# IOT Ingestion

Authors:

- Adam Edwards

## Overview

This document proposes the cloud infrastructure for the IOT ingestion backend.

## Problem statement

Project Echo plans to use IOT devices that are placed to record audio of animals, the location of these devices may be remote and the audio data they collect needs to be forwarded to the Echo Store. They will make use of the public internet so therefore require secure communication to the backend.

MQTT has been agreed upon as the most suitable communication protocol for sending / receiving the data.

## Considerations

### Assumptions

GCP project already setup with:

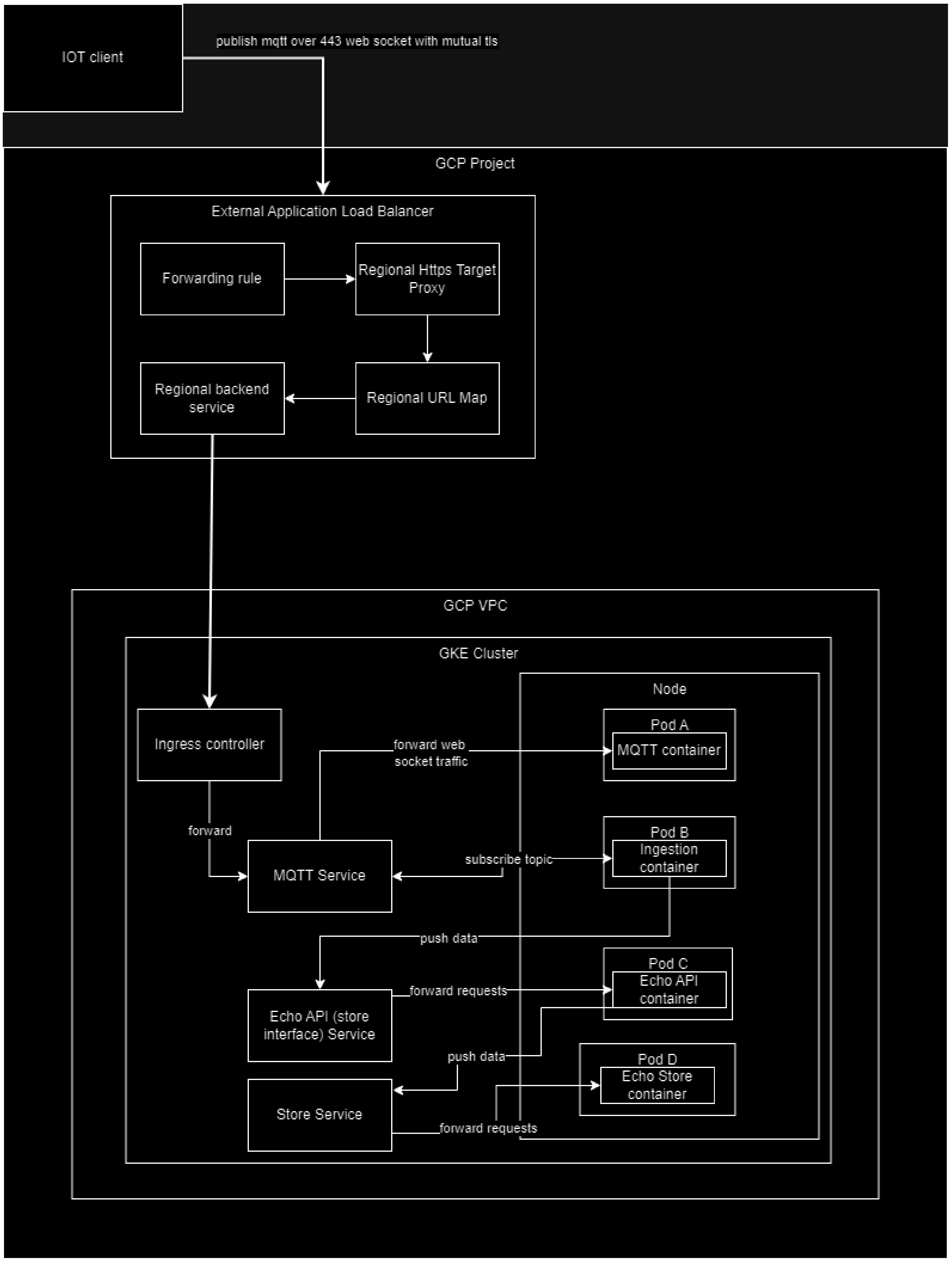
- VPC
- GKE cluster
- Echo API and store pods/services running on GKE cluster

### Security

As the MQTT service is going to be exposed to public internet, communication to the service must be locked down.

## Proposed Design

### Flow



## Sequence

1. IoT client subscribes/forwards requests to external domain of external application load balancer using websockets on 443 and mutual TLS.
2. Forwarding rule forwards to regional https target proxy, which finds forwards to regional url map which identifies the regional backend service. With the service identified the packet can be forwarded on to the GKE cluster.
3. The GKE cluster uses an ingress controller. The ingress controller forwards to the MQTT service which directs to the MQTT pod which has a container that runs the MQTT broker.
4. If a message has been published to the queue on the broker (IE the external traffic coming in wasn't to subscribe), the Ingestion container in Pod B is a subscriber, it will receive the message and handle it. The message is processed and a request made to the Echo API service.
5. Echo API handles the request and it connects to Store Service to load the data to store which uses MongoDB. It is expected that a network policy will be in place so that only the Echo API can interact with the Store Service.

## Security Discussion

SSL encryption to secure the payloads to the MQTT service to be used (though this is not critical) but mutual TLS is to be used to securely authenticate with the service. To keep things simple this is ideally handled by the external application load balancer. The external application load balancer can also act as TLS termination proxy, communication from it internally may be unencrypted.

The required secrets to be used are the private certificates and passkeys for the private certificates. These will not be stored in source code but will be built into the relevant docker images that are stored within GCP Artifact Registry. For storage they can be kept zipped and stored within Project Echo on Microsoft Teams.

## Reference Architecture

### GCP External Load Balancer

This is used to expose the service externally and provide for mutual TLS. Requests from it are forwarded on to the GKE cluster which has an Ingress Controller configured to handle the traffic. This should be a standard tier, regional load balancer.

### Google Kubernetes Engine

Project Echo is using Google Kubernetes Engine (GKE) to run its services including Echo Store, Echo API, so the MQTT and ingestion containers can run here.

### Google Cloud DNS

This is not shown within the flow diagram, but an external domain name is required. A domain name can be purchased from Google and hosted using Google Cloud DNS. A CNAME can then be created for the external application load balancer.

### HiveMQ Docker Image

This provides the MQTT broker service running as a container within a Kubernetes pod.

https://hub.docker.com/r/hivemq/hivemq4/

### Python Paho MQTT client library

Paho mqtt library can be used to subscribe to the MQTT topic ran by the MQTT broker and httpx / requests can be used to perform requests to send data Echo API to be put into Echo Store; these will be the building blocks for the ingestion service/container written in Python.

## Reference Links

https://cloud.google.com/load-balancing/docs/https#component

https://hub.docker.com/r/hivemq/hivemq4/

https://pypi.org/project/paho-mqtt/

https://cloud.google.com/domains/docs/overview

https://cloud.google.com/domains/docs/buy-register-domain