Spring Boot App:

**AuthenticationSpringSecurity**

**Code:**

* CustomAuthenticationProvider.java

package com.simplilearn.demo;

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

import org.springframework.security.authentication.\*;

import org.springframework.security.core.\*;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

public class CustomAuthenticationProvider implements AuthenticationProvider {

@Autowired

UserDAO userDAO;

@Override

public Authentication authenticate(Authentication authentication) throws AuthenticationException {

String name = authentication.getName();

String pwd = authentication.getCredentials().toString();

List<User> dummyUsers= userDAO.getDetails() ;

//jdk 8 -- stream

Optional<User> authenticatedUser = dummyUsers.stream().filter(

user -> user.getName().equals(name) && user.getPwd().equals(pwd)

).findFirst();

if(!authenticatedUser.isPresent()){

throw new BadCredentialsException("Some Text");

}

List<GrantedAuthority> authorities = new ArrayList<>();

authorities.add(new SimpleGrantedAuthority(authenticatedUser.get().getRole()));

Authentication auth = new UsernamePasswordAuthenticationToken(name, pwd, authorities);

return auth;

}

@Override

public boolean supports(Class<?> aClass) {

return aClass.equals(UsernamePasswordAuthenticationToken.class);

}

}

* MainController.java

package com.simplilearn.demo;

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

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

@RestController

public class MainController {

@RequestMapping("/")

public String hello(){

return "Hello World";

}

@RequestMapping("/protected")

public String protectedHello(){

return "Hello World, i was protected";

}

@RequestMapping("/admin")

public String admin(){

return "Hello World from admin";

}

}

* AuthenticationSpringSecurityApplication.java

package com.simplilearn.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AuthenticationSpringSecurityApplication {

public static void main(String[] args) {

SpringApplication.run(AuthenticationSpringSecurityApplication.class, args);

}

}

* SpringSecurityConfig.java

package com.simplilearn.demo;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

// Protecting the urls with a role-based access.

@Override

protected void configure(HttpSecurity http) throws Exception {

http.httpBasic().and().authorizeRequests()

.antMatchers("/").permitAll()

.antMatchers("/protected").hasRole("USER")

.antMatchers("/admin").hasRole("ADMIN");

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.authenticationProvider(new CustomAuthenticationProvider());

}

}

* User.java

package com.simplilearn.demo;

public class User {

private long ID;

private String name;

private String pwd;

private String role;

public User() {

}

public long getID() {return this.ID; }

public void setID(long id) { this.ID = id;}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPwd() {

return pwd;

}

public void setPwd(String pwd) {

this.pwd = pwd;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

* UserDAO

package com.simplilearn.demo;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

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

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;

import org.springframework.stereotype.Repository;

//use autowire this FeeedbackDao object to main controller

@Repository

public class UserDAO {

@Autowired

JdbcTemplate template;

// setter method

public void setTemplate(JdbcTemplate template) {

this.template = template;

}

public List<User> getDetails() {

return template.query("select \* from users", new RowMapper<User>() {

public User mapRow(ResultSet rs, int row) throws SQLException {

User e = new User();

e.setID(rs.getInt(1));

e.setName(rs.getString(2));

e.setPwd(rs.getString(3));

e.setRole(rs.getString(4));

return e;

}

});

}

public User getByName(String name) {

User entity = template.queryForObject("select \* from users where name=?",

BeanPropertyRowMapper.newInstance(User.class), name);

return entity;

}

}

* application.properties

spring.datasource.username=root

spring.datasource.password=Evk@2412

spring.datasource.url=jdbc:mysql://localhost:3306/spring\_security

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

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>2.6.7</version>

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

</parent>

<groupId>com.simplilearn.demo</groupId>

<artifactId>AuthenticationSpringSecurity</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>AuthenticationSpringSecurity</name>

<description>AuthenticationSpringSecurity</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-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>

