Uygulama Laboratuvarı: Kubernetes Nesnelerine Giriş

Gerekli tahmini süre: 45 dakika

Bu uygulama laboratuvarı, Kubernetes ile pratik deneyim sağlamak için tasarlanmıştır ve hizmet oluşturma, çeşitli kubectl komutlarını kullanma ve StatefulSet ve DaemonSet dağıtma konularına odaklanmaktadır.

Hedefler

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Bir Kubernetes Servisi oluşturmak
- Çeşitli `kubectl` komutlarını kullanmak
- Durumlu uygulamaları yönetmek için bir StatefulSet dağıtmak
- Tüm düğümlerde tek bir pod çalıştırmak için bir DaemonSet uygulamak

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

Ortamı Kurma

Menüyü kullanarak bir terminal penceresi açın: `Terminal > Yeni Terminal`.

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

Adım 1: kubectl Sürümünü Doğrulayın

İlerlemeye geçmeden önce, kubectl'in kurulu ve düzgün yapılandırılmış olduğundan emin olun. kubectl'in sürümünü kontrol etmek için aşağıdaki komutu çalıştırın:

```
kubectl version
```

Aşağıdaki çıktıyı görmelisiniz, ancak sürümler farklı olabilir:

Görev 1: nginx görüntüsü kullanarak bir Kubernetes Servisi oluşturma

Popüler bir açık kaynak web sunucusu olan **nginx**, yüksek performansı, kararlılığı ve düşük kaynak kullanımı ile bilinir. Ayrıca bir ters proxy, yük dengeleyici ve HTTP önbelleği olarak da işlev görebilir.

nginx görüntüsü kullanarak bir Kubernetes Servisi oluşturmak, Kubernetes kümesindeki diğer bileşenlerin veya dış kullanıcıların pod'larda çalışan nginx uygulamasına erişmesine olanak tanıyan bir ağ katmanının kurulmasını içerir. Kubernetes'te nginx'i bir hizmet olarak çalıştırmak için şu adımları izleyebilirsiniz:

1. **nginx görüntüsünü kullanarak `my-deployment1` adında bir Dağıtım oluşturun**

   ```
   kubectl create deployment my-deployment1 --image=nginx
   ```

`kubectl`: Kubernetes API'si ile etkileşimde bulunmak için kullanılan komut satırı aracı.

`create deployment`: Kubernetes'e yeni bir Dağıtım oluşturmak istediğinizi bildirir. Dağıtım, belirli sayıda kopyanın çalışmasını ve güncellenmesini sağlayan bir Kubernetes nesnesidir.

`my-deployment1`: Oluşturulan Dağıtımın adıdır. Bu durumda, Dağıtım my-deployment1 olarak adlandırılmıştır.

`--image=nginx`: Bu Dağıtım tarafından yönetilen Pod'lar için kullanılan konteyner görüntüsünü belirtir. nginx görüntüsü, popüler bir web sunucusu ve ters proxy sunucusudur.

Bu, nginx görüntüsünü kullanan my-deployment1 adında bir Dağıtım oluşturur. Dağıtımlar, uygulamaların dağıtımını ve ölçeklenmesini yönetir.

2. **Gerekli ayarları uygulamak için Dağıtımı yamanlayın**

   Dağıtım yapılandırmasını güncellemek için aşağıdaki komutu çalıştırın:

```
kubectl patch deployment my-deployment1 --type=json -p='[
  {
    "op": "replace",
    "path": "/spec/template/spec/containers/0/image",
    "value": "nginxinc/nginx-unprivileged:latest"
  },
  {
    "op": "add",
    "path": "/spec/template/spec/containers/0/ports",
    "value": [{"containerPort": 8080}]
  },
  {
    "op": "add",
    "path": "/spec/template/spec/securityContext",
    "value": {
      "runAsNonRoot": true,
      "runAsUser": 101,
      "seccompProfile": {"type": "RuntimeDefault"}
    }
  },
  {
    "op": "add",
    "path": "/spec/template/spec/containers/0/securityContext",
    "value": {
      "allowPrivilegeEscalation": false,
      "capabilities": {"drop": ["ALL"]}
    }
  }
]'
```

   Bu komut, gerekli yapılandırma ile Dağıtımı günceller.

3. **Dağıtımı bir hizmet olarak yayınlayın**

```
kubectl expose deployment my-deployment1 --port=80 --type=NodePort --name=my-service1
```

Bu, my-deployment1 Dağıtımını my-service1 adında bir Hizmet olarak açar ve onu NodePort üzerinden 80 numaralı portta erişilebilir hale getirir. NodePort hizmetleri, dış trafiğin hizmete erişmesine olanak tanır.

4. **Varsayılan ad alanındaki tüm hizmetleri listeleyin. Hizmetler, bir dizi pod'a erişim için sabit bir IP adresi ve DNS adı sağlar.**

```
kubectl get services
```

Bu komut, varsayılan ad alanındaki tüm hizmetleri listeler, nginx-service dahil olmak üzere, ve ClusterIP, NodePort ve hedef port gibi detaylar sağlar.

Bu adımları takip ederek, kümeniz içinde çalışan nginx pod'larına trafik yönlendiren `nginx` adlı bir Kubernetes Hizmeti oluşturursunuz; böylece, atanan NodePort aracılığıyla hem dahili hem de harici olarak erişilebilir hale gelirler.

Görev 2: Kubernetes Pod'larını ve Servislerini Yönetme

1. Pod'ların listesini al

```
kubectl get pods
```

Bu komut, my-deployment1 Dağıtımı tarafından oluşturulanlar da dahil olmak üzere tüm pod'ları gösterir.

2. Etiketleri göster

`<pod-name>` kısmını gerçek pod Adı ile değiştirin:

```
kubectl get pod <pod-name> --show-labels
```

Bu komut, belirtilen pod ile ilişkili etiketleri listeleyecek ve Kubernetes kümeniz içindeki niteliklerini ve kategorilerini belirlemenize yardımcı olacaktır.

3. Pod'u etiketleyin

`<pod-name>` kısmını gerçek pod adı ile değiştirin:

```
kubectl label pods <pod-name> environment=deployment
```

Komut, Kubernetes'te belirli bir pod'u `environment=deployment` anahtar-değer çifti ile etiketlemek için kullanılır. Bu etiket, pod'ları dağıtım ortamlarına göre kategorize etmeye ve yönetmeye yardımcı olur, böylece küme içindeki Kubernetes nesnelerini organize etmek ve seçmek daha kolay hale gelir.

4. Etiketleri göster

`<pod-name>` kısmını gerçek pod adı ile değiştirin:

```
kubectl get pod <pod-name> --show-labels
```

5. `nginx` imajını kullanarak bir test pod'u çalıştırın

```
kubectl run my-test-pod --image=nginxinc/nginx-unprivileged:latest --restart=Never --dry-run=client -o yaml | \
kubectl patch -f - --dry-run=client -o yaml --type=json -p='[
  {
    "op": "add",
    "path": "/spec/securityContext",
    "value": {
      "runAsNonRoot": true,
      "runAsUser": 101,
      "seccompProfile": {"type": "RuntimeDefault"}
    }
  },
  {
    "op": "add",
    "path": "/spec/containers/0/securityContext",
    "value": {
      "allowPrivilegeEscalation": false,
      "capabilities": {"drop": ["ALL"]}
    }
  }
]' | kubectl apply -f -
```

Bu komut, Kubernetes'e "my-test-pod" adında bir pod oluşturmasını ve nginx imajını kullanarak, pod'un herhangi bir nedenle durursa otomatik olarak yeniden başlamayacağını belirtir, çünkü `--restart=Never` kullanıyoruz.

6. Logları göster

```
kubectl logs <pod-name>
```

`<pod-name>` yerine pod'un gerçek adını yazın.

Bu komut, belirtilen pod tarafından üretilen logları alır ve görüntüler, böylece sorunları gidermenizi, etkinliği izlemenizi ve pod'un davranışı hakkında bilgi toplamanızı sağlar.

Görev 3: StatefulSet Dağıtımı

Bir StatefulSet, bir dizi pod'un dağıtımını ve ölçeklenmesini yönetir ve her bir Pod için yapışkan bir kimlik tutar, böylece her Pod'un kalıcı bir kimliği ve depolaması olur.

1. **`statefulset.yaml` adında bir dosya oluşturun ve düzenleme modunda açın.**

```
touch statefulset.yaml
```

2. **`statefulset.yaml` dosyasını açın, aşağıdaki kodu ekleyin ve dosyayı kaydedin:**

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: my-statefulset
spec:
  serviceName: "nginx"
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 101
        fsGroup: 101
        seccompProfile:
          type: RuntimeDefault
      containers:
      - name: nginx
        image: nginxinc/nginx-unprivileged:latest
        ports:
        - containerPort: 8080
          name: web
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
```

```
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

Açıklama:

- `apiVersion: apps/v1` ve `kind: StatefulSet`: Kaynağı StatefulSet olarak, stable apps/v1 API'sini kullanarak tanımlar.
- `metadata.name: my-statefulset`: StatefulSet'e bir isim verir.
- `spec.serviceName: "nginx"`: StatefulSet'i nginx adında bir başsız Servis ile ilişkilendirir.
- `spec.replicas: 3`: Üç pod kopyası oluşturur.
- `spec.selector.matchLabels`: StatefulSet'in app: nginx etiketiyle işaretlenmiş pod'ları yönetmesini sağlar.
- `spec.template`: Pod yapısını tanımlar:
  - `metadata.labels`: Tüm pod'lara app: nginx etiketini atar.
  - `spec.securityContext`: Gerekli çalışma zamanı kullanıcı, grup ve profil ayarlarını belirler.
  - `containers`: nginx konteynerini, imajını ve 8080 portundaki port eşlemesini yapılandırır ve konteyner düzeyinde güvenlik ayarlarını uygular.
- `volumeClaimTemplates`: Her bir kopya için www adında bir PersistentVolumeClaim oluşturur, her biri 1 Gi depolama isteğinde bulunur ve ReadWriteOnce erişim moduna sahiptir.

