Uygulama Laboratuvarı: Otomatik Ölçekleme ve Gizli Yönetimi

Bu uygulama laboratuvarı, Kubernetes ile pratik deneyim sağlamak amacıyla tasarlanmıştır ve dikey ve yatay pod otomatik ölçekleme ile gizli yönetimine odaklanmaktadır.

Hedefler

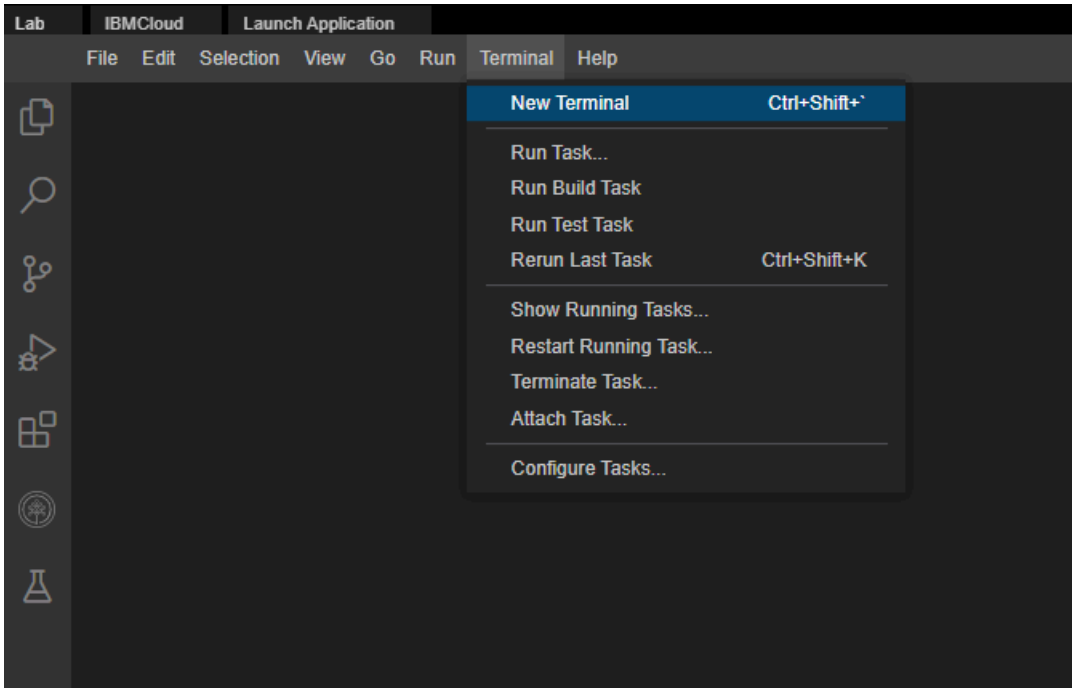
Bu uygulama laboratuvarında şunları yapacaksınız:

- Kubernetes'e bir uygulama inşa edip dağıtmak
- Pod kaynak isteklerini/sınırlarını ayarlamak için Dikey Pod Otomatik Ölçekleyici (VPA) uygulamak
- Kaynak kullanımına dayalı olarak pod replikalarının sayısını ölçeklendirmek için Yatay Pod Otomatik Ölçekleyici (HPA) uygulamak
- Bir Gizli oluşturmak ve bunu kullanmak için dağıtımı güncellemek

> **Not: Lütfen laboratuvarı kesintisiz bir oturumda tamamlayın, çünkü laboratuvar çevrimdışı moda geçebilir ve hatalara neden olabilir. Laboratuvar sürecinde herhangi bir sorun/hata ile karşılaşırsanız, laboratuvar ortamından çıkış yapın. Ardından, sistem önbelleğinizi ve çerezlerinizi temizleyin ve laboratuvarı tamamlamayı deneyin.**

Ortamı Kurun

Menü çubuğunda, `Terminal` üzerine tıklayın ve açılır menüden `Yeni Terminal` seçeneğini seçin.



