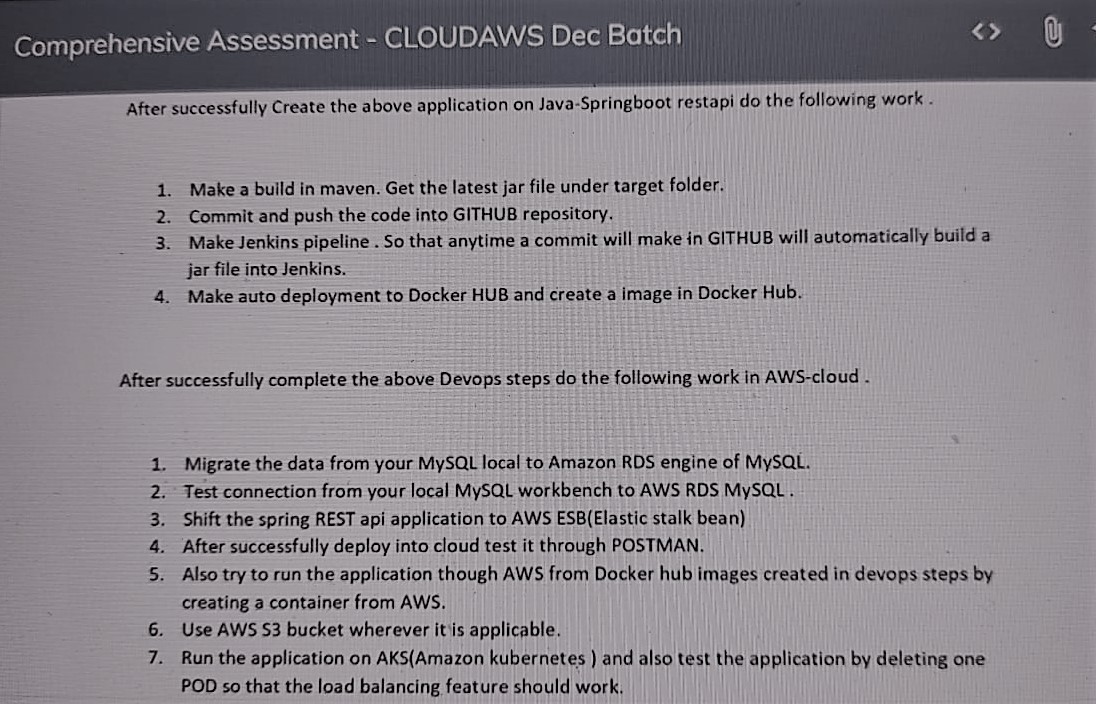
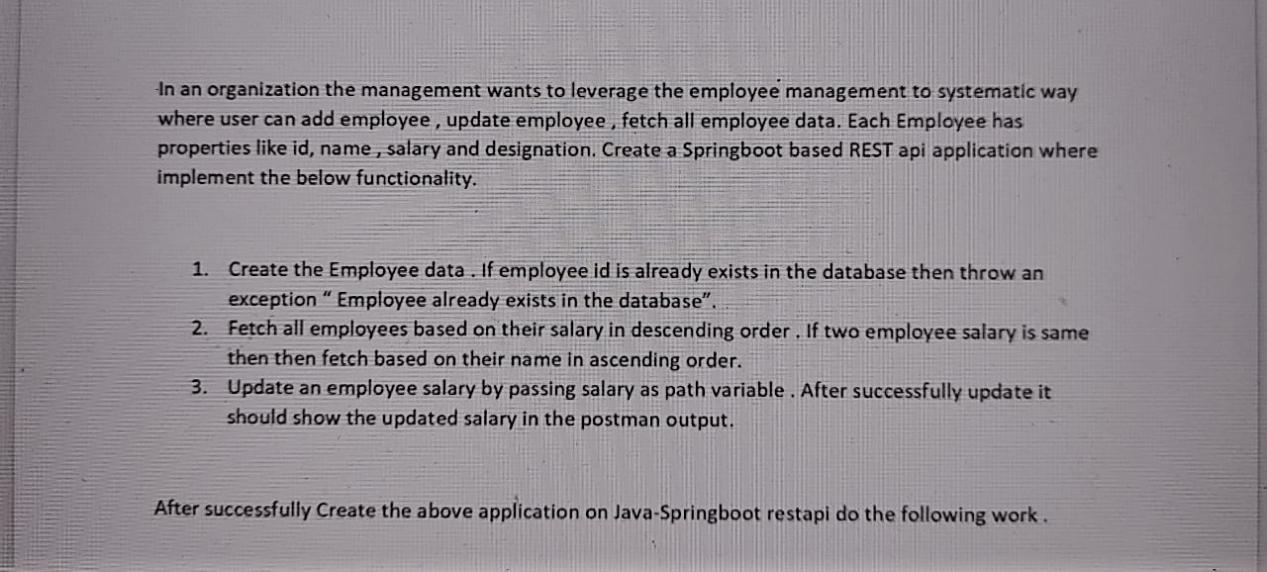
**SACHIN KAPKOTI CASE STUDY CODE ALONG WITH AWS SERVICES**

***QUESTION:***

******

******

***ANSWER :***

***EMPLOYEE.JAVA***

package com.springboot.employee.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Employee {

@Id

private int id;

private String name;

private int salary;

private String designation;

public Employee() {

}

public Employee(int id, String name, int salary, String designation) {

super();

this.id = id;

this.name = name;

this.salary = salary;

this.designation = designation;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

}

***MAINCLASS.JAVA***

package com.springboot.employee;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeProjectApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeProjectApplication.class, args);

}

}

***EMPLOYEEREPOSITORY.JAVA***

package com.springboot.employee.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.springboot.employee.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee,Integer> {

}

***EMPLOYEESERVICE.JAVA***

package com.springboot.employee.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.springboot.employee.Exception.EmployeeAlreadyExistException;

import com.springboot.employee.entity.Employee;

import com.springboot.employee.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

public List<Employee> getEmployees() {

return repository.findAll();

}

public Employee getEmployeeById(int id) {

return repository.findById(id).orElse(null);

}

public Employee updateEmployee(int id ,int salary) {

Employee existingEmployee = repository.findById(id).orElse(null);

existingEmployee.setSalary(salary);

return repository.save(existingEmployee);

}

public Employee addEmployee(Employee employee) throws EmployeeAlreadyExistException {

if(repository.existsById(employee.getId())) {

throw new EmployeeAlreadyExistException("Employee Id already exists kindly recheck");

}

else {

return repository.save(employee);

}

}

}

***EMPLOYEECOMPARE.JAVA***

package com.springboot.employee.service;

import java.util.Comparator;

import com.springboot.employee.entity.Employee;

public class EmployeeCompare implements Comparator<Employee> {

public int compare(Employee e1,Employee e2) {

if(e1.getSalary()==e2.getSalary()) {

return e1.getName().compareTo(e2.getName());

}

else if(e1.getSalary()<e2.getSalary()) {

return 1;

}else {

return -1;

}

}

}

***EMPLOYEECONTROLLER.JAVA***

package com.springboot.employee.controller;

import java.util.Collections;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.springboot.employee.Exception.EmployeeAlreadyExistException;

import com.springboot.employee.entity.Employee;

import com.springboot.employee.service.EmployeeCompare;

import com.springboot.employee.service.EmployeeService;

@RestController

@RequestMapping("/employee")

public class EmployeeController {

@Autowired

private EmployeeService service;

@PostMapping("/AddEmployee")

public Employee addEmployee(@RequestBody Employee employee) throws EmployeeAlreadyExistException{

return service.addEmployee(employee);

}

@GetMapping("/employees")

public List<Employee> getEmployeesBySalaryDesc(){

List<Employee> emplist=service.getEmployees();

Collections.sort(emplist, new EmployeeCompare());

return emplist;

}

@PutMapping("/update/{id}/{salary}")

public Employee updateEmployeeById(@RequestBody Employee e, @PathVariable Integer id,@PathVariable Integer salary) {

return service.updateEmployee(id,salary);

}

}

***EMPLOYEEALREADYEXISTEXCEPTION.JAVA***

**package** com.springboot.employee.Exception;

**public** **class** EmployeeAlreadyExistException **extends** Exception {

**public** EmployeeAlreadyExistException() {

}

**public** EmployeeAlreadyExistException(String message) {

**super**(message);

}

}

***POM.XML***

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.6.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>Employee-project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Employee-project</name>

<description>project for Spring Boot Employee</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

***APPLICATION.PROPERTIES***

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url= jdbc:mysql://sachinawscloud.cbvgdaawkqbe.us-east-1.rds.amazonaws.com/sachinCloud

spring.datasource.username = root

spring.datasource.password = sachinkapkoti

spring.jpa.show-sql = true

spring.jpa.hibernate.ddl-auto = create

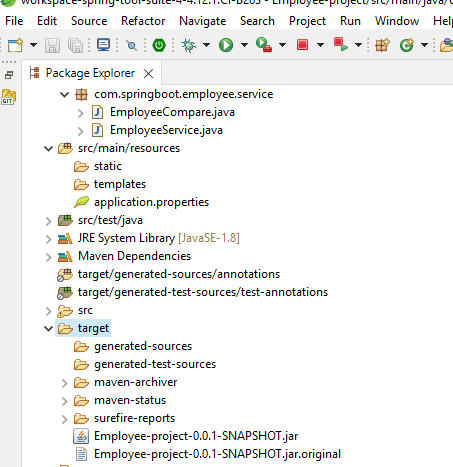
spring.jpa.database-platform = org.hibernate.dialect.MySQL5Dialect

