

## Nginx Access Log 적재 파이프라인 구현

요구사항에 따라 먼저 환경을 정리한 후, 필요한 도구들을 설치하였습니다. Airflow 는 사용해본 적이 없지만, 어떻게 사용하는지 스스로 학습하여 구현해 보았습니다.

1. **docker-compose-airflow.yaml** 에서 제공된 Airflow 이미지를 사용하는 대신, 새로운 Dockerfile 을 작성하여 사용하였습니다. 이유는 PySpark 를 함께 사용하기로 하였기 때문에, 커스터마이징하여 작성하는 것이 더 편리하다고 판단하였습니다.

```
kakaoenterprise-bigdata-platform-assessment_YOSHIMA > log-pipeline > docker-compose-airflow.yaml
47 x-airflow-common:
48   # In order to add custom dependencies or upgrade provider distributions you
49   # Comment the image line, place your Dockerfile in the directory where you
50   # and uncomment the "build" line below, Then run `docker-compose build` to
51   # image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:3.0.1}
52   build:
53     context: .
54     dockerfile: Dockerfile
55   # build:

kakaoenterprise-bigdata-platform-assessment_YOSHIMA > Dockerfile
1 FROM apache/airflow:3.0.1-python3.12
2
3 USER root
4
5 RUN apt-get update \
6     && apt-get install -y --no-install-recommends \
7         default-jdk-headless \
8         procps \
9     && rm -rf /var/lib/apt/lists/*
10
11 ENV JAVA_HOME=/usr/lib/jvm/java-1.17.0-openjdk-amd64
12 ENV PATH="${JAVA_HOME}/bin:${PATH}"
13
14 USER airflow
15
16 RUN pip install --no-cache-dir pyspark==3.5.1
```

2. PySpark 코드를 작성하여 **/log-pipeline/airflow/dags/** 경로에 두었습니다.
  - logs\_to\_minio.py: PySpark 기반 코드로, Nginx 에 있는 로그 파일을 가져와 Minio 에 Parquet 포맷으로 저장합니다.

```
kakaoenterprise-bigdata-platform-assessment_YOSHIMA > log-pipeline > airflow > dags > logs_to_minio.py > ...
1  import datetime
2  from pyspark.sql import SparkSession
3  import pytz
4
5  tz = pytz.timezone('Asia/Seoul')
6  now = datetime.datetime.now(tz)
7  day_str = now.strftime("%Y%m%d")
8  hour_str = now.strftime("%H")
9  print("Schedule: ", day_str, ' hour=', hour_str)
10
11  spark = SparkSession.builder \
12      .appName("NginxLogToMinIO") \
13      .config("spark.jars.packages", ", ".join(["org.apache.hadoop:hadoop-aws:3.3.5", "org.apache.hadoop:hadoop-common:3.3.5"])) \
14      .config("spark.hadoop.fs.s3a.endpoint", "http://minio:9000") \
15      .config("spark.hadoop.fs.s3a.access.key", "minio_admin") \
16      .config("spark.hadoop.fs.s3a.secret.key", "minio_password") \
17      .config("spark.hadoop.fs.s3a.path.style.access", "true") \
18      .config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem") \
19      .getOrCreate()
20
21  print("Read logs from Nginx...")
22  df = spark.read.option("multiline", "true").json("/var/log/nginx")
23  df.show()
24
25  print("Parse to parquet and put into Minio...")
26  minio_path = f"s3a://parquet/day={day_str}/hour={hour_str}"
27  df.write.mode("overwrite").parquet(minio_path)
28
29  spark.stop()
30
31  print("SUCCESSFULLY WRITE PARQUET TO MINIO!")
32
```

