Lab 2

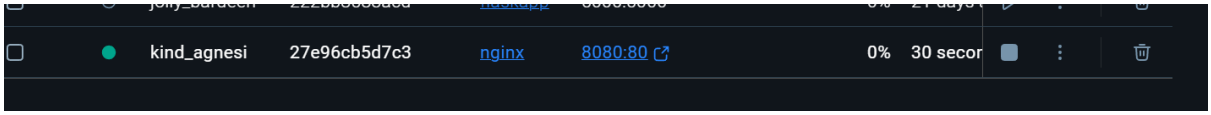
Mirella Glowinska

C0027574

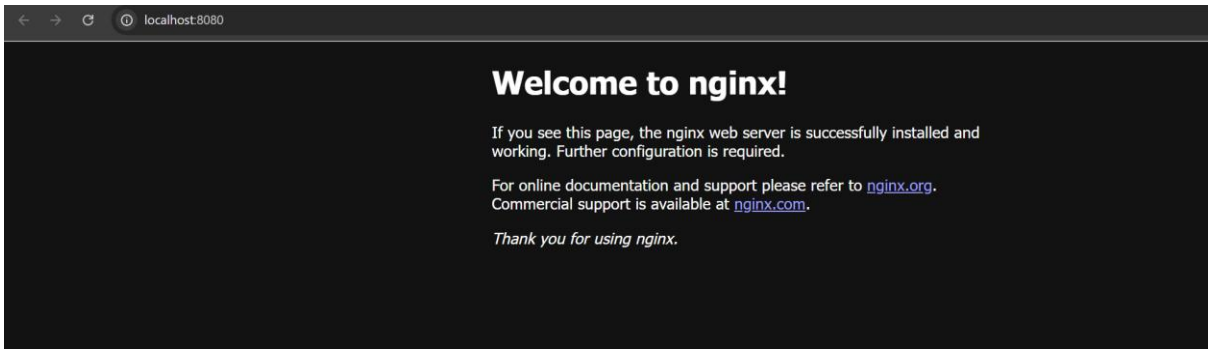
Tutorial 1:

Start a **NGINX** web server inside a Docker container. It allows access to the application via a web browser.

```
C:\Users\Mirella>docker run -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
103f50cb3e9f: Download complete
9dd21ad5a4a6: Download complete
513c3649bb14: Download complete
7cf63256a31a: Download complete
d014f92d532d: Download complete
bf9acace214a: Download complete
943ea0f0c2e4: Download complete
Digest: sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c9496
Status: Downloaded newer image for nginx:latest
27e96cb5d7c336dbbe0f45609c43d4906625ed4f3631ecca83d67584ba6b9dad
```



Web Page:



Verifying running container:

```
C:\Users\Mirella>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
27e96cb5d7c3   nginx     "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8080->80/tcp   kind_agnesi
```

Publishing Ports

- Publishing ports in a Docker creates a bridge between the isolated environment of the container and the outside world.

- By linking a specific on the container to a port on a host machine, they are enabling the external access to the services running inside the container.

Publishing all ports:

With the -P or --publish-all flag, it automatically publishes all exposed ports to ephemeral ports (temporary ports assigned by an operating system for client-side communications. Used during a session to send and receive data). This is quite useful when trying to avoid port conflicts in development or testing environments.

```
:\Users\Mirella>docker run -P nginx
docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
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
0-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
0-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
docker-entrypoint.sh: Configuration complete; ready for start up
025/03/13 09:31:32 [notice] 1#1: using the "epoll" event method
025/03/13 09:31:32 [notice] 1#1: nginx/1.27.4
025/03/13 09:31:32 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
025/03/13 09:31:32 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
025/03/13 09:31:32 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
025/03/13 09:31:32 [notice] 1#1: start worker processes
025/03/13 09:31:32 [notice] 1#1: start worker process 29
```

