UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Queue Service in the Cloud

Servicios y Aplicaciones Distribuidas

Máster Universitario en Ingeniería Informática

2020/2021

Diogo Jorge Freitas Cadavez
Irene Garcia do Amaral

Github

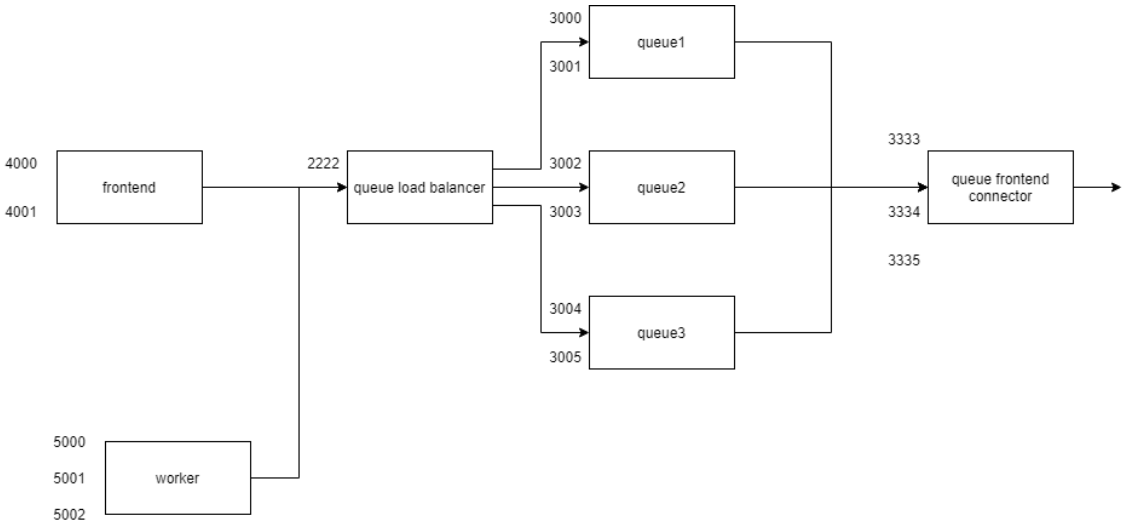
https://github.com/irenegarciaamaral/SAD_Project

Introduction

For this project we implemented a shopping cart API having into account the architecture that will be shown above.

All these systems were built using docker-compose to include all the docker containers needed. The docker containers communicate using node module 0MQ, the pattern used was PUSH/PULL, enabling the creation of a pipeline to transmit the data.

The Architecture



Graphic 1: Architecture of the distributed application

Microservices/Containers:

Frontend

```
1  FROM node:14.15.0
2  WORKDIR /usr/src/app/frontend
3  COPY ./service-registry/package*.json ./
4  RUN npm install
5  COPY ./service-registry .
6  CMD [ "npm","start"]
```

Figure 1: Docker file

Queue Load Balancer

This micro service receives the messages from the frontend and sends these messages to the three queues, alternating between them.

```
1  FROM node:14.15.0
2  WORKDIR /usr/src/app/queuelb
3  COPY ./queue-load-balancer/package*.json ./
4  RUN npm install
5  COPY ./queue-load-balancer .
6  CMD [ "npm","start"]
```

Figure 2: Docker file

Queue

The Queue sends the messages to the worker, and when the worker returns them to the Queue, they are forward to the Queue Frontend Connector.

Three instances of this module were used.

```
1  FROM node:14.15.0
2  WORKDIR /usr/src/app/queue
3  COPY ./queue-server/package*.json ./
4  RUN npm install
5  COPY ./queue-server .
6  CMD [ "npm","start"]
```

Figure 3: Docker file

Worker

The worker receives the messages and checks if they match to any valid request, and if so, perform the operations that are needed.

```
1  FROM node:14.15.0
2  WORKDIR /usr/src/app/worker
3  COPY ./worker-client/package*.json ./
4  RUN npm install
5  COPY ./worker-client .
6  CMD [ "npm","start"]
```

