Univerzita Pardubice
Fakulta elektrotechniky a informatiky

DevOps

Distanční opora

Ing. Roman Diviš, Ph.D.

DevOps

Úvod do DevOps

Úvod do DevOps

DevOps je soubor postupů a kulturních filozofií zaměřených na zkrácení životního cyklu vývoje softwaru, posílení spolupráce mezi vývojovými a provozními týmy a zlepšení kvality produktů. Termín "DevOps" v sobě spojuje slova "vývoj" a "provoz", což odráží jeho cíl integrovat a zefektivnit tyto tradičně oddělené role. DevOps vznikl z potřeby zlepšit agilitu poskytování IT služeb, vyvinul se z agilní metodiky a přizpůsobil její principy kontextu softwarových operací.

Mezi hlavní cíle DevOps patří zrychlení doby uvedení na trh, snížení míry selhání nových verzí, zkrácení doby mezi opravami a zlepšení střední doby do obnovy. Podporou kultury spolupráce a integrace pomáhá DevOps odstranit úzká místa, zlepšit efektivitu a zvýšit celkovou produktivitu. Mezi výhody patří:

- Zrychlení doby nasazení: Automatizací a zefektivněním procesu nasazení mohou organizace rychleji vydávat nové funkce a opravy.
- Zlepšení spolupráce a komunikace: Odbourání sil mezi týmy zvyšuje transparentnost a efektivitu.
- Zvýšení efektivity a snížení nákladů: Automatizace snižuje množství manuální práce, chyb a spotřebu zdrojů.
- Zvýšení kvality a spokojenosti zákazníků: Nepřetržité smyčky zpětné vazby zajišťují, že potřeby zákazníků a standardy kvality jsou plněny okamžitě.

Klíčové koncepty DevOps

Kontinuální integrace (CI - continuous integration)

Kontinuální integrace zahrnuje automatickou integraci změn kódu od více přispěvatelů do hlavního úložiště několikrát denně. Mezi klíčové postupy patří udržování jediného zdrojového úložiště, automatizace sestavování a ověřování každé integrace pomocí automatizovaných testů. Tím se zajistí včasné odhalení chyb integrace a omezí se problémy s integrací.

Kontinuální dodávání (CD - continuous delivery) a kontinuální nasazování (CD continuous deployment)

Kontinuální dodávání rozšiřuje CI tím, že zajišťuje, aby software mohl být kdykoli uvolněn do produkce. Zahrnuje automatizaci procesu uvolňování tak, aby bylo možné nasadit jakoukoli verzi do jakéhokoli prostředí s minimálním manuálním zásahem. Kontinuální nasazování jde ještě o krok dále tím, že automaticky nasazuje každou změnu, která projde pipeline, do produkce.

Životní cyklus DevOps

Životní cyklus DevOps je nepřetržitá smyčka sestávající z osmi fází:

1. **Plan:** Iniciace projektů, definování požadavků a plánování verzí.
2. **Code:** Psaní, kontrola a integrace kódu a správa řízení verzí.
3. **Build:** Převádění kódu do spustitelných řešení a automatizace sestavení.
4. **Test:** Automatizace a provádění testování kódu za účelem zajištění kvality.
5. **Release:** Provádění testů a testování: Správa schvalování vydání a automatizace procesů vydání.
6. **Deploy:** Vydávání nových verzí: Automatizace a správa procesů nasazení.
7. **Operate:** Provozování: Správa a podpora provozních systémů.
8. **Monitor:** Monitorování aplikací a infrastruktury z hlediska výkonu a stavu.

Terminologie

- **Infrastruktura jako kód (IaC):** Správa a zajišťování infrastruktury prostřednictvím kódu namísto manuálních procesů.
- **Mikroslužby:** Architektonický styl, který strukturuje aplikaci jako soubor malých autonomních služeb.
- **Kontejnerizace:** Zapouzdření nebo zabalení softwarového kódu a všech jeho závislostí tak, aby mohl být jednotně a konzistentně provozován v jakékoli infrastruktuře.
- **Orchestrace:** Automatizace správy, koordinace a uspořádání složitých počítačových systémů, služeb a middlewaru.

Přehled často užívaných nástrojů v rámci DevOps praktik

- **Git (Mercurial):** Distribuovaný systém správy verzí pro sledování změn ve zdrojovém kódu během vývoje softwaru.
- **Jenkins (TeamCity, Circle CI, ArgoCD):** Automatizační server, který se používá k automatizaci fází sestavování, testování a nasazování softwaru.

- **Docker (Podman, LXC, LXD):** Platforma pro vývoj, přenášení a spouštění aplikací v lehkých kontejnerech.
- **Kubernetes (Docker Swarm):** Systém pro automatizaci nasazení, škálování a správu (orchestraci) kontejnerových aplikací.
- **Ansible (Chef, Puppet)**: Systém automatizované správy konfigurace (a softwaru) na serverech s využitím definovaných pravidel

Výzvy a řešení v oblasti DevOps

Zavádění DevOps může představovat výzvy, jako je kulturní odpor, integrace starších systémů a bezpečnostní rizika. Mezi řešení patří podpora kultury neustálého učení, integrace bezpečnostních postupů do procesu DevOps (DevSecOps) a používání modulárních architektur pro zlepšení přizpůsobivosti systému.

DevOps není jen soubor nástrojů nebo postupů, ale kulturní změna, která podporuje spolupráci mezi tradičně oddělenými rolemi vývoje softwaru a provozu IT. Přijetím postupů DevOps mohou organizace zvýšit efektivitu, kvalitu a spokojenost zákazníků.

Naposledy změněno: čtvrtek, 21. března 2024, 09.36

◄ Video - úvod do DevOps

Přejít na...

DevOps roadmap (přehledová cesta technologií a nástrojů ve světě DevOps) ►

DevOps

DevOps vs SRE

DevOps vs. SRE (Site Reliability Engineering)

- **DevOps:** Koncept DevOps vznikl z potřeby zvýšit efektivitu organizace překlenutím rozdílu mezi vývojem softwaru (Dev) a provozem IT (Ops). Jejím účelem je zkrátit životní cyklus vývoje systému, zajistit nepřetržité dodávání a zvýšit kvalitu a spolehlivost softwaru prostřednictvím společného a iterativního přístupu.
- **SRE (Site Reliability Engineering):** Cílem SRE, které vymyslela společnost Google, je vytvářet škálovatelné a vysoce spolehlivé softwarové systémy. Tento termín a praxe zavádějí systematičtější přístup k dosažení spolehlivosti prostřednictvím inženýrských a provozních postupů se zaměřením na automatizaci, škálovatelnost a metriky spolehlivosti. SRE lze považovat za implementaci DevOps se specifickým důrazem na spolehlivost.

Klíčové zásady:

- **DevOps:**
  - Spolupráce a komunikace mezi vývojáři a provozem IT.
  - Automatizace procesu nasazení (CI/CD).
  - Neustálé zlepšování a učení.
- **SRE:**
  - Důraz na inženýrská řešení provozních problémů.
  - Měření a monitorování za účelem stanovení referenčních hodnot a cílů spolehlivosti.
  - Sdílení odpovědnosti mezi vývojáři a provozovateli s cílem vytvořit škálovatelné a spolehlivé systémy.

Praktiky a nástroje:

- **DevOps:** Zahrnuje postupy jako kontinuální integrace (CI), kontinuální dodávka (CD), infrastruktura jako kód (IaC) a monitorování a protokolování. Mezi nástroje často spojené s DevOps patří Jenkins, Git, Docker, Kubernetes a Ansible.
- **SRE:** Zavádí postupy, jako jsou cíle úrovně služeb (SLO), ukazatele úrovně služeb (SLI), rozpočty chyb a snižování pracnosti. Mezi běžné nástroje patří Prometheus pro monitorování, Terraform pro infrastrukturu jako kód a různé nástroje pro logování a správu incidentů.

Cíle a metriky:

- **DevOps**: Cílem je zkrátit dobu uvedení na trh, zvýšit četnost nasazení a zlepšit celkový životní cyklus aplikací od vývoje po provoz. Metriky se často zaměřují na četnost nasazení, dobu realizace změn, míru selhání změn a střední dobu do obnovy (MTTR).
- **SRE**: Zaměřuje se na spolehlivost a provozuschopnost systému a usiluje o splnění nebo překročení předem stanovených cílů úrovně služeb. Metriky v oblasti SRE zahrnují ukazatele úrovně služeb (SLI), cíle úrovně služeb (SLO) a rozpočty chyb, které kvantifikují přijatelné úrovně rizika služeb.

Kulturní aspekty:

- **DevOps:** Podporuje kulturu spolupráce, při níž se odbourávají bariéry mezi vývojovými a provozními týmy. Tento kulturní posun podporuje sdílenou odpovědnost, transparentnost a neustálé zlepšování.
- **SRE:** SRE sice sdílí étos spolupráce DevOps, ale zdůrazňuje disciplinovanější přístup k měření a dosahování spolehlivosti. Kultura SRE se řídí daty a klade velký důraz na automatizaci a omezení manuální práce nebo "dřiny".

Závěr:

Přestože cílem DevOps i SRE je zlepšit výkonnost a spolehlivost softwarových systémů, přistupují k tomuto cíli z trochu jiných úhlů. DevOps se zaměřuje na zlepšení celého životního cyklu vývoje softwaru prostřednictvím spolupráce a automatizace. Naproti tomu SRE se zaměřuje konkrétně na spolehlivost systému, a to pomocí souboru inženýrských postupů a provozních strategií. SRE však lze považovat za specifickou implementaci principů DevOps s přidaným důrazem na metriky a standardy spolehlivosti. V praxi se tyto disciplíny významně překrývají a mohou se vzájemně doplňovat při dosahování vysoce efektivního a spolehlivého procesu dodávání softwaru.

Naposledy změněno: čtvrtek, 21. března 2024, 08.43

DevOps

Praktické seznámení s verzovacím nástrojem Git

Git

Základní charakteristika Gitu

- populární verzovací systém
- open source, silná komunita, aktivně vyvíjen
- distribuovaný verzovací systém (každý klient má typicky u sebe dostupnou kompletní kopii repozitáře a jeho historie)
- základ pro mnoho online platforem pro hostování repozitářů - GitHub, GitLab, BitBucket, ...
- veškeré operace se provádějí přes konzolový nástroj Git, existuje řada nastaveb a implementací v rámci rozličných IDE a editorů

Základní Git příkazy

Přesuňte se do cílové složky - **cd /cesta**

Inicializujte nový Git repozitář pomocí **git init**. Příkaz init vytváří nový repozitář, v praxi to znamená, že dojde k vytvoření složky ".git", která obsahuje základní metadata repozitáře a později i jeho kompletní obsah. Nově vytvořený repozitář neobsahuje žádný commit a je nastavena výchozí větev (branch) **main** (příp. **master**, starší varianta výchozí větve).

Ve složce vytvořte soubory **index.html** a **style.css** a vložte do něj nějaký obsah. Nyní je čas vytvořit commit. Soubory je nejdříve nutné zařadit do sledování v rámci repozitáře - externí soubory jsou ve výchozím stavu zcela ignorovány. Příkazy **git add index.html** a **git add style.css** zařadí soubory do sledovaných. Nyní je možné vytvořit commit se stanovenou commit message **git commit -m "první commit"**.

Výše uvedené buď úspěšně vytvořilo commit nebo skončilo chybovým hlášením, že dosud není git nakonfigurován. Pokud poprvé používáte Git, je nutné nastavit jméno a email, který bude v metadatech commitů figurovat jako autor změny. Provedením **git config --global user.name "Jméno Autora"** a **git config --global user.email "autor@email.cz"** je dokončena nezbytná konfigurace. Nyní je třeba příkaz commit zopakovat.

Výše uvedený postup je možné opakovat a vytvořit další commity. Pro soubory, které jsou již sledovány v repozitáři lze užít zkratku a commitnout je všechny bez přidávání pomocí **add**, a to příkazem **git commit -am "automaticky potvrdí všechny sledované soubory"**.

Historii repozitáře si lze prohlédnout s využitím **git log** či základní grafické nastavby **gitk**. Zobrazení v **git log** lze vylepšit mnoha přepínači, např.: **git log --pretty --graph --oneline** zobrazí přehledný ascii graf v konzoli.

Aktuální stav repozitáře je možné kdykoliv zobrazit příkazem **git status**. Příkaz je rovněž užitečný ke sledování označených změn ke commitnutí, průběhu merge/rebase procesu aj.

Větvení / branching

Pro vytvoření nové vývojové větve se užívá příkaz **git branch nazev-nové-větve**, příkaz ji pouze vytvoří, aktivní větev není změněna. Přepnout se na jinou větev lze pomocí **git switch název-větve** (nebo starší varianta **git checkout název-větve**).

Pokud byl vývoj v jiné větvi dokončen a je čas zahrnout změny do hlavní vývojové větve, je třeba se přepnout do cílové (hlavní vývojové) větve **git switch main** a následně provést sloučení **git merge název-vývojové-větve**. Tento proces je automatizován, ale ne vždy dokáže všechny změny sloučit automaticky, může dojít ke konfliktům. Stav je oznámen do konzole a lze jej aktivně sledovat pomocí **git status**. Pokud dojde ke konfliktům, je nutné je ručně upravit (v textovém editoru) a následně potvrdit změny pomocí **git add** a poté dokončit merge.

Synchronizace se vzdálenými repozitáři

Pro práci se vzdálenými repozitáři slouží příkazy **fetch** a **push**, jako jejich nastavby pak fungují i příkazy **pull** a **clone**.

