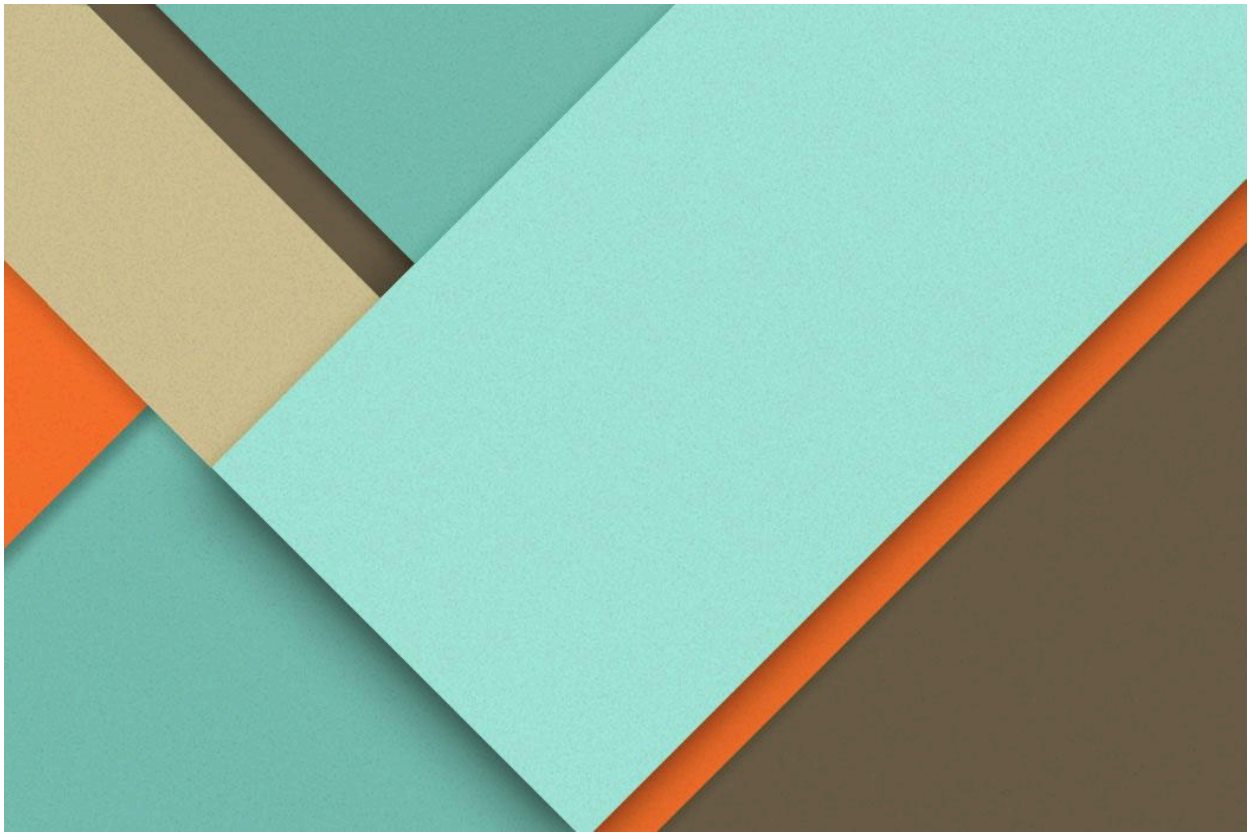


CI/CD Pipeline Project - ASI Insurance

28.02.2024

—

Akshay Imanuel N

Bangalore,

India

Objective

To create a microservice application architecture for an Insurance company through DevOps pipeline and deployment on Docker.

Tools used:

1. Jenkins
2. Github
3. Docker and 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.

Tasks

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 repository and Required Files:

I have created a new github repository, where I have stored the source files of the application.

https://github.com/akshay-imanuel/Dockerised-CICD-pipeline.git

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

2. Install nodejs

To install nodejs use the below command

sudo apt install nodejs -y

```
labsuser@master:~$ sudo apt install nodejs -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libjs-highlight.js libnode72 nodejs-doc
Suggested packages:
  apache2 | lighttpd | httpd npm
The following NEW packages will be installed:
  javascript-common libjs-highlight.js libnode72 nodejs nodejs-doc
0 upgraded, 5 newly installed, 0 to remove and 86 not upgraded.
Need to get 13.7 MB of archives.
```

Verify the installation using

node -v

```
labsuser@master:~$ node -v
v12.22.9
labsuser@master:~$
```

3. Install and Access Jenkins Application

Java is a prerequisite for Jenkins to be installed, hence we will install java first followed by jenkins itself.

sudo apt install openjdk-17-jre-headless

```
labsuser@master:~$ sudo apt install openjdk-17-jre-headless
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java java-common
Suggested packages:
  default-jre fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-mi
The following NEW packages will be installed:
```

Verify the installation

java –version

```
labsuser@master:~$ java --version
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing)
labsuser@master:~$
```

Installing jenkins: Use the code below to install jenkins

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

```
labsuser@master:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Verify the installation:

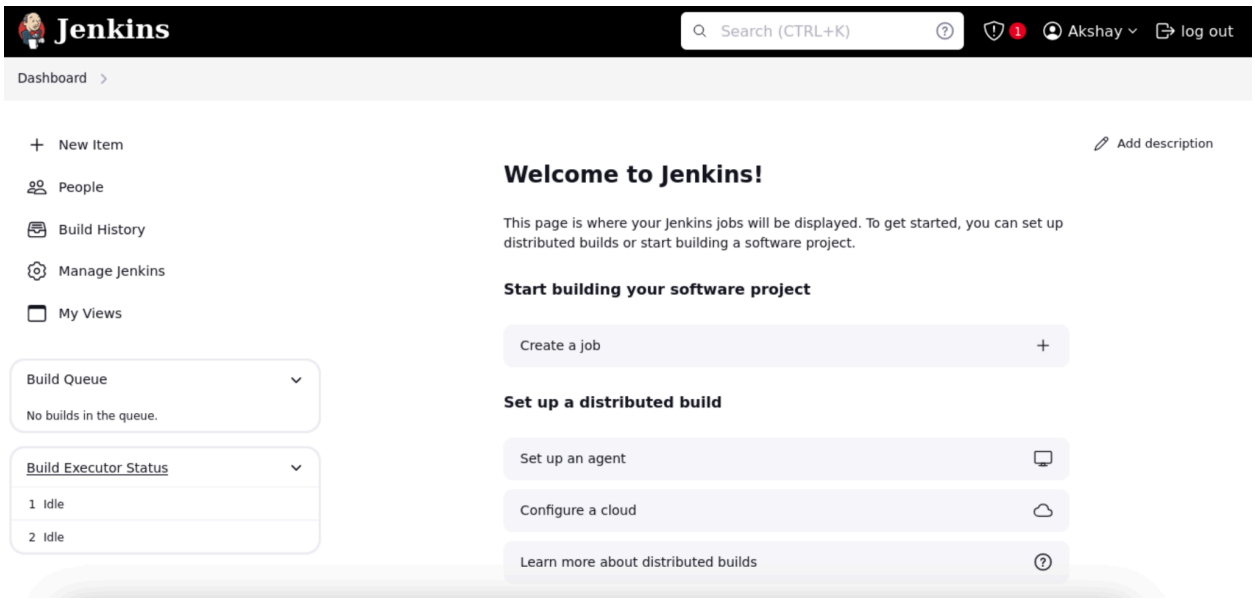
jenkins - -version

Sudo systemctl status jenkins

