Docker

Radim Kliment
radim.kliment@cvut.cz

Požadavky

- 3 bloky
- Maximální počet bodů 100
- ECTS stupnice
  - A: 100 – 90
  - B: 89 - 80
  - C: 79 - 70
  - D: 69 - 60
  - E: 59 - 50
  - F: 49 - 0

Domácí úkoly

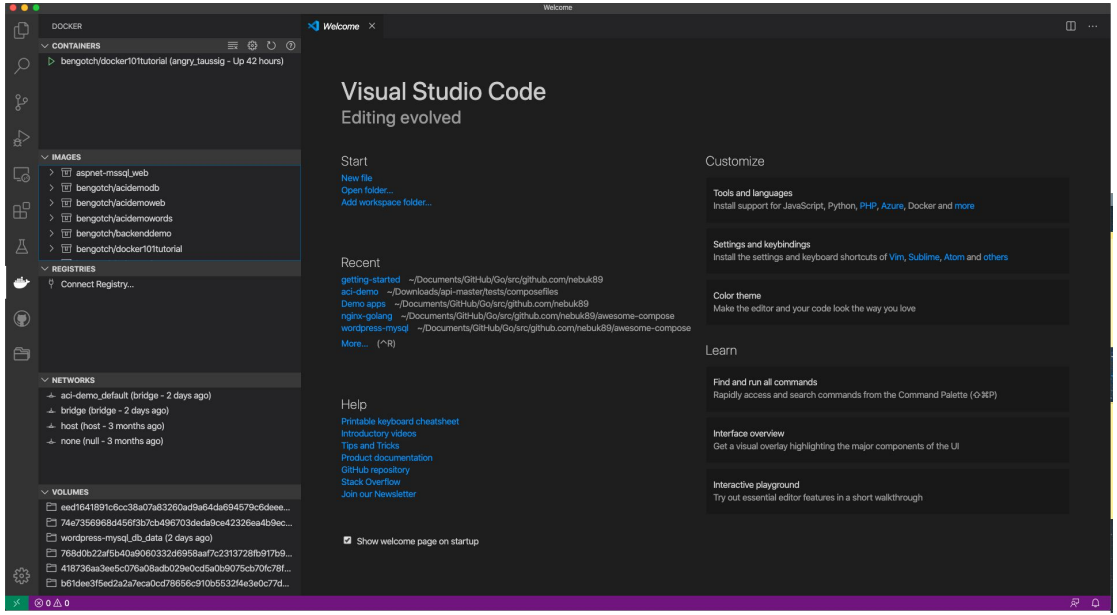
- Každý domácí úkol bude ve vlastní složce
  - odevzdání archivu -> (email, nebo teams)
- Maximum 35 bodů
  - Do pondělí včetně - plný počet bodů
  - Každý následující den -2 body z celkového hodnocení

Docker instalace

- Zapnutá virtualizace v BIOSu
- https://www.docker.com/get-started
  - Linux
    - Docker engine
  - Windows
    - Docker desktop
  - Mac
    - Docker desktop

Docker (doporučení)

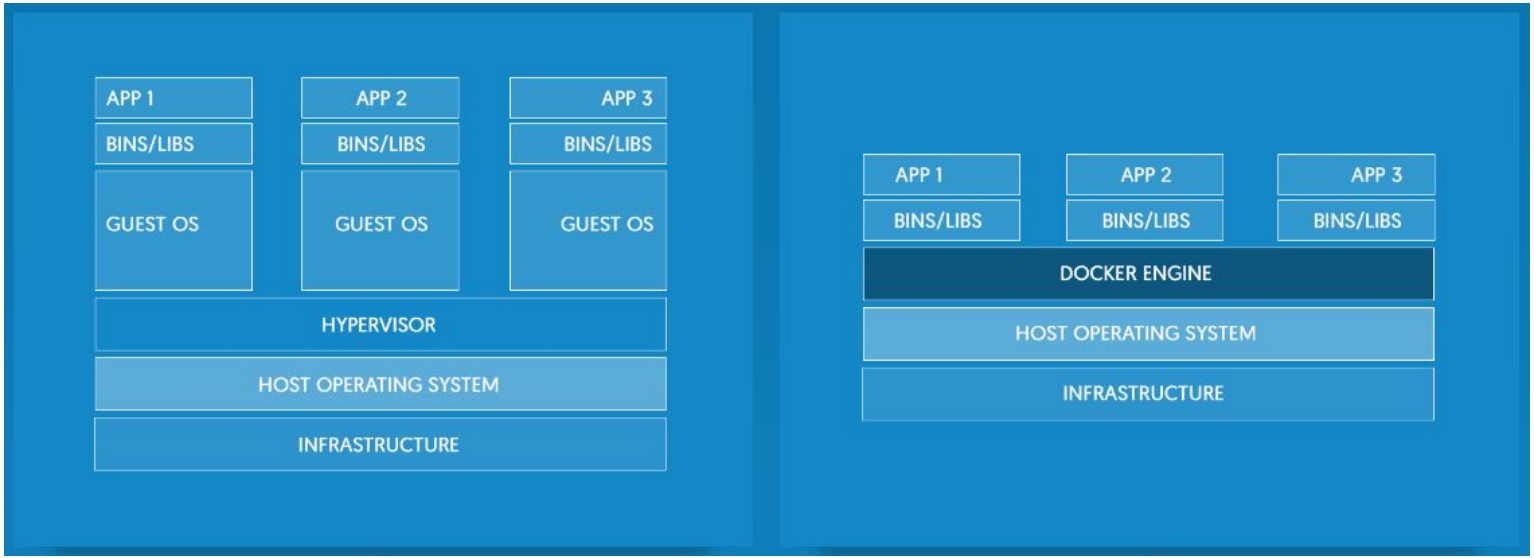
- Visual studio code + Docker plugin
  - https://code.visualstudio.com/



