SOFE 4630 Winter 2022 - Cloud Computing

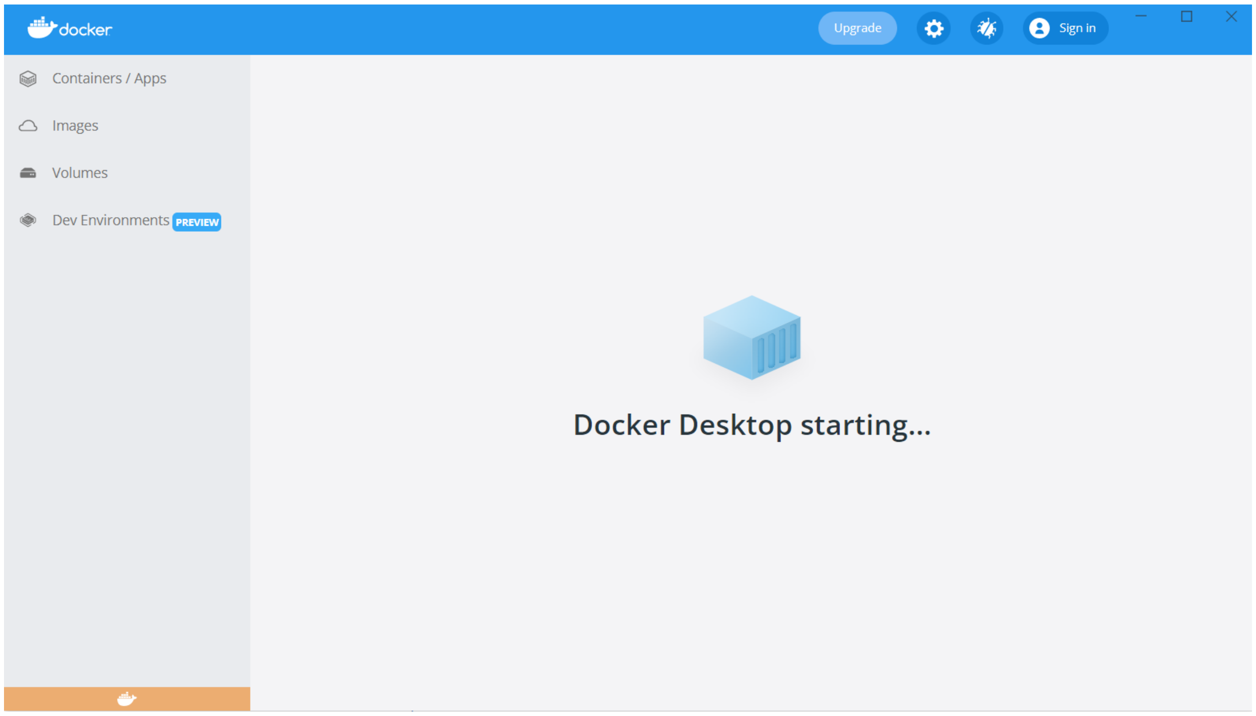
Project Milestone#1 : IaaS: Virtualization and Containerization

Course Group No: 12

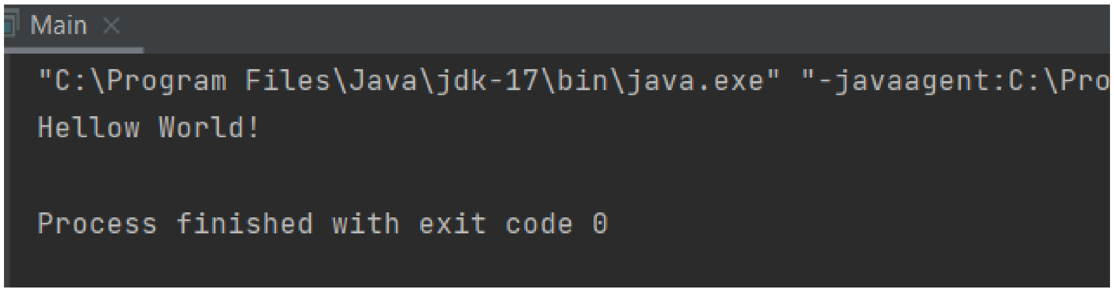
Individual Submission:

Samuel Rantetoding - 100694161

1. Watch the following introductory video that describes what is the difference between containerization and virtualization
    - Virtualization:
        - Use hypervisor
        - Guest OS, libraries, and applications need to be duplicated when scaling, therefore resulting in consuming more resources than containerization.
    - Containerization:
        - 3-time process: Create manifest, then docker image, then the container files is build
        - Use Runtime engine instead of hypervisor
        - Deploying containers is much more lightweight than virtualization as the OS guest is not concerned with containers as only the libraries and the application are scaled.
        - The containers are shared between the running processes or application, therefore more suitable for cloud-native architecture applications.
2. Install Docker on your local machine.

