CAPSTONE PROJECT

Insurance Domain-
Insure Me

SUBMITTED BY: DARSANA PRAVEEN.M
DATE: 04/03/2024

CONTENT:

Insure Me is a Global leading Insurance provider based out of USA. The company offers
products and services like Home Insurance, Health Insurance, Car Insurance and Life Insurances.
Initially the company was using a Monolithic application architecture, As the company grown, It
started facing difficulties in managing the application infrastructure and application deployments.
Insure-Me has decided to transform its monolithic application architecture to microservice
application architecture and opted to go DevOps by implementing CICD pipeline and necessary
automations. Insure me has decided to use AWS as primary cloud services provider to create
servers, databases, and application deployments.
The company's goal is to deliver the product updates frequently to production with High
quality & Reliability. They also want to accelerate software delivery speed, quality and reduce
feedback time between developers and testers.
Following are the problems the company is facing at the moment
✓ Building Complex builds is difficult
✓ Manual efforts to test various components/modules of the project
✓ Incremental builds are difficult to manage, test and deploy
✓ Creation of infrastructure and configure it manually is very time consuming
✓ Continuous manual monitoring the application is quite challenging.
In order to implement a POC, you are requested to develop a mavenized microservice using
spring boot and in memory h2 database.
1. a microservice which exposes below mentioned endpoints as APIs and uses in memory
h2 database to store the data.
a. /createPolicy (HTTP Method : POST) (Request Body : JSON)
b. /updatePolicy/{policy id} (HTTP Method : PUT) (Request Body : JSON)
c. /viewPolicy/{policy id} (HTTP Method : GET) (No Request Body)
d. /deletePolicy/{policy id} (HTTP Method : DELETE) (No Request Body)
2. Write necessary Junit testcase.
3. Generate HTML report using TestNG.
4. Push your code into your GitHub Repository.

Note : Preload some data into the database.

Later, you need to implement Continuous Integration & Continuous Deployment using following tools:

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

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

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.

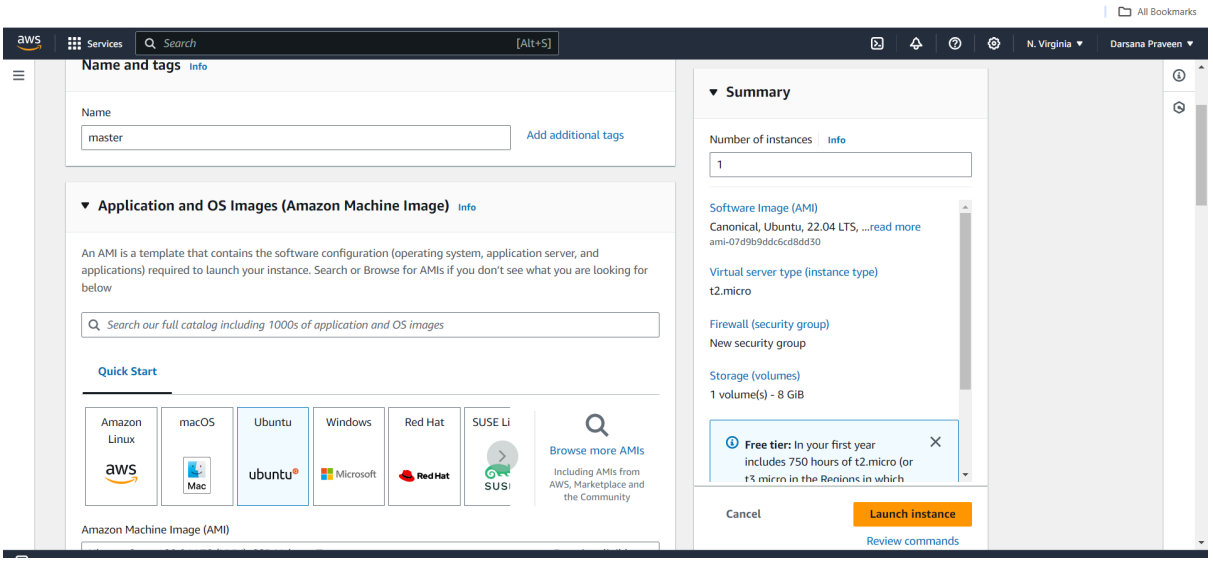
The deployment should then be tested using a test automation tool (Selenium), and if the build
is successful, it should be deployed to the prod server. All this should happen automatically and
should be triggered from a push to the GitHub master branch.

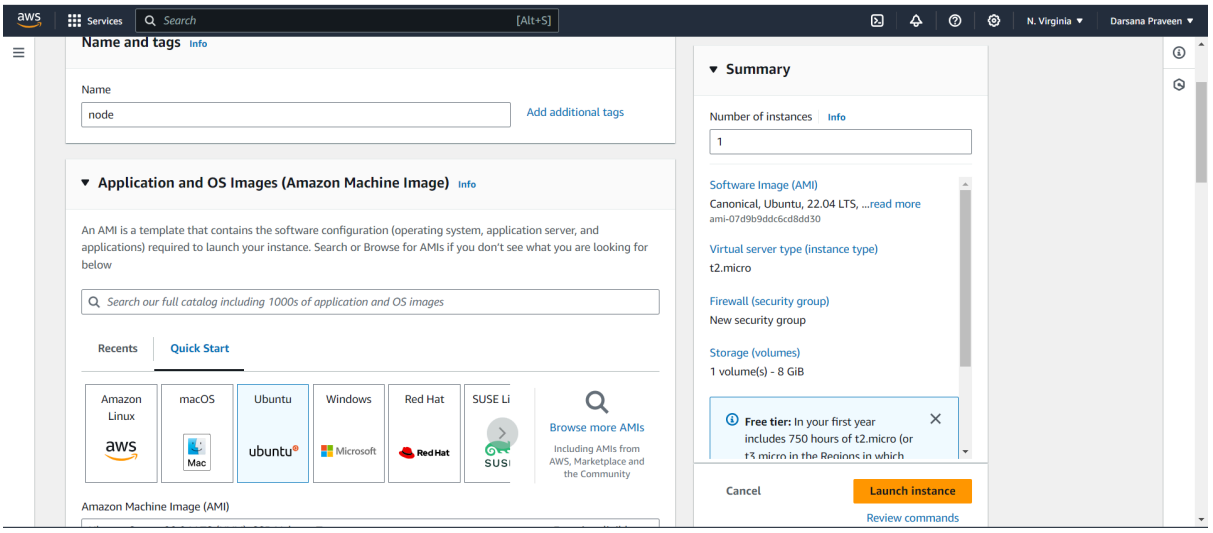
The solution is available at below repository, use it to validate the solution with the solution available at - https://github.com/StarAgileDevOpsTraining/star-agile-insurance-project.git

Note : To have a detailed information about running the application and exposed APIs, Input/Output format, Refer to the README.md in the GitHub repository.

SOLUTION:

Creating a master and node server instance





In the security groups inbound rule is also been edited to allow ALL TRAFFIC

Installing all necessary dependency

INSTALL JAVA:

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-28-64:/home/ubuntu# java --version
openjdk 11.0.22 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu222.04.1, mixed mode, sharing)
root@ip-172-31-28-64:/home/ubuntu#
```

i-0e6616d4889b1de8f (master)

PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

INSTALL MAVEN:

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