Figure 4: Docker file

Queue Frontend connector

This microservice forwards the messages from the queues to the frontend when these messages return from the worker.

```
1  FROM node:14.15.0
2  WORKDIR /usr/src/app/queueendcon
3  COPY ./queue-end-connector/package*.json ./
4  RUN npm install
5  COPY ./queue-end-connector .
6  CMD [ "npm","start"]
```

Figure 5: Docker file

Docker compose file

```yaml
version: "3.9"
services:
  queue:
    build: './Queue'
    ports:
      - "3000:3000"
      - "3001:3001"
    environment:
    - ZMQ_PUB_ADDRESS=tcp://queuelb:3000
    - ZMQ_WORK_BIND_ADDRESS=tcp://*:5000
    - ZMQ_PUB_WORKER_ADDRESS=tcp://worker:3001
    - ZMQ_FRONTEND_BIND_ADDRESS=tcp://*:3333
  queue2:
    build: './Queue'
    ports:
      - "3002:3002"
      - "3003:3003"
    environment:
    - ZMQ_PUB_ADDRESS=tcp://queuelb:3002
    - ZMQ_WORK_BIND_ADDRESS=tcp://*:5001
    - ZMQ_PUB_WORKER_ADDRESS=tcp://worker:3003
    - ZMQ_FRONTEND_BIND_ADDRESS=tcp://*:3334
  queue3:
    build: './Queue'
    ports:
      - "3004:3004"
      - "3005:3005"
    environment:
    - ZMQ_PUB_ADDRESS=tcp://queuelb:3004
    - ZMQ_WORK_BIND_ADDRESS=tcp://*:5002
    - ZMQ_PUB_WORKER_ADDRESS=tcp://worker:3005
    - ZMQ_FRONTEND_BIND_ADDRESS=tcp://*:3335
  frontend:
    build: './Frontend'
    ports:
      - "4000:4000"
      - "4001:4001"
    environment:
    - ZMQ_BIND_ADDRESS=tcp://*:2222
    - ZMQ_PUB_QUEUE_ADDRESS=tcp://queueendpoint:4001
    - APPID=4000
  worker:
    build: './Worker'
    ports:
      - "5000:5000"
      - "5001:5001"
      - "50002:5002"
    environment:
    - ZMQ_PUB_WORKER_ADDRESS=tcp://queue:5000
    - ZMQ_PUB_WORKER_ADDRESS_Q2=tcp://queue2:5001
    - ZMQ_PUB_WORKER_ADDRESS_Q3=tcp://queue3:5002
    - ZMQ_QUEUE_BIND_ADDRESS=tcp://*:3001
    - ZMQ_QUEUE_BIND_ADDRESS_Q2=tcp://*:3003
    - ZMQ_QUEUE_BIND_ADDRESS_Q3=tcp://*:3005
  queuelb:
    build: './QueueLB'
    ports:
     - "2222:2222"
    environment:
    - ZMQ_PUB_ADDRESS=tcp://frontend:2222
    - ZMQ_BIND_ADDRESS_QUEUE1=tcp://*:3000
    - ZMQ_BIND_ADDRESS_QUEUE2=tcp://*:3002
    - ZMQ_BIND_ADDRESS_QUEUE3=tcp://*:3004
```

```
64    queueendpoint:
65      build: './QueueEndCon'
66      ports:
67       - "3333:3333"
68       - "3334:3334"
69       - "3335:3335"
70      environment:
71      - ZMQ_PUB_ADDRESS=tcp://queue:3333
72      - ZMQ_PUB_ADDRESS2=tcp://queue2:3334
73      - ZMQ_PUB_ADDRESS3=tcp://queue3:3335
74      - ZMQ_BIND_ADDRESS_Frontend=tcp://*:4001
75    database:
76      image: mongo:latest
77      container_name: mongodb
78      ports:
79        - "27017:27017"
```

Future upgrades

There is a microservice include in the docker-compose file that has an image for MongoDB, this data base would communicate with the workers to handle its requests. Another Worker would also be added to the distributed application, in these situations one of the workers would be used to fetch data from the data base, and the other one to update data.