ASI INSURANCE

**Objectives:**

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

**Problem Statement and Motivation**

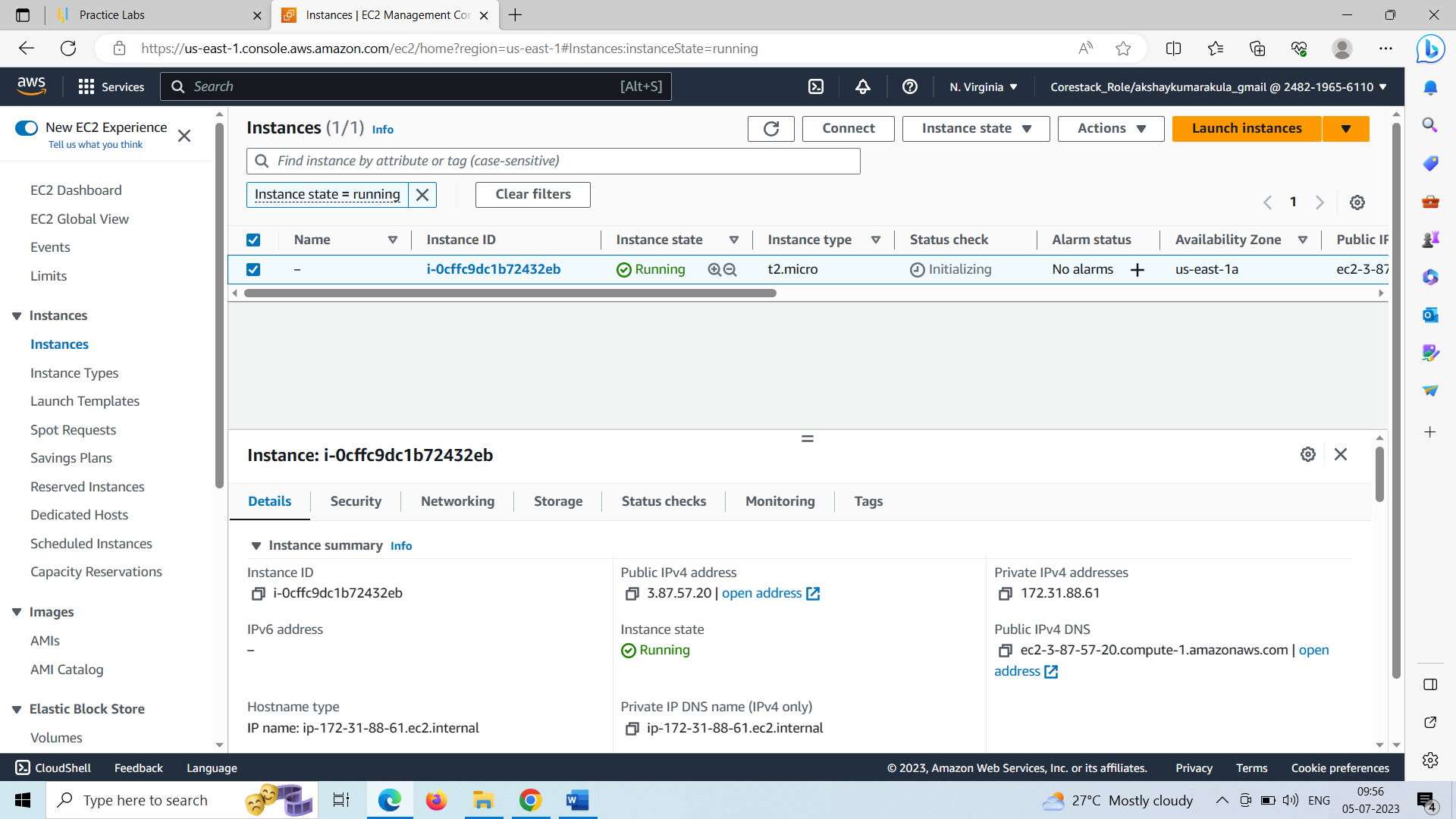
ASI Insurance is facing challenges in improving the SLA to its customers due to its organizational growth and existing monolithic application architecture. It requires transformation of the existing architecture to a microservice application architecture, while also implementing DevOps pipeline and automations.

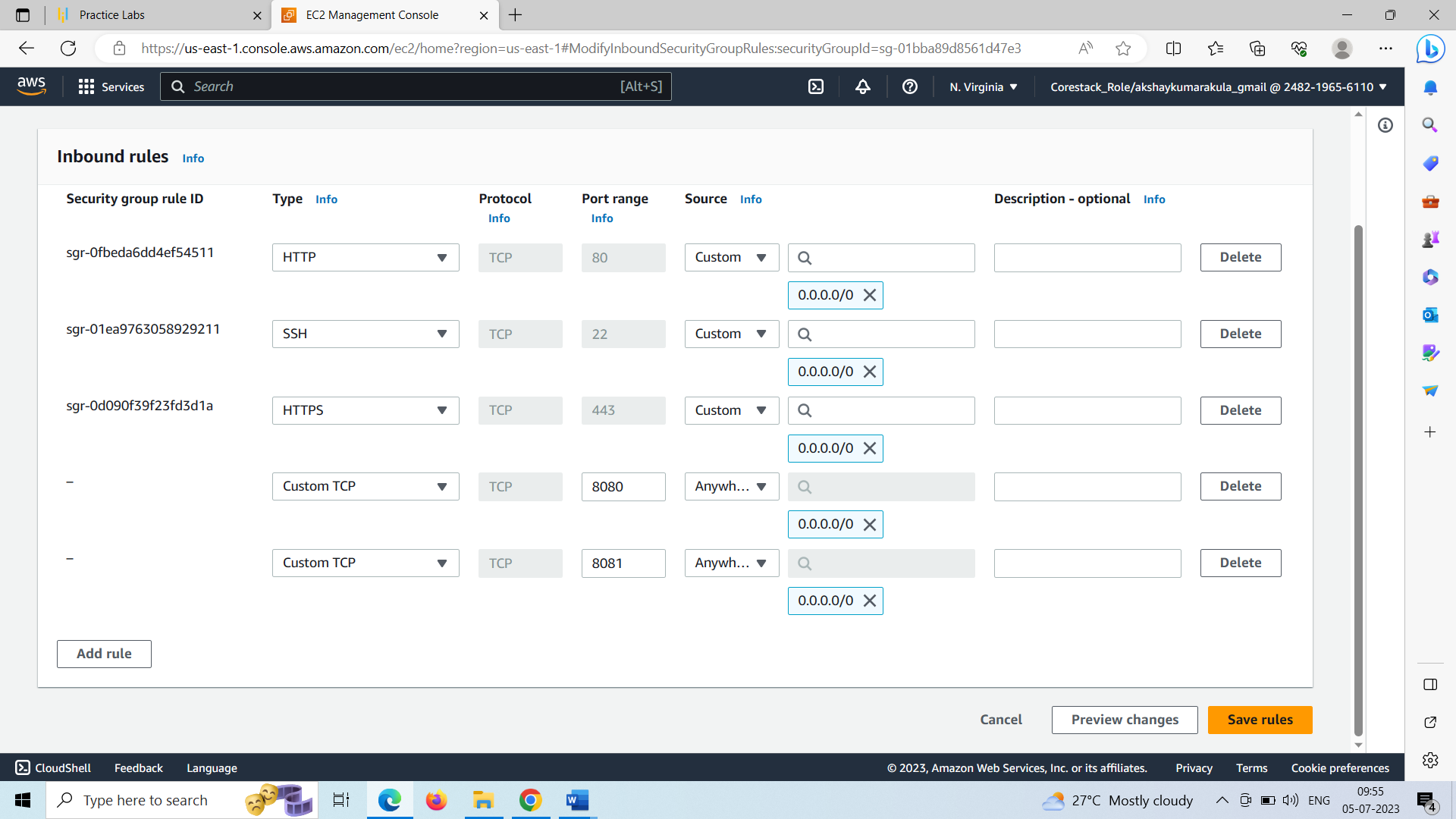
The successful completion of the project will enable ASI Insurance to improve its overall application deployment process, enhance system scalability, and deliver better products and services to its customers.

