Chart do Airflow no Kubernetes

á configuramos todo o ambiente Kubernetes em nossa máquina, entendemos como funciona o executor Kubernetes e criamos alguns volumes persistentes. Todos esses passos são necessários para que possamos instalar o chart do Airflow. Ou seja, configurar um ambiente Airflow dentro do Kubernetes.

O chart do Airflow disponibilizado pelo Helm utiliza uma imagem padrão do Airflow no momento da instalação. Esta imagem vem com uma versão específica do Airflow e algumas outras bibliotecas para serem instaladas.

No nosso caso, utilizaremos a versão 2.4.1 do Airflow e instalaremos uma biblioteca importante para a criação do DAG. Por isso, não usaremos a imagem padrão, mas criaremos uma e faremos com que o chart a utilize no momento da instalação.

Agora, criaremos um novo arquivo na pasta airflow e o chamaremos de dockerfile. Nele, colaremos as configurações da imagem que desejamos criar:

```
FROM apache/airflow:2.10.2
COPY requirements.txt /
RUN pip install --no-cache-dir "apache-airflow==${AIRFLOW_VERSION}" -r
/requirements.txt
```

basicamente o arquivo Dockerfile aponta para a imagem do airflow 2.10.2 que queremos usar, realiza uma cópia do arquivo requirements.txt com as dependências de bibliotecas para dentro do novo contâiner e as instala por meio do pip.

De volta ao terminal, antes de executarmos o comando para construir a nossa imagem, vamos registrar o minikube como nosso comando docker, utilizando o comando eval.

```
eval $(minikube -p minikube docker-env)
```

Com este comando, estamos especificando que todas as vezes que executarmos o comando "docker" ele será executado diretamente dentro do minikube.

Agora podemos construir nossa imagem. Para isso, precisamos estar na pasta airflow:

```
cd Documents/airflow
```

omearemos nossa imagem como airflow e utilizaremos . para especificar que o dockerfile desta imagem está no diretório atual:

```
docker build -t airflow .
```

Feito isso, temos os volumes persistentes criados e uma imagem customizada com as configurações que precisamos, o que significa que estamos prontos para instalarmos o chart do Airflow.

Configurando o Chart do Airflow

Nosso próximo passo é instalar o chart que será responsável por construir um ambiente Airflow no cluster Kubernetes.

Anteriormente, adicionamos esse chart ao nosso repositório do Helm quando o instalamos. Mas vamos conferir se esse chart foi devidamente adicionado. Para isso, executaremos o seguinte comando no terminal:

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

O retorno deve nos mostrar as informações referentes a esse chart, o que confirma que ele está no repositório do Helm.

No entanto, se nós instalarmos este chart com as configurações padrões do Helm, o Airflow será inicializado com os seguintes serviços:

- Serviços principais, como Scheduler, Webserver e Banco de dados Postgres;
- Executor Celery configurado com 1 worker;
- Serviço de mensageria Redis;
- Outros serviços secundários.

No nosso caso, não é interessantes trabalharmos com essa configuração porque queremos utilizar o executor Kubernetes, e não o Celery configurado com 1 worker. Dessa forma, para que possamos alterá-las, precisaremos criar um arquivo YAML alterando algumas das configurações de acordo com a necessidade.

Nesse arquivo vão as configurações que estariam no airflow.cfg. Você o encontra nesse projeto como aula_5_2_override_values.yml.

```
executor: KubernetesExecutor
defaultAirflowRepository: airflow
defaultAirflowTag: latest

dags:
  persistence:
    enabled: true
    existingClaim: airflow-dags-claim

workers.replicas: 0
redis.enabled: false

securityContext:
  fsGroup: 0
  runAsGroup: 0
  runAsUser: 0
```

```
logs:
  persistence:
    enabled: true
    existingClaim: airflow-logs-claim


config:
  webserver:
    expose_config: 'True'
  core:
    parallelism: 16
```

O primeiro parâmetro especificado, executor, trata do executor que queremos utilizar, no caso, o KubernetesExecutor. No segundo parâmetro, defaultAirflowRepository, passamos o nome da imagem customizada que criamos, ou seja, airflow. Em defaultAirflowTag, passamos latest, para que sempre seja utilizada a última versão da nossa imagem

Na seção "dags", informaremos o volume persistente do nosso cluster, que será utilizado como nossa pasta de DAGs. Como nós já criamos esse volume, nós passamos true para o parâmetro enable. Para existingClaim, passamos o nome do pvc que está vinculado ao nosso volume claim criado para os DAGs, que é o airflow-dags-claim.

