Cluster EKS e Escalonamento Automático de Pods (HPA) no AWS

PSI5120 - Tópicos em Computação em Nuvem
Nome: Matheus Fortunato Alves
No USP: 4484262

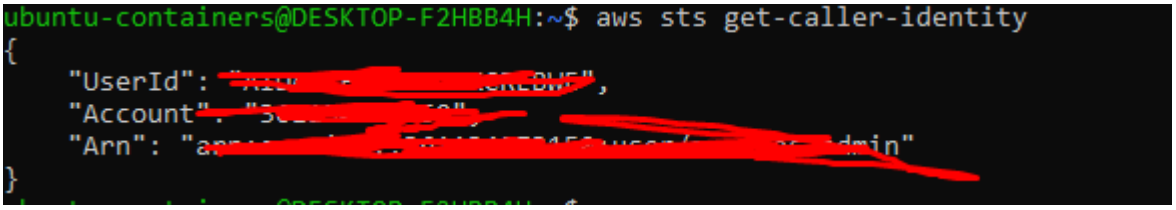
Parte 1 - Introdução e Pré Requisitos

Este laboratório tem como objetivo apresentar a criação e configuração de um cluster Kubernetes na AWS utilizando o Amazon EKS, bem como a implementação do **Horizontal Pod Autoscaler (HPA)** para escalonamento automático de pods com base em métricas de CPU.

O exercício permitirá:

- Verificar e configurar as ferramentas necessárias (AWS CLI, eksctl e kubectl);
- Criar um cluster EKS com nós gerenciados de forma simples;
- Validar a conectividade do kubectl com o cluster;
- Preparar o ambiente para testes de escalonamento automático de aplicações no Kubernetes.

Este laboratório supõe o uso de Linux (via VM ou WSL) e que o AWS CLI já está conectado a sua conta AWS via IAM



Passo 1 - Verificar Instalação das Ferramentas

Antes de iniciar, confirme que as ferramentas necessárias estão instaladas corretamente:

AWS CLI, eksctl e **kubectl**

```
aws --version
```

```
eksctl version
```

```
kubectl version --client
```

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.6.87.2-microsoft-standard-WSL2 exe/x86_64.ubuntu.20
ubuntu-containers@DESKTOP-F2HBB4H:~$ eksctl version
0.212.0
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl version --client
Client Version: v1.30.0-eks-036c24b
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu-containers@DESKTOP-F2HBB4H:~$
```

Passo 2 - Criar o Cluster EKS com Nós Gerenciados

Para a forma mais simples de cluster EKS, utilize nós gerenciados (EC2):

```
eksctl create cluster --name my-cluster --region sa-east-1 --nodes 2 --node-type t3.medium
```

O que este comando faz:

- Cria um cluster EKS chamado `my-cluster` na região `sa-east-1`.
- Cria **2 nós EC2 gerenciados**, onde os pods serão executados.
- Permite executar HPA normalmente, já que o Kubernetes pode acessar métricas de CPU/memória dos nós.

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ eksctl create cluster --name my-cluster-2 --region sa-east-1 --nodes 2 --node-type t3.medium
2025-08-16 20:40:50 [ℹ]  eksctl version 0.212.0
2025-08-16 20:40:50 [ℹ]  using region sa-east-1
2025-08-16 20:40:50 [ℹ]  setting availability zones to [sa-east-1b sa-east-1c sa-east-1a]
2025-08-16 20:40:50 [ℹ]  subnets for sa-east-1b - public:192.168.0.0/19 private:192.168.96.0/19
2025-08-16 20:40:50 [ℹ]  subnets for sa-east-1c - public:192.168.32.0/19 private:192.168.128.0/19
2025-08-16 20:40:50 [ℹ]  subnets for sa-east-1a - public:192.168.64.0/19 private:192.168.160.0/19
2025-08-16 20:40:50 [ℹ]  nodegroup "ng-a464ab1c" will use "" [AmazonLinux2023/1.32]
2025-08-16 20:40:50 [ℹ]  using Kubernetes version 1.32
2025-08-16 20:40:50 [ℹ]  creating EKS cluster "my-cluster-2" in "sa-east-1" region with managed nodes
2025-08-16 20:40:50 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-08-16 20:40:50 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=sa-east-1 --cluster=my-cluster-2'
2025-08-16 20:40:50 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-cluster-2" in "sa-east-1"
2025-08-16 20:40:50 [ℹ]  CloudWatch logging will not be enabled for cluster "my-cluster-2" in "sa-east-1"
2025-08-16 20:40:50 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=sa-east-1 --cluste
r=my-cluster-2'
2025-08-16 20:40:50 [ℹ]  default addons metrics-server, vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2025-08-16 20:40:50 [ℹ]
2 sequential tasks: { create cluster control plane "my-cluster-2",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create managed nodegroup "ng-a464ab1c",
    }
}
2025-08-16 20:40:50 [ℹ]  building cluster stack "eksctl-my-cluster-2-cluster"
2025-08-16 20:40:51 [ℹ]  deploying stack "eksctl-my-cluster-2-cluster"
```

Parte 2: Testando o Horizontal Pod Autoscaler (HPA)

Passo 1 - Deploy de uma aplicação de teste

Neste passo, vamos criar uma aplicação simples de **Apache/PHP** para testar o HPA.

