Cloud computing
Iaas: Virtualization and Containerization

Matthew English(100704553)

What are docker image, container, and registry?

Docker Image: A docker image is a file 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 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.

- docker build -t imagename directory : Builds the container
- docker images : Displays all images on the device
- docker run -d imagename: Runs the container in background
- docker ps: Displays the currently running containers
- docker logs: Prints logs of container running in background

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

- docker stop containerID : Stops container
- docker rm containerID: Deletes container
- docker-compose down: stops container when docker-compose up was used

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).

VIDEO1:
https://drive.google.com/file/d/138dQLiTdT2_Ga1JUkOhkLRWO9BfyJJak/view?usp=sharing

What’s a multi-container Docker application?

A multi-container docker application is just a docker application that is capable of running multiple containers while having them communicate with each other.

How are these containers communicated together?

Docker uses a virtual network to allow the containers to communicate. 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?

-docker rm -f appname

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

- docker pull mysql : created a mysql image
- docker run –name app -d -p 8080:8080 –network=app-network mywebapp:1.0 : ran docker container locally
- docker run –name app-db -d -e MYSQL_ROOT_PASSWORD=password MYSQL_DATABASE=myDB mysql : ran sql database locally
- docker network create app-network : Created app network
- docker network connect app-network app-db: connects database to docker network
- docker-compose up: aggregates the output of each container and all containers will be stopped when the command exits

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”).

VIDEO2:
https://drive.google.com/file/d/1qcdnWIhxwTonnWDCQsgyIV9XLb64KXDo/view?usp=sharing

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

VIDEO3:

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

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

- gcloud config set project projectID : sets current working directory

- gcloud config set compute/zone us-central1-a: Sets current working directory time zone
- gcloud container clusters create gk-cluster –num-nodes=1: creates a cluster with 1 node
- gcloud container clusters get-credentials gk-cluster: gets credentials to deploy the cluster
- 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 ports 80 and enables the website to be accessed
- kubectl get pods: Displaying active pods
- kubectl get service web-server: Displays information including external IP address used to access the web server.

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

VIDEO4:

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

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

Pod: Pods are groups of one or more containers with shared storage and network resources. It also includes specifications for how the run contains.

Service: A service is an abstraction which defines a logical set of pods and a policy by which to access them.

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.

Deployment: A deployment runs replicas of your application to replace any instances that fail or become unresponsive to ensure one or more instances of your application can be available for user requests. Deployments use a pod template with specifications to determine how each pod should look like and what applications should run inside its containers.

What's meant by replicas?

A replica is a 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?

ClusterIP: default type of service used to expose a service on an IP address internally to the cluster.

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

ExternalName: Map a service to a DNS name not to a typical selector. It maps the service to the contents of the externalName field by returning a CNAME record with its value.

LoadBalancer: Extension of Nodepart service. External load balancer routes are created where traffic is directed at the backend pods where the cloud provider decides how it is load balanced.