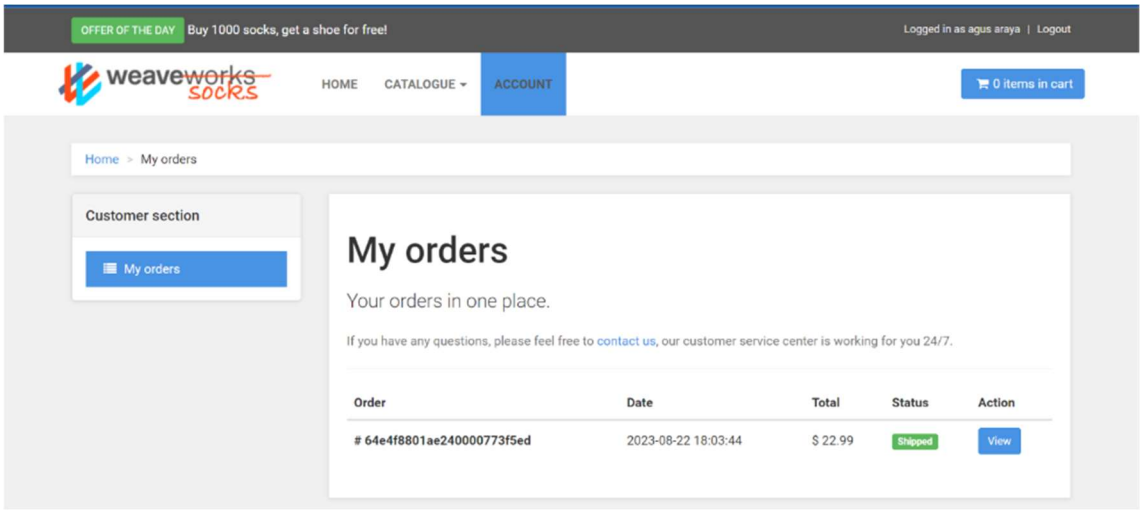
```
D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4>mkdir -p socks-demo

D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4>cd socks-demo

D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4\socks-demo>git clone https://github.com/microservices-demo/microservices-demo.g
it
Cloning into 'microservices-demo'...
remote: Enumerating objects: 10197, done.
remote: Total 10197 (delta 0), reused 0 (delta 0), pack-reused 10197
Receiving objects: 100% (10197/10197), 52.95 MiB | 10.04 MiB/s, done.
Resolving deltas: 100% (6208/6208), done.
```

```
D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4\socks-demo\microservices-demo>docker-compose -f deploy/docker-compose/docker-co
mpose.yml up -d
time="2023-10-15T19:05:18-03:00" level=warning msg="The \"MYSQL_ROOT_PASSWORD\" variable is not set. Defaulting to a bla
nk string."
[+] Running 12/15
 - user-sim 11 layers [▣▣▣▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                                    12.4s
 - orders 3 layers [▣▣▣]      0B/0B      Pulling                                                              12.4s
 - catalogue 4 layers [▣▣▣▣]      0B/0B      Pulling                                                          12.4s
 - rabbitmq 14 layers [▣▣▣▣▣▣▣▣▣▣▣▣▣▣] 3.146MB/51.44MB Pulling                                               12.4s
 - shipping 3 layers [▣▣▣]      0B/0B      Pulling                                                           12.4s
 - carts 9 layers [▣▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                                       12.4s
 - payment 4 layers [▣▣▣▣]      0B/0B      Pulling                                                          12.4s
 - queue-master 2 layers [▣▣]      0B/0B      Pulling                                                       12.4s
 - user 3 layers [▣▣▣]      0B/0B      Pulling                                                             12.4s
 - catalogue-db 12 layers [▣▣▣▣▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                             12.4s
 - orders-db 14 layers [▣▣▣▣▣▣▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                             12.4s
 - edge-router 4 layers [▣▣▣▣]      0B/0B      Pulling                                                      12.4s
 - carts-db Pulling                                                                                       12.4s
 - user-db 11 layers [▣▣▣▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                                  12.4s
 - front-end 8 layers [▣▣▣▣▣▣▣▣]      0B/0B      Pulling                                                    12.4s
```




