

MODULE NAME
CAPSTONEPROJECT
PROJECT NAME
INSURANCE-DOMAIN
SUBMITTED BY : RAHUL R
ON : 26-05-2025

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the Jenkins job should be triggered using a GitHub Webhook and Jenkins job should be triggered, The code should be checked out, compiled, tested, packaged and containerized and deployed to the preconfigured test-server automatically.

The deployment should then be tested using a test automation tool (Selenium), and if the build is successful, it should be deployed to the prod server. All this should happen automatically and should be triggered from a push to the GitHub master branch.

Tech stack

- ✓ 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.

Launch three Instances

Name	Instance type	OS
Jenkins-master - t3.medium		ubuntu
Production-1 - t2.micro		ubuntu
Test-1 - t2.micro		ubuntu

Jenkins-master machine will consist of Jenkins host and also ansible controller

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, and Network & Security. The main area displays a table of instances. A red box highlights the 'Instances' section, specifically the table row for 'jenkins-hosts'. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public DNS. The 'jenkins-hosts' instance is listed as 'Running' on 't3.medium' and has 3/3 checks passed. Below the table, a detailed view for the 'jenkins-hosts' instance is shown, with its Public IPv4 address (3.108.254.125) highlighted with a red box. The detailed view includes tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags.

Open all the instances in mobaxterm . and Install all the required packages like Java , Jenkins and also start the Jenkins and check if the service is running in the master machine

The screenshot shows a MobaXterm window with three terminal sessions. The left sidebar lists sessions for 'jenkins-host', 'production', and 'test'. The 'jenkins-host' session is active and shows Jenkins installed and running. The terminal output includes commands like 'java --version', 'systemctl status jenkins', and logs from the Jenkins service. A red box highlights the Jenkins service status output, which shows it is active and running. The 'production' and 'test' sessions are also visible in the background.

Install maven , git ,ansible

The screenshot shows a MobaXterm window titled 'jenkins-host'. It has three tabs: 'jenkins-host' (selected), '3.production', and '4.test'. The terminal window displays the following command outputs:

```
root@ip-172-31-12-16:/home/ubuntu# ansible --version
ansible [core 2.18.6]
  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
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb 4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-12-16:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.15, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1024-aws", arch: "amd64", family: "unix"
root@ip-172-31-12-16:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-12-16:/home/ubuntu#
```

The terminal window is highlighted with a red rectangle.

Now perform ansible configuration from controller node to the other two nodes (i.e production and test) also add the user in other two nodes also

The screenshot shows a MobaXterm window titled 'jenkins-host'. It has three tabs: 'jenkins-host' (selected), '3.production', and '4.test'. The terminal window displays the following command output:

```
root@ip-172-31-12-16:/home/ubuntu# adduser devops
info: Adding user 'devops' ...
info: Selecting new UID/GID from range 1000 to 59999 ...
info: Adding new group 'devops' (1001) ...
info: Adding new user 'devops' (1001) with group `devops (1001)' ...
info: Creating home directory '/home/devops' ...
info: Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for devops
Enter the new value, or press ENTER for the default
      Full Name []:
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []
Is the information correct? [Y/n]
info: Adding new user 'devops' to supplemental / extra groups 'users' ...
info: Adding user 'devops' to group 'users'
root@ip-172-31-12-16:/home/ubuntu#
```

The terminal window is highlighted with a red rectangle.

Now add devops user to sudoers file permission in master as well as other nodes also

The screenshot shows a terminal window titled "jenkins-host" with the command "nano /etc/sudoers". The file contains the following content:

```
#!/usr/bin/nano 7.2
# Defaults: sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"
# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults: sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults: sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults: sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_"

# Per-user preferences; root won't have sensible values for them.
#Defaults: sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults: sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults: sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
devops  ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d
```

A red box highlights the section starting with "# User privilege specification".

Now configure ssdd file in master as well as in other nodes in the ssh-config file which can be accessed using
vi /etc/ssh/sshd_config.d/60-cloudimg-settings.conf

The screenshot shows a MobaXterm interface with multiple sessions open. The current session is on the 'production' server. The terminal window displays the contents of the '/home/ubuntu/' directory. A red box highlights the configuration file '60-cloudimg-settings.conf' which contains the line 'PasswordAuthentication yes'. The status bar at the bottom shows system information including CPU usage, memory, network, and disk usage.

also change the sshd_configuration file using “vi /etc/ssh/sshd_config” permitlogin and pubkeyauthentication to yes and password authentication to yes

```
# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
:q

ip-172-31-12-16  0%  1.02 GB / 3.75 GB  0.01 Mb/s  0.00 Mb/s  33 min  ubuntu (x2)  /: 13%  /boot: 10%  /boot/efi: 6%
```

Generate the ssh key from the devops user using command ssh-keygen

The screenshot shows a MobaXterm interface with multiple sessions open. Session 2, titled 'jenkins-host', displays the following terminal output:

```
info: Adding user 'devops' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group 'devops' (1001) ...
info: Adding new user 'devops' (1001) with group 'devops' (1001) ...
info: Creating home directory '/home/devops' ...
info: Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for devops
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
info: Adding new user 'devops' to supplemental / extra groups 'users' ...
info: Adding user 'devops' to group 'users' ...
root@ip-172-31-12-16:/home/ubuntu# visudo
root@ip-172-31-12-16:/home/ubuntu# su - devops
devops@ip-172-31-12-16:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/devops/.ssh/id_ed25519):
Created directory '/home/devops/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/devops/.ssh/id_ed25519
Your public key has been saved in /home/devops/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:vLijmXfSYH5RpXWqIoYXXAFFmmtX3YDkuRJvWv devops@ip-172-31-12-16
The key's randomart image is:
++-[ED25519 256] ++
| +.*=.E..
| .*o. * o o
| .o. .oo. .
| . 0000
| . S+.=o.
| o .*.o.
| ..OB.
| o+.+
| ..
+---[SHA256]---+
devops@ip-172-31-12-16:~$
```

A red box highlights the SHA256 key fingerprint and the beginning of the randomart image.

