

Certification Project – Insure Me

Domain name: Insurance project

Assignment name: **Certification Project – Insure Me:**

submitted by: **Subodh Kamble**

submitted on: 23/05/25

Overview of project : Insure Me is a Global leading Insurance provider based out of the 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 grew, 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. Ensure me has decided to use AWS as the 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.

Main Objectives : 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.

Technology used :

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

Terraform - for infrastructure deployment

AWS - For creating ec2 machines as servers and deploy the web application

docker-compose - For container orchestration.

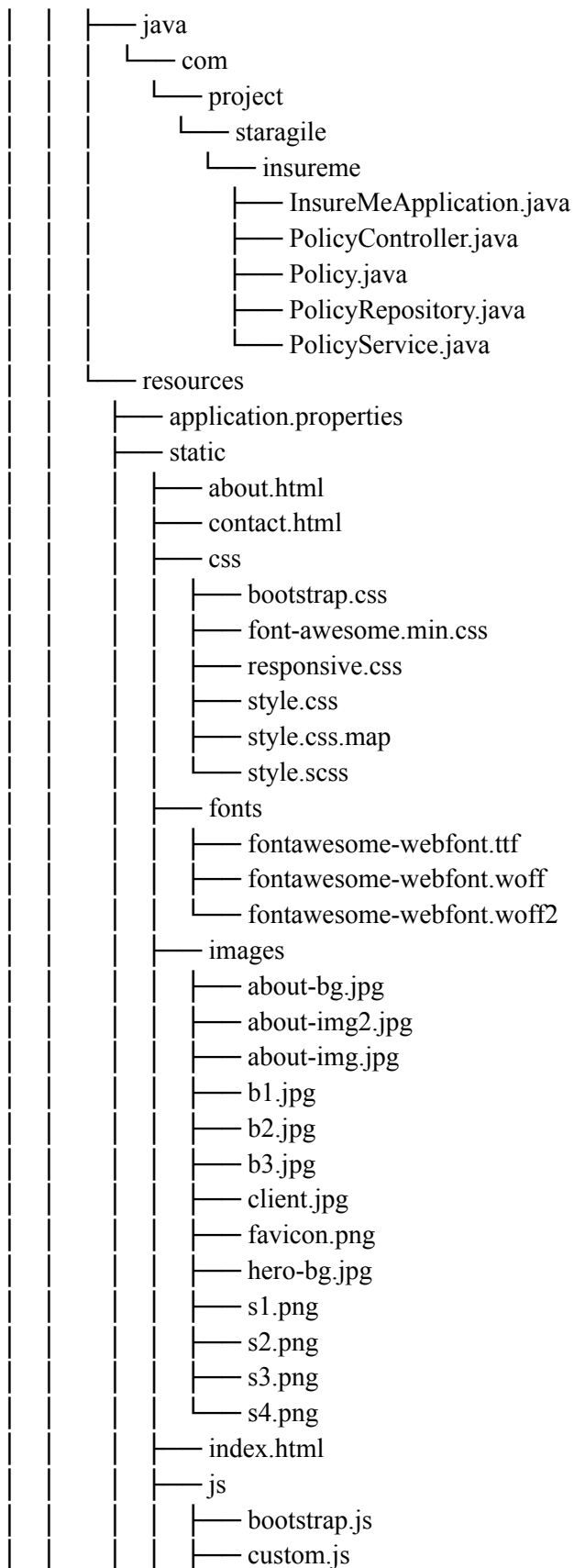
Step 1. Create IAM role and access keys and secret access keys

