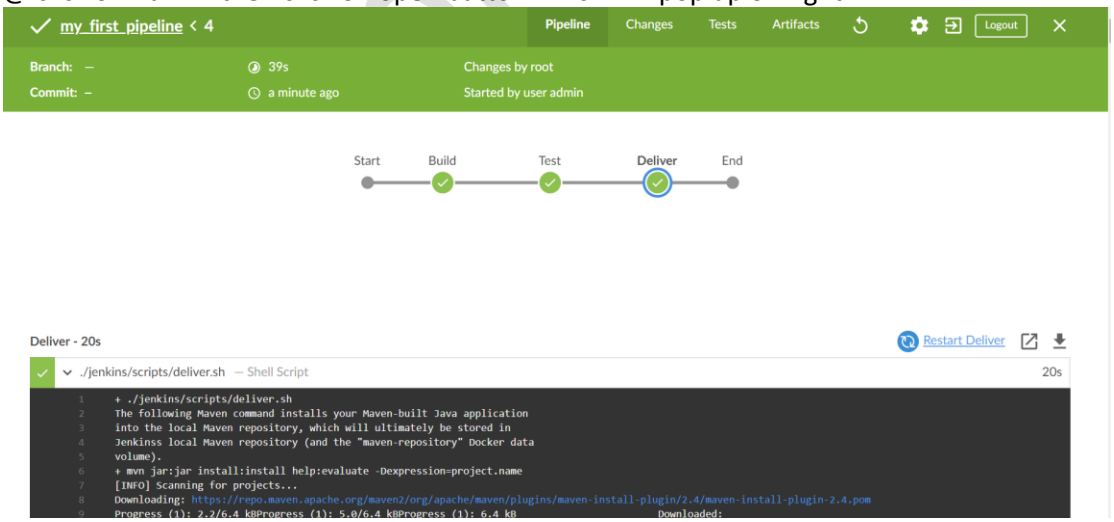
```
        }
    }
}
```

**Output:-**

```
[root@jenkins simple-java-maven-app]# cat Jenkinsfile
pipeline {
    agent {
        docker {
            image 'maven:3-alpine'
            args '-v /root/.m2:/root/.m2'
        }
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B -DskipTests clean package'
             }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
            post {
                always {
                    junit 'target/surefire-reports/*.xml'
                }
            }
        }
    }
}
```




@ click on Run → then click on open button which will pop up on right.




Note:-The "Build", "Test" and "Deliver" stages you created above are the basis for building more complex Java applications with Maven in Jenkins, as well as Java and Maven applications that integrate with other technology stacks. Because Jenkins is extremely extensible, it can be modified and configured to handle practically any aspect of build orchestration and automation.