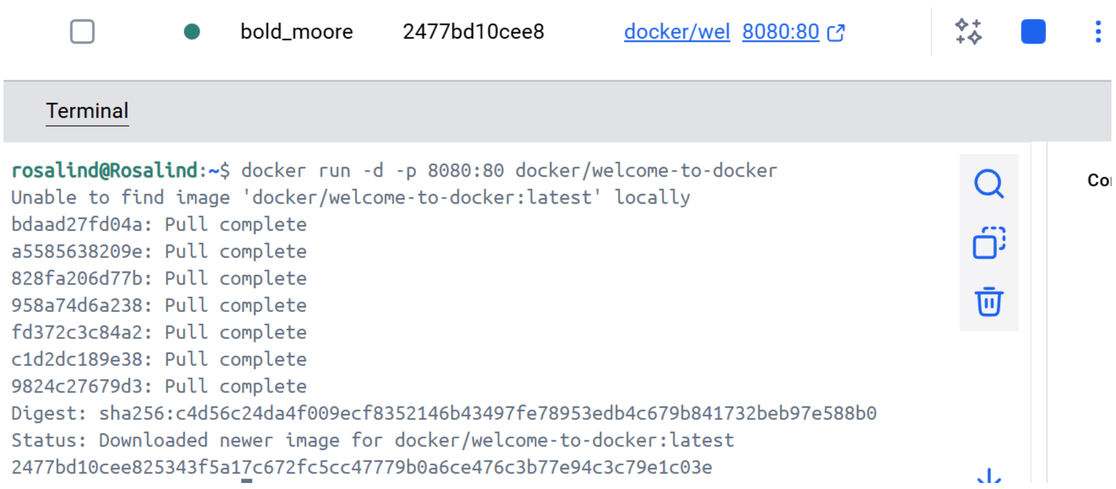
Content

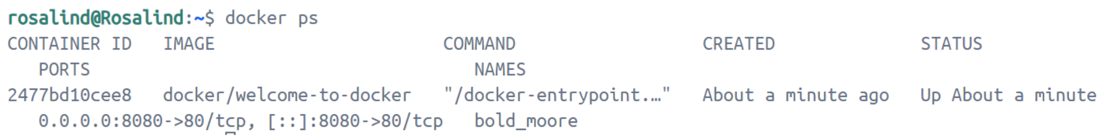
What is a container?

1. **run a container using the CLI**
   Command: *docker run -d -p 8080:80 docker/welcome-to-docker*



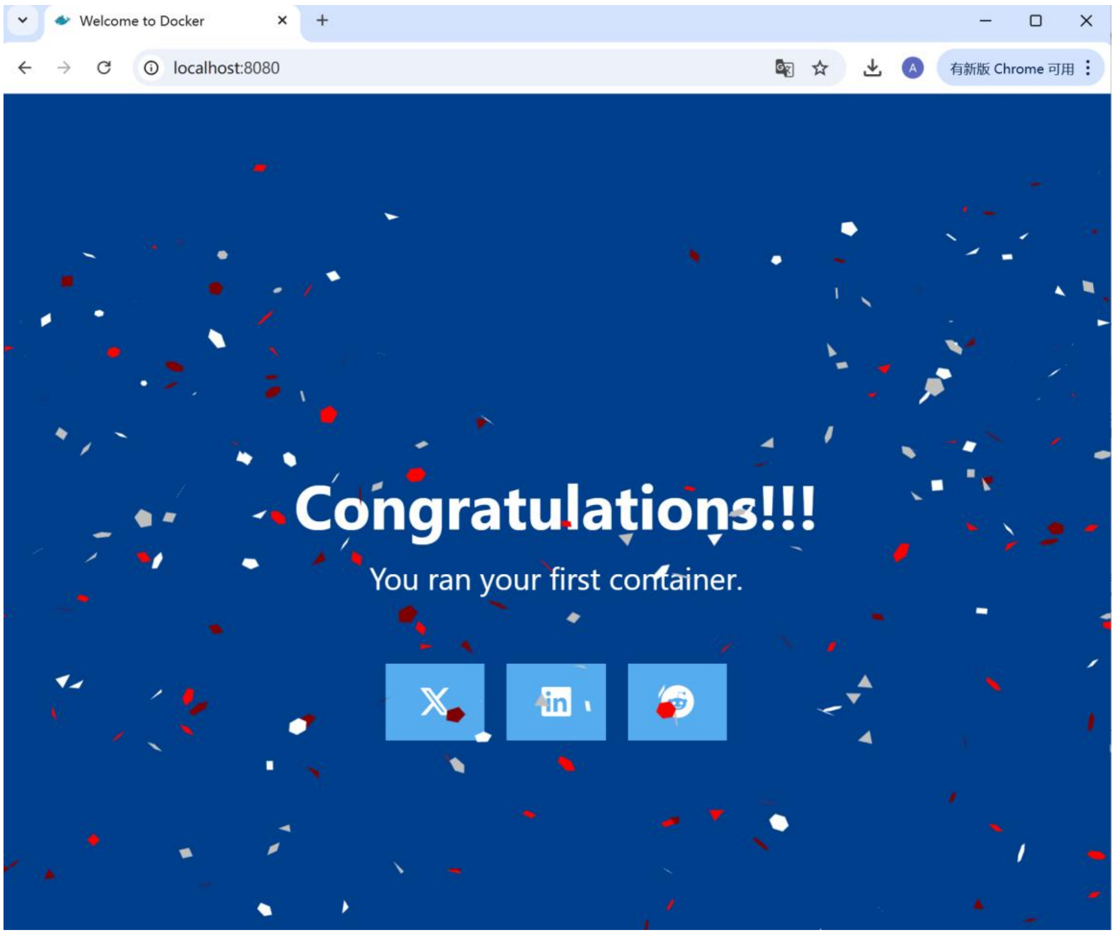
```
rosalind@Rosalind:~$ docker run -d -p 8080:80 docker/welcome-to-docker
Unable to find image 'docker/welcome-to-docker:latest' locally
bdaad27fd04a: Pull complete
a5585638209e: Pull complete
828fa206d77b: Pull complete
958a74d6a238: Pull complete
fd372c3c84a2: Pull complete
c1d2dc189e38: Pull complete
9824c27679d3: Pull complete
Digest: sha256:c4d56c24da4f009ecf8352146b43497fe78953edb4c679b841732beb97e588b0
Status: Downloaded newer image for docker/welcome-to-docker:latest
2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e
```

2. **View the running containers**
   Command: *docker ps*



```
rosalind@Rosalind:~$ docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS
   PORTS                                  NAMES
2477bd10cee8   docker/welcome-to-docker   "/docker-entrypoint.…"   About a minute ago   Up About a minute
   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   bold_moore
```

3. **Access the frontend from http://localhost:8080**

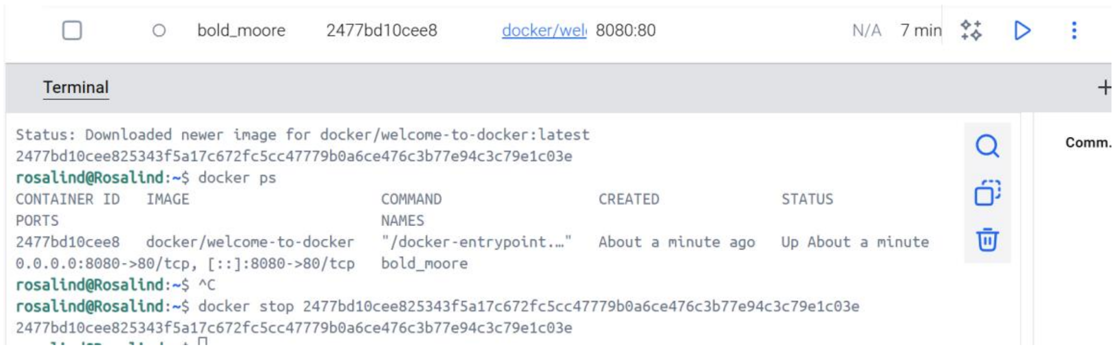


4. **Stop the container**

   First, run *docker ps* to get the ID of the container. Here Container ID is 2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e.

   Then Provide the container ID to the *docker stop* command

   docker stop 2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e



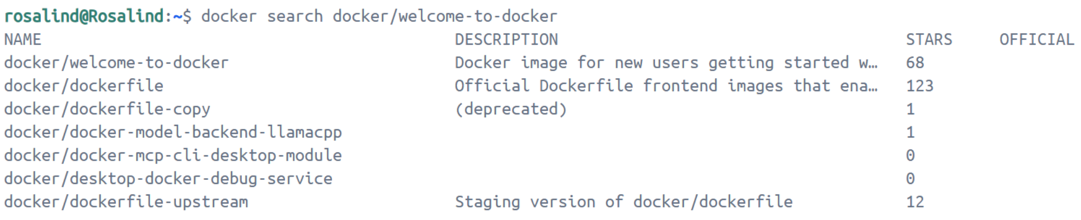
```
bold_moore    2477bd10cee8    docker/wel    8080:80    N/A    7 min
```

```
Status: Downloaded newer image for docker/welcome-to-docker:latest
2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e
rosalind@Rosalind:~$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED              STATUS
PORTS                                       NAMES
2477bd10cee8   docker/welcome-to-docker    "/docker-entrypoint.…"   About a minute ago   Up About a minute
0.0.0.0:8080->80/tcp, [::]:8080->80/tcp    bold_moore
rosalind@Rosalind:~$ ^C
rosalind@Rosalind:~$ docker stop 2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e
2477bd10cee825343f5a17c672fc5cc47779b0a6ce476c3b77e94c3c79e1c03e
```

what-is-an-image

1. search for images using the *docker search* command

docker search docker/welcome-to-docker

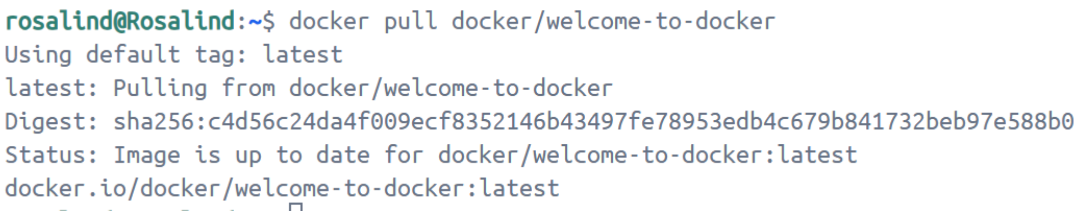


```
rosalind@Rosalind:~$ docker search docker/welcome-to-docker
NAME                                  DESCRIPTION                                     STARS     OFFICIAL
docker/welcome-to-docker              Docker image for new users getting started w…   68
docker/dockerfile                     Official Dockerfile frontend images that ena…   123
docker/dockerfile-copy                (deprecated)                                    1
docker/docker-model-backend-llamacpp                                                  1
docker/docker-mcp-cli-desktop-module                                                  0
docker/desktop-docker-debug-service                                                   0
docker/dockerfile-upstream            Staging version of docker/dockerfile            12
```

Here it shows the available images relatedt docker/welcome-to-docker on Docker Hub.

2. Pull the image using the *docker pull* command.

docker pull docker/welcome-to-docker

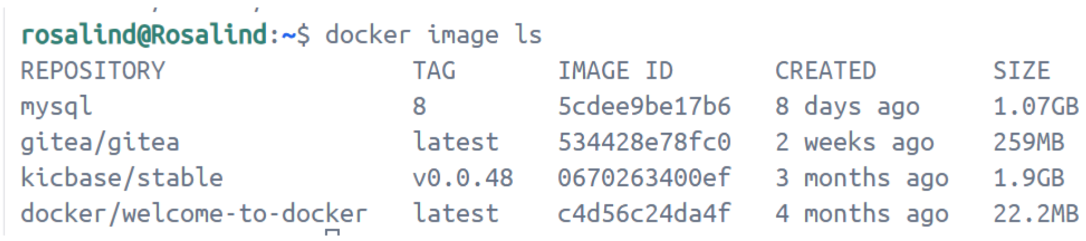


```
rosalind@Rosalind:~$ docker pull docker/welcome-to-docker
Using default tag: latest
latest: Pulling from docker/welcome-to-docker
Digest: sha256:c4d56c24da4f009ecf8352146b43497fe78953edb4c679b841732beb97e588b0
Status: Image is up to date for docker/welcome-to-docker:latest
docker.io/docker/welcome-to-docker:latest
```

3. List the downloaded images

docker image ls

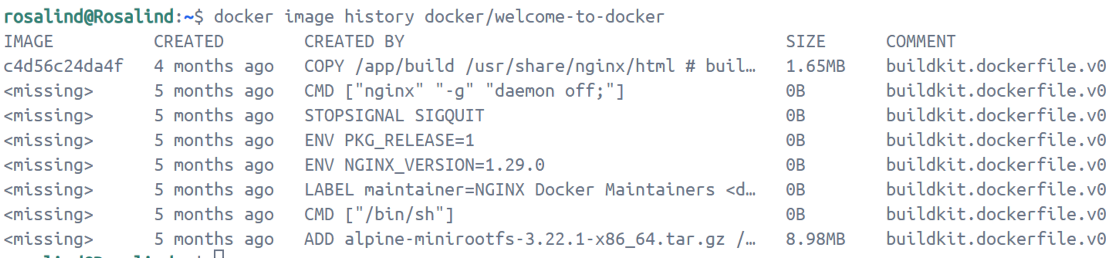
The command shows a list of Docker images currently available on your system. The docker/welcome-to-docker has a total size of approximately 22.2MB, which represented here reflects the uncompressed size of the image, not the download size of the layers.



```
rosalind@Rosalind:~$ docker image ls
REPOSITORY                 TAG       IMAGE ID       CREATED        SIZE
mysql                      8         5cdee9be17b6   8 days ago     1.07GB
gitea/gitea                latest    534428e78fc0   2 weeks ago    259MB
kicbase/stable             v0.0.48   0670263400ef   3 months ago   1.9GB
docker/welcome-to-docker   latest    c4d56c24da4f   4 months ago   22.2MB
```

4. List the image's layers

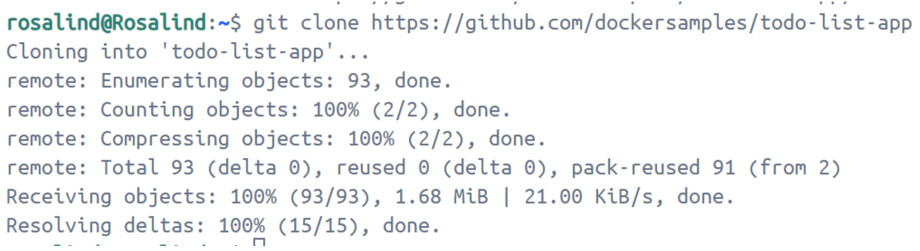
docker image history docker/welcome-to-docker



```
rosalind@Rosalind:~$ docker image history docker/welcome-to-docker
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
c4d56c24da4f   4 months ago   COPY /app/build /usr/share/nginx/html # buil…   1.65MB    buildkit.dockerfile.v0
<missing>      5 months ago   CMD ["nginx" "-g" "daemon off;"]                0B        buildkit.dockerfile.v0
<missing>      5 months ago   STOPSIGNAL SIGQUIT                              0B        buildkit.dockerfile.v0
<missing>      5 months ago   ENV PKG_RELEASE=1                               0B        buildkit.dockerfile.v0
<missing>      5 months ago   ENV NGINX_VERSION=1.29.0                        0B        buildkit.dockerfile.v0
<missing>      5 months ago   LABEL maintainer=NGINX Docker Maintainers <d…   0B        buildkit.dockerfile.v0
<missing>      5 months ago   CMD ["/bin/sh"]                                 0B        buildkit.dockerfile.v0
<missing>      5 months ago   ADD alpine-minirootfs-3.22.1-x86_64.tar.gz /…   8.98MB    buildkit.dockerfile.v0
```

What is Docker Compose?

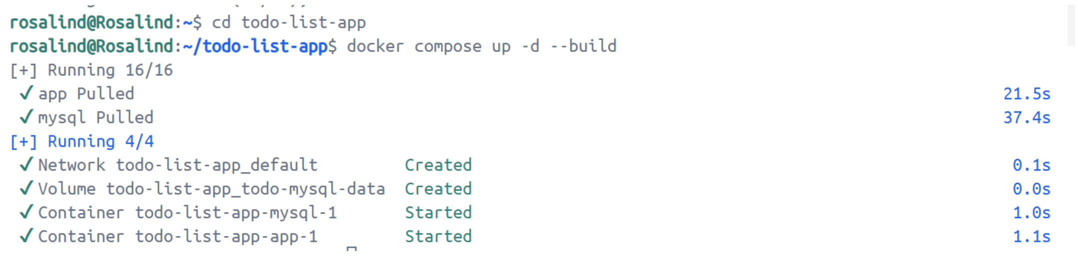
git clone https://github.com/dockersamples/todo-list-app



```
rosalind@Rosalind:~$ git clone https://github.com/dockersamples/todo-list-app
Cloning into 'todo-list-app'...
remote: Enumerating objects: 93, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 93 (delta 0), reused 0 (delta 0), pack-reused 91 (from 2)
Receiving objects: 100% (93/93), 1.68 MiB | 21.00 KiB/s, done.
Resolving deltas: 100% (15/15), done.
```

cd todo-list-app

docker compose up -d --build

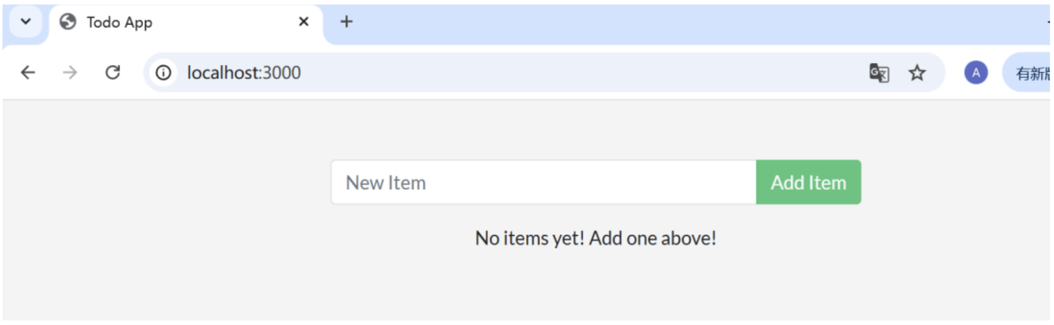


```
rosalind@Rosalind:~$ cd todo-list-app
rosalind@Rosalind:~/todo-list-app$ docker compose up -d --build
[+] Running 16/16
 ✓ app Pulled                                                                21.5s
 ✓ mysql Pulled                                                              37.4s
[+] Running 4/4
 ✓ Network todo-list-app_default          Created                             0.1s
 ✓ Volume todo-list-app_todo-mysql-data   Created                             0.0s
 ✓ Container todo-list-app-mysql-1        Started                             1.0s
 ✓ Container todo-list-app-app-1          Started                             1.1s
```

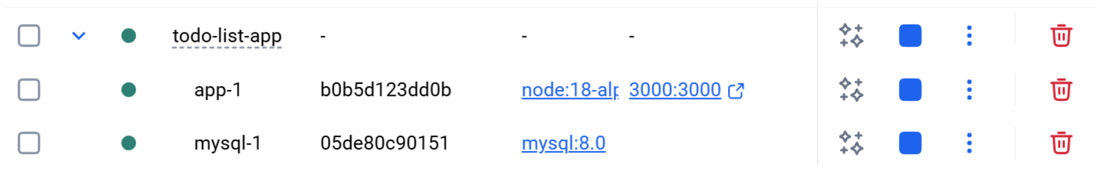
From this screenshot, it shows:

- Two container images were downloaded from Docker Hub - node and MySQL
- A network was created for the application
- A volume was created to persist the database files between container restarts
- Two containers were started with all of their necessary config

Open http://localhost:3000 in the browser.



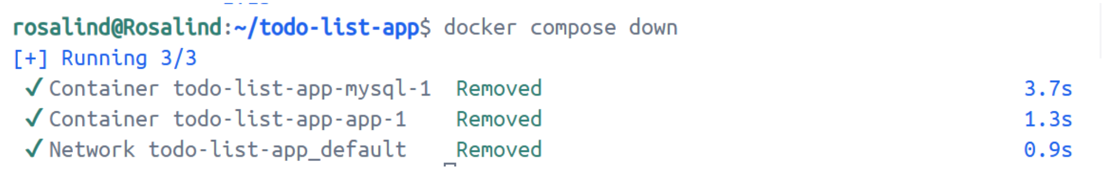
Look at the Docker Desktop GUI, we can see that the containers and dive deeper into their configuration.



☐	●	todo-list-app	-	-	-		
☐	●	app-1	b0b5d123dd0b	node:18-alp	3000:3000		
☐	●	mysql-1	05de80c90151	mysql:8.0			

Tear down the application when we are done.

Enter the command *docker compose down* in CLI to remove everything.

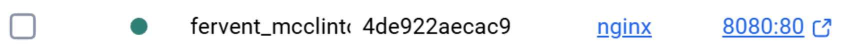


```
rosalind@Rosalind:~/todo-list-app$ docker compose down
[+] Running 3/3
 ✓ Container todo-list-app-mysql-1  Removed                                  3.7s
 ✓ Container todo-list-app-app-1    Removed                                  1.3s
 ✓ Network todo-list-app_default    Removed                                  0.9s
```

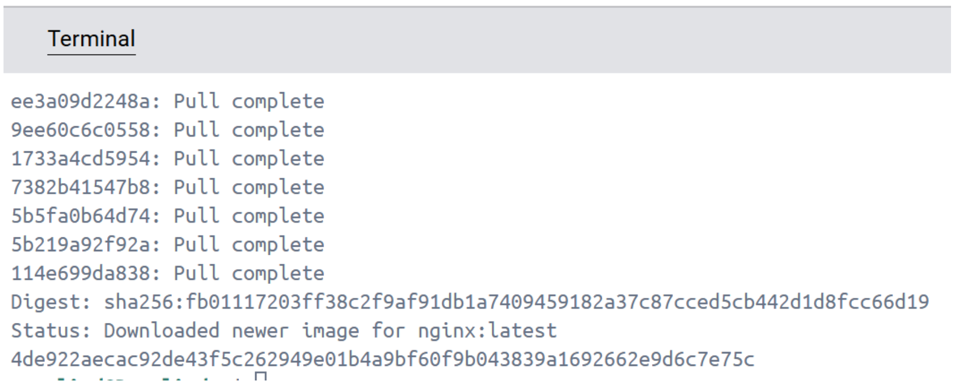
Publishing and exposing ports

1. Publish the container's port 80 to host port 8080

   docker run -d -p 8080:80 nginx



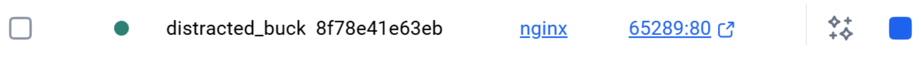
fervent_mcclint 4de922aecac9 nginx 8080:80



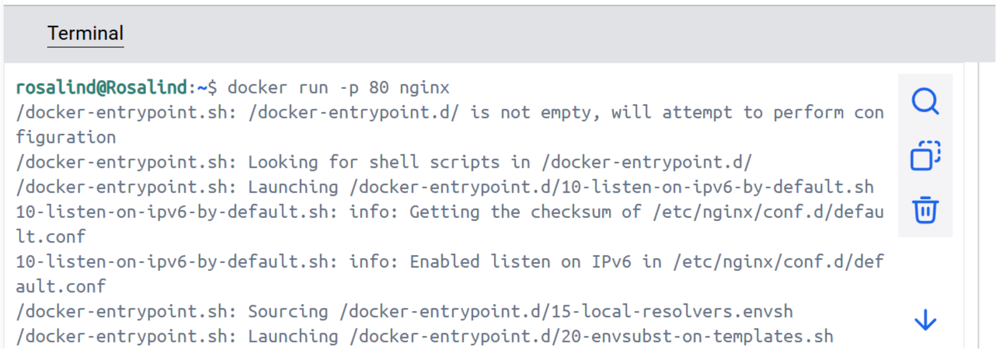
```
ee3a09d2248a: Pull complete
9ee60c6c0558: Pull complete
1733a4cd5954: Pull complete
7382b41547b8: Pull complete
5b5fa0b64d74: Pull complete
5b219a92f92a: Pull complete
114e699da838: Pull complete
Digest: sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
Status: Downloaded newer image for nginx:latest
4de922aecac92de43f5c262949e01b4a9bf60f9b043839a1692662e9d6c7e75c
```

2. publish the container's port 80 onto an ephemeral port on the host:

   docker run -p 80 nginx

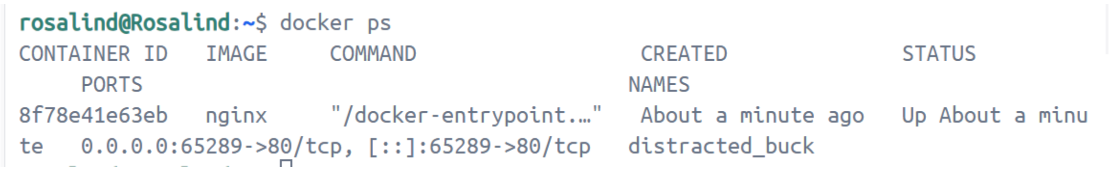


distracted_buck 8f78e41e63eb nginx 65289:80



```
rosalind@Rosalind:~$ docker run -p 80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform con
figuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/defau
lt.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/def
ault.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
```

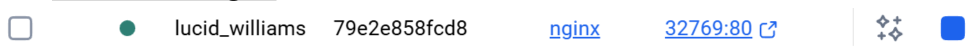
Then use *docker ps* to show the chosen port once the container is running



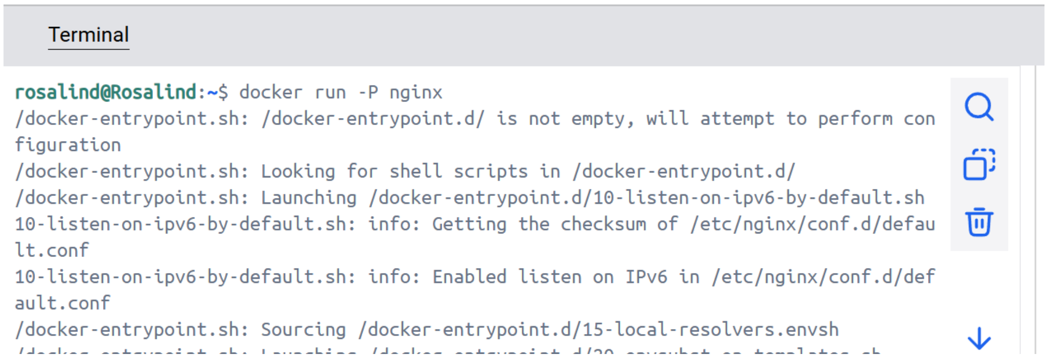
```
rosalind@Rosalind:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS
     PORTS                                     NAMES
8f78e41e63eb   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minu
te   0.0.0.0:65289->80/tcp, [::]:65289->80/tcp   distracted_buck
```

