

Medicure

Healthcare Domain

This project is developing an application which interacts about healthcare parameters.

1. Git - For version control for tracking changes in the code files .
2. Jenkins - For continuous integration and continuous deployment .
3. Docker - For containerizing applications .
4. Ansible - Configuration management tools.
5. Selenium - For automating tests on the deployed web application.
6. Terraform - For creation of infrastructure.
7. Kubernetes – for running containerized application in managed cluster.

Here is the EC2 server created with following specifications to act as master server. It is configured with java, maven, docker, Jenkins, Ansible and Terraform on the instance, this server acts as master server, which is used to implement CI/CD pipeline script.

Public IP: *3.110.151.185*

IAM Role: srija with EC2 Full Access.

Java Version: jdk-17

Maven version: maven 3.6.3.

Ansible Version: 2.14.5

Terraform Version: v1.4.6

Docker Version: 20.10.21

Jenkins configured on URL: *3.110.151.185:8080*

This is the EC2 dashboard showing that master-server is up and running.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table with one row for the instance 'Master-server'. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
Master-server	i-05578e7537bce2225	Running	t2.medium	2/2 checks passed...	No alarms	ap-south-1

Below the table, a message states: "Instances: i-0738d43f1df97748a (test-server), i-062448df6fee5fd37 (prod-server)".

Here is the screenshot of IP address of master server configuration.

The screenshot shows the AWS EC2 Instances Details page for the instance 'i-05578e7537bce2225'. A red arrow points to the 'Public IPv4 address' field, which contains the value '3.110.151.185'. Other visible details include the instance ID, state, VPC ID, and subnet ID.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-05578e7537bce2225 (Master-server)	3.110.151.185 open address	172.31.0.4

Other visible details include:

- IPv6 address: -
- Hostname type: IP name: ip-172-31-0-4.ap-south-1.compute.internal
- Answer private resource DNS name: IPv4 (A)
- Auto-assigned IP address: 3.110.151.185 [Public IP]
- IAM Role: -
- Instance state: Running
- Private IP DNS name (IPv4 only): ip-172-31-0-4.ap-south-1.compute.internal
- Instance type: t2.medium
- VPC ID: vpc-0243302b4b1acc768
- Subnet ID: subnet-0d2afffc0a2262b6f2
- Elastic IP addresses: -
- AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
- Auto Scaling Group name: -

This is the figure showing master server is connected through SSH connection using mobaxterm and all the software packages installed with their versions and status.

```

root@ip-172-31-0-4:/home/ubuntu# java --version
openjdk 17.0.6 2023-01-17
OpenJDK Runtime Environment (build 17.0.6+10-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 17.0.6+10-Ubuntu-0ubuntu122.04, mixed mode, sharing)
root@ip-172-31-0-4:/home/ubuntu# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 17.0.6, vendor: Private Build, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-1024-aws", arch: "amd64", family: "unix"
root@ip-172-31-0-4:/home/ubuntu# ansible --version
ansible [core 2.14.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  additional python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] (/usr/bin/python3)
    jinja version = 3.0.3
    libyaml = True
root@ip-172-31-0-4:/home/ubuntu# terraform --version
Terraform v1.4.6
on linux_amd64
root@ip-172-31-0-4:/home/ubuntu# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-05-04 05:11:32 UTC; 1h 55min ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
      Main PID: 800 (dockerd)
        Tasks: 10
       Memory: 90.1M
          CPU: 16.272s
         CGroup: /system.slice/docker.service
                   └─800 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
May 04 05:11:31 ip-172-31-0-4 dockerd[800]: time="2023-05-04T05:11:31.438165358Z" level=info msg="[graphdriver] using prior storage driver as fallback: overlay"
May 04 05:11:31 ip-172-31-0-4 dockerd[800]: time="2023-05-04T05:11:31.590455094Z" level=info msg="Loading containers: start."

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>


```

root@ip-172-31-0-4:/home/ubuntu# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-05-04 05:11:44 UTC; 1h 55min ago
     Main PID: 453 (java)
       Tasks: 54 (limit: 4686)
      Memory: 2.8G
        CPU: 5min 20.514s
       CGroup: /system.slice/jenkins.service
                 └─453 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 04 06:48:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:48:15.550+0000 [id=1550]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 06:50:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:50:15.550+0000 [id=1559]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 06:52:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:52:15.551+0000 [id=1568]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 06:54:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:54:15.551+0000 [id=1577]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 06:56:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:56:15.550+0000 [id=1586]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 06:58:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 06:58:15.550+0000 [id=1595]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 07:00:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 07:00:15.550+0000 [id=1604]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 07:02:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 07:02:15.550+0000 [id=1613]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 07:04:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 07:04:15.550+0000 [id=1622]           WARNING   o.j.p.p.DiskUsageCollector#coll
May 04 07:06:15 ip-172-31-0-4 jenkins[453]: 2023-05-04 07:06:15.550+0000 [id=1631]           WARNING   o.j.p.p.DiskUsageCollector#coll

```

Lines 1-20/20 (END)

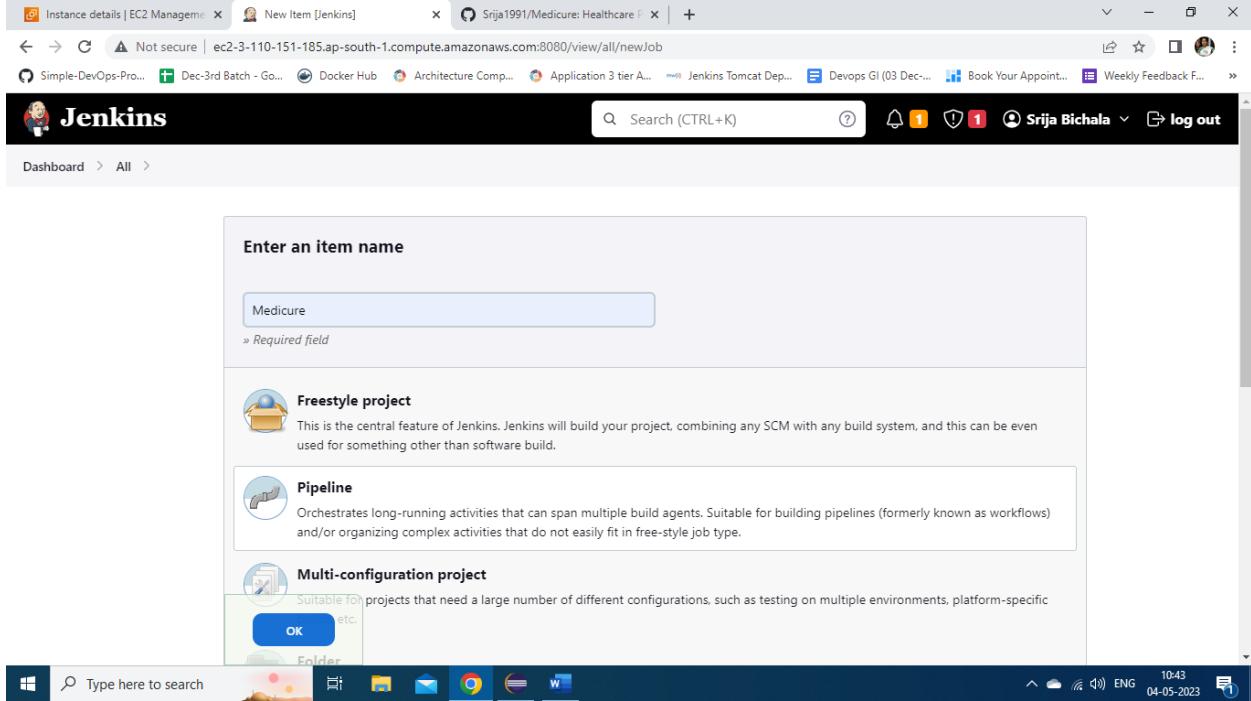
```

root@ip-172-31-0-4:/home/ubuntu#

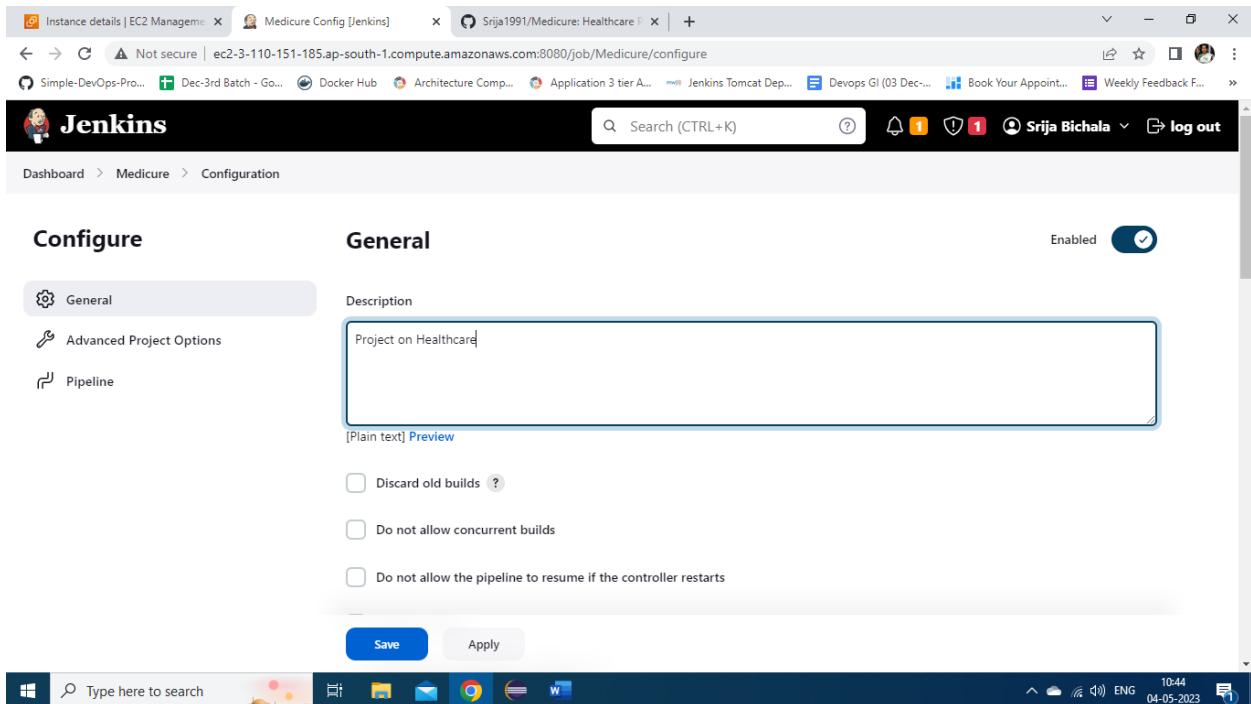
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Jenkins is accessed on URL : 3.110.151.185:8080 port and a new job is created named **Medicure** to build the packages using maven build tool, then create docker image out of it using **Docker file** and then push the docker image onto Docker hub and use the image later to create a deployment on test server created using terraform and configured using ansible tool.



Here is the description of job Medicure.



The source code for this job is fetched through Git hub and the file name to execute pipeline script is "**Jenkinsfile**".

Git URL: <https://github.com/Srija1991/Medicure.git>.

This screenshot shows the Jenkins Pipeline configuration page for the 'Medicure' job. The 'Pipeline' tab is selected. Under the 'Definition' section, it is set to 'Pipeline script from SCM'. The 'SCM' section is expanded, showing 'Git' selected. The 'Repositories' section contains a single repository with the URL 'https://github.com/Srija1991/Medicure.git'. The 'Credentials' dropdown is set to '- none -'. At the bottom, there are 'Save' and 'Apply' buttons.

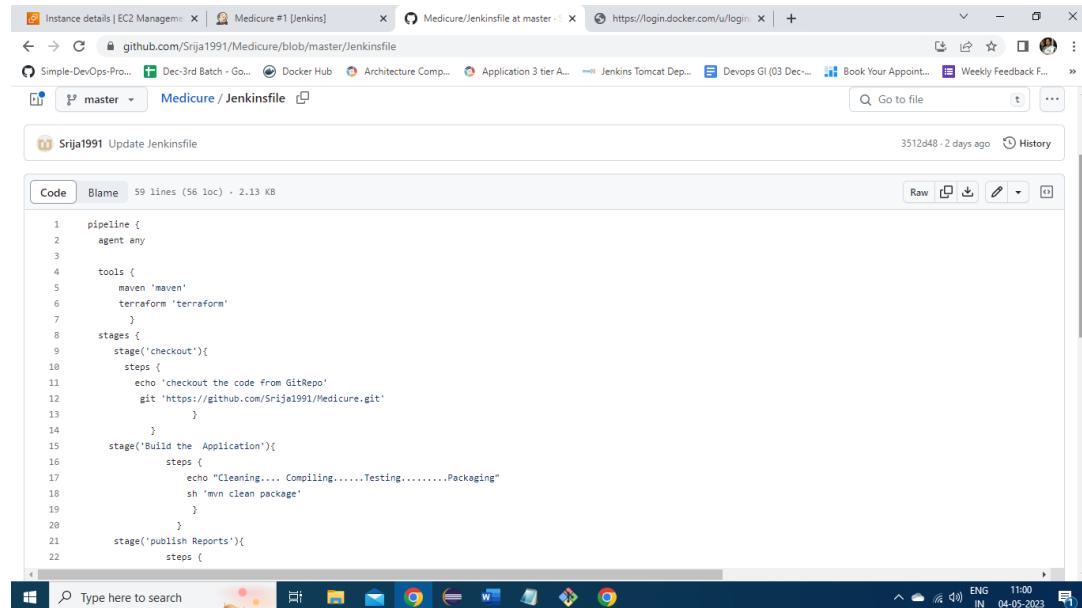
This screenshot shows the Jenkins Pipeline configuration page for the 'Medicure' job. The 'Pipeline' tab is selected. The 'Script Path' field is set to 'Jenkinsfile'. The 'Lightweight checkout' checkbox is checked. Below this, the 'Pipeline Syntax' section is visible. At the bottom, there are 'Save' and 'Apply' buttons.

Here is Jenkins file illustrating the flow of CI/CD pipeline.

Stage 1: Git checkout : Use the code available on git URL mentioned in the file Jenkinsfile.

Stage2: Build the application using maven command “*mvn clean package*”.

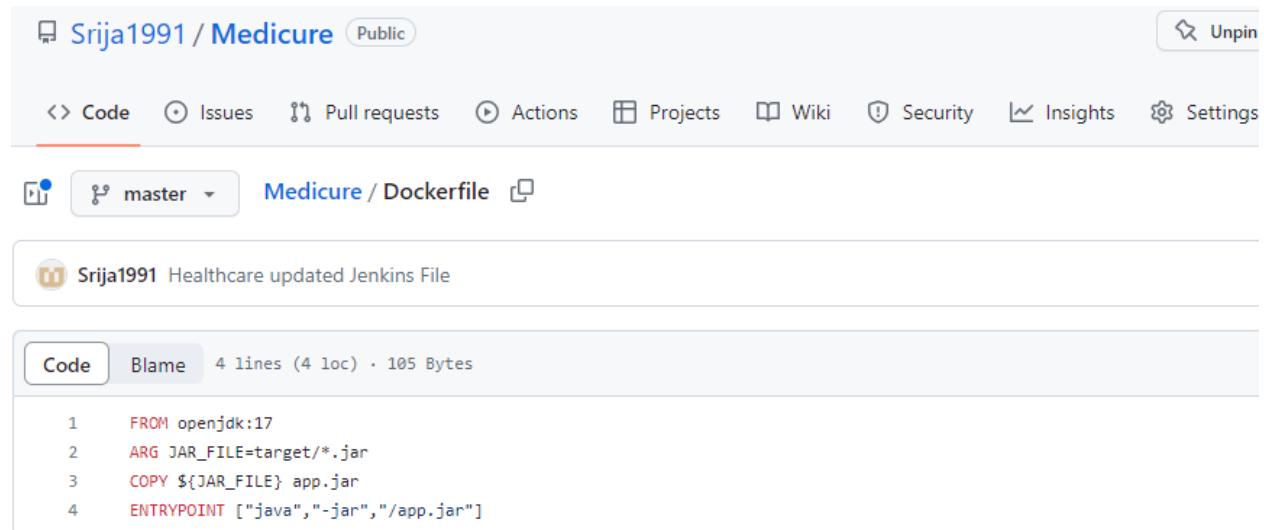
Stage 3: Publish the test reports using HTML publisher.



The screenshot shows a browser window with multiple tabs open. The active tab displays the Jenkinsfile for the 'Medicure' repository on GitHub. The Jenkinsfile contains the following code:

```
1 pipeline {
2     agent any
3
4     tools {
5         maven 'maven'
6         terraform 'terraform'
7     }
8     stages {
9         stage('checkout'){
10            steps {
11                echo 'checkout the code from GitRepo'
12                git 'https://github.com/Srija1991/Medicure.git'
13            }
14        }
15        stage('Build the Application'){
16            steps {
17                echo "Cleaning.... Compiling.....Testing.....Packaging"
18                sh 'mvn clean package'
19            }
20        }
21        stage('publish Reports'){
22            steps {
```

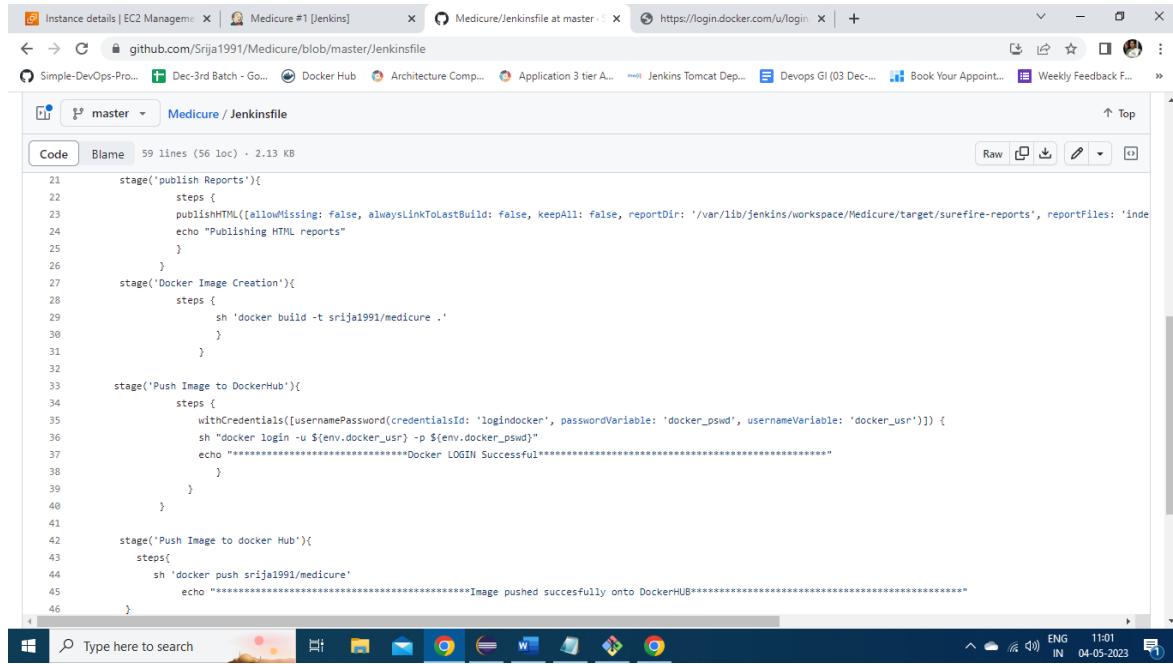
Stage 4: Create docker image using the Dockerfile and jar file created from build success step.



