

### **Problem Description:-**

As soon as the developer pushes the updated code on the GIT master branch, the code should be checked out, compiled, tested, packaged and containerized. A new test-server should be provisioned using terraform and should be automatically configured using Ansible with all the required software's and as soon as the server is available, the application must be deployed to the test-server automatically.

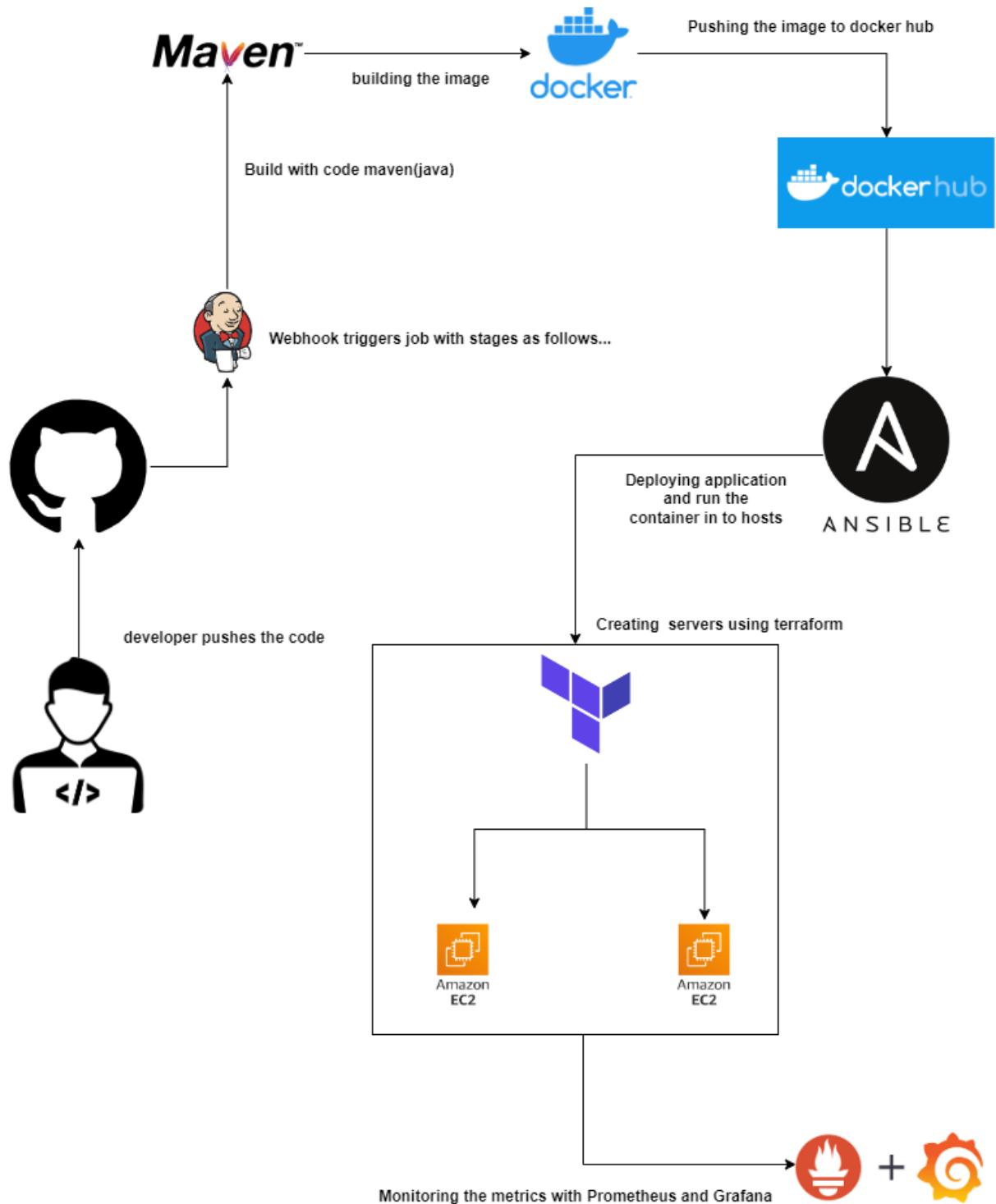
The deployment should then be tested using a test automation tool, and if the build is Successful, Prod server must be configured with all the software it should be pushed to the prod server. All this should happen automatically and should be triggered from a push to the GitHub master branch.

Continuous monitoring server must be configured to monitor the test as well as prod server using Prometheus and Grafana should be configured to display a dashboard with following metrics.

1. CPU utilization
2. Disk Space Utilization
3. Total Available Memory

## Problem Solution:-

### Approach-



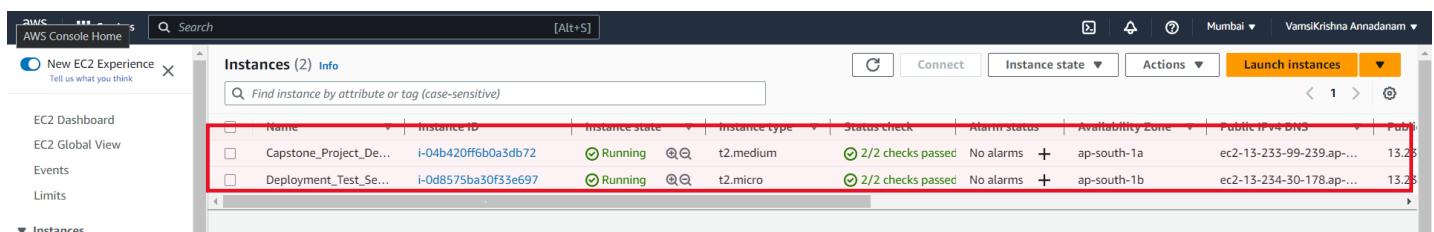
# Source git hub - <https://github.com/vamsikrishna918/Finance-Web-Application-E2E>

## Step 1:

### 1.A- EC2 instance setup

Creating 2 EC2 instances name as below

1. Capstone\_Project\_Development (T2.Medium with 30gib)
2. Deployment\_Test\_Server ( T2.Micro with 8gib)

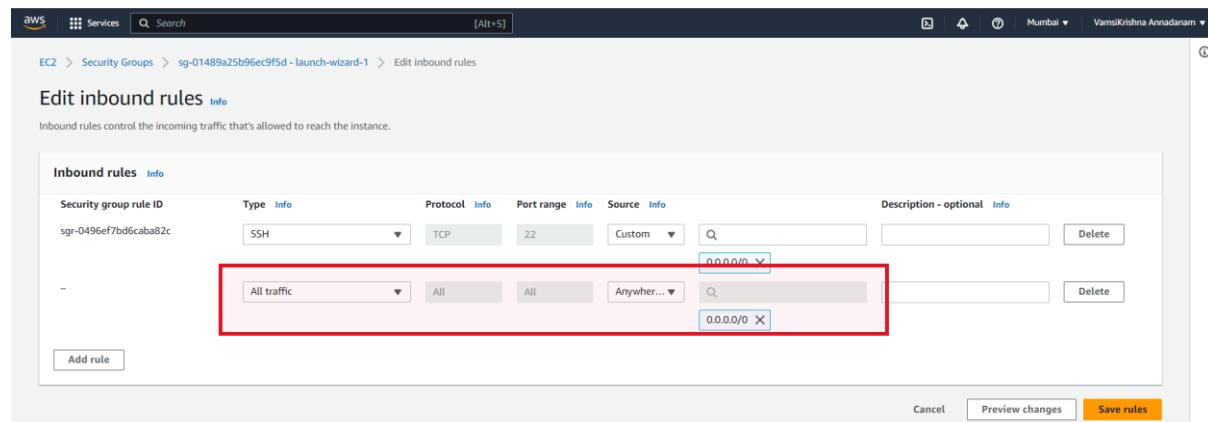


Name	Instance ID	Instance State	Status Check	Alarm Status	Availability Zone	Public IP v4 DNS	Pub
Capstone_Project_De...	i-04b420ff6b0a3db72	Running	2/2 checks passed	No alarms	ap-south-1a	ec2-13-233-99-239.ap...	13.23
Deployment_Test_Ser...	i-0d8575ba30f33e697	Running	2/2 checks passed	No alarms	ap-south-1b	ec2-13-234-30-178.ap...	13.23

### 1.B

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

Path: Security->Edit Inbound rules



Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0496ef7bd6cabab2c	SSH	TCP	22	Custom	0.0.0.0/0
-	All traffic	All	All	Anywhere	0.0.0.0/0

## 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](https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-debian)
8. Terraform - <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>
9. aws cli - <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

AWS Console Home Search [Alt+S]

```
root@ip-172-31-36-251:/# git --version
git version 2.34.1
root@ip-172-31-36-251:/# java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
root@ip-172-31-36-251:/# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.19, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-1025-aws", arch: "amd64", family: "unix"
root@ip-172-31-36-251:/# jenkins --version
2.401.1
root@ip-172-31-36-251:/# docker --version
Docker version 24.0.2, build cb74dfc
root@ip-172-31-36-251:/# ansible --version
ansible [core 2.14.6]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible 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-172-31-36-251:/#
```

i-04b420ff6b0a3db72 (Capstone\_Project\_Development)  
PublicIPs: 52.66.213.147 PrivateIPs: 172.31.36.251

## 1.d - Password less authentication

Password less authentication b/w 2 servers ( capstone\_project\_development (and) deployment\_test\_server)

On capstone\_project\_development instance

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 * +. . |
| . . = 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:AFIP1190A+Zj2rUDuoNXoW8BUVp4QwkkudTphdDLbmw ubuntu@ip-172-31-62-28
The key's randomart image is:
+---[RSA 3072]---+
| ..+ +B+0Boo   |
| . =+.***. .   |
| .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

