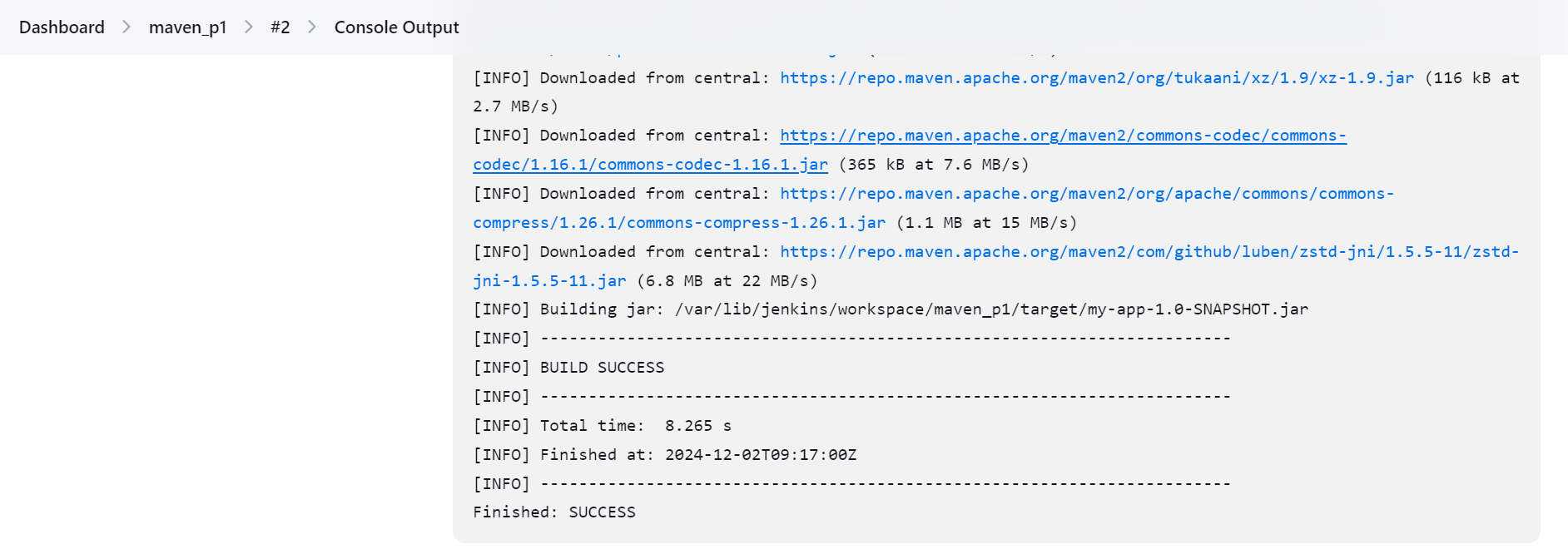
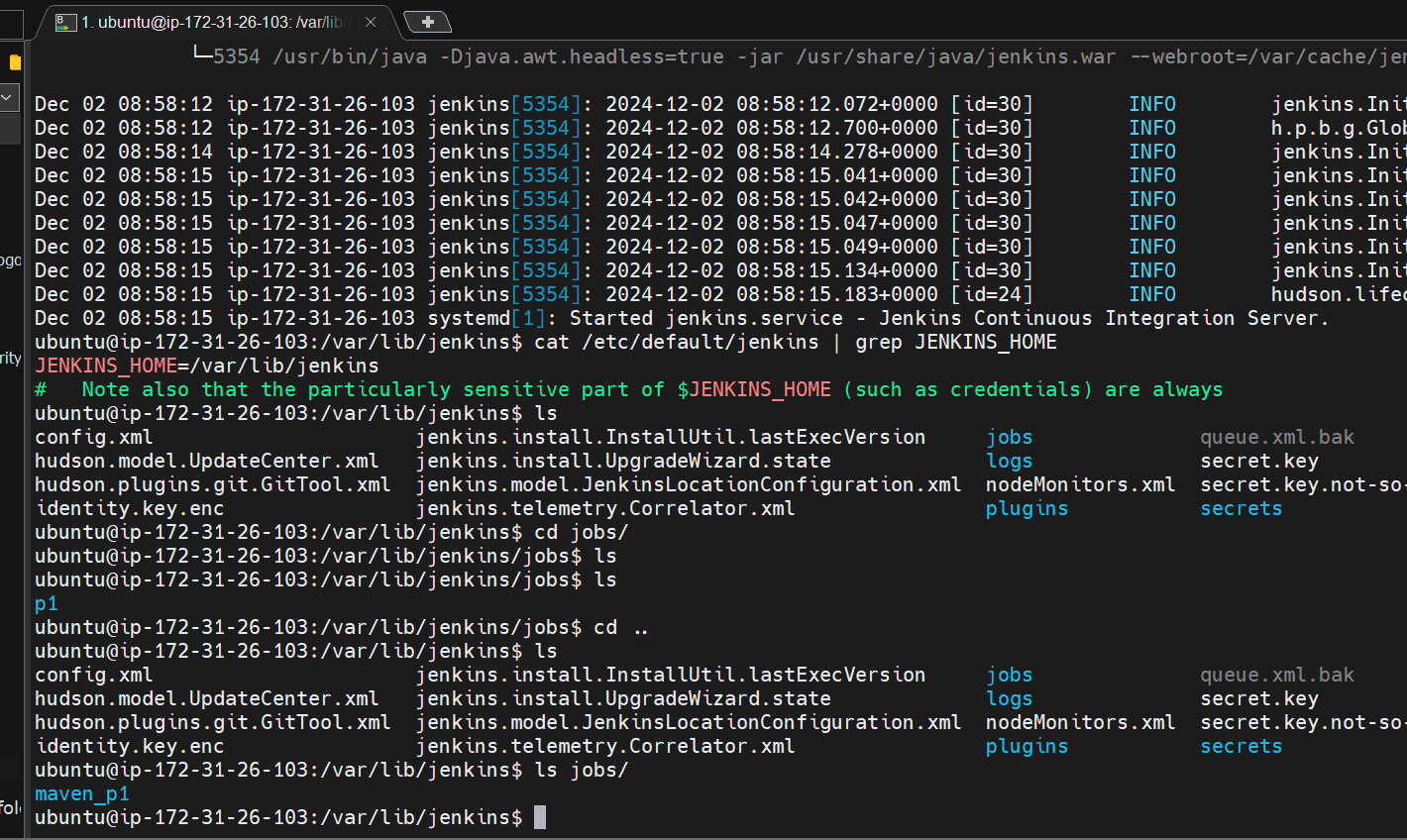
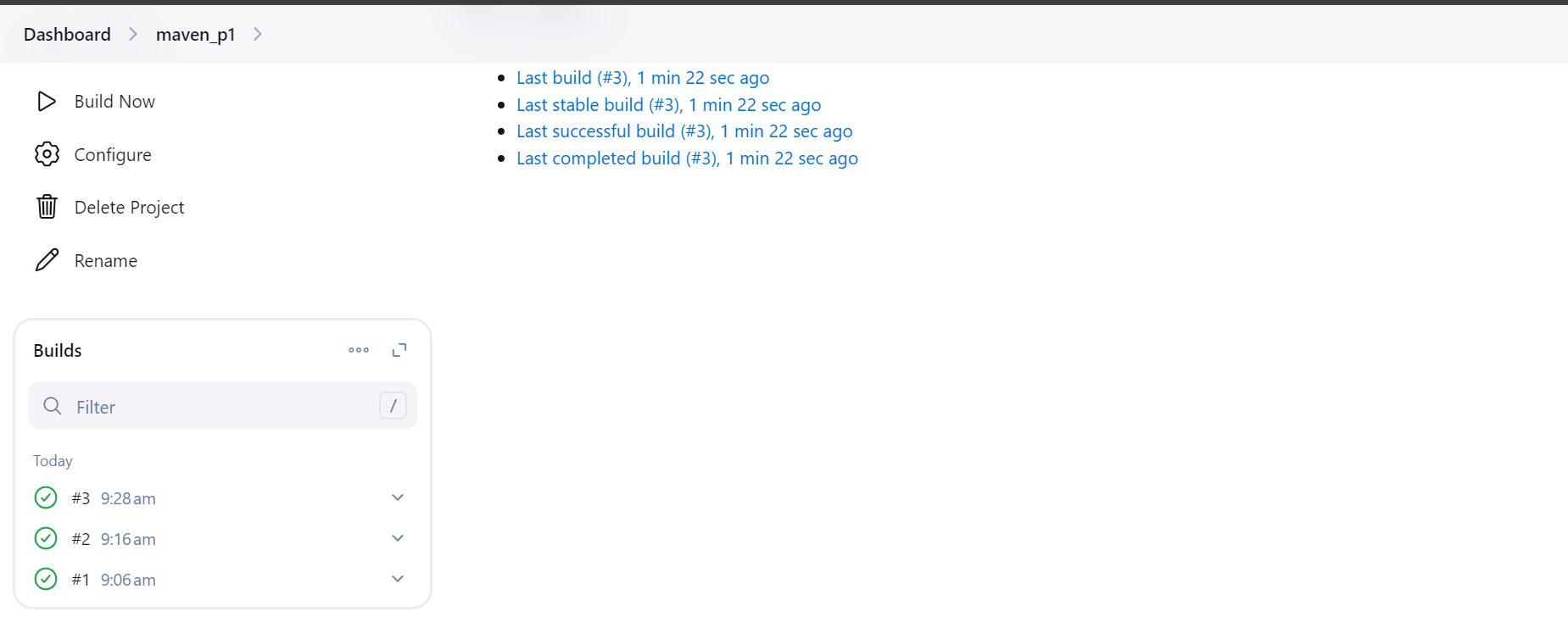
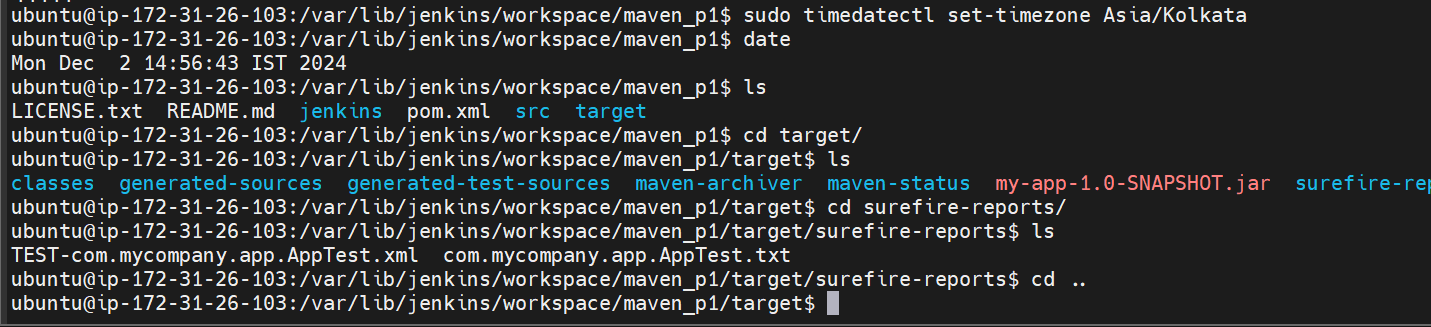
Initial creation and installation of the Jenkins jobs

