Poprvé s Dockerem

Docker je softwarový systém, cloud a firma podporující vývoj, šíření a spouštění aplikací v kontejnerech. Kontejnery (containers) v Linuxu umožňují sbalit do uzavřeného prostředí aplikaci a vše potřebné pro její běh (podobně jako virtuální stroje), jen linuxové jádro nikoliv (na rozdíl od virtuálních strojů). Tím se šetří hardwarové zdroje (staticky alokovaná CPU jádra a paměť, duplicitní operační systém). Autor softwaru vyvine (kontejnerizuje) ze standardních výchozích kontejnerů kontejner se svou aplikací, vytvoří jeho obraz (image, spustitelný balíček) a umístí jej na volně dostupný cloud neboli registr, např. Docker Hub, quay.io aj. Uživatelé odtud obraz stáhnou a spouští z něj své kontejnery. Kontejnery jsou až na přístup k jádru a síťový kanál izolovány od okolního softwarového prostředí a pro kontakt s vnějškem je třeba definovat komunikační kanály (sdílené adresáře, porty). Software i vývojové prostředí v kontejnerech si uživatelé mohou podle chuti doplnit a uložit jako vlastní obrazy. Lze tak např. na strojích se starší verzí linuxové distribuce vyvíjet a spouštět kontejnery založené na novější distribuci a jejích aktuálních balíčcích.

Docker je doma v Linuxu. V Ubuntu se obstará pomocí balíčků docker-engine nebo docker-ce (firemní originály pro Ubuntu), nikoliv docker.io (spíše zastaralý balíček z distribuce Ubuntu):

```
apt-get install docker-engine     v Ubuntu 14.04
apt install docker-ce             v Ubuntu 18.04
```

V obou případech je třeba instalaci nasměrovat na správné repozitáře pomocí tipů např. odsud:

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04

Tím se zajistí přítomnost běžícího servisu neboli démona (někdy je třeba restartovat: sudo service docker restart) a řádkového klienta neboli příkazu docker. Docker je připraven i pro Mac a Windows (Docker Desktop). Uživatel v Linuxu je standardně v kontejneru rootem, proto buď spouští kontejner pomocí sudo nebo se zařadí do uživatelské skupiny docker (id -Gn, /etc/group, …).

První doteky

Příkazy klienta pro zjištění verze, přítomných obrazů apod.:

```
docker --version        Docker version 17.05.0-ce, build 89658be
docker version          více o Client a Server
docker info             více o Containers, Images, Root Dir, operačním systému a hardwaru
docker                  pro seznam příkazů a voleb
docker run --help       pro návod ke klientskému příkazu run
```

Obrazy hello-world a ubuntu a co s nimi

K vyhledávání obrazů na Docker Hubu podle klíčových slov je určen příkaz klienta docker search, např.

```
docker search ubuntu
```

K získání přehledu o již instalovaných obrazech na vlastním stroji slouží:

```
docker images
```

Pro otestování funkčnosti poslouží elementární obraz hello-world (lze doplňovat názvy obrazů pomocí tabelátoru):

```
docker run hello-world
```

kterým klient osloví server, ten nalezne lokálně, případně stáhne z Docker Hubu požadovaný obraz (je tedy zbytný explicitní download pomocí docker pull hello-world), vytvoří z něj kontejner, v něm se spustí proces a ten pomocí klienta vypíše text na obrazovku. Součástí textu je návrh dalších kroků, jako číst dokumentaci nebo spustit v interaktivním režimu (volby -it) kontejner ubuntu s čerstvým Ubuntu:

```
docker run -it ubuntu
```

Tím se ocitneme uvnitř kontejneru. Jak bychom měli čekat, jádro je z našeho vnějšího systému,

```
uname -a                pro Linux be35bd712842 3.13.0-101-generic …
```

ale výbava v kontejneru je z čerstvé distribuce:

```
cat /etc/lsb-release    pro výstup obsahující slova 18.04, bionic, Ubuntu 18.04.2 LTS ad.
cat /etc/os-release     pro více údajů
```

Distribuce je čerstvá, ale celkem holá. Síťový kanál je funkční, můžeme tedy instalovat balíčky:

```
apt update                  pro občerstvení indexu databáze balíčků
apt install vim mc gfortran pro pokusy v textovém režimu
apt install firefox gnuplot pro pokusy v grafickém režimu
```

Poté lze už něco vyvinout, např. vytvořit program a spustit proces na popředí i pozadí:

```
cd /home; vi a.f90; gfortran a.f90; ./a.out; Ctrl-Z; bg
```

Z jiného okna lze pozorovat aktivity dockeru zvnějšku,
top pro seznam aktivních procesů uvnitř kontejnerů i mimo ně
a zevnitř kontejneru vidíme pomocí top podobné údaje (PID je jiný). Poslat procesu kill lze zevnitř i zvnějšku.
Je vhodná chvíle zacházet explicitně se jmény kontejnerů,
docker ps; docker ps -a pro seznam aktivních, resp. všech kontejnerů
docker container ls -a patrně totéž
docker pause be35bd712842 pro pozastavení kontejneru pomocí ID
docker unpause silly_wilson pro pokračování kontejneru pomocí systémem voleného NAME
docker rename silly_wilson myCont pro vlastní nastavení jména kontejneru
Zevnitř aktivního kontejneru můžeme odejít (detach) při zachování jeho aktivit pomocí klávesové kombinace,
Ctrp-p Ctrl-q
a vrátit se do něj příkazem
docker attach myCont
To lze udělat paralelně i z jiného okna a vidět pak v obou oknech totéž.
Kontejner lze restartovat; doposud instalované balíčky a vytvořené soubory jsou zachovány, běžící procesy zrušeny:
docker restart myCont
Zevnitř lze kontejner ukončit pomocí exit nebo Ctrl-D, zvenku pomocí
docker stop myCont
Ukončený kontejner lze znovu spustit:
docker start myCont; docker attach myCont
anebo definitivně smazat:
docker rm myCont
Ostřejší cestou je smazat všechny neaktivní kontejnery:
docker container prune

