Horizontal Pod Autoscaler

PDF (eks-ug.pdf#horizontal-pod-autoscaler)
Kindle (https://www.amazon.com/dp/B07GQDSDPZ)
RSS (doc-history.rss)

The Kubernetes Horizontal Pod Autoscaler (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/) automatically scales the number of pods in a deployment, replication controller, or replica set based on that resource's CPU utilization. This can help your applications scale out to meet increased demand or scale in when resources are not needed, thus freeing up your nodes for other applications. When you set a target CPU utilization percentage, the Horizontal Pod Autoscaler scales your application in or out to try to meet that target.

The Horizontal Pod Autoscaler is a standard API resource in Kubernetes that simply requires that a metrics source (such as the Kubernetes metrics server) is installed on your Amazon EKS cluster to work. You do not need to deploy or install the Horizontal Pod Autoscaler on your cluster to begin scaling your applications. For more information, see Horizontal Pod Autoscaler (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/) in the Kubernetes documentation.

Use this topic to prepare the Horizontal Pod Autoscaler for your Amazon EKS cluster and to verify that it is working with a sample application.

Note

This topic is based on the Horizontal pod autoscaler walkthrough (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/) in the Kubernetes documentation.

Prerequisites

- You have an existing Amazon EKS cluster. If you don't, see Getting started with Amazon EKS (./getting-started.html) .
- You have the Kubernetes Metrics Server installed. For more information, see Installing the Kubernetes Metrics Server (./metrics-server.html) .
- You are using a `kubectl` client that is configured to communicate with your Amazon EKS cluster (./getting-started-console.html#eks-configure-kubectl) .

Run a Horizontal Pod Autoscaler test application

In this section, you deploy a sample application to verify that the Horizontal Pod Autoscaler is working.

Note

This example is based on the Horizontal pod autoscaler walkthrough (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/) in the Kubernetes documentation.

To test your Horizontal Pod Autoscaler installation

1. Deploy a simple Apache web server application with the following command.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   This Apache web server pod is given a 500 millicpu CPU limit and it is serving on port 80.

2. Create a Horizontal Pod Autoscaler resource for the `php-apache` deployment.

```
kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

This command creates an autoscaler that targets 50 percent CPU utilization for the deployment, with a minimum of one pod and a maximum of ten pods. When the average CPU load is below 50 percent, the autoscaler tries to reduce the number of pods in the deployment, to a minimum of one. When the load is greater than 50 percent, the autoscaler tries to increase the number of pods in the deployment, up to a maximum of ten. For more information, see How does the Horizontal Pod Autoscaler work? (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#how-does-the-horizontal-pod-autoscaler-work) in the Kubernetes documentation.

3. Describe the autoscaler with the following command to view its details.

```
kubectl describe hpa
```

Output:

```
Name:                                                  php-apache
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Thu, 11 Jun 2020 16:05:41 -0500
Reference:                                             Deployment/php-apache
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  <unknown> / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 0 desired
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the
target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica
count: did not receive metrics for any ready pods
Events:
  Type     Reason                        Age                From                       
Message
  ----     ------                        ----               ----                       ------
-
  Warning  FailedGetResourceMetric       42s (x2 over 57s)  horizontal-pod-autoscaler  unable
to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  42s (x2 over 57s)  horizontal-pod-autoscaler 
invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: unable
to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedGetResourceMetric       12s (x2 over 27s)  horizontal-pod-autoscaler  did
not receive metrics for any ready pods
  Warning  FailedComputeMetricsReplicas  12s (x2 over 27s)  horizontal-pod-autoscaler 
invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: did not
receive metrics for any ready pods
```

As you can see, the current CPU load is `<unknown>`, because there's no load on the server yet. The pod count is already at its lowest boundary (one), so it cannot scale in.

4. Create a load for the web server by running a container.

```
kubectl run -it --rm load-generator --image=busybox /bin/sh --generator=run-pod/v1
```

If you don't receive a command prompt after several seconds, you may need to press `Enter`. From the command prompt, enter the following command to generate load and cause the autoscaler to scale out the deployment.

```
while true; do wget -q -O- http://php-apache; done
```

5. To watch the deployment scale out, periodically run the following command in a separate terminal from the terminal that you ran the previous step in.

```
kubectl get hpa
```

Output:

```
NAME         REFERENCE                TARGETS    MINPODS    MAXPODS    REPLICAS    AGE
php-apache   Deployment/php-apache    250%/50%   1          10         5           4m44s
```

As long as actual CPU percentage is higher than the target percentage, then the replica count increases, up to 10. In this case, it's `250%`, so the number of `REPLICAS` continues to increase.

Note

It may take a few minutes before you see the replica count reach its maximum. If only 6 replicas, for example, are necessary for the CPU load to remain at or under 50%, then the load won't scale beyond 6 replicas.

6. Stop the load. In the terminal window you're generating the load in (from step 4), stop the load by holding down the `Ctrl+C` keys. You can watch the replicas scale back to 1 by running the following command again.

```
kubectl get hpa
```

Output

```
NAME         REFERENCE                TARGETS   MINPODS    MAXPODS    REPLICAS    AGE
php-apache   Deployment/php-apache    0%/50%    1          10         1           25m
```

Note

The default timeframe for scaling back down is five minutes, so it will take some time before you see the replica count reach 1 again, even when the current CPU percentage is 0 percent. The timeframe is modifiable. For more information, see Horizontal Pod Autoscaler (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/) in the Kubernetes documentation.

7. When you are done experimenting with your sample application, delete the `php-apache` resources.

```
kubectl delete deployment.apps/php-apache service/php-apache
horizontalpodautoscaler.autoscaling/php-apache
```