| Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS | |
|---|---|
| **Semestr** | **ZS 2018/19** |
| **Autoři** | **Daniel Fiala (fiad00)**<br>**Ema Kučerová (xkuce02)**<br>**Jan Hintner (hinj00)** |
| **Téma** | **"GitOps": Weaveworks Explain Their Model for Using Developer Tooling to Implement CI/CD** |


## Abstrakt

Semestrální práce se zabývá vývojem založeném na modelu GitOps, který je typem best practices známých jako DevOps. Cílem práce je představit fungování GitOps a jeho principy a vysvětlit, jak je možné využít jeho plný potenciál, jeho přínosy a použitelnost. V první řadě čtenáře seznamuje se samotným gitem, na kterém je GitOps založeno. Dále popisuje fungovaní systému Kubernetes, který je důležitou součástí při využívání GitOps. Nakonec se zaměřuje na využití best practice CI/CD, který se zaměřuje na průběžnou integraci a průběžnou dodávku často a po malých částech, které jsou základním prvkem právě GitOps.



## Klíčová slova

GitOps, DevOps, Kubernetes, Git

# Obsah

# 1 Úvod

V této semestrální práci se především zaměříme na seznámení čtenáře s pilíři moderního vývoje softwaru a jeho fungování. Jelikož se tato oblast neustále rozvíjí, s ní vznikají i nástroje, které slouží k jejímu usnadnění. Zároveň vznikají i metody a metodiky, které jsou otevřené změnám a nejsou tak radikální, nebo upravují již známé modely. Abychom mohli pochopit model GitOps a nástroje, které jsou využity k jeho aplikování, je potřeba objasnit na čem model GitOps stojí. Základem této metodiky jsou principy DevOps, které se v posledních letech stávají stále populárnějšími. Principy DevOps jsou proto v práci vysvětleny stejně, jako další nezbytné prvky pro pochopení GitOps. Těmito prvky jsou průběžná integrace (CI), průběžná dodávka (CD) a průběžné nasazování. Dále je v práci představen i samotný git a v neposlední řádě je objasněn systém Kubernetes a kontejnerizace. Ačkoliv je vývoj softwaru velmi proměnlivou oblastí, snažíme se v práci zobecnit principy a poznatky tak, aby byly aplikovatelné využitelné dlouhodobě.

# 2 Git

Git je distribuovaný systém správy verzí neboli verzovací systém. Umožňuje jednoduché verzování projektu, tedy zaznamenávat historii projektu, ve které najdeme kdy a jaké byly přidány nové funkce, nebo jaké byly opraveny chyby. Je tak možné se k těmto změnám vracet a práce je tak rychlejší a kvalitnější a snadněji se zjišťují chyby.

S využitím gitu může každý z týmu pracovat na různých částech projektu ve stejném čase, na různých místech a později je sloučit dohromady v jeden produkt. Dříve bylo nutné zálohovat

předchozí verze pro případ nouze a aby se práce od několika vývojářů na jednom projektu sloučila, upravené soubory se ručně kopírovaly. To stálo hodně času, peněz i místa na disku.

Pokud projekt sdílíme například na FTP serveru, který slouží k přenosu dat mezi vzdálenými počítači a vývojář pošle na server soubor a jinému vývojáři poté pošle soubor se stejným názvem, soubor prvního vývojáře se tak přepíše a jeho funkce ztratí. Git ale tyto dva soubory dokáže sloučit a práce v týmu je tak znatelně jednodušší a rychlejší.

Další výhodou gitu je, že nevyužívá žádný centrální server. To znamená, pokud by se již zmíněný server nějakým způsobem poškodil, soubor by nebyl ztracen, protože každý pracovník, který udělal změnu v souboru, tak má k dispozici i historii projektu. Každý tedy může vytvořit nový server, který bude sloužit jako nový distribuční server. Nicméně soubory nemusí být distribuovány, tedy šířeny mezi spolupracovníky pouze přes server, ale například přes sdílený disk, nebo e-mailem, ačkoliv přes server je to jednodušší.

