Project Description:-

As soon as the developer pushes the updated code on the GitHub master branch, the Jenkins job should be triggered using a GitHub Webhook and, the code should be checked out, compiled, tested, packaged and containerized and deployed to the preconfigured test-server automatically. The deployment should then be tested using a test automation tool (Selenium), and if the build is successful, it should be deployed to the prod server. All this should happen automatically and should be triggered from a push to the GitHub master branch

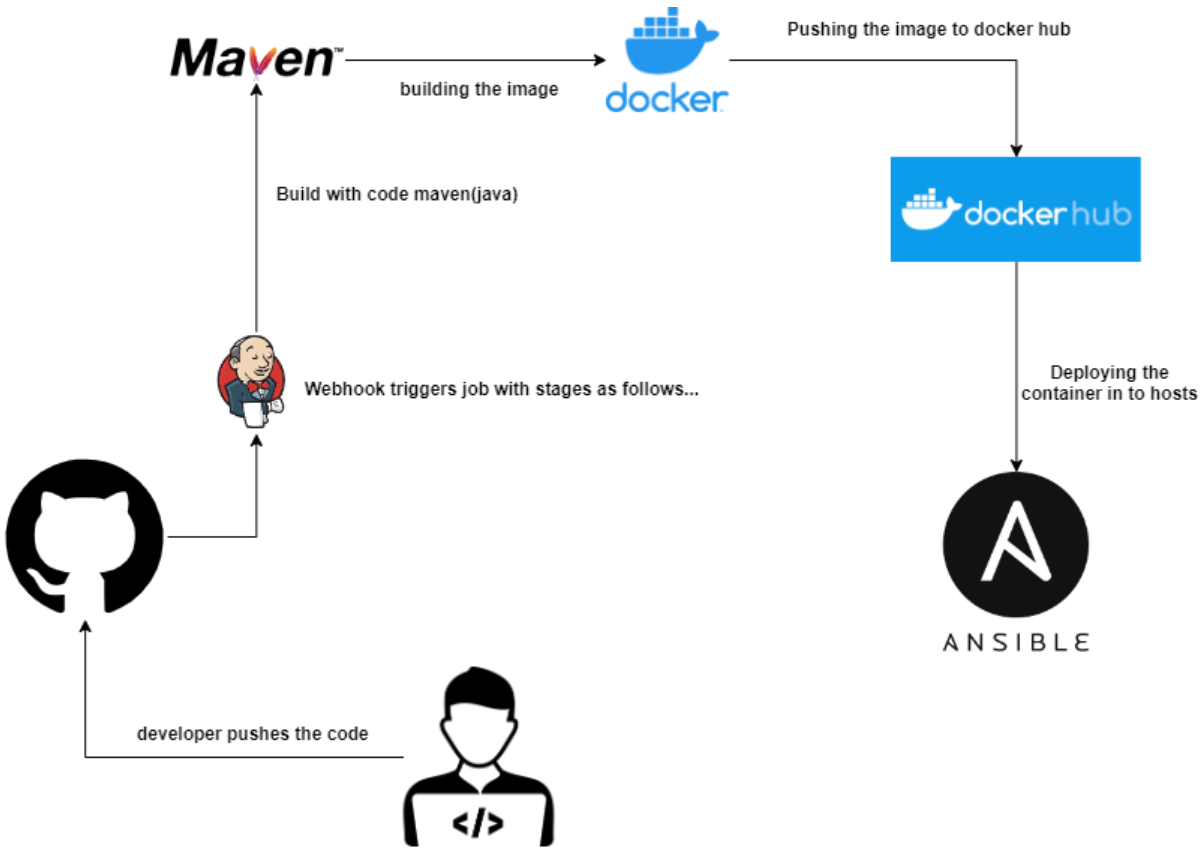
Need to implement Continuous Integration & Continuous Deployment using

- following tools:
- Git - For version control for tracking changes in the code files
- Jenkins - For continuous integration and continuous deployment
- Docker - For deploying containerized applications
- Ansible - Configuration management tools
- Selenium - For automating tests on the deployed web application
- AWS : For creating ec2 machines as servers and deploy the web application.

This project will be about how to test the services and deploy code to dev/stage/prod etc, just on a click of button

Solution:-

Approach-



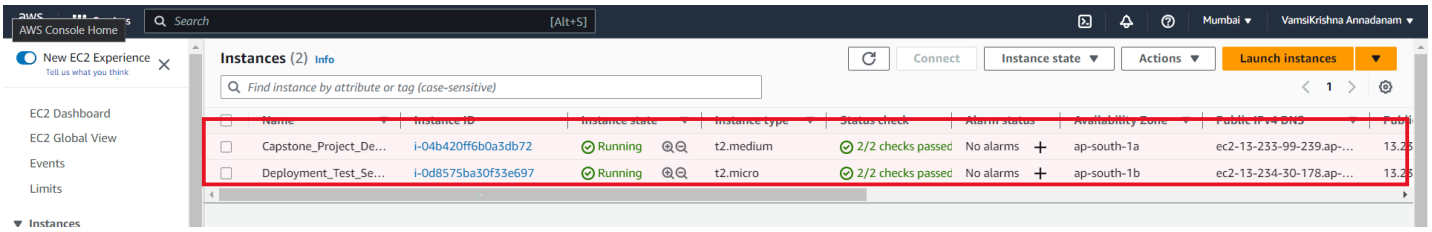
Source git hub- https://github.com/vamsikrishna918/Insurance-Web-Applicaion_E2E

Step 1:

1.A- EC2 instance setup

Creating 2 EC2 instanes name as below

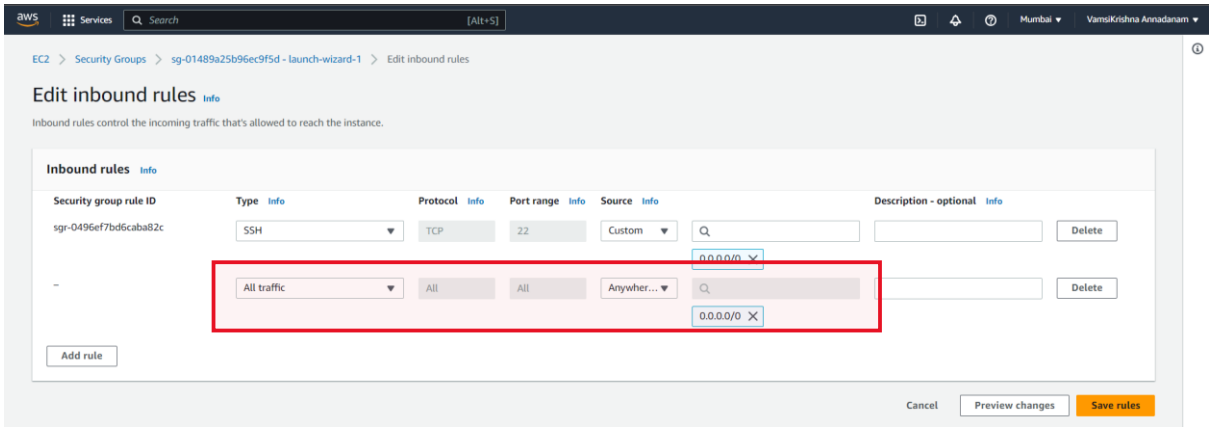
1. Capstone_Project_Development (T2.Medium with 30gib)
2. Deployment_Test_Server (T2.Micro with 8gib)