The screenshot shows a browser window with multiple tabs open. The active tab displays the Dockerfile for the 'Medicure' repository on GitHub. The Dockerfile contains the following code:

```
1 FROM openjdk:17
2 ARG JARFILE=target/*.jar
3 COPY ${JARFILE} app.jar
4 ENTRYPOINT ["java","-jar","/app.jar"]
```

Stage 5: Push the docker mage onto docker hub using appropriate credentials with the name *srija1991/medicure*.



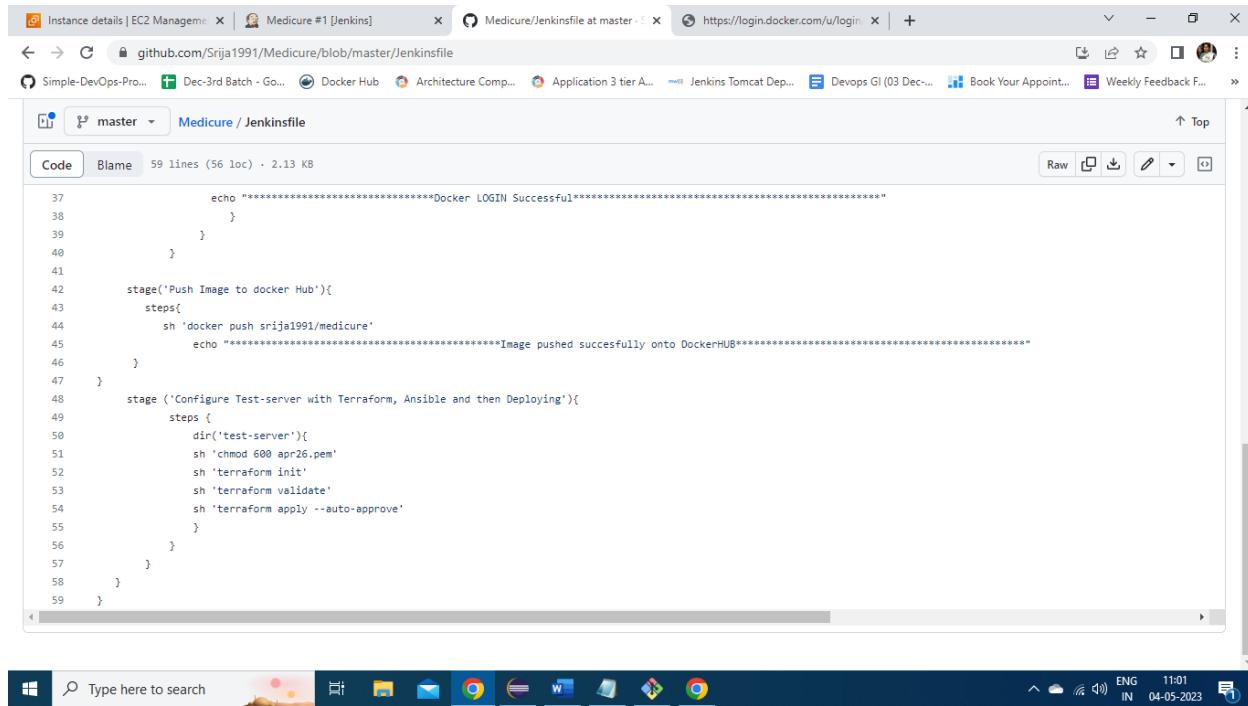
```
stage('publish Reports'){
    steps {
        publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/Medicure/target/surefire-reports', reportFiles: 'index.html'])
    }
}

stage('Docker Image Creation'){
    steps {
        sh 'docker build -t srija1991/medicure .'
    }
}

stage('Push Image to DockerHub'){
    steps {
        withCredentials([usernamePassword(credentialsId: 'logindocker', passwordVariable: 'docker_pswd', usernameVariable: 'docker_usr')]) {
            sh 'docker login -u ${env.docker_usr} -p ${env.docker_pswd}'
            echo "*****Docker LOGIN Successful*****"
        }
    }
}

stage('Push Image to docker Hub'){
    steps{
        sh 'docker push srija1991/medicure'
        echo "*****Image pushed succesfully onto DockerHUB*****"
    }
}
```

Now stage 6 is used to create an EC2 instance which acts as a test -server. Here is test-server folder which has *main.tf* file to create an EC2 instance in the region mentioned and attach the keypair “*apr26*”.



```
echo "*****Docker LOGIN Successful*****"

stage('Push Image to docker Hub'){
    steps{
        sh 'docker push srija1991/medicure'
        echo "*****Image pushed succesfully onto DockerHUB*****"
    }
}

stage ('Configure Test-server with Terraform, Ansible and then Deploying'){
    steps {
        dir('test-server'){
            sh 'chmod 600 apr26.pem'
            sh 'terraform init'
            sh 'terraform validate'
            sh 'terraform apply --auto-approve'
        }
    }
}
```

Srija1991 / Medicure Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master Medicure / test-server

Srija1991 Update playbook.yml

Name	Last commit message
..	
ansible.cfg	Healthcare updated Jenkins File
apr26.pem	Healthcare updated Jenkins File
inventory	Healthcare updated Jenkins File
main.tf	Healthcare updated Jenkins File
playbook.yml	Update playbook.yml
provider.tf	Healthcare updated Jenkins File

Main.tf file with the terraform script for creating test-server and run ansible playbook to install necessary packages to deploy application onto it.

```

resource "aws_instance" "test-server" {
  ami           = "ami-02eb7e4783e7e9317"
  instance_type = "t2.micro"
  key_name      = "apr26"
  vpc_security_group_ids = ["sg-0c69f259b0ea97dc0"]
  tags = [
    { Name = "test-server" }
  ]

  provisioner "local-exec" {
    command = "sleep 60 && echo 'Instance ready.'"
  }

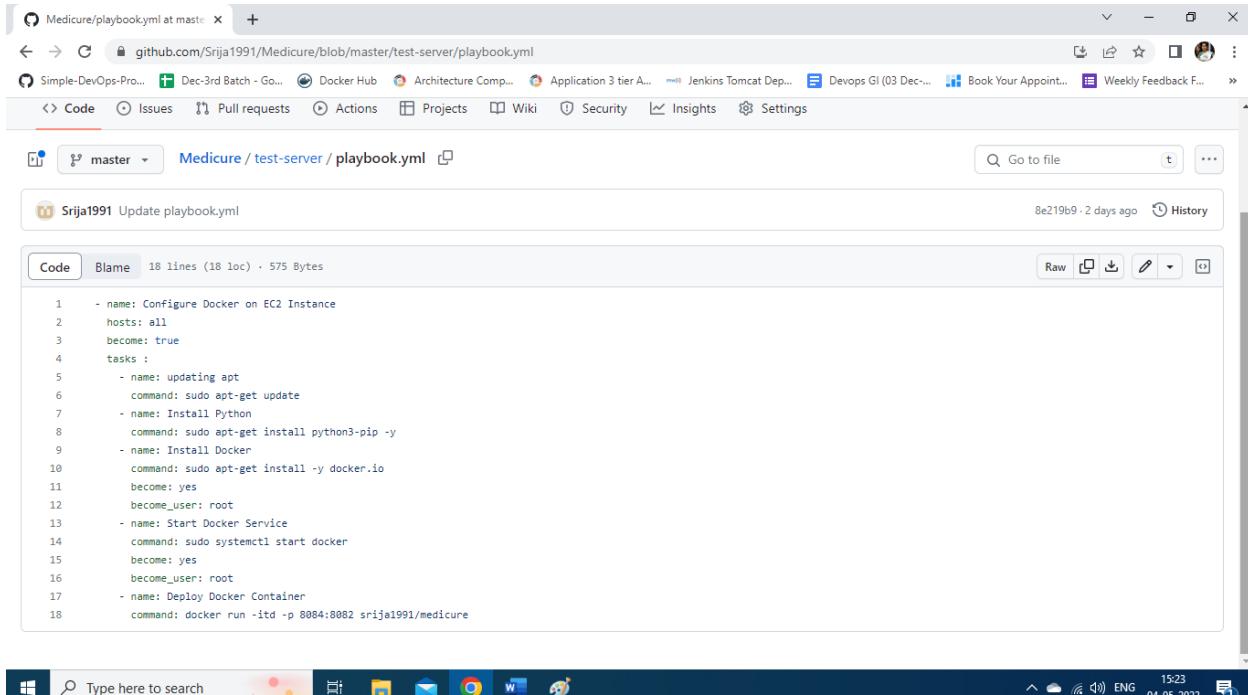
  connection {
    type     = "ssh"
    user     = "ubuntu"
    private_key = file("./apr26.pem")
    host     = self.public_ip
  }
}

provisioner "local-exec" {
  command = "echo ${aws_instance.test-server.public_ip} > inventory"
}

provisioner "local-exec" {
  command = "ansible-playbook /var/lib/jenkins/workspace/Medicure/test-server/playbook.yml"
}

```

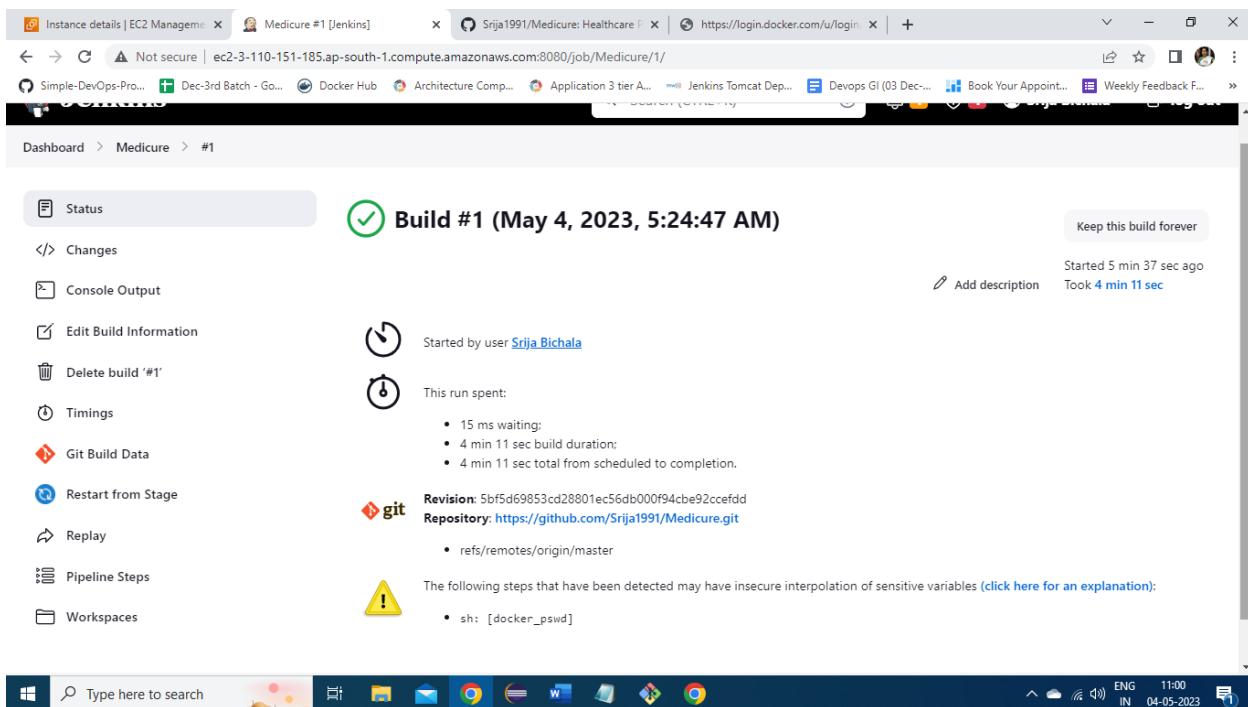
Here is *playbook.yml* to configure the test-server created and deploy the application on port number 8084.



The screenshot shows a GitHub code editor window for the file `Medicure/test-server/playbook.yml`. The file contains the following YAML code:

```
1 - name: Configure Docker on EC2 Instance
2   hosts: all
3   become: true
4   tasks :
5     - name: updating apt
6       command: sudo apt-get update
7     - name: Install Python
8       command: sudo apt-get install python3-pip -y
9     - name: Install Docker
10      command: sudo apt-get install -y docker.io
11      become: yes
12      become_user: root
13     - name: Start Docker Service
14       command: sudo systemctl start docker
15       become: yes
16       become_user: root
17     - name: Deploy Docker Container
18       command: docker run -itd -p 8084:8082 srija1991/medicure
```

Now the job is build sucessfully and here is console output .



The screenshot shows the Jenkins job details for Medicure #1. The build was successful, starting at 5:24:47 AM on May 4, 2023. It took 4 min 11 sec. The build log indicates:

```
Started by user Srijita Bichala
This run spent:
  • 15 ms waiting;
  • 4 min 11 sec build duration;
  • 4 min 11 sec total from scheduled to completion.

Revision: 5bf5d69853cd28801ec56db000f94cbe92ccefd
Repository: https://github.com/Srija1991/Medicure.git
  • refs/remotes/origin/master
```

The following steps that have been detected may have insecure interpolation of sensitive variables (click [here](#) for an explanation):

- sh: [docker_pswd]

The git checkout is successful.

The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The output window displays the following log:

```
Started by user Srija Bichala
Obtained Jenkinsfile from git https://github.com/Srija1991/Medicure.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Medicure
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: git
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Medicure/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Srija1991/Medicure.git # timeout=10
Fetching upstream changes from https://github.com/Srija1991/Medicure.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Srija1991/Medicure.git +refs/heads/*:refs/remotes/origin/*

```

Stage 2 Build the package is successful.

The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The output window displays the following log:

```
+ mvn clean package
[1;34mINFO[m] Scanning for projects...
[1;34mINFO[m]
[1;34mINFO[m] [1m-----[0;36mcom.project.staragile:medicure[0;1m >-----[m
[1;34mINFO[m] [1mBuilding medicure 0.0.1-SNAPSHOT[m
[1;34mINFO[m] [1m-----[0;32m[ jar ]-----[m
[1;34mINFO[m]
[1;34mINFO[m] [1m--- [0;32mmaven-clean-plugin:3.2.0:clean@[1m(default-clean)@0;1m ---[m
[1;34mINFO[m] Deleting /var/lib/jenkins/workspace/Medicure/target
[1;34mINFO[m]
[1;34mINFO[m] [1m--- [0;32mmaven-resources-plugin:3.2.0:resources@[1m(default-resources)@0;1m ---[m
[1;34mINFO[m] [1m--- [0;32mmaven-compiler-plugin:3.10.1:compile@[1m(default-compile)@0;1m ---[m
[1;34mINFO[m] Changes detected - recompiling the module!
[1;34mINFO[m] Compiling 5 source files to /var/lib/jenkins/workspace/Medicure/target/classes
[1;34mINFO[m]
[1;34mINFO[m] [1m--- [0;32mmaven-resources-plugin:3.2.0:testResources@[1m(default-testResources)@0;1m ---[m
[1;34mINFO[m] Using 'UTF-8' encoding to copy filtered resources.
[1;34mINFO[m] Using 'UTF-8' encoding to copy filtered properties files.
[1;34mINFO[m] Copying 1 resource
[1;34mINFO[m] Copying 32 resources
[1;34mINFO[m]
[1;34mINFO[m] [1m--- [0;32mmaven-testng-plugin:1.1.1:testng@[1m(default-testng)@0;1m ---[m
[1;34mINFO[m] skien non-existing resourceDirectory /var/lib/jenkins/workspace/Medicure/src/test/resources
```

A screenshot of a Windows desktop environment. At the top, there's a taskbar with several pinned icons: File Explorer, Mail, Google Chrome, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, and Microsoft Edge. Below the taskbar is a window titled "Medicure #1 Console [Jenkins]" showing Jenkins logs. The logs detail a Maven build process for a project named "Medicure". It shows the creation of a jar file, repackaging of dependencies, and publishing of reports. The log ends with a Docker build command being executed. At the bottom right of the screen, there's a system tray with icons for battery, signal strength, and network, along with a date and time indicator (04-05-2023, 11:03).

```
[0;34mINFO[m] Building jar: /var/lib/jenkins/workspace/Medicure/target/medicure-0.0.1-SNAPSHOT.jar
[0;34mINFO[m] Replacing main artifact with repackaged archive
[0;34mINFO[m] [1m-----[m
[0;34mINFO[m] [1mBUILD SUCCESS[m
[0;34mINFO[m] [1m-----[m
[0;34mINFO[m] Total time: 10.743 s
[0;34mINFO[m] Finished at: 2023-05-04T05:25:33Z
[0;34mINFO[m] [1m-----[m
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (publish Reports)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] publishHTML
[htmlpublisher] Archiving HTML reports...
[htmlpublisher] Archiving at PROJECT level /var/lib/jenkins/workspace/Medicure/target/surefire-reports to
/var/lib/jenkins/jobs/Medicure/htmlreports/HTML_20Report
[Pipeline] echo
Publishing HTML reports
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Docker Image Creation)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker build -t srija1991/medicure .
Sending build context to Docker daemon 765.2MB
```

Stage 3: Published the reports to specified folder.

Stage 4: Docker image is created.

A screenshot of a Windows desktop environment, identical to the previous one, showing the continuation of the Jenkins log. The log now includes the execution of a Docker build command, which sends the build context to the Docker daemon. The log concludes with the creation of a Docker image named "srija1991/medicure". The system tray at the bottom right shows the date as 04-05-2023 and the time as 11:03.

```
[Pipeline] echo
Publishing HTML reports
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Docker Image Creation)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker build -t srija1991/medicure .
Sending build context to Docker daemon 765.2MB
```

Image is pushed successfully onto docker hub.

Instance details | EC2 Manager | Medicure #1 Console [Jenkins] | Medicure/test-server at master | Docker Hub | Medicure

hub.docker.com

Simple-Dev Reload this page 3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec... Book Your Appoint... Weekly Feedback F... »

Want faster and simpler Kubernetes development? Test out Telepresence for Docker today.

docker hub Search Docker Hub Explore Repositories Organizations Help Upgrade srija1991

srija1991 Search by repository name All Content Create repository

srija1991 / medicure
Contains: Image | Last pushed: 12 minutes ago

srija1991 / financeme
Contains: Image | Last pushed: 6 days ago

srija1991 / insureme
Contains: Image | Last pushed: 12 days ago

srija1991 / springboot-demo
Contains: Image | Last pushed: 2 months ago

Inactive 0 12 Public

Inactive 0 7 Public

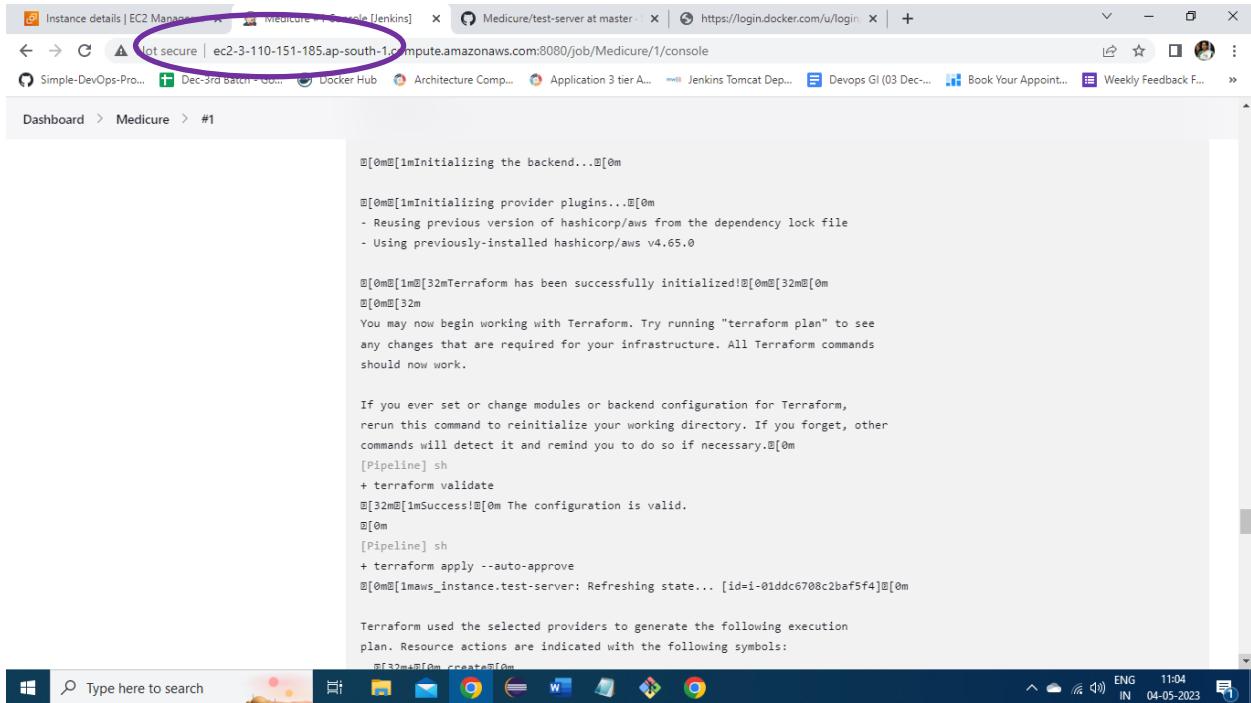
Inactive 0 4 Public

Inactive 0 10 Public

Create an Organization Manage Docker Hub repositories with your team

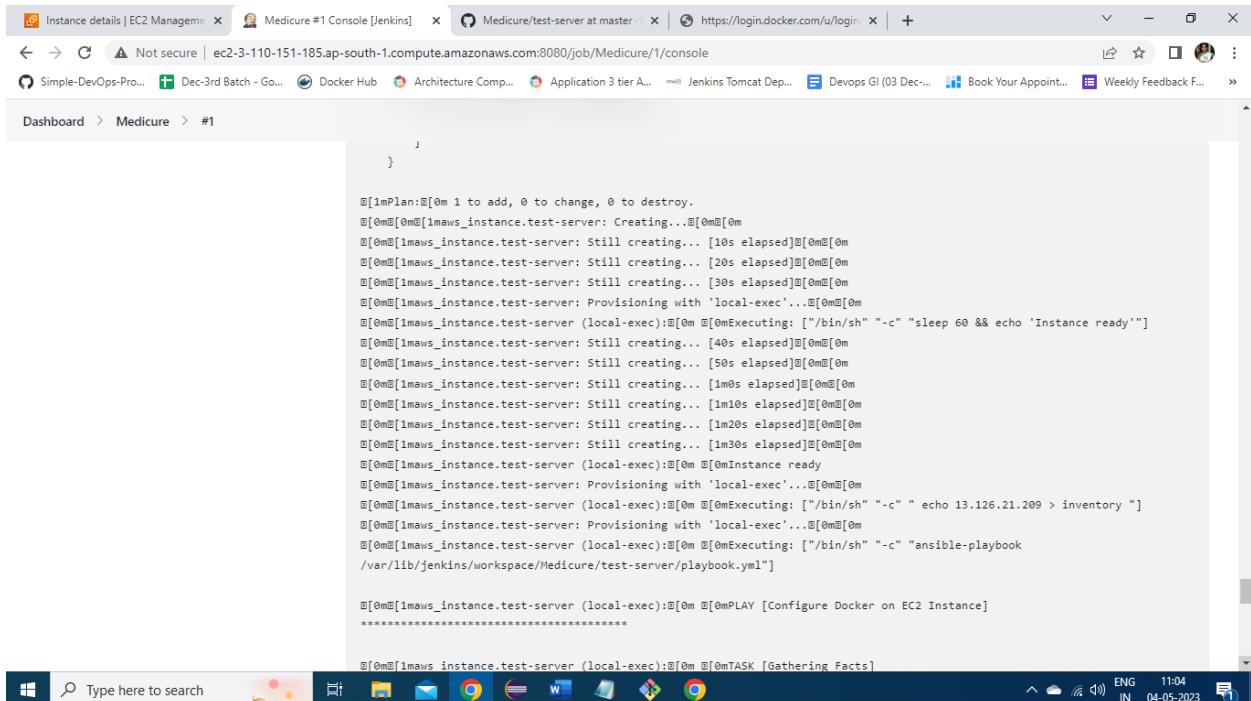
community ALL-HANDS

Terraform is initiated and test -server is created with Public IP: 13.126.21.109

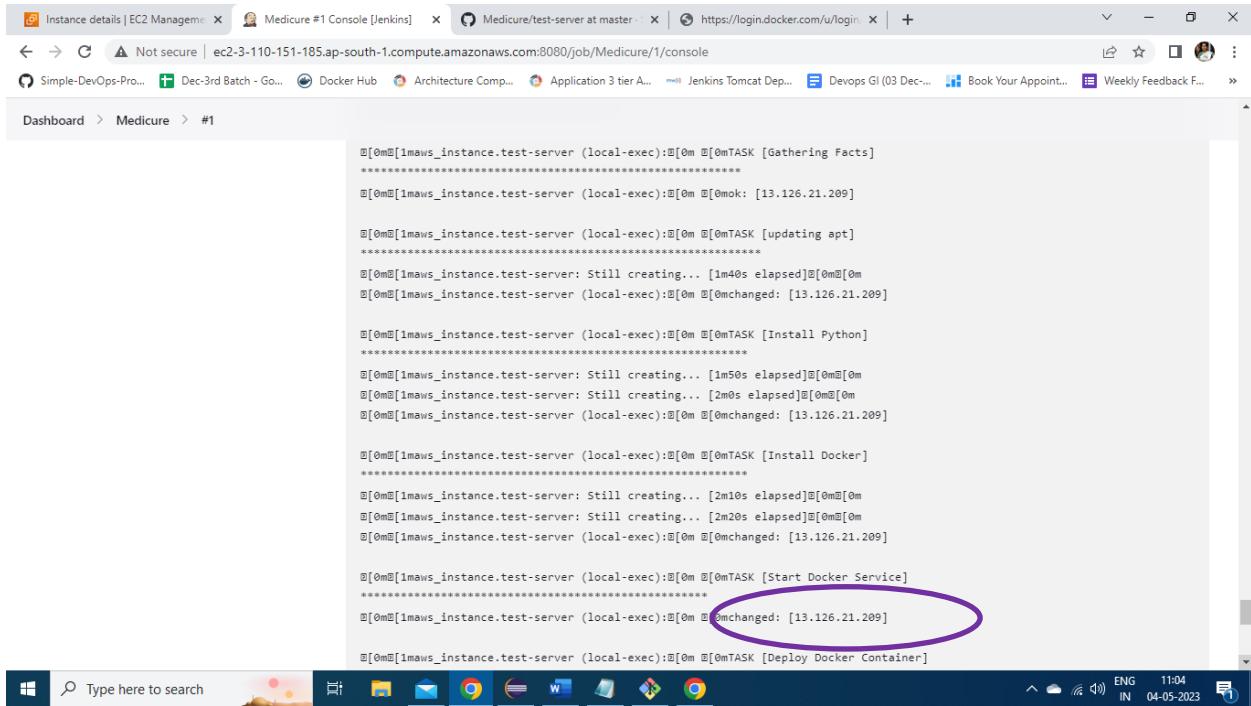


```
Instance details | EC2 Manager Medicure #1 Console [Jenkins] Medicure/test-server at master https://login.docker.com/u/login +  
Not secure | ec2-3-110-151-185.ap-south-1.compute.amazonaws.com:8080/job/Medicure/1/console  
Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec... Book Your Appoint... Weekly Feedback F...  
Dashboard > Medicure > #1  
  
[0m[1mInitializing the backend...[0m  
  
[0m[1mInitializing provider plugins...[0m  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v4.65.0  
  
[0m[1m[32mTerraform has been successfully initialized![0m[32m[0m  
[0m[12m  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.[0m  
[Pipeline] sh  
+ terraform validate  
[32m[1mSuccess![0m The configuration is valid.  
[0m  
[Pipeline] sh  
+ terraform apply --auto-approve  
[0m[1maws_instance.test-server: Refreshing state... [id=i-01ddc6708c2baf5f4][0m  
  
Terraform used the selected providers to generate the following execution  
plan. Resource actions are indicated with the following symbols:  
[32m[0m create[0m  
[0m[1mPlan: 1 to add, 0 to change, 0 to destroy.  
[0m[0m[1maws_instance.test-server: Creating...[0m[0m  
[0m[1maws_instance.test-server: Still creating... [10s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [20s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [30s elapsed][0m[0m  
[0m[1maws_instance.test-server: Provisioning with 'local-exec'...[0m[0m  
[0m[1maws_instance.test-server (local-exec):[0m[0m Executing: ["#!/bin/sh" "-c" "sleep 60 && echo 'Instance ready'"  
[0m[1maws_instance.test-server: Still creating... [40s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [50s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [1m0s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [1m10s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [1m20s elapsed][0m[0m  
[0m[1maws_instance.test-server: Still creating... [1m30s elapsed][0m[0m  
[0m[1maws_instance.test-server (local-exec):[0m[0m Instance ready  
[0m[1maws_instance.test-server: Provisioning with 'local-exec'...[0m[0m  
[0m[1maws_instance.test-server (local-exec):[0m[0m Executing: ["#!/bin/sh" "-c" "echo 13.126.21.209 > inventory"]  
[0m[1maws_instance.test-server: Provisioning with 'local-exec'...[0m[0m  
[0m[1maws_instance.test-server (local-exec):[0m[0m Executing: ["#!/bin/sh" "-c" "ansible-playbook /var/lib/jenkins/workspace/Medicure/test-server/playbook.yml"]  
  
[0m[1maws_instance.test-server (local-exec):[0m[0m PLAY [Configure Docker on EC2 Instance]  
*****  
[0m[1maws_instance.test-server (local-exec):[0m[0m TASK [Gathering Facts]
```

The playbook to configure test-server is started and completed successfully.



```
Instance details | EC2 Manager Medicure #1 Console [Jenkins] Medicure/test-server at master https://login.docker.com/u/login +  
Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec... Book Your Appoint... Weekly Feedback F...  
Dashboard > Medicure > #1  
  
[0m[1maws_instance.test-server (local-exec):[0m[0m PLAY [Configure Docker on EC2 Instance]  
*****  
[0m[1maws_instance.test-server (local-exec):[0m[0m TASK [Gathering Facts]
```



```
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [Gathering Facts]
*****
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mok: [13.126.21.209]

@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [updating apt]
*****
@[0m@[imaws_instance.test-server: Still creating... [1m40s elapsed]@[0m@[0m
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mchanged: [13.126.21.209]

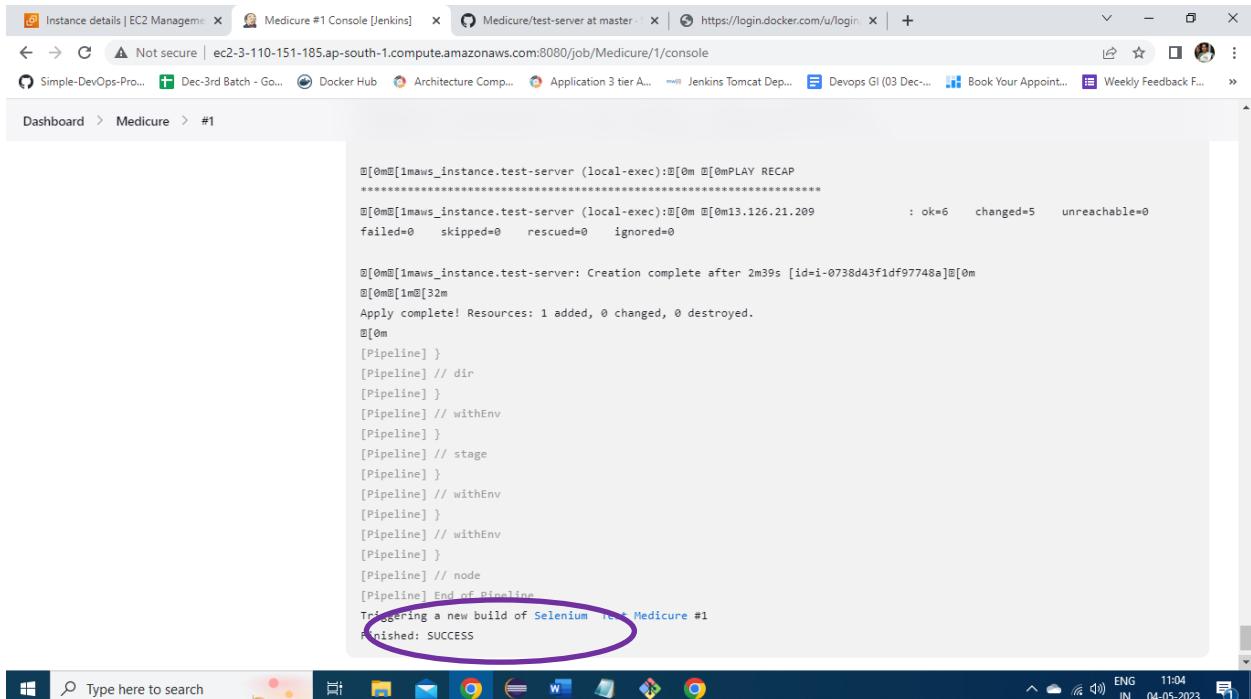
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [Install Python]
*****
@[0m@[imaws_instance.test-server: Still creating... [1m50s elapsed]@[0m@[0m
@[0m@[imaws_instance.test-server: Still creating... [2m0s elapsed]@[0m@[0m
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mchanged: [13.126.21.209]

@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [Install Docker]
*****
@[0m@[imaws_instance.test-server: Still creating... [2m10s elapsed]@[0m@[0m
@[0m@[imaws_instance.test-server: Still creating... [2m20s elapsed]@[0m@[0m
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mchanged: [13.126.21.209]

@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [Start Docker Service]
*****
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mchanged: [13.126.21.209]

@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mTASK [Deploy Docker Container]
```

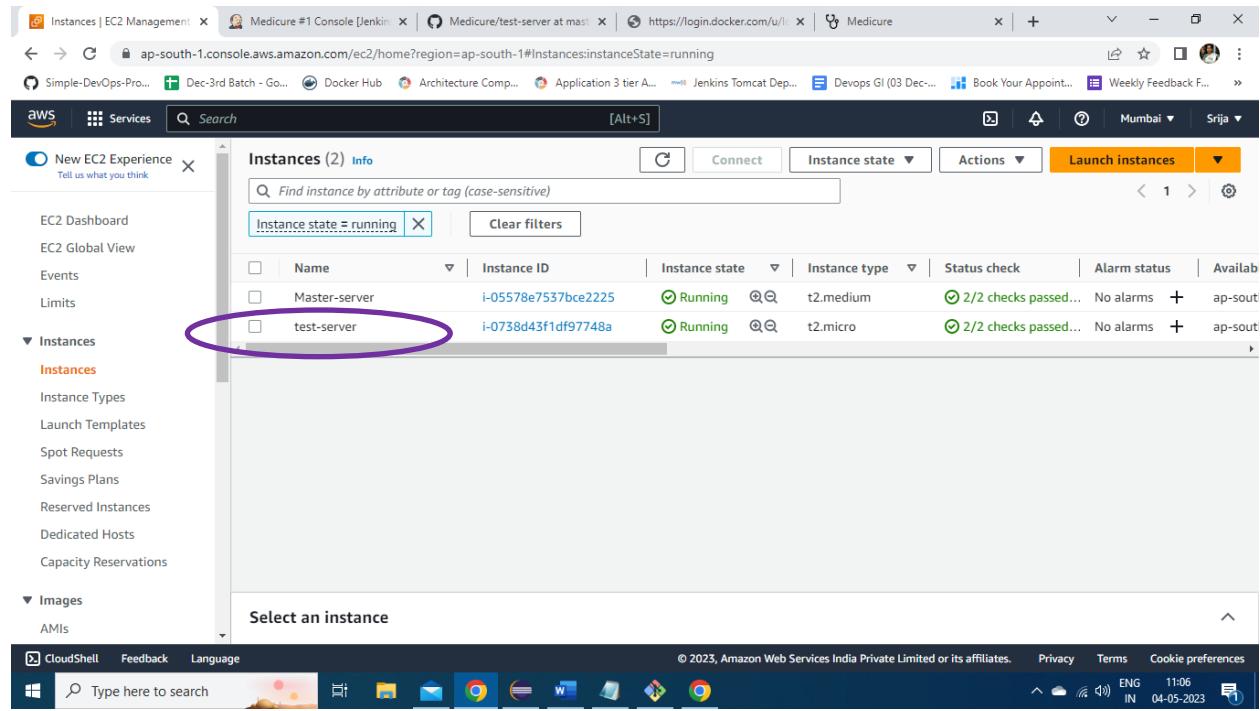
The application is successfully deployed on to the port 8084 and it triggers the selenium test for the server.



