Logo, company name

Description automatically generatedText

Description automatically generated with low confidenceLogo, company name

Description automatically generated

Person

consumer

producer

**Kafka Topic**

Location

Location Processor microservice

Connection microservice

Person microservice

Location microservice

**gRPC**

**REST API deployed at port 30001**

**REST API deployed at port 30002**

Icon

Description automatically generated

**Webpage deployed at port 30000**

Udaconnect frontend

**gRPC solution for location processor microservices**

**Justification**: Since location will be heavily use for tracking the user coordinates , this portion need to be in efficiently structure as it might get a lot of incoming data at any instance of time.

Secondarily gRPC is chosen is because of strict type enforcement, as location is the most crucial piece . Message pass into the architecture is required to be checked and ensure it doesn’t break or give unexpected outcome. gRPC using protobuf able enforce strict type of data type pass into Udaconnect system.

**Efficiency** : gRPC uses HTTP/2 protocol that improves the overall security and performance.

**Type enforcement** : gRPC does have type enforcement to ensure the value is valid as protobuf message must have correct type.

Connection is to connect everything from the person location to date and time and person\_id. Location microservice is only to gather the topic link to location processor microservice, therefore for location processor microservice there will be more than 1 technology use (gPRC and kafka is used in location registry microservice).

**Kafka solution for location microservices and location processor microservice:**

**Justification**: Location data can be very heavy traffic from each device that sending to Udaconnect. In order for scalability on handling large volume of incoming location detail and backlog.

**REST API for person, connection and frontend microservices:**

As person ,connection and frontend microservices does not running streaming data nor intensive incoming data surge in a short time. REST API could be the most reliable solution.