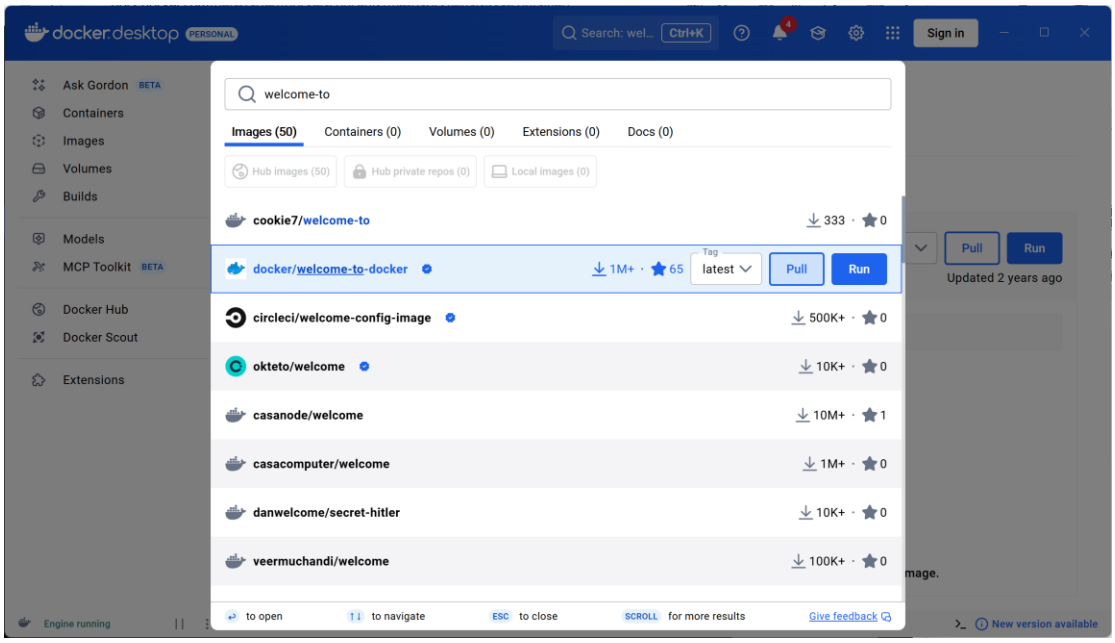
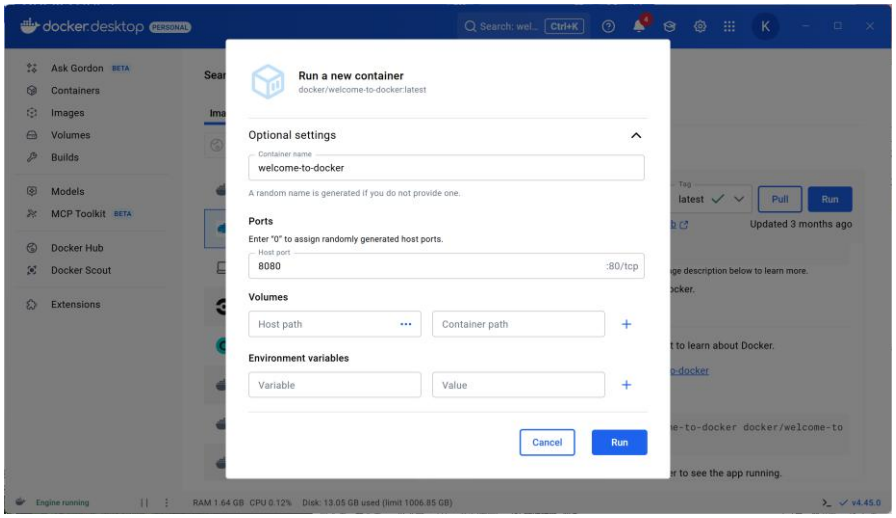
Lab 5

What is a container?

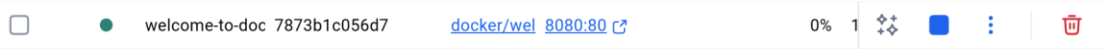
1. Open Docker Desktop and select the **Search** field on the top navigation bar.
2. Specify welcome-to-docker in the search input and then select the **Pull** button.



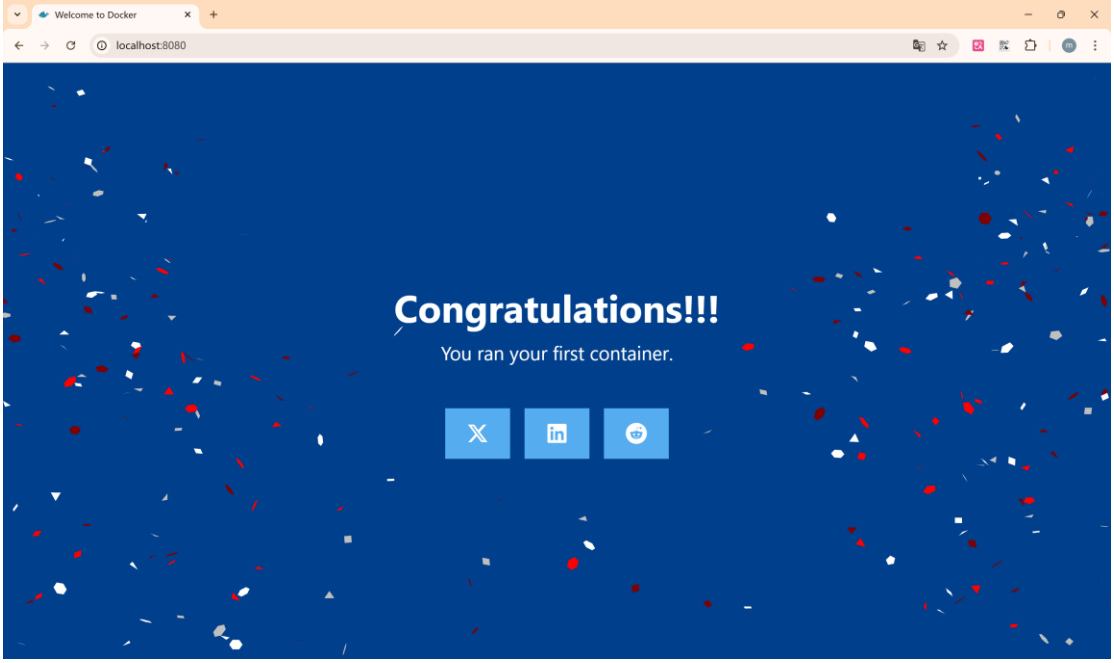
3. Once the image is successfully pulled, select the Run button.
4. Expand the Optional settings.
5. In the Container name, specify welcome-to-docker.
6. In the Host port, specify 8080.
7. Select Run to start your container.



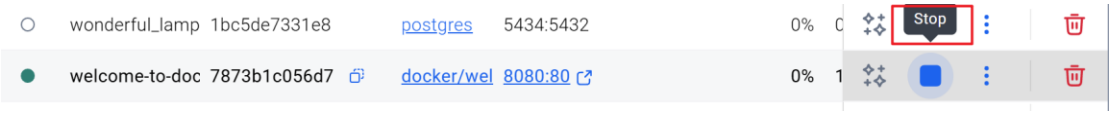
8. View your container:



9. Access the frontend:



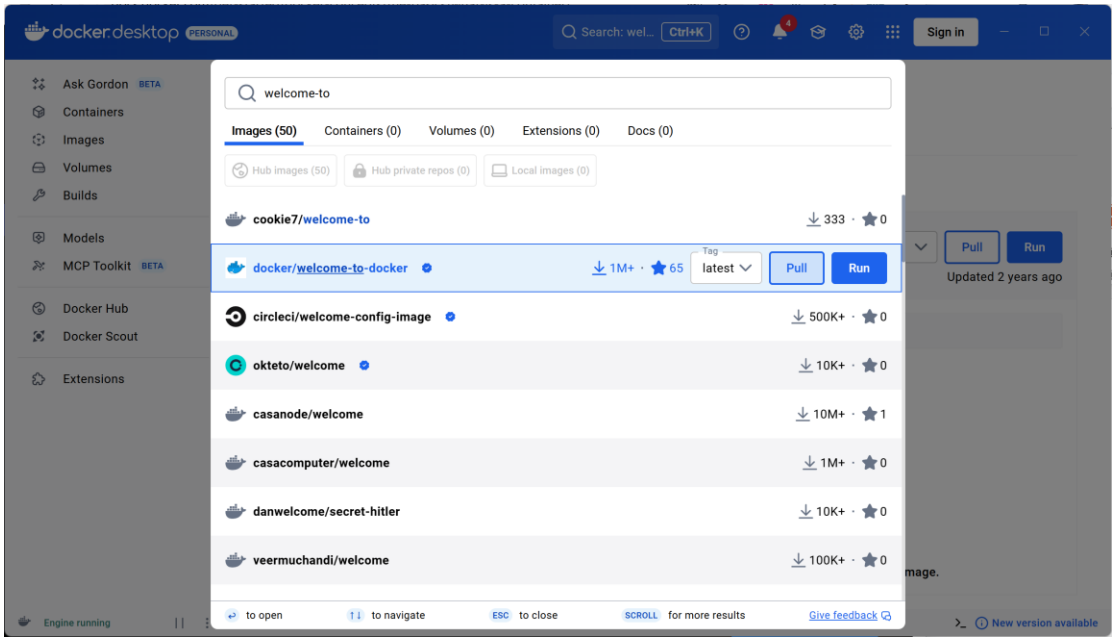
10. Stop your container:



What is an image?

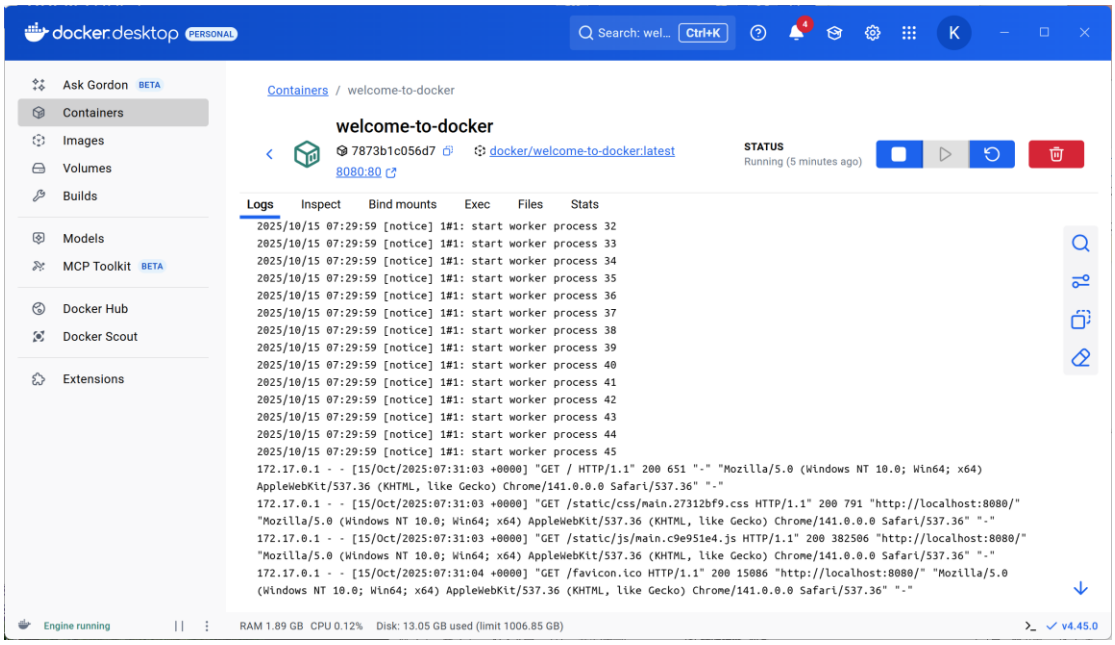
Search for and download an image

1. Open the Docker Desktop Dashboard and select the **Images** view in the left-hand navigation menu.
2. Select the **Search images to run** button.
3. In the **Search** field, enter "welcome-to-docker". Once the search has completed, select the docker/welcome-to-docker image.
4. Select Pull to download the image.



Learn about the image

1. In the Docker Desktop Dashboard, select the **Images** view.
2. Select the **docker/welcome-to-docker** image to open details about the image.



3. The image details page presents you with information regarding the layers of the image, the packages and libraries installed in the image, and any discovered vulnerabilities.

What is Docker Compose?

Start the application

1. Open a terminal and clone this sample application:

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

```
PS C:\study\云计算导论\Labs\Lab5> 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 | 2.93 MiB/s, done.
Resolving deltas: 100% (15/15), done.
```

2. Navigate into the todo-list-app directory:

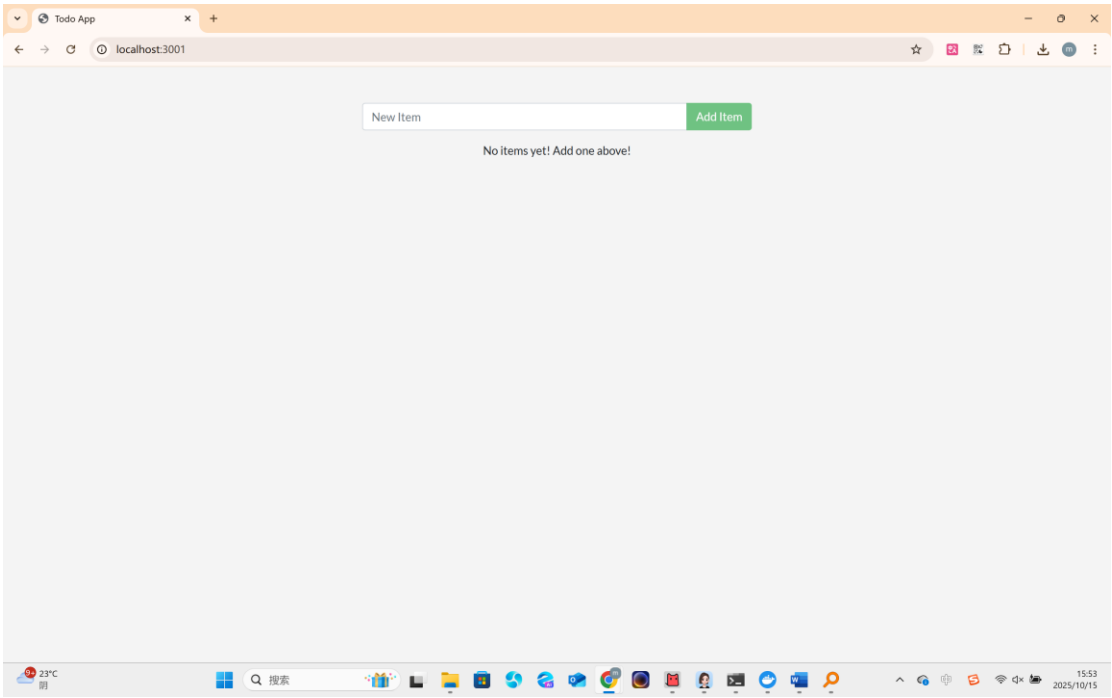
   cd .\todo-list-app\

3. Use the docker compose up command to start the application:

   docker compose up -d –build

```
PS C:\study\云计算导论\Labs\Lab5\todo-list-app> docker compose up -d --build
[+] Running 2/2
 ✔Container todo-list-app-mysql-1  Running                                    0.0s
 ✔Container todo-list-app-app-1    Started                                    1.4s
```

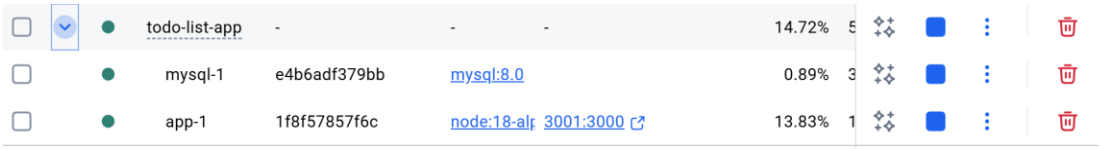
4. With everything now up and running, you can open http://localhost:3001 in your browser to see the site.



   (I changed the port from 3000 to 3001 in compose.yml as 3000 has been allocated)

```
ports:
  - 127.0.0.1:3001:3000
```

5. See the container in docker desktop:



Tear it down

1. In the CLI, use the docker compose down command to remove everything:

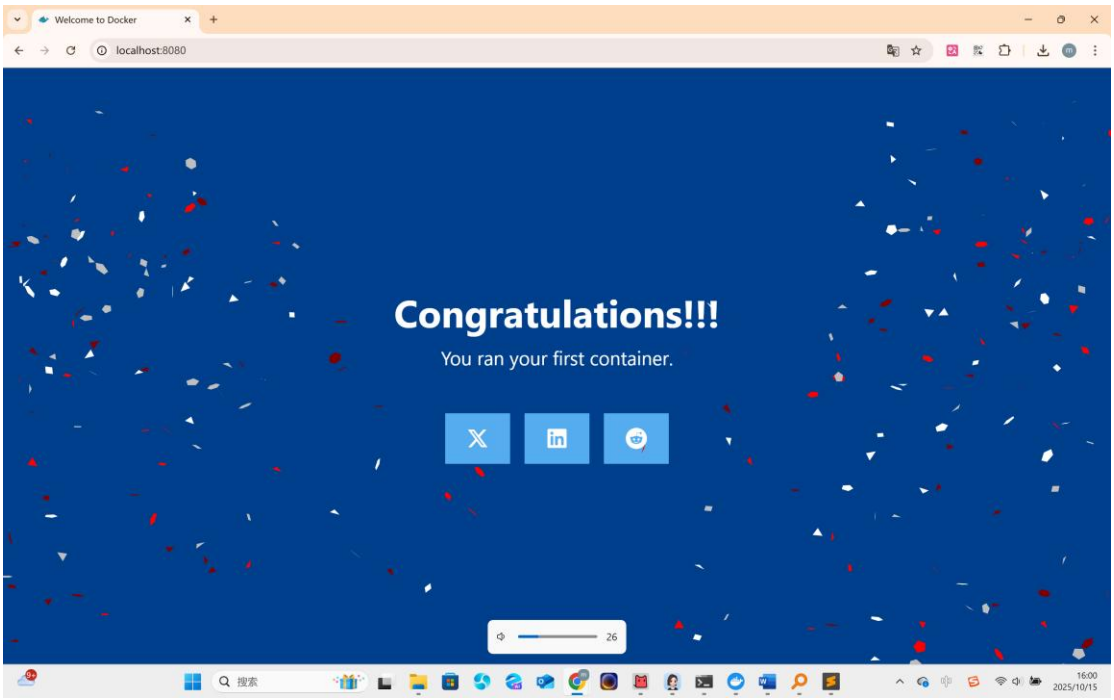
```
PS C:\study\云计算导论\Labs\Lab5\todo-list-app> docker compose down
[+] Running 3/3
 ✔Container todo-list-app-mysql-1  Removed                                    4.1s
 ✔Container todo-list-app-app-1    Removed                                    1.3s
 ✔Network todo-list-app_default    Removed                                    0.8s
```

2. Or use delete button in GUI.

Publishing and exposing ports

1. In a terminal, run the following command to start a new container:
   docker run -d -p 8080:80 docker/welcome-to-docker
2. Verify the published port by going to the **Containers** view of the Docker Desktop Dashboard.

3. Open the website by either selecting the link in the Port(s) column of your container or visiting http://localhost:8080 in your browser.



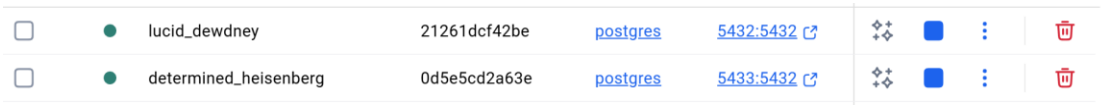
Overriding container defaults

Run multiple instances of the Postgres database

1. Start a container using the Postgres image with the following command:
   docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgres
2. Start a second Postgres container mapped to a different port.
   docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgres

```
PS C:\study\云计算导论\Labs\Lab5> docker run -d -e POSTGRES_PASSWORD=secret -p 5432:5432 postgre
21261dcf42be5eac65bb8d32bf09adefe2ff182692a50af8cab67f6fcbe43bef
PS C:\study\云计算导论\Labs\Lab5> docker run -d -e POSTGRES_PASSWORD=secret -p 5433:5432 postgre
0d5e5cd2a63e6c134e806eda4f9c9a16ff02e504fa2cb62da0beb7ea2f96f8fc
```

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



Run Postgres container in a controlled network

1. Create a new custom network by using the following command:
   docker network create mynetwork
2. Verify the network by running the following command:
   docker network ls

```
PS C:\study\云计算导论\Labs\Lab5> docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
61356d6bde49   bridge            bridge    local
27b910916eb9   gitea_gitea       bridge    local
6e003f9bcecc   host              host      local
de58a192ed0b   minikube          bridge    local
46b945fdc848   mynetwork         bridge    local
56c10408955f   none              null      local
7a6de00a2332   postgres_default  bridge    local
```

3. Connect Postgres to the custom network by using the following command:
   docker run -d -e POSTGRES_PASSWORD=secret -p 5434:5432 --network mynetwork postgres

Manage the resources

1. By default, containers are not limited in their resource usage. However, on shared systems, it's crucial to manage resources effectively. It's important not to let a running container consume too much of the host machine's memory.

2. This is where the **docker run** command shines again. It offers flags like **--memory** and **--cpus** to restrict how much CPU and memory a container can use:
   docker run -d -e POSTGRES_PASSWORD=secret --memory="512m" --cpus=".5" postgres
3. 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.

Override the default CMD and ENTRYPOINT in Docker Compose

Sometimes, you might need to override the default commands (CMD) or entry points (ENTRYPOINT) defined in a Docker image, especially when using Docker Compose.

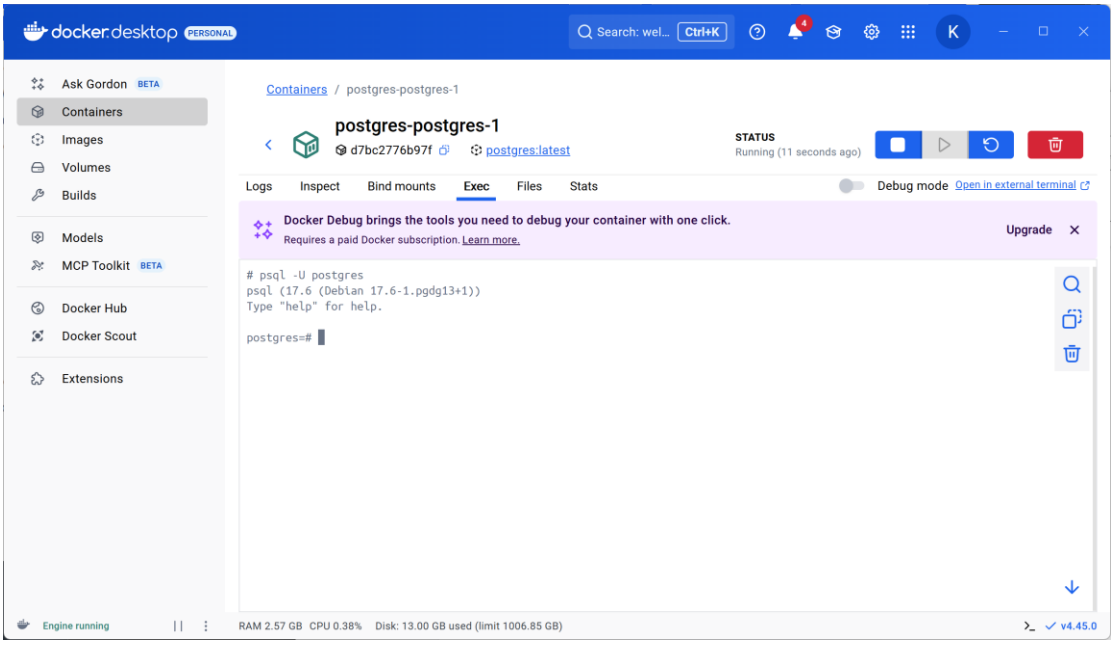
1. Create a compose.yml file with the following content:

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

2. Bring up the service by running the following command:
   docker compose up -d

```
PS C:\study\云计算导论\Labs\Lab5> docker compose up -d
[+] Running 2/2
 ✔ Network lab5_default        Created                                    0.4s
 ✔ Container lab5-postgres-1   Started                                    0.9s
```

3. Verify the authentication with Docker Desktop Dashboard.
   Open the Docker Desktop Dashboard, select the Postgres container and select Exec to enter into the container shell. You can type the following command to connect to the Postgres database:
   psql -U postgres



4. You can also override defaults directly using the docker run command with the following command:
   docker run -e POSTGRES_PASSWORD=secret postgres docker-entrypoint.sh -h localhost -p 5432
   This command runs a Postgres container, sets an environment variable for password authentication, overrides the default startup commands and configures hostname and port mapping.

Persisting container data

Use volumes

1. Start a container using the Postgres image with the following command:
   docker run --name=db -e POSTGRES_PASSWORD=secret -d -v postgres_data:/var/lib/postgresql/data postgres
2. Connect to the database by using the following command:
   docker exec -ti db psql -U postgres

```
PS C:\study\云计算导论\Labs\Lab5> docker run --name=db -e POSTGRES_PASSWORD=secret -d -v postgres_data:/var/lib/postgres
ql/data postgres
2887c130f3a2a9fbe6d265bd61c95fd13d09877a3128fc286894b3742cc1dcc5
PS C:\study\云计算导论\Labs\Lab5> docker exec -ti db psql -U postgres
psql (17.6 (Debian 17.6-1.pgdg13+1))
Type "help" for help.

postgres=#
```

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

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

4. 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)
```

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

   \q

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

   docker stop db

   docker rm db

```
PS C:\study\云计算导论\Labs\Lab5>  docker stop db
db
PS C:\study\云计算导论\Labs\Lab5>  docker rm db
db
```

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

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

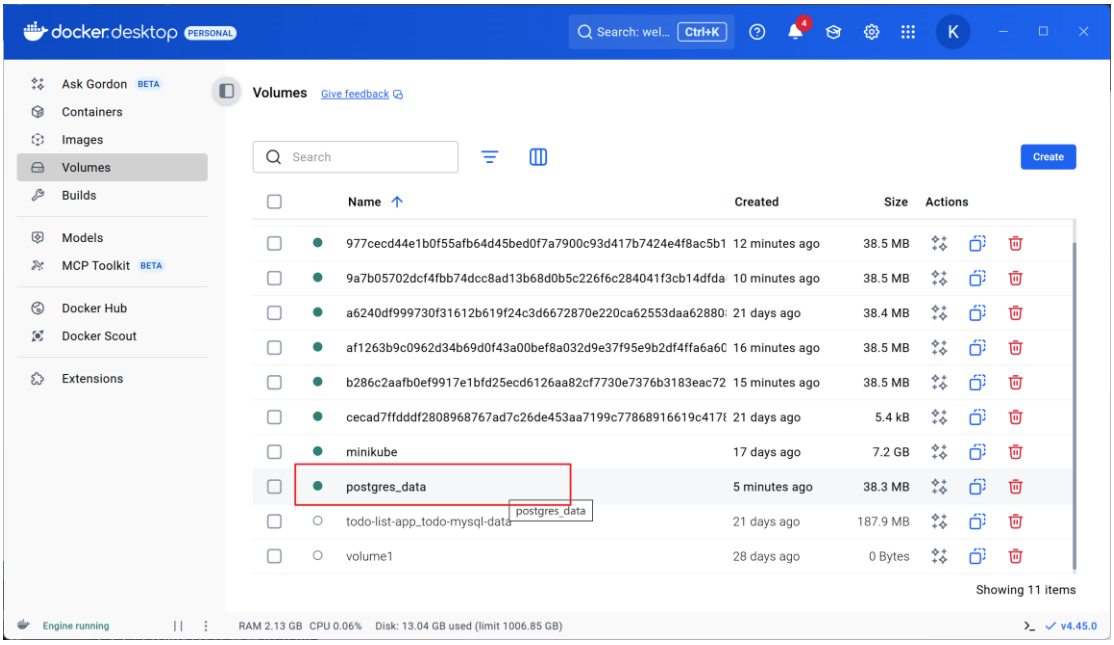
8. Verify the database still has the records by running the following command:

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

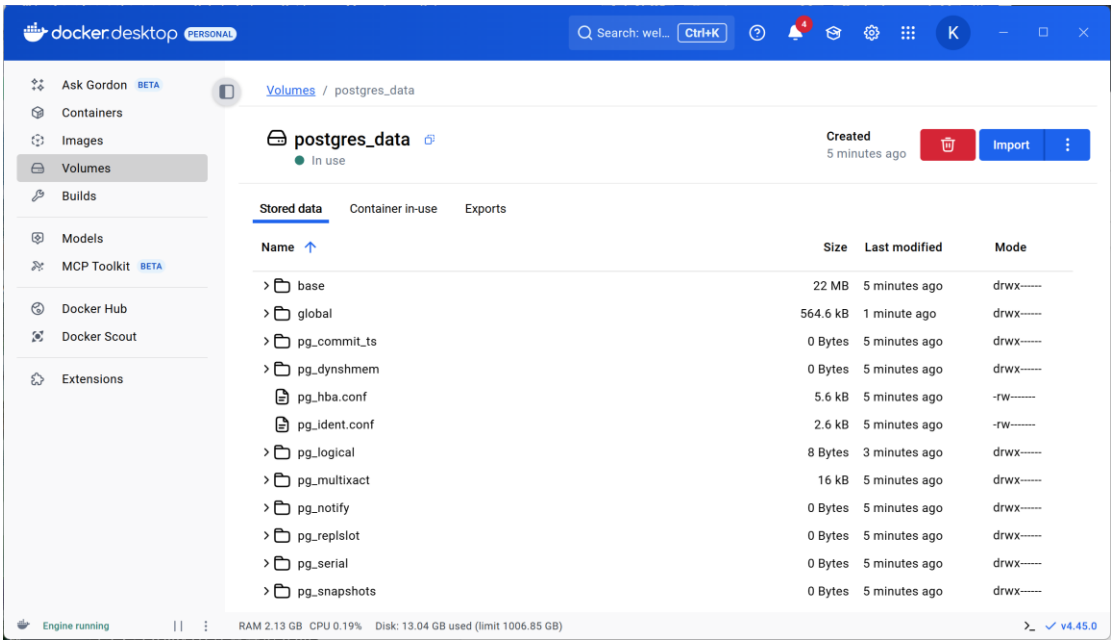
```
PS C:\study\云计算导论\Labs\Lab5> docker exec -ti new-db psql -U postgres -c "SELECT * FROM tasks"
 id | description
----+-------------
  1 | Finish work
  2 | Have fun
(2 rows)
```

View volume contents

1. Open the Docker Desktop Dashboard and navigate to the **Volumes** view. In this view, you should see the **postgres_data** volume.



2. Select the postgres_data volume's name.
3. The Data tab shows the contents of the volume and provides the ability to navigate the files. Double-clicking on a file will let you see the contents and make changes.



Remove volumes

1. Before removing a volume, it must not be attached to any containers. If you haven't removed the previous container, do so with the following command (the **-f** will stop the container first and then remove it):

```
docker rm -f new-db
```

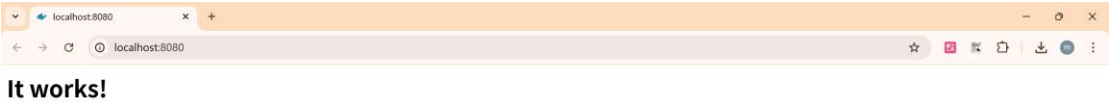
2. Use the **docker volume prune** command to remove all unused volumes:
   docker volume prune

Sharing local files with containers

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

```
PS C:\study\云计算导论\Labs\Lab5> docker run -d -p 8080:80 --name my_site httpd:2.4
85c1df9803de23bc1ca4e023105dea8db5fbe7e4007479bbe230e1f4a08e4f68
```

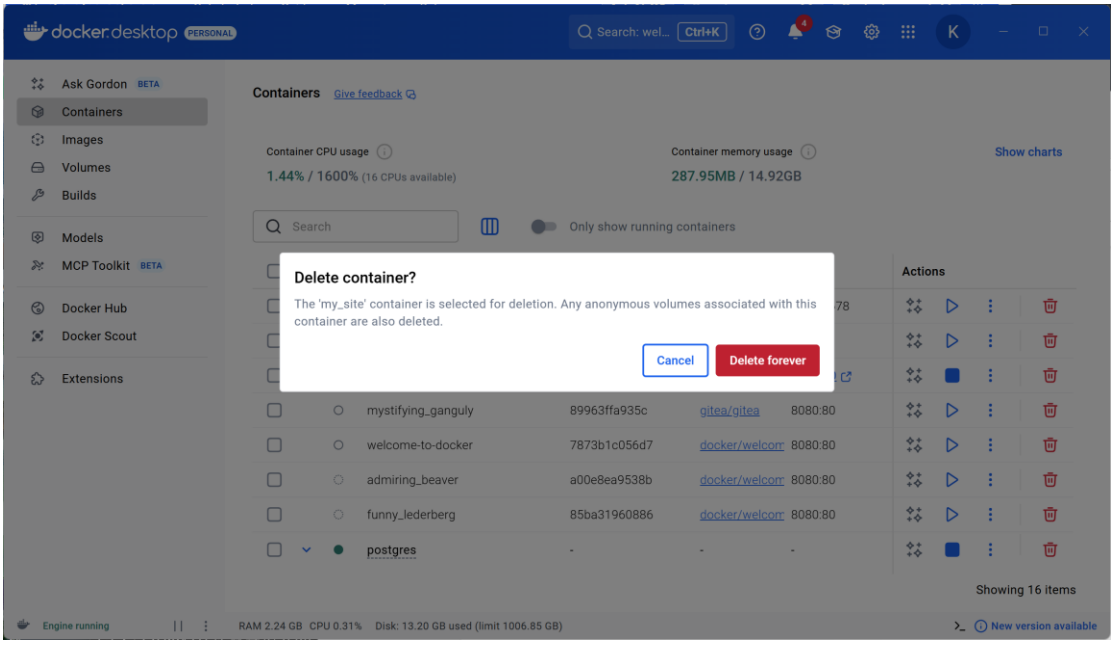
Open the browser and access http://localhost:8080:



Use a bind mount

Using a bind mount, you can map the configuration file on your host computer to a specific location within the container. In this example, you'll see how to change the look and feel of the webpage by using bind mount:

1. Delete the existing container by using the Docker Desktop Dashboard:



2. Create a new directory called public_html on your host system.
   mkdir public_html
3. 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">

<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>

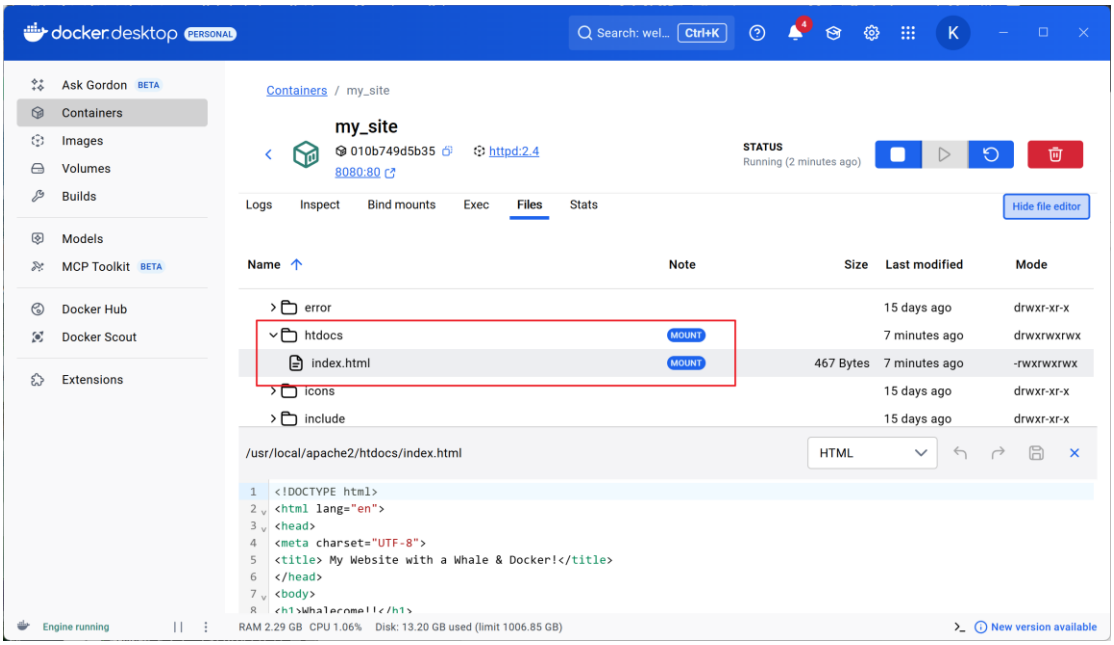
</body>

</html>

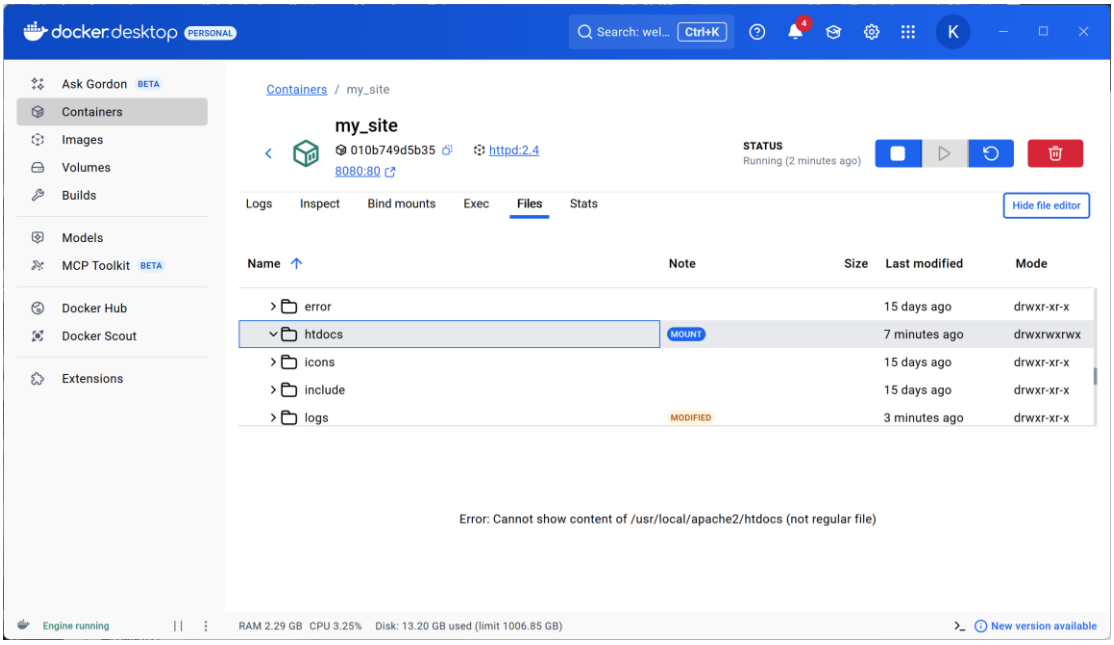
4. It's time to run the container. The --mount and -v examples produce the same result. You can't run them both unless you remove the my_site container after running the first one.
   docker run -d --name my_site -p 8080:80 --mount type=bind,source=C:\study\云计算导论\Labs\Lab5\public_html,target=/usr/local/apache2/htdocs/ httpd:2.4

Access the file on the Docker Desktop Dashboard

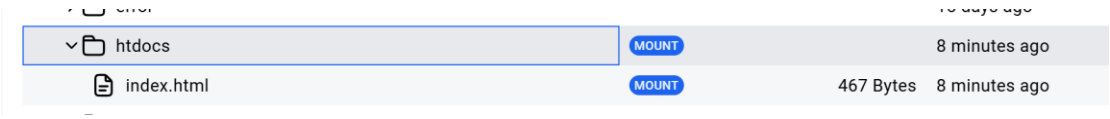
1. You can view the mounted files inside a container by selecting the container's **Files** tab and then selecting a file inside the /usr/local/apache2/htdocs/ directory. Then, select **Open file editor**.



2. 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.



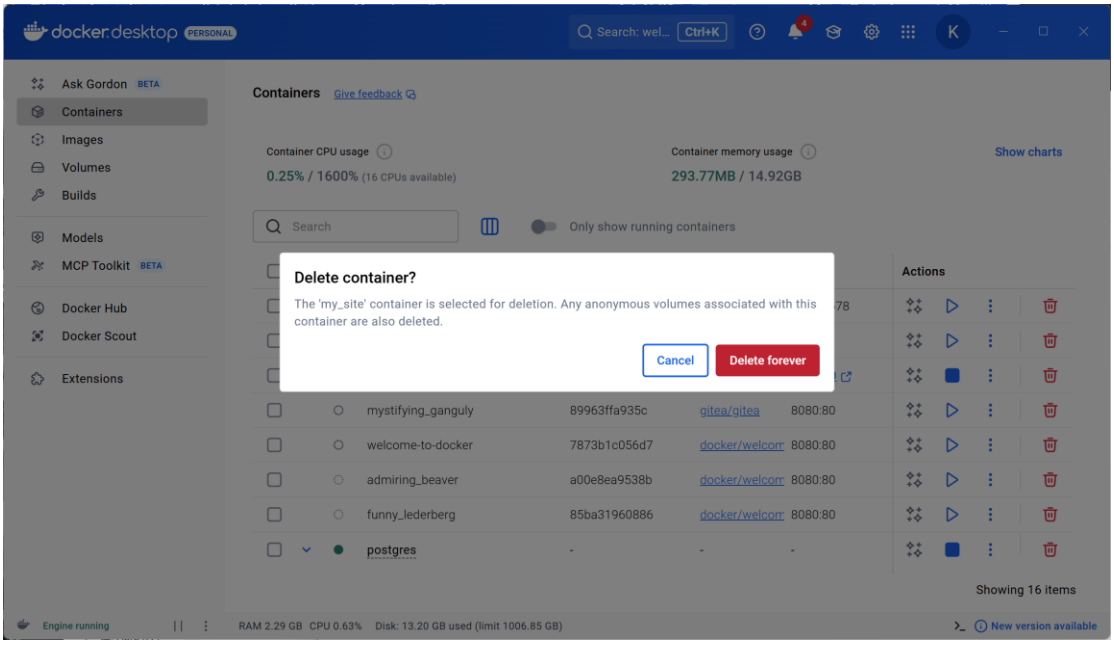
3. Recreate the HTML file on the host system and see that file re-appears under the Files tab under Containers on the Docker Desktop Dashboard. By now, you will be able to access the site too.



Stop your container

The container continues to run until you stop it.

1. Go to the **Containers** view in the Docker Desktop Dashboard.
2. Locate the container you'd like to stop.
3. Select the **Delete** action in the Actions column.



Multi-container applications

Set up

1. Get the sample application.

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

```
PS C:\study\云计算导论\Labs\Lab5> git clone https://github.com/dockersamples/nginx-node-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 | 1.30 MiB/s, done.
Resolving deltas: 100% (26/26), done.
```

2. Navigate into the nginx-node-redis directory:

   cd .\nginx-node-redis\

Build the images

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

   docker build -t nginx .

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis\nginx> docker build -t nginx .
[+] Building 10.7s (9/9) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.1s
 => => transferring dockerfile: 141B                                                                                    0.0s
 => [internal] load metadata for docker.io/library/nginx:1.29                                                           3.7s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                            0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [1/3] FROM docker.io/library/nginx:1.29@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486a0  5.2s
 => => resolve docker.io/library/nginx:1.29@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486a0  0.0s
 => => sha256:58d144c4badd75d84d5fcf15ff49774be2abf4e24d7d3e3d2311916213d59496 957B / 957B                             0.6s
 => => sha256:b459da543435bf9437ad7530b6c62d7d40b9e8d84b91945cb0443d4a14664ca2 405B / 405B                             0.9s
 => => sha256:8da8ed3552af163aaabf15e69703265c1770edbe15bc4cad5cbfcfa0ac943109 1.21kB / 1.21kB                         1.1s
 => => sha256:54e822d8ee0c638c33ce3efb93b0913fef8107da4436a7116b986ef44c281182 0B / 1.40kB                             6.5s
 => => sha256:250b90fb2b9a5a65efddd2ffe8f3b24b04b8463f43c69b759cbb85b72329380b 32.90MB / 32.90MB                       2.8s
 => => sha256:5d8ea9f4c626de724115534b787f32f091b5f5e41668a4a33a385466e8e3e185 629B / 629B                             0.6s
 => => extracting sha256:250b90fb2b9a5a65efddd2ffe8f3b24b04b8463f43c69b759cbb85b72329380b                              1.4s
 => => extracting sha256:5d8ea9f4c626de724115534b787f32f091b5f5e41668a4a33a385466e8e3e185                              0.0s
 => => extracting sha256:58d144c4badd75d84d5fcf15ff49774be2abf4e24d7d3e3d2311916213d59496                              0.0s
 => => extracting sha256:b459da543435bf9437ad7530b6c62d7d40b9e8d84b91945cb0443d4a14664ca2                              0.0s
 => => extracting sha256:8da8ed3552af163aaabf15e69703265c1770edbe15bc4cad5cbfcfa0ac943109                              0.0s
 => => extracting sha256:54e822d8ee0c638c33ce3efb93b0913fef8107da4436a7116b986ef44c281182                              0.0s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 222B                                                                                       0.0s
 => [2/3] RUN rm /etc/nginx/conf.d/default.conf                                                                         0.8s
 => [3/3] COPY nginx.conf /etc/nginx/conf.d/default.conf                                                                0.1s
 => exporting to image                                                                                                  0.5s
 => => exporting layers                                                                                                 0.2s
 => => exporting manifest sha256:b3ef71a86d65603828cfdefaee6b5bfa63b38f7aaf2e91cd70caec1719178a62                        0.0s
 => => exporting config sha256:03e0dff5d793b1b5d74cae8890357fe7ed3f0f89ae60940f73950b8320902955                         0.0s
 => => exporting attestation manifest sha256:d9c426b7e5df8e9b4af8869a825e40a3cd02bf0c1951af9c0177674017accaca            0.0s
 => => exporting manifest list sha256:73a1fff08d31b7abc258df9e35577350561c2be8850df31e4230120fe8227fd7                  0.0s
 => => naming to docker.io/library/nginx:latest                                                                         0.0s
 => => unpacking to docker.io/library/nginx:latest                                                                      0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/5hf8nool7tej2r9kyuvy1rfbx
```

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

   docker build -t web .

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis\web> docker build -t web .
[+] Building 25.8s (5/10)                                                                            docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.1s
[+] Building 25.9s (5/10)                                                                            docker:desktop-linux
 => [internal] load metadata for docker.io/library/node:21                                                              4.2s
 => [auth] library/node:pull token for registry-1.docker.io                                                             0.0s
 => [internal] load .dockerignore                                                                                       0.1s
 => => transferring context: 2B                                                                                         0.0s
 => [1/5] FROM docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56c8d207a67270868f51b3d155c4e33d  21.6s
[+] Building 26.1s (5/10)                                                                            docker:desktop-linux
 => [1/5] FROM docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56c8d207a67270868f51b3d155c4e33d  21.7s
 => => # error: failed to copy: httpReadSeeker: failed open: failed to do request: Get "https://registry-1.docker.io/v2
 => => # /library/node/blobs/sha256:6582c62583ef22717db8d306b1d6a0c288089ff607d9c0d2d81c4f8973cbfee3": EOF
 => => # retrying in 1s
 => => # error: failed to copy: httpReadSeeker: failed open: failed to do request: Get "https://registry-1.docker.io/v2
[+] Building 50.5s (11/11) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.1s
 => => transferring dockerfile: 175B                                                                                    0.0s
 => [internal] load metadata for docker.io/library/node:21                                                              4.2s
 => [auth] library/node:pull token for registry-1.docker.io                                                             0.0s
 => [internal] load .dockerignore                                                                                       0.1s
 => => transferring context: 2B                                                                                         0.0s
 => [1/5] FROM docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56c8d207a67270868f51b3d155c4e33d  38.0s
 => => resolve docker.io/library/node:21@sha256:4b232062fa976e3a966c49e9b6279efa56c8d207a67270868f51b3d155c4e33d   0.1s
 => => sha256:5d3106ce01c367e8274a6e83a2d6f3d10ecf9619db69a0b3eacebef36f214f88 449B / 449B                             0.6s
 => => sha256:c6cf28de8a067787ee0d08f8b01d7f1566a508b56f6e549687b41dfd375f12c7 49.58MB / 49.58MB                      11.7s
 => => sha256:08750d53428eebb9ceb8fd6d9516a08420d7747d6a740951c2b043d1ef88c150 1.25MB / 1.25MB                         1.6s
 => => sha256:2b542796ccab17c22fccf6f4a00414f43ecabd5a5d68fe53f8c8c5aee498b8dd 3.37kB / 3.37kB                         1.3s
 => => sha256:4684276225df21ffeac91884cd316f7bc0d99baeef8ff811f98c2ab46188c3cb 49.72MB / 49.72MB                       5.7s
 => => sha256:6582c62583ef22717db8d306b1d6a0c288089ff607d9c0d2d81c4f8973cbfee3 64.14MB / 64.14MB                      10.3s
 => => sha256:8914944355808bdd3db5552f0d3723fd0fa826675f774853796fafa221d850d42 24.05MB / 24.05MB                       7.4s
 => => sha256:bf2c3e352f3d2eed4eda4feeed44a1022a881058df20ac0584db70c138b041e2 211.21MB / 211.21MB                     23.5s
 => => extracting sha256:c6cf28de8a067787ee0d08f8b01d7f1566a508b56f6e549687b41dfd375f12c7                               3.6s
 => => extracting sha256:8914944355808bdd3db5552f0d3723fd0fa826675f774853796fafa221d850d42                               1.5s
 => => extracting sha256:6582c62583ef22717db8d306b1d6a0c288089ff607d9c0d2d81c4f8973cbfee3                               5.0s
 => => extracting sha256:bf2c3e352f3d2eed4eda4feeed44a1022a881058df20ac0584db70c138b041e2                               9.6s
 => => extracting sha256:2b542796ccab17c22fccf6f4a00414f43ecabd5a5d68fe53f8c8c5aee498b8dd                               0.0s
 => => extracting sha256:4684276225df21ffeac91884cd316f7bc0d99baeef8ff811f98c2ab46188c3cb                               4.0s
 => => extracting sha256:08750d53428eebb9ceb8fd6d9516a08420d7747d6a740951c2b043d1ef88c150                               0.1s
 => => extracting sha256:5d3106ce01c367e8274a6e83a2d6f3d10ecf9619db69a0b3eacebef36f214f88                               0.0s
 => [internal] load build context                                                                                       0.1s
 => => transferring context: 17.30kB                                                                                    0.0s
 => [2/5] WORKDIR /usr/src/app                                                                                          2.4s
 => [3/5] COPY package.json package-lock.json ./                                                                        0.1s
 => [4/5] RUN npm ci                                                                                                    3.6s
 => [5/5] COPY ./server.js ./                                                                                           0.1s
 => exporting to image                                                                                                  1.7s
 => => exporting layers                                                                                                 0.6s
 => => exporting manifest sha256:30f6119042568447a7ab78dddfc03270efbba6fc4f00a2a1942bae8651945f09                        0.0s
 => => exporting config sha256:2efa81938822368f5dc245d1a0552869e0d1549a30d2766f5141cf030ccf50f9                         0.0s
 => => exporting attestation manifest sha256:8363ea15bd852bb2d9ff877e5a4f7d8a0c8caedcb9a215733c77017c442c9ccd            0.0s
 => => exporting manifest list sha256:6f07e938c4c2ffbc3bda274a1d224a8d5148e970445b300834b79813f0abdabc                  0.0s
 => => naming to docker.io/library/web:latest                                                                           0.0s
 => => unpacking to docker.io/library/web:latest                                                                        0.9s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/jg1ekg3ht6phtbz1xfc4bzus8
```

Run the containers

1. Before you can run a multi-container application, you need to create a network for them all to communicate through. You can do so using the docker network create command:
   docker network create sample-app

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker network create sample-app
461451984dcccb8a9b2f5aa5512e958171c5bb2839abf8ff299ac542d17ea0f3
```

2. Start the Redis container by running the following command, 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

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker run -d  --name redis --network sample-app --network-alias redi
s redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
7e44f5a6338c: Pull complete
20770aaf8f7b: Pull complete
3ac4f782b24c: Pull complete
5c32499ab806: Pull complete
628b0785ec0d: Pull complete
4f4fb700ef54: Pull complete
fa85867e458c: Pull complete
Digest: sha256:f0957bcaa75fd58a9a1847c1f07caf370579196259d69ac07f2e27b5b389b021
Status: Downloaded newer image for redis:latest
c10bed0969fbc75f0ab45ce1d4372a78e5702221540e8b2d497996035d3444e8
```

3. Start the first web container by running the following command:
   docker run -d --name web1 -h web1 --network sample-app --network-alias web1 web
4. Start the second web container by running the following:
   docker run -d --name web2 -h web2 --network sample-app --network-alias web2 web
5. Start the Nginx container by running the following command:
   docker run -d --name nginx --network sample-app -p 80:80 nginx

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker run -d --name web1 -h web1 --network sample-app --network-alia
s web1 web
e0a6df65fe2714aad18a5ad6c08dd5d68e62b87841a53b45447ecc57c6487841
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker run -d --name web2 -h web2 --network sample-app --network-alia
s web2 web
2f66b47d6ce3b8a317e648c3d5a5518094872b705d9c841c516e27eff67f1af8
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker run -d --name nginx --network sample-app  -p 80:80 nginx
179dfef3fddb83651d4e435362803c5ab7a9cded0cbe438e83d3413d38d432ea
```

6. Verify the containers are up by running the following command:
   docker ps

```
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS              PORTS
                                                                                   NAMES
179dfef3fddb   nginx                          "/docker-entrypoint.…"   21 seconds ago       Up 21 seconds       0.0.0.0
:80->80/tcp, [::]:80->80/tcp                                                       nginx
2f66b47d6ce3   web                            "docker-entrypoint.s…"   44 seconds ago       Up 44 seconds
                                                                                   web2
e0a6df65fe27   web                            "docker-entrypoint.s…"   About a minute ago   Up About a minute
                                                                                   web1
c10bed0969fb   redis                          "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tc
p                                                                                  redis
22fa0e431ff2   postgres                       "docker-entrypoint.s…"   26 minutes ago       Up 26 minutes       5432/tc
p                                                                                  new-db
9008ad890dcf   postgres                       "docker-entrypoint.s…"   32 minutes ago       Up 32 minutes       5432/tc
p                                                                                  lab5-postgres-1
516a78690496   postgres                       "docker-entrypoint.s…"   34 minutes ago       Up 34 minutes       5432/tc
p                                                                                  crazy_hamilton
e0133d34c7ba   postgres                       "docker-entrypoint.s…"   36 minutes ago       Up 36 minutes       0.0.0.0
:5434->5432/tcp, [::]:5434->5432/tcp                                               recursing_brattain
0d5e5cd2a63e   postgres                       "docker-entrypoint.s…"   39 minutes ago       Up 39 minutes       0.0.0.0
:5433->5432/tcp, [::]:5433->5432/tcp                                               determined_heisenberg
21261dcf42be   postgres                       "docker-entrypoint.s…"   40 minutes ago       Up 40 minutes       0.0.0.0
:5432->5432/tcp, [::]:5432->5432/tcp                                               lucid_dewdney
d7bc2776b97f   postgres                       "docker-entrypoint.s…"   3 weeks ago          Up 31 minutes       5432/tc
p                                                                                  postgres-postgres-1
cf9549de7fd8   docker.gitea.com/gitea:1.24.6  "/usr/bin/entrypoint…"   4 weeks ago          Up About an hour    0.0.0.0
:3000->3000/tcp, [::]:3000->3000/tcp, 0.0.0.0:222->22/tcp, [::]:222->22/tcp   gitea
```

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

		Name	Container ID	Image	Port(s)	Actions
☐	●	nginx	179dfef3fddb	nginx	80:80	
☐	●	web2	2f66b47d6ce3	web		
☐	●	web1	e0a6df65fe27	web		
☐	●	redis	c10bed0969fb	redis		

8. With everything up and running, you can open http://localhost in your browser to see the site. Refresh the page several times to see the host that's handling the request and the total number of requests:

web2: Number of visits is: 7

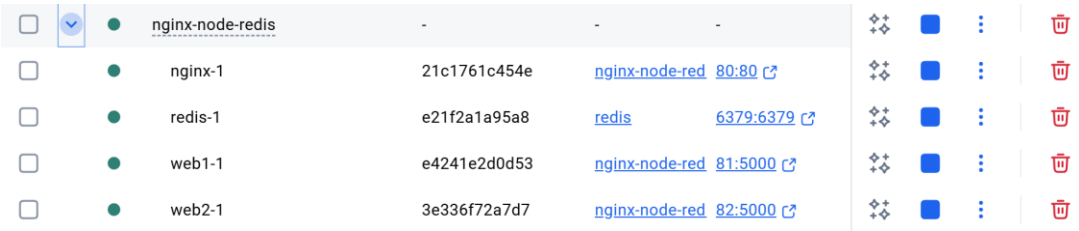
9. You can use the Docker Desktop Dashboard to remove the containers by selecting the containers and selecting the Delete button.

Simplify the deployment using Docker Compose

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

```
Windows PowerShell
=> CACHED [web1 5/5] COPY ./server.js ./                                                                    0.0s
=> [web1] exporting to image                                                                                0.3s
=> => exporting layers                                                                                      0.0s
=> => exporting manifest sha256:e54be521191d35d34d95c9570b8457088c8ac62bf882cad7ba0e0f9211b3ceb2            0.0s
=> => exporting config sha256:cd1336a5ba5f22edb57d6c00b63678977d7516dc64d708aea771afbf76afb8fc              0.0s
=> => exporting attestation manifest sha256:ae2a7000a94f8b75ea6f8910b609a2d852660db0e7926637bf35537a43af7a76  0.1s
=> => exporting manifest list sha256:5aead8ecd756fcdacf3ee48b95ffa4bf326d585e36625467b1937ee60979d3f2      0.0s
=> => naming to docker.io/library/nginx-node-redis-web1:latest                                              0.0s
=> => unpacking to docker.io/library/nginx-node-redis-web1:latest                                           0.0s
=> [web2] exporting to image                                                                                0.3s
=> => exporting layers                                                                                      0.0s
=> => exporting manifest sha256:c58809ed24c93b2591e76a3d889636cbc41e81b4e328c0d745451546808ff14c            0.0s
=> => exporting config sha256:6f9dc3c468fbb7b1c81723d41b281234de9407d2d1c19df39de93566ebcee2d5              0.0s
=> => exporting attestation manifest sha256:edc312228e56556a14df533f28a3bb06083edf5b83e7465d9281f0bcdbbf6293  0.1s
=> => exporting manifest list sha256:717a9e461f168cc0dc50a2fc5920b5c609ab3ebc9d4ba4fda1c580addea4f7f3      0.0s
=> => naming to docker.io/library/nginx-node-redis-web2:latest                                              0.0s
=> => unpacking to docker.io/library/nginx-node-redis-web2:latest                                           0.0s
=> CACHED [nginx 2/3] RUN rm /etc/nginx/conf.d/default.conf                                                 0.0s
=> CACHED [nginx 3/3] COPY nginx.conf /etc/nginx/conf.d/default.conf                                        0.0s
=> [nginx] exporting to image                                                                               0.3s
=> => exporting layers                                                                                      0.0s
=> => exporting manifest sha256:758408f7d49ce7661b1d6367c6634192a890cd0ff75dbf5a87926ec0b51b4268            0.0s
=> => exporting config sha256:aa297f8a4d90802832e91beeb40aa7d8e7185395b245343ddeca5bc8ecc179ff              0.0s
=> => exporting attestation manifest sha256:f97f523089c2121dff1638ca0c9a1d25555fa8cbbf9792054c18e1dac11215d8  0.1s
=> => exporting manifest list sha256:a272e2d7b52aa6b375ae6af36cfa1d311876cf69a16b95d4f3ef176c2b876ed5      0.0s
=> => naming to docker.io/library/nginx-node-redis-nginx:latest                                             0.0s
=> => unpacking to docker.io/library/nginx-node-redis-nginx:latest                                          0.0s
=> [web1] resolving provenance for metadata file                                                            0.1s
=> [web2] resolving provenance for metadata file                                                            0.1s
=> [nginx] resolving provenance for metadata file                                                           0.0s
[+] Running 8/8
 ✔nginx-node-redis-web2              Built                                                                  0.0s
 ✔nginx-node-redis-web1              Built                                                                  0.0s
 ✔nginx-node-redis-nginx             Built                                                                  0.0s
 ✔Network nginx-node-redis_default   Created                                                                0.4s
 ✔Container nginx-node-redis-web2-1  Started                                                                2.1s
 ✔Container nginx-node-redis-web1-1  Started                                                                2.4s
 ✔Container nginx-node-redis-redis-1 Started                                                                1.9s
 ✔Container nginx-node-redis-nginx-1 Started                                                                3.1s
PS C:\study\云计算导论\Labs\Lab5\nginx-node-redis>
```

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



3. Alternatively, you can use the Docker Desktop Dashboard to remove the containers by selecting the application stack and selecting the **Delete** button.