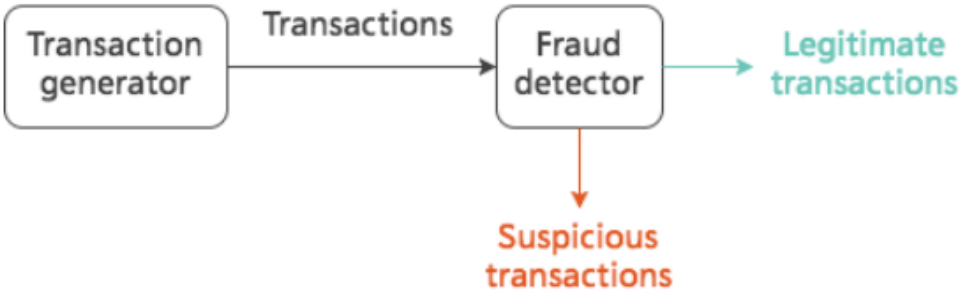
1. Introduction

The system we'll build is a simple fraud detection mechanism.



We'll produce fake transactions on one end, and filter and log those that look suspicious on the other end. This will involve:

- Transaction generator: an application that generates fake transactions and publishes them on a topic on the Kafka cluster.
- Defraud detector: a stream processing application that takes a stream of transactions as an input, performs some kind of filtering, then outputs the result into two separate streams — those that are legitimate, and those that are suspicious

Both applications will run in Docker containers and interact with our Kafka cluster.

2. System architecture

Basically, this systems consists of three components:

Kafka cluster	• It is used as a message broker, which receives and stores the messages that the generator sends, and from which the detector will retrieve and process the messages. • Set up locally using docker compose. Uses Docker images from Confluent Platform
Generator	• A python application which generates fake transaction messages and send them to the specific topic (‘queueing.transactions’) on the Kafka broker

Detector	• A python application which receives/consumes the messages that the generate have sent to the Kafa broker, identifies which are legal transactions and which are fraud ones, and then sends messages to different topics on the broker accordingly. .

3. Docker files and application code

1) Kafka cluster

Docker compose file: docker-compose_kafka.yml

```
version: '2.1'

services:
 zoo1:
   image: confluentinc/cp-zookeeper:7.1.1
   hostname: zoo1
   container_name: zoo1
   ports:
     - "2181:2181"
   environment:
     ZOOKEEPER_CLIENT_PORT: 2181
     ZOOKEEPER_SERVER_ID: 1
     ZOOKEEPER_SERVERS: zoo1:2888:3888



 kafka1:
   image: confluentinc/cp-kafka:7.1.1
   hostname: kafka1
   container_name: kafka1
   ports:
     - "9092:9092"
     - "19092:19092"
   environment:
       ……
   depends_on:
     - zoo1

# Give this composition access to the Kafka network
networks:
 default:
   external:
      name: kafka-network
```

2) Generator

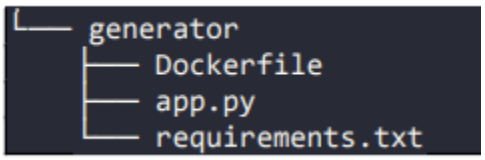
Related files are one Docker compose file and a subfolder with three files

❖ . Docker compose file: docker-compose-g.yaml

```
version: "3"
services:
   generator:
       build: ./generator
       #image: generator:latest
       environment:
         KAFKA_BROKER_URL: broker:9092
         TRANSACTIONS_TOPIC: queueing.transactions
         TRANSACTIONS_PER_SECOND: 1000

# Give this composition access to the Kafka network
networks:
 default:
   external:
      name: kafka-network
```

- Subfolder with three files. Subfolder structure:



- .Dockerfile: used to create a dockerized python application image for the generator

```
FROM python:3.6
WORKDIR /usr/app
ADD ./requirements.txt ./
RUN pip install -r requirements.txt
ADD ./ ./
CMD ["python3", "app.py"]
```

- Requirements.txt: used to specify Python packages that need to be installed in the image.

```
# requirements.txt
kafka-python
```

- App.py: the python application that generates fake messages and send to Kafka broker

```
…
TRANSACTIONS_TOPIC = os.environ.get("TRANSACTIONS_TOPIC")

KAFKA_BROKER_URL = "kafka1:19092"
TRANSACTIONS_PER_SECOND = 10
SLEEP_TIME = 1 / TRANSACTIONS_PER_SECOND

if __name__ == "__main__":
   print('TRANSACTIONS_TOPIC:{}'.format(TRANSACTIONS_TOPIC))
```

```
    producer = KafkaProducer(
        bootstrap_servers=KAFKA_BROKER_URL,
        # Encode all values as JSON
        value_serializer=lambda value: json.dumps(value).encode(),
    )
    i = 0
    while (i < 21):
        transaction = create_random_transaction()
        producer.send(TRANSACTIONS_TOPIC, value=transaction)
        # print(TRANSACTIONS_TOPIC)
        print(transaction)  # DEBUG
        sleep(SLEEP_TIME)
        i += 1
```

3) . Dectetor

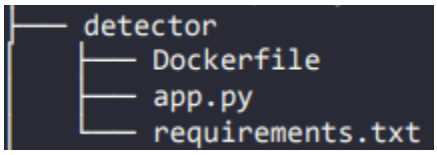
Related scripts are a Docker compose file and a subfolder with three files.

- ❖ Docker compose file: docker-compose-d.yam

```
version: "3"
services:
   generator:
       build: ./generator
       #image: generator:latest
       environment:
         KAFKA_BROKER_URL: broker:9092
         TRANSACTIONS_TOPIC: queueing.transactions
         TRANSACTIONS_PER_SECOND: 1000

# Give this composition access to the Kafka network
networks:
 default:
   external:
      name: kafka-network
```

- ❖ Subfolder with three files. Subfolder structure:



- .Dockerfile: the content is the same as that of the generator
- Rrequirements.txt: the content is the same as that of the generator
- App.py: The python application for receiving and processing the messages

```
# Get the  topic name from os environment, which are defined in the Dockerfile
TRANSACTIONS_TOPIC = os.environ.get("TRANSACTIONS_TOPIC")

KAFKA_BROKER_URL = "kafka1:19092"
LEGIT_TOPIC ="legit_topic"
```

```
FRAUD_TOPIC ="fraud_topic"


def is_suspicious(transaction):
   """Determine whether a transaction is suspicious."""
   return transaction["amount"] >= 900

if __name__ == "__main__":
   print(' hello !')
   print(TRANSACTIONS_TOPIC)

   consumer = KafkaConsumer(
   TRANSACTIONS_TOPIC,
   bootstrap_servers=KAFKA_BROKER_URL,
   value_deserializer=lambda value: json.loads(value),
   )
   producer = KafkaProducer(
   bootstrap_servers=KAFKA_BROKER_URL,
   value_serializer=lambda value: json.dumps(value).encode(),
   )
   print(type(consumer))
   for message in consumer:
       transaction: dict = message.value
       topic = FRAUD_TOPIC if is_suspicious(transaction) else LEGIT_TOPIC
       print(topic, transaction)
       producer.send(topic, value=transaction)
```