

Introduction to Docker

Angelo Quarta

20 May 2025

Table of Contents

1. Docker Platform

2. Docker Container

- Docker Image
- Dockerfile
- Docker Container

3. Container VS Virtual Machine



Docker Platform

Docker Platform

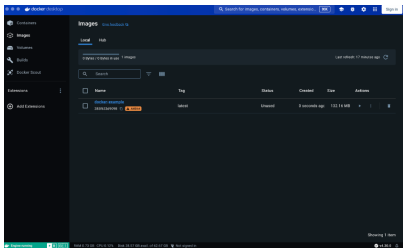


Docker is an open source software to **develop, distribute** and **run** your code.

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.

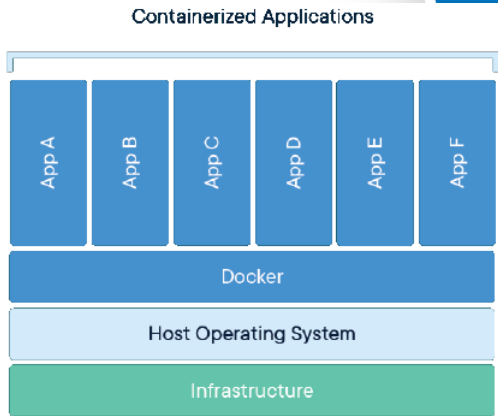


Docker Platform

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.



Use Case

Docker containers allow for **effortless reproducibility** of an application.

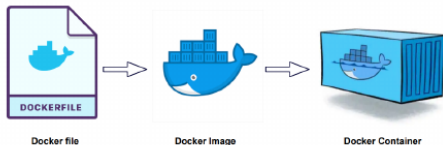
Example of use cases are, but not limited to:

- **Sharing** of an **executable application**.
- **Deployment** of an **application** on a server.
- **Submission** for a project or a research paper.

Creating a 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**.





Docker Image

Docker 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.



Dockerfile

Creating a 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
```



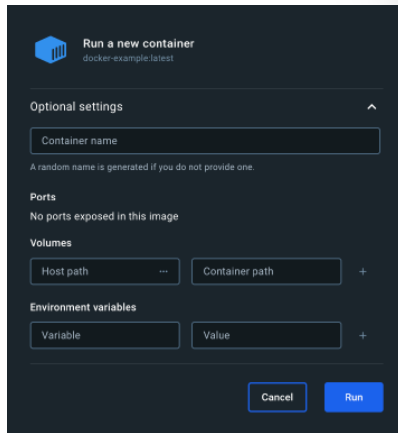
Docker Container

Creating a 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.



The screenshot shows the 'Run a new container' dialog box in Docker Desktop. At the top, there is a Docker logo and the text 'Run a new container' with 'docker-example:latest' below it. The dialog is divided into sections: 'Optional settings' with a 'Container name' input field and a note 'A random name is generated if you do not provide one.'; 'Ports' with the text 'No ports exposed in this image'; 'Volumes' with 'Host path' and 'Container path' input fields and a plus sign; and 'Environment variables' with 'Variable' and 'Value' input fields and a plus sign. At the bottom right, there are 'Cancel' and 'Run' buttons.

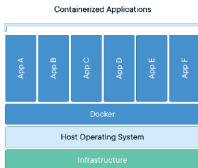


Docker Container VS Virtual Machine

Docker Container VS Virtual Machine

Container

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



Virtual Machine

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

