Flamingo Records

Small assignment V

A company called **Flamingo Records** are experiencing record-breaking traffic to their systems and are afraid they just cannot keep going on this way. They already hired a software developer to create a microservice structure for them, but they don't know how to deploy it! They recently heard that setting up microservices as **Docker containers** was the way to go, so they immediately contacted us because of our expertize working with containers!

Rules

There are some rules that apply for all applications running in containers:

1. All applications should be use of the **production** config when being run in a container
2. If you want to run services locally, you need to setup the proper **development** configuration. This means you have to setup your own instance of **RabbitMQ** server, **PostgreSQL** database and **MongoDb** database and provide the connection information within the configuration
3. **No code needs to be altered unless explicitly stated in the assignment description or within the code**

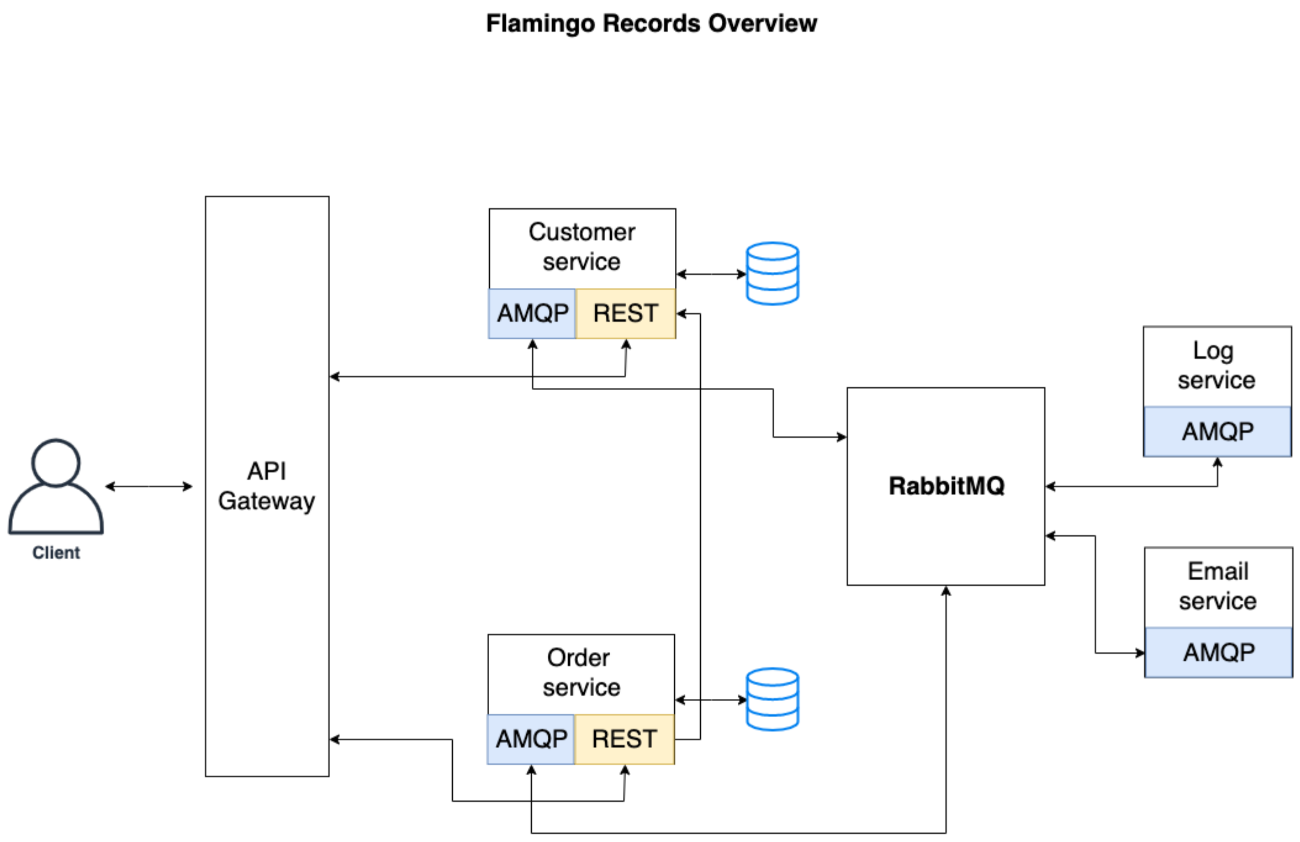
Assignment description

See assignment description below:

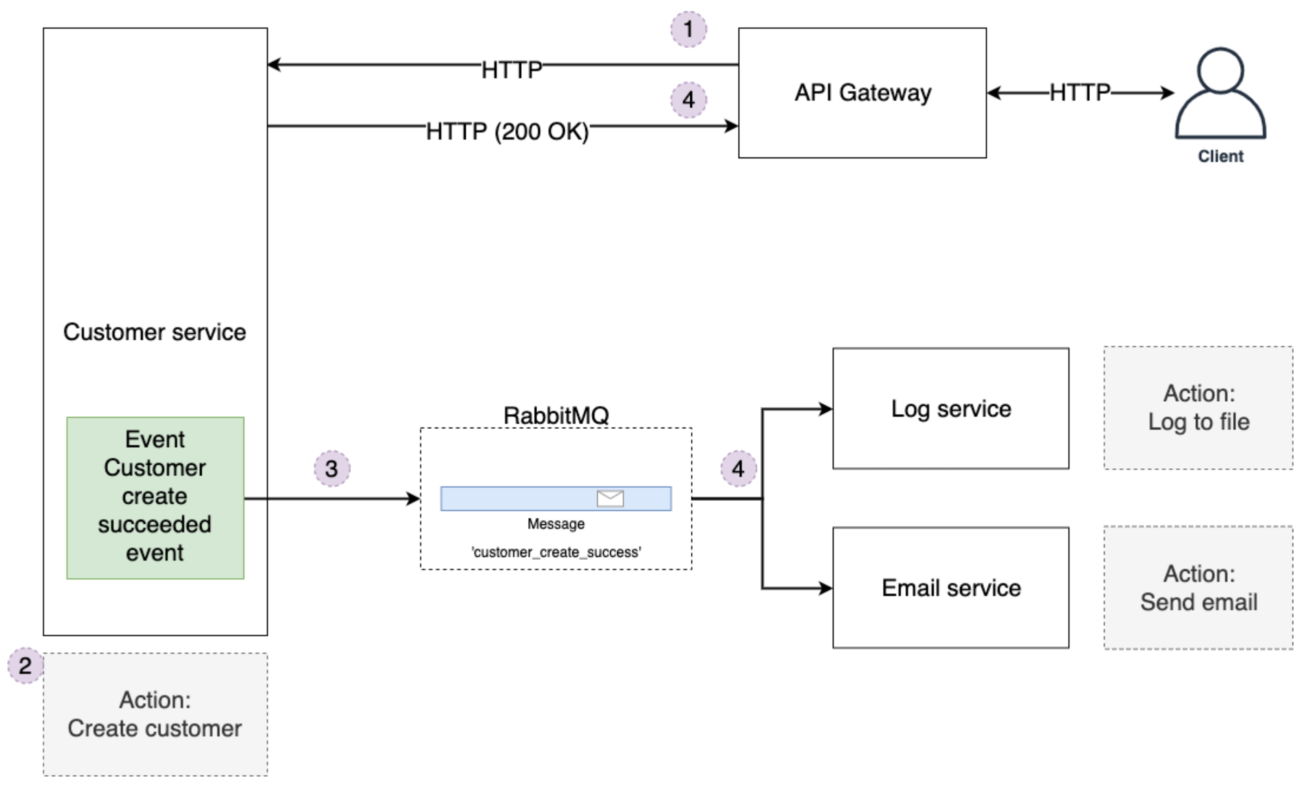
- **(10%) API gateway**
  - Create a Dockerfile
  - Port 7000 should be mapped to the port on which the application runs
  - Should be a part of the container network 'flamingo-network'
  - Name resolution with the name 'api-gateway'
- **(20%) Customer service**
  - **(5%)** Service
    - Create a Dockerfile
    - Should be a part of the container network 'flamingo-network'
    - Name resolution with the name 'customer-service'
    - Application should run on port 80 when the container starts
  - **(15%)** Customer database
    - Based on the official image **postgres**
    - Should be a part of the container network 'flamingo-network'
    - The default password should be Abc.12345
    - The default database should be customer_db
    - Name resolution with the name 'customer-db'
    - When started should execute the init.sql script located in /customer-db/db-scripts
- **(15%) Order service**
  - **(5%)** Service
    - Create a Dockerfile
    - Should be part of the container network 'flamingo-network'
    - Name resolution with the name 'order-service'
    - Application should run on port 80 when the container starts
  - **(10%)** Order database
    - Based on the official image **mongo**
    - Should be a part of the container network 'flamingo-network'
    - Name resolution with the name 'order-db'
- **(10%) Log service**
  - Create a Dockerfile
  - Should be part of the container network 'flamingo-network'
  - Name resolution with the name 'log-service'
- **(10%) Email service**
  - Create a Dockerfile
  - Should be part of the container network 'flamingo-network'
  - Name resolution with the name 'email-service'
  - Add your **Mailgun** configurations to **app.py**

