Travaux pratiques : créer un DAG pour Apache Airflow avec BashOperator

Temps estimé nécessaire : **40** minutes

Introduction

Dans ce laboratoire, vous allez créer des workflows à l'aide de BashOperator dans les DAG Airflow et simuler un processus ETL à l'aide de commandes bash programmées pour s'exécuter une fois par jour.

Objectifs

Après avoir terminé ce laboratoire, vous serez capable de :

- Découvrez l'interface Web d'Airflow
- Créer un DAG avec BashOperator
- Soumettez un DAG et exécutez-le via l'interface utilisateur Web

Prérequis

Assurez-vous d'avoir terminé la lecture des opérateurs DAG Airflow avant de poursuivre ce laboratoire. Il est fortement recommandé de bien connaître `bash`les commandes pour effectuer ce laboratoire.

À propos de Skills Network Cloud IDE

L'IDE Cloud de Skills Network (basé sur Theia et Docker) fournit un environnement pour les travaux pratiques liés aux cours et aux projets. Theia est un IDE (environnement de développement intégré) open source qui peut être exécuté sur un ordinateur de bureau ou sur le cloud. Pour réaliser ce laboratoire, vous utiliserez l'IDE Cloud basé sur Theia, exécuté dans un conteneur Docker.

Avis important concernant cet environnement de laboratoire

Veuillez noter que les sessions de cet environnement de laboratoire ne sont pas permanentes. Un nouvel environnement est créé pour vous à chaque fois que vous vous connectez à ce laboratoire. Toutes les données que vous avez pu enregistrer lors d'une session précédente seront perdues. Pour éviter de perdre vos données, prévoyez de terminer ces laboratoires en une seule session.

Exercice 1 : démarrer Apache Airflow

1. Cliquez sur **Boîte à outils du réseau de compétences** .
2. Dans la section **BIG DATA** , cliquez sur **Apache Airflow** .
3. Cliquez sur **Démarrer** pour démarrer Apache Airflow.

Remarque : soyez patient, il faudra quelques minutes pour qu'Airflow démarre.

Exercice 2 : ouvrir l'interface Web d'Airflow

1. Une fois qu'Airflow démarre correctement, vous devriez voir un résultat similaire à celui ci-dessous. Une fois **Apache Airflow** démarré, cliquez sur l'icône en surbrillance pour ouvrir **l'interface utilisateur Web d'Apache Airflow** dans la nouvelle fenêtre.

Vous devriez atterrir sur une page qui ressemble à ceci.

Exercice 3 : Créer un DAG

Créons un DAG qui s'exécute quotidiennement et extrait les informations utilisateur du fichier */etc/passwd* , les transforme et les charge dans un fichier.

Ce DAG aura deux tâches `extract`qui extraient les champs du `/etc/passwd`fichier et `transform_and_load`qui transforment et chargent les données dans un fichier.

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
```

```
# 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
```

Copié!

1. Créez un nouveau fichier en choisissant Fichier->Nouveau fichier et en le nommant `my_first_dag.py`.
2. Ensuite, copiez le code ci-dessus et collez-le dans `my_first_dag.py`.

Exercice 4 : Soumettre un DAG

Soumettre un DAG est aussi simple que de copier le fichier Python DAG dans le `dags`dossier du `AIRFLOW_HOME`répertoire.

Airflow recherche les fichiers sources Python dans le fichier `DAGS_FOLDER`. L'emplacement de `DAGS_FOLDER`peut être situé dans le fichier airflow.cfg, où il a été configuré comme `/home/project/airflow/dags`.

Airflow chargera les fichiers sources Python à partir de cet emplacement désigné. Il traitera chaque fichier, exécutera son contenu, puis chargera tous les objets DAG présents dans le fichier.

Par conséquent, lors de la soumission d'un `DAG`, il est essentiel de le positionner dans cette structure de répertoire. Alternativement, le `AIRFLOW_HOME`répertoire, représentant la structure `/home/project/airflow`, peut également être utilisé pour la soumission DAG.

1. Ouvrez un terminal et exécutez la commande ci-dessous pour définir le `AIRFLOW_HOME`.

```
1. 1
2. 2
1. export AIRFLOW_HOME=/home/project/airflow
2. echo $AIRFLOW_HOME
```
Copié!

2. Exécutez la commande ci-dessous pour soumettre le DAG qui a été créé dans l'exercice précédent.

```
1. 1
2. 2
1. export AIRFLOW_HOME=/home/project/airflow
2.  cp my_first_dag.py $AIRFLOW_HOME/dags
```
Copié!

3. Vérifiez que votre DAG a bien été soumis.

4. Exécutez la commande ci-dessous pour répertorier tous les DAG existants.

```
1. 1
1. airflow dags list
```
Copié!

5. Vérifiez que cela `my-first-dag`fait partie de la sortie.

```
1. 1
1. airflow dags list|grep "my-first-dag"
```
Copié!

   Vous devriez voir le nom de votre DAG dans la sortie.

6. Exécutez la commande ci-dessous pour répertorier toutes les tâches dans `my-first-dag`.

```
1. 1
1. airflow tasks list my-first-dag
```
Copié!

   Vous devriez voir 2 tâches dans la sortie.

Exercice pratique

Écrivez un DAG nommé `ETL_Server_Access_Log_Processing.py`.

1. Créez le bloc d'importations.
2. Créez le bloc Arguments DAG. Vous pouvez utiliser les paramètres par défaut
3. Créez le bloc de définition du DAG. Le DAG doit s'exécuter quotidiennement.
4. Créez la tâche de téléchargement. La tâche de téléchargement doit télécharger le fichier journal d'accès au serveur, qui est disponible à l'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. Créez la tâche d'extraction.

   Le fichier journal d'accès au serveur contient ces champs.

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

   La `extract`tâche doit extraire les champs `timestamp`et `visitorid`.

6. Créez la tâche de transformation. La `transform`tâche doit mettre en majuscule le `visitorid`.

7. Créez la tâche de chargement. La `load`tâche doit compresser les données extraites et transformées.

8. Créez le bloc de pipeline de tâches. Le bloc de pipeline doit planifier la tâche dans l'ordre indiqué ci-dessous :

   1. télécharger
   2. extrait
   3. transformer
   4. charger

9. Soumettre le DAG.
10. Vérifiez si le DAG est soumis.

▼ Cliquez ici pour **un indice** .

Suivez l'exemple de code Python donné dans le laboratoire et apportez les modifications nécessaires pour créer le nouveau DAG.

▼ Cliquez ici pour la **solution** .

Ajoutez au fichier les parties de code suivantes pour `ETL_Server_Access_Log_Processing.py`compléter les tâches indiquées dans le problème.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

```
# 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%20
    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
```

Copié!

Soumettez le DAG en exécutant la commande suivante.

```
1
```

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

Copié!

Vérifiez si le DAG est soumis sur l'interface utilisateur Web ou la CLI à l'aide de la commande ci-dessous.

```
1
```

```
airflow dags list
```

Copié!

Auteurs

Lavanya TS
Ramesh Sannareddy