DAQ Weather Station Implementation

Requirements

The requirements are very similar to the requirements of the current car logging system:

- Users need live access to the weather data on the frontend.
- All weather data must be stored.
- A high volume of data from the weather station needs to be processed.

Hence, we will use a similar setup to the car logging system.

Weather station connection protocol

The weather station will be sending data continuously and we would want lower latency as well, hence the WebSocket protocol is best to connect the weather station to AWS (same as car logging).

AWS compute for weather station

Firstly, same as the car logging system, we will use a load balancer that will send all the data to one service that handles storing the data, and samples of the data to another streaming service that serves this data live to users.

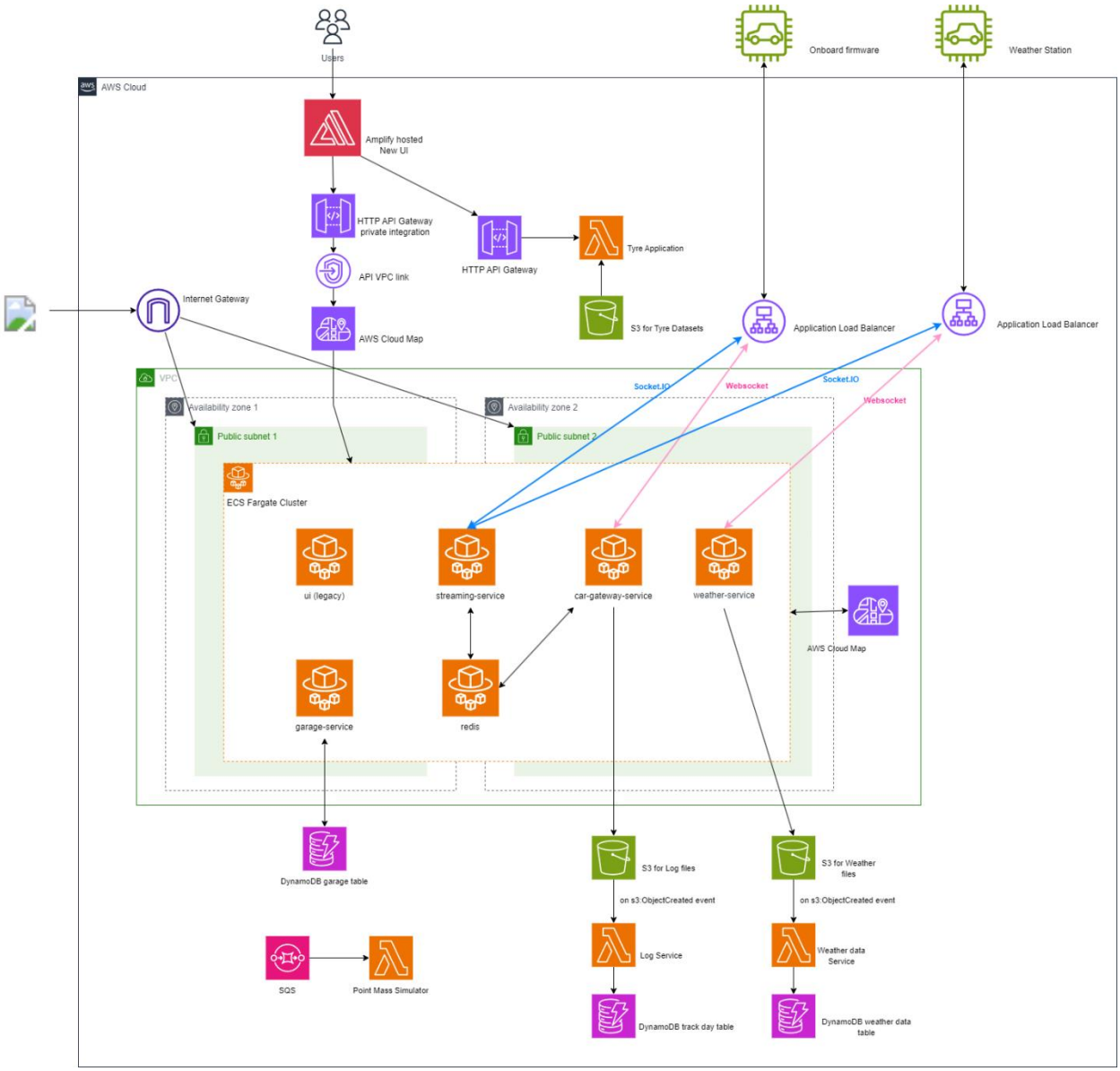
For the storing of the data, we can add a new containerised microservice called “weather-service” that can just be added to the Fargate cluster. This will handle a Websocket connection with the weather station (through load balancer), do processing on the data and then push the data to storage.

For the streaming service, we can use the existing infrastructure (“streaming-service” and the redis database) as it is already setup to serve data to front-end users fast through the HTTP API. Hence, we connect the streaming service to the load balancer to receive samples of weather data. Some changes might be required for the “car-gateway-service” depending on if it generates and stores logs off of the data in the redis database, as this would now also contain weather station data.

AWS storage for weather station

Assuming quite a lot of data will be produced by the weather station, we can again copy the storage setup used by the car logging system. We use an S3 bucket for bulk storage of files and use a DynamoDB table to store metadata and other relevant data that the front-end/ lambda functions will need to organize and find the correct weather data to serve to users. We can also copy the lambda function to automatically update the table when new objects are added to the bucket.

The updated draw.io diagram is as shown:



IaC Modifications

The modifications to the terraform code should be simple. For the ECS Fargate cluster, as Fargate takes care of all the EC2 instancing, load balancing etc, adding the weather-service should be simple. It should just require defining a new container with the “weather-service” container image under the ECS resource. Adding the S3 bucket, DynamoDB database and the miscellaneous lambda functions should be the same as already implemented for car logging, hence should be mostly just copy and paste.