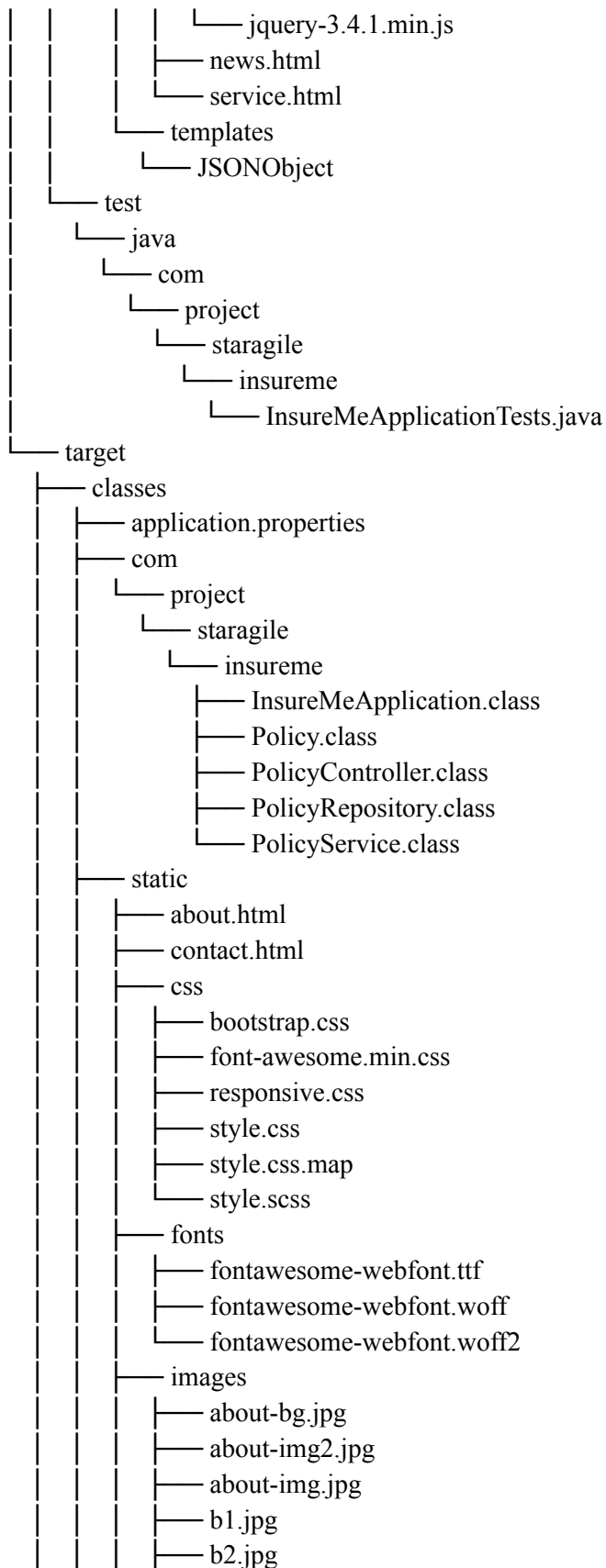
Create a IAM user as terraform

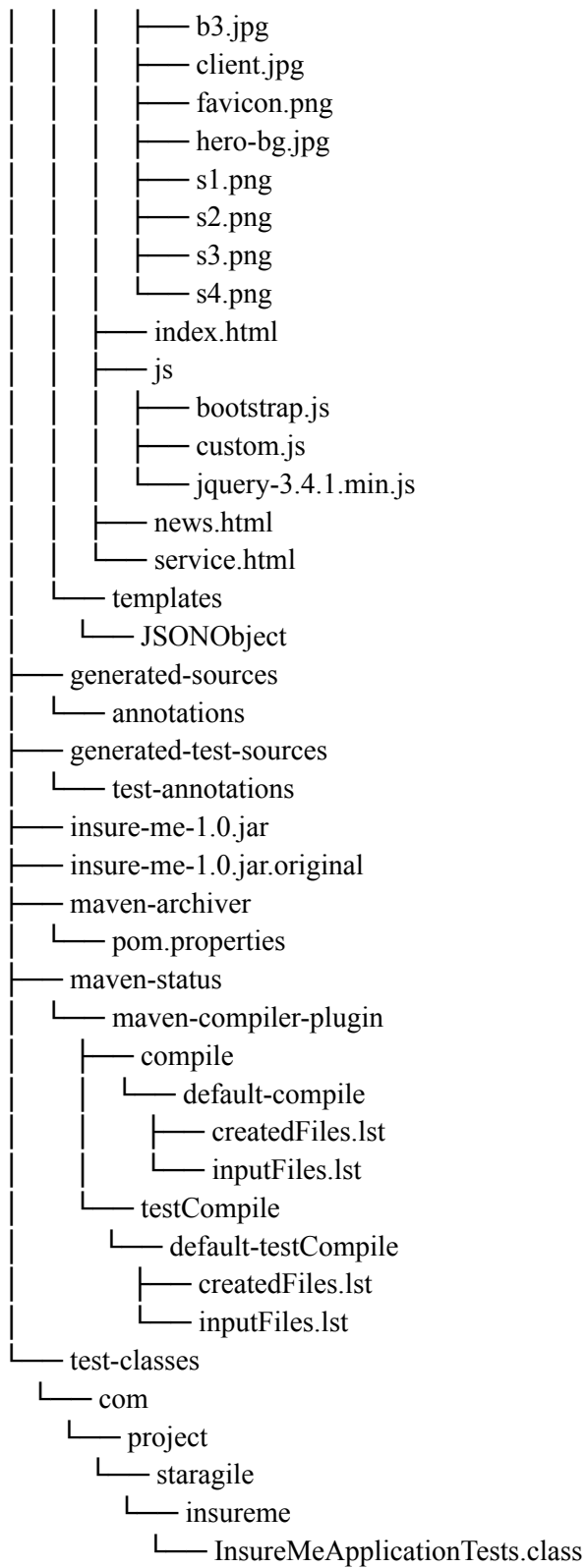
Structure of project:

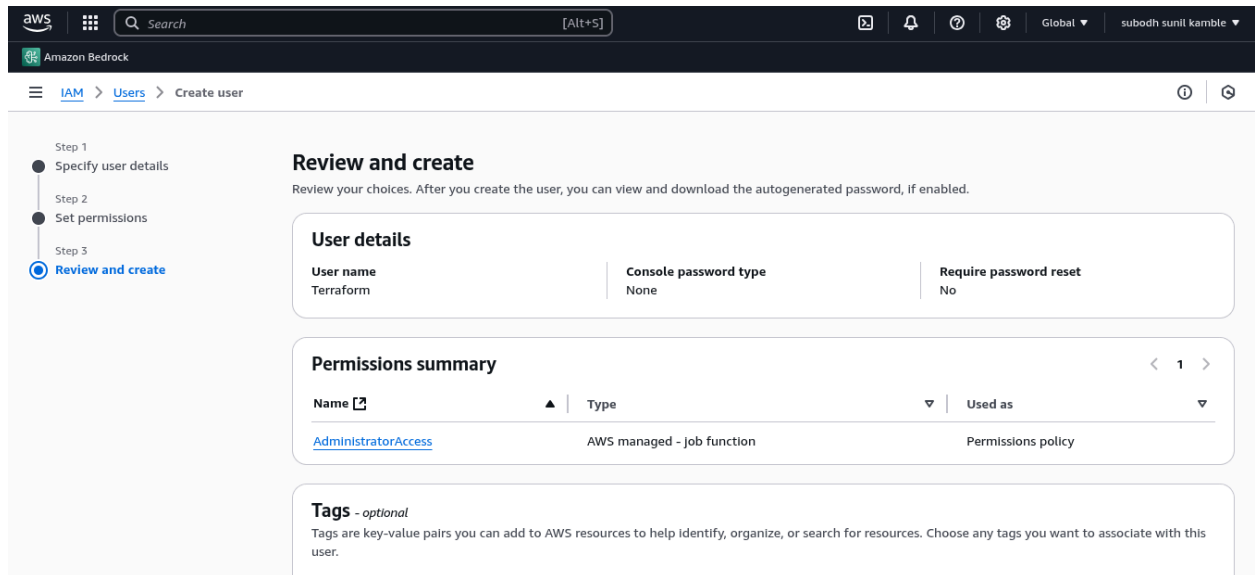
\$ tree .

```
insurance_capstone/
├── infrastructure
│   ├── ansible
│   │   ├── ansible-playbook.yml
│   │   ├── docker-compose-playbook.yml
│   │   ├── docker.sh
│   │   ├── grafana.sh
│   │   ├── inventory
│   │   ├── k8s-master.sh
│   │   ├── k8s-nodes.sh
│   │   ├── test-deployment.yml
│   │   └── trivy.sh
│   ├── infra.sh
│   ├── plan.md
│   ├── README.md
│   └── terraform
│       ├── insurance-key.pem
│       ├── inventory.tmpl
│       ├── main.tf
│       ├── outputs.tf
│       ├── security.tf
│       ├── terraform.tfstate
│       ├── terraform.tfstate.backup
│       ├── terraform.tfvars
│       ├── tls.tf
│       └── variables.tf
├── Insurance Domain Project - Insure Me.pdf
├── LICENSE
├── README.md
└── star-agile-insurance-project
    ├── deployment.yml
    ├── docker-compose.yml
    ├── Dockerfile
    ├── Jenkinsfile
    ├── mvnw
    ├── mvnw.cmd
    ├── pom.xml
    ├── README.md
    ├── selenium-insure-me-runnable.jar
    ├── service.yml
    ├── src
    │   └── main
```

