# Containers

Ján Magyar, MSc.

4th December 2018

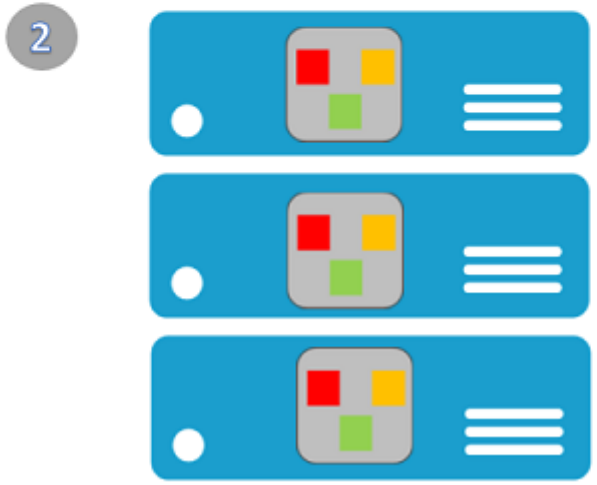
Application of Cloud Technologies

DCAI
Department of Cybernetics and Artificial Intelligence

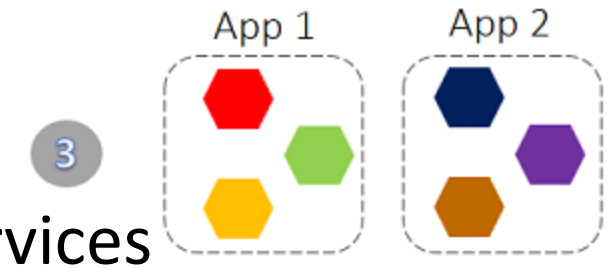
CIT
Center for Intelligent Technologies

# Monolithic applications

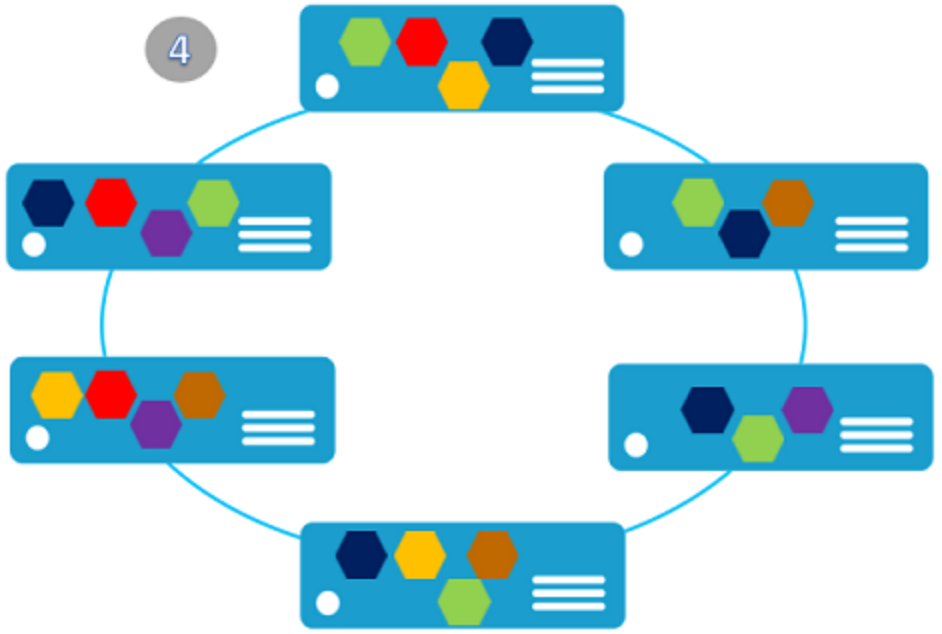
- single program on a single platform
- self-contained
- the application is responsible for every step needed for a function
- lack of modularity
- no code reuse
- hard to scale

