

TP3

1-Sistema Distribuido Simple

```
C:\Users\Hp>docker network create -d bridge mybridge
```

```
22fdf6dc94724e23108774a74fc9e2bb99ef02a565cdc606b255e86ccc2acf0a
```

```
C:\Users\Hp> docker run -d --net mybridge --name db redis:alpine
```

```
Unable to find image 'redis:alpine' locally
```

```
alpine: Pulling from library/redis
```

```
7264a8db6415: Pull complete
```

```
a28817da73be: Pull complete
```

```
536ccaebaffb: Pull complete
```

```
81efa230a080: Pull complete
```

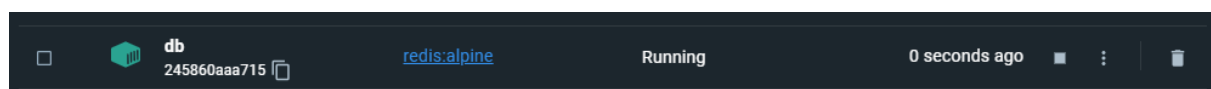
```
a325b46fd795: Pull complete
```

```
ed7e435438ca: Pull complete
```

```
Digest: sha256:32dcb8aedb557eb0d3463537f5fc3c765d1688d4940d487c1f0c3752f2c3916c
```

```
Status: Downloaded newer image for redis:alpine
```

```
245860aaa715d1946eb397af0dda9b653692469018d97bb40509bcfb09411769
```



```
C:\Users\Hp> 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

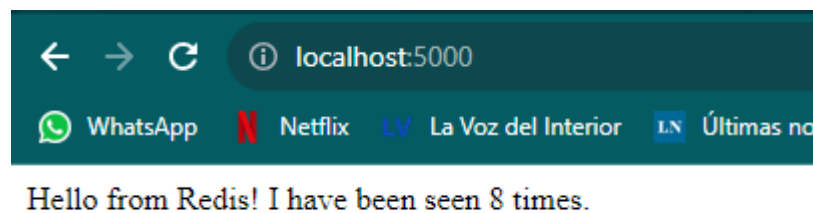
Status: Downloaded newer image for alexisfr/flask-app:latest

0d54594f92f887e6daf7368970160fe6941ec8f57f94e040fb5bc56f5ba4a8d2

<input type="checkbox"/>		db 245860aaa715 	redis:alpine	Running	9 minutes ago	  
<input type="checkbox"/>		web 0d54594f92f8 	alexisfr/flask-app:latest	Running	5000:5000 	5 seconds ago   

C:\Users\Hp>docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
0d54594f92f8	alexisfr/flask-app:latest	"python /app.py"	About a minute ago Up
About a minute	0.0.0.0:5000->5000/tcp	web	
245860aaa715	redis:alpine	"docker-entrypoint.s..."	10 minutes ago Up 10
minutes	6379/tcp	db	



2-Análisis del Sistema

```
PS C:\Users\Hp> docker rm -f web
web
PS C:\Users\Hp> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
PS C:\Users\Hp> docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask-
app:latest
a7f0956e11273ee5a5fc306f56079bc8e45deee4fe19e592b6b2ccd8cc65f367
PS C:\Users\Hp> |
```

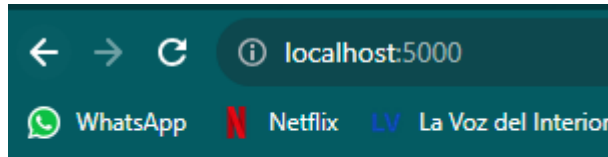
```
PS C:\Users\Hp> docker rm -f db
db
PS C:\Users\Hp> docker run -d --net mybridge --name db redis:alpine
506d2cc2c5ab11055d55f749c23d3c88fa71cf3696d159567a65db9089fa5dd5
PS C:\Users\Hp> |
```

```
PS C:\Users\Hp> docker rm -f db
db
PS C:\Users\Hp> docker rm -f web
web
PS C:\Users\Hp> docker network rm mybridge
mybridge
PS C:\Users\Hp> |
```

3- Utilizando Docker Compose

