CLOUD ENGINEERING

6. forduló

A kategória támogatója: IBM

RENDELKEZÉSRE ÁLLÓ IDŐ:

11:00

Ismertető a feladathoz

A hatodik lépésben folytatódik a jövőbeni fejlesztési alapok kialakítása.

Felhasznált idő: 02:03/11:00 **Elért pontszám:** 0/90

1. feladat 0/15 pont

DevOps fázisai: Kódolás (Code) - Összeépítés (Build) - Tesztelés (Test) - Telepítés (Deploy) - Monitorozás (Monitor)

Mely elemek hiányoznak a DevOps fázisai közül?

Válasz

- ○ Automatizálás (Automate) és Konfigurálás (Config)
- ○ Tervezés (Plan) és Üzemeltetés (Operate)
- ○ Üzemeltetés (Operate) és Automatizálás (Automate)
- ○ Tervezés (Plan) és Bemutatás (Showcase)

Magyarázat

A fejlesztést tervezéssel kezdjük, majd a kódolás és a kód összeépítés következik. Ezután a tesztelési fázis, majd ennek sikeressége után az alkalmazásunk telepíthető. A sort az üzemeltetés és a működés folyamatos monitorozása zárja. Monitorozás során változási igények jelenhetnek meg, minek hatására kódjavítást, fejlesztést kezdeményezhetünk és a

folyamat kezdődik újra. A legsikeresebb DevOps megvalósítás ezt a kört gyorsan végig futtatja és a fázisok egymást automatikusan meghívják.

2. feladat 0/5 pont

Melyek tipikus refaktorálást segítő eszközök?

Válaszok

- [x] ReSharper
- [ ] Sneet
- [x] Checkstyle
- [ ] Refacto
- [x] Project Analyzer
- [x] PMD
- [x] Findbugs
- [ ] Smellycode
- [x] VS Code

Magyarázat

A refaktoráló eszközök összegyűjtik a szükséges információkat, amelyek alapján el tudják végezni az összes kompenzációt a programozó helyett, s ezzel időt és munkát spórolnak meg. A modern, divatos imperatív nyelvekhez készült fejlesztői környezetek természetesen tartalmaznak refaktoráló funkciókat, ám refaktorálásra régebbi, vagy esetlegesen kevésbé elterjedt nyelvekben is szükség lehet.

3. feladat 0/10 pont

Amikor a fejlesztők új kódot vezetnek be, mindig fennáll a veszélye, hogy nem engedélyezett hozzáférést tesznek elérhetővé. Melyik eszköz csökkenti ezt a rizikót?

Válaszok

- [x] Kód felülvizsgálat (code review)
- [x] Kód tesztelés

- [ ] Hatékony patchelés
- [x] Penetrációs vizsgálat

Magyarázat

A patchelés az egyetlen, ami nem segít a fejlesztő kódjának ellenőrzésében csak bugokat javít.

4. feladat 0/10 pont

Mi(k) a jó gyakorlat(ok), hogy a saját support csapatunk hozzáférését optimalizáljuk a felhő erőforrásokhoz és szolgáltatásokhoz?

Válaszok

- [x] Használjuk a szolgáltató által adott IAM (Identity and Accesss Management) rendszert a hozzáférések korlátozására
- [ ] Használjunk megosztott felhasználói fiókot aminek minden csapattag tudja a jelszavát
- [x] Csak a minimális de elégséges jogokat osszuk ki
- [x] Ellenőrizzük a jogokat és csoportokat rendszeresen

Magyarázat

Hozzáférés managementben mindig a legfontosabb, hogy csak a minimális de elégséges jogokat osszuk ki a csapatoknak, valamint ezeket ellenőrizzük rendszeresen. Ennek a célja nem feltétlen a kontroll biztosítása, hanem hogy emberi hibából kifolyólag ne történhessen baj (erőforrások eldobása, API kulcs törlés, stb.)

Frissítés (2021.11.29.): Csak adott végpontról az IP cím tartományról lehessen elérni a felhőszolgáltató adminisztrációs felületét válaszlehetőséget a versenyzői visszajelzések alapján töröltük, mivel megfelelően lehet érvelni a helytelensége mellett is.

5. feladat 0/10 pont

Milyen megoldásokat választhatunk a futásidejű hibák kezelésére

Válaszok

- [x] Az alkalmazás megpróbálja önmagától megoldani a hibát egy előre definiált készlet alapján

- [ ] Költség / Fontosság összehasonlítása
- [x] Üzenetet küld egy meghatározott chat vagy email csoportnak a hibáról
- [x] Automatikusan létrehoz egy hibajegyet az erre létrehozott kezelő rendszerben

Magyarázat

A futásidejű hibák kezelésének számos módja ismert. Alkalmazásba épített hibakezelés (adathibák tipikus javítása), L1 support értesítése és hibajegy készítése mind jó módszer

6. feladat 0/5 pont

Valaszd ki a DevOps automatikus tesztelésben domináns szoftver megoldásait az alábbi listából.

Válaszok

- [x] Selenium
- [x] Qualibrate
- [ ] TestAutoQ
- [x] Ansible
- [x] Testcomplete
- [x] Testimony
- [ ] Gradle

Magyarázat

Az automatikus tesztelés kulcsfontosságú a kód minőségének, integrációjának megfelelőségéért. Ezáltal azonnali visszajelzés nyerhető a hibák elkerüléséért, a felhasználói elégedettség megőrzéséért.

7. feladat 0/15 pont

Egy REST API végpontokkal összekapcsolt 2 layer PaaS analitikai alkalmazásunk lehetőséget biztosít a felhasználónak valós idejű analitika futtatására a háttérben, amire a felhasználó választ vár az elkészültekor. Milyen tervezési struktúrát érdemes használni ennek a megvalósítására, hogy a felhasználó a lehető leghamarabb értesüljön a háttér folyamat eredményéről?

Válasz

- ○ Circuit breaker
- ○ Event Sourcing
- ◉ Async request-response
- ○ Queue based load-leveling

Magyarázat

Az Aszinkron kérdés-válasz design pattern biztosít lehetőséget a háttérben futó folyamat "monitorozására" egy a kérés státuszát visszaadó API végponton keresztül.

8. feladat 0/15 pont

Mely(ik) algoritmus(ok) használtak leginkább vezetőválasztásra egy elosztott rendszerben?

Válaszok

- ☐ Shuffling algoritmus
- ☑ Bully algoritmus
- ☑ Ring algoritmus
- ☐ Quicksort algoritmus

Magyarázat

Bully és Ring algoritmusok a legtöbbet használt vezetőválasztási algoritmusok. Mindkettő fault tolerant és bármennyi résztvevőre működik.

9. feladat 0/5 pont

Melyik NPM parancs installálja a legfrissebb ibm_db2 csomagot?

Válasz

- ○ npm install ibm_db2 --version=latest
- ○ npm install ibm_db2 -v latest
- ◉ npm install ibm_db2@latest

npm add ibm_db2@latest

Magyarázat

Az NPM registry név@verzió konvencióval azonosítja a csomagokat és az "install" parancsot használja

KÉSZÍTETTE

Megjelenés

Világos