DevOps (Deutsche Telekom IT Solutions)
1. forduló

Ismertető a feladathoz

Sok oka lehet annak, hogy egy szervezet érdeklődik a DevOps iránt. A legfontosabb az alapok elsajátítása ahhoz, hogy megértsük mit miért csinálunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 20 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 1 pont

A DevOps NEM:

- ( ) Kulturális mozgalom
- (•) Előre leírt, mindenre ráhúzható folyamatrendszer
- ( ) Együttműködés Development és Operations között
- ( ) Automatizálása a software delivery-nek

Magyarázat a megoldáshoz

A DevOps nem egy folyamatrendszer, hanem egy kulturális változás ami összehozza a fejlesztést és az üzemeltetést, és az automatizálásra támaszkodik

2. feladat 0 / 1 pont

Mikor érdemes a vállalatoknak érdeklődni a DevOps iránt?

- [ ] Amikor úgy tűnik, hogy az agilis gyakorlatok nem felelnek meg a szervezetnek
- [x] Amikor a Scrum és a Lean gyakorlatokat megvalósították
- [x] Ha más módszer nem adja meg a szükséges eredményeket

Magyarázat a megoldáshoz

A vállalatoknak akkor érdemes érdeklődniük a DevOps iránt, amikor az összes többi bevált módszer

a hatékonyság növelésére már nem eredményez jelentős változást. Nem kell megvárni, amíg a Scrum és a Lean gyakorlatok megvalósulnak a DevOps elindításához.

3. feladat 0 / 2 pont

Mi az alapja az agilis kiáltványnak?

- ( ) Visszacsatolás és előremenő iterációk létrehozása a munkarendszerünkbe
- ( ) A folyamatos és dinamikus tanulás kultúrájának megteremtése.
- (•) A működő szoftverek gyakori szállítása.
- ( ) Az áramlás növelése a munka láthatóvá tételével, a tételméretek és a munkaintervallumok csökkentésével, valamint a minőség növelése

Magyarázat a megoldáshoz

Ez az Agilis Kiáltvány egyik alapelve. Más elvek a szükség kicsi, önmotivált csapat, akik megbízható menedzsment modellben dolgoznak.

4. feladat 0 / 1 pont

A három út közül melyikhez tartozik a "napi munka javításának intézményesítése"?

- (•) Folyamatos tanulás és kísérletezés
- ( ) Visszajelzés
- ( ) Áramlás

Magyarázat a megoldáshoz

A Harmadik Út elve megköveteli a napi munka javítását, a lokális tanulságokat konvertálva globális tanulságokra, amelyeket az egész szervezet felhasználhat.

Vissza a kategóriáimhoz

DevOps (Deutsche Telekom IT Solutions)
2. fordulő

Ismertető a feladathoz

Fontos elemek a DevOps-ban

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 10 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 2 pont

A Done mely elemének megfelelő a DevOps számára?

- A kód a várt módon fut a fejlesztői laptopon és sikeresen teljesítette az egység teszteléseit.
- A kódot integráltár a fő ágba és automatizált egység teszteléseken mentek keresztül.
- A kód egy produktívszerű környezetben fut, és a felhasználói elfogadási teszket sikeresen teljesítette.

Magyarázat a megoldáshoz

Ez biztosítja, hogy a kód az elvárásoknak megfelelően működik, és potenciálisan release-re kész.

2. feladat 0 / 1 pont

A felhőszolgáltatók egy szánnai hadseregmajom használatával növelni kívánják szolgáltatásaik kapacitását.

Melyik majomra van szükség ebben az esetben?

- Orvosmajom
- Portás majom
- Késleltetési majom

Magyarázat a megoldáshoz

A házmester-majom biztosítja, hogy felhőkörnyezetük rendelleség és hullaszékmentes legyen;

fel nem használt erőforrásokat keres és ártalmatlanít.

3. feladat 0 / 1 pont

Az ABC vállalat a DevOps munkamódszerét alkalmazza, és egy olyan tanulási környezetet kíván támogatni, amely nyílt és hibátlan. Az ABC nemrégiben súlyos alkalmazáshibát tapasztalt, és vissza tudta állítani a alkalmazás szolgáltatás.

Mi az első feladat, amelyet el kell végezni a hibátlan poszt-mortem találkozón?

- Konstruálni a releváns eseményeik idővonalát, amint azok a fő alkalmazáshiba során történtek
- Meghatározni az ellenintézkedéseket annak megakadályozására, hogy a jövőben ismétlődjön a fő alkalmazáshiba
- Meghatározni a kiváltó okát annak, hogy az alkalmazás miért nem javasol korrekciós intézkedést, meggyőződni róla, hogy jelentős alkalmazáshiba nem forduljhat elő a jövőben
- Közzé tenni a post-mortem-et központosított helyen, ahol az egész szervezet hozzáférhet, és tanulni tud a fő alkalmazáshibából

Magyarázat a megoldáshoz

A hibátlan post-mortem találkozó első feladata a lehető legjobb rögzítése a releváns események ütemtervének megértése, amint azok bekövetkeztek. Ez magában foglalja az összes megtett és milyen időpontban (ideális esetben csevegési naplók, például IRC vagy Slack támogatják), milyen hatásokat figyeltek meg (ideális esetben a természési telemetria konkrét mutatóinak formájában, szemben a puszta szubjektív elbeszélésekről), az összes nyomon követési utat, és hogy milyen állításfogalások voltak figyelembe véve.

4. feladat 0 / 1 pont

Mi szükséges a rugalmasság megteremtéséhez a termelési kudarcok injektálásával?

- A meghibásodási mód meghatározása
- Holt-utáni értekezletek szervezése
- Az emberek képzése
- Tesztkörnyezet használata

Magyarázat a megoldáshoz

A hibamód meghatározása fontos annak biztosítása érdekében, hogy csak a hibamódok tervezetten működjenek.

5. feladat 0 / 1 pont

Mi a játéknap végrehajtásának első lépése?

