7.1 Message queue

Quick instructions

The services start up as specified in the exercise. No operations aside from `docker-compose build` and `docker-compose up` are required.

Host version information

- **OS**: Darwin macbook 22.1.0 Darwin Kernel Version 22.1.0
- **Docker**: Docker version 20.10.20, build 9fdeb9c
- **Docker Compose**: Docker Compose version v2.12.1

Overview

The microservices were built with Python and use the pika client for AMQP 0-9-1 messaging. The RabbitMQ readiness problem was solved by implementing a polling function for the RabbitMQ service's /metrics endpoint, which was found to be a good indicator of the service's readiness. The microservices go through the same polling loop before proceeding with their own logic; a fail system was also implemented where the microservices exit if RabbitMQ fails to start up within the defined limits. Additionally, the `origin` service waits until one of the queues declared by the `observer` service exists, making sure that the listener is active (a separate check could have also been made for `intermediate`, but the current configuration was found to be stable enough).

The microservices are (depending on service) configurable with environment variables sourced from `docker-compose.yml`. An exception to this is the HTTP server (built on nginx), which is configured with a static configuration file.

Benefits of topic-based communication

In my opinion, topic-based communication has significant benefits in decoupled systems of many components. With a message broker as a decoupling tool, individual application components can be updated, restarted, patched et cetera with minimal impact to other components.

Queue- and topic-based architectures have another strength in scaling problems – when a specific queue seems to become a bottleneck, the listener can simply be replicated and added as another listener to the same queue.

Main learnings

This was my first time working with RabbitMQ (and AMQP as a whole), and it certainly was a learning experience. Originally, I misunderstood some foundational things on exchanges, topics and queues and ended up with a buggy set of microservices. After tedious debugging however, I think I got a hang of the protocol. I am satisfied with the final product and don’t think there are any major issues within the code base. In particular, I enjoyed coming up with solutions to the synchronization problems and implementing the exercise with tools I am comfortable working with.