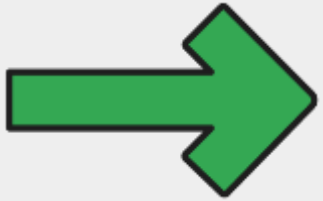
Google Developer Group
Incheon National University

MLOps

GDGoC Backend Seminar

Member 이은지

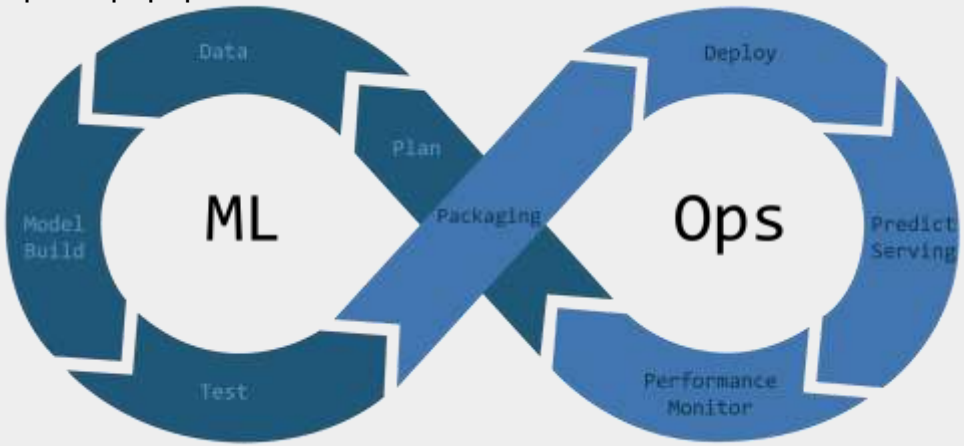


Contents

What is MLOps?

Machine Learning + Operation

- 머신러닝 모델의 개발부터 배포, 모니터링,
  유지보수까지 전체 생애주기를 자동화하고 최적화

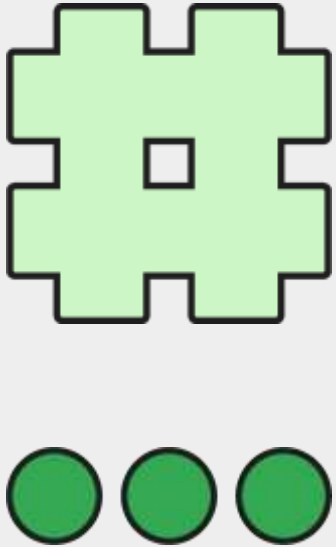


DevOps

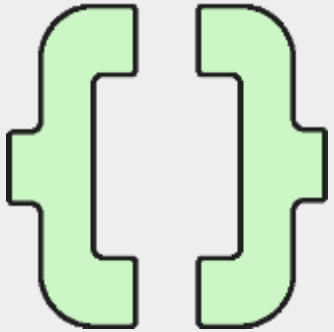
- DevOps는 Development(개발)와 Operations(운영)의 합성어로, 소프트웨어 개발과 IT 운영 팀 간의 협업을 강화하여 더 빠르고 안정적인 서비스 제공을 가능하게 하는 문화, 철학, 방식 및 도구의 조합

- DevOps의 주요 원칙

1. 지속적 통합(CI)
2. 지속적 배포(CD)
3. 협업과 커뮤니케이션
4. 자동화

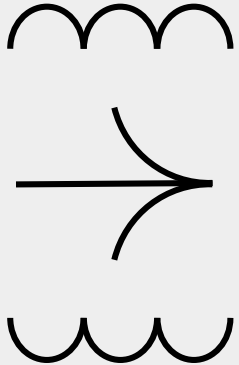


DevOps vs MLOps



DevOps:

- CI(지속적 통합)와 CD(지속적 배포)에 중점
- 코드 중심의 워크플로우



MLOps:

- CI, CD에 CT(지속적 학습) 개념 추가
- 코드 뿐만 아니라 데이터와 모델도 테스트 및 검증
- 모델 재학습 및 자동 배포 프로세스 포함

