synch_06

UCB MIDS W205 Summer 2018 - Kevin Crook's agenda for Synchronous Session #6

Update docker images (before class)

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

```
docker pull midsw205/base:latest
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.

Discuss Project 2: Tracking User Activity

Assignment 6 - Get and Clean Data

Assignment 7 - Setup Pipeline

Assignment 8 - Build and Write-up Pipeline

Create a docker cluster with kafka and zookeeper containers, create a kafka topic, publish messages to the topic, subscribe / consume the messages from the topic

Create a kafka directory and change to it:

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

Using vi, create a docker-compose.yml file with the following contents:

```
---
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"
```

Startup the container:

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

Verify the cluster is running:

```
docker-compose ps
```

Which should show something like:

```
    Name                        Command            State   Ports
    -----------------------------------------------------------------------
    kafkasinglenode_kafka_1       /etc/confluent/docker/run   Up
    kafkasinglenode_zookeeper_1   /etc/confluent/docker/run   Up
```

Check the zookeeper logs for entries regarding the binding:

```
docker-compose logs zookeeper | grep -i binding
```

Which should show something like:

```
zookeeper_1  | [2016-07-25 03:26:04,018] INFO binding to port 0.0.0.0/0.0.0.0:32181
(org.apache.zookeeper.server.NIOServerCnxnFactory)
```

Check the kafka logs to see that the kafka broker has started:

```
docker-compose logs kafka | grep -i started
```

Which should show something like:

```

    kafka_1      | [2017-08-31 00:31:40,244] INFO [Socket Server on Broker 1], Started 1 acceptor threa
    kafka_1      | [2017-08-31 00:31:40,426] INFO [Replica state machine on controller 1]: Started repl
    kafka_1      | [2017-08-31 00:31:40,436] INFO [Partition state machine on Controller 1]: Started pa
    kafka_1      | [2017-08-31 00:31:40,540] INFO [Kafka Server 1], started (kafka.server.KafkaServer)
```

Create a kafka topic called foo.

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

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

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

Which should show something like:

```
Created topic "foo".
```

Check the topic:

```
docker-compose exec kafka \
  kafka-topics \
    --describe \
    --topic foo \
    --zookeeper localhost:32181
```

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

```
docker-compose exec kafka kafka-topics --describe --topic foo --zookeeper localhost:32181
```

Which should show something like:

```
    Topic:foo   PartitionCount:1    ReplicationFactor:1 Configs:
    Topic: foo  Partition: 0    Leader: 1    Replicas: 1  Isr: 1
```

Publish messages to the kafka topic foo in the form of the numbers from 1 to 42:

```
docker-compose exec kafka \
  bash -c "seq 42 | kafka-console-producer \
    --request-required-acks 1 \
    --broker-list localhost:29092 \
    --topic foo && echo 'Produced 42 messages.'"
```

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

```
docker-compose exec kafka bash -c "seq 42 | kafka-console-producer --request-required-acks 1 --broker-li
```

Which should show something like:

```
Produced 42 messages.
```

Subscribe / consume the messages from the kafka topic foo:

```
docker-compose exec kafka \
  kafka-console-consumer \
    --bootstrap-server localhost:29092 \
    --topic foo \
    --from-beginning \
    --max-messages 42
```

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

```
docker-compose exec kafka kafka-console-consumer --bootstrap-server localhost:29092 --topic foo --from-
```

Which should show something like:

```
    1
    ....
    42
    Processed a total of 42 messages
```

Tear down the cluster:

```
docker-compose down
```

Breakout - discuss applications of MQ (Message Queue) oriented systems such as kafka

Consider this airline example:

- passengers make reservations
- pay for reservations
- reserve seats
- check in for flights
- check bags
- go through security
- board planes
- take flights
- retrive bags
- participate in frequent flyer programs
- etc.

Discuss the following:

- what topics should we create?
- for each topic, what messages could we publish to that topic?
- for each topic, what is the advantage of having multiple systems subscribe / consumer the same topic?
- give examples of analytical systems that could subscribe to the topics you mentioned and what types of analytics could be done?
- consider the Lambda Architecture mentioned in the readings. What layer of the Lamdba Architecture do MQ's most resemble?

Add a mids container to our docker cluster, publish "real" messages in json format, subscribe / consume the messages

We will be using the kafkacat utility.

For more information about kafkacat here is a link to the documentation https://github.com/git-hulk/kafka-cat

Using vi, edit your docker-compose.yml file to match the following contents:

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    expose:
      - "2181"
      - "2888"
      - "32181"
      - "3888"
    #ports:
      #- "32181:32181"
    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
  volumes:
    - ~/w205:/w205
  expose:
    - "9092"
    - "29092"
  extra_hosts:
    - "moby:127.0.0.1"

mids:
  image: midsw205/base:latest
  stdin_open: true
  tty: true
  volumes:
    - ~/w205:/w205
  extra_hosts:
    - "moby:127.0.0.1"
```

Download the following file of json data:

```
curl -L -o github-example-large.json https://goo.gl/WewtYn
```

Startup the cluster:

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

The cluster is getting bigger with more dependencies and may take a while to come up. We will look at the kafka logs to see kafka come up. The -f option tells it to hold on to the command line and output new data as it arrives in the log file. We can end this by using a control-C

```
docker-compose logs -f kafka
```

Create a topic called foo:

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

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

```
docker-compose exec kafka kafka-topics --create --topic foo --partitions 1 --replication-factor 1 --if-r
```

Which should show something like:

```
Created topic "foo".
```

Check the topic:

```
docker-compose exec kafka \
  kafka-topics \
    --describe \
    --topic foo \
    --zookeeper zookeeper:32181
```

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

```
docker-compose exec kafka kafka-topics --describe --topic foo --zookeeper zookeeper:32181
```

Which should show something like:

```
    Topic:foo   PartitionCount:1    ReplicationFactor:1 Configs:
    Topic: foo  Partition: 0    Leader: 1    Replicas: 1  Isr: 1
```

Let's use jq on the linux command line to examine the json files. Also try looking at them using vi.

```
docker-compose exec mids bash -c "cat /w205/github-example-large.json"
docker-compose exec mids bash -c "cat /w205/github-example-large.json | jq '.'"
docker-compose exec mids bash -c "cat /w205/github-example-large.json | jq '.[]' -c"
```

Publish some messages to the foo topic in kafka using the kafkacat utility:

```
docker-compose exec mids bash -c "cat /w205/github-example-large.json | jq '.[]' -c | kafkacat -P -b ka
```

Which should show something like:

```
Produced 100 messages.
```

Subscribe / consume the messages from the foo topic in kafka using the kafka console consumer utility in the kafka container:

```
docker-compose exec kafka \
  kafka-console-consumer \
    --bootstrap-server kafka:29092 \
    --topic foo \
    --from-beginning \
    --max-messages 42
```

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

```
docker-compose exec kafka kafka-console-consumer --bootstrap-server kafka:29092 --topic foo --from-begi
```

Alternatively, we can use the kafkacat utility in the mids container to subscribe / consume the messages:

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

We can also pipe this into a word count for the lines to see how many messages:

```
docker-compose exec mids bash -c "kafkacat -C -b kafka:29092 -t foo -o beginning -e" | wc -l
```

Tear down the cluster:

```
docker-compose down
```