

CAPSTONE PROJECT

Insurance Domain- Insure Me

Darsana Praveen M

**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

The screenshot shows the AWS CloudFormation console interface. In the 'Name and tags' section, the name 'master' is entered. Under 'Application and OS Images (Amazon Machine Image)', the 'Quick Start' tab is selected, showing various AMI icons including Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. The 'Summary' panel on the right shows 'Number of instances' set to 1, with the 'Software Image (AMI)' field populated with 'Canonical, Ubuntu, 22.04 LTS'. The 'Virtual server type (instance type)' is set to 't2.micro'. A tooltip indicates a 'Free tier' benefit. The 'Launch instance' button is highlighted in orange.

This screenshot shows the AWS CloudFormation console for creating a second instance. The 'Name and tags' section has 'node' as the name. The 'Application and OS Images (Amazon Machine Image)' section shows the 'Quick Start' tab selected, with the same set of AMI icons as the previous screenshot. The 'Summary' panel on the right shows 'Number of instances' set to 1, with the 'Software Image (AMI)' field populated with 'Canonical, Ubuntu, 22.04 LTS'. The 'Virtual server type (instance type)' is set to 't2.micro'. A tooltip indicates a 'Free tier' benefit. The 'Launch instance' button is highlighted in orange.

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.

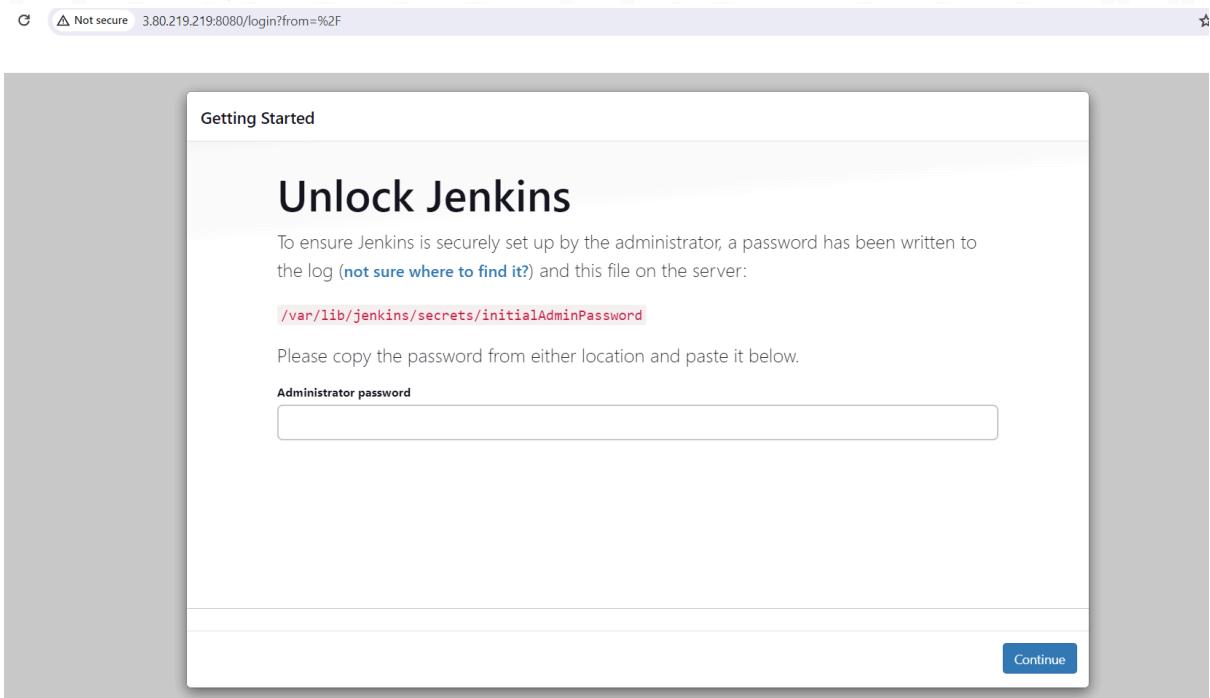
```
no VR guests are running outdated hypervisor (qemu) binaries  
root@ip-172-31-28-64:/home/ubuntu# docker --version  
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1  
root@ip-172-31-28-64:/home/ubuntu#
```

