

DevOps Certification:

Project2 - Build a Docker Jenkins Pipeline to implement CI/CD Workflow

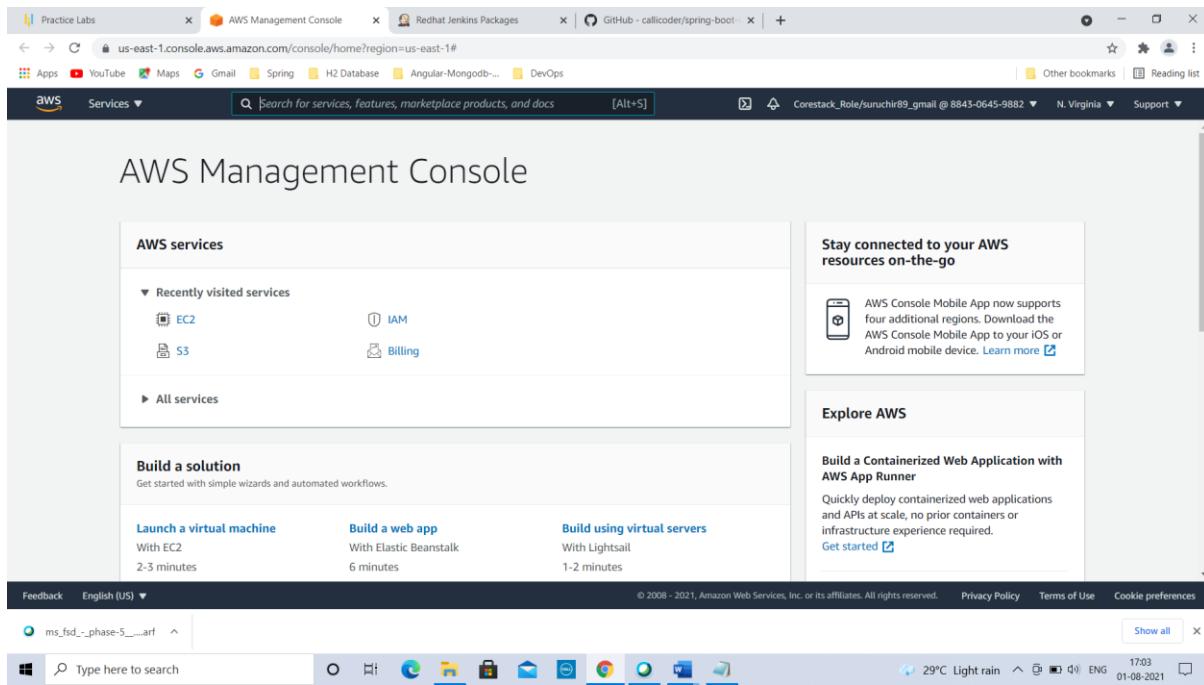
By: Suruchi S Rajguru

Github repository link: <https://github.com/SuruchiRajguru/Suruchi-DevOpsCertification.git>

Steps to automate the integration & deployment of the web application are as follows.

Login to LMS Account → Go to Practice Labs → AWS → Launch lab → AWS Web Console → Auth Url

AWS Management Console will appear on screen →



Go to AWS Services → Click on EC2 →

The screenshot shows the AWS EC2 Management Dashboard. On the left, there's a sidebar with 'EC2 Dashboard' selected. The main area displays 'Resources' with a summary of current usage across various services. A callout box highlights a note about launching Microsoft SQL Server Always On availability groups. To the right, there are sections for 'Account attributes' and 'Explore AWS'. The status bar at the bottom shows the date as 01-08-2021.

Scroll down & go to Launch instance→

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

This screenshot is similar to the one above, but it shows a dropdown menu open over the 'Launch instance' button. The dropdown contains options like 'Amazon Linux 2 x86_64 (ami-0a5a507c405cbe98d)' and 'Ubuntu Server 20.04 LTS (ami-0a5a507c405cbe98d)'. The rest of the dashboard and status bar are visible.

Click on Launch instance dropdown-->click on Launch instance.

Step 1: Choose an Amazon Machine Image (AMI)

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Free tier only

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0c2b8ca1dad447f8a (64-bit x86) / ami-06cf15d6d096df5d2 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

macOS Big Sur 11.4 - ami-059ff882c04ebcd21

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

Feedback English (US) ▾

ms_fsd_...phase-5_...arf

Type here to search

29°C Light rain 17:06 01-08-2021

Select Amazon Linux 2 AMI(HVM), SSD Volume Type→

Step 2: Choose an Instance Type

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

Feedback English (US) ▾

ms_fsd_...phase-5_...arf

Type here to search

29°C Light rain 17:07 01-08-2021

Select t2.micro (Free tier eligible) type → Click on Next: Configure Instance Details→

Step 3: Configure Instance Details

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1 [Launch into Auto Scaling Group](#)

Purchasing option: Request Spot Instances

Network: vpc-08a5507c405cbbe98d (default) [Create new VPC](#)

Subnet: No preference (default subnet in any Availability Zone) [Create new subnet](#)

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: Add instance to placement group

Capacity Reservation: Open

Domain join directory: No directory [Create new directory](#)

IAM role: None [Create new IAM role](#)

Shutdown behavior: Stop

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

Click on Next: Add storage→

Step 4: Add Storage

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-090e9376979c86d7b	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Click on Next: Add Tags→

Step 5: Add Tags

This resource currently has no tags

Choose the Add tag button or click to add a Name tag.
Make sure your IAM policy includes permissions to create tags.

Add Tag (Up to 50 tags maximum)

Cancel Previous Review and Launch Next: Configure Security Group

Click on Next: Configure Security Group→

Step 6: Configure Security Group

Click on Add Rule button

Set Type from SSH to All Traffic

Set Source from custom to Anywhere

Assign a security group: Create a new security group
 Select an existing security group

Security group name: launch-wizard-1

Description: launch-wizard-1 created 2021-08-01T17:11:36.017+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom CIDR, IP or Security Group	e.g. SSH for Admin Desktop
All traffic	All	0 - 65535	Anywhere	e.g. SSH for Admin Desktop

Add Rule

Warning
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.

Cancel Previous Review and Launch

Click on Review and Launch→

Step 7: Review Instance Launch

The screenshot shows the AWS Launch Instance Wizard Step 7: Review Instance Launch page. The page displays the configuration details for launching an instance, including the AMI, instance type, and security groups.

AMI Details: Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0c2b8ca1dad447f8a (Free tier eligible). It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a... Root Device Type: ebs Virtualization type: hvm.

Instance Type: t2.micro (Edit instance type)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups: launch-wizard-1 (Edit security groups)

Security group name	Description
launch-wizard-1	launched 2021-08-01T17:11:36.017+05:30

