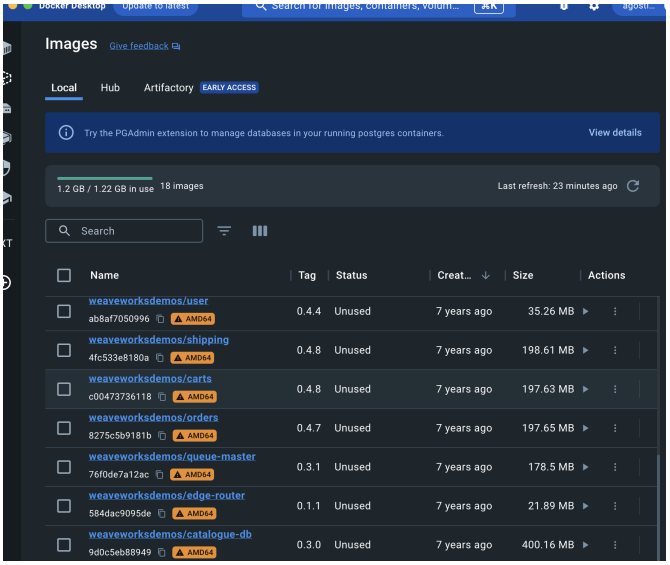
Trabajo Práctico 3 - Arquitectura de Sistemas Distribuidos

Debido a incompatibilidades con mi computadora Mac M2 tuve que utilizar una windows para este trabajo:



1- Sistema distribuido simple

Ejecutar el siguiente comando para crear una red en docker

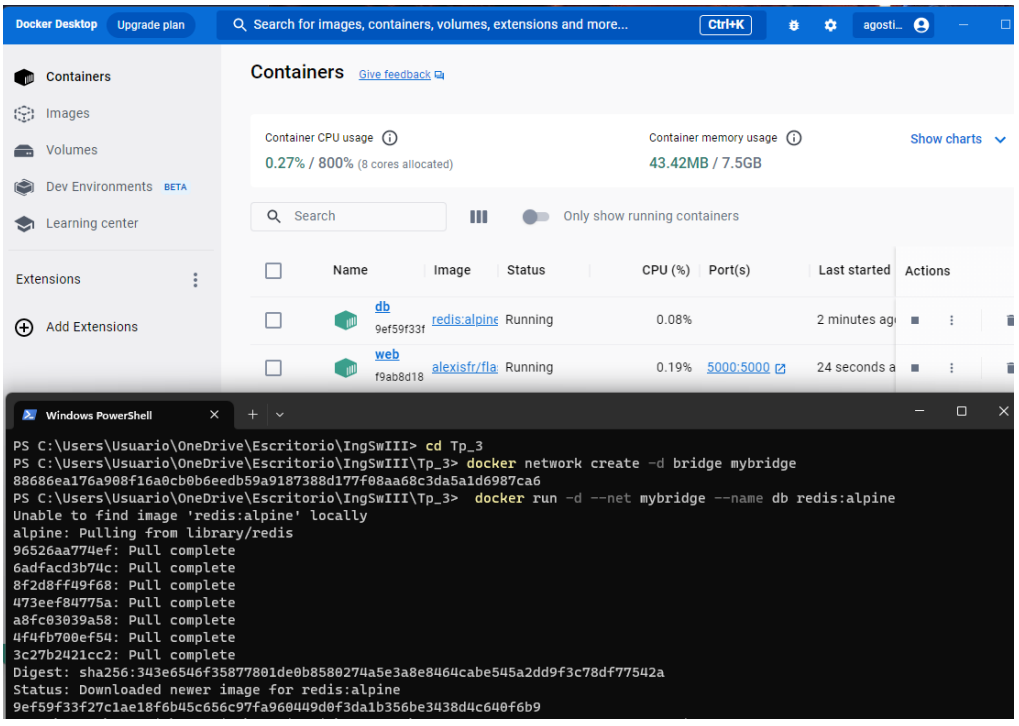
docker network create -d bridge mybridge

Instanciar una base de datos Redis conectada a esa Red.

