Kontejnery

Eliška Jégrová

28. 11. 2025

Obsah

It works on my machine!

- Aplikace často funguje na vývojářově počítači, ale ne na serveru.
- Na každém stroji může být jiná verze knihoven, balíčků nebo konfigurace.
- Kontejner používá **vlastní souborový systém** oddělený od zbytku systému:
  - obsahuje potřebné programy, knihovny a konfiguraci,
  - všude vypadá stejně, bez ohledu na konkrétní stroj.
- Stejný kontejner tedy můžeme spustit na různých počítačích a očekávat **stejné prostředí i chování aplikace**.

Co je kontejner a jak se liší od virtuálky

- **Kontejner**
  - běžící proces s vlastním pohledem na souborový systém,
  - má izolované prostředí (procesy, síť),
  - sdílí stejný kernel s hostitelským systémem.
- **Virtuální stroj**
  - běží v něm celý operační systém včetně vlastního kernelu,
  - emuluje „celý počítač“ (CPU, disk, síťová karta),
  - je těžší na zdroje a startuje pomaleji.

Sjednocení a izolace

- **Sjednocení prostředí**
  - stejný obraz kontejneru můžeme použít na více strojích,
  - ve vývoji, testu i produkci běží aplikace ve stejném prostředí,
  - odpadá „u mě to funguje, ale na serveru ne“ kvůli rozdílným verzím balíčků.
- **Izolace služeb**
  - každá služba může běžet ve svém vlastním kontejneru,
  - chyby nebo špatná konfigurace jedné služby méně ovlivní zbytek systému,
  - v každém kontejneru může být jiná verze programu nebo knihovny, aniž by si navzájem překážely.

Sdílení vrstev

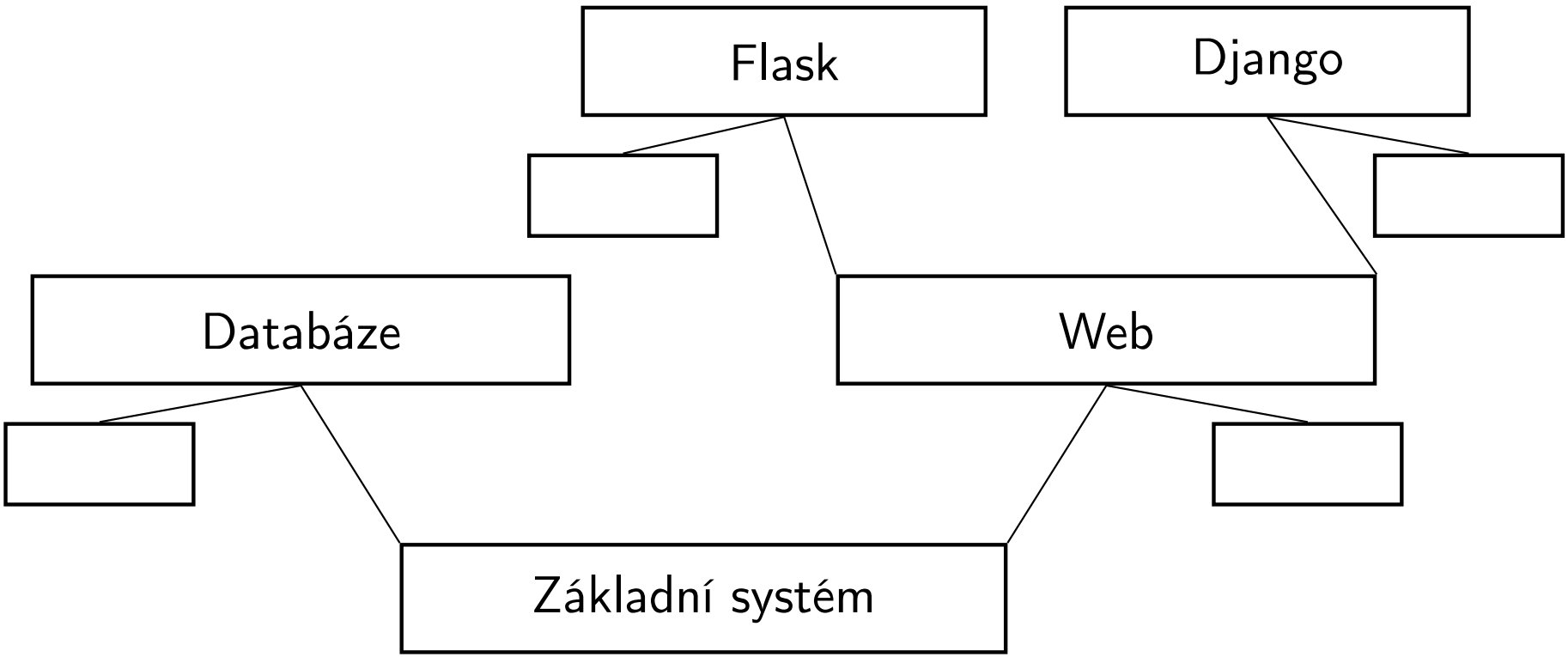
- Vrstvy obrazu jsou **jen pro čtení** a dají se **sdílet**:
  - více obrazů může používat stejné spodní vrstvy,
  - více kontejnerů z jednoho obrazu sdílí všechny jeho vrstvy.
- Každý kontejner má navíc jen svou **zapisovatelnou vrstvu**:
  - tam se ukládají změny (nové soubory, logy, dočasná data),
  - spodní vrstvy zůstávají stejné pro všechny.
- Výhody sdílení vrstev:
  - šetří místo na disku – stejné soubory nejsou uloženy vícekrát,
  - rychlejší stažení nových obrazů – znovu se stahují jen nové vrstvy,
  - rychlejší start kontejnerů – většina dat už je lokálně k dispozici.

