# **Mistplay Data Engineer Take Home Challenge**

Three Main sections in the document

1: Top level Design discussion of the whole solution

2: Dockerizing the Main design approach

3: Data checks at the consumer side, at the transformation layer, use of avro schemas, encoding , Dead letter queues to handle unwanted records, Backfilling

4: The amount of implementation Completed, how to run the solution

5: Answers to Analytics question

**Top Level Main Design**



**1)** On Local machine Mac with below tools and techs installed:-

Spark

Kafka

Python

Anaconda Spyder

Airflow

Docker

**2)** Clone the fork repo with git and create a feature branch

Then the design is divided in to modules independent of each other , starting with Kafka layer

**Kafka Layer**

**3)** Since i want to stream the data from the files

Start Zookeeper

Start kafka Server

Created 5 Topics (Customer, Customer\_Extended, Sales, Refund, Product)   
(In the Producer.py file)

Kafka is all set and running.

**4)** Next I want to Publish my 5 text file data to kafka using Kafka-Producer()

5 files published to Customer, Customer\_Extended, Sales, Refund, Product Topics

**5)** To execute the step-4, i wrote python script **Producer.py** which will read the text files and pushes it to Kafka-Producer using python Library (KafkaProducer of kafka).

**6)** After Step-5 the data’s are streaming via respective topics.

**Spark streaming Tranformation Layer**

**7)** Now I want to capture these data , do some transformations and load it on staging layer.

**8)** To execute Step-7 I used spark-Structured-Streaming (Spark-Consumer) to capture it , for this i used Pyspark(python Library) in **Consumer.py** script

**10)** Now my records are ready in staging tables , i want to write some sql queries and want to see some insights using charts.

**Analytics Layer**

**11)** To execute Step-10 i used pandas/SQL to get insights and answer the questions

**12)** Total 3 Python Jobs.

→ One for Kafka-Producer (JOB1)

→ One for spark-Streaming consumer (JOB2)

→ one for Sql queries and insights (JOB3)

**Dockerizing the whole solution**

Below are the Docker(**docker-compose.yaml** file) Containers i am using it:

**Postgresql for Airflow db , Airflow , Zookeeper for Kafka server, Kafka Server, Spark**

We will use Airflow to trigger the 3 jobs mentioned above.

Airflow will trigger JOB1 , JOB2 , JOB3 using airflow dags,

Now to get the better clarity I will make 3 DAGs to trigger 3 jobs

1: DAG1(data\_stream\_DAG.py), This will trigger the job JOB1 of producer

2: DAG2(spark\_consumer\_DAG.py), This will trigger the job JOB2 to consume data and load into tables.

3: DAG3(insight\_DAG.py), This will trigger the job JOB3 to show insights using charts.

**so now we have 4 container in total :-**

Container 1 (Postgresql for Airflow db)

Container 2 (Airflow + KafkaProducer + Spark\_hadoop)

Container 3 (Zookeeper for Kafka server)

Container 4 (Kafka Server)



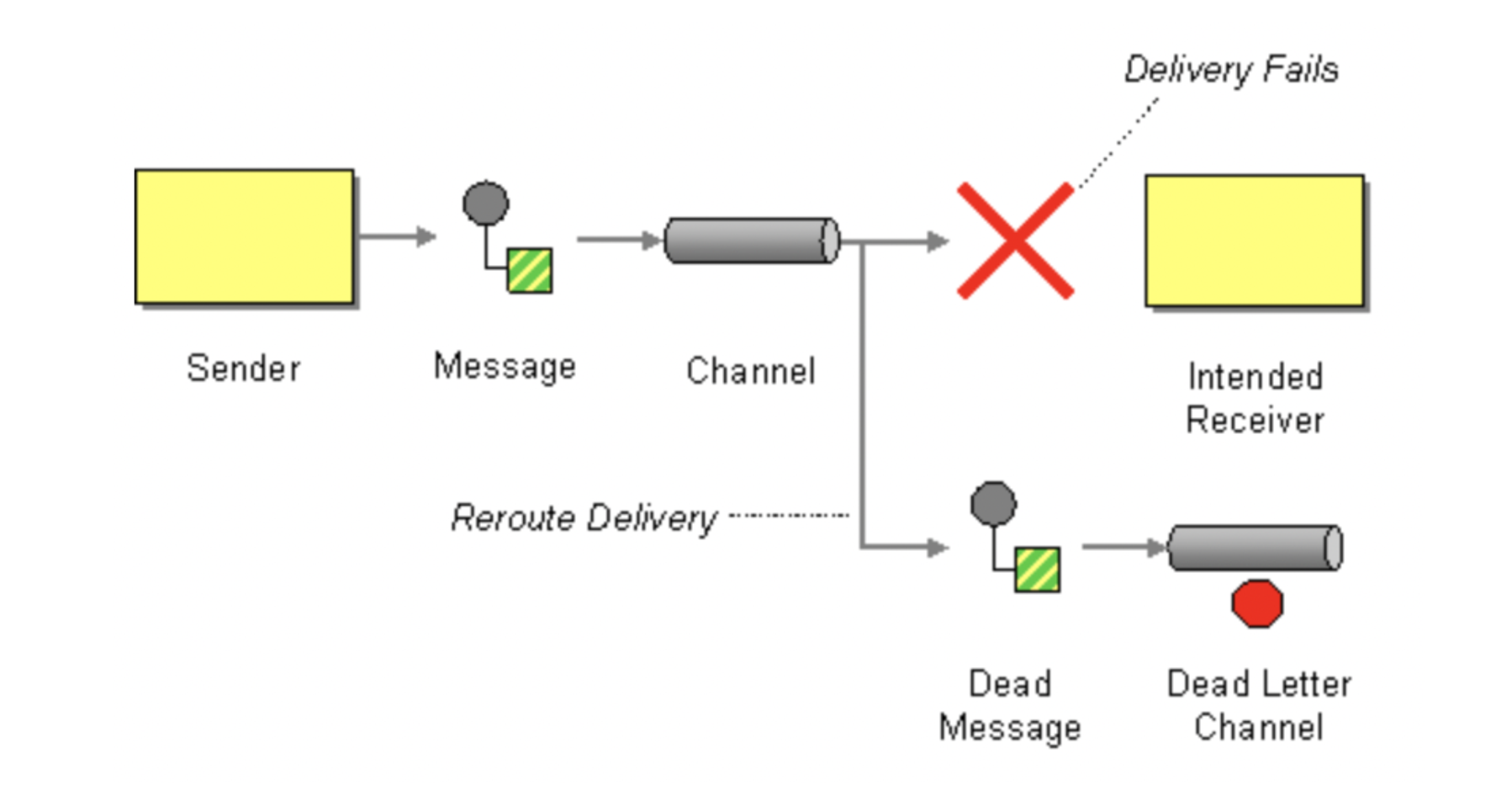
**Data Quality checks, Dead letter queues , backfilling**

1. **Data quality checks start off with the Consumer side and handling error events**- There is usually a contract between Publisher and Consumer that the events coming in have to be in an exact schema structure.

We can attach a Avro specific schema to the consumer (In here spark streaming consumer) , when the events published in the topic follows this structure we allow them to pass on to the staging layer else they can be passed to DLQ (dead letter queue topic/table)



DLQ example



*The current implementation i completed has a Schema check in the consumer but no DLQ implementation*

1. **Data quality checks at the Transformation side**

We can keep some tests at the staging/transformation layer to see some checks example like below

*1. Compare record counts between source and target that checks for any rejected records.*

*2. Checks on truncated data to make sure that data should not be truncated in the column of the target table.*

*3. Check whether unique values are loaded into the target. No duplicate records should be existing.*

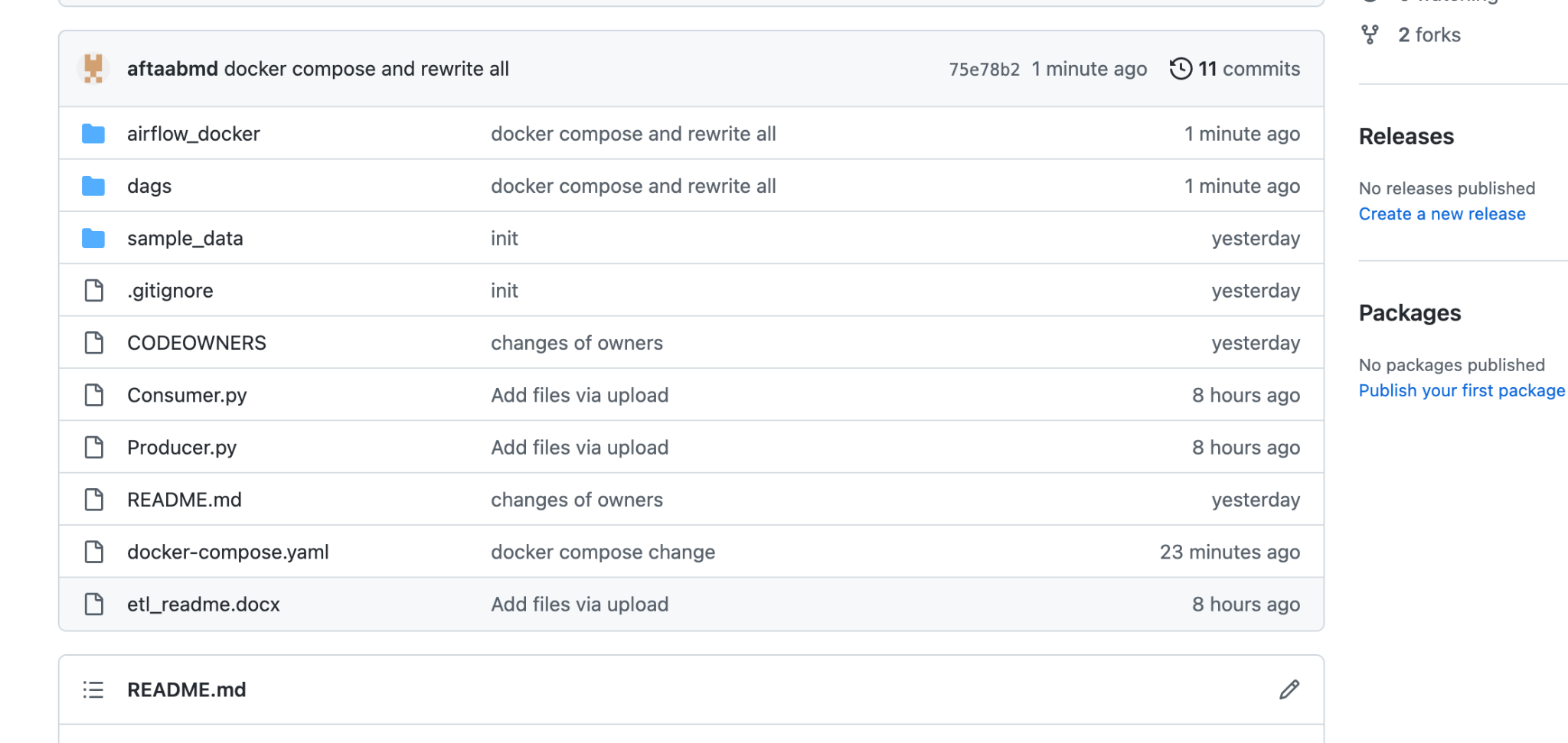
*4. Check boundary value analysis (ex: only >12-31-2018 year data has to load into the target)*

1. **Backfilling :**

We trace back the events in DLQ tables and check the issue with the error records, reprocess them and send it to kafka topic again at the producer side

**The amount of implementation Completed, how to run the solution**

**Structure of the Repo**



**airflow\_docker** – All the images file required for 4 containers (**Complete**)

**dags** - 3 dags for 3 jobs (**Incomplete**)

**Analytics.ipynb –** jupyter notebook to run pandas analytics scripts

**dags/src/spark\_consume\_data** – Consumer.py (Consumer spark to read 5 topics – **incomplete**)

**dags/src/spark\_consume\_data** – Analytics.py (Pandas python to solutions of the questions - In**complete ( few questions due to some clarification needed on data**)

**dags/src/stream\_data** – Producer.py (Producer to write 5 text files to 5 topics – **complete**)

**dags/src/stream\_data** – Producer.py (Producer to write 5 text files to 5 topics – **complete**)

**How to run the solution**

Start of with making sure all the ports are available for containers

Make sure your docker is running (If using mac)

This will install all the containers

**docker-compose up -d**

**Answers to Analytics question**

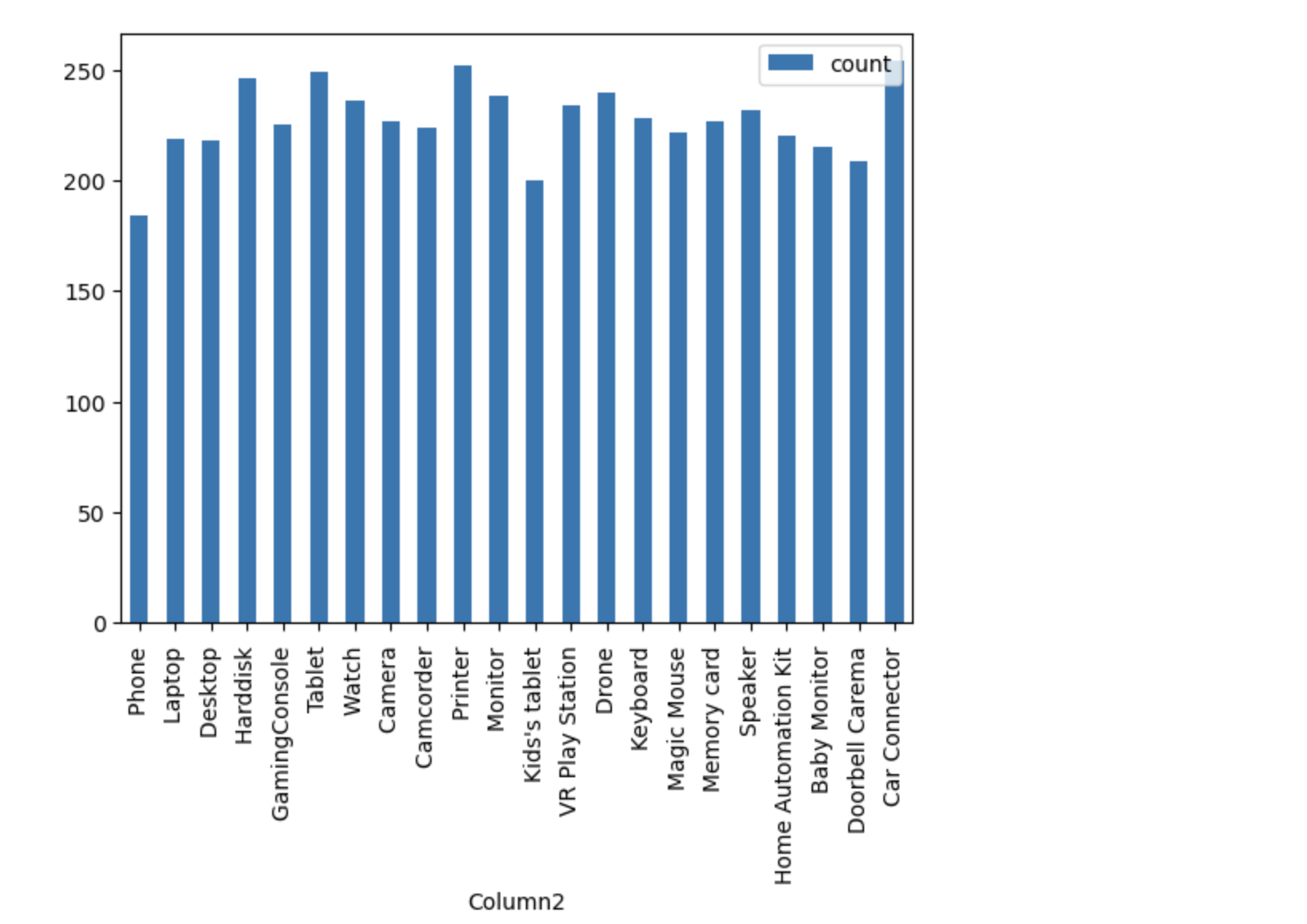
There are few Main details missing for the analytics task , what are the column names, in extended customer there are multiple address so which address is customers address

1. Write down the data quality issues with the datasets provided and the steps performed to clean (if any).

**The column names are missing for the data provided**

**Used pandas dataframe to read the text files with separators**

1. Display the distribution of sales by product name and product type.

****

1. Calculate the total amount of all transactions that happened in year 2013 and have not been refunded as of today.

1. Display the customer name who made the second most purchases in the month of May 2013. Refunds should be excluded.
2. Find a product that has not been sold at least once (if any).

**All products have an entry in sales list. So No products not sold**

1. Calculate the total number of users who purchased the same product consecutively at least 2 times on a given day.
2. Display all the details of a customer who is currently living at 1154 Winters Blvd.

**815008|MCBRIDE|BURNETT|14152996656|14156480309|Male|1154 WINTERS Blvd|CHATSWORTH|CA|USA|93635|2251 HART Blvd|PENNGROVE|WA|USA|96861|2694 HURST Blvd|GLENDALE|TX|USA|92017|SPEARS@hotmail.com|BISHOP@gmail.com|PATEL@twitter.com|RIVERS@facebook.com|MAHONEY@linkedin.com**