

Introduction to Docker

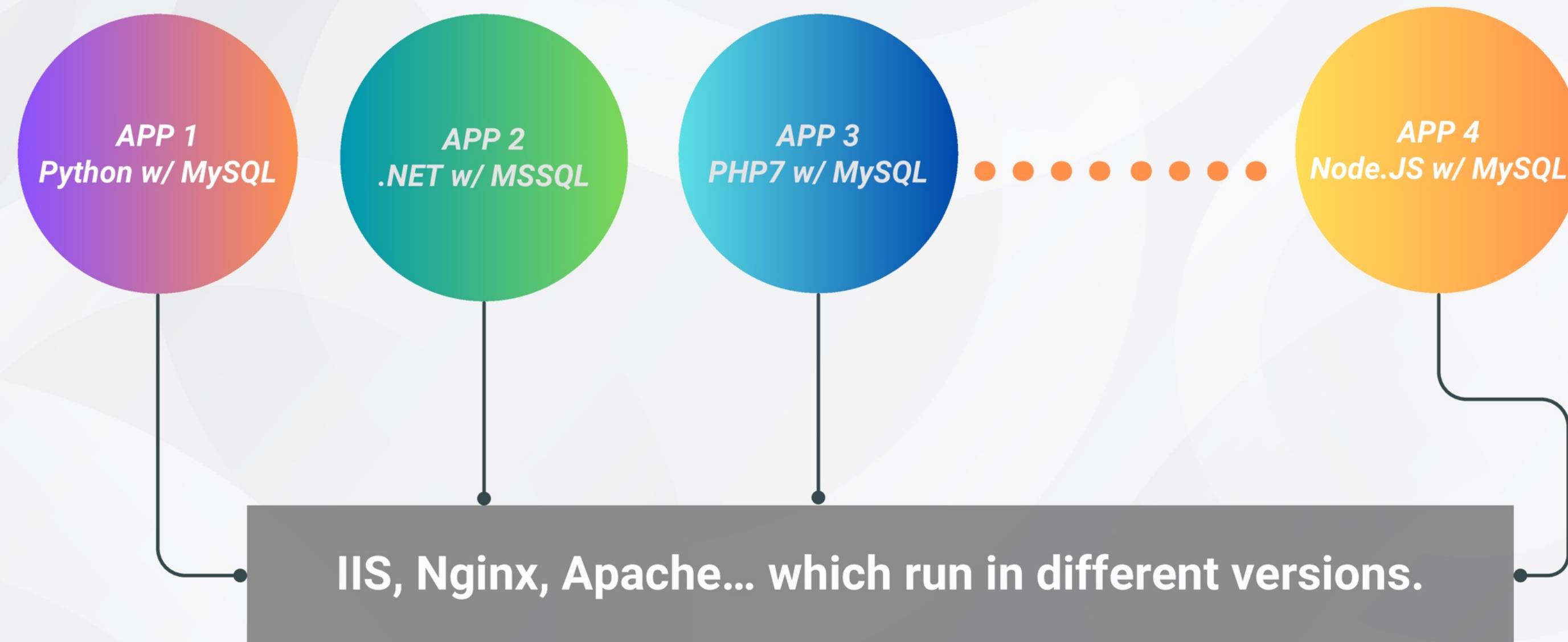
Learning Objectives



- Understand microservices and their advantages over monolithic architectures.
- Learn container technology with Docker and install Docker with prerequisites.
- Run web services in Docker and master common Docker commands.
- Explore docker-compose for multi-container setups and learn to build Docker images.
- Apply basic security practices for project hardening and update containers.

Web Application Development in The Previous Era

Monolith

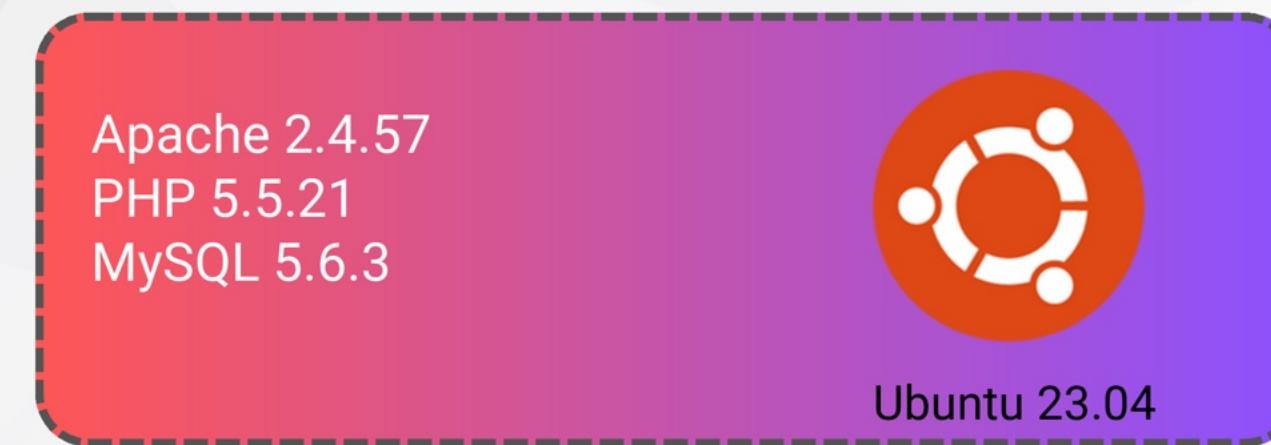


Pain Point



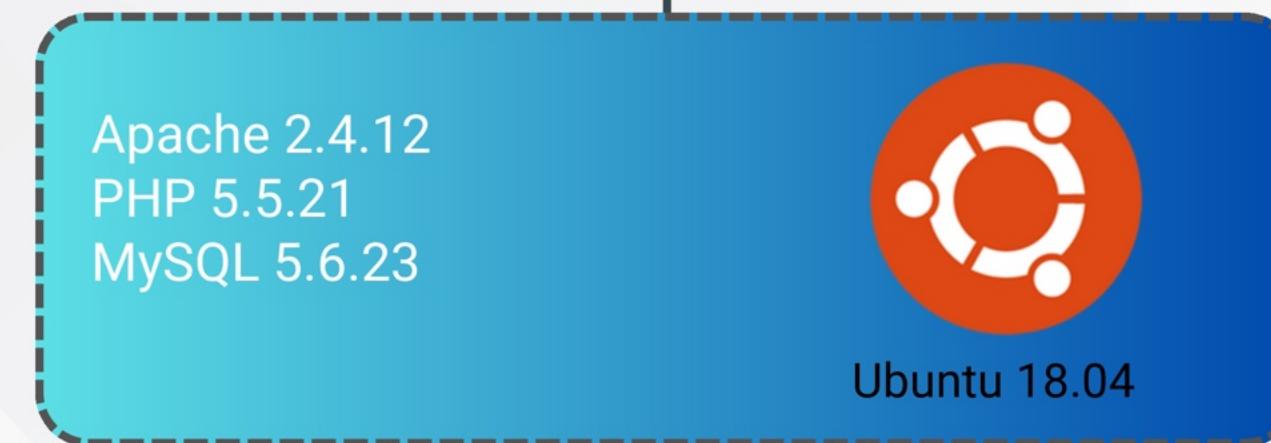
Which when it's time to deliver

Production Server



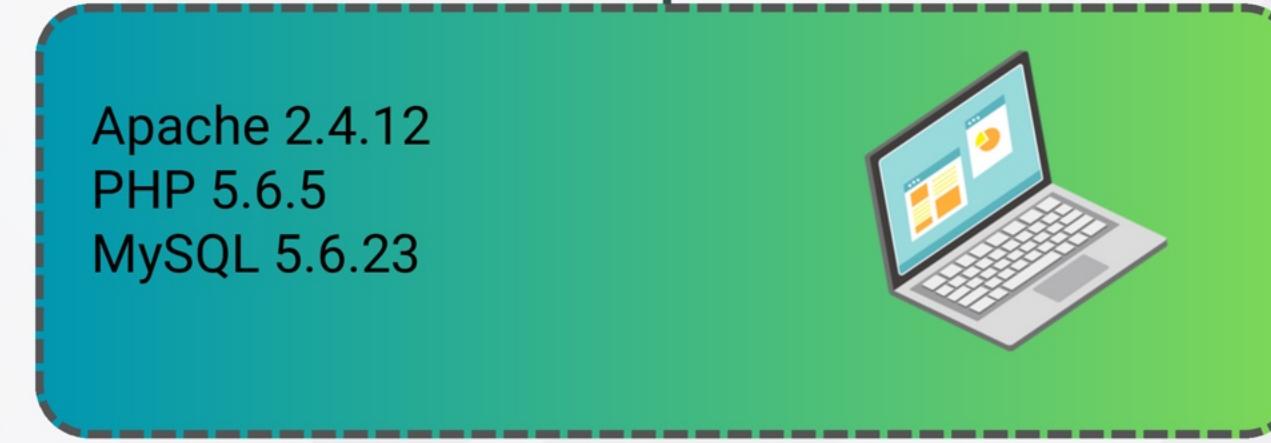
FAILED

Staging Server



WORK

Local Machine



WORK

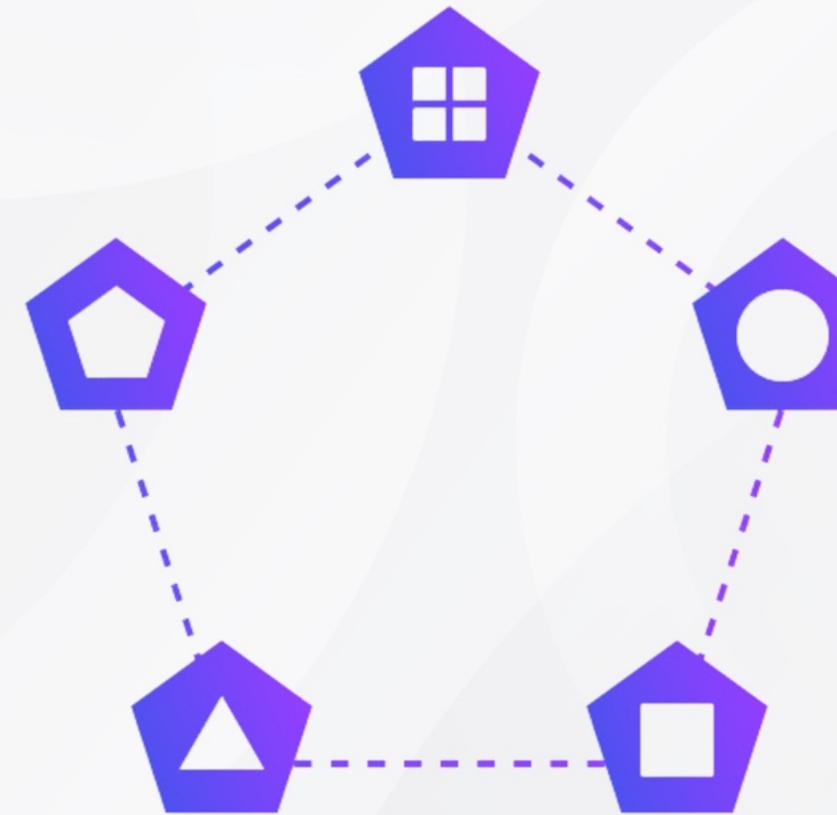
Solving

Solving with Microservices

Monolith



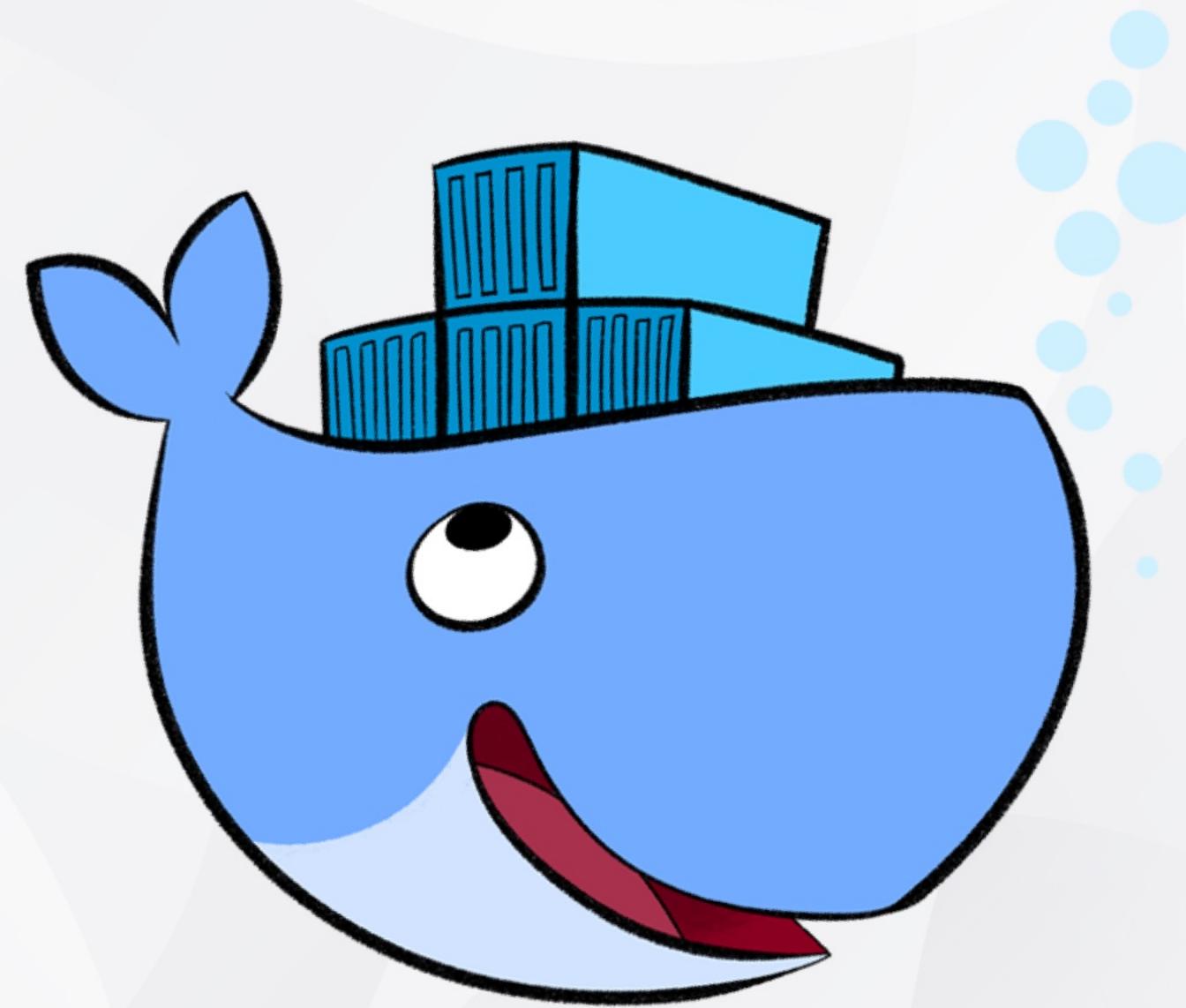
Microservices



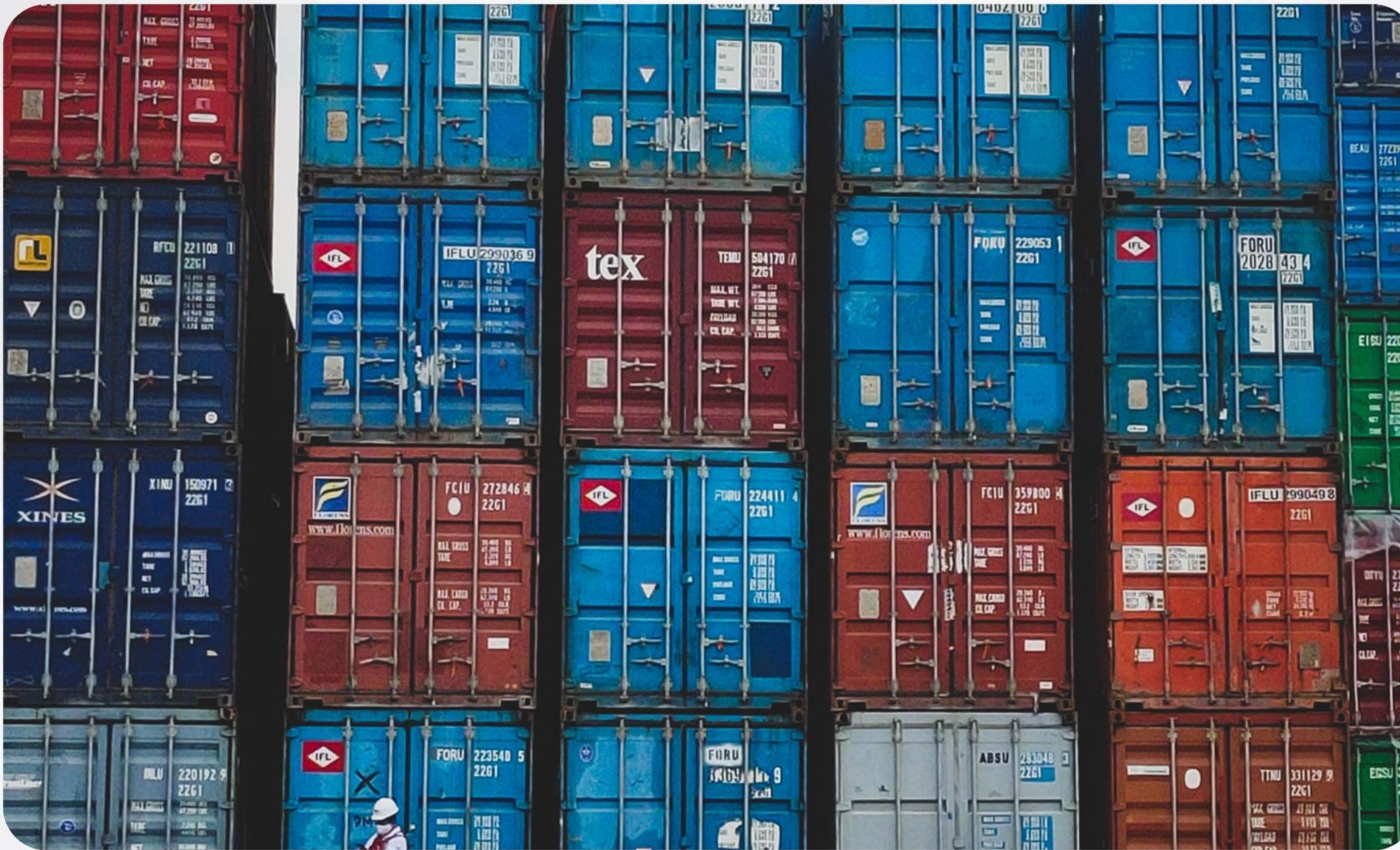
Make every system have the same environment