## 2.1 Pull request

Takzvané „pull requesty“ jsou požadavky vytvořené vývojářem ke sloučení změn, které byly vytvořeny v hlavním projektu. Zároveň zahrnuje proces přezkoumání těchto změn. Ti, kteří změny přezkoumávají mohou vkládat komentáře ke každému kousku kódu, o kterém si myslí, že by mohl být zlepšen nebo je zlepšení opravdu nutné. Po obdržení zpětné vazby na ni může autor reagovat, zahájit diskuzi, nebo zpětnou vazbu pouze přijmout a upravit kód podle požadavku.

# 3 Průběžná Integrace a průběžná dodávka (CI/CD)

Průběžná integrace a průběžná dodávka je kultura, sbírka postupů, které umožňují týmům při vývoji aplikací častěji a spolehlivěji provádět změny v kódu. Implementace je také známá jako CI/CD pipeline a je to jeden osvědčených postupů při využití DevOps v týmu.

## 3.1 Průběžná integrace (Continuous Integration, CI)

Průběžná integrace je sadou praktik, při které jsou provedené změny okamžitě otestovány a hlášeny ve chvíli, kdy jsou přidány do obsáhlejších kódů. Průběžná integrace poskytuje rychlou

zpětnou vazbu, aby bylo v případě, kdy se vada dostane do úložiště software snadné ji co nejrychleji identifikovat a opravit.

Průběžná integrace tak redukuje rizika, příprava testů je rychlejší. Indikace stavů umožňuje zjistit, současný stav, například co funguje a co ne a jak dlouho to ještě bude trvat.

Chybám nelze předejít, ale pomocí průběžné integrace je jednodušší najít a opravit je a do budoucna jim předejít. Při dobré automatizaci testování lze snížit počet chyb. Rychlejší rozmísťování pomocí CI umožňuje rychlejší zpětnou vazbu a ruší tak bariéru mezi vývojem a zákazníky.

## 3.2 Průběžná dodávka (Continuous Delivery, CD)

Průběžná dodávka je rozšířením konceptu průběžné integrace. Zatímco CI se zabývá pouze kompilací a testováním každé verze SW, CD se zaměřuje na to, co se děje po této části. S CD je každý software, který prošel automatizovaným testováním, považován za platného kandidáta na vydání.

Díky průběžné dodávce je tedy možné implementovat do výroby nebo do rukou uživatelů udržitelným a bezpečným způsobem všechny typy změn, včetně nových funkcí, konfiguračních změn, oprav chyb a experimentů.

## 3.3 Průběžné nasazování (Continuous Deployment)

Mnoho lidí průběžné nasazování zaměňuje s průběžnou dodávkou, ačkoliv průběžná dodávka neznamená, že každá změna musí být do výroby ihned nasazena. Znamená to, že každá změna může být kdykoliv nasazena. Zatímco průběžné nasazování nemusí být pro všechny firmy ideální, průběžná dodávka je nutností pro aplikaci DevOps. Pouze při neustálém doručování kódu je jistotou, že se změny projeví během několika minut od nasazení a k nasazení může kdykoliv dojít.

# 4 Kubernetes

Kubernetes je nejznámější platforma pro správu a řízení, tedy orchestraci kontejnerů a služeb, která usnadňuje deklarativní konfiguraci a automatizaci. Orchestruje výpočetní, síťovou a ukládací infrastrukturu za pracovní zátěže uživatelů. Technická definice orchestrace je

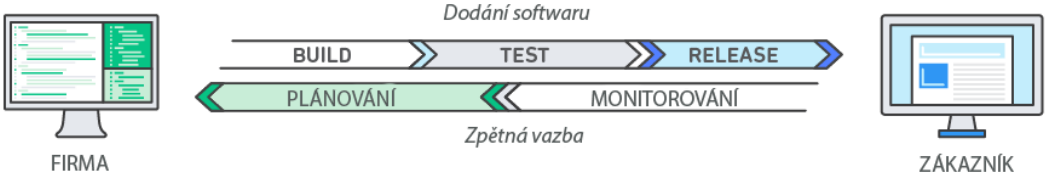
provedení definovaného pracovního postupu: nejprve proveďte A, potom B a poté C. Kubernetes je naopak sestaven ze souboru nezávislých kompozičních řídicích procesů, které průběžně řídí aktuální stav na požadovaný stav. Nezáleží na tom, jak se dostanete z bodu A do bodu C. Nevyžaduje se ani centralizované řízení. Výsledkem je systém, který je jednodušší a výkonnější, robustní, odolný a rozšiřitelný.