- Definiálja és hajtsa végre a gyakorlatokat
- Azonosítsa és kezelje a problémákat, és teszteije
- Tervezze meg a kimaradást
- Készítse elő és szüntesse meg az SPOF-okat

Magyarázat a megoldáshoz

A lépéseknek a következőknek kell lennük:

1. Az első lépés a kimaradás megtervezése

2. A terv alapján a csapat intézkedéseket hozva előkészítheti a leállást

(3) Az intézkedés teszteléndő eljárásokat foglalhat magában

4. A kiesést a tervek szerint hajtják végre

5. A meghatározott folyamatot be kell tartani.

6. feladat 0 / 1 pont

Mi a példa a nem funkcionális követelményekre?

- Van előre és hátra kompatibilitás a verziók között
- Lehetővé kell tenni a késedelmes menetrendekről való beszámolást
- Regisztrálja a pénzügyi tranzakciókat egy szállodai foglalási rendszerben

Magyarázat a megoldáshoz

A verziók közötti előre és hátra kompatibilitás a nem funkcionális példa követelmény

7. feladat 0 / 1 pont

Melyik tevékenységet kell elvégezni az újrafelhasználható Operations felhasználói történetek sikeres létrehozásához?

- Társítsa az Operations felhasználói történeteket a fejlesztések fejlesztéséhez és hibához
- Határozza meg a tevékenységeket az átadási folyamaton belül, majd automatizálja ezeket a tevékenységeket a megfelelő eszközök és támogató munkafolyamatok használatával
- Határozza meg az összes szükséges operatív munkatevékenységet és a teljesítéshez szükséges szerepléket

Magyarázat a megoldáshoz

Ez egy olyan tevékenység, amelyet figyelembe kell venni az újrafelhasználható termékek sikeres létrehozása során

8. feladat 0 / 1 pont

Vegyük a következő elemeket:

1. Kérések módosítása

2. Telepítési csővezeték eszközök

3. Összeállított program futtatható fájlok

4. Oktatóanyagok és szabványok

Melyik két elemet tárolják általában egyetlen megosztott forráskód-tárban?

- 1 és 2
- 1 és 4
- 2 és 3
- 2 és 4

Magyarázat a megoldáshoz

Mind a telepítési folyamatok eszközei (2), mind az oktatóanyagok és a szabványok (4) az egység részét képezik

megosztott forráskód-tárnak, mivel ezek artefaktumok, amelyek tudást és tanulást kódolnak.

9. feladat 0 / 1 pont

Mi a célja a helyi felfedezések globális fejlesztésekké történő átalakításának?

- Nem csak a Dev és Ops, hanem az egész szervezet gyakorlati állapotának emelése.
- Minden új és meglévő szolgáltatás megkönnyítése a kollektív tudás kiaknázásában.
- A munkakultúra együttműködésének fokozása, valamint a rendszerek biztonságosabbá és ellenállóbbá tétele.
- Olyan kultúra megerősítése, ahol mindenki jól érzi magát és felelősségteljes.

Magyarázat a megoldáshoz

Az a cél, hogy a helyi felfedezéseket globális fejlesztésekké alakítsák át.

10. feladat 0 / 1 pont

A fejlesztők megkönnyíthetik bármely mérnök számára a naplózás és a titkosítás helyes létrehozását és használatát alkalmazásukban és környezetükben.

Melyik nem egy megosztott forráskód-tárhaz, amely ezt támogatja?

- Kódkönyvtárak és ajánlott konfigurációk
- Telepítési csomagok
- Operációs rendszer (OS) csomagok és buildek
- Titkos kezelési eszközök

Magyarázat a megoldáshoz

A telepítési csomagok inkább egy mérnök, mint egy fejlesztő által szállíthatók, ezért nem egy tétel támogatja ezt.

Vissza a kategóriámhoz

DevOps (Deutsche Telekom IT Solutions)
3. forduló

Ismertető a feladathoz

Harmadik forduló

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 20 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Mi a különbség a systems of engagement (SoE) és a systems of record (SoR) között?

- A SoE és az SoR jellemzően azonos változási ütemű.
- A SoE jellemzően sokkal gyorsabb változási ütemű, mint a SoR.
- A SoE jellemzően sokkal alacsonyabb változási ütemű, mint a SoR.
- Ez a kapcsolat információs rendszerenként eltérő.

Magyarázat a megoldáshoz

A SoE változásának üteme nagyobb, mivel a felhasználói felülettet képviseli.

2. feladat 0 / 2 pont

Mi az előnye annak, ha a fejlesztés és az üzemeltetés közös eszközöket használ?

- Egységes backlog, ahol mindenki a fejlesztési projekteket helyezi előtérbe globális szempontok alapján.
- A fejlesztők visszajelzést kapnak arról, hogyan teljesítenek alkalmazásaik a produktív környezetben.
- Lehetővé teszi a csapat számára, hogy a szokásos munkaidőben telepítéseket hajtson végre, és egyszerűek legyenek az átállások.
- Az üzemeltetés tudás átalakítása automatizált kóddá, amely sokkal megbízhatóbb és szélesebb körű lehet és újrafelhasználható.

Magyarázat a megoldáshoz

Egységes backlog mindenkinek, globális szempontból kiemelten kezeli a fejlesztési projekteket. Olyan munka kiválasztása, amely a szervezet számára a legnagyobb értéket képviseli, vagy a legnagyobb hatással van a szervezet technikai adósság csökkentésére.

3. feladat 0 / 2 pont

"Több piacorientált eredmény lehet elérni, ha az üzemeltetési tudás integrálva van a fejlesztő csapatokban, azokat hatékonyabbá és produktívabbá téve."

Melyik megközelítés valósítaná meg ezt a legjobban? Hogyan integrálják az üzemeltetési képességeket a napi munkába?

- Fejlesztői kapcsolattartó rendelése az üzemeltetés csapathoz
- Önkiszolgáló infrastruktúra létrehozása
- Az üzemeltetéscsapat kiszervezése
- Fejlesztők képzése az üzemeltetés munkájának elvégzésére

Magyarázat a megoldáshoz

Ez az üzemeltetés-fejlesztés integrálásának három átfogó stratégiájának egyike:

• Hozzon létre önkiszolgáló képességeket, hogy a kiszolgáló csapatok fejlesztői eredményesek lehessenek.

• Az üzemeltetési mérnököket be kell ágyazni a szervezcsapatokba.

• üzemeltetés összekötőit rendelje a szervizcsapatokhoz, ha az üzemeltetés beágyazása nem lehetséges.

4. feladat 0 / 1 pont

Mely követelményeket kell a DevOps csapatnak figyelembe vennie egy backlog elem kidolgozásakor?

- A nem funkcionális és a funkcionális követelményeket
- Sem a nem funkcionális, sem a funkcionális követelményeket
- Csak a funkcionális követelményeket
- Csak a nem funkcionális követelményeket

Magyarázat a megoldáshoz

Mind a nem funkcionális és a funkcionális követelményeket figyelembe kell venni.

5. feladat 0 / 2 pont

DevOps-ban mi a megfelelő "Definition of done"?

- A követelmény kész, amikor a sikeres build történik.
- A követelmény kész, amikor sikeresen tesztelték.
- A követelmény kész, amikor a terméket elfogadják.
- A követelmény kész, amikor a termék már a produktív környezetre kerül és használható.

Magyarázat a megoldáshoz

A Done DevOpsban jó leírása, akkor teljesül, amikor valamilyen értéket teremt a felhasználónak, ha már a termék produktív környezetben van.

6. feladat 0 / 2 pont

Miért fontos az „value stream" (értékfolyam) fogalom a DevOps-ban?

- Az értékfolyam segít az kollégáknak a napi feladataik megismerésében és megértésében.
- Az értékfolyam segít elemezni a jelenlegi állapotot és valamelyik metrika javítás kísérletet.
- Az értékfolyam segít azonosítani, mikor valási elvégzete a munkájának egy részét.
- Az értékfolyam segít a sima és egyenletes "flow"(áramlás) megvalósítását az összes folyamatlépésen keresztül.

Magyarázat a megoldáshoz

Az értékfolyam koncepciója biztosítja az folyamatos és egyenletes és egységes áramlást az egyik lépésről a másikra.

7. feladat 0 / 2 pont

Melyik hasonlat írja le a legjobban az „installation pipeline"-t ?

- [x] Egy futószalag, például egy autógyárban
- [x] Modern processzorok, amelyek párhuzamos csővezetékeket használnak
- [ ] Több futószalag használat
- [ ] Egy csapat emberek különböző munkákra történő beosztási folyamat

Magyarázat a megoldáshoz

Humble és Farley amikor kitalálták a kifejezést, a csővezetékezés gondolatát használták fel a modern processzorok architektúrájából, amely sokkal gyorsabb eredmények elérését teszi lehetővé.

8. feladat 0 / 2 pont

Amikor a fejlesztők kódot vezetnek be, mindig fennáll annak a veszélye, hogy engedélyezik az illetéktelen hozzáférést.

Melyik "szűrő" NEM enyhíti ezt a kockázatot?

- Kódellenőrzés
- Kód tesztelés
- Hatékony "patch"-elés
- Penetration tesztelés

Magyarázat a megoldáshoz

A hatékony "patch"-elés nem fedi fel a fejlesztő kódjának bevezetését, csak megoldja hibákat.

9. feladat 0 / 2 pont

Mi a példa a telemetria létrehozására egy alkalmazásban?

- Az operációs rendszer (OS) változásai
- A rendszernaplók napi felülvizsgálata
- Változások a Security group-okban
- A felhasználói jelszó visszaállítása

Magyarázat a megoldáshoz

Ez az alkalmazásra jellemző telemetria használata, a többi viszont az infrastruktúrán történik.

10. feladat 0 / 2 pont

Melyik folyamat szolgál elsődleges vezérlőként az üzemeltetési és a biztonsági kockázatok csökkentésére, és támogatja a megfelelési követelményeket?

- Változáskezelési (Change management) folyamat
- Konfigurációkezelési (Configuration management) folyamat
- Kiadás- és telepítéskezelési (Release and deployment) folyamat
- Szolgáltatásszint-menedzsment (Service level management) folyamat

Magyarázat a megoldáshoz

Szinte minden informatikai szervezet meglévő változáskezeléssel rendelkezik, amelyek az elsődleges ellenőrzések az üzemeltetés és a biztonsági kockázatok csökkentésére szolgálnak. A biztonsági menedzserek a változáskezelési folyamatokra támaszkodnak a megfelelési követelmények szempontjából,és általában bizonyítékot igényelnek arra vonatkozóan, hogy minden változtatást megfelelően engedélyeztek.

Vissza a kategórialmhoz

DevOps (Deutsche Telekom IT Solutions)
4. forduló

Ismertető a feladathoz

Negyedik forduló

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 10 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Egy banknak hosszabb időre van szüksége az új ajánlatának piacra dobásához, az üzemeltetés pedig csak később tudja az új informatika rendszereket hozzá létrehozni. Mi igaz az egyes környezeteknek automatizált létrehozásával kapcsolatban (éles, teszt, fejlesztői, stb.)?

- (•) Az automatizáció minden környezetben használható.
- ( ) Az automatizáció a legtöbb környezetben használható, kivéve az éles környezetet, ahol biztonsági szempontok miatt el kell térni tőle.
- ( ) A DevOps megköveteli a "négy szem elve" (Four-Eyes Principle) használatát, így az üzemeltetés minden környezetet - különösen az éles környezetet - felülvizsgál.
- ( ) A DevOps megköveteli, hogy az éles környezet létrehozásánál - a többi környezet sikeres (test-success) automatizációját követően- a megfelelő lépéseket (Task/Step) manuálisan, egyesével ellenőrzötten kelljen végrehajtani.

Magyarázat a megoldáshoz

DevOps alapelv (Continuous Deployment -CD) során a telepítéshez nincs szükség manuális műveletekre, mert megfelelő tesztelést (Continous Integration) követően az élesítés során minden lépés automatizált scriptekkel valósul meg.

Ez bármelyik környezetre létrehozásához igaz.

2. feladat 0 / 2 pont

Az üzleti kockázatok minimalizálása érdekében, miért fontos a verziózás (versioncontrol)?

- ( ) Ha a környezet a kívánt verziótól eltér, legyen lehetőség azonnal riasztást generálni
- ( ) A tesztkörnyezet korábbi -minél hamarabbi- helyreállításához, így folytatható az éles környezetet megelőző tesztelés
- (•) A különböző verziók közötti előre- és visszalépés lehetőségéhez a teljes éles környezet újrateremthetőségének biztosítása mellett
- ( ) A forráskód megosztásának képességéhez, ami a különböző fejlesztői csapatok között hatékonyabb munkavégzést eredményez, ezáltal a fejlesztés felgyorsul

Magyarázat a megoldáshoz

A verziókontrollnak lehetővé kell tennie az éles környezetben használt verzió újbóli létrehozását ezáltal lehet csökkenteni az üzleti kockázatot (pl. katasztrófahelyzetben).

3. feladat 0 / 1 pont

Az automatizált tesztek célja, hogy a hibákat a lehető legkorábban megtaláljuk. A tesztelési piramis azt mutatja, hogy az automatizált tesztelést milyen sorrendben kell elvégezni. Melyik tesztet kell elvégezni először?

- ( ) Komponens tesztelés (Component testing)
- ( ) Elfogadói tesztelés (User Acceptance Test)
- ( ) Integrációs tesztelés (Integration testing)
- (•) Egységtesztelés (Unit testing)
- ( ) Feketedobozos tesztelés (BlackBox testing)
- ( ) Függvénytesztelés (Function testing)

Magyarázat a megoldáshoz

Az egységteszt (unittest), amelyet már a programozás korai szakaszában automatizálni érdemes (pl. TDD – Test Driver Development vagy BDD – Behaviour Driven Development).

4. feladat 0 / 2 pont

A DevOps csapat a tesztvezérelt fejlesztéssel (TDD) kívánja növelni a sebességét, úgy hogy az előző kérdésben szereplő tesztelés automatizációját választjuk.
Melyik a helyes TDD szemlélet?

- ( ) 1. Program átalakítások (Refactoring)
  2. Teszt esetek létrehozása
  3. Új modul megírása
- ( ) 1. Új modul megírása
  2. Minél több teszteset létrehozása
  3. Modul átalakítása (Refactoring)
- (•) 1. Teszteset létrehozása
  2. Új kódrészlet megírása
  3. Kódrész átalakítások (Refactoring)
- ( ) 1. Jellemzők létrehozása (Feature)
  2. Forgatókönyv megírása (Scenario)
  3. Új kódrész megírása
  4. Kódrész átalakítása (Refactoring)

Magyarázat a megoldáshoz

A tesztvezérelt fejlesztés (Test Driven Development (TDD) modell szerint először megírjuk a teszt esetet,mely "elhasal" (falling), majd a kódot addig készítjük, míg sikeres nem lesz, ezt követően a kódrész minőségi átalakítása történik (refactoring).

5. feladat 0 / 2 pont

Milyen módon lehet a legjobban automatizálni a környezetek létrehozását és megfelelő beállítását?

- ( ) ITIL alapú jegykezelő (change management) rendszer használatával a fejlesztői, teszt vagy elfogadási (UAT) környezet létrehozásához
- ( ) Olyan eszközzel, ami az éles környezetet lemásolja (klónozza) a fejlesztői, teszt és elfogadási (UAT) környezetnek
- ( ) Saját -adott környezeti- konfigurációs fájlok létrehozásával, majd a környezetek szinkronban tartásával
- (•) "Infrastruktúra mint kód"-dal (IaC), ami lehetővé teszi tetszőleges környezet deklaratív / imperatív létrehozását

Magyarázat a megoldáshoz

Az infrastruktúrát, mint kódot bárki felhasználhatja a környezetek felépítésére és konfigurálására (https://en.wikipedia.org/wiki/Infrastructure_as_code).

Vissza a kategóriáimhoz

DevOps (Deutsche Telekom IT Solutions)
5. forduló

Ismertető a feladathoz

Ötödik forduló

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 20 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 2 pont

Az alábbiak közül melyek támogatják a CI (continuous integration / folyamatos integráció) elveinek való megfelelést?

- [ ] feature branch használata hosszú fejlesztésekhez
- [x] SVN verziókezelő használata
- [ ] a fő ágon (main branch) csak a vezető fejlesztő végezhet módosításokat
- [x] GIT verziókezelő használata
- [ ] code freeze alkalmazása élesítés előtt

Magyarázat a megoldáshoz
A CI alapelve, hogy naponta többször integráljunk a verziókezelő fő ágával (main branch). A verziókezelő használata ezt értelemszerűen támogatja, a többi akadályozza.

2. feladat 0 / 2 pont

Ki üzemeltethet éles környezetben egy microservicet a DevOps elvek szerint?

- ( ) kizárólag egy dedikált csapat, amelyik csak üzemeltetéssel foglalkozik
- ( ) egy microservicet nem kell üzemeltetni, mert mindent biztosít a keretrendszer
- ( ) minden nap felváltva egy fejlesztő kötelezően vállalja az üzemeltetési feladatokat is
- (•) ugyanaz a csapat fejlesztheti és üzemeltetheti a microservicet

Magyarázat a megoldáshoz
Egy microservicet fejleszthet és üzemeltethet ugyanaz a csapat. A többi válasz ellentétes a DevOps elveivel (fejlesztés és üzemeltetés hatékony társítása).

3. feladat 0 / 2 pont

Melyik eszköz alkalmas arra önmagában, hogy **deklaratív IaC** (infrastructure as code) módszertannal írjuk le a szükséges infrastruktúrát?

- (•) Terraform
- ( ) Chef
- ( ) Ansible
- ( ) Molecule
- ( ) Jest

Magyarázat a megoldáshoz
A Chef és az Ansible elsősorban procedurális logikát alkalmaz. A Molecule és Jest teljesen másra való.

4. feladat 0 / 2 pont

Lehetséges-e használni az infrastructure as code eszközeit nem virtualizált (pl. bare metal cloud vagy saját adatközpont) környezetben?

- (•) igen
- ( ) nem, mivel automatizáltan nem lehetséges új fizikai szervert deklarálni
- ( ) nem, mivel nem lehetséges a fizikai gépek közvetlen kezelése (pl. elindítás, leállítás)
- ( ) néhány eszközt lehet (pl. puppet), de a többségét nem (pl. terraform)

Magyarázat a megoldáshoz
Semmi akadálya. Jó példa erre a Metalcloud (Terraform provider).

5. feladat 0 / 2 pont

A Continous Delivery (folyamatos szállítás) alkalmazása során megengedett-e a nem automatizált (azaz manuális) tesztek használata?

- ( ) nem, mivel ez ellentmond a folyamatosság követelményének
- (•) igen
- ( ) csak olyan esetekben, amikor nagyon költséges lenne az automatizálás
- ( ) nem, mivel így a termék automatikusan nem szállítható

Magyarázat a megoldáshoz
Fontos az automatizálás, de a Continuos Delivery elveivel egyáltalán nem ellentétes a manuális tesztelés alkalmazása.

6. feladat 0 / 4 pont

A következő kérdés az alábbi Terraform (HCL) kódra vonatkozik:

```
resource "kubernetes_replication_controller" "example" {
  metadata {
    name = "terraform-nginx-example"
    labels = {
      App = "TerraformNginxExample"
    }
  }

  spec {
    selector = {
      App = "TerraformNginxExample"
    }
    template {
      container {
        image = "nginx:${var.nginx_version}"
        name  = "example"

        port {
          container_port = 80
        }

        liveness_probe {
          http_get {
            path = "/index.html"
            port = 80
          }
          initial_delay_seconds = 30
          timeout_seconds       = 1
        }

        resources {
          limits {
            cpu    = "0.5"
            memory = "512Mi"
          }
          requests {
            cpu    = "250m"
            memory = "50Mi"
          }
        }
      }
    }
  }
}
```

Mi történik, ha a leírt container több memóriát igényelne, mint a limits alatt megadott (512Mi) határ?

- [ ] a meglévő container továbbra is futni fog, de új pod kerül elindításra, hogy ki tudja szolgálni a megnövekedett terhelést
- [ ] az adminisztrátor riasztást kap emailben
- [x] nem fog további memóriát kapni, így az alkalmazásban hibák keletkezhetnek (pl. OutOfMemoryError)
- [x] a containert leállítja a keretrendszer
- [ ] a container automatikusan lassítva lesz

Magyarázat a megoldáshoz
A memóriahasználatot nem tudja korlátozni a kubernetes, így az egyetlen megoldás az, hogy leállítja. Dokumentáció: https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs/resources/replication_controller

7. feladat 0 / 4 pont

Mi történik, ha a leírt container több processzort igényelne, mint a requests alatt megadott (250m) határ?

```
resource "kubernetes_replication_controller" "example" {
  metadata {
    name = "terraform-nginx-example"
    labels = {
      App = "TerraformNginxExample"
    }
  }

  spec {
    selector = {
      App = "TerraformNginxExample"
    }
    template {
      container {
        image = "nginx:${var.nginx_version}"
        name  = "example"

        port {
          container_port = 80
        }

        liveness_probe {
          http_get {
            path = "/index.html"
            port = 80
          }
          initial_delay_seconds = 30
          timeout_seconds       = 1
        }

        resources {
          limits {
            cpu    = "0.5"
            memory = "512Mi"
          }
          requests {
            cpu    = "250m"
            memory = "50Mi"
          }
        }
      }
    }
  }
}
```

- (•) amíg nem lépi át a limitsben megadott határt (0.5), a container megkapja a szükséges erőforrást
- ( ) az adminisztrátor riasztást fog kapni emailben
- ( ) a meglévő container továbbra is futni fog, de új pod kerül elindításra, hogy ki tudja szolgálni a megnövekedett terhelést
- ( ) a containert leállítja a keretrendszer
- ( ) a container automatikusan lassítva lesz

Magyarázat a megoldáshoz
A requests azt jelenti, hogy olyan gépen indítható el, ahol legalább ennyi erőforrás elérhető. Nincs akadálya, hogy ennél többet vegyen igénybe. Dokumentáció: https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs/resources/replication_controller

8. feladat 0 / 4 pont

Miért nem szabad az initial_delay_seconds értéket túl alacsonyra állítani?

```
resource "kubernetes_replication_controller" "example" {
  metadata {
    name = "terraform-nginx-example"
    labels = {
      App = "TerraformNginxExample"
    }
  }

  spec {
    selector = {
      App = "TerraformNginxExample"
    }
    template {
      container {
        image = "nginx:${var.nginx_version}"
        name  = "example"

        port {
          container_port = 80
        }

        liveness_probe {
          http_get {
            path = "/index.html"
            port = 80
          }
          initial_delay_seconds = 30
          timeout_seconds       = 1
        }

        resources {
          limits {
            cpu    = "0.5"
            memory = "512Mi"
          }
          requests {
            cpu    = "250m"
            memory = "50Mi"
          }
        }
      }
    }
  }
}
```

- ( ) induláskor hirtelen nagyon sok container fog párhuzamosan elindulni, mivel egyiket sem látja majd működőnek a keretrendszer
- (•) egy container sem fog sikeresen elindulni, mivel már azelőtt leállításra kerül, mielőtt elindulna
- ( ) a container a szükségesnél több kapacitást fog kapni, hogy gyorsabban el tudjon indulni
- ( ) nincs jelentősége, legfeljebb email riasztásokat fog eredményezni
- ( ) a container automatikusan lassítva lesz és ezért még lassabban fog tudni elindulni

Magyarázat a megoldáshoz
Ha a liveness feltétel nem teljesül, akkor a container automatikusan leállításra kerül. Dokumentáció: https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs/resources/replication_controller

Vissza a kategóriáimhoz

DevOps (Deutsche Telekom IT Solutions)
6. forduló

Ismertető a feladathoz

Hatodik forduló

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 20 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Hogyan tudjuk hatékonyan biztosítani, hogy az infrastruktúrát leíró kód (infrastructure as code) megfelel a csapatban elfogadott konvencióknak?

- [x] code review segítségével
- [ ] ezt nem lehet, de nem is szükséges biztosítani, mert a kód futtatásakor úgyis minden hiba kiderül
- [x] az adott eszközre kifejlesztett kódellenőrzőkkel (pl. tflint, ansible lint)
- [ ] bevezetjük a köztes kódra (bytecode) fordítást
- [ ] reguláris kifejezés alapú Linux parancsokkal (pl. grep, sed, awk)

Magyarázat a megoldáshoz

a megadottak közül csak a code review és a céleszközök használata hatékony megoldás

2. feladat 0 / 3 pont

DevOps környezetben gyakran hallani a „Cattle not Pets” (jószág, nem házikedvenc) kifejezést. Mire utal ez?

- ○ a kódot házikedvencként kell gondozni, hogy karbantartható maradjon
- ○ bármikor szükség lehet a szoftver teljes manuális tesztelésére
- ○ minden szerver párban fusson a magas rendelkezésre állás érdekében
- ○ többet ér egy jó csapat átlagos képességű tagokkal, mint egyetlen kiemelkedő fejlesztő
- ◉ a szerverek legyenek könnyen eldobható erőforrások

Magyarázat a megoldáshoz

https://www.theregister.com/2013/03/18/servers_pets_or_cattle_cern/

3. feladat 0 / 3 pont

A git-flow modell az alábbiak közül milyen ágakat (branch) használ?

- [x] develop
- [ ] archive
- [ ] squash
- [ ] final
- [x] hotfix
- [ ] review

Magyarázat a megoldáshoz

https://nvie.com/posts/a-successful-git-branching-model/

4. feladat 0 / 3 pont

Dockerfile írásakor az ADD és a COPY utasítások közül melyiket érdemes használni?

- ○ minden esetben az ADD utasítást célszerű használni, mert minden tekintetben többet tud, mint a COPY
- ○ a két utasítás ugyanazt csinálja, bármelyik használható
- ◉ a COPY használata jobban átlátható, így általában a COPY használata javasolt
- ○ a két utasítást általában párban célszerű használni, mert a lemásolt fájlt hozzá is kell adni a docker containerhez

Magyarázat a megoldáshoz

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#add-or-copy

5. feladat 0 / 3 pont

Terraform használata esetén hogyan lehet kikényszeríteni azt, hogy egy változások nélküli erőforrás törlődjön, és újra létrejöjjön, amikor legközelebb alkalmazzuk a módosításokat?

- ○ nem lehetséges, mivel a terraform csak a változtatásokat fogja elvégezni
- ○ ez csak úgy megoldható, ha valamilyen apró módosítást végzünk a fájlban (pl. szóköz hozzáadása sor végére)
- ○ terraform apply -recreate=true használatával
- ◉ terraform taint parancs használatával
- ○ terraform force-apply használatával

Magyarázat a megoldáshoz

https://www.terraform.io/docs/commands/taint.html

6. feladat 0 / 5 pont

Melyek azok a javítások, amelyek a hatékonyságot egyértelműen növelik a jelenlegi megoldáshoz képest?

```
ARG DATE

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE} AS build
ENV ZABBIX_BUILDER_ROOT /build
ENV ZABBIX_SOURCE_ROOT /build/deps/zabbix-4.0.1
ENV BUILD_DIST /build/dist
ENV JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_
WORKDIR /build
COPY deps deps
COPY ./scripts/zabbix-base-setup.sh .
RUN ./zabbix-base-setup.sh
COPY scripts/zabbix-java_gateway-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-server_agent-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-frontend-builder.sh $ZABBIX_BUILDER_ROOT/scripts/
RUN mkdir -p ${BUILD_DIST} && \
    mkdir -p ${ZABBIX_SOURCE_ROOT} && \
    tar -C /build/deps -zxvf /build/deps/zabbix-4.0.1.tar.gz && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-server_agent-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-java_gateway-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-frontend-builder.sh

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE}
ENV ORACLE_HOME /usr/lib/oracle/11.2/client64
ENV LD_LIBRARY_PATH /usr/lib/oracle/11.2/client64/lib
LABEL info="Zabbix Server/Frontend and Java gateway"
WORKDIR /zabbix
COPY --from=build /build/dist/frontend frontend
COPY --from=build /build/dist/bin bin
COPY --from=build /build/dist/sbin/zabbix_server sbin/
COPY --from=build /build/dist/sbin/zabbix_java sbin/
COPY --from=build /build/dist/oci8.so .
COPY ./configs/php.ini .
COPY ./deps/oracle-instantclient11.2-basic-11.2.0.4.0-1.x86_64.rpm deps/
COPY ./scripts/zabbix/zabbix-setup.sh .
COPY ./scripts/zabbix/zabbix-startup.sh /usr/bin/
RUN ./zabbix-setup.sh
RUN groupadd --gid 1571 PUBLIC
RUN useradd --uid 1001 --gid 1001 -G PUBLIC zabbix
EXPOSE 80
EXPOSE 10051
EXPOSE 10052
VOLUME [ "/zabbix/externalscripts", "/zabbix/logs", "/zabbix/configs" ]
STOPSIGNAL SIGTERM
ENTRYPOINT [ "zabbix-startup.sh" ]
```

- ○ multi-stage buildre való átírás
- ○ a leghosszabb RUN parancs feldarabolása
- ○ a második FROM utasítás törlése
- ◉ logikailag összefüggő COPY és RUN utasítások összevonása
- ○ ENTRYPOINT átírása CMD-re

Magyarázat a megoldáshoz

Ez már eleve egy multi-stage build. További forrás: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

7. feladat 0 / 5 pont

Mit jelent az EXPOSE 80 utasítás?

```
ARG DATE

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE} AS build
ENV ZABBIX_BUILDER_ROOT /build
ENV ZABBIX_SOURCE_ROOT /build/deps/zabbix-4.0.1
ENV BUILD_DIST /build/dist
ENV JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_
WORKDIR /build
COPY deps deps
COPY ./scripts/zabbix-base-setup.sh .
RUN ./zabbix-base-setup.sh
COPY scripts/zabbix-java_gateway-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-server_agent-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-frontend-builder.sh $ZABBIX_BUILDER_ROOT/scripts/
RUN mkdir -p ${BUILD_DIST} && \
    mkdir -p ${ZABBIX_SOURCE_ROOT} && \
    tar -C /build/deps -zxvf /build/deps/zabbix-4.0.1.tar.gz && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-server_agent-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-java_gateway-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-frontend-builder.sh

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE}
ENV ORACLE_HOME /usr/lib/oracle/11.2/client64
ENV LD_LIBRARY_PATH /usr/lib/oracle/11.2/client64/lib
LABEL info="Zabbix Server/Frontend and Java gateway"
WORKDIR /zabbix
COPY --from=build /build/dist/frontend frontend
COPY --from=build /build/dist/bin bin
COPY --from=build /build/dist/sbin/zabbix_server sbin/
COPY --from=build /build/dist/sbin/zabbix_java sbin/
COPY --from=build /build/dist/oci8.so .
COPY ./configs/php.ini .
COPY ./deps/oracle-instantclient11.2-basic-11.2.0.4.0-1.x86_64.rpm deps/
COPY ./scripts/zabbix/zabbix-setup.sh .
COPY ./scripts/zabbix/zabbix-startup.sh /usr/bin/
RUN ./zabbix-setup.sh
RUN groupadd --gid 1571 PUBLIC
RUN useradd --uid 1001 --gid 1001 -G PUBLIC zabbix
EXPOSE 80
EXPOSE 10051
EXPOSE 10052
VOLUME [ "/zabbix/externalscripts", "/zabbix/logs", "/zabbix/configs" ]
STOPSIGNAL SIGTERM
ENTRYPOINT [ "zabbix-startup.sh" ]
```

- ○ a host gép 80-as portján elérhető lesz egy szolgáltatás TCP protokollon
- ○ a host gép 80-as portján elérhető lesz egy szolgáltatás TCP és UDP protokollokon
- ◉ a container futásidőben nyitva tartja a 80-as portot TCP protokoll számára
- ○ a container építése során a 80-as TCP és UDP port nyitva lesz
- ○ a container eléri a host gépet a 80-as TCP porton

Magyarázat a megoldáshoz

https://docs.docker.com/engine/reference/builder/#expose

8. feladat 0 / 5 pont

Milyen felhasználóval fog futni a zabbix-startup.sh szkript, ha az indítás során nem adunk meg felhasználót?

```
ARG DATE

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE} AS build
ENV ZABBIX_BUILDER_ROOT /build
ENV ZABBIX_SOURCE_ROOT /build/deps/zabbix-4.0.1
ENV BUILD_DIST /build/dist
ENV JAVA_HOME /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_
WORKDIR /build
COPY deps deps
COPY ./scripts/zabbix-base-setup.sh .
RUN ./zabbix-base-setup.sh
COPY scripts/zabbix-java_gateway-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-server_agent-builder.sh $ZABBIX_BUILDER_ROOT/scripts
COPY scripts/zabbix-frontend-builder.sh $ZABBIX_BUILDER_ROOT/scripts/
RUN mkdir -p ${BUILD_DIST} && \
    mkdir -p ${ZABBIX_SOURCE_ROOT} && \
    tar -C /build/deps -zxvf /build/deps/zabbix-4.0.1.tar.gz && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-server_agent-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-java_gateway-builder.sh && \
    $ZABBIX_BUILDER_ROOT/scripts/zabbix-frontend-builder.sh

FROM docker-registry.ithon.hu:443/ithon-ol7:${DATE}
ENV ORACLE_HOME /usr/lib/oracle/11.2/client64
ENV LD_LIBRARY_PATH /usr/lib/oracle/11.2/client64/lib
LABEL info="Zabbix Server/Frontend and Java gateway"
WORKDIR /zabbix
COPY --from=build /build/dist/frontend frontend
COPY --from=build /build/dist/bin bin
COPY --from=build /build/dist/sbin/zabbix_server sbin/
COPY --from=build /build/dist/sbin/zabbix_java sbin/
COPY --from=build /build/dist/oci8.so .
COPY ./configs/php.ini .
COPY ./deps/oracle-instantclient11.2-basic-11.2.0.4.0-1.x86_64.rpm deps/
COPY ./scripts/zabbix/zabbix-setup.sh .
COPY ./scripts/zabbix/zabbix-startup.sh /usr/bin/
RUN ./zabbix-setup.sh
RUN groupadd --gid 1571 PUBLIC
RUN useradd --uid 1001 --gid 1001 -G PUBLIC zabbix
EXPOSE 80
EXPOSE 10051
EXPOSE 10052
VOLUME [ "/zabbix/externalscripts", "/zabbix/logs", "/zabbix/configs" ]
STOPSIGNAL SIGTERM
ENTRYPOINT [ "zabbix-startup.sh" ]
```

- ○ zabbix (id: 1001)
- ○ zabbix (id: 1571)
- ○ root (id: 0)
- ○ nobody (id: 65534)
- ◉ a bemutatott kód alapján ezt nem lehet eldönteni

Magyarázat a megoldáshoz

nem lehet eldönteni, mert nem ismerjük a FROM image tartalmát (ithon-ol7).

Vissza a kategóriáimhoz

DevOps (Deutsche Telekom IT Solutions)
7. forduló

Ismertető a feladathoz

Hetedik forduló

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 20 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 2 pont

Melyik kubernetes minta segíthet elérhetővé tenni egy alkalmazás naplóseményeit anélkül, hogy az alkalmazást (containert) módosítanánk?

- [ ] Nagykövet (ambassador) minta
- [x] Oldalkocsi (sidecar) minta
- [ ] Megfigyelő (observer) minta
- [ ] Többfázis (multi-stage) minta

Magyarázat a megoldáshoz

Csak a nagykövet (ambassador) és az oldalkocsi (sidecar) létező kubernetes minták. A kettő közül csak az oldalkocsi (sidecar) segíthet ebben az esetben. További forrás: https://stackoverflow.com/questions/35451056/differences-between-sidecar-and-ambassador-and-adapter-pattern

2. feladat 0 / 3 pont

A szünetmentes tápforrás karbantartása során a szerelő olyan szerencsétlenül ejtette el a csavarhúzót, hogy túlfeszültség miatt leégett a szerverünk és sajnos az adathordozók is menthetetlenül károsodtak. Az utolsó biztonsági mentés hat órával az incidens előtt készült, melynek segítségével két óra alatt sikerült újra működésre bírni a rendszert egy másik adatközpontban. Tovább egy órát igényelt, hogy leteszteljük az új rendszert élesítés előtt, így a felhasználók csak az incidens után három órával tudták újra használni az alkalmazásainkat. Milyen RTO (Recovery Time Objective) és RPO (Recovery Point Objective) teljesülhetett ebben az esetben?

- [ ] RTO: 8 óra, RPO: 3 óra
- [x] RTO: 6 óra, RPO: 24 óra
- [ ] RTO: 6 óra, RPO: 2 óra
- [ ] RTO: 6 óra, RPO: 3 óra

Magyarázat a megoldáshoz

A cél akkor is teljesül értelmezésünken, ha rövidebb időt sikerül elérni a kitűzöttnél. Az RPO legkisebb értéke hat óra lehet, mivel az volt az utolsó biztonsági mentés ideje. Az RTO legkisebb ideje három óra, mert ennyi idő alatt sikerült élesíteni a javított rendszert. Ezeknek a feltételeknek csak egy válaszlehetőség felelt meg.

3. feladat 0 / 3 pont

A csapatunk szeretne teljes mértékben megfelelni a CD (Continuous Delivery / Folyamatos szállítás) módszertannak. Mi történik akkor, ha a verziókezelő fő ágán (main branch) lévő kód tesztfejlesztése alacsonyabb a csapat által elfogadott minimumnál?

- [ ] A kód ebben a formájában nem kiadható, de ez nem is probléma, ha nem akarunk élesíteni a következő napokban.
- [ ] A kód így is kiadható, ettől még működőképes tökéletesen.
- [ ] A CD folyamat (pipeline) gondoskodik arról, hogy a hibás módosítás törlésre kerüljön.
- [ ] Az utolsó módosítást végző csapattag feladata, hogy azonnal kijavítsa a hibát (adódig nem mehet hazá).
- [x] A csapat a lehető leggyorsabban gondoskodik arról, hogy a kód újra kiadható legyen.

Magyarázat a megoldáshoz

A CD módszertan szerint a kód kiadhatóságáról szóló visszajelzés kell gyorsan a csapat tudomására jusson, hogy minél hamarabb javítani lehessen a hibákat és a kódot újra kiadhatóvá tegyék amint lehetséges.

4. feladat 0 / 3 pont

Hogyan lehet git verziókezelő használata esetén teljesen üres mappát elmenteni?

- [ ] git add és git commit parancsok szokásos használatával
- [ ] a git add parancs erre kitalált paraméterével: --include-empty
- [ ] a git commit parancs erre kitalált paraméterével: --include-empty
- [ ] egy fájlt el kell menteni a mappával együtt, majd a fájl törlése után a mappa megmarad a verziókezelőben
- [x] nem lehetséges

Magyarázat a megoldáshoz

https://git.wiki.kernel.org/index.php/GitFaq#Can_I_add_empty_directories.3F

5. feladat 0 / 3 pont

Az alábbiak közül melyek elfogadható tárolók (külön titkosítás nélküli) jelszavak számára?

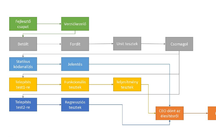
- [ ] git verziókezelő
- [ ] Dockerfile
- [x] Kubernetes Secrets
- [x] AWS Secrets Manager
- [x] Jenkins
- [ ] Puppet SafeStore
- [ ] Terraform kód
- [ ] Azure Pipelines Secrets

Magyarázat a megoldáshoz

A jelszavakat értelemszerűen a kódtól elkülönítetten szükséges tárolni. Az erre kitalált dedikált szolgáltatások elfogadhatók: Kubernetes Secrets, AWS Secrets Manager és Jenkins ide vonatkozó funkciója. Nem létező eszközök: Puppet SafeStore, Azure Pipelines Secrets (helyesen Azure Key Vault)

6. feladat 0 / 5 pont

Kizárólag az ábrára hagyatkozva az alábbiak közül miket valósíthat meg ez a folyamat?



- [x] Folyamatos integráció (Continuous Integration)
- [x] Folyamatos szállítás (Continuous Delivery)
- [ ] Folyamatos élesítés (Continuous Deployment)
- [ ] Egyiket sem valósíthatja meg a többi válasz közül

Magyarázat a megoldáshoz

7. feladat 0 / 5 pont

Hogyan érdemes biztosítani azt, hogy ugyanaz a termék mentén végig minden ellenőrző állomáson, mint amelyik élesítésre fog kerülni?

- [ ] az utolsó módosítás időpontja alapján azonosítjuk a kódot, majd ezt külön fordítjuk és telepítjük minden környezetben
- [ ] mindig a vezető fejlesztő fordítja le a kódot a saját számítógépén, így az ő felelőssége, hogy mindenhol ugyanazt indítsa el
- [x] a csomagolás fázis után kimentjük a telepítendő csomagot és mindenhova ezt telepítjük
- [ ] a szoftvert manuálisan telepítjük az első virtuális tesztgépre, majd ezt a virtuális gépet klónozva tudjuk elindítani további környezetekben

Magyarázat a megoldáshoz

8. feladat 0 / 5 pont

Milyen technikák segíthetnek meggyőzni a CEO-t, hogy automatikusan történjen meg a telepítés külön emberi beavatkozás és engedély nélkül?

- [x] A/B telepítés (A/B deployment)
- [ ] Regressziós tesztelés kihagyása
- [ ] Kódmódosítások tiltása élesítés előtt (code freeze)
- [ ] Funkciómódosítások tiltása élesítés előtt (feature freeze)
- [x] Funkciók futás idejű bekapcsolhatósága (feature flag / feature toggle)
- [ ] A fejlesztő és üzemeltető csapatok szétválasztása
- [x] Kék-zöld telepítés (Blue-green deployment)

Magyarázat a megoldáshoz

Vissza a kategóriához