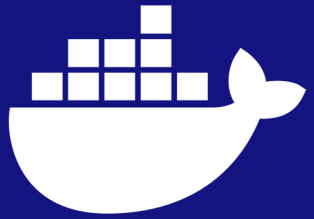
Výhody kontejnerů při distribuci softwaru i v enterprise prostředí



Ondřej Šika

ondrej@sika.io
@ondrejsika

Kontejnery v praxi, 22. 2. 2023

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

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

Cíl

Jednoduchá, rychlá a bezpečná distribuce aplikací

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

Současný stav (před kontejnery)

- Každý to dělá svým způsobem
  - Různí dodavatelé, co dodavatel to způsob
  - Různé požadavky (JAR, WAR, VM, ...)
  - Dokumentace, Ansible, ... (nekdy nic)
- VMWare Image
  - Vyřeší prvotní instalaci
  - Aktualizace jsou podobné první variantě

=> Společný prvek = Manuální práce a virtuální servery

Kontejnery

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

Co jsou to Kontejnery

Kontejnery jsou technologie virtualizace, které izolují aplikaci a její závislosti od okolního prostředí, na kterém běží. Kontejnery fungují na úrovni operačního systému, což znamená, že několik kontejnerů může běžet na jednom fyzickém nebo virtuálním stroji a sdílet stejný operační systém.

Každý kontejner obsahuje všechny nástroje, knihovny a soubory, které jsou potřebné pro běh aplikace, včetně operačního systému. Díky tomu je možné spustit aplikaci na jakémkoliv serveru, bez nutnosti instalace dalších závislostí.

Open Container Initiative

- Kontejnery před Dockerem nebyly kontejnery, jak je známe dnes
- První moderní kontejnery (a s nimi spojenou revoluci) přinesl až Docker
- Standard + kompatibilitu dnes zajišťuje Open Container Initiative (OCI)
- OCI spada pod Linux Foundation
- https://opencontainers.org/

Výhody distribuce SW v kontejnerech

- Velmi jasná hranice mezi aplikací a platformou
- Distribuce univerzálních binárních obrazů
- Aplikace a data jsou striktně odděleny
- Jednoduchý a exaktně definovaný interface (kontejner, Kubernetes)
- Jednoduchý způsob běhu aplikací ve velkém prostředí - Kubernetes
- Auditovatelnost, Predavaci / schvlalovací proces
- Checksums, Docker Content Trust
- Security scans, ...

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

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

```
FROM golang:1.19 as build
WORKDIR /build
COPY go.mod go.sum ./
RUN go mod download
COPY . .
RUN go build

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

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

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


"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

Kubernetes

@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
- 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

Co je Kubernetes?

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. –kubernetes.io

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

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

Unifikace prostředí

Kubernetes se ovládá pomocí YAML souboru a API. Díky tomu, že Kubernetes nás svazuje v tom, jak můžeme věci nasazovat, musíme všechny aplikace nasazovat podobným stylem a poskládané ze stejných komponent.

Už to není tak, že každý projekt byl nasazen na server jiným způsobem - vše se dělá stejně, což má za následek velmi jednoduchý onboarding do firmy nebo do projektu.

De-facto standard pro provoz aplikací

Kubernetes je dnes již velmi rozšířené řešení - od startupu po enterprise.

Stalo se vlastně takovým standardem, jako provozovat kontejnerizované aplikace a aplikace vůbec (provoz bez kontejneru nedává smysl).

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

Jednoduchý provoz v cloudu i on-premise

Všechny velké cloudy (AWS, GCP, Azure) poskytují Kubernetes. I většina menších cloudů (jako DigitalOcean), nabízí managed Kubernetes.

O Kubernetes se stará poskytovatel cloudu, vy jej jen používáte.

Pokud chcete provozovat Kubernetes na on-premise, můžete například využít RKE (distribuce Kubernetes od Rancheru) nebo si jej nainstalovat sami.

Kubernetes ekosystém

- Continuous Delivery - ArgoCD, Gitlab CI, Github Actions, ...
- Log Management - Elastic Stack, Loki, Splunk, CloudWatch, ...
- Monitoring - Prometheus Stack, CloudWatch, ...

- Backups, Disaster recovery, ...
- Service Mash, ...
- Security scans, ...

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

Helm

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

Co je to Helm Package

Helm je nástroj pro správu balíčků určených pro Kubernetes. Helm balíčky, známé také jako "charts", obsahují popis aplikace a všechny potřebné závislosti, jako jsou kontejnery, služby, konfigurace a další artefakty potřebné pro běh aplikace v Kubernetes.

Helm umožňuje jednoduché nasazení, správu a aktualizaci aplikací v Kubernetes pomocí příkazové řádky, GUI i GitOps. Umožňuje také správu různých verzí aplikace a řízení konfigurace aplikace pomocí proměnných.

Proč používat HELM

- Obsahuje konfiguraci pro Kubernetes
- Umožňuje doplnit "values" - proměnné, které mohou obsahovat specifickou konfiguraci nebo secrets
- De facto standard
- Funguje proti všem variantám Kubernetes prostředí (Kubernetes, OpenShift, OKD)
- Dá se použít s existujícím CD (ArgoCD, Gitlab CI, Fleet, ...)
- Jednoducha distribuce (v Gitu)

Hranice mezi aplikaci a Kubernetes

- Aplikace v Kuberneres používá pouze Kubernetes objekty
- Vše je popsáno v jednom nebo více Helm balíčcích
- Helm balíček nemá žádnou závislost na konkrétním prostředí
  - Struktura prostředí nehraje roli
  - Určitá závislost na verzi Kubernetes pro daný Helm balíček existuje

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

```
├── Chart.yaml
├── Makefile
├── README.md
├── templates
│   ├── NOTES.txt
│   ├── deployment.yml
│   ├── ingress.yml
│   └── service.yml
└── values.yaml
```

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

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

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

Distribuce Helm balíčku

- Helm balíček je malý .tgz (par kb, pouze archiv Kubernetes YAML manifestu)
- Nejjednodušší způsob distribuce je Git repozitář
- Artifactory, Harbor mají také podporu Helmu
  - Včetně mirroringu

Předávací process

- Předávání SW je jednoznačně definované
- Probíhá pomocí standardních nástrojů
  - Git
  - Registry
- Jednoduchá automatizace předávání a aktualizací
  - ArgoCD, Gitlab CI, ...
  - Repository Mirror, ...
- Možnost jednoduchého auditu
  - Manuální validace Helm balíčků
  - Automatizovaně - Security scan (JFrog XRAY, ...)

Závěr

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

Distribuce aplikací

Container Images + Helm Packages

(registry) (git)

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

Aplikace je

Nezavislá na runtime

Docker, Containerd, CRI-O, ...

"Nezávislá" na orchestrátoru

- Kubernetes - EKS, AKS, Tanzu, RKE2, ...
- OpenShift, OKD

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

Jasná hranice mezi aplikací a platformou

Hranice mezi kontejnerem a kontejnerizační platformou je

- Docker Images + Helm Package
- Konfigurace pomocí Helm values
- Aplikace vystavena pomocí Ingress objektu
- Data jsou v PersitentVolumes v Kubernetes
- Logy na STDOUT, STDERR - napojení na centrální log management

Žádná závislost na okolním prostředí (Kubernetes distribuce, velikost, ...)

@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