**Homework Assignment 3 Report** **05/05/2023**
**SWE-645, Spring 2023**

**Name : Ashrit Mullur Ravi**
**GMU ID : G01391973**

**Name : Vaishnavi Gururaj**
**GMU ID : G01399296**

**Name : Harsha Masandrapalya Vanarajaiah**
**GMU ID : G01396731**

**The GitHub link for the Project is https://github.com/Ashrit26/swe645_Assignment3**

**Part 1:**

**Develop a microservices-based application using SpringBoot, RESTful Web Services, and JPA/Hibernate along with Amazon Relational Database Service (Amazon RDS)/MySQL to persist and read data to/from a relational database.**

We developed the application SpringBoot in IntelliJ IDEA IDE
We started with https://start.spring.io/ to create and download a maven project and imported it into IntelliJ IDEA IDE, and then implemented the rest of the logic.

After importing, we added the necessary dependencies for Spring Web, Spring Data JPA, and MySQL Connector.
Then defined the JPA entity for the survey form in the Surveydata.java file and map it to a MySQL database table using annotations.
Then created a Spring Data JPA repository interface in SurveyRepo file to perform CRUD (Create, Read, Update, Delete) operations on the survey feedback entity.
Created a REST controller SurveyController.java file to handle the survey form endpoints.

Then built the project using Maven to get the target folder.

Later we created an environment to run your MySQL database using the Amazon Relational Database Service (Amazon RDS). We followed the following link to achieve the same.

https://aws.amazon.com/getting-started/hands-on/create-mysql-db/

Configured the MySQL database connection properties in the application.properties file with the necessary credentials.

Finally, we ran the project on the 8080 port, then used Postman (https://www.postman.com/) to test the working of our application. We used the POST method to add some data to the surveydata/add endpoint. For the GET method, we used surveydata/all endpoint to fetch the added data.

**Part 2:**

**Containerizing the microservices-based application that we had developed in Part 1, using Docker container technology.**

First, we installed the docker desktop in our machine. Then, we created an account on https://hub.docker.com/.
We then created a file called Dockerfile in Eclipse after which using the warfile which was built in the Homework-1, we built the docker image using the following command.
“sudo docker build –platform linux/amd64 -t swe645assignment3:0.1 .”

Then, we used the following command to verify the image:
“sudo docker run -it -p 8182:8080 swe645assignment3:0.1”

After ensuring the image was accurate, we pushed the image to docker hub using our credentials.

We then verified our image on the hub which was accessible from the internet.

**Part 3:**

**Deploying the containerized application on the open-source container orchestration platform Kubernetes to enable scalability and resiliency of the application.**

We created an AWS EC2 instance with Ubuntu Server 22.04 LTS (HVM), SSD Volume Type AMI. While creating the instance, we enabled public security and downloaded the private key (pem file (swe645assignment2.pem)) to access the server.
Under Instance Type we chose t2.large. We tried to run it multiple times on other types under free tiers like medium but since Rancher didn’t start, we used the large type.

We then executed this command to ensure that our key was not publicly viewable
“chmod 400 swe645assignment2.pem”

For connecting to the AWS EC2 machine, the following linux command was used:
ssh -i "swe645assignment2.pem" ubuntu@ec2-3-223-138-39.compute-1.amazonaws.com

Then update using following command:
“sudo apt-get update”

We installed docker using following command:
“sudo apt install docker.io”
After installing docker, we enabled the rancher image using the following command:
“sudo docker run --privileged=true -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher”

We could then open the Public IPv4 address from the instance in the browser to see the dashboard of rancher.
https://3.223.138.39/dashboard/home

The credentials of Rancher dashboard are:
Username: admin
Password: admin@123456

For security purposes, we then created our own password and logged in as an admin user.
After this, we created the custom cluster with our new cluster name. This cluster hosted our etcd, control panel, and worker nodes.
Once the cluster was active, it was ready for deployment.

Then under deployment, we created two deployments namely NodePort and LoadBalancer with their Private port as 8080. Also added our container image which we had pushed to the Docker Hub in part -1.
After the deployment, the pods become active for some time.

Clicking on the node port services lets us view our survey page. It renders to an address on the browser, we edited it by adding our war file name.
https://3.223.138.39/k8s/clusters/c-m-nj4fw7dj/api/v1/namespaces/default/services/http:project3:8080/proxy/surveydata/all

We then downloaded and saved the KubeConfig file (project3.yaml) for the cluster.

**Part 4:**
**Establishing a CI/CD pipeline that includes a Git source code repository at GitHub, and Jenkins for automated build and for the automated deployment of our application on Kubernetes**

We created an AWS EC2 instance for Jenkins with Ubuntu Server 22.04 LTS (HVM), SSD Volume Type AMI. While creating the instance, we enabled public security and downloaded the private key (pem file (swe645assignment2.pem)) to access the server.
Under Instance Type we chose t2.large. We tried to run it multiple times on other types under free tiers like medium but since Rancher didn't start, we used the large type.

We then executed this command to ensure that our key was not publicly viewable
"chmod 400 swe645assignment2.pem"

For connecting to the AWS EC2 machine, the following linux command was used:
ssh -i "swe645assignment2.pem" ubuntu@ec2-34-206-216-101.compute-1.amazonaws.com

Then update using following command:
"sudo apt-get update"

We installed docker using following command:
"sudo apt install docker.io"

Then we followed the instructions provided in this link https://pkg.jenkins.io/debian/ to install jenkins on our instance and configured it with necessary requirements

We used the Public IPv4 address from the instance to view the dashboard of Jenkins in our browser.
http://34.206.216.101:8080/

The Credentials of Jenkins Dashboard are:
Username: project2
Password: admin@123456

We created a Jenkins pipeline and installed necessary plugins as required. (CloudBees docker & Rancher).
(https://docs.cloudbees.com/docs/admin-resources/latest/plugins/docker-workflow/
https://plugins.jenkins.io/rancher/)

We configured the pipeline with the source as our GitHub repo in such a way that it polls the SCM every minute (*/1****) and provided access to JenkinsFile stored in our GitHub repository.

A JenkinsFile was created and pushed, which includes several stages such as build, push to Docker Hub and deployment on Rancher single node and Rancher.

We also had to add the credentials of both GitHub and DockerHub under the manage credentials section.

After setting up the pipeline successfully, any changes made to the code and pushed to GitHub triggered the SCM polling process, which in turn initiated a new build. Once all the necessary steps were completed, the newly built code was pushed to Docker Hub, and the image was updated in Rancher deployments, thereby hosting the latest build.
To verify our tasks, we added a few dummy data to the MYSQL database, the same was able to be updated in Rancher deployments, thereby hosting the latest build.