



FACULTY OF ENGINEERING AND APPLIED SCIENCE

SOFE 3980U Software Quality Winter 2023

LAB THREE

Deploying using Google Kubernetes Engine

Ontario Tech University

SOFE 3980U Software Quality

CRN: 73385

Instructor(s): Mohamed El-Darieby

2022-03-09

Tasfia Alam – 100584647

Ahmaad Ansari – 100785574

Jacky Lee – 100787870

Mohammad Mohammadkhani – 100798165

Ashley Tyrone – 100786450

Table of Contents

I. Introduction	3
A. Docker	3
B. Google Kubernetes Engine	3
C. Video Submission	3
II. Discussion	4
A. Docker and Kubernetes Summary	4
B. Advantages and Disadvantages of Docker Images	5
C. Deploying Application	5
III. Conclusion	8

I. Introduction

A. Docker

Docker is a popular platform that enables developers to build, ship, and run distributed applications in containers. It is an open-source technology that uses containerization to create isolated and portable environments for applications, which are then run on any host system. Docker containers provide a lightweight and consistent runtime environment, making it easier to develop, test, and deploy applications across different platforms.

B. Google Kubernetes Engine

Google Kubernetes Engine (GKE) is a fully-managed Kubernetes service provided by Google Cloud Platform (GCP). Kubernetes is an open-source container orchestration system designed to automate the deployment, scaling, and management of containerized applications. It is a powerful tool for developers who want to streamline their application deployment process and ensure high availability, scalability, and security. GKE provides a simple and flexible way to deploy containerized applications to a managed Kubernetes cluster without worrying about the underlying infrastructure. GKE handles the setup and management of the cluster, including nodes, networking, and security, so developers can focus on building and improving their applications.

C. Video Submission

Click [here](#) for the video submissions. If the hyperlink is not working copy and paste the following link into your browser:

`https://drive.google.com/drive/folders/1efKbOZVvB2wYIFjrgz1W2bKLZFzuLwa?usp=share_link`

The video can also be found in our [GitHub](#) repository. If the hyperlink is not working copy and paste the following link into your browser:

`https://github.com/ahmaad-ansari/SOFE3980U-Lab3-D2/tree/main`

II. Discussion

A. Docker and Kubernetes Summary

Docker is a containerization platform that allows developers to package their applications and dependencies into lightweight containers that can run consistently across different environments. It uses a layered filesystem and a client-server architecture to create, run, and manage containers. Some key Docker terminologies are defined below.

- **Image:** a read-only template that defines the base filesystem and configuration of a container.
- **Container:** a lightweight, executable package that includes everything needed to run an application, including code, runtime, system tools, libraries, and settings.
- **Dockerfile:** a text file that contains instructions for building a Docker image.
- **Registry:** a central place where Docker images are stored and distributed, such as Docker Hub.

Kubernetes, on the other hand, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It uses a declarative configuration and an API-driven approach to manage clusters of nodes that run containerized applications. Some key Kubernetes terminologies are defined below.

- **Pod:** the smallest and simplest unit in the Kubernetes object model, which represents a running process in a cluster.
- **Deployment:** a Kubernetes object that manages ReplicaSets and provides declarative updates to Pods and ReplicaSets.
- **Service:** a Kubernetes object that provides network access to a set of Pods.

Overall, Docker and Kubernetes are complementary technologies that enable developers and DevOps teams to create, deploy, and manage containerized applications at scale.

B. Advantages and Disadvantages of Docker Images

Advantages of using Docker images:

- **Portability:** Docker images are portable and can run on any host machine with Docker installed, making it easy to deploy applications across different environments.
- **Isolation:** Docker images provide an isolated environment for applications, ensuring that they run consistently regardless of the host machine's configuration.
- **Efficiency:** Docker images use a layered architecture, which means that each layer can be cached and reused, resulting in faster build times and reduced storage requirements.

Disadvantages of using Docker images:

- **Complexity:** Docker images can be complex to set up and manage, especially when dealing with large-scale deployments and multiple images.
- **Security:** Docker images can introduce security risks if not properly configured and secured, such as exposing sensitive data or vulnerabilities in the underlying software.
- **Debugging:** Debugging issues with Docker images can be challenging, particularly when dealing with complex applications or multiple layers.

C. Deploying Application

The application was deployed using two methods. The first method did not utilize the YAML files and was deployed using the following steps:

1. Build the application to generate the WAR file using the `mvn package` command.
2. Create a Dockerfile containing the steps to create the Docker image.
3. Build the Docker image using the command `docker build -t gcr.io/<Project-ID>/binarycalculator .`
4. Push the Docker image to the Container registry using the command `docker push gcr.io/<Project-ID>/binarycalculator`

5. Deploy the image using GKE by running the command `kubectl create deployment binarycalculator-deployment --image gcr.io/<Project-ID>/binarycalculator --port=8080`
6. Assign an IP to the deployment by running the command `kubectl expose deployment binarycalculator-deployment --type=LoadBalancer --name=binarycalculator-service`
7. Access the application using the IP address obtained in the previous step and port 8080.

The second method used to deploy the application used YAML files while using the following steps:

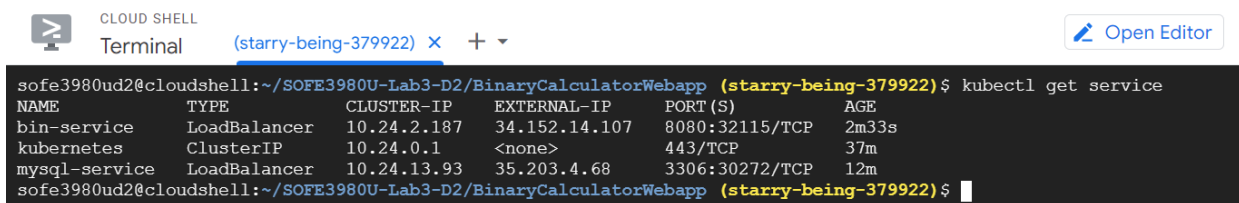
1. Create the YAML configuration files for the deployment and service.
2. Use the `kubectl create -f <input>` command to create the deployment and service in the Kubernetes cluster, using the YAML configuration files as input.
3. Create a load balancer service using the service.yaml file by running the command `kubectl create -f service.yaml`
4. Monitor the status of the deployment and service using the `kubectl get` command.

Source Code: Deployment YAML File

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bin-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bin
  template:
    metadata:
      labels:
        app: bin
    spec:
      containers:
        - image: gcr.io/starry-being-379922/binarycalculator
          name: bin
          ports:
            - containerPort: 8080
              name: bin
```

Source Code: Service YAML File

```
apiVersion: v1
kind: Service
metadata:
  name: bin-service
spec:
  type: LoadBalancer
  ports:
    - port: 8080
  selector:
    app: bin
```



A screenshot of a Cloud Shell terminal window. The terminal shows the command `kubectl get service` and its output, which is a table of services in the cluster. The output table has columns for NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. The services listed are bin-service, kubernetes, and mysql-service. The bin-service is a LoadBalancer with an external IP of 34.152.14.107 and port 8080. The kubernetes service is a ClusterIP with port 443. The mysql-service is a LoadBalancer with an external IP of 35.203.4.68 and port 3306. The terminal prompt is `sofe3980ud2@cloudshell:~/SOFE3980U-Lab3-D2/BinaryCalculatorWebapp (starry-being-379922)$`.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
bin-service	LoadBalancer	10.24.2.187	34.152.14.107	8080:32115/TCP	2m33s
kubernetes	ClusterIP	10.24.0.1	<none>	443/TCP	37m
mysql-service	LoadBalancer	10.24.13.93	35.203.4.68	3306:30272/TCP	12m

The IP address for the exposed MySQL service can be found below:

35.203.4.68

Click [here](#) to be routed to the binary web application. If the hyperlink is not working copy and paste the following link into your browser:

<http://34.152.14.107:8080/>

III. Conclusion

In conclusion, this lab exercise provided a hands-on experience for deploying a Maven web application using Docker and Kubernetes on Google Kubernetes Engine (GKE). The objective of the lab was to get familiar with Docker and Kubernetes, use Google Cloud Platform, and deploy the Maven web app on GKE. The lab covered the essential steps required to set up a GCP account, familiarize oneself with the dashboard, project(s), console, and editor, and enable GKE by creating a three-nodes cluster.

One of the essential concepts in deploying applications with Kubernetes is the use of YAML files to define resources such as deployments, services, and pods. While this lab did not explicitly cover YAML file creation, it is crucial to have a solid understanding of this topic to take full advantage of Kubernetes' capabilities.

The lab exercise also demonstrated how to deploy a pre-existing MySQL image using Kubernetes and access its logs and database through the CLI. By following the instructions and commands provided, learners can gain practical experience with cloud computing and containerization technologies, which are becoming increasingly relevant in the modern software development industry.

Overall, this lab exercise was a valuable opportunity for learners to practice deploying a web application using Docker and Kubernetes on a cloud platform, which can help prepare them for real-world development scenarios. It's recommended that learners continue to explore and experiment with cloud technologies, as they play a significant role in the future of software development.

