INSURANCE DOMAIN

Submission by : Pallavi
Submitted to : Vikul
Date of submission: 29-12-24

Life Insurance Project

The project involves creating a fully automated pipeline with Terraform, Jenkins, Docker, Ansible, Prometheus, and Grafana for monitoring.

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

Insure Me project code is attached below.

https://github.com/CSPPallavi/star-agile-insurance-project

Step 1: Create an EC2 Instance

Step 1: Launch two Ubuntu EC2 Instance on AWS as ansible controller and ansible worker node.

1. Log in to AWS Management Console:

o Access the AWS Management Console using below link and sign in with your credentials.

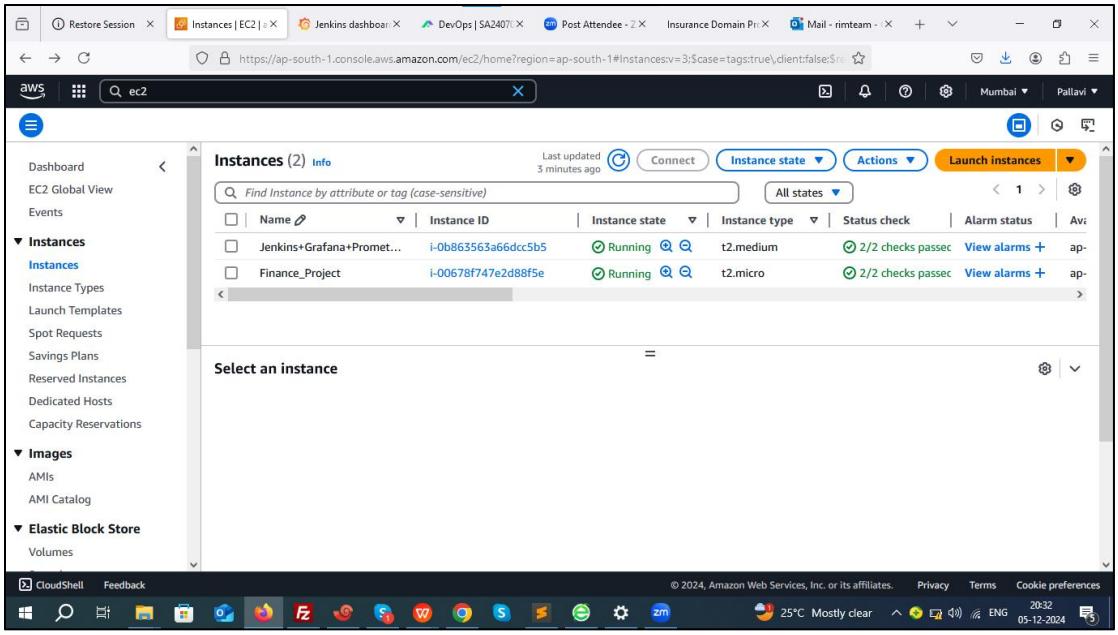
o https://aws.amazon.com/

2. Go to the EC2 Dashboard:

From the Services menu, search for "EC2" and select it to open the EC2 Dashboard.

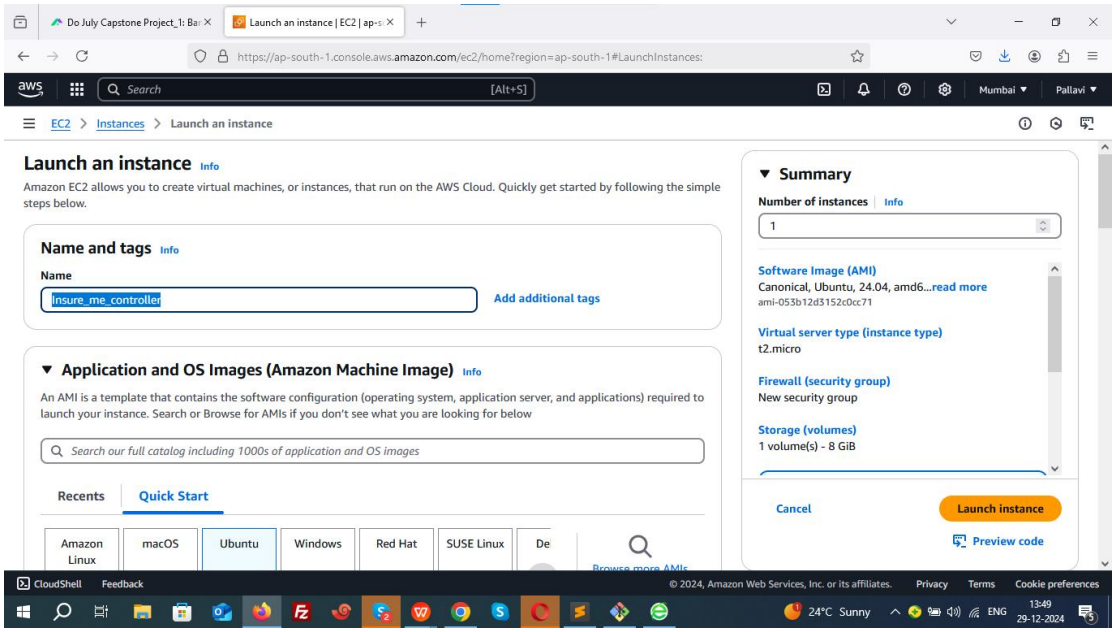
3. Start Instance Launch Process:

o Click on the instances on left side and "Launch Instance" button

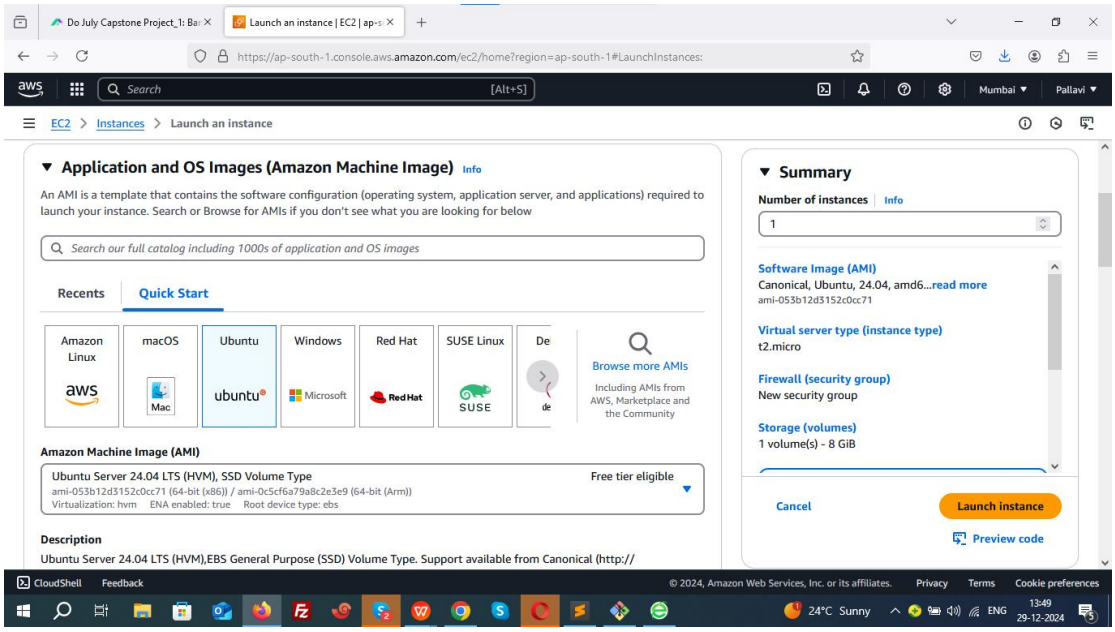


o **Name Your Instance:** Enter a name for the instance.

Insure_me_controller

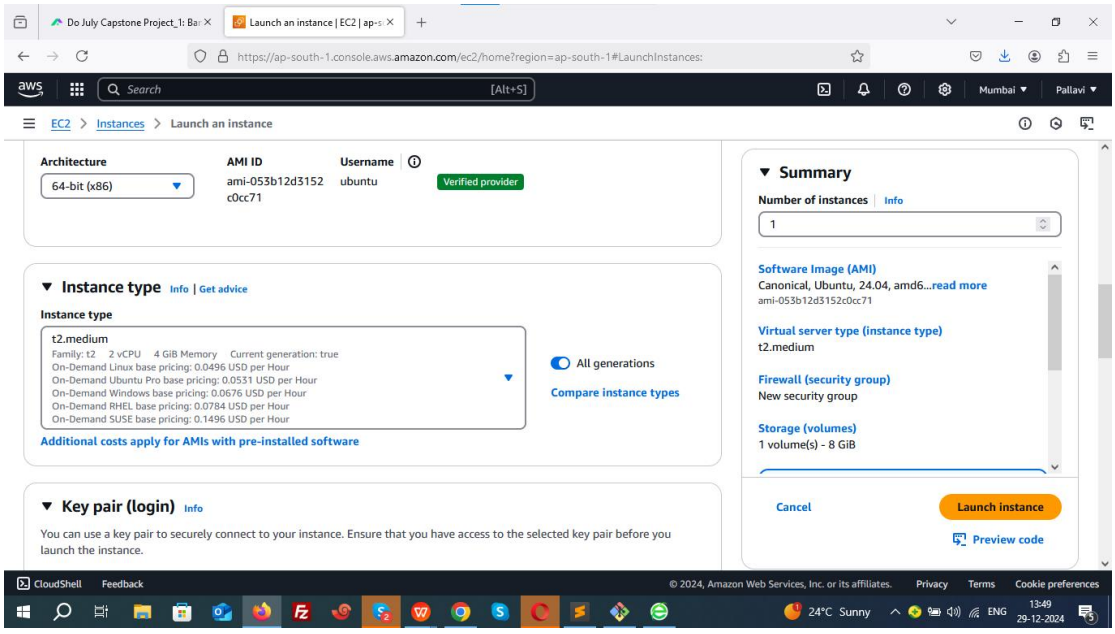


o **Choose an Amazon Machine Image (AMI):** Select an Ubuntu Server AMI

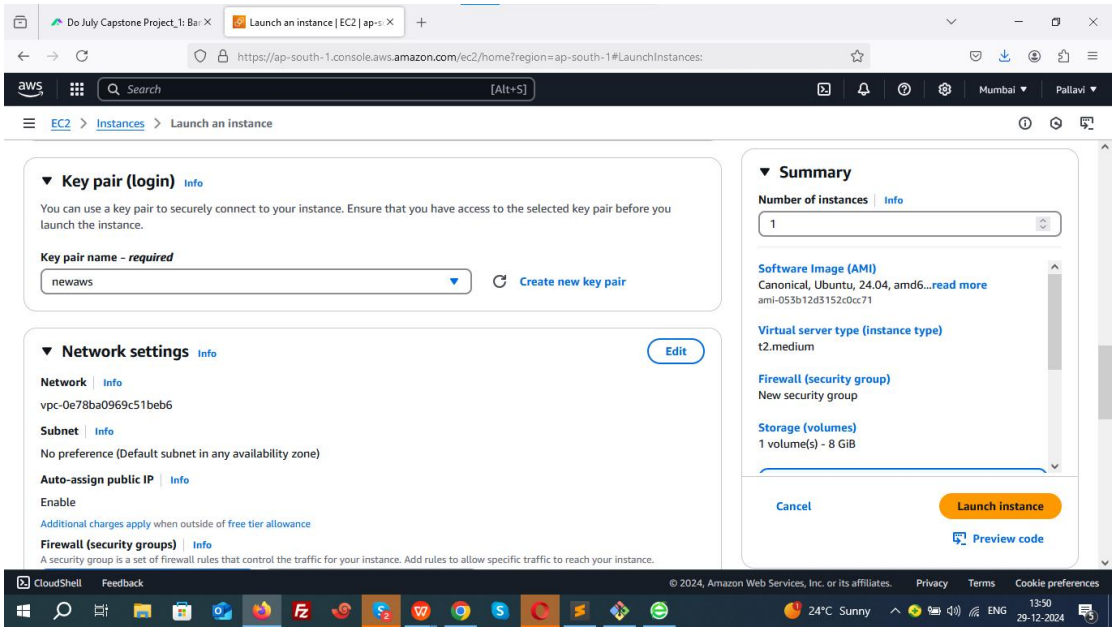


o **Select Instance Type:** Choose an appropriate instance type

t2.medium



o **Key Pair:** Choose an existing key pair or create a new one Click on create new key pair.Provide key pair name, select key pair type as RSA and private key format as .pem and click on create key pair.

