RL4Eng

Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

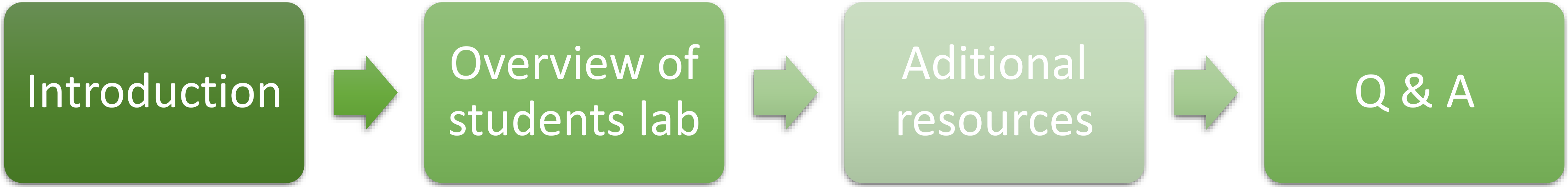
Carmen Carrión & Blanca Caminero

Introduction to Docker (staff version)

UCLM

Co-funded by the Erasmus+ Programme of the European Union

Contents



Introduction

Overview of students lab

Aditional resources

Q & A

Introduction

- Subject “*Planning and Integration of Systems and Services*”
- *Computer Science Degree, 4th year, Computing Engineering specialization*
- Syllabus:
    - BLOCK I: Introduction to Systems and Services Planning
    - BLOCK II: Integration of services in the corporate network
    - BLOCK III: Planning and management of the enterprise data center
    - BLOCK IV: Advanced aspects and emerging technologies

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Introduction

- BLOCK III: Planning and management of the enterprise data center
    - Unit 5. Introduction to capacity planning
    - Unit 6. Data Center network planning
    - Unit 7. Introduction to Cloud Computing

Introduction

- Lab sessions
  - Virtualization
    - Virtual machine deployment automation with VirtualBox and Vagrant [opensource tools]
    - Bare metal hypervisor: VMWare ESXi (emulated on VMWare Workstation) [trial versions for 30 days]
  - Introduction to Docker
  - Introduction to Kubernetes
    - Orchestation of aplications and services based on Docker containers
    - Kubernetes CLI (kubectl); minikube to emulate a Kubernetes cluster [opensource tools]

Introduction

- Lab sessions (cont.):
  - Load testing servers
    - Jmeter tool to emulate users [opensource]
    - Servers deployed with Docker and Vagrant
  - Introduction to AWS
    - EC2, VPC and IoT core services
    - AWS Academy suscription from UCLM

RL4Eng

Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

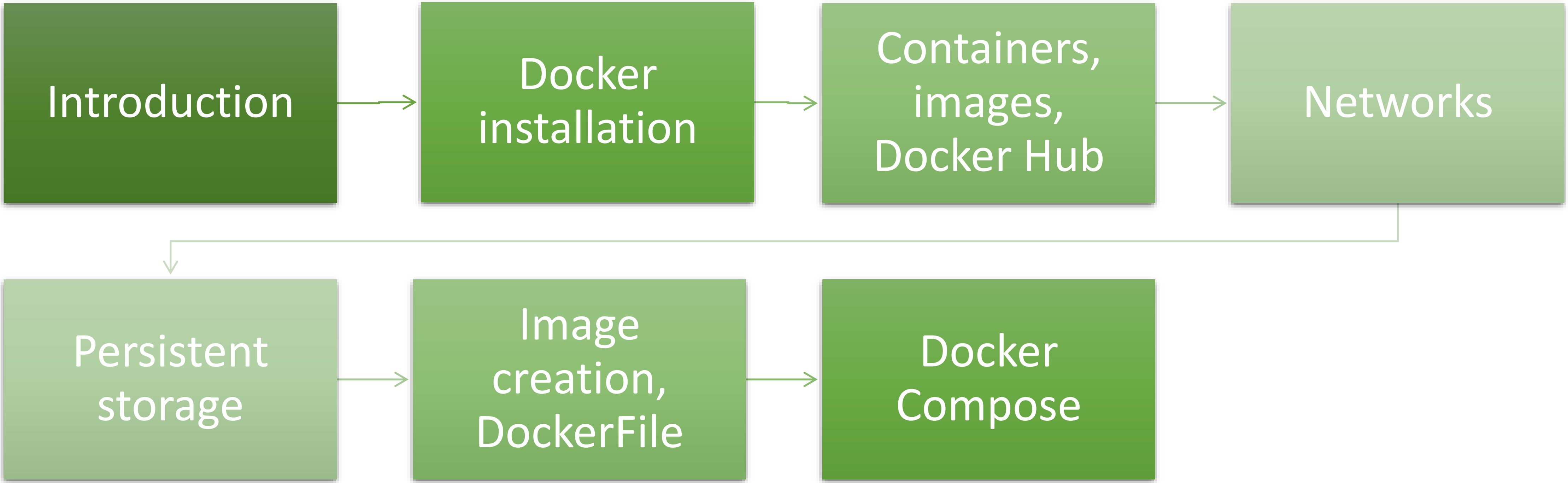
Carmen Carrión & Blanca Caminero

Introduction to Docker (students lab)

UCLM

Co-funded by the Erasmus+ Programme of the European Union

Contents



- Introduction
- Docker installation
- Containers, images, Docker Hub
- Networks
- Persistent storage
- Image creation, DockerFile
- Docker Compose

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

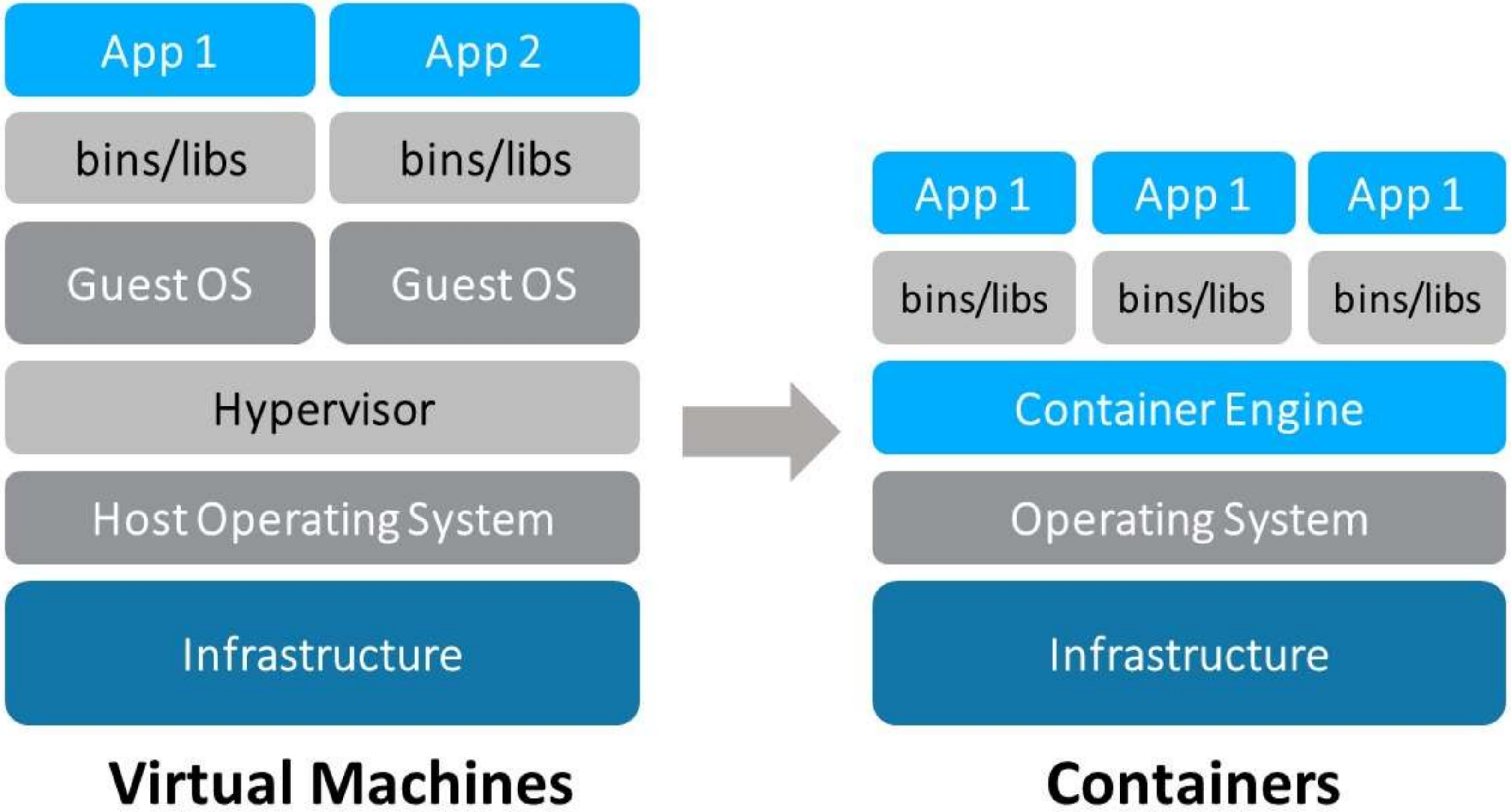
What is a container?