spring.jpa.properties.max\_allowed\_packet=2000

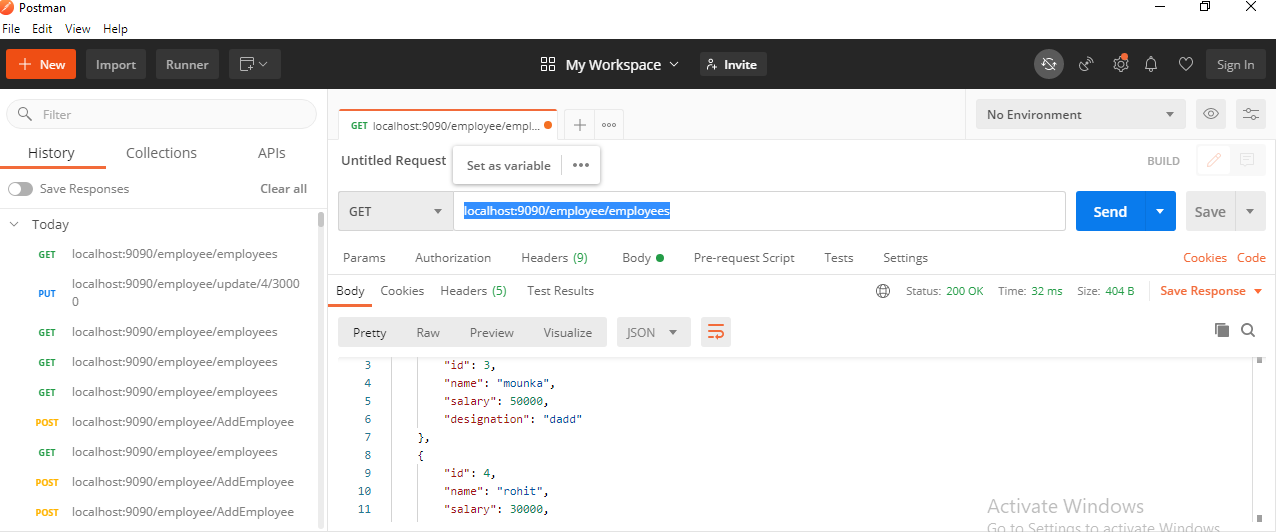
server.port=9090

***CREATING A BUILD IN MAVEN***

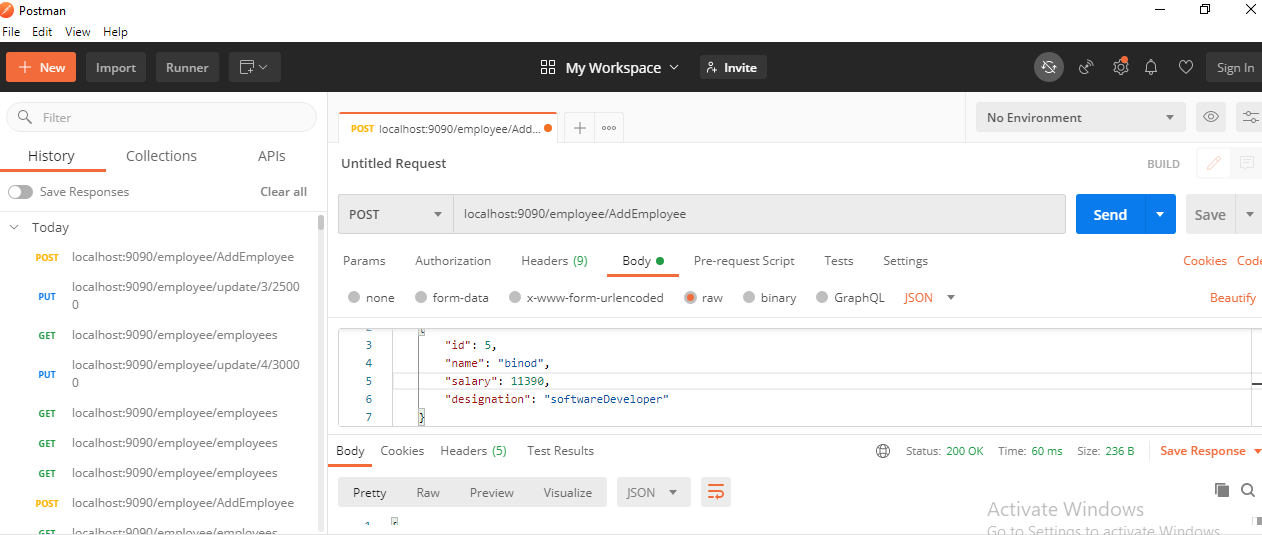
******

******

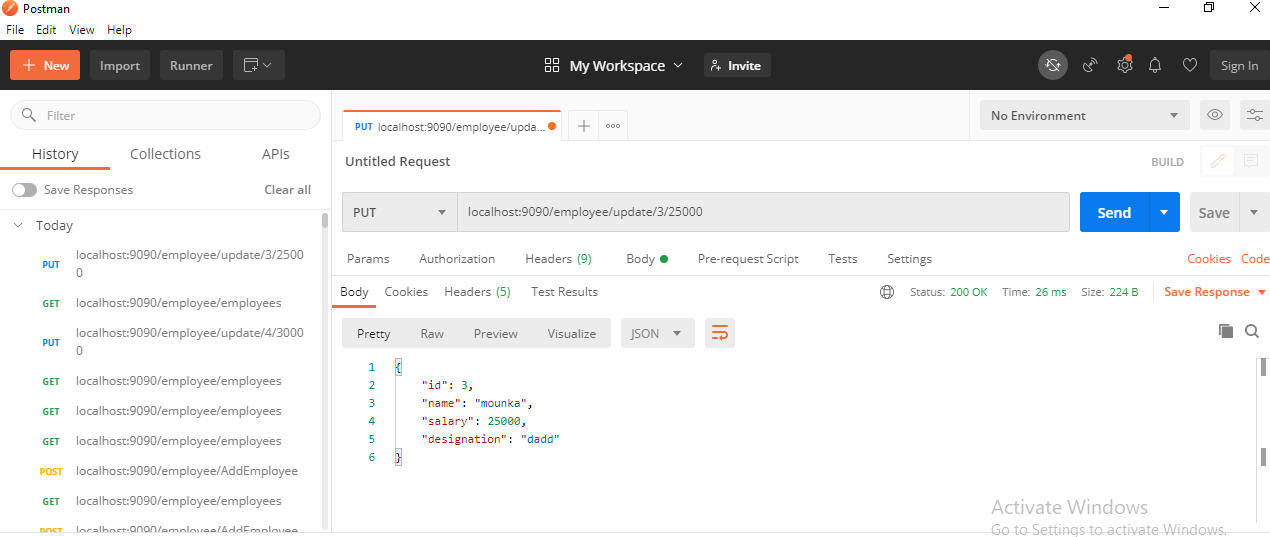
***OUTPUT OF CODE IN POSTMAN***

******

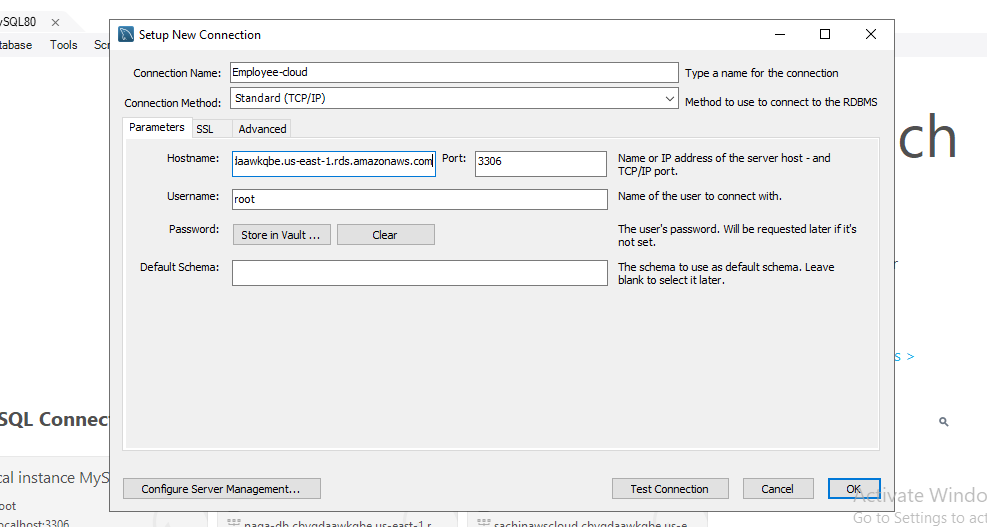
***POSTMAN POST MAPPING***

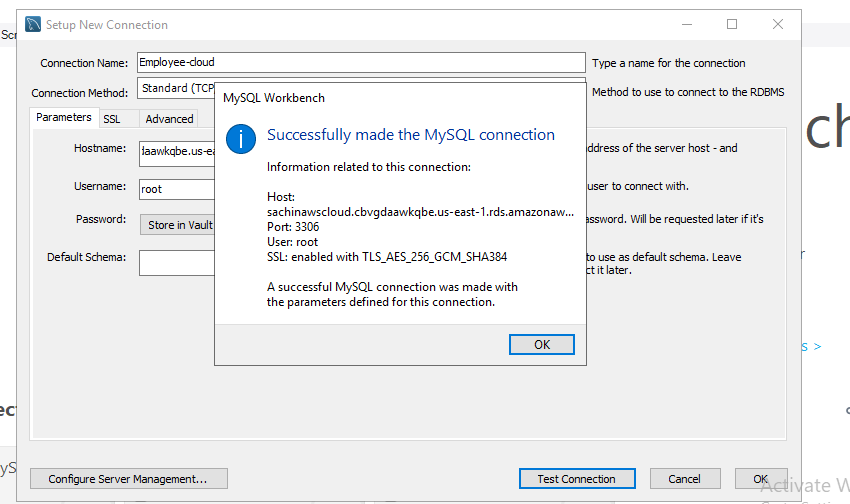
******

***POSTMAN PUT MAPPING***

******

***TESTING RDS CONNECTION IN MYSQL***

******

******

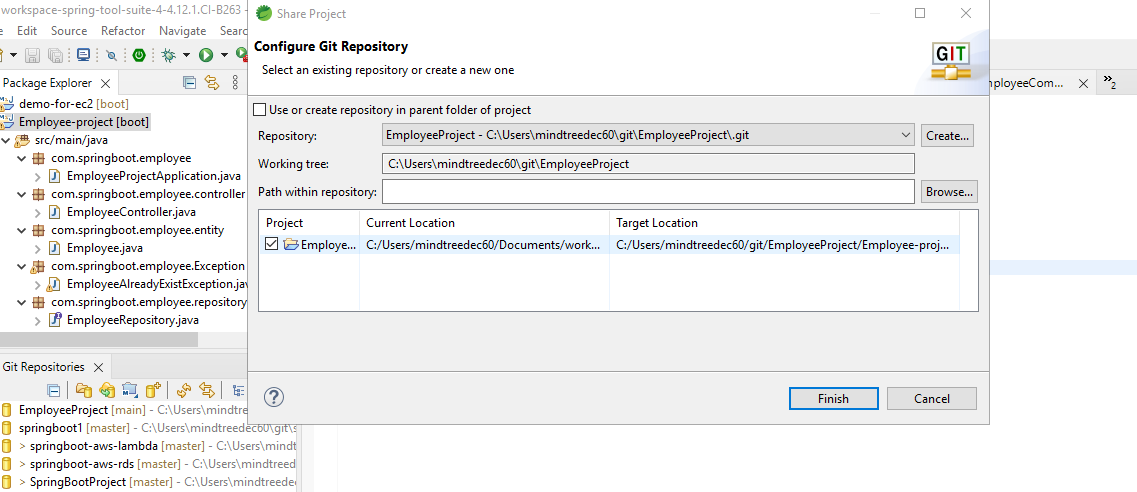
***PUSHING CODE TO GITHUB AND JENKINS***

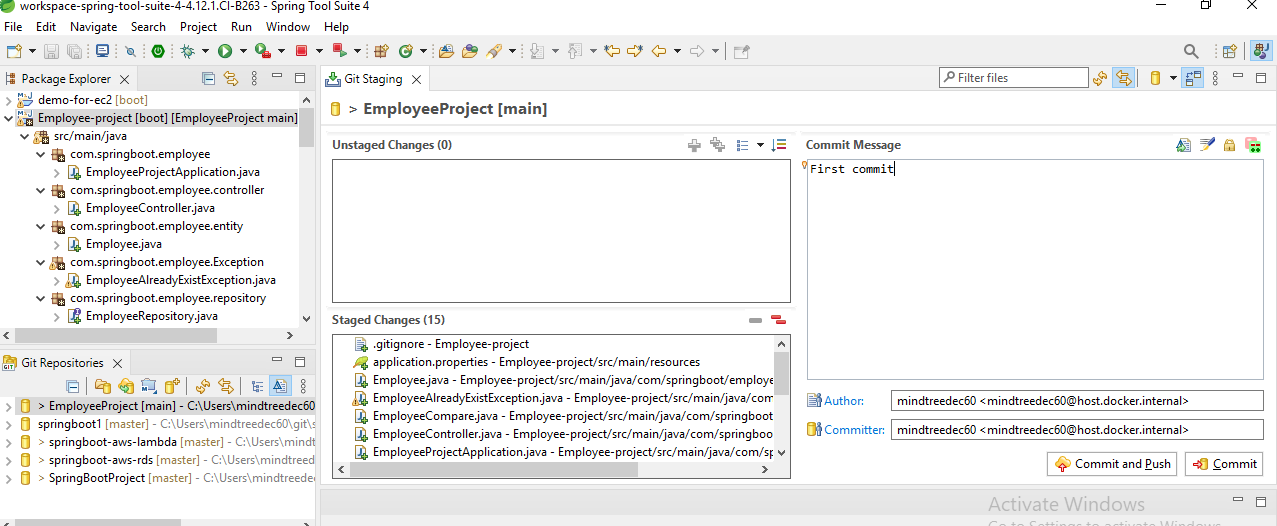
***INTEGRATION TO CREATE PIPELINE***

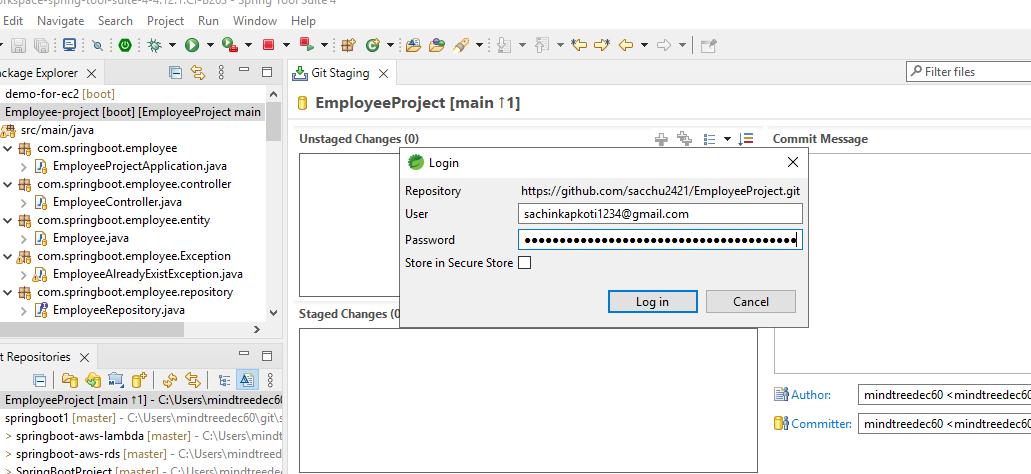
***Step1 - Make a new repository in your remote git repo***