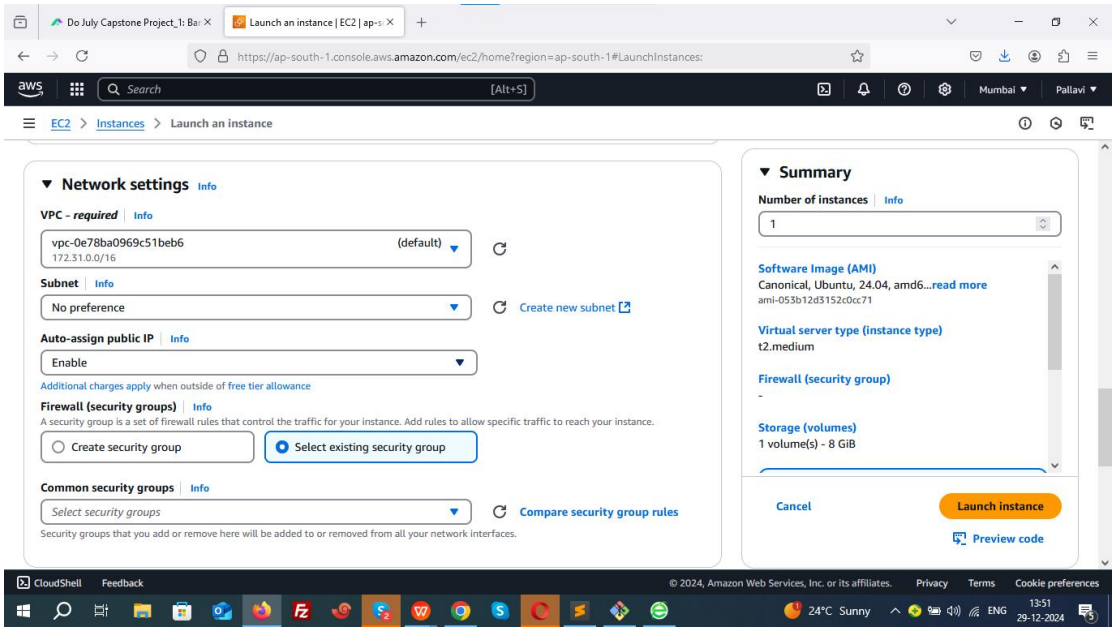


o **Configure Network Settings:**

Click on edit.

Use the default VPC settings.
Choose any one subnet

Ensure "Auto-assign Public IP" is enabled.



o **Create a New Security Group:**

Option Selection: Select the option "Create security group."

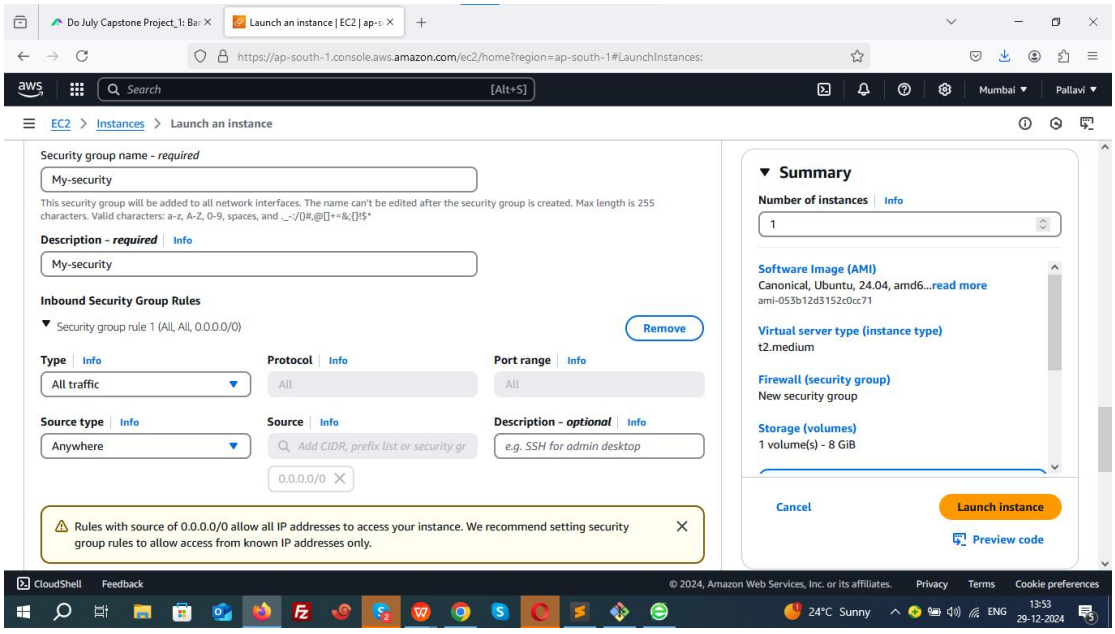
Security Group Name: Enter a name for your security group in the "Security group name" field

Description: Provide a brief description of the security group

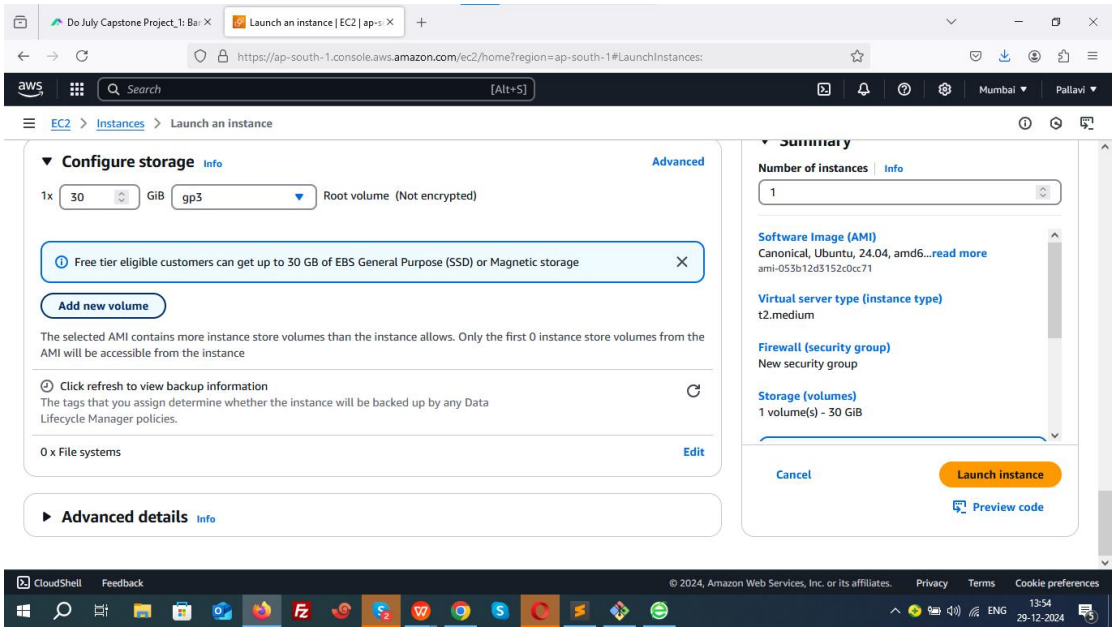
Configure Inbound Security Group Rules:

Rule 1

o **Type:** Select All traffic from the drop down menu.
o **Source Type:** Choose Anywhere to allow SSH access from any IP address.



o **Set Storage Options:** Adjust the storage size if necessary.

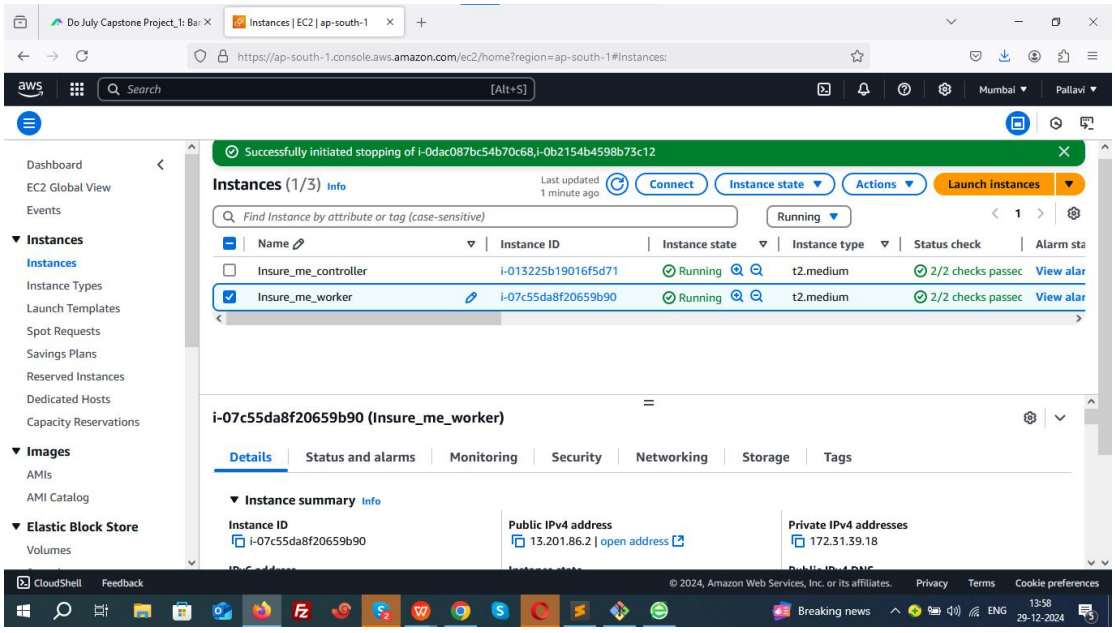


Review and Launch: Click "Launch Instance."

o **Wait for Instance Initialization:**

Monitor the instance until it’s in a "Running" state and passes the status checks.

Similary launch amother sever called Insure_me_worker

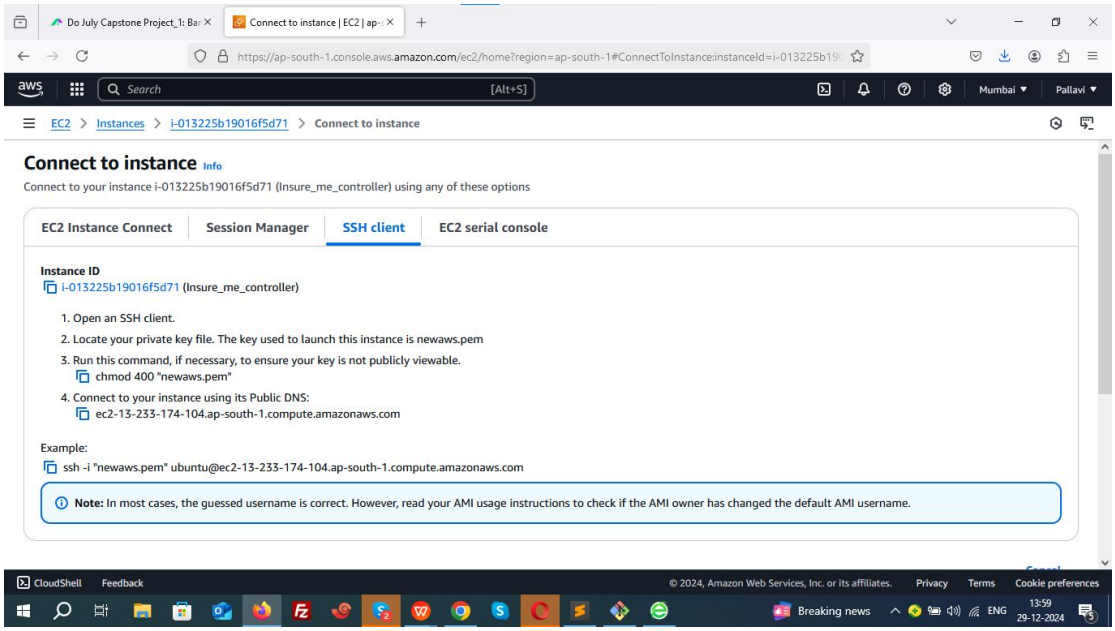


o **Connect to Your controller using git bash or any other.**

Select the instance and click on connect

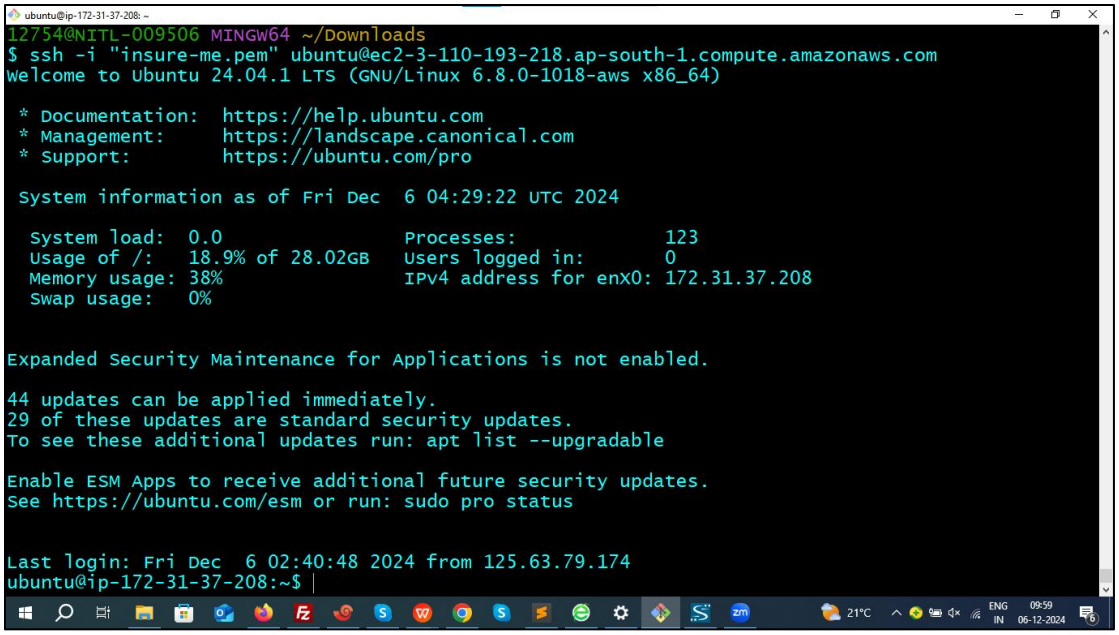
Select the tab SSH client and coppy ssh command.

ssh -i "insure-me.pem" ubuntu@ec2-3-110-193-218.ap-south-1.compute.amazonaws.com



Open git bash and go to the location where your pem key is stored.

