

Relatório 11 - Pipelines de Dados I - Airflow

Guilherme Loan Schneider

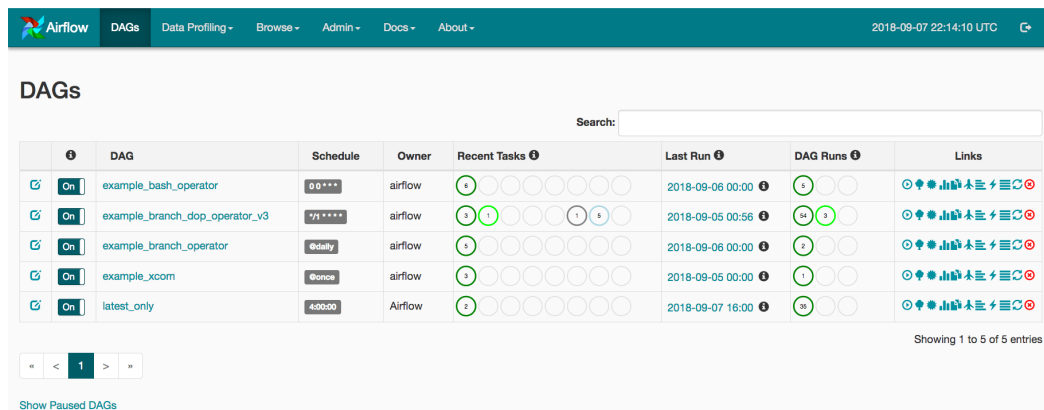
Descrição da atividade

O curso do card atual tem como objetivo dar os passos iniciais na utilização do Apache Airflow, bem como exemplificar a utilização de DAGs (Directed Acyclic Graph) e elucidar várias questões importantes para o bom entendimento.

1.1 The basics of Apache Airflow

Essa seção faz uma pequena introdução sobre o Airflow, mostrando o que é, como funciona, além de fazer a instalação inicial com o Docker Desktop e utilizando o Terminal.

Além disso, explica também os principais pontos da interface do Airflow.



The screenshot shows the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Data Profiling, Browse, Admin, Docs, and About. The main header displays the date and time: 2018-09-07 22:14:10 UTC. Below the header, the 'DAGs' section is visible, featuring a search bar and a table of DAGs. The table has columns for DAG, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links. Five DAGs are listed: example_bash_operator, example_branch_dop_operator_v3, example_branch_operator, example_xcom, and latest_only. Each row shows the DAG's status (On), its schedule (e.g., @daily, @once), the owner (airflow), recent task status (represented by colored circles), the last run time, the number of DAG runs, and a set of icons for various actions like refresh, zoom, and delete. A pagination bar at the bottom indicates 'Showing 1 to 5 of 5 entries'.

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
On	example_bash_operator	@daily	airflow	5	2018-09-06 00:00	5	Icons for DAG actions
On	example_branch_dop_operator_v3	@daily	airflow	5	2018-09-05 00:56	5	Icons for DAG actions
On	example_branch_operator	@daily	airflow	5	2018-09-06 00:00	5	Icons for DAG actions
On	example_xcom	@once	airflow	5	2018-09-05 00:00	1	Icons for DAG actions
On	latest_only	@once	Airflow	5	2018-09-07 16:00	1	Icons for DAG actions

1.2 The Forex Data Pipeline

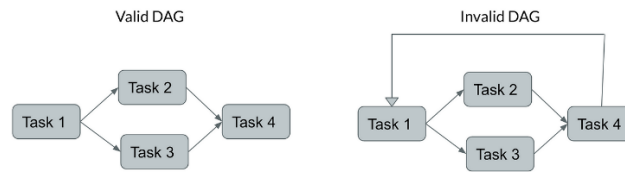
Esse modulo traz o primeiro conjunto de DAGs que serão implementadas, chamado de Forex Data Pipeline, onde em cada aula é abordado uma DAG individual.