```
root@ip-172-31-28-64:/home/ubuntu# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.22, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.2.0-1018-aws", arch: "amd64", family: "unix"
root@ip-172-31-28-64:/home/ubuntu#
```

i-0e6616d4889b1de8f (master)

PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

INSTALL ANSIBLE:

Ansible is a suite of software tools that enables infrastructure as code. It is open-source and the suite includes software provisioning, configuration management, and application deployment functionality

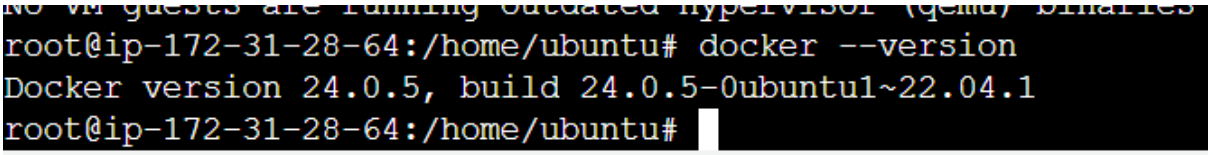
```
root@ip-172-31-28-64:/home/ubuntu# ansible --version
ansible [core 2.16.4]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
root@ip-172-31-28-64:/home/ubuntu#
```

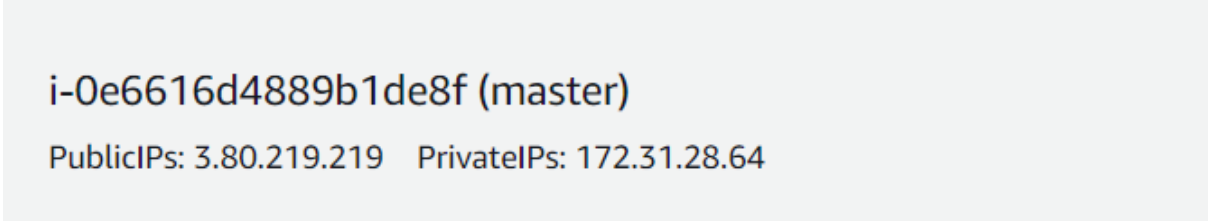
i-0e6616d4889b1de8f (master)

PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

INSTALL DOCKER:

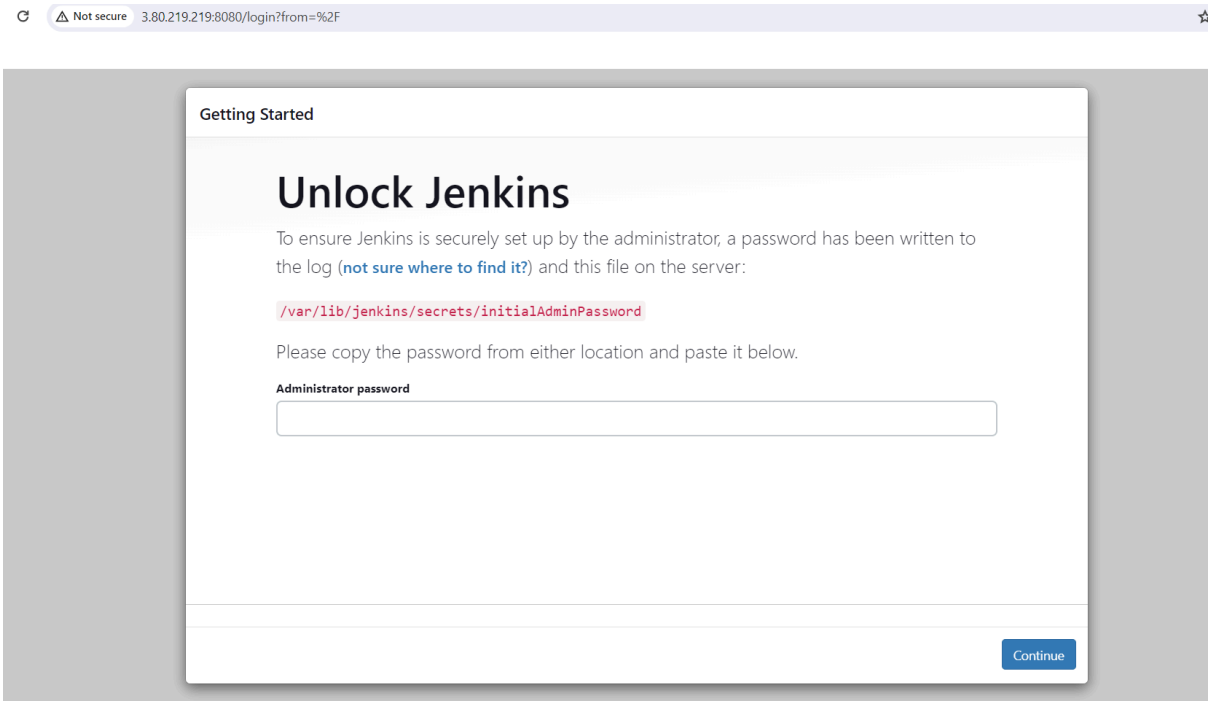
Docker is a platform for building and running containers. Containers are software units that package source code and its dependencies inside isolated environments.

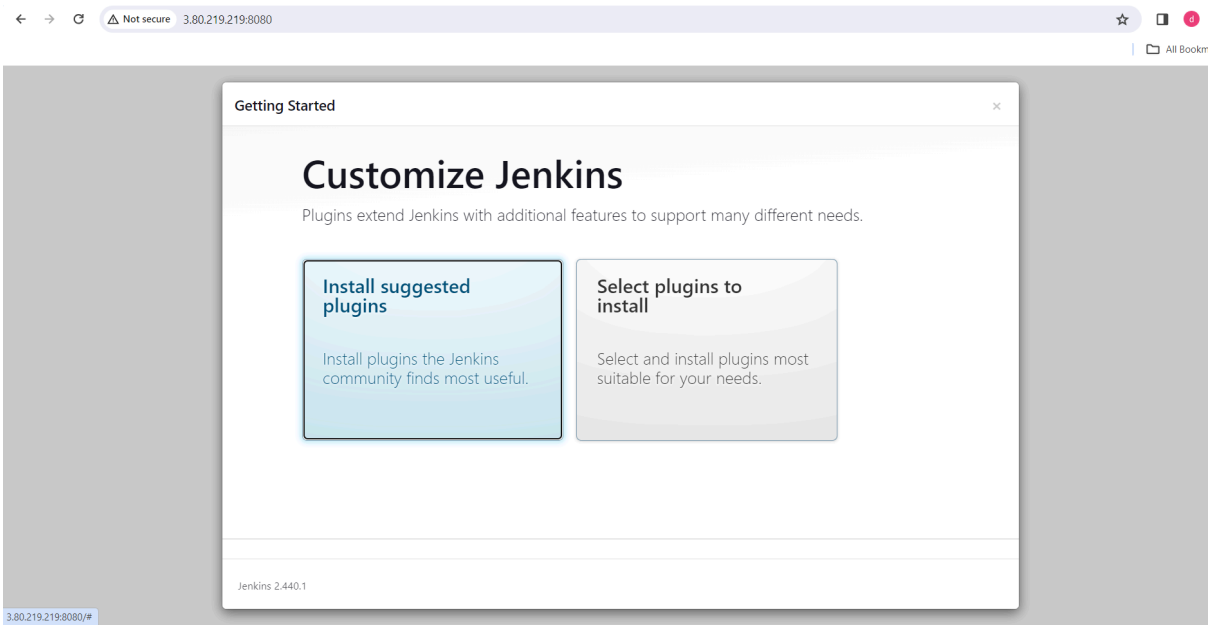


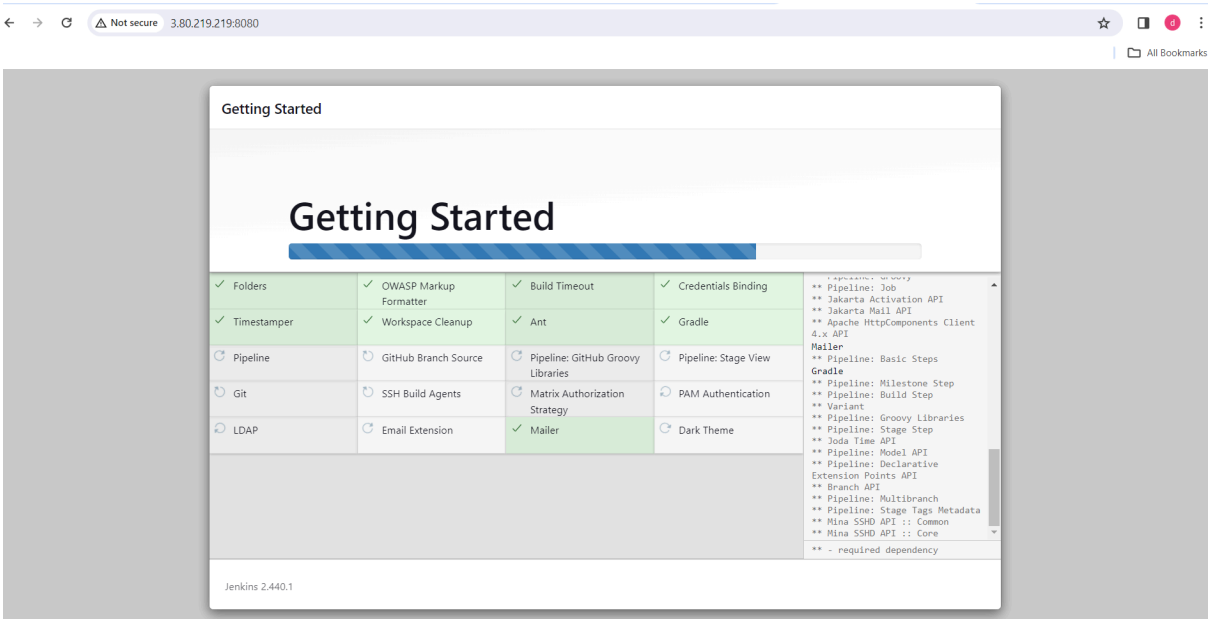


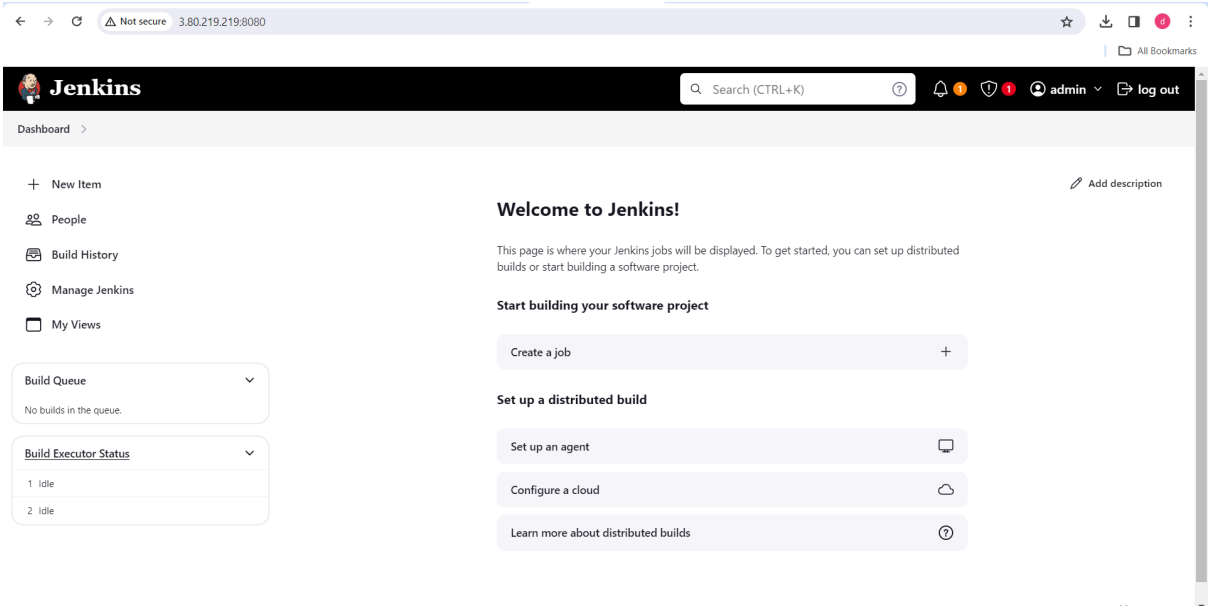
INSTALL JENKINS:

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration, and continuous delivery

