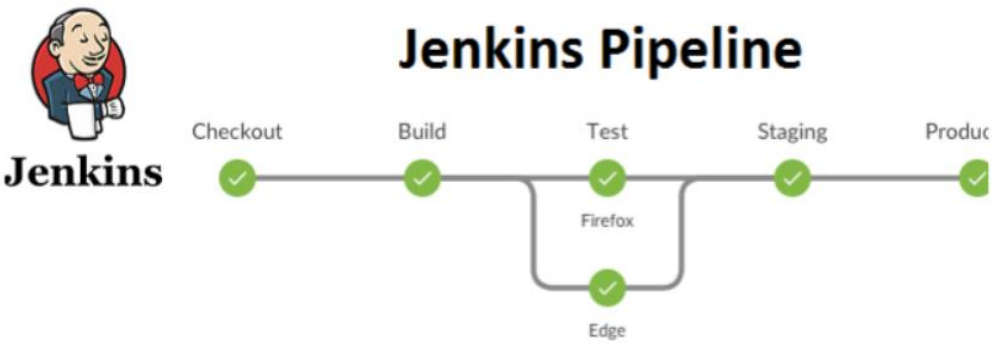


Jenkins Pipeline

Introduction:

A step-by-step guide to creating the Jenkins pipeline.

What is Jenkins Pipeline?

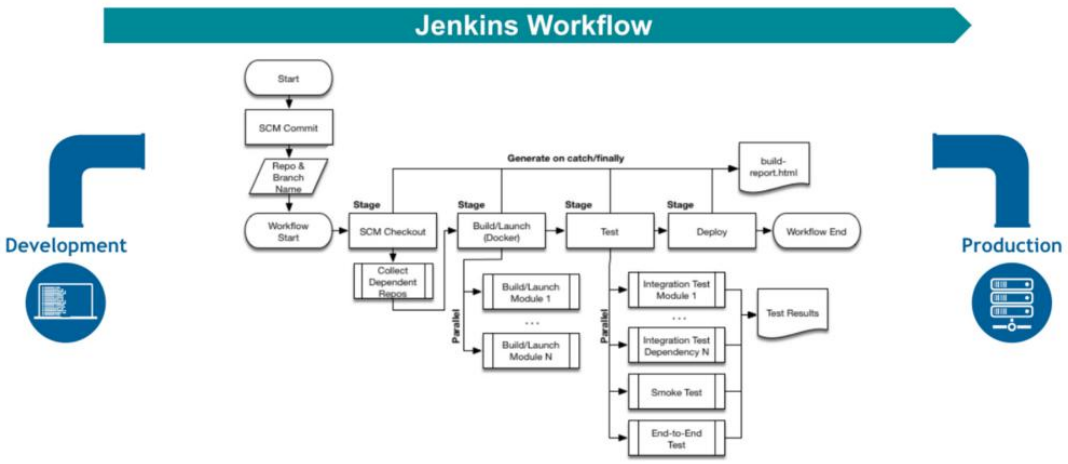
A Jenkins pipeline consists of several states or stages, and they get executed in a sequence one after the other. Jenkins File is a simple text file that is used to create a pipeline as code in Jenkins. It contains code in Groovy Domain Specific Language (DSL), which is simple to write and human-readable.

Either you can run Jenkins File separately, or you can run the pipeline code from Jenkins Web UI also. There are two ways you can create a pipeline using Jenkins.

- ✓ **Declarative** – a new way of creating Jenkins Pipeline. Here you write groovy code containing “pipeline” blocks, which is checked into an SCM (Source Code Management)