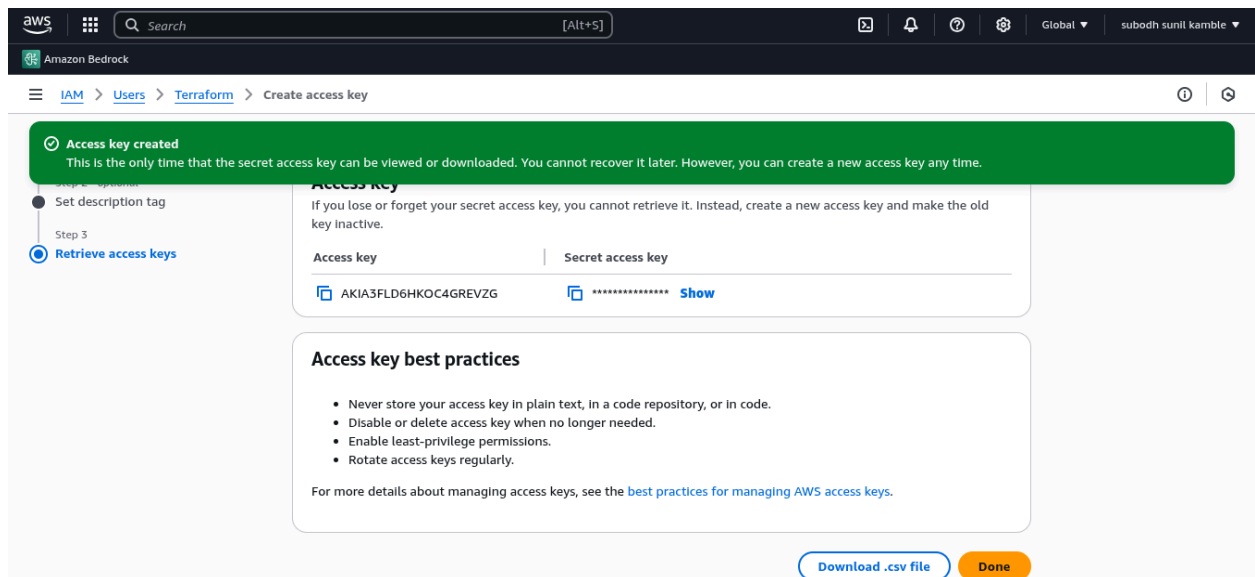








Create access keys and secret access keys



Step 2. Setup infrastructure

```
$ ls -ll
```

```
$ tree . infrastructure/
```

```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone

subodh@Subodh:~/Documents/staragile/projects/insurance_capstone$ ls -ll
total 248
drwxrwxr-x 4 subodh subodh 4096 May 21 10:57 infrastructure
-rw-rw-r-- 1 subodh subodh 233662 May 21 09:48 'Insurance Domain Project - Insure Me.pdf'
-rw-rw-r-- 1 subodh subodh 1070 May 21 11:15 LICENSE
-rw-rw-r-- 1 subodh subodh 165 May 21 11:15 README.md
drwxrwxr-x 5 subodh subodh 4096 May 21 11:14 star-agile-insurance-project
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone$ tree infrastructure/
infrastructure/
├── ansible
│   ├── ansible-playbook.yml
│   └── test-deployment.yml
├── infra.sh
├── plan.md
├── README.md
└── terraform
    ├── inventory.tpl
    ├── main.tf
    ├── outputs.tf
    ├── security.tf
    ├── terraform.tfvars
    └── variables.tf

3 directories, 11 files
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone$
```

\$ terraform init

```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure/terraform

subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/local...
- Installing hashicorp/aws v5.98.0...
- Installed hashicorp/aws v5.98.0 (signed by HashiCorp)
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/terraform$
```

\$ terraform apply -auto-approve

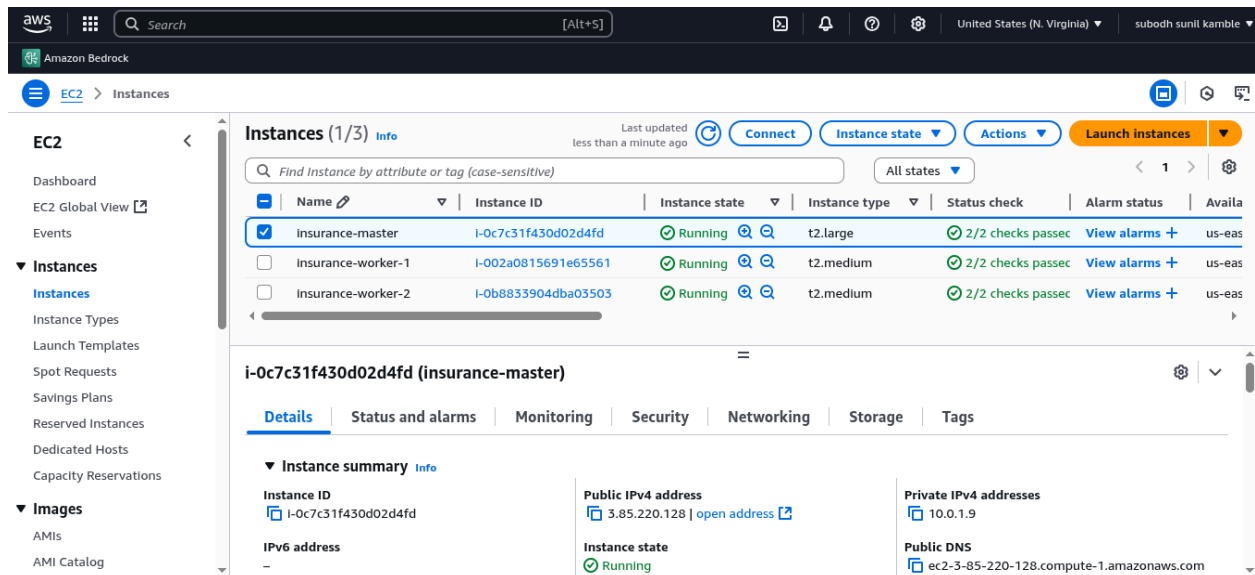
```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure/terraform
aws_instance.master: Creating...
aws_instance.workers[0]: Still creating... [10s elapsed]
aws_instance.workers[1]: Still creating... [10s elapsed]
aws_instance.master: Still creating... [10s elapsed]
aws_instance.master: Creation complete after 15s [id=i-0a4e451919d8ddaa0]
aws_instance.workers[1]: Creation complete after 15s [id=i-04d6c24f56998975e]
aws_instance.workers[0]: Creation complete after 16s [id=i-0feda473dd91fd015]
local_file.ansible_inventory: Creating...
local_file.ansible_inventory: Creation complete after 0s [id=749b4f06d17e513c56405fd1c50ed4cfb09e4bea]

Apply complete! Resources: 6 added, 0 changed, 5 destroyed.

Outputs:

master_private_ip = "10.0.1.247"
master_public_ip = "54.227.233.146"
worker_private_ips = [
  "10.0.1.214",
  "10.0.1.136",
]
worker_public_ips = [
  "54.198.8.233",
  "54.166.170.86",
]
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure/terraform$
```

