Docker

- **Szerző:** Sallai András

- Web: http://szit.hu

A Dockerről

A Docker egy adott operációs rendszert virtualizál számunkra, mindenféle virtuális gép, virtuális merevlemez létrehozása nélkül. Operációs renszer szintű, vagy **konténeres virtualizáció**nak is hívják. Ilyen célokra Linuxon csak az LXC állt rendelkezésünkre. Kezdetben maga a docker is ezt használta. A docker célja volt az LXC használatának egyszerűsítése. Ma a Dockernek saját konténerrendszere van. Használatával leegyszerűsíthető új programjaink tesztelése elszeparált környezetben. Nem szükséges külön kernelt indítani.

Mivel jobb ez, mint egy virtuális gép? Kevesebb helyet foglal egy virtuális vendégrendszer, gyors létrehozás, gyors működés, kevesebb helyfoglalás – szemben a teljes virtuális megoldásokkal.

A docker használatához szükség van egy lemezképre (image) az adott operációs rendszerből. Ez tulajdonképpen egy előtelepített rendszer. Ezek elkészíthetők vagy letölthetők a docker úgynevezett registry webhelyéről. A tárolóban általában 64 bites rendszereket tárolnak. Ha 64 bites rendszeren szeretnéd futtatni az adott rendszert, akkor ez nem gond. Egyébként keresned kell 32 bites képfájlokat.

Docker beszerzése

Telepítés

```
apt install docker.io
```

Több információ:

- https://docs.docker.com/engine/installation/linux/debian/

Képfájl beszerzése

Jelenleg egyetlen képfájlunk sincs. Ezt ellenőrizhetjük az images alparanccsal:

```
docker images
```

Lássuk a képfájlok beszerzését, amelyekből ezt követően rendszert indíthatunk. Elsőként 64 bites Debianos képfájl beszerzését nézzük meg. Ha 32 bites a rendszerünk akkor keressük meg lejjebb a 32 bites képfájl beszerzését.

Ha 64 bites rendszerünk van, akkor csak írjuk be:

```
docker run debian:buster /bin/echo 'Hello Vilag'
```

A parancs letölti a debian képfájlt, majd futtatja rajta a megadott parancsot, majd kilép.

A letöltött képfájlokat a következő paranccsal nézhetjük meg:

```
docker images
```

Egy képfájl törlése:

```
docker rmi tarolonev:tag
```

Például:

```
docker rmi debian:buster
```

32 bites képfájl

Figyelem! A Docker saját tárolóiban 64 bites képfájlok vannak. 32-bites képfájlokat nekünk kell készíteni. Ehhez is van segítség:

- https://github.com/docker-32bit

Töltsük le a megfelelő scriptet és futtassuk. Például Debian Jessie:

```
git clone https://github.com/docker-32bit/debian.git
debian/build-image.sh
```

A kimeneten figyeljük a „docker import” sort, mert itt látjuk, milyen néven tudunk majd rá hivatkozni. Például:

```
+ docker import - 32bit/debian:jessie
```

Indítsuk el a docker démont Linux Mint rendszeren:

```
service docker.io start
```

Debian rendszeren eleve fut a démon, nem szükséges indítani.

Teszt:

```
docker run 32bit/debian:jessie /bin/echo 'Hello Vilag'
```

32 bites fedora:

- https://registry.hub.docker.com/u/marina/fedora21-i386/

```
docker run --rm -t -i  marina/fedora21-i386 /bin/bash
```

Itt kereshetünk további képfájlokat:

```
https://registry.hub.docker.com/
```

Használat

Ha már töltöttünk le képfájlt, akkor indíthatunk belőle rendszert. Az elindított rendszert konténernek nevezzük. Egy konténert két módon indíthatunk:

- futtatás után törlődik
- futtatás után megmarad (változások maradnak)

A konténerek alapértelmezésként megmaradnak, tehát mentésre kerülnek. A konténerben futó operációs rendszeren minden munkánk megmarad. Ha az első módon, csak ideiglenesen szeretnénk egy új rendszer indítani, akkor szükségünk lesz a --rm kapcsolóra.

A következő példában egy szimpla Debiant indítunk, amely leállítás után is megmarad.

Debian indítás példa:

```
docker run -it debian
```

Vagy:

```
docker run -t -i 32bit/debian:jessie /bin/bash
```

A run alparancs, egy új konténert hoz létre. A -t azt jelenti kértünk egy terminált, a -i pedig interaktív módban indítottuk a dockert. A „32bit/debian:jessie” rész mondja meg, hogy mit szeretnénk indítani. A „/bin/bash” mondja meg, hogy az elindított Debian operációs rendszeren induljon egy Bash parancsértelmező.