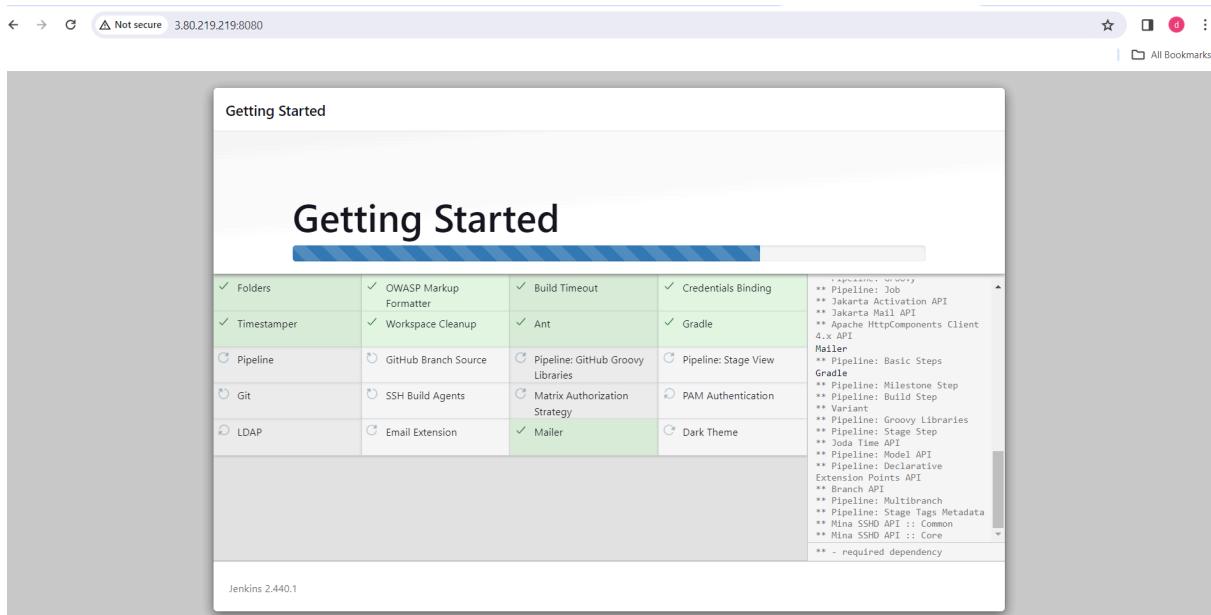
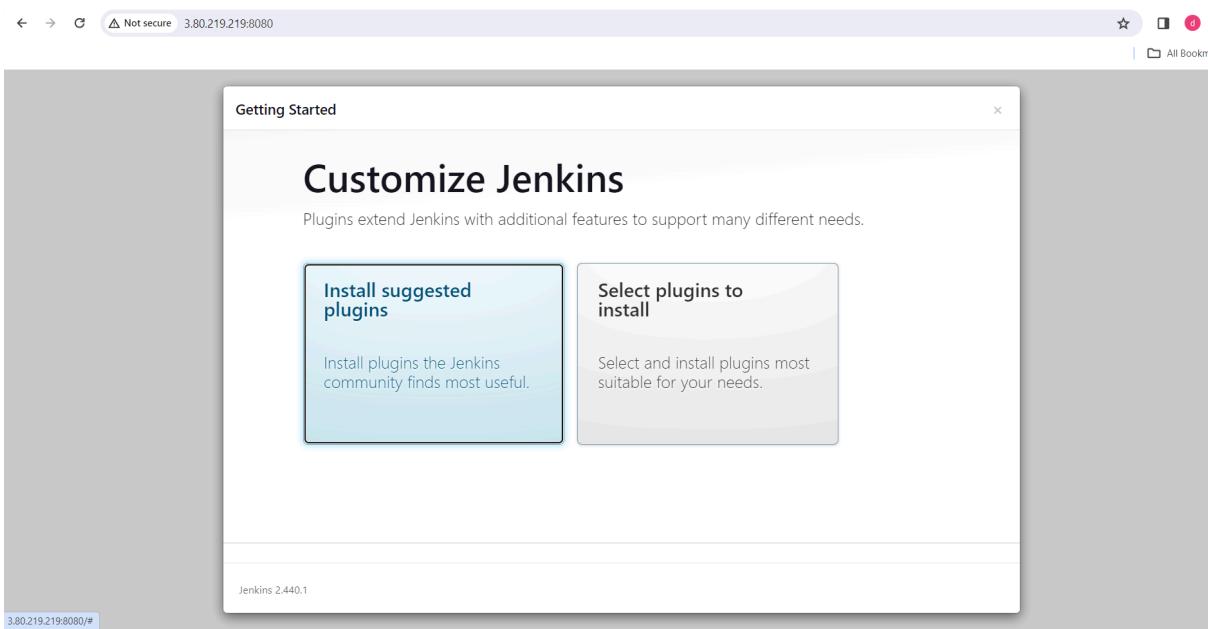
i-0e6616d4889b1de8f (master)

Public IPs: 3.80.219.219 Private IPs: 172.31.28.64

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





The screenshot shows the Jenkins dashboard at the URL 3.80.219.219:8080. The top navigation bar includes a 'Not secure' warning, the IP address, and user information for 'admin'. The main content area features a 'Welcome to Jenkins!' message and a 'Start building your software project' section. On the left, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (2 Idle). The right side has a 'Create a job' button and a 'Set up a distributed build' section with links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

Darsana Praveen M

Installation of Plugins in Jenkins:

The screenshot shows the 'Manage Jenkins > Plugins' page. The left sidebar includes 'Updates', 'Available plugins' (selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area displays a search bar for 'HTML publisher' and a list of available plugins. The 'Docker' plugin is selected for installation. Other listed plugins include 'Ansible' and 'HTML Publisher'. Each plugin entry shows its name, version, release date, and a brief description.

