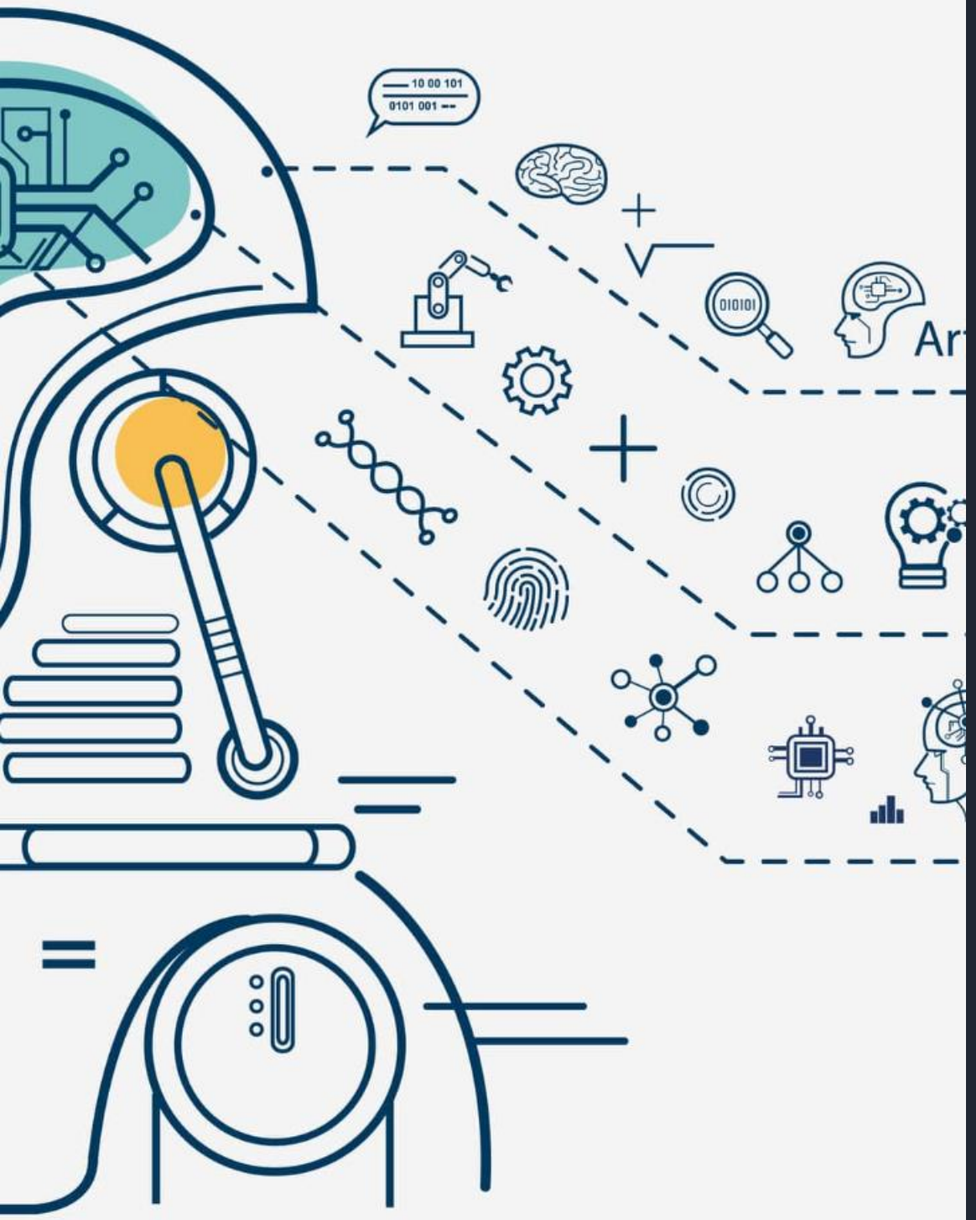
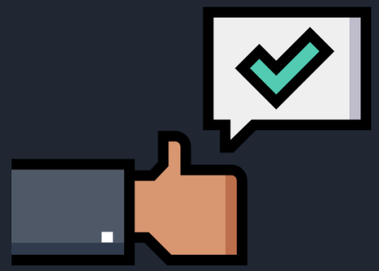


Airflow를 활용한 기계 학습 워크플로우의 기본 사용법

by Seungryul Lee

Airflow 소개



용어

Airflow는 대용량 데이터 파이프라인 설계를 위한 오픈소스 워크플로우(Workflow) 관리 플랫폼입니다

용도

데이터 처리, 자동화된 작업, 오케스트레이션 등에 활용되고 있습니다.

