Project 5 - DevOps

CI/CD Deployment for
Spring Boot Chat Demo Application
on AWS EC2

Payam Dowlatyari

Post Graduate Program in Full Stack Web Development

Caltech CTME - Simplilearn

April 2022

Table of Contents

Introduction

Purpose

The main purpose of this document is to explain the process of development of a CI/CD pipeline to demonstrate continuous deployment and host the application on AWS EC2 instance.

Intended Audience

This document is technical and it's designed for field experts. However, it'll be available to all eligible stakeholders that have knowledge to provide input.

Likewise, the main audience of this document are as follows

- **DevOps Experts** include all DevOps team members and engineers
- **Developers** include front-end engineers, backend engineers, database administrators, QA specialists, etc.
- **Project Managers** include product owners and scrum master

Product Scope

Chat Demo is a full stack project developed in Spring Boot in 4 Sprints and will be deployed on AWS EC2 cloud in two Sprints. The DevOps experts should automate the integration and deployment of the web application. The DevOps team is required to set up an environment where the application will be hosted and accessed by users. The source code is supposed to be fetched from a GitHub repository. The main repository should be created and become accessible to stakeholders.

Iteration Process

The deployment of the application is scheduled to be delivered in two sprints.

- **Kick-off meeting:** discuss type of cloud services and required configurations and finalize the development tools and features

- **Sprint week one:** designed the cloud architecture on AWS and create an instance of EC2 and install and test necessary tools and services

- **Sprint week two**: clone the repository on the EC2 instance and run the application along with debugging

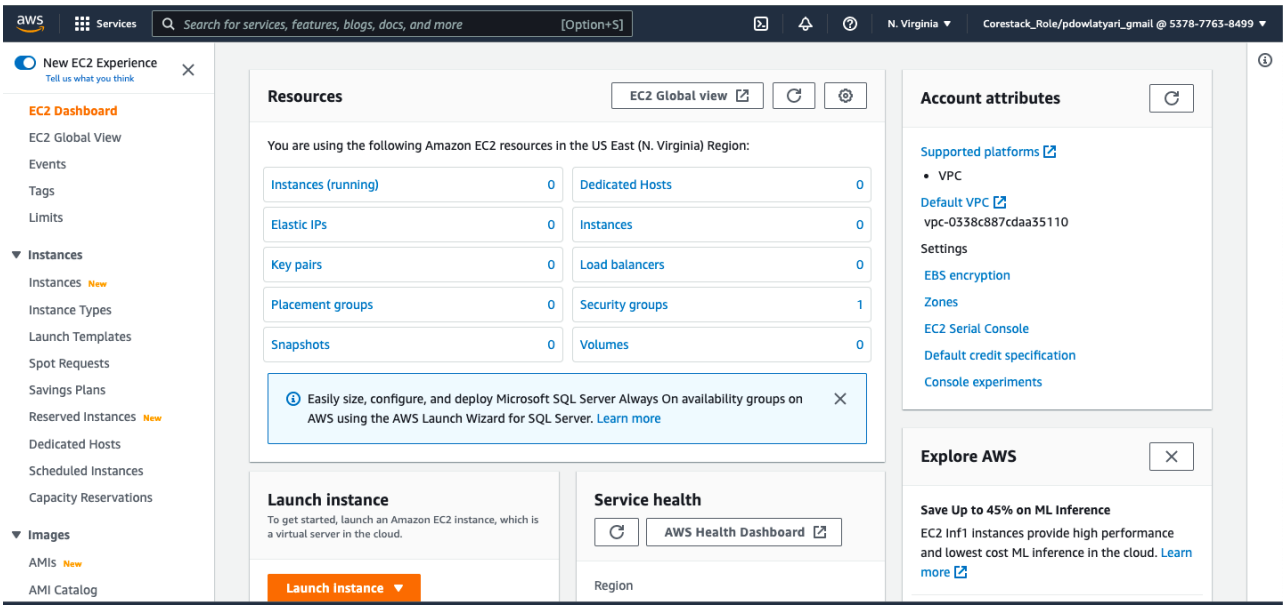
Deployment Process

AWS - EC2

After creating an AWS account and login, we need to search and open EC2. EC2 stands for Elastic Compute Cloud and is a part of Amazon.com's cloud-computing platform, Amazon Web Services, that allows users to rent virtual computers on which to run their own computer applications.

