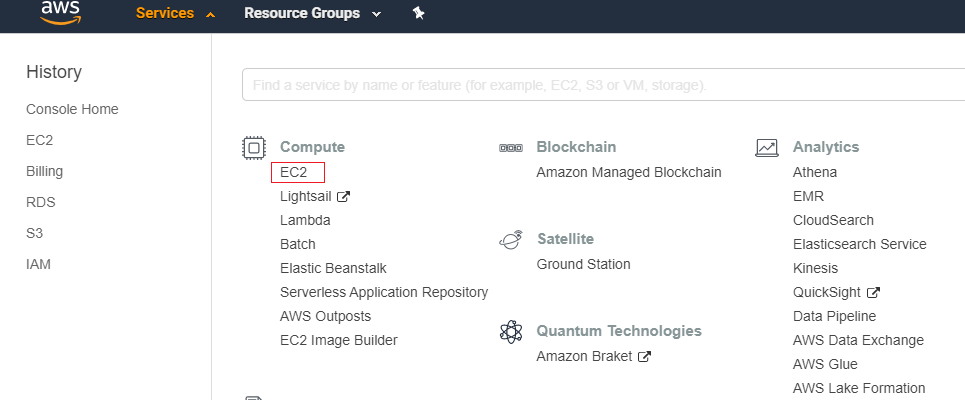
# **Food Delivery Application Deployment on AWS using Jenkins**

# **1. Deploying Spring Boot Application on AWS using Jenkins**

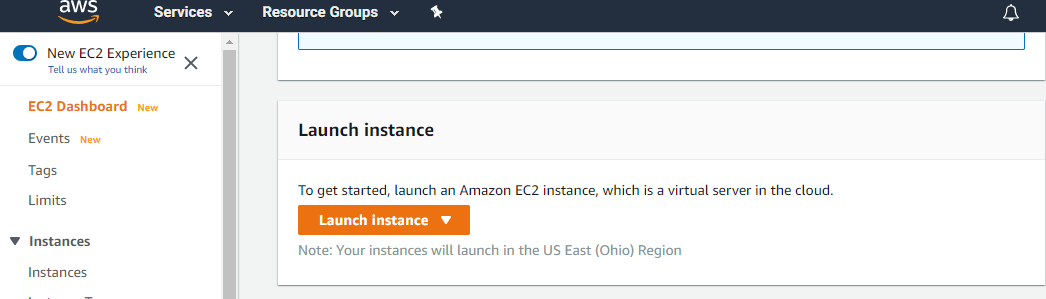
## 1.1 Launch an AWS EC2 Instance

**Step 1)** Sign in to the preview version of the AWS Management Console.

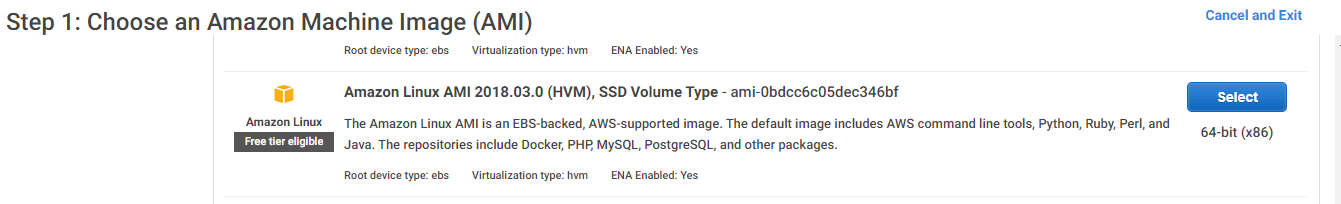
**Step 2)** Open the Amazon EC2 console by choosing EC2 under compute. If you are using Show All Services view, your screen looks like this:



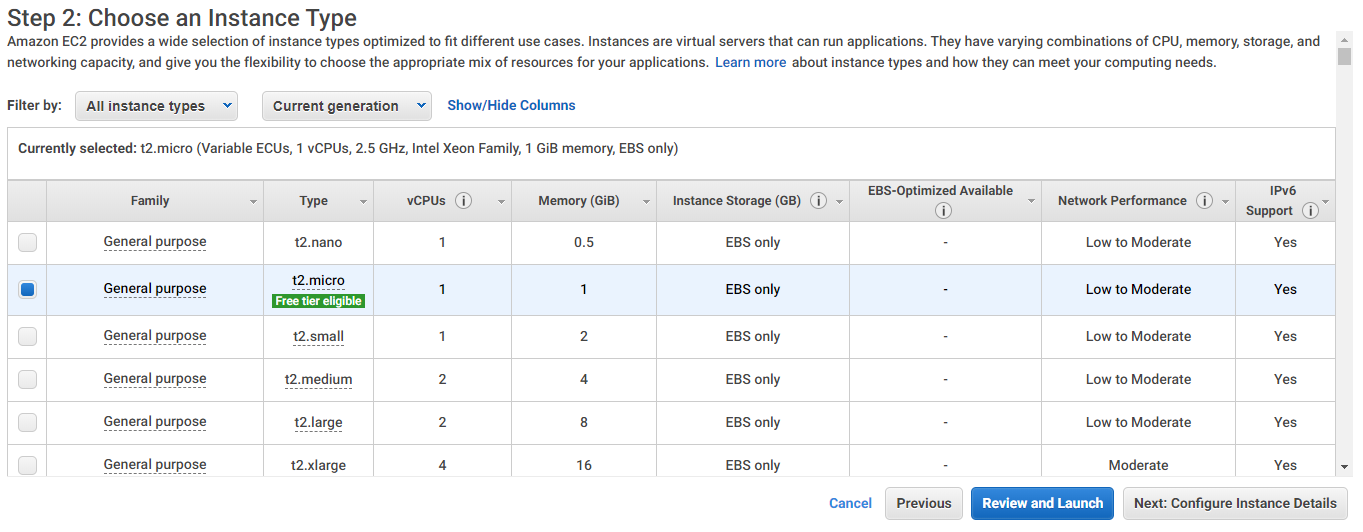
**Step 3)** From the Amazon EC2 dashboard, Choose Launch Instance.



**Step 4)** The choose an Amazon Machine Image (AMI) page displays a list of basic configurations called Amazon Machine Images (AMIs) that serve as templates for your instance. Select the Amazon Linux AMI. Notice that this configuration is marked Free tier eligible.

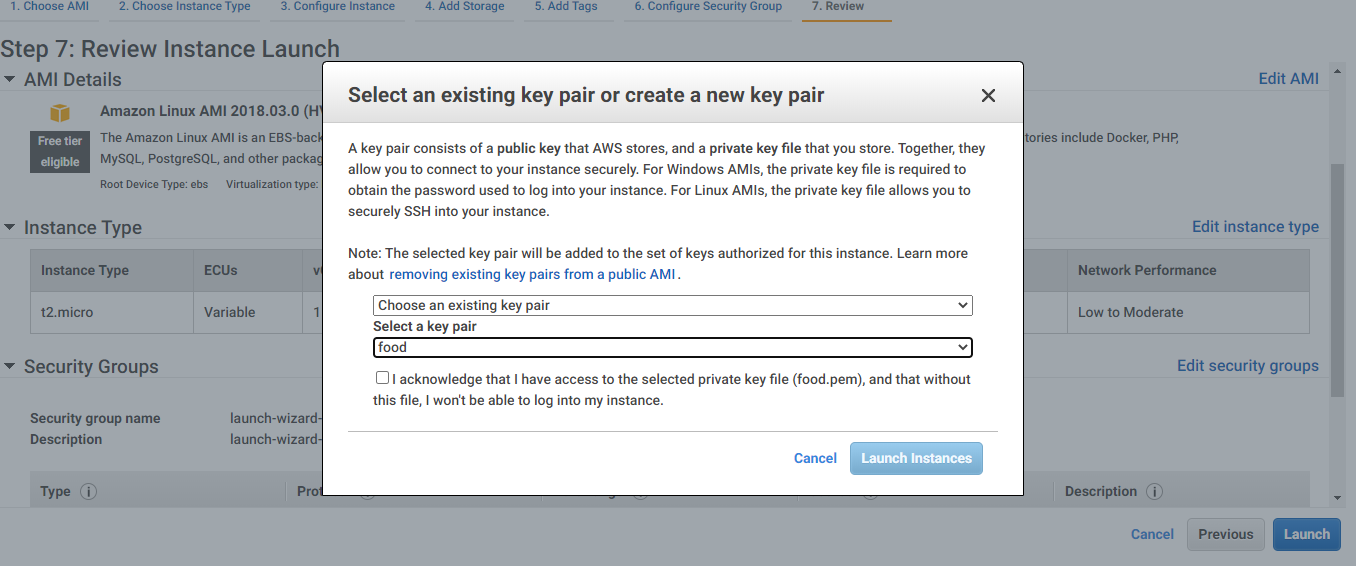


**Step 5)** On the Choose an Instance Type page, choose t.2micro as the hardware configuration of your instance and Review and Launch.

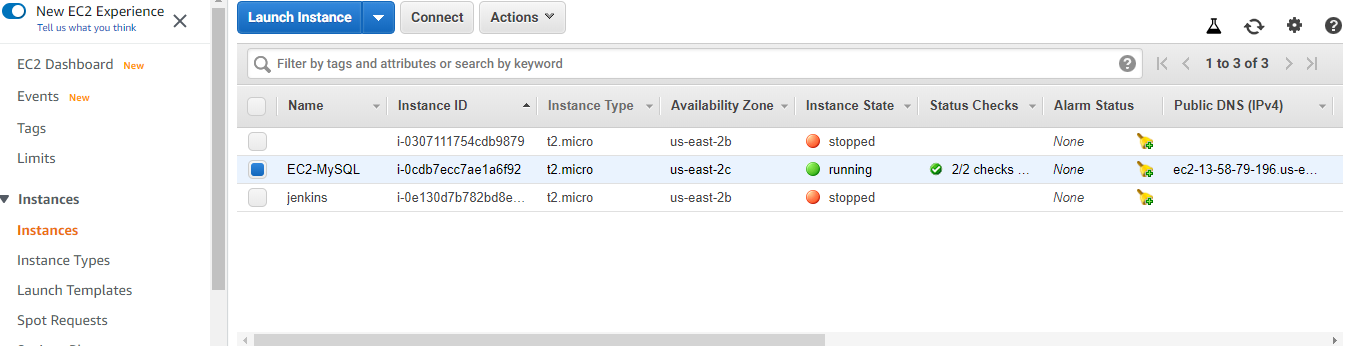


**Step 6)** On the Review Instance Launch page, choose Launch.

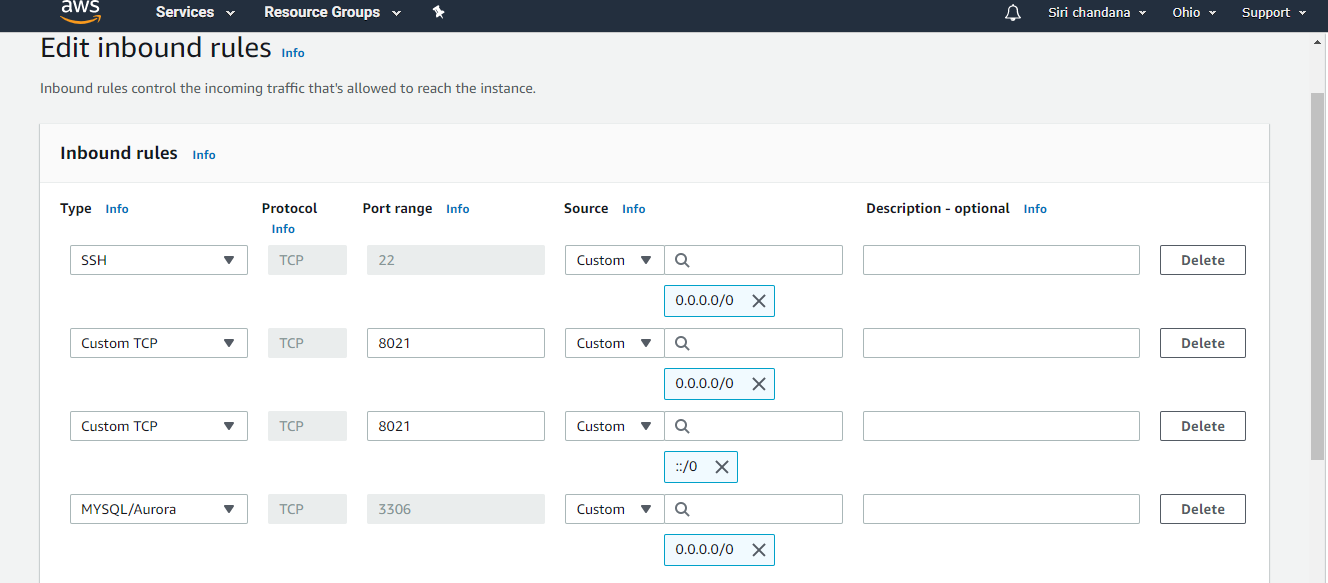
**Step 7)** In the Select an existing key pair or create a new key pair dialog box, choose Create a new key pair, enter a name for the key pair and then choose Download Key Pair.