```
docker-compose.yml
ing-software-3 > SimpleWebAPI > docker-compose.yml
1  version: '3.6'
2  services:
3    app:
4      image: alexisfr/flask-app:latest
5      depends_on:
6        - db
7      environment:
8        - REDIS_HOST=db
9        - REDIS_PORT=6379
10     ports:
11       - "5000:5000"
12     db:
13       image: redis:alpine
14       volumes:
15         - redis_data:/data
16 volumes:
17   redis_data:
18
```

```
Ver Object must be a mapping
PS D:\Santiago\Ingeniería en Sistemas\4to Año\2do Semestre\Ingeniería de Software III\Practico\ing-software-3\TP3> docker-compose up -d
[+] Running 4/4
✔ Network tp3_default      Created                                0.7s
✔ Volume "tp3_redis_data"  Created                                0.0s
✔ Container tp3-db-1       Started                               1.4s
✔ Container tp3-app-1      Started                               2.5s
```



Hello from Redis! I have been seen 6 times.

```
PS C:\Users\Hp> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
5686c185b294   alexisfr/flask-app:latest          "python /app.py"        About a minute Up About a minute 0.0.0.0:500
0->5000/tcp
245f503af858   redis:alpine                       "docker-entrypoint.s..." About a minute Up About a minute 6379/tcp
tp3-db-1

PS C:\Users\Hp> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
7e9e5e0834a9   bridge    bridge    local
55facf1b4589   host      host      local
8e42ab4673f8   none      null      local
85aac443c081   tp3_default bridge    local

PS C:\Users\Hp> docker volume ls
DRIVER    VOLUME NAME
local     2c5290892a64b487c722b942a46e0de0ed127b5c89f0fd1c06a2336664a9a727
local     3a4b98062a9200528180acf8d5f8673009f71324ca3334ecb53be587cfaea376
local     09eb99b84dd4462b8c7aa0e4297de9d22730fc77891341073ca8f9da3ab3b522
local     36bcacc24125c759b99fbc687f1b636a7261adb139db7afc177f99d1a83432dd
local     44c24a045e8dc89cc0afdb95a8eac36f84ee9598521731142dc539bd76234017
local     b098bcee62b838ab5e2a784d7b55f68a04c7ff67df44f2a24183e74da237431e
local     tp3_redis_data
```

Al ejecutar los comandos `docker ps`, `docker network ls` y `docker volume ls`, podrás ver las instancias en ejecución, las redes y los volúmenes creados por Docker Compose para los servicios definidos en el archivo `docker-compose.yaml`. Veamos en detalle qué hizo Docker Compose por nosotros en cada caso:

1. `docker ps`: Este comando muestra información sobre los contenedores en ejecución. Cuando usas Docker Compose, los servicios definidos en tu archivo `docker-compose.yaml` se ejecutan como contenedores. Al ejecutar `docker-compose up -d`, Docker Compose crea y ejecuta estos contenedores en segundo plano. Cuando ejecutas `docker ps`, verás una lista de los contenedores en ejecución junto con información sobre su estado, puertos mapeados y otros detalles.

Por ejemplo, en tu caso, el servicio `app` definido en `docker-compose.yaml` se ejecutará como un contenedor con el nombre `tp3_app_1` (el nombre se genera concatenando el nombre del directorio del proyecto, el nombre del servicio y un sufijo numérico). Al ejecutar `docker ps`, podrás ver información sobre este contenedor, como su estado, puertos mapeados, ID del contenedor, entre otros.

2. `docker network ls`: Este comando muestra información sobre las redes definidas en Docker. Docker Compose crea una red personalizada para los servicios definidos en el archivo `docker-compose.yaml`. Esta red permite que los contenedores de diferentes servicios se comuniquen entre sí utilizando los nombres de servicio en lugar de direcciones IP.

Si siguiendo con tu archivo de ejemplo, al ejecutar `docker network ls`, podrías ver una red con un nombre similar a `tp3_default`. Esta red se creó automáticamente por Docker Compose para conectar los contenedores de los servicios `app` y `db` definidos en tu archivo.

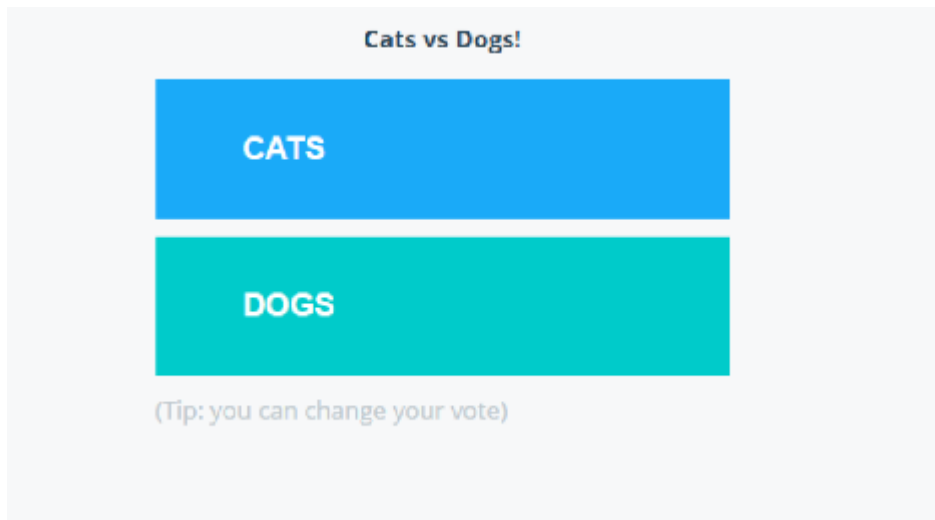
3. `docker volume ls`: Con este comando, puedes ver los volúmenes de Docker creados en el sistema. En tu archivo `docker-compose.yml`, se define un volumen llamado `redis_data`. Este volumen se utiliza para persistir los datos del contenedor del servicio `db` (en este caso, Redis). Cuando el contenedor se detiene o se elimina, los datos en este volumen persistirán para ser utilizados en futuras ejecuciones.

Al ejecutar `docker volume ls`, podrías encontrar un volumen con el nombre `tp3_redis_data` (similar al naming de red). Este volumen es gestionado por Docker Compose y asegura que los datos de Redis se conserven incluso si el contenedor se reemplaza o actualiza.

```
PS D:\Santiago\Ingenieria en Sistemas\4to Año\2do Semestre\Ingenieria de Software III\Practico\ing-software-3\TP3> docker-compose down
[+] Running 3/3
✓ Container tp3-app-1   Removed      0.6s
✓ Container tp3-db-1    Removed      0.7s
✓ Network tp3_default   Removed      0.9s
```

4-Análisis de otro sistema distribuido

```
PS D:\Santiago\Ingenieria en Sistemas\4to Año\2do Semestre\Ingenieria de Software III\Practico\ing-software-3\TP3\example-voting-app> docker-compose up -d
se -f docker-compose.yml up -d
[+] Running 9/9
✓ db 8 layers [#####] 0B/0B Pulled 18.4s
✓ 7264a8db6415 Already exists 0.0s
✓ 6ff36a0c8b9b Pull complete 2.2s
✓ 41485c1d4f30 Pull complete 2.2s
✓ b1c0fd40744e Pull complete 12.1s
✓ c03dbaaa41b8 Pull complete 12.1s
✓ bef2a2546759 Pull complete 12.2s
✓ d80603666f21 Pull complete 12.2s
✓ b76686cd2926 Pull complete 12.3s
[+] Building 52.4s (41/41) FINISHED docker:default
=> [result internal] load build definition from Dockerfile 0.1s
```



Voting App (Python): Aplicación web que permite emitir votos entre dos opciones. Se ejecuta en el puerto 5000.

Result App (Node.js): Aplicación web que muestra los resultados de la votación en tiempo real. Se ejecuta en el puerto 5001.

Redis: Cola de mensajes para recolectar votos. Se ejecuta en el puerto 6379.

Worker (Java/.NET): Aplicación que consume votos de Redis y los almacena en la base de datos PostgreSQL.

PostgreSQL: Base de datos respaldada por un volumen Docker.