On the top right you can choose the region where you like your application to be hosted by AWS.

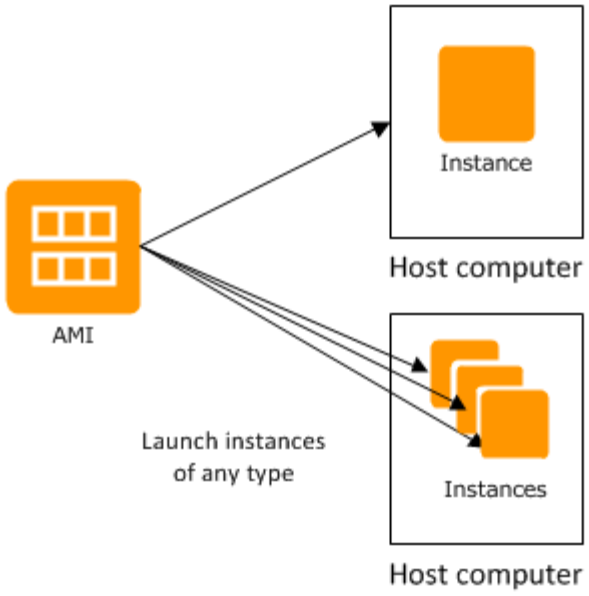
In resources, you can see if any instances are running at the moment. To launch a new instance you need to click on the launch instance button at the bottom of the page.



AMI and Instance

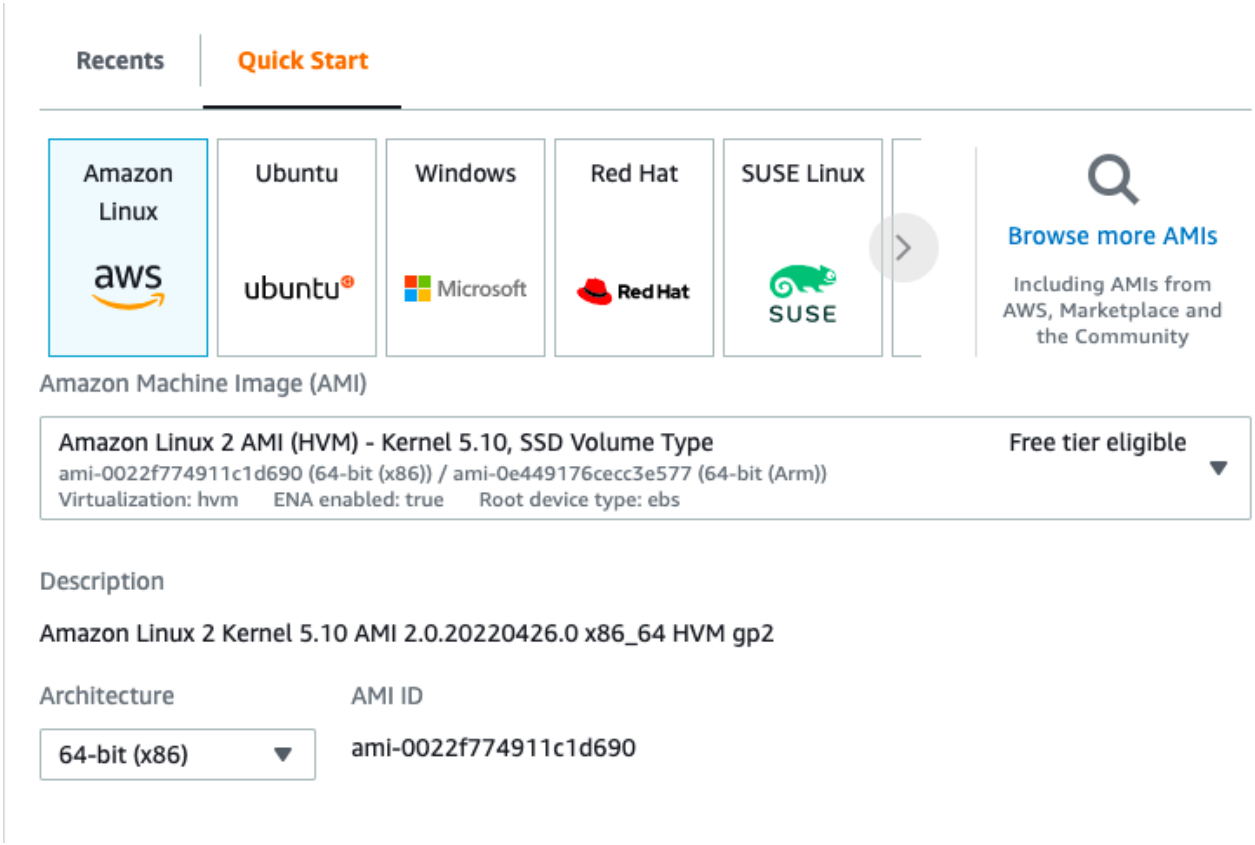
An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications).