3. publish all exposed ports to ephemeral ports with the -P or --publish-all flag

   docker run -P nginx



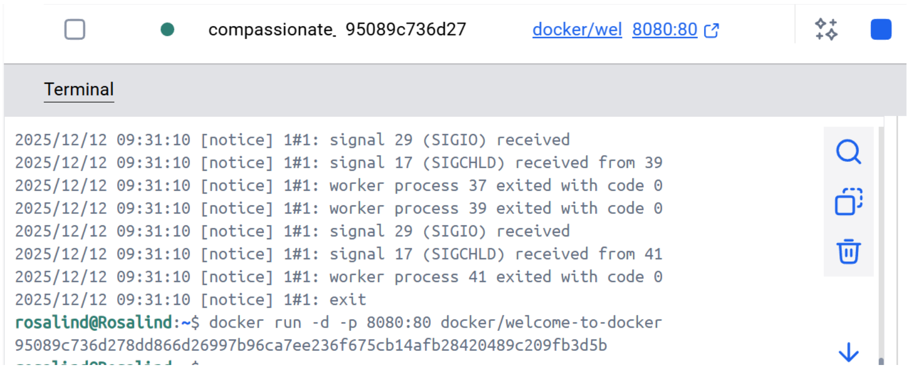
lucid_williams 79e2e858fcd8 nginx 32769:80



```
rosalind@Rosalind:~$ docker run -P nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform con
figuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/defau
lt.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/def
ault.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
```

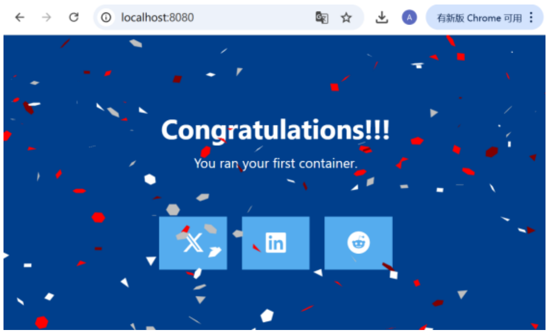
Use the Docker CLI

Start a new container: *docker run -d -p 8080:80 docker/welcome-to-docker*



```
2025/12/12 09:31:10 [notice] 1#1: signal 29 (SIGIO) received
2025/12/12 09:31:10 [notice] 1#1: signal 17 (SIGCHLD) received from 39
2025/12/12 09:31:10 [notice] 1#1: worker process 37 exited with code 0
2025/12/12 09:31:10 [notice] 1#1: worker process 39 exited with code 0
2025/12/12 09:31:10 [notice] 1#1: signal 29 (SIGIO) received
2025/12/12 09:31:10 [notice] 1#1: signal 17 (SIGCHLD) received from 41
2025/12/12 09:31:10 [notice] 1#1: worker process 41 exited with code 0
2025/12/12 09:31:10 [notice] 1#1: exit
rosalind@Rosalind:~$ docker run -d -p 8080:80 docker/welcome-to-docker
95089c736d278dd866d26997b96ca7ee236f675cb14afb28420489c209fb3d5b
```

Open the website by either selecting the link in the Port(s) column of the container like http://localhost:8080.

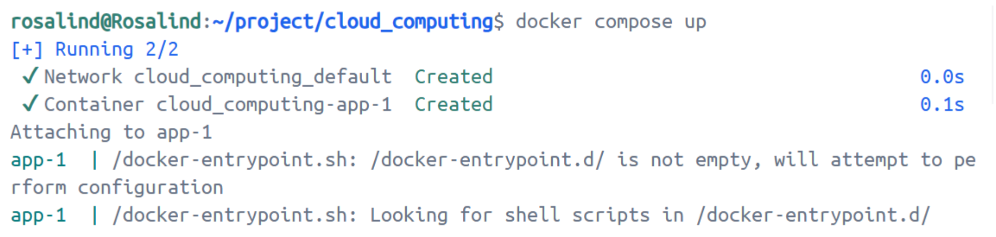


Use the Docker CLI

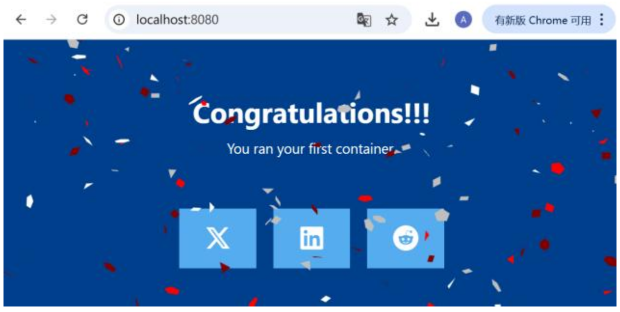
Create a compose.yaml file with the dollowing contents:

```
services:
  app:
    image: docker/welcome-to-docker
    ports:
      - 8080:80
```

Then enter the *docker run* command to run a container,



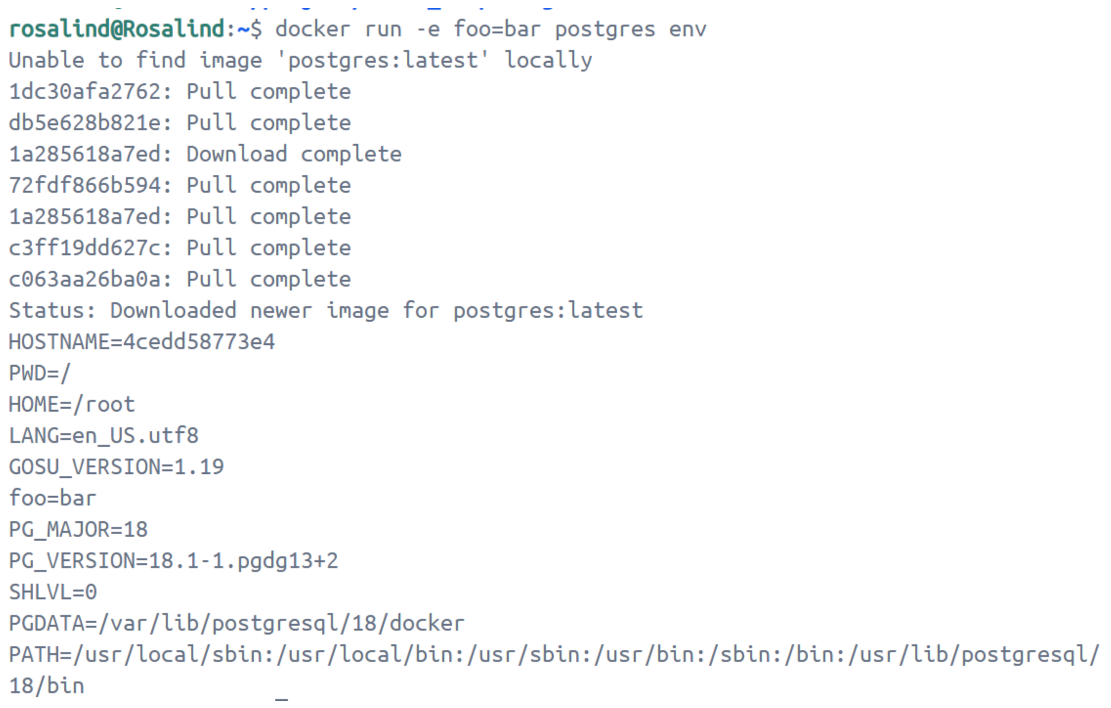
```
rosalind@Rosalind:~/project/cloud_computing$ docker compose up
[+] Running 2/2
 ✓Network cloud_computing_default  Created                                   0.0s
 ✓Container cloud_computing-app-1  Created                                   0.1s
Attaching to app-1
app-1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to pe
rform configuration
app-1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
```



Overriding container defaults

1. Sets an environment variable foo inside the container with the value bar.

   docker run -e foo=bar postgres env

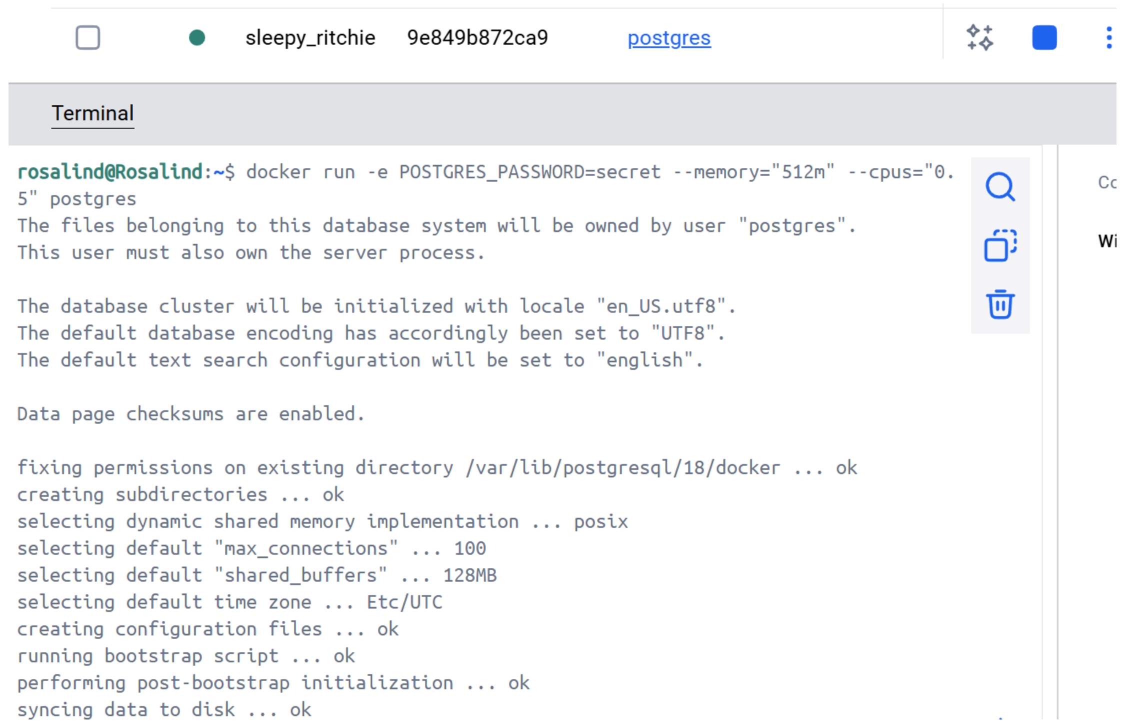


```
rosalind@Rosalind:~$ docker run -e foo=bar postgres env
Unable to find image 'postgres:latest' locally
1dc30afa2762: Pull complete
db5e628b821e: Pull complete
1a285618a7ed: Download complete
72fdf866b594: Pull complete
1a285618a7ed: Pull complete
c3ff19dd627c: Pull complete
c063aa26ba0a: Pull complete
Status: Downloaded newer image for postgres:latest
HOSTNAME=4cedd58773e4
PWD=/
HOME=/root
LANG=en_US.utf8
GOSU_VERSION=1.19
foo=bar
PG_MAJOR=18
PG_VERSION=18.1-1.pgdg13+2
SHLVL=0
PGDATA=/var/lib/postgresql/18/docker
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/
18/bin
```

2. Use the --memory and --cpus flags with the docker run command to restrict how much CPU and memory a container can use.

   docker run -e POSTGRES_PASSWORD=secret --memory="512m" --cpus="0.5" postgres

   Here the command limits container memory usage to 512 MB and defines the CPU quota of 0.5 for half a core.



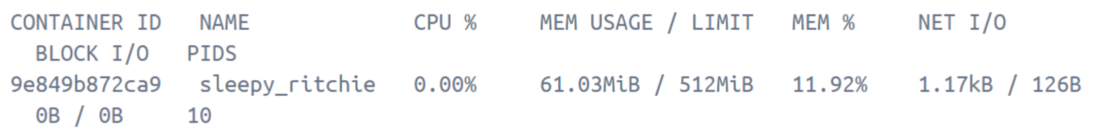
```
rosalind@Rosalind:~$ docker run -e POSTGRES_PASSWORD=secret --memory="512m" --cpus="0.
5" postgres
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are enabled.

fixing permissions on existing directory /var/lib/postgresql/18/docker ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default "max_connections" ... 100
selecting default "shared_buffers" ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
```

3. Use the *docker stats* command to monitor the real-time resource usage of running containers. This helps to understand whether the allocated resources are sufficient or need adjustment.



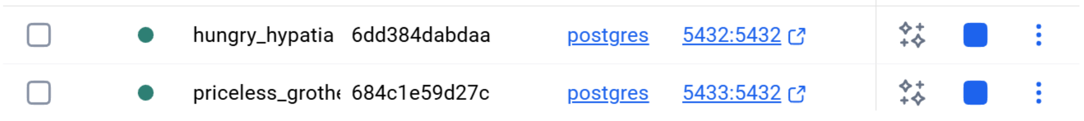
```
CONTAINER ID   NAME             CPU %     MEM USAGE / LIMIT   MEM %     NET I/O
  BLOCK I/O   PIDS
9e849b872ca9   sleepy_ritchie   0.00%     61.03MiB / 512MiB   11.92%    1.17kB / 126B
  0B / 0B     10
```

4. Start the Postgres database in the background, listening on the standard container port 5432 and mapped to port 5432 on the host machine.

