COMP.SE.140 - AMQP Exercise

This exercise was implemented in Mac OS with docker desktop. However, the same exercise was tested in ubuntu server.

Mac OS - 10.14.6

Docker version - 20.10.5, build 55c4c88

Docker compose version – 1 28.5

Uname –a - Darwin Chaliths-MBP.lan 18.7.0 Darwin Kernel Version 18.7.0: Tue Aug 20 16:57:14 PDT 2019; root:xnu-4903.271.2~2/RELEASE_X86_64 x86_64

Highlighted points and main learning

Docker volume was used to store the logs in OBSE server and get the logs for HTTPSERVE.

To handle the containers startups and waiting until RabbitMQ fully running, tried several methods. However, some of them did not work with the mac and docker compose version.

1. Using an external scrip t– This was the method which was stated in the official docker docs (https://docs.docker.com/compose/startup-order/) I also used the same method in this exercise. However, I found some versions incompatibilities with this method.
2. Using **depends_on** property - According to this (), this method supports version 2 compose YMLs.
3. Using **restart: on-failure** property. Without external scripts, I tried this method but faced some issues.
4. depends_on : consition - “service_completed_successfully” Tried this method in new updated version but did not get the expected behavior.

According to my further analysis I found out that it might be better to handle this kind of scenario from application level as well. As an example, if the nodejs script can check the host availability and retry until it is reachable, then we might not need to handle this in docker compose specifically.

Benefits of topic-based communication

- Provide queue and store the messages
- Host does not need to think about client (another end) and it is only passing the request to rabbitmq topic
- Multiple clints and multiple hosts
- Can use fanouts, meaning that one message will be enough to inform several different components, reducing the amount of communications.