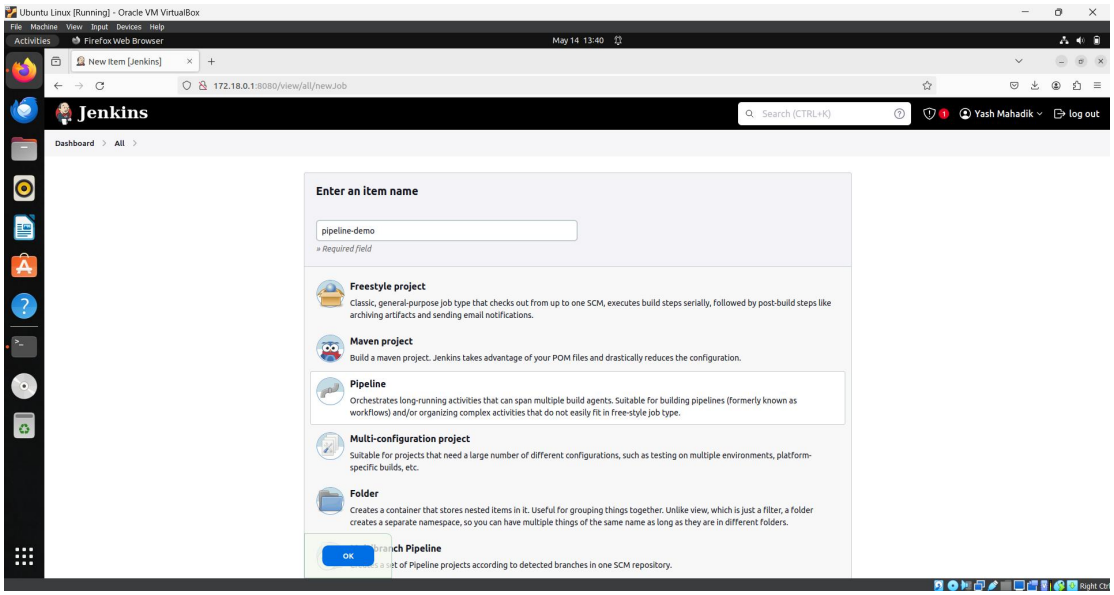
- ✓ **Scripted** – way of writing groovy code where the code is defined inside “node” blocks.

Workflow of Pipeline:

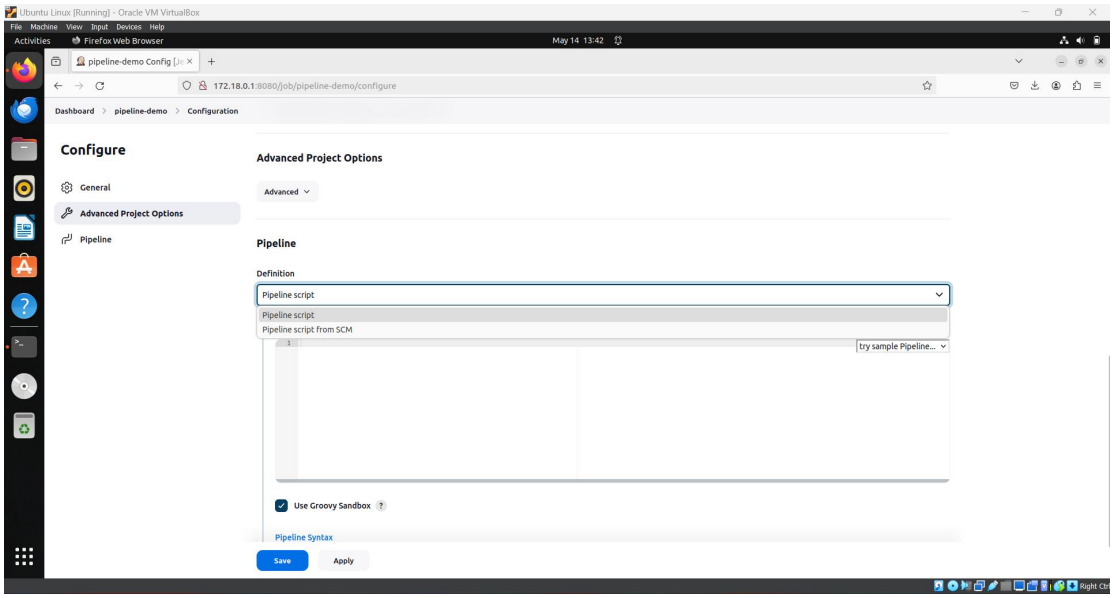


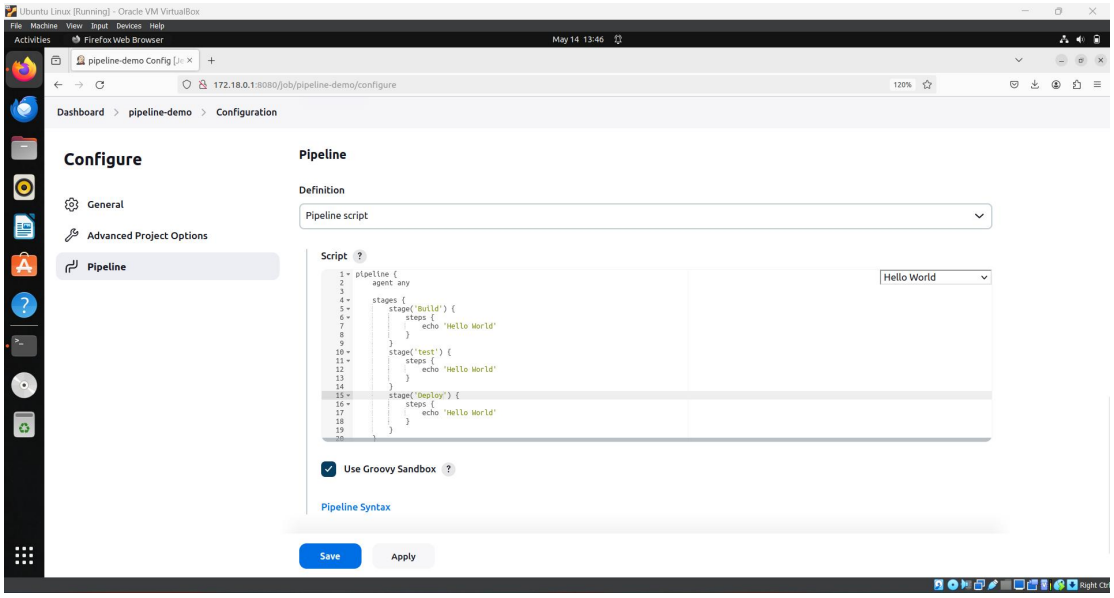
A step-by-step guide to Create a Jenkins Pipeline:

1) lets create a declarative pipeline. On the Jenkins dashboard, click on New Item. Then enter an item name, for example, ‘pipeline-demo’ and select the ‘Pipeline’ project. Then click on, OK.

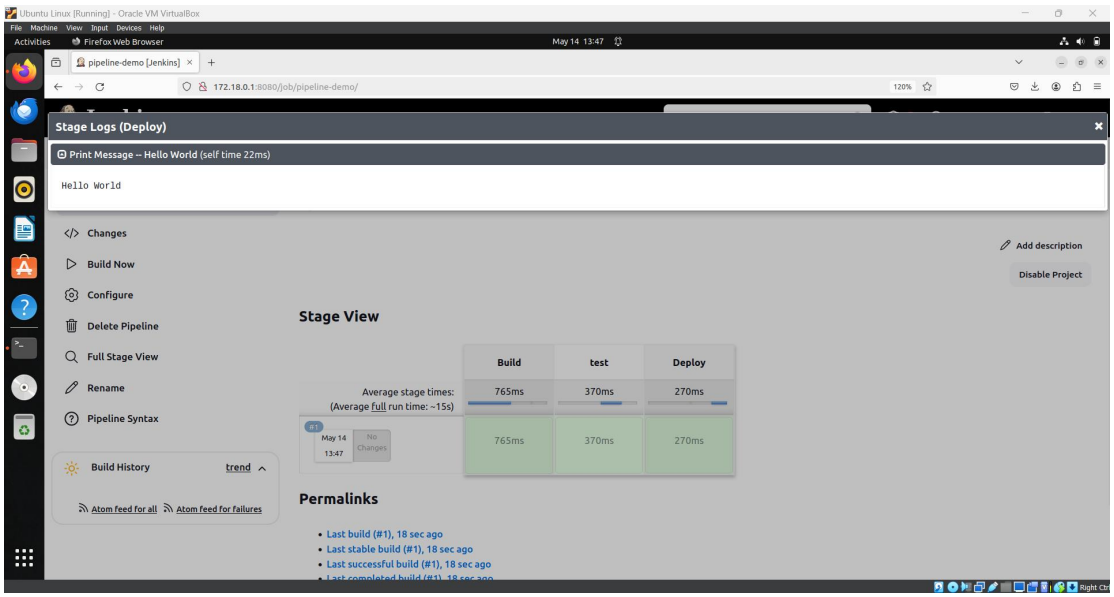


2) Click on the Pipeline tab as shown in the image below, and put your JenkinsFile code (Groovy Code) here.





