TECHNOLOGY

Computing

Caltech | Center for Technology & Management Education

# Configure AWS for Back End

Powered by simplilearn

Caltech | Center for Technology & Management Education

# Set Up AWS for Java Back End

Powered by simplilearn

# You Already Know

Before we begin, let's recall what we have covered till now:


Agile

Git


SQL

Angular


HTML


CSS


JavaScript


CoreJava

# You Already Know

Before we begin, let's recall what we have covered till now:


JDBC


JSP


Servlets


MongoDB


Maven

# You Already Know

Before we begin, let's recall what we have covered till now:

JUnit

Spring

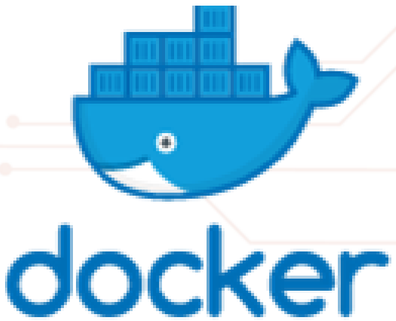
Spring Boot


Webservices


Microservices

# You Already Know

Before we begin, let's recall what we have covered till now:




Docker


Jenkins


AWS

Recap

## Front-End Backend Communication

- Used HTTP Client in Angular to communicate with Java back end
- Implemented communication for both admin and end-user projects

## Jenkins Pipeline

- Used Jenkins build automation server to build angular apps and Java back end
- Created Pipeline with multiple stages

Recap

## Docker

- Used Docker to build images for Angular apps and Java back end
- Used Jenkins to dockerize the Angular apps and Java back end with Jenkins

## Configure Angular Apps on EC2

- Created and configured EC2 instance for Angular apps
- Ran Jenkins pipeline to dockerize the Angular apps and deploy them

# A Day in the Life of a Full Stack Developer

As a full stack web developer, our key role is to develop both client and server software.

- Angular and Node can be used to build front end of the web page.

- Spring Boot, Java, and MySQL or MongoDB can be used to build at the back end.

# A Day in the Life of a Full Stack Developer

Bob needs to configure AWS for Front end. He brainstorms a bit and finds a solution.

Let me use Jenkins, Docker, and AWS to build CI/CD Pipelines, containerize the apps, and finally host them to AWS EC2 instance.


In this lesson, we will host the Java Backend app on EC2 instance, use Jenkins and Docker as DevOps tools to make it live and help Bob to complete his task effectively and quickly.

# Learning Objectives

By the end of this lesson, you will be able to:

- Explain the steps to create and configure EC2 instance on AWS
- Discuss the steps to create MySQL RDS database on AWS
- Understand the use Jenkins on EC2 for CI/CD
- Discuss how to integrate Docker in Jenkins to build and release the images as containers on EC2

# Create and Configure EC2 in AWS

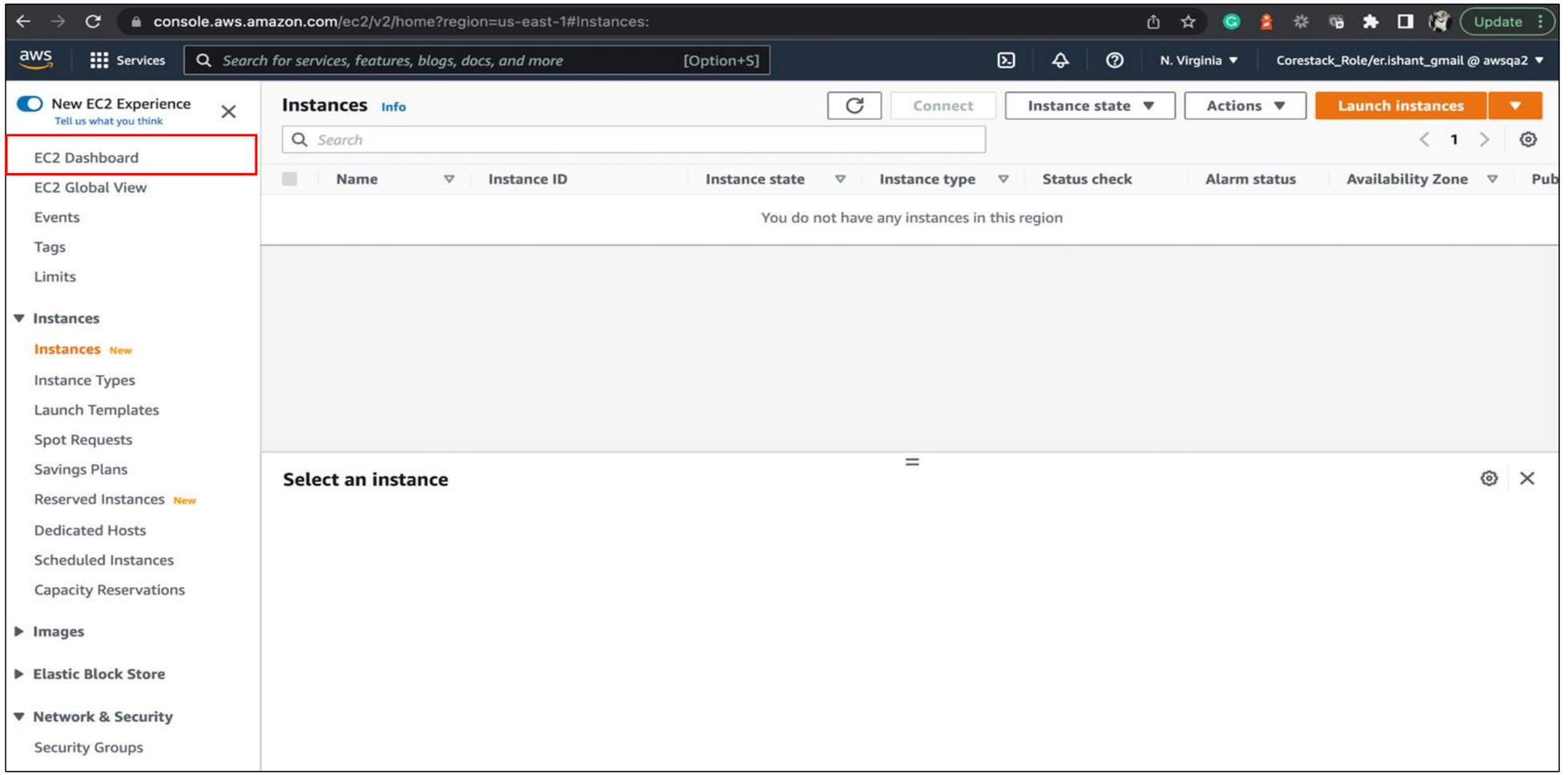
# Amazon Web Services

Amazon Web Services (AWS) is a cloud platform, which offers more than 200 fully featured services.


Angular web apps on EC2 instance is deployed using Jenkins and Docker.

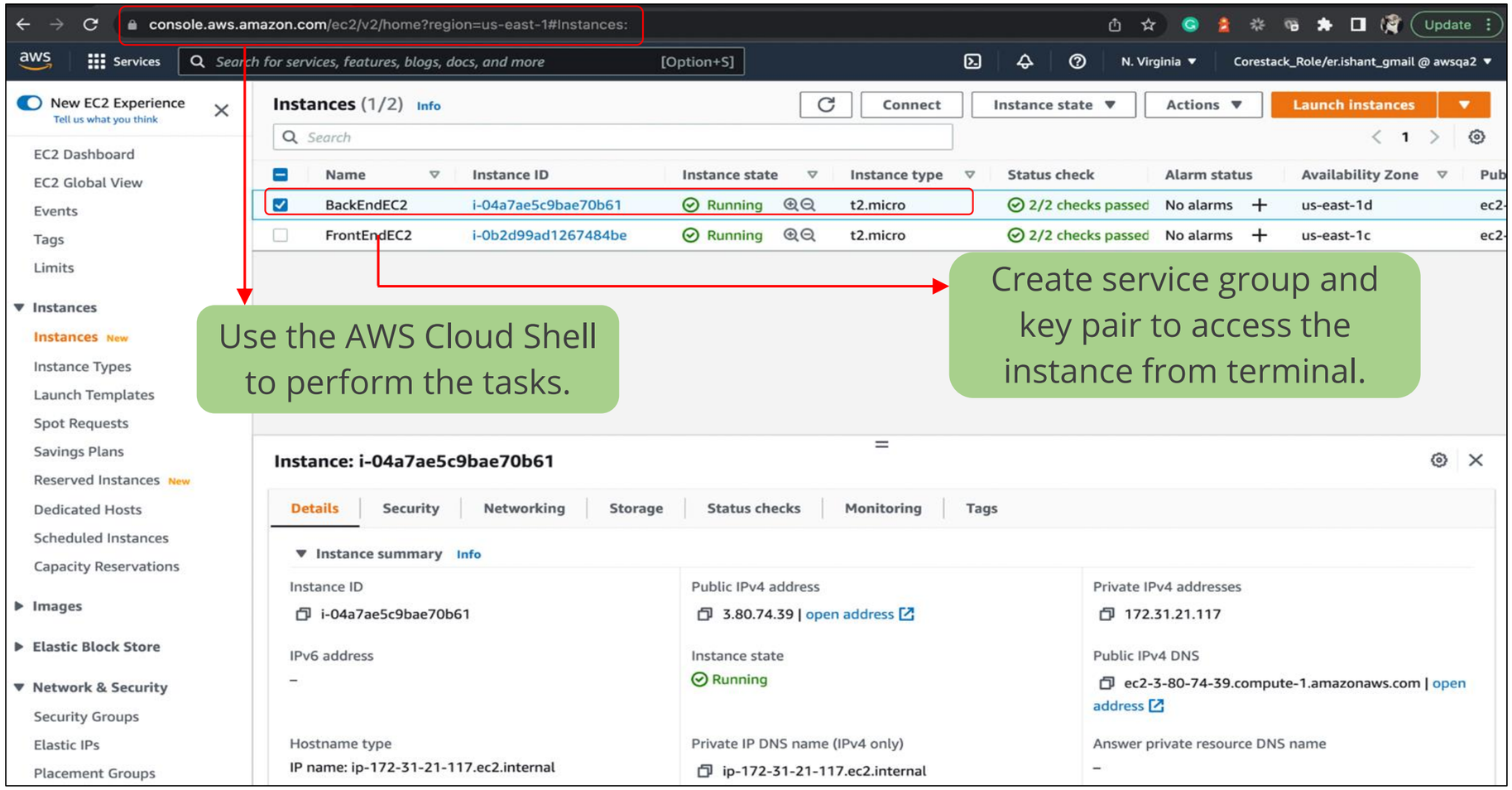
# Create EC2 Instance

Open the EC2 dashboard.



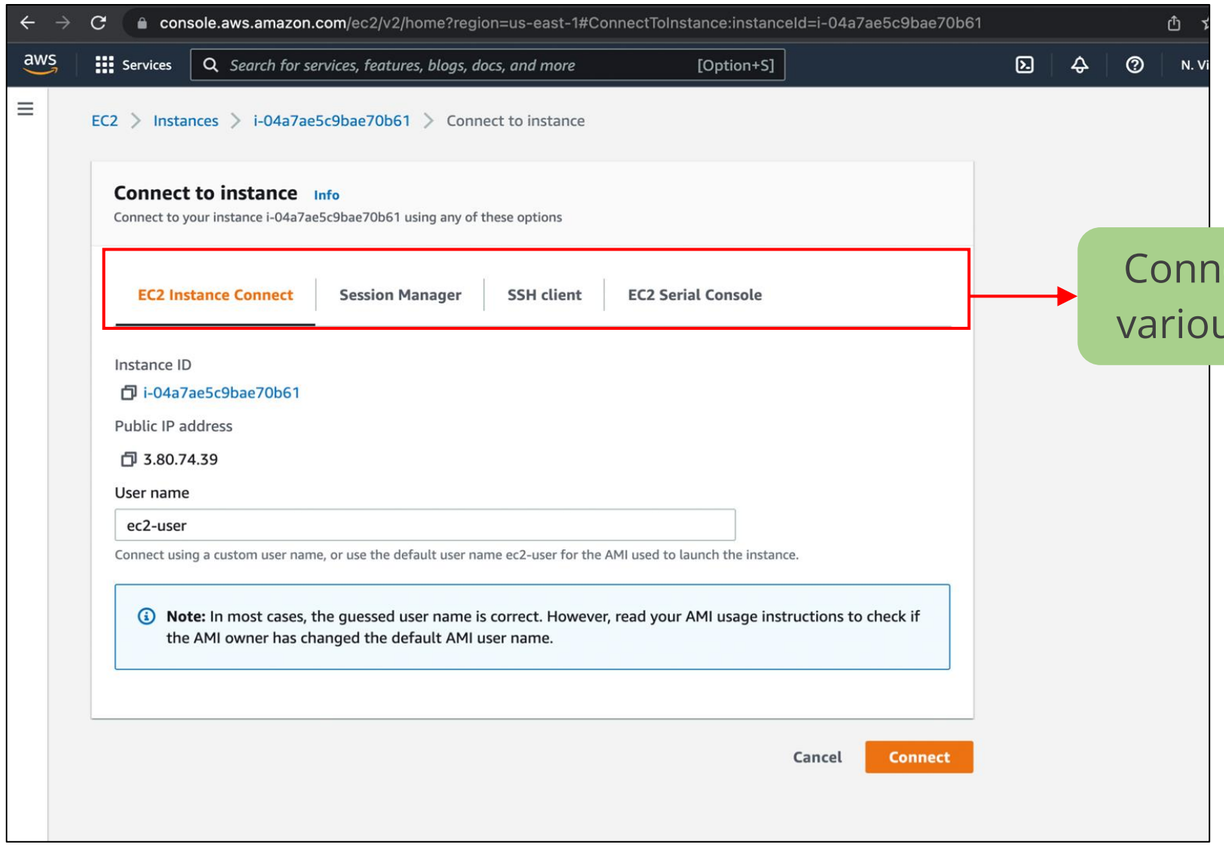
# Launch Instance

Launch a new instance.