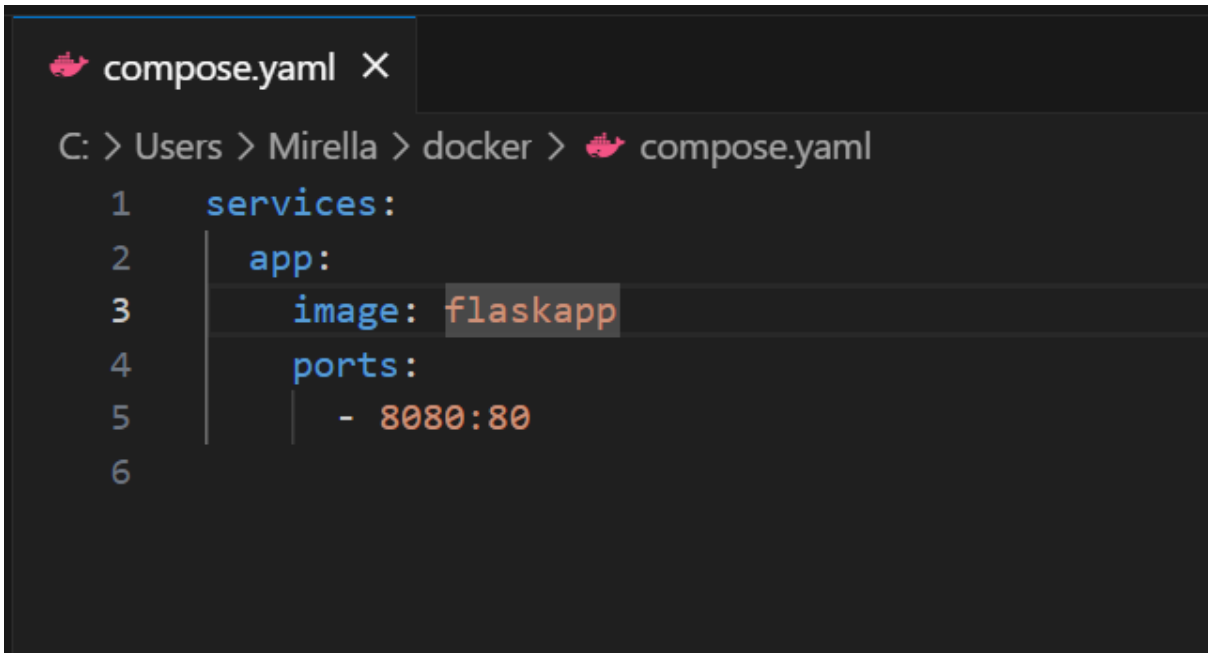
Docker Compose:

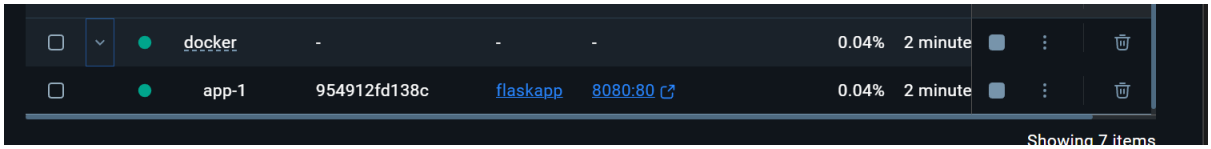
Running one or more images in the container.

It simplifies the configuration by e.g., defining ports environment variables and dependencies in one file.

It ensures the same setup across different environments.

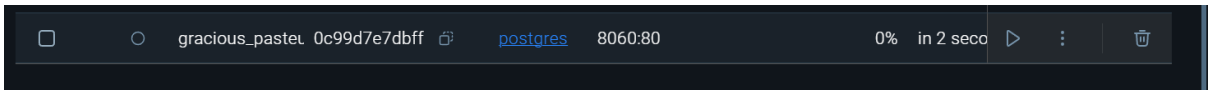
It automatically creates a shared network for services.





