# Assignment 1: Orchestration and contextualization using docker containers

Course: Data Engineering II

## 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, the 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

## 0. Install Docker on your VM

Start a small or medium VM on Swedish Science Cloud using Ubuntu 22.04 image. NOTE: Do *NOT* create new volume for your VM (unselect the option on “Launch Instance -> Source”).

Step 0 - Update and upgrade `apt`:

```
sudo apt update; sudo apt -y upgrade;
```

Step 1 - Install Docker with the convenience script:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Step 2 - Run post installation scripts to preface `docker` without `sudo`:

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

Step 3 - Make sure Docker is properly installed

```
docker --version
docker compose version
```

For more information visit:
https://docs.docker.com/engine/install/ubuntu/

## 1. Contextualize a container

Step 0 - Create a file named “dockerfile” and add the following contents in the file:

```
FROM ubuntu:22.04
RUN apt-get update
RUN apt-get -y upgrade
RUN apt-get -y install sl
ENV PATH="${PATH}:/usr/games/"
CMD ["echo", "Data Engineering-II."]
```

Step 2 - Contextualize a container.

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

Step 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
```

- 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. And then 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:

- Contents of the docker file used in this task.
- Explain the command

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

- 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

Step 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:18.04
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-2.4.3/spark-2.4
.3-bin-hadoop2.7.tgz
RUN tar xvf spark-2.4.3-bin-hadoop2.7.tgz
RUN mv spark-2.4.3-bin-hadoop2.7/ /usr/local/spark
ENV PATH="${PATH}:$SPARK_HOME/bin"
ENV SPARK_HOME="/usr/local/spark"
ENV SPARK_NO_DAEMONIZE="true"
RUN sleep 5
CMD screen -d -m $SPARK_HOME/sbin/start-master.sh ;
$SPARK_HOME/sbin/start-slave.sh spark://sparkmaster:7077
```

Step 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 sparkaio/first:v0
```

Then login to the container and run the following command to confirm that the spark setup is working correctly:

```
$SPARK_HOME/bin/spark-submit --class
org.apache.spark.examples.SparkPi --master
spark://sparkmaster:7077
$SPARK_HOME/examples/jars/spark-examples_2.11-2.4.3.jar
```

Expected output: `Pi is roughly 3.142115710578553`

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 on `docker compose` and then start with the task. Together with the `docker compose` based solution for Spark cluster, submit the answers to the following questions.

References:

- Introduction to Docker Compose | Baeldung on Ops (Note: Docker compose v1, which has been deprecated, used to be a standalone project, which uses command `docker-compose`, while docker compose v2 is an integrated command of docker, which uses command `docker compose`. Although this article is helpful to get one familiarized with docker compose, it is based on docker compose v1. All commands `docker-compose` in the article should be replaced as `docker compose`)
- Docker Compose Quickstart

## 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 essay 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.