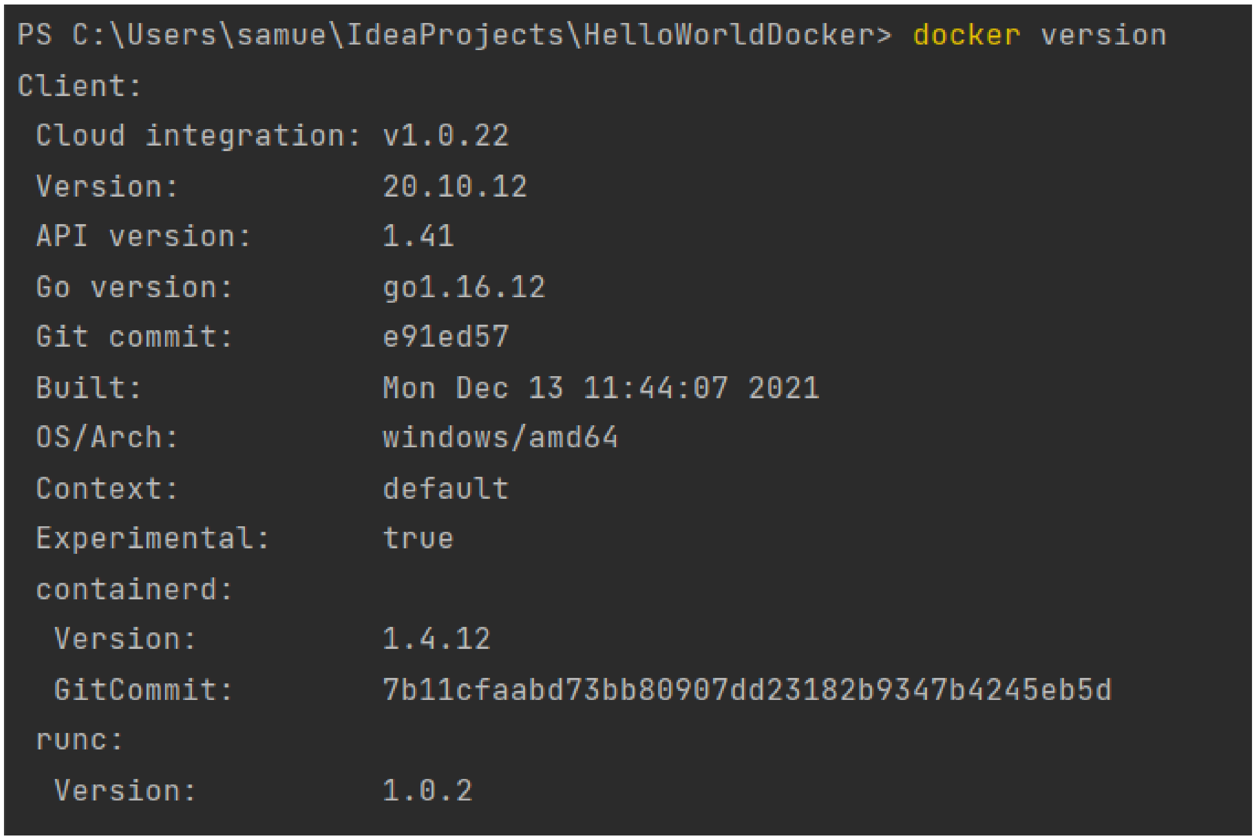


3. Follow the following video to create containers from images
    - Create Hello World Application:



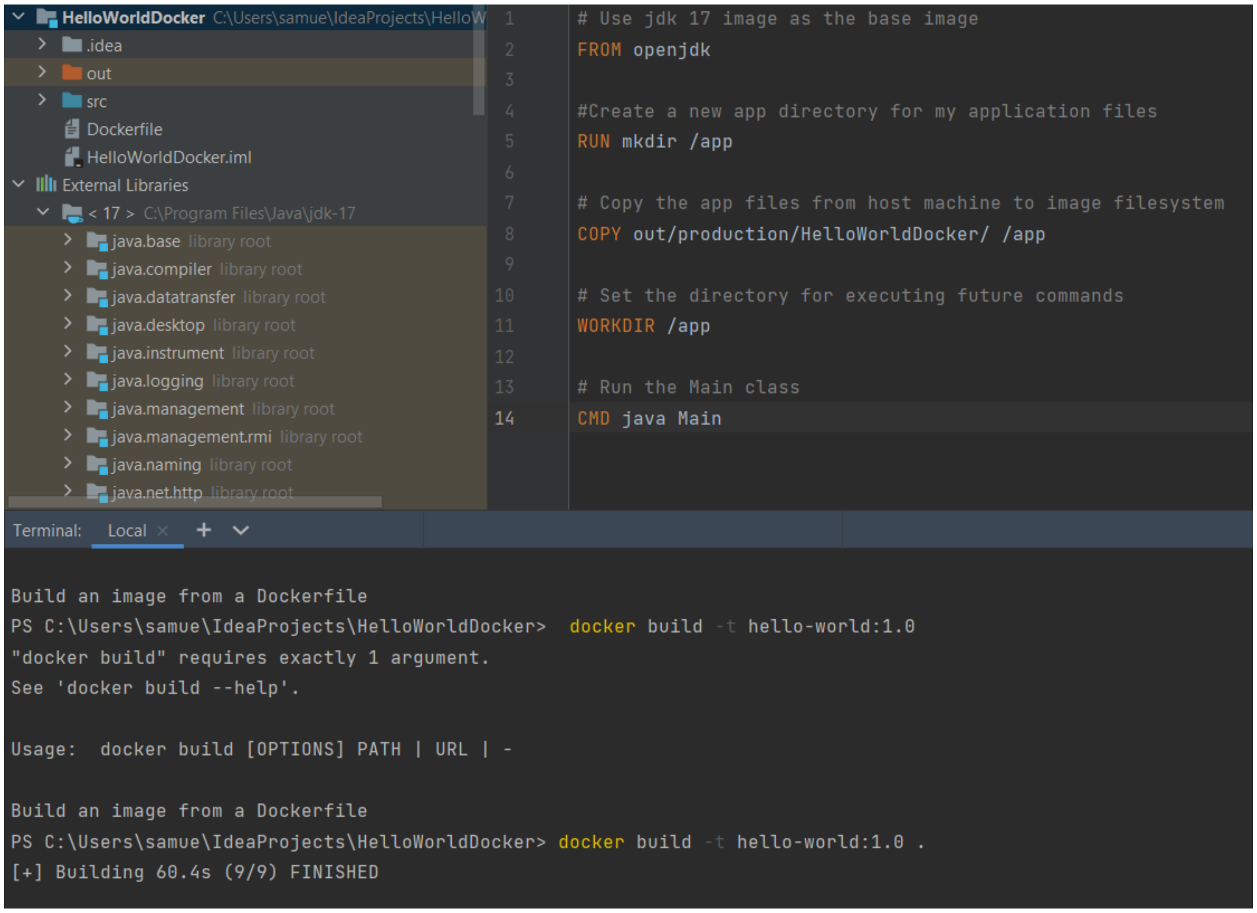
```
Main
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Pro
Hellow World!

Process finished with exit code 0
```