Ha elindult a konténer a fenti paranccsal, akkor megkapjuk a Bash várakozásijelét. Egy ps ax paranccsal ellenőrizzük, hogy valóban egy önálló konténerben fut-e a bash:

```
ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 /bin/bash
    8 ?        R+     0:00 ps ax
```

Ha a /bin/bash PID száma 1-es akkor működik. Debian 9 esetén a procps nincs telepítve, ezért a ps parancs nem működik. Telepítsük:

```
apt-get install procps
```

A run alparanccsal mindig egy új konténert hozunk létre.

A gazda gépen a ps alparanccsal lekérdezhetők a futó konténerek:

```
docker ps
```

A kimenet ehhez hasonló:

```
CONTAINER ID        IMAGE                  COMMAND             CREATED
STATUS              PORTS                NAMES
4a57ca04a052        32bit/debian:jessie   /bin/bash           8 seconds ago
Up 8 seconds                            kamilla
```

Ha kilépünk a konténerben futó Debianból az „exit” paranccsal, akkor a konténer leáll. A futó konténereket a következő paranccsal tekinthetjük meg:

```
docker ps
```

Az összes konténer megjelenítése:

```
docker ps -a
```

Egy létező konténer indítása:

```
docker start kontenernev
```

Egy konténer leállítása:

```
docker stop kontenernev
```

A konténer neve helyett, megadható a konténer azonosítója is.

Az induló rendszer elnevezése és gépnév:

```
docker run --hostname {hostname} --name {name} {image}
```

Konténerek törlése

Ha kiléptünk a konténerben futó Linuxból az exit paranccsal, a konténer megmarad, ha indításkor nem használtuk a --rm kapcsolót.

A konténer törlése az azonosítója alapján történhet. Ha nem tudjuk az azonosítót, lekérdezhetjük az összeset a következő paranccsal:

```
docker ps -a
```

Ezek után törlés, például:

```
docker rm 6b2687bbab4d
```

Ha konténer létrehozásakor elvolt nevezve, akkor a név alapján is törölhetem. Ha a konténer neve például „csirke” volt:

```
docker rm csirke
```

Minden leállított konténer törlése:

```
docker rm $(docker ps -a -q)
```

Konténer átnevezése

```
docker rename CONTAINER NEW_NAME
```

A CONTAINER lehet azonosító is.

Képfájlok átnevezése

```
docker tag szerver:latest sajatnev/szerver:latest
```

vagy

```
docker tag 53fa3634ea35 sajatnev/szerver:latest
```

Névtelen képfájl törlése:

```
docker rmi $(docker images | grep "^<none>" | awk '{print $3}')
```

Valójában az azonosító alapján töröljük.

Képfájlok keresése

Keressünk CentOS konténereket a központi tárolóban:

```
docker search centos
```

Dockerfile

A Dockerfile hasonló a Makefile-hoz. Leírunk vele egy képfájlt. A képfájlt már egy létező képfájlból szoktunk létrehozni. A Dockerfile tehát arra jó, hogy leírhatunk vele egy új képfájlt.

Lássunk egy nagyon egyszerű képfájlt:

```
FROM 32bit/debian:jessie

RUN apt-get -y install mc
```

Az első sor jelzi, hogy melyik már létező képfájl alapján szeretnék létrehozni. Esetünkben ez egy debian. A második és egyben utolsó sor jelzi, hogy futtatni kell a képfájl létrehozása során az „apt-get -y install mc” parancsot.

Megadható a karbantartó és az utolsó frissítés:

```
FROM 32bit/debian:jessie

MAINTAINER Nagy József "nagyjozsef@valahol.and"

ENV REFRESHED_AT 2015-07-21

RUN apt-get -y install mc
```

A RUN kulcsszó után a megadott parancs lehet akár többsoros is. Több soros parancsra példa:

```
RUN mkdir -p /valahol && \
    cd /valahol && \
    touch egy.txt
```

Az aktuális könyvtárból saját fájlokat adhatunk a képfájlhoz. Ilyenek lehetnek a konfigurációs fájlok.

Fájlok hozzáadása a képfájlhoz:

```
ADD fajlok/apache/apache2.conf /etc/apache2.conf
```

Előtte persze illik menteni:

```
RUN mv /etc/apache2/apache2.conf /etc/apache2.conf.eredeti
ADD fajlok/apache/apache2.conf /etc/apache2.conf
```

Ha szeretnénk egy szolgáltatást adott porton elérni a futó konténeren, akkor az EXPOSE direktívával után meg kell adni a portot.

A 80-as port exportálása:

```
EXPOSE 80
```

Ha elindítanak a képfájllal egy konténert, akkor még ez a parancs fusson le:

```
CMD '/usr/local/bin/init.sh
```

A parancs a konténeren belül fut. Ide elhelyezhetünk indító utasításokat, mint az apache, mysql, stb. Hiszen a konténerben nincs init rendszer.

A képfájl elkészítése:

```
docker build -t ujlemekepnev .
```

A „.” azt jelenti, hogy az aktuális könyvtárból adunk hozzá, állományt ha van ilyen.

Ha 32 bites rendszeren dolgozunk, itt megint gond lehet, ha le van töltve 64 bites képfájl, mert az esetleges telepítéseket (apt-get install nano), ebből a képfájlból próbálja meg a rendszer. Ezért töröljük a 64 bites képfájlokat. Például:

```
docker rmi debian:jessie
```

Ha elkészült teszteljük. Hogy minek kell indulnia a /usr/local/bin/init.sh scritben meg lett adva, ezért indításkor nem kell megadni:

```
docker run -i -t ujlemezkepnev
```

Ha volt port is kihelyeve, akkor indítsuk így:

```
docker run -p 1111:80 -i -t ujlemezkepnev
```

Ha nem adjuk meg a 1111 portot akkor létrehoz egyet automatikusan.

Script futtatása induláskor

Dockerfile

```
...

COPY start.sh /

ENTRYPOINT ["/start.sh"]
```

start.sh

```
#!/bin/bash
service ssh start
ip address show eth0 | grep "inet "
```

Adjunk futtatási jogot a start.sh scriptre:

```
chmod +x start.sh
```

Dokcer felhasználóként

Létre kell hozni egy docker csoportot ha az nem létezik:

```
$ sudo groupadd docker
```

Hozzá kell adnunk a kívánt felhasználót:

```
$ sudo gpasswd -a ${USER} docker
```

A ${USER) részt cseréljük ki a felhasználónévre. Jelentkezzünk ki és vissza, felhasználóként.

Indítsuk újra a dokcer démont:

```
$ sudo service docker restart
```

A felhasználó konténeri közös helyen található fizikailag:

```
/var/lib/docker/containers
```

Minta

Ha már van egy 32-bit debian jessie, és szeretnénk valamit kipróbálni akkor:

```
docker run --rm -t -i 32bit/debian:jessie /bin/bash
```

Ha elindult a rendszer, mehet a próba. Ha kiléptünk a rendszerből, a rendszer nincs többé.

Segítség

Linux Mint rendszeren a docke.io kézikönyv van segítségünkre:

```
man docker.io
```

Az egyes alparancsokat lekérdezhetjük:

```
docker
```

A docker kapcsolóit a következő módon kérdezhetjük le:

```
docker --help
```

Az alparancsokhoz újabb kapcsolók tartozhatnak. A „run” alparancs kapcsolóit például így kérdezzük le:

```
docker run --help
```

Segítség a hibafelderítéshez

A hiba tünete:

```
System error: exec format error
```

Ok: 64 bites rendszert próbáltál meg indítani 32 bites rendszeren.

Függelék

Rebuild

rebuld.sh

```
#!/bin/bash
imageName=debian
containerName=kontenerNev

docker build -t $imageName -f Dockerfile  .

echo Régi konténer törlése...
docker rm -f $containerName

echo Új konténer futtatása...
docker run -d -p 5000:5000 --name $containerName $imageName
```

Forrás:

- https://stackoverflow.com/

Rendszerjogok

Biztonsági okokból egy elindított rendszeren nem lehet néhány dolgot megtenni. Ilyen a gépnév megadása. Erre adhatunk jogot a --cap-add SYS_ADMIN kapcsolóval. Bánjunk ezzel óvatosan, mert így esetleg ki lehet törni a virtuális rendszerből.

```
docker run --rm -it --cap-add SYS_ADMIN debian
```

További információk a CAPABILITIES-ről:

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

A konténer újraindítása után a gépnév így sem marad meg. A gépnevet az első futtatáskor megadhatjuk a már fentebb említettek szerint:

```
docker run -it --hostname bar --name bar debian
```

Locale

```
apt-get install locales
export LANG=hu_HU.UTF-8
dpkg-reconfigure locales
echo "export LANG=hu_HU.UTF-8" >> ~/.bashrc
```

Forrás:

- https://wiki.debian.org/ChangeLanguage

Hálózat

A Debian rendszeren telepítve van az iproute2 csomag, amely tartalmazza az ip parancsot. Az Ubuntu rendszeren viszont nincs. Debian rendszeren az iproute2 csomag előtt a net-tools csomag volt használatos, amelyben megtalálható a ifconfig parancs is.

Kapcsolódás háttérben futó konténerhez

```
docker attach [OPTIONS] CONTAINER
```

Linkek

- http://docs.docker.com/
  - http://docs.docker.com/installation/ubuntulinux/
  - https://docs.docker.com/installation/debian/
  - http://docs.docker.com/windows/step_one/ (Winre)
- https://registry.hub.docker.com/
  - https://registry.hub.docker.com/_/centos/
- http://jmkeyes.github.io/post/creating-centos-6-docker-image/