Příkaz **git clone** je užitečný ve chvíli, kdy nechceme zakládat nový repozitář, ale vycházet ze vzdáleného repozitáře. Pomocí **git clone https://url-vzdáleného-repozitáře** dojde k provedení procesu klonování vzdáleného repozitáře a tento vzdálený repozitář je následně nastaven jako vzdálený repozitář pod názvem **origin**. Spravovat vzdálené repozitáře lze pomocí **git remote**. Typicky příkazem **git remote add origin https://cesta-k-repo-origin** se provádí provázání právě s výchozím vzdáleným repozitářem, ale vzdálených repozitářů může být k dispozici více a pod různými názvy.

Příkaz **git fetch** stahuje změny ze vzdáleného repozitáře, ale neprovádí jejich sloučení s aktivní větví. Zkratka **git pull** provede i okamžitý merge (nebo jinou operaci dle konfigurace) do lokální větve.

Pro zaslání změn na vzdálený repozitář se využívá příkaz **git push [specifikace-zaslaných-větví]**. Typické použití zahrnuje při prvním pushnutí provedení **git push -u origin main**, kdy je nastavena "upstream" větev a poté je možné provádět push do této větve pouze příkazem **git push**.

Praktické cvičení

V rámci procvičení činností s Gitem:

- vytvořte nový prázdný repozitář
- vložte do něj zdrojové kódy jednoduché aplikace (Hello world postačí)
- vytvořte novou vývojovou větev a proveďte na ní změny
- vytvořte konfliktní změny (stejný soubor, stejné místo v souboru) i na větvi main
- proveďte merge
- vytvořte repozitář na službě jako je GitHub, GitLab či jiné
- nahrajte obsah repozitáře do vzdáleného repozitáře

Naposledy změněno: neděle, 24. března 2024, 14.34

◄ Průběžné testování znalostí - DevOps koncepty a terminologie

Přejít na...

DevOps principy ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat
Stáhněte si mobilní aplikaci

DevOps

DevOps principy

Přehled principů DevOps

Základní principy DevOps jsou navrženy jako vodítko pro postupy, které jsou základem kooperativního a vysoce efektivního přístupu k vývoji a nasazování softwaru. Pochopení těchto zásad je pro efektivní implementaci postupů DevOps klíčové. Zaměřují se na zlepšení spolupráce mezi vývojovými a provozními týmy, automatizaci opakujících se úkolů, neustálé zlepšování procesů a produktů a využívání zpětné vazby a monitorování ke zvýšení výkonnosti a spolehlivosti systému.

Spolupráce a komunikace

Důležitost týmové spolupráce mezi vývojovými a provozními odděleními - V modelu DevOps vývojové a provozní týmy úzce spolupracují v průběhu celého životního cyklu vývoje softwaru. Cílem této spolupráce je odbourat tradiční sila, omezit úzká místa a zrychlit termíny dodání. Podporou kultury otevřenosti a sdílené odpovědnosti mohou organizace zajistit hladší přechod z vývoje do produkce a soudržnější pracovní postupy.

Nástroje a postupy pro zlepšení spolupráce - Nástroje, jako jsou systémy pro správu verzí (např. Git), platformy pro kontinuální integraci a kontinuální nasazení (CI/CD) (např. Jenkins, GitLab CI/CD) a komunikační platformy v reálném čase (např. Slack, Microsoft Teams), usnadňují lepší spolupráci. Transparentní a inkluzivní pracovní prostředí podporují také postupy, jako jsou pravidelné stand-up schůzky, párové programování a sdílená dokumentace.

Automatizace

Úloha automatizace v DevOps - Automatizace je základním kamenem filozofie DevOps. Díky automatizaci opakujících se a manuálních úkolů se týmy mohou soustředit na strategičtější a kreativnější práci. Automatizace v DevOps obvykle zahrnuje nasazování kódu, testování, konfiguraci a monitorování, což pomáhá omezit lidské chyby, zlepšit konzistenci a zrychlit procesy.

Přínosy automatizace opakujících se úloh - Automatizace vede k předvídatelnějším výsledkům, snižuje pravděpodobnost chyb a umožňuje rychlejší zpětnou vazbu a cykly řešení. Podporuje také škálovatelnost tím, že usnadňuje správu složitých systémů a nasazení, což v konečném důsledku vede k efektivnějšímu a spolehlivějšímu poskytování služeb.

Neustálé zlepšování

Přijímání změn a neustálé učení - Neustálé zlepšování v DevOps znamená neustálé hledání způsobů, jak systémy, nástroje a procesy vylepšovat. Zahrnuje přijetí růstového myšlení, kdy se zpětná vazba používá jako nástroj pro pozitivní změny a neúspěchy jsou vnímány jako příležitost k učení a růstu.

Zavedení smyček zpětné vazby pro neustálé zlepšování - Smyčky zpětné vazby jsou mechanismy, které usnadňují neustálé zlepšování tím, že umožňují týmům přemýšlet o výsledcích svých činností a činit informovaná rozhodnutí. V systému DevOps může zpětná vazba pocházet z různých zdrojů, včetně recenzí kódu, zpětné vazby od uživatelů, výkonnostních ukazatelů a hodnocení po incidentech. Začlenění smyček zpětné vazby do vývojového cyklu zajišťuje, že tým přizpůsobuje a vyvíjí své postupy na základě reálných dat a zkušeností.

Monitorování a zpětná vazba

Důležitost monitorování v DevOps - Monitorování v DevOps přesahuje tradiční systémové metriky a zahrnuje výkonnost aplikací, zkušenosti uživatelů a obchodní výsledky. Efektivní monitorování poskytuje přehled o stavu a výkonnosti systémů a pomáhá týmům předvídat problémy dříve, než ovlivní uživatele, a přijímat rozhodnutí založená na datech.

Využití zpětné vazby k proaktivnímu zlepšování - Zpětná vazba z monitorovacích nástrojů by měla být systematicky integrována do procesu vývoje. To může zahrnovat nastavení upozornění na anomálie, provádění analýz příčin opakujících se problémů a využívání poznatků získaných z monitorování k určování priorit vývoje a provozních zlepšení. Aktivní reakcí na zpětnou vazbu mohou týmy zvýšit spolehlivost systému a spokojenost uživatelů.

Naposledy změněno: čtvrtek, 21. března 2024, 09.00

◄ Praktické seznámení s verzovacím nástrojem Git

DevOps

Praktické seznámení s Docker (kontejnerová virtualizace) - spuštění kontejneru

Po nainstalování Docker lze začít experimentovat s jeho příkazy a možnostmi. Pro začátek bude stažen kontejner "busybox", což je minimalistická linuxová distribuce obsahující pouze základní nástroje. Pomocí něj se seznámíme se základy Dockeru. Příkazem:

$ docker pull busybox

Příkaz pull stáhne obraz busyboxu z registru Docker a uloží jej do našeho systému. Příkazem docker images můžete zobrazit seznam všech obrazů v systému.

$ docker images

```
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
busybox latest c51f86c28340 před 4 týdny 1,109 MB
```

Spusťme nyní kontejner Docker založený na tomto obrazu. K tomu použijeme příkaz docker run.

$ docker run busybox

```
$
```

Výsledek vypadá, jako že se vlastně nic nestalo. Nicméně kontejner byl skutečně spuštěn. Busybox nicméně definuje jako spouštěný proces "prázdný příkaz", který se rychle vykoná a skončí. Jeho ukončením je rovněž ukončen životní cyklus kontejneru. Pomocí docker run je možné ovlivnit příkaz, který bude v kontejneru spuštěn:

$ docker run busybox echo "hello from busybox"

```
hello from busybox
```

Nyní došlo k vypsání textového výstupu, stále platí, že tento příkaz byl velmi rychle proveden a po jeho skončení byl ukončen i samotný kontejner busyboxu. Běžící kontejnery je možné vypsat pomocí docker ps:

$ docker ps

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

Protože žádný kontejner neběží, vidíme prázdný řádek. Variant: docker ps -a; vypisuje všechny existující kontejnery

$ docker ps -a

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
305297d7a235 busybox "uptime" před 11 minutami Ukončeno (0) před 11 minutami distracted_goldstine
ff0a5c3750b9 busybox "sh" před 12 minutami Ukončeno (0) před 12 minutami elated_ramanujan
14e5bd11d164 hello-world "/hello" před 2 minutami Ukončeno (0) před 2 minutami thirsty_euclid
```

Výše uvedený výstup obsahuje dříve spuštěné kontejnery, které sice nyní neběží, ale jsou stále přítomny v paměti/souborovém systému dockeru. Nad libovolným kontejnerem je možné spustit i interaktivní session a experimentovat tak uvnitř běžícího kontejneru:

$ docker run -it busybox sh

/ # ls

```
bin dev etc home proc root sys tmp usr var
```

/ # uptime

```
 05:45:21 up 5:58, 0 uživatelů, průměrná zátěž: 0.00, 0.01, 0.04
```

Všechny následující příkazy jsou již spouštěny uvnitř kontejneru. Uvnitř je vidět virtuální souborový systém, který je v základu izolovaný od toho, kde běží docker. Po ukončení session dojde opět k ukončení kontejneru.

Nyní je možné již nepoužívané kontejnery smazat z diskového úložiště:

$ docker rm 305297d7a235 ff0a5c3750b9