Plugin	Version	Released
Docker	1.6	20 days ago
Ansible	307.va.1f3ef06575a...	2 mo 3 days ago
HTML Publisher	1.32	7 mo 2 days ago

The screenshot shows the Jenkins 'Plugins' page under 'Manage Jenkins'. The sidebar includes links for 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress' (which is selected). A table lists installed plugins with green checkmarks and the word 'Success' next to each. The table includes entries like LDAP, Email Extension, Mailer, Theme Manager, Dark Theme, Loading plugin extensions, Cloud Statistics, Authentication Tokens API, Docker Commons, Apache HttpComponents Client 5.x API, Docker API, Docker, Ansible, and HTML Publisher. Below the table are two links: 'Go back to the top page' and 'Restart Jenkins when installation is complete and no jobs are running'.

REST API Jenkins 2.44.0.1

Deployment to Node-server using ANSIBLE

Create a pipeline

The screenshot shows the Jenkins 'Create a new item' dialog. The title is 'Enter an item name' and the input field contains 'ansible-deployment'. Below the input field is a note '» Required field'. There are four options listed: 'Freestyle project' (selected), 'Pipeline', 'Multi-configuration project', and 'Folder'. Each option has a description and a blue 'OK' button. The 'Freestyle project' description says it's a classic general-purpose job type. The 'Pipeline' description says it's for orchestrating long-running activities. The 'Multi-configuration project' description says it's for projects with many configurations. The 'Folder' description says it's for grouping items together.

Code checkout from github

The screenshot shows the Jenkins Pipeline configuration page for a project named "ansible-deployment". The "Pipeline" tab is selected. The pipeline script is defined as follows:

```
1 pipeline{  
2     agent any  
3     stages{  
4         stage('checkout code'){  
5             steps{  
6                 git url: 'https://github.com/Darsana-123/star-agile-insurance-project.git', branch:'master'  
7             }  
8         }  
9     }  
10 }
```

Below the script, there is a checked checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons.

Code compile, testing, and package creation

The screenshot shows the Jenkins Pipeline configuration page for the same project. The "Pipeline" tab is selected. The pipeline script is defined as follows:

```
10 stage('code compile'){  
11     steps{  
12         echo 'starting compiling'  
13         sh 'mvn compile'  
14     }  
15 }  
16 stage('codetesting'){  
17     steps{  
18         sh 'mvn test'  
19     }  
20 }  
21 stage('package'){  
22     steps{  
23         sh 'mvn clean package'  
24     }  
25 }  
26 }
```

Below the script, there is a checked checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons.

ansible-deployment

Stage View

	checkout code	code compile	codetesting	package
#6 Mar 04 01:32 No Changes	1s	5s	26s	29s
#5 Mar 04 01:32 No Changes	2s	8s	25s	28s
#4 Mar 04 01:30 No Changes	378ms	14s	26s	85ms

Creating HTML Report:

Index page title[s] (Optional)

Report title ?

HTML Report

Publishing options

Generate Pipeline Script

```
publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/ansible-deployment/target/surefire-reports', reportFiles: 'index.html', reportName: 'HTML Report', reportTitles: '', useWrapperFileDirectly: true])
```

Global Variables

```

25
26
27
28 +
29 +     stage('HTML Report'){
30 +         steps{
31 +             publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: '/var/lib/jenkins/workspace/a
32 +         }
33     }
34 }
35

```

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

HTML report is created in the below location

The screenshot shows the Jenkins Stage Log for the 'HTML Report' stage. It displays the command [htmlpublisher] Archiving HTML reports... and the path [var/lib/jenkins/jobs/ansible-deployment/target/surefire-reports to /var/lib/jenkins/jobs/ansible-deployment/htmlreports/HTML_20Report]. Below the log, there's a sidebar with options like Changes, Build Now, Configure, Delete Pipeline, Full Stage View, and HTML Report. The Stage View section shows five stages: checkout code, code compile, codetesting, package, and HTML Report, with their respective average run times: 916ms, 5s, 21s, 23s, and 102ms.

