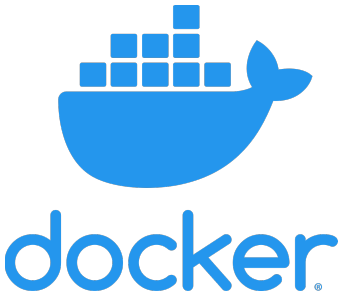


DOCKER - WORKSHOP 1

Created by Frédérick PERRIN

DOCKER "IN DEV" ?

BECAUSE:

BECAUSE:

- homogeneous installations

BECAUSE:

- homogeneous installations
- iso PROD env

BECAUSE:

- homogeneous installations
- iso PROD env
- easy onboarding

BECAUSE:

- homogeneous installations
- iso PROD env
- easy onboarding
- host machine stay clean

BECAUSE:

- homogeneous installations
- iso PROD env
- easy onboarding
- host machine stay clean
- easy to evolve / maintain

BECAUSE:

- homogeneous installations
- iso PROD env
- easy onboarding
- host machine stay clean
- easy to evolve / maintain
- can easily create POC

DOCKER, GET STARTED

QUICK INTRODUCTION

WHAT'S DOCKER ?

WHAT'S DOCKER ?

Docker is a platform for developers and sysadmins to build, run, and share applications with containers.

WHAT'S DOCKER ?

Docker is a platform for developers and sysadmins to build, run, and share applications with containers.

Based on **Linux** kernel and on one feature: containers (**LXC**). The goal is to run one process in an isolated environment

WHAT ABOUT VM'S ?

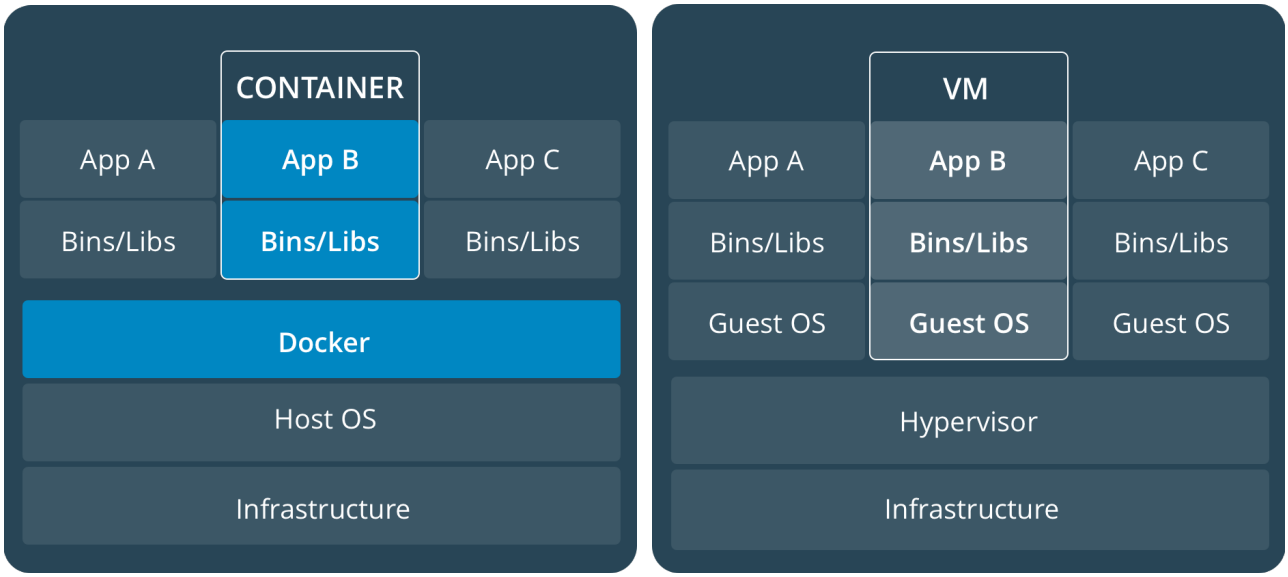
Everything works with vm's (isolation, abstraction layers, ...)

What the differences between vm and containers?

WHAT ABOUT VM'S ?

Everything works with vm's (isolation, abstraction layers, ...)

What the differences between vm and containers?



IMAGES VS CONTAINER

Fundamentally, a container is nothing but a running process

The image is the definition of a container, containers are just image's instances

you can run multiple instances (containers) of an image

WORKFLOW

WORKFLOW

Create /
Use image

WORKFLOW

Create /
Use image →

WORKFLOW

Create / Use image → Create a container

WORKFLOW

Create / Use image → Create a container →

WORKFLOW

Create / Use image → Create a container → Start container

BASIC COMMANDS AND CONCEPTS

BASIC COMMANDS AND CONCEPTS

docker's documentation is really well organized and clear

BASIC COMMANDS AND CONCEPTS

docker's documentation is really well organized and clear

Command line reference: https://docs.docker.com/engine/reference/commandline/docker/

COMMANDS TO KNOW 1/4

```
# The default docker images will show all top level images,
# their repository and tags, and their size.
docker images

# Manage images
docker image COMMAND

# Build an image from a Dockerfile
docker build [OPTIONS] PATH | URL | -

# Copy files/folders between a container
# and the local filesystem
docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH
```

COMMANDS TO KNOW 2/4

```
# Create a new container
docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

# Run a command in a running container
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

# Run a command in a new container
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

# Return low-level information on Docker objects
docker inspect [OPTIONS] NAME|ID [NAME|ID...]

# Fetch the logs of a container
docker logs [OPTIONS] CONTAINER

# List containers
```

COMMANDS TO KNOW 3/4

```
# Restart one or more containers
docker restart [OPTIONS] CONTAINER [CONTAINER...]

# Start one or more stopped containers
docker start [OPTIONS] CONTAINER [CONTAINER...]

# Stop one or more running containers
docker stop [OPTIONS] CONTAINER [CONTAINER...]

# Display the running processes of a container
docker top CONTAINER [ps OPTIONS]
```

COMMANDS TO KNOW 4/4

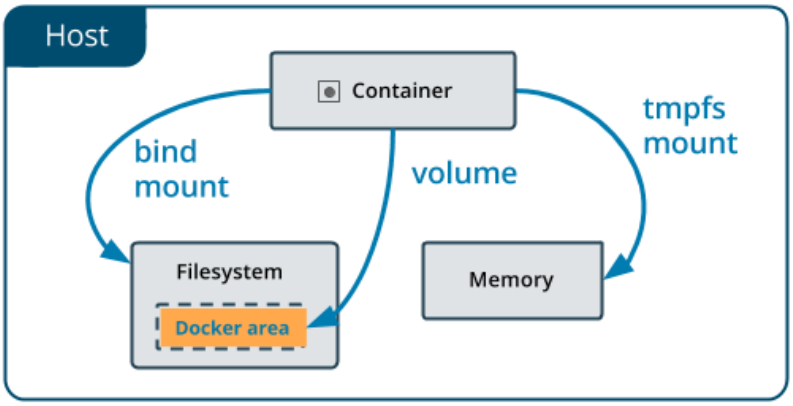
```
# Remove one or more containers
docker rm [OPTIONS] CONTAINER [CONTAINER...]

# Remove one or more images
docker rmi [OPTIONS] IMAGE [IMAGE...]
```

CONFIGURE CONTAINERS

VOLUMES

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers



Documentation: https://docs.docker.com/storage/volumes/

LINK / DEPENDENCIES

- container links (deprecated)
- networks (user defined networks, give a better isolation)

ENVIRONMENTS

you can define environment variables in image build and in container creation

It gives a powerfull tool to customize and configure your container

- embed configuration
- on run customization

NETWORKS

Use to connect containers between them. You can add networks, connect containers to them, isolate them.

Documentation: https://docs.docker.com/network/

DOCKERFILE

Docker can build images automatically by reading the instructions from a Dockerfile.

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image

Documentation: https://docs.docker.com/engine/reference/builder/

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4          maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

DOCKERFILE EXAMPLE

```
1  FROM debian:buster-slim
2
3  LABEL description="This an iad workshop on docker" \
4            maintainer="frederick.perrin@iadinternational.com"
5
6  RUN apt-get update && apt-get install -y procps
7
8  ENV COMPANY="docker"
9  COPY test.txt /mnt/
10 EXPOSE 80
11
12 ENTRYPOINT ["top", "-b"]
13 CMD ["-c"]
```

LET'S PLAY

DOCKER RUN

DOCKER RUN

```
# get image in local
docker pull debian:buster-slim
```

DOCKER RUN

```
# let's check our images
docker images
REPOSITORY   TAG          IMAGE ID       CREATED     SIZE
debian       buster-slim  f49666103347   8 days ago  69.2MB
```

DOCKER RUN

```
# your first container
docker run debian:buster-slim
```

DOCKER RUN

```
# show our containers
docker ps -a --filter ancestor=debian:buster-slim
```

DOCKER RUN

```
# clean those containers
docker rm ac3cc529dcfc e9d5dfdfc3f6
```

DOCKER RUN

```
# clean execution
docker run --rm debian:buster-slim
```

DOCKER RUN

```
# execute a command inside our
docker run --rm debian:buster-slim echo "I love docker !"
```

DOCKER RUN

```
# execute a shell
docker run --rm debian:buster-slim bash
```

DOCKER RUN

```
# -it flags attaches us to an interactive tty in the container.
docker run --rm -it debian:buster-slim bash
```

DOCKER RUN

```
# mount a volume
docker run -it -v $(pwd):/mnt --rm debian:buster-slim bash
```

DOCKER RUN

```
# use environment variable
docker run -e IAD=42 -v $(pwd):/mnt --rm debian:buster-slim env
```

DOCKER BUILD

DOCKER BUILD

```
# get Dockerfile from repo:
git clone git@github.com:yonzin/docker-workshop.git
```

DOCKER BUILD

```
# let's build our first image
docker build -t iad_workshop:1.0 -t iad_workshop:latest .

# got check our images
docker image ls
```

DOCKER BUILD

```
docker run -e IAD=42 -v $(pwd):/mnt --rm iad_workshop:1.0

docker ps test
```

DOCKER BUILD

```
docker run -e IAD=42 -v $(pwd):/mnt --rm iad_workshop -H

docker ps test
```

DOCKER BUILD

```
docker run -it --entrypoint bash --rm iad_workshop --debug

docker ps test

docker exec -ti test top
```

DOCKER BUILD

```
docker run -it --entrypoint bash --name test-workshop iad_workshop

docker start test-workshop
```

DOCKER BUILD

```
docker run -it --entrypoint bash --rm --name test iad_workshop

> ls -la /mnt
> cat /mnt/test.txt

docker run -it -v $(pwd):/mnt --entrypoint bash --rm iad_workshop

> ls -la /mnt
```

DOCKER-COMPOSE, KEZAKO ? AND WHY ?

DEFINITION

Compose is a tool for defining and running multi-container Docker applications.

With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

YES

This is IaC (infrastructure as code)

You can define your services, networks, volumes

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

```yaml
version: '3.8'
services:
  my-service:
        image: nginx
        container_name: my_awesome_service
        ports:
          - "8080:80"
        networks:
          - my-own-network
  my-second-service:
        build: ./my-second-service
        environment:
          - IAD=42
        volumes:
          - ./my-src:/home/service
        depends_on:
          - my-service
        networks:
          - my-own-network
```

Documentation: https://docs.docker.com/compose/compose-file/

DOCKER COMPOSE CLI

There is commands to manage docker compose files

Documentation: https://docs.docker.com/compose/reference/overview/

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

COMMANDS TO KNOW

```
# Services are built once and then tagged
docker-compose build

# Builds, (re)creates, starts, and attaches to containers for
docker-compose up

docker-compose start
docker-compose stop
docker-compose restart

# Removes stopped service containers.
docker-compose rm

# Stops containers and removes containers, networks,
# volumes, and images created by up.
docker-compose down

# Lists containers.
docker-compose ps
```

PROJECT DO IT YOURSELF

PROJECT DO IT YOURSELF

Let's build together our own docker dev environment

- Symfony
- reverse proxy
- and more ...

INSTRUCTIONS

I want a docker-compose file with those services:

- nginx
- php fpm
- PostgreSQL
- adminer

Directory structure is in the do-it-yourself folder (repo: https://github.com/yonzin/docker-workshop)

BONUS

- use a reverse-proxy (traefik) with a custom host
- mount a volume for nginx and fpm in foler infrastructure/logs
- add ssh-agent host config inside fpm container

WORKSHOP2

- try to migrate DockerDev yourself with the branch (epic/rework-CICD-project)
- remove all your containers
- check you don't have any modified file
- backup your .env and run make init_env and make init commands (README is coming)
- don't be afraid, workshop2 purpose is to present new DockerDev features and help you to migrate your env.

THE END

thank you for your attention workshop 2 is coming