docker run -d --net mybridge --name db redis:alpine

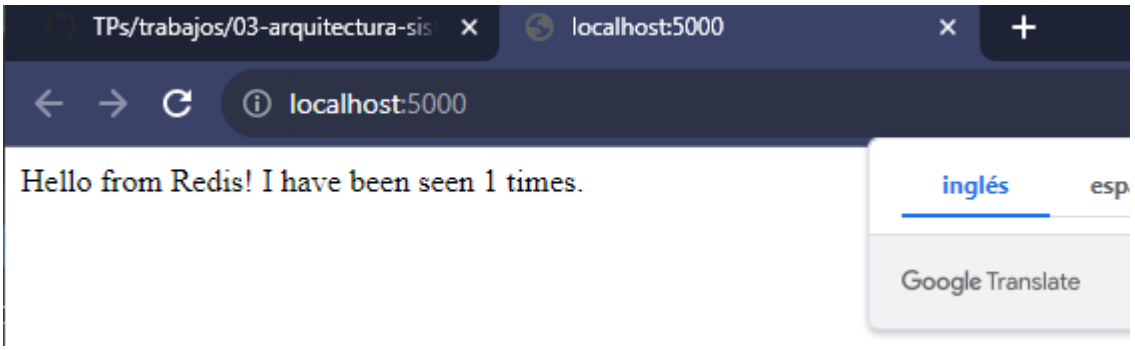
Levantar una aplicacion web, que utilice esta base de datos

docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask-app:latest



```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3>   docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=637
9 -p 5000:5000 --name web alexisfr/flask-app:latest
Unable to find image 'alexisfr/flask-app:latest' locally
latest: Pulling from alexisfr/flask-app
f49cf87b52c1: Pull complete
7b491c575b06: Pull complete
b313b08bab3b: Pull complete
51d6678c3f0e: Pull complete
09f35bd58db2: Pull complete
1bda3d37eead: Pull complete
9f47966d4de2: Pull complete
9fd775bfe531: Pull complete
2446eec18066: Pull complete
b98b851b2dad: Pull complete
e119cb75d84f: Pull complete
Digest: sha256:250221bea53e4e8f99a7ce79023c978ba0df69bdfe620401756da46e34b7c80b
Status: Downloaded newer image for alexisfr/flask-app:latest
f9ab8d1866ea96c5e085655397f4b4f6c9996b9b72ceeeb1cabc9466edb3a7f9
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> S
```

- Abrir un navegador y acceder a la URL: http://localhost:5000/



- Verificar el estado de los contenedores y redes en Docker: ¿Cuáles puertos están abiertos?

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker ps -a
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS                      PORTS
                                  NAMES
3321a73cb207   alexisfr/flask-app:latest   "python /app.py"         17 seconds ago   Created
                                  web
a185a429353c   redis:alpine               "docker-entrypoint.s…"   26 seconds ago   Up 25 seconds               6379/tc
p                                 db
9f375d2a8eed   example-voting-app-worker   "dotnet Worker.dll"      33 minutes ago   Up 17 minutes
                                  example-voting-app-worker-1
6e15d043dfc8   example-voting-app-result   "nodemon server.js"      33 minutes ago   Up 17 minutes               0.0.0.0
:5858->5858/tcp, 0.0.0.0:5001->80/tcp   example-voting-app-result-1
94aef35a08c4   example-voting-app-vote     "python app.py"          33 minutes ago   Up 17 minutes (healthy)     0.0.0.0
:5000->80/tcp                     example-voting-app-vote-1
313ff025f681   postgres:15-alpine         "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes (unhealthy)   5432/tc
p                                 example-voting-app-db-1
ea13835047c0   redis:alpine               "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes (unhealthy)   6379/tc
p                                 example-voting-app-redis-1
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> S
```

- Mostrar detalles de la red mybridge con Docker.

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker network inspect mybridge
[
    {
        "Name": "mybridge",
        "Id": "6fd1d89247ab0d0998d4ad45c40321b7b91fde26b285525dd89f92d062dd230a",
        "Created": "2023-10-16T15:11:46.849787979Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.23.0.0/16",
                    "Gateway": "172.23.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "a185a429353cf435092e0e9e5e7a87f06818a636a36217803787e336f8ab16c0": {
                "Name": "db",
                "EndpointID": "ab51cbf03f91447a01efa6083c76fd3270628bebef53ae2fe4453f25e3b0d012",
                "MacAddress": "02:42:ac:17:00:02",
                "IPv4Address": "172.23.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

- ¿Qué comandos utilizó?
    - ***docker ps -a:*** para ver los puertos abiertos
    - ***docker network inspect mybridge:*** para inspeccionar la red mybridge,

2- Análisis del sistema
Siendo el código de la aplicación web el siguiente:

```
import os
from flask import Flask
from redis import Redis
app = Flask(__name__)
redis = Redis(host=os.environ['REDIS_HOST'], port=os.environ['REDIS_PORT'])
bind_port = int(os.environ['BIND_PORT'])
@app.route('/')
def hello():
    redis.incr('hits')
    total_hits = redis.get('hits').decode()
    return f'Hello from Redis! I have been seen {total_hits} times.'
if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True, port=bind_port)
```

Explicar cómo funciona el sistema

El sistema consiste en una aplicación web. Utiliza Redis como una base de datos en memoria para llevar un registro de cuántas veces se visitó.

- Importa bibliotecas, ***os*** gestiona variables de entorno, ***Flask*** crea la aplicación web y ***Redis*** para interactuar con la base de datos en memoria.

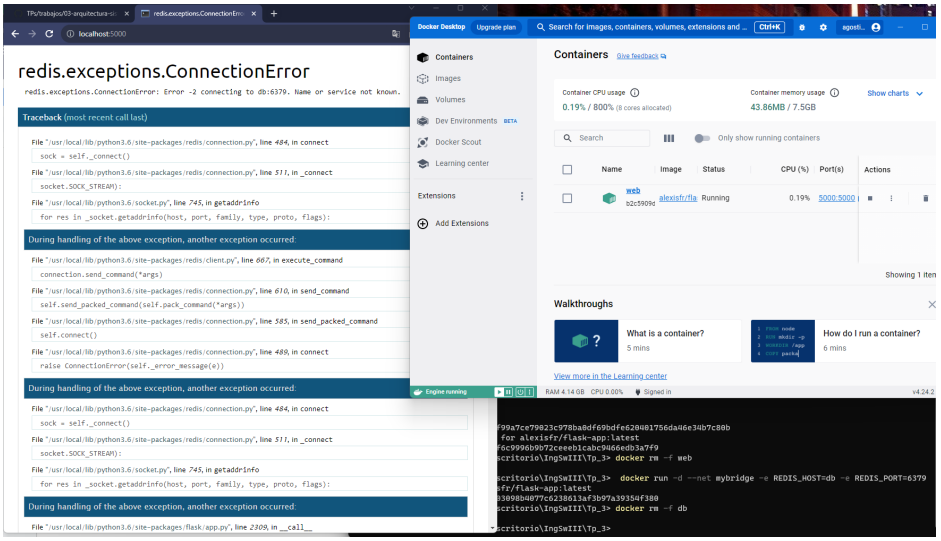
- Define una ruta que responde a las solicitudes GET. Cada vez que se visita esta ruta (la página) se incrementa el contador en Redis 'hits' y obtiene el valor actual del contador. Luego, devuelve un mensaje que muestra cuántas veces ha sido visitada.

¿Para qué sirven y porque están los parámetros -e en el segundo Docker run del ejercicio 1?
Se utilizan para establecer variables de entorno en el contenedor Docker. Se están configurando las variables REDIS_HOST y REDIS_PORT para que la aplicación Flask pueda conectarse al servidor Redis.

¿Qué pasa si ejecuta docker rm -f web y vuelve a correr docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask-app:latest ?
Si se ejecuta docker rm -f web para eliminar el contenedor de la aplicación web y luego ejecutas nuevamente docker run, se crea un nuevo contenedor. Sin perder el contador de visitas, no se borro la base de datos.