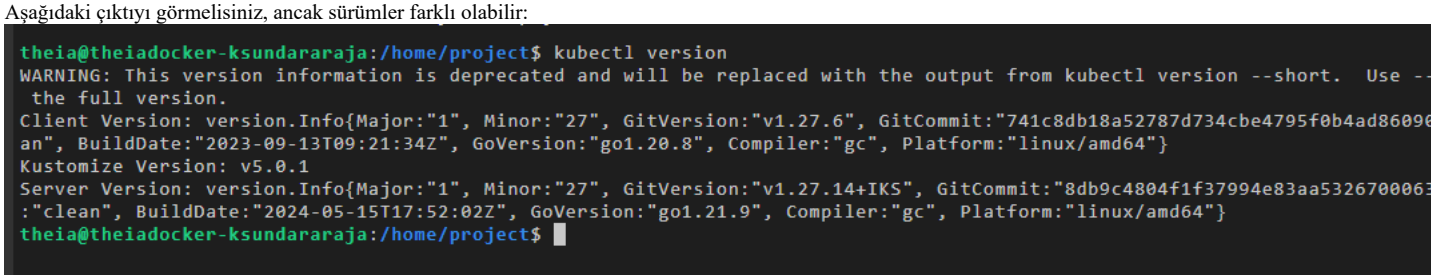
> Not: Eğer terminal zaten açıksa, lütfen bu adımı atlayın.

Adım 1: kubectl sürümünü doğrulayın

Devam etmeden önce, kubectl'nin yüklü ve doğru şekilde yapılandırıldığından emin olun. kubectl sürümünü kontrol etmek için aşağıdaki komutu çalıştırın:

```
kubectl version
```

Aşağıdaki çıktıyı görmelisiniz, ancak sürümler farklı olabilir:



Adım 2: Proje deposunu klonlayın

Projeye başlamak için başlangıç koduyla birlikte depoyu klonlayın.

```
git clone https://github.com/ibm-developer-skills-network/k8-scaling-and-secrets-mgmt.git
```

Egzersiz 1: Bir uygulamayı Kubernetes'e oluşturun ve dağıtın

Bu depodaki Dockerfile, uygulamanın kodunu zaten içeriyor. Sadece docker imajını oluşturacak ve bunu kayıt defterine göndereceksiniz.

Kubernetes'te dağıtılan uygulamanıza `myapp` adını vereceksiniz.

Adım 1: Docker imajını oluşturun

1. Proje dizinine gidin.

   ```
   cd k8-scaling-and-secrets-mgmt
   ```

2. Ad alanınızı dışa aktarın.

   ```
   export MY_NAMESPACE=sn-labs-$USERNAME
   ```

3. Docker imajını oluşturun.

   ```
   docker build . -t us.icr.io/$MY_NAMESPACE/myapp:v1
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ docker build . -t us.icr.io/$MY_NAM
[+] Building 21.9s (5/10)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 464B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/node:14
 => [auth] library/node:pull token for registry-1.docker.io
 => [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461
 => => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461
 => => sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa 776B / 776B
 => => sha256:1d12470fa662a2a5cb50378dcdc8ea228c1735747db410bbefb8e2d9144b5452 7.51kB / 7.51kB
 => => sha256:2cafa3fbb0b6529ee4726b4f599ec27ee557ea3dea7019182323b3779959927f 2.21kB / 2.21kB
 => => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 50.45MB / 50.45MB
/10)                                                                                          d
de101a38a045ff7bc656e3b0fbfc7c05cca5 7.86MB / 7.86MB
 => [internal] load build definition from Dockerfile
d995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131 10.00MB / 10.00MB
 => => transferring dockerfile: 464B
68b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 51.88MB / 51.88MB
 => [internal] load .dockerignore
894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 191.85MB / 191.85MB
 => => transferring context: 2B
05deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8 4.19kB / 4.19kB
 => [internal] load metadata for docker.io/library/node:14
```