Použití

- Izolování aplikace do vlastního kontejneru
- Závislosti a nastavení izolované od systému (serveru)
  - některé zdroje jsou sdílené s OS
- Reprodukovatelnost
  - Buildů
  - Služeb
- Microservice

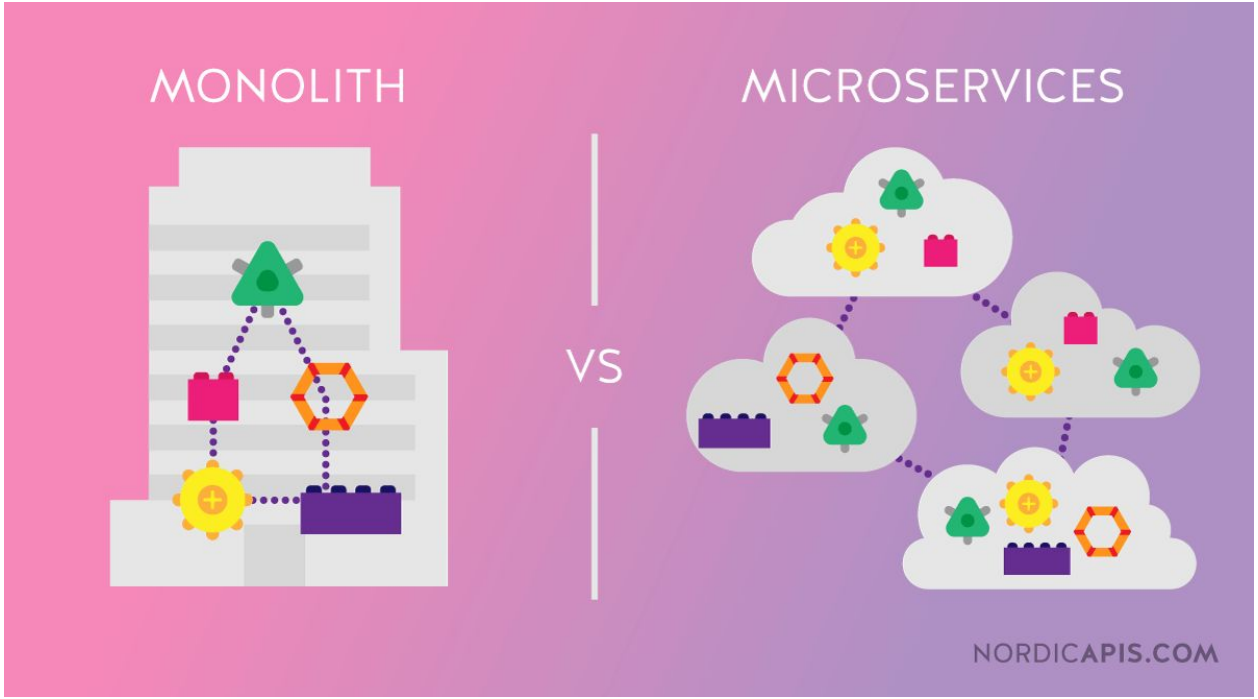
Docker vs Virtualizace

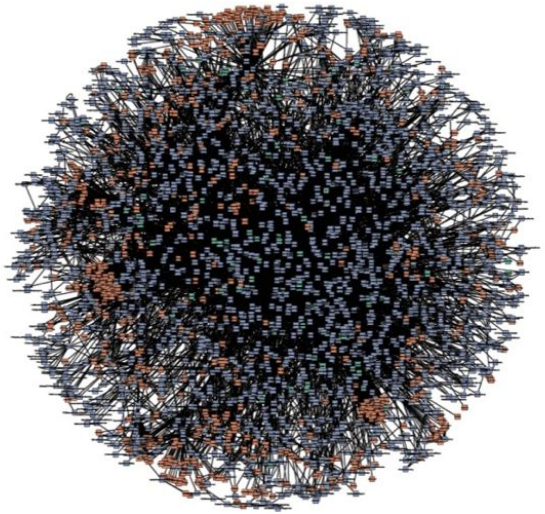


Container vs Virtual

Virtual Machine	Docker Container
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find	Pre-built docker containers are easily available
VMs can move to new host easily	Containers are destroyed and re-created rather than moving
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

Mono vs Micro



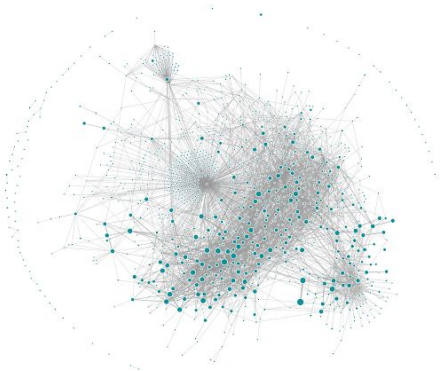


amazon

NETFLIX

Introducing Domain-Oriented Microservice Architecture

Adam Gluck July 23, 2020



Gergely Orosz
@GergelyOrosz

For the record, at Uber, we're moving many of our microservices to what @copyconstruct calls macroservices (wells-sized services).

Použití

- CI (GitLab)
- Build
- Web server
- Lightweight prebuilt apps
- Tests and runtime enviroments

Terminologie

- *Image*
  - Připravená aplikace (service) ke spuštění
  - obsahuje veškeré závislosti
- *Container*
  - Spuštěný *image*

Docker hub

- registry s připravenými image
  - webservery
  - runtime
  - aplikace
  - ...
- https://hub.docker.com/

hello world

- ***docker pull hello-world***
- **docker run hello-world**
  - stáhne *image pokud není na locale*
  - spustí *image* -> vytvoří *container*
  - stdout *container* vypíše do terminálu
- **docker images**
  - vypíše do konzole seznam všech stáhnutých *image*
- **docker container list**
  - vypíše do konzole seznam všech běžících *containerů*

hello-world - dekonstrukce

- minimal runtime (scratch)
- obsahuje binární spustitelný soubor *hello*