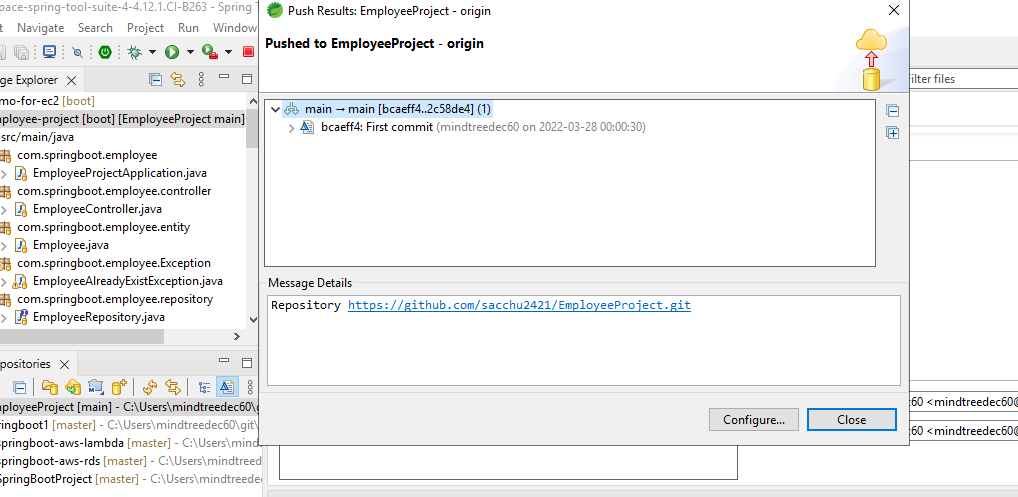
***Step 2 - Copy the repo url and clone in local repository***

***Step 3- Right click on the respective project you want to push and share it and thenonwards follow the below steps***

******

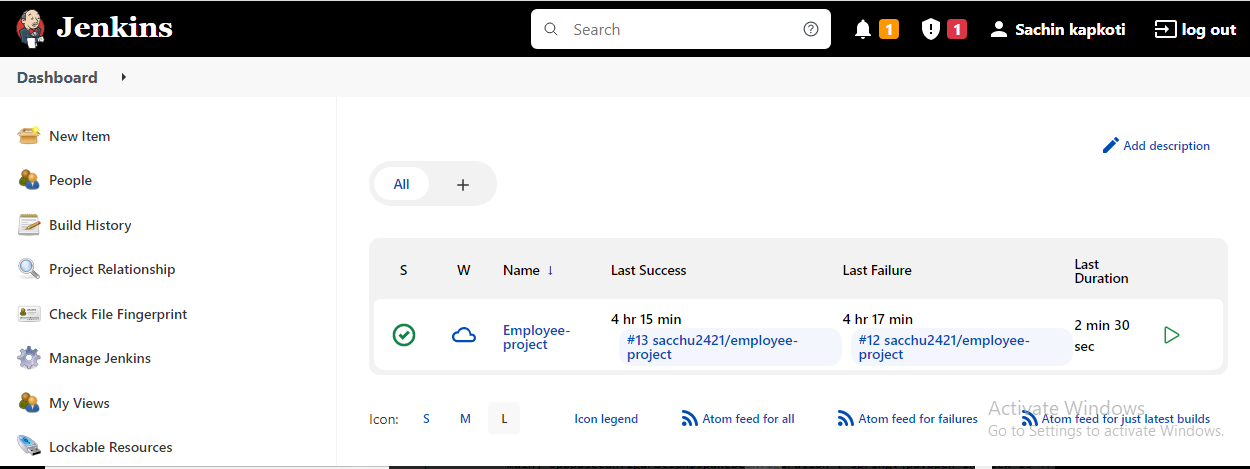
******

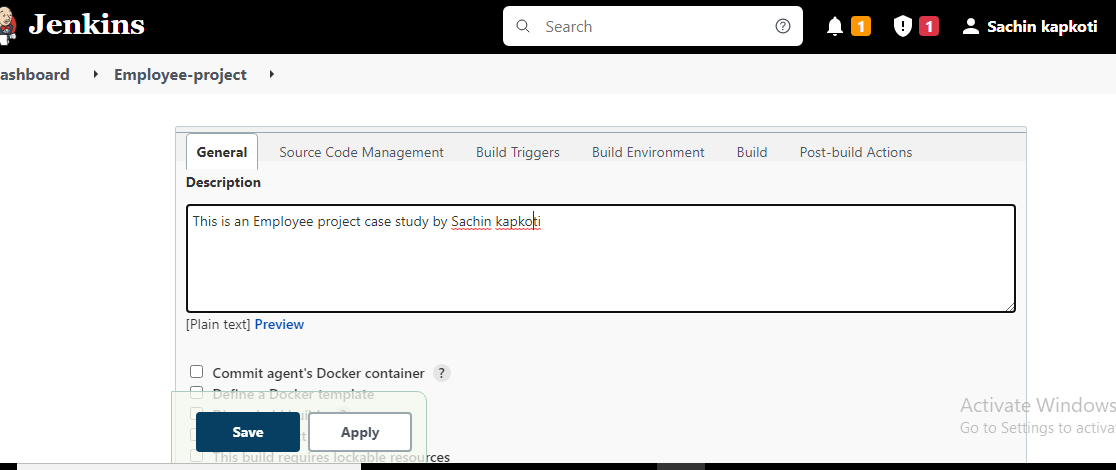
******

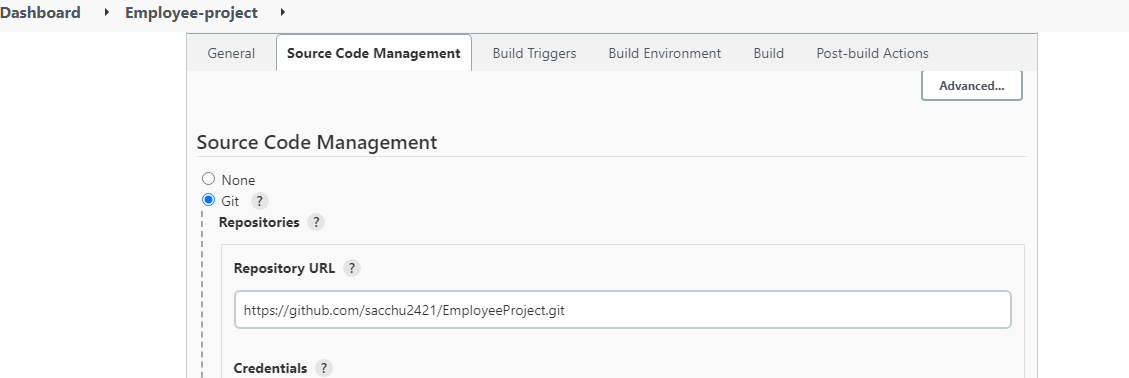
******

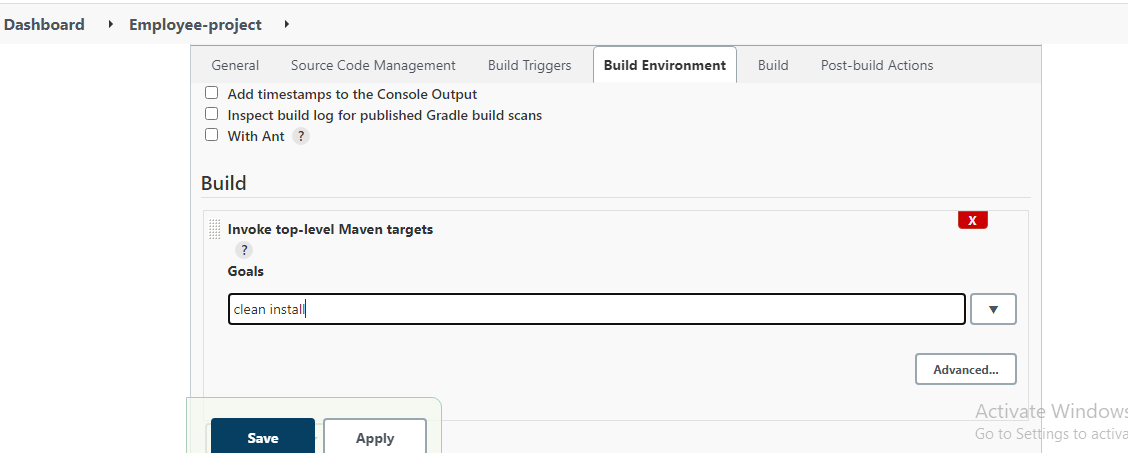
***CREATING JENKINS PIPELINE ALONG***

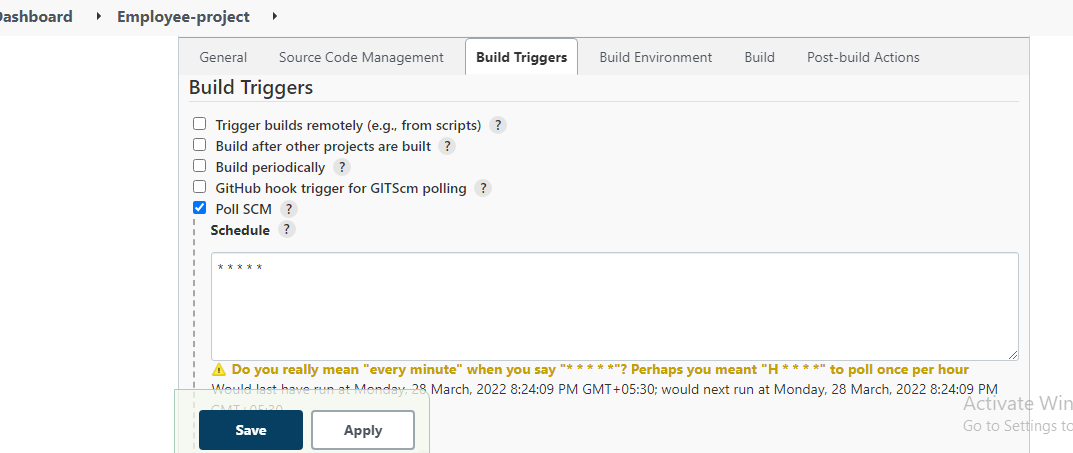
***PUSHING DOCKER IMAGE TO DOCKER HUB***