Adım 2: Görüntüyü gönderin ve listeleyin

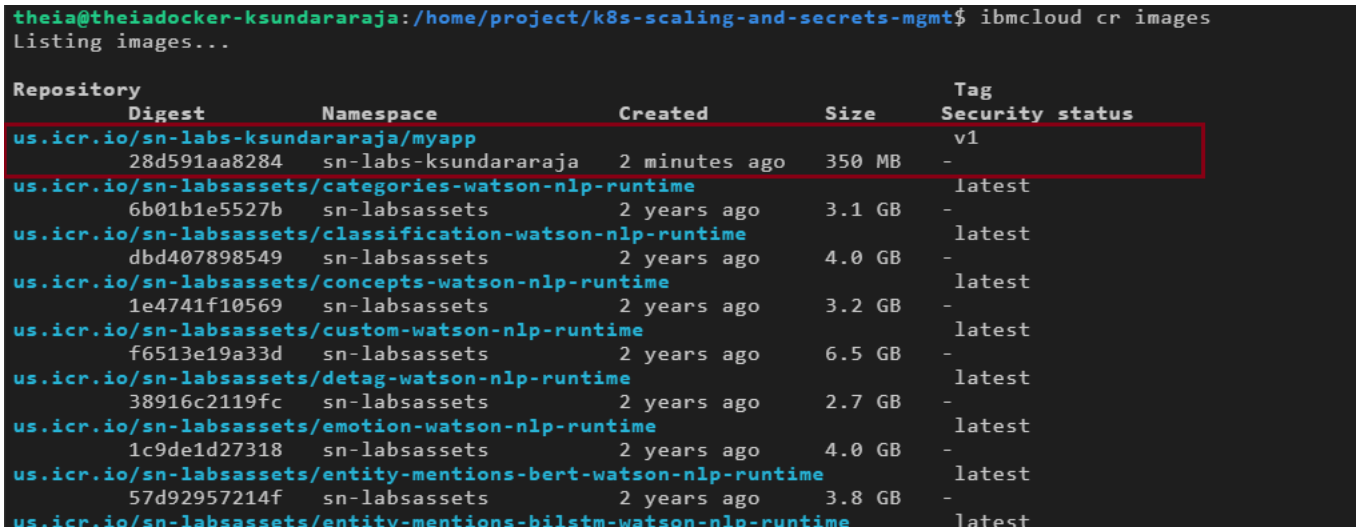
1. Etiketli görüntüyü IBM Cloud Container Registry'ye gönderin.

```
docker push us.icr.io/$MY_NAMESPACE/myapp:v1
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ docker push us.icr.io/$MY_NAMESPACE
The push refers to repository [us.icr.io/sn-labs-ksundararaja/myapp]
d60490235730: Pushed
003de62710da: Pushed
306c0ccb34b4: Pushed
769169bec673: Pushed
0d5f5a015e5d: Pushed
3c777d951de2: Pushed
f8a91dd5fc84: Pushed
cb81227abde5: Pushed
e01a454893a9: Pushed
c45660adde37: Pushed
fe0fb3ab4a0f: Pushed
f1186e5061f2: Pushed
b2dba7477754: Pushed
v1: digest: sha256:28d591aa82841c98be17f9d0f04bc9d56df6e3cce36b43320b64e5747cee2078 size: 3042
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

2. Mevcut olan tüm görüntüleri listeleyin. Yeni oluşturulan `myapp` görüntüsünü göreceksiniz.

```
ibmcloud cr images
```



Adım 3: Uygulamanızı dağıtın

1. Ana proje dizininde bulunan `deployment.yaml` dosyasını açın. İçeriği aşağıdaki gibi olacaktır:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  labels:
    app: myapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: myapp
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - image: us.icr.io/<your SN labs namespace>/myapp:v1
        imagePullPolicy: Always
        name: myapp
        ports:
        - containerPort: 3000
          name: http
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          capabilities:
            drop: ["ALL"]
          seccompProfile:
            type: "RuntimeDefault"
          runAsUser: 999
        resources:
          limits:
            cpu: 50m
          requests:
            cpu: 20m
```

2. `<your SN labs namespace>` kısmını gerçek SN lab'ınızın namespace'i ile değiştirin.

▶ Namespace'inizi almak için buraya tıklayın

3. Dağıtımı uygulayın.

```
kubectl apply -f deployment.yaml
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl apply -f deployment.yaml
deployment.apps/myapp created
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

4. Uygulama pod'larının çalıştığını ve erişilebilir olduğunu doğrulayın.