- po spuštění image se spustí soubor *hello*
- vypíše se hello-world do stdout

busybox - interactive

- **docker run name:tag**
- **docker run -it busybox**
- **docker run -it python:3.9**
- **docker run -it python:3.6.12**
- **docker run -it python:latest**
- Interaktivní připojení přímo do containeru
- Vhodné například pro testování různých verzí

docker - static website example

docker run -d -v /my/folder:/web -p 8080:8080 halverneus/static-file-server:latest

- -d
  - spustit jako daemon na pozadí
- -v
  - volume
  - připojí složku z local do containeru
- -p
  - publish portů
  - porty containeru jsou pak dostupné z venku
  - host:container

https://hub.docker.com/r/halverneus/static-file-server

docker - static website example

- http://127.0.0.1:8080/
- **docker container list**

docker - postup (image -> container)

- image z registry
  - *docker pull*
  - docker run
- vlastní image
  - docker build
  - docker run

docker stop and remove

- **docker container stop** [id, name]
- **docker container rm** [id, name]
- **docker image rm** [id, name]

docker prune

- vymazání nepoužívaných containers, images, volume
- **docker image prune**
- **docker container prune**
- **docker volume prune**
- **docker system prune -a**
  - *delete all, not only dangling*

docker publish x expose

- **publish** portů otevře porty *z venku* do containeru (-p)
- **expose** otevře porty pouze pro containery na stejné docker síti (--expose)

docker volume

- docker volume list
- mount lokální složky do kontejneru
  - anonymous
    - -v /slozka_v_kontejneru
  - host
    - -v /lokalni_slozka/absolutni_cesta:/slozka_v_kontejneru/absolutni_cesta
  - named
    - -v jmeno_volume:/slozka_v_kontejneru/absolutni_cesta

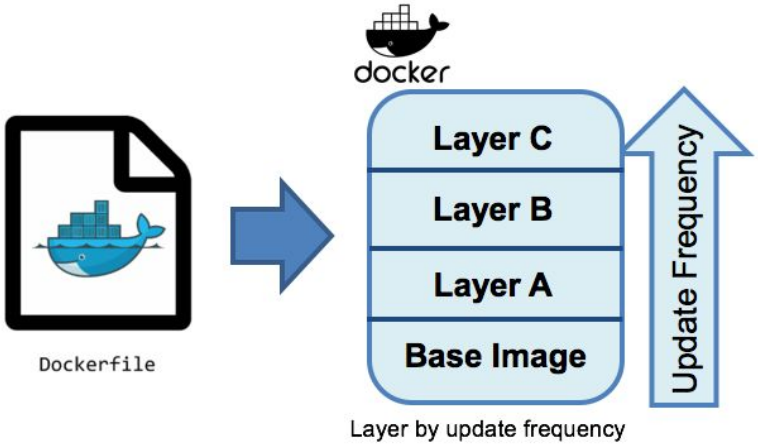
hello-world vs website

- hello-world kopíruje soubory do image
  - nezrcadlí po spuštění následné změny souborů do *containerů*
  - *container* je samostatný
- website mountuje složku do containeru
  - zrcadlí změny po spuštění, lze za běhu měnit obsah souborů

Dockerfile - (python example)

- FROM
- WORKDIR
- COPY
- RUN
- CMD
- VOLUME
- LABEL

Dockerfile - layers



Dockerfile

- soubor s instrukcemi k tvorbě image
  - **docker build -t jmeno_image:tag_image** [directory with Dockerfile]
  - **docker build -t jmeno_image:tag_image .**
  - **docker build -t muj_image:1.0 .**
  - **docker build -t muj_image:latest .**

docker attach

- **docker attach [id, name]**
- připojí *stdout*, *stdin* do běžícího *containeru*
- pokud container loguje, lze takto vypisovat aktuální logy

docker build and run

- docker build -t moje_app_image .
- docker run -d --name moje_app_container moje_app_image

docker exec

- spuštění příkazu v běžícím containeru
  - **docker container exec -it moje_app_container ls /var**
- spuštění interaktivní bash session v containeru
  - **docker container exec -it moje_app_container /bin/bash**

docker push

- publishne váš image do centrálního registry
- **docker push yourusername/imagename:tag**
  - push do centrální registry (docker hub)
- **docker push address.com:5000/imagename:tag**
  - push do vlastni registry

Vlastní registry

- možnost uploadu privátních image
- vhodné například pro použití v intranet infrastruktuře (CI)
  - https://docs.docker.com/registry/
  - gitlab.com

Propojení service (redis-pyapp)

- komunikace mezi containery
  - --link container_name:link_name
- python - redis example

Propojení service - docker-compose

- automatické linkování (v3.8)
- přehlednější
- orchestrace větších projektů

Domácí úkol 1 (ping) - 5b

- Vytvořte příkaz pro jednorázové spuštění docker image **busybox** který provede pomocí utility **ping** 10 ping requestů na libovolný server.
- Příkaz umístěte do souboru **run.sh**

Domácí úkol 2 (web) - 10b

- Vytvořte Dockerfile, který využije image: **halverneus/static-file-server**
- Vytvořte HTML hello world soubor **index.html** a nakopírujte ho dovnitř image do složky **/web**
- Do souboru **run.sh** napište příkazy pro **build** a **spuštění** tohoto image na portu 8080
- Otestujte pomocí adresy http://localhost:8080

Domácí úkol 3 (wget) - 5b

- Pomocí Dockerfile vytvořte image ubuntu 20.04
- Do image nainstalujte nástroj **wget**
- Po spuštění container stáhne pomocí nástroje wget následující RSS a vypíše ho do stdout
  - https://aktualne.cvut.cz/rss/newsflashes

Domácí úkol 4 (wget) - 5b

- Pomocí Dockerfile vytvořte image ubuntu 20.04
- Do image nainstalujte nástroj **wget**
- Vytvořte soubor **run.sh**, který provede **build** a **run** image
- Do containeru v příkazu run připojte lokální složku pomocí volume
- Pomocí nástroje wget stáhněte následující RSS jako soubor do připojeného volume. Bude tak přístupný mimo container.
    - https://aktualne.cvut.cz/rss/newsflashes

Domácí úkol 5 (freestyle) - až 10b

- Vyberte si libovolný vámi preferovaný programovací jazyk, nebo technologii
- Využijte Docker pro jakoukoliv část vývojového procesu
  - běh programu
  - build
  - generování dokumentace
  - ...