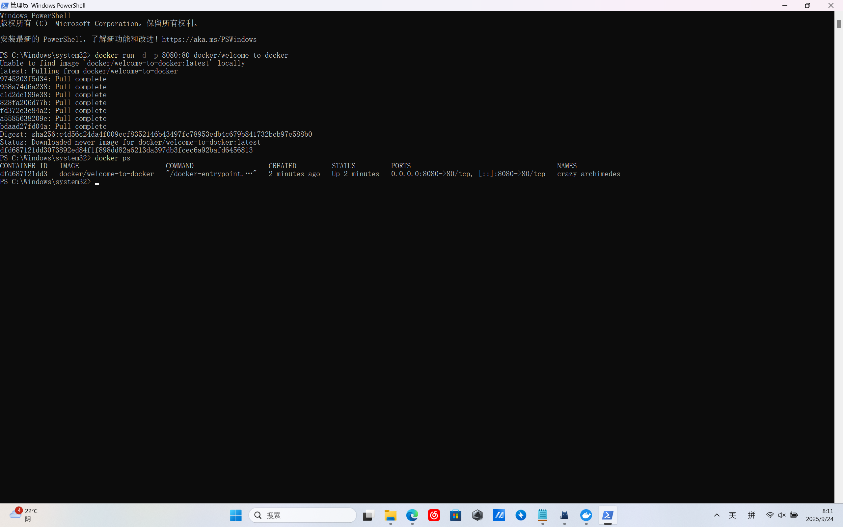
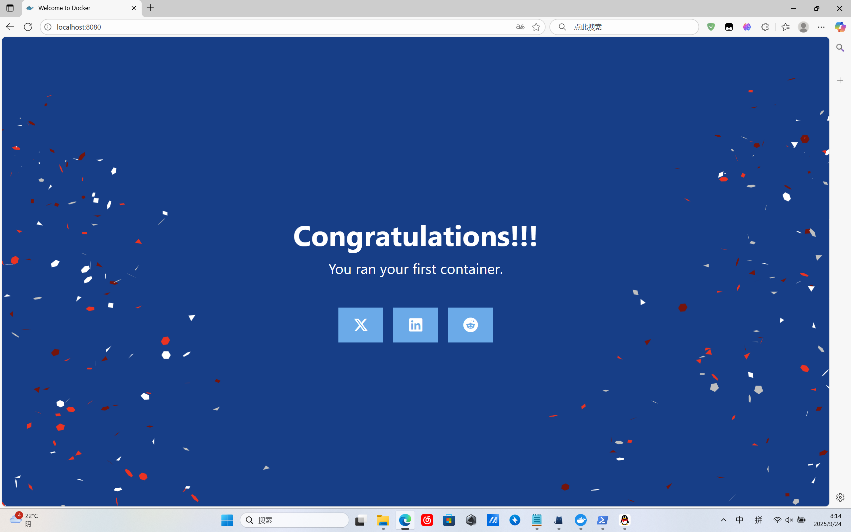
What is a container?

1. Open your CLI terminal and start a container by using the docker run command:
2. verify if the container is up and running by using the [docker ps](https://docs.docker.com/reference/cli/docker/container/ls/) command:

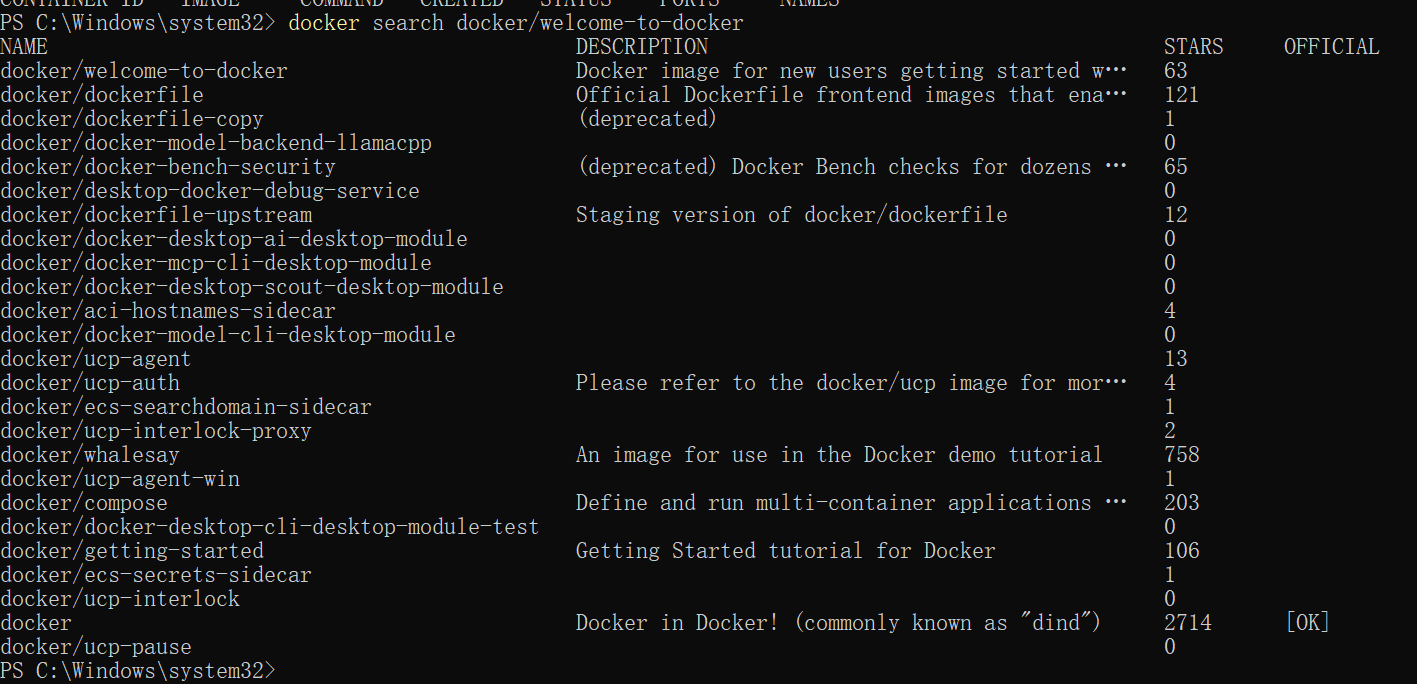


3. visit http://localhost:8080 in browser.



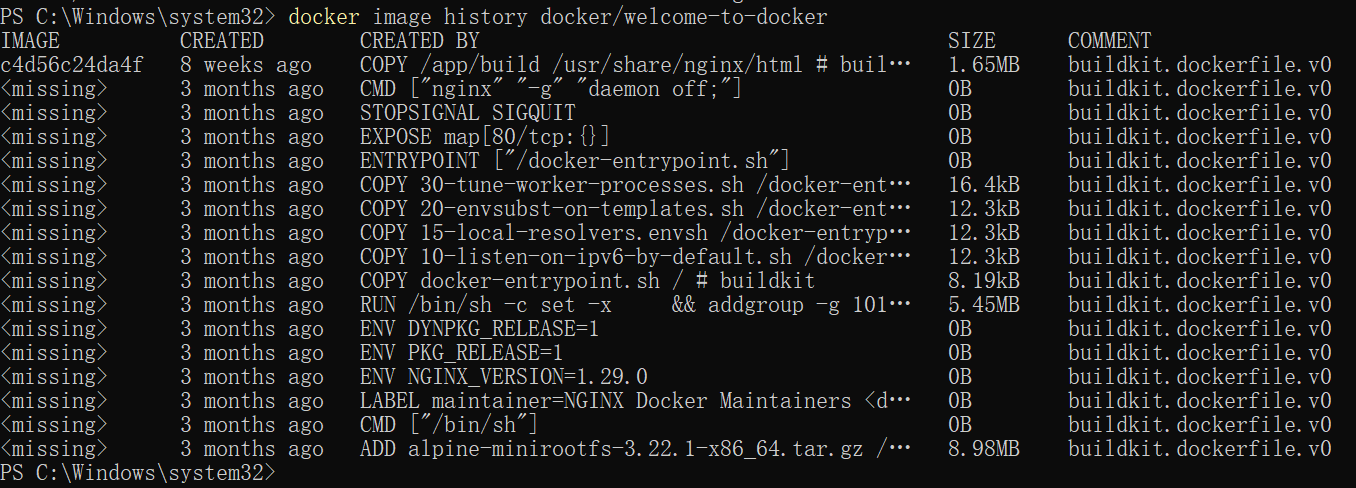
What is an image?

1.Open a terminal and search for images using the docker search command:



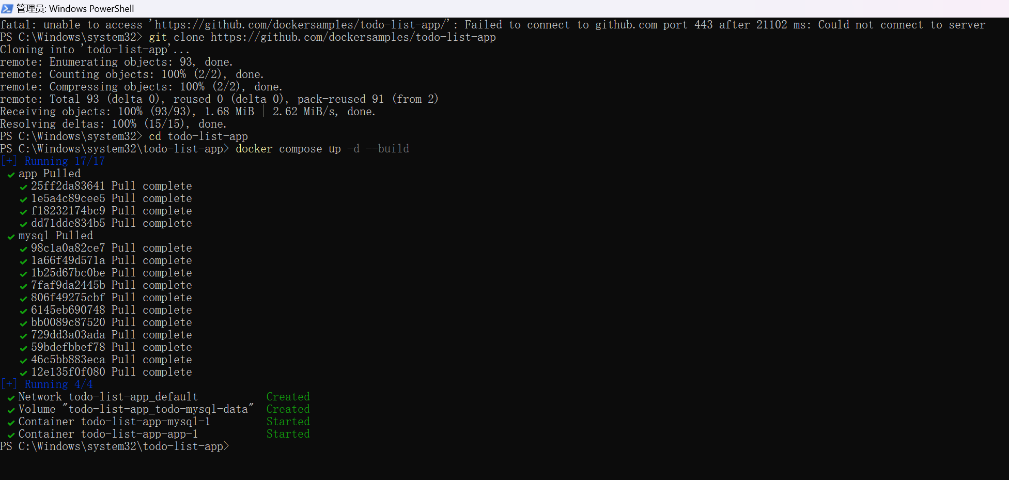
2. Pull the image using the [docker pull](https://docs.docker.com/reference/cli/docker/image/pull/) command.

3. List the image's layers using the docker image history command:



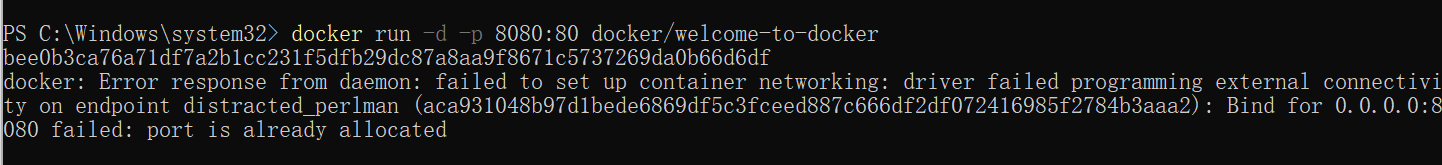
What is Docker Compose?

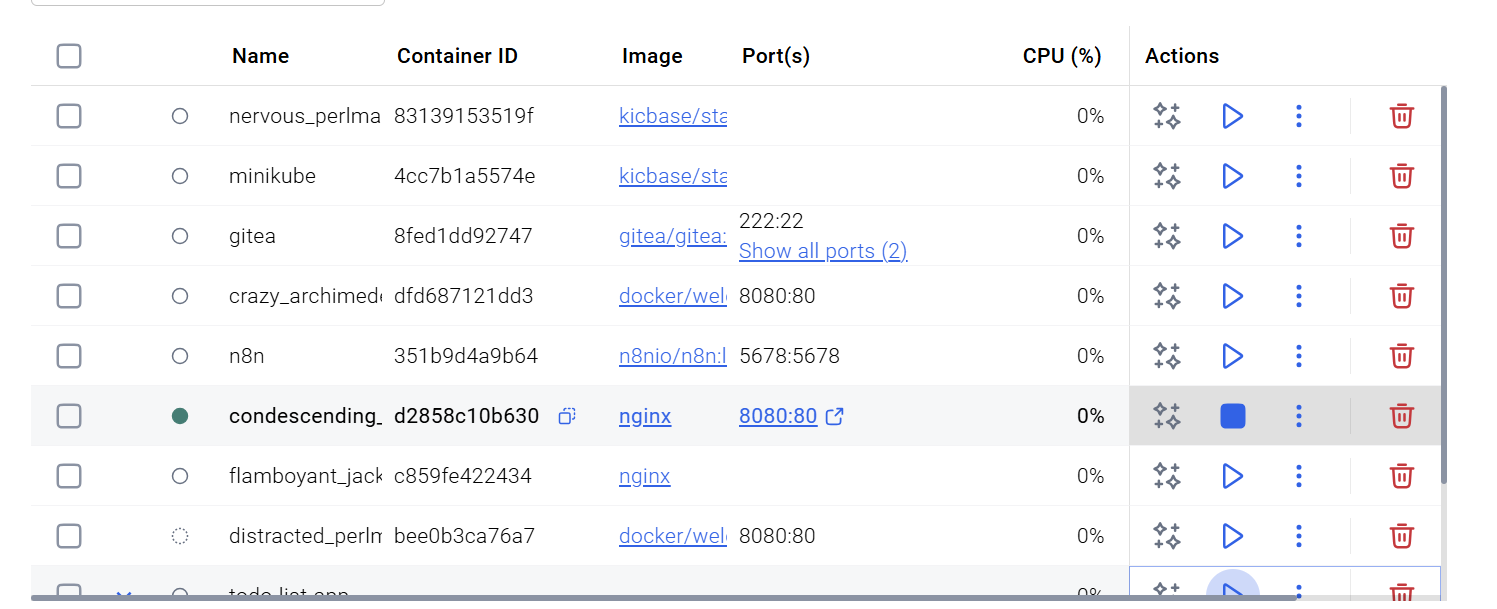
1. Open a terminal and clone this sample application.
2. Navigate into the todo-list-app directory:
3. Use the docker compose up command to start the application:

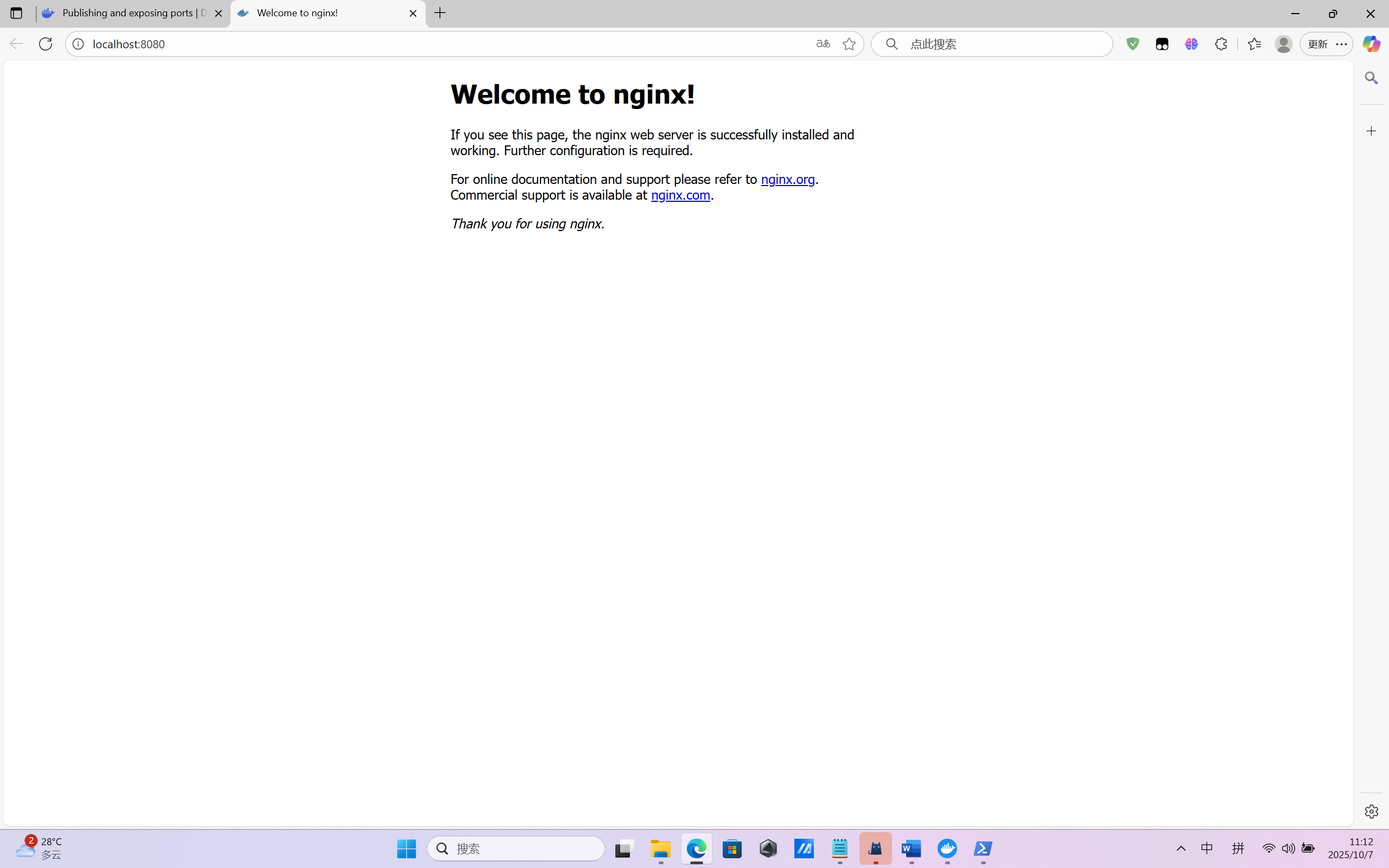


Publishing and exposing ports

1. In a terminal, run the following command to start a new container:

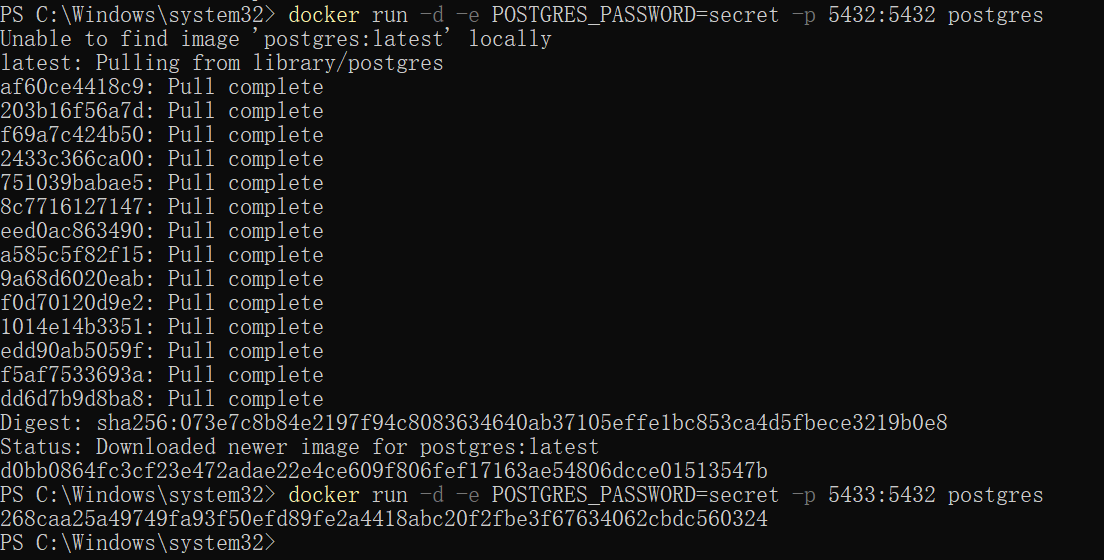




1. 

Overriding container defaults

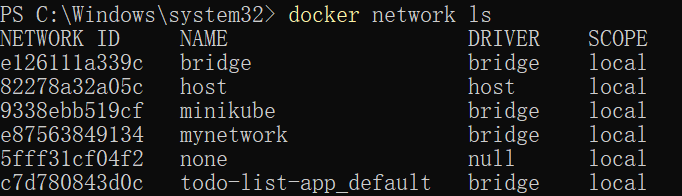
1. Start a container using the Postgres image with the following command:
2. Start a second Postgres container mapped to a different port.



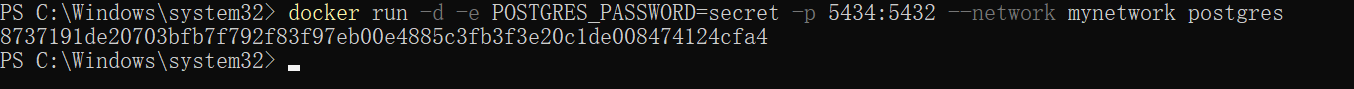
1. Run Postgres container in a controlled network

Create a new custom network by using the following command:

Verify the network by running the following command:

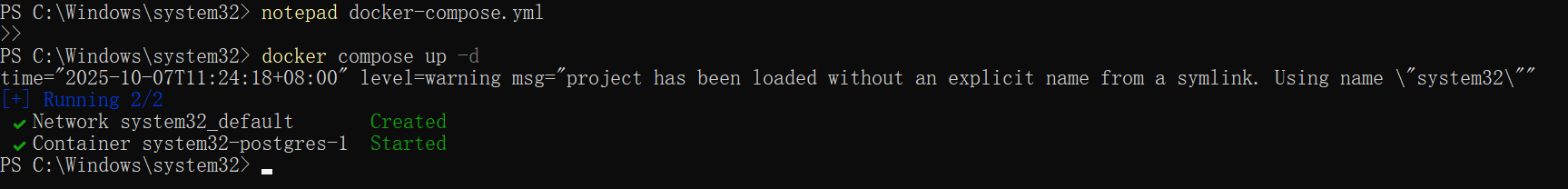


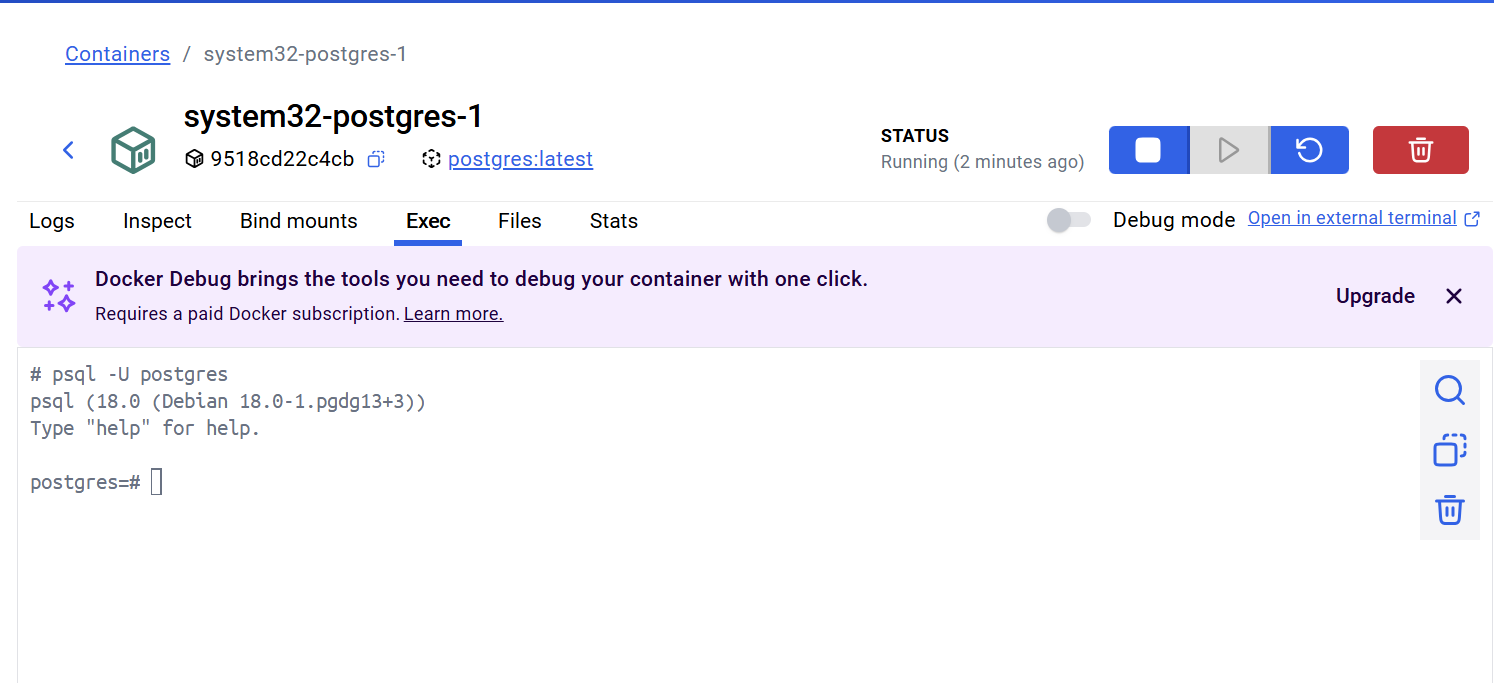
Connect Postgres to the custom network by using the following command:



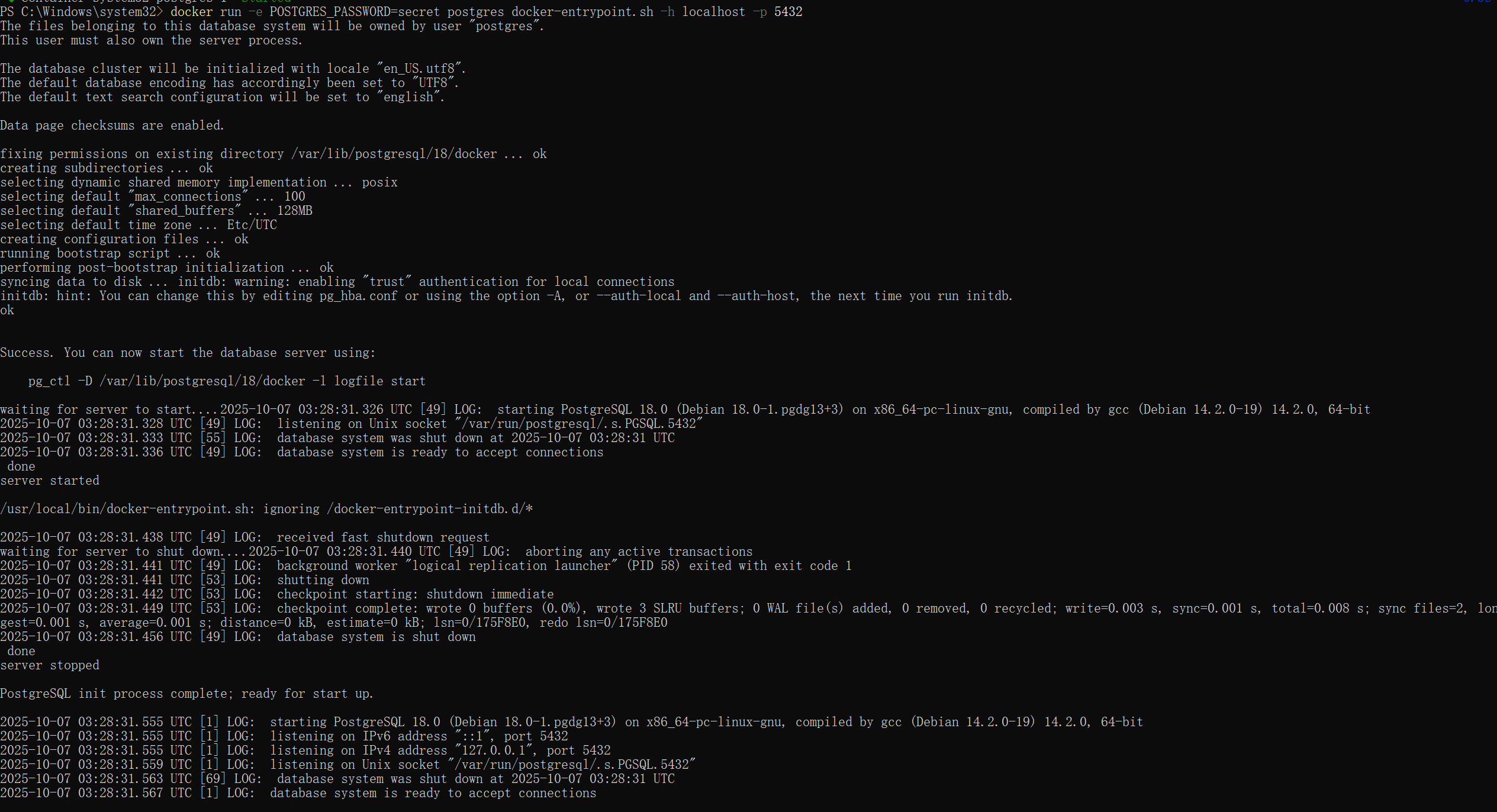
1. Manage the resources

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.





1. Override the default CMD and ENTRYPOINT with docker run

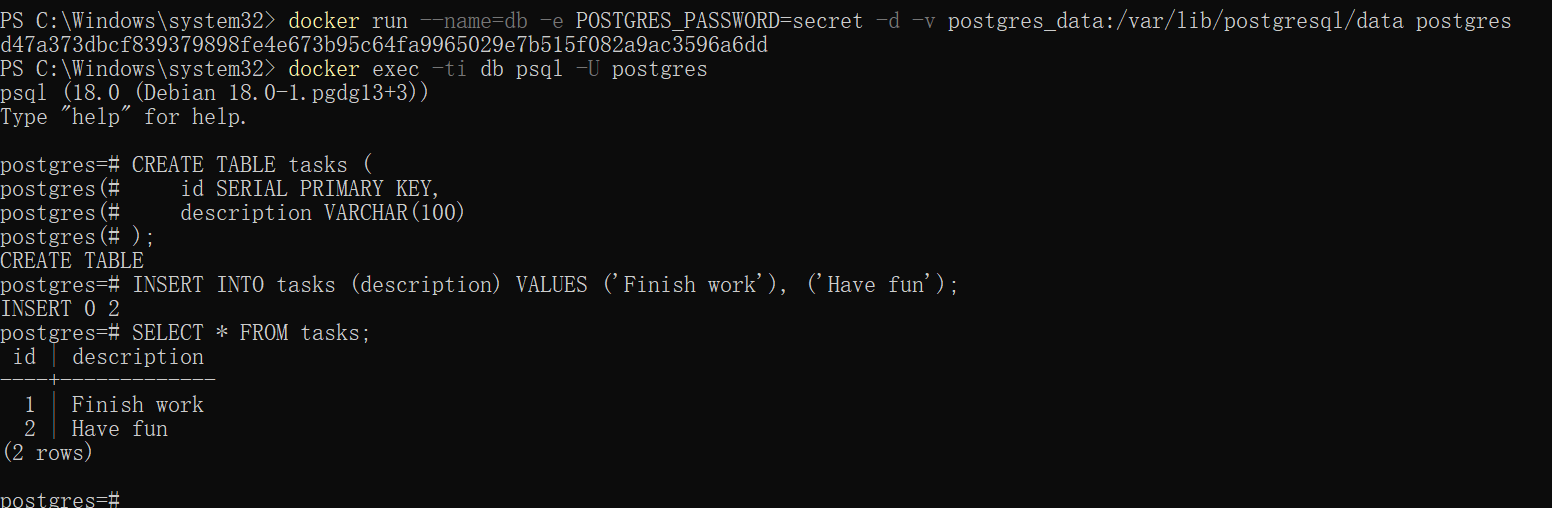


Persisting container data

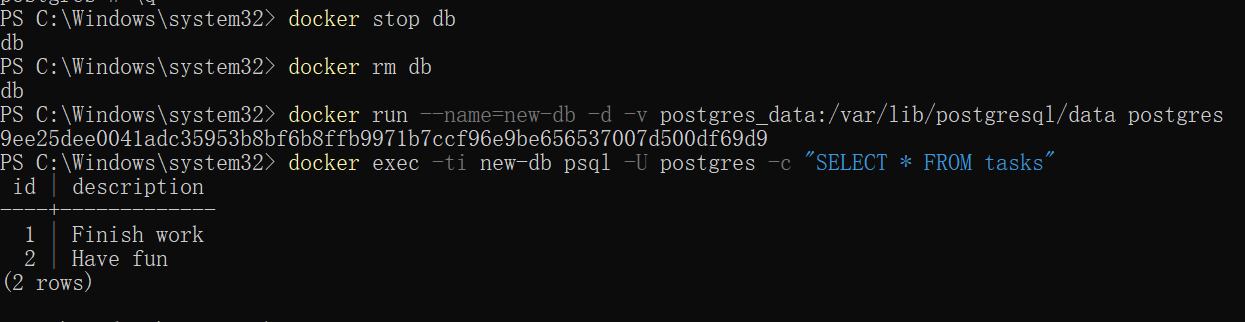
1. Start a container using the Postgres image with the following command:

Connect to the database

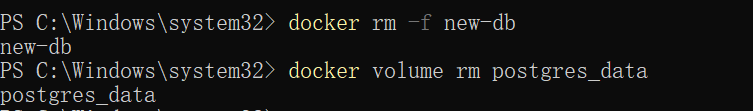
In the PostgreSQL command line, run the following to create a database table and insert two records:



1. Stop and remove the database container. Remember that, even though the container has been deleted, the data is persisted in the postgres\_data volume.
2. Start a new container by running the following command, attaching the same volume with the persisted data:
3. Verify the database still has the records by running the following command:

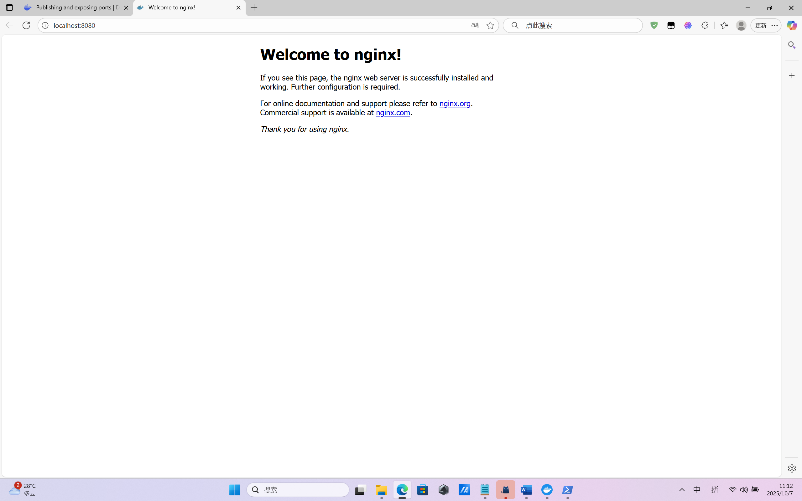


1. Remove volumes

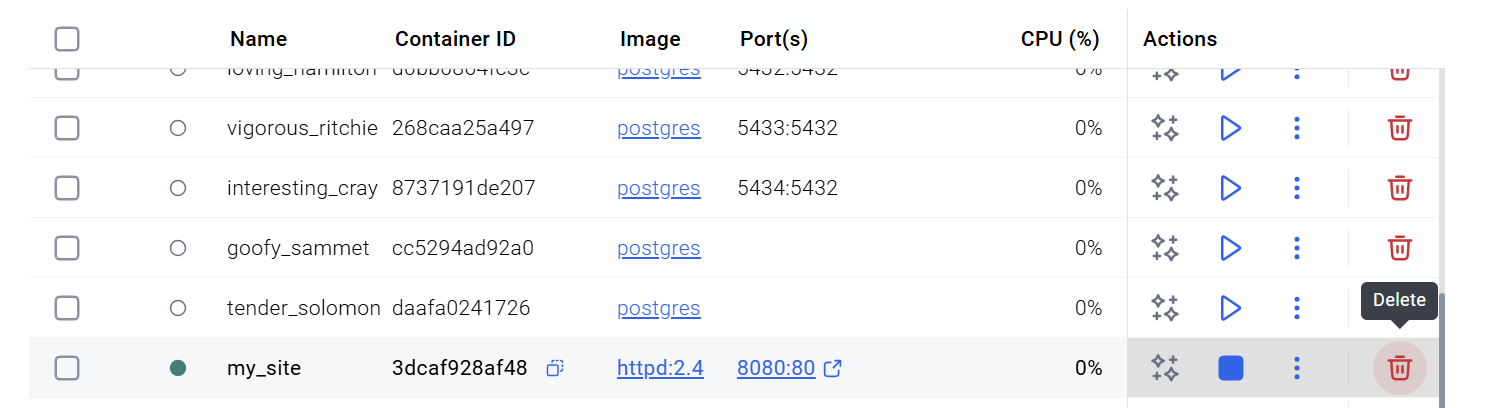


Sharing local files with containers

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



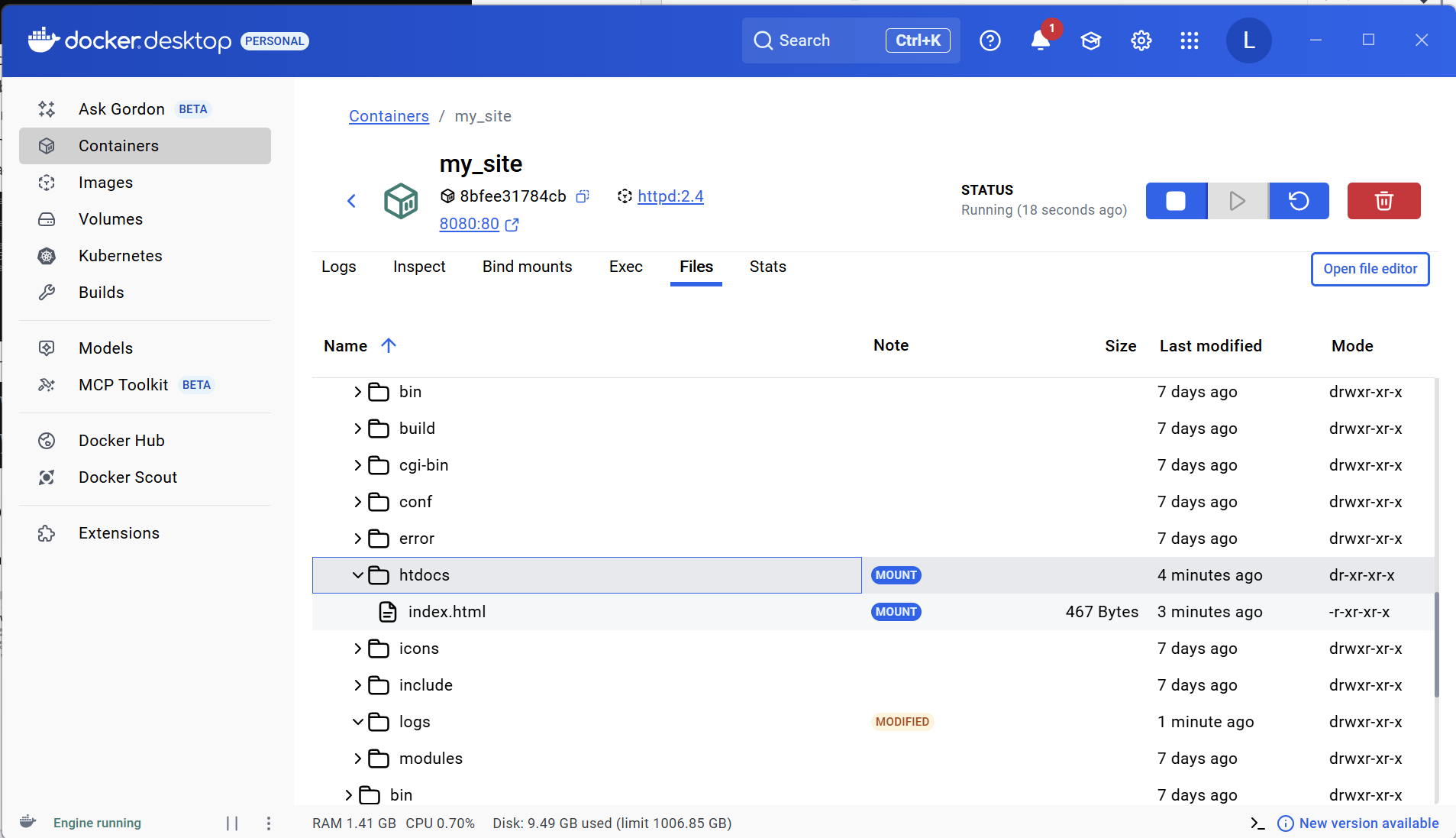
1. Delete the existing container by using the Docker Desktop Dashboard:



Create a new directory called on your host system.public\_html

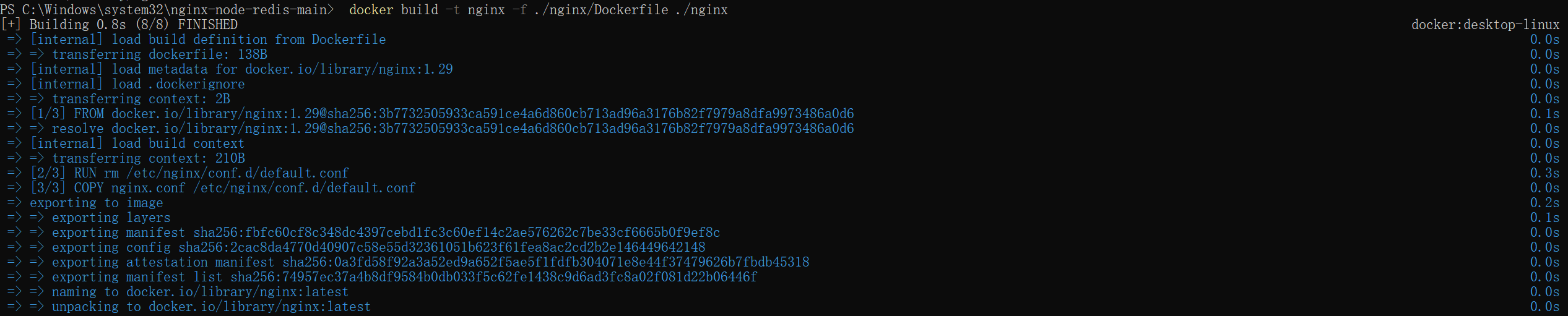
Navigate into the newly created directory and create a file called with the following content. This is a basic HTML document that creates a simple webpage that welcomes you with a friendly whale.

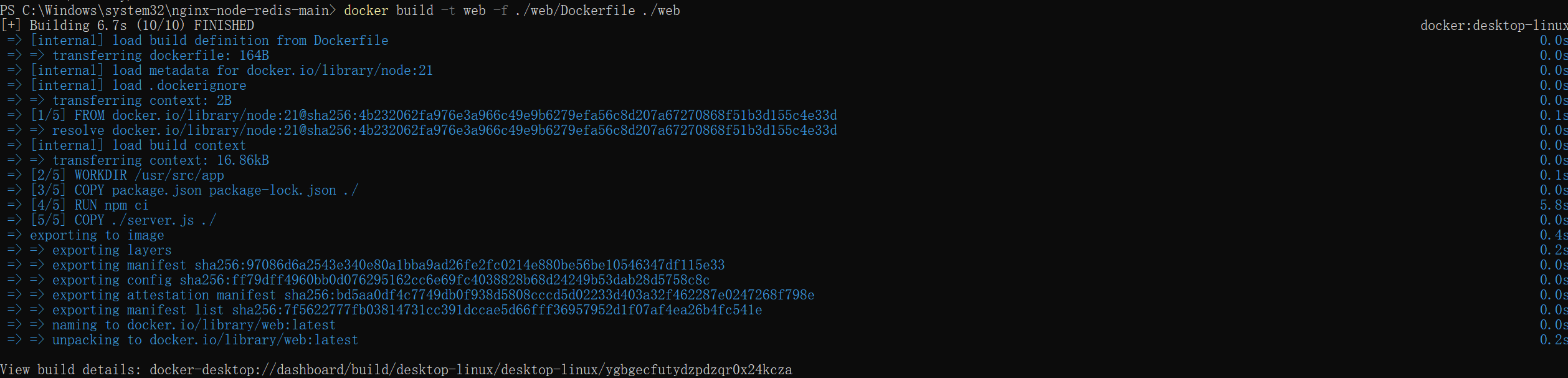
1. run the container



Multi-container applications

1. Use the following command in a terminal to clone the sample application repository.
2. Navigate into the directory to build the image by running the following command:nginx
3. Navigate into the directory and run the following command to build the first web image:web



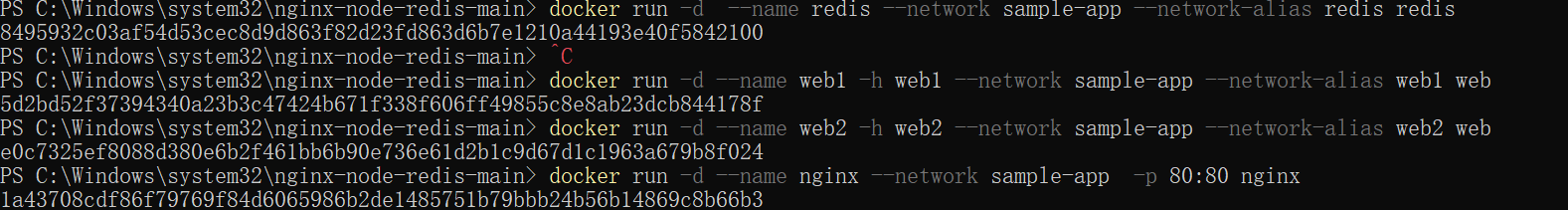


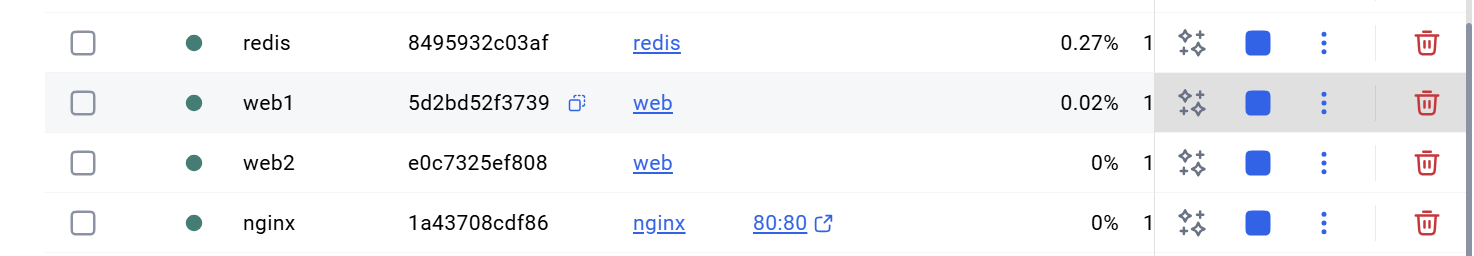
1. Before you can run a multi-container application, you need to create a network for them all to communicate through. You can do so using the command:docker network create
2. Start the Redis container by running the following command, which will attach it to the previously created network and create a network alias (useful for DNS lookups):

Start the first web container by running the following command:

Start the second web container by running the following:

Start the Nginx container by running the following command:



1. 
2. 