Vertical Pod Autoscaler

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

The Kubernetes Vertical Pod Autoscaler (https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler) automatically adjusts the CPU and memory reservations for your pods to help "right size" your applications. This adjustment can improve cluster resource utilization and free up CPU and memory for other pods. This topic helps you to deploy the Vertical Pod Autoscaler to your cluster and verify that it is working.

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) .

Deploy the Vertical Pod Autoscaler

In this section, you deploy the Vertical Pod Autoscaler to your cluster.

To deploy the Vertical Pod Autoscaler

1. Open a terminal window and navigate to a directory where you would like to download the Vertical Pod Autoscaler source code.
2. Clone the kubernetes/autoscaler (https://github.com/kubernetes/autoscaler) GitHub repository.

   ```
   git clone https://github.com/kubernetes/autoscaler.git
   ```

3. Change to the `vertical-pod-autoscaler` directory.

   ```
   cd autoscaler/vertical-pod-autoscaler/
   ```

4. (Optional) If you have already deployed another version of the Vertical Pod Autoscaler, remove it with the following command.

   ```
   ./hack/vpa-down.sh
   ```

5. If your cluster isn't in the China (Beijing) or China (Ningxia) Region, skip to step 6. Edit the manifest files in `./vertical-pod-autoscaler/deploy` using the following steps.
   1. View the manifest file or files that you downloaded and note the name of the image. Download the image locally with the following command.

      ```
      docker pull image:<tag>
      ```

2. Tag the image to be pushed to an Amazon Elastic Container Registry repository in China with the following command.

   ```
   docker tag image:<tag> <aws_account_id>.dkr.ecr.<cn-north-1>.amazonaws.com/image:<tag>
   ```

3. Push the image to a China Amazon ECR repository with the following command.

   ```
   docker push image:<tag> <aws_account_id>.dkr.ecr.<cn-north-1>.amazonaws.com/image:<tag>
   ```

4. Update the Kubernetes manifest file or files to reference the Amazon ECR image URL in your Region.

6. Deploy the Vertical Pod Autoscaler to your cluster with the following command.

   ```
   ./hack/vpa-up.sh
   ```

7. Verify that the Vertical Pod Autoscaler pods have been created successfully.

   ```
   kubectl get pods -n kube-system
   ```

   Output:

   ```
   NAME                                        READY   STATUS    RESTARTS   AGE
   aws-node-949vx                              1/1     Running   0          122m
   aws-node-b4nj8                              1/1     Running   0          122m
   coredns-6c75b69b98-r9x68                    1/1     Running   0          133m
   coredns-6c75b69b98-rt9bp                    1/1     Running   0          133m
   kube-proxy-bkm6b                            1/1     Running   0          122m
   kube-proxy-hpqm2                            1/1     Running   0          122m
   metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
   vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
   vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
   vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
   ```

Test your Vertical Pod Autoscaler installation

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

To test your Vertical Pod Autoscaler installation

1. Deploy the `hamster.yaml` Vertical Pod Autoscaler example with the following command.

   ```
   kubectl apply -f examples/hamster.yaml
   ```

2. Get the pods from the `hamster` example application.

   ```
   kubectl get pods -l app=hamster
   ```

   Output:

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Describe one of the pods to view its CPU and memory reservation.

   ```
   kubectl describe pod hamster-<c7d89d6db-rglf5>
   ```

   Output:

   ```
   Name:           hamster-c7d89d6db-rglf5
   Namespace:      default
   Priority:       0
   Node:           ip-192-168-9-44.<region-code>.compute.internal/192.168.9.44
   Start Time:     Fri, 27 Sep 2019 10:35:15 -0700
   Labels:         app=hamster
                   pod-template-hash=c7d89d6db
   Annotations:    kubernetes.io/psp: eks.privileged
                   vpaUpdates: Pod resources updated by hamster-vpa: container 0:
   Status:         Running
   IP:             192.168.23.42
   IPs:            <none>
   Controlled By:  ReplicaSet/hamster-c7d89d6db
   Containers:
     hamster:
       Container ID:  docker://e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
       Image:         k8s.gcr.io/ubuntu-slim:0.1
       Image ID:      docker-pullable://k8s.gcr.io/ubuntu-
   slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
       Port:          <none>
       Host Port:     <none>
       Command:
         /bin/sh
       Args:
         -c
         while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
       State:          Running
         Started:      Fri, 27 Sep 2019 10:35:16 -0700
       Ready:          True
       Restart Count:  0
       Requests:
         cpu:        100m
         memory:     50Mi
   ...
   ```

   You can see that the original pod reserves 100 millicpu of CPU and 50 mebibytes of memory. For this example application, 100 millicpu is less than the pod needs to run, so it is CPU-constrained. It also reserves much less memory than it needs. The Vertical Pod Autoscaler `vpa-recommender` deployment analyzes the `hamster` pods to

see if the CPU and memory requirements are appropriate. If adjustments are needed, the `vpa-updater` relaunches the pods with updated values.

4. Wait for the `vpa-updater` to launch a new `hamster` pod. This should take a minute or two. You can monitor the pods with the following command.

   Note

   If you are not sure that a new pod has launched, compare the pod names with your previous list. When the new pod launches, you will see a new pod name.

   ```
   kubectl get --watch pods -l app=hamster
   ```

5. When a new `hamster` pod is started, describe it and view the updated CPU and memory reservations.

   ```
   kubectl describe pod hamster-<c7d89d6db-jxgfv>
   ```

   Output:

   ```
   Name:           hamster-c7d89d6db-jxgfv
   Namespace:      default
   Priority:       0
   Node:           ip-192-168-9-44.<region-code>.compute.internal/192.168.9.44
   Start Time:     Fri, 27 Sep 2019 10:37:08 -0700
   Labels:         app=hamster
                   pod-template-hash=c7d89d6db
   Annotations:    kubernetes.io/psp: eks.privileged
                   vpaUpdates: Pod resources updated by hamster-vpa: container 0: cpu request,
   memory request
   Status:         Running
   IP:             192.168.3.140
   IPs:            <none>
   Controlled By:  ReplicaSet/hamster-c7d89d6db
   Containers:
     hamster:
       Container ID:  docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
       Image:         k8s.gcr.io/ubuntu-slim:0.1
       Image ID:      docker-pullable://k8s.gcr.io/ubuntu-
   slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
       Port:          <none>
       Host Port:     <none>
       Command:
         /bin/sh
       Args:
         -c
         while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
       State:          Running
         Started:      Fri, 27 Sep 2019 10:37:08 -0700
       Ready:          True
       Restart Count:  0
   ```

```
    Requests:
      cpu:        587m
      memory:     262144k
...
```

Here you can see that the CPU reservation has increased to 587 millicpu, which is over five times the original value. The memory has increased to 262,144 Kilobytes, which is around 250 mebibytes, or five times the original value. This pod was under-resourced, and the Vertical Pod Autoscaler corrected our estimate with a much more appropriate value.

6. Describe the `hamster-vpa` resource to view the new recommendation.

```
kubectl describe vpa/hamster-vpa
```

Output:

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"autoscaling.k8s.io/v1beta2","kind":"VerticalPodAutoscaler","metadata":
{"annotations":{},"name":"hamster-vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:
/apis/autoscaling.k8s.io/v1beta2/namespaces/default/verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
```

```
          Cpu:     587m
          Memory:  262144k
        Uncapped Target:
          Cpu:     587m
          Memory:  262144k
        Upper Bound:
          Cpu:     21147m
          Memory:  387863636
  Events:          <none>
```

7. When you finish experimenting with the example application, you can delete it with the following command.

```
kubectl delete -f examples/hamster.yaml
```