Ejercicio 2

Contenedores creados al ejecutar el Docker-compose.yml

```yaml
services:
  front-end:
    image: weaveworksdemos/front-end:0.3.12
    hostname: front-end
    restart: always
    cap_drop:
      - all
    read_only: true
  edge-router:
    image: weaveworksdemos/edge-router:0.1.1
    ports:
      - '80:80'
      - '8080:8080'
    cap_drop:
```

```
      - all
    cap_add:
      - NET_BIND_SERVICE
      - CHOWN
      - SETGID
      - SETUID
      - DAC_OVERRIDE
    read_only: true
    tmpfs:
      - /var/run:rw,noexec,nosuid
    hostname: edge-router
    restart: always
  catalogue:
    image: weaveworksdemos/catalogue:0.3.5
    hostname: catalogue
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
  catalogue-db:
    image: weaveworksdemos/catalogue-db:0.3.0
    hostname: catalogue-db
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_ALLOW_EMPTY_PASSWORD=true
      - MYSQL_DATABASE=socksdb
  carts:
    image: weaveworksdemos/carts:0.4.8
    hostname: carts
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
    environment:
      - JAVA_OPTS=-Xms64m -Xmx128m -XX:+UseG1GC -
Djava.security.egd=file:/dev/urandom -Dspring.zipkin.enabled=false
  carts-db:
    image: mongo:3.4
    hostname: carts-db
    restart: always
    cap_drop:
      - all
```

```
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
  orders:
    image: weaveworksdemos/orders:0.4.7
    hostname: orders
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
    environment:
      - JAVA_OPTS=-Xms64m -Xmx128m -XX:+UseG1GC -
Djava.security.egd=file:/dev/urandom -Dspring.zipkin.enabled=false
  orders-db:
    image: mongo:3.4
    hostname: orders-db
    restart: always
    cap_drop:
      - all
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
  shipping:
    image: weaveworksdemos/shipping:0.4.8
    hostname: shipping
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
    environment:
      - JAVA_OPTS=-Xms64m -Xmx128m -XX:+UseG1GC -
Djava.security.egd=file:/dev/urandom -Dspring.zipkin.enabled=false
  queue-master:
```

```
    image: weaveworksdemos/queue-master:0.3.1
    hostname: queue-master
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
  rabbitmq:
    image: rabbitmq:3.6.8
    hostname: rabbitmq
    restart: always
    cap_drop:
      - all
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
      - DAC_OVERRIDE
    read_only: true
  payment:
    image: weaveworksdemos/payment:0.4.3
    hostname: payment
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
  user:
    image: weaveworksdemos/user:0.4.4
    hostname: user
    restart: always
    cap_drop:
      - all
    cap_add:
      - NET_BIND_SERVICE
    read_only: true
    environment:
      - MONGO_HOST=user-db:27017
  user-db:
    image: weaveworksdemos/user-db:0.4.0
    hostname: user-db
    restart: always
    cap_drop:
```

```
      - all
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
    read_only: true
    tmpfs:
      - /tmp:rw,noexec,nosuid
  user-sim:
    image: weaveworksdemos/load-test:0.1.1
    cap_drop:
      - all
    read_only: true
    hostname: user-simulator
    command: "-d 60 -r 200 -c 2 -h edge-router"
```

Se crean los contenedores:

- front-end
- Edge-router
- Catalogue
- Catalogue-db
- Carts-db
- Orders
- Oders-db
- Shipping
- Queue-master
- Rabbitmq
- Payment
- User
- User-db
- User-sim

```
D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4\socks-demo>git clone https://github.com/microservices-demo/front-end.git
Cloning into 'front-end'...
remote: Enumerating objects: 1236, done.
remote: Total 1236 (delta 0), reused 0 (delta 0), pack-reused 1236Receiving objects:  96% (1187/1236), 47.39 MiB | 10.5
Receiving objects: 100% (1236/1236), 47.90 MiB | 10.53 MiB/s, done.

Resolving deltas: 100% (687/687), done.

D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4\socks-demo>git clone https://github.com/microservices-demo/user.git
Cloning into 'user'...
remote: Enumerating objects: 1063, done.
Receiving objects: 100% (1063/1063), 172.83 KiB | 1.56 MiB/s, done.
Resolving deltas:   9% (55/601)eused 0 (delta 0), pack-reused 1063
Resolving deltas: 100% (601/601), done.

D:\Facu\ingsw3\IngenieriaDeSoftware3\TP4\socks-demo>git clone https://github.com/microservices-demo/edge-router.git
Cloning into 'edge-router'...
remote: Enumerating objects: 50, done.
Receiving objects: 100% (50/50), 14.51 KiB | 571.00 KiB/s, done.ceiving objects:  52% (26/50)
Resolving deltas: 100% (12/12)
Resolving deltas: 100% (12/12), done.
```

3. ¿Por qué cree usted que se está utilizando repositorios separados para el código y/o la configuración del sistema? Explique puntos a favor y en contra.

La separación de repositorios para el código y la configuración del sistema tiene sus ventajas, como la organización clara, la colaboración eficiente y el versionado independiente. Sin embargo, también introduce complejidad adicional, dificulta el seguimiento y puede requerir una mayor sobrecarga inicial. La elección de utilizar repositorios separados o unificados depende de las necesidades del proyecto y las preferencias del equipo de desarrollo.

3. ¿Cuál contenedor hace las veces de API Gateway?

El contenedor que hace de API Gateway es edge-router. El repositorio que corresponde a este contenedor tiene un archivo traefik.toml. Traefik es un *Edge Router*; esto significa que es la **puerta a la plataforma**. Se encarga de *interceptar* cada petición que se realiza y enrutarla al servicio correcto. Traefik sabe la lógica y las reglas que determinan qué servicio es el encargado de gestionar cada petición.

3. Cuando ejecuto este comando:

```
curl http://localhost/customers
```

¿Cuál de todos los servicios está procesando la operación?

El servicio user realiza la operación y se comunica con front-end

6. ¿Y para los siguientes casos?

curl http://localhost/catalogue
curl http://localhost/tags

El servicio de catalogue realiza la operación y se comunica con front-end en ambos casos.

8. ¿Como perisisten los datos los servicios?

**Volúmenes de Docker:** Los volúmenes son una forma de persistir datos en Docker. Los volúmenes son directorios o archivos que están fuera del sistema de archivos del contenedor y se montan en el contenedor. Esto permite que los datos persistan incluso cuando el contenedor se detiene o se elimina. Los volúmenes pueden ser administrados por Docker y pueden compartirse entre varios contenedores.

Los servicios que deben almacenar datos, en este caso, user-db y catalogue-db, tienen creados volúmenes de Docker para la persistencia de datos.

8. ¿Cuál es el componente encargado del procesamiento de la cola de mensajes?

Queue-master

8. ¿Qué tipo de interfaz utilizan estos microservicios para comunicarse?

Estos microservicios utilizan principalmente comunicación basada en **HTTP/HTTPS** para interactuar entre sí. Esto se logra a través de interfaces API expuestas por los servicios que permiten a otros microservicios realizar solicitudes HTTP a rutas específicas para obtener o enviar datos.