Title: Data pipleline architecture

Context:

We are working for a client in the E-Commerce industry who has ~20 online shops, daily visited by around 1 million customers. Our Data Engineering Team is currently working together with a Data Science Team on a recommendation engine.

Decision:

Kafka will be used as the temporary persistence layer for the data pipeline.

Reasoning:

kafka was used as a middleware between synchronus communicators so that we can achieve the benefits of Asynchronus Communication.It provide us the following advantages

- Fault-tolerance which we needed as required by Product team as no data point is to be lost.
- Low latency and high throughput which again is much needed.
- Scalibility is relatively very easy to achieve in Kafka as it can be scaled horizontally.

Consequences:

- Need to set up a Zookeeper service and a Kafka service so that data could be publised/subscribed to a kafka-topic with the required specified details.
- Ensure Kafka service schdeuling to ensure Kafka-Cluster is available for access to any of the services at all times.

Follow-up:

Research and compare different Kafka-service deployment options to determine the best fit for our needs.That either we run it locally or run it on cloud using AWS service.

Decision:

Using Pyspark with Kafka-python to transform and write back the data to kafka-topic

Reasoning:

We will be using kafka-python embeded with pyspark to achieve an optimal transformation layer.

- Since we will be consuming data by kafka-consumer we will easily get the data simply by decoding the message instead of rather reading using pyspark function that requires aditionall steps like enforcing schema and extracting value column

- If we streamed the data from kafka topic using pyspark the df we would be getting are streaming objects.Meanwhile, kafka consumers would directly give us encoded JSON messages.
- Low latency and high throughput which again is much needed.
- Scalibility is relatively very easy to achieve using Kafka-consumer.
- Pyspark can be used as a transformation tool to manupilate the data.

Consequences:

- Need to set up a Pyspark service with Kafka-Python dependencies installed.
- Set up a multithreaded enviroment to run the different consumers concurrently for optimal results.

Decision:

Using AWS RDS instance as persistent storage layer for our Data pipeline.

Reasoning:
Scalability: It is easy to scale as it is a cloud service hence providing us all the advantages of cloud computing.

High availability and reliability: Amazon RDS is established in a way that it has fault-tolerance as it stores your data on multiple nodes.

Security: Amazon RDS provides a very secure enviroment for your database

Cost-effective: Since Amazon RDS provides a pay-as-you-go pricing method that allows you to pay only for the resources you use. This eliminates the need for up-front hardware investments and helps you optimize your costs.

Integration with other AWS services: integrating the database with other aws services for further processing is easy as it is easly integratable with the other AWS services

Consequences:

- Create an RDS instacne and simply connect it with my kafka-service