

JAVA SPRING MICROSERVICES

2. forduló



A kategória támogatója: Vodafone Magyarország
Zrt.

Ismertető a feladathoz

Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- Ha lejár a feladatlap ideje, a rendszer **AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Badge-ke**t a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

Maguk ismerik a káosz elméletet? – Dr. Ian Malcolm

Csapatotok könnyen vette az első akadályokat, most viszont az alkalmazást fel kell bontani több szolgáltatásra, a csapat eltökélt a microservice architektúra mellett.

Mire lesz szükséged a fordulóban?

- Internet kapcsolat
- Docker és docker-compose

Ha egy adott feladat egy annotációt vár válaszként, akkor oda csak az annotáció nevét add meg @ előjellel, nem szükséges teljes minősített nevet megadni, például: @NotNull.

Felhasznált idő: 00:00/30:00

Elért pontszám: 0/11

1. feladat 0/2 pont

A felsorolásban lévő elemek közül melyek **nem** konténerizációs technológiák?

Válaszok

☐ Podman

☐ LXC/LXD

☒ Cargo
Ez a válasz helyes, de nem jelölted meg.

☐ rkt

☒ GCP
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A fejlesztők többsége valószínűleg ha konténerizációs technológiáról van szó akkor a Dockerral találkozott, viszont a Docker előtt is már voltak más eszközök a virtualizáció és a konténerizáció megoldására, az LXC/LXD projektek, a Podman és az rkt hasonló problémákat old meg mint a Docker csak másabb felépítéssel rendelkeznek. *A listában a Cargo egy Rust-os csomag menedzsert is jelenthet, de létezik egy Java-hoz köthető konténerizációs technológia amely Java konténereket (nem Docker konténerekről van szó hanem leginkább alkalmazás szerverekről) képes menedzselni, ennek a neve (CodeHaus) Cargo.* A listában a GCP nem tartozik a kért csoportba, a GCP a Google Cloud Platformnak a rövidítése amin bár van Kubernetes de önmagában nem számít konténerizációs technológiának.

2. feladat 0/2 pont

A területeket menedzselő szolgáltatásnak rendelkeznie kell egy perzisztens tárral, jelen esetben egy adatbázissal, amely tárolni fogja a különböző területeket és azok paramétereit.

A lokális fejlesztéshez ezt a függőséget **docker-compose** segítségével szeretnénk futtatni. Egy junior kolléga bele is kezdett, de elakadt, a vezetője segítségét kéri:



Állítása szerint, miután kiadta a **docker-compose create** nevű parancsot, majd a **docker-compose start** parancsot, a következőt látja a konzolon:

```
Starting territory-management-db ... done
```

,viszont az adatbázist mégsem tudja elérni az **6543**-es porton.

Mi lehet a baj?

Válaszok

☐ Nem létezik a postgres:14 nevű Docker kép fájl a tárolóban, de ezt nem jelzi a konzol

☐ Mivel egy adatbázist szeretnénk elindítani, muszáj megadnunk egy **volume** kulcsszóval egy csatolt könyvtárat

☒ Hiányzik egy környezeti változó, és nem indult el a konténer.
Ez a válasz helyes, de nem jelölted meg.

☐ A 6543 számú port nem használt a konténeren belül, emiatt nem tud elindulni a konténer.

✓ A port átirányítás nem a megfelelő kulcsszóval történt meg.
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Ha a logokat megnézzük, akkor látjuk, hogy hiányzik egy konfigurációs beállítás, amelyet egy környezeti változóval tudunk beállítani, melyet megadhatunk a compose fájlban az **environment** vagy **env_file** nevű kulcsszóval. Az **env_file** esetén egy külső fájlban kell megadnunk a hiányzó változót.

Az **expose** kulcsszó használata helyett a **ports** kulcsszót kellene használni, és ha a 6543 porton szeretnék elérni a konténert, akkor a következőképpen kellene definiálnunk a YAML fájlt:

```
services:
  territory-management-db:
    image: postgres:14
    ports:
      - "6543:5432"
    environment:
      POSTGRES_PASSWORD: OITMVI
```

3. feladat 0/1 pont

Adott a következő két fájl egymás mellett, melyik MySQL Docker képfájl (image) címke (tag) kerül letöltésre és elindításra, ha a két fájl mellett terminálban állva a következő parancsot adjuk ki: **docker-compose up -d --quiet-pull**?

mysql.env:

```
MAJOR=5
MINOR=3
MYSQL_TAG=${MAJOR:-5}.${MINOR:-4}
MYSQL_RANDOM_ROOT_PASSWORD=true
```

docker-compose.yml:



```
services:
  mysql:
    container_name: mysql-${MYSQL_TAG-5.3}
    image: mysql:${MYSQL_TAG-5.5}
```

docker-compose.override.yml:



```
services:
  mysql:
    env_file:
      - mysql.env
    image: mysql:${MYSQL_TAG-5.7}
```

Válaszok

A helyes válasz:

5.7

mysql:5.7

Magyarázat

A `docker-compose up -d --quiet-pull` parancs futtatására a Docker felolvassa a `docker-compose.yml` fájlt és a `docker-compose.override.yml` fájlt is ha található mellette. Jelen esetben igen. A futtatandó konténerhez tartozó képfájl egy "címkéje" (angolul **tag**) egy környezeti változóból jön, amelyet bár definiáltunk a `mysql.env` fájlban, de azt a parancs megadásakor automatikusan nem olvassa fel a `docker-compose`, a fájlban bár megjelenik az `env_file` kulcsszó mellett, de ebben az esetben a már létrejött konténeren belül fog alkalmazódni a változókat tároló fájl tartalma.

A címke kicsit trükkösen van megadva, mivel nincs `.env` fájl amiből felolvassa így valójában nem létezik a kontextusban a `MYSQL_TAG` nevű változó így a `docker-compose.override.yml` fájlban a `${MYSQL_TAG-5.7}` kifejezés igazából az `5.7`-es értéket veszi fel, így ezzel a címkével rendelkező a képfájlt tölti le.

```
> docker-compose up -d --quiet-pull
Creating mysql-5.3 ... done

> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
36c34e54f67c	mysql:5.7	"docker-entrypoint.s..."	About a minute ago	Up About a minute	3306/tcp, 3306

4. feladat 0/2 pont

A fejlesztések indulásakor a dinoszauruszokat menedzselő szolgáltatás egy másik csapat kezében összpontosult, viszont most úgy néz ki, hogy a Te csapatodnak kell átvennie, de a forráskódhoz még nem adtak hozzáférést, ellenben egy misztikus chaten keresztül kaptál egy Docker képfájl nevet: **ghcr.io/ingen-sol/dinosaur-management-service**

Egyből munkához is látsz, hogy felfedezd mire képes a szolgáltatás és megpróbálsd elindítani, viszont a függőségek listáját nem közölték veled, így Neked kell rájönnöd, hogy milyen függőségek is kellenek az elindításhoz. A chaten elhangzott, hogy bizonyos szenzitív adatokat fel lehet fedni egy konfigurációs beállítással.

Milyen függőségeket igényel az alkalmazás?

A 4.-7. feladatok megoldásához a fenti linket használd!

Válaszok

☒ Redis
Ez a válasz helyes, de nem jelölted meg.

☐ PostgreSQL

☐ MongoDB

☐ Kafka

☒ MySQL
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A mini alkalmazás elindításához kettő függőség szükséges, egy MySQL amelynek simán egy **root** felhasználót kell létrehoznunk, a jelszava **dennis-nedry-is-a-traitor** és az adatbázis neve pedig **dinosaur-service**. A másik függőség egy Redis. Ez a két válaszlehetőség csak akkor derülhet ki, ha valaki letölti a docker képfájlt, és konténert kreál belőle.

5. feladat 0/1 pont

Milyen környezeti változó segítségével tudunk több információt kideríteni a szolgáltatásról?

Válaszok

A helyes válasz:

MODE

MODE="DEBUG"

MODE='DEBUG'

MODE=DEBUG

Magyarázat

Az adatbázis jelszava "biztonságos" módon az alkalmazás indításakor kiírodik, ha a **MODE** nevű környezeti változót **DEBUG**-ra állítjuk be.

6. feladat 0/1 pont

Milyen környezeti változókat kell megadnunk a **dinosaur-management-service** képfájlnak, hogy az általunk felderített függőségek elérhetőségeit konfigurálhassuk?

Válaszok

☐ SPRING_RABBITMQ_ADDRESSES

☒ SPRING_DATASOURCE_URL
Ez a válasz helyes, de nem jelölted meg.

☒ SPRING_REDIS_URL
Ez a válasz helyes, de nem jelölted meg.

☐ SPRING_KAFKA_BOOTSTRAP_SERVERS

☐ SPRING_DATA_MONGODB_HOST

Magyarázat

7. feladat 0/2 pont

Milyen jelszó szükséges az egyik függőség csatlakozásához?

Válasz

A helyes válasz:

dennis-nedry-is-a-traitor

Magyarázat

Egy lehetséges docker-compose alapú megoldás:

```
version: '3.7'
networks:
  local:
services:
  service:
    image: ghcr.io/ingen-sol/dinosaur-management-service
    environment:
      MODE: DEBUG
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql-db:3306/dinosaur-service
```

```
    SPRING_REDIS_URL: redis://redis:6379
  networks:
    local:
redis:
  image: redis:6.0.5-alpine
  networks:
    local:
mysql-db:
  image: mysql:5.7
  environment:
    MYSQL_DATABASE: dinosaur-service
    MYSQL_ROOT_PASSWORD: dennis-nedry-is-a-traitor
  networks:
    local:
```

Ha ezt a 3 konténert egy darab docker-compose fájlban definináljuk, akkor ismernünk kell a környezeti változók beállításának lehetőségeit, illetve a Docker hálózati beállításait, hogy a függőségeket elérje a szolgáltatás.



[Legfontosabb tudnivalók](#) [Kapcsolat](#) [Versenyszabályzat](#) [Adatvédelem](#)

© 2023 Human Priority Kft.

KÉSZÍTETTE **cone**

Megjelenés

Világos