- Check docker version:



```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker version
Client:
 Cloud integration: v1.0.22
 Version:           20.10.12
 API version:       1.41
 Go version:        go1.16.12
 Git commit:        e91ed57
 Built:             Mon Dec 13 11:44:07 2021
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true
 containerd:
  Version:          1.4.12
  GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:          1.0.2
```

- Build Dockerfile and run the command:

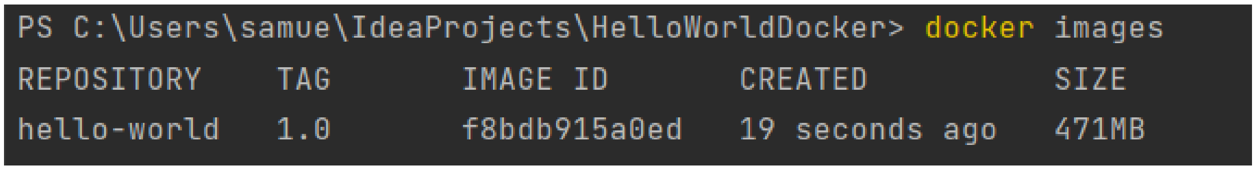


```
HelloWorldDocker C:\Users\samue\IdeaProjects\HelloW
  .idea
  out
  src
  Dockerfile
  HelloWorldDocker.iml
External Libraries
  < 17 > C:\Program Files\Java\jdk-17
    java.base library root
    java.compiler library root
    java.datatransfer library root
    java.desktop library root
    java.instrument library root
    java.logging library root
    java.management library root
    java.management.rmi library root
    java.naming library root
    java.net.http library root
```

```dockerfile
# Use jdk 17 image as the base image
FROM openjdk

#Create a new app directory for my application files
RUN mkdir /app

# Copy the app files from host machine to image filesystem
COPY out/production/HelloWorldDocker/ /app

# Set the directory for executing future commands
WORKDIR /app

# Run the Main class
CMD java Main
```

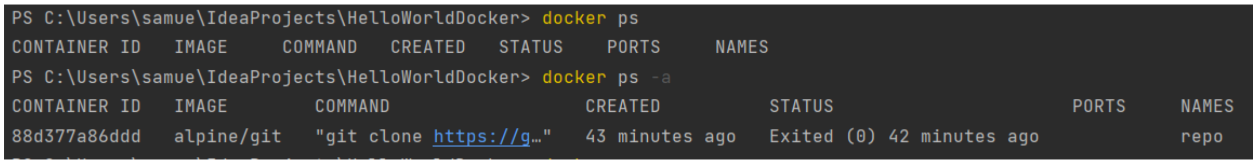
```
Build an image from a Dockerfile
PS C:\Users\samue\IdeaProjects\HelloWorldDocker>  docker build -t hello-world:1.0
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker build -t hello-world:1.0 .
[+] Building 60.4s (9/9) FINISHED
```

- Check docker images:



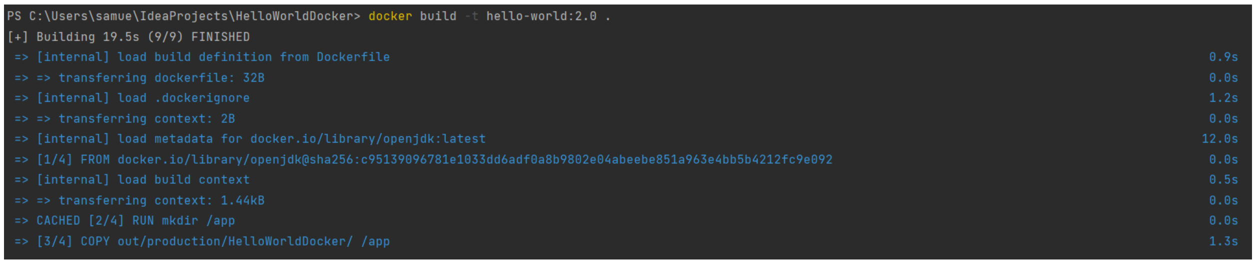
```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
hello-world   1.0       f8bdb915a0ed   19 seconds ago   471MB
```