# Connect to EC2 Instance

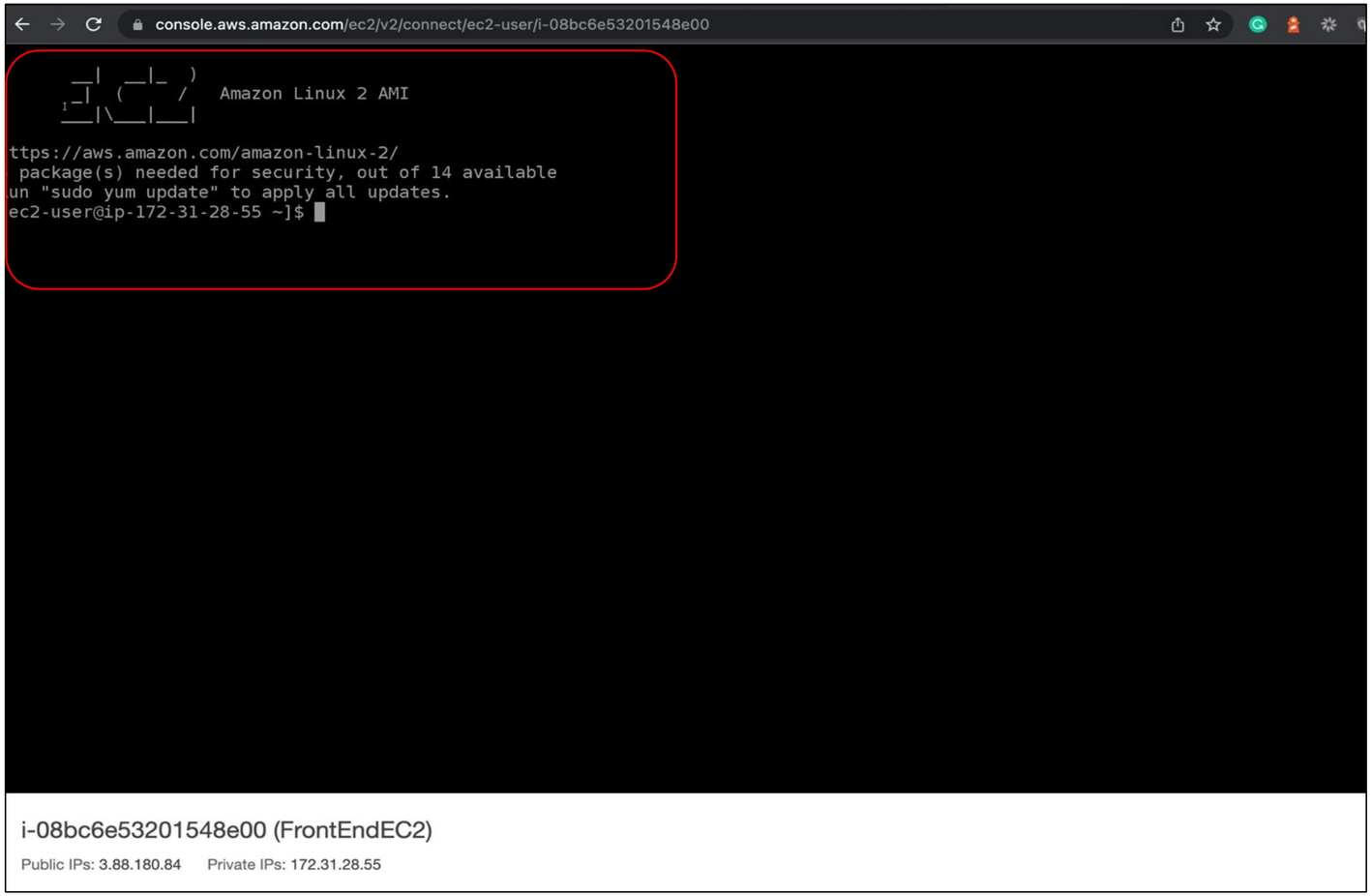
Connect through AWS Cloud Shell, which is the first option on clicking connect, that is EC2 Instance Connect.




Connect using one of the various methods, like ssh.

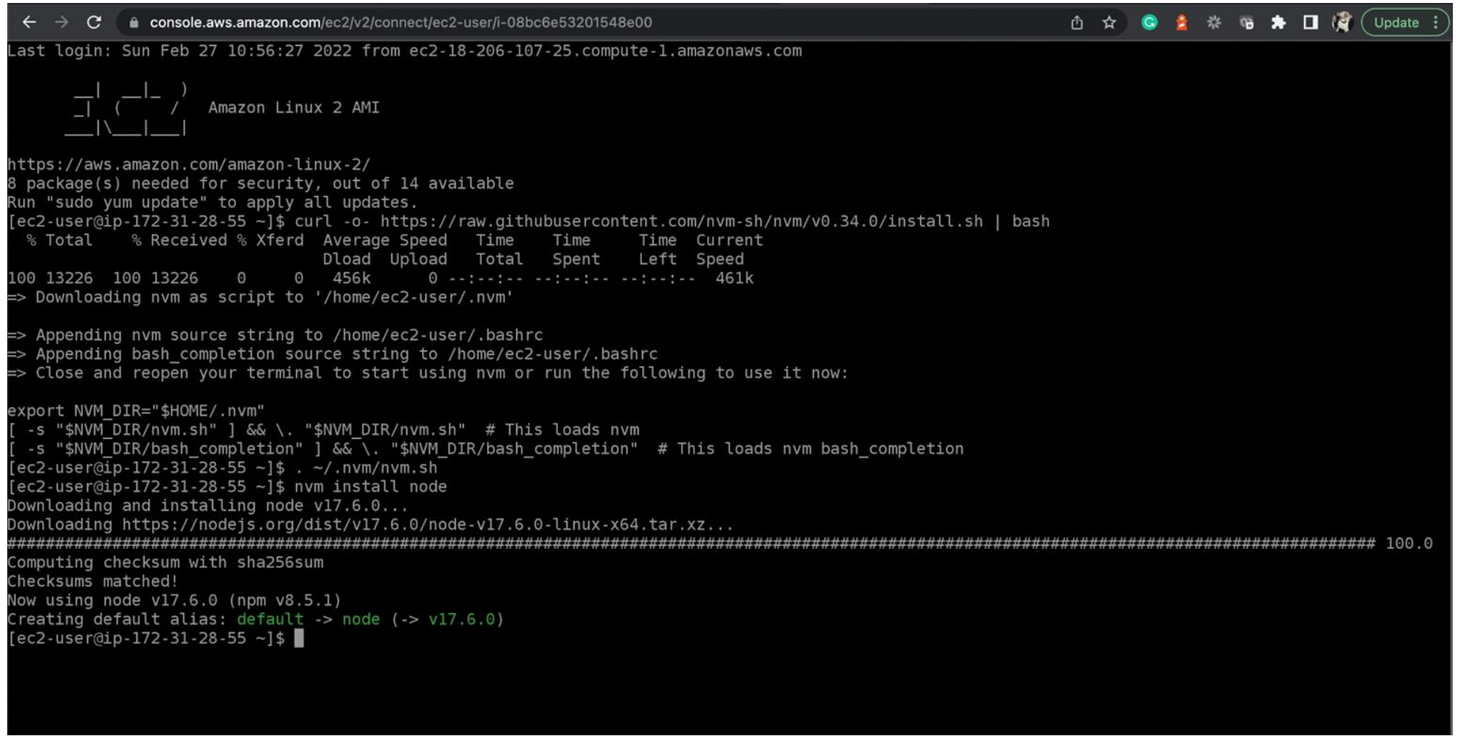
# Launch the EC2 Shell and Configure the Tools

Install the tools and software required to deploy Angular apps and configure Node, Git, Docker, and Jenkins.



# Configure Node on EC2

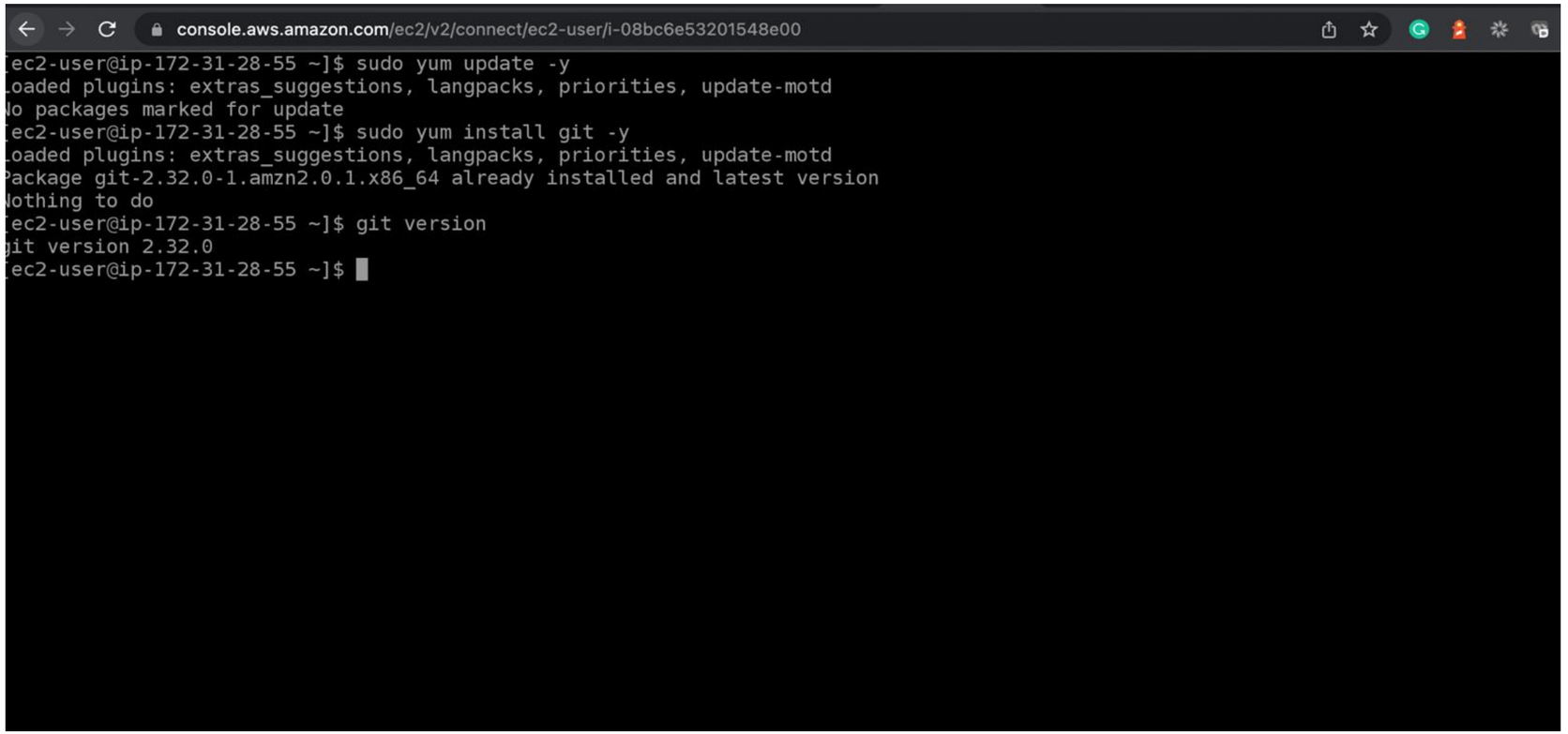
Configure the Jenkins Pipeline Project by passing GitHub repository URL.