From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud. You can launch multiple instances of an AMI, as shown in the following figure.



AMI and OS Images

For our application we choose Amazon Linux Kernel 64 bits. We can use Ubuntu or Windows as well.

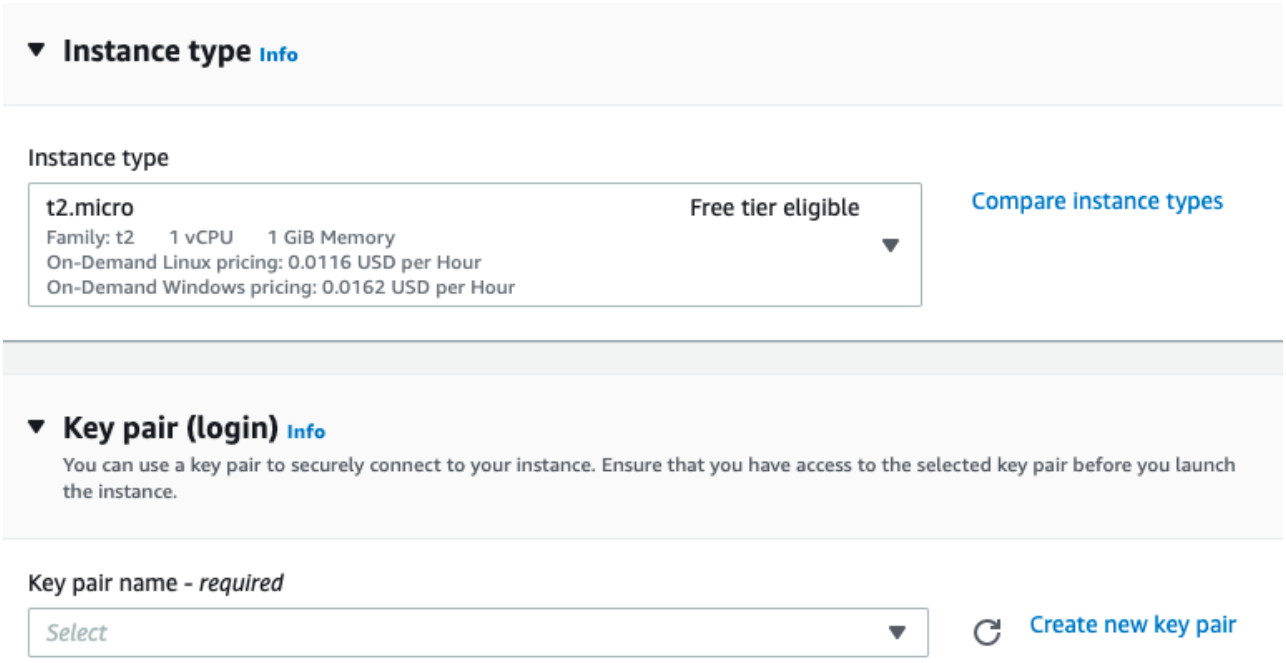


Instance Type

Each instance type offers different compute and memory capabilities. Select an instance type based on the amount of memory and computing power that you need for the application or software that you plan to run on the instance.

For this application we chose t2.micro with 1 vCPU, 1GiB memory, and other configurations as shown in the picture.

Creating a key pair is required for login remotely to the instance from our machine.