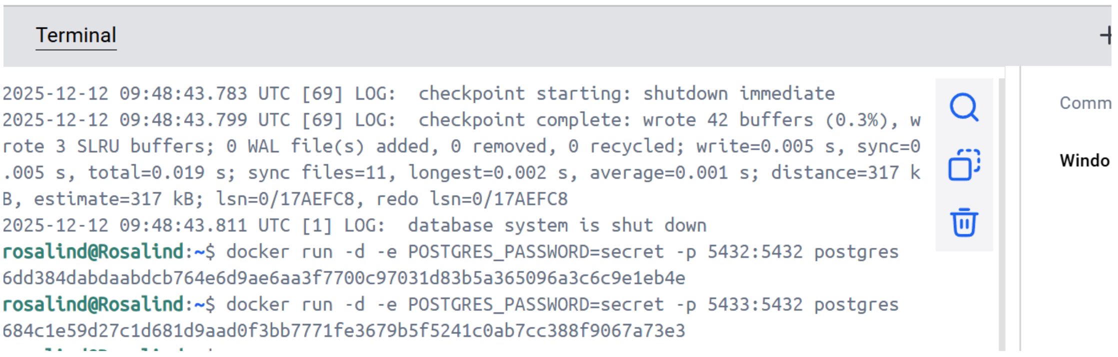
   docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgres

5. Start another Postgres container in the background, listening on the standard postgres port 5432 in the container, but mapped to port 5433 on the host machine.

   docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgres



```
hungry_hypatia   6dd384dabdaa   postgres   5432:5432
priceless_grothe 684c1e59d27c   postgres   5433:5432
```



Terminal

```
2025-12-12 09:48:43.783 UTC [69] LOG:  checkpoint starting: shutdown immediate
2025-12-12 09:48:43.799 UTC [69] LOG:  checkpoint complete: wrote 42 buffers (0.3%), w
rote 3 SLRU buffers; 0 WAL file(s) added, 0 removed, 0 recycled; write=0.005 s, sync=0
.005 s, total=0.019 s; sync files=11, longest=0.002 s, average=0.001 s; distance=317 k
B, estimate=317 kB; lsn=0/17AEFC8, redo lsn=0/17AEFC8
2025-12-12 09:48:43.811 UTC [1] LOG:  database system is shut down
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgres
6dd384dabdaabdcb764e6d9ae6aa3f7700c97031d83b5a365096a3c6c9e1eb4e
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgres
684c1e59d27c1d681d9aad0f3bb7771fe3679b5f5241c0ab7cc388f9067a73e3
```

6. Create a new custom network

   docker network create mynetwork

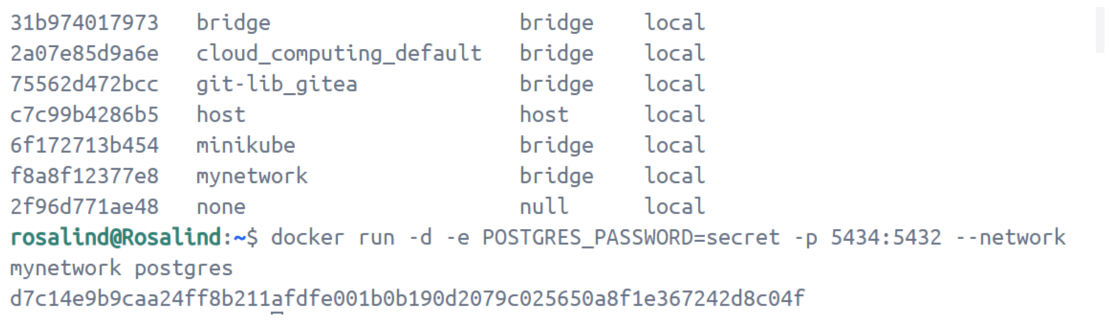
7. Verify the network by listing all networks, including the newly created "mynetwork".

   docker network ls

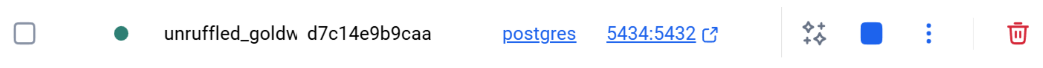
8. Connect Postgres to the custom network

   docker run -d -e POSTGRES_PASSWORD=secret -p 5434:5432 --network mynetwork postgres

   This will start Postgres container in the background, mapped to the host port 5434 and attached to the mynetwork network.



```
31b974017973   bridge                   bridge    local
2a07e85d9a6e   cloud_computing_default  bridge    local
75562d472bcc   git-lib_gitea            bridge    local
c7c99b4286b5   host                     host      local
6f172713b454   minikube                 bridge    local
f8a8f12377e8   mynetwork                bridge    local
2f96d771ae48   none                     null      local
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret -p 5434:5432 --network
mynetwork postgres
d7c14e9b9caa24ff8b211afdfe001b0b190d2079c025650a8f1e367242d8c04f
```

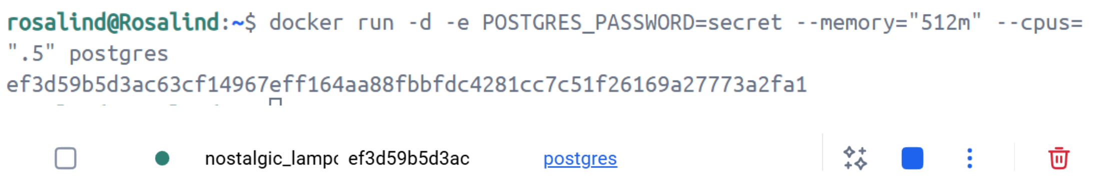


```
unruffled_goldw  d7c14e9b9caa   postgres   5434:5432
```

9. Run a container whose CPU and memory are restricted by flags --memory and --cpus

docker run -d -e POSTGRES_PASSWORD=secret --memory="512m" --cpus=".5" postgres

The --cpus flag specifies the CPU quota for the container. Here, it's set to half a CPU core (0.5) whereas the --memory flag specifies the memory limit for the container. In this case, it's set to 512 MB.



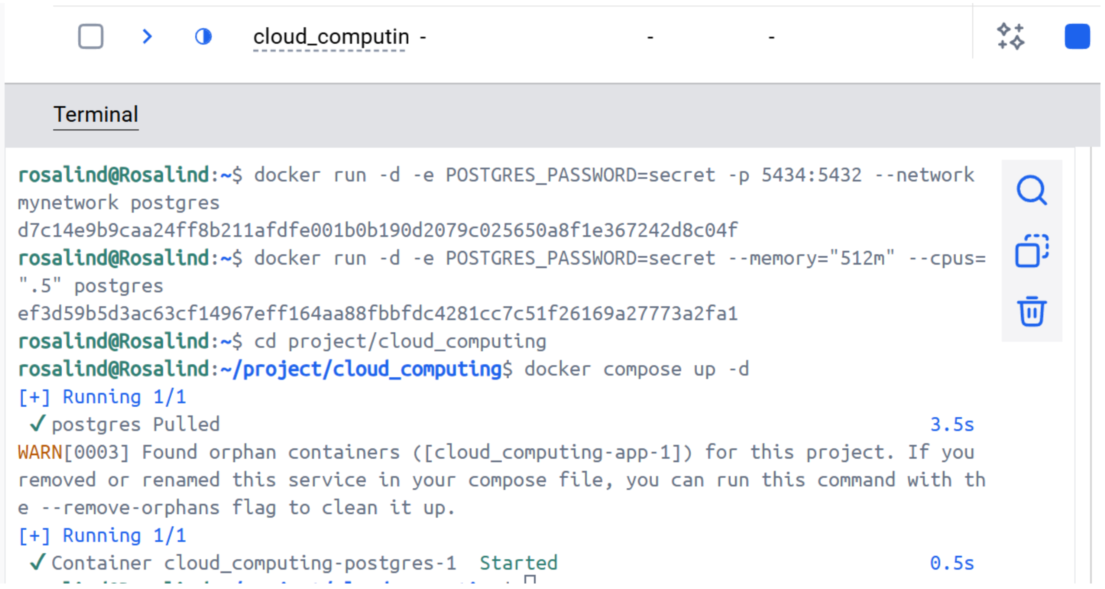
```
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret --memory="512m" --cpus=
".5" postgres
ef3d59b5d3ac63cf14967eff164aa88fbbfdc4281cc7c51f26169a27773a2fa1
```

10. Use Docker compose to override the default commands (CMD) or entry points (ENTRYPOINT) defined in a Docker image

```
services:
  postgres:
    image: postgres:18
    entrypoint: ["docker-entrypoint.sh", "postgres"]
    command: ["-h", "localhost", "-p", "5432"]
    environment:
      POSTGRES_PASSWORD: secret
```

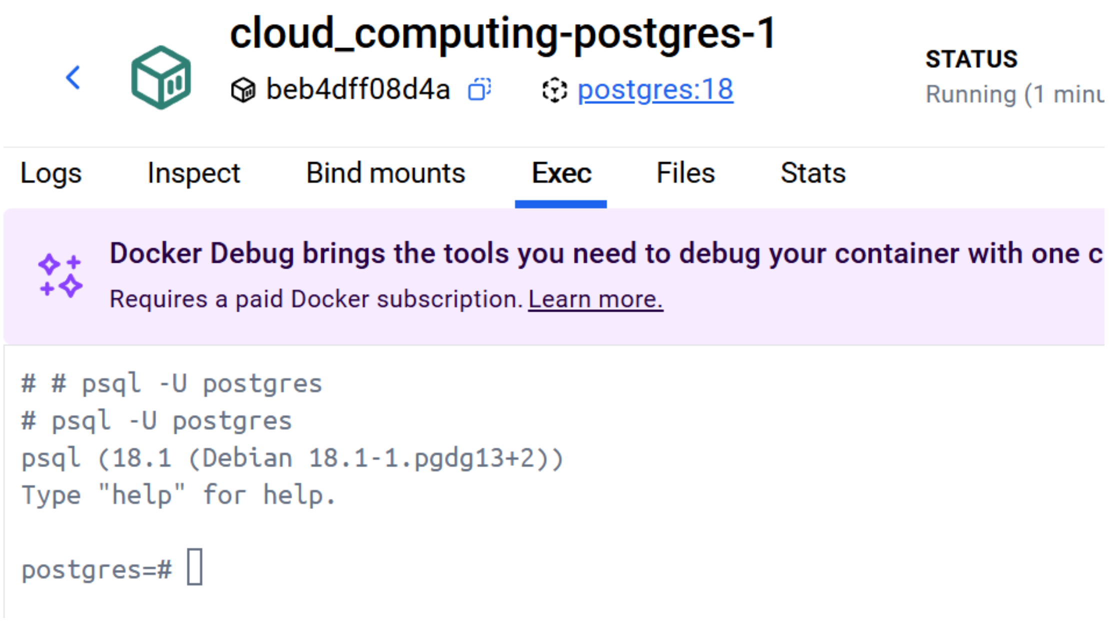
The Compose file defines a service named postgres that uses the official Postgres image, sets an entrypoint script, and starts the container with password authentication.

11. Bring up the service by running *docker compose up -d*



```
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret -p 5434:5432 --network
mynetwork postgres
d7c14e9b9caa24ff8b211afdfe001b0b190d2079c025650a8f1e367242d8c04f
rosalind@Rosalind:~$ docker run -d -e POSTGRES_PASSWORD=secret --memory="512m" --cpus=
".5" postgres
ef3d59b5d3ac63cf14967eff164aa88fbbfdc4281cc7c51f26169a27773a2fa1
rosalind@Rosalind:~$ cd project/cloud_computing
rosalind@Rosalind:~/project/cloud_computing$ docker compose up -d
[+] Running 1/1
 ✓ postgres Pulled                                                          3.5s
WARN[0003] Found orphan containers ([cloud_computing-app-1]) for this project. If you
removed or renamed this service in your compose file, you can run this command with th
e --remove-orphans flag to clean it up.
[+] Running 1/1
 ✓ Container cloud_computing-postgres-1  Started                            0.5s
```

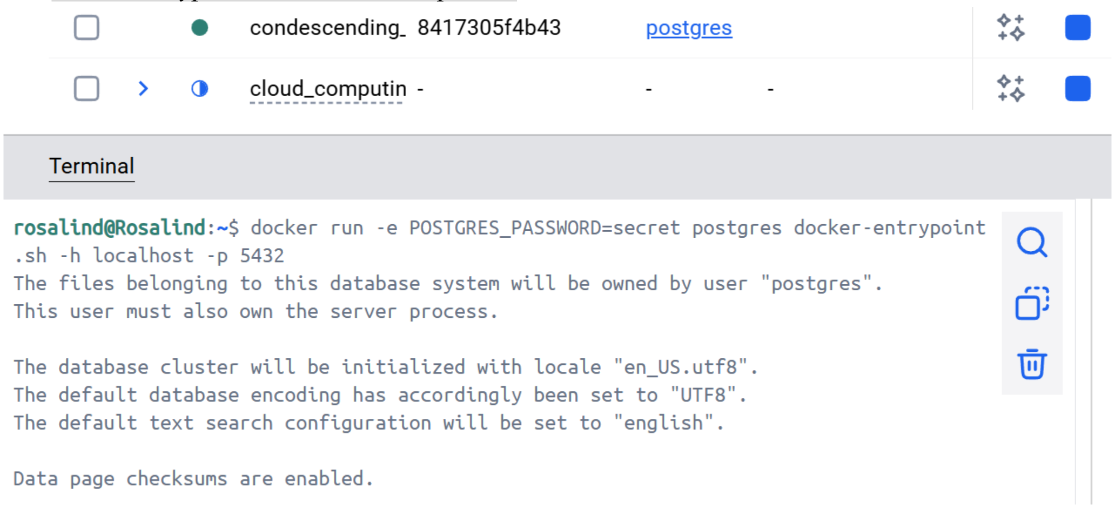
12. Open the Docker Desktop Dashboard, select the **Postgres** container and select **Exec** to enter into the container shell.


cloud_computing-postgres-1
beb4dff08d4a
postgres:18
STATUS
Running (1 minu
Logs
Inspect
Bind mounts
Exec
Files
Stats
Docker Debug brings the tools you need to debug your container with one c
Requires a paid Docker subscription. Learn more.

```
# # psql -U postgres
# psql -U postgres
psql (18.1 (Debian 18.1-1.pgdg13+2))
Type "help" for help.

postgres=#
```

13. Override defaults directly using *docker run -e POSTGRES_PASSWORD=secret postgres docker-entrypoint.sh -h localhost -p 5432*



```
rosalind@Rosalind:~$ docker run -e POSTGRES_PASSWORD=secret postgres docker-entrypoint
.sh -h localhost -p 5432
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are enabled.
```

