

## 데이터마이닝 1차 팀과제 script

2025-04-08

# Table of contents

<b>1</b>	<b>Airflow 소개</b>	<b>2</b>
1.1	개요 . . . . .	2
1.2	workflow tool이란? . . . . .	2
1.3	데이터 분석 분야에서의 workflow tool . . . . .	2
1.4	workflow 활용 사례 . . . . .	3
1.4.1	국내 기업 . . . . .	3
1.4.2	해외 기업 . . . . .	3
1.5	Workflow Tool 다운로드 통계 . . . . .	4
1.6	Airflow github star 수 . . . . .	4
1.7	Airflow를 사용하는 직종 비율 . . . . .	5
1.8	Airflow 사용자들의 만족도 . . . . .	6
1.9	채용 시장에서의 Airflow 수요 . . . . .	7
1.9.1	주요 기업 JD 발체 예시 . . . . .	7
<b>2</b>	<b>Airflow 개념 &amp; Demo</b>	<b>10</b>
2.1	Airflow 개념 . . . . .	10
2.1.1	동작 원리 . . . . .	10
2.2	Airflow Demo . . . . .	11
2.2.1	개요 . . . . .	11
2.2.2	Airflow 설치 . . . . .	14
2.2.3	DAG 설정 . . . . .	16
2.2.4	Connections 설정 . . . . .	18
2.2.5	DAG 실행 . . . . .	19
2.2.6	결과 . . . . .	21

# 1. Airflow 소개

## 1.1 개요

데이터 분석 및 엔지니어링 분야에서는 복잡한 데이터 처리 작업을 자동화하고 관리하기 위해 워크플로우 관리 도구를 도입합니다. 특히 대규모 데이터 처리 환경에서는 거의 필수적으로 사용하죠.

Apache Airflow는 현재 가장 널리 사용되는 워크플로우 관리 도구 중 하나로, Python으로 작업 흐름을 정의하고 스케줄링, 모니터링, 오류 처리 등의 기능을 제공합니다.

## 1.2 workflow tool이란?

워크플로우 툴(Workflow Tool)은 복잡한 작업 과정을 자동화하고 관리하는 소프트웨어를 칭입니다. 쉽게 말해, 여러 단계의 작업을 순서대로 실행하고 모니터링할 수 있게 도와주는 도구죠.

예를 들어보겠습니다:

- 매일 아침 데이터베이스에서 정보를 가져와
- 이를 정리하고 분석한 다음
- 분석 결과를 보고서로 만들어
- 이메일로 팀원들에게 자동 발송하는 과정

이런 반복적인 작업을 수동으로 진행하면 시간도 많이 소요되고 실수할 가능성도 높습니다. 워크플로우 툴은 이런 과정을 코드로 작성하여 자동화함으로써 사람의 개입 없이도 정확하게 작업이 수행되도록 합니다.

데이터 분석 분야에서 주로 사용되는 워크플로우 툴로는 Apache Airflow, Prefect, Dagster 등이 있으며, 이들은 작업 실패 시 자동 재시도, 작업 간 의존성 관리, 스케줄링, 모니터링 등의 기능을 제공합니다.

## 1.3 데이터 분석 분야에서의 workflow tool

데이터 분석 및 엔지니어링 분야에서 워크플로우 관리 도구의 중요성은 지속적으로 대두되고 있습니다. 이는 데이터의 규모와 복잡성이 증가하면서 수동적인 작업 관리로는 한계가 있기 때문입니다.

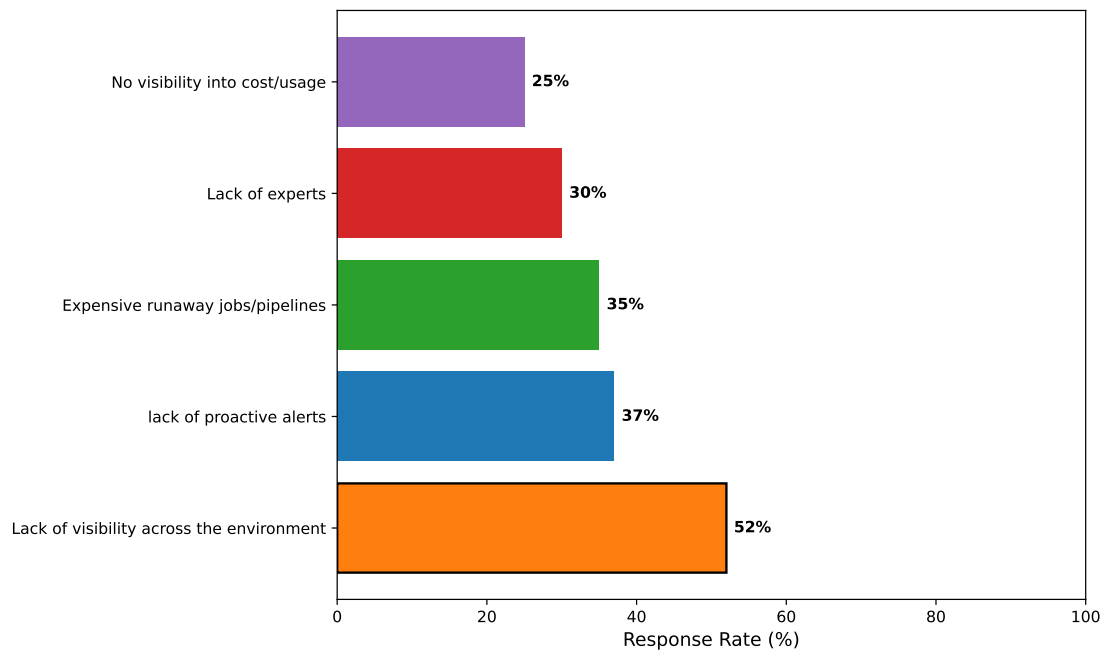


Figure 1.1: 데이터 팀이 직면한 문제 (Unravel 설문조사)

- [unravel 2022 dataops 설문조사](#)에 따르면, 데이터 팀이 가장 많이 겪는 문제는 환경 전체적인 가시성 부족입니다. 이는 데이터 파이프라인이 복잡해지면서 작업의 흐름을 파악하기 어려워지고, 오류 발생 시 대응이 늦어지는 문제를 의미합니다. Apache Airflow와 같은 워크플로우 관리 도구는 작업 자동화, 스케줄링, 모니터링을 통해 데이터 팀의 생산성을 향상시키고, 오류를 최소화하는 데 큰 도움을 줍니다. (최신 자료는 잘 안보이네요. 2022년 자료긴 하지만 뭐.. 이정도도 나쁘지 않죠)
- 또한 [Gartner의 2025 트렌드](#)에 따르면 데이터 분석 전략에서 주목해야 할 최신 트렌드로 **다양한 형태의 데이터를 통합 관리하는 시스템**과 **쉽게 활용 가능한 데이터 상품화**를 강조하고 있습니다. 이러한 트렌드를 실현하기 위해서는 데이터를 수집하고 처리하는 과정을 자동화하고 효율적으로 관리하는 것이 필수적인데, Apache Airflow와 같은 작업 자동화 도구가 바로 이런 필요성을 충족시키는 핵심 기술이 될 수 있습니다.

## 1.4 workflow 활용 사례

### 국내 기업

추가 예정

### 해외 기업

추가 예정

## 1.5 Workflow Tool 다운로드 통계

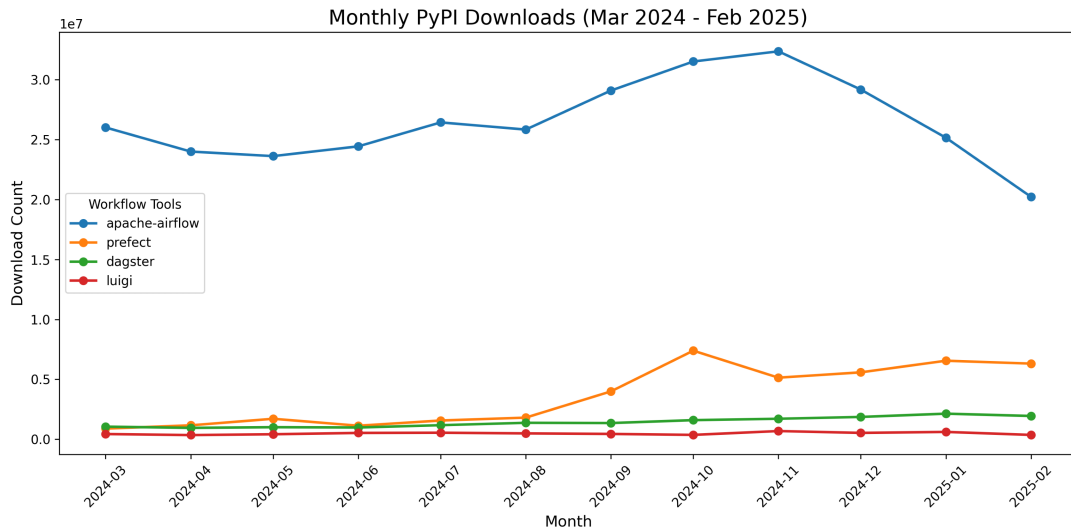


Figure 1.2: 2024.03~2025.02 데이터 관련 Workflow tool 다운로드 수 비교

- PyPI 다운로드 통계를 통해 경쟁 기술과 비교한 상대적인 Airflow의 사용 비율을 조사해 보았습니다.
- Airflow는 데이터 분석 workflow 툴 중에서도 가장 많은 다운로드 수를 기록하고 있습니다.
- 물론 다운로드가 반드시 실사용으로 이어진다고 보장할 수는 없지만 다른 tool에 비해 상대적으로 많은 관심을 받고 있다는 것을 알 수 있습니다.

## 1.6 Airflow github star 수

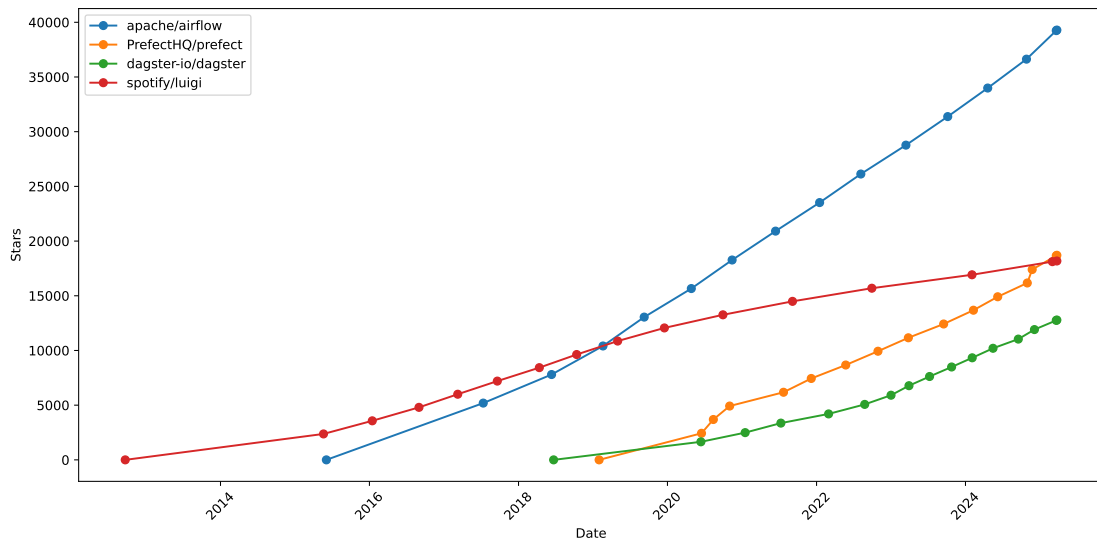


Figure 1.3: workflow tool github star 수 변화

- GitHub Star 수는 오픈소스 프로젝트의 인기도와 커뮤니티 활성화 정도를 보여주는 중요한 지표입니다.
- Apache Airflow는 2015년 출시 이후 꾸준히 성장하여 2025년까지 약 32,000개의 Star를 기록했습니다.
- 후발 주자인 Prefect와 Dagster도 성장세를 보이고 있으나, Airflow가 여전히 시장을 선도하고 있습니다.
- 특히 2021년 이후 Airflow의 Star 수 증가 속도가 더욱 가속화되는 추세를 보이고 있어, 워크플로우 도구 시장에서의 입지가 더욱 공고해지고 있습니다.

## 1.7 Airflow를 사용하는 직종 비율

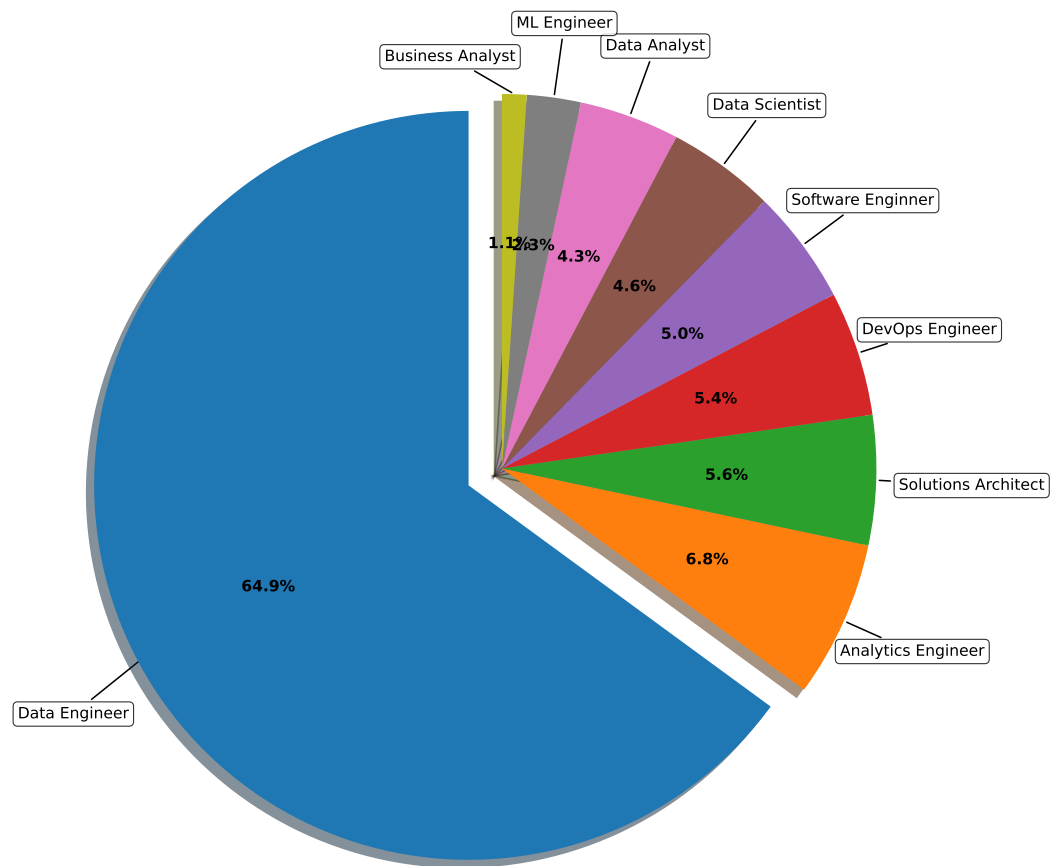


Figure 1.4: 2024 Airflow 이용자 중 직종별 비율

- [Airflow 공식 사이트 2024 설문조사 결과](#)를 참고해서 작성했습니다.
- airflow 기술을 요구하는 직종 중 **데이터 엔지니어** 직무에서 요구되는 비율이 높습니다.
- 생각보다 data engineer의 비율이 아주 높지는 않습니다. Airflow 기술을 익힌다면 데이터 분야의 다양한 직무에서 활동할 수 있는 기회가 높아질 것으로 기대됩니다.

## 1.8 Airflow 사용자들의 만족도

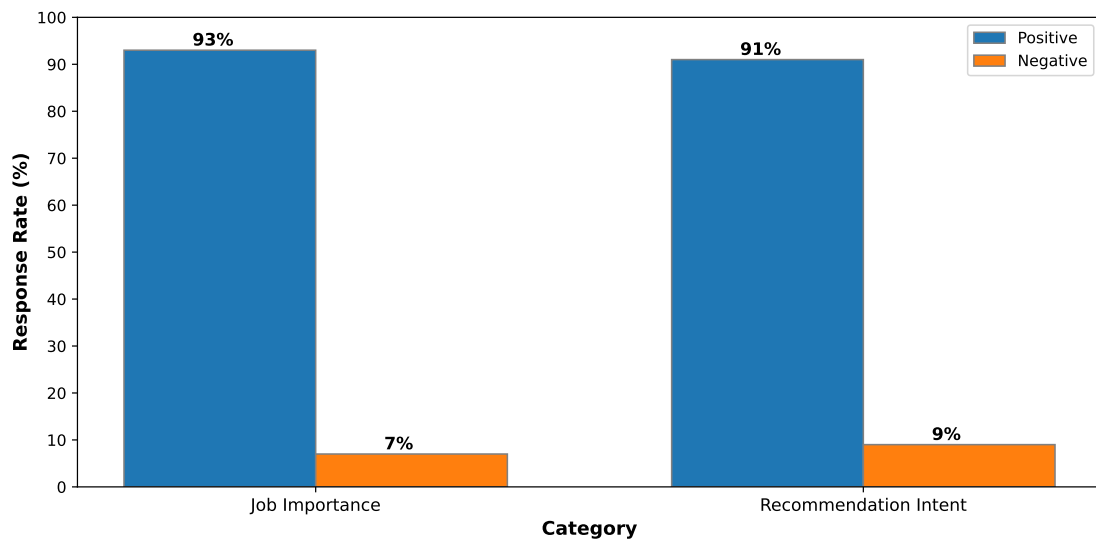


Figure 1.5: 2024 Airflow 사용자 만족도 조사 결과

- [Airflow 공식 사이트 2024 설문조사 결과](#)에 따르면, 93%의 사용자가 자신의 직무에서 Airflow가 중요하다고 답했습니다.
- 같은 조사에서 91%의 사용자가 Airflow를 다른 사람에게 추천할 의향이 있다고 답했습니다.
- 이처럼 높은 만족도는 Airflow가 실제 업무 환경에서 효과적으로 작동하고 있음을 보여줍니다.

## 1.9 채용 시장에서의 Airflow 수요

학생 입장에서 실제 기업에서 활용하고 있는 기술 현황에 대해 조사하기에는 채용 공고가 가장 직접적인 정보 제공원이라고 판단해서 이쪽으로 조사해봤습니다.

뭔가 대규모로 분석해보고 싶었는데 볼 수 있는 정보가 길어봤자 2주가 끝이더라고요. 그래서 가볍게 Job Description 예시 몇개 가져왔습니다.

### 주요 기업 JD 발췌 예시

## Data Engineer

[DataOps](#)[Infra](#)

[토스증권 소속 | 정규직](#)[지원하기](#)

### 합류하게 될 팀에 대해 알려드려요

- 토스증권 Data Engineer(Infra)는 Data Division내에 Data Infra Team에 속해 있어요.
- Data Infra Team은 크게 Hadoop Eco 기반의 빅데이터 인프라와 로그/검색 플랫폼(Elasticsearch)을 운영하고 Data 기술조직에서 데이터 입수, 각종 데이터 작업 등 다양하게 사용하는 Kubernetes에 대한 운영과 기술지원을 하고 있어요.

### Data Division을 소개합니다

- 토스증권 Data Division은 세계 최고로 데이터를 잘 다루는 증권사가 되기 위해 데이터 기술, 서비스 그리고 데이터 기반의 의사결정에 기여하고 있어요.
- 다양한 데이터 직군이 모여 밀접하게 협업하며 즐겁게 일하고 있어요.
- 또한 주기적으로 Tech Weekly를 진행하며 서로의 노하우를 공유하고 있어요. 본인의 흥미와 의지가 있다면 얼마든지 다른 직군의 업무와 노하우를 공유받을 수 있어요.





### 합류하면 함께 할 업무예요



- 토스증권의 Data Engineer(Platform)는 증권 서비스의 다양한 데이터를 효과적으로 저장하고 처리하기 위한 플랫폼을 운영하고 발전시키는 업무를 담당해요.
- Hadoop Ecosystem 기반의 데이터 플랫폼을 구성하여 운영중이며 사용하는 주요 컴포넌트로는 Hadoop, Spark, Impala, Hive, Kudu, Kafka, **Airflow**, Jenkins, k8s가 있어요.
- 토스증권 서비스에서 발생하는 국내/해외 중목, 주식매매 등 다양한 데이터를 가장 효율적이고 안전한 방법으로 처리하기 위한 플랫폼 아키텍처를 설계하고 구축해요.
- 증권사에 맞는 데이터 보안과 거버넌스를 지속적으로 강화해요.
- 도전적인 문제들을 해결하기 위해 새로운 환경을 고민하고 새로운 기술을 적극적으로 도입해요.

Figure 1.6: 토스 증권 Data Engineer JD



# MLOps Engineer Internship



 Platform & Infra  Magok, Seoul

## 팀 소개

Platform&Infra팀은 AI 모델의 개발부터 서비스 운영을 위한 배포에 이르기까지 AI 모델의 수명 주기를 최적화하고, 효율적으로 관리하기 위한 MLOps 파이프라인을 구축합니다. 또한 AI 서비스의 안정적인 운영 지원을 위한 보안성 강화, 인프라 관리 및 자원 최적화 업무를 수행합니다.

## 수행 업무

- AI 모델의 학습/추론 플랫폼을 설계하고 구축하며 운영합니다.
- AI/ML 플랫폼 운영 및 생산성 향상을 위한 다양한 서비스와 도구를 개발합니다.

## 지원자격

- Linux 및 CLI 환경을 다뤄본 경험이 있으신 분
- Python, Go 등 프로그래밍 언어를 활용한 웹 애플리케이션 개발을 해봤으면 좋아요.
- Docker 및 Kubernetes 같은 컨테이너 기술의 기본 개념을 이해하고 있으면 좋아요.

## 우대사항

- GCP, AWS, Azure 같은 Public Cloud 환경에서 개발을 해봤으면 좋아요.
- CI/CD 도구(Helm, Kustomize, ArgoCD 등)를 활용한 개발 및 운영을 해봤으면 좋아요.
- Triton, TensorRT 같은 AI 서빙 프레임워크를 사용해봤으면 좋아요.
- **Airflow** Kubeflow 같은 Workflow 툴을 사용해봤으면 좋아요.
- 기술적인 내용을 문서화하고 팀원들과 공유해봤으면 좋아요.

Figure 1.7: LG MLOps Engineer Internship

# Senior Software Engineer, Enterprise Data and Engineering

Hyderabad, Telangana, India
Mid

Apply

**Minimum qualifications:**

- Bachelor's degree or equivalent practical experience.
- 5 years of experience with software development in one or more programming languages, and with data structures/algorithms.
- 3 years of experience with ML/AI algorithms and tools, deep learning, or natural language processing or ML sub domain, including in Applied ML space.
- 3 years of experience testing, maintaining or launching software products, and 3 years of experience with software design (either distributed system design or ML design) and architecture.
- Experience in a leadership role (technical leadership or people management, supervision, or team leadership).

**Preferred qualifications:**

- Master's degree or PhD in Computer Science or a related technical field.
- 3 years of experience in a technical leadership or individual contributor role.
- Experience with ML frameworks, Applied ML across sub-domains.

**About the job**

Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. We're looking for engineers who bring fresh ideas from all areas, including information retrieval, distributed computing, large-scale system design, networking and data storage, security, artificial intelligence, natural language processing, UI design and mobile; the list goes on and is growing every day. As a software engineer, you will work on a specific project critical to Google's needs with opportunities to switch teams and projects as you and our fast-paced business grow and evolve. We need our engineers to be versatile, display leadership qualities and be enthusiastic to take on new problems across the full-stack as we continue to push technology forward.

In this role, you will deliver solutions to meet the data, reporting, and analytics needs of Googlers. You will drive high impact projects to deliver data management and investigative solutions for our partners across Google, create and maintain logical and physical database designs, and ensure the integrity of data under the purview of the projects, including establishing security procedures to protect and maintain the highest level of confidentiality and data security. In addition, you'll partner with internal teams to define and implement solutions that improve internal business processes, maintain highest levels of development practices, integrate third-party products with internal systems, and maintain highest levels of development practices, including technical design, solution development, systems configuration, test documentation/execution, issue identification and resolution, writing clean, modular and self-sustaining code.

At Corp Eng, we build world-leading business solutions that scale a more helpful Google for everyone. As Google's IT organization, we provide end-to-end solutions for organizations across Google. With an inclusive mindset, we deliver the right tools, platforms, and experiences for all Googlers as they create more helpful products and services for everyone. In the simplest terms, we are Google for Googlers.

**Responsibilities**

- Deliver data integration and pipeline solutions leveraging technologies such as Kafka, Spark, **Airflow** or similar technologies.
- Work well across Product Areas, build relationships, and deliver on shared objectives.

Figure 1.8: Google Senior Software Engineer, Enterprise Data and Engineering

국내 뿐만 아니라 해외 주요 기업에서도 Airflow를 활용한 데이터 파이프라인 구축 및 관리 역할에 수요가 있습니다. (그래봤자 3개만 발채했지만요)

이상의 내용들을 통해 데이터 분석 직무에서 workflow tool에 대한 중요성은 대두되고 있고, Airflow는 그 중에서도 높은 위상을 가지며, 현재 다양한 데이터 분석 직무에서 Airflow 기술을 활용하고 있음을 알 수 있습니다.

## 2. Airflow 개념 & Demo

### 2.1 Airflow 개념

#### 동작 원리

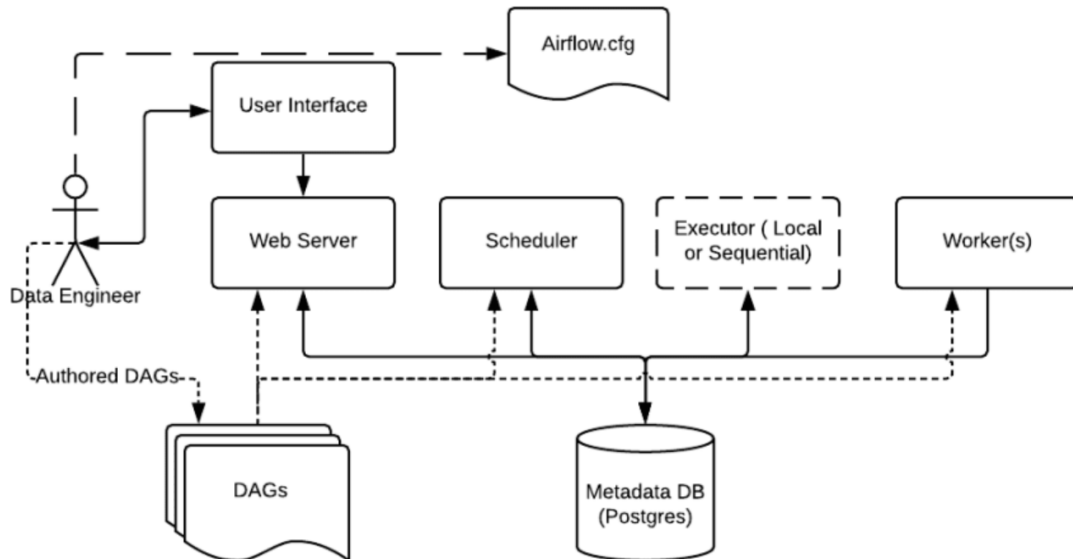


Figure 2.1: 동작 원리

- Web server : Airflow의 웹 UI 서버 입니다.
- Scheduler : 모든 DAG와 Task에 대하여 모니터링 및 관리하고, 실행해야할 Task를 스케줄링 해줍니다.
- DAG : Directed Acyclic Graph로 개발자가 Python으로 작성한 워크플로우 입니다. Task들의 dependency를 정의합니다.
- Worker : 실제 Task를 실행하는 주체입니다.
- Operator: Task를 정의하는 객체입니다. Airflow 내장 및 커뮤니티 정의 Operator를 사용하여 다양한 작업을 수행할 수 있습니다.
- Connection: 외부 시스템과의 연결을 정의합니다. 예를 들어, 데이터베이스, 클라우드 서비스 등과의 연결을 설정할 수 있습니다.
- Hook: Operator와 Connection을 연결하는 객체입니다. Hook을 사용하여 외부 시스템과의 상호작용을 쉽게 처리할 수 있습니다.

## 2.2 Airflow Demo

### 개요

Apache Airflow를 활용해 데이터 파이프라인을 구축해보겠습니다. 아래와 같은 흐름으로 진행될 예정입니다.

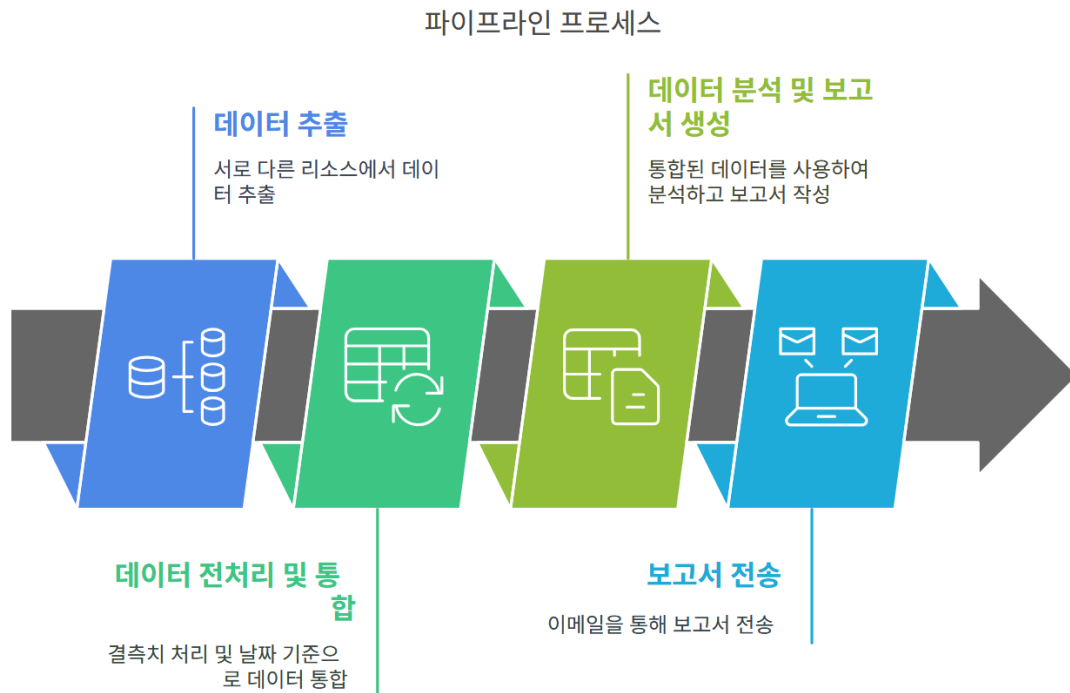


Figure 2.2: 파이프라인 프로세스

데이터베이스에는 각각 아래와 같은 demo 데이터를 저장했습니다.

---

**Listing 2.1** init-mysql1.sql

---

```
CREATE TABLE shopping_data (  
    amount FLOAT,  
    date DATE  
);  
  
INSERT INTO shopping_data (amount, date) VALUES  
    (300.00, '2025-04-01'),  
    (280.00, '2025-04-02'),  
    (290.00, '2025-04-03'),  
    (200.00, '2025-04-04'),  
    (null, '2025-04-05'),  
    (130.00, '2025-04-06'),  
    (140.00, '2025-04-07'),  
    (160.00, '2025-04-08'),  
    (270.00, '2025-04-09'),  
    (180.00, '2025-04-10'),  
    (120.00, '2025-04-11'),  
    (70.00, '2025-04-12'),  
    (60.00, '2025-04-13'),  
    (65.00, '2025-04-14'),  
    (null, '2025-04-15'),  
    (110.00, '2025-04-16'),  
    (250.00, '2025-04-17'),  
    (260.00, '2025-04-18'),  
    (255.00, '2025-04-19'),  
    (220.00, '2025-04-20');
```

---

---

**Listing 2.2** init-mysql2.sql

---

```
CREATE TABLE factor_data (  
    date DATE,  
    staff_count INT,  
    operating_hours FLOAT,  
    product_variety INT,  
    event_frequency INT,  
    store_cleanliness INT,  
    training_hours FLOAT  
);  
  
INSERT INTO factor_data (date, staff_count, operating_hours, \  
    product_variety, event_frequency, store_cleanliness, training_hours) VALUES  
    ('2025-04-01', 10, 12.0, 50, 2, 8, 5.0),  
    ('2025-04-02', 9, 11.5, 45, 1, 7, 4.5),  
    ('2025-04-03', 10, 12.0, 48, 2, 8, 5.0),  
    ('2025-04-04', 7, 10.0, 40, 1, 6, 3.0),  
    ('2025-04-05', 5, 9.0, 35, 0, 5, 2.0),  
    ('2025-04-06', 4, 8.5, 30, 0, 4, 1.5),  
    ('2025-04-07', 5, 9.0, 35, 1, 5, 2.0),  
    ('2025-04-08', 6, 10.0, 38, 1, 6, 3.0),  
    ('2025-04-09', 9, 11.5, 45, 2, 7, 4.0),  
    ('2025-04-10', 7, 10.5, 40, 1, 6, 3.5),  
    ('2025-04-11', 4, 8.0, 30, 0, 4, 1.0),  
    ('2025-04-12', 3, 7.5, 25, 0, 3, 0.5),  
    ('2025-04-13', 3, 7.0, 20, 0, 3, 0.5),  
    ('2025-04-14', 4, 8.0, 28, 0, 4, 1.0),  
    ('2025-04-15', 5, 9.0, 35, 1, 5, 2.0),  
    ('2025-04-16', 6, 9.5, 38, 1, 6, 2.5),  
    ('2025-04-17', 8, 11.0, 42, 2, 7, 4.0),  
    ('2025-04-18', 9, 11.5, 45, 2, 8, 4.5),  
    ('2025-04-19', 8, 11.0, 43, 2, 7, 4.0),  
    ('2025-04-20', 7, 10.5, 40, 1, 6, 3.5),  
    ('2025-04-21', 6, 10.0, 38, 1, 5, 3.0),  
    ('2025-04-22', 5, 9.0, 35, 1, 5, 2.0),  
    ('2025-04-23', 6, 9.5, 38, 1, 6, 2.5),  
    ('2025-04-24', 7, 10.0, 40, 1, 6, 3.0),  
    ('2025-04-25', 4, 8.0, 30, 0, 4, 1.0),  
    ('2025-04-26', 3, 7.5, 25, 0, 3, 0.5),  
    ('2025-04-27', 4, 8.0, 28, 0, 4, 1.0),  
    ('2025-04-28', 5, 9.0, 35, 1, 5, 2.0),  
    ('2025-04-29', 8, 11.0, 42, 2, 7, 4.0),  
    ('2025-04-30', 9, 11.5, 45, 2, 8, 4.5);
```

---

매 달 자동으로 위의 데이터를 불러와 판매량과 상관관계가 높은 요인을 분석하고, 그 결과를 이메일로 전송하는 파이프라인을 구축해보겠습니다.

## Airflow 설치

먼저 airflow를 설치합니다. 아래의 명령어로 간단하게 세팅할 수 있습니다.

```
curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.5.1/docker-compose.yaml'
```

그러면 airflow 세팅에 필요한 설정 파일이 자동으로 다운로드 됩니다.

저는 여기서 이번 demo를 위해 몇가지 설정을 추가했습니다.

---

### Listing 2.3 docker-compose.yml

```
AIRFLOW__EMAIL__EMAIL_BACKEND: airflow.utils.email.send_email_smtp
AIRFLOW__SMTP__SMTP_HOST: smtp.gmail.com
AIRFLOW__SMTP__SMTP_PORT: ${SMTP_PORT:-587}
AIRFLOW__SMTP__SMTP_USER: ${SMTP_USER}
AIRFLOW__SMTP__SMTP_PASSWORD: ${SMTP_PASSWORD}
AIRFLOW__SMTP__SMTP_MAIL_FROM: ${SMTP_FROM_MAIL}
```

---

먼저 메일 전송을 위한 설정을 추가해줍니다.

database 역할을 해줄 환경 두 개도 설정했습니다.

---

**Listing 2.4** docker-compose.yml

---

```
services:
  mysql1:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD:-root}
      MYSQL_DATABASE: ${SHOPPING_DB_NAME:-shopping_db}
      MYSQL_USER: ${MYSQL_USER:-user}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD:-secret}
    volumes:
      - ./mysql1-data:/var/lib/mysql
      - ./init-mysql1.sql:/docker-entrypoint-initdb.d/init.sql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", \
        "-prootpassword"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 10s

  mysql2:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD:-root}
      MYSQL_DATABASE: ${WEATHER_DB_NAME:-weather}
      MYSQL_USER: ${MYSQL_USER:-user}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD:-secret}
    volumes:
      - ./mysql2-data:/var/lib/mysql
      - ./init-mysql2.sql:/docker-entrypoint-initdb.d/init.sql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", \
        "-prootpassword"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 10s
```

---



이제 아래의 명령어를 실행한 후 localhost:8080으로 접속하면 Airflow UI에 접속할 수 있습니다.

```
docker compose up -d
```

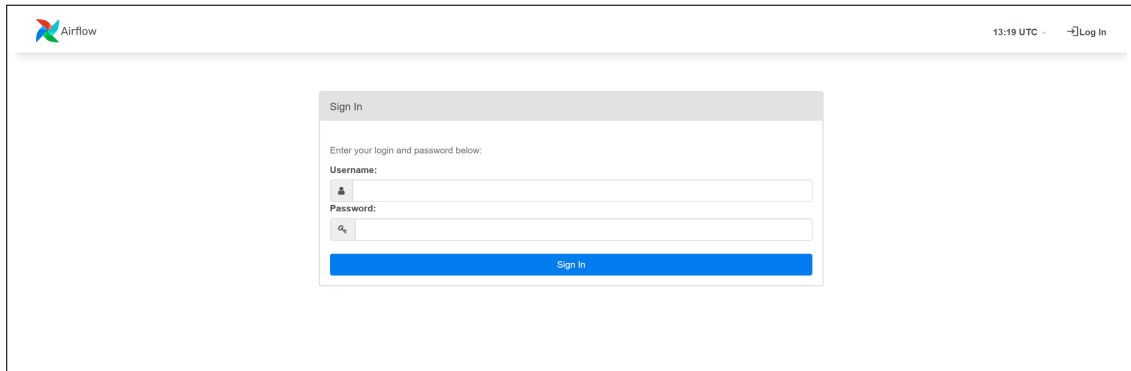


Figure 2.3: airflow 접속 화면

초기 id 와 password는 airflow로 설정되어 있습니다.

## DAG 설정

아래와 같이 파일을 작성해서 dags 폴더에 저장합니다.

### Listing 2.5 dags/demo.py

```
from airflow import DAG
from datetime import datetime
from airflow.operators.python import PythonOperator
import pandas as pd
from airflow.providers.mysql.hooks.mysql import MySqlHook
from airflow.operators.email import EmailOperator

with DAG(
    dag_id='integrate_shopping_factor_data',
    schedule='@monthly',
    start_date=datetime(2024, 4, 4),
    catchup=False,
) as dag:
```

---

**Listing 2.6** dags/demo.py

---

```
def extract_shopping_data():
    hook = MySqlHook(mysql_conn_id='shopping_source')
    conn = hook.get_sqlalchemy_engine().connect()
    df = pd.read_sql("SELECT * FROM shopping_data", conn)
    df.to_csv('/tmp/shopping_data.csv', index=False)

def extract_factor_data():
    hook = MySqlHook(mysql_conn_id='factor_source')
    conn = hook.get_sqlalchemy_engine().connect()
    df = pd.read_sql("SELECT * FROM factor_data", conn)
    df.to_csv('/tmp/factor_data.csv', index=False)

def preprocess_and_integrate():
    shopping_df = pd.read_csv('/tmp/shopping_data.csv')
    factor_df = pd.read_csv('/tmp/factor_data.csv')
    shopping_df['amount'] = shopping_df['amount'].fillna(\
                                                shopping_df['amount'].mean())
    integrated_df = pd.merge(shopping_df, factor_df, on='date', how='inner')
    integrated_df.to_csv('/tmp/integrated_data.csv', index=False)
    return "Data preprocessed and integrated"

extract_shopping_data_task = PythonOperator(
    task_id='extract_shopping_data',
    python_callable=extract_shopping_data,
)

extract_factor_data_task = PythonOperator(
    task_id='extract_factor_data',
    python_callable=extract_factor_data,
)

process_task = PythonOperator(
    task_id='preprocess_and_integrate',
    python_callable=preprocess_and_integrate,
)
```

---

**Listing 2.7** dags/demo.py

---

```
report_task = PythonOperator(
    task_id='generate_report',
    python_callable=generate_report,
    provide_context=True,
)

email_task = EmailOperator(
    task_id='send_email_report',
    to='cryscha123@naver.com',
    subject='Monthly Sales Insights Report',
    html_content="{ ti.xcom_pull(task_ids='generate_report', \
                                key='html_report') }",
)
```

---

**Listing 2.8** dags/demo.py

---

```
[extract_shopping_data_task, extract_factor_data_task] >> process_task \
>> report_task >> email_task
```

---

## Connections 설정

Airflow UI에서 Connections 메뉴를 통해 데이터 소스를 위한 Connection을 설정합니다.

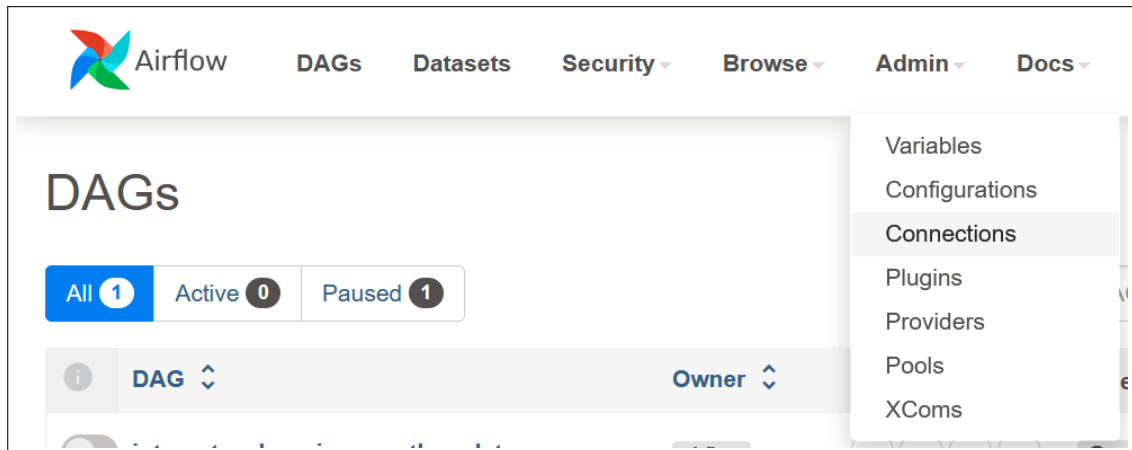


Figure 2.4: 상단 메뉴의 Connections를 눌러 줍니다.

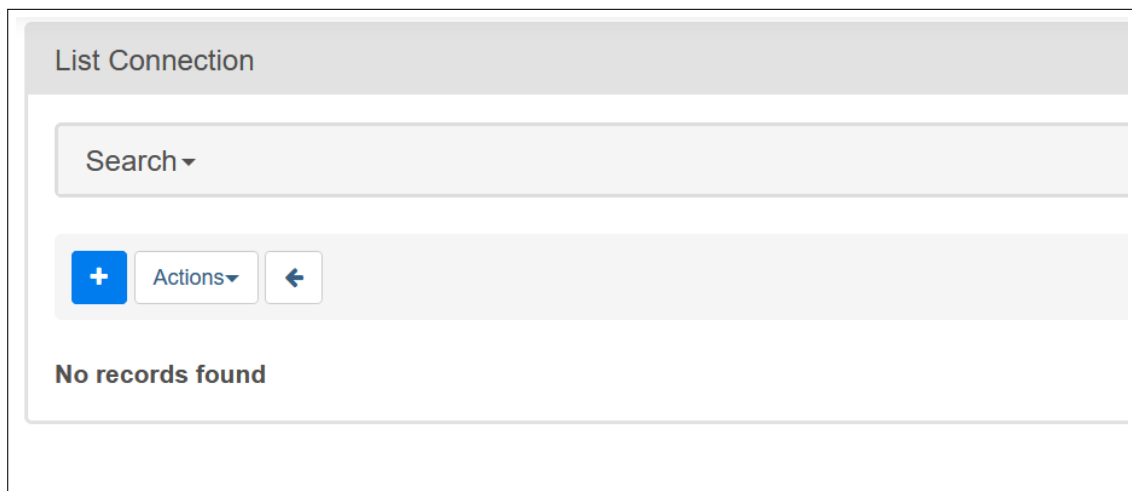


Figure 2.5: + 기호를 선택



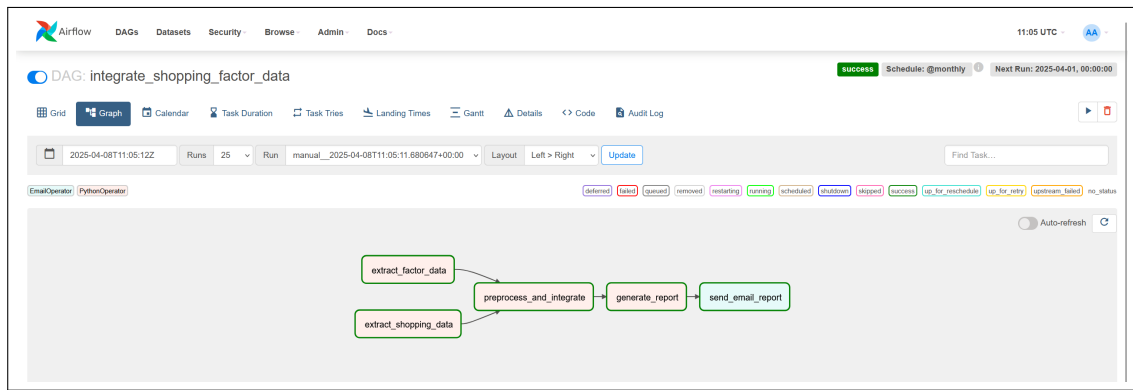


Figure 2.9: 결과화면에서 task간 의존 관계와 결과를 확인할 수 있습니다.

## 결과

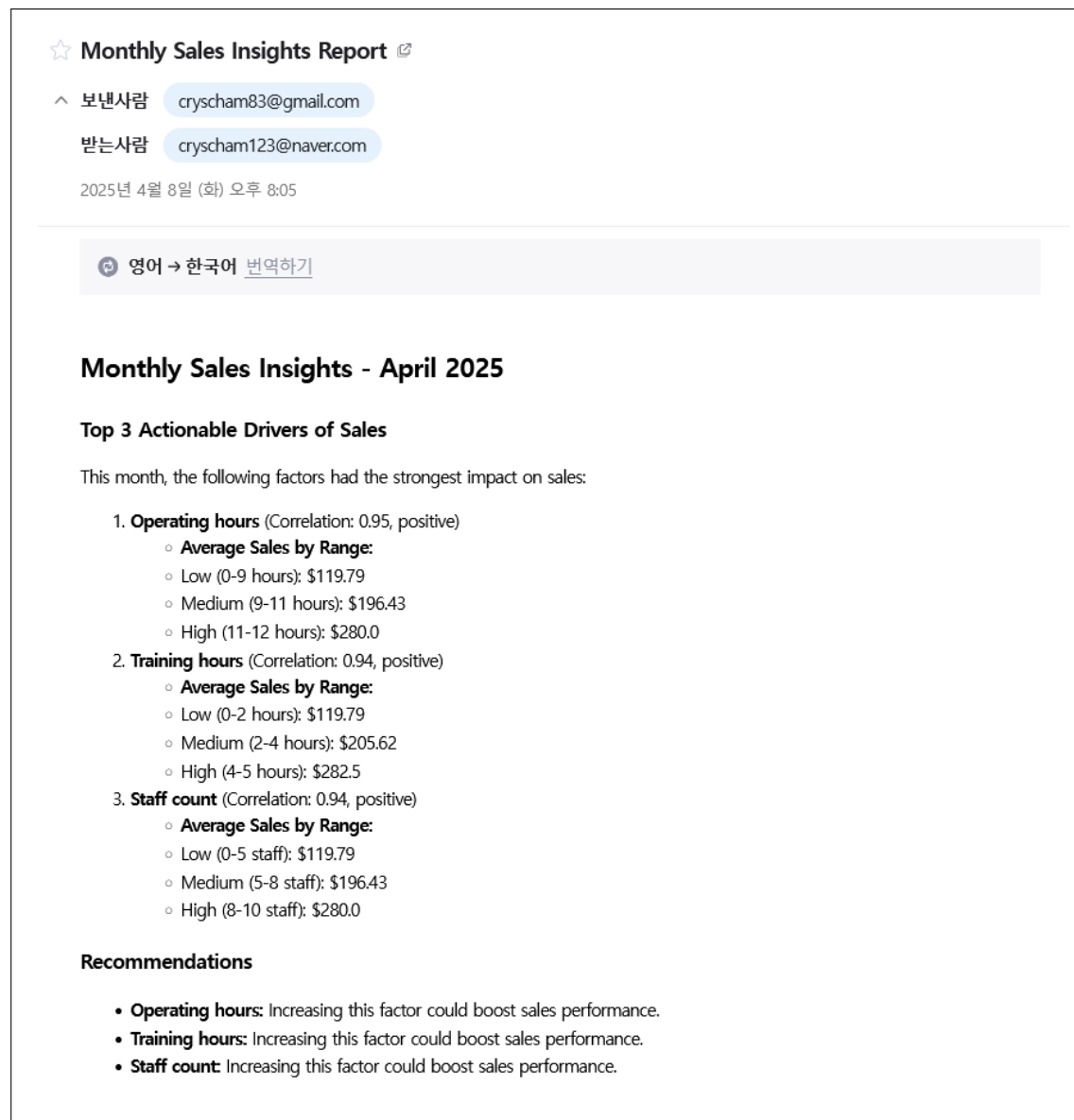


Figure 2.10: 메일도 정상적으로 확인해볼 수 있습니다.