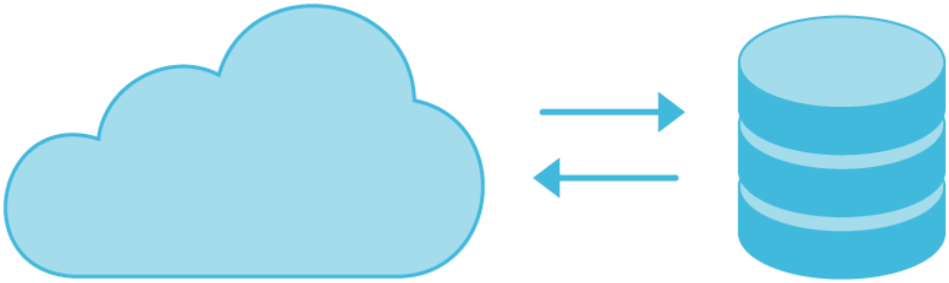


Cloud Technical Assessment

By Zachary Abran (z5589903)

0 Introduction

This PDF document will contain the submission for the Cloud Technical Assessment. The document will include both the Cloud Architecture Diagram and the implementation details for this system.

Problem Definition: DAQ wants to implement a local weather station collection system for the race engineers want to correlate local weather data with vehicle performance

1 Cloud Architecture Diagram - Theory

Weather System: Sensors that analyse air temp, humidity, track temperature, wind speed, and direction. A Docker Container may be run on hardware such as an edge device or Raspberry Pi.

Ingestion Layer: The weather station pushes data using MQTT/WebSockets into an AWS IoT Core to manage external access to services within a cluster (ingress), for authentication, and to ensure no loss of data. AWS IoT Core is recommended for its scalability & security.

Storage Pipeline: AWS Lambda is used as a bridge to transform and send the data from AWS IoT to the Redis Channel for real-time data. For historical data and long-term storage, older data will be moved from the Redis Channel into the S3 database and then, optionally, also into DynamoDB for structured queries.

