CLOUD BI

7. forduló

DXC TECHNOLOGY

A kategória támogatója: DXC Technology

Ismertető a feladathoz

Ebben a fordulóban Airflow témájú kérdésekkel találkozhatsz.

NEM lesz szükséged Google Cloud accountra, vagy azon történő munkára a feladatok megoldásához.

Egyes válaszlehetőségeknél "Option1", "Option2" stb. megjelöléssel találkozhatsz, ez szövegileg sosem része az adott válasznak, csupán a válaszok későbbi összekapcsolódását biztosítja a magyarázatokkal.

Felhasznált idő: 00:00/38:00 **Elért pontszám:** 0/10

1. feladat 0/1 pont

Az alábbiak közül melyik állítás **igaz** a DAG koncepciójára nézve?

Válasz

- ○ Option1: **Feladatok olyan jellegű gyűjteménye amelyben a feladatok mindegyike ugyanazon a szerveren fut.**
- ◉ Option2: **Feladatok olyan jellegű gyűjteménye amelyben a feladatok egymástól való függősége irányított gráfként reprezentálható amelyben nem megengedett az irányított kör.**
  Ez a válasz helyes, de nem jelölted meg.
- ○ Option3: **Feladatok olyan jellegű gyűjteménye ahol a feladatok egymás állapotától függetlenül külön-külön is ütemezhetőek.**
- ○ Option4: **Feladatok olyan jellegű gyűjteménye, amelyben minden feladat Python nyelven íródott.**

Magyarázat

A helyes válasz az Option2, hiszen a DAG mozaikszó is a fenti meghatározásból származik amely directed acyclic graph azaz irányított körmentes gráf.

2. feladat 0/1 pont

Az alábbiak közül melyik kódrészlet rajzolja ki a képen látható DAG-ban lévő taskok közötti függőséget megfelelően?

```python
import datetime as dt

from airflow.models import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.latest_only_operator import LatestOnlyOperator
from airflow.utils.dates import days_ago
from airflow.utils.trigger_rule import TriggerRule

dag = DAG(
    dag_id='latest_only_with_trigger',
    schedule_interval=dt.timedelta(hours=4),
    start_date=days_ago(2),
    tags=['example']
)

latest_only = LatestOnlyOperator(task_id='latest_only', dag=dag)
task1 = DummyOperator(task_id='task1', dag=dag)
task2 = DummyOperator(task_id='task2', dag=dag)
task3 = DummyOperator(task_id='task3', dag=dag)
task4 = DummyOperator(task_id='task4', dag=dag, trigger_rule=TriggerRule.ALL_DONE)
```

Válasz

- (●) Option1:

  latest_only >> task1 >> [task4,task3]

  task2 >> [task4, task3]

  Ez a válasz helyes, de nem jelölted meg.

- ( ) Option2:

  task4 >> task1 >>latest_only

  task4 >> task2

  task3 >> task1 >> latest_only

  task3 >> task2

- ( ) Option3:

  DAG >> latest_only >> task1 >> [task4,task3]

  DAG >> task2 >> [task4, task3]

- ( ) Option4:

  latest_only >> task1 >> task4 >> task3

  task2 >> task4 >> task3

Magyarázat

Az Option1 a helyes válasz.

Az Option2 helytelen válasz, nem a gyermekektől irányitjuk a gráf pontjait a szülők felé

Az Option3 helytelen válasz, hiszen a DAG önmaga nem szerepelhet a függőségi listán mivel nem a default Operatorból származik hanem ezért nem is használható fel a >> operátorral a kódban ilyen szintaktikával.

Az Option4 helytelen válasz, a task3 és task4 egymással párhuzamosan futó feladatok

3. feladat 0/4 pont

Adott az alábbi kódrészlet amely egy Airflow job implementációjának részlete. Mely állítások **igazak** az alábbiak közül?

```python
def teradata_sql_execute(**kwargs):
    host = kwargs.get('host')
    username = kwargs.get('username')
    password = kwargs.get('password')
    query = kwargs.get('query')
    con = teradatasql.connect(host = host, user = username, password = password, tmode = 'TERA')
    cur = con.cursor()
    for q in query.split(';'):
        logging.info(q)
        cur.execute(q)
        con.commit()
    con.close()

def sql_reader(path):
    with io.open(path, encoding='latin-1') as f:
        return f.read()

op_list = list()

for sql in params['dwh_data_sqls']:
    dwh = PythonOperator(
        task_id = f"dwh_data_{sql}",
        dag = dag,
        python_callable = teradata_sql_execute,
        op_kwargs = {
        "host": "test_server.cba.hu",
        "username": "test_user",
        "password": "{{  var.value.test_user_password  }}",
        "query": sql_reader(os.path.dirname(__file__) + "/" + "resource/data" + "/" + sql)
        },
    )
    op_list.append(dwh)

    for i, v in enumerate(op_list):
        if i>0:
            op_list[i-1]>>op_list[i]
```

Válaszok

- [ ] Option1: **Futási időben a task_id sorában szereplő {sql} értékadásáért az Airflow által vezérelt jinja template engine felel.**
- [x] Option3: **Tegyük fel hogy a dwh_data_sqls lista két elemü és elemeinek neve test1.sql és test2.sql, viszont az test2.sql fájl nem található a /resource/data/ lokáción. Az Airflow DAG broken állapotba kerül mivel nem találja a kódban hivatkozott fájlt a megfelelő helyen.**
  Ez a válasz helyes, de nem jelölted meg.
- [x] Option4: **A globális paraméterekben megadott test_user_password jelszó már elévült. Ez nem okoz problémát a DAG leképezése során, futási időben találkozunk majd a Teradata_sql_execute visszatérési értékével amely beszámol majd a hiba okáról.**
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Option5: **Amennyiben nincs más taskunk az op_list listán futó for in iteráció előtt és után úgy a DAG ez által a ciklus által leképezett párhuzamosan futó/futtatható taskok listájából fog állni.**
- [x] Option6: **PythonOperator használatára a python nyelven írodott Teradata_sql_execute funkció miatt volt szükség**
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Az Option1 helytelen állítás, mert a dwh_data_{sql} ben felhasznált kapcsos zárójel a python szöveg formázási nyelvi eszköze. Ennek köszönhető hogy az {sql} értéke behelyettesítődik a for in ciklusban és nem a jinja-nak.

Az Option3 helyes állítás. A DAG nak szüksége van minden benne meghivatkozott komponensre.

Broken DAG: [/usr/local/airflow/dags/test_airflow_job/test_airflow_job.py] [Errno 2] No such file or directory: '/usr/local/airflow/dags/test_airflow_job/resource/test2.sql'

Az Option4 helyes állítás. A meghivatkozott Teradata_sql_execute funkció jelenléte elég a DAG generálásához az Airflownak. Annak tartalmának helyessége és ezzel egyetemben a Teradata DWH ra való behívás a megadott user+pw segítségével is ekkor derül ki.

Az Option5 helytelen állítás. A for in ciklusban levő op_list[i-1] >> op_list[i] szülő à gyermek kapcsolatokat alakít ki a soron következő feladatok között másképpen szekvenciálisan követik egymást a feladatok.

Az Option6 helyes állítás. Rengeteg módon meg lehet hívni egy adatbázist de ebben az esetben mivel a meghívott funkció python nyelven írodott így ezt csak egy PythonOperator képes futtatni az elvárt eredménnyel.

Kedves Versenyzők!

Az "Option2: Az op_kwargs lista "password" eleme az airflow globális változóiból keresi ki a test_user_password-höz tartozó értéket és helyettesíti azt be futási időben." válaszlehetőséget töröltük, mivel "op_kwargs" változó nem "lista", hanem "szótár" (dictionary), ez az elírás pedig zavaró lehetett.

Elnézést kérünk a kellemetlenségért!

4. feladat 0/1 pont

Az alábbiak közül mi az Airflow backfill parancs funkciója?

Válasz

- Option1: **Egy DAG elbukott lépésének ütemezett újraindítása.**
- Option2: **Egy adott DAG tól függőségben álló másik DAG elindítása amennyiben a függőségi feltétel teljesül.**
- Option3: **Egy DAG ban levő szülő-feladat újraindítása amennyiben a feladat maga elbukna.**
- Option4: **Egy DAG historikus periódusának újrafuttatása egy megadott időszakra nézve.**
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Az Option1 helytelen, ütemezett újrafuttatásra a retries és retry_delay DAG argumentek segítségével van lehetőség

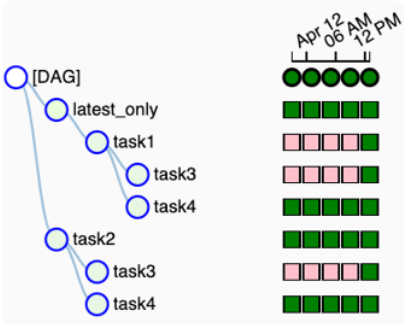
Az Option2 helytelen, ehhez a funkcionalitáshoz ExternalTaskSensor segítségére van szükség vagy valamilyen kézzel írt EventListenerre.

Az Option3 helytelen, ilyen funkcionalitás jelenleg nincs beépítve az Airflowban és kissé értelmetlen is lenne a mivoltja.

Az Option4 helyes válasz. Start_date és end_date változók meghatározásával megfuttatható egy DAG bármilyen multbeli időszakra és az adott időszakra leképzett DAG Run ok lefutnak ütemezetten.

5. feladat 0/3 pont

A 2-es feladatban látott DAG futásának az eredményét láthatjuk az Airflow fa nézetében. A zölden színezett mezők a sikeres futást, míg a rózsaszín mezők a kihagyott lépéseket mutatják. A DAG-ot délben élesítette a fejlesztő aznap reggel 4 órai start_date-tel és 2 óránkénti ütemezést adott meg. 1 óra elteltével az alábbi futási naplóval találkozott:



Az alábbi állítások közül melyek **igazak**?

Válaszok

- [x] Option1: **Mivel frissen élesített DAG-ról van szó amelynek a start_date paramétere a multra mutat így a látható 5 DAG run-t csak akkor láthatta az élesítés után 1 órával ha a DAG catchup paramétere true-ra volt állítva.**
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Option2: **A task1 kihagyott feladat egészen addig míg a DAG be nem hozta a multbéli futásokat és az aktuális execution datum paraméterhez érkezett. Feltételezhető hogy a trigger_rule változója a feladatnak ALL_DONE volt.**
- [x] Option3: **A latest_only feladat feltételezhető hogy LatestOnlyOperator-t használ amely magyarázza azt hogy a task1 múltbéli futásai ki lettek hagyva.**
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Option4: **A task3 szülője a task1 és task2 is. A task3 múltbéli kihagyása amiatt történt hogy a task3 trigger ruleja one_failed-re volt állítva.**
- [x] Option5: **A task4 feladat,- annak ellenére hogy a task1 és task2 gyermeke egyszerre, minden DAG run-ra lefutott. Ennek egy lehetséges módja hogy a trigger_rule-ja all_done-ra volt állítva.**
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Option6: **A task2 feladat trigger_ruleja all_success, emiatt minden esetben elbukik amennyiben a latest_only feladat is elbukik.**

Magyarázat

Az Option1 helyes állítás. A catch_up = true parameter biztosítja hogy a megadott start_date után az ütemezőnek megfelelő időintervallumonként DAG Run ok jönnek létre míg be nem érjük a jelenlegi időt és várjuk a következő futását a DAG-nak

Az Option2 helytelen állítás. A feltételezés alapján függetlenül attól hogy milyen állapotba kerültek a task1 szülői (jelen esetben csak a latest_only) a feladat meg kellett volna fusson zöld vagy piros eredménnyel.

Az Option3 helyes állítás. A feltételezés elfogadható mivel ez egy lehetséges magyarázata a futási naplóban látott eredményeknek.

Az Option4 helytelen állítás. One failed trigger_rule alapján bármelyik szülő piros azaz elbukott(failed) állapotba kerülése esetén futott volna meg vagy piros(failed) vagy zöld(success) állapottal visszatérve.

Az Option5 helyes állítás. All done esetén a szülők visszatérési állapotától függetlenül megfuthat egy feladat

Az Option6 helytelen állítás. A latest_only és task2 feladatok nem állnak semmilyen kapcsolatban.