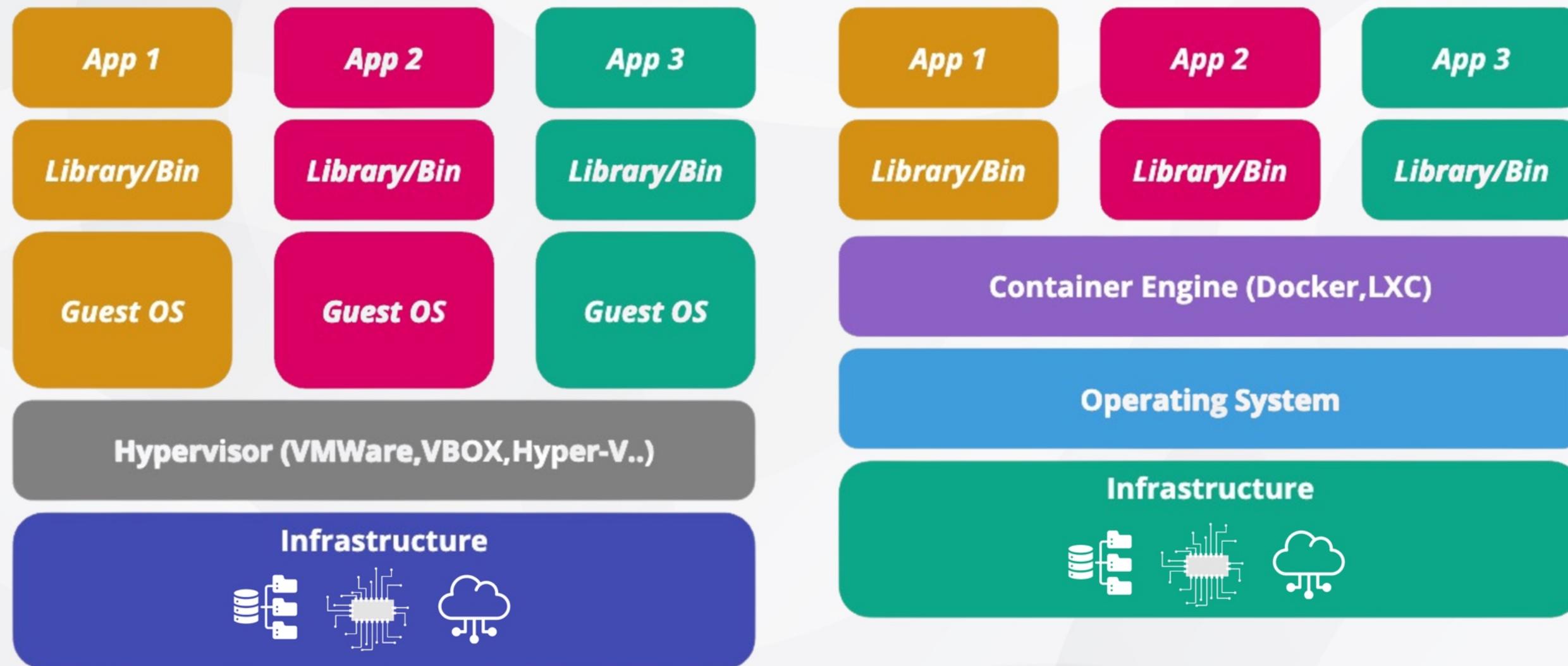
Docker is Microservices Tool



Docker is use Container Technology



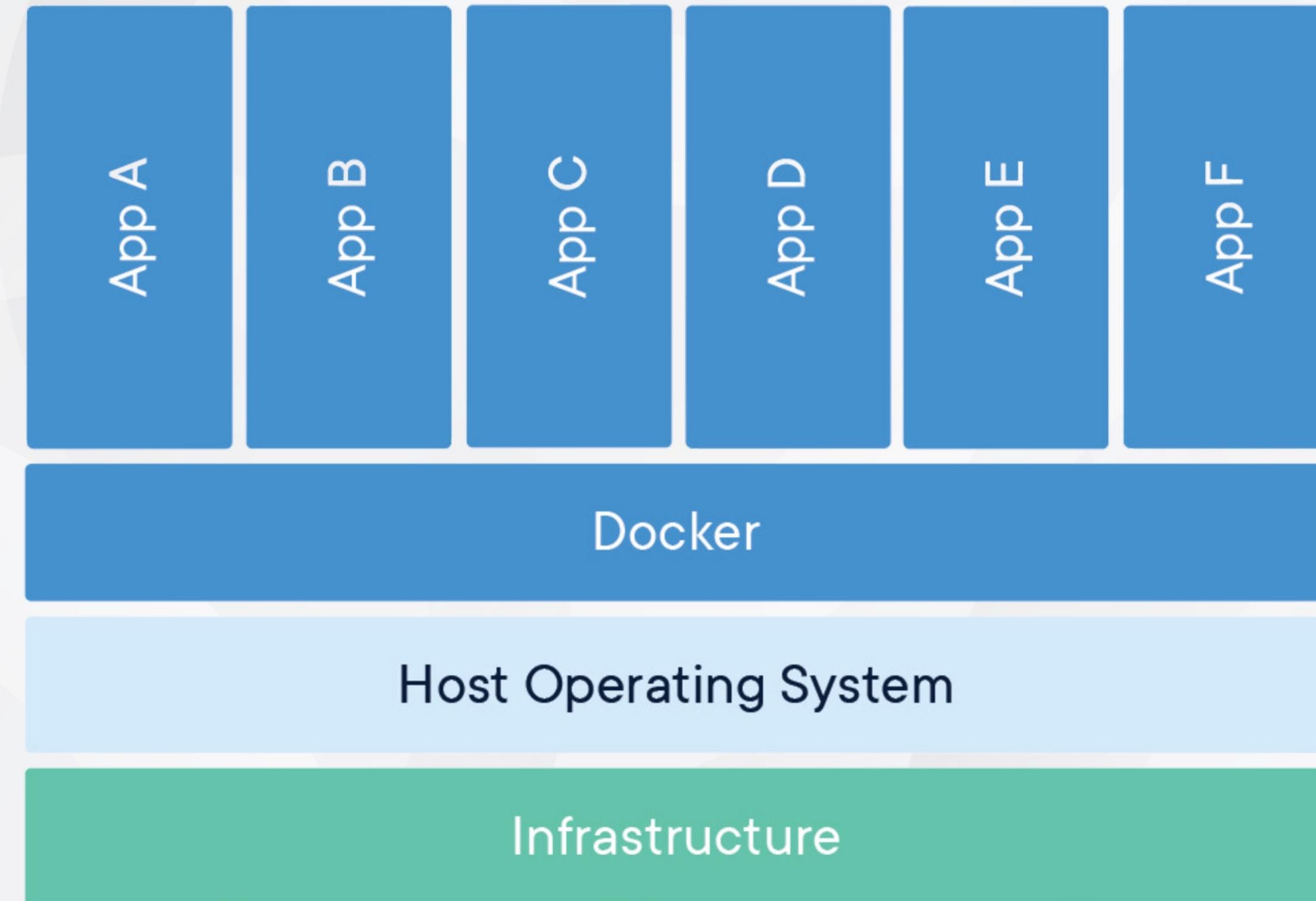
Virtual Machine vs Container



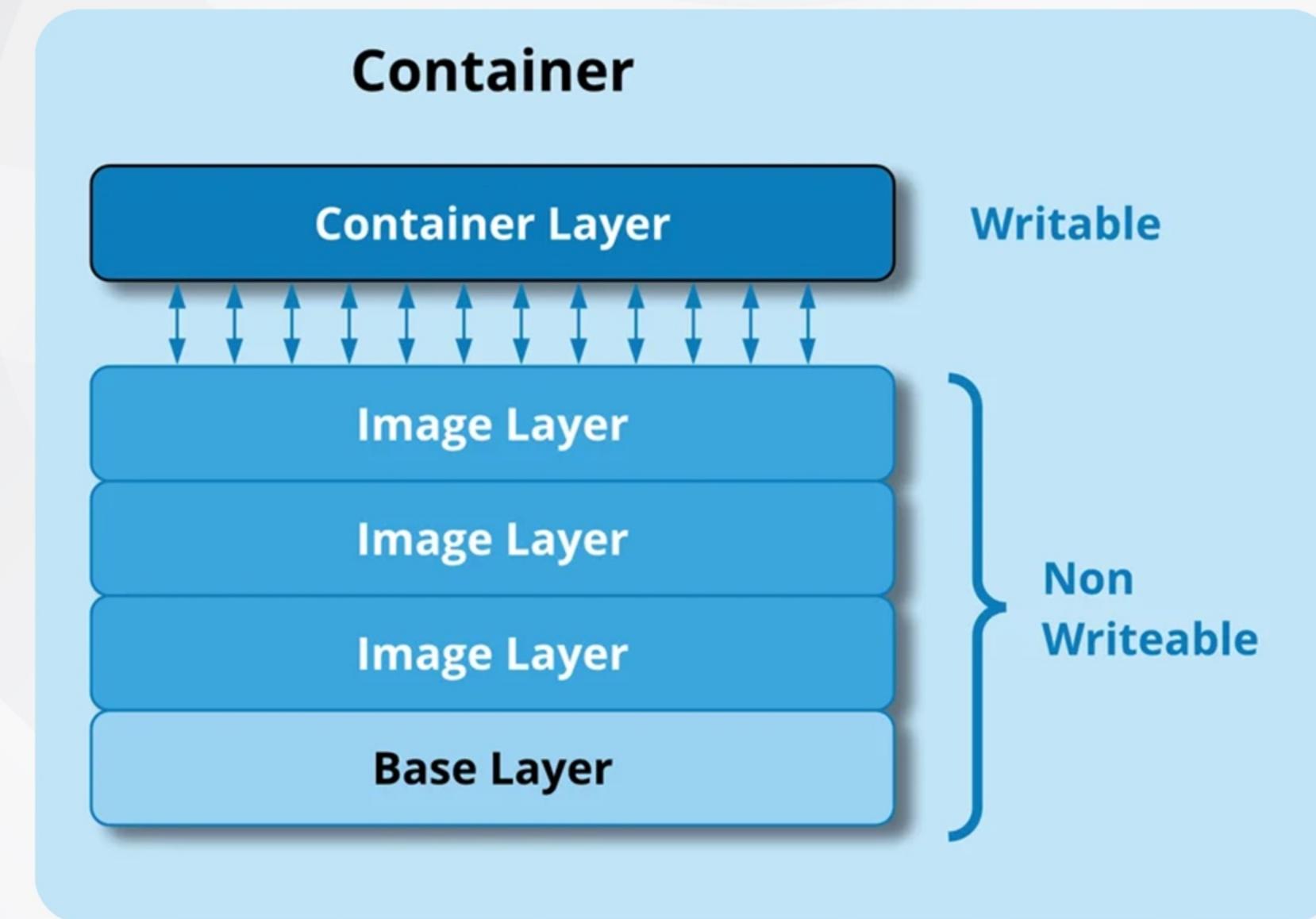
Virtual Machine

Containers

Docker



Docker Image



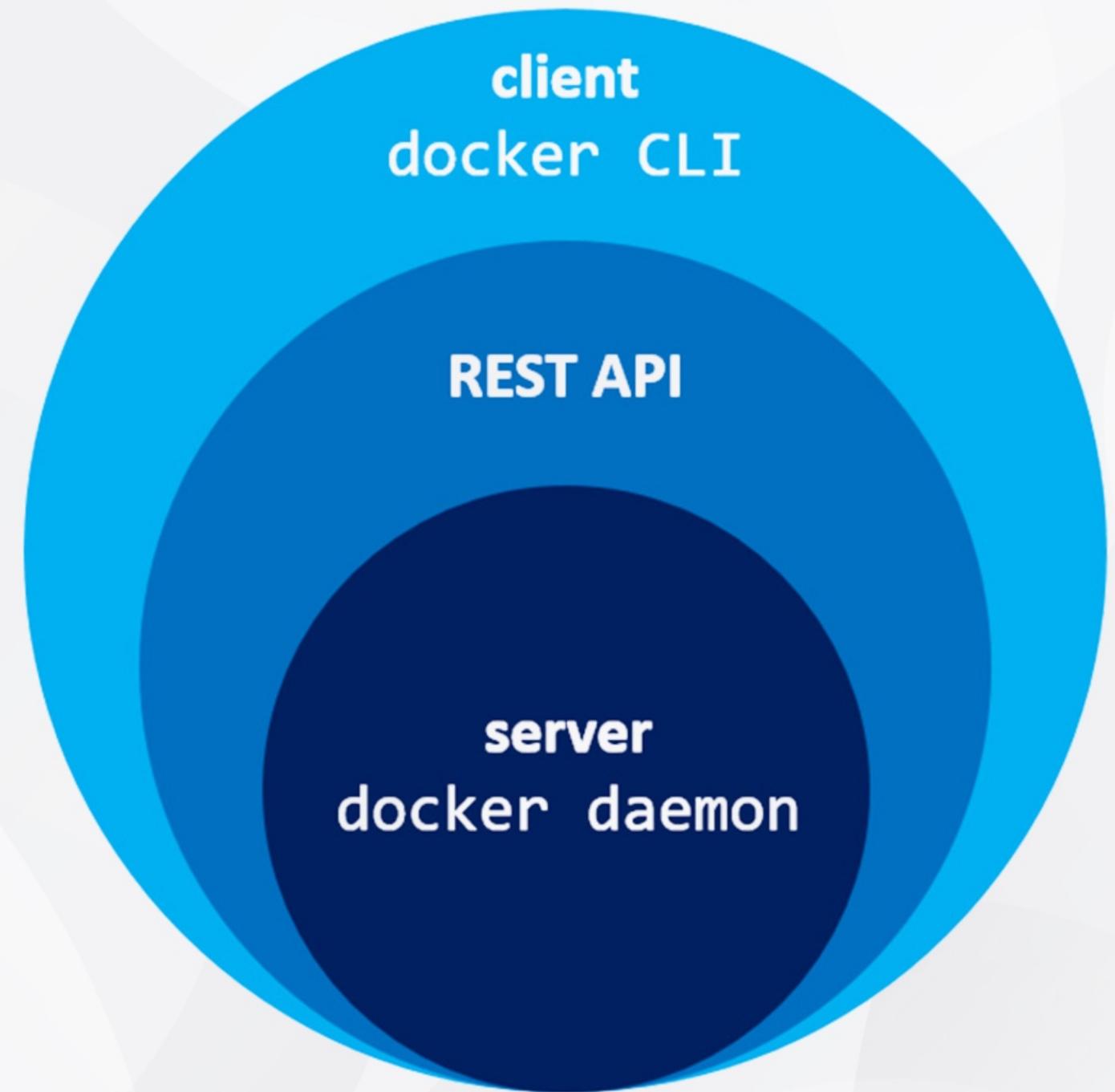
“**Image** like a **class** **Container** is a instance of **class**”

Community Edition & Enterprise Edition



How docker work?

Docker Components

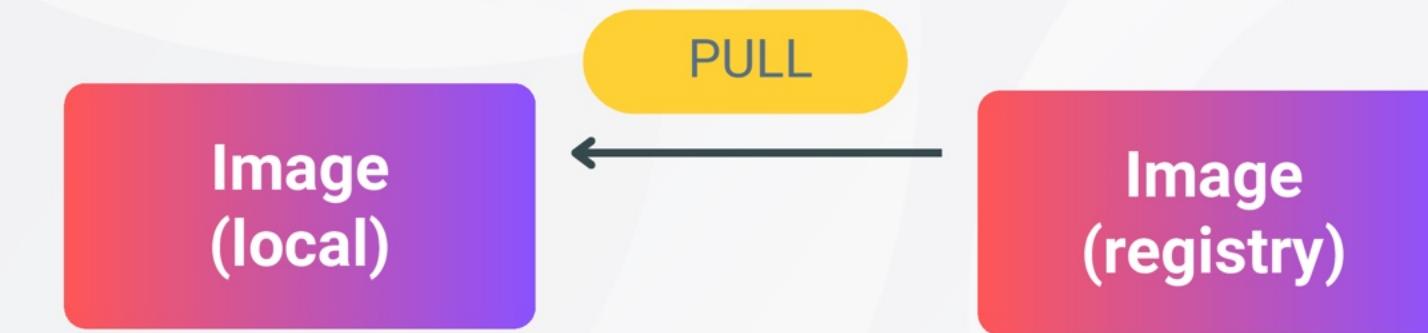


Docker Registry Architecture

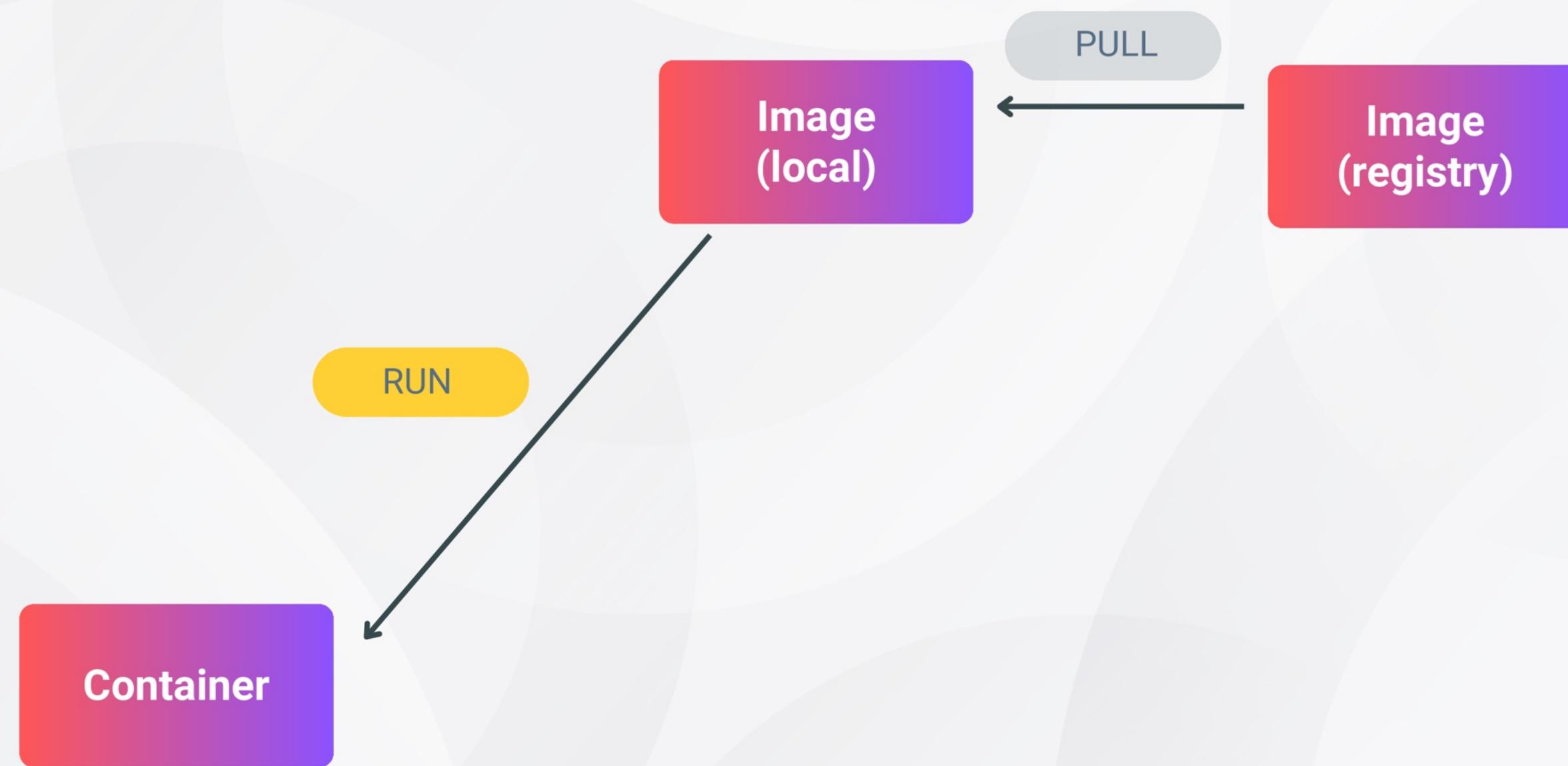
Relationship between Image and Container

Image
(registry)

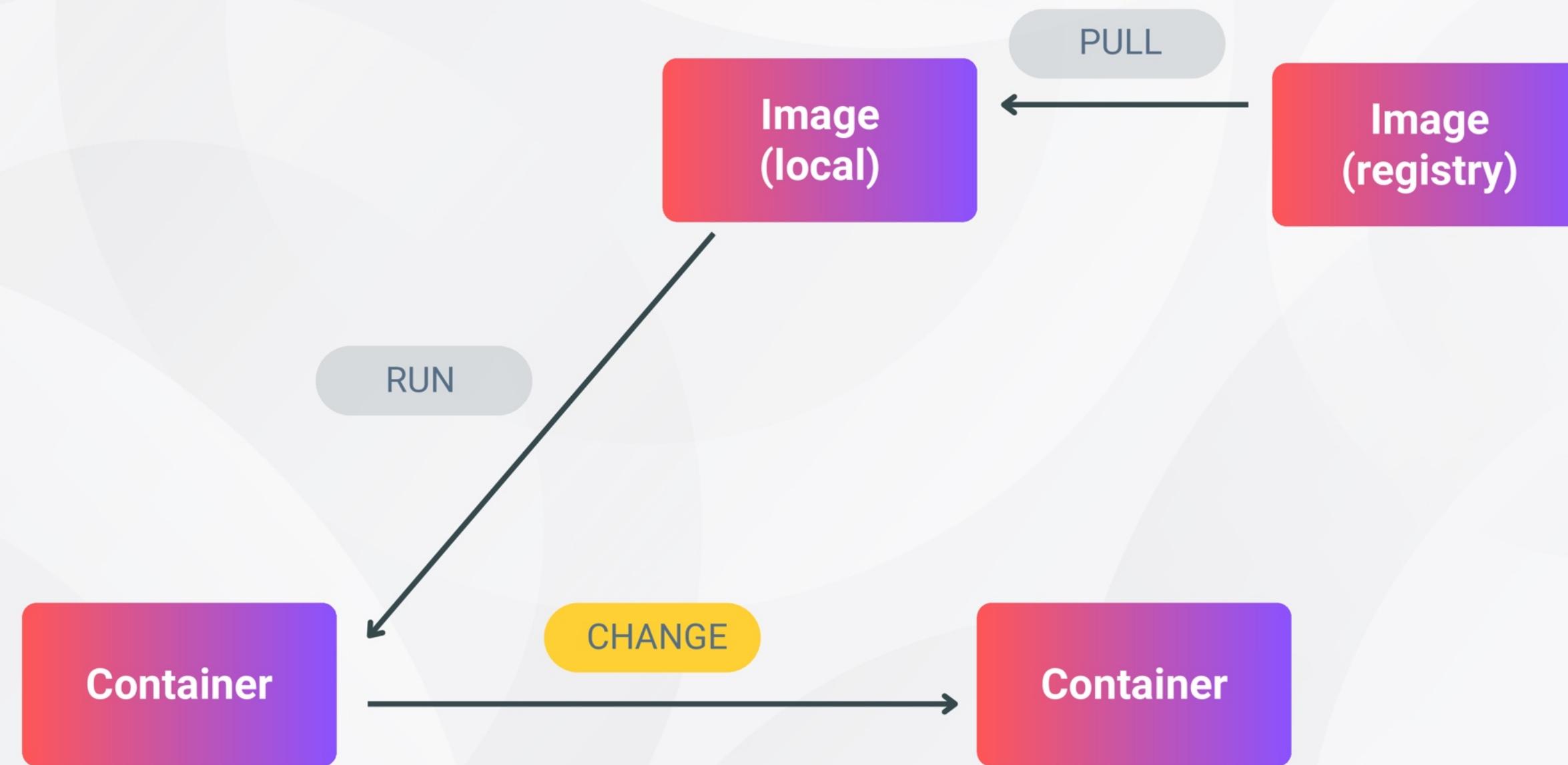
Relationship between Image and Container



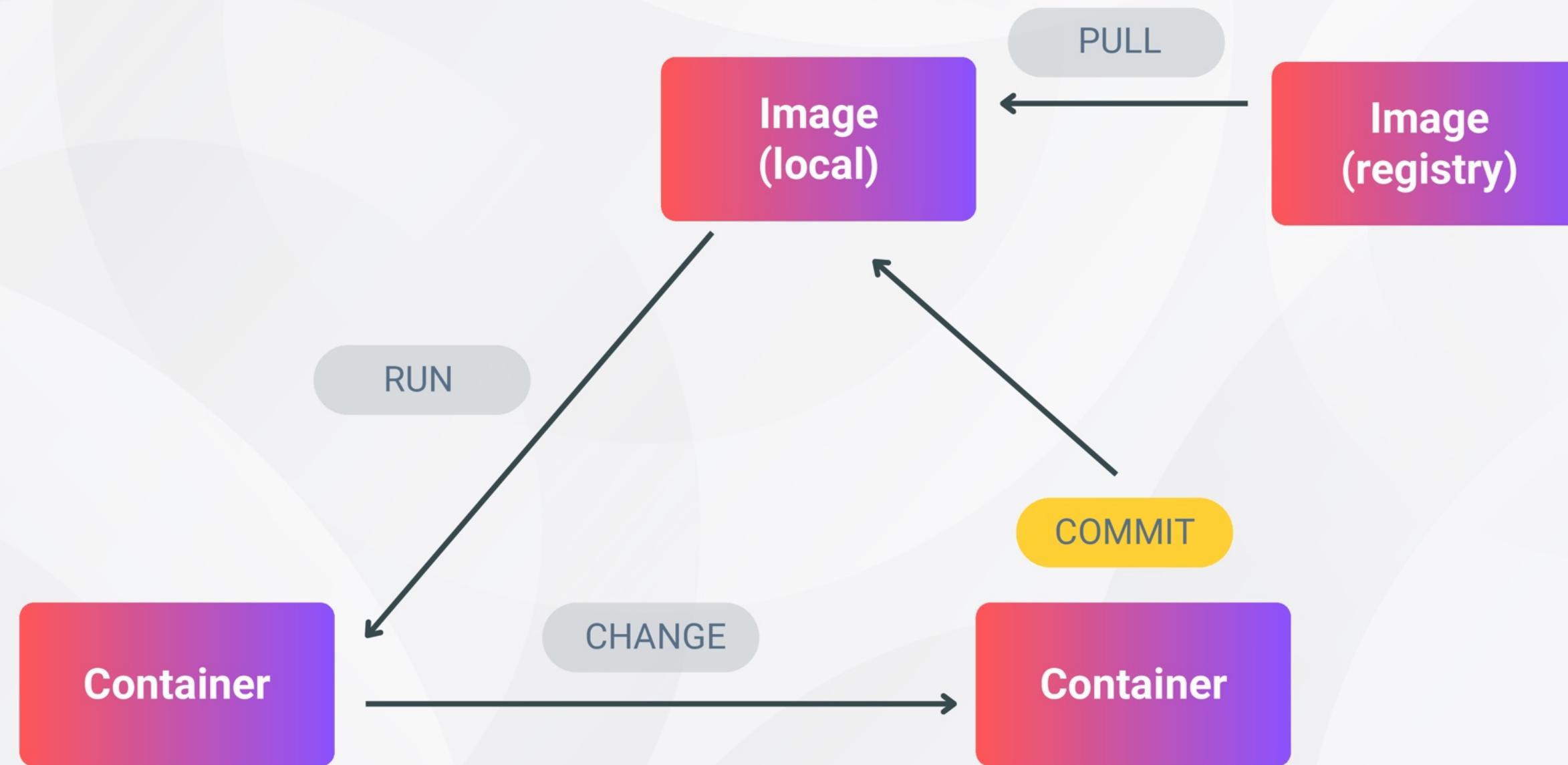
Relationship between Image and Container



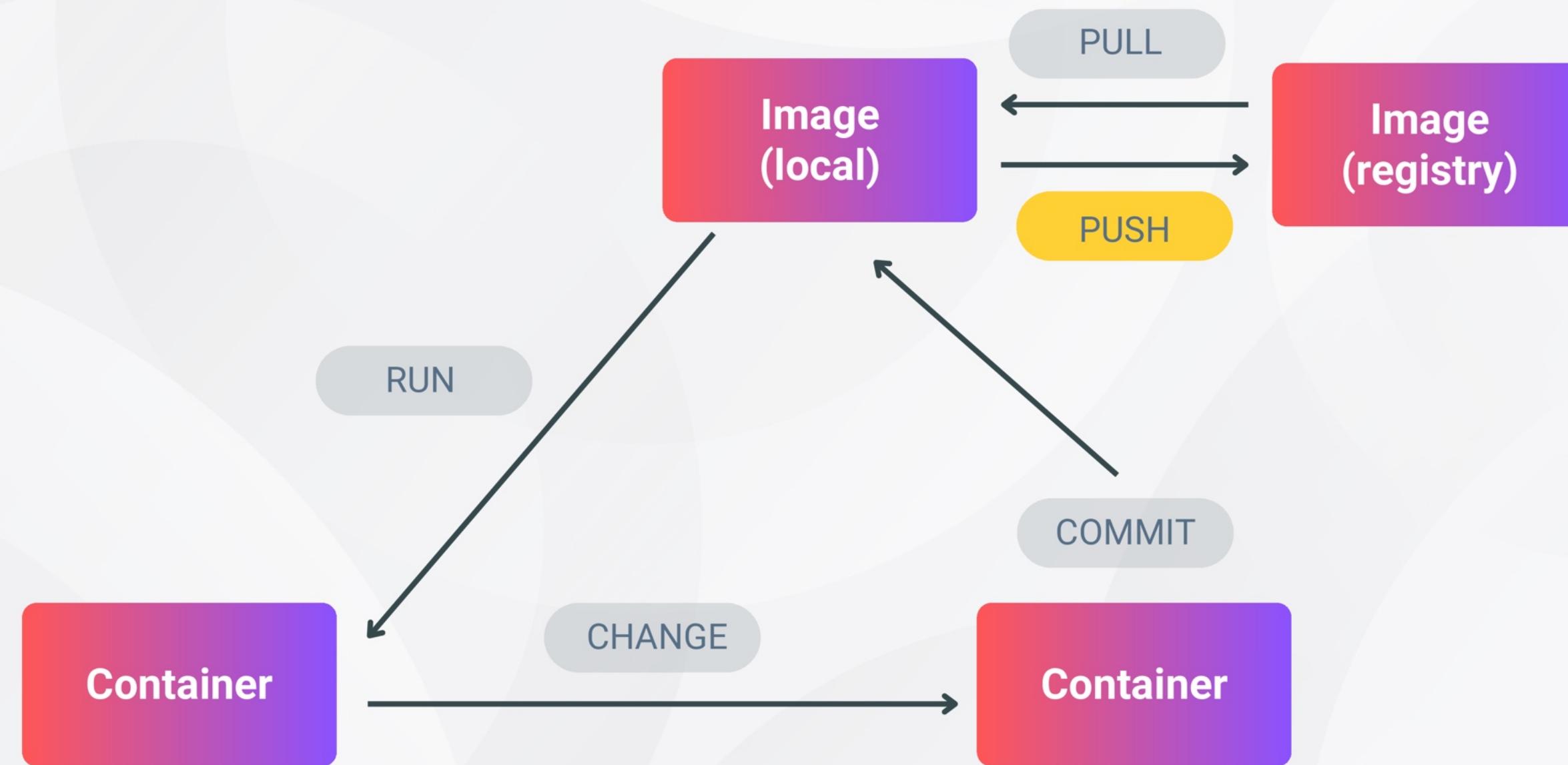
Relationship between Image and Container



Relationship between Image and Container



Relationship between Image and Container



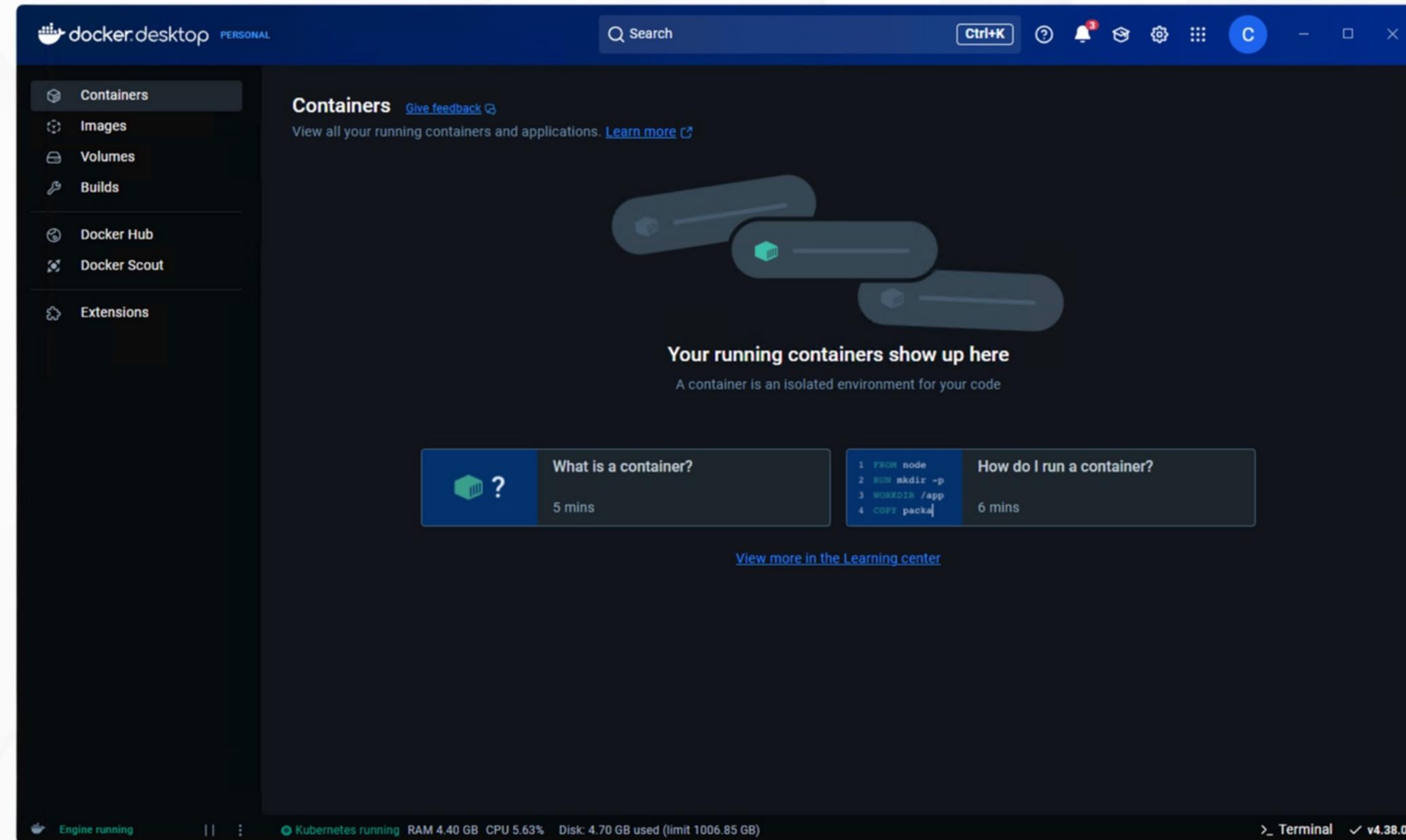
How can we use Docker?

Download

<https://www.docker.com>

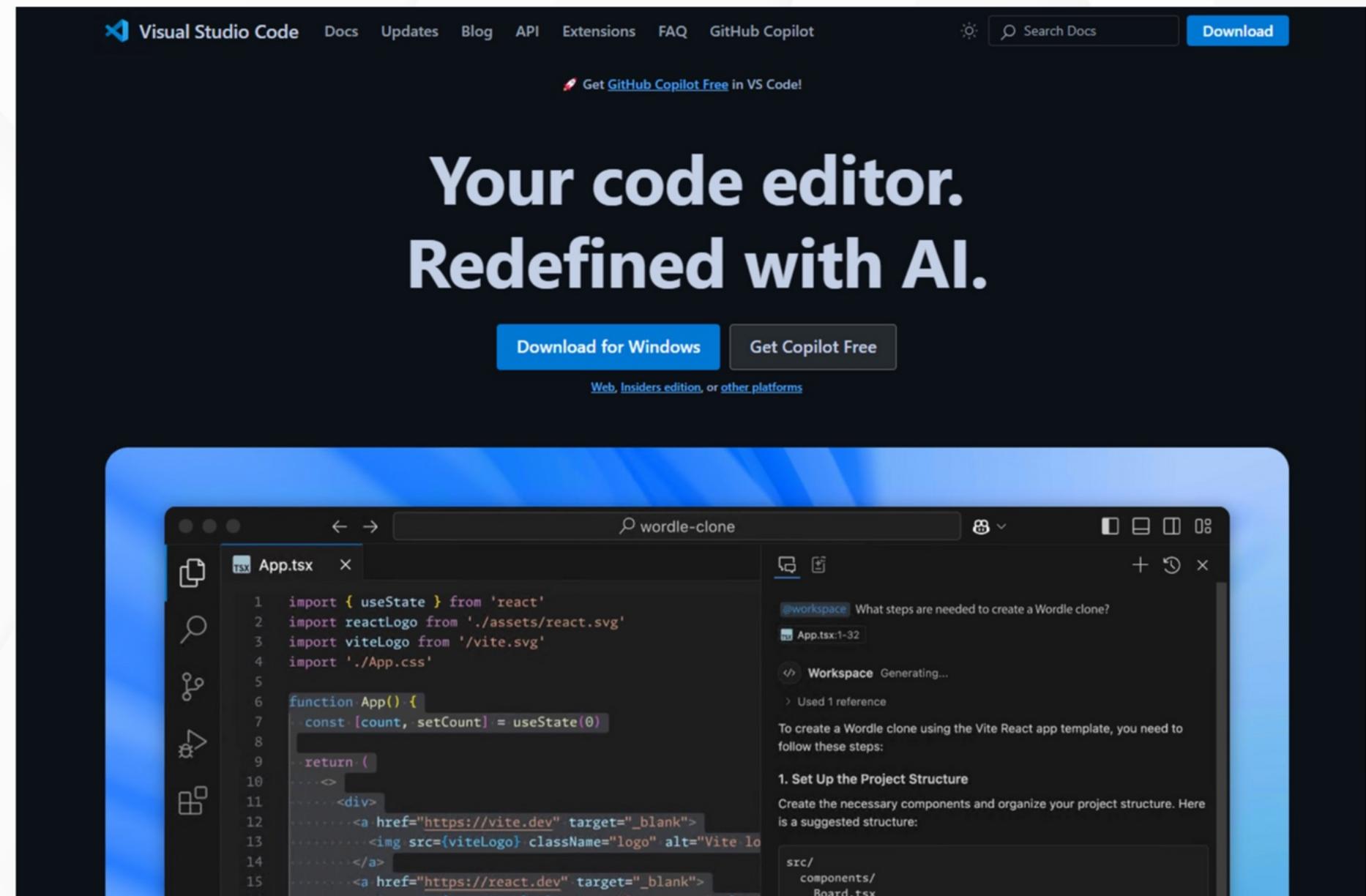
The screenshot shows the Docker website homepage. At the top, there is a navigation bar with links for Products, Developers, Pricing, Blog, About Us, Partners, a search icon, a "Sign In" button, and a prominent blue "Get Started" button. Below the navigation, the text "Docker Desktop" is displayed above a large, bold, dark blue headline: "The #1 containerization software for developers and teams". A subtitle below the headline reads "Your command center for innovative container development". At the bottom of this section are two blue buttons: "Create an account" and "Download for Mac - Intel Chip".

Docker Desktop



<https://www.docker.com>

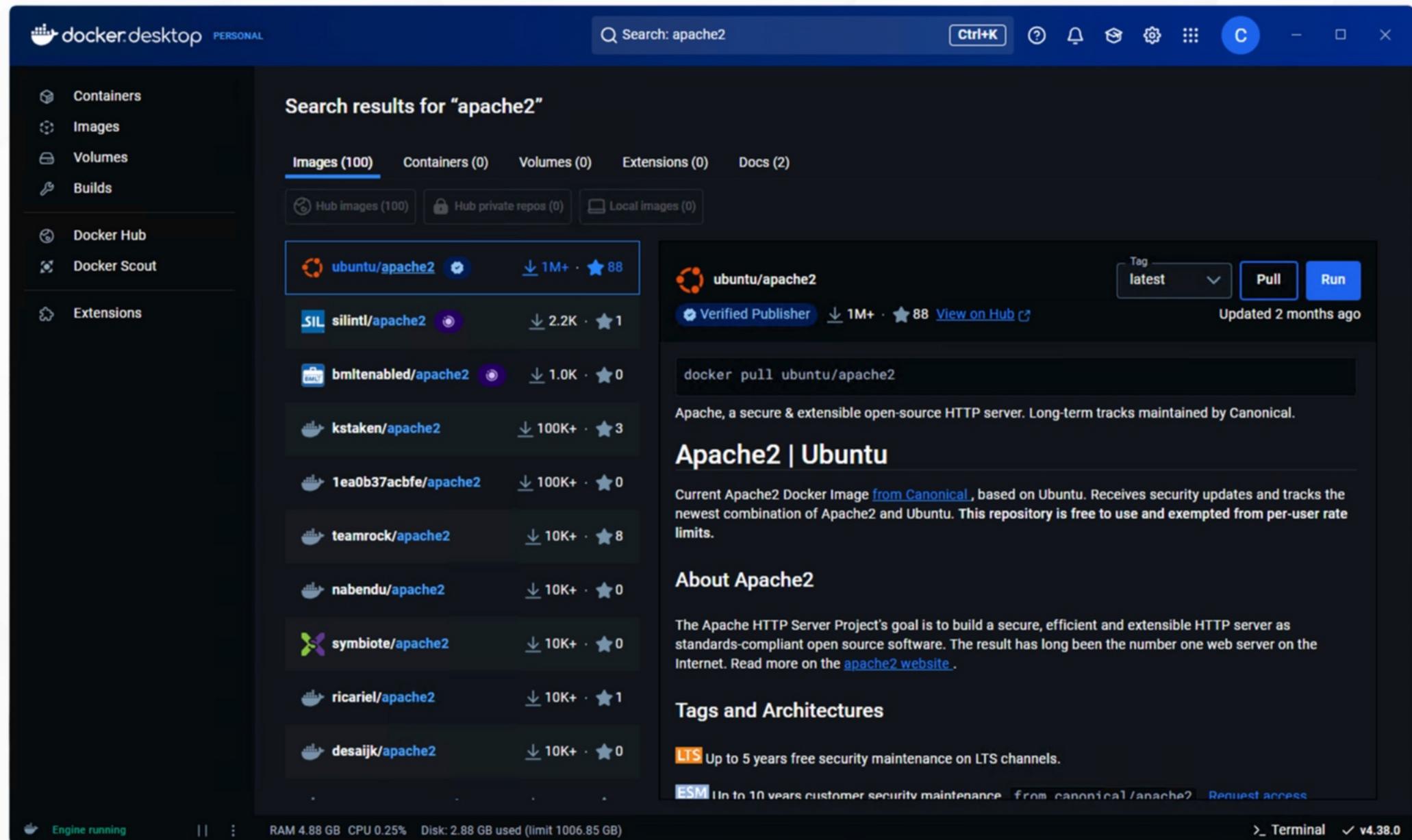
VSCode



<https://code.visualstudio.com>

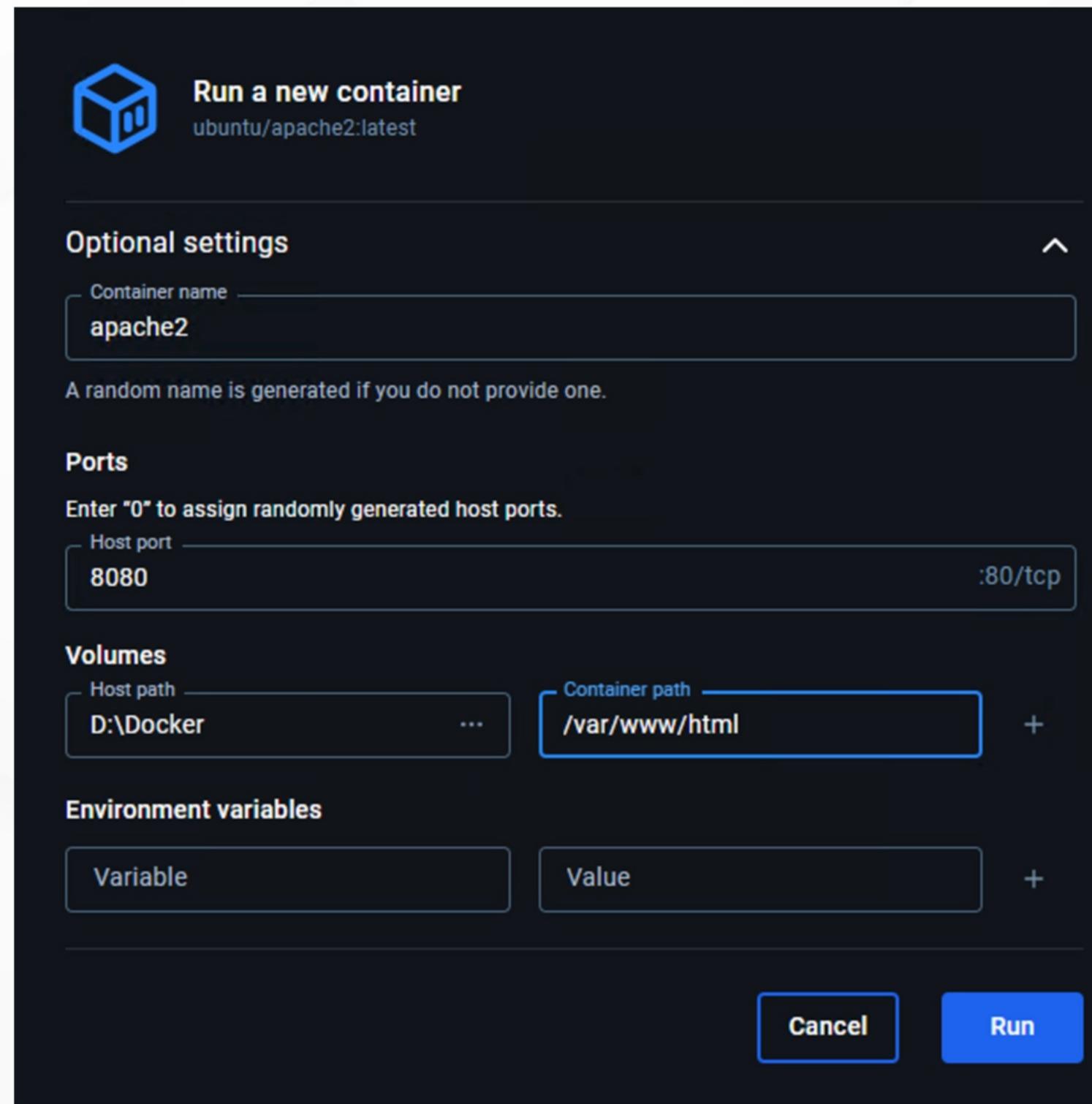
Run first container!

Run first web server container



- search “apache2”
- select “ubuntu/apache2”
- click “run”

apache2 container options



- Container name: **apache2**
- Host Port: **8080**
- Volumes (Host path): **D:\Docker**
- Container path: **/var/www/html**
- Environment variables:

web server available at : <http://localhost:8080>

docker image: <https://hub.docker.com/r/ubuntu/apache2>

Apache2 Run with CLI

```
docker run -d --name apache2 \
-p 8080:80 \
--volume D:\Docker\apache2:/var/www/html \
ubuntu/apache2:latest
```

Ports Options

[Host-Port]:[Container-Port]

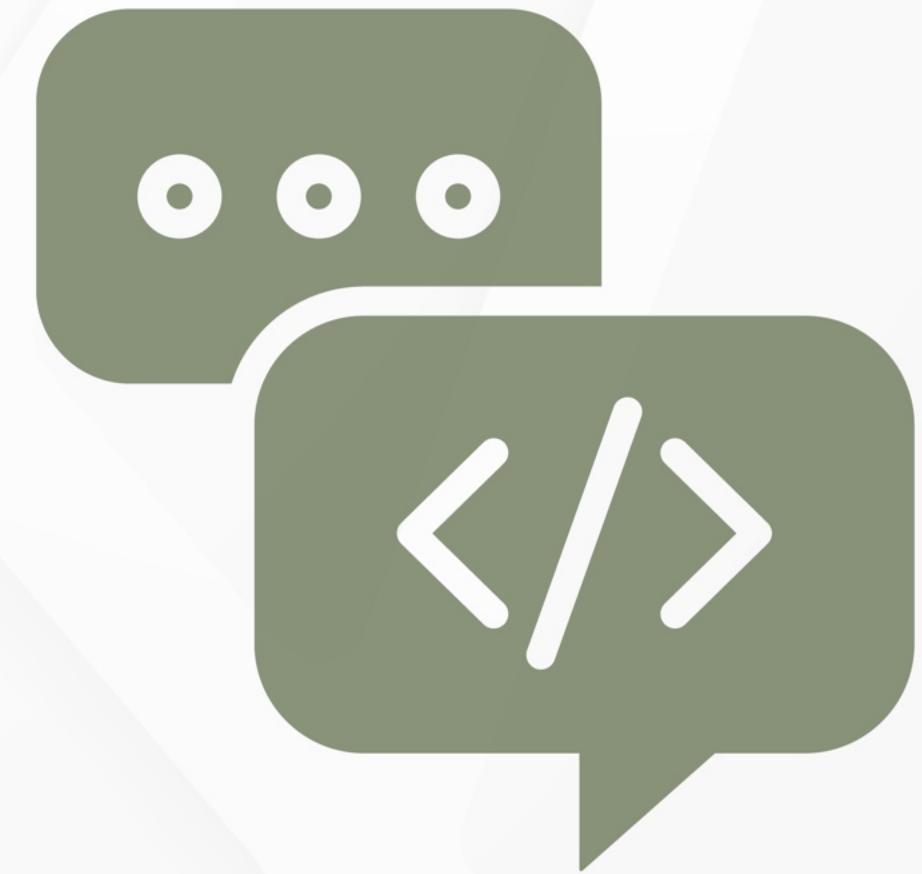
Allows external traffic to access the container's service via the host's port.

Volumes Options

[Host-Path]:[Container-Path]

Persistent storage that retains data even when containers are removed.

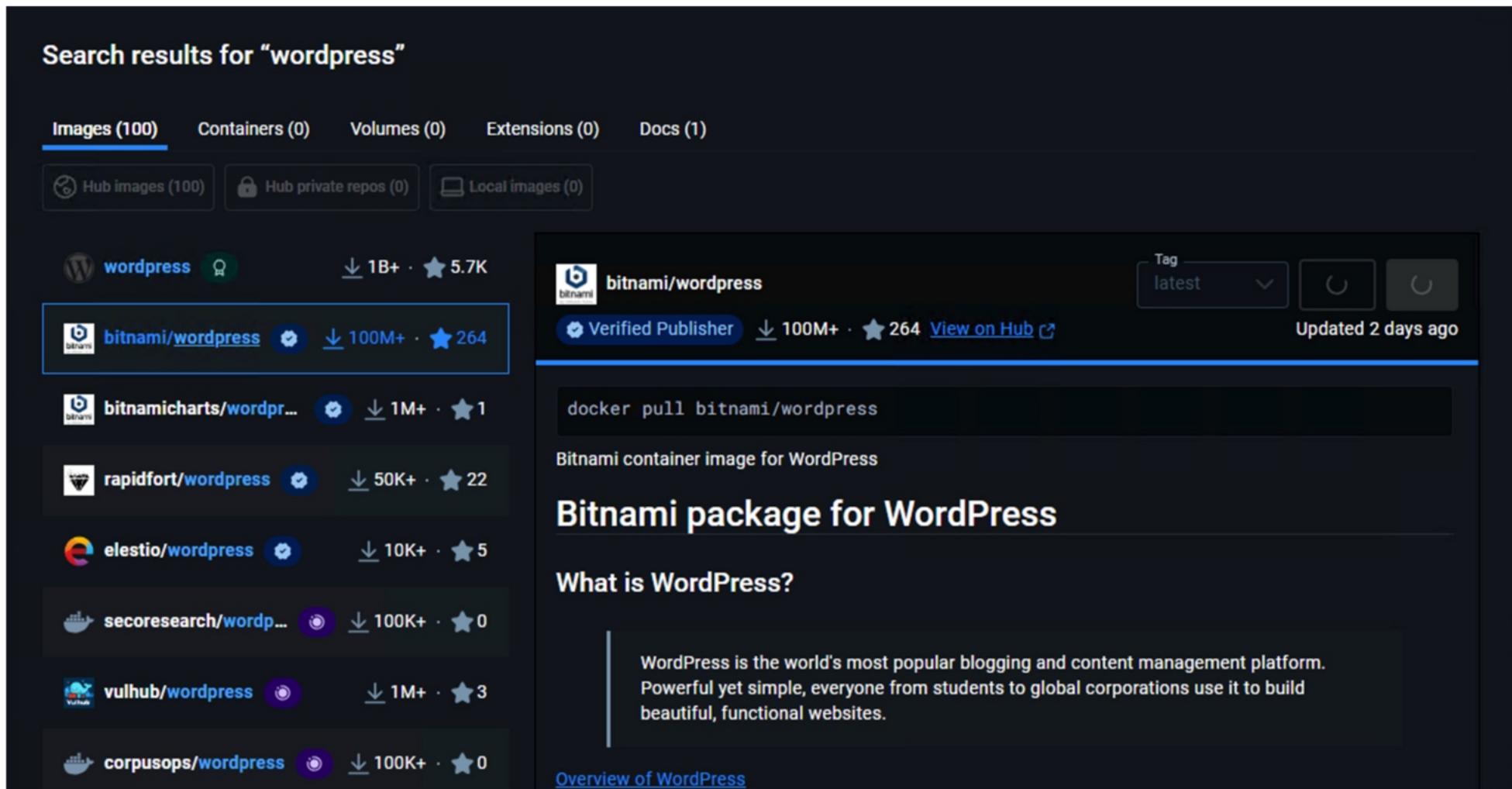
Environment (Variable) Options



Pass configuration settings to containers.

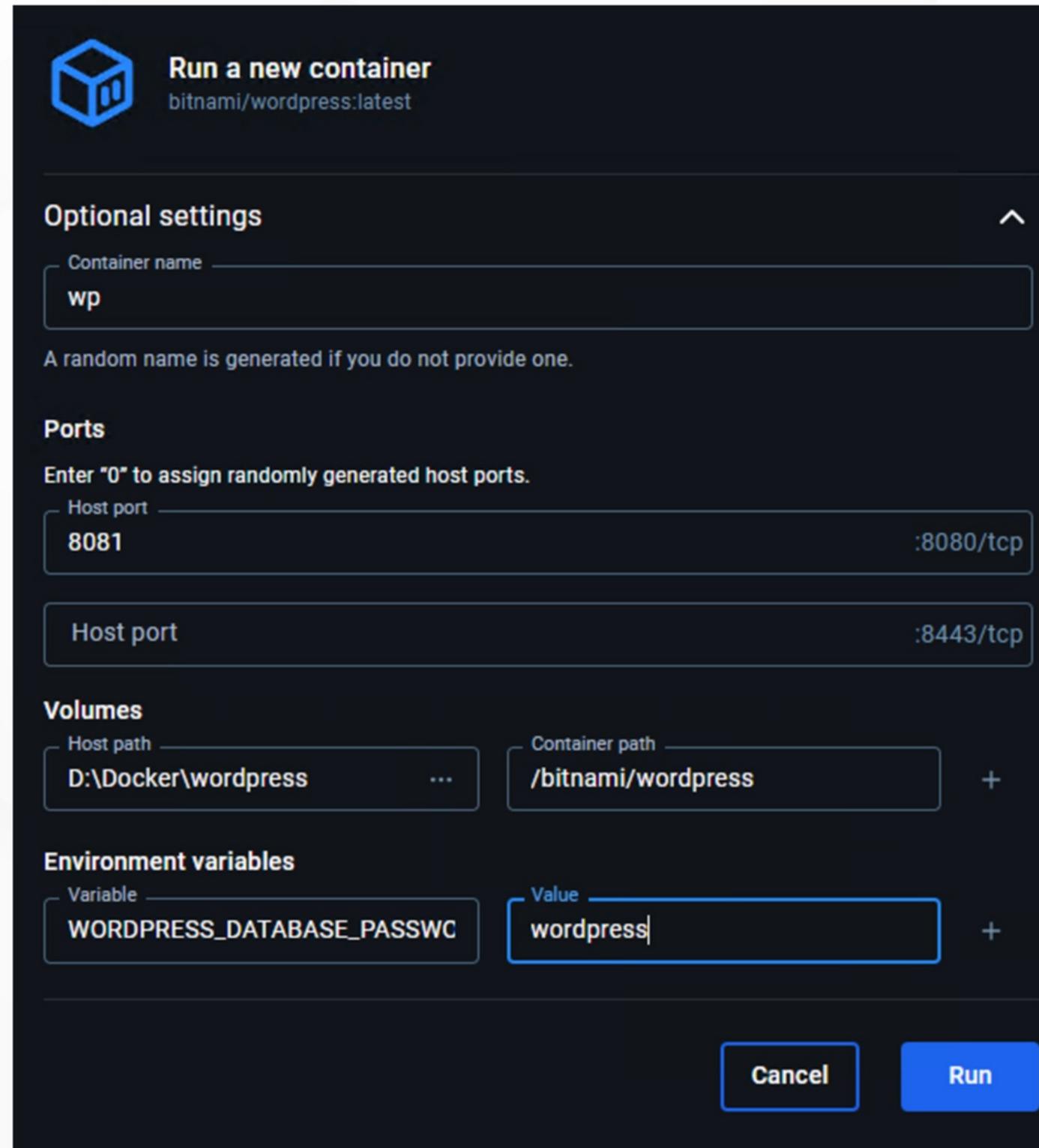
Run WordPress

Search “WordPress” Image



- search “wordpress”
- select “bitnami/wordpress”
- click “run”

WordPress container options



- Container name: **wordpress**
- Host Port: **8081, 8443**
- Volumes (Host path): **D:\Docker\wordpress**
- Container path: **/bitnami/wordpress**
- Environment variables:
 - WORDPRESS_DATABASE_USER=**bn_wordpress**
 - WORDPRESS_DATABASE_PASSWORD=**bitnami**
 - WORDPRESS_DATABASE_NAME=**bitnami_wordpress**

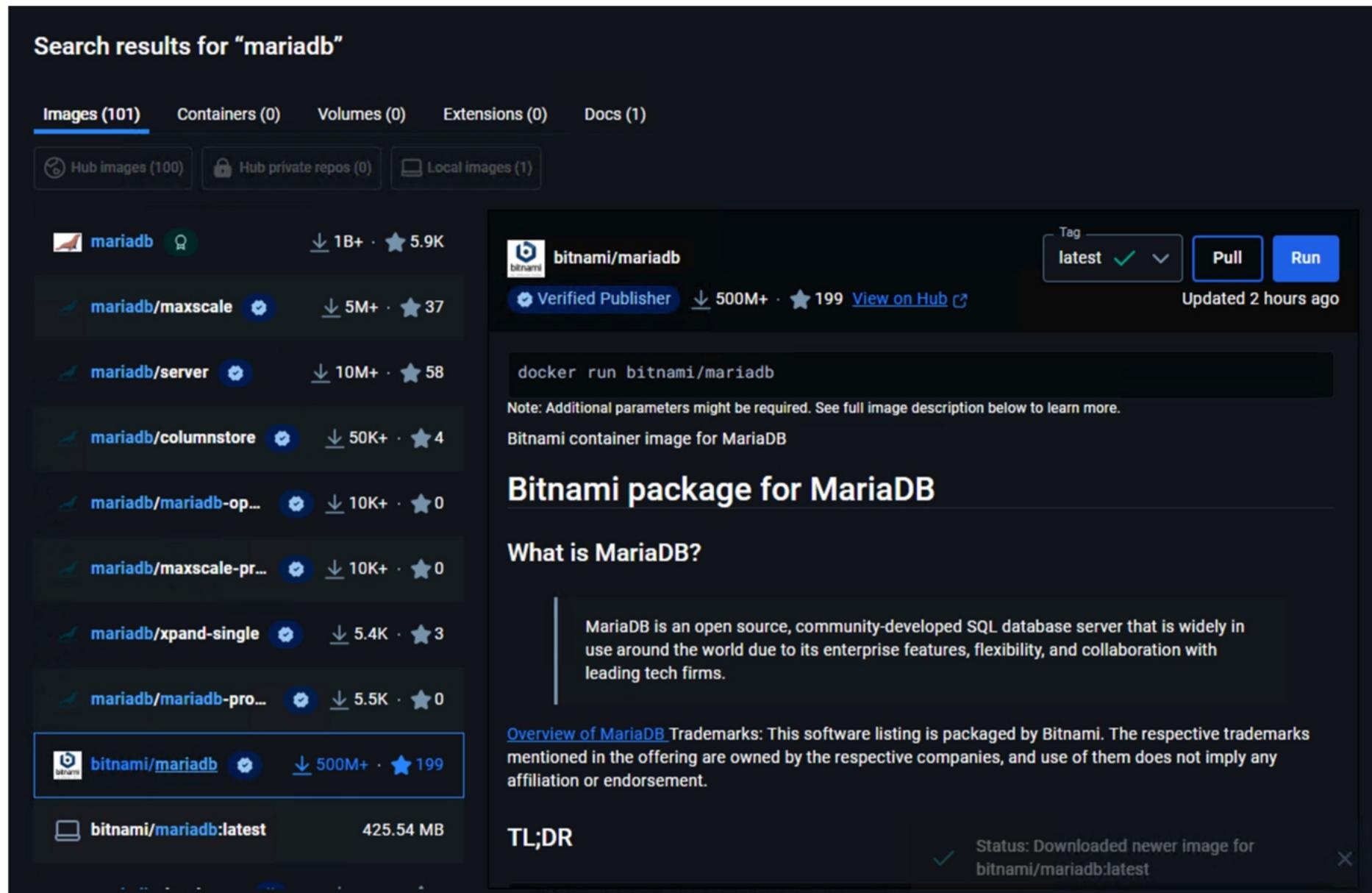
docker image: <https://hub.docker.com/r/bitnami/wordpress>

WordPress Run with CLI

```
docker volume create --name wordpress
docker run -d --name wordpress \
-p 8081:8080 -p 8443:8443 \
--env ALLOW_EMPTY_PASSWORD=yes \
--env WORDPRESS_DATABASE_USER=bn_wordpress \
--env WORDPRESS_DATABASE_PASSWORD=bitnami \
--env WORDPRESS_DATABASE_NAME=bitnami_wordpress \
--network wordpress-network \
--volume D:\Docker\wordpress:/bitnami/wordpress \
bitnami/wordpress:latest
```

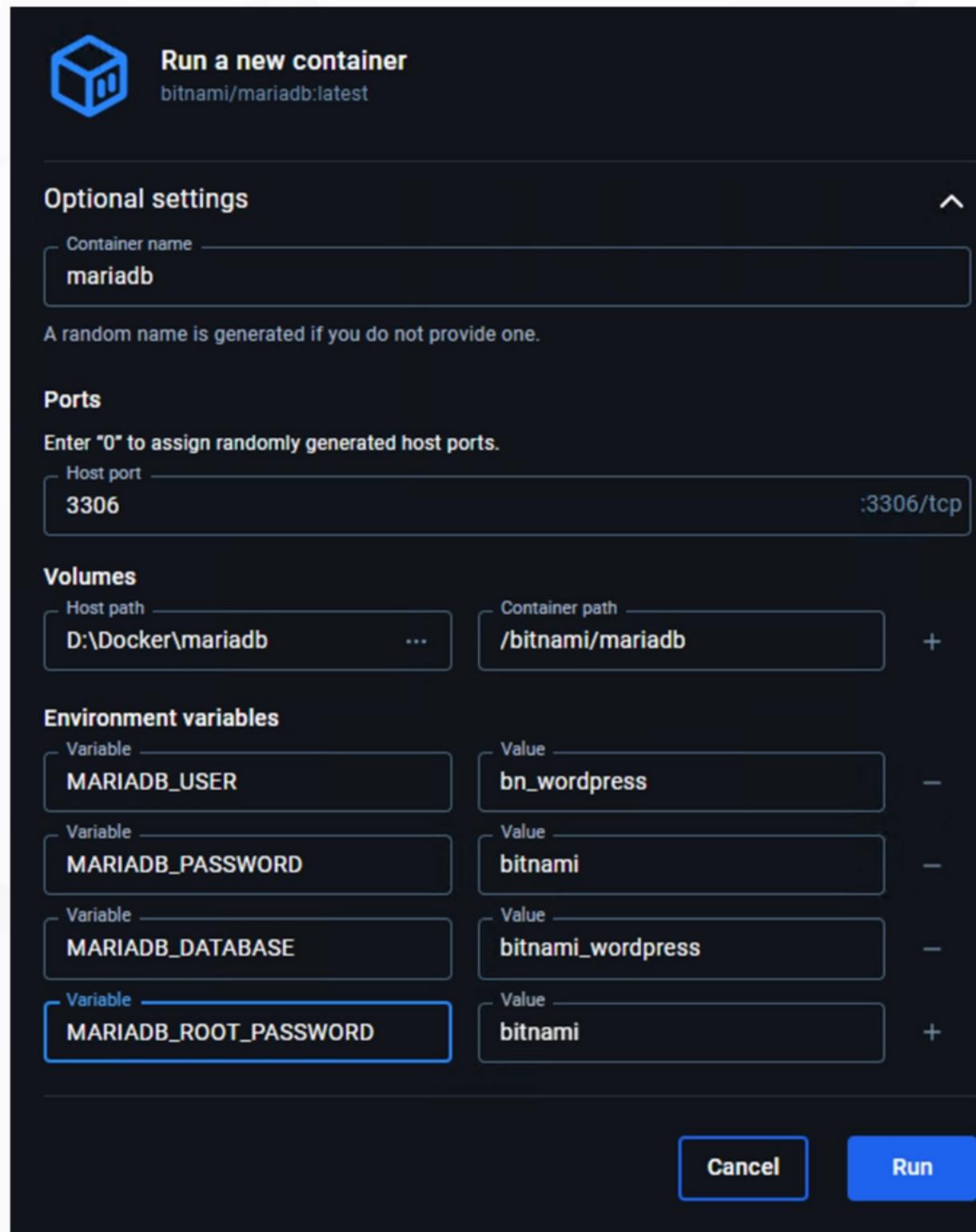
Run Database (MariaDB)

Search “MariaDB” Image



- search “mariadb”
- select “bitnami/mariadb”
- click “run”

MariaDB container options



- Container name: **mariadb**
- Host Port: **3306**
- Volumes (Host path): **D:\Docker\mariadb**
- Container path: **/bitnami/mariadb**
- Environment variables:
 - MARIADB_USER=**bn_wordpress**
 - MARIADB_PASSWORD=**bitnami**
 - MARIADB_DATABASE=**bitnami_wordpress**
 - MARIADB_ROOT_PASSWORD=**bitnami**

docker image: <https://hub.docker.com/r/bitnami/mariadb>

MariaDB Run with CLI

```
docker volume create --name mariadb
docker run -d --name mariadb \
--env ALLOW_EMPTY_PASSWORD=yes \
--env MARIADB_USER=bn_wordpress \
--env MARIADB_PASSWORD=bitnami \
--env MARIADB_DATABASE=bitnami_wordpress \
--network wordpress-network \
--volume D:\Docker\mariadb:/bitnami/mariadb \
bitnami/mariadb:latest
```

Docker Command



Docker Version

```
</> $ docker version

Client: Docker Engine - Community
Version: 24.0.5
API version: 1.43
Go version: go1.20.6
Git commit: ced0996
Built: Fri Jul 21 20:35:35 2023
OS/Arch: linux/amd64
Context: default

Server: Docker Engine - Community
Engine:
Version: 24.0.5
API version: 1.43 (minimum version 1.12)
Go version: go1.20.6
Git commit: a61e2b4
Built: Fri Jul 21 20:35:35 2023
OS/Arch: linux/amd64
Experimental: true
containerd:
Version: 1.6.22
GitCommit: 8165feabfdf38c65b599c4993d227328c231fca
runc:
Version: 1.1.8
GitCommit: v1.1.8-0-g82f18fe
docker-init:
Version: 0.19.0
GitCommit: de40ad0
```

Client Info

Server Info

Docker Info



```
$ docker info

Client: Docker Engine - Community
Version: 24.0.5
Context: default
Debug Mode: false
Plugins:
buildx: Docker Buildx (Docker Inc.)
  Version: v0.11.2
  Path: /usr/libexec/docker/cli-plugins/docker-buildx
compose: Docker Compose (Docker Inc.)
  Version: v2.20.2
  Path: /usr/libexec/docker/cli-plugins/docker-compose

Server:
Containers: 18
Running: 17
Paused: 0
Stopped: 1
Images: 70
Server Version: 24.0.5
.....
```

Docker Hub

<https://hub.docker.com>

The screenshot shows the Docker Hub homepage. At the top is a blue header bar with the Docker Hub logo, a search bar, and navigation links for Explore, Repositories, Organizations, Help, and an Upgrade button. A user profile for 'smilexth' is also visible. On the left, there's a sidebar titled 'Filters' with sections for Products (Images, Extensions, Plugins), Trusted Content (Docker Official Image, Verified Publisher, Sponsored OSS), Operating Systems (Linux, Windows), Architectures (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE, x86, x86-64), and a 'Suggested' dropdown menu. The main content area displays four repository cards: 'alpine' (Docker Official Image, 1B+, 10K+, Updated a month ago, Pulls: 11,710,561 last week), 'nginx' (Docker Official Image, 1B+, 10K+, Updated 7 days ago, Pulls: 13,783,477 last week), 'busybox' (Docker Official Image, 1B+, 3.1K, Updated 2 months ago, Pulls: 8,661,233 last week), and 'ubuntu' (Docker Official Image, 1B+, 10K+, Updated 12 days ago, Pulls: 28,548,181 last week). Each card includes a brief description and a 'Learn more' link.

Pull base image(s)



```
$ docker pull ubuntu
```

```
Using default tag: latest latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete b78396653dae: Pull complete Digest:
sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6 Status:
Downloaded newer image for ubuntu:latest
```

Pull base image(s)



```
</> $ docker pull ubuntu

Using default tag: latest
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
```

Pull base image(s)



```
$ docker pull ubuntu:17.10
```

```
Using default tag: 17.10 latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete b78396653dae: Pull complete Digest:
sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6 Status:
Downloaded newer image for ubuntu:latest
```

Pull base image(s)



\$ docker pull ubuntu:17.10

```
Using default tag: 17.10 latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete b78396653dae: Pull complete Digest:
sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6 Status:
Downloaded newer image for ubuntu:latest
```

Images List



\$ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ghcr.io/hassio-addons/jupyterlab/amd64	0.12.0	0b264c1f2404	13 days ago	1.03GB
ghcr.io/home-assistant/amd64-hassio-supervisor	2023.09.2	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/amd64-hassio-supervisor	latest	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/qemu86-64-homeassistant	2023.9.2	e65d980c0e5b	2 weeks ago	1.81GB
ghcr.io/hassio-addons/node-red/amd64	14.5.0	2bdcfadcf154	2 weeks ago	432MB
ghcr.io/hassio-addons/ssh/amd64	15.0.8	d6d5709fb46f	2 weeks ago	303MB
ghcr.io/hassio-addons/vscode/amd64	5.10.2	6810ca970e64	2 weeks ago	921MB
homeassistant/amd64-addon-mosquitto	6.3.1	973c50be2983	3 weeks ago	209MB
ghcr.io/esphome/esphome-hassio	2023.8.3	caf2a5393e4d	3 weeks ago	348MB
zigbee2mqtt/zigbee2mqtt-amd64	1.33.0-1	a0d3178a6a35	3 weeks ago	170MB
eb9ee392/amd64-addon-racksync-autossh-tunnel	2023.8.2	13ea7d26e548	5 weeks ago	123MB
eb9ee392/amd64-addon-racksync-cloudflare-ddns	2023.8.1	4180ac205e6e	6 weeks ago	89.9MB
eb9ee392/amd64-addon-racksync-cloudflare-tunnel-zerotrust	2023.8.1	a02155e98ae8	6 weeks ago	126MB

Container Run

```
</> $ docker run ubuntu /bin/echo "hello, world"
```

```
"hello, world"
```

```
</> $ docker runubuntu /bin/echo "hello, world"
```

```
"hello, world"
```



Container List

 \$ docker ps

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
97fcfa592136	homeassistant/amd64-addon-configurator:5.6.0		/init"	3 days ago	Up 3 days (healthy)	
f4cdaf1c9a5c	ghcr.io/home-assistant/qemux86-64-homeassistant:2023.9.2	addon_core_configurator	/init"	10 days ago	Up 18 hours	
01218d798963	ghcr.io/hassio-addons/vscode/amd64:5.10.2	homeassistant	/init"	10 days ago	Up 10 days (healthy)	
72dalb61d9cf	ghcr.io/hassio-addons/node-red/amd64:14.5.0	addon_a0d7b954_vscode	/init"	10 days ago	Up 10 days (healthy)	

 \$ docker ps -a ◀..... a = see all available containers.

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
97fcfa592136	homeassistant/amd64-addon-configurator:5.6.0		/init"	3 days ago	Up 3 days (healthy)	
f4cdaf1c9a5c	ghcr.io/home-assistant/qemux86-64-homeassistant:2023.9.2	addon_core_configurator	/init"	10 days ago	Up 18 hours	
01218d798963	ghcr.io/hassio-addons/vscode/amd64:5.10.2	homeassistant	/init"	10 days ago	Up 10 days (healthy)	
72dalb61d9cf	ghcr.io/hassio-addons/node-red/amd64:14.5.0	addon_a0d7b954_vscode	/init"	10 days ago	Up 10 days (healthy)	
79f84385c64d	ghcr.io/hassio-addons/ssh/amd64:15.0.8	addon_a0d7b954_nodered	/init"	10 days ago	Up 10 days	

Container Run with Argument



```
$ docker run -d ubuntu /bin/sh -c \
"while true; do echo hello world; sleep 1; done;"
```

```
ac74639343196e056d3a2e27f8f63f1417529a6d8ea7176ee0e50075d34eea6a
```



```
$ docker ps
```

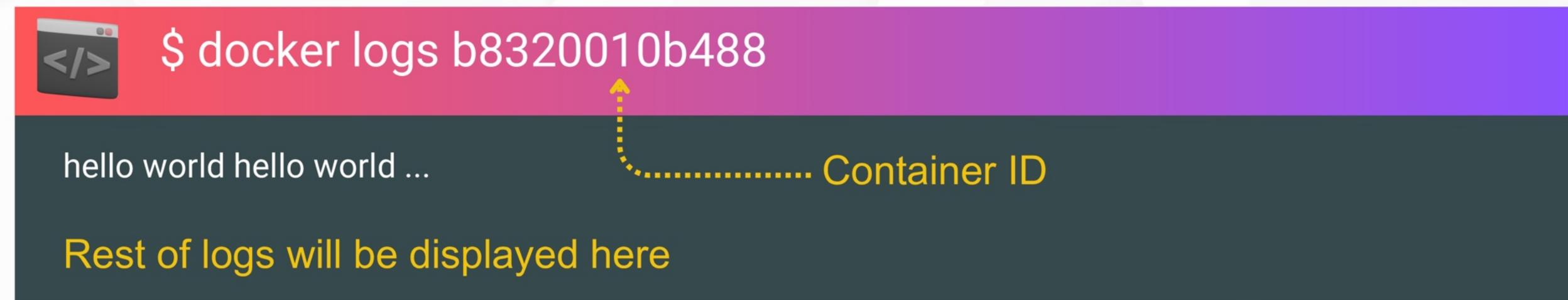
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b8320010b488	ubuntu	/bin/sh -c 'while t...	2 seconds ago	Up 1 second		epic_einstein



Container ID

Random Container Name

Container Logs



A terminal window interface. The top bar is red with a black terminal icon on the left and the command '\$ docker logs b8320010b488' in white. The main area is dark grey and displays the log output: 'hello world hello world ...'. A yellow dotted arrow points from the text 'Container ID' to the container ID 'b8320010b488' in the command line. Below the logs, the text 'Rest of logs will be displayed here' is shown in yellow.

```
$ docker logs b8320010b488
hello world hello world ...
Rest of logs will be displayed here
```

Container ID

Expose Port



```
$ docker run -d -p 8090:80 nginx:alpine
```

```
bc648a0e5ef9e7364e67f94f9993083c38b4be566f0f8312e812fd7325344f27
```



```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
bc648a0e5ef9 seconds ago	nginx:alpine Up 41 seconds	"nginx -g 'daemon off...'" 443/tcp, 0.0.0.0:8090->80/tcp	"nginx -g 'daemon of..." 42	flamboyant_pike

Expose Port

-p <host_port>:<guest_port>

Public Port
(Port: 8090)

Internal Port
(Port: 80)

```
$ docker run -d -p 8090:80 nginx:alpine
```

bc648a0e5ef9e7364e67f94f9993083c38b4be566f0f8312e812fd7325344f27

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
POR TS seconds ago	NAMES bc648a0e5ef9	nginx:alpine "nginx -g 'daemon of..."	42	443/tcp, 0.0.0.0:8090->80/tcp flamboyant_pike
Up 41 seconds				

How to enter the container



```
$ docker run -d --name my-nginx nginx
```

```
ee3a421f38a9b12ffffc4666cbc23b0a0053b184f217d64f1cd736a8f485ba0d
```



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ee3a421f38a9 seconds	nginx:alpine 80/tcp, 443/tcp	"nginx -g 'daemon of..." my-nginx	50 seconds ago	Up 49	



```
$ docker exec -it my-nginx bash
```

```
root@729e05ec6bed:/#
```

Execute to container or shell



```
$ docker run -d --name my-nginx nginx
```



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ee3a421f38a9 seconds	nginx:alpine 80/tcp, 443/tcp	"nginx -g 'daemon of..." my-nginx	50 seconds ago	Up 49	



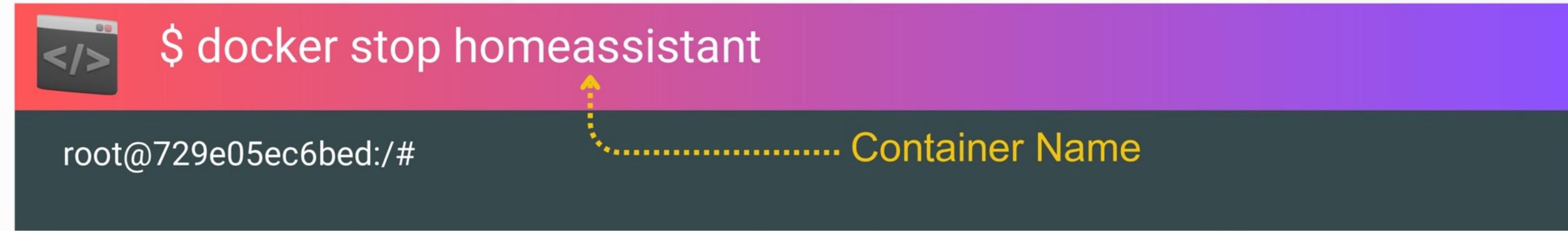
```
$ docker exec -it my-nginx bash
```

```
root@729e05ec6bed:/#
```



Container Name (Service)

Stop Container



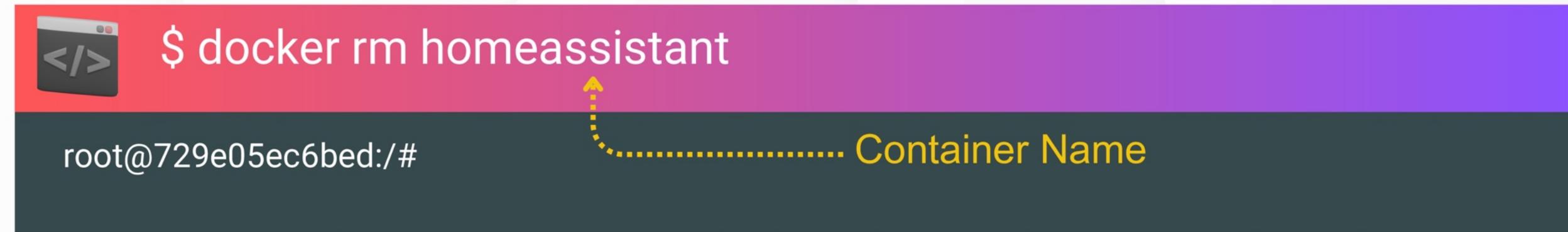
A terminal window with a red header bar containing a terminal icon and the command '\$ docker stop homeassistant'. The main body of the terminal is dark grey with white text showing the prompt 'root@729e05ec6bed:/#' and a yellow dotted arrow pointing from the word 'Container Name' to the end of the command line.

```
$ docker stop homeassistant
```

root@729e05ec6bed:/#

..... Container Name

Delete Container

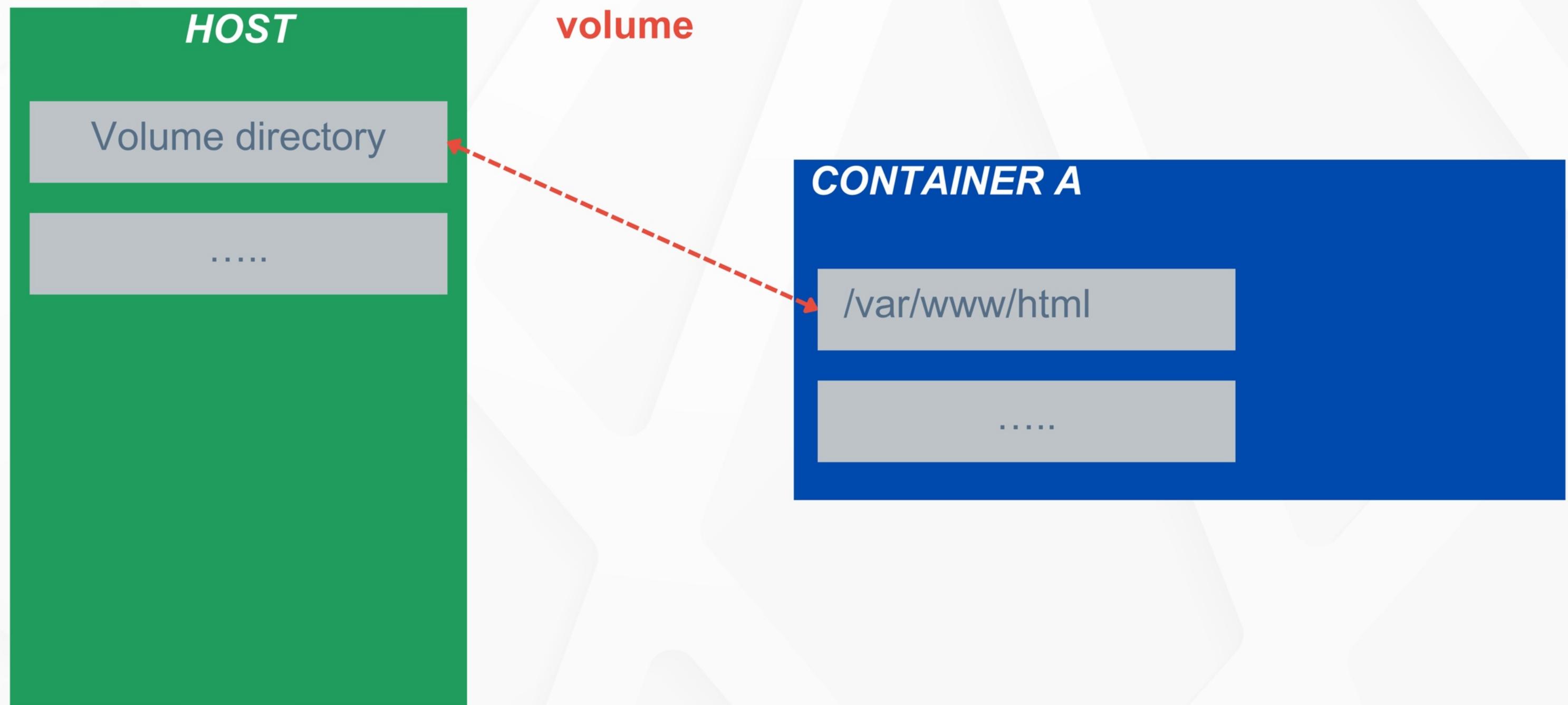


```
$ docker rm homeassistant
```

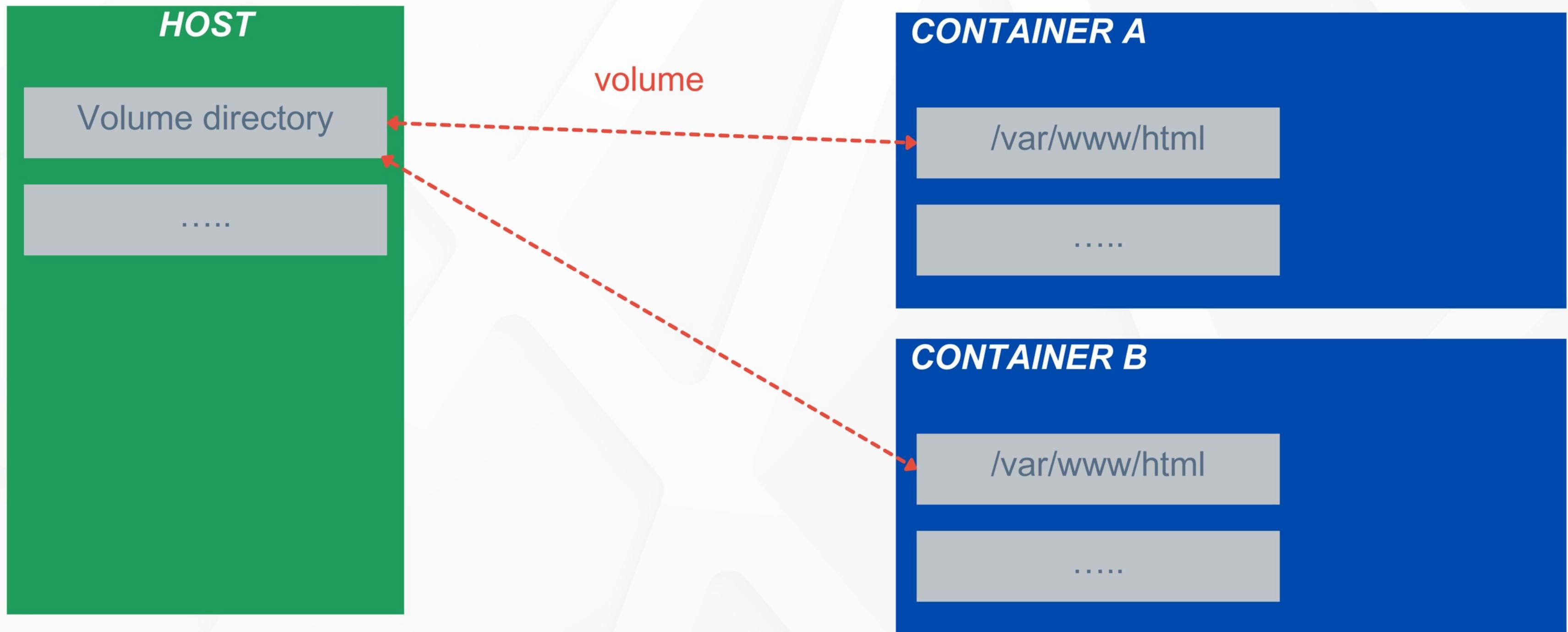
root@729e05ec6bed:/#

..... Container Name

Volume



Share volume between containers



Share volume to container

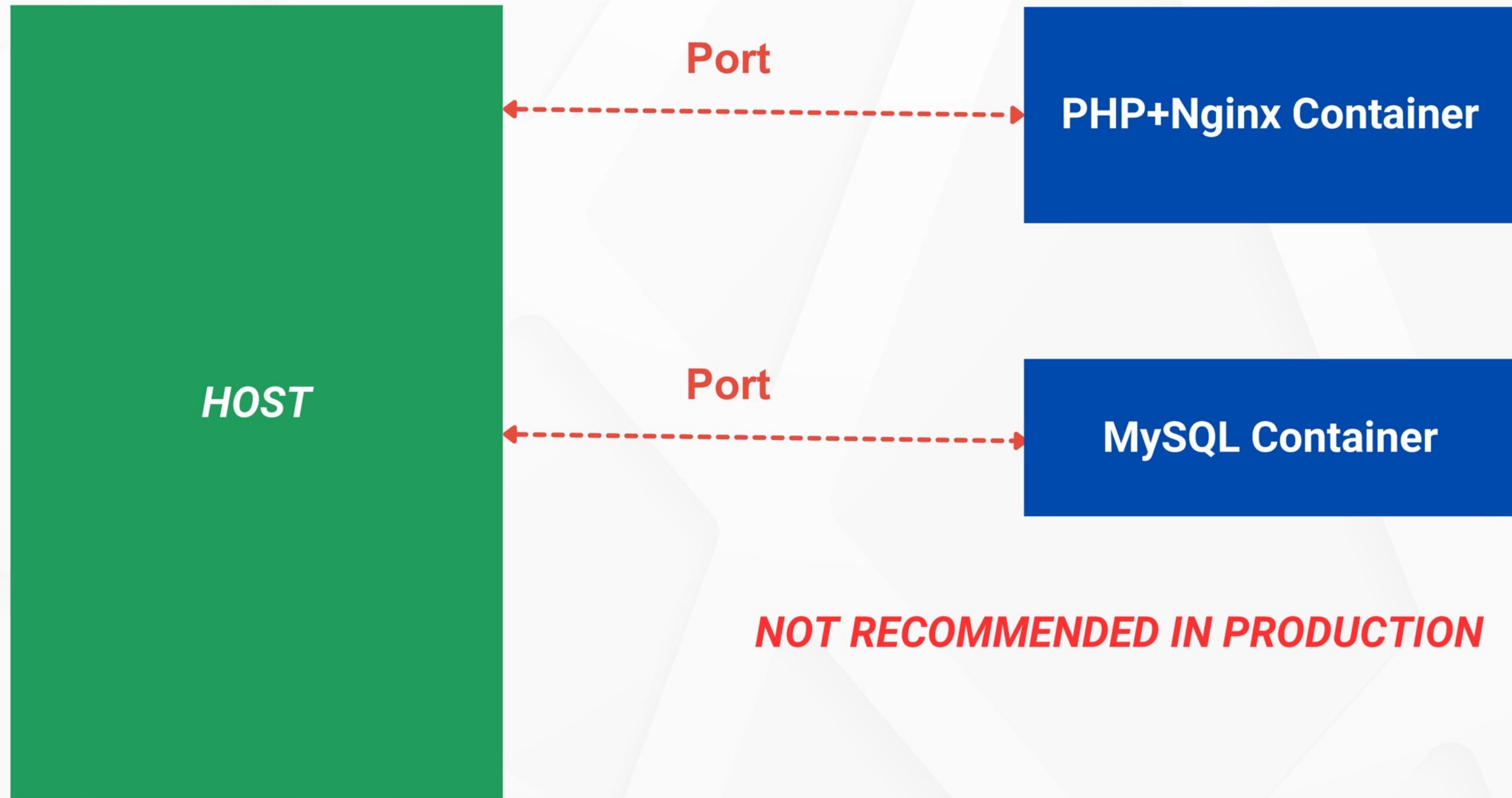


```
$ docker run -d -v ./my-project:/var/www/ --name wordpress my-wordpress
```

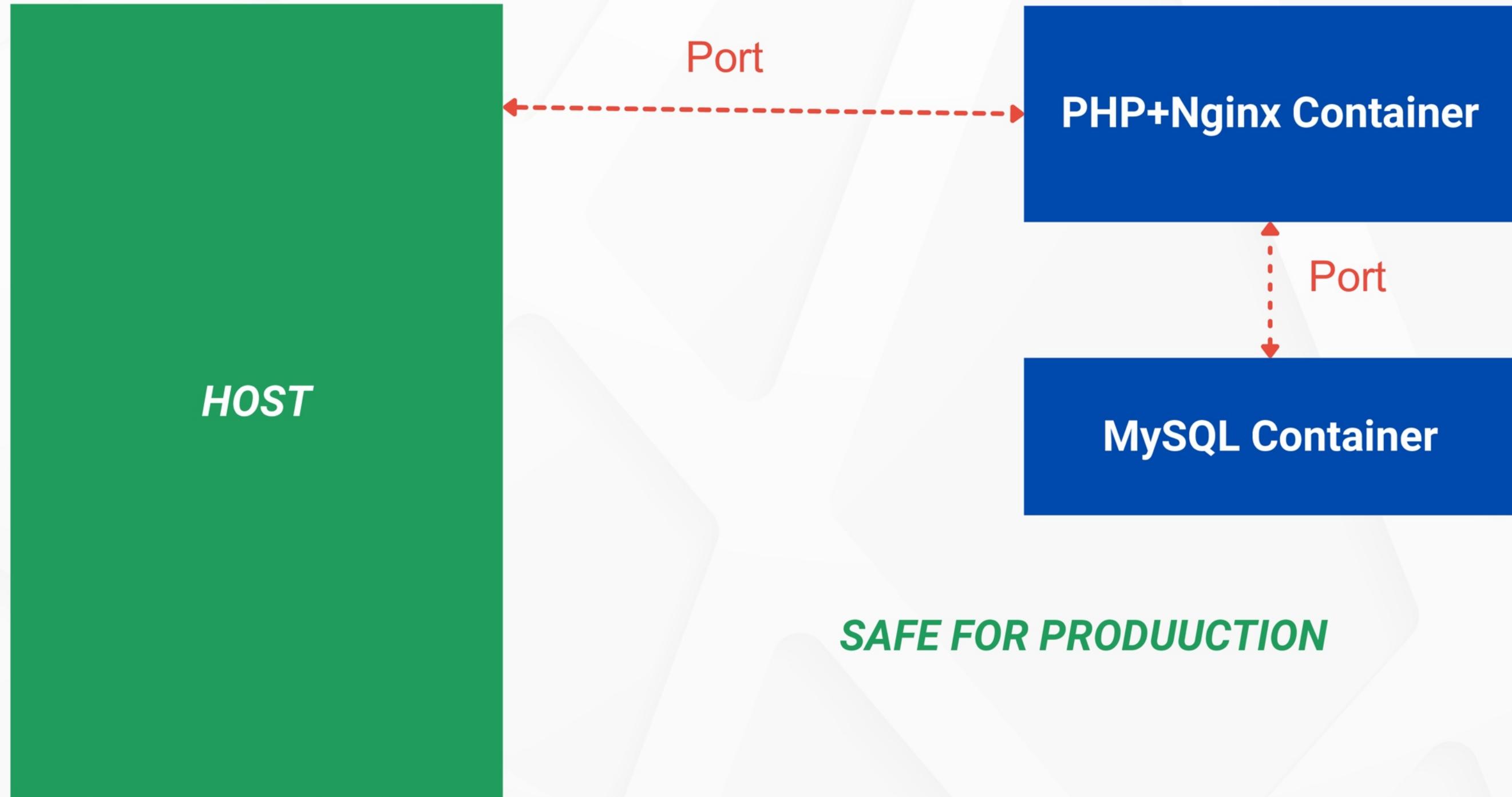


```
$ docker run -d -v ./my-project:/var/www/ --name wordpress my-wordpress  
-v <host_path>:<container_path>
```

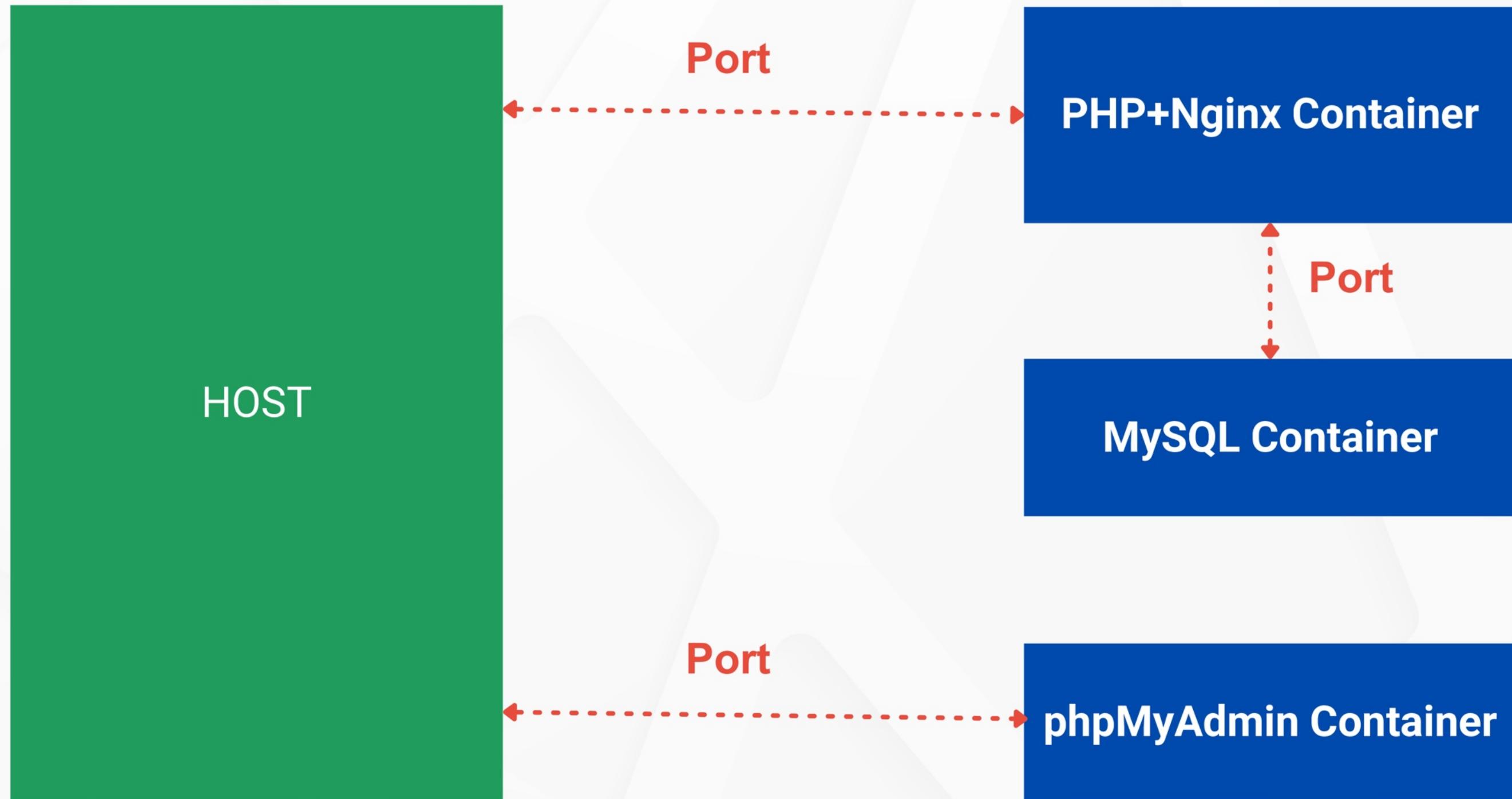
Container communication



Container communication



Container communication



Docker Network

Create a network



```
$ docker network create my-net
```

```
90a679f09e80a0e4a4618ae95dac9d721c768d8f6760b31242e526bd62fb2554
```



```
$ docker run -d --name nginx1 --net my-net nginx:alpine
```

```
263f4ec6131d933e879e06f5291f8c39b3d1a81abee094555b9e8b0e6e716164
```

Create a container for your network.



```
$ docker run -d --name nginx2 --net my-net nginx:alpine
```

```
a3d741fedd733855207ac81bc8792b0af6b3ddc8ec481d1a07d0a8d7f15ce039
```

Container communication

Create a network



```
$ docker network create my-net
```



Network Name

```
90a679f09e80a0e4a4618ae95dac9d721c768d8f6760b31242e526bd62fb2554
```



```
$ docker run -d --name nginx1 --net my-net nginx:alpine
```

```
263f4ec6131d933e879e06f5291f8c39b3d1a81abee094555b9e8b0e6e716164
```

Create a container for your network.



```
$ docker run -d --name nginx2 --net my-net nginx:alpine
```



Network Name

```
a3d741fedd733855207ac81bc8792b0af6b3ddc8ec481d1a07d0a8d7f15ce039
```

Inspect The Network

Check the network to see which containers are there.



```
$ docker network inspect my-net
```

```
...
"Containers": { "263f4ec6131d933e879e06f5291f8c39b3d1a81abee094555b9e8b0e6e716164": {
  "Name": "nginx1", "EndpointID":
  "9e990663ad0b354cd4a50cab4ccaf0d30a84c529d9ef1923d857f1825e55ad04", "MacAddress":
  "02:42:c0:a8:40:02", "IPv4Address": "192.168.64.2/20", "IPv6Address": "" },
  "a3d741fedd733855207ac81bc8792b0af6b3ddc8ec481d1a07d0a8d7f15ce039": { "Name":
  "nginx2", "EndpointID":
  "171bb27f7be6d5203246d192d1039cd60f61d5ec8faa02b9d5da481910208a17", "MacAddress":
  "02:42:c0:a8:40:03", "IPv4Address": "192.168.64.3/20", "IPv6Address": "" } }, ...
```

Container communication check



```
</> $ docker exec -it nginx1 ping nginx2
```

PING nginx2 (192.168.64.3): 56 data bytes
64 bytes from 192.168.64.3: seq=0 ttl=64 time=0.250 ms
64 bytes from 192.168.64.3: seq=1 ttl=64 time=0.138 ms ^C

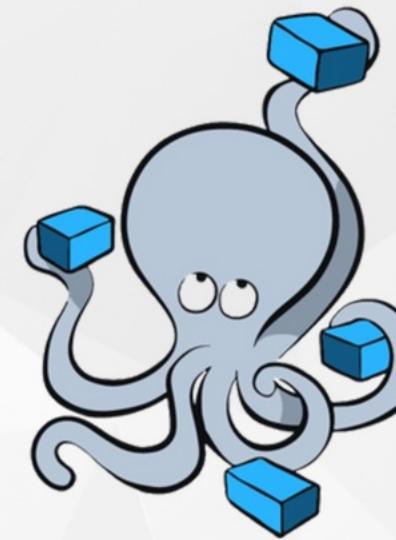
Try testing the ping back of each container.

```
</> $ docker exec -it nginx2 ping nginx1
```

PING nginx1 (192.168.64.2): 56 data bytes
64 bytes from 192.168.64.2: seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.64.2: seq=1 ttl=64 time=0.127 ms ^C

Docker Compose

What is docker-compose?



docker
Compose

“ Docker Compose is a way to combine various Docker commands into a single yaml file. And you can manage multiple containers using a single command.

docker-compose.yml



\$ vi docker-compose.yml

```
version: '3.7'
services:
  mariadb:
    container_name: mariadb
    image: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      TZ: "Asia/Bangkok"
    ports:
      - "3306:3306"
    volumes:
      - ./data/mariadb:/var/lib/mysql
    networks:
      - mariadb-network
    restart: always
  networks:
    mariadb-network:
```

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose.yml



\$ vi docker-compose.yml

1. Version:

- The file is written in version '3.7' of the Docker Compose file format.

2. Services:

- This section describes the services that are to be created when docker-compose up is run.
- **mariadb:**
 - This is the name of the service.
 - **container_name:** The name of the container when it's instantiated will be "mariadb".
 - **image:** This service will use the official "mariadb" Docker image from DockerHub.
 - **environment:** Sets environment variables inside the container.
 - **MYSQL_ROOT_PASSWORD:** The password for the root user of MariaDB is set to "rootpassword".
 - **TZ:** Sets the timezone inside the container to "Asia/Bangkok".
 - **ports:** Maps port 3306 of the host to port 3306 of the container.
 - **volumes:** Maps the ./data/mariadb directory on the host to /var/lib/mysql inside the container. This ensures data persistence across container restarts.
 - **networks:** The service is connected to a custom network named "mariadb-network".
 - **restart:** If the container exits, Docker will always try to restart it.

3. Networks:

- **mariadb-network:** This section defines a custom network. Networks allow you to define communication between containers and can also ensure isolation between containers.

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



```
$ docker-compose up -d
```

*** Create and start all services

..... d = detach from command after run



```
$ docker-compose start
```

*** Start all services



```
$ docker-compose stop
```

*** Stop all services

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



```
$ docker-compose up -d nginx
```

*** Build and start only nginx

..... Service Name



```
$ docker-compose start nginx
```

*** Start only nginx



```
$ docker-compose stop nginx
```

*** Stop only nginx

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



```
$ docker-compose restart
```

```
*** restart all services
```



```
$ docker-compose restart nginx
```

```
*** restart only nginx
```



..... Service Name

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



```
$ docker-compose down
```



```
*** Destroy all services (without destroying volume )
```



```
$ docker-compose down -v
```



```
*** Destroy all services and delete volumes as well.
```

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



```
$ docker-compose logs
```

*** See log



```
$ docker-compose exec -it nginx bash
```

*** Get into container / execute command.

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command

```
 $ docker-compose logs  
*** See log
```

```
 $ docker-compose exec -it nginx bash  
*** Get into container / execute command.  
  
..... Get into terminal
```

Reference: <https://docs.docker.com/compose/reference/overview/>

docker-compose command



\$ docker-compose ps

*** See how services work.



\$ docker-compose config

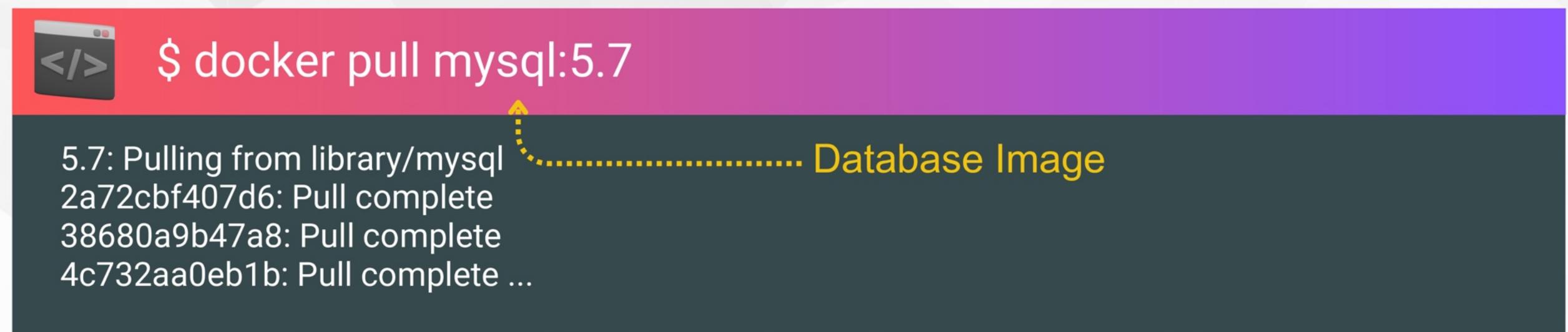
*** to check files docker-compose Is it written correctly?

Reference: <https://docs.docker.com/compose/reference/overview/>

Docker Images

Pull Images

Pull MySQL image onto the machine.



```
</> $ docker pull mysql:5.7
5.7: Pulling from library/mysql
2a72cbf407d6: Pull complete
38680a9b47a8: Pull complete
4c732aa0eb1b: Pull complete ...
..... Database Image
```

A terminal window with a red header bar containing a terminal icon and the command '\$ docker pull mysql:5.7'. The main body of the terminal shows the output of the command, which includes several lines of text and a yellow dotted arrow pointing from the word 'Database' to the word 'Image' at the end of the last line.

List Images

List all images

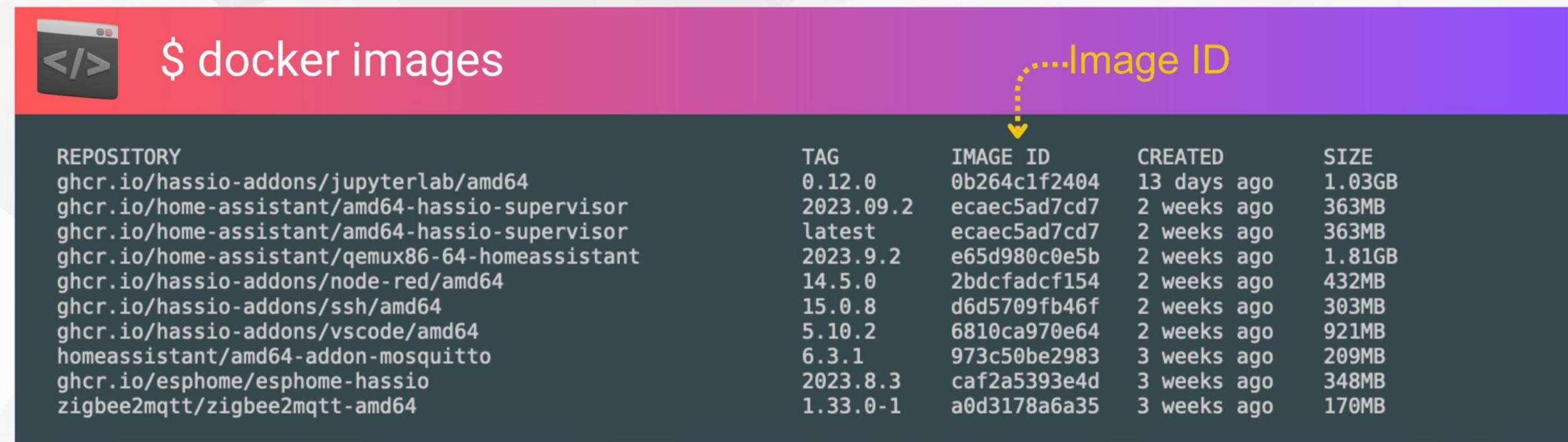


\$ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ghcr.io/hassio-addons/jupyterlab/amd64	0.12.0	0b264c1f2404	13 days ago	1.03GB
ghcr.io/home-assistant/amd64-hassio-supervisor	2023.09.2	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/amd64-hassio-supervisor	latest	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/qemu86-64-homeassistant	2023.9.2	e65d980c0e5b	2 weeks ago	1.81GB
ghcr.io/hassio-addons/node-red/amd64	14.5.0	2bdcfadcf154	2 weeks ago	432MB
ghcr.io/hassio-addons/ssh/amd64	15.0.8	d6d5709fb46f	2 weeks ago	303MB
ghcr.io/hassio-addons/vscode/amd64	5.10.2	6810ca970e64	2 weeks ago	921MB
homeassistant/amd64-addon-mosquitto	6.3.1	973c50be2983	3 weeks ago	209MB
ghcr.io/esphome/esphome-hassio	2023.8.3	caf2a5393e4d	3 weeks ago	348MB
zigbee2mqtt/zigbee2mqtt-amd64	1.33.0-1	a0d3178a6a35	3 weeks ago	170MB

List Images

List all images



```
</> $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ghcr.io/hassio-addons/jupyterlab/amd64	0.12.0	0b264c1f2404	13 days ago	1.03GB
ghcr.io/home-assistant/amd64-hassio-supervisor	2023.09.2	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/amd64-hassio-supervisor	latest	ecaec5ad7cd7	2 weeks ago	363MB
ghcr.io/home-assistant/qemu86-64-homeassistant	2023.9.2	e65d980c0e5b	2 weeks ago	1.81GB
ghcr.io/hassio-addons/node-red/amd64	14.5.0	2bdcfadcf154	2 weeks ago	432MB
ghcr.io/hassio-addons/ssh/amd64	15.0.8	d6d5709fb46f	2 weeks ago	303MB
ghcr.io/hassio-addons/vscode/amd64	5.10.2	6810ca970e64	2 weeks ago	921MB
homeassistant/amd64-addon-mosquitto	6.3.1	973c50be2983	3 weeks ago	209MB
ghcr.io/esphome/esphome-hassio	2023.8.3	caf2a5393e4d	3 weeks ago	348MB
zigbee2mqtt/zigbee2mqtt-amd64	1.33.0-1	a0d3178a6a35	3 weeks ago	170MB

List Unused Images

List Unused images

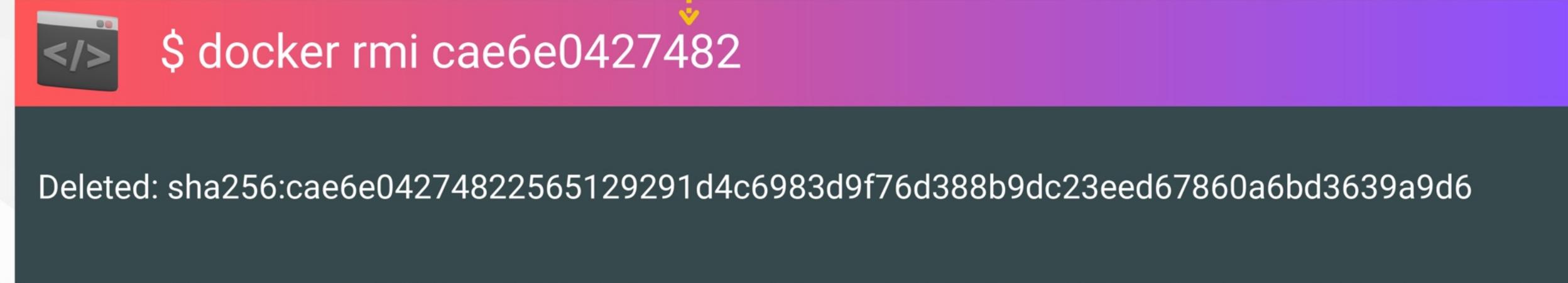


```
$ docker images -f dangling=true
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	cae6e0427482	6 weeks ago	14.6MB
<none>	<none>	62efc2d02d54	6 weeks ago	13.4MB
<none>	<none>	eela50e5beea	8 weeks ago	1.89GB
<none>	<none>	78574c30c497	8 weeks ago	1.89GB

Remove Images

Delete Image



```
</> $ docker rmi cae6e0427482
Deleted: sha256:cae6e04274822565129291d4c6983d9f76d388b9dc23eed67860a6bd3639a9d6
```

Image ID to remove

Try to run Database

Create Database Container

Create a db container



```
$ docker run -d -v wp_db-data:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=passroot \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=wordpress \
--name wpdb --net wp-net mysql:5.7
```

5ed3a8c12c0cee025cc4c1dede687f398af9ce51b3f7382d7b2bce9c98534437



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5ed3a8c12c0c	mysql:5.7	"docker-entrypoint.s..."	2 seconds ago	Up 1 second	3306/tcp	wpdb

Create Database Container

Create a db container

```
</> $ docker run -d -v wp_db-data:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=passroot \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=wordpress \
--name wpdb --net wp-net mysql:5.7
```

5ed3a8c12c0cee025cc4c1dede687f398af9ce51b3f7382d7b2bce9c98534437

Volume Mapping

```
</> $ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5ed3a8c12c0c	mysql:5.7	"docker-entrypoint.s..."	2 seconds ago	Up 1 second	3306/tcp	wpdb

Create Database Container

Create a db container



```
$ docker run -d -v wp_db-data:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=passroot \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=wordpress \
-e MYSQL_PASSWORD=wordpress \
--name wpdb --net wp-net mysql:5.7
```



5ed3a8c12c0cee025cc4c1dede687f398af9ce51b3f7382d7b2bce9c98534437
MySQL Image..... MySQL Image



```
$ docker ps
```

CONTAINER ID NAMES	IMAGE	COMMAND	CREATED	STATUS	PORTS
5ed3a8c12c0c wpdb	mysql:5.7	"docker-entrypoint.s..."	2 seconds ago	Up 1 second	3306/tcp

Let's try creating Wordpress.



Create wordpress with docker-compose



\$ vi docker-compose.yml

```
version: '3.7'
services:
  wordpress:
    image: wordpress:latest
    volumes:
      - ./public_html:/var/www/html
    ports:
      - "8088:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
      TZ: "Asia/Bangkok"
    networks:
      - backend
```

Reference:

<https://github.com/racksync/wordpress-docker>

Create wordpress with docker-compose



```
$ docker-compose up -d
```

```
Creating network "sites_default" with the default driver  
Creating volume "sites_wp_db-data" with  
default driver  
Creating sites_wordpress_1 ... done  
Creating sites_db_1 ... done
```



```
$ docker-compose ps
```

Name	Command	State	Ports
sites_db_1	docker-entrypoint.sh mysqld	Up	3306/tcp
apach ...	0.0.0.0:80->80/tcp	Up	sites_wordpress_1 docker-entrypoint.sh

Build your App onto Docker Images

How to Build Docker Image

Dockerfile instructions



Dockerfile

```
FROM httpd:2.4

COPY . /usr/local/apache2/htdocs

WORKDIR /usr/local/apache2/htdocs

EXPOSE 80

ENV HOST=0.0.0.0

ENV PORT=80
```

Reference:

<https://github.com/racksync/devops-workshop/blob/main/html/Dockerfile>

Let's Build first image

Running Dockerfile to Build Your Image



```
$ docker build -t your_dockerhub_username/your_image_name:your_tag .
```

```
[+] Building 1.0s (3/3) FINISHED
docker:desktop-linux
=> [internal] load .dockerignore
0.0s
=> => transferring context: 2B
0.0s
=> [internal] load build definition from Dockerfile
0.0s
=> => transferring dockerfile: 157B
.....
```

Build Example

Image Build Example



```
$ docker build -t ckalpha/apache2:latest .  
  
[+] Building 8.6s (9/9) FINISHED  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 157B  
=> [internal] load metadata for docker.io/library/httpd:2.4  
=> [auth] library/httpd:pull token for registry-1.docker.io  
=> [internal] load build context  
=> => transferring context: 1.09kB  
=> [1/3] FROM docker.io/library/httpd:2.4@sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f  
=> => resolve docker.io/library/httpd:2.4@sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f  
=> => sha256:9a10d47f6b0712b4d85ec8b4e01643225c12064aa1e551dd5fba7af5ca92c91a 4.01MB / 4.01MB  
=> => sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f 1.86kB / 1.86kB  
=> => sha256:865a19d7e91047406d8a2f435030fd5ca1bba85f86d05f7da679b63aabea3754 1.37kB / 1.37kB  
=> => sha256:ce6083df2933e4b2da1a320f39fc84e39f430115f7b5e6e52b83a0924d711a58 9.39kB / 9.39kB  
=> => sha256:e886f0f47ef56fcadb6ecaf2116056bbdb273e0afe07ed034498b198d386c04e 29.16MB / 29.16MB  
=> => sha256:d738112ed1f386aec91c06a1f24c2268aa1810f1453489339ccefc0fea3e3a 176B / 176B  
=> => sha256:d301d23958b1c363661cc4020f24765595569c0bc150ae40e338ae0ef8dfdf5f 292B / 292B  
=> => sha256:33fb7e051279ba05f301c48466186a40e49ba2275a66b0ed150c8fc190e39fa0 30.33MB / 30.33MB
```

Use your image

Running your image



```
$ docker run -d --name apache2 ckalpha/apache2
```

```
79ab6b1b1bdcd4df647bd0ad12727f9d00e00a39f415ce8b1cf55a1e0a9d8e73
```

Build PHP Server

How to Build Docker Image

Dockerfile instructions



Dockerfile

```
FROM php:8.0-apache  
  
WORKDIR /var/www/html  
  
COPY ./src/index.php .  
  
COPY ./src/landing.html .  
  
EXPOSE 80
```

Reference:

<https://github.com/racksync/devops-workshop/blob/main/php/Dockerfile>

Let's Build PHP Image

Running Dockerfile to Build Your Image



```
$ docker build -t username/your_image_name:your_tag .
```

```
[+] Building 1.0s (3/3) FINISHED
docker:desktop-linux
=> [internal] load .dockerignore
0.0s
=> => transferring context: 2B
0.0s
=> [internal] load build definition from Dockerfile
0.0s
=> => transferring dockerfile: 157B
.....
```

Build Example

Image Build Example



```
$ docker build -t ckalpha/php:latest .  
  
[+] Building 8.6s (9/9) FINISHED  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 157B  
=> [internal] load metadata for docker.io/library/httpd:2.4  
=> [auth] library/httpd:pull token for registry-1.docker.io  
=> [internal] load build context  
=> => transferring context: 1.09kB  
=> [1/3] FROM docker.io/library/httpd:2.4@sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f  
=> => resolve docker.io/library/httpd:2.4@sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f  
=> => sha256:9a10d47f6b0712b4d85ec8b4e01643225c12064aa1e551dd5fba7af5ca92c91a 4.01MB / 4.01MB  
=> => sha256:5123fb6e039b83a4319b668b4fe1ee04c4fb7c4c8d1d6ef843e8a943a9aed3f 1.86kB / 1.86kB  
=> => sha256:865a19d7e91047406d8a2f435030fd5ca1bba85f86d05f7da679b63aabea3754 1.37kB / 1.37kB  
=> => sha256:ce6083df2933e4b2da1a320f39fc84e39f430115f7b5e6e52b83a0924d711a58 9.39kB / 9.39kB  
=> => sha256:e886f0f47ef56fcadb6ecaf2116056bbdb273e0afe07ed034498b198d386c04e 29.16MB / 29.16MB  
=> => sha256:d738112ed1f386aec91c06a1f24c2268aa1810f1453489339ccefc0fea3e3a 176B / 176B  
=> => sha256:d301d23958b1c363661cc4020f24765595569c0bc150ae40e338ae0ef8dfdf5f 292B / 292B  
=> => sha256:33fb7e051279ba05f301c48466186a40e49ba2275a66b0ed150c8fc190e39fa0 30.33MB / 30.33MB
```

Use your image

Running your image with simple option



```
$ docker run -d --name php-app ckalpha/php
```

```
79ab6b1b1bdcd4df647bd0ad12727f9d00e00a39f415ce8b1cf55a1e0a9d8e73
```

Running your image with volume and ports mapping



```
$ docker run -d --name php -p 8088:80 -v D:\Docker\php\src:/var/www/html php
```

```
79ab6b1b1bdcd4df647bd0ad12727f9d00e00a39f415ce8b1cf55a1e0a9d8e73
```

Build React App

Build React App Image

Dockerfile instructions



Dockerfile

```
FROM node:alpine AS build
WORKDIR /app
COPY react-app/package*.json ./
RUN npm install
COPY react-app/..
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 8001
CMD ["nginx", "-g", "daemon off;"]
```

Reference:

<https://github.com/racksync/devops-workshop/blob/main/react/Dockerfile>

Let's Build React Image (without cache)

Running Dockerfile to Build Your Image



```
$ docker build --no-cache -t react:latest .
```

```
[+] Building 1.0s (3/3) FINISHED
docker:desktop-linux
=> [internal] load .dockerignore
0.0s
=> => transferring context: 2B
0.0s
=> [internal] load build definition from Dockerfile
0.0s
=> => transferring dockerfile: 157B
.....
```

Use React image

Running your image with ports mapping



```
$ docker run -d --name php -p 8088:8081
```

```
79ab6b1b1bdcd4df647bd0ad12727f9d00e00a39f415ce8b1cf55a1e0a9d8e73
```

Publish Images into Docker Hub

Login to Docker Hub

You need to login first



```
$ docker login  
Authenticating with existing credentials...  
...  
.....  
Login Succeeded
```

Publish Images

You need to login first



```
$ docker push your_dockerhub_username/your_image_name:your_tag
```

...

.....

Portainer

Install Portainer

<https://www.portainer.io/install>

The screenshot shows the Portainer.io Community Edition dashboard. On the left, a sidebar menu includes Home, primary (selected), Dashboard, App Templates, Stacks, Containers, Images, Networks, Volumes, Events, Host, and Settings. A notification at the bottom left indicates a new version available (2.19.1). The main area displays Environment info for the primary environment (standalone Docker 20.10.5+dfsg1, URL: /var/run/docker.sock, GPUs: none, Tags: -). Below this are summary cards for Stacks (3), Containers (6, with 6 running, 0 stopped, 0 healthy, 0 unhealthy), Images (7, 2.3 GB total), Volumes (4), and Networks (6).

Dashboard

portainer.io COMMUNITY EDITION

primary

Home

Dashboard

App Templates

Stacks

Containers

Images

Networks

Volumes

Events

Host

New version available 2.19.1

Dismiss See what's new

portainer.io Community Edition 2.18.4

Environment info

Environment primary 4 8.3 GB - Standalone 20.10.5+dfsg1

URL /var/run/docker.sock

GPUs none

Tags -

3 Stacks

6 Containers

6 running 0 stopped
0 healthy 0 unhealthy

7 Images 2.3 GB

4 Volumes

6 Networks

Docker in Cloud

Top Cloud Providers



Reference

Repositories & Reference

Exercise & Example

- DevOps Exercise - <https://github.com/racksync/devops-workshop>
- DB Server - <https://github.com/racksync/mariadb-pma-docker>
- WordPress - <https://github.com/racksync/wordpress-docker>
- Frigate - <https://github.com/racksync/frigate-docker>
- Cloudflare Dynamic DNS - <https://github.com/racksync/cloudflare-ddns-docker>
- Cloudflare Tunnel - <https://github.com/ckalpha/cloudflared-tunnel-docker>
- Plate Recognizer - <https://github.com/racksync/platerecognizer-docker>

Official

- Docker Images - <https://hub.docker.com>

Lesson & Articles

- Docker Images - <https://hub.docker.com>
- Docker Lesson - <https://github.com/smancke/docker-intro>
- Docker Lesson - <https://dockerlabs.collabnix.com/workshop/docker>

RACKSYNC COMPANY LIMITED

Address : 135/5 Moo.1, Khuntalae Sub-District, Muang District, Surat Thani, 84100

 devops@racksync.com

 <https://racksync.com>

 FB : <https://www.fb.com/racksync>

 Tel : 08 5880 8885