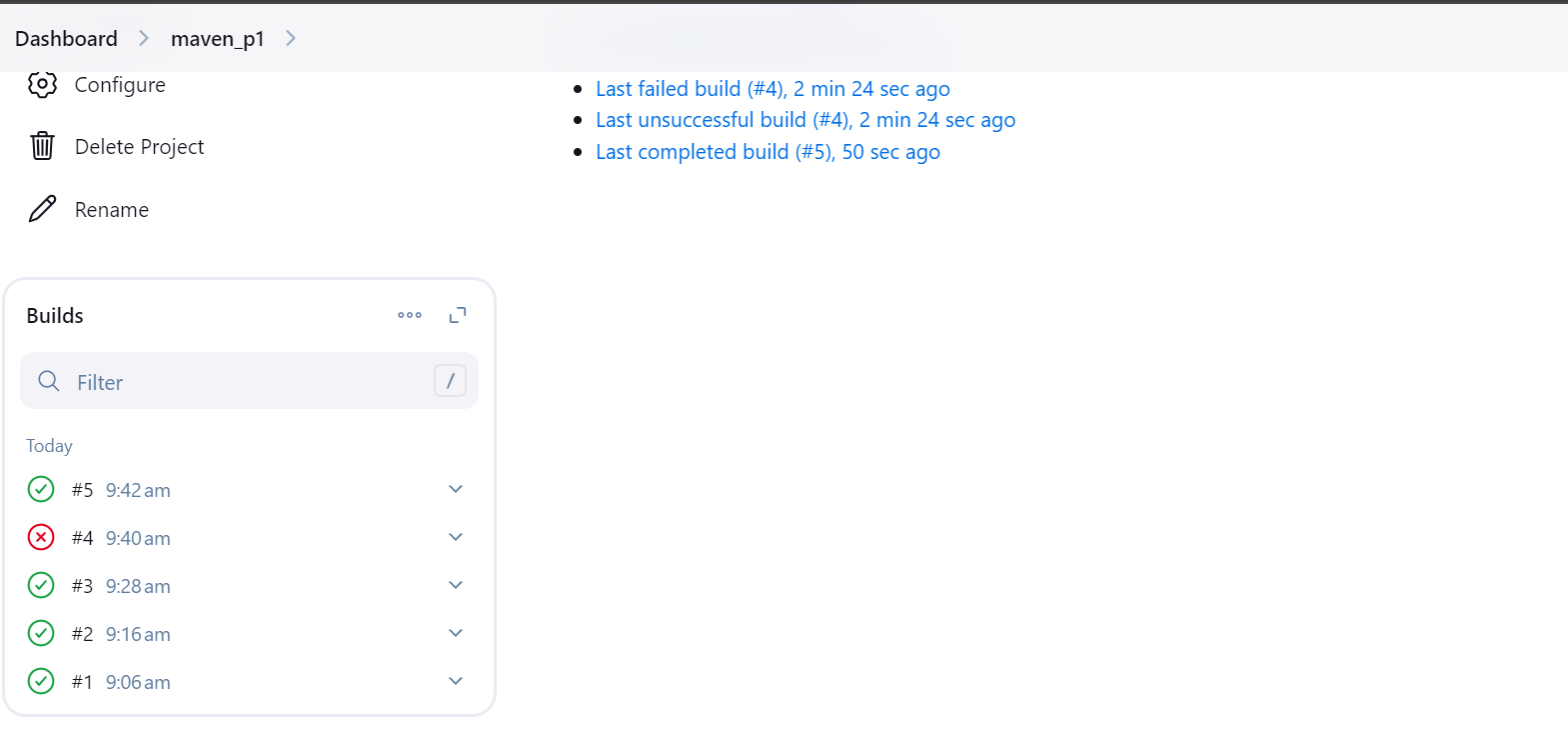


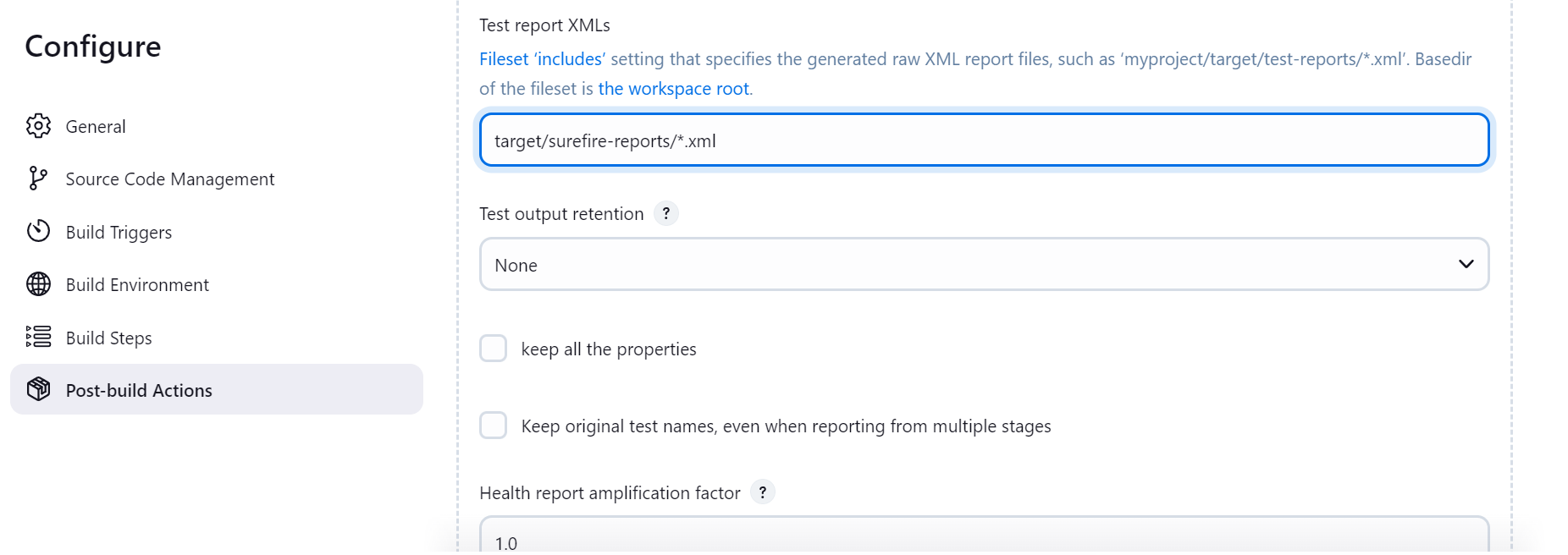


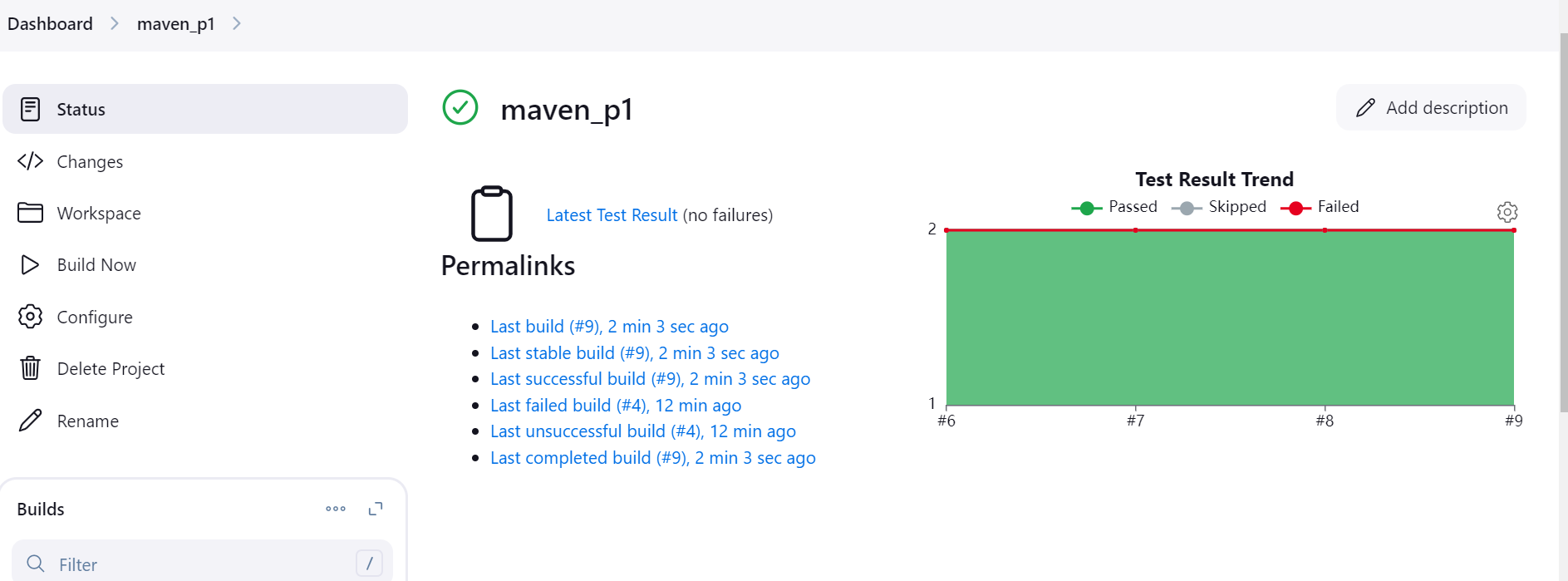


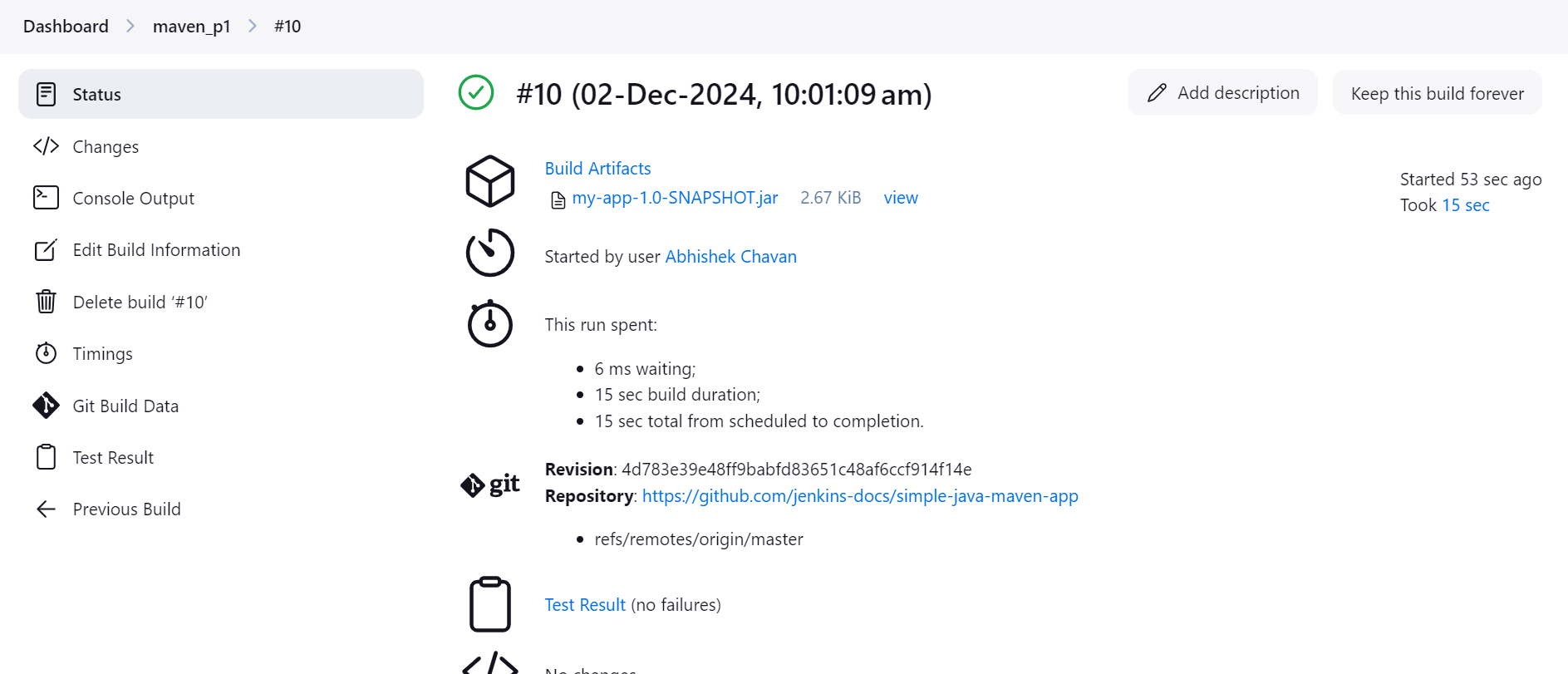
Simple build using maven application

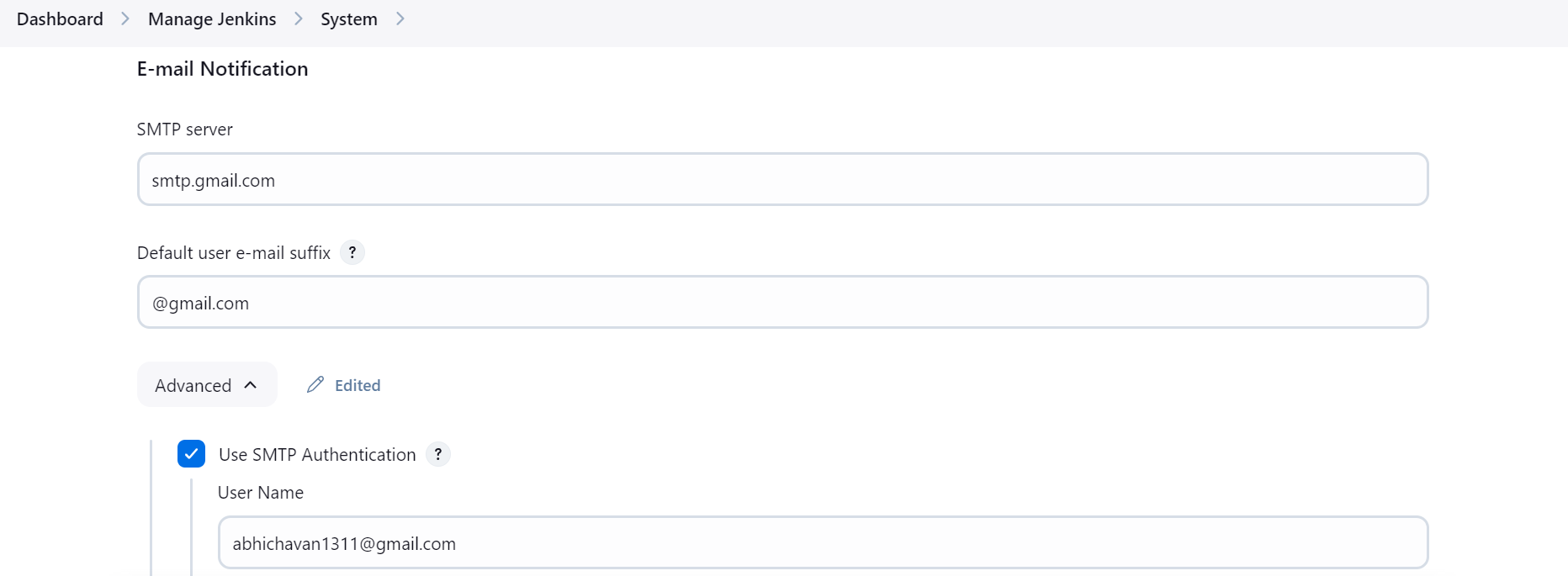
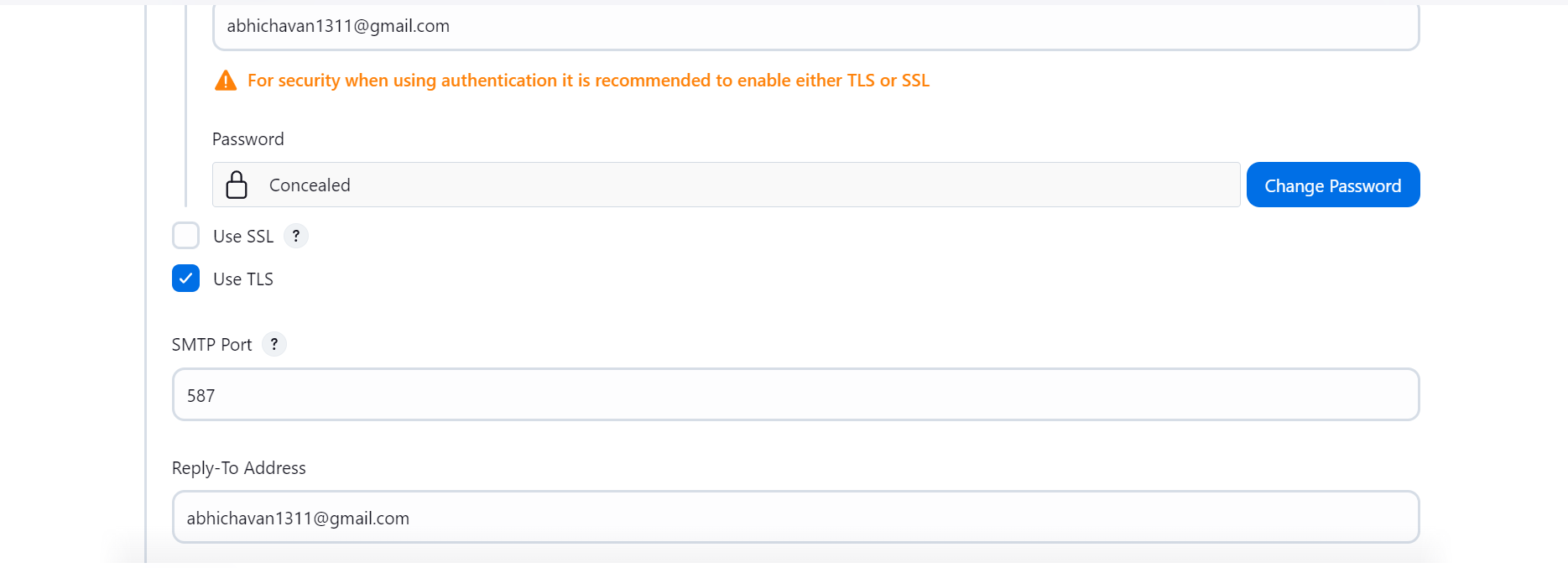