- **(30%) Message broker (RabbitMQ)**
  - Based on the official image **rabbitmq**
  - Should be a part of the container network ‘flamingo-network’
  - Name resolution with the name ‘message-broker’
  - Hostname must be set to ‘message-broker’
  - Default user should be ‘user’
  - Default password should be ‘pass’
  - Default virtual host should be ‘/‘
- **(5%) Network**
  - Create a bridge network called ‘flamingo-network’
- **(20%) Docker compose**
  - Setup a docker-compose.yml at the root folder for easier start and teardown of your services

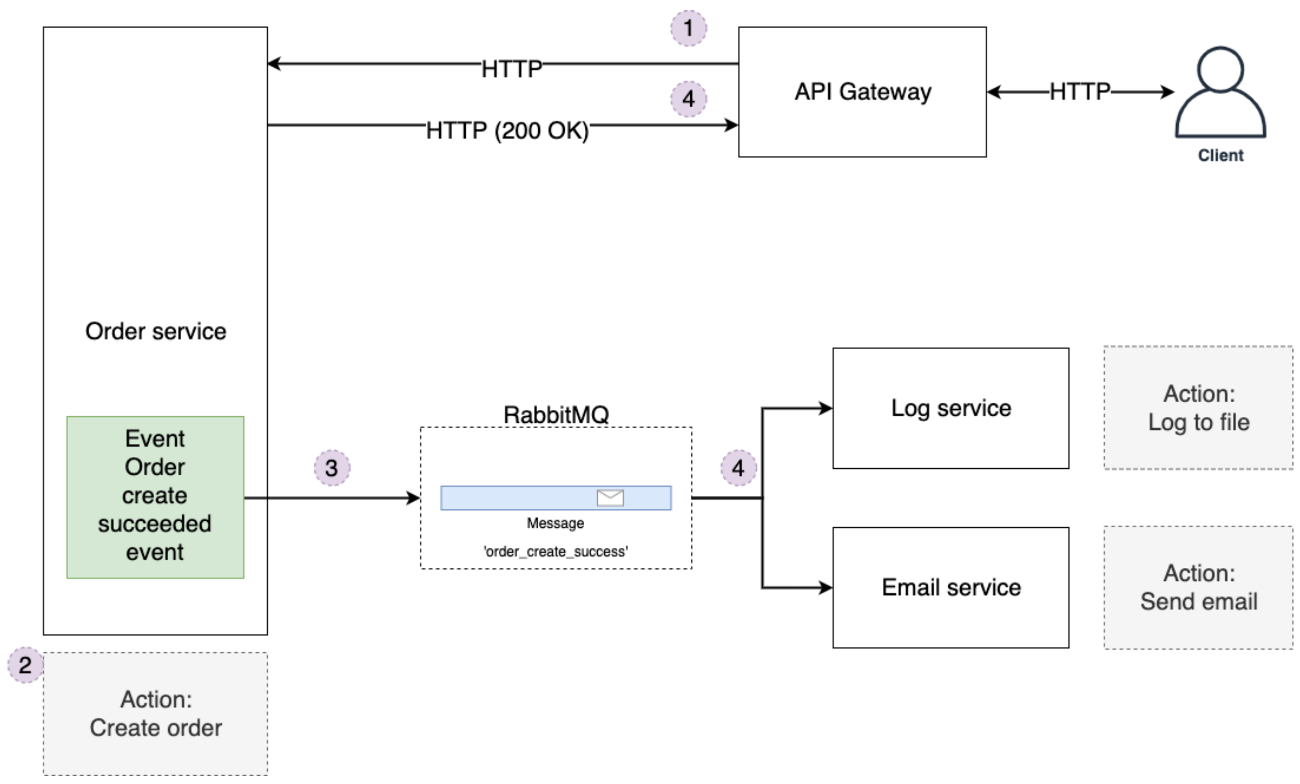
Overview



Create customer flow



Create order flow



Submission

A single compressed file (***.zip, *.rar**) containing the following files:

- The files within **template.zip** where all configurations regarding **Docker** have been added
- commands.txt - *which includes all the commands used to create the images, run the containers, etc… (basically everything that involves the **Docker** setup of the microservice structure)*