Step:1 AWS UBUNTU INSTANCE + Connect using Moba x-term

Step:2 Docker installation

link: https://docs.docker.com/engine/install/ubuntu/

> sudo apt-get update

> sudo apt-get install ca-certificates curl gnupg lsb-release

> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

> echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

> sudo apt-get update

> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

----------to verify docker installation

> sudo docker -v

STEP:3 INSTALL JDK

sudo apt-get update

sudo apt install default-jdk -y

To check the java version

>java --version

STEP: 4 INSTALL MAVEN

sudo apt install maven -y

Step:5 Install Jenkins

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

To start Jenkins Service

sudo service jenkins start

sudo service jenkins status

goto> aws>ec2instance>copy the ipaddress:8080 in browser to begin with Jenkins configuration

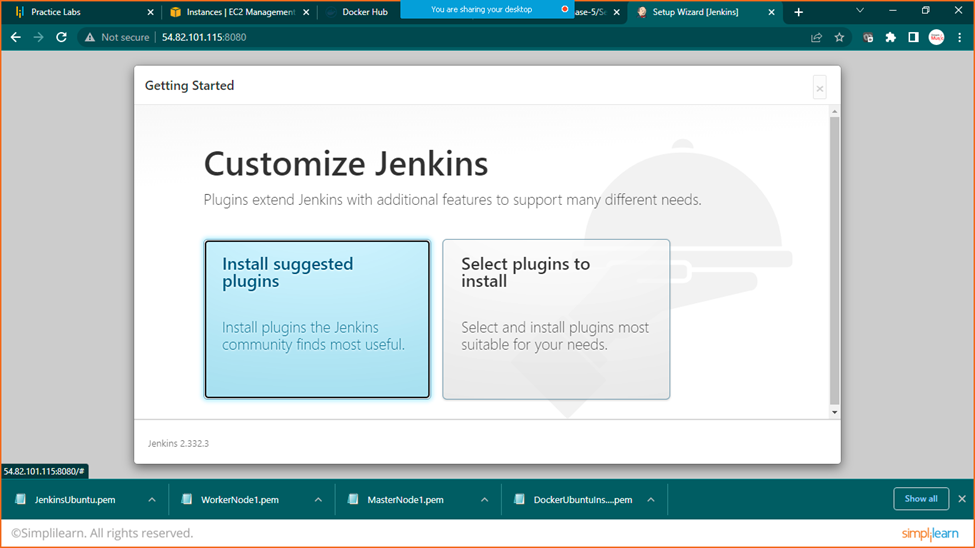
goto to the browser and enter: ipaddress:8080

to get secret password

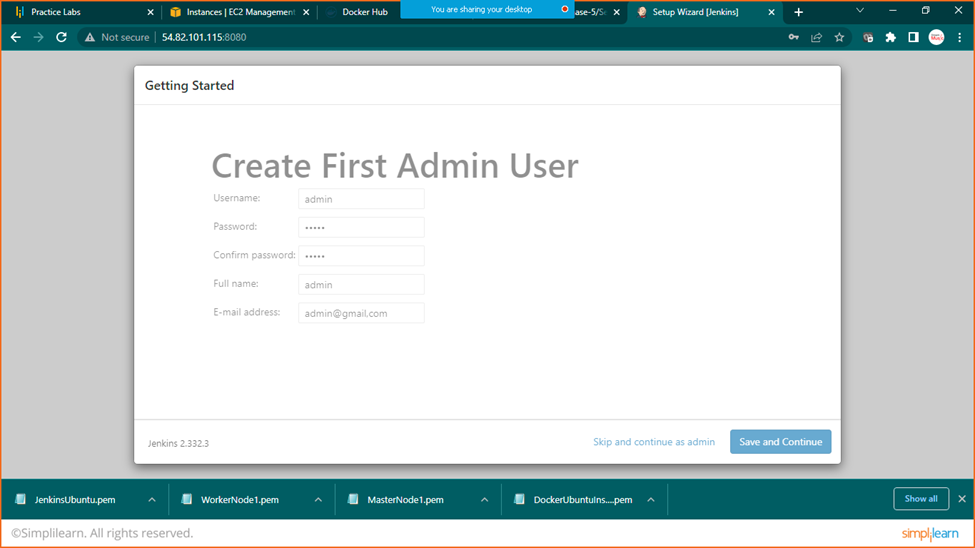
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

sudo chmod 777 /var/run/docker.sock

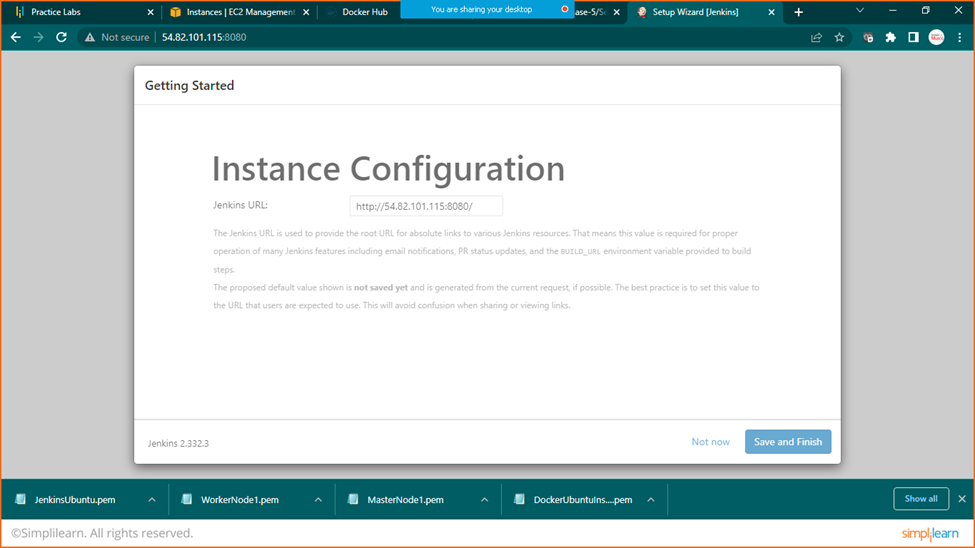
And then create a job in jenkins .



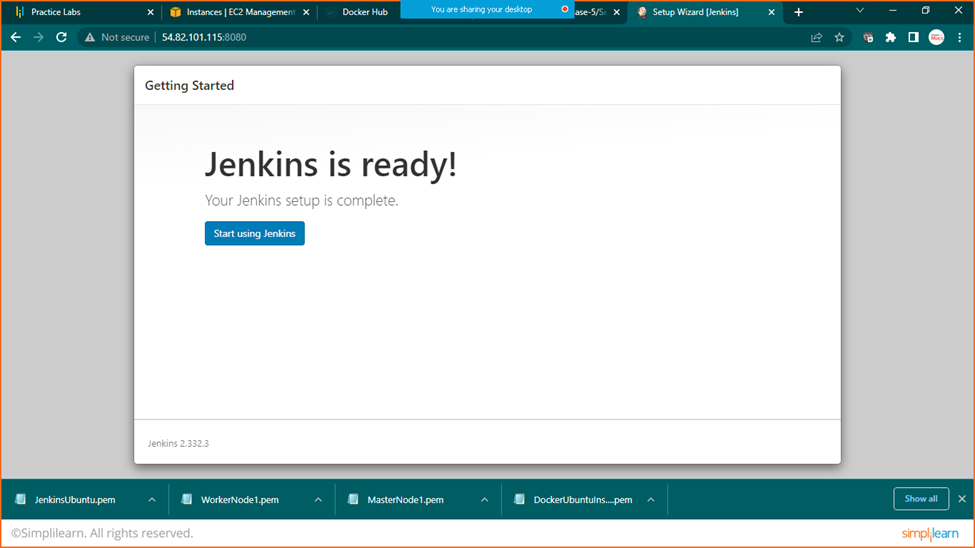
Click on install suggested plugins



Add details and click on save and continue



Click on save and finish



Start using Jenkins