```
305297d7a235
ff0a5c3750b9
```

V příkazu je možné uvést ID kontejneru (nebo více ID kontejnerů) nebo jejich názvy.

Jelikož předchozí kontejnery byly zcela izolovány od vnějšího prostředí, nabízí příkaz docker run řadu přepínačů a voleb, které umožňují nastavit provázání s reálným diskovým a síťovým prostředím. Jedny z možností jak zpřístupnit kontejner je například uvedena u základního obrazu webového serveru Apache:

$ docker run -dit --name my-apache-app -p 8080:80 -v "$PWD":/usr/local/apache2/htdocs/ httpd:2.4

Parametr -d je zkratkou pro detach, kontejner je po spuštění přesunut na pozadí a ponechán spuštěný. Parametr -p (port) provádí mapování portů, v tomto případě port 8080 hostitelského systému napojí na port 80 uvnitř kontejneru. Parametr -v (volume) provádí napojení souborových systémů, v tomto případě cestu "$PWD", kde $PWD je Linuxová proměnná prostředí, která obsahuje aktuální pracovní adresář (je zde možné uvést cestu přímo), napojí do cesty /usr/local/apache2/htdocs v kontejneru. Stačí tak v aktuální pracovní složce vytvořit soubor index.html s obsahem "Hello World", spustit uvedený příkaz a následně v hostitelském systému spustit webový prohlížeč a otevřít adresu http://localhost:8080, prohlížeč by měl zobrazit "Hello World" poskytnuté pomocí webserveru Apache běžícím v kontejneru.

Naposledy změněno: pondělí, 25. března 2024, 16.23

◄ Průběžné testování znalostí - DevOps principy

Přejít na...

Tutoriál na docker-curriculum.com ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat

Stáhněte si mobilní aplikaci

DevOps

DevOps a cloud

Úvod do cloud computingu

Cloud computing představuje poskytování výpočetních služeb - serverů, úložišť, databází, sítí, softwaru, analytiky a inteligence - prostřednictvím internetu ("cloudu"), které nabízí rychlejší inovace, flexibilní zdroje a úspory z rozsahu. V dnešním digitálním věku se cloud computing stal základem pro podniky, které se snaží zajistit efektivitu, škálovatelnost a nákladovou efektivitu svých IT operací. Umožňuje firmám vyhnout se počátečním nákladům a složitosti vlastnictví a údržby IT infrastruktury a místo toho zvolit model "pay-as-you-go".

Různé typy cloudových služeb: IaaS, PaaS, SaaS

- **Infrastruktura jako služba (IaaS):** Poskytuje virtualizované výpočetní zdroje přes internet. Nabízí základní výpočetní infrastrukturu: servery, úložiště a síťové prostředky, které uživatelům umožňují provozovat libovolný operační systém nebo aplikace. Příklady: Amazon Web Services (AWS), Microsoft Azure a Google Compute Engine.
- **Platforma jako služba (PaaS):** Nabízí běhové prostředí pro vývoj, testování a správu aplikací. PaaS je navržena tak, aby vývojářům usnadnila rychlé vytváření webových nebo mobilních aplikací bez starostí o nastavení nebo správu základní infrastruktury serverů, úložišť, sítí a databází potřebných pro vývoj. Mezi příklady patří Google App Engine, Microsoft Azure App Services a Heroku.
- **Software jako služba (SaaS):** Dodává softwarové aplikace přes internet na základě předplatného. Poskytovatelé SaaS spravují infrastrukturu a platformy, na kterých aplikace běží. Příklady: Google Workspace, Salesforce a Microsoft 365.

Synergie DevOps a cloudu

Cloud computing a DevOps jsou vzájemně propojené strategie, které zvyšují agilitu a efektivitu vývoje a nasazování softwaru. Cloud nabízí ideální prostředí pro implementaci postupů DevOps díky své flexibilitě, škálovatelnosti a dostupnosti zdrojů. V kontextu DevOps cloud poskytuje zdroje na vyžádání, což týmům umožňuje rozjíždět a škálovat infrastrukturu podle potřeby bez počátečních investic nebo zpoždění.

Synergie mezi DevOps a cloud computingem spočívá v jejich společném zaměření na zlepšení spolupráce, automatizaci a bezproblémovou integraci v celém životním cyklu vývoje. Cloudová prostředí podporují DevOps tím, že poskytují řadu automatizovaných nástrojů a služeb pro kontinuální integraci a kontinuální dodávku (CI/CD), infrastrukturu jako kód (IaC) a monitorování a analýzu, které jsou nezbytné pro efektivní implementaci DevOps.

Přínosy integrace cloudu do DevOps

Integrace cloudových služeb do postupů DevOps přináší řadu výhod, mezi něž patří:

- **Škálovatelnost a flexibilita:** Cloudová prostředí umožňují týmům DevOps škálovat zdroje nahoru nebo dolů podle poptávky, což zajišťuje optimální výkon a nákladovou efektivitu.
- **Rychlost a efektivita:** Cloudové služby umožňují rychlé poskytování zdrojů, což výrazně zkracuje dobu potřebnou k nastavení infrastruktury a prostředí. Tím se zrychlují vývojové cykly a doba uvedení na trh.
- **Zlepšení spolupráce:** Cloudové platformy usnadňují lepší spolupráci mezi vývojovými, provozními a dalšími týmy tím, že poskytují sdílená a konzistentní prostředí dostupná odkudkoli.
- **Zlepšené inovace:** Díky snížení režijních nákladů na správu infrastruktury se mohou týmy více soustředit na inovace a zlepšování funkcí aplikací.
- **Úspora nákladů:** Využitím modelu průběžného placení cloudových služeb mohou organizace výrazně snížit náklady spojené s nedostatečně využívanými zdroji.

Závěrem lze říci, že kombinace DevOps a cloud computingu mění způsob vývoje, nasazení a správy softwaru. Využitím možností cloudu mohou týmy DevOps dosáhnout vyšší efektivity, lepšího výkonu a rychlejších inovací.

Naposledy změněno: čtvrtek, 21. března 2024, 09.04

DevOps

Cloud poskytovatelé a DevOps nástroje

Cloudoví poskytovatelé a nástroje

Přehled hlavních poskytovatelů cloudu: AWS, Azure, Google Cloud

- **AWS (Amazon Web Services):** AWS je komplexní a široce rozšířená cloudová platforma, která nabízí více než 200 plně funkčních služeb z datových center po celém světě. AWS je proslulá svou všestranností a rozsáhlou nabídkou služeb a poskytuje řadu nástrojů a služeb pro výpočetní techniku, úložiště, databáze, analytiku, strojové učení a postupy DevOps.
- **Azure (Microsoft Azure):** Cloudová platforma společnosti Microsoft nabízí širokou škálu cloudových služeb, včetně řešení pro výpočetní techniku, analytiku, úložiště a sítě. Azure podporuje různé programovací jazyky, frameworky a operační systémy a poskytuje robustní služby pro DevOps, například Azure DevOps, který se bezproblémově integruje s nástroji pro vývoj softwaru společnosti Microsoft.
- **Google Cloud Platform (GCP):** GCP je známá svými vysoce výkonnými výpočetními a datovými analytickými schopnostmi a poskytuje škálovatelné a spolehlivé cloudové služby. Google Cloud klade velký důraz na umělou inteligenci a strojové učení a nabízí také různé nástroje a služby přizpůsobené pro DevOps, včetně Google Cloud Build, Container Registry a Kubernetes Engine.

Klíčové cloudové nástroje a služby používané v DevOps

- **Continuous Integration and Continuous Delivery (CI/CD):** Nástroje jako AWS CodeBuild, Azure Pipelines a Google Cloud Build automatizují proces integrace změn kódu a nasazování aplikací do produkčních prostředí.
- **Infrastruktura jako kód (IaC):** Služby jako AWS CloudFormation, Azure Resource Manager a Google Cloud Deployment Manager umožňují týmům definovat a poskytovat cloudovou infrastrukturu pomocí kódu.
- **Monitorování a protokolování:** Nástroje jako Amazon CloudWatch, Azure Monitor a Google Operations (dříve Stackdriver) poskytují monitorování, protokolování a diagnostiku aplikací a infrastruktury v cloudu.
- **Správa konfigurace a automatizace:** Řešení jako AWS OpsWorks, Azure Automation a Google Cloud Operations Suite (pro monitorování, protokolování a diagnostiku) pomáhají spravovat konfigurace serverů, provozovat aplikace a automatizovat opakující se úlohy.

Řešení DevOps založená na cloudu

Cloudová řešení DevOps nabízejí integrovaná prostředí pro správu celého životního cyklu vývoje softwaru, od kódování a testování až po nasazení a monitorování. Tato řešení usnadňují spolupráci mezi týmy, zefektivňují pracovní postupy a zvyšují efektivitu. Využitím cloudových CI/CD pipelines mohou týmy automatizovat vytváření, testování a nasazování aplikací, čímž se zrychlí doba uvedení na trh a zajistí konzistence a kvalita.

Případové studie pracovních postupů DevOps poháněných cloudem:

1. **Komerční společnost škálující pomocí AWS:** Případová studie o tom, jak společnost zabývající se elektronickým obchodem využila služby AWS jako AWS CodeDeploy, AWS Lambda a Amazon ECS k automatizaci procesů nasazení, správě bezserverových funkcí a škálování kontejnerových aplikací, což vedlo ke zkrácení doby nasazení a zvýšení škálovatelnosti.
2. **Firma poskytující finanční služby inovuje na platformě Azure:** Tato případová studie zkoumá, jak firma poskytující finanční služby přijala Azure DevOps (včetně Azure Pipelines a Azure Repos) a Azure Kubernetes Service (AKS), aby zefektivnila svůj proces dodávání softwaru, zlepšila bezpečnost a zachovala soulad s předpisy.
3. **Transformace mediální společnosti pomocí Google Cloud:** Příklad mediální společnosti, která využila Google Kubernetes Engine (GKE) a Google Cloud Build k modernizaci životního cyklu vývoje aplikací, snížení nákladů na infrastrukturu a zvýšení frekvence nasazování.

Naposledy změněno: čtvrtek, 21. března 2024, 09.08

◄ DevOps a cloud

Přejít na...

Průběžné testování znalostí - DevOps a cloud ►

Nápověda a dokumentace (anglicky)

Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
- Čeština (cs)
- English (en)

Souhrn uchovávaných dat

Stáhněte si mobilní aplikaci

DevOps

Dohled a odpovědnosti

Týmová dynamika a spolupráce

DevOps zásadně mění způsob interakce a spolupráce IT týmů. Odbouráním tradičních sil mezi vývojem, provozem a zajištěním kvality podporuje DevOps kulturu sdílené odpovědnosti a neustálého zlepšování. Tato integrovaná týmová struktura podporuje otevřenou komunikaci a spolupráci, což vede k inovativnějším řešením a rychlejšímu řešení problémů.

V prostředí DevOps jsou členové týmu povzbuzováni k tomu, aby překračovali hranice svých tradičních rolí. Vývojáři mohou přebírat provozní úkoly, zatímco provozní pracovníci mohou být zapojeni do procesu vývoje již od dřívější fáze. Tato mezioborová spolupráce zajišťuje, že software je navržen s ohledem na provozuschopnost a že nasazení je stabilnější a spolehlivější.

Strategie pro efektivní týmovou práci a komunikaci v prostředí DevOps

- Podpora kultury transparentnosti a otevřené komunikace: Povzbuzování členů týmu, aby otevřeně sdíleli své nápady, problémy a zpětnou vazbu. Pravidelné schůzky, nástroje pro spolupráci a otevřené komunikační kanály to mohou usnadnit.
- Důraz na neustálé učení a zlepšování: Vytvoření prostředí, kde je podporováno učení se z neúspěchů a neustálý osobní a profesní rozvoj.
- Zavádějte párové programování a křížové školení: Párování vývojářů s provozními pracovníky na projektech může zlepšit vzájemné porozumění a sdílení znalostí napříč funkcemi.
- Využívejte nástroje pro spolupráci: Využívání nástrojů, které usnadňují komunikaci a spolupráci, jako je Slack, Microsoft Teams, JIRA nebo Trello, aby byli všichni sladěni a informováni.

Vedení a řízení v oblasti DevOps

Vedení hraje klíčovou roli při řízení kulturní změny nezbytné pro DevOps. Vedoucí pracovníci musí ve svých organizacích prosazovat principy spolupráce, automatizace, neustálého zlepšování a činnosti zaměřené na zákazníka. To zahrnuje stanovení jasných očekávání, modelování žádoucího chování a podporu prostředí, kde se oceňuje experimentování a učení se z chyb.

Vedoucí pracovníci by také měli zajistit, aby týmy měly k dispozici nástroje, zdroje a školení, které potřebují k úspěchu v prostředí DevOps. To zahrnuje investice do technologií, které usnadňují automatizaci a spolupráci, a také poskytování příležitostí k profesnímu rozvoji.

Praktiky řízení podporující principy DevOps

- Podporovat autonomii a posílení pravomocí: Poskytování týmům autonomie při rozhodování a přebírání odpovědnosti za svou práci. Toto posílení pravomocí vede k vyššímu zapojení a odpovědnosti.
- Zavedení smyček zpětné vazby: Vytvoření mechanismy pro průběžnou zpětnou vazbu mezi členy týmu i mezi týmem a zákazníky. To pomáhá rychle identifikovat a řešit problémy a lépe sladit produkty s potřebami zákazníků.
- Uznávejte a odměňujte spolupráci: Ocenění a odměňování chování, které podporuje týmovou práci, sdílení znalostí a spolupráci mezi jednotlivými funkcemi.
- Přijímejte techniky štíhlého a agilního řízení: Využívání štíhlé a agilní metodiky k řízení práce a neustálému zlepšování procesů. To může zahrnovat zavádění sprintů, kanban boardů a pravidelných stand-up schůzek, abyste udrželi projekty na správné cestě a týmy v souladu.

Přetvořením týmové dynamiky a přijetím podpůrných postupů vedení a řízení může DevOps vést k efektivnějším a účinnějším pracovním postupům. Podporou kultury spolupráce, neustálého zlepšování a vzájemného respektu mohou organizace prolomit tradiční bariéry a dosáhnout větších inovací a úspěchů ve svých projektech.