- Run “docker ps and docker ps -a” command to check process status:



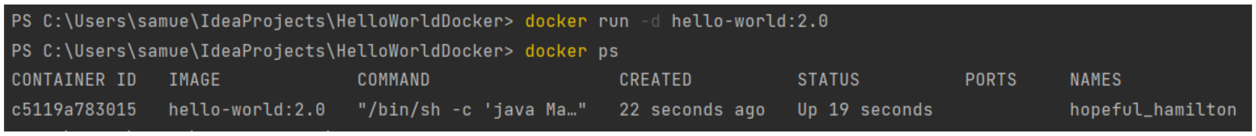
```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker ps -a
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS                      PORTS     NAMES
88d377a86ddd   alpine/git   "git clone https://g…"   43 minutes ago   Exited (0) 42 minutes ago             repo
```

- Create another docker images



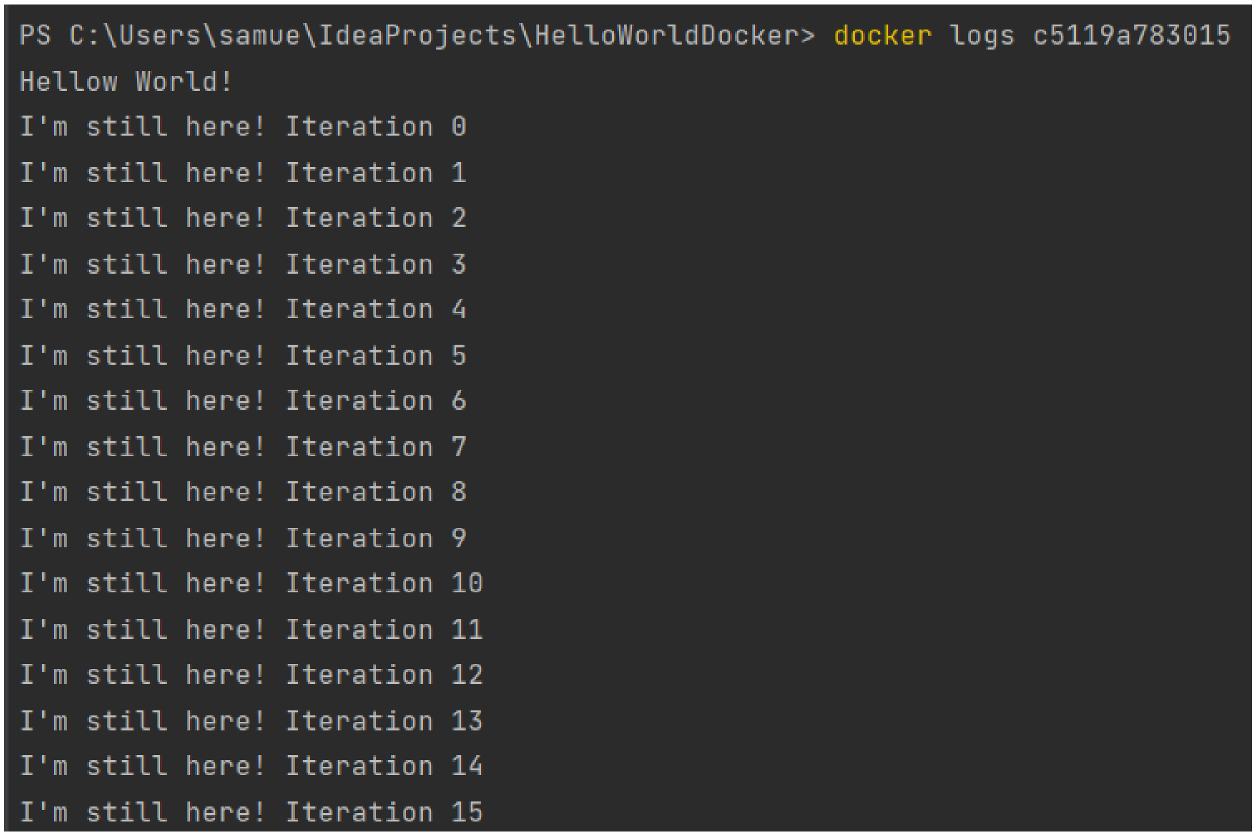
```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker build -t hello-world:2.0 .
[+] Building 19.5s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                0.9s
 => => transferring dockerfile: 32B                                                                 0.0s
 => [internal] load .dockerignore                                                                   1.2s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/openjdk:latest                                  12.0s
 => [1/4] FROM docker.io/library/openjdk@sha256:c95139096781e1033dd6adf0a8b9802e04abeebe851a963e4bb5b4212fc9e092  0.0s
 => [internal] load build context                                                                   0.5s
 => => transferring context: 1.44kB                                                                 0.0s
 => CACHED [2/4] RUN mkdir /app                                                                     0.0s
 => [3/4] COPY out/production/HelloWorldDocker/ /app                                                1.3s
```

