Laboratório Prático: Crie um DAG para Apache Airflow com PythonOperator

Tempo estimado necessário: **40** minutos

Introdução

Neste laboratório, você explorará a interface do usuário (UI) da web do Apache Airflow. Em seguida, você criará um Grafo Acíclico Direto (DAG) usando PythonOperator e, finalmente, o executará através da UI da web do Airflow.

Objetivos

Após concluir este laboratório, você será capaz de:

- Explorar a UI da web do Airflow
- Criar um DAG com PythonOperator
- Submeter um DAG e executá-lo através da UI da web

Pré-requisitos

Por favor, certifique-se de que você completou a leitura sobre os **Operadores DAG do Airflow** antes de prosseguir com este laboratório. Você deve estar familiarizado com operações de entrada e saída (I/O) em Python e pacotes de requisição para concluir este laboratório.

Sobre o Skills Network Cloud IDE

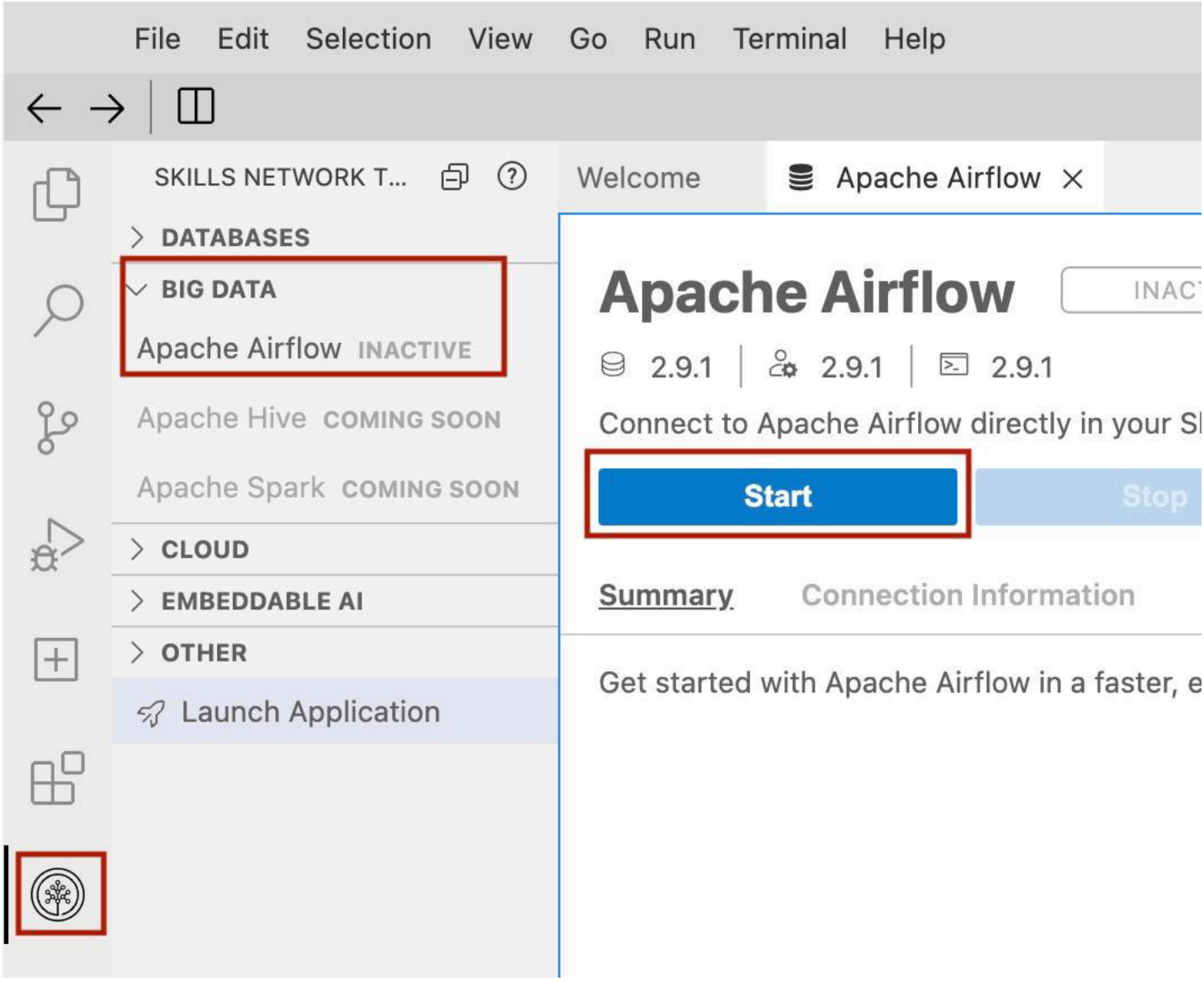
O Skills Network Cloud IDE (baseado no Theia e Docker) fornece um ambiente para laboratórios práticos para cursos e laboratórios relacionados a projetos. O Theia é um IDE (Ambiente de Desenvolvimento Integrado) de código aberto que pode ser executado em um desktop ou na nuvem. Para completar este laboratório, você usará o Cloud IDE baseado no Theia, executando em um contêiner Docker.

