Homework Sesi 26 Bootcamp DevOps Engineer Digital Skola

1. Buat HPA ketika deploymement memory menyentuh 80% dan buat min replica nya 2 dan max replica 10
2. Jelaskan secara detail hpa yang telah dibuat diatas

HPA memerlukan akses ke metrik sumber daya (seperti penggunaan CPU dan memori) untuk dapat melakukan penskalaan otomatis. Pastikan **metrics-server** sudah terinstall. Jika belum install dengan aktifkan addon:

minikube addons enable metrics-server

Lalu, verifikasi dengan:

kubectl get deployment metrics-server -n kube-system

Pastikan melihat output yang menunjukkan bahwa Deployment metrics-server memiliki setidaknya satu pod yang READY.

Buat file deployment.yaml:

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

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-ku # nama deployment
spec:
  replicas: 3 # Jumlah awal pod
  selector:
    matchLabels:
      app: nginx-brother # Label selector untuk pod
  template:
    metadata:
      labels:
        app: nginx-brother # Label untuk pod
    spec:
      containers:
      - name: nginx-container # Nama container
       image: avimajid/aplikasi-nginx-simple:c5ad98cf #nama image Docker di
Docker Hub
       ports:
        - containerPort: 80 # Port yang diekspos oleh aplikasi di dalam
container
       resources:
         requests:
           memory: 256Mi # Contoh: meminta 256 megabyte memori.
```

Jalankan perintah **kubectl apply -f deployment.yaml** untuk menerapkan deployment.

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

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 deployment
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx-ku   3/3     3            3           7s
```

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 pod
NAME                        READY   STATUS    RESTARTS   AGE
nginx-ku-676c54bd6b-cwqfb   1/1     Running   0          5m33s
nginx-ku-676c54bd6b-klvld   1/1     Running   0          5m33s
nginx-ku-676c54bd6b-wkrsl   1/1     Running   0          5m33s
```

Selanjutnya buat sebuah file bernama hpa.yaml dengan isi sebagai berikut:

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-memori
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-ku
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

Terapkan hpa dengan arahkan ke direktori tempat file hpa.yaml disimpan, dan jalankan perintah berikut:

kubectl apply -f hpa.yaml

Untuk melihat status HPA, jalankan perintah:

kubectl get hpa hpa-memori

```
ubuntu_admin@kube:~$ kubectl get hpa
NAME          REFERENCE              TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
hpa-memori    Deployment/nginx-ku    memory: 1%/80%   2         10        3          16s
```

Karena deployment membuat 3 replica di awal, hpa yang disetup minimal 2 replica akan menyesuaikannya beberapa saat kemudian.

```
ubuntu_admin@kube:~$ kubectl get hpa
NAME          REFERENCE              TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
hpa-memori    Deployment/nginx-ku    memory: 1%/80%   2         10        2          9m5s
```

```
ubuntu_admin@kube:~$ kubectl get deployment
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx-ku   2/2     2            2           10m
```

```
ubuntu_admin@kube:~$ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
nginx-ku-676c54bd6b-cwqfb   1/1     Running   0          10m
nginx-ku-676c54bd6b-klvld   1/1     Running   0          10m
```

Berikut adalah penjelasan mendetail dari setiap bagian file hpa.yaml:

- **apiVersion: autoscaling/v2**: Menentukan versi API Kubernetes yang digunakan untuk Horizontal Pod Autoscaler. autoscaling/v2 adalah versi yang direkomendasikan karena mendukung lebih banyak fitur dan jenis metrik.
- **kind: HorizontalPodAutoscaler**: Menentukan jenis resource Kubernetes yang akan dibuat, yaitu Horizontal Pod Autoscaler.
- **metadata:**: Berisi metadata tentang HPA.
    - **name: hpa-memori**: Nama unik untuk objek HPA ini di dalam namespace Kubernetes.
- **spec:**: Mendefinisikan spesifikasi dari HPA.
    - **scaleTargetRef:**: Merupakan referensi ke sumber daya (target) yang akan diskalakan oleh HPA.
        - **apiVersion: apps/v1**: Versi API dari sumber daya target. Karena menargetkan Deployment, gunakan apps/v1.
        - **kind: Deployment**: Jenis sumber daya target. Dalam kasus ini, adalah Deployment.
        - **name: nama-deployment**: Nama dari Deployment yang akan dikelola penskalaannya oleh HPA ini.
    - **minReplicas: 2**: Menentukan jumlah minimum replika (Pod) yang harus selalu berjalan untuk Deployment yang ditargetkan. HPA akan memastikan bahwa jumlah replika tidak akan pernah kurang dari 2, meskipun penggunaan memori sedang rendah.

- **maxReplicas: 10**: Menentukan jumlah maksimum replika (Pod) yang dapat dibuat oleh HPA untuk Deployment yang ditargetkan. HPA tidak akan pernah menskalakan Deployment melebihi 10 replika, meskipun penggunaan memori terus meningkat.
- **metrics:**: Mendefinisikan metrik yang akan digunakan oleh HPA untuk menentukan kapan perlu melakukan penskalaan. Ini adalah array, sehingga dapat menentukan lebih dari satu metrik (misalnya, berdasarkan CPU dan memori).
    - **- type: Resource**: Menunjukkan bahwa akan menggunakan metrik sumber daya (CPU atau memori) yang disediakan oleh Kubernetes.
    - **resource:**: Mendefinisikan sumber daya spesifik yang akan dipantau.
        - **name: memory**: Menentukan bahwa akan memantau penggunaan memori.
        - **target:**: Mendefinisikan target nilai untuk metrik ini.
            - **type: Utilization**: Menunjukkan bahwa target didefinisikan sebagai persentase penggunaan sumber daya.
            - **averageUtilization: 80**: Menentukan target rata-rata penggunaan memori sebesar 80% di seluruh Pod dalam Deployment. Jika rata-rata penggunaan memori melebihi 80%, HPA akan mulai menambah jumlah replika (hingga maxReplicas). Jika rata-rata penggunaan memori turun di bawah ambang batas tertentu (biasanya di bawah target), HPA akan mulai mengurangi jumlah replika (hingga minReplicas).

Cara Kerja HPA Ini:

Ketika HPA ini aktif, ia akan secara berkala (biasanya setiap beberapa detik) memeriksa metrik penggunaan memori dari Pod-Pod yang dikelola oleh Deployment nama-deployment. Jika rata-rata penggunaan memori di seluruh Pod melebihi 80%, HPA akan secara otomatis meningkatkan jumlah replika Deployment, dengan tidak melebihi batas maksimum 10 replika. Sebaliknya, jika rata-rata penggunaan memori turun di bawah ambang batas yang ditentukan, HPA akan mengurangi jumlah replika, dengan tidak kurang dari batas minimum 2 replika.

Pastikan Deployment yang ditargetkan memiliki definisi sumber daya (resource requests) untuk memori di dalam spesifikasi Podnya. HPA menggunakan informasi ini untuk menghitung persentase penggunaan memori. Jika resource requests tidak didefinisikan, HPA mungkin tidak dapat bekerja dengan benar untuk metrik memori.

Dengan konfigurasi HPA ini, aplikasi akan secara otomatis melakukan penskalaan berdasarkan kebutuhan memori, memastikan ketersediaan dan kinerja yang lebih baik saat terjadi lonjakan beban.