Machine Learning Operation(MLOps)

- Ch1. Introduction to MLOps(2)

Introduction to MLOps(2)

1. MLOps Process Level
2. Docker and Kubernetes

MLOps Process Level

MLOps 성숙도 단계

Level0

Build and deploy Manually

- Manual script-driven
- Interactive process
- Disconnect between ML and Ops
- No CI/CD
- Infrequent monitoring

Level1

Automate the training phase

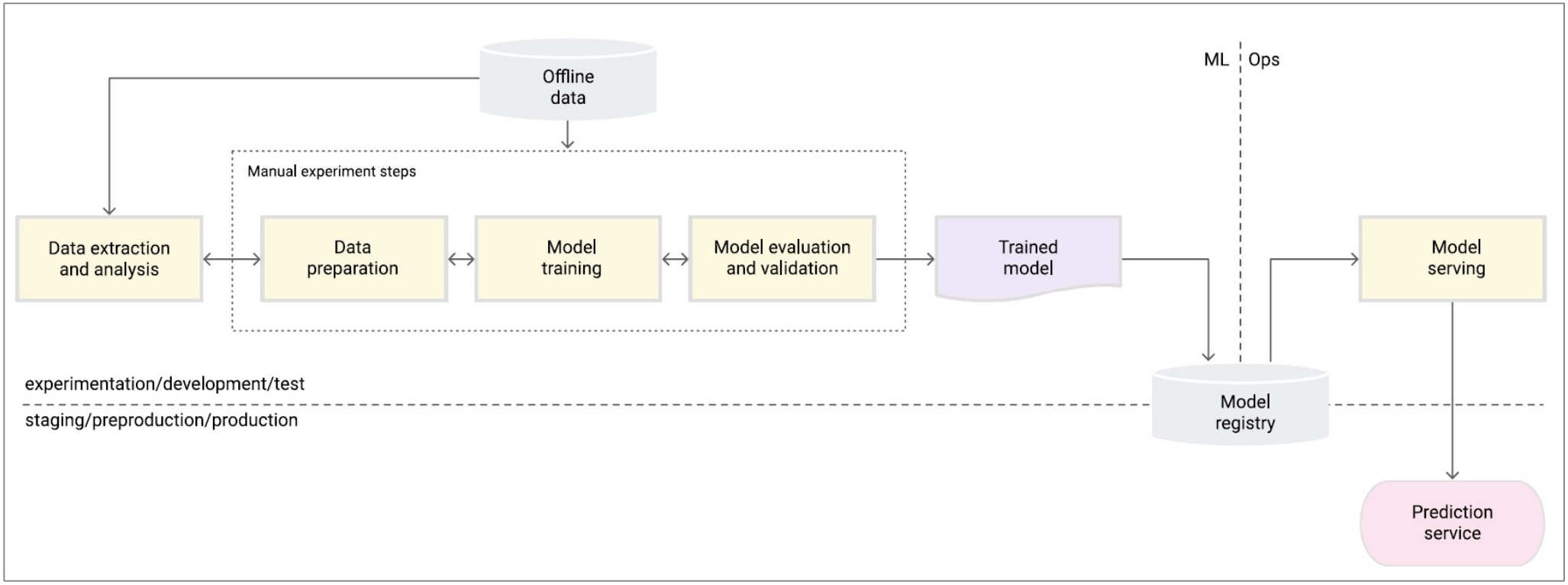
- Rapid experiment
- Continuous training in production
- Experimental operational symmetry
- Modularized code for code and pipeline
- Continuous deployment of model pipeline

Level2

Automate training, validation, and deployment

- Source control
- Service test and build
- Deployment service
- Model registry
- Feature store
- ML metadata store
- ML pipeline orchestrator

Level 0 MLOps



[그림] 모델을 예측 서비스로 제공하기 위한 수동 ML 단계

Level 0 MLOps

- Manual script-driven, Interactive process:
  데이터 분석, 데이터 준비, 모델 학습, 모델 검증을 포함한 모든 단계가 수동.
  각 단계를 수동으로 실행하고, 한 단계에서 다른 단계로 수동 전환.
  이 프로세스는 일반적으로 작업 가능한 모델이 생성될 때까지 데이터 과학자가 노트북에서 작성하고 실행하는 실험 코드에 의해 구성됨

- Disconnect between ML and Ops:
  모델을 만드는 데이터 과학자와 모델을 예측 서비스로 제공하는 엔지니어를 분리함
  데이터 과학자는 엔지니어링팀에 학습된 모델을 아티팩트로 전달.(
  이 전달에는 학습된 모델을 스토리지 위치에 배치하거나, 모델 객체를 코드 저장소에 확인하거나, 모델 레지스트리에 업로드하는 작업이 포함될 수 있습니다. 그런 다음 모델을 배포하는 엔지니어는 짧은 지연 시간을 제공하기 위해 필요한 기능을 프로덕션 단계에서 사용할수 있도록 해야 합니다. 이를 통해 학습-서빙 편향 이발생할 수 있습니다. (https://developers.google.com/machine-learning/guides/rules-of-ml/?hl=ko#training-serving_skew)

- Infrequent release iterations:
  데이터 프로세스는 데이터 과학팀이 모델 구현을 변경하거나 새 데이터로 모델을 재학습시키는 등 자주 변경되지 않는 몇 가지 모델을 관리한다고 가정합니다. 새 모델 버전은 1년에 두어 번만 배포됩니다.