Paste the command



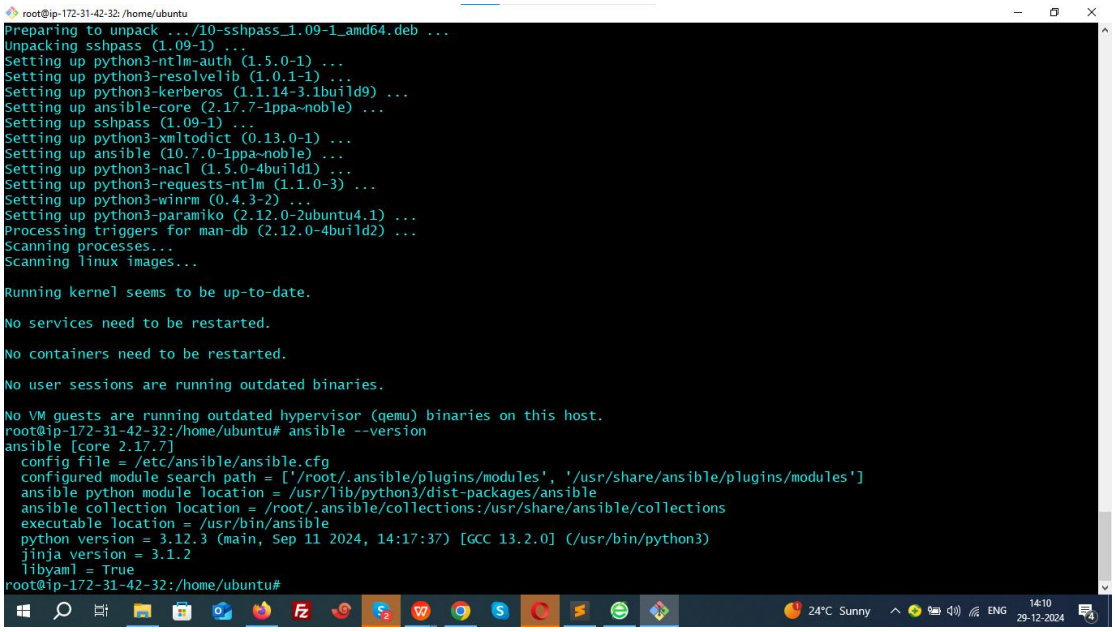
Step 2: Configure Ansible on the controller EC2 Instance

1. Connect to the Newly Created controller Instance:

Use SSH with the generated private key.

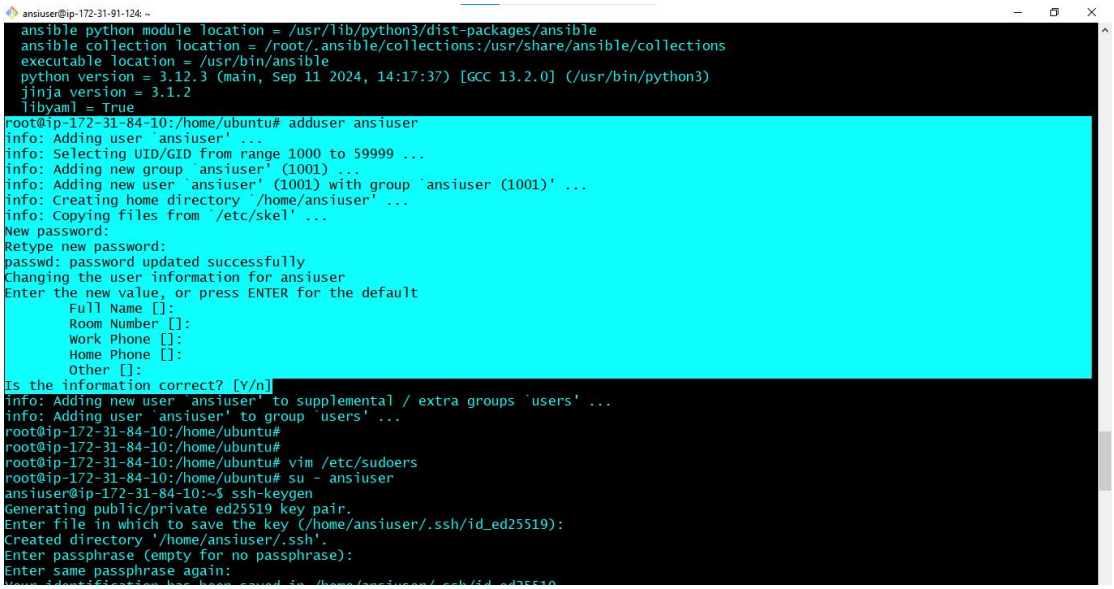
2. Ansible Installation

--> Install required dependencies
sudo apt-get install -y software-properties-common
--> Add the Ansible repository
sudo apt-add-repository ppa:ansible/ansible
--> Press [ENTER] when prompted to continue
--> Update the package index
sudo apt-get update
--> Install Ansible
sudo apt-get install -y ansible
--> Verify installation
ansible --version



2. Become a root user and create a New User (ansiuser)

--> Add a new user named ansiuser
sudo adduser ansiuser



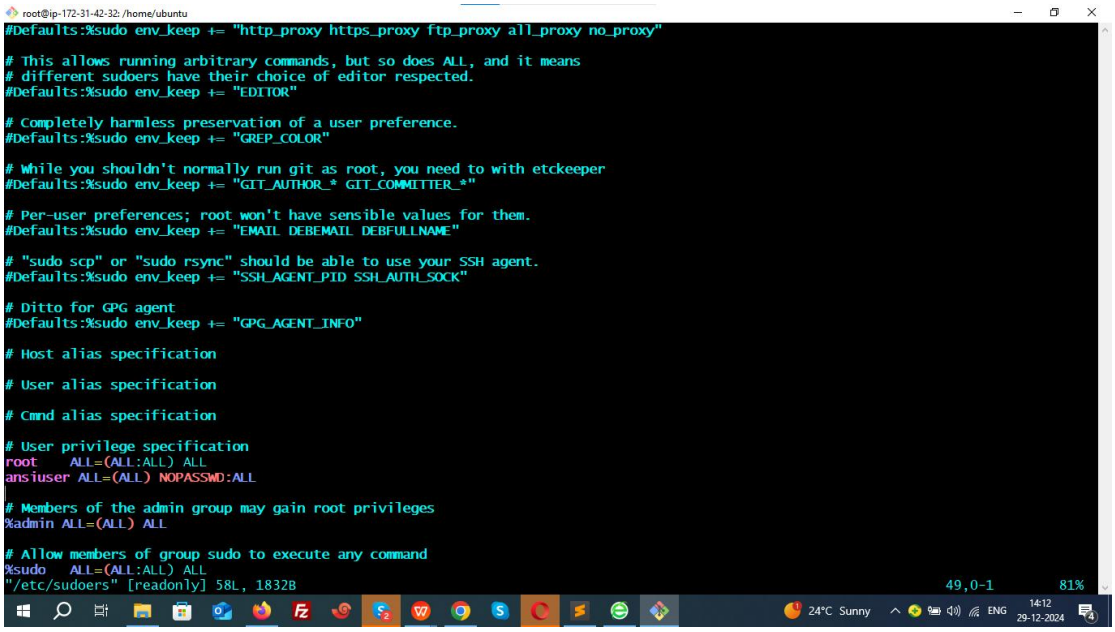
3. Grant Root Privileges to ansiuser

--> Edit the sudoers file
sudo vim /etc/sudoers

Add the following line at the end of the file:

plaintext

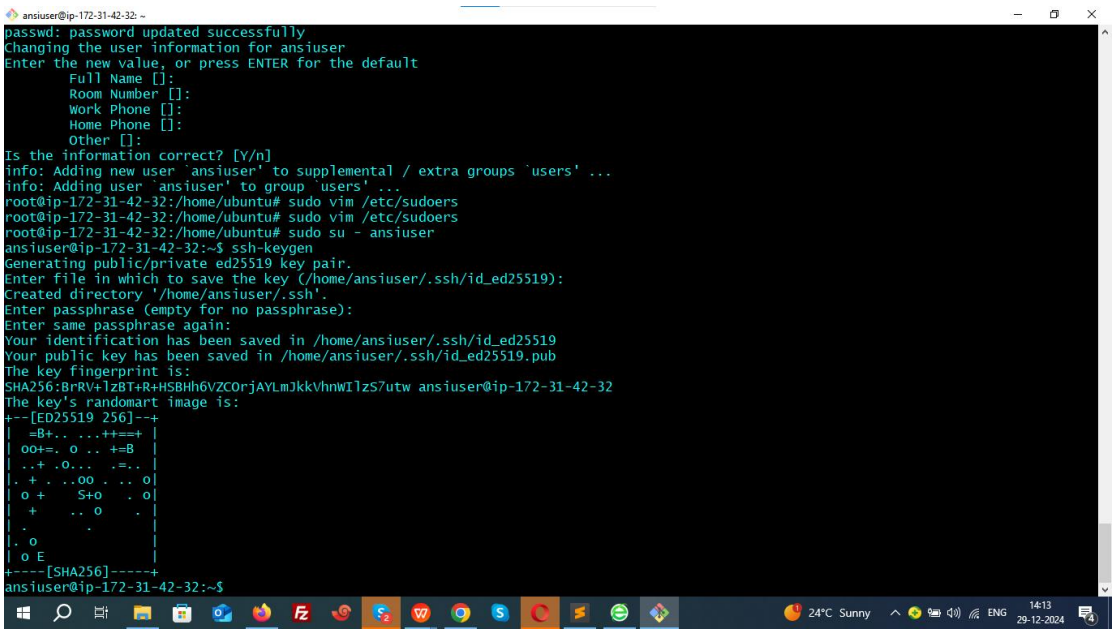
ansiuser ALL=(ALL) NOPASSWD:ALL



4. Switch to ansiuser and Generate SSH Key

--> Switch to the new user
sudo su - ansiuser

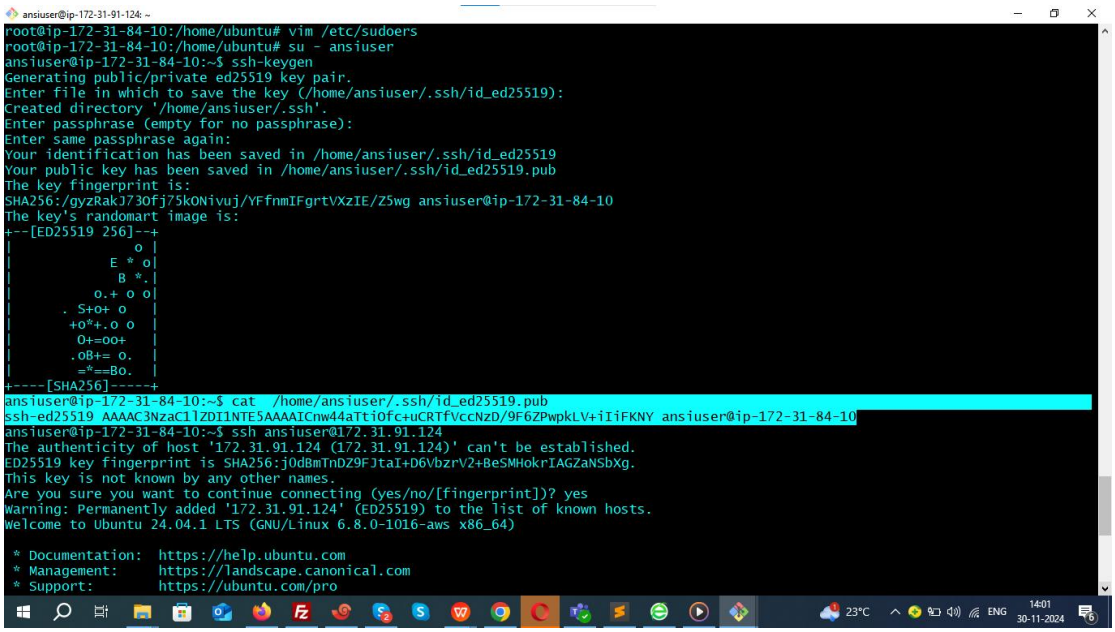
--> Generate an SSH key pair
ssh-keygen--> Press Enter for all prompts to create the key with default settings



--> Display the public key

cat /home/ansiuser/.ssh/id_ed25519.pub

Copy the output of the public key for use on the managed node.



