Khalil Zayed, 100672228

Project Milestone - IaaS: Khalil Zayed - 100672228

Docker image = set of instruction or template that is executed in a container
Docker container = running instance of an image, the set of instructions executed
Docker registry = registry where you can store and redistribute your docker image

Dockerfile:

```
# this command tells docker to load an OpenJDK 11 image as the base image
# if it does not exist on machine, it will download it from docker hub and save for next builds
FROM openjdk:11

# this command will create a new app directory in the image, not the local machine
RUN mkdir /app

# Copy the app files from host machine to image filesystem
# the first address is the address of the app files on the host machine and the second one is the
# address of the folder to copy the files into in the image
COPY out/production/HelloWorldDocker/ /app

# Set the working directory for executing future commands in the image to the newly created
# app folder
WORKDIR /app

# Run the Main class that we copied from the local machine into the image filesystem earlier
# this should print whatever in the main class which is “hello world!”
CMD java Main
```

Docker stop:
According to docker documentation, “docker stop [options] container_name” will kill the container. If time is specified in options, it will wait the amount of seconds before killing it.

Docker remove:
According to docker documentation, “docker rm [options] container” will remove the container. To force remove a running container, you may specify “--force” as an option in the command.

Video:
https://drive.google.com/file/d/1Jtv-u19TgOvIIjM-_-jaHS3iYifdIKu5/view?usp=sharing

Multi-container Docker applications
Multi-container docker applications allow for databases to be integrated with the front-end while residing in a separate environment. This way you can manage and update them separately, which makes things simpler in production.

Khalil Zayed, 100672228

Communication between containers
Containers are able to communicate together through a bridged network that docker creates by default and connects all containers locally. Additionally, the user may create their own network and allocate it to the containers to allow them to communicate with each other through it.

Docker stop and remove images
To stop a docker application, use the commands above to stop and remove a container. To delete an image, use “docker image rm [options] image_name”. This will delete the image from the docker.

Command	**Description**
docker build -t my-web-app:1.0 .	This command will build the image using the Dockerfile in the directory and giving it the name my-web-app and version 1.0
docker run --name app -d -p 8080:8080 my-web-app:1.0	This command will start a new container using the my-web-app:1.0 image, giving it the name app and exposing the host port of 8080 to the container port of 8080
docker network create app-network	Create a new network in the docker and give it the app-network name
docker network ls	Show the current list of networks in docker
docker network connect app-network app-db	Connect the app-db container to the app-network network
docker-compose up -d	Start docker using the instructions in the .yaml file in the directory. This will execute the instructions in the file and start all the containers and the application

Video:
https://drive.google.com/file/d/1M0IRcHYm3gcbJNJ-2MYdp-q7PFPeCm7I/view?usp=sharing

Khalil Zayed, 100672228

Kubernetes:

Video:
https://drive.google.com/file/d/1fmV9zkMIDkkekYLI2Cfg-4AIpw8zbY1B/view?usp=sharing

Command	**Description**
docker cp index.html <id>:/usr/share/nginx/html/	This command copies index.html from the local machine to the container directory at id.
docker commit <id> <name>	Commits the image in the container at id to the container registry
docker tag <name> <registry_link>	Creates a tag for the image in container registry
docker push <registry_link>	Pushes the image to the container registry
gcloud config set compute/zone us-central1-a	This command sets the compute zone for the cloud instance to US central
gcloud container clusters create gk-cluster --num-nodes=1	Creates a new cluster for running containers with 1 node
gcloud container clusters get-credentials gk-cluster	This command gets the necessary credentials and saves it locally so that Kubernetes engine can point to it later on.
kubectl create deployment <name> --image=<registry_link>	Creates a new deployment using the image in the container registry
kubectl expose deployment <name> --type LoadBalancer --port 80 --target-port 80	Creates a service object that exposes the deployment using port 80
kubectl get pods	Lists all pods running in Kubernetes
kubectl get service <name>	Gets information about the specified service

Video of YML:
https://drive.google.com/file/d/154MIK42UkzUR1zNPIu2Cgst0LRXkfqCo/view?usp=sharing

Pods
Pods represent the smallest unit of deployable components in Kubernetes. They may consist of one or more containers with shared resources. They come with instructions on how to run the containers.

Khalil Zayed, 100672228

Service
In Kubernetes, an application that is running on several pods may be exposed to other components in a network. Pods usually have their own IP addresses and are able to communicate with other services on a network.

Nodes
A node is an object that resides virtually or physically in a cluster. Kubernetes uses nodes to run pods with containers and manage their workload.

Deployment
Deployments are used to control states in Pods or ReplicaSets. A state is declared in a deployment and the deployment controller will update the current state with the desired state at a steady rate.

ReplicaSet
A ReplicaSet keeps track of a set of replica Pods currently deployed. It is used to ensure availability of replica Pods for any reason. A ReplicaSet contains instructions to identify the required Pods, the number of replicas and the pod template used to create new pods. This allows it to create and delete replicas as needed.

Types of Kubernetes services
There are four types of services:
- **Cluster IP**: the service will use the same IP as the cluster. This makes it only reachable within the cluster itself.
- **NodePort**: the service will have the nodes use the same port in the network. A ClusterIP will be created for the nodes with the node port. This way it can be accessed from outside the cluster using <NodeIP>:<NodePort>.
- **LoadBalancer**: specifying this will provide an external load balancer for your service. This means that the ClusterIP and NodePort will be created and have the external load balancer route to it.
- **ExternalName**: use this to specify a DNS name for your service instead of a default name.