UCB MIDS W205 Summer 2018 - Kevin Crook's agenda for Synchronous Session #9

Update docker images (before class)

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

```
docker pull midsw205/base:0.1.8
docker pull confluentinc/cp-zookeeper:latest
docker pull confluentinc/cp-kafka:latest
```

Update the course-content repo in your docker container in your droplet (before class)

See instructions in previous synchronous sessions.

Project 3 - Understanding User Behavior Project

Assignment-09 - Define your Pipeline

Assignment-10 - Setup Pipeline, Part 1

Assignment-11 - Setup Pipeline, Part 2

Assignment-12 - Synthesis Assignment

Activity - setup a web server running a simple web API service which will service web API calls by writing them to a kafka topic, using curl make web API calls to our web service to test, manually consume the kafka topic to verify our web service is working

Create a directory:

```
mkdir ~/w205/flask-with-kafka
cd ~/w205/flask-with-kafka
```

Create a `docker-compose.yml` with the following. Remember to fix the drive mappings if needed:

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    expose:
      - "2181"
      - "2888"
      - "32181"
      - "3888"
    extra_hosts:
      - "moby:127.0.0.1"
```

```
  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:32181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    expose:
      - "9092"
      - "29092"
    extra_hosts:
      - "moby:127.0.0.1"

  mids:
    image: midsw205/base:0.1.8
    stdin_open: true
    tty: true
    volumes:
      - ~/w205:/w205
    expose:
      - "5000"
    ports:
      - "5000:5000"
    extra_hosts:
      - "moby:127.0.0.1"
```

Start the docker cluster:

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

Create the kafka topic events (as we have done before):

```
docker-compose exec kafka \
   kafka-topics \
     --create \
     --topic events \
     --partitions 1 \
     --replication-factor 1 \
     --if-not-exists \
     --zookeeper zookeeper:32181
```

The same command on 1 line for convenience:

```
docker-compose exec kafka kafka-topics --create --topic events --partitions 1 --replication-factor 1 --
```

Should see the following output:

```
Created topic "events".
```

Example scenario: * You're a mobile game developer. During gameplay, your users perform various actions such as * purchase a sword * purchase a knife * join a guild * To process these actions, your mobile app makes API calls to a web-based API-server.

We will use the python flask module to write a simple API server.

- Use the python `flask` library to write our simple API server. Create a file `~/w205/flask-with-kafka/game_api.py` with the following python code:

```
#!/usr/bin/env python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def default_response():
    return "\nThis is the default response!\n"

@app.route("/purchase_a_sword")
def purchase_sword():
    # business logic to purchase sword
    return "\nSword Purchased!\n"
```

Run the python script using the following command. This will tie up this linux command line window. We will see output from our python program here as we make our web API calls:

```
docker-compose exec mids env FLASK_APP=/w205/flask-with-kafka/game_api.py flask run
```

Using another linux command line window, use curl to make web API calls. Note that TCP port 5000 is the port we are using.

```
docker-compose exec mids curl http://localhost:5000/
docker-compose exec mids curl http://localhost:5000/purchase_a_sword
```

In the flask window, stop flask with a control-C

Edit our python flask script to publish to the kafka topic in a addition to writing to standard output:

```
#!/usr/bin/env python
from kafka import KafkaProducer
from flask import Flask
app = Flask(__name__)
event_logger = KafkaProducer(bootstrap_servers='kafka:29092')
events_topic = 'events'

@app.route("/")
def default_response():
    event_logger.send(events_topic, 'default'.encode())
    return "\nThis is the default response!\n"

@app.route("/purchase_a_sword")
def purchase_sword():
    # business logic to purchase sword
    # log event to kafka
    event_logger.send(events_topic, 'purchased_sword'.encode())
    return "\nSword Purchased!\n"
```

Run the python flask script as before:

```
docker-compose exec mids env FLASK_APP=/w205/flask-with-kafka/game_api.py flask run
```

In another linux command line windows, use curl to make web API calls.

```
docker-compose exec mids curl http://localhost:5000/
docker-compose exec mids curl http://localhost:5000/purchase_a_sword
```

Use kafkacat to consume the messages that our web service wrote to the kafka topic:

```
docker-compose exec mids bash -c "kafkacat -C -b kafka:29092 -t events -o beginning -e"
```

Optional, if you want, go back and generate more web API calls and consume the topic to see how they show up.

In the flask window, stop flask with a control-C

Tear down the docker cluster:

```
docker-compose down
```