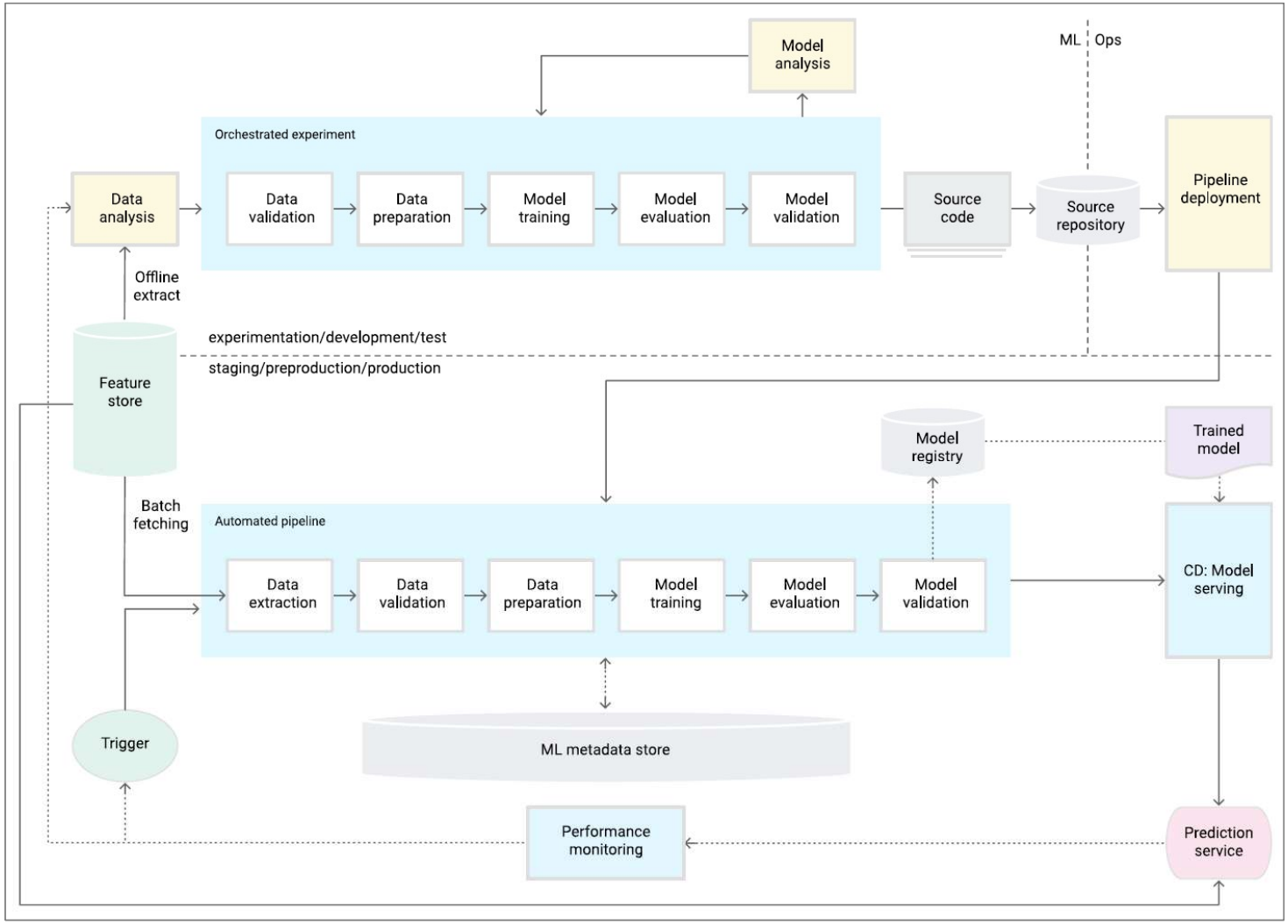
- No CI:
  구현 변경사항이 거의 없으므로 CI가 무시됩니다. 일반적으로 코드 테스트는 노트북 또는 스크립트 실행의 일부입니다. 실험 단계를 구현하는 스크립트와 노트북은 소스로 제어되며 학습된 모델, 평가 측정항목, 시각화와 같은 아티팩트를 생성합니다.

- No CD:
  모델 버전이 자주 배포되지 않으므로 CD는 고려되지 않습니다.

