

VPA – Vertical Pod Autoscaler

Passo a Passo

Para configurar o Vertical Auto Scaler é preciso, primeiro, fazer o deploy do servidor que faz a coleta das métricas dos Nodes e dos Pods. Para isso:

- 1) No terminal onde está o seu cluster do Kubernetes, digite o comando “kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>”

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

- 2) Digite o comando “kubectl get pods -A” e verifique se o POD “metrics-server” está em “running”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 hack % kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-6d4b75cb6d-blprk	1/1	Running	2 (12m ago)	2d5h
kube-system	coredns-6d4b75cb6d-tnrtg	1/1	Running	2 (12m ago)	2d5h
kube-system	etcd-ortiz-control-plane	1/1	Running	2 (12m ago)	2d5h
kube-system	kindnet-zxpm5	1/1	Running	2 (12m ago)	2d5h
kube-system	kube-apiserver-ortiz-control-plane	1/1	Running	2 (12m ago)	2d5h
kube-system	kube-controller-manager-ortiz-control-plane	1/1	Running	2 (12m ago)	2d5h
kube-system	kube-proxy-nlm69	1/1	Running	2 (12m ago)	2d5h
kube-system	kube-scheduler-ortiz-control-plane	1/1	Running	2 (12m ago)	2d5h
kube-system	metrics-server-678f4bf65b-4wg8p	0/1	Running	0	30s
local-path-storage	local-path-provisioner-9cd9bd544-wxr54	1/1	Running	2 (12m ago)	2d5h

- 3) Verifique os eventos do seu pod “metrics-server”, caso exista alguma falha com “HTTP probe failed with statuscode: 500” siga o passo 4, caso contrário siga para o passo 6:

Para verificar os eventos do pod, digite: kubectl describe pods -n kube-system “nome do pod de métricas”:

```
kubectl describe pods -n kube-system metrics-server-678f4bf65b-4wg8p
```

O evento com erro é:

```
Events:
  Type     Reason      Age          From          Message
  ----     -
  Normal   Scheduled   5m35s        default-scheduler   Successfully assigned kube-system/metrics-server-678f4bf65b-4wg8p to ortiz-control-plane
  Normal   Pulled      5m34s        kubelet         Container image "k8s.gcr.io/metrics-server/metrics-server:v0.6.1" already present on machine
  Normal   Created     5m34s        kubelet         Created container metrics-server
  Normal   Started     5m33s        kubelet         Started container metrics-server
  Warning  Unhealthy   25s (x32 over 5m5s)  kubelet         Readiness probe failed: HTTP probe failed with statuscode: 500
```

- 4) Para corrigir esse erro, será preciso editar o POD “metrics-server” e adicionar as linhas abaixo na especificação do containers:

command:

- /metrics-server
- kubelet-insecure-tls
- kubelet-preferred-address-types=InternalIP

Para isso digite “kubectl edit deploy -n kube-system metrics-server”:

```
kubectl edit deploy -n kube-system metrics-server
```

Pressione a tecla “i” e insira as linhas acima logo abaixo da linha “--metric-resolution=15s”

```
spec:
  containers:
  - args:
    - --cert-dir=/tmp
    - --secure-port=4443
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --kubelet-use-node-status-port
    - --metric-resolution=15s
    command:
    - /metrics-server
    - --kubelet-insecure-tls
    - --kubelet-preferred-address-types=InternalIP
```

Cuidado com a indentação, pois as novas linhas precisam iniciar na mesma coluna das demais linhas.

Saia salvando, pressionando a tecla “esc” e depois digitando “:wq”.

- 5) Verifique se agora o POD do “metrics-server” está em execução sem nenhum erro nos últimos eventos, para isso digite “kubectl describe pods -n kube-system “nome do pod do metric server”:

```
Events:
  Type       Reason      Age   From              Message
  ---       -
  Normal    Scheduled   103s  default-scheduler Successfully assigned kube-system/metrics-server-69cb89695d-t5ll5 to ortiz-control-plane
  Normal    Pulled      103s  kubelet           Container image "k8s.gcr.io/metrics-server/metrics-server:v0.6.1" already present on machine
  Normal    Created     103s  kubelet           Created container metrics-server
  Normal    Started     103s  kubelet           Started container metrics-server
```

- 6) Veja que agora é possível executar os comandos “kubectl top pods -A” e “kubectl top nodes”. Esse comando vai trazer a utilização de CPU e memória dos seus PODs e Nodes.

```
marceloortiz@MacBook-Air-de-Marcelo-2 hack % kubectl top nodes
NAME                                CPU(cores)   CPU%   MEMORY(bytes)  MEMORY%
ortiz-control-plane                 240m         6%     747Mi          18%
```

```
marceloortiz@MacBook-Air-de-Marcelo-2 hack % kubectl top pods -A
NAMESPACE      NAME                                CPU(cores)   MEMORY(bytes)
kube-system    coredns-6d4b75cb6d-blprk          7m           12Mi
kube-system    coredns-6d4b75cb6d-tnrtg          6m           12Mi
kube-system    etcd-ortiz-control-plane           30m          48Mi
kube-system    kindnet-zxpm5                     2m           10Mi
kube-system    kube-apiserver-ortiz-control-plane  82m          319Mi
kube-system    kube-controller-manager-ortiz-control-plane  31m          42Mi
kube-system    kube-proxy-nlm69                   1m           10Mi
kube-system    kube-scheduler-ortiz-control-plane  6m           16Mi
kube-system    metrics-server-69cb89695d-t5ll5    10m          18Mi
local-path-storage  local-path-provisioner-9cd9bd544-wxr54  3m           6Mi
```

Agora já podemos fazer o deploy de uma aplicação qualquer. No nosso caso vamos fazer um deploy de um nginx. Essa aplicação que vamos fazer a configuração do HPA.