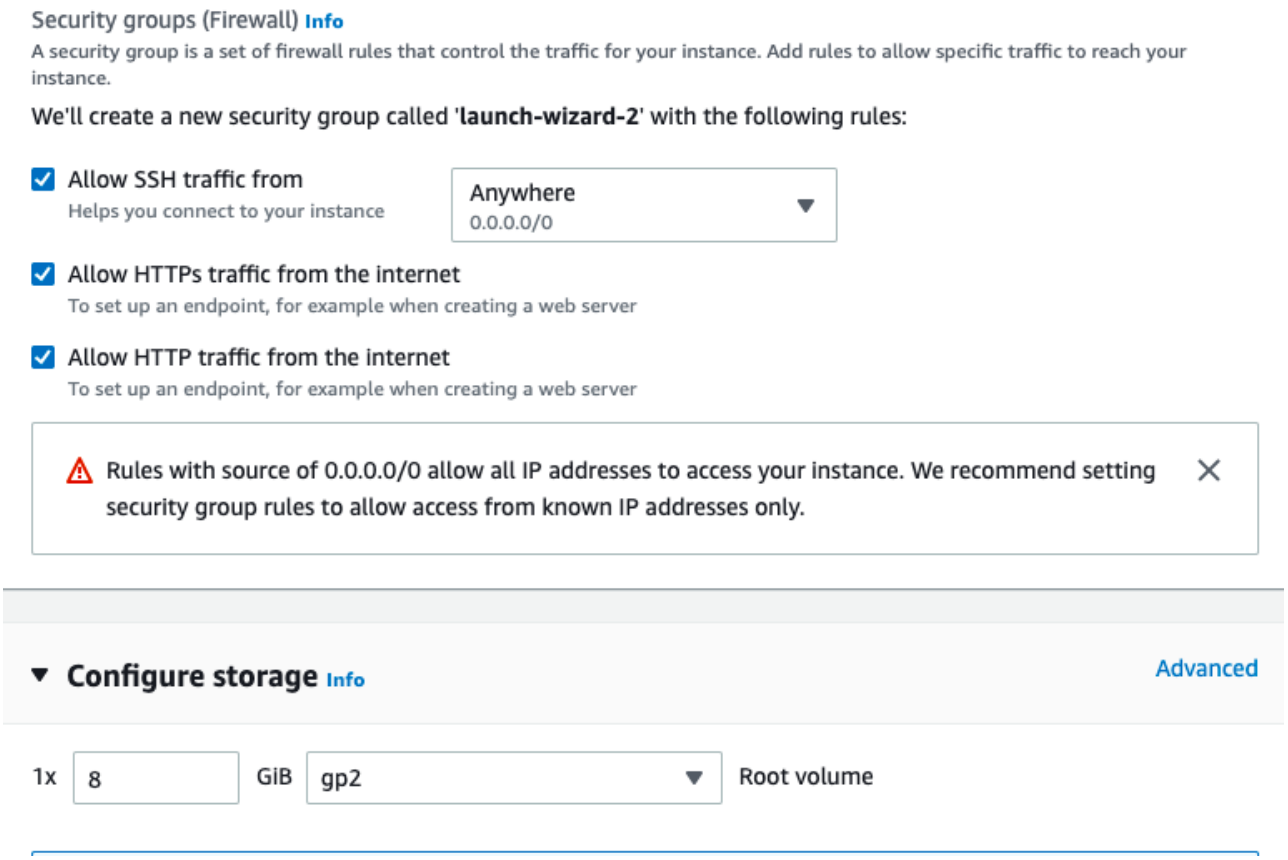


Network Settings

You can restrict access by only allowing trusted hosts or networks to access ports on your instance. For example, you can restrict SSH access by restricting incoming traffic on port 22.

For our application, we are allowing access from all protocols including SSH, HTTP, and HTTPS with the source of 0.0.0.0/0 from anywhere.