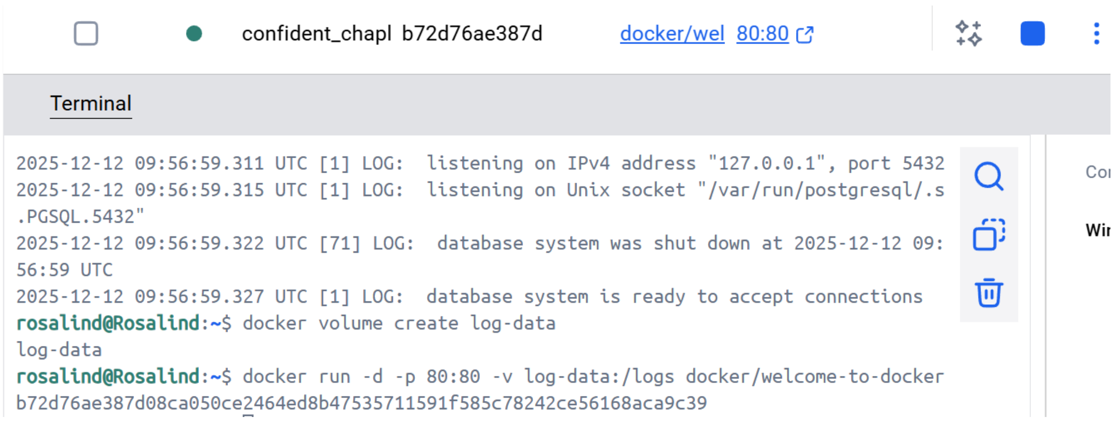
Persisting container data

1. Create a volume named as log-data

   docker volume create log-data

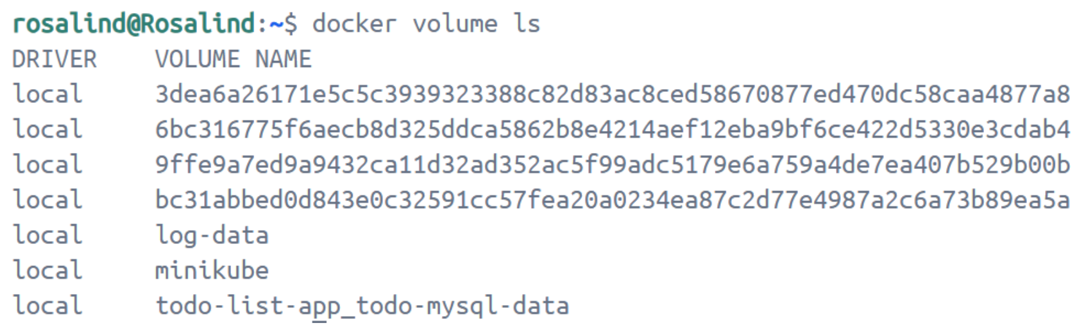
2. When starting a container with the following command, the volume will be mounted (or attached) into the container at /logs

   docker run -d -p 80:80 -v log-data:/logs docker/welcome-to-docker



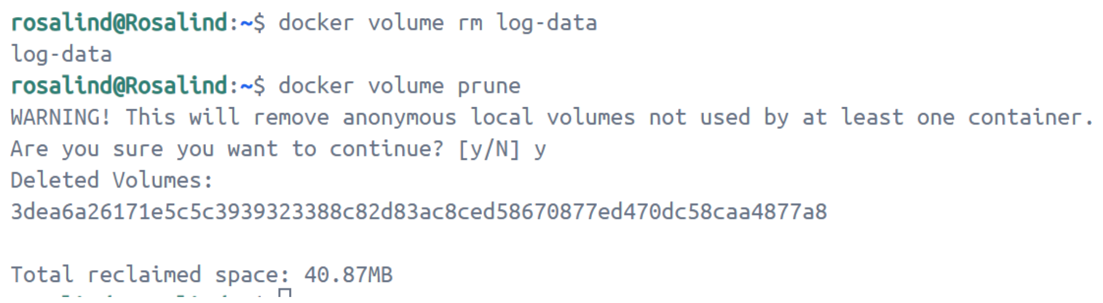
```
2025-12-12 09:56:59.311 UTC [1] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2025-12-12 09:56:59.315 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s
.PGSQL.5432"
2025-12-12 09:56:59.322 UTC [71] LOG:  database system was shut down at 2025-12-12 09:
56:59 UTC
2025-12-12 09:56:59.327 UTC [1] LOG:  database system is ready to accept connections
rosalind@Rosalind:~$ docker volume create log-data
log-data
rosalind@Rosalind:~$ docker run -d -p 80:80 -v log-data:/logs docker/welcome-to-docker
b72d76ae387d08ca050ce2464ed8b47535711591f585c78242ce56168aca9c39
```

3. *list all volumes*: list all volumes



```
rosalind@Rosalind:~$ docker volume ls
DRIVER    VOLUME NAME
local     3dea6a26171e5c5c3939323388c82d83ac8ced58670877ed470dc58caa4877a8
local     6bc316775f6aecb8d325ddca5862b8e4214aef12eba9bf6ce422d5330e3cdab4
local     9ffe9a7ed9a9432ca11d32ad352ac5f99adc5179e6a759a4de7ea407b529b00b
local     bc31abbed0d843e0c32591cc57fea20a0234ea87c2d77e4987a2c6a73b89ea5a
local     log-data
local     minikube
local     todo-list-app_todo-mysql-data
```

4. *docker volume rm <volume-name-or-id>*: remove a volume (only works when the volume is not attached to any containers)
5. *docker volume prune*: remove all unused (unattached) volumes



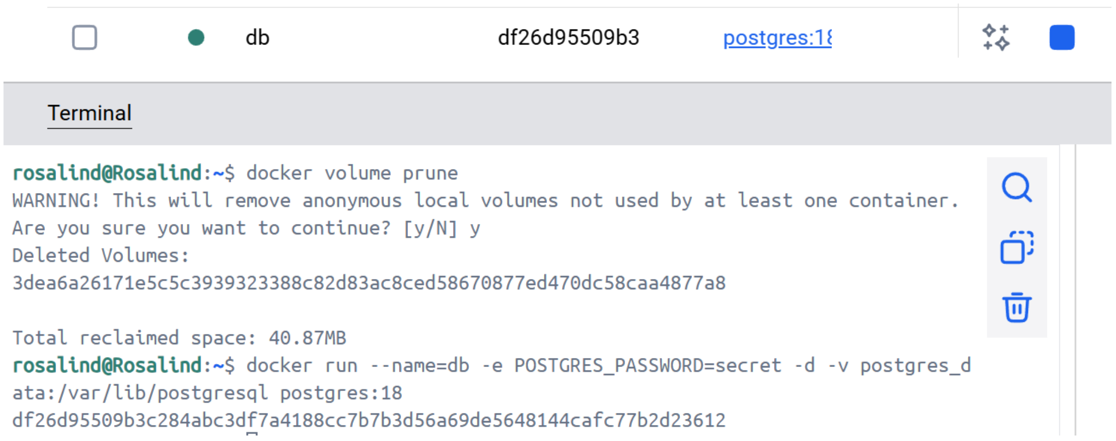
```
rosalind@Rosalind:~$ docker volume rm log-data
log-data
rosalind@Rosalind:~$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
3dea6a26171e5c5c3939323388c82d83ac8ced58670877ed470dc58caa4877a8

Total reclaimed space: 40.87MB
```

6. Start a container using the Postgres image.

   docker run --name=db -e POSTGRES_PASSWORD=secret -d -v postgres_data:/var/lib/postgresql postgres:18

   This will start the database in the background, configure it with a password, and attach a volume to the directory PostgreSQL will persist the database files.

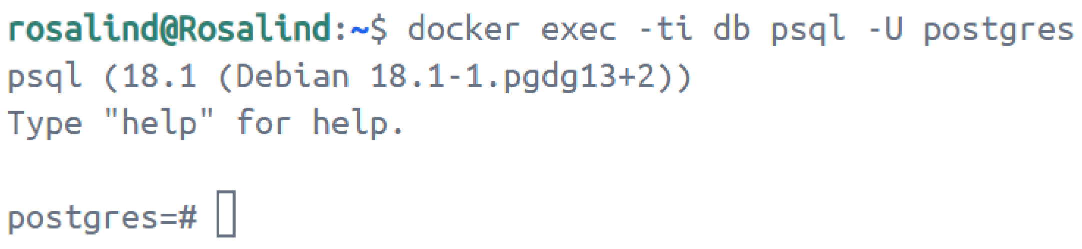


```
rosalind@Rosalind:~$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
3dea6a26171e5c5c3939323388c82d83ac8ced58670877ed470dc58caa4877a8

Total reclaimed space: 40.87MB
rosalind@Rosalind:~$ docker run --name=db -e POSTGRES_PASSWORD=secret -d -v postgres_d
ata:/var/lib/postgresql postgres:18
df26d95509b3c284abc3df7a4188cc7b7b3d56a69de5648144cafc77b2d23612
```

7. Connect to the database

    docker exec -ti db psql -U postgres



```
rosalind@Rosalind:~$ docker exec -ti db psql -U postgres
psql (18.1 (Debian 18.1-1.pgdg13+2))
Type "help" for help.

postgres=#
```

8. In the PostgreSQL command line, run the following to create a database table and insert two records:

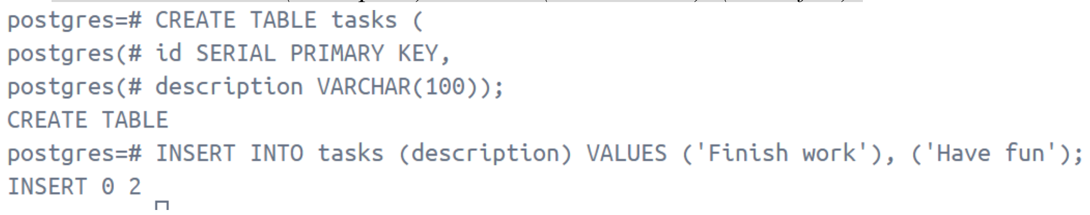
    CREATE TABLE tasks (

    id SERIAL PRIMARY KEY,

    description VARCHAR(100)

    );

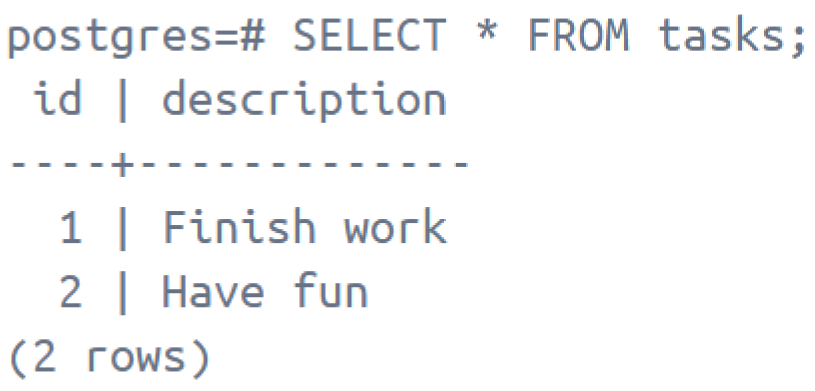
    INSERT INTO tasks (description) VALUES ('Finish work'), ('Have fun');



```
postgres=# CREATE TABLE tasks (
postgres(# id SERIAL PRIMARY KEY,
postgres(# description VARCHAR(100));
CREATE TABLE
postgres=# INSERT INTO tasks (description) VALUES ('Finish work'), ('Have fun');
INSERT 0 2
```

9. Verify the data is in the database by running the following in the PostgreSQL command line

    *SELECT * FROM tasks;*

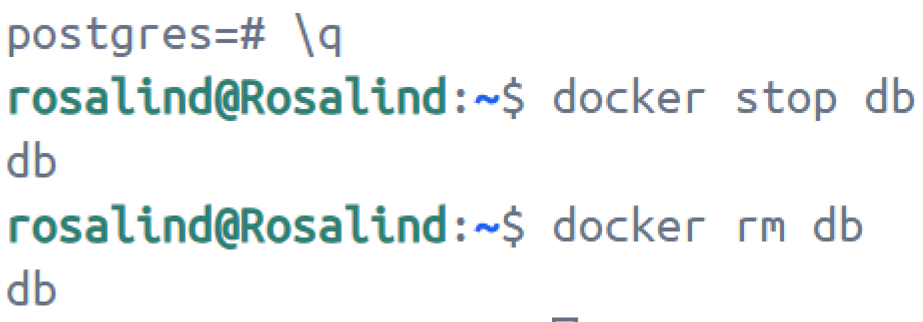


```
postgres=# SELECT * FROM tasks;
 id | description
----+-------------
  1 | Finish work
  2 | Have fun
(2 rows)
```

10. \q: exit out of the PostgreSQL shell
11. Stop and remove the database container. Note: even though the container has been deleted, the data is persisted in the postgres_data volume.

    docker stop db

    docker rm db



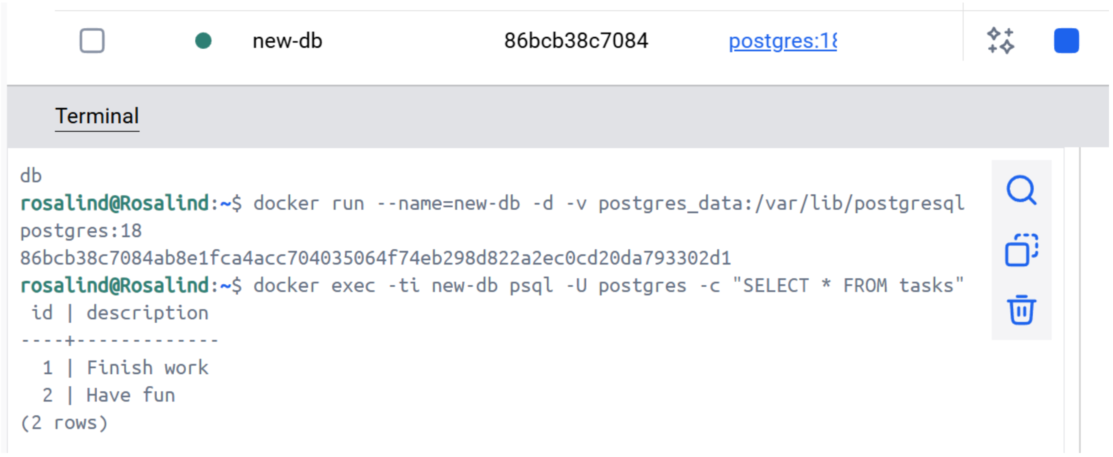
```
postgres=# \q
rosalind@Rosalind:~$ docker stop db
db
rosalind@Rosalind:~$ docker rm db
db
```

