Additional Notes/Hints for local kafka setup

Local Kafka Cluster

- A good resource for setting up a local kafka cluster using docker-compose can be found here: https://hub.docker.com/r/bitnami/kafka/. It also describes how to access the cluster from a client application.
  Ready to use example of a docker-compose.yml

```
version: "3"
services:
  zookeeper:
    image: 'bitnami/zookeeper:latest'
    ports:
      - '2181:2181'
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes
  kafka:
    image: 'bitnami/kafka:latest'
    ports:
      - '9092:9092'
    environment:
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_BROKER_ID=1
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CLIENT:PLAINTEXT,EXTERNAL:PLAINTEXT
      - KAFKA_CFG_LISTENERS=CLIENT://:9092,EXTERNAL://:9093
      - KAFKA_CFG_ADVERTISED_LISTENERS=CLIENT://kafka:9092,EXTERNAL://localhost:9093
      - KAFKA_CFG_INTER_BROKER_LISTENER_NAME=CLIENT
    depends_on:
      - zookeeper
```

- To allow your dockerized application to speak with kafka you need to create a docker network or use the default one. Your application container must be part of the network for example by creating it with this statement.
  `docker run --network work_default <your application-image>`
- To create kafka topics, insert test data and consume data you can use the kafka-cli which is included in the example kafka image (`bitnami/kafka`).
- Useful kafka-cli commands are:
  - kafka-topics.sh
  - Kafka-console-producer.sh
  - Kafka-console-consumer.sh

Kafka python library

- A very good python kafka client is `python-kafka`
- Many code snippets for consuming and producing are part of their documentation https://pypi.org/project/kafka-python/