Aviso importante sobre este ambiente de laboratório

Por favor, esteja ciente de que as sessões para este ambiente de laboratório não são persistentes. Um novo ambiente é criado para você toda vez que você se conecta a este laboratório. Quaisquer dados que você possa ter salvo em uma sessão anterior serão perdidos. Para evitar a perda de seus dados, planeje completar estes laboratórios em uma única sessão.

Exercício 1: Iniciar o Apache Airflow

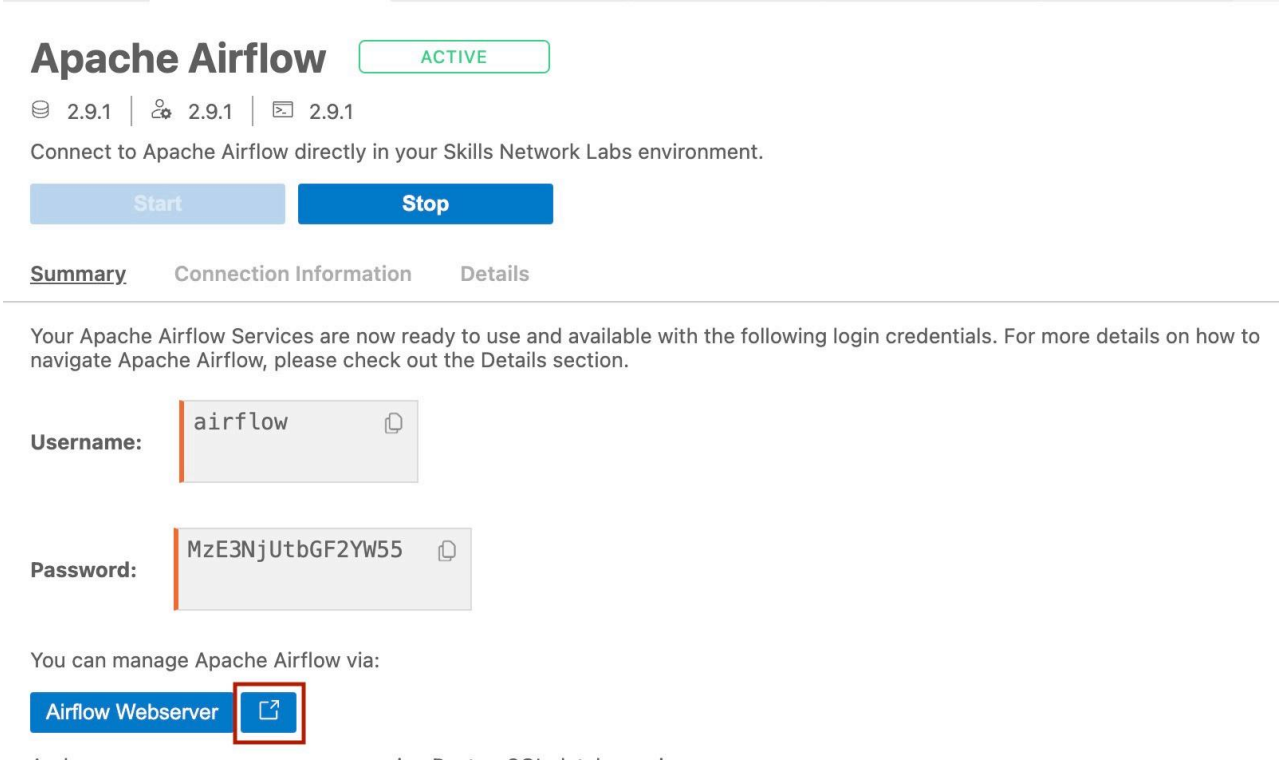
1. Clique em **Skills Network Toolbox**.
2. Na seção **BIG DATA**, clique em **Apache Airflow**.
3. Clique em **Iniciar** para iniciar o Apache Airflow.