### 4.1 Kontejnerizace

Kontejnery jsou abstrakcí v aplikační vrstvě, které obalí kód a jejich závislosti a vzniká tak balíček, takže aplikace běží rychle a spolehlivě z jednoho výpočetního prostředí do druhého. Lze spustit více kontejnerů na jednom počítači. Kontejnery mezi sebou sdílí jádro operačního systému a každý z nich běží jako samostatný proces v uživatelském prostoru. Kontejnery zabírají méně prostoru než virtuální stroje, mohou zpracovávat více aplikací a vyžadují méně virtuálních strojů a operačních systémů.

# 5 DevOps

DevOps představuje sadu postupů, která pomáhá automatizaci mezi týmy IT a vývojem softwaru tak, aby tyto týmy mohly rychleji a spolehlivěji obstarávat činnosti s tím spojené, tedy samotná tvorba, testování a vydání softwaru. Tyto postupy jsou založeny na budování kultury spolupráce mezi týmy, které před zavedením pracovaly poměrně odděleně. Napomáhá k rychlejšímu vydávání verzí softwaru, řešení kritických chyb, lepšímu řízení a celkově vyšší důvěře mezi jednotlivými články řetězce okolo vývoje.




*Obrázek 1: Diagram zpětné vazby mezi firmou a zákazníkem (Zdroj: 2, počeštěno autorem)*

## 5.1 Součásti DevOps

### 5.1.1 Kultura

Peter Drucker, známý manažerský guru si uvědomil, jak je kultura důležitá k výkonu organizace. Říká, že „kultura konzumuje strategii na snídani“. V nedávné době Dr. Ron Westrum obhajoval „model tří kultur“, který popisuje atributy a pozorovatelné chování podnikové kultury a způsob zpracování informací. Jako tři typy kultury považuje kulturu patologickou (mocensko-orientovanou), byrokratickou (pravidly orientovanou) a generativní (výkonně orientovanou). Samotná kultura má vliv na organizační výkonnost bezpočtem způsobů.

### 5.1.2 Automatizace

Výhoda počítačů spočívá především v tom, že dělají stejný úkol stejně rychle a stejným způsobem dokola. To samé se nedá říci o lidech. Automatizace opakujících se a časově náročných úkolů, které jsou náchylné k chybám, mohou přinést velké dividendy. Nabízí se i zapojení ChatOps[1]. Současně bez ohledu na již aktuální stav automatizace ve firmě, možnosti jejího zlepšení rychlosti, konzistence a kvality jsou takřka nekonečné.

### 5.1.3 Štíhlá metodika

Stejné principy štíhlé metodiky, které jsme mohli pozorovat v oblasti výroby v 80. letech pronikají v současnosti i do světa IT. Hlavním nástrojem této metodiky je mapování toku hodnot.

### 5.1.4 Měření

Firmy v drtivé většině chtějí rychlejší tok funkcí do produkce, vyšší kvalitu a větší hodnotu. Právě proto je nesmírně důležité sledovat metriky spojené s těmito „hodnotami“ a tyto informace dále využít pro řízení zpětné vazby a rozhodování. Dohnání měření do extrémů můžeme pozorovat kupříkladu na společnosti Etsy, která měří prakticky všechno v rámci svého podniku, co vůbec měřitelné může být. Jen za rok 2013 to bylo čtvrt milionu časových řad.

[1] Použití chatovacích klientů, chatbotů a komunikačních nástrojů v reálném čase, které usnadňují způsob, jakým jsou komunikační a provozní úkoly komunikovány a prováděny.