1.2.1 O que é uma DAG?

Uma DAG é um conjunto de tasks que deverão ser feitas, em uma determinada ordem (chamado de dependências) e não deverão possuir loops. Uma única task (representada na figura abaixo) simboliza uma função que deverá ser feita, onde a Task 1 deverá verificar se é possível acessar um determinado site, a Task 2 acessa informações, e a Task 4 baixa essas informações e salva. A Task 3 poderia ser ao invés de acessar todas as informações novamente, apenas atualizar as que já existem.

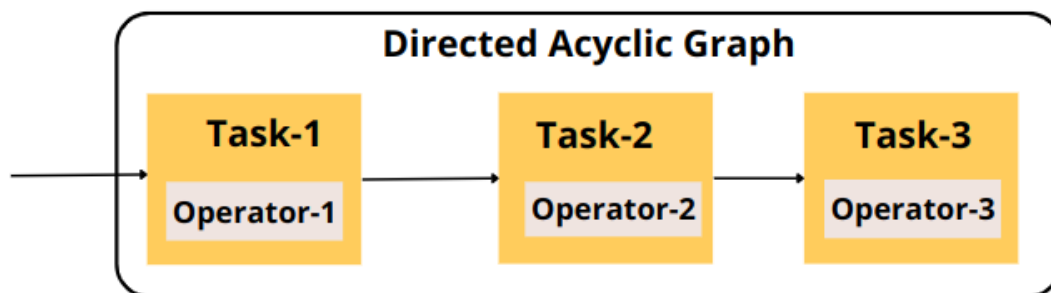
É importante destacar também que as DAGs possuem campos default na hora de cria-las, como o dag_id que é o nome da sua DAG, description que pode ser utilizado para descrever como funciona a sua DAG, start_date que define a data que sua DAG deve começar, schedule_interval que indicará a frequência de execução (a cada hora,

dia, semana), `default_args` que são os argumentos padrão de operadores e por fim o `catchup`, que indicará se deverá ser executado operações “atrasadas”.



1.2.2 O que são operadores?

Um operador é a parte lógica de como os dados serão processados em cada Task de uma DAG. Abaixo é mostrado um exemplo implementado em aula de um operador. Cada operador representa uma unidade de trabalho, como rodar um script Python, executar uma consulta SQL ou chamar uma API.



```
with DAG(dag_id='start_and_schedule_dag', schedule_interval=timedelta(hours=1), default_args=default_args) as dag:

    # Task 1
    dummy_task_1 = DummyOperator(task_id='dummy_task_1')

    # Task 2
    dummy_task_2 = DummyOperator(task_id='dummy_task_2')

    dummy_task_1 >> dummy_task_2
```

1.2.3 Utilizando Operadores e Sensores

Ao nos depararmos com a utilização dos operadores, é importante também compreendermos o que podemos acessar/executar/fazer com esse tipo de funcionalidade. No curso utilizamos tipos diferentes, tanto de operadores, quanto de sensores.

Os sensores (Sensors) são utilizados para quando deve-se esperar por um evento acontecer, como verificar se um arquivo específico esteja disponível em uma determinada pasta (FileSensor). Abaixo estão os sensors utilizados:

- HttpSensor - Verifica se uma URL ou API está respondendo antes de continuar o fluxo.
- FileSensor - Aguarda até que um arquivo específico esteja disponível em um diretório antes de prosseguir.

Em seguida, estão os principais operadores utilizados:

- a) PythonOperator - Executa uma função Python dentro do fluxo de trabalho.
- b) BashOperator - Roda comandos de terminal (bash), como scripts shell ou comandos do sistema.
- c) HiveOperator - Executa consultas SQL no Apache Hive (usado para armazenar e consultar grandes volumes de dados).
- d) SparkSubmitOperator - Envia um job para ser executado no Apache Spark, útil para processar grandes volumes de dados.
- e) EmailOperator - Envia emails como parte do fluxo de trabalho, útil para notificações ou relatórios.
- f) SlackAPIPostOperator - Envia mensagens para canais do Slack, útil para alertas ou acompanhamento de tarefas.

