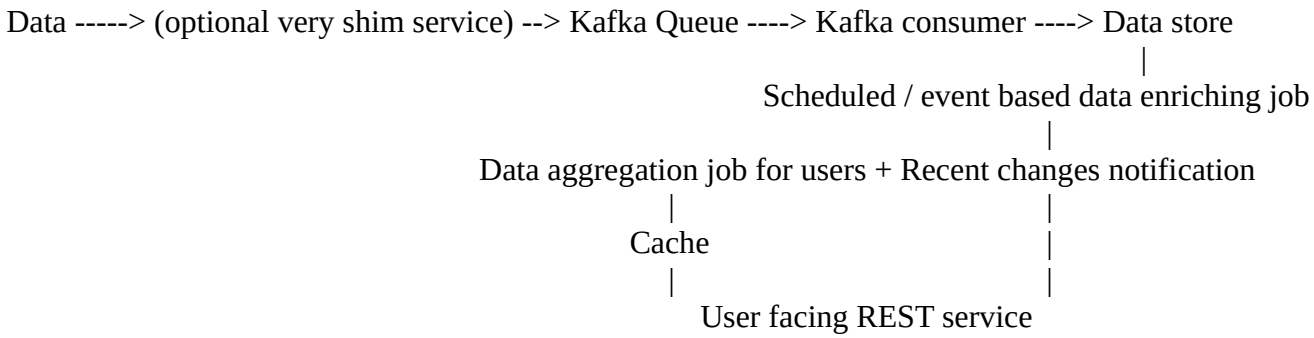
For this problem, I would design the following solution:



For Data ingestion,
My preference would be to receive the data in a queued buffer so that I can handle failures, high backpressure, and fault tolerance. For this I would prefer to use Kafka, as it is a suitable, production ready, battle tested tool.
Ideally I would ask my clients to integrate with kafka to push their data into it. If however, that isn't possible and they would prefer to send me data over REST / protobuf, I would build a shim service which quickly ingests the data and passes it on to Kafka

Data Store + Consumer:
Once the data is fed into kafka, I now need to consume it. For this, I'll set up a daemon which regularly consumes the data, and then writes the data in an appropriate schema on a distributed storage using columnar key value format.

I am assuming that a lot of the other data for enriching this data might be present here which might need to joined with this data. Possible candidates for such a store can be HDFS + Parquet, Cassandra, BigTable or Dynamo. I would use whichever of this is already in use for existing data. HDFS + Parquet has a slight edge over other solutions because it integrates best with Spark.

Data Enriching:

This depends on the type of enrichment we would expect. However, given the other assignment, if I assume that we want to enrich data using Spark, I would write a spark job to join and process the data. This can be scheduled periodically, or in micro batches using spark streaming. Also I would transform the data into denormalized data more ameanable to how users would consume it, and store the resulting data in another table.

Again, depending on the choice of data store, if this table is indeed in hadoop land, I would introduce an intermediate in-memory cache, which would store the results for the user. If the data store is something like Cassandra or a SQL database, I can query them directly.

Frontend:
I would build a REST service on Docker + Kubernetes which takes user requests, and then queries the data and provides the results.