3) Click on Build and you will see the stage view of pipeline successfully.



The **pipeline** block consists of all the instructions to build, test, and deliver software. It is the key component of a Jenkins Pipeline.

An **agent** is assigned to execute the pipeline on a node and allocate a workspace for the pipeline.

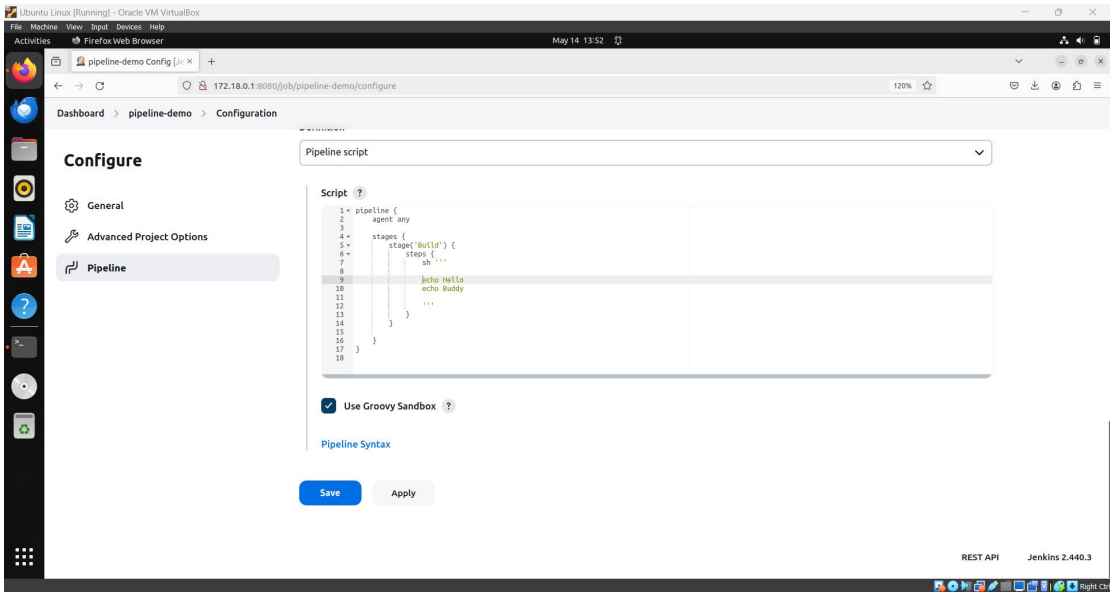
A **stage** is a block that has steps to build, test, and deploy the application. Stages are used to visualize the Jenkins Pipeline processes.

A **step** is a single task to be performed, for example, create a directory, run a docker image, delete a file, etc.

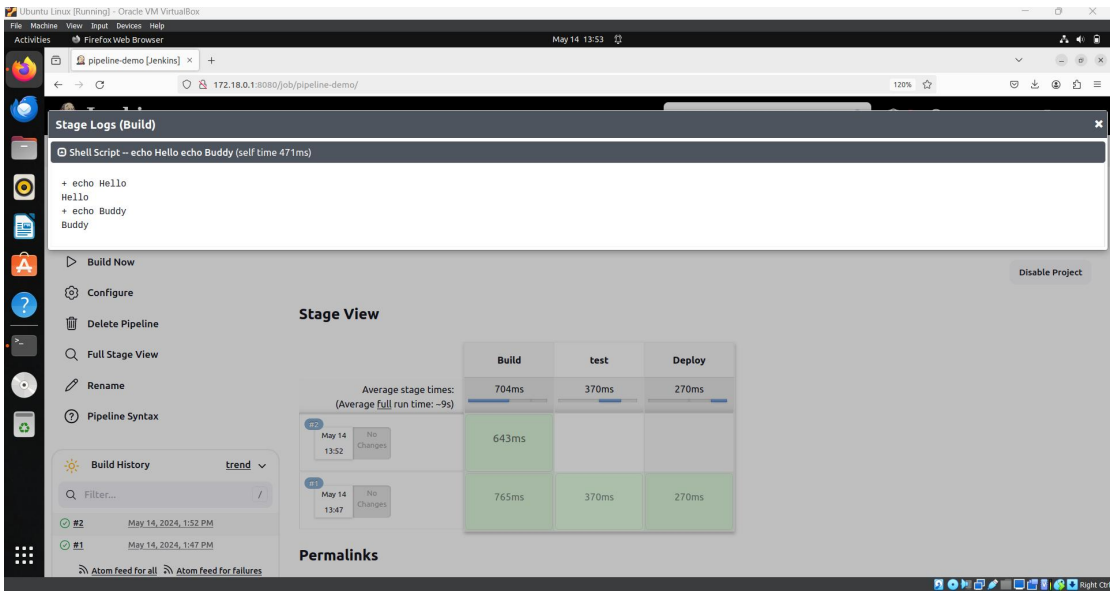
#Multi-step pipeline setup

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        sh '''

        echo Hello
```

```
            echo Buddy
            echo $NAME $AGE
            '''
        }
    }

