This is simple project using Springboot Application to display Employee details

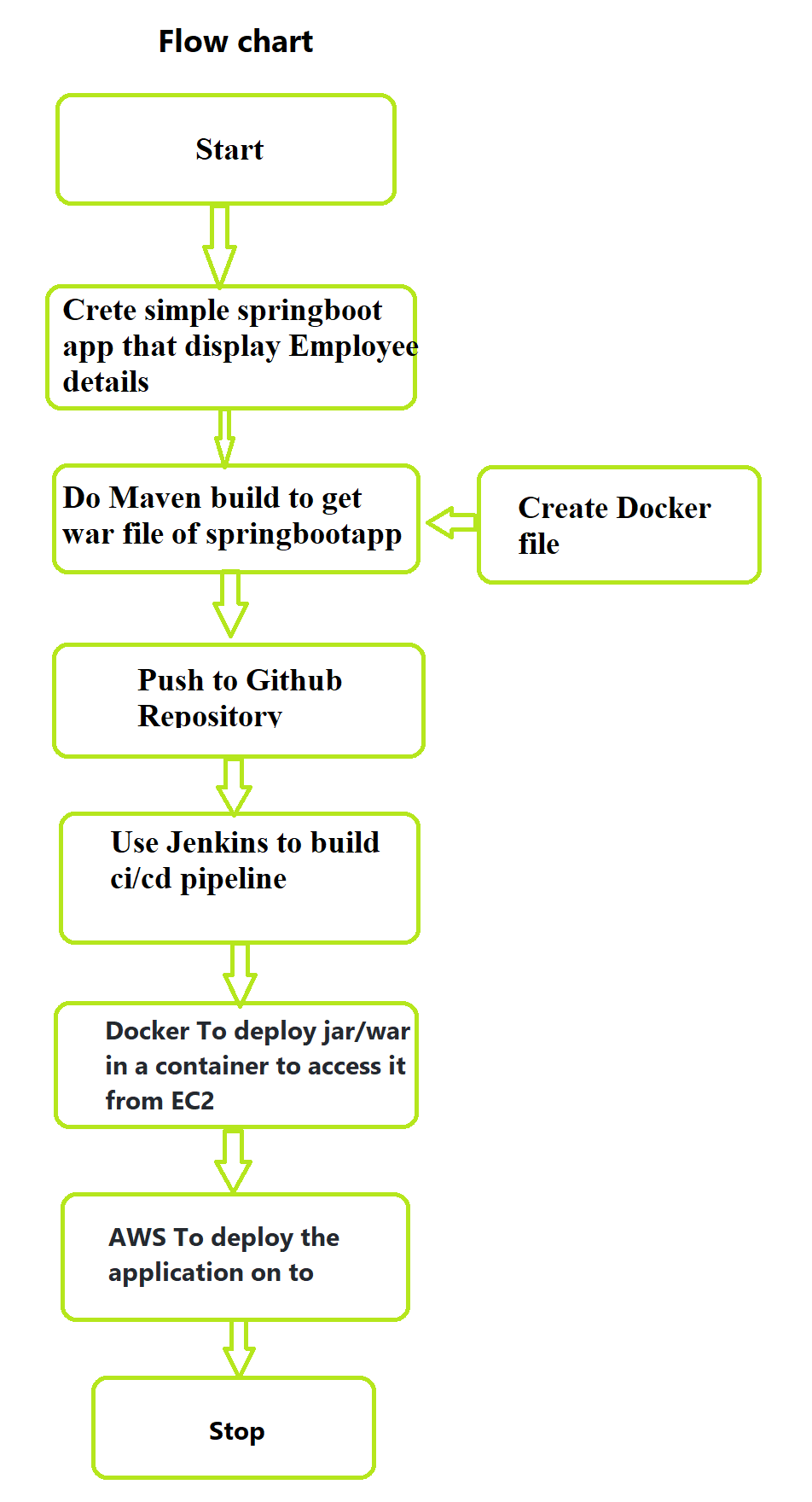
**Tools Used**

1. Eclipse: An IDE to code for the application
2. Java: A programming language to develop the prototype
3. Maven: A build tool for adding all the external dependencies
4. Git: To connect and push files from the local system to GitHub
5. GitHub: To store the application code and track its versions
6. Jenkins: To build CI/CD pipelines
7. Docker: To deploy jar in a container to access it from EC2
8. AWS: To deploy the application

**Data Structures & Techniques Used**

1. Spring Boot : Popular Java Framework to acheive inversion of control and dependency injection
2. Spring MVC : Model View Controller by Spring