Obraz, kontejner a vrstvy

- **Kontejner**:
  - běžící instance vytvořená z obrazu,
  - vypadá jako samostatný souborový systém,
- **Obraz kontejneru** (image):
  - obsahuje nainstalovaný systém / aplikaci a její závislosti,
  - je pouze pro čtení.
- **Vrstvy obrazu**:
  - obraz je složený z několika **vrstev**,
  - každá vrstva obsahuje změny oproti předchozí (přidané soubory, úpravy, smazání),
  - všechny vrstvy obrazu jsou jen pro čtení.
- **Vrstva kontejneru**:
  - kontejner používá všechny vrstvy obrazu,
  - navíc má svou **zapisovatelnou vrstvu** pro změny.

Vrstvy aplikací a systému

- **Sdílení vrstev**:
  - více obrazů může sdílet stejné spodní vrstvy,
  - více kontejnerů ze stejného obrazu sdílí všechny jeho vrstvy,



Docker Hub, Docker a Podman

- **Docker**
  - původní a nejznámější nástroj pro práci s kontejnery,
  - používá vlastní démon (`dockerd`), ke kterému se klient `docker` připojuje.
- **Podman**
  - na Fedoře se používá místo Dockeru,
  - nástroj pro kontejnery, kde jsou příkazy velmi podobné (`podman run`, `podman ps`, ...).
- **Docker Hub**
  - veřejný server (registr), kde jsou uložené obrazy kontejnerů,
  - najdeme tam jak oficiální obrazy (distribuce, databáze, web servery),
  - tak i obrazy vytvořené uživateli.

Stažení obrazu a první kontejner

- Stažení obrazu (pokud ještě není lokálně):

```
$ podman pull ubuntu
```

- Spuštění kontejneru s interaktivním shellem:

```
$ podman run -it --rm ubuntu
```

  - `-it` – interaktivní režim (terminál),
  - `--rm` – po ukončení shellu se kontejner smaže,
  - `ubuntu` – jméno obrazu,

- Spuštění kontejneru na pozadí:

```
$ podman run -d --rm ubuntu
```

Seznam kontejnerů a obrazů

- Běžící kontejnery:

```
$ podman ps
```

- Všechny kontejnery (včetně ukončených):

```
$ podman ps -a
```

- Lokálně dostupné obrazy:

```
$ podman images
```

- Hodí se pro kontrolu:
  - jestli kontejner ještě běží,
  - z jakého obrazu byl vytvořen,
  - které obrazy už zabírají místo na disku.

Zastavení a mazání kontejnerů a obrazů

- Zastavení běžícího kontejneru:

  `$ podman stop <id_nebo_jmeno>`

  - `<id_nebo_jmeno>` získáš z `podman ps -a`

- Smazání (ukončeného) kontejneru:

  `$ podman rm <id_nebo_jmeno>`

- Smazání obrazu:

  `$ podman rmi <image>`

  - `<image>` získáš z `podman images` (může to být jméno nebo ID obrazu),
  - obraz nelze smazat, pokud z něj ještě existují kontejnery – ty je potřeba nejdříve zastavit a odstranit.