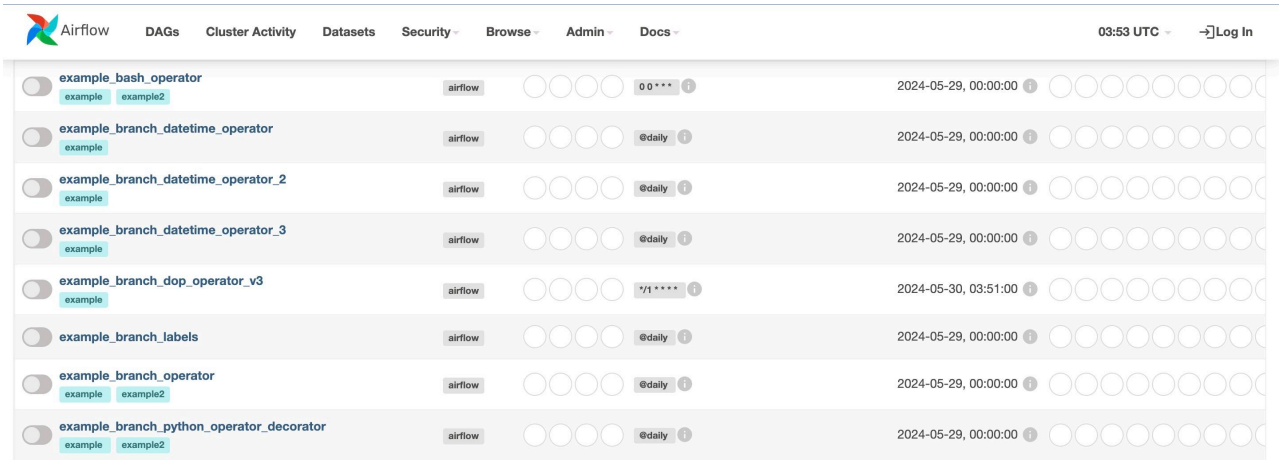
Nota: Por favor, tenha paciência, levará alguns minutos para o Airflow iniciar. Se houver um erro ao iniciar o Airflow, por favor, reinicie-o.

Exercício 2: Abra a Interface Web do Airflow

1. Quando o Airflow iniciar com sucesso, você deve ver uma saída semelhante à abaixo. Assim que **Apache Airflow** tiver iniciado, clique no ícone destacado para abrir a **Interface Web do Apache Airflow** em uma nova janela.



Você deve acessar uma página que se parece com esta.



Exercício 3: Criar um DAG com PythonOperator

Em seguida, você criará um DAG, que definirá um pipeline de tarefas, como `extract`, `transform`, `load` e `check` com PythonOperator.

1. Crie um arquivo `DAG`, `my_first_dag.py`, que será executado diariamente. Para criar um novo arquivo, escolha `File->New File` e nomeie-o como `my_first_dag.py`. O arquivo `my_first_dag.py` define as tarefas `execute_extract`, `execute_transform`, `execute_load` e `execute_check` para chamar as respectivas funções Python.

```
# Import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow.models import DAG
# Operators; you need this to write tasks!
from airflow.operators.python import PythonOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
# Define the path for the input and output files
input_file = '/etc/passwd'
extracted_file = 'extracted-data.txt'
transformed_file = 'transformed.txt'
output_file = 'data_for_analytics.csv'
def extract():
    global input_file
    print("Inside Extract")
    # Read the contents of the file into a string
    with open(input_file, 'r') as infile, \
            open(extracted_file, 'w') as outfile:
        for line in infile:
            fields = line.split(':')
            if len(fields) >= 6:
                field_1 = fields[0]
```

```
                field_3 = fields[2]
                field_6 = fields[5]
                outfile.write(field_1 + ":" + field_3 + ":" + field_6 + "\n")
def transform():
    global extracted_file, transformed_file
    print("Inside Transform")
    with open(extracted_file, 'r') as infile, \
            open(transformed_file, 'w') as outfile:
        for line in infile:
            processed_line = line.replace(':', ',')
            outfile.write(processed_line + '\n')
def load():
    global transformed_file, output_file
    print("Inside Load")
    # Save the array to a CSV file
    with open(transformed_file, 'r') as infile, \
            open(output_file, 'w') as outfile:
        for line in infile:
            outfile.write(line + '\n')
def check():
    global output_file
    print("Inside Check")
    # Save the array to a CSV file
    with open(output_file, 'r') as infile:
        for line in infile:
            print(line)
# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'Your name',
    'start_date': days_ago(0),
    'email': ['your email'],
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
# Define the DAG
dag = DAG(
    'my-first-python-etl-dag',
    default_args=default_args,
    description='My first DAG',
    schedule_interval=timedelta(days=1),
)
# Define the task named execute_extract to call the `extract` function
execute_extract = PythonOperator(
    task_id='extract',
    python_callable=extract,
    dag=dag,
)
# Define the task named execute_transform to call the `transform` function
execute_transform = PythonOperator(
    task_id='transform',
    python_callable=transform,
    dag=dag,
)
# Define the task named execute_load to call the `load` function
execute_load = PythonOperator(
    task_id='load',
    python_callable=load,
    dag=dag,
)
# Define the task named execute_load to call the `load` function
execute_check = PythonOperator(
    task_id='check',
    python_callable=check,
    dag=dag,
)
# Task pipeline
execute_extract >> execute_transform >> execute_load >> execute_check
```

Exercício 4: Enviar um DAG

Enviar um DAG é tão simples quanto copiar o arquivo Python do DAG para a pasta `dags` no diretório `AIRFLOW_HOME`.

