TP 3 - ARQUITECTURA SISTEMAS DISTRIBUIDOS

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

```
C:\Users\sofia>docker ps -a
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS                      PORTS
               NAMES
69ad631c890f   alexisfr/flask-app:latest   "python /app.py"         24 seconds ago   Up 20 seconds               0.0.0.0:5
000->5000/tcp   web
a9b87379de27   redis:alpine               "docker-entrypoint.s…"   5 minutes ago    Up 5 minutes                6379/tcp
               db
d9615be38a28   postgres:9.4               "docker-entrypoint.s…"   24 hours ago     Exited (0) 21 hours ago
               my-postgres
05468a216409   mywebapi                   "dotnet SimpleWebAPI…"   6 days ago       Exited (0) 6 days ago
               myapi2
3cf36d1c6640   mywebapi                   "dotnet SimpleWebAPI…"   6 days ago       Exited (0) 6 days ago
               myapi

C:\Users\sofia>
```

- Mostrar detalles de la red mybridge con Docker.


```
C:\Users\sofia>docker network inspect mybridge
[
    {
        "Name": "mybridge",
        "Id": "560d3e4d9630b124de13b44d85bc011893c16a88c5fd92ad19f48402c71024d3",
        "Created": "2023-08-15T17:25:51.066752721Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "69ad631c890f15c39c0212843ff614f2dc4c0e0a6816d2d79fe328bf8d791e31": {
                "Name": "web",
                "EndpointID": "d122c1eced140bc404572bb179a6ba98ebedadae89d0640d5c2c3039267c6837",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
```


- ¿Qué comandos utilizó?

docker ps -a y **docker network inspect mybridge**

- Explicar cómo funciona el sistema. ¿Para qué sirven y porque están los parámetros -e en el segundo Docker run del ejercicio 1?

El código es una aplicación web que cuenta cuántas veces se accede a una página y almacena este dato en Redis. La aplicación se ejecuta en Docker para facilitar la gestión.

El parámetro -e en el comando Docker se utiliza para establecer variables de entorno dentro del contenedor. En este caso, se usan -e REDIS_HOST=db para definir la dirección de la base de datos Redis como "db" (un nombre de servicio en la red), y -e REDIS_PORT=6379 para definir el puerto de Redis como 6379.

- ¿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 elimino el contenedor de la aplicación web y luego lo vuelvo a correr, sigue funcionando igual y no se reinicia el conteo de visitas a la página, ya que no se eliminó la base de datos.

```
C:\Users\sofia>docker rm -f web
web

C:\Users\sofia>docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask-app:latest
5e702ec54f7885d0500df2295584c8a178d8bef89aa9659721d2a87df9c8e42a

C:\Users\sofia>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS              PORTS                    NAMES
5e702ec54f78   alexisfr/flask-app:latest   "python /app.py"         About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp   web
a9b87379de27   redis:alpine               "docker-entrypoint.s…"   10 minutes ago       Up 10 minutes       6379/tcp                 db

C:\Users\sofia>
```

- ¿Qué ocurre en la página web cuando borro el contenedor de Redis con docker rm -f db? ¿Y si lo levanto nuevamente con docker run -d --net mybridge --name db redis:alpine ?

Si borro el contenedor de Redis y luego lo levanto, se reinicia el conteo de visitas a la página, empieza desde cero visitas.

```
C:\Users\sofia>docker rm -f db
db

C:\Users\sofia>docker run -d --net mybridge --name db redis:alpine
3e79846b49daf25cfb36441b0b06a8ce7250d6d0b58d4ac5703073c06ba85353

C:\Users\sofia>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED         STATUS         PORTS                    NAMES
3e79846b49da   redis:alpine               "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   6379/tcp                 db
5e702ec54f78   alexisfr/flask-app:latest   "python /app.py"         2 minutes ago   Up 2 minutes   0.0.0.0:5000->5000/tcp   web

C:\Users\sofia>
```

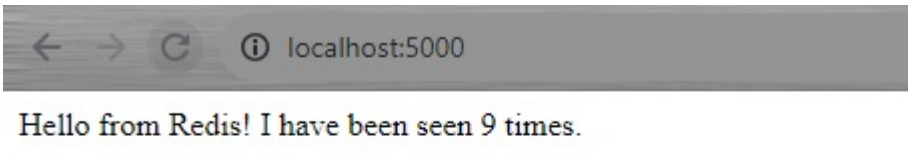
- ¿Qué considera usted que haría falta para no perder la cuenta de las visitas?

Para no perder la cuenta de las visitas se deben utilizar volúmenes.

- Ejecutar docker-compose up -d

```
C:\Users\sofia\OneDrive\Escritorio\docker_compose>docker-compose up
Creating network "docker_compose_default" with the default driver
Creating volume "docker_compose_redis_data" with default driver
Creating docker_compose_db_1 ... done
Creating docker_compose_app_1 ... done
Attaching to docker_compose_db_1, docker_compose_app_1
db_1   | 1:C 15 Aug 2023 17:50:45.512 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
db_1   | 1:C 15 Aug 2023 17:50:45.513 # Redis version=7.0.12, bits=64, commit=00000000, modified=0, pid=1, just started
db_1   | 1:C 15 Aug 2023 17:50:45.513 # Warning: no config file specified, using the default config. In order to specify a config file use redis-ser
ver /path/to/redis.conf
db_1   | 1:M 15 Aug 2023 17:50:45.514 * monotonic clock: POSIX clock_gettime
db_1   | 1:M 15 Aug 2023 17:50:45.517 * Running mode=standalone, port=6379.
db_1   | 1:M 15 Aug 2023 17:50:45.518 # Server initialized
db_1   | 1:M 15 Aug 2023 17:50:45.518 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low m
emory condition. Being disabled, it can can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328.
To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to
take effect.
db_1   | 1:M 15 Aug 2023 17:50:45.520 * Ready to accept connections
app_1  |  * Serving Flask app "app" (lazy loading)
app_1  |  * Environment: production
app_1  |    WARNING: Do not use the development server in a production environment.
app_1  |    Use a production WSGI server instead.
app_1  |  * Debug mode: on
app_1  |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
app_1  |  * Restarting with stat
app_1  |  * Debugger is active!
app_1  |  * Debugger PIN: 270-815-763
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:37] "GET / HTTP/1.1" 200 -
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:40] "GET / HTTP/1.1" 200 -
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:40] "GET / HTTP/1.1" 200 -
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:40] "GET / HTTP/1.1" 200 -
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:40] "GET / HTTP/1.1" 200 -
app_1  | 172.21.0.1 - - [15/Aug/2023 17:51:41] "GET / HTTP/1.1" 200 -
```

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



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

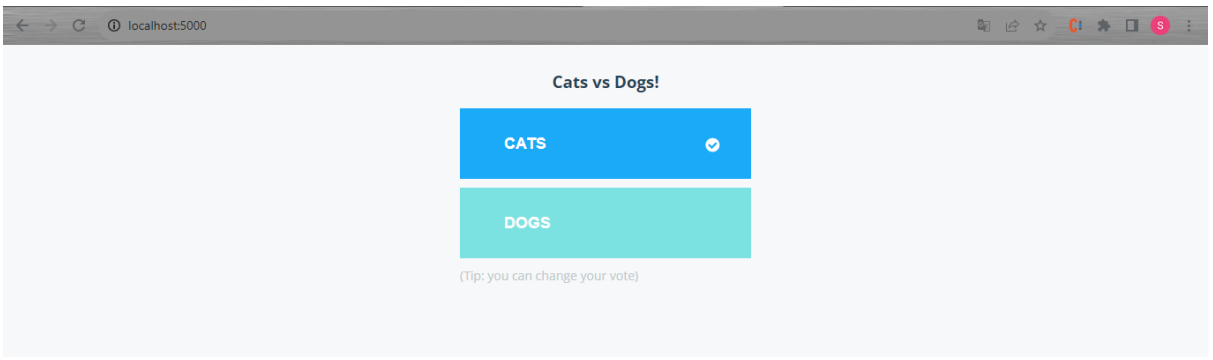
Docker Compose es una herramienta que ayuda a manejar aplicaciones compuestas por varios contenedores. En el archivo docker-compose.yaml, se describen los servicios y sus configuraciones. Esto permite crear y coordinar todos los contenedores de la aplicación con un solo comando. En tu caso, se definió una aplicación web y una base de datos Redis, y Docker Compose se encargará de asegurar que funcionen juntos de manera eficiente y coherente.

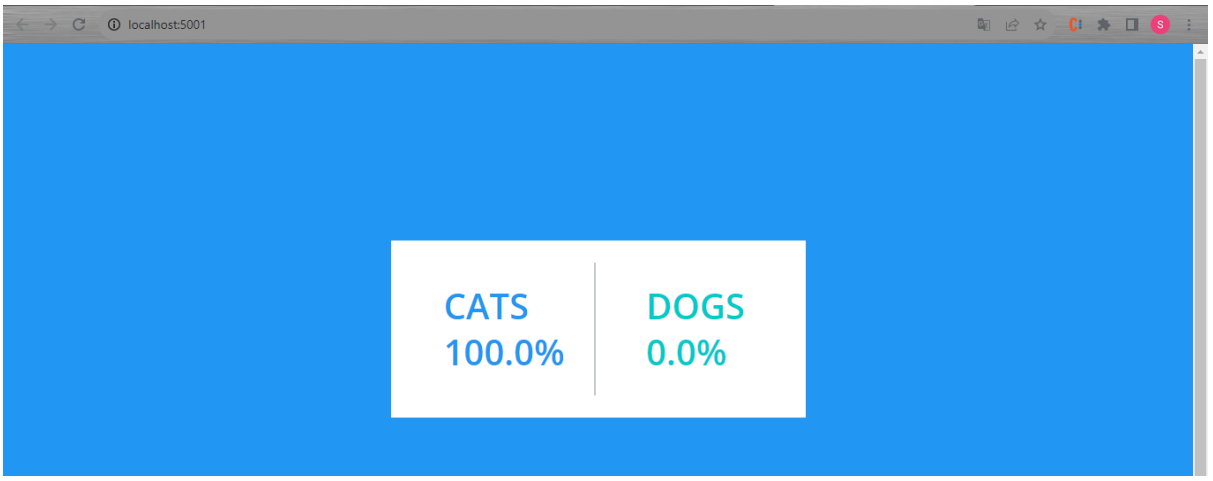
```
C:\Users\sofia\OneDrive\Escritorio\docker_compose>docker-compose down
Removing docker_compose_app_1 ... done
Removing docker_compose_db_1  ... done
Removing network docker_compose_default

C:\Users\sofia\OneDrive\Escritorio\docker_compose>
```

- **cd example-voting-app**
- **docker-compose -f docker-compose.yml up -d**

```
PS C:\Users\sofia\OneDrive\Escritorio\vote-app\example-voting-app> docker-compose -f docker-compose.yml up -d
Creating network "example-voting-app_back-tier" with the default driver
Creating network "example-voting-app_front-tier" with the default driver
Creating example-voting-app_redis_1 ... done
Creating example-voting-app_db_1    ... done
Creating example-voting-app_vote_1  ... done
Creating example-voting-app_worker_1 ... done
Creating example-voting-app_result_1 ... done
PS C:\Users\sofia\OneDrive\Escritorio\vote-app\example-voting-app>
```





Análisis de https://github.com/dockersamples/example-voting-app

- Explicar como está configurado el sistema, puertos, volumenes componenetes involucrados, utilizar el Docker compose como guía.

Docker Compose permite definir, configurar y gestionar múltiples contenedores y sus relaciones de una manera organizada y eficiente, simplificando el despliegue y la administración de la aplicación.

- **Servicio "vote":**

Se construye a partir del directorio ./vote.
Utiliza el comando python app.py para iniciar la aplicación.
Depende del servicio "redis" para funcionar correctamente.
Mapea el puerto del contenedor 80 al puerto del host 5000.
Tiene un volumen compartido ./vote:/app para mantener los archivos y cambios locales sincronizados con el contenedor.
Pertenece a las redes "front-tier" y "back-tier".

- **Servicio "result":**

Se construye a partir del directorio ./result.
Utiliza el comando nodemon server.js para iniciar el servidor.
Depende del servicio "db" para funcionar correctamente.
Mapea los puertos del contenedor 80 y 5858 (para depuración) a los puertos del host 5001 y 5858, respectivamente.

Tiene un volumen compartido ./result:/app para sincronizar los archivos entre el host y el contenedor.
Pertenece a las redes "front-tier" y "back-tier".

- **Servicio "worker":**

Se construye a partir del directorio ./worker.
Depende de los servicios "redis" y "db" para funcionar correctamente.
Pertenece a la red "back-tier".

- **Servicio "redis":**

Utiliza la imagen predefinida redis:alpine.
Tiene un volumen compartido ./healthchecks:/healthchecks para mantener los scripts de comprobación de salud.
Pertenece a la red "back-tier".

- **Servicio "db":**

Utiliza la imagen predefinida postgres:15-alpine.
Define las variables de entorno para configurar el usuario y la contraseña de PostgreSQL.
Tiene dos volúmenes: uno para los datos de PostgreSQL y otro para los scripts de comprobación de salud.
Realiza una comprobación de salud utilizando el script /healthchecks/postgres.sh cada 5 segundos.
Pertenece a la red "back-tier".

- **Servicio "seed":**

Se construye a partir del directorio ./seed-data.
Se ejecuta solo cuando se especifica el perfil "seed" al usar el comando docker compose --profile seed up -d.
Depende del servicio "vote" para funcionar correctamente.
Pertenece a la red "front-tier".

- **Volumen "db-data":**

Se crea para almacenar los datos de PostgreSQL.

- **Redes "front-tier" y "back-tier":**

Proporcionan aislamiento y comunicación entre los servicios. "front-tier" es para los componentes frontales y "back-tier" es para los componentes traseros.

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

Este código implementa una aplicación Flask que permite a los usuarios votar entre dos opciones y luego envía los votos a Redis.

- **Definición de opciones:**

Se obtienen las opciones para votar desde las variables de entorno OPTION_A y OPTION_B. Si no se proporcionan, se utilizan valores predeterminados ("Cats" y "Dogs").

- **Método POST y GET:**

Si el método de la solicitud es POST, se procesa el voto recibido:

- Se obtiene el voto de la solicitud POST y se guarda en la base de datos Redis junto con el ID del votante en formato JSON.

Se genera un ID de votante único si aún no tiene uno (se guarda en una cookie).

En resumen, este código crea una aplicación web Flask que permite a los usuarios votar entre dos opciones. Los votos se almacenan en Redis junto con el ID único del votante. La aplicación también gestiona cookies para rastrear a los votantes y presenta una interfaz amigable para votar.

- Revisar el código del worker example-voting-app\worker\program.cs para entender cómo procesa los datos.

Este código del worker en C# procesa los datos almacenados en la base de datos Redis y los actualiza en una base de datos PostgreSQL.

- **Procesamiento del voto:**

Se muestra un mensaje en la consola indicando que se está procesando un voto específico.
Se verifica si la conexión a la base de datos PostgreSQL está abierta. Si no lo está, se intenta reconectar.
Si la conexión está abierta, se actualiza la base de datos con el voto y el ID del votante.

- **Funciones de apoyo:**

OpenDbConnection: Abre y gestiona una conexión a la base de datos PostgreSQL. Crea la tabla "votes" si no existe.
OpenRedisConnection: Abre y gestiona una conexión a Redis. Utiliza una dirección IP en lugar de un nombre de host debido a problemas conocidos en StackExchange.Redis.
GetIp: Obtiene la dirección IP asociada a un nombre de host.
UpdateVote: Actualiza la base de datos PostgreSQL con los votos. Si ya existe un voto para el votante, se actualiza en lugar de insertar uno nuevo.
En resumen, el worker se encarga de procesar los votos almacenados en Redis, actualizarlos en la base de datos PostgreSQL y realizar la gestión de conexiones para asegurar la continuidad del proceso.

- Revisar el código de la aplicación que muestra los resultados example-voting-app\result\server.js para entender como muestra los valores.

Este código en JavaScript muestra los valores de los resultados de votación utilizando Express y Socket.IO para la comunicación en tiempo real.

- **Obtención y emisión de votos:**

La función getVotes(client) se llama después de establecer una conexión exitosa a la base de datos.
Se realiza una consulta SQL para obtener los votos y sus recuentos de la tabla "votes".
Los resultados se procesan en la función collectVotesFromResult(result) para organizarlos en un formato legible.
Los resultados se emiten a través de Socket.IO a través del evento "scores".

- **Función para procesar los resultados:**

La función collectVotesFromResult(result) toma los resultados de la consulta SQL y los organiza en un objeto votes, donde las claves son "a" y "b" (correspondientes a las opciones de voto) y los valores son los recuentos de votos.

En resumen, la aplicación se conecta a la base de datos PostgreSQL y obtiene periódicamente los resultados de votación. Luego, los resultados se emiten a través de Socket.IO para que los clientes puedan ver y actualizarse en tiempo real en la interfaz web.

- Escribir un documento de arquitectura sencillo, pero con un nivel de detalle moderado, que incluya algunos diagramas de bloques, de secuencia, etc y descripciones de los distintos componentes involucrados es este sistema y cómo interactúan entre sí.