Měření důležitých aspektů inženýrských a obchodních operací poskytuje cenné informace, umožňující rychleji reagovat a zlepšovat se.

### 5.1.5 Sdílení

To, s jakou jednoduchostí a překážkami kolují informace ve firmě souvisí s její organizační výkonností. Míra, do které organizace sdílí informace, je přímo ovlivněna její kulturou. Existuje spousta indikátorů sdílení, jako code review (posouzení/recenze kódu), lunch-and-learn meetingy atp. Čím otevřenější je organizace, když jde o sdílení a komunikaci, tím lépe jí to umožní fungovat.

## 5.2 Přínosy

Při správné implementaci je DevOps schopno transformovat fungování softwarového inženýrství ve firmě a vytvořit tak hodnotu pro zaměstnance i zákazníky, což přispěje ke zvýšení výkonu a hodnoty podniku jako takového. Plně a vhodně implementované může přinést mimo jiné níže uvedené výhody.

### 5.2.1 Zvýšená kvalita služby

Vnímaná kvalita služby a její spolehlivost závisí na její dostupnosti bez závažných chyb (MTTF[2]) a na schopnosti co nejrychleji uvést službu do stabilního a bezchybného stavu při výskytu nějaké chyby (MTTR[3]). Díky možnosti rychlé zpětné vazby a vysoké rychlosti dodávání softwaru jsou nedostatky služby odstraňovány mnohem rychleji, než tomu bylo doposud (zkrácení doby MTTR), což vede ke zlepšení vnímané kvality a spolehlivosti služby. [3]

### 5.2.2 Zvýšená spolehlivost poskytování služby

Včasnost doručení objednané služby je mimořádně důležitým faktorem spokojenosti zákazníka. Zvýšení, s jakou rychlostí je možné zavést změny ve službě na základě poskytnuté zpětné

[2] Mean-Time-To-Failure – Střední doba mezi poruchami. Statistická veličina, která slouží k ohodnocení spolehlivosti.

[3] Mean-Time-To-Repair – Střední doba do opravy. Průměrná doba nutná k opravě měřeného subjektu.

vazby, což je v nesmírně důležité v nejistých a často se měnících podnikových prostředích. S využitím agilního přístupu a DevOps jsou velké projekty rozděleny do menších, které jsou nepřetržitě dodávány. Jak počáteční, tak i následné požadavky a změny jsou dodávány za mnohem kratší dobu, přičemž další změny jsou možné kdykoliv.

### 5.2.3 Zvýšená hodnota zákazníkům skrze reakci na změny

Je velmi důležité, aby se podniky neustále přizpůsobovaly rychle se měnícímu prostředí, reagovaly na konkurenci a zůstaly v inovačním cyklu. Všechny tyto aspekty ovlivňují, zda firma zůstane relevantní a konkurence schopná. Prakticky je nutné fungovat ve světě nejistoty. Toto je příčinou, že v době, kdy vznikne nová myšlenka, tak většinu detailů projektu ani nelze znát. Dokonce ani poté, co už jsou definovány, musí být neustále předmětem další změny. Proto je důležité moci na změny reagovat co možná nejrychleji.

### 5.2.4 Lepší použitelnost služby

Rychlejší dodávání a změny služby vedou k mnohem kratší době potřebné pro obdržení zpětné vazby od zákazníka. Díky tomu zákazník dostává včasné a časté aktualizace, což opět umožňuje dříve vyhodnotit a otestovat jeho spokojenost a reakci už v počátku zavedení změny. Možností je i vyzkoušet více verzí produktu současně mezi různými skupinami zákazníků. To umožní porovnávat různé funkce nebo schopnosti svých produktů navzájem.