- Run “docker run” command:



```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker run -d hello-world:2.0
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PORTS     NAMES
c5119a783015   hello-world:2.0   "/bin/sh -c 'java Ma…"   22 seconds ago   Up 19 seconds             hopeful_hamilton
```

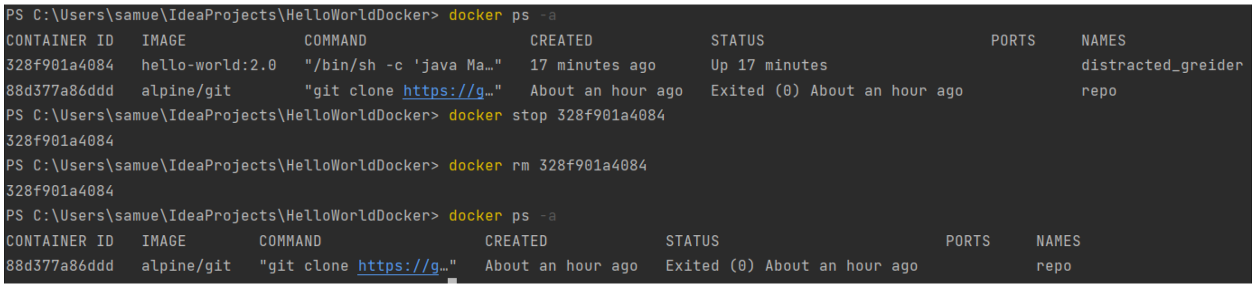
- Run “docker logs” command:



```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker logs c5119a783015
Hellow World!
I'm still here! Iteration 0
I'm still here! Iteration 1
I'm still here! Iteration 2
I'm still here! Iteration 3
I'm still here! Iteration 4
I'm still here! Iteration 5
I'm still here! Iteration 6
I'm still here! Iteration 7
I'm still here! Iteration 8
I'm still here! Iteration 9
I'm still here! Iteration 10
I'm still here! Iteration 11
I'm still here! Iteration 12
I'm still here! Iteration 13
I'm still here! Iteration 14
I'm still here! Iteration 15
```

4. Answer the following questions:
5. What are docker image, container, and registry?
   - Docker **image**: a file containing the list of instructions that will be used to create docker containers that will run on docker applications.
   - Docker **container**: component of the docker that will allow the application and its dependencies to be a package that will allow the user to share the package in an easy way.
   - **Registry**: a server side application that is highly scalable that can be used to store the docker images for transmitting it to in-house development workflow.
6. List the Docker commands used in the video with a brief description for each command and option.
   - docker build -t hello-world:2.0 .
     - To build the docker image, which retrieves from the Dockerfile. The -t option is used to give the tag to the name of the docker image, which in this case is 2.0.
   - docker ps
     - To list all the process status of each container that is running.
   - docker ps -a

        - To list all the process status of all containers from the one that is running and stopped.
    - docker run -d hello-world:2.0
        - Run the docker container with the -d option to run the container in the background and print the value of the container ID in the terminal.
    - docker logs c5119a783015
        - Show the logs of the container ID that is executed.
7. At the end of the video, there are two running containers, what commands can be used to stop and delete those two containers?
    - To stop the containers, docker **stop** *Container_ID* can be used.
    - To delete the containers, docker **rm** *Container_ID* can be used.



```
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED              STATUS                         PORTS     NAMES
328f901a4084   hello-world:2.0   "/bin/sh -c 'java Ma…"   17 minutes ago       Up 17 minutes                            distracted_greider
88d377a86ddd   alpine/git        "git clone https://g…"   About an hour ago    Exited (0) About an hour ago             repo
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker stop 328f901a4084
328f901a4084
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker rm 328f901a4084
328f901a4084
PS C:\Users\samue\IdeaProjects\HelloWorldDocker> docker ps -a
CONTAINER ID   IMAGE        COMMAND                  CREATED              STATUS                         PORTS     NAMES
88d377a86ddd   alpine/git   "git clone https://g…"   About an hour ago    Exited (0) About an hour ago             repo
```

8. 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).
    - Video Demo Link: https://drive.google.com/file/d/15RdKO9XimzoCCGkF18T4LhpURQOdKaj3/view?usp=sharing

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

7. Answer the following questions:

8. What's a multi-container Docker application?
    - An application that utilizes more than one container running on the same or one server.
    - This can be done through docker compose, which will allow two or more containers to interact with each other through shared network protocols.

9. How these containers are communicated together?
    - By exposing the port number to the container, it will allow between the docker containers, even though the content of the containers itself will not be available outside of the docker container.
    - Docker needs to publish and bind the port onto the host machine, which will allow the content to be available outside of the container, which will create the communication between the docker containers..