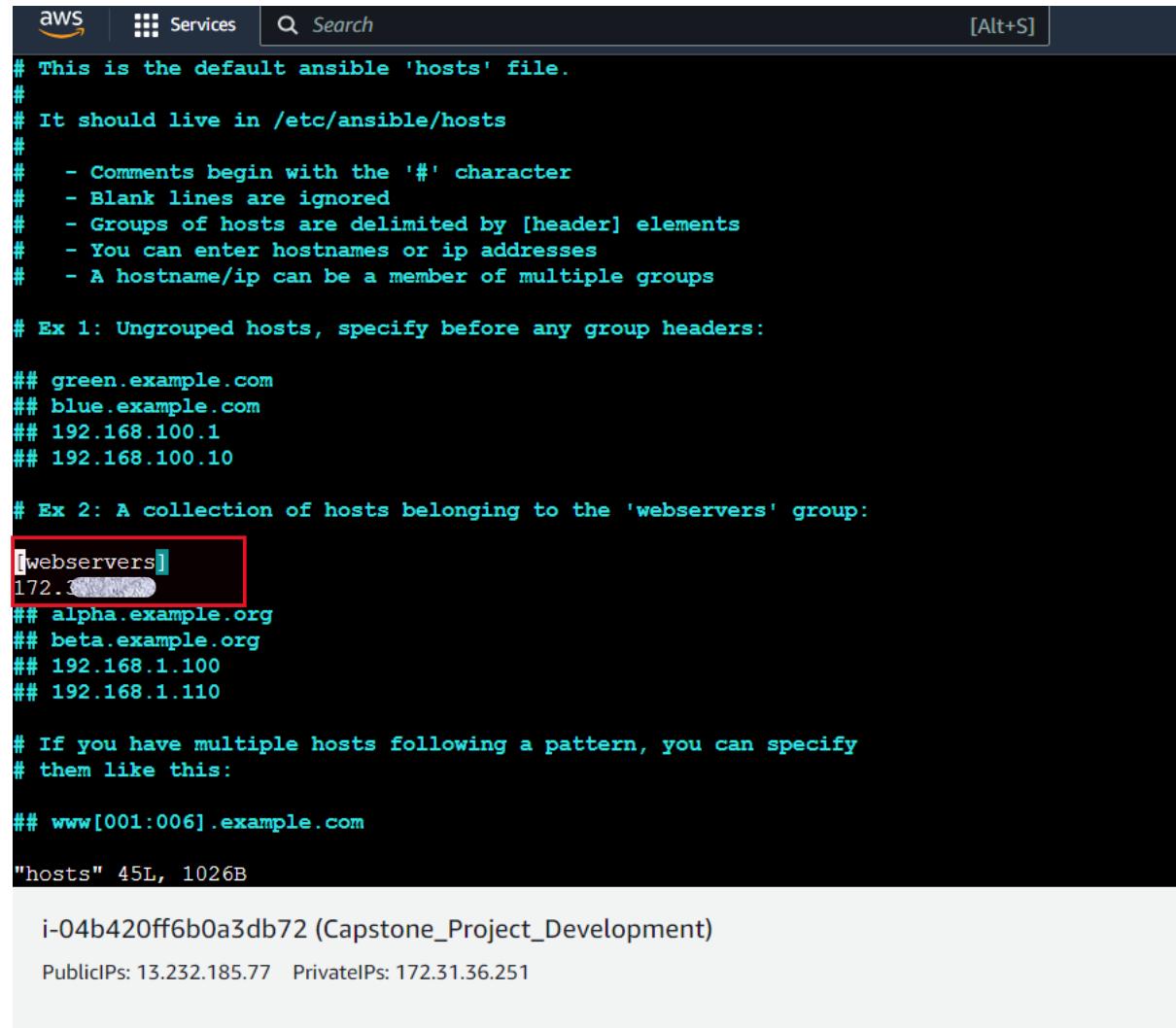
Do command: sudo su -

Cd /ect/ansible

```
root@ip-172-31-36-251:/etc# cd /etc
r AWS Console Home 5-251:/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(deployment\_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:

[webserver]
172.31.36.251
## 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
```

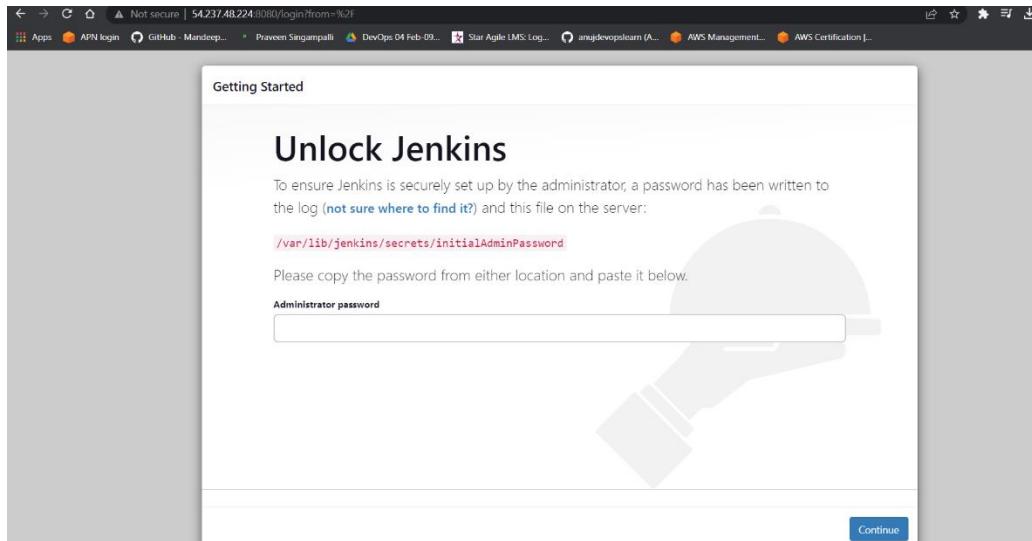
## 1.f Jenkins setup

Setting up the Jenkins (default running on 8080)

User- vams\*\*\*\*\*

Pas- pas\*\*\*\*\*

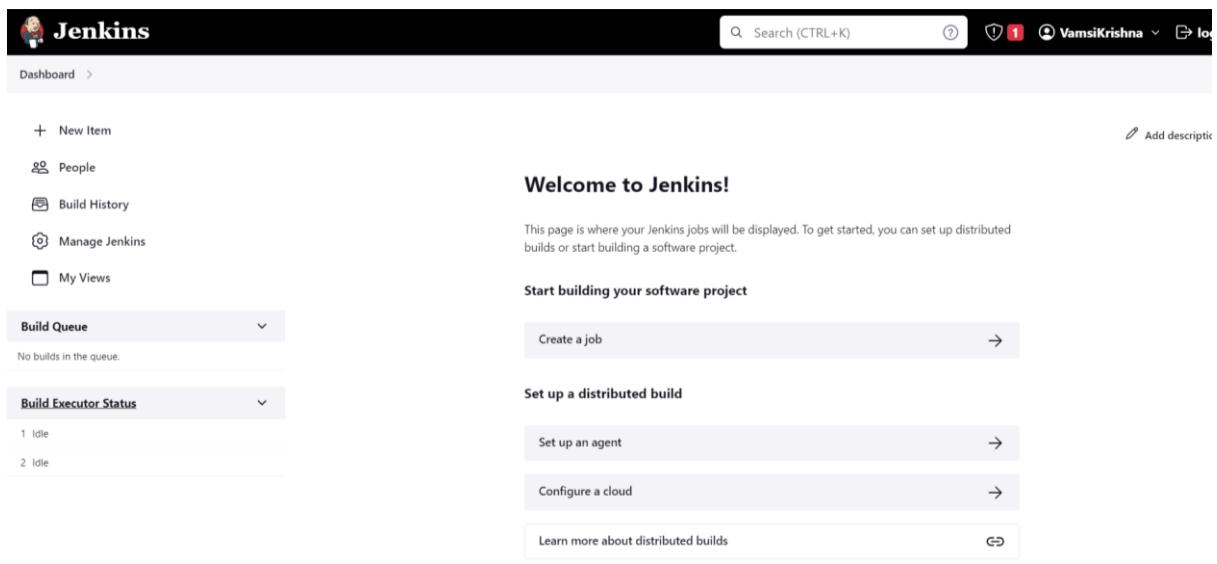
We can access Jenkins using [http://IP :8080](http://IP:8080)



### 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

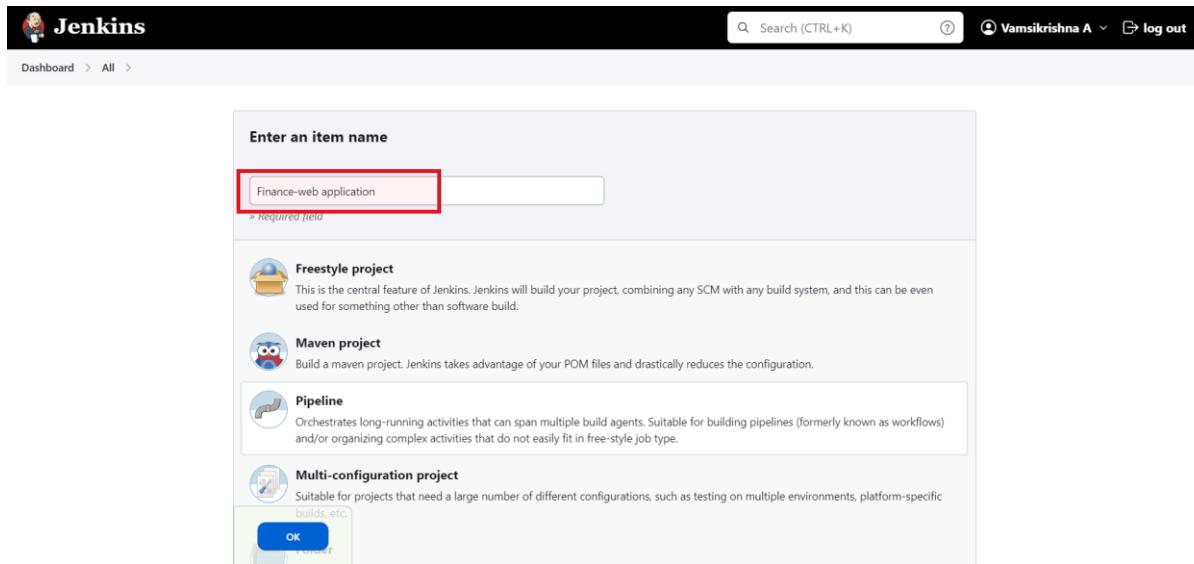
- Ansible
- Html publisher
- Docker
- Git
- Maven

The screenshot shows the Jenkins 'Available plugins' page. The left sidebar has links for Updates, Available plugins (which is selected and highlighted in blue), Installed plugins, Advanced settings, and Download progress. The main content area has a search bar labeled 'Search available plugins'. Below it is a table with columns 'Install', 'Name', and 'Released'. The table lists three plugins:

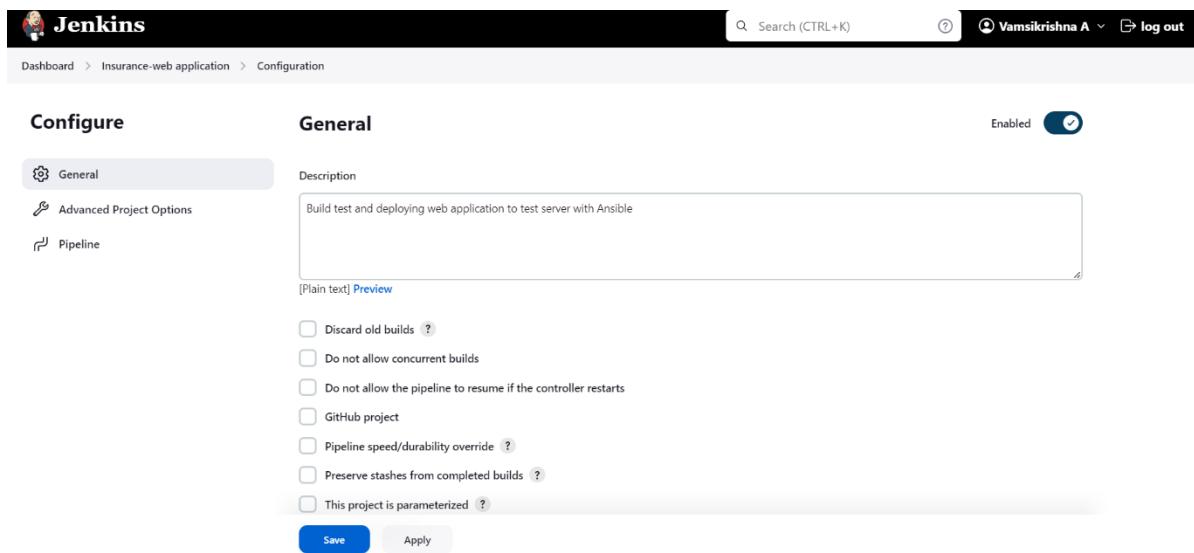
Install	Name	Released
<input type="checkbox"/>	<a href="#">Command Agent Launcher</a> 100.v2f6722292ee8 Agent Management	1 mo 23 days ago
<input type="checkbox"/>	<a href="#">Oracle Java SE Development Kit Installer</a> 66.vd8fa_64ee91b.d Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	2 mo 1 day ago
<input type="checkbox"/>	<a href="#">JavaScript GUI Lib: ACE Editor bundle</a> 1.1 JavaScript GUI Lib: ACE Editor bundle plugin. <small>This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. <a href="#">Learn more</a>.</small>	7 yr 3 mo ago

At the bottom, there are two buttons: 'Install without restart' and 'Download now and install after restart'. A status message says 'Update information obtained: 39 min ago' and a 'Check now' link.

## Step 2: Creating job



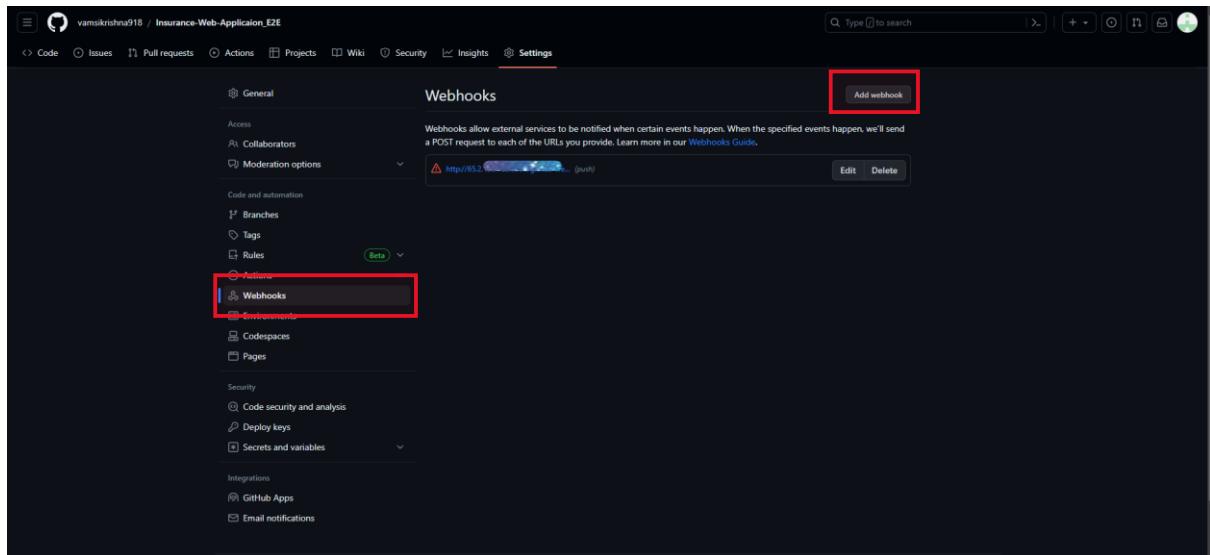
The screenshot shows the Jenkins interface for creating a new job. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information. Below it, a breadcrumb trail shows 'Dashboard > All >'. A modal window titled 'Enter an item name' is open, containing a text input field with the value 'Finance-web application'. This field is highlighted with a red border. Below the input field, a message says '» Required field'. The modal also contains several project type options: 'Freestyle project' (selected), 'Maven project', 'Pipeline', and 'Multi-configuration project'. Each option has a brief description and a small icon. At the bottom of the modal is a blue 'OK' button.



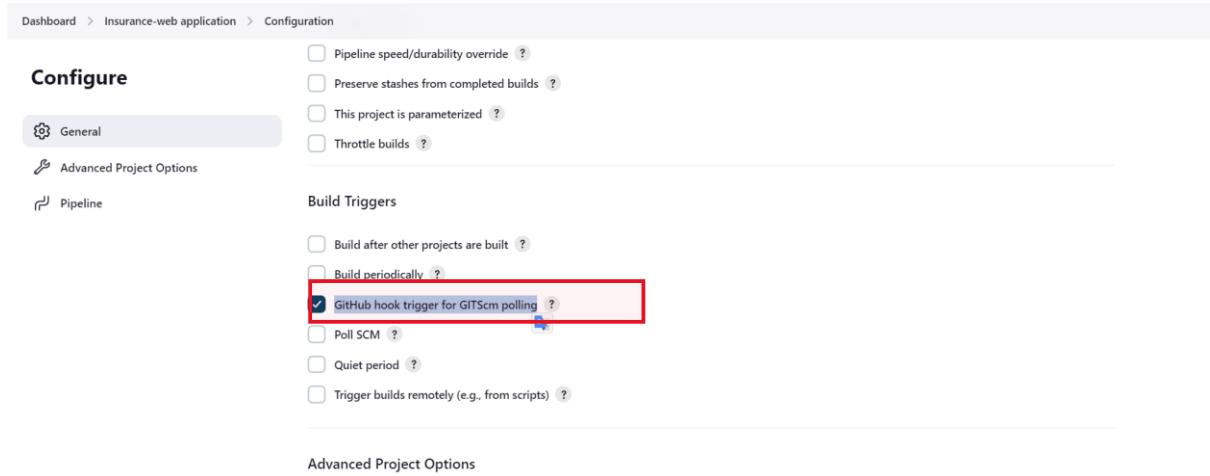
The screenshot shows the 'General' configuration page for a Jenkins job named 'Insurance-web application'. The top navigation bar includes the Jenkins logo, search, and user info. The breadcrumb trail shows 'Dashboard > Insurance-web application > Configuration'. On the left, a sidebar lists 'Configure' (selected), 'General', 'Advanced Project Options', and 'Pipeline'. The main content area is titled 'General' and shows the 'Enabled' status as 'On'. A 'Description' field contains the text 'Build test and deploying web application to test server with Ansible'. Below the description are several checkboxes for build options: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', and 'This project is parameterized'. At the bottom are 'Save' and 'Apply' buttons.

**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{
                git branch: 'main', url: 'https://github.com/vamsikrishna918/Finance-Web-Application-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

The screenshot shows two Jenkins management pages. The top page is 'Credentials' and the bottom page is 'Stores scoped to Jenkins'. Both pages are under the 'Manage Jenkins' section.

**Credentials Page:**

T	P	Store ↓	Domain	ID	Name
File	System	System	(global)	Dockerhub	vamsi12358/***** (Dockerhub)
File	System	System	(global)	dockercreds	dockercreds
File	System	System	(global)	testcreds	jenkins (test server pem file)
File	System	System	(global)	ansiblecreds	ubuntu (ansible credentials)

**Stores scoped to Jenkins Page:**

P	Store ↓	Domains
System	System	(global)

Icon: S M L

# Jenkins pipeline - deploying to test server job

Dashboard > Finance-web application > Configuration

**Configure**

Definition Pipeline script

General Advanced Project Options Pipeline

Script ?

```
1 pipeline(
2     agent any
3
4     stages{
5         stage("Code checkout"){
6             steps{
7                 git branch: 'main', url: 'https://github.com/vamsikrishna918/Finance-Web-Application-E2E'
8             }
9         }
10        stage("Build with Maven"){
11            steps{
12                echo ****building with maven****
13                sh '''mvn clean package'''
14            }
15        }
16        stage("Publish the report"){
17            steps{
18                echo "generating test reports"
19                publishHTML(alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/finance')
20            }
21        }
22        stage("Image prune"){
23        }
24    }
25 }
```

Use Groovy Sandbox ?

Save Apply

Dashboard > Finance-web application > Configuration

**Configure**

Definition Pipeline script

General Advanced Project Options Pipeline

Script ?

```
1 pipeline{
2     agent any
3
4     stages{
5         stage("Pushing Docker image to Hub"){
6             steps{
7                 script {
8                     echo ****Pushing Docker image to Hub****
9                     withCredentials([string(credentialsId: 'dockerhubpwd', variable: 'dockerhubpwd')]) {
10                         sh "docker login -u vamsi12358 -p ${dockerhubpwd} docker.id"
11                         sh 'docker push vamsi12358/financemeapp:v1'
12                     }
13                 }
14             }
15         }
16         stage("deploying to Test server"){
17             steps{
18                 script {
19                     echo ****Deploying Application to Test server****
20                     ansiblePlaybook become: true, credentialsId: 'ansiblecreds', disableHostKeyChecking: true, inventory: '/etc/ansible/hosts'
21                 }
22             }
23         }
24     }
25 }
```

Use Groovy Sandbox ?

Save Apply

## Build the job-

The screenshot shows a CI/CD pipeline interface for a project named "Finance-web application". The main area is titled "Stage View" and displays a grid of build stages. The columns are labeled: Code checkout, Build with Maven, Publish the report, Image prune, Build Docker image, Push Docker image to DockerHub, and deploying to Test server. A red box highlights the first three columns. Below the grid, there is a table of build history entries, each with a timestamp and a status indicator (e.g., Jun 24 23:36, No Changes, Success). The interface includes a sidebar with various configuration and monitoring options.

