

# **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

<b>Introduction</b>	<b>1</b>
Purpose	1
<b>Intended Audience</b>	<b>1</b>
Product Scope	1
<b>Iteration Process</b>	<b>1</b>
<b>Deployment Process</b>	<b>2</b>
<b>AWS - EC2</b>	<b>2</b>
<b>AMI and Instance</b>	<b>3</b>
<b>AMI and OS Images</b>	<b>4</b>
<b>Instance Type</b>	<b>5</b>
<b>Network Settings</b>	<b>6</b>
<b>Launch Instance</b>	<b>7</b>
<b>Connect to Instance</b>	<b>8</b>
<b>Connect Locally and Setup the Environment</b>	<b>9</b>
<b>Clone the Repo and Install Maven and Docker</b>	<b>9</b>
<b>Run The Application</b>	<b>10</b>
<b>Test Application</b>	<b>11</b>

# 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

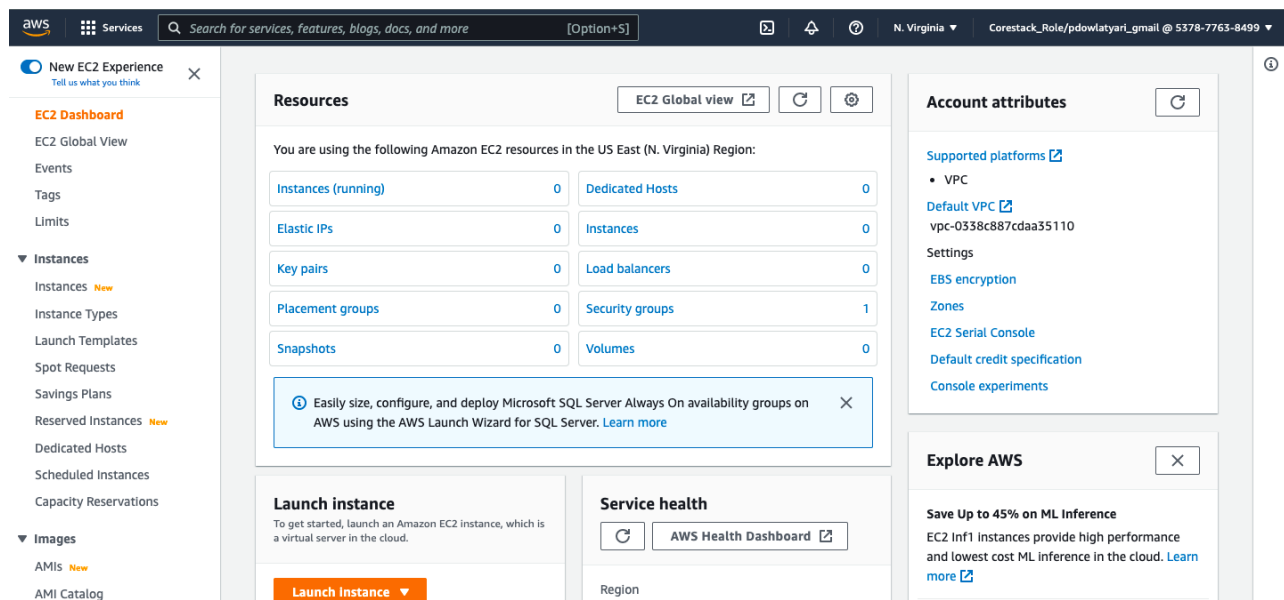
# Deployment Process

## 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.

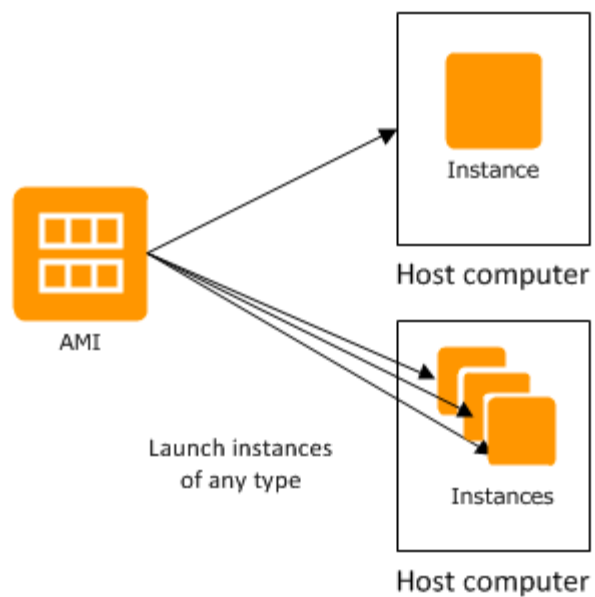
In resources, you can see if any instances are running at the moment. To launch a new instance you need to click on the launch instance button at the bottom of the page.



