Capstone project – ASI Insurance

Name: Kotian Rakshith Padmanabha

Email: rakshithpk21@gmail.com

Github link for the project: https://github.com/kotianrakshith/CapstoneProject2

Objective: To create a micro-service application architecture for an Insurance company through DevOps pipeline and deployment on Docker.

Tools to use:

1. Jenkins
2. Github
3. Docker Hub
4. AWS

Description

ASI Insurance is facing challenges in improving the SLA to its customers due to its organizational growth and existing monolithic application architecture. It requires transformation of the existing architecture to a microservice application architecture, while also implementing DevOps pipeline and automations.

The successful completion of the project will enable ASI Insurance to improve its overall application deployment process, enhance system scalability, and deliver better products and services to its customers.

Task (Activities)

1. Create the Dockerfile, Jenkinsfile, Ansible playbook, and the source file of the static website
2. Upload all the created files to GitHub
3. Go to the terminal and install NodeJS 16
4. Open the browser and access the Jenkins application
5. Create Jenkins pipeline to perform CI/CD for a Docker container
6. Create Docker Hub Credentials and other necessary pre-requisites before running build
7. Set up Docker remote host on AWS and configure deploy stage in pipeline
8. Execute Jenkins Build
9. Access deployed application on Docker container

1. **Creating Github Repositry and file:**

   I have created new git hub repository:
   https://github.com/kotianrakshith/CapstoneProject2

   Here I have stored the source file provided by the simplilearn for building the application.

   I have created the Dockerfile, Jenkins file and Ansible playbook file also in the git hub:

   Jenkins file link:
   https://github.com/kotianrakshith/CapstoneProject2/blob/main/Jenkinsfile

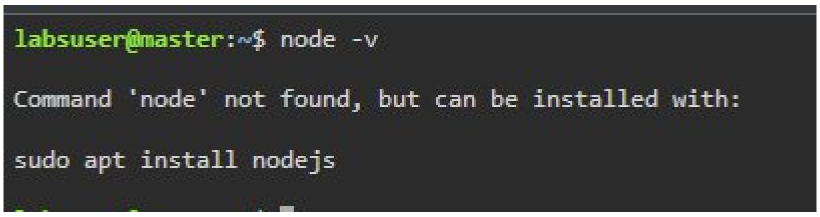
   Dockerfile link: https://github.com/kotianrakshith/CapstoneProject2/blob/main/Dockerfile

   Ansible playbook link:
   https://github.com/kotianrakshith/CapstoneProject2/blob/main/ansible-playbook.yml

   As of now these files are empty, we will add update the file as we go step by step testing and building the pipeline

2. **Install NodeJS 16**

   First we will check if the NodeJS is already present in our system or not

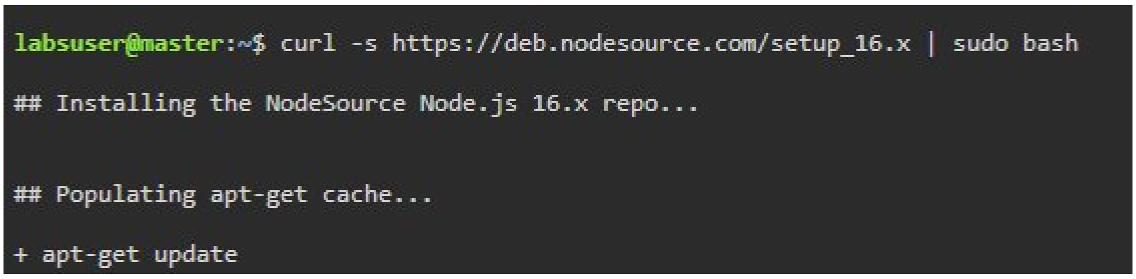


```
labsuser@master:~$ node -v

Command 'node' not found, but can be installed with:

sudo apt install nodejs
```

   As we can see it is not present, so we will proceed with the installation:

   First we will install the repo

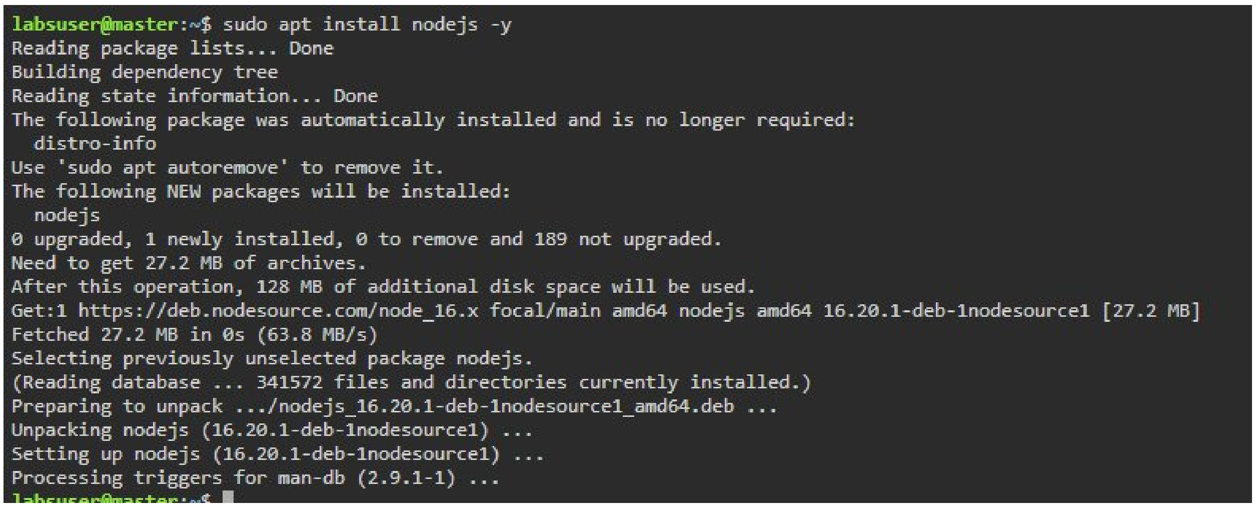
        curl -s https://deb.nodesource.com/setup_16.x | sudo bash



```
labsuser@master:~$ curl -s https://deb.nodesource.com/setup_16.x | sudo bash

## Installing the NodeSource Node.js 16.x repo...


## Populating apt-get cache...

+ apt-get update
```

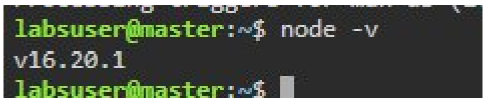
   Then we can run the install command:

        sudo apt install nodejs -y



```
labsuser@master:~$ sudo apt install nodejs -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  distro-info
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 189 not upgraded.
Need to get 27.2 MB of archives.
After this operation, 128 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_16.x focal/main amd64 nodejs amd64 16.20.1-deb-1nodesource1 [27.2 MB]
Fetched 27.2 MB in 0s (63.8 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 341572 files and directories currently installed.)
Preparing to unpack .../nodejs_16.20.1-deb-1nodesource1_amd64.deb ...
Unpacking nodejs (16.20.1-deb-1nodesource1) ...
Setting up nodejs (16.20.1-deb-1nodesource1) ...
Processing triggers for man-db (2.9.1-1) ...
labsuser@master:~$
```

Now when we check we should see that it is installed:

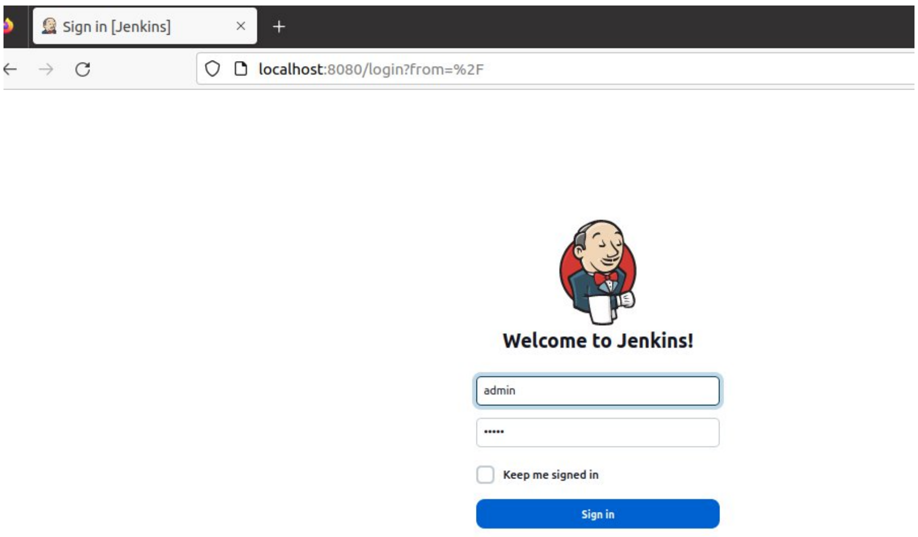


```
labsuser@master:~$ node -v
v16.20.1
labsuser@master:~$
```

3. **Access the Jenkins Application:**

   We can access the jenkins through port 8080 if installed.

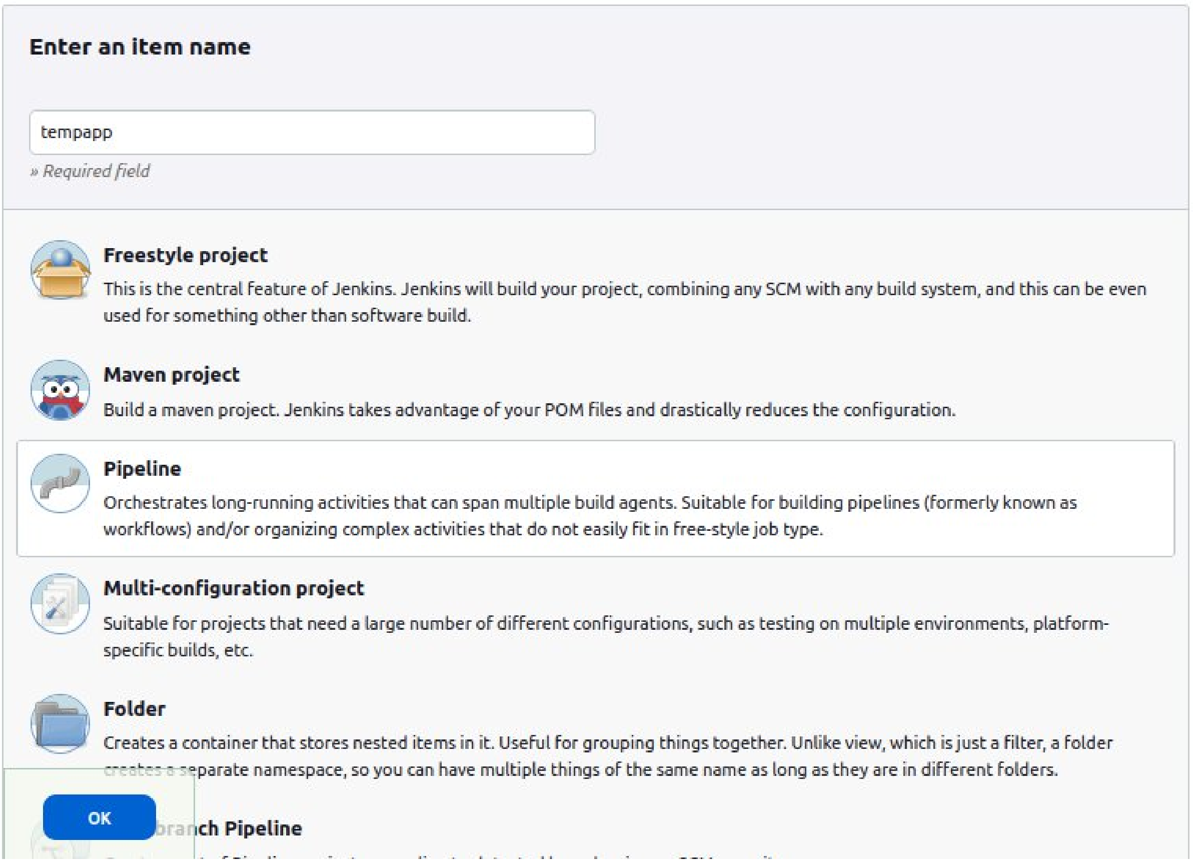
   As I have already installed jenkins in my system I can open localhost:8080 in my browser to access the jenkins:



   As I have already setup, it is asking for my passsword and user for login. If not setup it will ask you to go through the initial setup.

