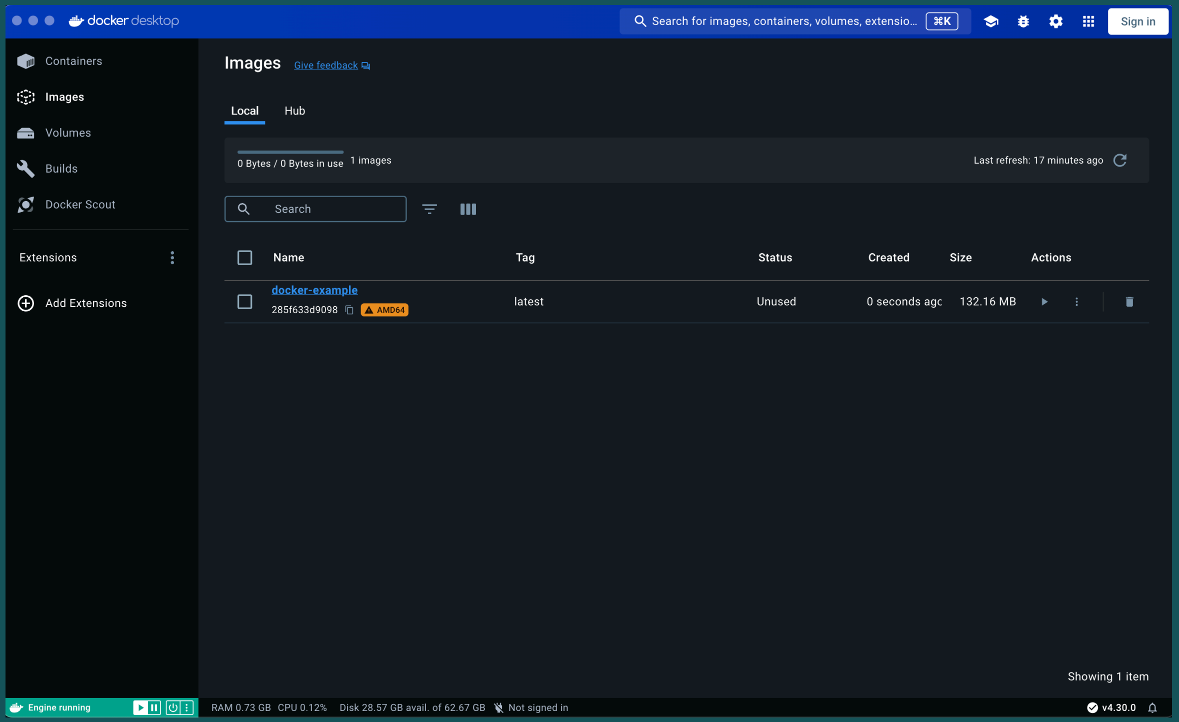
INTRODUCTION TO DOCKER

DOCKER PLATFORM

- Docker is an open source software to **develop**, **distribute** and **run** your code.
- It provides an engine that can be used as a command-line tool, or as a desktop interface.
- With docker you can **separate** an **application** from the **hardware**.
- This allows you to make your code easily **reproducible**, independently from the machine.



0 Bytes / 0 Bytes in use 1 images
Last refresh: 17 minutes ago

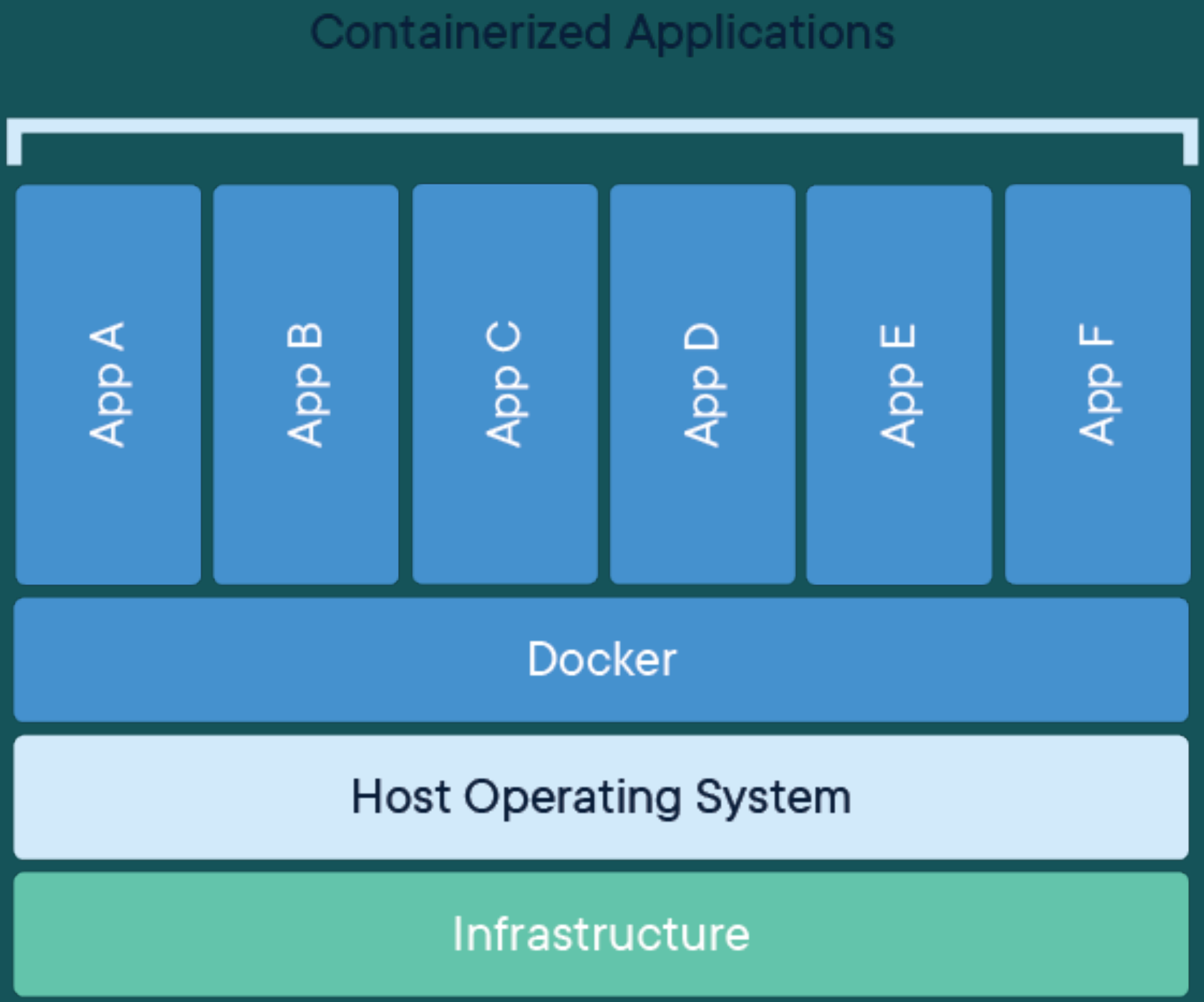
Name	Tag	Status	Created	Size	Actions
docker-example 285f633d9098 AMD64	latest	Unused	0 seconds agc	132.16 MB	

Showing 1 item

Engine running RAM 0.73 GB CPU 0.12% Disk 28.57 GB avail. of 62.67 GB Not signed in v4.30.0

DOCKER PLATFORM

- Docker allows to package the application into **containers**.
- **Containers** “contain” everything needed to run your code, so you don't need to rely on what's installed on the host machine.
- You can share containers, so your code will be executed in the same way, regardless of the host machine.

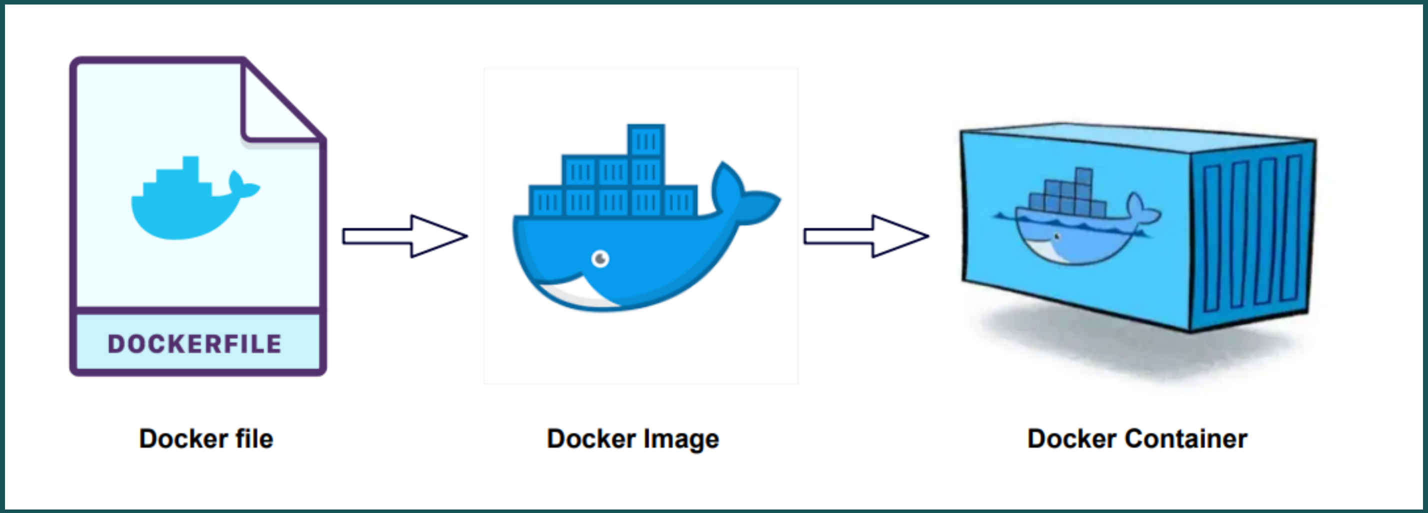


DOCKER CONTAINER'S USE CASES

- Docker containers allow for **effortless reproducibility** of an application.
- Example of use cases are, but not limited to:
    - **Sharing** of an **executable application**.
    - **Deployment** of an **application** on a server.
    - **Submission** for a project or a research paper.
    - Etc.

CREATION OF A DOCKER CONTAINER

- With a running instance of the **Docker engine**, a **container** can be created by:
    1. Writing a **Dockerfile**.
    2. Creating the **Image** from the **Dockerfile**.
    3. Running the **Image** to create the **Container**.

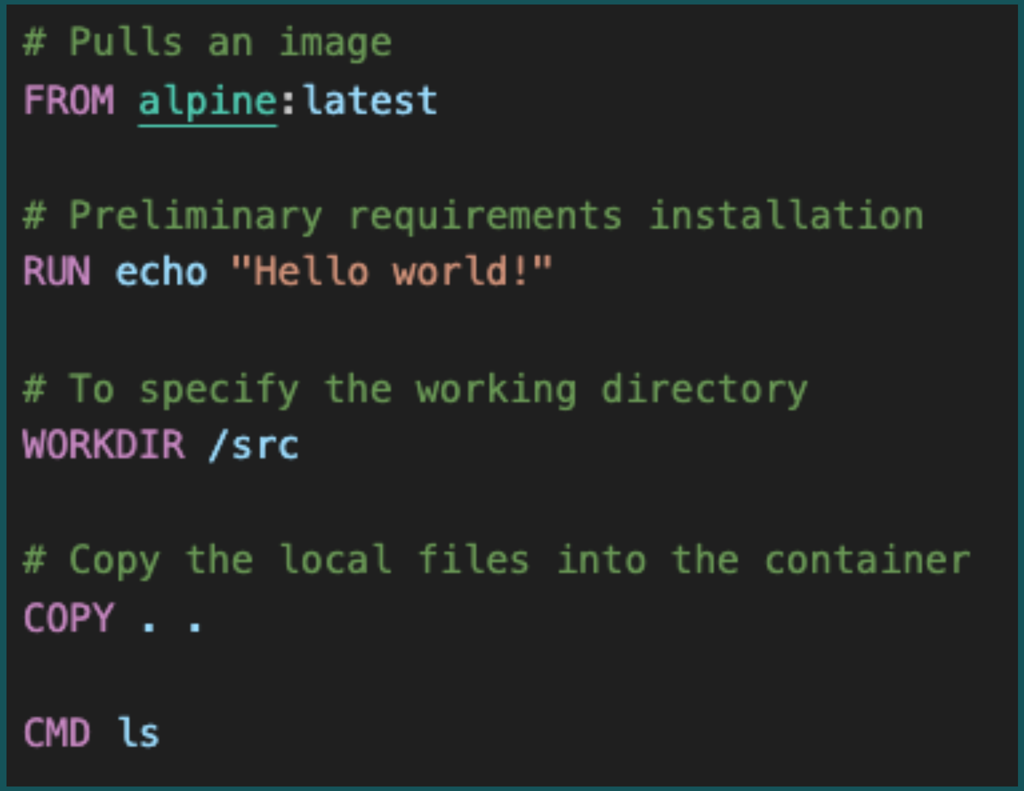


CREATION OF A DOCKER CONTAINER - IMAGE

- An **image** is a read-only template with instructions for creating a **Docker container**.
- Often, an **image** extends an existing one, with some additional customization .
- For example, you may build an image which is based on the **ubuntu** image, but install also a **Python** distribution.
- To create your own image, you need to create a **Dockerfile** defining the steps to create the **image** and run it.

CREATION OF A DOCKER CONTAINER - DOCKERFILE

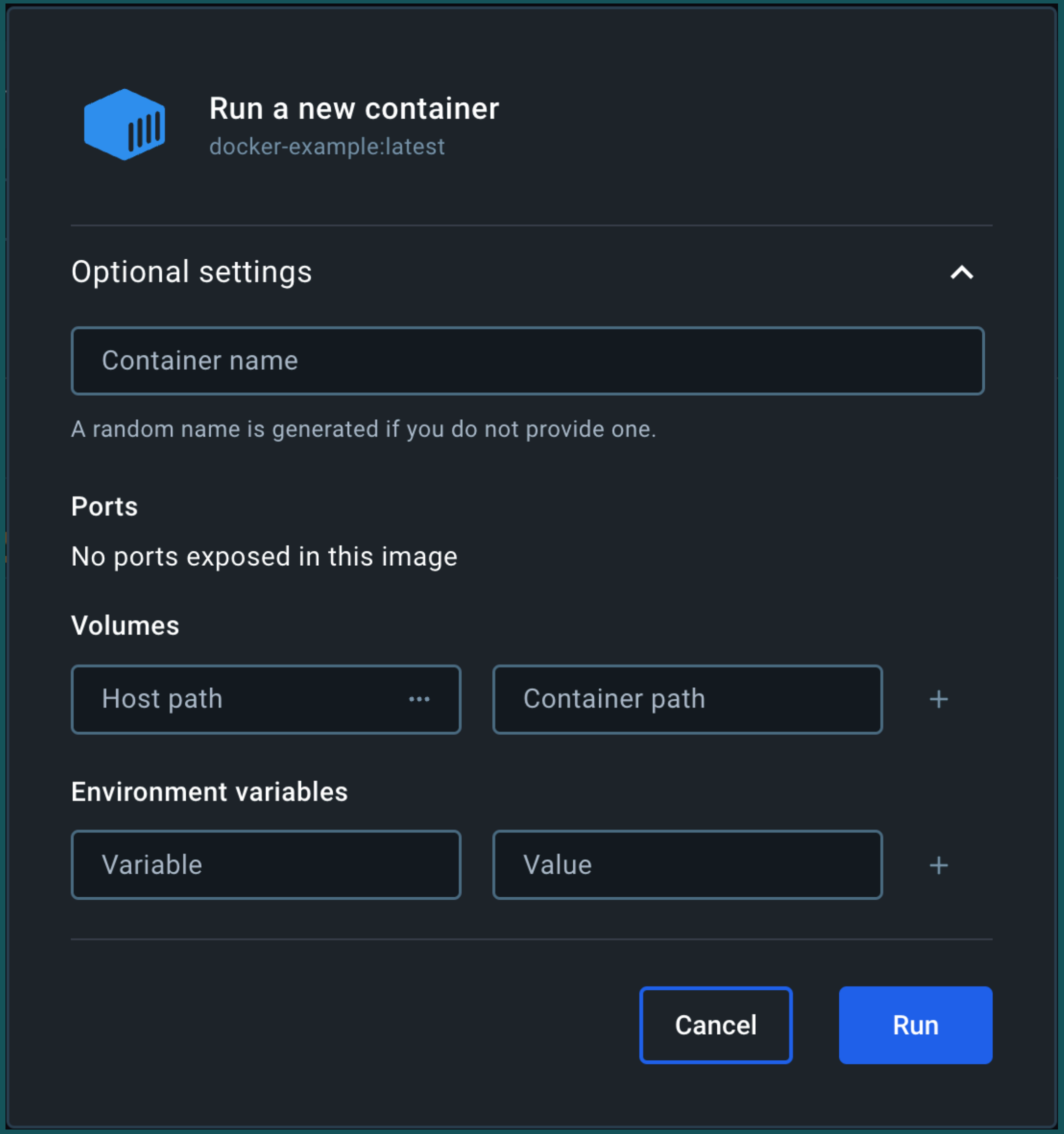
- The main arguments are :
  - **FROM**: the starting docker image, e.g. an OS.
  - **RUN**: preliminary operations on the base image, e.g. installation of other software.
  - **WORKDIR**: the home directory of the container.
  - **COPY**: the local files to copy inside the container.
  - **CMD**: the instruction to run when the container is started.



```
# Pulls an image
FROM alpine:latest

# Preliminary requirements installation
RUN echo "Hello world!"

# To specify the working directory
WORKDIR /src

# Copy the local files into the container
COPY . .

CMD ls
```

CREATION OF A DOCKER CONTAINER – CONTAINER

- A container is a runnable instance of an **image**, which can be started, stopped and deleted via the **Docker engine APIs**.
- A **container** is defined by its **image**, plus your **configuration options**.
- When a **container** is deleted, any changes to its internal state disappear, i.e. any file created or installations are removed.


Run a new container
docker-example:latest

Optional settings

Container name

A random name is generated if you do not provide one.

Ports

No ports exposed in this image

Volumes

Host path	Container path

Environment variables

Variable	Value

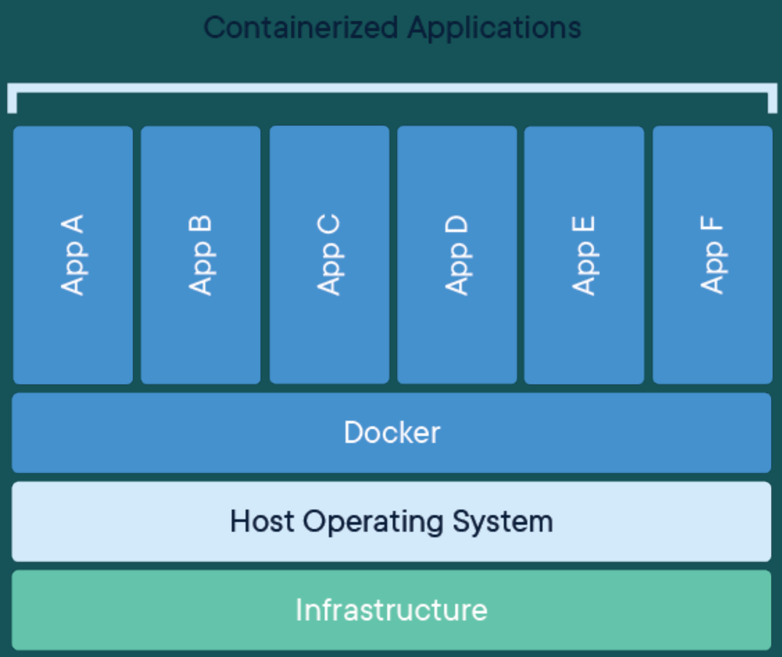
DOCKER CONTAINERS VS VIRTUAL MACHINES

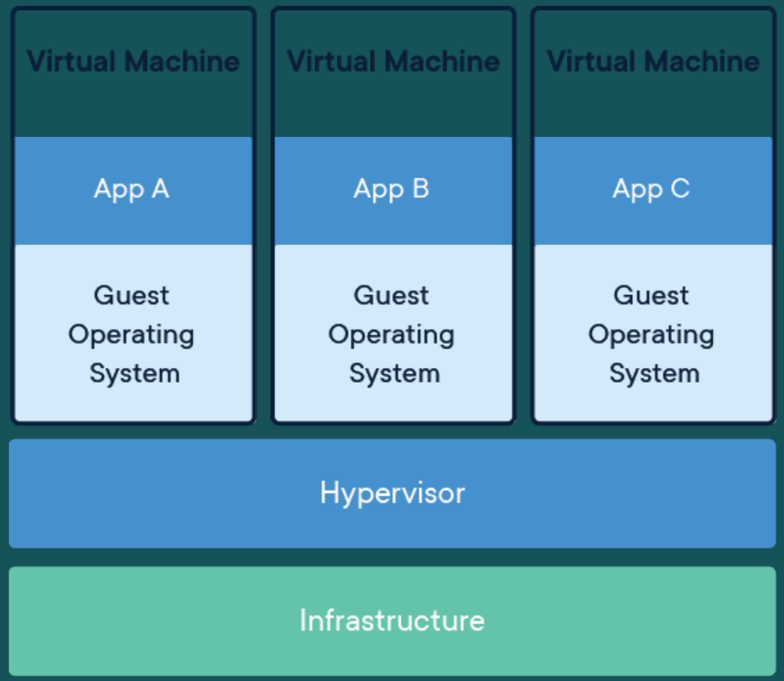
CONTAINER

- Abstraction at the **application layer**.
- Multiple containers can share the machine OS kernel.
- Easily distributable and reproducible.

VIRTUAL MACHINE (VM)

- Abstraction of **physical hardware**.
- Each VM includes a full copy of an OS, taking up several GBs.
- Slow to boot.


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure


Virtual Machine
Virtual Machine
Virtual Machine
App A
App B
App C
Guest Operating System
Guest Operating System
Guest Operating System
Hypervisor
Infrastructure