Way2Automation: Webservice LoadTest mit HPA

Der **Horizontal Pod Autoscaler (HPA)** ist eine Kubernetes-Ressource, die dafür sorgt, dass die Anzahl der Pods in einem Deployment oder einem ReplicaSet automatisch an die aktuelle Last angepasst wird. Der HPA überwacht dabei die Auslastung bestimmter Metriken, wie z. B. die **CPU- oder Speicherauslastung** der Pods, und skaliert die Anzahl der Pods nach oben oder unten, um die Anforderungen zu erfüllen.

Wie funktioniert HPA?

1. **Metrik-Überwachung**: Der HPA überwacht kontinuierlich Metriken wie CPU- oder Speicherauslastung (sofern ein Metrics Server installiert ist). Andere Metriken, wie benutzerdefinierte Anwendungsmetriken, können ebenfalls überwacht werden, wenn sie entsprechend konfiguriert sind.
2. **Skalierungsentscheidungen**: Wenn die Zielmetriken, z. B. die CPU-Auslastung, über einem festgelegten Schwellenwert liegen, erhöht der HPA die Anzahl der Pods. Wenn die Last abnimmt und die Metriken unter das Zielniveau fallen, wird die Anzahl der Pods reduziert.
3. **Automatisches Anpassen**: Der HPA passt die Anzahl der laufenden Pods basierend auf den aktuellen Anforderungen automatisch an, ohne dass ein manuelles Eingreifen erforderlich ist.

Warum braucht man HPA?

1. **Effiziente Ressourcennutzung**: Der HPA hilft dabei, Ressourcen effizient zu nutzen. Wenn die Last niedrig ist, skaliert er die Anzahl der Pods herunter, um Ressourcen im Cluster freizugeben. Bei höherer Last skaliert er nach oben, sodass die Anwendung in der Lage ist, mehr Anfragen zu verarbeiten, ohne dass Benutzer eine Verzögerung oder Leistungseinbußen erleben.
2. **Kostenreduktion**: Da HPA Ressourcen dynamisch anpasst, kann er dazu beitragen, die Cloud-Kosten zu reduzieren. Indem er nur dann mehr Ressourcen anfordert, wenn sie benötigt werden, vermeidet man, dass zu viele Ressourcen ständig belegt und bezahlt werden, wenn sie nicht notwendig sind.
3. **Automatische Skalierung für variable Lasten**: Viele Anwendungen erleben Lastspitzen zu bestimmten Zeiten, z. B. während der Geschäftszeiten oder bei saisonalen Events. Mit HPA kann die Anwendung automatisch auf diese Lastspitzen reagieren, ohne dass jemand manuell eingreifen muss.
4. **Verbesserte Benutzererfahrung**: Durch die automatische Skalierung gewährleistet HPA eine konstante Performance der Anwendung. Nutzer spüren keine Verzögerungen, auch wenn die Last steigt, da der HPA zusätzliche Pods bereitstellt, um die Anfragen zu verteilen.

1. Voraussetzungen

Stelle sicher, dass die folgenden Voraussetzungen erfüllt sind:

- **Kubernetes Cluster**: Ein laufender Cluster.
- **kubectl CLI**: Mit Zugriff auf den Cluster.
- **Metrics Server**: Für den Horizontal Pod Autoscaler (HPA) ist ein Metrics Server erforderlich. Wenn dieser noch nicht installiert ist:

```
kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/
download/components.yaml
```

2. Deployment- und Load-Test-Konfigurationen anwenden

Speichere die Konfigurationen in folgenden Dateien und wende sie im Cluster an.

busybox-load-tester.yaml

Die `busybox-load-tester` Datei, die den Lasttest durchführt, sollte wie folgt aussehen:

```
yaml
Code kopieren
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-load-tester
spec:
  replicas: 10
  selector:
    matchLabels:
      app: busybox-load-tester
  template:
    metadata:
      labels:
        app: busybox-load-tester
    spec:
      containers:
        - name: busybox
          image: busybox
          command: ["/bin/sh", "-c"]
          args:
            - |
              while true; do
                STARTKAPITAL=$(shuf -i 0-10000 -n 1)
                ZINSSATZ=$(shuf -i 3-10 -n 1)
                LAUFZEIT=$(shuf -i 0-1000 -n 1)

                wget -q -O- http://way2millionapp-service:8080
                sleep 1
                wget -q -O- --post-data="{\"startkapital\":
$STARTKAPITAL, \"zinssatz\": $ZINSSATZ, \"laufzeit\": $LAUFZEIT}" \
                --header="Content-Type: application/json" http://way2millionapp-
service:8080/berechne
              done
```

deployment_remoterepo.yaml

Die `way2millionapp` Anwendungskonfiguration:

```
yaml
Code kopieren
apiVersion: apps/v1
kind: Deployment
metadata:
  name: way2millionapp
  labels:
    app: way2millionapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: way2millionapp
  template:
    metadata:
      labels:
        app: way2millionapp
    spec:
      containers:
        - name: way2millionapp
          image: martinfailsfast/way2millionapp:latest
          imagePullPolicy: IfNotPresent
          ports:
          - containerPort: 8080
      resources:
        requests:
          cpu: "150m"
          memory: "128Mi"
        limits:
          cpu: "300m"
          memory: "256Mi"
      imagePullSecrets:
      - name: regcred
```

Deployment anwenden

Mit folgenden Befehlen werden die Deployments in den Cluster gebracht:

```
kubectl apply -f deployment_remoterepo.yaml
kubectl apply -f busybox-load-tester.yaml
```

3. Autoscaler manuell aktivieren

Anstatt eine YAML-Datei für den HPA zu erstellen, kann der Autoscaler auch direkt mit dem folgenden `kubectl`-Befehl aktiviert werden:

```
kubectl autoscale deployment way2millionapp --min=1 --max=5 --cpu-percent=50
```

Dieser Befehl erstellt einen HPA für das Deployment `way2millionapp` und skaliert die Pods automatisch, wenn die CPU-Auslastung über 50% liegt. Hierbei:

- **--min=1**: Legt die minimale Anzahl der Pods fest (1).
- **--max=5**: Legt die maximale Anzahl der Pods fest (5).
- **--cpu-percent=50**: Ziel-CPU-Auslastung, bei der zusätzliche Pods bereitgestellt werden sollen.

4. Überwachung des HPA und der Skalierung

Verwende die folgenden Befehle, um den HPA und die Pod-Skalierung zu überwachen:

HPA Status überwachen

Überwache den Status des HPA und die CPU-Auslastung mit:

```
kubectl get hpa way2millionapp -w
```

Pod-Status überwachen

Überwache die Anzahl der laufenden Pods für `way2millionapp`:

```
kubectl get pods -l app=way2millionapp -w
```

Ressourcenverbrauch und Details prüfen

Für detaillierte Informationen zur CPU- und Speicherauslastung der Pods verwende:

```
kubectl top pod -l app=way2millionapp
```

Logs der Anwendung ansehen

Um die eingehenden Anfragen und die Aktivitäten in den Logs der `way2millionapp`-Pods zu sehen, führe diesen Befehl aus:

```
kubectl logs -l app=way2millionapp -f
```

5. Lasttest und Skalierung beobachten

Da der `busybox-load-tester` kontinuierlich Anfragen an `way2millionapp` sendet, sollte die CPU-Last auf dem `way2millionapp` Deployment steigen. Der HPA reagiert darauf und skaliert die Anzahl der Pods, um die Anfragen zu bewältigen.

Beobachte die Anzahl der aktiven Pods und die CPU-Auslastung, um sicherzustellen, dass der HPA ordnungsgemäß skaliert.

6. Bereinigung nach dem Test

Um die erstellten Ressourcen zu löschen, führe folgende Befehle aus:

```
kubectl delete -f deployment_remoterepo.yaml
kubectl delete -f busybox-load-tester.yaml
kubectl delete hpa way2millionapp
```