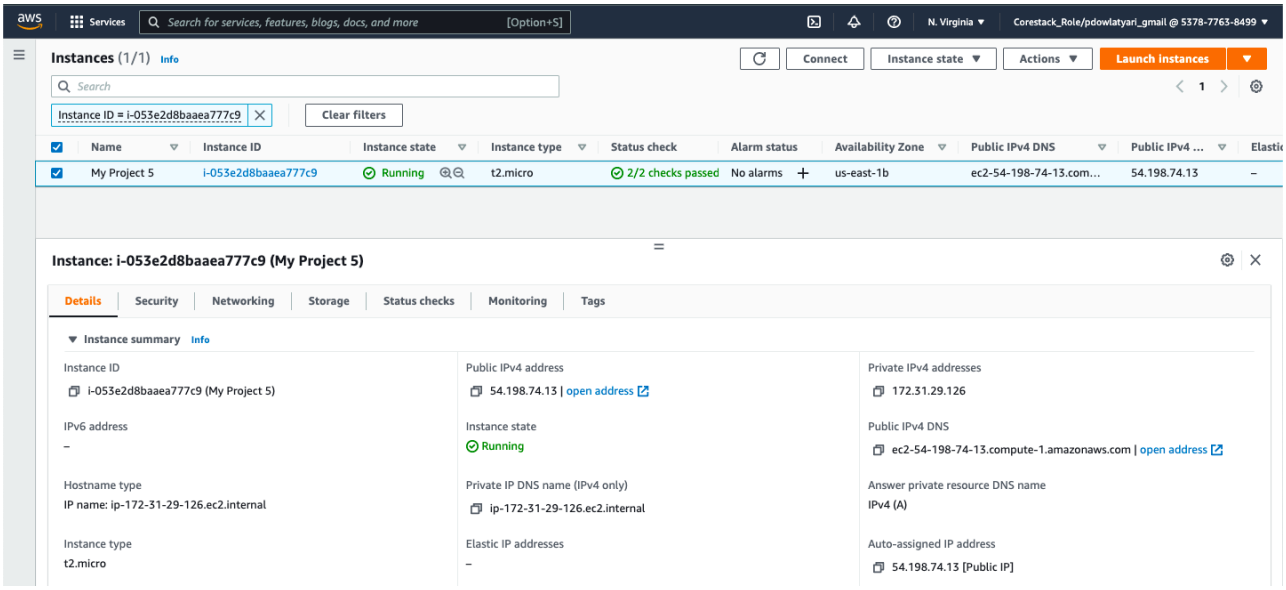
We also specify 8GB storage for start. That may increase as needed in the future.



Launch Instance

Finally, we can launch the instance. After a few minutes our instance is ready and like the following picture its status changes from pending to running.

