WOMEN WHO CODE® /medellín

Containerization.

A guide to understanding and working with docker

WOMEN WHO
CODE®/medellín

Class 3

MAY 11th, 2024

About the last class

- Docker has five main components image, container, registry, demon and client
- Image = “package” & Container = “application running”
- DockerHub is an image as GitHub is a code
- DockerHub is the place where I can “save” public or private images
- A Docker container has states (lifecycle) and those states are given by Linux signals

GOALS

01 Warming up

02 Working with custom images & Dockerfile syntax

03 “Persist data on Docker”

04 Publish docker images

1. Warming up

It is time to remember past classes

WOMEN WHO CODE /medellín

Warming up

Complete the following activity:

1. Investigate how to create a Docker container from the official MySQL.
2. Create a MySQL instance on Docker and expose it on the port “**3307”**, use **“sebastian”** as default user and **“GreatTeacher”** as password.
3. Create a new Database with your first name and a table within it with your last name.
4. With the partner next to you. Establish a connection on their MySQL instance. Explain why it is possible locally. Tip: Be connected on the same Network

Bonus:

- Can you do the same with MongoDB?

MySQL Solution

```
# Run an expose MySQL instance

docker run --name mysql_instance \
-p 3307:3306 \
-e MYSQL_ROOT_PASSWORD=GreatTeacher \
-e MYSQL_USER=sebastian \
-e MYSQL_PASSWORD=GreatTeacher \
-e MYSQL_DATABASE=sebastian \
-d mysql:latest
```

https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/docker-mysql-more-topics.html

WOMEN WHO
CODE./medellín

Mongo DB Solution

```
# Run an expose MongoDB instance
docker run -d -p 27017:27017 --name mongodb_instance mongo:latest
--auth

# Execute a command inside the container
docker exec -it mongodb_instance mongo admin

# Create default user
docker      exec      -it      mongodb_instance      mongosh      admin

db.createUser({
  user: "sebastian",
  pwd: "GreatTeacher",
  roles: [{ role: "root", db: "admin" }]
})
```

WOMEN WHO
CODE./medellín

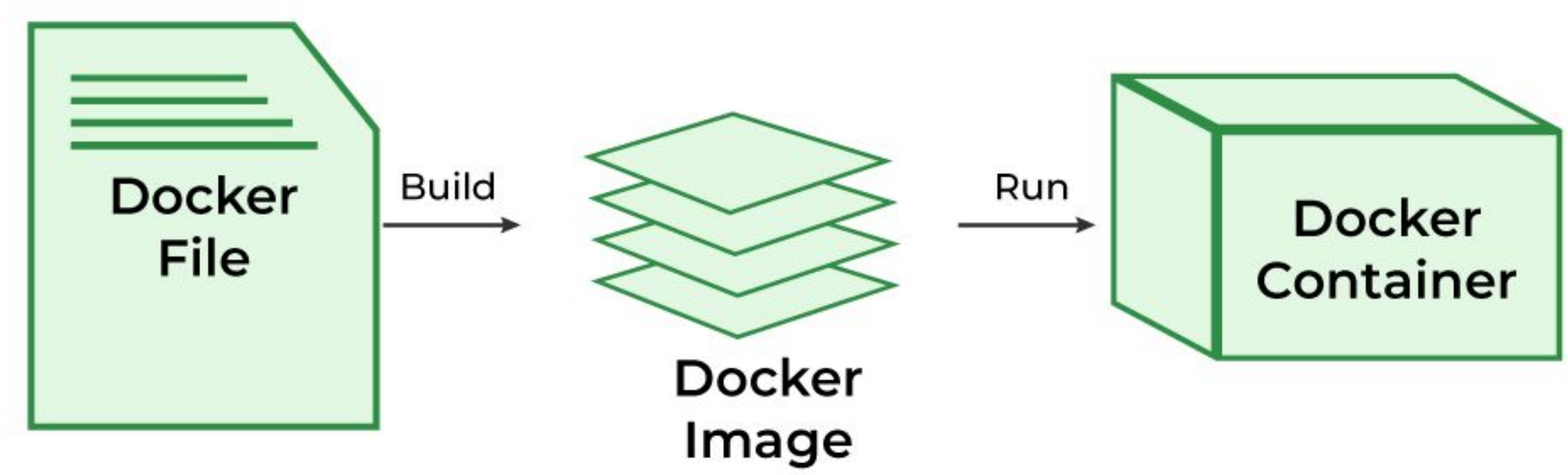
2. Working with custom images & Dockerfile syntax

WOMEN WHO CODE®/medellín

What is a Dockerfile?