Successful execution of all the terraform script



Ansible playbooks files


```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure$ tree ansible/
ansible/
├── ansible-playbook.yml
├── inventory
└── test-deployment.yml

1 directory, 3 files
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure$
```

NOTE: if ansible error for UTF-8 will be there follow following commands

\$ export LANG=en_IN.UTF-8

\$ export LANGUAGE=en_IN:en

\$ export LC_ALL=en_IN.UTF-8

```
subodh@Subodh: ~$ locale
LANG=en_IN
LANGUAGE=en_IN:en
LC_CTYPE="en_IN"
LC_NUMERIC="en_IN"
LC_TIME="en_IN"
LC_COLLATE="en_IN"
LC_MONETARY="en_IN"
LC_MESSAGES="en_IN"
LC_PAPER="en_IN"
LC_NAME="en_IN"
LC_ADDRESS="en_IN"
LC_TELEPHONE="en_IN"
LC_MEASUREMENT="en_IN"
LC_IDENTIFICATION="en_IN"
LC_ALL=
subodh@Subodh: ~$ export LANG=en_IN.UTF-8
subodh@Subodh: ~$ export LANGUAGE=en_IN:en
subodh@Subodh: ~$ export LC_ALL=en_IN.UTF-8
subodh@Subodh: ~$
```

To check ansible version :

\$ ansible --version

```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible
Documents/ Downloads/
subodh@Subodh:~$ cd Documents/staragile/projects/insurance_capstone/infrastructure/ansible/
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible$
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible$ ansible --
version
ansible [core 2.19.0b4]
  config file = None
  configured module search path = ['/home/subodh/.ansible/plugins/modules', '/usr/share/ansible/pl
ugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/subodh/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.13.3 (main, Apr 10 2025, 21:38:51) [GCC 14.2.0] (/usr/bin/python3)
  jinja version = 3.1.6
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible$
```

Ansible setup for keys

```
$ mkdir -p ~/.ssh && chmod 700 ~/.ssh && cp insurance-key.pem ~/.ssh/insurance-key.pem && chmod
400 ~/.ssh/insurance-key.pem
```

```
$ ansible-playbook -i inventory ansible-playbook.yaml
```

```
subodh@Subodh: ~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible
subodh@Subodh:~/Documents/staragile/projects/insurance_capstone/infrastructure/ansible$ ansible-playbook
-i inventory ansible-playbook.yml

PLAY [Configure k8s Jenkins and Docker on EC2 Instances] *****

TASK [Gathering Facts] *****
ok: [54.198.8.233]
ok: [54.166.170.86]
ok: [54.227.233.146]

TASK [Update apt cache] *****
changed: [54.198.8.233]
changed: [54.166.170.86]
changed: [54.227.233.146]

TASK [Install required packages] *****

```

```

TASK [Gathering Facts] *****
*****
ok: [23.20.184.205]
ok: [54.224.66.37]
ok: [13.217.116.56]

TASK [Install docker-compose] *****
*****
ok: [13.217.116.56]
changed: [23.20.184.205]
changed: [54.224.66.37]

PLAY RECAP *****
*****
13.217.116.56      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    i
gnored=0
23.20.184.205     : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    i
gnored=0
54.224.66.37      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    i
gnored=0

```

Errors need to solve manually in the server

```

aws
Search [Alt+5]
United States (N. Virginia) subodh sunil kamble
Amazon Bedrock

ubuntu@master:~$ kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
master    Ready    control-plane   24m   v1.29.15
worker-1   Ready    <none>        17m   v1.29.15
worker-2   Ready    <none>        17m   v1.29.15
ubuntu@master:~$ java --version
openjdk 11.0.27 2025-04-15
OpenJDK Runtime Environment (build 11.0.27+6-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.27+6-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
ubuntu@master:~$ git --version
git version 2.25.1
ubuntu@master:~$ docker version
Client: Docker Engine - Community
Version: 28.1.1
API version: 1.49
Go version: go1.23.8
Git commit: 4eba377
Built: Fri Apr 18 09:52:18 2025
OS/Arch: linux/amd64
Context: default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.49/v
ersion": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@master:~$

```

i-0e3ea0a60f3d1fb2e (insurance-master)

PublicIPs: 13.217.116.56 PrivateIPs: 10.0.1.5

All the tools and software is installed using Ansible and is checked by printing versions and info

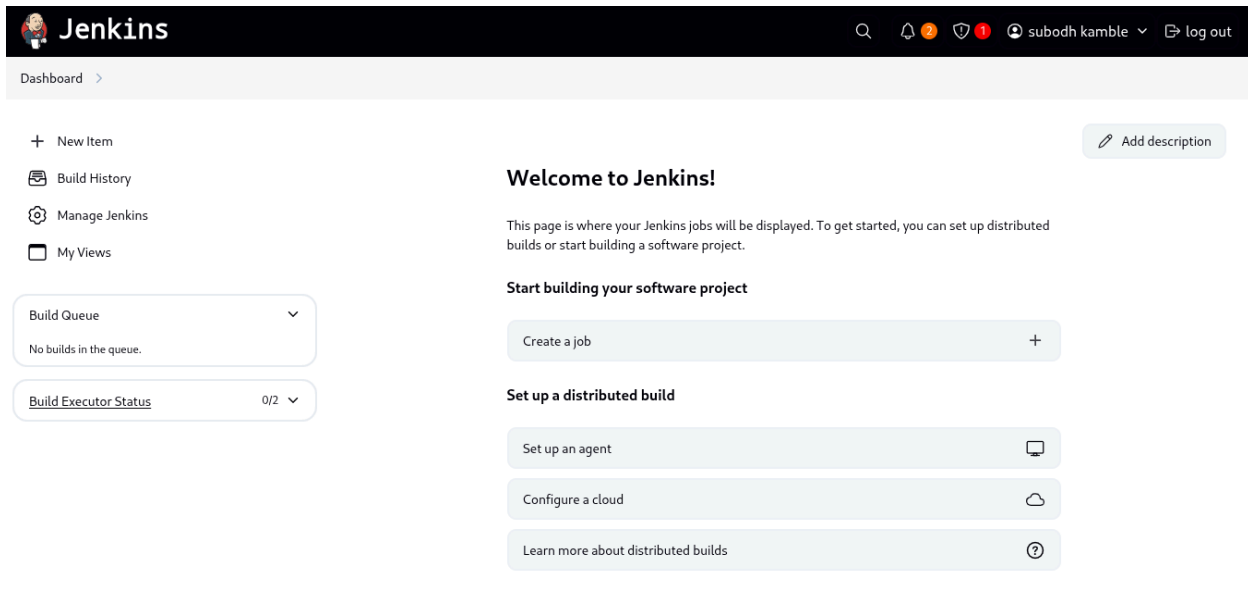
```

$ kubectl --version
$ java --version
$ git --version
$ docker version
$ docker-compose --version

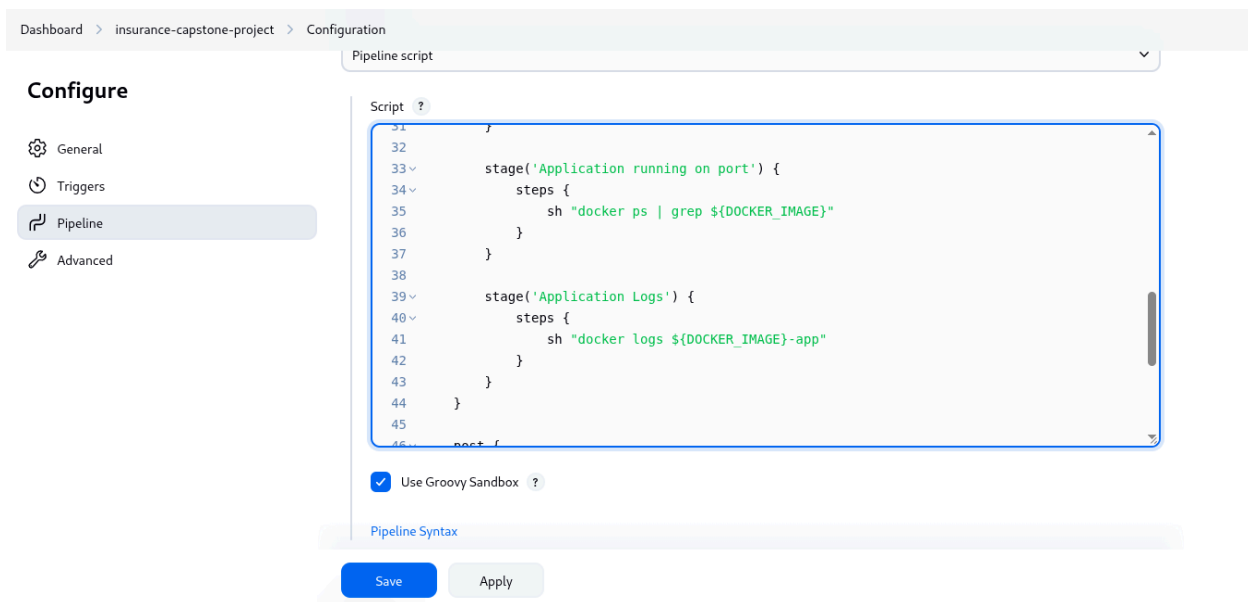
```

Open jenkins on ip : <public ip>:8080

Setup username and other account settings and install suggested plugins



Create a new jenkins pipeline



```

pipeline {
  agent any

  environment {
    DOCKER_IMAGE = 'insureme'
    DOCKER_TAG = 'latest'
    DEPLOY_DIR = 'star-agile-insurance-project'
  }
}

```

```

stages {
    stage('Clone Repository') {
        steps {
            git branch: 'main', url: 'https://github.com/tusuii/insurance_capstone.git'
        }
    }

    stage('Build Application') {
        steps {
            dir("${DEPLOY_DIR}") {
                sh 'mvn clean package'
            }
        }
    }

    stage('Build Docker Image') {
        steps {
            dir("${DEPLOY_DIR}") {
                sh 'docker-compose up --build -d'
            }
        }
    }

    stage('Application running on port') {
        steps {
            sh "docker ps | grep ${DOCKER_IMAGE}"
        }
    }

    stage('Application Logs') {
        steps {
            sh "docker logs ${DOCKER_IMAGE}-app"
        }
    }
}

post {
    success {
        echo "✅ Pipeline completed successfully!"
    }
    failure {
        echo "❌ Pipeline failed!"
    }
    always {