Reference link on AWS documentation: https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html

# Configure Git on EC2 Instance

Configure Git on EC2 Instance.



# Configure Java on EC2 Instance

Use the following commands to install Java:

## Install JDK 8

- sudo yum update -y
- sudo yum install java-1.8.0-openjdk

## Install JDK 11 - Java

- sudo amazon-linux-extras install java-openjdk11

# Configure Java on EC2 Instance

Use the following commands to install Java:

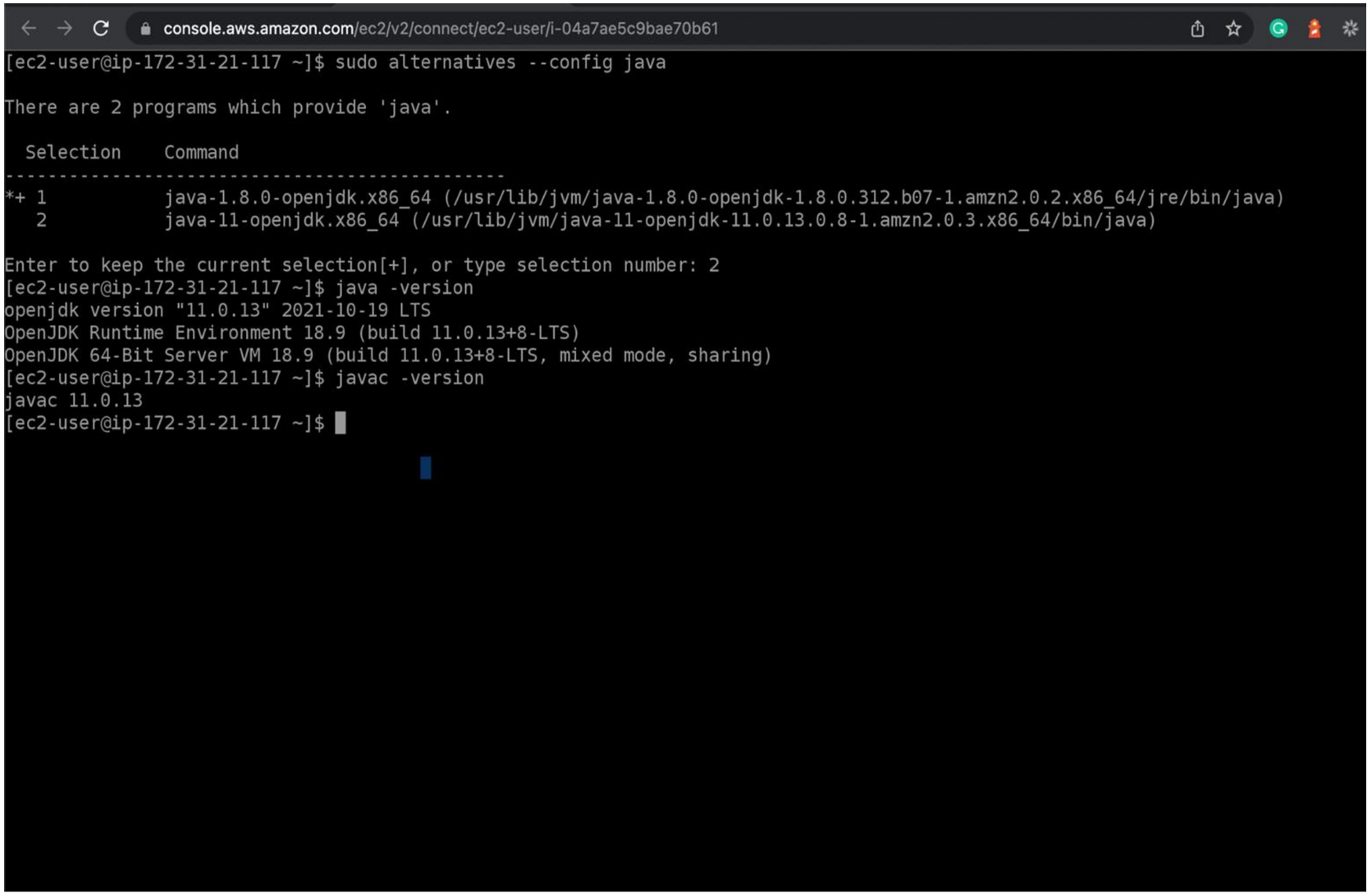
## Install JDK 11 Development - javac

- sudo yum install java-11-openjdk-devel

## Configure Java Version 8 or 11

- sudo alternatives --config java

# Configure Java on EC2 Instance



# Configure Docker on EC2 Instance

Now, install Docker for containerizing the Angular apps. Here are some useful commands:

1

Install Docker:
**sudo yum install docker**

2

Add the ec2-user to the docker group to execute Docker commands without using sudo, exit the terminal, and re-login to make the change effective.

**sudo usermod -a -G docker ec2-user**

**exit**

# Configure Docker on EC2 Instance

3

Enable docker service:
**sudo systemctl enable docker**

4

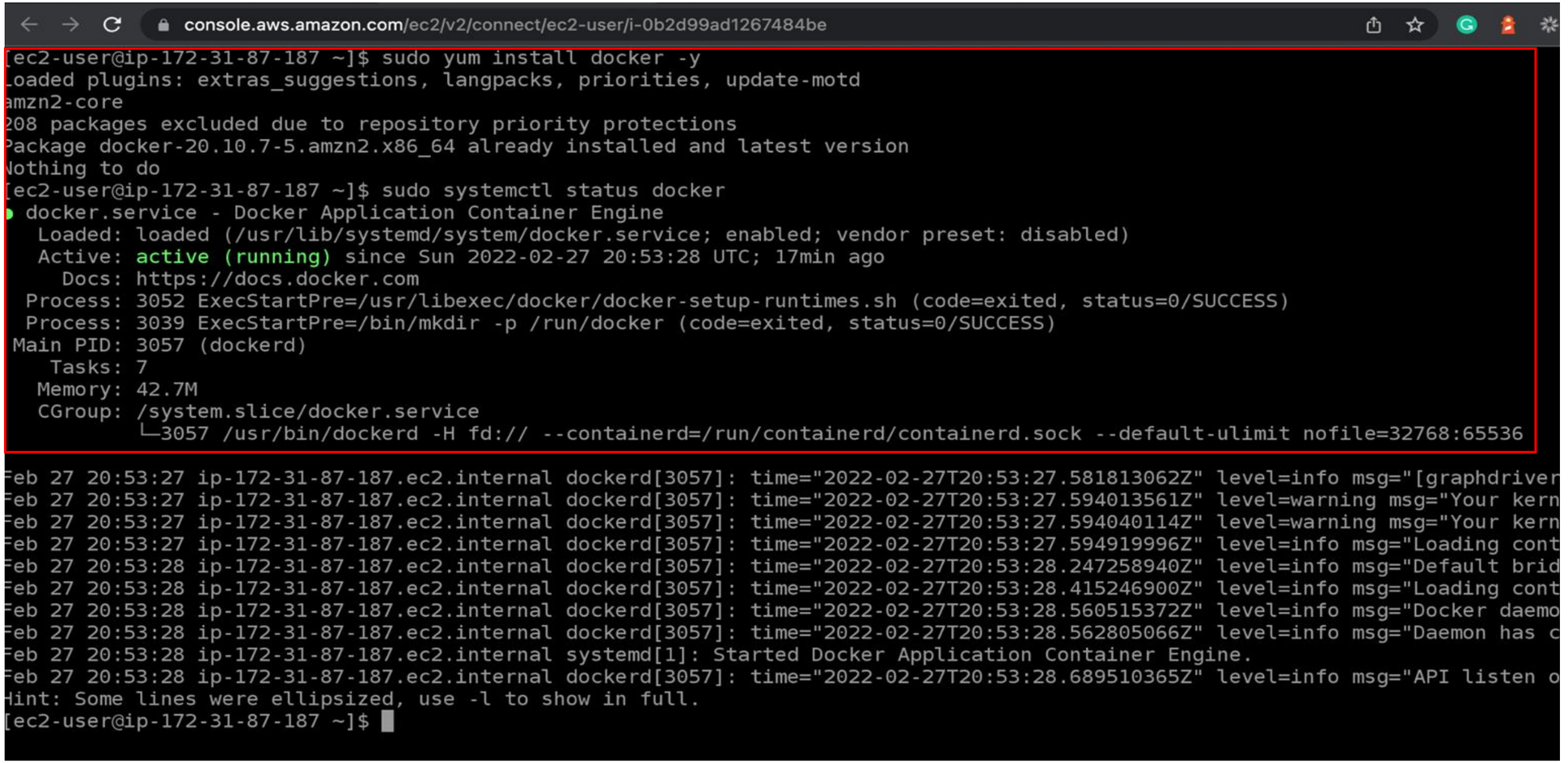
Start docker service:
**sudo systemctl start docker**

5

Check the docker service:
**sudo systemctl status docker**

# Configure Docker on EC2 Instance

The image below shows the visual of the commands:



# Configure Jenkins on EC2 Instance

To configure Jenkins on EC2 instance, install Java first. Use the following commands:

1

Install JDK 8:
**sudo yum update -y**
**sudo yum install java-1.8.0-openjdk**

2

# Install JDK 11 – java:
**sudo amazon-linux-extras install java-openjdk11**

3

# Install JDK 11 Development – javac:
**sudo yum install java-11-openjdk-devel**

# Configure Jenkins on EC2 Instance

4

Configure Java Version 8 or 11:
**sudo alternatives --config java**

5

Read OS name:
**cat /etc/os-release**

# Configure Jenkins on EC2 Instance

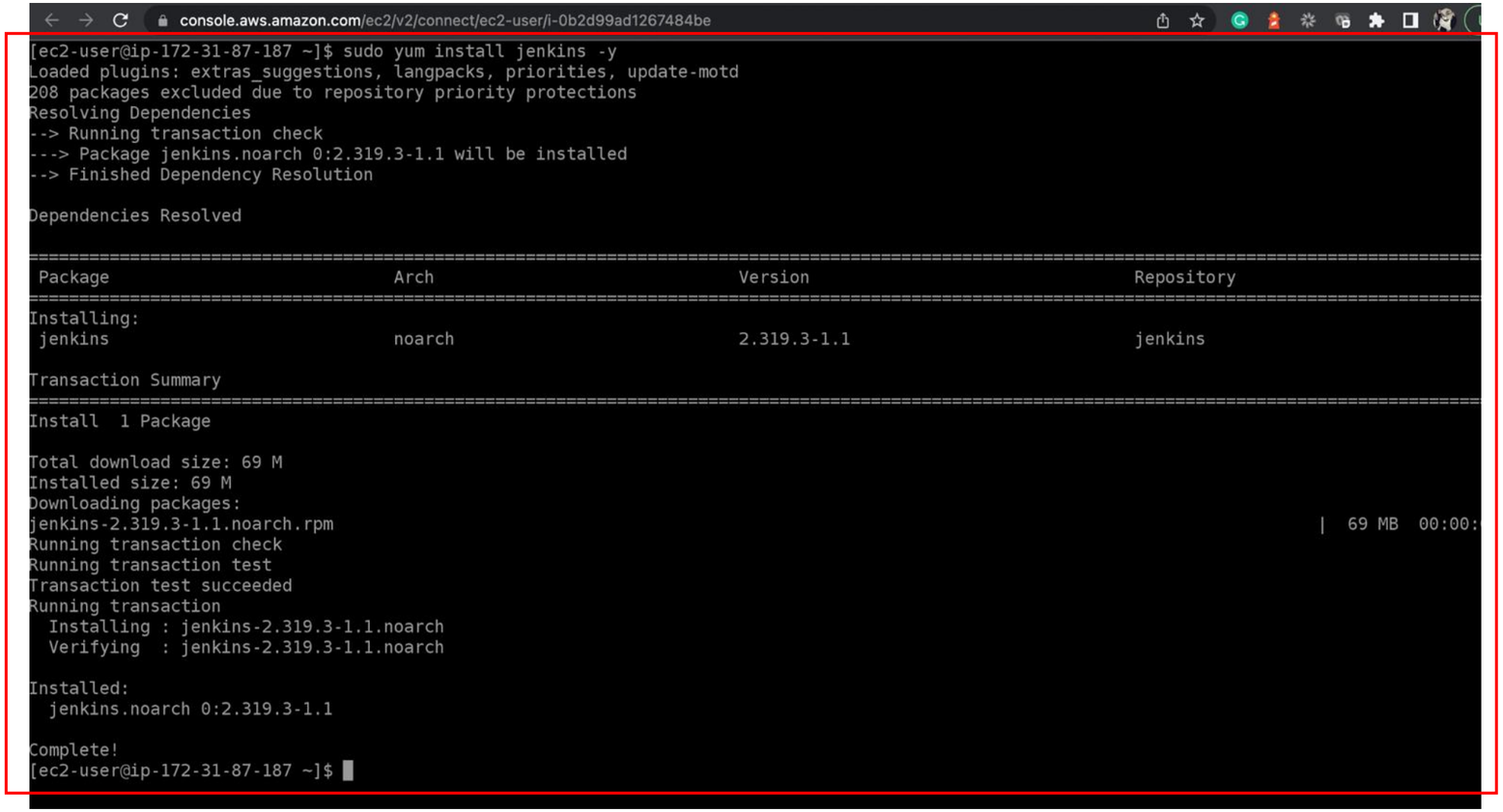
6

From Jenkins Installation document follow instructions:
[https://www.jenkins.io/doc/book/installing/linux/]

