CAPSTONE PROJECT

Insurance Project

SUBMITTED BY: SALMAN SADIQ

DATE:02/03/2024

INSURANCE PROJECT DETAILS

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

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

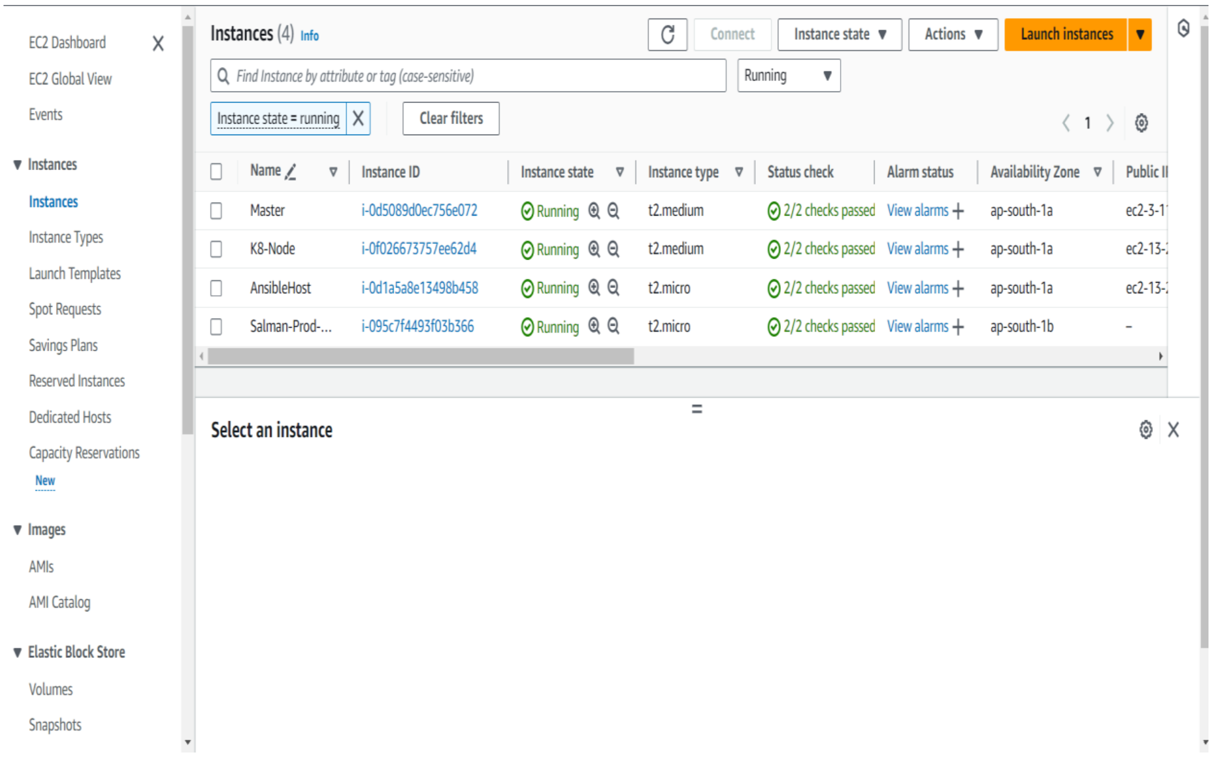
✓ Docker - For deploying containerized applications

✓ 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

1) AWS EC2 Machines



EC2 Dashboard
EC2 Global View
Events
▼ Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
▼ Images
AMIs
AMI Catalog
▼ Elastic Block Store
Volumes
Snapshots

Instances (4) Info

Find Instance by attribute or tag (case-sensitive) | Running

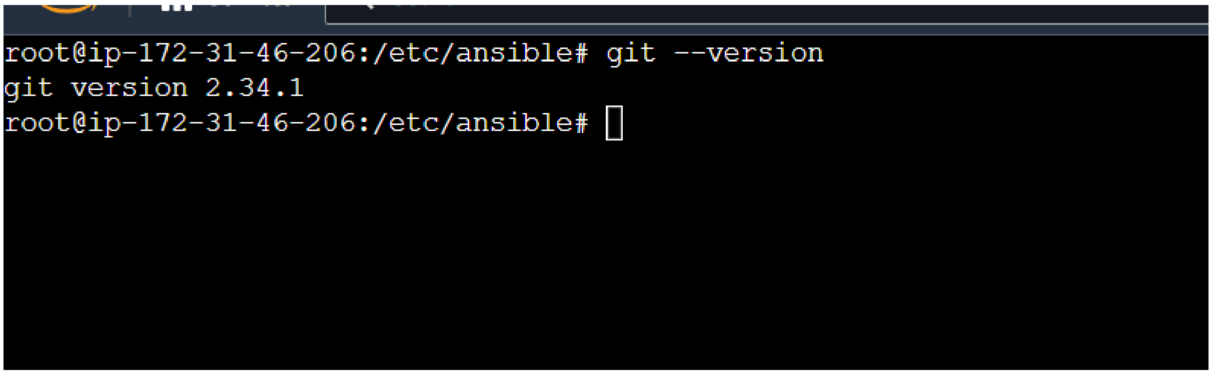
Instance state = running | Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
Master	i-0d5089d0ec756e072	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-3-1
K8-Node	i-0f026673757ee62d4	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-
AnsibleHost	i-0d1a5a8e13498b458	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-
Salman-Prod-...	i-095c7f4493f03b366	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	–

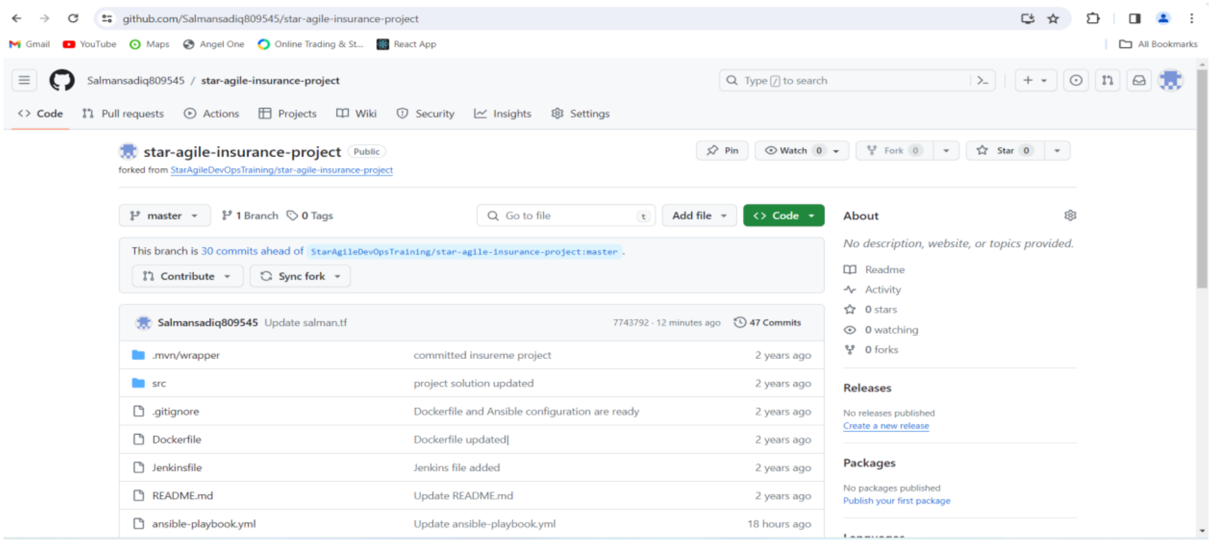
Select an instance

Here 4 Machines have been created for Ansible , Terraform, Kubernetes Deployment

2)Git Configuration



```
root@ip-172-31-46-206:/etc/ansible# git --version
git version 2.34.1
root@ip-172-31-46-206:/etc/ansible# 
```



github.com/Salmansadiq809545/star-agile-insurance-project

Salmansadiq809545 / star-agile-insurance-project

star-agile-insurance-project Public
forked from StarAgileDevOpsTraining/star-agile-insurance-project

master · 1 Branch · 0 Tags

This branch is 30 commits ahead of StarAgileDevOpsTraining/star-agile-insurance-project:master .