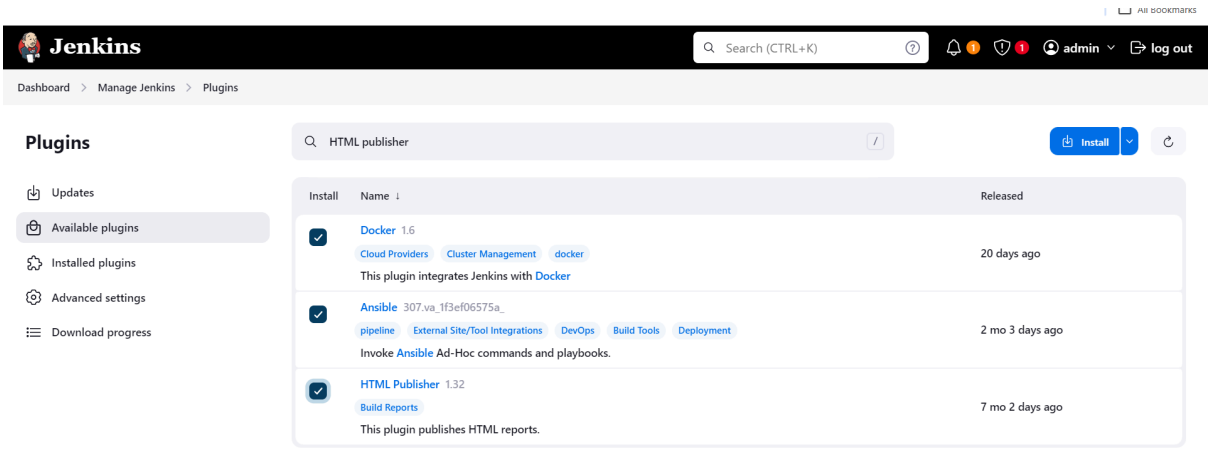


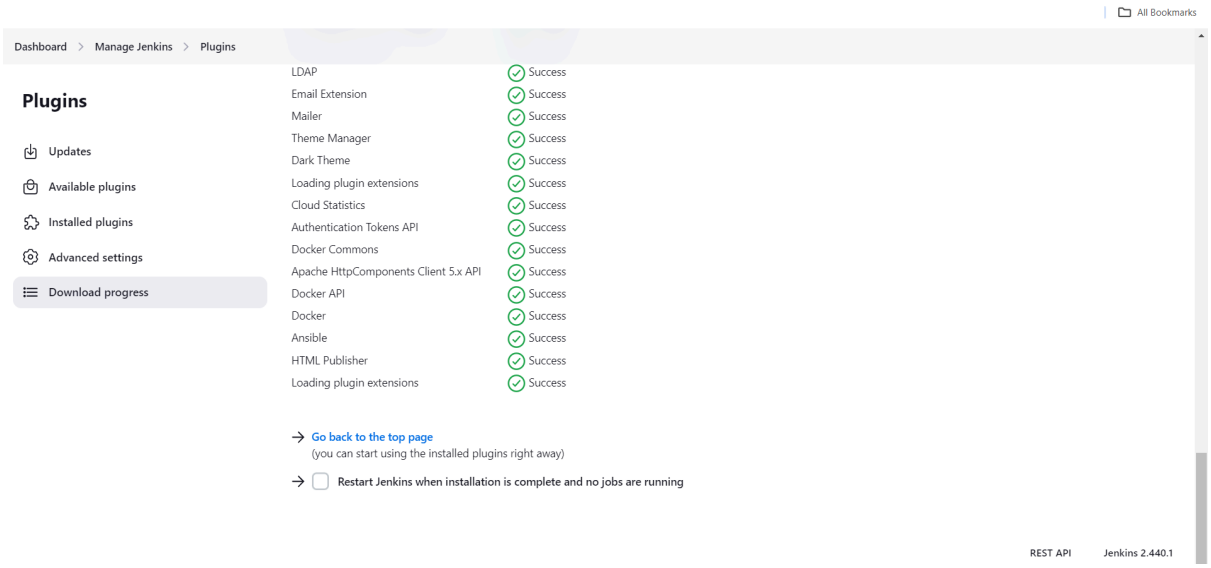

Not secure
3.80.219.219:8080
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.440.1
3.80.219.219:8080/#


Not secure
3.80.219.219:8080
All Bookmarks
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: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
** Pipeline: Job
** Jakarta Activation API
** Jakarta Mail API
** Apache HttpComponents Client 4.x API
Mailer
** Pipeline: Basic Steps
Gradle
** Pipeline: Milestone Step
** Pipeline: Build Step
** Variant
** Pipeline: Groovy Libraries
** Pipeline: Stage Step
** Joda Time API
** Pipeline: Model API
** Pipeline: Declarative Extension Points API
** Branch API
** Pipeline: Multibranch
** Pipeline: Stage Tags Metadata
** Mina SSHD API :: Common
** Mina SSHD API :: Core
** - required dependency
Jenkins 2.440.1



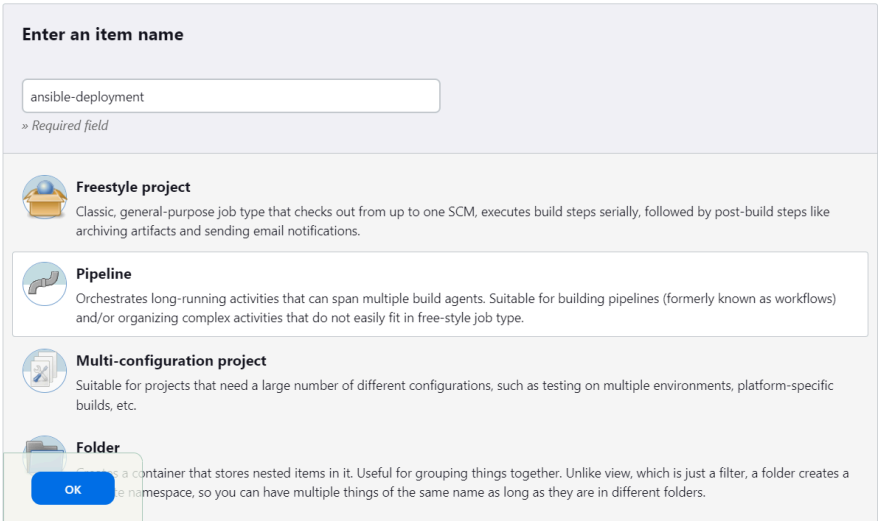
Installation of Plugins in Jenkins:



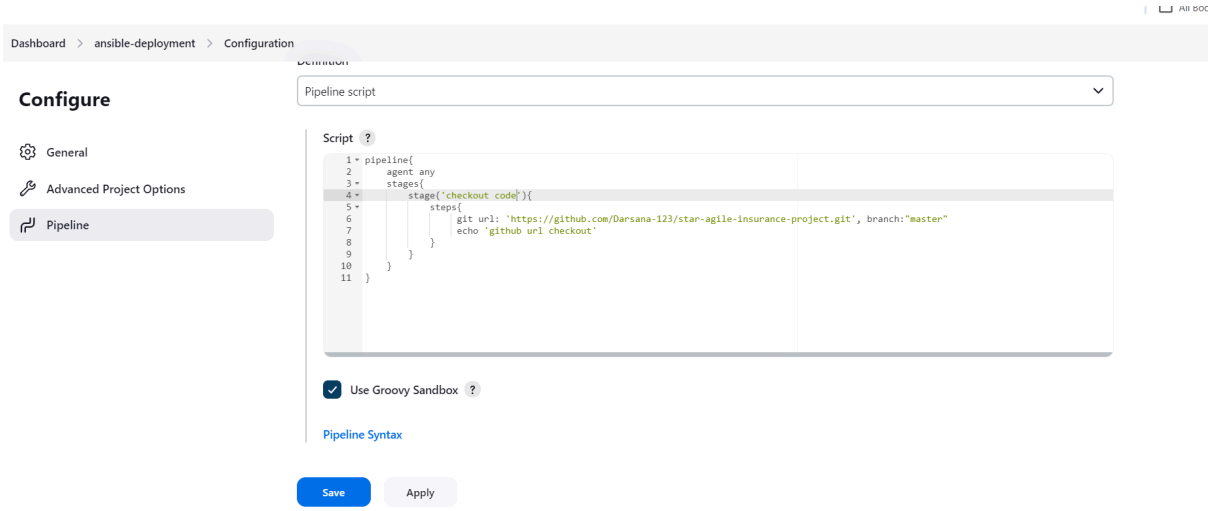


Deployment to Node-server using ANSIBLE

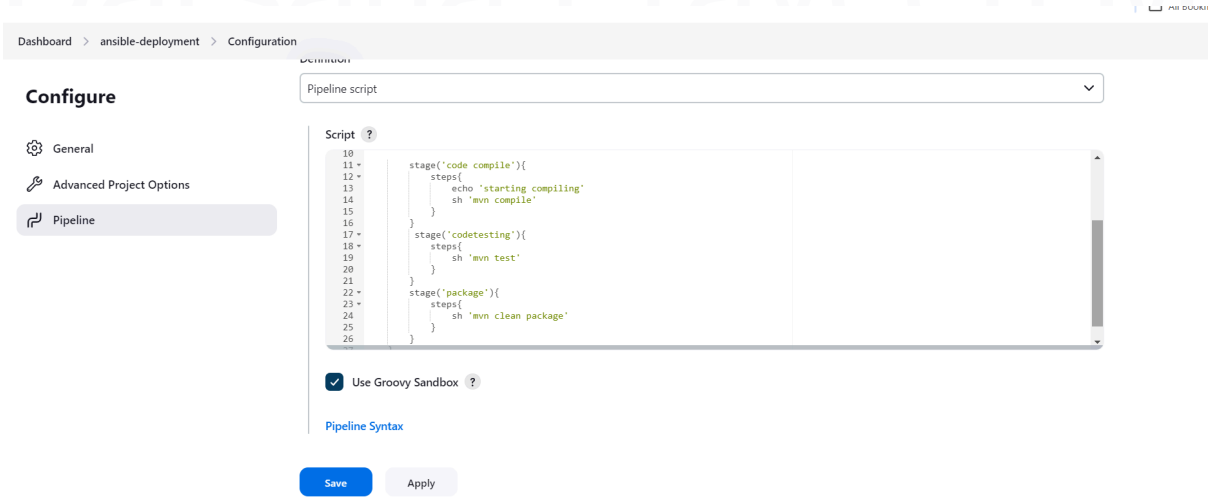
Create a pipeline

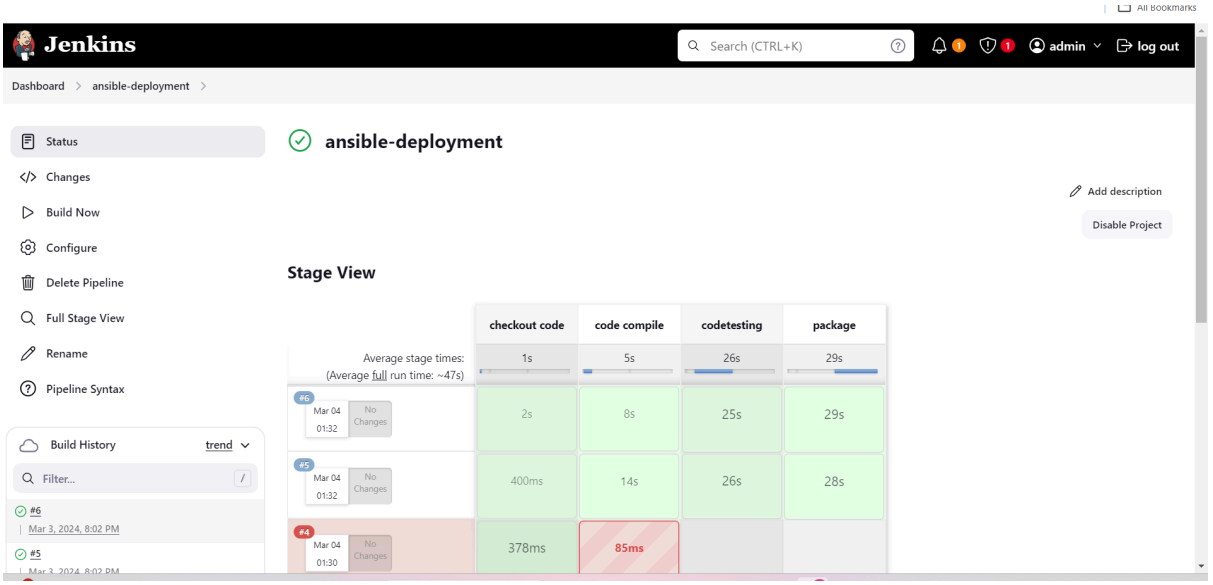


Code checkout from github

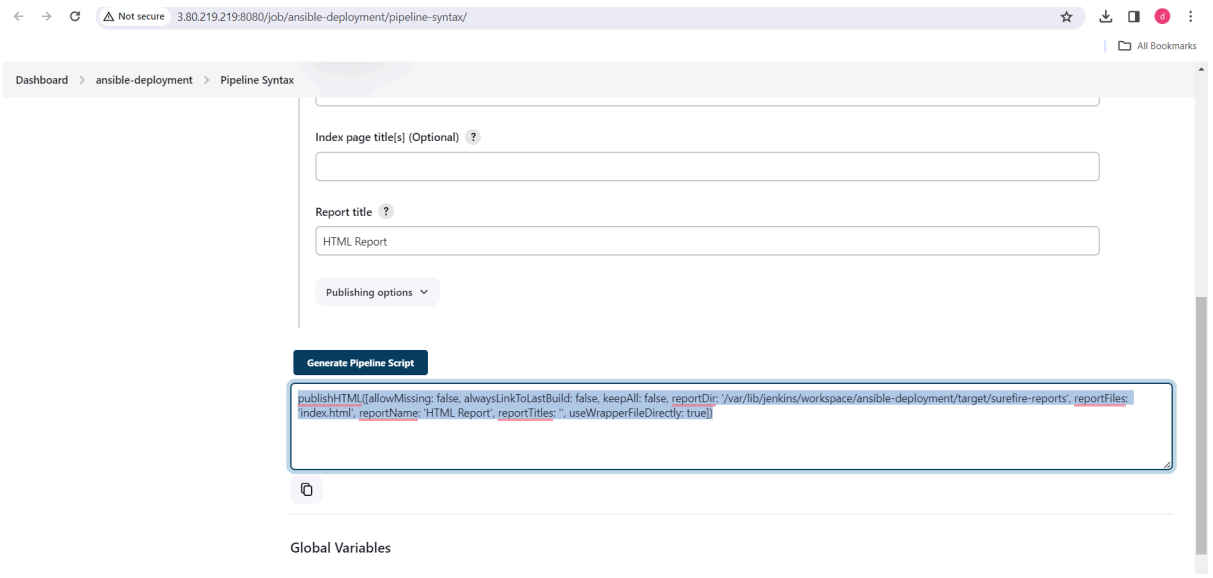


Code compile, testing, and package creation





Creating HTML Report:



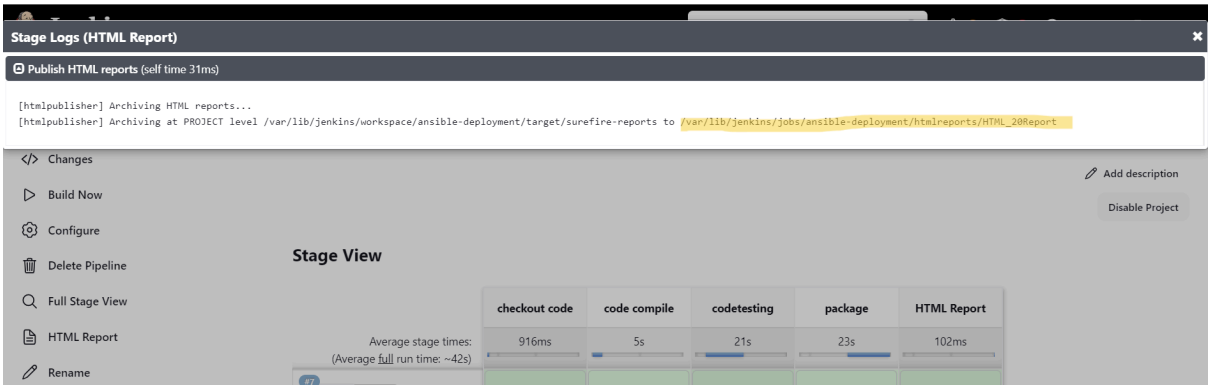
```
        }
    }