**sudo amazon-linux-extras install epel**
**sudo yum-config-manager --enable epel**
**sudo yum install daemonize -y**
**sudo yum install jenkins -y**
**sudo systemctl daemon-reload**

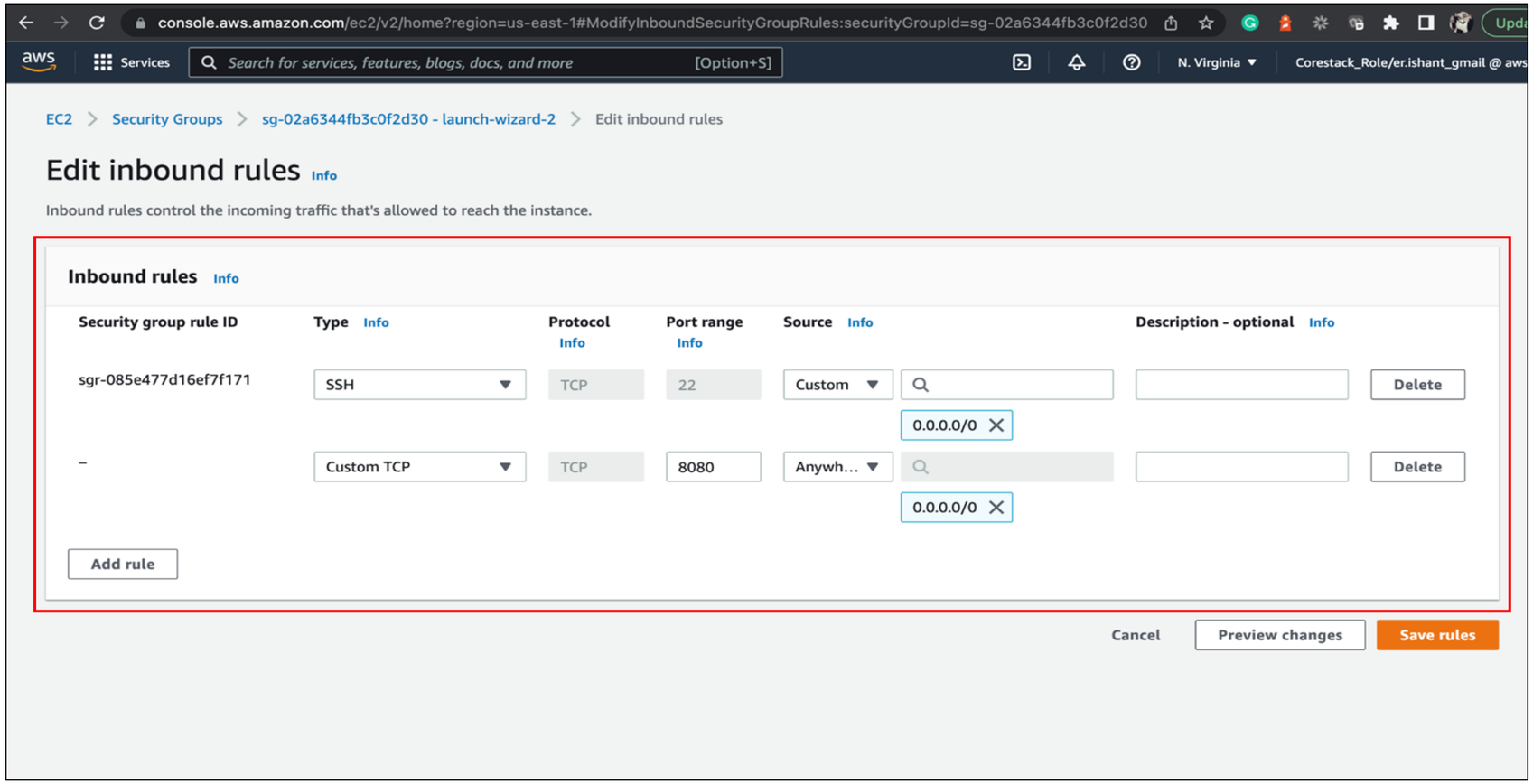
**sudo systemctl enable jenkins**
**sudo systemctl start jenkins**
**sudo systemctl status jenkins**

