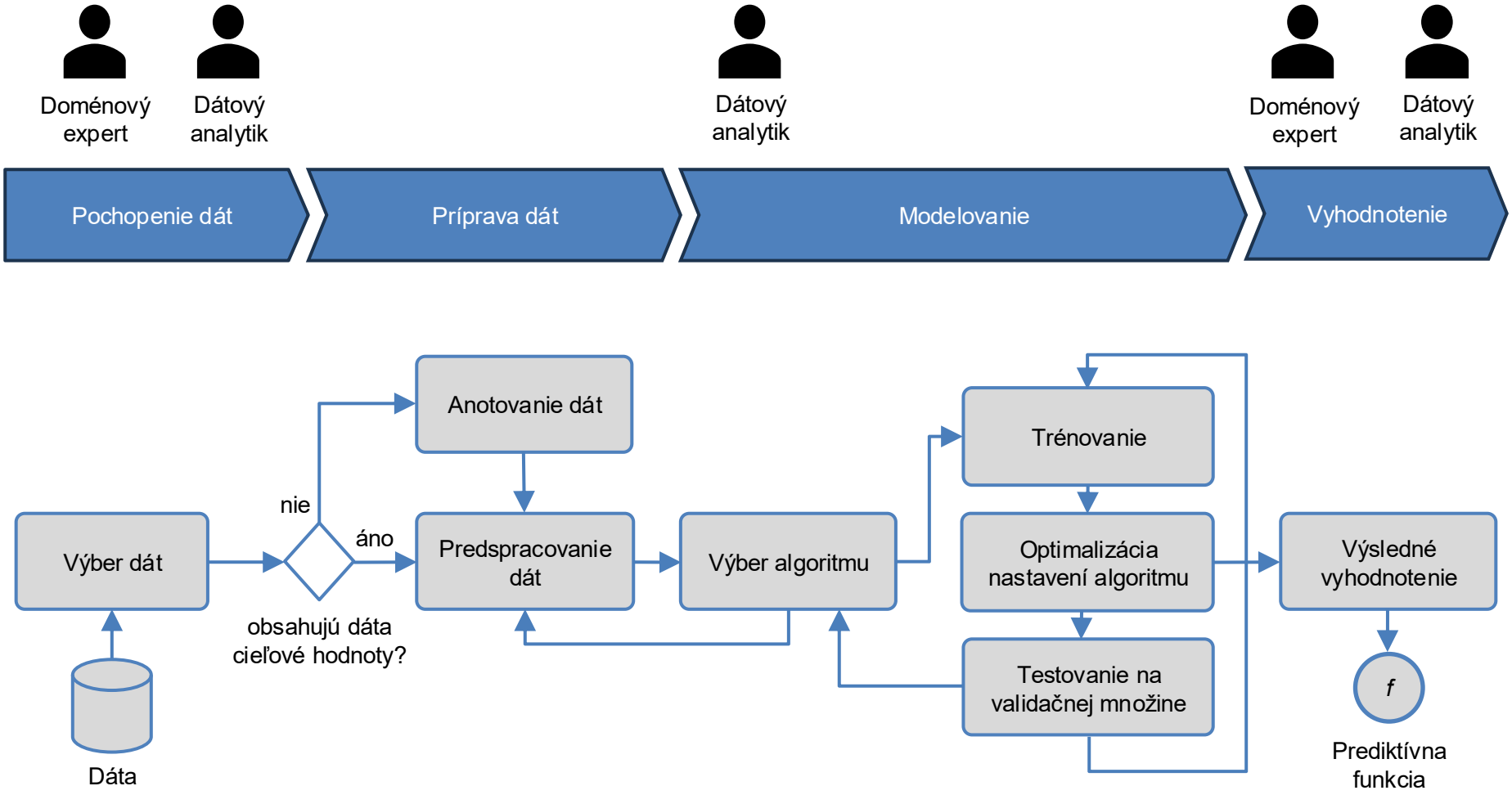
Integrácia a nasadenie softvérových aplikácií 2

Peter Bednár

Vývojové prostredie

Vývojové prostredie (1)



Vývojové prostredie (2)

Integrované prostredie = dáta + kód + modely + infraštruktúra

- Riadiaca platforma celého životného cyklu strojového učenia, ktorá umožňuje spoľahlivé experimentovanie, reprodukovateľnosť, spoluprácu a nasadenie

Vývojové prostredie - požiadavky (1)

- Produktivita a experimentovanie
- Vývojové prostredie poskytuje nástroje, ktoré umožňujú rýchlu iteráciu nápadov:
  - Interaktívne programovanie (napr. notebooky, IDE),
  - Okamžitú spätnú väzbu o správaní modelu,
  - Rýchle prototypovanie
- Skrátenie cyklu experiment–vyhodnotenie–zlepšenie, čo je v ML zásadné, keďže rozhodnutia sú založené na empirických výsledkoch

