synch_04

UCB MIDS W205 Summer 2018 - Kevin Crook's agenda for Synchronous Session #4

Start updated docker images in your droplet (not in docker container)

We are going to start running clusters of docker containers in your droplet. Some of the images we will be running are very large and require several minutes to bring down. To help save class time, please update these images prior to class start. Also, we many find issues with the images and need to push new images with fixes. So, updating images before class and before you do any meaningful work on assignments is always a good idea.

Run these command in your droplet (but **NOT** in a docker container):

```
docker pull midsw205/base
docker pull confluentinc/cp-zookeeper:latest
docker pull confluentinc/cp-kafka:latest
```

Update the course-content repo in your docker container in your droplet

See instructions in previous synchronous sessions.

Review the Query Project

We will spend a few minutes reviewing the Query Project now that everyone has been actively working on it and is more familiar with the dataset.

Breakout - group discussions about the Query Project

We will go into breakout so everyone can discuss the Query Project. The following is a list of suggested topics to discuss, but this is open ended, you group can discuss anything they feel will help with the project. Time permitting, we will repeat this in following weeks.

- Discuss the 3 tables in the dataset, the data in each, and what type of queries we could find out from each table
- What is a trip?
- What is a commuter trip?
- For each of the following pricing options, how would you design a query to detect the customer base?
- a flat price for a single one-way trip
- a day pass that allows unlimited 30-minute rides for 24 hours
- an annual membership
- What kinds of analytical questions could we ask to help us analyze the customer base, if they are worth targeting, and how to target them?
- Does the fixed station model with docks help us with analytics? Does it hurt us with the customer base?

Revisit docker concepts in single container mode

We will revisit docker concepts in a single container mode. This will prepare us to understand docker in a cluster mode.

Run our regular docker container that we have been using to clone and update our course-content repo and been using to work on our assignments.

- docker in our executable
- run tell docker to run a container as an instance of an image
- -it
- i and t are separate arguments
- -i means interactive mode (standard input / output should be piped appropriately)
- -t means create a pseudo terminal (allows us to use curses to format the display)
- –rm tells docker to remove the containter after it stops. Without this option a container will remain after it stops and need to be restarted or cleaned up (advanced topic for later).
- -v specified a volume mount. We mount the directory /home/science/w205 in our droplet to the the directory /w205 in our docker container. Data saved to /w205 in our docker container will outlive the container.
- midsw205/base:latest
- midsw205 is the docker hub repo
- base is the image
- latest is the version
- bash is the command we will be running in the container, which of course is the bash shell

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base:latest bash
```

Some docker commands

Open another terminal window and try the following commands. These should be run inside the droplet, but **NOT** inside a docker container. From this point on, using multiple terminal windows will prove very helpful.

What containers are running right now?

```
docker ps
```

What containers exist, running or not?

```
docker ps -a
```

What images do I have?

```
docker images
```

Run a container without the –rm and it will stay after it's stopped

```
docker run -it -v /home/science/w205:/w205 midsw205/base:latest bash
exit
```

```
docker ps -a
```

Remove containter

```
docker rm -f <container>
```

Microservices using Docker

Discuss services and web services. Discuss how bugs in services can propogate between web API calls. Discuss how microservices can limit propogation of bugs, but has the disadvantage of higher overhead.

Run the following snipped of code which starts a container with a bash shell, manually runs the pwd command, and exits the container.

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base:latest bash
pwd
exit
```

Run the same code as a microservice:

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base pwd
```

We could also run a bq query to Google BigQuery as a microservice:

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base:latest \
bq query --use_legacy_sql=false 'SELECT count(*) FROM `bigquery-public-data.san_francisco.bikeshare_sta
```

Same command on 1 line to make copy and paste easier:

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base:latest bq query --use_legacy_sql=false 'SE
```

Running a docker cluster using docker compose

Docker compose is a utility that allows us to create and run docker clusters.

We specify a docker cluster using a .yml file.

The following commands should be run in your droplet, but not inside a docker container.

Create a kafka directory:

```
mkdir ~/w205/kafka
```

Change to the kafka directory:

```
cd ~/w205/kafka
```

Use vi to create a file docker-compose.yml with the contents presented below:

```
vi docker-compose.yml
```

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    network_mode: host
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    extra_hosts:
      - "moby:127.0.0.1"

  kafka:
    image: confluentinc/cp-kafka:latest
    network_mode: host
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: localhost:32181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:29092
```

```
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    extra_hosts:
      - "moby:127.0.0.1"
```

Bring up the docker cluster. The -d option tells us to detach the cluster from the current terminal - essentiall it will run "headless". Anytime we use the docker-compose command, current directory is important, as docker-compose will look for a docker-compose.yml file and use that information. If we are in a directory without a docker-compose.yml file, docker-compose will throw an error. If we are in a directory with the wrong docker-compose.yml, we may corrupt things and need to do a manual cleanup at the docker level.

```
docker-compose up -d
```

Check and see if our cluster is running:

```
docker-compose ps
```

Check in docker to see the individual containters and verity that they are properly part of a docker cluster:

```
docker ps -a
```

Tear down the cluster:

```
docker-compose down
```

Verify that the cluster is properly down:

```
docker-compose ps
```

```
docker ps -a
```