## Docker hub-

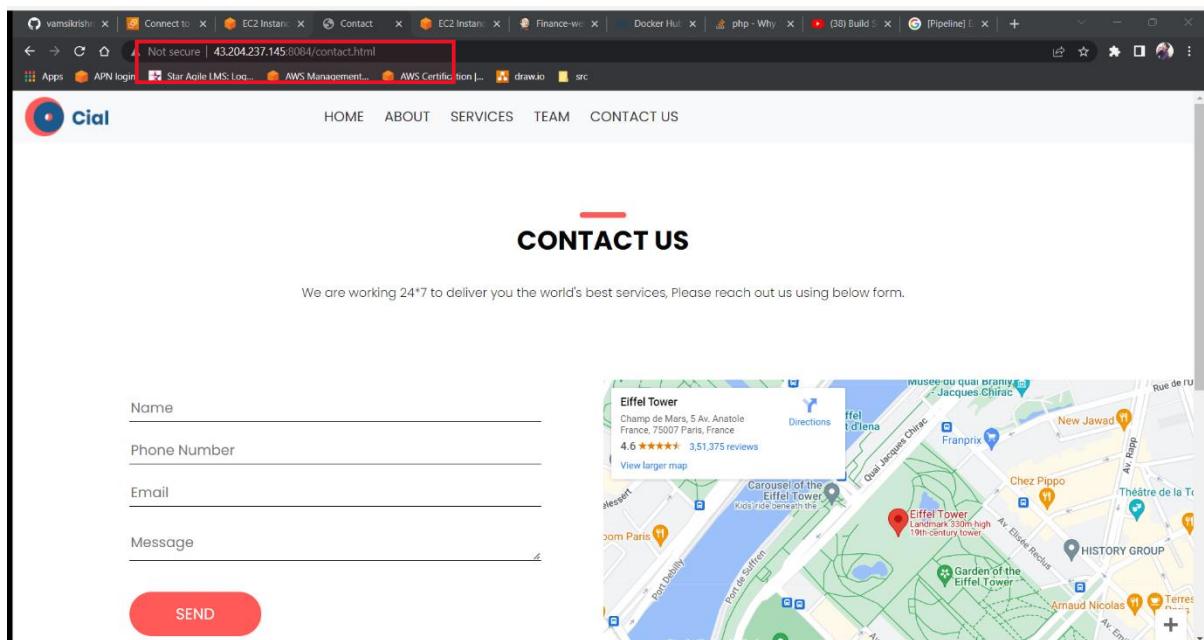
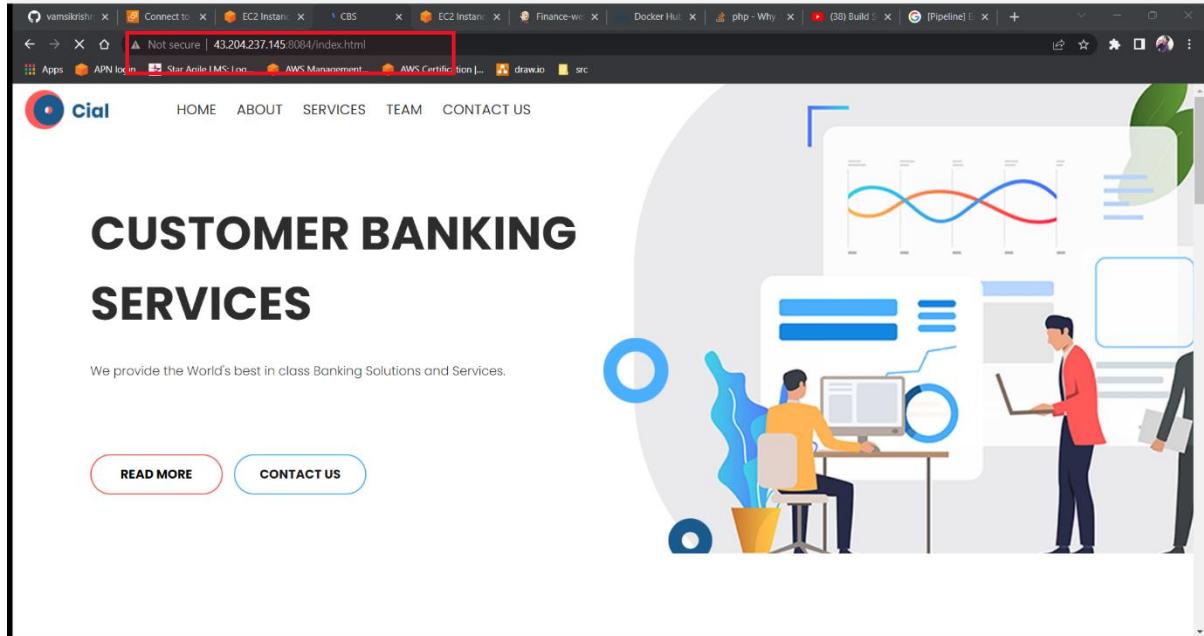
<https://hub.docker.com/>

The screenshot shows the Docker Hub user interface for the user "vamsi12358". The top navigation bar includes links for Explore, Repositories, Organizations, Help, and a user profile section. The main content area displays a list of repositories under the user's account. One repository, "vamsi12358 / financeapp", is highlighted with a red box. To the right of the repository list, there are two promotional banners: one for creating an organization and managing repositories, and another for Docker Community All-Hands.

## Deployed Application on test server

Accessing the web-application on test server

Deployment\_test\_server public ip:8084/

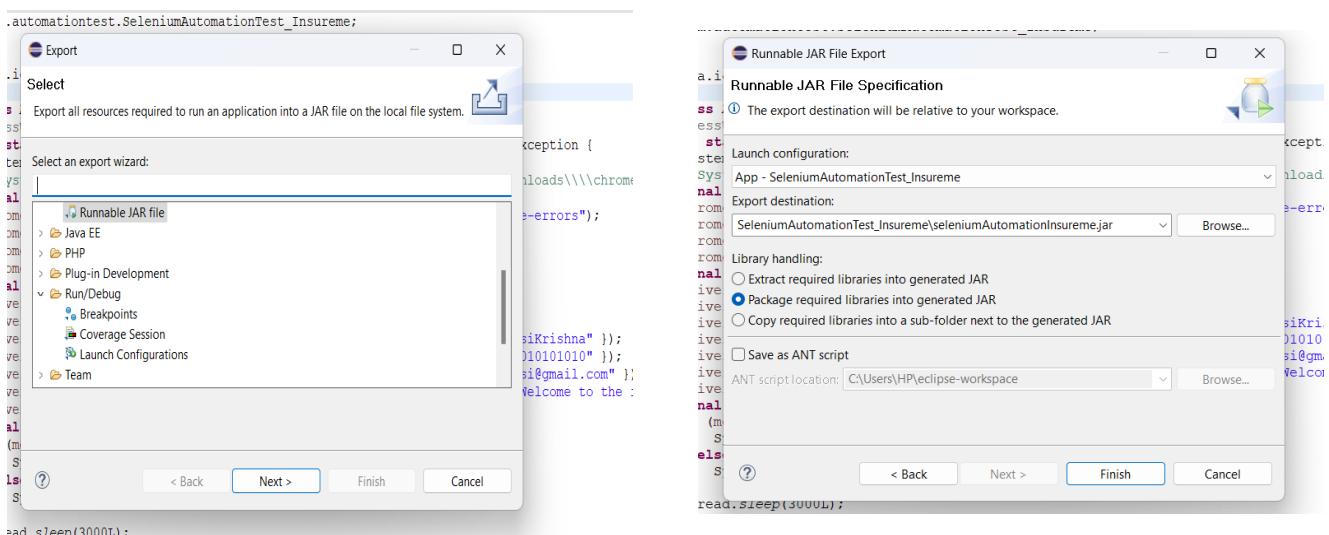
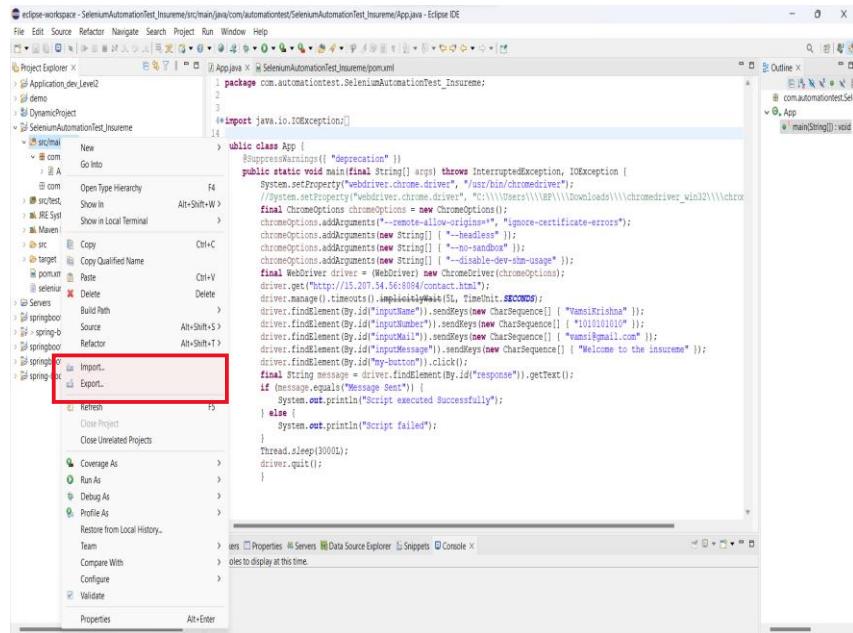


## Test -Automation with selenium

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

```
App.java X SeleniumAutomationTest_Insureme/pom.xml
4 import java.io.IOException;
5
6 import org.apache.commons.io.FileUtils;
7 import org.openqa.selenium.By;
8 import org.openqa.selenium.OutputType;
9 import org.openqa.selenium.TakesScreenshot;
10 import org.openqa.selenium.WebDriver;
11 import org.openqa.selenium.chrome.ChromeDriver;
12 import org.openqa.selenium.chrome.ChromeOptions;
13 import io.github.bonigarcia.wdm.WebDriverManager;
14
15 /**
16  * finance-me application test
17 *
18 */
19 public class App
20 {
21     public static void main( String[] args ) throws InterruptedException, IOException
22     {
23         //System.setProperty("webdriver.chrome.driver", "D:\\chrome-driver\\chromedriver.exe");
24         System.setProperty("webdriver.chrome.driver", "/usr/bin/chromedriver");
25
26         WebDriverManager.chromedriver().setup();
27         ChromeOptions chromeOptions = new ChromeOptions();
28
29         //To the scripts without opening GUI and need to process at bg use --headless
30         chromeOptions.addArguments("--headless");
31         chromeOptions.addArguments("--disable-dev-shm-usage");
32         chromeOptions.addArguments("--remote-allow-origins=*");
33         WebDriver driver = new ChromeDriver(chromeOptions);
34
35         System.out.println( "Selenium test scripts executed started ...." );
36
37
38         System.out.println("Opening finance-me Browser");
39         driver.get("http://3.110.221.50:8084/contact.html");
```

## 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](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-

The screenshot shows the Jenkins interface for creating a new item. The title bar says "Jenkins". The top navigation bar includes "Search (CTRL+K)", the user name "Vamsikrishna A", and a "log out" link. Below the navigation is a breadcrumb trail: "Dashboard > All >". The main content area has a form titled "Enter an item name" with a placeholder "Financeme-selenium-test-automation" and a note "» Required field". Below the form are four project type options:

- 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.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds etc.

At the bottom of the dialog is an "OK" button.

Setup the build trigger, which the automation job builds after the Finance me application deployed on test server(deployment\_test\_server).

The screenshot shows the Jenkins configuration page for the "Financeme-selenium-test-automation" job. The top navigation bar includes "Dashboard > Financeme-selenium-test-automation > Configuration". The left sidebar has tabs: "General" (selected), "Advanced Project Options", and "Pipeline". The main content area is titled "Configure".

