



**UNIVERSIDAD
CATÓLICA
DE CÓRDOBA**
JESUITAS

**Ingeniería de Software 3
TRABAJO PRÁCTICO N°3**

Profesor: Ing Fernando Bono.

Alumno: Menel Angelo (1804789)

EJERCICIO 1:

```
~ docker network create -d bridge mybridge
465b06acf48a0de91cab2b82de9357f9a504dc3d1e10f47a2db6bffd636e9ddc
~ docker run -d --net mybridge --name db redis:alpine
Unable to find image 'redis:alpine' locally
alpine: Pulling from library/redis
213ec9aee27d: Pull complete
c99be1b28c7f: Pull complete
8ff0bb7e55e3: Pull complete
6d80de393db7: Pull complete
8dbffc478db1: Pull complete
7402bc4c98a0: Pull complete
Digest: sha256:dc1b954f5a1db78e31b8870966294d2f93fa8a7fba5c1337a1ce4ec55f311bc3
Status: Downloaded newer image for redis:alpine
c968f1bc65af0fa4b2c84e600f6de94590045f0e3b4ce411e7efe26f3643949d
```

```
~ Ing_3/Trabajo_Practico_3 develop docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                               NAMES
5005085119ca   alexisfr/flask-app:latest   "python /app.py"        42 seconds ago   Up 41 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   trabajo_practico_3_app_1
de60b52ba326   redis:alpine             "docker-entrypoint.s..." 42 seconds ago   Up 41 seconds   6379/tcp                                trabajo_practico_3_db_1

~ Ing_3/Trabajo_Practico_3 develop docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
@7b8acb0fe12   bridge   bridge    local
@f6c9749736d   host     host      local
a8d3d063a5be   none     null      local
26ff86b332f1   trabajo_practico_3_default bridge     local

~ Ing_3/Trabajo_Practico_3 develop docker volume ls
DRIVER    VOLUME NAME
local     207ef2d6cabdb481ca0b46def6f86118f277c21ca78b3d5fb03dd097b210f69e
local     dfb8a345189fa8e84d122c9a9b88979267072365f3a313949d564d2ec8fe2ae9
local     trabajo_practico_3_redis_data
```

```
~ docker network inspect mybridge
[
  {
    "Name": "mybridge",
    "Id": "465b06acf48a0de91cab2b82de9357f9a504dc3d1e10f47a2db6bffd636e9ddc",
    "Created": "2022-08-25T14:27:52.339721729-03:00",
    "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": {
      "a5281edcc4394bb7dc531cc2aca895b3dc29d2f6c7c94b05db3d606c955011dc": {
        "Name": "web",
        "EndpointID": "4b92078553794ef511a9a4ea1bbd96fa0becd03628862b9377c9460c402b16eb",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
      },
      "c968f1bc65af0fa4b2c84e600f6de94590045f0e3b4ce411e7efe26f3643949d": {
        "Name": "db",

```

¿Cuáles puertos están abiertos?

- Los puertos levantados son el 5000 para la aplicación web y el 6379 para la db no está abierto pero reservado.

¿Qué comandos uso?

- docker ps: para ver los puertos levantados.
- docker network inspect {network_name}.

EJERCICIO 2:

```
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'])

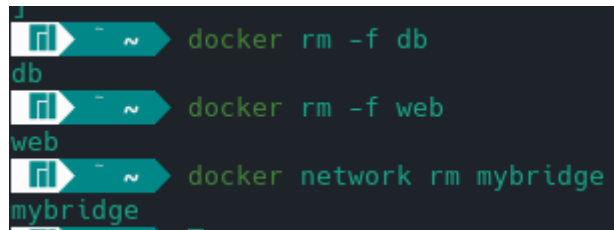
@app.route('/')

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)
```