  }
}
```

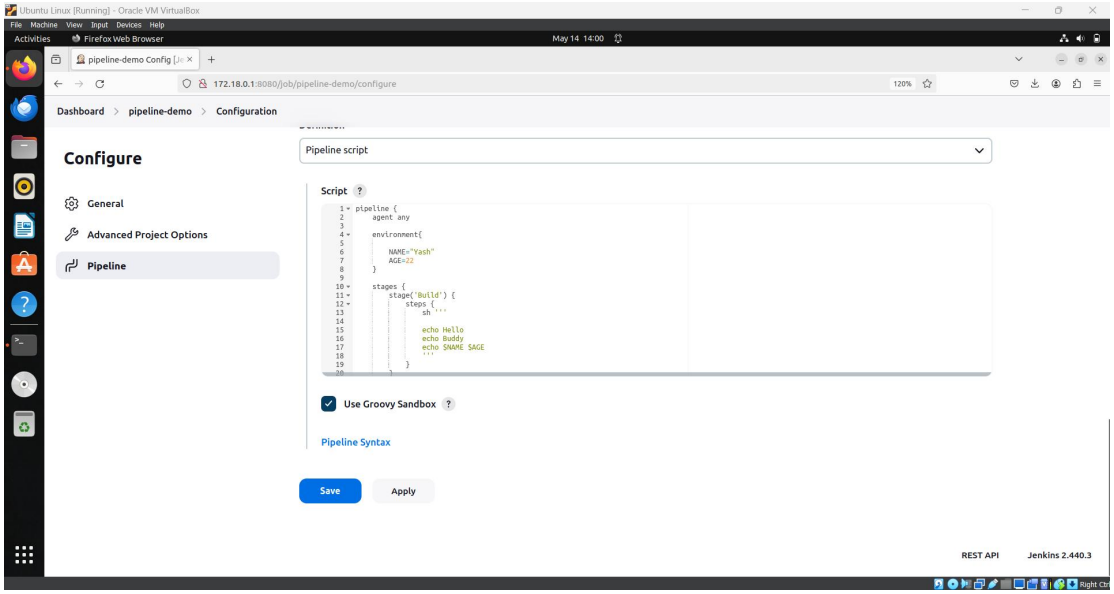


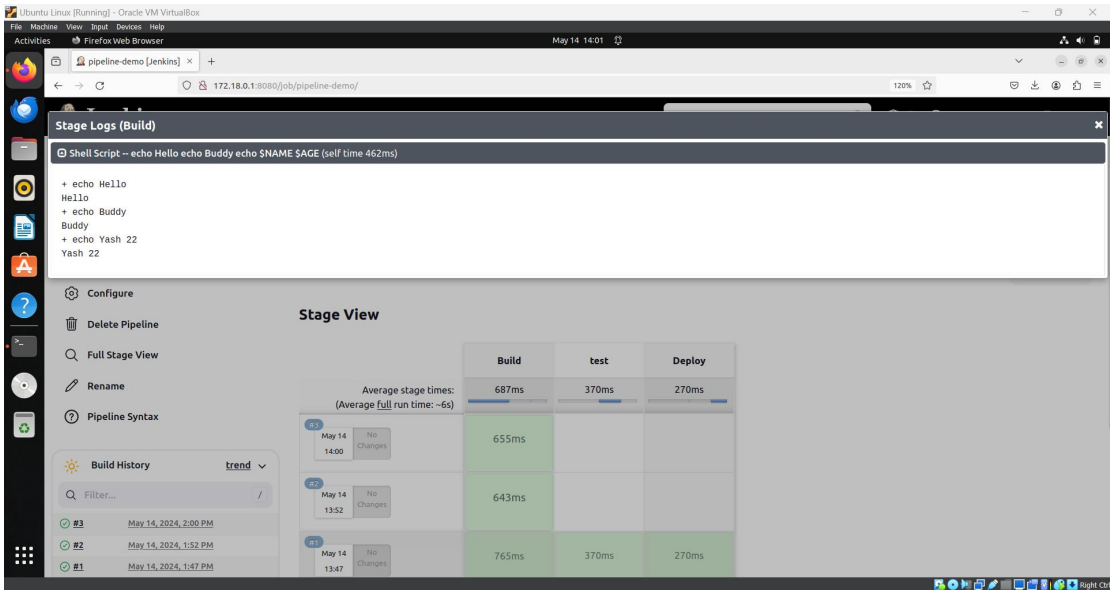
On Building you can see the stage view successfully.



#Environment Var Jenkins

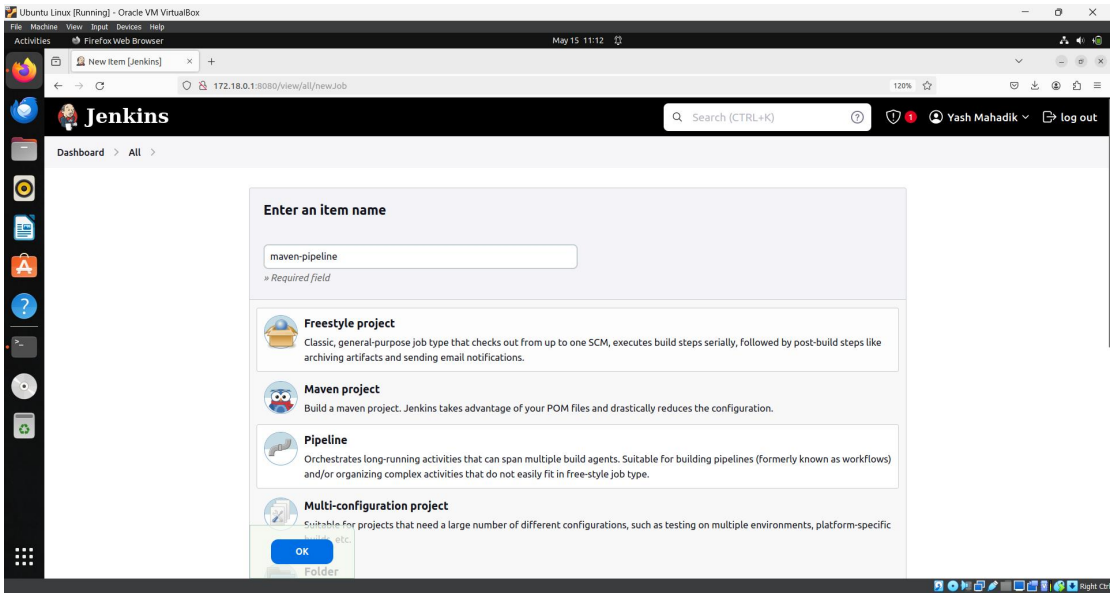
```
pipeline {
  agent any

