TEXT S OTÁZKAMI

CI/CD (Continuous Integration / Continuous Deployment or Delivery) je sada postupů v moderním vývoji softwaru (DevOps), jejichž cílem je automatizovat a zrychlit cyklus dodávání softwaru. Continuous Integration (CI) se zaměřuje na časté a automatizované slučování kódu od více vývojářů do sdíleného repozitáře. Každé sloučení spouští automatický build a sadu testů, aby se včas odhalily integrační problémy. Continuous Deployment (CD) navazuje na CI a automatizuje nasazení úspěšně otestovaného kódu do produkčního prostředí.

GitHub Actions je platforma pro automatizaci pracovních postupů (workflows) přímo v prostředí GitHubu. Umožňuje implementovat CI/CD. Základní koncepty jsou definovány v YAML souborech.

Základní stavební kameny GitHub Actions jsou:

- Workflow: Celý automatizovaný proces je definovaný v jednom YAML souboru. Je spouštěn specifickou událostí (event), jako je nahrání změn do repozitáře (push) nebo manuální pokyn (workflow_dispatch). Workflow se skládá z jednoho nebo více jobů.
- Job: Sada kroků, které se vykonávají na stejném výpočetním prostředí – runneru. Pokud není stanoveno jinak, všechny joby v rámci jednoho workflow startují současně (paralelně). Jejich vzájemnou závislost a pořadí spuštění lze definovat pomocí klíčového slova needs. Například needs: build znamená, že job se nespustí, dokud job s názvem build úspěšně neskončí.
- Runner: Serverové prostředí, na kterém se job spouští. Pro každý job je vytvořena čistá, nová instance runneru. Tato instance je po dokončení jobu zničena.
- Step: Nejmenší jednotka práce. Může spustit příkaz (např. run) nebo použít action (např. uses)
- Artefakt: Soubory, které jeden job vytvoří a uloží, aby si je jiný job mohl stáhnout a použít. Protože každý job běží v oddělené instanci runneru, přímé sdílení souborů mezi nimi není možné. Artefakty jsou jediným způsobem, jak předat data (kromě zdrojového kódu) z jednoho jobu do druhého.

Otázky:

1.Workflow obsahuje dva joby: build a deploy. Job deploy má v definici uvedeno needs: build. Job build vytvoří soubor app.bin v pracovním adresáři. Jakým způsobem může job deploy tento soubor použít?

a) Přímo, protože needs: build zajistí, že deploy běží na stejném runneru a má přístup ke stejnému souborovému systému.
b) Nemůže ho použít, protože joby jsou plně izolované a needs definuje pouze pořadí spuštění, nikoli sdílení dat.
c) Musí ho použít skrze artefakty: build musí soubor nahrát jako artefakt a deploy si ho následně musí stáhnout.
d) Použije akci actions/checkout@v4, která kromě zdrojového kódu stáhne i všechny soubory vytvořené předchozími joby.

2.Job s názvem build-app úspěšně vytvoří soubor program.exe. Následující job s názvem upload-release potřebuje tento soubor zpracovat. Jaký mechanismus popsaný v textu musí být použit?

a) Job upload-release musí použít run k přímému zkopírování souboru program.exe z instance runneru jobu build-app.
b) Job build-app musí soubor program.exe nahrát jako artefakt, který si následně job upload-release stáhne.
c) Oba joby musí být spuštěny na stejném typu runneru (např. ubuntu-latest), což automaticky zajistí sdílení souborů.
d) Musí se použít klíčové slovo needs, které zajistí předání souboru mezi joby.

3.V jobu jsou dva kroky vykonané za sebou: krok A vytvoří soubor output.txt pomocí run, krok B má přečíst tento soubor. Co z toho podle textu platí?

a) Krok B nemůže číst output.txt, protože každý krok běží na jiné instanci runneru.
b) Krok B může číst output.txt, protože všechny kroky jobu se vykonávají na téže nově vytvořené instanci runneru.
c) Krok B může číst output.txt pouze pokud bylo předtím použito actions/upload-artifact.
d) Krok B může číst output.txt jen pokud job obsahuje needs odkaz na předchozí job.

4.Který z následujících popisů nejpřesněji vystihuje vztah mezi pojmy Job, Step a Runner tak, jak je definován v textu?

a) Runner je sada Jobů, přičemž každý Job se skládá z jednoho Stepu.
b) Job je sada Stepů, která se celá vykoná na jedné, nově vytvořené instanci Runneru
c) Step je sada Jobů, která se vykoná na jednom Runneru.
d) Runner je jeden konkrétní Step, který je spuštěn v rámci Jobu.

5.Pokud má pracovní postup (workflow) tři joby (A, B, C) a žádný z nich neobsahuje klíčové slovo needs, co můžeme s jistotou tvrdit o jejich spuštění?

a) Spustí se postupně v pořadí A, B, C.
b) Spustí se pouze job A; joby B a C se musí spustit manuálně.
c) Spustí se všechny současně a budou sdílet jedno společné prostředí runneru pro úsporu zdrojů.
d) Spustí se všechny současně a každý poběží v odděleném, izolovaném prostředí.

LOGICKÉ

Představte si, že máte za úkol seřadit pole (seznam) čísel. Zvolíte k tomu algoritmus **insertion sort,**, který se vyznačuje tím, že postupně buduje finální setříděné pole na jeho začátku.

Algoritmus pracuje následovně:

1. Pole se pomyslně rozdělí na dvě části: **setříděná** (na začátku obsahuje pouze první prvek) a **nesetříděná** (zbytek pole).
2. V každém kroku se vezme první prvek z **nesetříděné** části (nazvěme ho **aktuální prvek**).
3. Tento **aktuální prvek** se porovnává s prvky v setříděné části, a to od konce (zprava doleva).
4. Dokud je **aktuální prvek** menší než prvek v setříděné části, se kterým je porovnáván, posune se tento větší prvek o jednu pozici doprava, aby se uvolnilo místo.
5. Jakmile se najde správná pozice (tj. prvek vlevo je menší nebo se došlo na začátek pole), **aktuální prvek** se na tuto volnou pozici vloží.

Tímto se setříděná část rozšíří o jeden prvek a celý proces se opakuje, dokud nezmizí nesetříděná část.

Tvůj seznam čísel je:
[7, 3, 5, 8, 2, 9, 4]

6.Jak bude tento seznam vypadat v okamžiku, kdy je čtvrtý prvek původního seznamu (číslo 8) právě zpracován a zařazen na svou správnou pozici?

a) [3, 5, 7, 8, 2, 9, 4]
b) [3, 5, 7, 2, 8, 9, 4]
c) [2, 3, 5, 7, 8, 9, 4]
d) [3, 5, 8, 7, 2, 9, 4]

Jsi DevOps inženýr a analyzuješ logy z produkčního prostředí, abys zjistil příčinu stížností uživatelů na "zamrzání" a extrémní zpomalení webové aplikace. Systém se skládá ze tří hlavních komponent: WEB (webový server), APP (aplikační server) a DB (databáze).

Tvým úkolem je identifikovat celkovou dobu, po kterou byl problém přímo viditelný pro koncové uživatele.

Klíčové informace pro analýzu:

- WEB Server: Zpracovává HTTP požadavky. Standardní odpověď je HTTP 200. Pokud aplikační server neodpoví včas, WEB server vrátí chybu HTTP 504 Gateway Timeout, což je přímý indikátor problému pro uživatele.
- APP Server: Zpracovává logiku. Zaznamenává varování (WARN), pokud požadavek trvá příliš dlouho. Zaznamenává chybu (ERROR), pokud se nemůže připojit k databázi.
- DB Server: Zaznamenává chyby, jako je LOCK WAIT TIMEOUT, která nastane, když jedna transakce příliš dlouho blokuje jinou.

```
[37m[16:08:10] [34mDB   (INFO)[0m  Query executed successfully. (user_id=451)
[37m[16:09:15] [33mAPP  (WARN)[0m  Response time high: 3500ms for /api/v1/orders
[37m[16:10:20] [31mDB   (ERROR)[0m Transaction failed: LOCK WAIT TIMEOUT on table `invoices`
[37m[16:10:45] [31mWEB  (ERROR)[0m Response: [36mHTTP 504 Gateway Timeout[0m for /api/v1/dashboard
[37m[16:11:30] [31mAPP  (ERROR)[0m Failed to acquire DB connection.
[37m[16:11:55] [31mWEB  (ERROR)[0m Response: [36mHTTP 504 Gateway Timeout[0m for /api/v1/reports
[37m[16:12:15] [32mWEB  (INFO)[0m  Response: HTTP 200 OK for /status
[37m[16:12:50] [34mDB   (INFO)[0m  Deadlock resolved. Transaction for `invoices` rolled back.
[37m[16:13:20] [32mAPP  (INFO)[0m  DB connection pool recovered.
[37m[16:14:10] [31mWEB  (ERROR)[0m Response: [36mHTTP 504 Gateway Timeout[0m for /api/v1/profile
[37m[16:15:05] [31mWEB  (ERROR)[0m Response: [36mHTTP 504 Gateway Timeout[0m for /api/v1/orders
[37m[16:16:00] [32mWEB  (INFO)[0m  Response: HTTP 200 OK for /
```

7.Jak dlouho celkově trval časový interval, ve kterém se vyskytovaly výpadky služby viditelné pro uživatele (tj. od prvního HTTP 504 do posledního HTTP 504)?

a) 4 minut a 20 sekund
b) 4 minuty a 55 sekund
c) 1 minuta a 10 sekund
d) 30 sekund

Jako agent v nepřátelském území musíš dešifrovat zprávu od svého velení. Kódy se neustále mění, ale pro tento týden platí dohoda: klíčem je první sloka básně "Havran" od Edgara Allana Poea. Zpráva je série číselných dvojic, které odkazují na konkrétní písmena v textu básně. Tato metoda je bezpečná, protože bez znalosti přesného textu klíče je série čísel bezcenná.

Jak to funguje:

Klíčem je daný text. Každé písmeno v něm má souřadnice: [číslo řádku, číslo znaku na řádku].

- Ignorují se mezery a interpunkce. Počítají se pouze písmena.
- Číslování řádků i znaků začíná od 1.

Referenční text (klíč):

RÁZ KDYSI O PŮLNOCI ČASU ZLÉHO JSEM NAD KNIHAMI DUMAL VZÁCNÝCH VĚD AČ SKORO SPAL JSEM NAD TOU ČETBOU TESKNOU V TOM NĚKDO ZATKAL DVEŘE VCHODU VPŘED TO NÁVŠTĚVNÍK JENŽ TLUČE NA DVEŘE TU ŘEK JSEM SI NIC TO NENÍ NAD TO VĚT

Jak najít písmeno podle souřadnic:

- Příklad: Souřadnice [2, 8]
    1. Jdi na druhý řádek: *AČ SKORO SPAL JSEM NAD….*
    2. Najdi osmé písmeno na tomto řádku (ignoruj mezery): A-Č-S-K-O-R-O-**S**
    3. Hledané písmeno je S.

Tvá spojka ti předala následující sérii souřadnic. Dešifruj ji, abys zjistil/a heslo pro setkání. Jako tvůj klíč použij klíč v zadání.

Zašifrovaná zpráva: [2, 15] [1, 7] [4, 1] [3, 9] [3, 20] [1, 30]

8.Jak zní dešifrované heslo?

a) MSNŘVM
b) JYNŘVN
c) MSNŘVN
d) JYNPVN

Mladší administrátor má za úkol provést aktualizaci tří databázových replik: Primární (P), Sekundární (S) a Terciární (T). Aktualizační skript může běžet vždy jen na jedné replice a vyžaduje manuální spuštění a dohled. Během aktualizace je replika dočasně nekonzistentní. Existují však pravidla pro synchronizaci, která nesmí být porušena, aby nedošlo ke ztrátě dat:

- Primární replika (P) nesmí být ponechána v nekonzistentním stavu (tedy během aktualizace) současně se Sekundární (S), protože by se S mohla pokusit o synchronizaci z neúplných dat.
- Terciární replika (T) se používá pro analytické dotazy a nesmí být nikdy nekonzistentní ve stejný okamžik jako Primární (P), jinak by analytické reporty pracovaly s chybnými daty.

Administrátor může provádět dvě operace: AKTUALIZOVAT (spustí a dohlíží na aktualizaci jedné repliky) a STABILIZOVAT (ukončí dohled, replika je opět konzistentní). Každá operace se počítá jako jeden krok.

9.Jaký je minimální počet kroků potřebných k aktualizaci všech tří replik?

a) 5 kroků
b) 6 kroků
c) 7 kroků
d) 9 kroků

Systémový administrátor spravuje cluster se dvěma servery, Server A a Server B. Na Serveru A běží 4 virtuální stroje (VMs), každý o velikosti 100 GB. Cílem je přesunout všechny 4 VM na Server B. Síťové připojení mezi servery má však specifické vlastnosti:

- Propustnost sítě je 200 GB za minutu.
- Proces "živé migrace" jednoho VM zatíží síť po dobu 30 sekund. Během této doby nelze zahájit další migraci.
- Po každé migraci musí systém provést 15sekundovou synchronizaci a kontrolu, během které je síť také plně vytížena a nelze zahájit novou migraci.

10.Jaký je minimální čas potřebný k přesunu všech 4 virtuálních strojů ze Serveru A na Server B?

a) 2 minuty
b) 2 minuty a 45 sekund
c) 3 minuty
d) 4 minuty

POROZUMĚNÍ TEXTU

Standardy Wi-Fi, označované jako IEEE 802.11, se neustále vyvíjejí. Starší standardy jako 802.11g operovaly pouze v přetíženém frekvenčním pásmu 2,4 GHz, které je náchylné na rušení od jiných zařízení (mikrovlnné trouby, Bluetooth). Novější standardy, počínaje 802.11n, zavedly schopnost pracovat i v pásmu 5 GHz, které je méně zarušené a nabízí více kanálů, což umožňuje vyšší rychlosti. Pásmo 5 GHz má však oproti 2,4 GHz kratší dosah a horší prostupnost přes pevné překážky, jako jsou zdi. Moderní routery (standard 802.11ac, 802.11ax/Wi-Fi 6) jsou proto dvoupásmové a vysílají na obou frekvencích současně, což zařízením umožňuje vybrat si optimální pásmo podle vzdálenosti a podmínek.

11.Jaký kompromis musí uživatel zvážit při volbě mezi pásmy 2,4 GHz a 5 GHz, jak popisuje text?

a) Pásmo 2,4 GHz je vždy rychlejší, ale má menší dosah.
b) Pásmo 5 GHz nabízí potenciálně vyšší rychlost a méně rušení, ale za cenu kratšího dosahu a horšího signálu přes překážky.
c) Novější standardy jako Wi-Fi 6 opustily pásmo 2,4 GHz a fungují výhradně na 5 GHz.
d) Pásmo 5 GHz má delší dosah, ale je výrazně pomalejší kvůli většímu rušení.

Když zapnete počítač, první program, který se spustí, není operační systém, ale firmware uložený na čipu na základní desce. Historicky to byl BIOS (Basic Input/Output System), u moderních počítačů je to jeho nástupce UEFI (Unified Extensible Firmware Interface). Jeho prvním úkolem je inicializovat a otestovat základní hardware (procesor, paměť, disky) v procesu zvaném POST (Power-On Self-Test). Poté vyhledá na pevných discích, SSD či jiných médiích speciální program zvaný zavaděč (bootloader). Teprve tento zavaděč je zodpovědný za nalezení a spuštění jádra (kernelu) operačního systému, čímž mu předá řízení nad počítačem.

12.Jaká posloupnost nejlépe odpovídá událostem při startu počítače podle textu?

a) Operační systém -> POST test -> spuštění UEFI/BIOS.
b) Spuštění zavaděče -> inicializace hardwaru v UEFI/BIOS -> načtení jádra OS
c) UEFI/BIOS provede test hardwaru -> spustí zavaděč -> zavaděč spustí operační systém
d) Jádro operačního systému spustí UEFI/BIOS, který následně otestuje hardware.

Komprese dat snižuje jejich velikost a dělí se na dva základní typy. Bezeztrátová (lossless) komprese (např. u ZIP, PNG) funguje na principu hledání opakujících se vzorů a jejich nahrazení kratší reprezentací. Při dekompresi je možné obnovit data do naprosto identické podoby jako před kompresí. Ztrátová (lossy) komprese (např. u JPEG, MP3) dosahuje mnohem většího zmenšení souboru tím, že nevratně odstraňuje informace, které jsou pro lidské smysly (zrak, sluch) považovány za postradatelné. Například u obrázku se zahodí jemné barevné nuance, které oko stejně nerozliší. Výsledný soubor již nikdy nebude bitově shodný s originálem, ale pro člověka může vypadat nebo znít téměř identicky.

13.Pro jaký typ dat by bylo použití ztrátové komprese zcela nevhodné?

a) Pro fotografii z dovolené určenou k publikování na sociální síti.
b) Pro hudební skladbu určenou k poslechu na mobilním telefonu.
c) Pro archivaci textového dokumentu nebo spustitelného souboru programu.
d) Pro video streamované přes internet, kde je důležitá nízká datová náročnost.

C++ a Python reprezentují dva odlišné přístupy k programování. C++ je kompilovaný jazyk se statickou typovou kontrolou, což znamená, že typy proměnných (např. celé číslo, text) musí být deklarovány předem a jsou kontrolovány již při kompilaci. To umožňuje vytvořit vysoce optimalizovaný a rychlý kód s přímou správou paměti, ale za cenu vyšší složitosti a delšího vývojového cyklu. Python je naopak interpretovaný jazyk s dynamickou typovou kontrolou. Typy proměnných se neurčují předem a kontrolují se až za běhu. To výrazně zjednodušuje a zrychluje psaní kódu (méně kódu pro stejnou funkcionalitu), ale výsledný program je pomalejší a spotřebovává více paměti.

14.Co je hlavním kompromisem při volbě Pythonu namísto C++, jak naznačuje text?

a) Python umožňuje psát pouze velmi krátké programy, zatímco C++ zvládá i velké projekty.
b) Výměna za rychlejší a jednodušší vývoj je nižší výkon a vyšší paměťová náročnost výsledného programu.
c) Python je vhodný pouze pro webové aplikace, zatímco C++ je určen pro matematické výpočty.
d) Kód v Pythonu je obtížněji čitelný a vyžaduje mnohem více řádků pro stejný úkol než v C++.

Existují dva hlavní přístupy k vykreslování webových stránek. Při tradičním Server-Side Rendering (SSR) server připraví kompletní HTML stránku se všemi daty a pošle ji prohlížeči, který ji pouze zobrazí. To je rychlé pro první zobrazení a dobré pro SEO (vyhledávače vidí hned celý obsah). Při Client-Side Rendering (CSR) server pošle jen minimální HTML "kostru" a soubor JavaScriptu. Prohlížeč pak tento skript spustí, ten si dodatečně vyžádá data ze serveru (přes API) a dynamicky celou stránku sestaví a zobrazí až na straně klienta. Následné interakce jsou velmi rychlé (mění se jen části stránky), ale první načtení může být pomalejší a SEO je komplikovanější.

15.Co je klíčovým rozdílem v tom, co prohlížeč obdrží od serveru při prvním načtení stránky v modelu CSR oproti SSR?

a) Při CSR obdrží prohlížeč hotovou, kompletní HTML stránku, zatímco při SSR dostane jen JavaScript.
b) Při SSR neprobíhá žádná komunikace se serverem, vše se děje v prohlížeči.
c) Model SSR je pomalejší pro první zobrazení, protože server musí posílat mnohem více dat než u CSR.
d) Při CSR obdrží prohlížeč v podstatě "stavebnici" (kostru a skript), ze které si stránku sám dynamicky postaví, na rozdíl od hotové stránky u SSR.