**272: Assignment 1 - Hello World Microservice**

**Overview**

**Service name : Hello World**

**Description**: There are two microservices, Hello with one endpoint /Hello and another World, with /world endpoint. These both are dockerized and deployed on Kubernetes. The third service Hello World makes a call to these and returns the combined result.

## **Table of Contents**

1. Introduction
2. Prerequisites
3. Setup and Installation
4. Usage
5. Configuration
6. Troubleshooting

## **Introduction**

Hello-microservice and World-microservice are RESTful API services that return a Hello message and world message respectively. It is built using Node.js and Express, Dockerized for containerization, and deployed on Kubernetes for scalability and management.

## **Prerequisites**

Before starting, ensure you have the following installed:

* Docker
* Kubernetes
* kubectl
* Docker Buildx

**Setup and installation**

* Docker Setup
  + Clone the repository hello and world separately
  + git clone <https://github.com/DrishtiSachdev7/272-assignment1/tree/assign1/Hello>
  + cd hello
  + git clone <https://github.com/DrishtiSachdev7/272-assignment1/tree/assign1/World>
  + cd world
* Build the docker image
  + docker build -t dockerhub-username/hello-microservice:latest .
  + docker build -t dockerhub-username/world-microservice:latest .
* Push the docker image
  + docker push your-dockerhub-username/hello-microservice:latest
  + docker push your-dockerhub-username/world-microservice:latest
* Kubernetes setup
  + For using Google Kubernetes Engine (GKE) using Google Cloud Provider,
  + Setup google cloud environment and enable Google Kubernetes Engine API.
  + Install and configure Google Cloud SDK
    - Google cloud SDK installation for specific OS, <https://cloud.google.com/sdk/docs/install>
    - Authenticate and set project, run gcloud init.
  + Create a Kubernetes Cluster on Google Kubernetes Engine
    - * This can be done using the gcloud command line tool.
  + After creating the cluster, configure kubectl to use it.
  + The docker image that was created earlier has to be pushed to Google Cloud Registry.
    - Authenticate Docker to use GCR
    - Build and push the Docker image on GCR
      * docker build -t gcr.io/PROJECT\_ID/hello-microservice:latest .
      * docker push gcr.io/PROJECT\_ID/hello-microservice:latest
  + Deploy Application to GKE
    - * Create a deployment.yaml and service.yaml file respectively for hello and world microservices, as pushed in the hello and world folders.
    - Apply the configuration file using kubectl.
    - Verify the deployment, check the status of the pods, check the service and logs.
      * kubectl get pods
      * kubectl get services
      * kubectl logs <pod-name>
      * kubectl describe pod <pod-name>
      * kubectl describe svc hello-microservice

**Usage**

Once deployed, the microservice can be accessed through the LoadBalancer's external IP.

kubectl get svc hello-microservice

kubectl get svc world-microservice

For the api call, http://<EXTERNAL-IP>/api/hello

**Configuration**

Port: Hello service runs on port 3000 inside the container and World service on port 8080 inside the container and 80 externally.

**Troubleshooting**

**Common issues**

1. Pod Status: ImageBackPullOff
   1. Ensure the docker image is correctly pushed and tagged
   2. Verify that the image repository URL and tag are correct in your deployment YAML.
2. Service not exposing
   1. Ensure that the LoadBalancer service is correctly applied and that the external IP is provisioned.
   2. Check for port numbers exposed in the service at all places.
3. exec format error
   1. This usually happens when the Docker image is built for a different CPU architecture than the one on which it is being run. For example, if the image is built for ARM architecture and we are trying to run it on an x86\_64 (AMD64) system, the error will occur
   2. To resolve this, verify docker image architecture, and build and push the docker image for the correct architecture.
   3. We can use buildx for multi architecture support

docker buildx create --use

docker buildx build -platform linux/amd64,linux/arm64 -t gcr.io/<PROJECT\_ID>/<service>

* 1. Reapply deployment

**Links to docker images,**

[**https://console.cloud.google.com/artifacts/docker/augmented-ward-434500-u6/us/gcr.io/my-app?project=augmented-ward-434500-u6**](https://console.cloud.google.com/artifacts/docker/augmented-ward-434500-u6/us/gcr.io/my-app?project=augmented-ward-434500-u6)

[**https://console.cloud.google.com/artifacts/docker/augmented-ward-434500-u6/us/gcr.io/world-microservice?project=augmented-ward-434500-u6**](https://console.cloud.google.com/artifacts/docker/augmented-ward-434500-u6/us/gcr.io/world-microservice?project=augmented-ward-434500-u6)