SOFE 4630U: Cloud Computing

Dr. Mohamed El-darieby

Date: February 3rd, 2022

Project Milestone #1 - IaaS: Virtualization and Containerization

Github:[Yusuf-shaik/Cloud-Lab-1 (github.com)]

Videos: [Lab Videos]

COURSE GROUP #1	
Shanjay Kailayanathan	100624670
Jana Kanagalingam	100603975
Tolu Elebute	100724471
Yusuf Shaik	100655451
Yakhneshan Sivaperuman	100703400

What are docker image, container, and registry?

Container: executable portable, lightweight, and isolated package that allows users to package their applications and dependencies in a manner that makes it easy to share. For a container to exist, it needs to run an image.

Image: template containing instructions that is used to create and run a docker container. It contains run instructions, libraries, and dependencies that the application relies on. An image can run multiple containers.

Registry: It's a server side application that contains multiple docker images for commonly used applications. Popular images can be pulled to integrate into a new container. A registry stores and allows the distribution of docker images.

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

Dockerfile commands:

- FROM: Sets the base image and initializes a new build stage
- RUN: Create a new app directory for your application files
- COPY: Copies the app files from your host machine to image file system
- WORKDIR: Sets the directory for executing future commands
- CMD: Runs the main class when your container starts

Docker terminal commands:

Docker build -t <image_name>:<image_tag> <path>

This builds the docker image. *-t* allows users to specify the name of the image, and the image tag. <path> is the path to the dockerfile. E.g:

docker build -t hello-world:1.0 .

"." Is used here since the dockerfile is in the current directory.

Docker images: Lists the current images along with the image id, time it was created, and the image size.

Docker run <image_name>:<image_tag>: Instantiate an image and run container based on image. Running with *-d* flag allows container to run in the background

Docker ps: Lists running containers along with their info

Docker ps -a: Lists all containers including running/stopped along with their info

Docker logs <container id>: Shows the output of what is occurring in the docker container specified

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

Stop container: docker stop <container id>

Remove container: docker rm (-f) <container id>. -f is used to force remove the container, not mandatory

What's a multi-container Docker application?

A multi container docker application is two or more containers that are used together to complete an application. E.g: web app and database. These two containers need to communicate over a network in order for the application to work. Hence the name, multi container docker application.

How these containers are communicated together?

The containers communicate by using network protocols. Important note is that for containers to communicate, they need to be on the same network. We use bridge networks to communicate securely between two docker containers. Commands are:

Create network: "docker network create <network name>"

Connect container to network: "docker network connect <network name> <app name>"

Run app on network: docker run –name app -d -p 8080:8080 –network=app-network my-web-app:1.0

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

Stop docker container: docker stop containerid

Remove docker container: docker rm containerid

Docker image rm <imagename> was used to delete the image

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

- ***docker run --name app -d -p 8080:8080 my-web-app:1.0:*** creates container in detached mode on 8080 port on my-web-app:1.0 image
- ***Docker pull mysql:*** pull mysql image from docker repository
- ***Docker run –name app-db -d -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=myDB mysql:*** run app with environment variables (password for mysql)
- ***docker network create app-network:*** creating own network
- ***docker network connect app-network app-db:*** This connects the app container with db container on the same network
- ***docker compose up -d:*** This automatically creates a bridge network and attaches containers to it.

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

gcloud config set project <project name>

docker run -d -p 8080:80 nginx:latest --> run the web server container

docker ps --> list available containers

git clone https://github.com/goergedaoud/SOFE4630U-tut1.git --> clone assignment repo

docker cp index.html 338ee85cc2e8:/usr/share/nginx/html/

docker commit 338ee85cc2e8 cad/web:version 1 --> create local repo of container

docker images --> list docker images on device

docker tag cad/web:version1 us.gcr.io/cloud-lab-1-340101/cad-site:version1 --> name image before pushing to container registry

docker push us.gcr.io/cloud-lab-1-340101/cad-site:version1 --> push to container registry

gcloud config set compute/zone us-central1-a --> set google compute engine location

gcloud container clusters create gk-cluster --num-nodes=1 --> create cluster with given number of nodes gcloud container clusters get-credentials gk-cluster --> configure kubectl to use cluster that was just created

docker images --> list docker images

kubectl create deployment web-server --image=us.gcr.io/cloud-lab-1-340101/cad-site:version1 --> deploy application to cluster

kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 80 --> expose deployed application on ports

kubectl get pods --> inspect running pods

kubectl get service web-server --> inspect kubernetes service that is running

What is Kubernetes' pod, service, node, and deployment?

- Node runs the services, they are a set of worker machines that host the pods and make up a kubernetes cluster. When a service or application is run, it's placed in a pod, and each node will contain 1 or more pods. A node is either a VM or an actual machine/device. Each node contains kubelet which enables communication between the node and the kubernetes cluster management.

- Pods are the most smallest and most basic deployable objects also contain shared networking and storage. Containers are placed into pods to run on nodes. Each pod contains one or more containers that can run alone, or be a copy of other containers.

- A kubernetes service enables assignment of name and an IP address to a group of pods in a cluster that perform the same function.
- For the pods that hold an application that has been containerized, deployment tells kubernetes how to create or modify instances of them based on a given configuration.

What's meant by replicas?

A replica is a copy of another container in kubernetes. For example, if we want to ensure high availability of our database, we will create 3+ replicas of this database when deploying. These replicas are exact copies of eachother, and will ensure the functionality remains in the chance that one container fails.

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

- ClusterIP: A given cluster-internal ip address to the service makes it only reachable within the cluster.
- NodePort: This service exposes the service outside of the cluster and does this by adding a cluster-wide port on top of ClusterIP. It can be perceived as an extension of the ClusterIP service.
- LoadBalanacer: In extension to the NodePort service, LoadBalancer integrates NodePort with cloud-based load balancers. Requests are forwarded to the services within the cluster.
- ExternalName: ExternalName service maps a service to a DNS name.