Kubernetes

O Kubernetes é um orquestrador de containers, ou seja, ele é utilizado para lidar com um conjunto de containers. Ele consegue gerenciar um cluster com várias máquinas e, por conta da sua arquitetura, se torna uma ferramenta muito interessante.

O Kubernetes gerencia um cluster basicamente dividido por: um master que gerencia o cluster e mantém tudo no estado desejado; e os nodes (ou nós), que costumamos chamar de workers (ou trabalhadores), que são responsáveis pela execução da aplicação.

Caso queira saber mais sobre essas ferramentas, deixo aqui indicações de leitura:

- Kubernetes: conhecendo a orquestração de containers
- Documentação Kubernetes
- Documentação Docker

Minikube

Minikube é uma das formas mais fáceis para configurar um cluster kubernetes na nossa máquina local. Ele pode ser usado em sistemas Windows ou Unix e é usado principalmente para criar um ambiente de testes para aplicações.

Para que o minikube funcione da melhor forma na nossa máquina, a documentação do minikube recomenda que tenhamos no mínimo as seguintes especificações:

- 2 CPUs ou mais
- 2 GB de memória livre
- 20 GB de espaço livre em disco
- Conexão de internet
- Gerenciador de contêiner ou máquina virtual instalado

Caso queira conhecer mais sobre essa ferramenta, acesse a documentação:

- Documentação minikube

Para que realizemos a instalação do Minikube, é importante que tenhamos um gerenciador de máquinas virtuais, no caso, o Docker, que instalamos anteriormente. Inclusive, é necessário garantir que o Docker esteja rodando antes de iniciarmos a instalação e configuração do Minikube, porque ele precisa desse gerenciador para funcionar corretamente.

Inicializaremos o Docker com o comando sudo service docker start, que pedirá nossa senha.

```
sudo service docker start
```

Com o Docker inicializado, podemos partir para a instalação no Minikube, e, assim como fizemos para instalá-lo, seguiremos os comandos da documentação do próprio Minikube. Optaremos pela versão mais atual até o momento, que é a 1.28.0. Para isso, usaremos dois comandos. São eles:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Para checar se a instalação foi realizada corretamente, iniciaremos o Minikube com o comando mikube start.

```
minikube start
```

Ao executá-lo, deve nos retornar uma mensagem semelhante à esta:

```
minikube type: Control Plane host: Running kubelet: Running apiserver: Running kubeconfig: Configured
```

Isso significa que o cluster está configurado e os demais tópicos estão sendo executados.

É interessante sabermos que, ao instalarmos e inicializarmos o Minikube, automaticamente também é instalado o Kubectl, uma ferramenta de linha de comando utilizada para controlar e configurar clusters no Kubernetes.

Para executar comandos do Kubectl dentro do minikube, sempre utilizaremos a estruturação minikube kubectl -- e o comando que desejamos executar, como, por exemplo help.

```
minikube kubectl --help
```

O comando help nos retorna outros comandos do Kubectl. Aqui, o utilizamos de maneira exemplificativa.

Com o minikube instalado, podemos acessar o dashboard do Kubernetes para visualizar, configurar e gerenciar o cluster. Para isso, executaremos o comando minikube dashboard que deve nos retornar, entre outras informações, uma URL pela qual acessaremos o dashboard.

```
minikube dashboard
```

Acessando a URL pelo navegador, estaremos na dashboard do Kubernetes, onde conseguimos gerenciar todos os serviços que estão rodando.

Na barra lateral esquerda, iremos em "Nodes", na seção de "Cluster". Note que temos um único node rodando, no caso, o minikube, então o acessaremos. Feito isso, teremos acesso a algumas informações

sobre este node, além da alocação de recursos da nossa máquina.

Agora, novamente na barra lateral, acessaremos "Pods". Perceba que não temos nenhum pods rodando, mas, posteriormente, quando instalarmos e configurarmos o Airflow no Kubernetes, teremos alguns pods em execução responsáveis pela execução dos serviços do Airflow e das tarefas dos DAGs.

Caso você não se lembre do que é um pods, há uma atividade revisando essa conceituação.

Para finalizar, vejamos como podemos parar o cluster Kubernetes quando quisermos.

No terminal, teclaremos "Ctrl + C" para encerrar o processo de execução da dashboard e, para encerrar o Minikube, executaremos minikube stop.

```
minikube stop
```

Pronto! O Minikube foi finalizado! Quando quisermos executá-lo novamente, precisaremos inicializar o Docker e executar o comando minikube start.

Helm

Assim como temos o docker hub para gerenciar imagens docker ou pypi para bibliotecas python, existe o Helm para gerenciar os pacotes para Kubernetes. Os pacotes do Helm são chamados de charts.

Utilizando o Helm é possível instalar, atualizar ou desinstalar pacotes para aplicativos Kubernetes de uma forma bem mais simples e automatizada, utilizando apenas alguns comandos. Para instalar aplicativos no Kubernetes sem a utilização do Helm, nós teríamos que definir todas as configurações manualmente utilizando um arquivo YAML bem detalhado.

Sendo assim, como principais vantagens do Helm, podemos citar:

- gerenciamento fácil e automatizado de pacotes para Kubernetes;
- oferece diferentes charts com todas as configurações prontas;
- os charts são sempre mantidos atualizados.

Para mais informações sobre o Helm, acesse a documentação:

- Documentação Helm

Os pacotes do Helm são chamados de charts e o Airflow possui um chart específico que nós podemos utilizar para realizar a configuração de um ambiente Airflow dentro do Kubernetes. Mas para que a gente consiga utilizar esse chart, teremos que instalar e configurar o Helm na nossa máquina.

Rezalizaremos essa etapa seguindo os comandos apresentados na documentação da ferramenta. Optaremos pela versão 3.10.3, que é a mais recente até o momento.

Executaremos um comando que deve baixar um script de instalação do Helm:

```
```

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

O comando a seguir serve para dar algumas permissões:

```
chmod 700 get_helm.sh
```

Por fim, utilizaremos o seguinte comando para executar o script de instalação que foi baixado:

```
./get_helm.sh
```

Para testarmos se a instalação foi realizada com sucesso, podemos pesquisar o chart do Airflow utilizando o comando a seguir:

```
helm search hub apache-airflow
```

Ele vai procurar em todos os repositórios do Helm o chart específico que estamos pesquisando, no caso, o apache-airflow.

Note que deve nos retornar algumas informações sobre este chart, como sua versão e a versão do app, que, no caso, seria o Airflow. Isso significa que o Helm foi instalado com sucesso!

Agora, adicionaremos esse chart do Airflow ao nosso repositório do Helm para que possamos utilizá-lo posteriormente. Para isso, basta executarmos o comando a seguir:

```
helm repo add apache-airflow https://airflow.apache.org
```

A execução deve nos retornar uma informação de que o Apache Airflow foi adicionado ao nosso repositório. Para conferirmos se ele foi devidamente adicionado, podemos pesquisar novamente por este chart, desta vez, diretamente no nosso repositório. Para isso, usaremos este comando:

```
helm search repo apache-airflow
```

Note que nos retornou novamente as informações referentes ao chart do Airflow, mas como pesquisamento diretamente no repositório do Helm, isso comprova que o chart foi devidamente adicionado.

Agora que já configuramos todo o cluster Kubernetes na nossa máquina e também instalamos o Helm, nosso próximo passo agora é instalar o chart do Airflow que acabamos de adicionar para que possamos construir um ambiente Airflow no nosso Kubernetes.