On the worker node

1. Create the User (ansiuser**)**

--> Add a new user named ansiuser
sudo adduser ansiuser

2. Grant Root Privileges to ansiuser

--> Edit the sudoers file
sudo vim /etc/sudoers

Add the following line at the end of the file:

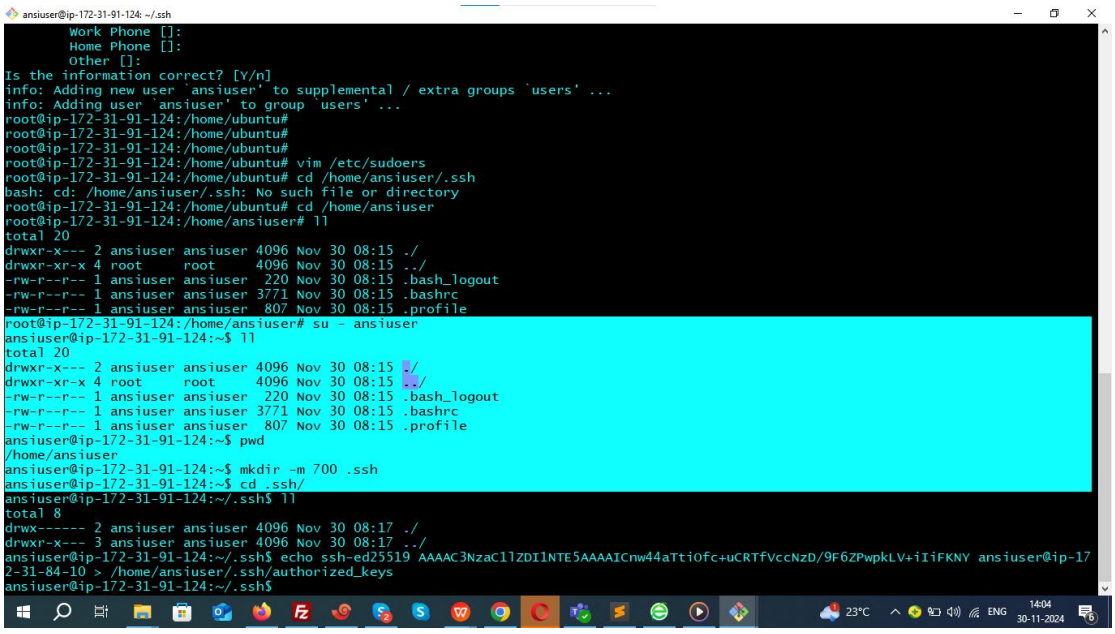
plaintext

ansiuser ALL=(ALL) NOPASSWD:ALL

3. Create .ssh **Directory and Authorize the Public Key**

--> Switch to the new user
sudo su - ansiuser

--> Create the .ssh directory with appropriate permissions
mkdir -m 700 /home/ansiuser/.ssh



--> Create the authorized_keys file and paste the public key

echo "<public_key>" > /home/<username>/.ssh/authorized_keys

echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICnw44aTtiOfc+uCRTfVccNzD/9F6ZPwpkLV+iIiFKNY ansiuser@ip-172-31-84-10" > /home/ansiuser/.ssh/authorized_keys

--> Set proper permissions for the authorized_keys file

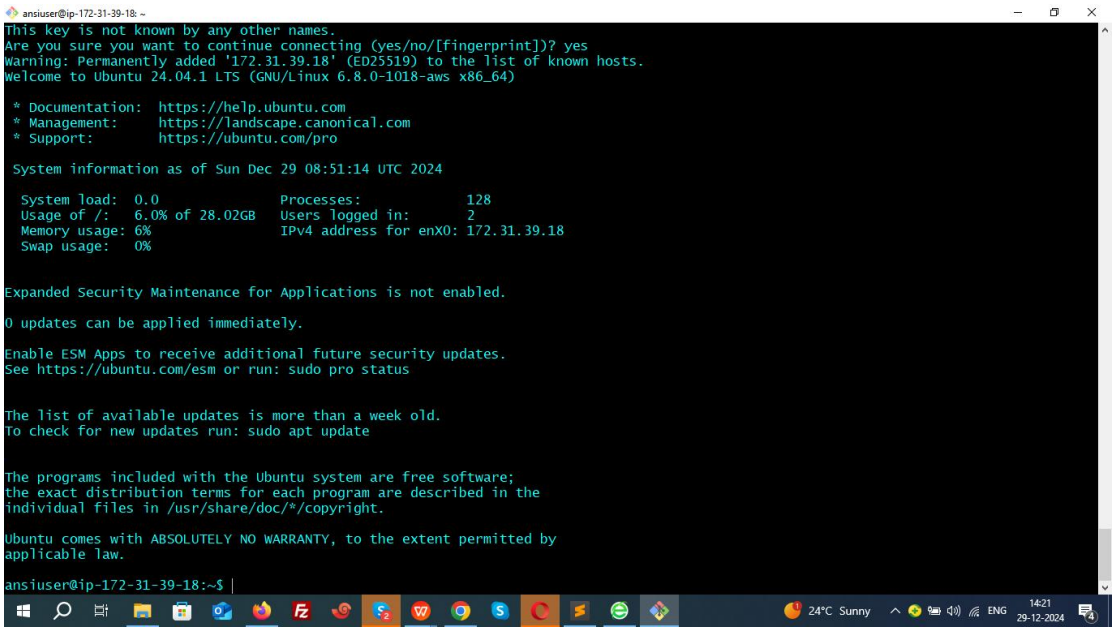
chmod 600 /home/ansiuser/.ssh/authorized_keys

Verify SSH Access

From the **controller node**, test SSH access to the managed node as ansiuser:

ssh ansiuser@<host_private_IP>

If configured correctly, you will be able to log in without a password.



Additional Steps for Ansible Inventory and Ping Test

By default the inventory file for ansible will be available at /etc/ansible/hosts directory

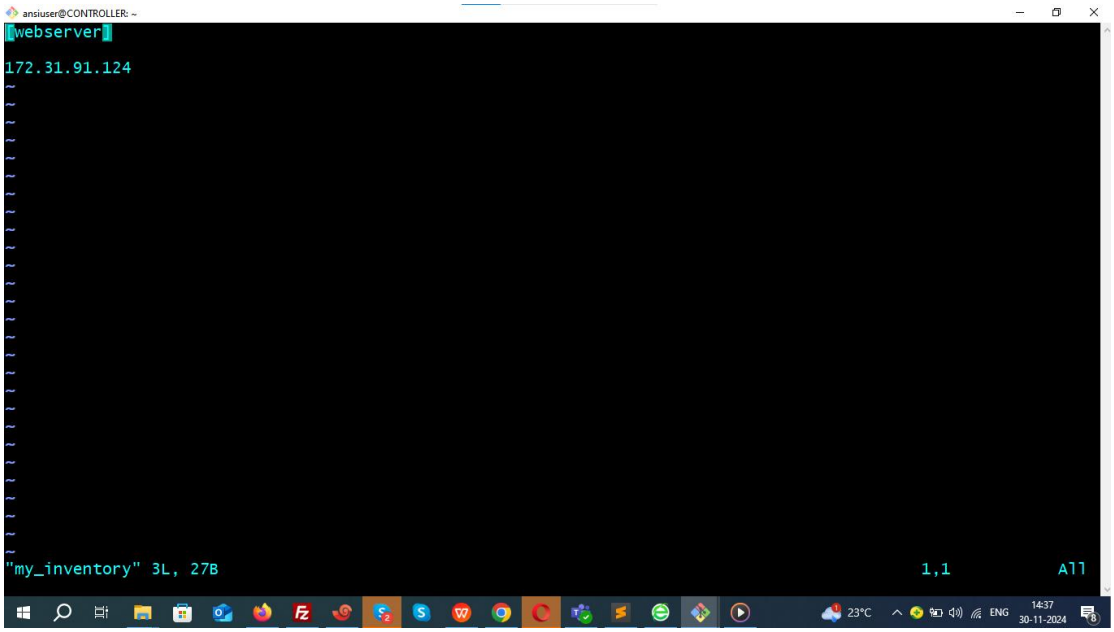
Edit this file as below.

vim hosts

Press i to insert below data

[webserver]

172.31.39.18

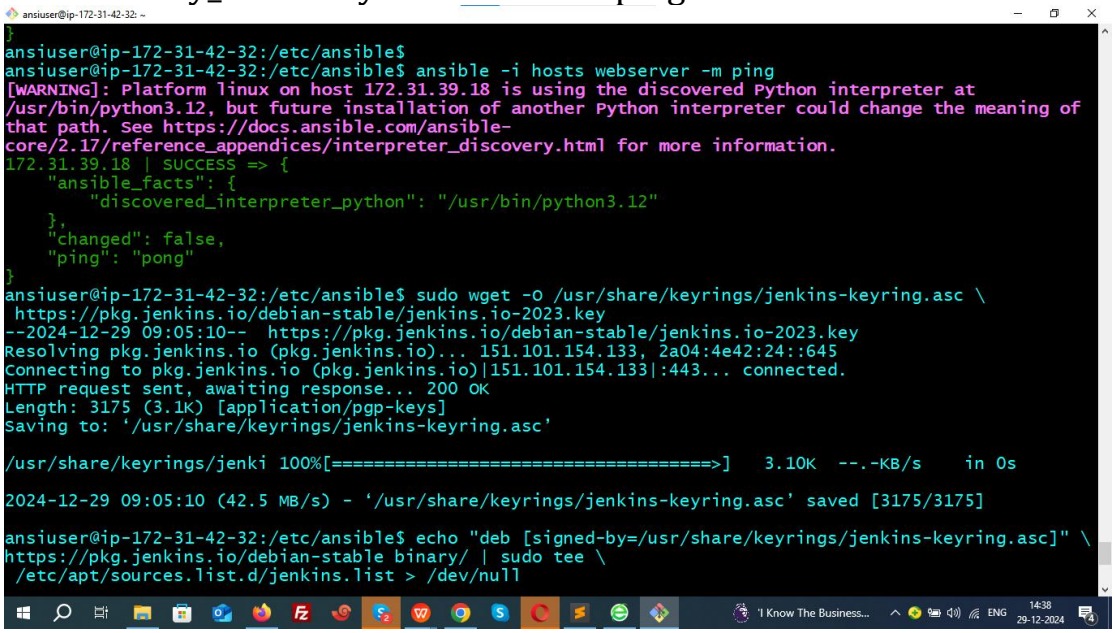


Save the file.

Test Ansible Connection:

Execute below Ad Hoc command

ansible -i <inventory_file_name> <target_host_group_name> -m <module_name>
ansible -i my_inventory webserver -m ping



- SUCCESS: Indicates the host was reachable.
- ping: "pong": Confirms successful communication.

