Architecture

There will be driver and customer apps continuously sending their location and other details to our server. From the scalability perspective, when the number of customers and drivers increase in millions, one would require intermittent storage before processing the data. In the case when the processing node fails in the server, we would still have the data which is yet to be processed.

We will need a scalable stream processing solution which will be able to handle the stream processing in a fault-tolerant and efficient manner. After processing, we will have to save the data on some storage, so that it can be visualised whenever needed to understand the ongoing activity better. For that, we will need a storage solution and visualisation tool.

Kafka as Intermittent Storage

You have learnt Kafka, and you know Kafka offers good intermittent storage with horizontal scalability, fault-tolerance, wicked fast data save and retrieval capabilities. Due to these reasons, Kafka becomes our number one choice for intermittent storage.

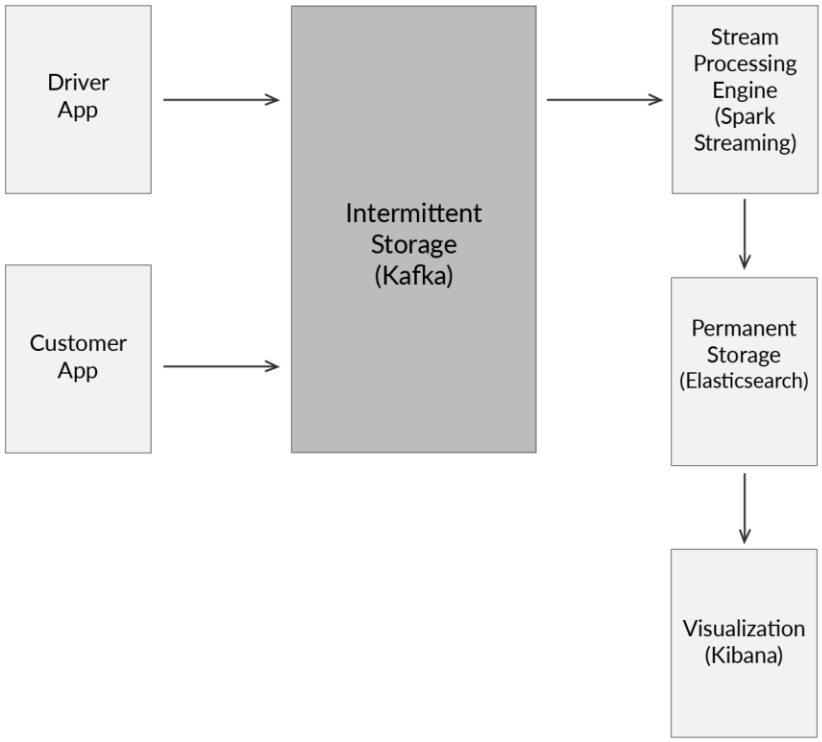
Millions of customer and drivers hitting API requests on the same server can lead to server crashes and data loss. Kafka will hold the data temporarily on its brokers until we have not processed the data.

Elasticsearch as Permanent Storage

We will use ElasticSearch as the permanent storage because it is good for storing, retrieving and managing document-oriented semi-structured data. We will be using Kibana as a visualisation tool to visualise the surge, on the map.

Spark Streaming as Streaming Processing Engine

Since Spark Streaming is a fault-tolerant, high throughput scalable stream processing engine, we will use it in our project to calculate the surge in real time. The following architecture would be a great approach for this problem statement.



Architecture