SWE 645 – ASSIGNMENT 3

Group Members:

1. **Meghana Kancherla**
   Gno: G01379905
2. **Hemanth Sani**
   Gno: G01379929
3. **Rishitha Sai Kavuru**
   Gno: G01394551
4. **Roshan Vemula**
   Gno: G01389688

Links:

GET and POST URL: https://ec2-44-212-55-21.compute-1.amazonaws.com/k8s/clusters/c-ct5z8/api/v1/namespaces/hw3namespace/services/http:hw3loadbalancer:8080/proxy/api/students

Github URL: https://github.com/MeghanaKancherla/SWE645HW3_Database

Credentials:

Rancher credentials:

User ID: admin

Password: Rancher@1234

URL: https://ec2-44-212-55-21.compute-1.amazonaws.com/dashboard

Jenkins credentials:

User ID: swe645hw2_jenkins

Password: Jenkins@123

URL: http://ec2-18-206-118-33.compute-1.amazonaws.com:8080/

Docker Hub credentials:

User ID: meghanakancherla

Password: Docker@123

Steps for creating AWS RDS and connecting it to Eclipse and creating a CI/CD pipeline

Steps to create AWS RDS:

1. Login to AWS and selecte “RDS” under services.
2. Select ‘Standard create’.
3. Select ‘MSQL” as the engine.
4. Select the template as ‘Dev/Test’
5. Give and DB identifier name and set the master username and password.
6. Select instance configuration as ‘Burstable classes’
7. Select storage as ‘General purpose SSD’ and allocated storage as ‘5’.
8. Select ‘yes’ for public access.
9. Create a new VPC security group.
10. Under additional configuration give a database name.
11. Leave everything else as default values and then create database.

Steps to create a springboot application and connect AWS RDS to Eclipse IDE and running the application:

1. To create a Spring boot application, go to ‘https://start.spring.io/’. Here select project as ‘Maven’, language as ‘Java’, Spring boot as ‘3.1.2’, then provide the artifact name, packaging as ‘jar’, java version as ‘17’ and add the dependencies: ‘Spring Web’, ‘Spring Data JPA’, and ‘MySQL Driver’. Then download the zip file.
2. Open the file in any IDE, I have used ‘Eclipse IDE’. Then create StudentEntity, Controller, Repository, Service and ServiceImplementation classes. Write the necessary code in all the files.
3. Download ‘mysql connector java’ from google.
4. To connect the database, In the menu select preference and click on other then select database (if not found add it).
5. Select ‘New’ and from the list select ‘MySQL’ and then click next. Then select the small circular icon on the right. From the jar list, add the ‘mysql-connector-java-5.1.38-bin’ jar file.
6. Now you can give the connection details, the database name, URL (it’s the endpoint from RDS preceded by jdbc:mysql://url/database_name. Then provide the username and password mentioned while creating the RDS connection.
7. Click on ‘Test connection’, if it says ‘Ping succeeded’ then it’s successfully connected.
8. I have also connected the database to ‘MySQL workbench’, Then I created a table and added all the columns to it.
9. Now to run the spring boot application. In eclipse terminal type ‘mvn clean package’. This command builds the project and generates a jar file under target folder.

10. We start the springboot application by running the jar file. ‘java -jar location-of-jar-file’

Using postman for running the commands:

1. Install postman.
2. For the post command, select POST and type in the POST URL(in my case http://localhost:8080/api/students) based on the post url you have in your code, then type the insert values under the body. It should be in json format. The values will get added to the database.
3. Using GET, put the get url and then you will be able to see the values for the table in JSON format.

Creating a Dockerfile, Jenkinsfile and connecting the image to Rancher and creating a CI/CD pipeline.

1. Now to connect to Kubernetes, we need to push the code into git hub. We need to create a Dockerfile and Jenkins file and then add it to the root directory of the repository.
2. In the Dockerfile, we should first give the FROM command where we set the base image from ‘ubuntu’.
3. Then we need to install java and unzip the file. I created a directory and then in that directory I copied the jar file from the target folder.
   Note: To add the target folder into git, we need to remove it from gitignore file.
4. Now we need to run this jar file. “CMD [“/usr/java/bin”, “-jar”, “file.jar”]”.
5. These are the commands to be added in the docker file.
6. In Jenkins file, almost all the commands are same as in HW2. We just need to update the file paths, docker image and cluster names.
7. Then we run the Jenkins file. We can see a new image is created in docker hub and then we can see that a pipeline is created and using the node port or load balancer url we can access the database results. We can perform delete and update operation as well by appending the url with the id.