3장. Airflow 스케줄링 & 데이터 증분 처리

#airflow #schedule #dataengineer #DAG #task

DAG안 테스크는 어디에 위치하던지 재시작 가능

Task와 Operator

- Operator는 단일 테스크를 나타낸다.
- DAG는 오퍼레이터 집합에 대한 실행을 오케스트레이션 하는 역할을 한다.
- Airflow 에서 task는 올바른 실행을 보장하기 위한 Operator의 wrapper / manager
  - ex) Operator의 종류 : BashOperator, PythonOperator , , , etc

3.1 정기적으로 실행하기

3.1.1 스케줄 간격 조정하기

- DAG 초기화 시, `schedule_interval` 인수를 설정하여 조정 - Default : None -> UI 또는 API를 이용한 수동 트리거
- `start_date` `end_date` 지정하여 스케줄링 가능

```
dag = DAG(
                  dag_id= "",
                  schedule_interval="@daily",
                  start_date=dt.datetime(year=2023, month=1, day=1),
                  end_date=dt.datetime(year=2023, month=1, day=5)
)
```

- 자주 사용하는 Airflow 프리셋
  - `@once` : 1회
  - `@hourly` : 매시간 1회
  - `@daily` : 매일 자정 1회
  - `@weekly` : 매주 일요일 자정 1회
  - `@monthly` : 매월 1일 자정 1회
  - `@yearly` : 매년 1월 1일 자정 1회

3.1.2 Cron 기반 스케줄 간격 설정

```
* * * * *

- 분(0-59), 시간(0-23), 일(1-31), 월(1-12), 요일(0-6)

ex)
0 * * * * = 매시간 정시
0 0 * * * = 매일 자정
45 23 * * SAT = 매주 토요일 23시 45분
```

3.1.3 빈도 기반 스케줄 간격

- cron 방식의 제약은 **특정 빈도마다 스케줄을 정의할 수 없다는 점**
- 일정 주기마다 스케줄 실행 가능
  - `datetime.timedelta()` : 빈도기바의 스케줄 사용 가능

3.2 데이터 증분 처리하기

- daily 실행되는 DAG에 대해, 모든 이벤트 전체 다운은 비효율적.

3.2.1 이벤트 데이터 증분 가져오기

- `BashOperator` 를 이용하여 데이터 가져오기 가능
  - 아래 curl 명령어 처럼 operator 생성

```
curl -O <url>?start_date=2023-01-01&end_date=2023-01-05
```

3.2.2 실행 날짜를 사용하여 동적 시간 참조

- 시간 workflow의 경우, TASK가 실제 수행되는 시점을 정의할 수 있는 추가변수 제공
  - `execution_date` : task의 시작시간을 나타내는 타임 스탬프
  - `next_excution_date` : 스케줄 간격의 종료 시간을 의미
  - 그 외 `prev_excution_date` 등 매개변수도 제공
- 각각의 변수들은 축약어 제공
  - excution_date 의 YYYY-MM-DD 의 표현 : `ds`
  - `next_ds` , `next_ds_nodash` 등등
- 실행시간 변수 / 축약어를 사용하여 데이터 파티셔닝 가능
  - 예) 파일 명 -> {{ds}}.json