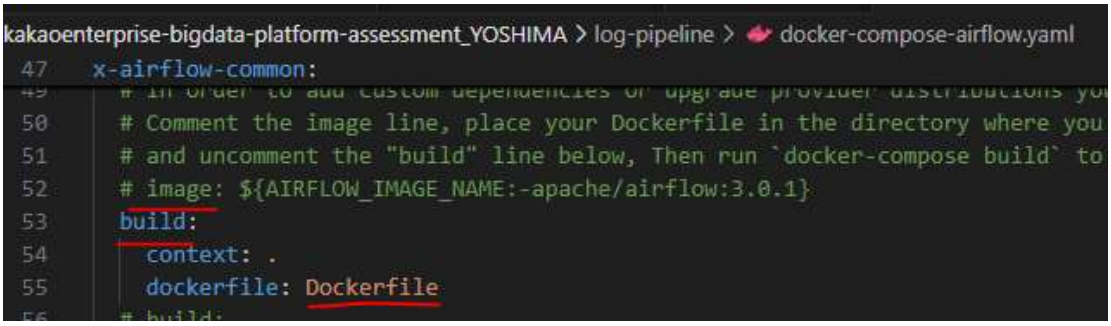
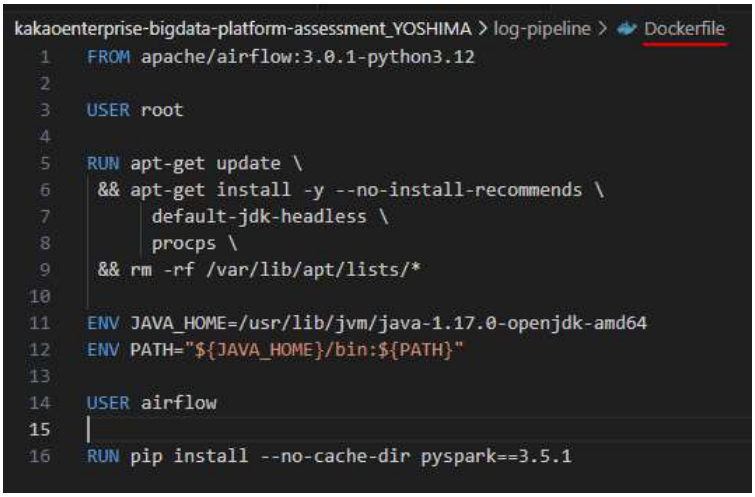
Nginx Access Log 적재 파이프라인 구현

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.





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 > ...

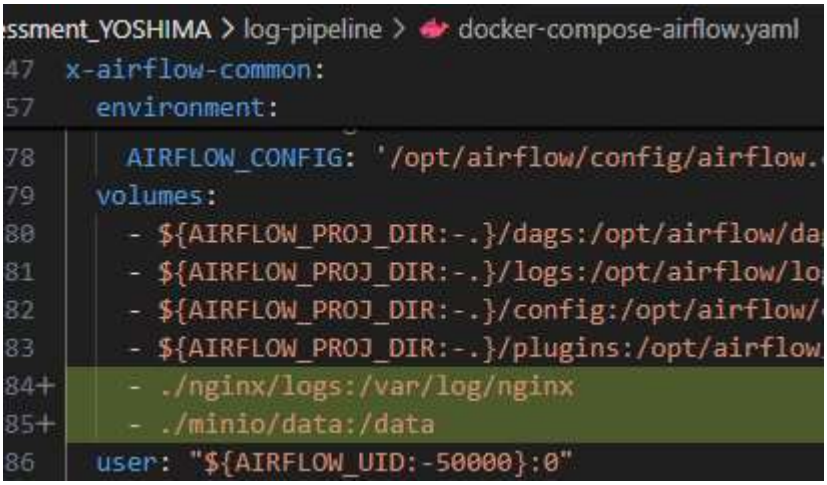
```python
import datetime
from pyspark.sql import SparkSession
import pytz

tz = pytz.timezone('Asia/Seoul')
now = datetime.datetime.now(tz)
day_str = now.strftime("%Y%m%d")
hour_str = now.strftime("%H")
print("Schedule: ", day_str, ' hour=', hour_str)

spark = SparkSession.builder \
    .appName("NginxLogToMinIO") \
    .config("spark.jars.packages", ",".join(["org.apache.hadoop:hadoop-aws:3.3.5", "org.apache.hadoop:hadoop-common:3.3.5"])) \
    .config("spark.hadoop.fs.s3a.endpoint", "http://minio:9000") \
    .config("spark.hadoop.fs.s3a.access.key", "minio_admin") \
    .config("spark.hadoop.fs.s3a.secret.key", "minio_password") \
    .config("spark.hadoop.fs.s3a.path.style.access", "true") \
    .config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem") \
    .getOrCreate()

print("Read logs from Nginx...")
df = spark.read.option("multiLine", "true").json("/var/log/nginx")
df.show()

print("Parse to parquet and put into Minio...")
minio_path = f"s3a://parquet/day={day_str}/hour={hour_str}"
df.write.mode("overwrite").parquet(minio_path)

spark.stop()

print("SUCCESSFULLY WRITE PARQUET TO MINIO!")
```

- **hourly_logs_to_minio.py**: Airflow 에서 스케줄링을 설정한 Python 코드입니다.

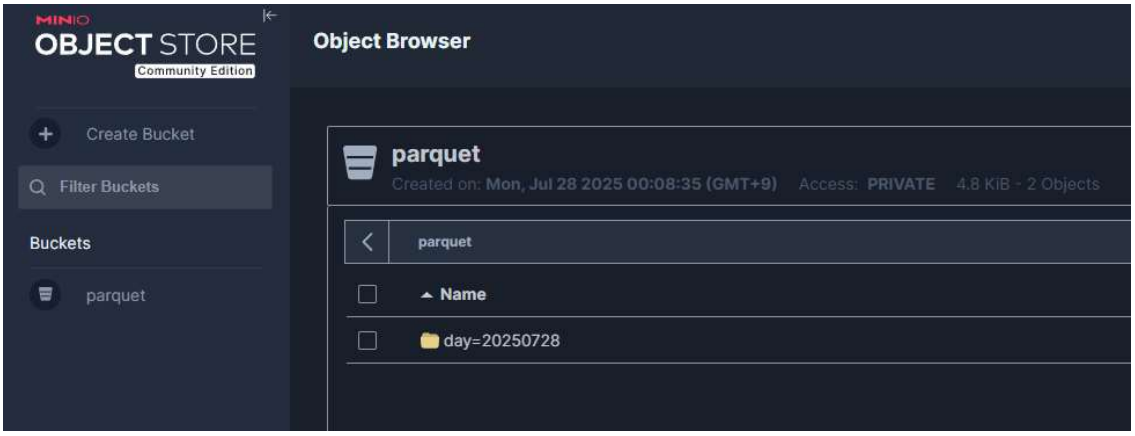
kakaoenterprise-bigdata-platform-assessment_YOSHIMA > log-pipeline > airflow > dags > hourly_logs_to_minio_run.py >

```python
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2024, 1, 1),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG(
    'run_external_script_hourly',
    default_args=default_args,
    description='Run external script from log-pipeline every hour',
    schedule='@hourly',
    catchup=False,
)

script_path = '/opt/airflow/dags/logs_to_minio.py'

run_script = BashOperator(
    task_id='run_log_pipeline_script',
    bash_command=f'python {script_path}',
    dag=dag,
)
```

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



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

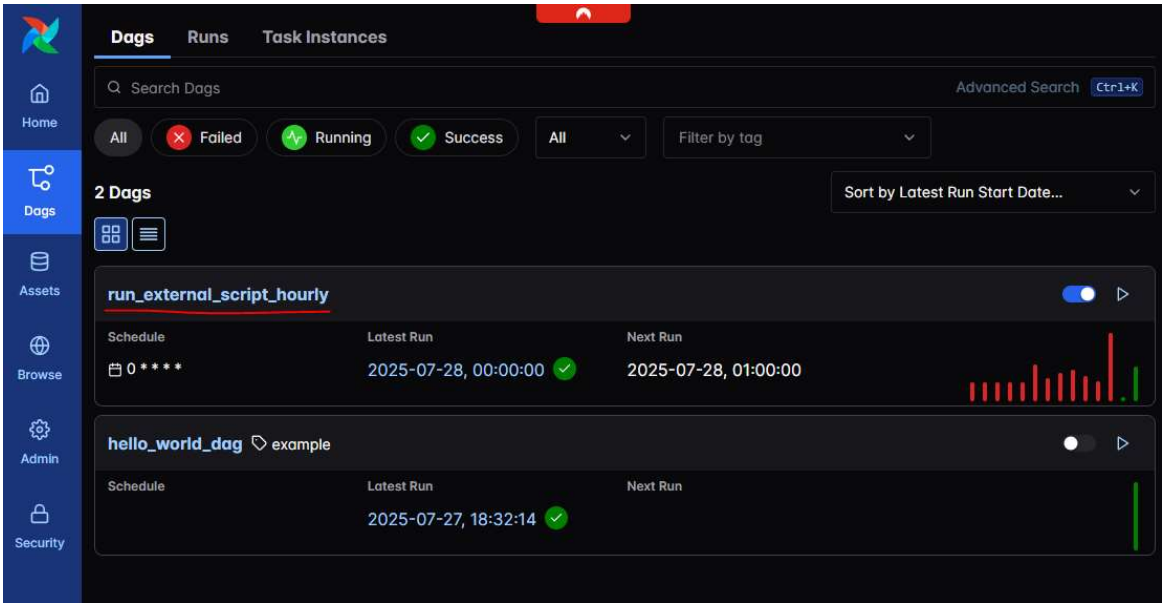


또한 테스트를 위해 nginx/logs 에 과제 파일에 포함된 JSON 형식의 더미 데이터를 file-access.json 이라는 이름으로 저장하였습니다.

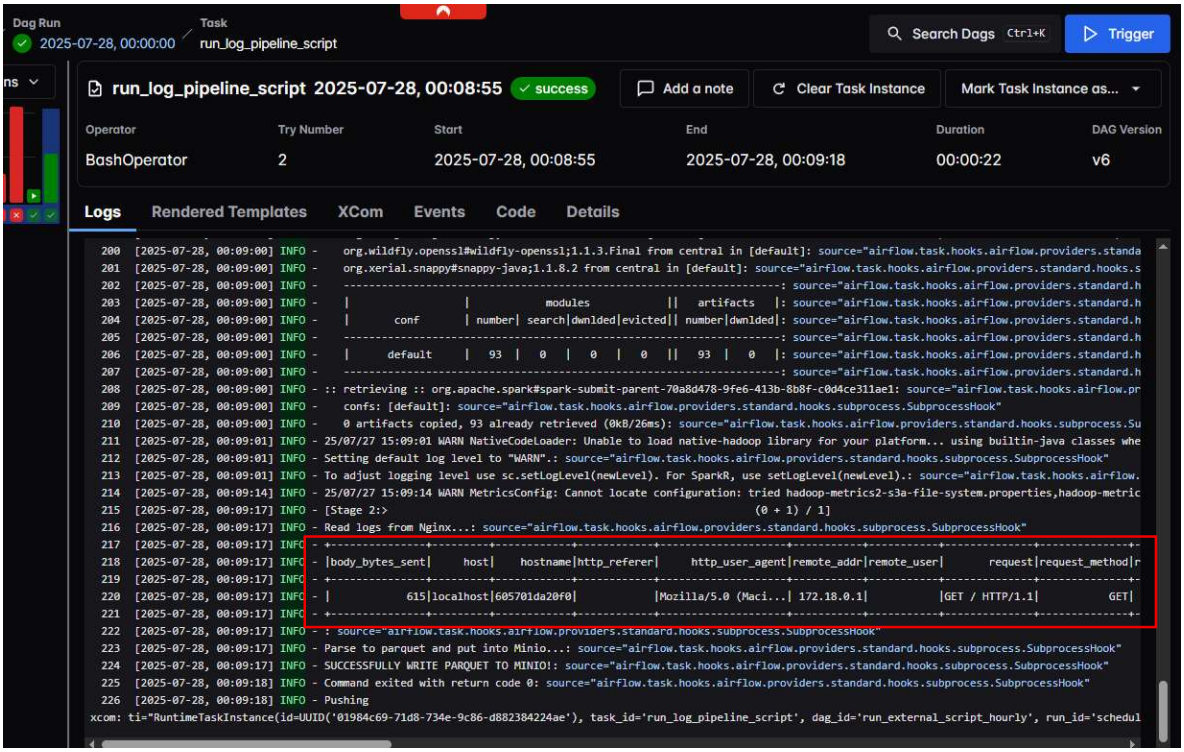
4. 실행 시에는 log-pipeline 폴더로 이동한 뒤 아래 명령어를 입력하면 됩니다:

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

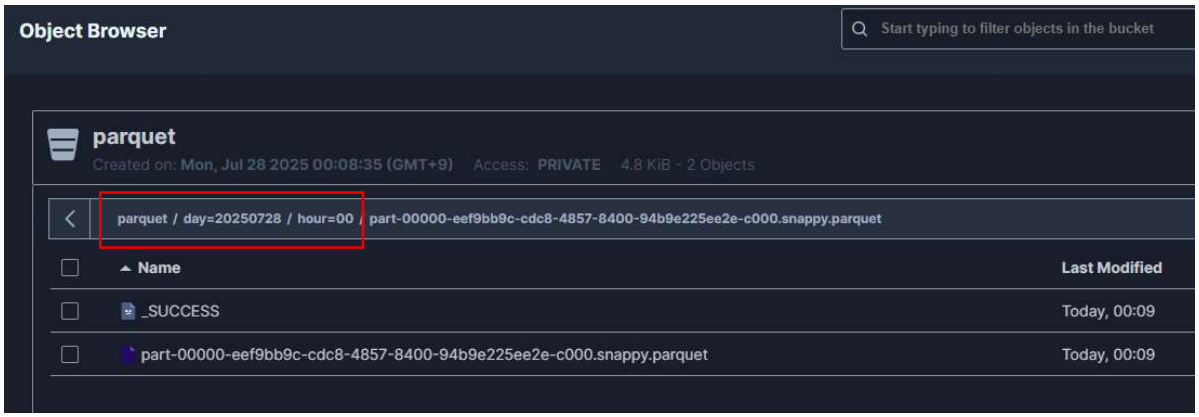
5. Airflow 웹 UI (http://localhost:8080/)에 접속하여 DAGs 메뉴를 클릭하면, 설정된 스케줄링 프로세스를 확인할 수 있습니다. 해당 프로세스는 1 시간 간격으로 실행됩니다.



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