Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT, HSNLab
2019

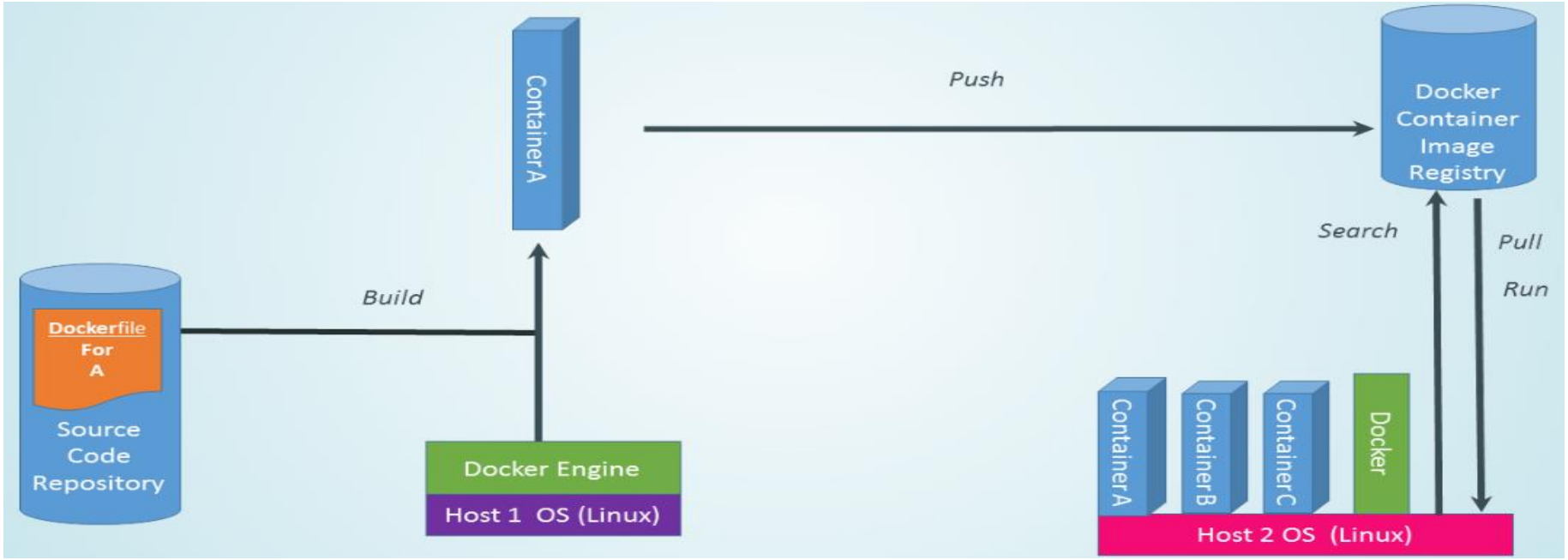
Mi a Docker?

- Docker = Linux container engine
- Open Source project
  - első verzió: 3/2013 by dotCloud
  - átnevezték Docker Inc-re
- Eredetileg Python kód, később Go
- https://www.docker.io/
- git repository:
  https://github.com/dotcloud/docker.git

Docker terminológia

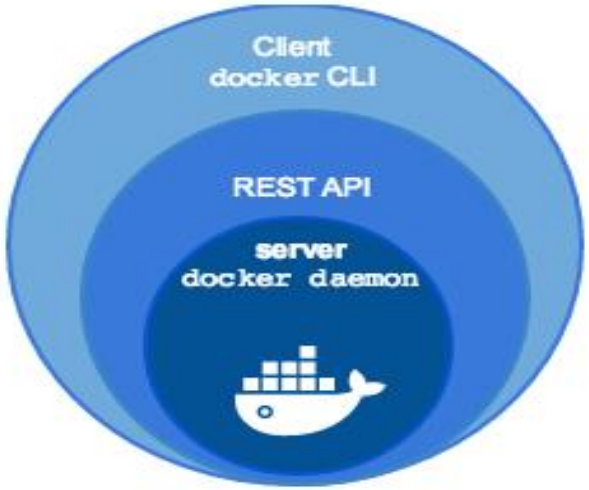
- Képfájl (image) = egy VM-nek megfelelő fájl együttes, amely tartalmaz minden olyan kiegészítést (lib, db, config, stb), ami szükséges az igényelt alkalmazás futtatásához
- Konténter (container) = egy Docker image futtatott példánya
- Tárház (registry) = képfájlok tára
  - Alapesetben helyi (on-host)
  - A Docker cég fenntart egy nyilvános, globális, on-line adatbázist (github-hoz hasonló)

Mi a Docker?

