IVS - Praktické aspekty vývoje software

Kontejnerizace

Ing. Martin Sakin

2026

Agenda

01 Co je to kontejner?

02 Jak funguje izolace?

03 Docker vs. Podman

04 Praktické ukázky

05 Orchestrace a velký svět

Definice problému

1. Vývojář vytvoří / zprovozní / vyladí aplikaci
2. Vše funguje, testy jsou zelené
3. Nasazení na produkčním serveru kompletně selhává

Definice problému

Důsledky:

- Ztrácení hodin debugováním prostředí místo kódu
- „Dependency hell" – konfliktní závislosti mezi projekty
- Nereprodukovatelné bugy (u vývojáře nenastane, na serveru ano)
- Obtížné škálování (jak zajistíte stejné prostředí na 100 serverech?)

Tradiční řešení:

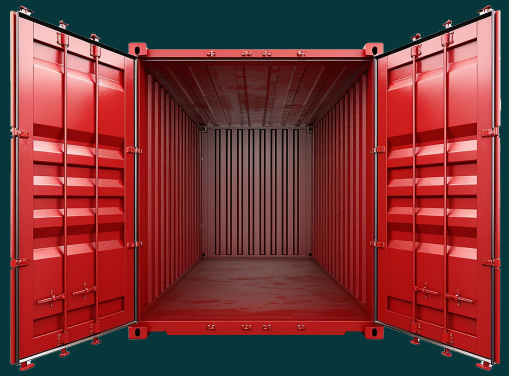
- Dokumentace prostředí → rychle zastarává, lidská chyba
- Shell skripty → křehké, platformě závislé
- Virtuální stroje → těžké, pomalé, velké obrazy (GBs)

Kontejnerizace

Metoda virtualizace na úrovni operačního systému, která umožňuje spustit aplikaci a její závislosti v izolovaném procesu.

Kontejner

Zabalení **aplikace** spolu se všemi jejími **závislostmi** (knihovny, konfigurace)



K čemu je užitečný?

Přenositelný	běží stejně na laptopu, serveru i v cloudu
Izolovaný	vlastní souborový systém, síť, procesy
Lehký	sdílí kernel OS (rychlý start)
Verzovatelný	image jsou neměnné

Řeší problém “u mě to funguje”

VM *vs* container

Kontejnerizace je metoda virtualizace na úrovni operačního systému.

Virtual Machine

- Digitální replika fyzického stroje
- Vlastní Guest OS
- Start v minutách (0.5 - 5m)
- Velikost v GB (2-20GB)
- Izolace přes HW (hypervisor)

X

Container

- Balíček softwarového kódu
- Sdílí Host Kernel
- Start v milisekundách (0.5-3s)
- Velikost v MB (50-300 MB)
- Izolace přes procesy (kernel features)

VM *vs* container

Kontejnerizace je metoda virtualizace na úrovni operačního systému.

Virtual Machine X **Container**

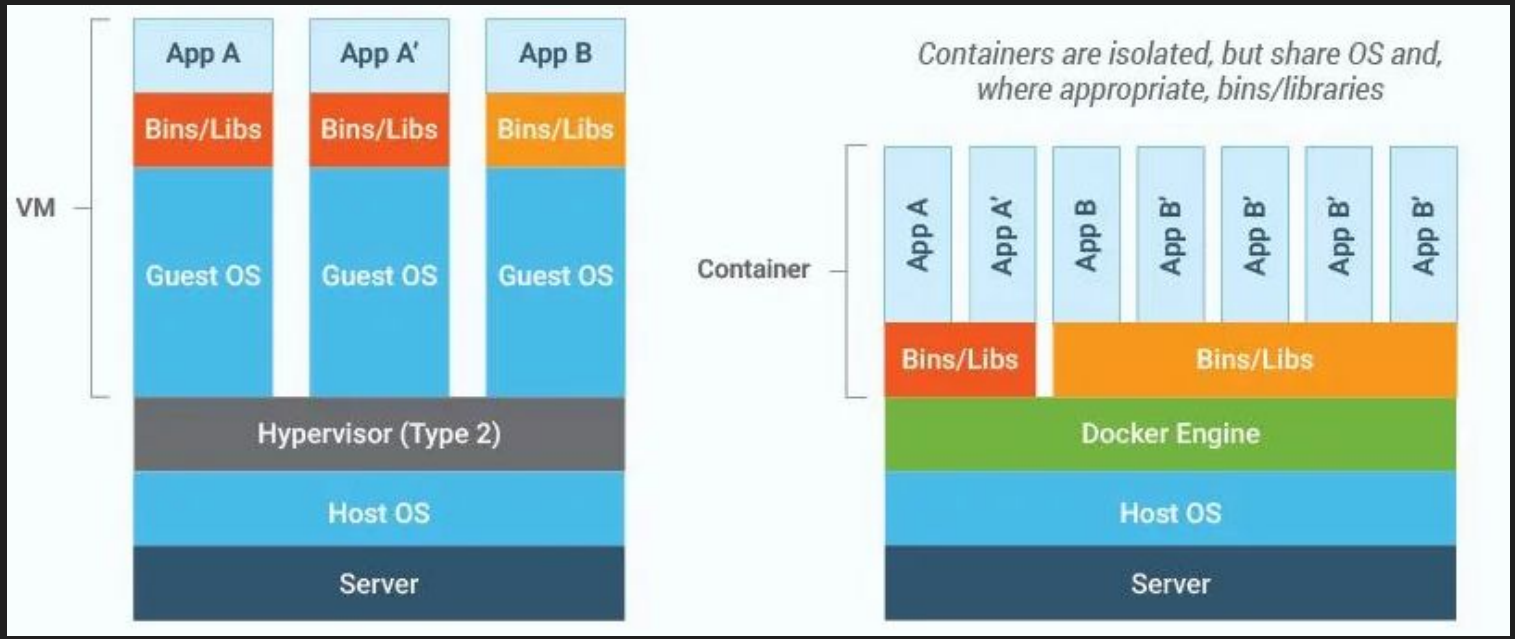


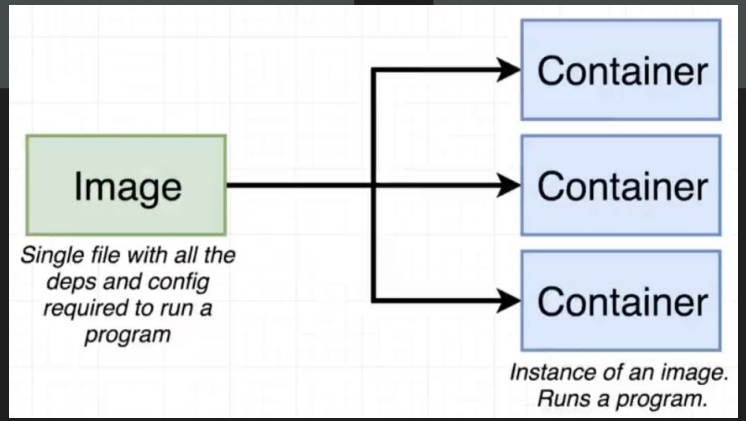
Image & container

Image

- “snapshot of an environment”
- Fyzicky uložené na disku
- Jednotka, kterou můžeme přesouvat
- Šablona

Container

- Instance / otisk
- Běžící procesy



Jak to funguje?

Kontejner je v podstatě izolovaný proces.

Využívá funkce linuxového jádra:

Namespaces

Vytváří pohled na systém. Proces v kontejneru si myslí, že má vlastní síť, uživatele, procesy a souborový systém.

Control Groups

Limitují zdroje. Definují kolik CPU, RAM nebo I/O operací může daný kontejner spotřebovat. (+ prioritizace)

Filesystem layers

Umožňují efektibní ukládání dat - jen rozdíly

Docker *vs* Podman

docker

podman

Hlavní rozdíly

Docker

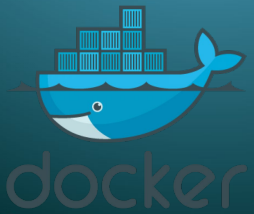
- Daemon s root právy (dockerd)
- Client-server
- Kontejner
- Linux, Windows, macOS
- Monolithic
- Docker Desktop je placený

Podman

- Daemonless (jako std proces)
- Rootless (jako běžný uživatel)
- Kontejner i Pod
- Primárně Linux
- Modular
- Open Source

- Stejné příkazy
- Stejný image formát (OCI image spec)

Docker



- Vytvořen firmou Docker (2013)
- Architektura klient-server
  - Docker Daemon (dockerd)
    - Centrální background service
    - Spravuje images, container, síť, volumes
    - Naslouchá na Unix socketu
    - Problém: Běží jako root uživatel
  - Docker CLI
    - Klientský nástroj
    - Komunikuje s daemon přes REST API
- Docker Compose - multi-container development

Podman = POD MANager

- Vytvořen RedHatem jako bezpečná alternativa k Dockeru
- Každý kontejner je přímý potomek uživatelského procesu
- Fork-exec model
- Je možné spouštět kontejnery bez root práv
- UID uvnitř kontejneru → jiný UID na hostiteli
- Bezpečnostní benefit: i kdyby byl kontejner kompromitován, úročník má práva jen normálního uživatele
- Kontejnery obvykle běží jako systemd services

Další společné info

podman

docker

- **Repository**
  - cloudové úložiště images
  - Odtud je stahuji pomocí příkazu **`pull`**
  - docker.io nebo quay.io
  - Firmy si z bezpečnostních důvodů mohou vytvořit vlastní
- **Dockerfile** / **Containerfile**
  - Popis pro vytvoření vlastního image
- Docker/Podman **compose**
  - skupinová správa kontejnerů
  - YAML definition file
  - jedna řídící jednotka

Společné příkazy

podman

docker

```
$ podman pull docker.io/…    # stažení image
$ podman run      # vytvoření a spuštění kontejneru z image
$ podman exec     # spuštění příkazu v kontejneru
$ podman logs -f # logy z kontejneru
$ podman start    # spuštění
$ podman stop     # zastavení
$ podman ps -a    # přehled všech kontejnerů
$ podman images   # přehled všech image
$ podman rm <container_name>  # smazání kontejneru
$ podman rmi <image_name>     # smazání image
```

Demo

Instalace nástoje

$ sudo apt install **podman**

Stažení image

$ podman pull docker.io/library/nginx

Vytvoření/spuštění containeru

$ podman run --name mujserver -p 8000:80 nginx

http://localhost:8000

Vytvětlivky:
--name vlastni_pojmenování_kontejneru
-p vnější_port : vnitřní_port
-d (odpojeni od terminalu)
-rm (smaž po ukončení)
nginx (název zdrojového image)

Vlastní image - Dockerfile

```
FROM docker.io/library/python:3.14
WORKDIR /work/
COPY myapp/binary /work/application
RUN chmod 775 /work
CMD [“./myapp”]
```

```
$ podman build -t <image_name> .
```

Volumes

- Kontejnery jsou dočasné
- Pokud je smažeme, data zmizí
- Volumes umožňují data uložit mimo kontejner
- Persistní data

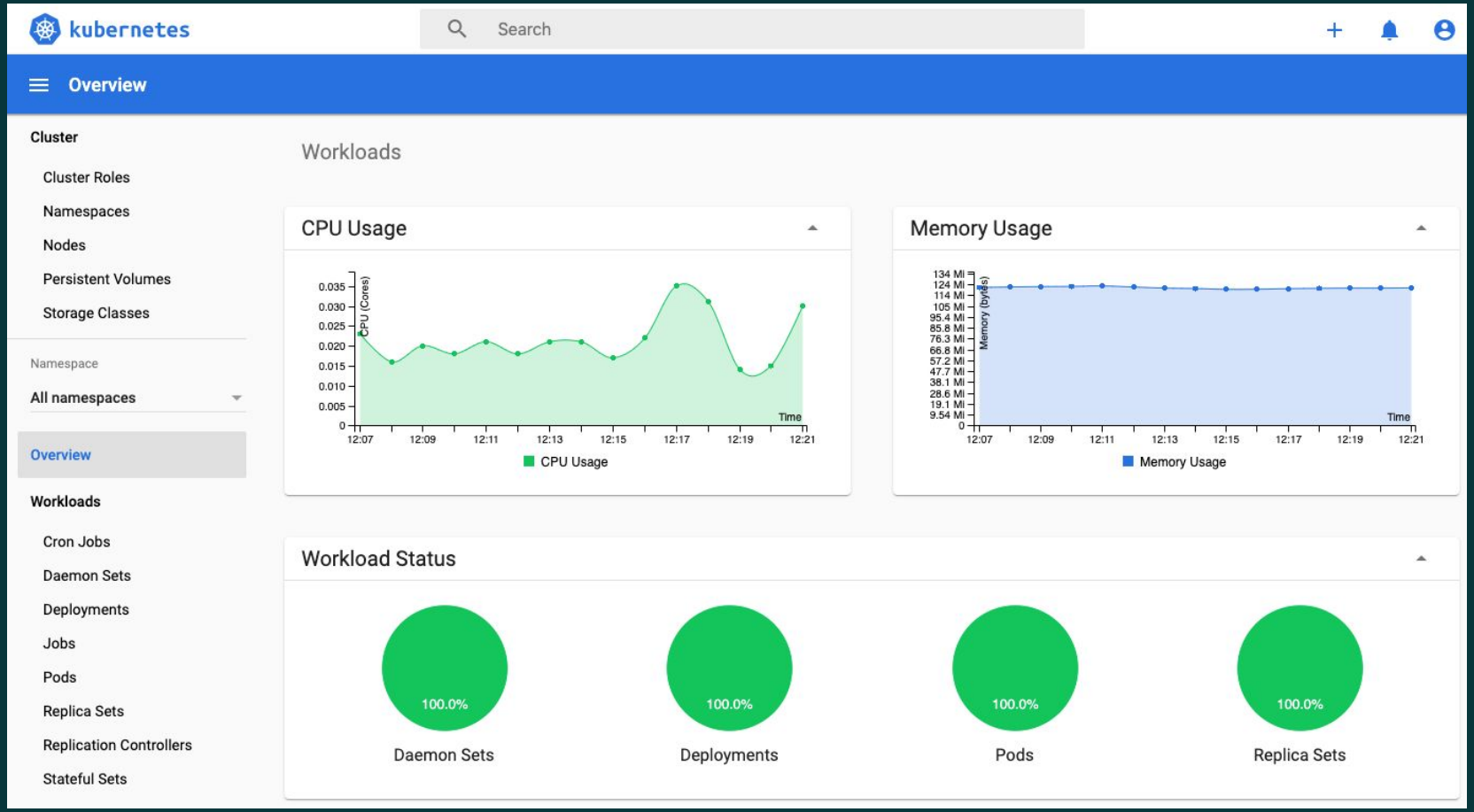
```
$ podman volume ls
```

Orchestrace

Kubernetes

- Slouží k orchestraci kontejnerů, zjednodušuje administraci
- Open-source systém pro automatizaci: nasazení, škálování a správy kontejnerizovaných aplikací
- *De-facto* standard pro cloudový vývoj
- Umožňuje škálování, restartování a správu aplikací
- Self-healing
- Základní jednotka je **pod**, který může obsahovat jeden nebo více kontejnerů