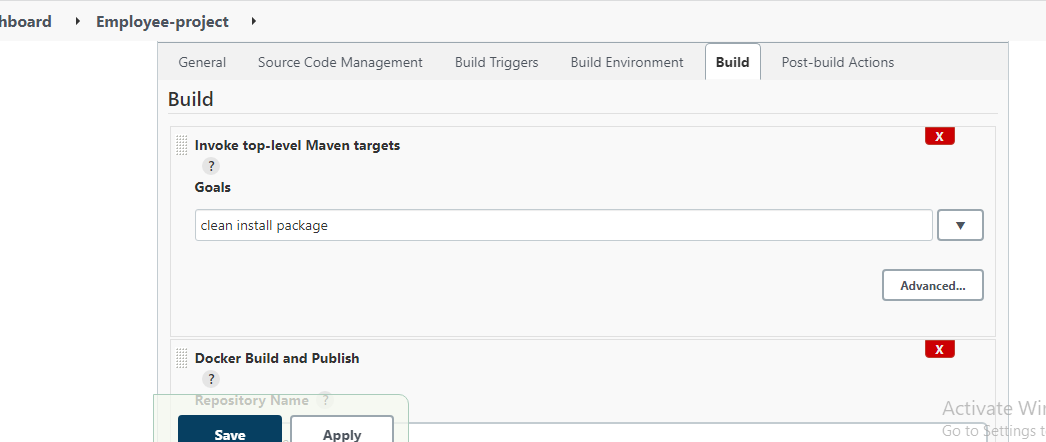
******

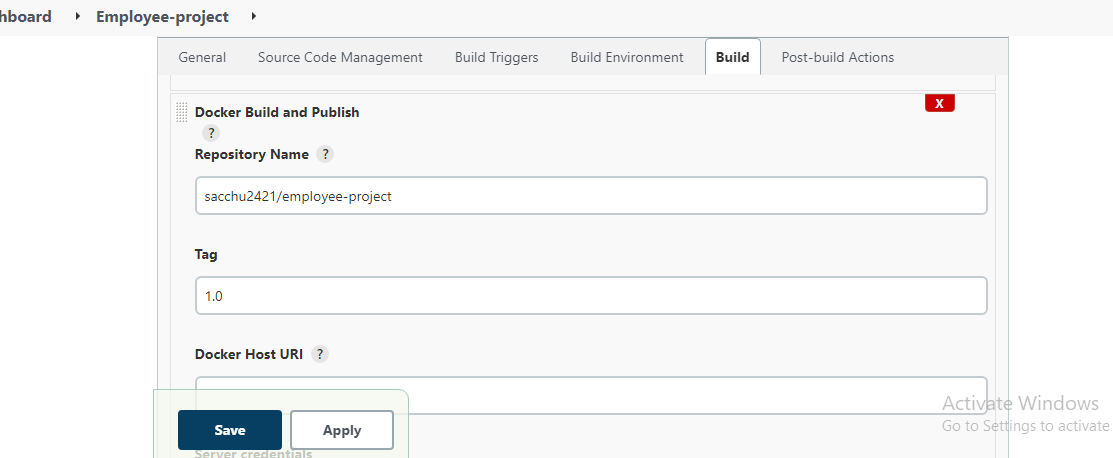
******

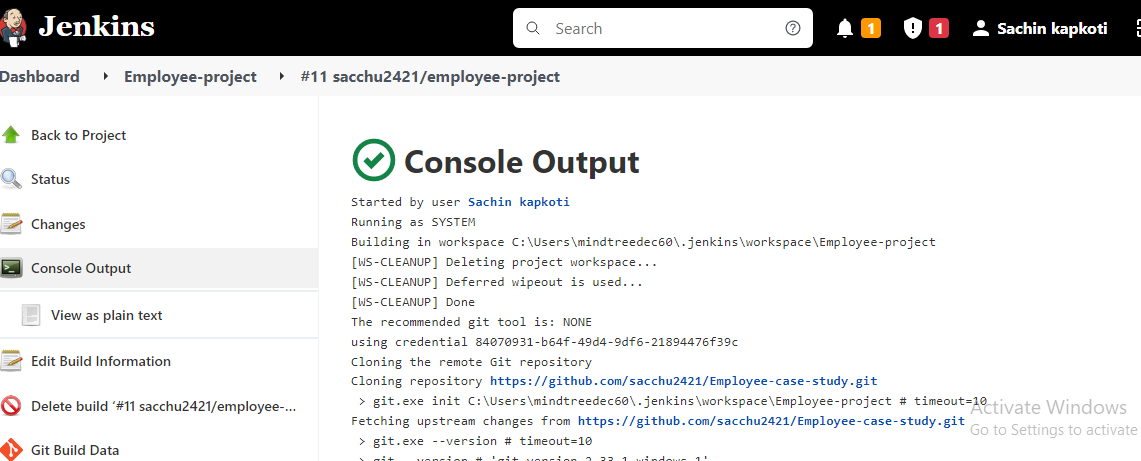
******

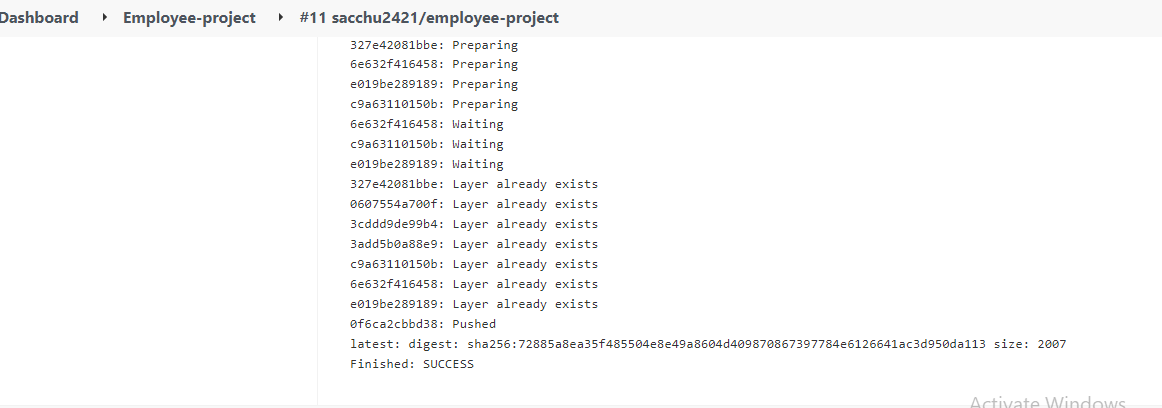
******

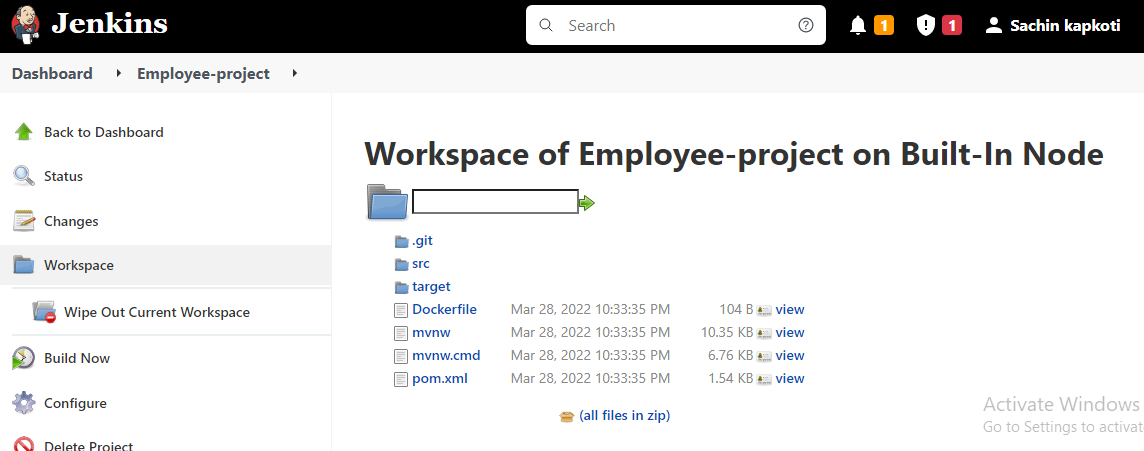
******

******

******

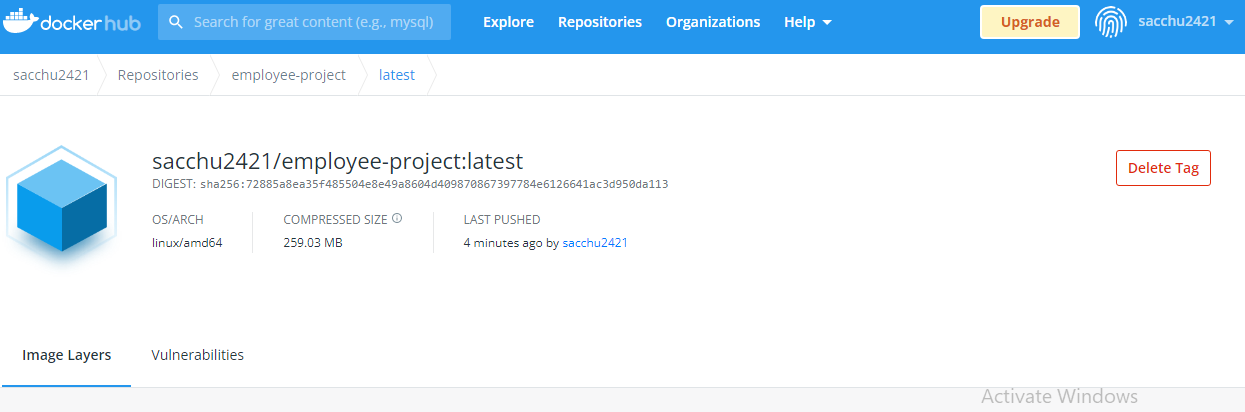
******

******

******

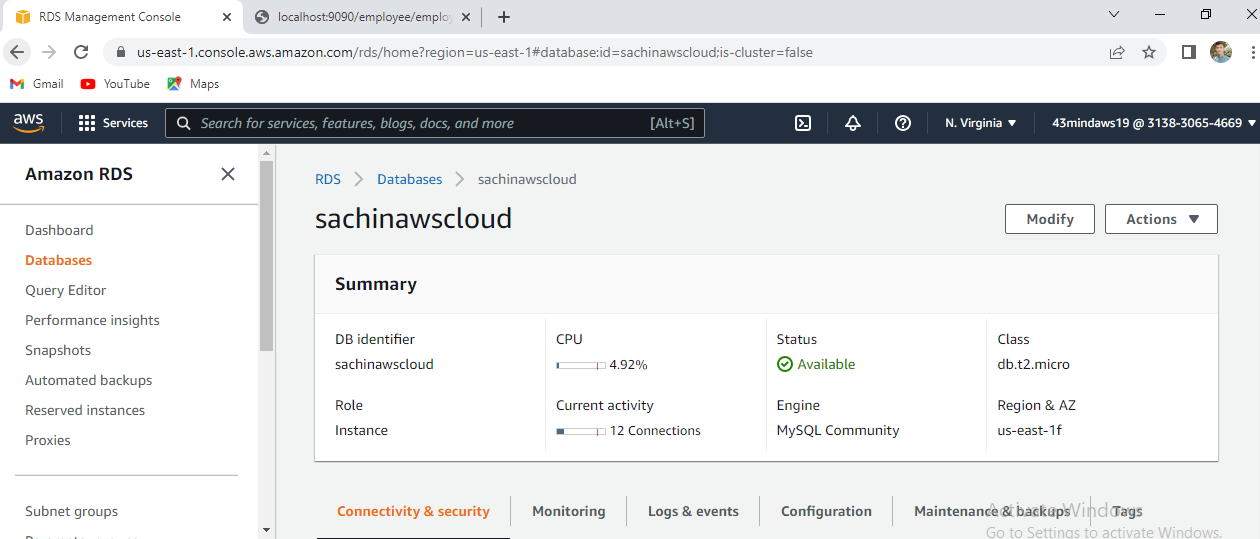
***NOW CHECKING THE THE DOCKERHUB***

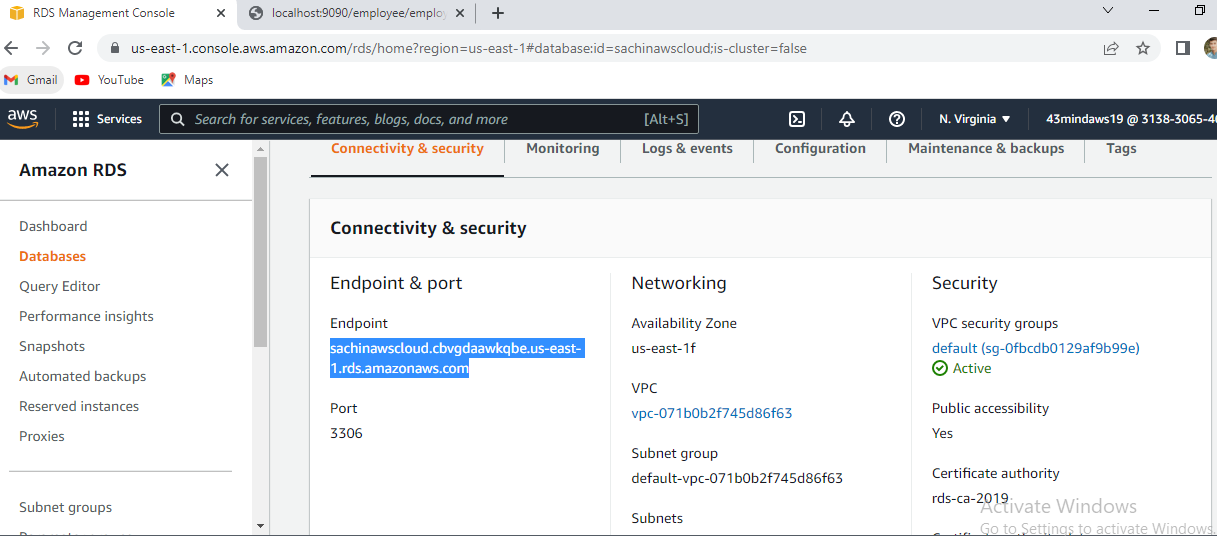
***AFTER THE BUILD IS DONE***

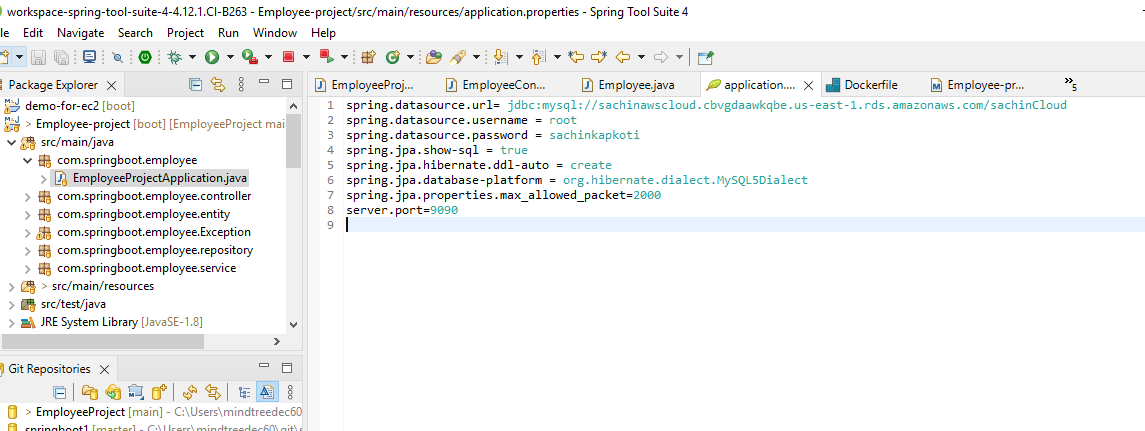
******

***PUSHING THE SPRINGBOOT PROJECT TO ELASTIC***

***BEAN STALK***

******

******

******

***CREATING ELASTIC BEAN STALK ENVIRONMENT***

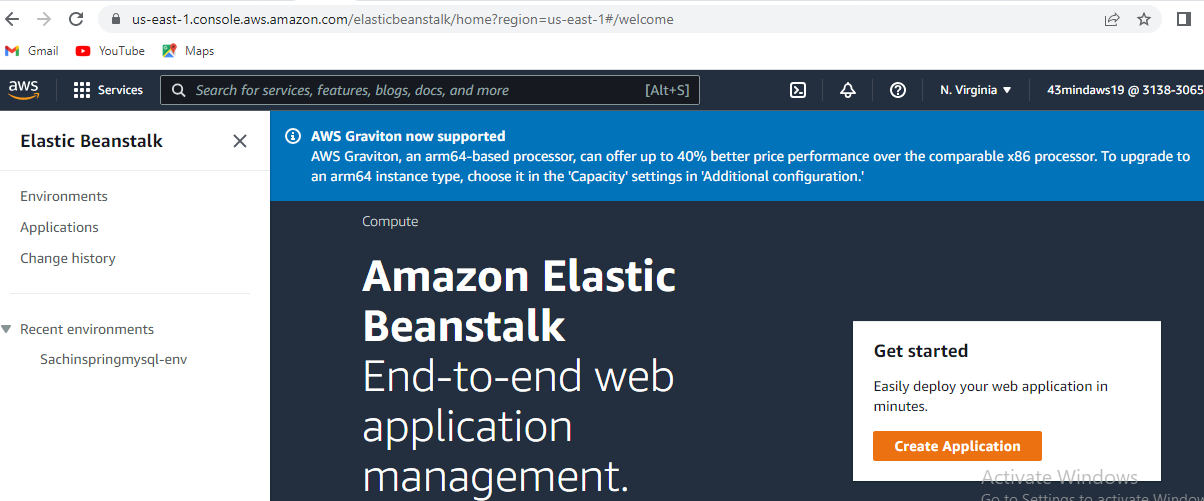
**What is Elastic Beanstalk used for?**

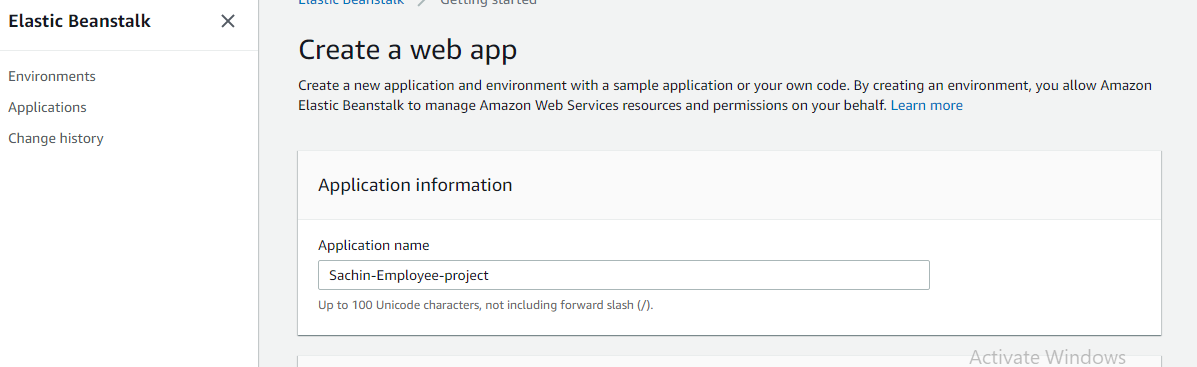
Elastic Beanstalk is a platform within AWS that is used for **deploying and scaling web applications**. In simple terms this platform as a service (PaaS) takes your application code and deploys it while provisioning the supporting architecture and compute resources required for your code to run..

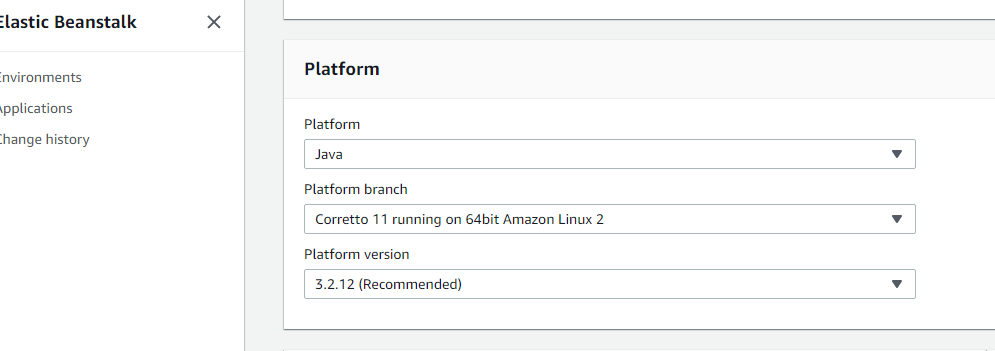
**NOTE** - Below is provided all the steps to create environment in EBS

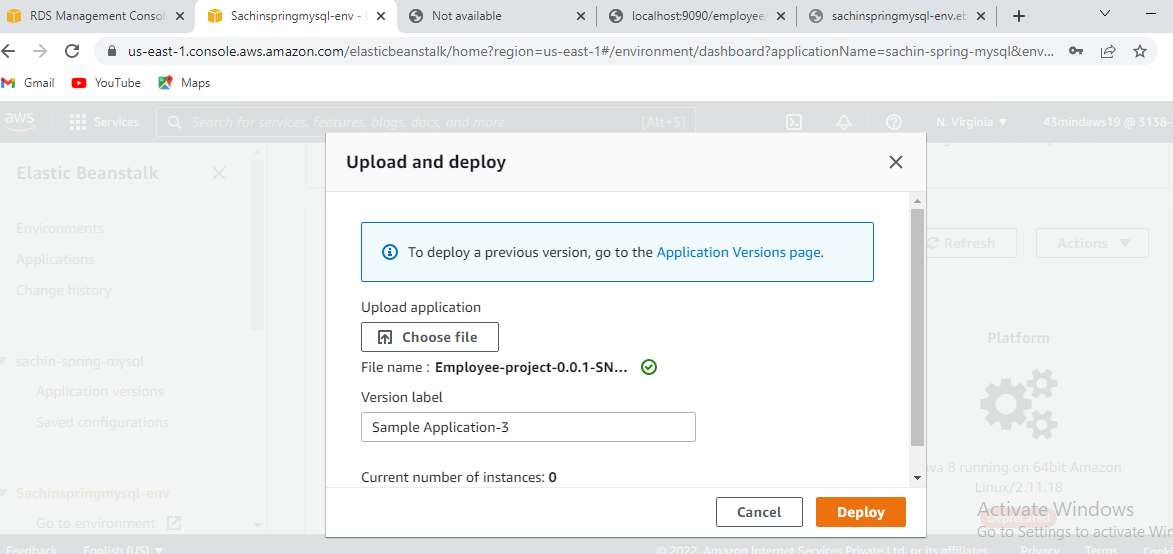
And the respective postman output

***Firstly for using aws services we need to open the aws console using aws.amazon.com***

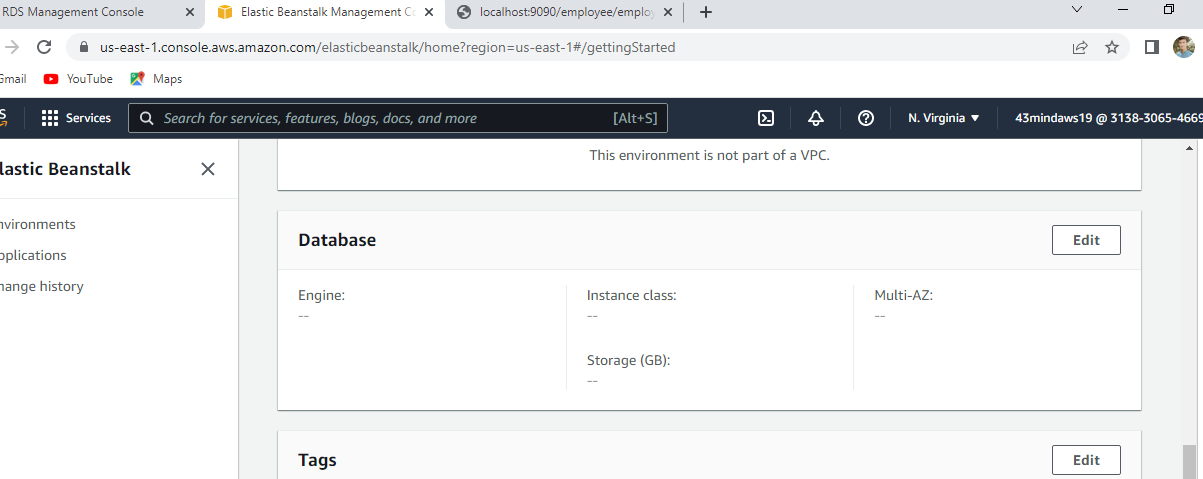
******

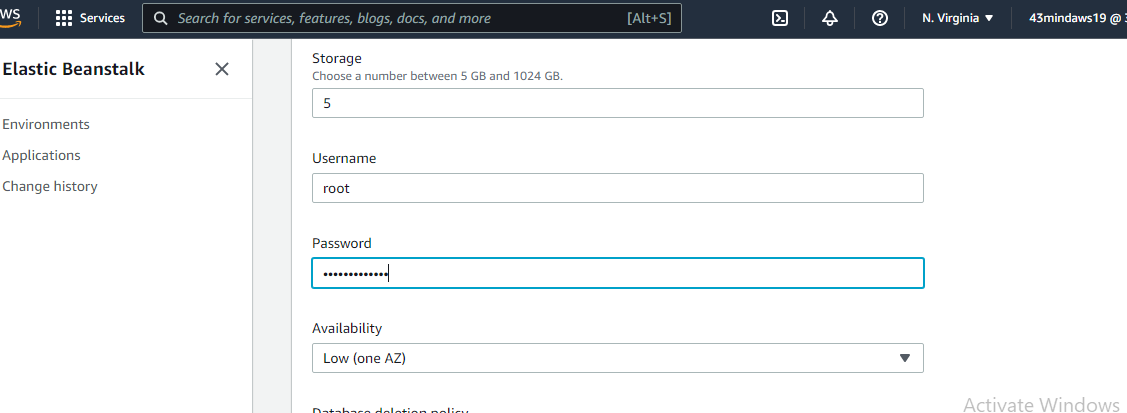
******

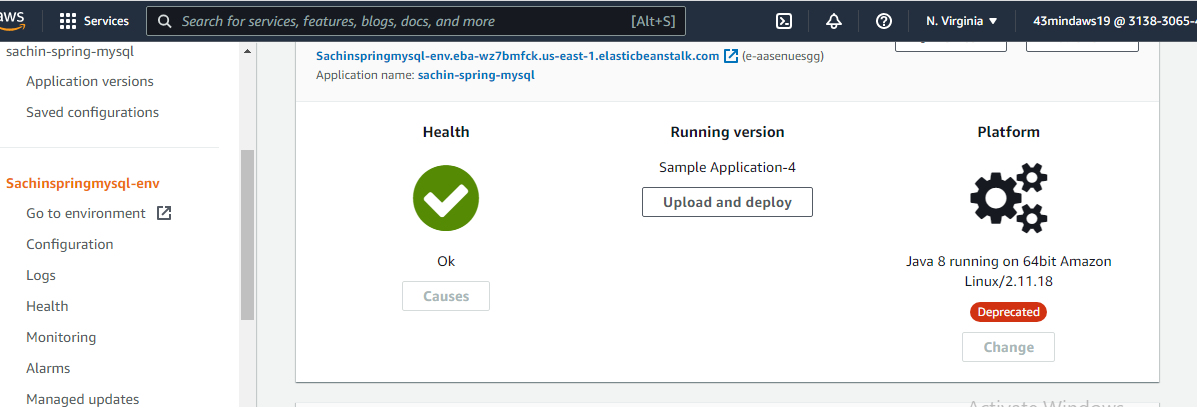
******

******

***We need to configure the RDS database with the EBS***

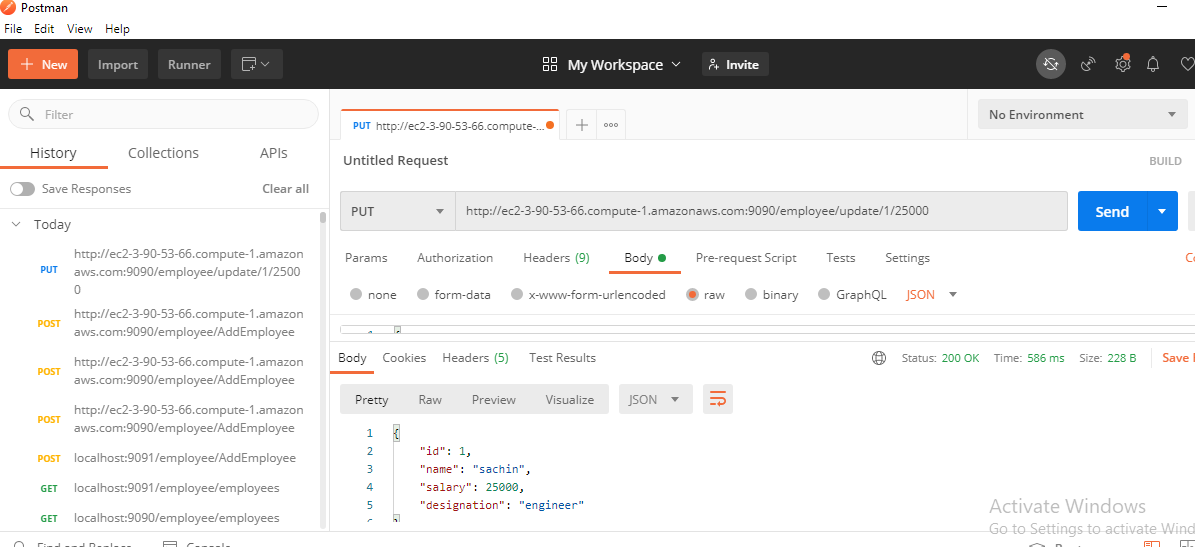
******

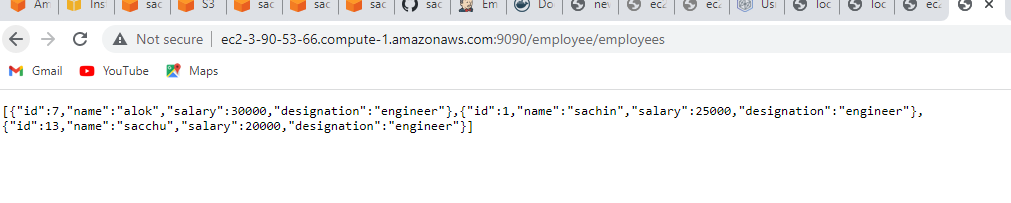
******

******

***TESTING EBS (SPRINGBOOT APP) ON POSTMAN***

***USING PUT MAPPING TO UPDATE SALARY***

******

******

***CREATING AN EC2 INSTANCE AND RUNNING DOCKER IMAGES ON AWS WITH HELP OF AWS S3***

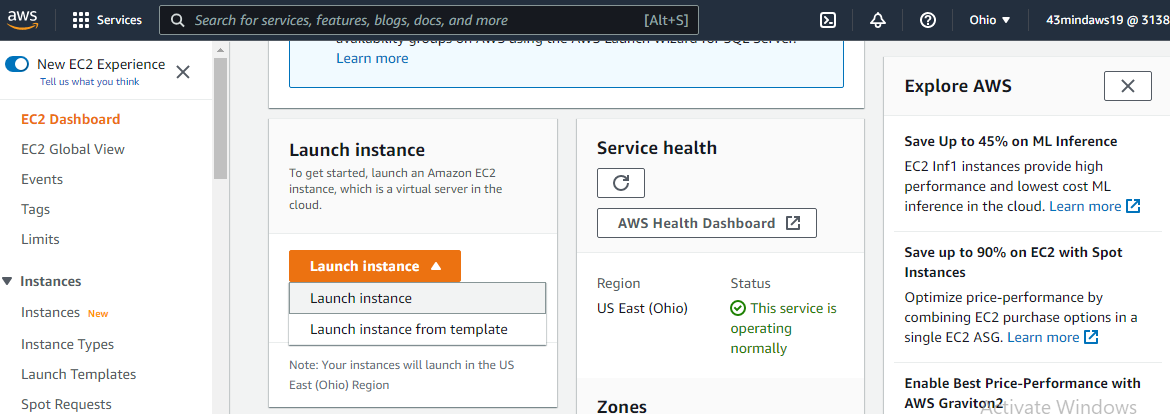
**What is EC2 ?**

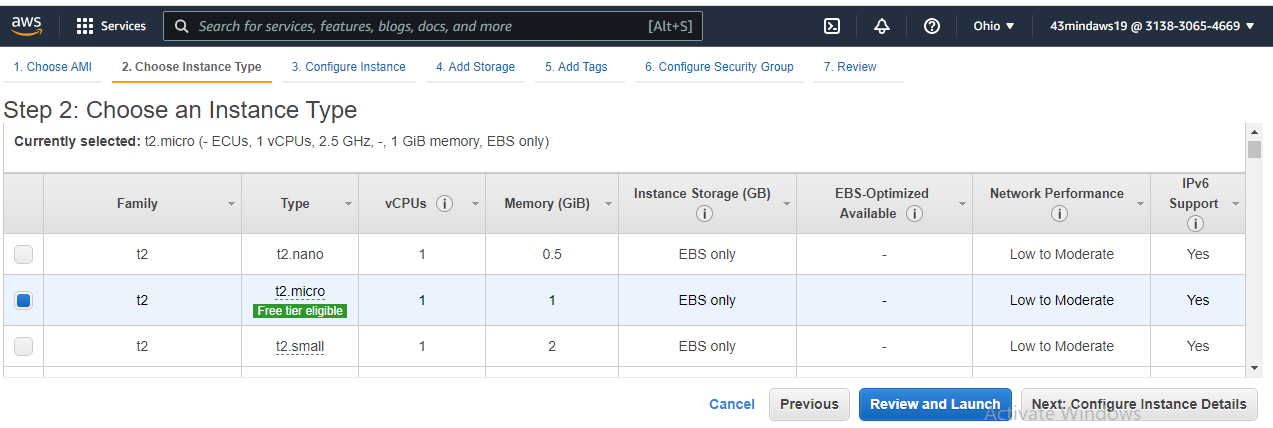
To put it simply, an EC2 is **a virtual machine that represents a physical server for you to deploy your applications**. Instead of purchasing your own hardware and connecting it to a network, Amazon gives you nearly unlimited virtual machines to run your applications while they take care of the hardware.

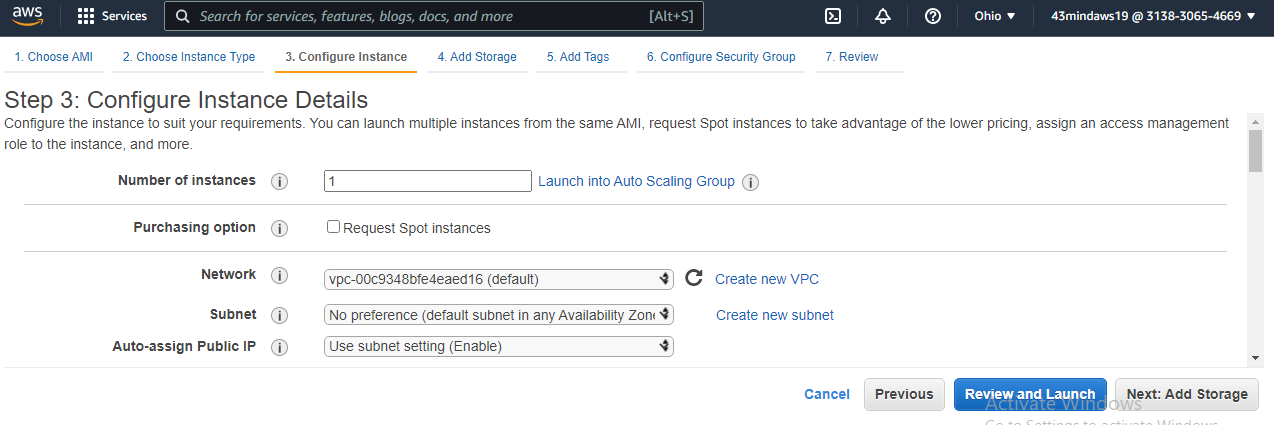
**What is S3 bucket?**

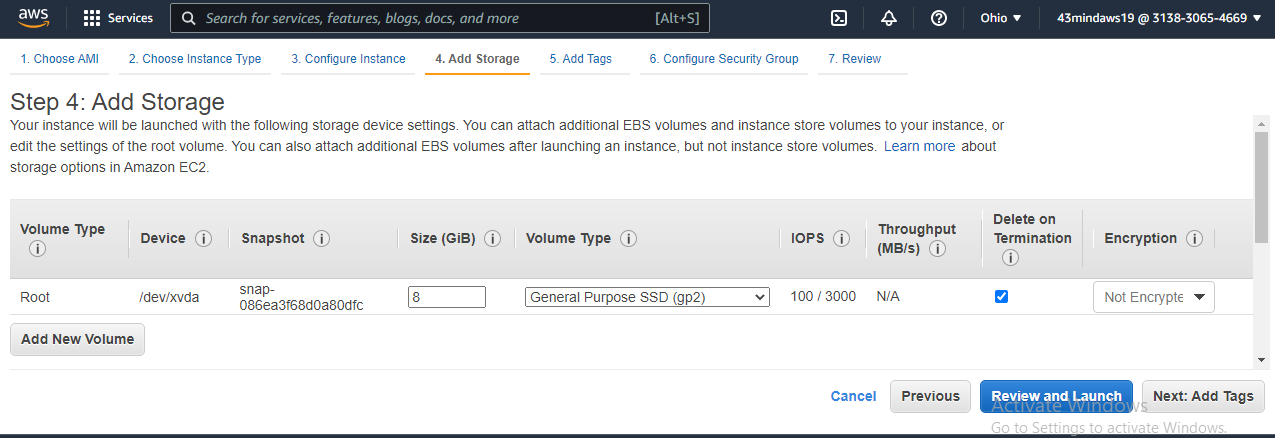
An Amazon S3 bucket is **a public cloud storage resource available in Amazon Web Services' (AWS) Simple Storage Service (S3), an object storage offering**. Amazon S3 buckets, which are similar to file folders, store objects, which consist of data and its descriptive metadata.

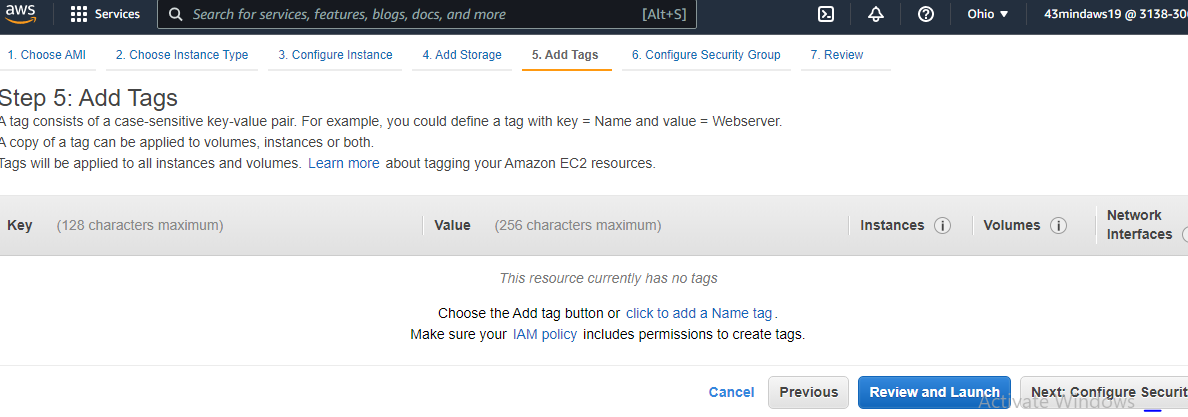
***Here below shown the steps to create ec2 instance along with s3 bucket and run the spring boot App using EC2 and S3***

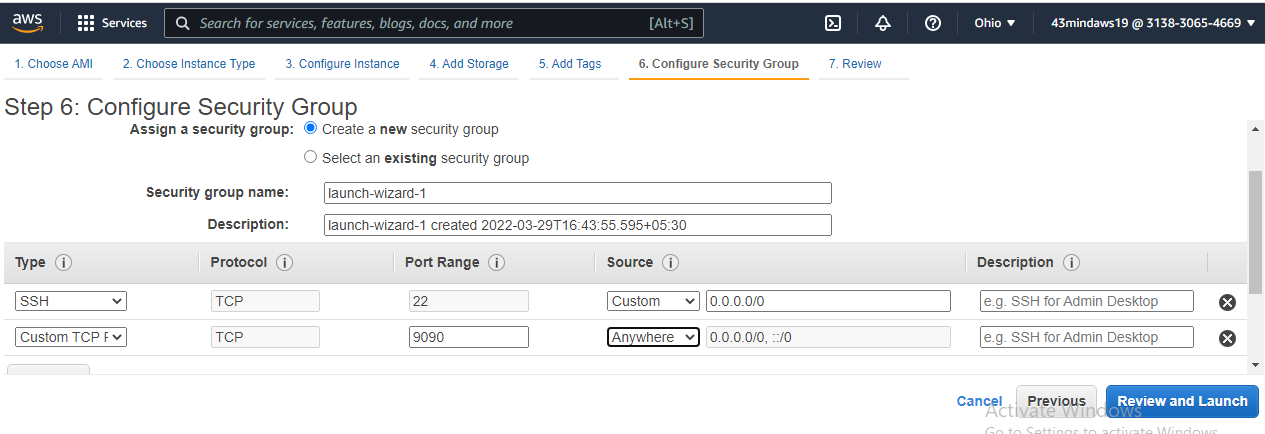
******

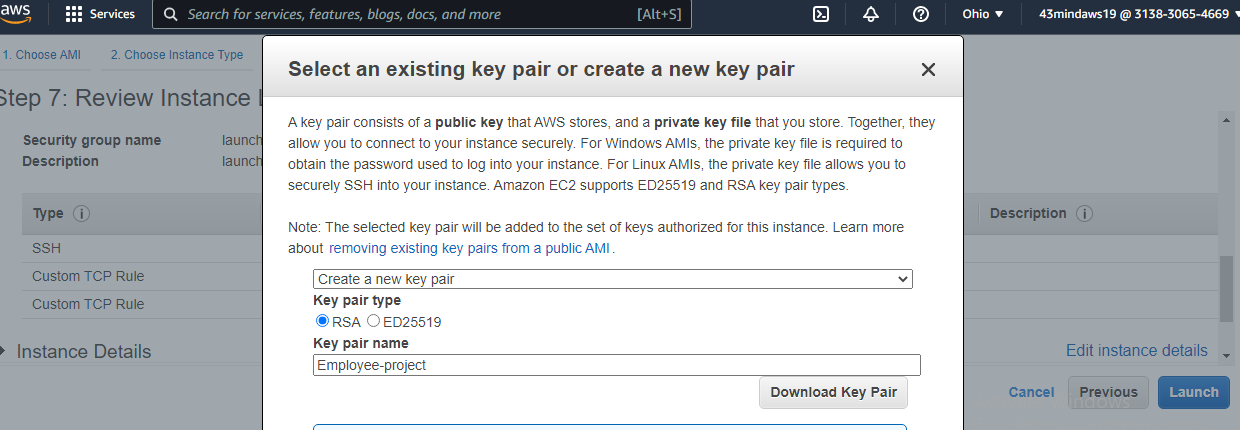
******

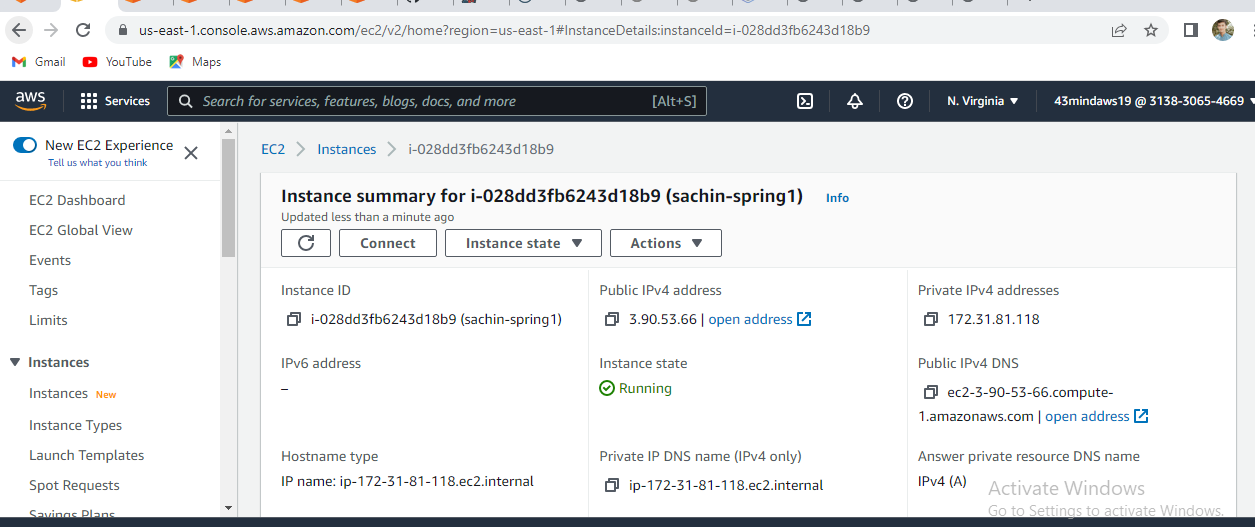
******

******

******

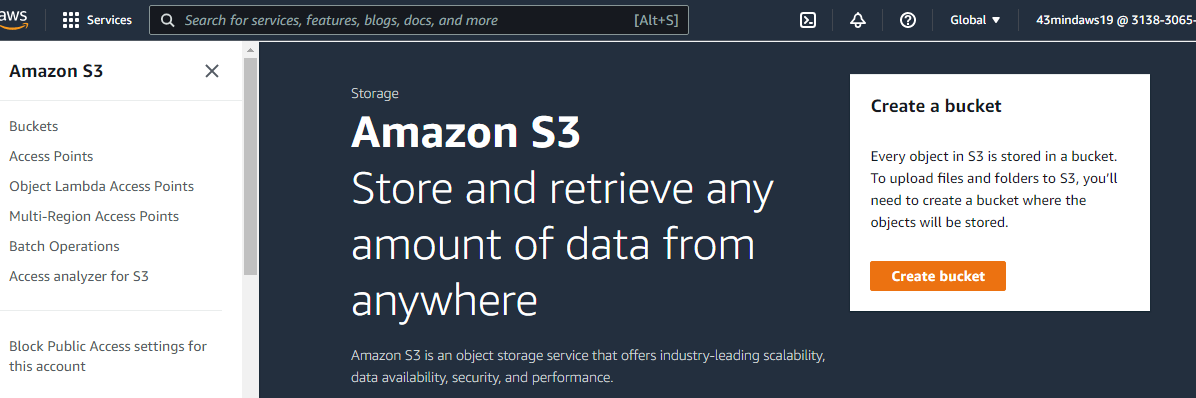
******

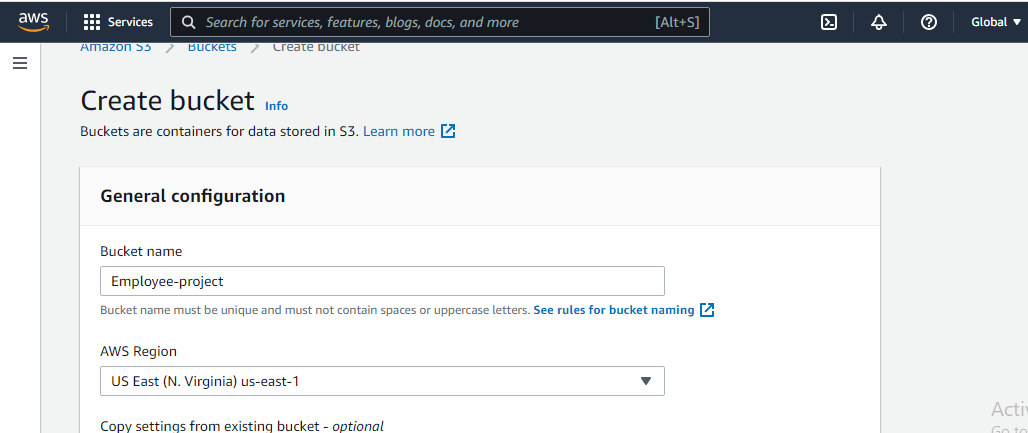
******

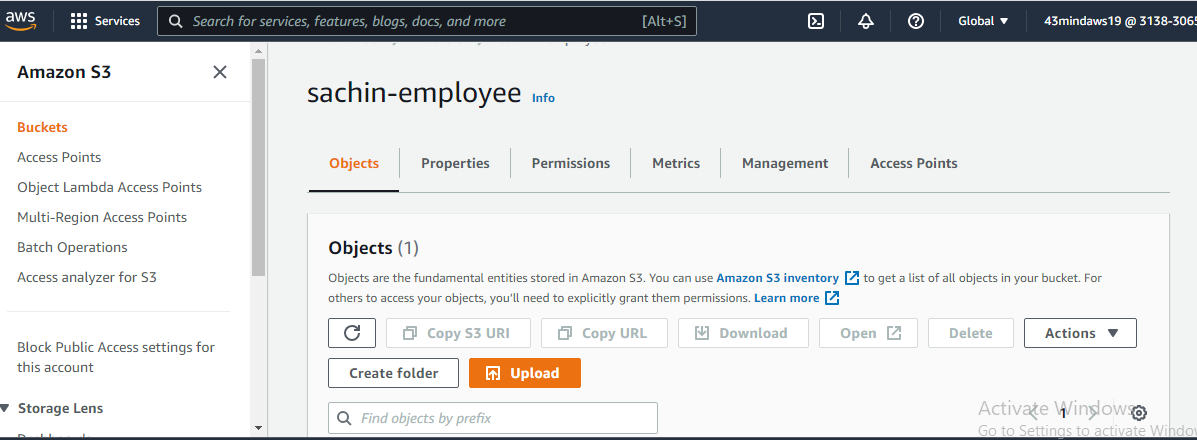
******

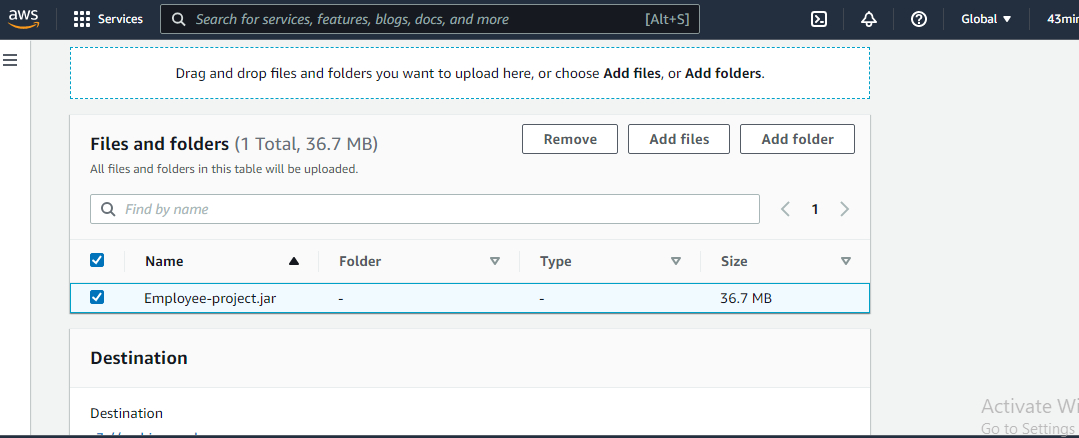
***CREATING S3 BUCKET AND UPLOADING***

***JAR FILE TO S3 BUCKET***

******

******

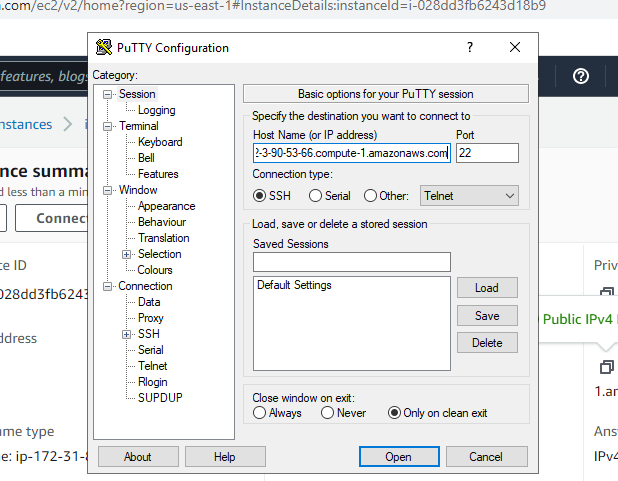
******

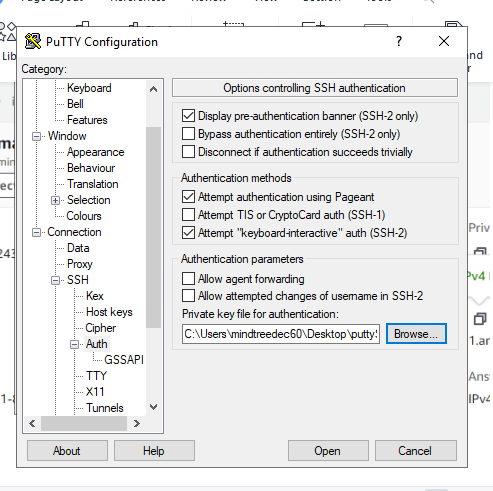
******

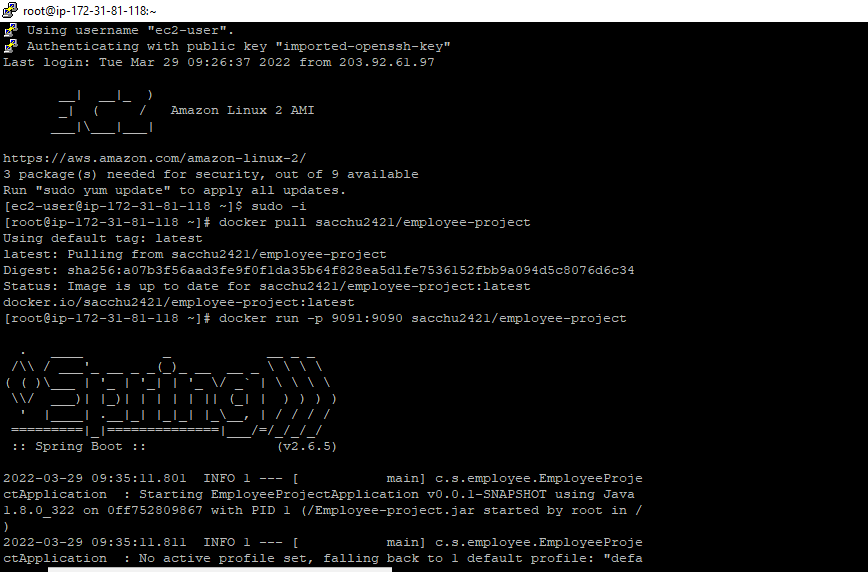
***NOW THE JAR IS DEPLOYED WE HAVE TO USE PUTTY/GITBASH TO RUN THE APP***

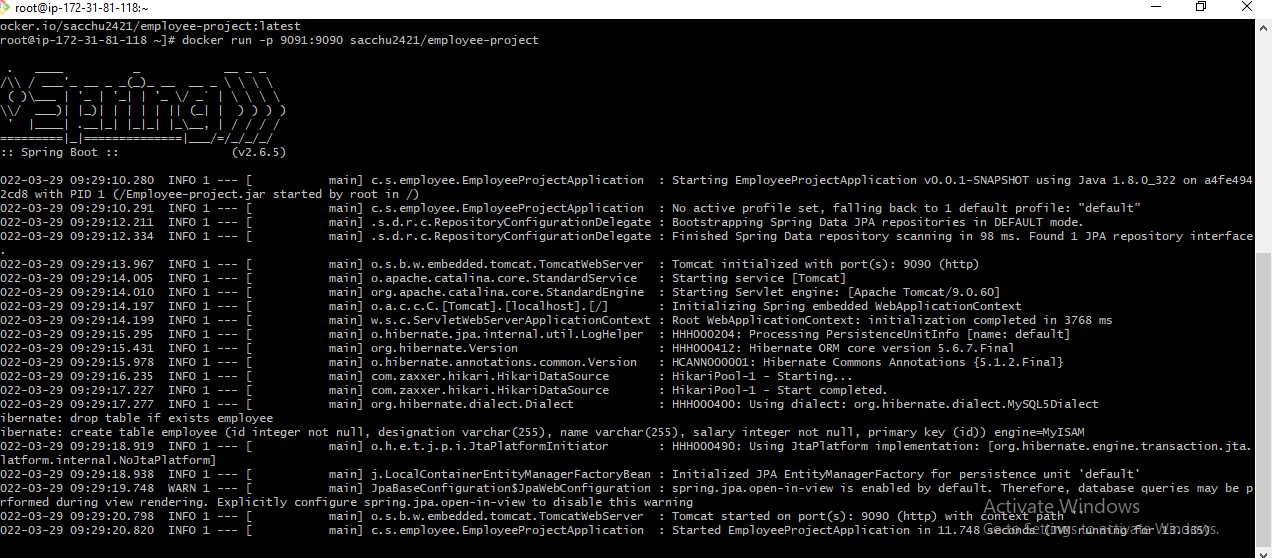
**What is PuTTY and why it is used?**

Description. PuTTY is **a free implementation of SSH (and telnet) for PCs running Microsoft Windows** (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC (for example your own or one in an internet cafe).

******

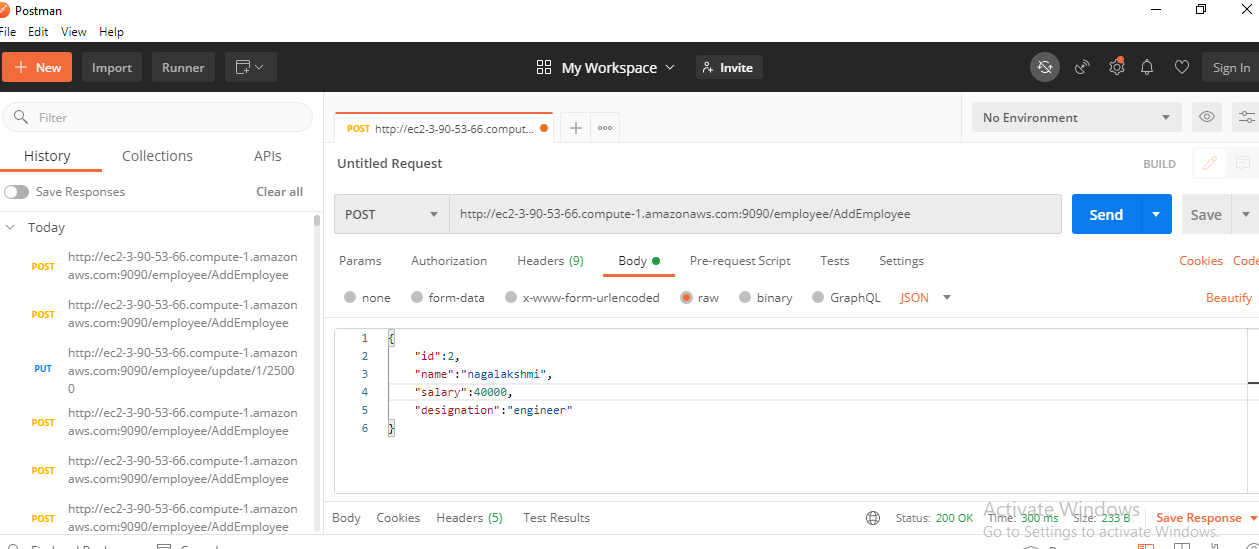
******

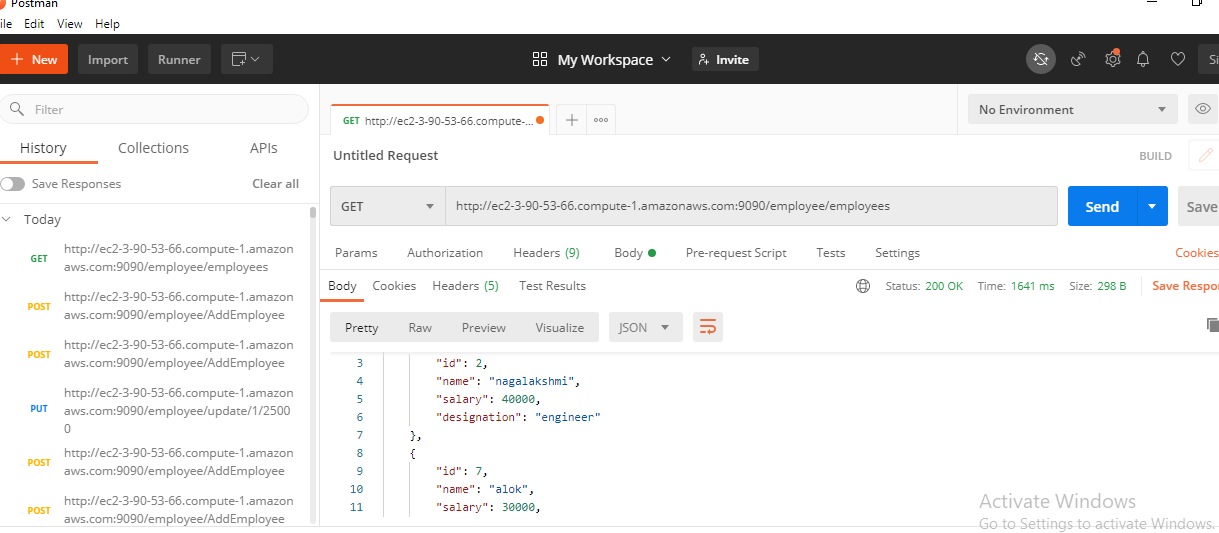
******

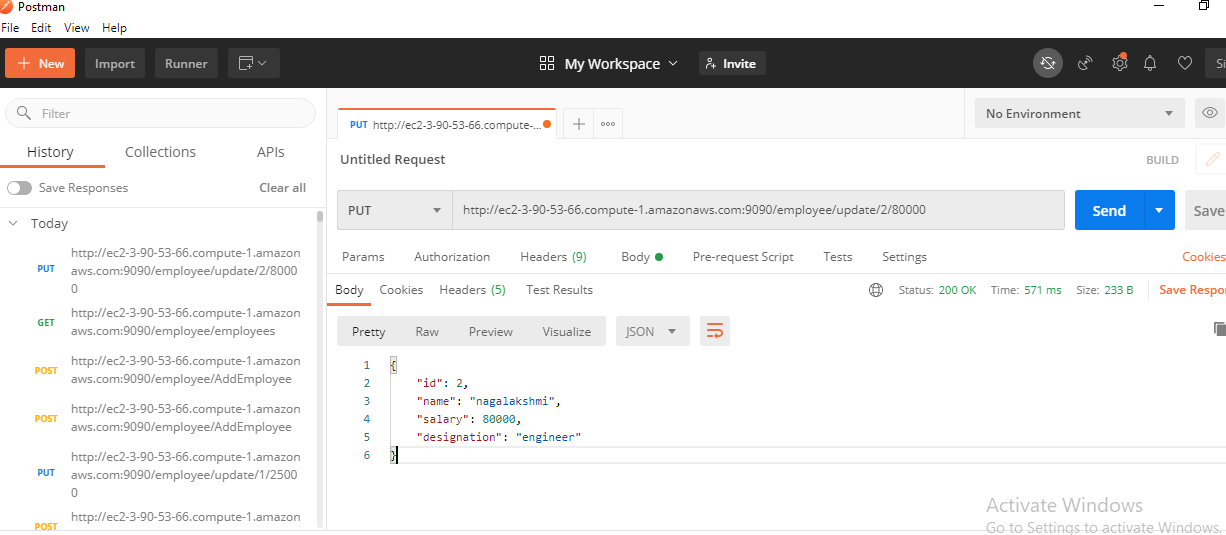
******

***TESTING THE APP USING EC2 URL IN POSTMAN***

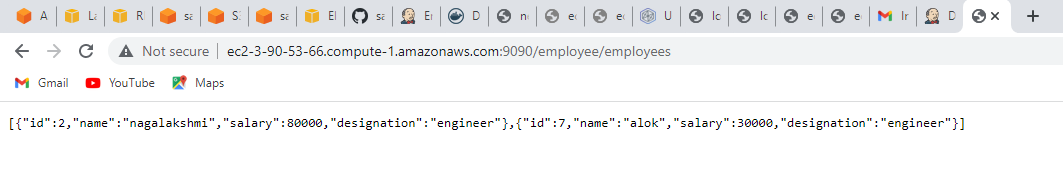
***AND BROWSER WITH ALL MAPPINGS***

******

******

******

***AT THE END CHECKING ALL THE DATA IN BROWSER USING EC2 URL***

******