**Application Flow Chart**

****

**Step by Step Build Process**

**1. Create a Java Project**

Create a simple Spring project using Eclipse and Maven and push it to GitHub

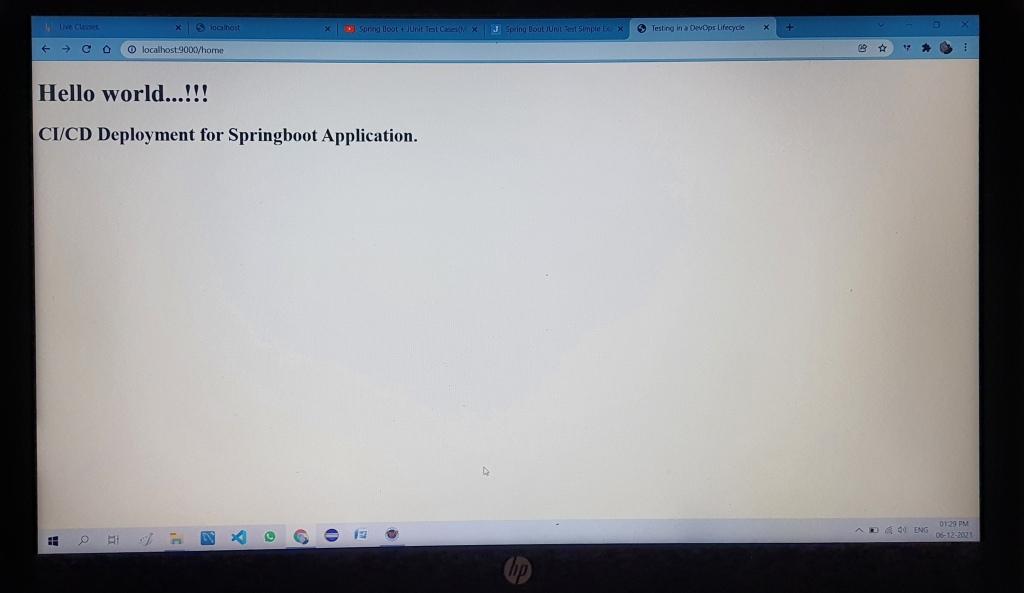
**Sample Output**

**2. Configure Jenkins**

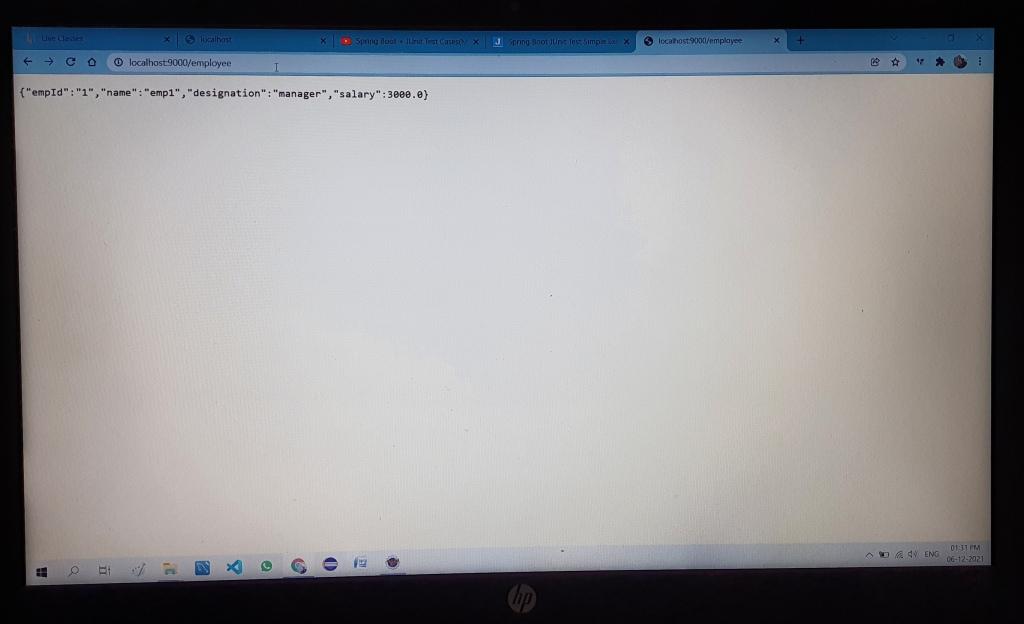
Create a new task and use Git repository as source code management

**Home page Results from Spring boot application**

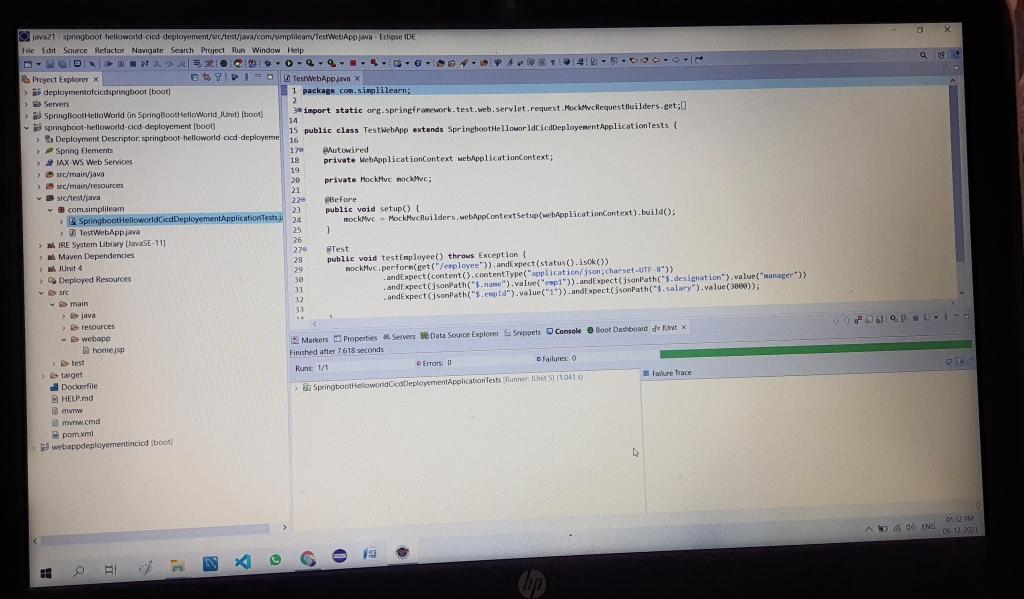
**Home page**



**Employee page**

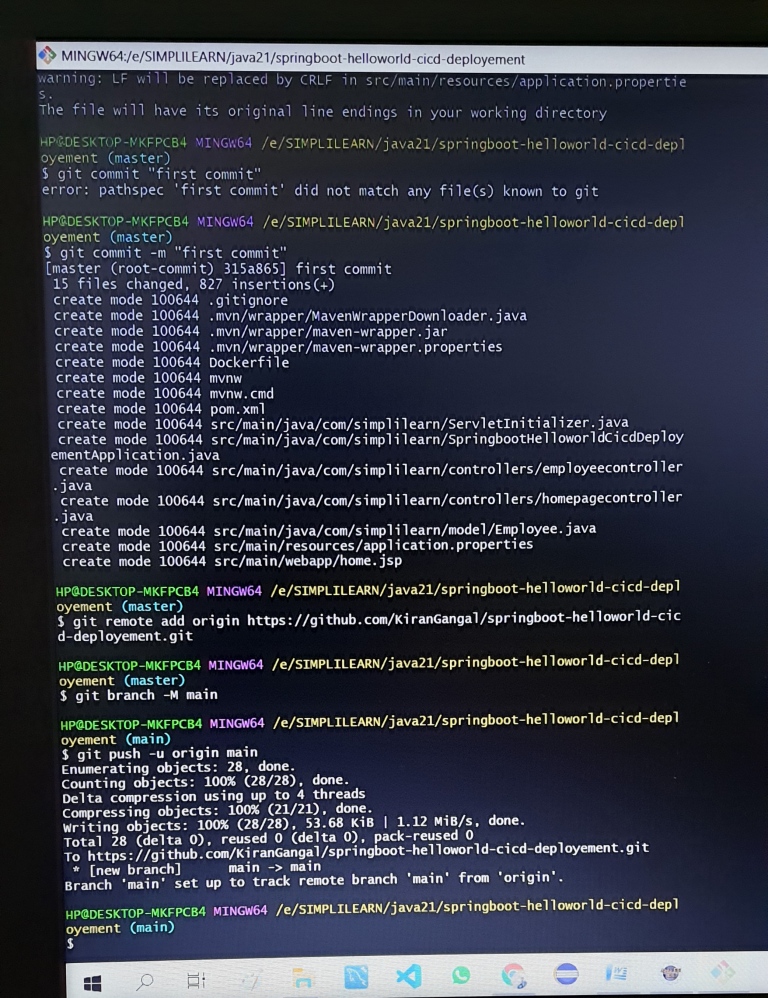


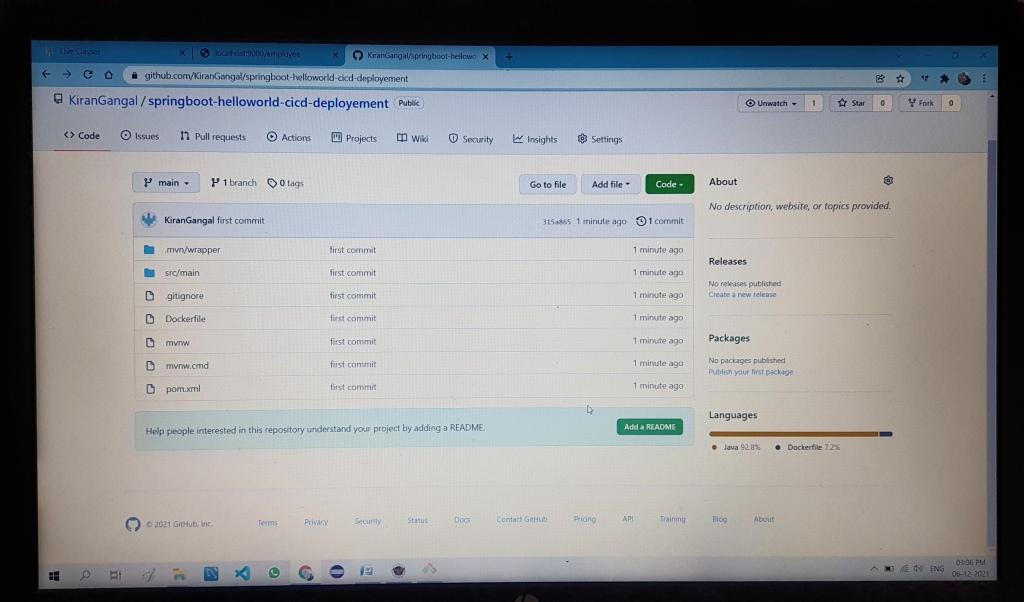
**Junit testing**

****

**Git Hub Repository**

Push the spring boot app to github

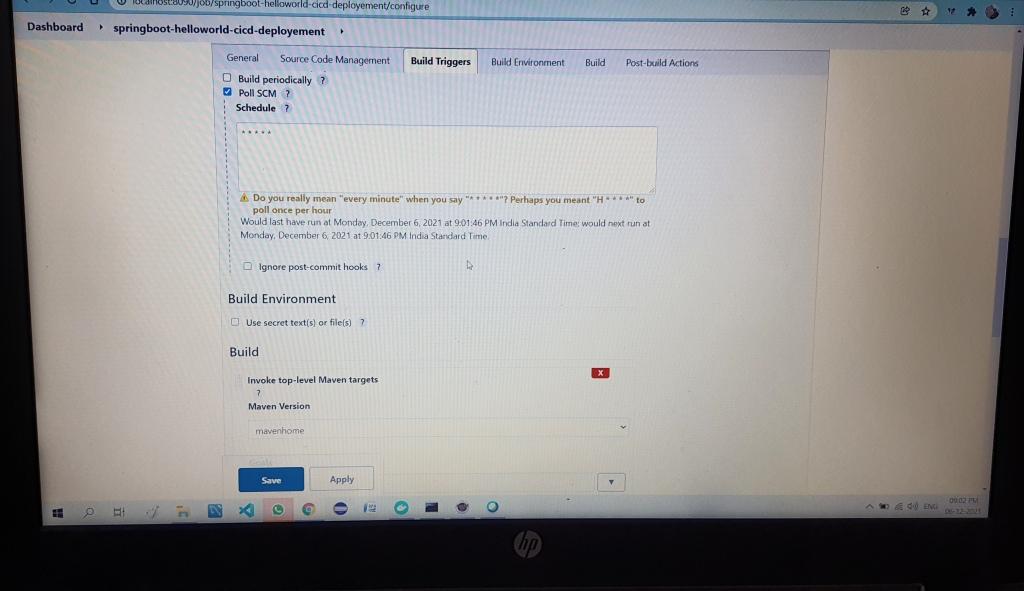




**Jenkins**

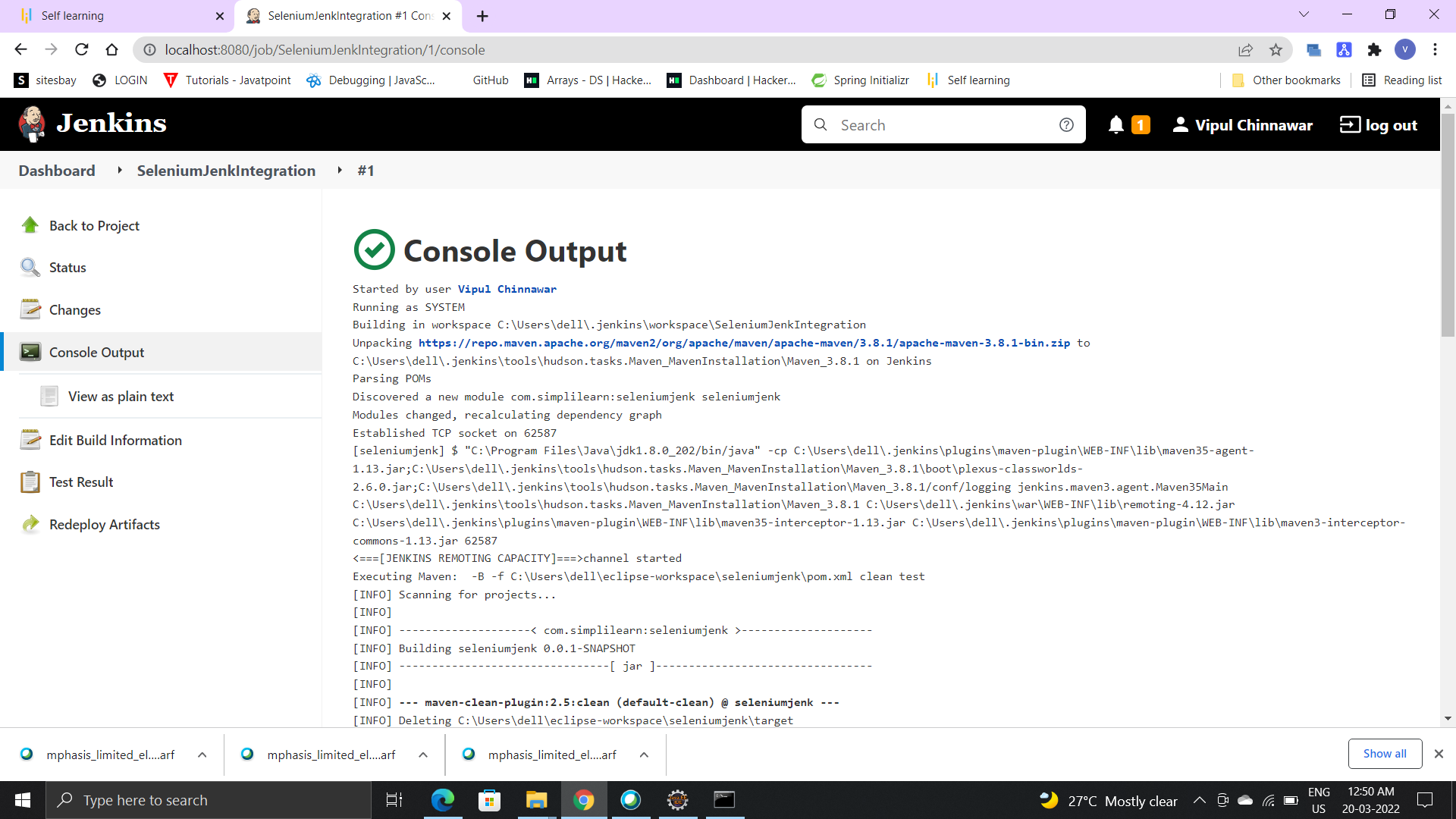
**Jenkin settings**

Select Poll SCM option in order to let jenkins start a new build whenever there is a change in git repository



**Jenkins Build Application**

After all the configuration is done, click build now and wait the magic



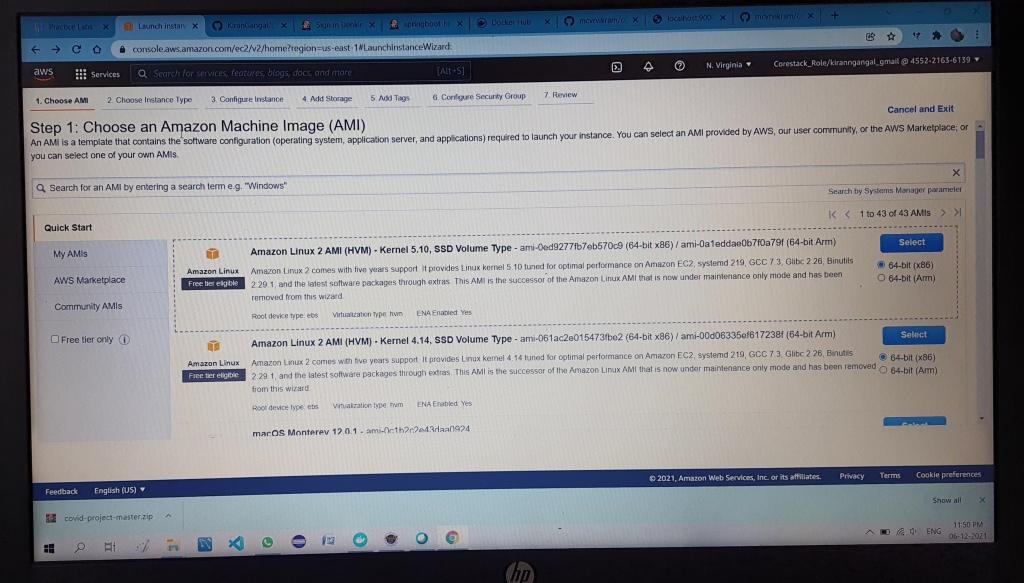
**Deploy Application in Amazon EC2**

* Create a linux EC2 instance and launch it
* Then pull the docker image from docker hub using command 'docker pull userid/imageid'

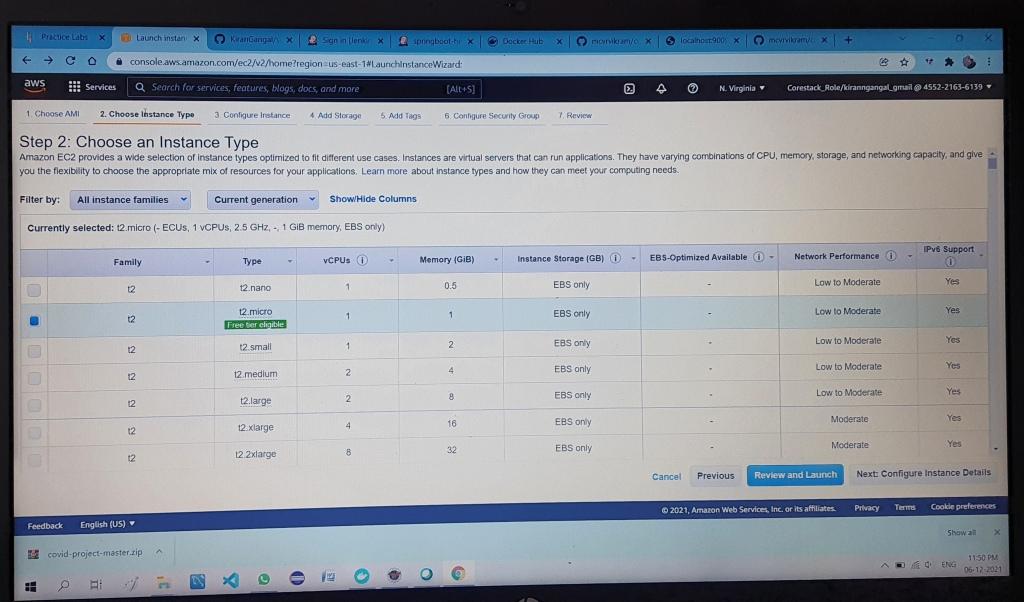
**Amazon EC2 Configurations**

Go to AWS website

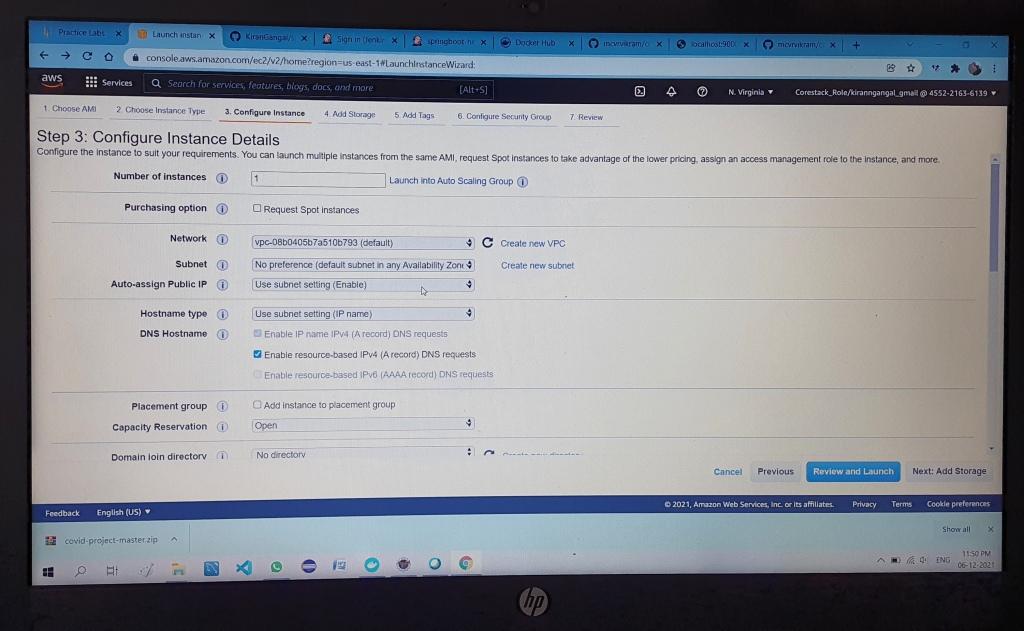
Step1



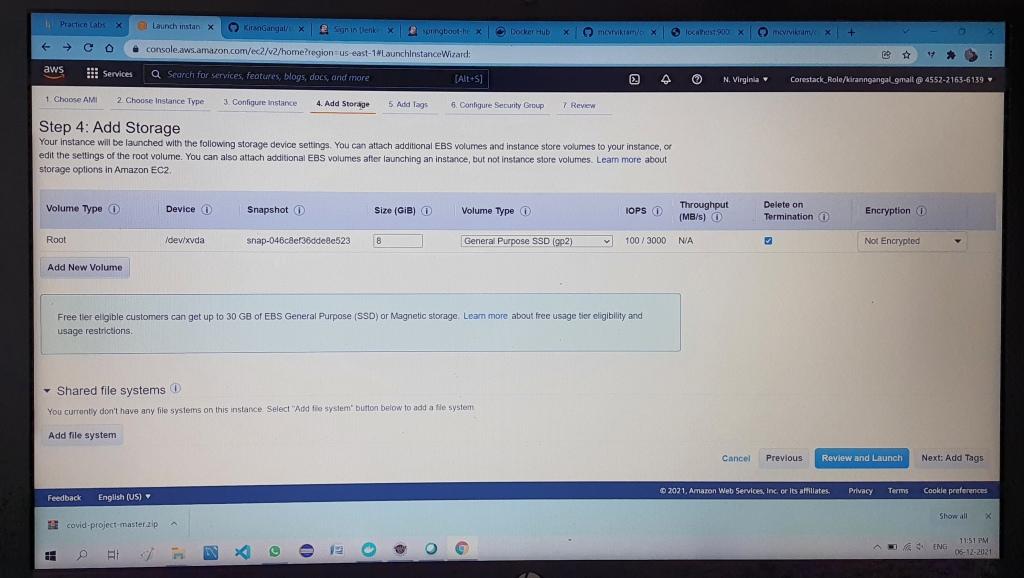
Step2



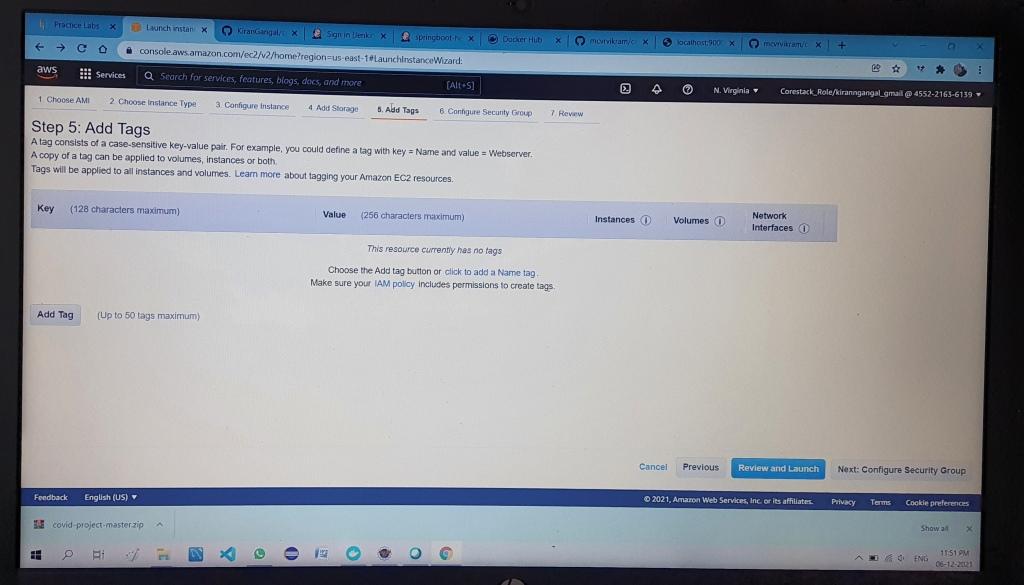
Step 3



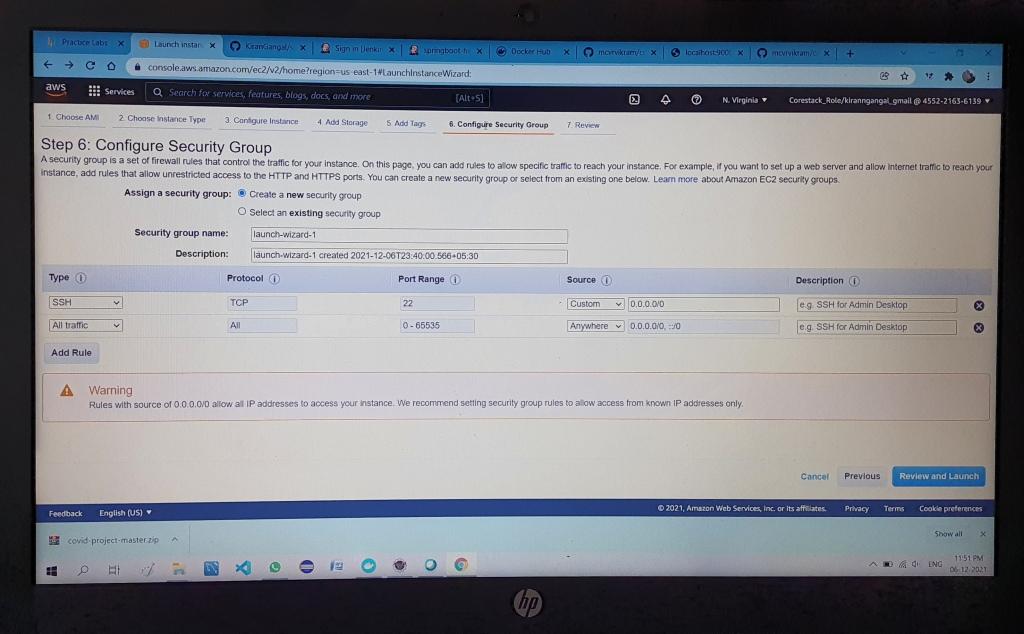
Step 4



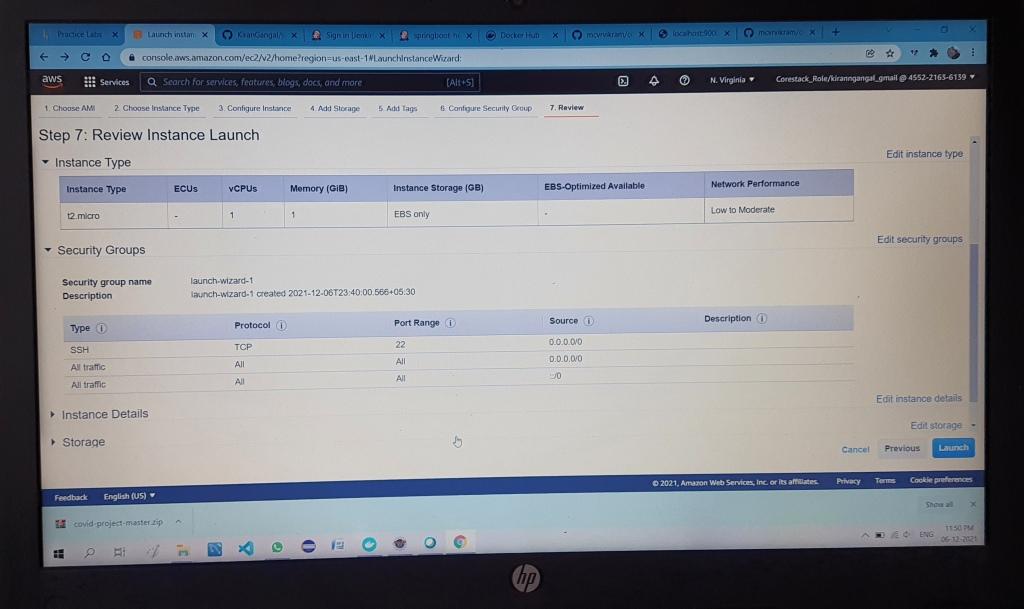
Step 5



Step 6

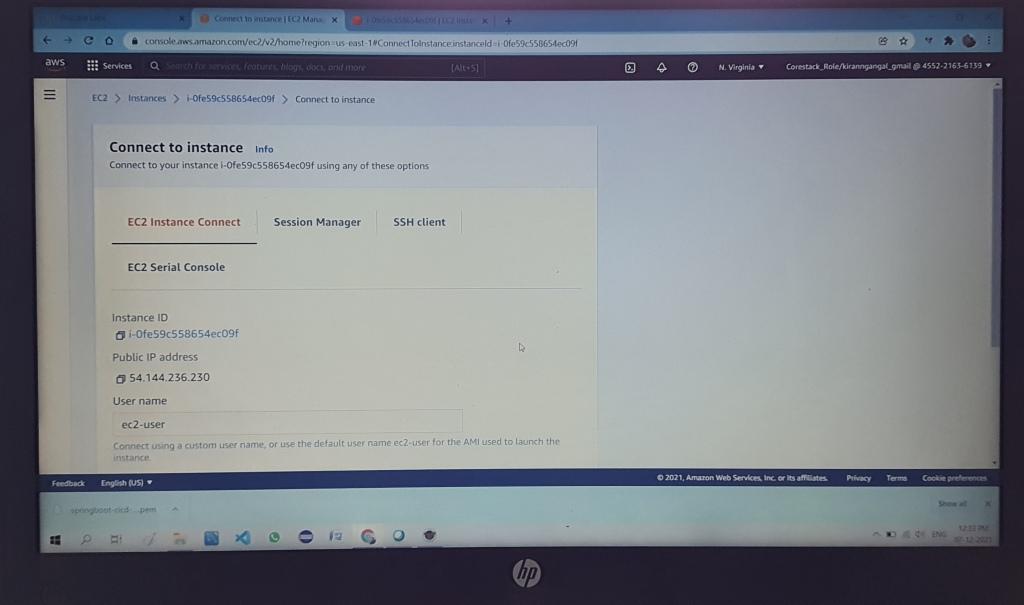


Step 7



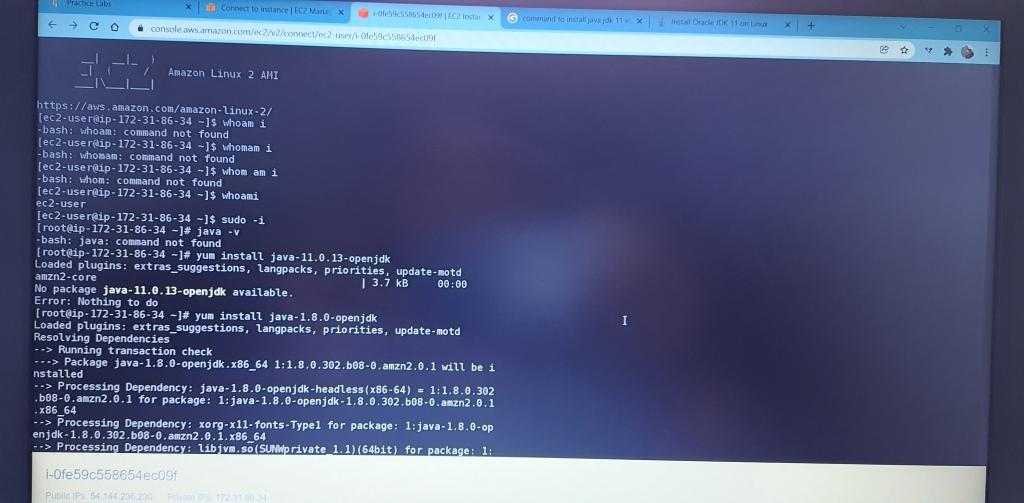
Step 8

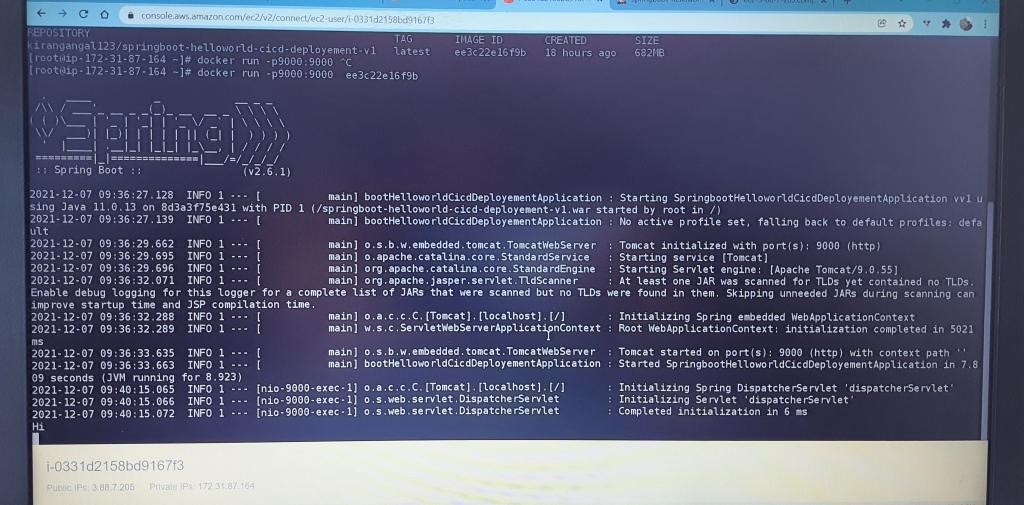
Run the docker image using command 'docker run -p portnumber:portnumber imageid

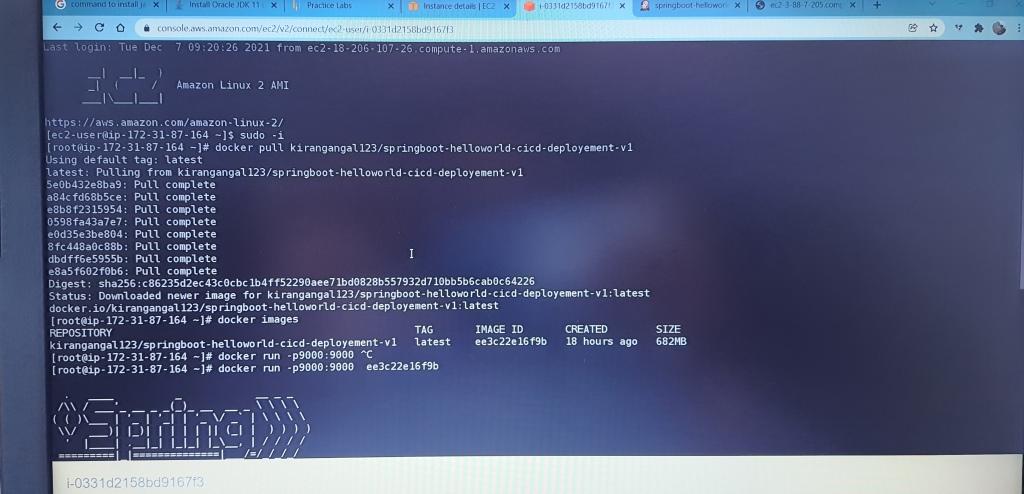


