# Docker

## Kurs DevOps – Wykład 2

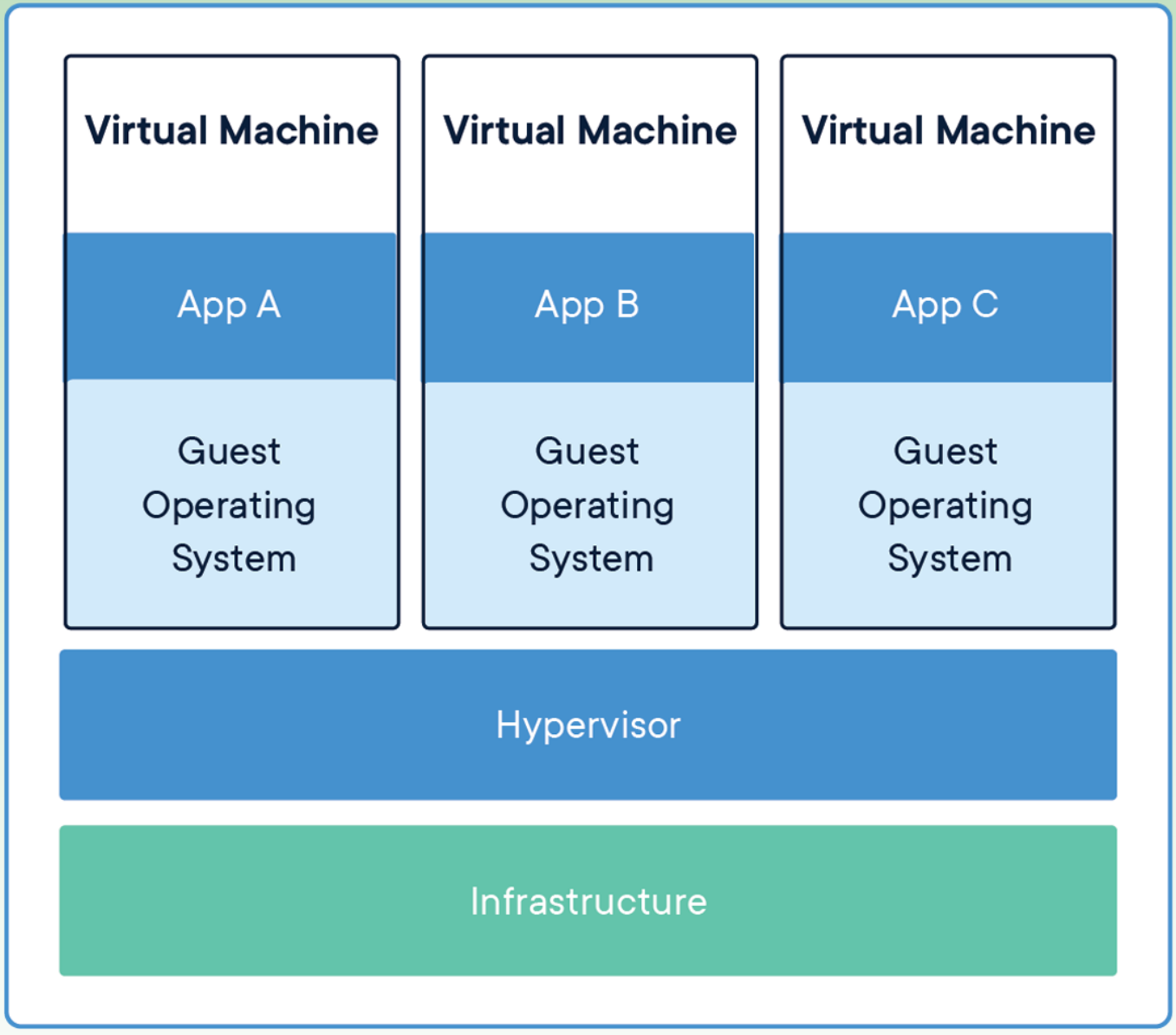
Kuba Nowak

Uniwersytet Wrocławski

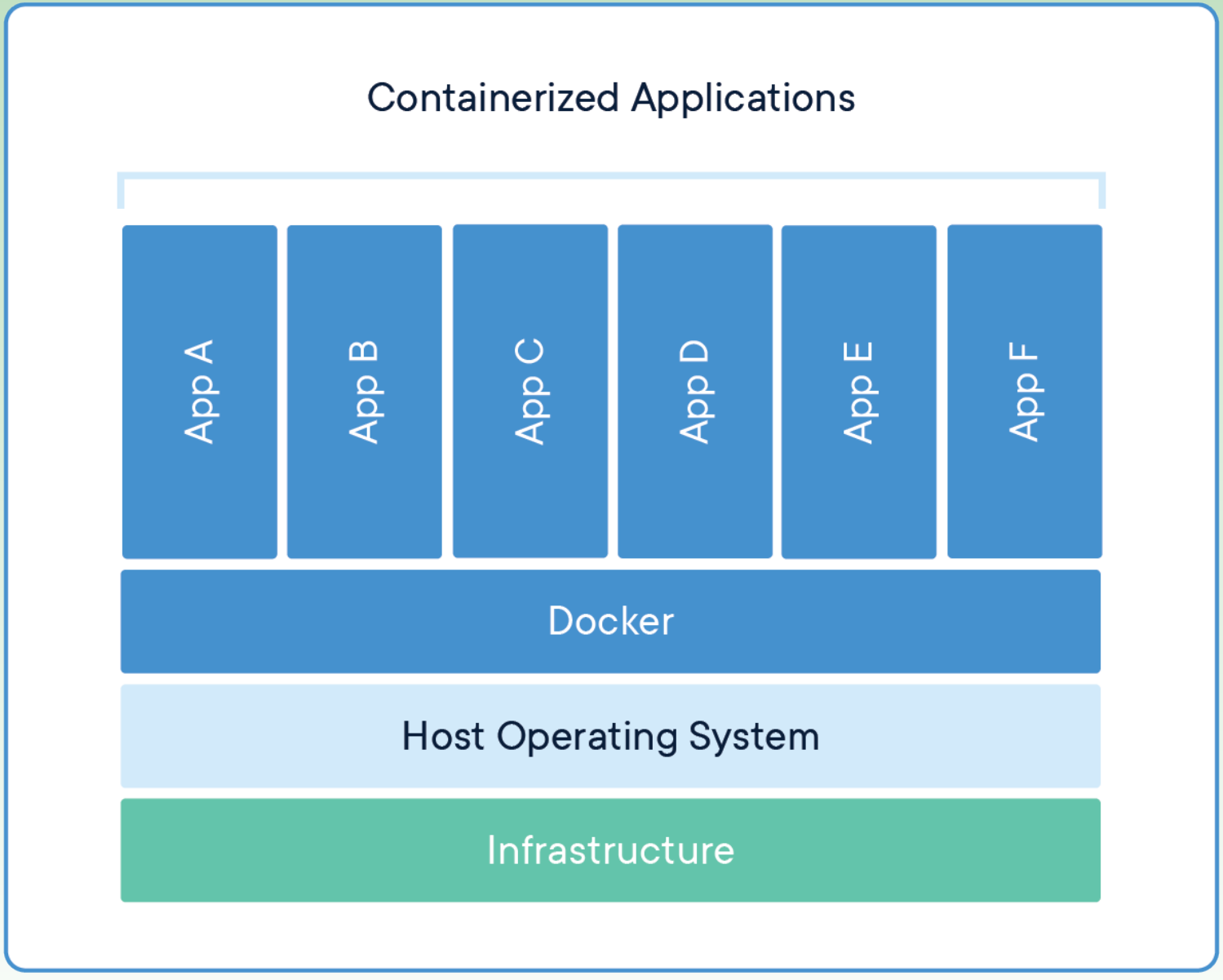
15 października 2025

# Czego oczekiwalibyśmy od środowiska?

- Reprodukowalność
- Stabilność
- Bezpieczeństwo (izolacja)
- Wydajność (szybkość)
- Łatwość konfiguracji



