Laboratório Prático: Crie um DAG para Apache Airflow com BashOperator

Tempo estimado: **40** minutos

Introdução

Neste laboratório, você criará fluxos de trabalho usando BashOperator em DAGs do Airflow e simulará um processo ETL usando comandos bash que estão agendados para rodar uma vez por dia.

Objetivos

Após concluir este laboratório, você será capaz de:

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

Pré-requisitos

Por favor, certifique-se de que você completou a leitura sobre os Operadores de DAG do Airflow antes de prosseguir com este laboratório. É altamente recomendável que você esteja familiarizado com comandos `bash` para realizar este laboratório.

Sobre o Skills Network Cloud IDE

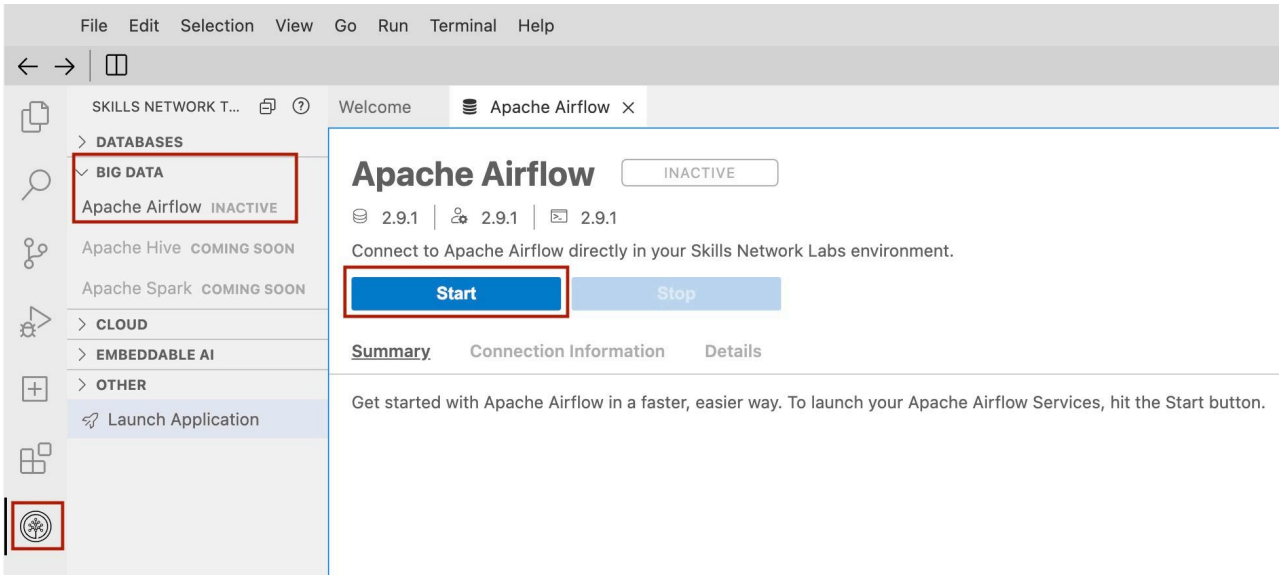
O Skills Network Cloud IDE (baseado no Theia e Docker) fornece um ambiente para laboratórios práticos relacionados a cursos e projetos. O Theia é um IDE (Ambiente de Desenvolvimento Integrado) de código aberto que pode ser executado em um desktop ou na nuvem. Para completar este laboratório, você usará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