Buttons: Cancel, Previous, Launch

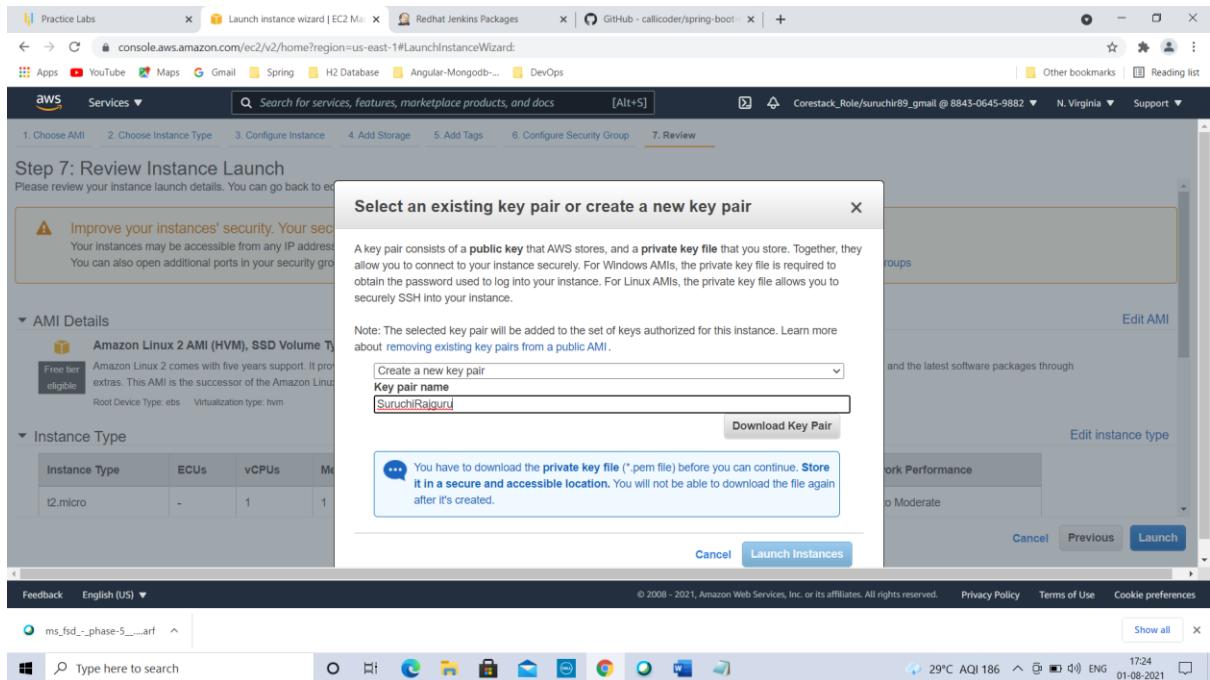
Feedback: English (US) ▾

System Status: 29°C AQI 186 17:23 01-08-2021

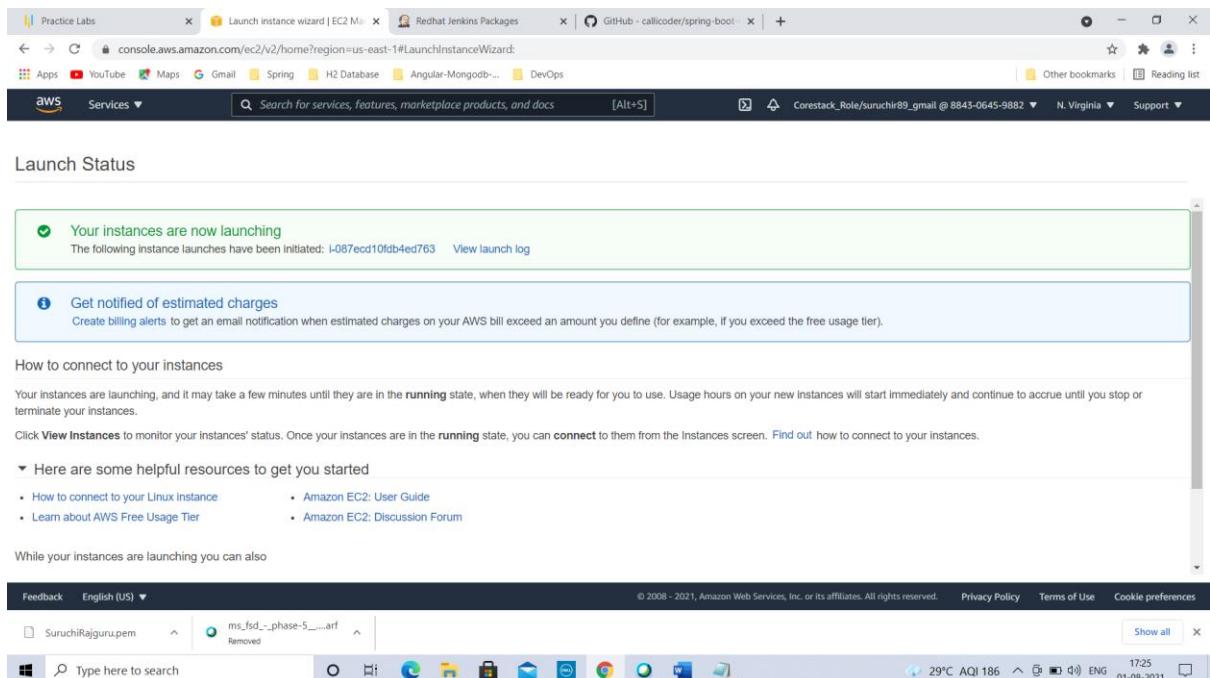
Review all steps by scrolling down & then Click on Launch→

Select an existing key pair or create new key pair

Select existing key pair or create new key pair from drop down box→give key pair name→Download key pair before clicking on Launch instance→key will be stored in your System at appropriate location.(Default Downloads folder)



Click on Launch Instances→



Click on instance id→

You are on EC2 Instance now→

Check Instances(1)—one instance is created →check instance state as running→

Screenshot of the AWS Management Console showing the EC2 Instances page. A search bar at the top contains the instance ID "i-087ecd10fdb4ed763". The main table shows one instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-087ecd10fdb4ed763	Running	t2.micro	Initializing	No alarms	us-east-1e	ec2-18-207-236-

A modal window titled "Select an instance above" is open, prompting the user to click on the instance row.

Click on Instance ID → Instance summary for created instance is visible here → Click on connect →

Screenshot of the AWS Management Console showing the Instance details page for instance "i-08c42491125b1df9a". The "Connect" button is highlighted.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-08c42491125b1df9a	34.202.234.56 open address	172.31.52.102

The "Connect" button is located at the top right of the instance details page.

Go to SSH client→

Connect to instance [Info](#)
Connect to your instance i-087ecd10fdb4ed763 using any of these options

EC2 Instance Connect | Session Manager | **SSH client** | EC2 Serial Console

Instance ID
i-087ecd10fdb4ed763

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is SuruchiRajguru.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 SuruchiRajguru.pem`
4. Connect to your instance using its Public DNS:
 `ec2-18-207-236-30.compute-1.amazonaws.com`

Example:
 `ssh -i "SuruchiRajguru.pem" ec2-user@ec2-18-207-236-30.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.

Copy ssh Key→

Open command prompt from location where .pem file is stored(Go to downloads folder & from there open command prompt)→copy ssh key & press enter→

You are connected to the EC2 Instance now.

Click on EC2 Instance Connect

Connect to instance [Info](#)
Connect to your instance i-087ecd10fdb4ed763 using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 Serial Console

Instance ID
i-087ecd10fdb4ed763

Public IP address
18.207.236.30

User name
ec2-user

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

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.

Click on connect

You are on EC2 instance now

```
Last login: Mon Aug 2 10:58:57 2021 from ec2-18-206-107-25.compute-1.amazonaws.com
[ec2-user@ip-172-31-52-102 ~]$
```

i-08c42491125b1df9a

Public IPs: 34.202.234.56 Private IPs: 172.31.52.102



To become a root user, type command in command prompt:

`sudo -i` you will become a root user now

Execute following commands

- 1) `yum update -y` updates packages
To install git, type:
- 2) `yum install git -y` success message: complete

To install maven, type:

- 3) `yum install maven -y` success message: complete

To check Maven version, type:

- 4) `mvn -version` displays Maven Version, Path

To clone the repository from Github, type:

- 5) `git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git`

To list files or directories, type:

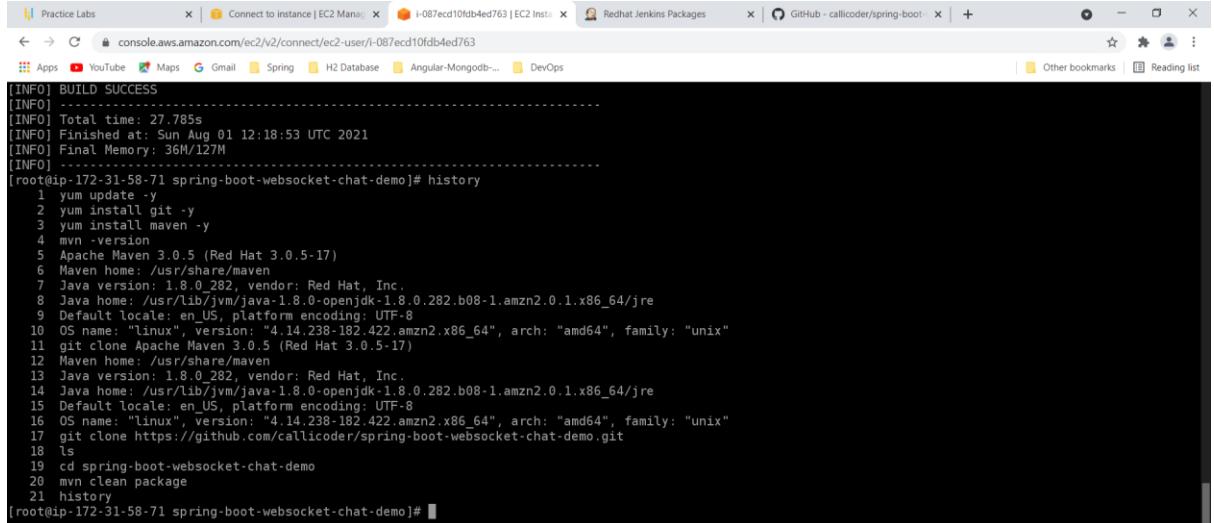
- 6) `ls` displays spring-boot-websocket-chat-demo inside it

Move into project `spring-boot-websocket-chat-demo` using,

- 7) `cd spring-boot-websocket-chat-demo`

To clean the package & load the project

8) mvn clean package cleans the package & loads the project



```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 27.785s
[INFO] Finished at: Sun Aug 01 12:18:53 UTC 2021
[INFO] Final Memory: 36M/127M
[INFO]
[INFO]
[root@ip-172-31-58-71 spring-boot-websocket-chat-demo]# history
  1 yum update -y
  2 yum install git -y
  3 yum install maven -y
  4 mvn -version
  5 Apache Maven 3.0.5 (Red Hat 3.0.5-17)
  6 Maven home: /usr/share/maven
  7 Java version: 1.8.0_282, vendor: Red Hat, Inc.
  8 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
  9 Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
11 git clone Apache Maven 3.0.5 (Red Hat 3.0.5-17)
12 Maven home: /usr/share/maven
13 Java version: 1.8.0_282, vendor: Red Hat, Inc.
14 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
15 Default locale: en_US, platform encoding: UTF-8
16 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
17 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git
18 ls
19 cd spring-boot-websocket-chat-demo
20 mvn clean package
21 history
[root@ip-172-31-58-71 spring-boot-websocket-chat-demo]#
```

i-087ecd10fdb4ed763
Public IPs: 18.207.236.30 Private IPs: 172.31.58.71



To install docker, type:

9) yum install docker -y complete

To start the docker, type:

10) systemctl start docker

To check docker images,

11) docker images

To build new docker image, type:

12) docker build -t springbootimage .