Para desabilitarmos os workers, colocamos o parâmetro workers.replicas como 0, redis.enabled igual é definido como false , para desabilitar o serviço de mensageria do Redis

Em securityContext, informamos o grupo e o usuário que serão executados nos containers. Definimos todos esses parâmetros tudo 0, assim informamos que o usuário será o root e evitamos problemas com permissões

Na categoria logs, seguimos o mesmo raciocício da configuração referente aos DAGs, então especificamos o volume persistente que será tilizado para armazenar os logs que forem gerados pelo Airflow. Em existingClaim passamos o nome do pvc que está vinculado ao pb: airflow-logs-claim.

Na seção de configuração, podemos especificar vários parâmetros que também existem no arquivo airflow.cfg. Mas, aqui, optamos por passar apenas expose_config como True, para que possamos ter acesso àquela aba de parâmetros dentro da interface do Airflow, e parallelism como 16. Caso quiséssemos passar mais parâmetros, bastaria consultar a documentação do Airflow para saber como defini-los.

Instalando o Chart do Airflow

Na execução do comando de instalação do chart especificamos algumas informações, como o nome do chart (apache-airflow/airflow), o nome do namespace onde queremos realizar a instalação (airflow) e o nome do arquivo que contém as configurações customizadas (override-values.yml):

```
helm upgrade --install airflow apache-airflow/airflow --namespace airflow -
f aula_5_2_override_values.yml
```

Ao executá-lo, a instalação será iniciada. Basta aguardarmos.

Note que nos traz várias informações no log de execução, como as credenciais para logarmos na interface do Airflow (username e password) e o comando que precisamos executar para que possamos expor a porta 8080 onde o Airflow está rodando versão do Airflow.

Antes de executarmos o comando informado, acessaremos a dashboard do Kubernetes para conferir se todos os serviços do Airflow estão rodando corretamente:

```
minikube dashboard
```

Agora acessaremos a dashboard através do link informado.

Note que estamos no namespace padrão, então mudaremos para airflow - namespace onde instalamos o Airflow. Feito isso, temos acesso à aba "Workload", que mostra todas as cargas de trabalho em execução do Kubernetes.

Na aba "Pods", encontraremos um pod sendo executado para cada um dos serviços do Airflow. Note que temos um para cada seguinte situação:

- execução do webserver;
- serviço de triggerer (novo serviço do Airflow utilizado para execução de comandos assícronos);
- serviço do statsd (para coletar algumas métricas do Airflow);
- serviço de scheduler (agendador;
- postgres (banco de dados).

Note que todos estão na cor verde, o que representa o status de "run", ou seja, em execução. Podemos, ainda, nos deparar com as cores vermelho ou laranja, mas isso ocorre apenas quando o Airflow ainda está inicializando esses serviços.

Agora que conferimos que todos os serviços estão rodando, abriremos um novo terminal (pois a dashboard trava o terminal atual) e executaremos o comando para expor a porta 8080:

```
minikube kubectl -- port-forward svc/airflow-webserver 8080:8080 --
namespace airflow
```

Feito isso, podemos abrir o navegador e acessar "localhost:8080" para acessarmos o Airflow. Para logar, usamos as credenciais fornecidas no log de execução, que, por padrão, são "admin" tanto para o username quanto para password.

Ao acessarmos a interface do Airflow, nos será recomendado a alteração de senha.

Note que aparece a versão instalada na parte inferior esquerda. No entanto, nenhum DAG aparece porque ainda não criamos um, além do mais, na hora de setar as configurações no arquivo YAML, não definimos que os DAGs de exemplo do Airflow deveriam aparecer, por isso também não são exibidos.

Agora que o Airflow está instalado e configurado com o executor kubernetes, podemos criar um DAG para analisar como as tarefas serão executadas com esse executor