Install Java , Jenkins, git and docker Using Ansible Playbook:

Add Jenkins key and repo manually.
ADD GPG KEY:
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

ADD Jenkins repository:
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]"
\https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null

Create an Ansible playbook:

vim insurance_me.yml

Add the following:

```
- name: Install and set up devops tools
  hosts: localhost
  become: true
  tasks:
  - name: Update the apt repo
    command: apt-get update
  - name: Install multiple packages
    package: name={{item}} state=present
    loop:
     - git
     - docker.io
     - openjdk-17-jdk
  - name: update apt-repo
    command: sudo apt-get update
```

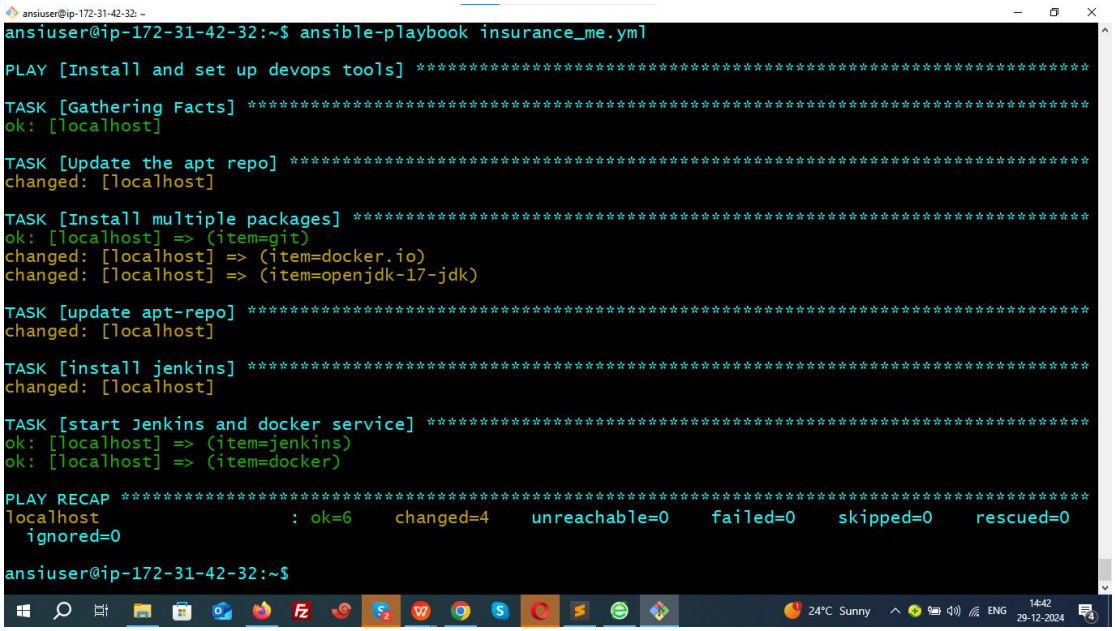
```
- name: install jenkins
  command: sudo apt-get install jenkins -y
- name: start Jenkins and docker service
  service: name={{item}} state=started
  loop:
  - jenkins
  - docker
```

Run the playbook:

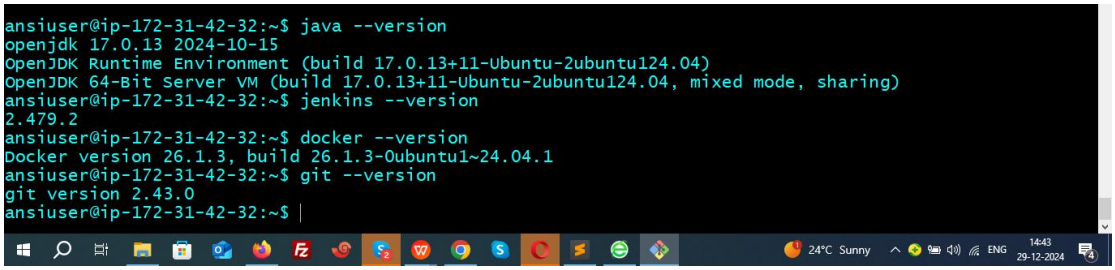
ansible-playbook insurance_me.yml

Run the playbook:

ansible-playbook finance_me.yml

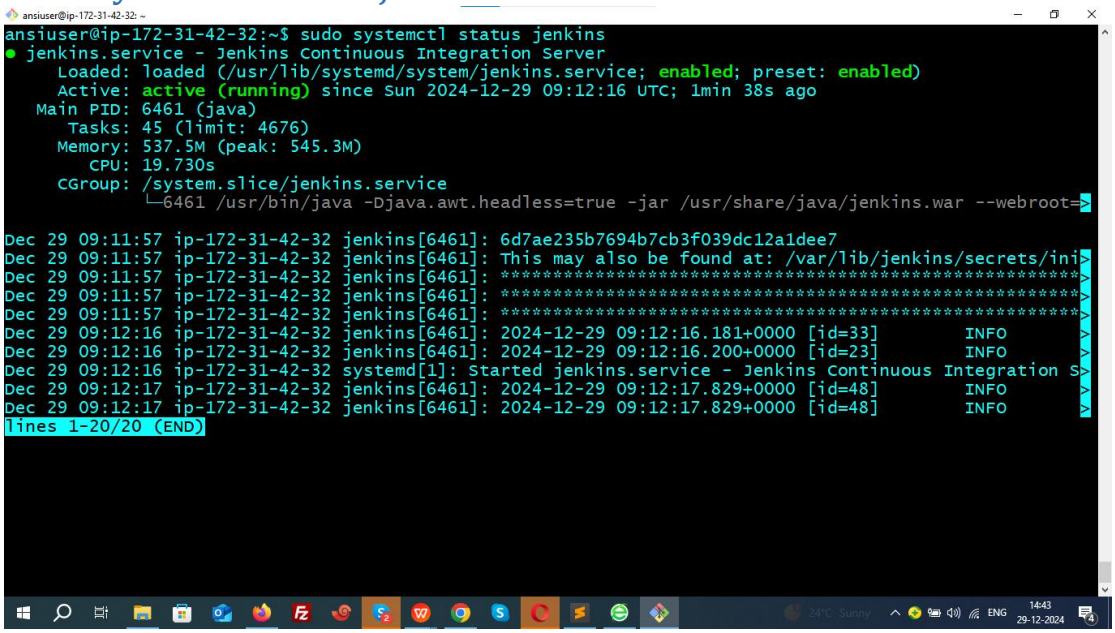


Verify installations.



Verify:

sudo systemctl status jenkins



3. Now access to the jenkins dashboard using below url:

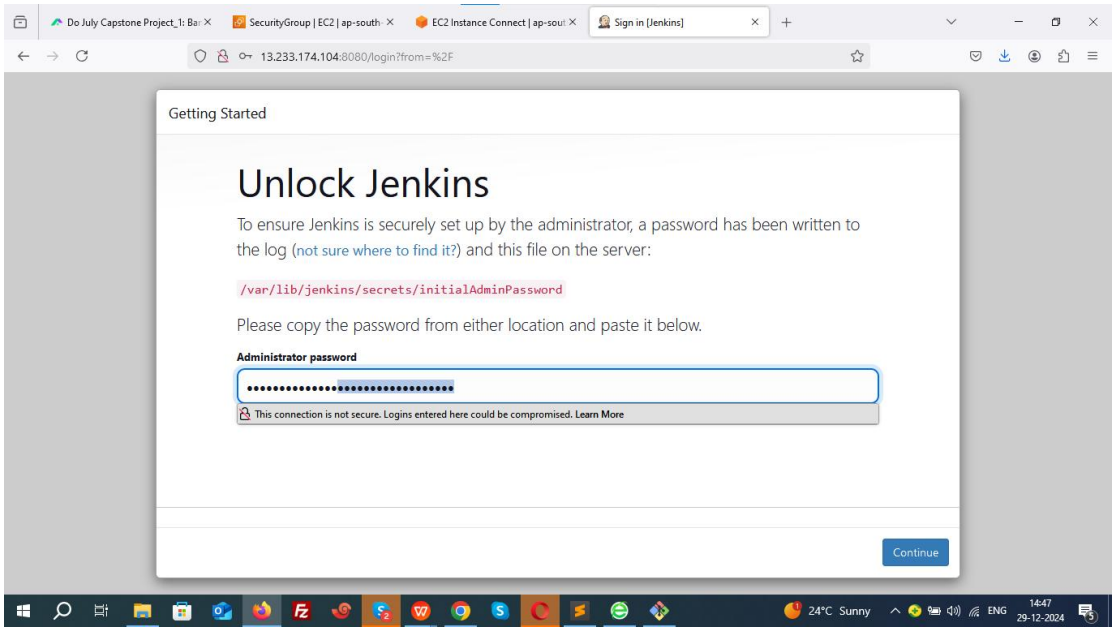
Enable port number 8080 on your controller

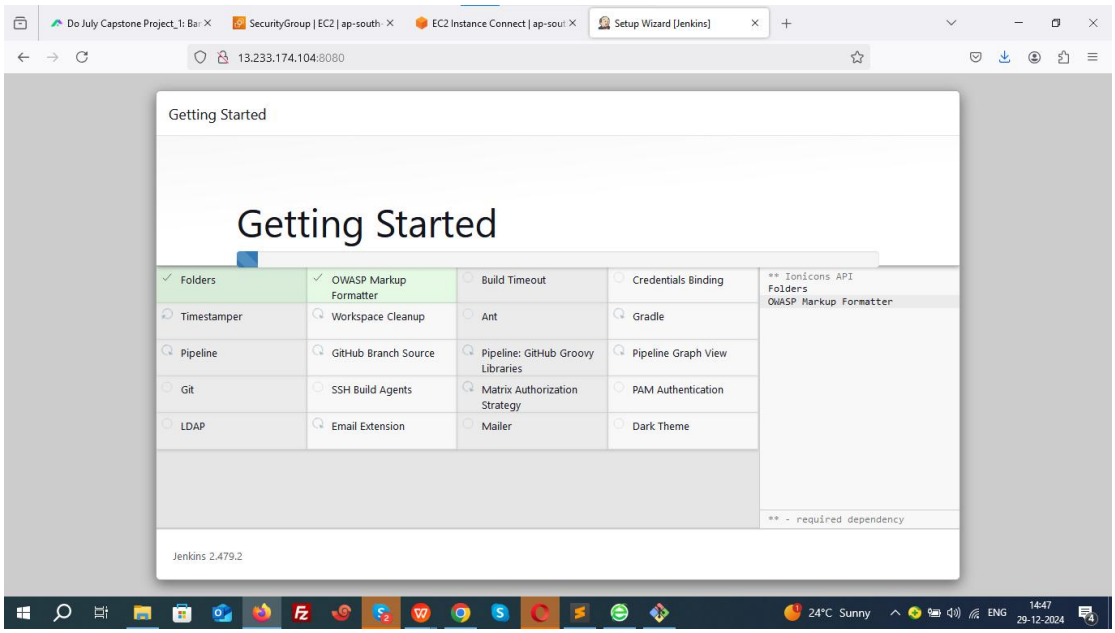
http://13.233.174.104:8080/

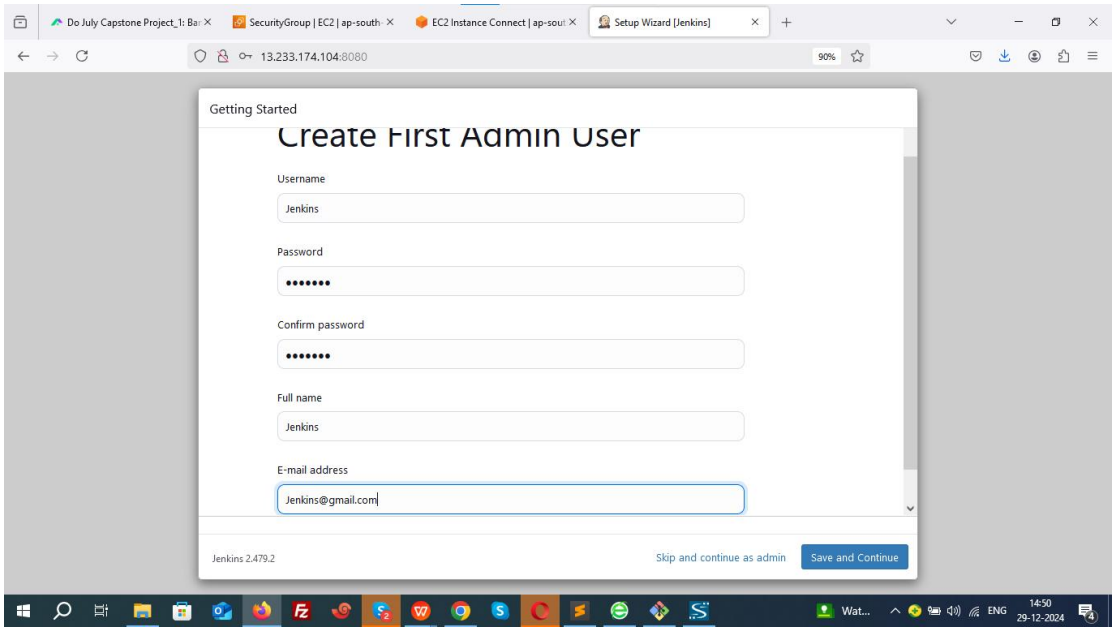
Default port: 8080

4. Set up plugins and admin user.

cat /var/lib/jenkins/secrets/initialAdminPassword


Sign in [Jenkins]
13.233.174.104:8080/login?from=%2F
Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/lib/jenkins/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
This connection is not secure. Logins entered here could be compromised. Learn More
Continue