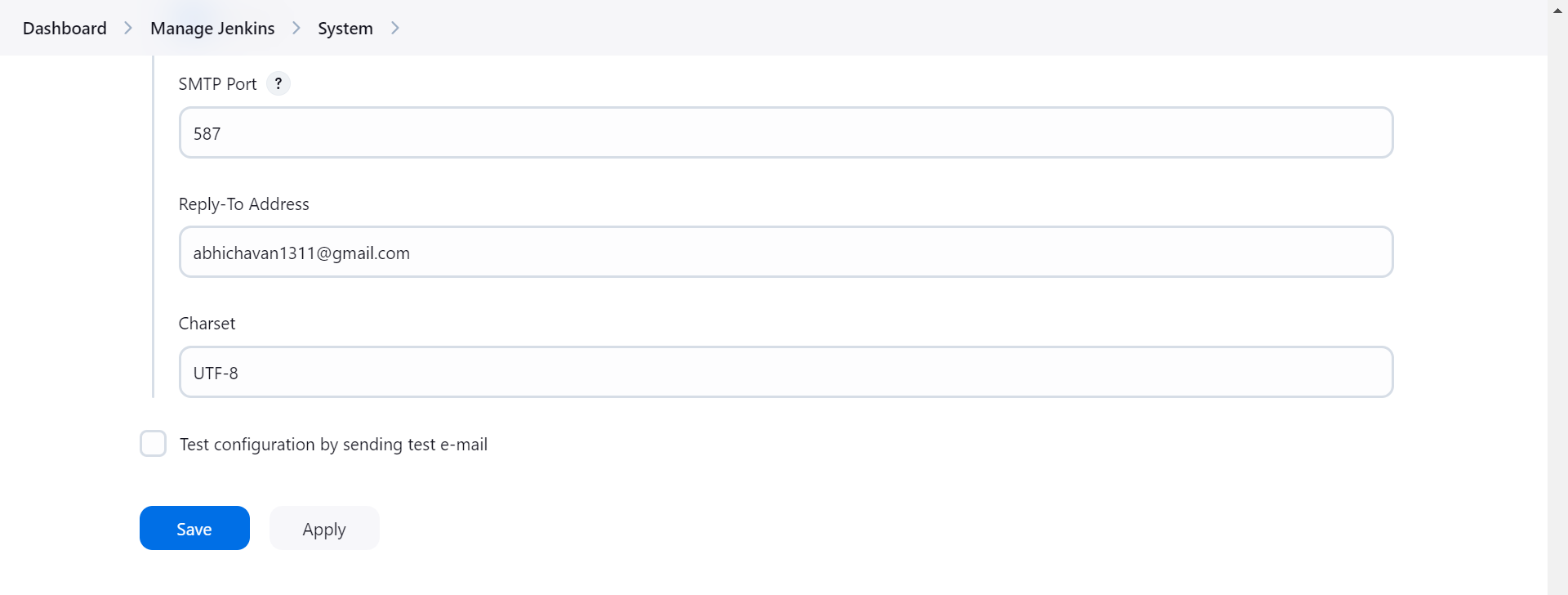
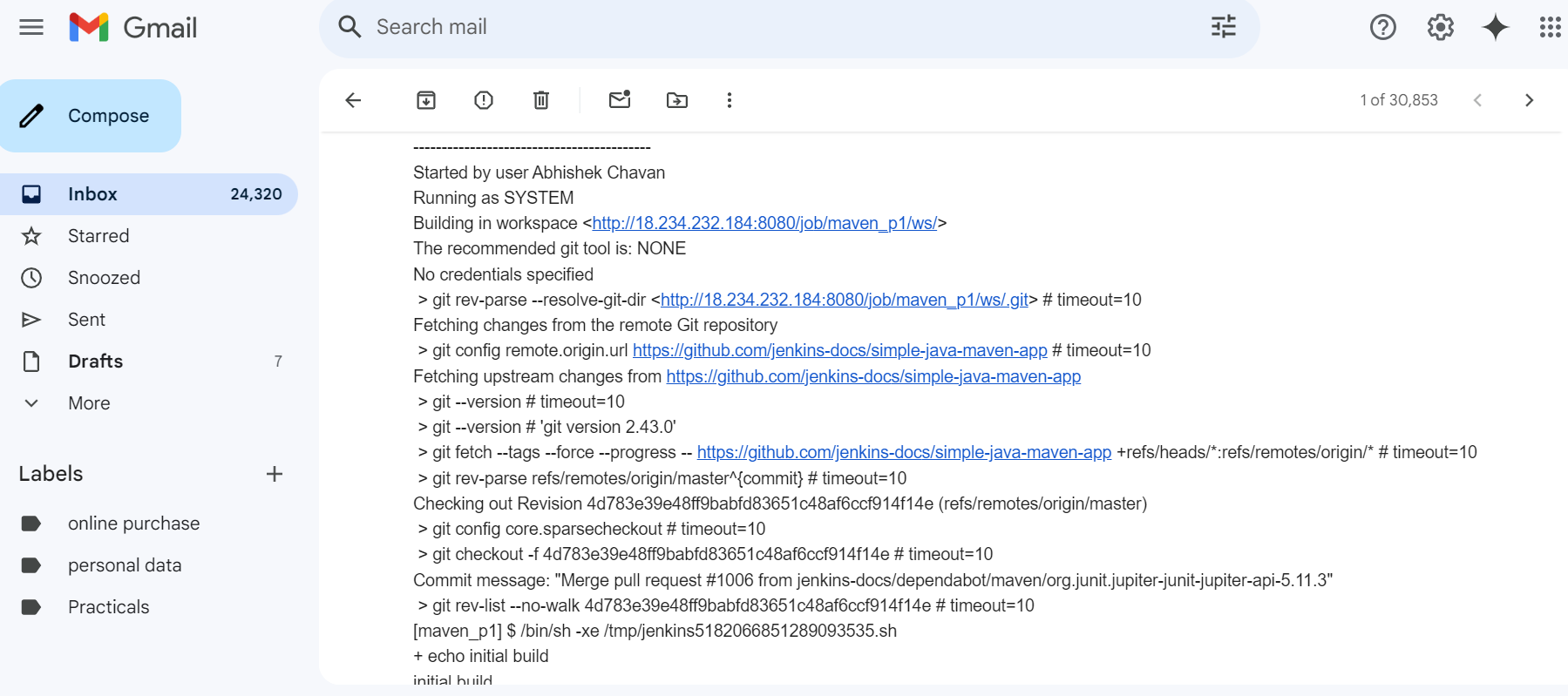
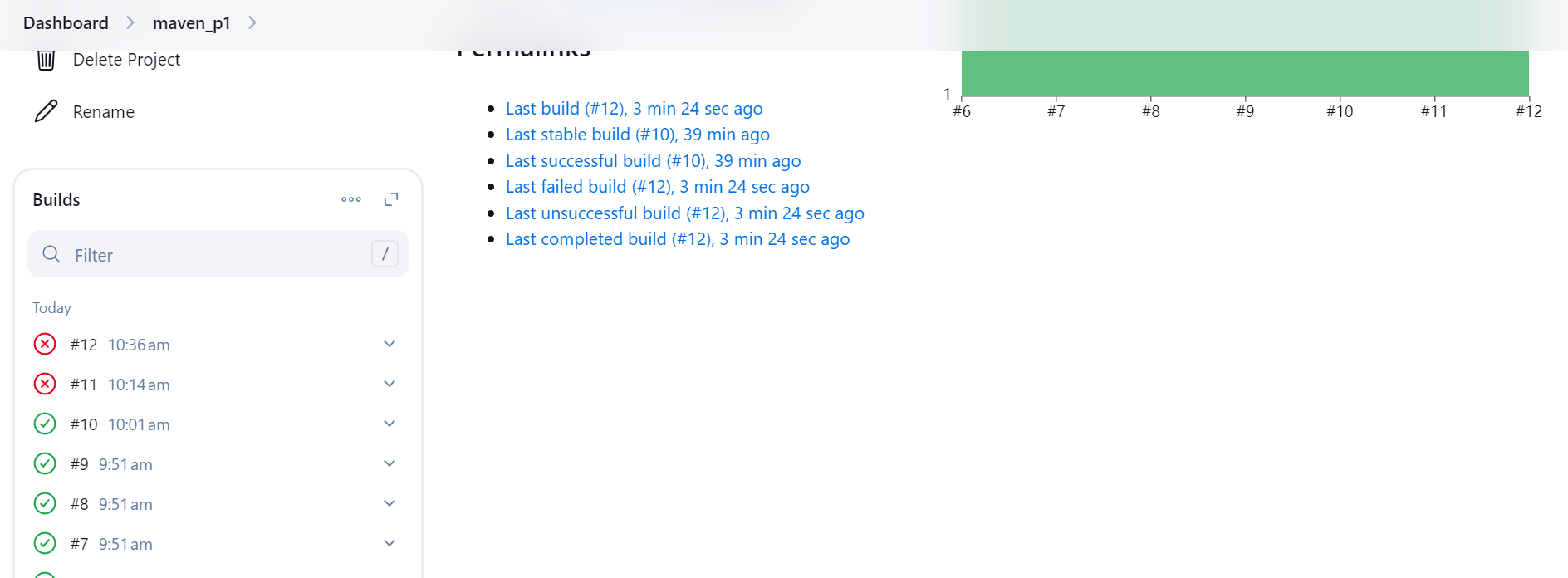


Publish test case reports on Jenkins UI graph  
  




Archive the artifacts in ci-cd itself using post build action  
  
  
  
  
Trigger email if there is unstable build

Authenticate yourself as valid Jenkins user to different remote private repo.  
  
Generate key  
ubuntu@ip-172-31-26-103:~/.ssh$ ssh-keygen

Generating public/private ed25519 key pair.

Enter file in which to save the key (/home/ubuntu/.ssh/id\_ed25519):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/ubuntu/.ssh/id\_ed25519

Your public key has been saved in /home/ubuntu/.ssh/id\_ed25519.pub

The key fingerprint is:

SHA256:1eSab/aujhDFzNZQcH0rJzdY4ZNewuJrmwCECHUqaME ubuntu@ip-172-31-26-103

The key's randomart image is:

+--[ED25519 256]--+

|..... . oo+. ..|

| E.. + . + B .o.o|

|... o . . B = ==o|

|. . . + + =.\*o|

| S o . =..|

| o . . |

| . . \* |

| . \* + |

| ..=oo |

+----[SHA256]-----+

ubuntu@ip-172-31-26-103:~/.ssh$

ubuntu@ip-172-31-26-103:~/.ssh$

ubuntu@ip-172-31-26-103:~/.ssh$ ls -ltra

total 20

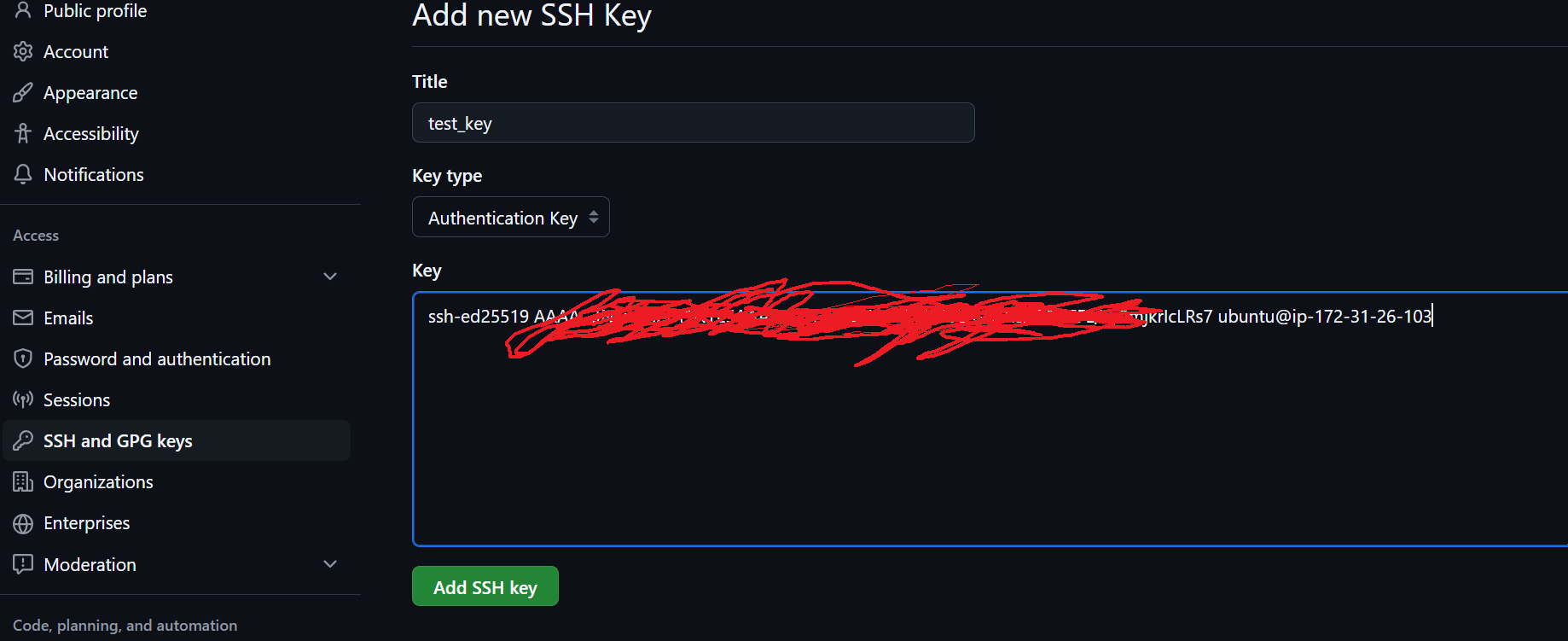
-rw------- 1 ubuntu ubuntu 389 Dec 2 11:31 authorized\_keys

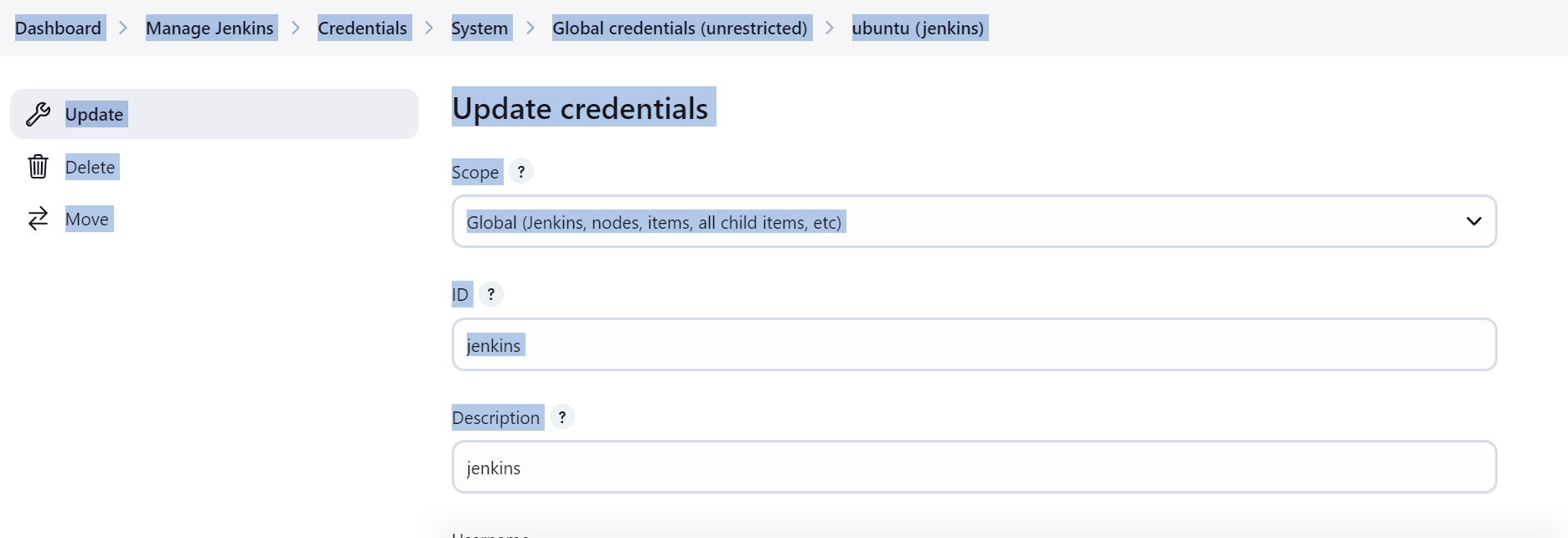
drwxr-x--- 4 ubuntu ubuntu 4096 Dec 2 14:56 ..

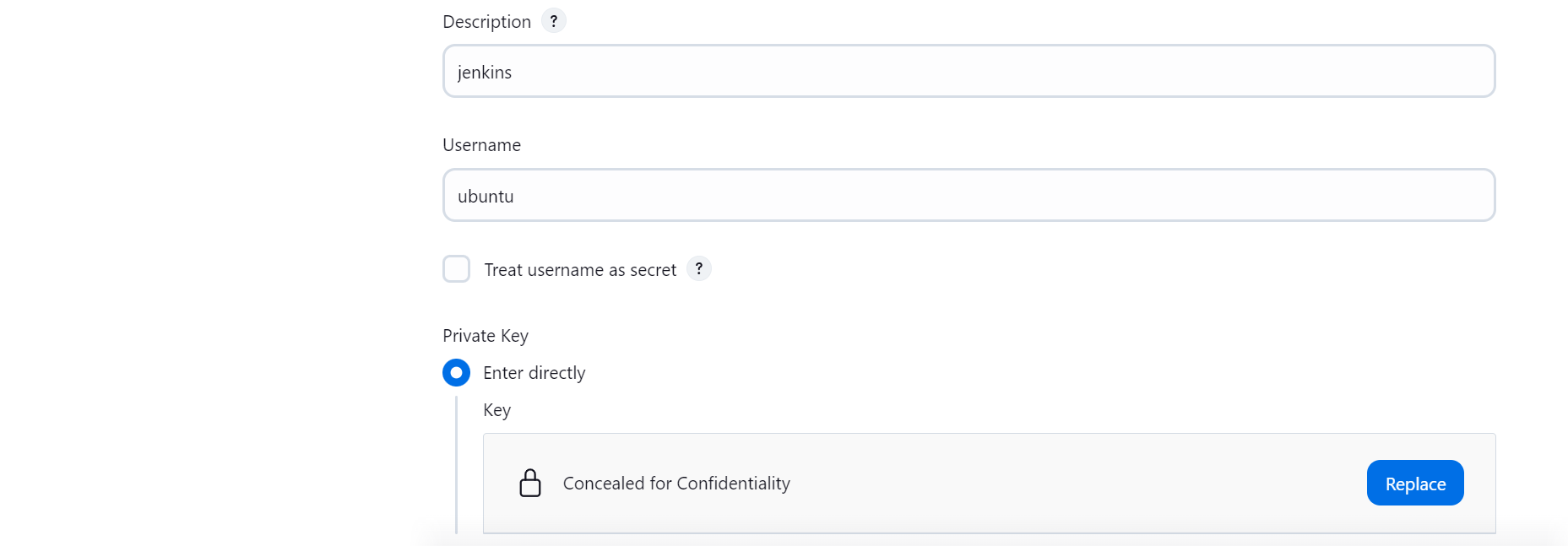
-rw-r--r-- 1 ubuntu ubuntu 105 Dec 2 17:14 id\_ed25519.pub

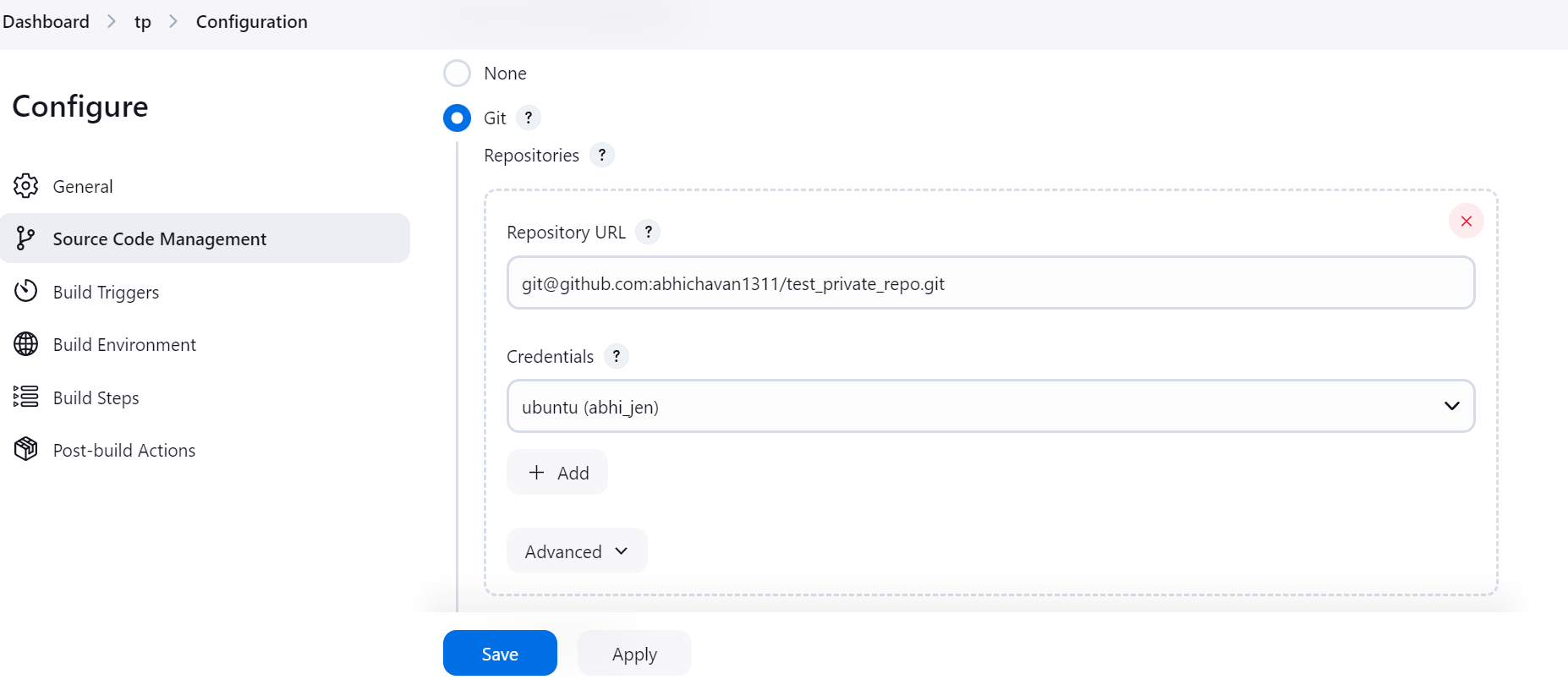
-rw------- 1 ubuntu ubuntu 419 Dec 2 17:14 id\_ed25519

drwx------ 2 ubuntu ubuntu 4096 Dec 2 17:14 .

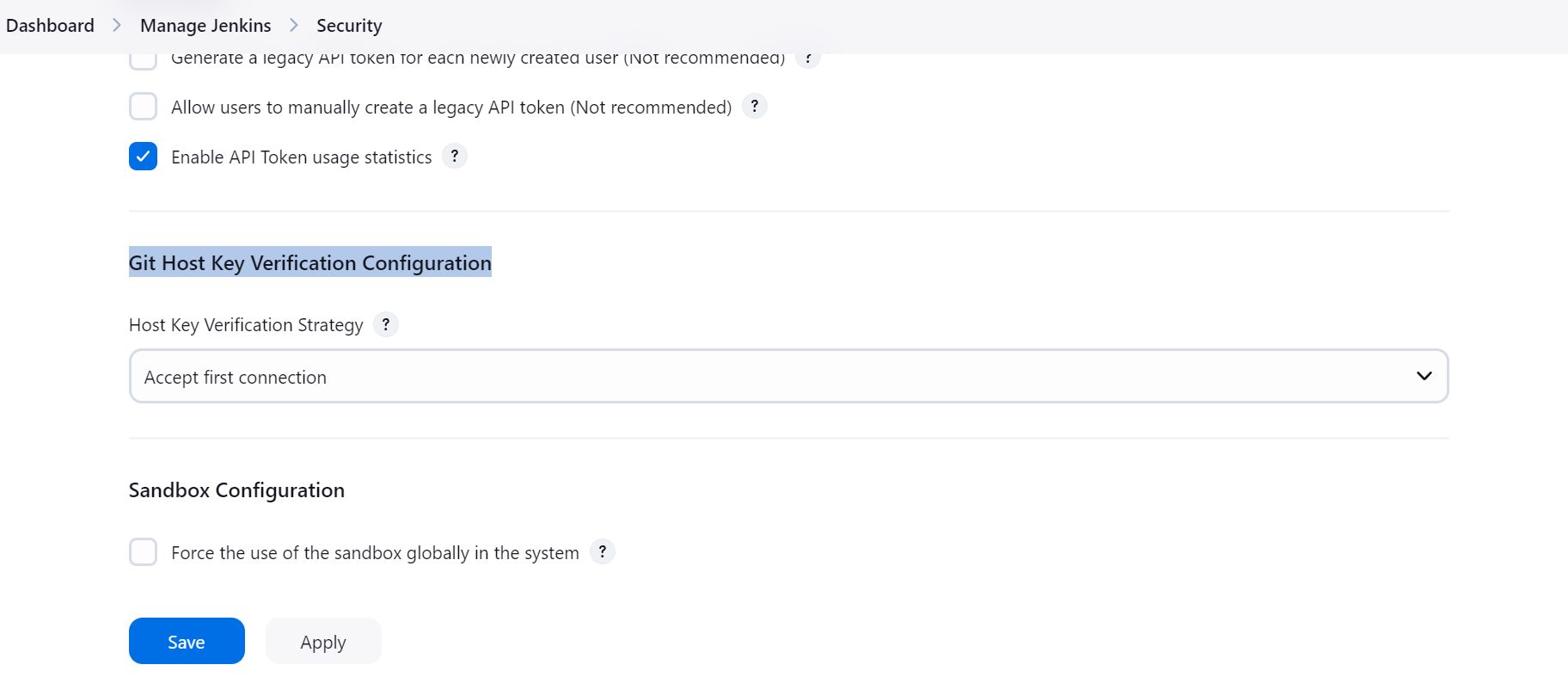
ubuntu@ip-172-31-26-103:~/.ssh$  
  
Now create a private repo, then go to github settings >> ssh and gpg keys >> add new ssh key >>  
  


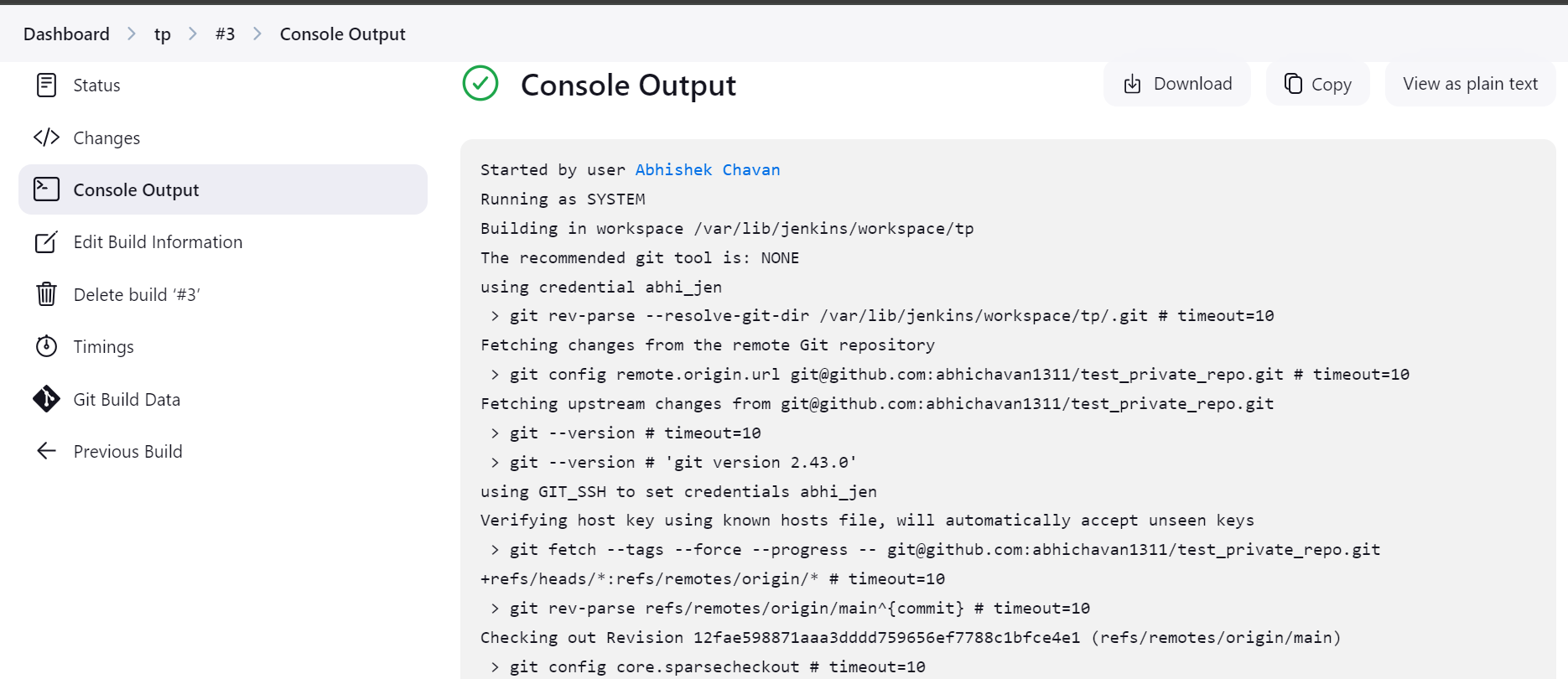
go to Jenkins settings, manage Jenkins >> credentials >> global credentials >> select kind as SSH Username with private key-  
  




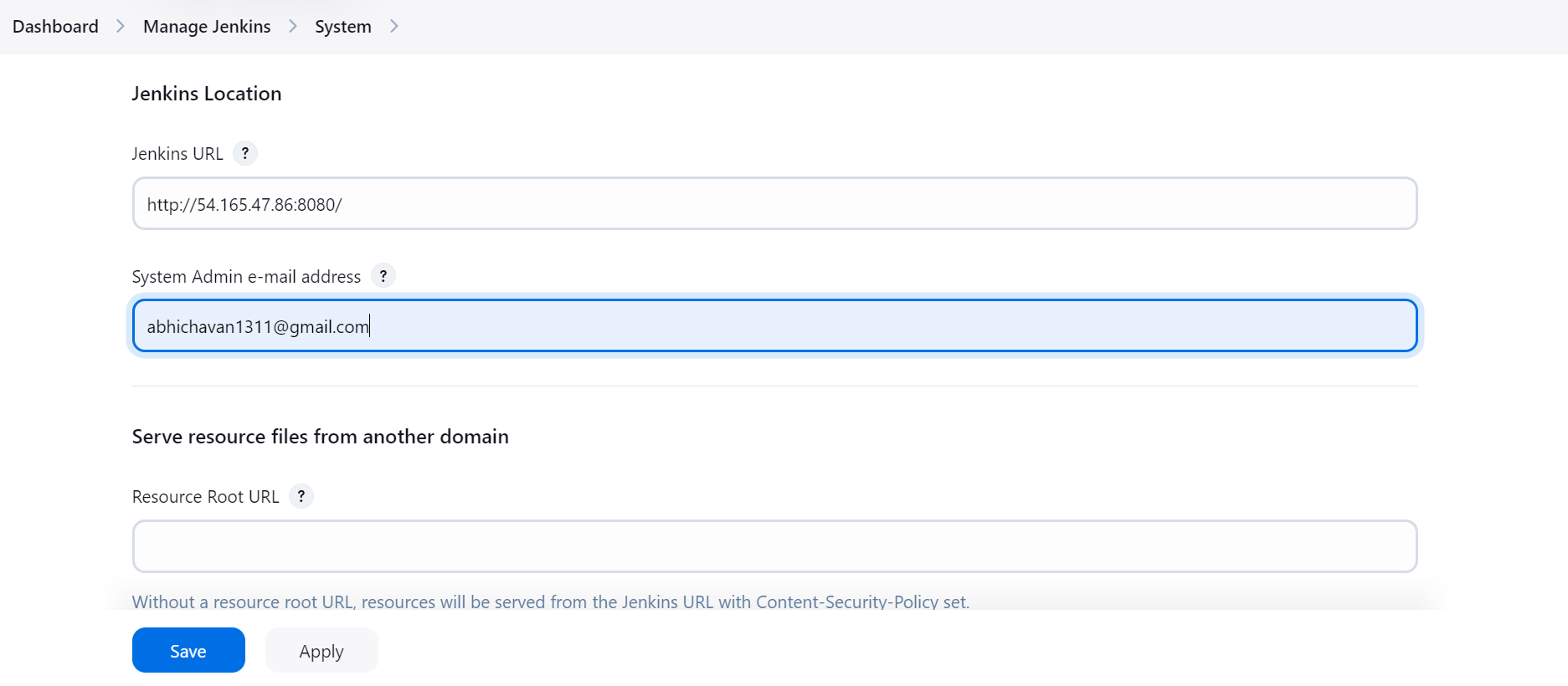
Now, create a new job add the ssh github url in repo url and add credentials  
  


If you encounter with any error then select appropriate options in Git Host Key Verification Configuration, depends on the condition



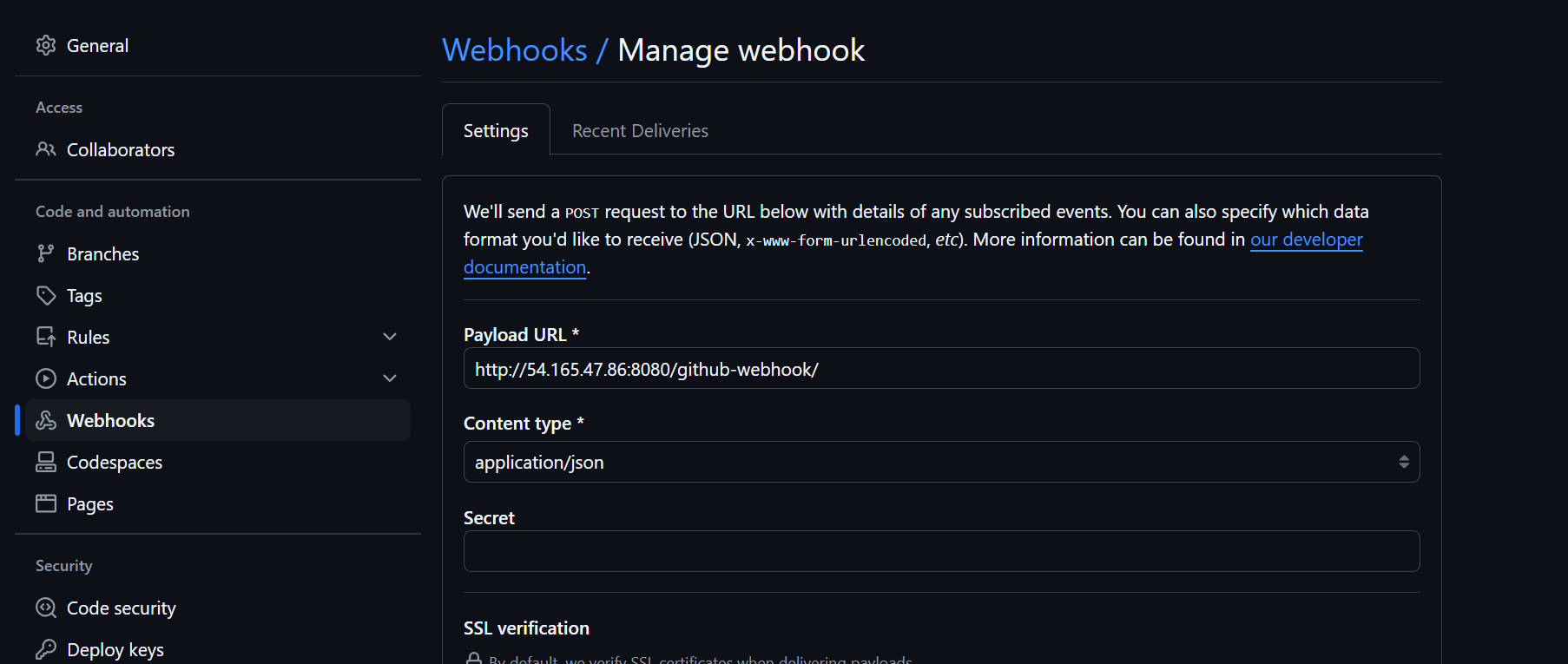
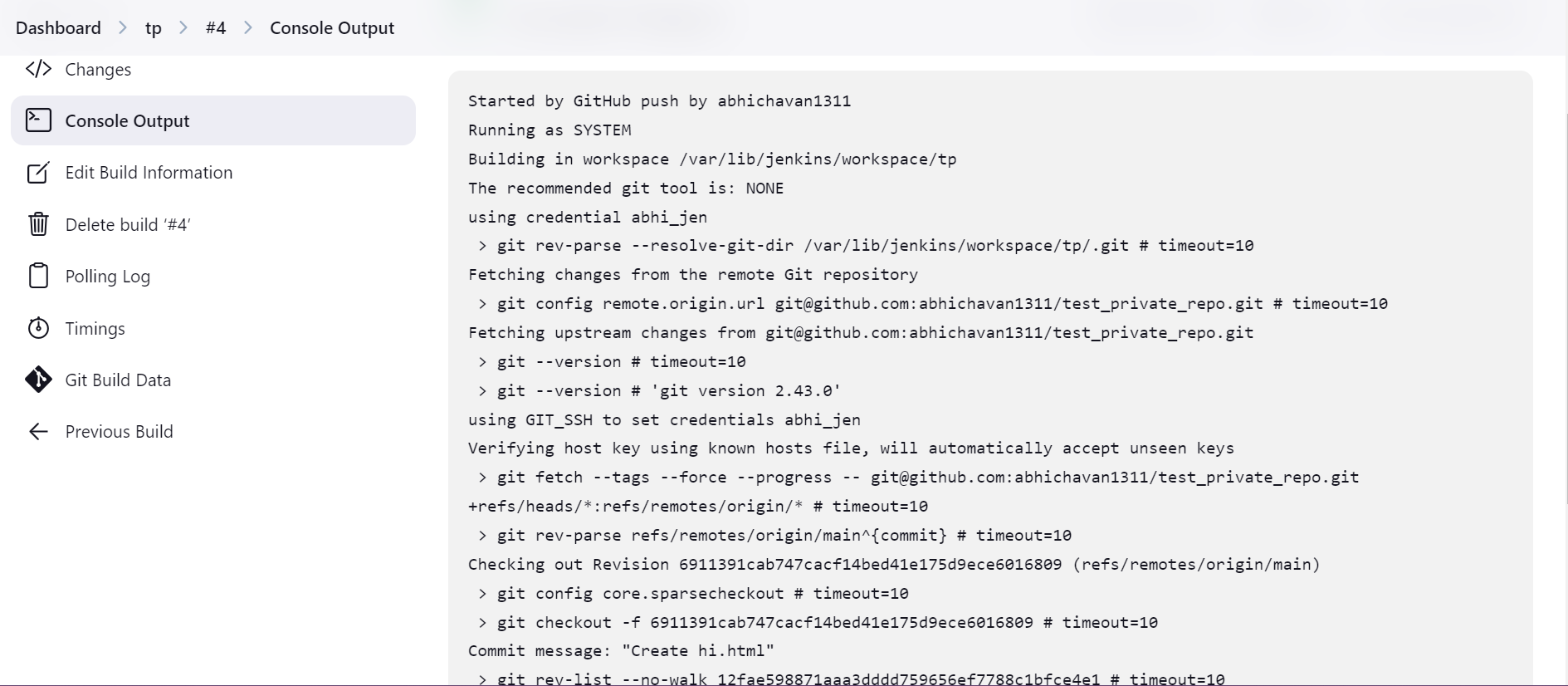
Create job using the same private repo and build it  
  




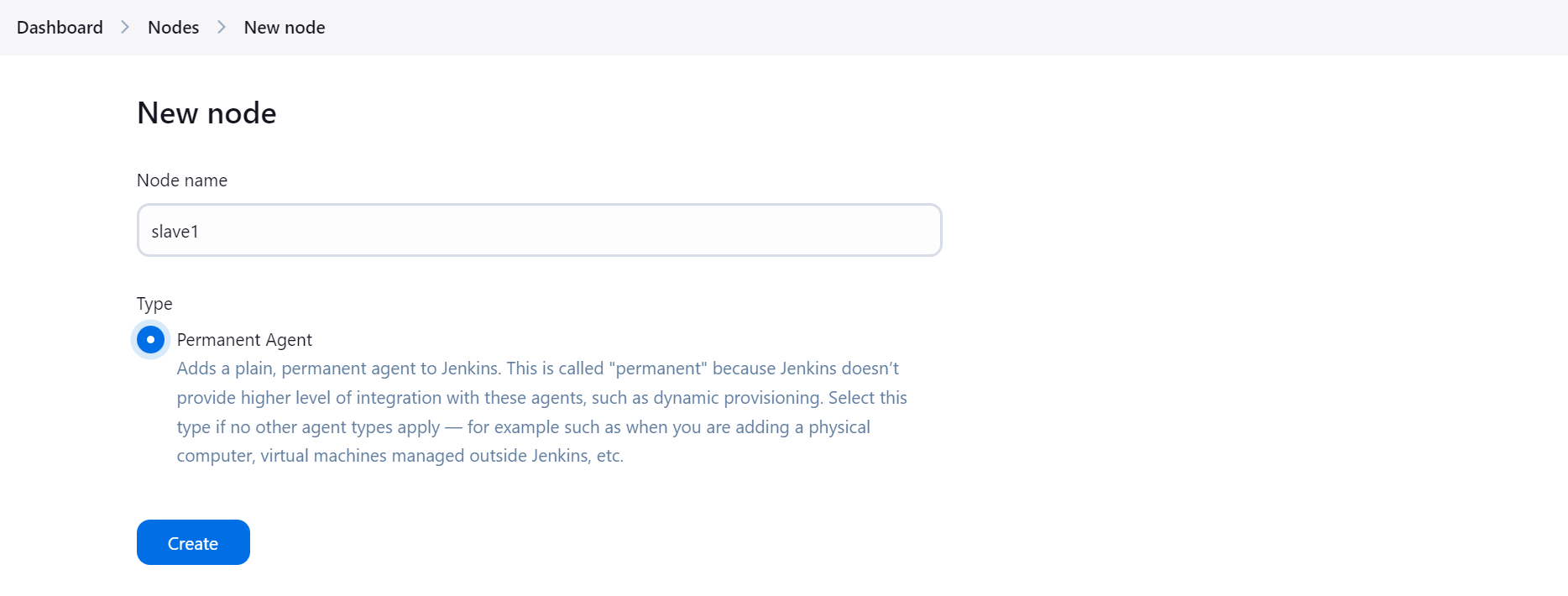
Change system IP if the instance is start and stopped due to some reason  
  


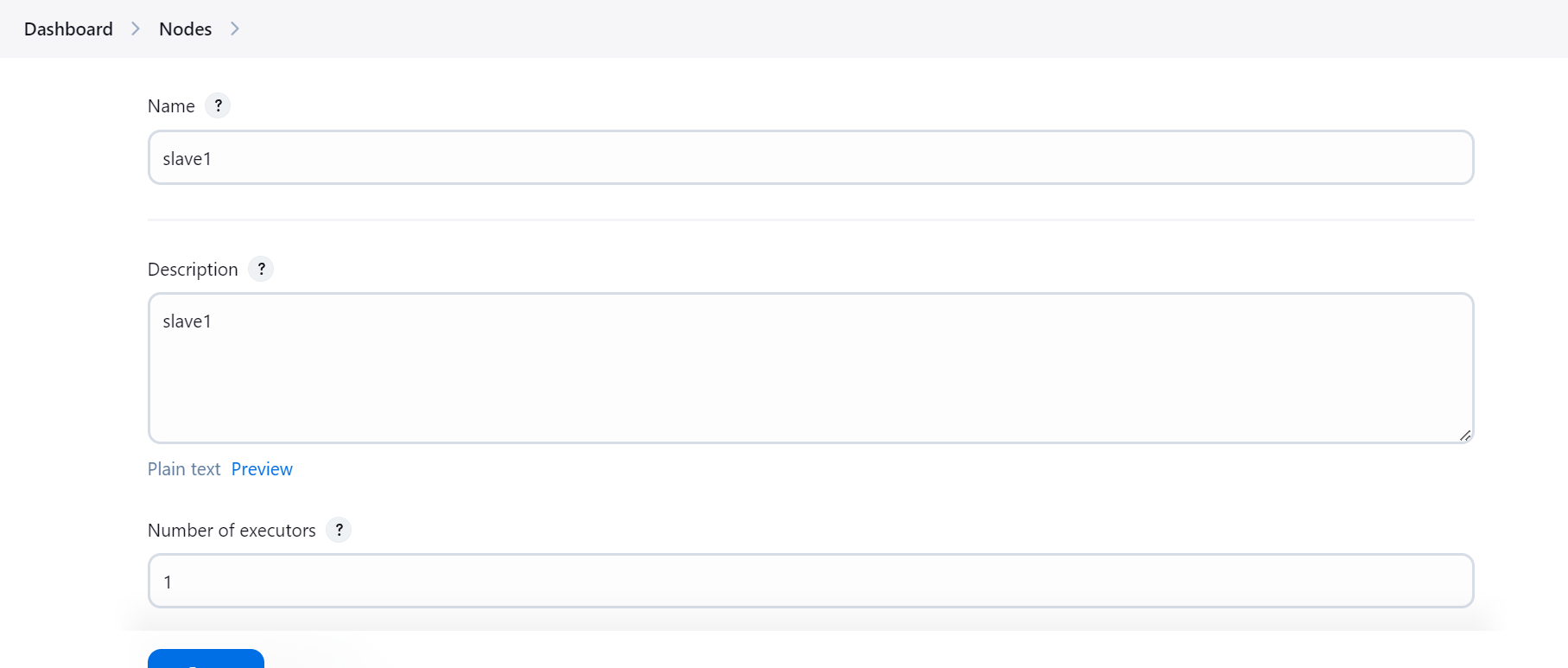
Trigger job using webhook, poll scm or build periodically option (SS added for webhook only)

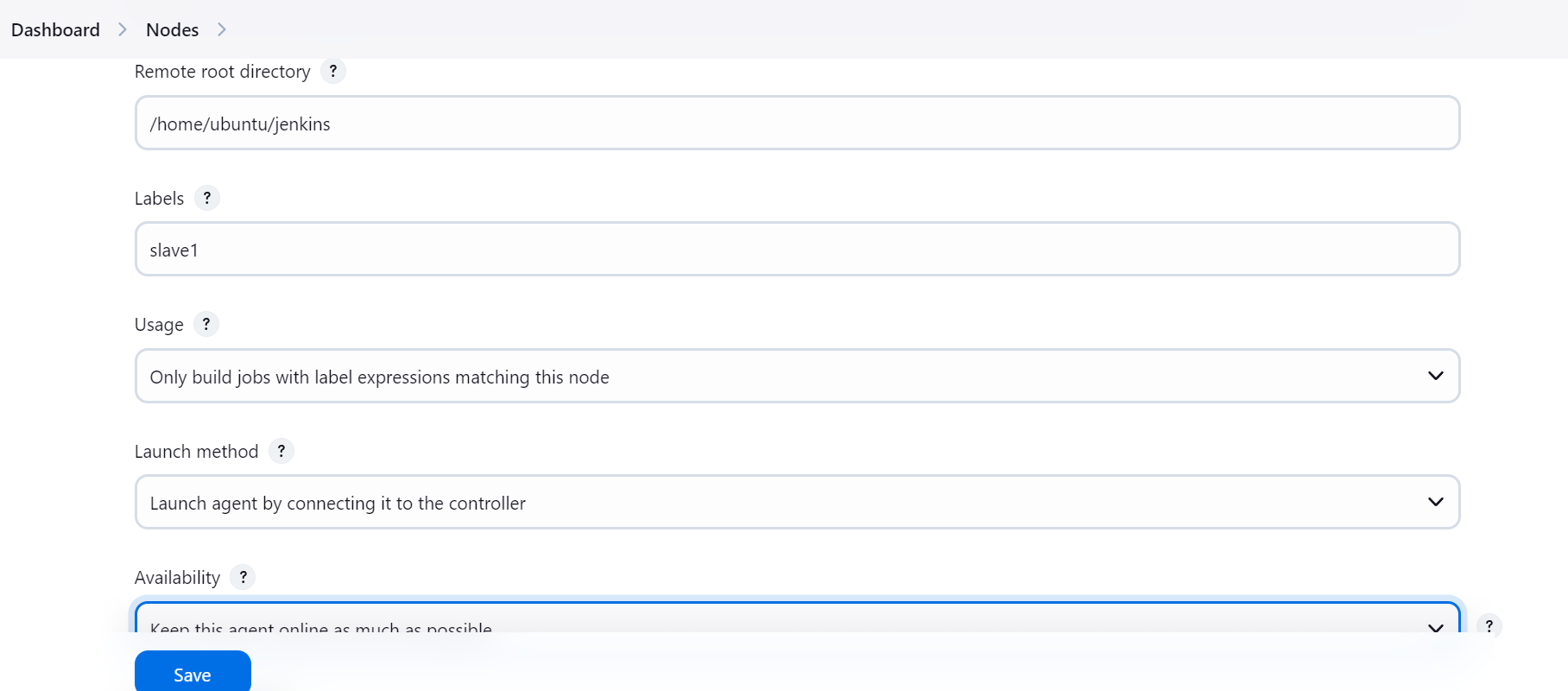
Go to repo >> settings >> webhook >> add webhookurl >> select content type and the **events which would you like to trigger this webhook -**

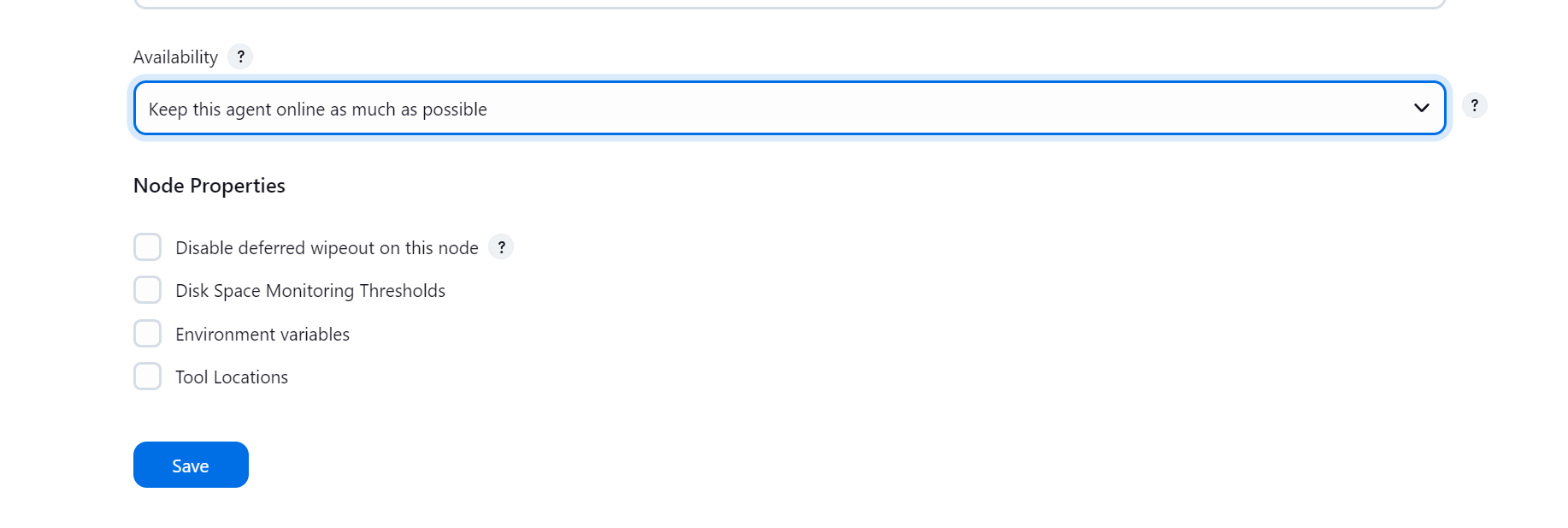
  
  
  
  
now create job using GitHub hook trigger for GITScm polling and just save it. Once you make any changes in the repo files and push it, job will start automatically building-  
  




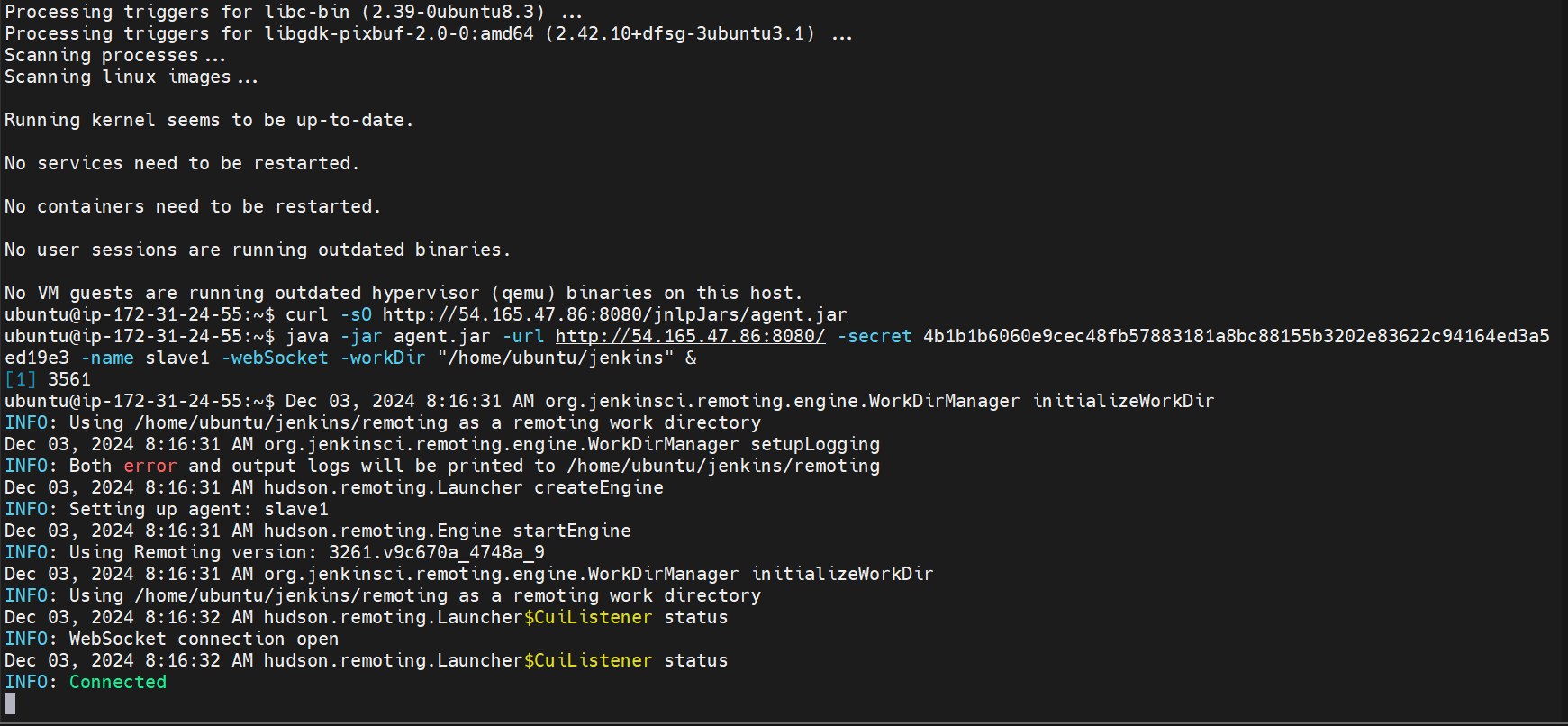
Components of distributed Jenkins (master-slave terminology)  
Master-Slave configuration   
Create 1 more server other then master and install java init as per your requirement,in our case we have named it slave 1.  
go to Jenkins dashboard >> nodes >> follow below steps  
  


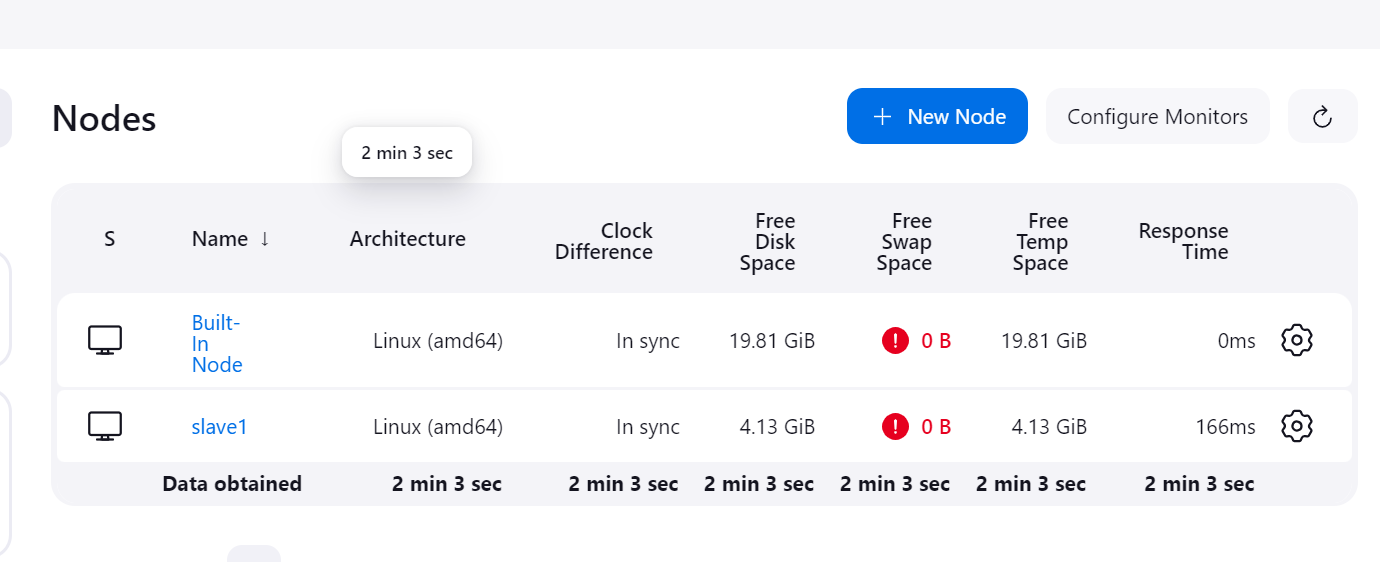
Note: it is recommended to keep your no of executors = vCPU



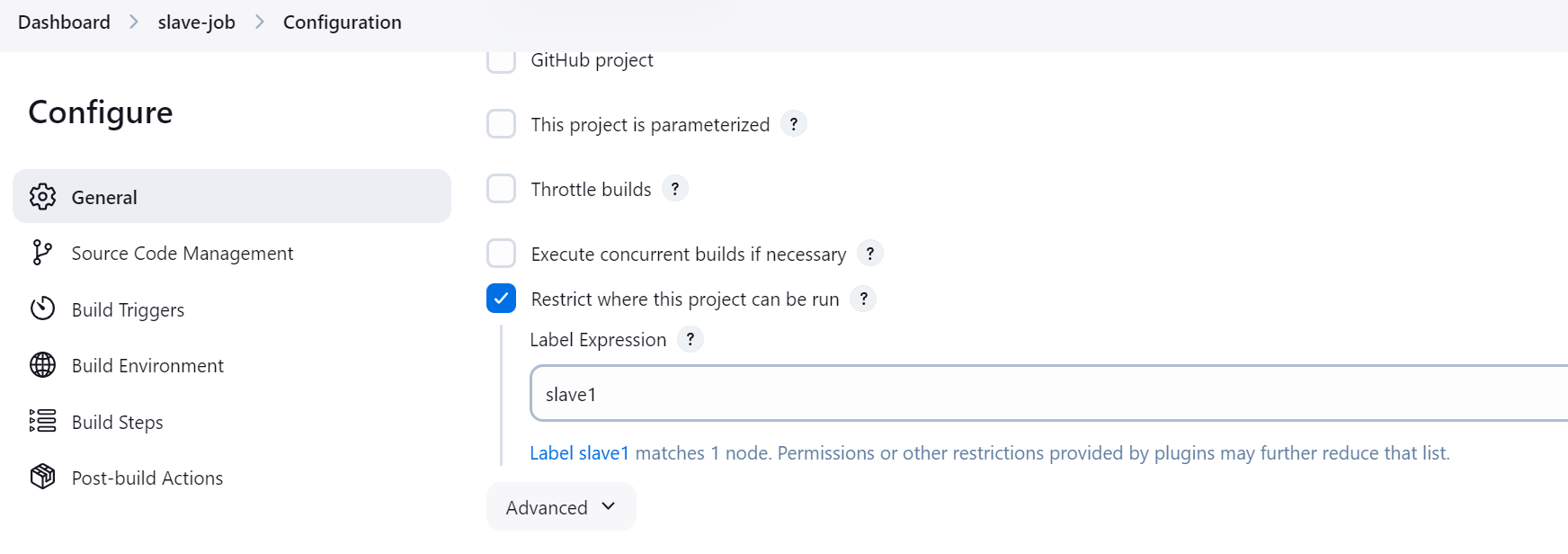


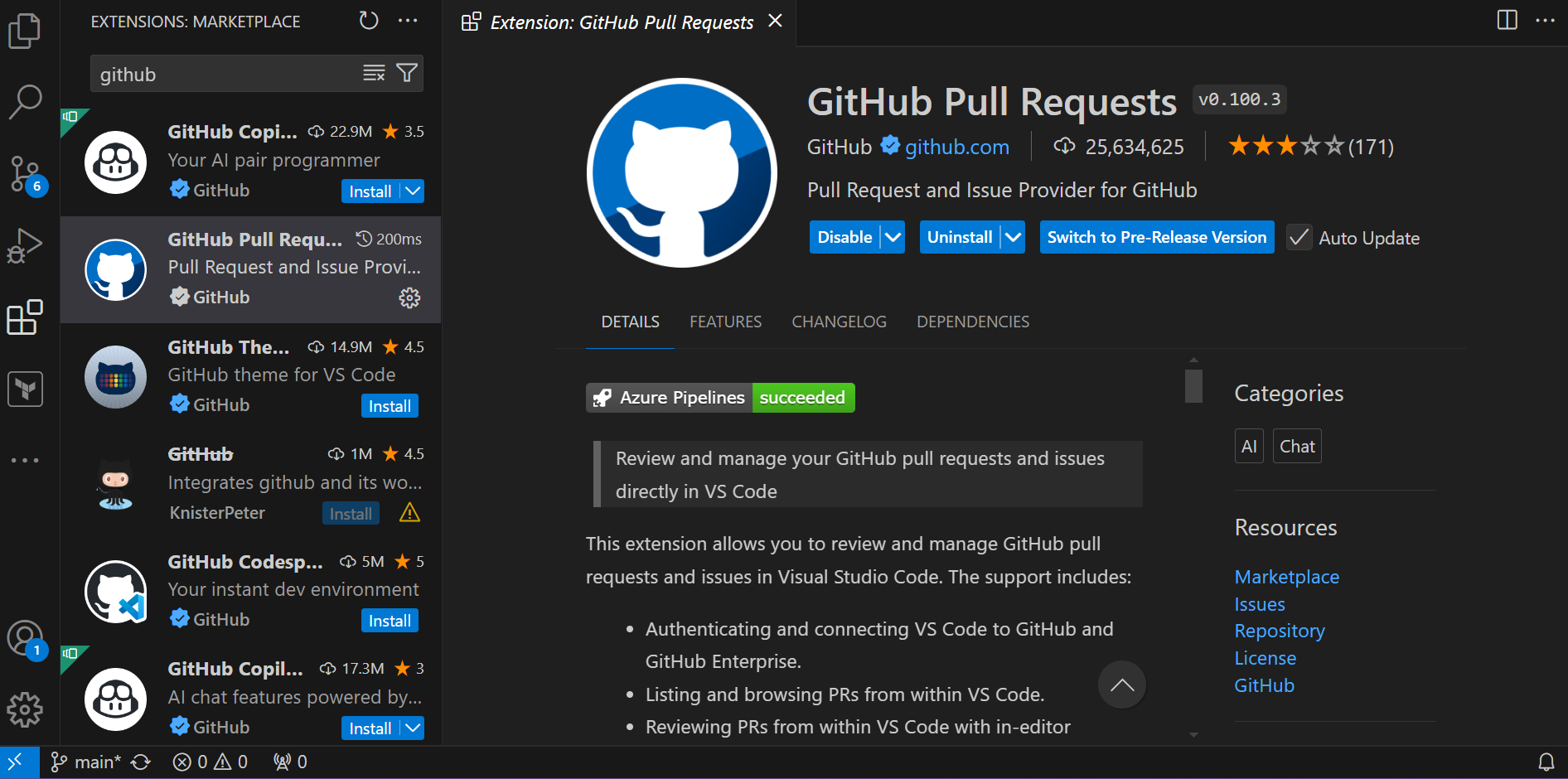
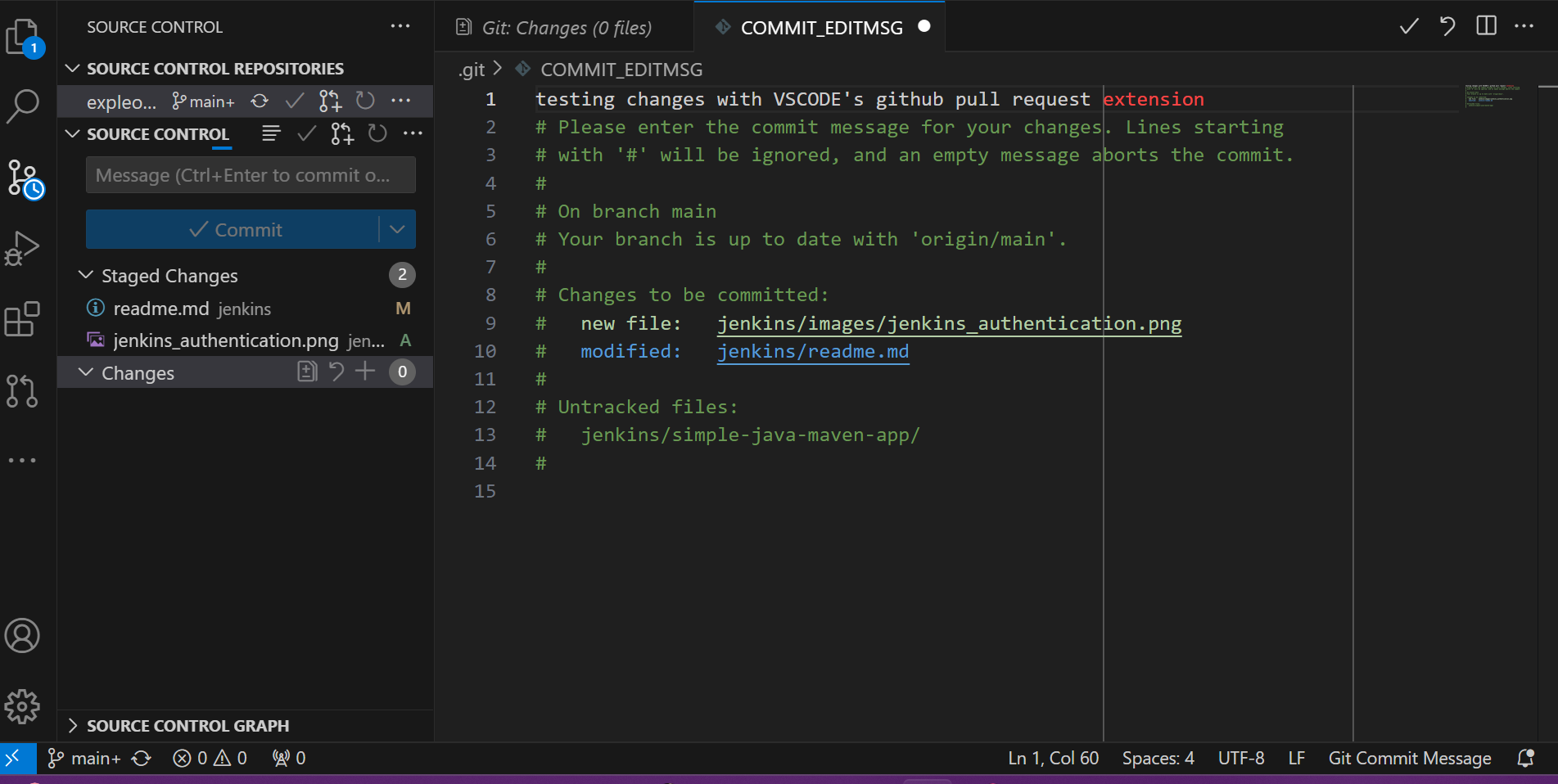
Follow the instructions given in slave1 node once you create it and you will be able to stablish connection between master and slave as shown below,

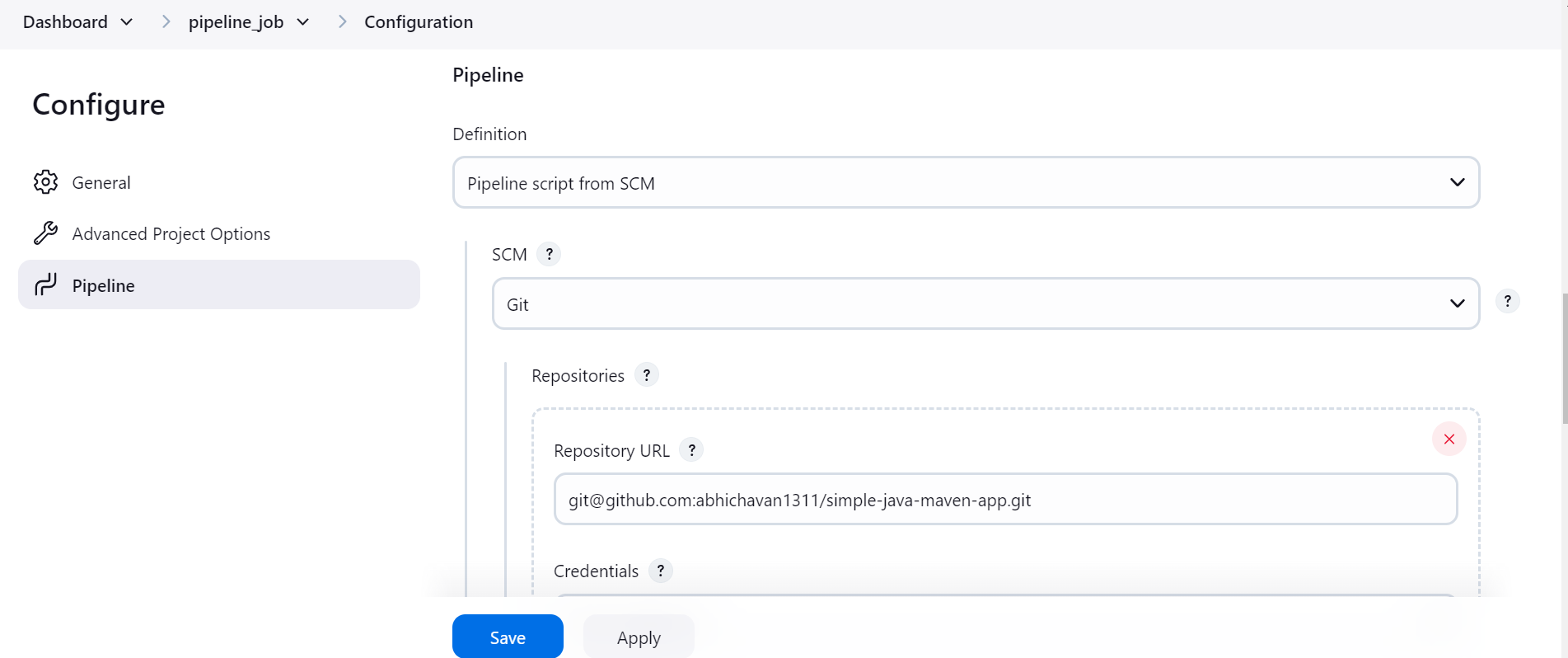


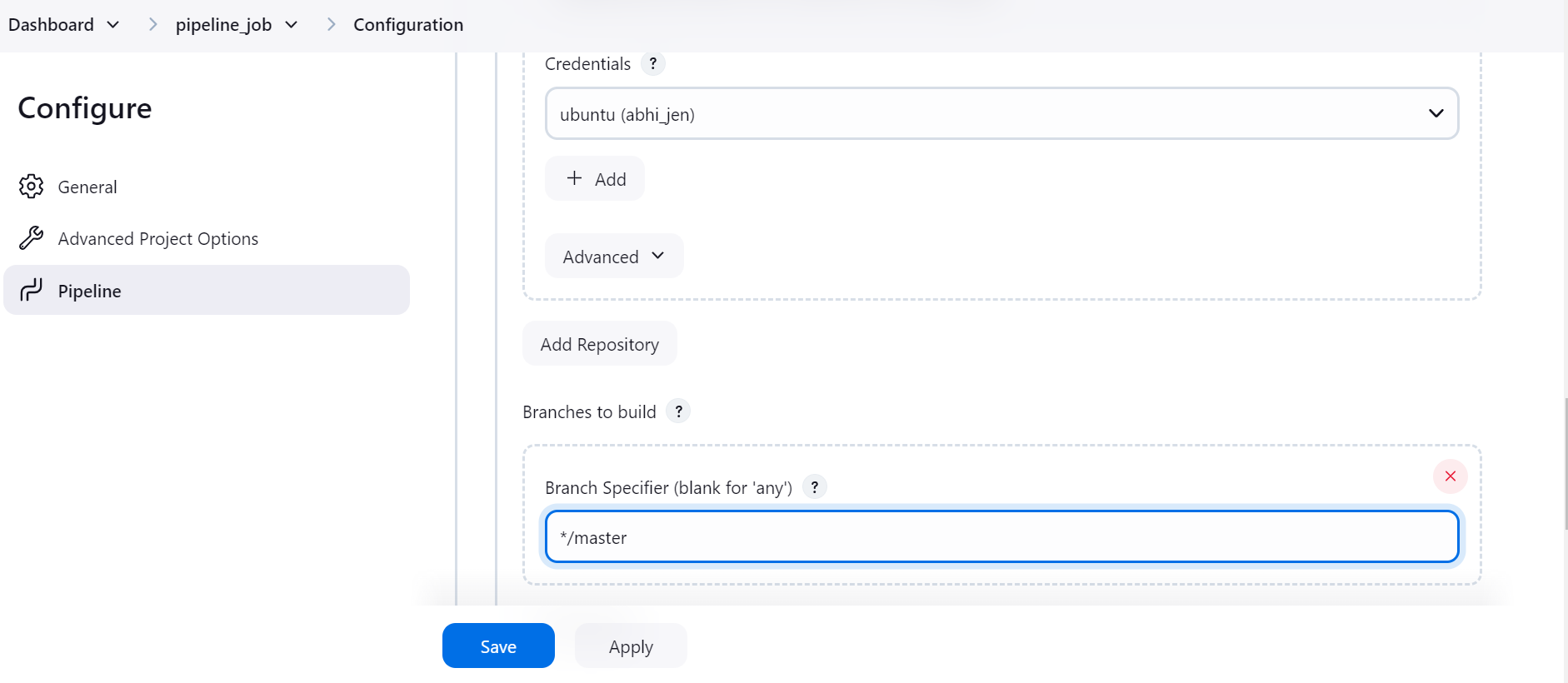


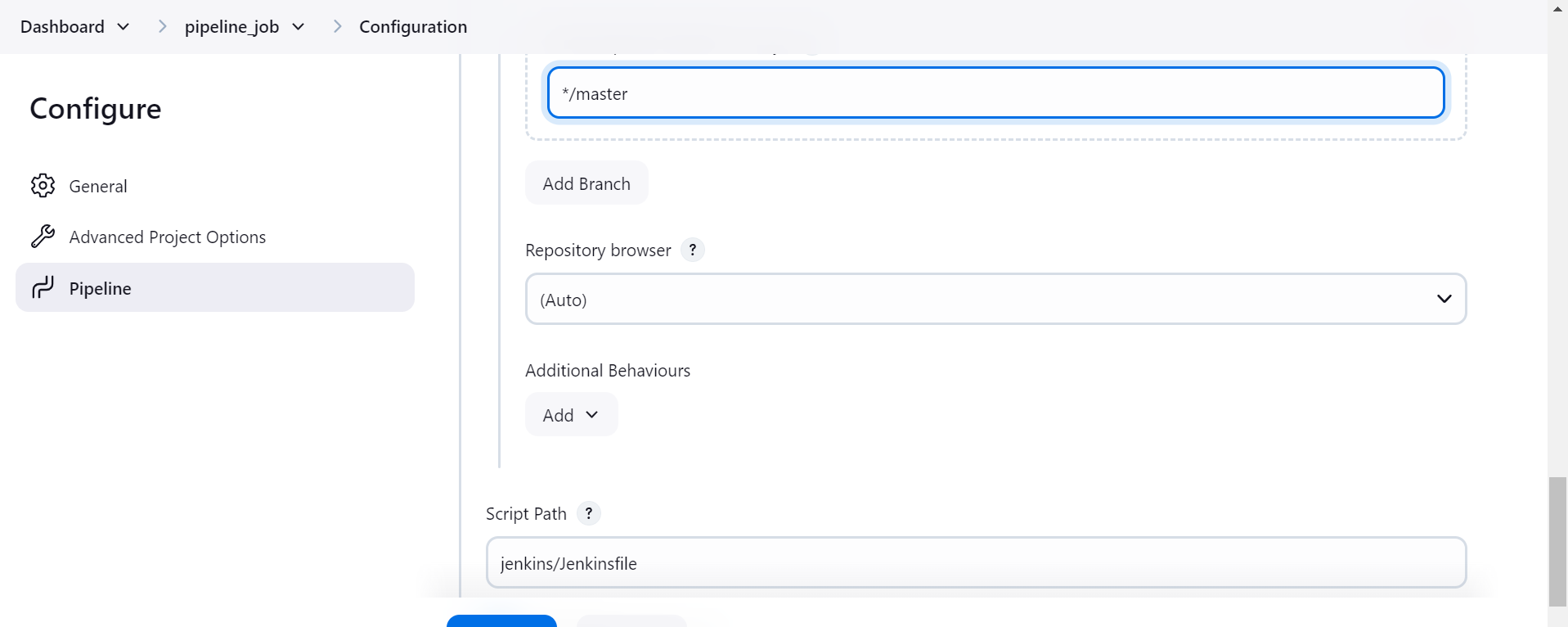
Now create anew job using “Restrict where this project can be run”,


using github with VScode extensions >> install github pull request plugin  
  
  
  


Create pipeline script from SCM  






Jenkinsfile code  
  
pipeline {

agent any

stages {

stage('Build') {

steps {

echo "Build stage"

}

}

stage('Test') {

steps {

echo "Test stage"

}

}

stage('Deliver') {

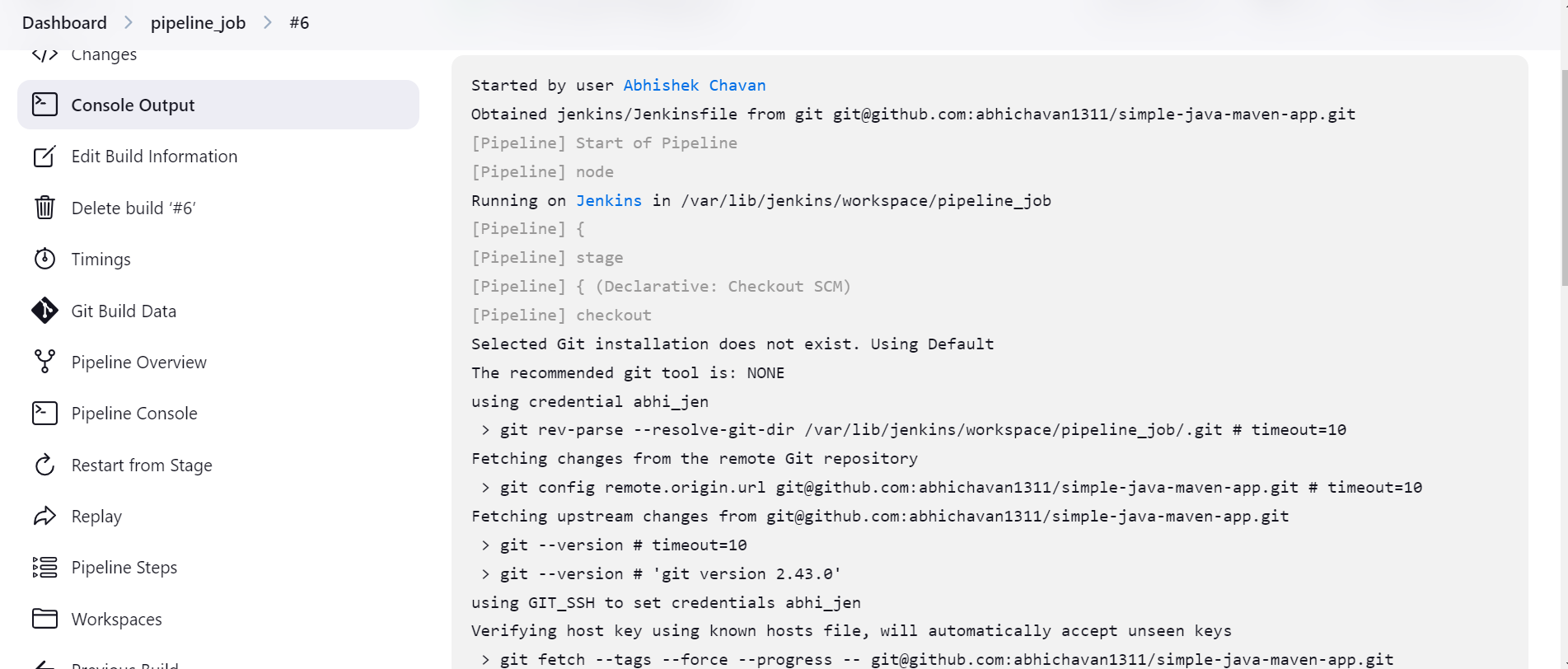
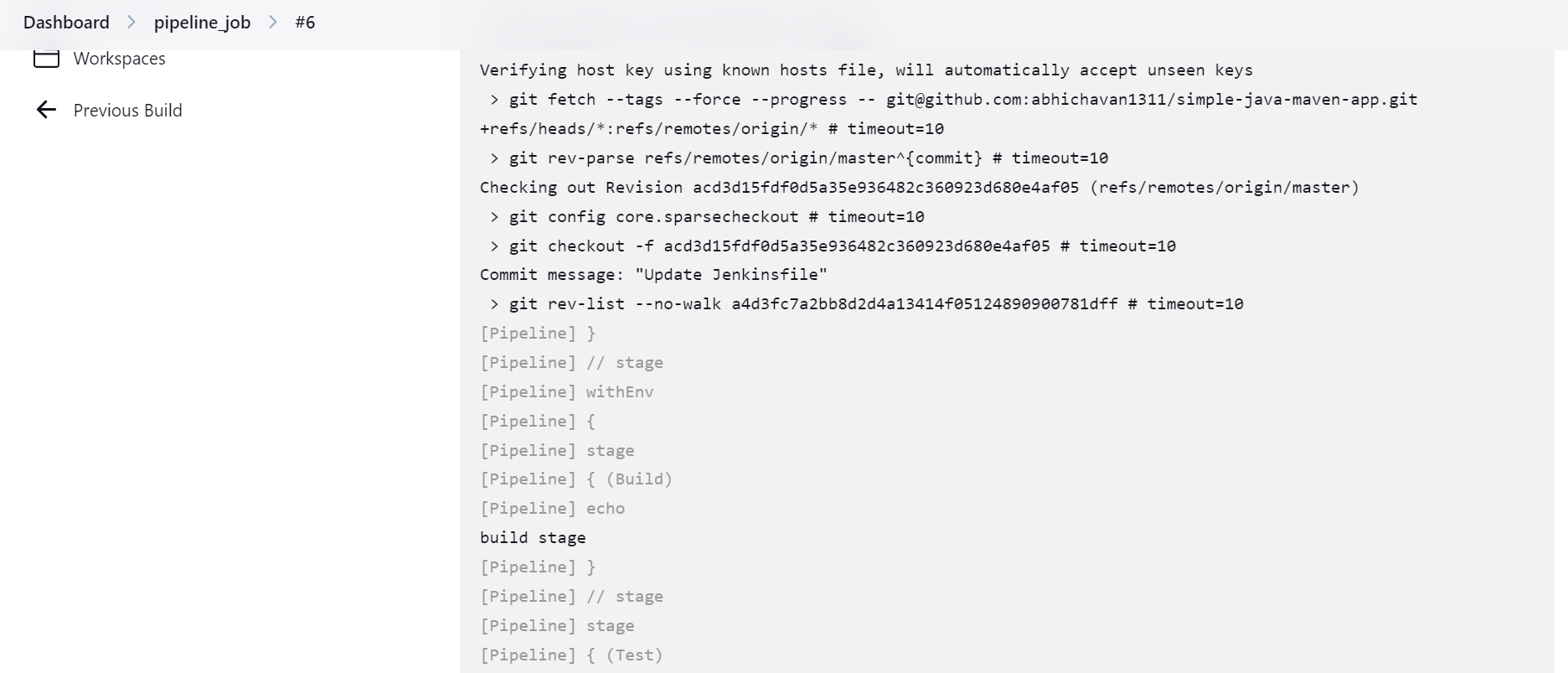
steps {

echo "Deployment stage"