12. Start a new container, attaching the same volume with the persisted data

    docker run --name=new-db -d -v postgres_data:/var/lib/postgresql postgres:18

13. Verify the database still has the records

    *docker exec -ti new-db psql -U postgres -c "SELECT * FROM tasks"*

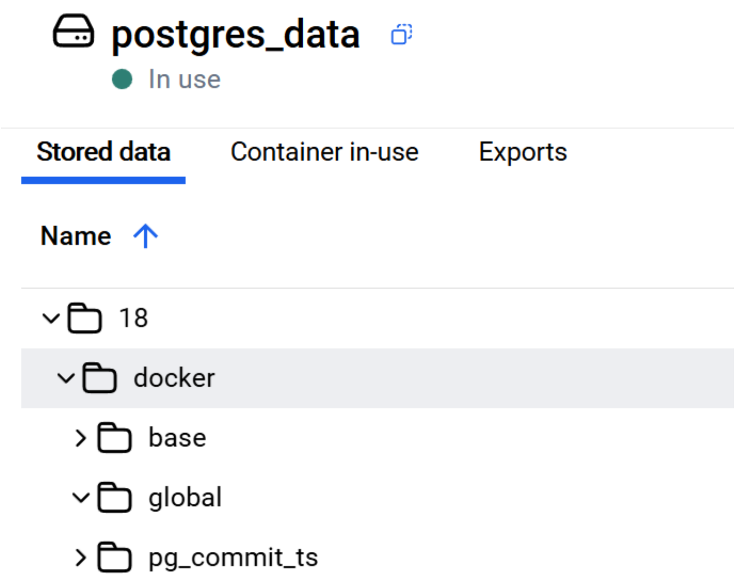


new-db 86bcb38c7084 postgres:1

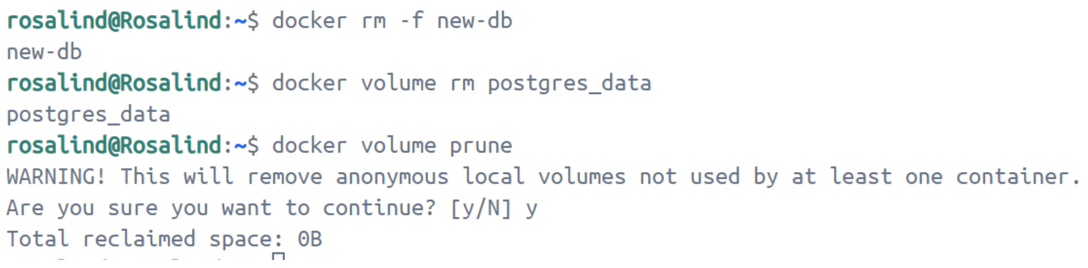
Terminal

```
db
rosalind@Rosalind:~$ docker run --name=new-db -d -v postgres_data:/var/lib/postgresql
postgres:18
86bcb38c7084ab8e1fca4acc704035064f74eb298d822a2ec0cd20da793302d1
rosalind@Rosalind:~$ docker exec -ti new-db psql -U postgres -c "SELECT * FROM tasks"
 id | description
----+-------------
  1 | Finish work
  2 | Have fun
(2 rows)
```

14. View volume contents by the Docker Desktop Dashboard



15. Remove volumes:
    - *docker rm -f new-db:* -f means that it stop the container first and then remove it
    - *docker volume rm postgres_data*
    - *docker volume prune*: remove all unused volumes



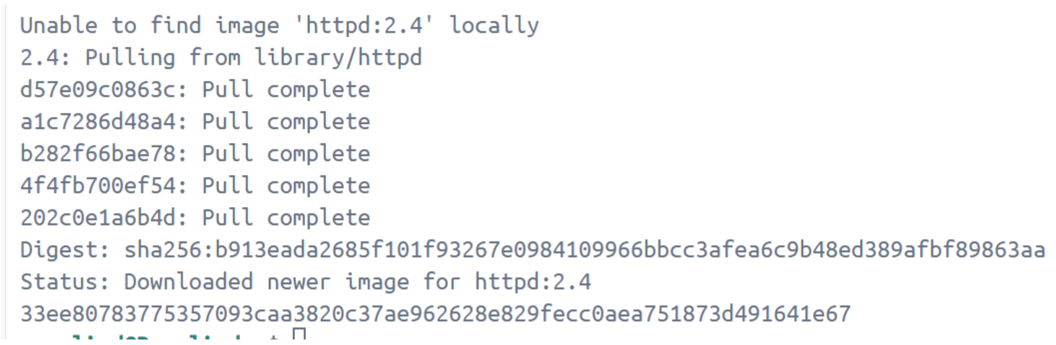
```
rosalind@Rosalind:~$ docker rm -f new-db
new-db
rosalind@Rosalind:~$ docker volume rm postgres_data
postgres_data
rosalind@Rosalind:~$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
```

Sharing local files with containers

1. Start a container using the httpd image

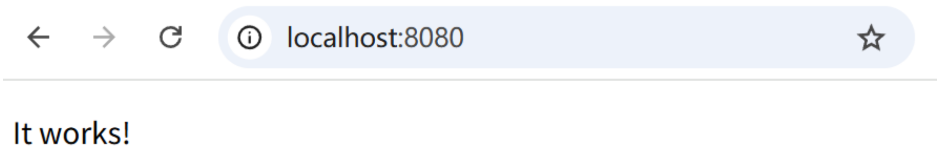
   docker run -d -p 8080:80 --name my_site httpd:2.4

   This will start the **httpd** service in the background, and publish the webpage to port **8080** on the host.



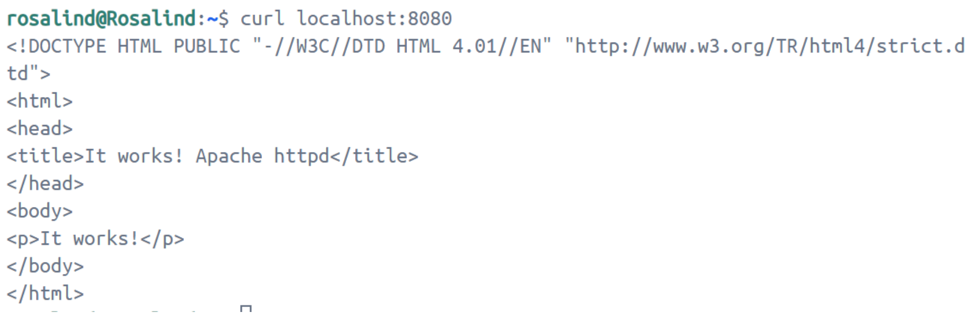
```
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
d57e09c0863c: Pull complete
a1c7286d48a4: Pull complete
b282f66bae78: Pull complete
4f4fb700ef54: Pull complete
202c0e1a6b4d: Pull complete
Digest: sha256:b913eada2685f101f93267e0984109966bbcc3afea6c9b48ed389afbf89863aa
Status: Downloaded newer image for httpd:2.4
33ee80783775357093caa3820c37ae962628e829fecc0aea751873d491641e67
```

2. Open http://localhost:8080 or use the curl command to verify if it's working fine or not

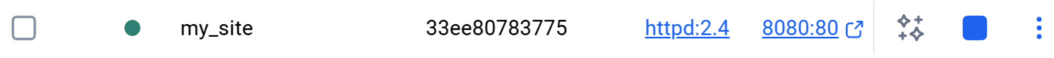


localhost:8080

It works!



```
rosalind@Rosalind:~$ curl localhost:8080
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.d
td">
<html>
<head>
<title>It works! Apache httpd</title>
</head>
<body>
<p>It works!</p>
</body>
</html>
```



my_site 33ee80783775 httpd:2.4 8080:80

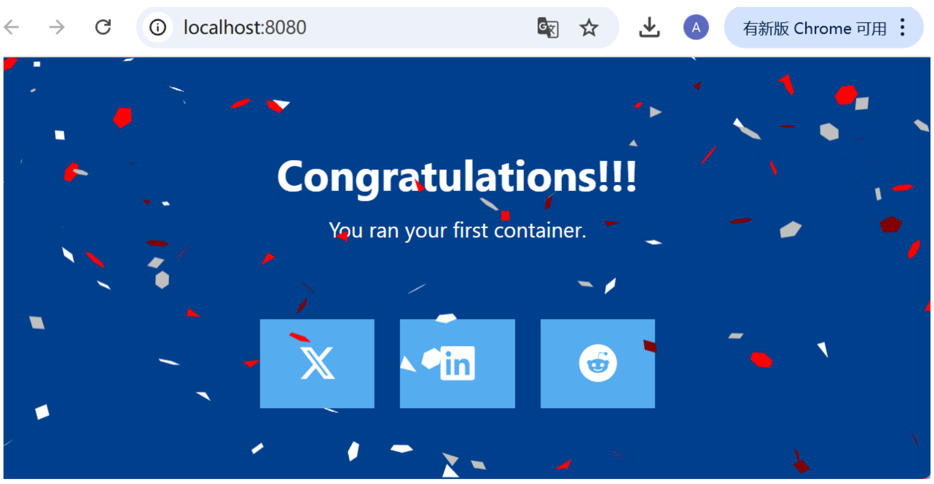
3. Create a directory and create index.html in it.

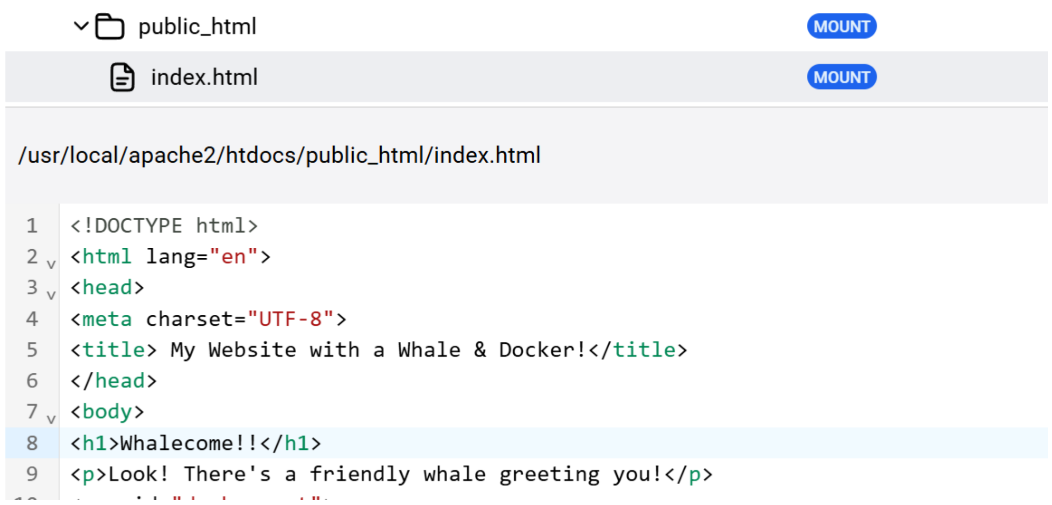
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title> My Website with a Whale & Docker!</title>
</head>
<body>
<h1>Whalecome!!</h1>
<p>Look! There's a friendly whale greeting you!</p>
<pre id="docker-art">
    ##         .
```

```
  ## ## ##        ==
 ## ## ## ## ##       ===
 /""""""""""""""""""\___/ ===
{                       /  ===-
\______ O           __/
\    \         __/
 \____\_______/

Hello from Docker!
</pre>
</body>
</html>
```

Then run the command *docker run -d --name my_site -p 8080:80 -v .:/usr/local/apache2/htdocs/ httpd:2.4*.

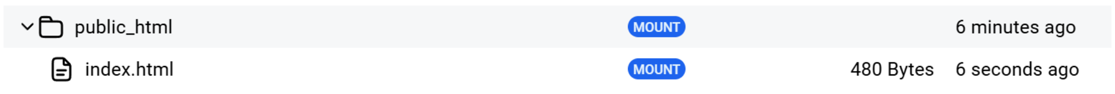




/usr/local/apache2/htdocs/public_html/index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title> My Website with a Whale & Docker!</title>
</head>
<body>
<h1>Whalecome!!</h1>
<p>Look! There's a friendly whale greeting you!</p>
```

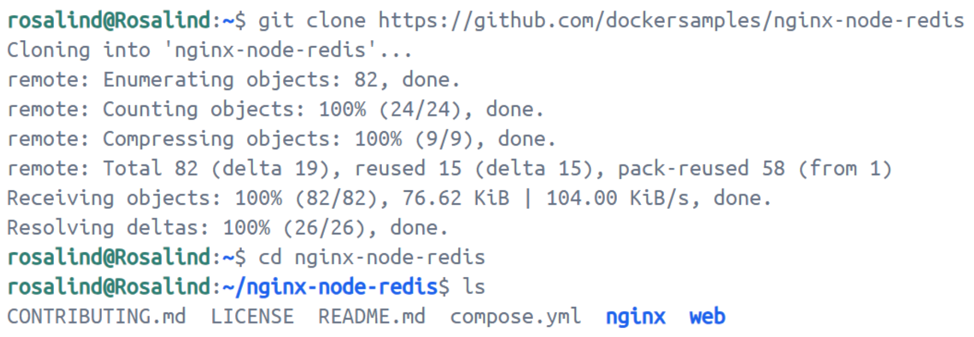
Then delete it, and create a new HTML file on the main computer. You'll find the file reappears under the Files tab under Containers on the Docker Desktop Dashboard.



public_html	MOUNT		6 minutes ago
index.html	MOUNT	480 Bytes	6 seconds ago

Multi-container applications

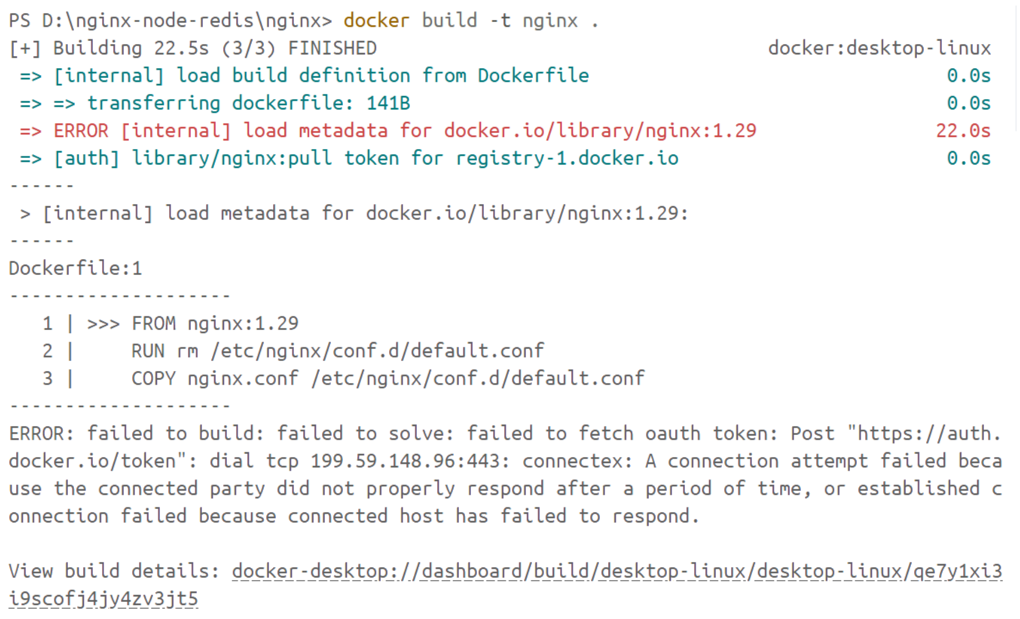
1. git clone https://github.com/dockersamples/nginx-node-redis
2. cd nginx-node-redis



```
rosalind@Rosalind:~$ git clone https://github.com/dockersamples/nginx-node-redis
Cloning into 'nginx-node-redis'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 82 (delta 19), reused 15 (delta 15), pack-reused 58 (from 1)
Receiving objects: 100% (82/82), 76.62 KiB | 104.00 KiB/s, done.
Resolving deltas: 100% (26/26), done.
rosalind@Rosalind:~$ cd nginx-node-redis
rosalind@Rosalind:~/nginx-node-redis$ ls
CONTRIBUTING.md  LICENSE  README.md  compose.yml  nginx  web
```

3. docker build -t nginx .

   Encountering a network connection problem that won't download



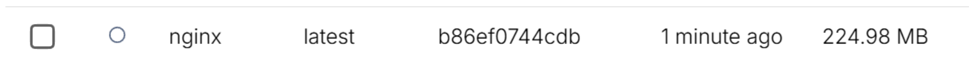
```
PS D:\nginx-node-redis\nginx> docker build -t nginx .
[+] Building 22.5s (3/3) FINISHED                                   docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 141B                                                0.0s
 => ERROR [internal] load metadata for docker.io/library/nginx:1.29                22.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                        0.0s
------
 > [internal] load metadata for docker.io/library/nginx:1.29:
------
Dockerfile:1
--------------------
   1 | >>> FROM nginx:1.29
   2 |     RUN rm /etc/nginx/conf.d/default.conf
   3 |     COPY nginx.conf /etc/nginx/conf.d/default.conf
--------------------
ERROR: failed to build: failed to solve: failed to fetch oauth token: Post "https://auth.
docker.io/token": dial tcp 199.59.148.96:443: connectex: A connection attempt failed beca
use the connected party did not properly respond after a period of time, or established c
onnection failed because connected host has failed to respond.

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qe7y1xi3
i9scofj4jy4zv3jt5
```

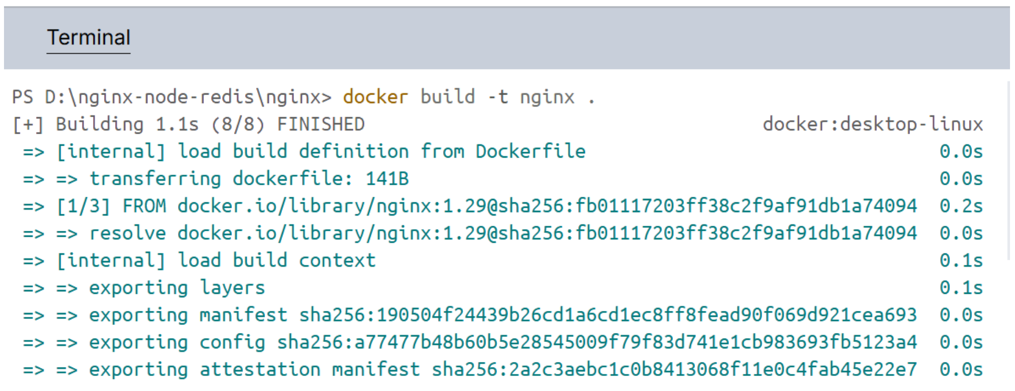
   So I used a mirror website, based on https://mirrors.ustc.edu.cn/help/dockerhub.html

4. navigate into the nginx directory to build the image

   docker build -t nginx .



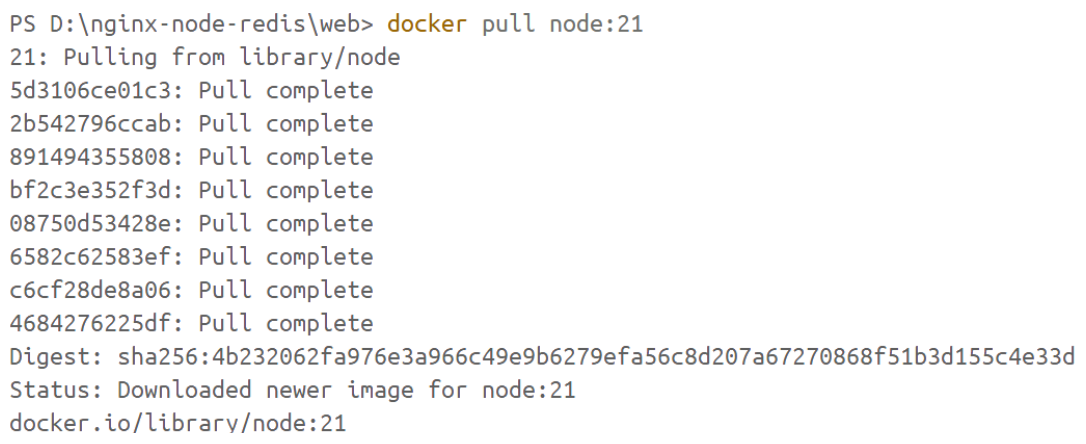
☐	○	nginx	latest	b86ef0744cdb	1 minute ago	224.98 MB



Terminal

```
PS D:\nginx-node-redis\nginx> docker build -t nginx .
[+] Building 1.1s (8/8) FINISHED                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 141B                                                0.0s
 => [1/3] FROM docker.io/library/nginx:1.29@sha256:fb01117203ff38c2f9af91db1a74094  0.2s
 => => resolve docker.io/library/nginx:1.29@sha256:fb01117203ff38c2f9af91db1a74094  0.0s
 => [internal] load build context                                                   0.1s
 => => exporting layers                                                             0.1s
 => => exporting manifest sha256:190504f24439b26cd1a6cd1ec8ff8fead90f069d921cea693  0.0s
 => => exporting config sha256:a77477b48b60b5e28545009f79f83d741e1cb983693fb5123a4  0.0s
 => => exporting attestation manifest sha256:2a2c3aebc1c0b8413068f11e0c4fab45e22e7  0.0s
```

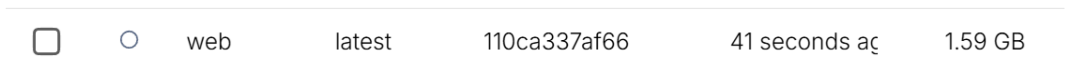
Similarly, first navigate to the web folder and enter `*docker pull node:21*`.



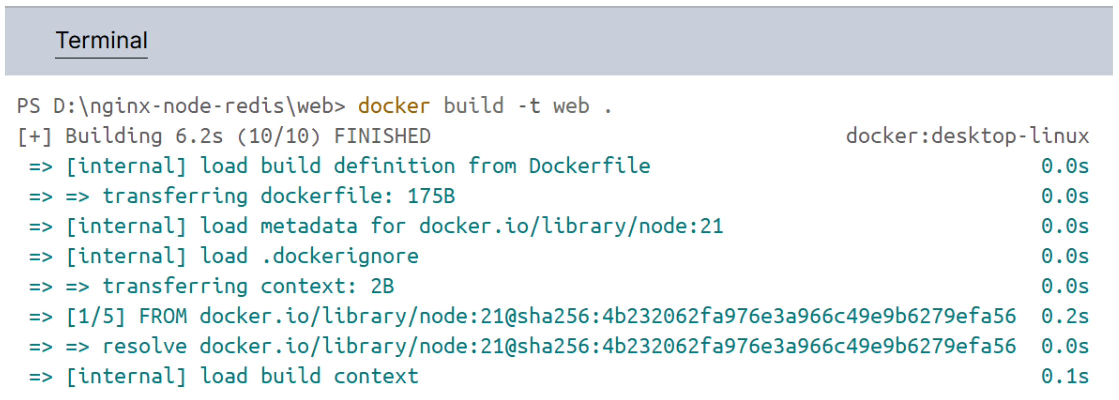
```
PS D:\nginx-node-redis\web> docker pull node:21
21: Pulling from library/node
5d3106ce01c3: Pull complete
2b542796ccab: Pull complete
891494355808: Pull complete
bf2c3e352f3d: Pull complete
08750d53428e: Pull complete
6582c62583ef: Pull complete
c6cf28de8a06: Pull complete
4684276225df: Pull complete
Digest: sha256:4b232062fa976e3a966c49e9b6279efa56c8d207a67270868f51b3d155c4e33d
Status: Downloaded newer image for node:21
docker.io/library/node:21
```

5. Then Navigate into the web directory and run the following command to build the first web image:

    docker build -t web .



☐	○	web	latest	110ca337af66	41 seconds ag	1.59 GB



Terminal

```
PS D:\nginx-node-redis\web> docker build -t web .
[+] Building 6.2s (10/10) FINISHED                                      docker:desktop-linux
 => [internal] load build definition from Dockerfile                                    0.0s
 => => transferring dockerfile: 175B                                                    0.0s
 => [internal] load metadata for docker.io/library/node:21                              0.0s
 => [internal] load .dockerignore                                                       0.0s
 => => transferring context: 2B                                                         0.0s
 => [1/5] FROM docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56   0.2s
 => => resolve docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56   0.0s
 => [internal] load build context                                                       0.1s
```

6. create a network for them all to communicate through.

    docker network create sample-app

7. Start the Redis container, which will attach it to the previously created network and create a network alias (useful for DNS lookups)

    docker run -d --name redis --network sample-app --network-alias redis redis

8. Start the first web container

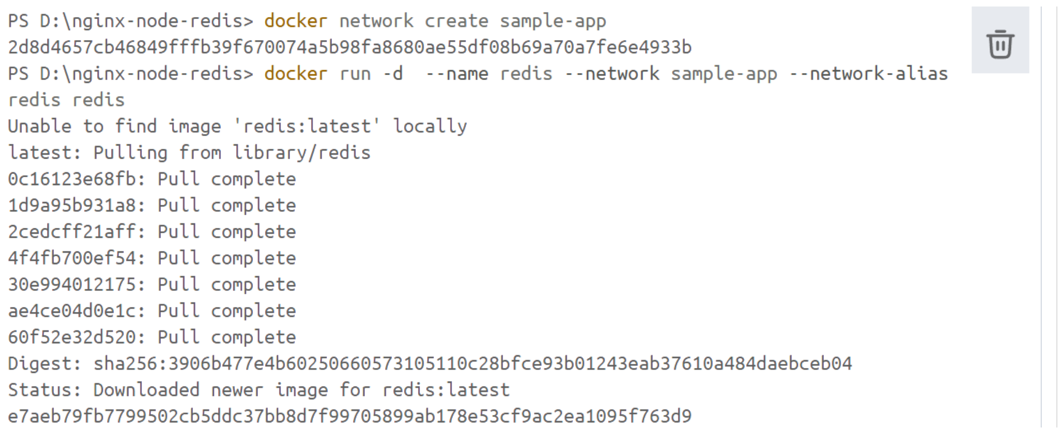
    docker run -d --name web1 -h web1 --network sample-app --network-alias web1 web

9. Start the second web container

    docker run -d --name web2 -h web2 --network sample-app --network-alias web2 web

10. Start the Nginx container

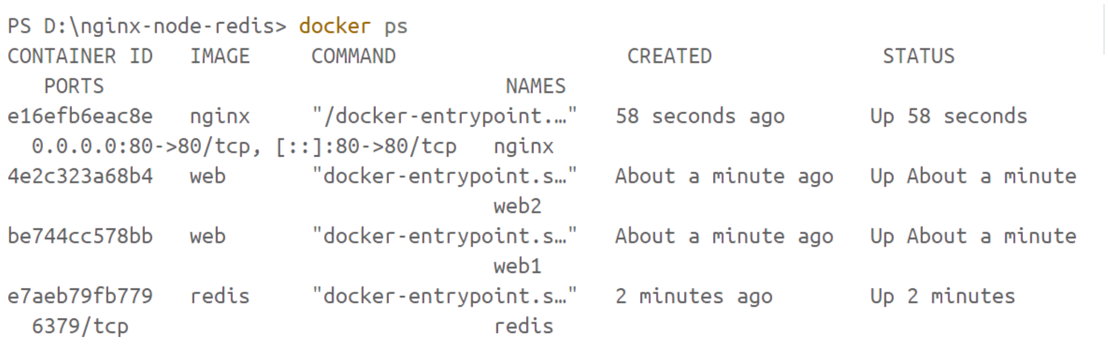
    docker run -d --name nginx --network sample-app   -p 80:80 nginx



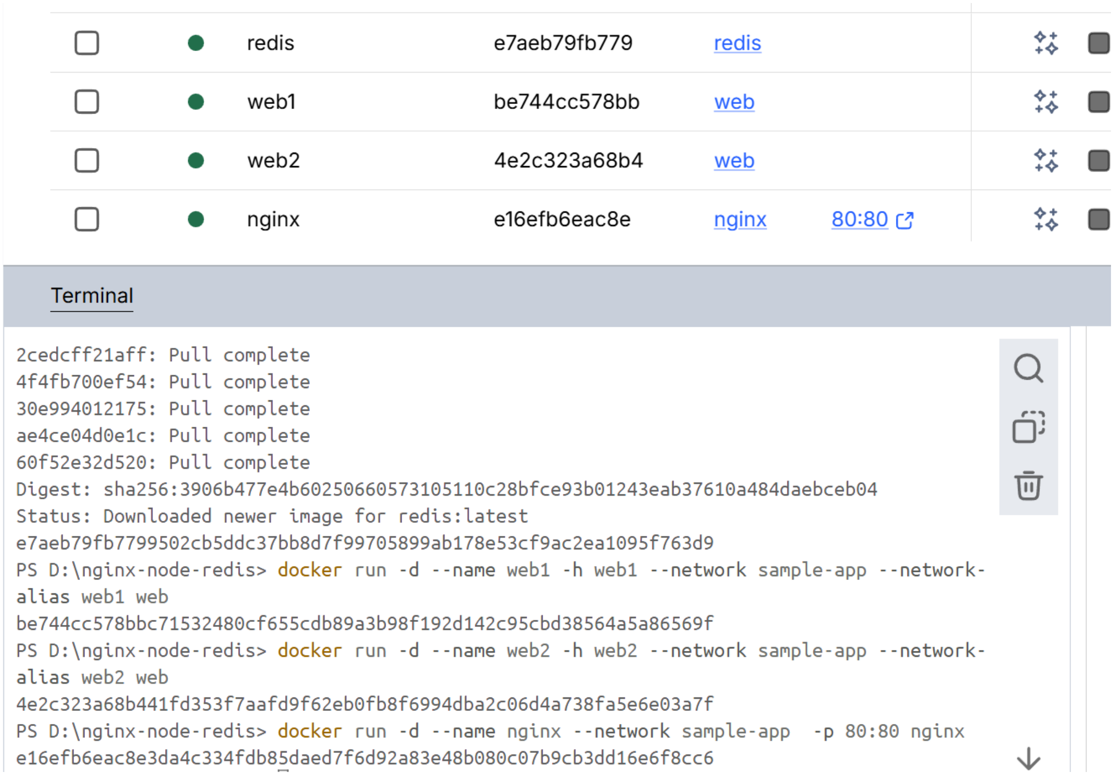
```
PS D:\nginx-node-redis> docker network create sample-app
2d8d4657cb46849fffb39f670074a5b98fa8680ae55df08b69a70a7fe6e4933b
PS D:\nginx-node-redis> docker run -d  --name redis --network sample-app --network-alias
redis redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
0c16123e68fb: Pull complete
1d9a95b931a8: Pull complete
2cedcff21aff: Pull complete
4f4fb700ef54: Pull complete
30e994012175: Pull complete
ae4ce04d0e1c: Pull complete
60f52e32d520: Pull complete
Digest: sha256:3906b477e4b60250660573105110c28bfce93b01243eab37610a484daebceb04
Status: Downloaded newer image for redis:latest
e7aeb79fb7799502cb5ddc37bb8d7f99705899ab178e53cf9ac2ea1095f763d9
```

11. Verify the containers

docker ps

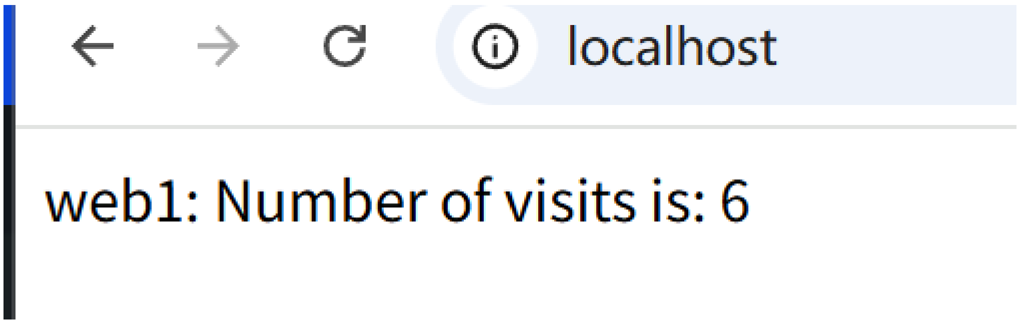


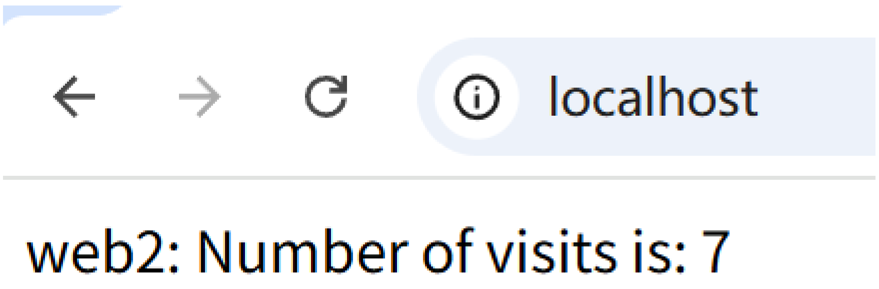
```
PS D:\nginx-node-redis> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS
   PORTS                                NAMES
e16efb6eac8e   nginx     "/docker-entrypoint.…"   58 seconds ago       Up 58 seconds
  0.0.0.0:80->80/tcp, [::]:80->80/tcp   nginx
4e2c323a68b4   web       "docker-entrypoint.s…"   About a minute ago   Up About a minute
                                       web2
be744cc578bb   web       "docker-entrypoint.s…"   About a minute ago   Up About a minute
                                       web1
e7aeb79fb779   redis     "docker-entrypoint.s…"   2 minutes ago        Up 2 minutes
  6379/tcp                             redis
```

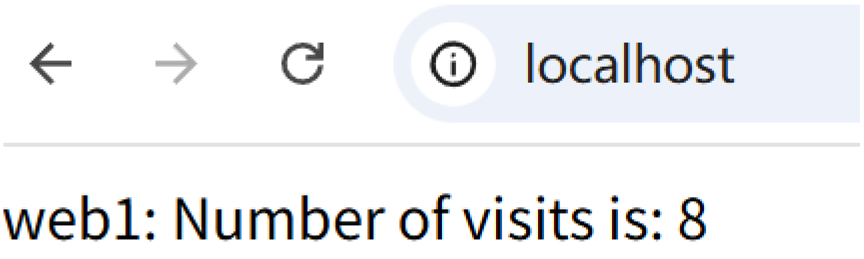


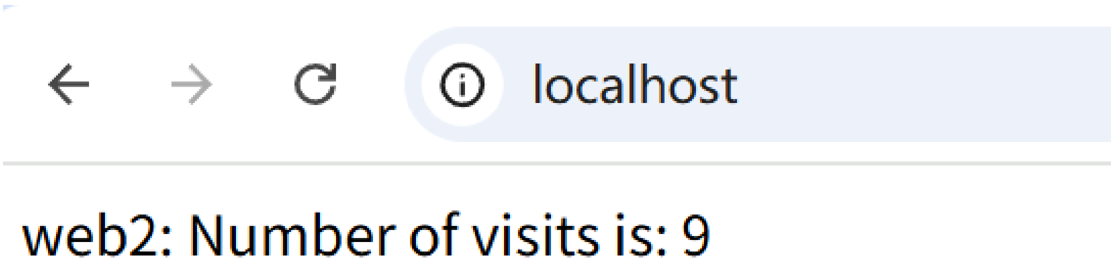
☐	●	redis	e7aeb79fb779	redis		
☐	●	web1	be744cc578bb	web		
☐	●	web2	4e2c323a68b4	web		
☐	●	nginx	e16efb6eac8e	nginx	80:80	

```
2cedcff21aff: Pull complete
4f4fb700ef54: Pull complete
30e994012175: Pull complete
ae4ce04d0e1c: Pull complete
60f52e32d520: Pull complete
Digest: sha256:3906b477e4b60250660573105110c28bfce93b01243eab37610a484daebceb04
Status: Downloaded newer image for redis:latest
e7aeb79fb7799502cb5ddc37bb8d7f99705899ab178e53cf9ac2ea1095f763d9
PS D:\nginx-node-redis> docker run -d --name web1 -h web1 --network sample-app --network-
alias web1 web
be744cc578bbc71532480cf655cdb89a3b98f192d142c95cbd38564a5a86569f
PS D:\nginx-node-redis> docker run -d --name web2 -h web2 --network sample-app --network-
alias web2 web
4e2c323a68b441fd353f7aafd9f62eb0fb8f6994dba2c06d4a738fa5e6e03a7f
PS D:\nginx-node-redis> docker run -d --name nginx --network sample-app  -p 80:80 nginx
e16efb6eac8e3da4c334fdb85daed7f6d92a83e48b080c07b9cb3dd16e6f8cc6
```


```
localhost
web1: Number of visits is: 6
```


```
localhost
web2: Number of visits is: 7
```

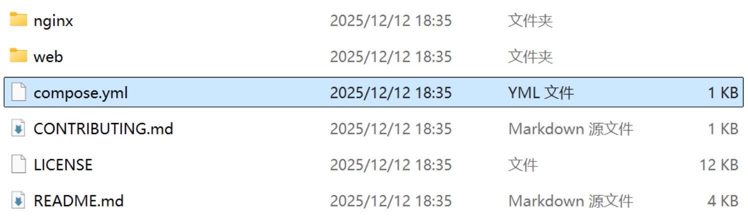

```
localhost
web1: Number of visits is: 8
```


```
localhost
web2: Number of visits is: 9
```

Then refresh the page several times to see the host that's handling the request and the total number of requests. The output shows consecutive increments for both the web1 and web2 containers and the actual counter value stored in Redis is updated only after the response is sent back to the client.

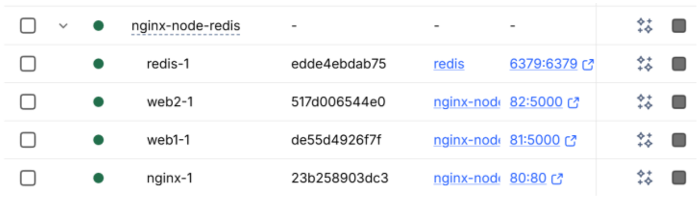
12. compose.yaml

```
services:
  redis:
    image: redis
    ports:
      - '6379:6379'
  web1:
    restart: on-failure
    build: ./web
    hostname: web1
    ports:
      - '81:5000'
  web2:
    restart: on-failure
    build: ./web
    hostname: web2
    ports:
      - '82:5000'
  nginx:
    build: ./nginx
    ports:
    - '80:80'
    depends_on:
    - web1
    - web2
```

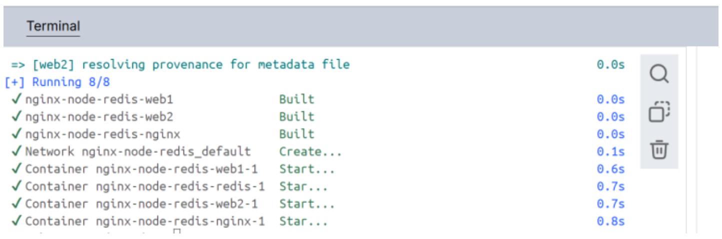


nginx	2025/12/12 18:35	文件夹	
web	2025/12/12 18:35	文件夹	
compose.yml	2025/12/12 18:35	YML 文件	1 KB
CONTRIBUTING.md	2025/12/12 18:35	Markdown 源文件	1 KB
LICENSE	2025/12/12 18:35	文件	12 KB
README.md	2025/12/12 18:35	Markdown 源文件	4 KB

13. docker compose up -d --build



nginx-node-redis	-	-	-
redis-1	edde4ebdab75	redis	6379:6379
web2-1	517d006544e0	nginx-nod	82:5000
web1-1	de55d4926f7f	nginx-nod	81:5000
nginx-1	23b258903dc3	nginx-nod	80:80



Terminal

```
 => [web2] resolving provenance for metadata file                    0.0s
[+] Running 8/8
 ✓ nginx-node-redis-web1                Built                          0.0s
 ✓ nginx-node-redis-web2                Built                          0.0s
 ✓ nginx-node-redis-nginx               Built                          0.0s
 ✓ Network nginx-node-redis_default     Create...                      0.1s
 ✓ Container nginx-node-redis-web1-1    Start...                       0.6s
 ✓ Container nginx-node-redis-redis-1   Star...                        0.7s
 ✓ Container nginx-node-redis-web2-1    Start...                       0.7s
 ✓ Container nginx-node-redis-nginx-1   Star...                        0.8s
```