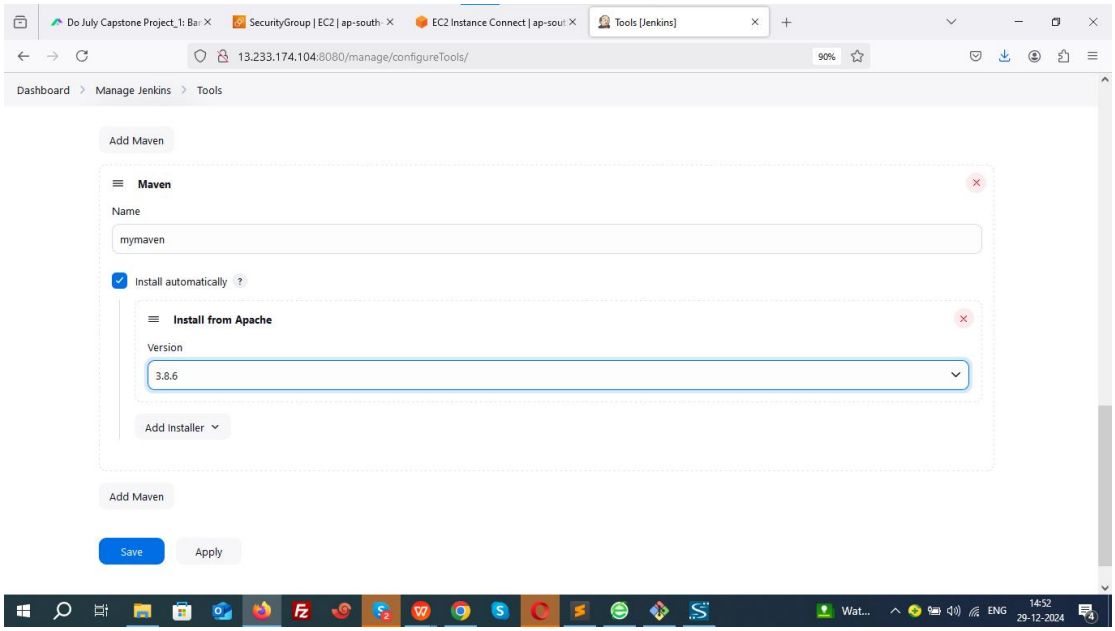

Setup Wizard [Jenkins]
13.233.174.104:8080
Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
** Ionicons API
Folders
OWASP Markup Formatter
** - required dependency
Jenkins 2.479.2



5. **Configure Maven**:

Go to **Manage Jenkins > Global Tool Configuration**.

Add Maven and set it up.



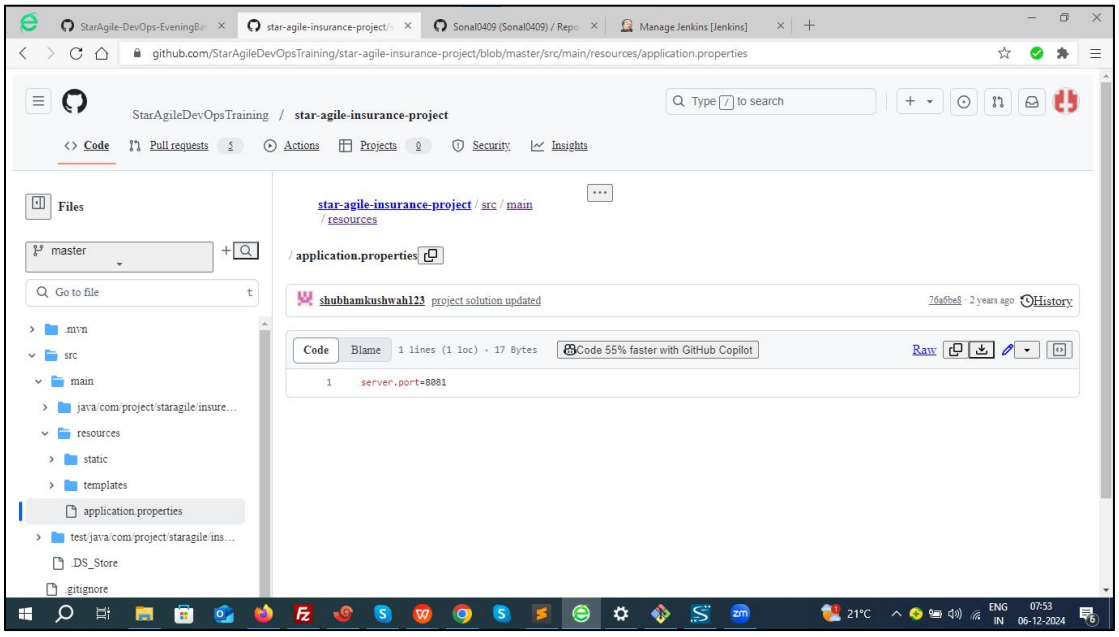
Make sure to give same name for maven in pipeline.

6. **Create Pipeline**:

Points to be noted:

In source code go to src/main/resources/application.properties

Check server port

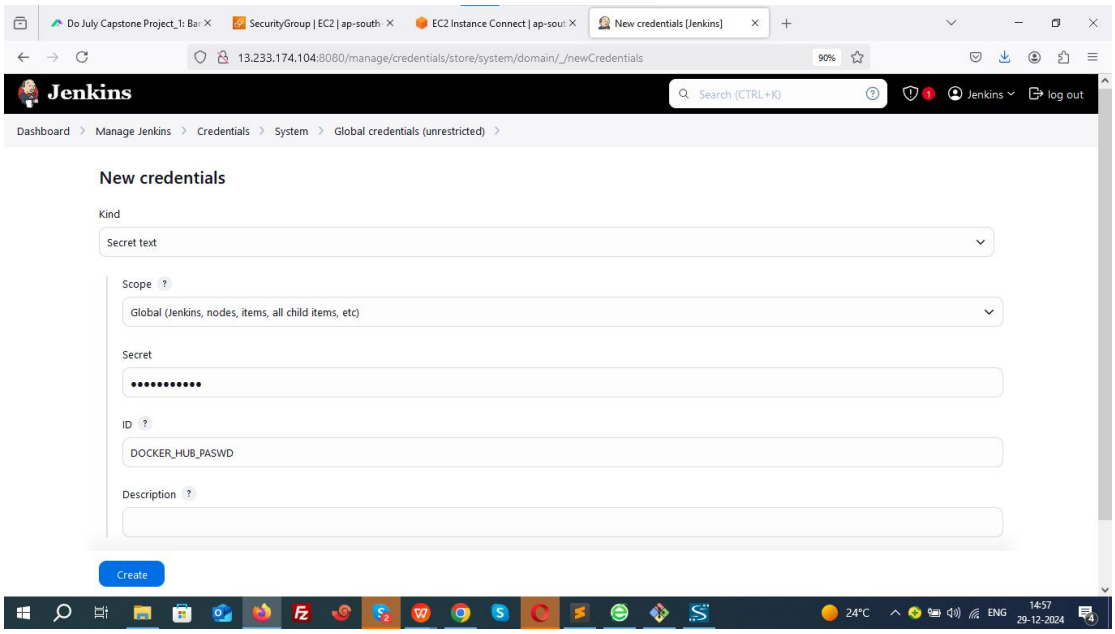


Map the internal container port (8081) to an external port on your host machine.

docker run -d -p 8081:8081 insurance_me:$BUILD_NUMBER

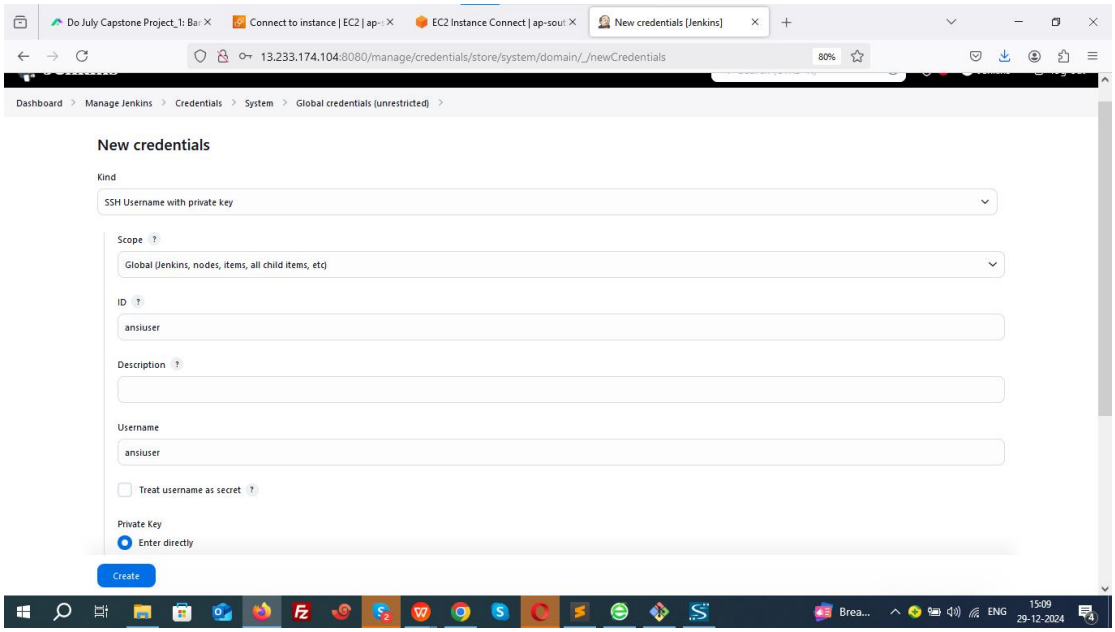
Enable 8081 in your ec2 security groups.

1. Check and Configure Docker Hub Credentials in Jenkins



2. Install Ansible plugin.
3. Add ansiuser credential In Jenkins

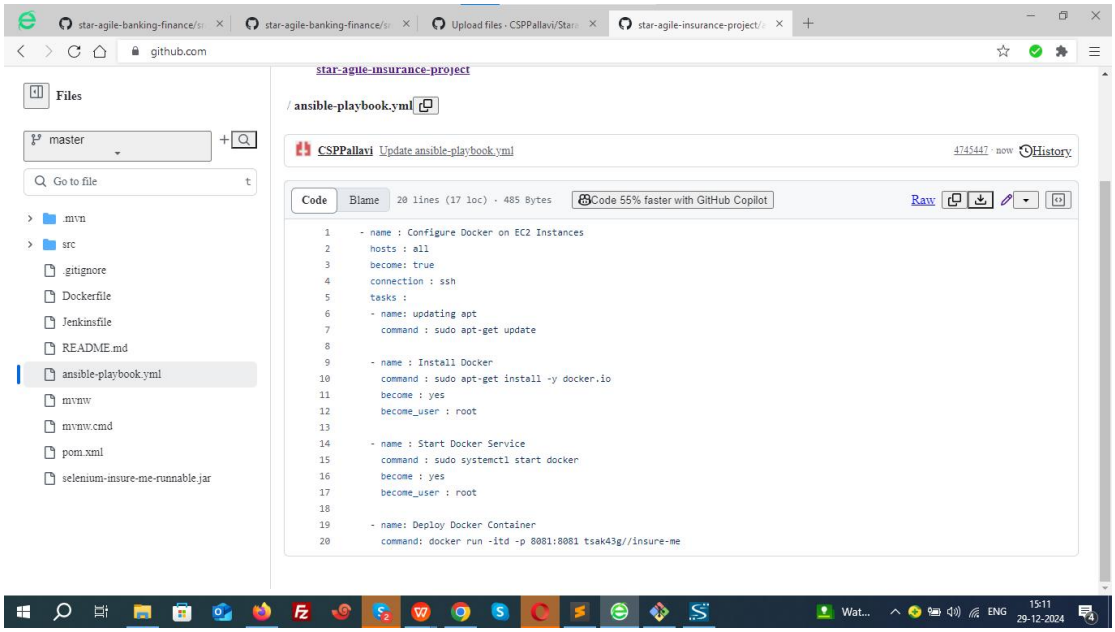
(In Jenkins we have created ssh key for ansiuser)



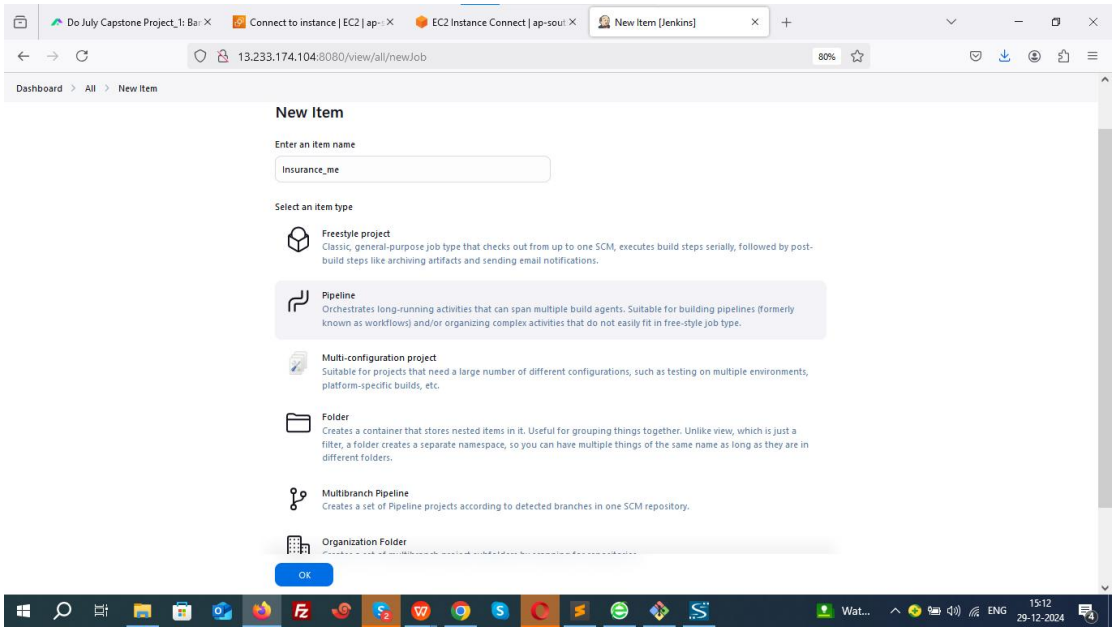
4. We are going to run playbook which is already available on git hub.

ansible-playbook.yml

Just edit image name



Navigate to **New Item > Pipeline**.



Add below pipeline

```
pipeline {
   agent any
   tools {
      maven 'mymaven'
   }
   stages {
      stage('Clone Repo') {
         steps {
            git 'https://github.com/CSPPallavi/star-agile-insurance-project.git'
         }
      }
```

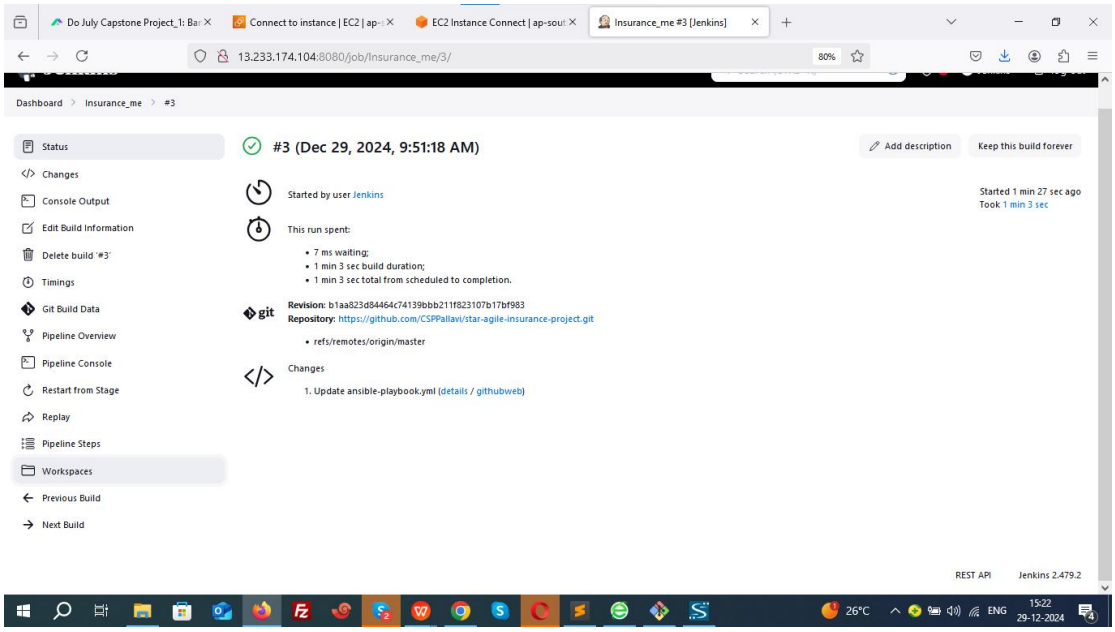
```
        stage('Test Code') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Build Code') {
            steps {
                sh 'mvn package'
            }
        }
        stage('Build Image') {
            steps {
                sh 'docker build -t banking_finance:$BUILD_NUMBER .'
            }
        }
        stage('Push the Image to DockerHub') {
            steps {
                withCredentials([string(credentialsId:
'DOCKER_HUB_PASWD', variable: 'DOCKER_HUB_PASWD')]) {
                    sh 'docker login -u tsak43g -p
${DOCKER_HUB_PASWD}'
                }
```

```
            sh 'docker tag banking_finance:$BUILD_NUMBER
tsak43g/insure-me:$BUILD_NUMBER'
            sh 'docker push tsak43g/insure-me:$BUILD_NUMBER'
        }
    }
    stage('Deploy to Worker Server') {
        steps {
            ansiblePlaybook(
                credentialsId: 'ansiuser',  // Use the credentials ID for
the SSH private key required by Ansible
                playbook: 'ansible-playbook.yml',
                inventory: '/etc/ansible/hosts',
                disableHostKeyChecking: true,
                become: true
            )
        }
    }
  }
}
```

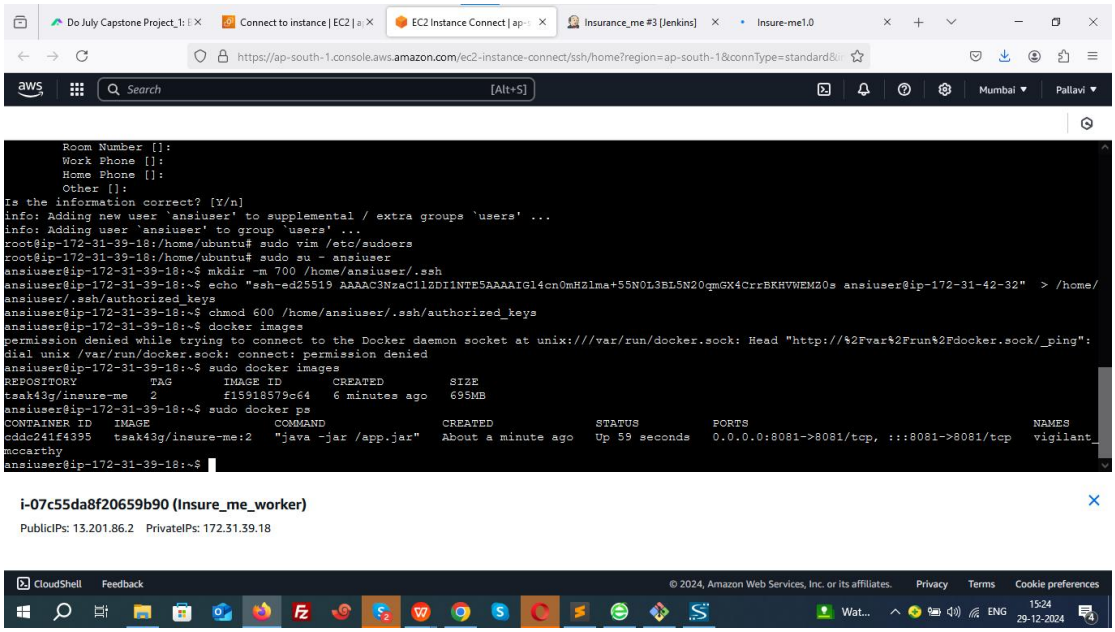
7. **Befor building the job allow Jenkins user to Run Docker Commands**:

chmod -R 777 /var/run/docker.sock

8. Build the job successfully.

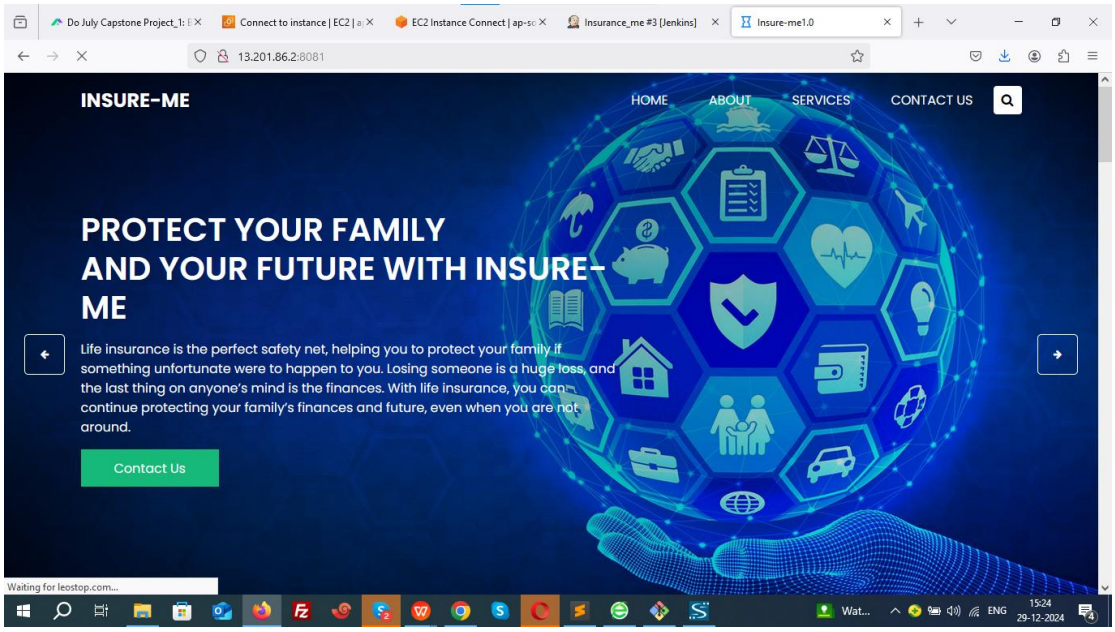


9. Verify docker image and container.



Access the container

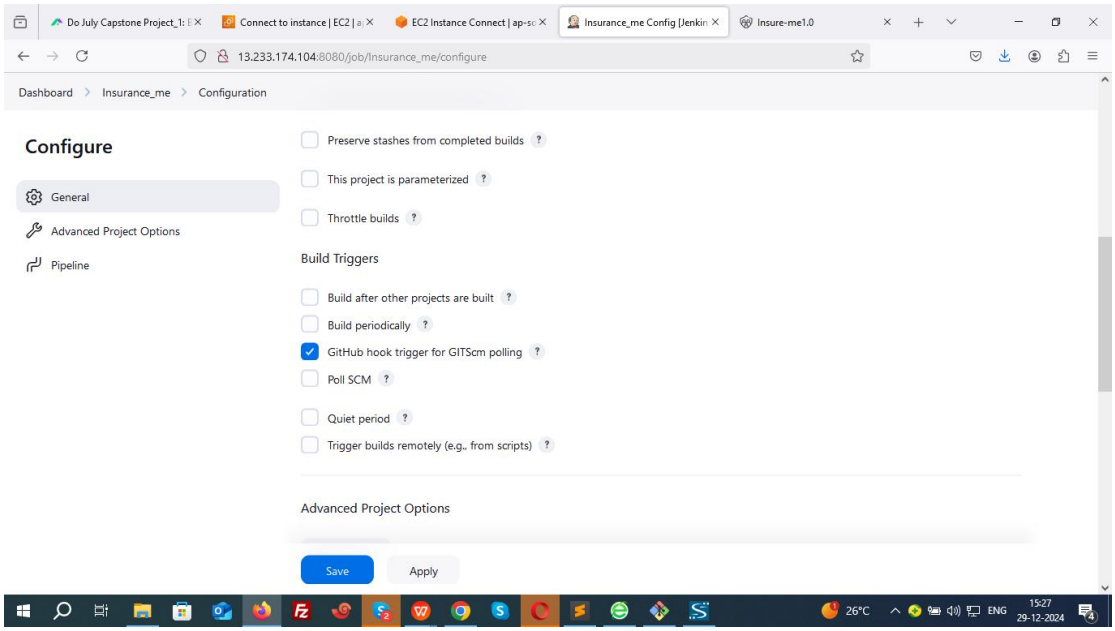
http://13.201.86.2:8081/



Application is deployed successfully in docker container.