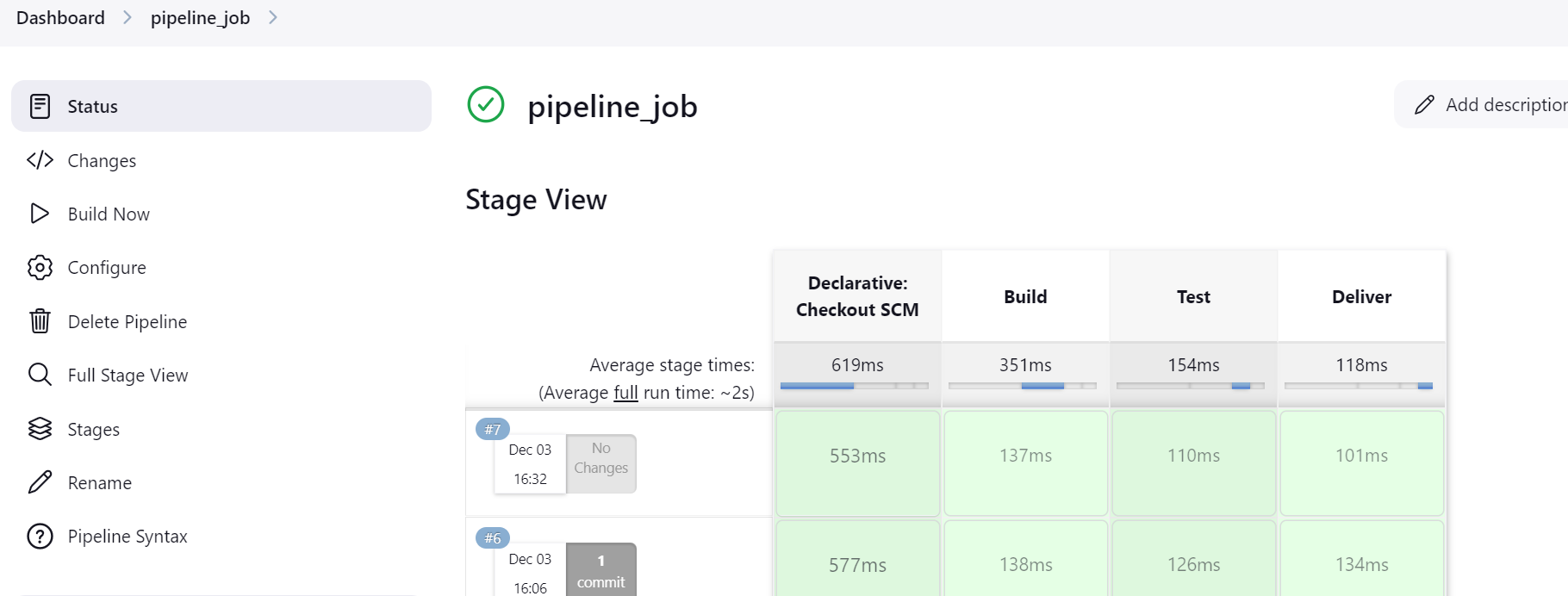
}

}

}}





Off-loading the work to agents or slaves  
  
pipeline {

agent {

label 'slave1'

}

stages {

stage('Build') {

steps {

echo "Build stage"

}

}

stage('Test') {

steps {

echo "Test stage"

}

}

stage('Deliver') {

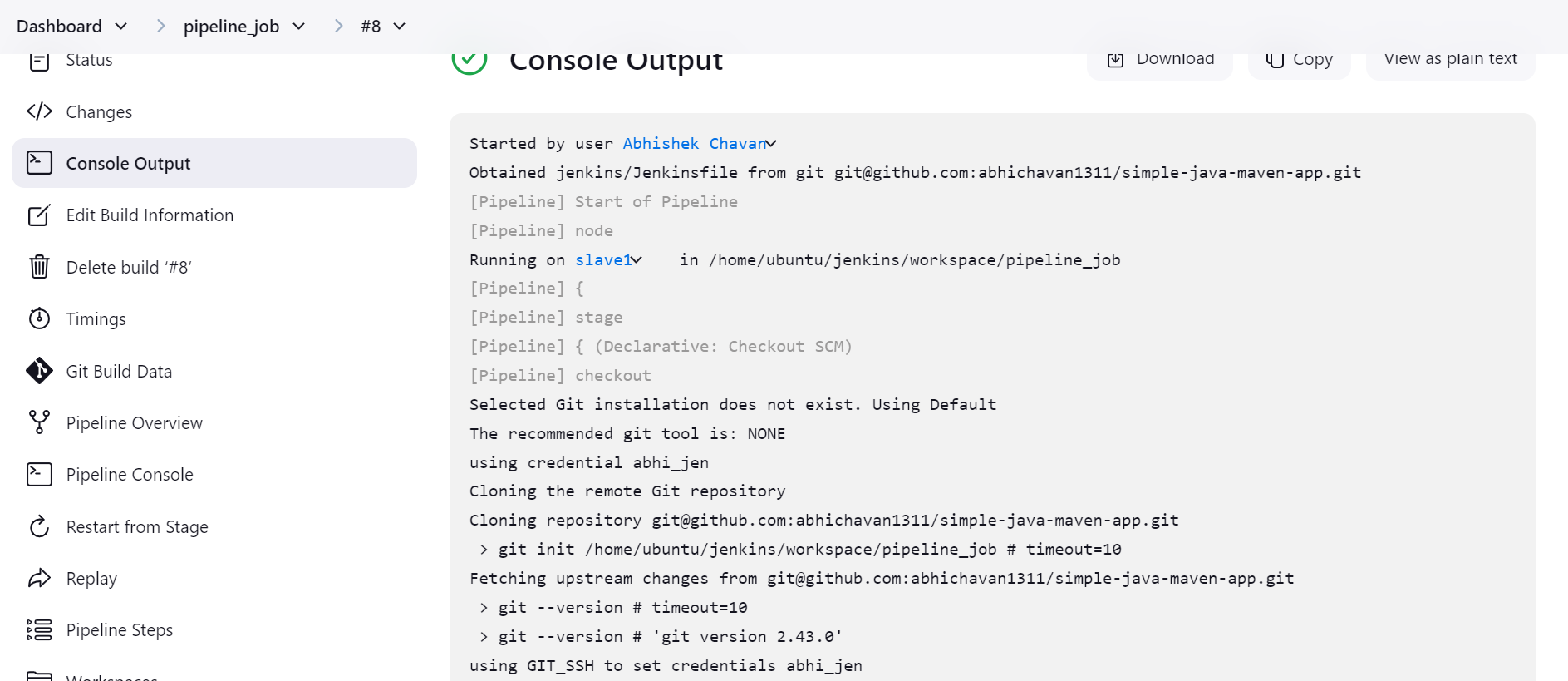
steps {

echo "Deployment stage"

}

}

}

}  
  


Creating pipeline for maven based application

pipeline {

    agent {

        label 'slave1'

    }

    tools {

         maven 'maven1'

    }

    stages {

        stage('Build') {

            steps {

                sh 'mvn clean package -DskipTests'

                sh 'ls -l target/'

            }

        }

        stage('Archive') {

            steps {

                echo "Archiving artifacts"

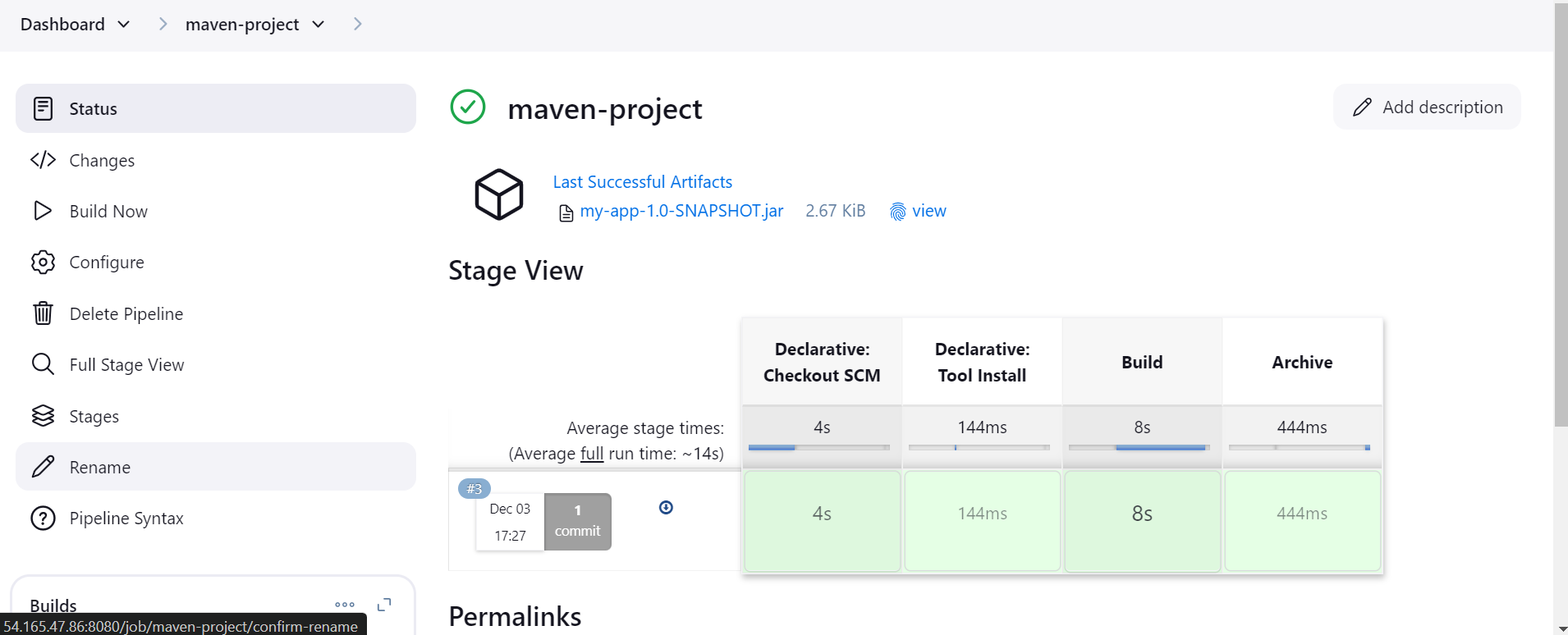
                archiveArtifacts artifacts: '\*\*/target/\*.jar', fingerprint: true

            }

        }

    }

}



Creating pipeline for maven based application using env variables, parameters  
  
pipeline {

    agent {

        label 'slave1'

    }

    parameters {

         string defaultValue: 'chavan', name: 'LASTNAME'

    }

    environment {

         NAME = "Abhishek"

    }

    tools {

         maven 'maven1'

    }

    stages {

        stage('Build') {

            steps {

                sh 'mvn clean package -DskipTests'

                sh 'ls -l target/'

                echo "hi $Name ${params.LASTNAME}"

            }

        }

        stage('Archive') {

            steps {

                echo "Archiving artifacts"

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

            }

        }

    }

}



Creating pipeline for maven based application using choice parameters, stash and unstash post action  
  
pipeline

{

agent {

  label 'master'

}

parameters {

    choice choices: ['blue', 'green'], name: 'select\_environment'

}

environment{

    NAME = "abhishek"

}

tools {

  maven 'maven1'

}

stages{

    stage('build')

    {

        steps {

            sh 'mvn clean package -DskipTests=true'

        }

    }

    stage('test')

    {

        parallel {

            stage('testA')

            {

                agent { label 'slave1' }

                steps{

                    echo " This is test A"

                    sh "mvn test"

                }

            }

            stage('testB')

            {

                agent { label 'slave1' }

                steps{

                echo "this is test B"

                sh "mvn test"

                }

            }

        }

        post {

        success {

             dir("webapp/target/")

            {

            stash name: "maven-build", includes: "\*.war"

                 }

                 }

            }

    }

    stage('deploy\_green')

    {

        when { expression {params.select\_environment == 'green'}

        beforeAgent true}

        agent { label 'slave1' }

        steps

        {

            dir("/var/www/html")

            {

                unstash "maven-build"

            }

            sh """

            cd /var/www/html/

            jar -xvf webapp.war

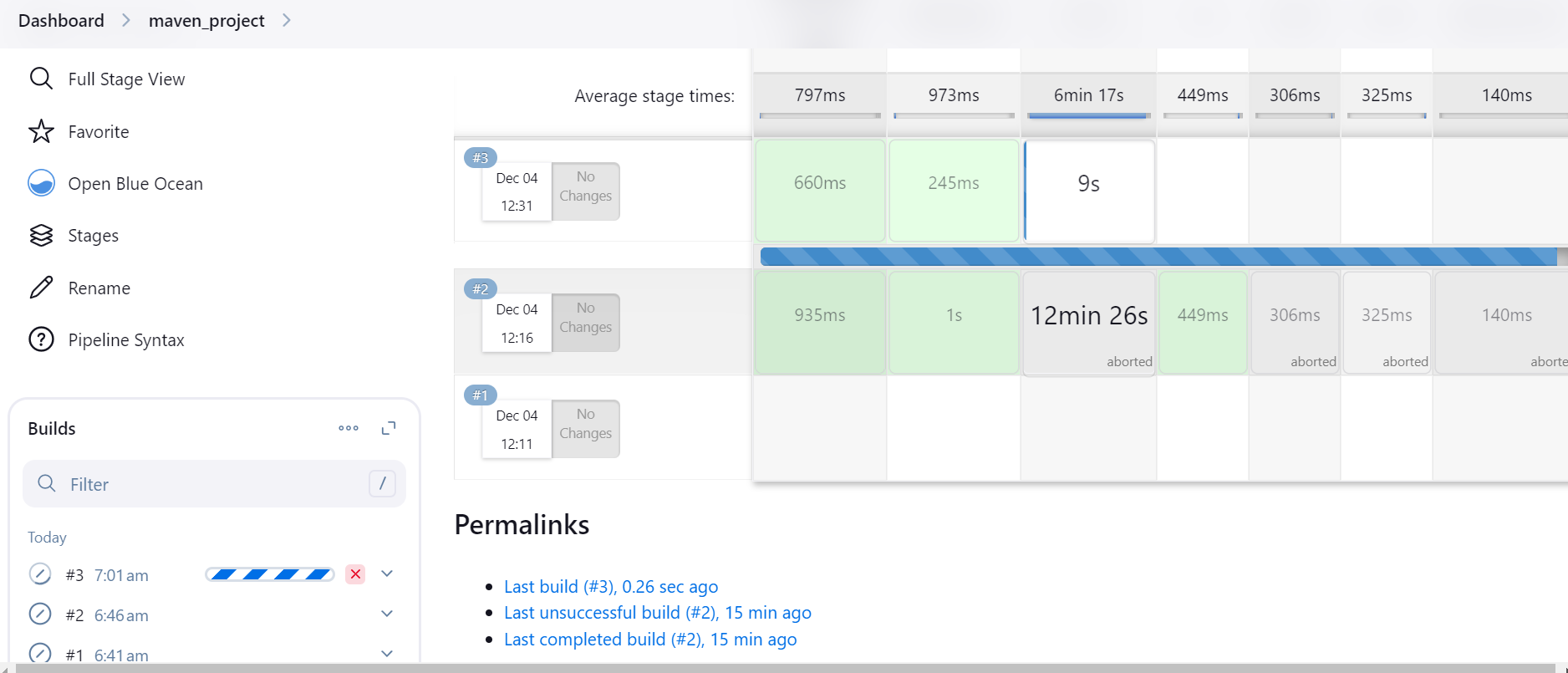
            """

        }

    }

}

}



Note: it seems that with free tier account above project is not feasible as Jenkins server got crashed 6-7 times, so we could not complete the build pipeline here.  
  
