SOFE4630 Cloud Computing (Winter 2022 - Dr. M. El-darieby)

Project Milestone-- IaaS: Virtualization and Containerization

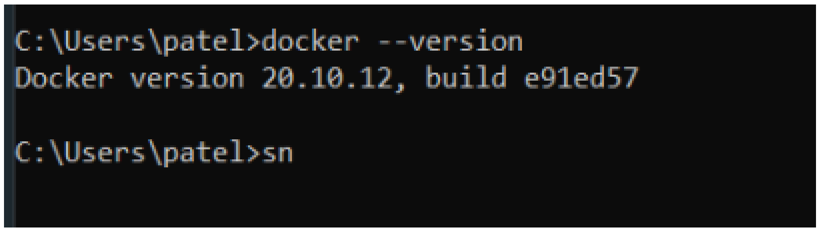
Group 5

Date: 02/03/2022
Student Names:
Alexander Pitcher (100693749)
Mitul Patel (100700131)
Clarissa Branje (100716458)
Amin Khakpour(100669547)

Activity Video 1 HelloWorld Docker:
https://drive.google.com/file/d/1BhVnkhIRf5DoOe_i0NU-bfh2oRpNIPX-/view?usp=sharing
Activity Video 2 MultiContainer Docker:
https://drive.google.com/file/d/1cZZRlUFCR-UX0x2KcvVZJyslRnlOdXZ2/view
Activity Video v3 Deploying HTML File:
https://drive.google.com/file/d/1zi5bbIlhU5fqLiBNxcm3p3lr8_4oFo-l/view?usp=sharing
Activity Video Final GKE, Cluster, MyWebApp deployment:
https://drive.google.com/file/d/132OUEGLxkURml4TrB4UHfmDSh1_RiluT/view?usp=sharing

1. Watch the following introductory video that describes what is the difference between containerization and virtualization
   Virtualization:
   - Hosts Multiple Operating system
   - Uses up more resources
   - Requires a hypervisor
   - Separates available resources (computing power, memory, storage.. etc)'

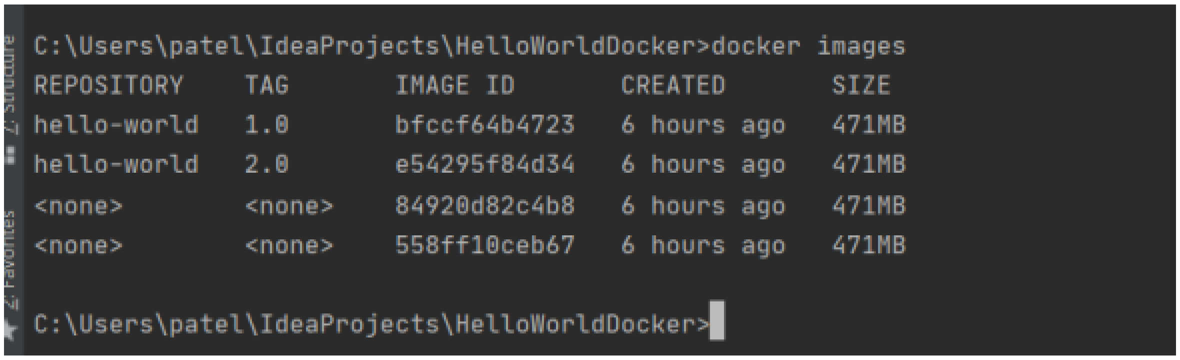
   Containerization
   - Enables deploying multiple application on same OS
   - One server or VM can run multiple container
   - Uses up less resources compared to Virtualization
2. Install Docker on your local machine



```
C:\Users\patel>docker --version
Docker version 20.10.12, build e91ed57

C:\Users\patel>sn
```

3.



```
C:\Users\patel\IdeaProjects\HelloWorldDocker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
hello-world   1.0       bfccf64b4723   6 hours ago   471MB
hello-world   2.0       e54295f84d34   6 hours ago   471MB
<none>        <none>    84920d82c4b8   6 hours ago   471MB
<none>        <none>    558ff10ceb67   6 hours ago   471MB

C:\Users\patel\IdeaProjects\HelloWorldDocker>
```