**Run Jar in AWS EC2**

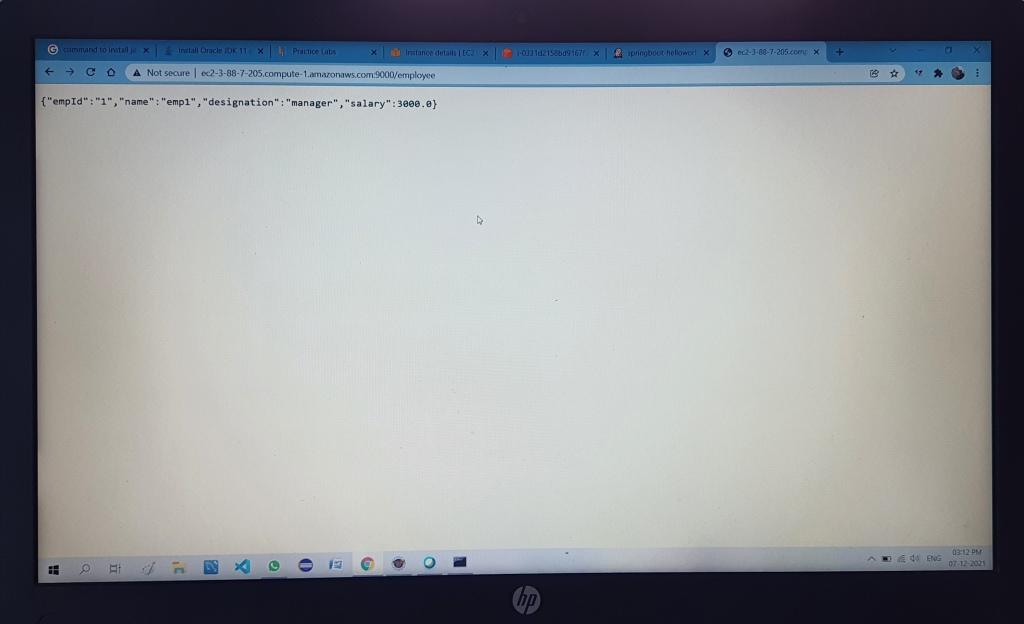
* Once our application is launched in linux EC2 instance, we can go to our EC2 instance and copy the public IP
* Launch our application using the URL publicIP:portnumber and That's it!







**Final Output**

****