Building Docker image:

Giving sudo permissions to Jenkins

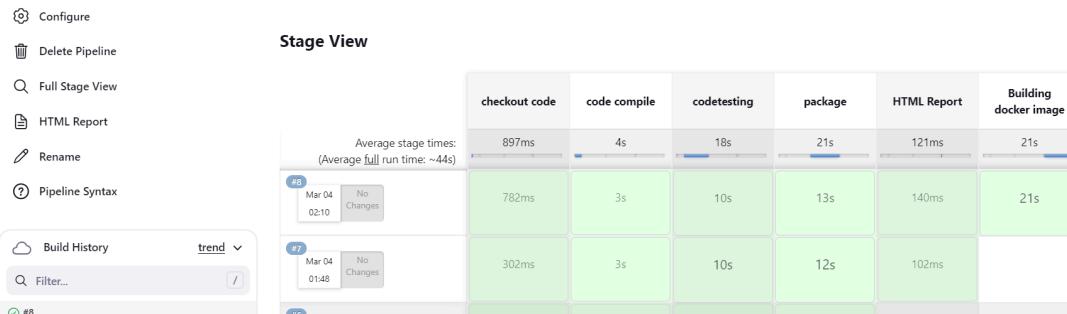
```

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)

Public IPs: 3.80.219.219 Private IPs: 172.31.28.64



DockerHub Login & Pushing the image to Dockerhub

Set the docker hub credentials to Jenkins for accessing docker hub

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator. The left sidebar lists various documentation resources: Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GSLS. The main content area is titled 'OVERVIEW' and contains a snippet generator for Pipeline Script. A specific step, 'withCredentials: Bind credentials to variables', is highlighted with a blue box. Below it, the 'withCredentials' section is expanded, showing a 'Secret text' input field containing 'dockerpass' and a 'Credentials' dropdown menu.

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 ?
```

```

37      }
38    }
39  }
40  stage('Dockerhub login and image pushing'){
41    steps{
42      withCredentials([string(credentialsId: 'dockerpass', variable: 'dockerpass')) {
43        sh " docker login -u darsanapraveen -p ${dockerpass}"
44        sh 'docker push darsanapraveen/insuranceproject:1.0 '
45      }
46    }
47  }
48 }
49 }
50
51
52 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Dashboard > ansible-deployment >

Status ✓ **ansible-deployment**

Changes Build Now Add description

Configure Disable Project

Delete Pipeline

Full Stage View HTML Report

Rename Pipeline Syntax

Build History trend Filter...

Stage View

	checkout code	code compile	codetesting	package	HTML Report	Building docker image	Dockerhub login and image pushing
Average stage times: (Average full run time: ~43s)	831ms	4s	16s	19s	117ms	12s	6s
#9 Mar 04 02:32 No Changes	376ms	3s	10s	12s	111ms	3s	6s
#8 Mar 04 No	782ms	3s	10s	13s	140ms	21s	

hub.docker.com

dockerhub Explore Repositories Organizations Search Docker Hub Create repository

darsanapraveen Search by repository name All Content

darsanapraveen / insuranceproject Contains: Image | Last pushed: 1 minute ago Security unknown 0 Public

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)

Public IPs: 3.80.219.219 Private IPs: 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)

Public IPs: 3.80.219.219 Private IPs: 172.31.28.64

Setting up the jenkins tool

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Tools' section. A new 'Ansible' tool is being configured. The 'Name' field contains 'ansible'. The 'Install automatically' checkbox is checked. There is a 'Save' button at the bottom.

Creating an Ansible playbook file in Github repository

```

1 - name : Configure Docker on EC2 Instances
2 hosts : all
3 become: true
4 connection : ssh
5 tasks :
6 - name: updating apt
7   command : sudo apt-get update
8
9 - name : Install Docker
10  command : sudo apt-get install -y docker.io
11  become : yes
12  become_user : root
13
14 - name : Start Docker Service
15  command : sudo systemctl start docker
16  become : yes
17  become_user : root
18
19 - name: Deploy Docker Container
20  command: docker run -itd -p 8084:8081 darsanapraveen/insuranceproject:1.0
21

```

The screenshot shows a GitHub repository named 'star-agile-insurance-project'. An Ansible playbook file, 'ansible-playbook.yml', is open in the code editor. The file contains the following YAML code:

 Darsana-123 / star-agile-insurance

<> Code Pull requests Actions

 Files

master   

 Go to file 

- >  .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

Jenkins Credentials Provider: Jenkins

Username: ubuntu

Treat username as secret ?

Private Key:

- Enter directly

Key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEaIsVxr4T/oeiSKSn8QS/F9Z7Ip1MTD9tv1DzShL5/iK25VRKT
nXw2c1XAEm3Jkv5Dv73+Vs06vd0oJ1K71410K9Ly+2tCLMw0wCKF1DeiSAwOnz9b
-----END RSA PRIVATE KEY-----
```

Passphrase:

Dashboard > ansible-deployment > Pipeline Syntax

Sample Step: ansiblePlaybook: Invoke an ansible playbook

Ansible tool: ansible

Playbook file path in workspace: ansible-playbook.yml

Inventory file path in workspace: /etc/ansible/hosts

SSH connection credentials: ubuntu

Vault credentials: - none -

Dashboard > ansible-deployment > Pipeline Syntax

Task to start at

Number of parallel processes to use

Disable the host SSH key check

Colorized output

Extra parameters

Generate Pipeline Script

```
ansiblePlaybook.credentialsId: 'privatekey', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/hosts', playbook: 'ansible-playbook.yml', vaultTmpPath: ''
```

Darsana Praveen M

Ansible Deployment

Configure Disable Project

Delete Pipeline

Full Stage View HTML Report Rename Pipeline Syntax

Build History trend Filter... #10

	checkout code	code compile	codetesting	package	HTML Report	Building docker image	Dockerhub login and image pushing	ansible deployment
Average stage times: (Average full run time: ~49s)	781ms	4s	15s	18s	111ms	9s	5s	47s
#10 Mar 04 03:34 No Changes	377ms	3s	11s	13s	92ms	3s	5s	47s
#9 Mar 04 02:32 No Changes	376ms	3s	10s	12s	111ms	3s	6s	



```

Dashboard > ansible-deployment > #10

ok: [18.207.128.252]

TASK [updating apt] ****
changed: [18.207.128.252]

TASK [Install Docker] ****
changed: [18.207.128.252]

TASK [Start Docker Service] ****
changed: [18.207.128.252]

TASK [Deploy Docker Container] ****
changed: [18.207.128.252]

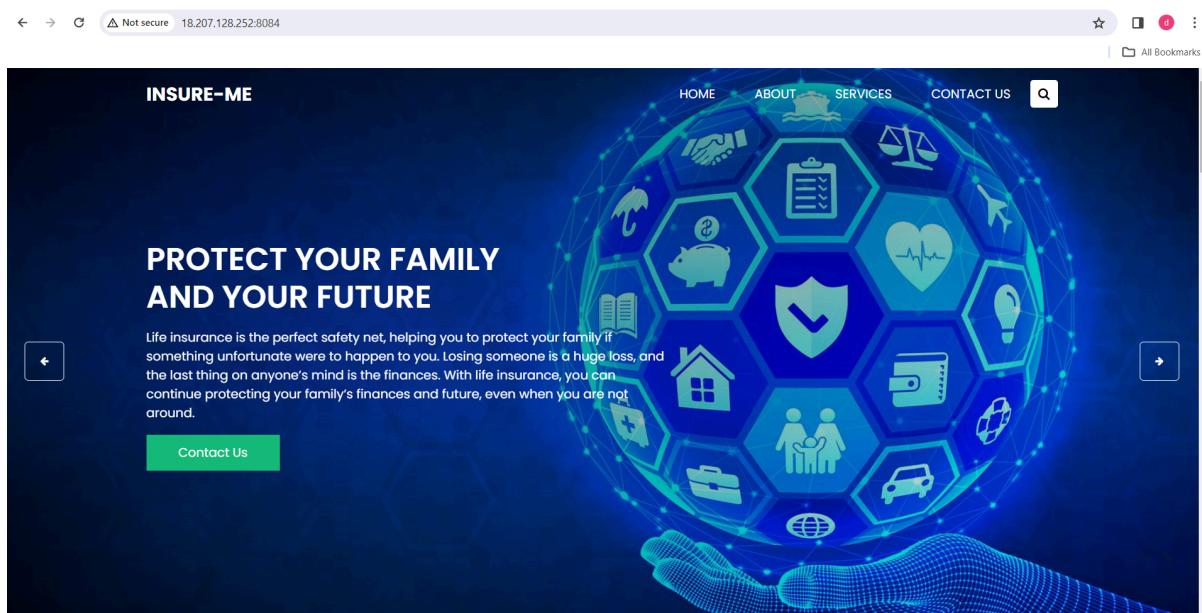
PLAY RECAP ****
18.207.128.252 : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

REST API Jenkins 2.440.1

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.

The screenshot shows the AWS EC2 Instances page. There are two instances listed: 'master' (Running, t2.medium) and 'node' (Running, t2.micro). The 'Actions' dropdown for the 'master' instance is open, with 'Security' highlighted. Other options include Connect, View details, Manage instance state, Instance settings, Networking, Image and templates, and Monitor and troubleshoot.

The screenshot shows the 'Modify IAM role' dialog box. The 'Instance ID' is set to 'i-0e6616d4889b1de8f (master)'. The 'IAM role' dropdown contains 'roletf'. There is a 'Create new IAM role' button. At the bottom are 'Cancel' and 'Update IAM role' buttons.

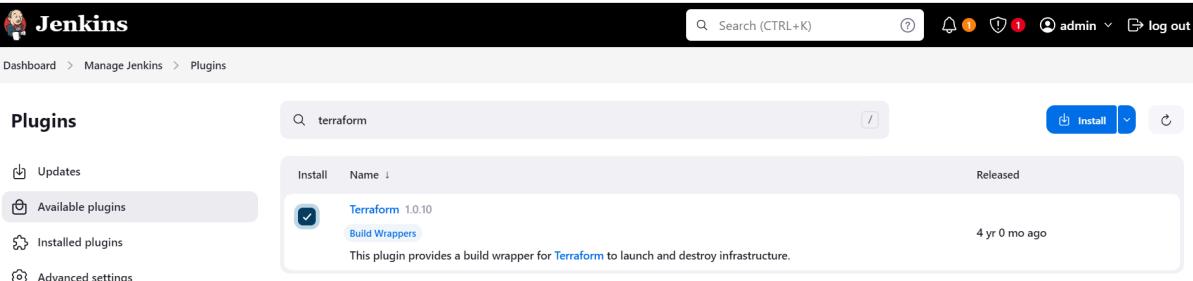
Terraform installation:

```
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)

Public IPs: 3.80.219.219 Private IPs: 172.31.28.64

Plugin installation on Jenkins:



The screenshot shows the Jenkins interface for managing plugins. The top navigation bar includes the Jenkins logo, a search bar with placeholder text 'Search (CTRL+K)', and user information for 'admin'. Below the header, the URL 'Dashboard > Manage Jenkins > Plugins' is visible. The main content area is titled 'Plugins' and features a sidebar with links for 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', and 'Advanced settings'. A search bar at the top of the main content area contains the text 'terraform'. Below it, a table lists the 'Terraform 1.0.10' plugin, which is checked for installation. The table columns include 'Install' (with a blue checkmark icon), 'Name' (Terraform 1.0.10), and 'Released' (4 yr 0 mo ago). A note below the table states: 'This plugin provides a build wrapper for Terraform to launch and destroy infrastructure.'

We have to create a terraform file to our Github:

```

111  vpc = true
112  network_interface = aws_network_interface.proj-ni.id
113  associate_with_private_ip = "10.0.1.10"
114  }
115
116
117  # Creating an ubuntu EC2 instance
118  resource "aws_instance" "Prod-Server" {
119    ami           = "ami-0ef82eba2c7a0eef"
120    instance_type = "t2.micro"
121    availability_zone = "us-east-1b"
122    key_name      = "darsana"
123    network_interface {
124      device_index = 0
125      network_interface_id = aws_network_interface.proj-ni.id
126    }
127    user_data = <<EOF
128    #!/bin/bash
129    sudo apt-get update -y
130    sudo apt install docker.io -y
131    sudo systemctl enable docker
132    sudo docker run -itd -p 8085:8081 darsanapraveen/insuranceproject:1.0
133    sudo docker start $(docker ps -aq)
134  EOF
135  tags = [
136    { Name = "Prod-Server" }
137  ]
138 }

```

Create a pipeline for terraform deployment:

Enter an item name

» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
A container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a namespace, so you can have multiple things of the same name as long as they are in different folders.

The screenshot shows a configuration interface for a Terraform deployment. The top navigation bar includes a back arrow, a 'Not secure' warning, the URL '3.80.219.219:8080/job/terraform-deployment/configure', and a star icon. The left sidebar has links for 'Dashboard', 'terraform-deployment', 'Configuration', 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' link is highlighted with a grey background. The main area is titled 'Configure' and contains a 'Pipeline script' dropdown set to 'Pipeline'. Below it is a code editor with Groovy syntax highlighting, displaying the following pipeline script:

```
1 * pipeline{
2     agent any
3     stages{
4         stage('checkout the code from github'){
5             steps{
6                 git url: 'https://github.com/Darsana-123/star-agile-insurance-project.git', branch:'master'
7                 echo 'github url checkout'
8             }
9         }
10        stage('Terraform Init'){
11            steps{
12                sh 'terraform init'
13            }
14        }
15    }
16}
17
```

Below the code editor is a checkbox labeled 'Use Groovy Sandbox' with a checked status. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins interface for the 'terraform-deployment' pipeline. The top navigation bar includes the Jenkins logo, a search bar with placeholder 'Search (CTRL+K)', and user account information ('admin' and 'log out'). Below the header, the pipeline name 'terraform-deployment' is displayed with a green checkmark icon. On the left, a sidebar lists various pipeline management options: Status (highlighted), Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main content area is titled 'Stage View' and displays four stages: 'checkout the code from github', 'Terraform Init', 'Terraform Plan', and 'Terraform apply'. Each stage has a progress bar indicating its duration: 945ms, 3s, 6s, and 11s respectively. A note at the bottom states 'Average stage times: (Average full run time: ~50s)'. To the right of the stages, there are buttons for 'Add description' and 'Disable Project'.

```
j
  id          = "sg-0d335433b2873c372"
  name        = "proj-sg"
  tags        = {
    "Name" = "proj-sg1"
  }
  # (90m# (7 unchanged attributes hidden)@[0m@[0m
}

@[0mPlan:@[0m 1 to add, 1 to change, 0 to destroy.
@[0m@[0m@1aws_security_group.proj-sg: Modifying... [id=sg-0d335433b2873c372]@[0m@[0m
@[0m@[1aws_security_group.proj-sg: Modifications complete after 0s [id=sg-0d335433b2873c372]@[0m
@[0m@[1aws_instance.Prod-Server: Creating...@[0m@[0m
@[0m@[1aws_instance.Prod-Server: Still creating... [10s elapsed]@[0m@[0m
@[0m@[1aws_instance.Prod-Server: Still creating... [20s elapsed]@[0m@[0m
@[0m@[1aws_instance.Prod-Server: Still creating... [30s elapsed]@[0m@[0m
@[0m@[1aws_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
```

Screenshot of the AWS EC2 Instances page showing three running instances: master, node, and Prod-Server.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
master	i-0e6616d4889b1debf	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1b	ec2-3-80-2
node	i-0ee527f8caaac634	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-18-20
Prod-Server	i-083735d9ea5830f40	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-

Details for Instance: i-083735d9ea5830f40 (Prod-Server)

- Details** (selected)
- Status and alarms [New](#)
- Monitoring
- Security
- Networking
- Storage
- Tags

Instance summary

Instance ID	44.206.100.36 (Open address)	Public IPv4 address	Private IPv4 addresses
IPv6 address	-	Running	10.0.1.10
Hostname type	IP name in.10.0.1.10 or? internal	Private IP DNS name (IPv4 only)	Public IPv4 DNS

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

Screenshot of a web browser showing the "INSURE-ME" website at 44.206.100.36:8085.

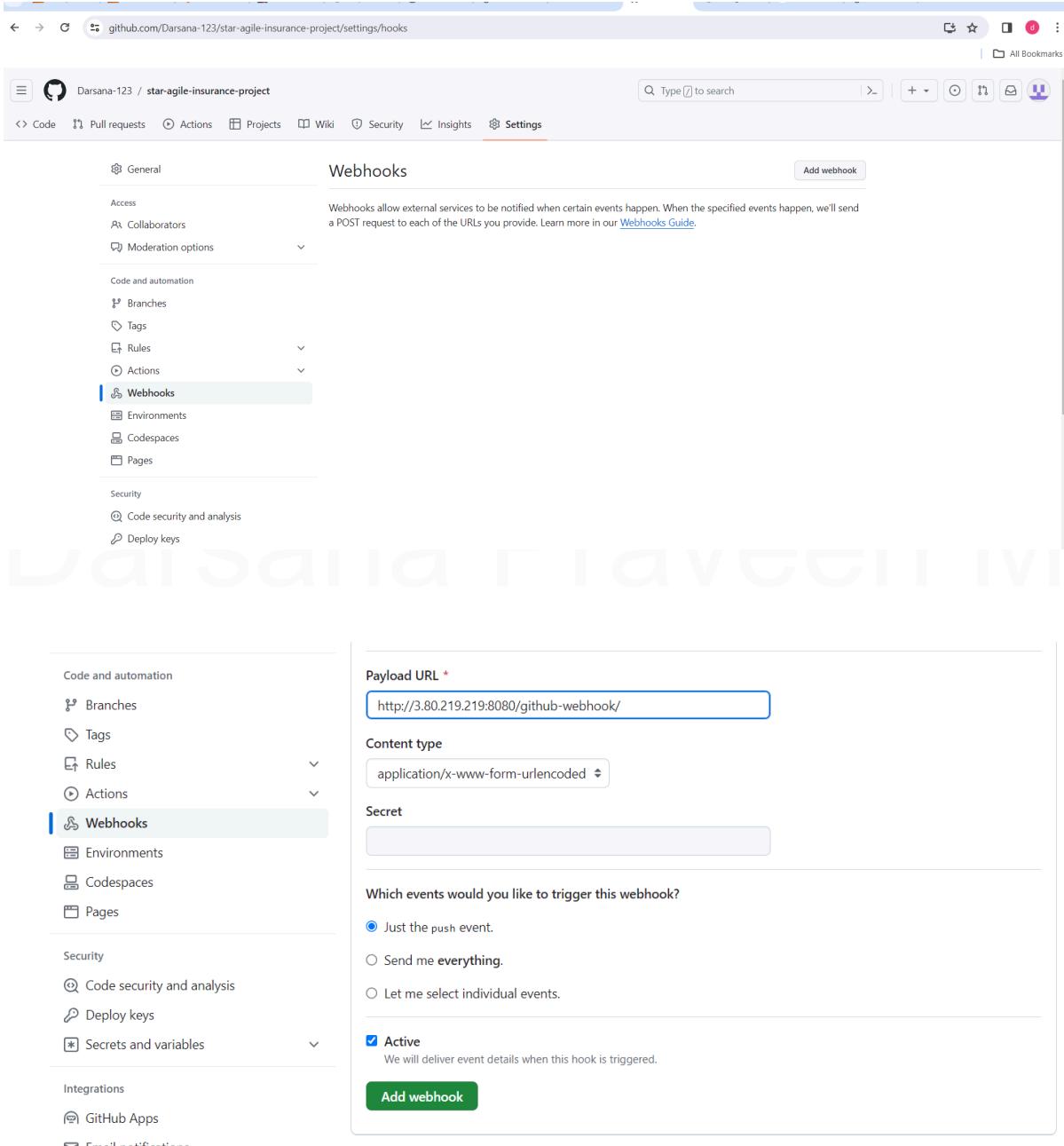
The page features a large blue hexagonal graphic containing various icons related to insurance and family protection, such as a handshake, a piggy bank, a heart, a shield, and a house. The text "PROTECT YOUR FAMILY AND YOUR FUTURE" is prominently displayed in the center of the graphic.