**Task (Activities):**

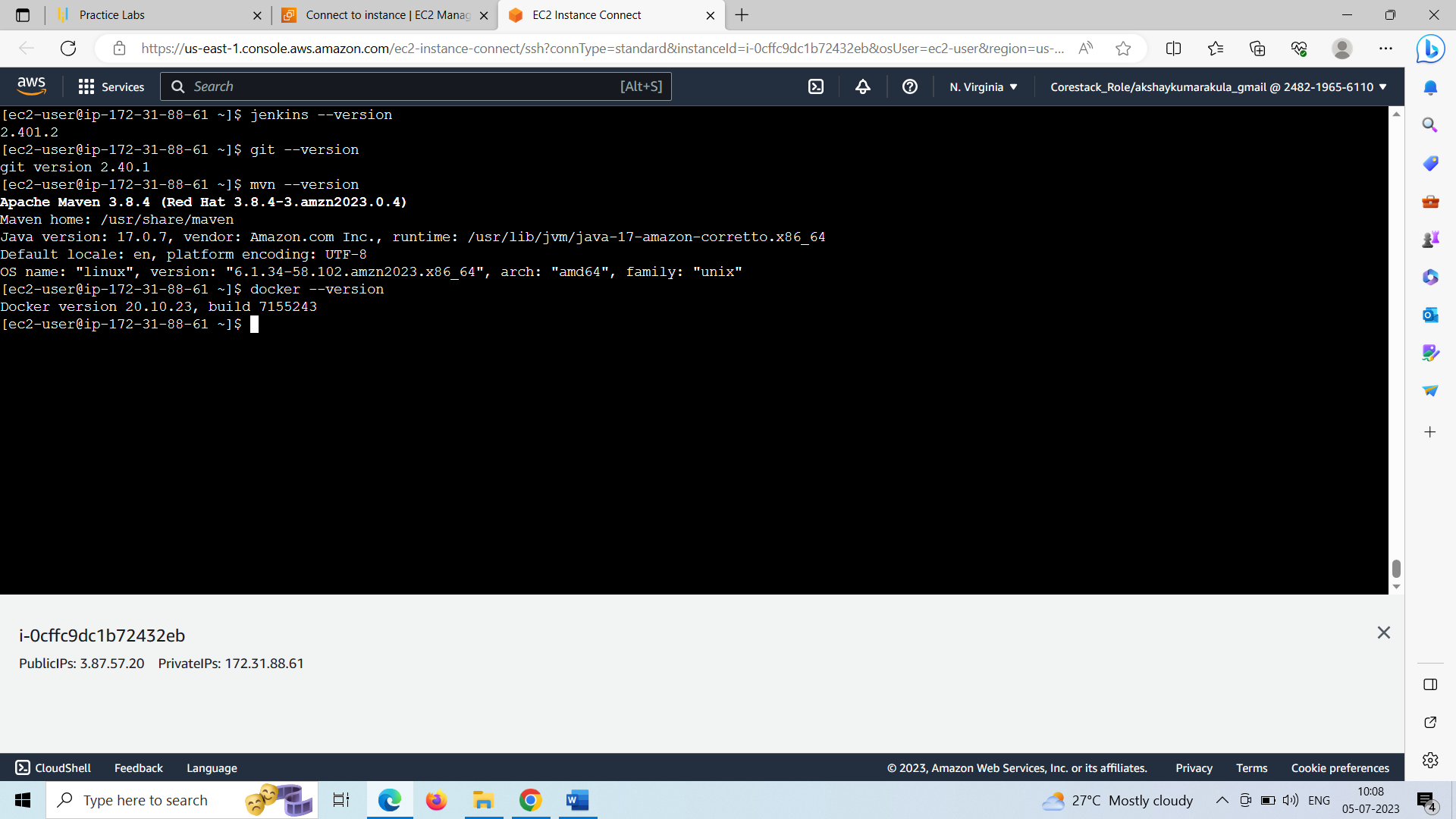
1. Create the Dockerfile, Jenkinsfile, Ansible playbook, and the source file of the static website
2. Upload all the created files to GitHub
3. Go to the terminal and install NodeJS 16
4. Open the browser and access the Jenkins application
5. Create Jenkins pipeline to perform CI/CD for a Docker container
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

**Step – 1:** Create an ec2 instance on the aws and edit inbound security rules to enable port 8080 and 8081 to run Jenkins and the static website on public ip

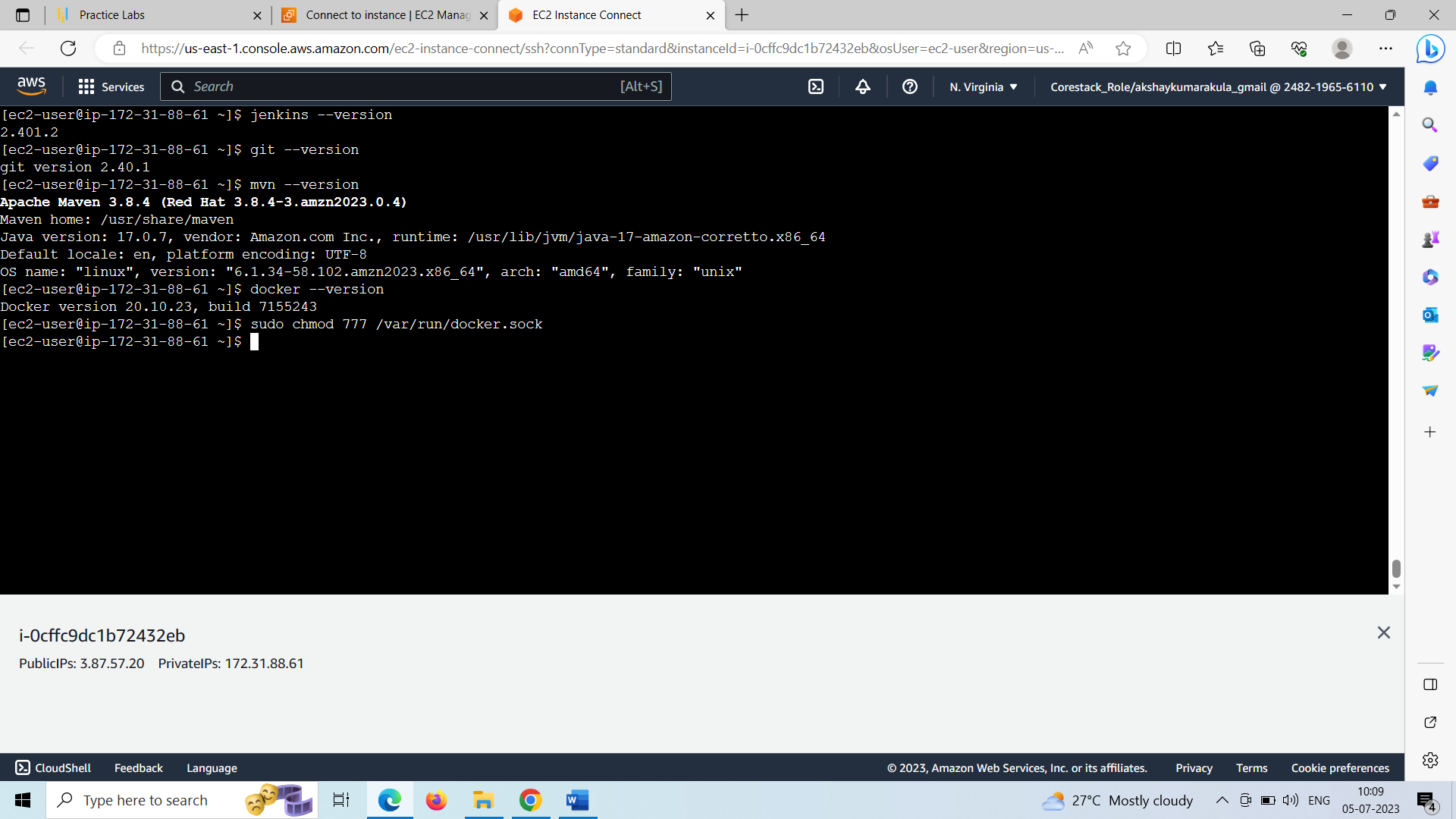




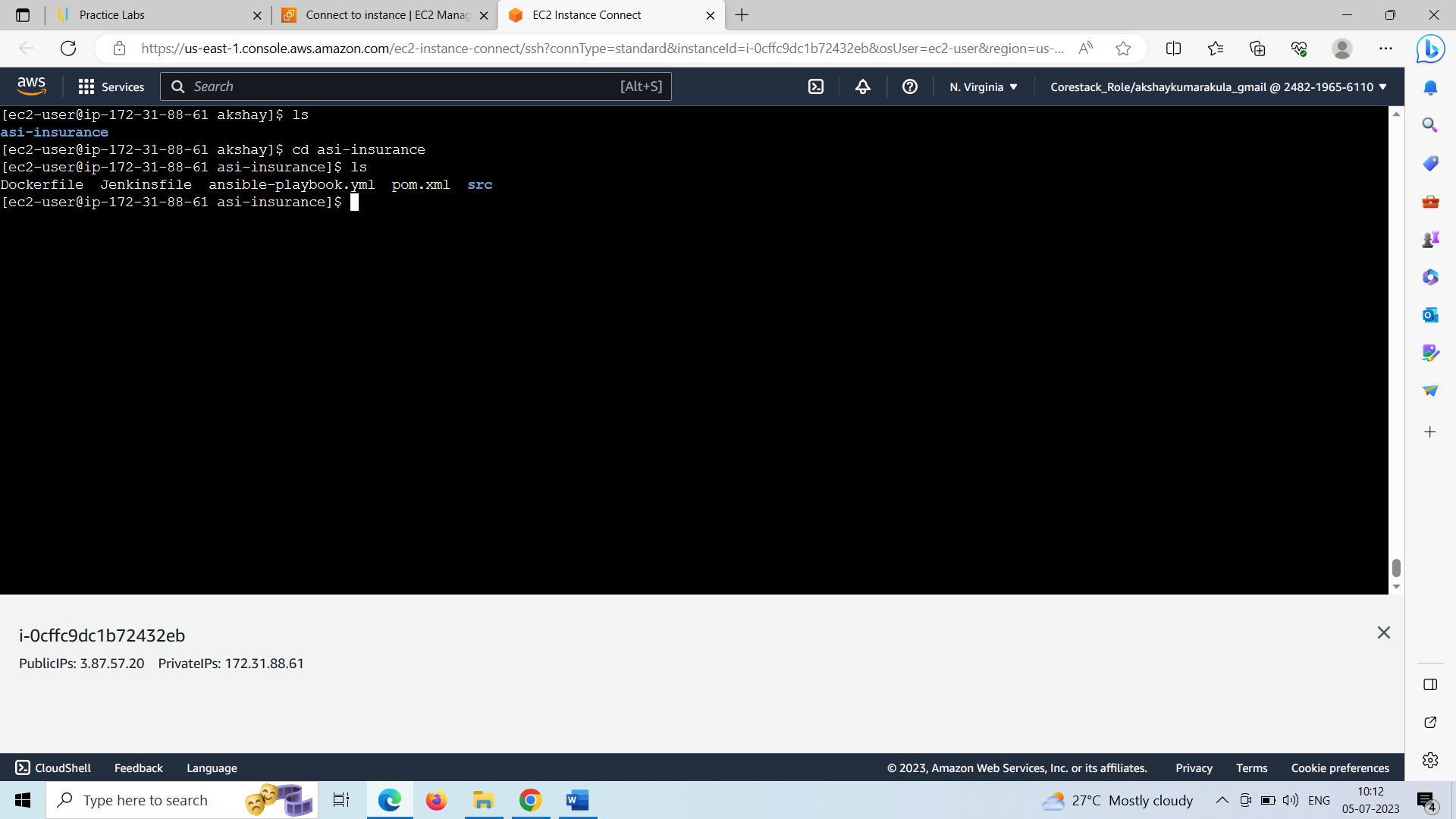
**Step – 2:** Connectto the ec2 instance and install Jenkins, git, maven and docker on the ec2 instance



