05

Faça o que eu fiz na aula

Percebemos que o portal de notícias tem efeito sanfona em relação aos usuários que o acessam: hora temos muitos usuários, hora o número é bem baixinho, principalmente no período noturno. Então, vamos economizar no número de máquinas que disponibilizamos para nosso ambiente colocando um controle automático de pods. Para isso, vamos editar o arquivo `deployment-aplicacap.yml` , que contém as características do nosso pod e vamos adicionar quantas máquina queremos ter em cada pod:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aplicacao-noticia-deployment
spec:
  selector:
    matchLabels:
      name: aplicacao-noticia-pod
  template:
    metadata:
      labels:
        name: aplicacao-noticia-pod
    spec:
      containers:
        - name: container-aplicacao-cpu
          image: jnlucas/noticia-alura:v3
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: 200m
```

Agora que temos a definição do nosso pod com uma CPU de 200MHz, precisamos atualizar esse arquivo dentro do cluster kubernetes. Para isso, vamos no terminal e digitar os seguintes comandos:

```
kubectl delete -f statefulset-sistema.yml
kubectl delete -f servico-statefulset.yml

kubectl create -f statefulset-sistema.yml
kubectl create -f servico-statefulset.yml
```

Legal, agora que o cluster está sendo monitorado, vamos adicionar os limites que queremos que o ambiente respeite. Vamos definir a quantidade máxima e mínima de máquinas e a quantidade máxima de CPU que entendemos ser saudável para o ambiente. Para isso, digite no terminal:

```
kubectl autoscale deployment aplicacao-noticia-deployment --cpu-percent=50 --min=1 --max=10
```

Pronto! Com isso ao acessar o minikube dashboard, conseguiremos perceber que além de monitorado, nosso ambiente se replica sempre que há necessidade. Para acessar o dashboard, digite no terminal:

```
minikube dashboard
```