Add the private ip address in the ansible hosts file i.e inventory file of both the nodes

The screenshot shows a terminal session titled "jenkins-host" with the command "ansible hosts" running. The output displays various examples of host definitions, including a section for "nodes" containing two entries: "172.31.14.55 ansible_user=devops" and "172.31.9.158 ansible_user=devops". A red box highlights this "nodes" section.

```
## blue.example.com
## 192.168.100.1
## 192.168.100.10

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

# You can also use ranges for multiple hosts:
## db-[99:101]-node.example.com

# Ex 3: A collection of database servers in the 'dbservers' group:
## [dbservers]
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Ex4: Multiple hosts arranged into groups such as 'Debian' and 'openSUSE':
## [Debian]
## alpha.example.org
## beta.example.org

## [openSUSE]
## green.example.com
## blue.example.com
[nodes]
172.31.14.55 ansible_user=devops
172.31.9.158 ansible_user=devops
"/etc/ansible/hosts" 56L, 1248B
```

Copy the ssh key into the nodes and check if the nodes are connected using the ssh devops@<private_ip>
If it gets into the machine then the machine is configured to the controller

The screenshot shows a terminal session titled "jenkins-host" with the command "ssh-copyid" running. It attempts to copy the SSH key from the local machine to a remote host at 172.31.9.158. The process fails because no identities are found. The user is prompted to continue connecting, and after entering "yes", the key is successfully added. A red box highlights the failed attempt and the successful addition of the key.

```
root@ip-172-31-12-16:~/.ssh# service ssh restart
root@ip-172-31-12-16:~/.home/devops# ssh-copyid devops@172.31.9.158
Command 'ssh-copyid' not found, did you mean:
  command 'ssh-copy-id' from deb openssh-client (1:9.6p1-3ubuntu13.9)
Try: apt install <deb name>
root@ip-172-31-12-16:~/.home/devops# cd .ssh
root@ip-172-31-12-16:~/.home/devops/.ssh# ssh-copy-id devops@172.31.9.158
/usr/bin/ssh-copy-id: ERROR: No identities found
root@ip-172-31-12-16:~/.home/devops/.ssh# su - devops
devops@ip-172-31-12-16:~$ ssh-copy-id devops@172.31.9.158
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
devops@172.31.9.158's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'devops@172.31.9.158'"
and check to make sure that only the key(s) you wanted were added.

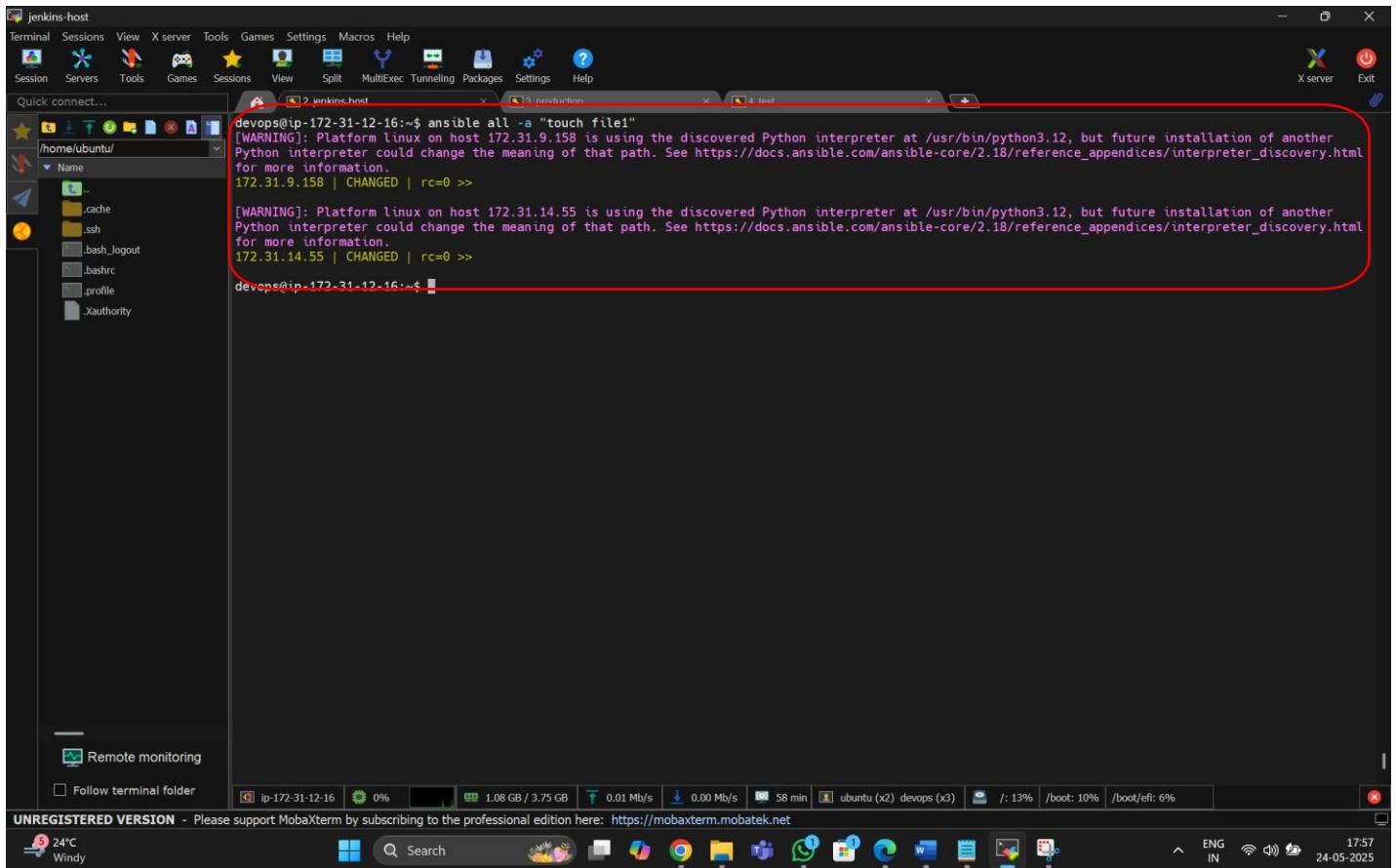
devops@ip-172-31-12-16:~$ ssh-copy-id devops@172.31.14.55
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.14.55 (172.31.14.55)' can't be established.
ED25519 key fingerprint is SHA256:00xznd4Y0AFeFh3CuAtqza5TijjYHxbYlStU13l7LA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
devops@172.31.14.55's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'devops@172.31.14.55'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-12-16:~$
```