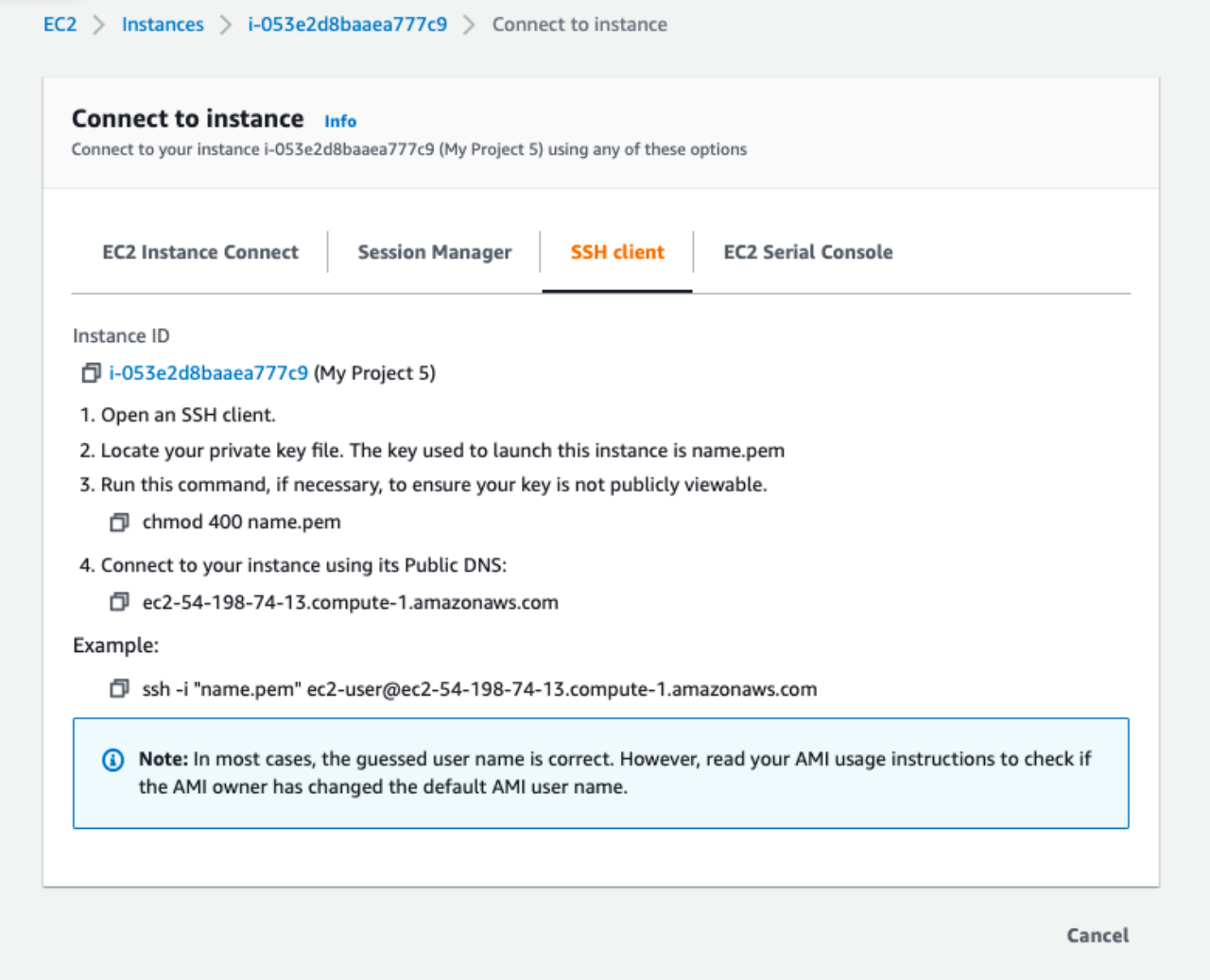
At this stage everything is set and we can easily connect to our instance using our local machine.



Connect to Instance

To connect to the instance we launched, we should select the checkmark and click on the connect button on the top of the page.

For SSH users, as we are, select SSH Client tab and use the public DNC for connecting to the instance from a local machine.



Connect Locally and Setup the Environment

First, we use **chmod 400** to change the access permissions of file system objects and give the user read permission.