**Step 8)** Once the instance is ready and running, it will appear as below



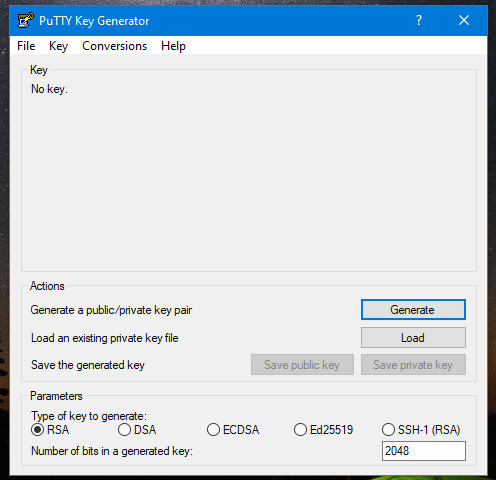
**Step 9)** Go to Security group and add the rules for MySQL and Tomcat port (port number of your application) as below



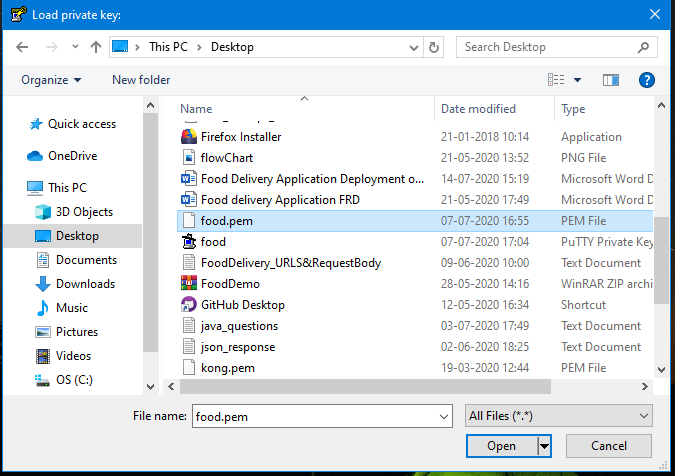
## 1.2 Connecting to Instance using PuTTY

**Step 1)** Go to <https://www.putty.org/> and download the PuTTY or PuTTYgen.

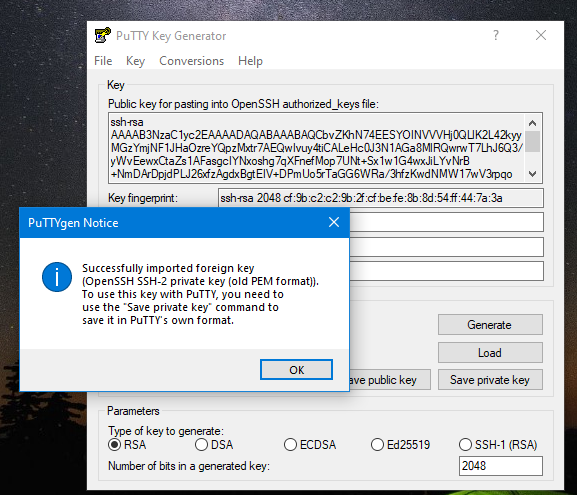
**Step 2)** Click on Start menu -> All Programs -> PuTTY -> PuTTYgen.



**Step 3)** Next, click on the option ‘Load’. As PuTTY supports its native file format, it will only show files that have .ppk file extension. Therefore, users have to choose the ‘All Files’ option from the drop-down bar. It will display all key files included the .pem file.

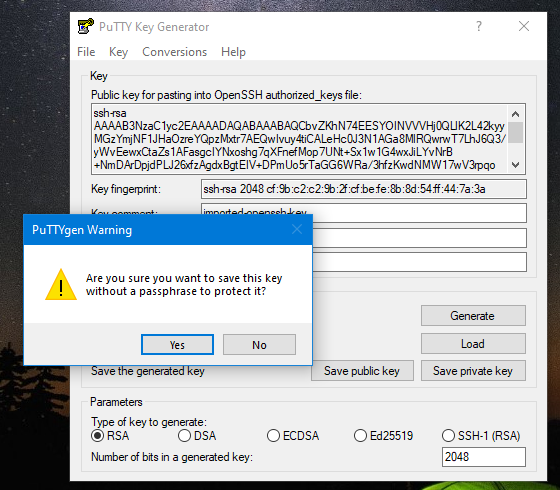


**Step 4)** Now select the .pem file that you want to convert. As PuTTYgen is used for SSH connectivity, so it crucial for users to select the specific file that they plan to convert and click ‘Open’. To confirm, click on ‘OK”.



**Step 5)** In the resultant window,click on ‘Save private key’ which will convert and save the key file in PuTTY compatible format.

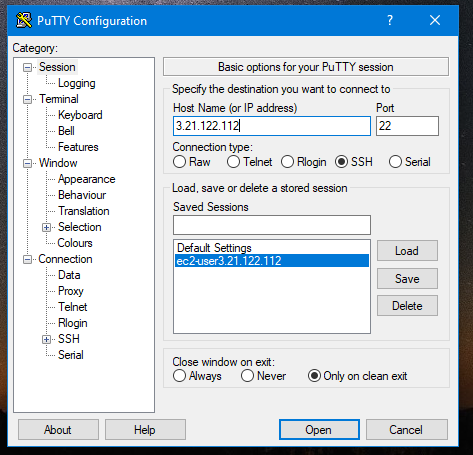
**Step 6)** PuTTYgen will prompt a warning of saving the key without a passphrase. Hit ‘Yes’ on it.



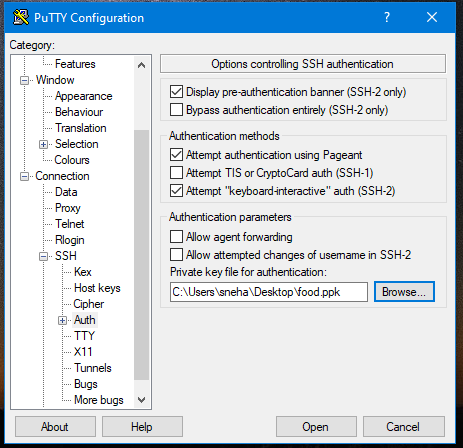
**Step 7)** Now, give the name to your file and PuTTYgen will automatically add .ppk file extension.

**Step 8)** Once the .pem file is converted .ppk then users can connect to remote hosts using PuTTY’s SSH client.

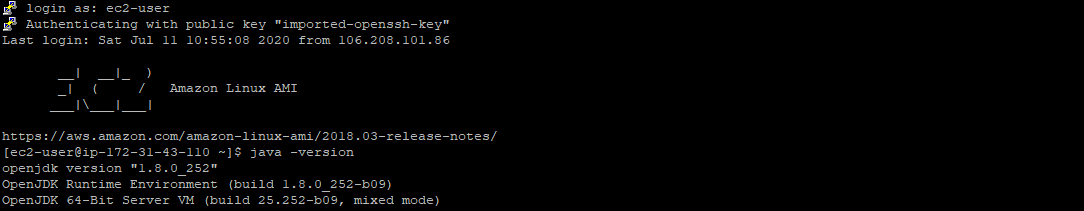
**Step 9)** First, open PuTTY and input the host IP address i.e. Public IP address of EC2 instance.



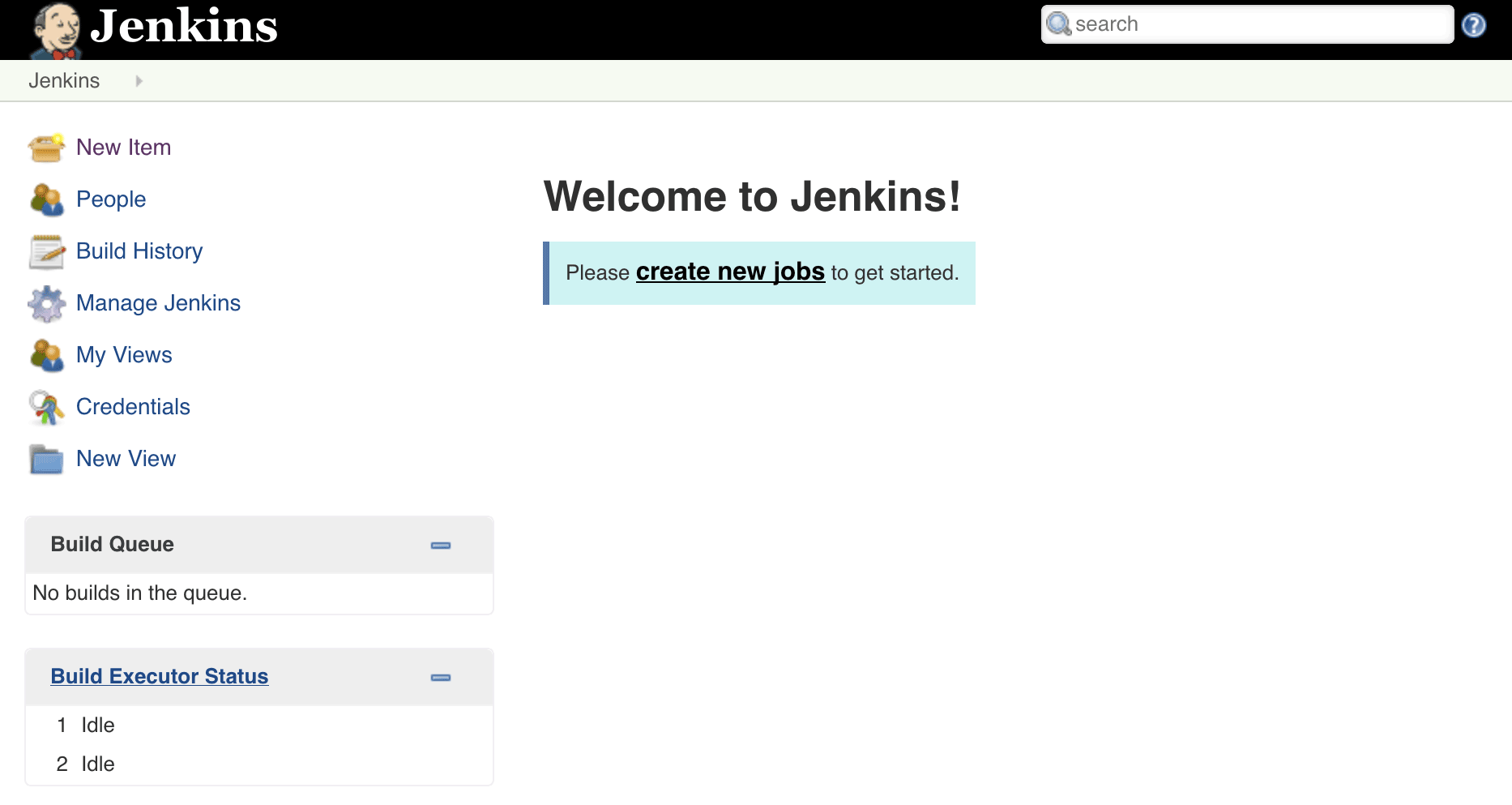
**Step 10)** Go to SSH -> Auth -> browse the ppk file, select the file and click open.



