Jednotná konfigurace aplikace v kubernetes pro různá nasazení

Sklik/Devops

František Řezníček

SEZNAM.CZ

Agenda

- Špatné konce aneb takhle tedy ne přátelé …
- Jaké jsou možnosti při konfigurování aplikace
- Světlo na konci tunelu aneb jak to dělat správně
- Příklady
- Co si odneseme aka Takeaways

SEZNAM.CZ

Špatné konce aneb takhle tedy ne přátelé ...

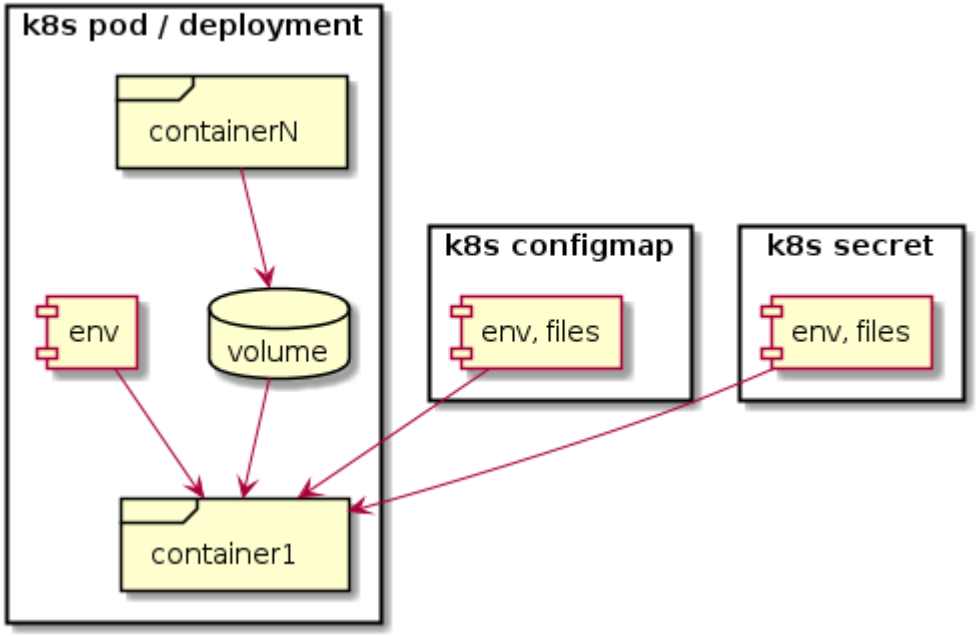
- **různé konfigurace projektu jsou v různých konfiguračních souborech s velkou mírou duplikace**
- každý projekt má vlastní sadu CI/CD skriptů používající vlastní fáze
- každý projekt má vlastní adresářovou strukturu (nekompatibilní s ostatními projekty)
- jednu konfigurační volbu v projektu lze nastavit různými způsoby v několika různých souborech
- každý projekt volí jinou cestu jak se finální balík (deb, docker image) dostává do produkce
- každý projekt má různý způsob jak poznat že daný balík odpovídá konkrétnímu stavu zdrojového repozitáře
- neexistují doporučení jak správně konfigurovat aplikace v kubernetes i jinde