4. **Create Jenkins pipeline:**

   First we will create a test pipeline and we will build each step one by one and in this process we will update all our required files in the Github and in the end we will make the necessary changes to run the complete pipeline for the actual app.


Enter an item name

tempapp

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

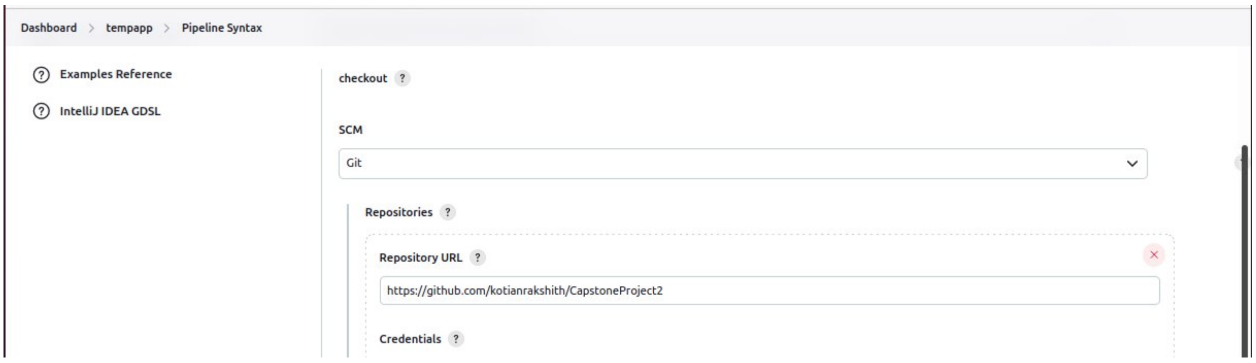
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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

branch Pipeline

For the first stage we will checkout the git and use mvn clean install to build the jar file

To get the checkout script we are using pipeline syntax to populate the script



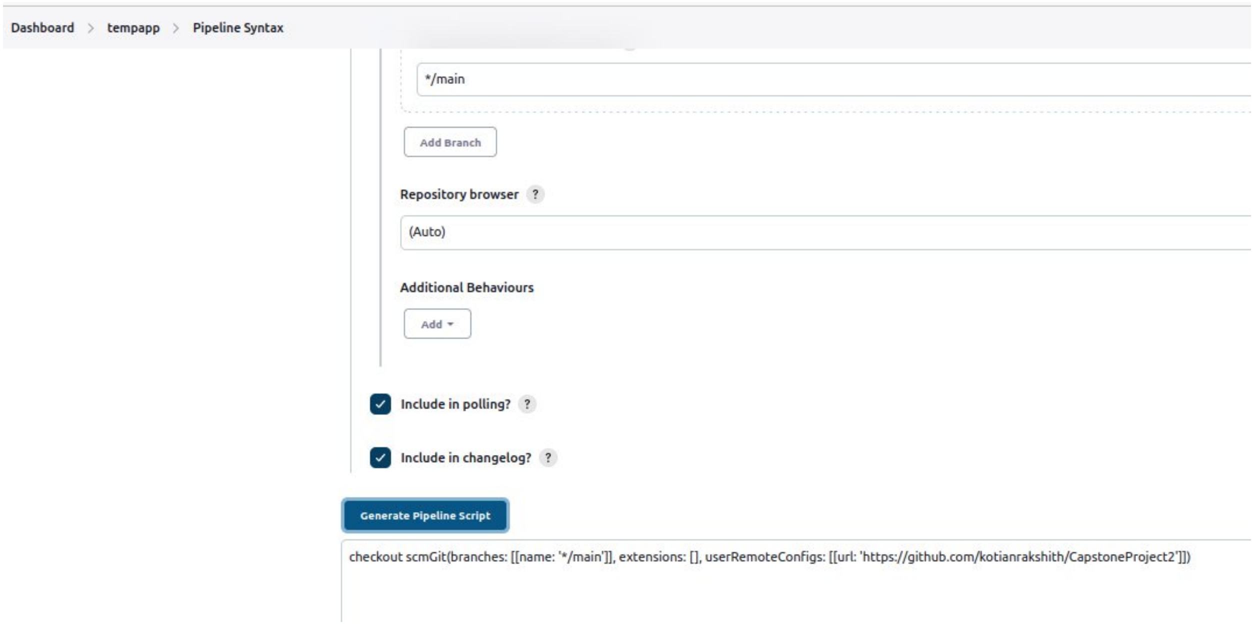
checkout

SCM: Git

Repositories

Repository URL: https://github.com/kotianrakshith/CapstoneProject2

Credentials


Dashboard > tempapp > Pipeline Syntax

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

✓ Include in polling? ?

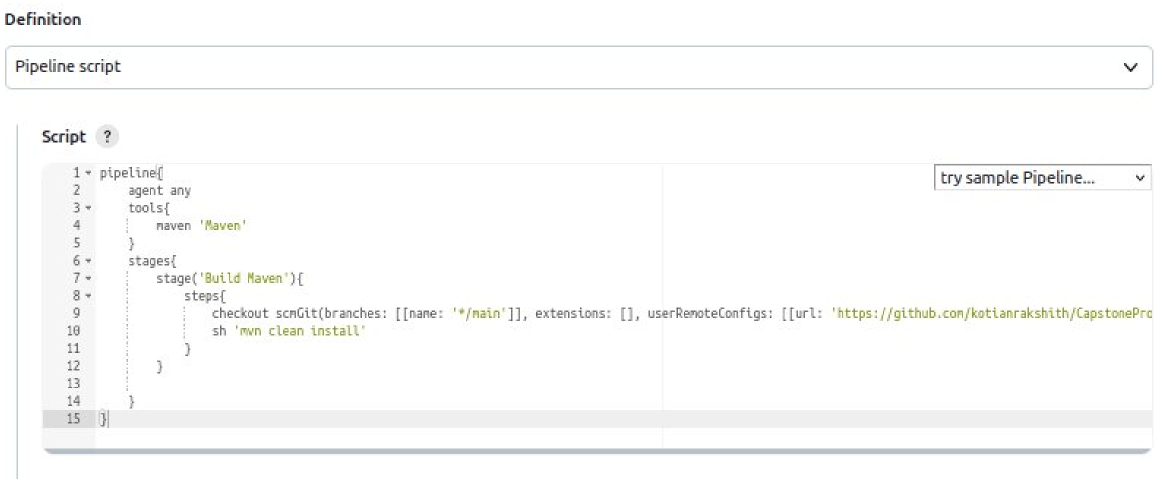
✓ Include in changelog? ?

Generate Pipeline Script

```
checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/kotianrakshith/CapstoneProject2']])
```

Wiith this we can build the belwo script for this stage:

```
stages{
        stage('Build Maven'){
            steps{
                checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url:
'https://github.com/kotianrakshith/CapstoneProject2']])
                sh 'mvn clean install'
            }
        }


}
```


Definition

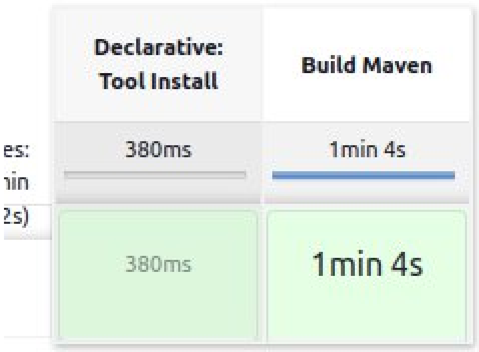
Pipeline script

Script ?

try sample Pipeline...

```
pipeline{
    agent any
    tools{
        maven 'Maven'
    }
    stages{
        stage('Build Maven'){
            steps{
                checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/kotianrakshith/CapstonePro
                sh 'mvn clean install'
            }
        }

    }
}
```

When we build this till here, we can see that this build is complete and successull



Now we can add build docker image to this scirpt.

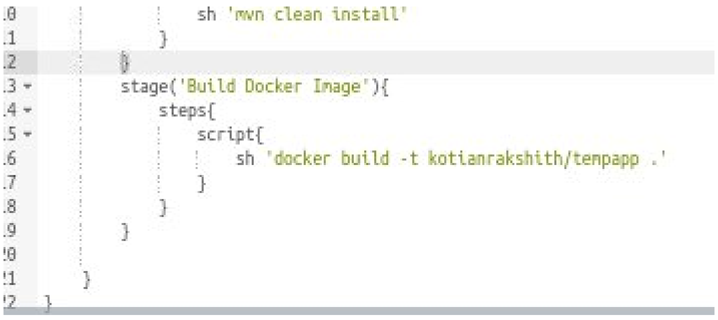
For building the docker image Dockerfile is needed. We can use the below code in Dockerfile to create a simple docker image:

```
FROM openjdk:8-jdk-alpine

COPY target/*.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]
```

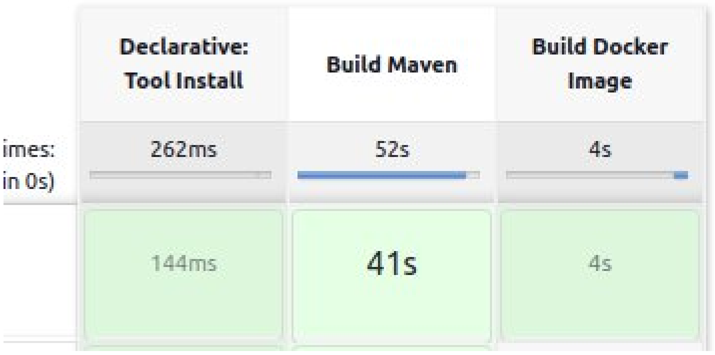
We add this in the github and we add the below script in the pipeline to build the docker image:

```
stage('Build Docker Image'){

      steps{

         script{

            sh 'docker build -t kotianrakshith/tempapp .'

         }

      }

   }
```

(a test name is used, later it will be changed to correct name when we run the final pipeline)



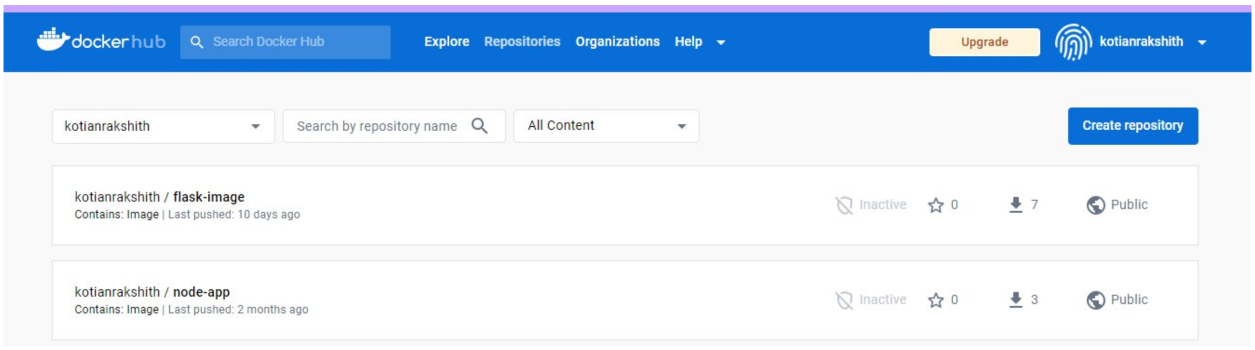
```
                sh 'mvn clean install'
            }
        }
        stage('Build Docker Image'){
            steps{
                script{
                    sh 'docker build -t kotianrakshith/tempapp .'
                }
            }
        }
    }
}
```


	Declarative: Tool Install	Build Maven	Build Docker Image
imes: in 0s)	262ms	52s	4s
	144ms	41s	4s

Now when we build we can see new stage is added and it is also successfull.

