Project Milestone 1 - IaaS: Virtualization and Containerization

Esam Uddin - 100711116

2/2/2022

Given Lab 1 Repository: https://github.com/goergedaoud/SOFE4630U-tut1.git

Group Project Repository:

https://github.com/esam191/Intelligent-Transportation-System

Learning Objectives:

- Understand containers.
- Build containers from images.
- Build customized images.
- Build and run multi-container Docker applications.
- Get familiar with Google Cloud Platform (GCP).
- Run docker images on GCP.
- Get familiar with how to use Kubernetes on GCP.

Procedure:

1. **Watch the following introductory video that describes what is the difference between containerization and virtualization:**

   https://youtu.be/0qotVMX-J5s

2. **Install Docker on your local machine.**

3. **Follow the following video to create containers from images:**

   https://youtu.be/FzwIs2jMESM

4. **Answer the following questions:**

5. **What are docker images, container, and registry?**

Docker images are essentially templates used to build containers while docker containers are runnable instances of docker images. A registry is just a storage and delivery system that stores the docker images that we created.

6. **List the Docker commands used in the video with a brief description for each command and option.**

   docker build -t hello-world:1.0 . : used to build a docker image

   docker images: displays list of images created

   docker run hello-world:1.0 : creates new containers and runs it

   docker ps : lists the containers running currently

   docker ps -a : lists all containers

   docker logs [container name] : shows the logs of the specific container

7. **At the end of the video, there are two running containers, what commands can be used to stop and delete those two containers?**

command to stop container: docker container stop [container id]

command to remove container: docker container rm [container id]

8. **Prepare a video showing the container(s) created on your machine, displaying their logs, stopping them, and then deleting them. (Note: the JDK version must match that installed in your machine and used to compile the java code. If you have a problem compiled it can download it from the repository from the path: "/v1/out/production/HelloWorldDocker/Main.class" and use OpenJDK:14 in your Dockerfile).**

Video Link:

https://drive.google.com/drive/folders/1y1nNh2LI36MY3u8DzK7ZhPvgpFSgzWV_?usp=sharing

9. **Follow the following video to build a multi-container Docker application:**

   https://youtu.be/_m9JYAvFB8s

7. **Answer the following questions:**

8. **What's a multi-container Docker application?**

A multi-container application allows us to run multiple processes with a process manager and they are required because each container should be responsible for one thing and it should do that well. Therefore, multi-container apps help with scaling APIs differently than databases, you can also update versions.

9. **How are these containers communicated together?**

Networking allows these containers to communicate with one another. As long as the containers are on the same network they can talk to each other, else they can't. That's why we simply create a network and attach the containers to it.

10. **What command can be used to stop the Docker application and delete its images?**

command to stop image: docker container stop [container id]

command to remove image: docker container rm [container id]

11. **List the new docker commands used in the video with a brief description for each command and option.**

docker pull mysql : get latest version of mysql

mvn clean install : creates the .war file in the target directory

docker rm -f [name] : another way to remove container without stopping it (force remove)

sudo docker run --name app -d -p 8080:8080 my-web-app:1.0 : creates new app container with the specified port number (binds the port number with the host machine) docker run --name app-db -d -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=myDB mysql : creates new application container for the mySQL database

docker network create [network name] : creates new bridge network with specified name

docker network ls : displays list of current networks

docker network connect [network name] [container name]: connects specified container with the specified network

docker run --name app -d -p 8080:8080 --network=app-network my-web-app:1.0 : connects the app container with the database container through the network and runs the container

docker-compose up : brings application up by creating and starting the containers

12. **Prepare a video showing the created application, run the webapp, stop the application and delete the application containers. (Note: if you have a problem generating the war file, you can download it from the repository from the path: "/v2/target/MyWebApp.war").**

Video Link:

https://drive.google.com/drive/folders/1y1nNh2Ll36MY3u8DzK7ZhPvgpFSgzWV_?usp=sharing

13. **Create a free Google Cloud Account. The first two videos in the following playlist may be helpful:**

    https://youtu.be/4D3X6Xl5c_Y?list=PLlivdWyY5sqKh1gDR0WpP9iIOY00IE0xL

10. **Watch the following videos to get familiar with Kubernetes:**

    https://youtu.be/RI5M1CzgEH4

11. **Follow the following video to deploy dockers containers (valid until the shell session is expired) on GCP or by using Kubernetes (until you change it):**

    https://youtu.be/vIKy3pDz3jM

    (note: you can use /v3/index.html from the).

12. **Prepare a video showing how the container is deployed using Docker and Kubernetes in GCP.**

Video Link:

https://drive.google.com/drive/folders/1y1nNh2LI36MY3u8DzK7ZhPvgpFSgzWV_?usp=sharing

13. **List all used GCP shell commands and their description in your report.**

docker run -p 8080:80 nginx:latest : run nginx webserver on port 8080 (docker pulls latest nginx image and runs container)

docker cp index.html [container-id]:/usr/share/nginx/html/ : copying the index.html file on the specified container id

docker commit [container-id] cad/web:version1 : building new image with updated changes (commit)

docker tag cad/web:version1 us.gcr.io/youtube-demo-255723/cad-site:version1: tag image with the host name, project id, and repository name

docker push us.gcr.io/youtube-demo-255723/cad-site:version1: push newly created image with the same information as above

gcloud config set project youtube-demo-255723 : configuring project name

gcloud config set compute/zone us-central1-a : configuring zone (optional)

gcloud container clusters create gk-cluster --num-nodes=1 : creates GKE cluster with number of nodes

gcloud container clusters get-credentials gk-cluster : creating GKE cluster with the given credentials (configures kubectl to use cluster created) kubectl create deployment web-server --image=us.gcr.io/youtube-demo-255723/cad-site:version1 : deploys an application to the cluster

kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 80 : exposing the deployment with given port number

kubectl get pods : view the pods currents running

kubectl get service : inspecting hello-server service

14. **Prepare a Kubernetes YML (or YAML) file to load the webApp used in steps 6:8 and deploy it using the Kubernetes engine on GCP. The file is a little different than that used by docker-compose.**

    - The hostname of all containers is the same and can be accessed by localhost, the address of the MySQL should be changed to localhost and recompiled. (Note: if you have a problem generating the war file, you can download it from the repository from the path "/KGS/target/MyWebApp.war").
    - Create a new image using the new war file and push it to Google Container Registry.
    - Follow the comments and fill the missing lines in the "/webApp.yml" file.
    - Apply the YML file into Kubernetes and run the server (what is the appropriate Cloud shell command?).

15. **Prepare another video describing the YML file and showing how it's deployed on GCP.**

Video Link:

https://drive.google.com/drive/folders/1y1nNh2Ll36MY3u8DzK7ZhPvgpFSgzWV_?usp=sharing

Answer the following questions:

16. **What is Kubernetes' pod, service, node, and deployment?**

A node is a worker virtual machine in accordance with the cluster used. A pod is a group of one or more containers that share storage resources. The nodes contain services that are required in order to run Pods. Kubernetes places containers into Pods that can then be run on Nodes (pods are matched to nodes). Kubernetes opens a random port for each service on the local node. Deployment helps Kubernetes create instances of pods that store containerized applications. Important thing to note is that they can scale several replica pods as well.

17. What's meant by replicas?

Replicas are copies of given pods that are running. Kubernetes runs several instances of a pod while a certain number of pods are held constant.

18. What are the types of Kubernetes' services? What is the purpose of each?

The types of Kubernetes' services include ClusterIP, service is accessible only within the cluster, LoadBalances, utilizes google cloud's load balancer (assuming cloud provider is GCP), NodePort, has a static port on each Node's IP.

19. Upload your report, used files, and videos (links) into your repository.

GitHub Repository Link:

https://github.com/esam191/Intelligent-Transportation-System