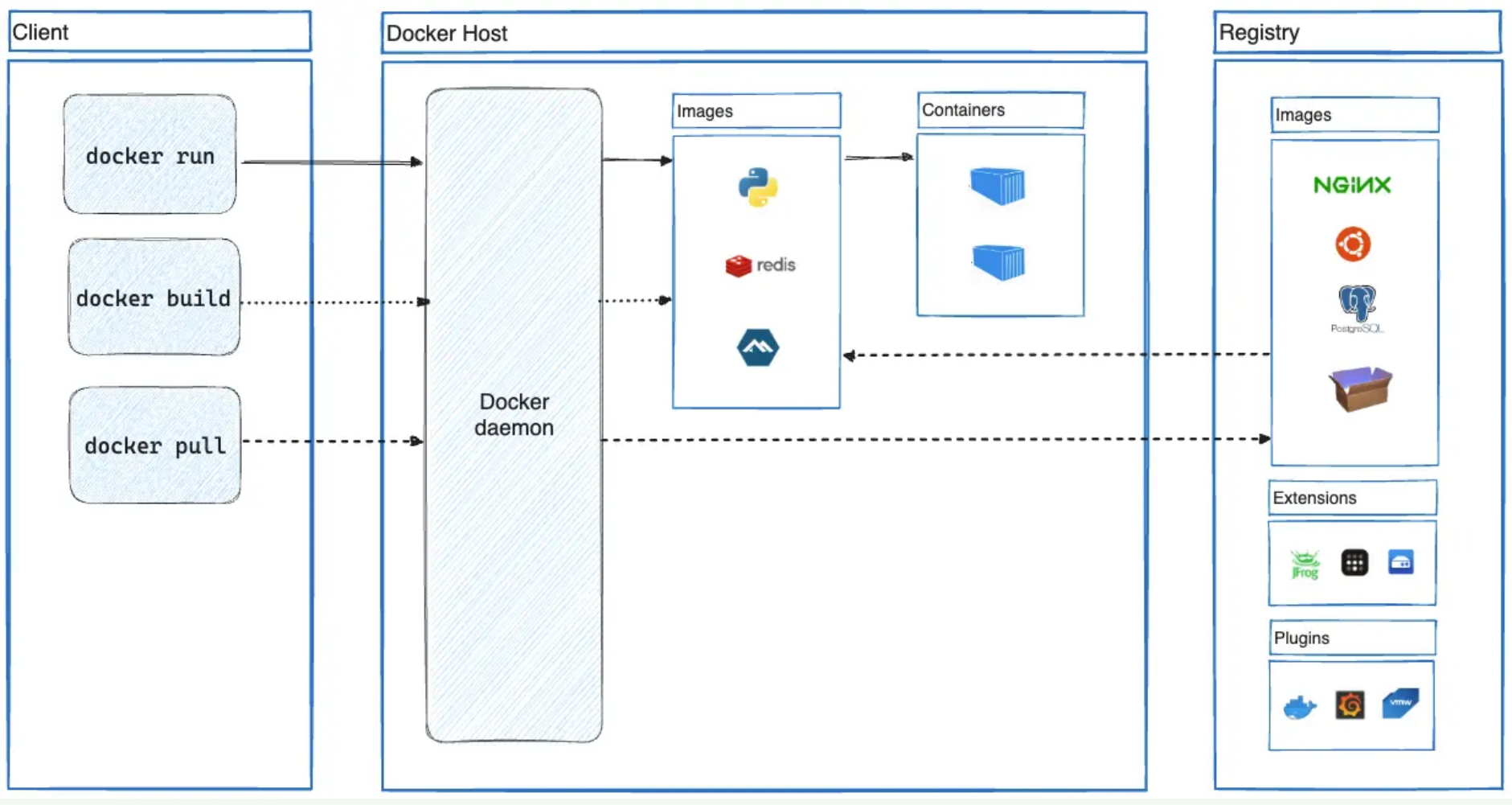

Źródło:
`https://www.docker.com/app/uploads/2021/11/container-vm-whatcontainer_2.png`




Źródło: https://www.docker.com/app/uploads/2021/11/docker-containerized-appliction-blue-border_2.png

# Docker

- Bazuje na funkcjonalnościach Linuksa, przy pomocy których można zapewnić izolację:
  - namespaces
  - cgroups
  - capabilities
- Kontenery nie mają swojego jądra.
- Procesy w kontenerze są procesami hosta.




Źródło: `https://docs.docker.com/get-started/images/docker-architecture.webp`

# Demonstracja 1

# Docker a Windows

Docker Desktop:

- GUI do zarządzania dockerami.
- Główny sposób pracy na Windowsach.
- Darmowa licencja do użytku prywatnego (korporacje muszą płacić).

# Docker a Windows

Docker Desktop:

- GUI do zarządzania dockerami.
- Główny sposób pracy na Windowsach.
- Darmowa licencja do użytku prywatnego (korporacje muszą płacić).

**Uwaga**

Nie będziemy się zajmować Docker Desktop. Windowsowcy są zdani wyłącznie na siebie.

# Docker a Windows

Docker Desktop:

- GUI do zarządzania dockerami.
- Główny sposób pracy na Windowsach.
- Darmowa licencja do użytku prywatnego (korporacje muszą płacić).

## Uwaga

Nie będziemy się zajmować Docker Desktop. Windowsowcy są zdani wyłącznie na siebie.

## Ćwiczenia

Jeśli ktoś chce pracować na Windowsie z Dockerem, to na ćwiczeniach oczekiwane jest, że będzie korzystał z wiersza poleceń, by wykonać zadania. Nie z GUI Docker Desktop.

# System budowania obrazów

Architektura klient-serwer:

**Buildx** klient budowania wbudowany w polecenie `docker build`.

**BuildKit** serwer budowania będący częścią `dockerd`

## Inne backendy

Zamiast BuildKita wbudowanego w `dockerd` można użyć:
`docker-container`, `kubernetes`, `remote`.

# Configuration as a Code

## Pytanie

Jak konfigurować obrazy?

# Configuration as a Code

## Pytanie

Jak konfigurować obrazy?

- Reprodukowalność
- Kontrola wersji
- Odpowiedzialność za dokonane zmiany
- Skalowalność

# Configuration as a Code

## Pytanie

Jak konfigurować obrazy?

- Reprodukowalność
- Kontrola wersji
- Odpowiedzialność za dokonane zmiany
- Skalowalność

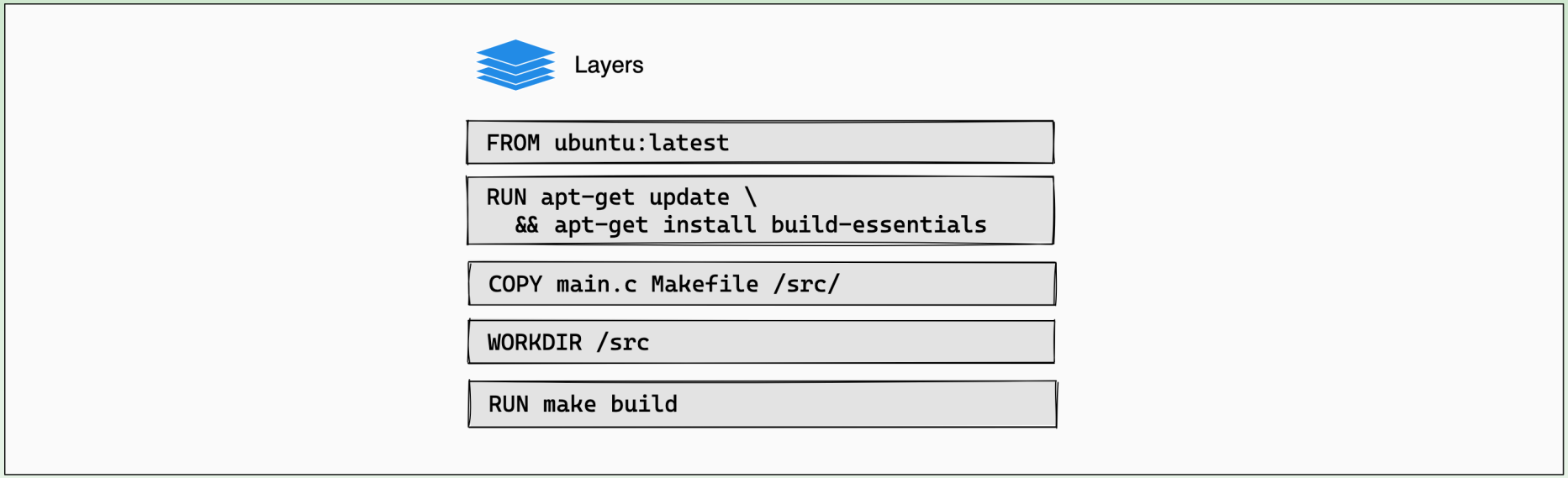
**Configuration as a Code** to sposób przechowywania konfiguracji jako plik tekstowy "kod".

# Dockerfile

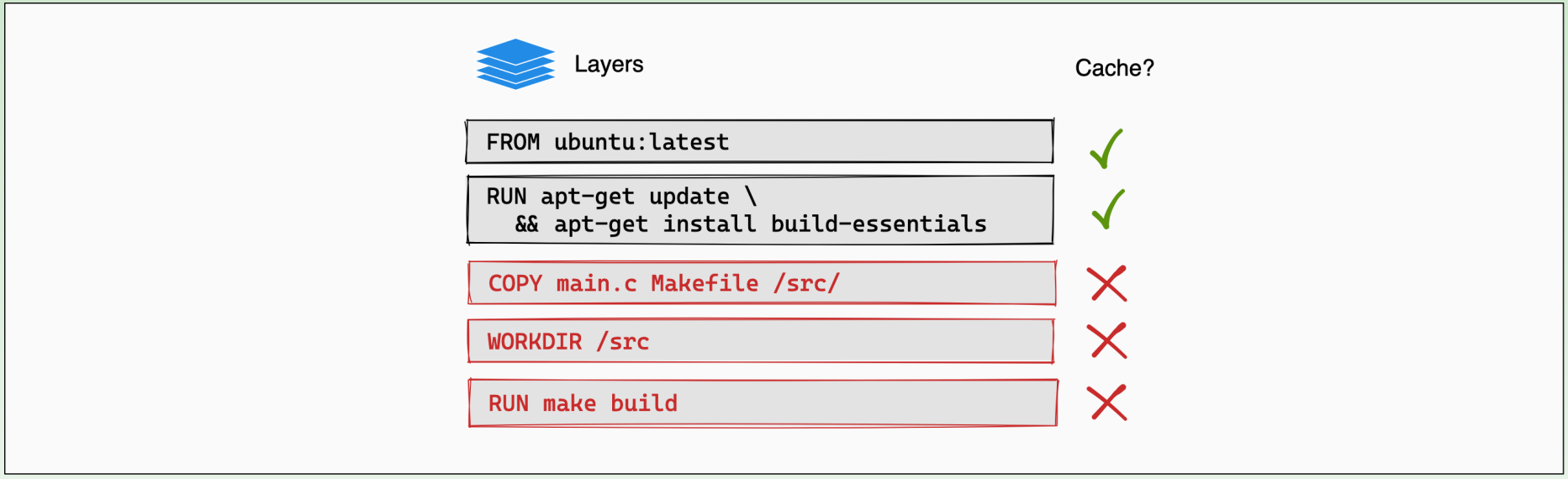
Opisuje jak zbudować obraz. Najczęstsze polecenia:

- FROM określa *obraz podstawowy*, który będziemy rozszerzać.
- WORKDIR ustawia katalog roboczy *wewnątrz* kontenera
- COPY kopiuje plik między hostem, a obrazem
- RUN uruchamia polecenie
- ENV ustawia zmienną środowiskową
- EXPOSE informuje o używanych portach sieciowych
- USER ustawia UID użytkownika
- CMD konfiguruje domyślnie uruchamiane polecenie

`https://docs.docker.com/get-started/docker-concepts/building-images/writing-a-dockerfile/`
`https://docs.docker.com/reference/dockerfile/`




Źródło: https://docs.docker.com/build/cache/




Źródło: `https://docs.docker.com/build/cache/`

# Demonstracja 2

# Główne elementy dockera

**Obraz** jaki jest, każdy widzi[1].

**Kontener** uruchomiona instancja obrazu.

**Wolumen** dysk, będący niezależny od obrazu/kontenera.

**Sieć** do której można podłączyć kontenery, tak by się ze sobą komunikowały.

**Serwis** to zbiór instancji współpracujących ze sobą, by dostarczyć funkcjonalność.

[1] A jak nie widzi, to się zgłosi.

# Przechowywanie danych

**Overlay** Na niemodyfikowalny system plików z obrazu, nakładany się overlay, który po zamknięciu instancji jest usuwany.

**tmpfs** Podobnie jak overlay, ale efektywniejszy, bowiem nie sprawdza systemu plików z obrazu.

**Wolumen** Wirtualny dysk, zarządzany w pełni przez dockera. Można go podłączać do różnych kontenerów.

**Katalog współdzielony** wykorzystuje `mount --bind` by zobaczyć dysk hosta (nieprzenośne np. na Windowsa).

**Inne** jest wiele mniej popularnych backendów (np. rclone)

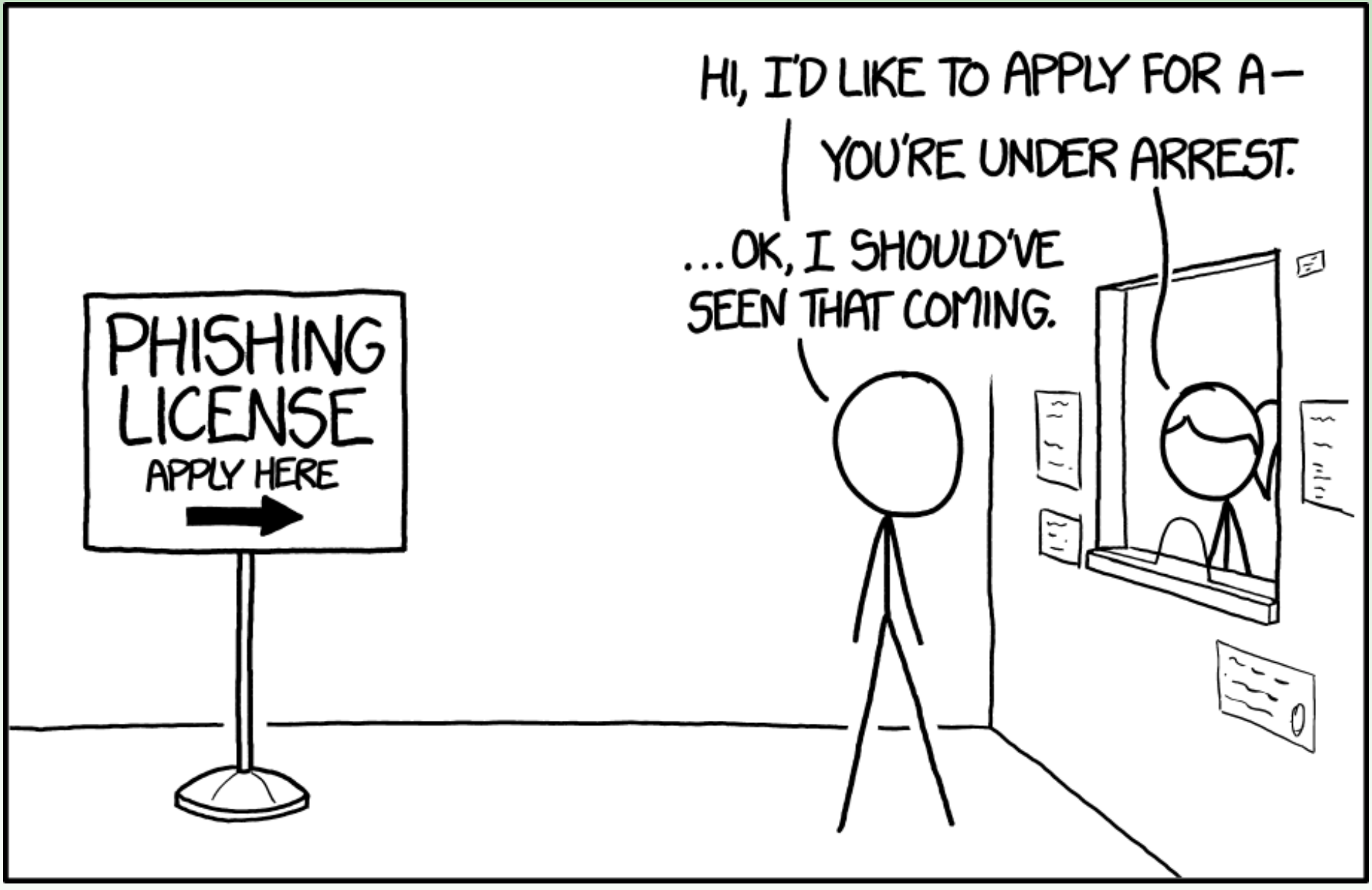
# Sieć

`docker network` wrapuje interfejs SO w jeden generyczny, ułatwiający zarządzanie sieciami wirtualnymi.
Typy sieci:

- **bridge** wirtualna sieć, w której każdy może się komunikować z każdym. Może być podłączona do internetu bądź nie.
- **host** kontenery widzą urządzenia sieciowe hosta.
- **overlay** łączy dockery z wielu hostów w jedną sieć.
- **ipvlan** tworzy na hoście interfejsy VLAN z których korzysta kontener.
- **macvlan** post hosta ma wiele MAC adresów i sortuje pakiety między interfejsami wirtualnymi (a część z nich należy do kontenerów).
- **none** brak sieci.

`https://docs.docker.com/engine/network/`

# Demonstracja 3




Źródło: https://xkcd.com/1694/

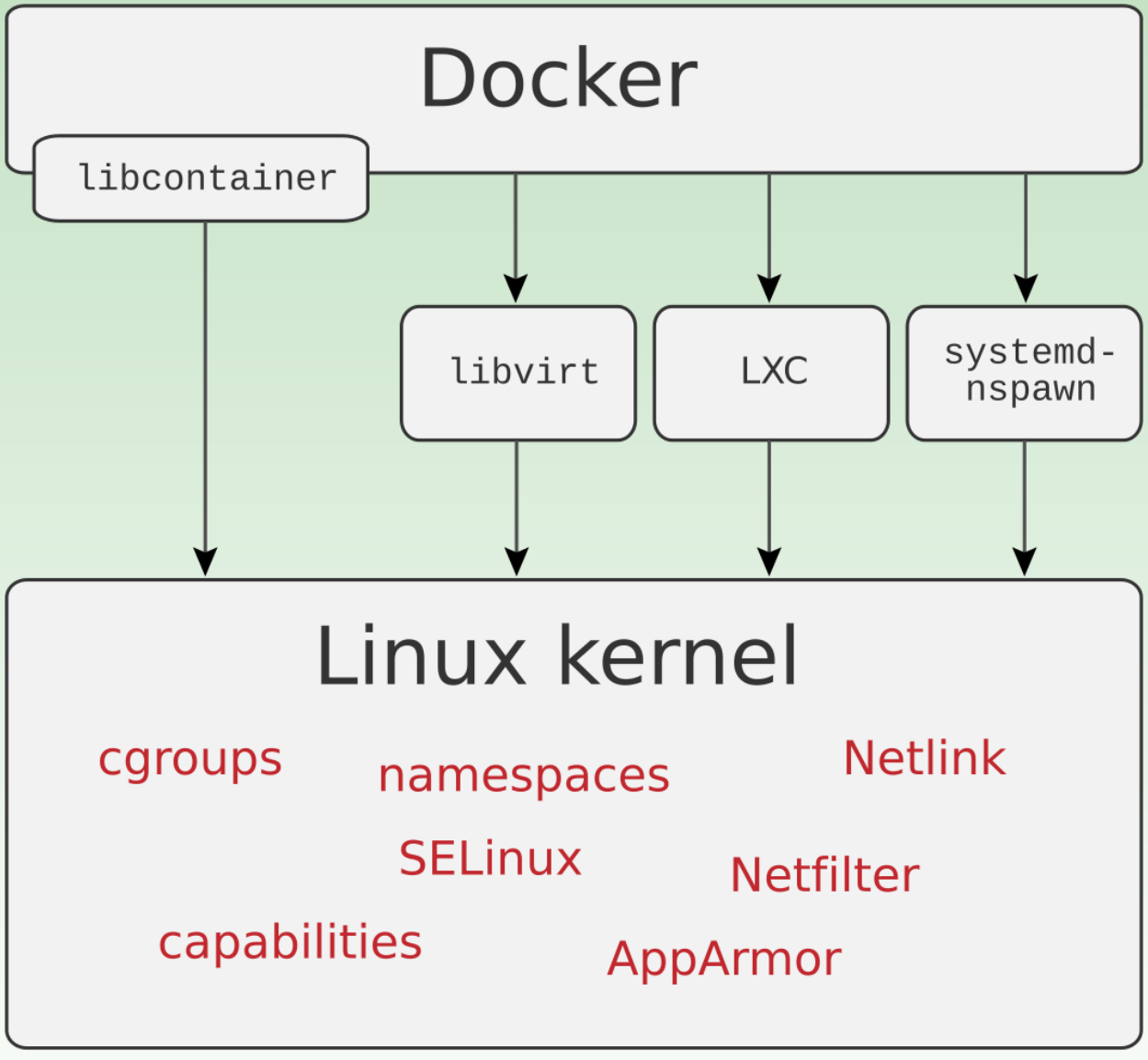
# Licencje

Żeby nie było za prosto:

- Docker Engine na Linuka – Apache 2.0
- Części Docker Destop – GPL
- Części Docker Desktop – licencja płatna (dla dużych przedsiębiorstw)

# Backendy dockera

- Linux namespaces + cgroups + capabilities (Linux)
- Virtual Machine Managers (MacOS)
- Apple Virtualization framework (MacOS)
- QEMU (na MacOS)
- LXC (Linux, Windows)
- Hyper-V (Windows)
- WSL 2 (Windows)




Źródło: https://en.wikipedia.org/wiki/File:Docker-linux-interfaces.svg

# Izolacja

## Linux namespace

Pozwala podzielić system operacyjny na części, które nie widzą się wzajemnie.

Aktualne przestrzenie wspierane przez Linuksa:

- cgroup
- Inter Proces Communication
- Network
- Mount
- PID
- Time
- User
- Unix Time System (czyli nazwa hosta...)

`https://man.archlinux.org/man/namespaces.7.en`

# Linux namespace

Automatycznie montowane w przestrzeni montażowej kontenera:

- /sys
- /proc/sys
- /proc/sysrq-trigger
- /proc/irq
- /proc/bus

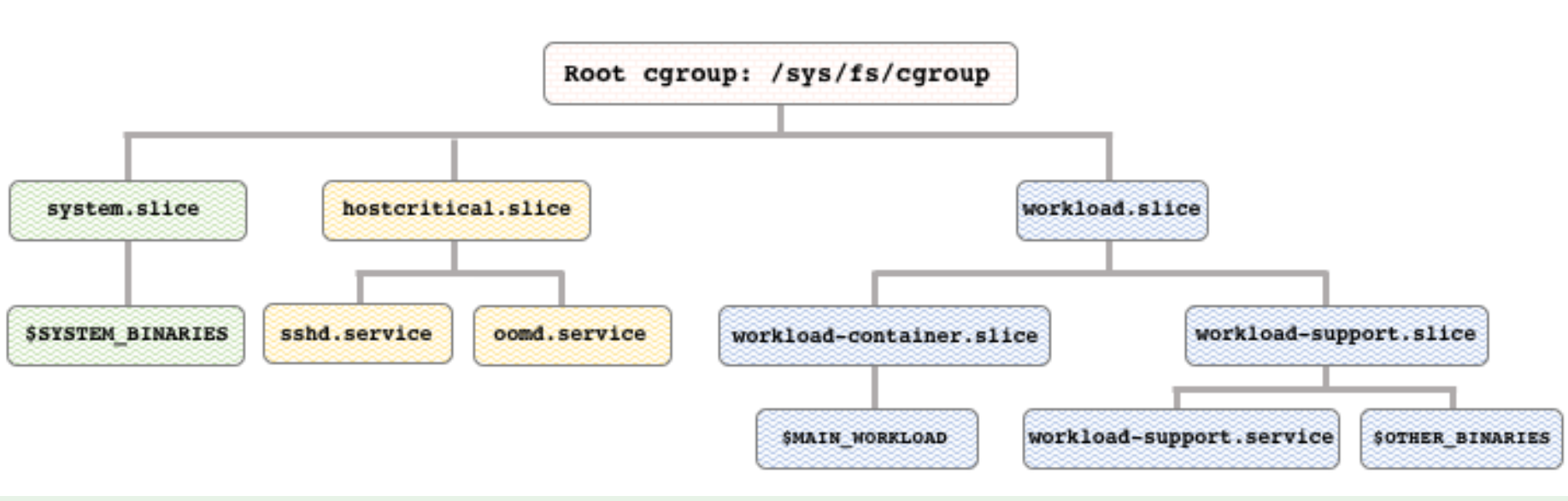
# cgrupy

## Control Groups

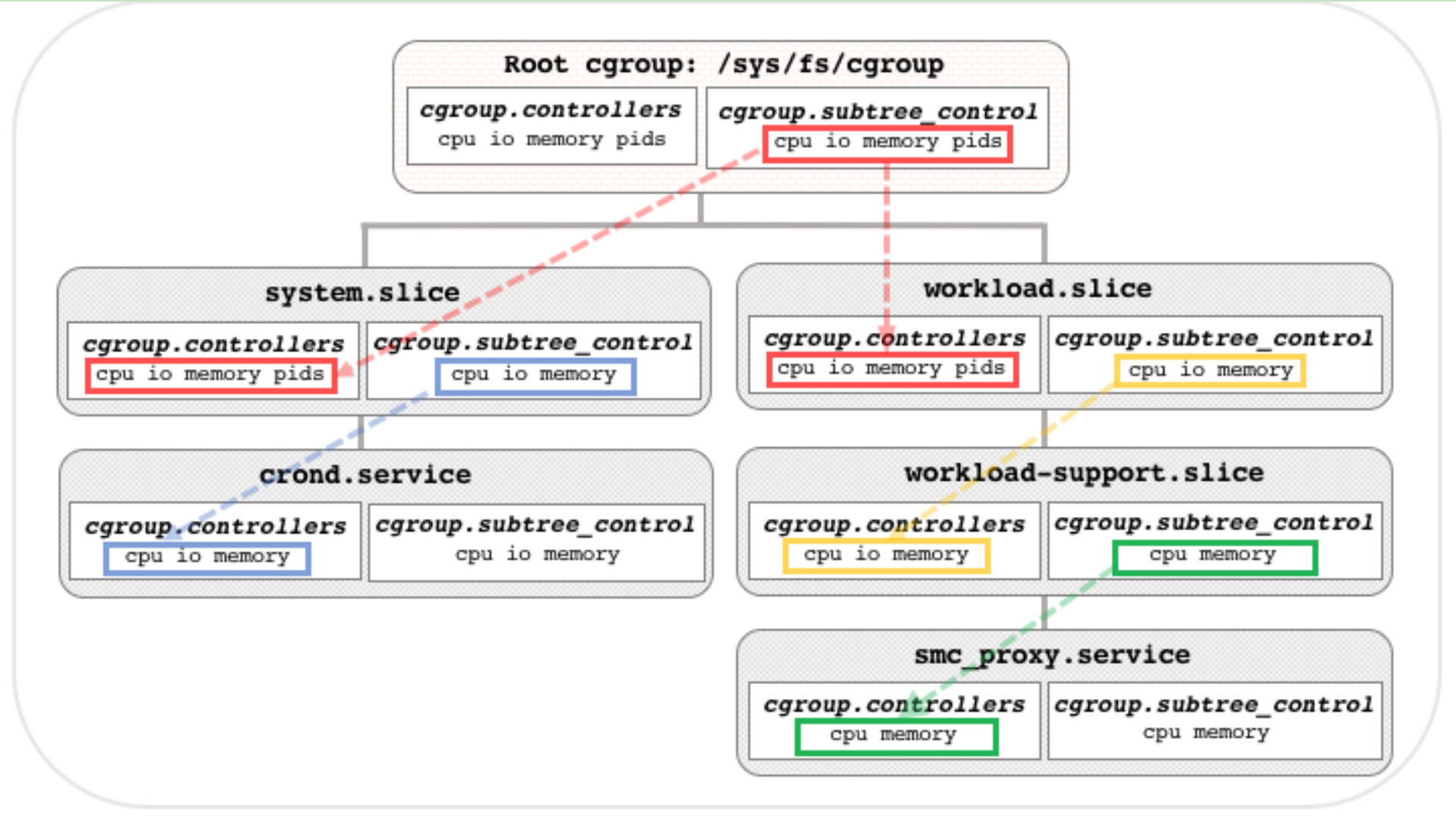
Funkcjonalność jądra Linuksa, pozwalająca na monitorowanie, priorytetyzację i ograniczanie zużycia zasobów.

- Bazuje na hierarchiach procesów (wielu w `v1` pojedynczej w `v2`).
- Do hierarchii podłączone są kontrolery monitorujace zużycie zasobów.
- Dla każdej grupy w hierarchii można określić limit zasobów.
- Kontrolowane przy pomocy *sysfs* bądź `systemd`.

`https://docs.kernel.org/admin-guide/cgroup-v2.html`




Źródło: https://facebookmicrosites.github.io/cgroup2/docs/assets/FbtaxHierarchyExternal.png




Źródło:
https://facebookmicrosites.github.io/cgroup2/docs/assets/hierarchyEXTERNAL.png

# Kontrolery cgroup

- cpu
- cpuset
- memory
- io
- hugetlb
- freezer
- perf_event
- pids
- rdma

`https://man7.org/linux/man-pages/man7/cgroups.7.html`

# Capabilities

System Linuksa, pozwalający przyznać procesowi część uprawnień root-a. Aktualnie z powodu ograniczeń implementacyjnych może być ich nie więcej niż 64.

Kilka ciekawych kategorii:

- CAP_BPF
- CAP_KILL
- CAP_NET_ADMIN
- CAP_NET_BIND_SERVICE
- CAP_NET_RAW
- CAP_PERFMON
- CAP_SYS_ADMIN

`https://man7.org/linux/man-pages/man7/capabilities.7.html`

# Zmiana uprawnień kontenera

Pozbycie się uprawnień pozwalających na zmianę uid i gid:

```
docker run --cap-drop setuid --cap-drop setgid -ti rhel7 /bin/sh
```

Zdobycie wszystkich uprawnień poza sys-admin:

```
docker run --cap-add all --cap-drop sys-admin -ti rhel7 /bin/sh
```

# Bezpieczeństwo

**Uwaga**

Pomimo przedstawionych mechanizmów, docker nie jest bezpieczny i można go wyeksploitować. Nie należy w nim uruchamiać niezaufanych programów.

# Bezpieczeństwo

## Uwaga

Pomimo przedstawionych mechanizmów, docker nie jest bezpieczny i można go wyeksploitować. Nie należy w nim uruchamiać niezaufanych programów.

## Reklama

Docker nie zabezpiecza przed atakami bazującymi na HW: RowHammer, Spectre, Meltdow, RowPress...
Osoby chcące się dowiedzieć czemu, zapraszam na Architektury Komputerów :)

# Bezpieczeństwo

## Uwaga

Pomimo przedstawionych mechanizmów, docker nie jest bezpieczny i można go wyeksploitować. Nie należy w nim uruchamiać niezaufanych programów.

## Reklama

Docker nie zabezpiecza przed atakami bazującymi na HW: RowHammer, Spectre, Meltdow, RowPress...
Osoby chcące się dowiedzieć czemu, zapraszam na Architektury Komputerów :)

Temat bezpieczeństwa dockerów jest bardzo szeroki i nie będziemy w niego dogłębnie wnikali. Raczej zajmiemy się tym jak docker działa i jak go używać.

# Alternatywy dla Dockera

- Podman
- Buildah
- Maszyny wirtualne
- LXC
- Linuks namespaces / capabilities