To check image creation, again type:

13) docker images

```

Step 2/7 : MAINTAINER Rajeev Kumar Singh <callicoder@gmail.com>
--> Running in e3207a912a36
Removing intermediate container e3207a912a36
--> 0e6ff3c2c0f1
Step 3/7 : VOLUME /tmp
--> Running in d2cdelbcc1d6
Removing intermediate container d2cdelbcc1d6
--> 6554fde93f05
Step 4/7 : EXPOSE 8080
--> Running in 3de2efeb348
Removing intermediate container 3de2efeb348
--> bcce81f498f9
Step 5/7 : ARG JAR_FILE=target/websocket-demo-0.0.1-SNAPSHOT.jar
--> Running in 9d056ab91cf7
Removing intermediate container 9d056ab91cf7
--> 79945df531e2
Step 6/7 : ADD ${JAR_FILE} websocket-demo.jar
--> 9c611d445e1b
Step 7/7 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/urandom","-jar","/websocket-demo.jar"]
--> Running in da7a7c7ea2da
Removing intermediate container da7a7c7ea2da
--> 223f687bb698
Successfully built 223f687bb698
Successfully tagged springbootimage:latest
[root@ip-172-31-58-71 spring-boot-websocket-chat-demo]# docker images
REPOSITORY          TAG           IMAGE ID            CREATED             SIZE
springbootimage     latest        223f687bb698   10 seconds ago   131MB
openjdk              8-jdk-alpine  a3562aa0b991   2 years ago      105MB
[root@ip-172-31-58-71 spring-boot-websocket-chat-demo]#

```

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71



```

2021-08-02 17:06:05.939 INFO 1 --- [extShutdownHook] c.e.w.controller.WebSocketEventListener : User Disconnected : Suruchi
2021-08-02 17:06:05.944 INFO 1 --- [extShutdownHook] o.s.m.s.b.SimpleBrokerMessageHandler : Stopping...
2021-08-02 17:06:05.945 INFO 1 --- [extShutdownHook] o.s.m.s.b.SimpleBrokerMessageHandler : BrokerAvailabilityEvent[available=false, SimpleBrokerMessageHandler [DefaultSubscriptionRegistry[cache[0 destination(s)], registry[0 sessions]]]]
2021-08-02 17:06:05.945 INFO 1 --- [extShutdownHook] o.s.m.s.b.SimpleBrokerMessageHandler : Stopped.
2021-08-02 17:06:05.963 INFO 1 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'brokerChannelExecutor'
2021-08-02 17:06:05.965 INFO 1 --- [extShutdownHook] o.s.s.c.ThreadPoolTaskScheduler : Shutting down ExecutorService 'messageBrokerTaskScheduler'
2021-08-02 17:06:05.966 INFO 1 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'clientOutboundChannelExecutor'
2021-08-02 17:06:05.967 INFO 1 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'clientInboundChannelExecutor'
[root@ip-172-31-16-245 spring-boot-websocket-chat-demo]# history
1 yum update -y
2 yum install git -y
3 yum install maven -y
4 mvn -version
5 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo
6 ls
7 cd spring-boot-websocket-chat-demo
8 mvn clean package
9 yum install docker -y
10 systemctl start docker
11 docker images
12 docker build -t springbootimage .
13 docker images
14 docker run -p 8085:8080 --name springcontainer springbootimage
15 history
[root@ip-172-31-16-245 spring-boot-websocket-chat-demo]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
943786cceda4        springbootimage   "java -Djava.secure..."   4 minutes ago       Exited (130) About a minute ago   springcontainer
[root@ip-172-31-16-245 spring-boot-websocket-chat-demo]#

```

i-0904f37bfffef48d24

Public IPs: 3.87.155.138 Private IPs: 172.31.16.245



springbootimage is created →

```
Last login: Sun Aug 1 12:03:07 2021 from ec2-18-206-107-26.compute-1.amazonaws.com
[ec2-user@ip-172-31-58-71 ~]$ sudo -i
[root@ip-172-31-58-71 ~]# history
1  yum update -y
2  yum install git -y
3  yum install maven -y
4  mvn -version
5  Apache Maven 3.0.5 (Red Hat 3.0.5-17)
6  Maven home: /usr/share/maven
7  Java version: 1.8.0_282, vendor: Red Hat, Inc.
8  Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
9  Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
11 git clone Apache Maven 3.0.5 (Red Hat 3.0.5-17)
12 Maven home: /usr/share/maven
13 Java version: 1.8.0_282, vendor: Red Hat, Inc.
14 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
15 Default locale: en_US, platform encoding: UTF-8
16 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
17 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git
18 ls
19 cd spring-boot-websocket-chat-demo
20 mvn clean package
```

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71

```
3  yum install maven -y
4  mvn -version
5  Apache Maven 3.0.5 (Red Hat 3.0.5-17)
6  Maven home: /usr/share/maven
7  Java version: 1.8.0_282, vendor: Red Hat, Inc.
8  Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
9  Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
11 git clone Apache Maven 3.0.5 (Red Hat 3.0.5-17)
12 Maven home: /usr/share/maven
13 Java version: 1.8.0_282, vendor: Red Hat, Inc.
14 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
15 Default locale: en_US, platform encoding: UTF-8
16 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
17 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git
18 ls
19 cd spring-boot-websocket-chat-demo
20 mvn clean package
21 history
22 yum install docker -y
23 systemctl start docker
24 docker images
25 docker build springbootimage .
26 docker images
27 docker build -t springbootimage .
28 docker images
29 docker run -p 8085:8080 --name springcontainer springbootimage
30 history
```

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71

```
3  yum install maven -y
4  mvn -version
5  Apache Maven 3.0.5 (Red Hat 3.0.5-17)
6  Maven home: /usr/share/maven
7  Java version: 1.8.0_282, vendor: Red Hat, Inc.
8  Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
9  Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
11 git clone Apache Maven 3.0.5 (Red Hat 3.0.5-17)
12 Maven home: /usr/share/maven
13 Java version: 1.8.0_282, vendor: Red Hat, Inc.
14 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
15 Default locale: en_US, platform encoding: UTF-8
16 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
17 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git
18 ls
19 cd spring-boot-websocket-chat-demo
20 mvn clean package
21 history
22 yum install docker -y
23 systemctl start docker
24 docker images
25 docker build springbootimage .
26 docker images
27 docker build -t springbootimage .
28 docker images
29 docker run -p 8085:8080 --name springcontainer springbootimage
30 history
```

To create the container from the image

14) docker run -p 8085:8080 - -name springcontainer1 springbootimage

```

2021-08-01 12:43:18.528 INFO 1 --- [main] c.e.w.WebsocketDemoApplication : Starting WebsocketDemoApplication v0.0.1-SNAPSHOT on 6c4637f
2931 with PID 1 (/websocket-demo.jar started by root in /)
2021-08-01 12:43:18.537 INFO 1 --- [main] c.e.w.WebsocketDemoApplication : No active profile set, falling back to default profiles: defa
ult
2021-08-01 12:43:22.141 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-08-01 12:43:22.180 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-08-01 12:43:22.182 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Server engine: [Apache Tomcat/9.0.27]
2021-08-01 12:43:22.363 INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-08-01 12:43:22.370 INFO 1 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 3678
ms
2021-08-01 12:43:23.276 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'clientInboundChannelExecutor'
2021-08-01 12:43:23.286 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'clientOutboundChannelExecutor'
2021-08-01 12:43:23.349 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskScheduler : Initializing ExecutorService 'messageBrokerTaskScheduler'
2021-08-01 12:43:23.492 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'brokerChannelExecutor'
2021-08-01 12:43:25.152 INFO 1 --- [main] o.s.b.a.w.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
2021-08-01 12:43:25.601 INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Starting...
2021-08-01 12:43:25.606 INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : BrokerAvailabilityEvent[available=true, SimpleBrokerMessageHa
ndler [DefaultSubscriptionRegistry[cache[0 destination(s)], registry[0 sessions]]]]
2021-08-01 12:43:25.607 INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler : Started.
2021-08-01 12:43:25.699 INFO 1 --- [main] o.s.w.b.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-08-01 12:43:25.706 INFO 1 --- [main] c.e.w.WebsocketDemoApplication : Started WebsocketDemoApplication in 8.305 seconds (JVM runnin
g for 9.498)
2021-08-01 12:43:39.278 INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost]. [/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-08-01 12:43:39.283 INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-08-01 12:43:39.316 INFO 1 --- [nio-8080-exec-1] o.s.w.servlet.DispatcherServlet : Completed initialization in 32 ms
2021-08-01 12:44:15.337 INFO 1 --- [tboundChannel-1] c.e.w.controller.WebSocketEventListener : Received a new web socket connection
2021-08-01 12:44:23.491 INFO 1 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[1 current WS(1)-HttpStream(0)-HttpPoll(0), 1
total, 0 closed abnormally (0 connect failure, 0 send limit, 0 transport error)], stompSubProtocol[processed CONNECT(1)-CONNECTED(1)-DISCONNECT(0)], stompBro

