Relatório 11 - Pipelines de Dados - Airflow (I)

Yuri Vacelh Zamulhak Zdebski

Descrição da atividade

Este relatório apresenta uma análise do treinamento em Apache Airflow, abrangendo tanto os aspectos teóricos quanto práticos do curso. O objetivo principal foi desenvolver habilidades através do estudo das seções selecionadas, da realização dos exercícios e de códigos autorais.

The Basics of Apache Airflow

Why and what is Airflow

Sendo bem direto, é uma substituição do software cron (software que permite agendar tarefas para serem executadas). Podendo orquestrar um pipeline de dados de de forma dinâmica, se utilizando de Python como linguagem o que deixa mais acessível. A definição dada na documentação é "*Apache Airflow is a way to programmatically author, schedule, and monitor data pipelines*".

Seus principais componentes são:

- Web Server: Servidor que roda a parte grafica
- Scheduler: Daemon responsavel por agendar tarefas
- Metadata database: Armazena os metadados relacionados a usuarios e tarefas
- Executor: Define como as tarefas são executadas
- Worker: Processo que executa as tarefas, definido pelo Executor

Além dos componentes, existem 5 conceitos importantes quanto ao Apache Airflow, são eles:

- DAG: Grafo representando o data pipeline
- Operator: Descreve uma tarefa dentro do data pipeline
- Task: Uma instancia de um operator
- TaskInstance: Representa uma execução especifica de uma task
- Worflow: Combinação de tudo isso

Installing Airflow

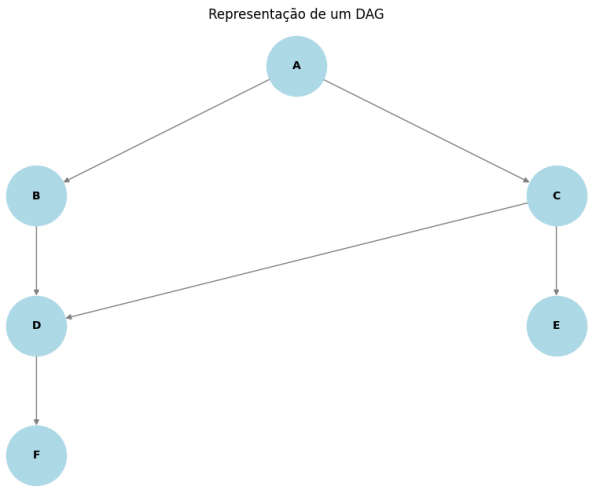
Tive algumas dificuldades na instalação do software por conta de algumas bibliotecas legado que estavam no Dockerfile, instalando as dependências mais atuais resolveu o problema.

Forex Data Pipeline

What is a DAG

Um DAG (Directed Acyclic Graph) é um tipo de grafo orientado em que os vértices são conectados por arestas direcionadas, e não há ciclos, ou seja, não é possível retornar a um vértice partindo dele e seguindo as arestas.

Figura 1: Exemplo de DAG



Fonte: Autoria própria.

No contexto do Airflow, um DAG representa uma coleção de tasks (e suas dependências) a serem agendados. Um nó seria a task em si e uma aresta uma dependência entre N tasks.

What is an Operator

No Apache Airflow, operators são os blocos fundamentais que definem as tarefas dentro de um fluxo de trabalho (DAG). Cada operator representa uma unidade de execução, como executar uma função Python (PythonOperator), rodar uma query em banco de dados (PostgresOperator) ou interagir com serviços externos, como o Google Cloud ou AWS. Operators são altamente configuráveis e podem ser combinados para criar fluxos de trabalho complexos e automatizados. Eles são divididos em 3 classes, os Action Operators, que executam ações, Transfer Operators, que movem dados de um sistema para outro e Sensor Operators, que esperam que algo aconteça.

Mastering your DAGs

Parâmetros importantes

Os dois parâmetros mais importantes para configurar corretamente em um DAG no Apache Airflow são o start_date e o schedule_interval. O start_date define a data e o horário a partir dos quais o DAG começará a ser agendado, enquanto o schedule_interval especifica o intervalo de tempo entre as execuções do DAG.

No Apache Airflow, o execution_date representa o momento lógico em que uma instância de DAG é executada, correspondendo ao período de agendamento definido pelo schedule_interval. Ele não indica o momento real em que o DAG está sendo executado, mas sim o intervalo de tempo para o qual o DAG foi agendado. Por exemplo, em um DAG com agendamento diário, se o start_date for 2024-11-01 e o schedule_interval for 1 day, a execução para 2024-11-02 terá um execution_date de 2024-11-01. Esse comportamento ocorre porque o Airflow executa o DAG considerando os dados do período anterior, permitindo pipelines baseados em dados passados.

Backfill and Catchup

DagRun é um objeto que é instanciado pelo scheduler toda vez que um DAG é acionado. Dentro desse objeto existem logs para quando execuções são perdidas (por exemplo a execução é parada). Quando catchup está ativo, essas Tasks perdidas são agendadas novamente.

Timezones

Timezones são regiões no mundo que estão na mesma zona temporal, ou seja, os dias e noites acontecem ao mesmo tempo.

Em Python, objetos do tipo datetime.datetime podem ter seu atributo tzinfo como aware ou naive. O Airflow suporta timezones, por padrão eles são setados como utc.

How make tasks dependent

Quando configuramos uma task com o parâmetro depends_on_past, isso significa que ela dependerá da execução bem-sucedida de instâncias anteriores de si mesma. Se ocorrer uma falha em uma execução passada, as execuções futuras da task ficarão bloqueadas até que a falha seja corrigida. Por outro lado, o parâmetro wait_for_downstream atua no nível do DagRun, fazendo com que o DAG espere que todas as tasks downstream em execuções anteriores sejam totalmente concluídas antes de permitir a execução de futuras instâncias do DAG.

Web Server

O web server do Apache Airflow é responsável por fornecer a interface gráfica (UI) que permite aos usuários visualizar e interagir com os DAGs e suas execuções. Ele é baseado em Flask e exibe informações como o status das tasks, logs de execução, gráficos de dependências e configurações do DAG. Além disso, o web server permite agendar, pausar ou acionar manualmente execuções de DAGs, além de oferecer ferramentas para monitoramento e troubleshooting. O servidor pode ser executado de forma independente ou integrado com o executor de Airflow, como o Celery ou Kubernetes.

Failures in Dags

É possível monitorar erros em um pipeline tanto no nível de DAG quanto de task (como timeouts, falhas em callbacks, etc.). No nível de task, além de monitorar esses erros, é possível configurar funções personalizadas para lidar com falhas, além de definir alertas por e-mail para notificar sobre problemas durante a execução.

Testing Dags

A biblioteca utilizada foi a Pytest, amplamente reconhecida por sua simplicidade, flexibilidade e capacidade de gerar relatórios claros, o que a torna ideal para processos de testes em aplicações complexas, como pipelines no Apache Airflow. Os testes foram organizados em cinco categorias principais:

- Testes de validação: Verifica a consistência das configurações do DAG, como dependências e parâmetros.
- Testes de definição: Avalia a estrutura do DAG, garantindo dependências bem configuradas e ausência de ciclos.
- Testes unitários: Testam tasks individualmente, assegurando que funcionem corretamente em isolamento.
- Testes de integração: Validam a interação entre tasks, garantindo que dependências sejam respeitadas.
- Testes de pipeline de ponta a ponta (end-to-end): Simulam a execução completa do pipeline, verificando o funcionamento geral do DAG.