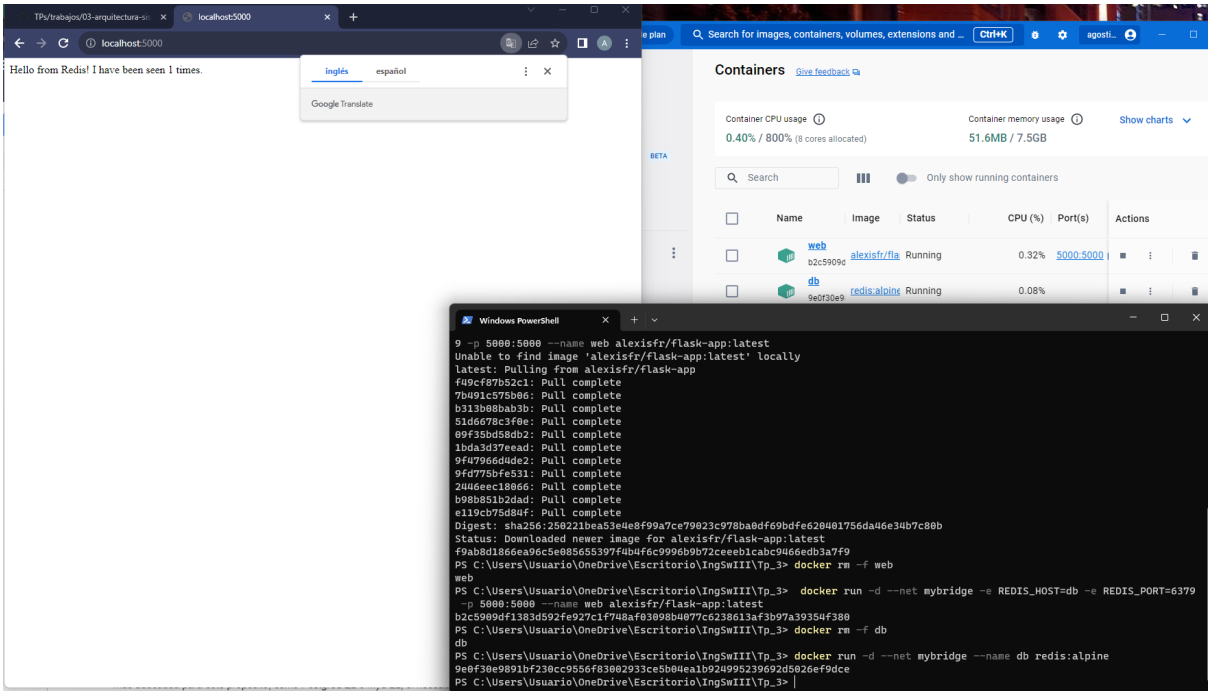
```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker rm -f web
web
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3>  docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379
 -p 5000:5000 --name web alexisfr/flask-app:latest
b2c5909df1383d592fe927c1f748af03098b4077c6238613af3b97a39354f380
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3>
```

¿Qué ocurre en la página web cuando borro el contenedor de Redis con docker rm -f db?
Cuando se borra el contenedor de Redis con ***docker rm -f db***, la aplicación web ya no puede conectarse a Redis y muestra errores al intentar acceder.

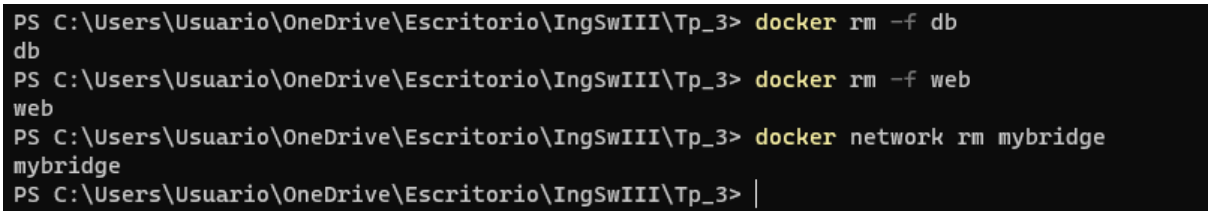


Y si lo levantó nuevamente con docker run -d --net mybridge --name db redis:alpine ? ¿Qué considera usted que haría falta para no perder la cuenta de las visitas?
Si se vuelve a levantar el contenedor de Redis, la conexión entre la aplicación web y Redis se restablece. Pero, el contador de visitas comienza desde cero, porque Redis es una base de datos en memoria y no persiste los datos después de detener o eliminar el contenedor.

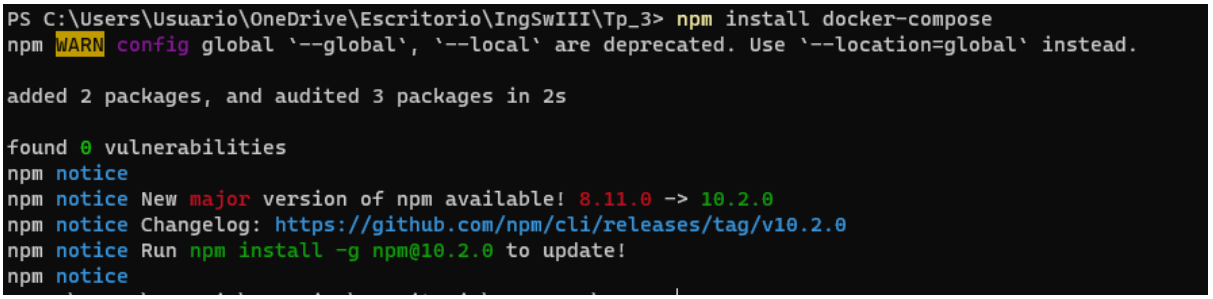
Para evitar perder la cuenta de las visitas al reiniciar los contenedores, deberías configurar volúmenes.



Para eliminar los elementos creados corremos:
docker rm -f db
docker rm -f web
docker network rm mybridge



3- Utilizando docker compose
Normalmente viene como parte de la solución cuando se instaló Docker
De ser necesario instalarlo hay que ejecutar:
sudo pip install docker-compose
npm install docker-compose



- Crear el siguiente archivo docker-compose.yaml en un directorio de trabajo:

```
version: '3.6'
services:
  app:
    image: alexisfr/flask-app:latest
    depends_on:
      - db
    environment:
      - REDIS_HOST=db
      - REDIS_PORT=6379
    ports:
      - "5000:5000"
  db:
    image: redis:alpine
    volumes:
      - redis_data:/data
volumes:
  redis_data:
```

docker-compose.yml | Release Notes: 1.83.1

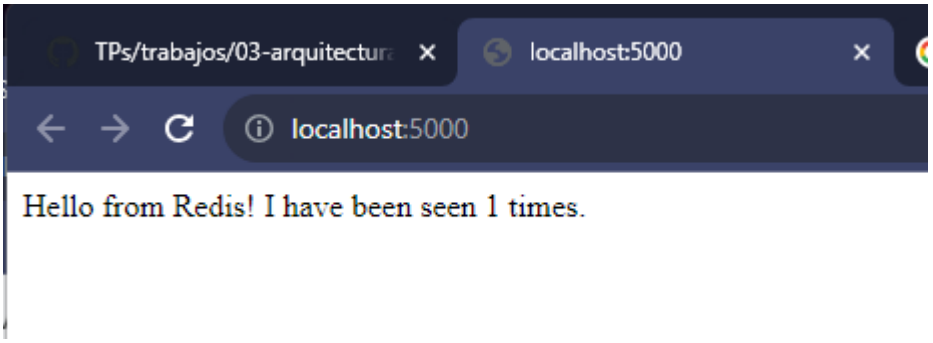
C: > Users > Usuario > OneDrive > Escritorio > IngSwIII > Tp_3 > docker-compose.yml

```yaml
version: '3.6'
services:
  app:
    image: alexisfr/flask-app:latest
    depends_on:
      - db
    environment:
      - REDIS_HOST=db
      - REDIS_PORT=6379
    ports:
      - "5000:5000"
  db:
    image: redis:alpine
    volumes:
      - redis_data:/data
volumes:
  redis_data:
```

- Ejecutar ***docker-compose up -d***

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker-compose up -d
[+] Building 0.0s (0/0)                                                    docker
[+] Running 4/4
 ✓ Network tp_3_default       Created
 ✓ Volume "tp_3_redis_data"   Created
 ✓ Container tp_3-db-1        Started
 ✓ Container tp_3-app-1       Started
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3>
```

- Acceder a la url http://localhost:5000/



Ejecutar ***docker ps, docker network ls y docker volume ls***

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS                      PORTS
                  NAMES
352b98d6655c   alexisfr/flask-app:latest   "python /app.py"         27 seconds ago   Up 26 seconds               0.0.0.0
:5000->5000/tcp   tp_3-app-1
70ca6f21a746   redis:alpine               "docker-entrypoint.s…"   27 seconds ago   Up 26 seconds               6379/tc
p                 tp_3-db-1
313ff025f681   postgres:15-alpine         "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes (unhealthy)   5432/tc
p                 example-voting-app-db-1
ea13835047c0   redis:alpine               "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes (unhealthy)   6379/tc
p                 example-voting-app-redis-1
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker network ls
NETWORK ID     NAME                           DRIVER    SCOPE
b0c8a3789ee0   bridge                         bridge    local
85eac2740624   example-voting-app_back-tier    bridge    local
0b9e8786bb57   example-voting-app_front-tier   bridge    local
b599365682e9   host                           host      local
519a10191b29   none                           null      local
a464a9b142f9   tp_3_default                   bridge    local
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker volume ls
DRIVER    VOLUME NAME
local     28b9cb4cc09edca36d93c7f00b825cd497e151fadb7b3a345c4e94ec2547c492
local     7640c672d163fd1c8832b0bb471e7b8d64e61ba8330ea49694f29e7033c3e4a7
local     e20f26fcfe2f05e739ce25fb78b72cd91b29ab60c5fcb00acf77312a6157ce10
local     example-voting-app_db-data
local     tp_3_redis_data
```

- ¿Qué hizo Docker Compose por nosotros? Explicar con detalle.

Docker Compose es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores. Se utiliza el archivo para configurar los servicios de la aplicación, para que con un solo comando, se creen y se inicien todos los contenedores necesarios de la aplicación. Simplificó la administración de los contenedores interconectados.

Desde el directorio donde se encuentra el archivo docker-compose.yml ejecutar:
docker-compose down

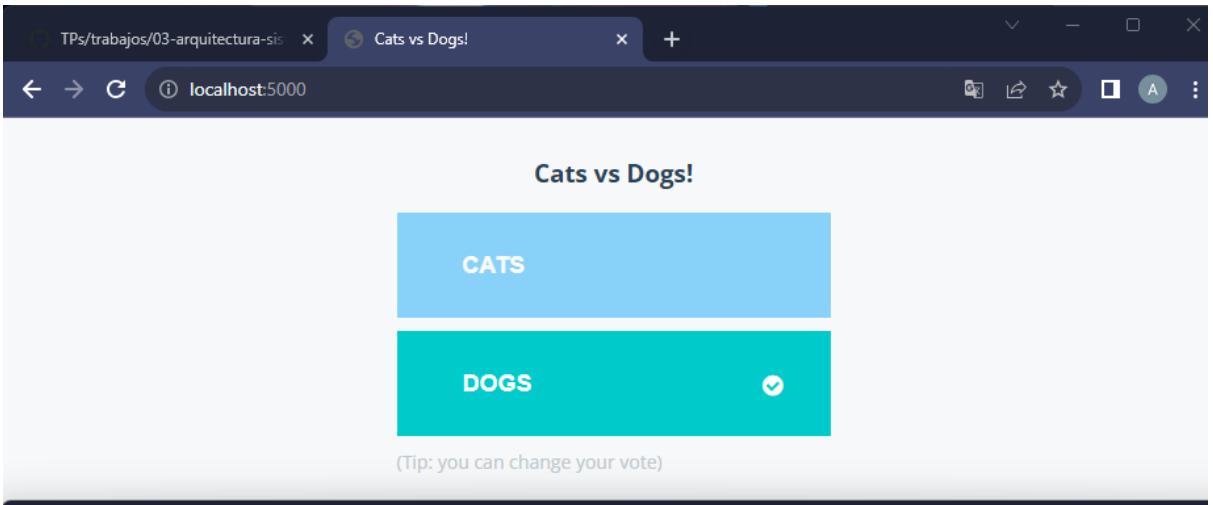
```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> docker-compose down
[+] Running 3/3
 ✓ Container tp_3-app-1  Removed
 ✓ Container tp_3-db-1   Removed
 ✓ Network tp_3_default  Removed
```

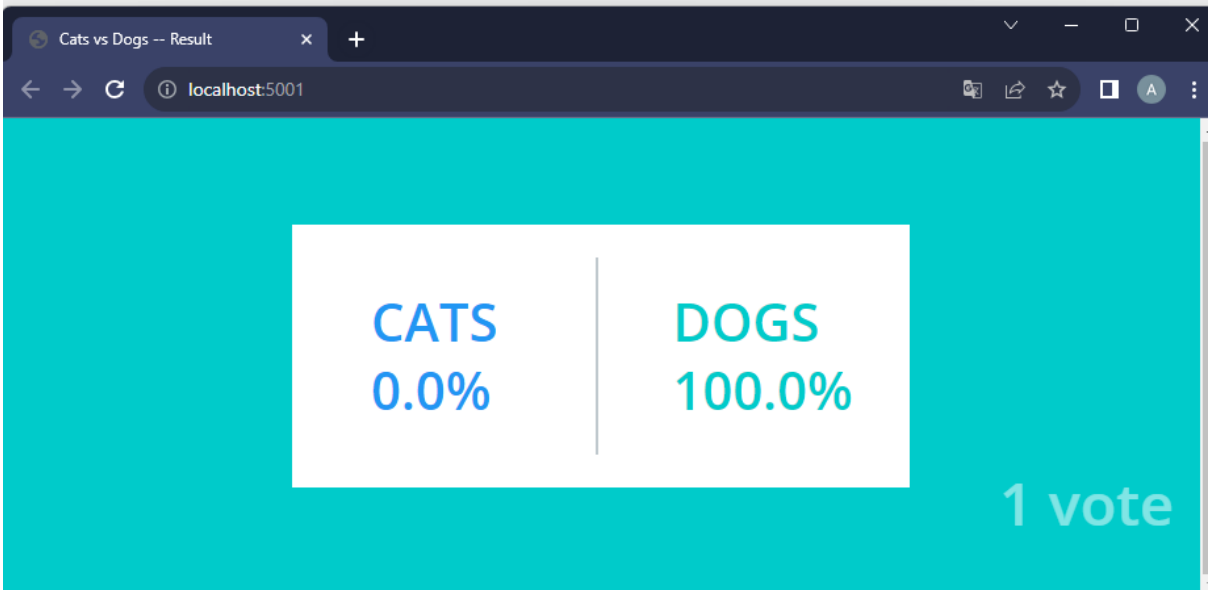
4- Aumentando la complejidad, análisis de otro sistema distribuido.
Este es un sistema compuesto por:
Una aplicación web de Python que te permite votar entre dos opciones
Una cola de Redis que recolecta nuevos votos
Un trabajador .NET o Java que consume votos y los almacena en...
Una base de datos de Postgres respaldada por un volumen de Docker
Una aplicación web Node.js que muestra los resultados de la votación en tiempo real.
Pasos:

Clonar el repositorio ***https://github.com/dockersamples/example-voting-app***
Abrir una línea de comandos y ejecutar
cd example-voting-app
docker-compose -f docker-compose.yml up -d

```
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> git clone https://github.com/dockersamples/example-voting-app
Cloning into 'example-voting-app'...
remote: Enumerating objects: 1099, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 1099 (delta 0), reused 1 (delta 0), pack-reused 1091
Receiving objects: 100% (1099/1099), 1.16 MiB | 2.26 MiB/s, done.
Resolving deltas: 100% (411/411), done.
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3> cd example-voting-app
PS C:\Users\Usuario\OneDrive\Escritorio\IngSwIII\Tp_3\example-voting-app> docker-compose -f docker-compose.yml up -d
[+] Running 9/9
 ✓ db 8 layers [⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                                                          22.7s
   ✓ 96526aa774ef Already exists                                                                           0.0s
   ✓ c34fb5e28db2 Pull complete                                                                            0.8s
   ✓ 70e8a68669fa Pull complete                                                                            0.8s
   ✓ 08ae27ed4de0 Pull complete                                                                           16.7s
   ✓ c0bee3658992 Pull complete                                                                            1.9s
   ✓ efd5b655acb8 Pull complete                                                                            1.9s
   ✓ 15bff2d1d13a Pull complete                                                                            3.0s
   ✓ 615923b9e50e Pull complete                                                                            3.0s
[+] Building 132.6s (41/41) FINISHED                                                                docker:default
 => [worker internal] load .dockerignore                                                                    0.1s
 => => transferring context: 2B                                                                             0.0s
 => [worker internal] load build definition from Dockerfile                                                 0.1s
```

Una vez terminado acceder a ***http://localhost:5000/*** y ***http://localhost:5001***
Emitir un voto y ver el resultado en tiempo real.





Explicar cómo está configurado el sistema, puertos, volúmenes componentes involucrados, utilizar el Docker compose como guía.

El sistema está configurado por:
Servicios:
-"vote": Construye la imagen utilizando a partir del directorio "./vote". Ejecuta la aplicación localmente. Depende del servicio "redis" y se asegura de que esté “healthy” antes de iniciar, ejecutando la prueba "CMD curl -f http://localhost" cada 15 segundos, con un tiempo de espera de 5 segundos y 3 intentos antes de marcarlo como “unhealthy”. El directorio local "./vote" se monta en "/app" en el contenedor. El puerto 5000 del host se mapea al puerto 80 del contenedor.

-"result": Construye la imagen utilizando a partir del directorio "./result".Utiliza "nodemon server.js" para ejecutar el servicio localmente. Depende de los servicios "db" y "redis" y se asegura de que ambos estén saludables antes de iniciar. El directorio local "./result" se monta en "/app" en el contenedor. El puerto 5001 del host se mapea al puerto 80 del contenedor, y el puerto 5858 del host se mapea al puerto 5858 del contenedor.

-"worker": Construye la imagen utilizando a partir del contexto "./worker".Depende de los servicios "redis" y "db" y se asegura de que ambos estén saludables antes de iniciar. Se une a la red "back-tier".

-"redis": Construye la imagen utilizando utilizando la imagen "redis:alpine". Este servicio se une a la red "back-tier". El volumen se monta el directorio local "./healthchecks" en "/healthchecks" en el contenedor.
Healthcheck: Se ejecuta un script de healthcheck llamado "redis.sh" en el contenedor cada 5 segundos.

-"db": Construye la imagen utilizando la imagen "postgres:15-alpine". Se configuran las variables de entorno POSTGRES_USER y POSTGRES_PASSWORD para el contenedor de la base de datos. Se utiliza un volumen llamado "db-data" para persistir los datos de la base de datos. Además, se monta el directorio local "./healthchecks" en "/healthchecks" en el contenedor.
Healthcheck: Se ejecuta un script de healthcheck llamado "postgres.sh" en el contenedor cada 5 segundos.

-"seed": Construye a partir del directorio "./seed-data". Solo se ejecutá si se especifica el perfil "seed" al iniciar Docker Compose. Depende del servicio "vote" y se asegura de que "vote" esté saludable antes de iniciar. Se une a la red "front-tier". Está configurado para no reiniciar automáticamente.

Define un **volumen "db-data"** que se utiliza para persistir los datos de la base de datos PostgreSQL.

Define dos **redes**: "front-tier" y "back-tier" a las que se unen varios servicios para permitir la comunicación entre ellos.

Los servicios se utilizan para ejecutar una aplicación en contenedores Docker, y las redes permiten la comunicación entre estos contenedores. Los healthchecks se utilizan para garantizar la salud de los servicios antes de permitir que se inicien.

Revisar el código de la aplicación Python example-voting-app\vote\app.py para ver cómo envía votos a Redis.

La aplicación Flask recopila votos de los usuarios cuando se envían mediante una solicitud POST y almacena esos votos en una cola Redis llamada "votes". La cola Redis es accedida a través de una conexión Redis configurada en la función "get_redis". Esto permite que la aplicación almacene y procese los votos de los usuarios en tiempo real.