```

i-087ecd10fdb4ed763
Public IPs: 18.207.236.30 Private IPs: 172.31.58.71

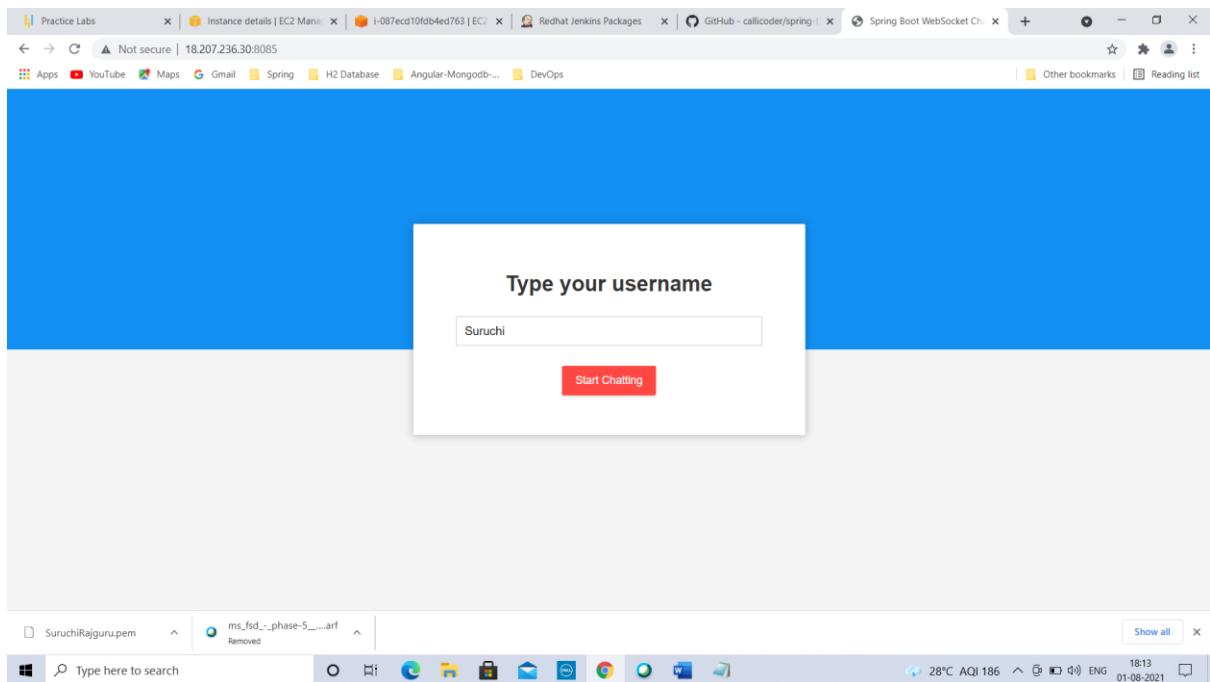
Go to Instance summary & check Public IPv4 address

Instance ID	Public IPv4 address	Private IPv4 addresses
i-087ecd10fdb4ed763	18.207.236.30 open address	172.31.58.71
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-18-207-236-30.compute-1.amazonaws.com open address
Private IPv4 DNS	Instance type	Elastic IP addresses
ip-172-31-58-71.ec2.internal	t2.micro	-
VPC ID	AWS Compute Optimizer finding	IAM Role
vpc-08a5507c405cbe98d	User: arnawssts:884306459882:assumed-role/Corestack_Role/suruchi89_gmail is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: * with an explicit deny	-
Subnet ID	Retry	
subnet-0dd6d3c538b4e49a0		

copy address & paste in browser URL followed by colon & port on which you want to run the container

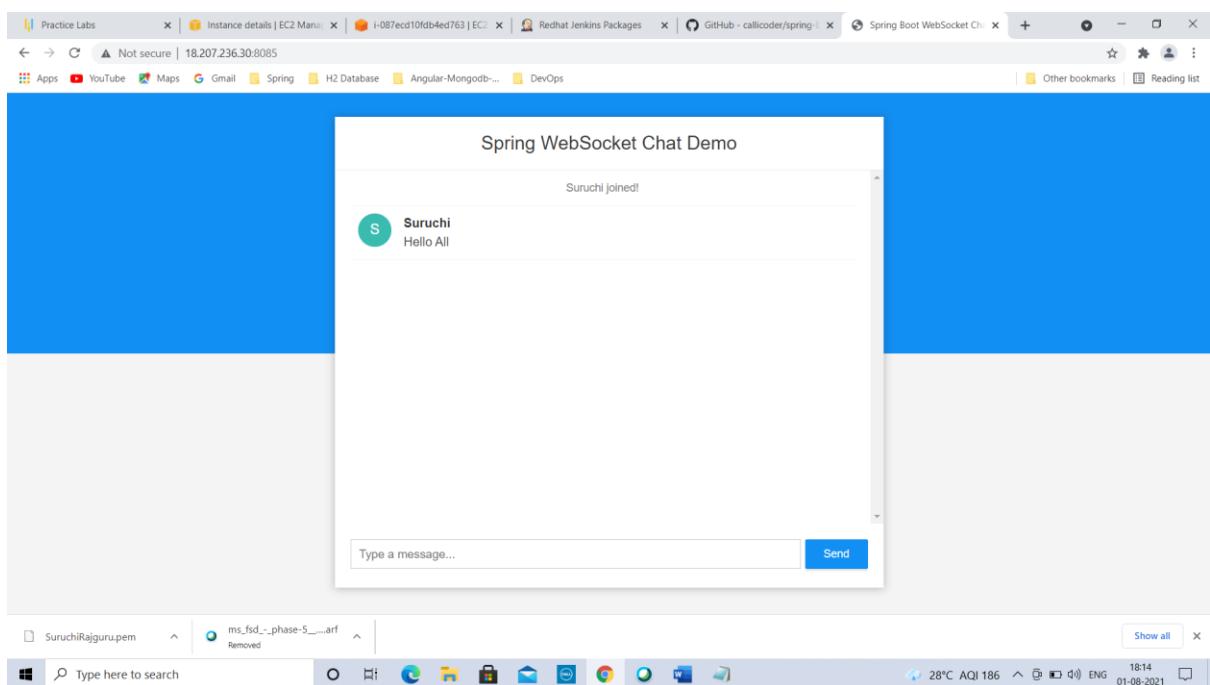
Here it is: 18.207.236.30:8085

Press enter...page is loaded



Type your username in the textbox & click on Start chatting button.

Enter your message & press send button.



=====

To open Jenkins website in new browser, type: <https://pkg.jenkins.io/redhat-stable/>

To install Jenkins on EC2 instance, copy following 3 commands & paste inside our instance turn by turn

To get right package to install Jenkins on instance

1) sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhat-stable/jenkins.repo>
(wget means getting something from internet)

To set the key, installing Jenkins as a service

2) sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>

```
Practice Labs | Instance details | EC2 Manu... | I-087ecd10fdb4ed763 | EC2 | Redhat Jenkins Packages | GitHub - callicoder/spring- | Spring Boot WebSocket Ch... | + | - | _ | Practice Labs | Instance details | EC2 Manu... | I-087ecd10fdb4ed763 | EC2 | Redhat Jenkins Packages | GitHub - callicoder/spring- | Spring Boot WebSocket Ch... | + | - | _ | console.aws.amazon.com/ec2/v2/connect/ec2-user/i-087ecd10fdb4ed763

Apps YouTube Maps Gmail Spring H2 Database Angular-Mongodb... DevOps Other bookmarks Reading list

20 mvn clean package
21 history
22 yum install docker -y
23 systemctl start docker
24 docker images
25 docker build springbootimage .
26 docker images
27 docker build -t springbootimage .
28 docker images
29 docker run -p 8085:8080 --name springcontainer springbootimage
30 history
31 docker run -p 8086:8080 --name springcontainer1 springbootimage
32 docker run -p 8085:8080 --name springcontainer1 springbootimage
33 history
[root@ip-172-31-58-71 ~]# sudo -i
[root@ip-172-31-58-71 ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2021-08-01 12:55:06-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.250.133, 2a04:4e42:2f::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.250.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[=====] 85      ...-K/s    in 0s

2021-08-01 12:55:06 (3.94 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[root@ip-172-31-58-71 ~]# sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
[root@ip-172-31-58-71 ~]# 
```

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71



3) yum install Jenkins

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71



To check history, type: history

```
9 Default locale: en_US, platform encoding: UTF-8
10 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
11 git clone Apache Maven 3.0.5 (Red Hat 3.0.5-17)
12 Maven home: /usr/share/maven
13 Java version: 1.8.0_282, vendor: Red Hat, Inc.
14 Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.282.b08-1.amzn2.0.1.x86_64/jre
15 Default locale: en_US, platform encoding: UTF-8
16 OS name: "linux", version: "4.14.238-182.422.amzn2.x86_64", arch: "amd64", family: "unix"
17 git clone https://github.com/callicoder/spring-boot-websocket-chat-demo.git
18 ls
19 cd spring-boot-websocket-chat-demo
20 mvn clean package
21 history
22 yum install docker -y
23 systemctl start docker
24 docker images
25 docker build springbootimage .
26 docker images
27 docker build -t springbootimage .
28 docker images
29 docker run -p 8085:8080 --name springcontainer springbootimage
30 history
31 docker run -p 8086:8080 --name springcontainer1 springbootimage
32 docker run -p 8085:8080 --name springcontainer1 springbootimage
33 sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
34 sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
35 yum install jenkins
36 history
[root@ip-172-31-58-71 ~]#
```