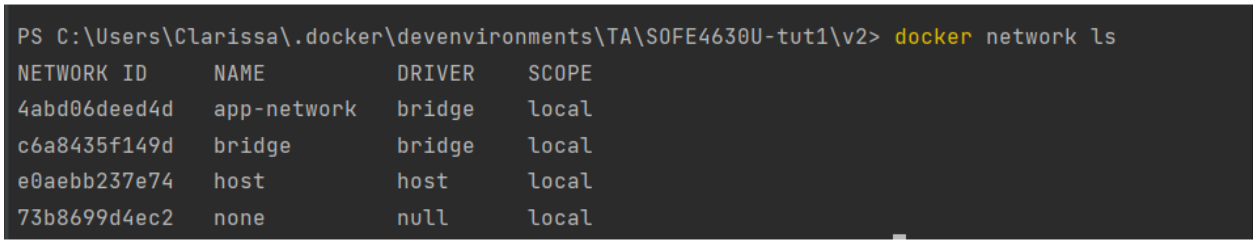
4. Answer the following questions
5. What are docker images, container, and registry?
   - Docker image: A read-only template with instructions for creating a container for use on the Docker platform.
   - Container: Standard component that allows you to package your application and its dependencies in an easy to share way. They are portable, isolates, and lightweight. Great for all different types of applications (standalone, enterprise, database, languages, etc). Cheaper and has no requirement of a guest os like a VM.
   - Registry: A content delivery and storage system that holds named Docker images in various tagged versions.

6. List the Docker commands

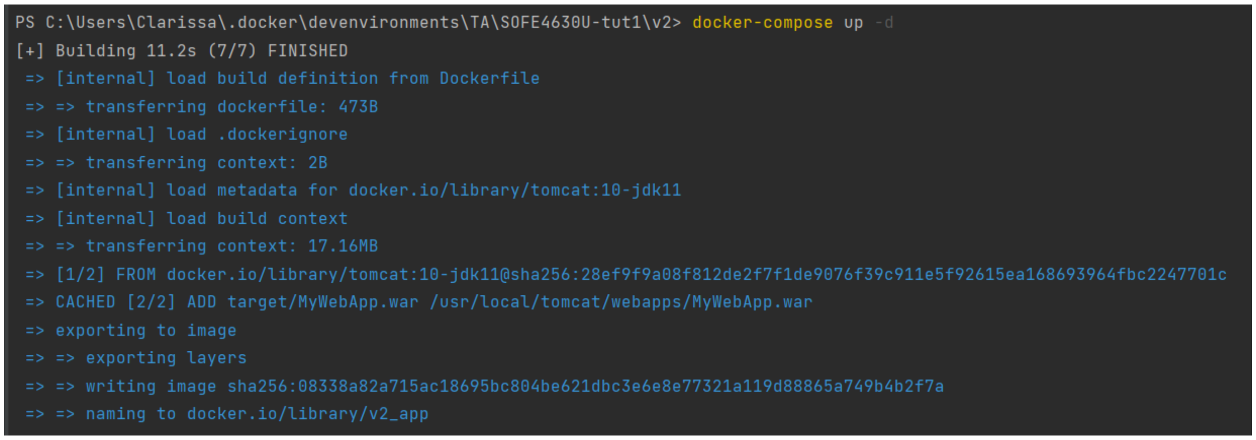
   - FROM: Choses the docker image to pull and what version of it to use
   - COPY: copies from the host application
   - RUN: Creates a new directory
   - WORKDIR: Create a directory for future commands
   - CMD: Run a program
   - Docker images => lists all the docker images
   - Docker build -t hello-world:1.0 => builds a docker image with the name hello-world and tag 1.0
   - Docker run hello-world:1.0 => runs the docker image on a container
   - Docker ps => shows all running containers
   - Docker ps -a => shows all running and non-running containers
   - Docker run -d hello-world:2.0 => runs the application in the background
   - Docker logs [id] => shows the container output of based on the id
   - Docker stop [container id] => stops the running container
   - Docker pause [id] => pauses the container
7. 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_id] => The command above stops the two containers
   - Docker image rm [container_id] => deletes the image

8. Video Link: https://drive.google.com/file/d/1BhVnkhIRf5DoOe_i0NU-bfh2oRpNlPX-/view?usp=sharing

9. Follow the following video to build a multi-container Docker application



```
PS C:\Users\Clarissa\.docker\devenvironments\TA\SOFE4630U-tut1\v2> docker network ls
NETWORK ID     NAME          DRIVER    SCOPE
4abd06deed4d   app-network   bridge    local
c6a8435f149d   bridge        bridge    local
e0aebb237e74   host          host      local
73b8699d4ec2   none          null      local
```



```
PS C:\Users\Clarissa\.docker\devenvironments\TA\SOFE4630U-tut1\v2> docker-compose up -d
[+] Building 11.2s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 473B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/tomcat:10-jdk11
 => [internal] load build context
 => => transferring context: 17.16MB
 => [1/2] FROM docker.io/library/tomcat:10-jdk11@sha256:28ef9f9a08f812de2f7f1de9076f39c911e5f92615ea168693964fbc2247701c
 => CACHED [2/2] ADD target/MyWebApp.war /usr/local/tomcat/webapps/MyWebApp.war
 => exporting to image
 => => exporting layers
 => => writing image sha256:08338a82a715ac18695bc804be621dbc3e6e8e77321a119d88865a749b4b2f7a
 => => naming to docker.io/library/v2_app
```

10. Answer the following questions:
    a. What's a multi-container Docker application?

       A multi-container Docker application utilizes more than one container for an application to run. This could be adding a database or any other application extensions. With this being hard to keep track of and installing correctly, Docker ensures each container is performing only one thing correctly.

    b. How are these containers communicated together?