Vývojové prostredie - požiadavky (2)

- Spracovanie a prieskum dát
- Systémy strojového učenia sú orientované na dáta. Prostredie musí podporovať:
  - Získavanie dát z rôznych zdrojov
  - Čistenie, transformáciu a tvorbu atribútov
  - Exploračnú analýzu dát (EDA) a vizualizáciu

Vývojové prostredie - požiadavky (3)

- Vývoj a tréning modelov
- Vývojové prostredie umožňuje:
  - Implementáciu algoritmov pomocou ML knižníc a rámcov
  - Využitie hardvérovej akcelerácie (GPU/TPU)
  - Konfiguráciu nastavení algoritmov (hyper-parametrov) a tréningových pipeline-ov
- Abstrahuje nízkoúrovňovú zložitosť, aby sa vývojári mohli sústrediť na logiku modelu namiesto infraštruktúry

Vývojové prostredie - požiadavky (4)

- Správa závislostí a prostredí
- Vývojové prostredie musí zabezpečiť čo najväčšiu reprodukovateľnosť:
  - Správa verzií kódu
  - Konzistentné verzie knižníc
  - Izolované prostredia (napr. virtuálne prostredia, kontajnery)

Vývojové prostredie - požiadavky (5)

- Sledovanie experimentov a reprodukovateľnosť
  - Okrem sledovanie verzií kódu je potrebné sledovať verzie dát a modelov
  - Sledovanie experimentov (metriky, nastavenia algoritmov, artefakty, ako sú napr. dáta v jednotlivých fázach spracovania, modely alebo vizualizácie a reporty z vyhodnotenia)

Vývojové prostredie - požiadavky (6)

- Testovanie a validácia
- Okrem metrík presnosti prostredie podporuje:
  - Jednotkové a integračné testy dátových pipeline-ov
  - Validáciu predpokladov modelu
  - Kontroly zaujatosti, robustnosti a meranie škálovateľnosti

Vývojové prostredie - požiadavky (7)

- Spolupráca a škálovanie tímu
- Pri tímovom vývoji ML prostredie umožňuje:
  - Zdieľané kódy a dátové množiny
  - Štandardizované pracovné postupy
  - Zdieľanie znalostí medzi doménovými expertmi a dátovými analytikmi

Vývojové prostredie - požiadavky (8)

- Nasadenie a integrácia MLOps
- Kontinuálne prepojenie s produkčným prostredím:
  - Príprava inštalačného balíka pre inferenciu
  - CI/CD pipeline pre ML
  - Monitorovanie výkonu po nasadení
  - Integrovanie spätnej väzby od používateľov

Správa závislostí a prostredí

Sledovanie verzií kódu

- Systém na sledovanie zmien v zdrojovom kóde v čase
  - Nie len zdrojový kód ale aj ďalšie textové artefakty (konfiguračné súbory, dokumentácia, inštalačné skripty)
- Uchováva kompletnú históriu úprav
- Umožňuje návrat k predchádzajúcim verziám
- Podporuje tímovú spoluprácu a umožňuje paralelný vývoj
- Zlepšuje kvalitu kódu

Úvod do Git

- Distribuovaný systém správy verzií (DVCS)
- Navrhnutý pre rýchlosť, efektivitu a škálovateľnosť
- Každý vývojár má kompletnú lokálnu kópiu repozitára
- Široko používaný v modernom vývoji softvéru

Základné pojmy v Git

- Repozitár – úložisko projektu s históriou
- Commit – snímka zmien v kóde
- Branch (vetva) – samostatná línia vývoja
- Merge – zlúčenie zmien z vetiev
- Tag – pomenované označenie snímky kódu (napr. číslom verzie)

Základný pracovný tok Git

1. clone - vytvorenie lokálnej kópie, alebo pull načítanie zmien z repozitára
2. zmena kódu
3. commit zmien (zaznamenaných lokálne)
4. push odoslanie zmien do repozitára