1.3 Mastering your DAGs

Essa seção tem como objetivo mostrar o funcionamento das DAGs, juntamente com parâmetros que podemos alterar nelas.

1.3.1 DagRun, Backfill e Catchup

- a) DagRun – Representa uma instância de uma DAG, contendo as suas tasks a serem executadas;
- b) Catchup – Esse processo consiste na execução de tasks anteriores que não foram executadas por algum motivo na DAG. (Exemplo: Uma dag foi criada no dia 01/03/2025 com `schedule_date` definido para diário, e ela foi ativada apenas no dia 14/03/2025, o parâmetro Catchup quando em True faz com que ela seja executada desde o dia 01/03, até o dia 14/03).
- c) Backfill – Processo de executar um dag ou tarefa específica em um dag nos últimos dias. Por exemplo, se um dag estiver em execução desde o início de um mês e uma nova tarefa tiver sido adicionada a ele, e essa tarefa recém-adicionada precisar ser executada nos últimos dias, preenchendo as DAGs executadas anteriormente.

1.3.2 Pontos importantes ao utilizar datetime

- a) Sempre especificar o tempo e o fuso horário, isso fará com que o python receba um objeto “aware” (esse tipo de objeto leva em conta que o fuso horário está definido), evitando o tipo “naive”, que não leva em consideração o fuso horário;
- b) Ao criar um datetime sem o fuso horário definido, não significa que ele estará no UTC;
- c) Uma solução é importar o `timezone` do Airflow para criar esses objetos aware.

1.3.3 Tornando as tasks dependentes

- a) `depends_on_past`
 - a. Definido a nível da task;
 - b. Se a instancia anterior falhou, a próxima não é executada;
 - c. Consequentemente, a task atual não tem status;
 - d. A primeira instância com `start_date` poderá ser executada.
- b) `wait_for_downstream`
 - a. Definido a nível da task;

- b. Uma instancia de uma task X irá esperar pela finalização de todas as tasks anteriores a ela terminarem no estado “successfull” antes de executar.

1.3.4 Como estruturar os diretórios DAG

Deve-se haver essa preocupação na organização dos arquivos por conta de existir muitas DAGs, tornando difícil seu manuseio, e DAGs utilizando muitos arquivos externos.

Utilizando arquivos .zip:

1. Deve-se criar um arquivo zip com todas as DAGs e seus arquivos extras;
2. As DAGs devem estar no root do arquivo zip;
3. Em seguida, o Airflow irá escanear e carregar os arquivos DAGs.

Utilizando o DagBag:

Uma DagBag é uma coleção de DAGs, analisados de uma árvore de pastas e tem configurações de alto nível. (é como um "coletor" de todas as DAGs registradas no Airflow).

1. A dagbag torna mais fácil de utilizar diferentes pastas de desenvolvimento (dev/staging/prod);
2. Um sistema consegue rodar diferentes sets independentes de configuração;
3. Permite adicionar novas pastas a partir de um script na pasta padrão das DAGs.

1.3.5 Erros nas DAGs

A nível da DAG:

- dagrun_timeout
- sla_miss_callback
- on_failure_callback
- on_sucess_callback

A nível das tasks:

- email
- email_on_failure
- email_on_retry
- retries
- retry_delay
- retry_exponential_backoff
- max_retry_delay
- execution_timeout
- on_failure_callback
- on_sucess_callback
- on_retry_callback

Conclusões

A partir das aulas, foi possível compreender os conceitos básicos até a implementação e gerenciamento avançado de DAGs. Além disso, a estrutura e funcionamento das DAGs, a utilização de operadores e sensores, além de práticas essenciais para o agendamento e monitoramento eficiente dos fluxos de trabalho. Por fim, destaca-se boas práticas no uso do datetime, organização de diretórios e tratamento de erros.

Referencias

[Apache Airflow: The Hands-On Guide \(Seção 1 à 4\)](#)