Name – Arpit Gupta
Student Id – 169572237

CLO835 - Assignment 2

Report: Deployment of Containerized Applications in Kubernetes (Assignment 2)

GitHub Repo: Arpit-commits/Arpit-CLO835-Assignment2
Demo Link : https://drive.google.com/file/d/1aHd1nar4HPpvxe9p4BCrZNvTHK0ozDux/view?usp=sharing

I am using branch structure where
Branch 1 – Version1 – is my first attempt where I am just loading a text based webpage and adding some changes like adding text into html file and by doing everything creating a new image and deploying it.

Branch 2 – Version2 – in this I have updated my code and taken that code from our 1st assignment submission, I have made all the changes according to that.

I have done the attempt one where I am running everything locally then I have Used this Version 1 & Version 2 to deploy the changes

I have also created everything from terraform and all the dependencies are installed using that , terraform file had 20gb volume and its assigning 2gb swap memory to run everything properly as I was facing low memory issue previously and I am creating ecr repo from the terraform also.

Terraform File is saved in my private github repository : Arpit-commits/Arpit-CLO835-Assignment2-TerraformFiles

Question And Answer :

1. **Deployment of local single node cluster**

    a. What is the IP of the K8s API server in your cluster?

    Answer: https://127.0.0.1:40079

2. **Deployment of MySQL and web application pods**

    - **b. Can both applications listen on the same port inside the container? Explain your answer.**

    Answer: No, both applications cannot listen on the same port inside the container because the Flask app uses port 8080 and MySQL uses port 3306, and each container runs a single process that requires a unique port to avoid conflicts.

    - **c. Connect to the server running web application pod and get a valid response.**

    Demonstrated in video with curl http://3.81.39.64:30000/

    - **d. Examine the logs of the invoked application to demonstrate the response from the server was reflected in the log file.**

    Demonstrated in video with kubectl logs flask-replicaset-58k8n -n flask

3. **Deploy ReplicaSets of the applications with 3 replicas**

    - **Explanation: The pod created in step 2 (from the initial Deployment) is not governed by the ReplicaSet created in step 3 because the Deployment creates its own ReplicaSet, which is independent of the manually applied ReplicaSet. The ReplicaSet pods failed initially due to a missing environment and db port from app.py file but are now running after the fix.**

    4 . Is the replicaset created in step 3 part of this deployment? Explain.

    Answer: No, the ReplicaSet created is not part of this deployment. Deployment creates its own ReplicaSet of 3

4. **Expose web application on NodePort 30000**

    - **Demonstrated in Video.**

5. **Update the image version in the deployment manifest**

    - **Demonstrated in video (updated to v3.0).**

6. **Explain the reason we are using different service types**

- **Answer: The Flask app uses a NodePort service to expose port 30000 externally for browser and curl access, while MySQL uses a ClusterIP service for internal cluster access only**

Error Report Summary

ECR Image Not Getting Uploaded

- **Problem:** I couldn't send my container image to AWS ECR, which stopped the workflow.
- **Solution:** I checked and fixed my AWS configuration and credentials, making sure the image could be uploaded properly.

2. **Pods Experiencing ECR Pull Issues**
   - **Problem:** My pods were having trouble downloading the image from ECR, which led to errors during deployment.
   - **Solution:** I updated the authentication settings (image pull secrets) so the pods could access and pull the image without any issues.
3. **IP Not Forwarding to Port 30000**
   - **Problem:** The expected traffic wasn't reaching port 30000, causing connection problems.
   - **Solution:** I adjusted the security group settings to open the necessary inbound rule, which allowed the IP to correctly forward traffic to port 30000.
4. **Missing kind-config File**
   - **Problem:** The configuration file needed to set up my local Kubernetes cluster (kind) was missing, so the cluster couldn't start properly.
   - **Solution:** I created the required kind-config file with the right settings, which allowed the local cluster to initialize and run smoothly.
5. **Curl Not Working Inside the Pod**
   - **Problem:** The curl command wasn't available inside the pod, making it difficult to test connectivity and diagnose issues.
   - **Solution:** I manually installed the curl package in the pod, enabling me to perform the necessary tests and troubleshoot effectively.

6 . **CrashLoopBackOff in Flask ReplicaSet Pods**

- **Error:** The Flask ReplicaSet pods (flask-replicaset-*) were in a CrashLoopBackOff state with multiple restarts
- Solution : The DBPORT environment variable was missing in flask-replicaset.yaml, causing a ' in app.py when trying to convert os.environ.get("DBPORT") to an integer.
  updated the app.py for db port added limits for memory