**create login page java Web project using spring boot and docker volumes**

how to crearte login page java Web project using springboot and docker volumes and how to manage docker volumes with code and command and example step by step on ubuntu 22.04 ?

ChatGPT

Creating a login page in a Java web project using Spring Boot and managing Docker volumes involves several steps. Below is a step-by-step guide to help you achieve this on Ubuntu 22.04:

**Prerequisites**

1. **Install Java Development Kit (JDK)**

sh

Copy code

sudo apt update

sudo apt install openjdk-17-jdk

1. **Install Maven**

sh

Copy code

sudo apt install maven

1. **Install Docker**

sh

Copy code

sudo apt update

sudo apt install docker.io

sudo systemctl start docker

sudo systemctl enable docker

1. **Install Docker Compose (Optional)**

sh

Copy code

sudo apt install docker-compose

**Step 1: Create a Spring Boot Project**

1. **Generate a Spring Boot project** from [Spring Initializr](https://start.spring.io/) with the following dependencies:
   * Spring Web
   * Spring Security
   * Spring Data JPA
   * H2 Database (or any database of your choice)
   * Thymeleaf (for the login page)
2. **Unzip the project** and navigate to the project directory:

sh

Copy code

unzip demo.zip

cd demo

**Step 2: Configure Spring Security**

1. **Add Spring Security configuration** in src/main/java/com/example/demo/SecurityConfig.java:

java

Copy code

package com.example.demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.antMatchers("/login").permitAll()

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.defaultSuccessURL("/")

.permitAll()

.and()

.logout()

.permitAll();

}

@Bean

@Override

public UserDetailsService userDetailsService() {

InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

manager.createUser(User.withDefaultPasswordEncoder().username("user").password("password").roles("USER").build());

return manager;

}

}

1. **Create a login page** in src/main/resources/templates/login.html:

html

Copy code

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form th:action="@{/login}" method="post">

<div>

<label>Username:</label>

<input type="text" name="username"/>

</div>

<div>

<label>Password:</label>

<input type="password" name="password"/>

</div>

<div>

<button type="submit">Login</button>

</div>

</form>

</body>

</html>

**Step 3: Build the Spring Boot Application**

sh

Copy code

mvn clean package

**Step 4: Create Dockerfile**

1. **Create a Dockerfile** in the project root:

Dockerfile

Copy code

FROM openjdk:17-jdk-slim

VOLUME /tmp

ARG JAR\_FILE=target/demo-0.0.1-SNAPSHOT.jar

COPY ${JAR\_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

**Step 5: Create Docker Compose File (Optional)**

1. **Create docker-compose.yml** in the project root:

yaml

Copy code

version: '3.8'

services:

web:

build: .

ports:

- "8080:8080"

volumes:

- ./data:/data

**Step 6: Build and Run Docker Container**

1. **Build the Docker image**:

sh

Copy code

docker build -t springboot-demo .

1. **Run the Docker container**:

sh

Copy code

docker run -d -p 8080:8080 -v $(pwd)/data:/data springboot-demo

**Step 7: Access the Application**

1. **Open a web browser** and navigate to http://localhost:8080/login.
2. **Login using the credentials**:
   * Username: user
   * Password: password

**Managing Docker Volumes**

1. **Create a Docker volume**:

sh

Copy code

docker volume create my\_volume

1. **List Docker volumes**:

sh

Copy code

docker volume ls

1. **Inspect a Docker volume**:

sh

Copy code

docker volume inspect my\_volume

1. **Remove a Docker volume**:

sh

Copy code

docker volume rm my\_volume

**Using Docker Volumes in Docker Compose**

1. **Define volumes in docker-compose.yml**:

yaml

Copy code

version: '3.8'

services:

web:

build: .

ports:

- "8080:8080"

volumes:

- my\_volume:/data

volumes:

my\_volume:

1. **Run Docker Compose**:

sh

Copy code

docker-compose up

By following these steps, you can set up a login page for a Java web project using Spring Boot, Dockerize the application, and manage Docker volumes effectively on Ubuntu 22.04.

4o