Jaké jsou možnosti při konfigurování aplikace v kubernetes - hledisko aplikace

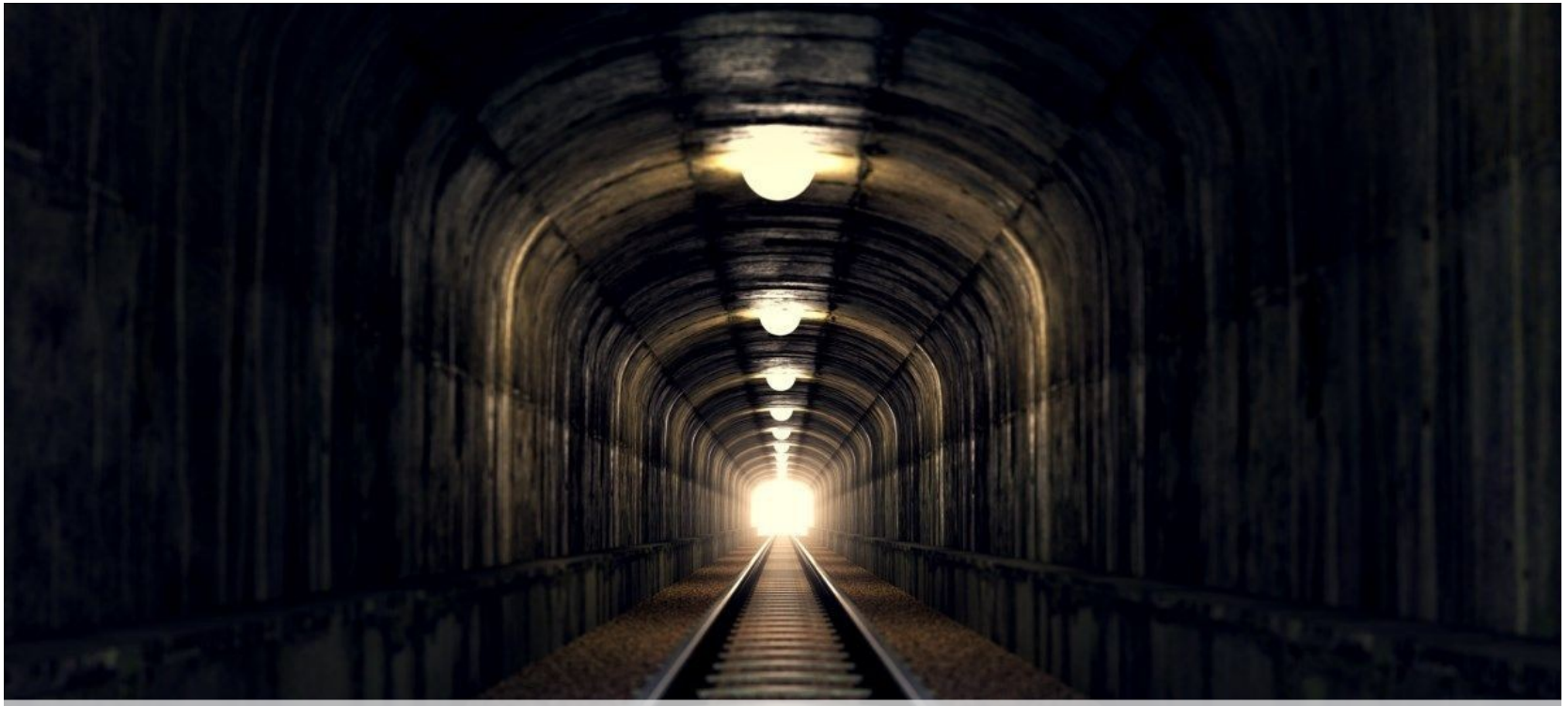
Aplikace většinou respektuje konfiguraci prostřednictvím:

- konfiguračního souboru
  - ten je statický nebo generovaný z šablony konfiguračního souboru na základě proměnných prostředí `goenvtemplatorem2`.
- přímo proměnných prostředí

Jaké jsou možnosti při konfigurování aplikace v kubernetes - hledisko aplikace v kubernetes



Světlo na konci tunelu



Pokud chceme být jako tým / produkt / firma úspěšní

- Nesnažme se za každou cenu vymýšlet vymyšlené.
- Poučme se jak z interních projektů tak externích open-source projektů
- Komunikujme a sdílejme napříč týmy více

Pojďme sjednocovat CI/CD postupy

- Používejme společnou CI/CD knihovnu ci-scripts
  https://gitlab.kancelar.seznam.cz/generic/ci-scripts
- Používejme kompatibilní způsoby automatického nasazování
  - na devu různé jmenné prostory pro master a pro staging
    - Sklik příklad (nasazování do devového kubernetes):
      - master komit nasazuje do jmenného prostoru `sklik_master`
      - komit v masteru zatagovaný dle konvence nasazuje do jmenného prostoru `sklik_staging`
  - směřujeme k automatickému nasazování i v produkci v budoucnu (autoadmins app)
- Výhody
  - zamezení duplikace CI/CD kódu
  - snažší udržování
  - standardizace

Pojďme sjednocovat adresářovou strukturu

- Používejme kompatibilní adresářovou strukturu
  - konfigurace aplikace v adresáři `conf/` (konfigurační soubory)
  - rozdíly mezi implicitní (provozní) konfigurací a devel konfigurací v souborech `conf/*.env` (např. `conf/development.env`)
  - kubernetes manifesty v souborech `kubernetes/*.yaml*`
  - mtail programy v adresáři `mtail/*.mtail*`

- Výhody
  - snazší orientace v projektech napříč týmy, usnadnění znovupoužití kódu jiným týmem

Pojďme používat konfigurační šablony I

- Při nasazování aplikace existuje hned několik prostředí do nichž chceme aplikaci nasadit:
  - produkční prostředí
  - testovací či lokální prostředí
  - vývojářské prostředí
  - předprodukční prostředí
  - ...
- Abychom zabránili duplikaci jednotlivých konfiguračních sad je třeba používat konfigurační šablony.
- Používáme GOLANG šablony (`goenvtemplator2`, `helm`, golang sprig, ...)
- Rozlišujeme tva typy šablon
  - šablony aplikačních konfiguračních souborů (v `conf/`)
  - šablony kubernetes manifestů (v `kubernetes/`)

Pojďme používat konfigurační šablony II

- Odlišení jednotlivých sad konfiguračních souborů se děje na dvou místech:
  - souborem `conf/*.env` (příklad conf/development.env)
    - tímto způsobem se ovlivňují šablony aplikačních konfiguračních souborů
  - proměnnými prostředí v CI souboru (`.gitlab-ci.yaml`, Jenkinsfile,...)
    - tímto způsobem se ovlivňují šablony kubernetes manifestů

- Výhody
  - zamezení duplikace konfiguračních souborů
  - flexibilní testování (konfigurace závislá pouze na proměnných prostředí)
  - šablony jsou zavedený způsob řešení konfiguračního hellu

Pojďme sjednocovat jednotlivé aplikační konfigurace - I

- Udržujme implicitní konfiguraci shodnou s produkční konfigurací aplikace
  - výjimku tvoří hesla, certifikáty a generované konfigmapy (z conf/*.env)
- Uložme implicitní (produkční) konfiguraci na jednom místě (v jednom konfiguračním souboru)
  - vyhněme se několika implicitním konfiguracím v různých souborech (např. entrypoint + Dockerfile + kubernetes manifest)
- Generujme další konfigurace (devová) prostřednictvím souborů `conf/*.env`
- Udržujme ostatní konfigurace co nejvíce stejné (soubory `conf/*.env` co nejkratší)

Pojďme sjednocovat jednotlivé aplikační konfigurace - II

- Výhody
  - je zřejmé jaká je produkční konfigurace
  - je zřejmé jak se liší konkrétní (testovací) konfigurace od té provozní

- Nevýhody
  - při změně produkční konfigurace je třeba tuto zpětně zanést do implicitní konfigurace aplikace

Pojďme udržet pořádek v generovaných balících / artefaktech / verzích

- Používejme důsledně semantické verzování v2 https://semver.org
  - tagujme verzi podle schématu `vX.Y.Z` nebo `<komponenta>-X.Y.Z`
- Používejme docker labely dle schematu http://label-schema.org/rc1
  - info o projektu, verzi, komitu, git repozitáři, ...
- Důsledně dodržujme tři úložiště generovaných balíků / docker obrazy / artefaktů
  - dočasné úložiště pojme artefakty generované z CI pro každý komit, automaticky je promazáváno (repo.dev/temporary, cid.dev)
  - vývojářské úložiště pojme artefakty generované z CI pro komit z masteru který je správně zatagován (repo.dev/testing, docker.dev)
  - produkční úložiště, přesun artefaktů z toho co má patřičnou kvalitu (repo.dev/stable, doc.ker)

Dodržujme (kubernetes) konvence I

- Ukládejme kubernetes manifesty do správně pojmenovaných souborů `<app>-<objekt>.yaml`
  - příklady: `export-manager-sentry-secret.yaml.example`, `frontend-api-deployment.yaml.tmpl`
- Vždy směřujme aplikace na kubernetes service objekty
  - uvnitř kubernetes je daná aplikace k dispozici na PQDN `<service-name>` nebo `<service-name>.<namespace-name>`
  - pro závislost na externí službě (např. databázi) použijeme `externalName` service
  - pro závislost na externí službě kde potřebujeme upravit i číslo portu použijeme kube2world proxy https://gitlab.kancelar.seznam.cz/Sklik-DevOps/kube2world

Dodržujme (kubernetes) konvence II

- Nezapomínejme na vyplněná metadata a ostatní požadavky:
  - label `app`
  - vývojářský tým (emailová adresa)
  - logovací formát
  - vystavování a sběr metrik
  - definované resources
  - definované readiness/livenessProbe

https://gitlab.kancelar.seznam.cz/Sklik-DevOps/DevOps/wikis/checklist-deployment-musthave

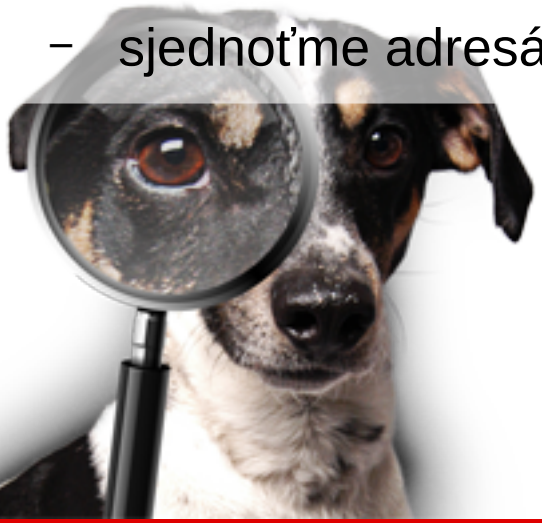
Příklady

Kompletní sada příkladů je k dispozici na https://gitlab.kancelar.seznam.cz/Sklik-DevOps/DevOps/wikis/kubernetes-app-configuration-best-practices#p%C5%99%C3%ADklady

SEZNAM.CZ

Co si odneseme / Takeways

- Nejsme v tom sami a tak není třeba vymýšlet již vymyšlené
- Existují cesty jak efektivně konfigurovat aplikace
  - sjednoťme konfigurace aplikací, vyjděme z provozní konfigurace
  - používejme konfigurační šablony pro zamezení duplikace
  - používejme standardizované CI/CD (ci-scripts)
  - používejme sémantické verzování (v2)
  - sjednoťme adresářové struktury



SEZNAM.CZ

Děkuji za Vaši pozornost!

SEZNAM.CZ

...najdu tam, co neznám!

Příklad adresářové struktury