- El sistema instancia un framework llamado Flask, luego se instancia el motor de DB Redis bajo el host de Redis. se crea la variable port el cual es el puerto que está reservado para Redis. bind_port es el puerto en el cual está escuchando la aplicación web. Luego se indica la ruta para correr la aplicación. Se define una función hello, el cual utiliza el motor de db para incrementar en 1 el atributo hits, se trae el dato de hits desde la DB y se imprime el texto con la variable total_hits. Por último se crea el main donde se corre el framework bajo el host 0.0.0.0 y el puerto 5000.
- Se utiliza el -e para poder vincular un contenedor a otro usando el puerto del segundo y su nombre de host.
- Si se ejecuta docker rm -f web se elimina el contenedor de la aplicación web y lo volvemos a correr nos va a mostrar los mismos datos debido a que la información está almacenada en el otro contenedor.
- Cuando borramos el contenedor de la DB, estamos eliminando toda la información que veníamos almacenando y si lo corremos de vuelta empezamos con la información original.

```
❯ docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -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: sha256:250221bea53e4e8f99a7ce79023c978ba0df69bdf620401756da46e34b7c80b
Status: Downloaded newer image for alexisfr/flask-app:latest
a5281edcc4394bb7dc531cc2aca895b3dc29d2f6c7c94b05db3d606c955011dc
```

A terminal window with a dark background and green text. It shows three commands being executed: 'docker rm -f db', 'docker rm -f web', and 'docker network rm mybridge'. Each command is preceded by a prompt character and a tilde '~'. The output of each command is shown on the line immediately following it: 'db', 'web', and 'mybridge' respectively.

```
~ docker rm -f db
db
~ docker rm -f web
web
~ docker network rm mybridge
mybridge
```

Cuando borramos el contenedor web, la imagen de db sigue corriendo pero no vamos a poder acceder a la misma. Al volver a levantar la imagen de web, vamos a seguir teniendo los datos, porque db nunca freno. Pero si borramos db y web, al levantar web vamos a empezar de cero sin datos en db.

Cuando borro el contenedor de Redis con `docker rm -f db`, lo que sucede es que este comando va a forzar su eliminación por más que esté corriendo. Se borra la imagen de la base de datos, pero todavía podemos acceder, por lo que nos muestra un error de que la base de datos no existe. Si levanto nuevamente db con `docker run -d --net mybridge --name db redis:alpine`, podemos acceder a la página web ya que la imagen de web está corriendo y se levanta la imagen de la base de datos pero desde cero.

De esta manera podemos ver la independencia de las dos imágenes.

Para no perder la cuenta de las visitas se podría pensar en almacenar las mismas en otra base de datos de respaldo, para que en el caso de que sea borrada la original, no se pierdan los datos.

EJERCICIO 3:

El docker compose lo que hace es crear una red con distintos contenedores a partir de un archivo yaml, éste además de crear los contenedores y vincularlos también los levanta y quedan corriendo hasta que se bajen con el `docker compose down`.

EJERCICIO 4:

El sistema se levanta utilizando 2 networks (back-tier y front-tier), se levantan 5 contenedores (redis, postgres y 3 aplicaciones web, uno para votar, otro para el resultado y un tercero llamado worker). Tanto la app web de voto y resultado escuchan en diferentes puertos (5000 y 5001 relativamente), el motor de DB escucha sobre el puerto 49513 y la base de datos postgres tiene reservado el puerto 5432. Se utiliza un solo volumen para almacenar localmente los datos de la DB.

EJERCICIO 5:

My Redis databases redisinsight

[+ ADD REDIS DATABASE](#)

Limited offer: Up to 6 months free with \$200 credits. Try Redis Cloud with enhanced database capabilities. >

Or follow the guides: [Build from source](#) [Docker](#) [Homebrew](#)

<input type="checkbox"/> Database Alias	Host:Port	Last connection ↑
<input type="checkbox"/> redis	127.0.0.1:49153	about 1 hour ago ✎ 🗑

redis

[Open](#) [Clone](#)

Connection Type: Standalone

Host: 127.0.0.1

Port: 49153

Username

Password

👁

[Cancel](#) [Apply changes](#)

Connection settings

PostgreSQL connection settings

▼ Connection settings

Initialization

Shell Commands

Client identification

Transactions

General

Metadata

Errors and timeouts

► Data editor

► SQL Editor

Main PostgreSQL Driver properties SSH Proxy SSL

Server

Host: 127.0.0.1 Port: 5432

Database: postgres

Authentication

Authentication: Database Native

Username: postgres

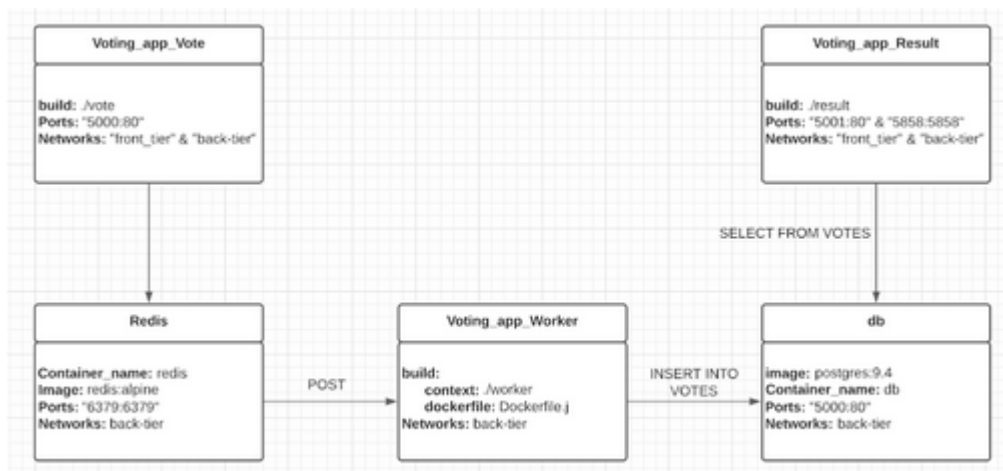
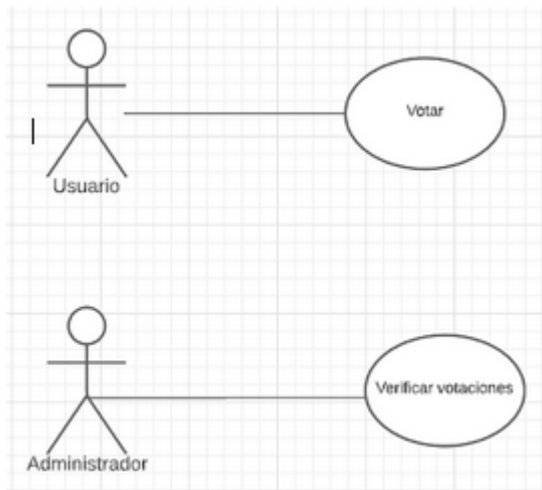
Password: ☒ Save password locally

Advanced

Session role: Local Client:

ⓘ You can use variables in connection parameters.

Driver name: PostgreSQL [Edit Driver Settings](#)



Explicando el diagrama tenemos que:

- **Vote:** el usuario ingresa a la vista localhost:5000 y procede a realizar un voto seleccionando alguna de las opciones (cats o dogs). Al hacer click en una de las opciones se define un voter_id que guarda un registro de quien voto y cuál opción eligió, en donde posteriormente se hace un POST con la información del voto.
- **Redis:** hace de intermediario entre la webapp de vote y el worker.
- **Worker:** va a procesar la información que luego se va a almacenar en la base de datos. Este se conecta a redis y a db, verificando si la conexión de ambas está activa. En el caso de que ambas conexiones estén funcionando, actualiza la base de datos con un insert del id y el voto.
- **db:** como dijimos antes es una base de datos de tipo Postgres la cual en este caso lo que hace es recibir la información por parte del worker y la añade a una tabla de la misma base de datos. Por otro lado recibe las sentencias SELECT por parte de Results para mostrar la información.
- **Results:** esto hace referencia a la webapp Results en la vista localhost:5001 en donde se visualizan los resultados de la votación. Haciendo analogía con el diagrama de casos de uso, esto sería a lo que accedería el Administrador. Esta app, mediante una función, realiza un SELECT a la tabla "votes" y así trae todos los votos agrupados en el caso de ser cats o dogs, y la cantidad total de tuplas, es decir, la cantidad de votos (lo vemos en DBaever). Luego, en el navegador se nos muestra los resultados de las votaciones con sus porcentajes y cantidad de votos.