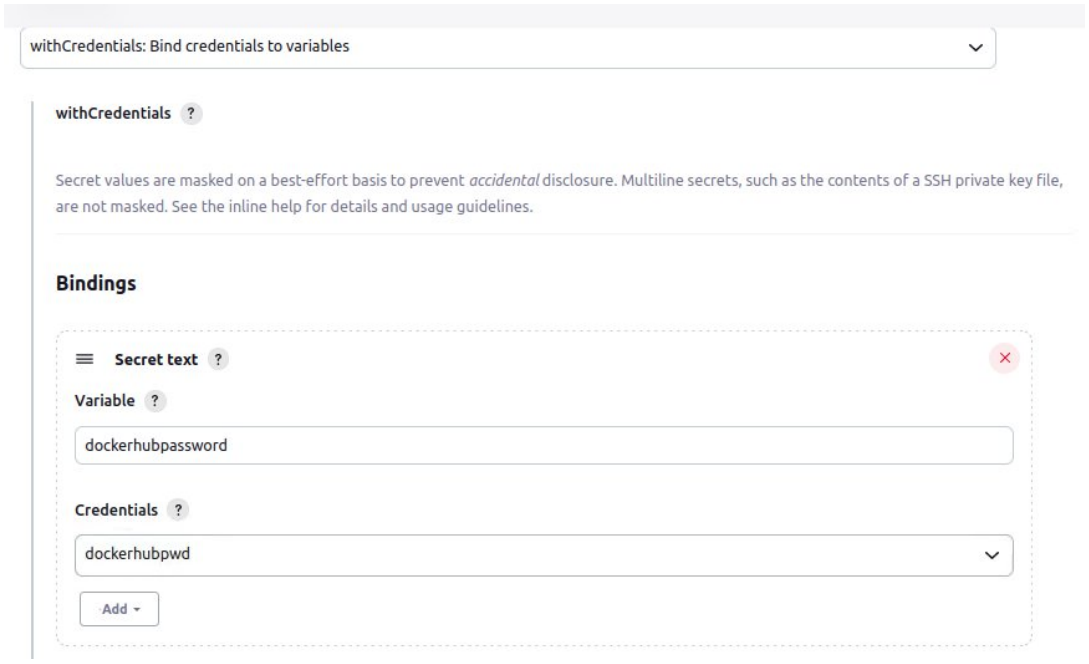
Next stage is pushing the docker image to docker hub.

I already have a dockerhub account with id: kotianrakshith . I will be using this repository to upload the docker image



kotianrakshith / flask-image Contains: Image | Last pushed: 10 days ago	Inactive	0	7	Public
kotianrakshith / node-app Contains: Image | Last pushed: 2 months ago	Inactive	0	3	Public

As dockerhub has credentials, I will be using the dockerhub token and using the pipline syntax to change it into secret text:


withCredentials: Bind credentials to variables

withCredentials ?

Secret values are masked on a best-effort basis to prevent *accidental* disclosure. Multiline secrets, such as the contents of a SSH private key file, are not masked. See the inline help for details and usage guidelines.

Bindings

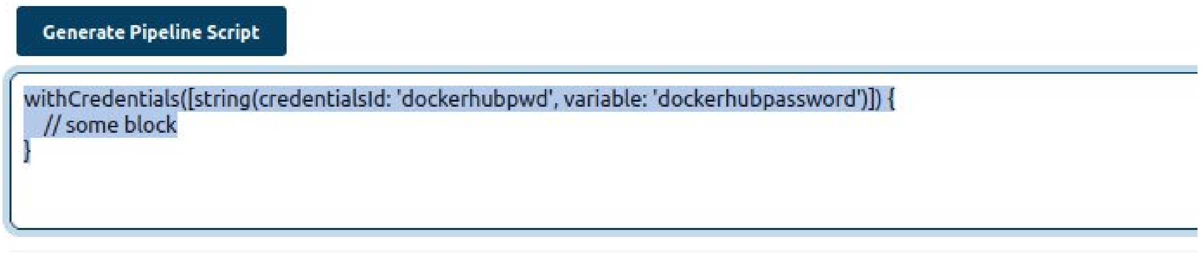
Secret text ?

Variable ?

dockerhubpassword

Credentials ?

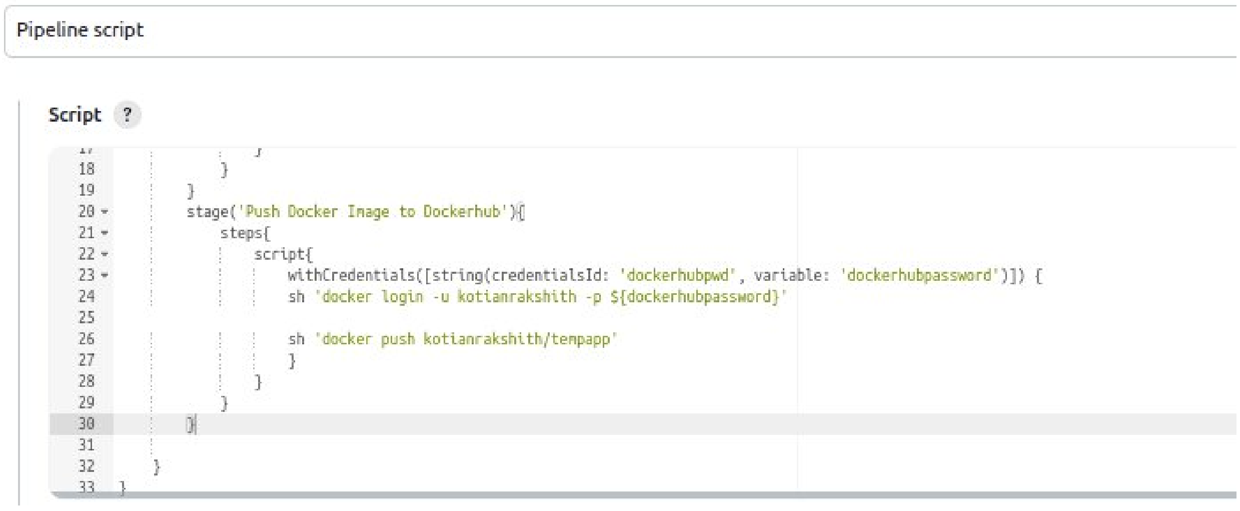
dockerhubpwd



```
withCredentials([string(credentialsId: 'dockerhubpwd', variable: 'dockerhubpassword')]) {
   // some block
}
```

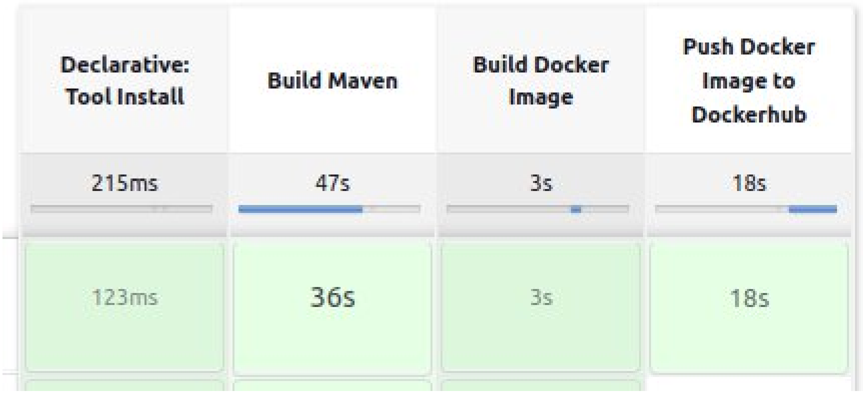
Using this line we can write the script for pushing the image to dockerhub:

```
                        stage('Push Docker Image to Dockerhub'){

                              steps{

                                script{

                                    withCredentials([string(credentialsId: 'dockerhubpwd', variable: 'dockerhubpassword')])
                        {

                                    sh 'docker login -u kotianrakshith -p ${dockerhubpassword}'



                                    sh 'docker push kotianrakshith/tempapp'

                                    }

                                }

                }

            }
```



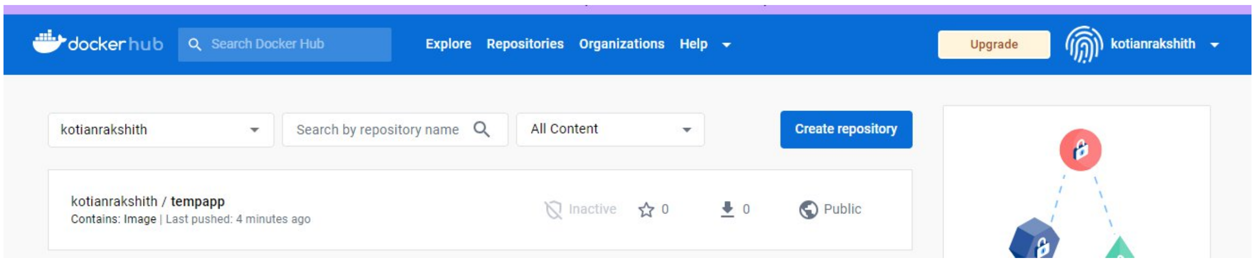
```
            }
        }
        stage('Push Docker Image to Dockerhub'){
            steps{
                script{
                    withCredentials([string(credentialsId: 'dockerhubpwd', variable: 'dockerhubpassword')]) {
                    sh 'docker login -u kotianrakshith -p ${dockerhubpassword}'

                    sh 'docker push kotianrakshith/tempapp'
                    }
                }
            }
        }

    }
}
```

After you save and build, now you can see another stage added and that it is successfully built.



Declarative: Tool Install	Build Maven	Build Docker Image	Push Docker Image to Dockerhub
215ms	47s	3s	18s
123ms	36s	3s	18s

That means there should be an image uploaded in the repository



kotianrakshith / tempapp
Contains: Image | Last pushed: 4 minutes ago
Inactive ☆ 0 ⬇ 0 Public

As you can see the test image has been uploaded.

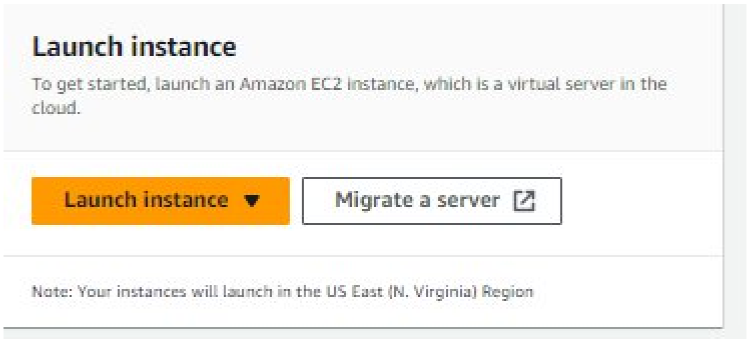
Now only the last step is pending which is deploy the container on docker.

5. **<u>Setup Docker Remote host on AWS.</u>**

For this project I will setup an EC2 instance and install the docker on the virtual machine.

