Group 9: Assignment 3.1 | Container Virtualization

I. DESIGN & IMPLEMENTATION | The third iteration of our url shortener application added container virtualization to our two services: a login-service and a shortener-service. To implement the container virtualization we used Docker[1]. The latter is also known as dockerization. Figure 1 depicts the final design of this version of our application. Each web service developed in Python + Flask is dockerized. On each container we develop a dockerfile to expose a specific port in order for external clients to be able to access them. Each of our dockerfile consists of the image definition, setting up a working directory, exposing the application ports, and preparing (i.e. installing dependencies) and launching the application. The login-service is mapped to port 8081 and the shortener-service is mapped to port 8080. Furthermore, we made use of docker-compose[2] to easily set up and launch both services with one simple command (i.e. `docker-compose up -d`). In addition to this, in the docker-compose file we created an intra-container network which is essential for containers to be able to communicate between them.

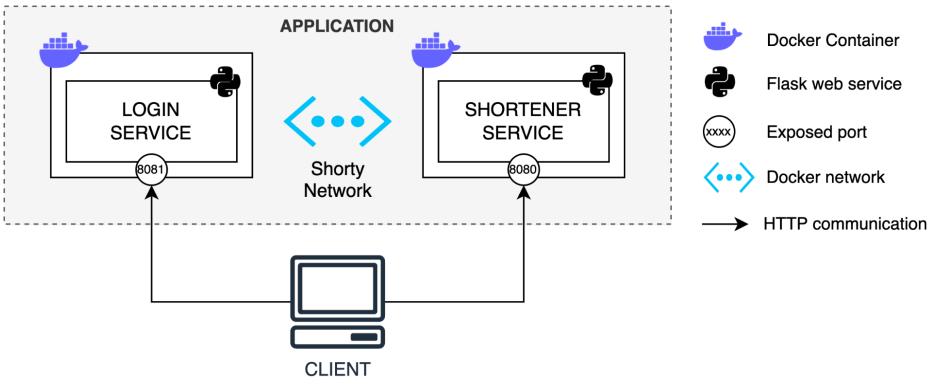

Figure 1. The design of our application under a container virtualization architecture.

II. EXPERIENCE WITH DOCKER | Docker can be confusing at the beginning but rewarding at the end. At the beginning we had to transform the theory learned in class to practice. The latter turned out to be challenging and confusing. However, once we started to really understand and dive in depth on the concepts of images, volumes, networks and the docker paradigm, we were able to successfully dockerize our services. Afterwards, we were able to acknowledge and experience the advantages of dockerizing our application. We felt that in some way, our application became fully "portable" since it works on top of container virtualization. Hence, setting up the environment for testing and development in any computer is a relatively easy task. Thus, the testing, development and deployment of our application became more robust. We also discover that docker-compose is a powerful tool on top of docker that fully completes the circle of advantages of docker. We find it incredible that with one simple docker-compose command our entire application environment was up and running. As a conclusion, we prefer docker as a deployment method instead of a separate deployment as individual processes. One of the biggest challenges we faced during the implementation of our migration was to set up the network for inter-contained communication since we needed to learn differences between internal and external networks and how to set them up. Since the internet is full of resources we did not have any more problems setting up the dockerfiles. The real barrier and challenge was to really understand in-depth the concepts of docker containerization. The latter is the only drawback we could find of docker.

[1] https://www.docker.com/
[2] https://docs.docker.com/compose/

III. MEMBERS CONTRIBUTION

Tong Wu	*Kubernetes config (3.2)*
Zhaolin Fang	*Kubernetes config (3.2)*
Leonardo Kuffo	*Dockerfiles & docker-compose file (3.1)*