Executores

O Airflow possui diferentes executores e quando novas versões dessa ferramenta surgem, novos executores também podem aparecer. Os executores são divididos em dois tipos, sendo eles os executores locais e os executores remotos.

Os executores locais como o próprio nome já diz são aqueles que executam tarefas apenas localmente, ou seja, dentro da nossa máquina. Já os executores remotos possuem a possibilidade de executar as tarefas remotamente, ou seja, utilizando mais de uma máquina.

Até o momento, na versão 2.3.2 temos os seguintes executores disponíveis:

Executores locais:

- Executor Sequencial;
- Executor Local.

Executores remotos:

- Executor Celery;
- Executor Kubernetes;
- Executor CeleryKubernetes;
- Executor LocalKubernetes;
- Executor Dask.

Caso queira conhecer mais informações sobre os executores, acesse a documentação: Executor.

LocalExecutor

No Docker, por default, o executor configurado no arquivo docker-compose.yaml é o CeleryExecutor com os recursos necessários para execução, para mudar é preciso alterar a tag AIRFLOW__CORE__EXECUTOR para LocalExecutor.

Caso não esteja usando o docker segue o passo a passo:

Para começar, temos que acessar o arquivo de configuração do Airflow que fica dentro da pasta de instalação "airflow". Abra a pasta em questão no VS Code e, na barra lateral esquerda, onde constam os arquivos internos a ela, procure por "airflow.cfg". Dê dois cliques para abri-lo e perceba que possui diversas variáveis de configuração.

Pressione "Ctrl + F" para abrir a barra de pesquisa do Airflow e busque por "executor". Essa pesquisa irá destacar a ocorrência do termo pesquisado no corpo deste arquivo. Perceba que, entre as ocorrências, há uma variável com este nome definida como "SequentialExecutor". Podemos apagar essa definição e substituir por "LocalExecutor" para que passemos a utilizar o executor local. Tecle "Ctrl + S" para salvar essa alteração.

Alteramos o executor do Airflow, mas não é suficiente para que possamos utilizá-lo porque, por padrão, essa ferramenta utiliza o banco de dados SQLite, que só permite uma conexão por vez. Como queremos

que nossas tarefas sejam executadas em paralelo, precisamos de um banco que suporte mais de uma conexão simultaneamente, então utilizaremos o Postgres, sugerido pela própria documentação do Airflow.

```
sudo apt install postgresql postgresql-contrib
```

Após a execução do comando anterior, tecle "y" para confirmar e aguarde a conclusão do carregamento.

Com o Postgres instalado, vamos acessar o usuário para conseguir criar o banco de dados a ser utilizado. Para isso, utilizamos o comando sudo -i -u postgres.

```
sudo -i -u postgres
```

Dentro do usuário, criaremos um banco de dados chamado "airflow_db" utilizando o código createdb airflow_db.

```
createdb airflow_db
```

Agora podemos acessá-lo com psql airflow_db.

```
psql airflow_db
```

Em seguida, criaremos um usuário e senha que o Airflow utilizará para acessar este banco. O comando para esta criação é:

```
CREATE USER airflow_user WITH PASSWORD 'airflow_pass'
```

Criamos um usuário chamado "airflow_user" e uma senha "airflow_pass". Agora, precisamos garantir que esse usuário tenha acesso a todos os privilégios necessários para alterar o banco de dados. Para isso, utilizaremos GRANT ALL PRIVILEGES ON DATABASE airflow_db TO airflow_user, que concede todos os privilégios de "airflow_db" ao usuário que criamos, "airflow_user". Isso permitirá que o usuário adicione e exclua dados do banco, além de realizar conexões.

```
GRANT ALL PRIVILEGES ON DATABASE airflow_db TO airflow_user
```

Agora que criamos e configuramos nosso banco, podemos usar o comando exit, duas vezes, para sair do banco de dados e do usuário.

Hora de conectar esse banco ao Airflow! No arquivo de configuração no VS Code, pesquisaremos por "sql_alchemy", que nos indicará a variável "sql_alchemy_conn". Perceba que ela armazena uma url, porque o

Airflow utiliza o framework SQLAlchemy para fazer a conexão com o banco de dados.