Tutorial 2:

```
See 'docker run --help'.

C:\Users\Mirella>docker run -d -p 8060:80 postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
fcccafd45a4d: Download complete
42e76ffa3e07: Download complete
bc13f9b1d80d: Download complete
878a40f56a67: Download complete
dc87fb4dbc03: Download complete
600e770d797e: Download complete
783086ffbe8e: Download complete
420a047e4570: Download complete
55c54708c8e7: Download complete
543c6dea2e39: Download complete
6424ae1ae883: Download complete
553d1749e29f: Download complete
a21a08dbca2c: Download complete
Digest: sha256:81f32a88ec561664634637dd446487efd5f9d90996304b96210078e90e5c8b21
Status: Downloaded newer image for postgres:latest
0c99d7e7dbff6b598058e60d19e1160769cba3a8f2b7b3fb62f11f7095ef4597
```

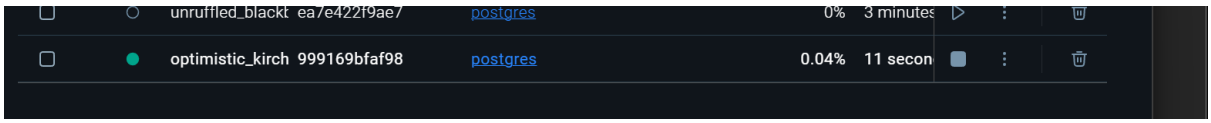


Setting up the environment variable:

```
0ea67e8a53807466adc8bd1b275c401386948e9bf60e7104e77fe70f8f310): Bind for 0.0.0.0:8080 failed: por

C:\Users\Mirella>docker run -e foo=bar postgres env
HOSTNAME=ea7e422f9ae7
PWD=/
HOME=/root
LANG=en_US.utf8
GOSU_VERSION=1.17
foo=bar
PG_MAJOR=17
PG_VERSION=17.4-1.pgdg120+2
SHLVL=0
PGDATA=/var/lib/postgresql/data
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/postgresql/17/bin
```

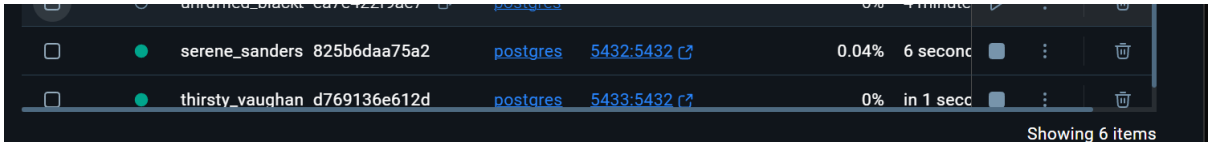
Restricting how many CPUs will the container use:

```
C:\Users\Mirella>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 disabled.

fixing permissions on existing directory /var/lib/postgresql/data ... 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
```

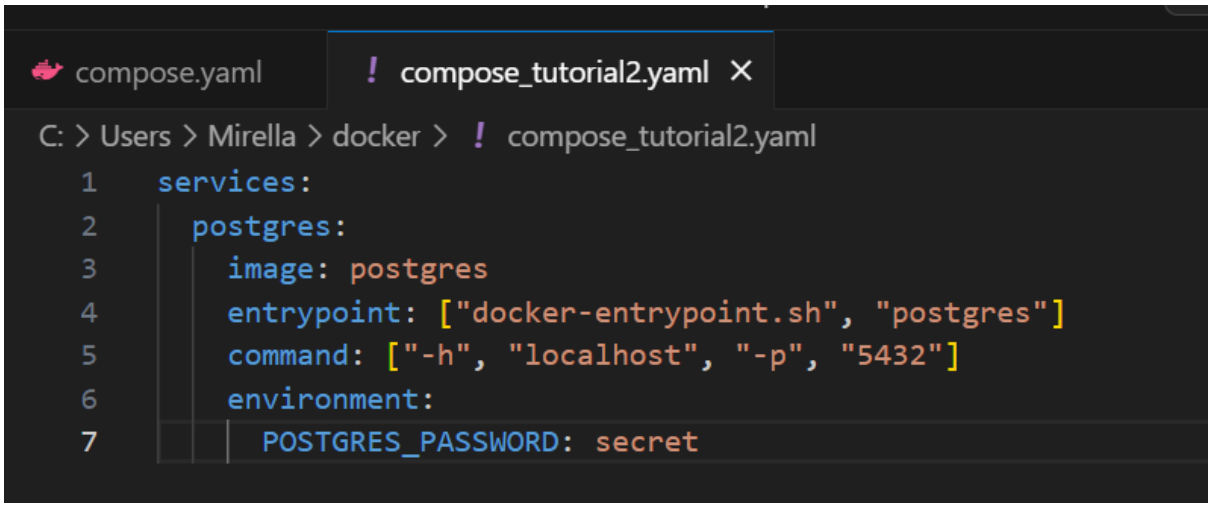


Running multiple containers:

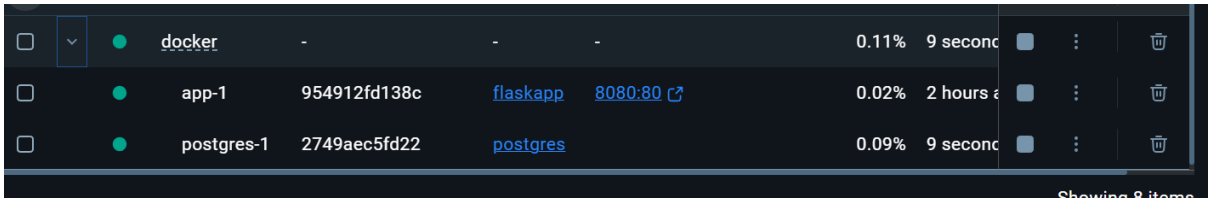
```
C:\Users\Mirella>docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgres
825b6daa75a27b30a7b1b55f8fc11cfc5e915467de18f73b3cd41f80883cebbc

C:\Users\Mirella>docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgres
d769136e612d73cf6839a17b7f113de61e2412dc276d2cc2125e2959421e54df
```



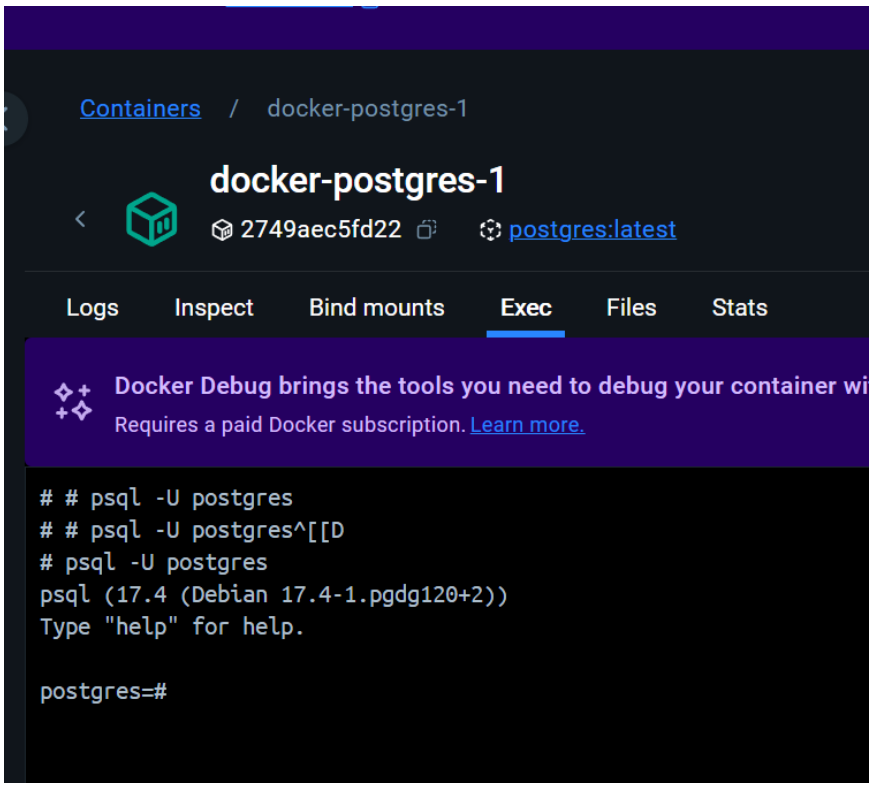
Overwriting default CMD and entry point in the Docker compose:




```
C:\Users\Mirella\docker>docker compose -f compose_tutorial2.yaml up -d
time="2025-03-13T13:15:42Z" level=warning msg="Found orphan containers ([docker-app-1]) for this project. If you removed
 or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 1/1
 ✔Container docker-postgres-1  Started                                                                          4.7s
```



Verify the authentication:



Tutorial 3:

Creating a Docker Volume:

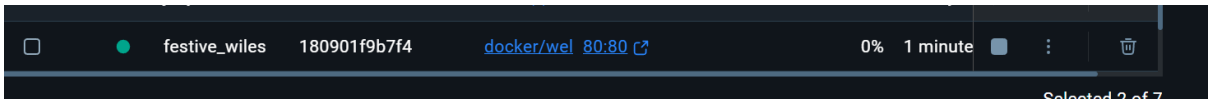
By default, files inside a container are deleted when the container is removed. A volume ensures that your data remains untouched even after the container is deleted.

Start a Container with a Volume:

Run a container from the docker/welcome-to-docker image.

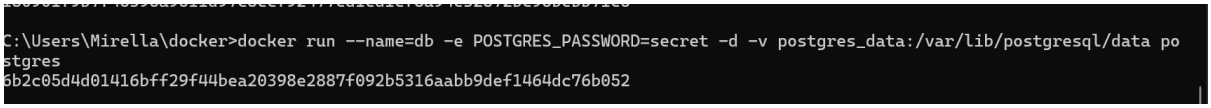
-v log-data:/logs attaches the volume log-data to the /logs directory in the container. Any logs stored in /logs inside the container will persist in the log-data volume. If the container is removed, the logs remain.

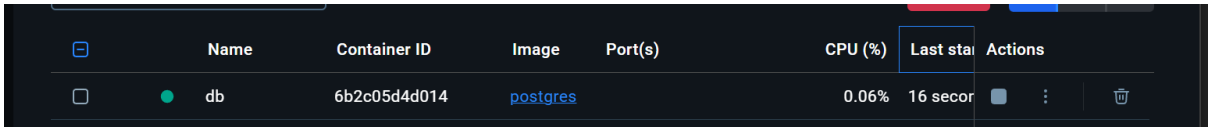
```
C:\Users\Mirella\docker>docker run -d -p 80:80 -v log-data:/logs docker/welcome-to-docker
180901f9b7f45396a9611d97c8ccf92477cd1cd1cf8a94e32872bc95bebb71c5
```



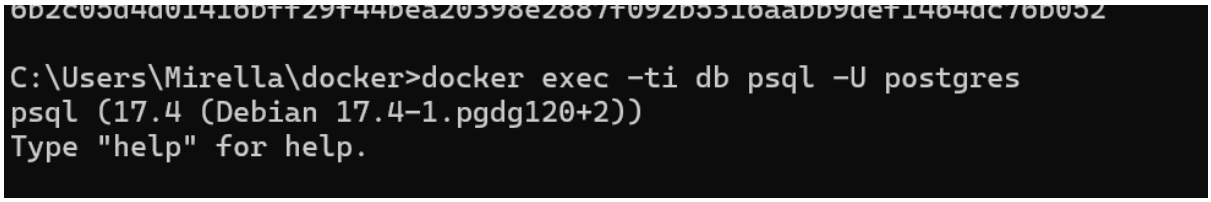
Start a PostgreSQL container with a volume:

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

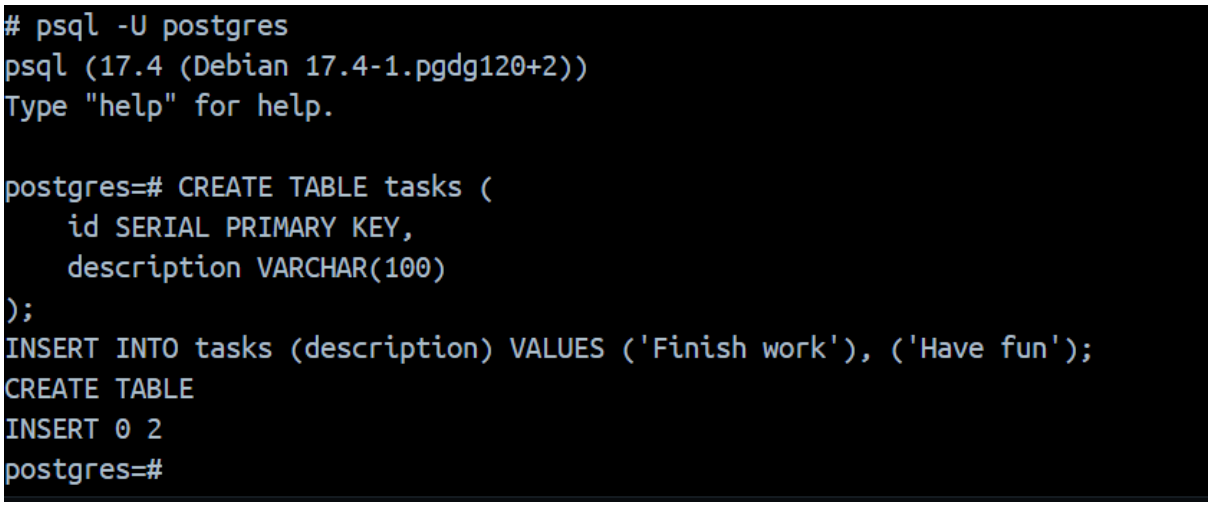




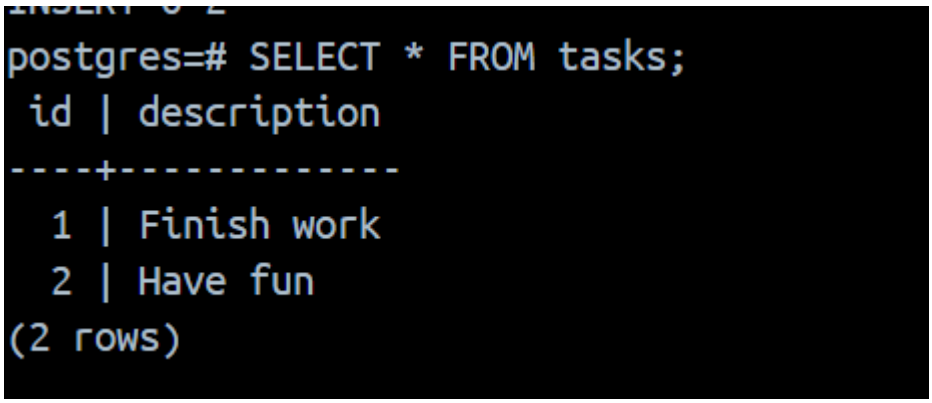
Connect to the PostgreSQL:



Creating a Table and Insert Data:



Verification the data is in the table:

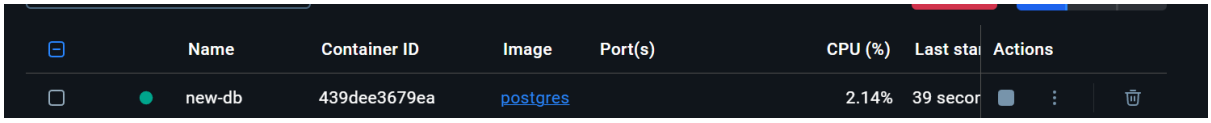


Stop and remove container:

The data is in the postgres_data volume.

```
C:\Users\Mirella\docker>docker stop db
db

C:\Users\Mirella\docker>docker rm db
db

C:\Users\Mirella\docker>
```

Start a new container and attaching the same volume:




```
C:\Users\Mirella\docker>docker run --name=new-db -d -v postgres_data:/var/lib/postgresql/data postgres
439dee3679eab6f985ae5d210b4c6149c06815b18d4e4cbce39d5cc14d41f4b2
```

Verification that the database has the records:

```
C:\Users\Mirella\docker>docker exec -ti new-db psql -U postgres -c "SELECT * FROM tasks"
 id | description
----+-------------
  1 | Finish work
  2 | Have fun
(2 rows)
```

Commands to remove volumes:

1. Remove postgres_data volume:

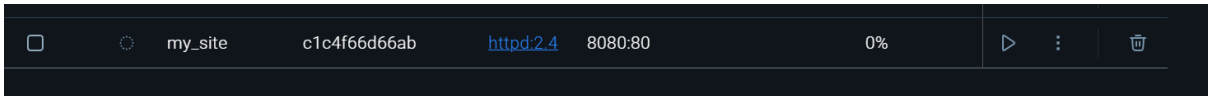
   docker volume rm postgres_data

2. Remove all unused volumes:
   docker volume prune

Task 4:

Starting a container running an HTTP server:

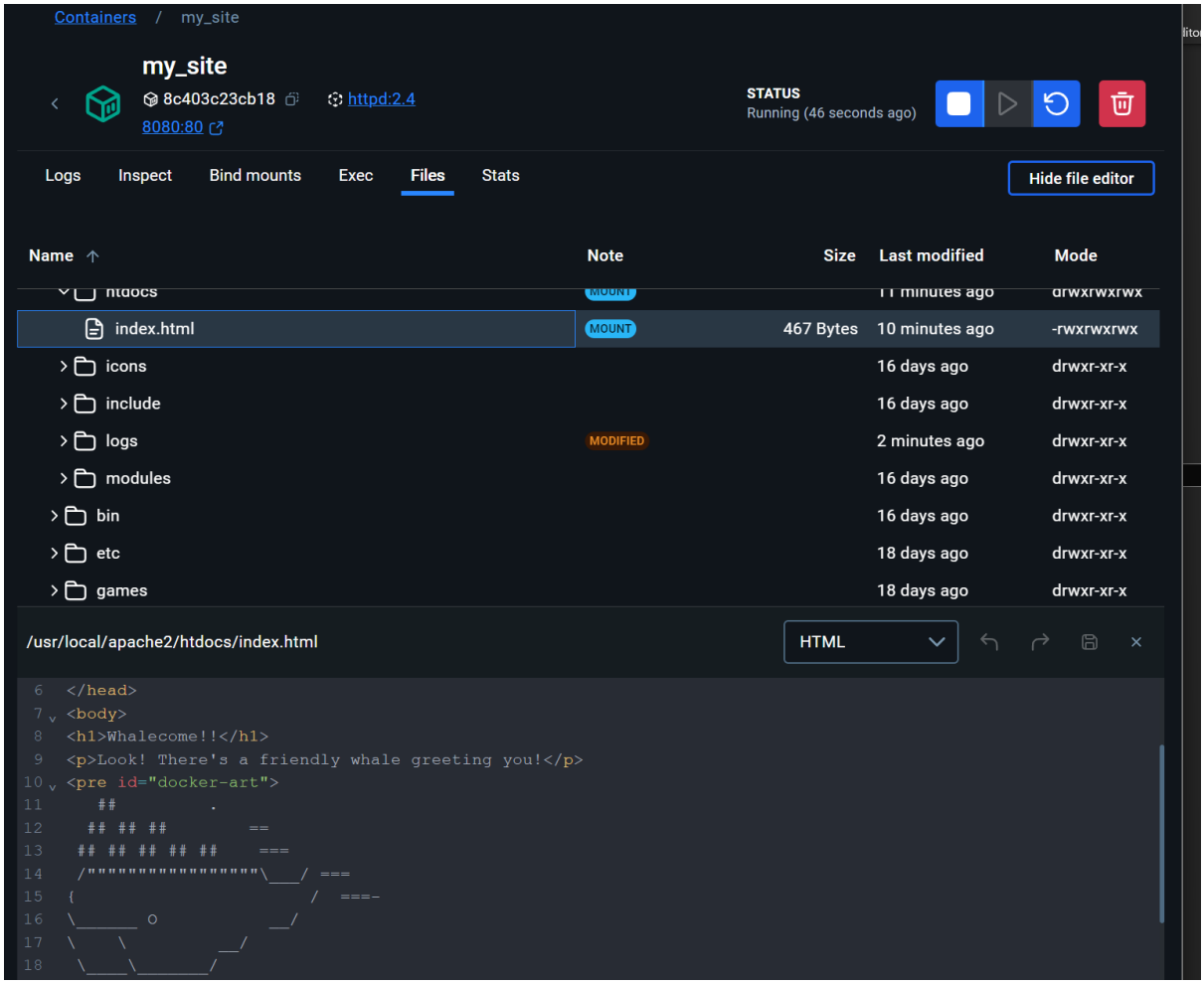
```
C:\Users\Mirella\docker>docker run -d -p 8080:80 --name my_site httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
38fd0d422c41: Download complete
d2f10b557009: Download complete
fdebd6c6e1b2: Download complete
4f4fb700ef54: Download complete
470035b3d48f: Download complete
Digest: sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d2671c8ab6e
Status: Downloaded newer image for httpd:2.4
c1c4f66d66ab772a23a7675f2ff7b2b5d2109519441a40745dd31b6b0c3677c7
docker: Error response from daemon: driver failed programming external connectivity on endpoint my_site (6b0511e3b5d256c
39e77d0aab714d76a50e0e2c7165561e188d93bd1d766c3cd): Bind for 0.0.0.0:8080 failed: port is already allocated.
```



Creating a container

```
C:\Users\Mirella\docker\public_html>docker run -d --name my_site -p 8080:80 -v C:/Users/Mirella/docker/public_html:/usr/
local/apache2/htdocs/ httpd:2.4
8c403c23cb189b7b2a4e8e7d946c7754ad4adc613d30eafe236262e49b0c32ad
```

Mapping path from a volume to a physical disc.



Tutorial 5:

Cloning a project

```
C:\Users\Mirella\docker>git clone https://github.com/dockersamples/nginx-node-redis
Cloning into 'nginx-node-redis'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (73/73), done.
remote: Total 78 (delta 25), reused 4 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (78/78), 75.06 KiB | 4.17 MiB/s, done.
Resolving deltas: 100% (25/25), done.
```

Build the image:

```
C:\Users\Mirella\docker\nginx-node-redis>docker build -t nginx .
[+] Building 0.3s (1/1) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                          0.1s
 => => transferring dockerfile: 2B                                                            0.0s
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/dcglhpz5z785fsuyxdhgor8wl
```

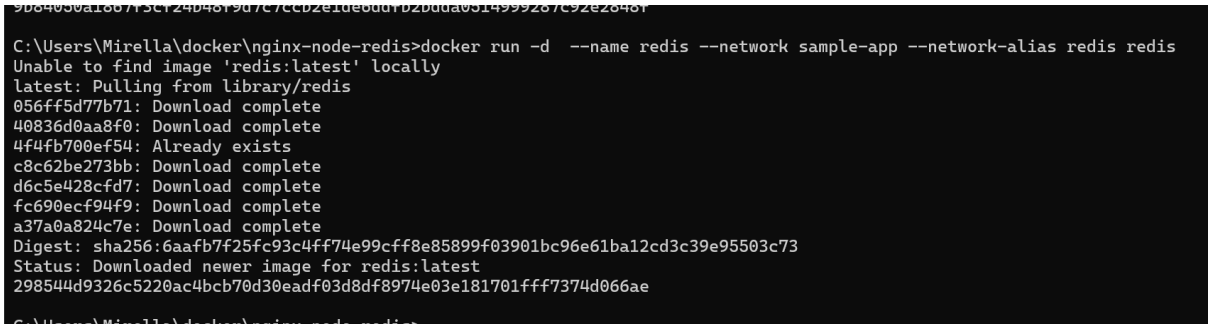
Build the first web page:

```
C:\Users\Mirella\docker\nginx-node-redis>docker build -t web .
[+] Building 0.1s (1/1) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 2B                                                            0.0s
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/1comp41eplx35dgftdz6cabke
```

Creating a network:

```
C:\Users\Mirella\docker\nginx-node-redis>docker network create sample-app
9b84050a1867f3cf24b48f9d7c7ccb2e1de6ddfb2bdda0514999287c92e2848f
```

Starting the redis container:



Starting the first web container:

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

Starting a second web container:

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

Starting NGIX Container: