3장_2 Backfill, Catch up

Airflow 운용 시, 현재 시점 보다 과거의 배치 작업을 주기적으로 진행하거나, 특정 조건을 골라 재실행 하고 싶은 상황에 활용 가능하다.

> **Summary**
>
> - Airflow를 통해 과거의 시작 날짜부터 과거 간격을 정의할 수 있음.
>   - 과거 데이터셋 로드하거나 분석하기 위해 DAG과거 기록 실행 가능 **백필**

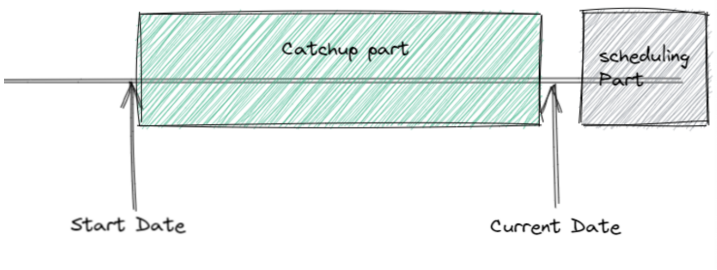
3.2.1 과거 시점 작업 수행

- AIrflow는 실행되지 않은 과거 스케줄 간격 예약하고 실행
  - 과거 날짜를 지정하고 DAG 활성화 시, 현재 시간 실행전 모든 간격이 생성됨
- 모든 과거 스케줄 간격마다 작업을 수행하는 것이 아닌 현재 기준으로만 수행 하는 법
  - `catchup` : false 가장 최근 스케줄 간격에 대해서만 실행

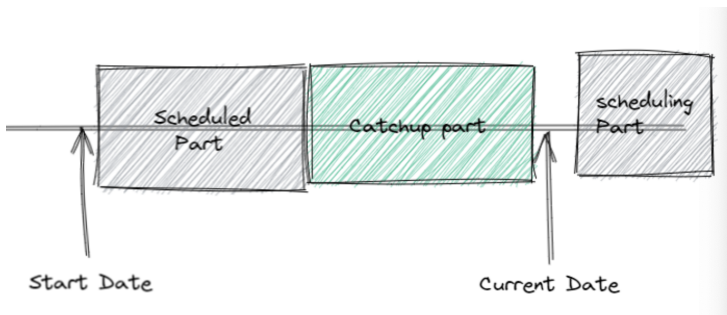
```
dag = DAG(
        dag_id="08_no_catchup",
        schedule_interval="@daily",
        start_date=dt.datetime(2023, 1, 3),
        end_date=dt.datetime(2023, 1, 7),
        catchup=False,
)
```

- Airflow 구성 파일 configure file 에서 `catchup_by_default` 설정할 수 있음

Catch Up



- Start Date가 현재로 부터 1년전 이라면, Start_Date 부터 스케줄링 된 부분의 DagRun 들이 비게 됨.
- `Catch up = True` 시, 이 부분이 순차적으로 Dag Run을 실행시켜 빈공간 채운 후 스케줄링 진행.
  - 예) 데이터 이관 시, 과거 날짜 기준으로 일정 주기 마다 데이터 집계 하여 이관할 경우
- 또한 중간에 스케줄 되었던 Dag가 중단되어 빈 경우, 다시 채워주기 가능



> **Warning**
>
> DagRun이 한번에 실행 되기 때문에 서버 과부하 올 수 있음 -> 속성값 설정 필요

BackFill

DAG가 이미 배포되어 실행 중이며, 해당 DAG를 사용하여 시작 날짜 이전에 데이터를 처리하기 원할 때 사용한다.

- 전체 재시작을 하는데 사용하지 않고, 일정 기간동안 실패한 작업에 대해 재실행하는데 사용
- 예시)

```
airflow dags backfill -s 2023-01-05 -e 2023-02-05 example_dag
```

예시 코드 및 결과화면

- 예시 코드
  - catchup=true
  - 시작일~ 종료일 까지 cron에 따라 수행된 이후 스케줄링 수행

```
# catchup을 사용한 데이터 백필
import pendulum
import datetime as dt
from airflow import DAG
from airflow.operators.python import PythonOperator

kr_tz = pendulum.timezone("Asia/Seoul")

dag = DAG(
        dag_id="08_no_catchup",
        schedule_interval="*/2 * * * *",
        start_date=dt.datetime(2023, 1, 6, 17, 45, tzinfo=kr_tz),
        end_date=dt.datetime(2023, 1, 6, 18, 5, tzinfo=kr_tz),
        # catchup=False,
)

def _print_hello():
        print("Hello, It's work!")

print_hello = PythonOperator(
task_id="print_hello", python_callable=_print_hello, dag=dag
)

print_hello
```

- 결과 화면

		State	Dag Id	Logical Date	Run Id	Run Type	Queued At	Start Date	End Date	Note	External Trigger	Conf	Durati
☐		success	08_no_catchup	2023-01-06, 17:46:00	scheduled__2023-01-06T08:46:00+00:00	scheduled	2023-01-06, 20:34:45	2023-01-06, 20:34:45	2023-01-06, 20:34:47		False	{}	2s
☐		success	08_no_catchup	2023-01-06, 17:48:00	scheduled__2023-01-06T08:48:00+00:00	scheduled	2023-01-06, 20:34:47	2023-01-06, 20:34:47	2023-01-06, 20:34:49		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 17:50:00	scheduled__2023-01-06T08:50:00+00:00	scheduled	2023-01-06, 20:34:49	2023-01-06, 20:34:49	2023-01-06, 20:34:50		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 17:52:00	scheduled__2023-01-06T08:52:00+00:00	scheduled	2023-01-06, 20:34:51	2023-01-06, 20:34:51	2023-01-06, 20:34:53		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 17:54:00	scheduled__2023-01-06T08:54:00+00:00	scheduled	2023-01-06, 20:34:53	2023-01-06, 20:34:53	2023-01-06, 20:34:55		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 17:56:00	scheduled__2023-01-06T08:56:00+00:00	scheduled	2023-01-06, 20:34:56	2023-01-06, 20:34:56	2023-01-06, 20:34:57		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 17:58:00	scheduled__2023-01-06T08:58:00+00:00	scheduled	2023-01-06, 20:34:57	2023-01-06, 20:34:57	2023-01-06, 20:34:59		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 18:00:00	scheduled__2023-01-06T09:00:00+00:00	scheduled	2023-01-06, 20:35:00	2023-01-06, 20:35:00	2023-01-06, 20:35:01		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 18:02:00	scheduled__2023-01-06T09:02:00+00:00	scheduled	2023-01-06, 20:35:01	2023-01-06, 20:35:01	2023-01-06, 20:35:03		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 18:04:00	scheduled__2023-01-06T09:04:00+00:00	scheduled	2023-01-06, 20:35:03	2023-01-06, 20:35:03	2023-01-06, 20:35:04		False	{}	1s
☐		success	08_no_catchup	2023-01-06, 20:34:44	manual__2023-01-06T11:34:44.749870+00:00	manual	2023-01-06, 20:34:44	2023-01-06, 20:34:45	2023-01-06, 20:34:45		True	{}	<1s

- catchup=False 인 경우, 현재 시간 기준으로 전 작업만 수행됨