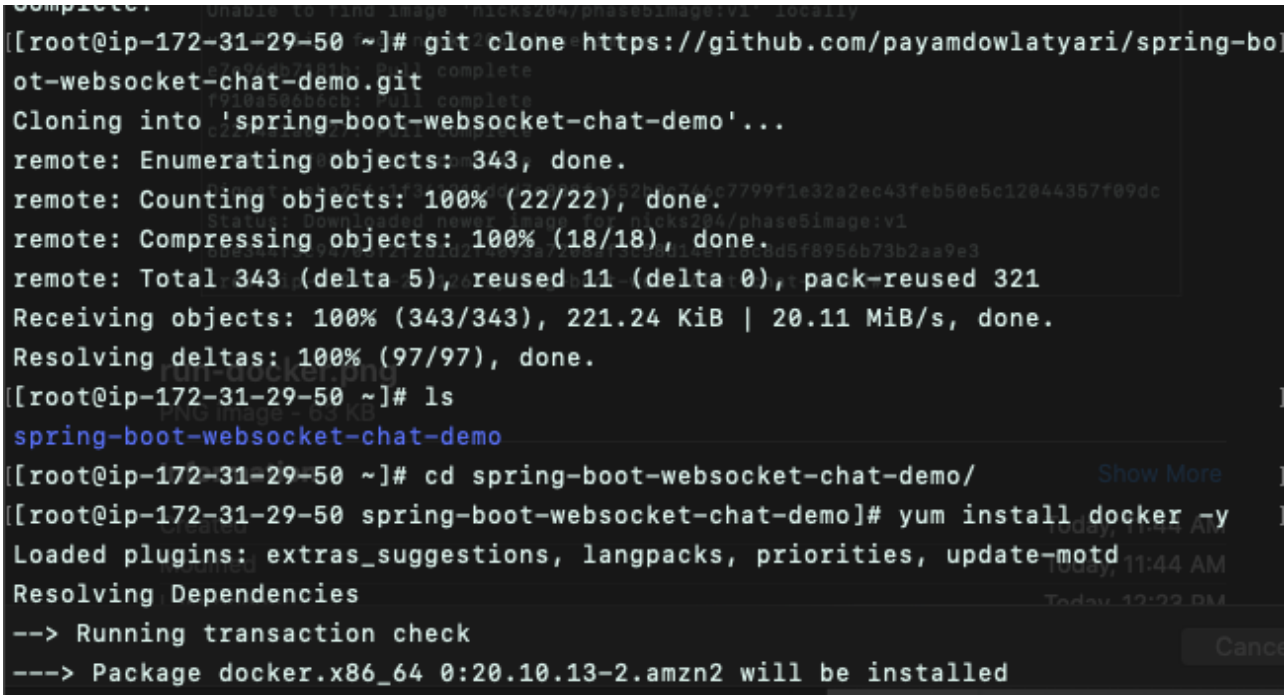
Then use the **SSH** command provided by AWS to make the connection.

```
[payamdowlatyari@Payams-MacBook-Air ~ % chmod 400 name.pem                      ]
[payamdowlatyari@Payams-MacBook-Air ~ % ssh -i "name.pem" ec2-user@ec2-54-198-74-]
13.compute-1.amazonaws.com
The authenticity of host 'ec2-54-198-74-13.compute-1.amazonaws.com (54.198.74.13
)' can't be established.
ECDSA key fingerprint is SHA256:BgplNopUC6khSpp/tFTWKjxQnTEZVRMHr5r7Gr8r3To.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-54-198-74-13.compute-1.amazonaws.com,54.198.74.1
3' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 2 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-29-126 ~]$
```

Clone the Repo and Install Maven and Docker

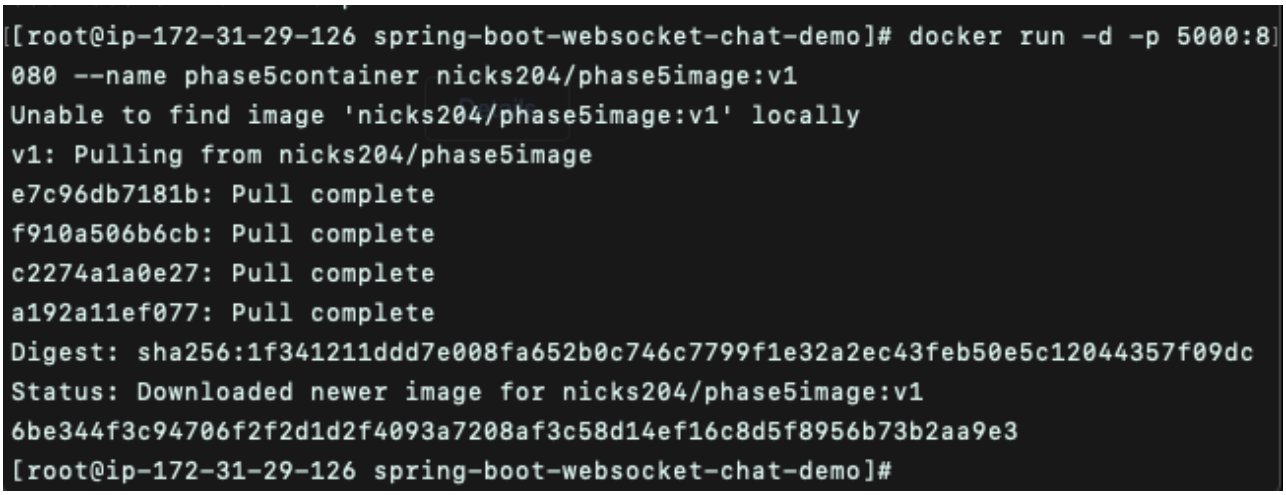
Then we should change to the root user, **update yum** and **install git** and **maven**.