i-087ecd10fdb4ed763

Public IPs: 18.207.236.30 Private IPs: 172.31.58.71

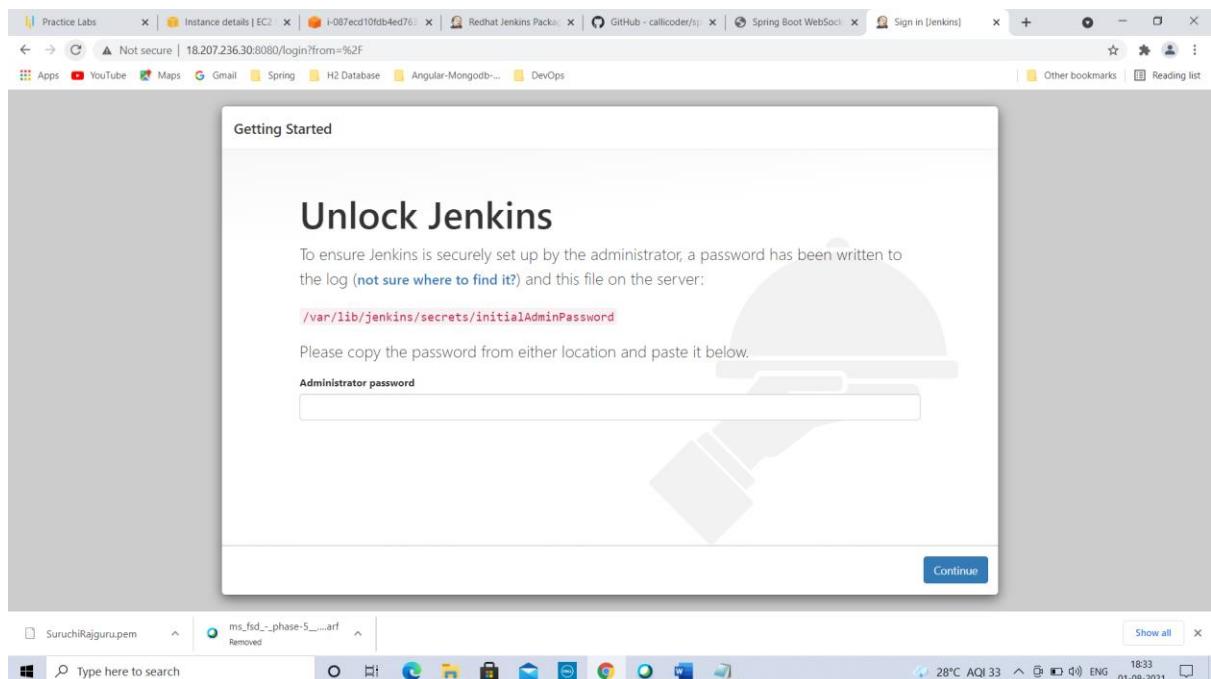


Open new browser tab → Copy ip address of your machine followed by colon & of default port of Jenkins 8080

After installing service, start Jenkins by typing :

systemctl start Jenkins

Refresh the browser url, it displays getting started page

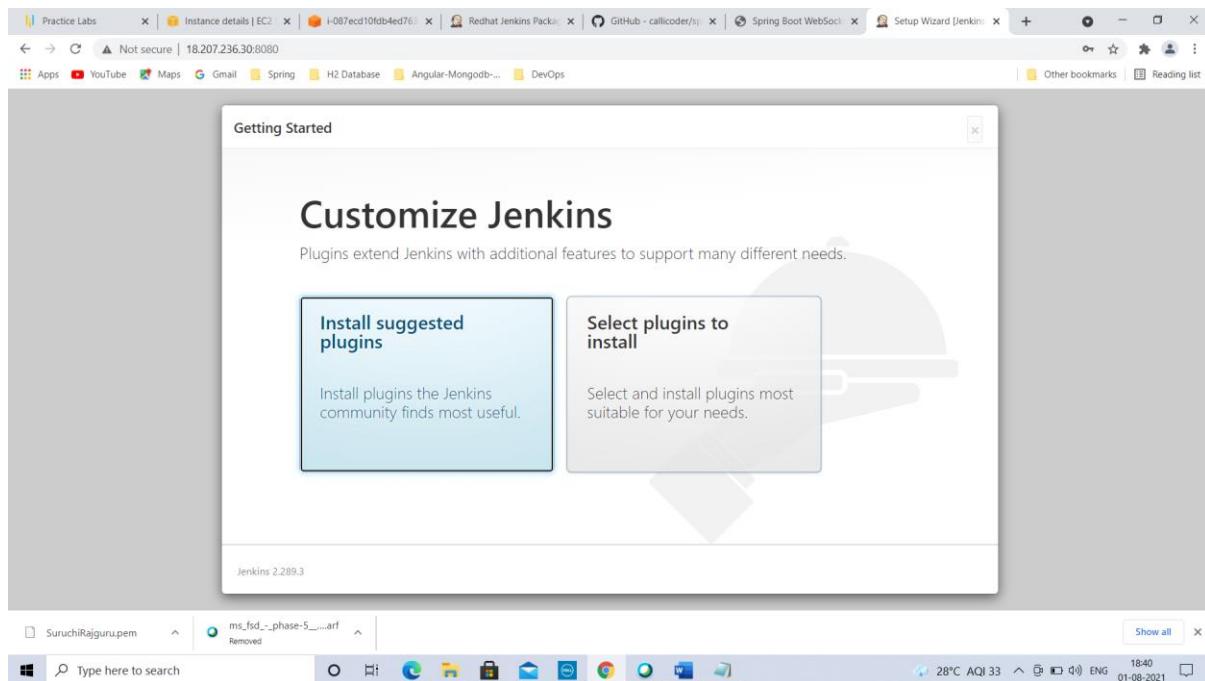


```
copy path from browser:  
/var/lib/Jenkins/secrets/initialAdminPassword
```

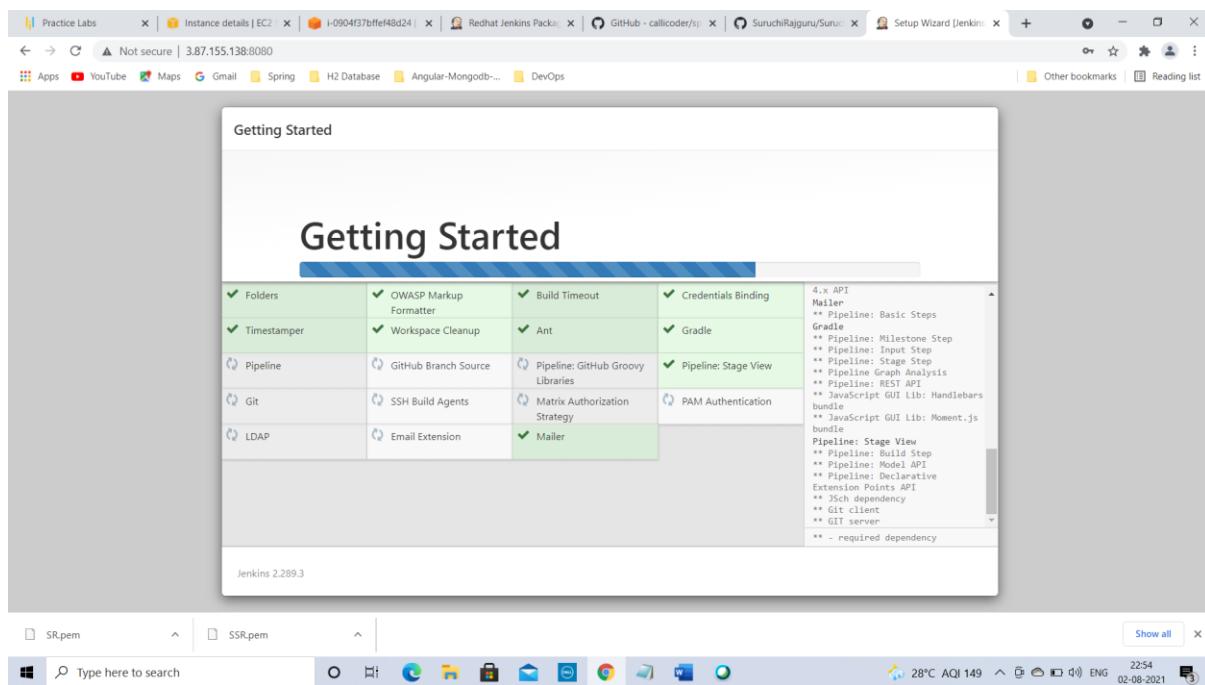
Go to command prompt & type:
cat /var/lib/Jenkins/secrets/initialAdminPassword