# Microservices



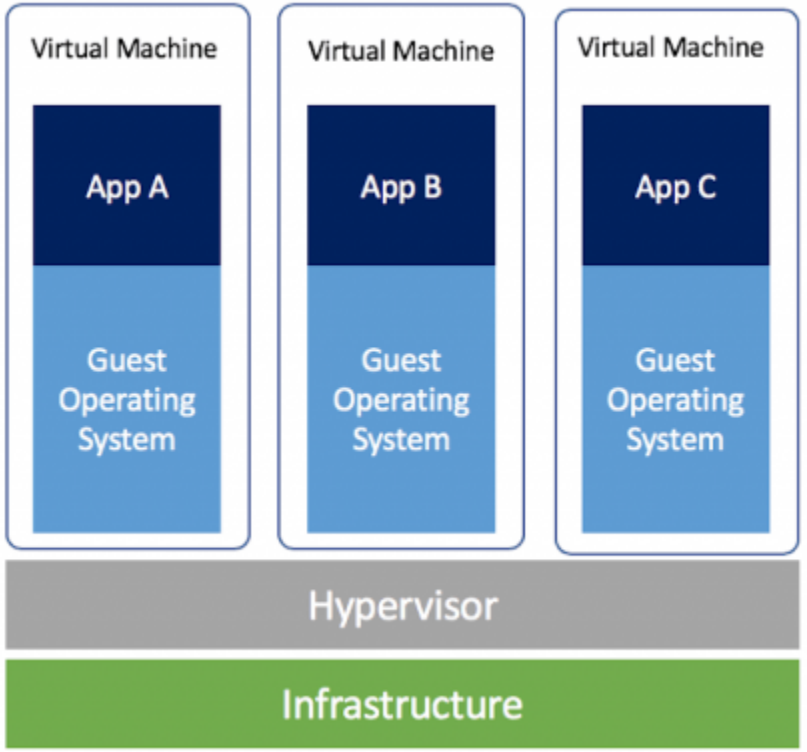

- application = collection of loosely coupled services
- each service is self-contained
  - might rely on other services
- a service is responsible for only one step
- modular solution
- reusable code
- easy scalability



# How to handle all the services?

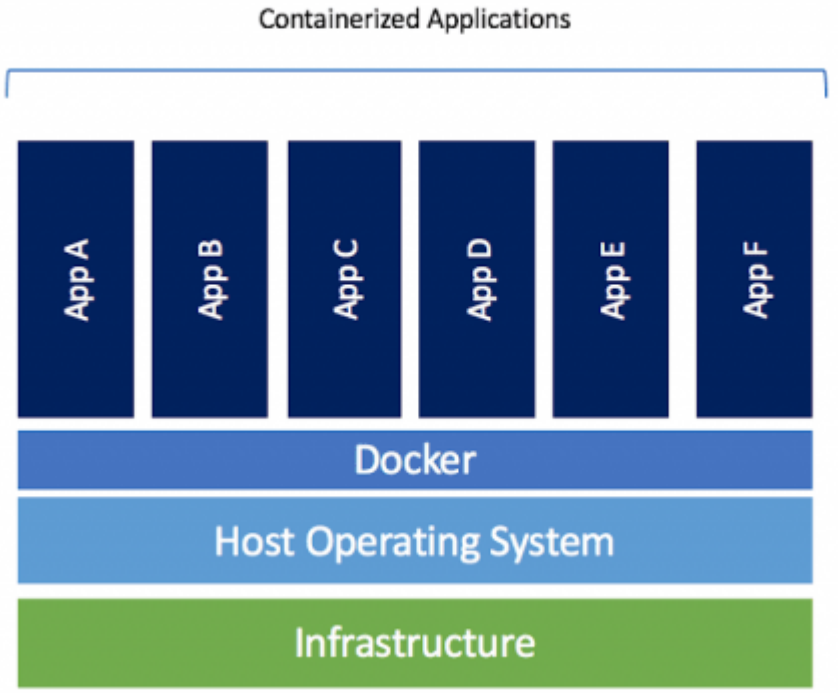
# Virtual machines

- emulation of an entire computer system
- we can set up a kernel with dependencies and applications
- hypervisor

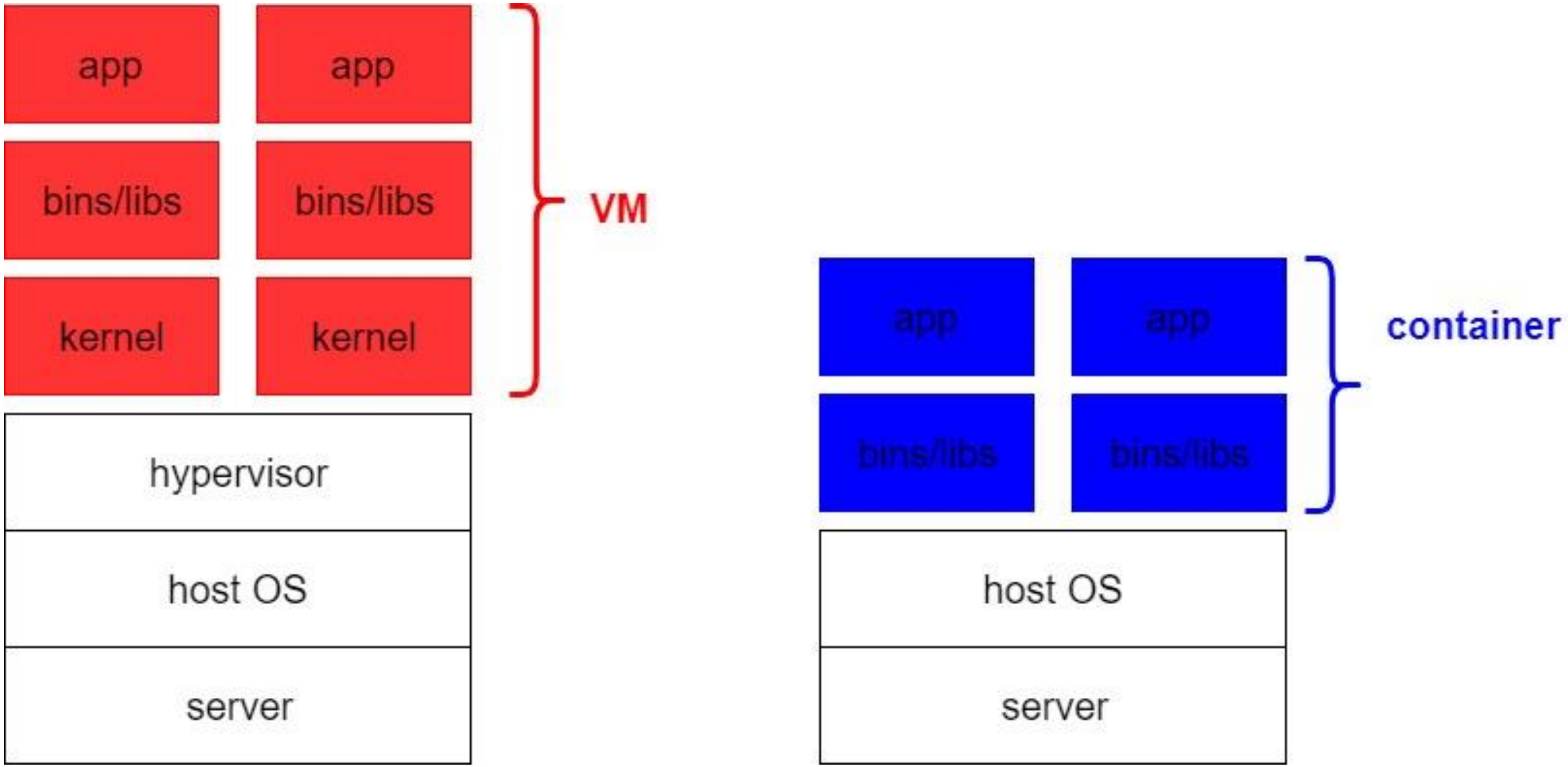


# Containers

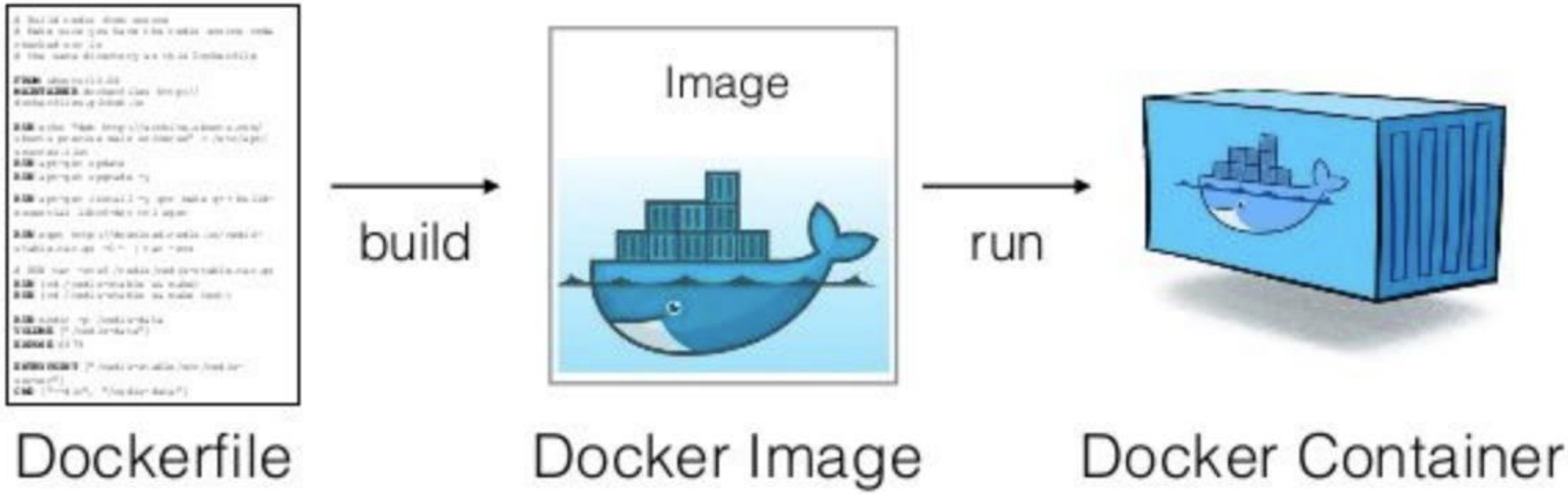
- operating-system-level virtualization
- the kernel allows the existence of multiple user-space instances
- similar to virtual environments



# VMs vs containers



# Containers 101

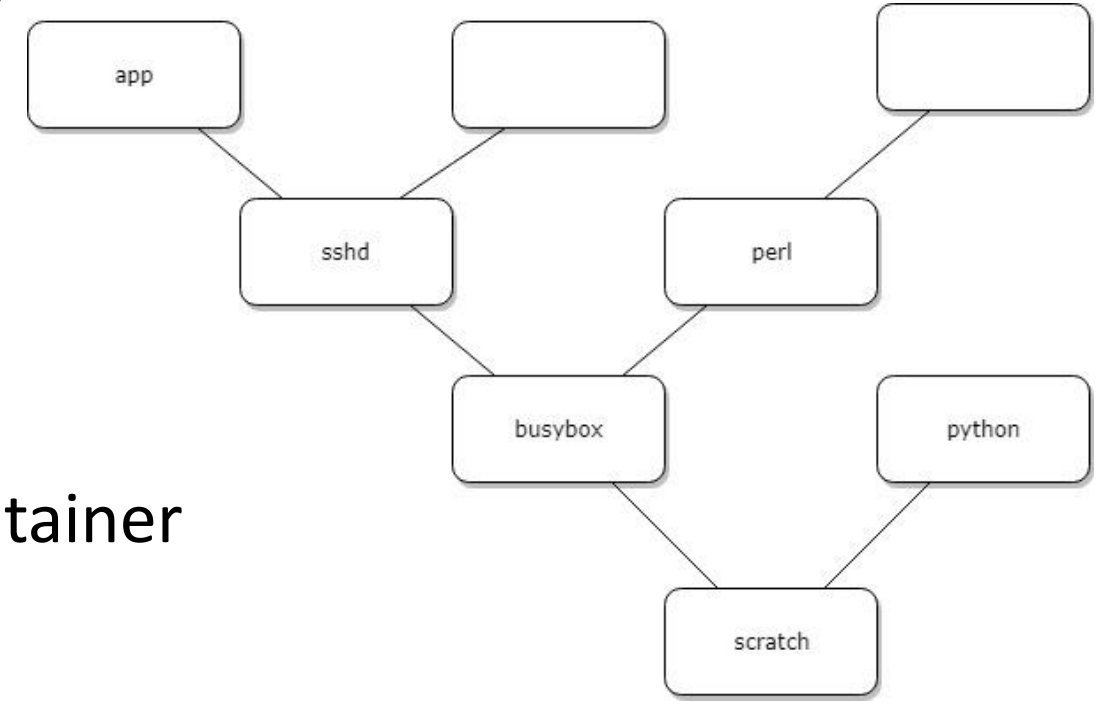


# What is a dockerfile

- textfile
- “recipe” of how to create an image
- list of instructions
  - usually shell commands
- define
  - what operating system we want to use
  - what dependencies the application has
- we BUILD a dockerfile to get an image

# What is an image?

- a snapshot of the system in which we want to run a process
- binary representation of an environment
- possible to build image stacks/hierarchies
  - easier maintenance
- contains
  - OS
  - software
  - application code
- we RUN an image to get a container



# What is a container?

- instance of an image
- fully isolated sandbox with inherent dependencies
- has
  - own process namespace
  - cgroups
    - limit what a process can do
    - resource limits
    - limit capabilities

# Container life cycle

- usually one process/container
- might run multiple processes in one computer but there is **one** main process
- container life cycle ≈ container process life cycle
- update
  - delete container
  - rebuild image
  - rerun container

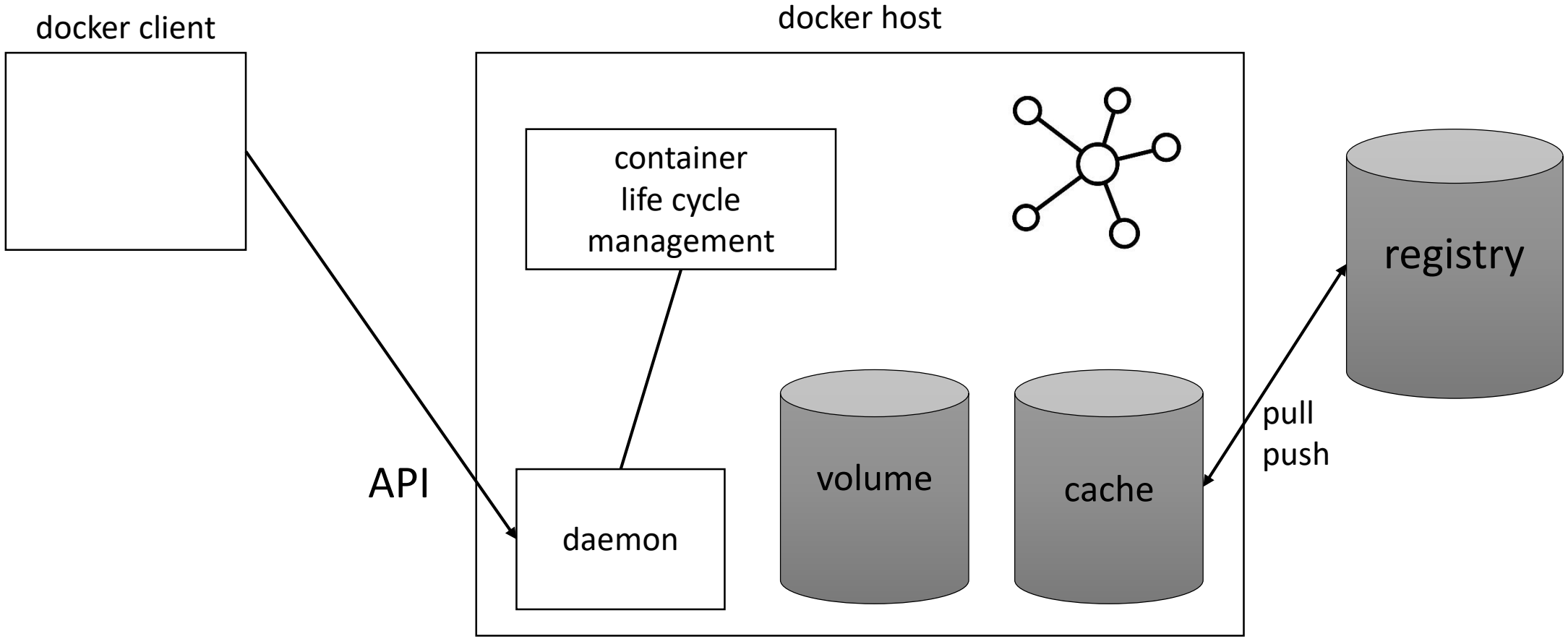
# But what about data?

- the container gets deleted once the main process ends
- to save data for further use, we can use **volumes**

# Important operations

- BUILD – create an image from dockerfile
- RUN – create a container from image
- PULL – load an image from registry
- PUSH – add an image to registry

# Tying it all together




docker

# any questions?

# Sources

- https://www.youtube.com/watch?v=YFl2mCHdv24
- https://www.youtube.com/watch?v=VqLcWftIaQI
- https://www.youtube.com/watch?v=EnJ7qX9fkcU
- https://www.youtube.com/watch?v=L1ie8negCjc