TEXT S OTÁZKAMI

Git je distribuovaný verzovací systém, což znamená, že každý vývojář má na svém počítači kompletní kopii repozitáře, včetně celé jeho historie. To je zásadní rozdíl oproti centralizovaným systémům (jako SVN), kde je historie uložena na jediném centrálním serveru.

Základní jednotkou v Gitu je *commit* (revize). Commit je snímek (snapshot) všech souborů v projektu v daném čase. Každý commit má unikátní identifikátor (hash) vypočítaný z jeho obsahu a z identifikátoru předchozího commitu. Tímto způsobem jsou commity svázány do řetězce a tvoří historii. Tato struktura zajišťuje integritu – změna v jakémkoli starším commitu by změnila jeho hash a tím i hashe všech následujících commitů, což by bylo okamžitě detekováno.

Práce v Gitu se obvykle odehrává ve větvích. Větev je v podstatě jen lehký, pohyblivý ukazatel na konkrétní commit. Když vývojář vytváří novou funkcionalitu, obvykle si vytvoří novou větev, aby svou práci izoloval od hlavní (často nazývané *main* nebo *master*). Po dokončení práce může svou větev sloučit (merge) zpět do hlavní větve. Tento postup umožňuje paralelní vývoj více funkcí najednou, aniž by si vývojáři navzájem přepisovali nedokončenou práci.

Otázky:

1.Když se v Gitu změní starší commit (např. prostřednictvím rebase), co se stane s historií?

a) Změní se pouze tento konkrétní commit, ostatní commity zůstanou nedotčeny.
b) Změní se hash daného commitu a všechny následující commity, což vytvoří novou historii.
c) Git automaticky vygeneruje nový commit pro každou změnu, aniž by to ovlivnilo následnou historii.
d) Git pouze označí změněné commity jako neplatné, ale historie zůstává nezměněná.

2.Co přesně ovlivňuje hash (identifikátor) commitu v Gitu?

a) Datum a čas commitu – Git generuje hash hlavně podle timestampu.
b) Obsah souborů ve snapshotu a identifikátor(y) předchozího commitu.
c) Název repozitáře a jméno autora commitu.
d) Náhodné číslo vygenerované při vytvoření commitu.

3.Co je větev (branch) v Gitu v nejpřesnějším smyslu?

a) Kopie všech souborů projektu uložená odděleně pro každou funkci.
b) Sada commitů, které se automaticky synchronizují s jinými větvemi.
c) Lehký pohyblivý ukazatel (reference) na konkrétní commit.
d) Speciální typ commitu, který sleduje rozvoj projektu.

4.Co je to "commit" v Gitu?

a) Snímek všech souborů v projektu v daném čase, který slouží k zálohování dat.
b) Unikátní identifikátor pro každý repozitář.
c) Snímek všech souborů v projektu v daném čase, včetně identifikátorů všech předchozích commitů.
d) Historie všech změn provedených v projektu bez záznamu o konkrétních souborech.

5.Jaký je rozdíl mezi distribuovaným a centralizovaným verzovacím systémem, jak je uvedeno v textu?

a) V distribuovaném systému je historie uložena na jednom serveru, v centralizovaném na více serverech.
b) V centralizovaném systému je historie uložena na jednom centrálním serveru, v distribuovaném na více počítačích.
c) V distribuovaném systému si každý vývojář může upravit historii repozitáře podle potřeby.
d) V centralizovaném systému má každý vývojář vlastní kopii repozitáře, v distribuovaném má jednu.

LOGICKÉ

Představ si, že máš velkou databázi seřazenou číselně a chceš rychle najít požadované číslo. Použiješ **binary search**, což je algoritmus, který funguje na principu dělení seznamu na poloviny. Každý krok spočívá v porovnání hledaného čísla s prostředním číslem seznamu. Podle výsledku tohoto porovnání se rozhodneš, zda hledat v levé, nebo pravé polovině seznamu. Pokud máme už jen dva prvky, jako prostřední se bere první z nich. Tento proces opakuješ, dokud nenajdeš hledané číslo nebo dokud nezjistíš, že číslo v seznamu není.

Tvůj seznam čísel je:
[2, 5, 7, 10, 12, 15, 20, 25, 30, 35, 40]

6.Pokud máš následující seřazený seznam čísel a hledáš číslo *20*, kolik iterací binárního vyhledávání budeš potřebovat, než číslo najdeš?

a) 2 iterace
b) 3 iterace
c) 4 iterace
d) 5 iterací

Jako IT administrátor monitoruješ server, který vykazuje různé chyby a varování. V logu se objevují nejen běžné chyby (ERROR), ale také **FATAL ERROR**, který označuje vážný problém, který vede k **vypnutí serveru**. Tvým úkolem je analyzovat logy a zjistit, jak dlouho byl server mimo provoz.

```
2025-10-25 09:00:00 INFO System started successfully
2025-10-25 09:10:32 WARNING Low disk space on server
2025-10-25 09:15:48 ERROR Failed to connect to database
2025-10-25 09:20:00 INFO User login successful
2025-10-25 09:30:11 ERROR Timeout occurred during data transfer
2025-10-25 09:40:05 INFO System running smoothly
2025-10-25 09:45:00 ERROR Database connection lost
2025-10-25 10:00:00 WARNING Memory usage high on server
2025-10-25 10:05:20 ERROR Critical database error
2025-10-25 10:10:30 FATAL ERROR Server shutting down due to critical error
2025-10-25 10:15:00 INFO Server restarted successfully
2025-10-25 10:25:35 WARNING Low disk space on server
2025-10-25 10:30:00 ERROR System crash detected
2025-10-25 10:40:20 INFO System restarted successfully
2025-10-25 10:50:30 ERROR Network failure detected
```

7.Jak dlouho byl server mimo provoz?

a) 4 minut a 30 sekund
b) 5 minut a 10 sekund
c) 40 minut a 19 sekund
d) 20 minut a 30 sekund

Jako expert na kybernetickou bezpečnost dostaneš za úkol dešifrovat text, který byl zašifrován pomocí **Vigenèrovy šifry**. Vigenèrova šifra je metoda šifrování, která používá klíčové slovo k posunu písmen v textu. Na rozdíl od jednodušší Caesarovy šifry, kde jsou všechna písmena posunuta o stejný počet míst, zde se posun pro každé písmeno mění podle písmena v klíči.

1. Přiřazení čísel k písmenům: Každému písmenu abecedy přiřadíme číslo (používáme americkou abecedu) (A=0, B=1, C=2, ..., Z=25).
2. Klíč: Použijeme klíčové slovo, které se opakuje, dokud nepokryje celou délku zprávy. Pokud je zpráva delší než klíč, klíč se začne opakovat od začátku.
    a. Příklad: Pro zprávu ZAHRADA a klíč KLIC bude klíč použit jako KLICKLI.
3. Dešifrování: Pro dešifrování se od číselné hodnoty zašifrovaného písmene odečte číselná hodnota písmene klíče.
    a. Vzorec: Původní písmeno = (Zašifrované písmeno - Písmeno klíče)
    b. Důležité pravidlo "přetečení": Pokud vám po odečtení vyjde záporné číslo, pokračujete v počítání od konce abecedy. Například pokud dešifrujete písmeno B (hodnota 1) s klíčem D (hodnota 3), výpočet je 1 - 3 = -2. Od A

jdete o dvě pozice zpět, což vás dostane na Y. (Jiný způsob: k zápornému výsledku přičtěte 26. Tedy -2 + 26 = 24, což odpovídá písmenu Y).

8.Dešifrujte slovo ***DODJFW* pomocí klíče ***CRYPTO*****.**

a) AZXUKT
b) BXFUMI
c) FNFPJZ
d) BXZUMV

Správce sítě musí přesunout tři citlivé komponenty z jednoho serveru na druhý přes pomalý a nestabilní přenosový kanál. K dispozici má speciální přenosový nástroj, který může nést najednou pouze jednu komponentu. Správce musí dohlížet na přenos, a proto cestuje vždy s nástrojem. Problém je v závislostech mezi komponentami, které nesmí být ponechány na jednom serveru bez dozoru správce v určitých kombinacích:

- Data Processing Script (S) nesmí být ponechán sám s Transaction Log (L), protože by ho automaticky zpracoval a smazal dříve, než proběhne záloha.
- User Session Database (U) nesmí být ponechána sama se Scriptem (S), protože by se skript mohl pokusit o nepovolené operace a poškodit ji.

9.Kolik nejméně jednosměrných přenosů musí správce provést, aby bezpečně přesunul všechny tři komponenty na cílový server?

a) 5 přenosů
b) 7 přenosů
c) 9 přenosů
d) 11 přenosů

Body zobrazují procentuální zastoupení řádků kódu napsaných v různých programovacích jazycích v rámci softwarového projektu "Phoenix", který má celkem 200 000 řádků kódu.

- Python: 40 %
- JavaScript: 35 %
- SQL: 15 %
- Ostatní: 10 %

10.Vývojáři se rozhodli přepsat polovinu kódu z kategorie "Ostatní" do jazyka JavaScript. Kolik řádků kódu napsaných v JavaScriptu bude v projektu po této změně?

a) 70 000
b) 75 000
c) 80 000
d) 90 000

POROZUMĚNÍ TEXTU

Počítačový hardware je souhrnné označení pro veškeré fyzické vybavení počítače. Jedná se o všechny součástky, které jsou hmatatelné a viditelné. Mezi základní komponenty patří procesor (CPU), který řídí chod celého počítače a provádí většinu výpočtů, operační paměť (RAM), která slouží k dočasnému ukládání dat, se kterými počítač právě pracuje, a pevný disk (HDD nebo SSD) pro dlouhodobé ukládání dat. Dále máme grafickou kartu, která zpracovává obraz díky výpočtům, a základní desku, která všechny tyto komponenty propojuje.

11.Podle textu, která z následujících tvrzení je *pravdivá*?

a) Procesor (CPU) je jedinou komponentou, která provádí výpočty.
b) Pevný disk (HDD nebo SSD) je volitelný, zatímco grafická karta je nezbytná pro běh počítače.
c) Operační paměť (RAM) slouží pro dlouhodobé ukládání dat, zatímco pevný disk (HDD/SSD) je pouze dočasné úložiště.
d) Základní deska propojuje všechny komponenty počítače a umožňuje jejich vzájemnou komunikaci.

Zdrojový kód napsaný programátorem musí být převeden do strojového kódu, aby ho procesor mohl vykonat. K tomu slouží dva hlavní přístupy. Kompilátor (compiler) přeloží celý zdrojový kód najednou a vytvoří z něj spustitelný soubor (např. .exe). Tento soubor je pak velmi rychlý při spuštění, protože veškerý překlad již proběhl. Naopak interpret vezme zdrojový kód a překládá ho instrukci po instrukci v reálném čase při každém spuštění programu. Programy spouštěné interpretem jsou typicky pomalejší, ale jejich výhodou je snadnější přenositelnost mezi různými operačními systémy bez nutnosti nové kompilace pro každou platformu.

12.Který scénář by podle textu nejlépe ospravedlnil použití interpreta namísto kompilátoru?

a) Vývoj vysoce výkonné 3D hry, kde je klíčová maximální rychlost.
b) Tvorba programu, který se bude denně spouštět na jediném neměnném hardwaru.
c) Distribuce aplikace, která musí bez úprav fungovat na Windows, macOS i Linuxu.
d) Vytvoření ovladače pro hardwarovou komponentu, který vyžaduje přímý přístup k procesoru.

V počítačových sítích se pro přenos dat používají různé protokoly. Dva z nejzákladnějších jsou TCP (Transmission Control Protocol) a UDP (User Datagram Protocol). TCP je spojově orientovaný, což znamená, že před odesláním dat naváže spojení, kontroluje doručení segmentů, jejich správné pořadí a v případě ztráty si vyžádá jejich znovuzaslání. To zaručuje spolehlivost, ale za cenu vyšší režie a menší rychlosti. UDP naopak žádné spojení nenavazuje a datagramy jednoduše odesílá k cíli bez jakékoliv zpětné vazby. Je nespolehlivý, ale extrémně rychlý.

13.Pro jakou z následujících aplikací/situací by byl podle popsaných vlastností vhodnější protokol UDP než TCP?

a) Online bankovnictví.
b) Přenos e-mailu mezi servery.
c) Přenos dat o stavu serveru pro monitoring systém.
d) Streaming videa v reálném čase.

Lidé si lépe pamatují jména (např. www.google.com) než číselné IP adresy (např. 142.250.184.196), které počítače používají k identifikaci na síti. Systém DNS funguje jako distribuovaný telefonní seznam pro internet. Když zadáte jméno domény do prohlížeče, váš počítač se nejprve zeptá DNS serveru: "Jaká IP adresa odpovídá tomuto jménu?". DNS server najde odpovídající záznam a pošle IP adresu zpět. Teprve s touto adresou může váš prohlížeč navázat spojení se správným serverem a načíst stránku. Bez DNS by byl internet pro lidi prakticky nepoužitelný.

14.Co musí nevyhnutelně předcházet načtení webové stránky po zadání jejího jména do prohlížeče, pokud počítač nezná její IP adresu z dřívějška?

a) Přímé navázání spojení se serverem pomocí doménového jména.
b) Odeslání požadavku na DNS server za účelem překladu jména na IP adresu.
c) Restartování routeru pro obnovení síťového připojení.
d) Kontrola, zda je zadané jméno uloženo v telefonním seznamu uživatele.

Kontejnerizace je odlehčená forma virtualizace. Místo virtualizace celého hardwaru a spouštění kompletního operačního systému (jako u VM) sdílejí všechny kontejnery jádro (kernel) hostitelského operačního systému. Každý kontejner obsahuje pouze aplikaci a její závislosti (knihovny, nastavení). Důsledkem je, že kontejnery startují v řádu sekund (místo minut u VM), mají mnohem menší velikost a nižší režii. Díky této vysoké přenositelnosti je zaručeno, že aplikace poběží všude stejně, čímž se efektivně a s minimálními nároky řeší problém „na mém stroji to fungovalo“.

15.Jaký je hlavní rozdíl mezi kontejnery a virtuálními stroji (VM) v kontextu virtualizace a jaké výhody plynou z použití kontejnerů?

a) Kontejnery virtualizují celý hardware, zatímco virtuální stroje virtualizují pouze jádro hostitelského operačního systému. Kontejnery startují rychleji a mají menší velikost než VM, což zajišťuje efektivnější využití prostředků.
b) Kontejnery sdílejí jádro hostitelského systému a obsahují pouze aplikaci a její závislosti, což vede k nižší režii a rychlejšímu startu. Virtuální stroje spouštějí kompletní operační systém, což znamená větší režii a delší čas potřebný k jejich spuštění.
c) Kontejnery poskytují plnou izolaci operačního systému a aplikace, zatímco virtuální stroje sdílejí jádro hostitelského systému. Virtuální stroje startují rychleji než kontejnery, protože obsahují veškeré závislosti aplikace.
d) Kontejnery jsou určeny pouze pro testovací prostředí, zatímco virtuální stroje jsou ideální pro produkční nasazení, protože kontejnery mají větší režii při spouštění a nemohou běžet v cloudu.