Exercício 1: Iniciar Apache Airflow

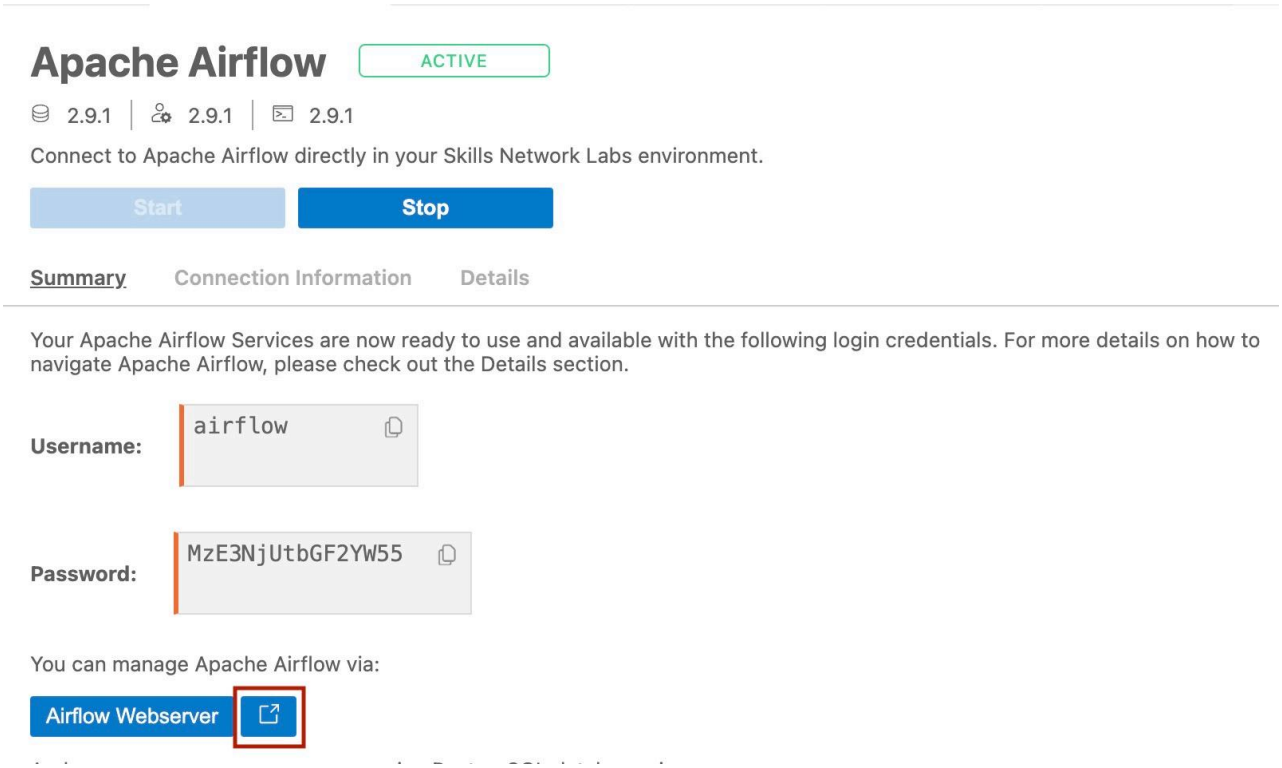
1. Clique em **Skills Network Toolbox**.
2. Na seção **BIG DATA**, clique em **Apache Airflow**.
3. Clique em **Start** para iniciar o Apache Airflow.



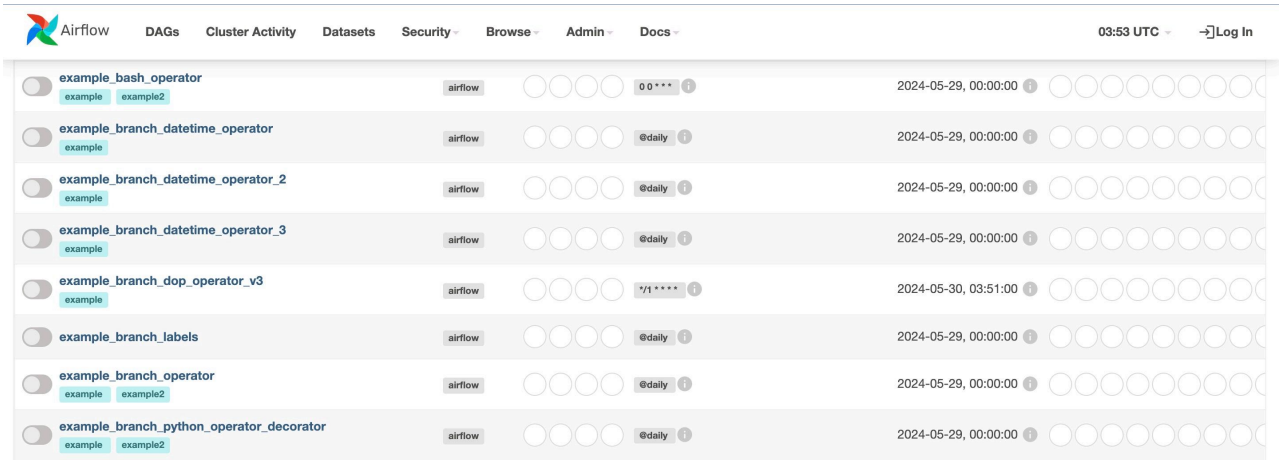
Nota: Por favor, seja paciente, levará alguns minutos para o Airflow iniciar.

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 chegar a uma página que se parece com esta.



Exercício 3: Criar um DAG

Vamos criar um DAG que roda diariamente, extrai informações de usuários do arquivo */etc/passwd*, transforma-as e as carrega em um arquivo.

Este DAG terá duas tarefas `extract` que extrai campos do arquivo `/etc/passwd` e `transform_and_load` que transforma e carrega os dados em um arquivo.

```
# 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.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
#defining DAG arguments
# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'your_name_here',
    'start_date': days_ago(0),
    'email': ['your_email_here'],
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
# defining the DAG
# define the DAG
dag = DAG(
    'my-first-dag',
    default_args=default_args,
```