STEP 5: WEBHOOK CONFIGURATION:

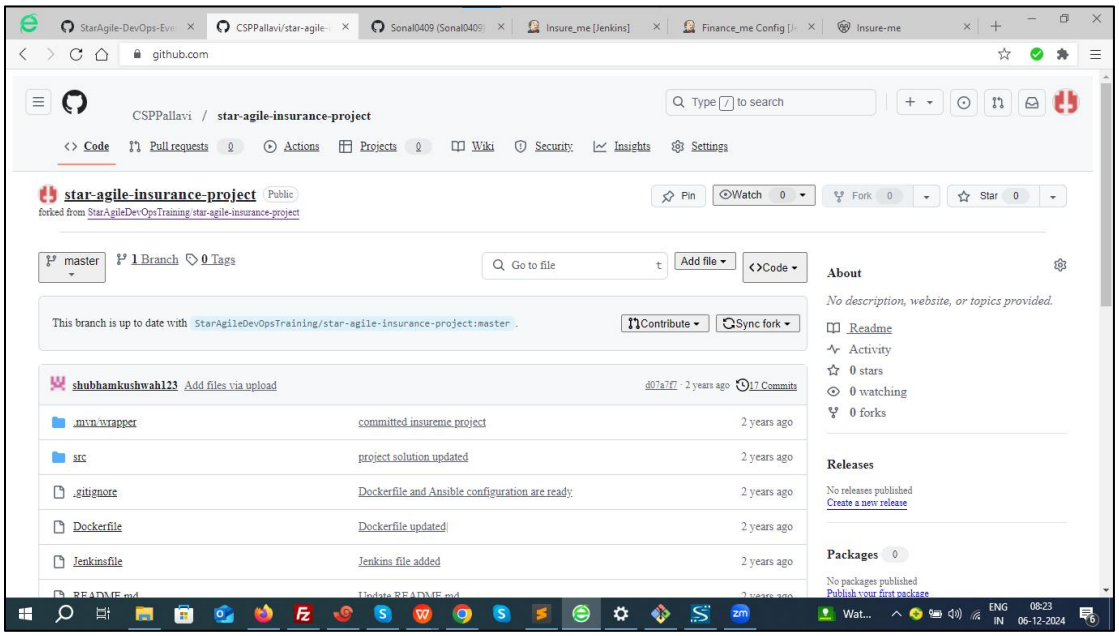
Go to your Jenkins dashboard and add build trigger- **GitHub hook trigger for GITScm polling**



Configuring a webhook in Git is a crucial step to enable automated actions, such as triggering a CI/CD pipeline in Jenkins or notifying an application about repository events.

Step 1: Open Your Repository Settings

1. Navigate to the repository where you want to configure the webhook.

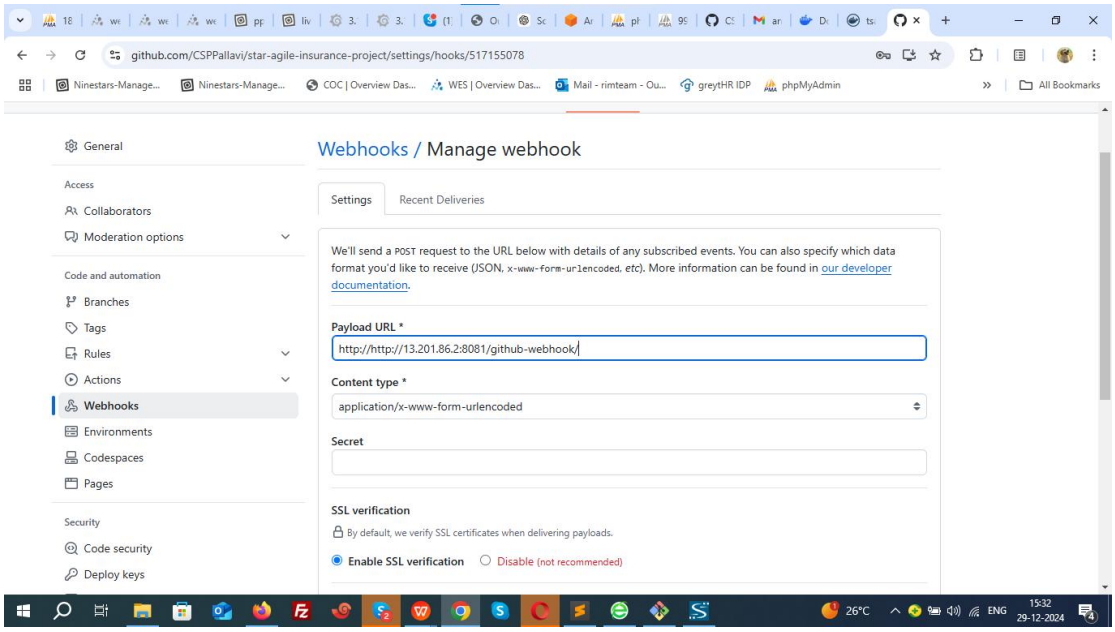


Step 2: Find the Webhooks Section

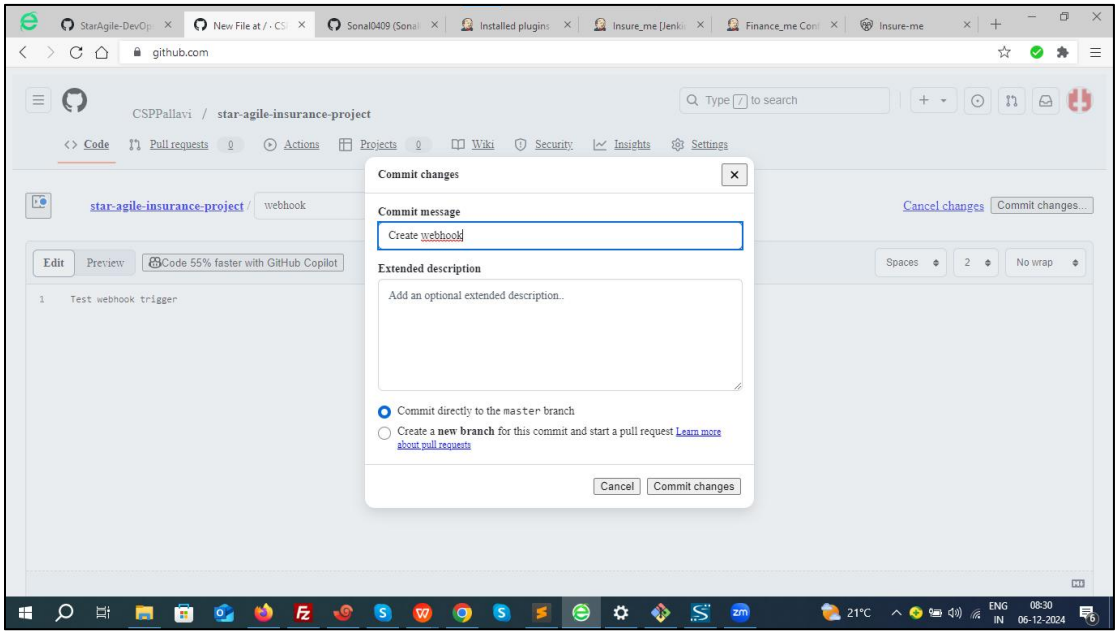
Go to **Repository Settings** > **Webhooks**.

Step 3: Add a New Webhook

Fill in the webhook details:
Payload URL:
· This is the endpoint that will receive the webhook payload.
· http://<Jenkins-URL>/github-webhook/
http://13.233.174.104:8080//github-webhook/

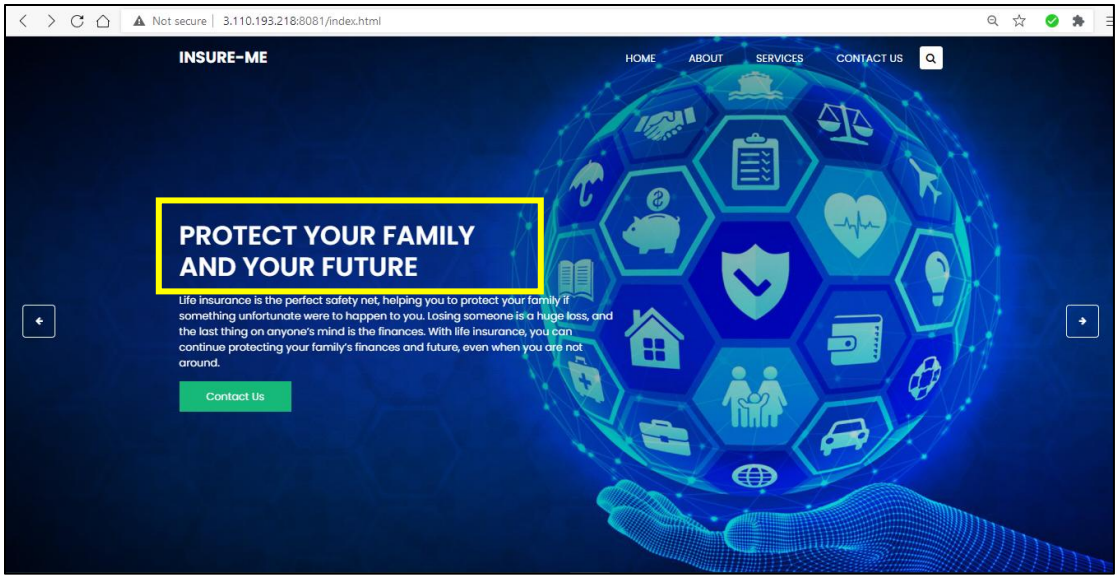


Step 4 : Commit changes.



Step 5 : Testing webhook.

Lets make changes in the application Index.html page.

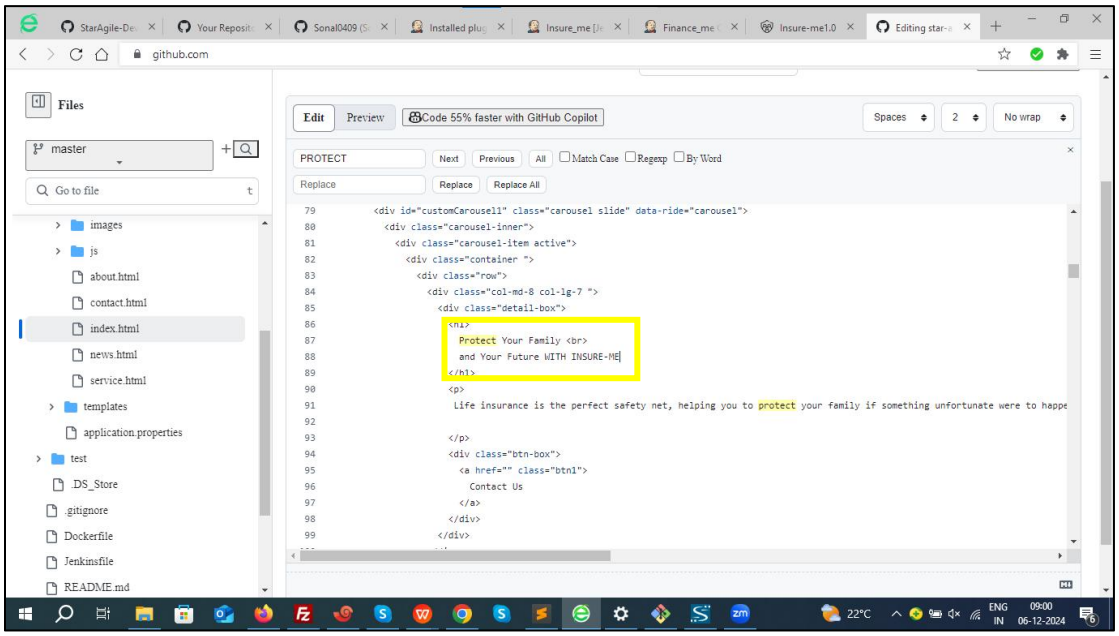


Let's make changes in welcome page where it says PROTECT YOUR FAMILY AND YOUR FUTURE.

Let's change this to PROTECT YOUR FAMILY AND YOUR FUTURE WITH INSURE-ME

Go to index.html path in your git hub

star-agile-insurance-project/src/main/resources/static



Now commit the changes.

Whenever changes are done to code, new build will start automatically.