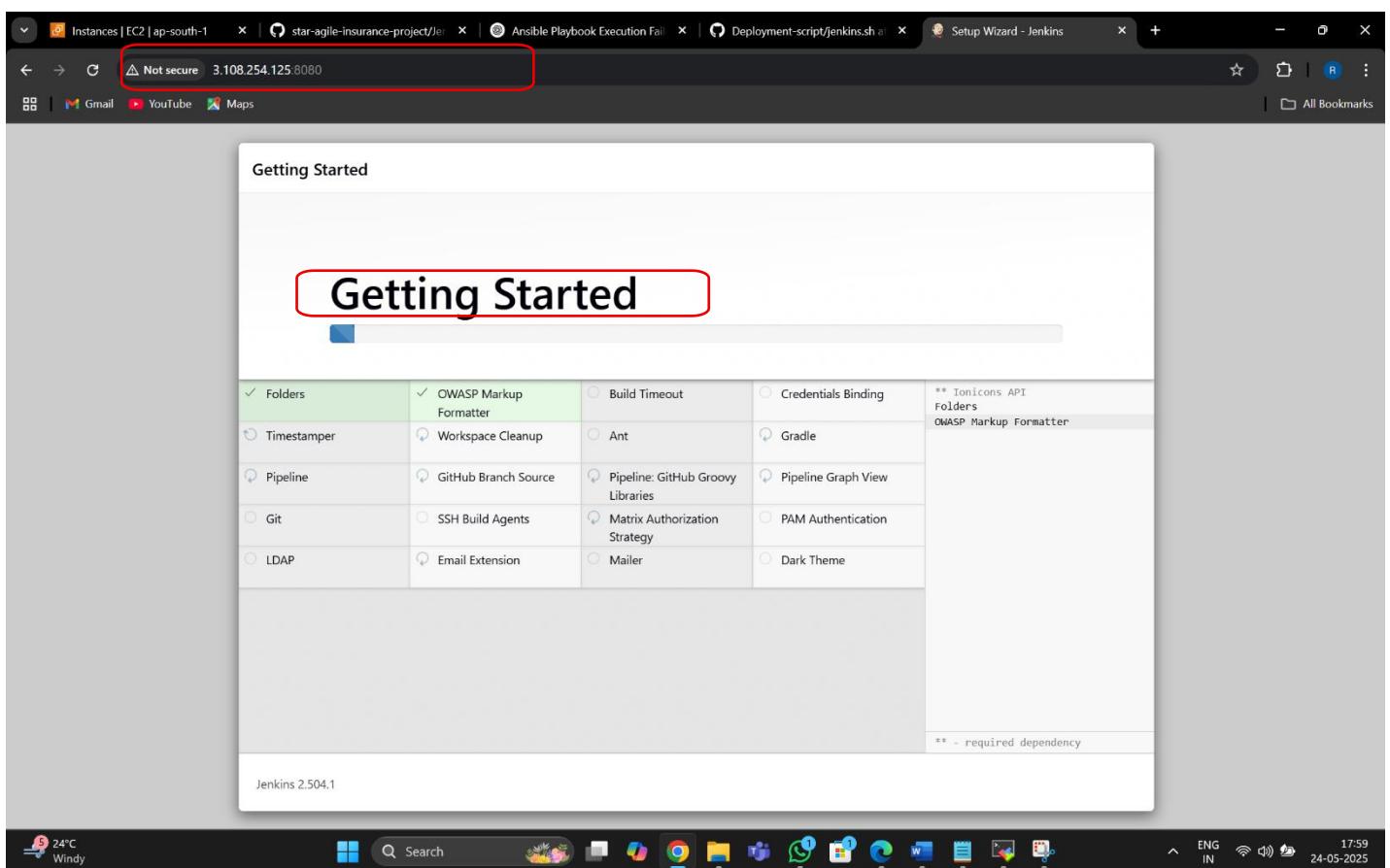
To verify the connection run the ansible command "ansible all -a "touch file1" and check whether the changes happened to the nodes



```
jenkins-host
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
1 jenkins-host 2 Jenkins 3 Jenkins 4 Jenkins
/home/ubuntu/
Name
.. .cache .ssh .bash_logout .bashrc .profile .xauthority
Remote monitoring
Follow terminal folder ip-172-31-12-16 0% 1.08 GB / 3.75 GB 0.01 Mb/s 0.00 Mb/s 58 min ubuntu (x2) devops (x3) /: 13% /boot: 10% /boot/efi: 6%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
24°C Windy Search 24-05-2025 17:57
```

```
devops@ip-172-31-12-16:~$ ansible all -a "touch file1"
[WARNING]: Platform linux on host 172.31.9.158 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.9.158 | CHANGED | rc=0 >>
[WARNING]: Platform linux on host 172.31.14.55 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.14.55 | CHANGED | rc=0 >>
devops@ip-172-31-12-16:~$
```

Access the Jenkins using the private ip address of the master machine on port 8080



Navigate to the manage Jenkins -> plugins -> available plugins -> install ansible

The screenshot shows the Jenkins Plugins management interface. A search bar at the top contains the text "ansible". Below it, a sidebar on the left lists "Updates", "Available plugins" (which is selected and highlighted in grey), "Installed plugins", and "Advanced settings". The main area displays a table of available plugins. One plugin, "Ansible 524.v9fa_a_4c989224", is highlighted with a red box. This plugin is categorized under "External Site/Tool Integrations" and has tabs for "pipeline", "DevOps", "Build Tools", and "Deployment". Its status is "Released" and it was last updated "2 mo 7 days ago". A note below the plugin says "Invoke Ansible Ad-Hoc commands and playbooks.". Another plugin, "Ansible Tower 0.17.0", is listed below it, with a note stating "This plugin connects Jenkins with Ansible Tower" and was last updated "3 mo 18 days ago". At the top right of the main area is a large "Install" button with a downward arrow icon. The bottom right corner of the screen shows a Windows taskbar with various pinned icons and system status indicators.

Do ansible configuration and set the path in tools configuration

The screenshot shows the Jenkins Tools configuration interface. The top navigation bar includes links for "Instances | EC2 | ap-south-1", "star-agile-insurance-project/Jenkins", "Ansible Playbook Execution Fail", "Deployment-script/jenkins.sh", and "Tools - Jenkins". The main content area is titled "Tools" and shows sections for "Maven installations" and "Ansible installations". Under "Ansible installations", there is a form to "Add Ansible". The "Name" field is set to "ansible" and the "Path to ansible executables directory" field contains "/usr/bin/". A checkbox labeled "Install automatically ?" is present. At the bottom of the form are "Save" and "Apply" buttons, which are both enclosed in a red box. The bottom right corner of the screen shows a Windows taskbar with various pinned icons and system status indicators.