Para conectar ao banco de dados que queremos, substituiremos a url por outra, estruturada a partir do que nos é informado na documentação do Airflow. A estrutura consiste, basicamente, em colocar o usuário "airflow_user", a senha "airflow_pass" e o nome do banco de dados "airflow_db". Ficará assim:

```
sql_alchemy_conn =
postgresql+psycopg2://airflow_user:airflow_pass@localhost/airflow_db
```

Com a variável configurada, nos resta inicializar o banco de dados no Airflow. Para isso, retorne ao terminal da máquina, acesse a pasta "airflow" com o comando cd Documents/airflow, ative o ambiente (source venv/bin/activate) e exporte a variável de ambiente Airflow_Home com export AIRFLOW_HOME=~/Documents/airflow. Por fim, inicialize o banco de dados no Airflow com airflow db init.

```
cd Documents/airflowalura
source venv/bin/activate
AIRFLOW_HOME=~/Documents/airflowalura
airflow db init
```

De volta ao terminal, execute o comando airflow standalone para que possamos acessar a interface do Airflow.

```
airflow standalone
```

Após a execução, acesse "localhost8080" no navegador e entre com usuário e senha.

```
Você pode encontrar esta senha no arquivo "standalone_admin_password.txt",
dentro da pasta "airflowalura".
```

Na interface do Airflow, vamos procurar por "get_stocks_dag" e abri-lo. Em seguida, podemos ativá-lo no símbolo superior esquerdo. Ative, também, o "Auto-refresh" para que recarregue automaticamente.

Perceba que "Total runnig" varia de valores, mas em certo momento corresponde ao número 4, o que significa que há 4 DAG runs em execução. No VS Code, dentro de "airflowalura", podemos observar que a pasta "stocks" foi criada com as subpastas correspondentes às ações, onde constam os arquivos csv. As tarefas estão sendo executadas!

Além da configuração do executor, existem vários outros ajustes importantes para que possamos trabalhar com produção de tarefas em paralelo. Veremos algumas dessas configurações nos próximos vídeos!

Paralelismo

Agora conheceremos dois outros parâmetros muito importantes quando trabalhamos com tarefas executadas em paralelo. Para encontrá-los, abra o arquivo "airflow.cfg", dentro de "airflowalura", no VS Code.

Aproximadamente na linha 38, encontraremos o primeiro parâmetro, "max_active_tasks_per_dag", definido como "16". Ele é utilizado para definir a quantidade máxima de tarefas que podem ser executadas ou agendadas simultaneamente em um DAG.

Por volta da linha 46, há o segundo parâmetro, chamado "max_active_runs_per_dag", também definido como "16". Ele estabelece a quantidade máxima de DAG runs que podem ser executados simultaneamente em um mesmo DAG. DAG run é basicamente a instância de execução do DAG no tempo.

Vamos redefinir estes parâmetros. "max_active_tasks_per_dag" passará a valer "2", e "max_active_runs_per_dag", "4". Tecle "Ctrl + S" para salvar essa alteração e vamos para a execução do Airflow no terminal da máquina.

No terminal, acesse a pasta "airflowalura", ative o ambiente virtual, exporte a variável de ambiente Airflow Home e execute o Airflow.

Após a execução anterior, acesse "localhost8080" no navegador e abra "get_stocks_dag". Em seguida, exclua o histórico de execução desse DAG clicando no símbolo da lixeira, no canto superior direito, retorne para a interface do Airflow e recarregue a página até que esse DAG reapareça na listagem.

Quando o DAG reaparecer na interface principal, abra-o e ative-o. Não esqueça de ativar, também, o "Auto-refresh". Note que na primeira aparição da figura que se assemelha a um gráfico, havia 4 colunas, o que corresponde a 4 DAG runs. Perceba que são executados, no máximo, 4 DAG runs e 2 tarefas por vez, conforme definimos nos parâmetros.

Outra forma de ter acesso a essas observações é clicando em "Details", na parte superior desta aba. Em "Details" vamos nos atentar a dois parâmetros: "Max Active Runs" e "Concurrency". O primeiro equivale a "max_active_runs_per_dag", e o segundo, a "max_active_tasks_per_dag", que definimos no arquivo de configuração do Airflow como 4 e 2, respectivamente.

Note que, em "Max Active Runs", consta o dado "3/4" informando que estão sendo executados 3 DAG runs e que o número máximo de DAG a serem executados simultaneamente seria 4.

DAG Runs

Um DAG Run é uma execução propriamente dita de um DAG no tempo, incluindo horários, tempos de execução de cada uma das tasks, status de cada uma delas e outras informações. Ou seja, ele representa basicamente uma instância de um DAG no tempo.

Quando um DAG é executado, um DAG Run é criado e todas as tasks desse DAG são executadas. Um detalhe é que cada DAG Run é executado de forma separada dos demais, isso significa que você pode executar o mesmo DAG diversas vezes ao mesmo tempo, ou executar vários DAGs diferentes ao mesmo tempo.

Se quiser entender um pouco mais sobre os DAG Runs, consulte a documentação:

- DAG Run