13

Conclusão

Transcrição

Parabéns! Chegamos ao fim do curso sobre Kubernetes. Vamos relembrar o que foi visto?

Começamos com a aplicação web, cujas tarefas dividimos em containers, com `db` por exemplo, referente ao `docker-compose.yaml`, do banco de dados. Também criamos um container referente à parte web, em que fizemos a customização da imagem do projeto através do arquivo `Dockerfile`.

Feito isso, instalamos o **minikube** na nossa máquina local, em que fizemos alguns testes. No entanto, como aprendemos, o Kubernetes não possui um objeto denominado "Container", portanto tivemos que criar o objeto mais básico, o `Pod`.

Ele faz a abstração do container, porém não estabelece o estado desejado da aplicação para o gerenciamento pelo Kubernetes, sendo necessário abstraí-lo em outro com mais recursos, o `Deployment`.

Em seu arquivo de configuração YAML, definimos a abstração do objeto `Pod`, porém, como vimos, *pods* são muito instáveis, sendo possível escaloná-los, aumentando ou diminuindo sua quantidade. Por conta disso, não conseguimos acessar um `Pod` diretamente.

Para fazê-lo é preciso abstrair o acesso em uma camada, um objeto mais estável, de serviço (`Service`), que implementamos e definimos como tipo `LoadBalancer`, justamente para quando existirem grandes quantidades de acessos à aplicação, sendo balanceados nos *pods* da aplicação web.

Feitas as configurações, fomos ao banco de dados e criamos um `Pod`, colocando as informações SQL, as variáveis de ambiente, para termos `loja`, `root`, e permitirmos acesso por senhas vazias.

Vimos novamente que o `Pod` não é a melhor opção a ser utilizada de forma direta no Kubernetes. Neste momento, diferentemente da parte web, há uma preocupação com o estado do `Pod`, pois ele pode deixar de funcionar, e todas as informações cadastradas no MySQL podem ser perdidas.

Fazemos, então, a abstração deste objeto, e de outro mais semântico, o `StatefulSet`, capaz de fornecer o estado desejado da camada de aplicação para o Kubernetes gerenciar, e então realizamos os mapeamentos de volumes.

Uma vez criados os volumes, pode-se ter outros recursos no *cluster* acessando-os. Desta forma, tivemos que configurar as permissões do `Pod`, de consumo de recursos no volume recém criado.

Criamos o arquivo de permissões `permissoes.yaml` para a configuração do quanto este `Pod` do MySQL terá de permissão de acesso referente ao volume que criamos.

Por fim, colocamos os arquivos dos diretórios `app` e `db` em outro, chamado `prod`, remetendo ao ambiente de produção. No entanto, para realizar a integração com o Google Cloud, foi preciso definir `5.5` como a versão utilizada do MySQL.

Configuramos o *cluster* no Google Cloud, após o qual foi possível acessarmos a aplicação a partir de um endereço IP.

Agradeço a vocês por terem chegado até aqui, e espero encontrá-los em uma próxima oportunidade!