# Configure Jenkins on EC2 Instance



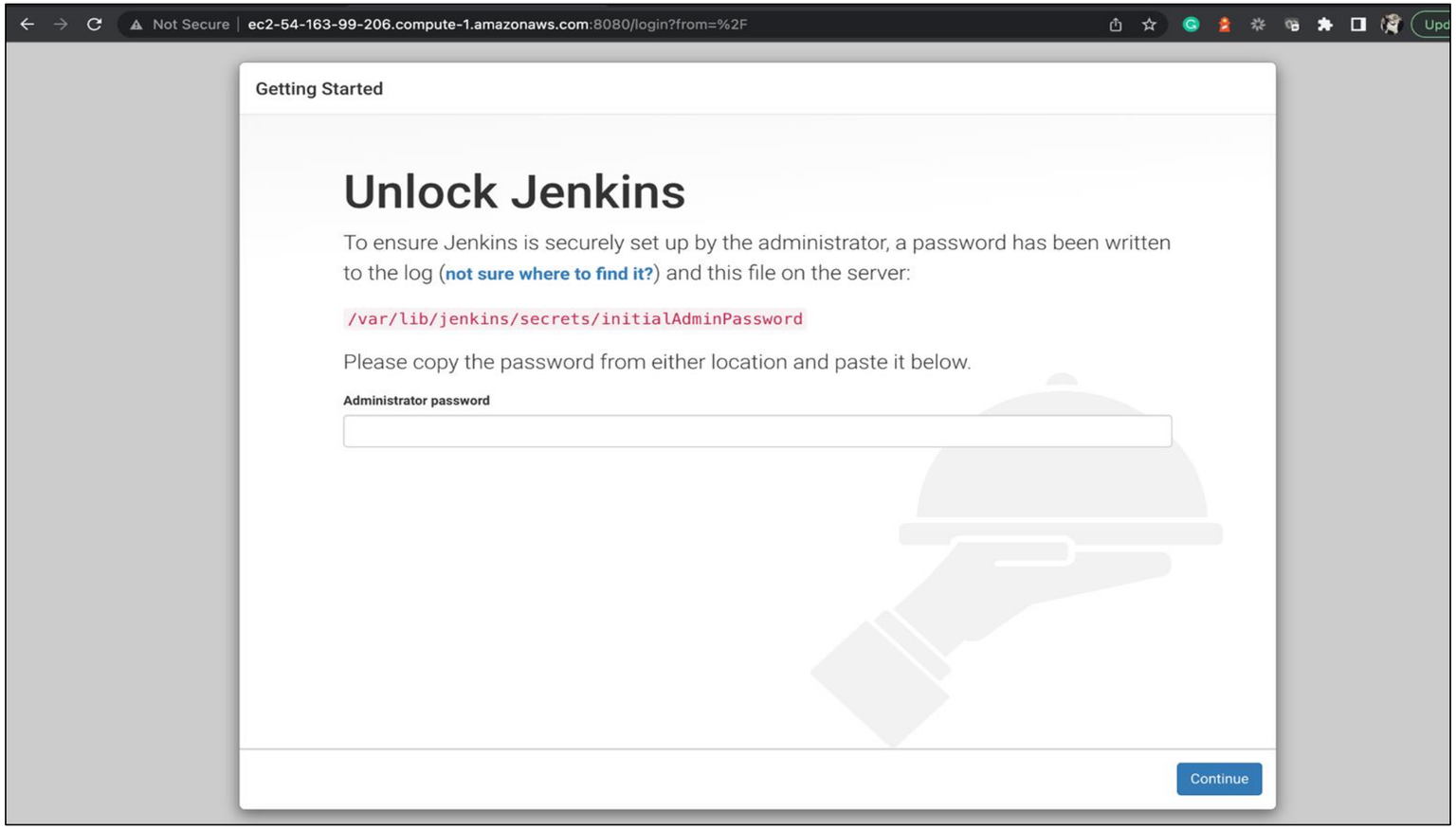
# Configure Jenkins on EC2

**Edit** the Inbound rules of security group associated to EC2 instance by opening port 8080 so that Jenkins can run.



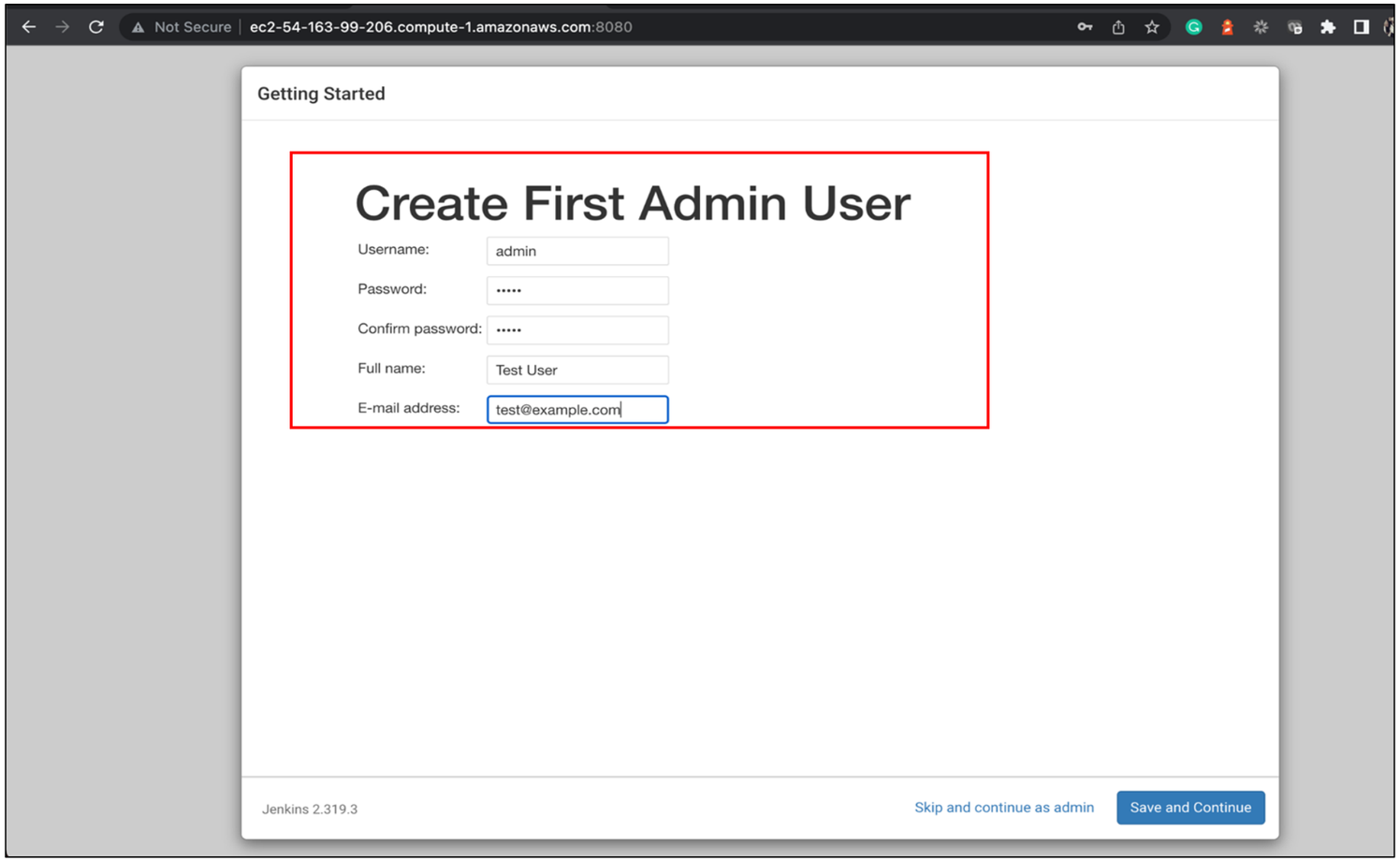
# Access Jenkins on Port 8080 of EC2

From here, proceed by logging into the Jenkins with the **initial password.**



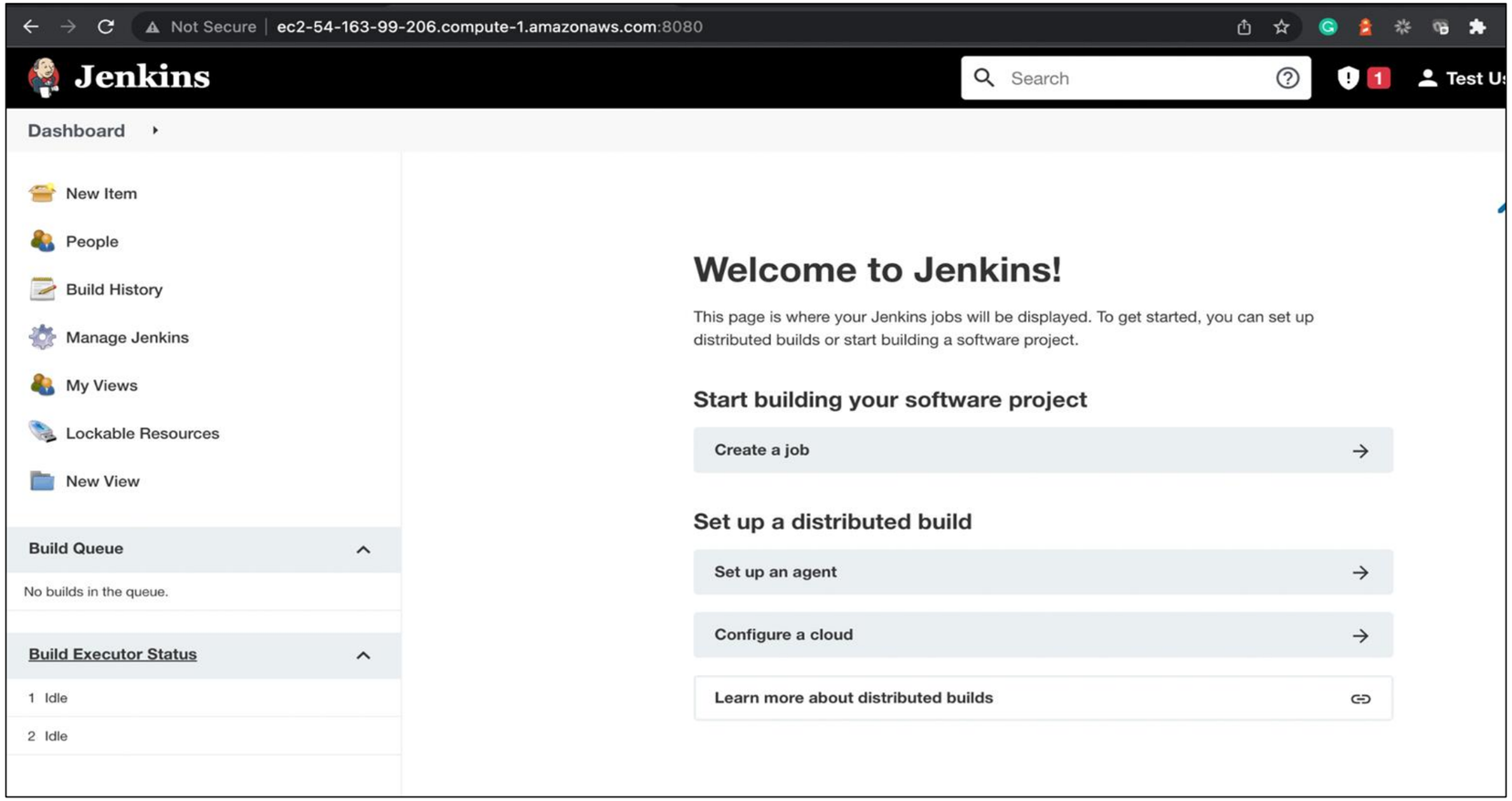
# Access Jenkins on Port 8080 of EC2

Create a **new user.**



# Configure Jenkins on EC2

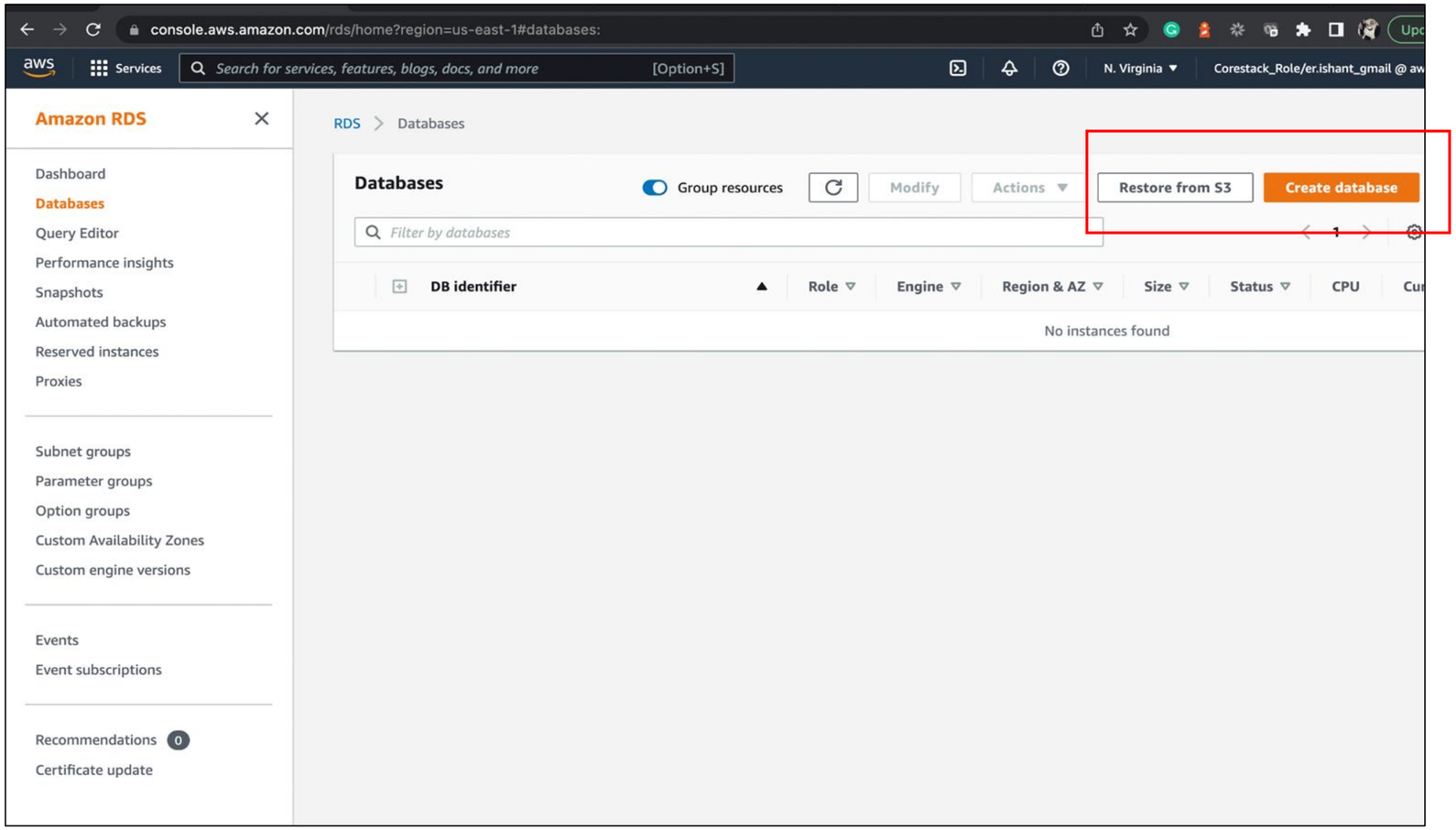
Jenkins is ready to be used on EC2 now.



# Create AWS MySQL RDS Instance

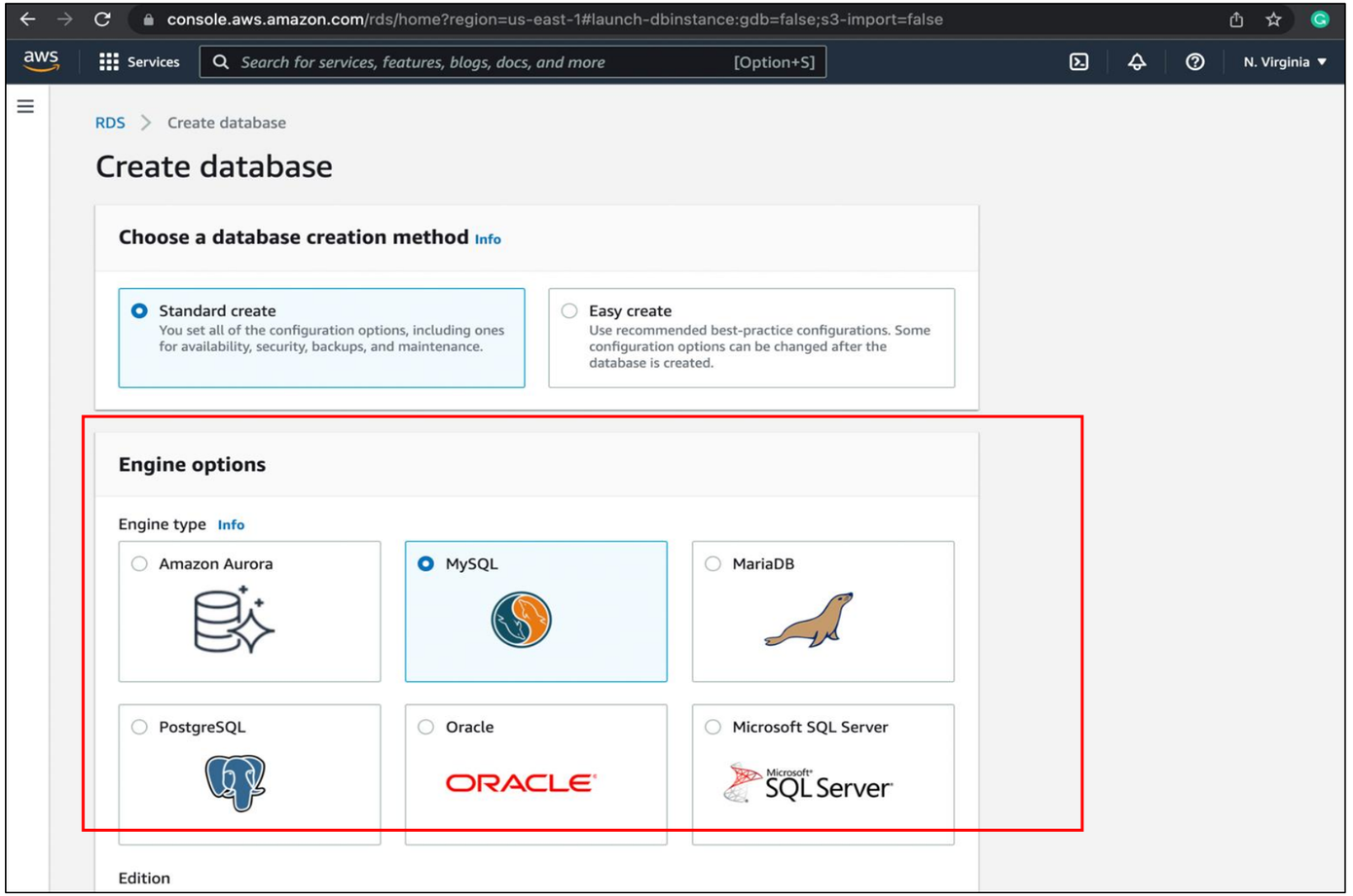
# Create MySQL Database

From the Amazon RDS, click on the **Create database** button.