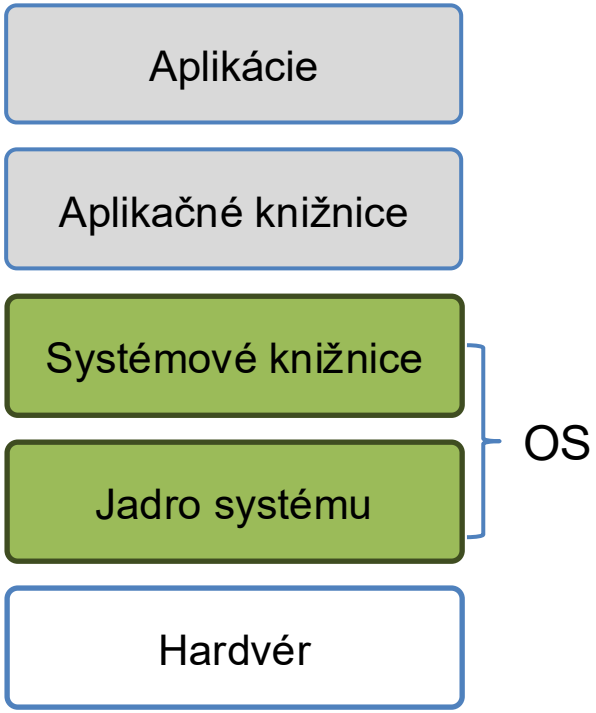
- Je možné vytvoriť samostatnú vetvu vývoja (checkout) pre novú funkčnosť
  - Hlavná vetva zostáva stabilná, vo vývojovej vetve je možné experimentovať
- Konflikty sa riešia pri zlúčení (merge)

GitHub

- GitHub je cloudová platforma na hosťovanie Git repozitárov
- Umožňuje spoluprácu vývojárov na diaľku
- Poskytuje nástroje ako:
  - Pull Requests
  - Code Review
  - Issues (sledovanie úloh a chýb)
  - CI/CD pipeline (automatizované testovanie a nasadenie)
- Často sa používa na open-source aj komerčné projekty