Then I will configure Ansible in my jenkins server with this node so ansible can help in deploying the docker container.

First we will go to launch instance in EC2 instance:

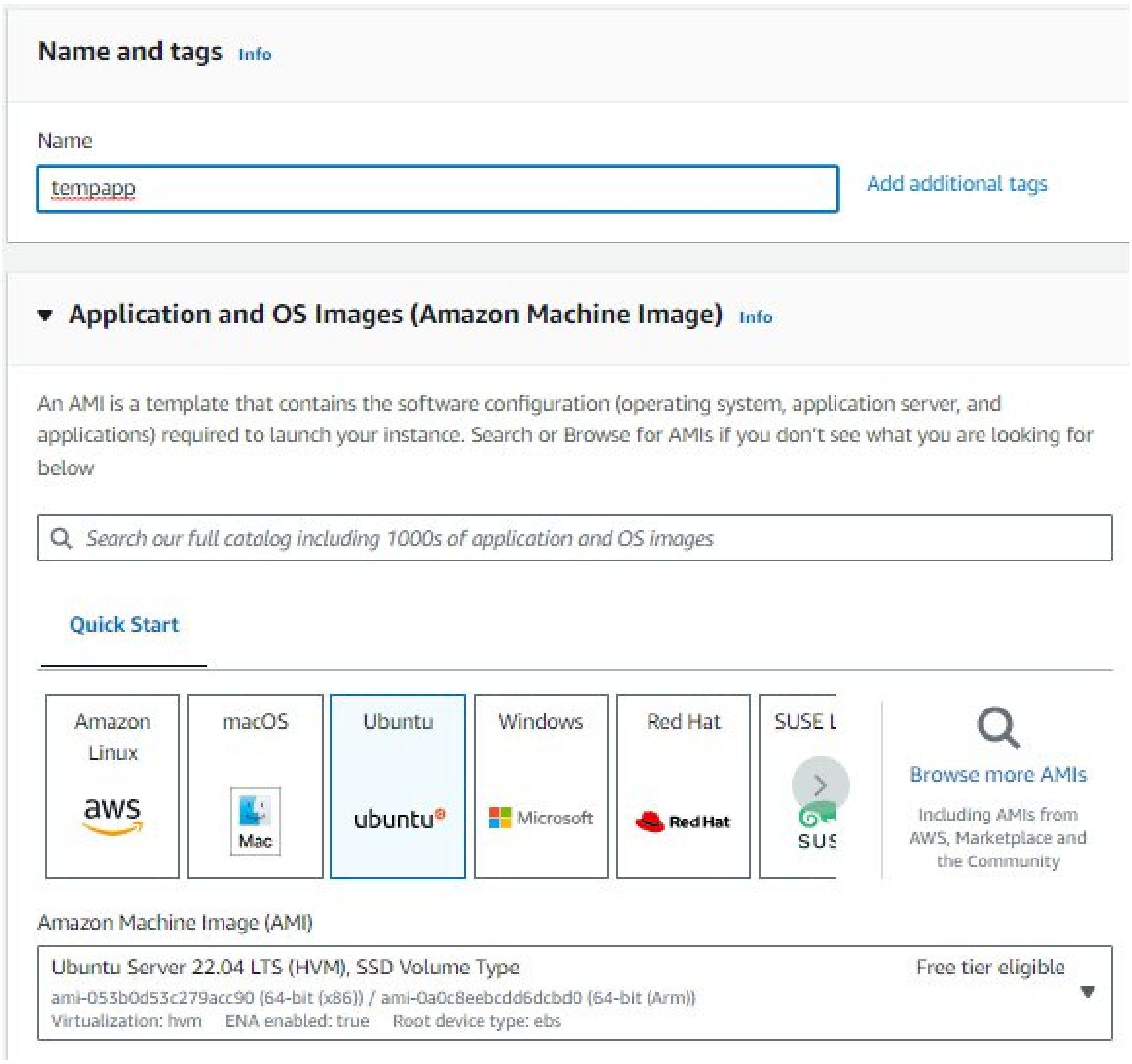


Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US East (N. Virginia) Region

Then chose a os image(im chosing ubutnu image with t2.micro instance type)


Name and tags Info

Name

tempapp

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

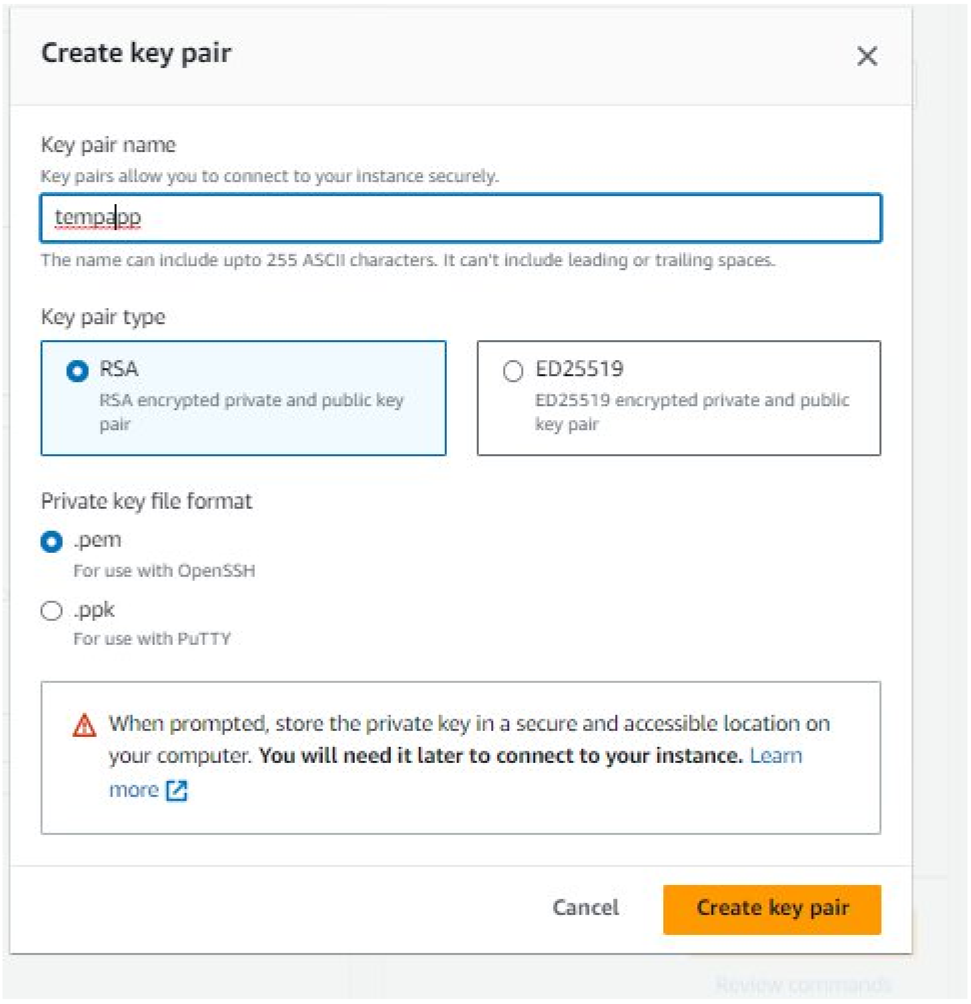
Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-053b0d53c279acc90 (64-bit (x86)) / ami-0a0c8eebcdd6dcbd0 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Then create a key pair so you can use it to login to your system:



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

tempapp

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

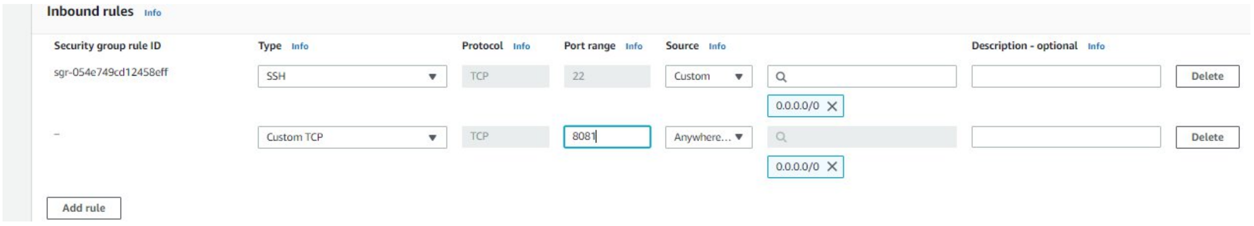
[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Then we keep all the configuration as it is and create the EC2 instance.

Then we edit inbound rules to add the port 8081, so we can access the application when we deploy it.



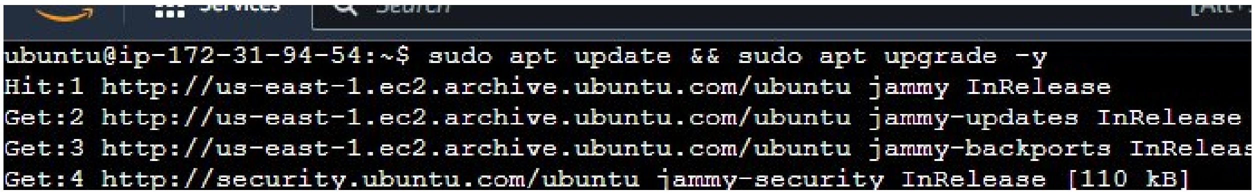
Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-054e749cd12458eff	SSH	TCP	22	Custom	0.0.0.0/0	
–	Custom TCP	TCP	8081	Anywhere...	0.0.0.0/0	

Now we connect connect to the ec2 instance and we install docker.

First we upgrade the system

sudo apt update && sudo apt upgrade -y



```
ubuntu@ip-172-31-94-54:~$ sudo apt update && sudo apt upgrade -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InReleas
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
```

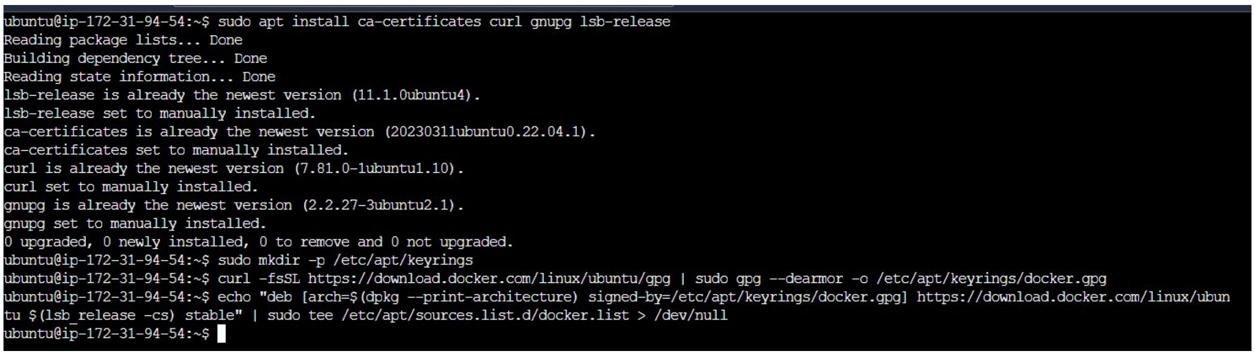
Now we install the packages, keys, required repositories:

sudo apt install ca-certificates curl gnupg lsb-release

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

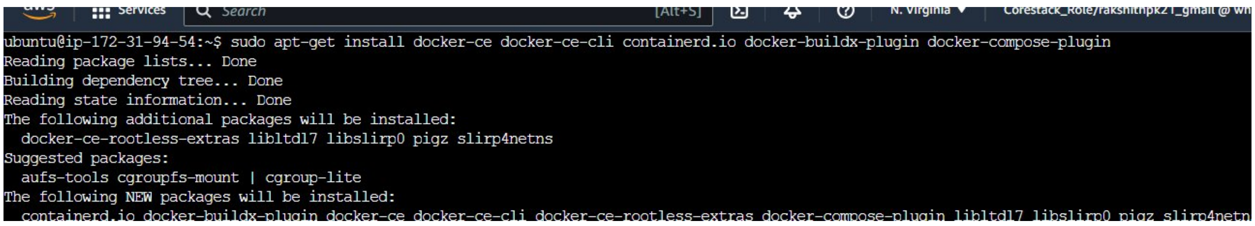


```
ubuntu@ip-172-31-94-54:~$ sudo apt install ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.10).
curl set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ip-172-31-94-54:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-94-54:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
ubuntu@ip-172-31-94-54:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubun
tu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
ubuntu@ip-172-31-94-54:~$
```