  environment{

    NAME="Yash"
    AGE=22
  }

  stages {
    stage('Build') {
      steps {
        sh '''

        echo Hello
        echo Buddy
        echo $NAME $AGE
        '''
      }
    }

  }
}
```





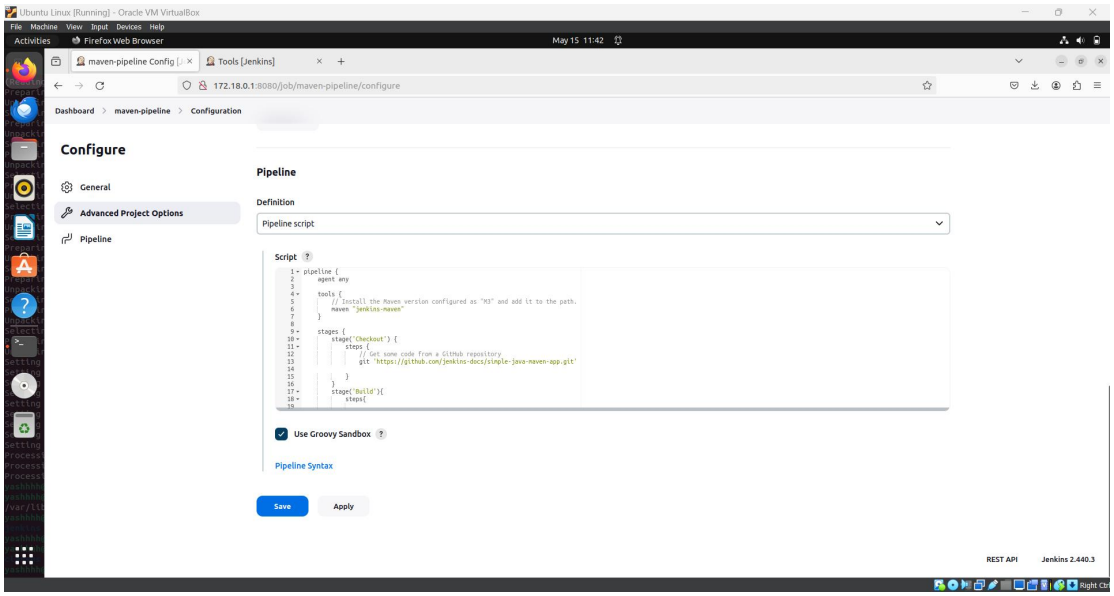
#Maven -Pipeline Build

1) Lets create a declarative pipeline. On the Jenkins dashboard, click on New Item. Then enter an item name, for example, ‘maven-pipeline’ and select the ‘Pipeline’ project. Then click on, OK.

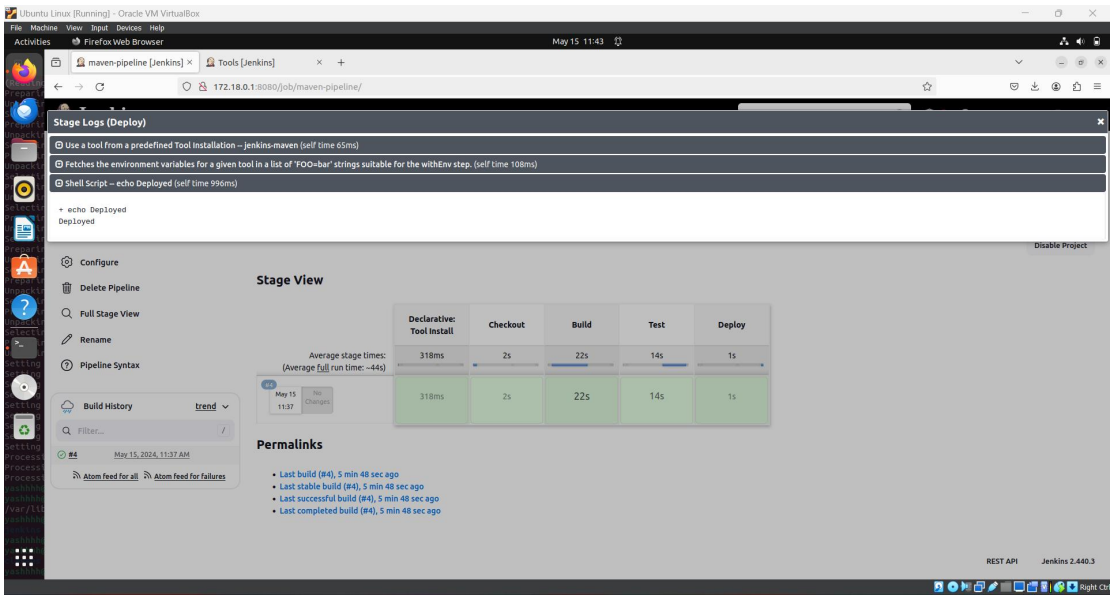


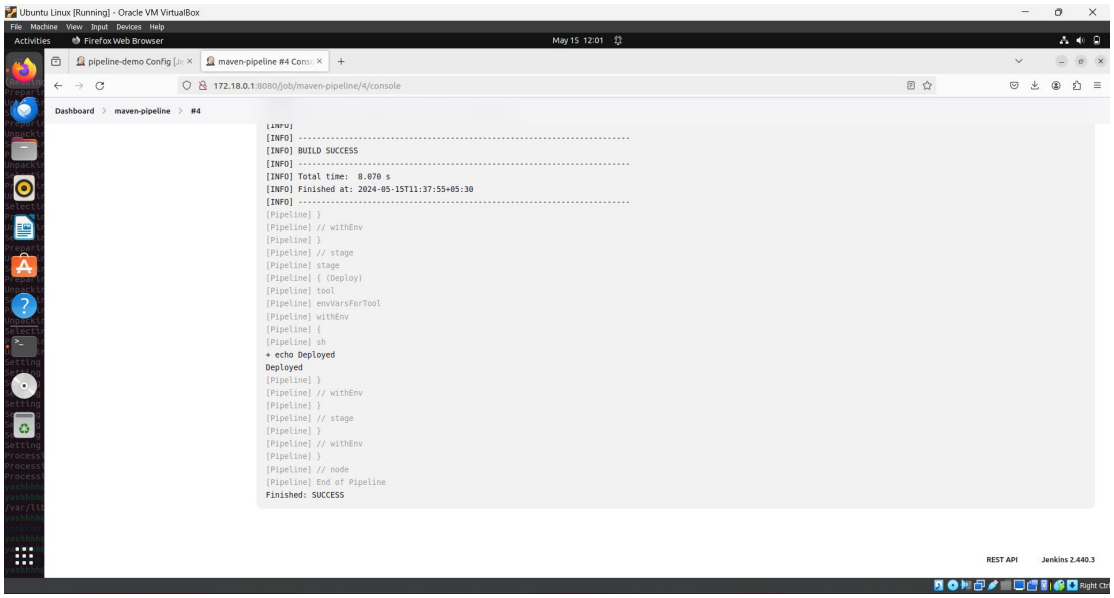
2)Click on the Pipeline tab as shown in the image below, and put your JenkinsFile code (Groovy Code) here.

```
pipeline {
  agent any

  tools {
    // Install the Maven version configured as "M3" and add it to the path.
    maven "jenkins-maven"
  }

  stages {
    stage('Checkout') {
      steps {
        // Get some code from a GitHub repository (gitlink)
        git 'https://github.com/jenkins-docs/simple-java-maven-app.git'

      }
    }
    stage('Build'){
      steps{

        sh 'mvn clean package'
      }

    }
    stage('Test'){
      steps{

        sh 'mvn test'
      }

    }
    stage('Deploy'){
      steps{

        sh 'echo Deployed'
      }

    }

  }
}
}



]
```



3)After Building the pipeline successfully you can see the stage view even you cane see the output in Console output.





Conclusion:

Building a Jenkins pipeline empowers you to transform your software delivery process into a well-orchestrated, automated workflow that accelerates innovation and drives business value. So, embrace the power of pipelines and embark on a journey towards more efficient and reliable software delivery.