TaskFlow e Decorators

O TaskFlow é uma forma mais simples e prática de criar DAGs usando código Python.

A API TaskFlow foi criada a partir da versão 2.0 do Airflow e oferece uma maneira de declarar explicitamente mensagens passadas entre tarefas em um DAG. Além disso, ela também declara automaticamente as dependências entre as tarefas. Dessa forma, podemos citar duas grandes vantagens de utilizar TaskFlow:

- Comunicação entre as tarefas de forma mais simplificada:

As tarefas, por padrão, são totalmente isoladas uma das outras e não podem compartilhar dados. Para conseguir compartilhar informações entre tarefas sempre precisamos utilizar o mecanismo XCom.

Utilizando o TaskFlow, o Airflow consegue declarar de forma automática as dependências entre as tarefas a partir da chamada de uma função Python. Assim não precisamos nos preocupar com a utilização dos xcoms.

- Criação de DAGs com mais facilidade:

O TaskFlow simplifica como um DAG e suas tarefas são declarados. Ele faz isso encapsulando grande parte do código que é utilizado para a criação das tarefas, em decoradores. Utilizando esses decoradores o código fica mais simples e mais fácil de entender.

Decorators

Um decorator serve basicamente para adicionar a uma determinada função algumas funcionalidades que ele possui implementadas por baixo dos panos. Então, ele consegue adicionar essas funcionalidades sem ter que mudar o código da função em si.

O decorador de tarefas (@task) permite que os usuários convertam qualquer função Python em uma instância de tarefa utilizando o PythonOperator por baixo dos panos. Já o decorador DAG (@dag) permite aos usuários instanciar o DAG sem utilizar um gerenciador de contexto.

Caso queira aprofundar seus conhecimentos a respeito dessa API e entender como ela funciona por baixo dos panos, deixo aqui dois links bem interessantes da documentação do Airflow:

- TaskFlow; e
- Tutorial sobre a API TaskFlow.

Bibliotecas diversas

Durante a criação do nosso DAG foram importadas diferentes bibliotecas para que pudéssemos desenvolver todo o código das nossas funções. Vamos conhecer melhor um pouco de cada uma dessas bibliotecas:

- **yfinance**: essa biblioteca do python extrai dados da API do Yahoo Finance e nos permite acessar esses dados de forma bem prática utilizando módulos e métodos.

- **airflow.decorators**: esse módulo disponibiliza os decorators do Airflow. Nós importamos os decorators @task e @dag por meio do código from airflow.decorators import DAG, task. Esses decorators são importados para que possamos criar nossa task e nosso DAG utilizando taskflow.
- **airflow.macros**: macros são uma maneira de passar dados dinâmicos para os DAGs em tempo de execução. Esse módulo possui métodos que podemos utilizar para trabalhar com esses macros, como a função ds_add que nós utilizamos na nossa task para acessar o dia anterior à data de execução no Airflow.
- **pathlib**: esse módulo é utilizado para trabalhar com caminhos (path) do sistema. Nós importamos a classe Path desse módulo, que é utilizada para instanciar um caminho concreto de acordo com o que for passado como argumento e possui métodos como o mkdir que utilizamos para criação de novas pastas.
- **pendulum**: essa biblioteca facilita o nosso trabalho com datas e horas. Então ela basicamente é utilizada para facilitar a manipulação dos tipos datetime no python.

Você também pode acessar as documentações dessas bibliotecas para explorar mais cada uma delas:

- yfinance;
- airflow.decorators;
- airflow.macros;
- pathlib;
- pendulum.