Assignment 4: Orchestration and contextualization using docker containers

Course: Data Engineering I

Introduction

This assignment covers the practical part of the discussed concepts of contextualization and orchestration for distributed applications. The assignment consists of three tasks. Task 1 and 2 are compulsory for everyone and Task 3 can award one point toward higher marks. For all three tasks, Docker container engine is used on a single Linux virtual machine. Task 1 covers basic introduction of the Docker environment. Task 2 requires a reasonably firm understanding of the Docker environment, Dockerfile settings and construction of a multi-container environment using docker-compose (one of the essential tools to build, connect and run multiple containers). Task 3 is a theoretical task that requires a literature survey of different orchestration and contextualization frameworks, theoretical understanding of their architectures and reflection on gains and challenges related to the frameworks.

Good Luck!

Task 1+2 are compulsory and together award 1 point. Task 3 can award 1 extra point counting towards higher marks.

Task 1. Introduction to Docker containers and DockerHub

Step 0. Install Docker on your VM.

Start a small VM on SNIC Science Cloud using Ubuntu 20.04 image.

0 - Switch to the root user:

```
    > sudo bash

    # apt update; apt -y upgrade;
```

1 - Install the required packages

```
    # apt install apt-transport-https ca-certificates
curl software-properties-common
```

2 - First, add the GPG key for the official Docker repository to the system

```
    # curl -fsSL https://download.docker.com/
    linux/ubuntu/gpg |sudo apt-key add -
```

3 - Add the Docker repository to APT sources

```
    # add-apt-repository "deb [arch=amd64] https://
    download.docker.com/linux/ubuntu bionic stable"
```

4 - Update the package database with the Docker packages from the newly added repo

```
    # apt-get update
```

5 - Install Docker

```
    # apt install docker-ce
```

6 - (Optional) Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running

```
    # systemctl status docker
```

For more information visit: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

Step 1. Contextualize a container.

0 - Create a file name "Dockerfile " and add the following contents in the file:

```
    FROM ubuntu:20.04
    RUN apt-get update
    RUN apt-get -y upgrade
    RUN apt-get install sl
    ENV PATH="${PATH}:/usr/games/"
    CMD ["echo", "Data Engineering-I."]
```

2 - Contextualize a container.

```
    # docker build -t mycontainer/first:v1 .
```

3 - Step 2 will create a new image that is contextualized according to the instructions available in the Dockerfile. Now start a container based on the contextualized image:

In batch mode

```
    # docker run mycontainer/first:v1

    Data Engineering-I.
```

In interactive mode:

```
    # docker run -it mycontainer/first:v1 bash
```

The command will give you access to the root terminal of the newly started container.

4 - Run the following command:

```
    # sl
```

The output will be a running train.

Now we have a running docker container with some extra packages installed in it.

Question:

1 - Explain the difference between contextualization and orchestration processes.

2 - Explain the followings commands and concepts:

i) Contents of the docker file used in this task.

ii) Explain the command

```
    # docker run -it mycontainer/first:v1 bash
```

iii) Show the output and explain the following commands:

```
    # docker ps
    # docker images
    # docker stats
```

3 - What is the difference between `docker run`, `docker exec` and `docker build` commands?

4 - Create an account on DockerHub and upload your newly built container to your DockerHub area. Explain the usability of DockerHub. Make your container publicly available and report the name of your publicly available container.
5 - Explain the difference between `docker build` and `docker-compose` commands.

Task 2. Build a multi-container Apache Spark cluster using `docker-compose`

Introduction to Apache Spark

Apache Spark is a distributed computing framework that utilizes the Map-Reduce paradigm to allow parallel processing. A Spark cluster consists of a master and multiple worker nodes setup. It is a highly scalable framework that works equally well for both batch and streaming processing workflows.

Installation instructions

0 - There are different ways to orchestrate and contextualize a containerized Spark cluster. Following Dockerfile gives you an ALL-IN-ONE primitive solution based on a single Dockerfile

```
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get -y upgrade
RUN apt install -y openjdk-8-jre-headless
RUN apt install -y scala
RUN apt install -y wget
RUN apt install -y screen
RUN wget https://archive.apache.org/dist/spark/
spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
RUN tar xvf spark-3.2.0-bin-hadoop3.2.tgz
RUN mv spark-3.2.0-bin-hadoop3.2/ /usr/local/spark
ENV PATH="${PATH}:$SPARK_HOME/bin"
ENV SPARK_HOME="/usr/local/spark"
ENV SPARK_NO_DAEMONIZE="true"
RUN sleep 15
CMD screen -d -m $SPARK_HOME/sbin/start-
master.sh ; $SPARK_HOME/sbin/start-worker.sh
spark://sparkmaster:7077
```

1 - Build the image based on the Dockerfile

```
# docker build -t sparkaio/first:v0 .
```

2 - Run the container with the following command:

```
# docker run -h sparkmaster <Generated-Image-ID or
image name>
```

Login to the container and run the following command to confirm that the spark setup is working correctly: (notice that every # means one command line)

```
# docker exec -it container_id /bin/bash
# $SPARK_HOME/bin/spark-submit --class
org.apache.spark.examples.SparkPi --master spark://
sparkmaster:7077 $SPARK_HOME/examples/jars/spark-
examples_2.12-3.2.0.jar
```

You should get an output like: `Pi is roughly 3.1448357241786207`

Based on the above configurations, you have created a single container-based Spark framework. Now your task is to prepare a configuration file, compatible with docker-compose, and run a Spark cluster with at least one master node and one additional worker. Please note that the task is to run a *multi-container setup*, not a multi-node setup. The suggestion is to first read available online tutorials (For example https://www.baeldung.com/docker-compose) on `docker-compose` and then start with the task. Together with the `docker-compose` based solution for Spark cluster, submit related files and the answers to the following three questions:

Questions

1 - Explain your `docker-compose` configuration file.
2 - What is the format of the `docker-compose` compatible configuration file?
3 - What are the limitations of `docker-compose`?

Task 3. Introduction to different orchestration and contextualization frameworks (1 point)

Write an easy on the role of runtime orchestration and contextualization for large scale distributed applications. Briefly discuss the features and design philosophy of at least four relevant frameworks. In Task 2, you have orchestrated a multi-container Spark cluster using `docker-compose`. Discuss how the features of frameworks like Kubernetes and Docker Swarm can improve your current solution for providing a Spark cluster. The expected length of the easy will be one A4 page 11 pt Arial.