Relatório- Card- 11-Prática - Pipeline de dados - airflow (I)

Jefferson korte junior

Pipelines de Dados e Apache Airflow

1. Pipelines de Dados

Um **pipeline de dados** é uma série de etapas pelas quais os dados passam, desde a sua origem até o destino. O objetivo principal é **movimentar dados brutos** de uma fonte para um sistema de armazenamento ou análise, transformando-os em informações úteis.

Etapas Comuns (ETL)

- **Extração (Extraction):** Os dados são retirados das fontes originais, como bancos de dados, APIs e arquivos CSV.
- **Transformação (Transformation):** Os dados são limpos, filtrados e convertidos para um formato adequado para análise.
- **Carregamento (Loading):** Os dados processados são armazenados em um **data warehouse** ou banco de dados.

Benefícios: Automação de tarefas repetitivas

2. Apache Airflow

O **Apache Airflow** é uma plataforma de código aberto utilizada para **criação, agendamento e monitoramento** de workflows (fluxos de trabalho). Ele é essencial para **gerenciar pipelines de dados** de maneira eficiente e automatizada.

Principais Características

- **Orquestração de Tarefas:** Define e organiza tarefas em um pipeline.
- **Agendamento:** Permite a execução de tarefas em horários específicos.
- **Monitoração:** Ferramentas para acompanhar o status e o progresso das tarefas.
- **Escalabilidade:** Gerencia workflows complexos e grandes volumes de dados.

3. Principais Componentes do Airflow

DAG (Directed Acyclic Graph)

- Um **gráfico acíclico direcionado**, onde cada nó é uma tarefa e as arestas representam dependências.
- Define a **sequência** e a **relação** entre tarefas dentro do pipeline.

Operadores (Operators)

Determina o que é feito, representando uma tarefa singular no processo da pipeline. Operadores são independentes e podem executar simultaneamente. O mesmo resultado deve ser produzido em reexecuções. Uma tarefa é criada instanciando a classe de um operador, que define a natureza da tarefa e sua execução. Ao instanciar, a tarefa vira um nó na DAG.

Tipos comuns:

PythonOperator → Executa funções Python

BashOperator → Executa comandos Bash

PostgresOperator → Executa SQL no PostgreSQL

S3FileTransferOperator → Transfere arquivos para S3

TaskInstance

1 - Representa a execução de uma **tarefa específica** dentro de um DAG em um determinado momento.

2 - Contém informações sobre o **status** da tarefa (sucesso, falha, etc.).

Workflow

1 - O **fluxo completo** de tarefas interconectadas dentro de um DAG.

2 - Define a **sequência e a dependência** das tarefas no pipeline.

Instalação do Airflow Usando Docker

A instalação pode ser realizada utilizando um **contêiner Docker** baseado na imagem `python:3.5-slim`. As etapas incluem:

1 - Criar e iniciar um container Docker

2 - Definir a variável de ambiente `AIRFLOW_HOME,`

3 - Atualizar pacotes e instalar dependências,

4 - Criar usuário `airflow` e configurar ambiente virtual

5 -Instalar `apache-airflow` versão `1.10.10`,

6 -Inicializar o banco de dados (`airflow initdb`)

7- Iniciar `scheduler` e `webserver`

Comando para construir uma imagem Docker personalizada:

```
Unset
docker build -t airflow-basic .
```

Comandos CLI do Airflow

Aqui estão alguns comandos úteis para **gerenciar o ambiente** e **executar DAGs**:

Gerenciar Contêineres

```
Unset
docker ps                     # Exibir contêineres ativos
docker exec -it container_id /bin/bash  # Acessar um contêiner
```

Inicialização e Configuração

```
Unset
airflow initdb                # Inicializar metadatabase
airflow resetdb               # Resetar metadatabase
airflow upgradedb             # Atualizar metadatabase
```

Execução e Monitoramento

```
Unset
airflow webserver             # Iniciar Web Server
airflow scheduler             # Iniciar Scheduler
airflow list_dags             # Listar DAGs disponíveis
```

```
airflow trigger_dag example_python_operator  # Acionar um DAG
manualmente
```

Depuração e Testes

```
Unset
airflow list_tasks example_python_operator  # Listar tarefas
de um DAG
airflow test example_python_operator print_the_context
2018-05-07  # Testar uma tarefa específica sem dependências
```

Backfill e Catchup

Backfill: Execução retroativa de DAGs para garantir que todas as execuções pendentes sejam realizadas.

Catchup: Configuração que permite ao Airflow executar DAGs desde o `start_date` até o presente.

Bom para corrigir falhas e recuperar dados históricos.

Lidando com Timezones no Airflow

O Airflow usa **pendulum** para **gerenciar fusos horários**, garantindo que os DAGs rodem no horário correto.

Configuração:

No arquivo `airflow.cfg`

Diretamente no DAG usando pendulum

Como o Web Server do Airflow Funciona

O **Web Server** do Airflow é uma interface gráfica baseada em **Flask** para visualizar, controlar e monitorar DAGs.

1 - Exibe **logs** e **status de execução**

2 - Permite **interações com workflows**

3 - Facilita administração sem precisar usar comandos CLI

Lidando com Falhas em DAGs

O Airflow permite configurar **retries** e **alertas** para falhas:

Retries:

python

```
Unset
retries=3  # Número de tentativas
retry_delay=timedelta(minutes=5)  # Tempo entre tentativas
```

Alertas:

- Configuração via `EmailOperator` para **notificações** em caso de falha.

Comparação entre os dois cursos

> No **primeiro curso**, há uma abordagem mais **manual**, com a criação de ambiente virtual e instalação do Airflow sem um gerenciador específico.
>
> No curso da **Alura**, o fluxo é mais estruturado, com a instalação e configuração via **VirtualBox e Ubuntu**, além de um foco maior no uso do **Apache Airflow** para orquestrar pipelines de dados.

Conclusão:

Nesse card foi apresentado Pipelines de Dados - Airflow. O Apache Airflow é uma ferramenta para orquestrar workflows, permitindo a criação e o gerenciamento de fluxos de tarefas de forma dinâmica e escalável. Usando DAGs, ele organiza as tarefas em pipelines, garantindo dependências e execução eficiente. Este material explica como estruturar DAGs, lidar com falhas, criar dependências entre tarefas e testar pipelines.

Referências:

Apache Airflow: The Hands-On Guide

Curso Alura - Orquestrando seu primeiro pipeline de dados

Curso de Docker - Youtube - **Introdução ao Docker para iniciantes | Docker Tutorial #docker**