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

- Este Deployment cria um Pod com **limite de CPU de 500 millicpu**.
- O servidor estará **servindo na porta 80**.

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
deployment.apps/php-apache created
service/php-apache created
```

Passo 2 - Criar o Horizontal Pod Autoscaler

Crie o HPA para a aplicação `php-apache`:

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

Explicação:

- `--cpu-percent=50` → alvo de utilização de CPU (50%).
- `--min=1` → número mínimo de Pods.
- `--max=10` → número máximo de Pods.

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
horizontalpodautoscaler.autoscaling/php-apache autoscaled
```

O HPA ajustará automaticamente a quantidade de Pods conforme a carga de CPU.

Passo 3 - Verificar o HPA

Para conferir o status do HPA:

```
kubectl get hpa
```

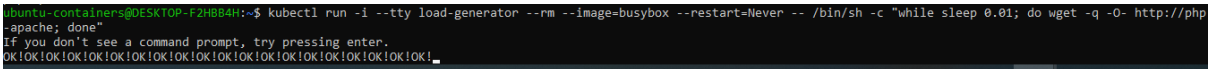
```
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa
NAME         REFERENCE              TARGETS              MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: <unknown>/50%   1         10        1          27s
```

Inicialmente, a carga de CPU é 0%, então o HPA mantém o **mínimo de 1 Pod**.

Passo 4 - Gerar carga na aplicação

Para testar o escalonamento automático, execute um **gerador de carga**:

```
kubectl run -i --tty load-generator --rm --image=busybox
--restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O-
http://php-apache; done"
```



- Este comando cria um container que faz requisições contínuas ao `php-apache`.

Passo 5 - Monitorar o HPA

Em outro terminal, acompanhe o escalonamento:

```
kubectl get hpa php-apache
```

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa php-apache
NAME         REFERENCE                  TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: 245%/50%   1         10        1          86s
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa
NAME         REFERENCE                  TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: 143%/50%   1         10        5          107s
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa --watch
NAME         REFERENCE                  TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: 143%/50%   1         10        5          114s
php-apache   Deployment/php-apache   cpu: 61%/50%    1         10        6          2m1s
php-apache   Deployment/php-apache   cpu: 57%/50%    1         10        6          2m16s
php-apache   Deployment/php-apache   cpu: 52%/50%    1         10        7          2m31s
php-apache   Deployment/php-apache   cpu: 48%/50%    1         10        7          2m46s
```

- O número de **RÉPLICAS aumenta** conforme a CPU supera o alvo (50%).
- Pode levar alguns minutos para o HPA escalar até o máximo necessário.

Passo 6 - Parar a carga e observar o scale-down

Para parar o gerador de carga, pressione **Ctrl+C** no terminal que está rodando o load-generator.

```
php-apache   ClusterIP   10.100.90.227   <none>        80/TCP    95s
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php
-apache; done"
If you don't see a command prompt, try pressing enter.
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!^Cpod "load-generator" deleted
pod default/load-generator terminated (Error)
```

Depois, monitore novamente:

```
kubectl get hpa
```

- O HPA reduzirá gradualmente o número de Pods para **o mínimo configurado (1)**.
- O tempo padrão para scale-down é de aproximadamente **5 minutos**, podendo ser ajustado.

```
ubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa --watch
NAME         REFERENCE               TARGETS        MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: 143%/50%   1         10        5          114s
php-apache   Deployment/php-apache   cpu: 61%/50%    1         10        6          2m1s
php-apache   Deployment/php-apache   cpu: 57%/50%    1         10        6          2m16s
php-apache   Deployment/php-apache   cpu: 52%/50%    1         10        7          2m31s
php-apache   Deployment/php-apache   cpu: 48%/50%    1         10        7          2m46s
php-apache   Deployment/php-apache   cpu: 37%/50%    1         10        7          3m1s
php-apache   Deployment/php-apache   cpu: 45%/50%    1         10        7          3m16s
php-apache   Deployment/php-apache   cpu: 42%/50%    1         10        7          3m31s
php-apache   Deployment/php-apache   cpu: 10%/50%    1         10        7          3m46s
php-apache   Deployment/php-apache   cpu: 1%/50%     1         10        7          4m1s
php-apache   Deployment/php-apache   cpu: 0%/50%     1         10        7          4m16s
^Cubuntu-containers@DESKTOP-F2HBB4H:~$ kubectl get hpa --watch
NAME         REFERENCE               TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   cpu: 0%/50%   1         10        7          7m4s
php-apache   Deployment/php-apache   cpu: 0%/50%   1         10        7          8m17s
php-apache   Deployment/php-apache   cpu: 0%/50%   1         10        6          8m32s
php-apache   Deployment/php-apache   cpu: 0%/50%   1         10        2          8m47s
php-apache   Deployment/php-apache   cpu: 0%/50%   1         10        1          9m2s
```

7. Limpeza dos recursos do laboratório

Após finalizar os testes, remova os recursos criados:

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