CS 450 Project Part 1

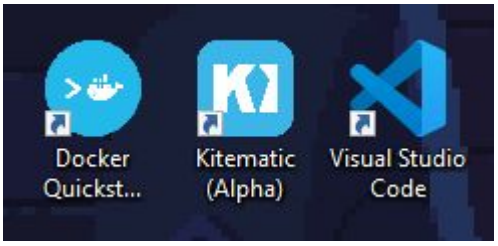
Intended outcomes: A1(Education and practice), A2(Open-source software, github project)

Project Topic: Docker and Kubernetes

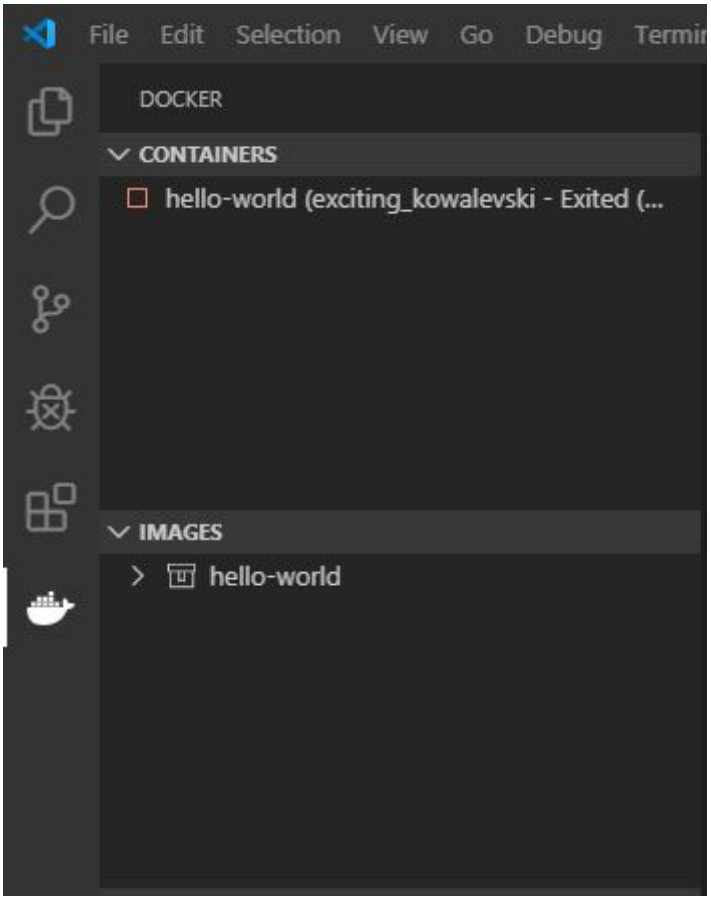
Project Contributors: Ozan TARLAN, S012259, ozan.tarlan@ozu.edu.tr

Education and practice report(Effort 7-8 hours):

- Researched on the topic Docker and Kubernetes
- Docker Toolbox (Docker desktop was not suitable for Windows 10 home edition) and Vscode downloaded



- Set up for docker and made sure it was working properly with hello world.



- Decided on a small yet powerful linux distribution called alpine linux. Ideal for a Docker container to run on.

- Found a sample code from https://www.geeksforgeeks.org/quick-sort/ Its a quicksort algorithm written in java.
- My aim was to use the alpine linux image and execute a sample java code using openjdk image in a Docker container.


```
docker pull openjdk
```

qsort java > Dockerfile > ...

```
1   FROM alpine:latest
```

- Found a tutorial where alpine and openjdk images are used together to execute a java Hello world code. https://www.youtube.com/watch?v=mean7OgfF44
- Modified the Dockerfile from the tutorial to use it for my image.

qsort java > Dockerfile > ...

```
1   FROM alpine:latest
2   WORKDIR /app
3   COPY QuickSort.java /app
4
5
6   RUN apk add openjdk8
7   ENV JAVA_HOME /usr/lib/jvm/java-1.8-openjdk
8   ENV PATH $PATH:$JAVA_HOME/bin
9
10  RUN javac QuickSort.java
11
12  ENTRYPOINT java QuickSort
```

- built and run the image:

```
PS C:\Users\Ozan\Desktop\cs 450 proj part 1\qsort java> docker build -t quicksort_java_final .
Sending build context to Docker daemon   5.12kB
Step 1/8 : FROM alpine:latest
 ---> 965ea09ff2eb
Step 2/8 : WORKDIR /app
 ---> Using cache
 ---> b601ffc6ad15
Step 3/8 : COPY QuickSort.java /app
 ---> Using cache
 ---> e8d0bb06740a
Step 4/8 : RUN apk add openjdk8
 ---> Using cache
 ---> f956c70ab2e3
Step 5/8 : ENV JAVA_HOME /usr/lib/jvm/java-1.8-openjdk
 ---> Using cache
 ---> a1a73b21a271
Step 6/8 : ENV PATH $PATH:$JAVA_HOME/bin
 ---> Using cache
 ---> 679348cf8003
Step 7/8 : RUN javac QuickSort.java
 ---> Using cache
 ---> e8140731f668
Step 8/8 : ENTRYPOINT java QuickSort
 ---> Using cache
 ---> 7842f138b1db
Successfully built 7842f138b1db
Successfully tagged quicksort_java_final:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x
nd reset permissions for sensitive files and directories.
PS C:\Users\Ozan\Desktop\cs 450 proj part 1\qsort java> docker run quicksort_java_final
sorted array
1 5 7 8 9 10
PS C:\Users\Ozan\Desktop\cs 450 proj part 1\qsort java>
```

- My container successfully executed and gave expected results.
- Researched Kubernetes to orchestrate and manage the docker containers.
- Researched Kubernetes concepts like load balancing and elastic scaling to use in the project part 2.

Project's aim for Part 2:

- To use Kubernetes to create, manage, orchestrate 10s of copies of the same Docker instance.
- To utilize load balancing and elastic scaling using Kubernetes.
- To upload the project to the Github.