- Deployment refers to the prediction service:
  이 프로세스는 전체 ML 시스템을 배포하는 대신 학습된 모델을 예측 서비스로 배포하는 것에만 관여함

- Lack of active performance monitoring:
  프로세스는 모델 성능 저하 및 기타 모델 동작 드리프트를 감지하는 데 필요한 모델 예측 및 작업을 추적하거나 기록하지 않습니다

Level 1 MLOps

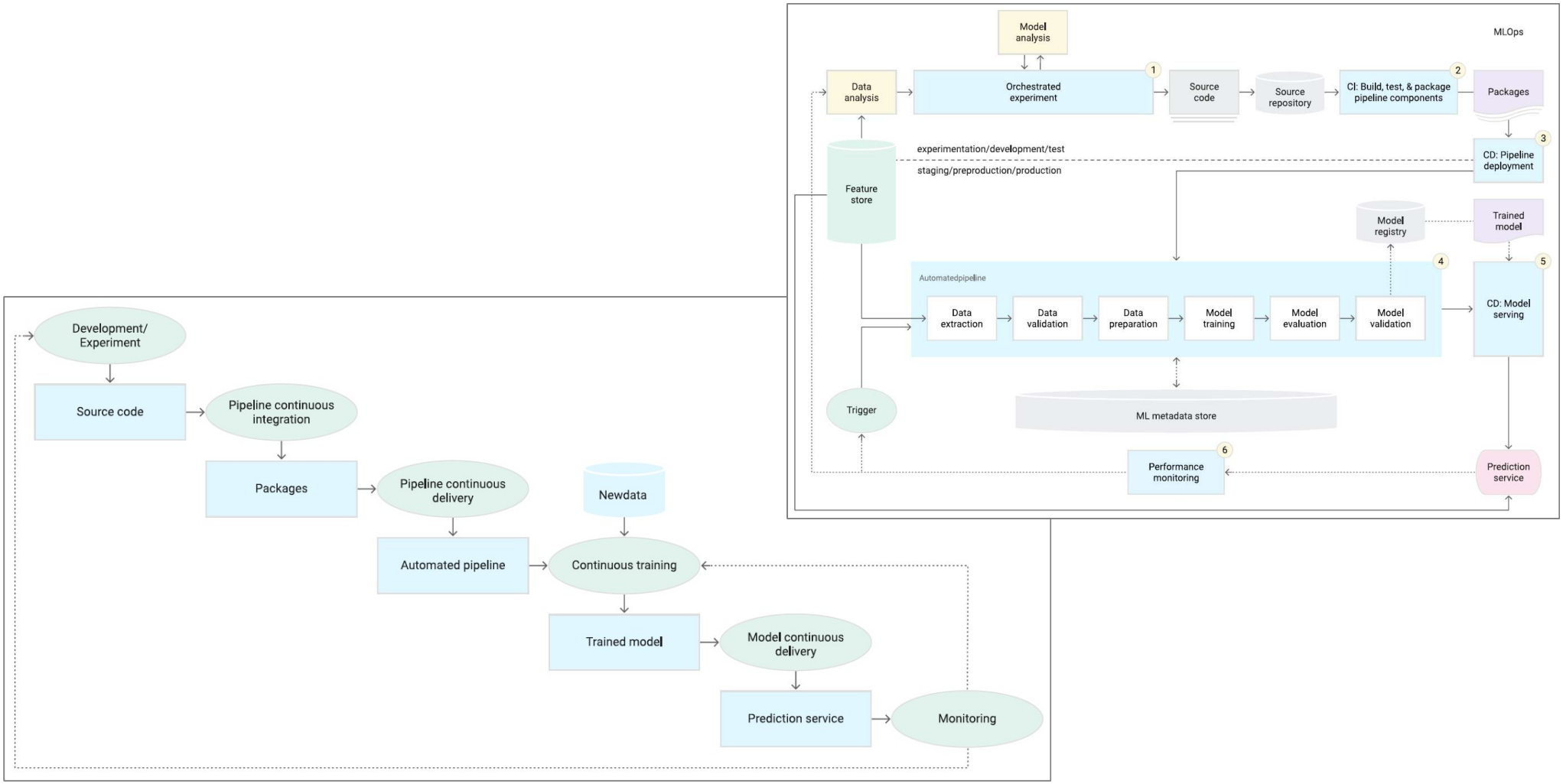


[그림] CT용 ML 파이프라인 자동화

Level 1 MLOps

- **빠른 실험:**
  ML 실험 단계가 조정됩니다. 단계 간 전환은 자동으로 이루어지므로 실험을 빠르게 반복하고, 전체 파이프라인을 프로덕션으로 더 빠르게 이동할 수 있습니다.
- **프로덕션 단계에서 모델의 CT:**
  다음 섹션에서 설명하는 실시간 파이프라인 트리거를 기반으로 하는새로운 데이터를 사용하여 모델이 프로덕션 단계에서 자동으로 학습됩니다.
- **실험-운영 균형:**
  개발 또는 실험 환경에서 사용되는 파이프라인 구현은 사전 프로덕션 및 프로덕션 환경에서 사용되며, 이는 DevOps 통합을 위한 MLOps 방식에 있어 핵심적인 요소입니다.
- **구성요소 및 파이프라인의 모듈화된 코드:**
  ML 파이프라인을 구성하려면 ML 파이프라인 전체에서구성요소를 재사용, 구성, 공유할 수 있어야 합니다. 따라서 EDA 코드는 여전히 노트북에 있을 수 있지만 구성요소의 소스 코드는 모듈화되어야 합니다. 또한 구성요소를 컨테이너화하여 다음을 수행하는 것이 좋습니다.
  - 커스텀 코드 런타임에서 실행 환경을 분리합니다.
  - 개발 및 프로덕션 환경 간에 코드를 재현 가능하도록 합니다.
  - 파이프라인의 각 구성요소를 격리합니다. 구성요소는 자체 런타임 환경 버전을 가질 수 있으며다양한 언어 및 라이브러리를 갖습니다.