1.B

Setting up the security groups – allowing all Inbound traffic to both instances

Path: Security->Edit Inbound rules

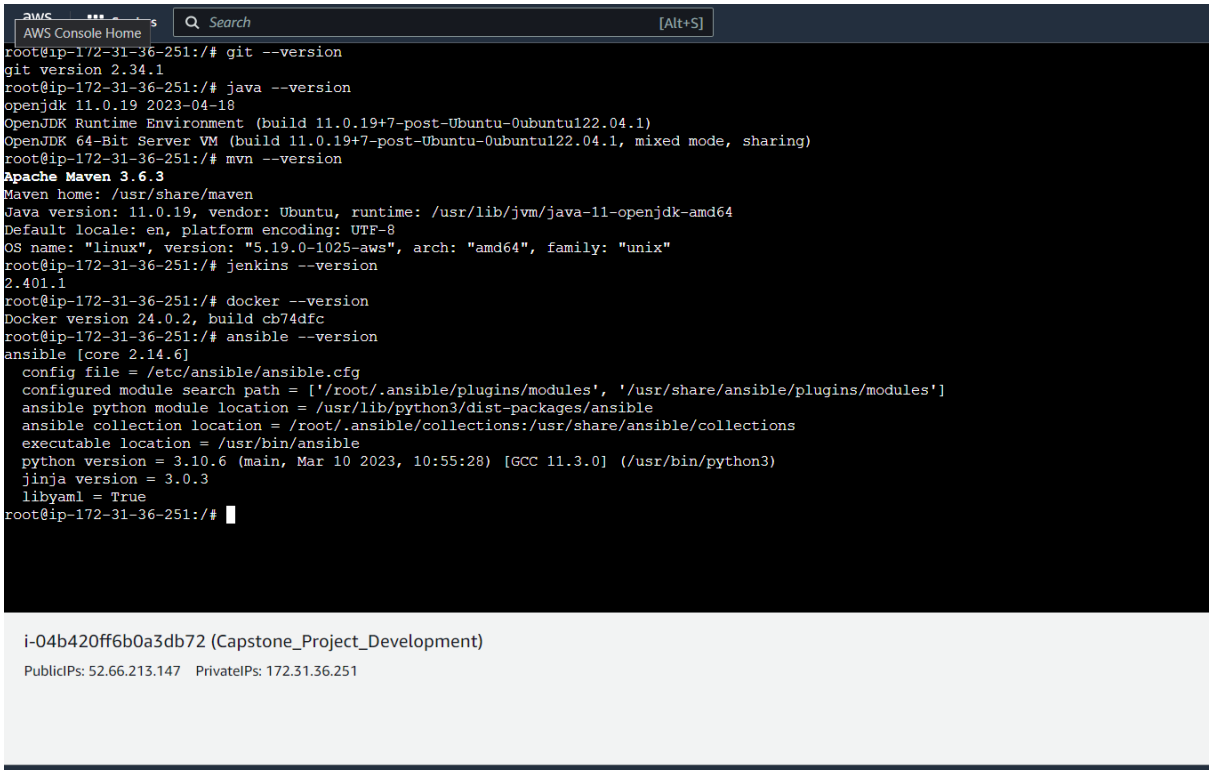


1.C Installing packages

Installing the necessary packages in instance (Capstone_Project_Development)

Follow the below sources.

1. Update packages – apt update
2. Git- https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
3. Java- https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-22-04
4. Maven- sudo apt install maven
5. Jenkins - https://www.jenkins.io/doc/book/installing/linux/#debianubuntu
6. Docker - https://docs.docker.com/engine/install/ubuntu/
7. Ansible- https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-debian



1.d - Password less authentication

Password less authetication b/w 2 servers (capstone_project_development (and) deployment_test_server)

On capstone_project_development instacne

Do command: ssh-keygen

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
/home/ubuntu/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dTOLZQqBjiD62h/CdZPwb1mvsEz7Mkc25pcZdtR6TrI ubuntu@ip-172-31-53-60
The key's randomart image is:
+---[RSA 3072]----+
|        ..       |
|. .   .  .       |
|.. ..o  . . *  . |
|.   .o.. o * +. .|
| .  . = S + .. . |
| ... . o o=.o + o|
| oo .   *= o.= * |
|. .. . +o+o.+ E .|
|   ..   +=o.     |
```

It will generate public and private keys as below

```
ubuntu@ip-172-31-53-60:~/ansible$ ls /home/ubuntu/.ssh/
authorized_keys  id_rsa  id_rsa.pub  known_hosts
ubuntu@ip-172-31-53-60:~/ansible$
```

Copy the id_rsa.pub content, paste it in the deployment_test_server authotized keys file

On deployment_test_server instacne

Do command: ssh-keygen

```
ubuntu@ip-172-31-62-28:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:AFIPl190A+Zj2rUDuoNXoW8BUVp4QwkkudTphdDLbmw ubuntu@ip-172-31-62-28
The key's randomart image is:
+---[RSA 3072]----+
|  ..+ +B+OBoo    |
|   . =+.***. .   |
|     .o=o=*..    |
|      ..=B = .   |
|       oS + o    |
|        .E+ . .  |
|       .o+ o     |
|        . o      |
|                 |
+----[SHA256]-----+
ubuntu@ip-172-31-62-28:~$
```

```
ubuntu@ip-172-31-62-28:~$ ls ~/.ssh/
authorized_keys  id_rsa  id_rsa.pub
ubuntu@ip-172-31-62-28:~$ vim ~/.ssh/authorized_keys
```

Copy the id_rsa.pub content in capstone_project_development.

paste it in the deployment_test_server **authotized_keys file**

1.e ansible setup

On capstone_project_development instacne

Do command: sudo su –

Cd /ect/ansible

```
root@ip-172-31-36-251:/etc# cd /etc
AWS Console Home
root@ip-172-31-36-251:/etc# cd ansible/
root@ip-172-31-36-251:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ip-172-31-36-251:/etc/ansible#
```

Command:- Vi hosts

Insert the deployment server(deloyment_test_server) ip

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers:

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:

[webservers]
172.3[redacted]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern, you can specify
# them like this:

## www[001:006].example.com

"hosts" 45L, 1026B
```

i-04b420ff6b0a3db72 (Capstone_Project_Development)

PublicIPs: 13.232.185.77 PrivateIPs: 172.31.36.251

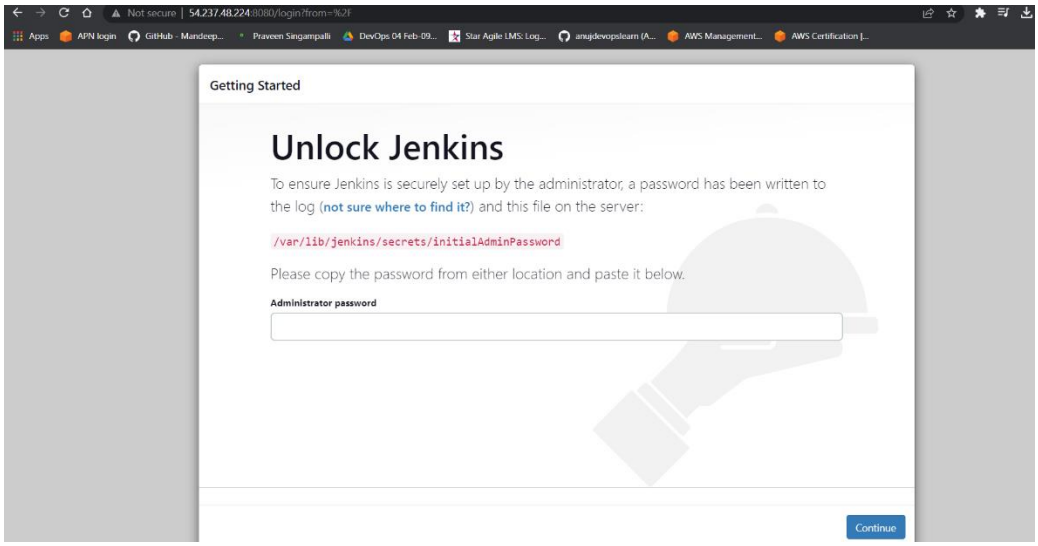
1.f Jenkins setup

Setting up the Jenkins (default running on 8080)

User- vams*****

Pas- pas*****

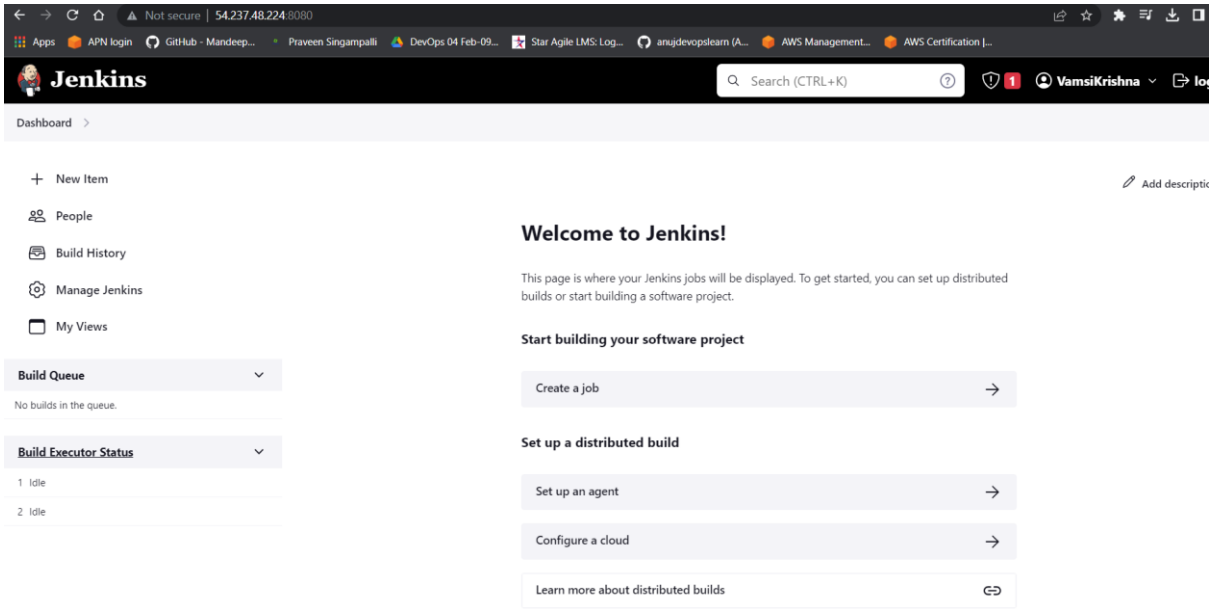
We can access Jenkins using http://IP :8080



