Hamza Farhat Ali

100657374

Note: I am talking in all my videos, please listen with audio on

All the files and videos for individual work is on this group repository link that can be found in the main document hand in as well:

What are docker image, container, and registry?

Docker image

- *A docker image* is a file used to execute code in a Docker container.
- This docker image acts as a template of instructions to build the container
- A docker image contains application code, libraries, tools, dependencies, and other files to run the application on the container.
- *A docker container* are the instance from the image that is deployed. This is the Operating System virtualization.
- This software would run on a cloud resource, on hardware elsewhere.
- *A docker registry* is used to push the docker images onto to be used as a container.
- The registry holds the storage and content delivery system.

- Docker is the software package that allows one container to be built with all dependencies needed to run an application.
- Instead of setting up the app server, and all other technologies beforehand to run an application.

Source: https://docs.docker.com/registry/introduction/

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

- *docker build -t <tag> <path>*

- This command is used to build the docker image with a tag, acts like a name tag
- There is also a path to create the build in that follow's in command

- *docker run [-d] <image>*

- This command runs the Docker container which means the code in the Docker image gets executed
- -d option is the running the container in detached mode which means the container will run in the background and not display anything in the console

- *docker ps [-a]*

- This command displays all currently running containers
- -a option shows all Docker containers including containers that already ran and terminated

- *docker images*

- This command displays all Docker images currently on the system. It displays information about the images such as the repository, tag, imageid, creation date and size.

- *docker logs <Container ID>*

- This command will show the console log of the container

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 <container name/ID>*

- *docker rm <Container name/ID>*

What's a multi-container Docker application?

- A multi-container application is usually used to host applications using the microservice architecture.
- This allows for multiple containers to communicate with each other instead of isolation
- An example are database shared between containers.
- In this situation, there will be two containers. One for the web application and another container for the database.

How these containers are communicated together?

- If 2 containers are on the same network they can work together
- The containers communicate with each other through a bridge network.
- This virtual bridge by default connects to all containers on the cloud instance.
- These containers are assigned IP addresses to communicate with each other.

Source: https://docs.docker.com/get-started/07_multi_container/

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

docker-compose down -rmi all

RMI refers to registry, deletes content

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

- *docker pull <image name>*

- Pulls an image from a registry

- *docker run -name <Name> -d -e <var_name=var_value> -p <host machine port:container port>*

- This command runs the Docker container
- -name option specifies a name of the container to save as
- -e option sets the environmental variables of the container
- -p option publishes the containers port to the host

- *docker network create <name>*

  - This command creates a bridge network for the containers to communicate on

- *docker network connect <network name> <container name>*

  - This command creates a container to a network with name to save as

- *docker network ls*

  - This command lists all networks the docker daemon knows about in the current duire

- *docker-compose up*

  - This command initializes the containers defined in the docker-compose.yml file

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

- *gcloud config set project <project name>*

  - This command sets the project being used in the cloud

- *docker cp <filename> <container-id>:<container-path>*

  - Copies files from the local filesystem to the container

- *docker commit <container id> <img-name>*

  - This command creates a new image from the specified container
  - Used for when creating images when changes to the container has been made

- *docker tag <source-img-name> <img-name>*

  - This command creates another image that refers to a source image

- *docker push <img-name>*

  - This command pushes the image to the container registry

- *gcloud config set compute/zone <region>*

  - This command changes the region of the project to the region specified

- *gcloud container clusters create <cluster-name> --num-nodes=<num-nodes>*

- This command creates a cluster for containers with the specified name
- –num-nodes option just lets the user specify the number of nodes to be created in the cluster

- *gcloud container clusters get-credentials <cluster-name>*

- This command retrieves the credentials for a running cluster
- This updates the kubeconfig file with the credentials of the cluster

- *kubectl create deployment <deployment-name> --image=<img-name>*

- This command creates a deployment with the specified name and image

- *kubectl expose deployment <deployment-name> --type <service> --port <port> --target-port <port>*

- This command exposes the deployment with the specified name
- --type option allows the user to specify what type of kubernetes service like for example a LoadBalancer

- *kubectl get pods*

- This command lists all pods

- *kubectl get service <deployment-name>*

- This command lists the service with the specified name. Sort of like ls

Apply the YML file into Kubernetes and run the server (what is the appropriate Cloud shell command?).

- *kubectl apply -f webApp.yml*

The group helped me understand this.

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

- Pod - A group of containers with shared resources which includes network, storage and scripts for compiling the containers
- Service - Where an application is running, this is the network service
- Node - Location for pod instance to be running

Deployment - This provides the updates when describing the code in production.

- This desired state in a deployment changes the current cloud state in accordance to deployment .
- It is used to update the pods, replicas and whatever other dependencies in kubernetes

Source: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

What's meant by replicas?

- A replica is a duplication of identical pods.
- This improves availability of the service
- A replica is defined with replicaSets as well to control for a specified number of pod replicas running together.

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

- *ClusterIP* – This is the default type of service that exposes the service on a cluster-internal IP.
- This provides network connectivity between containers
- *NodePort* – Exposes a service via a static port via each node's IP
- This routes incoming traffic from NodePort to user's service, even when on a different node
- *LoadBalancer* – Exposes the service via a load balancer.
- These are automatically created, and had integration with cloud based balancers
- This distributed the tension on one server and evenly spreads resources with multiple nodes
- *ExternalName* – Similar to other services but can be accessed with DNS name instead of the clusterIP address
- This Extername help's serve as a way to return an alias to an external service residing outside the cluster, for security purposes.

Source: https://kubernetes.io/docs/concepts/services-networking/service/