Web Services and Cloud Based Systems - Assignment 3

Arumoy Shome
VU ID: 2636393

Shruti Rao
VU ID: 2631481

Group 13

The project setup (Figure 1) contains two folders - *docker-main/* and *docker-sidekick/* at the root.The *docker-main/* directory contains the url shortner service files, a *Dockerfile*, a *docker-compose.yml* file and a *requirements.txt* file. The *docker-sidekick/* directory contains an additional *Dockerfile*.

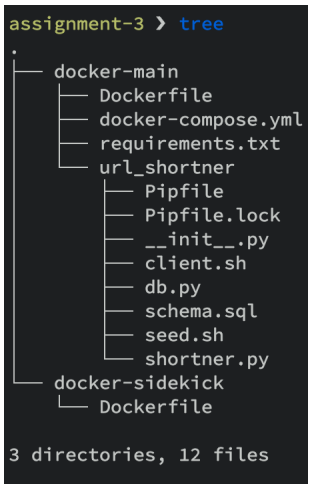

Figure 1: Project Structure

Assignment 3.1

Containers on the Same Host

First, a Dockerfile is created to define an image. The docker-main/Dockerfile contains instructions to obtain latest Python runtime as parent image. It sets the working directory at */app* and copies the contents of current directory to the container at */app*. It installs packages specified in *requirements.txt* using `pip`. It defines Ubuntu and Flask specific environment variables and sets flask to run at `http://0.0.0.0:5000/` during runtime [1].

For the second container, the *docker-sidekick/Dockerfile* is created with *Ubuntu:16.4* runtime. *Curl* is installed so that the service running in the main container can be called from the sidekick container. To enable communication between containers on the same host, a bridge network is used. Containers get connected to the bridge network by default [1]. By *attaching* to the sidekick container, we can curl the service using the IP address of the main container [2].

Containers on Remote Host

To enable communication between containers on remote hosts, *Docker Swarm* is required. Here, a swarm refers to a group of machines operating as a cluster with a swarm manager and workers [1]. To imitate the concept of two different hosts, *VirtualBox* is used to set up a two virtual machines (VM) [3]. The first VM acts as the manager while the second VM is a worker. First, the main VM that holds the url shortner service is initialized to be the swarm manager. Next, the second VM is

added to the swarm as the worker. The url-shortner app is deployed using the swarm manager and the *docker-compose.yml*.

The *docker-compose.yml* file is a *YAML* file that defines how the url shortner application container should behave in "production". The *docker-compose.yml* file pulls the url-shortner image from Docker Hub if not already present. It runs 2 instances of that image as a service, limiting each instance to use no more than 10% of a single core of CPU time. As a safety mechanism, it can immediately restart a container if it fails. Port 4000 is mapped to 5000 and containers are made to share the port via a load-balanced network. Finally, the network is defined with the default settings [1].

The service can now be accessed by visiting the IP address of either VM or by curling for the service from second VM.

Experience with Docker

Docker was a great experience from the get go. The documentation was very well organized and enabled us with no prior experience to set up the system and start containers without much hassle. We were able to focus on the concepts and the application rather than having to struggle setting up Docker. It worked extremely well right out of the box. The containers were comparatively very lightweight with only megabytes in size and faster to start than virtual machines.

Assignment 3.2

The deployment process for Kubernetes was very similar to that of Docker Swarm. We used Docker for Mac that comes with a standalone Kubernetes server and client. With the same setup as in the previous section, Kubernetes was enabled to deploy containers via orchestration rather than Docker Swarm. To allow Kubernetes to orchestrate the containers of a cluster, some adjustment had to be made on the Docker app for MacOS [1]. After which, the same `docker stack` commands which are used for deploying with Docker Swarm were used to deploy Kubernetes pods [1]. The *docker-compose.yml* file was updated to make 3 replicas of the service and 3 VMs were used for the Kubernetes cluster. By default, Kubernetes deploys pods to the overlay network called Ingress that manages data traffic. The Ingress bridge serves as a Load Balancer as incoming requests to the service are forwarded to the next available pod by the bridge. The experience with Kubernetes was very pleasant as everything worked out of the box.

References

[1] *Get Started, Part 1: Orientation and setup.* 2019. URL: `https://docs.docker.com/get-started/`.

[2] *Welcome.* URL: `http://flask.pocoo.org/`.

[3] *Welcome to VirtualBox.org!* URL: `https://www.virtualbox.org/`.