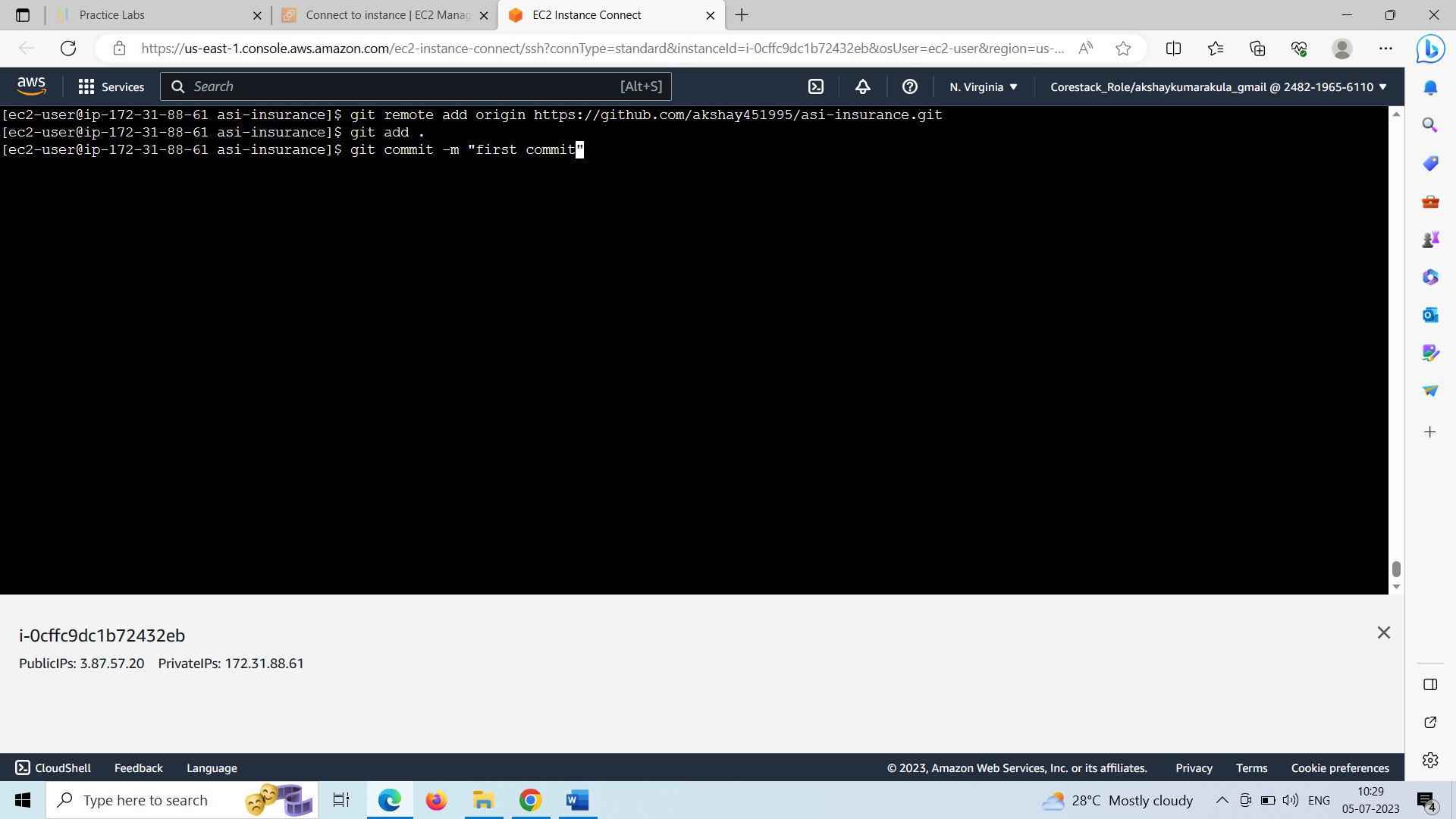
**Step – 3:** Changed the permissions of docker.sock file to give access to docker to build containers and images