- **모델의 지속적 배포:**
  프로덕션 단계의 ML 파이프라인은 새 데이터로 학습된 새 모델에 예측 서비스를 지속적으로 배포합니다. 학습 및 검증된 모델을 온라인 예측용 예측 서비스로 제공하는 모델 배포단계가 자동화됩니다.
- **파이프라인 배포:**
  수준 0에서는 학습된 모델을 프로덕션에 예측 서비스로 배포합니다. 수준 1의 경우, 학습된 모델을 예측 서비스로 제공하기 위해 자동으로 반복 실행되는 전체 학습 파이프라인을 배포합니다

Level 2 MLOps



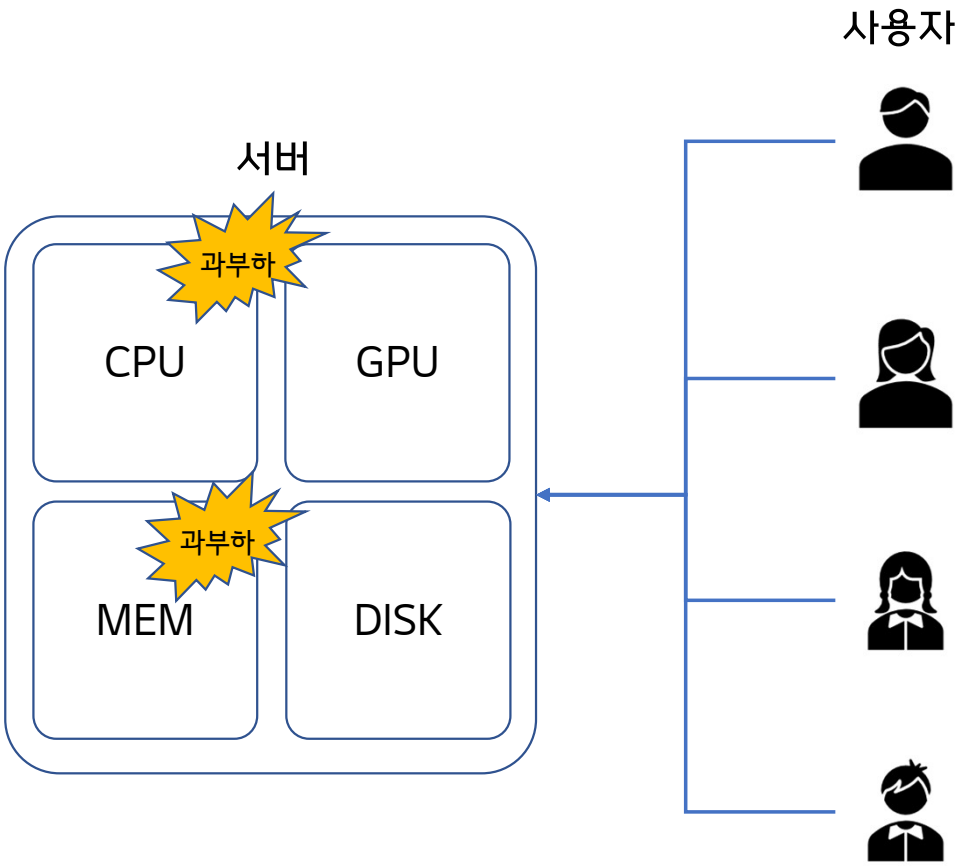
[그림] ML CI/CD 자동화 파이프라인의 단계

Level 2 MLOps

- 개발 및 실험:
  새 ML 알고리즘과 실험 단계가 조정되는 새 모델링을 반복적으로 시도합니다. 이 단계의 출력은 ML 파이프라인 단계의 소스 코드이며, 소스 코드는 소스 저장소로 푸시됩니다.
- 파이프라인 지속적 통합:
  소스 코드를 빌드하고 다양한 테스트를 실행합니다. 이 단계의 출력은 이후단계에서 배포될 파이프라인 구성요소(패키지, 실행 파일, 아티팩트)입니다.
- 파이프라인 지속적 배포:
  CI 단계에서 생성된 아티팩트를 대상 환경에 배포합니다. 이 단계의 출력은모델의 새 구현이 포함되는, 배포된 파이프라인입니다.
- 자동화된 트리거:
  파이프라인은 일정 또는 트리거에 대한 응답에 따라 프로덕션 단계에서 자동으로실행됩니다. 이 단계의 출력은 모델 레지스트리로 푸시되는 학습된 모델입니다.
- 모델 지속적 배포:
  학습된 모델을 예측의 예측 서비스로 제공합니다. 이 단계의 출력은 배포된 모델예측 서비스입니다.
- 모니터링:
  실시간 데이터를 기반으로 모델 성능의 통계를 수집합니다. 이 단계의 출력은 파이프라인을 실행하거나 새 실험 주기를 실행하는 트리거입니다.