Install docker plugins

The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "docker". On the left sidebar, the "Available plugins" tab is selected. In the main list, two Docker-related plugins are highlighted with red boxes:

- Docker 1274.vc0203fdf2e74**: This plugin integrates Jenkins with Docker. It was released 2 months and 17 days ago.
- Docker Pipeline 611.v16e84da_6d3ff**: This plugin provides Docker containers from pipelines. It was released 2 months and 20 days ago.

A blue "Install" button is located in the top right corner of the main content area. The bottom of the screen shows a Windows taskbar with various icons and system status.

Add the docker hub credentials provide username and password
Provide id of docker hub as dockercreds

The screenshot shows the Jenkins Update credentials page for the "dockercreds" credential. The "Username" field contains "reddyrahul", the "Password" field is set to "Concealed", and the "ID" field contains "dockercreds". The "Save" button at the bottom is highlighted with a red box.

Add the global credentials of the ansible ssh key in the Jenkins
Paste the secret key into the private key section

The screenshot shows the Jenkins Global credentials configuration page. The 'ID' field is set to 'ansible-ssh'. The 'Username' field is set to 'devops'. In the 'Private Key' section, the 'Enter directly' option is selected, and a large red box highlights the 'Key' input area where the Ansible SSH private key has been pasted. A smaller red box highlights the 'Create' button at the bottom left.

Create a new item and enter the name of the project and select the item type to pipeline

The screenshot shows the Jenkins New Item creation page. The 'Item name' field is set to 'Insurance'. The 'Item type' dropdown is set to 'Pipeline', which is highlighted by a large red box. Other options shown are 'Freestyle project' and 'Multi-configuration project'. A smaller red box highlights the 'OK' button at the bottom left.

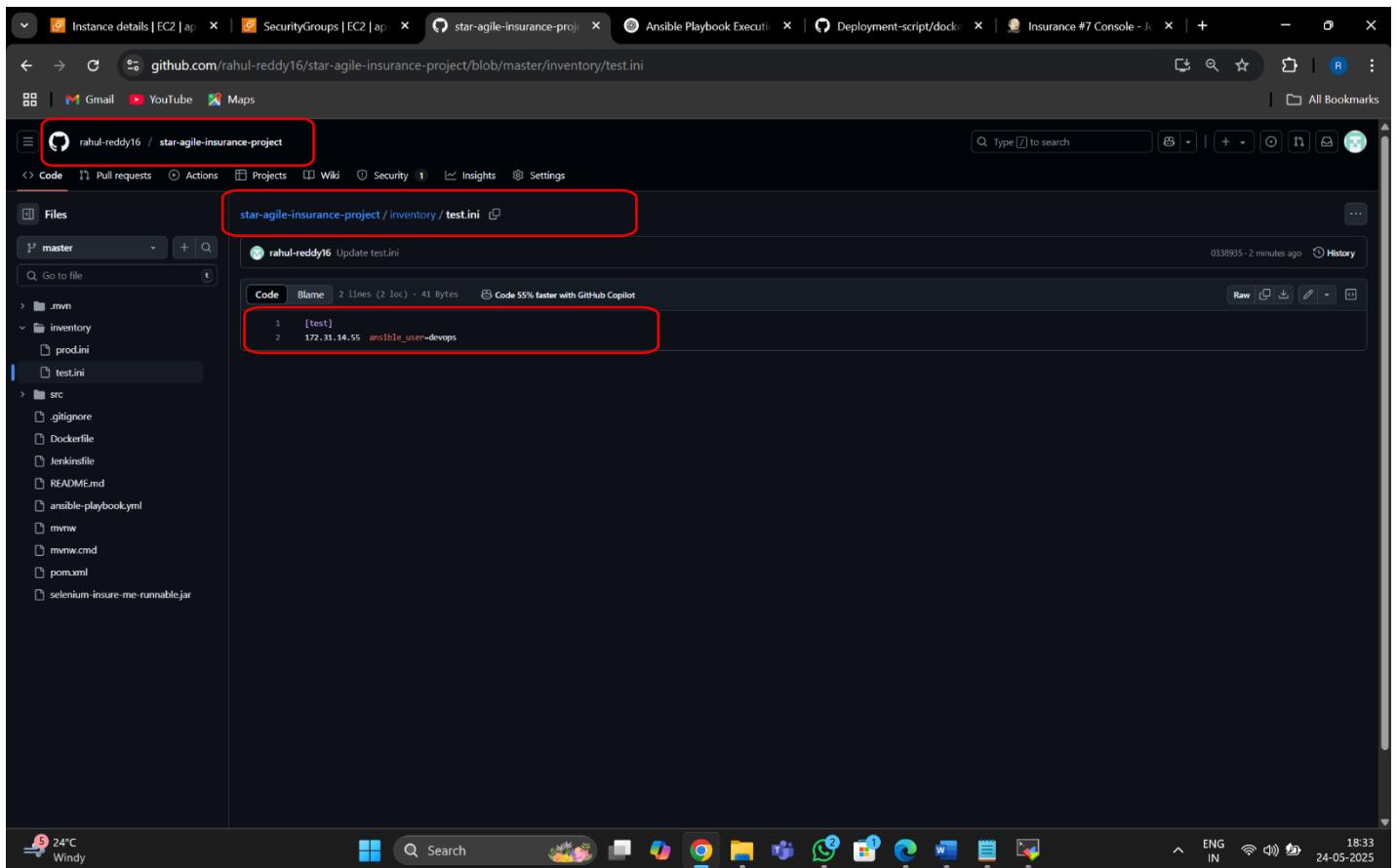
Configure the project with github project repository

The screenshot shows a configuration page for a GitHub project. The 'GitHub project' section is highlighted with a red box. It contains a 'Project url' field with the value <https://github.com/rahul-reddy16/star-agile-insurance-project/>. Other sections like General, Triggers, Pipeline, and Advanced are visible but not selected.