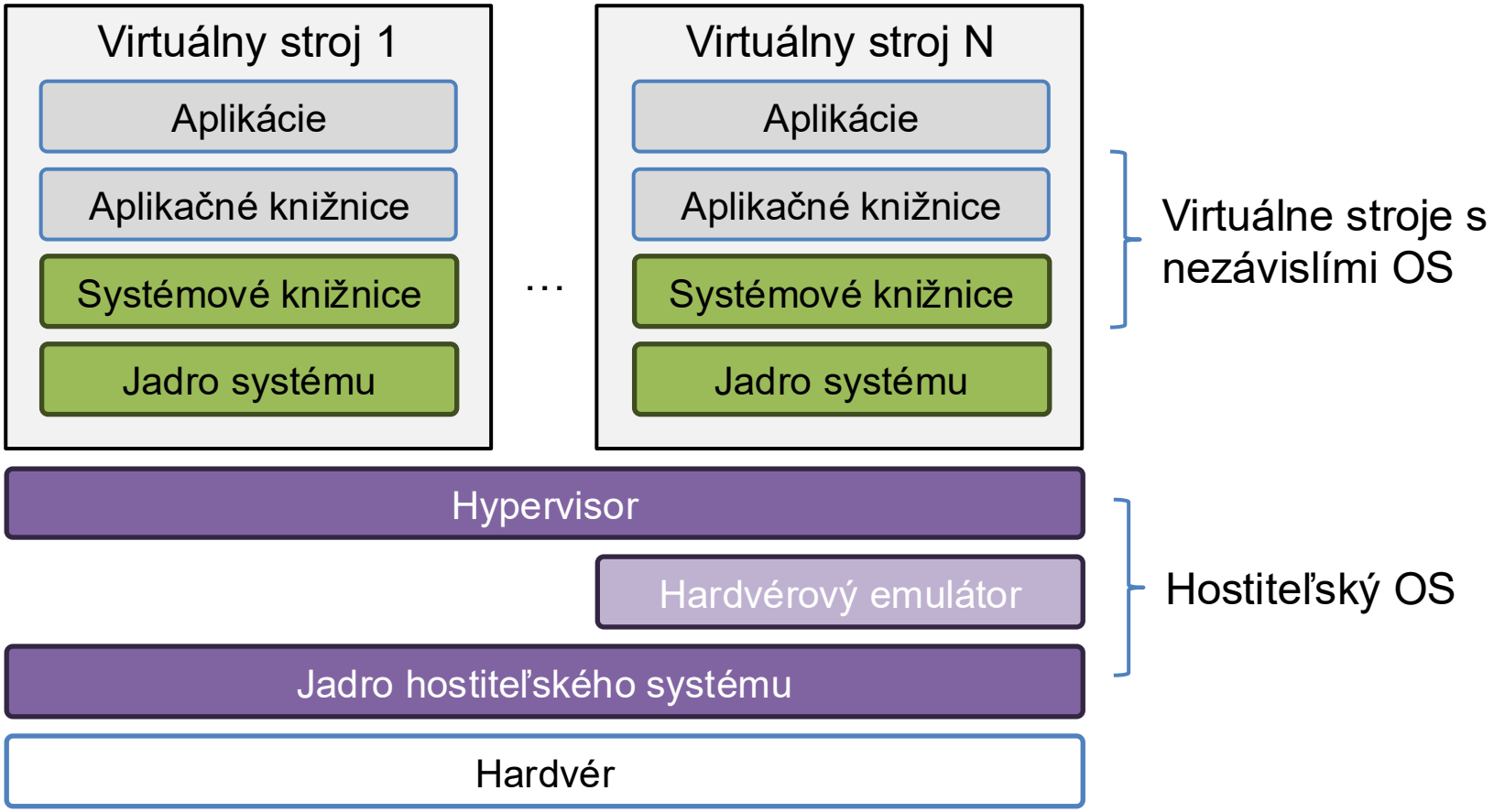
Správa prostredia

Programové prostredie

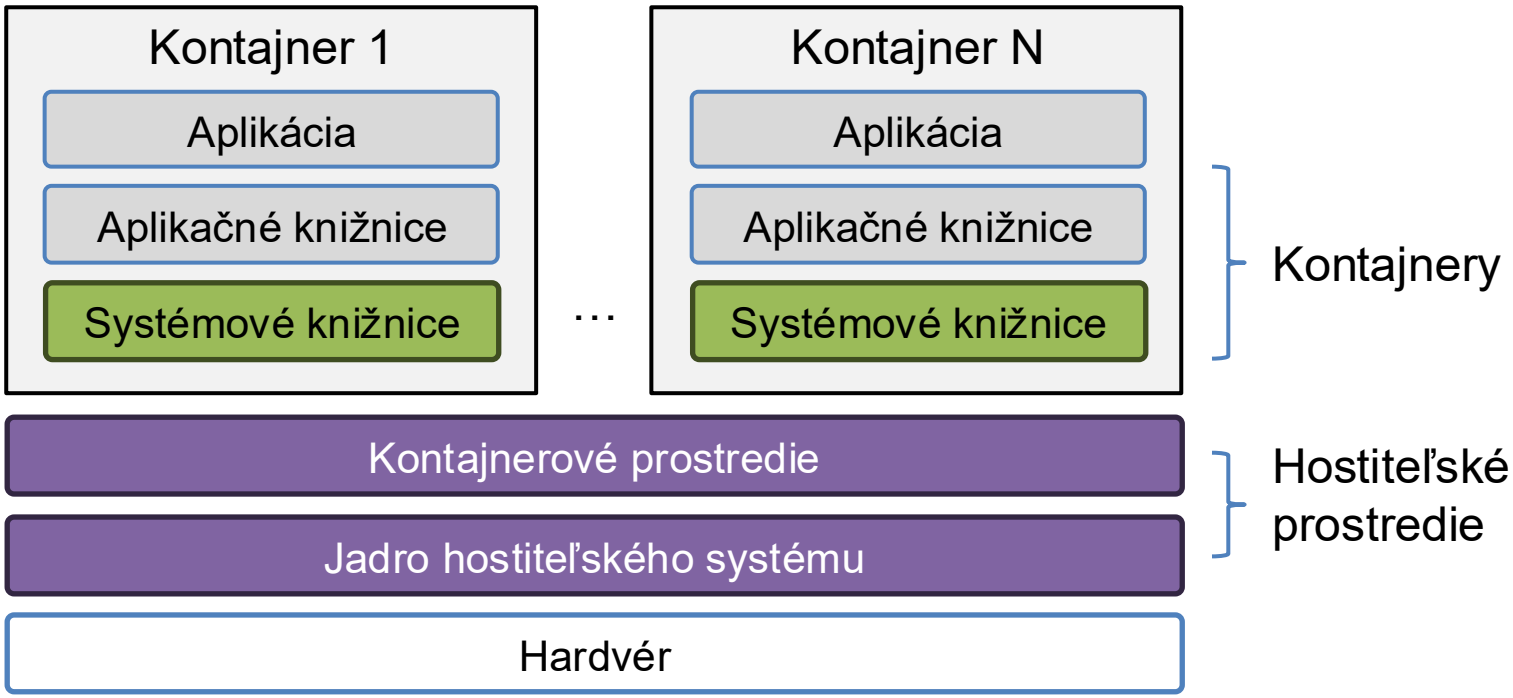


- Základné požiadavky na programové prostredie
  - Efektívne využívanie výpočtových zdrojov
  - Škálovateľnosť
  - Flexibilnosť pri nasadení
  - Jednoduchá údržba
  - Prenositeľnosť
  - Bezpečnosť

Virtualizácia



Kontajnerizácia



Virtualizácia vs Kontajnerizácia

Virtualizácia

- Silná izolácia záťaže a vyššia miera bezpečnosti
- Podpora rôznych operačných systémov na jednom hostiteľovi

Kontajnerizácia

- Efektívnejšia réžia
- Rýchlejší štart/reštart aplikácií
- Vyššia hustota izolovaných aplikácií na jednom hostiteľskom serveri
- Optimalizované prostredie pre (mikro)služby

Docker

- Kontajnerové prostredie, ktoré umožňuje vytváranie a spúšťanie kontajnerov

- Základné pojmy
  - Image (obraz) – uložená kópia konfigurovaného prostredia, ktorá slúži ako šablóna pri vytvorení kontajnera
  - Dockerfile – konfiguračný súbor, ktorý definuje ako má byť image zostavený
  - Docker image repozitár – úložisko ktoré umožňuje zdielanie image-ov

Konfigurácia kontajnerov – Dockerfile (1)

- Textový konfiguračný súbor na základe ktorého Docker vytvorí image

- Základné príkazy:
  - FROM – základný image (znovapoužiteľnosť/rozšíriteľnosť)
  - RUN – spustenie programu pre konfirguráciu prostredia počas vytvárania image-u
  - COPY – pridávanie súborov do image-u
  - CMD/ENTRYPOINT – definuje, ktorý program sa má spustiť pri spustení kontajnera

Konfigurácia kontajnerov – Dockerfile (2)

```
FROM python:3.12-slim

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY app.py .

CMD ["python", "app.py"]
```

docker build -t peterbednar/hello-world:0.1.0 .

docker container run peterbednar/hello-world:0.1.0

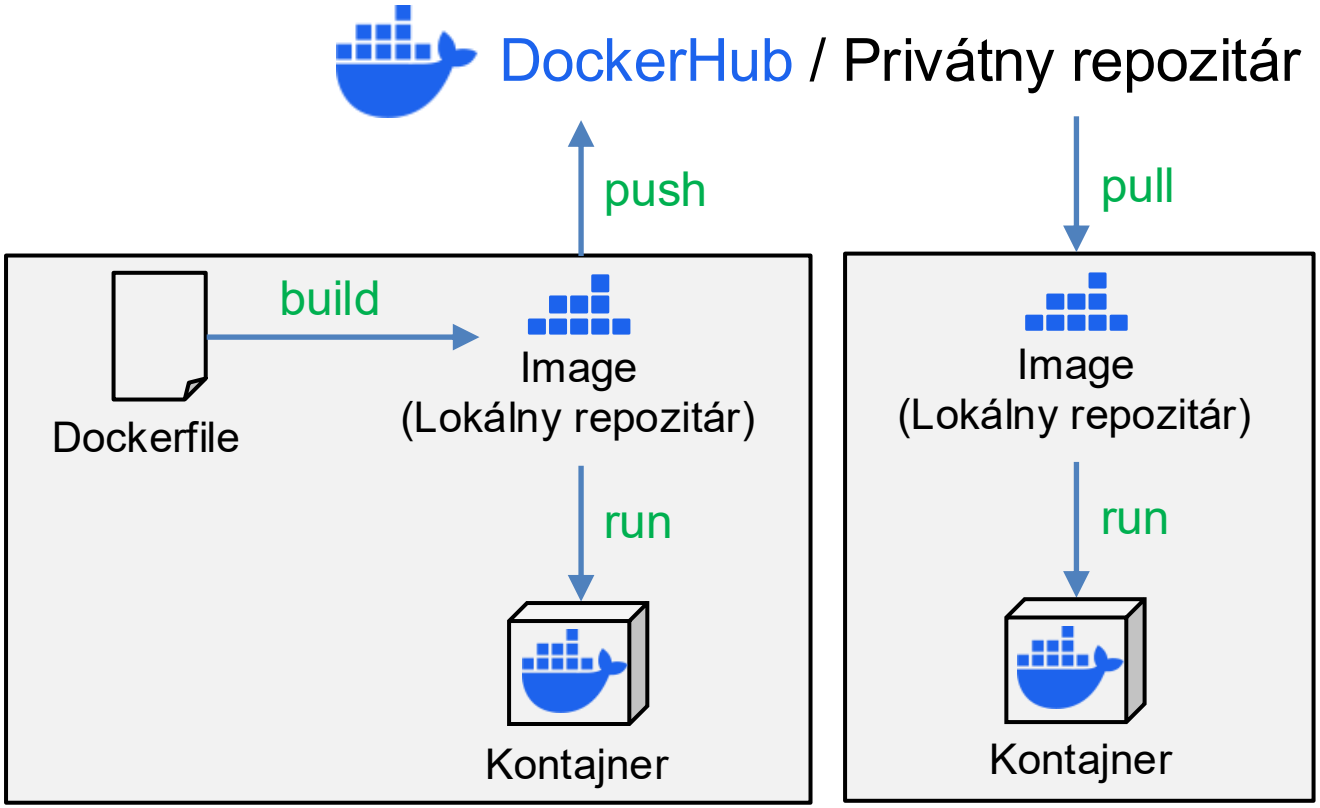
DockerHub

- Oficiálna klaudová služba Docker
- Prednastavený repozitár pre zdieľanie image
- Oficiálne verzie pre viacero štandardných aplikácií

```
docker image push peterbednar/hello-world:0.1.0

docker image pull peterbednar/hello-world:0.1.0
```

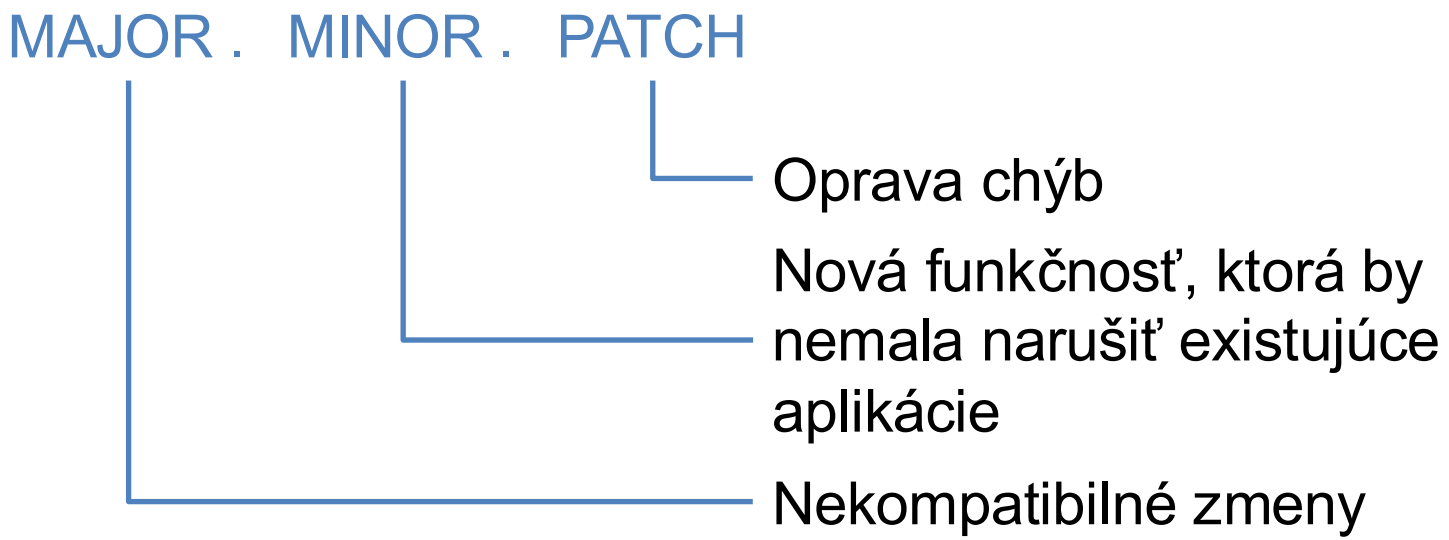
Docker prostredie – zhrnutie



Správa verzii

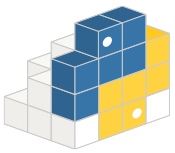
Sémantické označovanie verzií

- Štandardizovaná schéma pre označovanie verzií
  - Napr. numpy 2.4.2



Manažovanie Python prostredia – pip

- pip
  - Základný program pre inštalovanie balíkov (knižnice/aplikácie) v Python prostredí
  - Balíky sú stiahnuté z repozitárov
    - PiPy – oficiálny repozitár
    - Inštalácia priamo z git alebo z privátnych repozitárov
  - Manažuje iba balíky pre Python (nie samotnú inštaláciu Python interpretra)



Manažovanie Python prostredia – conda

- conda
  - Cross-platformový manažér prostredia
  - Manažuje nie len Python balíky ale aj Python verzie a binárne závislosti (ostatné aplikačné knižnice napr. v C/C++)
  - Využíva vlastnú sieť repozitárov (napr. conda-forge), ale umožňuje aj inštaláciu cez pip
  - Populárny pre dátovú analytiku a ML

Manažment balíkov – pip

```
pip install numpy
pip install "numpy==2.4.1"
pip install "nympy==2.4.*"
pip install "numpy>=2.3,<2.4.1,!=2.3.8

pip install –r requirements.txt
```

```
numpy
pandas==3.0.1
scikit-learn 1.8.0
...
```

```
pip list
pip uninstall numpy
```

Manažment balíkov – conda

```
conda install numpy
conda install "numpy=2.4.1"
conda install "nympy=2.4.*"
conda install "numpy>=2.3,<2.4.1,!=2.3.8

conda list
conda remove numpy
```

Virtuálne Python prostredia (1)

- Umožňujú lokálne vivíjať viacero projektov s rôznymi závislosťami a replikovať Python prostredie

- pip

```
python -m venv myenv

source myenv/bin/activate          myenv\Scripts\activate

pip install numpy pandas
python app.py
```

Virtuálne Python prostredia (2)

- conda

```
conda create -n myenv python=3.11

conda activate myenv

conda install numpy pandas

python app.py
```

Virtuálne Python prostredia (3)

- conda

conda create -f environment.yml

```
name: myenv
channels:
  - conda-forge
dependencies:
  - python=3.11
  - numpy
  - pandas
  ...
```

Stabilné softvérové prostredie (1)

app.py
- pandas
- scikit-learn=1.7.2

app.py
pandas 2.3.1
scikit-learn 1.7.2
numpy 1.3.8

pandas 2.3.1
- numpy>=1.2.6

. . .

scikit-learn 1.7.2
- numpy>=1.3
- pandas>=2.0

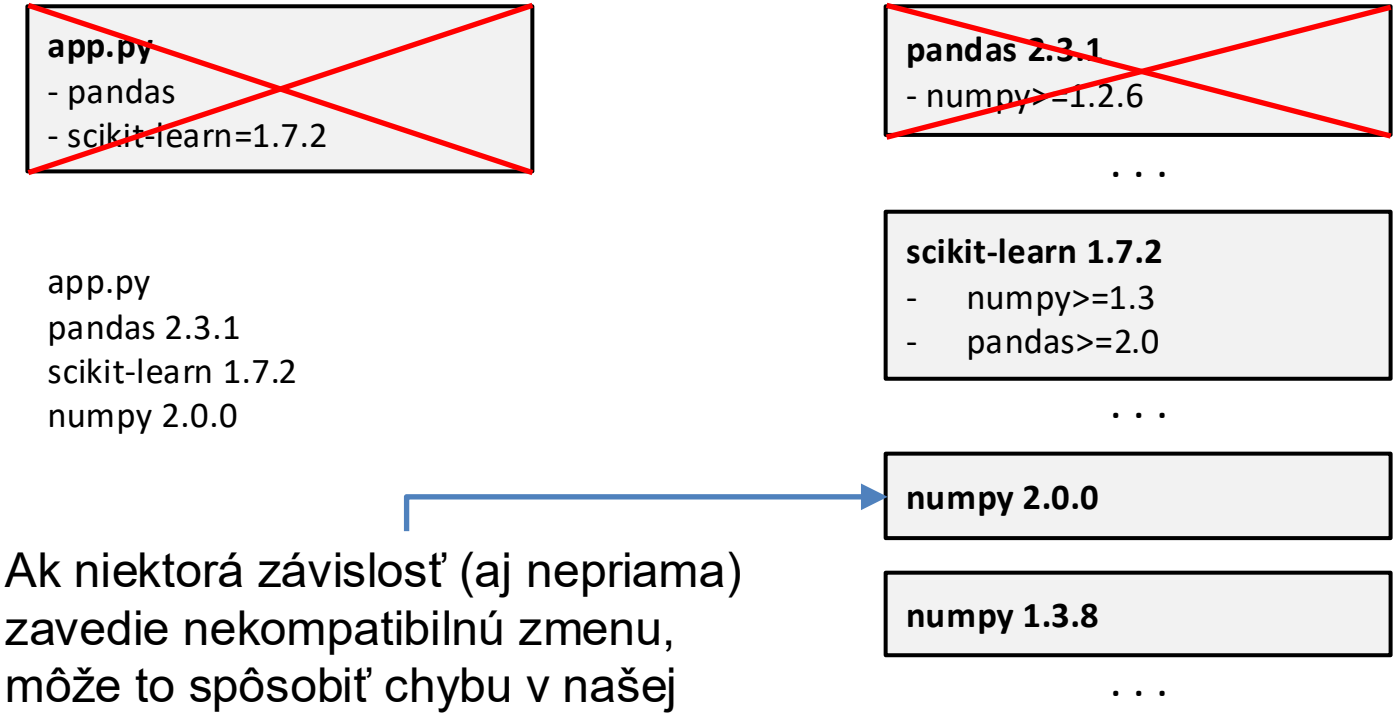
. . .

numpy 1.3.8

numpy 1.3.7

. . .