**Step 11)** Once the PuTTY is connected, login as ec2-user



**Step 12)** Install java, git, maven and Jenkins.

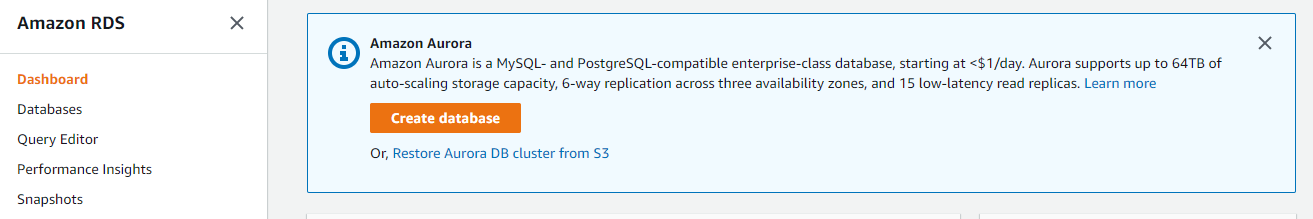


**Step 13)** Create on more ec2 t2.micro instance using above steps.

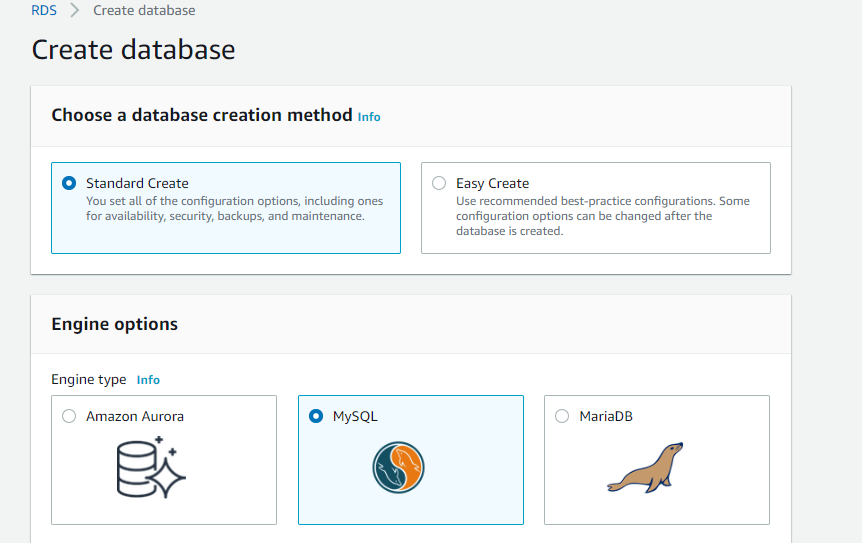
**Step 14)** Now install java connecting through PuTTY.

## 1.3 RDS Database

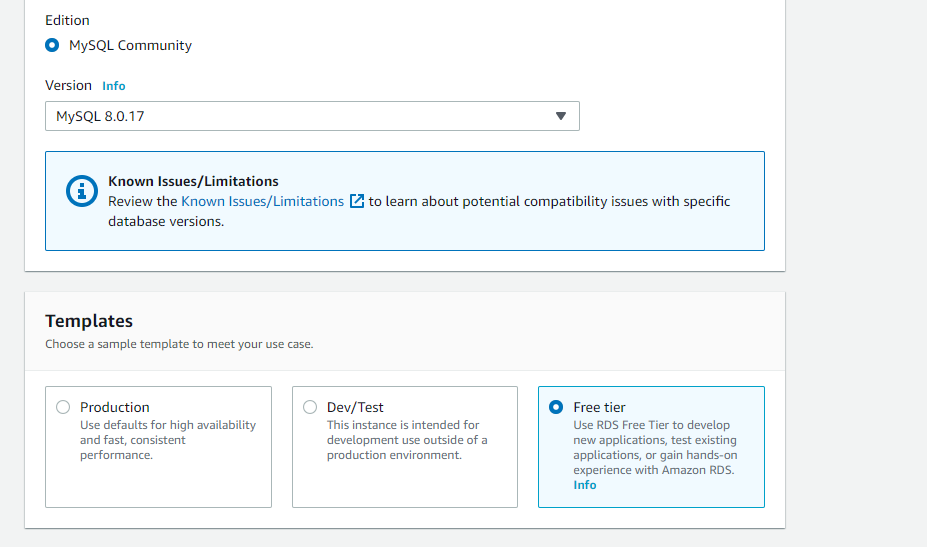
**Step 1)** Go to Amazon RDS and choose Create database.



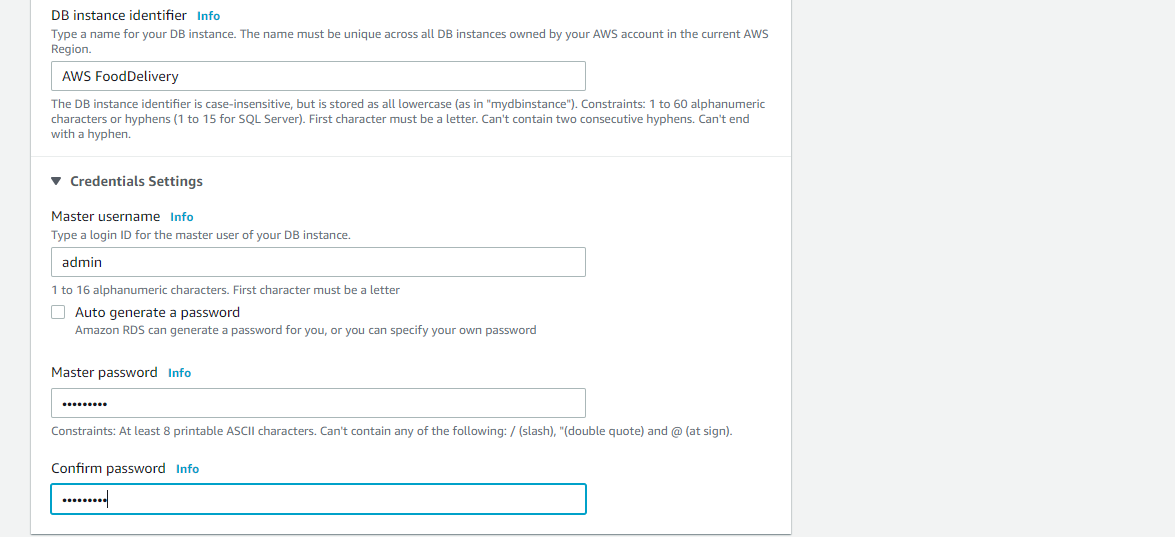
**Step 2)** Click the MySQL icon.



**Step 3)** Leave the default value of edition and engine version and select the Free Tier template.

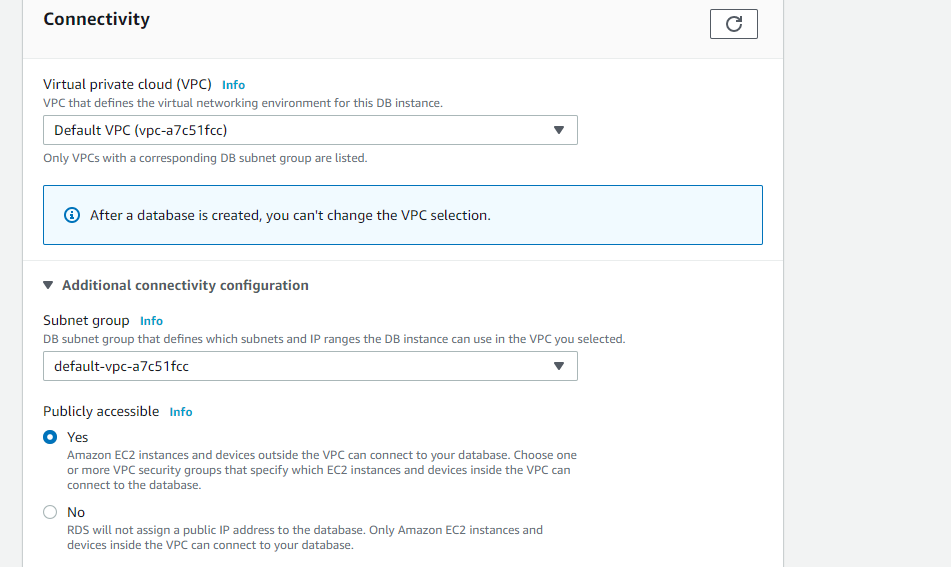


**Step 4)** Configure your DB instance by setting username and password

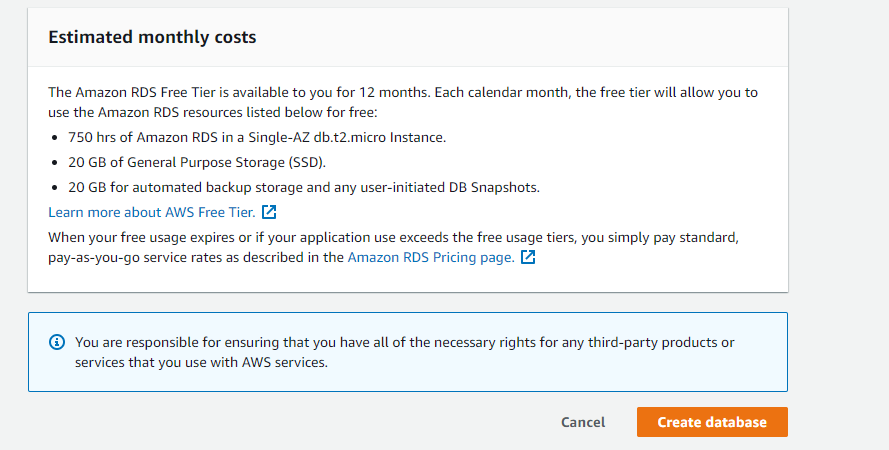


**Step 5)** In Connectivity section, make Public accessibility to Yes.

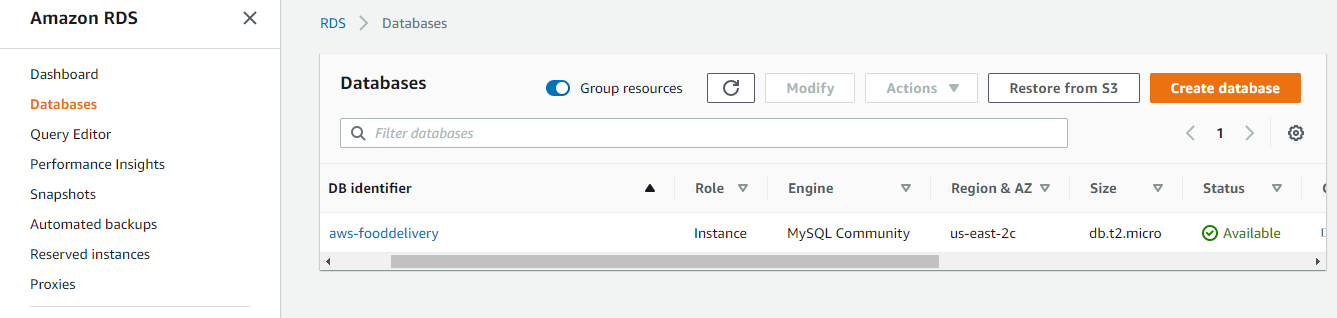
This will allocate an IP address for your database instance so that you can directly connect to the database from your own device.



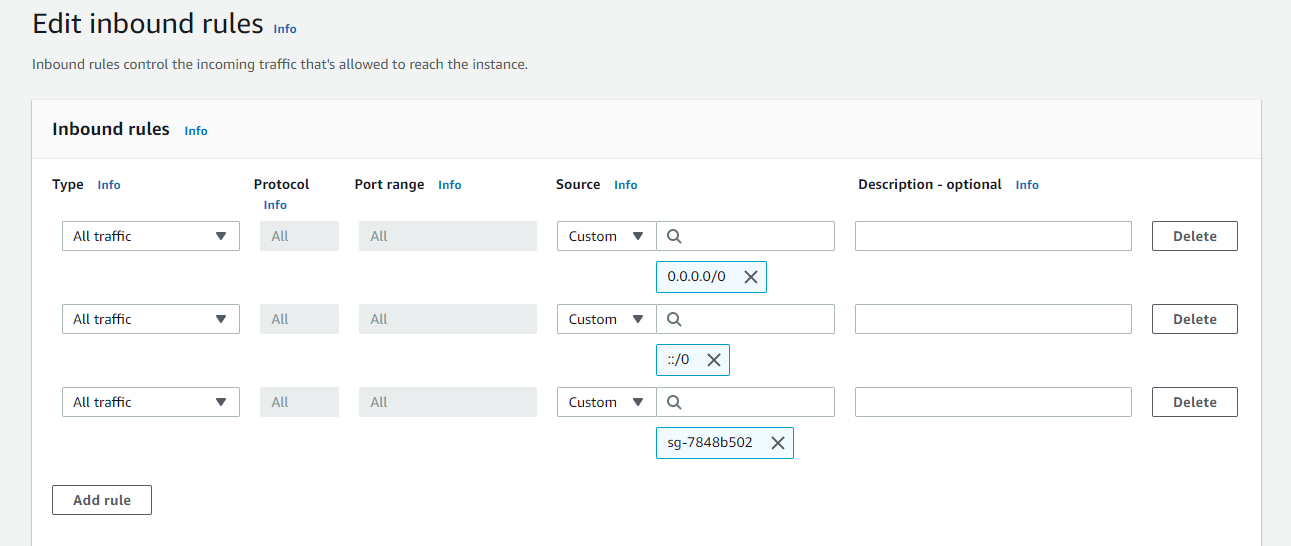
**Step 6)** Click Create Database



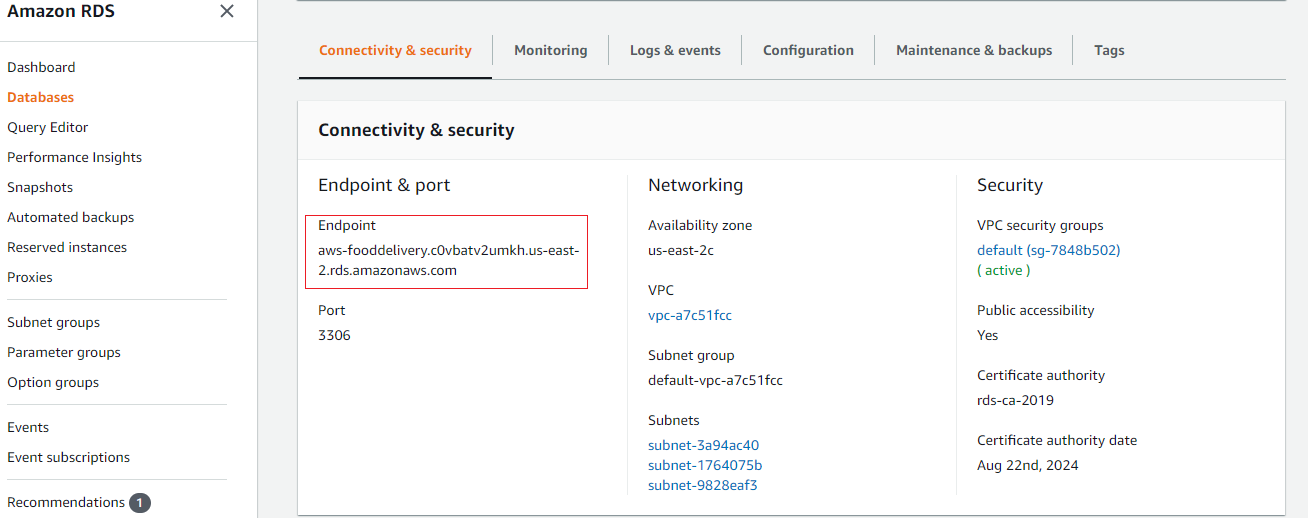
**Step 7)** Once the Database is ready, it will appear as Available



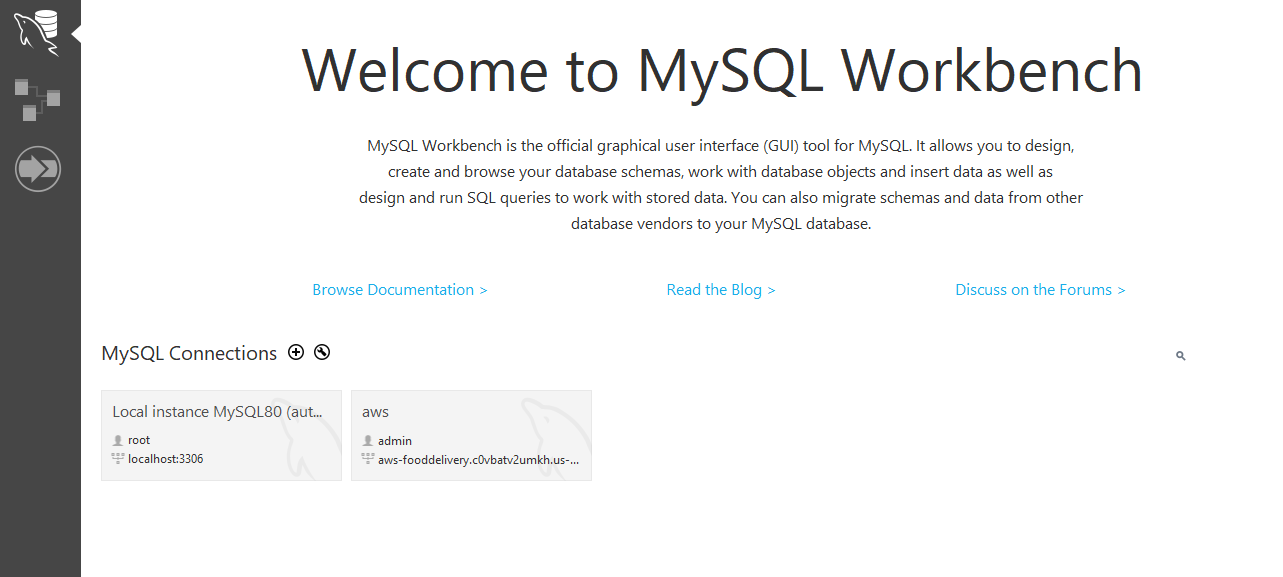
**Step 8)** Go to Security group and add rule All traffic and select the default RDS security group.



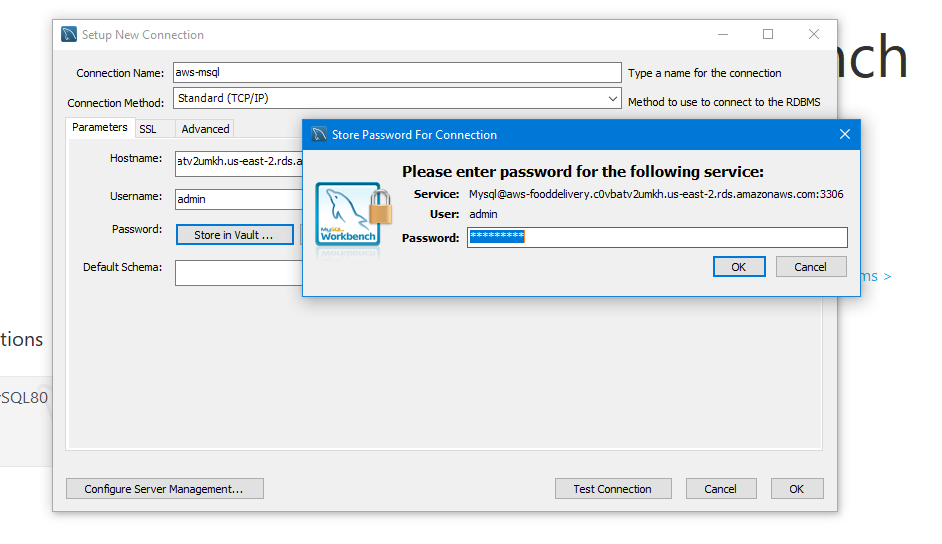
**Step 9)** Copy the Endpoint as it is used to connect the local MySQL workbench



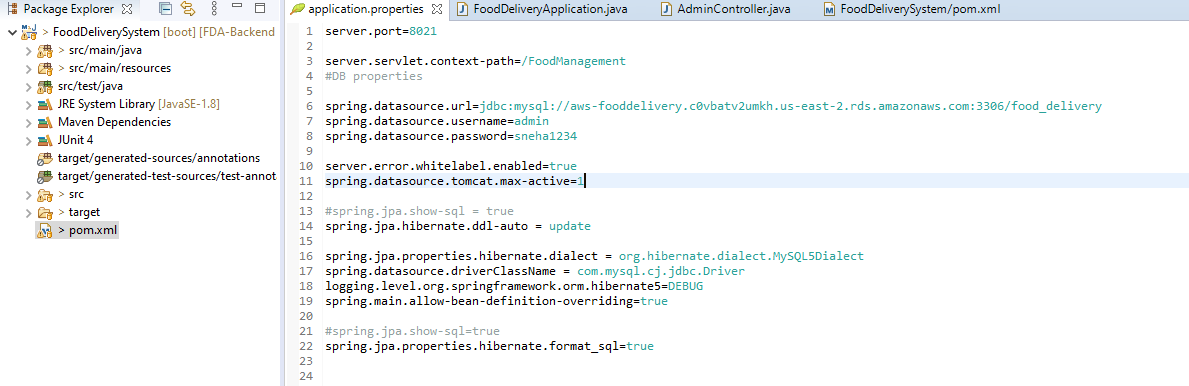
**Step 10)** Go to MySQL Workbench and in MySQL Connections add the RDS details



**Step 11)** In Hostname: give the Endpoint, provide the Username and password as per RDS database and click on Test Connection.



**Note:** In application.properties file add the RDS Mysql connections



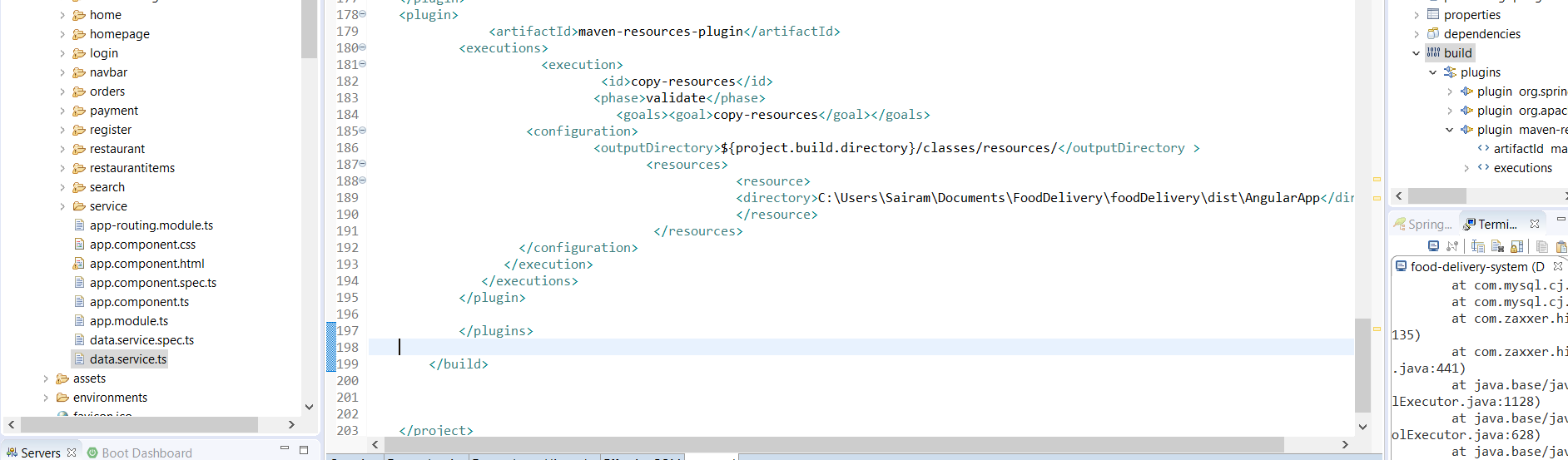
**MAKING A SINGLE JAR FILE BY MERGING BOTH FRONTEND AND BACKEND**

**Step 1)** Import the angular code to sts by using Projects from folder or archive in General

**Step 2)** generate dist folder in angular code

**Step 3 )** Check weather it is running successful by npm install and npm start

**Step 4 )** In Spring backend code you have to add an plugin for integrating UI code with backend



Here, you have to mention the directory of the dist folder that was generated in angular code

**Step 5)** Once , you done adding plugin run the application and check in console

## 1.4 Installing plugins on Jenkins

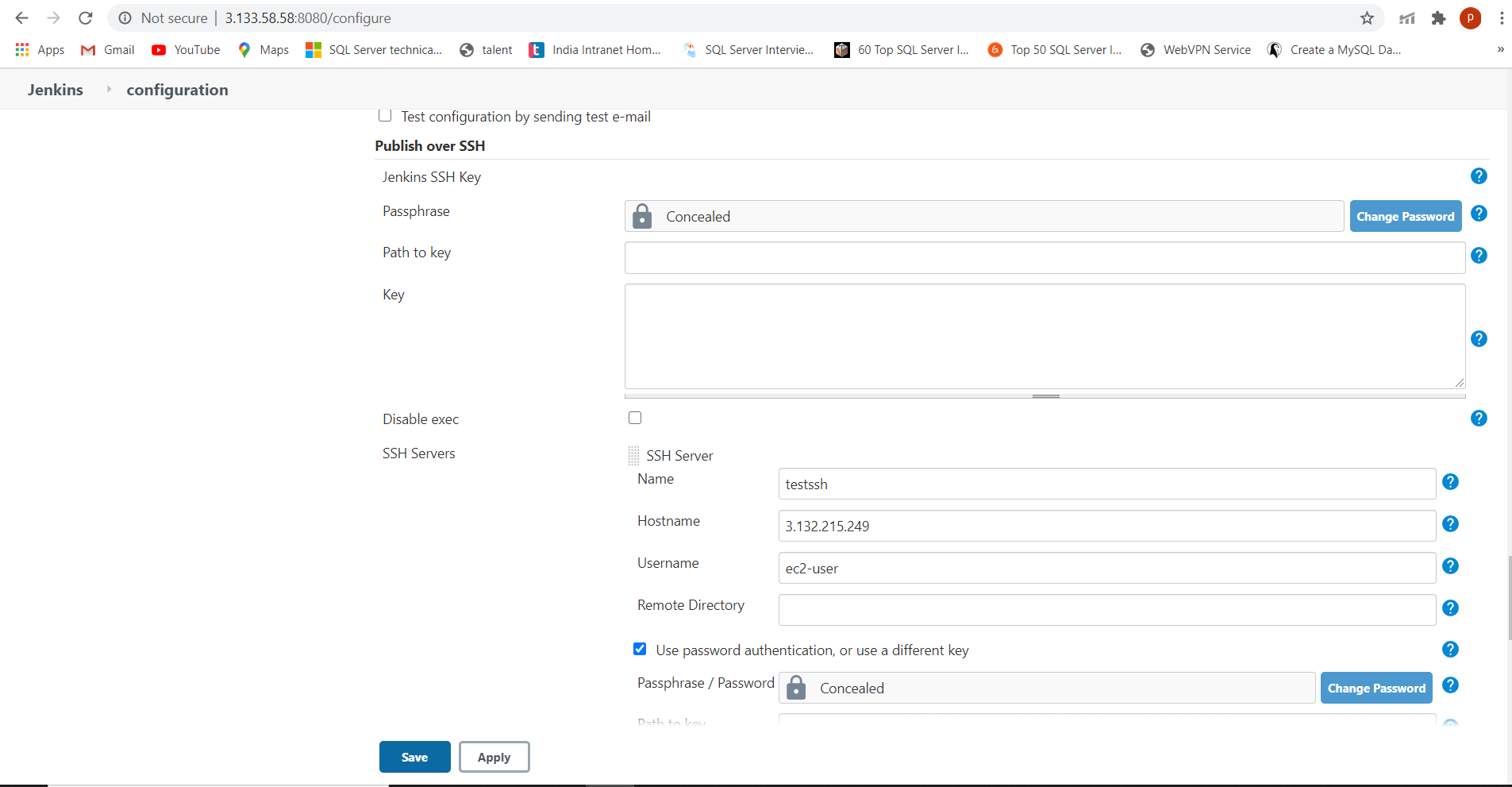
**Step 1)** In EC2 instance of Jenkins, create a job and do the configuration.

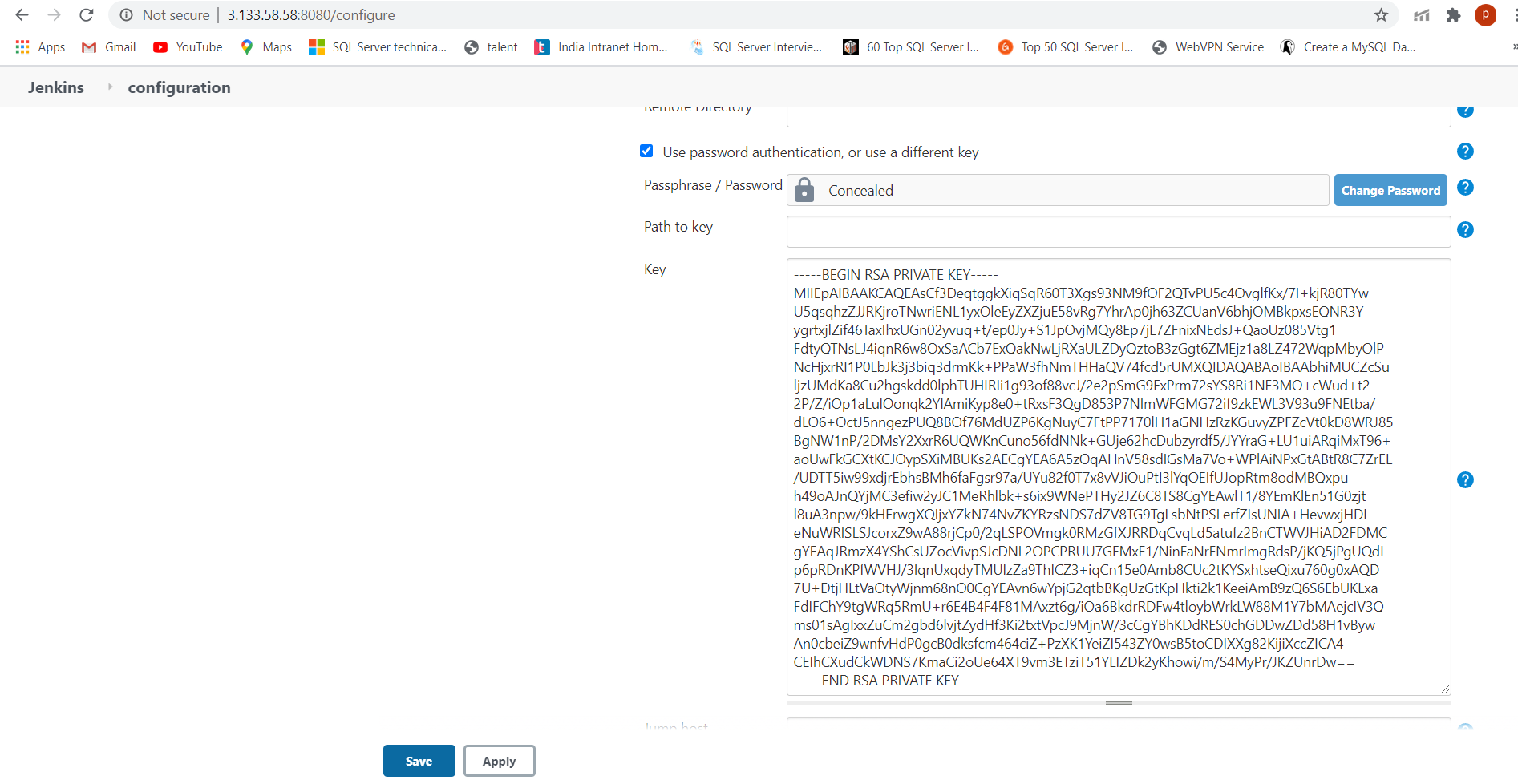
**Step 2)** Pull the code from github and generate a jar file in workspace**.**

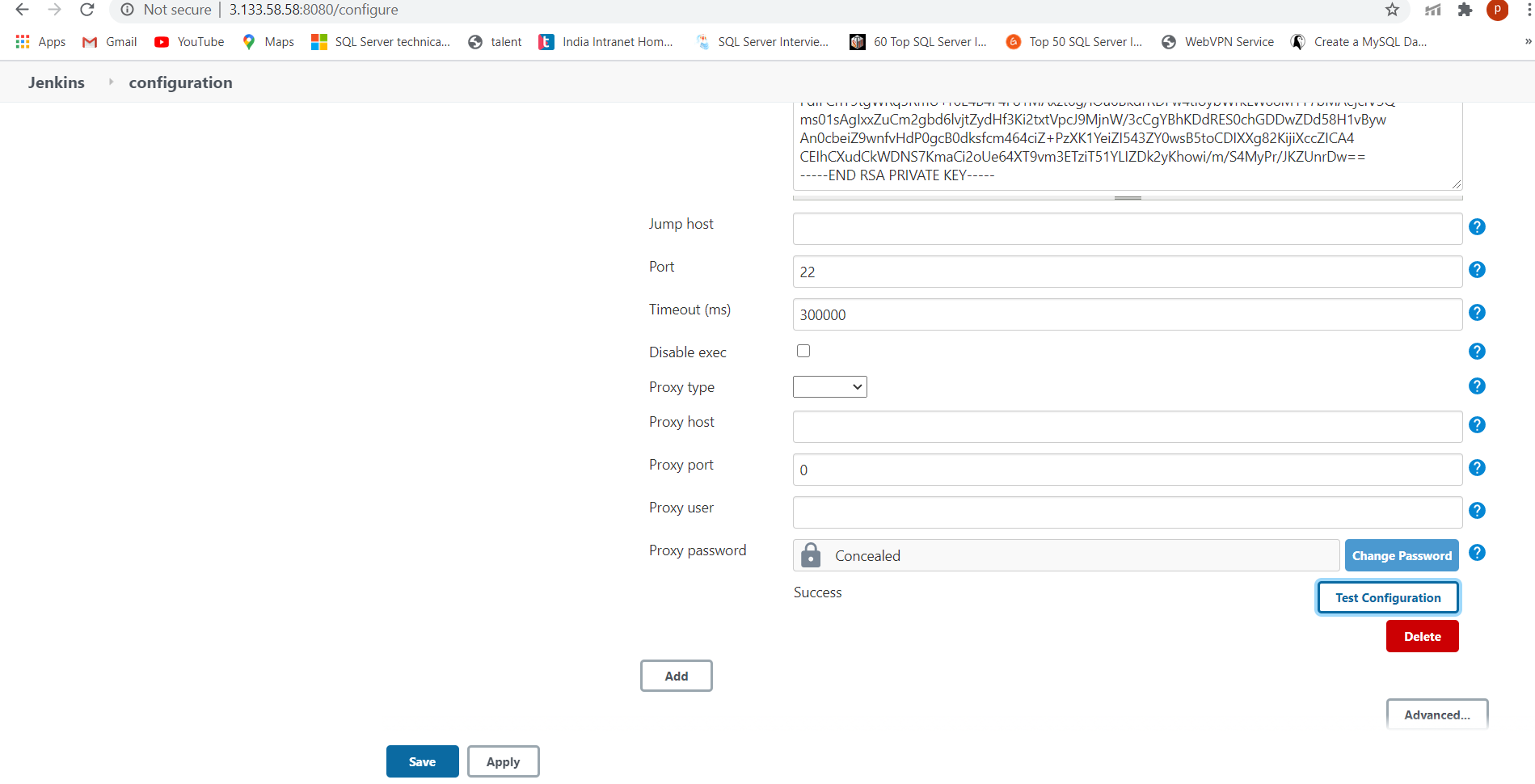
**Step 3)** Install Publish over ssh plugin

* Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available -> Publish Over SSH -> Install

**Step 4)** Under Manage Jenkins -> System Configure -> publish over ssh – give the details like below





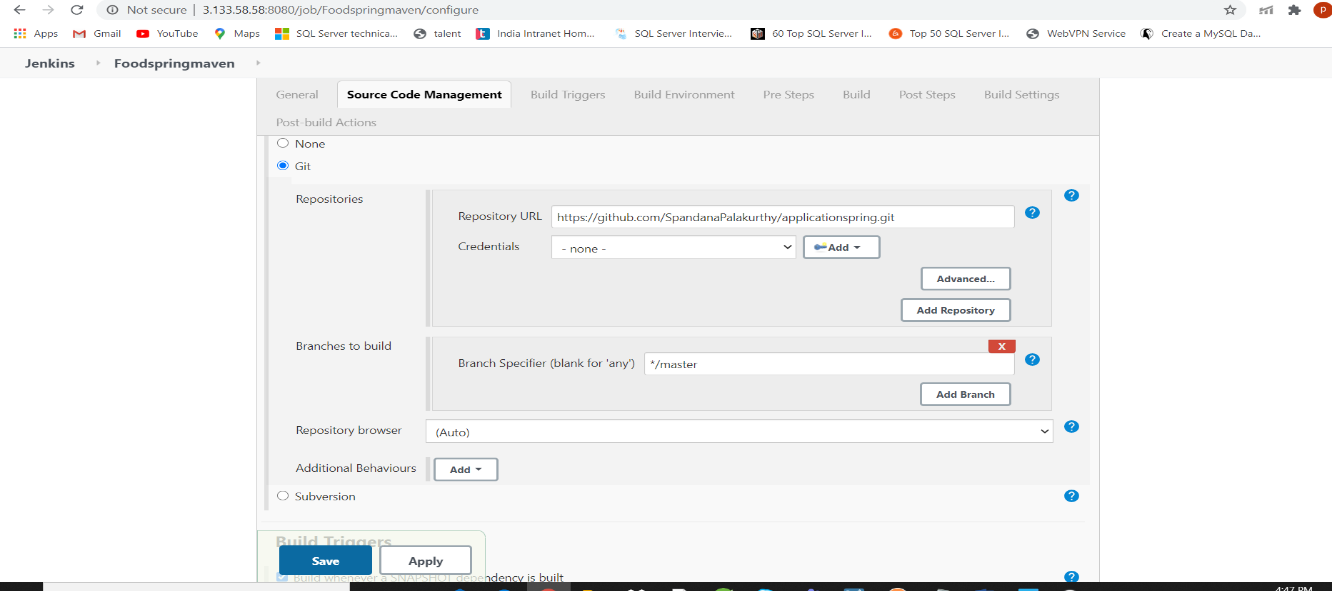


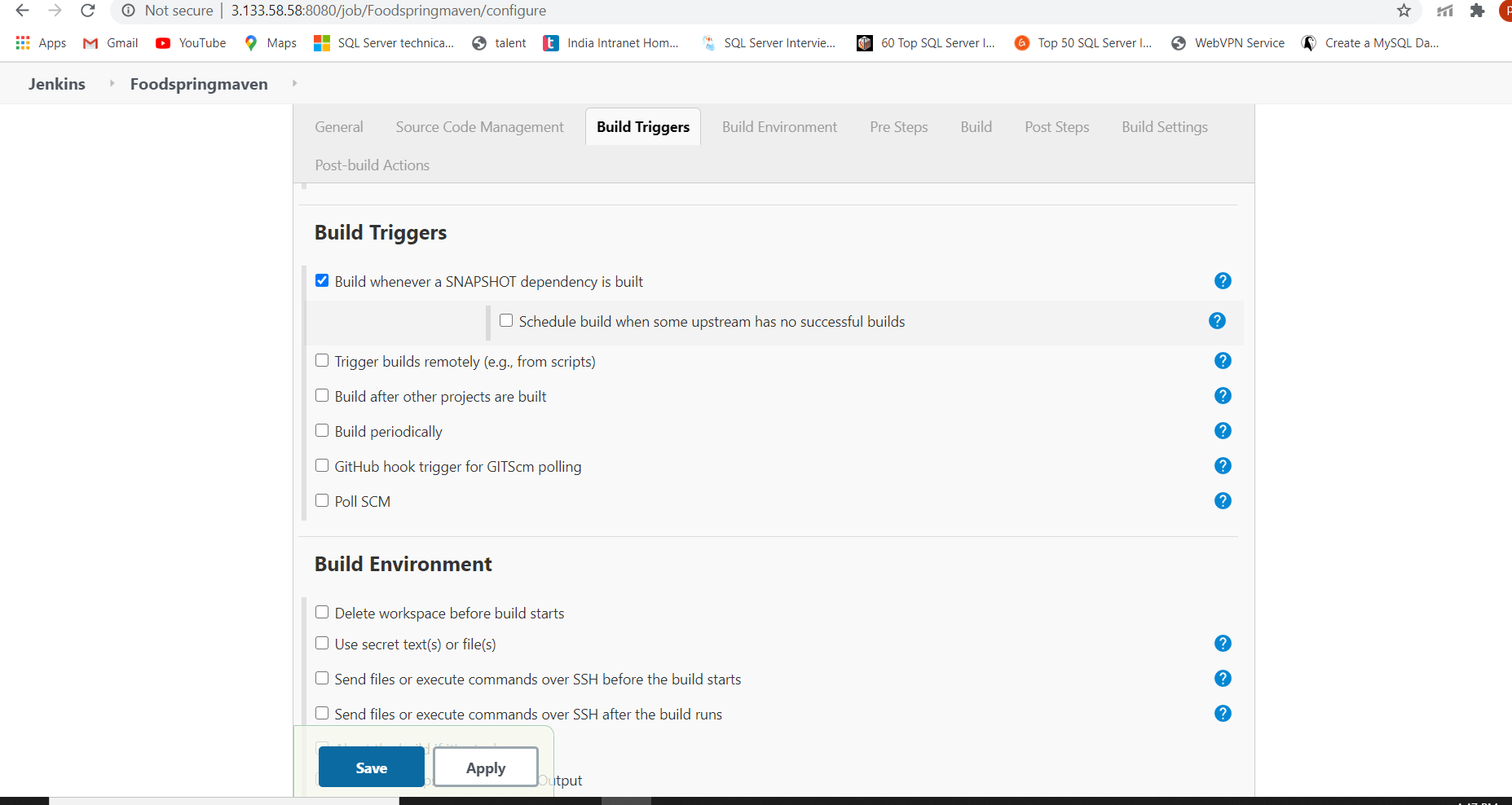
* In name content – specify the name
* Hostname – give : IPv4 Public IP addresss of second ec2 instance that u want to transfer the file
* Username : ec2-user
* Click on add server
* Give under key – pem.file data of second ec2 instance that u want to transfer the file
* Click on test configuration-- > An message should appear like success

## 1.5 Creating job Maven in Jenkins

**Step 1)** Create a job maven and give name as springfoodmaven

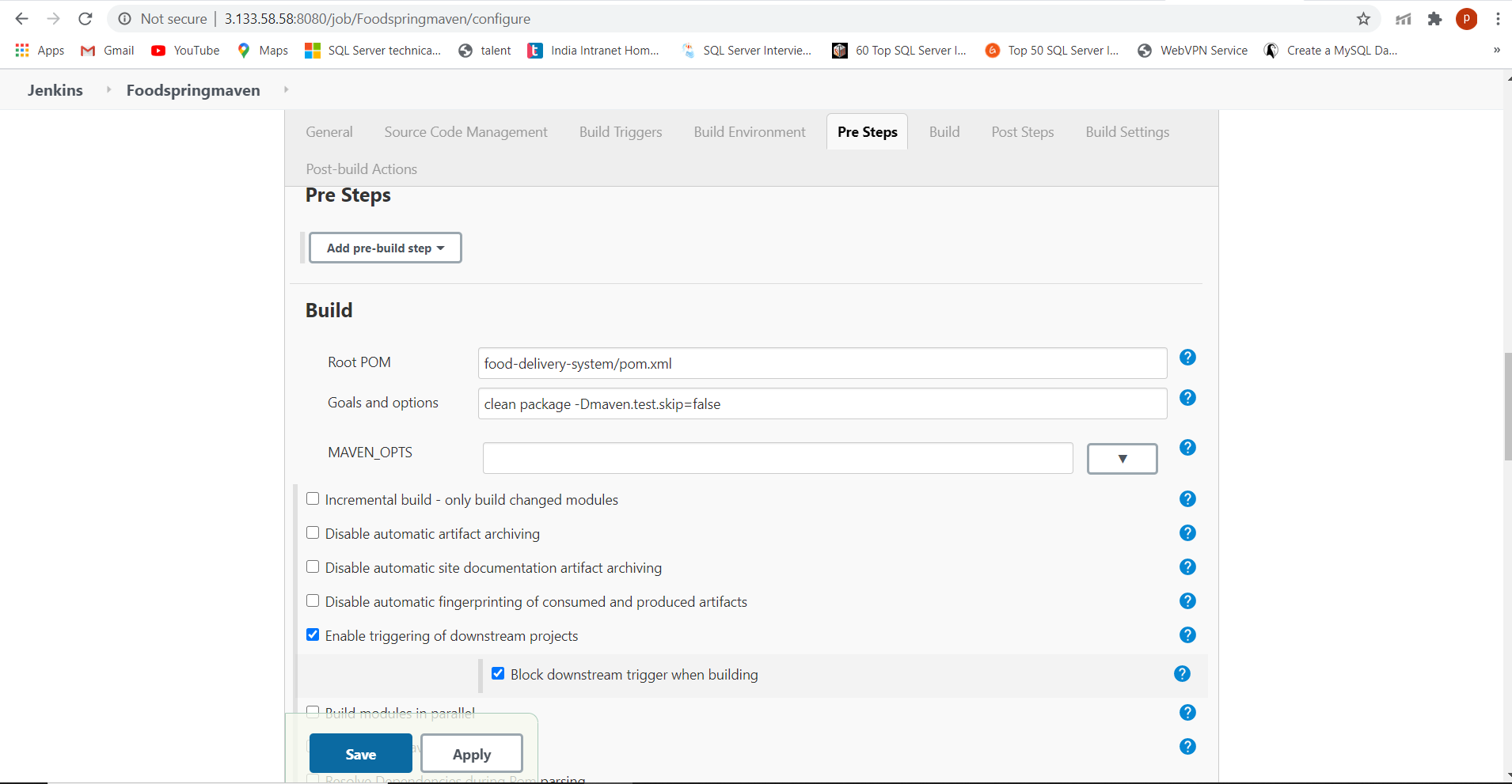
**Step 2)** Configure the details as below





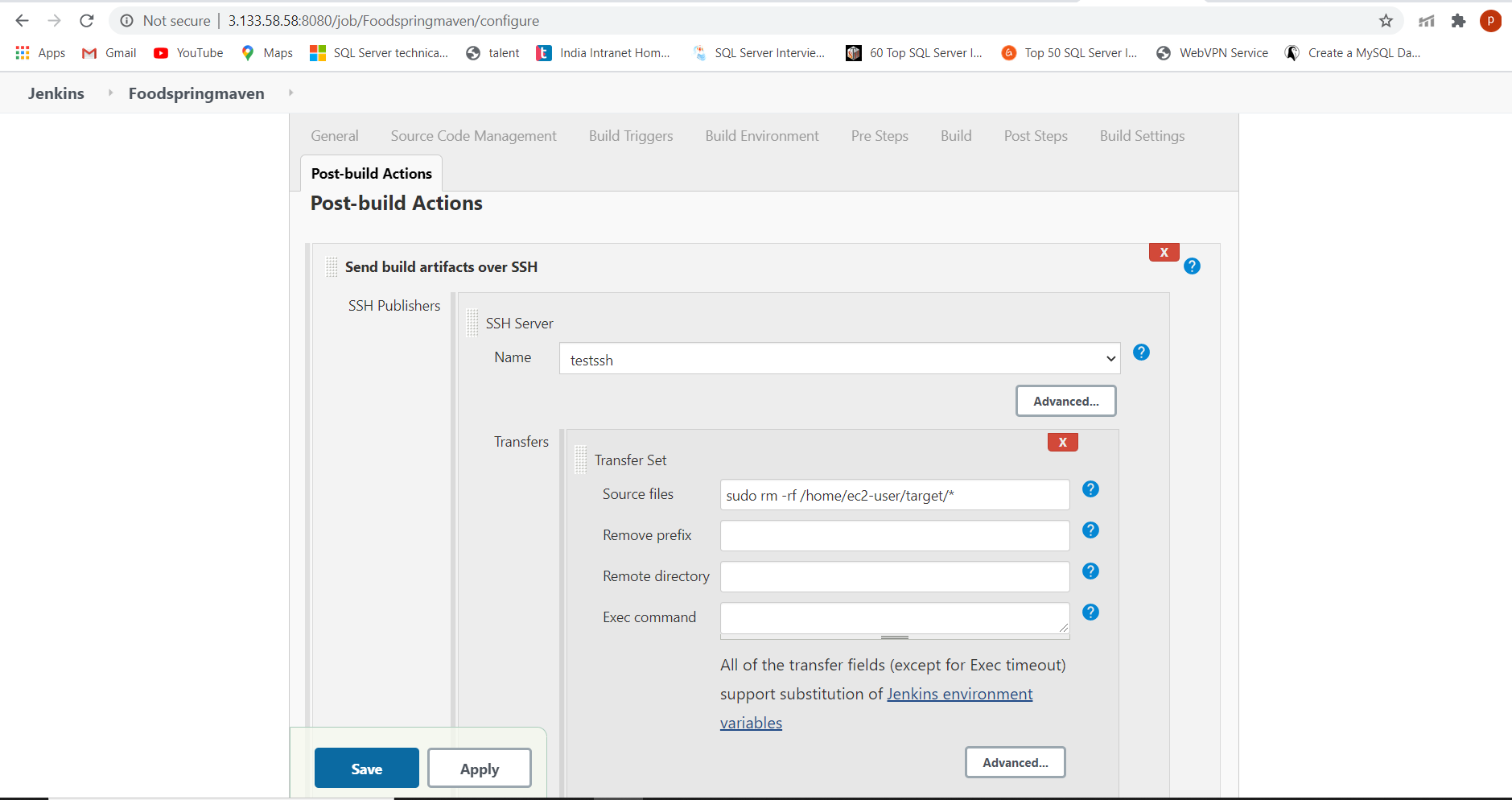
**Step 3)** Go to Build -> Goals and options give command as

