Escalado Automático Minikube

Para desplegar un servicio en Minikube con escalado automático, puedes seguir estos pasos. Vamos a crear un ejemplo simple utilizando un Deployment y un Horizontal Pod Autoscaler (HPA) en Kubernetes.

1. Iniciar Minikube

Primero, asegúrate de que Minikube esté corriendo:

```
minikube start
```

2. Crear un Deployment

Crea un archivo YAML para definir un Deployment. Este Deployment desplegará una aplicación simple, como un servidor web Nginx.

```yaml
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "100m"
```

Aplica el Deployment:

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

3. Crear un Horizontal Pod Autoscaler (HPA)

Ahora, crea un Horizontal Pod Autoscaler para escalar automáticamente el número de réplicas del Deployment basado en el uso de CPU.

```yaml
# hpa.yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

Aplica el HPA:

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

4. Verificar el HPA

Puedes verificar el estado del HPA con el siguiente comando:

```
kubectl get hpa
```

Esto te mostrará el uso actual de CPU y el número de réplicas que están corriendo.

5. Generar carga para probar el escalado

Para probar el escalado automático, puedes generar carga en los pods. Una forma simple de hacer esto es usando kubectl run para crear un pod temporal que haga solicitudes a los pods de Nginx.

```
kubectl run -i --tty load-generator --image=busybox -- /bin/sh -c "while true; do wget -q -O- http://nginx-deployment; done"
```

6. Monitorear el escalado

Mientras se genera la carga, puedes monitorear cómo el HPA escala el número de réplicas:

```
kubectl get hpa -w
```

Verás cómo el número de réplicas aumenta a medida que el uso de CPU supera el umbral del 50%.

7. Limpiar

Cuando hayas terminado, puedes eliminar los recursos creados:

```
kubectl delete -f deployment.yaml
kubectl delete -f hpa.yaml
```

Y detener Minikube:

```
minikube stop
```

Este es un ejemplo básico de cómo configurar el escalado automático en Minikube. Puedes ajustar los valores de `minReplicas`, `maxReplicas`, y `targetCPUUtilizationPercentage` según tus necesidades.

Otras Métricas

Además de targetCPUUtilizationPercentage, Kubernetes permite escalar automáticamente basándose en otras métricas personalizadas y estándar. A partir de Kubernetes 1.23, el **Horizontal Pod Autoscaler (HPA)** soporta métricas de recursos, métricas personalizadas y métricas externas. Aquí te explico las opciones disponibles:

---

1. Métricas de Recursos (Resource Metrics)

Estas son métricas nativas de Kubernetes relacionadas con el uso de recursos de los Pods. Las más comunes son:

- **CPU**: Uso de CPU (como en el ejemplo anterior).
- **Memoria**: Uso de memoria (RAM).

Para usar la memoria como métrica, puedes configurar el HPA de la siguiente manera:

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 50
```

En este caso, el HPA escalará los Pods si el uso promedio de memoria supera el 50%.

---

2. Métricas Personalizadas (Custom Metrics)

Las métricas personalizadas permiten escalar basándose en métricas específicas de tu aplicación, como:

- Número de solicitudes por segundo (RPS).
- Longitud de una cola de mensajes.
- Número de conexiones activas.
- Cualquier otra métrica que exponga tu aplicación a través de un endpoint (por ejemplo, Prometheus).

Para usar métricas personalizadas, necesitas un **adaptador de métricas** como **Prometheus Adapter** o **Kubernetes Metrics Server**.

Ejemplo de HPA con una métrica personalizada:

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Pods
    pods:
      metric:
        name: requests_per_second
      target:
        type: AverageValue
        averageValue: 100
```

En este caso, el HPA escalará los Pods si el promedio de solicitudes por segundo (`requests_per_second`) supera 100.

---

3. Métricas Externas (External Metrics)

Las métricas externas permiten escalar basándose en métricas que provienen de fuentes externas a Kubernetes, como:

- Número de mensajes en una cola de AWS SQS.
- Latencia de un servicio externo.
- Uso de un recurso en la nube (por ejemplo, AWS CloudWatch).

Ejemplo de HPA con una métrica externa:

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: External
    external:
      metric:
        name: sqs_queue_length
      target:
        type: AverageValue
        averageValue: 10
```

En este caso, el HPA escalará los Pods si la longitud de la cola de SQS (sqs_queue_length) supera 10.

4. Métricas de Objetos (Object Metrics)

Estas métricas están relacionadas con un objeto específico en Kubernetes, como un Ingress o un Service. Por ejemplo, puedes escalar basándote en el número de solicitudes HTTP por segundo que recibe un Ingress.

Ejemplo de HPA con métrica de objeto:

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Object
    object:
      metric:
        name: requests_per_second
      describedObject:
        apiVersion: networking.k8s.io/v1
        kind: Ingress
        name: my-ingress
      target:
        type: Value
        value: 100
```

En este caso, el HPA escalará los Pods si el número de solicitudes por segundo en el Ingress supera 100.

---

5. Múltiples Métricas

Puedes combinar varias métricas en un solo HPA. Kubernetes evaluará todas las métricas y escalará si alguna de ellas supera su umbral.

Ejemplo de HPA con múltiples métricas:

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 50
  - type: Pods
    pods:
      metric:
        name: requests_per_second
      target:
        type: AverageValue
        averageValue: 100
```

En este caso, el HPA escalará si el uso de CPU supera el 50%, el uso de memoria supera el 50%, o el número de solicitudes por segundo supera 100.

6. Consideraciones Adicionales

- **Prometheus**: Si usas Prometheus para monitorear tus aplicaciones, puedes integrarlo con Kubernetes para exponer métricas personalizadas y externas.
- **Keda (Kubernetes Event-Driven Autoscaling)**: Es una herramienta que facilita el escalado automático basado en eventos, como colas de mensajes (Kafka, RabbitMQ, etc.).
- **Cool-down Periods**: Kubernetes tiene períodos de enfriamiento para evitar escalados demasiado rápidos o frecuentes.