Docker Kontejnery v Praxi

Ondřej Šika
Freelance & SikaLabs s.r.o.

ondrej@sika.io
@ondrejsika

Konference EurOpen, Šlovice, 16. 5. 2023

Ondřej Šika

Jsem DevOps lektor, architekt a konzultant z Prahy.

Navrhnu a implementuji Vám na míru DevOps architekturu od verzování v Gitu po provoz v Cloudu.

Dělám populární školení, kde své znalosti předávám tak, abyste si mohli vše udělat sami a bez zbytečných přešlapů a slepých cest.

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Můj Open Source DevOps Stack

- Git, Gitlab, Github - Versioning & Collaboration
- Gitlab CI - Continuous Integration, Continuous Deployment
- **Docker**, Kubernetes - Containers & Orchestration
- RKE2 & Rancher - Kubernetes Provisioning
- Terraform - Infrastructure management
- Gobble - Configuration management
- Prometheus, Grafana - Monitoring Stack
- Elastic Stack / Loki - Log Management
- DigitalOcean, AWS, VMWare, Proxmox - Public or Private Cloud
- Ceph - On Premise Storage
- Tergum - Backups, DR

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Docker. Proč?

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Proč jsou kontejnery potřeba?

- Izolace aplikací
- Jednoduchá distribuce aplikací
- Jednoduchý provoz aplikací
- Unifikace prostředí pro provoz aplikací
- Urychlení vývoje a testování

Jde to i bez Dockeru, ale...

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Před Dockerem

- 2 směry - Virtualizace a náznaky kontejnerizace
- Distribuce pomocí balíčku a VM Images (VMWare)
- Unifikace na úrovni VMs nebo Linuxu (... nic moc)

Kontejnery vs Virtulizace před Dockerem

- **Virtualizace**
  - Emulace HW
  - VMWare, Virtual Box
- **Kontejnerizace**
  - Izolace v rámci jednoho Kernelu
  - Chroot, BSD Jails, OpenVZ, XEN, LXC, ... Docker

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Co je to Kontejner

Kontejner je forma izolace aplikace a její závislost od okolního prostředí, na kterém běží (podobně jako virtualizace). Kontejnery fungují na úrovni operačního systému (Kernelu), což znamená, že několik kontejnerů může běžet na jednom fyzickém nebo virtuálním stroji.

Každý kontejner obsahuje vlastní souborový systém se všemi nástroji, knihovnami a soubory, které jsou potřebné pro běh aplikace, včetně souboru operačního systému. Díky tomu je možné spustit aplikaci na jakémkoliv serveru, bez nutnosti instalace jakychkoliv závislostí.

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Co je to Docker

Docker je kontejnerizační platforma.

Docker nám umožňuje velmi jednoduše pracovat s kontejnery. Umožňuje nám vytvářet a distribuovat images pro kontaineru a kontejnery spouštět.

Docker není jediná implementace moderního přístupu práce s kontejnery, je tu například Podman, Containerd, ... ale Docker byl první, kdo změnil přístup, jak se kontejnery využívají

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Změna k přístupu ke kontejnerům

- **Kontejnerizace == Lightweight Virtualizace**
  - Kontejnery se používají jako "lehké" VMs - kontejner se vytvořil, spustil a pomoci SSH jsme se do něj připojili a pracovali jako s VM
  - OpenVZ, XEN, LXC

- **Docker**
  - Docker změnil přístup k práci s kontejnery a **přinesl revoluci**
  - Docker kontejner se pouští z image, který se jednoduše sestavuje (build) a distribuuje

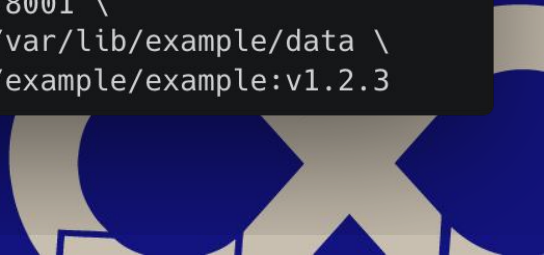
Klíčová změna přístupu

- **Docker Run**
  - Pro spuštění kontejneru potřebujeme konkrétní image
- **Docker Build**
  - Docker image se velmi jednoduše sestaví - potřebujeme jen Docker
- **Docker Push / Pull**
  - Docker umožňuje velmi jednoduše distribuovat images pro kontejnery

```
FROM golang:1.20 as build
WORKDIR /build
COPY go.mod go.sum ./
RUN go mod download
COPY . .
RUN go build

FROM debian:11-slim
COPY /build/example /usr/local/bin/example
RUN ["/usr/local/bin/example"]
VOLUME /var/lib/example/data
EXPOSE 8000
EXPOSE 8001
```

```
docker run --name example -d \
    -p 8000:8000 -p 8001:8001 \
    -v application-data:/var/lib/example/data \
    registry.company.com/example/example:v1.2.3
```



Výhody kontejneru

- Velmi jasná hranice mezi aplikací a platformou
- Distribuce univerzálních binárních obrazů
- Aplikace a data jsou striktně odděleny
- Jednoduchý a exaktně definovaný interface
- Izolace běžících aplikací
- Jednoduchý způsob běhu aplikací
- Auditovatelnost, předávací /schvalovací proces
- Checksums, Docker Content Trust
- Security scans, ...

Jasná hranice mezi aplikací a platformou

Hranice mezi kontejnerem a kontejnerizační platformou je

- Konfigurace pomocí ENV proměnných
- Kontejner image (obsahuje vše, co je pro běh aplikace potřeba)
- Exposed port (eg.: aplikace 8000, metriky 8001)
- Data volume mount (eg.: /var/lib/postgresql, ...)
- Logy na STDOUT, STDERR - napojení na centrální log management

Distribuce univerzálních binárních obrazů

- Kontejner obsahuje vše, co je potřeba pro běh aplikace
  - FS operačního systému
  - Systémové knihovny
  - Runtime environment (Java JRE)
  - Aplikační závislosti
  - Build aplikace nebo zdrojové kódy
- Naopak neobsahuje
  - Konfiguraci
  - Data
- Image je binárně kompatibilní pro různá prostředí i runtime
  - Docker, Podman, Containerd
  - Kubernetes, OpenShift

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

"Platform independent"

Aplikace, které běží v kontejnerech, nejsou závislé na runtime

- Docker
- Containerd
- Podman, CRI-O, ...

Různé orchestrátory

- Docker Compose
- Kubernetes
- OpenShift / OKD

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Container Registry

- Container registry jsou preferovaný způsob distribuce kontejnerů
  - docker pull, docker push, ...
- Container registry je standard, který používá
  - Gitlab, Github
  - Harbor
  - Artifactory
  - Nexus

- Odevzdání práce je push container images do registrů zákazníka
- Zákazník může mirrorovat registry od dodavatele

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Docker Compose

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

```yaml
version: "3.9"
name: example
services:
    example:
        build: .
        image: registry.company.com/example:v1.2.3
        ports:
            - 8000:8000
            - 8001:8001
        volumes:
            - example-data:/var/lib/example/data
volumes:
    example-data:
```

Co je to Docker Compose

Docker Compose je způsob,, jak jednoduše definovat vše, co je potřebné k buildu, distribuci a spuštění aplikace v jednom YAML souboru.

Docker Compose nám také umožní velmi jednoduše spouštět aplikace, které mají více různých komponent (db, app, ..)

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Proč použít Docker Compose

- Velmi jednoduché používání
- Nativní nástroj od Dockeru
- Ideální pro lokální spouštění a vývoj

Kubernetes

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example
spec:
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
        - name: main
          image: sikalabs/hello-world-server
          ports:
            - containerPort: 80
...
```

```
helm upgrade --install \
    example ./path/to/helm/package.tgz \
    --namespace example \
    --create-namespace \
    --values example.values.yml \
    --wait
```

Co je to Kubernetes

Kubernetes je platforma pro provoz aplikací.

Kubernetes nám umožňuje efektivně spravovat velké množství aplikací na více serverech z jednoho mista a unifikovaným způsobem..

Práce s Kubernetes je o trochu složitější, než se samotným Dockerem, případně Docker Compose, ale poskytuje nám mnoho výhod, jako je vysoká dostupnost, minimalizace manuálních činností, škálování, balíčky, ...

Kubernetes je často next-step pro prostředí, kde docker na jednom serveru přestává stačit...

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Proč používat Kubernetes?

- Unifikace prostředí pro provoz aplikací
- De-facto standard provozu software
- Ovládání pomocí YAML souboru + Helm Balíčky (GitOps)
- Deployment (deklarativní) požadovaného stavu
- Přístup ke clusteru místo k jednotlivým serverům
- Automatizace manuálních tasků
- Autoscaling
- Opensource, pod CNCF
- Velký ekosystém kolem Kubernetes
- Jednoduchý provoz v cloudu i on-premise

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Demo time 🎉🎉

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Závěr

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Docker nám přináší mnoho výhod

- Izoluje aplikace
- Unifikuje prostředí pro provoz aplikací
- Definuje interface mezi aplikací a platformou
- Usnadňuje vývoj, distribuci a běh aplikací

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Díky za pozornost

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Otázky?

@ondrejsika ondrej@sika.io sika.io /in/ondrejsika

Email

ondrej@sika.io

Twitter

@ondrejsika

LinkedIn

/in/ondrejsika