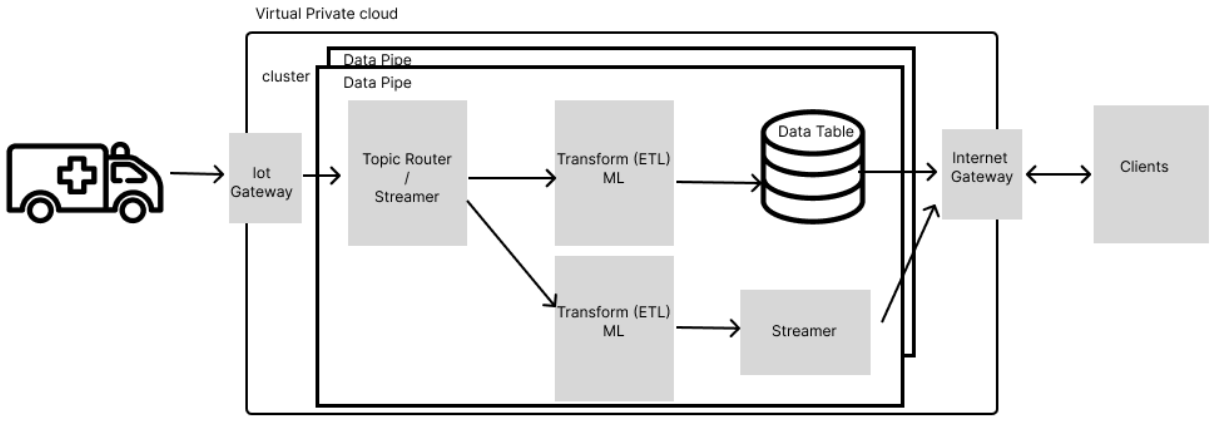
Ortus Data Pipeline Architecture

Jeff Morgan

Introduction.

This document proposes a framework architecture for a data services pipeline for IOT data transmitted from a fleet of vehicles in real time. The architecture needs to support efficient data ingestion and processing to enable analytics, business intelligence and dashboards to access the information in a form that is appropriate. This architecture also needs to address the security transport and access on the information ingested and exported.

Framework Architecture

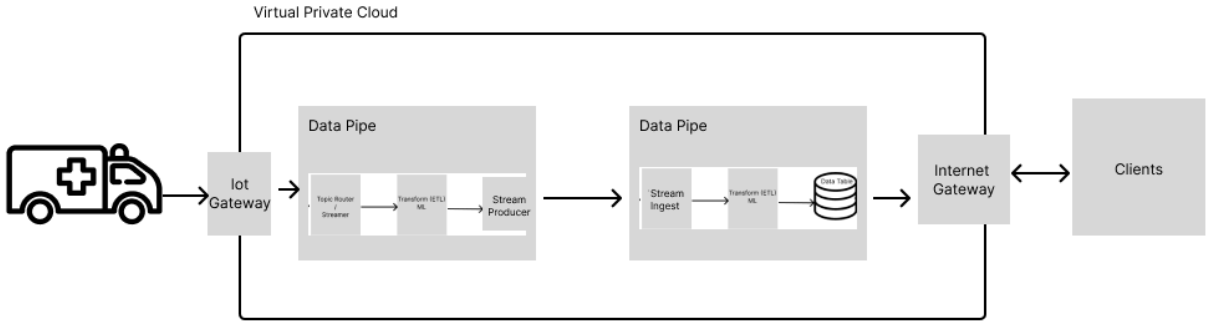


The figure above shows the general telematics data pipeline. The virtual private cloud contains most of the elements in the architecture and provides the security assurances required of such an architecture. Gateways provides access to and from the Virtual private cloud allowing only authorised clients with the correct credentials to access the cloud.

The IOT gateway allows data from the fleet of vehicles to be ingested into the data pipe securely using both encryption and authorised channels. Incoming data is organised into topics allowing many concurrent subscribers to read the incoming data and perform transforms before

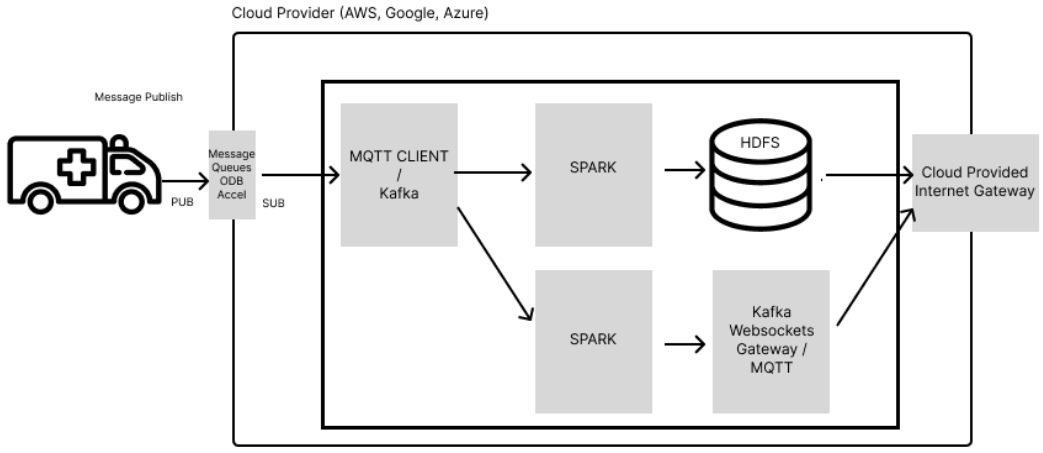
sending the data into a Data table for non real-time client access/archive or streamed to clients for real-time processing. All access to such data is performed via the Internet gateway which will only allow access from pre-authorised clients.