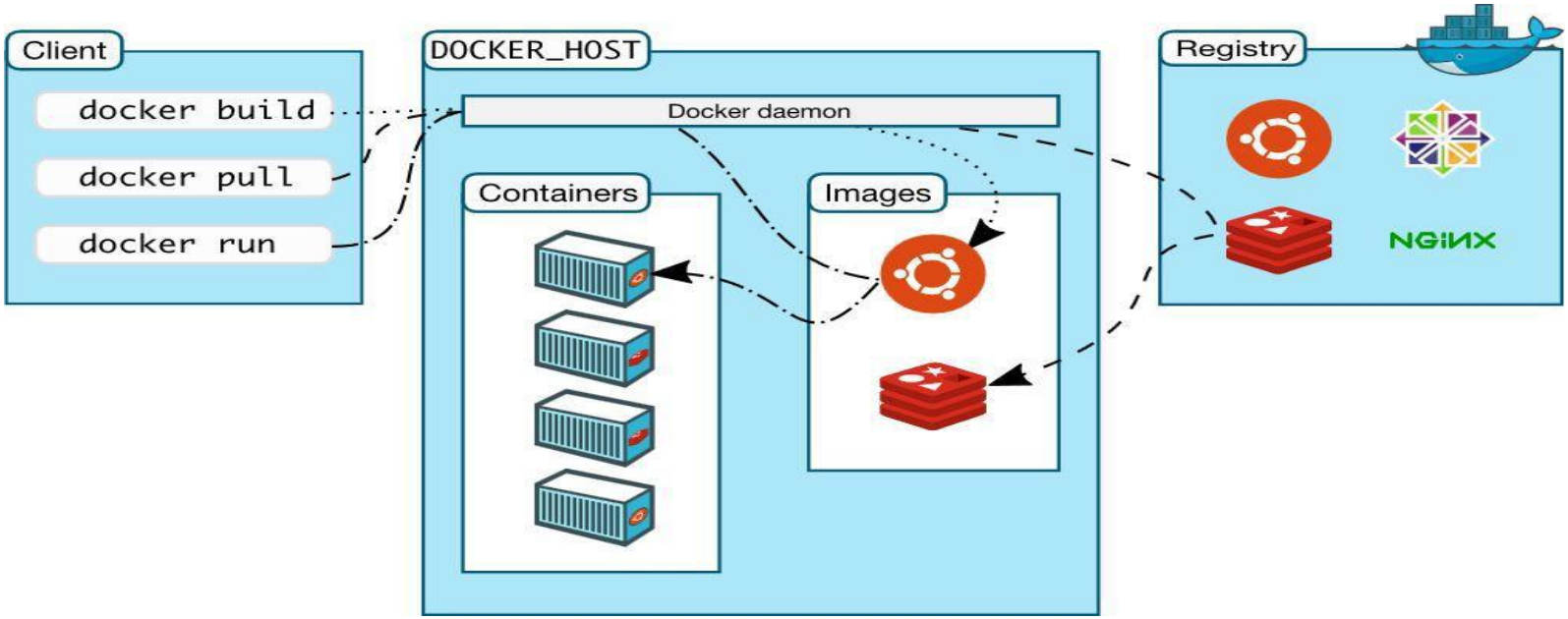

Container A
Push
Docker Container Image Registry
Search
Pull
Run
Build
Dockerfile For A
Source Code Repository
Docker Engine
Host 1 OS (Linux)
Container A
Container B
Container C
Docker
Host 2 OS (Linux)

Docker Engine

- start docker service
- Minden „docker parancsot” ez hajt végre
- Nyilvántartja a lokális hoston levő képfájlokat