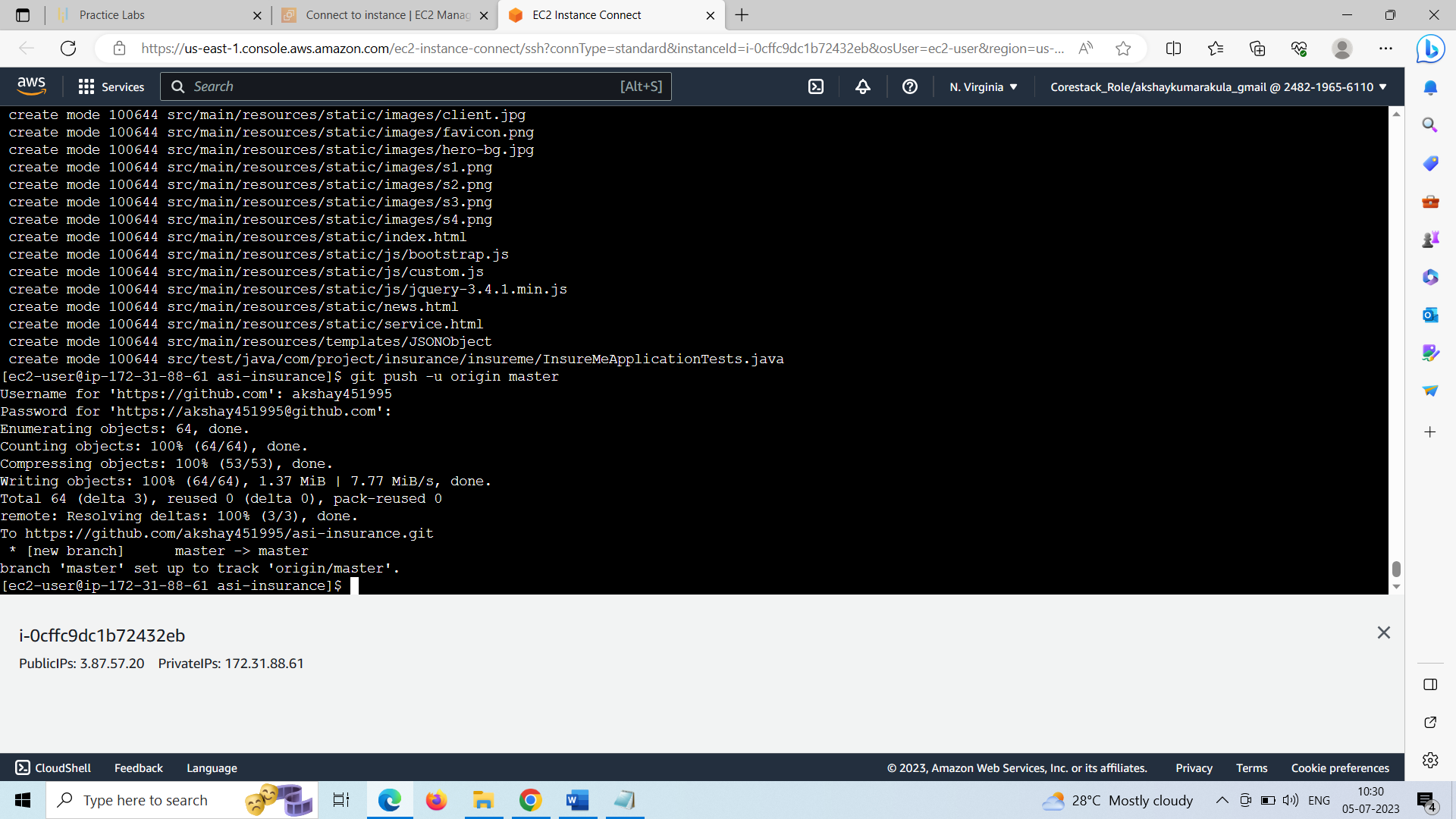


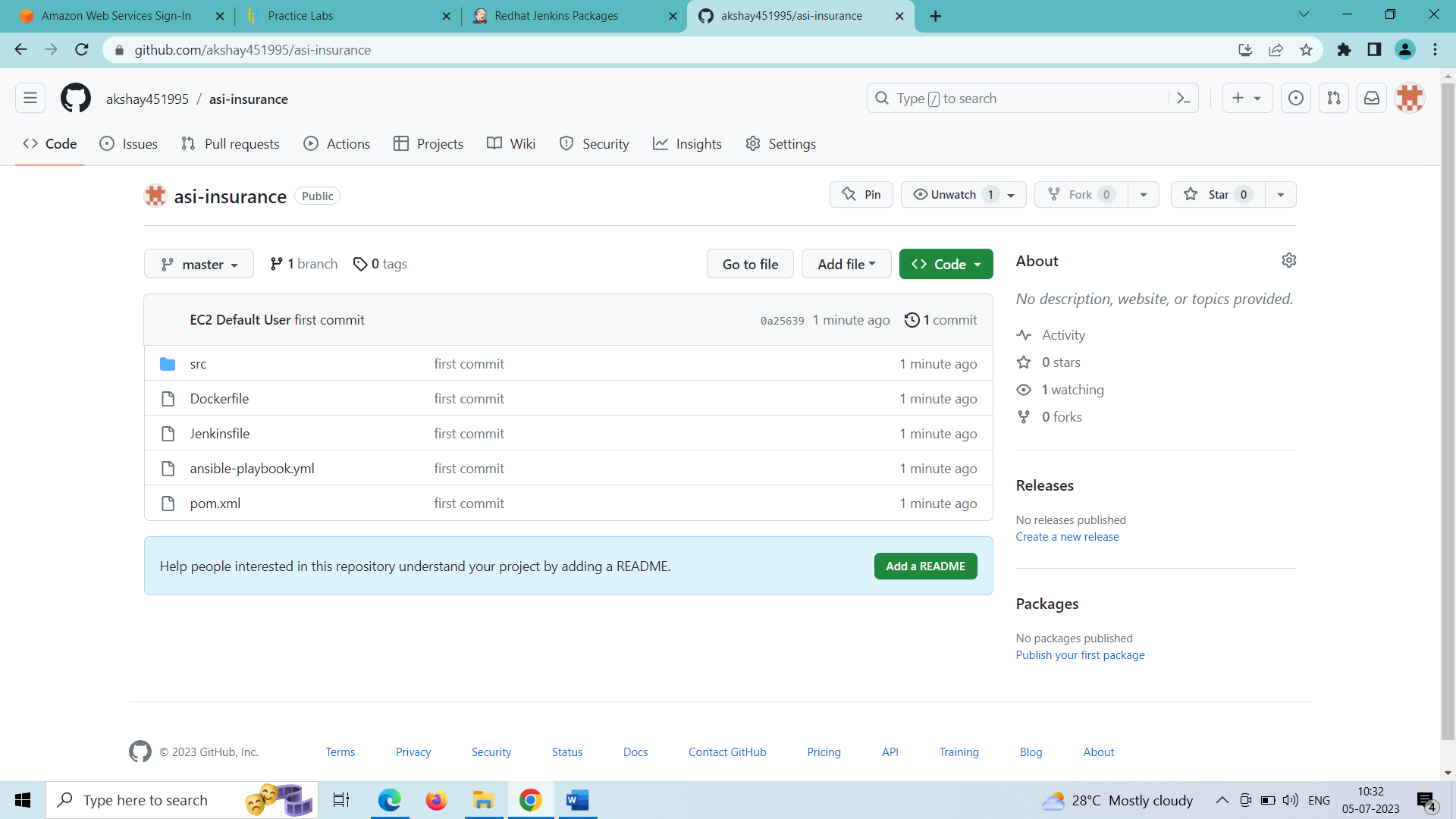
**Step – 4:** Created a directory named asi-insurance and created the Jenkinsfile, Dockerfile, ansible playbook (ansible-playbook.yml) and the source file of the static website inside the asi-insurance directory



**Step – 5:** Created an empty repository inside the git hub account named asi-insurance and uploaded all the files from the asi-insurance directory to the asi-insurance git repository



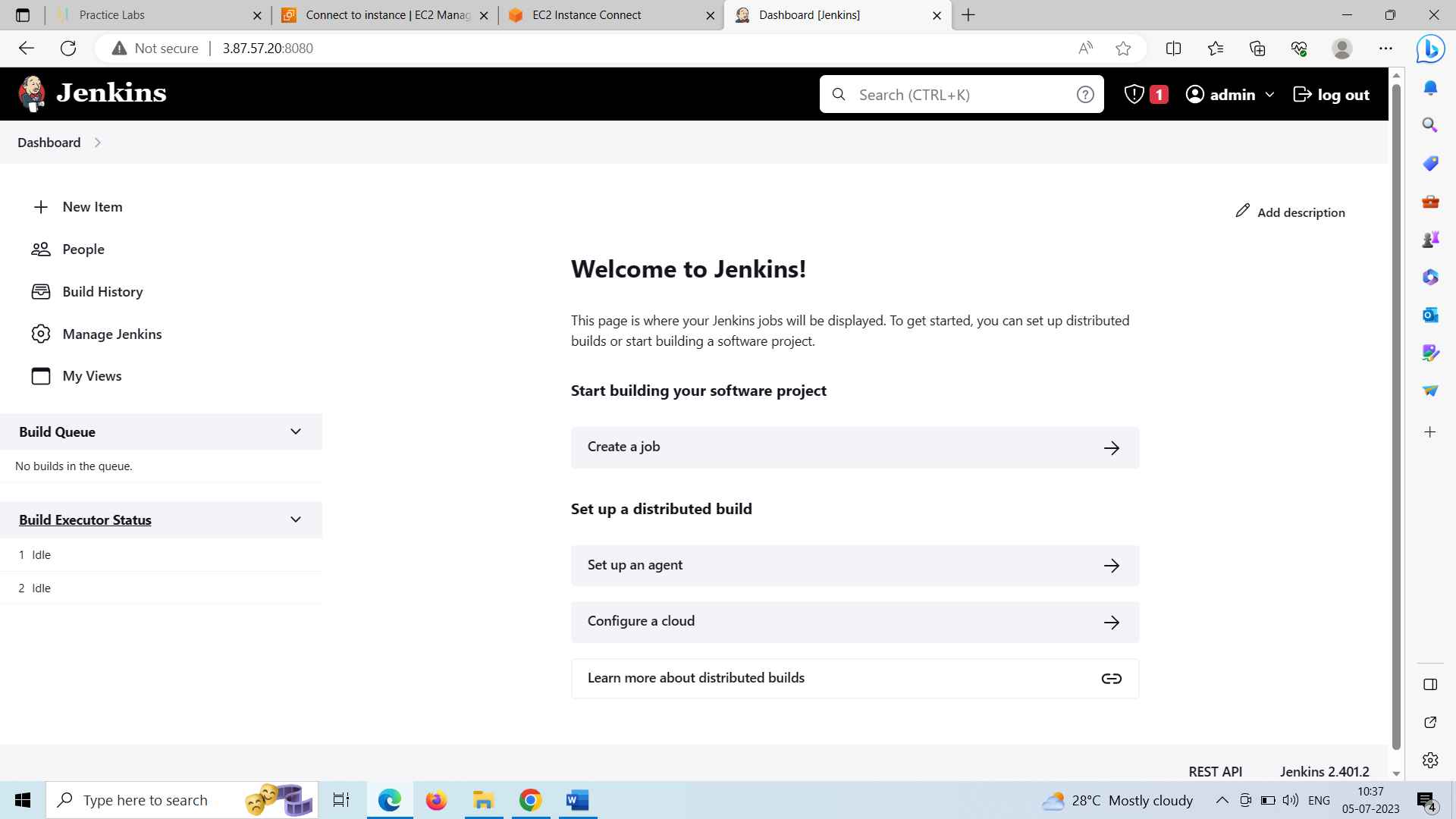




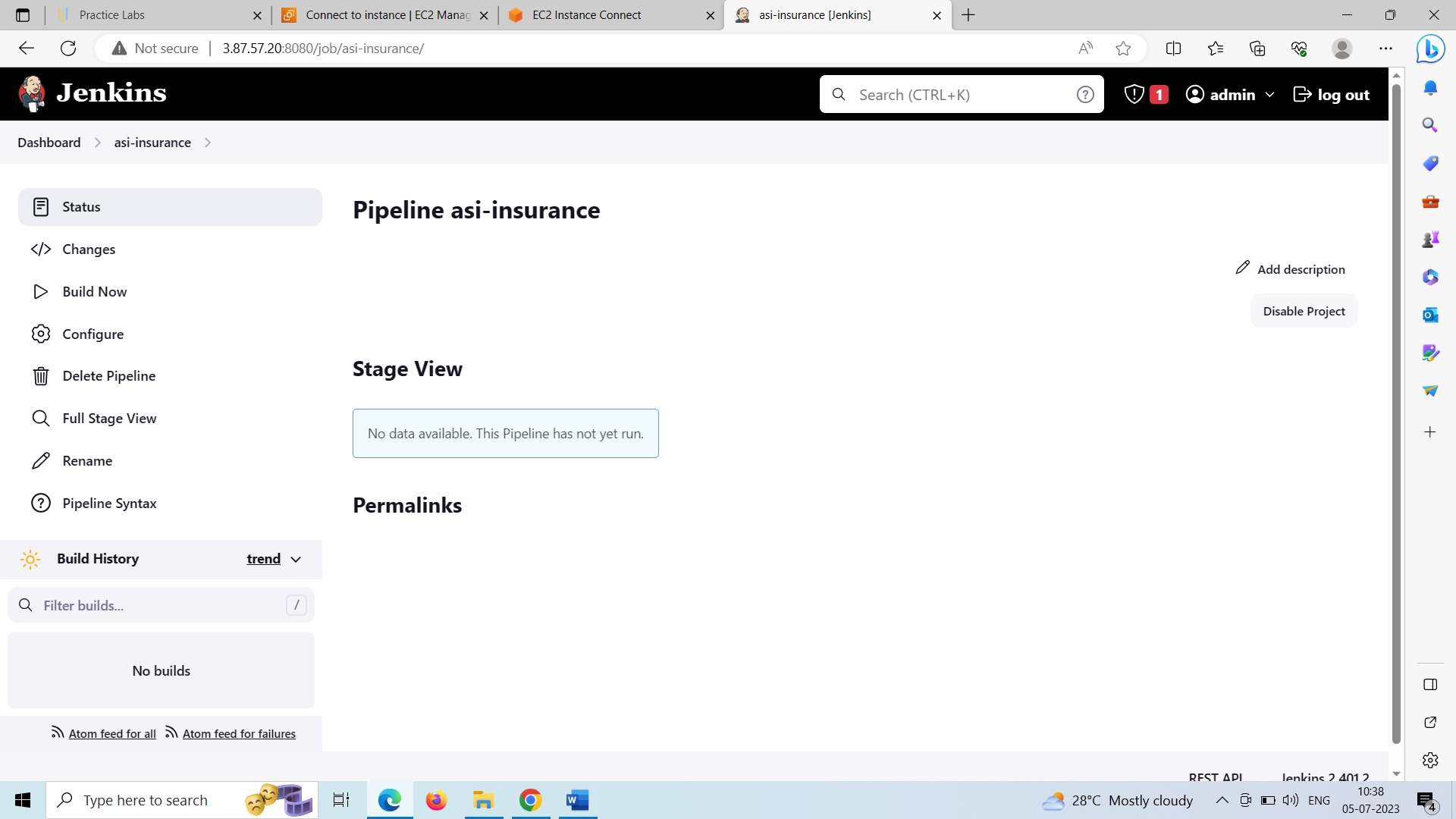
**Step – 6:** Installing nodeJS inside the terminal:

sudo yum install nodejs -y

**Step – 7:** Opened the browser to access Jenkins application and set all the admin credentials and installed required plugins

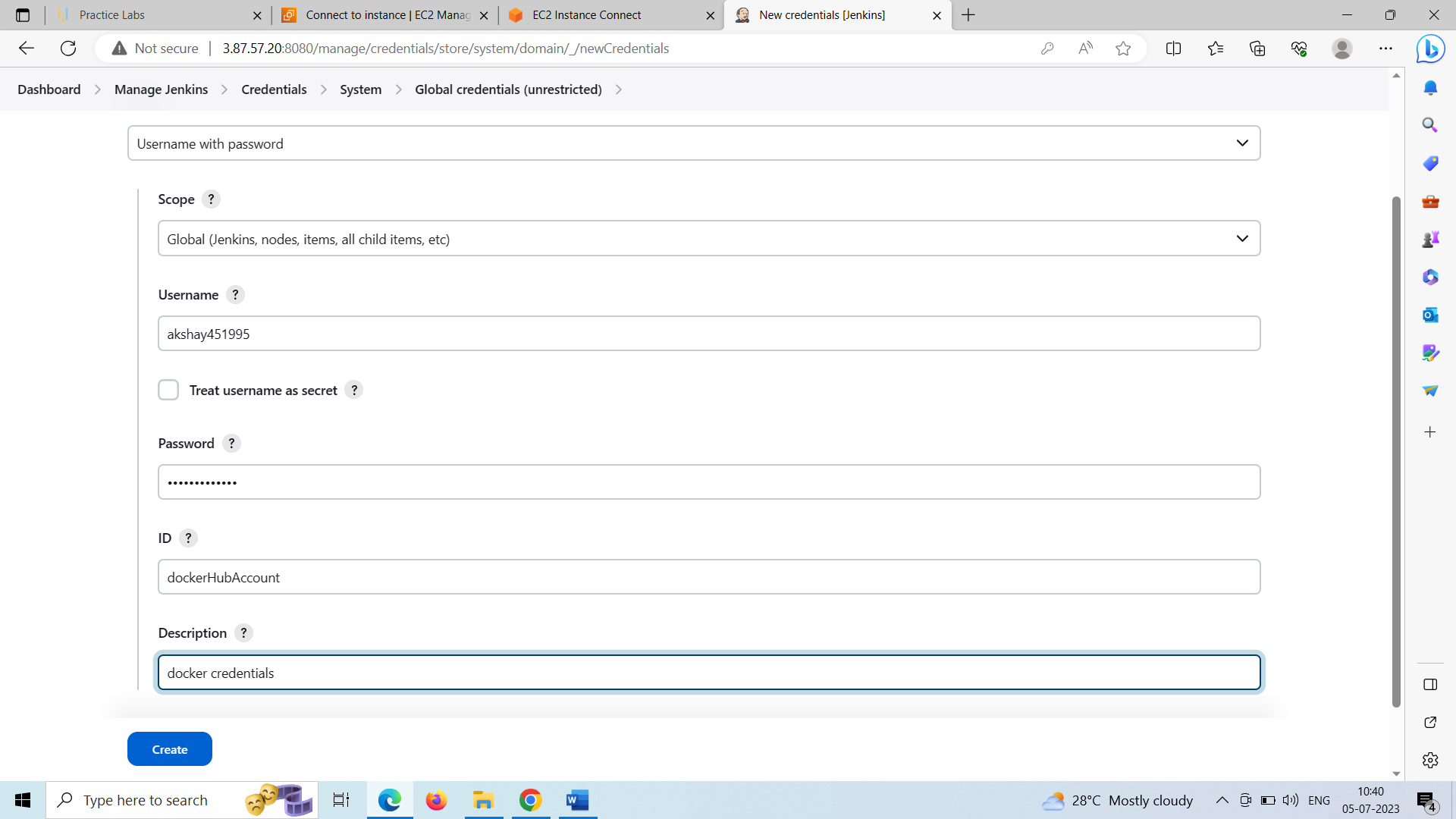


**Step – 8:** Created asi-insurance pipeline to perform CI/CD for Docker container

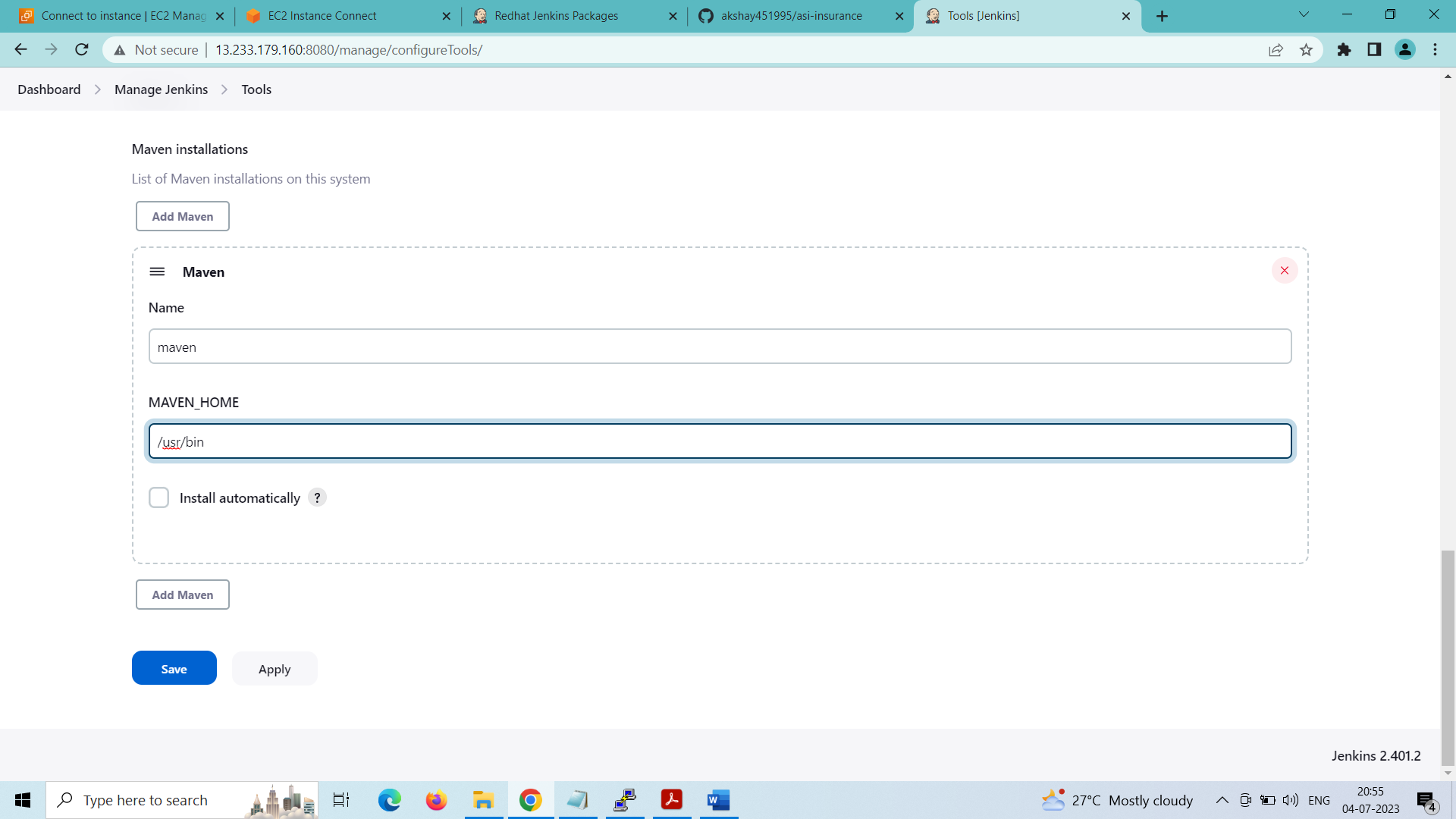


**Step – 9:** Created docker hub credentials and configured maven tool required for building the source code from SCM

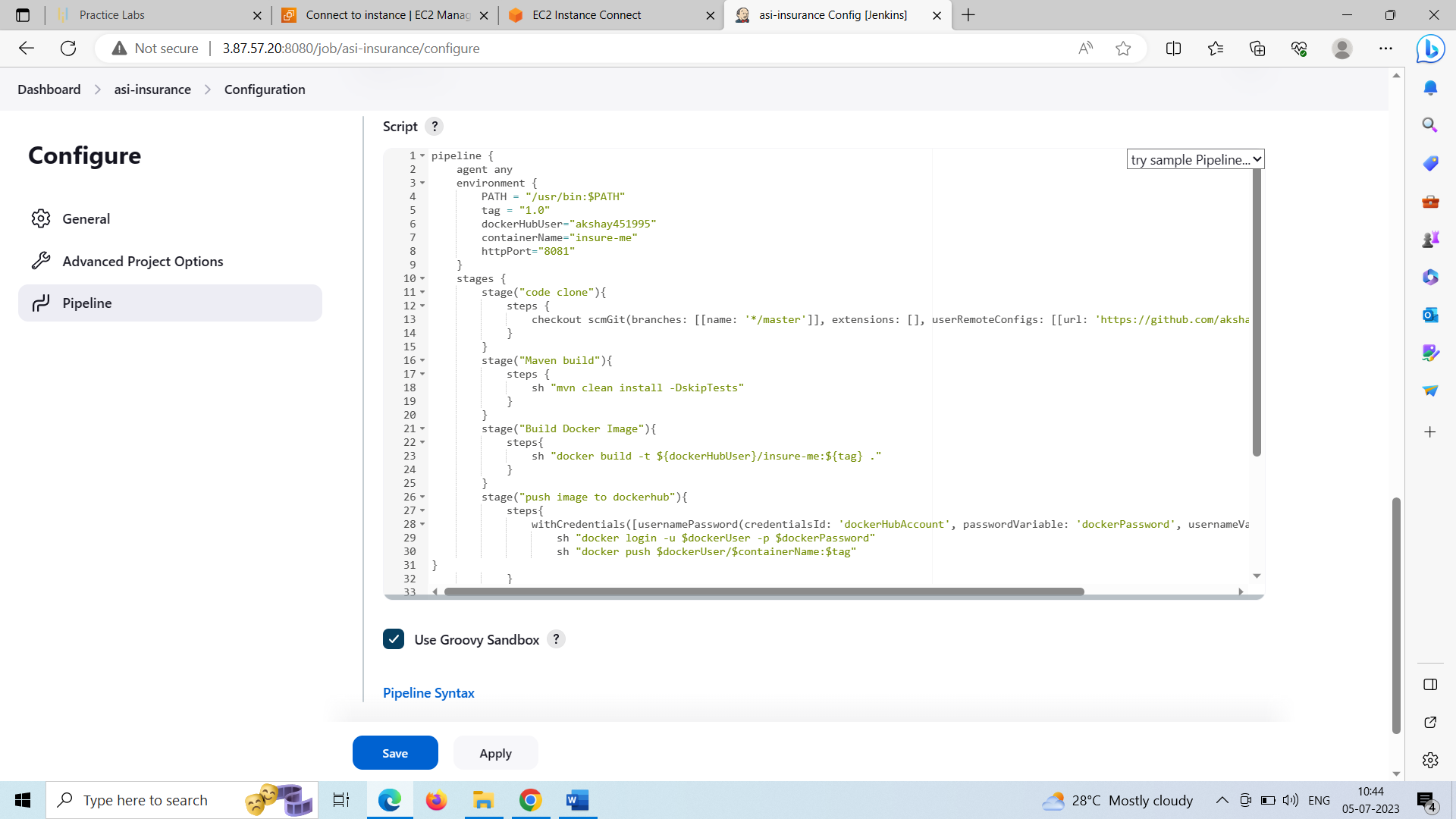
Docker hub credentials:



Maven tool configuration:



**Step – 10:** Configure the asi-insurance pipeline job and write the pipeline script using pipeline syntax option below:



Inside the pipeline script:

pipeline {

agent any

environment {

PATH = "/usr/bin:$PATH"

tag = "1.0"

dockerHubUser="akshay451995"

containerName="insure-me"

httpPort="8081"

}

stages {

stage("code clone"){

steps {

checkout scmGit(branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/akshay451995/asi-insurance.git']])

}

}

stage("Maven build"){

steps {

sh "mvn clean install -DskipTests"

}

}

stage("Build Docker Image"){

steps{

sh "docker build -t ${dockerHubUser}/insure-me:${tag} ."

}

}

stage("push image to dockerhub"){

steps{

withCredentials([usernamePassword(credentialsId: 'dockerHubAccount', passwordVariable: 'dockerPassword', usernameVariable: 'dockerUser')]) {

sh "docker login -u $dockerUser -p $dockerPassword"

sh "docker push $dockerUser/$containerName:$tag"

}

}

}

stage("Docker container deployment"){

steps{

sh "docker rm $containerName -f"

sh "docker pull $dockerHubUser/$containerName:$tag"

sh "docker run -d --rm -p $httpPort:$httpPort --name $containerName $dockerHubUser/$containerName:$tag"

echo "Application started on port: ${httpPort} (http)"

}

}

}

}

Description of pipeline:

1st step – setting of environment variables of the pipeline

2nd step – cloning the code from github

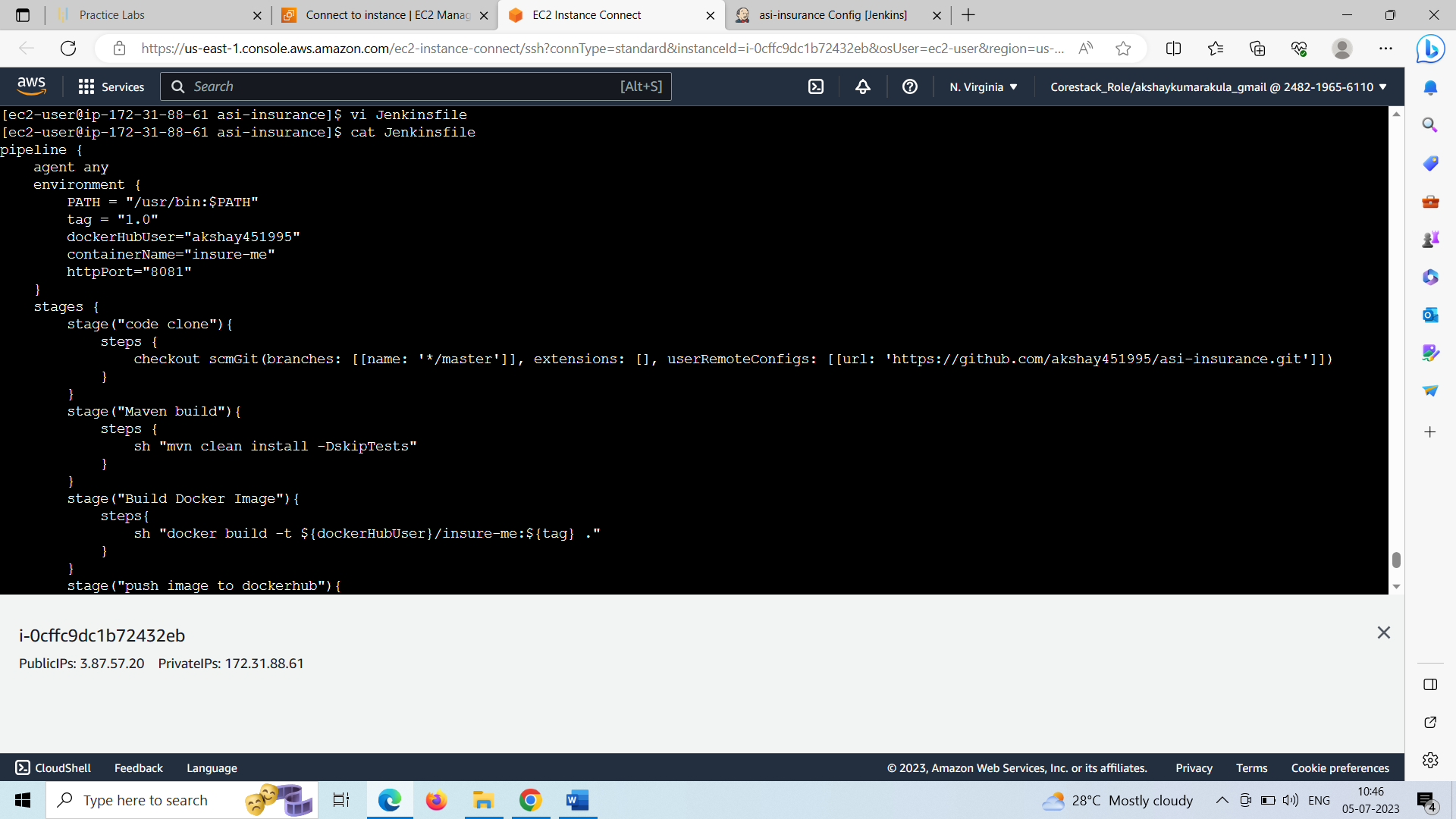
3rd step – building the maven project to install the static website application

4th step – building docker image

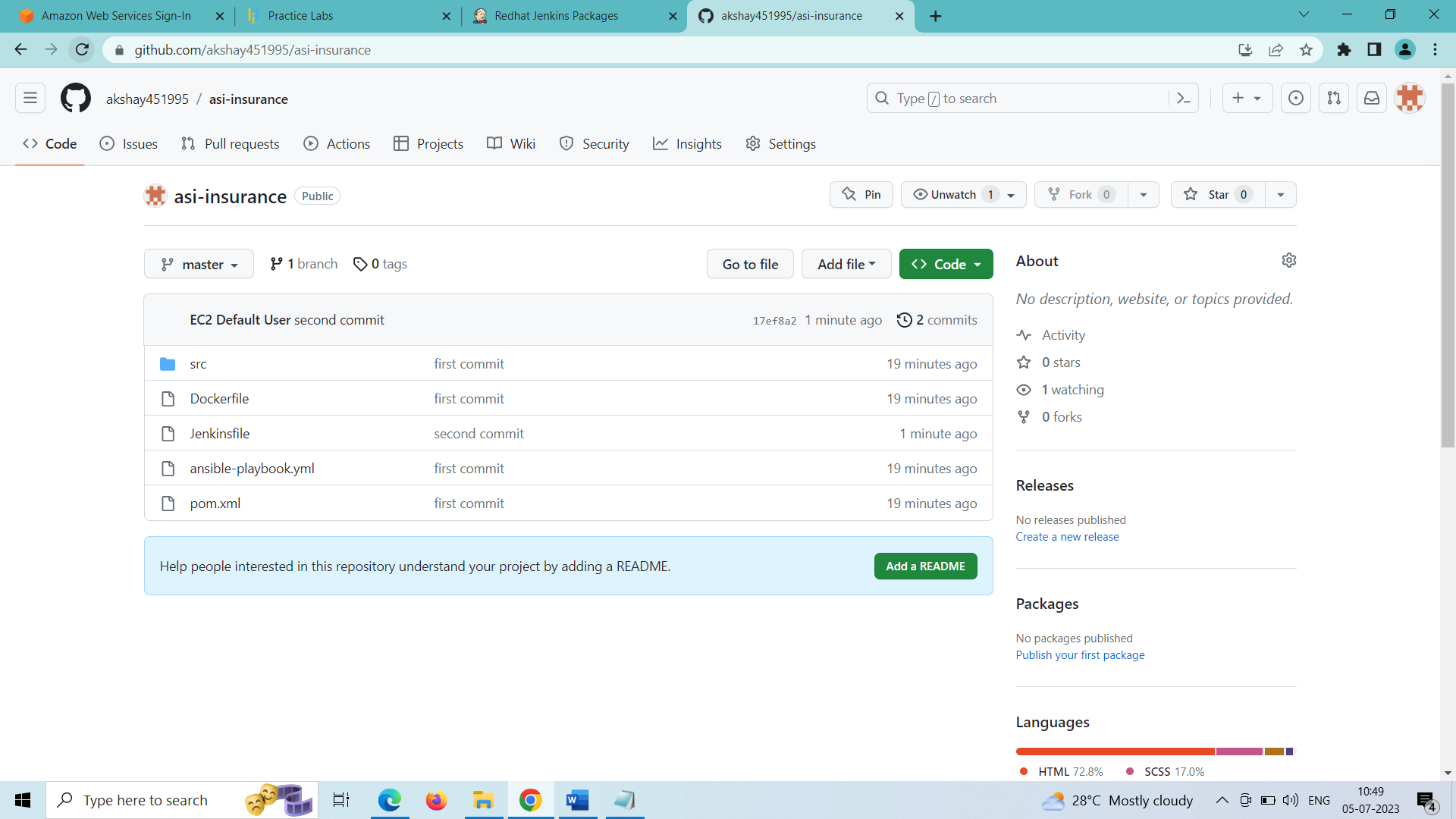
5th step – logging into the docker hub using docker credentials and pushing the docker image to the docker hub

6th step – creating and deploying docker container from the pulled docker image

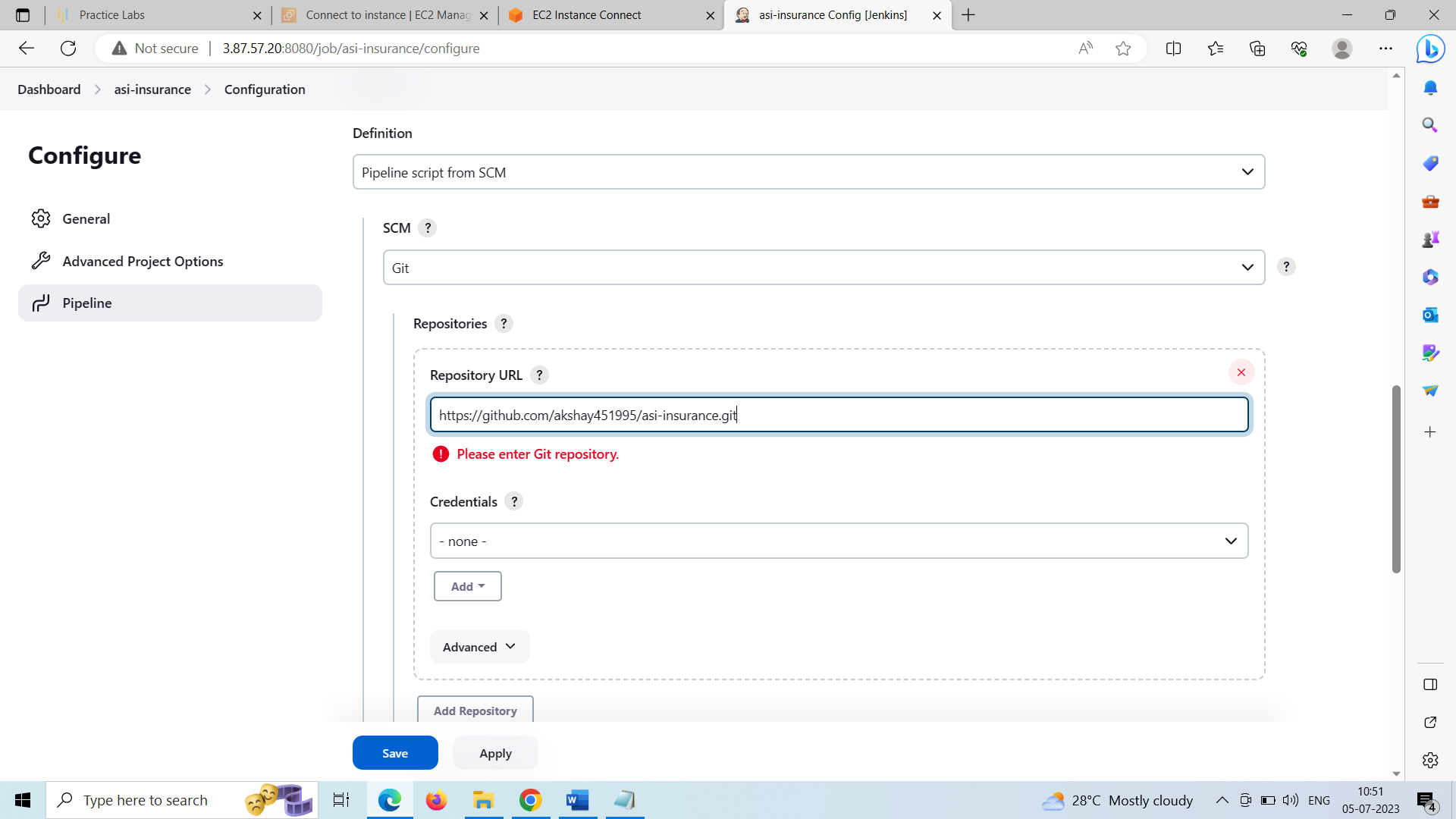
**Step – 11:** Copy this script into the Jenkinsfile inside the asi-insurance directory from the teriminal and commit the changes to the git repository:



Commited the pipeline script to the Jenkinsfile in github as “second commit”:



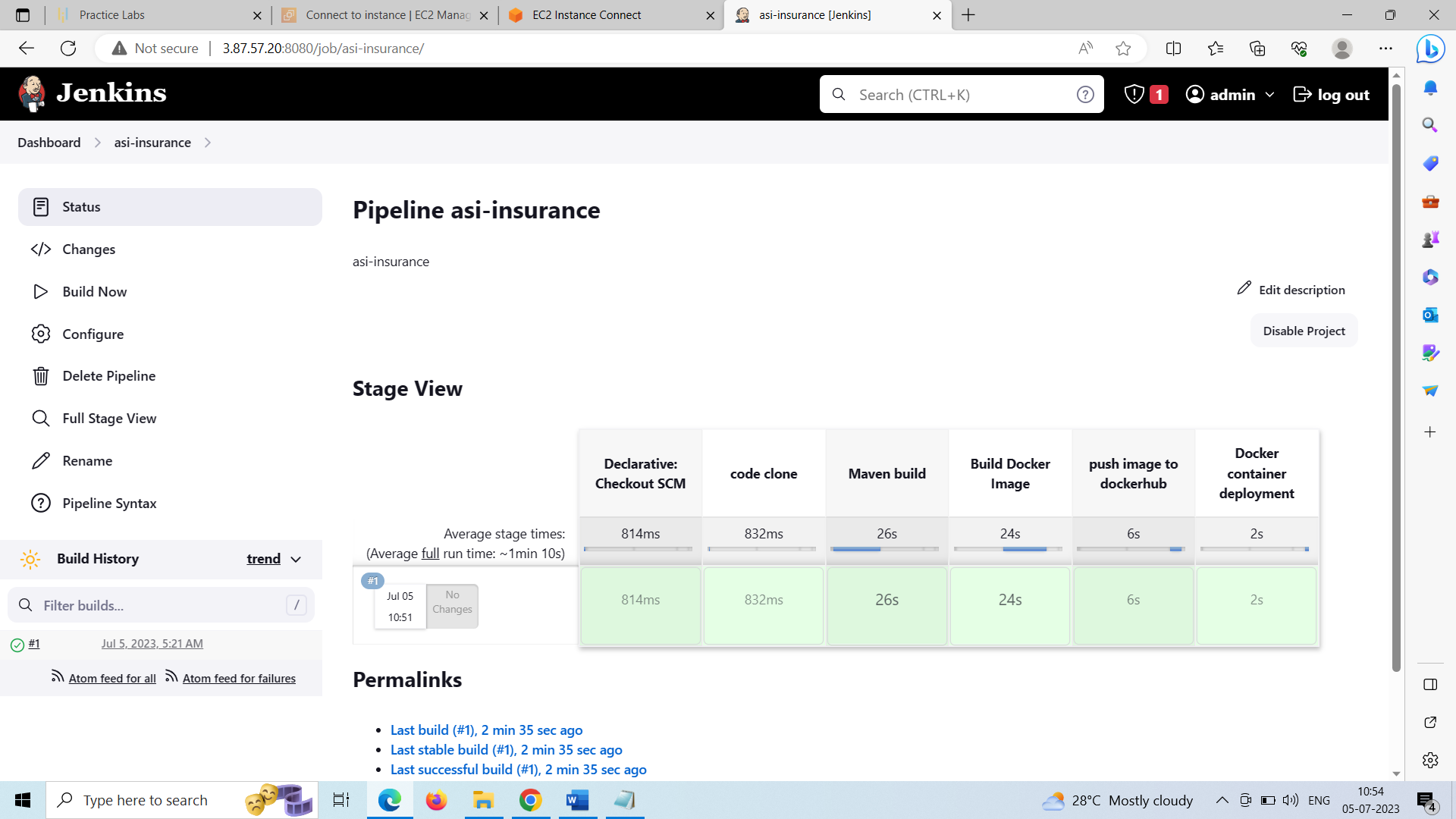
**Step – 12:** Configure the Jenkins asi-insurance pipeline by changing from Pipeline script to Pipeline script from SCM



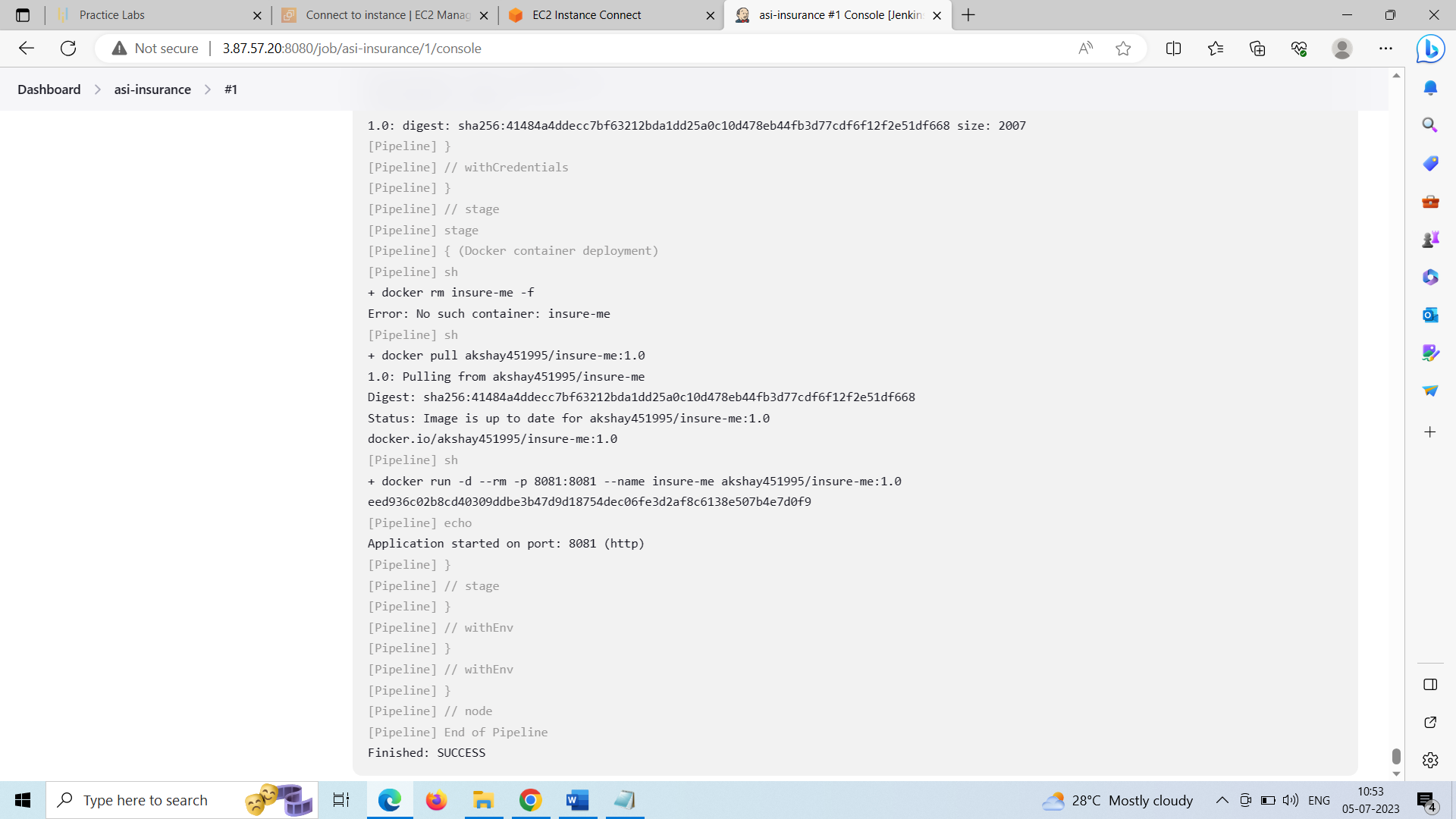
Add the git repository and its credentials if any

**Step – 13:** Execute the Jenkins build by clicking on build now for the asi-insurance pipeline

Stage view of the pipeline: pipeline executed successfully



Console output: docker container with the application built successfully on port 8081

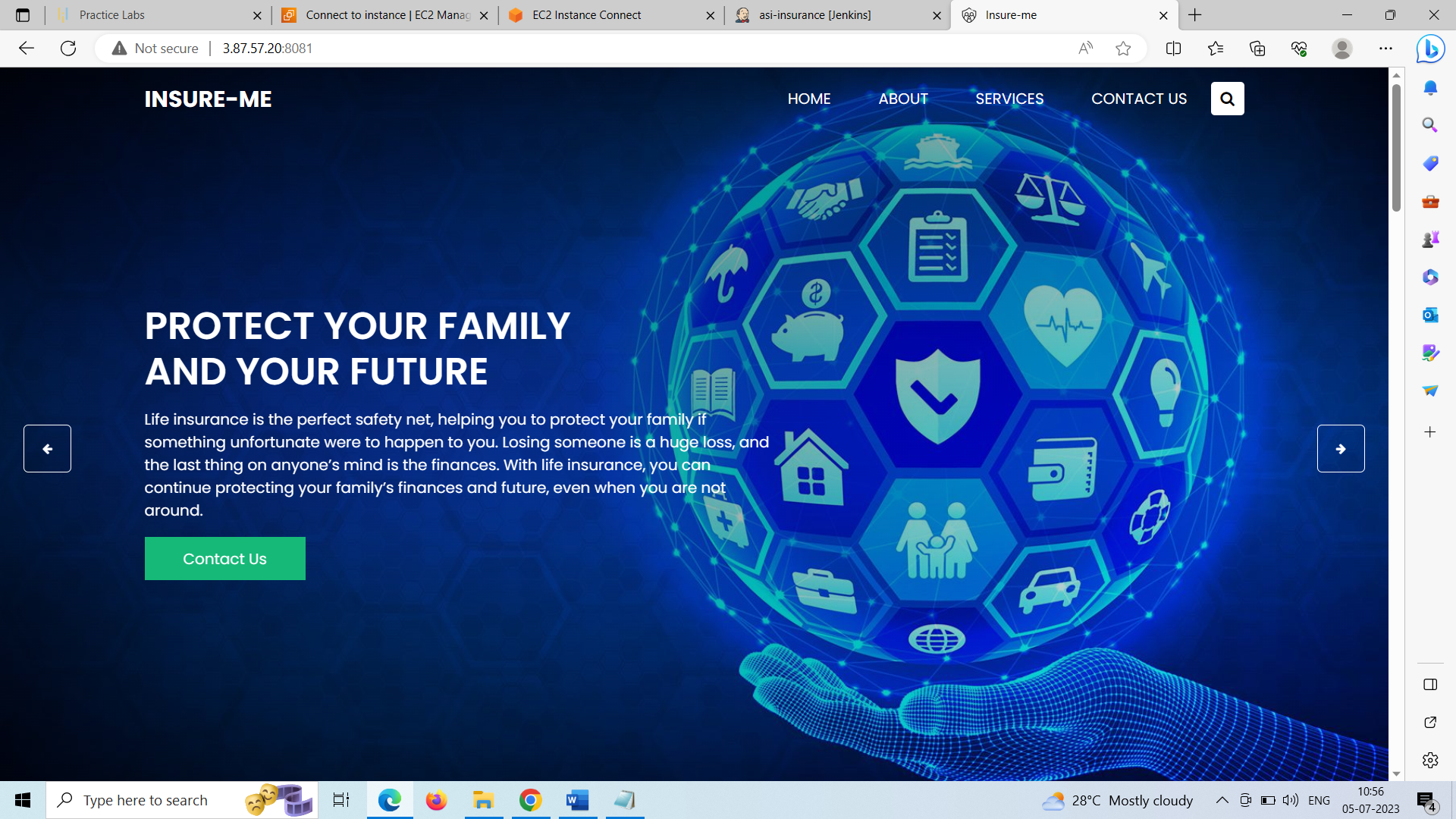


**Step – 14:** Now access the static website using the public IP of the machine with port 8081

http://<public-ip>:8081

<http://3.87.57.20:8081>

Output Screenshot:



Service section page:

