12

Kubernetes

Transcrição

No vídeo anterior, dividimos nossa aplicação em dois containers: o web e o do banco de dados, cujas implementações fizemos em nosso servidor que está em produção pela Alura Sports.

Conforme os meses se passaram, a empresa cresceu bastante, e atualmente possui sedes espalhadas por várias cidades do Brasil. Todos os funcionários devem acessar esta aplicação para cadastrar os produtos de artigos esportivos.

Para atender a esta alta demanda de acessos que a Alura Sports terá, eles saíram deste cenário com apenas um servidor no ambiente de produção para oito servidores. E agora? Será que colocamos os containers da nossa aplicação no primeiro servidor? No primeiro e no segundo? Distribuímos em todos eles...?

Será que estes servidores possuem recursos disponíveis para trabalhar com os containers da nossa aplicação? Caso haja grande demanda de acessos e necessidade de escalonamento destes containers, será que é fácil? Rápido?

Vejam quantas dúvidas começam a surgir a partir do momento em que somos responsáveis pelo gerenciamento dos containers que formam a aplicação!

Para resolver isto, pensaremos no seguinte cenário: em um jogo de futebol, há jogadores diferentes, com habilidades e características variadas. No entanto, eles jogam em conjunto para formarem o time. Alguns jogadores ficam na defesa (os zagueiros), outros ocupam o meio campo e a lateral, na transição entre defesa e ataque, e há os que ficam no ataque.

Quem é que se responsabiliza pelo gerenciamento dos jogadores, informando-os suas posições em campo? Ou no caso de um deles se machucar e ter que ser substituído... Tudo isso é tarefa do técnico, responsável por montar uma estratégia, um plano de ação, e gerenciar os jogadores.

Na nossa aplicação, existe algo parecido: há a aplicação total, do cadastramento de produtos, que dividimos em partes menores (em containers), com características distintas e habilidades específicas também. Um container, sozinho, não forma a nossa aplicação. Assim como os jogadores, eles precisam trabalhar em conjunto para isso.

No caso, quem é que gerenciará os containers informando, por exemplo, que é necessária uma determinada quantidade de containers web e outra de containers do banco de dados? O **Kubernetes**!

Voltando à nossa situação atual, em que temos um computador representando este que estamos utilizando no momento, precisaremos especificar ao Kubernetes sobre as configurações que queremos em nossa aplicação.

Fazemos estas configurações em um arquivo denominado YAML, como fizemos com o `docker-compose`. Aqui, especificaremos como desejamos que seja a nossa aplicação, que até o momento possui um container web e outro relativo ao banco de dados.

Portanto, este arquivo de `.yaml` será passado à **máquina principal do sistema**, o servidor responsável por receber este arquivo de configuração YAML. Ele delegará a implementação destes containers em outros servidores do sistema.

Por exemplo: o servidor principal decidiu que a parte do banco de dados será alocada ao servidor `A`, e a parte web para o servidor `B`. A máquina principal recebe o nome de "Mestre" (em inglês, "*Master*"), justamente por estar gerenciando os outros dois servidores.

As máquinas que de fato recebem a implementação dos containers que formam a aplicação são chamados de servidores ou máquinas "**Nós**" (em inglês, "*Node*").

O conjunto de máquinas *Master*e *Nodes* é o que forma o ***Cluster***, administrado pelo Kubernetes. A ideia é que ele faça **uma constante verificação do estado deste *cluster***.

Havíamos configurado junto ao Mestre que teríamos um container web e outro para banco de dados. Isso quer dizer que se o servidor `A` (do banco de dados) deixar de funcionar, isso afetará o estado do *cluster*, diferente da configuração original passa ao Mestre.

Deste modo, o Kubernetes tentará encontrar, de alguma forma, uma máquina *Node* disponível para realocar o container do banco de dados. É possível que ele faça isso com o servidor remanescente, `B` , para que tudo esteja da forma como o arquivo YAML havia especificado.

Agora que entendemos um pouco melhor sobre esta questão de gerenciamento dos containers que o Kubernetes é capaz de fazer, vamos trabalhar com isso em nossa máquina local.