1. Abra um terminal e execute o comando abaixo para definir o `AIRFLOW_HOME`.

```
export AIRFLOW_HOME=/home/project/airflow
echo $AIRFLOW_HOME
```

2. Execute o comando abaixo para enviar o DAG que foi criado no exercício anterior.

```
cp my_first_dag.py $AIRFLOW_HOME/dags
```

3. Verifique se o seu DAG realmente foi enviado.

4. Execute o comando abaixo para listar todos os DAGs existentes.

```
airflow dags list
```

5. Verifique se `my-first-python-etl-dag` faz parte da saída.

```
airflow dags list|grep "my-first-python-etl-dag"
```

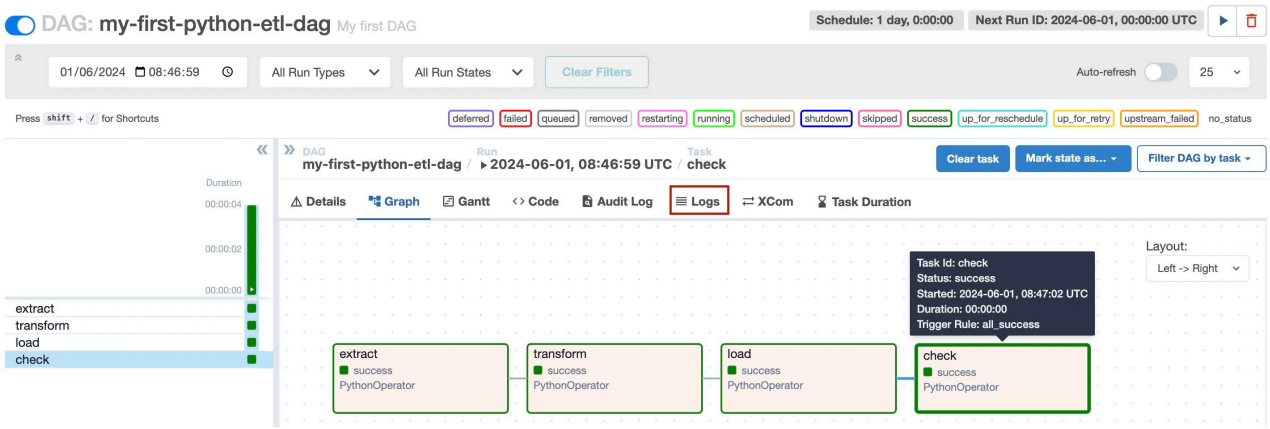
6. Você deve ver o nome do seu DAG na saída.

7. Execute o comando abaixo para listar todas as tarefas em `my-first-python-etl-dag`.

```
airflow tasks list my-first-python-etl-dag
```

8. Você deve ver todas as quatro tarefas na saída.

9. Você pode executar a tarefa pela interface web. Você pode verificar os logs das tarefas clicando na tarefa individual na visualização em gráfico.



Exercício Prático

Escreva um DAG chamado `ETL_Server_Access_Log_Processing` que irá extrair um arquivo de um servidor remoto e, em seguida, transformar o conteúdo e carregá-lo em um arquivo.

A URL do arquivo é a seguinte:
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Apache%20Airflow/Build%20a%20DAG%20using%20Airflow/web-server-access-log.txt

O arquivo de log de acesso do servidor contém os seguintes campos.

a. `timestamp` - TIMESTAMP
b. `latitude` - float
c. `longitude` - float
d. `visitorid` - char(37)
e. `accessed_from_mobile` - boolean
f. `browser_code` - int

Tarefas

1. Adicione tarefas no arquivo DAG para baixar o arquivo, ler o arquivo e extrair os campos `timestamp` e `visitorid` do `web-server-access-log.txt`.
2. Coloque o `visitorid` em maiúsculas para todos os registros e armazene-o em uma variável local.
3. Carregue os dados em um novo arquivo `capitalized.txt`.
4. Crie o bloco de importações.
5. Crie o bloco de Argumentos do DAG. Você pode usar as configurações padrão.
6. Crie o bloco de definição do DAG. O DAG deve ser executado diariamente.
7. Crie as tarefas de extração, transformação e carga para chamar o script Python.
8. Crie o bloco de pipeline de tarefas.
9. Submeta o DAG.
10. Verifique se o DAG foi submetido.

▶ Clique aqui para **dica**.
▶ Clique aqui para a **solução**.

Autores

Lavanya T S

Outros Colaboradores

Rav Ahuja