The "Build Triggers" section is highlighted with a red box. It contains the following configuration:

- Build after other projects are built**:
  - Projects to watch:
    - Finance-web application,
  - Trigger only if build is stable
  - Trigger even if the build is unstable
  - Trigger even if the build fails
  - Always trigger, even if the build is aborted
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Quiet period

At the bottom of the "Build Triggers" section are "Save" and "Cancel" buttons.

```

pipeline{
    agent any

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

```

Dashboard > Financeme-selenium-test-automation > Configuration

**Definition**

**Configure**

Pipeline script

**General**

Advanced Project Options

Pipeline

```

1 * pipeline{
2     agent any
3
4     stages{
5         stage("Test automation Code checkout"){
6             steps{
7                 git branch: 'main', url: 'https://github.com/vamsikrishna918/Selenium-TestAutomation-Financeme'
8             }
9         }
10        stage("Executing jar"){
11            steps{
12                sh 'sudo java -jar seleniumAutomationFinanceme.jar'
13            }
14        }
15    }
16}
17

```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

[Save](#) [Apply](#)

REST API Jenkins 2.401.1

**Jenkins**

Dashboard > Financeme-selenium-test-automate >

**Pipeline Financeme-selenium-test-automate**

Status Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax

**Stage View**

Test automation Code checkout	Executing jar
3s	8s

Average stage times: (Average full run time: ~12s)

Jun 24 16:12 No Changes

**Permalinks**

- Last build (#2), 21 hr ago

Build History trend

#2 Jun 24, 2023, 10:42 AM

#1 Jun 24, 2023, 9:52 AM

Dashboard > Financeme-selenium-test-automate > #2

```

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 56670
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 10:43:04 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Jun 24, 2023 10:43:04 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 10:43:04 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Unable to find CDP implementation matching 114.
Jun 24, 2023 10:43:04 AM org.openqa.selenium.chromium.ChromiumDriver lambda$new$3
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 'org.seleniumhq.selenium:selenium-devtools-v86:4.0.0' where the version ("v86") matches the version of the chromium-based browser you're using and 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

```

REST API Jenkins 2.401.1

Dashboard > Finance-web application > #10

```

TASK [updating apt] ****
changed: [172.31.3.8]

TASK [Install Docker] ****
changed: [172.31.3.8]

TASK [Start Docker Service] ****
changed: [172.31.3.8]

TASK [Deploy Dockee Container] ****
changed: [172.31.3.8]

PLAY RECAP ****
172.31.3.8 : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline]
[Pipeline] // script
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Triggering a new build of Financeme-selenium-test-automation #4
Finished: SUCCESS

```

REST API Jenkins 2.401.1

Jenkins

Dashboard >

+ New item      Add description

People      All      +

Build History      Project Relationship      Check File Fingerprint      Manage Jenkins      My Views

**Build Queue (1)**

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁️	Finance-web application	23 min #9	1 hr 3 min #7	49 sec
✗	⛈️	Financeme-selenium-test-automation	N/A	22 min #3	18 sec
✗	🌦️	Insurance-web application	1 day 2 hr #43	9 hr 41 min #60	1 min 49 sec
✓	☀️	Insureme-selenium-test-automation	9 hr 40 min #2	N/A	12 sec

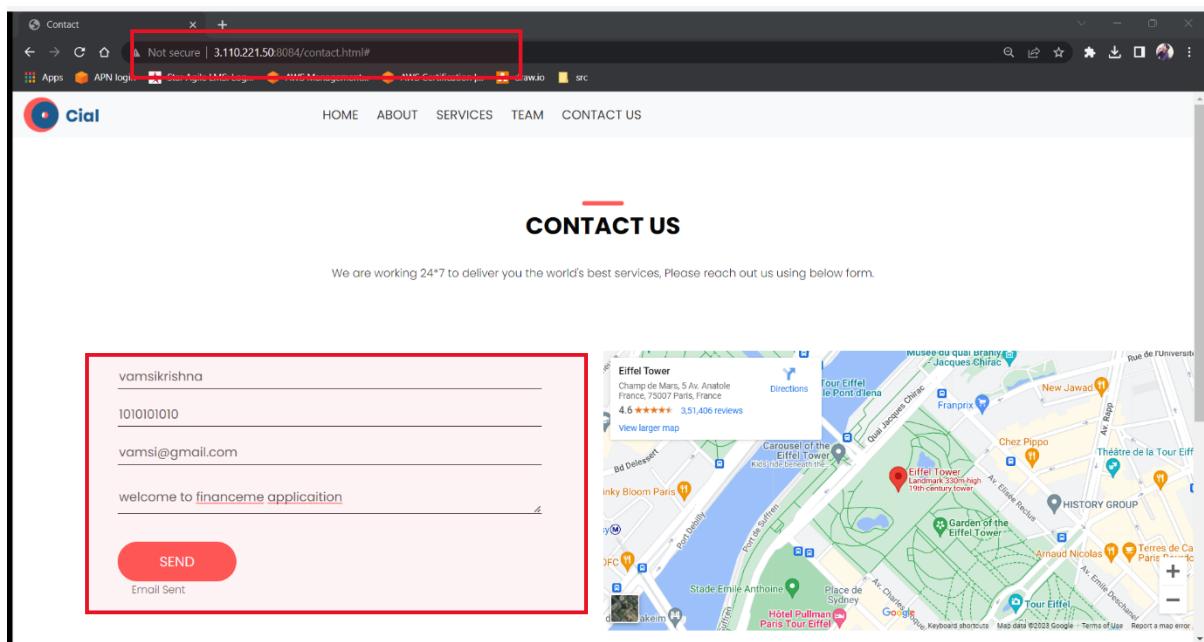
Icon: S M L      Icon legend      Atom feed for all      Atom feed for failures      Atom feed for just latest builds

Build Executor Status

1 Idle  
2 Idle

REST API Jenkins 2.401.1

Sample executed on local machine-



## Terraform –

Creating a EC2 instance with terraform in IAC as a production server and deploying the Finance application on the server.

Installing the **terraform** and **AWS config** on the instance ( Capstone\_Project\_Development )

- aws cli - <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>
- terraform - <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>

```

aws Services Search [Alt+S]
root@ip-172-31-36-251:~# aws --version
aws-cli/2.12.3 Python/3.11.4 Linux/5.19.0-1027-aws exe/x86_64/ubuntu.22 prompt/off
root@ip-172-31-36-251:~# terraform --version
Terraform v1.5.1
on linux_amd64
root@ip-172-31-36-251:~#

```

## Configuring the AWS

Setup aws user :

IAM-> Create a use and attach administrator access privilege roles

And setup the Credentials keys

The screenshot shows the 'Access key best practices & alternatives' step in the AWS IAM Access Key creation wizard. It asks the user to choose a use case for the access key. The 'Command Line Interface (CLI)' option is selected, with a note explaining it's for enabling the AWS CLI to access the account. Other options include 'Local code', 'Application running on an AWS compute service', 'Third-party service', 'Application running outside AWS', and 'Other'. A warning section at the bottom lists recommended alternatives: using AWS CloudShell or AWS CLI V2 for authentication. A checkbox at the bottom allows the user to accept these recommendations and proceed.

**Access key best practices & alternatives**

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

Command Line Interface (CLI)  
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Application running outside AWS  
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

Other  
Your use case is not listed here.

**Alternatives recommended**

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

## Retrieve access keys

Access key	Secret access key
 AKIA6NZVGHJH2AFRCMW	 ***** <a href="#">Show</a>

## Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [Best practices for managing AWS access keys](#).

```
aws Services Q Search [Alt+S] Mumbai Vamsik
root@ip-172-31-36-251:~# aws configure
AWS Access Key ID [None]: AKI[REDACTED]Y2
AWS Secret Access Key [None]: KUL6[REDACTED]
Default region name [None]:
Default output format [None]:
root@ip-172-31-36-251:~# aws configure
AWS Access Key ID [*****7JY2]: ^C
root@ip-172-31-36-251:~#
```

Create a IAM role for EC2 service and tag the role to Capstone\_Project\_Development

**Instance summary for i-04b420ff6b0a3db72 (Capstone\_Project\_Development) [Info](#)**

Updated less than a minute ago

Instance ID	Public IPv4 address
i-04b420ff6b0a3db72 (Capstone_Project_Development)	3.129.14.36
IPv6 address	-
Hostname type	IP name
Answer private resource DNS name	IPv4 (A)
Auto-assigned IP address	3.129.14.36.70 [Public IP]
IAM Role	ec2-jenkins
IMDSv2	Optional
Subnet ID	subnet-0000000000000000

**Actions**

- Connect
- Instance state
- Actions

**Private IPv4 addresses**

- 172.31.36.251

**Public IPv4 DNS**

- ip-3.129.14.36.70.ec2-internal.us-east-1.amazonaws.com

**Change security groups** (highlighted)

- Manage instance state
- Instance settings
- Networking
- Security (highlighted)
- Get Windows password
- Image and templates
- Modify IAM role
- Monitor and troubleshoot

**Elastic IP addresses**

- 

**AWS Compute Optimizer finding**

- Opt-in to AWS Compute Optimizer for recommendations. | Learn more

**Auto Scaling Group name**

-

## Terraform Job creation-

Refer - <https://github.com/vamsikrishna918/Terraform-Ec2> for the script

The screenshot shows the Jenkins Pipeline configuration page for a job named "Terraform- prod". The "General" section includes a description "Terraform EC2 instance creation" and an "Enabled" toggle switch. The "Pipeline" section is selected, showing a Groovy script for the pipeline definition:

```
pipeline{  
    tools {  
        terraform 'Terraform'  
    }  
    stages{  
        stage("IAC Code checkout"){  
            steps{  
                echo "terraform code check out"  
                git branch: 'main', url: 'https://github.com/vamsikrishna918/Terraform-Ec2'  
            }  
        }  
        stage("Terraform initialization"){  
            steps{  
                echo " initialization provider"  
                sh 'terraform init'  
            }  
        }  
    }  
}
```