## 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.

Recents

Quick Start

Amazon Linux  
aws

Ubuntu  
ubuntu

Windows  
Microsoft

Red Hat  
Red Hat

SUSE Linux  
SUSE

Search

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type  
ami-0022f774911c1d690 (64-bit (x86)) / ami-0e449176cecc3e577 (64-bit (Arm))  
Virtualization: hvm    ENA enabled: true    Root device type: ebs

Free tier eligible

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20220426.0 x86\_64 HVM gp2

Architecture

AMI ID

64-bit (x86)

ami-0022f774911c1d690

## 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.

### ▼ Instance type [Info](#)

#### Instance type

**t2.micro**

Free tier eligible

[Compare instance types](#)

Family: t2 1 vCPU 1 GiB Memory

On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

### ▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

#### Key pair name - *required*

Select

[Create new key pair](#)

## 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.

### Security groups (Firewall) [Info](#)


A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

We'll create a new security group called 'launch-wizard-2' with the following rules:

- ☒ Allow SSH traffic from 

Anywhere  
0.0.0.0/0 ▼

  
Helps you connect to your instance
- ☒ Allow HTTPs traffic from the internet  
To set up an endpoint, for example when creating a web server
- ☒ Allow HTTP traffic from the internet  
To set up an endpoint, for example when creating a web server

 Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. [×](#)

### ▼ Configure storage [Info](#)

[Advanced](#)

1x  GiB  ▼ Root volume



## 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.

The screenshot displays the AWS Management Console interface. At the top, the navigation bar shows the AWS logo, 'Services', a search bar, and the user's profile. The main content area is titled 'Instances (1/1) Info'. Below this, there's a search bar and a filter for 'Instance ID = i-053e2d8baaea777c9'. A table lists the instance details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic
My Project 5	i-053e2d8baaea777c9	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-198-74-13.com...	54.198.74.13	-

Below the table, the 'Instance: i-053e2d8baaea777c9 (My Project 5)' details are shown. The 'Details' tab is active, displaying the following information:

- Instance summary**
  - Instance ID: i-053e2d8baaea777c9 (My Project 5)
  - IPv6 address: -
  - Hostname type: IP name: ip-172-31-29-126.ec2.internal
  - Instance type: t2.micro
- Public IPv4 address**: 54.198.74.13 | [open address](#)
- Instance state**: Running
- Private IP DNS name (IPv4 only)**: ip-172-31-29-126.ec2.internal
- Elastic IP addresses**: -
- Private IPv4 addresses**: 172.31.29.126
- Public IPv4 DNS**: ec2-54-198-74-13.compute-1.amazonaws.com | [open address](#)
- Answer private resource DNS name IPv4 (A)**: -
- Auto-assigned IP address**: 54.198.74.13 [Public IP]

## 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.

EC2 > Instances > i-053e2d8baaea777c9 > Connect to instance

### Connect to instance [Info](#)


Connect to your instance i-053e2d8baaea777c9 (My Project 5) using any of these options



EC2 Instance Connect

Session Manager


**SSH client**


EC2 Serial Console

Instance ID  
 i-053e2d8baaea777c9 (My Project 5)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is name.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
 `chmod 400 name.pem`
4. Connect to your instance using its Public DNS:  
 `ec2-54-198-74-13.compute-1.amazonaws.com`

Example:

 `ssh -i "name.pem" ec2-user@ec2-54-198-74-13.compute-1.amazonaws.com`

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

## 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,54.198.74.13' (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.

```
[[root@ip-172-31-29-50 ~]# git clone https://github.com/payamdowlatyari/spring-bo
ot-websocket-chat-demo.git
Cloning into 'spring-boot-websocket-chat-demo'...
remote: Enumerating objects: 343, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 343 (delta 5), reused 11 (delta 0), pack-reused 321
Receiving objects: 100% (343/343), 221.24 KiB | 20.11 MiB/s, done.
Resolving deltas: 100% (97/97), done.
[[root@ip-172-31-29-50 ~]# ls
spring-boot-websocket-chat-demo
[[root@ip-172-31-29-50 ~]# cd spring-boot-websocket-chat-demo/
[[root@ip-172-31-29-50 spring-boot-websocket-chat-demo]# yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.13-2.amzn2 will be installed
```