Initial password is stored in that location

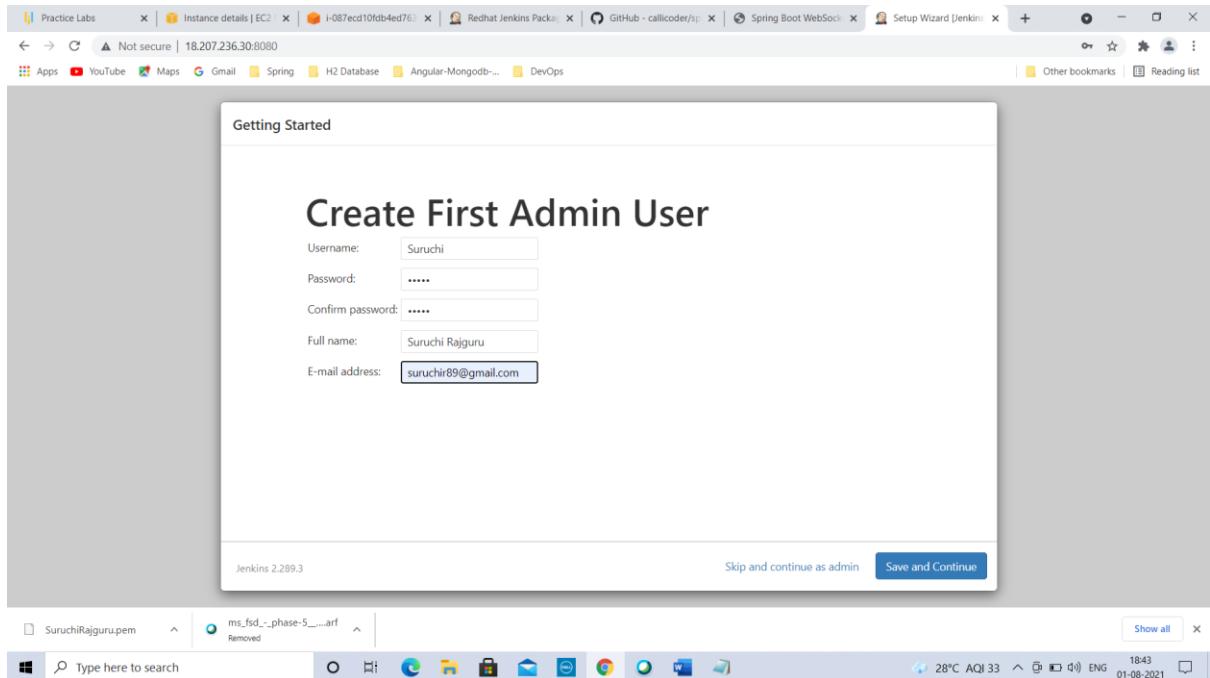
Copy the password & paste it using (shift+insert) in browser Administrator textbox & click continue.



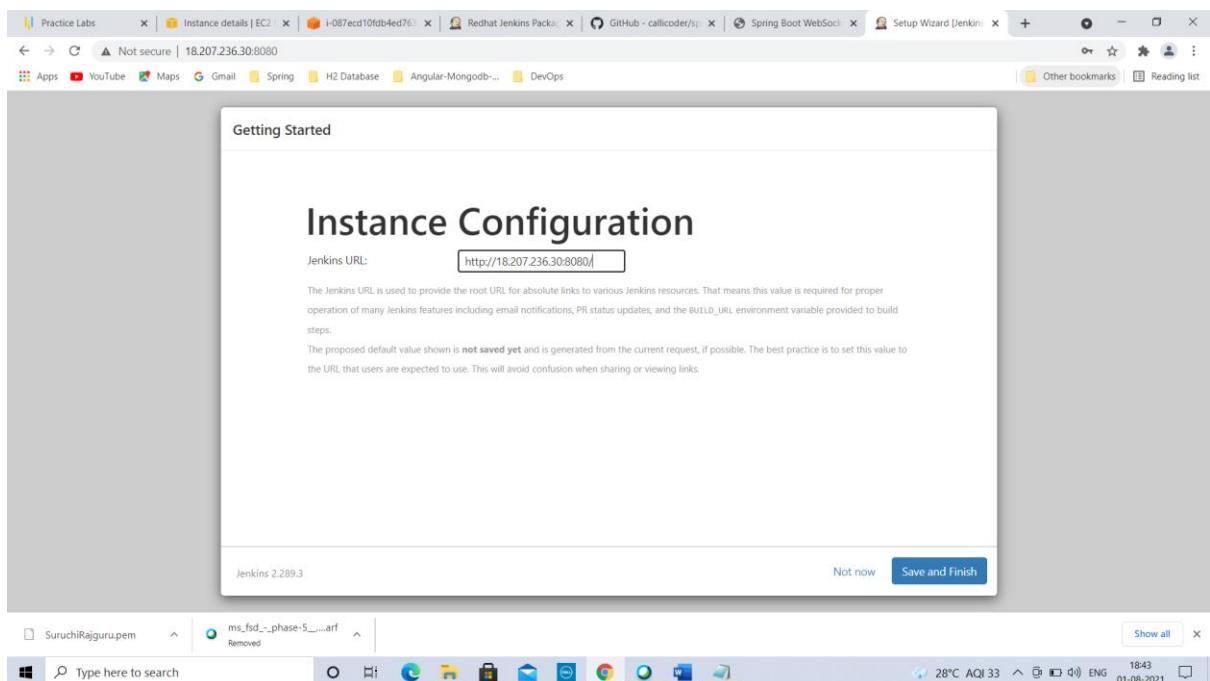
Click on Install suggested plugins → Plugins will be installed.



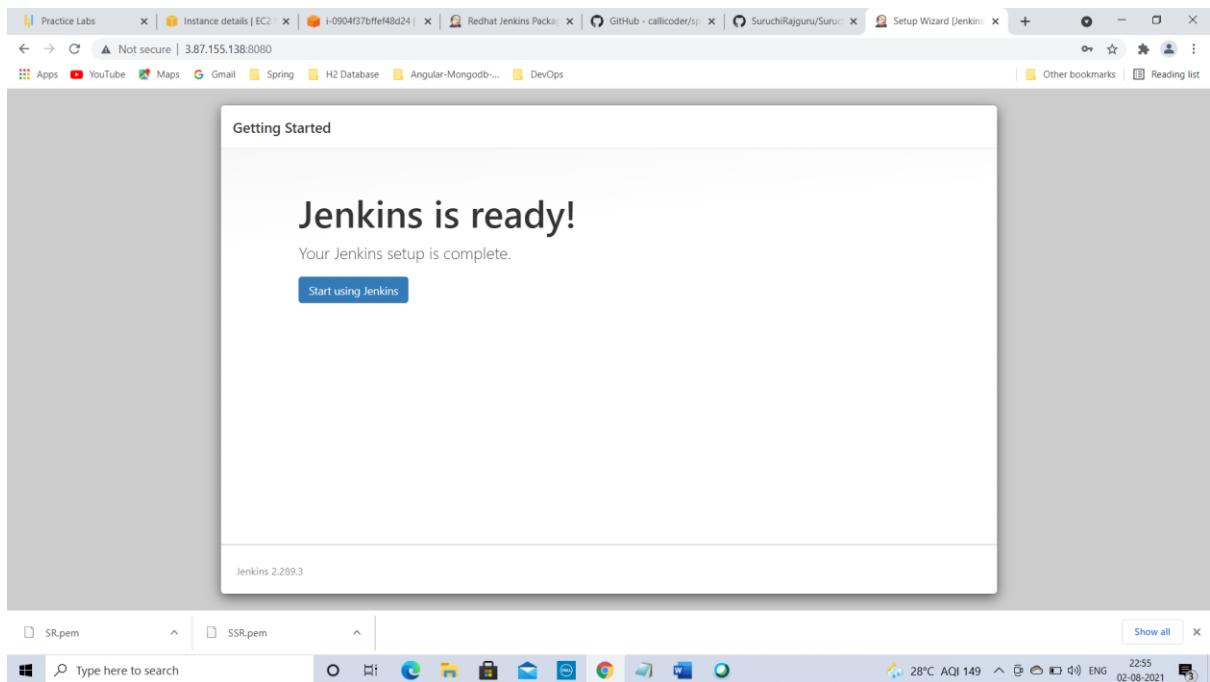
Enter details to Create First Admin User



Click on Save & Continue



Click on Save & Finish.



Click on Start using Jenkins.

You are on Jenkins home page.

Click on Manage Jenkins-> Manage plugins->

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, and Manage Jenkins. The main area has two sections: 'System Configuration' and 'Security'. Under 'System Configuration', there are links for Configure System, Global Tool Configuration, and Manage Plugins. Under 'Security', there are links for Configure Global Security, Manage Credentials, and Configure Credential Providers. A search bar at the top right says 'search'.

Go to available tab-> search for docker->

The screenshot shows the Jenkins Plugin Manager page with the 'Available' tab selected. A search bar at the top contains the text 'docker'. Below it, a table lists several Docker-related plugins: Docker, Docker Commons, Docker Pipeline, and Docker API. Each plugin entry includes its name, version, release date, and a brief description. At the bottom of the table, there are buttons for 'Install without restart' and 'Download now and install after restart'. The status bar at the bottom right shows the date as 02-08-2021 and the time as 22:56.

select Docker & install without restart →

The screenshot shows the Jenkins Update Center page with a list of installed plugins and their status. All items listed under 'Pipeline: Stage View', 'Git', 'SSH Build Agents', 'Matrix Authorization Strategy', 'PAM Authentication', 'LDAP', 'Email Extension', 'Mailer', 'Loading plugin extensions', 'Authentication Tokens API', 'Docker Commons', 'Docker API', 'Docker', and 'Loading plugin extensions' are marked as 'Success'. Below the list, there are two informational messages: 'Go back to the top page (you can start using the installed plugins right away)' and a checkbox for 'Restart Jenkins when installation is complete and no jobs are running'.

Docker is installed now.

Click on Manage Jenkins → Global Tool Configuration->

The screenshot shows the Jenkins Global Tool Configuration page. It includes sections for Maven Configuration, JDK, and Git. Under Maven Configuration, 'Default settings provider' is set to 'Use default maven settings' and 'Default global settings provider' is set to 'Use default maven global settings'. Under JDK, there is a 'JDK installations' section with an 'Add JDK' button. Under Git, there is a 'Git installations' section. At the bottom, there are 'Save' and 'Apply' buttons. The page is part of a larger browser window showing other tabs and system status at the top and bottom.

Scroll down on the same page.