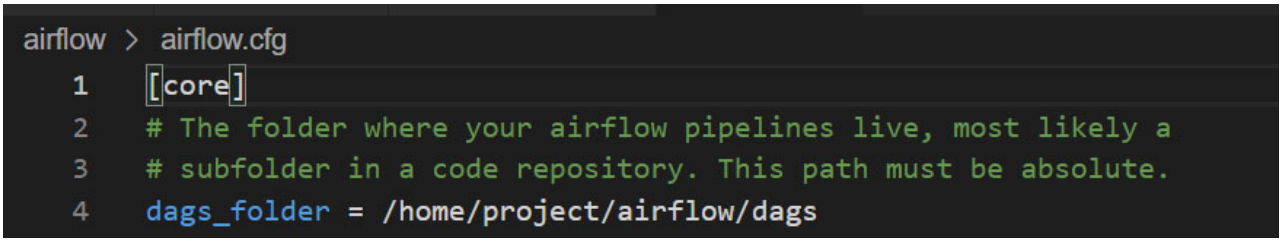
```
    description='My first DAG',
    schedule_interval=timedelta(days=1),
)
# define the tasks
# define the first task
extract = BashOperator(
    task_id='extract',
    bash_command='cut -d":" -f1,3,6 /etc/passwd > /home/project/airflow/dags/extracted-data.txt',
    dag=dag,
)
# define the second task
transform_and_load = BashOperator(
    task_id='transform',
    bash_command='tr ":" "," < /home/project/airflow/dags/extracted-data.txt > /home/project/airflow/dags/transformed-data.csv',
    dag=dag,
)
# task pipeline
extract >> transform_and_load
```

1. Crie um novo arquivo escolhendo Arquivo->Novo Arquivo e nomeando-o `my_first_dag.py`.
2. Em seguida, copie o código acima e cole-o em `my_first_dag.py`.

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`.

O Airflow procura arquivos de origem Python dentro da pasta `DAGS_FOLDER` especificada. A localização da `DAGS_FOLDER` pode ser encontrada no arquivo airflow.cfg, onde foi configurada como `/home/project/airflow/dags`.



O Airflow carregará os arquivos de origem Python deste local designado. Ele processará cada arquivo, executará seu conteúdo e, em seguida, carregará quaisquer objetos DAG presentes no arquivo.

Portanto, ao enviar um `DAG`, é essencial posicioná-lo dentro desta estrutura de diretório. Alternativamente, o diretório `AIRFLOW_HOME`, que representa a estrutura `/home/project/airflow`, também pode ser utilizado para o envio do DAG.

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.

```
export AIRFLOW_HOME=/home/project/airflow
 cp my_first_dag.py $AIRFLOW_HOME/dags
```

3. Verifique se o seu DAG foi realmente enviado.
4. Execute o comando abaixo para listar todos os DAGs existentes.

```
airflow dags list
```

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

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

Você deve ver o nome do seu DAG na saída.

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

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

Você deve ver 2 tarefas na saída.

Exercício de prática

Escreva um DAG chamado `ETL_Server_Access_Log_Processing.py`.

1. Crie o bloco de imports.
2. Crie o bloco de Argumentos do DAG. Você pode usar as configurações padrão.
3. Crie o bloco de definição do DAG. O DAG deve ser executado diariamente.
4. Crie a tarefa de download. A tarefa de download deve baixar o arquivo de log de acesso do servidor, que está disponível na URL:

```
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
```

5. Crie a tarefa de extração.

   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

   A tarefa `extract` deve extrair os campos `timestamp` e `visitorid`.

6. Crie a tarefa de transformação. A tarefa `transform` deve capitalizar o `visitorid`.
7. Crie a tarefa de carga. A tarefa `load` deve comprimir os dados extraídos e transformados.
8. Crie o bloco de pipeline de tarefas. O bloco de pipeline deve agendar as tarefas na ordem listada abaixo:

   1. download
   2. extract
   3. transform
   4. load

9. Envie o DAG.
10. Verifique se o DAG foi enviado.

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

```
# 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.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
#defining DAG arguments
# 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),
}
# defining the DAG
# define the DAG
dag = DAG(
    'ETL_Server_Access_Log_Processing',
    default_args=default_args,
    description='My first DAG',
    schedule_interval=timedelta(days=1),
)
# define the tasks
# define the task 'download'
download = BashOperator(
    task_id='download',
    bash_command='curl "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Apache%20Airflow/Buil
    dag=dag,
)
# define the task 'extract'
extract = BashOperator(
    task_id='extract',
    bash_command='cut -f1,4 -d"#" web-server-access-log.txt > /home/project/airflow/dags/extracted.txt',
    dag=dag,
)
# define the task 'transform'
transform = BashOperator(
    task_id='transform',
    bash_command='tr "[a-z]" "[A-Z]" < /home/project/airflow/dags/extracted.txt > /home/project/airflow/dags/capitalized.txt',
    dag=dag,
)
# define the task 'load'
load = BashOperator(
    task_id='load',
    bash_command='zip log.zip capitalized.txt' ,
    dag=dag,
)
# task pipeline
download >> extract >> transform >> load
```

Envie o DAG executando o seguinte comando.

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

Verifique se o DAG foi enviado na interface da Web ou na CLI usando o comando abaixo.

```
airflow dags list
```

Autores

Lavanya T S
Ramesh Sannareddy

© IBM Corporation. Todos os direitos reservados.