       Containers can communicate by networking with each other.

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

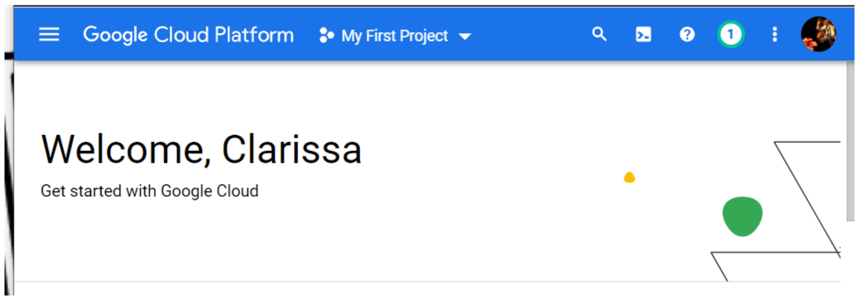
       docker image rm -f (image id) stop a running container and remove it with force

11. List the new docker commands used in the video with a brief description for each command and option
    - ADD: Add the java file I am using to my docker server
    - EXPOSE: Expose a port to docker
    - CMD: Run with tomcat

- Docker pull mysql: pulls mysql liberty from the docker.io library
- Docker run --name app-db -d -e MYSQL_ROOT_PASSWORD=password -e
- MYSQL_DATABASE=myDB mysql: Runs a Mysql container with the root password and db name along with the mysql take to use the library which was pulled earlier
- docker run --name app -d -p 8080:8080 my-web-app:1.0:Runs the docker container with port 8080, with a -p option which means publish and binds port 8080 with the external port for 8080 and d flags allows for it to run in the background.
- Docker network create: creates a network bridge which allows for containers to communicate with each other
- Docker network ls: displays Network Id, name, driver and scope
- docker network connect app-network app-db: Connects network to the app-db
- Docker run --name app -d -p 8080:8080 --network my-web-app:1.0: Runs the application container and publishes the port on 8080 and exposes the endpoint while connecting the web-app to the network.
- Docker-compose up - d: using docker compose, you can manage the build commands in a yml file called docker compose. The command runs the docker-compose file

12. Prepare a video showing the created application, run the webapp, stop the application and delete the application containers.
    https://drive.google.com/file/d/1cZZRIUFCR-UX0x2KcvVZJysIRnlOdXZ2/view?usp=sharing

13.
    a. Create a free Google Cloud Account.



    b. Prepare a video showing how the container is deployed using Docker and Kubernetes in GCP.
    https://drive.google.com/file/d/132OUEGLxkURml4TrB4UHfmDSh1_RiluT/view?usp=sharing

    c. GCP shell commands

- docker run -d -p 8080:80 nginx:latest: Set the docker to run with port 8080
- docker cp index.html CONTAINER_ID:/usr/share/nginx/html/: Find and run the file (index.html)

- docker build -t my-web-app:1.0: Builds the docker image
- docker tag my-web-app:1.0 us.gcr.io/<project_id>/webapp:version1 : Tags the image with the name tag
- us.gcr.io/<project_id>/webapp:version1: Tells google cloud about the deployment tag
- docker push my-web-app:1.0 us.gcr.io/<project_id>/webapp:version1: Commands pushs the image app on to google

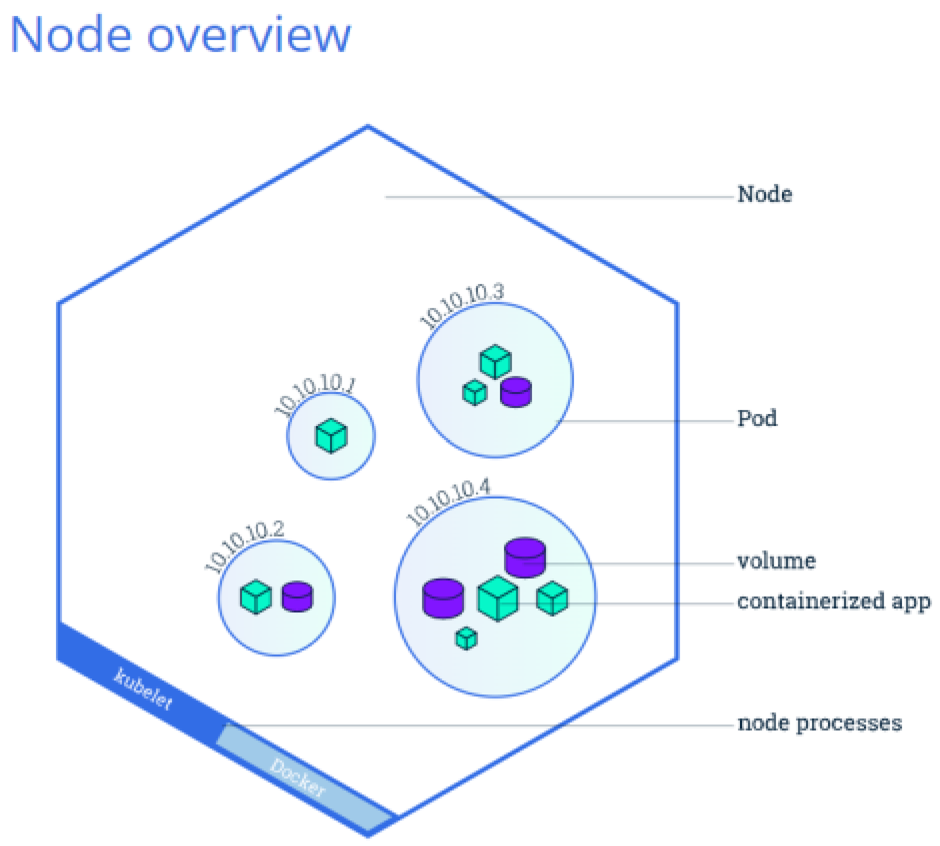
container registry

- gcloud config set project <project_id> This sets the gcloud configuration to the project id and it is used to deploy container in GKE cluster
- gcloud config set compute/zone us-central1-a: used to set the compute/zon on the GKE cluster
- gcloud container clusters create gk-cluster--num-nodes=1: Used to create a GKE cluster with the number of nodes
- gcloud container clusters get-credentials gk-cluster: Configures the kubectl to use the clusters created
- kubectl apply -f webApp.yml: This command applies the yml file on the application to build the containers
- kubectl create deployment web-server--image=us.gcr.io/<project_id>/cad-site:version1: This command is used to deploy an application to the cluster
- kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 8080: This command is used to expose the deployment cluster to a port 80 and a target port of 80
- kubectl get pods: Lists the running pods to view the application
- kubectl get service: Inspects the webapp to get the external ip address.

14. Prepare a video showing how the container is deployed using Docker and Kubernetes in GCP
    https://drive.google.com/file/d/1CvnVTX9fY2uM5dE7DgRpSpSay0_xEirg/view?usp=sharing
15. Prepare another video describing the YML file and showing how it's deployed on GCP.
    https://drive.google.com/file/d/1zi5bbIlhU5fqLiBNxcm3p3lr8_4oFo-l/view?usp=sharing

16. Answer the following questions
    a. What is Kubernetes' pod, service, node, and deployment?
        - Pod: Kubernetes' smallest and most basic deployable objects. In acluster, a Pod represents a single instance of a running process. One or more containers, such as Docker containers, are contained in pods. When many containers are running in a Pod, they are handled as a single entity that shares the Pod's resources.
        - Service: a logical abstraction for a cluster's deployed collection of pods (which all perform the same function). Because pods are ephemeral, a service assigns a name and a unique IP address to a set of pods that serve certain functions (web services, image processing, etc). (clusterIP).

- Node: a Node is a worker machine that can be virtual or physical, depending on the cluster. The control plane is in charge of each Node. A Node can have numerous pods, and the Kubernetes control plane handles pod scheduling throughout the cluster's Nodes automatically.



- Deployment: Used to tell Kubernetes how to create or modify an instance of a pod that contains a containerized application. Deployments allow you to increase the number of replica pods, allow controlled rollouts of updated code, and revert to previous deployment versions as needed.

b. What's meant by replicas?
   - a process that runs many instances of a Pod while maintaining a consistent number of Pods. Its goal is to keep a certain number of Pod instances operating in a cluster at any given moment to prevent users from losing access to their applications if one of them dies or becomes unavailable.

c. What are the types of Kubernetes' services? what is the purpose of each?
   - LoadBalancer: Send connections until the first server in the pool is full, then send new connections to the next available server. This algorithm is ideal when virtual machines are costly.

- NodePort: An open port on each node of the cluster. Kubernetes transparently forwards incoming NodePort traffic to the service, even if the application is running on a different node.
- ExternalName: A special case of a service that has no selector and uses the DNS name instead.
- Headless Services: A service with a service IP, but instead of load balancing, it returns the IP of the associated pod. This allows you to interact directly with the pod without going through a proxy.
- ClusterIp: Responsible for assigning a cluster-internal IP address to ClusterIP service. This makes the services reachable within the cluster