The Data Pipes running as a cluster of processes which means they can be arranged into chains support serial data processes or run as separate data paths supporting differing client needs



Technical details

In the scenario given the dataset is classed as On-Board Diagnostics (OBD) information. The information reflects the current operational characteristics of a vehicle as it performs various trips. Each vehicle is represented by a device ID (Int) and this is the only way to determine one vehicle from another.



Ingestion

Each vehicle will have a box which collects the OBD information and publishes this to the IOT gateway. Most major cloud providers (AWS, AZURE, Google) have an IOT gateway that can support data ingestion from IOT devices. One approach that is commonly used is a PUB/SUB model. In this was the IOT device periodically “Pub”lishes data packets to a queue from outside the cloud, and components inside the cloud can “Sub”scribe to that queue to obtain the data. This class of interaction is known as Asynchronous Message Queue Protocol (AMQP) and the most common implementation is known as MQTT.

With MQTT the IOT device cna publish a packet of data to a topic, and the data then can be read by subscribers. The topic can be a simple string (eg ‘ODB/update) it can contain identifying information such as the deviceId (eg. ‘ODB/update/deviceId). Subscribers can decide the level of filtering when receiving information. For example a subscriber might be interested in data from all vehicles so would subscribe to ‘ODB/update’ those which are concerned with a specific vehicle would also supply the deviceID of interest. The message associated with the PUB event from the IOT device is typically just a string. This could be a JSON String or a CSV string whatever is the easiest for the Publishing device.

For example a message sent to the topic ‘ODB/update/1’ with a JSON string would be


```
"{\"FIELD1\":0,\"timeStamp\":\"2017-10-3118:45:59.000000\",\"tripID\"
:0,\"accData\":\"ffb0fd700220fff43fffe74808e73e03f33400ef33fbe13bf9d0
4100d942feef36fedc4504e1430df43e09fc3902f73cfdf431fdea4002e437ffef34f
6de3df7d838edd640fccf49f7d83afdd93e08da46\",\"gps_speed\":2.2,\"batte
ry\":14.48,\"cTemp\":\"\",\"dtc\":\"\",\"eLoad\":\"44.313725490196084
\",\"iat\":\"\",\"imap\":\"\",\"kpl\":0,\"maf\":0,\"rpm\":903,\"speed
\":4,\"tAdv\":\"\",\"tPos\":\"15.686274509803923\",\"deviceID\":1},"
```


Any component in the cloud subscribing to ‘ODB/update/` would receive the above message and this would be passed on to the Transform.

Spark

Data is transformed using a Spark job. Spark Streaming has an MQTT adaptor which allows a Spark notebook to subscribe and receive data directly from MQTT, (note: This is also true of KAFKA). Therefore any spark notebook can Extract data from our Vehicles ODB packets for Transformation and then the result can be Loaded into a destination.

The destination could be a Table (HDFS) or a Stream using a KAFKA channel. It is likely that both would be needed for streaming to a live dashboard and for replaying a specific drive from a Table to review. The datatable will need a schema as described below in Hive format.

```
CREATE TABLE odb (deviceID INT, timeStamp Timestamp, tripID INT, accData
ARRAY<Double>, gps_speed INT, battery Double, cTemp INT, dtc INT, eLoad Double,
iat Double, imap Double, kpi Double, maf Double, rpm Double, speed INT, tAdv
INT, iPos Double) STORED AS ORC;
```

Optimised Row Columnar (ORC) is a typical storage format to support SQL access via the Internet gateway. A graphing tool such as graphina could be used to access this data table as part of an analytics dashboard

For the Streaming side of the operation there are a couple of choices depending on use case. The most obvious is a Kafka channel which is supported via Spark streaming as am output but this would need a second step to something like websockets for application access. The other option would be to return to MQTT and use Spark stream to publish to a topic that clients can subscribe to in order to display the live stream data.

Internet gateway

This provides access security for external clients such as applications and analytics. Most cloud service providers have such a component that allows paths into the virtual cloud network via a trusted model. Typically this is a combination of API keys and authentication, the latter is in general supported by the cloud provider however other SAAS services Auth0 could also be used. Trusted ports can also be opened and accessed by trusted applications, this is particularly useful so accessing MQTT brokers to stream information to applications

Cluster Management (Databricks?)

The one thing this architecture has avoided is that of cluster management, how are the data pipes deployed, scaled and managed, how are the spark notebooks loaded and where are they developed, stored and managed.

Clearly this is a massive omission that needs addressing. While much of the above could be handled with bespoke implementation and utilisation of services such as zookeeper and git there is another opportunity.

DataBricks is a product which could be used to implement this kind of architecture and it provides in effect the tools needed to do cluster management, and configure scaling policies, not only that but Databricks will enable the development, deployment of spark notebooks via git repos providing effective source code control.

The implementation of what has been described to this point with Databricks would add the cluster management and scalability required of a solution such as this.

What about Snowflake ?

The key difference here is that Snowflake is a SAAS whereas Databricks is PAAS. The platform approach provides a few key benefits, such as broader user appeal, extended data handling including unstructured data, key tools for development, testing and deployment. Therefore in conclusion Databricks would offer more and be more flexible to support this use case.