### 5.2.5 Efektivita operací

Agilní vývoj a DevOps jsou postaveny na štíhlých principech (Lean principles). Jejich podstatou je zaměření na maximalizaci kvality, minimalizaci zbytečných výrobních kroků a zvyšování hodnoty produktu tím, že budeme dodávat přesně to, co zákazník požaduje, a to navíc ve stanovený čas. Právě minimalizací zbytečných kroků, jako jsou čekací doby nebo režijní náklady, které jsou spojeny s průběžným zlepšováním, jsou základním principem. To pomáhá nejen agilnímu vývoji, ale má také pozitivní vliv na náklady, jelikož omezuje jejich růst.

### 5.2.6 Redukce úzkých míst

Rozostření hranice mezi Dev a Ops zahrnuje učení jedné strany o činnostech té druhé. Díky tomu mají zaměstnanci možnost pochopit proces jako celek, snáze nalézt slabá místa a následně ho i vylepšit. Například pokud zaměstnanci z části Operations (Ops) znají základy

programování a naopak, mohou být odstraněny překážky tím, že nebude nutné se spoléhat na vysoce specializované osoby.

# 6 GitOps

GitOps je způsob, jak provádět průběžnou integraci (Continuous Integration), která slouží mimo jiné k urychlení nalezení nedostatků a chyb u softwarových projektů ve fázi vývoje. Využívá git jako jediný zdroj „pravdy“ pro deklarativní infrastrukturu a aplikace. S gitem, jakožto centrem kontinuálního dodání (delivery pipeline), může každý vývojář provést takzvané „pull requesty“ a zjednodušit tak nasazení aplikací i provozních operací společnosti do Kubernetes. S využitím známých nástrojů, jako je Git, jsou vývojáři produktivnější a mohou se více zaměřit na vytváření nových funkcí než na operační úkoly.

GitOps staví na neměnné infrastruktuře. Plně využívá posun směrem k neměnné infrastruktuře a deklarativní kontejnerové orchestraci. Aby bylo možné minimalizovat riziko změny po nasazení, a to jak zamýšleného nebo nahodilého, je nezbytné udržovat reprodukovatelný a spolehlivý proces nasazení. Celkový požadovaný stav celého systému je popsán právě v gitu. Používají se takzvané „kontejnery“ právě pro neměnnost, společně s dalšími nástroji v cloud native[4] prostředí, jako jsou Terraform[5] nebo Ansible[6] pro automatizaci a správu konfigurace. Tyto nástroje společně s kontejnery a deklarativní povahou Kubernetes poskytují skvělou kombinaci pro úplné zotavení v případě úplného zhroucení.

## 6.1 Přínosy

Využívání pull requestů pro správu infrastruktury se může zdát jako poněkud zvláštní postup. Když se ovšem podíváme pozorněji, uvědomíme si, že tato praxe má veliký smysl, která přináší množství přínosů pro fungování společnosti. Ve firmě bude jediný nástroj a rozhraní pro ovládání její infrastruktury. Díky tomu se vyloučí potřeba různých nástrojů pro řízení různých

[4] Přístup k vytváření a spouštění aplikací využívajících výhody modelu poskytování cloud computingu

[5] Software pro IaC (Infrastructure as Code – Infrastruktura jako kód)

[6] Software, který vytváří platformu pro konfigurační správu a řízení počítačů skrze SSH nebo PowerShell

typů infrastruktury. Současně poskytuje kontrolu verzí všech změn provedených v konfiguraci. To je velmi užitečné pro navrácení provedených změn, stejně jako pro účely auditu. K detekci provedených změn a automatickému generování upozornění dostáváme možnost využít „diff“, který ve zkratce zobrazuje změny mezi jednotlivými větvemi, stromy nebo například dvěma soubory na disku. To znamená nejen to, že můžeme neustále sledovat změny, ale také to, že pokud se skutečné podmínky odchylují od toho, jak se mají věci nakonfigurovat, bude problém snadno rozpoznán a vyřešen. Bezesporu velkým přínosem je široká znalost gitu mezi vývojáři, není tedy nutné školit týmy s novým nástrojem, který by infrastrukturu spravoval. Ačkoliv se nejedná o primární zamýšlený účel nástroje, jedná se o praxi, která přesto přináší významnou přidanou hodnotu.