Webový server – porty na hostitelském systému

- Zpřístupnění portu 80 z kontejneru na portu 8080 hostitele:

```
$ podman run -p 8080:80 --rm httpd
```

- znamená: `port_hosta:port_kontejneru` → `8080:80`,
- v prohlížeči otevři:

```
http://localhost:8080
```

- nebo z terminálu:

```
$ curl localhost:8080
<html><body><h1>It works!</h1></body></html>
```

Webový server v kontejneru a vlastní obsah

- **Připojení vlastního adresáře**:

```
$ mkdir ~/container_htdocs
$ cd ~/container_htdocs
$ echo Ahoj! > index.html

$ podman run -p 8080:80 \
-v ~/container_htdocs:/usr/local/apache2/htdocs/:Z \
--rm httpd
```

- `-v host_adresar:cesta_v_kontejneru` – připojení adresáře,
- `:Z` – úprava SELinux kontextu, aby měl kontejner k souborům přístup,
- po obnovení stránky na `localhost:8080` uvidíš `Ahoj!`.

Připojení adresáře do kontejneru

- Na hostiteli si připrav vlastní obsah:

```
$ mkdir ~/container_htdocs
$ cd ~/container_htdocs
$ echo Ahoj! > index.html
```

- Spusť `httpd` s připojeným adresářem:

```
$ podman run -p 8080:80
-v ~/container_htdocs:/usr/local/apache2/htdocs/:Z
httpd
```

- `-v host_adresar:cesta_v_kontejneru` – připojení adresáře,
- `:Z` – nastaví SELinux kontext tak, aby měl kontejner k souborům přístup (na Fedoře je to často potřeba),
- po obnovení stránky `http://localhost:8080` uvidíš text `Ahoj!` z hostitelského adresáře.

Dockerfile – vlastní obraz

- **Dockerfile** (nebo neutrálněji `Containerfile`):
  - textový soubor - popisuje jak vyrobit nový obraz,
- Příklad `Dockerfile` založeného na `httpd`:

```
FROM httpd
RUN echo Ahoj > /usr/local/apache2/htdocs/index.html
```

- Vytvoření nového obrazu příkazem `build`:

```
$ podman build -t mujhttpd .
```

- `-t mujhttpd` – nově vzniklý obraz se označí jménem `mujhttpd`,
- tečka `.` – adresář, ve kterém je `Dockerfile` (tzv. build context),
- výsledný obraz `mujhttpd` je stejný jako `httpd`, jen má upravený `index.html` s textem `Ahoj`.

Samostatná práce I – základní práce s kontejnery

- **1. Ubuntu kontejner**
  - stáhni obraz `ubuntu`,
  - spusť interaktivní kontejner s `--rm`,
  - kontejner ukonči (`exit`) a ověř pomocí `podman ps -a`, že kontejner po skončení zmizel.
- **2. Seznam kontejnerů a obrazů**
  - spusť jeden kontejner `ubuntu` na pozadí (-d),
  - pomocí `podman ps` najdi jeho ID nebo jméno,
  - kontejner zastav a smaž,
  - pomocí `podman images` se podívej, jaké obrazy máš lokálně.

Samostatná práce II – webový server a vlastní obraz

- **3. Webový server `httpd`**
  - spusť kontejner s `httpd`, který zpřístupní port 80 kontejneru na portu 8080 hostitele,
  - v prohlížeči nebo přes `curl` ověř, že se zobrazuje stránka „It works!“.
- **4. Vlastní obsah přes připojený adresář**
  - v adresáři `~/container_htdocs` vytvoř soubor `index.html` s vlastním textem,
  - spusť `httpd` tak, aby tento adresář byl připojený jako `/usr/local/apache2/htdocs/` (použij `-v` a `:Z`),
  - ověř v prohlížeči, že se zobrazuje právě tvůj `index.html`.
- **5. Vlastní obraz z Dockerfile**
  - v novém adresáři vytvoř `Dockerfile`, který vychází z `httpd` a přepíše `index.html` vlastním textem,
  - obraz postav pomocí `podman build` a dej mu jméno (např. `mujhttpd`),
  - spusť kontejner z nového obrazu a ověř, že se v prohlížeči zobrazuje text z tvého `Dockerfile`.