tictactoeCloudAWS

TicTacToe Game with Cognito Authentication and RDS database deployed on EC2 instance.

This guide is intended for users with Windows OS!!!

Setting up and connecting to RDS

1. To set up RDS follow this instruction: https://aws.plainenglish.io/deploy-spring-boot-application-with-amazon-rds-7cec634ef3a1
2. When RDS is set up, add this to your application.properties:

   spring.jpa.hibernate.ddl-auto=update
   spring.datasource.url=jdbc:mysql://:/<database \name>
   spring.datasource.username=username
   spring.datasource.password=password
   spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

3. Add this dependency to your pom.xml file:

   <dependency>
   <groupId>com.mysql</groupId>
   <artifactId>mysql-connector-j</artifactId>
   <scope>runtime</scope>
   </dependency>

4. After all this your database should be connected to your app.

Cognito configuration and app integration

1. Create user pool using the beginning of this vide tutorial: https://www.youtube.com/watch?v=o2IM9oI6Eqk . Only two differences are:
   - enable ALLOW_USER_PASSWORD_AUTH in the Client Authentication Flow
   - don't use Hosted UI
2. Connect Cognito with your app using this guide: https://dev.to/daviidy/api-security-how-to-implement-authentication-and-authorization-with-aws-cognito-in-spring-boot-4713?fbclid=IwAR1RlEKeoMiZwmdQf8b9IOl-8C1DKezTgGCButUdDape5mgLguxveRD9jQQ
3. Make sure that your application.properties file contains these lines:

   spring.security.oauth2.client.registration.cognito.client-id=<client ID>
   spring.security.oauth2.client.registration.cognito.client-secret=<client secret>
   spring.security.oauth2.client.registration.cognito.scope=openid

spring.security.oauth2.client.provider.cognito.issuer-uri=https://cognito-idp.<region>.amazonaws.com/<User Pool ID>
spring.security.oauth2.client.registration.cognito.client-name=FrontAppClient
aws.accessKeyId=<AWS access key>
aws.secretKey=<AWS secret key>
aws.region=<aws_region>
spring.security.oauth2.resourceserver.jwt.jwk-set-uri=https://cognito-idp.<region>.amazonaws.com/<User Pool ID>/.well-known/jwks.json
sample.jwe-key-value= classpath:simple.priv

EC2 Configuration and Docker Image Deployment

1. Create EC2 instance and make sure that it is launched. Make sure to allow port 8080 in Inbound Rules in Security Groups.
2. Download PuTTY, emulator that will allow us to ssh into our EC2 instance.
3. Connect to your EC2 instance using PuTTY with the help of this tutorial: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html
4. After establishing connection, download docker. To do that run this commands:
   > sudo yum update -y sudo yum install docker sudo service docker start
5. To build docker image you have to move your app .jar file to EC2 instance:
   - run 'mvn clean package' in your IDE terminal
   - after .jar file is created run this command in your cmd:
     > pscp -i path\to\your.ppk path\to.jar\file ec2-user@"Public DNS":/home/ec2-user
6. Make sure that .jar file was moved to PuTTY, using > ls command
7. Create Dockerfile, it should look like this:

   ```
   FROM openjdk:17-oracle
   ARG JAR_FILE=<jar_file_name>.jar
   COPY ${JAR_FILE} .
   EXPOSE 8080
   CMD [ "java", "-jar", "/<jar_file_name>.jar"]
   ```

8. Move Dockerfile.prod to EC2 instance as well:
   > pscp -i path\to\your.ppk path\to\Dockerfile ec2-user@<Public DNS>:/home/ec2-user
9. To build and run docker image, run this commands:
   > docker build -t <your_choice>/docker -f Dockerfile.prod . docker run -p 8080:8080 <your_choice>/docker
10. Now your app should beaccessible from http://<Public DNS>:8080