Naposledy změněno: pátek, 22. března 2024, 09.52

◄ Základy orchestrace kontejnerů s využitím Kubernetes

Přejít na...

Přehled DevOps rolí ►

Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
- Čeština (cs)
- English (en)

Souhrn uchovávaných dat

Stáhněte si mobilní aplikaci

DevOps

Přehled DevOps rolí

Pochopení rolí v DevOps

DevOps není jedna role, ale kultura nebo filozofie, která zahrnuje více rolí spolupracujících na dosažení společných cílů. Pochopení těchto rolí a jejich odpovědností je pro úspěšnou implementaci postupů DevOps klíčové. Zde je přehled klíčových rolí v prostředí DevOps:

DevOps Engineer

Inženýr DevOps funguje jako most mezi týmy vývoje softwaru a provozu. Je zodpovědný za implementaci automatizačních nástrojů, nastavení CI/CD potrubí, zajištění spolehlivosti systému a zlepšení frekvence nasazování při zachování standardů bezpečnosti a shody s předpisy.

Odpovědnosti:

- Implementace a správa CI/CD pipelines.
- Automatizovat pracovní postupy a procesy nasazování.
- Sledování výkonu systému a řešení problémů.
- Spolupracovat s vývojáři a pracovníky IT a zajistit dodržování osvědčených provozních postupů.

Požadované dovednosti:

- Znalost automatizačních nástrojů (např. Jenkins, Ansible, Terraform).
- Dobrá znalost cloudových služeb (AWS, Azure, Google Cloud).
- Znalost skriptovacích jazyků (např. Python, Bash).
- Znalost kontejnerizace (např. Docker, Kubernetes).

Site Reliability Engineer (SRE)

Inženýři pro spolehlivost webu se zaměřují na vytváření škálovatelných a vysoce spolehlivých softwarových systémů. Uplatňují inženýrské principy při řešení problémů, optimalizaci výkonu systému a zajišťování bezporuchovosti služeb. Pracovníci SRE jsou zodpovědní za definování cílů úrovně služeb (SLO) a zavádění automatizace s cílem snížit množství manuální práce (dřiny).

Odpovědnosti:

- Vyvíjet a udržovat cíle úrovně služeb (SLO) a ukazatele úrovně služeb (SLI).
- Automatizovat provozní úkoly s cílem snížit pracnost.
- Zlepšovat spolehlivost a efektivitu systému prostřednictvím technických řešení.
- Provádět přezkumy po nehodách a zavádět preventivní opatření.

Požadované dovednosti:

- Znalost kódování a automatizace (např. Python, Go).
- Hluboká znalost sítí, zabezpečení a správy databází.
- Zkušenosti s nástroji pro monitorování a upozorňování (např. Prometheus, Grafana).
- Schopnost analyzovat a optimalizovat výkon systému.

Systémový administrátor

Systémoví administrátoři udržují a konfigurují IT systémy a servery. V kontextu DevOps úzce spolupracují s vývojáři, aby zajistili, že infrastruktura podporuje požadavky na software a harmonogramy nasazení. Přecházejí na infrastrukturu jako kód (IaC) a automatizaci, aby mohli systémy spravovat efektivněji.

Zodpovědnosti:

- Instalace, konfigurace a údržba serverů a sítí.
- Zajišťovat zálohování, obnovu a zabezpečení dat.
- Pomáhat při přechodu na cloudová prostředí a jejich správě.
- Spolupracovat s vývojáři na vytváření spolehlivé infrastruktury pro aplikace.

Požadované dovednosti:

- Znalost serverového hardwaru a sítí.
- Znalost nástrojů pro monitorování a zabezpečení systému.
- Zkušenosti s cloud computingem a virtualizací.
- Znalost nástrojů pro automatizaci a správu konfigurace (např. Puppet, Chef).

Kromě těchto rolí existují v týmu DevOps i další důležité pozice, například inženýři QA, kteří zajišťují, aby produkty splňovaly standardy kvality prostřednictvím automatizovaného testování, a vlastníci produktů, kteří udávají týmu směr a priority. Klíčem k úspěšným postupům DevOps je spolupráce a komunikace mezi těmito rolemi, které mají společnou odpovědnost za úspěch produktu. Pochopením specifických odpovědností a požadovaných dovedností jednotlivých rolí mohou organizace lépe implementovat a optimalizovat své strategie DevOps.

Naposledy změněno: čtvrtek, 21. března 2024, 09.13

◄ Dohled a odpovědnosti

Přejít na...

Zodpovědnost a vlastnictví ►

DevOps

Zodpovědnost a vlastnictví

Zodpovědnost a vlastnictví

V systému DevOps je odpovědnost zásadní pro zajištění toho, aby se všichni členové týmu podíleli na úspěchu projektu. Zahrnuje jasné vymezení odpovědností a stanovení očekávání pro jednotlivé role v týmu. Na rozdíl od tradičních modelů, které mohou odpovědnost svalovat na jednotlivce, však DevOps klade důraz na kolektivní odpovědnost, kdy je úspěch či neúspěch společným výsledkem.

Pro zvýšení odpovědnosti:

1. **Definice jasných rolí a odpovědností:** Zajištění, aby každý člen týmu rozuměl svým konkrétním rolím, odpovědnosti a tomu, jak přispívá k dosažení celkových cílů.
2. **Stanovit měřitelné cíle a úkoly:** Stanovení jasných, měřitelných cílů, které jsou v souladu s obchodními výsledky, abyste pomohli členům týmu soustředit se na poskytování hodnoty.
3. **Zavedení transparentních pracovních postupů:** Využívání nástrojů pro řízení projektů k udržení přehledu o úkolech, postupu a závislostech mezi členy týmu.
4. **Podporovat kulturu důvěry:** Podpora otevřené komunikace a konstruktivní zpětné vazby, kdy členové týmu mohou otevřeně diskutovat o úspěších i neúspěších bez obav z obviňování.

Koncepce sdílené odpovědnosti ve vývojovém a provozním cyklu

Sdílené vlastnictví znamená, že vývojáři, provozní pracovníci a odborníci na zajištění kvality společně vlastní software a systémy, které vytvářejí a udržují. Tento přístup založený na spolupráci zajišťuje:

- **Sdílenou odpovědnost za kvalitu:** Všichni jsou zodpovědní za kvalitu a spolehlivost softwaru, a to nejen po jeho nasazení, ale od jeho koncepce až po údržbu.
- **Spolupráce při řešení problémů:** Týmy spolupracují na řešení problémů, což vede k udržitelnějším a inovativnějším řešením.
- **Soulad s cíli:** Sdílené vlastnictví sjednocuje celý tým ke společným cílům, jako je zvýšení spokojenosti zákazníků a spolehlivosti systému.

Rozvíjející se role a neustálé vzdělávání

Rychlé tempo technologického pokroku a měnící se požadavky trhu vyžadují, aby se odborníci na DevOps neustále učili a přizpůsobovali. S tím, jak se vyvíjejí nástroje, postupy a prostředí, musí se vyvíjet i dovednosti a znalosti týmů DevOps. Neustálé vzdělávání je nezbytné nejen pro osobní kariérní rozvoj, ale také pro udržení efektivity a konkurenceschopnosti týmu a organizace.

Strategie profesního rozvoje ve vyvíjejícím se prostředí DevOps

- **Využití online vzdělávání a certifikací:** Povzbuzování členů týmu k účasti na online kurzech, webinářích a certifikacích týkajících se DevOps, cloud computingu, automatizace a dalších relevantních oblastí.
- **Zavádět pravidelná školení a semináře:** Pořádání interních školení a workshopů, aby byl tým informován o nejnovějších postupech, nástrojích a technologiích.
- **Podporovat účast na oborových konferencích a setkáních:** Účast na oborových akcích může poskytnout přehled o nových trendech a osvědčených postupech a také příležitosti k navazování kontaktů.
- **Podporovat sdílení znalostí:** Vytvoření kultury, ve které jsou členové týmu povzbuzováni ke sdílení znalostí a zkušeností prostřednictvím obědů, interních blogů nebo týmových schůzek.
- **Podporovat mezioborová školení:** Umožnění členům týmu seznámit se s různými aspekty procesu vývoje a nasazení softwaru.

Zdůrazněním odpovědnosti, podporou sdílené odpovědnosti a podporou neustálého vzdělávání mohou týmy DevOps zlepšit spolupráci, zvýšit efektivitu a udržet si náskok v rychle se měnícím světě technologií. Tyto zásady pomáhají vytvořit proaktivní, inovativní a odolnou organizační kulturu, která dokáže zvládnout výzvy a příležitosti moderních IT prostředí.

Naposledy změněno: pátek, 22. března 2024, 09.55

◄ Přehled DevOps rolí

Přejít na...

DevOps

Plánování, design

Plánování a návrh v DevOps

Efektivní plánování v DevOps není jen o technických detailech, ale také o sladění projektů s cíli organizace a zajištění toho, aby každá akce přispívala k celkové strategii. Strategické plánování v DevOps zahrnuje pochopení současného stavu IT prostředí, identifikaci oblastí pro zlepšení a stanovení jasných, realizovatelných cílů. Vyžaduje holistický pohled, který zohledňuje nejen životní cyklus vývoje a nasazení softwaru, ale také širší obchodní kontext.

Zavedení strategického plánování v DevOps:

1. **Provedení analýzy současného stavu:** Prověření stávající vývojové, nasazovací a provozní procesy a identifikace úzkých míst, neefektivity a rizikových oblastí.
2. **Definice jasných cílů a metrik:** Stanovení konkrétních, měřitelných, dosažitelných, relevantních a časově omezených cílů (SMART), které budou v souladu s širšími obchodními cíli.
3. **Vytvoření plánu:** Podrobný plán, který nastíní kroky potřebné k dosažení těchto cílů, včetně časového harmonogramu, zdrojů a odpovědností.
4. **Zapojení zainteresovaných stran:** Zajištění, aby byly do procesu plánování zapojeny všechny relevantní strany, včetně vývojových, provozních, bezpečnostních a obchodních zainteresovaných stran, a podpořte tak spolupráci a soulad.

Sladění plánů DevOps s obchodními cíli a strategií IT - skutečná hodnota DevOps vychází z jeho schopnosti podporovat obchodní cíle a posilovat celkovou strategii IT. To vyžaduje hluboké porozumění potřebám, výzvám a příležitostem podniku a tomu, jak je DevOps může řešit. Sladění plánů DevOps s obchodními cíli:

1. **Spolupráce s vedoucími pracovníky podniku:** Pravidelná komunikace se zúčastněnými stranami z byznysu, abyste bylo porozuměno jejich cílům, prioritám a problémovým místům.
2. **Propojení aktivit DevOps s obchodními výsledky:** Demonstrace jak konkrétní iniciativy DevOps, například zlepšení frekvence nasazování nebo snížení míry selhání změn, povedou k lepším obchodním výsledkům.
3. **Prioritizace na základě dopadu na podnikání:** Zaměření se na zlepšení DevOps, která budou mít nejvýznamnější dopad na obchodní cíle, jako je zvýšení spokojenosti zákazníků nebo zrychlení doby uvedení na trh.
4. **Měření a informování o úspěchu:** Pomocí klíčových ukazatelů výkonnosti (KPI) sledování dopadu postupů DevOps na obchodní výsledky a pravidelně tyto výsledky sdílejte se zúčastněnými stranami.

Zaměřením se na základní prvky plánování DevOps a zajištěním souladu s obchodními cíli mohou organizace zajistit, aby jejich iniciativy DevOps byly strategické, cílené a efektivní. Toto sladění nejen zvyšuje efektivitu a kvalitu provozu IT, ale také přináší hmatatelné obchodní výsledky, které prokazují hodnotu DevOps při dosahování organizačních cílů.

Naposledy změněno: pátek, 22. března 2024, 09.59

◄ Úvod do konfiguračního managementu s využitím Ansible

Přejít na...

Návrh DevOps pipeline ►

DevOps

Návrh DevOps pipeline

Návrh DevOps pipeline

DevOps pipeline je soubor automatizovaných procesů a nástrojů, které umožňují vývojovým a provozním týmům spolupracovat a automatizovat životní cyklus vývoje softwaru. Klíčové fáze obvykle zahrnují:

1. **Správa zdrojového kódu (SCM):** Zde je kód řízen a ukládán ve verzích, pomocí nástrojů, jako je Git, Mercurial, Bazaar aj.. Slouží jako výchozí bod pro změny vstupující do pipeline.
2. **Kontinuální integrace (CI):** Zahrnuje automatické sestavování a testování kódu při každé změně, která je odevzdána do úložiště zdrojových kódů. To pomáhá včas identifikovat a opravit chyby, čímž se zlepšuje kvalita.
3. **Kontinuální dodání (CD):** Rozšiřuje CI o automatické nasazení všech změn kódu do testovacího nebo stagingového prostředí po fázi sestavení. Umožňuje týmům zajistit, aby se kód v prostředí podobném produkčnímu choval podle očekávání.
4. **Kontinuální nasazení:** Jde o krok dále než CD tím, že automaticky nasazuje každou změnu, která projde skrze pipeline, do produkce, což umožňuje aktualizace pro uživatele v reálném čase.
5. **Monitorování a zpětná vazba:** Zahrnuje sledování výkonu aplikací a infrastruktury, což týmům umožňuje rychle reagovat na problémy a zpětnou vazbu.

Návrh účinné a efektivní pipeline

Při navrhování pipeline je třeba zvážit následující:

- **Škálovatelnost:** Pipeline by měla zvládnout zvýšené pracovní zatížení a velikost týmu.
- **Zabezpečení:** Integrace bezpečnostních opatření v každé fázi (známé také jako DevSecOps).
- **Znovupoužitelnost:** Navržení jednotlivých fází a úloh tak, aby je bylo možné opakovaně používat v různých projektech nebo pipeline.
- **Udržovatelnost:** Zajištění, aby se pipeline dala snadno aktualizovat a udržovat.
- **Smyčky zpětné vazby:** Implementace mechanismů okamžité zpětné vazby pro rychlou identifikaci a řešení problémů.

Integrace různých nástrojů do uceleného pracovního postupu

Integrace je klíčem k úspěšnému DevOps pipeline a zajišťuje, že jednotlivé nástroje spolu komunikují a bezproblémově spolupracují s ostatními. Strategie pro efektivní integraci zahrnují:

- **API a Webhooks:** Využití rozhraní API a webových háčků (webhook) pro komunikaci mezi nástroji, což umožňuje sdílení dat a akcí v reálném čase.
- **Univerzální balíčkové formáty:** Použití univerzálních balíčkovacích formátů (například kontejnery Docker), které lze snadno přenášet a spouštět v různých fázích a prostředích.
- **Automatizační skripty:** Skripty pro automatizaci interakcí mezi nástroji, čímž se omezí ruční předávání a chyby.
- **Jediný zdroj pravdy:** Vytvoření jediného zdroje pravdy, například systém správy verzí, kde jsou uloženy a verzovány všechny konfigurace pipeline a kód softwarové aplikace.

Naposledy změněno: pátek, 22. března 2024, 10.01

DevOps

Vývoj, nasazení

Přehled vývojových postupů v prostředí DevOps

V prostředí DevOps jsou vývojové postupy navrženy tak, aby zlepšily spolupráci mezi vývojovými a provozními týmy, zvýšily efektivitu a urychlily dodávku softwaru. Mezi klíčové postupy patří:

- **Agile Development:** Přijetí agilních metodik pro řízení vývoje s flexibilitou, krátkými iteracemi a smyčkami zpětné vazby.
- **Kultura spolupráce:** Podpora kultury, v níž vývojáři, provozní pracovníci a pracovníci zajišťující kvalitu úzce spolupracují v průběhu celého životního cyklu vývoje softwaru.
- **Automatizace:** Automatizace opakujících se úkolů s cílem snížit počet chyb a uvolnit čas vývojářů pro cennější činnosti.
- **Testování:** Integrace testování do procesu vývoje, aby se zajistilo včasné odhalení a řešení problémů.

Důraz na používání řízení verzí a větvení funkcí. Řízení verzí je v DevOps zásadní, protože umožňuje více vývojářům pracovat na stejné kódové základně současně bez konfliktů. Poskytuje historii změn a možnost vrátit se v případě potřeby k předchozím verzím. Větvení funkcí rozšiřuje řízení verzí tím, že umožňuje vývojářům vytvářet větve pro konkrétní funkce nebo opravy. Tento přístup:

- Udržuje hlavní větev stabilní.
- Umožňuje izolovaný vývoj a testování nových funkcí.
- Usnadňuje vzájemné hodnocení a slučování kódu prostřednictvím požadavků na stažení.

Principy CI/CD v DevOps

Kontinuální integrace (CI) a kontinuální dodání/nasazení (CD) jsou základními postupy v DevOps:

- **Kontinuální integrace (CI):** Zahrnuje automatickou integraci změn kódu od více přispěvatelů do hlavního projektu, a to několikrát denně. Cílem je rychle odhalit a opravit chyby integrace, zlepšit kvalitu softwaru a zkrátit dobu potřebnou k ověření a vydání nových aktualizací softwaru.
- **Kontinuální dodání/nasazení (CD):** Rozšiřuje CI o automatické nasazení všech změn kódu do testovacího nebo stagingového prostředí po fázi sestavení a následně do produkčního prostředí po úspěšných testech a revizích. Cílem je, aby se nasazení stalo předvídatelnou, rutinní záležitostí, kterou lze provést kdykoli.

Nastavení a údržba pipeline CI/CD

Pipeline CI/CD automatizuje kroky od integrace kódu po nasazení. Nastavení a údržba pipeline CI/CD zahrnuje:

- **Automatizace procesu sestavování:** Každá revize kódu spustí automatizovanou sekvenci sestavení a testování, která zajistí, že se nové změny dobře integrují se stávající kódovou základnou.
- **Automatizace testování:** Začlenění automatizovaných jednotkových, integračních a end-to-end testů pro ověření funkčnosti a výkonu.
- **Automatické nasazení:** Používání nástrojů a skriptů k automatizaci procesu nasazení, čímž se zajistí, že software lze kdykoli spolehlivě vydat.
- **Monitorování a zpětná vazba:** Zavedení monitorování pro sledování výkonu aplikace a zpětné vazby od uživatelů, přičemž tyto informace jsou zpětně předávány do procesu vývoje za účelem neustálého zlepšování.

Při navrhování pipeline CI/CD je třeba brát v úvahu:

- **Škálovatelnost:** Zajištění, aby pipelina zvládla zvýšenou zátěž a velikost týmu.
- **Zabezpečení:** Do pipeline je třeba zabudovat bezpečnostní kontroly a skenování.
- **Udržovatelnost:** Je třeba udržovat kód a procesy pipeline dobře zdokumentované a kontrolované podle verzí.
- **Flexibilita:** V případě potřeby by měl být umožněny ruční zásahy a vrácení zpět.

Integrací těchto postupů vývoje a nasazení mohou týmy DevOps zlepšit spolupráci, zefektivnit pracovní postupy a urychlit dodávky, a to vše při zachování vysokých standardů kvality a spolehlivosti. Tento integrovaný přístup nejen zvyšuje efektivitu vývojového procesu, ale také úzce souvisí s obchodními cíli, což vede k efektivnějším a úspěšnějším výsledkům.

Naposledy změněno: pátek, 22. března 2024, 09.05

◄ Tutoriál dockerizace Java aplikace

Přejít na...

Infrastructure-as-a-Code ►

DevOps

Infrastructure-as-a-Code

Infrastruktura jako kód (IaC)

Infrastruktura jako kód (IaC) je klíčový postup DevOps, který zahrnuje správu a poskytování výpočetní infrastruktury prostřednictvím strojově čitelných definičních souborů, nikoliv prostřednictvím fyzické konfigurace hardwaru nebo interaktivních konfiguračních nástrojů. IaC mění způsob, jakým se infrastruktura nastavuje a udržuje, čímž je rychlejší, efektivnější a méně náchylná k lidským chybám.

V DevOps hraje IaC klíčovou roli tím, že:

- **Usnadňuje automatizaci:** IaC umožňuje automatizovat celé nastavení infrastruktury, což usnadňuje rychlé a konzistentní poskytování, škálování a správu.
- **Zlepšuje spolupráci:** Protože konfigurace infrastruktury jsou definovány v kódu, lze je kontrolovat podle verzí, revidovat a spolupracovat na nich stejně jako na kódu aplikace.
- **Zvýšení konzistence a spolehlivosti:** IaC zajišťuje, že každé prostředí je zajištěno přesně stejným způsobem, což snižuje rozdíly mezi vývojovým, testovacím a produkčním prostředím.
- **Zvýšení efektivity:** Týmy mohou nasazovat infrastrukturu podle potřeby bez nutnosti ručního nastavování, což výrazně snižuje časovou náročnost a potřebné zdroje.

Nástroje a postupy pro implementaci IaC

Pro usnadnění postupů IaC bylo vyvinuto několik nástrojů, z nichž každý má vlastní syntaxi a možnosti. Mezi nejoblíbenější nástroje IaC patří např:

- **Terraform:** Terraform je open-source nástroj vytvořený společností HashiCorp a umožňuje uživatelům definovat infrastrukturu pomocí vysokoúrovňového konfiguračního jazyka. Je nezávislý na cloudu a dokáže spravovat více poskytovatelů služeb současně. Kromě Terraform je rovněž podobným způsobem využíván alternativní nástroj Pulumi.
- **Ansible:** Nástroj s otevřeným zdrojovým kódem pro automatizaci správy konfigurace, nasazování aplikací a automatizaci úloh. Ansible používá pro své playbooky (skripty IaC) syntaxi YAML. Mezi další často využívané nástroje automatizace správy konfigurace patří také Chef, či Puppet.
- **AWS CloudFormation:** Služba poskytovaná společností Amazon Web Services, která uživatelům poskytuje snadný způsob vytváření a správy kolekce souvisejících prostředků AWS, jejich řádného a předvídatelného poskytování a aktualizace.
- **Azure Resource Manager (ARM):** Poskytuje vrstvu pro správu, která umožňuje vytvářet, aktualizovat a odstraňovat prostředky v účtu Azure. Pro nasazení používáte šablonu, což je soubor JSON definující prostředky, které v Azure potřebujete.

Integrací infrastruktury jako kódu do procesů DevOps mohou organizace dosáhnout agilnější, flexibilnější a spolehlivější správy infrastruktury. Tím se nejen sníží možnost lidské chyby, ale také se výrazně zvýší rychlost a efektivita nasazování a správy infrastruktury, čímž se správa IT infrastruktury více sladí s vývojem aplikací.

Naposledy změněno: pátek, 22. března 2024, 09.09

◄ Vývoj, nasazení

Přejít na...

Terraform - populární IaC řešení ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka

Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat
Stáhněte si mobilní aplikaci

DevOps

Strategie nasazení

Strategie nasazení

Strategie nasazení jsou klíčové pro řízení uvolňování nových verzí softwaru do produkčních prostředí. Mohou minimalizovat prostoje a dopad chyb na koncové uživatele. Zde jsou uvedeny některé běžné strategie nasazení používané v praxi DevOps:

Modrozelené nasazení (Blue-Green deployment)

Při modrozeleném nasazení jsou udržována dvě identická prostředí. Prostředí "Blue" je aktuální produkční prostředí, zatímco prostředí "Green" je klon s novou verzí softwaru. Jakmile je prostředí "Green" plně otestováno a připraveno, provoz se přepne z prostředí "Blue" do prostředí "Green", čímž se zelené prostředí stane novým produkčním prostředím. Tato strategie umožňuje rychlý rollback a minimální prostoje, ale vyžaduje dvojnásobnou infrastrukturu.

Nasazení kanárků (Canary deployment)

Nasazení Canary zahrnuje uvolnění nové verze softwaru pro malou podskupinu uživatelů předtím, než se rozšíří na celou uživatelskou základnu. Tento přístup umožňuje týmům sledovat výkon a stabilitu nové verze a v případě problémů ji vrátit zpět, čímž se snižuje riziko pro celou uživatelskou základnu. Procento uživatelů vystavených nové verzi se postupně zvyšuje v závislosti na úspěšnosti nasazení.

Průběžné aktualizace: (Rolling deployment)

Rolling updates postupně nahrazují instance staré verze softwaru novou verzí. Obvykle se tak děje bez vyřazení systému z provozu, čímž je zajištěno, že nedojde k výpadku. Tato metoda však vyžaduje, aby systém během procesu nasazení zpracovával různé verze, což může být složité v závislosti na změnách v nové verzi.

Volba vhodných strategií pro různé scénáře

- **Změny s vysokým rizikem:** Pro vysoce rizikové aktualizace může být ideální modrozelené nasazení, protože umožňuje okamžité vrácení zpět, pokud nová verze selže. Tato strategie je výhodná v případě, že změny mají významný potenciální dopad na podnikové operace.
- **Ověřování funkcí:** Kanárková nasazení jsou užitečná zejména tehdy, když je třeba ověřit dopad nových funkcí u skutečných uživatelů v produkčním prostředí. Tento přístup je vhodný pro postupné zavádění funkcí nebo A/B testování.
- **Aktualizace služeb:** Průběžné aktualizace jsou vhodné pro služby, které vyžadují nepřetržitou dostupnost a kde jsou změny přírůstkové a je nepravděpodobné, že by způsobily větší narušení. Tato metoda se běžně používá pro aktualizace, které nezahrnují významné změny datových schémat nebo smluv o poskytování služeb.

Při výběru strategie nasazení je třeba zvážit faktory, jako je složitost aplikace, kritičnost služby, dostupnost zdrojů a tolerance organizace k riziku a výpadkům. Zvolená strategie také musí být v souladu s provozními možnostmi týmu a celkovými obchodními cíli.

Pochopením a výběrem vhodných strategií nasazení mohou týmy DevOps zajistit hladší, bezpečnější a kontrolovanější uvolňování softwaru. To nejen zvyšuje spolehlivost a dostupnost služeb, ale také zlepšuje uživatelskou zkušenost a důvěru v aplikaci.

Naposledy změněno: pátek, 22. března 2024, 09.12

◄ Pulumi - alternativní moderní IaC řešení

Přejít na...

Průběžné testování znalostí - Vývoj, nasazení ►

DevOps

Release management

Release management

Správa vydání je kritickým aspektem vývoje softwaru a provozu IT, který se zaměřuje na plánování, rozvrhování a řízení sestavení softwaru v různých fázích a prostředích. Zahrnuje testování a nasazování verzí. Tento koncept je základním kamenem jak tradičních rámců IT, jako je ITIL (Information Technology Infrastructure Library), tak moderních postupů, jako je DevOps. Pochopení toho, jak správa verzí zapadá do ITIL a DevOps, může poskytnout komplexní pohled na to, jak lze efektivně a spolehlivě dodávat software.

Správa vydání v ITIL

V rámci ITIL je Release Management součástí fáze Service Transition, která se zaměřuje na zajištění efektivního poskytování IT služeb podniku. ITIL definuje release jako soubor autorizovaných změn služby IT. Hlavním cílem je chránit živé prostředí a zajistit, aby byly uvolněny správné komponenty.

Klíčové koncepty:

- **Politika release:** Popisuje typy release a standardy pro jejich nasazení.
- **Balíček releasů:** Jedna jednotka releasu, která zahrnuje všechny komponenty potřebné pro nasazení.
- **Deployment:** Vlastní přesun vydání do živého prostředí.

Proces

1. **Plánování:** Definování rozsahu, harmonogramu a potřebných zdrojů.
2. **Sestavení:** Sestavení a sestavení vydání z různých komponent.
3. **Testování:** Ověřte, zda verze splňuje obchodní a technické požadavky.
4. **Nasazení:** Přesun vydání do produkčního prostředí.
5. **Přezkoumání a uzavření:** Vyhodnocení úspěšnosti vydání a identifikace oblastí pro zlepšení.

Správa verzí v systému DevOps

DevOps urychluje proces správy vydání tím, že podporuje spolupráci mezi vývojovými a provozními týmy. Na rozdíl od ITIL, která může být preskriptivní, je DevOps flexibilnější a zaměřuje se na neustálé zlepšování a automatizaci.

Klíčové koncepty:

- **Kontinuální integrace (CI):** Vývojáři často slučují změny kódu do sdíleného úložiště, což automaticky spouští sestavení a testy.
- **Kontinuální dodání (CD):** Automaticky připravuje změny kódu k vydání a zajišťuje, že software může být kdykoli nasazen.
- **Kontinuální nasazení:** Rozšiřuje CD o automatické nasazení každé změny, která projde potrubím, do produkce.

Proces

1. **Vývoj:** Kód je vyvíjen v malých přírůstcích a průběžně integrován.
2. **Sestavení:** Automatizované nástroje zkompilují a zabalí software.
3. **Testování:** Provádějí se automatizované testy, které ověřují funkčnost, výkon a zabezpečení.
4. **Nasazení:** Změny jsou automaticky nasazeny do produkčních prostředí, často prostřednictvím modrozelených nebo kanárkových nasazení.
5. **Monitorování:** Ke sledování výkonu a stavu aplikace v produkčním prostředí se používají nástroje pro průběžné monitorování.

Integrace ITIL a DevOps při správě verzí.

Ačkoli se může zdát, že ITIL a DevOps jsou v rozporu kvůli svému odlišnému původu a zaměření, jejich integrace může vést k vyváženému přístupu ke správě vydání:

- **Přijmout myšlení založené na službách:** Sladit vývoj i provoz s poskytováním hodnoty prostřednictvím IT služeb.
- **Automatizovat a zlepšovat:** Využívejte nástroje a postupy DevOps v rámci ITIL k automatizaci opakujících se úkolů a zefektivnění procesů.
- **Soustředit se na hodnotu:** Stanovovat priority vydání na základě obchodní hodnoty a dopadu na zákazníka.
- **Učte se a přizpůsobte:** Zapojte smyčky zpětné vazby z monitorování a kontrol po vydání s cílem neustále zlepšovat procesy vydání.

Pochopení správy vydání v rámci ITIL i DevOps poskytuje komplexní přístup k poskytování softwaru a služeb. Kombinací strukturovaného přístupu ITIL s agilitou a automatizací DevOps mohou organizace dosáhnout efektivnějších, spolehlivějších a konzistentnějších procesů releasů. Tato integrace nejen optimalizuje provoz IT, ale také maximalizuje hodnotu pro podnikání.

Naposledy změněno: pátek, 22. března 2024, 09.20

◄ Tutoriál k blue/green deployment modelu v Kubernetes

Přejít na...

Průběžné testování znalostí - Release management ►

DevOps

Monitorování, měření, reportování

Monitorování, měření a reportování v DevOps

Monitorování a měření jsou nedílnou součástí procesu DevOps a poskytují přehled o výkonu, stavu a efektivitě aplikací a infrastruktury. Umožňují týmům rychle odhalovat a řešit problémy, zvyšovat spolehlivost služeb a přijímat informovaná rozhodnutí na základě dat v reálném čase.

Důležitost monitorování a měření v systému DevOps:

- **Optimalizace výkonu:** Identifikace a řešení úzkých míst výkonu.
- **Proaktivní řešení problémů:** Odhalení a řešení problémů dříve, než ovlivní uživatele.
- **Zpětná vazba pro neustálé zlepšování:** Poskytování poznatků založených na datech pro zdokonalování procesů a zlepšování kvality produktů.
- **Transparentnost a viditelnost:** Nabídněte všem zúčastněným stranám jasný přehled o stavu a výkonnosti systému.

Klíčové metriky a ukazatele ke sledování:

- **Metriky výkonnosti:** Doba odezvy, latence, propustnost a chybovost.
- **Stav systému:** Využití procesoru, paměti, diskových vstupů/výstupů a šířky pásma sítě.
- **Obchodní metriky:** Míra konverze, zapojení uživatelů a spokojenost zákazníků.
- **Metriky nasazení:** Frekvence nasazení, doba realizace změn, míra neúspěšnosti změn a střední doba do obnovy (MTTR).

Nástroje a techniky pro monitorování

Přehled oblíbených nástrojů pro monitorování:

- **Prometheus:** Sada nástrojů pro monitorování a upozorňování systémů s otevřeným zdrojovým kódem, která je známá svým výkonným dotazovacím jazykem a integrací s nástrojem Grafana pro vizualizaci. Nástroj umožňuje aktivně sledovat a zaznamenávat stav (vzdálených) systémů a poskytuje vysokou míru rozšiřitelnosti pomocí jednoduchého API pro realizaci vlastního monitorovacího agenta.
- **Grafana:** Víceplatformní open-source analytická a interaktivní vizualizační webová aplikace. Po připojení k podporovaným zdrojům dat poskytuje grafy, diagramy a upozornění dostupných skrze webový prohlížeč.
- **Nagios:** Široce používaný open-source monitorovací systém, který umožňuje organizacím identifikovat a řešit problémy IT infrastruktury dříve, než ovlivní kritické podnikové procesy. Nástroj primárně slouží pro sledování dostupnosti a stavu jednotlivých síťových uzlů. Umožňuje vytvoření virtuální mapy počítačové sítě, což mu umožňuje, že automaticky pak rozpozná při chybě síťového zařízení, že není třeba rovněž upozorňovat na další zařízení, která byla původně dostupná skrze nyní nedostupné zařízení.
- **Elastic Stack (ELK):** Skládá se z nástrojů Elasticsearch, Logstash a Kibana a poskytuje výkonnou platformu pro vyhledávání, analýzu a vizualizaci dat protokolů. Systém umožňuje centralizaci a efektivní správu logů z rozsáhlého (distribuovaného) systému.

Zavádění účinných strategií monitorování:

- **Definice klíčových ukazatelů výkonnosti (KPI):** Je třeba určit metriky, které jsou pro vaši aplikaci a obchodní cíle nejdůležitější.
- **Konfigurace výstrah a oznámení:** Zahrnuje nastavení výstrah pro klíčové metriky, abyste došlo k automatickému upozornění příslušných týmů na potenciální problémy nebo zhoršení výkonu.
- **Správa protokolů:** Implementace komplexní správy protokolů pro zachycení, ukládání a analýzu dat protokolů pro získání dalších informací a řešení problémů.
- **Pravidelné revize:** Vykonávání pravidelných revizí výkonu s cílem vyhodnotit stav systému a identifikovat oblasti, které je třeba zlepšit.
- **Integrace monitorování s CI/CD:** Začlenění monitorovacích nástrojů a postupů do procesu CI/CD a zajistit tak průběžné vyhodnocování výkonu a stavu systému.

Smyčky zpětné vazby a neustálé zlepšování

V DevOps jsou smyčky zpětné vazby mechanismy, které usnadňují nepřetržitý tok informací z monitorování a provozních zkušeností zpět k vývojovým a provozním týmům. Vytvořením smyček zpětné vazby mohou týmy rychleji identifikovat a řešit problémy, často dříve, než mají dopad na koncového uživatele, poučit se ze selhání a incidentů a předcházet tak jejich budoucímu výskytu, přizpůsobovat a zdokonalovat procesy, nástroje a chování na základě reálných zkušeností a metrik.

Neustálé zlepšování v DevOps je poháněno iterativním zpracováním zpětné vazby s cílem zlepšit vývojové postupy i provozní stabilitu. To zahrnuje:

- **Iterativní vývoj:** Začlenění zpětné vazby do procesu vývoje za účelem zdokonalení a vylepšení vlastností a funkcí.
- **Optimalizace procesů:** Zefektivnění procesů na základě zpětné vazby s cílem zvýšit efektivitu a snížit plýtvání.
- **Zvyšování kvalifikace a znalostí:** Identifikace příležitostí ke vzdělávání na základě zpětné vazby s cílem zlepšit schopnosti a výkonnost týmu.

Nejlepší postupy pro podávání zpráv v DevOps

Efektivní reporting v DevOps zajišťuje transparentnost, podporuje odpovědnost a rozhodování založené na datech. Mezi osvědčené postupy patří:

- **Pravidelný reporting:** Vytváření pravidelných zpráv o klíčových metrikách, incidentech a iniciativách na zlepšení, abyste byla zachována transparentnost a informovali se zúčastněné strany.
- **Actionable Insights:** Zajištění, aby zprávy zdůrazňovaly využitelné poznatky, nikoliv pouze hrubé údaje, které by vedly ke zlepšení.
- **Vizuální řídicí panely:** Využívání řídicích panelů k zajištění přehledu o stavu systému, výkonnostních ukazatelích a probíhajících problémech v reálném čase.
- **Přizpůsobené reporty:** Přizpůsobené reporty potřebám a zájmům různých zúčastněných stran, tak aby byly předávány informace, které jsou pro každou skupinu nejdůležitější.

Komunikace zjištění a poznatků se zainteresovanými stranami

Jasná a efektivní komunikace je v systému DevOps klíčová, aby všechny zúčastněné strany pochopily aktuální stav, rizika a plány na zlepšení. Mezi efektivní komunikační strategie patří např:

- Zapojení zainteresovaných stran: Pravidelná spolupráce se zúčastněnými stranami, abyste došlo k pochopení jejich potřeby a byli infromování o vývoji a změnách.
- Transparentní komunikace: Být otevřený a upřímný ohledně problémů, neúspěchů a úspěchů.
- Kanály zpětné vazby: Kanály pro přijímání zpětné vazby od zúčastněných stran a včasné reagování na ni, abyste bylo podpořeno prostředí spolupráce.
- Řízení změn: Včasné sdělení o plánovaných změnách, odůvodnění a očekávané dopady, abyste byl minimalizován odpor a byla zajištěna hladká implementace.

Využitím smyček zpětné vazby a postupů neustálého zlepšování mohou týmy DevOps zvýšit efektivitu, spolehlivost a kvalitu svých aplikací a služeb. Efektivní podávání zpráv a komunikace zajišťují, že všechny zúčastněné strany jsou sladěny, informovány a zapojeny, což dále podporuje cíle neustálého zlepšování a spolupráce, které jsou vlastní filozofii DevOps.

Naposledy změněno: pátek, 22. března 2024, 09.37

◄ Tutoriál ke canary deployments v Kubernetes

Přejít na...

Průběžné testování znalostí - Monitorování, měření, reportování ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat
Stáhněte si mobilní aplikaci

DevOps

Automatizace DevOps

Automatizace DevOps

Automatizace v DevOps označuje proces navrhování, implementace a správy systémů a pracovních postupů pomocí softwaru, nikoliv manuální práce. To zahrnuje automatizaci integrace, nasazení, monitorování a správy procesů vývoje softwaru a provozních úloh. V rámci DevOps zahrnuje automatizace širokou škálu činností včetně odevzdávání kódu, testování, nasazování, zajišťování infrastruktury, konfigurace, monitorování a shromažďování zpětné vazby. Cílem je minimalizovat manuální zásahy, což následně snižuje počet chyb, zvyšuje efektivitu a zrychluje cykly dodávek.

Automatizace hraje klíčovou roli při umožnění základních principů DevOps, jako je kontinuální integrace, kontinuální doručování a kontinuální monitorování. Mezi hlavní přínosy patří:

- Zvýšení efektivity a rychlosti: Automatizace opakujících se úloh snižuje časovou náročnost a zvyšuje rychlost vývojových a nasazovacích cyklů.
- Zvýšení konzistence a spolehlivosti: Automatizace zajišťuje konzistentní a opakovatelné provádění úloh, čímž se snižuje riziko lidské chyby a zvyšuje spolehlivost systému.
- Zlepšená škálovatelnost: Automatizované procesy lze snadno rozšířit nebo snížit, aby se přizpůsobily změnám v poptávce, aniž by bylo nutné vynaložit značné dodatečné úsilí.
- Zlepšení spolupráce a komunikace: Odstraněním ručních překážek a snížením závislosti na konkrétních osobách podporuje automatizace lepší spolupráci a komunikaci mezi vývojovými a provozními týmy.
- Vyšší kvalita a spokojenost zákazníků: Průběžné testování a monitorování umožněné automatizací pomáhá udržovat vysokou kvalitu a rychle reagovat na potřeby a zpětnou vazbu zákazníků.

Integrací automatizace do postupů DevOps mohou organizace dosáhnout racionálnějšího, efektivnějšího a účinnějšího vývoje softwaru a provozních pracovních postupů. To vede k rychlejším dodacím lhůtám, lepší kvalitě produktů a vyšší spokojenosti zákazníků, což je v úzkém souladu s obchodními cíli a požadavky trhu.

Automatizační nástroje a technologie

Přehled běžných automatizačních nástrojů:

- **Shell skripty/batch skripty:** Skriptování podporované přímo operačním systémem umožňuje vytváření jednoduchých programů, které často slouží jako základ pro automatizaci a nebo jako doplňková technologie pro realizaci jednotlivých automatizačních úkonů. Na OS Linux v minulosti byly rovněž velmi časté Perl skripty, AWK skripty aj.
- **Jenkins:** Jedná se o open-source automatizační server, který usnadňuje kontinuální integraci a kontinuální dodávku. Jenkins automatizuje části vývoje softwaru související s vytvářením, testováním a nasazováním, čímž usnadňuje kontinuální integraci a technické aspekty kontinuálního dodávání.
- **Ansible:** Nástroj s otevřeným zdrojovým kódem pro poskytování softwaru, správu konfigurace a nasazování aplikací. Nepoužívá žádné agenty ani další vlastní bezpečnostní infrastrukturu, což usnadňuje nasazení.
- **Docker:** Je to sada produktů typu platforma jako služba, které používají virtualizaci na úrovni operačního systému k poskytování softwaru v balíčcích zvaných kontejnery. Kontejnery jsou navzájem izolované a obsahují vlastní software, knihovny a konfigurační soubory; mohou spolu komunikovat prostřednictvím přesně definovaných kanálů.

Nejlepší postupy pro efektivní automatizaci

- Začněte v malém a rozšiřujte: Automatizaci je nejlepší začínat od jednodušších ale často se opakujících úkolů a poté ji dále rozšiřovat.
- Udržujte dokumentaci: Podrobná dokumentace o procesech automatizace, konfiguracích a úpravách může pomocí v orientaci v posléze složitém systému, při jeho rozšiřování, úpravách nebo při řešení chyb, pokud dojde k selhání některých procesů.