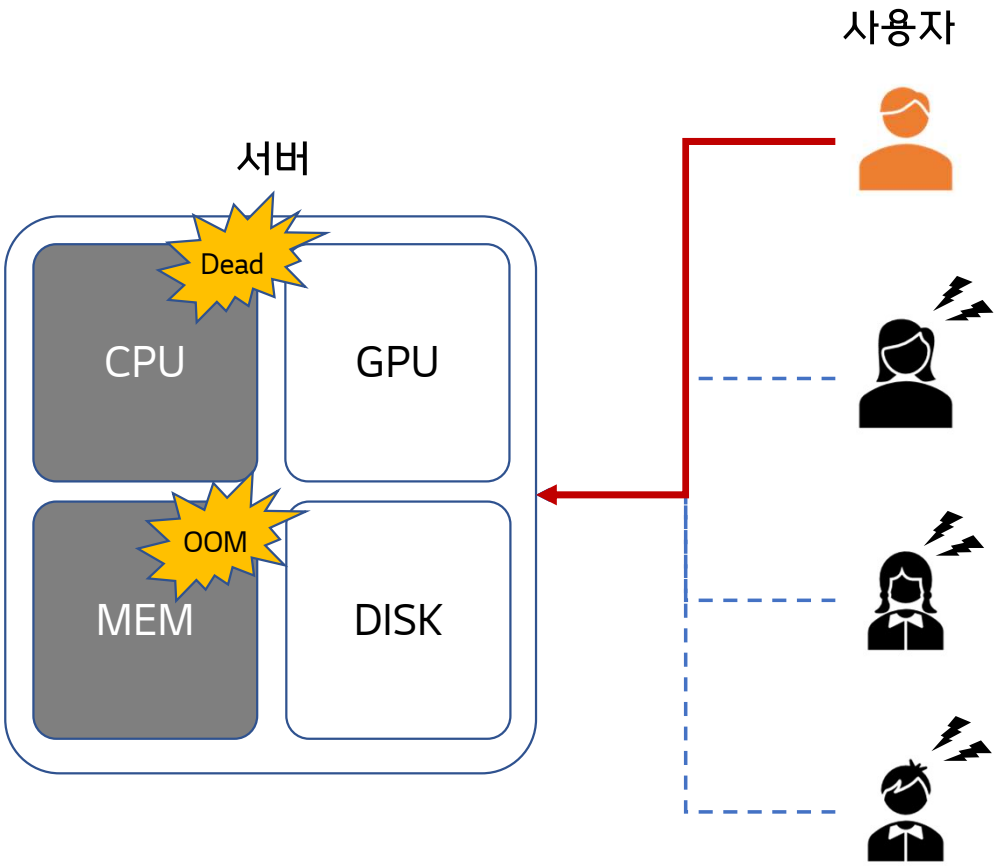
Docker and Kubernetes

기존의 서버 사용 방식



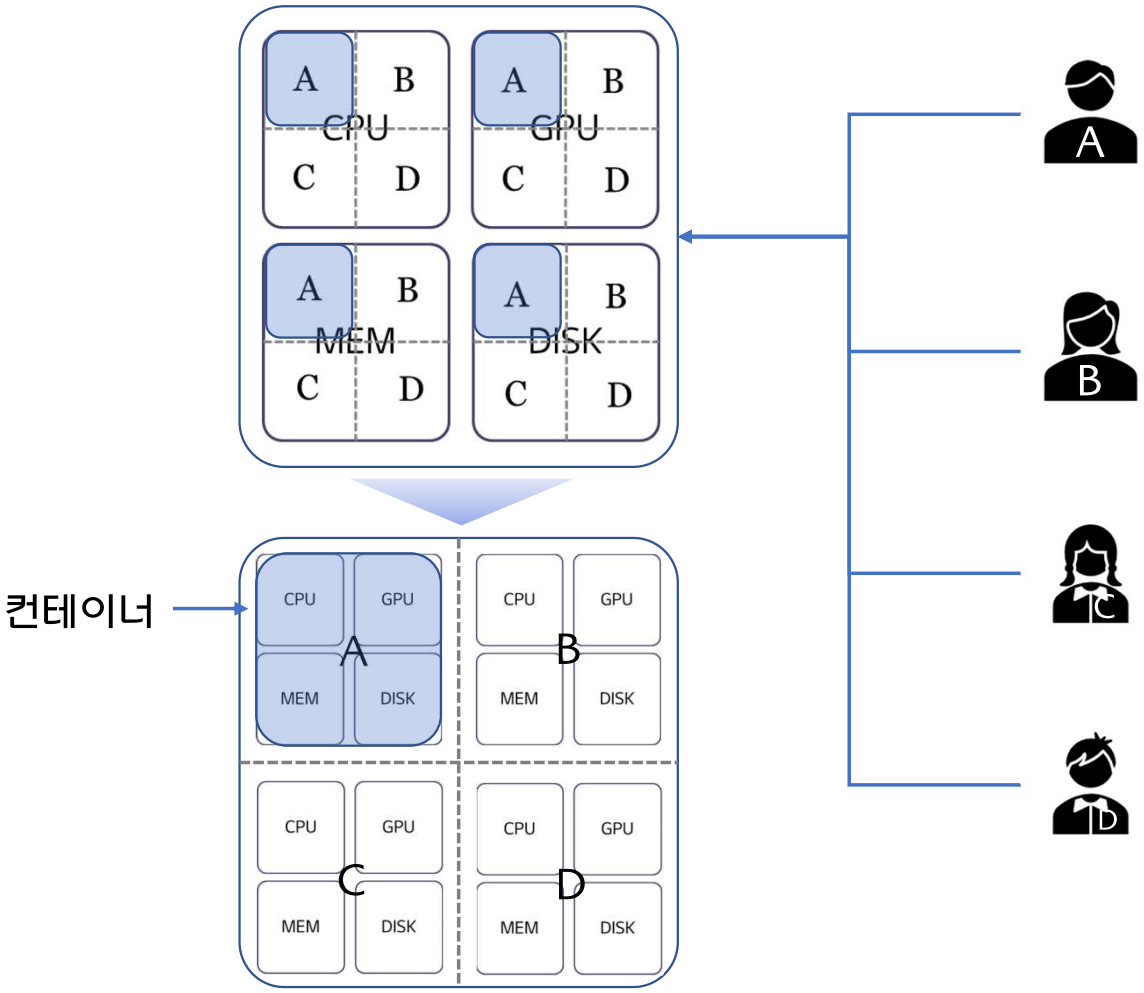
- 1대의 컴퓨터에 여러 사람이 접속하여 작업을 요청함
  (프로그램을 실행함)
- 동시 접속자들의 접속상태를 유지하기 위해 최소한의 CPU와 메모리가 소모됨
- 컴퓨터의 운영상태를 최적화 하기 위해 작업의 우선순위를 자동으로 결정하기 위한 알고리즘을 연구함

기존의 서버 사용 방식



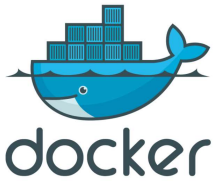
- 하나의 잘못된 작업으로 인해 컴퓨터 전체의 문제를 일으킴
- 같은 공간상에서 각 사용자의 작업이 이루어지므로 보안에 매우 취약함
- 각 사용자들의 원하는 환경 세팅이 다른 경우 대응이 불가능함

기존의 서버 사용 방식

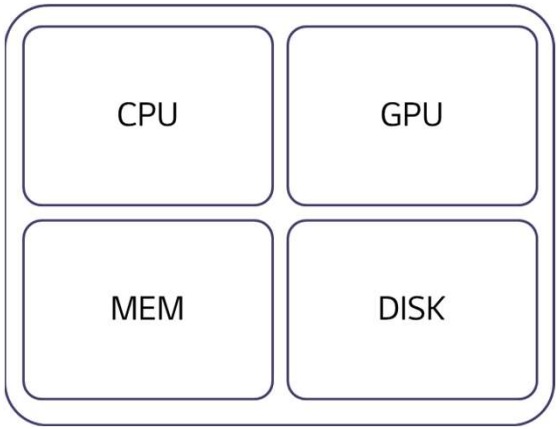


- 하나의 잘못된 작업으로 인해 컴퓨터 전체의 문제를 일으킴
- 같은 공간상에서 각 사용자의 작업이 이루어지므로 보안에 매우 취약함
- 각 사용자들의 원하는 환경 세팅이 다른 경우 대응이 불가능함

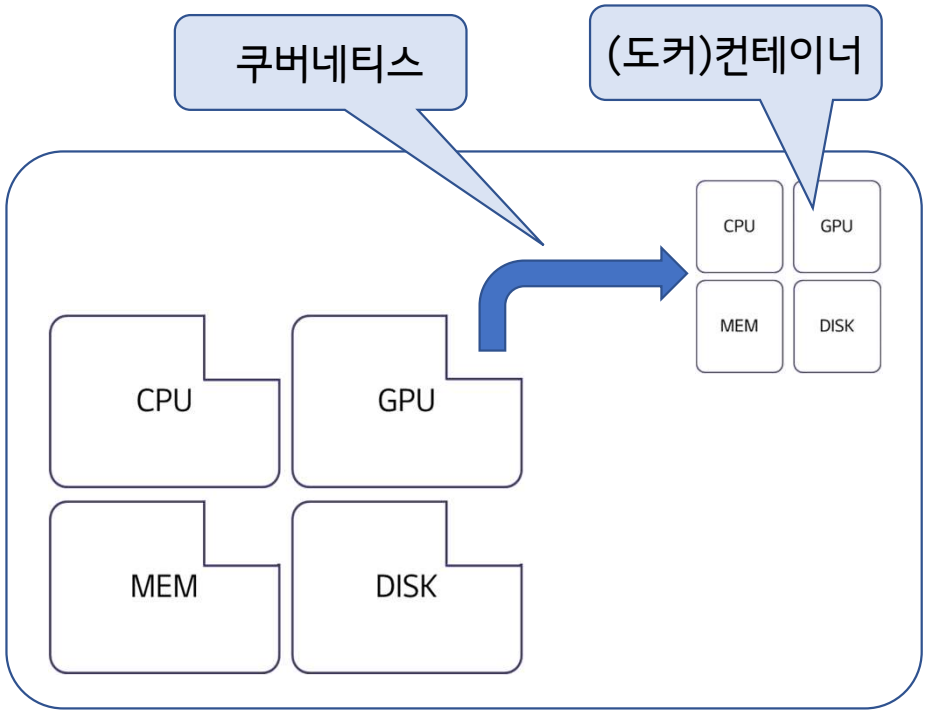
Docker and Kubernetes



컨테이너를 만들어주는 OSS
: docker



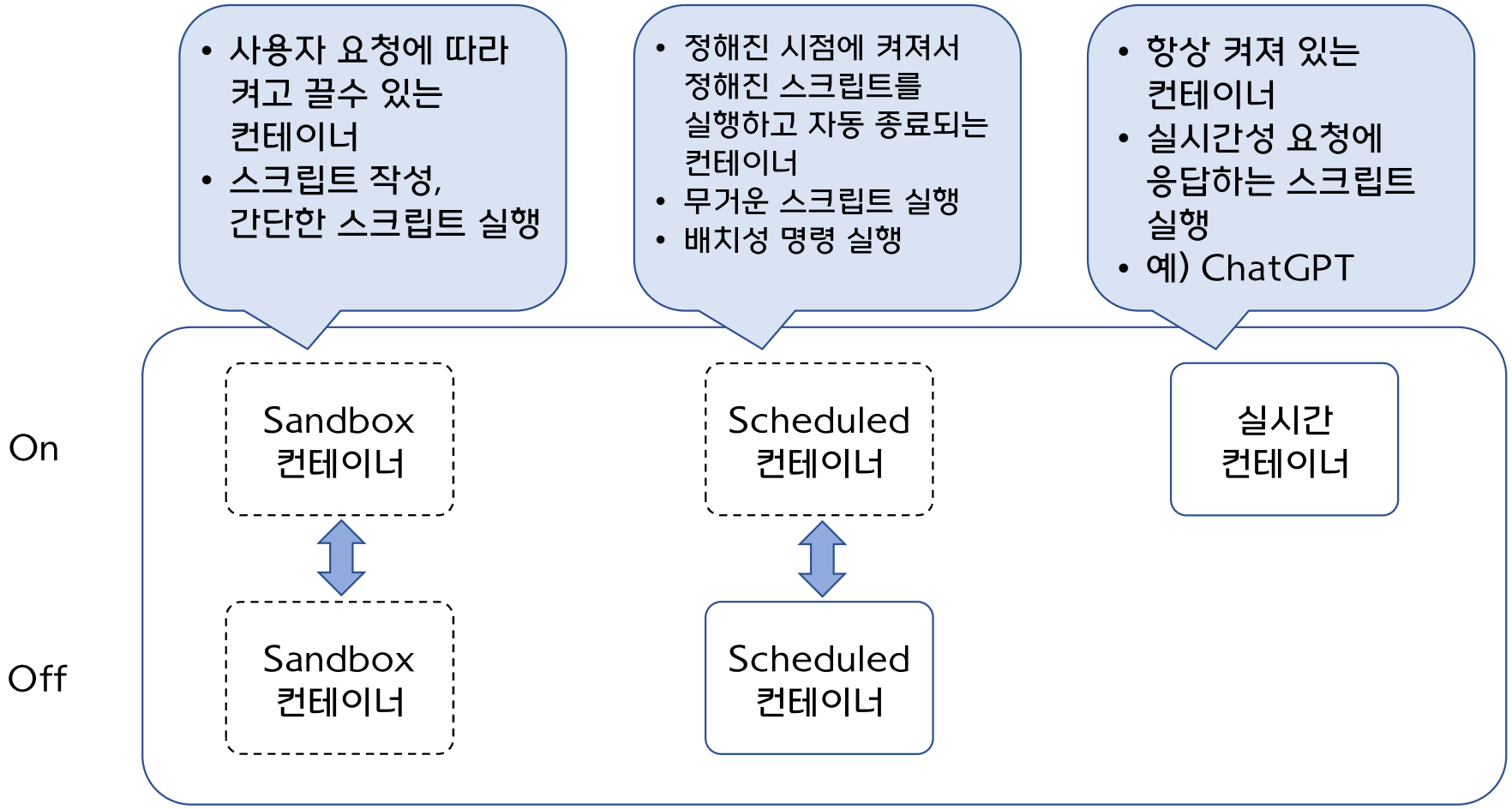
도커 컨테이너를 컨트롤하는
OSS: Kubernetes



가상화 서버 방식

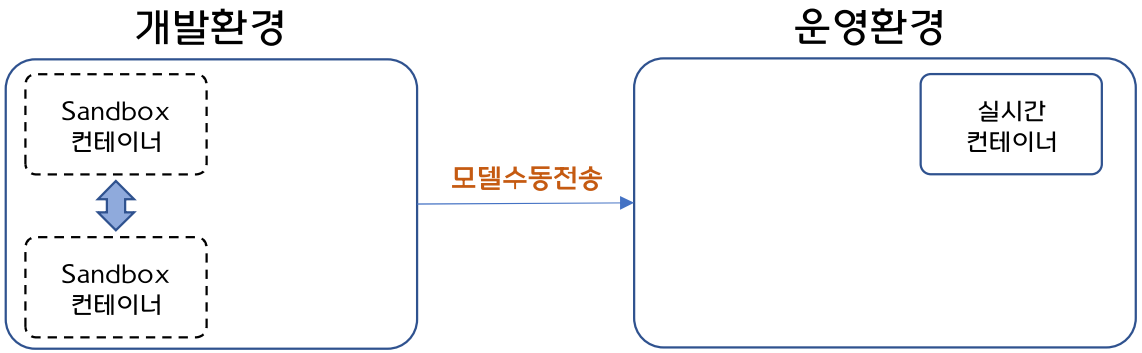
- 쿠버네티스를 이용하여 여러 역할별 도커 컨테이너를 생성/관리하는 플랫폼을 내에서 데이터 분석을 수행하는 것이 일반적인 추세임
- 머신러닝 모델을 사용하는 방식은 데이터 탐색 → 학습스크립트 생성 → 학습(모델생성) → 성능검증 → 모델배포 → 모델운영(추론) → 성능 모니터링 → 재학습의 과정을 거치게 됨