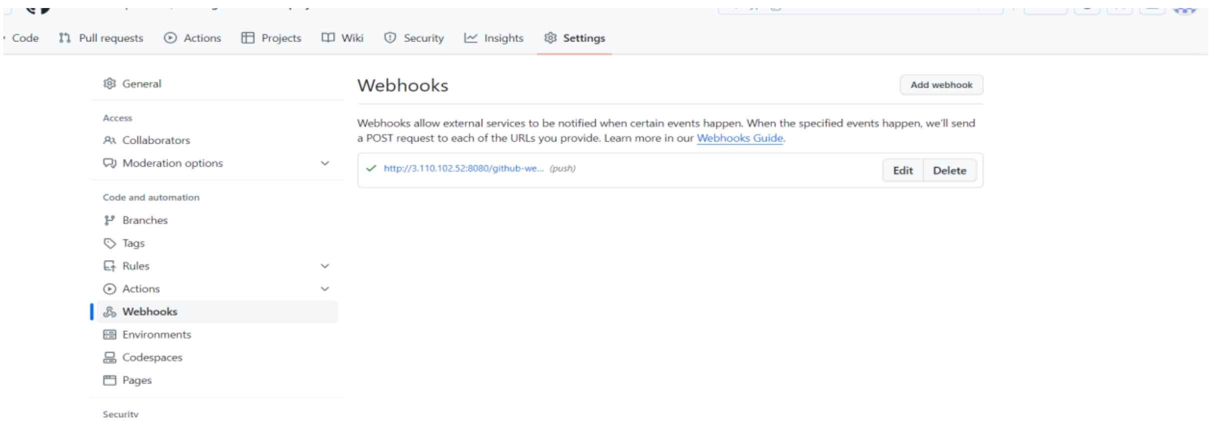
Salmansadiq809545 Update salman.tf · 7743792 · 12 minutes ago · 47 Commits

```
.mvn/wrapper          committed insureme project                        2 years ago
src                   project solution updated                          2 years ago
.gitignore            Dockerfile and Ansible configuration are ready    2 years ago
Dockerfile            Dockerfile updated|                               2 years ago
Jenkinsfile           Jenkins file added                                2 years ago
README.md             Update README.md                                  2 years ago
ansible-playbook.yml  Update ansible-playbook.yml                       18 hours ago
```

About
No description, website, or topics provided.
Readme
Activity
0 stars
0 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package



Webhooks

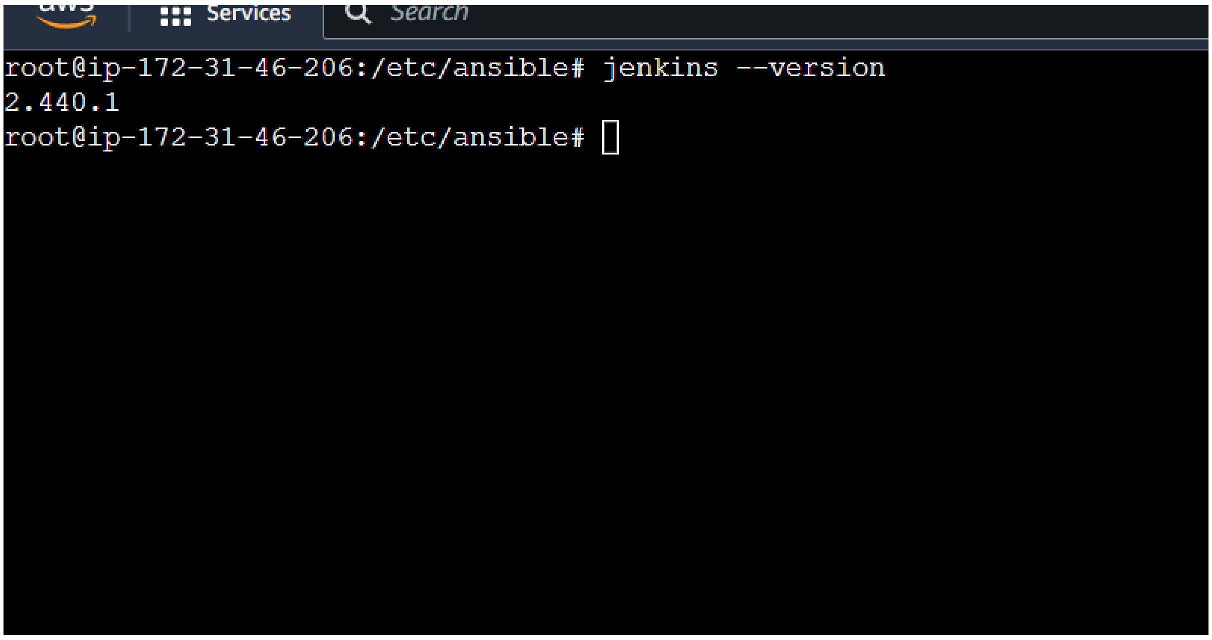
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://3.110.102.52:8080/github-we... (push)

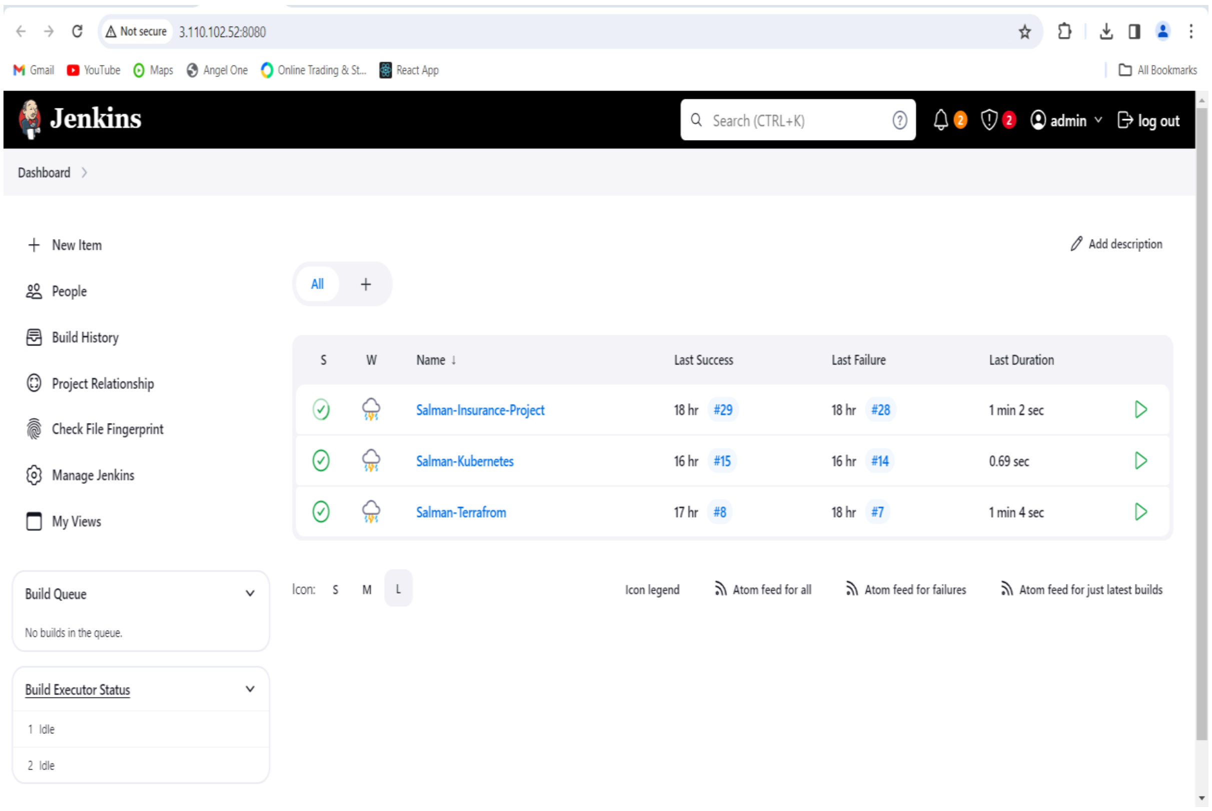
Here we can see the git repo from where the code has been taken

And git has been installed in Master Machine with web Hook configured for triggering automatic deployment.