```

```
. _ _ _ _ _  
/\ / _ _ ' _ _ _ _ _ \ \ \ \  
( ) \_ | '_| '_||'_\ \_ | \ \ \ \  
\ \ __) | | | | | | | | | | | ) ) ) )  
' | _ | . _ | | | | | \_, | / / / /  
=====|_|=====|_/=//_/_/_/  
  
:: Spring Boot ::                (v2.7.4)  
  
2025-05-23 08:49:14.543 INFO 1 --- [          main] c.p.s.insureme.InsureMeApplication : Starting InsureMeApplication  
v1.0 using Java 11.0.16 on a075e694df2a with PID 1 (/app.jar started by root in /)  
2025-05-23 08:49:14.570 INFO 1 --- [          main] c.p.s.insureme.InsureMeApplication : No active profile set, falling  
back to 1 default profile: "default"  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Declarative: Post Actions)  
[Pipeline] echo  
Pipeline execution completed.  
[Pipeline] echo  
✅ Pipeline completed successfully!  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

The screenshot shows the Jenkins web interface. At the top, the browser address bar displays '13.217.116.56:8080/job/insurance-capstone-project/'. The Jenkins header includes the logo and the project name 'insurance-capstone-project'. The left sidebar contains navigation links: Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The main content area shows the 'insurance-capstone-project' status as successful (green checkmark). Below this, the 'Stage View' is displayed, showing a table of build stages and their durations for two builds (#16 and #15). Build #16 is successful, while Build #15 is failed (red background). The stages are: Clone Repository, Build Application, Build Docker Image, Application running on port, Application Logs, and Declarative: Post Actions. The table also shows average stage times and full run times. At the bottom, there are 'Permalinks' for the last build and the last stable build.

insurance-capstone-project Edit description

capstone project insurance CI/CD pipeline

Stage View

| | Clone Repository | Build Application | Build Docker Image | Application running on port | Application Logs | Declarative: Post Actions |
|--|------------------|-------------------|--------------------|-----------------------------|------------------|---------------------------|
| Average stage times: (full run time: ~38s) | 479ms | 12s | 10s | 333ms | 258ms | 164ms |
| #16 May 23 14:18 No Changes | 622ms | 12s | 19s | 574ms | 431ms | 198ms |
| #15 May 23 14:12 1 commit | 337ms | 12s | 734ms failed | 92ms failed | 86ms failed | 131ms |

Builds Filter

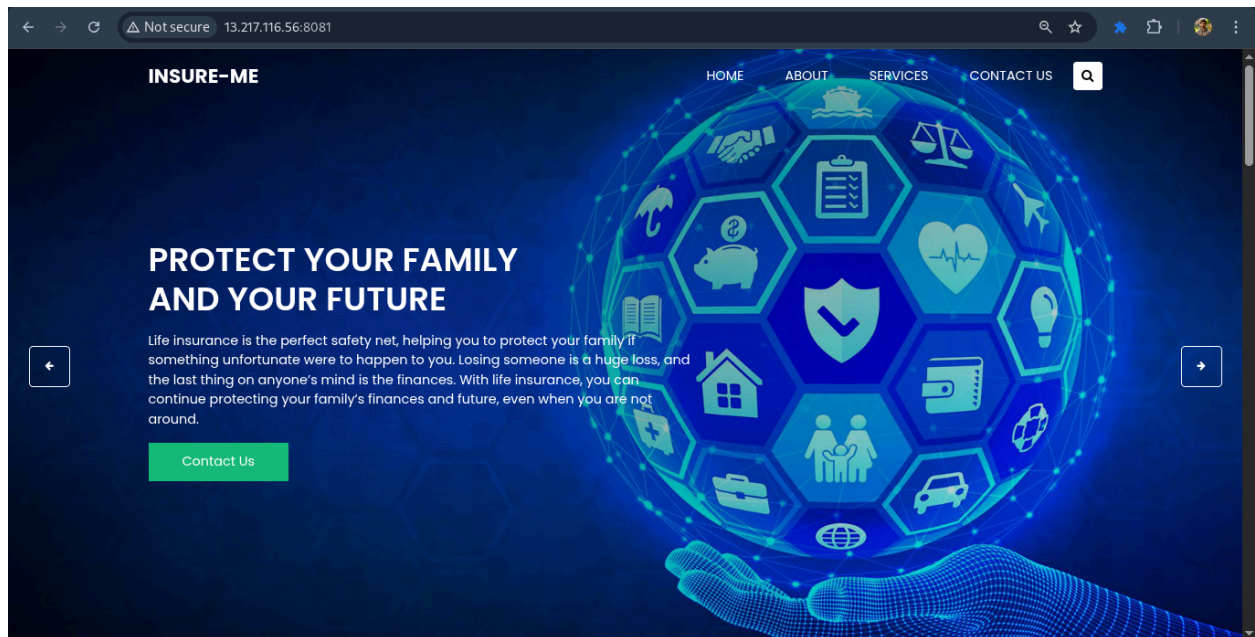
Today

- ✓ #16 8:48 am
- ✗ #15 8:42 am

Permalinks

- Last build (#16), 28 min ago
- Last stable build (#16), 28 min ago

Application running on
<http://13.217.116.56:8081/>



Setup Observability: setup prometheus and grafana
Steps to setup prometheus and grafana

- **Machine 1 (Docker-server / prometheus-machine)**

Responsibilities:

- Install Docker
- Run Ubuntu containers
- Expose Docker metrics
- Install and run Prometheus

- **Machine 2 (Grafana-monitoring-machine)**

Responsibilities:

- Install and run Grafana
- Connect to Prometheus as a data source
- Visualize Docker container metrics

Common Setup on Both Machines

Step 1: Switch to Superuser and Update

```
bash
CopyEdit
sudo su
apt update
```

Grafana Monitoring Machine Setup

Step 2: Download and Install Grafana

```
bash
CopyEdit
wget https://dl.grafana.com/enterprise/release/grafana-enterprise-8.4.4.linux-amd64.tar.gz
ls
tar -zxvf grafana-enterprise-8.4.4.linux-amd64.tar.gz
cd grafana-8.4.4
./bin/grafana-server
```

Note: Grafana by default runs on port 3000.

Ensure **port 3000 is open** in your instance's **Security Group** or firewall rules.

Step 3: Access Grafana in Browser

```
text
CopyEdit
http://<Grafana-machine-public-ip>:3000
```

- **Username:** admin
- **Password:** admin

Docker + Prometheus Machine Setup

Step 4: Install Docker and Run Containers

