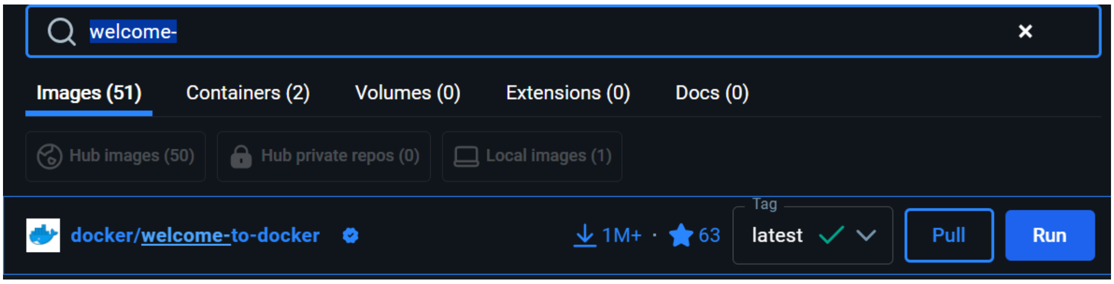
Search for containers



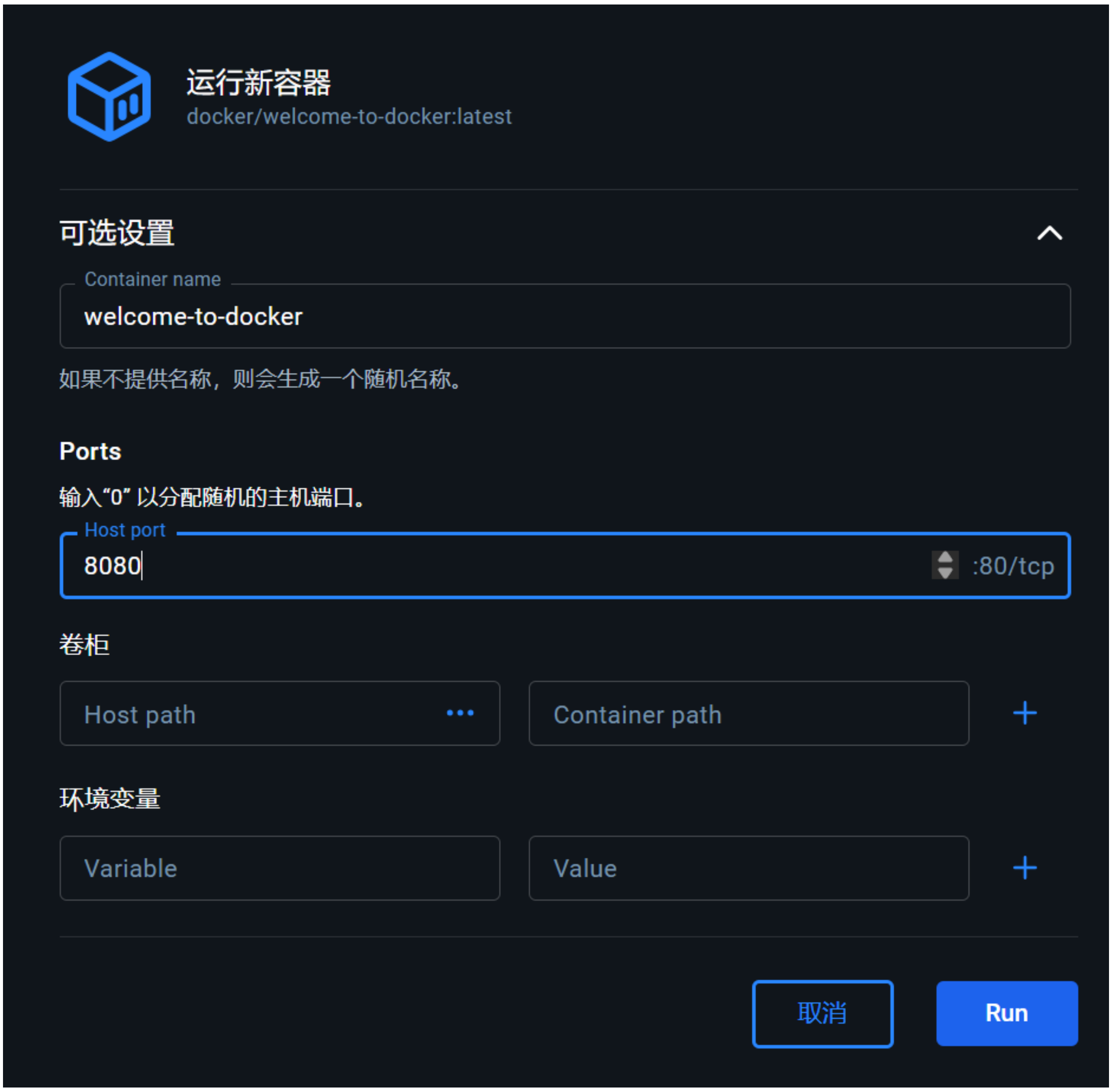
welcome-

Images (51) Containers (2) Volumes (0) Extensions (0) Docs (0)

Hub images (50) Hub private repos (0) Local images (1)

docker/welcome-to-docker 1M+ · ★ 63 Tag latest ✓ Pull Run

Deploy containers



运行新容器
docker/welcome-to-docker:latest

可选设置

Container name
welcome-to-docker

如果不提供名称，则会生成一个随机名称。

Ports

输入“0” 以分配随机的主机端口。

Host port
8080 :80/tcp

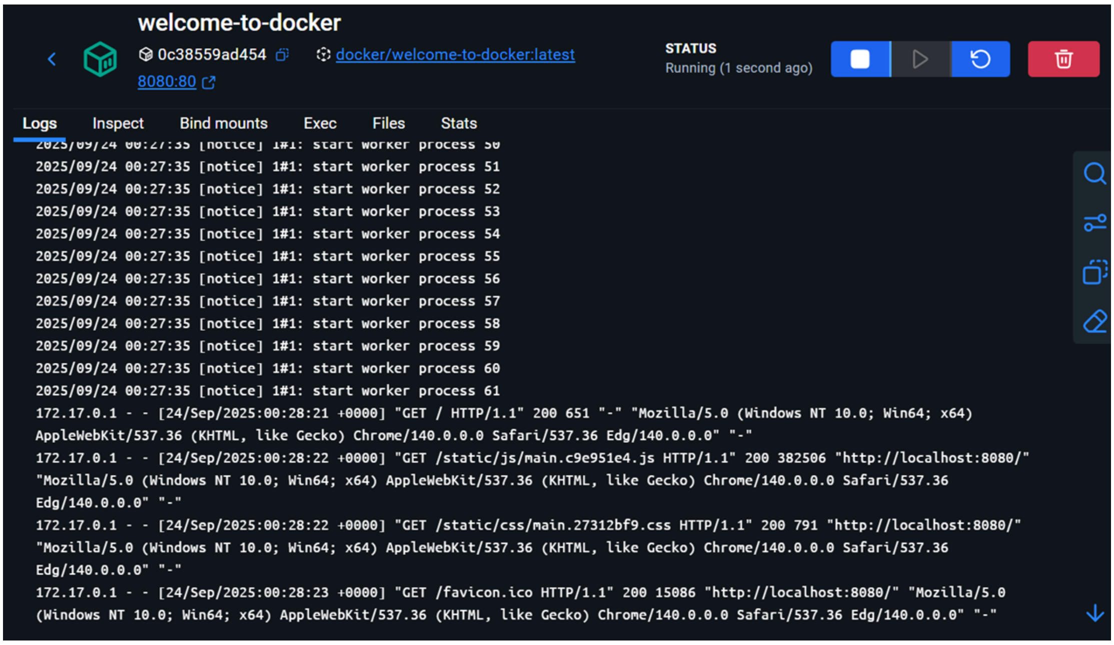
卷柜

Host path ··· Container path +

环境变量

Variable Value +

取消 Run



welcome-to-docker

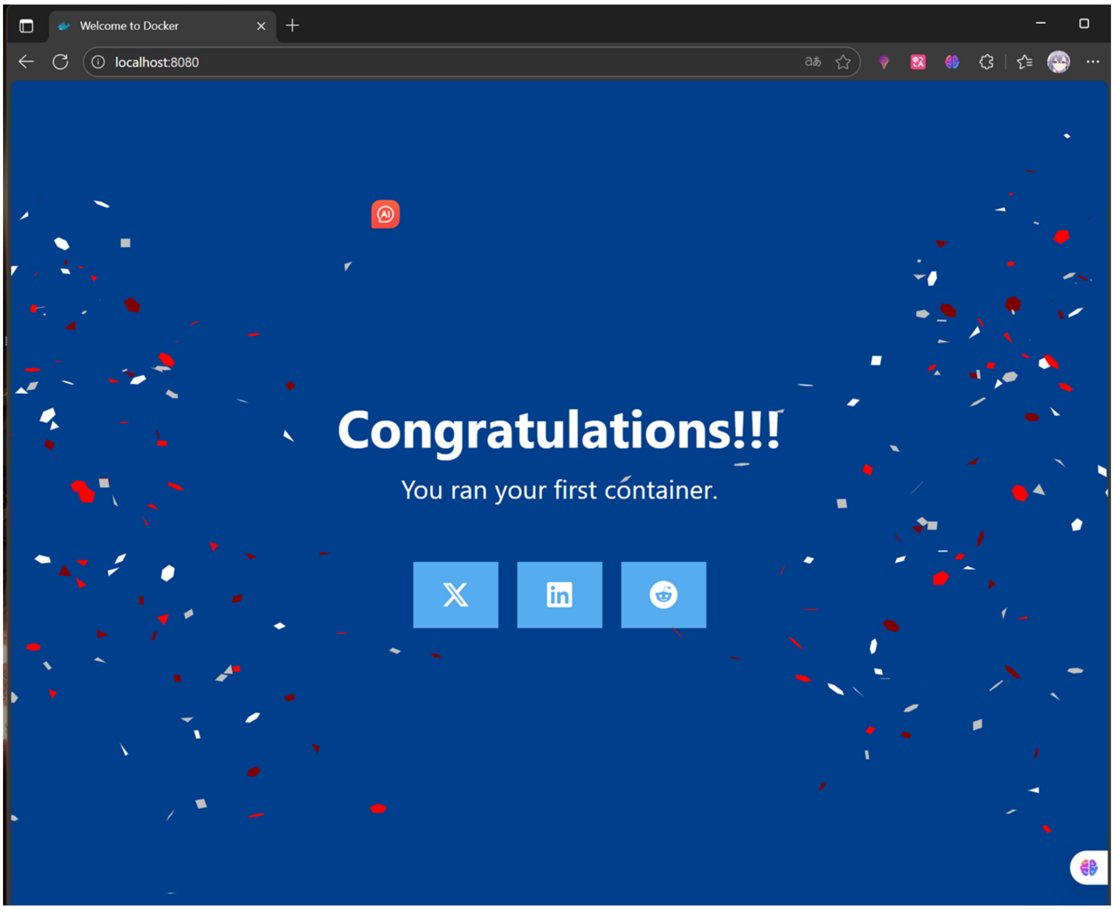
0c38559ad454   docker/welcome-to-docker:latest

8080:80

STATUS
Running (1 second ago)

```
2025/09/24 00:27:35 [notice] 1#1: start worker process 50
2025/09/24 00:27:35 [notice] 1#1: start worker process 51
2025/09/24 00:27:35 [notice] 1#1: start worker process 52
2025/09/24 00:27:35 [notice] 1#1: start worker process 53
2025/09/24 00:27:35 [notice] 1#1: start worker process 54
2025/09/24 00:27:35 [notice] 1#1: start worker process 55
2025/09/24 00:27:35 [notice] 1#1: start worker process 56
2025/09/24 00:27:35 [notice] 1#1: start worker process 57
2025/09/24 00:27:35 [notice] 1#1: start worker process 58
2025/09/24 00:27:35 [notice] 1#1: start worker process 59
2025/09/24 00:27:35 [notice] 1#1: start worker process 60
2025/09/24 00:27:35 [notice] 1#1: start worker process 61
172.17.0.1 - - [24/Sep/2025:00:28:21 +0000] "GET / HTTP/1.1" 200 651 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0" "-"
172.17.0.1 - - [24/Sep/2025:00:28:22 +0000] "GET /static/js/main.c9e951e4.js HTTP/1.1" 200 382506 "http://localhost:8080/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Edg/140.0.0.0" "-"
172.17.0.1 - - [24/Sep/2025:00:28:22 +0000] "GET /static/css/main.27312bf9.css HTTP/1.1" 200 791 "http://localhost:8080/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Edg/140.0.0.0" "-"
172.17.0.1 - - [24/Sep/2025:00:28:23 +0000] "GET /favicon.ico HTTP/1.1" 200 15086 "http://localhost:8080/" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0" "-"
```

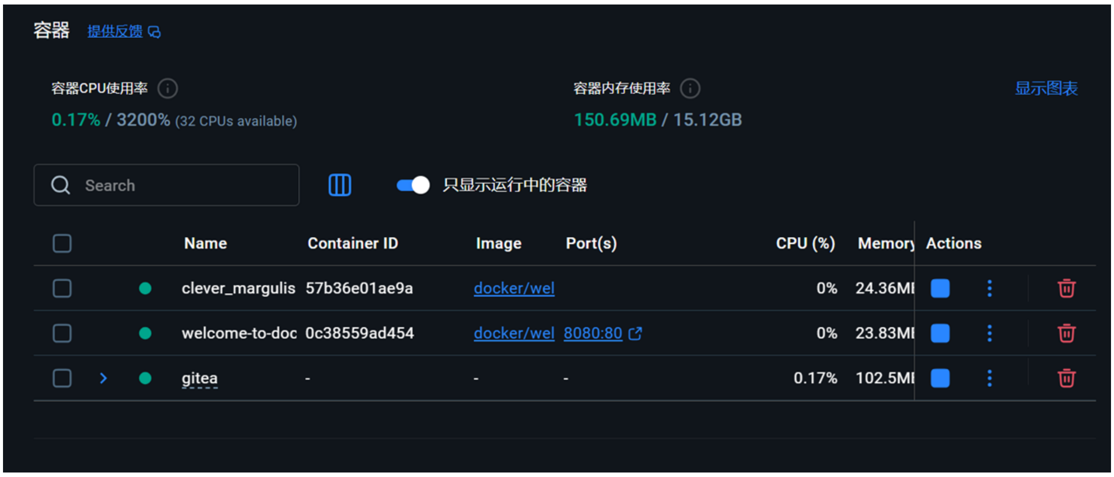
Access port



Congratulations!!!

You ran your first container.

Container interface



容器 提供反馈

容器CPU使用率
0.17% / 3200% (32 CPUs available)

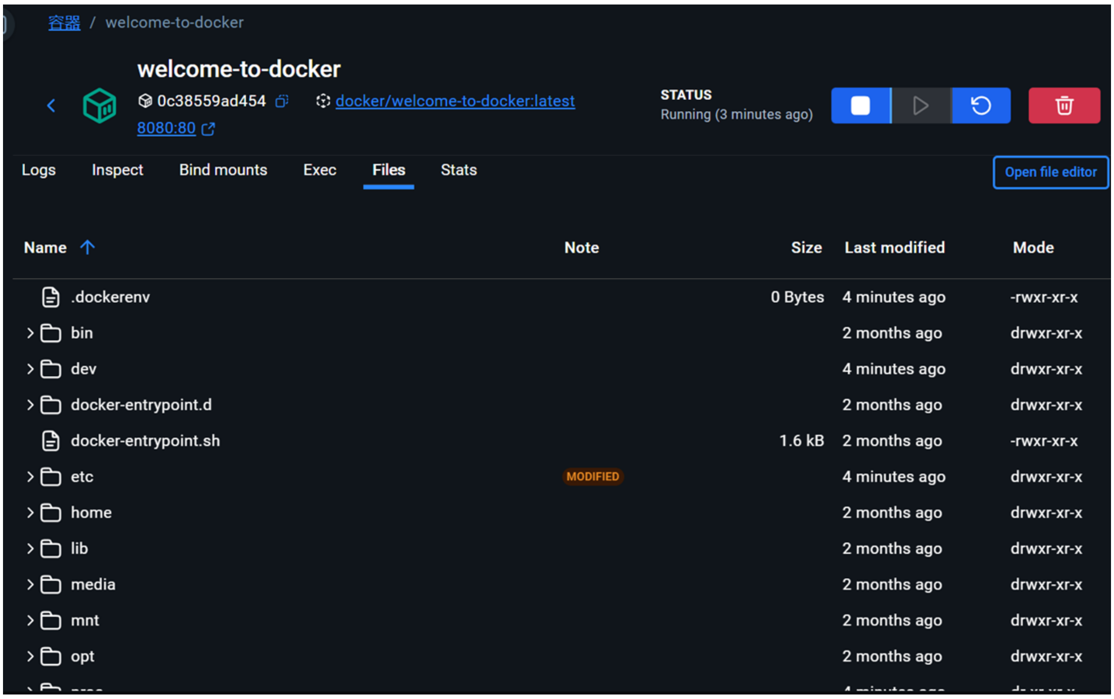
容器内存使用率
150.69MB / 15.12GB

显示图表

Search

只显示运行中的容器

□			Name	Container ID	Image	Port(s)	CPU (%)	Memory	Actions		
□		●	clever_margulis	57b36e01ae9a	docker/wel		0%	24.36MI	■	⋮	🗑
□		●	welcome-to-doc	0c38559ad454	docker/wel	8080:80	0%	23.83MI	■	⋮	🗑
□	›	●	gitea	-	-	-	0.17%	102.5MI	■	⋮	🗑



容器 / welcome-to-docker

welcome-to-docker

0c38559ad454 docker/welcome-to-docker:latest

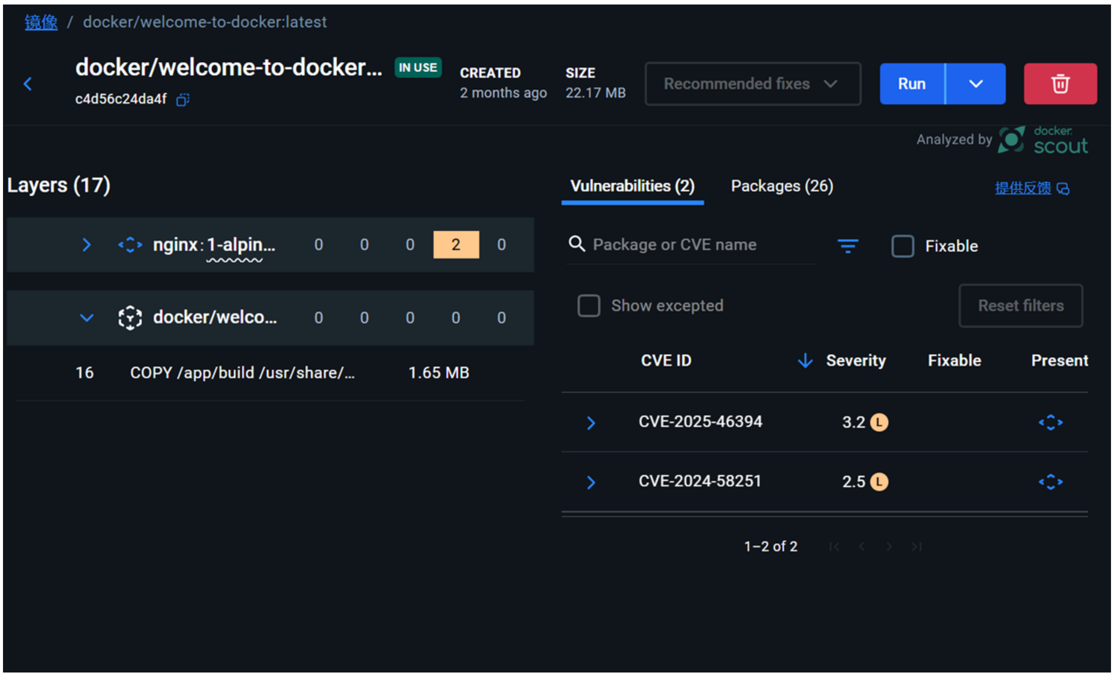
8080:80

STATUS
Running (3 minutes ago)

Logs Inspect Bind mounts Exec Files Stats

Open file editor

Name ↑	Note	Size	Last modified	Mode
.dockerenv		0 Bytes	4 minutes ago	-rwxr-xr-x
› bin			2 months ago	drwxr-xr-x
› dev			4 minutes ago	drwxr-xr-x
› docker-entrypoint.d			2 months ago	drwxr-xr-x
docker-entrypoint.sh		1.6 kB	2 months ago	-rwxr-xr-x
› etc	MODIFIED		4 minutes ago	drwxr-xr-x
› home			2 months ago	drwxr-xr-x
› lib			2 months ago	drwxr-xr-x
› media			2 months ago	drwxr-xr-x
› mnt			2 months ago	drwxr-xr-x
› opt			2 months ago	drwxr-xr-x



镜像 / docker/welcome-to-docker:latest

docker/welcome-to-docker... IN USE
c4d56c24da4f

CREATED 2 months ago

SIZE 22.17 MB

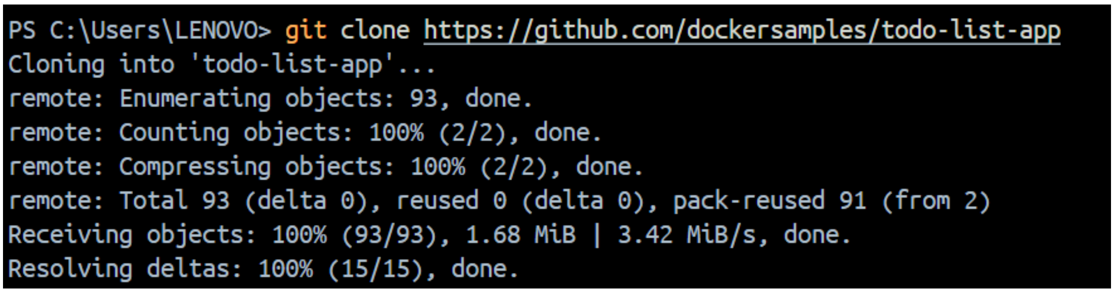
Layers (17)

>	nginx:1-alpin...	0	0	0	2	0
⌄	docker/welco...	0	0	0	0	0
16	COPY /app/build /usr/share/...	1.65 MB				

CVE ID	Severity	Fixable	Present
CVE-2025-46394	3.2 L		
CVE-2024-58251	2.5 L		

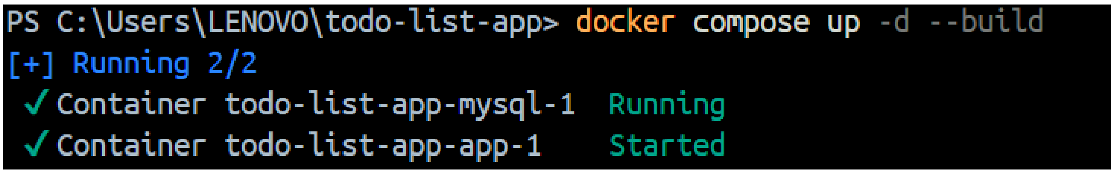
1–2 of 2

clone

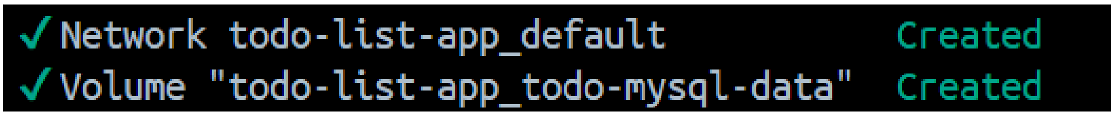


```
PS C:\Users\LENOVO> git clone https://github.com/dockersamples/todo-list-app
Cloning into 'todo-list-app'...
remote: Enumerating objects: 93, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 93 (delta 0), reused 0 (delta 0), pack-reused 91 (from 2)
Receiving objects: 100% (93/93), 1.68 MiB | 3.42 MiB/s, done.
Resolving deltas: 100% (15/15), done.
```

Configuration

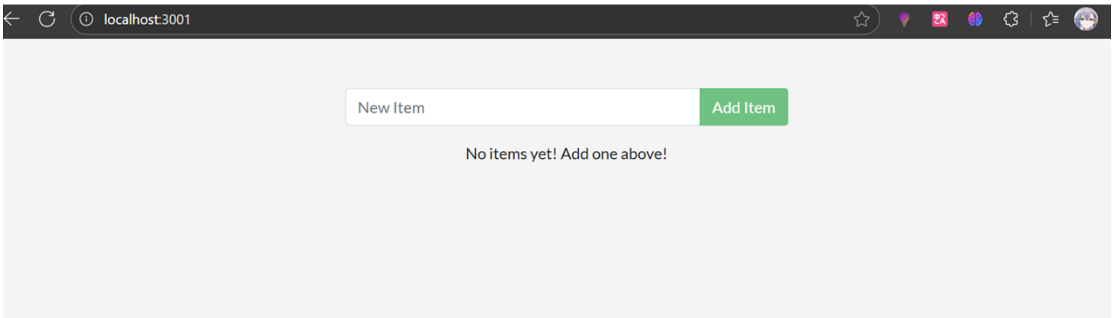


```
PS C:\Users\LENOVO\todo-list-app> docker compose up -d --build
[+] Running 2/2
 ✓ Container todo-list-app-mysql-1  Running
 ✓ Container todo-list-app-app-1    Started
```



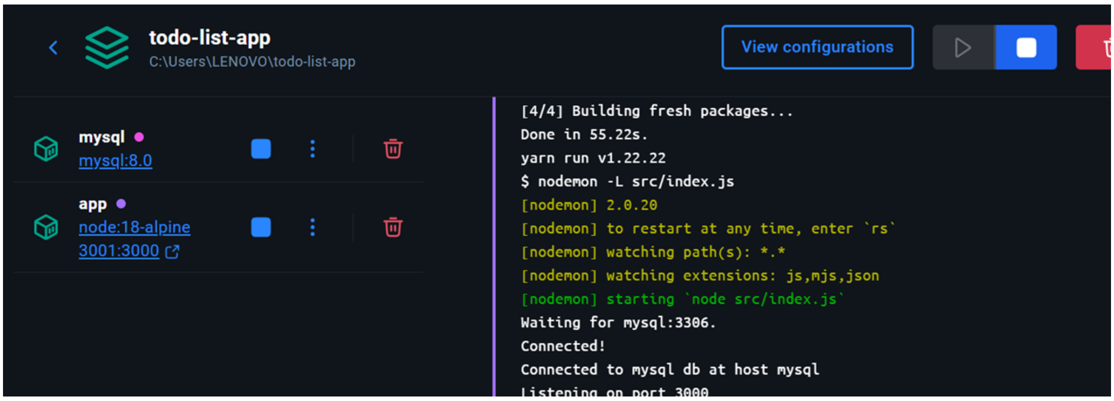
```
 ✓ Network todo-list-app_default          Created
 ✓ Volume "todo-list-app_todo-mysql-data"  Created
```

Application operation interface



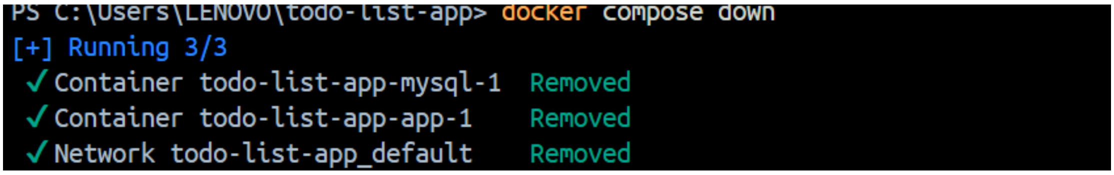
No items yet! Add one above!

View the graphical interface



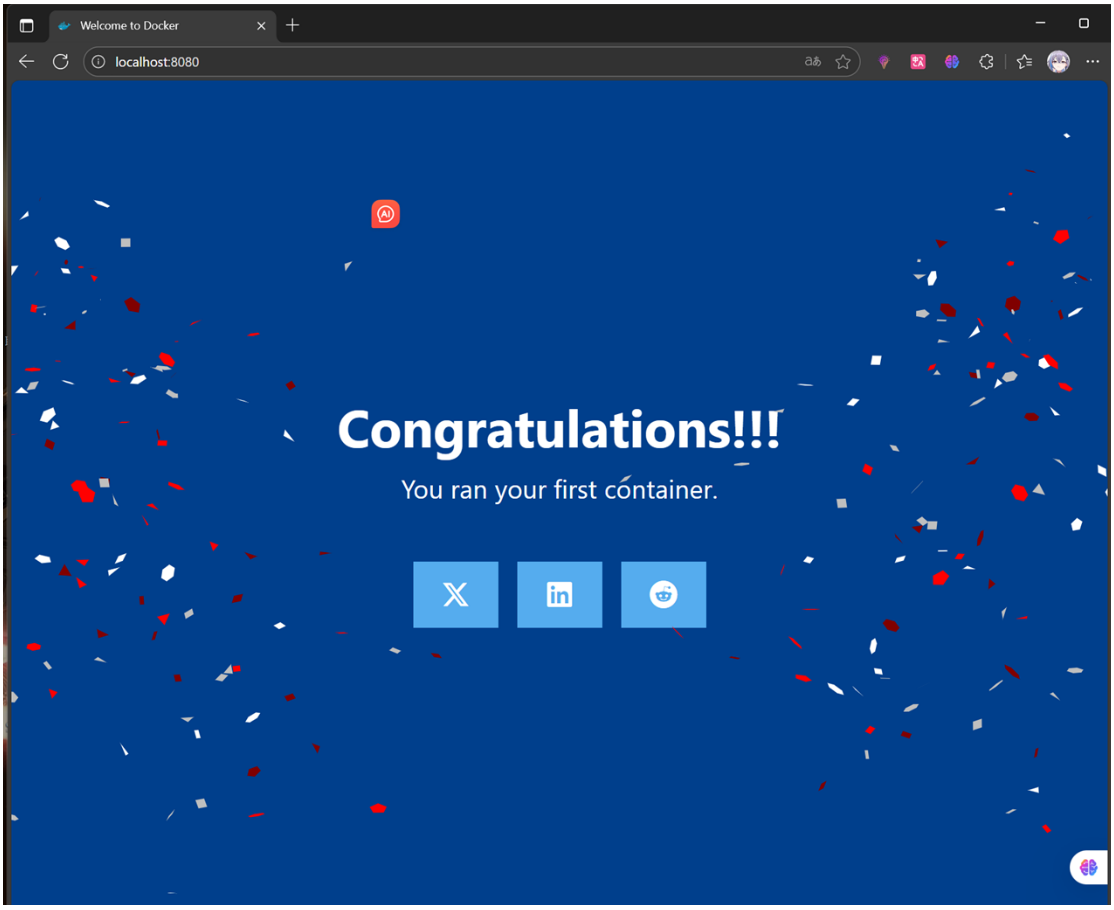
```
[4/4] Building fresh packages...
Done in 55.22s.
yarn run v1.22.22
$ nodemon -L src/index.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Waiting for mysql:3306.
Connected!
Connected to mysql db at host mysql
Listening on port 3000
```

Uninstall



```
PS C:\Users\LENOVO\todo-list-app> docker compose down
[+] Running 3/3
 ✓ Container todo-list-app-mysql-1  Removed
 ✓ Container todo-list-app-app-1    Removed
 ✓ Network todo-list-app_default    Removed
```

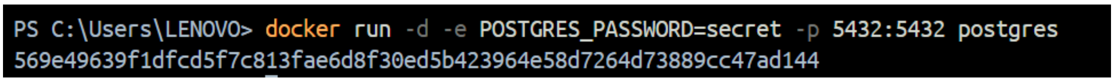
Publishing port



Congratulations!!!

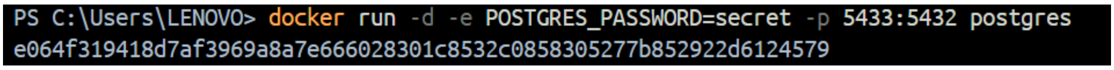
You ran your first container.

Start a container using the Postgres image with the following command:



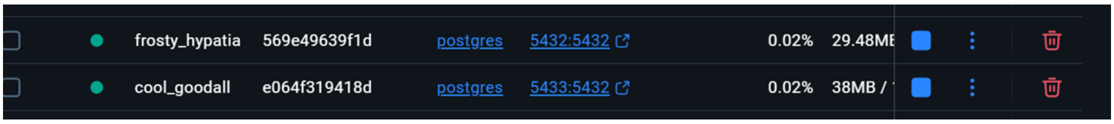
```
PS C:\Users\LENOVO> docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgres
569e49639f1dfcd5f7c813fae6d8f30ed5b423964e58d7264d73889cc47ad144
```

Start a second Postgres container mapped to a different port.



```
PS C:\Users\LENOVO> docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgres
e064f319418d7af3969a8a7e666028301c8532c0858305277b852922d6124579
```

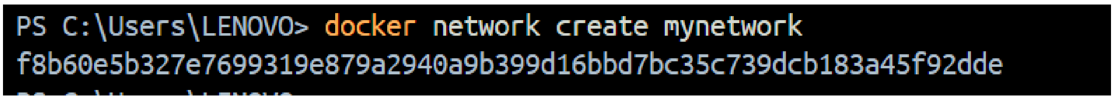
Verify that both containers are running by going to the **Containers** view in the Docker Desktop Dashboard.



□	●	frosty_hypatia	569e49639f1d	postgres	5432:5432	0.02%	29.48MB
□	●	cool_goodall	e064f319418d	postgres	5433:5432	0.02%	38MB /

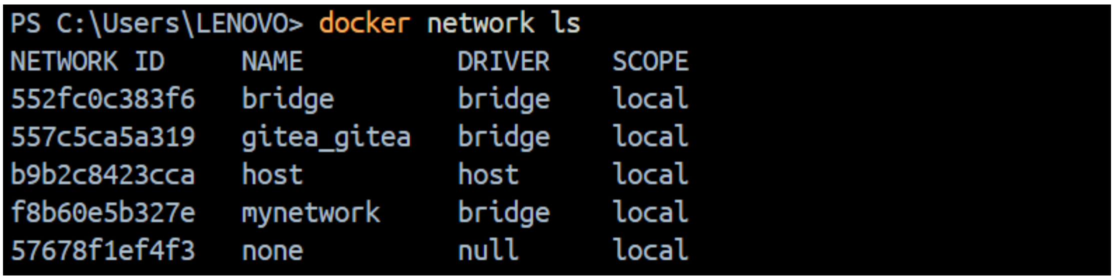
Run Postgres container in a controlled network

Create a new custom network by using the following command:



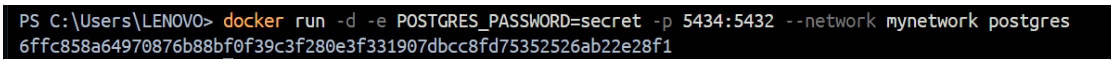
```
PS C:\Users\LENOVO> docker network create mynetwork
f8b60e5b327e7699319e879a2940a9b399d16bbd7bc35c739dcb183a45f92dde
```

Verify the network by running the following command:



```
PS C:\Users\LENOVO> docker network ls
NETWORK ID     NAME          DRIVER    SCOPE
552fc0c383f6   bridge        bridge    local
557c5ca5a319   gitea_gitea   bridge    local
b9b2c8423cca   host          host      local
f8b60e5b327e   mynetwork     bridge    local
57678f1ef4f3   none          null      local
```

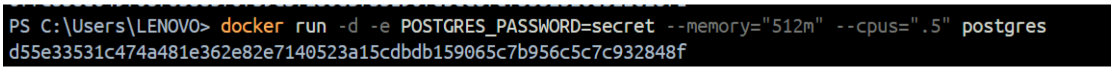
Connect Postgres to the custom network by using the following command



```
PS C:\Users\LENOVO> docker run -d -e POSTGRES_PASSWORD=secret -p 5434:5432 --network mynetwork postgres
6ffc858a64970876b88bf0f39c3f280e3f331907dbcc8fd75352526ab22e28f1
```

This will start Postgres container in the background, mapped to the host port 5434 and attached to the mynetwork network. You passed the --network parameter to override the container default by connecting the container to custom Docker network for better isolation and communication with other containers.

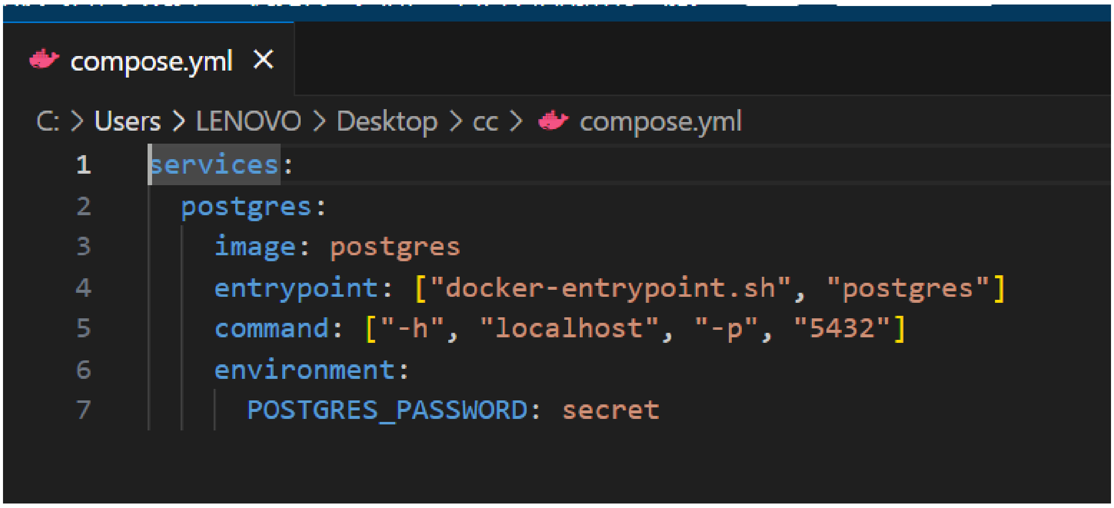
Manage the resources



```
PS C:\Users\LENOVO> docker run -d -e POSTGRES_PASSWORD=secret --memory="512m" --cpus=".5" postgres
d55e33531c474a481e362e82e7140523a15cdbdb159065c7b956c5c7c932848f
```

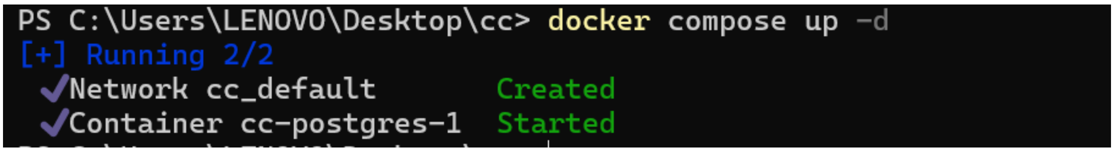
Override the default CMD and ENTRYPOINT in Docker Compose

Create a compose.yml file with the following content:



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

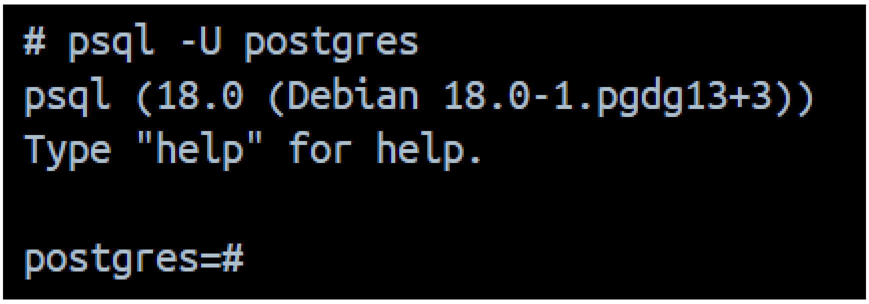
Bring up the service by running the following command:



```
PS C:\Users\LENOVO\Desktop\cc> docker compose up -d
[+] Running 2/2
 ✓Network cc_default         Created
 ✓Container cc-postgres-1    Started
```

Verify the authentication with Docker Desktop Dashboard.

type the following command to connect to the Postgres database:

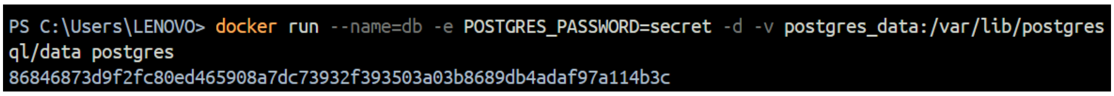


```
# psql -U postgres
psql (18.0 (Debian 18.0-1.pgdg13+3))
Type "help" for help.

postgres=#
```

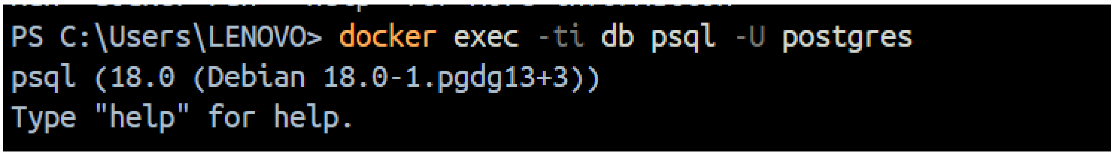
Use volumes

Start a container using the Postgres image with the following command:



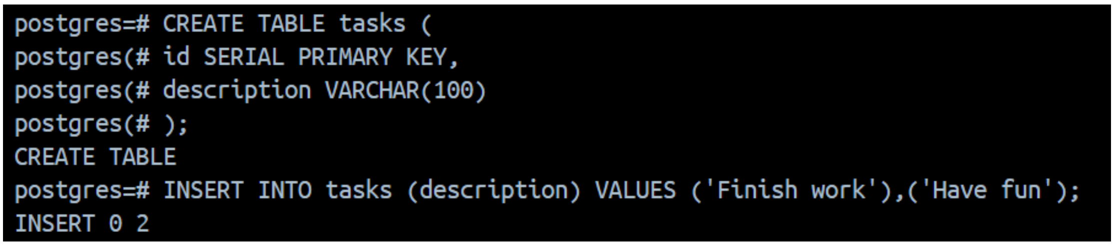
```
PS C:\Users\LENOVO> docker run --name=db -e POSTGRES_PASSWORD=secret -d -v postgres_data:/var/lib/postgres
ql/data postgres
86846873d9f2fc80ed465908a7dc73932f393503a03b8689db4adaf97a114b3c
```

Connect to the database by using the following command:



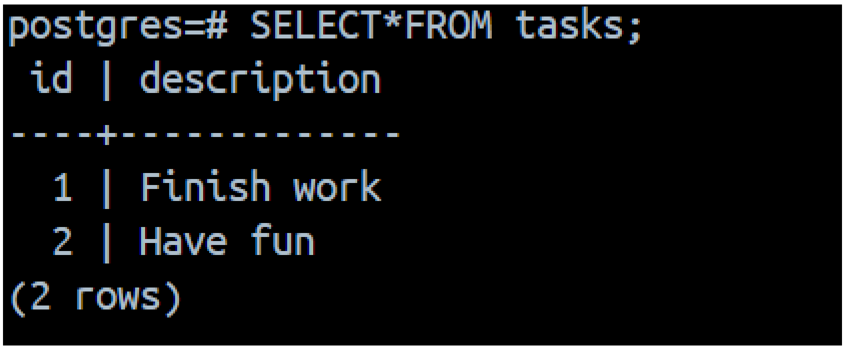
```
PS C:\Users\LENOVO> docker exec -ti db psql -U postgres
psql (18.0 (Debian 18.0-1.pgdg13+3))
Type "help" for help.
```

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



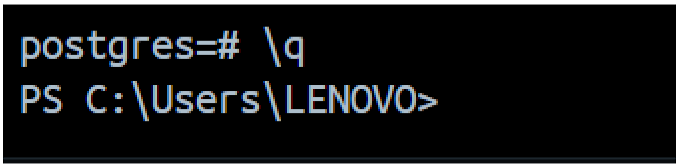
```
postgres=# CREATE TABLE tasks (
postgres(# id SERIAL PRIMARY KEY,
postgres(# description VARCHAR(100)
postgres(# );
CREATE TABLE
postgres=# INSERT INTO tasks (description) VALUES ('Finish work'),('Have fun');
INSERT 0 2
```

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



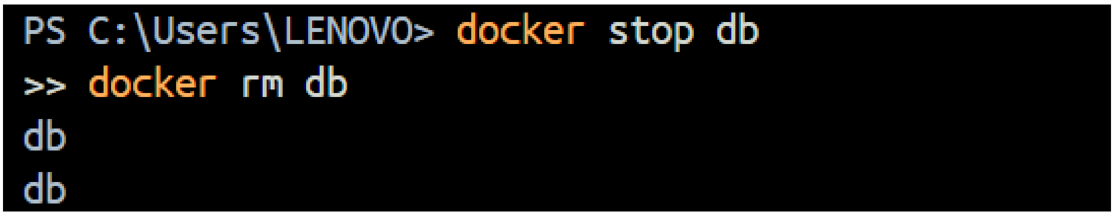
```
postgres=# SELECT*FROM tasks;
 id | description
----+-------------
  1 | Finish work
  2 | Have fun
(2 rows)
```

Exit out of the PostgreSQL shell by running the following command:



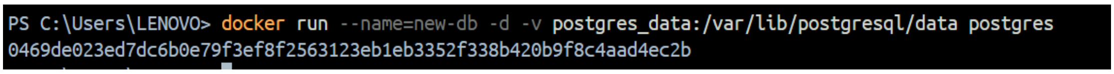
```
postgres=# \q
PS C:\Users\LENOVO>
```

Stop and remove the database container. Remember that, even though the container has been deleted, the data is persisted in the postgres_data volume.



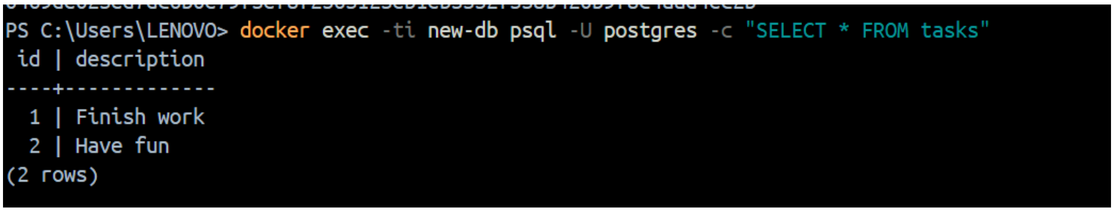
```
PS C:\Users\LENOVO> docker stop db
>> docker rm db
db
db
```

Start a new container by running the following command, attaching the same volume with the persisted data:



```
PS C:\Users\LENOVO> docker run --name=new-db -d -v postgres_data:/var/lib/postgresql/data postgres
0469de023ed7dc6b0e79f3ef8f2563123eb1eb3352f338b420b9f8c4aad4ec2b
```

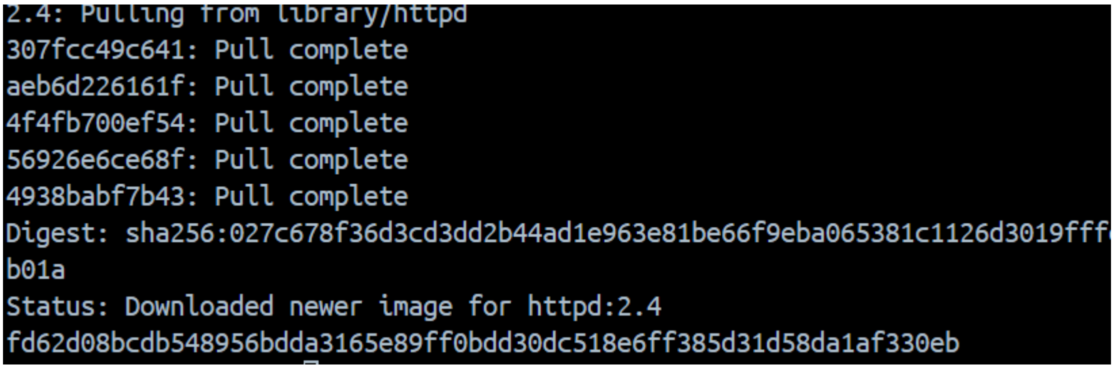
Verify the database still has the records by running the following command:



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

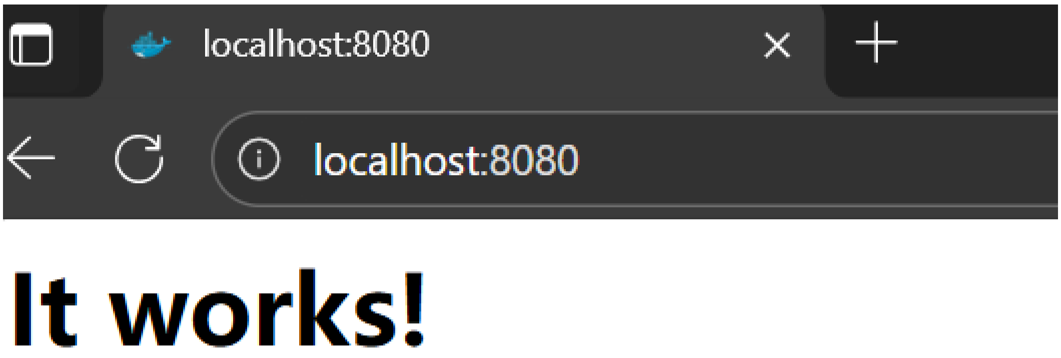
Sharing local files with containers

Start a container using the httpd image with the following command:



```
2.4: Pulling from library/httpd
307fcc49c641: Pull complete
aeb6d226161f: Pull complete
4f4fb700ef54: Pull complete
56926e6ce68f: Pull complete
4938babf7b43: Pull complete
Digest: sha256:027c678f36d3cd3dd2b44ad1e963e81be66f9eba065381c1126d3019fff
b01a
Status: Downloaded newer image for httpd:2.4
fd62d08bcdb548956bdda3165e89ff0bdd30dc518e6ff385d31d58da1af330eb
```

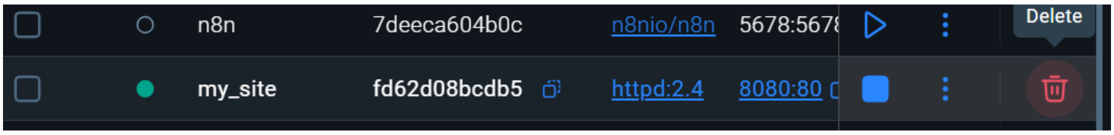
Open the browser and access http://localhost:8080



It works!

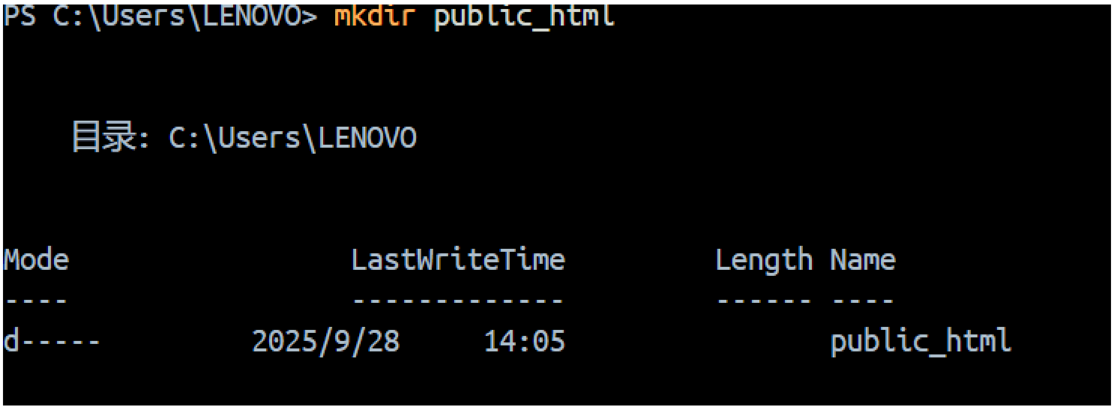
Use a bind mount

Delete the existing container by using the Docker Desktop Dashboard:



□	○	n8n	7deeca604b0c	n8nio/n8n	5678:567
□	●	my_site	fd62d08bcdb5	httpd:2.4	8080:80

Create a new directory called public_html on your host system.

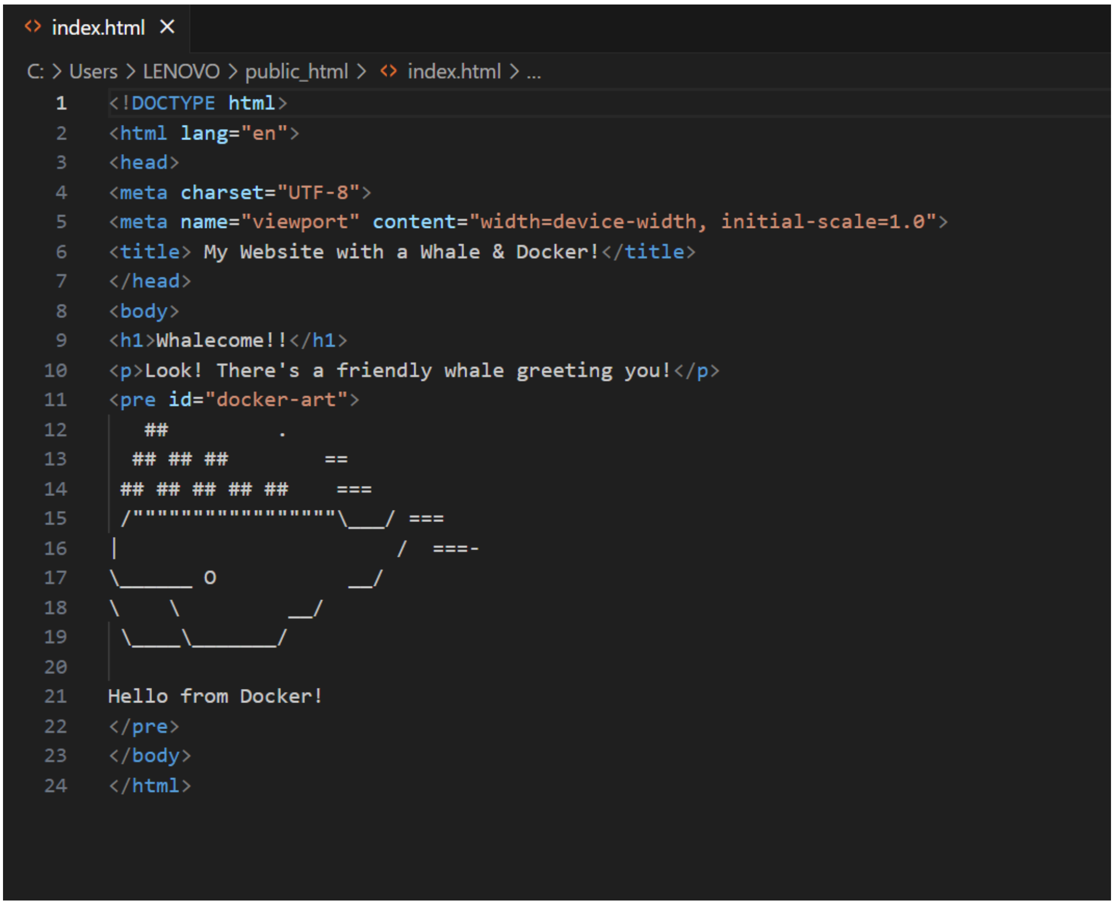


```
PS C:\Users\LENOVO> mkdir public_html


    目录: C:\Users\LENOVO


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2025/9/28     14:05                public_html
```

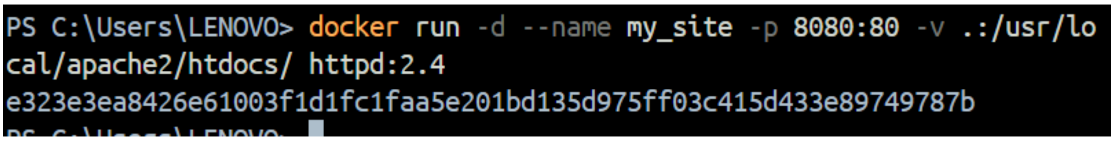
Navigate into the newly created directory public_html and create a file called index.html with the following content. This is a basic HTML document that creates a simple webpage that welcomes you with a friendly whale



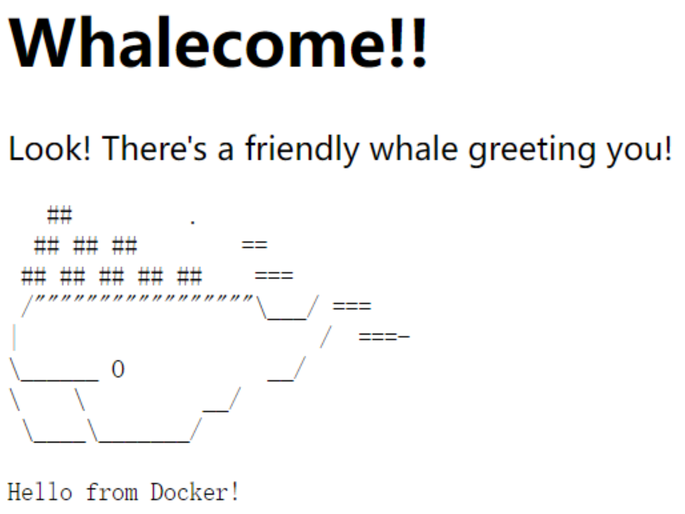
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title> My Website with a Whale & Docker!</title>
</head>
<body>
<h1>Whalecome!!</h1>
<p>Look! There's a friendly whale greeting you!</p>
<pre id="docker-art">
   ##         .
  ## ## ##        ==
 ## ## ## ## ##    ===
 /"""""""""""""""""\___/ ===
|                       /  ===-
\______ o           __/
\    \         __/
 \____\_______/

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

run the container



```
PS C:\Users\LENOVO> docker run -d --name my_site -p 8080:80 -v .:/usr/lo
cal/apache2/htdocs/ httpd:2.4
e323e3ea8426e61003f1d1fc1faa5e201bd135d975ff03c415d433e89749787b
PS C:\Users\LENOVO>
```



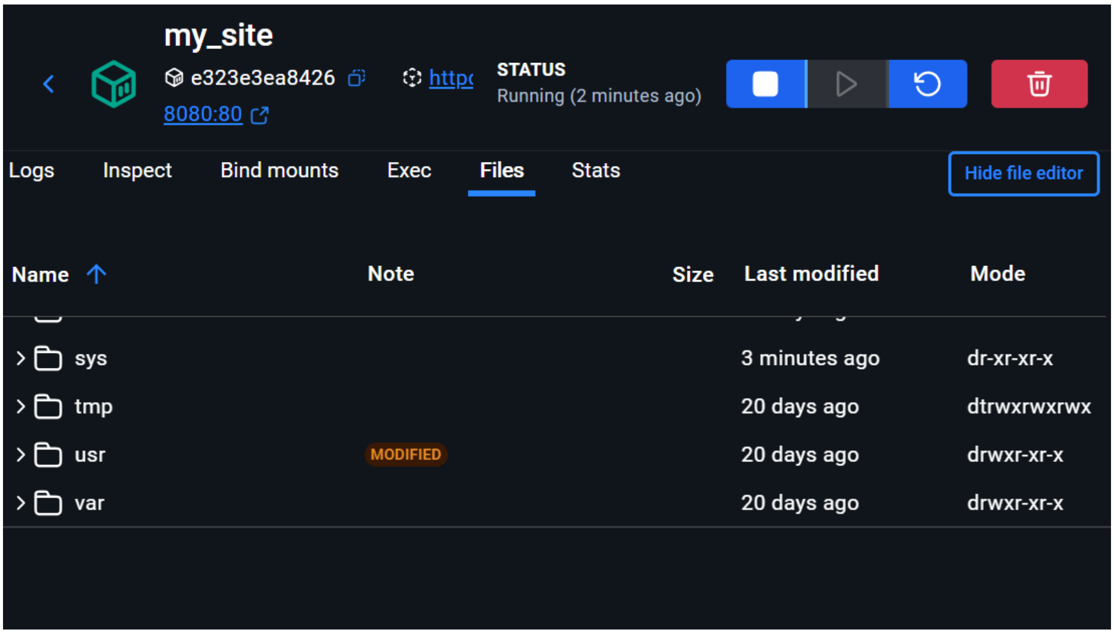
Whalecome!!

Look! There's a friendly whale greeting you!

```
   ##         .
  ## ## ##        ==
 ## ## ## ## ##    ===
 /"""""""""""""""""\___/ ===
|                       /  ===-
\______ o           __/
\    \         __/
 \____\_______/

Hello from Docker!
```

Access the file on the Docker Desktop Dashboard



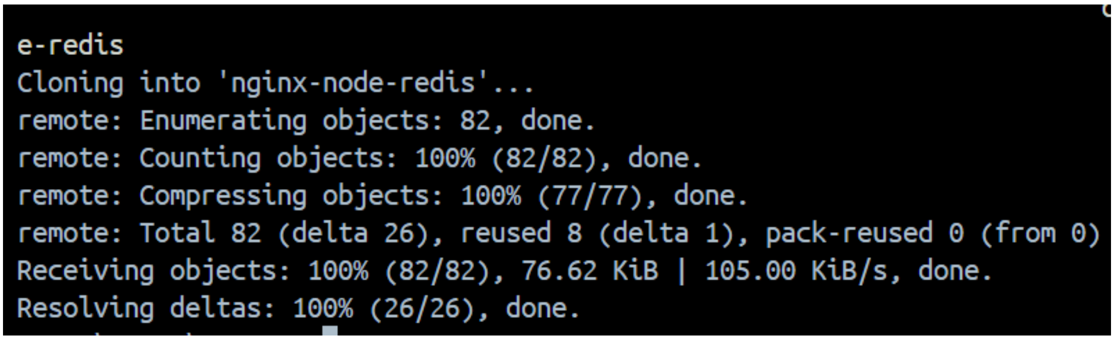
Name	Note	Size	Last modified	Mode
sys			3 minutes ago	dr-xr-xr-x
tmp			20 days ago	dtrwxrwxrwx
usr	MODIFIED		20 days ago	drwxr-xr-x
var			20 days ago	drwxr-xr-x

Delete the file on the host and verify the file is also deleted in the container. You will find that the files no longer exist under **Files** in the Docker Desktop Dashboard.

Recreate the HTML file on the host system and see that file re-appears under the **Files** tab under **Containers** on the Docker Desktop Dashboard. By now, you will be able to access the site too.

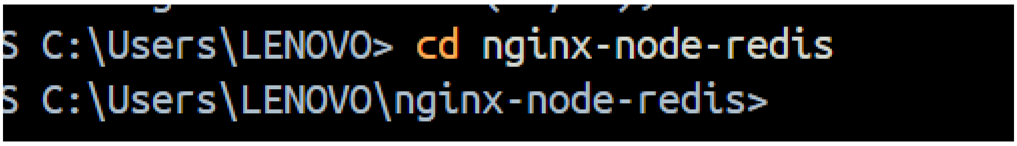
build and run a counter web application based on Node.js, an Nginx reverse proxy, and a Redis database using the docker run commands.

Use the following command in a terminal to clone the sample application repository.

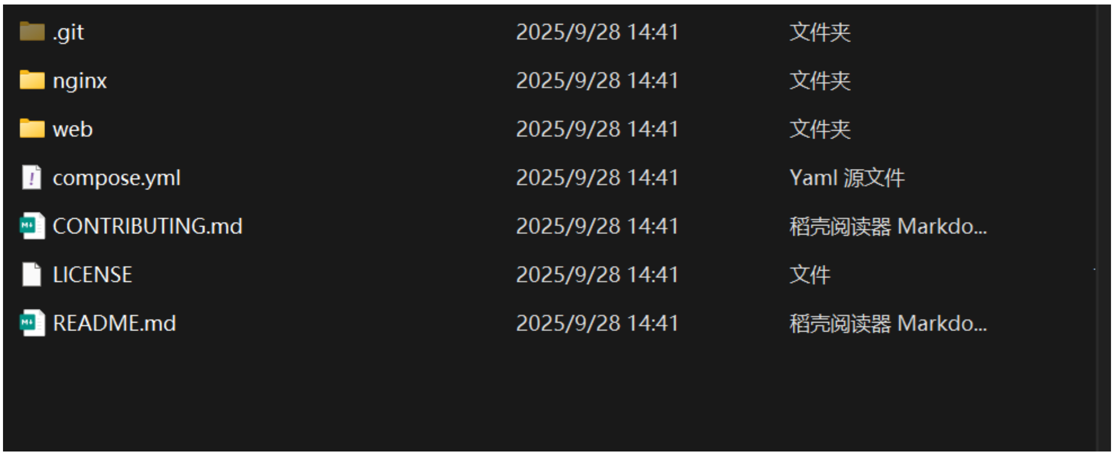


```
e-redis
Cloning into 'nginx-node-redis'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 82 (delta 26), reused 8 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (82/82), 76.62 KiB | 105.00 KiB/s, done.
Resolving deltas: 100% (26/26), done.
```

Navigate into the nginx-node-redis directory:

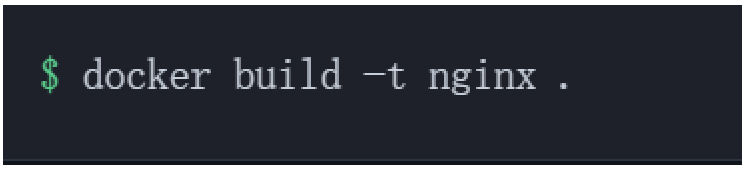


```
S C:\Users\LENOVO> cd nginx-node-redis
S C:\Users\LENOVO\nginx-node-redis>
```



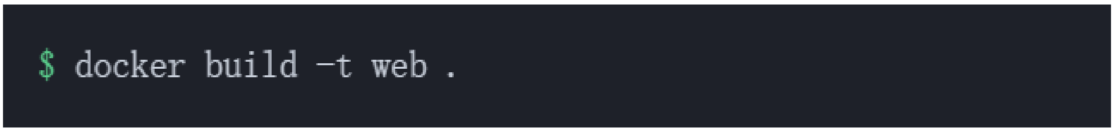
.git	2025/9/28 14:41	文件夹
nginx	2025/9/28 14:41	文件夹
web	2025/9/28 14:41	文件夹
compose.yml	2025/9/28 14:41	Yaml 源文件
CONTRIBUTING.md	2025/9/28 14:41	稻壳阅读器 Markdo...
LICENSE	2025/9/28 14:41	文件
README.md	2025/9/28 14:41	稻壳阅读器 Markdo...

Navigate into the nginx directory to build the image by running the following command:



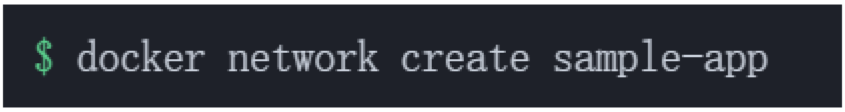
```
$ docker build -t nginx .
```

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



```
$ docker build -t web .
```

create a network for them all to communicate through.



```
$ docker network create sample-app
```

Start the Redis container by running the following command, which will attach it to the previously created network and create a network alias

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

Start the first web container by running the following command:

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

Start the second web container by running the following:

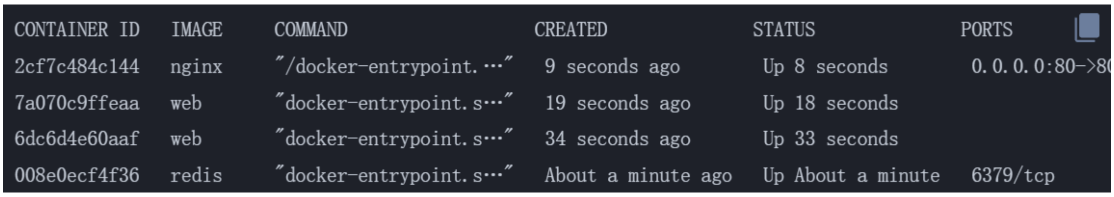
```
docker run -d --name web2 -h web2 --network sample-app --network-alias web2 web
```

Start the Nginx container by running the following command:

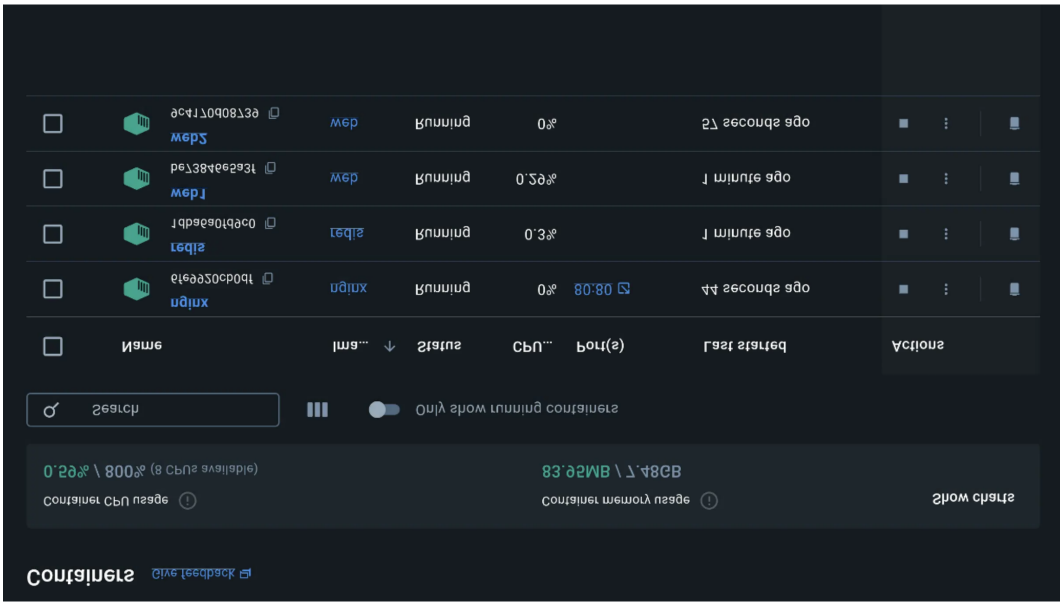
```
docker run -d --name nginx --network sample-app  -p 80:80 nginx
```

Verify the containers are up by running the following command:

```
docker ps
```



```
CONTAINER ID   IMAGE    COMMAND                   CREATED              STATUS               PORTS
2cf7c484c144   nginx    "/docker-entrypoint.…"    9 seconds ago        Up 8 seconds         0.0.0.0:80->8
7a070c9ffeaa   web      "docker-entrypoint.s…"    19 seconds ago       Up 18 seconds
6dc6d4e60aaf   web      "docker-entrypoint.s…"    34 seconds ago       Up 33 seconds
008e0ecf4f36   redis    "docker-entrypoint.s…"    About a minute ago   Up About a minute    6379/tcp
```

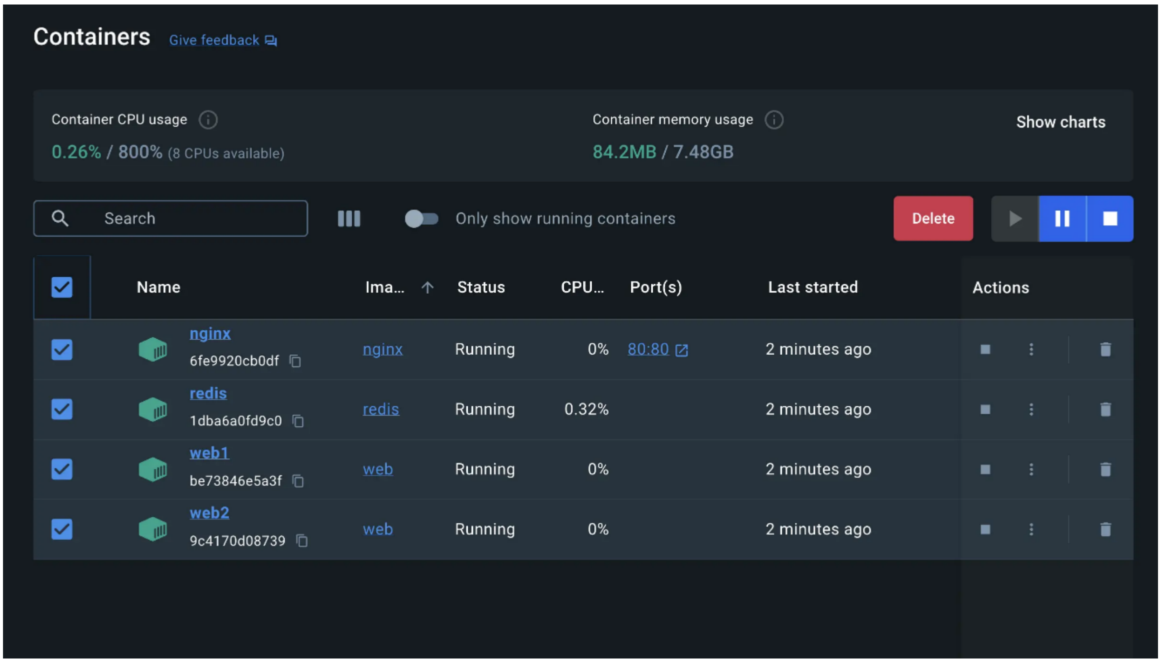


Containers Give feedback

Container CPU usage
0.26% / 800% (8 CPUs available)

Container memory usage
83.95MB / 7.48GB

□	Name	Ima...	Status	CPU...	Port(s)	Last started	Actions
□	nginx 6fe9920cb0df	nginx	Running	0%	80:80	44 seconds ago	
□	redis 1dba6a0fd9c0	redis	Running	0.3%		1 minute ago	
□	web1 be73846e5a3f	web	Running	0.29%		1 minute ago	
□	web2 9c4170d08739	web	Running	0%		57 seconds ago	



Containers

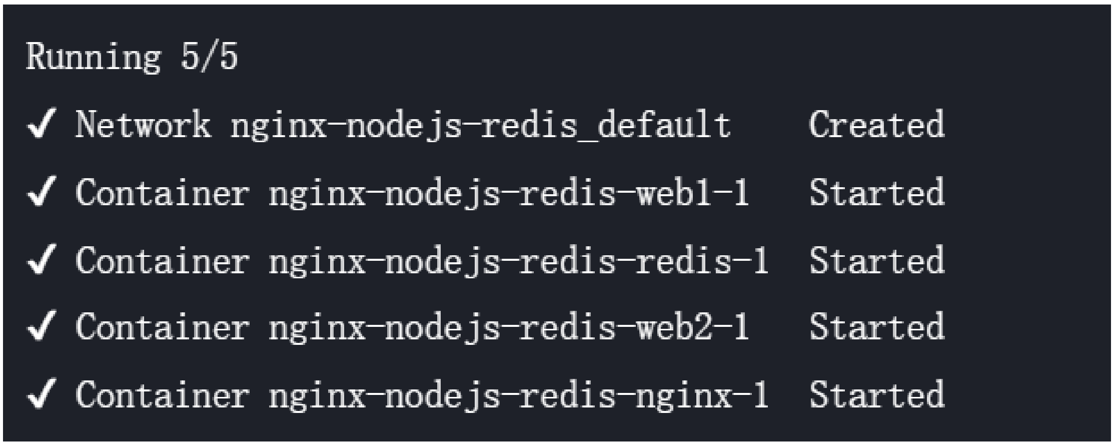
Give feedback

Container CPU usage
0.26% / 800% (8 CPUs available)

Container memory usage
84.2MB / 7.48GB

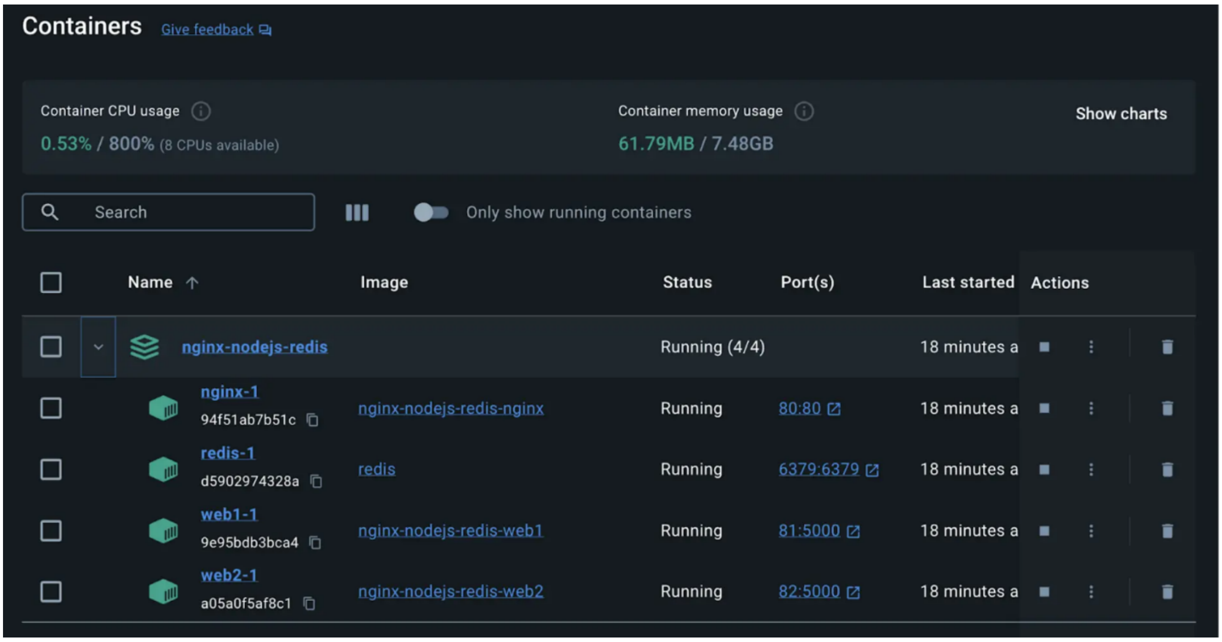
☑	Name	Ima...	Status	CPU...	Port(s)	Last started	Actions
☑	nginx 6fe9920cb0df	nginx	Running	0%	80:80	2 minutes ago	
☑	redis 1dba6a0fd9c0	redis	Running	0.32%		2 minutes ago	
☑	web1 be73846e5a3f	web	Running	0%		2 minutes ago	
☑	web2 9c4170d08739	web	Running	0%		2 minutes ago	

Use the docker compose up command to start the application
docker compose up -d –build



```
Running 5/5
✓ Network nginx-nodejs-redis_default      Created
✓ Container nginx-nodejs-redis-web1-1     Started
✓ Container nginx-nodejs-redis-redis-1    Started
✓ Container nginx-nodejs-redis-web2-1     Started
✓ Container nginx-nodejs-redis-nginx-1    Started
```

If you look at the Docker Desktop Dashboard, you can see the containers and dive deeper into their configuration.



Containers Give feedback

Container CPU usage 0.53% / 800% (8 CPUs available)

Container memory usage 61.79MB / 7.48GB

□	Name	Image	Status	Port(s)	Last started	Actions
□	nginx-nodejs-redis		Running (4/4)		18 minutes a	
□	nginx-1 94f51ab7b51c	nginx-nodejs-redis-nginx	Running	80:80	18 minutes a	
□	redis-1 d5902974328a	redis	Running	6379:6379	18 minutes a	
□	web1-1 9e95bdb3bca4	nginx-nodejs-redis-web1	Running	81:5000	18 minutes a	
□	web2-1 a05a0f5af8c1	nginx-nodejs-redis-web2	Running	82:5000	18 minutes a	