3) Jenkins Installation and configuration of 3 pipelines



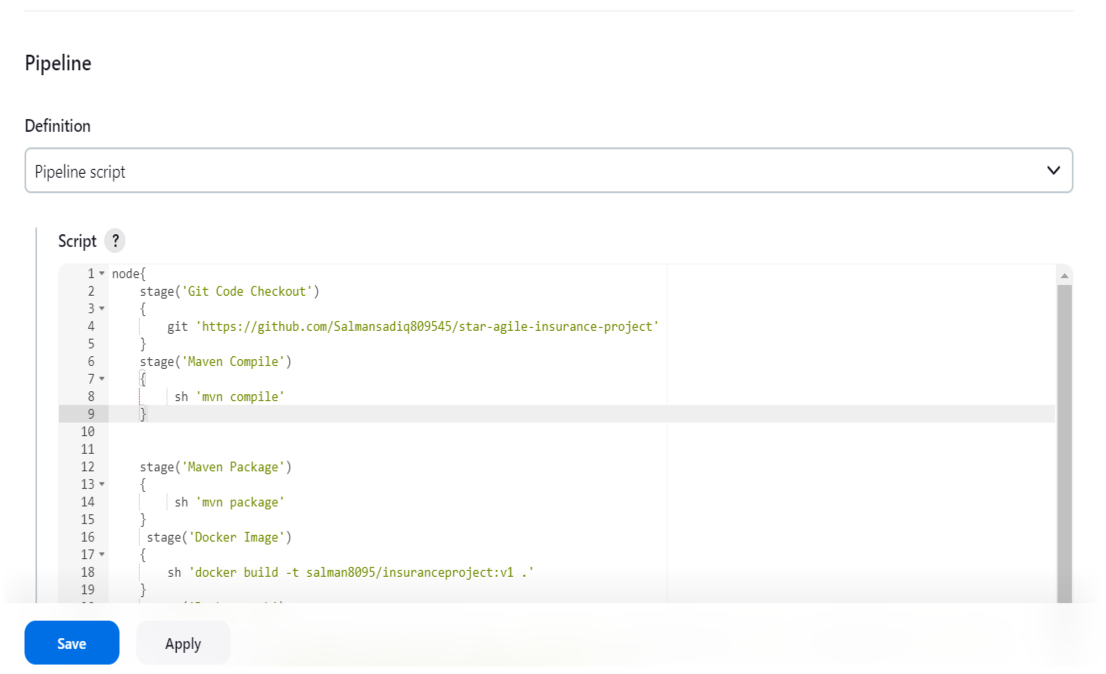
```
root@ip-172-31-46-206:/etc/ansible# jenkins --version
2.440.1
root@ip-172-31-46-206:/etc/ansible#
```



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		Salman-Insurance-Project	18 hr #29	18 hr #28	1 min 2 sec
✓		Salman-Kubernetes	16 hr #15	16 hr #14	0.69 sec
✓		Salman-Terrafrom	17 hr #8	18 hr #7	1 min 4 sec

Here we have installed Jenkins and configured 3 pipelines for Ansible , Terraform, and Kubernetes

Ansible Jenkins Pipeline



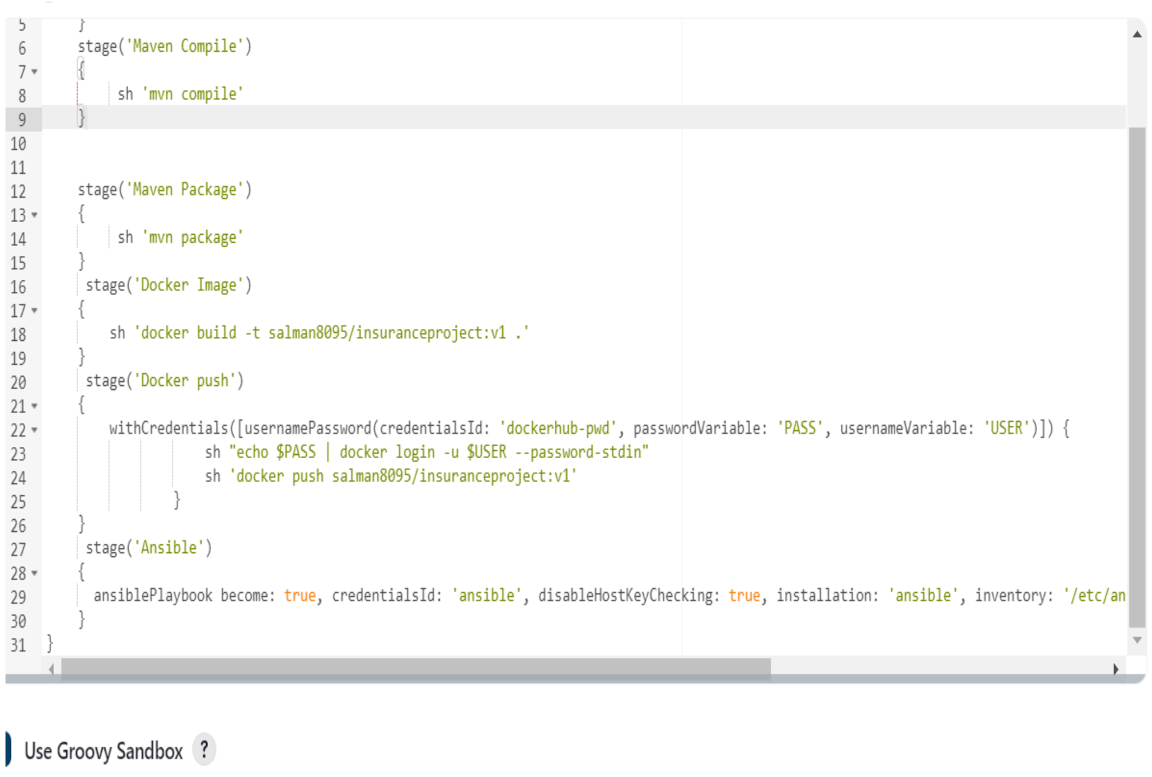
Pipeline

Definition

Pipeline script

Script ?

```
node{
    stage('Git Code Checkout')
    {
        git 'https://github.com/Salmansadiq809545/star-agile-insurance-project'
    }
    stage('Maven Compile')
    {
         sh 'mvn compile'
    }


    stage('Maven Package')
    {
         sh 'mvn package'
    }
     stage('Docker Image')
    {
        sh 'docker build -t salman8095/insuranceproject:v1 .'
    }
```

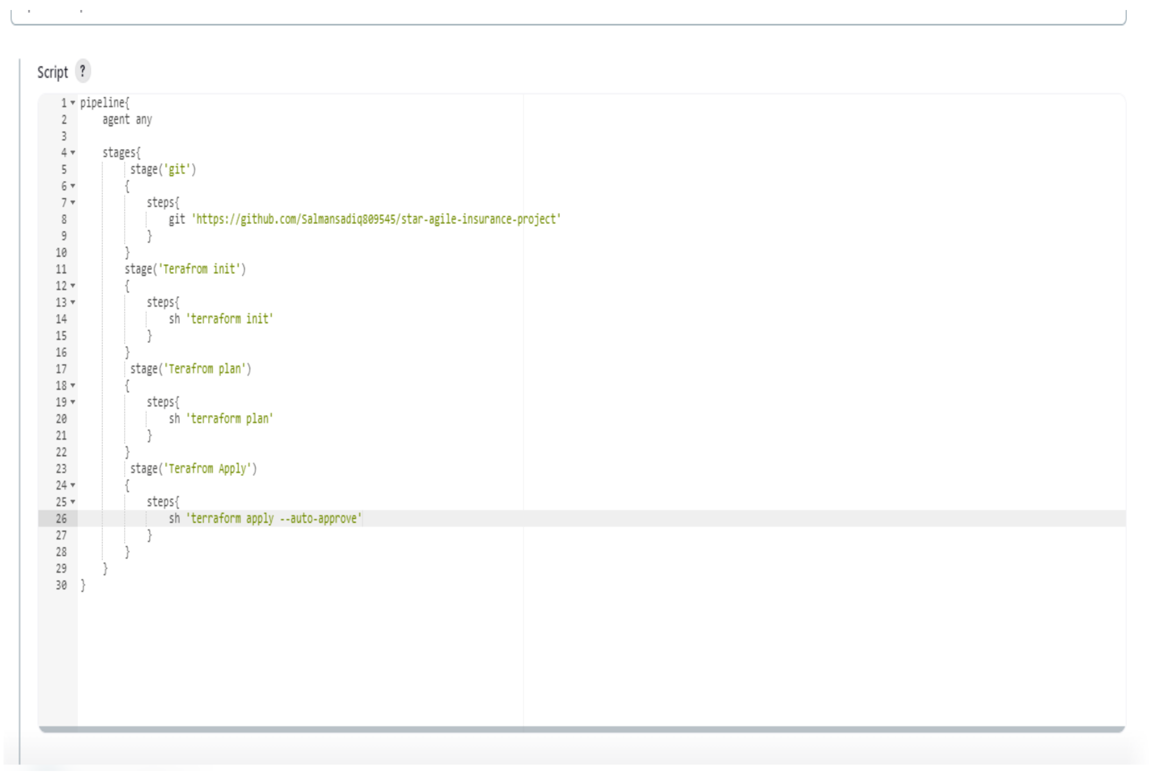


```
    }
    stage('Maven Compile')
    {
         sh 'mvn compile'
    }


    stage('Maven Package')
    {
         sh 'mvn package'
    }
     stage('Docker Image')
    {
        sh 'docker build -t salman8095/insuranceproject:v1 .'
    }
     stage('Docker push')
    {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-pwd', passwordVariable: 'PASS', usernameVariable: 'USER')]) {
                sh "echo $PASS | docker login -u $USER --password-stdin"
                sh 'docker push salman8095/insuranceproject:v1'
            }
    }
     stage('Ansible')
    {
      ansiblePlaybook become: true, credentialsId: 'ansible', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/an
    }
}
```

Use Groovy Sandbox ?

Here is the Ansible deployment pipeline where we deploy the containerized Application using Docker at port 8088

Terraform Jenkins Pipeline

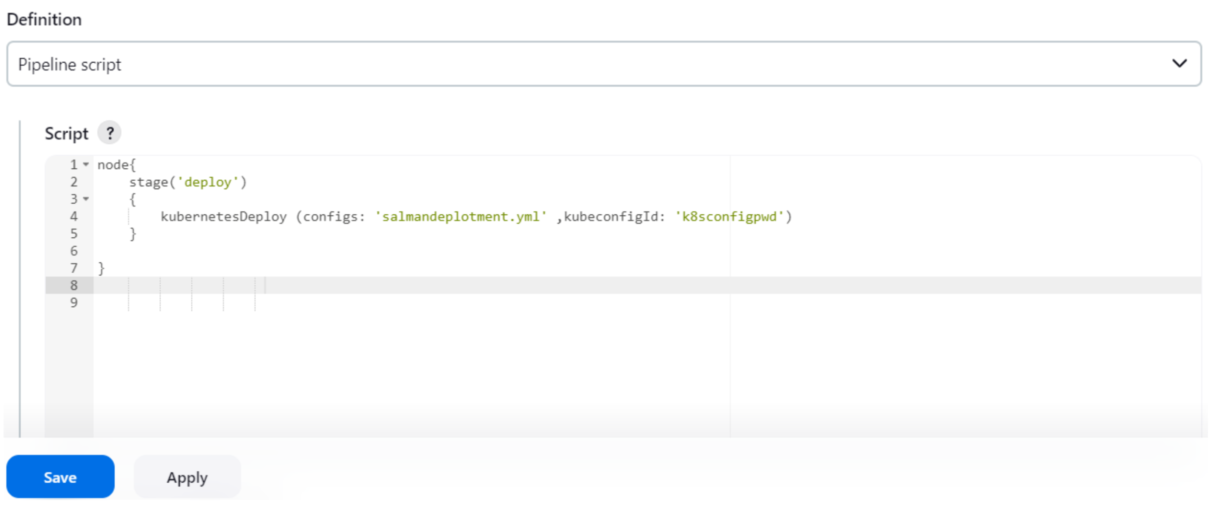


Script

```
pipeline{
    agent any

    stages{
        stage('git')
        {
            steps{
                git 'https://github.com/Salmansadiq809545/star-agile-insurance-project'
            }
        }
        stage('Terafrom init')
        {
            steps{
                sh 'terraform init'
            }
        }
        stage('Terafrom plan')
        {
            steps{
                sh 'terraform plan'
            }
        }
        stage('Terafrom Apply')
        {
            steps{
                sh 'terraform apply --auto-approve'
            }
        }
    }
}
```

Here is the Terraform pipeline written in jenkins

Kubernetes Jenkins Pipeline



Definition

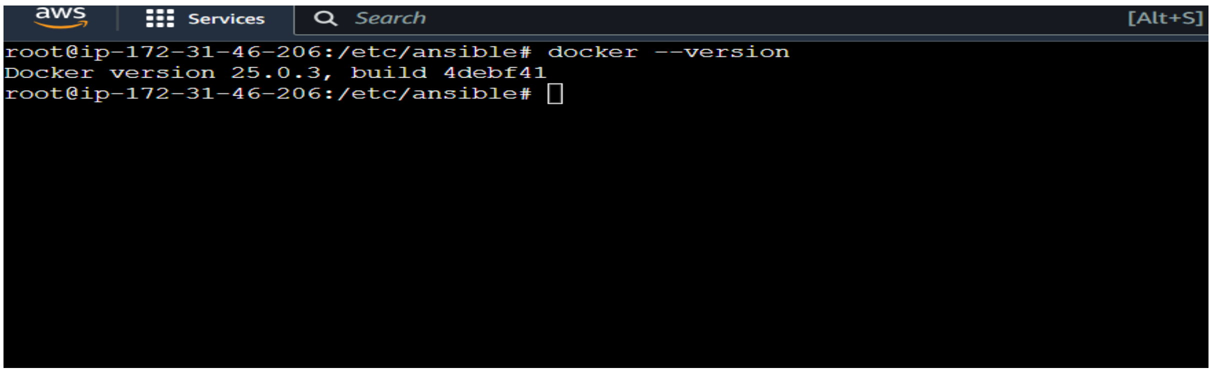
Pipeline script

Script

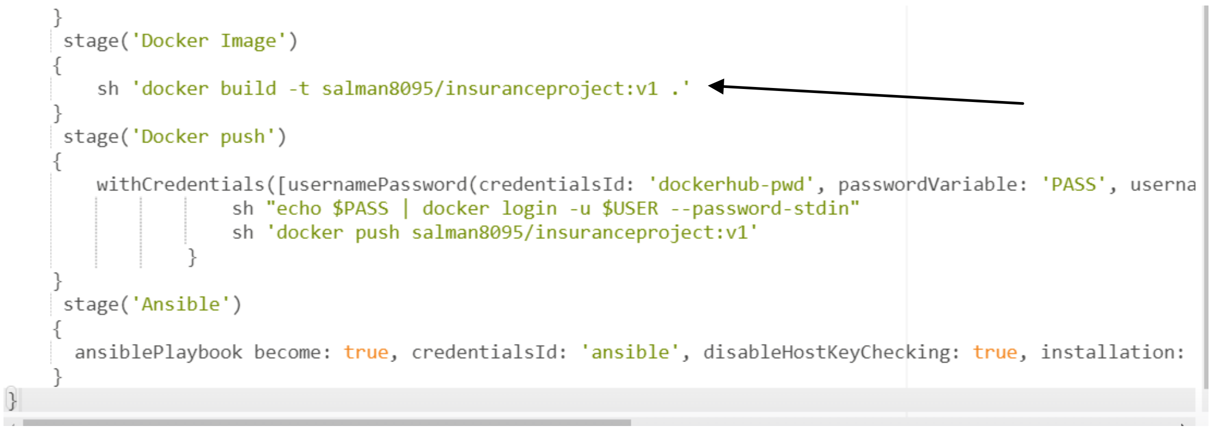
```
node{
    stage('deploy')
    {
        kubernetesDeploy (configs: 'salmandeplotment.yml' ,kubeconfigId: 'k8sconfigpwd')
    }

}
```

Here is a simple pipeline for deploying kubernetes cluster on node port

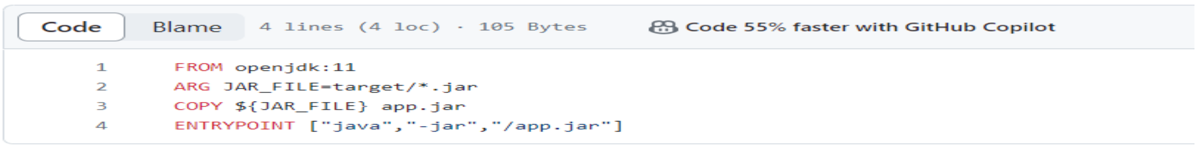
4)Docker Installtion and Configuration



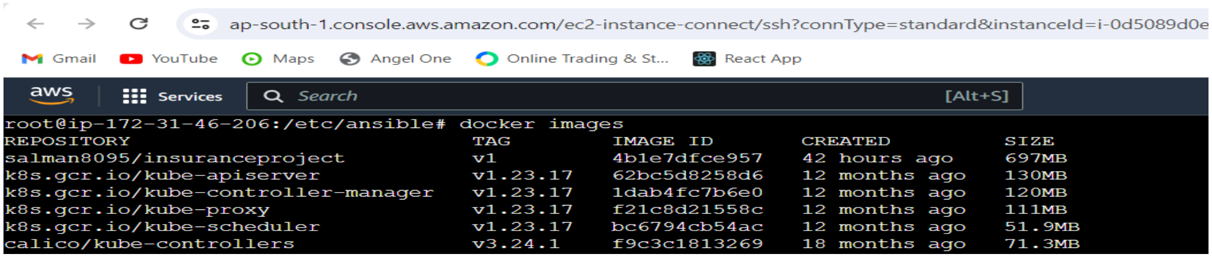
```
root@ip-172-31-46-206:/etc/ansible# docker --version
Docker version 25.0.3, build 4debf41
root@ip-172-31-46-206:/etc/ansible# 
```



```
    }
    stage('Docker Image')
    {
        sh 'docker build -t salman8095/insuranceproject:v1 .'
    }
    stage('Docker push')
    {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-pwd', passwordVariable: 'PASS', userna
                    sh "echo $PASS | docker login -u $USER --password-stdin"
                    sh 'docker push salman8095/insuranceproject:v1'
                }
    }
    stage('Ansible')
    {
      ansiblePlaybook become: true, credentialsId: 'ansible', disableHostKeyChecking: true, installation:
    }
}
```


Code
Blame
4 lines (4 loc) · 105 Bytes
Code 55% faster with GitHub Copilot

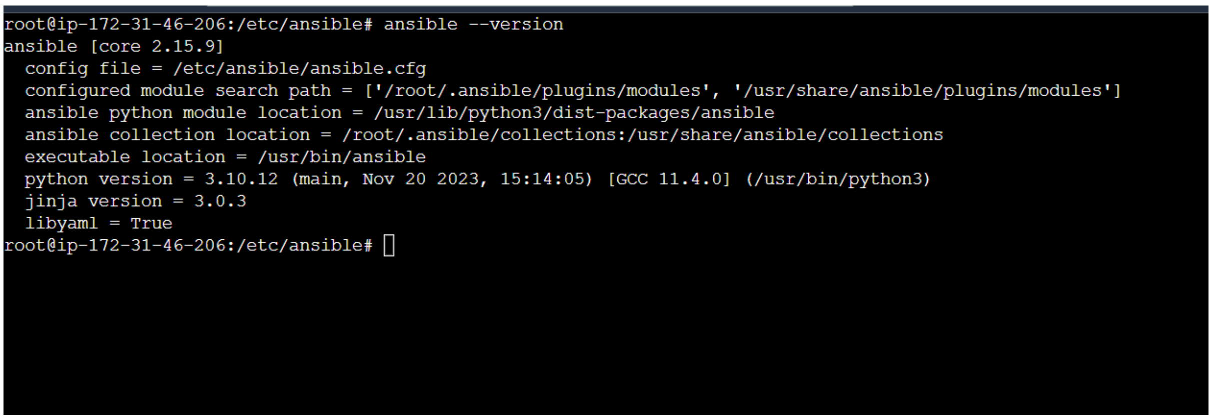
```
FROM openjdk:11
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```


ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d5089d0e
Gmail
YouTube
Maps
Angel One
Online Trading & St...
React App
aws
Services
Search
[Alt+S]

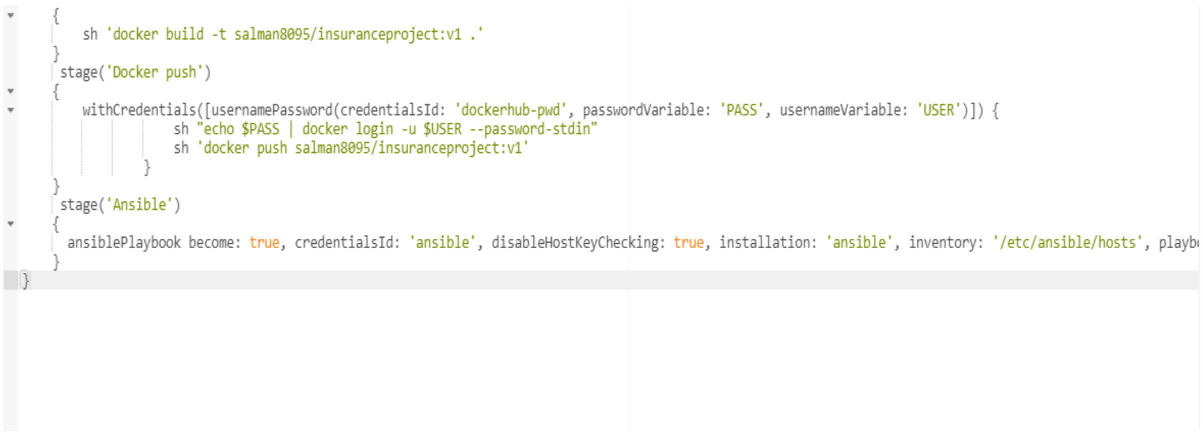
```
root@ip-172-31-46-206:/etc/ansible# docker images
REPOSITORY                           TAG        IMAGE ID       CREATED          SIZE
salman8095/insuranceproject          v1         4b1e7dfce957   42 hours ago     697MB
k8s.gcr.io/kube-apiserver            v1.23.17   62bc5d8258d6   12 months ago    130MB
k8s.gcr.io/kube-controller-manager   v1.23.17   1dab4fc7b6e0   12 months ago    120MB
k8s.gcr.io/kube-proxy                v1.23.17   f21c8d21558c   12 months ago    111MB
k8s.gcr.io/kube-scheduler            v1.23.17   bc6794cb54ac   12 months ago    51.9MB
calico/kube-controllers              v3.24.1    f9c3c1813269   18 months ago    71.3MB
```

Here is the pipeline code to build docker images and copy of docker file and after building docker image we push the image to docker hub

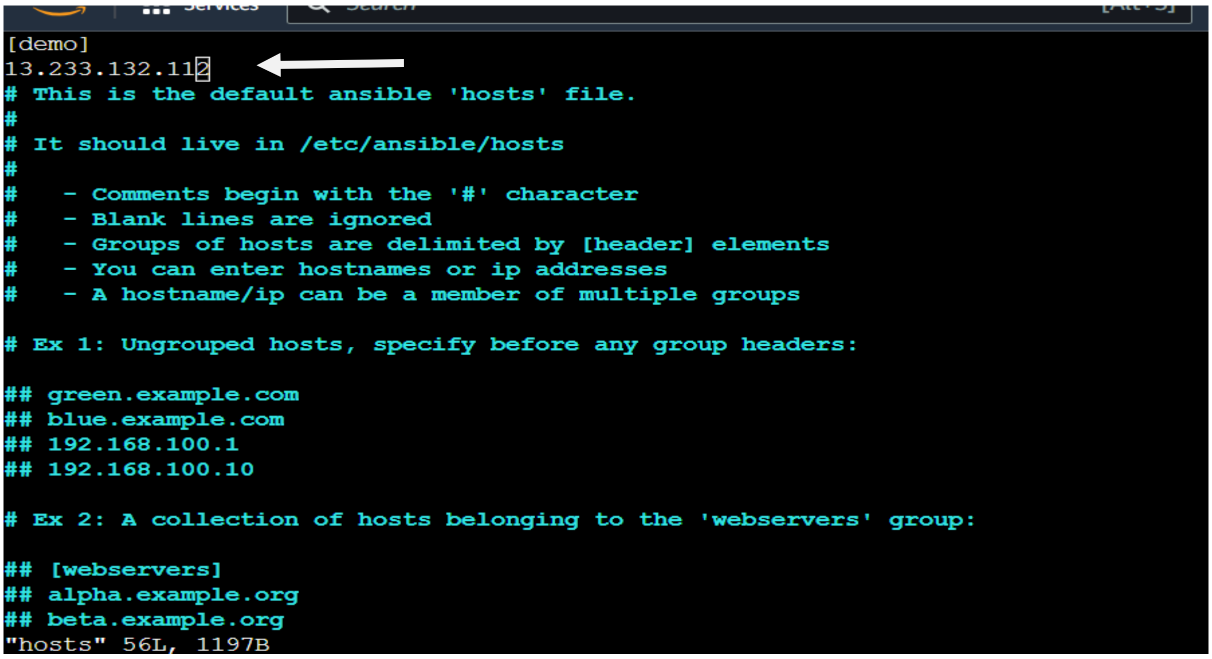
5) Ansible Installation and Configuration



```
root@ip-172-31-46-206:/etc/ansible# ansible --version
ansible [core 2.15.9]
  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-172-31-46-206:/etc/ansible# 
```



```
    {
        sh 'docker build -t salman8095/insuranceproject:v1 .'
    }
    stage('Docker push')
    {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-pwd', passwordVariable: 'PASS', usernameVariable: 'USER')]) {
                sh "echo $PASS | docker login -u $USER --password-stdin"
                sh 'docker push salman8095/insuranceproject:v1'
            }
    }
    stage('Ansible')
    {
      ansiblePlaybook become: true, credentialsId: 'ansible', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/hosts', playb
    }
}
```

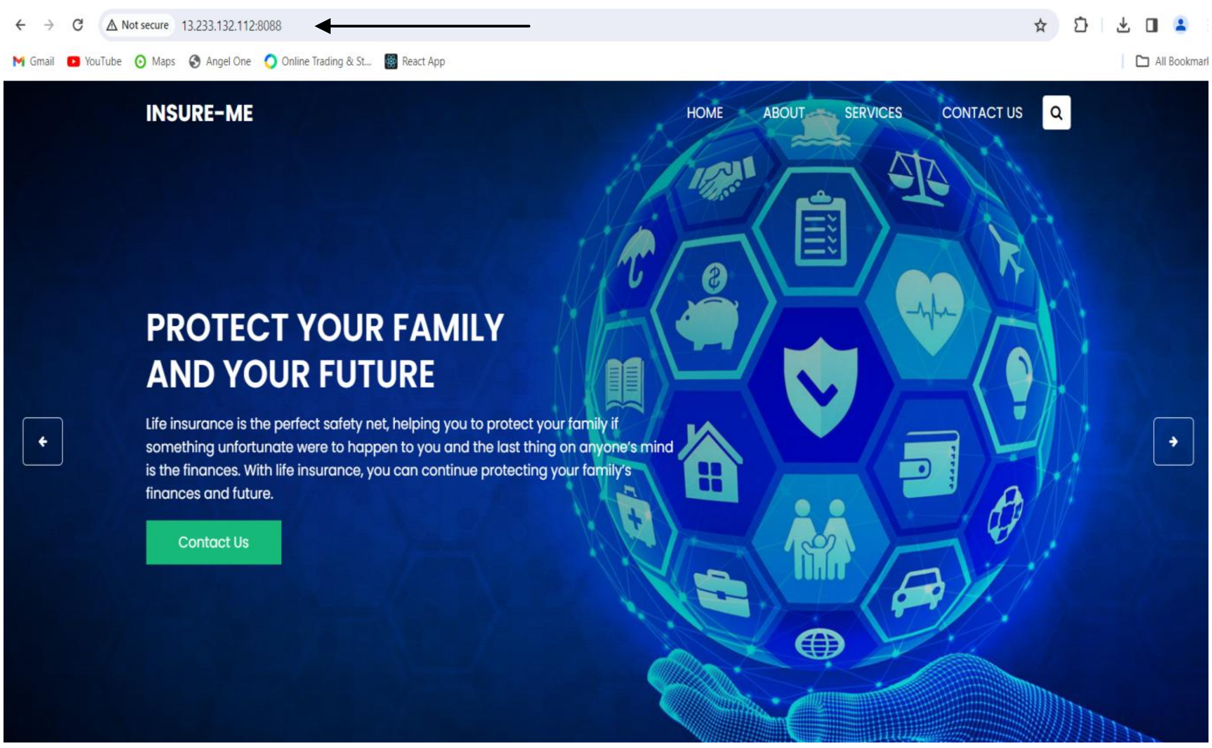


```
[demo]
13.233.132.112
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip 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]
## alpha.example.org
## beta.example.org
"hosts" 56L, 1197B
```

Here we can see the we have specified host where container will be deployed in ansible host that is demo group

Ansible-Playbook.yml

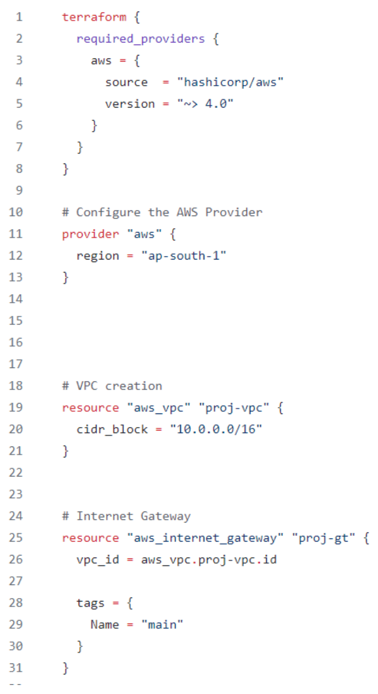
```
---
- name : Installing Container in Hosts
  hosts : all
  become: true
  connection : ssh
  tasks :
  - name: updating apt
    command : sudo apt-get update

  - name : Install Docker
    command : sudo apt install docker.io -y
    become : yes
    become_user : root

  - name : Start Docker Service
    command : sudo systemctl start docker
    become : yes
    become_user : root



  - name: Deploy Docker Container
    command: docker run -itd -p 8088:8081 --name C01 salman8095/insuranceproject:v1
```



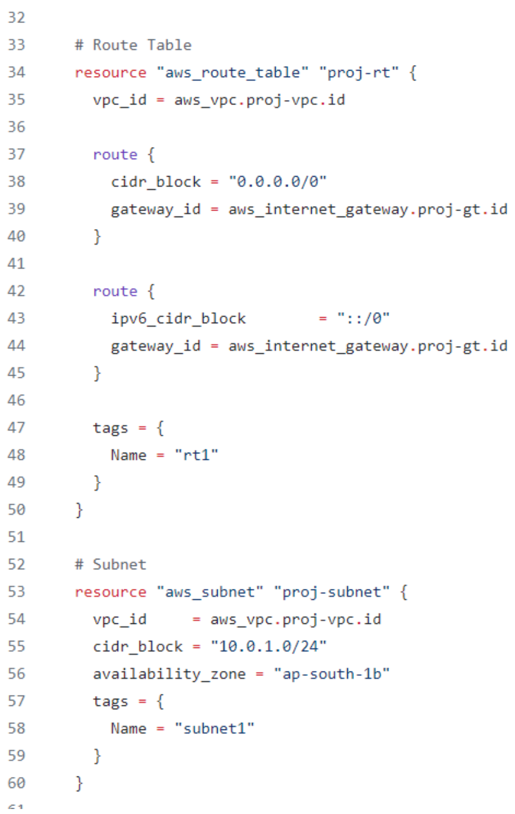
Life insurance is the perfect safety net, helping you to protect your family if something unfortunate were to happen to you and the last thing on anyone's mind is the finances. With life insurance, you can continue protecting your family's finances and future.

Here we can see the Ansible-playbook.yml file for deploying container and it has been successfully been hosted in port 8088

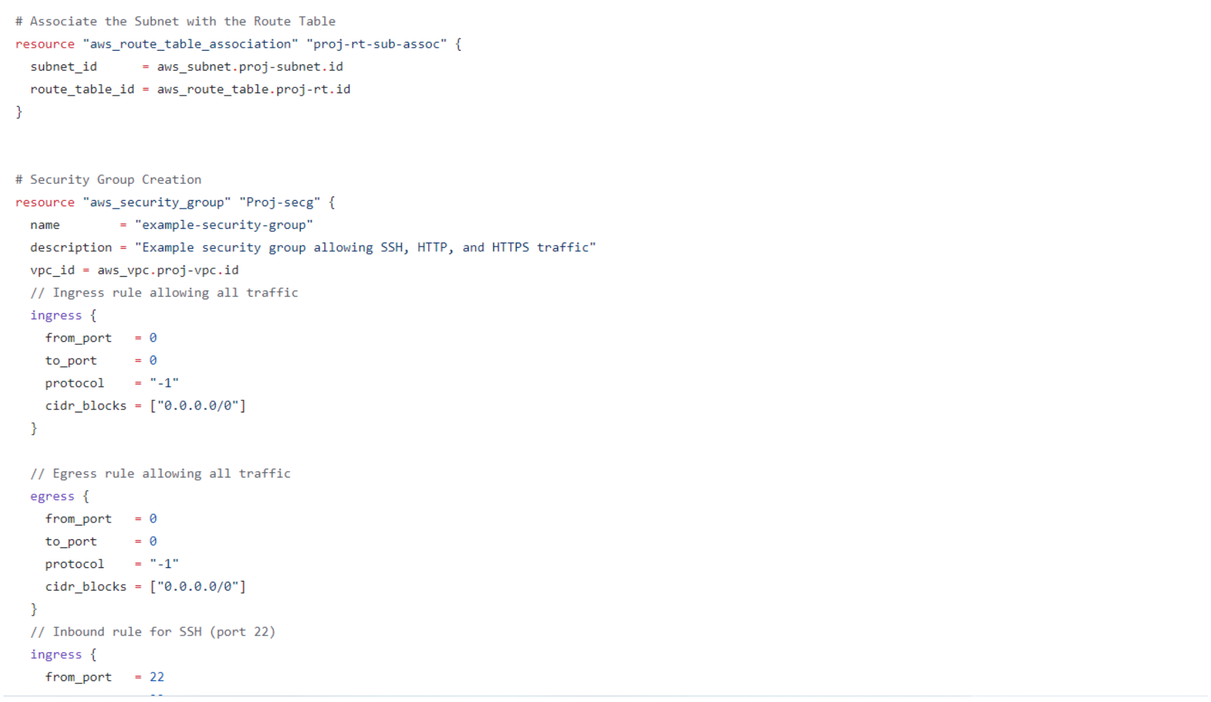
6) Terraform.tf file and its infrastructure creation and configuration



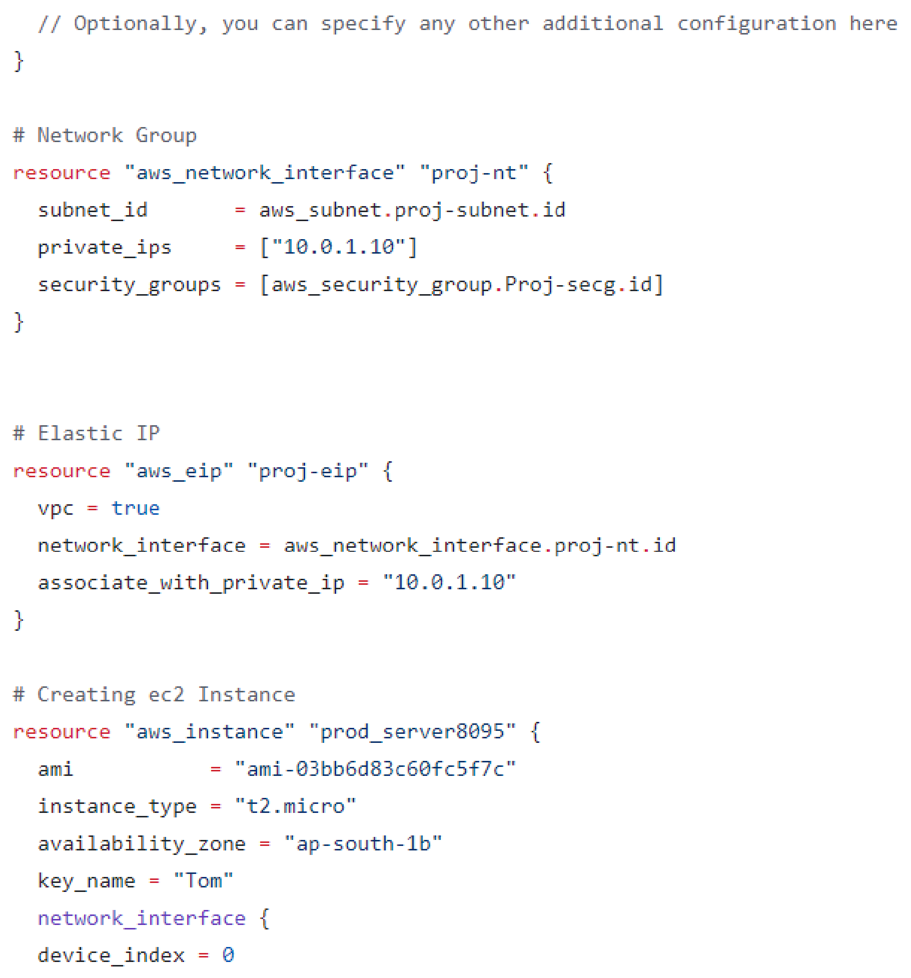
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "ap-south-1"
}




# VPC creation
resource "aws_vpc" "proj-vpc" {
  cidr_block = "10.0.0.0/16"
}


# Internet Gateway
resource "aws_internet_gateway" "proj-gt" {
  vpc_id = aws_vpc.proj-vpc.id

  tags = {
    Name = "main"
  }
}
```



```

# Route Table
resource "aws_route_table" "proj-rt" {
  vpc_id = aws_vpc.proj-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.proj-gt.id
  }

  route {
    ipv6_cidr_block        = "::/0"
    gateway_id = aws_internet_gateway.proj-gt.id
  }

  tags = {
    Name = "rt1"
  }
}

# Subnet
resource "aws_subnet" "proj-subnet" {
  vpc_id     = aws_vpc.proj-vpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "ap-south-1b"
  tags = {
    Name = "subnet1"
  }
}
```