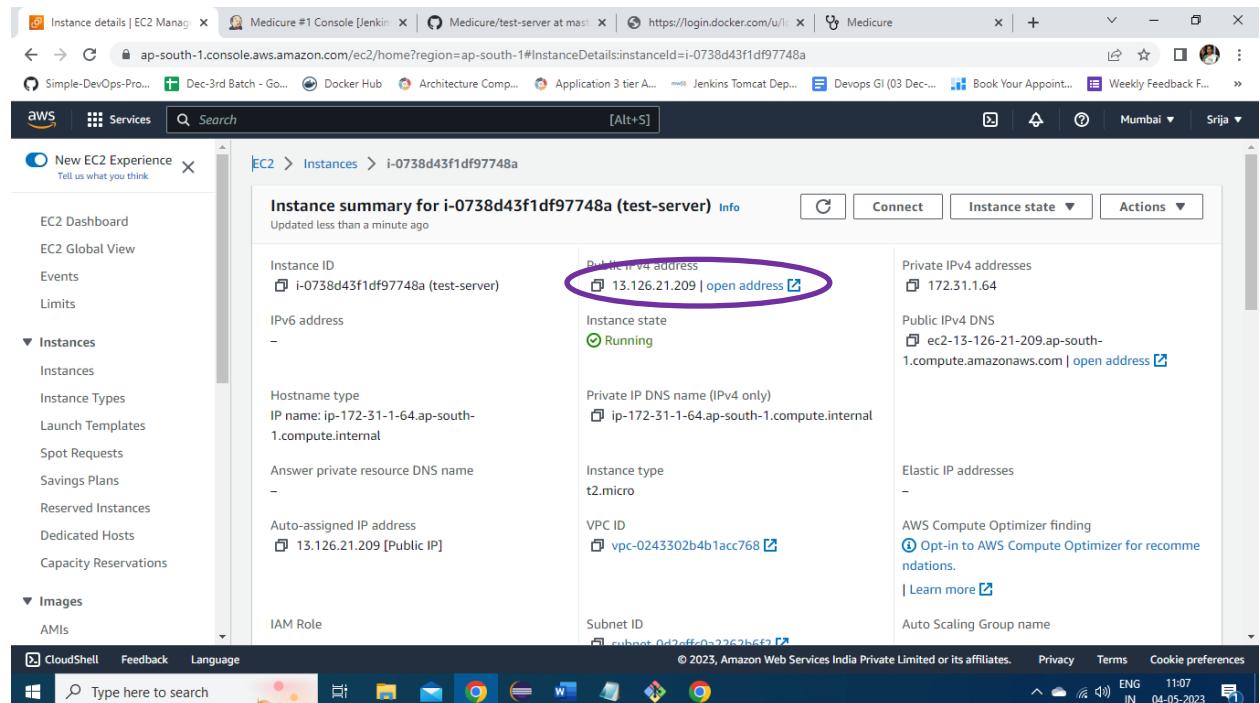
```
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0mPLAY RECAP
*****
@[0m@[imaws_instance.test-server (local-exec):@[0m @[0m13.126.21.209] : ok=6    changed=5    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

@[0m@[imaws_instance.test-server: Creation complete after 2m39s [id=i-0738d43f1df97748a]@[0m
@[0m@[imaws_instance.test-server: Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
@[0m
[Pipeline]
[Pipeline] // dir
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Triggering a new build of Selenium test Medicure #1
Finished: SUCCESS
```

Here is the created test-server using terraform and the IP details of test-server are shown here.

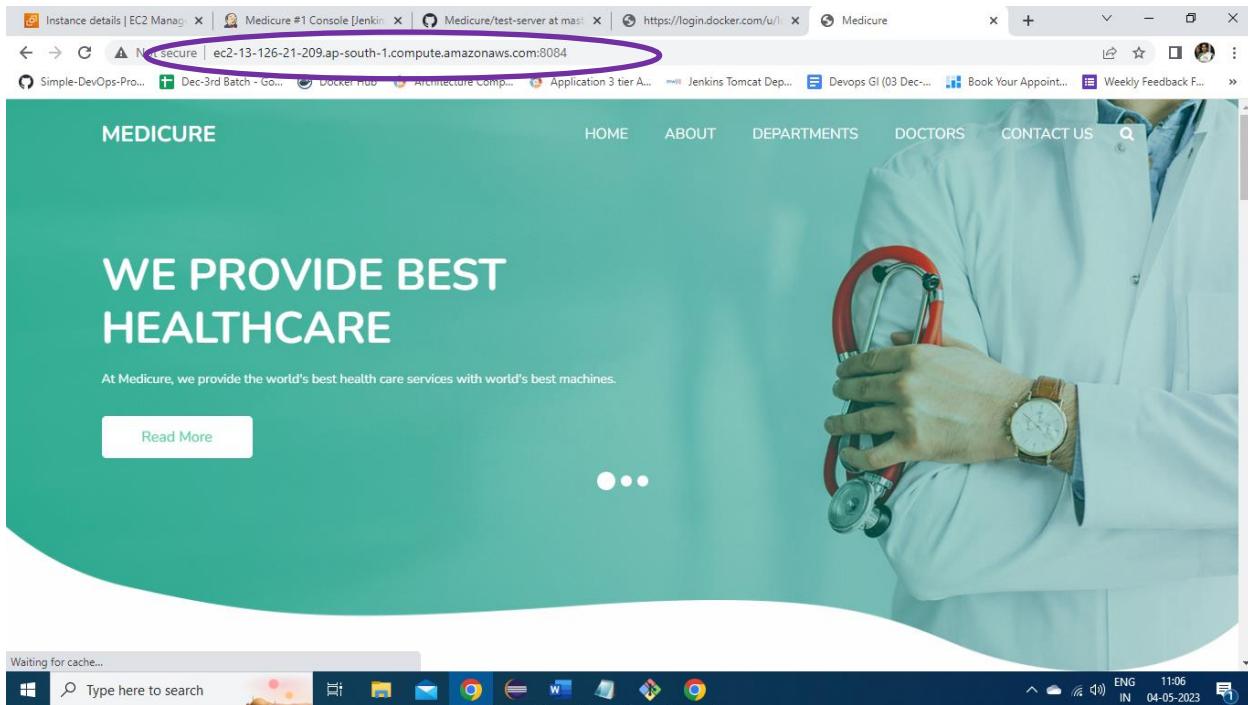


The screenshot shows the AWS EC2 Instances page. The left sidebar shows navigation options like EC2 Dashboard, Instances, and Images. The main area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability zone. Two instances are listed: 'Master-server' (Instance ID: i-05578e7537bce2225) and 'test-server' (Instance ID: i-0738d43f1df97748a). Both instances are in the 'Running' state. The 'test-server' row is highlighted with a purple circle.



The screenshot shows the AWS EC2 Instance Details page for the 'test-server' instance. The left sidebar is identical to the previous screenshot. The main area shows detailed information for the instance. Under the 'Public IPv4 address' section, the value '13.126.21.209' is circled in purple. Other details include the instance ID (i-0738d43f1df97748a), instance state (Running), private IP DNS name (ip-172-31-1-64.ap-south-1.compute.internal), instance type (t2.micro), VPC ID (vpc-0243302b4b1acc768), and subnet ID (subnet-0d2efffc0a2262b6ef).

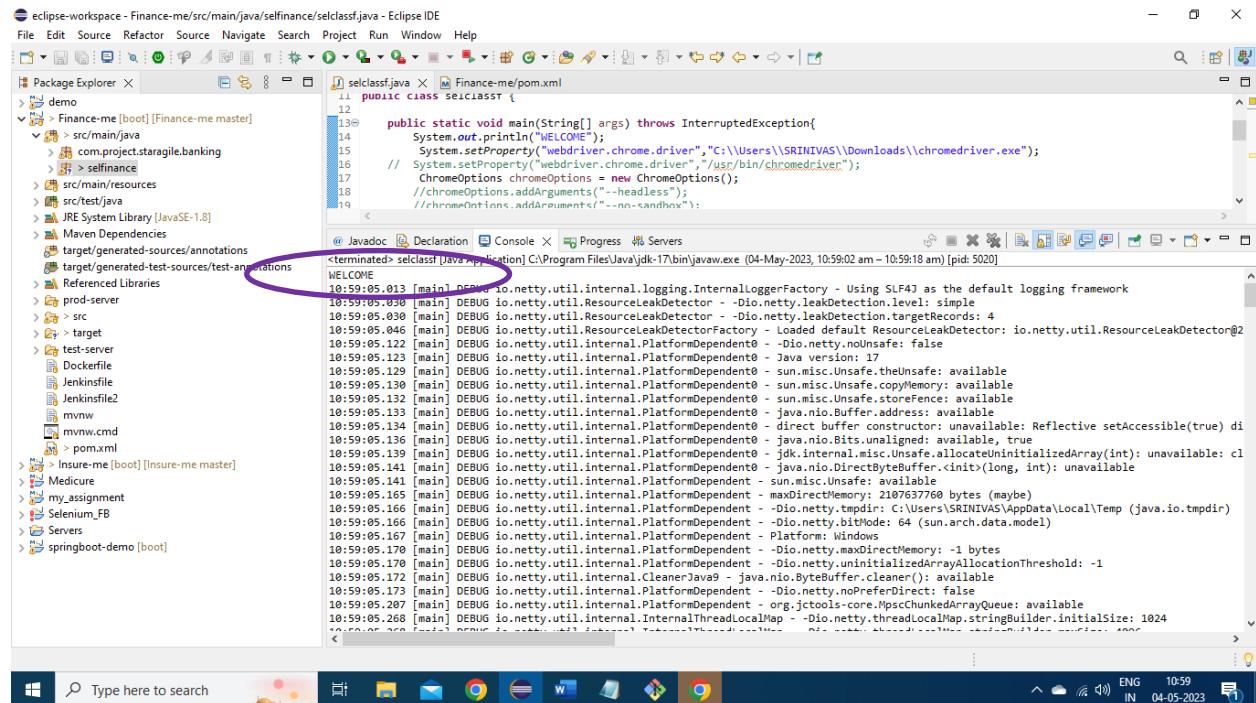
The application on test-server: 13.126.21.109:8084



Here is the selenium script written in eclipse to test the deployed application on test-server contacts page.

```
public class selclass {
    public static void main(String[] args) throws InterruptedException{
        System.out.println("WELCOME");
        System.setProperty("webdriver.chrome.driver","C:\\Users\\SRINIVAS\\Downloads\\chromedriver.exe");
        // System.setProperty("webdriver.chrome.driver","/usr/bin/chromedriver");
        ChromeOptions chromeOptions = new ChromeOptions();
        //chromeOptions.addArguments("--headless");
        //chromeOptions.addArguments("--no-sandbox");
        //chromeOptions.addArguments("--disable-dev-shm-usage");
        WebDriver driver = new ChromeDriver(chromeOptions);
        System.out.println("Welcome To Selenium");
        chromeOptions.addArguments("--remote-allow-origins=*");
        driver.get("http://13.126.21.209:8084/contact.html");
        driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        WebElement nameInput = driver.findElement(By.cssSelector("input[placeholder='Your Name']"));
        nameInput.sendKeys("Srija");
        WebElement Number = driver.findElement(By.cssSelector("input[placeholder='Phone Number']"));
        Number.sendKeys("123456789");
        WebElement mail = driver.findElement(By.cssSelector("input[placeholder='Email']"));
        mail.sendKeys("reachsrija@gmail.com");
        WebElement messageInput = driver.findElement(By.cssSelector("input.message-box[placeholder='Message']"));
        messageInput.sendKeys("Hello, How are you?");
        WebElement sendButton = driver.findElement(By.xpath("//button[contains(text(),'SEND')]"));
        sendButton.click();
        System.out.println("Script Executed Successfully");
    }
}
```

This selenium script is run as java application to test the working of it on windows platform.



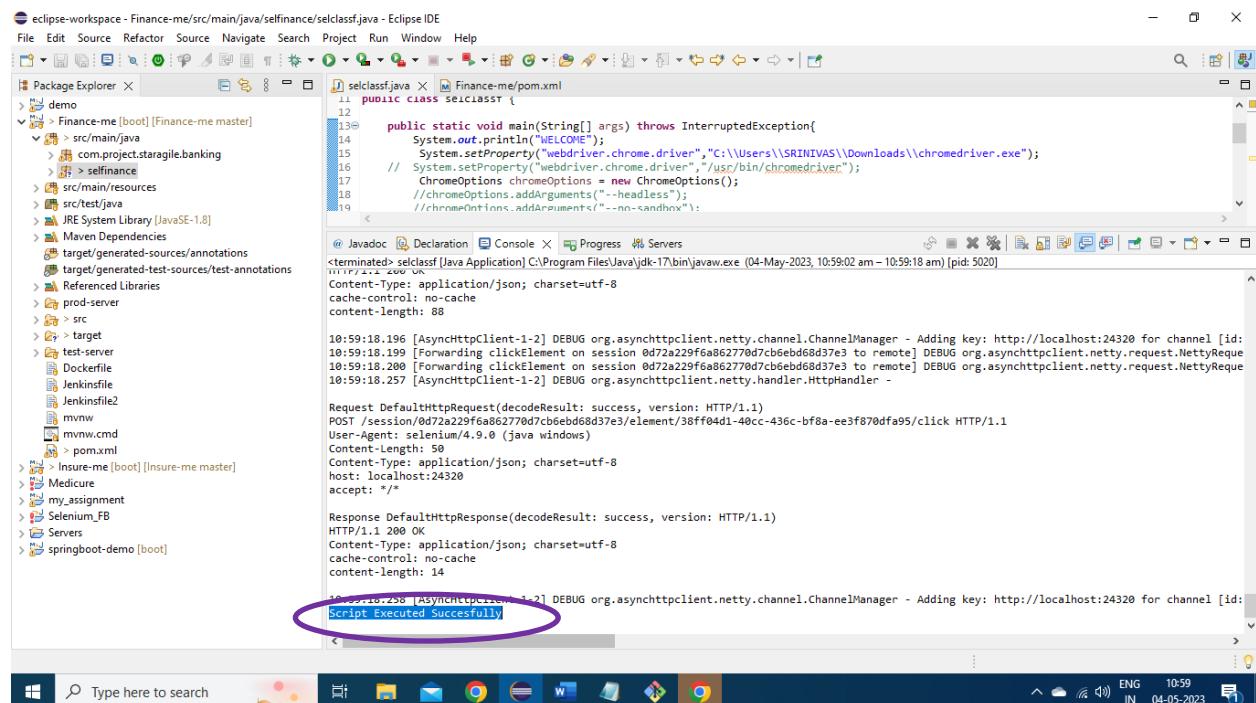
```

eclipse-workspace - Finance-me - src/main/java/selfinance/selclass.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X Finance-me/pom.xml
selclassjava X selclass.java
public class selclass {
    public static void main(String[] args) throws InterruptedException{
        System.out.println("WELCOME");
        System.setProperty("webdriver.chrome.driver","C:\\Users\\SRINIVAS\\Downloads\\chromedriver.exe");
        // System.setProperty("webdriver.chrome.driver","/usr/bin/chromedriver");
        ChromeOptions chromeOptions = new ChromeOptions();
        //chromeOptions.addArguments("--headless");
        //chromeOptions.addArguments("--no-sandbox");
    }
}

@Javadoc Declaration Console X Progress Servers
<terminated> selclass [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (04-May-2023, 10:59:02 am - 10:59:18 am) [pid: 5020]
WELCOME
10:59:05.013 [main] DEBUG io.netty.util.internal.logging.InternalLoggerFactory - Using SLF4J as the default logging framework
10:59:05.020 [main] DEBUG io.netty.util.ResourceLeakDetector - Dio.netty.leakDetection.level: simple
10:59:05.030 [main] DEBUG io.netty.util.ResourceLeakDetector - Dio.netty.leakDetection.targetRecords: 4
10:59:05.046 [main] DEBUG io.netty.util.ResourceLeakDetectorFactory - Loaded default ResourceLeakDetector: io.netty.util.ResourceLeakDetector@2
10:59:05.122 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.noUnsafe: false
10:59:05.123 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Java version: 17
10:59:05.129 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe.theUnsafe: available
10:59:05.130 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe.copyMemory: available
10:59:05.132 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe.storeFence: available
10:59:05.133 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.Buffer.address: available
10:59:05.134 [main] DEBUG io.netty.util.internal.PlatformDependent0 - direct buffer constructor: unavailable: Reflective setAccessible(true) di
10:59:05.136 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.Bits.unaligned: available, true
10:59:05.141 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.DirectByteBuffer.<init>(long, int): unavailable
10:59:05.141 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe: available
10:59:05.165 [main] DEBUG io.netty.util.internal.PlatformDependent0 - maxDirectMemory: 2107637760 bytes (maybe)
10:59:05.166 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.tmpdir: C:\Users\SRINIVAS\AppData\Local\Temp (java.io.tmpdir)
10:59:05.166 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.bitMode: 64 (sun.arch.data.model)
10:59:05.167 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Platform: Windows
10:59:05.170 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.maxDirectMemory: -1 bytes
10:59:05.170 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.uninitializedMemoryAllocationThreshold: -1
10:59:05.172 [main] DEBUG io.netty.util.internal.Cleaner@98 - java.nio.ByteBuffer.Cleaner: available
10:59:05.173 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Dio.netty.noPreferDirect: false
10:59:05.207 [main] DEBUG io.netty.util.internal.PlatformDependent0 - org.jctools-core.MpscChunkedArrayQueue: available
10:59:05.268 [main] DEBUG io.netty.util.internal.InternalThreadLocalMap - Dio.netty.threadLocalMap.StringBuilder.initialSize: 1024

```

Here is it is successfully started and it automates the application and enters the basic details of a customer willing to contact the Medicure.



```

eclipse-workspace - Finance-me - src/main/java/selfinance/selclass.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X Finance-me/pom.xml
selclassjava X selclass.java
public class selclass {
    public static void main(String[] args) throws InterruptedException{
        System.out.println("WELCOME");
        System.setProperty("webdriver.chrome.driver","C:\\Users\\SRINIVAS\\Downloads\\chromedriver.exe");
        // System.setProperty("webdriver.chrome.driver","/usr/bin/chromedriver");
        ChromeOptions chromeOptions = new ChromeOptions();
        //chromeOptions.addArguments("--headless");
        //chromeOptions.addArguments("--no-sandbox");
    }
}

@Javadoc Declaration Console X Progress Servers
<terminated> selclass [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (04-May-2023, 10:59:02 am - 10:59:18 am) [pid: 5020]
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
cache-control: no-cache
content-length: 88

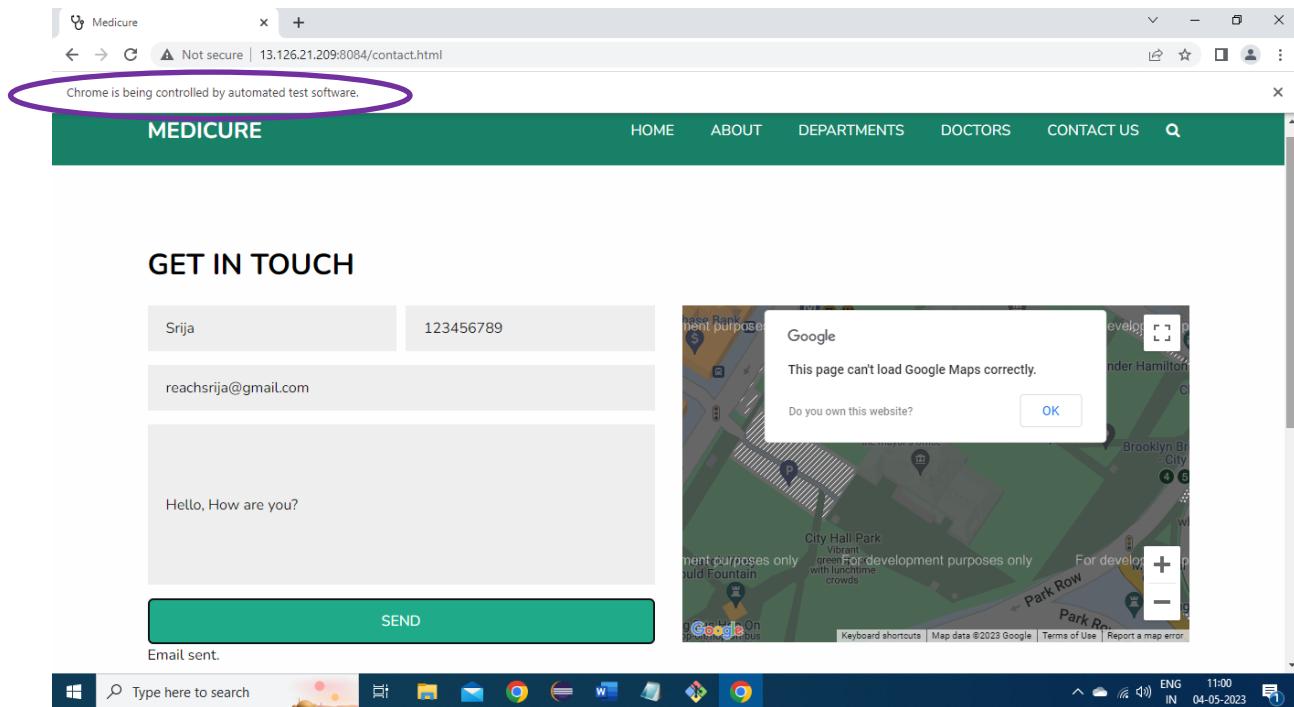
10:59:18.196 [AsyncHttpClient-1-2] DEBUG org.asynchttpclient.netty.channel.ChannelManager - Adding key: http://localhost:24320 for channel [id: 0d72a229f6a862770d7cb6ebd68d37e3] to remote] DEBUG org.asynchttpclient.netty.request.NettyRequest
10:59:18.199 [Forwarding clickElement on session 0d72a229f6a862770d7cb6ebd68d37e3 to remote] DEBUG org.asynchttpclient.netty.request.NettyRequest
10:59:18.200 [Forwarding clickElement on session 0d72a229f6a862770d7cb6ebd68d37e3 to remote] DEBUG org.asynchttpclient.netty.request.NettyRequest
10:59:18.257 [AsyncHttpClient-1-2] DEBUG org.asynchttpclient.netty.handler.HttpHandler -
Request.DefaultHttpRequest(decodeResult: success, version: HTTP/1.1)
POST /session/0d72a229f6a862770d7cb6ebd68d37e3/element/38ff04d1-40cc-436c-bf8a-ee3f870dfa95/click HTTP/1.1
User-Agent: selenium/4.9.0 (java windows)
Content-Length: 58
Content-Type: application/json; charset=utf-8
host: localhost:24320
accept: /*

Response.DefaultHttpResponse(decodeResult: success, version: HTTP/1.1)
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
cache-control: no-cache
content-length: 14

10:59:18.258 [AsyncHttpClient-1-2] DEBUG org.asynchttpclient.netty.channel.ChannelManager - Adding key: http://localhost:24320 for channel [id: 0d72a229f6a862770d7cb6ebd68d37e3] to remote] DEBUG org.asynchttpclient.netty.request.NettyRequest
Script Executed Successfully

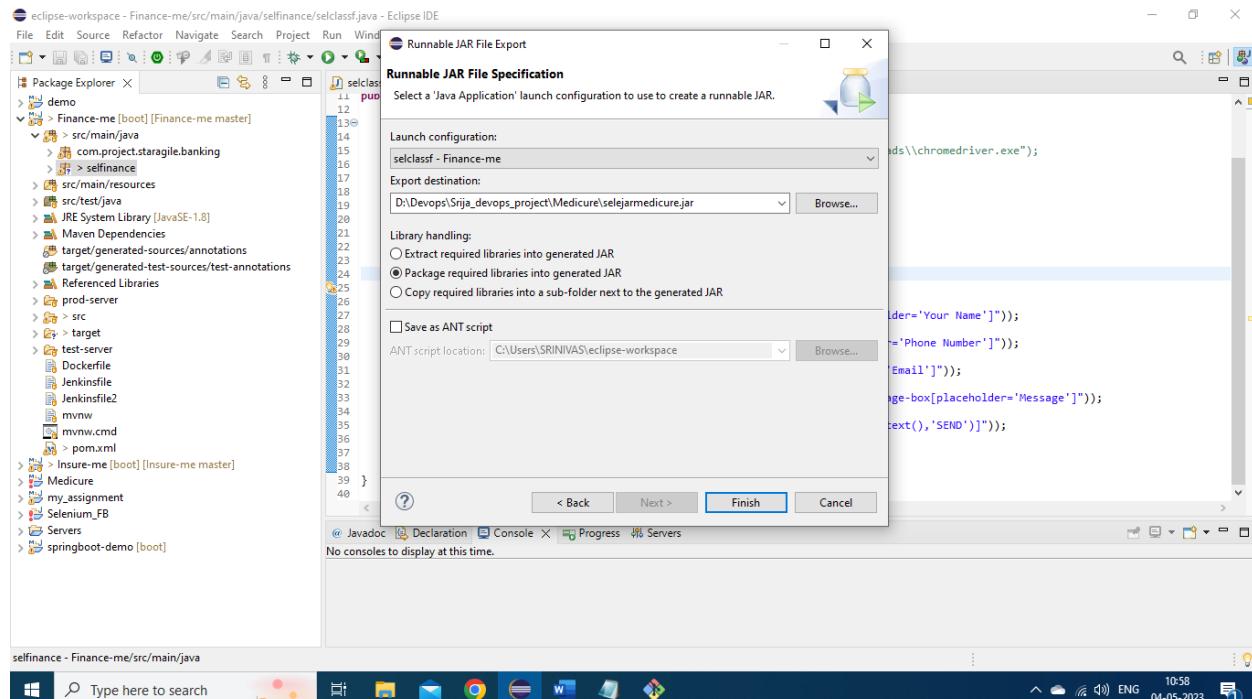
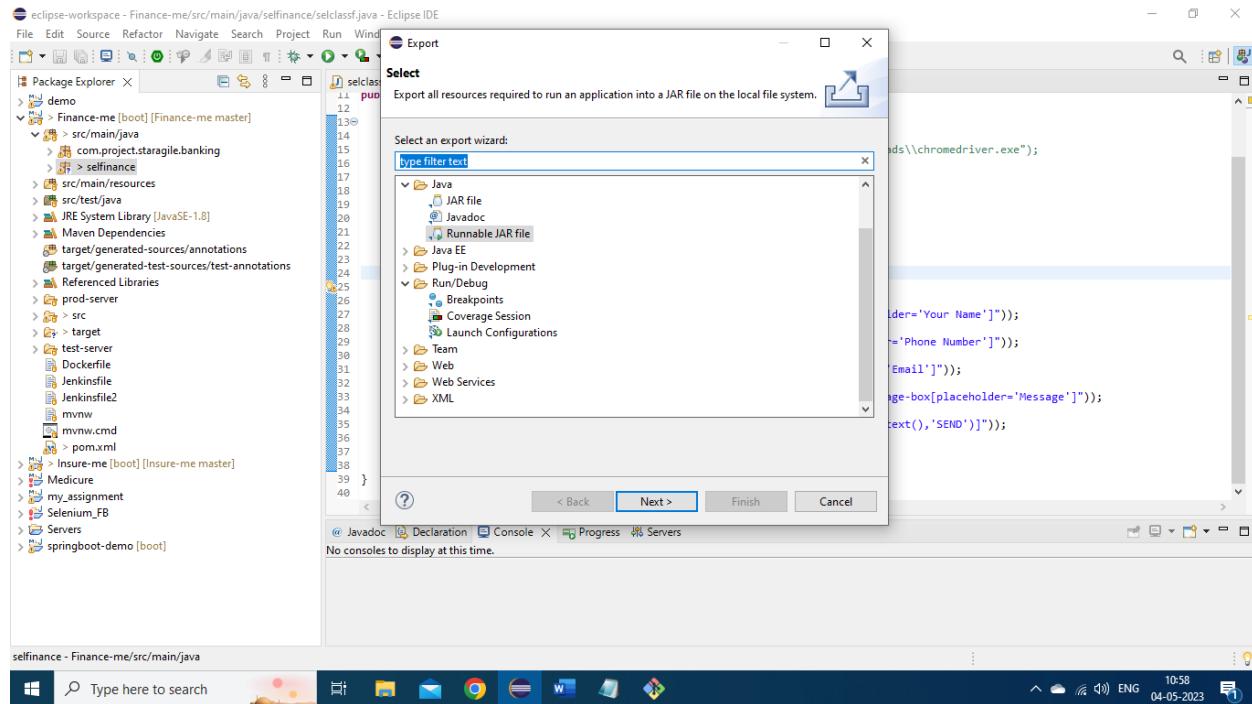
```

Here is the automated application run though selenium.

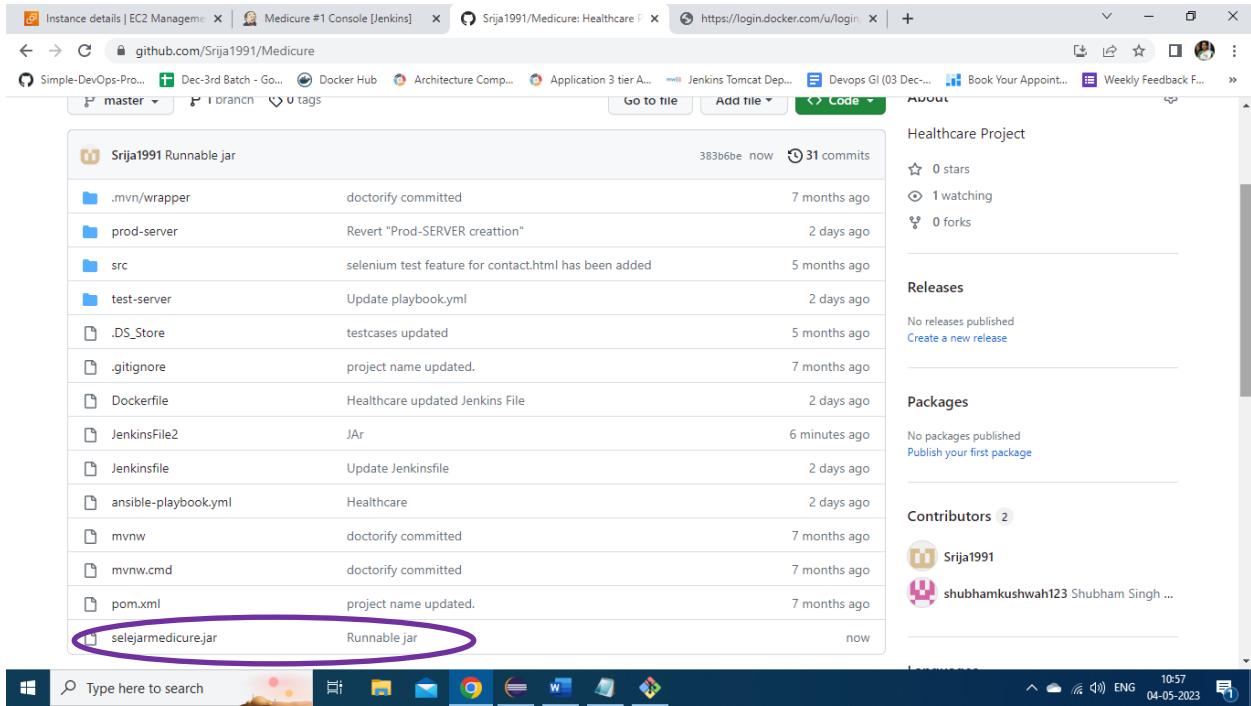


Now the selenium script is modified to run on Linux platform and exported as runnable jar to git hub .

A screenshot of the Eclipse IDE interface. On the left is the "Package Explorer" view showing project structure with multiple modules like "Finance-me", "Insure-me", and "springboot-demo". The central area is a code editor with the file "selclass.java" open. The code contains a main method that sets up a ChromeDriver instance, navigates to a URL, interacts with an input field, sends keys, and prints "Script Executed Successfully". The bottom of the screen shows the Eclipse toolbar and the Windows taskbar with the date/time "04-05-2023 10:57".

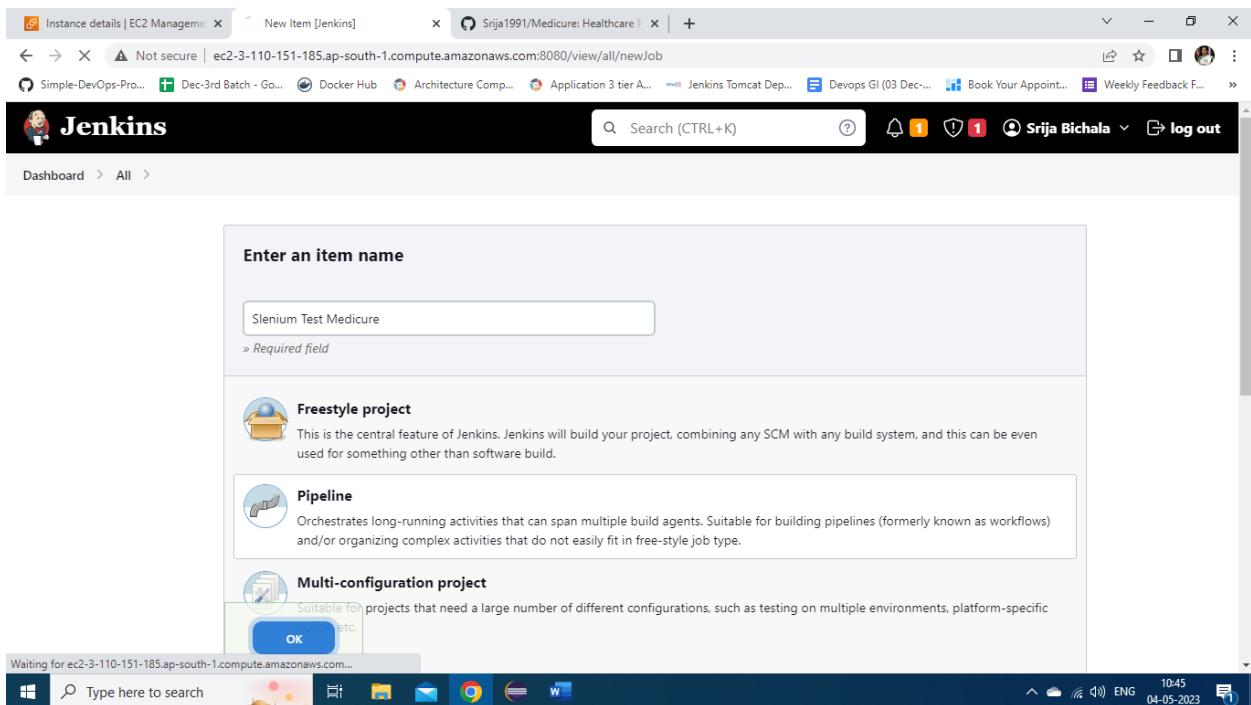


The exported jar is pushed onto GitHub to execute the selenium script on EC2 instance.

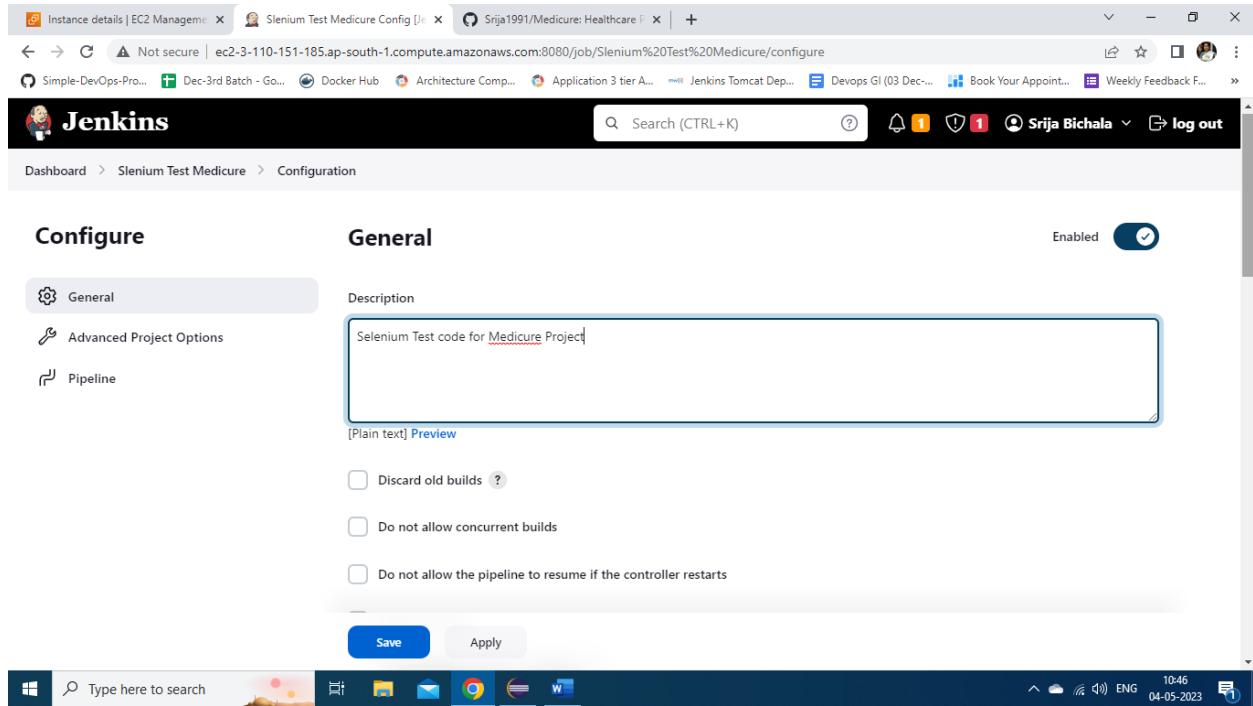


A screenshot of a Microsoft Edge browser window. The address bar shows 'github.com/Srija1991/Medicure'. The page displays a GitHub repository named 'Medicure'. The 'src' directory contains a file named 'selejarmedicure.jar' which is highlighted with a purple oval. Other files listed include '.mvn/wrapper', 'prod-server', 'test-server', '.DS_Store', '.gitignore', 'Dockerfile', 'Jenkinsfile', 'ansible-playbook.yml', 'mvnw', 'mvnw.cmd', and 'pom.xml'. The right sidebar shows project details like 'Healthcare Project', 'Releases', 'Packages', and 'Contributors'.

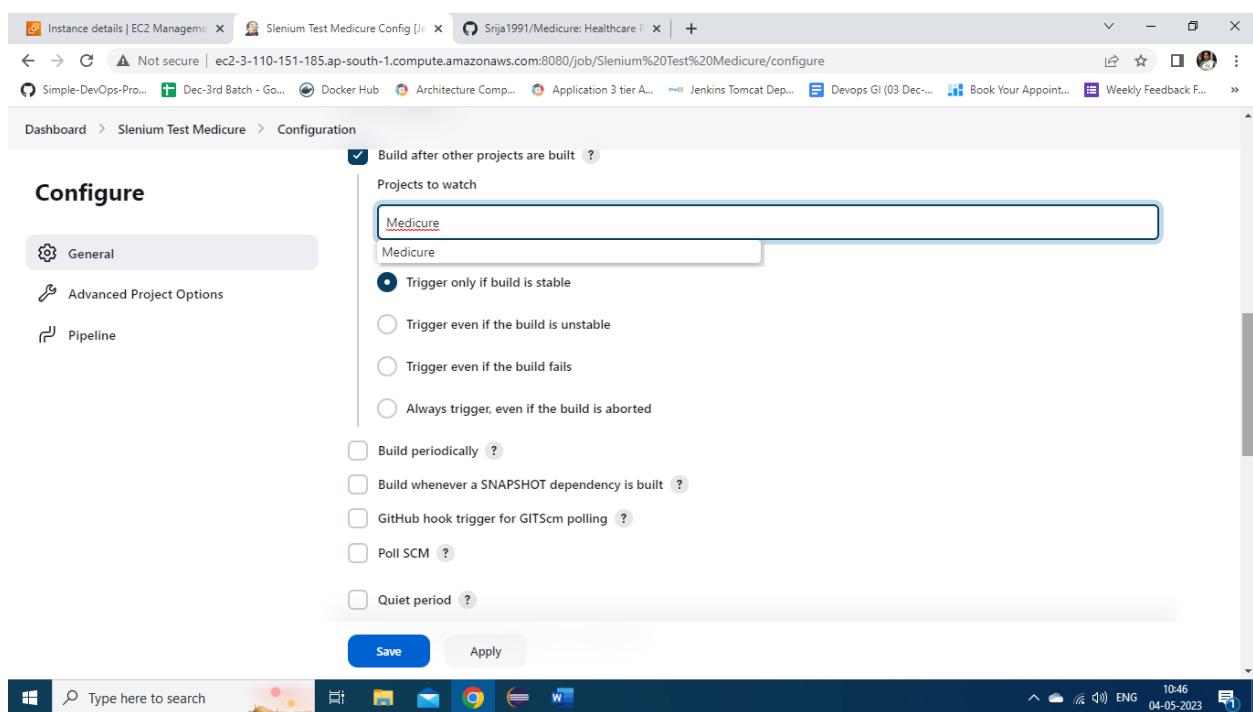
A new job named Selenium Test Medicure is created to run the selenium script after the successful execution of Medicure job.



A screenshot of a Microsoft Edge browser window showing the Jenkins dashboard. A new job is being created, with the item name field set to 'Selenium Test Medicure'. The Jenkins interface shows options for 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'OK' button is visible at the bottom of the form. The status bar at the bottom indicates 'Waiting for ec2-3-110-151-185.ap-south-1.compute.amazonaws.com...'.



The screenshot shows the Jenkins General configuration page for a project named "Selenium Test Medicure". The "Enabled" toggle switch is turned on. The "Description" field contains the text "Selenium Test code for Medicure Project". Under the "Advanced Project Options" section, there are three checkboxes: "Discard old builds", "Do not allow concurrent builds", and "Do not allow the pipeline to resume if the controller restarts". At the bottom are "Save" and "Apply" buttons.



The screenshot shows the Jenkins "Build after other projects are built" configuration page. The "Projects to watch" dropdown is set to "Medicure". The "Trigger only if build is stable" radio button is selected. Other options include "Trigger even if the build is unstable", "Trigger even if the build fails", and "Always trigger, even if the build is aborted". There are also checkboxes for "Build periodically", "Build whenever a SNAPSHOT dependency is built", "GitHub hook trigger for GITScm polling", "Poll SCM", and "Quiet period". At the bottom are "Save" and "Apply" buttons.

Here is the pipeline script to execute the jar on EC2 instance.

The screenshot shows the AWS Lambda Pipeline configuration screen. The pipeline is named "Pipeline". The "Definition" section is set to "Pipeline script". The script content is as follows:

```
1 node{
2   stage('Git Checkout'){
3     git "https://github.com/Srija1991/Medicure.git"
4     echo 'Git checkout successful'
5   }
6   stage('Run Jar File'){
7     sh 'java -jar selejarmedicure.jar'
8   }
9 }
```

The status message at the bottom says "The script is already approved". There are "Save" and "Apply" buttons at the bottom.

The console output for this job is shown here. It shows the chrome is started successfully and script is executed successfully.

The screenshot shows the Jenkins Console Output page for build #1 of the "Selenium Test Medicare" job. The title is "Console Output". The log output is as follows:

```
Started by user Srija Bichala
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Selenium Test Medicare
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Checkout)
[Pipeline] git
The recommended git tool is: git
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Selenium Test Medicare/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Srija1991/Medicure.git # timeout=10
Fetching upstream changes from https://github.com/Srija1991/Medicure.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Srija1991/Medicure.git +refs/heads/*:refs/remotes/origin/* # timeout=10
```

Instance details | EC2 Manager | Selenium Test Medicure #1 | Sreja1991/Medicure: HealthCare | Docker Hub | Medicare

Not secure | ec2-3-110-151-185.ap-south-1.compute.amazonaws.com:8080/job/Selenium%20Test%20Medicure/1/console

Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec-... Book Your Appoint... Weekly Feedback F...

Dashboard > Selenium Test Medicure > #1

```
[Pipeline] stage
[Pipeline] { (Run Jar File)
[Pipeline] sh
  +--> /opt/jdk1.8.0_261/bin/java -jar /opt/telefarmedicure.jar
  HELCOM
05:43:57.996 [main] DEBUG io.netty.util.internal.logging.InternalLoggerFactory - Using SLF4J as the default logging framework
05:43:57.996 [main] DEBUG io.netty.util.ResourceLeakDetector - -Dio.netty.leakDetection.level: simple
05:43:57.996 [main] DEBUG io.netty.util.ResourceLeakDetector - -Dio.netty.leakDetection.targetRecords: 4
05:43:58.000 [main] DEBUG io.netty.util.ResourceLeakDetectorFactory - Loaded default ResourceLeakDetector: io.netty.util.ResourceLeakDetector@7fa98a66
05:43:58.036 [main] DEBUG io.netty.util.internal.PlatformDependent0 - -Dio.netty.noUnsafe: false
05:43:58.036 [main] DEBUG io.netty.util.internal.PlatformDependent0 - Java version: 17
05:43:58.038 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe: available
05:43:58.039 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe.copyMemory: available
05:43:58.039 [main] DEBUG io.netty.util.internal.PlatformDependent0 - sun.misc.Unsafe.storeFence: available
05:43:58.040 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.Buffer.address: available
05:43:58.044 [main] DEBUG io.netty.util.internal.PlatformDependent0 - direct buffer constructor: unavailable: Reflective setAccessible(true) disabled
05:43:58.044 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.Bits.unaligned: available, true
05:43:58.042 [main] DEBUG io.netty.util.internal.PlatformDependent0 - jdk.internal.misc.Unsafe.allocateUninitializedArray(int): unavailable: class io.netty.util.internal.PlatformDependent0$7 cannot access class jdk.internal.misc.Unsafe (in module java.base) because module java.base does not export jdk.internal.misc to unnamed module @2890c451
05:43:58.043 [main] DEBUG io.netty.util.internal.PlatformDependent0 - java.nio.DirectByteBuffer.<init>(long, int): unavailable
```

Type here to search ENG 11:14 IN 04-05-2023

Instance details | EC2 Manager | Selenium Test Medicure #1 | Sreja1991/Medicure: HealthCare | Docker Hub | Medicare

Not secure | ec2-3-110-151-185.ap-south-1.compute.amazonaws.com:8080/job/Selenium%20Test%20Medicure/1/console

Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec-... Book Your Appoint... Weekly Feedback F...

Dashboard > Selenium Test Medicure > #1

```
05:43:58.408 [main] DEBUG io.netty.buffer.ByteBufUtil - -Dio.netty.threadLocalDirectBufferSize: 0
05:43:58.408 [main] DEBUG io.netty.buffer.ByteBufUtil - -Dio.netty.maxThreadLocalCharBufferSize: 16384
Starting ChromeDriver 112.0.5615.49 (bd2a7cb881c11e8cfe3078709382934e3916914-refs/branch-heads/5615@(#936)) on port 10156
Only local connections are allowed.
please see https://chromium.googlesource.com/chromium/org.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
05:43:58.784 [Forwarding newSession on session null to remote] DEBUG io.netty.channel.DefaultChannelId - -
Dio.netty.processId: 3409 (auto-detected)
05:43:58.786 [Forwarding newSession on session null to remote] DEBUG io.netty.util.NetUtil - -Djava.net.preferIPv4Stack: false
05:43:58.786 [Forwarding newSession on session null to remote] DEBUG io.netty.util.NetUtil - -Djava.net.preferIPv6Addresses: false
05:43:58.787 [Forwarding newSession on session null to remote] DEBUG io.netty.util.NetUtilInitializations - Loopback interface: lo (lo, 0:0:0:0:0:1%lo)
05:43:58.788 [Forwarding newSession on session null to remote] DEBUG io.netty.util.NetUtil - /proc/sys/net/core/somaxconn: 4096
05:43:58.789 [Forwarding newSession on session null to remote] DEBUG io.netty.channel.DefaultChannelId - -
Dio.netty.machineId: 02:42:20:ff:fe:b5:c4:b5 (auto-detected)
05:43:58.842 [AsyncHttpClient-1-2] DEBUG io.netty.buffer.AbstractByteBuf - -Dio.netty.buffer.checkAccessible: true
05:43:58.842 [AsyncHttpClient-1-2] DEBUG io.netty.buffer.AbstractByteBuf - -Dio.netty.buffer.checkBounds: true
05:43:58.842 [AsyncHttpClient-1-2] DEBUG io.netty.util.ResourceLeakDetectorFactory - Loaded default ResourceLeakDetector: io.netty.util.ResourceLeakDetector@37bf5376
05:43:58.854 [AsyncHttpClient-1-2] DEBUG org.asynchttpclient.netty.channel.NettyConnectListener - Using new Channel '[id: 0x00ea1271, L:/127.0.0.1:54194 - R:localhost/127.0.0.1:10156]' for 'POST' to '/session'
05:43:58.880 [AsyncHttpClient-1-2] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.maxCapacityPerThread: 4096
05:43:58.880 [AsyncHttpClient-1-2] DEBUG io.netty.util.Recycler - -Dio.netty.recycler.ratio: 8
```

Type here to search ENG 11:14 IN 04-05-2023

```

Request DefaultHttpRequest(decodeResult: success, version: HTTP/1.1)
POST /session/ad27cd68f241e5b8c3de5e583422b1e/element/eb782295-57cb-4b4d-b5d4-e4c4af15470b/click HTTP/1.1
User-Agent: selenium/4.9.0 (java unix)
Content-Length: 50
Content-Type: application/json; charset=utf-8
host: localhost:10156
accept: */*

Response DefaultHttpResponse(decodeResult: success, version: HTTP/1.1)
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
cache-control: no-cache
content-length: 14

05:44:02.897 [AsyncHttpClient-1-2] DEBUG org.asynchttpclient.netty.channel.ChannelManager - Adding key:
http://localhost:10156 for channel [id: 0x0ea1271, L:/127.0.0.1:54194 - R:localhost/127.0.0.1:10156]
Script Executed Successfully
[Pipeline] //
[Pipeline] // stage
[Pipeline] //
[Pipeline] // node
[Pipeline] // stage
[Pipeline] Finished: SUCCESS

```

After successful execution of Selenium script, a new job named *Prod_server_Terraform* is created to create Production server using terraform.

Enter an item name

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

Suitable for etc.
OK
Folder

The screenshot shows the Jenkins configuration interface for a job named "Prod_Server_Terraform". The "General" tab is selected. The "Description" field contains the text "Creation of Production server using terraform". Under the "Advanced" section, three checkboxes are present: "Discard old builds", "Do not allow concurrent builds", and "Do not allow the pipeline to resume if the controller restarts". At the bottom are "Save" and "Apply" buttons.

The pipeline script for this is provided through *JenkinsFile2* on the github.

The screenshot shows the Jenkins configuration interface for the same job. The "Pipeline" tab is selected. The "Definition" dropdown is set to "Pipeline script from SCM". The "SCM" dropdown is set to "Git". The "Repositories" section shows a single repository with the URL "https://github.com/Srija1991/Medicure.git". The "Credentials" dropdown is set to "- none -". At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Pipeline configuration interface. On the left, there are tabs for General, Advanced Project Options, and Pipeline, with Pipeline selected. In the center, there's a 'Repository browser' dropdown set to '(Auto)'. Below it is an 'Additional Behaviours' section with a 'Add' button. Under 'Script Path', the value 'JenkinsFile2' is entered and highlighted with a purple oval. There's also a checked checkbox for 'Lightweight checkout'. At the bottom are 'Save' and 'Apply' buttons.

Here is the script to create new instance of t2.medium.

```

1 pipeline {
2   agent any
3
4   tools {
5     maven 'maven'
6     terraform 'terraform'
7   }
8   stages{
9     stage ('Configure Prod-server with Terraform, Ansible and then Deploying'){
10       steps {
11         dir('prod-server'){
12           sh 'chmod 600 apr26.pem'
13           sh 'terraform init'
14           sh 'terraform validate'
15           sh 'terraform apply --auto-approve'
16         }
17       }
18     }
19   }
20 }
21
22

```

```

1 resource "aws_instance" "prod-server" {
2   ami           = "ami-02eb74a783e7e9317"
3   instance_type = "t2.medium"
4   key_name      = "apr26"
5   vpc_security_group_ids = ["sg-0c69f259b0ea97dc0"]
6   connection {
7     type    = "ssh"
8     user    = "ubuntu"
9     private_key = file("./apr26.pem")
10    host    = self.public_ip
11  }
12  provider "remote-exec" {
13    inline = [ "echo 'wait to start instance' "]
14  }
15  tags = [
16    { Name = "prod-server" }
17  ]
18}
19

```



Now the job is executed and it is success creating a t2.medium instance which is manually configured for Kubernetes setup, since the project need to deploy using Kubernetes-cluster upon successful testing.

```

Started by user Srija Bichala
Obtained JenkinsFile from git https://github.com/Srija1991/Medicure.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Prod_Server_Terraform
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: git
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Prod_Server_Terraform/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Srija1991/Medicure.git # timeout=10
Fetching upstream changes from https://github.com/Srija1991/Medicure.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Srija1991/Medicure.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 3ddfed626128ee0d001e7e5d07a55ccb575e0fb (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10

```



```
[Pipeline] sh
+ chmod 600 apr26.pem
[Pipeline] sh
+ terraform init

[0m@1mInitializing the backend...@0m

[0m@1mInitializing provider plugins...@0m
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.65.0

@0m@1m[32mTerraform has been successfully initialized!@0m@32m@0m
@0m@32m
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.@0m
[Pipeline] sh
+ terraform validate
@32m@1mSuccess!@0m The configuration is valid.
@0m
[Pipeline] sh
+ terraform apply --auto-approve
```

```
}
```

```
@1mPlan:@0m 1 to add, 0 to change, 1 to destroy.
@0m@1maws_instance.prod-server: Destroying...
[id=i-0662e7ebcd607f888]@0m@0m
@0m@1maws_instance.prod-server: Still destroying... [id=i-0662e7ebcd607f888, 10s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Still destroying... [id=i-0662e7ebcd607f888, 20s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Still destroying... [id=i-0662e7ebcd607f888, 30s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Destruction complete after 40s@0m
@0m@1maws_instance.prod-server: Creating...
@0m@1maws_instance.prod-server: Still creating... [10s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Still creating... [20s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Still creating... [30s elapsed]@0m@0m
@0m@1maws_instance.prod-server: Provisioning with 'remote-exec'...
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Connecting to remote host via SSH...
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Host: 3.108.66.191
@0m@1maws_instance.prod-server (remote-exec):@0m@0m User: ubuntu
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Password: false
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Private key: true
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Certificate: false
@0m@1maws_instance.prod-server (remote-exec):@0m@0m SSH Agent: false
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Checking Host Key: false
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Target Platform: unix
@0m@1maws_instance.prod-server (remote-exec):@0m@0m @0m@0m Connecting to remote host via SSH...
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Host: 3.108.66.191
@0m@1maws_instance.prod-server (remote-exec):@0m@0m User: ubuntu
@0m@1maws_instance.prod-server (remote-exec):@0m@0m Password: false
```

```

[0m@1aws_instance.prod-server (remote-exec): [0m [0m SSH Agent: false
[0m@1aws_instance.prod-server (remote-exec): [0m [0m Checking Host Key: false
[0m@1aws_instance.prod-server (remote-exec): [0m [0m Target Platform: unix
[0m@1aws_instance.prod-server (remote-exec): [0m [0m Connected!
[0m@1aws_instance.prod-server: Still creating... [48s elapsed] [0m [0mwait to start instance
[0m@1aws_instance.prod-server: Creation complete after 40s [id=i-062448df6fee5fd37] [0m
[0m@1aws_instance.prod-server: Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
[0m
[Pipeline]
[Pipeline] // dir
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Here is the prod-server created and its specifications are shown here.

Name	Instance ID	Instance State	Instance Type	Status Check	Alarm Status	Availability Zone
Master-server	i-05578e7537bce2225	Running	t2.medium	2/2 checks passed...	No alarms	ap-south-1
test-server	i-0738d43f1df97748a	Running	t2.micro	2/2 checks passed...	No alarms	ap-south-1
prod-server	i-062448df6fee5fd37	Running	t2.medium	2/2 checks passed...	No alarms	ap-south-1

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Limits, Instances (with sub-options like Instances, Instance Types, Launch Templates, etc.), and Images (with sub-options like AMIs). The main area displays the instance summary for 'i-062448df6fee5fd37 (prod-server)'. Key details shown include:

- Instance ID: i-062448df6fee5fd37 (prod-server)
- Public IPv4 address: 3.108.66.191 (highlighted with a purple circle)
- Instance state: Running
- Private IPv4 addresses: 172.31.39.255
- Public IPv4 DNS: ec2-3-108-66-191.ap-south-1.compute.amazonaws.com
- Private IP DNS name (IPv4 only): ip-172-31-39-255.ap-south-1.compute.internal
- Instance type: t2.medium
- VPC ID: vpc-0243302b4b1acc768
- Subnet ID: subnet-028b370252739e107

At the bottom, there are links for CloudShell, Feedback, Language, and a search bar.

The prod-server is configured through Kubernetes setup and now to deploy the application through cluster a new job is created namely *Deploy_Prod_server*.

The screenshot shows the Jenkins dashboard. At the top, there's a search bar and a user profile for 'Srija Bichala'. Below it, the dashboard shows a list of recent items. A modal window is open for creating a new item, with the title 'Enter an item name' and a text input field containing 'Deploy_Prod_server'. The input field has a red border and a 'Required field' message below it.

The modal then branches into three options:

- Freestyle project**: Described as the central feature of Jenkins, allowing combining any SCM with any build system.
- Pipeline**: Described as orchestrating long-running activities suitable for building pipelines (formerly known as workflows).
- Multi-configuration project**: Described as projects that need a large number of different configurations, such as testing on multiple environments.

At the bottom of the modal, there are 'OK' and 'Cancel' buttons, and a note about 'Suitable for Jenkinsfile, etc.' and 'Folder'.

At the very bottom of the screen, there's a Windows taskbar with icons for File Explorer, Mail, Google Chrome, Word, and File Explorer again.

The screenshot shows the Jenkins configuration interface for a job named 'Deploy_Prod_server'. The left sidebar lists 'General', 'Advanced Project Options', and 'Pipeline'. The main area is titled 'General' with a status of 'Enabled' (indicated by a green checkmark). A 'Description' field contains the text 'Deploy Medicure on Production Server'. Below it are three checkboxes: 'Discard old builds', 'Do not allow concurrent builds', and 'Do not allow the pipeline to resume if the controller restarts'. At the bottom are 'Save' and 'Apply' buttons.

Configure

General

Description

Deploy Medicure on Production Server

[Plain text] Preview

Discard old builds ?

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

Save Apply

Here is the pipeline script to deploy through Kubernetes.

Stage : Create SSH connection and apply the deployment.yml file

The screenshot shows the Jenkins Pipeline configuration page. The pipeline is named 'Deploy_Prod_server' and has a single stage named 'Kubernetes deployment'. The script section contains the following Groovy code:

```
8 }  
9  
10 stage('Kubernetes deployment'){  
11 steps{  
12 shagent(['kubedeploy'])  
13 sh 'scp -o StrictHostKeyChecking=no deployment.yml ubuntu@172.31.39.255:/home/ubuntu'  
14 script{  
15 try{  
16 sh "ssh ubuntu@172.31.39.255 sudo kubectl apply -f ."  
17 }catch(error){  
18 sh "ssh ubuntu@172.31.39.255 sudo kubectl create -f ."  
19 }  
20 }  
21 }  
22 }  
23 }  
24 }
```

A message below the script states: "The script has not yet been approved, but it will be approved on save". There is also a checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons.

Here is the deployment.yml file pushed onto git repo.

The screenshot shows a GitHub repository page for 'Medicure'. The repository has 35 commits. A purple oval highlights the 'deployment.yml' file, which was updated 5 minutes ago. Other files shown include '.mvn/wrapper', 'prod-server', 'src', 'test-server', '.DS_Store', '.gitignore', 'Dockerfile', 'JenkinsFile2', 'Jenkinsfile', 'ansible-playbook.yml', 'mvnw', 'mvnw.cmd', 'pom.xml', and 'selejarmedicare.jar'. The GitHub interface includes sections for Releases, Packages, Contributors, and Languages.

The deployment.yml file to create deployment using Kubernetes cluster with scaling factor 2, and using Nodeport service to deploy on port 31000.

The screenshot shows the 'deployment.yml' file content in a code editor. The file defines a Deployment object with a spec section containing a replicas field set to 2. A blue arrow points to this line. The file also defines a Service object with a selector matching the Deployment's app label. The code editor interface includes tabs for Code, Blame, and Raw, and various file operations like copy, download, and edit.

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: medicure
5   labels:
6     app: medicure
7   spec:
8     replicas: 2 ←
9     selector:
10       matchLabels:
11         app: medicure
12     template:
13       metadata:
14         labels:
15           app: medicure
16       spec:
17         containers:
18           - name: medicure-container
19             image: srija1991/medicure
20             ports:
21               - containerPort: 8080
22 ...
23 apiVersion: v1
24 kind: Service
25 metadata:
26   name: medicure-service
27   labels:
```

```

14     labels:
15       app: medicure
16   spec:
17     containers:
18       - name: medicure-container
19         image: srija1991/medicure
20       ports:
21         - containerPort: 8080
22       ...
23     apiVersion: v1
24   kind: Service
25   metadata:
26     name: medicure-service
27     labels:
28       app: medicure
29   spec:
30     selector:
31       app: medicure
32     type: NodePort
33   ...
34     ports:
35       - nodePort: 31000
36     port: 8082
37     targetPort: 8082

```

The job is run successfully and here is the console output .

Started by user Srija Bichala
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Deploy_Prod_server
[Pipeline] {
[Pipeline] stage
[Pipeline] { (github source code)
[Pipeline] git
The recommended git tool is: git
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Deploy_Prod_server/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Srija1991/Medicure.git # timeout=10
Fetching upstream changes from https://github.com/Srija1991/Medicure.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Srija1991/Medicure.git +refs/heads/*:refs/remotes/origin/* # timeout=10

Instance details | EC2 | EC2 Instance Connect | Medicure | Deploy_Prod_server #2 | cAdvisor - / | Node Exporter | + | - | X

Not secure | ec2-3-110-151-185.ap-south-1.compute.amazonaws.com:8080/job/Deploy_Prod_server/2/console

Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec-... Book Your Appoint... Weekly Feedback F...

Dashboard > Deploy_Prod_server > #2

```
[Pipeline] { (Kubernetes deployment)
[Pipeline] sshagent
[ssh-agent] Using credentials ubuntu (kubedeploy)
[ssh-agent] Looking for ssh-agent implementation...
[ssh-agent]   Exec ssh-agent (binary ssh-agent on a remote machine)
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-XXXXXowT00t/agent.5093
SSH_AGENT_PID=5093
Running ssh-add (command line suppressed)
Identity added: /var/lib/jenkins/workspace/Deploy_Prod_server@tmp/private_key_9747631685878148770.key
(/var/lib/jenkins/workspace/Deploy_Prod_server@tmp/private_key_9747631685878148770.key)
[ssh-agent] Started.
[Pipeline] {
[Pipeline] sh
+ scp -o StrictHostKeyChecking=no deployment.yml ubuntu@172.31.39.255:/home/ubuntu
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ ssh ubuntu@172.31.39.255 sudo kubectl apply -f .
deployment.apps/medicure unchanged
service/medicure-service configured
[Pipeline] }
[Pipeline] // script
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
```

Instance details | EC2 | EC2 Instance Connect | Medicure | Deploy_Prod_server #2 | cAdvisor - / | Node Exporter | + | - | X

Not secure | ec2-3-110-151-185.ap-south-1.compute.amazonaws.com:8080/job/Deploy_Prod_server/2/console

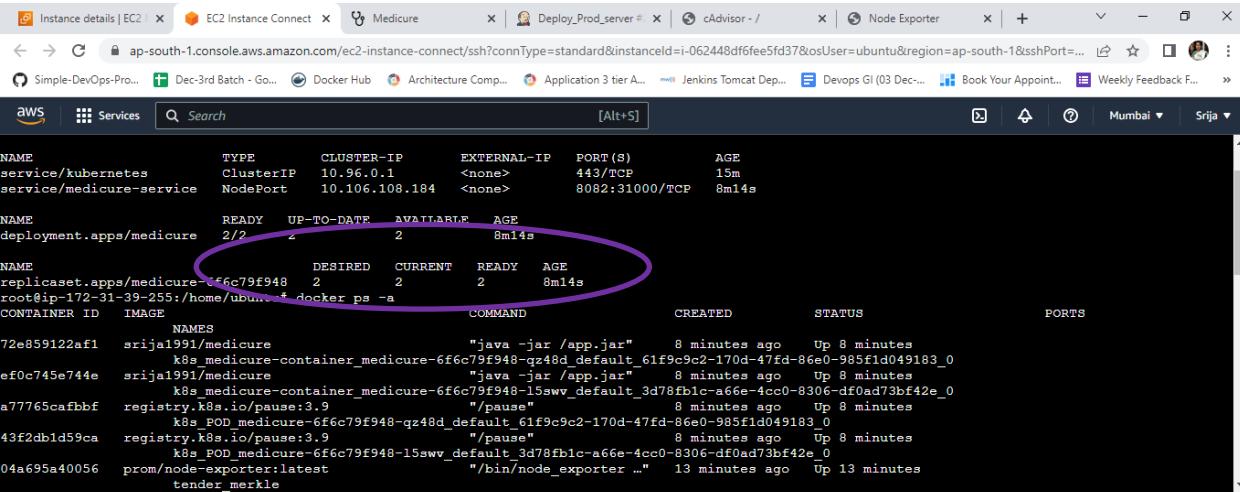
Simple-DevOps-Pro... Dec-3rd Batch - Go... Docker Hub Architecture Comp... Application 3 tier A... Jenkins Tomcat Dep... Devops GI (03 Dec-... Book Your Appoint... Weekly Feedback F...

Dashboard > Deploy_Prod_server > #2

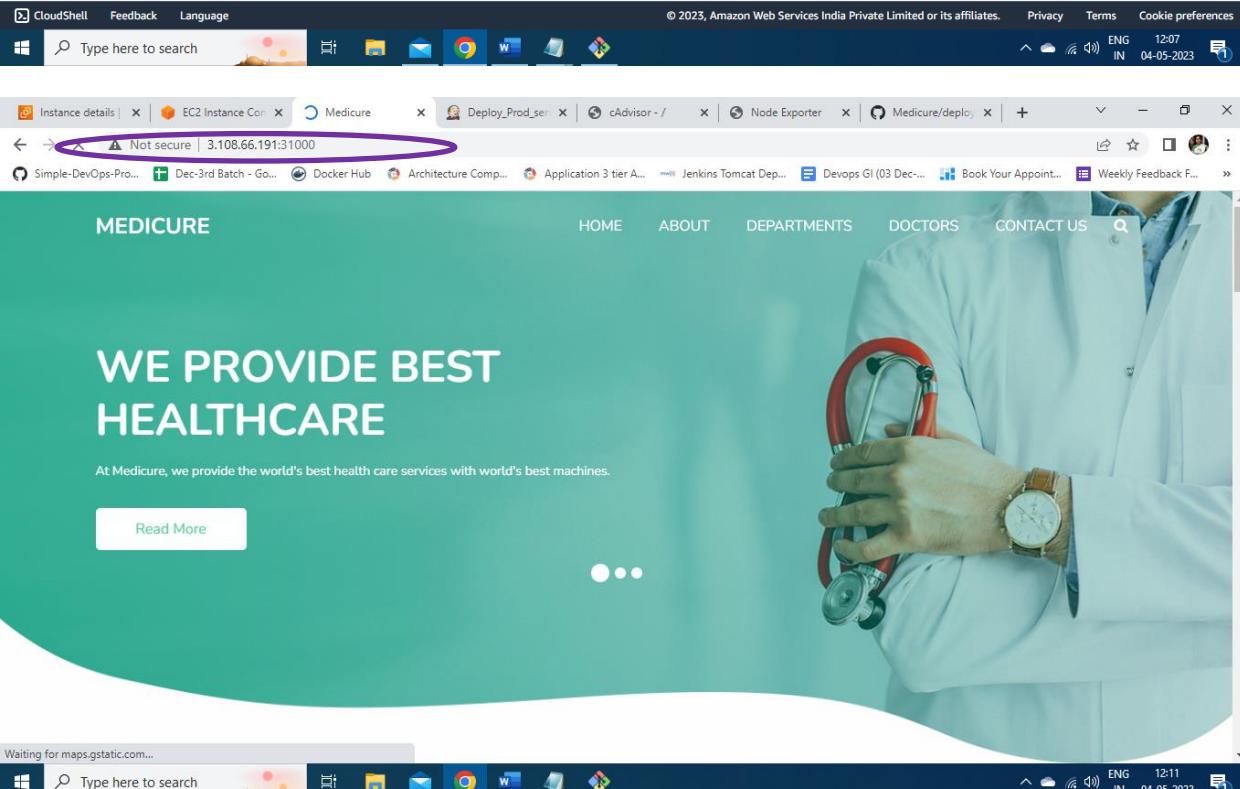
```
[Pipeline] {
[Pipeline] sh
+ scp -o StrictHostKeyChecking=no deployment.yml ubuntu@172.31.39.255:/home/ubuntu
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ ssh ubuntu@172.31.39.255 sudo kubectl apply -f .
deployment.apps/medicure unchanged
service/medicure-service configured
[Pipeline] }
[Pipeline] // script
[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 5096 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
```

Finished: SUCCESS

The prod-server is manually entered and checked for the pods and deployments using command `kubectl get all`. Here we can see the deployments 2/2 running stating that two pods are created using cluster on the port 31000. IP address of prod_server is 3.108.66.191.



A screenshot of the AWS CloudWatch Metrics interface. It shows a table of metrics for various Kubernetes components. The table includes columns for NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. Two rows are circled in purple: 'deployment.apps/medicure' and 'replicaset.apps/medicure-6f6c79f948'. The 'deployment.apps/medicure' row shows 2/2 ready pods. The 'replicaset.apps/medicure-6f6c79f948' row shows 2 desired and current pods, both ready, with an age of 8m14s. Below the table, a terminal window shows the output of 'docker ps -a', listing several containers including 'srija1991/medicure' and 'registry.k8s.io/pause:3.9'.



A screenshot of a browser displaying the Medicure website. The URL bar shows 'Not secure | 3.108.66.191:31000'. The page features a green header with the word 'MEDICURE' and a navigation menu with links to HOME, ABOUT, DEPARTMENTS, DOCTORS, and CONTACT US. A large image of a doctor wearing a white coat and a stethoscope is on the right. The main content area has a teal background with the text 'WE PROVIDE BEST HEALTHCARE' and a 'Read More' button. At the bottom left, it says 'Waiting for maps.gstatic.com...'.

On the same production server cAdvisor and Node-exporter containers are started to take the metrics for monitoring using Prometheus and Grafana.

Here is the cAdvisor running on port 8080 on production server.

The screenshot shows a Microsoft Edge browser window with the URL <http://3.108.66.191:8080/containers/>. The page title is "cAdvisor". The content area displays an owl logo and the word "cAdvisor". Below this is a navigation bar with a single item: "root". Under the heading "Docker Containers" is the heading "Subcontainers". A list of four items is shown: "/init.scope", "/kubepods.slice", "/system.slice", and "/user.slice". The browser's status bar at the bottom shows the date and time as "04-05-2023 12:10".

Here is the Node-exporter running on port 9100 on prod_server.

The screenshot shows a Microsoft Edge browser window with the URL <http://3.108.66.191:9100>. The page title is "Node Exporter". The main content area contains the heading "Node Exporter" and a sub-section titled "Metrics". The browser's status bar at the bottom shows the date and time as "04-05-2023 12:10".

Now Prometheus_Grafana Server is configured to monitor the production server.

The screenshot shows the AWS EC2 Instances page with the search bar set to "Instance state = running". A table lists four instances: Master-server, test-server, Prometheus_Grafana, and prod-server. The Prometheus_Grafana row is selected, highlighted with a blue border. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. Below the table, a modal window titled "Instance: i-03b139f2b343dd9b1 (Prometheus_Grafana)" is open, displaying detailed instance information.

Here is the Instance summary of the monitoring server.

The screenshot shows the AWS EC2 Instance Details page for the instance i-03b139f2b343dd9b1 (Prometheus_Grafana). The "Public IPv4 address" field, which contains "52.66.143.157", is circled in purple. The page displays various instance details such as Instance ID, Public IP, Private IP, Instance type, and VPC ID. A purple arrow points from the text "Public IPv4 address" to the circled value "52.66.143.157".

The monitoring server is configured by adding the IP address of cAdvisor and node-exporter available from production server in the *prometheus.yml* file.

```

# The job name is added as a label 'job=<job name>' to any timeseries scraped from this config.
- job_name: "prometheus"

  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.

  static_configs:
    - targets: ["localhost:9090"]

- job_name: "Monitoring Pipeline through Nodeexporter"
  static_configs:
    - targets: ["3.108.66.191:9100"]

- job_name: "Monitoring through cAdvisor"
  static_configs:
    - targets: ["108.66.191:8080"]

      # - job_name: "Node_exporter"
      #static_configs:
      #  - targets: ['172.31.7.129:9100']

-- INSERT --

```

i-03b139f2b343dd9b1 (Prometheus_Grafana)
PublicIPs: 52.66.143.157 PrivateIPs: 172.31.36.123

The Prometheus is started using command `./prometheus`.

```

root@ip-172-31-36-123:/home/ubuntu/prometheus# ls
LICENSE NOTICE  console_libraries  consoles  data  prometheus  prometheus.yml  promtool
root@ip-172-31-36-123:/home/ubuntu# ./prometheus
ts=2023-05-04T06:48:02.925Z caller=main.go:491 level=info msg="No time or size retention was set so using the default time retention" duration=15d
ts=2023-05-04T06:48:02.925Z caller=main.go:535 level=info msg="Starting Prometheus Server" mode=server version="(version=2.37.6, branch=HEAD, revision=8ade24a23af6be0f35414d6e8ce09598446c29a2)"
ts=2023-05-04T06:48:02.925Z caller=main.go:540 level=info build_context="(go=gol.19.6, user=root@ef96027a7c3e, date=20230220-09:36:40)"
ts=2023-05-04T06:48:02.925Z caller=main.go:541 level=info host_details="(Linux 5.19.0-1024-aws #25~22.04.1-Ubuntu SMP Tue Apr 18 23:41:58 UTC 2023 x86_64 ip-172-31-36-123 (none))"
ts=2023-05-04T06:48:02.926Z caller=main.go:542 level=info fd_limits="(soft=1048576, hard=1048576)"
ts=2023-05-04T06:48:02.926Z caller=main.go:543 level=info vm_limits="(soft=unlimited, hard=unlimited)"
ts=2023-05-04T06:48:02.935Z caller=web.go:553 level=info component=web msg="Start listening for connections" address=0.0.0.0:9090
ts=2023-05-04T06:48:02.937Z caller=main.go:972 level=info msg="Starting TSDB ..."
ts=2023-05-04T06:48:02.943Z caller=repair.go:56 level=info component=tldb msg="Found healthy block" mint=1680601222649 maxt=1680609600000 ulid=01GY54NFKWU68457R9ZEA2DCX5
ts=2023-05-04T06:48:02.945Z caller=repair.go:56 level=info component=tldb msg="Found healthy block" mint=1681652366751 maxt=1681653600000 ulid=01GYAEEFKN2QG382QDFPPN1203D
ts=2023-05-04T06:48:02.947Z caller=repair.go:56 level=info component=tldb msg="Found healthy block" mint=1680875737649 maxt=1680883200000 ulid=01GYBYTJ2VWBSCKRDY0B43ZYK
ts=2023-05-04T06:48:02.948Z caller=repair.go:56 level=info component=tldb msg="Found healthy block" mint=1681881122085 maxt=1681884000000 ulid=01GZ3QS3SGANZA2RRBW2XXB0287
ts=2023-05-04T06:48:02.949Z caller=repair.go:56 level=info component=tldb msg="Found healthy block" mint=1681884000291 maxt=1681891200000 ulid=01GZ3QS3YJXZQ4G0AJVCT58PNJ

```

i-03b139f2b343dd9b1 (Prometheus_Grafana)
PublicIPs: 52.66.143.157 PrivateIPs: 172.31.36.123

Here we can see the metrics are added as Prometheus targets on port number 9090 of monitoring server successfully and are available to monitor.

The screenshot shows the Prometheus Targets page with two sections of scraped metrics:

- Monitoring Pipeline through Nodeexporter (1/1 up)**: Shows one target at `http://3.108.66.191:9100/metrics` labeled as UP. Labels include `instance="3.108.66.191:9100"` and `job="Monitoring Pipeline through Nodeexporter"`. Last Scrape was 6.615s ago with a duration of 18.298ms.
- Monitoring thorough Cadvisor (1/1 up)**: Shows one target at `http://3.108.66.191:8080/metrics` labeled as UP. Labels include `instance="3.108.66.191:8080"` and `job="Monitoring thorough Cadvisor"`. Last Scrape was 8.956s ago with a duration of 525.325ms.

From the monitoring server port number 3000 is allocated to Grafana monitoring and logged in to monitor.

The screenshot shows the Grafana login page. The URL in the address bar is `52.66.143.157:3000/login`, which is circled in red. The page features the Grafana logo and the text "Welcome to Grafana". It has fields for "Email or username" (containing "admin") and "Password" (containing "*****"). A "Log in" button is at the bottom, and a "Forgot your password?" link is below it. At the bottom of the page, there are links for Documentation, Support, Community, Open Source, version v9.4.7, and New version available.

A new datasource is added named prometheus6 and configured to collect metrics.

The screenshot shows the Grafana configuration interface for a new Prometheus data source. The 'Name' field is highlighted with a red oval, containing the value 'Prometheus-6'. The 'HTTP' section includes fields for 'URL' (set to 'http://52.66.143.157:9090/'), 'Allowed cookies', and 'Timeout'. The 'Auth' section offers 'Basic auth' or 'With Credentials' options. A success message 'Datasource updated' is displayed in a green box at the top right. The browser address bar shows the URL '52.66.143.157:3000/datasources/edit/Nusw0Ys4k'.

Type: Prometheus

Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the [free-forever Grafana Cloud plan](#).

Name: **Prometheus-6** Default:

HTTP

URL: http://52.66.143.157:9090/
Allowed cookies: New tag (enter key to add)
Timeout: Timeout in seconds

Auth

Basic auth With Credentials

Datasource updated

HTTP method: POST

Type and version

Prometheus type: Choose

Misc

Disable metrics lookup:
Default Editor: Choose
Custom query parameters: Example: max_source_resolution=5m&timeout=10

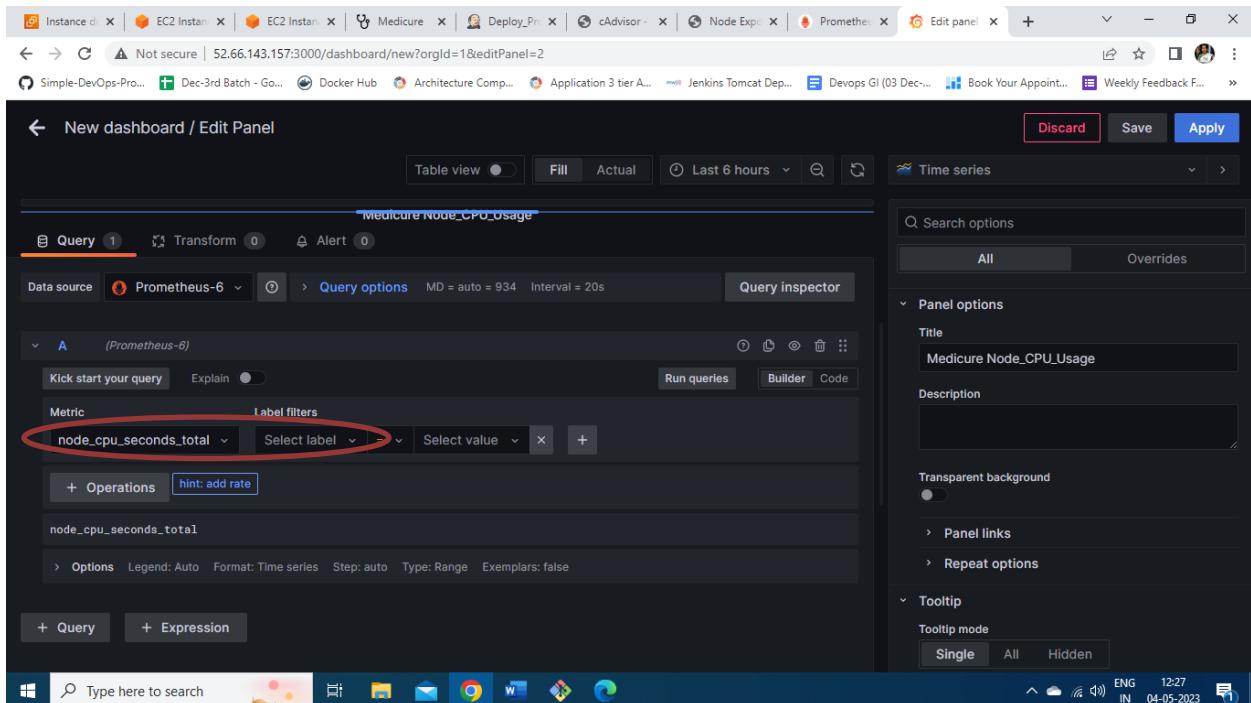
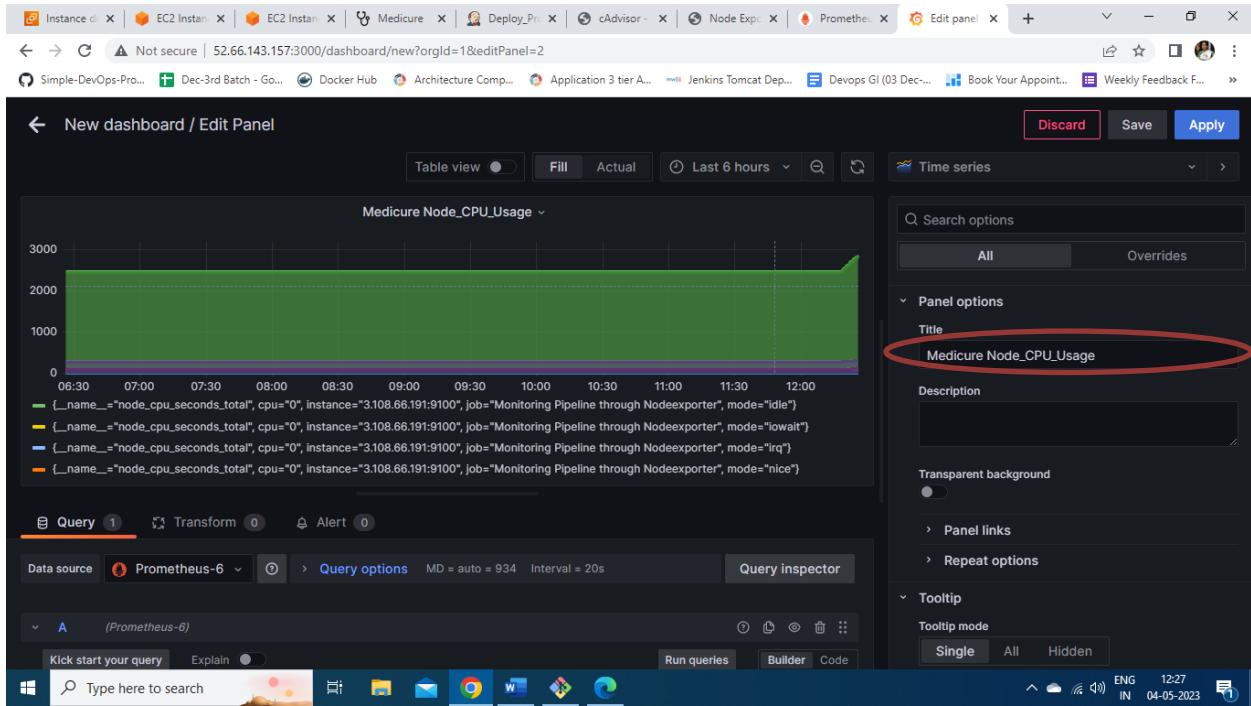
Exemplars

+ Add

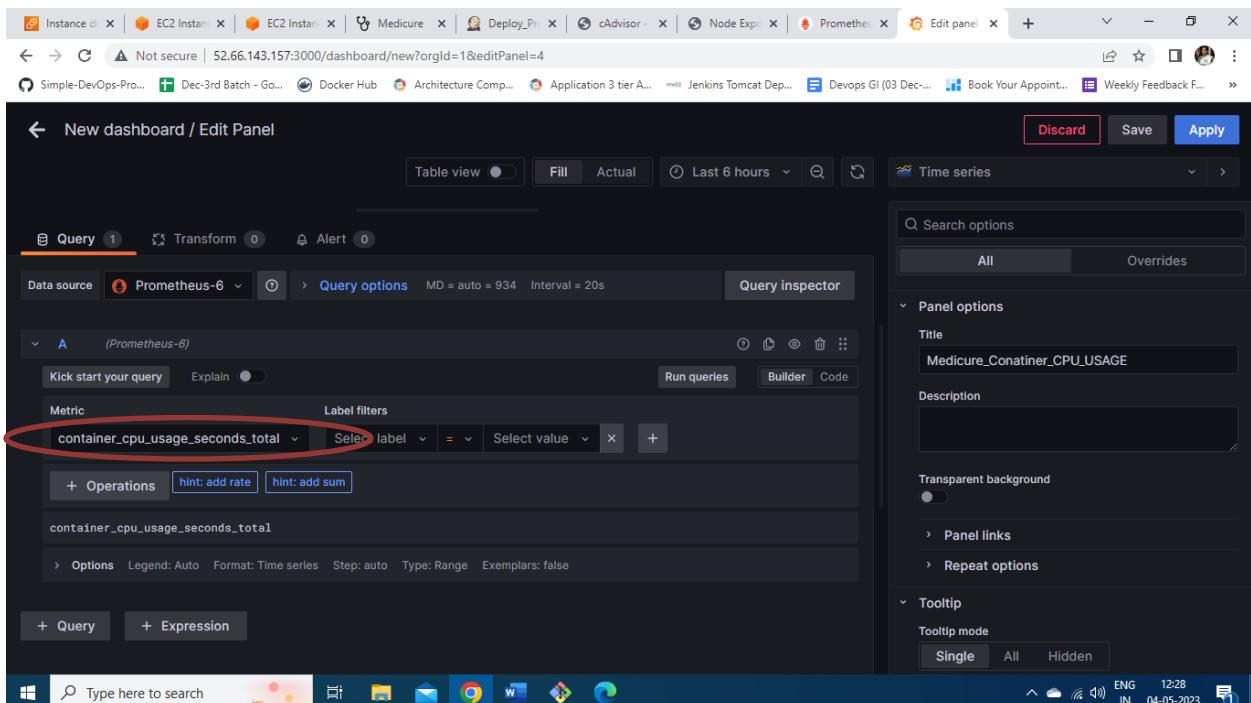
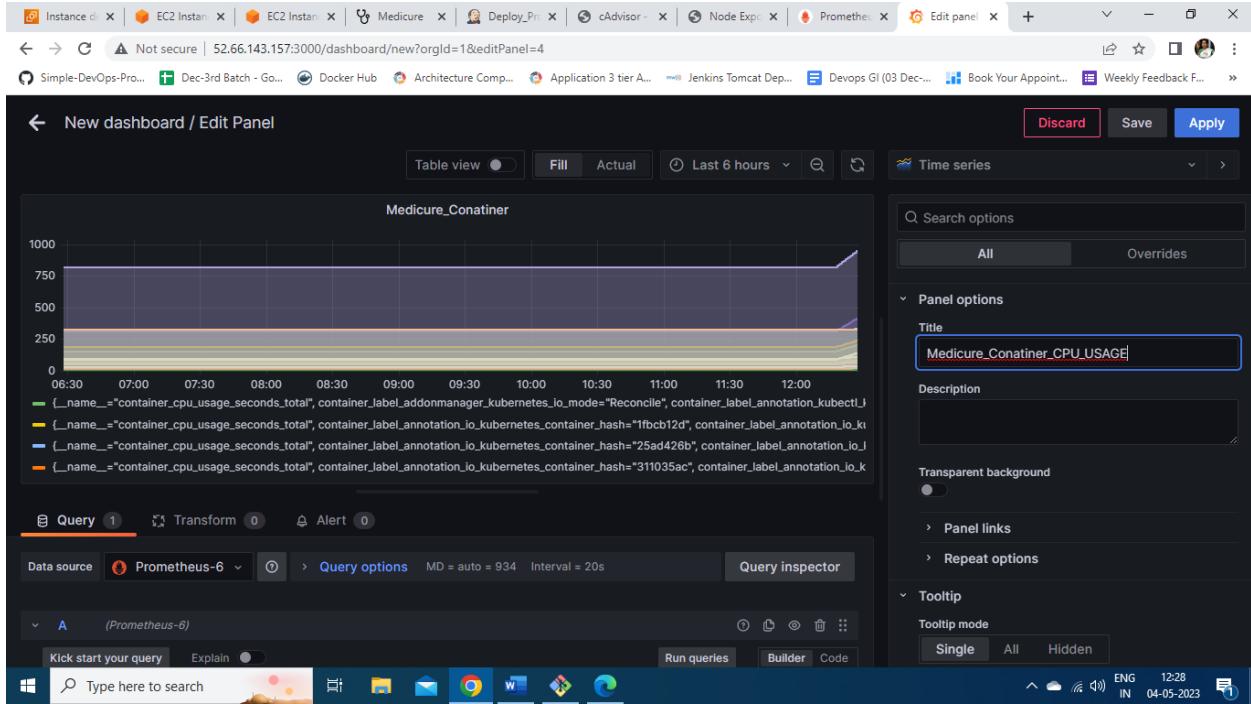
Back Explore Delete Save & test

Documentation | Support | Community | Open Source | v9.4.7 (4add91f03d) | New version available!

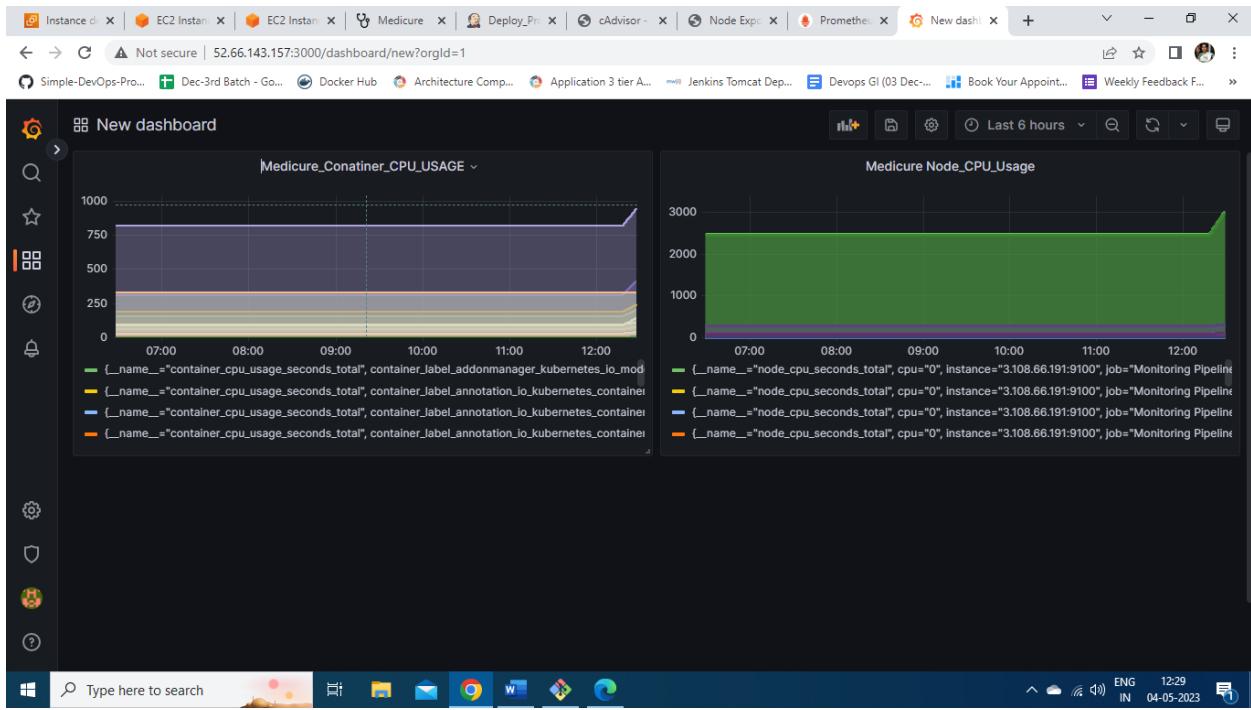
Now a Dashboard is created and a query called node_cpu_usage is run and monitored.



Here is graph showing the query container_cpu_usage.



The dashboard for monitoring CPU usgae is shown here.



THANK YOU