Now run the update command again(sudo apt update)

Now install the docker ce:

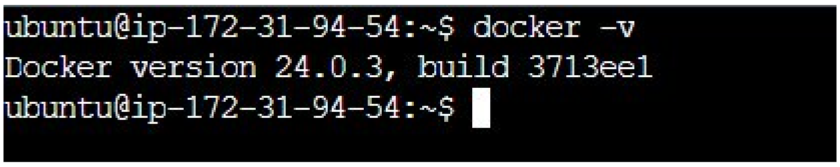
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin



```
ubuntu@ip-172-31-94-54:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netn
```

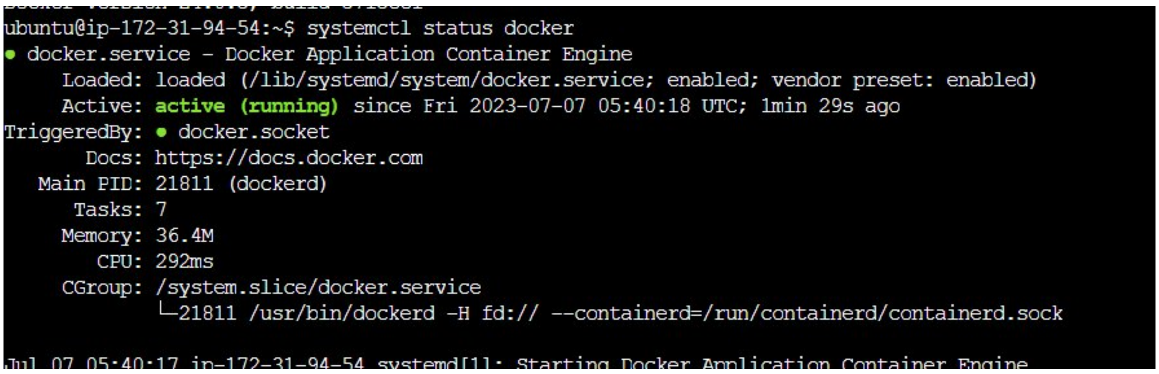
Now check the docker version:

docker -v



```
ubuntu@ip-172-31-94-54:~$ docker -v
Docker version 24.0.3, build 3713ee1
ubuntu@ip-172-31-94-54:~$
```

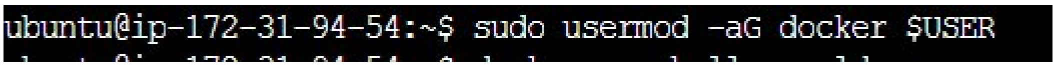
If you check the docker you see it is running:



```
ubuntu@ip-172-31-94-54:~$ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-07-07 05:40:18 UTC; 1min 29s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 21811 (dockerd)
      Tasks: 7
     Memory: 36.4M
        CPU: 292ms
     CGroup: /system.slice/docker.service
             └─21811 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 07 05:40:17 ip-172-31-94-54 systemd[1]: Starting Docker Application Container Engine
```

Then you can add your user for docker group

sudo usermod -aG docker $USER



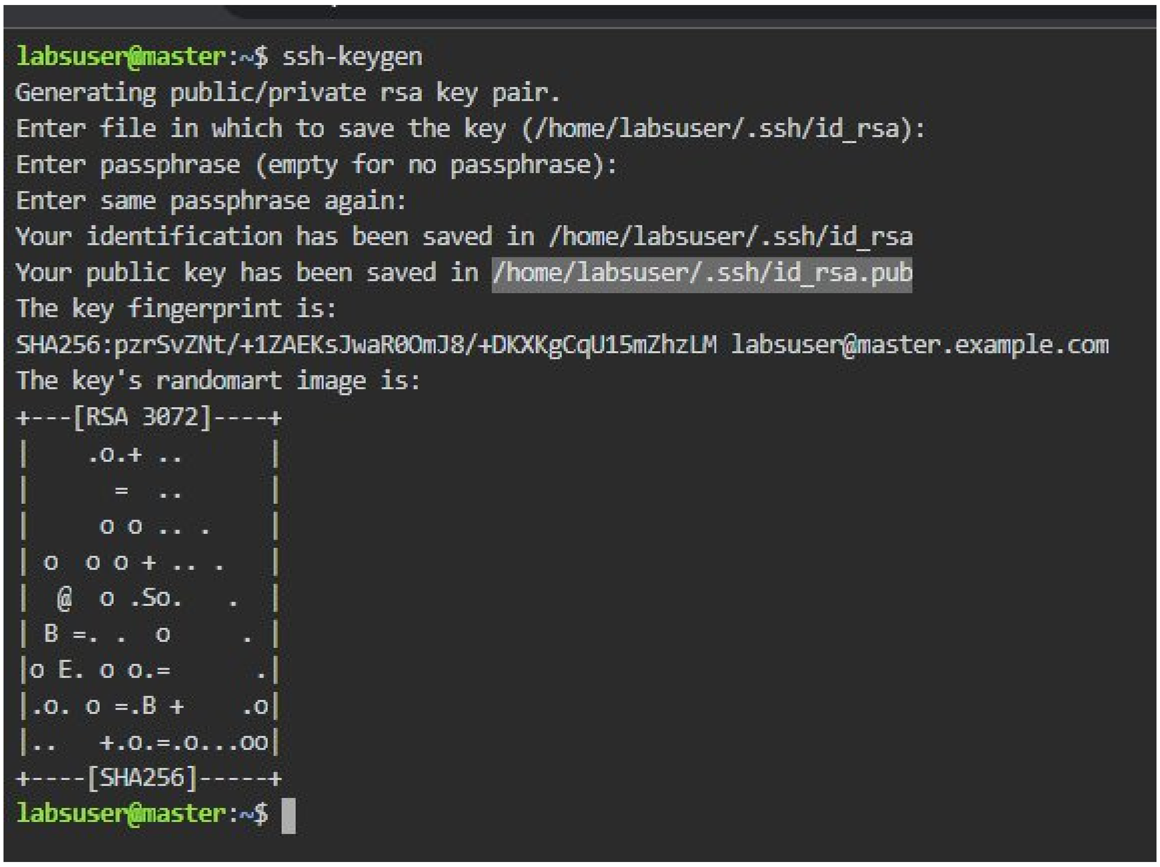
```
ubuntu@ip-172-31-94-54:~$ sudo usermod -aG docker $USER
```

Then reload the session using command: newgrp docker

(As Im using the simplilearn AWS system that resets after few hours, I may be using different vms as I progress)

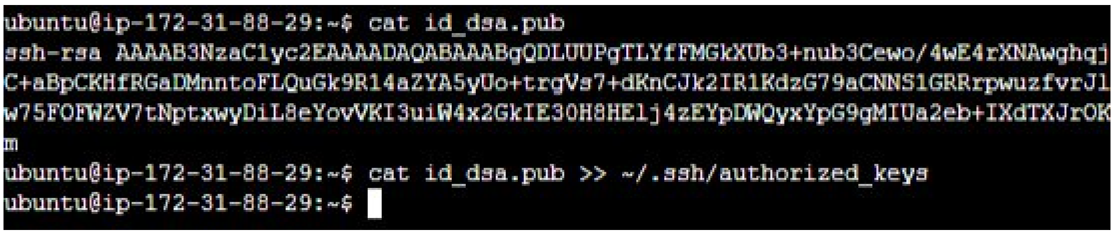
Now we will setup our Ansible installed on our jenkins system with passwordless access to this node.

First create public key pair using ssh-keygen command:



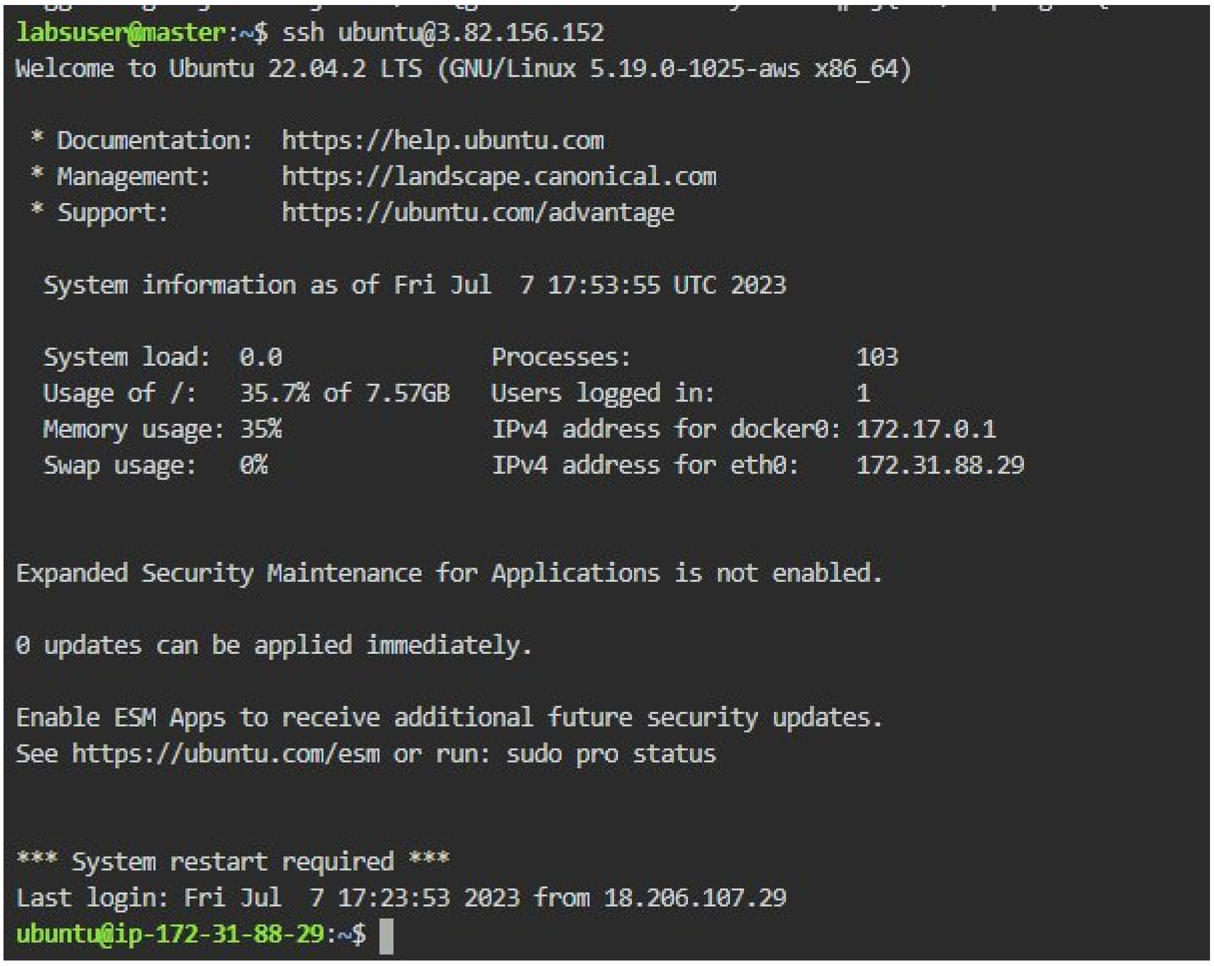
```
labsuser@master:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/labsuser/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/labsuser/.ssh/id_rsa
Your public key has been saved in /home/labsuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:pzrSvZNt/+1ZAEKsJwaR0OmJ8/+DKXKgCqU15mZhzLM labsuser@master.example.com
The key's randomart image is:
+---[RSA 3072]----+
|     .o.+ ..     |
|       =  ..     |
|      o o .. .   |
| o  o o + .. .   |
|  @  o .So.   .  |
| B =. .  o     . |
|o E. o o.=      .|
|.o. o =.B +    .o|
|..   +.o.=.o...oo|
+----[SHA256]-----+
labsuser@master:~$
```

Then copy the public key and put in the authorized_keys file in the EC2 instance



```
ubuntu@ip-172-31-88-29:~$ cat id_dsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDLUUPgTLYfFMGkXUb3+nub3Cewo/4wE4rXNAwghqj
C+aBpCKHfRGaDMnntoFLQuGk9R14aZYA5yUo+trgVs7+dKnCJk2IR1KdzG79aCNNS1GRRrpwuzfvrJl
w75FOFWZV7tNptxwyDiL8eYovVKI3uiW4x2GkIE30H8HElj4zEYpDWQyxYpG9gMIUa2eb+IXdTXJrOK
m
ubuntu@ip-172-31-88-29:~$ cat id_dsa.pub >> ~/.ssh/authorized_keys
ubuntu@ip-172-31-88-29:~$
```

Now if i run the ssh command it should allow you to login



```
labsuser@master:~$ ssh ubuntu@3.82.156.152
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jul  7 17:53:55 UTC 2023

  System load:  0.0               Processes:                103
  Usage of /:   35.7% of 7.57GB   Users logged in:          1
  Memory usage: 35%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.88.29


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Fri Jul  7 17:23:53 2023 from 18.206.107.29
ubuntu@ip-172-31-88-29:~$
```

Let us add this node as a host in ansible host file

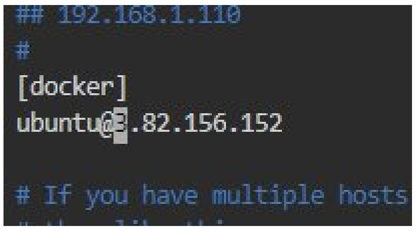
We will go to

    sudo vi /etc/ansible/hosts

Here i will add it as a group docker

[docker]

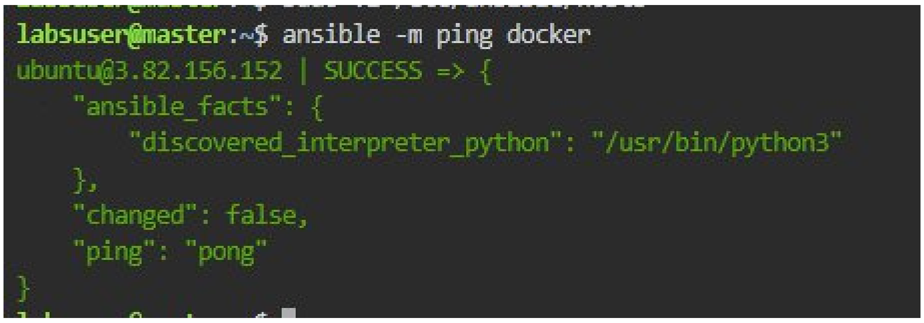
ubuntu@3.82.156.152



```
## 192.168.1.110
#
[docker]
ubuntu@3.82.156.152

# If you have multiple hosts
```

Now we will ping to see if it works

ansible -m ping docker



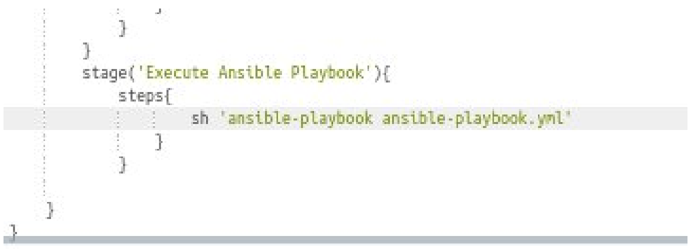
```
labsuser@master:~$ ansible -m ping docker
ubuntu@3.82.156.152 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Now we can write the ansible playbook for deploying the docker and write the final part of the script for our deployment.

```
---
- hosts: docker
  tasks:
    - name: Stop Container if any
      docker_container:
        name: tempapp
        state: stopped
      ignore_errors: true
    - name: deploy testapp docker container
      docker_container:
        image: docker.io/kotianrakshith/tempapp
        name: tempapp
        state: started
        auto_remove: true
        ports:
          - "8081:8081"
```

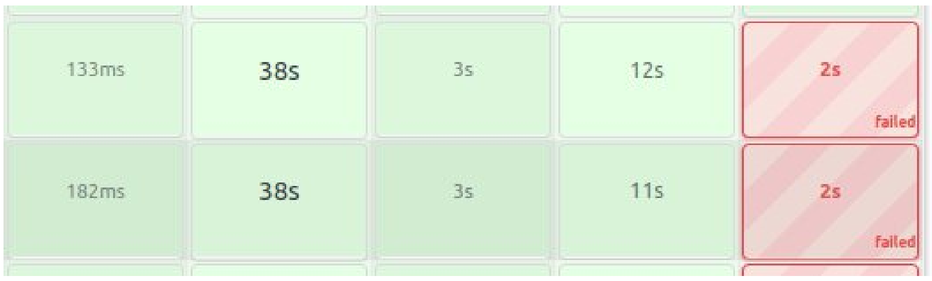
And the script for the same

```
        stage('Execute Ansible Playbook'){
            steps{
                sh 'ansible-playbook ansible-playbook.yml'
            }
        }
```



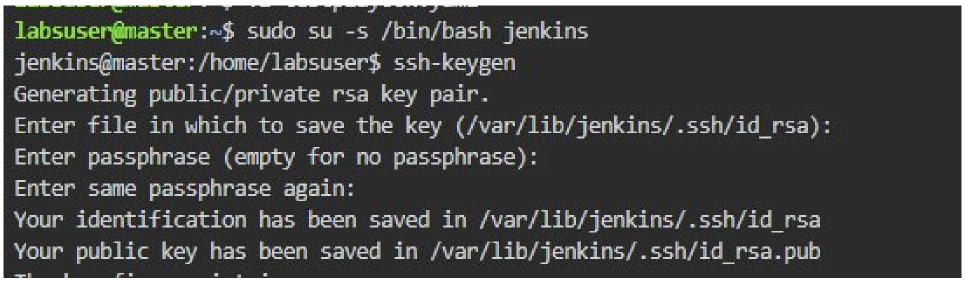
```
            }
        }
        stage('Execute Ansible Playbook'){
            steps{
                sh 'ansible-playbook ansible-playbook.yml'
            }
        }
    }
}
```

First it failed :



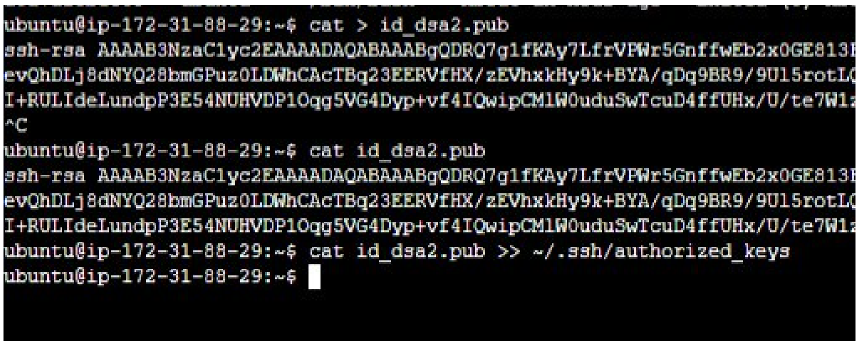
133ms	38s	3s	12s	2s failed
182ms	38s	3s	11s	2s failed

And as the error was it was unable to ssh into ec2 instance, we had to repeat adding public key for the jenkins user



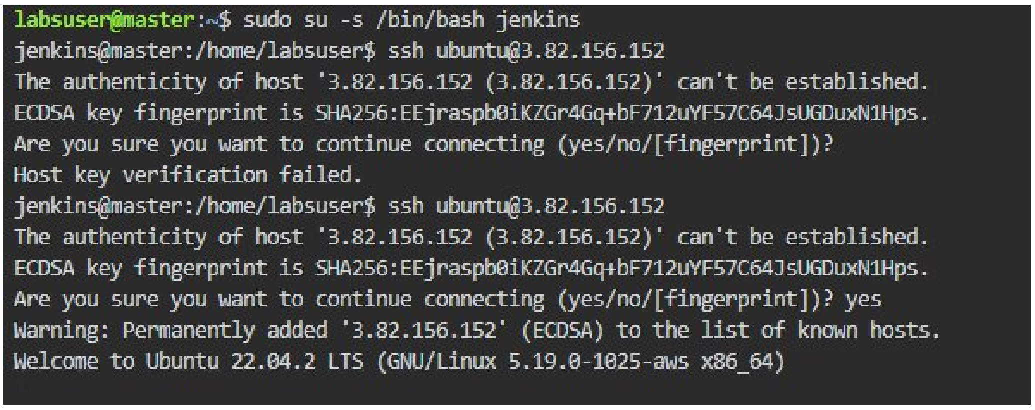
```
labsuser@master:~$ sudo su -s /bin/bash jenkins
jenkins@master:/home/labsuser$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
```

Then we added that in the authorized_keys file in ec2 instance



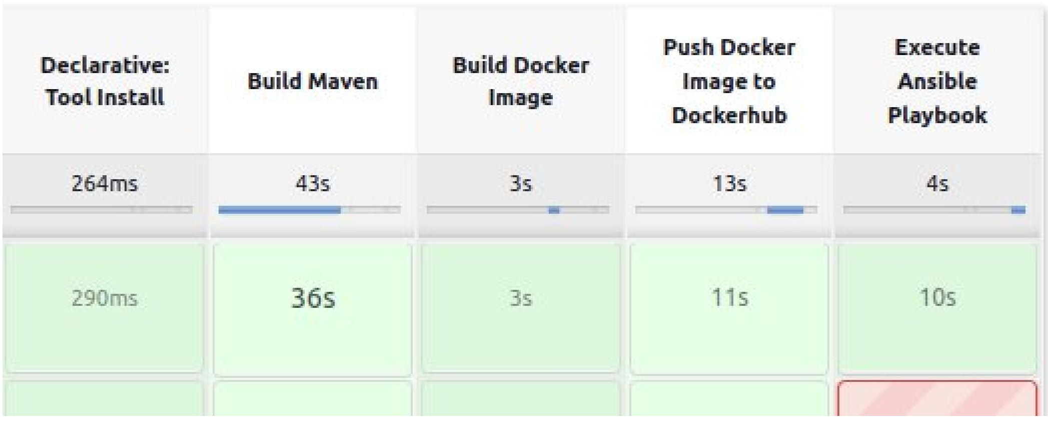
```
ubuntu@ip-172-31-88-29:~$ cat > id_dsa2.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDRQ7g1fKAy7LfrVPWr5GnffwEb2x0GE813
evQhDLj8dNYQ28bmGPuz0LDWhCAcTBq23EERVfHX/zEVhxkHy9k+BYA/qDq9BR9/9U15rotL
I+RULIdeLundpP3E54NUHVDP1Oqg5VG4Dyp+vf4IQwipCM1W0uduSwTcuD4ffUHx/U/te7W1
^C
ubuntu@ip-172-31-88-29:~$ cat id_dsa2.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDRQ7g1fKAy7LfrVPWr5GnffwEb2x0GE813
evQhDLj8dNYQ28bmGPuz0LDWhCAcTBq23EERVfHX/zEVhxkHy9k+BYA/qDq9BR9/9U15rotL
I+RULIdeLundpP3E54NUHVDP1Oqg5VG4Dyp+vf4IQwipCM1W0uduSwTcuD4ffUHx/U/te7W1
ubuntu@ip-172-31-88-29:~$ cat id_dsa2.pub >> ~/.ssh/authorized_keys
ubuntu@ip-172-31-88-29:~$
```

Then we ssh into the instace to see if we were able to ssh correctly



```
labsuser@master:~$ sudo su -s /bin/bash jenkins
jenkins@master:/home/labsuser$ ssh ubuntu@3.82.156.152
The authenticity of host '3.82.156.152 (3.82.156.152)' can't be established.
ECDSA key fingerprint is SHA256:EEjraspb0iKZGr4Gq+bF712uYF57C64JsUGDuxN1Hps.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Host key verification failed.
jenkins@master:/home/labsuser$ ssh ubuntu@3.82.156.152
The authenticity of host '3.82.156.152 (3.82.156.152)' can't be established.
ECDSA key fingerprint is SHA256:EEjraspb0iKZGr4Gq+bF712uYF57C64JsUGDuxN1Hps.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.82.156.152' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)
```

Then we run the build again and we can see that it was success:



Declarative: Tool Install	Build Maven	Build Docker Image	Push Docker Image to Dockerhub	Execute Ansible Playbook
264ms	43s	3s	13s	4s
290ms	36s	3s	11s	10s

6. **<u>Execute the Jenkins build:</u>**

Now that we have tested, we can create the proper jenkins file, ansible file by just changing the naming in our app.

We will use the name 'insuranceapp' for this application.

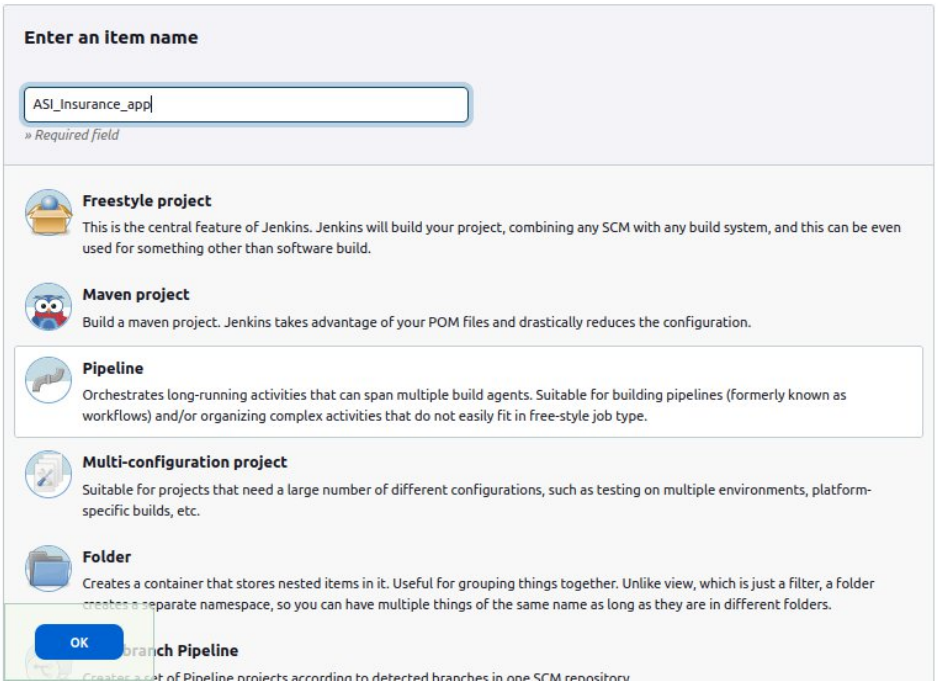
I have made the changes to the code and uploded all the file in the github

Jenkins file link: https://github.com/kotianrakshith/CapstoneProject2/blob/main/Jenkinsfile

Dockerfile link: https://github.com/kotianrakshith/CapstoneProject2/blob/main/Dockerfile

Ansible playbook link: https://github.com/kotianrakshith/CapstoneProject2/blob/main/ansible-playbook.yml

Now we will create the actual final pipeline using jenkins file:



Enter an item name

ASI_Insurance_app

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

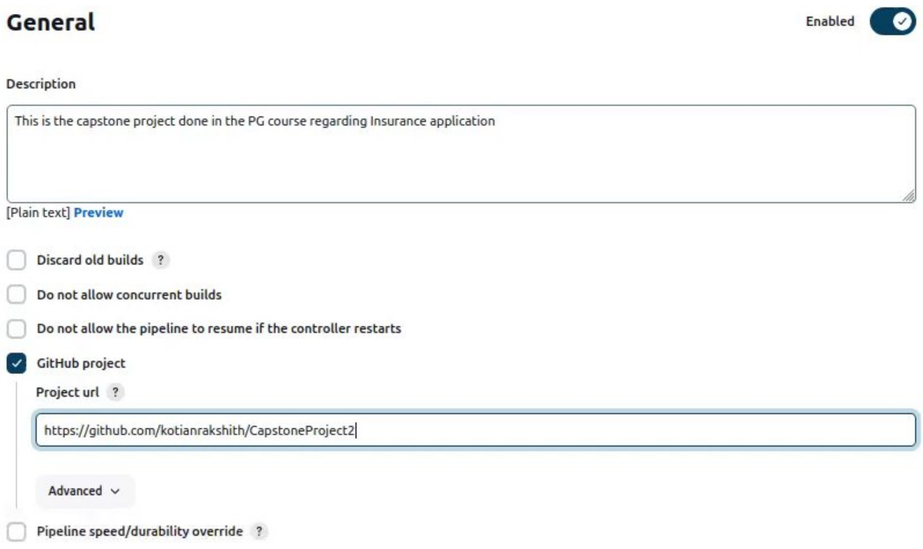
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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

OK

ch Pipeline
t of Pipeline projects according to detected branches in one SCM repository

We will give description and github url:



General Enabled

Description

This is the capstone project done in the PG course regarding Insurance application

[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

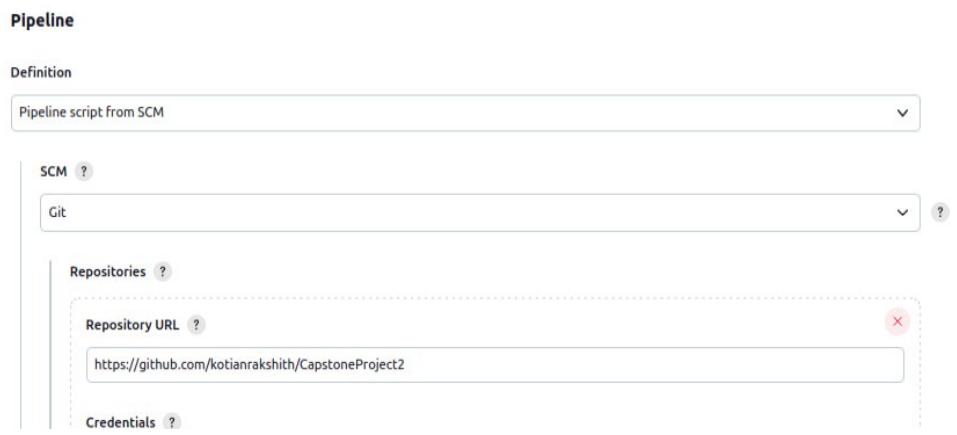
Project url

https://github.com/kotianrakshith/CapstoneProject2

Advanced

- [ ] Pipeline speed/durability override

In the pipeline we are choosing pipeline script from SCM:



Pipeline

Definition

Pipeline script from SCM

SCM

Git

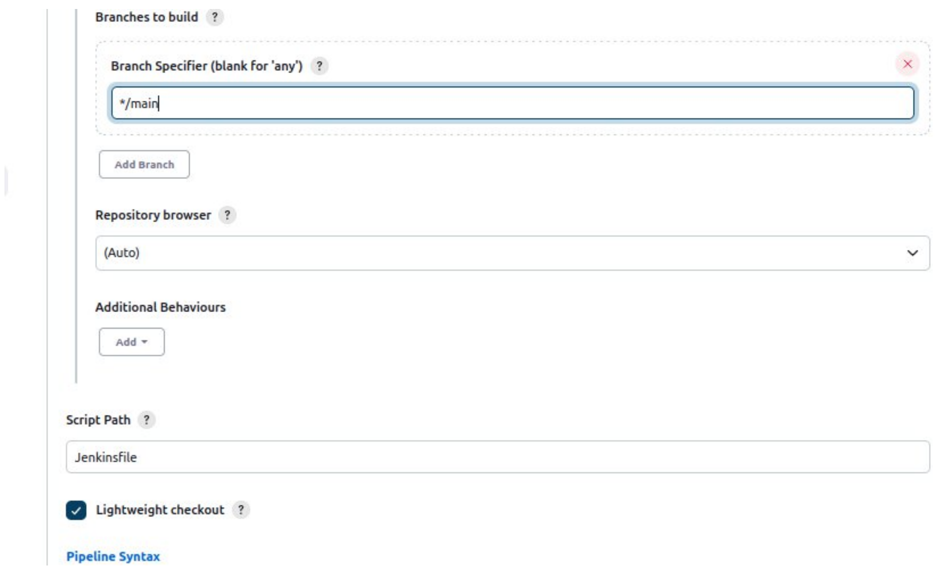
Repositories

Repository URL

https://github.com/kotianrakshith/CapstoneProject2

Credentials

Give the correct branch name and jenkinsfile name



Branches to build

Branch Specifier (blank for 'any')

*/main

Repository browser

(Auto)

Additional Behaviours

Script Path

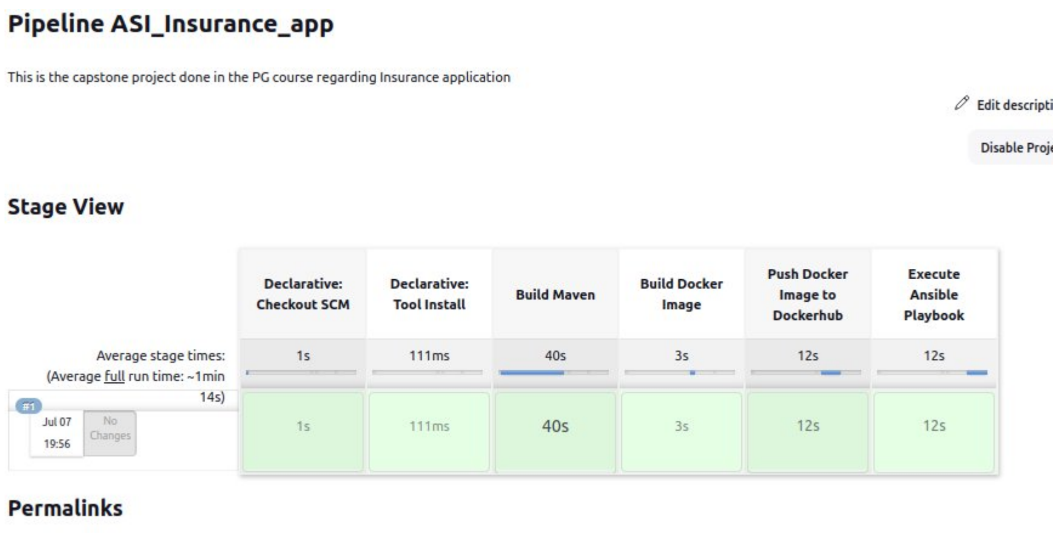
Jenkinsfile

✓ Lightweight checkout

Pipeline Syntax

Once saved you can build the pipeline:

As we have tested all the script before it should work as expected and build correctly.



Pipeline ASI_Insurance_app

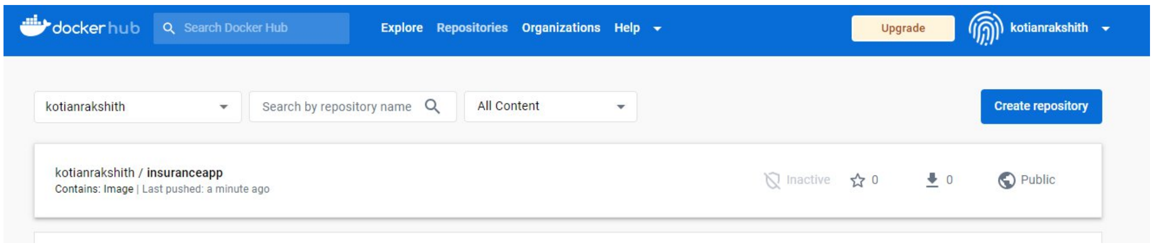
This is the capstone project done in the PG course regarding Insurance application

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	Build Maven	Build Docker Image	Push Docker Image to Dockerhub	Execute Ansible Playbook
Average stage times: (Average full run time: ~1min 14s)	1s	111ms	40s	3s	12s	12s
#1 Jul 07 19:56 No Changes	1s	111ms	40s	3s	12s	12s

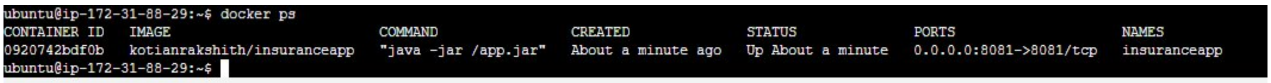
Permalinks

Now we can check dockerhub if the image has been uploaded:



kotianrakshith / insuranceapp
Contains: Image | Last pushed: a minute ago
Inactive 0 0 Public

Now let us check in the ec2 instance if the container is deployed:



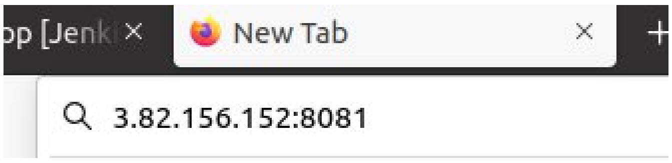
```
ubuntu@ip-172-31-88-29:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                CREATED          STATUS           PORTS                    NAMES
0920742bdf0b   kotianrakshith/insuranceapp    "java -jar /app.jar"   About a minute ago   Up About a minute   0.0.0.0:8081->8081/tcp   insuranceapp
ubuntu@ip-172-31-88-29:~$
```

As we can see it is deployed correctly.

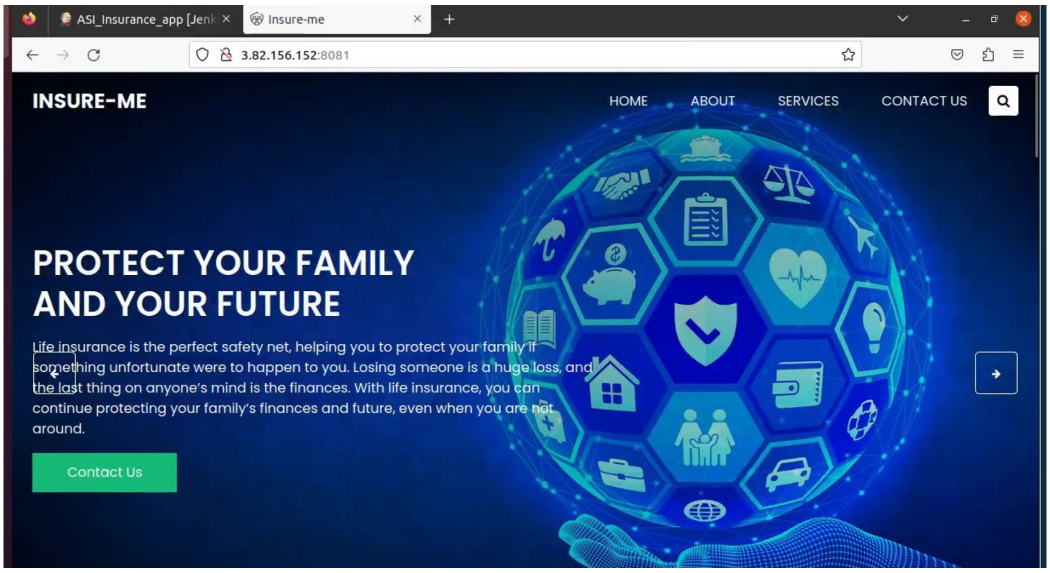
7. Testing the deployment:

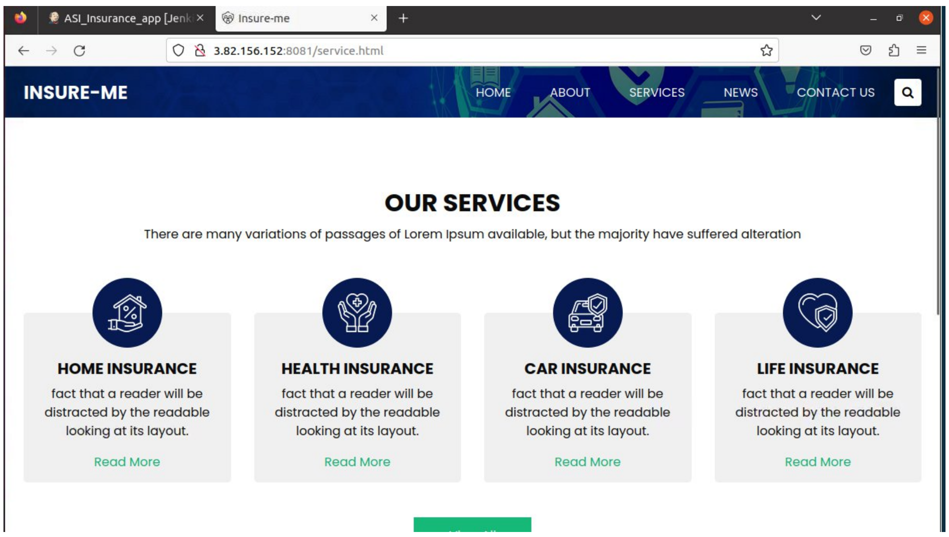
Now as we have deployed in aws and used port 8081, we should we able to access our deployed applcation through public ip of the ec2 instance.

publicip:8081 in our case 3.82.156.152:8081



As we expected it is loading correctly


ASI_Insurance_app [Jenk
Insure-me
3.82.156.152:8081
INSURE-ME
HOME
ABOUT
SERVICES
CONTACT US
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.
Contact Us


ASI_Insurance_app [Jenk | Insure-me

3.82.156.152:8081/service.html

INSURE-ME

HOME ABOUT SERVICES NEWS CONTACT US

OUR SERVICES

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration

HOME INSURANCE

fact that a reader will be distracted by the readable looking at its layout.

Read More

HEALTH INSURANCE

fact that a reader will be distracted by the readable looking at its layout.

Read More

CAR INSURANCE

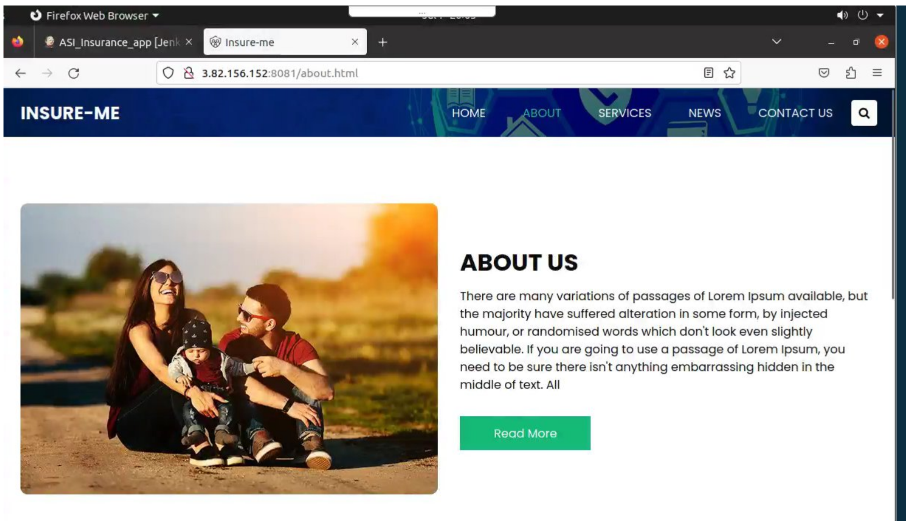
fact that a reader will be distracted by the readable looking at its layout.

Read More

LIFE INSURANCE

fact that a reader will be distracted by the readable looking at its layout.

Read More


Firefox Web Browser

ASI_Insurance_app [Jenk | Insure-me

3.82.156.152:8081/about.html

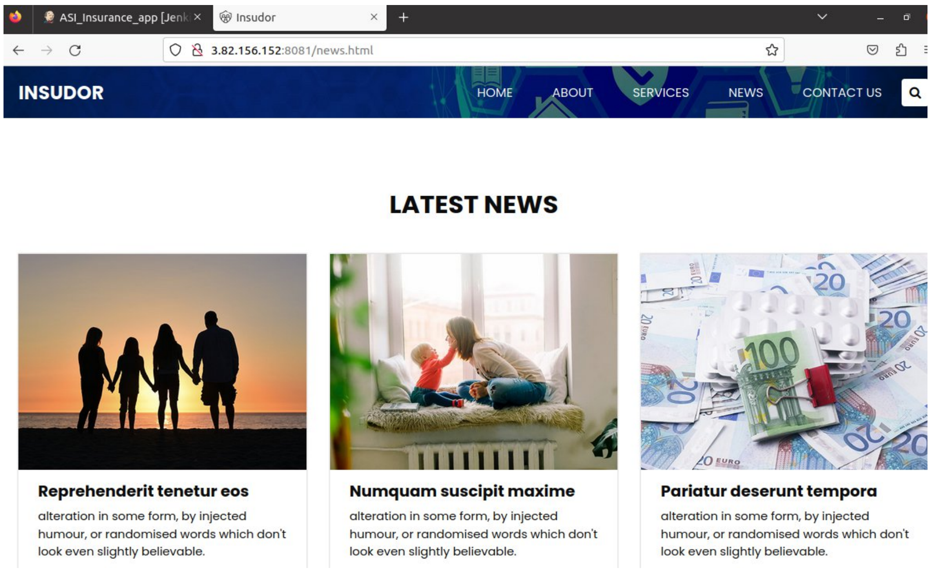
INSURE-ME

HOME ABOUT SERVICES NEWS CONTACT US

ABOUT US

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All

Read More


ASI_Insurance_app [Jenk | Insudor

3.82.156.152:8081/news.html

INSUDOR

HOME ABOUT SERVICES NEWS CONTACT US

LATEST NEWS

Reprehenderit tenetur eos

alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

Numquam suscipit maxime

alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

Pariatur deserunt tempora

alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

So that concludes the project, we can improve on this project by making this an automated build by using poll scm or by using webhooks so it will run whenever there is a build made. But you can also click on build now whenever there is a change done and it should deploy the updated application to the docker container in your ec2 instance.