    stage('HTML Report'){
        steps{
            publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/a
        }
    }
}
}
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

HTML report is created in the below location



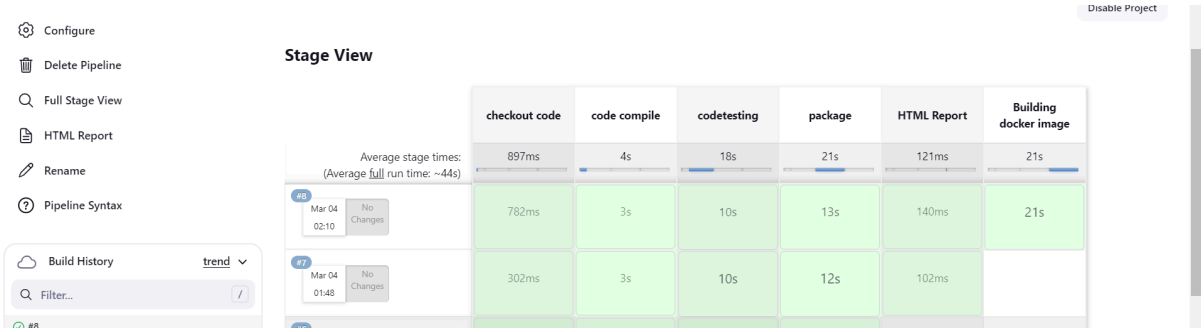
Building Docker image:

Giving sudo permissions to Jenkins

```
root@ip-172-31-28-64:/home/ubuntu# sudo usermod -aG docker jenkins
root@ip-172-31-28-64:/home/ubuntu# service jenkins restart
root@ip-172-31-28-64:/home/ubuntu#
```

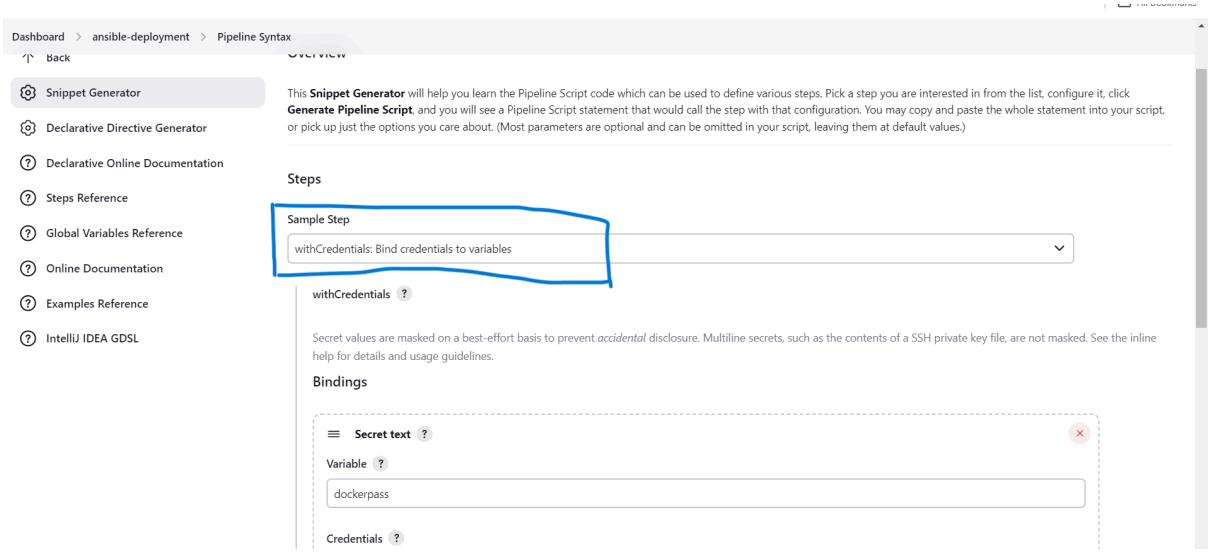
i-0e6616d4889b1de8f (master)

PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

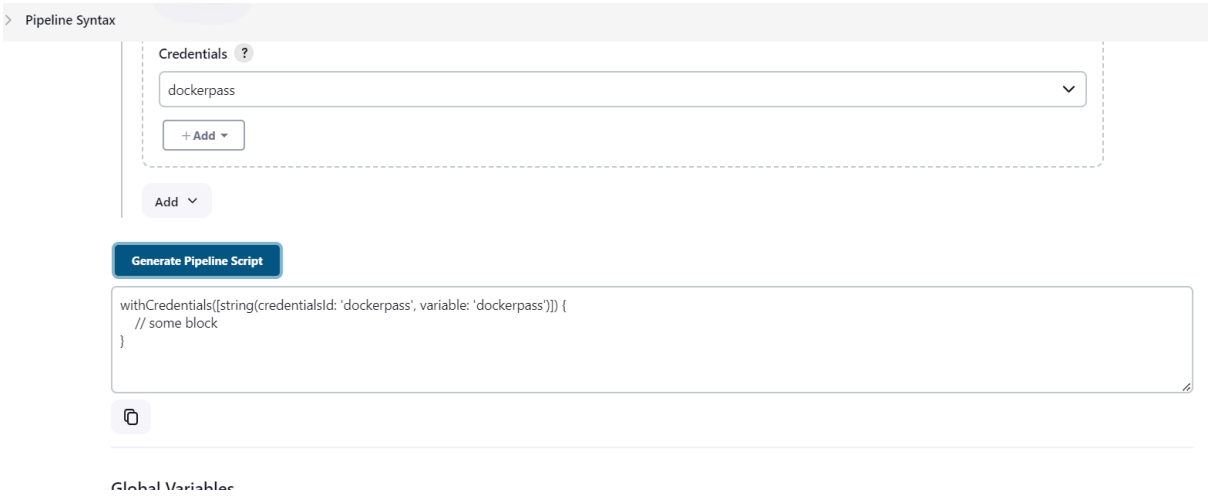


DockerHub Login & Pushing the image to Dockerhub

Set the docker hub credentials to Jenkins for accessing docker hub




ard > ansible-deployment > Pipeline Syntax
Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Secret text
Scope ?
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID ?
dockerpass
There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the snippet generator. See the Global Variable Reference for details.

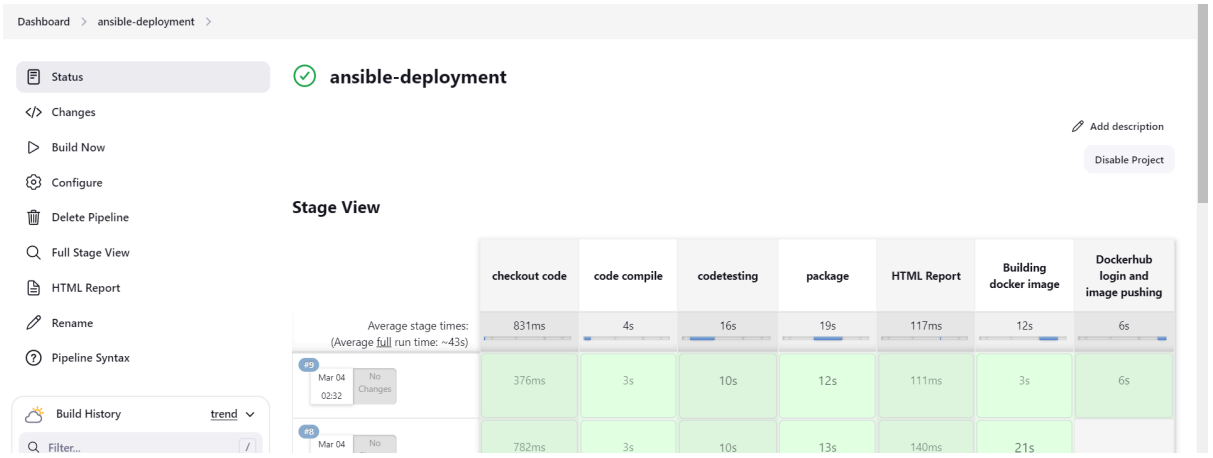

Pipeline Syntax
Credentials ?
dockerpass
+ Add
Add
Generate Pipeline Script
withCredentials([string(credentialsId: 'dockerpass', variable: 'dockerpass')]) {
// some block
}
Global Variables

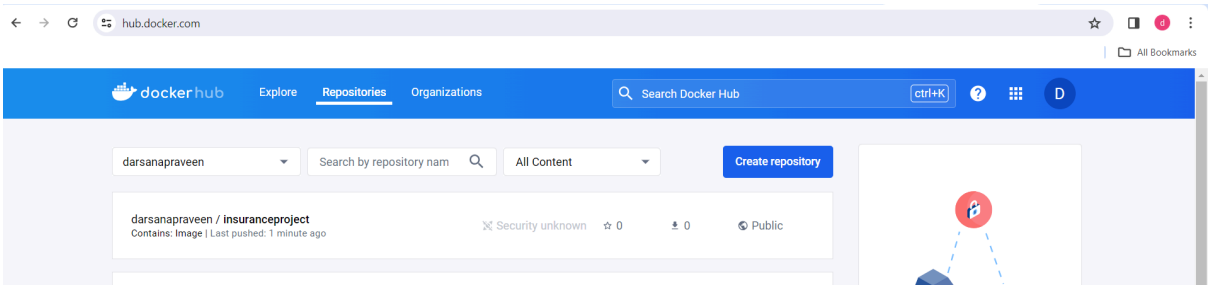
Script ?

```
        }
    }

    stage('Dockerhub login and image pushing'){
        steps{
            withCredentials([string(credentialsId: 'dockerpass', variable: 'dockerpass')]) {
                sh " docker login -u darsanapraveen -p ${dockerpass}"
                sh 'docker push darsanapraveen/insuranceproject:1.0 '
            }
        }
    }
  }
}
```

Use Groovy Sandbox ?

Pipeline Syntax





Set up the host in master

```
root@ip-172-31-28-64:/home/ubuntu# cd /etc/ansible
root@ip-172-31-28-64:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ip-172-31-28-64:/etc/ansible# vi hosts
root@ip-172-31-28-64:/etc/ansible#
```

i-0e6616d4889b1de8f (master)

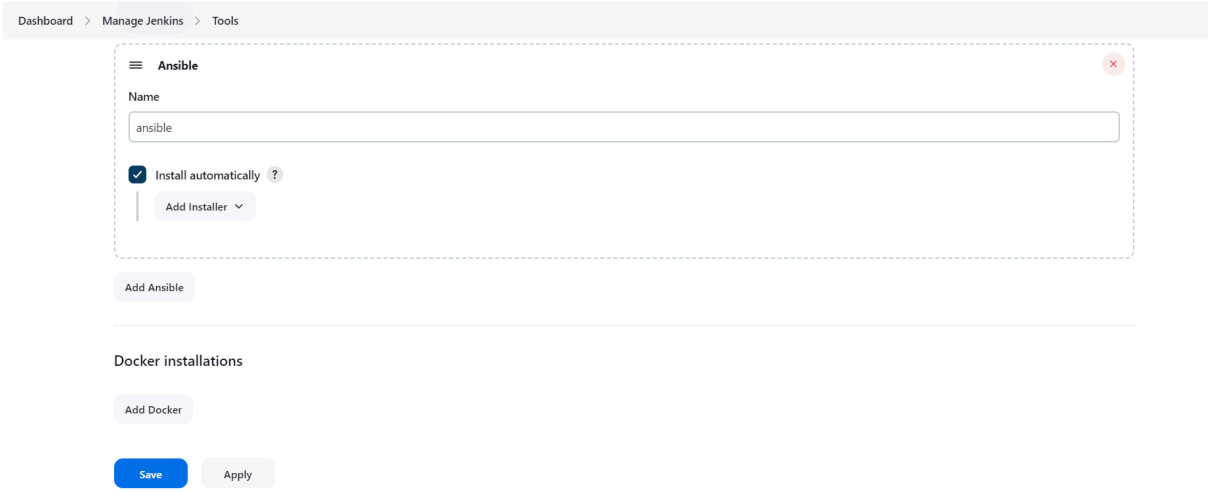
PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

```
# 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
[demo]
18.207.128.252
-- INSERT --
```

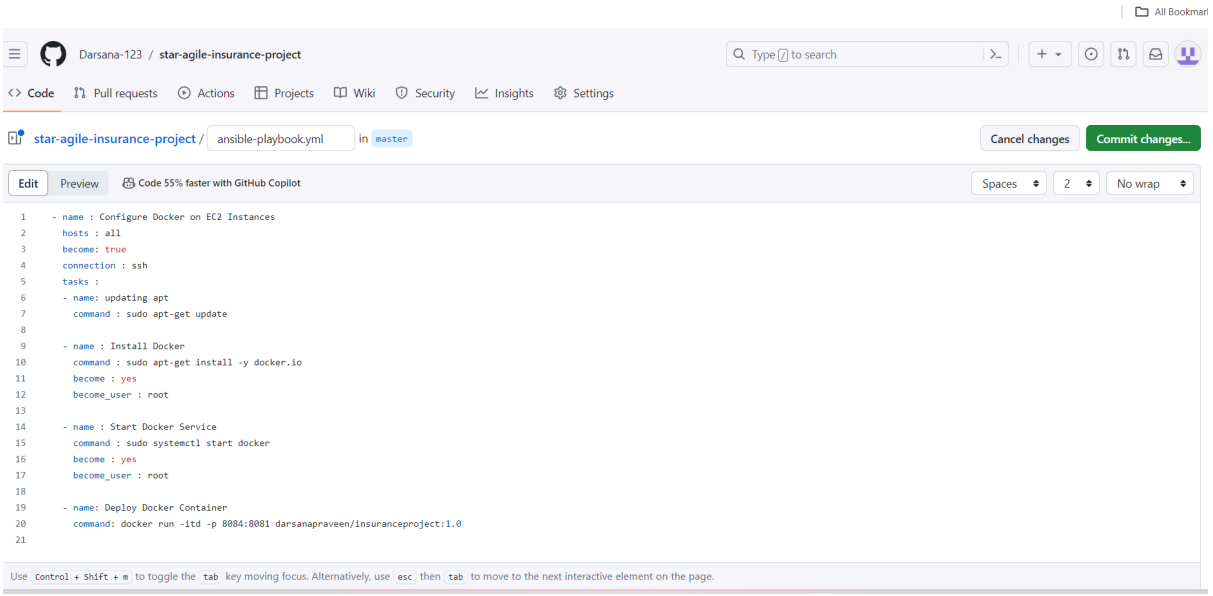
i-0e6616d4889b1de8f (master)

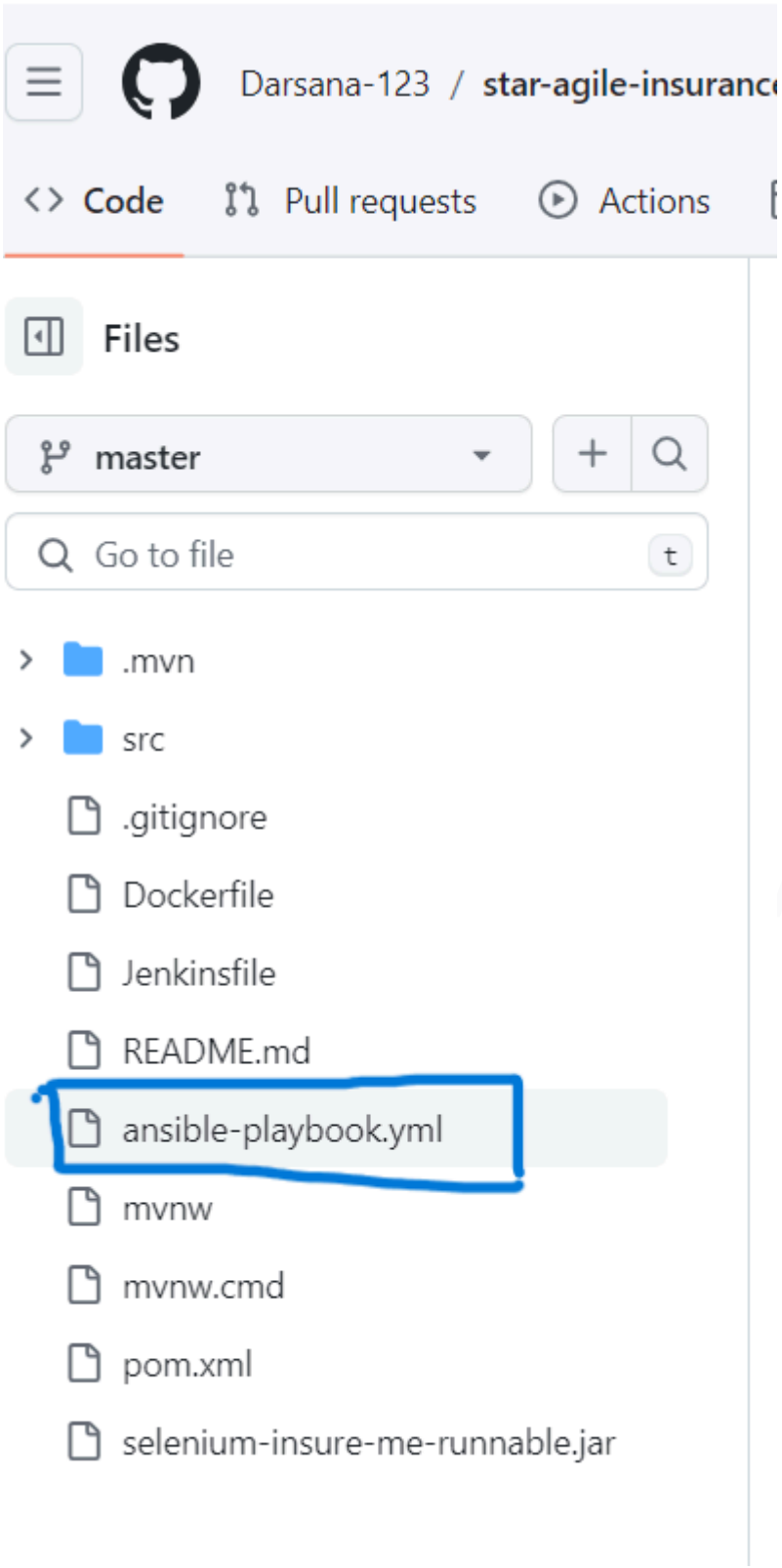
PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

Setting up the jenkins tool

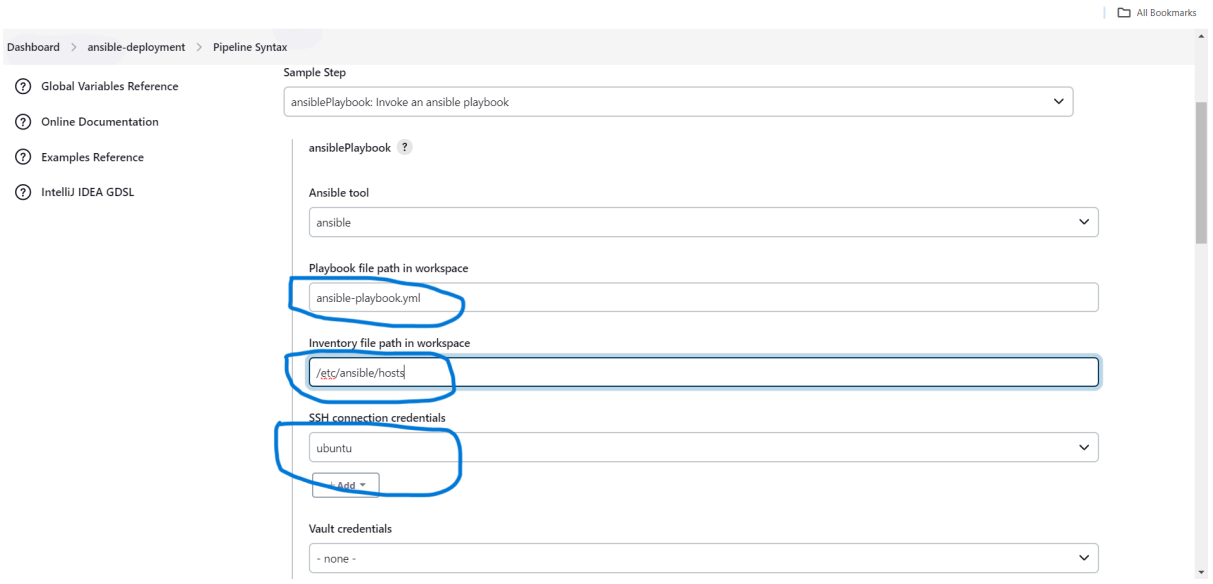


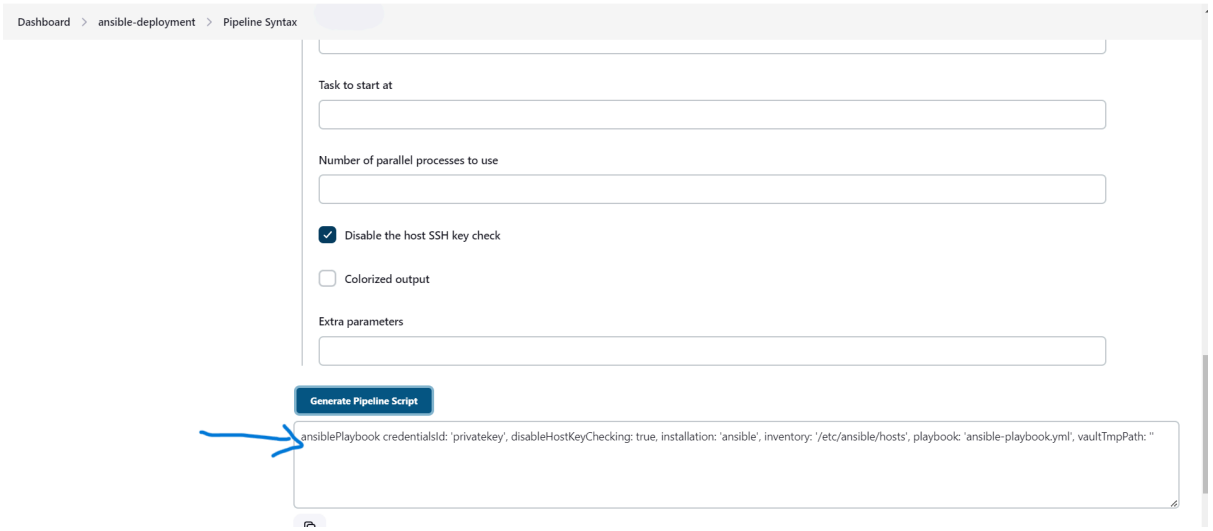
Creating an Ansible playbook file in Github repository




Darsana-123 / star-agile-insuranc
Code
Pull requests
Actions
Files
master
Go to file
t
.mvn
src
.gitignore
Dockerfile
Jenkinsfile
README.md
ansible-playbook.yml
mvnw
mvnw.cmd
pom.xml
selenium-insure-me-runnable.jar

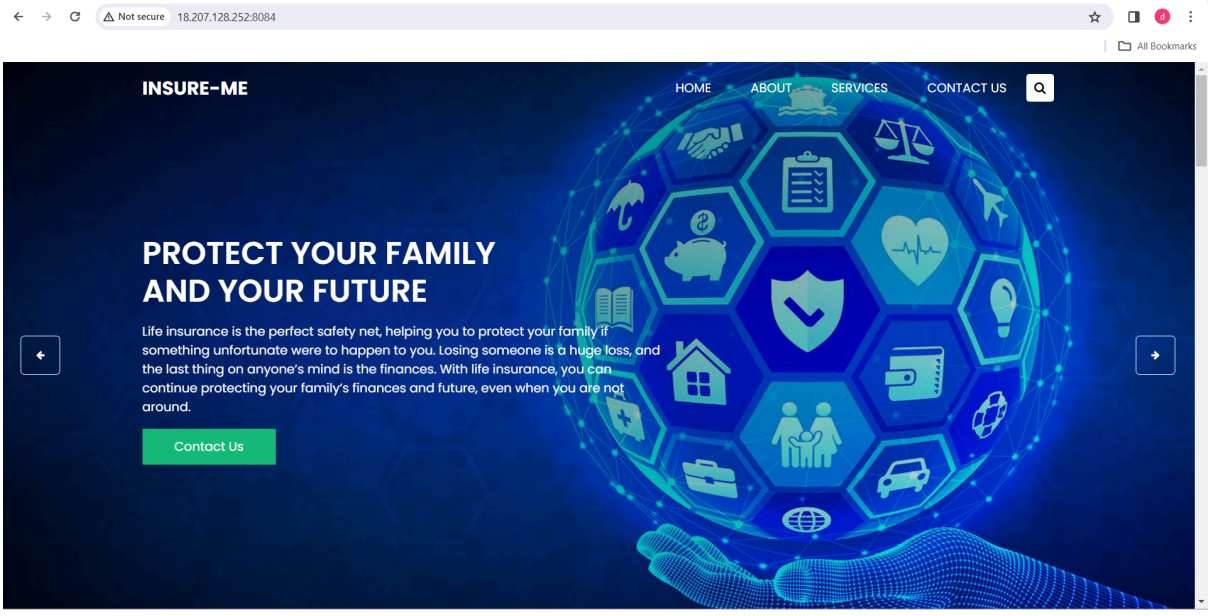
Creating the SSH Credentials of node-server and giving to Ansible