- A container is a lightweight, independent package that includes software and everything necessary for its execution (code, libraries, system tools, environment variables, ...)
- Characteristics
  - Portability
  - Low system overhead
  - Resource isolation
- They exploit features of the Linux kernel

RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

What is a container?



https://rancher.com/playing-catch-docker-containers/

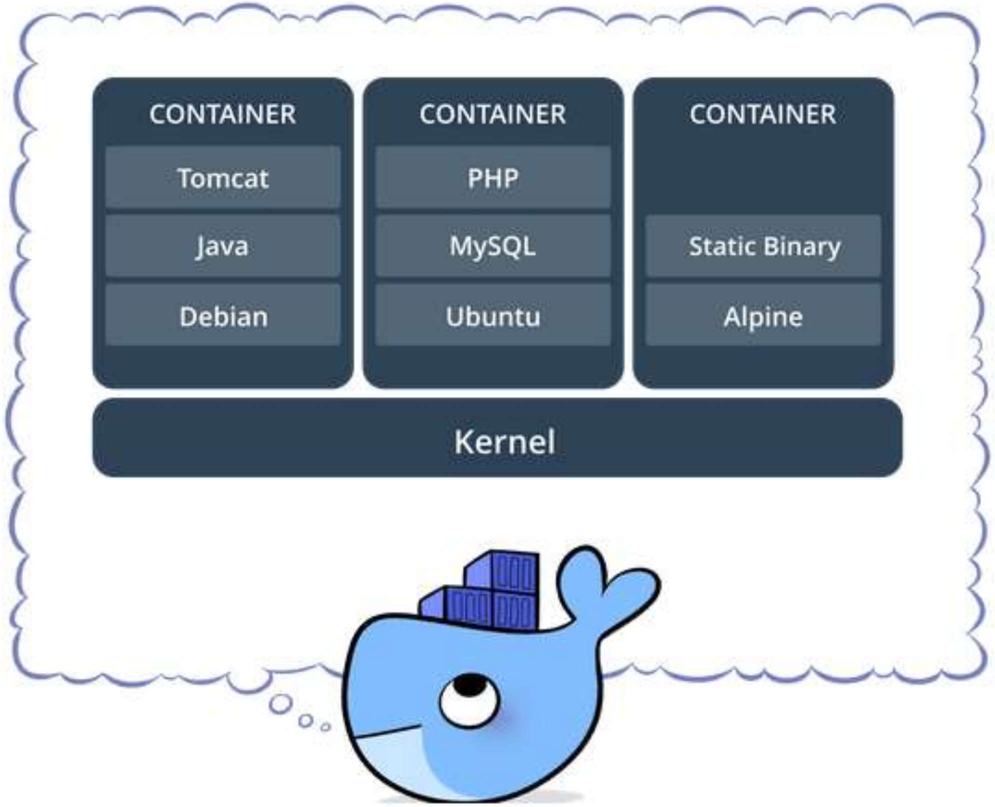
RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

What is Docker?

- Docker is the most widespread container support platform today.
- Free Software - Community Edition
  - Enterprise Edition
- Originally, only for Linux (it already supports Mac and Windows 10)
- Standardized, adopted by large companies
- Large catalog of images available

