Project Milestone 1: IaaS Virtualization and Containerization

Raveenth Maheswaran   100704540

Single Docker Container

What are docker image, container, and registry?

- Docker image is a file to execute the code within the Docker container. Technically speaking, it acts as a set of instructions to build the container like a template. In order to build the container, the image must have the application code, dependencies, tools, and libraries to make the application run.
- Docker container is the virtualized runtime environment where the applications are isolated from the underlying system. Its functionality is comparable to virtual machines; however, Docker containers provide more scalability, portability, resource-friendly, and isolation.
- The registry is the server-side application and allows you to store and distribute Docker images and repositories.

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

- docker version
  - Outputs the current docker version installed in the local machine
- docker build -t hello-world:1.0 .
  - Builds the docker image based off of the commands provided in the Dockerfile
  - -t: an option (short for tag) to provide a name and tag in 'name:tag' format
  - '.': the dot specifies the current directory
- docker images
  - Outputs all the docker images created in the local machine
- docker run -d hello-world:1.0
  - Creates a docker container and runs with the specified image
  - -d: Stands for detached, where the container will start and runs without taking up the console windows with constant messages
    - It will output the container ID
- docker ps
  - Shows the list of running containers
  - -a: Shows all containers (running and stopped)
- docker log fdfb47
  - Outputs the current docker container log messages based off the container id

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]
    - container name can be found by using docker ps -a
- docker rm [container name]
    - rm stands for remove
    - container name can be found by using docker ps -a
    - -f can be used to forcefully remove a container

Video Demonstration: https://drive.google.com/file/d/1g5KwYGcseYO-Wu8fjBT-9JfIDt3_cGN7/view?usp=sharing

Multi-Container Docker

What's a multi-container Docker application?

- A multi-container Docker application is an application which requires multiple well-communicative dockers to expand functionality and features to suit the user's needs
- It has multiple benefits such as scalability, simplistic set-up sequence, and simultaneous run-time by utilizing multiple ports.

How are these containers communicating together?

- These containers typically communicate via network bridges.

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

- docker stop
- docker rm -f

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

Database Container

- docker pull mysql
    - Pulls the latest version of a certain image or repository from the Docker Hub (registry)
    - In this case, it is mysql
- docker run –name app-db -d -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=myDB mysql
    - Creates a container and names it app-db via --name
    - -d: Stands for detached, where the container will start and runs without taking up the console windows with constant messages

- -e: Stands for environment variable, where it sets environment variables for certain functionalities
  - In this case, MYSQL_ROOT_PASSWORD and MYSQL_DATABASE are the environment variables
- mysql is the image that is going to be used in the docker container (can be any other image, as long as it is pulled via docker pull)

- docker ps
  - Shows the list of running containers
  - -a: Shows all containers (running and stopped)
- docker log app-db
  - Outputs the current docker container log messages based off the container id

Application Container

- mvn clean install
  - Rebuild war file to be placed in Docker image
- docker build -t my-web-app:1.0 .
  - Builds the docker image based off of the commands provided in the Dockerfile
  - -t: an option (short for tag) to provide a name and tag in 'name:tag' format
  - '.': the dot specifies the current directory
- docker images
  - Outputs all the docker images created in the local machine
- docker run –name app -d -p 8081:8080 my-web-app:1.0
  - Creates a container and names it app via --name
  - -d: Stands for detached, where the container will start and runs without taking up the console windows with constant messages
  - my-web-app:1.0 is the Docker image, where it will be in the Docker container to run the application
  - -p: publishes a container's port to the host
    - 8081:8080 → first port is host, second port is container port
- docker ps
  - Shows the list of running containers
  - -a: Shows all containers (running and stopped)
- docker log app
  - Outputs the current docker container log messages based off the container id
- docker rm -f app
  - Removes/deletes the docker container
  - -f: force meaning to stop the container and then remove it
- docker network create app-network
  - Creates bridges network connection called app-network
- docker network ls

    - Lists all the docker networks
- docker network connect app-network app-db
    - Connects the specified docker network to a container
        - In this case app-network is used to connect to app-db
- docker run –name app -d -p 8080:8080 –network=app-network my-web-app:1.0
    - Creates a container and names it app via --name
    - -d: Stands for detached, where the container will start and runs without taking up the console windows with constant messages
    - my-web-app:1.0 is the Docker image, where it will be in the Docker container to run the application
    - -p: publishes a container's port to the host
        - 8080:8080 → first port is host, second port is container port
- docker-compose up -d
    - Runs the dockercompose.yml file that will automatically set and start up the two containers and the network bridge

Video Demonstration: https://drive.google.com/file/d/121HBlxpN-7YzEaXhkl-YkP9OZwzgHKzt/view?usp=sharing

Kubernetes

List all used GCP shell commands and their descriptions in your report

- gcloud config set project kuberneteswebdemo
    - Set the terminal to the specified project
- docker run -d -p 8080:80 nginx:latest
    - Runs the nginx docker
- docker ps
    - Lists all containers
- vim index.html
    - Used vim to insert the index.html
    - Later found out that you can upload it through cloud console
- docker cp index.html f6ff741dc08b:/usr/share/nginx/html/
    - Copies the index.html file to the container's html location
- docker commit
    - Commits any changes within that docker container

Video Demonstration: https://drive.google.com/file/d/1DO7GAwbT-9p8Tjrtfw2wAFUHzBORuJyP/view?usp=sharing

Video Demonstration: https://drive.google.com/file/d/1KxctkChn_gwlCujSculYw1uFo8yP6snH/view?usp=sharing

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

- A Kubernetes' pod is the most basic deployable objects in Kubernetes. Within these pods contains one or more containers, and the shared storage, network resources, and a specification on how to run the containers
- A Kubernetes' service is the abstraction where a set of Pods are logically set and has a policy to access them
- A Kubernetes' node is a physical or virtual worker machine, where it runs Pods and manages by the Master
- Kubernetes' deployment is a resource object that declares updates to applications

What's meant by replicas?

- Replicas are duplicate Pods that provides extra reliability and maintainability incase a Pod fails or becomes inaccessible
- ReplicasSet is a process that runs the multiple duplicate Pods and keep the specified number of Pods constant

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

- **ClusterIP**: A service from within the cluster
- **NodePort**: A service that utilizes a static port on each node's IP
- **LoadBalancer**: A service that provides the cloud provider's load balancer
- **ExternalName**: A service to predefine an externalName field by returning a value for the CNAME record