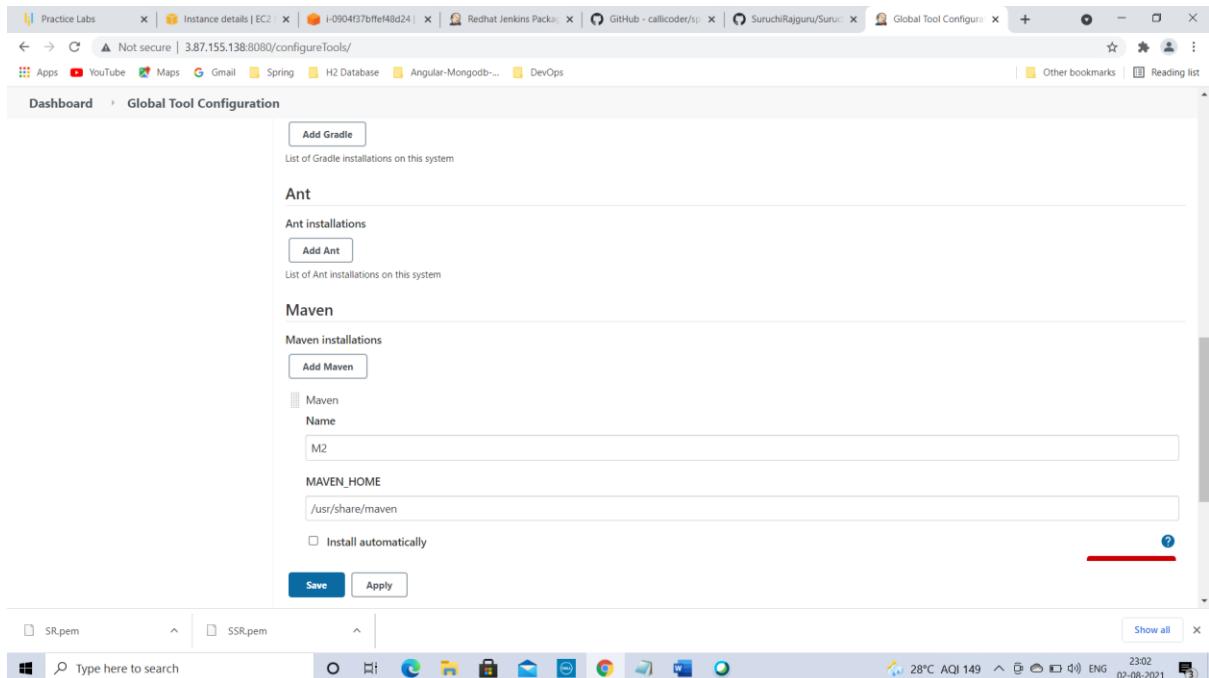
Go to Maven->
Click on Add Maven->

Enter the details as below:

Name:M2

Uncheck install automatic
Maven path-/usr/share/maven

(To get maven path...type mvn -version in command prompt.
Copy maven-home:/usr/share/maven---copy paste this from command prompt to Jenkins settings
in browser)



Click on Save →

Go to dashboard->

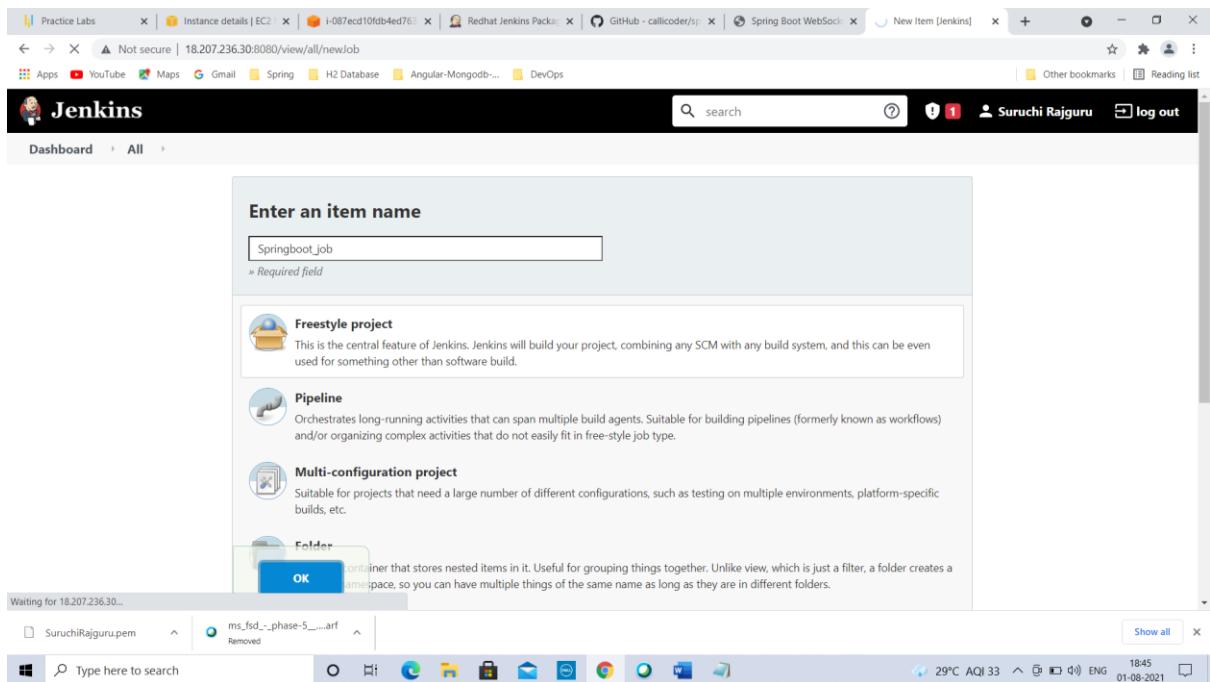
To create job first:

Click on New Item->

Enter Item Name : Springboot_Job

Select Freestyle project->

Click on OK



Go to Springboot_Job→

Click on Configure ->

Go to Source Code Management->

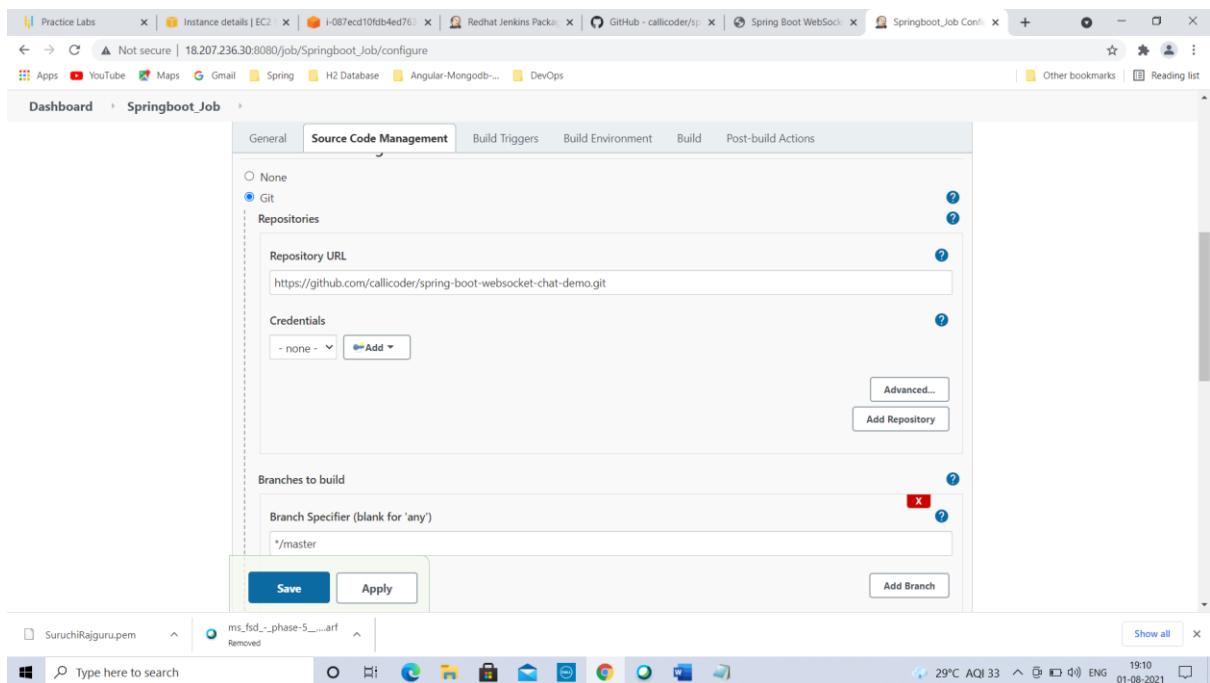
Select Git radio button→

Repositories→

Repository URL: <https://github.com/callcoder/spring-boot-websocket-chat-demo>

(Copy Github url of spring boot application)

Branch specifier: */master



Go to Build Section->

Build → Add Build Step->Invoke top level Maven targets(select value from dropdown)

Maven version: M2

Goal: package

The screenshot shows the Jenkins job configuration interface. The 'Build' tab is selected. Under the 'Invoke top-level Maven targets' section, the 'Maven Version' is set to 'M2' and the 'Goals' are set to 'package'. There are 'Add build step' and 'Advanced...' buttons. Below this is the 'Post-build Actions' section with an 'Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.

Add build step->Execute shell command->

commands: docker build -t springbootimage .

```
docker run -p 8089:8080 --name springcontainer1 springbootimage
```

Click on Save

The screenshot shows the Jenkins job configuration interface after adding the 'Execute shell' step. The 'Build' tab is selected. It contains two steps: 'Invoke top-level Maven targets' (Maven Version: M2, Goals: package) and 'Execute shell' (Command: docker build -t springbootimage .; docker run -p 8089:8080 --name springcontainer1 springbootimage). There is an 'Advanced...' button for the shell step. At the bottom are 'Save' and 'Apply' buttons.

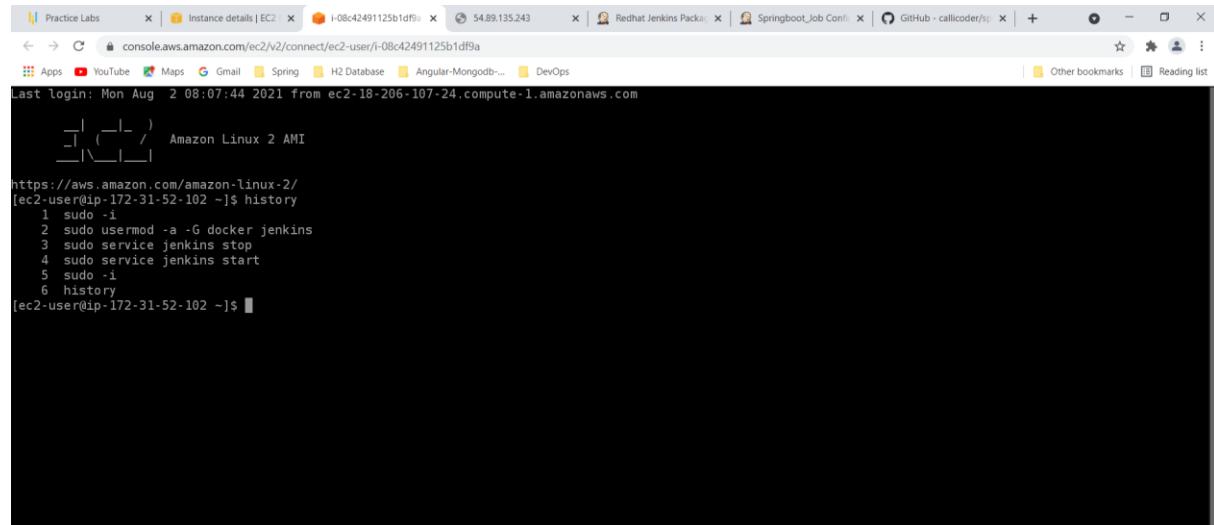
Go to command prompt & type below commands:

```
sudo -i
```

```
sudo usermod -a -G docker Jenkins
```

```
sudo service Jenkins stop
```

```
sudo service Jenkins start
```



```
Last login: Mon Aug 2 08:07:44 2021 from ec2-18-206-107-24.compute-1.amazonaws.com
[ec2-user@ip-172-31-52-102 ~]$ history
1 sudo -i
2 sudo usermod -a -G docker Jenkins
3 sudo service Jenkins stop
4 sudo service Jenkins start
5 sudo -i
6 history
[ec2-user@ip-172-31-52-102 ~]$
```

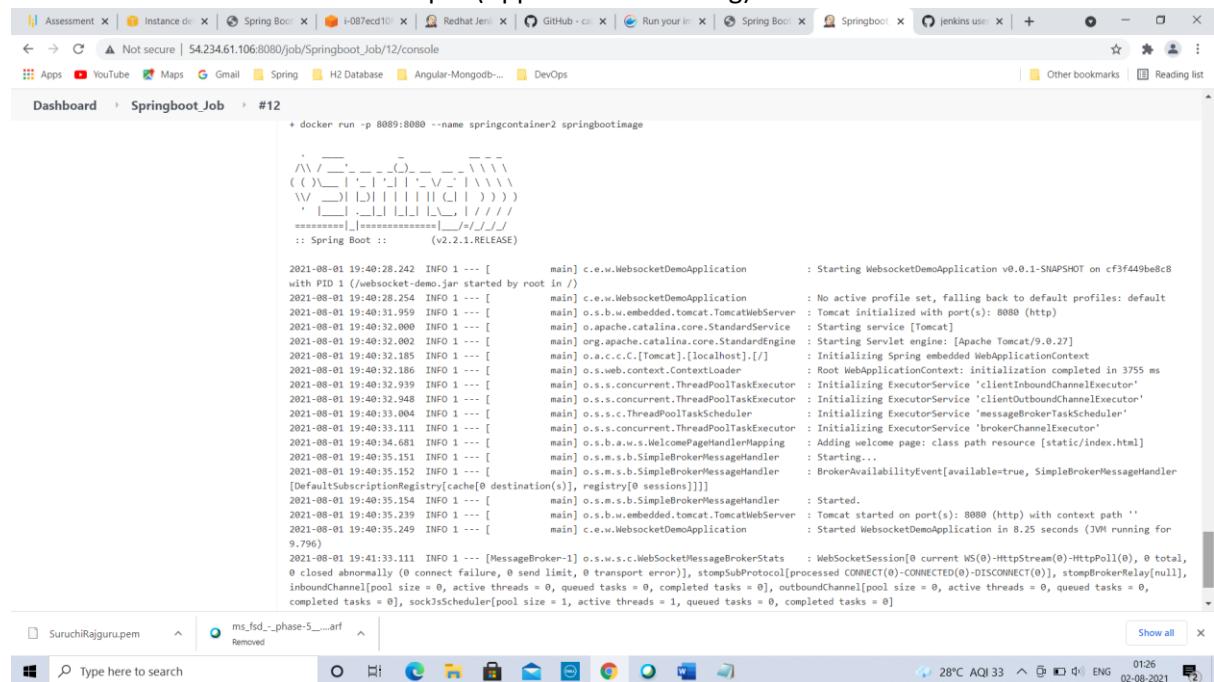
i-08c42491125b1df9a

Public IPs: 54.89.135.243 Private IPs: 172.31.52.102



Go to Jenkins tab & Login again

Go to Springboot_Job → Click on Build NowProject builds & go to build history → click on recent build → #12 → click on Console Output (Application is running)



```
+ docker run -p 8089:8080 --name springcontainer2 springbootimage
.
.
.
:: Spring Boot ::      (v2.2.1.RELEASE)

2021-08-01 19:40:28.242  INFO 1 --- [main] c.e.websocketDemoApplication        : Starting WebsocketDemoApplication v0.0.1-SNAPSHOT on cf3f449be8c8
with PID 1 (/websocket-demo.jar started by root in /)
2021-08-01 19:40:28.254  INFO 1 --- [main] c.e.websocketDemoApplication        : No active profile set, falling back to default profiles: default
2021-08-01 19:40:31.959  INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-08-01 19:40:32.000  INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-08-01 19:40:32.002  INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2021-08-01 19:40:32.185  INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]    : Initializing Spring embedded WebApplicationContext
2021-08-01 19:40:32.186  INFO 1 --- [main] o.s.web.context.ContextLoader       : Root WebApplicationContext: initialization completed in 3755 ms
2021-08-01 19:40:32.939  INFO 1 --- [main] o.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'clientInboundChannelExecutor'
2021-08-01 19:40:32.948  INFO 1 --- [main] o.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'clientOutboundChannelExecutor'
2021-08-01 19:40:33.004  INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskScheduler : Initializing ExecutorService 'messageBrokerTaskScheduler'
2021-08-01 19:40:33.111  INFO 1 --- [main] o.s.b.a.w.WelcomePageHandlerMapping  : Adding welcome page: class path resource [static/index.html]
2021-08-01 19:40:34.681  INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler  : Starting...
2021-08-01 19:40:35.151  INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler  : BrokerAvailabilityEvent{available=true, SimpleBrokerMessageHandler}
2021-08-01 19:40:35.152  INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler  : [DefaultSubscriptionRegistry]cache[0 destination(s)], registry[0 sessions]]
2021-08-01 19:40:35.154  INFO 1 --- [main] o.s.m.s.b.SimpleBrokerMessageHandler  : Started.
2021-08-01 19:40:35.239  INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-08-01 19:40:35.249  INFO 1 --- [main] c.e.websocketDemoApplication        : Started WebsocketDemoApplication in 8.25 seconds (JVM running for 9.796)
2021-08-01 19:41:33.111  INFO 1 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats  : WebSocketSession[0 current WS(0)-HttpStream(0), 0 total, 0 closed abnormally, 0 connect failure, 0 send limit, 0 transport error], stompSubProtocol[processed CONNECT(0)-CONNECTED(0)-DISCONNECT(0)], stompBrokerRelay(null), inboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], outboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], sockJsScheduler[pool size = 1, active threads = 1, queued tasks = 0, completed tasks = 0]
2021-08-01 19:41:33.111  INFO 1 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats  : WebSocketSession[0 current WS(0)-HttpStream(0), 0 total, 0 closed abnormally, 0 connect failure, 0 send limit, 0 transport error], stompSubProtocol[processed CONNECT(0)-CONNECTED(0)-DISCONNECT(0)], stompBrokerRelay(null), inboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], outboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], sockJsScheduler[pool size = 1, active threads = 1, queued tasks = 0, completed tasks = 0]
```



Conclusion: As a Full Stack Developer, successfully built a CI/CD pipeline to demonstrate continuous deployment and host the application on AWS EC2 instance.

Successfully automated the integration and deployment of the web application.

As the source code is supposed to be fetched from a GitHub repository, Github link of the project is :
<https://github.com/callcoder/spring-boot-websocket-chat-demo.git>

Technologies used:

- 1) GitHub 2) Jenkins 3) AWS EC2/ Virtual machine