Part 1.
Introduction

What is Docker (1/3)

"Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud."

source: `https://www.docker.com/whatisdocker/`

What is Docker (2/3)

- a container manager
  - lightweight virtualisation
    (host and guest systems share the same kernel)
  - based on linux namespaces and cgroups

- massively copy-on-write
  - immutable images
  - instant deployment
  - suitable for micro-services (one process, one container)

$\rightarrow$ immutable architecture

What is Docker (3/3)

- a build system
  - images may be built from sources
  - using a simple DSL (`Dockerfile`)
- a set of REST APIs
  - Engine API (control the docker engine)
  - Plugin API (extend the engine $\rightarrow$ network, storage, authorisation)
  - Registry API (publish/download images)
  - Swarm API (manage a clusted of docker machines)

How Docker helps?

- **normalisation:** same environment (container image) for
  - development
  - jobs on the computing grid
  - continuous integration
  - peer review
  - demonstrations, tutorials
  - technology transfer

- **archival** *(ever tried to reuse old codes)*
  - source → Dockerfile = recipe to rebuild the env from scratch
  - binary → docker image = immutable snapshot of the software with its runtime environment
    → can be re-run at any time later

In practice

A docker image is an immutable snapshot of the filesystem

A docker container is

- a temporary file system
  - layered over an immutable fs (docker image)
  - fully writable (copy-on-write[1])
  - dropped at container's end of life (unless a `commit` is made)
- a network stack
  - with its own private address *(by defaut in 172.17.x.x)*
- a process group
  - one main process launched inside the container
  - all sub-processes SIGKILLed when the main process exits

[1] several possible methods: overlayfs (default), btrfs, lvm, zfs, aufs

Installation

https://docs.docker.com/engine/installation/

Native installation:

- requires linux kernel >3.8

Docker Machine:

- a command for provisionning and managing docker nodes deployed:
  - in a local VM (virtualbox)
  - remotely (many cloud APIs supported)