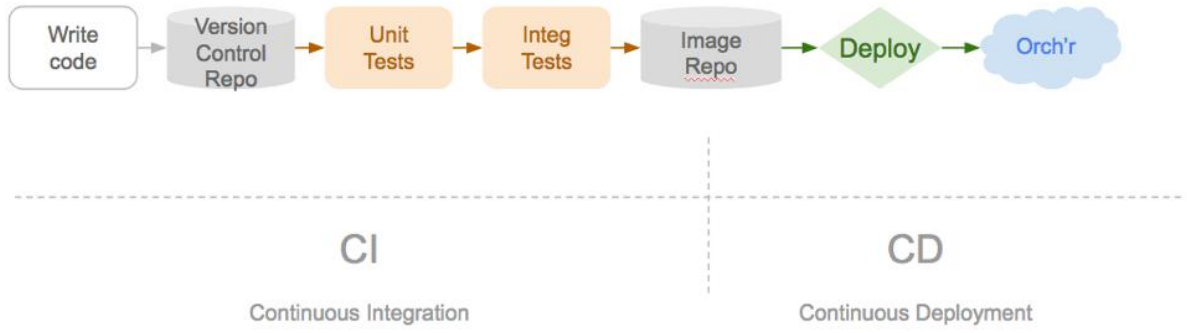
### 6.1.1 Bezpečnostní přínosy

Ačkoliv nebyl GitOps primárně navržen s ohledem na bezpečnostní potřeby IT, může poskytnout jisté přínosy i pro bezpečnost, a to hlavně díky tomu, že pomáhá týmům realizovat jednotné nástroje a sledovat data. V důsledku toho snižuje počet proměnných v řízení infrastruktury a poskytuje hlubokou a nepřetržitou viditelnost stavu dané infrastruktury, stejně jako to, jak tento stav vypadá v porovnání s požadovaným stavem podle konfiguračních dat. Méně proměnných znamená menší „plochu“ zranitelnosti a snížené riziko, že se něco pokazí, zatímco větší viditelnost usnadňuje odhalování bezpečnostních problémů, když vzniknou. To samozřejmě ale neznamená, že samotný GitOps je řešením všech bezpečnostních problémů, spíše k jejich řešení napomáhá.

## 6.2 GitOps pipeline

Ve Weaveworks postupují při zavedení jejich služby Weave Cloud na AWS[7] tak, že jejich end to end pipeline je napojena přímo do jejich clusterů a zahrnuje každý krok v jejich tvorby, nasazení, řízení a monitorování životního cyklu. Jejich pipeline se lehce liší od toho, co jsou ostatní zvyklí obvykle dělat. V části CI probíhají testy, jejich výsledek je doručen do image uložiště kontejneru a systém CD nasadí automaticky kontejner (případně manuálně nebo na požádání), viz obrázek.