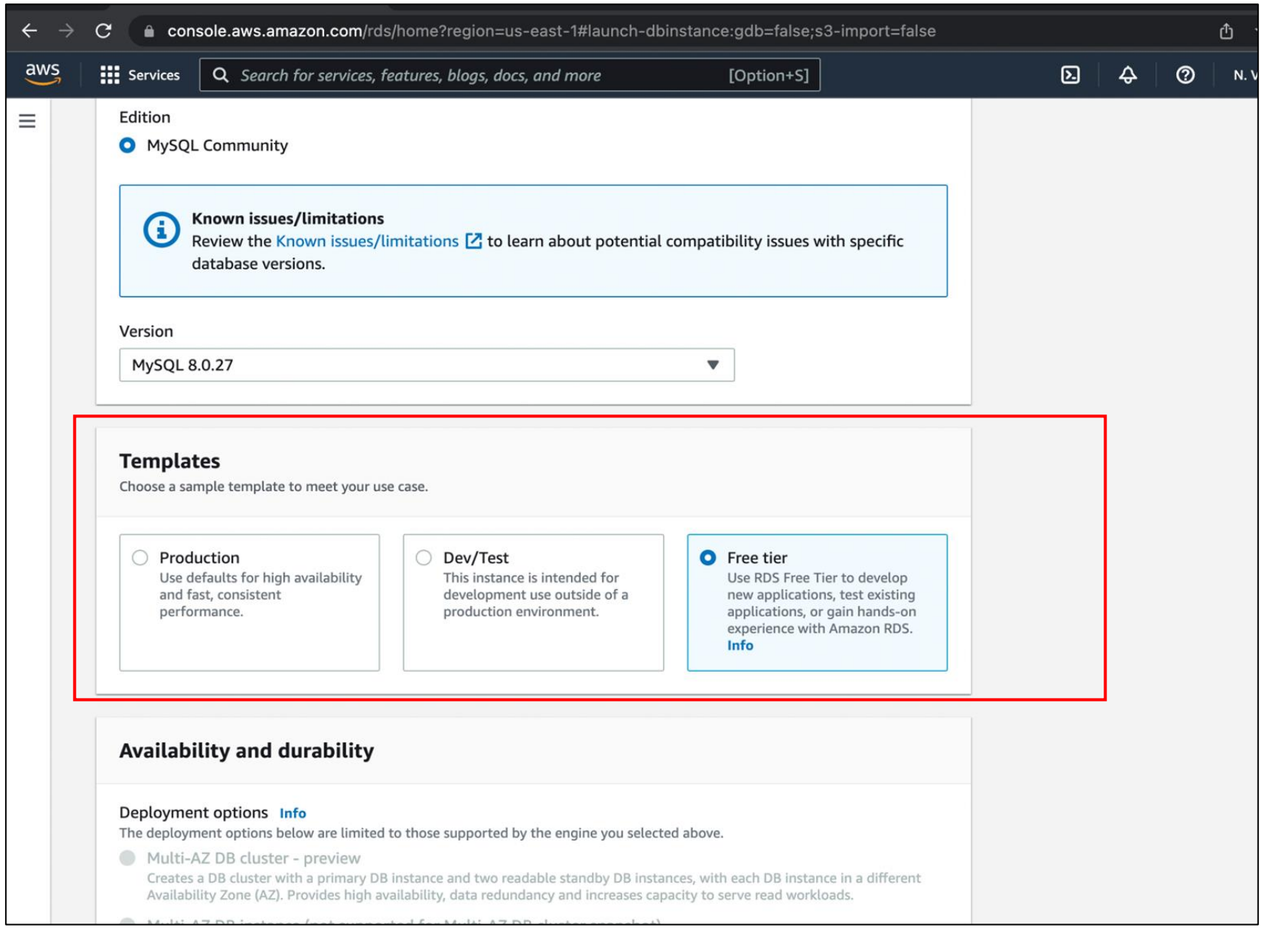
# Create MySQL Database

**Choose** MySQL from the database selection.



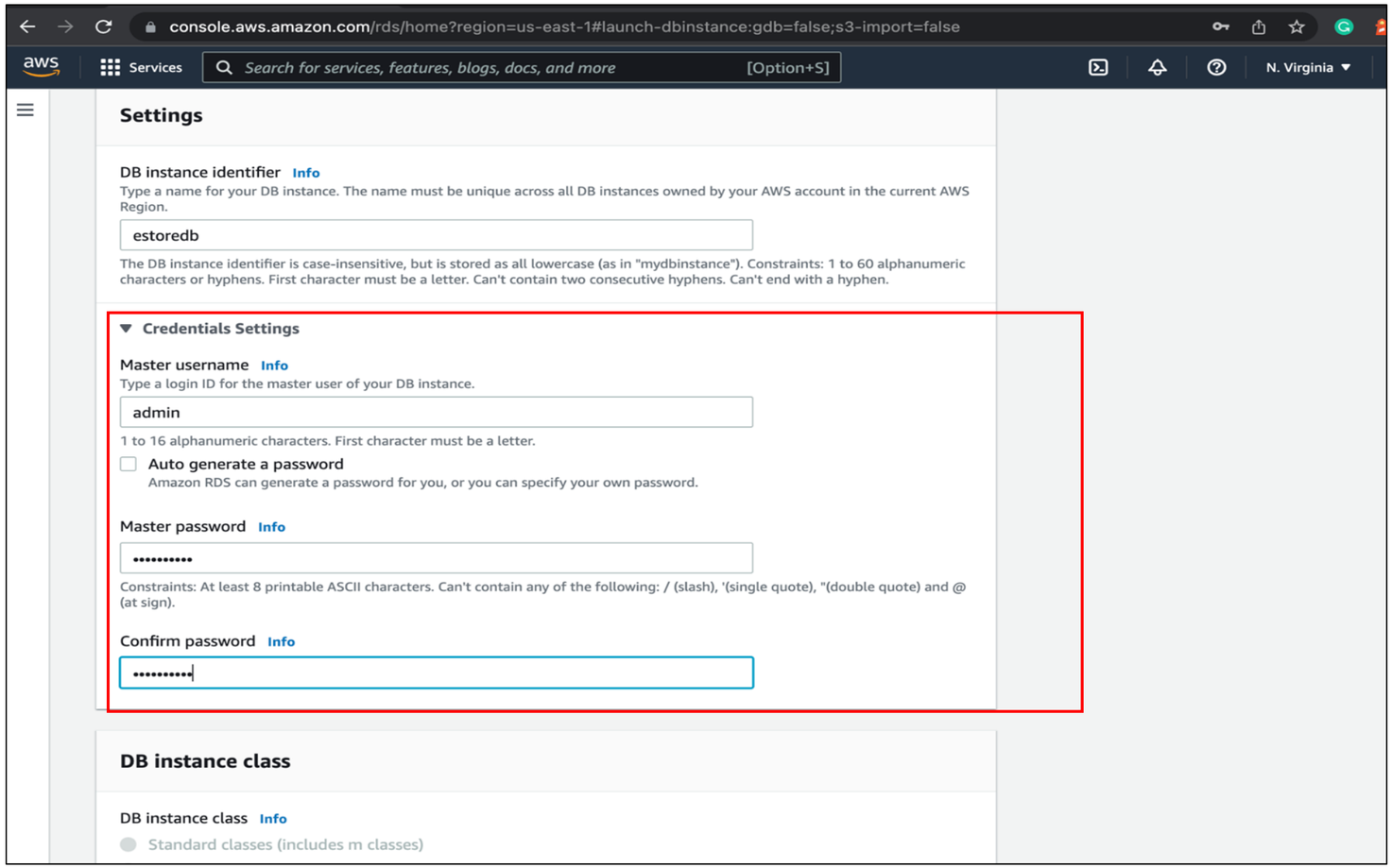
# Create MySQL Database

From Templates, select **the Free tier** option.



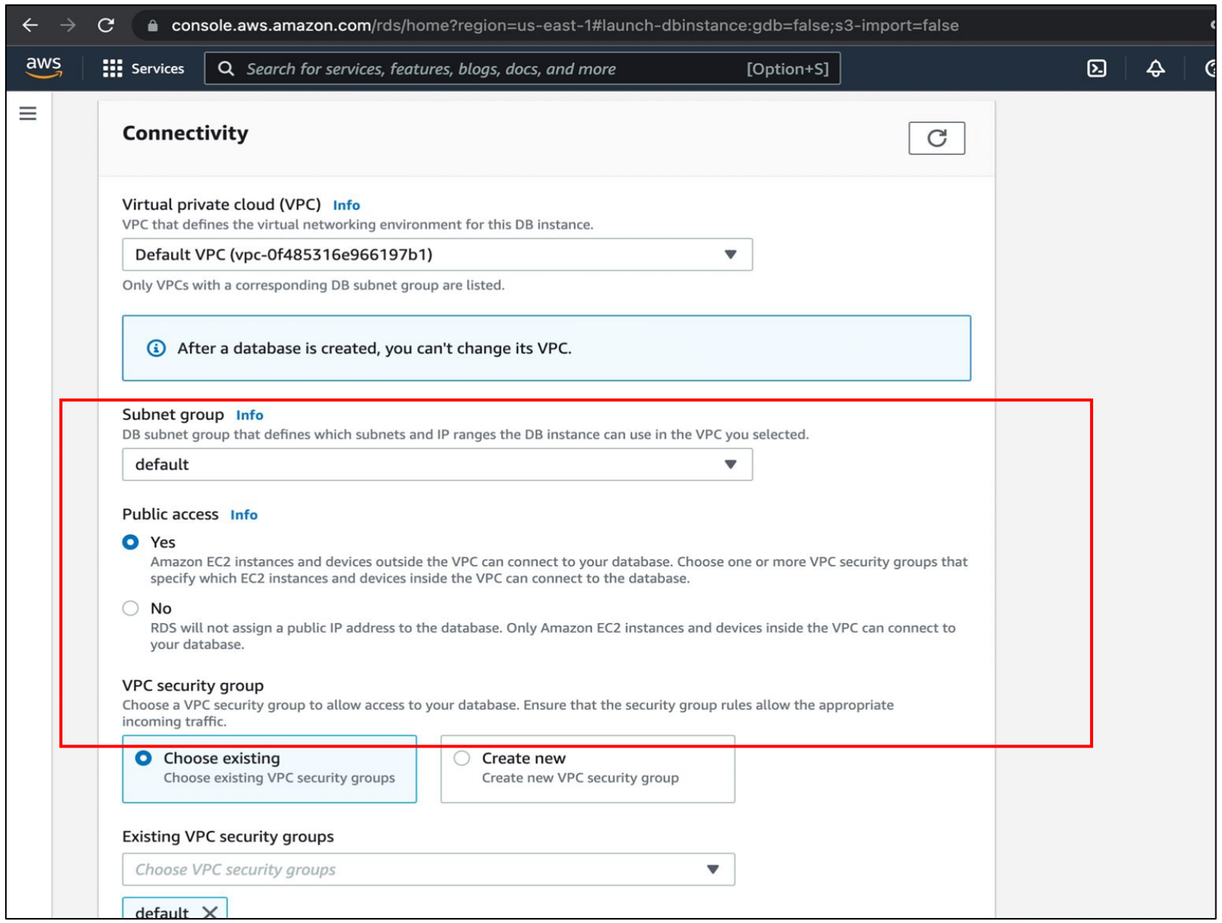
# Create MySQL Database

In the **settings,** provide a name, username, and password for the database.