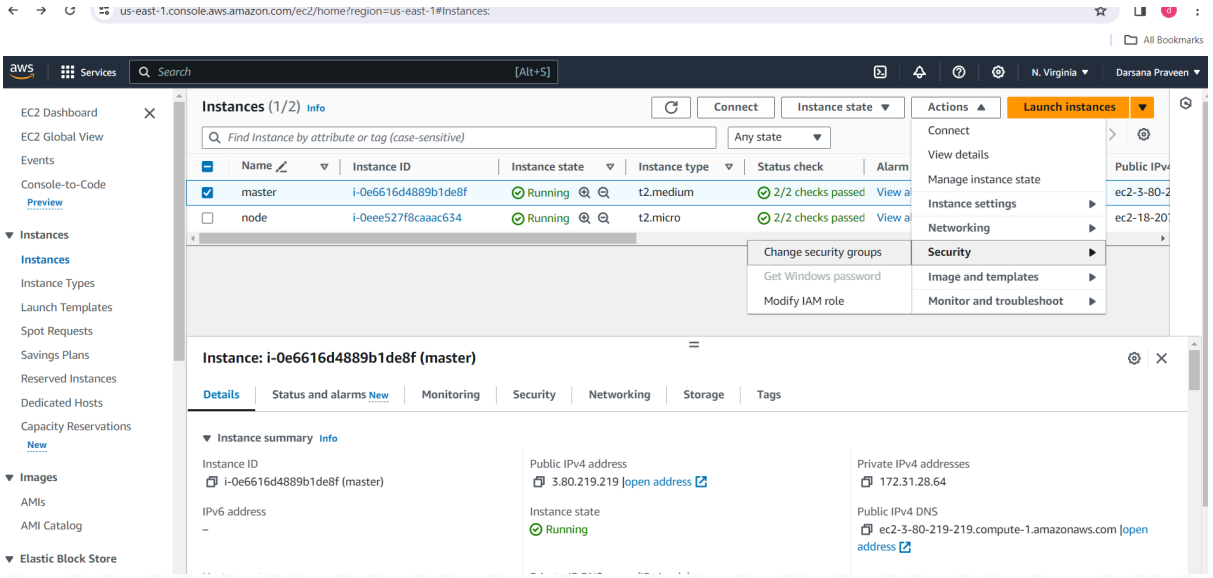
Ansible Deployment

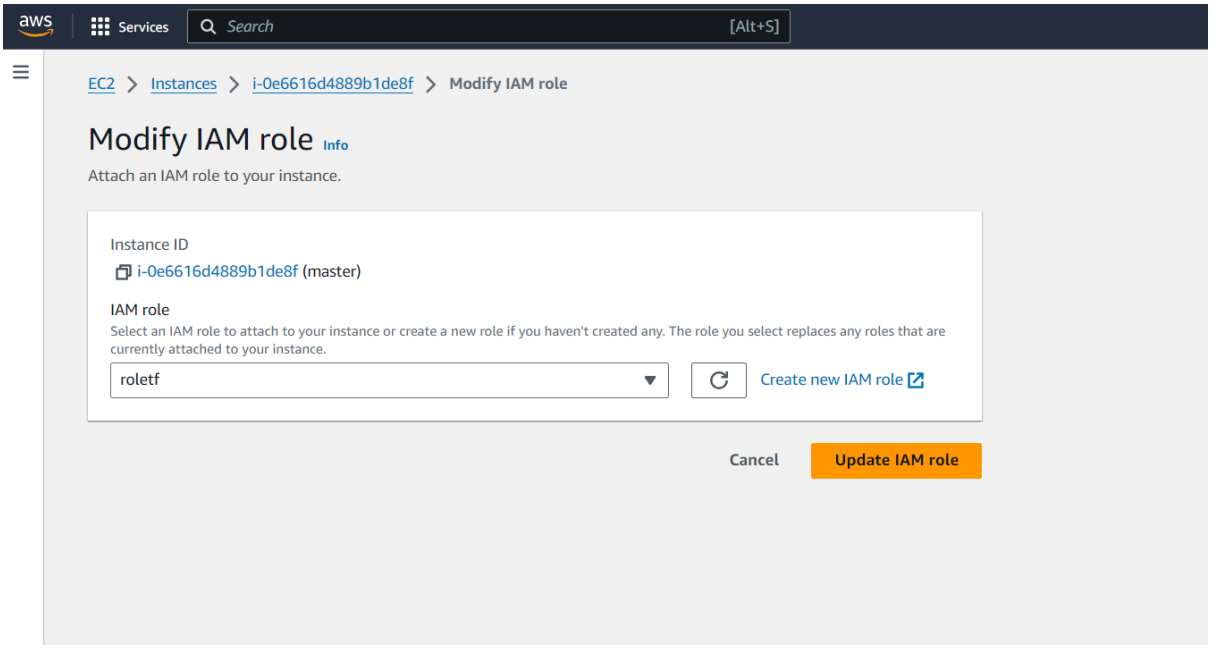
Open node-server and copy the publicIP and give the port number



Terraform deployment:

We have to give the EC2 full access permission to our master machine using the IAM role.





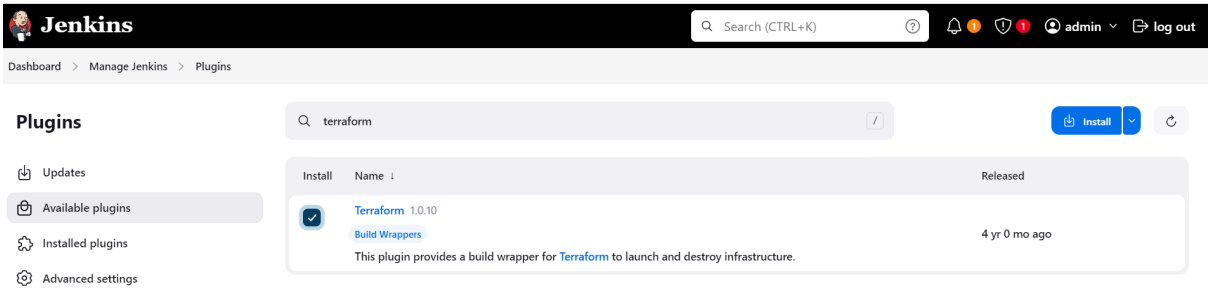
Terraform installation:

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-28-64:/home/ubuntu# terraform --version
Terraform v1.7.4
on linux_amd64
root@ip-172-31-28-64:/home/ubuntu#
```

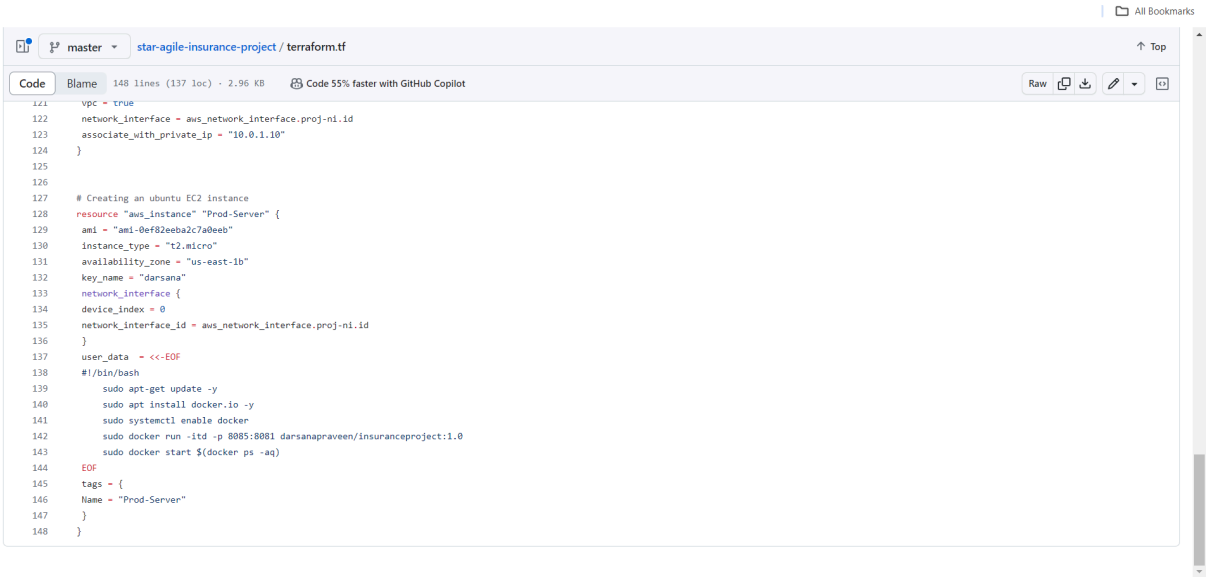
i-0e6616d4889b1de8f (master)

PublicIPs: 3.80.219.219 PrivateIPs: 172.31.28.64

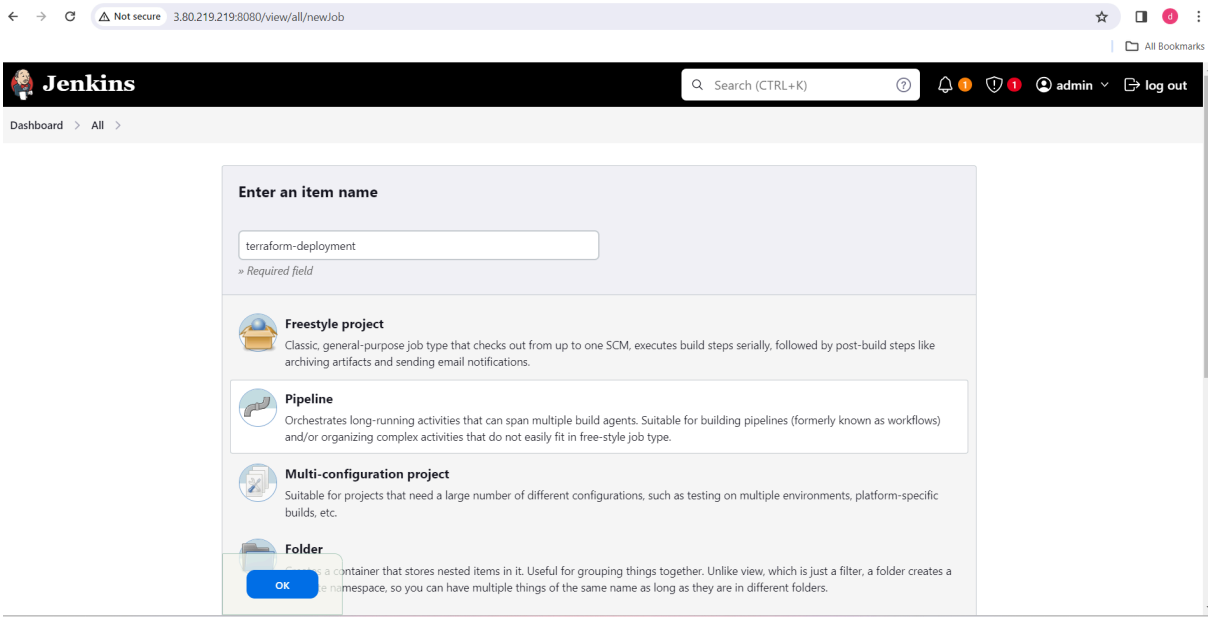
Plugin installation on Jenkins:

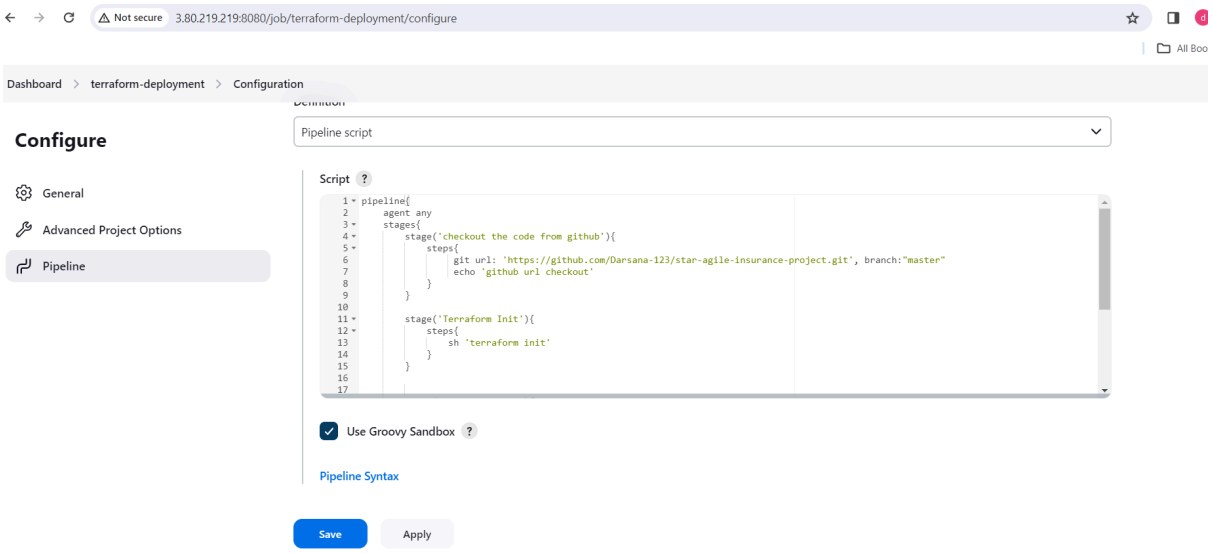


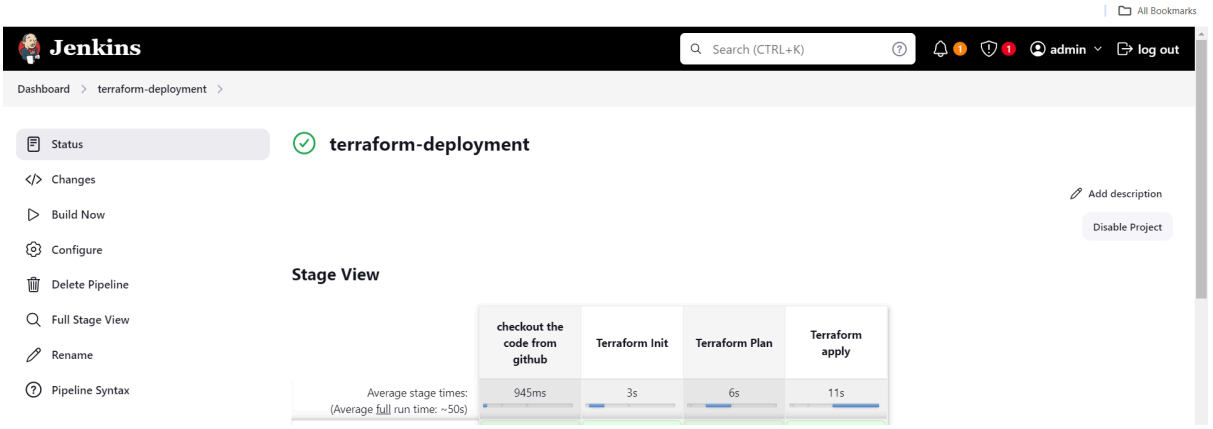
We have to create a terraform file to our Github:

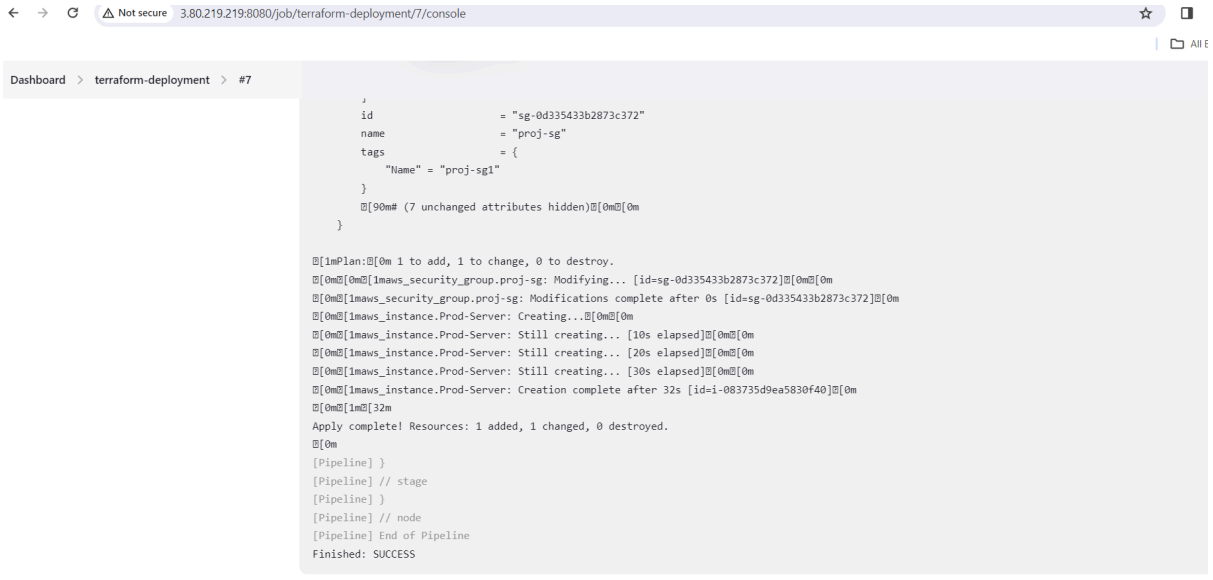


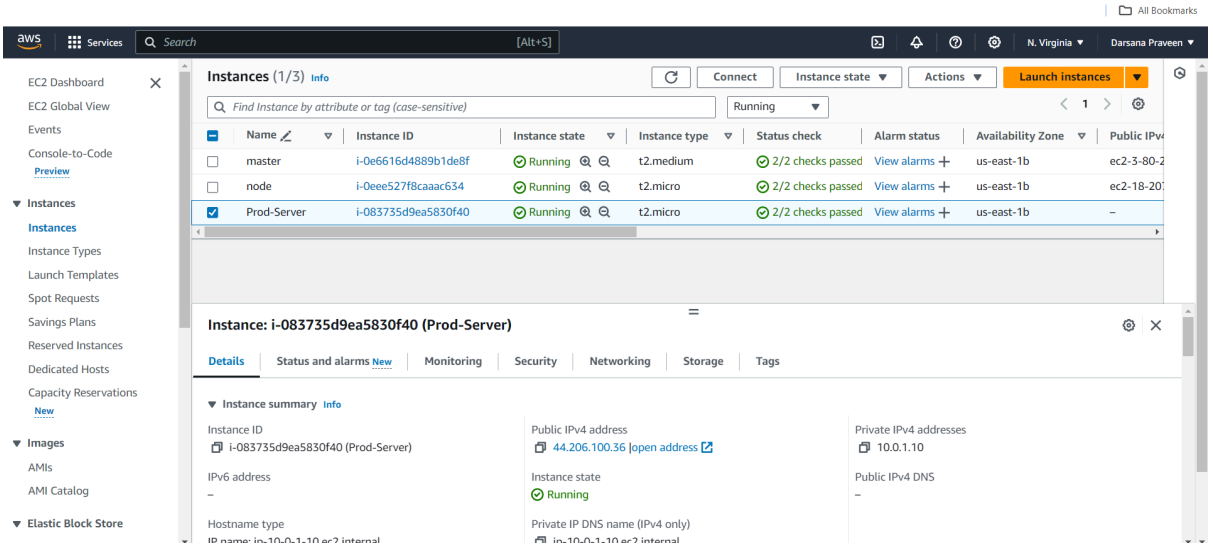
Create a pipeline for terraform deployment:



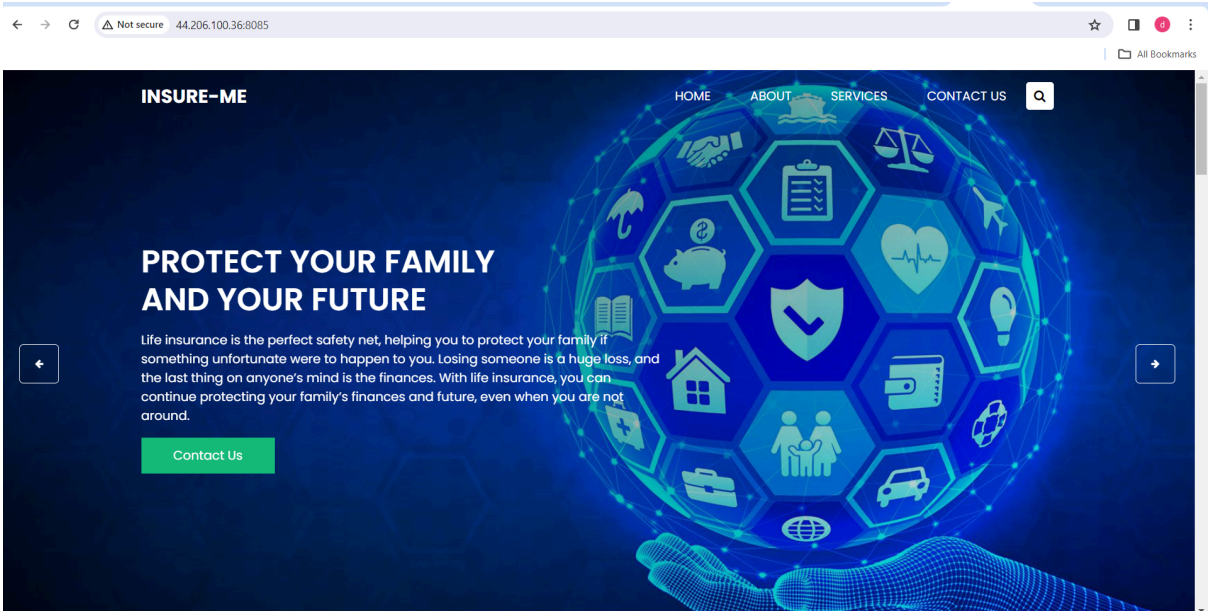

Not secure 3.80.219.219:8080/job/terraform-deployment/configure
Dashboard > terraform-deployment > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline script
Script
pipeline{
agent any
stages{
stage('checkout the code from github'){
steps{
git url: 'https://github.com/Darsana-123/star-agile-insurance-project.git', branch:"master"
echo 'github url checkout'
}
}
stage('Terraform Init'){
steps{
sh 'terraform init'
}
}
Use Groovy Sandbox
Pipeline Syntax
Save
Apply


Jenkins
Search (CTRL+K)
admin
log out
Dashboard > terraform-deployment >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
terraform-deployment
Add description
Disable Project
Stage View
checkout the code from github
Terraform Init
Terraform Plan
Terraform apply
Average stage times:
(Average full run time: ~50s)
945ms
3s
6s
11s

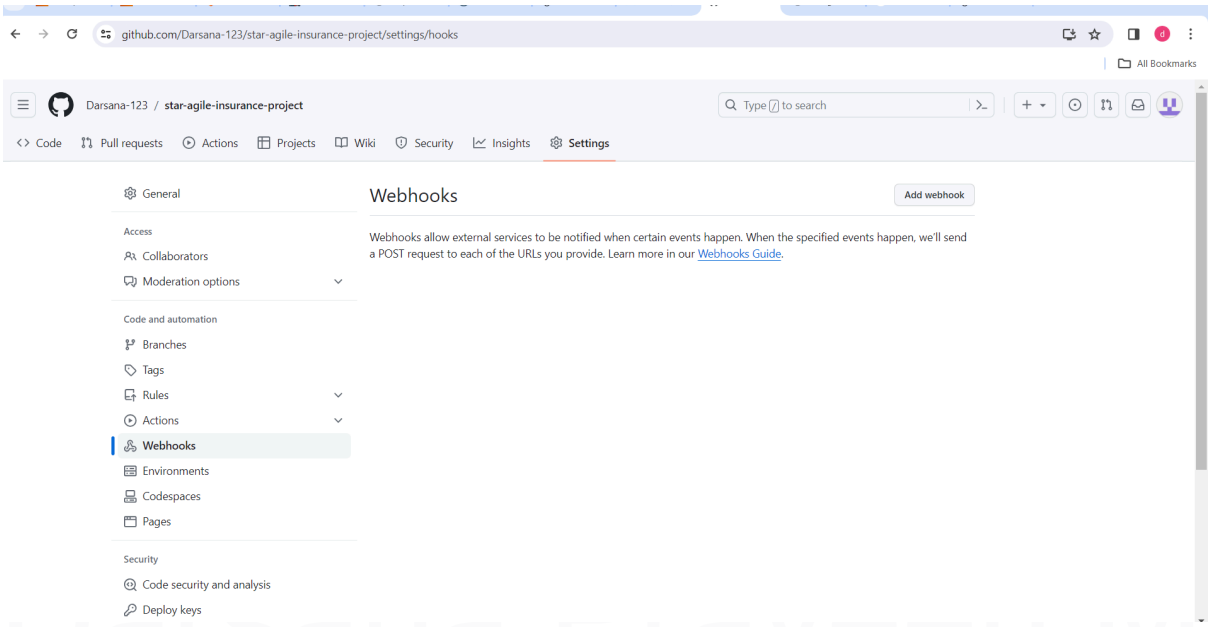

Not secure 3.80.219.219:8080/job/terraform-deployment/7/console
Dashboard > terraform-deployment > #7
id = "sg-0d335433b2873c372"
name = "proj-sg"
tags = {
"Name" = "proj-sg1"
}
[90m# (7 unchanged attributes hidden)[0m[0m
}
[1mPlan:[0m 1 to add, 1 to change, 0 to destroy.
[0m[0m[1maws_security_group.proj-sg: Modifying... [id=sg-0d335433b2873c372][0m[0m
[0m[1maws_security_group.proj-sg: Modifications complete after 0s [id=sg-0d335433b2873c372][0m
[0m[1maws_instance.Prod-Server: Creating...[0m[0m
[0m[1maws_instance.Prod-Server: Still creating... [10s elapsed][0m[0m
[0m[1maws_instance.Prod-Server: Still creating... [20s elapsed][0m[0m
[0m[1maws_instance.Prod-Server: Still creating... [30s elapsed][0m[0m
[0m[1maws_instance.Prod-Server: Creation complete after 32s [id=i-083735d9ea5830f40][0m
[0m[1m[32m
Apply complete! Resources: 1 added, 1 changed, 0 destroyed.
[0m
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

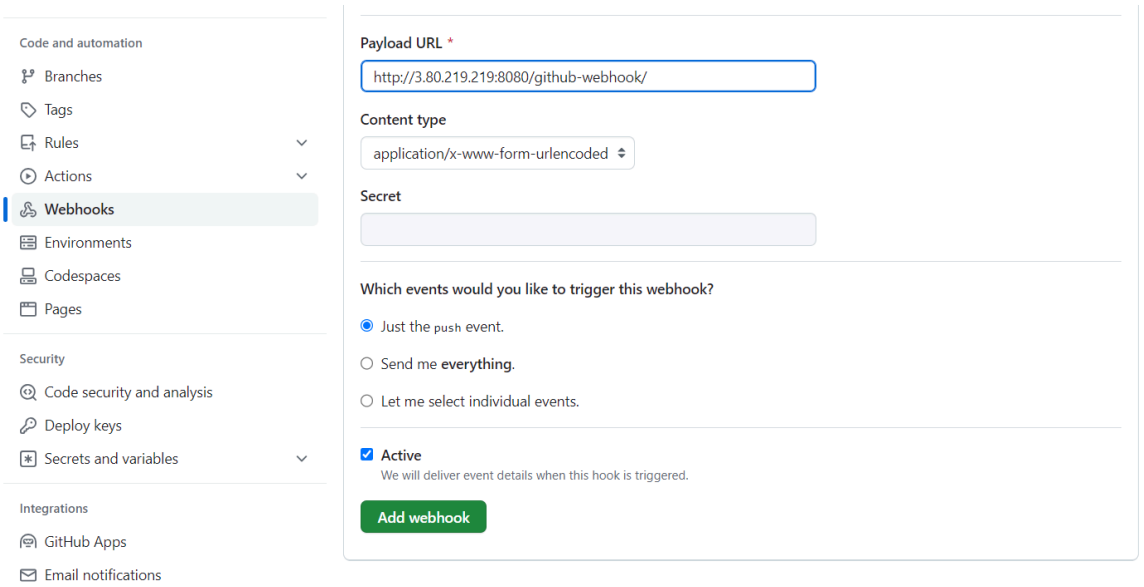


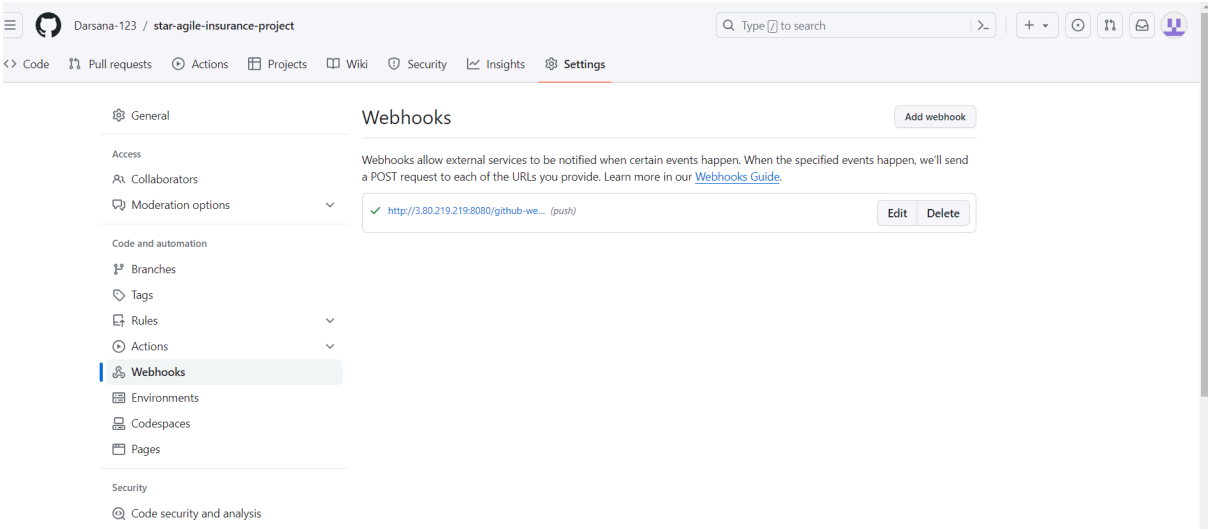
Opening the application through the port we used to expose in terraform file



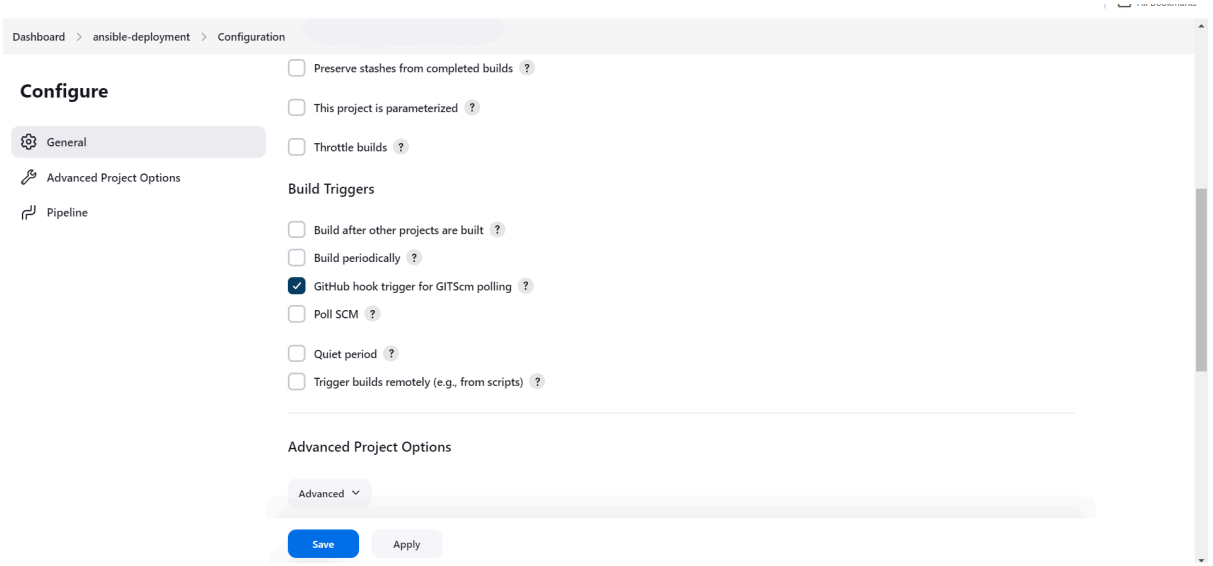
Creating a WEBHOOK to our repo for the automatic triggering actions

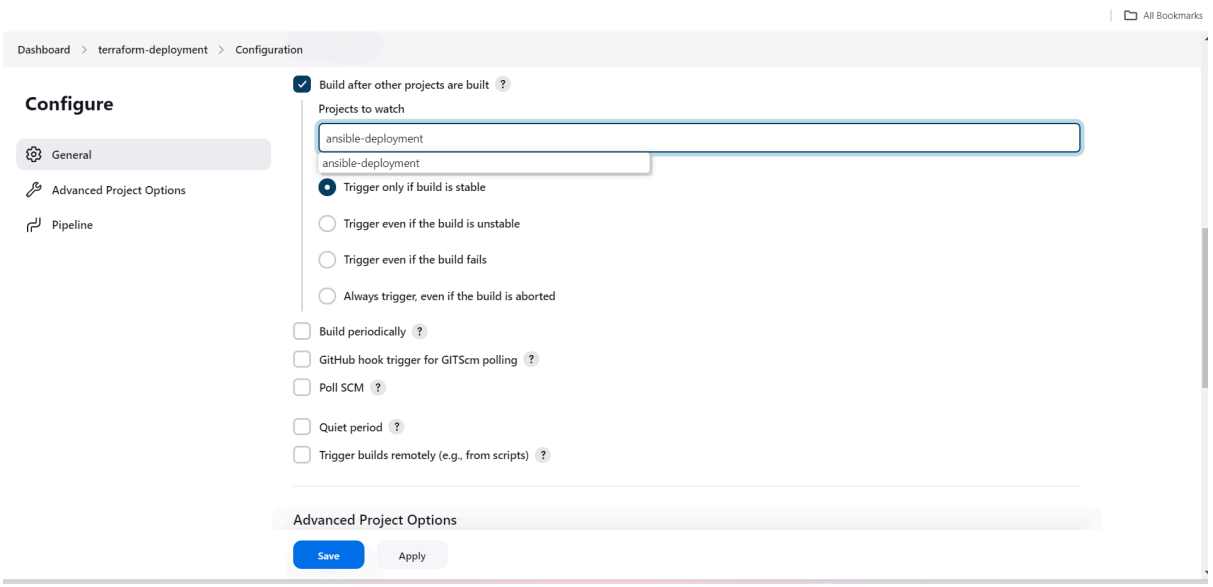






Setting up webhook to the job1





Verifying the automation of workflow: