▣ **강의명:** CSCI E-103: 재현 가능한 머신러닝

▣ **주차:** Lecture 13

▣ **교수명:** Anindita Mahapatra

Eric Gieseke

▣ **목적:** Lecture 13의 핵심 개념 학습

Contents

Lecture 13: 데이터 이니셔티브 가치 극대화와 지속적 개선

(CI/CD, IaC, Observability, and Final Project)

개요 (Overview)

이번 강의는 데이터 파이프라인을 구축한 이후, 이를 **어떻게 안정적으로 운영하고 자동화하며 품질을 유지할 것인가**에 초점을 맞춥니다. 단순히 "돌아가는 코드"를 만드는 것을 넘어, 기업 환경에서 실제 가치를 창출하기 위한 엔지니어링 관행(DevOps)을 데이터 영역(DataOps/MLOps)에 적용하는 방법을 배웁니다.

□ 핵심 요약

- **기말 과제(Final Project):** 팀 구성 및 역할, 발표 자료(20장 내외) 작성 요령 안내.
- **SDLC (소프트웨어 개발 수명 주기):** 개발(Dev) → 스테이징(Stage) → 운영(Prod) 단계별 전략.
- **인프라 자동화 (IaC):** 테라폼(Terraform) 등을 활용해 인프라를 코드로 관리하는 법.
- **CI/CD 관측성(Observability):** 자동화된 테스트/배포 파이프라인과 데이터 품질/비용 모니터링.
- **Databricks Asset Bundles (DAB):** 프로젝트 자산을 패키징하여 배포하는 최신 표준.

1 기말 과제 (Final Project) 가이드

학기말 프로젝트는 "Lakehouse 아키텍처 구축"을 목표로 하며, 팀 단위로 진행됩니다. 실제 현업의 데이터 프로젝트 제안 및 구축 과정을 모사합니다.

1.1 발표 및 제출 요구사항

- **분량:** 슬라이드 최대 20장.
- **시간:** 발표 15분 + 질의응답(Q&A) 5분.
- **일정:** 마지막 강의(12/15 예정) 시간에 팀별 발표 진행 (타임존 이슈가 있는 팀 우선 배정 가능).
- **참여:** 모든 팀원이 발표에 참여해야 함 (슬라이드 넘기는 역할 포함).

1.2 팀 구성 및 역할 (R&R)

팀당 2명씩 짝을 이루어 아래의 핵심 역할을 분담하고 발표에 포함해야 합니다.

Table 1: 기말 프로젝트 팀원별 역할 정의

역할 (Role)	담당 업무 (Responsibilities)	발표 포인트
데이터 아키텍트	전체 설계 및 아키텍처	확장성, 품질, 성능, 신뢰성, 거버넌스 전략 설명
데이터 엔지니어	데이터 파이프라인 구축	데이터 수집(Ingestion), 변환, 조인, 집계 과정 시연
ML 전문가	모델 개발 및 관리	모델 훈련, 추론(Inference), 모델 수명 주기 관리
데이터 분석가	BI 대시보드 및 인사이트	쿼리 작성, 대시보드 시각화, 알림 설정, 비즈니스 인사이트 도출

[프로젝트 필수 포함 사항] 발표에는 반드시 **"문제 정의(Problem Statement)"**, **"팀 소개(역할별)"**, **"아키텍처 리뷰"**, **"인사이트 요약"**, 그리고 **"향후 개선 계획(Next Steps)"**이 포함되어야 합니다. 팀 이름(가상의 컨설팅 회사명 등)을 정하는 것도 잊지 마세요.

2 데이터 프로젝트의 개발 수명 주기 (SDLC)

데이터 파이프라인을 만들 때, 로컬에서 대충 코드를 짜고 바로 운영 서버에 올리는 것은 매우 위험합니다. 안정적인 서비스를 위해 3단계 환경을 거칩니다.

[비유: 요리의 3단계]

- **Dev (개발):** 집 부엌에서 혼자 새로운 레시피를 실험해보는 단계. 재료를 조금만 씀.
- **Stage (스테이징):** 친구들을 불러 시식회를 하는 단계. 실제 손님상과 비슷하게 차려놓고 문제 없는지 확인.
- **Prod (운영):** 실제 레스토랑에서 손님에게 돈 받고 파는 단계. 실수가 용납되지 않음.

2.1 환경별 특징 및 전략

Table 2: *SDLC 3*단계 상세 비교

구분	1. 개발 (Development)	2. 스테이징 (Staging)	3. 운영 (Production)
목표	비즈니스 로직 구현 및 단위 테스트	운영 환경 모사 및 통합 테스트	실제 서비스 제공 및 안정성 유지
데이터	가짜 데이터(Canned data) 또는 소량 샘플	운영 데이터와 유사한 대용량 데이터	실제 실시간/배치 데이터 전체
리소스	단일 노드 클러스터, Spot 인스턴스 (저비용)	성능 테스트를 위한 확장된 리소스	고가용성 정책, 안정적인 인스턴스
권한	개발자 개인 계정(User) 사용	서비스 주체(Service Principal) 도입 시작	**오직 서비스 주체(Service Principal)만 사용**
상태	상호작용(Interactive) 중심	자동화(Automated)로 전환	**완전 자동화 (Fully Automated)**

2.2 서비스 수준 계약 (SLA) 및 지표

운영 단계에서는 시스템이 얼마나 잘 돌아가는지 약속(SLA)하고 측정해야 합니다.

- **가용성(Availability):** 시스템이 정상 작동하는 시간 비율. (예: 99.999% = 연간 다운타임 5분 미만)
- **재해 복구(Disaster Recovery):**
  - **RTO (Recovery Time Objective):** 사고 후 복구까지 걸리는 시간 (얼마나 빨리 고치나?)
  - **RPO (Recovery Point Objective):** 데이터 백업 주기 (과거 데이터를 어디까지 잃어도 되나?)

3 인프라 자동화(IaC)와 CI/CD

수백 개의 워크스페이스와 클러스터를 사람이 일일이 클릭해서 만들면 실수하기 쉽고 관리가 불가능합니다. 이를 코드로 관리하는 것이 **IaC(Infrastructure as Code)** 입니다.

3.1 IaC: 코드로 인프라 관리하기

[비유: 건물 설계도] 웹 콘솔에서 클릭으로 서버를 만드는 건 '손으로 흙을 빚어 집을 짓는 것'과 같습니다. 똑같은 집을 또 지으려면 처음부터 다시 빚어야 합니다. **IaC**는 '3D 프린터 설계도'를 만드는 것입니다. 설계도(코드)만 있으면 버튼 하나로 똑같은 집을 100채든 1,000채든 완벽하게 지을 수 있습니다.

3.1.1 프로비저닝 vs 구성 관리

- **프로비저닝 (Provisioning):** 인프라 자체를 생성 (예: Terraform, CloudFormation).
  - 특징: 불변(Immutable). 변경이 필요하면 기존 것을 부수고 새로 짓는 방식을 선호.
  - *Databricks* 활용: 워크스페이스 생성, 네트워크 설정 등 큰 틀을 잡을 때 주로 사용.
- **구성 관리 (Config Management):** 이미 있는 인프라 내부의 소프트웨어를 관리 (예: Ansible, Puppet).
  - 특징: 가변(Mutable). 기존 서버에 들어가서 라이브러리 버전을 업데이트하는 방식.

3.2 CI/CD (지속적 통합/지속적 배포)

코드 변경 사항이 자동으로 테스트되고 배포되는 파이프라인입니다.

1. **CI (통합):** 개발자가 Git에 코드를 'commit'하면 자동으로 빌드하고 유닛 테스트를 수행.
2. **CD (배포):** 테스트를 통과하면 스테이징/운영 환경으로 자동 배포(Deploy).
3. **이점:** 배포 주기가 빨라지고(분기별 → 매일), 버그를 조기에 발견.

3.3 데이터 메쉬(Data Mesh)와 워크스페이스

조직이 커지면 중앙 팀이 모든 데이터를 관리할 수 없습니다. **데이터 메쉬**는 데이터를 '제품(Product)'으로 취급하고, 도메인(마케팅, 영업 등)별로 소유권을 분산시킵니다.

- **워크스페이스 분리:** 각 도메인/프로젝트별로 별도 워크스페이스를 할당하여 격리(Isolation) 확보.
- **Unity Catalog:** 분산된 도메인 데이터를 하나로 묶어주는 중앙 거버넌스 역할.

4 관측성(Observability)과 데이터 품질 관리

시스템이 "잘 돌아가는지" 확인하는 것을 넘어, "무엇이 잘못되었고 왜 그런지"를 파악하는 능력입니다.

4.1 비용 및 사용량 모니터링 (System Tables)

Databricks의 'system' 카탈로그를 통해 모든 과금 및 사용 정보를 SQL로 조회할 수 있습니다.

- **Cost Management:** 누가(어떤 태그가) 돈을 얼마나 썼는지 확인 → 예산 초과 방지.
- **Audit Logs:** 누가 언제 어떤 데이터에 접근했는지 감사 추적.
- **Cluster Usage:** 클러스터가 놀고 있는데 켜져 있는지(유휴 상태) 확인.

4.2 데이터 품질 모니터링

데이터 팀은 보통 "데이터를 제때 주는 것(Delivery SLA)"에 집중하지만, 사용자는 "데이터가 정확한 것(Quality)"을 기대합니다. 이를 자동화해야 합니다.

Table 3: 데이터 품질 모니터링 도구 비교

기능	설명	특징
Anomaly Detection	AI가 과거 패턴을 학습하여 이상 징후 감지.	설정이 매우 쉽음(버튼 클릭). 신선도(Freshness)와 완전성(Completeness) 체크.
Lakehouse Monitoring	테이블의 통계적 프로파일(분포, Null 비율 등) 및 드리프트(변화) 감지.	대시보드 자동 생성. 데이터 내용의 변화(Drift)를 상세히 추적 가능.
DQX (Data Quality X)	코드로 정의하는 데이터 품질 프레임워크.	파이프라인 코드 내에서 'age > 18' 같은 구체적 규칙(Rule) 적용 가능.

[비유: 건강 검진]

- **Anomaly Detection:** 스마트워치가 "평소보다 심박수가 높아요"라고 알려주는 것 (자동, 패턴 기반).
- **Lakehouse Monitoring:** 정기 건강검진 결과표(혈압, 콜레스테롤 수치 등)를 받아보는 것 (상세 통계).
- **DQX:** 의사가 "술은 주 1회 이하로"라고 처방을 내리고 지키는지 체크하는 것 (명시적 규칙).

5 Databricks Asset Bundles (DAB)

DAB는 데이터 프로젝트(코드, 잡 설정, 파이프라인 등)를 **하나의 패키지로 묶어** 개발, 스테이징, 운영 환경으로 손쉽게 배포하는 최신 도구입니다.

5.1 왜 DAB를 쓰는가?

기존에는 API를 일일이 호출하거나 수동으로 옮겨야 했지만, DAB를 쓰면 'bundle deploy' 명령어 하나로 프로젝트 전체를 다른 환경에 똑같이 복제할 수 있습니다.

5.2 DAB의 핵심 구성 요소

- **YAML 설정 파일 ('databricks.yml'):** 프로젝트의 모든 설정(어떤 잡을 실행할지, 어떤 클러스터를 쓸지)이 정의된 설계도.
- **Target (타겟):** 배포할 목적지 (Dev, Stage, Prod). 타겟별로 설정을 다르게(Override) 할 수 있음.
- **CLI 명령어:** 'databricks bundle validate', 'databricks bundle deploy', 'databricks bundle run'.

5.3 DAB 워크플로우 예시 (CLI)

```
# 1. 번들유효성검사   (YAML 문법등확인   )
databricks bundle validate

# 2. 개발(dev) 환경에배포
databricks bundle deploy -t development

# 3. 개발환경에서잡실행테스트     ()
databricks bundle run -t development <job_key>

# 4. 테스트( 통과후 ) 운영(prod) 환경에배포
databricks bundle deploy -t production
```

Listing 1: DAB를 활용한 배포 및 실행 절차

부록: 핵심 용어 정리 (Glossary)

용어	원어	쉬운 설명
서비스 주체	Service Principal	사람이 아닌 '로봇 ID'. 자동화 작업이나 파이프라인 실행 시 사용하는 계정.
스팟 인스턴스	Spot Instance	클라우드 남는 자원을 싸게 쓰는 것. 회수될 수 있어 중요 작업엔 비추천.
데이터 메쉬	Data Mesh	데이터를 중앙에서 독점하지 않고, 각 부서(도메인)가 직접 관리/제공하는 방식.
드리프트	Drift	데이터의 통계적 속성이 과거와 달라지는 현상 (예: 갑자기 Null이 늘어남).
테라폼	Terraform	인프라 생성 도구. 코드로 서버, 네트워크 등을 정의하고 생성함.
단위 테스트	Unit Test	코드의 가장 작은 단위(함수 등)가 의도대로 작동하는지 검사하는 것.
통합 테스트	Integration Test	여러 모듈을 합쳤을 때 서로 잘 연결되어 작동하는지 검사하는 것.

부록: 학습 및 프로젝트 체크리스트

기말 프로젝트 및 시험 대비 점검

- ☐ **프로젝트 준비:** 팀원 컨택 및 역할 분담(아키텍트, 엔지니어, ML, 분석가) 완료했는가?
- ☐ **발표 자료:** 문제 정의, 아키텍처, 파이프라인 시연, 인사이트, 향후 계획이 모두 포함되었는가?
- ☐ **개념 이해:** SDLC 3단계(Dev/Stage/Prod)의 데이터/리소스/권한 차이를 설명할 수 있는가?
- ☐ **자동화:** IaC(Terraform)와 DAB(Asset Bundle)의 차이(인프라 vs 프로젝트)를 구분할 수 있는가?
- ☐ **모니터링:** System Table로 비용을 추적하고, Lakehouse Monitoring으로 데이터 품질을 감시하는 방법을 아는가?
- ☐ **CI/CD:** 코드가 변경되었을 때 배포까지의 자동화 흐름을 그릴 수 있는가?