02

Escalonando pods

Transcrição

Em nosso *cluster*, temos o objeto `Deployment`, que está abstraindo o `Pod`, dentro do qual há o container da aplicação web. Vamos tentar acessá-la usando o comando abaixo, e verificar se o `Pod` está em execução:

```
kubectl get pods
```

Observaremos que nosso `Pod` de nome `aplicacao-deployment-17598552-74tm4` está em execução no *cluster*! Precisaremos saber qual o endereço IP associado a ele para acessar seu conteúdo. Usaremos o seguinte comando:

```
kubectl describe pods
```

No entanto, não precisaremos de todas as suas informações, basta o endereço IP. Poderemos redirecionar a saída deste comando para o `grep` para que apenas isto seja filtrado. Obteremos:

```
kubectl describe pods | grep IP
IP: 172.17.0.4
```

Copiaremos este IP, abriremos um *browser* qualquer e veremos se é possível acessá-lo colando-o na barra de endereço. O que aparece é uma página de erro: "Não foi possível conectar". Ou seja, não conseguimos acessar o nosso `Pod` que está sendo rodado no *cluster* gerenciado pelo Kubernetes diretamente.

Por que isto ocorre?

Uma das dificuldades que tínhamos ao realizarmos o gerenciamento dos containers por nós mesmos era a questão de **escalonamento** da aplicação. Mas o Kubernetes, como vimos, consegue nos ajudar nesta tarefa.

Por conta do surgimento de novas sedes da Alura Sports em várias cidades do Brasil, todos os funcionários têm que acessar esta aplicação web. Deixaremos o cenário em que estávamos trabalhando, com um único objeto `Pod` dentro de `Deployment`, aumentando a aplicação para podermos atender esse aumento de acessos.

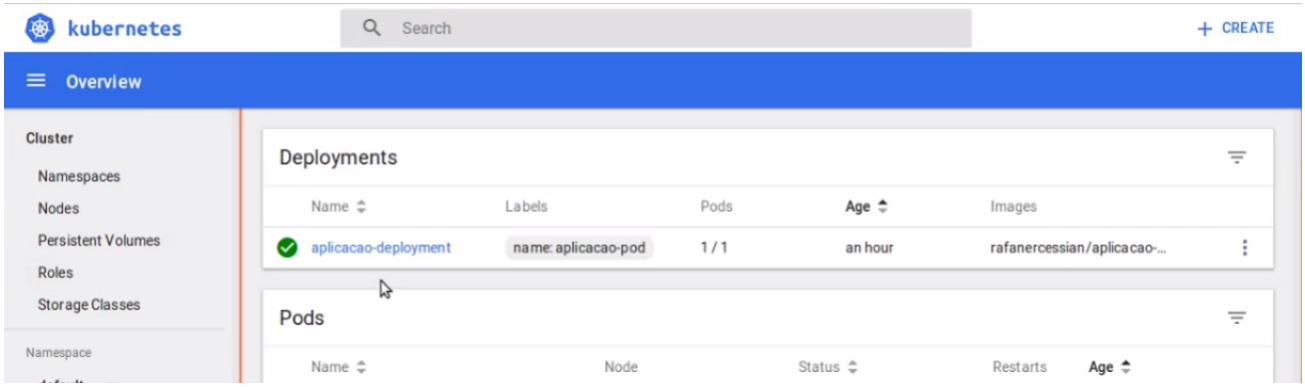
Neste momento, teremos três objetos `Pod`. Mas a pergunta, agora, é a seguinte: ao fazermos este escalonamento com todos os `Pod`, o endereço IP será o mesmo para todos eles? Vamos tentar fazer um teste e analisar o resultado obtido.

Voltaremos ao terminal e continuaremos nos comunicando com o *cluster* para o teste... Mas está ficando trabalhoso fazê-lo desta forma, sempre, não? Felizmente, é possível usarmos um painel administrativo para justamente visualizarmos o status do *cluster*.

Usaremos:

```
minikube dashboard
```

Com "Enter", somos redirecionados a uma página com um painel informando o status do *cluster*, contendo também "*Deployments*", "*Pods*", "*Replica Sets*", com esta aparência:



Nele, pode-se ver que há um `Pod` sendo abstraído pelo objeto `Deployment`.

Por conta da grande quantidade de acessos da aplicação, vamos tentar pedir ao `Deployment`, já que ele possui mais recursos, para que ele ajude neste escalonamento.

No menu localizado na lateral esquerda, iremos à aba "*Deployments*" e, do lado direito, clicaremos nos três pontinhos que indicam "*Actions*" (ou "Ações"), selecionando a opção "*Scale*" para escalonamento. Na nova janela que se abre, colocaremos `3` na quantidade de *pods* desejados, e clicaremos em "OK".

Nossa aplicação agora roda três *pods*, sendo que cada um deles possui o container da aplicação web. Clicando na aba "*Pods*" do menu à direita, há uma listagem com estes *pods* em execução. Vamos perguntar ao *cluster* sobre os endereços IP associados a eles, e verificar se são iguais.

Para isso, voltaremos ao terminal e usaremos o comando `kubectl get pods`, com o qual receberemos as informações relativas aos *pods*. E indagaremos ao *cluster* quais os endereços IP usando o comando abaixo:

```
kubectl describe pods | grep IP
```

Obteremos:

```
IP: 172.17.0.5
IP: 172.17.0.4
IP: 172.17.0.6
```

Ou seja, para cada `Pod` foi vinculado um IP diferente! Perceba, então, que os *pods* representam um mundo em constante alteração, ora podemos estar aumentando esta quantidade de *pods* para atender à demanda de acessos, ora eles podem vir a diminuir, e por aí vai.

No painel administrativo, poderíamos aumentar a quantidade de *pods* clicando em "*Deployments*" e repetindo o processo de selecionar os três pontinhos e "*Scale*", aumentando o número de `3` para `4`, por exemplo.

Deste modo nossa aplicação passa a rodar `4` *pods* em nosso *cluster*, o que nos dá mais um endereço IP. Isto se confirma se reutilizarmos o comando `kubectl describe pods | grep IP`. Isto nos retorna:

```
IP: 172.17.0.5
IP: 172.17.0.4
```

```
IP: 172.17.0.6
IP: 172.17.0.7
```

Podemos escalonar inversamente, isto é, diminuir a quantidade de *pods* indo à "*Deployments*" e selecionando "*Scale*" em "*Actions*", colocando `3` no lugar de `4`.

Então, **os objetos `Pod` são muito instáveis**, estão em constante alteração. Por conta disto, o Kubernetes não permite que acessemos um `Pod` diretamente pelo endereço IP. Precisaremos buscar uma forma de **isolar estes objetos em outro mais estabilizado**, oferecido pelo Kubernetes, com um IP fixo que não sofrerá alterações.

É o que faremos na próxima etapa! Configuraremos este objeto mais estabilizado para permitir o acesso à nossa aplicação que está sendo rodada nos nossos *pods*.