Docker Lab 2

Publishing Ports

Publishing a port to 8080:

A screenshot of a computer

AI-generated content may be incorrect.

Evidence of published Port:

A screenshot of a computer

AI-generated content may be incorrect.

How to run all ports:



How to allow docker to choose an ephemeral port:

Docker choses the host port for you instead of manually inputting one





# Overriding Default Configuration for containers

How to set environment variables(single):  


How to set environment variables using a .env file:



Restricting system resources manually:



How to monitor real time stats of docker containers running:

A screen shot of a black screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

To run multiple instances of the same container, you need to set the ports to connect to different host ports, eg a mysql or postgres container can be linked to 111 and 112 on the host machine but 113 on the container side.

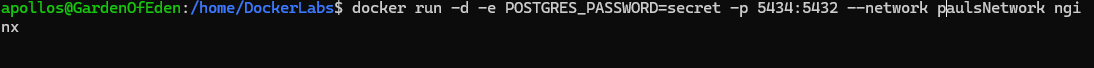
## Network Creation and Controlled Networks

Create a network:

A screenshot of a computer

AI-generated content may be incorrect.

Run a container using said network:



How to set a host manually and an entrypoint script:



## Creating Volumes and Persisting container data:

*docker volume create* is the command to create volumes, eg:



To attach a volume when running a container. Use the -v flag as follows:

.If the volume isn’t creating, it will automatically be created for you.



### Volume Commands:

*docker volume ls* - list all volumes

*docker volume rm <volume-name-or-id>* - remove a volume (only works when the volume is not attached to any containers)

*docker volume prune* - remove all unused (unattached) volumes

docker rm -f <vol> - remove specific volumes

## Sharing Local Files

Using Docker CLI

* Share files using -v or --mount flags.
* Example with -v:

docker run -v /HOST/PATH:/CONTAINER/PATH -it nginx

* Example with --mount:

docker run --mount type=bind,source=/HOST/PATH,target=/CONTAINER/PATH nginx

**File Permissions**

* Use :ro (read-only) or :rw (read-write) to set access permissions.
* Changes in read-write mode reflect on both host and container.

**Synchronized File Shares**

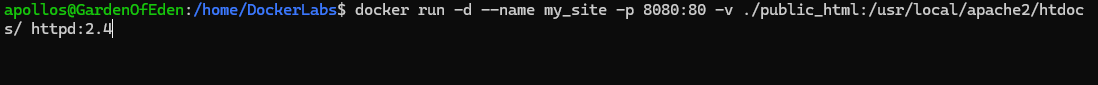
* Improve performance for large codebases by using synchronized filesystem caches.

**Hands-on Guide**

* **Run a Container**: Start an httpd container.



* **Create a Directory**: Make a public\_html directory and add an index.html file.
* **Use a Bind Mount**: Map the local directory to a container directory.
* **Access Files**: View and edit files using Docker Desktop.



**Stopping the Container**

* Use Docker Desktop to stop and delete the container.

## Multi-container Apps

This is a tutorial on how to use docker Compose to simplify setup.

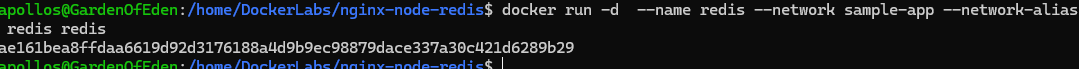
I followed the hands on guide within the page to create a counter web application based on Node.js, a Nginx reverse proxy and a redis database:

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

 A screenshot of a computer program

AI-generated content may be incorrect. A screenshot of a computer screen

AI-generated content may be incorrect.

A blue flag with a white background

AI-generated content may be incorrect.

Docker compose makes this much faster using this one command:  
A screenshot of a computer screen

AI-generated content may be incorrect.

A black screen with blue text

AI-generated content may be incorrect.

Containers are structured under the one project:

A screenshot of a computer

AI-generated content may be incorrect.