[7] Amazon Web Services – platforma pro on-demand cloud computing

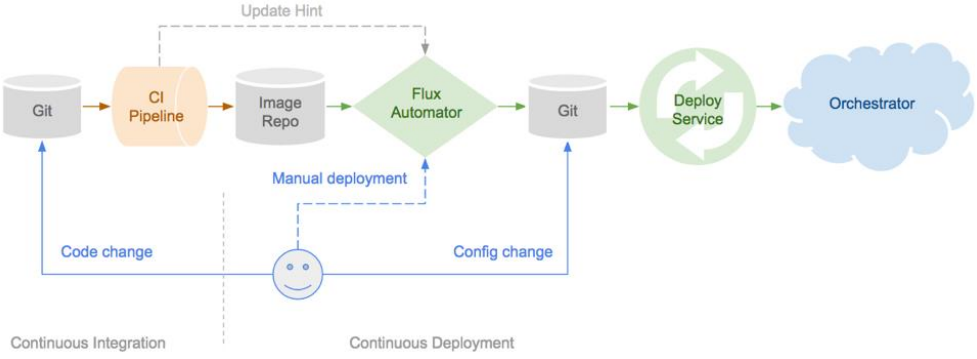



*Obrázek 2: Ukázka zjednodušeného diagramu Weaveworks pipeline [8]*

Automatizovaný git a synchronizace clusteru je velmi podstatnou částí. Je zásadní pro to, jak ve Weaveworks GitOps dělají. Za tímto účelem vytvořili nový nástroj zvaný Weave Flux, který umožňuje nasazení GitOps a pochopení, jak řídit nasazení Kubernetes. Současně automatizuje uvolňování kontejnerů do Kubernetes, stejně jako služeb, nasazování nebo síťovou politiku. Jedná se tedy o takové „lepidlo“, které mapuje všechny vztahy mezi kódem, službami a spuštěnými clustery.

Hlavní funkcí Fluxu je automatizovaná synchronizace, která zajišťuje, že pokud provedete změny ve vašem repozitáři, jsou tyto změny automaticky nasazeny do vašeho clusteru. Jedná se o jednoduché, ale dramatické zlepšení současného stavu.

Následující diagram znázorňuje, že vše před nasazením je uloženo v gitu. Součást implementace je rozdělena na dvě části. Automatizační část Flux, která sleduje nové sestavení a aktualizace konfigurací pro vytvoření nové verze a synchronizační část Flux, která zajišťuje správný stav. Oproti předchozímu diagramu, který byl zjednodušením, představuje skutečnou podobu Weaveworks pipeline.




*Obrázek 3: Ukázka podoby Weaveworks pipeline [8]*

Doporučuje se, aby systém starající se o sestavení generoval obraz kontejneru a nasadil je do clusterů z uložiště imagů. Kontejnery jsou velmi důležité, jelikož ve Weaveworks používají neměnný vzor serveru. Jakmile je tedy kód v gitu jednou odeslán, nesmí se nic dalšího měnit. Tímto chtějí minimalizovat riziko odchylky od zamýšleného stavu systému. Nevylučuje se ani používání virtuálních strojů, ovšem pomocí kontejnerů je celý proces usnadněn a je méně pravděpodobné, že dojde k chybě.

### 6.3 Budoucnost GitOps

Je poměrně zřejmé, že koncept GitOps je úzce spojen s jedním dodavatelem softwaru (Weaveworks) a jedním otevřeným nástrojem (Git). To by mohlo některé lehce odradit, jelikož spousta vývojářů nerada spoléhá na nástroje a koncepty, které jsou vázány na konkrétní firmy. Ovšem i pokud se budete snažit vyhnout specifickému přístupu ke GitOps, který Weaveworks podporuje, je jeho širší pojetí velkou hodnotou. Současně by někoho mohlo napadnout, že by chtěl využít jiný verzovací systém nebo si dokonce vytvořit vlastní. Na tomto prakticky nezáleží. Konečná hodnota GitOps spočívá v jeho schopnosti zjednodušit způsob správy infrastruktury a nasazení aplikací, řízení změn a transparentnosti procesů v organizaci. Využití konkrétních nástrojů není tedy to podstatné. Tím podstatným jsou výsledky.

# 7 Závěr

Cílem této semestrální práce bylo seznámit čtenáře s principy a fungováním nově vzniklé metodiky GitOps, kde jediným zdrojem pravdy jsou technologie a služby git. Pro správné pochopení této metodiky a jejích nástrojů jsme v práci popsali DevOps, které je základem pro GitOps a jeho prvky jsou pouze transformovány na nástroje git. Zároveň jsme se zaměřili na průběžnou integraci (CI), průběžnou dodávku (CD) a průběžné nasazování, které tvoří jeden z hlavních pilířů GitOps.

V úvodu se zaměřujeme na seznámení čtenáře se samotným gitem, jakožto verzovacím systémem a základem pro GitOps, které klade důraz na to, že vše musí být na gitu. Následně je objasněna kontejnerizace a představen systém pro správu kontejnerů Kubernetes na kterém GitOps staví. Poté objasňujeme již zmíněné pipeliny CI/CD/CDE, jelikož GitOps zároveň klade důraz na automatizaci. Další část se věnuje samotnému DevOps jeho základům a přínosům do businessu.