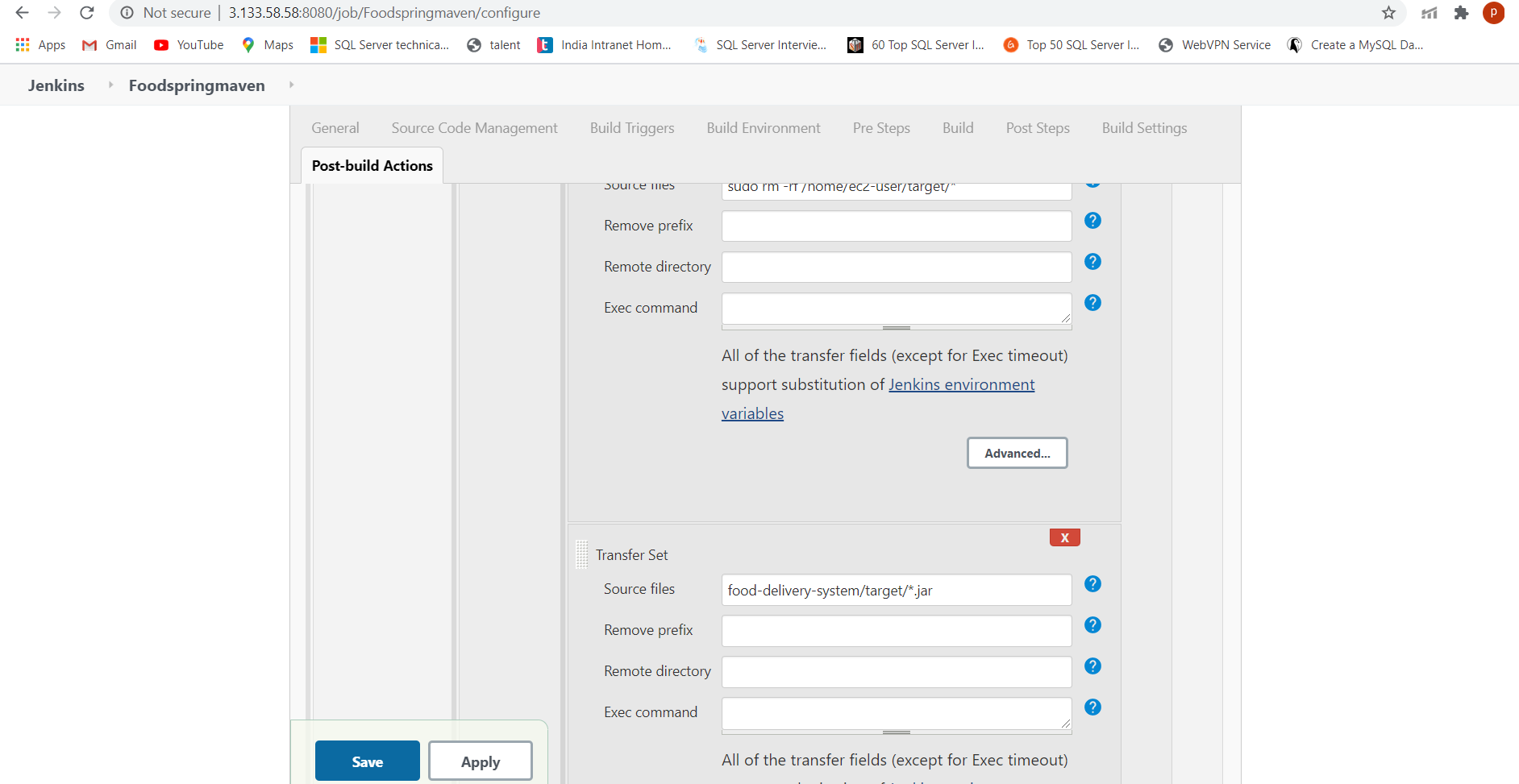
Clean package -Dmaven.test.skip=false



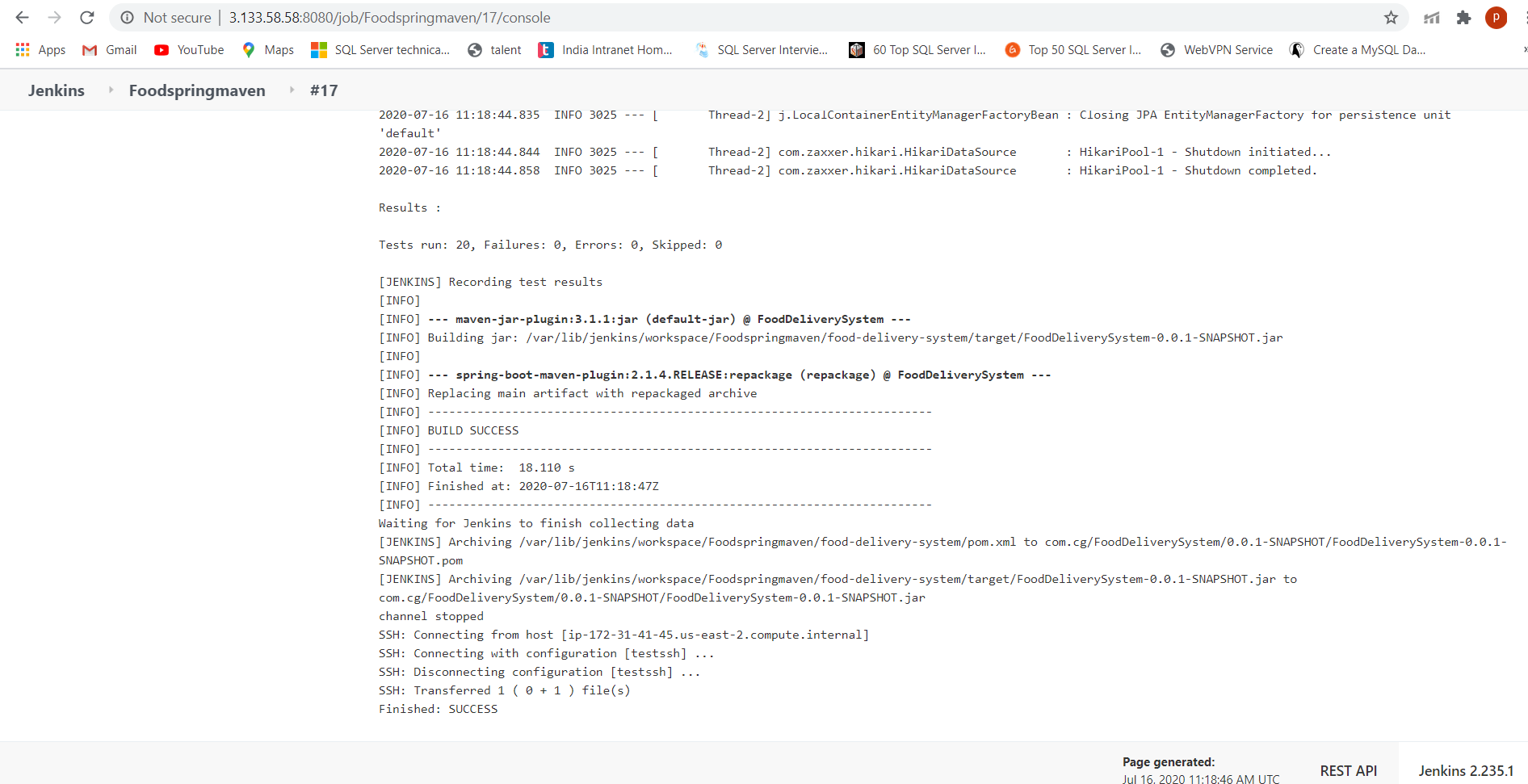
**Step 4)** Go to -> Post-build Actions, in Source files give the command as

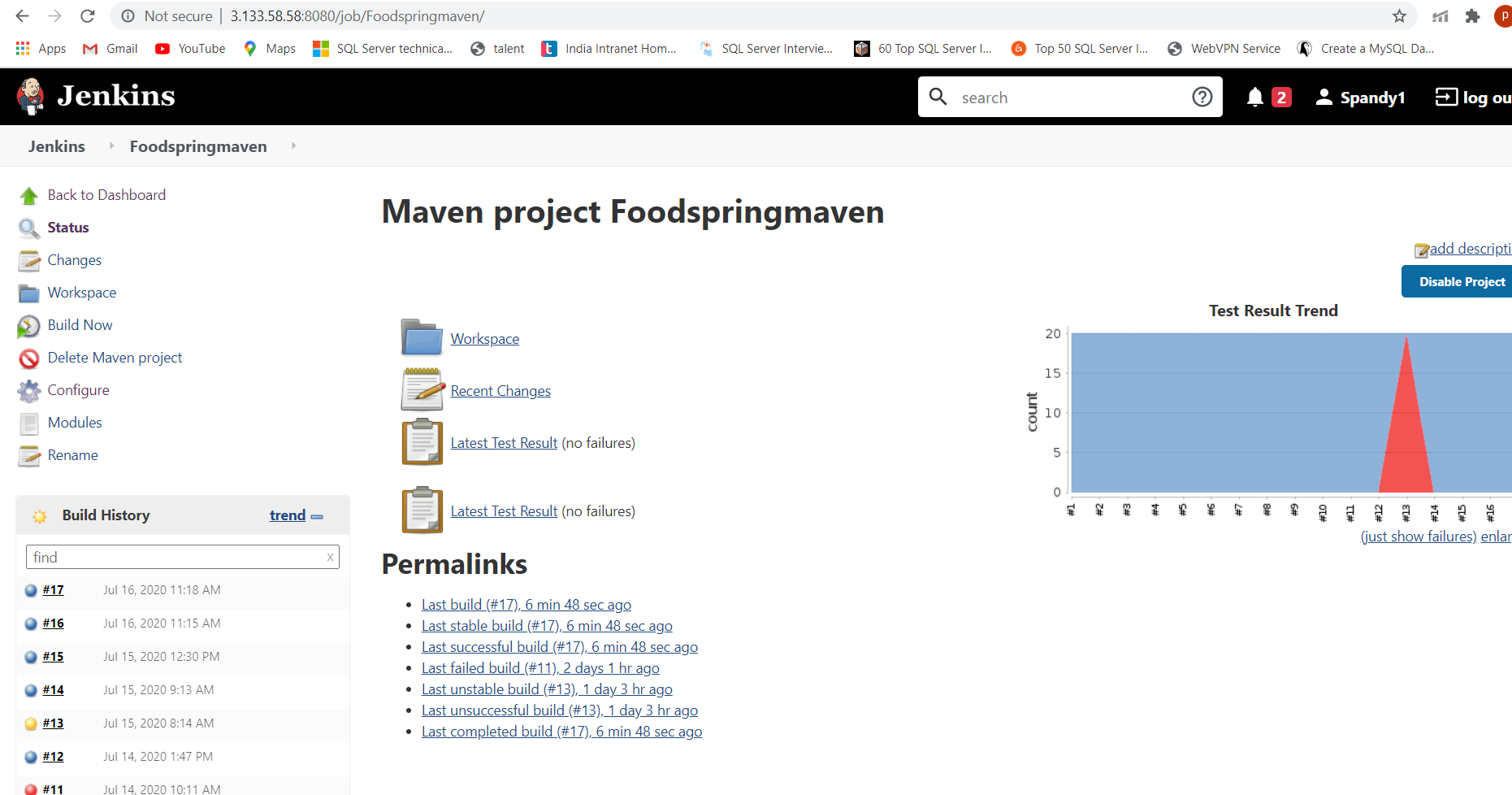
sudo rm -rf/home/ec2-user/target/\*





**Step 5)** Once build is success, you will see the result as below





You can see that file transferred msg like Transferred 1 ( 0 + 1 ) file(s)