3. StatefulSet yapılandırmasını uygulayın.

```
kubectl apply -f statefulset.yaml
```

Bu komut, Kubernetes'e YAML dosyasında tanımlanan kaynakları oluşturmasını söyler.

4. **StatefulSet'in oluşturulduğunu doğrulayın.**

```
kubectl get statefulsets
```

StatefulSet'i uyguladıktan sonra, StatefulSet'in oluşturulduğunu ve çalıştığını doğrulamalısınız. Bu, kubectl get komutunu kullanarak yapılabilir.

Bu adımları izleyerek, Kubernetes'te bir StatefulSet'i başarıyla uygulayabilirsiniz. kubectl apply komutu StatefulSet'i oluşturmak için kullanılır ve kubectl get komutu, StatefulSet'in beklendiği gibi çalıştığını doğrulamanıza yardımcı olur.

Görev 4: DaemonSet Uygulaması

Bir DaemonSet, belirli bir Pod'un kümedeki tüm (veya bazı) düğümlerde çalışmasını sağlar. Bu, günlük toplama, izleme veya ağ hizmetleri gibi düğümler arasında temel hizmetler sağlayan sistem düzeyindeki uygulamaları dağıtmak için özellikle faydalıdır.

1. **`daemonset.yaml` adında bir dosya oluşturun ve düzenleme modunda açın:**

```
touch daemonset.yaml
```

2. **`daemonset.yaml` adında bir dosya oluşturun ve düzenleme modunda açın.**

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: my-daemonset
spec:
  selector:
    matchLabels:
      name: my-daemonset
  template:
    metadata:
      labels:
        name: my-daemonset
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 101
        seccompProfile:
          type: RuntimeDefault
      containers:
      - name: my-daemonset
        image: nginxinc/nginx-unprivileged:latest
        ports:
        - containerPort: 8080
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
```

Açıklama

- `apiVersion: apps/v1` ve `kind: DaemonSet`: Kaynağı, kararlı apps/v1 API'sini kullanarak bir DaemonSet olarak tanımlar.
- `metadata.name: my-daemonset`: DaemonSet'in adını ayarlar.
- `spec.selector.matchLabels`: DaemonSet'in name: my-daemonset etiketiyle etiketlenmiş pod'ları yönettiğinden emin olur.
- `spec.template.metadata.labels`: Pod şablonuna aynı etiketi uygular, böylece seçiciyle eşleşir.
- `spec.template.spec.securityContext`: Pod düzeyindeki çalışma zamanı ayarlarını tanımlar.
- `spec.template.spec.containers`: Konteyneri, görüntüsünü, bağlantı noktasını ve konteyner düzeyindeki güvenlik ayarlarını yapılandırır.

3. DaemonSet'i Uygula

```
kubectl apply -f daemonset.yaml
```

Bu komut, Kubernetes'e daemonset.yaml dosyasında tanımlanan yapılandırmayı uygulamasını söyler. Apply komutu, YAML dosyasında sağlanan yapılandırmaya göre Kubernetes kaynaklarını oluşturmak veya güncellemek için kullanılır.

4. DaemonSet'in oluşturulduğunu doğrula

```
kubectl get daemonsets
```

`kubectl get daemonsets` komutunun çıktısı, Kubernetes kümenizde "my-daemonset" adlı DaemonSet hakkında bilgi verir.

- **NAME**: DaemonSet'in adı, bu durumda "my-daemonset".
- **DESIRED**: İstenilen DaemonSet pod'larının sayısı. Sizin durumunuzda, bu 7 olarak ayarlanmış.
- **CURRENT**: Çalışan mevcut DaemonSet pod'larının sayısı. Şu anda 6 pod'un çalıştığını gösteriyor.
- **READY**: Kullanıma hazır ve mevcut olan DaemonSet pod'larının sayısı. Tüm 6 çalışan pod hazır.
- **UP-TO-DATE**: En son yapılandırma ile güncel olan DaemonSet pod'larının sayısı.
- **AVAILABLE**: Kullanıma hazır olan DaemonSet pod'larının sayısı.
- **NODE SELECTOR**: DaemonSet'in küme içindeki hangi düğümlerde çalışması gerektiğini belirtir. Bu durumda, `<none>` olarak ayarlanmış, yani DaemonSet belirli düğümlerle sınırlı değil.
- **AGE**: DaemonSet'in yaşı, ne kadar süredir çalıştığını gösterir.

Sonuç

Tebrikler! Kubernetes ile ilgili uygulama laboratuvarını tamamladınız. Bir Kubernetes Servisi oluşturdunuz, çeşitli kubectl komutları kullandınız, durum bilgisi uygulamaları için StatefulSets dağıttınız ve küme düğümleri arasında eşit pod dağıtımı için DaemonSets'i uyguladınız.

Yazar(lar)

Manvi Gupta