The next step is to clone the git repository, **cd** to the application directory and install **docker**.



Run The Application

After that we start docker and run the application image using docker container

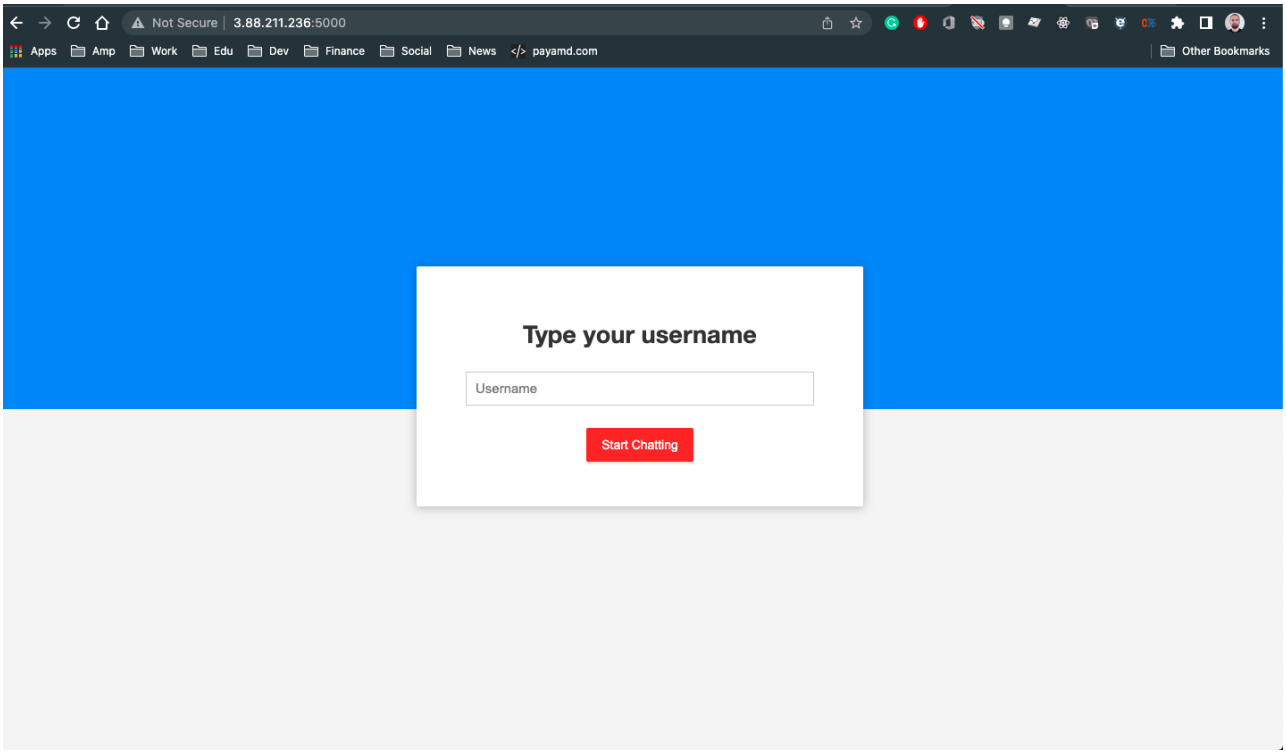


Typing **docker ps** command shows the container and application image is running through docker container

```
[root@ip-172-31-29-126 spring-boot-websocket-chat-demo]# docker images
REPOSITORY              TAG       IMAGE ID       CREATED       SIZE
nicks204/phase5image    v1        81cd88f81fd3   3 weeks ago   131MB
[root@ip-172-31-29-126 spring-boot-websocket-chat-demo]# docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED
   STATUS          PORTS                                       NAMES
6be344f3c947   nicks204/phase5image:v1   "java -Djava.securit…"   20 minutes ago
   Up 20 minutes   0.0.0.0:5000->8080/tcp, :::5000->8080/tcp   phase5container
[root@ip-172-31-29-126 spring-boot-websocket-chat-demo]#
```

Test Application

To test the application We open the web browser on our **Public IPv4** address **port 5000** which in our case it would be **http://3.88.211.236:5000/**



The final step is testing the functionality of the application.

At this stage our application is up and running.