Stabilné softvérové prostredie (2)



Ak niektorá závislosť (aj nepriama) zavedie nekompatibilnú zmenu, môže to spôsobiť chybu v našej aplikácii

Stabilné softvérové prostredie (3)

app.py
- pandas=2.3.1
- scikit-learn=1.7.2
- numpy=1.3.8

app.py
pandas 2.3.1
scikit-learn 1.7.2
numpy 1.3.8

Bezpečnejšie je špecifikovať verzie všetkých závislostí (aj keď ich priamo nevyužívame v našom kóde)

pandas 2.3.1
- numpy>=1.2.6

. . .

scikit-learn 1.7.2
- numpy>=1.3
- pandas>=2.0

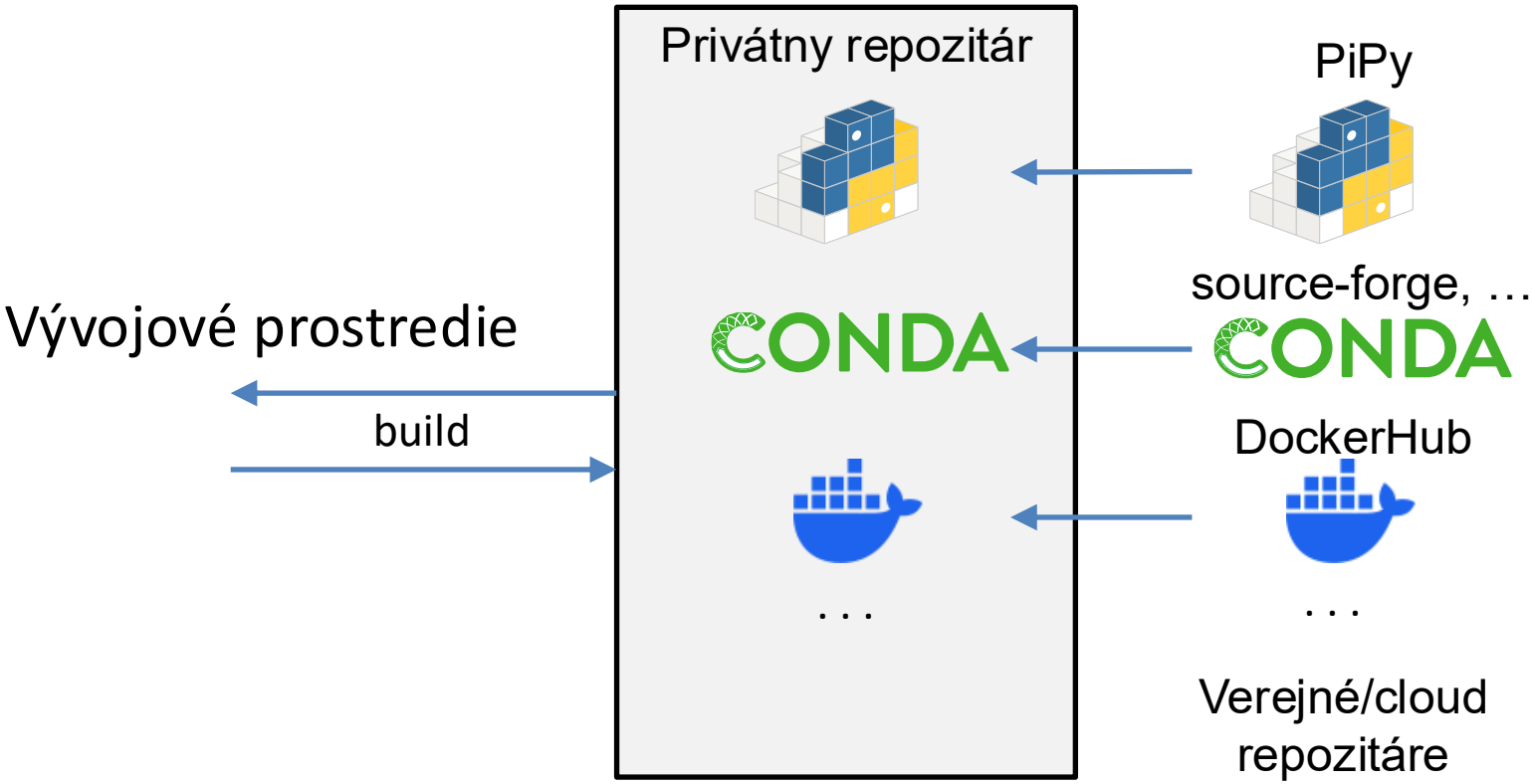
. . .

numpy 2.0.0

numpy 1.3.8

. . .

Privátne repozitáre



Sonatatype Nexus, AWS CodeArtifact, GitHub packages, …