```
kubectl get pods
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-6cc7f9ffcf-2xnm6   1/1     Running   0          29s
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

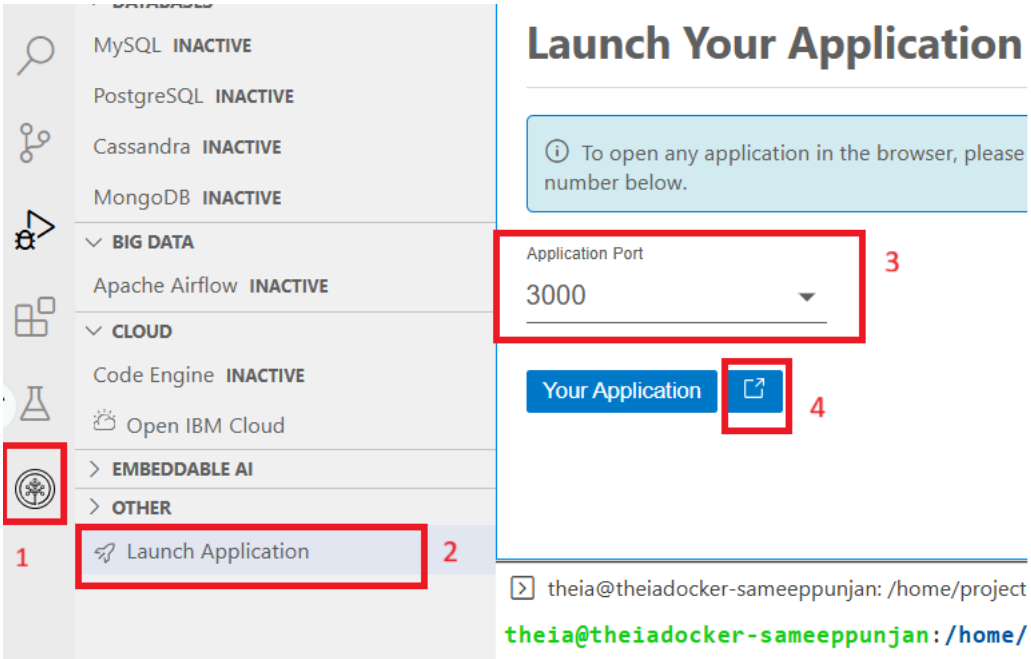
Adım 4: Uygulama çıktısını görüntüle

1. Uygulamayı port yönlendirmesi ile başlatın:

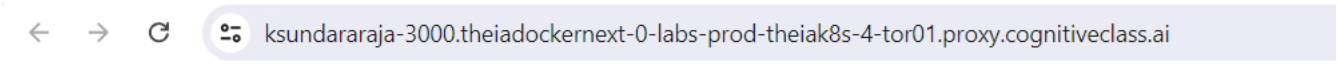
```
kubectl port-forward deployment.apps/myapp 3000:3000
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl port-forward deployment.app
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

2. Uygulamayı `3000` Portu'nda başlatın ve uygulama çıktısını görüntüleyin.
    1. Skills Network Toolbox simgesine tıklayın.
    2. Uygulamayı Başlat'a tıklayın.
    3. Port Numarasını girin.
    4. Ekran görüntüsünde gösterildiği gibi başlat düğmesine tıklayın.



3. `Hello from MyApp. Your app is up!` mesajını görmelisiniz.



MyApp

Hello from MyApp. Your app is up!

4. Devam etmeden önce sunucuyu durdurmak için `CTRL + C` tuşlarına basın.
5. Uygulamayı internete açmak için bir ClusterIP hizmeti oluşturun:

```
kubectl expose deployment/myapp
```

```
^Ctheia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl expose deployment/myapp
service/myapp exposed
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

Alıştırma 2: Dikey Pod Autoscaler (VPA) Uygulama

Dikey Pod Autoscaler (VPA), bir pod içinde çalışan konteynerler için kaynak taleplerini ve sınırlarını yönetmenize yardımcı olur. Gözlemlenen kaynak kullanımına dayalı olarak CPU ve bellek taleplerini ve sınırlarını otomatik olarak ayarlayarak pod'ların verimli bir şekilde çalışması için gerekli kaynaklara sahip olmasını sağlar.

Adım 1: Bir VPA yapılandırması oluşturun

`myapp` dağıtımı için kaynak taleplerini ve sınırlarını otomatik olarak ayarlamak üzere bir Dikey Pod Autoscaler (VPA) yapılandırması oluşturacaksınız.

Aşağıdaki içeriğe sahip `vpa.yaml` dosyasını keşfedin:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: myvpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind: Deployment
    name: myapp
  updatePolicy:
    updateMode: "Auto"  # VPA will automatically update the resource requests and limits
```

Açıklama

Bu YAML dosyası, `myapp` dağıtımı için bir VPA yapılandırmasını tanımlar. `updateMode: "Auto"` ayarı, VPA'nın bu dağıtımdaki podlar için kaynak taleplerini ve sınırlarını gözlemlenen kullanımına göre otomatik olarak güncelleyeceği anlamına gelir.

Adım 2: VPA'yı Uygula

Aşağıdaki komutu kullanarak VPA yapılandırmasını uygulayın:

```
kubectl apply -f vpa.yaml
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl apply -f vpa.yaml
verticalpodautoscaler.autoscaling.k8s.io/myvpa created
```

Adım 3: VPA'nın ayrıntılarını al

1. Oluşturulan VPA'yı al:

   ```
   kubectl get vpa
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get vpa
NAME    MODE   CPU   MEM       PROVIDED   AGE
myvpa   Auto   25m   262144k   True       29s
```

Bu çıktı şunu gösteriyor:

- `myvpa` adındaki VPA, otomatik modda olup, yönettiği podlar için 25 milli-çekirdek CPU ve 256 MB bellek önermektedir.
- 29 saniye önce oluşturulmuş ve o zamandan beri bu önerileri sağlamaktadır.

2. VPA'nın detaylarını ve mevcut çalışma durumunu alın.

   ```
   kubectl describe vpa myvpa
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl describe vpa myvpa
Name:         myvpa
Namespace:    sn-labs-ksundararaja
Labels:       <none>
Annotations:  <none>
API Version:  autoscaling.k8s.io/v1
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2024-06-25T15:17:04Z
  Generation:          1
  Resource Version:    287538855
  UID:                 57f5fac3-8720-4340-b877-3831490fb03f
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         myapp
  Update Policy:
    Update Mode:  Auto
Status:
  Conditions:
    Last Transition Time:  2024-06-25T15:17:20Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  myapp
      Lower Bound:
        Cpu:     25m
        Memory:  262144k
      Target:
        Cpu:     25m
        Memory:  262144k
      Uncapped Target:
        Cpu:     25m
        Memory:  262144k
      Upper Bound:
        Cpu:     60m
        Memory:  262144k
```

Açıklama

`kubectl describe vpa myvpa` çıktısı CPU ve bellek için öneriler sunmaktadır:

Kaynak	Tanım
Alt Sınır	VPA'nın önerdiği minimum kaynaklar.
Hedef	VPA'nın önerdiği optimal kaynaklar.

Kaynak	Tanım
Sınırsız Hedef	Önceden tanımlanmış herhangi bir sınır olmadan hedef.
Üst Sınır	VPA'nın önerdiği maksimum kaynaklar.

Kaynak	CPU	Bellek
Alt Sınır	25m	256MiB (262144KiB)
Hedef	25m	256MiB
Sınırsız Hedef	25m	256MiB
Üst Sınır	671m	1.34GiB (1438074878KiB)

Bu öneriler, VPA'nın doğru çalıştığını ve gözlemlenen kullanıma dayalı hedef değerler sağladığını göstermektedir.

Alıştırma 3: Yatay Pod Otomatik Ölçekleyici (HPA) Uygulama

Yatay Pod Otomatik Ölçekleyici (HPA), gözlemlenen CPU/ram kullanımı veya diğer özel metriklere dayalı olarak pod kopyalarının sayısını otomatik olarak ölçeklendirir. VPA, bireysel podlar için kaynak taleplerini ve sınırlarını ayarlar. Ancak, HPA yükü yönetmek için pod kopyalarının sayısını değiştirir.

Adım 1: HPA yapılandırması oluşturun

`myapp` dağıtımının kopya sayısını CPU kullanımı temelinde ölçeklendirmek için bir Yatay Pod Otomatik Ölçekleyici (HPA) yapılandırması oluşturacaksınız.

Aşağıdaki içeriğe sahip `hpa.yaml` dosyasını keşfedin:

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: myhpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp
  minReplicas: 1         # Minimum number of replicas
  maxReplicas: 10        # Maximum number of replicas
  targetCPUUtilizationPercentage: 5  # Target CPU utilization for scaling
```

Açıklama

Bu YAML dosyası, `myapp` dağıtımı için bir Yatay Pod Otomatik Ölçekleyici (HPA) yapılandırmasını tanımlar. HPA, tüm podlar arasındaki ortalama CPU kullanımının %5 civarında kalmasını sağlayacaktır. Kullanım daha yüksekse, HPA çoğaltıcı sayısını artıracak, daha düşükse belirtilen 1 ile 10 çoğaltıcı aralığında çoğaltıcı sayısını azaltacaktır.

Adım 2: HPA'yı Yapılandırın

HPA yapılandırmasını uygulayın:

```
kubectl apply -f hpa.yaml
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/myhpa created
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

Adım 3: HPA'yı Doğrulayın

Oluşturulan HPA kaynağının durumunu öğrenmek için aşağıdaki komutu çalıştırın:

```
kubectl get hpa myhpa
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get hpa myhpa
NAME    REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
myhpa   Deployment/myapp   0%/5%     1         10        1          61s
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

Bu komut, mevcut ve hedef CPU kullanımına ve kopya sayısına dair ayrıntılar sağlar.

Adım 4: Kubernetes proxy’sini başlatın

Başka bir terminal açın ve Kubernetes proxy’sini başlatın:

```
kubectl proxy
```

```
theia@theiadocker-ksundararaja:/home/project$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Adım 5: Spam gönderin ve uygulamanın yükünü artırın

Yeni bir terminal açın ve uygulamayı birden fazla istekle spamlamak için aşağıdaki komutu girin:

```
for i in `seq 100000`; do curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/myapp/proxy; done
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple App - v1</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <h1>MyApp</h1>
    <p>Hello from MyApp. Your app is up!</p>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple App - v1</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <h1>MyApp</h1>
    <p>Hello from MyApp. Your app is up!</p>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple App - v1</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <h1>MyApp</h1>
    <p>Hello from MyApp. Your app is up!</p>
</body>
</html>
```

Yeni terminalde daha fazla komutla devam edin.

Adım 6: Otomatik ölçeklemenin etkisini gözlemleyin

1. Otomatik ölçekleme ile birlikte replikaların arttığını gözlemlemek için aşağıdaki komutu çalıştırın:

```
kubectl get hpa myhpa --watch
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get hpa myhpa --watch
NAME    REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
myhpa   Deployment/myapp   95%/5%    1         10        5          3m26s
myhpa   Deployment/myapp   45%/5%    1         10        10         3m30s
myhpa   Deployment/myapp   25%/5%    1         10        10         4m16s
myhpa   Deployment/myapp   19%/5%    1         10        10         4m31s
```

2. Uygulamanızın otomatik ölçeklendiğini gösteren, kopya sayısında bir artış göreceksiniz.

3. Bu komutu `CTRL + C` tuşlarına basarak sonlandırın.

Adım 7: HPA'nın detaylarını gözlemleyin

1. Aşağıdaki komutu çalıştırarak yatay pod otomatik ölçeklendiricisinin detaylarını gözlemleyin:

```
kubectl get hpa myhpa
```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get hpa myhpa
NAME    REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
myhpa   Deployment/myapp   16%/5%    1         10        10         5m17s
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$
```

2. Şimdi, kopya sayısının arttığını fark edeceksiniz.

3. Diğer iki terminalde çalışan proxy ve yük oluşturma komutlarını `CTRL + C` tuşlarına basarak durdurun.

Egzersiz 4: Bir Gizli Oluştur ve dağıtımı güncelle

Kubernetes Secrets, şifreler, OAuth jetonları ve SSH anahtarları gibi hassas bilgileri güvenli bir şekilde saklamanıza ve yönetmenize olanak tanır. Gizli bilgiler base64 ile kodlanmıştır ve uygulamalarınızda ortam değişkenleri olarak veya dosya olarak monte edilerek kullanılabilir.

Adım 1: Bir Gizli Oluştur

`secret.yaml` dosyasının içeriğini keşfedin:

```
apiVersion: v1
kind: Secret
metadata:
  name: myapp-secret
type: Opaque
data:
  username: bXl1c2VybmFtZQ==
  password: bXlwYXNzd29yZA==
```

Açıklama

Bu YAML dosyası, `mysecret` adında bir gizli anahtar tanımlar ve iki anahtar-değer çifti içerir: `username` ve `password`. Değerler base64 kodlu dizelerdir.

Adım 2: Dağıtımı gizli anahtarı kullanacak şekilde güncelleyin

`deployment.yaml` dosyasının sonuna aşağıdaki satırları ekleyin:

```
env:
- name: MYAPP_USERNAME
  valueFrom:
    secretKeyRef:
      name: myapp-secret
      key: username
- name: MYAPP_PASSWORD
  valueFrom:
    secretKeyRef:
      name: myapp-secret
      key: password
```

Açıklama

- `name:` - Ortam değişkenlerini tanımlar: 'MYAPP_USERNAME' ve 'MYAPP_PASSWORD', sırasıyla.
- `valueFrom:` - Ortam değişkeninin değerinin sabit kodlanmak yerine başka bir yerden alınacağını belirtir.
- `secretKeyRef:` - Ortam değişkeninin değerinin bir Kubernetes sırrından gelmesi gerektiğini gösterir.
- `name: myapp-secret` - Değeri almak için 'myapp-secret' sırrının adını belirtir.
- `key:` - Sır içinde hangi anahtarın 'MYAPP_USERNAME' ve 'MYAPP_PASSWORD' ortam değişkenlerinin değeri için kullanılacağını belirtir.

Bu güncellemelerle, `myapp` uygulaması artık gerekli kimlik bilgilerini almak için bu ortam değişkenlerini okuyabilir, bu da onu daha güvenli ve esnek hale getirir.

Adım 3: Sırrı ve dağıtımı uygulama

1. Aşağıdaki komutu kullanarak sırrı uygulayın:

   ```
   kubectl apply -f secret.yaml
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl apply -f secret.y
secret/mysecret created
```

2. Güncellenmiş dağıtımı aşağıdaki komutla uygulayın:

   ```
   kubectl apply -f deployment.yaml
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl apply -f deployment.yaml
deployment.apps/myapp configured
```

Adım 4: Sırrı ve dağıtımı doğrulayın

Artık kullanılan sırrın ve dağıtımın uygulanıp uygulanmadığını doğrulayacaksınız.

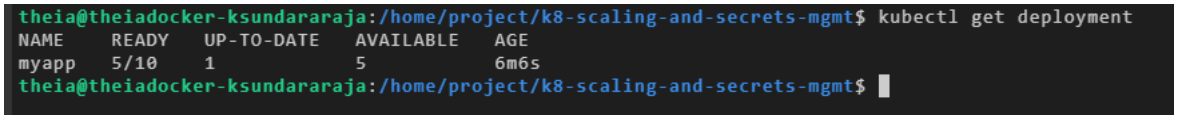
1. `myapp-secret`'in adını, türünü ve oluşturulma zaman damgasını gösteren ayrıntıları almak için aşağıdaki komutu çalıştırın:

   ```
   kubectl get secret
   ```

```
theia@theiadocker-ksundararaja:/home/project/k8s-scaling-and-secrets-mgmt$ kubectl get secret
NAME           TYPE                                  DATA   AGE
dh             kubernetes.io/dockerconfigjson        1      12m
icr            kubernetes.io/dockerconfigjson        1      12m
myapp-secret   Opaque                                2      16s
```

2. Dağıtımın durumunu, replikalar ve mevcut replikalar hakkında bilgi de dahil olmak üzere göstermek için aşağıdaki komutu çalıştırın.

```
kubectl get deployment
```



Sonuç

Bu laboratuvar, Kubernetes üzerinde `myapp` adlı bir uygulama oluşturup dağıtarak başladı.
Bunun ardından, `myapp` dağıtımı için kaynak isteklerini ve sınırlarını otomatik olarak ayarlamak üzere bir Dikey Pod Autoscaler (VPA) yapılandırdınız.
Sonrasında, CPU kullanımı temelinde `myapp` dağıtımının kopya sayısını ölçeklendirmek için Yatay Pod Autoscaler (HPA) uyguladınız.
Son olarak, bir Secret oluşturdunuz ve `myapp` dağıtımını bunu kullanacak şekilde güncellediniz.

Author(s)

Nikesh Kumar