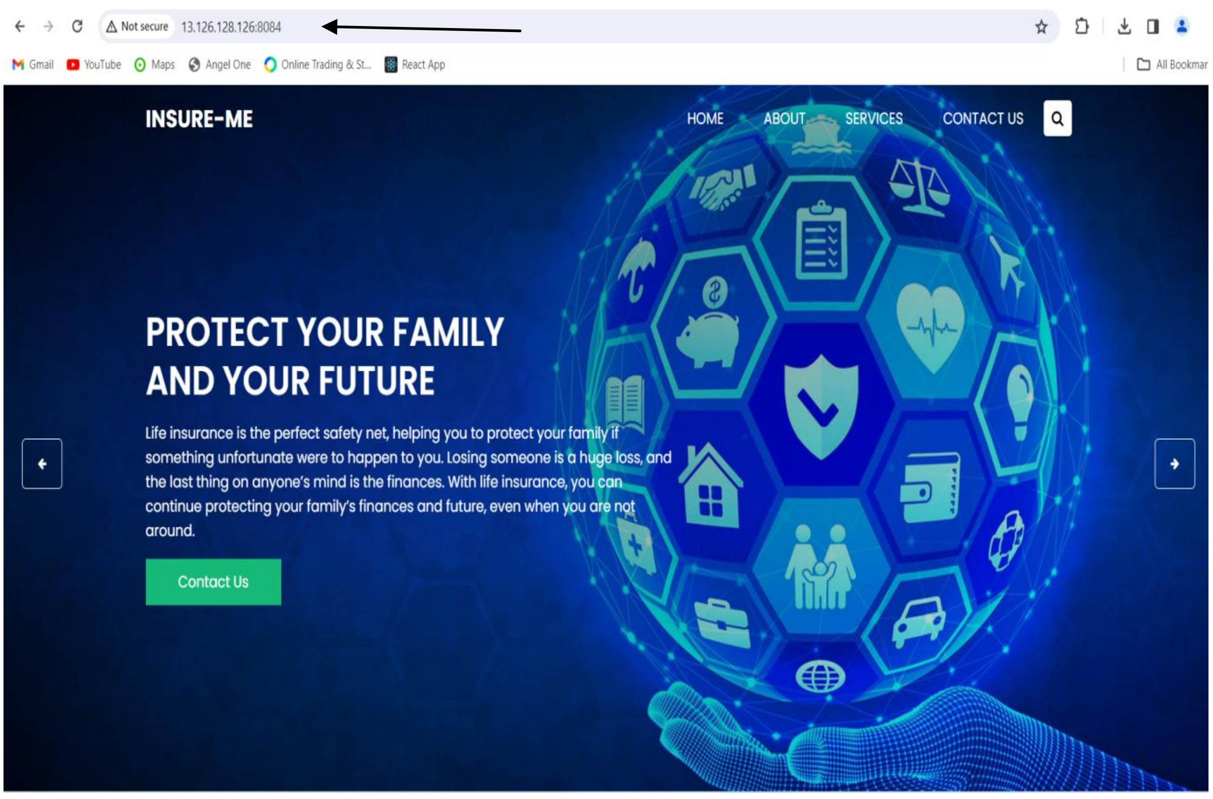
```
# Associate the Subnet with the Route Table
resource "aws_route_table_association" "proj-rt-sub-assoc" {
  subnet_id      = aws_subnet.proj-subnet.id
  route_table_id = aws_route_table.proj-rt.id
}


# Security Group Creation
resource "aws_security_group" "Proj-secg" {
  name        = "example-security-group"
  description = "Example security group allowing SSH, HTTP, and HTTPS traffic"
  vpc_id = aws_vpc.proj-vpc.id
  // Ingress rule allowing all traffic
  ingress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  // Egress rule allowing all traffic
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  // Inbound rule for SSH (port 22)
  ingress {
    from_port   = 22
```



```
    // Optionally, you can specify any other additional configuration here
  }

  # Network Group
  resource "aws_network_interface" "proj-nt" {
    subnet_id       = aws_subnet.proj-subnet.id
    private_ips     = ["10.0.1.10"]
    security_groups = [aws_security_group.Proj-secg.id]
  }



  # Elastic IP
  resource "aws_eip" "proj-eip" {
    vpc = true
    network_interface = aws_network_interface.proj-nt.id
    associate_with_private_ip = "10.0.1.10"
  }

  # Creating ec2 Instance
  resource "aws_instance" "prod_server8095" {
    ami           = "ami-03bb6d83c60fc5f7c"
    instance_type = "t2.micro"
    availability_zone = "ap-south-1b"
    key_name = "Tom"
    network_interface {
    device_index = 0
```

Here we can see how terraform infrastructure is created like vpc, and network insance

```
# Creating ec2 Instance
resource "aws_instance" "prod_server8095" {
  ami           = "ami-03bb6d83c60fc5f7c"
  instance_type = "t2.micro"
  availability_zone = "ap-south-1b"
  key_name = "Tom"
  network_interface {
  device_index = 0
  network_interface_id = aws_network_interface.proj-nt.id
}
user_data = <<-EOF
#!/bin/bash
    sudo apt-get update -y
    sudo apt-get update -y
    sudo apt-get install docker.io -y
    sudo systemctl enable docker
    sudo docker run -itd -p 8084:8081 --name C01 salman8095/insuranceproject:v1
    sudo docker start $(docker ps -aq)
  EOF
  tags = {
    Name = "Salman-Prod-Server"
  }
}
```



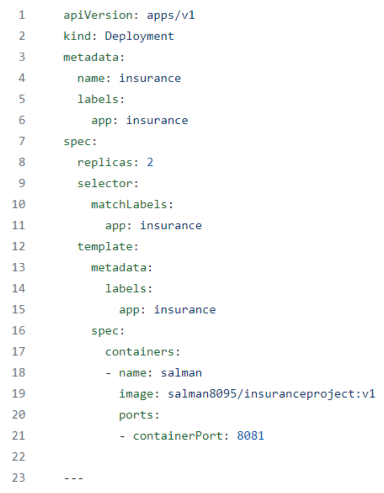
PROTECT YOUR FAMILY
AND YOUR FUTURE

Life insurance is the perfect safety net, helping you to protect your family if something unfortunate were to happen to you. Losing someone is a huge loss, and the last thing on anyone's mind is the finances. With life insurance, you can continue protecting your family's finances and future, even when you are not around.