Poslední část semestrální práce je věnována samotnému GitOps. Nejprve je objasněno o co se jedná a jaké jsou spojitosti s předchozími částmi práce. Následně jsou popsány přínosy využívání GitOps samotného a následně i bezpečnostní přínosy tohoto řešení. Poté je popsána samotná pipeline GitOps a nástroj vytvořený týmem Weaveworks, který stojí za metodikou GitOps. Je to nástroj Weave Flux a slouží právě k nasazování GitOps v praxi.

Nakonec je pak v práci zmíněna potenciální budoucnost této metodiky a vysvětleno, že není tak podstatné, jaké konkrétní nástroje jsou využity, ale podstatné jsou výsledky, transparentnost, automatizace, zjednodušování, čehož lze vhodně dosáhnout právě díky GitOps.

# 8 Zdroje


Alexis, 2017. *GitOps - Operations by Pull Request* [online]. Dostupné z: https://www.weave.works/blog/gitops-operations-by-pull-request

Alexis, 2018. *What Is GitOps Really?* [online]. Dostupné z: https://www.weave.works/blog/what-is-gitops-really

Alexis. *The GitOps Pipeline - Part 2* [online]. 2017 [cit. 2018-12-17]. Dostupné z: https://www.weave.works/blog/the-gitops-pipeline

BRYANT, Daniel, 6.9.2018. *"GitOps": Weaveworks Explain Their Model for Using Developer Tooling to Implement CI/CD* [online]. Dostupné z: https://www.infoq.com/news/2018/09/gitops-weaveworks?utm_campaign=rightbar_v2&utm_source=infoq&utm_medium=news_link&utm_content=link_text

CAUM, Caum. *Continuous Delivery Vs. Continuous Deployment: What's the Diff?* [online]. 2013 [cit. 2018-12-17]. Dostupné z: https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff

DOLNÍČEK, Lukáš a Stephen DOMBROSKI. *Štíhlé principy a procesně orientovaná výroba* [online]. 2013 [cit. 2018-12-17]. Dostupné z: https://www.systemonline.cz/rizeni-vyroby/stihle-principy-a-procesne-orientovana-vyroba.htm

MORELLI, Brandon. *Trunk-based Development vs. Git Flow* [online]. 2017 [cit. 2018-12-17]. Dostupné z: https://codeburst.io/trunk-based-development-vs-git-flow-a0212a6cae64

RILEY, Chris. *GitOps 101: What Is GitOps, and Why Would You Use It?* [online]. 2018 [cit. 2018-12-17]. Dostupné z: https://www.twistlock.com/2018/08/06/gitops-101-gitops-use/

SACOLICK, Isaac, 2018. *What is CI/CD? Continuous integration and continuous delivery explained* [online]. Dostupné z: https://www.infoworld.com/article/3271126/ci-cd/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

SCHMIDT, Michael. *Ten ways DevOps will benefit your IT department* [online]. 2015 [cit. 2018-12-17]. Dostupné z: https://devops.com/ten-ways-devops-will-benefit-department/

*Co znamená DevOps?* [online]. [cit. 2018-12-17]. Dostupné z: https://cs.atlassian.com/devops

*GitOps* [online]. Dostupné z: https://www.weave.works/technologies/gitops/

*What is Kubernetes?* [online]. Dostupné z: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

*What is Continuous Delivery?* [online]. Dostupné z: https://continuousdelivery.com

*GitOps* [online]. Dostupné z: https://www.weave.works/technologies/gitops/

*What is a Container: A standardized unit of software* [online]. [cit. 2018-12-17]. Dostupné z: https://www.docker.com/resources/what-container?fbclid=IwAR2Ga2wIYXTvKald_yBw4f7Pd3Ksa_tosSBRbAFXW5FXN53XeTkBNDOgqlI

*What is Continuous Delivery?* [online]. Dostupné z: https://continuousdelivery.com

*What is DevOps?* [online]. [cit. 2018-12-17]. Dostupné z: https://aws.amazon.com/devops/what-is-devops/