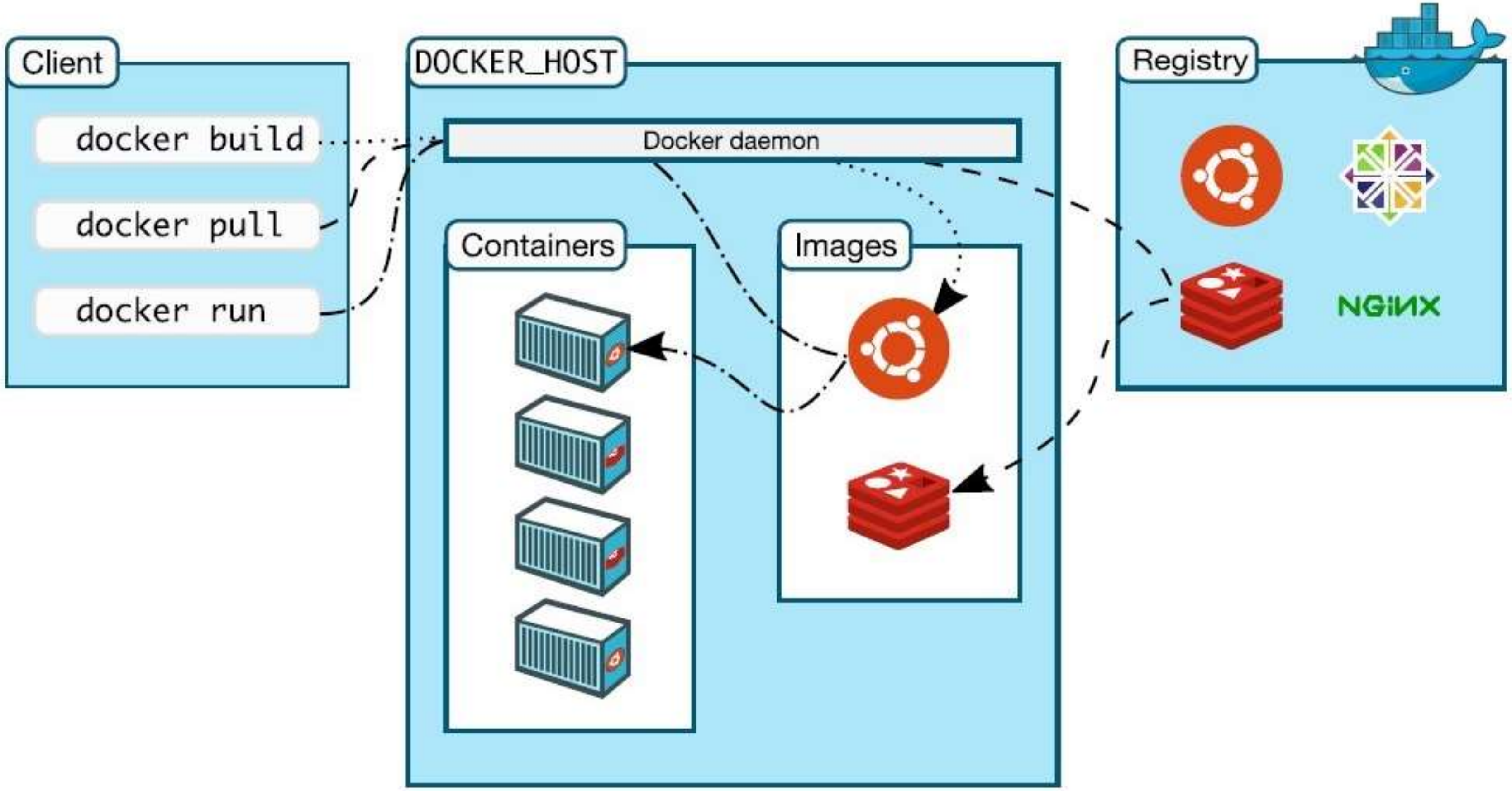
- + info : https://www.docker.com/



RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

What is Docker?



https://docs.docker.com/engine/docker-overview/#docker-architecture

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Time to get your hands dirty!

RL4Eng

Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Installation

- Ubuntu: There is a script that can be used to quickly install a version of Docker for development environments:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod –aG docker $USER
```

  - More detailed instructions at: https://docs.docker.com/engine/install/ubuntu/

- Verify:

```
docker version

docker info
```

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Containers

- My first container → hello-world

```
docker run hello-world
```

- Something more real (and interactive!)

```
docker run -it ubuntu
```

-t -> tty

-i -> interactive

- Checking containers in the system

```
docker ps
```

active

all

```
docker ps -a
```

Containers

```
docker ps
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c3163ab452cf ubuntu "/bin/bash" 9 seconds ago Up 8 seconds practical_ritchie
```

```
docker ps -a
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1bb94ae27aa9 hello-world "/hello“ 4 seconds ago Exited (0) 2 seconds ago wonderful_leavitt
C3163ab452cf ubuntu "/bin/bash“ About an hour ago Up About an hour practical_ritchie
```

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Containers

--rm → delete the container when finished

--name <name> → meaningful name

- Generic command

```
docker run --name <name> --rm -t -i <image> <command>
```

- Container management*

```
docker stop <container-id>
docker kill <container-id>
docker rm <container-id>
```

* You can use the container ID or its name

Containers

Exercise

- Launch an Ubuntu container from a terminal
  - Do it with a name so you can identify it easily
- Open another terminal
- Try deleting it (`rm`) from the new terminal
  - Is it possible? What should you do to be able to delete it?
- How can we delete all the containers that we have created and are no longer using?
  - Try `docker ps -aq` and mix it with `docker rm`

RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Solution

- Launch an Ubuntu container from a terminal

```
docker run -ti --name my_ubuntu ubuntu
```

- Open another terminal & try deleting it (`rm`) from the new terminal

```
docker rm my_ubuntu
```

(fails) *You need to stop the container first*

How can we delete all the containers that we have created and are no longer using?

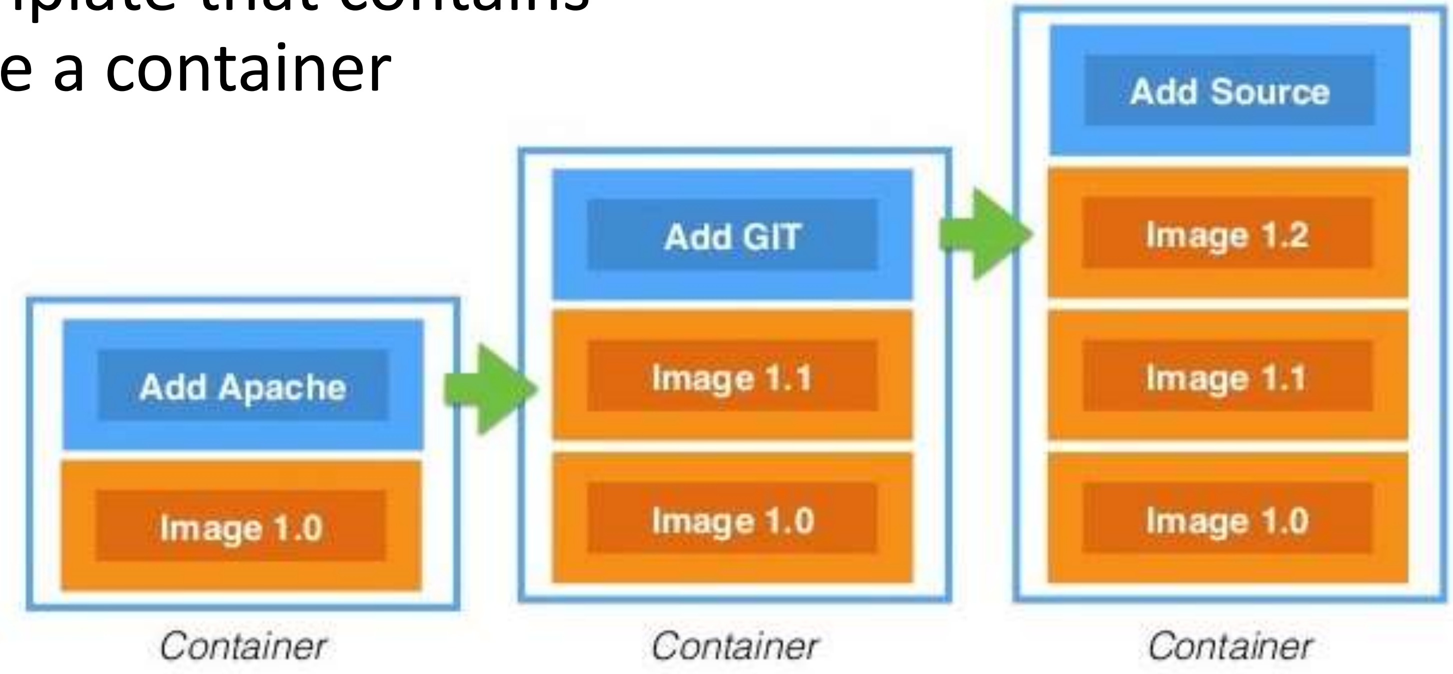
```
docker rm $(docker ps –aq)
```

if containers are not stopped, you should also kill them:

```
docker rm –-kill $(docker ps –aq)
```

Images

- An image is a read-only template that contains all the information to create a container
  - An image is usually based on another image, with some customization elements
    → "Layered" structure



RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

http://www.slideshare.net/FabioFerrari31/docker-containers-talk-linux-day-2015

Images

```
docker image ls
docker images
```

List images

```
docker rmi < image_id >
```

Delete image

Requirement: stopped container

How can I delete all the images at once?

[Hint: how did you do it with the containers?]

```
docker image history <image_id> _ _
```

See “layers”

RL4Eng

Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

DockerHub

Where does the first `hello-world` come from ? And `Ubuntu` ?

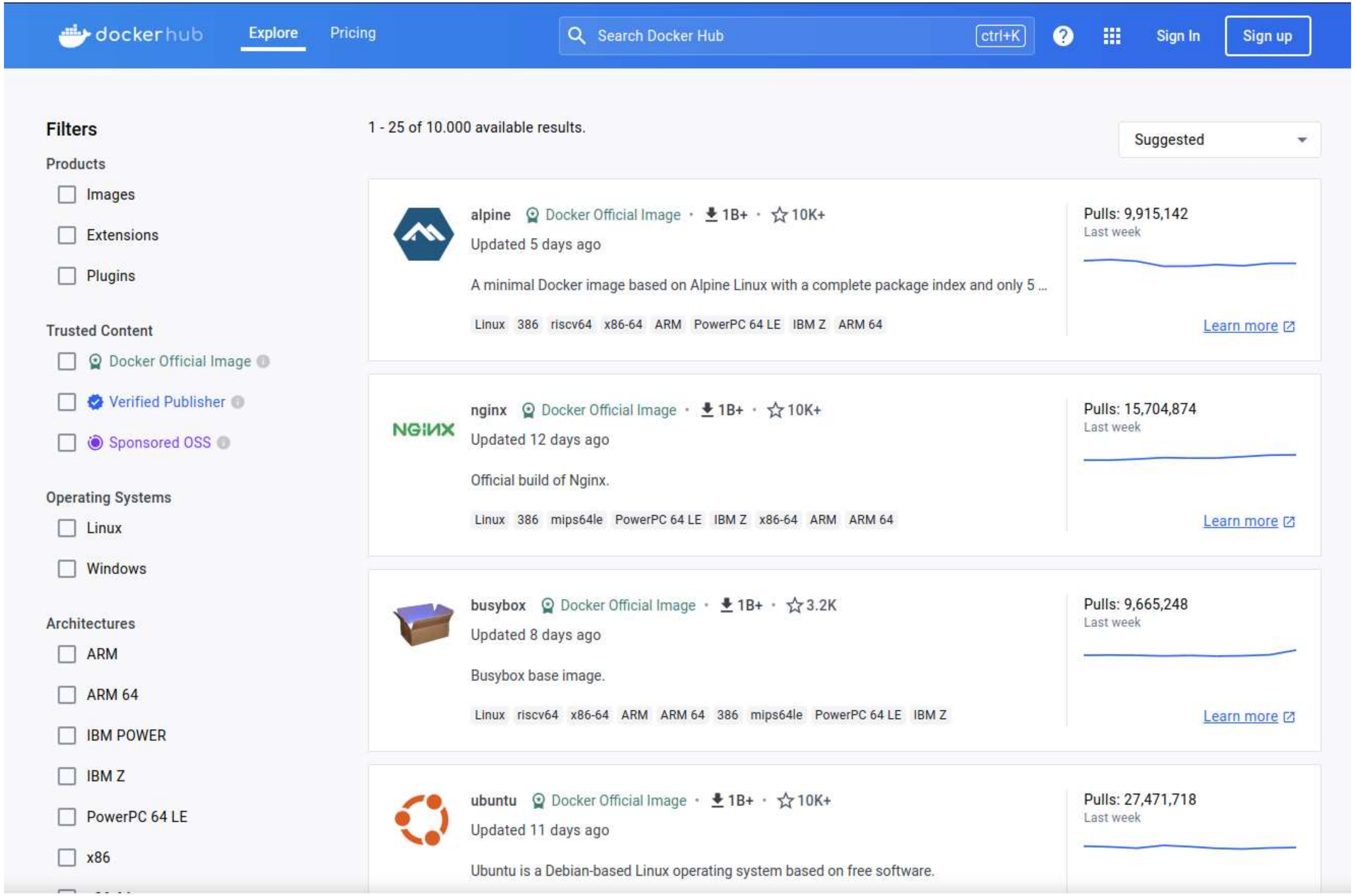
Where do the images come from?

- They can be downloaded from a registry → Docker Hub

https://hub.docker.com/

- You can create them yourself → `Dockerfile`
- You can upload your own images to the repository

DockerHub



Filters

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

1 - 25 of 10.000 available results.

Suggested

alpine Docker Official Image · 1B+ · ☆ 10K+
Updated 5 days ago
A minimal Docker image based on Alpine Linux with a complete package index and only 5 ...
Linux 386 riscv64 x86-64 ARM PowerPC 64 LE IBM Z ARM 64
Pulls: 9,915,142
Last week
Learn more

nginx Docker Official Image · 1B+ · ☆ 10K+
Updated 12 days ago
Official build of Nginx.
Linux 386 mips64le PowerPC 64 LE IBM Z x86-64 ARM ARM 64
Pulls: 15,704,874
Last week
Learn more

busybox Docker Official Image · 1B+ · ☆ 3.2K
Updated 8 days ago
Busybox base image.
Linux riscv64 x86-64 ARM ARM 64 386 mips64le PowerPC 64 LE IBM Z
Pulls: 9,665,248
Last week
Learn more

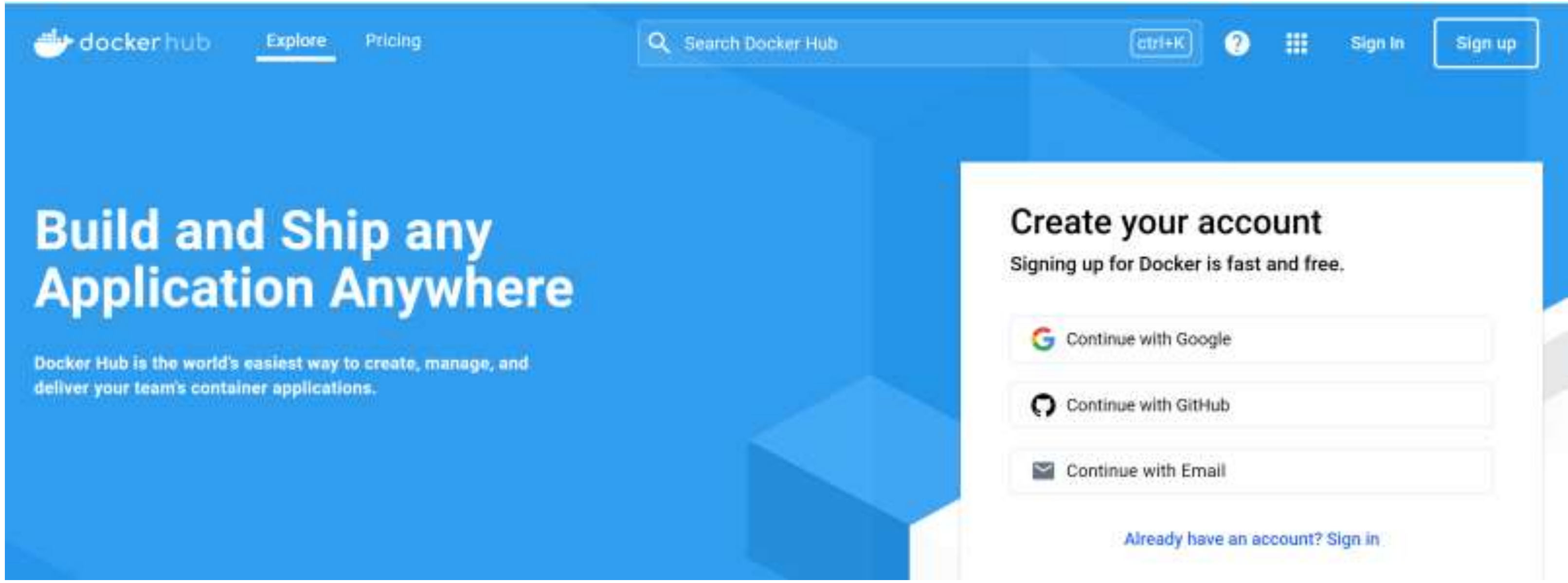
ubuntu Docker Official Image · 1B+ · ☆ 10K+
Updated 11 days ago
Ubuntu is a Debian-based Linux operating system based on free software.
Pulls: 27,471,718
Last week

DockerHub

- Docker Hub Registration

```
docker login
```

```
docker logout
```



Build and Ship any Application Anywhere

Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

Create your account

Signing up for Docker is fast and free.

https://hub.docker.com/

- Download/Upload images

```
docker pull
```

```
docker push
```

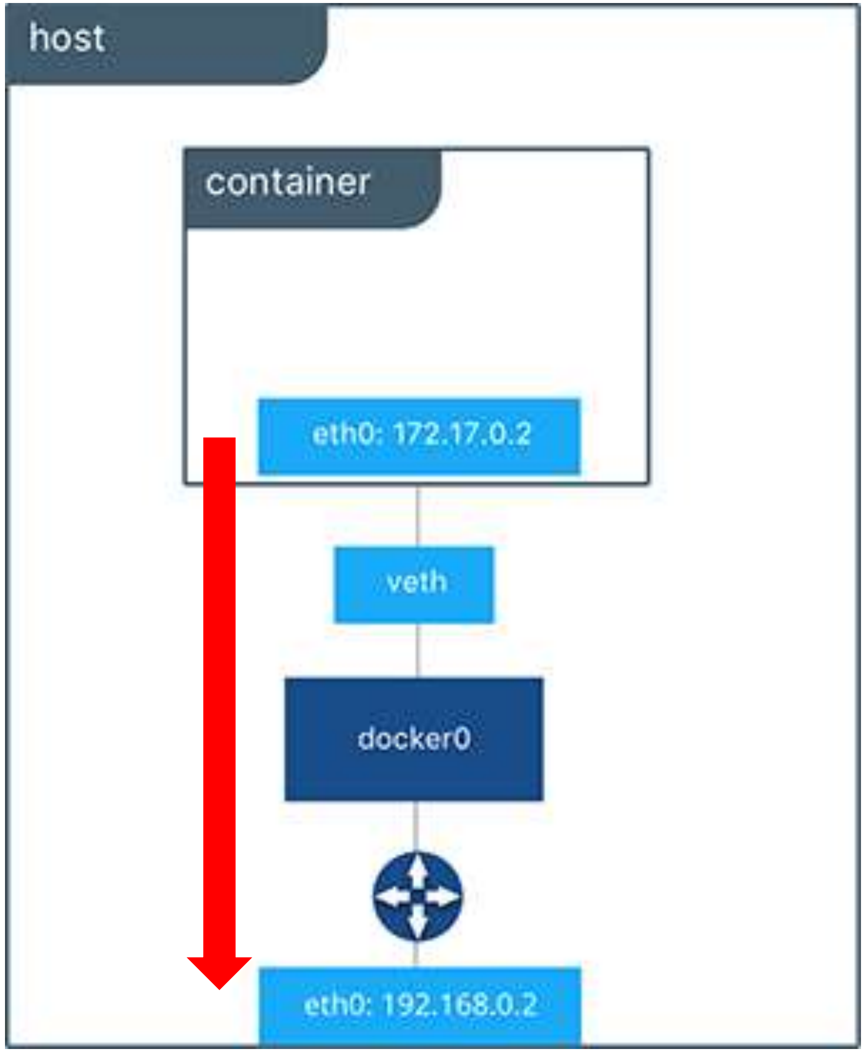
- Version control, tagging
- Image name → user-id/repository:tag

Networks

- Docker by default connects containers to a bridge type network (`docker0` interface)
- All containers have connectivity to the network the host connects to

Try running `ping 8.8.8.8` from a `busybox` container

```
docker run -it busybox
```



RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

busybox is a stripped-down image containing simplified versions of various Unix utilities

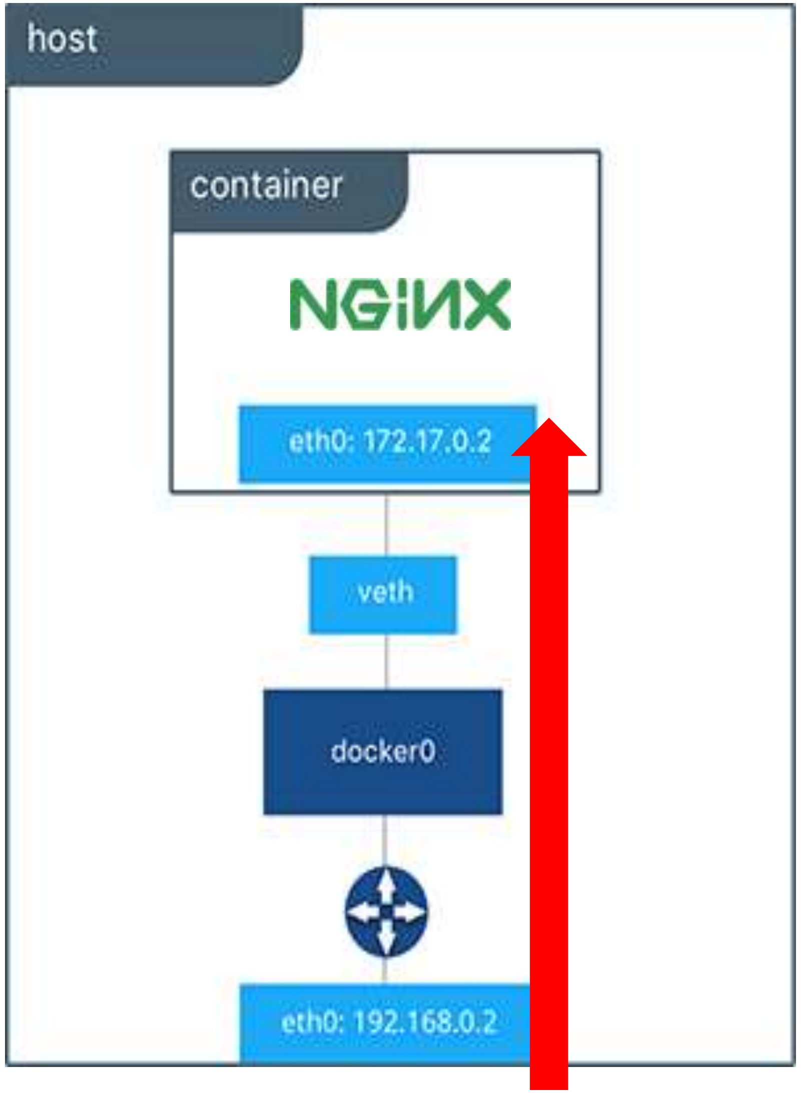
Networks

- But how do we access a service implemented in a container? Port binding
- Example: nginx web server

```
docker run -p 8080:80 -d nginx
```

- The web server (which by default uses port 80) will be accessible on our machine (localhost) through port 8080

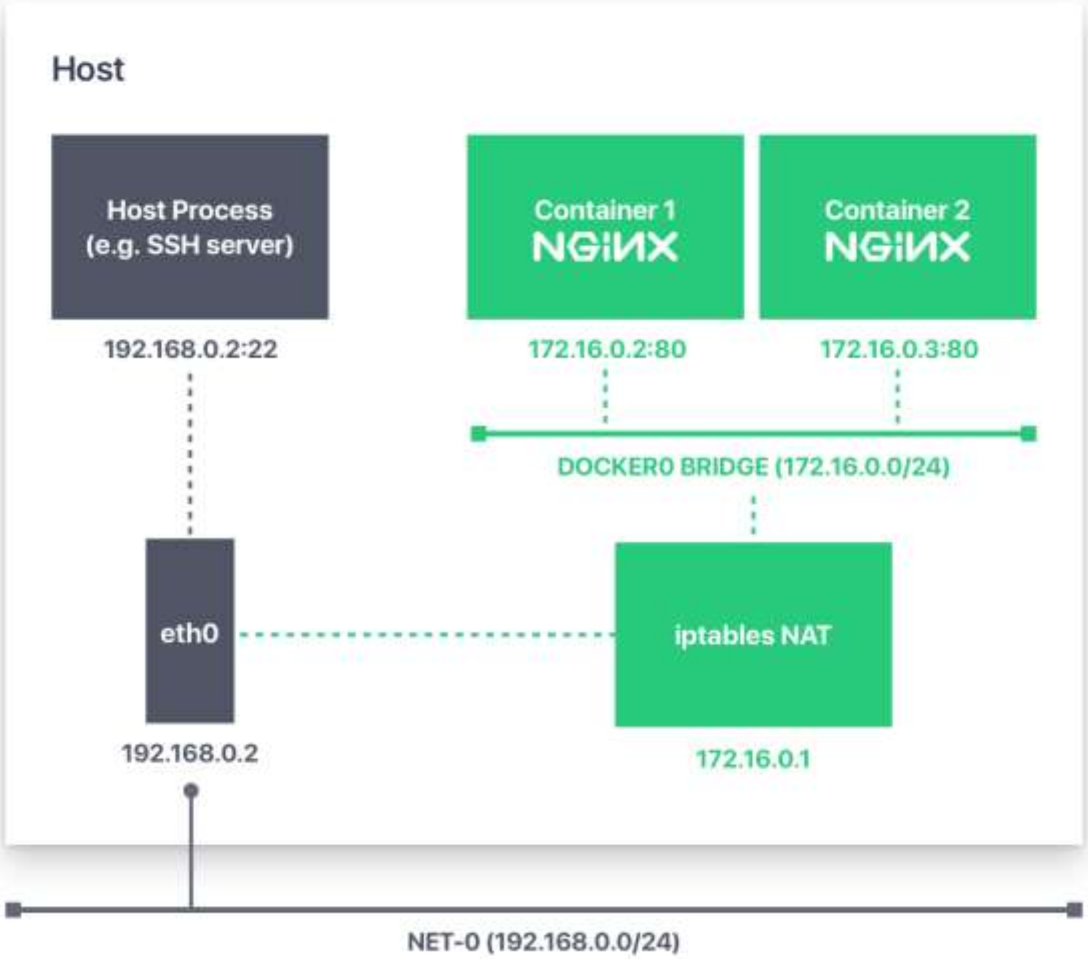
```
http://localhost:8080
```



RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Networks



https://mesosphere.com/blog/networking-docker-containers/

Networks

- To communicate containers with each other, we can create internal Docker networks, which also include an *embedded DNS server*
  - The docker network has similar attributes to a physical network, allowing containers greater flexibility when connecting and disconnecting
- Create a network: `docker network create <network-name>`
- List available networks: `docker network ls`
- Connect a container to a network when creating it:

```
docker run … --net=<network-name> …
```

Networks - example

- Create a network and connect a container to it:

```
docker network create backend-network
docker run -d --name=miredis --net=backend-network redis
```

- Create a second container connected to that network, and communicate with the first:

`alpine` is another minimal image based on the Alpine Linux OS, more complete than busybox

```
docker run --net=backend-network alpine ping -c1 miredis
```

- The Docker network contains a built-in DNS server, at IP 127.0.0.11:

```
docker run --net=backend-network alpine cat /etc/resolv.conf
```

- To view containers connected to a network:

```
docker network inspect backend-network
```

Co-funded by the
Erasmus+ Programme
of the European Union

Networks

- More possibilities :
  - It is possible to connect an existing container to a network:

    ```
    docker network connect <net_name> <container> _
    ```

  - Disconnection:

    ```
    docker network disconnect <net_name> <container>
    ```

  - Delete a network:

    ```
    docker network rm <net_name>
    ```

Persistent storage

- A container is immutable, so it cannot store data.
- How do we add persistence? One option is to share a directory between the host and the container (bind mount)

```
docker run -v <local_dir>:<cont_dir> <image>
```

- It is also possible to create volumes, memory spaces managed by Docker, that exist independently of containers

```
docker volume create <vol_name>
```

Persistent storage: bind mount

- Example: Customize the initial web page of the nginx web server
  - Local directory: $(pwd)/myweb *(it contains an index.html file)*
  - Container directory: /usr/share/nginx/html
  - Do not forget to expose (publish) the port!!

```
docker run -v $(pwd)/myweb:/usr/share/nginx/html -p 8080:80 -d nginx
```

The index.html file is on the host, the container also sees it as "modified" (it is the same file). Check it by reloading the web page in your browser.

RL4Eng
Development of Remote and Virtual
Laboratories for Teaching and Training
Engineering Students in the South
Mediterranean and Sub-Saharan Higher
Education Institutions

Co-funded by the
Erasmus+ Programme
of the European Union

Persistent storage: bind mount

Exercise

Start up another nginx server, with a different initial web page. It must run at the same time as the previous one

- Local directory: ???
- Container directory: ???
- Home port: ???
- Container port: ???

Solution

- Start another nginx server , with a different initial web page, at the same time as the previous one
- Local directory: [copy the myweb folder with another name (otherweb, for example) and modify the index.html]
- Container directory: /usr/share/nginx/html
- Home port: 8081 [cannot be the same as the previous one]
- Container port: 80

```
docker run -v $(pwd)/otherweb:/usr/share/nginx/html -p 8081:80 -d nginx
```

Persistent storage: Volumes

- Volumes exist independently of containers
  - They can be created, listed, inspected, destroyed...

```
docker volume create <vol_name>
docker volume ls
docker volume inspect <vol_name>
docker volume rm < vol_name >
```

  - They can be mounted in one or more containers

```
docker run -v <vol_name>:<mount_point> … <image>
```

Persistent storage: Volumes

Exercise

- Create a volume named “DATA_VOL”
- Create an *nginx* container, similar to the previous ones
  - Now the volume should be mounted to the directory where *nginx* stores web content
- Create another interactive container based on Alpine, which mounts the previous volume in the “/data” directory
  - Within the command line of this container modify the content of the index.html file
    - For example: echo “HELLO WORLD FROM THIS CONTAINER” > index.html
- Observe how when accessing the web server the content modified by the second container in the shared volume is displayed

Co-funded by the Erasmus+ Programme of the European Union

Solution

- 1. Create a Volume: *docker volume create DATA_VOL*
- 2. Create a Nginx Container:

docker run -d -p 8080:80 --name nginx-container -v DATA_VOL:/usr/share/nginx/html nginx

- 3. Create an Alpine Container:

docker run -it --name alpine-container -v DATA_VOL:/data alpine

  - - interactive mode (`-it`).

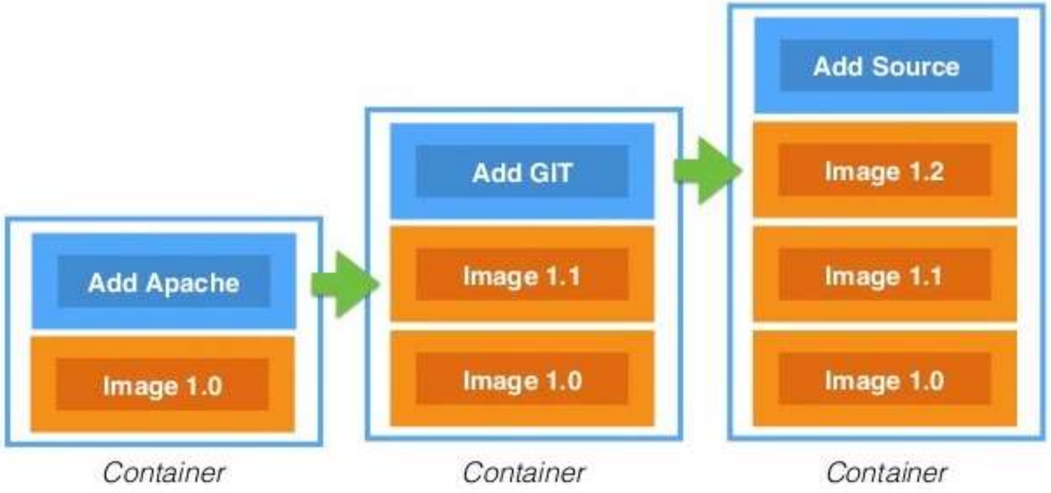
- 4. Modify the content of `index.html` in Alpine Container:

echo "HELLO WORLD FROM THIS CONTAINER" > /data/index.html

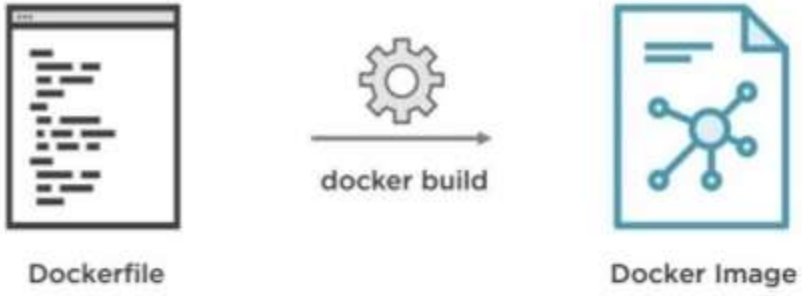
- 5. Access the Web Server: http://localhost:8080. See the modified content from Alpine

Creating images

- An image contains a series of layers
  - Layers are stored in a cache to reuse them
  - Layers can be shared between images and containers



- We can create our own images, starting from a base image and with the customization we need
  - Installing packages, copying files to it, running configuration files...
  - Image creation recipe → Dockerfile



```
docker build -t <dockerhub-user>/<image-name>:<tag> <dir>
```

Creating images

- Dockerfile example :

```
FROM ubuntu

LABEL maintainer="abcd"

RUN apt-get update && \
    apt-get install -y apache2 &&\
    apt-get clean

COPY index.html /var/www/html

EXPOSE 80

CMD apachectl -D FOREGROUND
```

File organization:

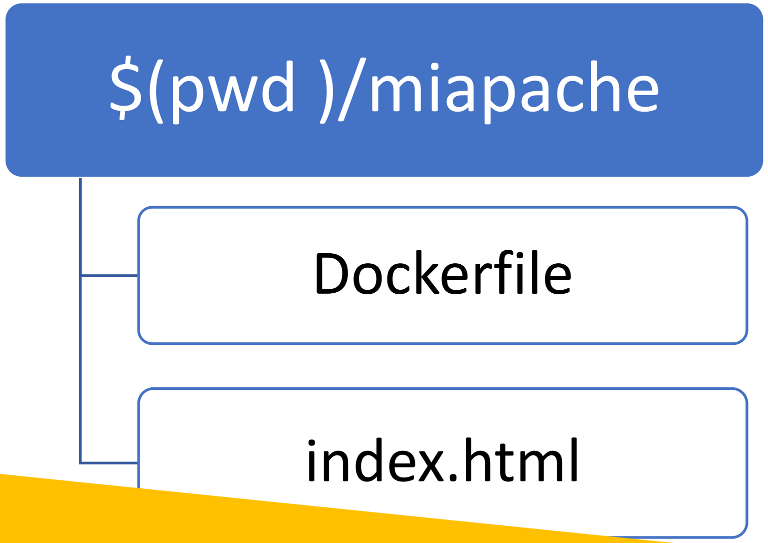


Image that includes an apache2 server on Ubuntu, plus my main website

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Creating images

- Building the image (from directory $(pwd)/myapache):

```
docker build -t <dockerhub-user>/myapache .
```

- We start the container:

```
docker run -p 8080:80 <dockerhub-user>/myapache
```

- We can also upload the image to Docker Hub*:

```
docker push <dockerhub-user>/myapache
```

*you must be previously logged in to Docker Hub (docker login)

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Creating images: Dockerfile commands

FROM	Adds a base image	**WORKDIR**	Changes the working directory for RUN, CMD, ENTRYPOINT, COPY, ADD
RUN	Runs a command inside the container	**ENV**	Environment variable declaration
COPY/ADD	Copy files from our host	**CMD**	Modifies the default command (only one per Dockerfile)
USER	Sets the user to use as the default from now on	**EXPOSE**	Specifies the available ports (it does not make a bind!)
LABEL	Image metadata, as a key-value pair	**ENTRYPOINT**	Sets the process that is executed when your container starts

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Source: https://github.com/cidaen/tutorial-docker

Creating images

Exercise

- Create image *mynginx*, with a static website from nginx container
  - Create a new directory
  - Copy the example Dockerfile and make the necessary changes
  - Copy the index.html and modify it
  - Create the image (docker build)
  - Run the new container
  - Access the web server through localhost
- See image layers and compare with the “myapache” image (docker image history <image>)

RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Solution

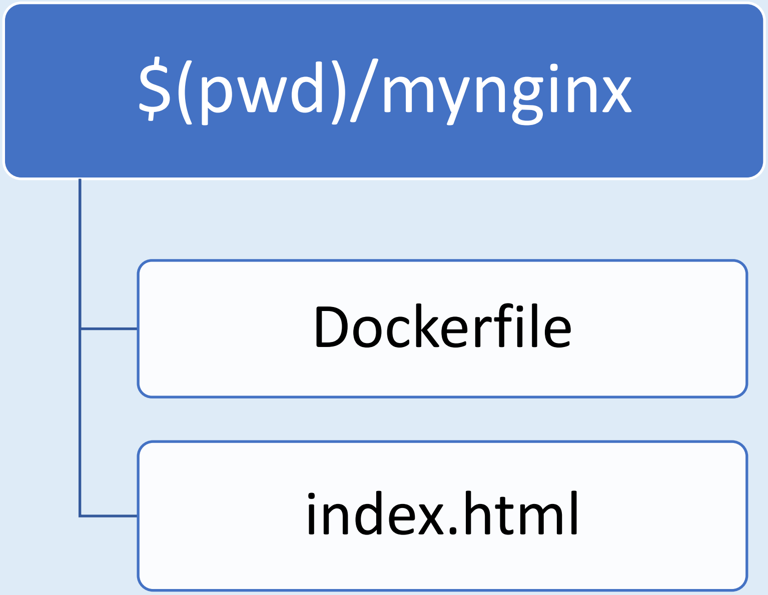
- Exercise create static website from nginx container

```
FROM nginx

LABEL maintainer =“abcd"

COPY index.html /usr/share/nginx/html

EXPOSE 80
```

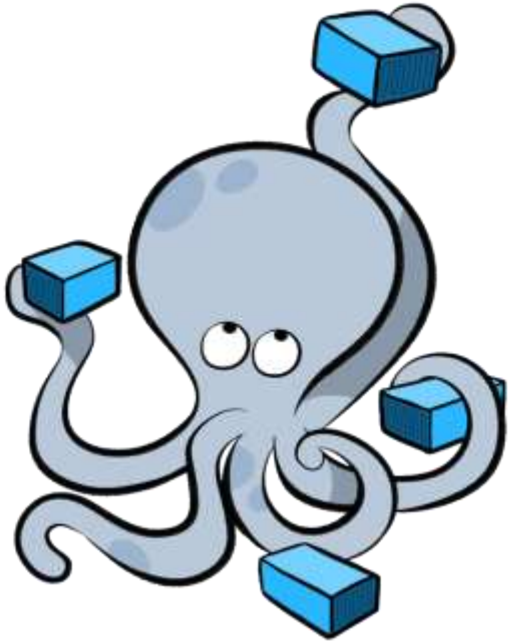


$(pwd)/mynginx

Dockerfile

index.html

Docker Compose



- In most cases it will be necessary to have several containers to deploy an application, along with volumes, networks, etc.
- Docker Compose is a tool that allows you to coordinate the execution of several containers more easily
- It is based on the use of the `docker-compose.yml` file where the containers and other elements to be executed are described
  - The format is based on YAML (Yet Another Markup Language)

```
container_name :
property: value
- or options
```

- Installation: https://docs.docker.com/compose/install/
- To check: `docker compose version`

RL4Eng
Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Co-funded by the Erasmus+ Programme of the European Union

Docker Compose

- A directory must be created for each project
- The `docker-compose.yml` file must be stored in that directory, along with the rest of the files that may be necessary (for example, to build an image, data files, etc ...)
- Basic commands (*they are executed in the same directory where the* *`docker-compose.yaml`* *file is*):
  - Start containers: `docker-compose up -d` (-d: to work decoupled from the terminal)
  - Stop containers: `docker-compose stop`
  - Delete containers and network: `docker-compose down`
  - Delete stopped containers: `docker-compose rm`

RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

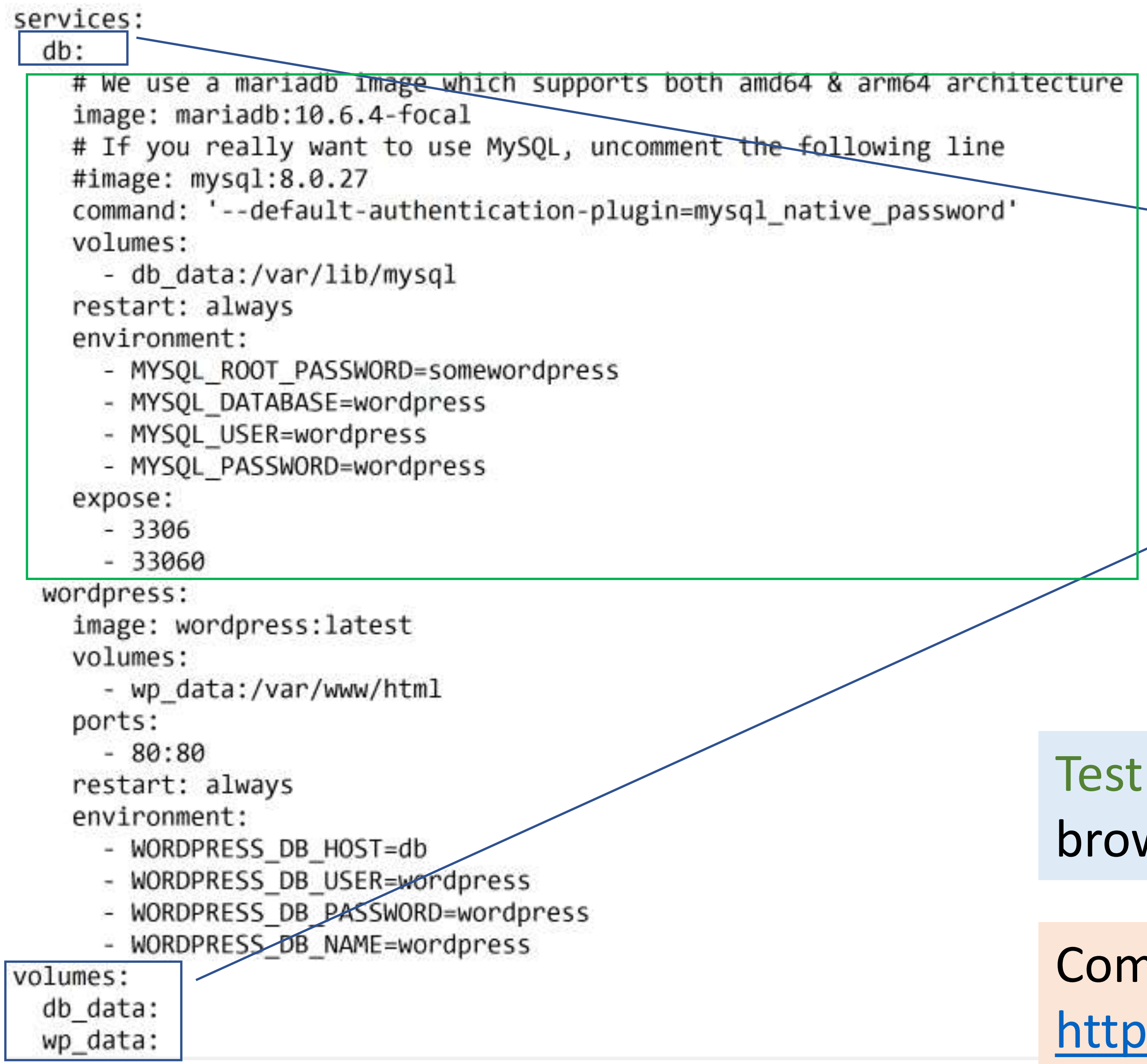
Co-funded by the Erasmus+ Programme of the European Union

Docker Compose

- We will use the example from https://github.com/docker/awesome-compose/tree/master/official-documentation-samples/wordpress/

- Project to launch a Wordpress server composed of:
  - A Wordpress container
  - A database container (mariadb)
  - Two volumes for data persistence of both

Docker Compose



```
services:
  db:
    # We use a mariadb image which supports both amd64 & arm64 architecture
    image: mariadb:10.6.4-focal
    # If you really want to use MySQL, uncomment the following line
    #image: mysql:8.0.27
    command: '--default-authentication-plugin=mysql_native_password'
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
      - MYSQL_PASSWORD=wordpress
    expose:
      - 3306
      - 33060
  wordpress:
    image: wordpress:latest
    volumes:
      - wp_data:/var/www/html
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=wordpress
      - WORDPRESS_DB_PASSWORD=wordpress
      - WORDPRESS_DB_NAME=wordpress
volumes:
  db_data:
  wp_data:
```

Container declaration

Setting up a container

Volume declaration

By default, it also creates a network where containers are connected

Test: after `docker-compose up -d`, open a browser on `localhost`

Complete command reference at: https://docs.docker.com/compose/compose-file/compose-file-v3/

RL4Eng

Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

That’s all, folks!!

UCLM

Co-funded by the
Erasmus+ Programme
of the European Union

Additional resources

- https://docs.docker.com/ → oficial documentation for the Docker Project
- https://training.play-with-docker.com/ → Docker labs and tutorials authored both by Docker, and by members of the community

- https://kodekloud.com/courses/docker-for-the-absolute-beginner/ → covers the basics, free enrollment
- https://container.training/ → recorded workshops and training materials

Q & A

RL4Eng Development of Remote and Virtual Laboratories for Teaching and Training Engineering Students in the South Mediterranean and Sub-Saharan Higher Education Institutions

Thanks for your attention!!!

UCLM

Co-funded by the Erasmus+ Programme of the European Union