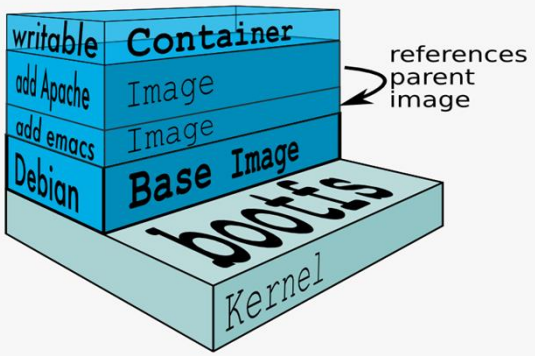


Docker rendszer áttekintés



Docker képfájlok

7

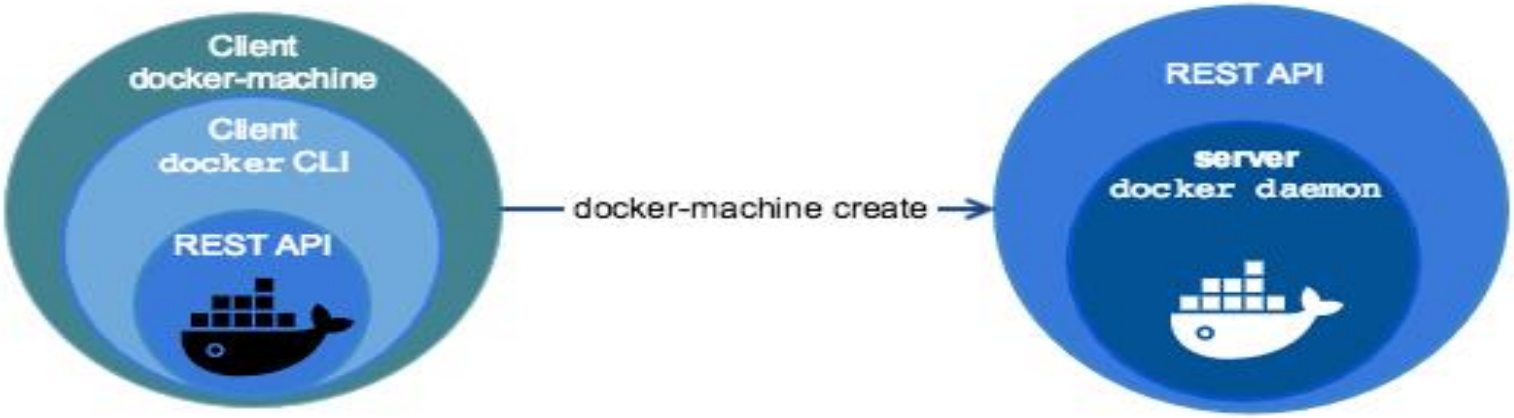


[Source: https://docs.docker.com/terms/layer/]

- Rétegektből áll
- Unió típusú fájlrendszer
  - union file system
  - Az összes rétegből egy képfájlt generál
  - A rétegeket tömörítve tárolja
- Sablon alapján létrehozva
  - Dockerfile
  - Kiinduálsi pint: base image (e.g. ubuntu, fedora, stb.)
  - Saját szinatxis az újabb rétegek hozzáadásához

- Adott képfájl rétegeinek látványos megjelenítése:
  - https://imagelayers.io/

Docker Machine



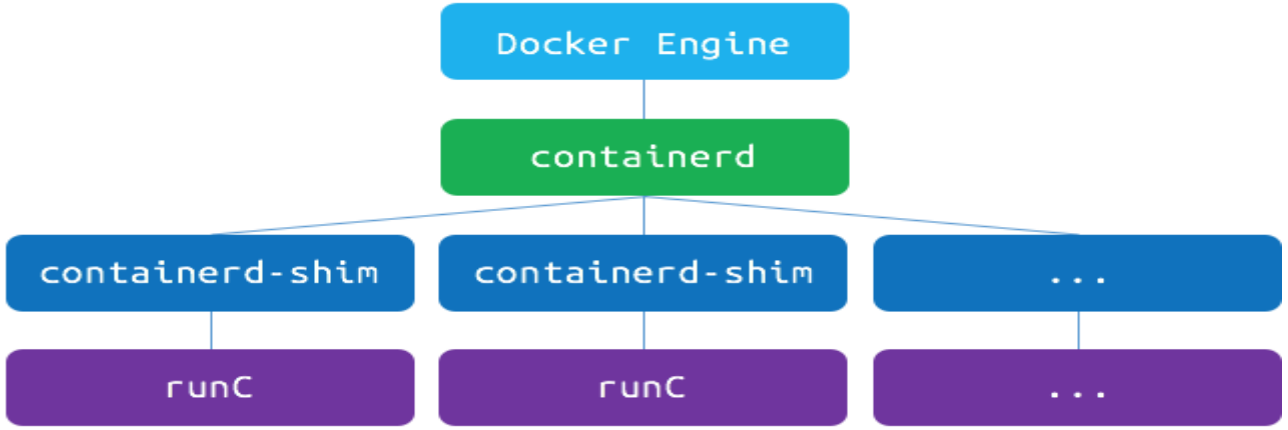
- Távoli állomásokon is lehet konténereket kezelni
  - Automatikus host létrehozás
  - Docker Engine install
  - docker kliens konfig

Docker Machine

- Távoli állomásokon is lehet konténereket kezelni
  - saját parancsa van (docker-machine)

10

Runtime


Docker Engine
containerd
containerd-shim
containerd-shim
...
runC
runC
...

11

Docker Compose

- Összetett feladatok
  - Saját kliens a Docker ökoszisztémán belül
- Több szolgáltatást (konténert) indít egyszerre
- Dockerfile -> alkalmazások sajátosságai
- docker-compose.yml
- docker-compose up

```
version: '2'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Docker workflow 1/2

- Fejlesztés egy dev környezetben (local machine v. container)
- Konténerben futtatni a többi szolgáltatás (services) (pl. adatbázisok)
  - És ugyanúgy működik minden más gépen
    - Kernel verziók
- A „valós” működés tesztelése során :
  - *Másodpercek* alatt fordul (build)
  - *Azonnal* fut

Docker workflow 2/2

- Ha a lokális build OK, akkor
  - Feltölthető a registry-be (public/private)
  - Automatikusan futtatható
  - Üzemi (production, enterprise) környezetben
  - Egyszerű átjárást biztosít a dev és production környezet között
- Hiba esetén: Rollback
  - Vissza lehet térni egy korábbi verzióra

a.) Képfájlok (Docker images) készítése (run/commit megoldással)

- 1) docker run ubuntu bash
- 2) apt install <this> <and that>
- 3) docker commit <containerid> <imagename>
- 4) docker run <imagename> bash
- 5) git clone git://.../mycode
- 6) pip install -r requirements.txt
- 7) docker commit <containerid> <imagename>
- 8) GOTO 4 (ha kell)
- 9) docker tag <imagename> <user/image>
- 10) docker push <user/image>

a.) Előnyök/hátrányok

- Előnyök

– Kényelmes, ismert lépések
– roll back/forward – szükség szerint

- Hátrányok

– Kézi-vezérelt folyamat
– Iteratív változások „felgyűlnek”
– Teljes újrafordítás (rebuild) folyamata sok hibalehetőséget tartogat

b.) Dockerfiles

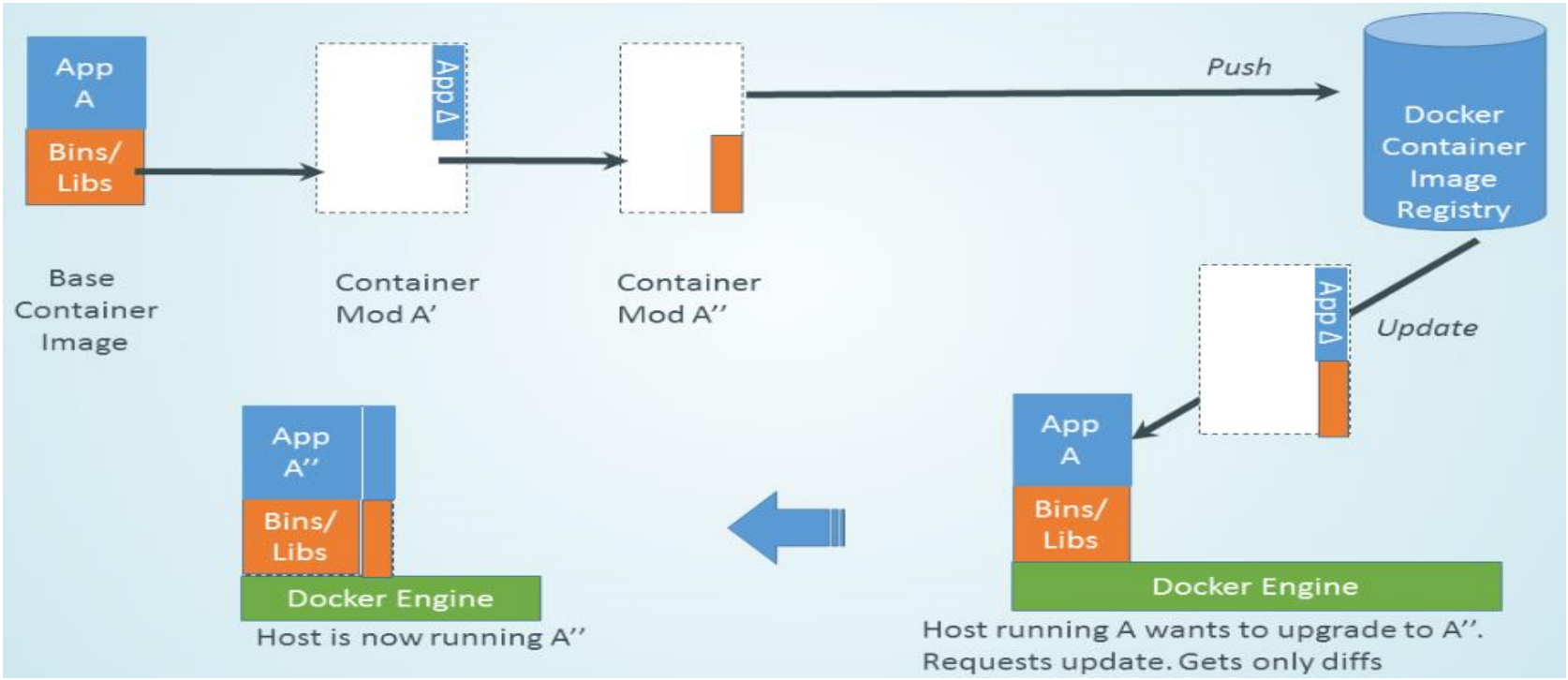
- FROM - egy már létező képfájlból indul ki az új képfájl; ez lesz az alap réteg
  - pl. egy Linux disztribúció
  - Gyakran egy lecsupaszított Linux képfájlt használnak (alpine, busybox)
- COPY – fájlokat másol át a host adott könyvtárából (directory) a képfájlba (pl. konfig, forráskód, szkript)
- RUN – a képfájlba telepítendő, előkészítendő feladatok futtatása
  - Pl. apt update, apt install <program_csomag>, apt
  - Minden külön sor egy külön réteget hoz létre
  - Egymás után felfűzött parancsok („&&” segítségével) egy réteget képeznek
    - Pl. apt update && apt install –y git
- EXPOSE – egy portot nyit majd a konténer számára
- CMD – a konténer indításakor futtatott parancs

- Docker docs on Dockerfile: https://docs.docker.com/engine/reference/builder/

b.) Előnyök

- Gyorsan tanulható
- Könnyen újra-fordítható
  - Caching rendszer segíti ezt
- Egy fájlban meghatározható a build folyamat

Docker – miért gyors?



A Docker rendszer

- docker-ce és docker-ee
  - A korábbi ingyenes docker helyett docker-ce
  - Fizetős, felhasználói támogatással: docker-ee
- Multi-arch, multi-OS
- Stabil kontroll API
- Stabil plugin API
- Hibatűrés (resiliency)
- Aláírással ellátott
- Klaszeterezhető

Docker vs. VM

20

- **Latency**: Applications with a low tolerance for latency are going to do better on physical. This something we see quite a bit in financial services (trading applications are prime example).
- **Capacity**: VMs made their bones by optimizing system load. If your containerized app doesn't consume all the capacity on a physical box, virtualization still offers a benefit here.
- **Mixed Workloads**: Physical servers will run a single instance of an operating system. So, you if you wish to mix Windows and Linux containers on the same host, you'll need to use virtualization
- **Disaster Recovery**: Again, like capacity optimizations, one of the great benefits of VMs are advanced capabilities around site recovery and high availability. While these capabilities may exist with physical hosts, the are a wider array of options with virtualization.
- **Existing Investments and Automation Frameworks** : A lot of the organizations have already built a comprehensive set of tools around things like infrastructure provisioning. Leveraging this existing investment and expertise makes a lot of sense when introducing new elements.
- **Multitenancy**: Some customers have workloads that can't share kernels. In this case VMs provide an extra layer of isolation compared to running containers on bare metal.
- **Resource Pools / Quotas**: Many virtualization solutions have a broad feature set to control how virtual machines use resources. Docker provides the concept of resource constraints, but for bare metal you're kind of on your own.
- **Automation/APIs**: Very few people in an organization typically have the ability to provision bare metal from an API. If the goal is automation you'll want an API, and that will likely rule out bare metal.
- **Licensing Costs**: Running directly on bare metal can reduce costs as you won't need to purchase hypervisor licenses. And, of course, you may not even need to pay anything for the OS that hosts your containers.

Docker előnyei

- Könnyű installációs folyamat
- Minden alkalmazás fut rajta, sok környezetben
- Ismételhető build folyamat
- Nagy hype, erős közösség, gyors javítások
- Új virtualizációs folyamatok

- **Hátrány**
- A Docker konténer típusa
  - A gazdarendszer OS-e határozza meg
- „Orchestration”
- Hálózati kommunikáció

Docker hátrányai

- A Docker konténer típusa
  - A gazdarendszer OS-e határozza meg
- „Orchestration”
- Hálózati kommunikáció

- De: jelentős és még mindig aktív fejlesztések az elmúlt félévben is
  - A hype és erős közösség előnye

23

Biztonság?

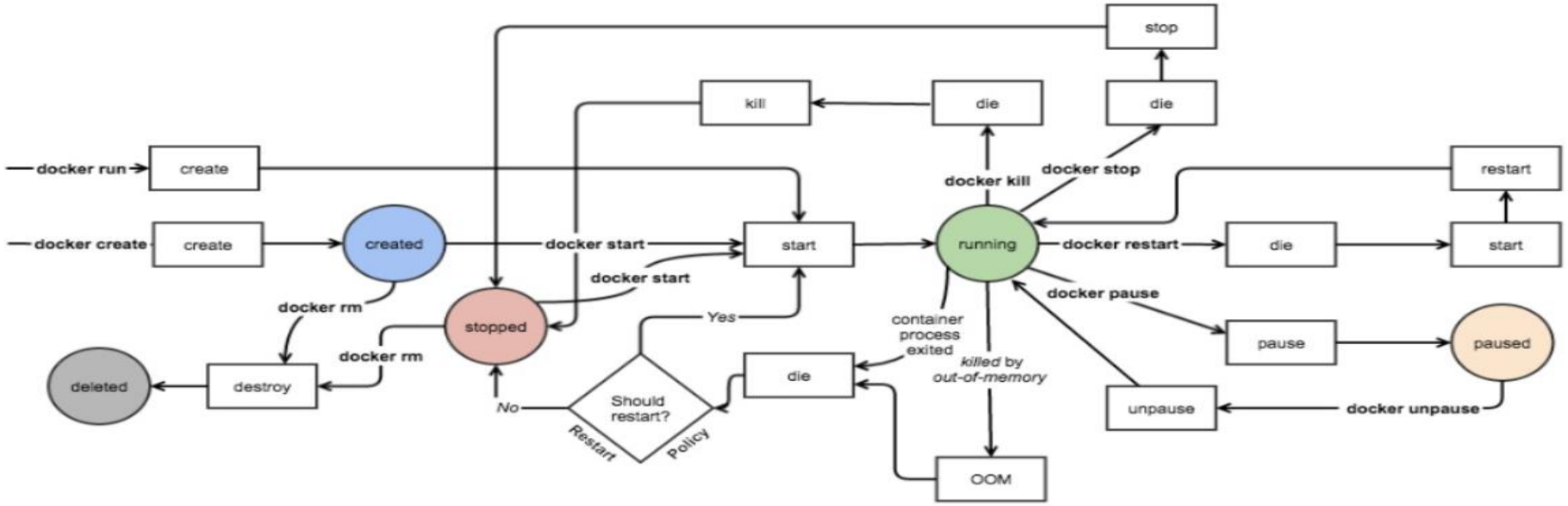
- Docker elérése REST API-n keresztül HTTP felett?
  - Authentikáció!
- Docker démon root jogokkal fut
  - A konténerek már OK
  - „visszanyúlhatnak”?
- docker-1.3 --cap-add, --cap-drop
  - man capabilities
  - „overview of Linux capabilities”
  - „Starting with kernel 2.2”
  - „per-thread attribute”
- Várhatóan további változások lesznek
  - Docker démon

Források

- Docker történet dióhéjban:
  http://www.infoworld.com/article/3025870/paas/the-sun-sets-on-original-docker-paas.html
- Docker áttekintés:
  http://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment
- „The Docker Book”
  http://www.dockerbook.com/#toc
- Docker Meetup @Budapest
  http://www.ustream.tv/recorded/60277876

Docker állapotgép

25



- die, kill ≠ destroy