```
labsuser@master:~$ jenkins --version
2.440.1
labsuser@master:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-02-28 11:18:29 UTC; 4min 13s ago
   Main PID: 16896 (java)
      Tasks: 46 (limit: 4598)
     Memory: 1.1G
        CPU: 1min 1.826s
     CGroup: /system.slice/jenkins.service
             └─16896 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Feb 28 11:17:59 master jenkins[16896]: b5f78d4961f741cea2bdc115833dee6d
Feb 28 11:17:59 master jenkins[16896]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Feb 28 11:17:59 master jenkins[16896]: ************************************************************
Feb 28 11:17:59 master jenkins[16896]: ************************************************************
Feb 28 11:17:59 master jenkins[16896]: ************************************************************
Feb 28 11:18:29 master jenkins[16896]: 2024-02-28 11:18:29.246+0000 [id=32]        INFO        jenkins.InitReactorRunner$1#onAttained: Completed initialization
Feb 28 11:18:29 master jenkins[16896]: 2024-02-28 11:18:29.274+0000 [id=24]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Feb 28 11:18:29 master systemd[1]: Started Jenkins Continuous Integration Server.
Feb 28 11:18:29 master jenkins[16896]: 2024-02-28 11:18:29.556+0000 [id=49]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.M
Feb 28 11:18:29 master jenkins[16896]: 2024-02-28 11:18:29.557+0000 [id=49]        INFO        hudson.util.Retrier#start: Performed the action check updates server successfully at the
lines 1-20/20 (END)
```

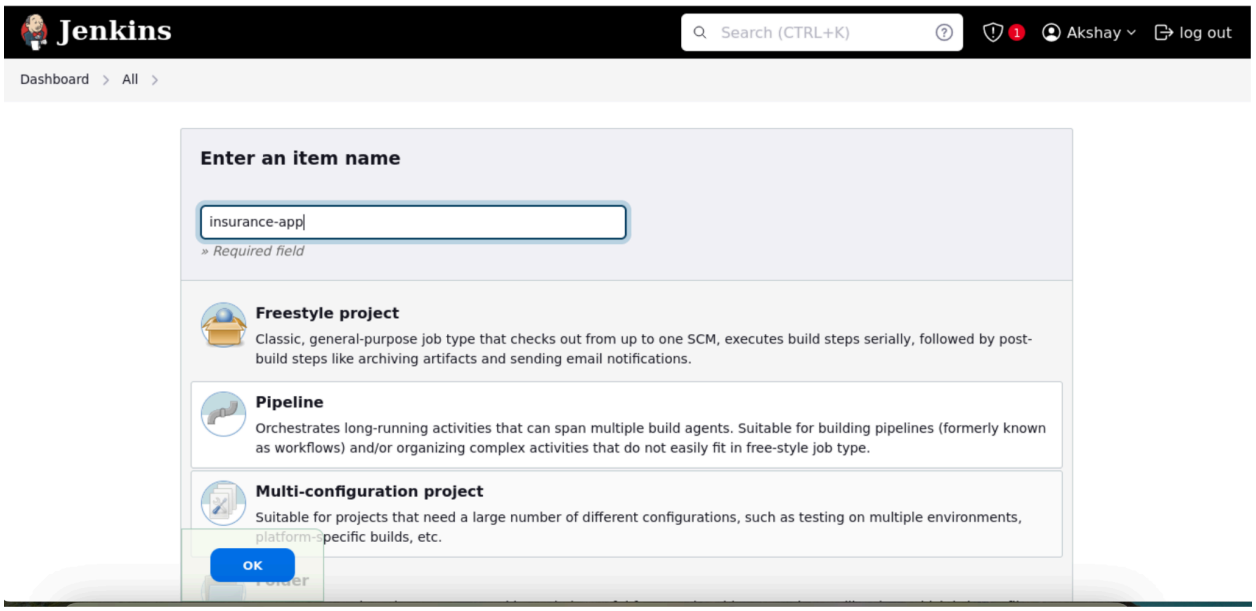
Access jenkins using the public ip and the port 8080, as seen below. And enter the token



4. Create Jenkins Pipeline

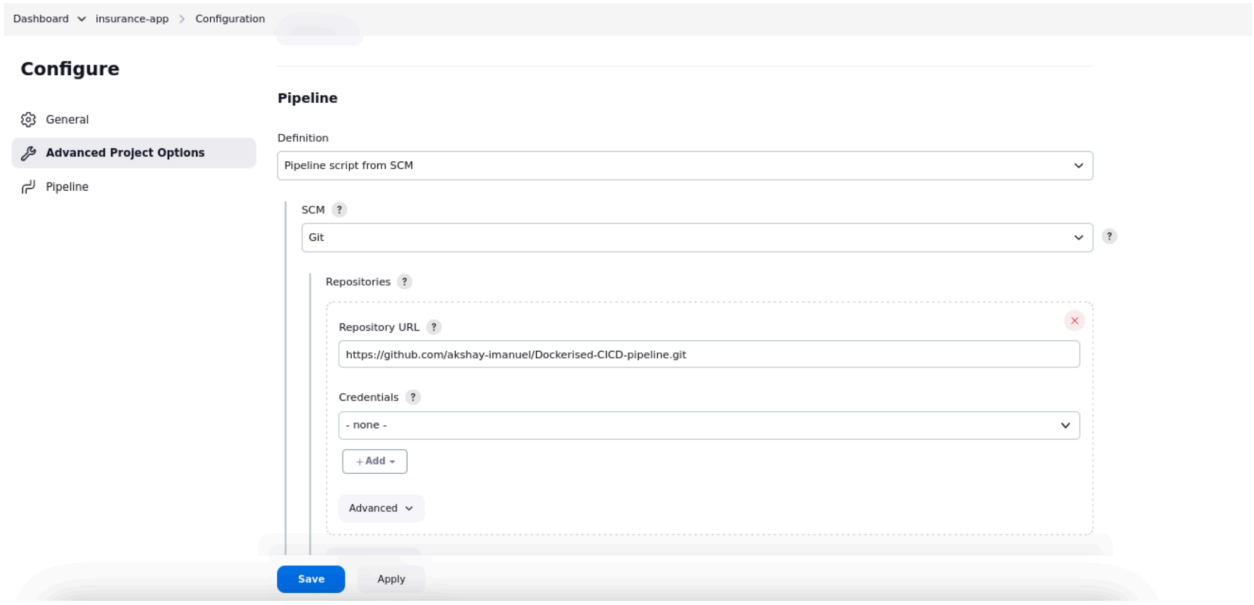
First we will create a test pipeline and we will build each step one by one and in this process we will update all our required files in Github and in the end we will make the necessary changes to run the complete pipeline for the actual app.

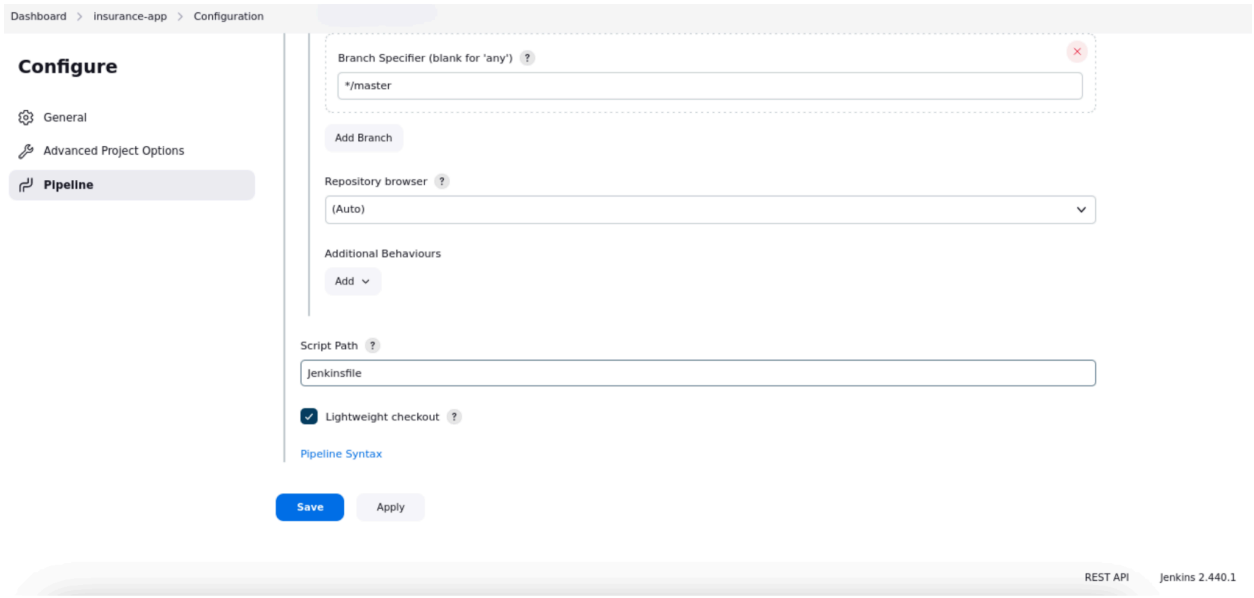
Lets create a pipeline:



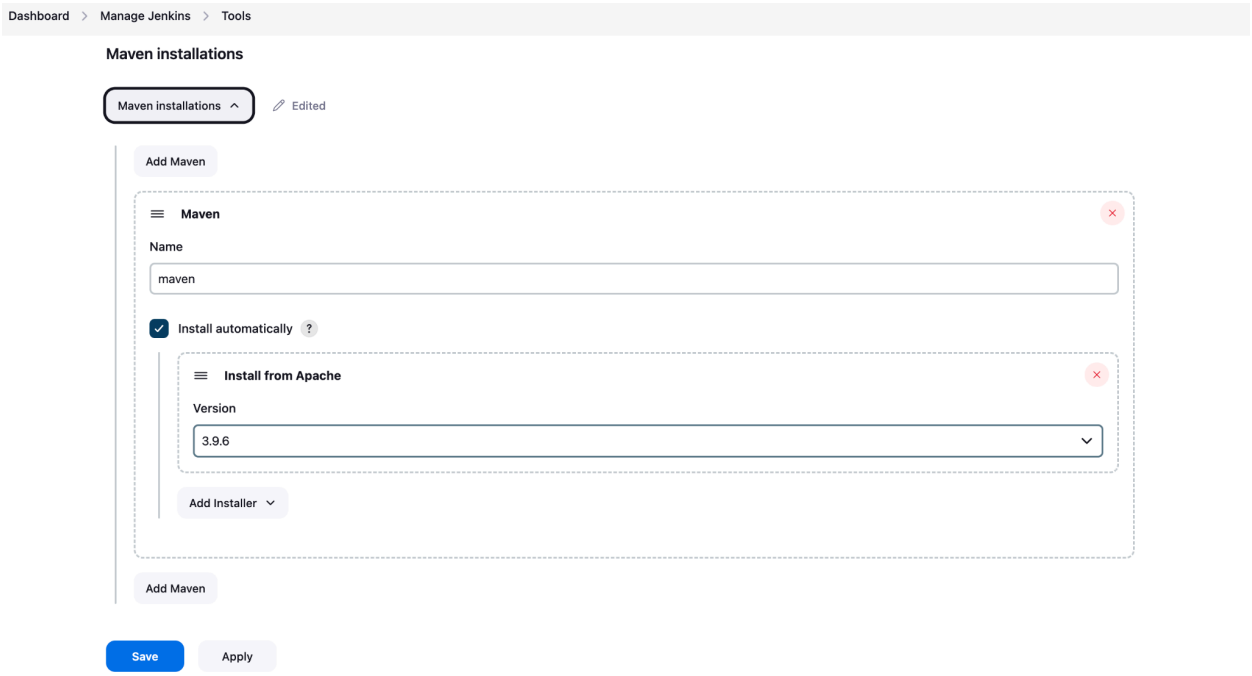
Once Clicked on Ok you will be navigated to Jenkins job configuration page where we can provide Jenkins job details such as parameters, Jenkins pipeline configuration etc.

Enter the github repo link and your default branch, and give the Jenkinsfile name in the repo

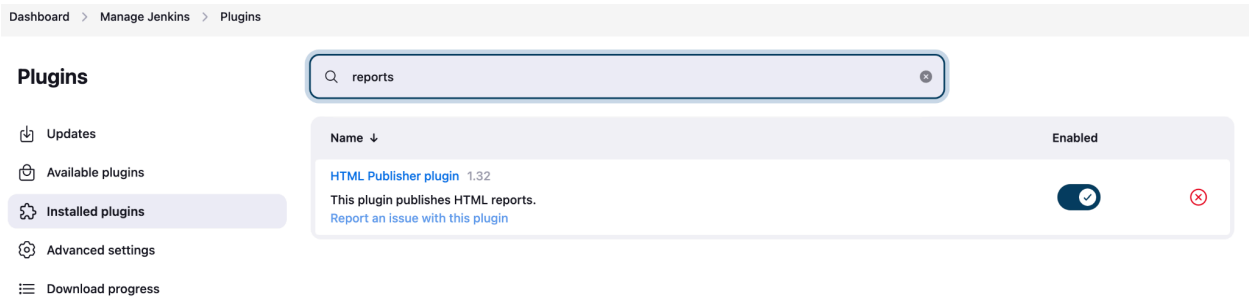

Dashboard
insurance-app
Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/akshay-imanuel/Dockerised-CICD-pipeline.git
Credentials
- none -
+ Add
Advanced
Save
Apply


Dashboard
insurance-app
Configuration
Configure
General
Advanced Project Options
Pipeline
Branch Specifier (blank for 'any')
*/master
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Script Path
Jenkinsfile
Lightweight checkout
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.440.1

1. Add dependency maven tool

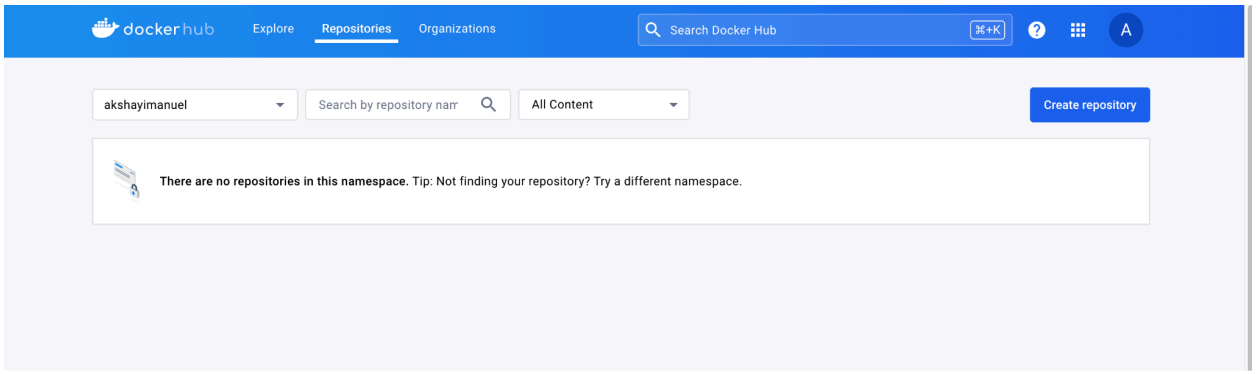


2. Install the reports plugin

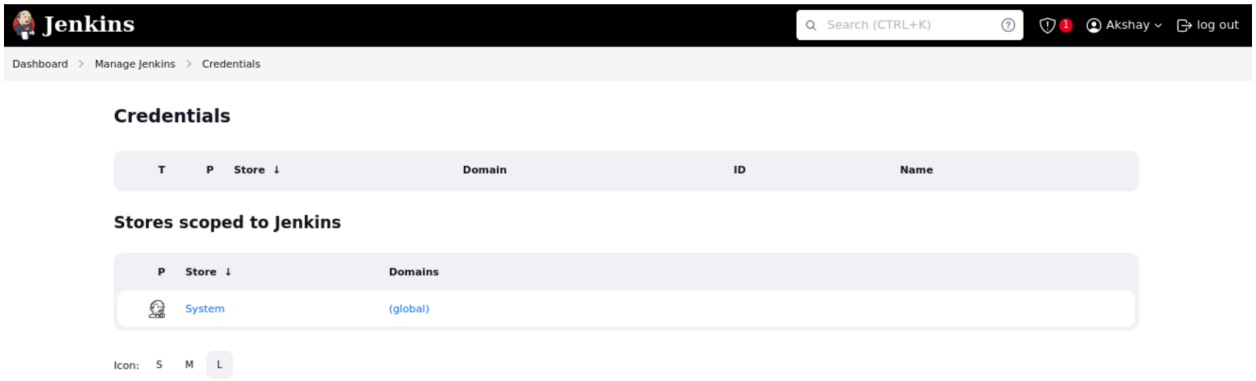


5. Create Docker Hub Credentials and other necessary prerequisites before running build.

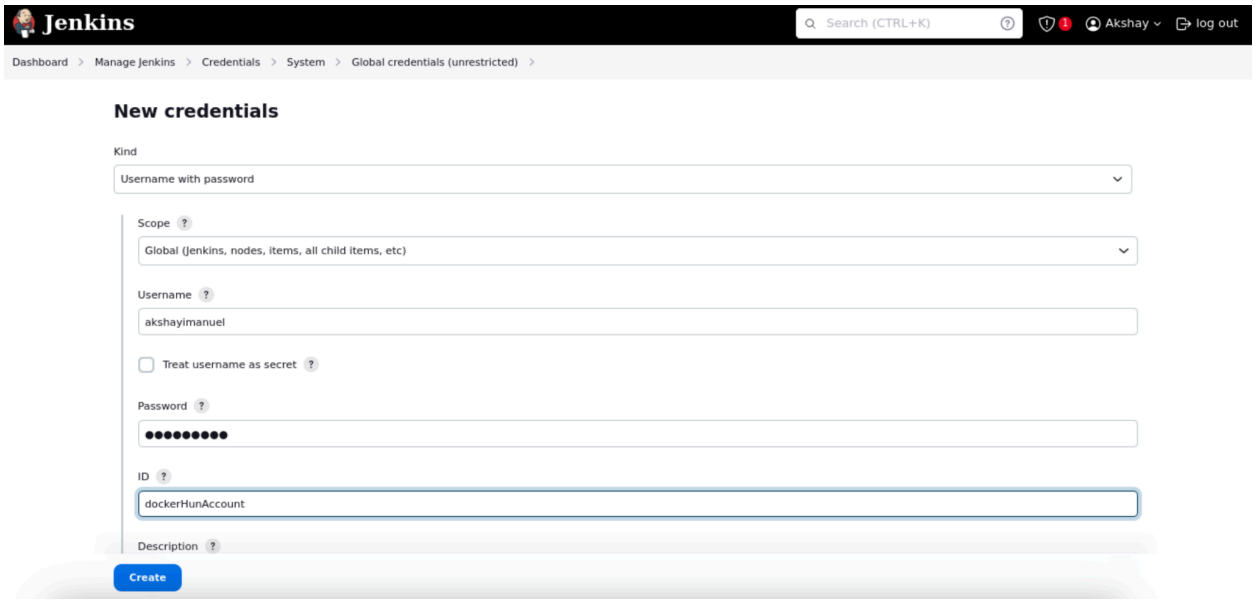
Once Jenkins job configuration are done, we have to create a Docker credential which will be used to push Docker image to Docker hub



Once a Docker hub account is created, navigate to Manage Jenkins à Manage Credentials and then select the global domain.



Click on new Credential by clicking on Add Credentials to create a new Docker hub credentials as per below details:

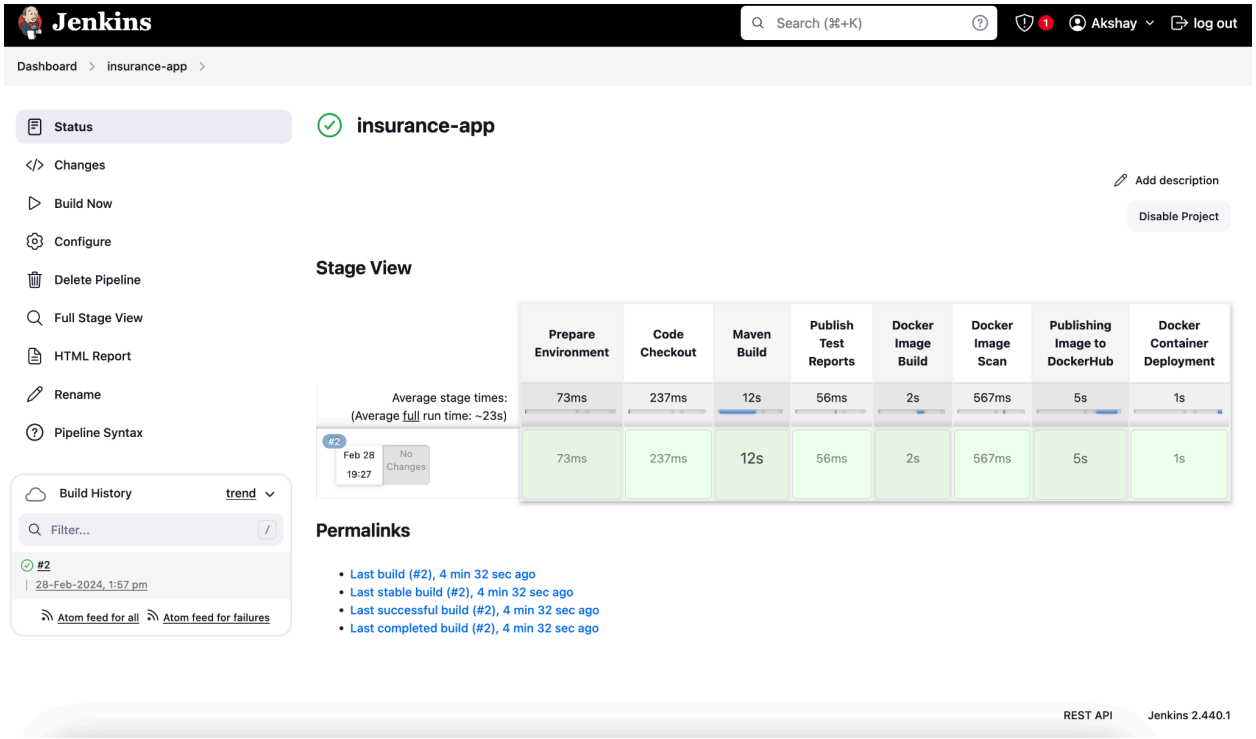


Next provide full access to Docker Sock file using below command:

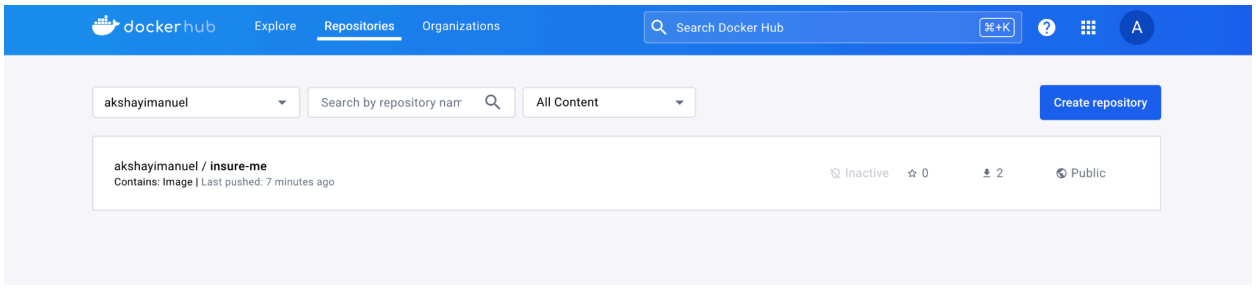
chmod 777 /var/run/docker.sock

6. Execute Jenkins Build

1. Verify Jenkins Build success



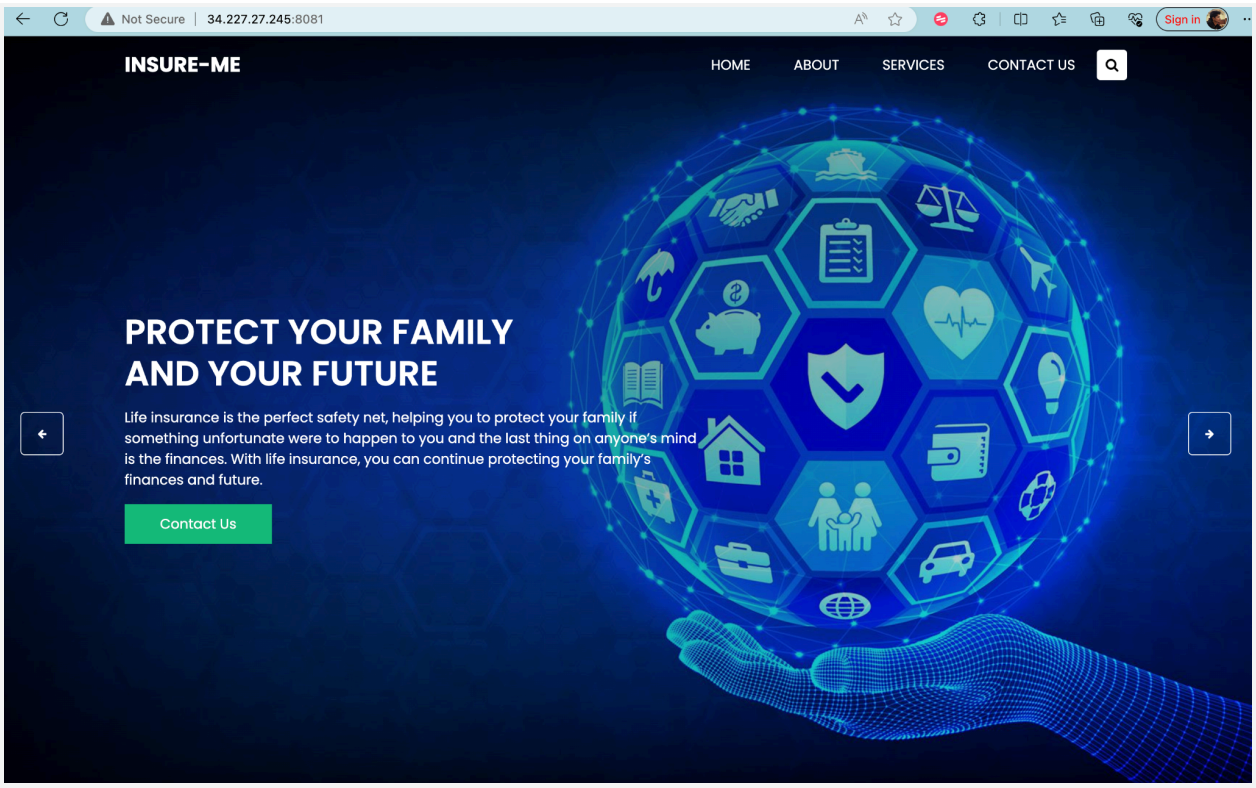
2. Verify the Image push to Docker Hub

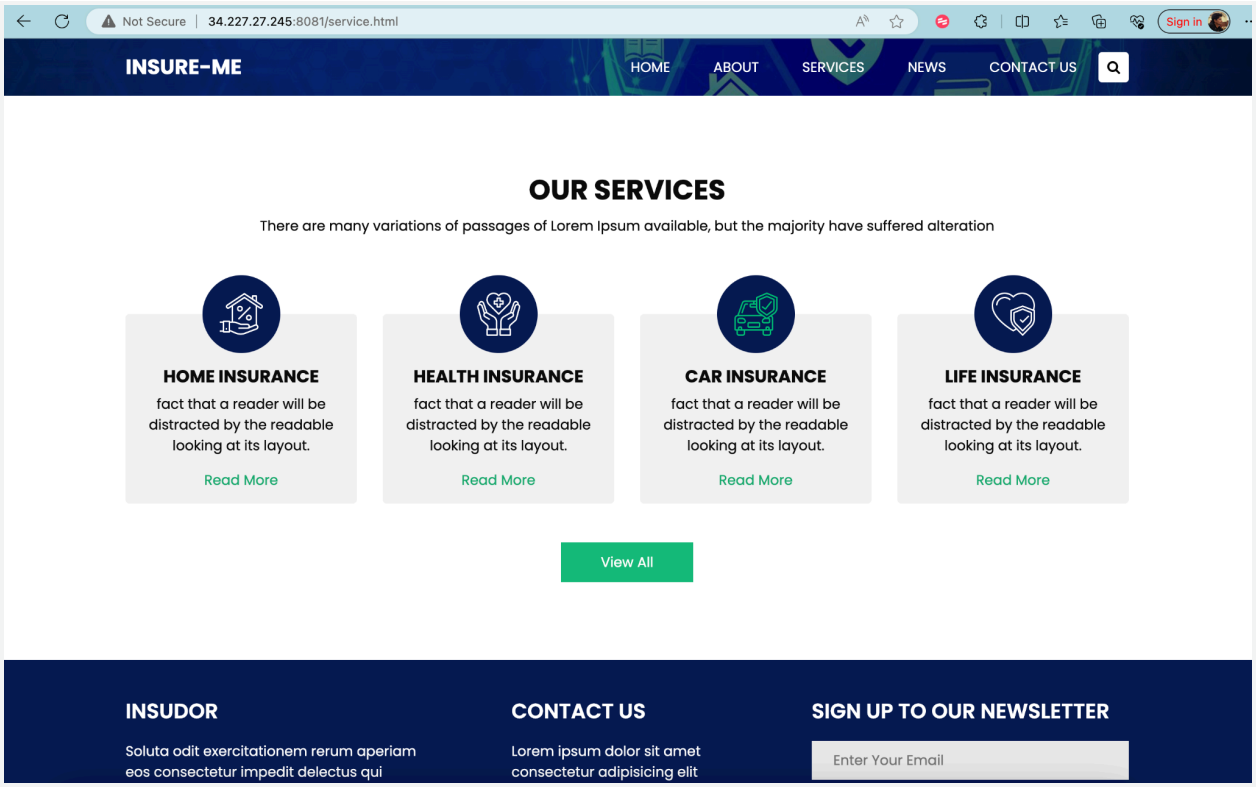


3. Verify the application deployment on the docker container

Once Docker container is deployed using server IP address and port 8081 to access application.

http://34.227.27.245 :8081





Outcome

The successful completion of the project will enable ASI Insurance

- To improve its overall application deployment process.
- Enhance system scalability.
- Deliver better products and services to its customers.