- Pravidelné revize a aktualizace: Pravidelná revize a aktualizace automatizačních skriptů a konfigurace, zajišťuje aby se přizpůsobily novým požadavkům a technologiím.
- Zajišťování bezpečnosti: Od samého počátku je třeba dbát na bezpečnostní principy a postupy v rámci automatizačních procesů. Automatizace často znamená, že skripty musí provádět připojení na vzdálené prostředky, kde je vyžadována autentizace a autorizace a je třeba zvolit vhodný a bezpečný způsob pro takovou činnost.
- Spolupracujte a sdílejte znalosti: Podpora kultury spolupráce, kde jsou znalosti a osvědčené postupy sdíleny mezi týmy.

Výzvy v automatizaci

Automatizace procesů DevOps může výrazně zvýšit efektivitu a konzistenci, ale představuje také několik výzev:

- Složitost a problémy s integrací: Automatizační nástroje a procesy mohou být složité a integrace více nástrojů (každý s vlastní konfigurací a závislostmi) může být náročná.
- Odolnost vůči změnám: Členové týmu se mohou bránit přijetí nových nástrojů a změně stávajících pracovních postupů, zejména pokud nejsou přesvědčeni o jejich přínosu.
- Nedostatek odborných znalostí: Nedostatečné znalosti o nástrojích nebo osvědčených postupech pro automatizaci mohou vést k neoptimální implementaci.
- Problémy se škálováním: Nastavení automatizace, která dobře fungují v malých prostředích, nemusí být efektivně škálovatelná, když se požadavky zvyšují.
- Zabezpečení a dodržování předpisů: Automatizace pracovních postupů může přinést nová bezpečnostní zranitelná místa, zejména pokud není zabezpečení integrováno do automatizačních procesů od samého počátku.
- Údržba a aktualizace: Automatizované systémy vyžadují průběžnou údržbu a aktualizace, které mohou být časově náročné a složité.

Naposledy změněno: sobota, 23. března 2024, 07.18

◄ Tutoriál k ELK Stack

Přejít na...

Jenkins - opensource automatizační server ►

DevOps

CI/CD

Kontinuální integrace/kontinuální dodání/kontinuální nasazení (CI/CD)

Kontinuální integrace (CI) a kontinuální dodání/nasazení (CD) jsou základními postupy DevOps, které zjednodušují a automatizují kroky spojené s vývojem a nasazením softwaru. Cílem těchto postupů je zvýšit rychlost, kvalitu a spolehlivost dodávek softwaru.

- **Kontinuální integrace (CI):** Zahrnuje automatické vytváření a testování změn kódu, jakmile jsou odevzdány do systému správy verzí. To pomáhá rychle identifikovat a opravit chyby integrace, zlepšuje kvalitu softwaru a urychluje proces dodání. V rámci pipeline dochází typicky ke kompilaci, otestování a vyhodnocení kvality kódu. Pipeline může zahrnout velké množství nástrojů, typů testů i testovaných platformem.
- **Kontinuální dodání/nasazení (CD):** Rozšiřuje kontinuální integraci automatickým vytvořením balíčku, který je možné nasadit na cílové prostředí nebo přímo nasazením do cílového prostředí. Continuous delivery (dodání) by mělo prakticky vytvořit artefakt, který představuje balíček vytvořeného softwaru, který je snadno nasaditelný. Continuous deployment (nasazení) pak automatizuje i proces nasazení do produkčního prostředí. V praxi se tyto pojmy poměrně často zaměňují či ignorují a jako CD je chápáno continuous delivery, které provádí i automatické nasazení do cílového prostředí.

V rámci automatizačních nástrojů je typicky vytvořena pipeline ("potrubí"), kde na vstupu je zdrojový kód aplikace a jednotlivé kroky pipeline pak představují sestavení, testování, nasazení. Jednotlivé kroky se mohou skládat z mnoha dílčích kroků či úloh, mohou být vykonávány sériově či paralelně. Pipeline obvykle funguje ve stejném režimu jako testování, selhání libovolného kroku znamená typicky selhání celé pipeline a zrušení vykonávání následných kroků.

Nástroje běžně používané v procesech CI/CD:

- **Jenkins:** Automatizační server s otevřeným zdrojovým kódem, který nabízí zásuvné moduly pro podporu sestavování, nasazování a automatizace jakéhokoli projektu. Jenkins je vysoce přizpůsobitelný a má rozsáhlou komunitu.
- **GitLab CI:** Je součástí GitLabu a jedná se o webový nástroj pro životní cyklus DevOps, který poskytuje službu CI/CD integrovanou do platformy GitLab.
- **CircleCI:** Je to cloudový nástroj CI/CD, který rychle, bezpečně a ve velkém měřítku automatizuje proces vývoje.
- **Travis CI:** Hostovaná služba kontinuální integrace sloužící k sestavování a testování softwarových projektů umístěných na serverech GitHub a Bitbucket.
- **GitHub Actions:** Umožňuje automatizovat pracovní postupy přímo z úložiště GitHub. Můžete sestavovat, testovat a nasazovat kód přímo z GitHubu.

Naposledy změněno: sobota, 23. března 2024, 09.50

◄ Úvod do CI/CD nástroje Jenkins

Přejít na...

Průběžné testování znalostí - CI/CD ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)

Čeština (cs)
English (en)

Souhrn uchovávaných dat

Stáhněte si mobilní aplikaci

DevOps

Testování

Testování v DevOps

Testování je kritickou součástí metodiky DevOps, která je základem celkové kvality a spolehlivosti softwaru. V kultuře DevOps není testování izolovanou fází, ale nedílnou součástí celého životního cyklu vývoje a nasazení softwaru.

Přechod od tradičního testování k testování v kontextu DevOps:

- Kontinuální integrace a testování: V systému DevOps začíná testování již v počátečních fázích vývoje a je integrováno do procesu kontinuální integrace (CI). Tento posun od postupného testování ke kontinuálnímu testování umožňuje okamžitou zpětnou vazbu a rychlejší iterace.
- Automatizace: Zatímco tradiční testování může zahrnovat spoustu ruční práce, DevOps klade důraz na automatizované testování, které proces urychluje a zvyšuje efektivitu. Automatizované testy lze spouštět s každou revizí kódu, což zajišťuje okamžitou zpětnou vazbu.
- Přístup s posunem doleva: DevOps podporuje přístup "shift-left", kdy se testování provádí na začátku vývojového procesu. Cílem tohoto přístupu je identifikovat a opravit problémy dříve, než se stanou složitějšími a jejich řešení nákladnější.
- Spolupráce a odpovědnost: Na rozdíl od tradičních modelů, kde je testování výhradní odpovědností týmů QA, DevOps podporuje sdílenou odpovědnost mezi vývojáři, testery a provozem. Tato spolupráce zajišťuje, že kvalita je společným cílem v celém procesu vývoje a nasazení.

Díky integraci testování do celého procesu vývoje, integrace a nasazení mohou týmy DevOps zajistit rychlejší vydání bez kompromisů v kvalitě. Tento kontinuální a společný přístup k testování je nezbytný pro poskytování spolehlivého a kvalitního softwaru, který splňuje potřeby uživatelů a obchodní cíle.

Typy testování

Při vývoji softwaru se používá několik typů testování, aby se zajistilo, že aplikace splňuje své specifikace a očekávání uživatelů. Pochopení těchto různých typů může týmům pomoci použít správné testy ve správných fázích vývoje:

Jednotkové testování / unit testing:

- Definice: Testování jednotek zahrnuje testování jednotlivých komponent nebo jednotek kódu izolovaně od zbytku aplikace.
- Účel: Cílem je ověřit, že každá jednotka softwaru funguje tak, jak byla navržena.
- Praktika: Vývojáři obvykle píší testy jednotek a spouštějí je při každé změně, aby se ujistili, že nebyly zavedeny žádné nové chyby.

Integrační testování / integration testing:

- Definice: Integrační testování posuzuje interakci mezi integrovanými jednotkami nebo součástmi aplikace s cílem odhalit chyby rozhraní.
- Účel: Zajišťuje, aby různé části aplikace fungovaly společně podle očekávání.
- Praktika: Tyto testy se obvykle provádějí po jednotkových testech a mohou být automatizované nebo manuální.

Systémové testování / system testing:

- Definice: Systémové testování je komplexní testovací proces, který hodnotí kompletní a plně integrovaný softwarový produkt.
- Účel: Cílem je ověřit, zda celá aplikace splňuje stanovené požadavky.
- Praktika: Provádí je tým nezávislý na vývojovém týmu a pokrývá celou architekturu systému.

Akceptační testování / acceptation testing:

- Definice: Akceptační testování neboli uživatelské akceptační testování (UAT) je závěrečná fáze testování před vydáním softwaru.
- Účel: Posuzuje, zda software splňuje požadavky koncových uživatelů a zda je připraven k nasazení.
- Praktika: Toto testování obvykle provádí zákazník nebo koncoví uživatelé a může být manuální nebo automatizované na základě předem definovaných kritérií přijatelnosti.

Automatizované vs. manuální testování

Automatické testování:

- Definice: Automatizované testování využívá specializované nástroje k automatickému provádění testů, porovnávání skutečných výsledků s předpokládanými výsledky a hlášení výsledků.
- Přínosy: Je rychlejší než manuální testování, může být spolehlivější z hlediska přesnosti opakování a je nákladově efektivní pro procesy kontinuální integrace a kontinuálního nasazení.

- Případy použití: Nejvhodnější pro regresní testování, zátěžové testování a opakování stejných testů v různých prostředích.

Manuální testování:

- Definice: Manuální testování zahrnuje lidské testery, kteří hrají roli koncových uživatelů a ručně provádějí testovací případy bez použití automatizačních nástrojů.
- Výhody: Je flexibilnější a může odhalit jemné problémy s uživatelským prostředím nebo vizuální problémy, které by automatizované testy mohly přehlédnout.
- Případy použití: Nejlépe se hodí pro průzkumné testování, testování použitelnosti a ad hoc testování, kde je klíčová lidská intuice a kreativita.

Implementace automatizovaného testování

Automatizované testování je klíčovou součástí pipeline DevOps, která pomáhá zajistit rychlé a efektivní ověření změn kódu. Nastavení automatizovaného testování zahrnuje:

1. Identifikace testovacích případů pro automatizaci: Ne všechny testy jsou vhodné pro automatizaci. Identifikujte opakující se, časově náročné testy, které nevyžadují lidský úsudek.
2. Výběr správných nástrojů: Vyberte nástroje, které nejlépe vyhovují technologickému balíku aplikace, dovednostem týmu a potřebám testování.
3. Psaní testovacích skriptů: Vypracování automatizovaných testovacích skriptů na základě vybraných testovacích případů. Zajistěte, aby byly udržovatelné a opakovaně použitelné.
4. Integrace s CI/CD Pipeline: Nakonfigurujte nástroj CI/CD tak, aby spouštěl automatizované testy při každé revizi kódu nebo sestavení. Tím zajistíte okamžitou zpětnou vazbu o dopadu změn.
5. Udržování sad testů: Pravidelně kontrolujte a aktualizujte testovací skripty, aby se přizpůsobily novým funkcím a změnám v aplikaci.

Nástroje pro automatizované testování:

Několik nástrojů vyhovuje různým potřebám testování:

- Selenium: Oblíbený nástroj pro automatizaci webových prohlížečů. Podporuje více programovacích jazyků a frameworků.
- JUnit/TestNG: Široce používaný pro jednotkové testování aplikací v jazyce Java. Poskytují anotace pro identifikaci testovacích metod a očekávaných výsledků.
- Cypress: Moderní nástroj pro automatizaci webu vytvořený pro moderní web. Používá se pro end-to-end testování, podporuje načítání v reálném čase a interaktivní testování.
- Postman: Používá se pro testování API a umožňuje testerům vytvářet, sdílet, testovat a dokumentovat API.

Kontinuální testování

Kontinuální testování zahrnuje integraci automatizovaných testů do každé fáze pipeline CI/CD a poskytuje průběžnou zpětnou vazbu o obchodních rizicích spojených s kandidátem na vydání softwaru. Integrace zahrnuje:

- Testy před odesláním (commitem): Spuštění rychlých, izolovaných jednotkových testů před odevzdáním kódu, aby se zajistilo, že změny nerozbijí stávající funkčnost.
- Testy po odeslání (commitem): Po sloučení kódu proveďte komplexnější testy, včetně integračních a systémových testů, abyste ověřili změny v kontextu celé aplikace.
- Testy před nasazením: Před nasazením do produkce proveďte akceptační a regresní testy, abyste zajistili, že aplikace splňuje požadavky uživatelů a obchodní požadavky.
- Testy po nasazení: Po nasazení proveďte testování a monitorování, abyste zajistili, že aplikace bude v ostrém prostředí fungovat podle očekávání.

Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat
Stáhněte si mobilní aplikaci

DevOps

Bezpečnost

Bezpečnost v DevOps (DevSecOps)

DevSecOps představuje integraci bezpečnostních postupů do procesu DevOps. Jedná se o přístup ke kultuře, automatizaci a návrhu platformy, který integruje bezpečnost jako sdílenou odpovědnost v celém životním cyklu IT.

- DevSecOps je postaven na myšlence, že bezpečnost by měla být klíčovou součástí procesů vývoje a provozu, nikoliv oddělená nebo považovaná za vedlejší.
- Zahrnuje začlenění bezpečnostních kontrol a postupů do všech kroků životního cyklu vývoje softwaru (SDLC), od počátečního návrhu přes integraci, testování, nasazení a dodání softwaru.
