SOFE 4630U
Cloud Computing

Project Milestone #1
2022/02/03

Iaas: Virtualization and Containerization

Sabesan Sivakumar(100701928),
Jerusha Macwan(100723319),
Matthew English(100704553),
Pranjal Saloni (100653360)
https://github.com/PranjalS1/CloudComputingPM.git

Describe what is the difference between containerization and virtualization

What are docker image, container, and registry?

<u>Group Answer:</u>

Docker Image: A docker image is a read only file containing a set of instructions that can be used to execute a docker container. Images contain code, libraries and other necessary tools and dependencies that another user would need to install to run the application. Docker image provides a convenient way to package up applications and preconfigured server environments

Docker Container: A docker container is a software that virtually packages applications for deployment. Containers are deployed instances built from images.

Docker Registry: A registry is a stateless scalable server side application that is used to distribute docker images. The registry can control where images are stored and how they are distributed.

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

<u>Group Answer:</u>

- docker build -t hello-world:1.0 .: Builds a docker image from Dockerfile with the name hello-world with version 1.0 in the current directory. The '.' specifies the directory
- docker run hello-world:1.0: Creates and instantiates a container from the specified image hello-world
- docker rm -f 1244e706371e: Removes the container specified by the container id in the argument
- docker ps: lists all the currently running containers
- docker ps -a: The "-a" command lists all containers, whether running or not
- docker logs app: displays the logs of the container specified in the argument, in this case 'app'
- docker images: lists all the images created with their ID and when they were created.

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

<u>Group Answer:</u>

docker stop (container ID): stops a running container docker rm -f (container ID): deletes the container; “-f” forcefully removes it.

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).

https://drive.google.com/file/d/1Lkp3_hWQCsP_wJS6IRIpr6ZlA_vJ-kkf/view?usp=sharing

Build a multi-container Docker application
What’s a multi-container Docker application?

Group Answer:

A multi-container docker application is a docker application that is capable of running multiple containers while having them communicate with each other. Multi container docker applications can be managed using docker-compose.

How are these containers communicated together?

Group Answer:

Docker containers communicate on the same network over virtual networks created by Docker. In a network, a container has an IP address, and optionally a hostname. By default, the bridge network allows container-to-container communication by ip address.

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

Group Answer:

docker stop: stops the docker application; the entire containerID is not need, just first 4 values

docker rmi -q (image ID): deletes image; “-q” deletes multiple images

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

Group Answer:

- docker pull mysql: pulls mysql images
- docker build -t: builds an image with image name, tag and version
- docker run --name app -d -p 8080:8080 --network=app-network mywebapp:1.0: runs the docker application on port 8080
- docker run --name app-db -d -e MYSQL_ROOT_PASSWORD=password MYSQL_DATABASE=myDB mysql: sets root password and database name; locally runs sql database
- docker network create app-network: creates network
- docker network connect app-network app-db: connects docker app container with the database container on app-network.

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 2:
https://drive.google.com/file/d/1DKAk1o9i-gG0WF7nLat09CNqHInl2evy/view?usp=sharing

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

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)

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

https://drive.google.com/file/d/1fLoO1Xrsf0GKNRNlcRshCycQcLCby6W1/view?usp=sharing

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

Group Answer:

- gcloud config set project projectID: sets to current project
- gcloud config set compute/zone us-central1-a: sets current directory's timezone
- gcloud services enable container.googleapis.com: enables google apis

- gcloud container clusters create gk-cluster --num-nodes=1: creates a cluster with the name “gk-cluster” and specified number of nodes
- gcloud container clusters get-credentials gk-cluster: retrieves credentials for cluster deployment
- kubectl create deployment web-server –image=us.gcr.io/projectID/cad-site:version 1: creates deployment using gcr image
- kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 80: exposes web server to port 80
- kubectl get pods: shows a list of active pods
- kubectl get service web-server: displays external IP address to access the web server

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?).

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

Video 4:

https://drive.google.com/file/d/1hQPuBTnbyabS1BDVAaJFuJ04ykhTGeqH/view?usp=sharing

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

<u>Group Answer:</u>

Kubernetes’ pod: Pods are the smallest, most basic deployable objects in Kubernetes. A Pod represents a single instance of a running process in your cluster. Pods contain one or more containers, such as Docker containers.

Kubernetes service: It is a concept that specifies a rational collection of pods and a protocol for accessing them. This model can also be referred to as micro services.

Kubernetes node: A node is a virtual or physical machine depending on the cluster that runs containers that have been placed into pods. The nodes are controlled by a control plane and contain services required to run pods.

Kubernetes deployment: Is what updates the Pods to ensure that all the pods are loosely coupled and that if one pod were to fail the application can still run efficiently since they are replicas in multiple pods.

What's meant by replicas?

Group Answer:

Replicas are the process that is run to maintain a specified amount of pod instances to ensure reliability of the application and to prevent loss of access if failure occurs.

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

Group Answer:

There are four types of Kubernetes services:

ClusterIP: Exposes the Service on a cluster-internal IP. This is the default ServiceType and choosing this value makes the Service only reachable from within the cluster.

NodePort: Node ports are the open ports on every cluster node. Traffic is forwarded from NodePort to the service.

LoadBalancer: This type of service involves a client submitting a request to a network load balancer's Ip address.

ExternalName: A special case of service that does not have selectors. It does not define any ports or endpoints. Rather, it serves as a way to return an alias to an external service residing outside the cluster.

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