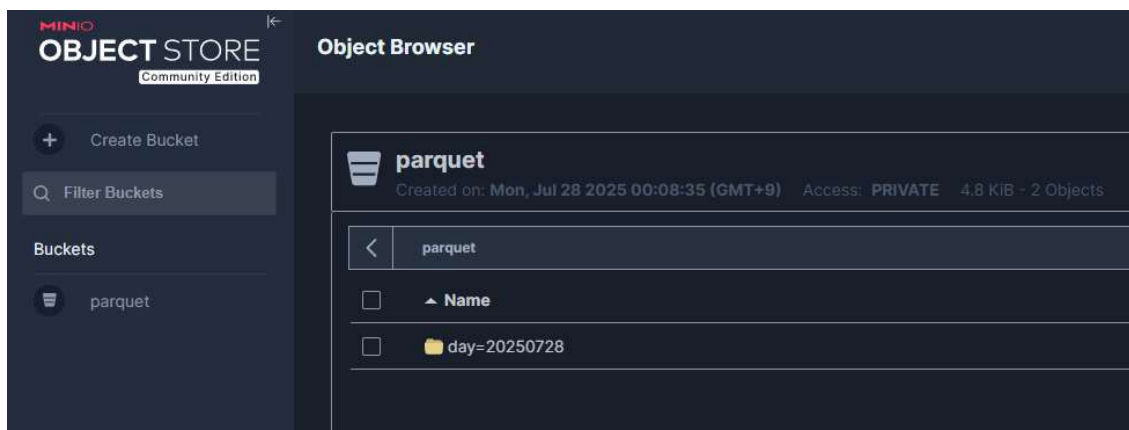
- **hourly\_logs\_to\_minio.py**: Airflow 에서 스케줄링을 설정한 Python 코드입니다.

```
kakaoenterprise-bigdata-platform-assessment_YOSHIMA > log-pipeline > airflow > dags > hourly_logs_to_minio_run.py >
1  from airflow import DAG
2  from airflow.operators.bash import BashOperator
3  from datetime import datetime, timedelta
4
5  default_args = {
6      'owner': 'airflow',
7      'start_date': datetime(2024, 1, 1),
8      'retries': 1,
9      'retry_delay': timedelta(minutes=5),
10 }
11
12  dag = DAG(
13      'run_external_script_hourly',
14      default_args=default_args,
15      description='Run external script from log-pipeline every hour',
16      schedule='@hourly',
17      catchup=False,
18  )
19
20  script_path = '/opt/airflow/dags/logs_to_minio.py'
21
22  run_script = BashOperator(
23      task_id='run_log_pipeline_script',
24      bash_command=f'python {script_path}',
25      dag=dag,
26  )
27
```

3. nginx/logs 와 minio/data 경로는 docker-compose-airflow.yaml 의 volumes 항목에서 설정하였습니다.

```
ssment_YOSHIMA > log-pipeline > docker-compose-airflow.yaml
47 x-airflow-common:
57   environment:
78     AIRFLOW_CONFIG: '/opt/airflow/config/airflow.
79   volumes:
80     - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/da
81     - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/lo
82     - ${AIRFLOW_PROJ_DIR:-.}/config:/opt/airflow/
83     - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow
84+     - ./nginx/logs:/var/log/nginx
85+     - ./minio/data:/data
86   user: "${AIRFLOW_UID:-50000}:0"
```

Minio 웹 UI (<http://localhost:9001/>)에 접속하여, bucket 폴더(Parquet)를 생성합니다.

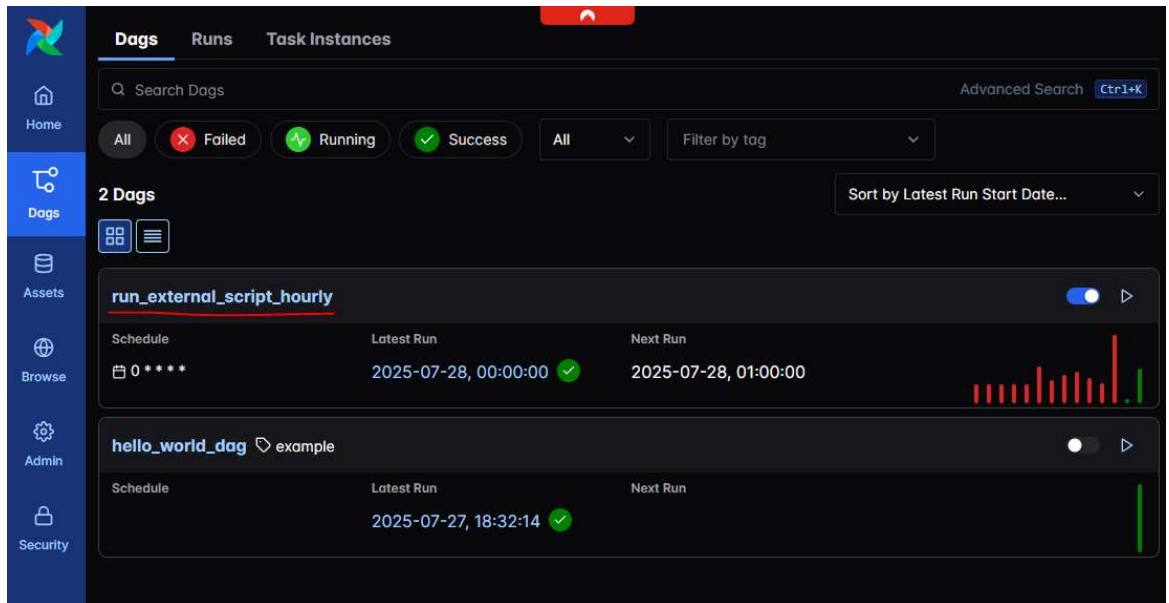


또한 테스트를 위해 nginx/logs 에 과제 파일에 포함된 JSON 형식의 더미 데이터를 file-access.json 이라는 이름으로 저장하였습니다.

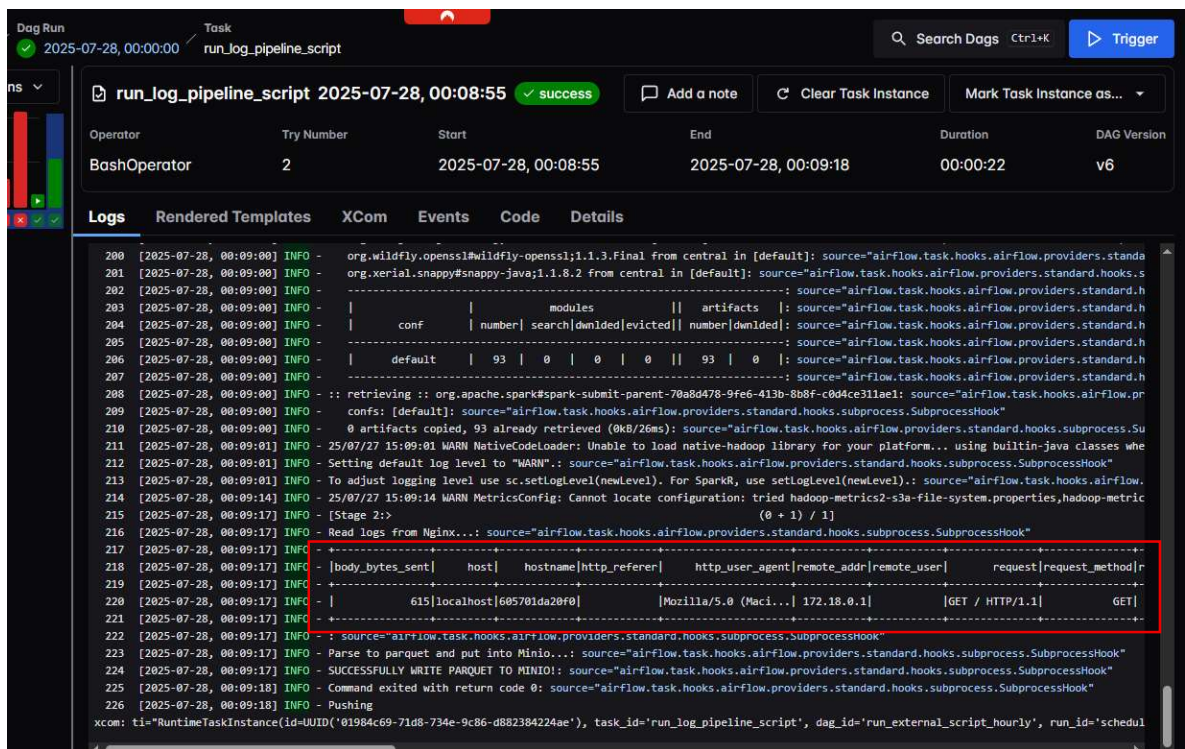
4. 실행 시에는 log-pipeline 폴더로 이동한 뒤 아래 명령어를 입력하면 됩니다:

```
docker-compose -f docker-compose.yaml -f docker-compose-airflow.yaml up -d
```

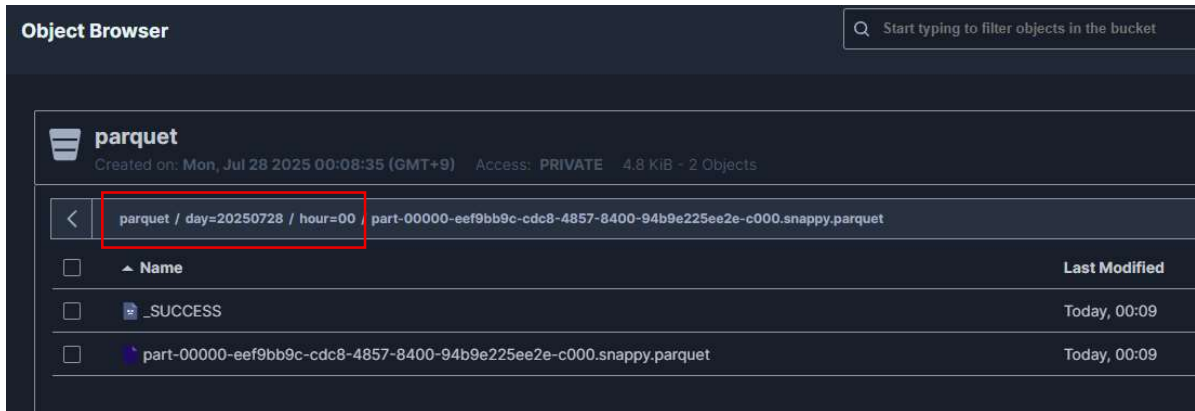
5. Airflow 웹 UI (<http://localhost:8080/>)에 접속하여 DAGs 메뉴를 클릭하면, 설정된 스케줄링 프로세스를 확인할 수 있습니다. 해당 프로세스는 1 시간 간격으로 실행됩니다.



실행 로그를 통해 자세한 내용을 확인할 수 있으며, JSON 데이터가 어떻게 처리되는지도 로그에 기록해두었습니다.



6. 1 시간 간격으로 Minio 에 Parquet 파일이 생성되는 것을 Minio 웹 UI 에서도 확인하실 수 있습니다.



7. 작업을 종료하고 싶을 경우에는 아래 명령어를 입력하시면 됩니다:

```
docker-compose -f docker-compose.yaml -f docker-compose-airflow.yaml down
```

**PySpark** 를 사용한 이유: Apache Spark 는 빅데이터 처리에 효과적인 도구로, Scala (Java 기반) 또는 PySpark 를 주로 사용합니다. 저는 PySpark 를 자주 사용해왔고, 이번 과제에도 적합하다고 판단하여 사용하였습니다.

**Airflow** 는 처음 사용해보았으나, 짧은 시간 내에 학습하고, 스케줄링 구조를 이해하여 설계하고 구현해보았습니다.

이 문서를 통해 제가 수행한 내용을 명확히 전달드릴 수 있기를 바라며, 좋은 평가 부탁드립니다.

감사합니다,

요시마 드림