Command :

Sudo cat /var/lib/Jenkins/secret/initialAdminPassword

After completing the initial setup



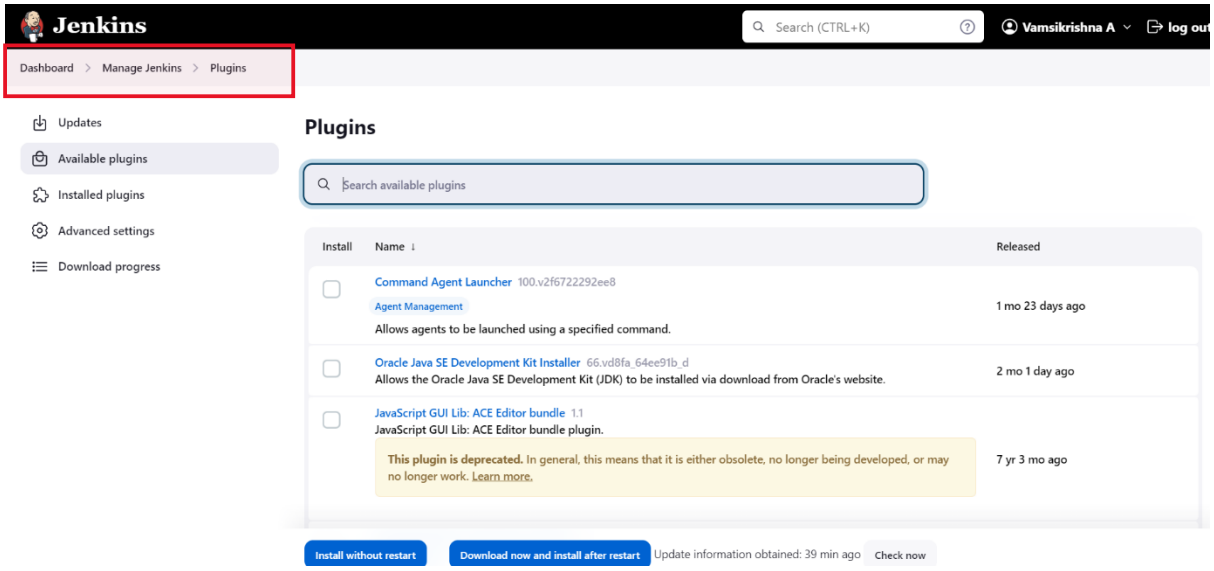
1.e

Plugin installation :

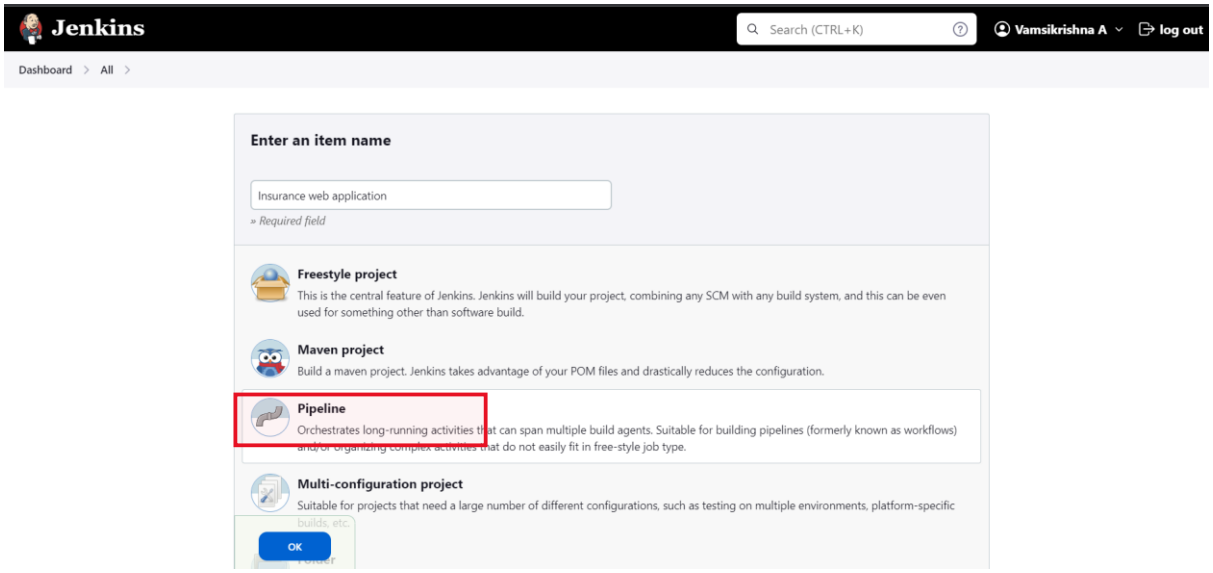
For integration of setup install the following plugins

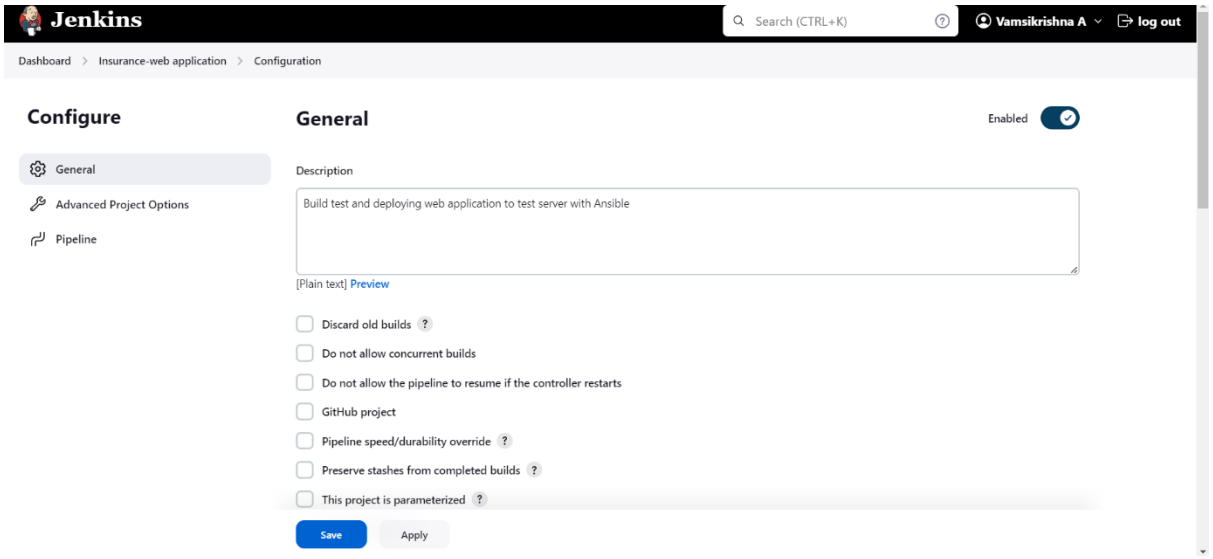
Path – dashboard->manage Jenkins-> plugins->available plugins

- Ansible
- Html publisher
- Docker
- Git
- Maven



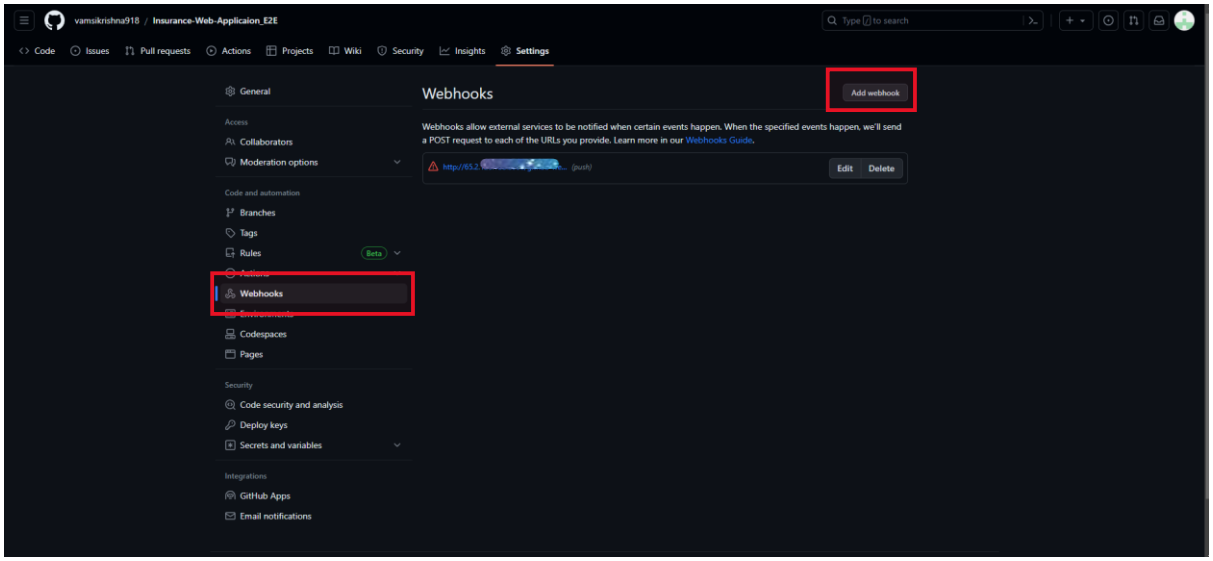
Step 2: Creating job



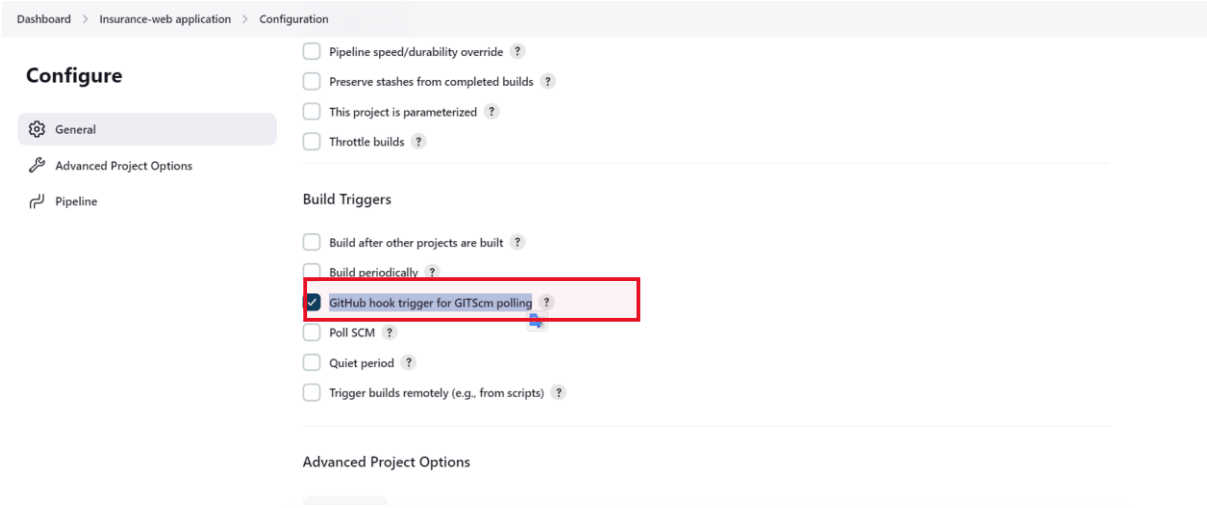


Setting up the webhook triggers- (The Jenkins will always watches the git repository if any commit happens it will automatically Runs the Jenkins job.)

GitHub -> repo-> settings-> webhooks-> add webhook -> add the URL of Jenkins



And in Jenkins check the **GitHub hook trigger for GITScm polling**



Create pipeline with the following stages:

1. Code checkout –
   checking out the code repo in **GitHub**
   use git:Git as sample step in syntax generator
2. Build –
   building the code, we have checkout with building tools like **maven**
3. publish the report-
   storing the generated reports in a directory
   use publishHTML:publishHTML reports as sample step in syntax generator
4. Image prune-
   docker image prune command allows you to clean up unused images
5. Build Docker image-
   building the docker image with Dockerfile
   #**Refer** Docker file in repository
6. Push Docker image to Hub-
   pushing the created image to docker registry (Docker Hub)
   use withCredentials:Bind credentials to variables as sample step in syntax generator

7. deploying to Test server –
   containerizing the image and deploying it into all hosts servers using Ansible.
   Use ansiblePlaybook:Invoke an ansible Playbook as sample step in syntax generator
   #Refer ansible-playbook file in repository

#**refer** the Jenkins file from git repo

Note- we have used Declarative pipeline syntax,

use **syntax generator/snippet generator** to generate the syntax as mentioned in steps.

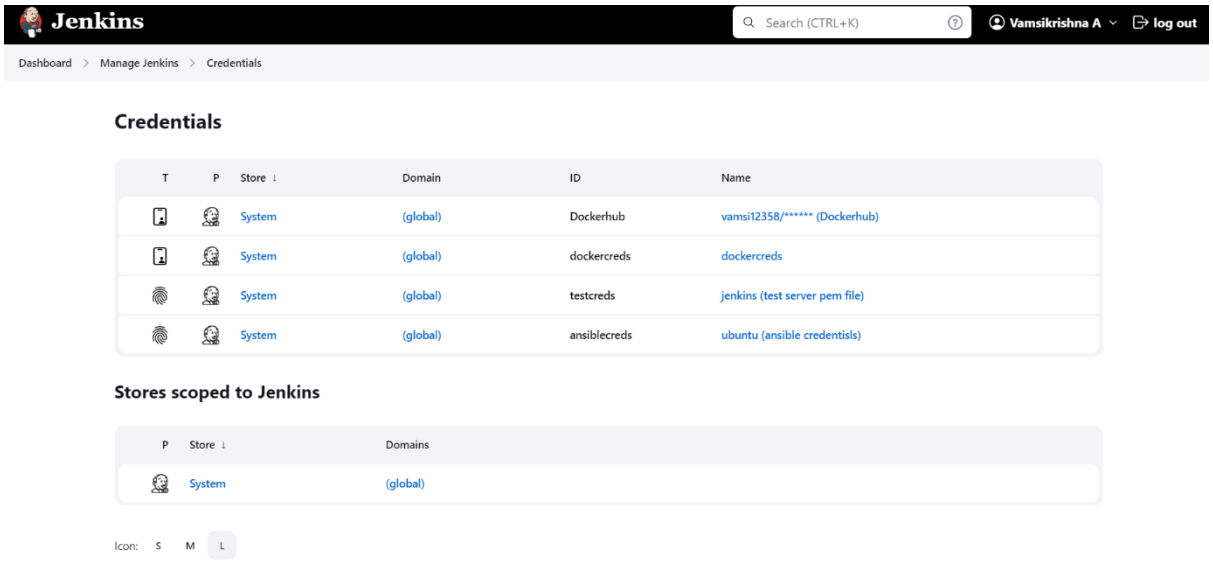
```
pipeline{
   agent any

   stages{
      stage("Code checkout"){
         steps{
            checkout scmGit(branches: [[name: '*/main']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/vamsikrishna918/Insurance-
Web-Applicaion_E2E']])
         }
      }
       stage("Build"){
         steps{
            echo "****building with maven****"
            sh '''mvn clean package '''

         }
      }
      stage("publish the report")
```

```
{
   steps{
  echo "generating test reports"
  publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/insureme project/target/surefire-reports', reportFiles: 'index.html', reportName: 'HTML Report', reportTitles: '', useWrapperFileDirectly: true])
   }
}
stage("Image prune"){
   steps{
      echo "****deleting the previous images***"
      sh ' docker image prune -af '

   }
}


stage("Build Docker image"){
   steps{
      script {
       echo "****Creating Docker image****"
       sh 'docker build -t vamsi12358/insureme .'
       sh 'docker tag vamsi12358/insureme vamsi12358/insuremeapp:v7'
      }
   }
}
```

```
        stage("Push Docker image to Hub"){
            steps{
                script {
                 echo "****Pushing Docker image to Hub****"


                 withCredentials([string(credentialsId: 'dockercreds', variable:
'dockerhubpwd')]) {
                  sh "docker login -u vamsi12358 -p ${dockerhubpwd} docker.io"
                  sh 'docker push vamsi12358/insuremeapp:v7'
                 }


                }
            }
        }
         stage("deploying to Test server"){
            steps{
                script {
                 echo "****Deploying Application to Test server****"
                 ansiblePlaybook become: true, credentialsId: 'ansiblecreds',
disableHostKeyChecking: true, inventory: '/etc/ansible/hosts', playbook:
'ansible-playbook.yml'
                }
            }
        }


    }
}
```

Credentials setup

- Docker
- Deploying server(deployment_test_server) ssh key setup



Jenkins pipeline

Dashboard > Insurance-web application > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Pipeline

Definition

Pipeline script

Script

```
pipeline{
    agent any

    stages{
        stage("Code checkout"){
            steps{
                checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/vamsikrishna
            }
        }
         stage("Build"){
            steps{
                echo "****building with maven****"
                sh '''mvn clean package '''

            }
        }
        stage("publish the report")
        {
            steps{
          echo "generating test reports"
          publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/insurem
            }
        }
        stage("Image prune"){
            steps{
                echo "****deleting the previous images***"
                sh ' docker image prune -af '
```

Save Apply

Dashboard > Insurance-web application > Configuration

Configure

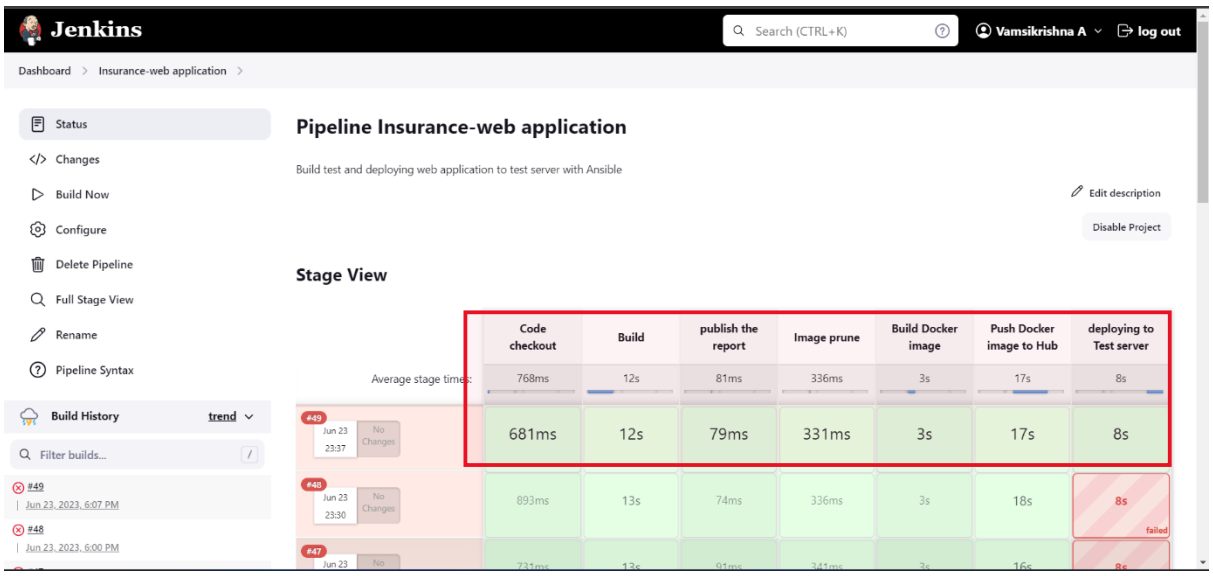
- General
- Advanced Project Options
- Pipeline

```
        stage("Image prune"){
            steps{
                echo "****deleting the previous images***"
                sh ' docker image prune -af '

            }
        }


        stage("Build Docker image"){
            steps{
                script {
                echo "****Creating Docker image****"
                sh 'docker build -t vamsi12358/insureme .'
                sh 'docker tag vamsi12358/insureme vamsi12358/insuremeapp:v7'
                }
            }
        }
        stage("Push Docker image to Hub"){
            steps{
                script {
                echo "****Pushing Docker image to Hub****"

                withCredentials([string(credentialsId: 'dockercreds', variable: 'dockerhubpwd')]) {
                 sh "docker login -u vamsi12358 -p ${dockerhubpwd} docker.io"
                 sh 'docker push vamsi12358/insuremeapp:v7'
                }

                }
            }
        }
         stage("deploying to Test server"){
            steps{
                script {
                echo "****Deploying Application to Test server****"
                ansiblePlaybook become: true, credentialsId: 'ansiblecreds', disableHostKeyChecking: true, inventory: '/etc/ansible/hos
                }
```

Use Groovy Sandbox

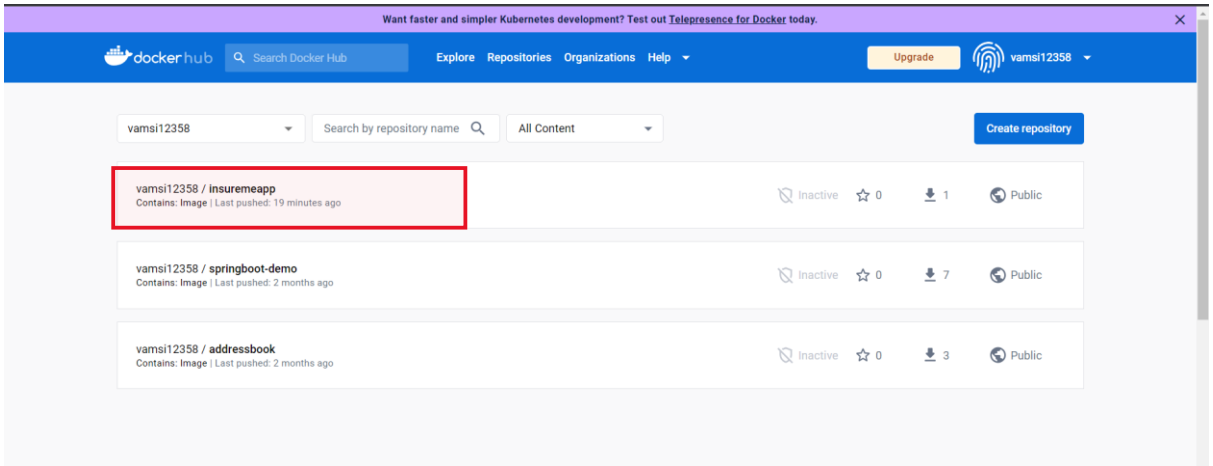
Save Apply

Build the job-



Docker hub-

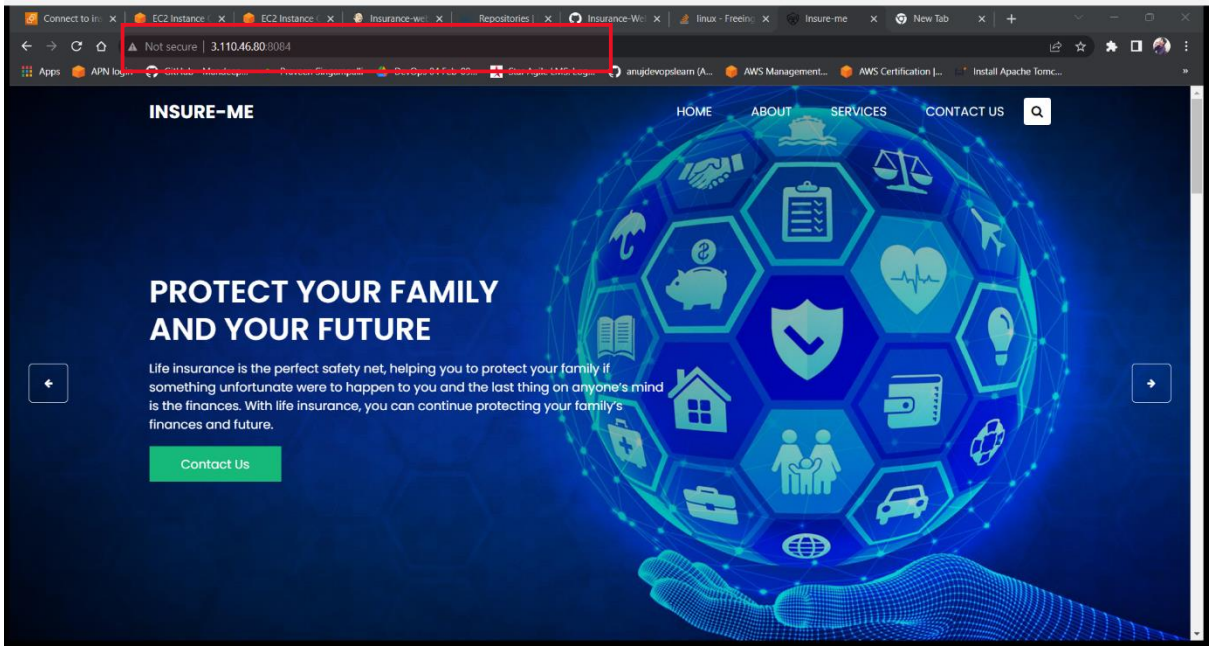
https://hub.docker.com/

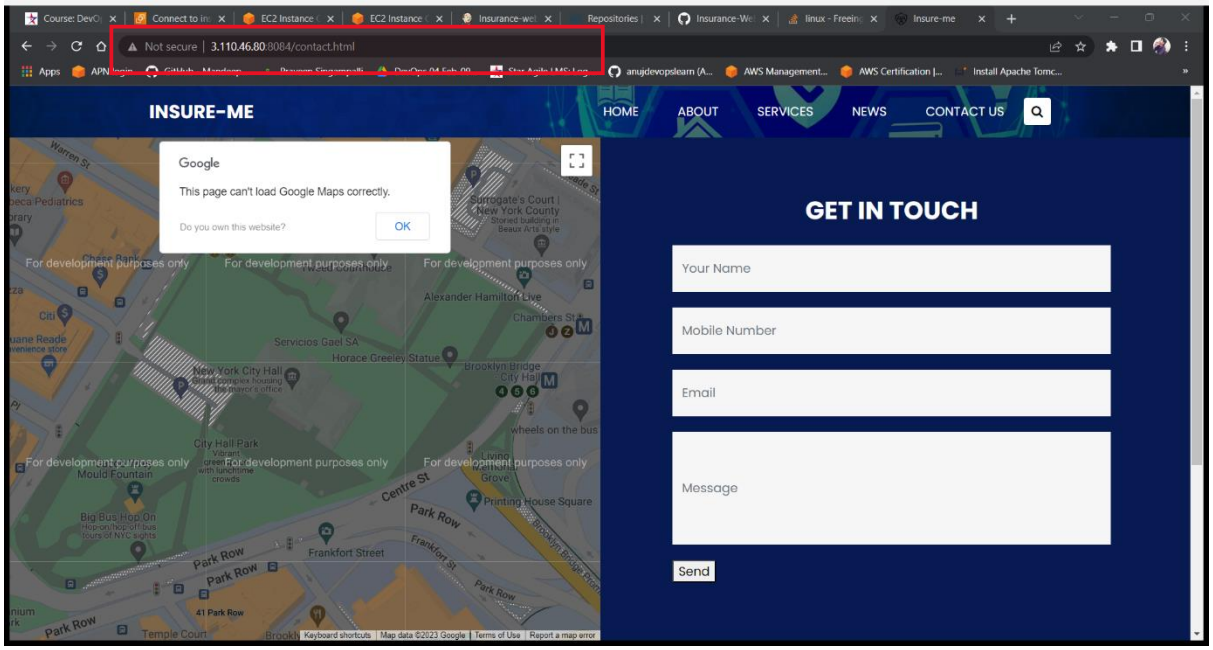


Deployed Application on test server

Accessing the web-applcation on test server

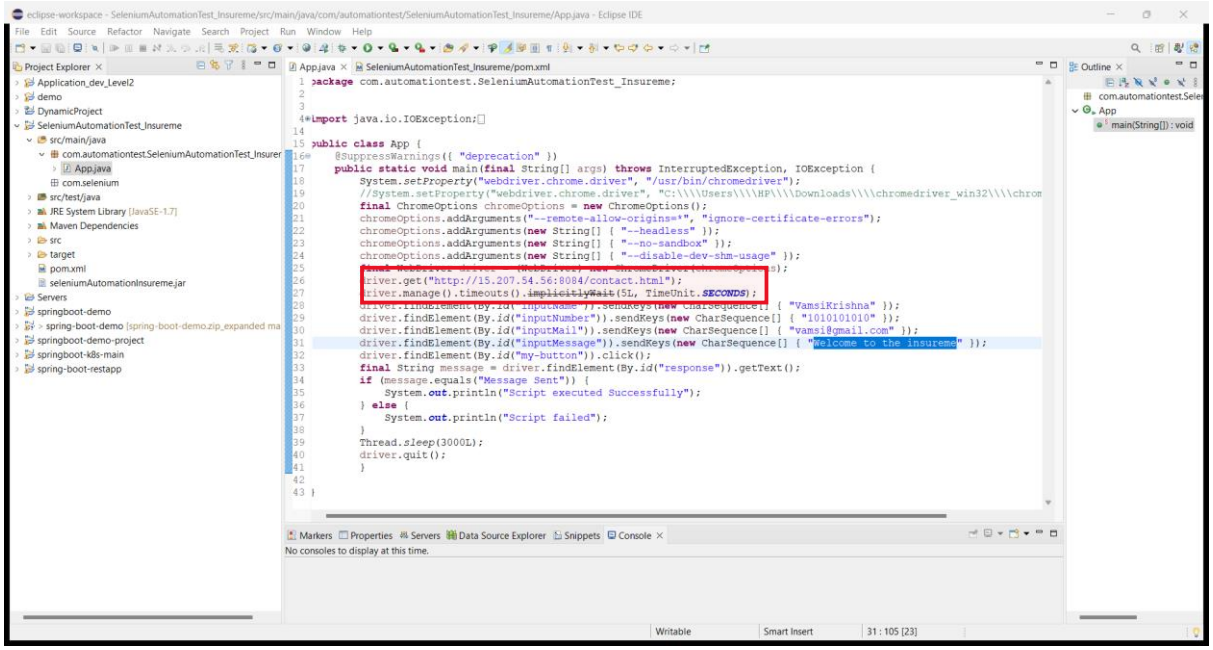
Deployment_test_server public ip:8084/



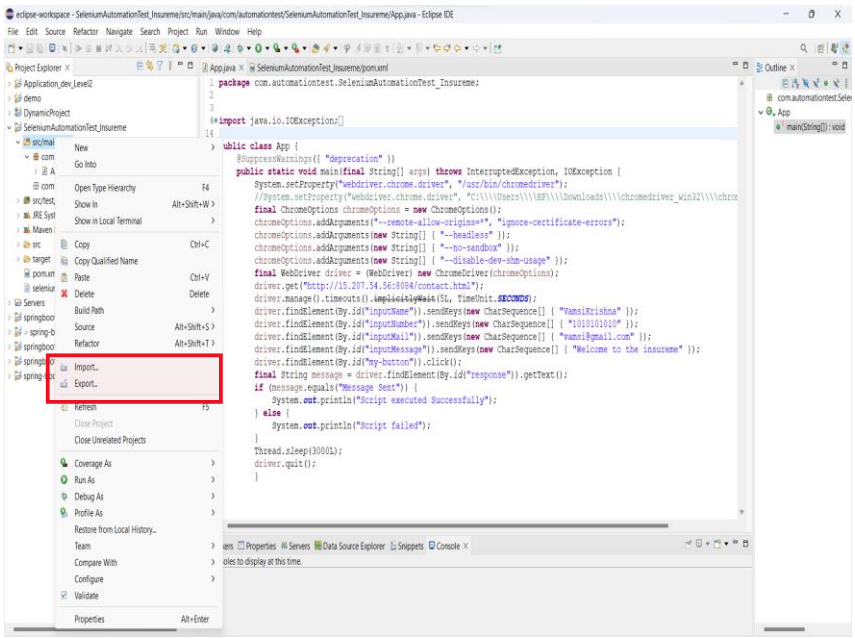


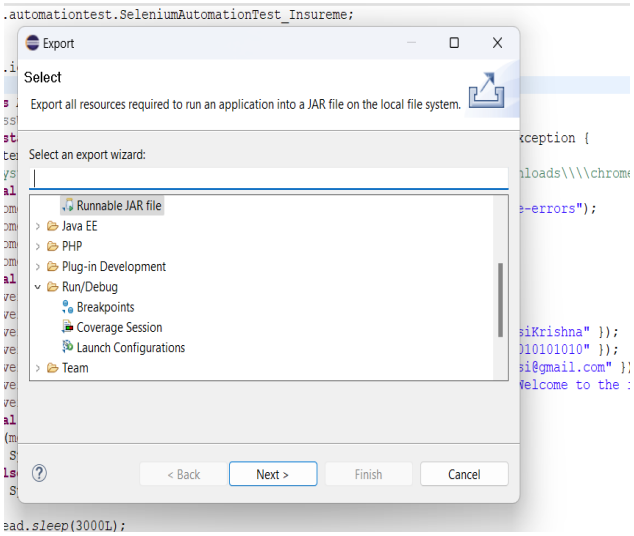
Test -Automation with selenium

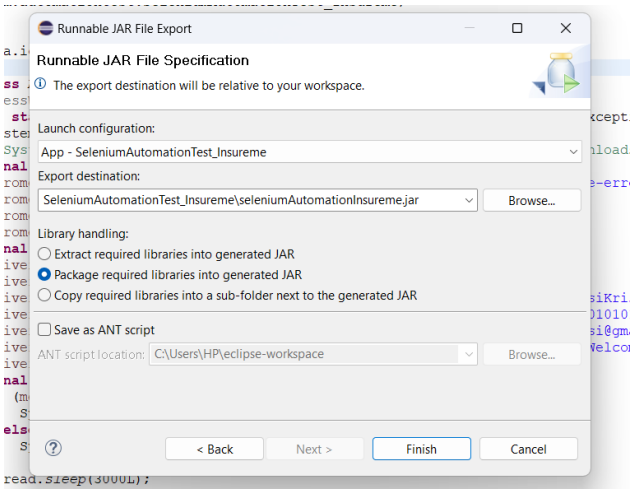
Refer - https://github.com/vamsikrishna918/Selenium-TestAutomation-Insureme



Export to Runnable jar







Push the jar to git hub, we can crate a job and use in pipeline

Installing the Chrome driver in our ubuntu instance (Capstone_Project_Development)

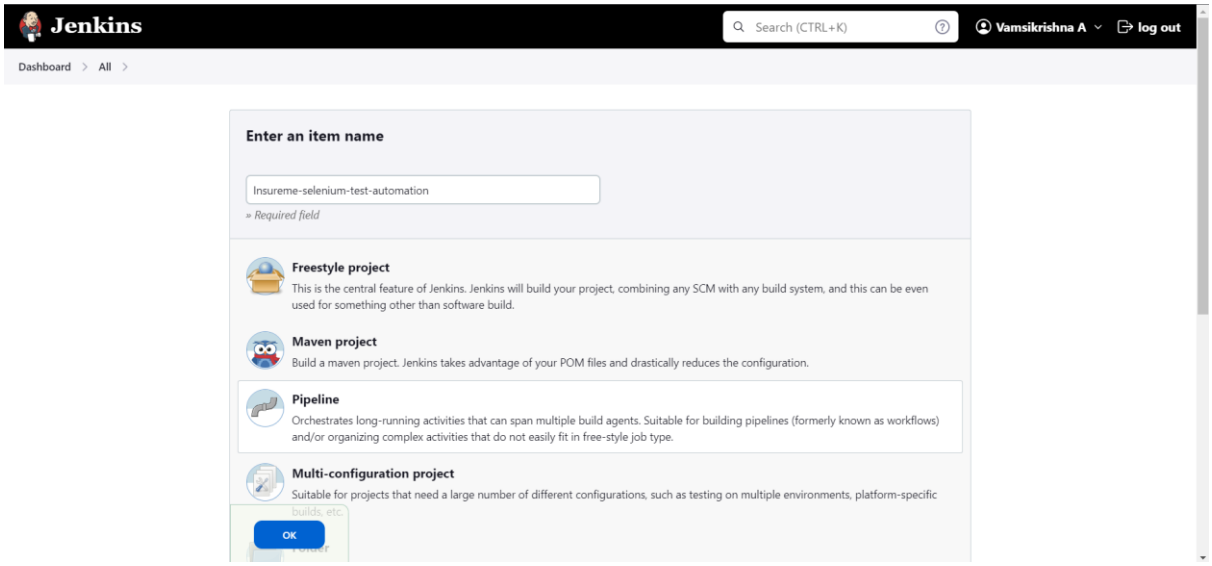
On Capstone_Project_Development instance-

Run the below commands and install.

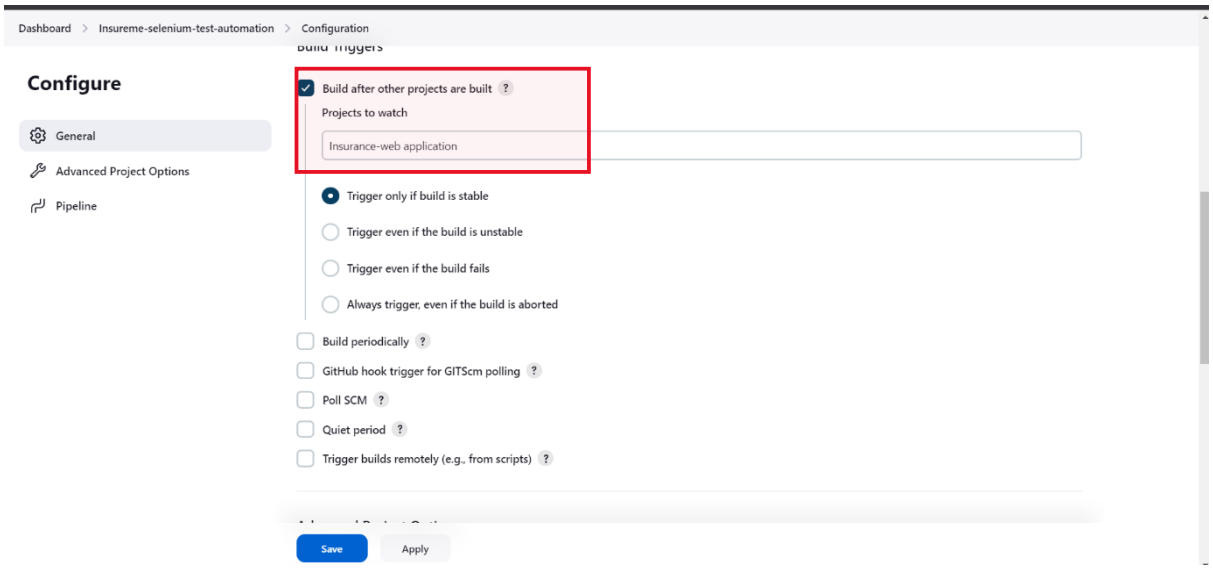
- sudo apt install -y unzip xvfb libxi6 libgconf-2-4
- sudo curl -sS -o - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add
- sudo bash -c "echo 'deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main' >> /etc/apt/sources.list.d/google-chrome.list"
- sudo apt -y update
- sudo apt -y install google-chrome-stable
- google-chrome --version
- wget https://chromedriver.storage.googleapis.com/114.0.5735.90/chromedriver_linux64.zip
- ls
- unzip chromedriver_linux64.zip
- ls
- sudo mv chromedriver /usr/bin/chromedriver

- sudo chown root:root /usr/bin/chromedriver
- sudo chmod +x /usr/bin/chromedriver
- which chromedriver

Creating a job for automation test-

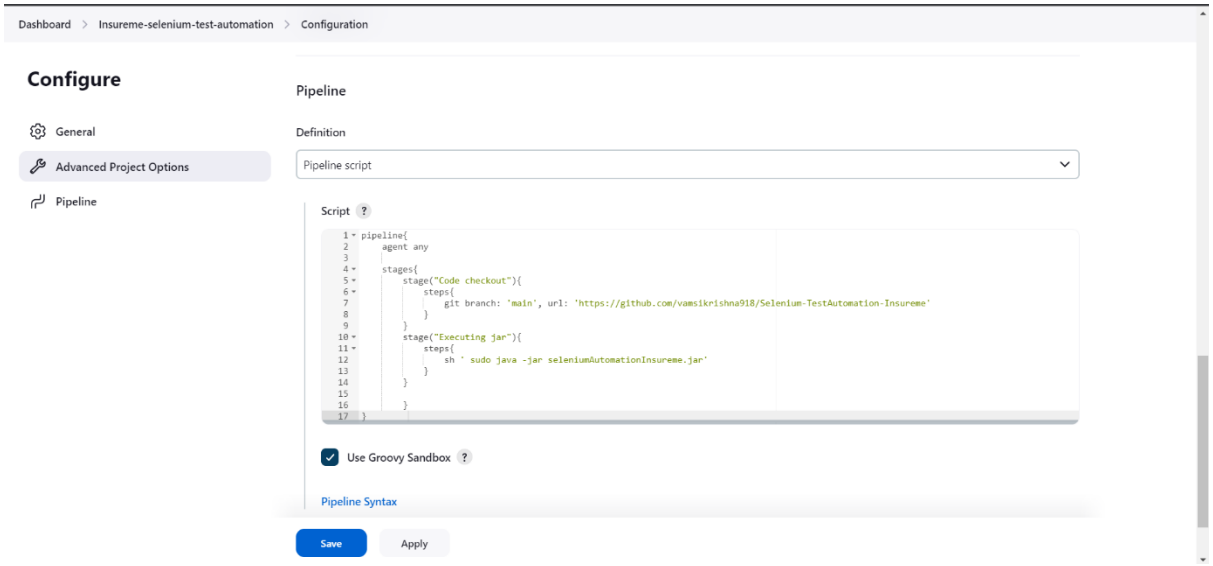


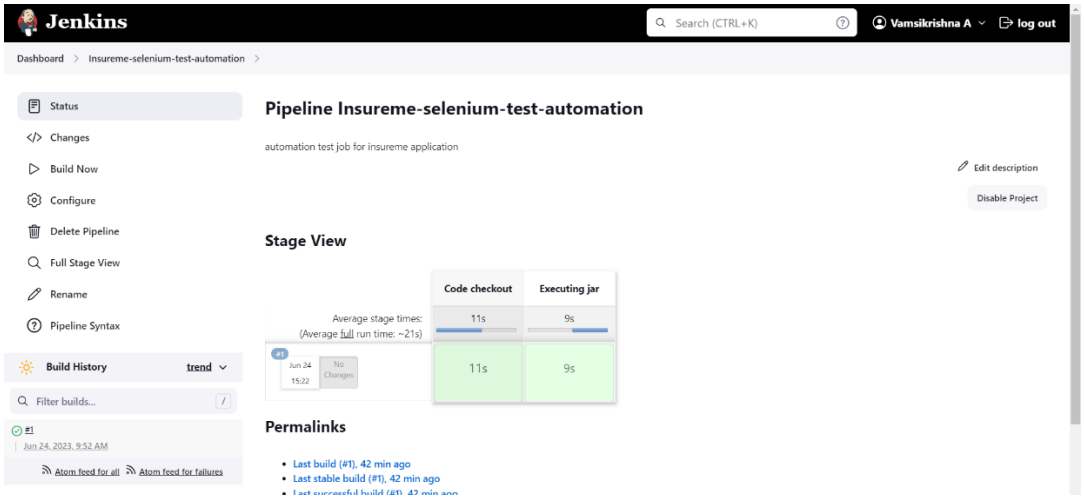
Setup the build trigger, which the automation job builds after the Insure me application deployed on test server(deployment_test_server).

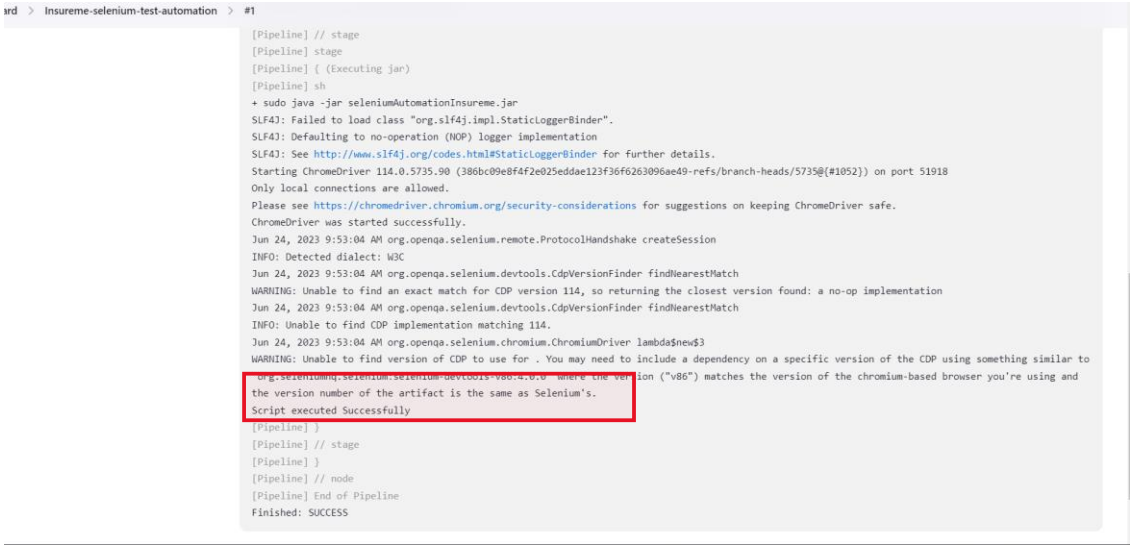


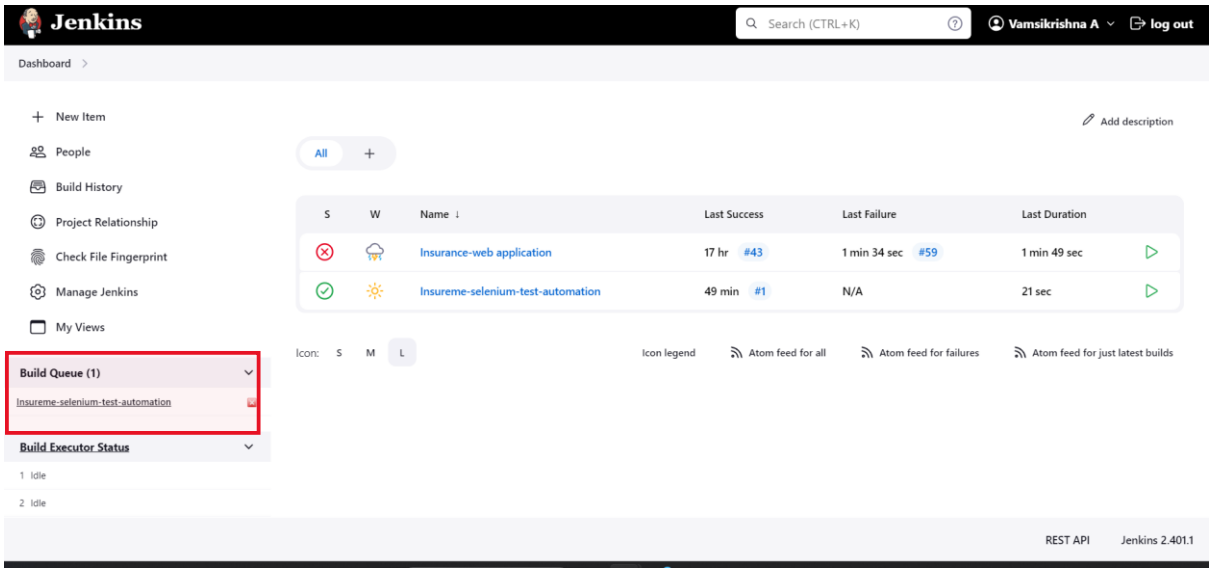
Pipeline script-

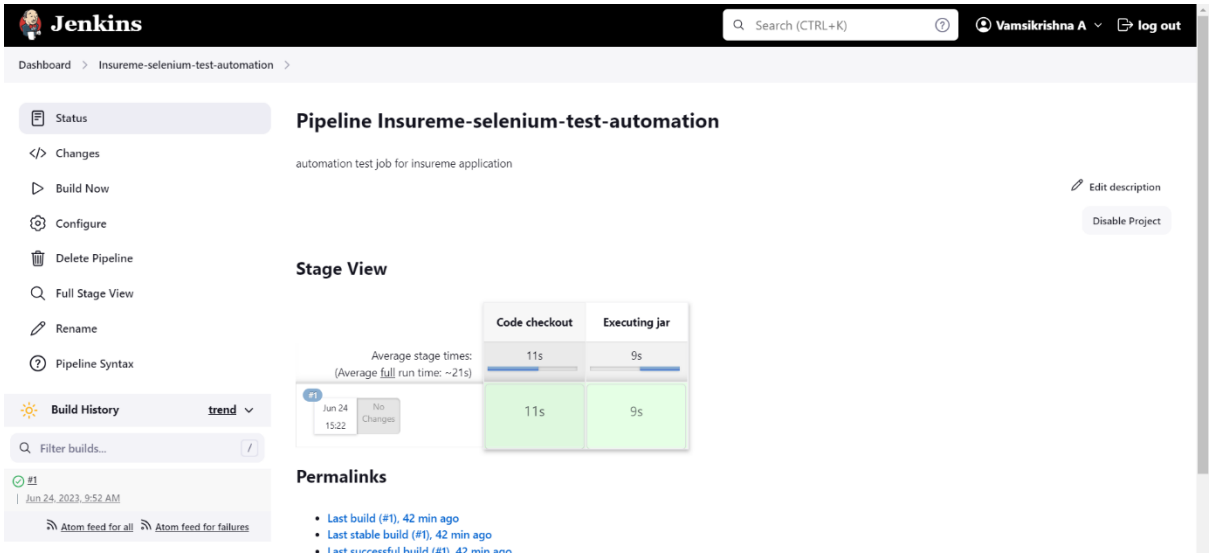
```
pipeline{
   agent any


   stages{
      stage("Test auomation Code checkout"){
         steps{
            git branch: 'main', url: 'https://github.com/vamsikrishna918/Selenium-TestAutomation-Insureme'
         }
      }
      stage("Executing jar"){
         steps{
            sh ' sudo java -jar seleniumAutomationInsureme.jar'
         }
      }


      }
}
```




Jenkins
Search (CTRL+K)
Vamsikrishna A
log out
Dashboard > Insureme-selenium-test-automation >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
Filter builds...
#1
Jun 24, 2023, 9:52 AM
Atom feed for all
Atom feed for failures
Pipeline Insureme-selenium-test-automation
automation test job for insureme application
Edit description
Disable Project
Stage View
Code checkout
Executing jar
Average stage times:
(Average full run time: ~21s)
11s
9s
Jun 24
15:22
No Changes
Permalinks
Last build (#1), 42 min ago
Last stable build (#1), 42 min ago


Insureme-selenium-test-automation > #1
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Executing jar)
[Pipeline] sh
+ sudo java -jar seleniumAutomationInsureme.jar
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 114.0.5735.90 (386bc09e8f4f2e025eddae123f36f6263096ae49-refs/branch-heads/5735@{#1052}) on port 51918
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Jun 24, 2023 9:53:04 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Jun 24, 2023 9:53:04 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 114, so returning the closest version found: a no-op implementation
Jun 24, 2023 9:53:04 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Unable to find CDP implementation matching 114.
Jun 24, 2023 9:53:04 AM org.openqa.selenium.chromium.ChromiumDriver lambdanew3
WARNING: Unable to find version of CDP to use for . You may need to include a dependency on a specific version of the CDP using something similar to
the version number of the artifact is the same as Selenium's.
Script executed Successfully
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS


Jenkins
Search (CTRL+K)
Vamsikrishna A
log out
Dashboard >
New Item
People
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Build Queue (1)
Insureme-selenium-test-automation
Build Executor Status
1 Idle
2 Idle
Add description
All
S
W
Name ↓
Last Success
Last Failure
Last Duration
Insurance-web application
17 hr #43
1 min 34 sec #59
1 min 49 sec
Insureme-selenium-test-automation
49 min #1
N/A
21 sec
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds
REST API
Jenkins 2.401.1



Sample executed on local machine-