DEVOPS

3. forduló

<epam>

A kategória támogatója: EPAM

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

Felhasznált idő: 00:00/10:00 **Elért pontszám:** 0/8

1. feladat 0/1 pont

Sokat hallani, hogy a DevOps az nem is egy szerep, hanem valójában egy szervezeti kultúra.

Az alábbi lehetőségek közül választva, hogy építenéd fel egy projekten a csapatot(okat), hogy leghatékonyabban tudd segíteni ennek a kultúrának az elterjedését?

Válasz

- Külön fejlesztői, tesztelő, üzemeltetői csapatokat hoznál létre, a köztük lévő kommunikáció felügyeletét, lebonyolítását rábíznád egy dedikált személyre(ekre), pl. csapatvezetők, projekt manager, így biztosítva, hogy minden csapattag tisztában legyen a feladataival, hatáskörével, illetve a kommunikációs csatornák egy kézben tartásával biztosítanád, hogy a szükséges információ mindig eljusson a megfelelő személyekhez.
- Külön fejlesztői és üzemeltetői csapatot hoznál létre, majd kialakítanál egy olyan release folyamatot amelynek során a fejlesztők az eddig használatos telepítése és beállítási útmutatók helyett, élő szóban magyaraznák el az üzemeltetésnek, hogy mire kell figyelni telepítés során és milyen fontos változások fognak történni.
- Létrehoznál egy önálló DevOps csapatot, amely a fejlesztők és az üzemeltető csapatok közé "beékelődve" megpróbálna átvenni mindkét féltől bizonyos feladatokat, annak érdekében, hogy a fejlesztés és üzemeltetés közötti éles határvonalat megprobálja elmosni, jobban átjárhatóvá tenni.

- [x] Egy csapatot hoznál létre, melyben a lehető legtöbb releváns fejlesztéssel, üzemeltetéssel kapcsolatos szaktudás képviselve van, és ezen csapat feladata lenne az adott alkalmazás teljes életciklusának lefedése, a tervezési döntésektől kezdve, a kód implementálásán át az éles üzemeltetés, ezzel próbálva minden csapattagot érdekeltté téve egy jól működő, jól karbantartható és üzemeltethető szoftver kialakításában.
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

DevOps, mint kultúra alappillérei a közös felelősségvállalás, a területek közötti éles határvonalak lebontása és a kommunikáció minél egyszerűbbé tétele annak érdekében, hogy mindenki elkötelezett legyen a közös cél elérésében. Ennek kapcsán érdemes olyan csapatstruktúrákban gondolkodni, amik elősegítik ezen célok elérését.

2. feladat 0/1 pont

Téged kérnek meg, hogy javasolj egy **branching stratégiát,** amely legjobban támogatja az újonnan kialakítandó alkalmazással kapcsolatos követelményeket, amelyeket egy tapasztalt fejlesztőkből álló csapat fog implementálni. Mit ajánlanál az alábbi kritériumok alapján?

- Lehetőleg minimalizálja az előforduló merge conflictok számát és nagyságát a fejlesztési folyamat során
- Ne legyen egyszerre túl sok aktív branch ág
- Bármely tesztek által validált commit telepíthető legyen az éles rendszerre
- Bármely fejlesztő által készített kódbeli változtatások minél hamarabb jelenjenek meg mindenkinél, függetlenül attól, hogy azok már teljeskörűen működnek vagy sem

Válasz

- [ ] Gitflow model, mivel régóta jól bevált megoldás, a különálló release, hotfix, feature, develop branchek létrehozásával jól biztosítja a szeparációt a különböző fejlesztők által aktívan fejlesztett kódrészletek között, így megelőzi, hogy a létrejövő merge conflictok gyakran lelassítsák a produktív munkavégzést.

  Mivel mindenki lát minden branchet, így könnyen megoldható, hogy akinek szüksége van az adott, még a develop branchbe nem commitált változásokra, azokat könnyen beszerezze.

- [ ] Nem érdemes formális branch stratégiát választani. Mivel cél, hogy kevés aktív branch legyen, így minden fejlesztő dolgozik helyileg a saját branchén, majd előre megbeszélt időpontokban összeülnek és közösen minden változást betesznek a fő ágba, így csökkentve a merge conflictok számát, hiszen csak ezen jól deffiniált időpontokban történhet meg.

- [x] Trunk based development model, mivel célja a trunk branch mindig egészséges állapotban tartása, így könnyen megvalósítható, hogy ez a branch bármikor telepíthető legyen.
  Ez a válasz helyes, de nem jelölted meg.

- [ ] Github flow model, amely egy egyszerű stratégia, master és feature branchekkel dolgozik, és garantálja, hogy a master branch mindig olyan állapotban van, ami telepíthető az éles rendszerre.

Magyarázat

Trunk based development stratégia célja, hogy bármilyen változás minél hamarabb bekerüljön a közös trunk branchbe amennyiben átmegy a megfelelő teszteken. Ha ez a változás egy még nem telejsen működő funkcióhoz tartozik, akkor feature flagekkel megoldható ezen kódrészlet kikapcsolása az éles rendszerben, így megelőzve hogy nem működő funkciók legyenek a felhasználók számára elérhetőek, azonban megadja a lehetőséget a fejlesztők, tezstelők számára, ezen félig kész funkciók tesztelésére, kipróbálására.

3. feladat 0/1 pont

Mit csinál Pythonban az [::-1] kifejezés?

Válaszok

- [ ] Törli a lista utolsó elemét
- [x] Megfordítja a lista elemeinek sorrendjét
  Ez a válasz helyes, de nem jelölted meg.
- [x] Nincs ilyen kifejezés Pythonban
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Ez egy IPv6 hálózati cím

Magyarázat

A lista elemeit fordított sorrendbe rendezi az eredeti állapothoz képest (a válasz egyben a magyarázat is).

Kedves Versenyzők!

A Python kifejezés hibásan lett feltéve a kérdésben, bár nem ez volt a szándék, de így elfogadjuk a „Nincs ilyen kifejezés" válaszlehetőséget is.

4. feladat 0/1 pont

Egy AWS Lambda funkció hibafelderítése során a kódban implementálsz loggolást annak érdekében, hogy információkat szerezz, hogy mi történik a Lambda funkció futása során, azonban a várt logsorok nem jelennek meg Cloudwatchban.

A loggolás kódrészletet lokálisan futtatva szépen láthatóak a log üzenetek a terminálon, így vélhetően a kód ezen része megfelelően működik.

Mi okozhatja a logok hiányát a CloudWatchban?

Válasz

- ( ) Hiányzik a CloudWatch agent a Lambda funkcióból
- (•) A Lambda funkcióhoz társított IAM role-ban nincs megfelelő jogosultság a CloudWatch service használatához
  Ez a válasz helyes, de nem jelölted meg.
- ( ) AWS Lambda logokat a CloudTrail-ben kell keresni
- ( ) Hiányzik a Lambda funkcióhoz szükséges CloudWatch log group

Magyarázat

A CloudWatch logs automatikusan integrálva van az AWS Lambda function-ökkel, a felsoroltak közül csak a Lambda IAM Role-jában (Execution role) szükséges megadni a CloudWatch-hoz a hozzáférést.

https://aws.amazon.com/lambda/faqs/

https://docs.aws.amazon.com/lambda/latest/dg/lambda-permissions.html

5. feladat 0/1 pont

A NAT instance-re (NAT-olásra használt EC2 instance) és NAT Gateway-re melyik állítás **NEM igaz**?

Válasz

- [ ] A NAT Gateway subnet-enként jön létre
- [x] A NAT Gateway támogatja a port forwarding-ot
  Ez a válasz helyes, de nem jelölted meg.
- [ ] A NAT instance bastion host/jump host-ként használható
- [ ] A NAT instance Security Group-ot használ

Magyarázat

A NAT instance-on megoldható a port forwarding, de a NAT Gateway nem támogatja.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html

6. feladat 0/1 pont

Egy Terraform kód használata során felmerül a kérdés, hogy egy adott foo nevű változónak értéket kell adni. Milyen mód(ok)on tudod ezt megtenni?

Válaszok

- [x] terraform plan -var 'foo=bar'
  Ez a válasz helyes, de nem jelölted meg.
- [x] terraform plan -var-file=vars/testenv.tfvars

  Amennyiben a foo változó definiálva van a vars/testenv.tfvars fájlban
  Ez a válasz helyes, de nem jelölted meg.
- [x] TF_VAR_foo=bar terraform plan
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Magában a Terraform kódban kell az értékadást megvalósítani, ezen kívül nincs lehetőség ennek a változónak az értékét "kívülről" módosítani.

Magyarázat

Mindegyik formátum támogatott, sőt adott könyvtárból automatikusan is betölt változó definicióra használt file-okat, például a terraform.tfvars-t.

7. feladat 0/1 pont

Egy build és deploy pipeline kialakítása során az egyes lépéseket milyen sorrendben hajtanád végre, ha cél a **minél gyorsabb visszajelzés** a fejlesztő számára?

Válasz

- [x] 1. linting
  2. unit teszt
  3. statikus kód analízis
  4. alkalmazás buildelése
  5. alkalmazás csomagolása (pl. docker image készítése)
  6. integrációs teszt
  7. teljesítmény teszt
  8. blue-green deployment

  Ez a válasz helyes, de nem jelölted meg.
- [ ] 1. alkalmazás buildelése

2. alkalmazás csomagolása (pl. docker image készítése)
3. linting
4. unit teszt
5. teljesítmény teszt
6. statikus kód analízis
7. integrációs teszt
8. blue-green deployment

- [ ] 1. statikus kód analízis
2. linting
3. unit teszt
4. blue-green deployment
5. alkalmazás buildelése
6. alkalmazás csomagolása (pl. docker image készítése)
7. integrációs teszt
8. teljesítmény teszt

- [ ] 1. linting
2. unit teszt
3. statikus kód analízis
4. integrációs teszt
5. teljesítmény teszt
6. alkalmazás buildelése
7. blue-green deployment
8. alkalmazás csomagolása (pl. docker image készítése)

Magyarázat

Érdemes a különböző tesztelési megoldásokat erőforrás, időtartam, komplexitás szerint rendezni, annak érdekében, hogy az alapvető hibák már a folyamat elején, gyorsan kiderüljenek, ezzel időt és erőforrást spórolva.

8. feladat 0/1 pont

Sikeresen beállítottad a Jenkins szervereteket, azonban jó lenne valamilyen stratégiát alkalmazni, hogy egy esetleges hiba esetén a beállítások és a rajta futó jobok beállításai könnyen visszaállíthatóak legyenek. Mi(ke)t választanál az alábbiak közül?

Válaszok

- [ ] Jenkins a háttérben egy Mysql adatbázisban tárolja a konfigurációkat, így elég annak a megfelelő mentési és visszaállítási tervéről gondoskodni. Új Jenkins telepítésekor elég ennek az adatbázisnak a betöltése, és visszakapjuk a kívánt állapotot.

- [x] Jenkins a beállításokat a saját $HOME könyvtárában tárolja, így ennek mentéséről és visszaállíthatóságáról érdemes gondoskodni.
Ez a válasz helyes, de nem jelölted meg.

- [x] Az egész Jenkist futtató szerver rendszeres, teljeskörű, diszk szinten történő mentése elegendő.
Ez a válasz helyes, de nem jelölted meg.

- [x] Minden Jenkins beállítást és jobot egy git repositoryban érdemes tárolni, pl. Jenkis Configuration as Code (JCasC), Job DSL, Pipeline segítségével. Ezekből nagyon könnyen reprodukálható a környezet.
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Mai környezetekben ajánlott a különböző deklaratív leírók használata (pl. Job DSL), amivel könnyen, akár automatizált módon tudjuk a szolgáltatásainkat kezelni, beleértve a Jenkins szervert is.

Ezen felül megoldás lehet a Jenkins lokális fájlainek valamilyen formában történő mentése, amikből szintén megoldható a visszaállítás.