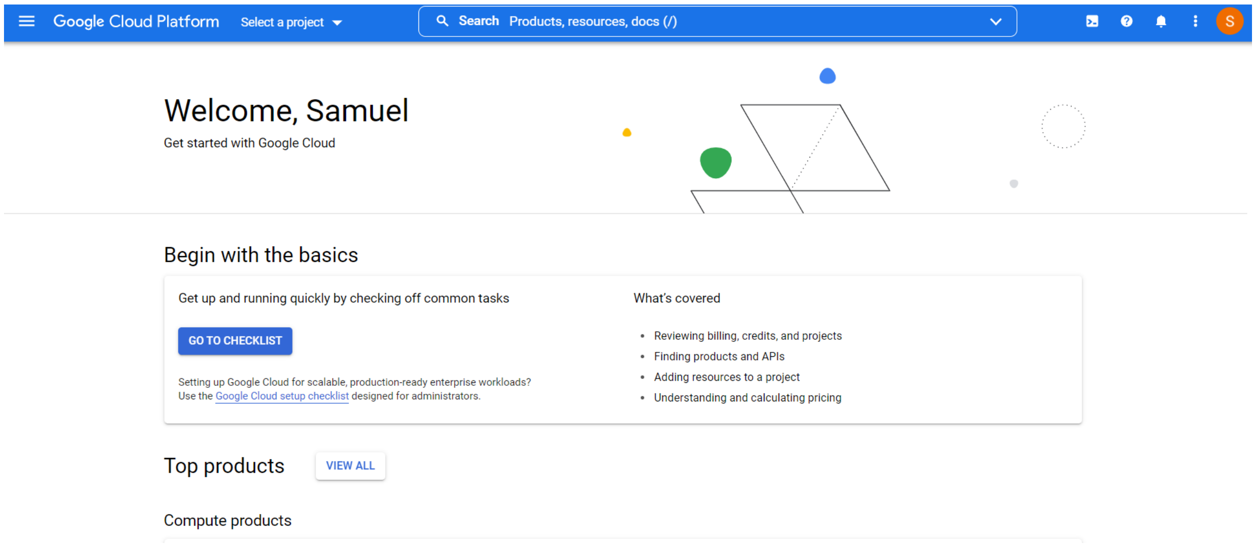
10. What command can be used to stop the Docker application and delete its images?

- **docker rm -f** "*name*", docker rm is used to remove the container that is running with the -f option to force kill the running application.
- **docker stop** "*name*" + **docker rm** "*name*", docker stop with docker rm are used to stop the running application and remove the docker application respectively.
- **docker-compose down,** another way to remove and stop the docker application that is runned through the command docker-compose up.

11. List the new docker commands used in the video with a brief description for each command and option.
    - **docker pull mysql**, which is used to pull the images from the latest version of mysql. The use of ":" can force the docker to pull the specified version of the mysql images.
    - **docker run --name app-db -d -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=myDB mysql**, which is used to create the database container with configuration settings, like the root password and database name set by using the "-e" option, which is the environment variable.
    - **docker run --name app -d my-web-app:1.0**, which is to process the application with the isolated containers that we want to use.
    - **docker network ls**, which is to list all the network available that is created by the docker compose to be used to connect the two containers for the application.
    - **docker network connect app-network app-db**, which is used to connect the container with the network that we want to connect.
    - **docker rm -f app**, which is used to remove the app as well as to force kill the running process.
    - **docker-compose up -d**, which is used to build and start to run the application by linking the containers to a host/services with the "-d" option used to run the application in background/detach mode.
    - **docker-compose down**, which is used to stop and remove the application that is set through the docker-compose up command.

12. 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").
    - Video demo link: https://drive.google.com/file/d/1zH3dRxribA3EZVx3Ef-fKy3dCokYFJP-/view?usp=sharing

13. Create a free Google Cloud Account. The first two videos in the following playlist may be helpful


Google Cloud Platform Select a project

Search Products, resources, docs (/)

Welcome, Samuel

Get started with Google Cloud

Begin with the basics

Get up and running quickly by checking off common tasks

GO TO CHECKLIST

Setting up Google Cloud for scalable, production-ready enterprise workloads? Use the Google Cloud setup checklist designed for administrators.

What's covered

- Reviewing billing, credits, and projects
- Finding products and APIs
- Adding resources to a project
- Understanding and calculating pricing

Top products VIEW ALL

Compute products

10. Watch the following videos to get familiar with Kubernetes:
11. Follow the following video to deploy dockers containers (valid until the shell session is expired) on GCP or by using Kubernetes (until you change it)
12. Prepare a video showing how the container is deployed using Docker and Kubernetes in GCP.
    - Video Demo link: https://drive.google.com/file/d/1beQMFTgs9X1EZrjCQ2C_dALDYIM08aFb/view?usp=sharing

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

Docker in GCP:

- **docker run -d -p 8080:80 nginx:latest**
    - Run the docker container in port 8080 of image nginx with the latest version.
    - The "-d" tag is used to allow the process to run in the background.
    - The "-p" tag is used to publish the port to the host machine.
- **docker cp index.html *container_id*:/usr/share/nginx/html/**
    - Have similar functionality as cp in Unix shell command.
    - Used to copy the content of index.html to the docker container created.
- **docker commit *container_id* cad/web:version1**
    - Commit the container when changes were made to the original image.
- **docker tag cad/web:version1 us.gcr.io/pivotal-tower-340119/cad-site:version1**
    - Make a tag to the target image that is referenced to the original image.
- **docker push us.gcr.io/pivotal-tower-340119/cad-site:version1**
    - Push the image created to the container registry.

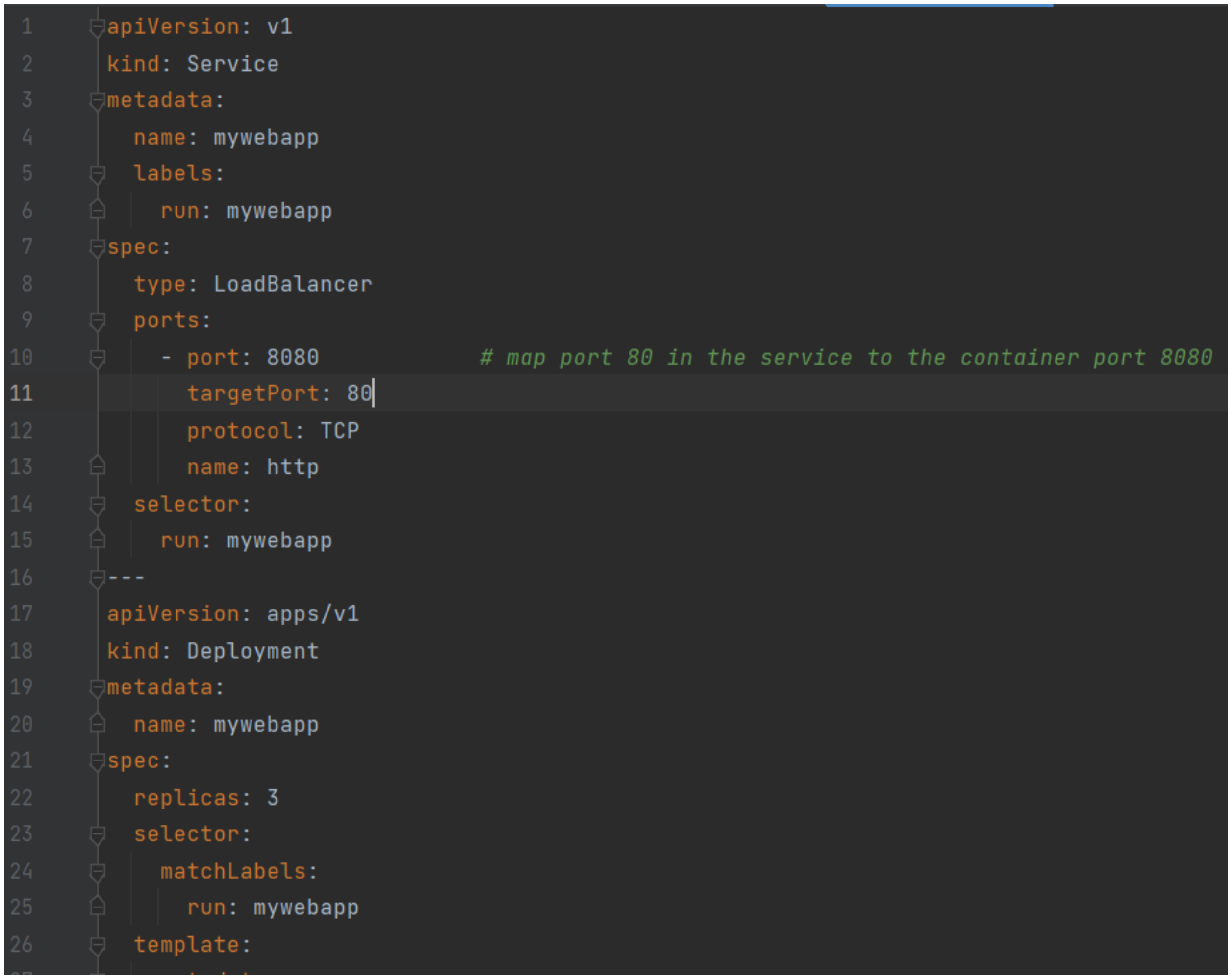
Kubernetes in GCP:

- **gcloud config set project pivotal-tower-340119**
    - Used to set configuration settings of the project

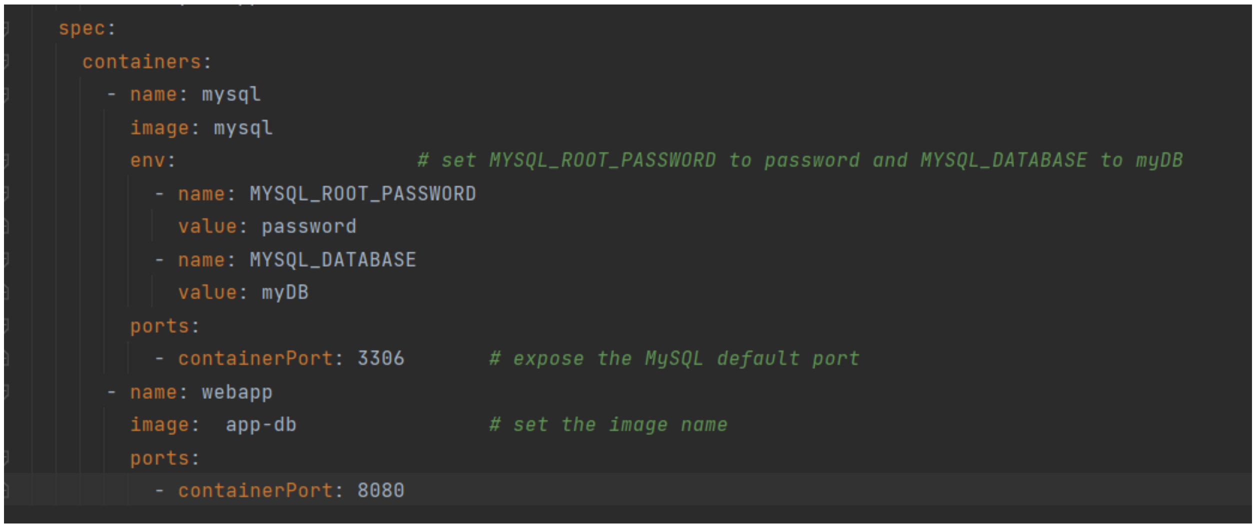
    - This will allow the user to make any changes to the project path that is set, which in this case is pivotal-tower-340119.
- **gcloud config set compute/zone us-central1-a**
    - Set the region that will be used as the default setting, which in this case is us-central1-a.
- **gcloud container clusters create gk-cluster –num-nodes=1**
    - To create Kubernetes Cluster in GCP (Google Cloud Platform).
    - The name of the cluster will be gk-cluster.
    - The number of Kubernetes' nodes is set to 1.
- **gcloud container clusters get-credentials gk-cluster**
    - Update the credentials of the gk-cluster to allow permission for the user to make changes to it.
- **kubectl create deployment web-server –images=us.gcr.io/pivotal-tower-340119/cad-site:version1**
    - Create a deployment called web-server with the image taken from the container registry.
- **kubectl expose deployment web-server –type LoadBalancer –port 80 –target-port 8080**
    - Expose the application to port 80.
    - The "-type LoadBalancer" is used to add Kubernetes service called LoadBalancer, which will allow the service to run outside of the cluster through an external IP address.
- **kubectl get pods**
    - Retrieve the status of the running pods.
- **kubectl get service web-server**
    - Retrieve the output of the pods to find the external IP address that will be used to access the web-server application.

14. Prepare a Kubernetes YML (or YAML) file to load the webApp used in steps 6:8 and deploy it using the Kubernetes engine on GCP. The file is a little different than that used by docker-compose.

- The hostname of all containers is the same and can be accessed by localhost, the address of the MySQL should be changed to localhost and recompiled. (Note: if you have a problem generating the war file, you can download it from the repository from the path "/KGS/target/MyWebApp.war").
- Create a new image using the new war file and push it to Google Container Registry.
- Follow the comments and fill the missing lines in the "/webApp.yml" file.
- Apply the YML file into Kubernetes and run the server (what is the appropriate Cloud shell command?).



```yaml
apiVersion: v1
kind: Service
metadata:
  name: mywebapp
  labels:
    run: mywebapp
spec:
  type: LoadBalancer
  ports:
    - port: 8080           # map port 80 in the service to the container port 8080
      targetPort: 80
      protocol: TCP
      name: http
  selector:
    run: mywebapp
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mywebapp
spec:
  replicas: 3
  selector:
    matchLabels:
      run: mywebapp
  template:
```



```yaml
    spec:
      containers:
        - name: mysql
          image: mysql
          env:                        # set MYSQL_ROOT_PASSWORD to password and MYSQL_DATABASE to myDB
            - name: MYSQL_ROOT_PASSWORD
              value: password
            - name: MYSQL_DATABASE
              value: myDB
          ports:
            - containerPort: 3306       # expose the MySQL default port
        - name: webapp
          image:  app-db                # set the image name
          ports:
            - containerPort: 8080
```

15. Prepare another video describing the YML file and showing how it's deployed on GCP.
16. Answer the following question:
17. What is Kubernetes' pod, service, node, and deployment?
    - **Kubernetes' pod:**
        - Deployment unit instance in Kubernetes similar to docker containers.

        - The most simple deployment in Kubernetes.
    - **Kubernetes' service:**
        - service that will allow you to utilize a set of pods in a cluster.
        - The set of pods need to have the same purpose/function in order for the service to run.
    - **Kubernetes' node:** a machine in Kubernetes that is used to contain all the pods and control the running pods/containers.
    - **Kubernetes' deployment:** a description of the state of the pods.

18. What's meant by replicas?
    - A running process contains multiple number of pods
    - Has the purpose to maintain the number of pods to be constant to prevent a loss in accessing the application if one of the pods runs into a failure.
19. What are the types of Kubernetes' services? What is the purpose of each?
    - **ClusterIP**
        - The basic service of Kubernetes.
        - Unable to request to Kubernetes' pods outside of the cluster.
    - **LoadBalancer**
        - Allow to use the service outside of the cluster through external IP address.
        - Asynchronously created.
    - **NodePort**
        - Allow the service to run in the same port number.
        - Can run outside of the cluster by assigning the *Node_IP:Node_port* to the service.
        - The node must be ensured to be available (cannot use the port that is unavailable).
    - **ExternalName**
        - Depict the service to the DNS name.
        - The service is assigned to the content of the externalName field, which will result in the return of the CNAME record and value.
20. Upload your report, used files, and videos (links) into your repository.