Životní cyklus DevOps

Autor: David Müller

16.5.2024

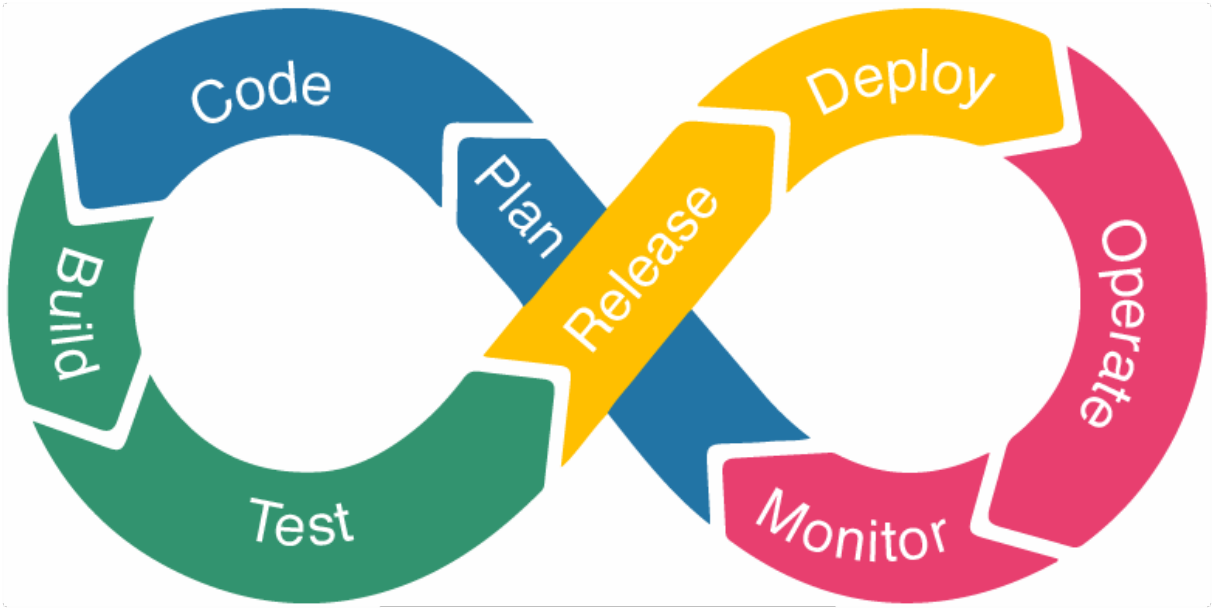
Nástroje

- GitLab - Je systém pro správu verzí, ukládání a společný vývoj kódu, umožňuje také spravovat kontejnery
- Docker - Umožňuje zabalit aplikaci do "kontejneru" a tu pak spouštět bez ohledu na operační systém
- Kubernetes - Usnadňuje práci s kontejnery, plánuje spuštění kontejnerů na virtuálních. počítačích v závislosti na jejich dostupnosti
- Rancher - Umožňuje centralizovanou správu Kubernetes klastrů (Souboru uzlů pro spuštění kontejnerizovaných aplikací), automatizované nasazení a poskytuje uživatelům intuitivní grafické rozhraní automatizovaných procesů
- HashiCorp Vault - Je nástroj pro správu tajemství a ochranu dat. Umožňuje bezpečné ukládání, přístup a správu citlivých dat
- Helm - Zjednodušuje právu aplikací v Kubernetes. Automaticky spravuje závislosti mezi různými balíčky. Umožňuje snadno instalovat a aktualizovat aplikace. Helm balíčky, známé jako "charts", obsahují vše potřebné pro instalaci a spuštění aplikace

Prostředí

- Produkční prostředí - Je fáze vývojového cyklu, ve které je software nasazen do ostrého provozu a používán reálnými uživateli
- Testovací prostředí - Je izolovaná kopie produkčního prostředí, kde testeři mohou bezpečně provádět testy bez rizika ovlivnění reálných uživatelů
- Vývojové prostředí - Je prostředí s možností manipulace se softwarem, rychlou odezvou vůči změnám, maximální přístup k debugovacím a chybovým informacím

Životní cyklus DevOps



1. Plánování (Plan)
   - Stanovení jasných a měřitelných cílů
   - Určení rozsahu projektu
   - Analýza požadavků **Koncových uživatelů**
   - Vymezení rolí a odpovědností každého člena týmu - **Developer**, **Release Manager**, **Tester**, **Support**, **DevOps Engineer**
2. Vývoj a psaní kódu (Code)
   - **Developer** píše kód podle stanovených požadavků
   - Kód se průběžně ho ukládá do **GitLabu**, který zajišťuje správu různých verzí
   - Vývojáři si navzájem kontrolují kód, aby v něm odhalili chyby a zlepšili kvalitu - Code review
   - Důležitou součástí vývoje je dokumentace, která popisuje, jak kód funguje, jak ho používat a jak ho rozšiřovat
   - **DevOps Engineer** konfiguruje a spravuje nástroje pro správu verzí
3. Sestavení aplikace z kódu (Build)
   - Uvedení vytvořeného kodu do spustitelné podoby
   - **DevOps Engineer** automatizuje procesy sestavení aplikace
   - Nahrání aplikace na **Docker** - Vytvoření Dockerfile
   - Příkazem "build" vytvoření obrazu (image)
   - Příkazem "run" spuštění obrazu a vytvoření kontejneru
   - Pro spravovaní většího množství kontejnerů lze použít **Kubernetes**
   - Pro správu Kubernetes klastrů lze využít **Rancher**
4. Testování aplikace (Test)
   - Testování ověřuje, že kód funguje správně
   - **Tester** přebírá nový build, který je automaticky nasazen na testovací prostředí pomocí **Kubernetes**
   - Provádí se automatizované testy, manuální testy, výkonové testy a bezpečnostní testy

5. Uvolnění aplikace (Release)
    - **Release Manager** kontroluje že aplikace prošla všemi potřebnými testy a ověřuje, že aplikace je stabilní a připravena k nasazení
    - Zajištění finálního schválení od všech zainteresovaných stran
    - Lze použít nástroj **Helm** - poskytuje kompletní informace o správě životního cyklu aplikace.
6. Nasazení aplikace (Deploy)
    - Implementace nové verze aplikace do produkčního prostředí u **Koncového zákazníka**
    - **HashiCorp Vault** poskytuje lepší kontrolu nad citlivými údaji a pomáhá splnit bezpečnostní standardy cloudu.
    - **DevOps Engineer** automatizuje procesy pro rychlé a spolehlivé nasazení
7. Provoz (Operation)
    - **DevOps Engineer** monitoruje aplikaci a reaguje na případné incidenty nebo problémy.
    - Lze použít **Rancher** k monitorování výkonu a dostupnosti aplikace
8. Monitorování (Monitoring)
    - **Účetní** monitoruje finanční transakce a spravuje fakturaci v e-shopu.
    - **Support** získává zpětnou vazbu od **koncových uživatelů** a poskytuje report pro další vylepšení