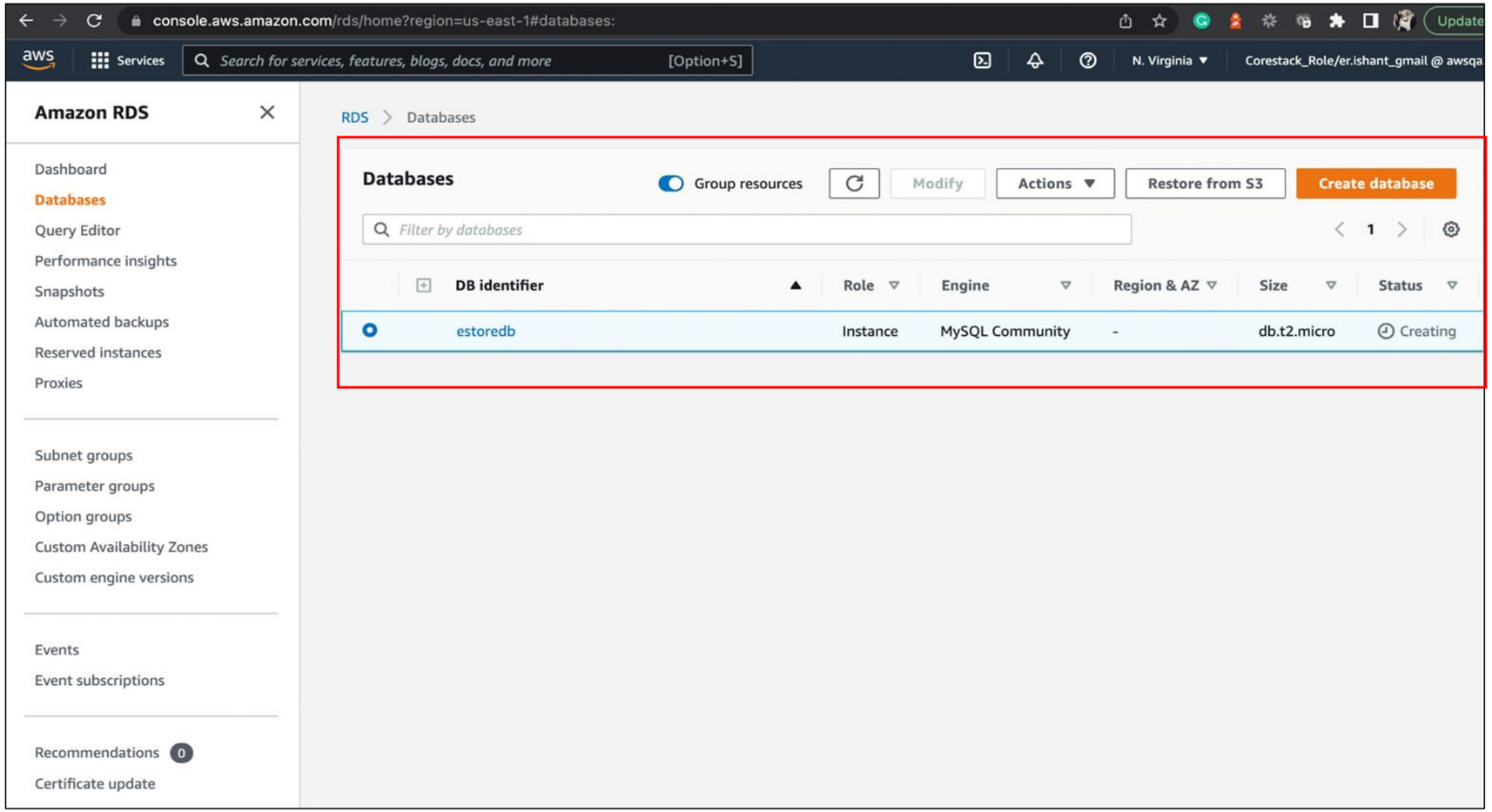
# Create MySQL Database

On the **Connectivity** page, in the **Public access** section, select **Yes** and create the database.



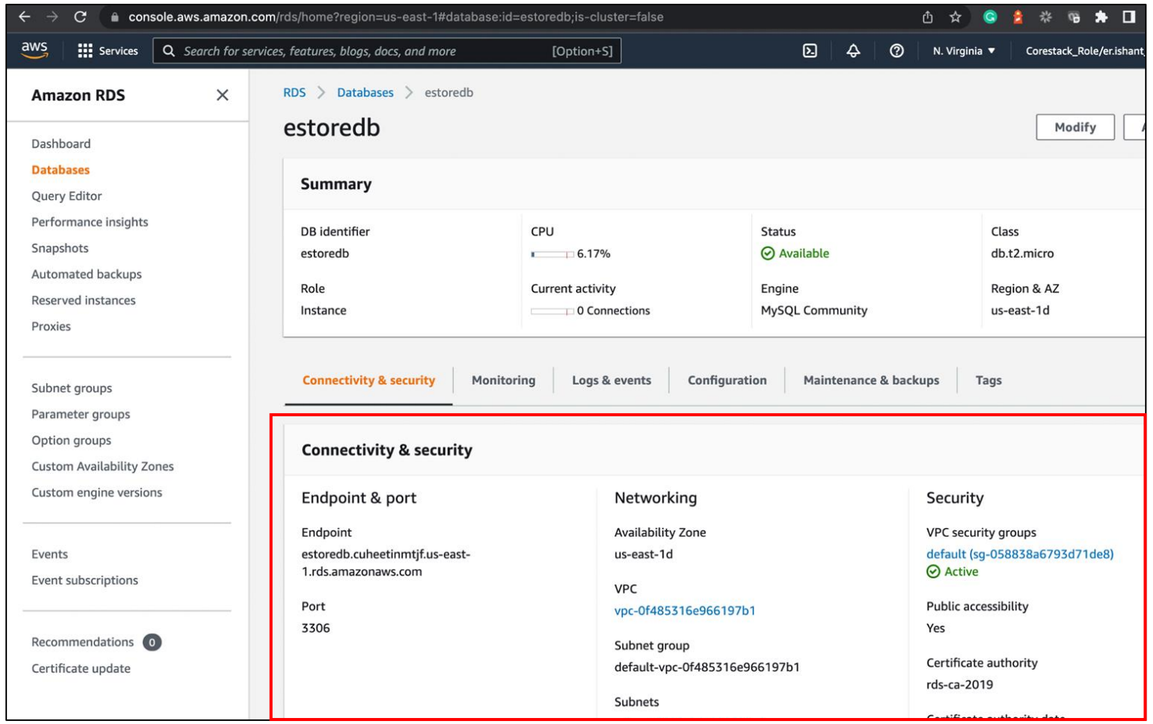
# Create MySQL Database

The database is **ready.**



# Create MySQL Database

Click the **Connectivity & security** tab to note the end point and port numbers.
Use the same numbers in the application.properties file of your Java back-end project.




Do not forget to connect to MySQL and create the estoredb database manually.

Caltech | Center for Technology & Management Education

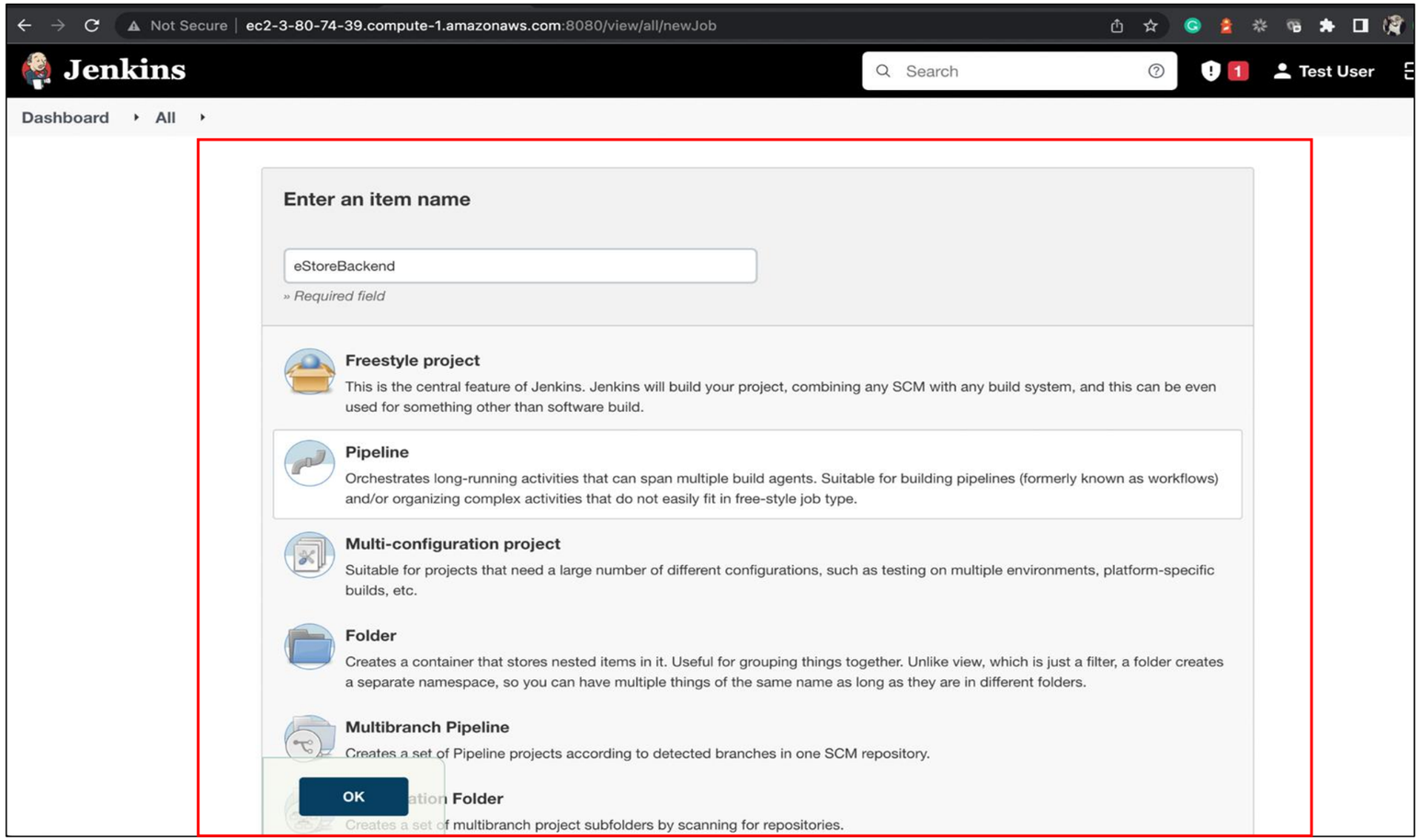
# Build and Deploy with Jenkins Docker

Powered by simplilearn

# Set Up Jenkins for the Java Backend Project

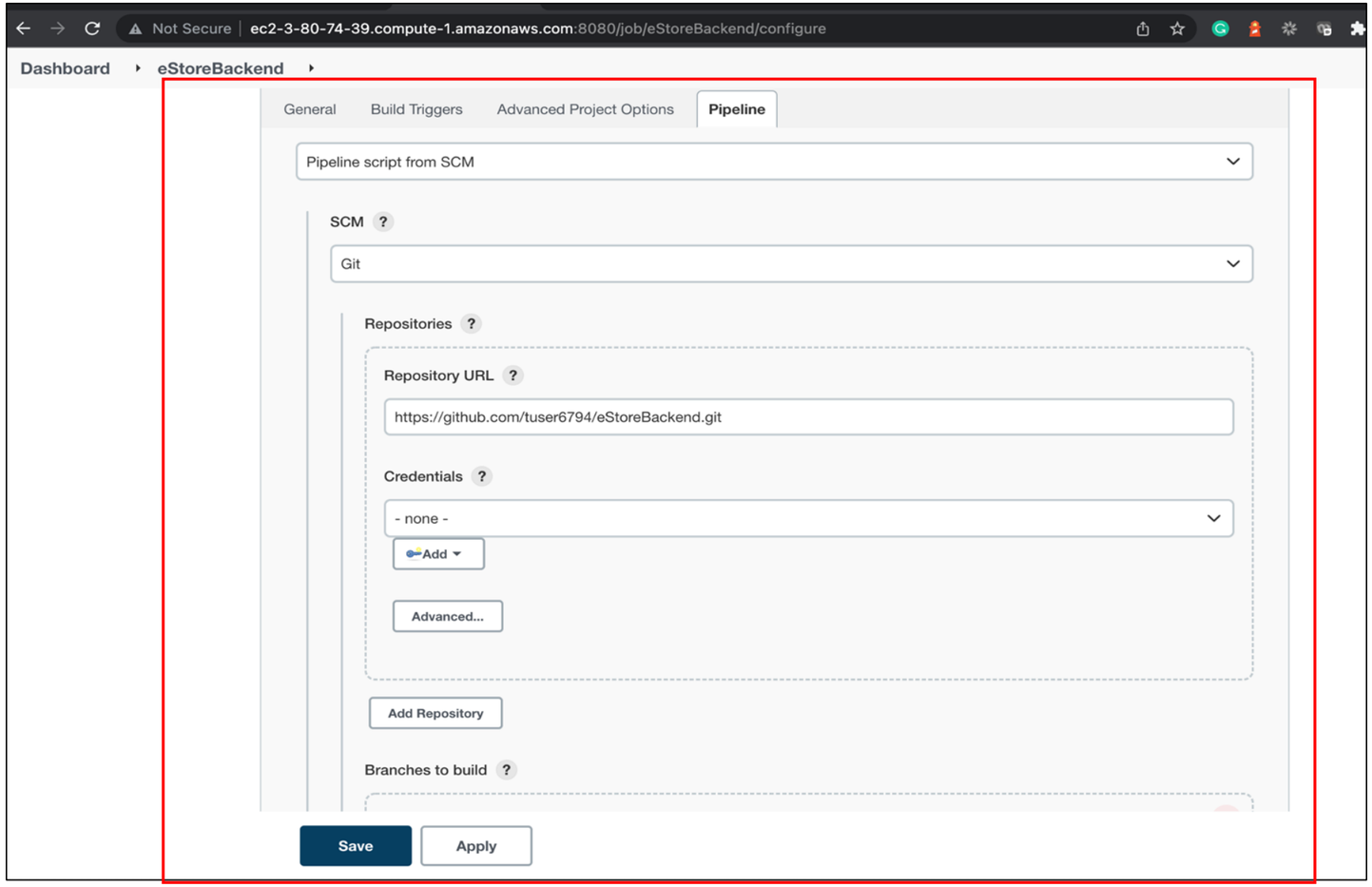
# Create a Jenkins Pipeline Project for the Java Backend Project