- Cílem je, aby se bezpečnost stala nedílnou součástí pracovního postupu, nikoli samostatnou nebo závěrečnou fází.

Důležitost integrace zabezpečení do procesu DevOps:

- Rychlá detekce a zmírnění: Díky integraci zabezpečení v rané fázi vývojového procesu mohou týmy rychleji identifikovat a řešit zranitelnosti.
- Nákladová efektivita: Obecně je nákladově efektivnější řešit bezpečnostní problémy během vývoje než po nasazení softwaru.
- Soulad s předpisy a důvěra: Dodržování osvědčených postupů a předpisů v oblasti zabezpečení pomáhá udržet důvěru zákazníků a splňuje právní a regulační požadavky.
- Zlepšení spolupráce: DevSecOps podporuje spolupráci mezi vývojovými, bezpečnostními a provozními týmy, což vede k bezpečnějšímu a kvalitnějšímu softwaru.
- Odolnost: Integrace zabezpečení do procesu DevOps zvyšuje celkovou odolnost softwaru proti kybernetickým hrozbám.

Posunutím zabezpečení doleva - tedy jeho dřívějším zařazením do procesu vývoje - umožňuje DevSecOps týmům vytvářet bezpečný software rychlostí a v rozsahu, který moderní podniky vyžadují. Přeměňuje zabezpečení z blokátoru na nástroj, který umožňuje organizacím rychle inovovat, aniž by to bylo na úkor bezpečnosti.

Nejlepší bezpečnostní postupy v DevOps

Začlenění bezpečnosti do DevOps - někdy označované jako DevSecOps - vyžaduje soubor osvědčených postupů a zásad, které mají zlepšit bezpečnost a kvalitu softwaru, aniž by byla obětována rychlost a efektivita, kterou DevOps umožňuje.

1. Zapojení zabezpečení do CI/CD Pipeline: Integrace bezpečnostních nástrojů a procesů přímo do kontinuální integrace/kontinuálního nasazení (CI/CD). Tím je zajištěno, že bezpečnostní kontroly budou prováděny automaticky a důsledně v každé fázi vývoje softwaru.
2. Přesun zabezpečení doleva: Zahrnutí bezpečnostních aspektů již v rané fázi životního cyklu vývoje. To zahrnuje zapojení bezpečnostních týmů již od fáze plánování a začlenění bezpečnostního testování a kontroly do raných fází vývoje.
3. Automatizujte bezpečnostní procesy: Automatizace bezpečnostního testování, skenování zranitelností a kontroly konfigurace k rychlejší a efektivnější identifikaci a nápravě bezpečnostních problémů.
4. Praktikovat nejmenší oprávnění: Zajištění, aby uživatelé, systémy a procesy pracovaly s minimálními úrovněmi přístupu nebo oprávněními nezbytnými pro výkon jejich funkcí. Tím se sníží potenciální dopad narušení.
5. Pravidelné aktualizace: Zachování všech systémů, knihoven a frameworků používané v procesu vývoje a nasazení v aktuálním stavu s nejnovějšími záplatami na ochranu před známými zranitelnostmi.
6. Správa tajemství: Bezpečná správa tajemství, jako jsou klíče API, pověření a certifikáty. Zajištění jejich šifrování a přísnou kontrolu přístupu k nim.
7. Průběžné monitorování a reakce: Zavedení nepřetržitého monitorování infrastruktury a aplikací za účelem odhalení hrozeb a reakce na ně v reálném čase. To by mělo zahrnovat protokolování, detekci anomálií a procesy reakce na incidenty.

Praktiky bezpečného kódování:

- Analýza kódu: Využívání nástrojů statické a dynamické analýzy kódu k identifikaci potenciálních bezpečnostních problémů během vývoje.
- Přezkumy kódu: Provádění pravidelných revizí kódu se zaměřením na identifikaci bezpečnostních chyb.
- Školení o bezpečnosti pro vývojáře: Pravidelné školení vývojářů o bezpečnosti, aby byli informováni o postupech bezpečného kódování a nových hrozbách.

Skenování zranitelností a hodnocení rizik:

- Pravidelné skenování: Pravidelné skenování zranitelností kódu i infrastruktury s cílem identifikovat a řešit zranitelnosti dříve, než mohou být zneužity.

- Komponenty třetích stran: Využívání nástrojů pro sledování knihoven a komponent třetích stran, zda neobsahují známé bezpečnostní problémy.
- Ohodnocení rizik: Hodnocení rizik s cílem identifikovat, klasifikovat a stanovit priority bezpečnostních rizik a informovat o strategiích jejich zmírňování.

Zavádění osvědčených postupů zabezpečení:

- Vytvoření kultury zaměřené na bezpečnost: Podpoření kultury, ve které je bezpečnost odpovědností všech, nejen bezpečnostního týmu.
- Zapojení zabezpečení do agilních pracovních postupů: Začlenění bezpečnostních úkolů a kontrolních bodů do agilních sprintů a příběhů.
- Dokumentace a školení: Údržba komplexní dokumentace bezpečnostních zásad a postupů a průběžné školení všechn členů týmu v oblasti bezpečnosti.

Nástroje a technologie pro DevSecOps

Implementace DevSecOps zahrnuje řadu nástrojů určených k automatizaci a zvýšení bezpečnosti v celém procesu vývoje a nasazení. Zde je přehled některých široce používaných nástrojů:

SonarQube:

- Přehled: SonarQube je nástroj pro statickou analýzu kódu, který slouží k identifikaci chyb, zranitelností a "code smells" v kódu. Podporuje více jazyků a bez problémů se integruje do CI/CD potrubí.
- Použití v DevSecOps: Lze jej použít k provádění automatizovaných revizí kódu, sledování technického dluhu a zajištění toho, aby kódová základna zůstala čistá a bezpečná.

OWASP ZAP (Zed Attack Proxy):

- Přehled: OWASP ZAP je bezpečnostní nástroj s otevřeným zdrojovým kódem, který slouží k vyhledávání zranitelností webových aplikací ve fázích testování. Nabízí automatické skenery i nástroje pro ruční testování zabezpečení.
- Využití v DevSecOps: ZAP lze integrovat do CI/CD pipelines pro průběžné testování zabezpečení, což pomáhá týmům identifikovat a opravovat bezpečnostní zranitelnosti v rané fázi vývojového cyklu.

Bezpečnostní skenování Dockeru:

- Přehled: Docker Security Scanning je služba, která skenuje obrazy Docker na zranitelnosti. Zkoumá vrstvy v obrazech na přítomnost známých zranitelností pomocí databáze CVE (Common Vulnerabilities and Exposures).
- Použití v DevSecOps: Díky integraci Docker Security Scanning do CI/CD pipeline mohou týmy zajistit, že jejich obrazy Docker neobsahují známé zranitelnosti před jejich nasazením do produkce.

Snyk:

- Přehled: Snyk je nástroj, který pomáhá odhalovat zranitelnosti a licenční problémy open-source. Lze jej integrovat přímo do pracovního postupu vývoje.
- Použití v DevSecOps: Snyk zajišťuje nepřetržité monitorování a zasílá upozornění při objevení nových zranitelností v existujících projektech, čímž zajišťuje bezpečnost a aktuálnost závislostí.

HashiCorp Vault:

- Přehled: Hashicorp Vault je nástroj pro správu tajemství, šifrování jako službu a správu privilegovaného přístupu.
- Použití v DevSecOps: Vault umožňuje týmům DevOps bezpečně ukládat citlivé informace, jako jsou klíče API, hesla a certifikáty, a přistupovat k nim, čímž se snižuje riziko narušení bezpečnosti.

Aqua Security:

- Přehled: Aqua Security poskytuje komplexní bezpečnostní řešení pro kontejnerové aplikace, od vývoje až po produkci.
- Použití v DevSecOps: Skenuje obrazy kontejnerů na zranitelnosti, kontroluje konfigurace kontejnerů a vynucuje bezpečnostní zásady za běhu.

Fortify:

- Přehled: Fortify nabízí sadu nástrojů pro statické a dynamické testování bezpečnosti aplikací (SAST a DAST) a analýzu složení softwaru (SCA).
- Použití v DevSecOps: Fortify lze integrovat do CI/CD pipelines a automaticky identifikovat bezpečnostní problémy v kódu, open-source komponentách a běžících aplikacích.

Zapojení automatizace zabezpečení do CI/CD Pipeline

Automatizace zabezpečení zahrnuje integraci nástrojů a procesů zabezpečení přímo do pipeline kontinuální integrace/kontinuálního nasazení (CI/CD). Tím je zajištěno, že kontroly a testy zabezpečení jsou prováděny automaticky v každé fázi vývoje a nasazení softwaru, což usnadňuje průběžné hodnocení zabezpečení.

1. Analýza kódu: Integrace nástrojů pro statické testování zabezpečení aplikací (SAST) do pipeline CI, které skenují zdrojový kód na zranitelnosti ihned po odevzdání kódu.
2. Skenování závislostí: Použití nástrojů pro analýzu složení softwaru (SCA) k automatickému skenování závislostí na známé zranitelnosti, kdykoli jsou přidány nebo aktualizovány nové knihovny nebo balíčky.
3. Skenování kontejnerů: Využití nástrojů pro skenování kontejnerů, které budou před nasazením obrazů Docker zkoumat, zda neobsahují bezpečnostní problémy.
4. Dynamické testování: Integrace nástrojů pro dynamické testování zabezpečení aplikací (DAST), abyste bylo možné provádět automatizované testy běžících aplikací v předprodukčních nebo stagingových prostředích.
5. Skenování infrastruktury jako kódu: Bezpečnostní kontroly konfigurací infrastruktury jako kódu (IaC), abyste bylo zajištěno, že skripty pro poskytování infrastruktury dodržují osvědčené postupy zabezpečení.

Naposledy změněno: neděle, 24. března 2024, 09.00

◄ Průběžné testování znalostí - Testování

Přejít na...

Průběžné testování znalostí - Bezpečnost ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat
Stáhněte si mobilní aplikaci

DevOps

Praktický projekt DevOps

Praktický návrh projektu DevOps: Aplikace online knihkupectví

Přehled projektu:

Cílem projektu je vytvořit aplikaci online knihkupectví v jazyce Java. Projekt bude zahrnovat implementaci kompletní CI/CD pipeline, která zajistí postupy, jako je řízení verzí, kontinuální integrace, kontinuální nasazení, automatizované testování, kontrola kvality kódu, logování a monitorování.

Jednotlivé komponenty a technologie uvedené jsou primárně pouze doporučením, po konzultaci lze zvolit i jiné.

Používané komponenty

- Zdrojový kód: Webová aplikace založená na jazyce Java (doporučený framework Spring Boot kvůli snadnému nastavení a komunitní podpoře).
    - Maven nebo Gradle pro sestavení a správu projektu.
- Repozitář: GitHub/GitLab pro správu verzí.
- Kontinuální integrace/kontinuální nasazení:
    - Jenkins: Jako automatizační server pro vytváření potrubí CI/CD. Mezi alternativy patří GitLab CI nebo CircleCI pro ty, kteří dávají přednost integrovanější platformě.
    - Docker: Pro kontejnerizaci aplikace Java a zajištění konzistence v různých prostředích.
    - SonarQube: Pro průběžnou kontrolu kvality kódu a měření technického dluhu.
- Testování:
    - JUnit: Pro jednotkové testování kódu jazyka Java.
    - Selenium: Pro automatizované testy v prohlížeči, které simulují interakce s uživatelem.
    - JaCoCo: Pro analýzu pokrytí kódu integrovanou do sestavovacích skriptů Maven nebo Gradle.
- Nasazení a orchestrace:
    - Kubernetes: Pro orchestraci nasazení, škálování a správu kontejnerů. Pro lokální testování lze použít Minikube nebo K3s.
    - Helm (volitelně): Pro správu aplikací Kubernetes, která zjednodušuje nasazení aplikací a služeb.
- Protokolování a monitorování:
    - ELK Stack (Elasticsearch, Logstash, Kibana): Pro logování, monitorování a vizualizaci logů z aplikací a infrastruktury Java.
    - Prometheus a Grafana: Pro monitorování výkonu aplikace a infrastruktury a vizualizaci metrik.

Fáze projektu

Vývoj aplikace:

- Vývoj základní aplikace online knihkupectví v jazyce Java s využitím frameworku Spring Boot.
- Zajistěte, aby aplikace obsahovala funkce, jako je prohlížení knih, přidávání do košíku a pokladna.

Nastavení CI/CD Pipeline:

- Inicializujte repozitář Git a integrujte jej s Jenkinsem pro kontinuální integraci.
- Nastavte potrubí Jenkins pro automatizaci procesů sestavení, testování a nasazení.
- Nakonfigurujte SonarQube v pipeline pro průběžnou kontrolu kvality kódu.

Kontejnerizace a orchestrace:

- Dockerizujte aplikaci Java a odešlete obraz na DockerHub.
- Vytvoření souborů pro nasazení a služeb Kubernetes pro nasazení aplikace.
- Použijte Helm k zabalení a nasazení těchto konfigurací do Kubernetes (volitelně).

Testování:

- Napište jednotkové testy pomocí JUnit a integrujte je do sestavovacího potrubí.
- Nastavte Selenium pro automatizované testy uživatelského rozhraní.
- Nakonfigurujte JaCoCo pro pokrytí kódu a zajistěte, aby byl součástí procesu CI.

Monitorování a protokolování:

- Nastavte ELK Stack pro sběr, analýzu a vizualizaci protokolů z aplikace.
- Nasaďte Prometheus a Grafana pro monitorování stavu aplikace a infrastruktury.

Dokumentace:

- Zdokumentujte proces nastavení a konfigurace, včetně způsobu spouštění testů a nasazení aplikace.
- Poskytněte pokyny pro monitorování a řešení problémů s aplikací.

Naposledy změněno: neděle, 24. března 2024, 09.26

◄ Integrace nástroje SonarQube do Jenkins CI/CD

Přejít na...

(krycí list) ►

Nápověda a dokumentace (anglicky)
Kontaktujte podporu stránek

Jste přihlášeni jako Lukáš Čegan (Odhlásit se)
DANTE_DEV

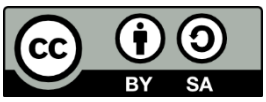
Moje stránka
Kalendář
Kontakty
Mahara
Kurzy
Čeština (cs)
Čeština (cs)
English (en)

Souhrn uchovávaných dat

Stáhněte si mobilní aplikaci

UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vytvořeno v rámci projektu **Digitalizace studijních Agend, Nové TechnologiE, systémy a přístupy k výuce na UPCE**, reg. č. NPO_UPCE_MSMT-16591/2022.



Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY