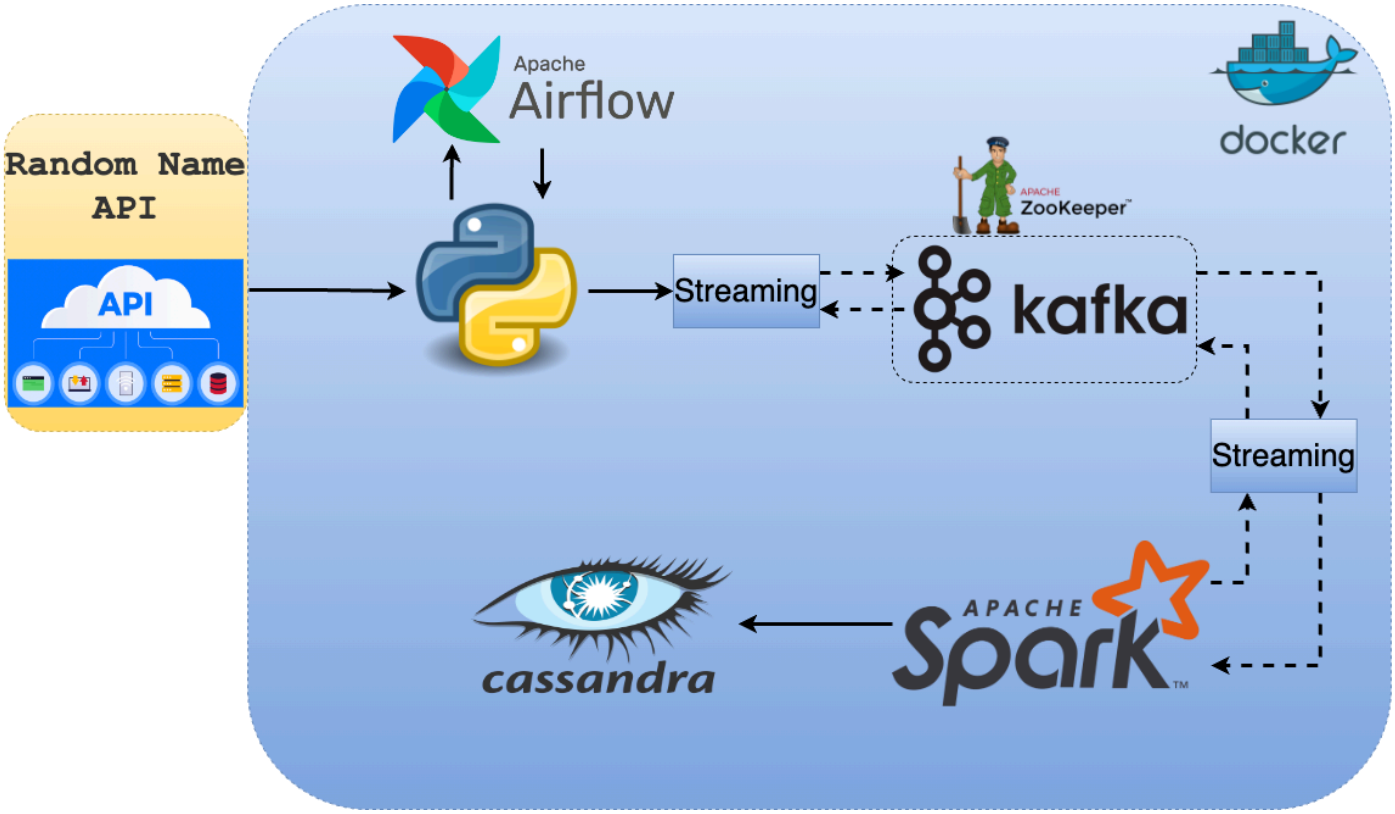
Kafka Use Case

Kafka Use case to Ingest data

By Dhandapani Yedappalli Krishnamurthi Aug 21, 2025

Tech Stack

1. Python - Programming Language
2. API - Data Source
3. Apache Airflow - ETL orchestration
4. Apache Kafka - Message Broker
5. Apache Spark - Cluster Compute Engine
6. Apache Cassandra - No- SQL storage (Data Warehouse)
7. Docker - Container



Kafka in 100 Seconds

Data Source: - https://randomuser.me/
Python Script:- Python script
Airflow DAG:-
https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/stream_to_kafka_dag.py

Spark:-
https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/spark_streaming.py

Streaming:
https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/spark_streaming.py

Docker:

https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/docker-compose.yml

Step By Step:

Step 1:

```
git clone https://github.com/dogukannulu/docker-airflow.git
```

```
Step 2:
```

```
docker build --rm --build-arg AIRFLOW_DEPS="datadog,dask" --build-arg
PYTHON_DEPS="flask_oauthlib>=0.9" -t puckel/docker-airflow .
```

```
Step 3:
```

```
docker-compose -f docker-compose-LocalExecutor.yml up -d
```

```
Step 4:
      Run Airflow

      https://localhost:8080
```

```
Step 5:

      docker exec -it <airflow_container_name> /bin/bash
curl -O <https://bootstrap.pypa.io/get-pip.py>
sudo yum install -y python3 python3-devel

python3 get-pip.py --user

pip3 install <list all necessary libraries here>
```

Step 5:

Multinode Kafka Cluster

https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/docker-compose.yml

Pre-Requisites:

It has all the necessary services:
Kafka, Zookeeper, Kafka-Connect, Schema-Registry, and Kafka-UI.

Step 6:

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

Step 7:

```
docker exec -it cassandra /bin/bash
```

Step 8:

```
cqlsh -u cassandra -p cassandra
```

Step 9:

```
CREATE KEYSPACE spark_streaming WITH replication =
{'class':'SimpleStrategy','replication_factor':1};
```

Step 10:

```
CREATE TABLE spark_streaming.random_names(full_name text primary key,
gender text, location text, city text, country text, postcode int, latitude
float, longitude float, email text);
DESCRIBE spark_streaming.random_names;
```

Running DAGS

https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/stream_to_kafka.py

https://github.com/dogukannulu/kafka_spark_structured_streaming/blob/main/stream_to_kafka_dag.py

Reference:-

https://medium.com/@dogukannulu/data-engineering-end-to-end-project-1-7a7be2a3671