장점

다양한 데이터소스에 접근 가능하며 확장성과 안정성이 뛰어나다는 특징이 있습니다.

Machine learning workflow

왜?

기계 학습 워크플로우는 데이터 과학자들에게 필수적입니다. 워크플로우에 따라 프로세스를 간소화하고 자동화할 수 있습니다.

누구를 위해?

기계 학습 워크플로우는 데이터 과학자, 소프트웨어 엔지니어, 머신 러닝 엔지니어, 데이터 엔지니어 등 다양한 직종에 필요합니다.

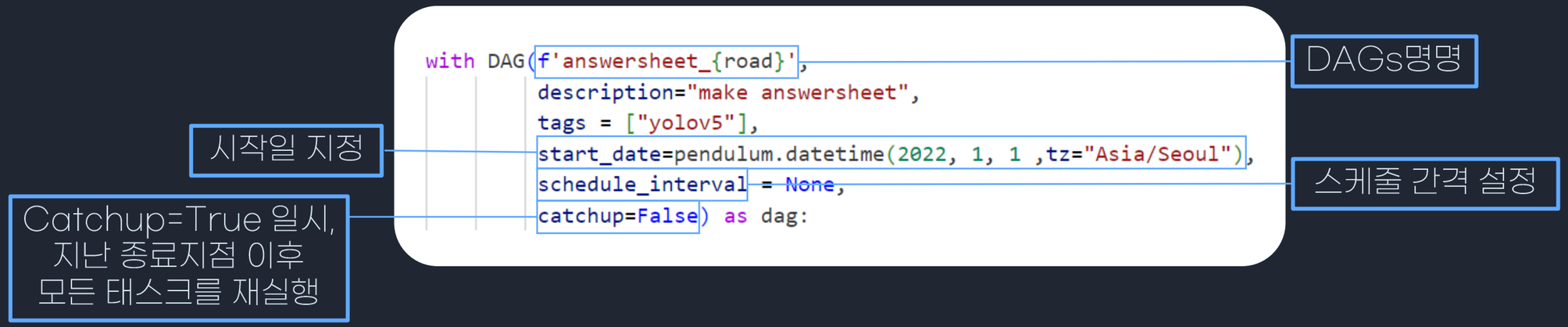
어떻게?

기계 학습 워크플로우는 데이터 수집, 전처리, 모델링, 검증, 배포 등의 단계를 자동화할 수 있습니다.

Airflow의 기본 구성 요소-DAGs

1 DAGs

실행될 수 있는 작업들을 정의한 그래프입니다. DAG(Directed Acyclic Graph)는 방향성이 존재하며 사이클이 없음을 의미합니다.



Airflow의 기본 구성 요소 - Operators

2 Operators

DAG 안에서 수행될 수 있는 작업 단위입니다. Operator는 Task라고도 불립니다. 예를 들어, PythonOperator는 호출할 파이썬 함수를 실행한다는 의미입니다.

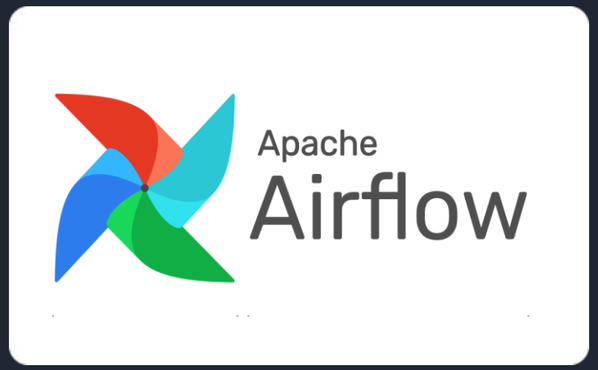
오퍼레이터명

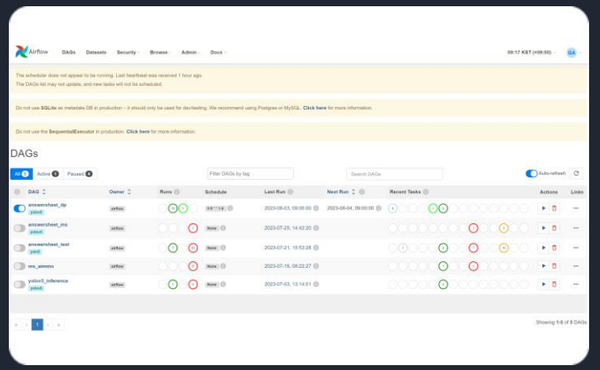
Xcom을 활용한 인스턴스 호출

```
unzip_files = PythonOperator(
    task_id="unzip_files",
    python_callable=_unzip_files,
    op_kwargs={'extract_path' :"{{ti.xcom_pull(task_ids='set_variable', key = 'extract_path')}}",
               }
)
```

Python함수명

Airflow 설정





설치 방법

Airflow는 Linux 기반 운영체제에서 작동 가능하며 이에 따라 Docker를 활용한 설치 또는 Linux 기반 운영체제 설치를 통해 활용할 수 있으며, Python 패키지 관리자인 pip를 이용해 설치할 수 있습니다.

설정 파일

Airflow는 기본적으로 airflow.cfg라는 설정 파일을 사용합니다. 이 파일 안에서 DAGs 폴더 위치, 수집 주기 등을 설정할 수 있습니다.

웹 인터페이스

Airflow는 웹 인터페이스를 제공합니다. 웹 인터페이스를 통해 DAGs의 실행 상태, Log를 확인할 수 있으며 필요한 경우 수동으로 작업을 실행할 수도 있습니다.

Airflow를 사용한 데이터 처리

1

데이터 수집

Airflow를 이용해 다양한 데이터 소스(API, 외부 데이터베이스 등)에서 데이터를 수집할 수 있습니다.

2

데이터 전처리

Python으로 작성된 Operator를 통해 데이터 전처리 작업을 수행할 수 있습니다. 예를 들어, Pandas를 이용해 데이터를 정제하고 가공할 수 있습니다.

3

모델링

머신 러닝 모델을 학습하고 적용하기 위해 Operator를 이용할 수 있습니다. Scikit-learn, Tensorflow 등의 라이브러리를 이용해 모델을 학습하고, 예측 결과를 다양한 형태로 출력할 수 있습니다.

4

데이터 저장

Operator를 이용해 데이터 처리 결과를 다양한 데이터 소스(데이터베이스, 파일 등)에 저장할 수 있습니다.

Airflow를 사용한 스케줄링

스케줄링

Airflow를 이용해 작업을 주기적으로 스케줄링할 수 있습니다. 예를 들어, 매일 10시 정각에 데이터 수집 및 전처리 작업을 자동으로 수행하도록 설정할 수 있습니다.

Cron 표현식

스케줄링을 할 때 Cron 표현식을 사용할 수 있습니다. 이를 통해 작업을 정해진 시간에 주기적으로 수행할 수 있습니다.

Task 단위 스케줄링

Airflow의 DAGs는 Task 단위로 스케줄링이 가능합니다.
이를 통해 특정 Task가 실행될 때 다른 Task가 자동으로 실행되도록 구성할 수 있습니다.

Cron 표현식 구성

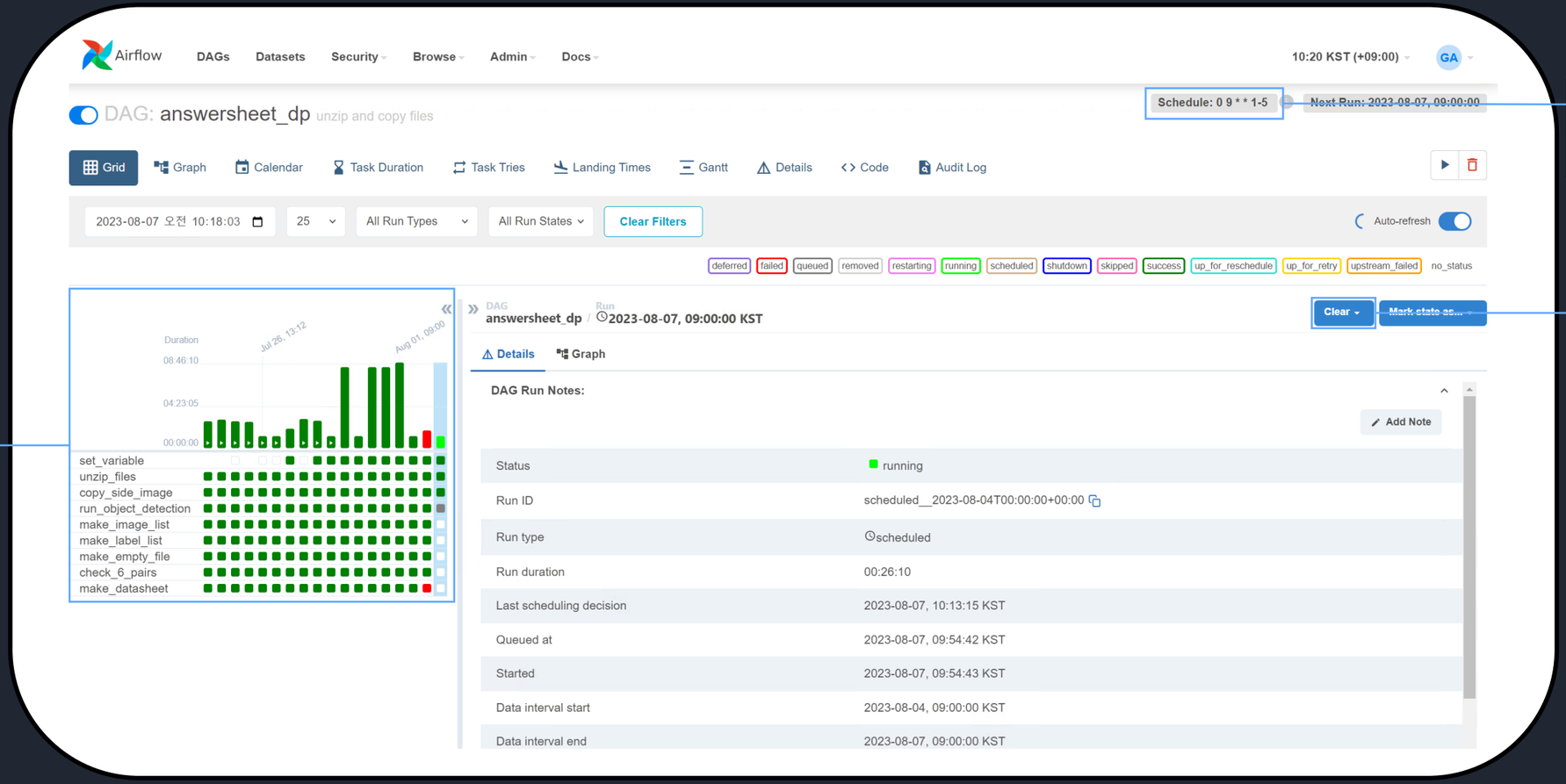
{분} {시} {일} {월} {요일}

ex) 평일 01시 10분 : 10 1 * * 1-5

```
with DAG(f'answersheet_{road}',
         description="make answersheet",
         tags = ["yolov5"],
         start_date=pendulum.datetime(2022, 1, 1 ,tz="Asia/Seoul")
         schedule_interval = '0 9 * * 1-5',
         catchup=False) as dag:
```

평일 9시 실행

Airflow Grid 예시 - 대왕판교



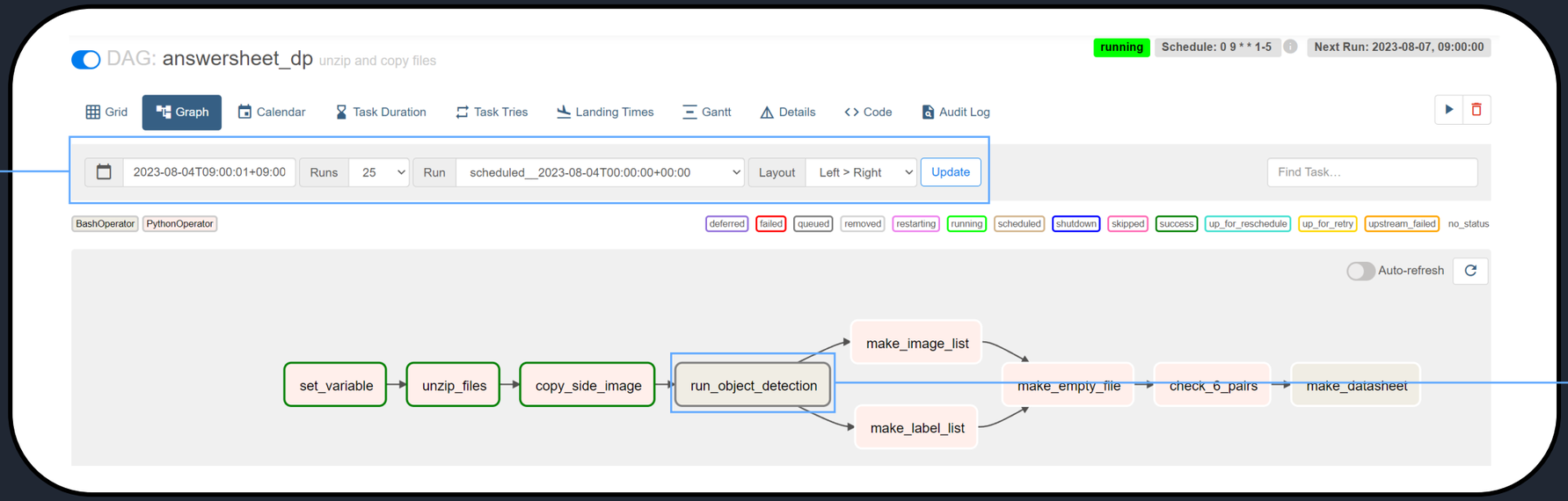
Cron을 활용한 태스크 스케줄

각 Dag의 태스크 처리 가능

실행 일자별 태스크 확인 및 처리여부 확인 가능

Airflow Graph 예시 - 대왕판교

일자별, 시간별 태스크 처리여부 확인 가능



각 태스크의 status 확인 가능

Airflow의 한계

WSL

A100 GPU는 WSL환경에서 실행 할 수 없으며, 다른 GPU를 사용해야 합니다. (필요시, 멀티OS를 활용하는 방안도 가능)

XCom

XCom 인스턴스가 큰 경우, 오류가 발생할 수 있으므로 custom XCom backend를 설정해야합니다.

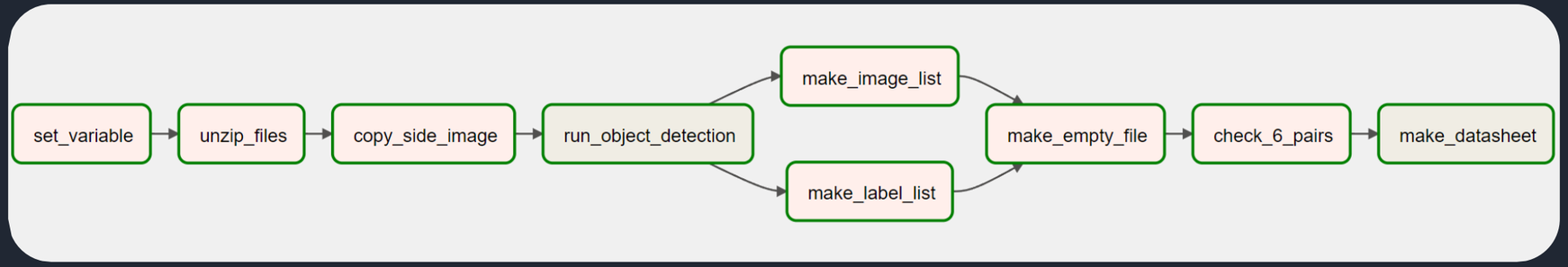
Complexity

학습곡선이 가파르고 일부 사용사례의 경우 코딩 기술과 사용자 지정 연산자가 필요하며 디버그 및 모니터링이 어려울 수 있습니다.

What is XCom?

서로 다른 operator 간의 데이터를 공유하기 위해 필요한 교차 통신 매커니즘 입니다.

Airflow 적용 예시 - 대왕판교



1

데이터 수집

Dags: set variable , unzip files, copy side image

- Date를 활용한 변수 지정
- 파일 압축 해제
- 객체 검지를 위한 이미지 복사

2

Yolov5 객체 검출

Dags: run object detection

- Bash Operator를 활용한 YOLOv5 객체 검출

3

데이터 전처리

Dags: make image list, make label list, make empty file, check 6 pairs

- 이미지 명단과 레이블 명단 비교
- 레이블 명단 내에 없는 빈 레이블 생성
- 이미지 세트 수가 6개 이하인 경우 추가 레이블 생성

4

데이터 저장

Dags: make datasheet

- 기존 저장 양식에 부합하는 데이터 세트 생성