- 7) Crie um arquivo com o nome “deploy.yaml” com as informações abaixo:

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: scaling-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        resources:
          requests:
            memory: "50Mi"
            cpu: "500m"
```

- 8) Faça o deploy desse arquivo com o comando “kubectl apply -f deploy.yaml”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 k8s % kubectl apply -f deploy.yaml
deployment.apps/scaling-deploy created
```

- 9) Verifique o consumo desse POD com o comando “kubectl top pods”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 k8s % kubectl top pods
NAME                                CPU(cores)   MEMORY(bytes)
scaling-deploy-8f458dc5b-4z699      0m           6Mi
```

Veja que o consumo de CPU está bem baixo.

Agora vamos instalar o Vertical Auto Scaler, para isso:

- 10) No diretório de sua preferência, faça o clone do repositório do VPA com o comando “git clone <https://github.com/kubernetes/autoscaler.git>”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 k8s % git clone https://github.com/kubernetes/autoscaler.git
Cloning into 'autoscaler'...
remote: Enumerating objects: 153436, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 153436 (delta 0), reused 0 (delta 0), pack-reused 153435
Receiving objects: 100% (153436/153436), 163.83 MiB | 32.35 MiB/s, done.
Resolving deltas: 100% (98420/98420), done.
Updating files: 100% (28564/28564), done.
```

- 11) Entre no diretório de instalação do VPA, com o comando “cd /autoscaler/vertical-pod-autoscaler/hack”;

- 12) Execute a instalação com o comando “./vpa-up.sh”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 hack % ./vpa-up.sh
customresourcedefinition.apiextensions.k8s.io/verticalpodautoscalercheckpoints.autoscaling.k8s.io created
customresourcedefinition.apiextensions.k8s.io/verticalpodautoscalers.autoscaling.k8s.io created
clusterrole.rbac.authorization.k8s.io/system:metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:vpa-actor created
clusterrole.rbac.authorization.k8s.io/system:vpa-checkpoint-actor created
clusterrole.rbac.authorization.k8s.io/system:evictioner created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-actor created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-checkpoint-actor created
```

- 13) Agora é preciso fazer o deploy da configuração do VPA, para isso, crie um arquivo chamado “vpa.yaml” com o conteúdo abaixo:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: scaling-deploy
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: scaling-deploy
  updatePolicy:
    updateMode: "Off" #Auto
```

- 14) Faça o deploy desse arquivo com o comando “kubectl create -f vpa.yaml”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 k8s % kubectl create -f vpa.yaml
verticalpodautoscaler.autoscaling.k8s.io/scaling-deploy created
```

- 15) Para ver as informações do VPA, digite o comando “kubectl get vpa”:

```
marceloortiz@MacBook-Air-de-Marcelo-2 k8s % kubectl get vpa
NAME           MODE    CPU    MEM      PROVIDED  AGE
scaling-deploy Off     25m    262144k  True      74s
```

- 16) Para verificar a sugestão do VPA para o tamanho do POD que ele está analisando, digite o comando “kubectl describe vpa “nome_do_pod”:

```

Type:
Recommendation:
  Container Recommendations:
    Container Name:  nginx
    Lower Bound:
      Cpu:    25m
      Memory: 262144k
    Target:
      Cpu:    25m
      Memory: 262144k
    Uncapped Target:
      Cpu:    25m
      Memory: 262144k
    Upper Bound:
      Cpu:    999m
      Memory: 1045422996

```

17) A recomendação fica na seção “Target”, que no nosso exemplo está com Cpu de 25m e memória de 262144k.

18) Agora vamos estressar o nosso pod, para isso, conecte no bash do POD que foi deployado no passo 8, para isso digite o comando “kubectl -it exec nome_do_pod -- bash”:

```

marceloortiz@MacBook-Air-de-Marcelo-2 k8s % kubectl -it exec scaling-deploy-6686b77c59-s8tg9 -- bash
root@scaling-deploy-6686b77c59-s8tg9:/#

```

19) Dentro do POD, execute os comandos “apt update && apt install stress”:

```

root@scaling-deploy-6686b77c59-s8tg9:/# apt update && apt install stress
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main arm64 Packages [8069 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main arm64 Packages [167 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main arm64 Packages [2600 B]
Fetched 8447 kB in 3s (4689 kB/s)

```

20) Ainda dentro do POD, digite o comando “stress --cpu 1” e aguarde 5 minutos:

```

root@scaling-deploy-6686b77c59-s8tg9:/# stress --cpu 1
stress: info: [317] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd

```

21) Finalize o processo do stress pressionando as teclas “ctl” + “c” e saia do POD digitando o comando “exit”;

22) Agora verifique as novas recomendações do VPA, para isso digite o comando “kubectl describe vpa”:


```
Recommendation:
Container Recommendations:
Container Name:  nginx
Lower Bound:
  Cpu:          25m
  Memory:       262144k
Target:
  Cpu:          1168m
  Memory:       262144k
Uncapped Target:
  Cpu:          1168m
  Memory:       262144k
Upper Bound:
  Cpu:          61272m
  Memory:       2600215673
```

Veja que agora, ele já trás novas recomendações de configuração do POD, com o target de CPU de 1168m e de memória de 262144k.

23) Para limpar o ambiente:

- a. Remova a instalação do VPA, para isso, no diretório "autoscaler/vertical-pod-autoscaler/hack" digite o comando `./vpa-down.sh`;
- b. Remova o deploy da aplicação utilizada como exemplo, digitando o comando `kubectrl delete -f deploy.yaml`.