# DEVOPS

7. forduló

<epam>

A kategória támogatója: EPAM

## Ismertető a feladathoz

**Felhasznált idő:** 10:00/10:00 **Elért pontszám:** 0/4

## 1. feladat 0/1 pont

A Kubernetes controllerek felelősek azért, hogy az adott resource állapota minél közelebb legyen az elvárt állapothoz. **Melyik resource-hoz tartozó Kubernetes controller menedzsel közvetlenül POD-okat?**

### Válaszok

- [ ] Service
- [x] Deployment
  **Ez a válasz helytelen, de megjelölted.**
- [x] Job
  **Ez a válasz helyes, és meg is jelölted.**
- [x] ReplicaSet
  **Ez a válasz helyes, de nem jelölted meg.**

### Magyarázat

- A Service resource controller-e hálózati konfigurációt menedzsel, Pod-okat nem
- A Deployment controller nem menedzsel közvetlenül Pod-okat, helyettük ReplicaSet-eket, melyek lehetővé teszik a rollout, rollback funkcionalitásokat.
- Job controller célja, hogy a Pod-okat "Completed" állapotig futassa konfiguralható retry logika menten.
- ReplicaSet feladata, hogy megadott darabszámú és megadott Pod specifikáció szerinti Pod fusson folyamatosan, közvetlen módon általában nem szokták használni ezt a resource-t, a Deployment resource automatikusan létrehoz ReplicaSet-eket, hogy megvalósitsa a szükseges elvárt állapotot.

## 2. feladat 0/1 pont

Kubernetes esetén POD létrehozásakor többféle konténerdefiníciót lehet megadni annak érdekében, hogy komplex futtatási logikát tudjunk létrehozni. **Mely típusok használhatóak az alábbiak közül a POD yaml fájlban?**

### Válaszok

- [x] container
  **Ez a válasz helyes, de nem jelölted meg.**
- [x] init-container
  **Ez a válasz helyes, de nem jelölted meg.**
- [ ] ephemeral-container
- [ ] termination-container

### Magyarázat

- Container jelenti a POD-ban a szokásos konténereket, melyek általában tartósan kell, hogy fussanak (kivetel pl. a Job-okban definiált konténerek, melyeknél a cél, hogy sikeresen termináljanak)
- Init-container a POD spec mezőjében megadható és a POD lifecycle-ben az inicializálás a feladatkörük: sikeres return-code-dal kell a benne futo processz-nek visszatérnie és csak ezután kezdődik el a "normális" konténerek indítása
- Az ephemeral container letező dolog, viszont a POD YAML-ben nem lehet megadni létrehozáskor. Már létező POD-hoz lehet utólagosan hozzáadni (általában debug céllal) és a Kubernetes nem nyújtja a szokásos container lifecycle garanciákat erre.
- Termination-container: ez jelenleg nem létező konténer típus

## 3. feladat 0/1 pont

Hálózat konfigurálásakor próbáljuk a routingot és az ehhez kapcsolódó biztonsági beállításokat optimalizálni, hatékonnyá tenni. Ennek kapcsán **mire szolgál az rp_filter Linux kernel beállítás?**

### Válasz

- ( ) Átterhel néhány útválasztási (routing) számítási feladatot a hardweres Route Processor-nak, ha a hardware támogatja, így gyorsítva a csomagok feldolgozását és továbbítását.
- ( ) Engedi/tiltja a hálózati csomagokat, amik multicast Rendezvous Point eszközöknek szólnak.
- (•) Engedi/tiltja a csomagokat, amik egy adott hálózati csatolóra (NIC) érkeznek be, de a rájuk küldendő válaszcsomagok másik NIC-en kerülnének kiküldésre a routing szabályok alapján.
  **Ez a válasz helyes, de nem jelölted meg.**
- ( ) A nem router szerepkört betöltő gépeken való a routereknek szóló forgalom kiszűrésére.

### Magyarázat

Az rp_filter egy csatolónkénti (per-interface) beállítás, ami a több alhálózatra kötött (multi-homed) gépeken hasznos, vagy más hálózati eszközökön, ahol asszimetrikus útválasztási (routing) házirend van. A többi válaszlehetőség értelmetlen.

## 4. feladat 0/1 pont

Jenkins használata során szükségünk lenne az optimális megoldás megtalálására, hogy a lehetőségekhez képest optimális számú agent álljon rendelkezésre a különböző típusú jobok végrehajtására, azonban a költségeket is alacsonyan tudjuk tartani.

**Mely megoldások a leginkább célravezetők?** Ha a jobok futtatásának eloszlása nem jól tervezhető, nagyon terhelt, illetve hosszú üresjárati időszakok váltják egymást véletlenszerűen.

## Válaszok

- [ ] Agentjeinket pl. az Amazon felhőjébe telepítjük, EC2 reserved instance-okra, így a szükséges kapacitás mindig rendelkezésre fog állni, és az 1 vagy 3 éves hosszútávú szerződés miatt elég kedvezményt kapunk, így a költség is alacsonyabb, kiszámítható lesz.
- [x] Jenkins dinamikus agenteket használunk, on-demand vagy spot-instance (ha az agenten futtatott job természete ezt megengedi) EC2-kel, figyelve az egyszerre maximálisan létrehozható agentek számának beállítására, így biztosítva a rendelkezésre álló agentek számát, illetve a költségek kontrollját is.
  **Ez a válasz helyes, de nem jelölted meg.**
- [x] Jenkins esetén lehetőségünk van az agenteket Kubernetesben dinamikusan létrehozni az aktuális workload függvényében. Így a Kubernetes cluster megfelelő dinamikus skálázásával tudjuk biztosítani a szükséges kapacitást, illetve a költségek kontrollját.
  **Ez a válasz helyes, de nem jelölted meg.**
- [ ] On-prem infrastuktúrát használunk az agentek futtatására, hiszen az már úgy is "ki van fizetve", így nincs további költsége, valamint állandóan a mi rendelkezésünkre áll.

## Magyarázat

Alapvetően a dinamikus agentek használata tud biztosítani egy optimális egyensúlyt a rendelkezésre álló kapacitások és a költségek között. Történjen ez virtuális gépek, vagy konténerek használatával.

Reserved instanceok használata - bár költségoptimalizálás szempontjából hasznos lehet-, azonban a kiszámíthatatlan workload miatt nem biztos, hogy hosszú távú szerződés megfelelő esetünkben.