Kubernetes - Conclusion et évaluation

120 minutes Normal

DataScientest • com

Kubernetes

5. Conclusion et Évaluation

a. Conclusion

Dans ce cours, nous avons pu voir l'intérêt d'une plateforme comme Kubernetes. Nous avons pu voir l'architecture et différents objets qui constituent l'utilisation de Kubernetes. Nous avons enfin vu comment déployer des applications avec Kubernetes.

Kubernetes, étant un outil très complexe, nous n'avons pas abordé tous les éléments qui le composent, ni les aspects de sécurité ou même de persistance des données. On aurait aussi pu parler longuement de l'articulation de Kubernetes avec les hébergeurs de Cloud.

b. Évaluation

Présentation

données. Notre API est constituée de deux containers:

- le premier contient une base de données MySQL: `datascientest/mysql-k8s:1.0.0`
- le second contient une API FastAPI

Le container de l'API FastAPI n'est pas encore construit mais les différents fichiers sont déjà créés:

- le `Dockerfile`

```
FROM ubuntu:20.04

ADD files/requirements.txt files/main.py ./

RUN apt update && apt install python3-pip libmysqlclient-dev -y && pip install -r requirements.txt

EXPOSE 8000

CMD uvicorn main:server --host 0.0.0.0
```

- le fichier `main.py` qui contient l'API

```
from pydantic import BaseModel
from sqlalchemy.engine import
create_engine

# creating a FastAPI server
server = FastAPI(title='User API')

# creating a connection to the database
mysql_url = ''  # to complete
mysql_user = 'root'
mysql_password = ''  # to complete
database_name = 'Main'

# recreating the URL connection
connection_url = 'mysql://{user}:
{password}@{url}/{database}'.format(
    user=mysql_user,
    password=mysql_password,
    url=mysql_url,
    database=database_name
)

# creating the connection
mysql_engine =
create_engine(connection_url)


# creating a User class
class User(BaseModel):
    user_id: int = 0
    username: str = 'daniel'
    email: str =
'daniel@datascientest.com'


@server.get('/status')
async def get_status():
    """Returns 1
    """
    return 1


@server.get('/users')
async def get_users():
    with mysql_engine.connect() as
```

```
Users;')

    results = [
        User(
            user_id=i[0],
            username=i[1],
            email=i[2]
            ) for i in
results.fetchall()]
    return results


@server.get('/users/{user_id:int}',
response_model=User)
async def get_user(user_id):
    with mysql_engine.connect() as
connection:
        results = connection.execute(
            'SELECT * FROM Users WHERE
Users.id = {};'.format(user_id))

    results = [
        User(
            user_id=i[0],
            username=i[1],
            email=i[2]
            ) for i in
results.fetchall()]

    if len(results) == 0:
        raise HTTPException(
            status_code=404,
            detail='Unknown User ID')
    else:
        return results[0]
```

- le fichier `requirements.txt` qui contient les librairies Python à installer

```
3 mysqlclient
4 uvicorn
```

Les consignes

Le but de cet exercice est de créer un `Deployment` avec 3 `Pods`, chacun de ces Pods contenant à la fois un container MySQL et un container FastAPI. Il faudra ensuite créer un `Service` et un `Ingress` pour permettre l'accès à l'API.

Il faudra donc compléter le code fourni pour l'API et reconstruire l'image Docker correspondante (et la téléverser dans DockerHub), de manière à permettre la communication entre l'API et la base de données. De plus, il faudra changer le code de l'API pour récupérer le mot de passe de la base de données: `datascientest1234`. Toutefois, ce mot de passe ne peut pas être codé en dur et doit donc être mis dans un `Secret`.

Les rendus

Les rendus attendus sont un ensemble de fichiers avec éventuellement un fichier de commentaire:

- le fichier `main.py` remanié
- un fichier `my-deployment-eval.yml` contenant la déclaration du `Deployment`
- un fichier `my-service-eval.yml` contenant la déclaration du `Service`
- un fichier `my-ingress-eval.yml` contenant la déclaration de l'`Ingress`
- un fichier `my-secret-eval.yml` contenant la déclaration du `Secret`

Bon courage !

Validated