Below the graphic, a paragraph of text explains the benefits of life insurance:

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.

A green "Contact Us" button is located at the bottom left of the page.

Creating a WEBHOOK to our repo for the automatic triggering actions



The screenshot shows the GitHub repository settings page for 'star-agile-insurance-project'. The left sidebar has a 'Webhooks' section selected under 'Code and automation'. The main area is titled 'Webhooks' and contains fields for 'Payload URL' (set to 'http://3.80.219.219:8080/github-webhook/'), 'Content type' (set to 'application/x-www-form-urlencoded'), and a 'Secret' field. Below these are options for triggering events ('Just the push event.', 'Send me everything.', 'Let me select individual events.') and an 'Active' checkbox (which is checked). A green 'Add webhook' button is at the bottom.

The screenshot shows the GitHub Settings page for the repository 'star-agile-insurance-project'. The 'Webhooks' section is selected. A single webhook is listed with the URL <http://3.80.219.219:8080/github-webhook/push>. There are 'Edit' and 'Delete' buttons next to it. The left sidebar shows other settings like General, Collaborators, and Webhooks.

Setting up webhook to the job1

The screenshot shows the Jenkins Pipeline configuration for the project 'ansible-deployment'. Under the 'Build Triggers' section, the 'GitHub hook trigger for GITScm polling' option is checked. Other options like 'Build after other projects are built', 'Build periodically', 'Poll SCM', and 'Trigger builds remotely' are available but not selected.

Verifying the automation of workflow:

Jenkins

Dashboard > ansible-deployment > #12

Status Changes Console Output View as plain text Edit Build Information Delete build '#12' Polling Log Git Build Data Restart from Stage Replay Pipeline Steps Workspaces Previous Build

Console Output

```
Started by GitHub push by Darsana-123
[Pipeline] Start of Pipeline
[Pipeline] node
[Pipeline] stage
[Pipeline] { (checkout code)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/ansible-deployment/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Darsana-123/star-agile-insurance-project.git # timeout=10
Fetching upstream changes from https://github.com/Darsana-123/star-agile-insurance-project.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Darsana-123/star-agile-insurance-project.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 7c75d1384547d6256ef4edc9ccf45f841647bdc2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 7c75d1384547d6256ef4edc9ccf45f841647bdc2 # timeout=10
```

Dashboard > ansible-deployment > #12

```
TASK [Start Docker Service] *****
changed: [18.207.128.252]

TASK [stop existing containers] *****
changed: [18.207.128.252]

TASK [delete all containers] *****
changed: [18.207.128.252]

TASK [Deploy Docker Container] *****
changed: [18.207.128.252]

PLAY RECAP *****
18.207.128.252      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Triggering a new build of terraform-deployment #10
Finished: SUCCESS
```

The screenshot shows the Jenkins interface for a specific build. The left sidebar contains links like Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Delete build '#10', Git Build Data, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build. The main content area is titled 'Console Output' and displays the build logs. The logs show the start of the build by an upstream project, a GitHub push event, and the execution of a Jenkins pipeline stage. It details the checkout of code from GitHub, the use of git version 2.34.1, and the fetching of upstream changes from a GitHub repository. The logs conclude with a check-in of revision 7r76d130d5a76c956af4e0rrf45f8d16d7hr2.

```
Started by upstream project "ansible-deployment" build number 12
originally caused by:
Started by GitHub push by Darsana-123
[Pipeline] Start of Pipeline
[Pipeline] node
[Pipeline] {
[Pipeline] stage
[Pipeline] { (checkout the code from github)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/terraform-deployment/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Darsana-123/star-agile-insurance-project.git # timeout=10
Fetching upstream changes from https://github.com/Darsana-123/star-agile-insurance-project.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/Darsana-123/star-agile-insurance-project.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 7r76d130d5a76c956af4e0rrf45f8d16d7hr2 (refs/remotes/origin/master)
```