- 머신러닝 모델활용의 각 단계별로 다른 성격의 컨테이너가 필요하고, 머신러닝 플랫폼, 데이터 분석 플랫폼에서는 성격이 다른 여러 형태의 컨테이너를 생성, 운영, 수정, 삭제할 수 있는 기능을 탑재하고 있음 (쿠버네티스 활용)
- sandbox 컨테이너: 사용자가 원하는 시점에 켜고 끌수 있고, 원하는 내용을 자유롭게 수행할 수 있음
- Scheduled 컨테이너: 지정된 시점에 켜져서 정해진 스크립트를 실행하고 완료되면 자동종료됨
- 실시간 컨테이너: 실시간 추론을 위해 항상 켜져 있고 실시간요정에 대해 응답을 전송하는 컨테이너

가상화 서버 방식

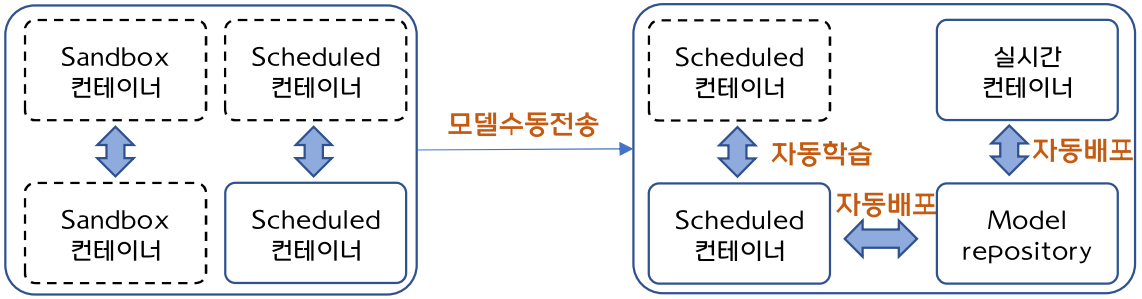


MLOps와 가상화 서버

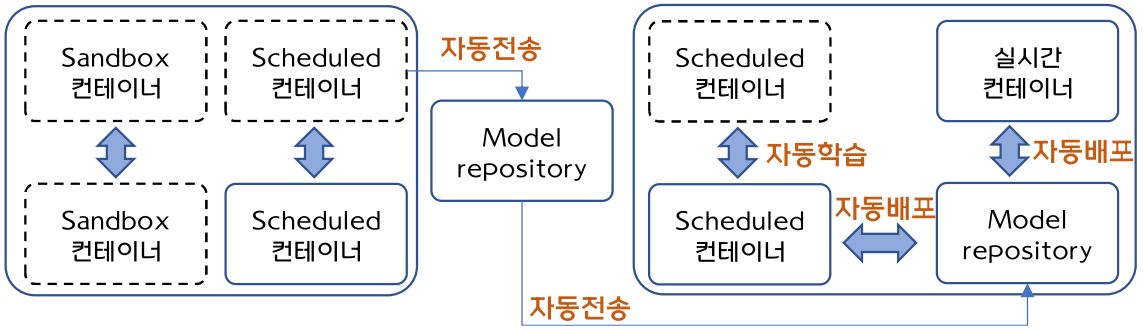
- Level0 MLOps



- Level1 MLOps



- Level2 MLOps



End of Document