**Udagram by microservices**

This repository is refracted a monolith Udagram application into microservices. Ionic was used in the fronend and in the backend Docker/kubernetes are used.

**About the Project - Udagram Image Filtering Microservice**

Udagram is a simple cloud application developed alongside the Udacity Cloud Engineering Nanodegree. It allows users to register and log into a web client, post photos to the feed, and process photos using an image filtering microservice. Following are the services involved in this project:

* “user” - allows users to register and log into a web client,
* “feed” - allows users to post photos, and process photos using image filtering
* “frontend” - acts as an interface between the user and the backend-services
* "reverseproxy" - For resolving multiple services running on same port in separate containers

Correspondingly, the project is split into following parts:

1. The RestAPI Feed Backend, a Node-Express feed microservice.
2. The RestAPI User Backend, a Node-Express user microservice.
3. The Simple Frontend - A basic Ionic client web application which consumes the RestAPI Backend.
4. Nginx as a reverse-proxy server, when different backend services are running on the same port, then a reverse proxy server directs client requests to the appropriate backend server and retrieves resources on behalf of the client.

**Dependencies and Getting Setup**

## Containers

1. Navigate to Microservice->Container->exercise->deployment->docker
2. For running the app properly, it requires user AWS account credentials in root/.aws/ folder, and a AWS RDS database of which the following information should be stored in environment variables. POSTGRESS\_USERNAME: $POSTGRESS\_USERNAME POSTGRESS\_PASSWORD: $POSTGRESS\_PASSWORD POSTGRESS\_DB: $POSTGRESS\_DB POSTGRESS\_HOST: $POSTGRESS\_HOST AWS\_REGION: $AWS\_REGION AWS\_PROFILE: $AWS\_PROFILE AWS\_BUCKET: $AWS\_BUCKET JWT\_SECRET: $JWT\_SECRET
3. If step 2 is configured peoperly, in the terminal "docker-compose up" would start the containers from images, and start the application in "[http://localhost:8100](http://localhost:8100/)".

## Kubernetes

(This requires running kubernetes cluster. For running locally, user can start by "minikube start", assuming user has minikube installed.)

1. Navigate to Microservice->Container->exercise->deployment->k8
2. Similar to step 2 of Container section above, user should configure the variables in configmap-description.yaml. For aws-secret.yaml and env-secret-file.yaml, user should encode their secrets in base64 format, by "echo "secretString" | base64" in the terminal, and changing approapriate values in those two files.
3. The following services should be started by the command "kubectl apply -f "service.yaml"" i. backend-feed-service.yaml ii. backend-user-service.yaml iii. frontend-service.yaml iv. reverseproxy-service.yaml
4. The following deployments should be started by the command "kubectl apply -f "deployment.yaml"" i. backend-feed-deployment.yaml ii. backend-user-deployment.yaml iii. frontend-deployment.yaml iv. reverseproxy-deployment.yaml
5. User can observe nodes by "kubectl get nodes".