Finished: SUCCESS

**Step 6)** Go to second instance ec2 -open terminal and go to that folder by giving the command

cd food-delivery-system/target

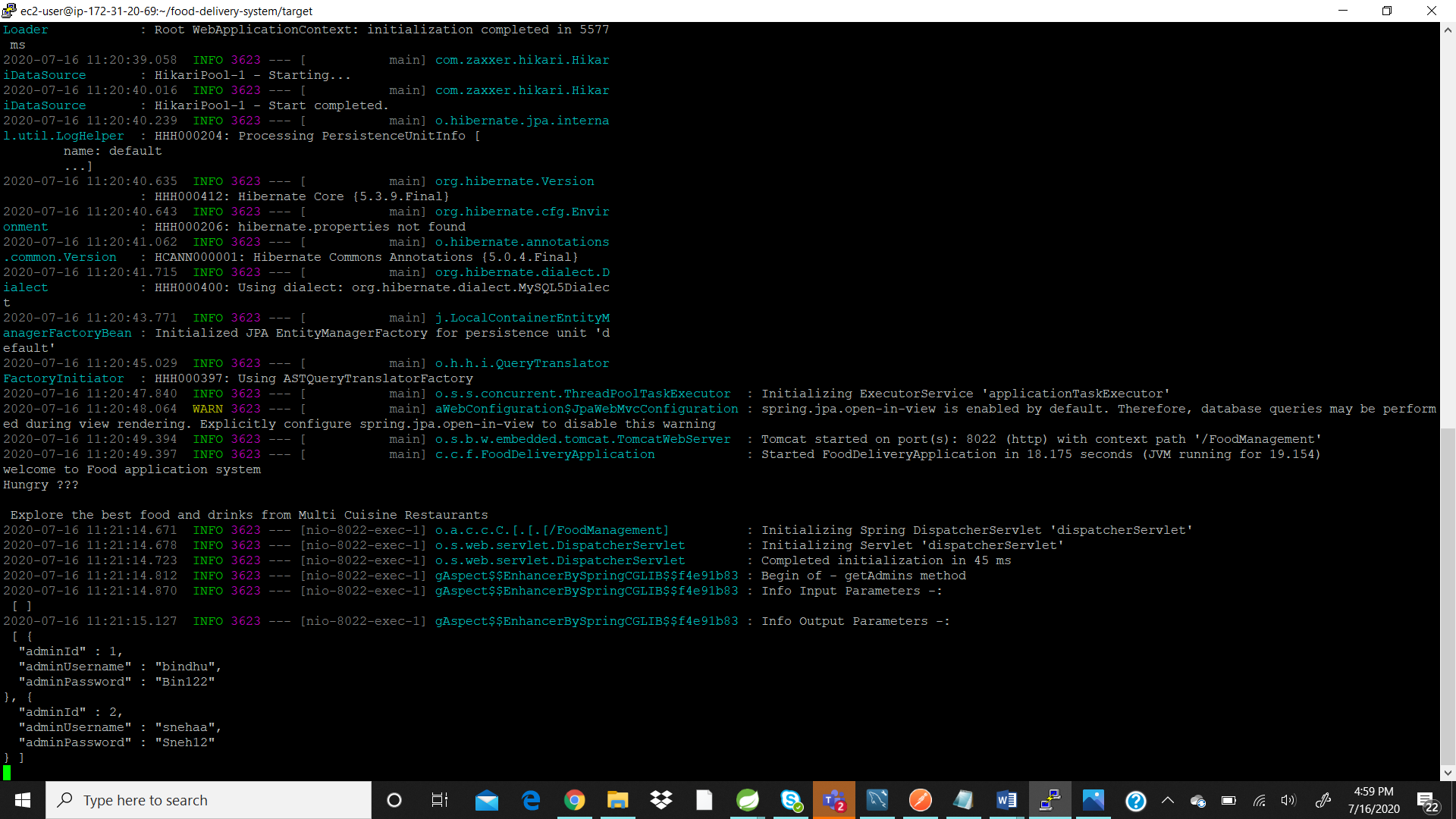
**Step 7)** Once the jar is copied to another instance, you can check using below command



**Step 8)** Run the jar using below command

Java -jar FoodDeliverySystem-0.0.1-SNAPSHOT.jar

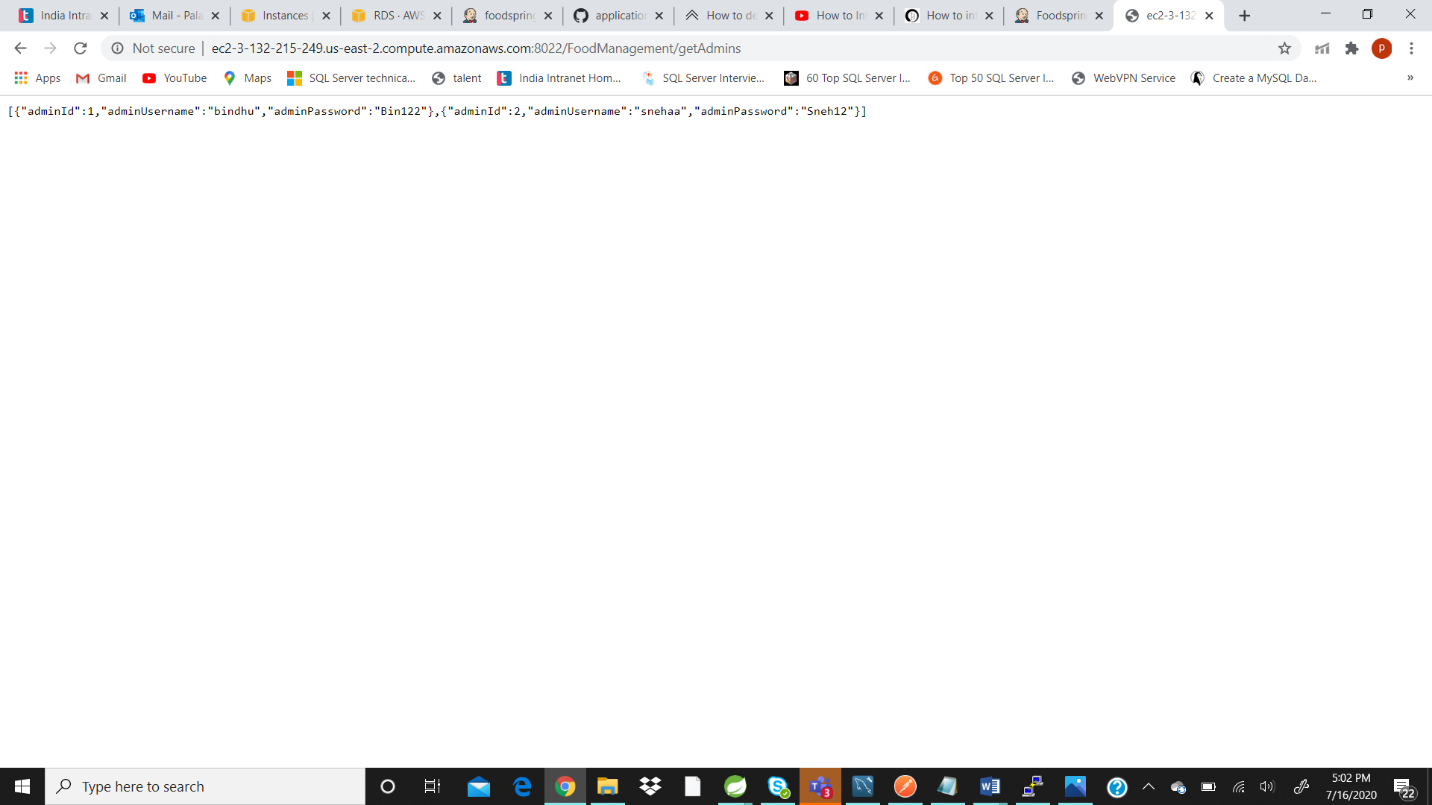


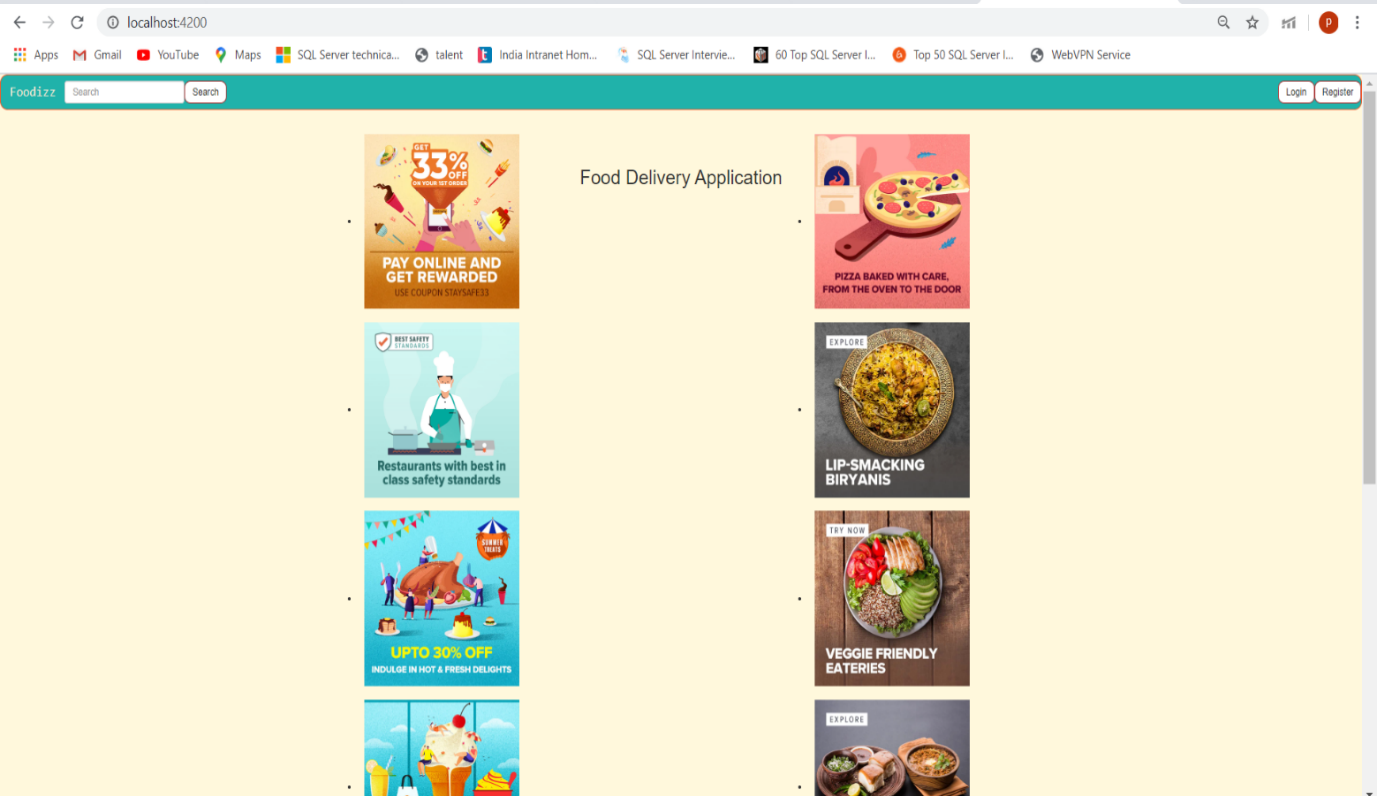


**Step 9)** Make sure that the on which port the project is running enable that port in security groups

Go to chrome 🡪 give

Public DNS (IPv4) address of file copied instance : portno





# **3. Team Members**

* Palakurthy Spandana
* Sneha Manga
* Kongara BinduSri
* Kamble Aditya
* Pooja Tiwari