A **Dockerfile** is a text file that contains instructions **for building** a Docker **image**, which is a **packaged version** of an application and its **dependencies**. These instructions include specifying the **base image**, installing **dependencies**, setting **environment variables,** and defining **startup commands**. With a Dockerfile, you can consistently build images for your applications, enabling easy deployment and execution in containerized environments

What is the new step?



Dockerfile format

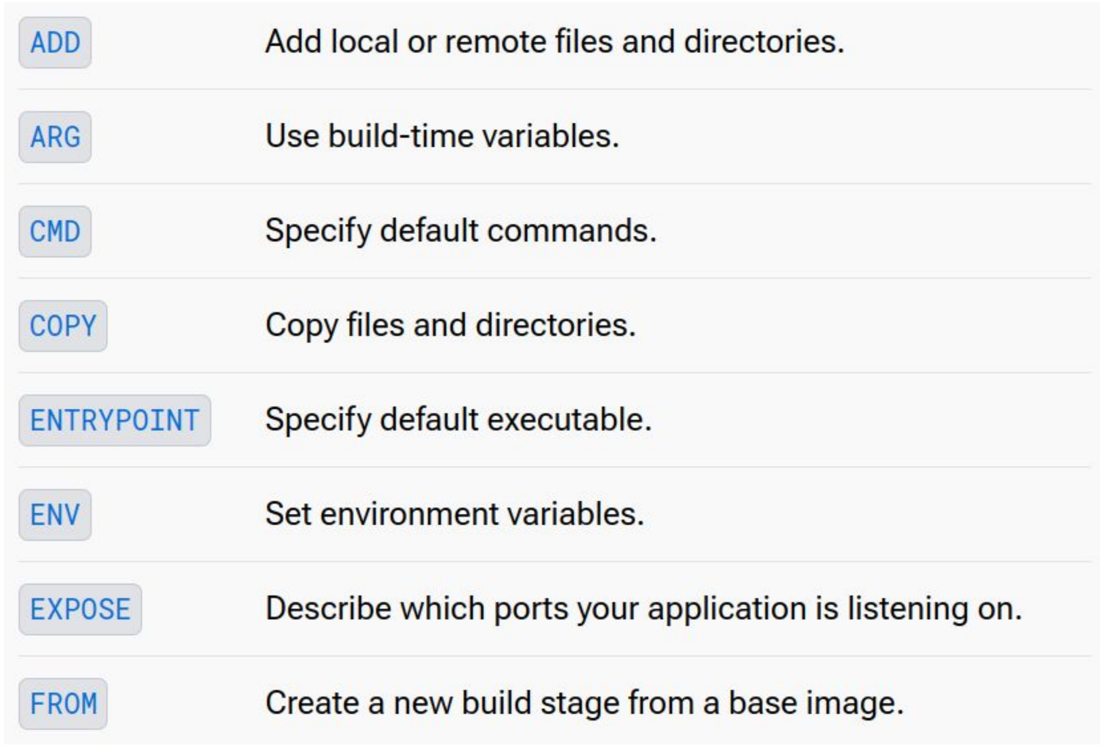
Dockerfile it is just a file on your project root folder

```
./my_project/
└── ./my_project/Dockerfile
```

Such as

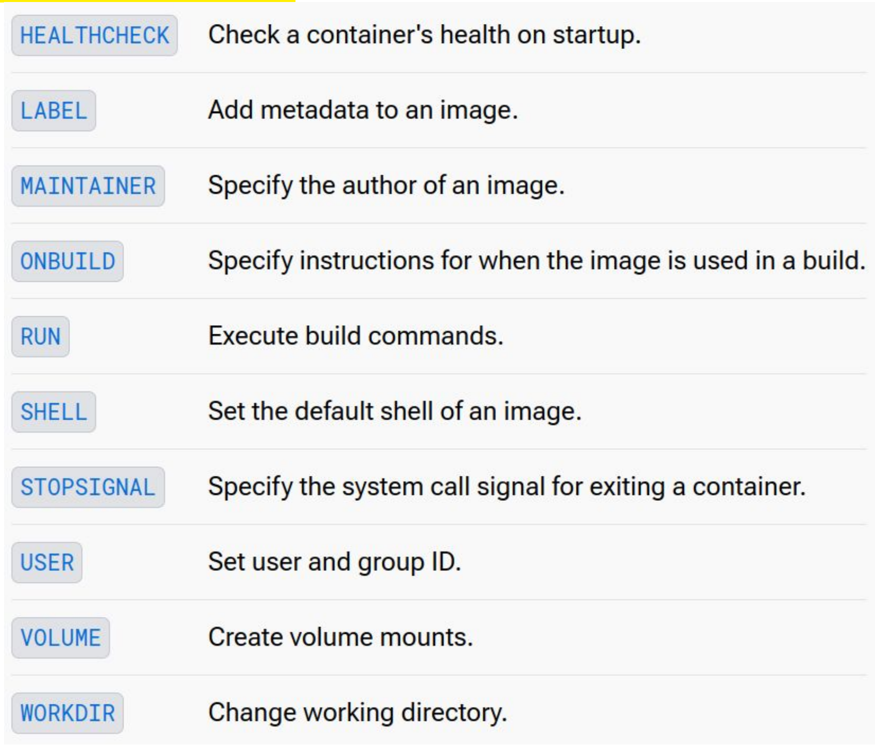
```
# Comment

INSTRUCTION arguments
```

Dockerfile syntax



`ADD`	Add local or remote files and directories.
`ARG`	Use build-time variables.
`CMD`	Specify default commands.
`COPY`	Copy files and directories.
`ENTRYPOINT`	Specify default executable.
`ENV`	Set environment variables.
`EXPOSE`	Describe which ports your application is listening on.
`FROM`	Create a new build stage from a base image.

Dockerfile syntax



`HEALTHCHECK`	Check a container's health on startup.
`LABEL`	Add metadata to an image.
`MAINTAINER`	Specify the author of an image.
`ONBUILD`	Specify instructions for when the image is used in a build.
`RUN`	Execute build commands.
`SHELL`	Set the default shell of an image.
`STOPSIGNAL`	Specify the system call signal for exiting a container.
`USER`	Set user and group ID.
`VOLUME`	Create volume mounts.
`WORKDIR`	Change working directory.

https://docs.docker.com/reference/dockerfile/

Examples

https://github.com/sebastianpinosanchez/docker-course

Docker ignore

You can use a .dockerignore file to exclude files or directories from the build context.

```
# .dockerignore

node_modules
bar
```

This helps avoid sending unwanted files and directories to the builder, improving build speed, especially when using a remote builder.

3. “Persist data on Docker”

Docker Volumes

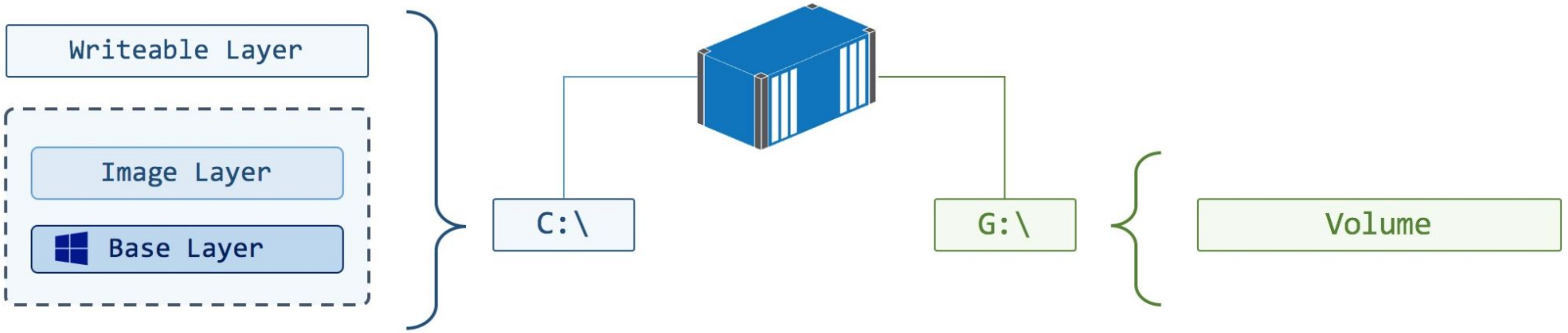
A Docker volume is a persistent data storage mechanism used by Docker containers to store and manage data independently of the container's lifecycle. Volumes provide a way to share data between containers, as well as persist data across container restarts or recreations.

Docker volumes are separate from the container's file system and can exist independently. They are managed by Docker and can be mounted into one or multiple containers simultaneously. Volumes can be used to store configuration files, databases, logs, or any other type of data that needs to persist beyond the lifetime of a container.

How can we use a volume

```
# Create a new volume
docker volume create mysql_data

# Start MySQL instance
docker run -d \
  --name mysql \
  -e MYSQL_ROOT_PASSWORD=your_password \
  -v mysql_data:/var/lib/mysql \
  mysql:latest
```

It looks like


Writeable Layer
Image Layer
Base Layer
C:\
G:\
Volume