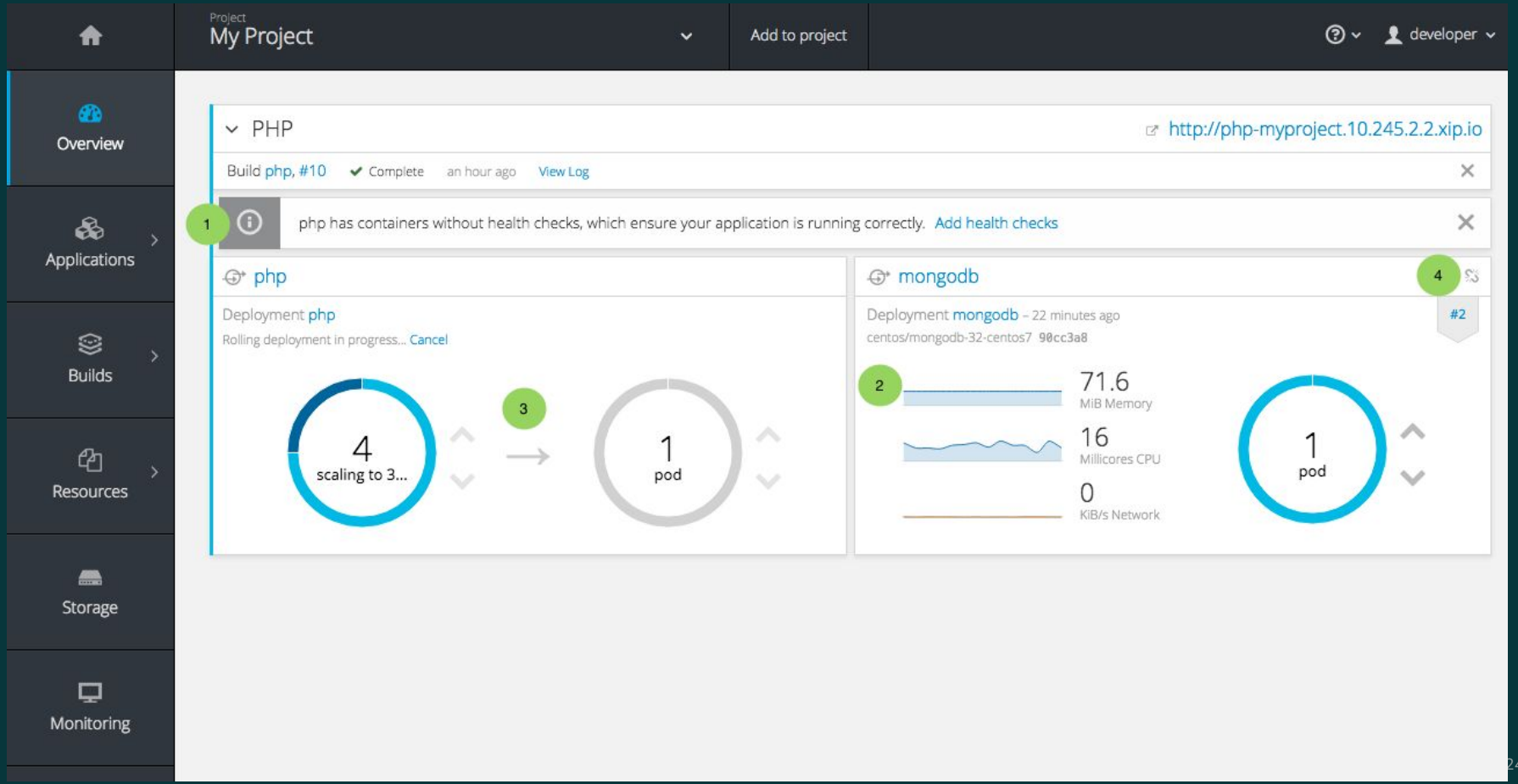
Kubernetes


kubernetes
Search
Overview
Cluster
Cluster Roles
Namespaces
Nodes
Persistent Volumes
Storage Classes
Namespace
All namespaces
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Workloads
CPU Usage
CPU (Cores)
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
12:07
12:09
12:11
12:13
12:15
12:17
12:19
12:21
Time
CPU Usage
Memory Usage
Memory (bytes)
134 Mi
124 Mi
114 Mi
105 Mi
95.4 Mi
85.8 Mi
76.3 Mi
66.8 Mi
57.2 Mi
47.7 Mi
38.1 Mi
28.6 Mi
19.1 Mi
9.54 Mi
0
Memory Usage
Workload Status
100.0%
Daemon Sets
100.0%
Deployments
100.0%
Pods
100.0%
Replica Sets

OpenShift

- Fork RedHatu
- Enterprise Kubernetes
- CI/CD
- GUI
- Spouštění podů on-demand
- Free 30 dní: https://developers.redhat.com/developer-sandbox

OpenShift


Project
My Project
Add to project
developer
Overview
Applications
Builds
Resources
Storage
Monitoring
PHP
http://php-myproject.10.245.2.2.xip.io
Build php, #10
Complete
an hour ago
View Log
1
php has containers without health checks, which ensure your application is running correctly.
Add health checks
php
Deployment php
Rolling deployment in progress... Cancel
4
scaling to 3...
3
1
pod
mongodb
4
Deployment mongodb – 22 minutes ago
#2
centos/mongodb-32-centos7 90cc3a8
2
71.6
MiB Memory
16
Millicores CPU
0
KiB/s Network
1
pod

Děkuji za pozornost :)