## Run The Application

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

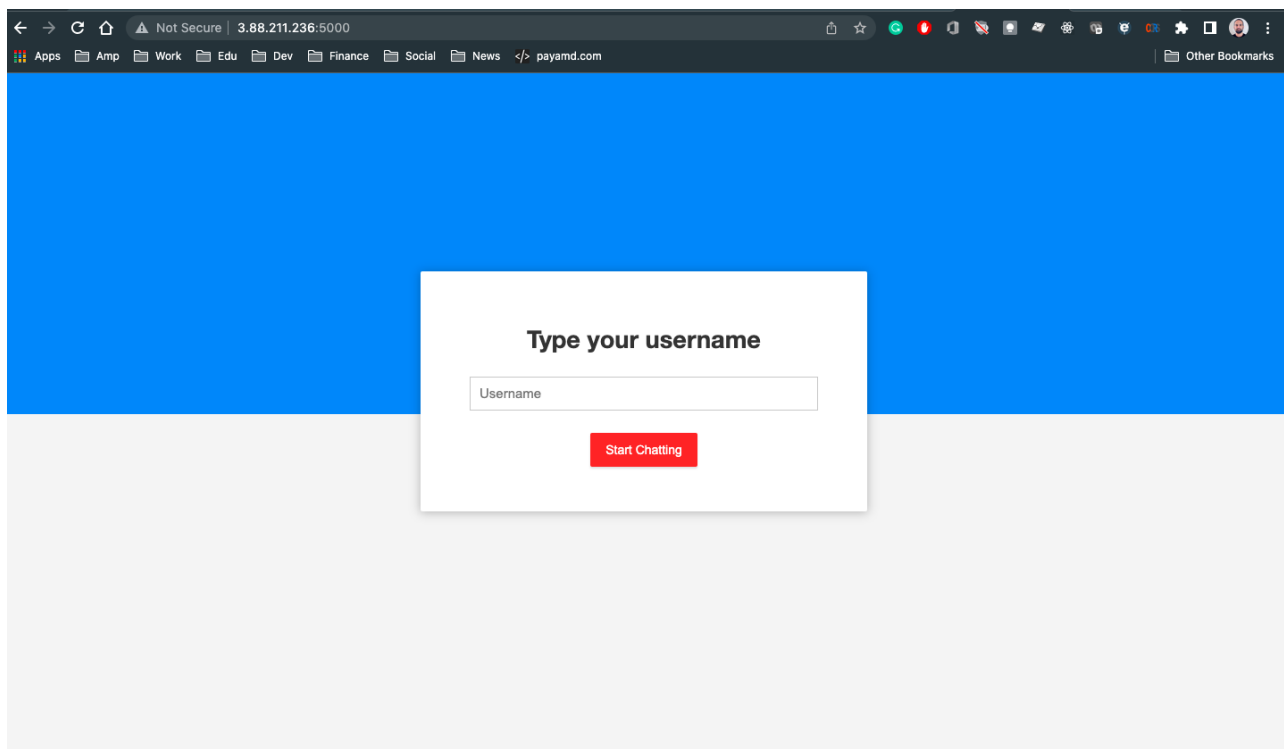
```
[[root@ip-172-31-29-126 spring-boot-websocket-chat-demo]# docker run -d -p 5000:8
080 --name phase5container nicks204/phase5image:v1
Unable to find image 'nicks204/phase5image:v1' locally
v1: Pulling from nicks204/phase5image
e7c96db7181b: Pull complete
f910a506b6cb: Pull complete
c2274a1a0e27: Pull complete
a192a11ef077: Pull complete
Digest: sha256:1f341211ddd7e008fa652b0c746c7799f1e32a2ec43feb50e5c12044357f09dc
Status: Downloaded newer image for nicks204/phase5image:v1
6be344f3c94706f2f2d1d2f4093a7208af3c58d14ef16c8d5f8956b73b2aa9e3
[[root@ip-172-31-29-126 spring-boot-websocket-chat-demo]#
```

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.