Create a new project in Jenkins of type **Pipeline.**



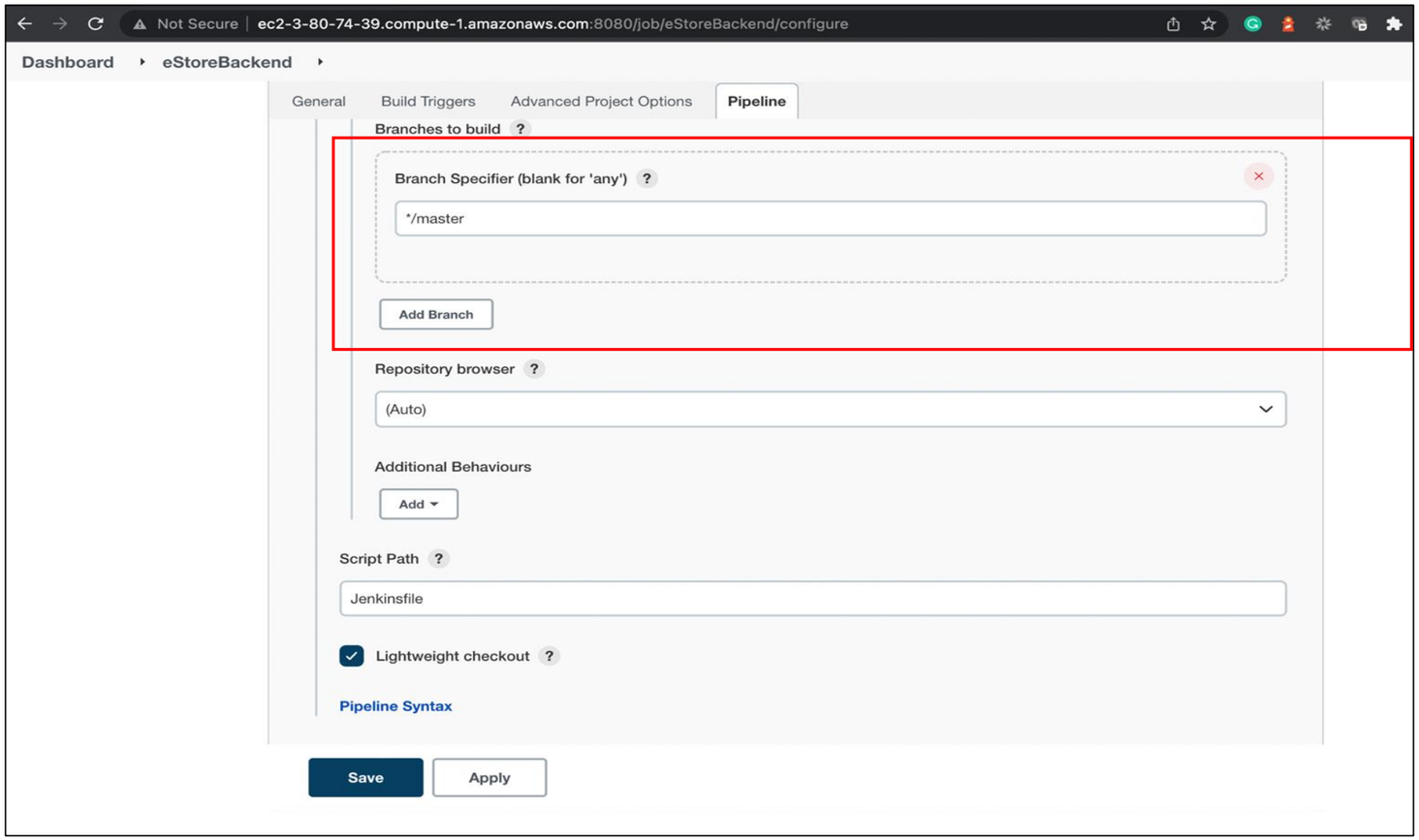
# Create a Jenkins Pipeline Project for the Java Backend Project

**Configure** the Jenkins Pipeline Project by passing **GitHub repository URL**.



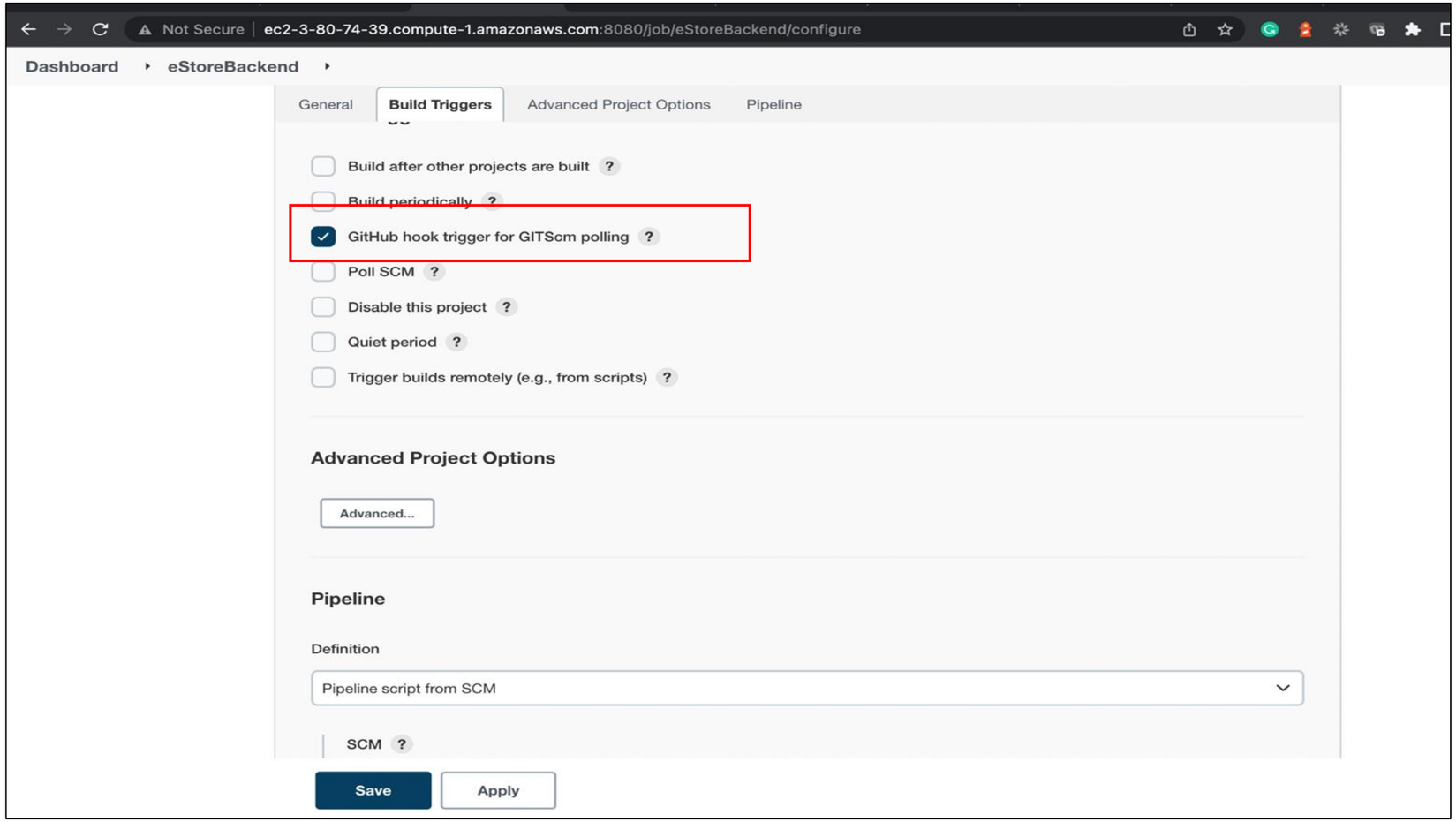
# Create a Jenkins Pipeline Project for the Java Backend Project

From the branch specifier, select the **default branch**, which is master. The Script Path contains the Jenkinsfile, which is created in the root directory of our project.



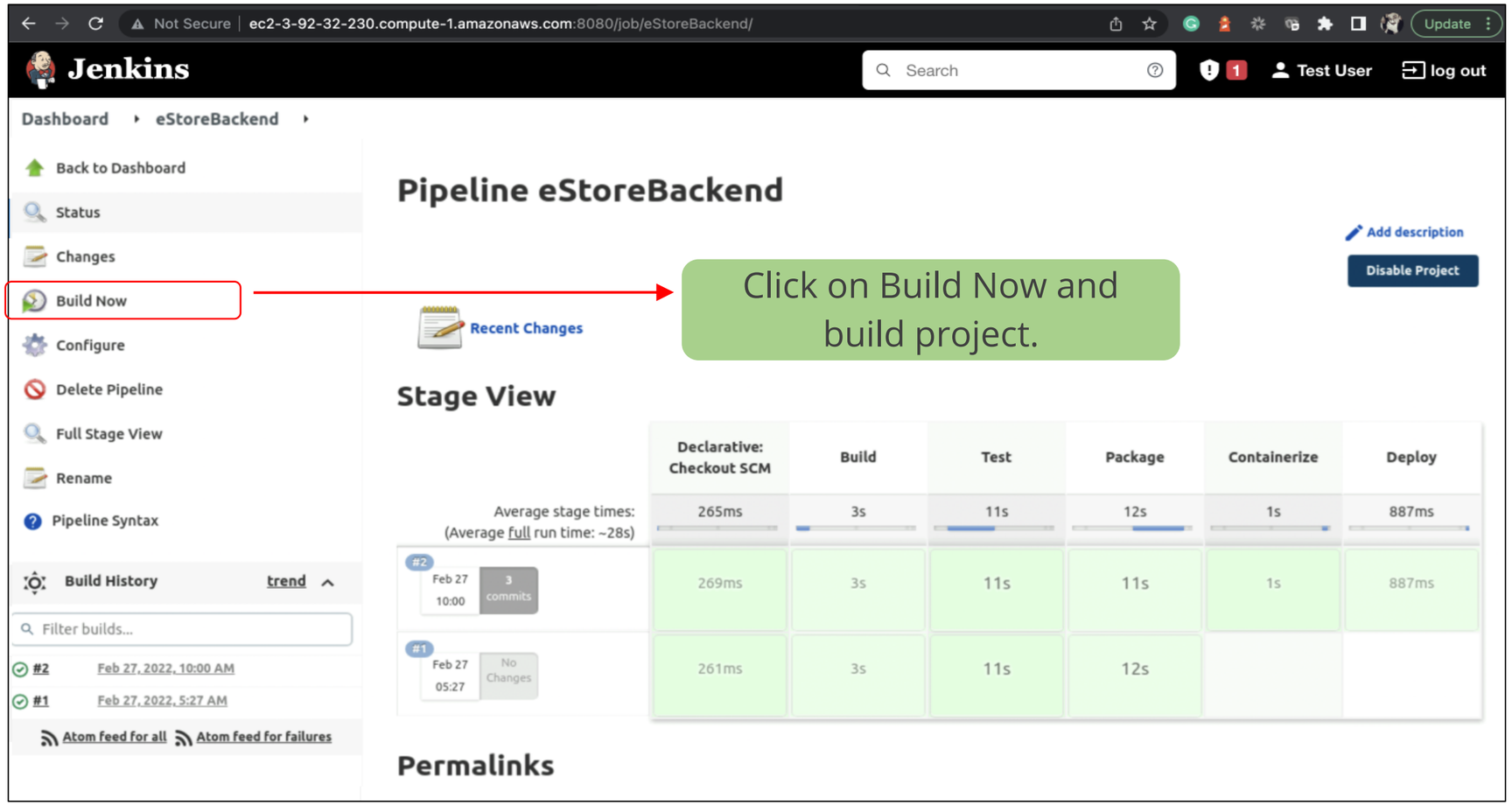
# Optional: GitHub Hook Trigger for the Java Backend Project

Configure **GitHub trigger** for GITScm Polling. Users can configure the same using ngrok.



# Run the Jenkins Pipeline Project for the Java Backend Project

Notice stages appearing as mentioned in the Jenkinsfile.
**Source > Test > Build > Containerize > Deploy**



# Key Takeaways

- EC2 instances is created and configured on AWS.
- MySQL RDS database is created on AWS.
- Jenkins Pipeline is built to dockerize the Java backend.
- Docker is integrated in Jenkins to build and release the images as containers on EC2.