Homework Sesi 25 Bootcamp DevOps Engineer Digital Skola

1. Create sebuah deployment menggunakan image docker yang sudah di build sebelumnya
2. Create sebuah service dan hubungkan service tersebut ke deployment
3. Lakukan port-forward dan pastikan deployment dan service yang anda buat sudah benar

Prasyarat:

Pastikan Minikube dan kubectl sudah terinstal di sistem operasi.

Langkah 1: Membuat File YAML untuk Deployment

Buat sebuah file bernama deployment.yaml dengan isi sebagai berikut.

```
ubuntu_admin@kube:~$ nano deployment.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-ku
spec:
  replicas: 3 #Jumlah pod yang diinginkan
  selector:
    matchLabels:
      app: nginx-brother # Label selector untuk pod
  template:
    metadata:
      labels:
        app: nginx-brother # Label untuk pod
    spec:
      containers:
      - name: nginx-keren # Nama container
        image: avimajid/aplikasi-nginx-simple:c5ad98cf #nama image Docker di
Docker Hub
        ports:
        - containerPort: 80
```

Penjelasan:

- **apiVersion: apps/v1**: Menentukan versi API Kubernetes yang digunakan untuk Deployment.
- **kind: Deployment**: Menentukan jenis resource yang akan dibuat, yaitu Deployment.
- **metadata: name**: Nama untuk Deployment.
- **spec: replicas**: Jumlah replika (pod) yang diinginkan untuk aplikasi.
- **spec: selector: matchLabels**: Digunakan untuk menyeleksi pod yang dikelola oleh Deployment berdasarkan label.
- **spec: template: metadata: labels**: Label yang akan diterapkan pada setiap pod yang dibuat oleh Deployment.
- **spec: template: spec: containers**: Daftar container yang akan dijalankan di dalam pod.
    - name: Nama container.
    - image: Image Docker yang akan digunakan.

- ports: containerPort: Port yang diekspos oleh aplikasi di dalam container.

Langkah 2: Menerapkan Deployment ke Minikube

Buka terminal atau command prompt, arahkan ke direktori tempat file deployment.yaml disimpan, dan jalankan perintah **kubectl apply -f deployment.yaml**

```
ubuntu_admin@kube:~$ kubectl apply -f deployment.yaml
deployment.apps/nginx-ku created
```

Langkah 3: Memastikan Deployment Berjalan dengan Baik

Lihat status Deployment dengan command **kubectl get deployments [nama-deployment]**. Ganti **[nama-deployment]** dengan label app yang ditentukan pada file deployment.yaml

```
ubuntu_admin@kube:~$ kubectl get deployments nginx-ku
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx-ku   3/3     3            3           99m
```

Pastikan nilai pada kolom READY sesuai dengan jumlah replicas yang ditentukan.

Lihat pod yang dibuat oleh Deployment, dengan perintah **kubectl get pods -l app=[nama-aplikasi]**. Ganti **[nama-aplikasi]** dengan label app yang ditentukan pada file deployment.yaml.

```
ubuntu_admin@kube:~$ kubectl get pods -l app=nginx-brother
NAME                        READY   STATUS    RESTARTS   AGE
nginx-ku-57548c668b-ck2mk   1/1     Running   0          100m
nginx-ku-57548c668b-fhqkq   1/1     Running   0          100m
nginx-ku-57548c668b-hdwdn   1/1     Running   0          100m
```

Pastikan status semua pod adalah Running.

Langkah 4: Membuat File YAML untuk Service

Buat sebuah file bernama service.yaml dengan isi sebagai berikut:

```
ubuntu_admin@kube:~$ nano service.yaml
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-ku-service
spec:
  selector:
    app: nginx-brother
  ports:
  - protocol: TCP
    port: 8080 # Port yang akan diekspos oleh service
    targetPort: 80 # Port aplikasi di dalam container
  type: ClusterIP # Tipe service yang akan dibuat
```

Penjelasan:

- **apiVersion: v1**: Menentukan versi API Kubernetes yang digunakan untuk Service.
- **kind: Service**: Menentukan jenis resource yang akan dibuat, yaitu Service.
- **metadata: name**: Nama untuk Service.

- **spec: selector**: Digunakan untuk menyeleksi pod yang akan dihubungkan oleh Service berdasarkan label. Pastikan label ini sesuai dengan label pada pod Deployment (app: nginx-brother).
- **spec: ports**: Daftar port yang diekspos oleh Service.
    - protocol: Protokol yang digunakan (TCP atau UDP).
    - port: Port yang akan diekspos oleh Service di dalam cluster.
    - targetPort: Port aplikasi di dalam container yang akan diteruskan oleh Service.
- **spec: type**: Tipe Service. Dalam contoh ini, digunakan ClusterIP, yang membuat Service hanya dapat diakses dari dalam cluster. Tipe lain yang umum digunakan di Minikube adalah NodePort.

Langkah 5: Menerapkan Service ke Minikube

Di direktori tempat file service.yaml disimpan, jalankan perintah **kubectl apply -f service.yaml**:

```
ubuntu_admin@kube:~$ kubectl apply -f service.yaml
service/nginx-ku-service created
```

Langkah 6: Memastikan Service Berjalan dengan Baik

Untuk melihat status Service, jalankan perintah **kubectl get services [nama-service]**. Ganti **[nama-service]** dengan nama yang diberikan pada file service.yaml.

```
ubuntu_admin@kube:~$ kubectl get services nginx-ku-service
NAME               TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
nginx-ku-service   ClusterIP   10.99.7.119    <none>        8080/TCP   63m
```

Pastikan Service memiliki alamat IP pada kolom CLUSTER-IP.

Langkah 7: Melakukan Port-Forward untuk Mengakses Aplikasi

Karena tipe Service yang digunakan adalah ClusterIP, maka perlu melakukan port-forward untuk mengakses aplikasi dari luar cluster (misalnya dari browser lokal). Jalankan perintah **kubectl port-forward service/[nama-service] 8080:8080**. Ganti **[nama-service]** dengan nama service yang dibuat.

```
ubuntu_admin@kube:~$ kubectl port-forward service/nginx-ku-service 8080:8080
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Perintah ini akan meneruskan traffic dari port 8080 di localhost ke port 8080 di dalam Service.

- **kubectl port-forward service/nginx-ku-service**: Perintah untuk melakukan port-forward pada Service.
- **8080:8080**: Format [port-lokal]:[port-service]. Traffic dari port 8080 di komputer lokal akan diteruskan ke port 8080 pada Service.

Langkah 8: Mengakses Aplikasi

Buka terminal baru dan jalankan perintah **curl localhost:8080**. Jika konfigurasi Deployment dan Service sudah benar, maka aplikasi yang berjalan di dalam container akan dapat diakses. Aplikasi yang dijalankan ini diambil dari image hasil build dengan CICD Gitlab yang telah dipush ke Dockerhub yang isi dari file html-nya adalah berikut:

```
ubuntu_admin@kube:~$ curl localhost:8080
<!DOCTYPE html>
<html>
<head>
    <title>Selamat Datang di Nginx!</title>
</head>
<body>
    <h1>Halo dari Nginx via GitLab CI/CD!</h1>
    <p>Aplikasi ini dideploy secara otomatis.</p>
</body>
```

Log yang menunjukkan bahwa service diakses akan muncul.

```
ubuntu_admin@kube:~$ kubectl port-forward service/nginx-ku-service 8080:8080
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

Namespace yang digunakan pada project aplikasi ini adalah default.