Below the script, the "Use Groovy Sandbox" checkbox is checked. At the bottom of the page are "Save" and "Apply" buttons.

```
pipeline{  
    agent any  
    tools {  
        terraform 'Terraform'  
    }  
}
```

```
stages{

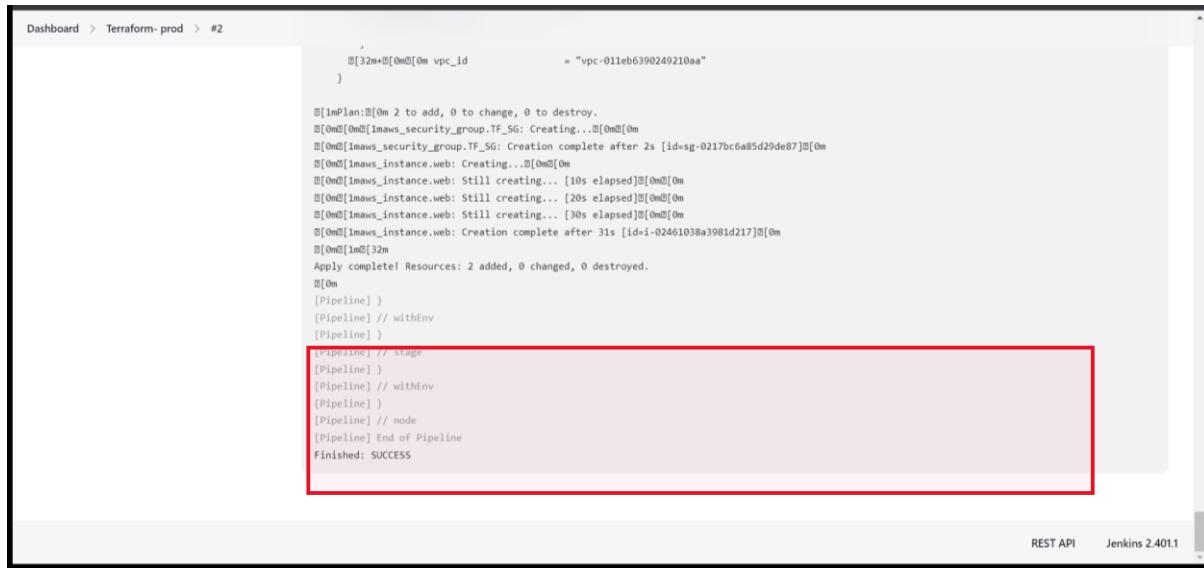
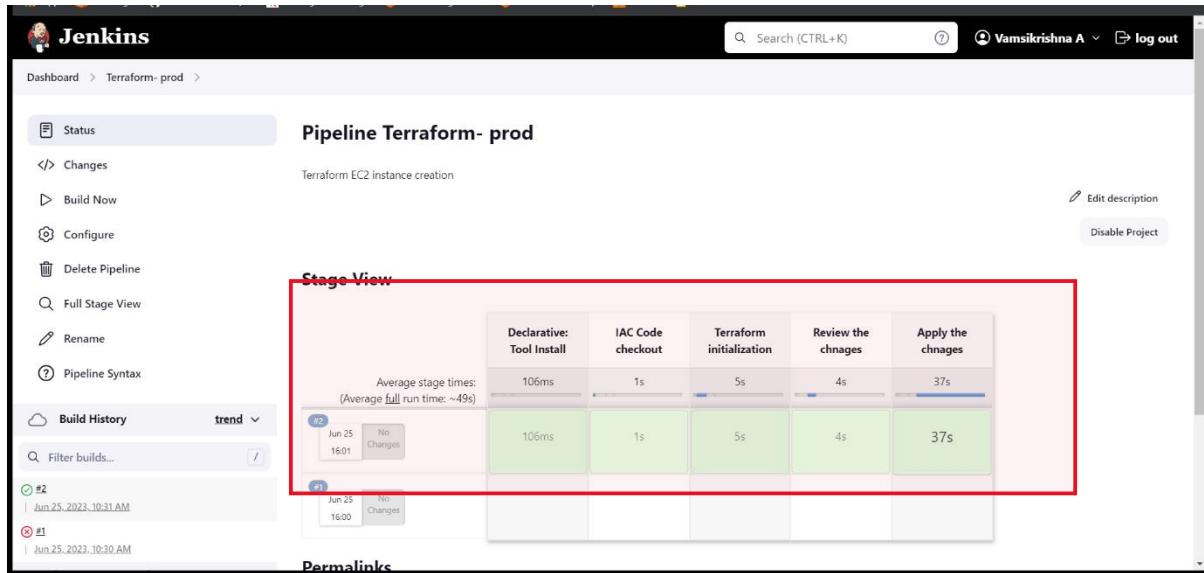
    stage("IAC Code checkout"){
        steps{
            echo "terraform code check out"
            git branch: 'main', url: 'https://github.com/vamsikrishna918/Terraform-Ec2'
        }
    }

    stage("Terraform initialization"){
        steps{
            echo " initialization provider"
            sh 'terraform init'
        }
    }

    stage("Review the chnages"){
        steps{
            echo " Review the infrastructure changes"
            sh 'terraform plan'
        }
    }

    stage("Apply the chnages"){
        steps{
            echo " Creating the infrastructure"
            sh 'terraform apply -auto-approve'
        }
    }

}
```

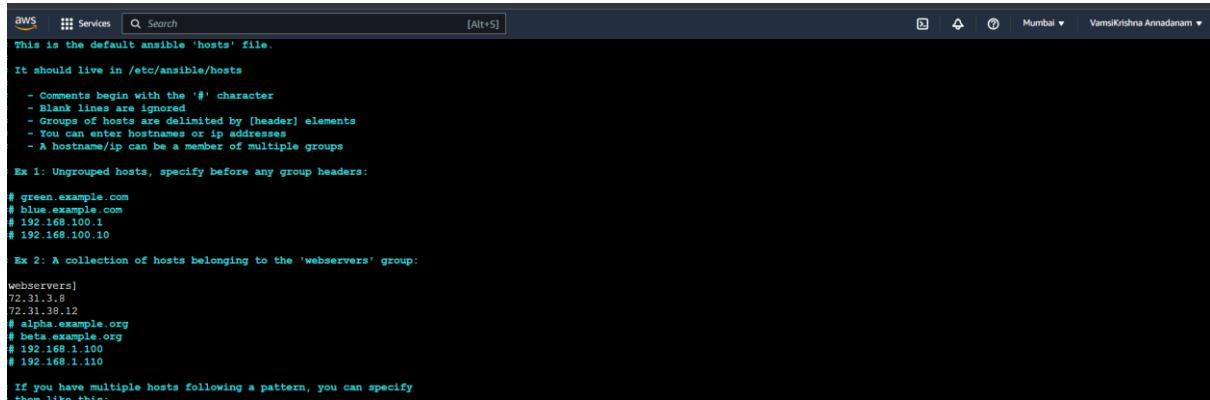


The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. The main content area displays a table titled 'Instances (5) Info' with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. The instances listed are: Capstone\_Project\_Development (Running, t2.medium, 2/2 checks passed, No alarms, ap-south-1a, ec2-3-109-62-70.ap-s...), Automated\_prod\_server (Terminated, t2.micro, -), Automated\_prod\_server (Terminated, t2.micro, -), Deployment\_Test\_Server (Running, t2.micro, 2/2 checks passed, No alarms, ap-south-1a, ec2-35-154-130-96.ap-s...), and Automated\_prod\_server\_instance (Running, t2.micro, Initializing, No alarms, ap-south-1a, ec2-35-154-130-96.ap-s...). A red box highlights the 'Deployment\_Test\_Server' row.

The screenshot shows the AWS EC2 Security Group details page for the 'Deployment\_Test\_Server'. The left sidebar is identical to the previous screenshot. The main content area is divided into sections: Security details (IAM Role: -, Owner ID: 546387079634, Launch time: Sun Jun 25 2023 16:01:23 GMT+0530 (India Standard Time)), Inbound rules (listing six rules with names like sgr-087600238600c3fc8, port ranges All/TCP, and security groups 'security group using Terraform'), and Outbound rules (listing two rules with names like sgr-067956a5f4c350fb0, port ranges All/All, and security groups 'security group using Terraform').

## Setting up the Hosts

Insert the private ip of the automated\_prod\_server in host file on the (Capstone deployment server) as below



```
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

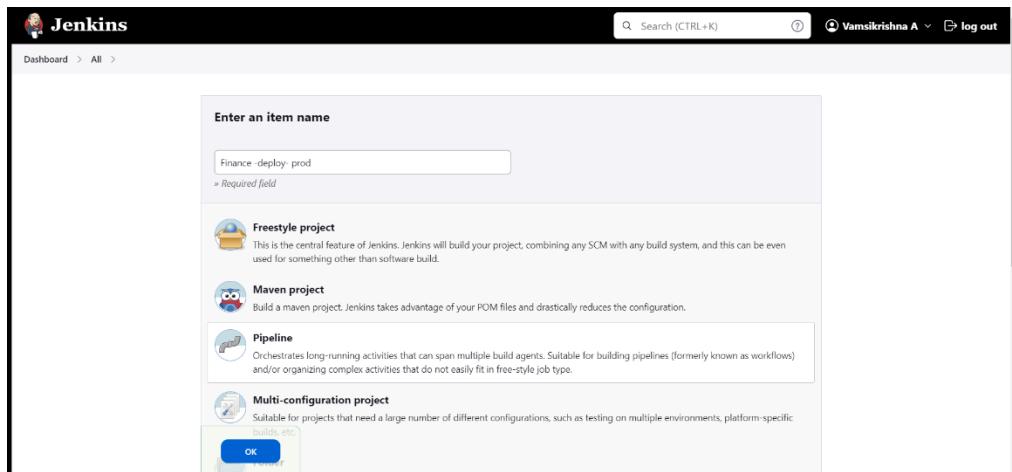
Ex 2: A collection of hosts belonging to the 'webservers' group:

[webservers]
72.31.3.8
72.31.3.12
#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:
```

```
root@ip-172-31-36-251:~# ls /etc/ansible
ansible.cfg  hosts  roles
root@ip-172-31-36-251:~# cd /etc/ansible/
root@ip-172-31-36-251:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ip-172-31-36-251:/etc/ansible# vi hosts
root@ip-172-31-36-251:/etc/ansible# cat hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# -- Comments begin with the '#' character
```

### Finance me application deploy to automated prod server-



Build trigger setup – job builds after the automated instance creation

Dashboard > Finance -deploy- prod > Configuration

**Configure**

**General**

**Advanced Project Options**

**Pipeline**

**build triggers**

Build after other projects are built ?

Projects to watch

Terraform- prod.

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Always trigger, even if the build is aborted

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Quiet period ?

Trigger builds remotely (e.g., from scripts) ?

**Save** **Apply**

This screenshot shows the Jenkins Pipeline configuration page under the 'Configuration' tab for the 'Finance -deploy- prod' project. The 'Pipeline' tab is selected. The 'Triggers' section is displayed, containing various options for triggering builds. The 'Trigger only if build is stable' option is selected. Other options like 'Build periodically' and 'GitHub hook trigger for GITScm polling' are also listed. At the bottom are 'Save' and 'Apply' buttons.

Dashboard > Finance -deploy- prod > Configuration

**Configure**

**General**

**Advanced Project Options**

**Pipeline**

**Definition**

Pipeline script

**Script** ?

```
1 pipeline{
2     agent any
3
4     stages{
5         stage("Code checkout"){
6             steps{
7                 git branch: 'main', url: 'https://github.com/vamsikrishna918/Finance-Web-Application-E2E'
8             }
9         }
10        stage("Deploying to Automated PROD server"){
11            steps{
12                script {
13                    echo "****Deploying Application to prod server****"
14                    ansiblePlaybook become: true, credentialsId: 'ansiblecreds', disableHostKeyChecking: true, inventory: '/etc/ansible/hosts'
15                }
16            }
17        }
18    }
19}
```

Use Groovy Sandbox ?

**Pipeline Syntax**

**Save** **Apply**

REST API Jenkins 2.401.1

This screenshot shows the Jenkins Pipeline configuration page under the 'Configuration' tab for the 'Finance -deploy- prod' project. The 'Pipeline' tab is selected. The 'Definition' section is displayed, showing a Groovy script for the pipeline. The script defines a pipeline with two stages: 'Code checkout' and 'Deploying to Automated PROD server'. It uses the 'git' step in the first stage and the 'script' step in the second stage to run an Ansible playbook. A 'Use Groovy Sandbox' checkbox is checked. At the bottom are 'Save' and 'Apply' buttons, and status information 'REST API Jenkins 2.401.1'.

Jenkins

Dashboard > Finance -deploy- prod >

**Pipeline Finance -deploy- prod**

Status Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax

Add description Disable Project

**Stage View**

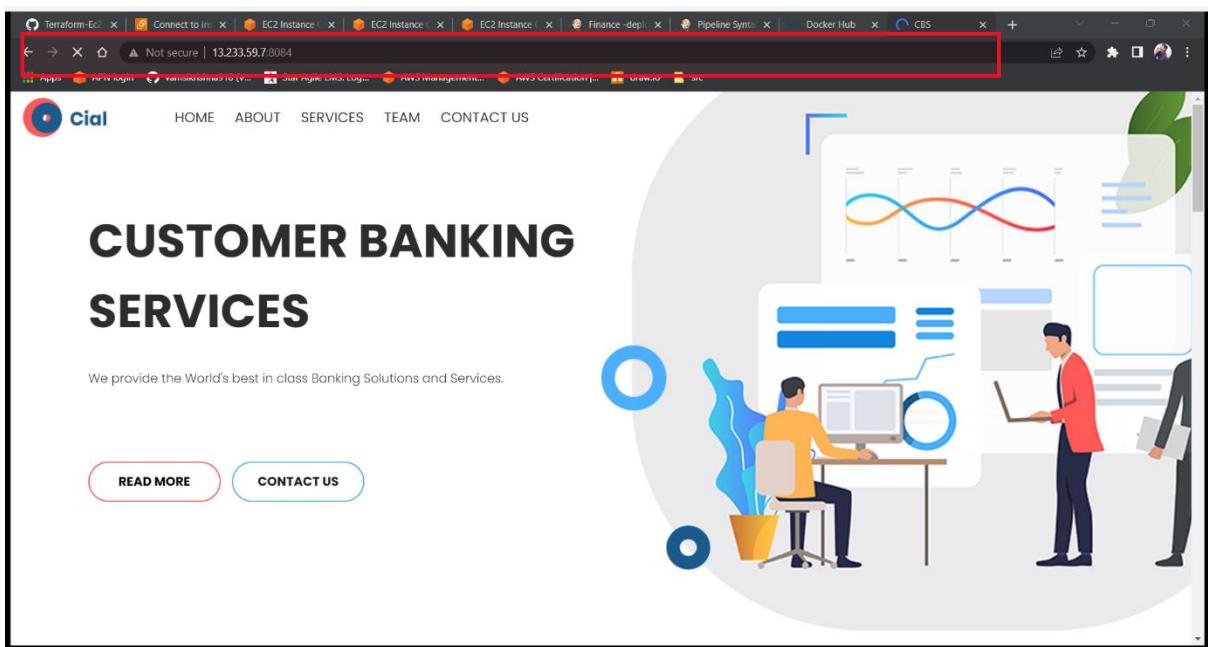
Code checkout	Deploying to Automated PROD server
Average stage times: (Average full run time: ~1min 12s)	802ms 18s
#8 Jun 25 17:20 No Changes	697ms
	1min 11s
#7 Jun 25 17:18 No Changes	695ms
	9s

Build History trend Filter builds... #8 | Jun 25, 2023, 11:50 AM

Dashboard > Finance -deploy- prod > #8

```
-----  
TASK [updating apt] ****  
changed: [13.233.59.7]  
  
TASK [Install Docker] ****  
changed: [13.233.59.7]  
  
TASK [Start Docker Service] ****  
changed: [13.233.59.7]  
  
TASK [Deploy Docker Container] ****  
changed: [13.233.59.7]  
  
PLAY RECAP ****  
13.233.59.7 : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
[Pipeline] )  
[Pipeline] // script  
[Pipeline] )  
[Pipeline] // stage  
[Pipeline] )  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

REST API Jenkins 2.401.1



## Monitoring the servers-

Creating a instance to monitor both test and prod server.

The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Capstone_Project_Development	i-04b420ff6b0a3db72	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1a	ec2-3-109-62-70.ap-so...
Automated_prod_server_Instance	i-08d66022137d50d90	Terminated	t2.micro	-	No alarms	ap-south-1a	-
Monitoring instance	i-07ccb1c815ba09ed	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-13-233-93-7.ap-so...
Deployment Test Server	i-0d8575ha10f33a697	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1b	ec2-13-233-98-215.ap-...

Installing Prometheus –

**Command – apt update && apt install Prometheus**

Default port of Prometheus – 9090

```
root@ip-172-31-37-231:~# service prometheus status
● prometheus.service - Monitoring system and time series database
  Loaded: loaded (/lib/systemd/system/prometheus.service; enabled; vendor preset: enabled)
  Active: active (running) since Sun 2023-06-25 12:56:59 UTC; 3min 43s ago
    Docs: https://prometheus.io/docs/introduction/overview/
      Main PID: 3601 (prometheus)
        Tasks: 7 (limit: 1141)
       Memory: 23.8M
          CPU: 439ms
         CGroup: /system.slice/prometheus.service
             └─ 3601 /usr/bin/prometheus

Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.882Z caller=head.go:481 level=info component=tedb msg="Replaying on-disk memory mappable chunks if any"
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.882Z caller=head.go:515 level=info component=tedb msg="On-disk memory mappable chunks replay completed" duration="0.000000s"
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.882Z caller=head.go:521 level=info component=tedb msg="Replaying WAL, this may take a while"
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.882Z caller=head.go:592 level=info component=tedb msg="WAL segment loaded" segment=0 maxSegment=0
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.883Z caller=head.go:598 level=info component=tedb msg="WAL replay completed" checkpoint_replay_duration=48.000000ms
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.884Z caller=main.go:850 level=info fs_type="EXT4 SUPER_MAGIC"
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.885Z caller=main.go:853 level=info msg="TSDB started"
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.885Z caller=main.go:980 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.892Z caller=main.go:1017 level=info msg="Completed loading of configuration file" filename=/etc/prometheus/
Jun 25 12:56:59 ip-172-31-37-231 prometheus[3601]: ts=2023-06-25T12:56:59.892Z caller=main.go:795 level=info msg="Server is ready to receive web requests."
tail -f /var/log/prometheus.log
```

The screenshot shows a browser window with the URL `65.0.130.101:9090/classic/targets` in the address bar. The page title is "Prometheus". The main content is titled "Targets". There are two sections: "node (1/1 up)" and "prometheus (1/1 up)". Each section has a table with columns: Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. In the "node" section, the endpoint is `http://localhost:9100/metrics`, state is UP, labels include `instance="localhost:9100"` and `job="node"`, last scrape was 5.995s ago, and scrape duration was 82.79ms. In the "prometheus" section, the endpoint is `http://localhost:9090/metrics`, state is UP, labels include `instance="localhost:9090"` and `job="prometheus"`, last scrape was 107ms ago, and scrape duration was 4.287ms.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<code>http://localhost:9100/metrics</code>	UP	<code>instance="localhost:9100"</code> <code>job="node"</code>	5.995s ago	82.79ms	

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<code>http://localhost:9090/metrics</code>	UP	<code>instance="localhost:9090"</code> <code>job="prometheus"</code>	107ms ago	4.287ms	

We need to monitor the test and prod servers,

Install **Node exporter** on both servers which acts as a agent to get the metrics

**Command** - apt install prometheus-node-exporter

i-0024fe4ac699063f2 (Automated\_prod\_server\_instance)  
PublicIPs: 13.233.52.7 PrivateIPs: 172.31.46.69

i-0d8575ba30f33e697 (Deployment\_Test\_Server)  
PublicIPs: 13.253.98.225 PrivateIPs: 172.51.3.8

On monitoring instance-

Check out the Prometheus directory

**Command** – cd /etc/Prometheus/

And edit the **.yml** file and insert the **ip's** which need to be monitored.

```
root@ip-172-31-37-231:/# cd /etc/prometheus/
root@ip-172-31-37-231:/etc/prometheus# ls
console_libraries  consoles  prometheus.yml
root@ip-172-31-37-231:/etc/prometheus# █
```

## Scrape : configs

- job name: any name

```
# If prometheus-node-exporter is installed, grab stats about the local
```

# machine by default.

`static_configs:`

- targets: ['IP(to be monitored):9100']

Restart the Prometheus –

Command- service Prometheus restart

The screenshot shows the Prometheus Targets page with three sections of targets:

- Automated\_prod\_server\_instance (1/1 up)**:
  - Endpoint: http://13.233.59.7:9100/metrics
  - State: UP
  - Labels: instance="13.233.59.7:9100", job="Automated\_prod\_server\_instance"
  - Last Scrape: 1.805s ago
  - Scrape Duration: 112.1ms
- Deployment\_Test\_Server (0/1 up)**:
  - Endpoint: http://13.233.98.225:9100/metrics
  - State: UNKNOWN
  - Labels: instance="13.233.98.225:9100", job="Deployment\_Test\_Server"
  - Last Scrape: Never
  - Scrape Duration: 0s
- node (1/1 up)**:
  - Endpoint: http://localhost:9100/metrics
  - State: UP
  - Labels: instance="localhost:9100", job="node"
  - Last Scrape: 4.265s ago
  - Scrape Duration: 87.2ms

**Visualization of metrics with Grafana server –**

## On monitoring instance-

Installation of Grafana- <https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/>

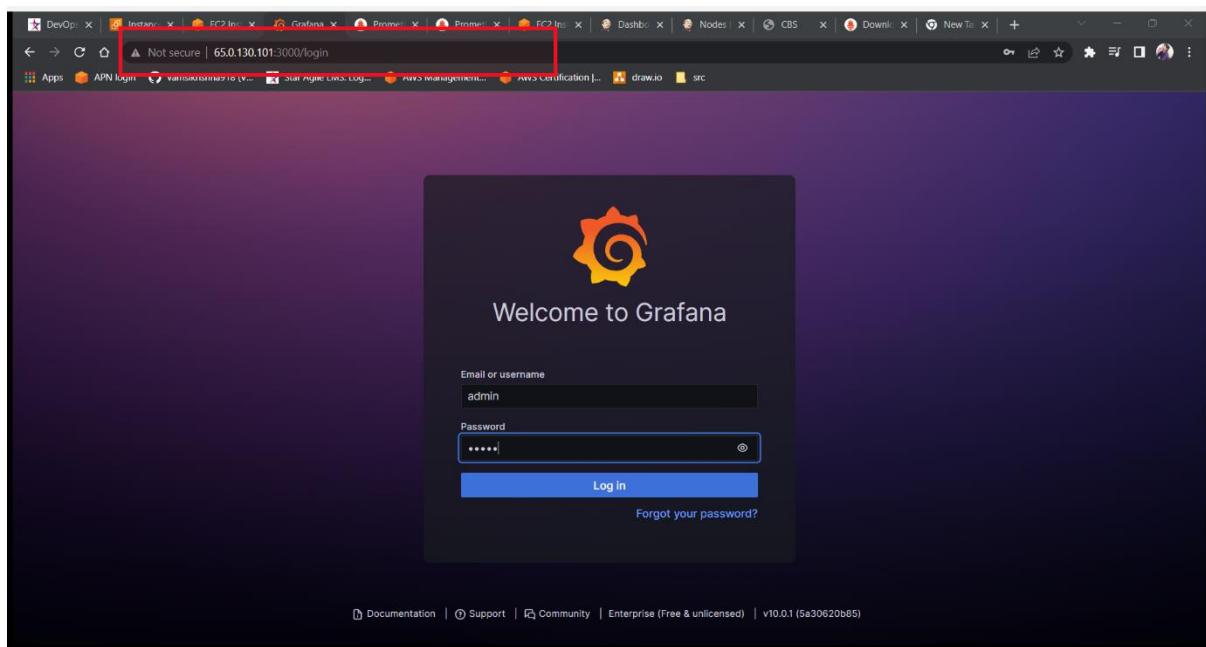
```
root@ip-172-31-37-231:/# grafana server --version
Version 10.0.1 (commit: 5a30620b85, branch: HEAD)
root@ip-172-31-37-231:/# service grafana server restart
grafana: unrecognized service
root@ip-172-31-37-231:/# service grafana-server restart
root@ip-172-31-37-231:/# 
```

i-07ccb1c815ba09ecd (Monitoring instance)

PublicIPs: 65.0.130.101 PrivateIPs: 172.31.37.231

Default port of Grafana – 3000

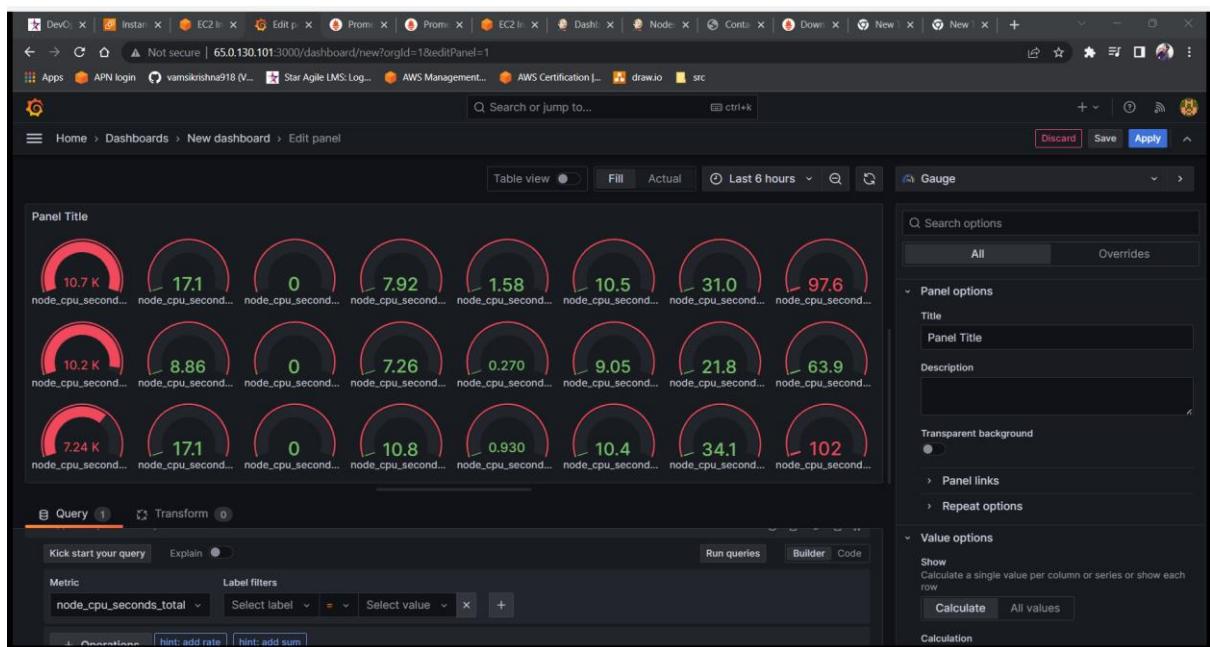
Default username and password - admin & admin



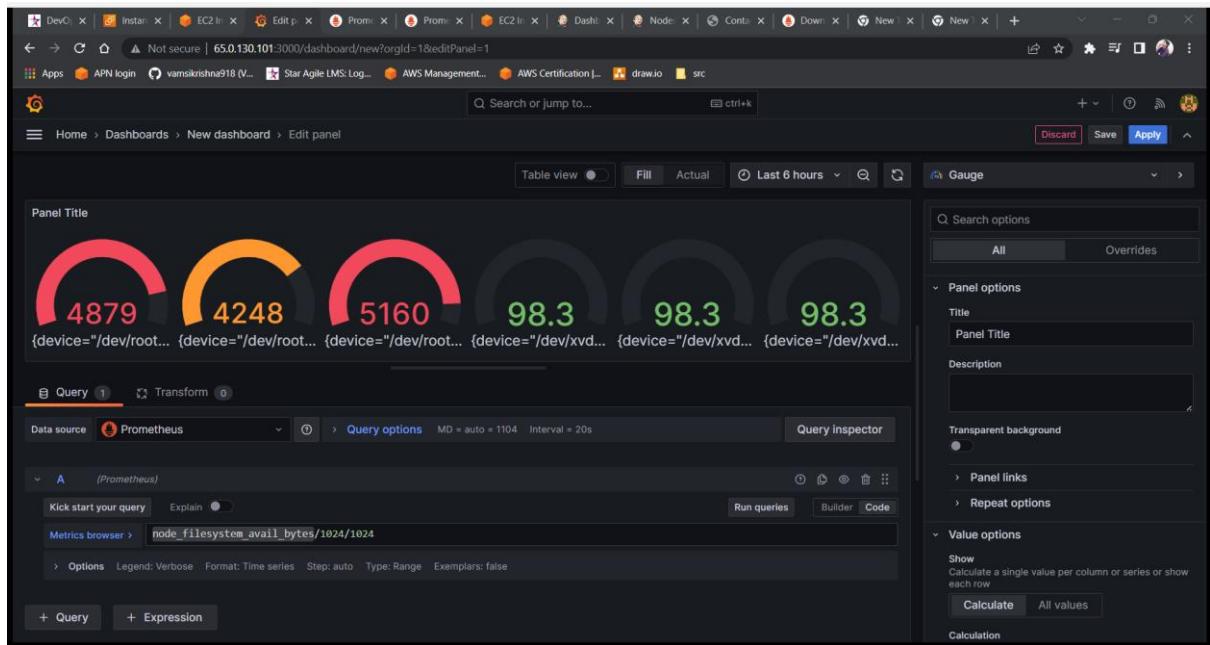
Add the data source:

The screenshot shows the Grafana administration interface. On the left, a sidebar titled 'Administration' has a 'Data sources' section selected. The main content area is titled 'Prometheus-2' and indicates it is a 'Prometheus' type data source. There are two tabs: 'Settings' (which is selected) and 'Dashboards'. A green badge says 'Alerting supported'. The 'Settings' tab contains several configuration sections: 'HTTP' (with fields for 'Prometheus server URL' set to 'http://65.0.130.101:9090/'), 'Auth' (with options for 'Basic auth', 'TLS Client Auth', and 'Skip TLS Verify'), and 'Metrics' (with a 'Metrics URL' field set to 'http://65.0.130.101:9090/metrics'). At the top right of the main content area are buttons for 'Build a dashboard' and 'Explore'.

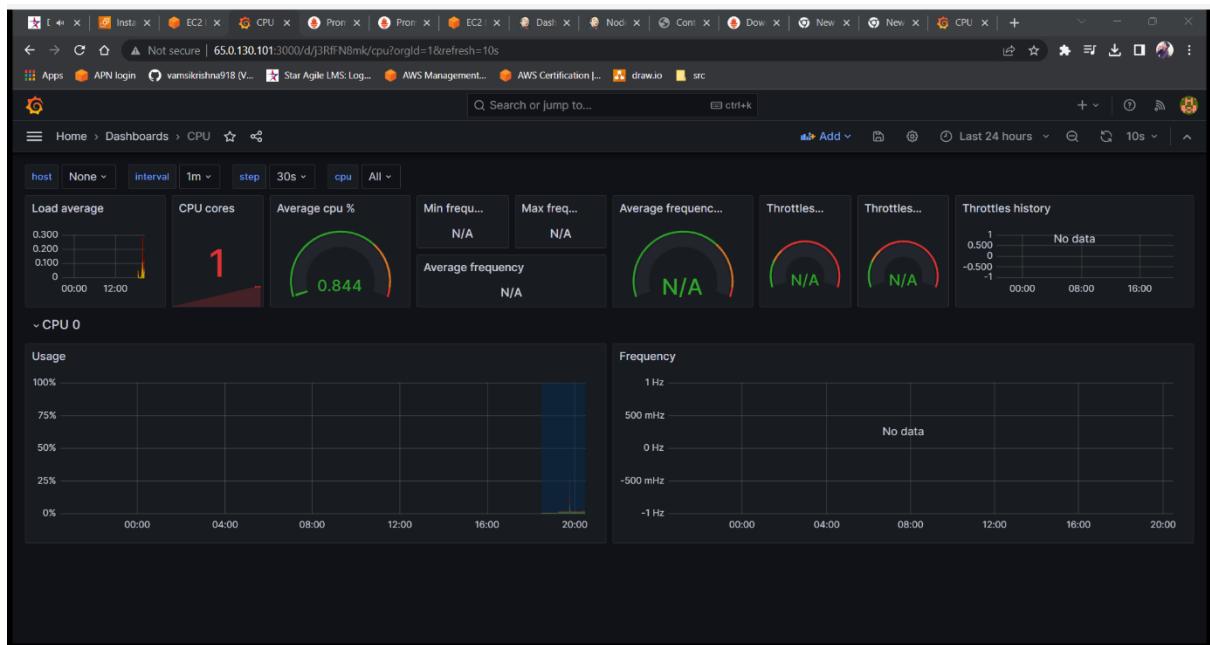
**CPU metrics – query ( Node\_cpu\_seconds\_total)**



## Disk Utilization – query (node\_filesystem\_avail\_bytes)



CPU dashboard import- <https://grafana.com/grafana/dashboards/9617-cpu/>



Memory metrics – <https://grafana.com/grafana/dashboards/3002-memory/>