```
bash
CopyEdit
apt install docker.io -y
service docker start
```



```
docker run -dt --name c01 ubuntu
docker run -dt --name c02 ubuntu
```

Step 5: Enable Docker Metrics on Port 9323

Edit Docker Daemon Configuration

```
bash
CopyEdit
vi /etc/docker/daemon.json
```

Add the following JSON content:

```
json
CopyEdit
{
  "metrics-addr" : "0.0.0.0:9323",
  "experimental" : true
}
```

Restart Docker Service

```
bash
CopyEdit
service docker restart
```

Step 6: Install Prometheus

```
bash
CopyEdit
wget
https://github.com/prometheus/prometheus/releases/download/v2.34.0/prometheus-2.34.0.linux-amd64.tar
.gz
tar zxvf prometheus-2.34.0.linux-amd64.tar.gz
cd prometheus-2.34.0.linux-amd64
```

Open the metrics endpoint in browser or curl:

```
bash
CopyEdit
curl http://localhost:9323/metrics
```

Step 7: Configure Prometheus to Scrape Docker Metrics

Open and Edit Prometheus Configuration

```
bash
CopyEdit
vi prometheus.yml
```

Add the following scrape job:

```
yaml
CopyEdit
- job_name: "docker"
  static_configs:
    - targets: ["localhost:9323"]
```

Save and exit (Esc, then :wq)

Step 8: Start Prometheus

```
bash
CopyEdit
./prometheus
```

Prometheus runs on **port 9090**. Ensure **ports 9090 and 9323 are open** in your firewall/security group.

Access Prometheus:

```
text
CopyEdit
http://<Docker-server-public-ip>:9090
```

Configure Grafana to Use Prometheus

Step 9: Add Prometheus as a Data Source in Grafana

1. Open Grafana:
`http://<Grafana-machine-public-ip>:3000`
2. Go to "**Configuration**" → "**Data Sources**"
3. Click "**Add data source**"
4. Select **Prometheus**

Set the URL:

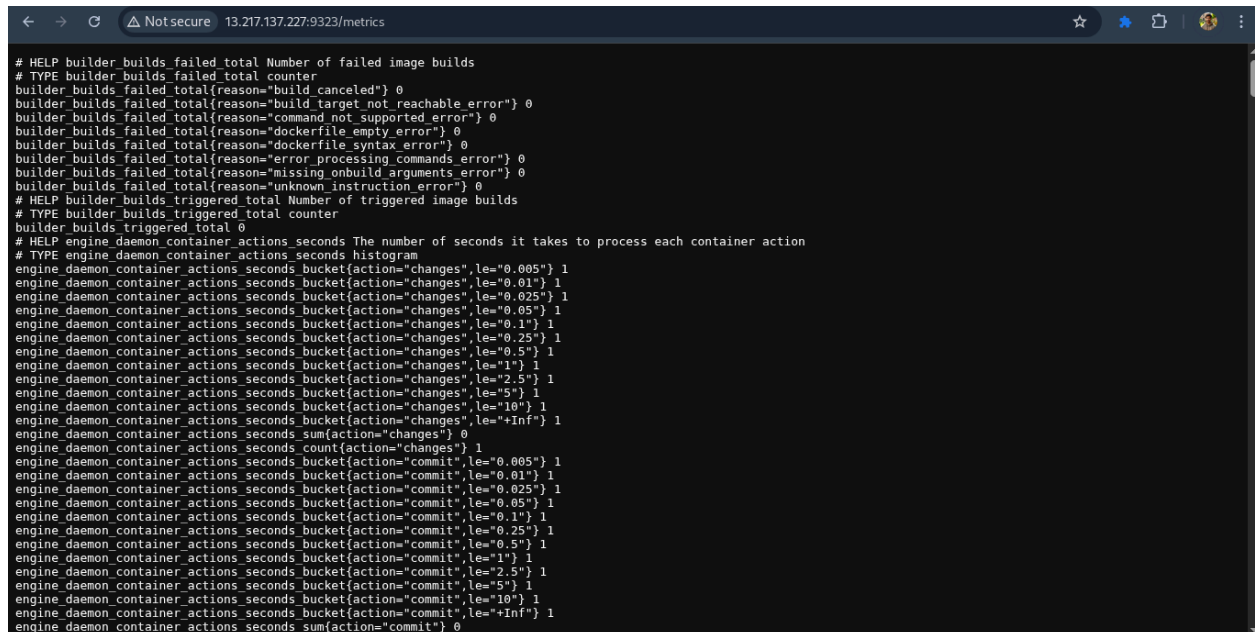
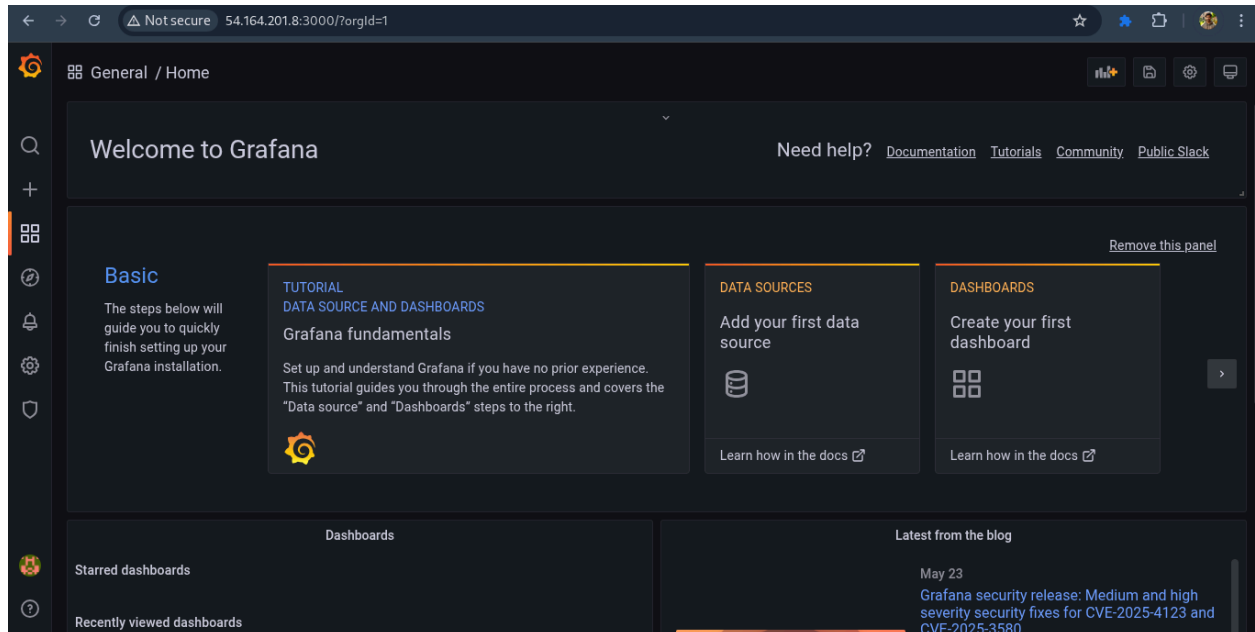
pgsql

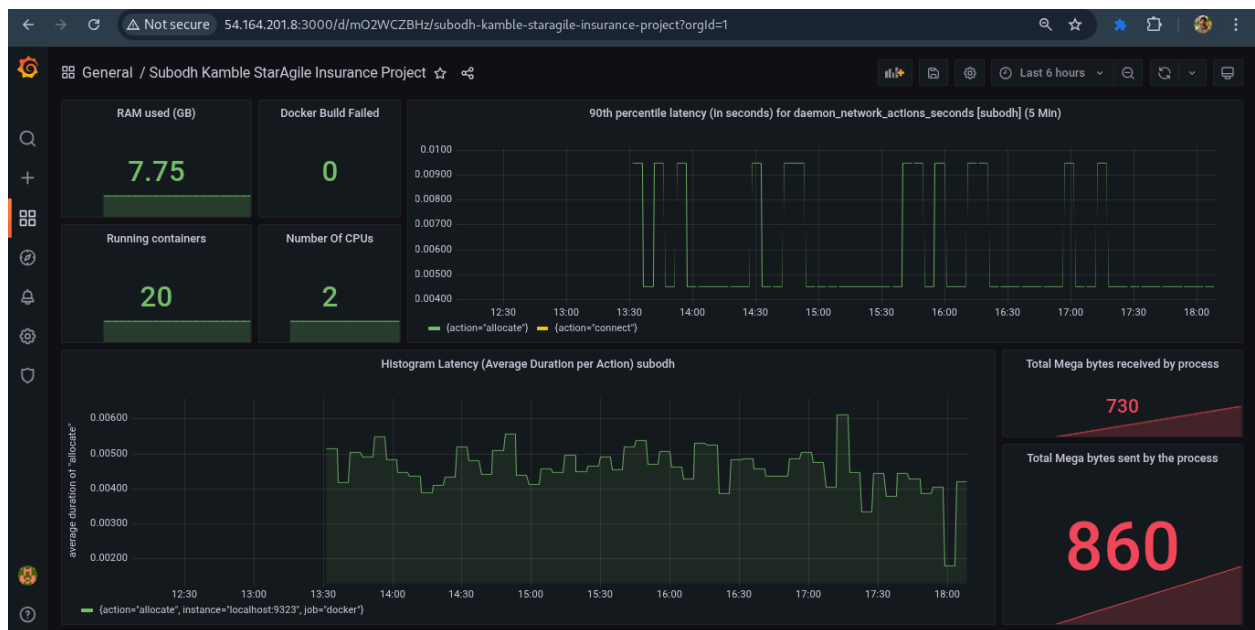
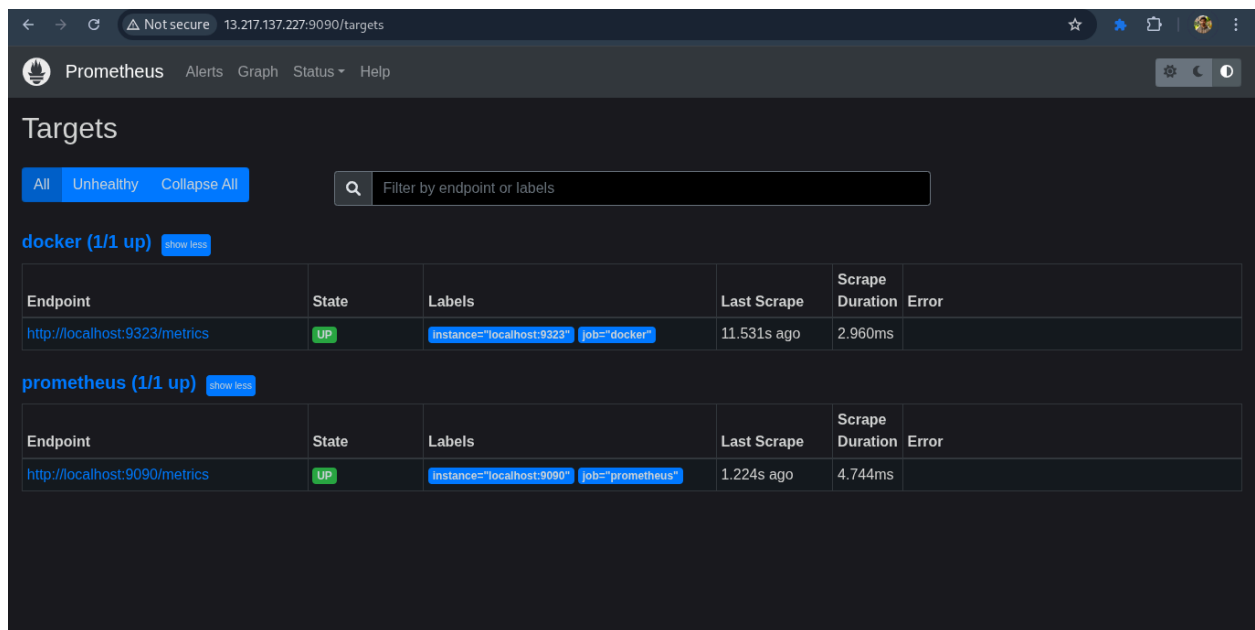
CopyEdit

http://<Docker-server-public-ip>:9090/

5. Click "Save & Test"

You should see a success message.





Final dashboard for the project

Giithub: https://github.com/tusuii/insurance_capstone.git