Here we can see the we have containerized and deployed the application using terraform in port 8084

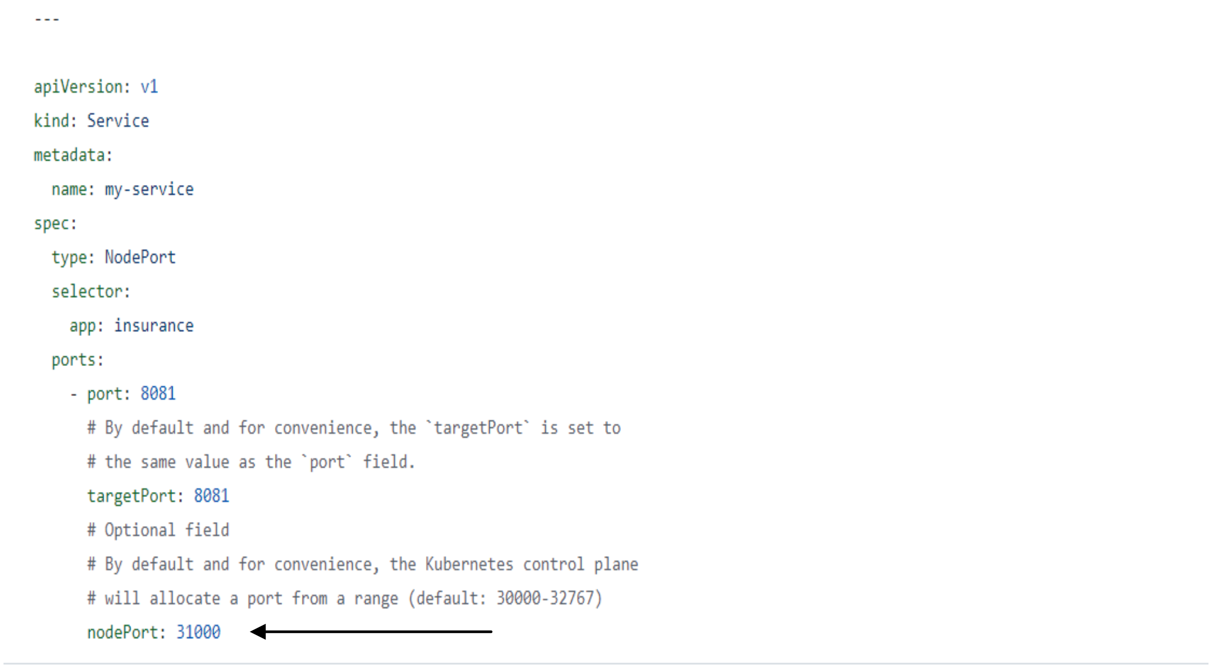
7) Kubernetes Deployment and configuration

Deployment.yml file



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: insurance
  labels:
    app: insurance
spec:
  replicas: 2
  selector:
    matchLabels:
      app: insurance
  template:
    metadata:
      labels:
        app: insurance
    spec:
      containers:
      - name: salman
        image: salman8095/insuranceproject:v1
        ports:
        - containerPort: 8081

---
```

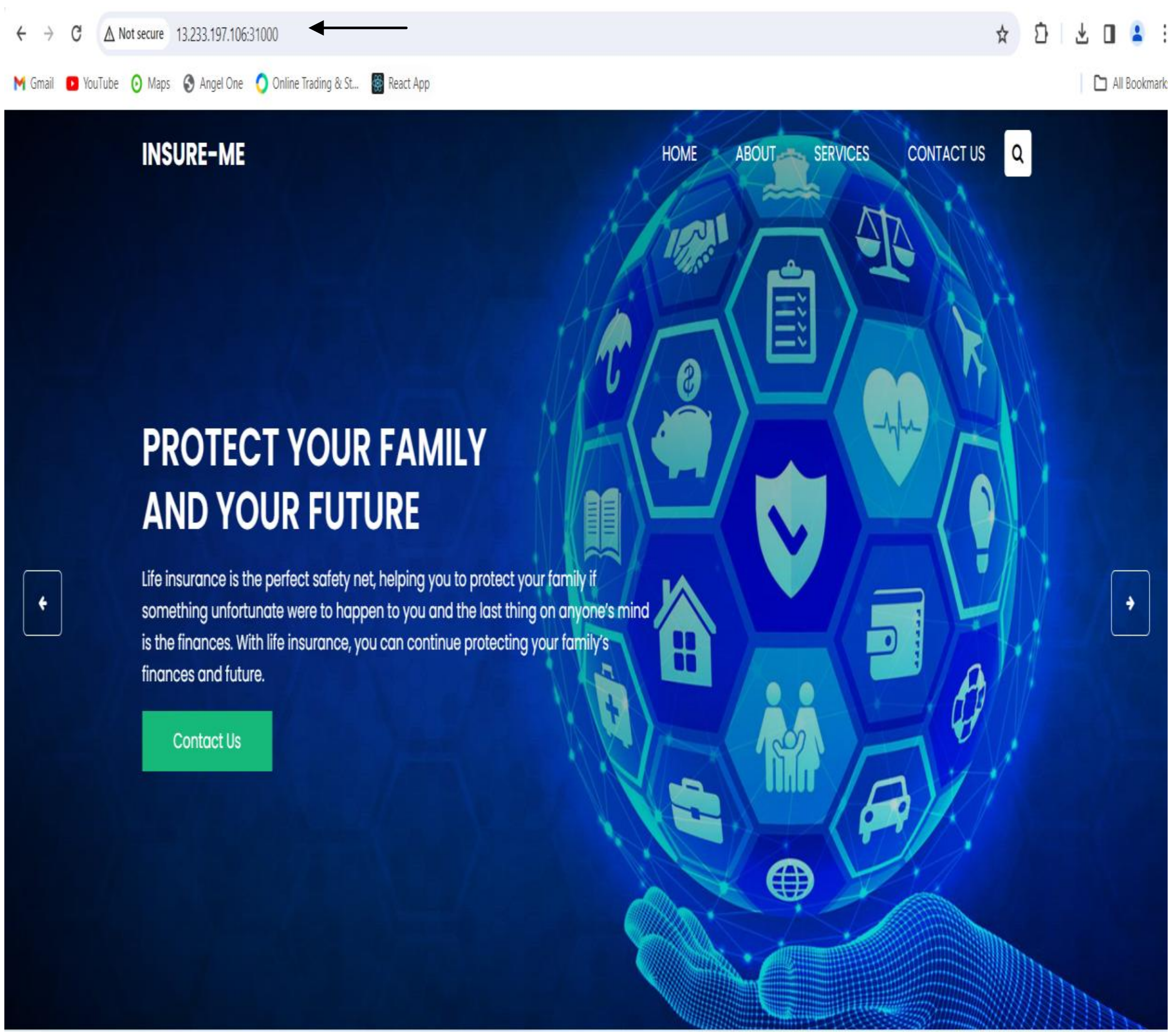
Service.yml file



```yaml
---

apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: insurance
  ports:
    - port: 8081
      # By default and for convenience, the `targetPort` is set to
      # the same value as the `port` field.
      targetPort: 8081
      # Optional field
      # By default and for convenience, the Kubernetes control plane
      # will allocate a port from a range (default: 30000-32767)
      nodePort: 31000
```

Here we have created a deployment and service of node port for the containerized application which is deployed at port 31000

Here we can see we have deployed the containerized application using kubernetes



PROTECT YOUR FAMILY AND YOUR FUTURE

Life insurance is the perfect safety net, helping you to protect your family if something unfortunate were to happen to you and the last thing on anyone's mind is the finances. With life insurance, you can continue protecting your family's finances and future.

THANK YOU

END OF

INSURANCE

PROJECT