Alternativou k výše uvedeným krokům je nejprve interaktivní kontejner s vlastním jménem připravit a teprve poté jej spustit, včetně volby pro jeho okamžité smazání po ukončení,
docker create -it --name myCont --rm ubuntu
docker start myCont; docker attach myCont
Neprodlené zopakování příkazu create vede k odmítnutí kvůli konfliktu jména kontejneru myCont, zatímco příkazem
docker create -it --name myCont2 --rm ubuntu
se z téhož obrazu vytvoří další, nezávislý kontejner s novým jménem.

Pro více informace o dění v kontejnerech poslouží příkazy:
docker stats; docker inspect myCont; docker top myCont

Vytvoření vlastních obrazů

Ve vlastním kontejneru odvozeném od výchozího obrazu (jako výše) ručně vytvoříme požadované prostředí a to pak zafixujeme jako lokální obraz následujícím příkazem,
docker commit -m 'msg' -a 'author' container_id repository/name
Nový obraz můžeme odeslat na cloud (příkaz docker push) a na jiném stroji si jej z cloudu vzít (docker pull).
Přenášet obrazy lze i bez cloudu pomocí tar souborů:
docker save -o file.tar images pro vytvoření tar souboru z jednoho nebo více obrazů
docker load -i file.tar pro instalování obrazů z lokálního tar souboru
Obrazy lze vytvářet i pomocí skriptu, tzv. dockerfile, příkazem
docker build …
Uživatel tedy stahuje kontejner s připravenou aplikací z Docker Hubu či jiného registru nebo stahuje dockerfile např. z gitu a z něj vytváří kontejner pomocí docker build (viz odkaz níže pro zajištění ssh přístupu do kontejneru).

Sdílení dat s hostitelem

Uživatel může při vytváření kontejneru z obrazu specifikovat sdílený adresář uvedením jeho jména v systému a v kontejneru,
docker run -it -v host_dir:container_dir image
Atribut :ro (read-only) zajistí nepoškoditelnost dat ve sdíleném adresáři, viz níže spouštění projektu Aspect. Po vytvoření kontejneru k němu nelze přiidávat další sdílené adresáře.
(Opomenuté) sdílení adresářů lze nahradit ručním kopírováním dat mezi systémem a kontejnerem,
docker cp files container:dir pro kopírování souborů ze systému do kontejneru
docker cp container:file . pro kopírování souboru z kontejneru do aktuálního adresáře

Grafické aplikace

Dostane-li kontejner proměnnou s adresou pro grafický výstup a sdílený adresář pro X11 socket, je schopen posílat grafický výstup na lokální linuxový displej (nebo vzdálenému NX klientovi). Nezbytné je povolit přístup k X-serveru:

```
xhost +                                   pro otevření přístupu k X serveru; inverzní příkaz je xhost -
docker run -it -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix ubuntu
apt update; apt install gnuplot; gnuplot -e "plot x**2; pause 2"   pro test výstupu grafiky z kontejneru
```

Připojením --device /dev/dri lze prý zpracovat grafiku z kontejneru s hardwarovou akcelerací.
Pomýšlet lze i na grafický kontakt linuxového stroje se vzdáleným kontejnerem, což ovšem vyžaduje více úsilí.

Akustické aplikace

Kontejner může dostat kontakt na audio driver a být tak schopen přehrávat zvuk:

```
docker create -it --device /dev/snd --name mplayer ubuntu        pro vytvoření kontejneru s audio kanálem
docker cp music.mp3 mplayer:/home                                pro přenos audio souborů do kontejneru
docker start mplayer; docker attach mplayer                      pro start a vstup do kontejneru
apt update; apt install mplayer; cd /home; mplayer music.mp3     pro instalaci a spuštění přehrávače
```

a dále `docker commit mplayer local/mplayer` pro vytvoření obrazu z tohoto kontejneru
a pak `docker run -it --device /dev/snd --rm -v ~/mp3:/home/mp3 local/mplayer` pro kontejner se sdílenými daty

Jistě se vyplatí nalézt podstatně vybavenější audio kontejnery pomocí Googlu nebo docker search.

SSH přístup do kontejneru

https://docs.docker.com/engine/examples/running_ssh_service

Kontejnerizované aplikace

FEniCS

```
fenicsproject run                          pro vytvoření kontejneru z aktuální verze obrazu
fenicsproject run quay.io/fenicsproject/stable:2017.2.0   pro kontejner z verze 2017.2.0
dolfin-version                             pro ujištění se o instalované verzi
```

Aspect

```
docker run -it -v "$(pwd):/home/dealii/aspect/model_input:ro" gassmoeller/aspect:latest bash
                                           pro vytvoření kontejneru s read-only vstupním adresářem
./aspect model_input/your_input_file.prm                  pro spuštění laditelné verze projektu Aspect
./aspect-release model_input/your_input_file.prm          pro spuštění produkční verze Aspectu
docker cp container:/home/dealii/aspect/model_output .
                                           pro vynesení výstupních dat z kontejneru do hostitelského systému
```

L. H., 22. 3. 2019