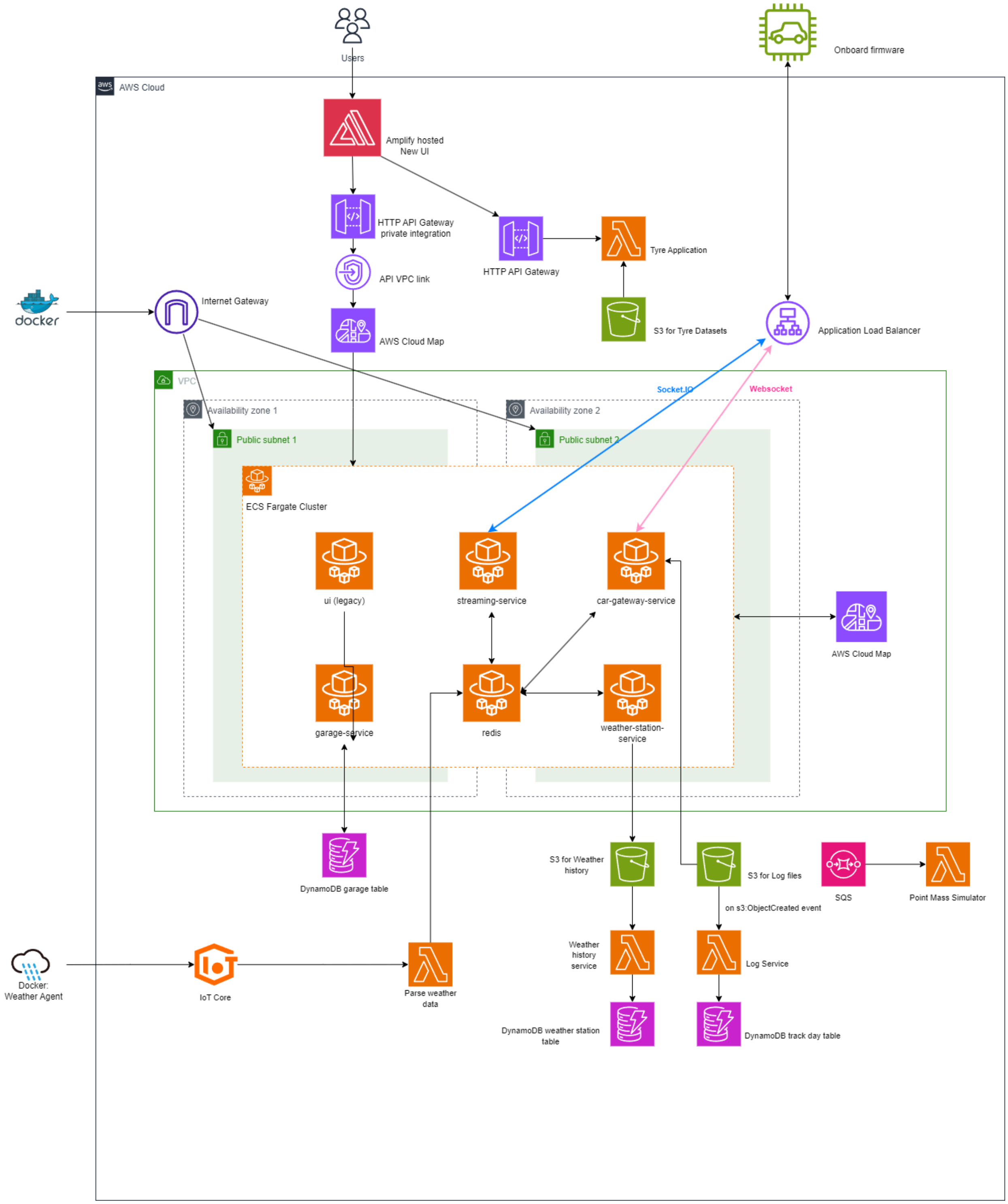
2 Implementation

As recommended in the considerations section, the system will use Terraform (Infrastructure as Code (IaC)) to deploy the system, and Docker to containerise applications.

The weather system can be deployed using Terraform by extending the Terraform codebase to include all the AWS resources required for the weather station integration, since Redback already uses it for its systems. Teraform can be used to create and deploy the AWS IoT Core setup, defining any IoT rules to forward incoming data to multiple destinations (Lambda, S3, DynamoDB). AWS Lambda functions can be deployed using the Terraform (aws_lambda_function) resource; these functions can process incoming weather data and publish it into Redis, DynamoDB, and S3. For data storage, Terrfarom can be used for S3 to create a new provision bucket with life cycle rules for the incoming weather logs, create a table for structured weather snapshots for the DynamoDB table, and deploy a Redis cluster for the real-time weather data. Assuming that the Terraform code is stored in GitHub, any changes (new bucket, Lambda, IoT rule) are redeployed consistently.

Docker can be used to implement the cloud ECS and potentially onboard firmware on the weather station, since Docker allows firmware to be easily run inside a Docker Container. Docker can package any sensor drivers and data publishing code from the weather system. In the Cloud, any new ingestion services for the data coming from the weather system can be deployed as Docker Containers in ECS, similar to how the current ingestion services run for the car. Docker Containers can be used to build, test, and push to Amazon ECR using the various Docker commands and then deployed to ECS via Terraform.

3 Cloud Architecture Diagram


Users
Onboard firmware
AWS Cloud
Amplify hosted
New UI
HTTP API Gateway
private integration
HTTP API Gateway
Tyre Application
API VPC link
docker
Internet Gateway
AWS Cloud Map
S3 for Tyre Datasets
Application Load Balancer
VPC
Socket.IO
Websocket
Availability zone 1
Availability zone 2
Public subnet 1
Public subnet 2
ECS Fargate Cluster
ui (legacy)
streaming-service
car-gateway-service
AWS Cloud Map
garage-service
redis
weather-station-service
DynamoDB garage table
S3 for Weather history
S3 for Log files
SQS
Point Mass Simulator
on s3:ObjectCreated event
Weather history service
Log Service
Docker: Weather Agent
IoT Core
Parse weather data
DynamoDB weather station table
DynamoDB track day table