add the triggers

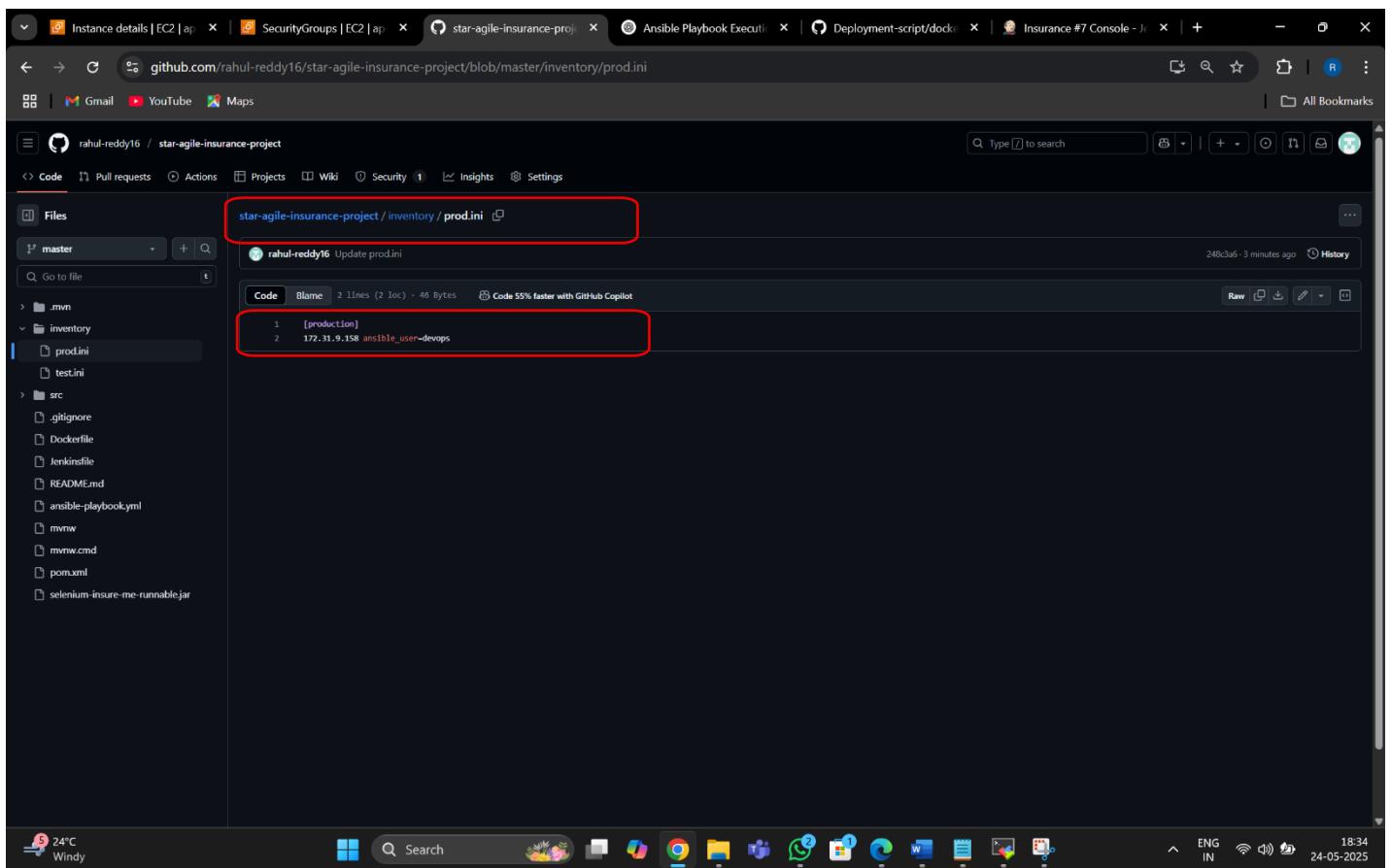
The screenshot shows the triggers configuration page. The 'Poll SCM' and 'GitHub hook trigger for GITscm polling' options are selected and highlighted with a red box. The 'Schedule' field shows 'H * * * *'. Other trigger options like 'Build after other projects are built' and 'Build periodically' are also listed.

Add the inventory file in the github repo for the purpose of proof of concept I have added the private key in to the inventory file with test.ini and prod.ini with there respective private ip's



The screenshot shows a GitHub repository page for 'rahul-reddy16/star-agile-insurance-project'. The 'test.ini' file is displayed, containing the following content:

```
[test]
172.31.34.55 ansible_user=devops
```



The screenshot shows a GitHub repository page for 'rahul-reddy16/star-agile-insurance-project'. The 'prod.ini' file is displayed, containing the following content:

```
[production]
172.31.9.158 ansible_user=devops
```

Now write the Dockerfile

The screenshot shows a GitHub repository page for 'star-agile-insurance-project'. The 'Dockerfile' tab is selected. A red box highlights the code area. The code is as follows:

```
FROM openjdk:11
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Also write the ansible playbook to deploy the containers

The screenshot shows a GitHub repository page for 'star-agile-insurance-project'. The 'ansible-playbook.yml' tab is selected. A red box highlights the code area. The code is as follows:

```
- name: Configure Docker on EC2 Instances
  hosts: all
  become: true
  connection: ssh

  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install Docker
      apt:
        name: docker.io
        state: present

    - name: Start Docker service
      service:
        name: docker
        state: started
        enabled: yes

    - name: Run Docker container
      docker_container:
        name: insure-me
        image: reddyrajuhul/insure-me:latest
        state: started
        ports:
          - "8084:8081"
```

Build the project in jenkins

```
pipeline {
    agent any

    stages {
        stage('Checkout Code') {
            steps {
                git url: 'https://github.com/rahul-reddy16/star-agile-insurance-project.git', branch: 'master'
            }
        }

        stage('Build with Maven') {
            steps {
                sh 'mvn clean package'
            }
        }

        stage('Build Docker Image') {
            steps {
                sh 'docker build -t reddyrahul/insure-me:latest .'
            }
        }

        stage('Push Docker Image') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'dockercreds', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
                    sh 'echo $PASSWORD | docker login -u $USERNAME --password-stdin'
                    sh 'docker push reddyrahul/insure-me:latest'
                }
            }
        }

        stage('Deploy to Test Using Ansible') {
            steps {
                ansiblePlaybook credentialsId: 'ansible-ssh',
                    disableHostKeyChecking: true,
                    installation: 'ansible',
                    inventory: 'inventory/test.ini',
                    playbook: 'ansible-playbook.yml'
            }
        }

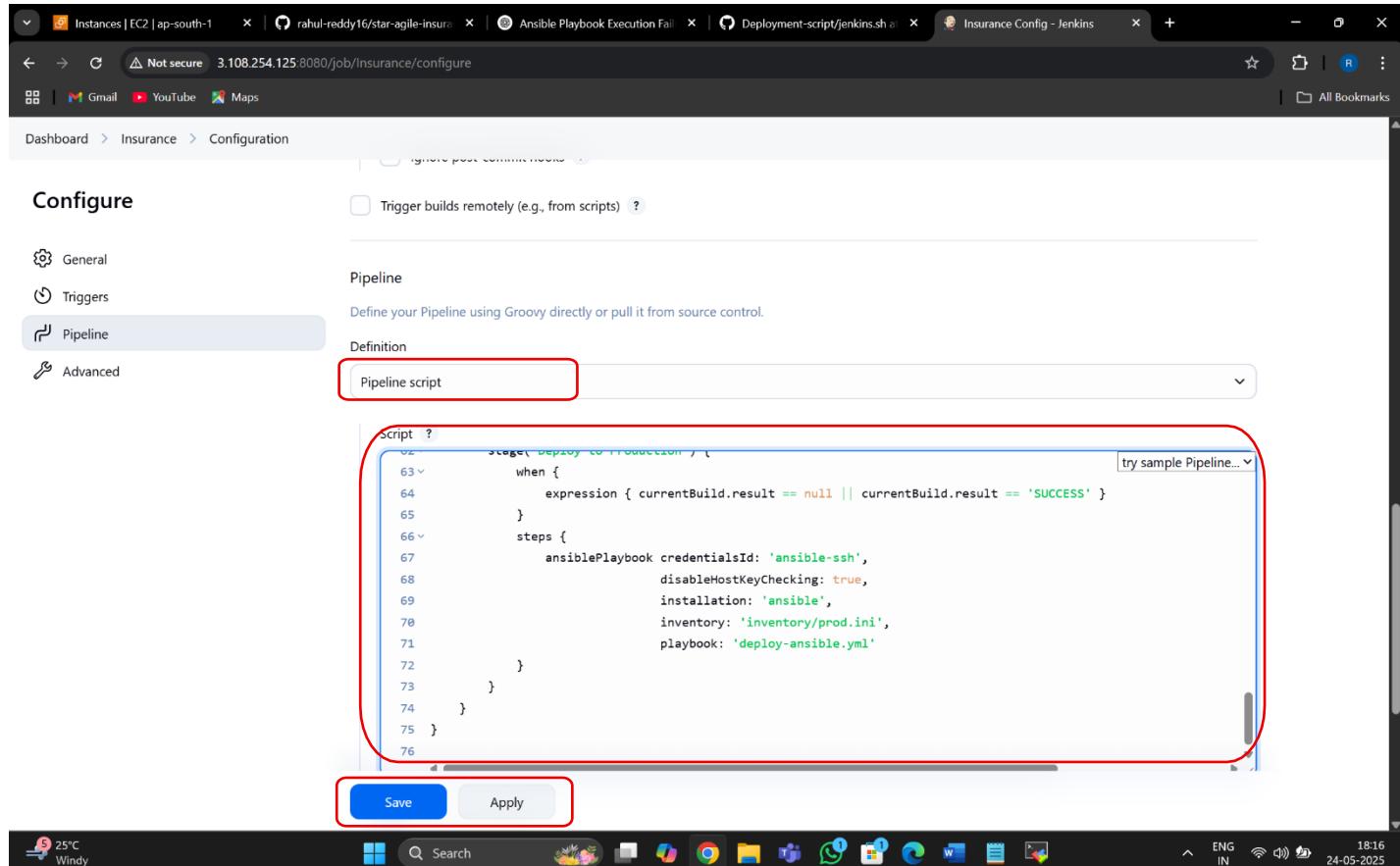
        stage('Wait for Test App to Be Ready') {
            steps {
                sh 'sleep 6'
            }
        }

        stage('Run Selenium Tests on Test machine') {
            steps {
                sh 'mvn test'
            }
        }

        stage('Check Build Status') {
            steps {
                script {
                    echo "Current build result: ${currentBuild.result}"
                }
            }
        }

        stage('Deploy to Production') {
            when {
                expression { currentBuild.result == null || currentBuild.result == 'SUCCESS' }
            }
            steps {
                ansiblePlaybook credentialsId: 'ansible-ssh',
                    disableHostKeyChecking: true,
                    installation: 'ansible',
                    inventory: 'inventory/prod.ini',
                    playbook: 'deploy-ansible.yml'
            }
        }
    }
}
```

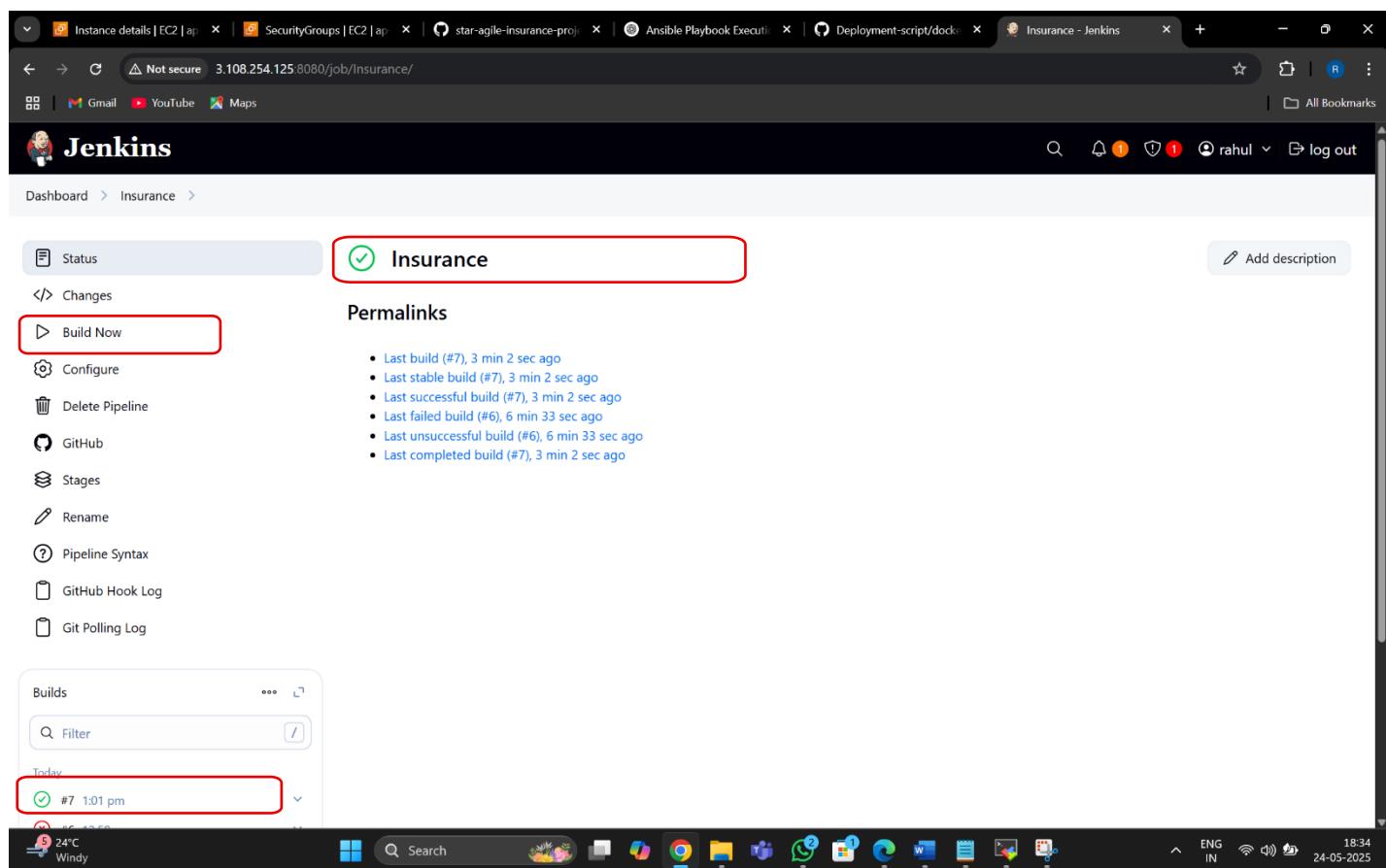
Write the script pipeline in jenkins



The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' tab is selected in the left sidebar. In the main area, there is a 'Definition' section with a dropdown menu set to 'Pipeline script'. A red box highlights this dropdown. Below it is a code editor containing Groovy pipeline code. Another red box highlights the code editor. At the bottom of the page are two buttons: 'Save' and 'Apply'.

```
script ?  
stage('Deploy to Production') {  
    when {  
        expression { currentBuild.result == null || currentBuild.result == 'SUCCESS' }  
    }  
    steps {  
        ansiblePlaybook credentialsId: 'ansible-ssh',  
            disableHostKeyChecking: true,  
            installation: 'ansible',  
            inventory: 'Inventory/prod.ini',  
            playbook: 'deploy-ansible.yml'  
    }  
}
```

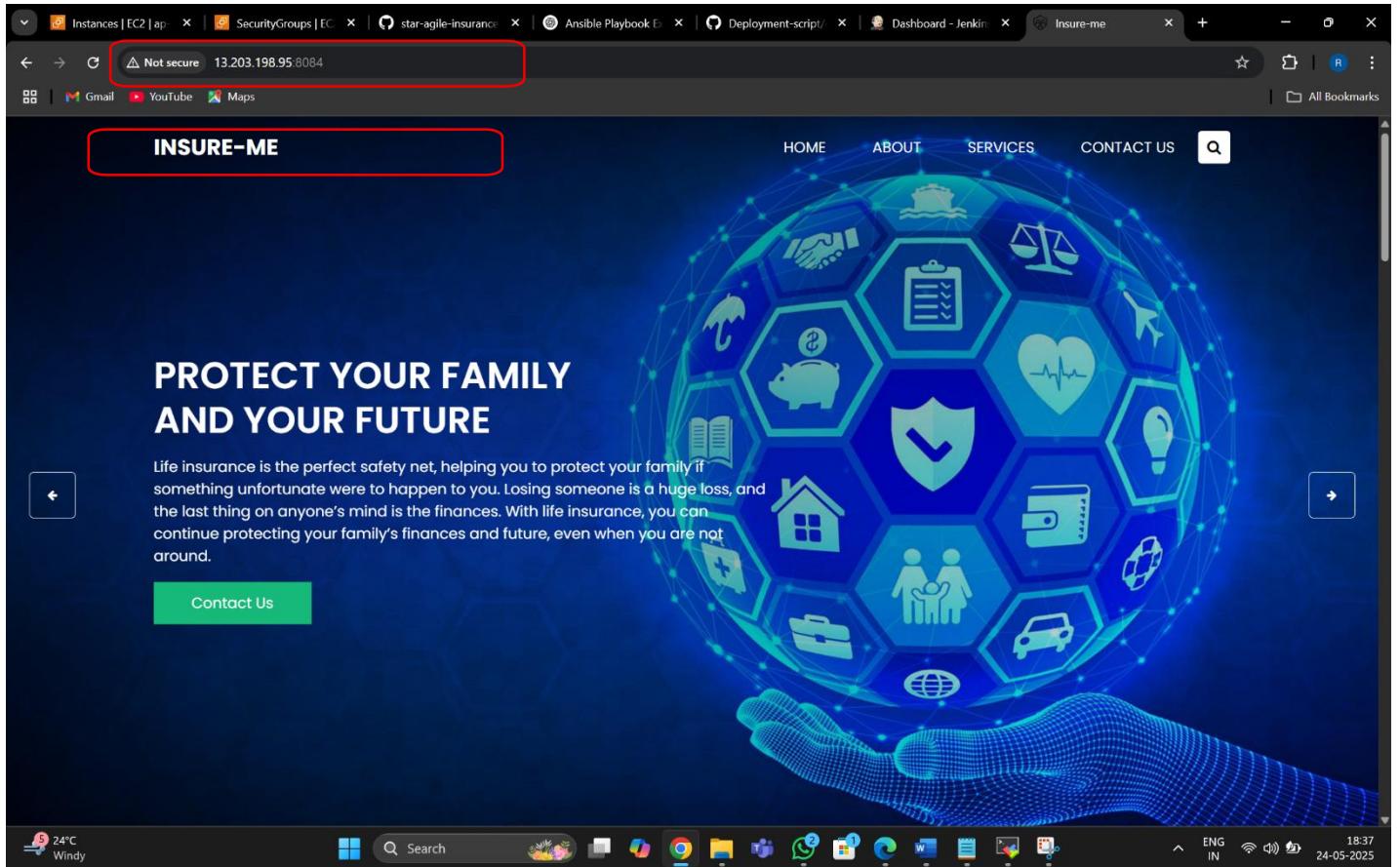
Then save and build



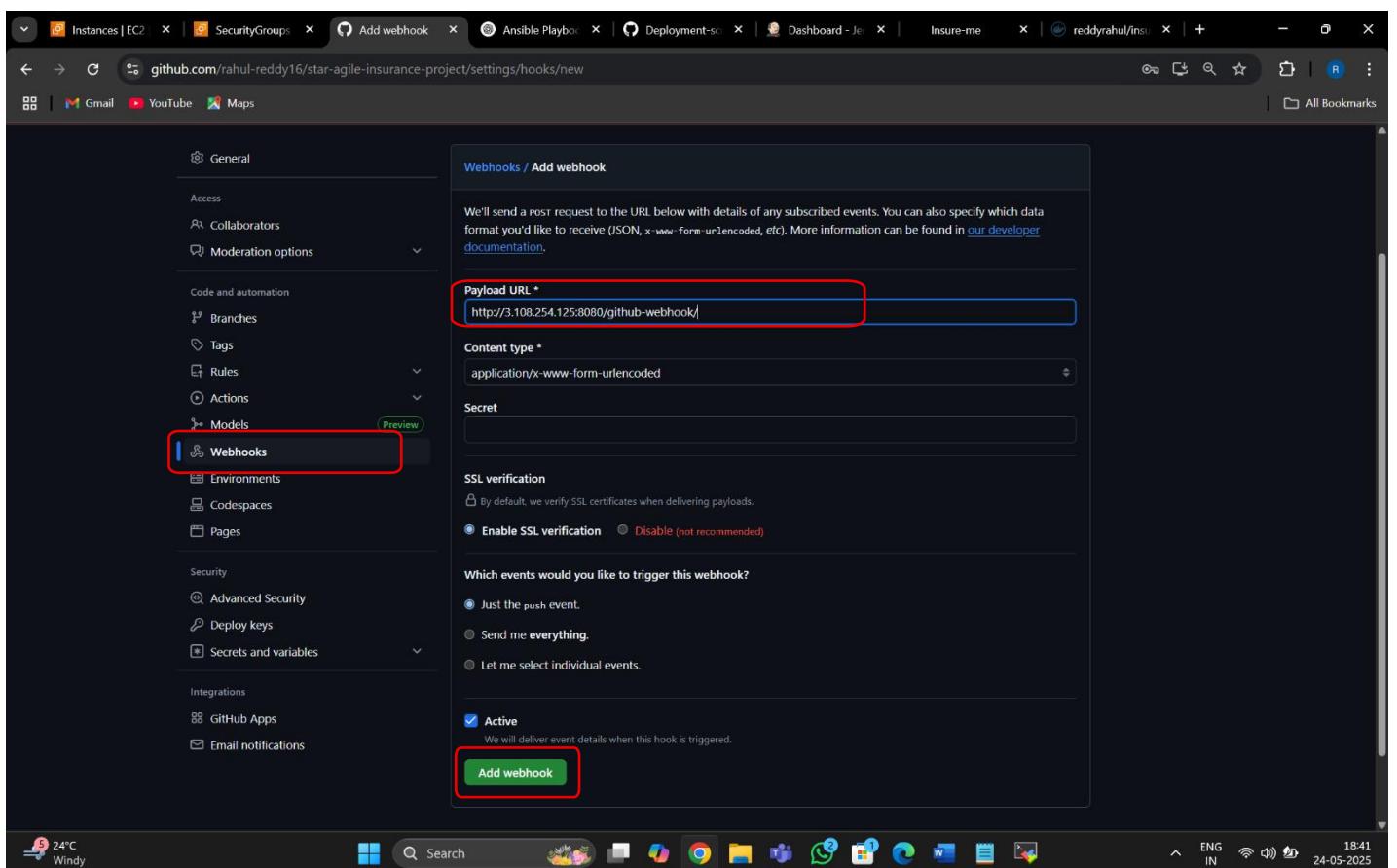
The screenshot shows the Jenkins Pipeline status page for the 'Insurance' pipeline. The pipeline name 'Insurance' is highlighted with a red box. On the left, a sidebar lists pipeline actions: 'Build Now' (highlighted with a red box), 'Configure', 'Delete Pipeline', 'GitHub', 'Stages', 'Rename', 'Pipeline Syntax', 'GitHub Hook Log', and 'Git Polling Log'. Below this is a 'Builds' section with a table showing build history. The most recent build, '#7 1:01 pm', is highlighted with a red box. The status of this build is green. The table includes columns for 'Build', 'Status', and 'Duration'.

Build	Status	Duration
#7 1:01 pm	Green	3 min 2 sec ago
#6 1:01 pm	Red	6 min 33 sec ago
#5 1:01 pm	Green	3 min 2 sec ago
#4 1:01 pm	Green	3 min 2 sec ago
#3 1:01 pm	Green	3 min 2 sec ago
#2 1:01 pm	Green	3 min 2 sec ago
#1 1:01 pm	Green	3 min 2 sec ago

Now access the application using the production ip address on the port 8084



Add the github webhook in the settings of the project repository



Verfying in the docker hub if the image has got created successfully

The screenshot shows a browser window with multiple tabs open, including 'Instances | EC2', 'SecurityGroups', 'star-agile-insure', 'Ansible Playbook', 'Deployment-SC', 'Dashboard - Jenkins', 'Insure-me', and 'reddyrahul/insure-me'. The main content is the Docker Hub interface for the repository 'reddyrahul/insure-me'.

Left Sidebar: Shows the user's profile ('reddyrahul'), repository count (1), and links for 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', and 'Usage'.

Repository Overview: Displays the repository 'reddyrahul/insure-me' under 'General'. It shows the last push was 6 minutes ago, the repository size is 943.5 MB, and there are 0 private repositories.

Tags Section: Shows a single tag named 'latest'.

Tag	OS	Type	Pulled	Pushed
latest	Image	Image	less than 1 day	7 minutes

Buildcloud Integration: An advertisement for Buildcloud, which integrates with Docker Build Cloud to accelerate image build times.

Bottom Bar: Includes a weather icon (24°C Windy), a search bar, and various system icons like battery, signal, and date/time (24-05-2025).