MLOps's principle

1. Testing

- Features and Data Tests
- ML Model Test
- ML Infra Test

2. Monitoring

- 모델 성능, 데이터 드래프트, 예측 품질 실시간 추적

3. Versioning

- 버전 관리로 데이터셋과 모델 관련 코드의 변경 사항을 체계적으로 관리

4. Continous X

- CI/CD 개념을 ML에 확장한 지속적 통합 & 배포

5. Automation

- 데이터 전처리부터 모델 배포까지 전체 파이프라인 자동화

6. Reproducibility

- 동일한 입력으로 항상 동일한 결과 보장

ML Level

MLOps 0단계

- 완전한 수동

MLOps 1단계

- 지속적 학습 및 지속적 모델 배포
- ML Pipeline을 자동화하여 모델을 지속적으로 학습시키고, 모델 예측 서비스에 지속적으로 제공

MLOps 2단계

- 지속적 통합 및 지속적 배포
- 자동화 시스템을 활용해 피처 추출, 모델 알고리즘, 하이퍼파라미터 실험으로 생성된 새로운 모델을 서비스에 빠르게 적용 가능

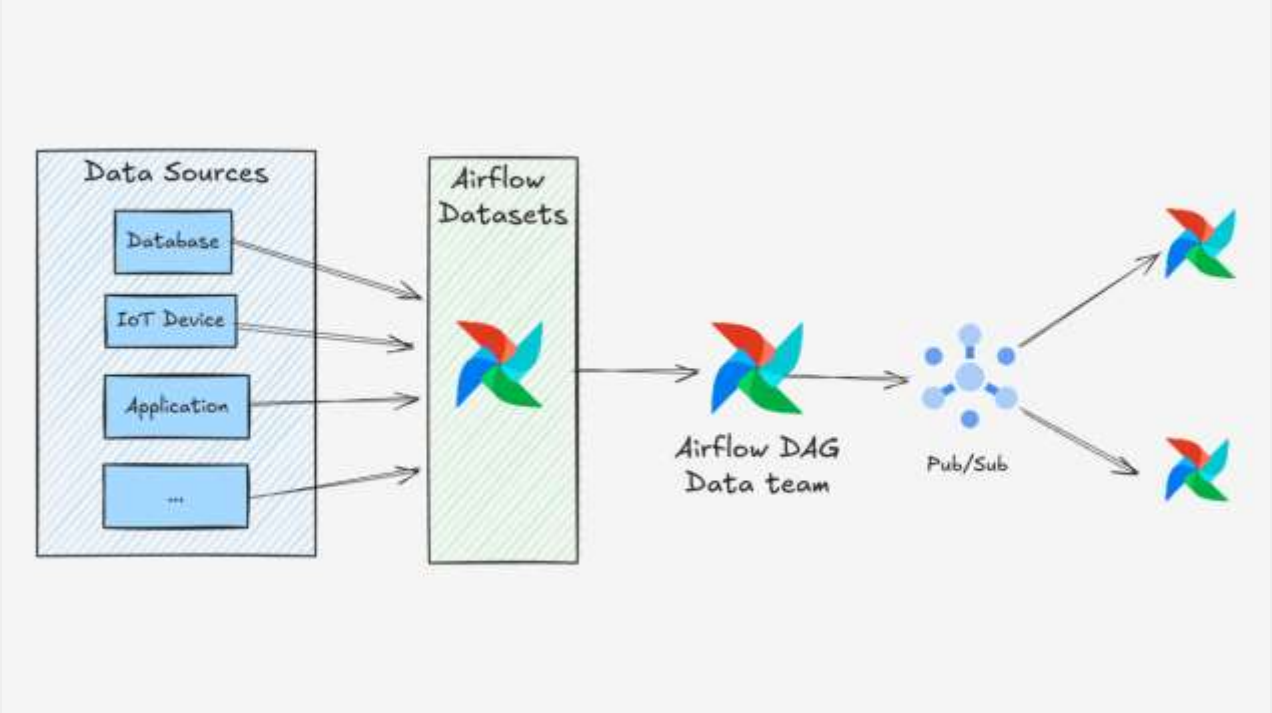
Tools

1. 모델 개발: TensorFlow, Pytorch
2. 파이프라인: Kubeflow, Airflow
3. 모델 서빙: TensorFlow Serving
4. 모니터링: Prometheus, Grafana

Airflow Pipeline

- Airflow는 데이터 파이프라인을 구축, 스케일링 관리하는 오픈 소스 플랫폼
- DAG를 사용해 작업을 정의하고, 각 작업은 하나 이상의 Task로 구성됨
- Task는 노드로 표현되며, 노드는 서로 연결되어 작업 흐름을 나타냄

Airflow Pipeline



Airflow DAG

```python
# DAG의 기본 인자 정의
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

# DAG 객체 생성
with DAG(
    'simple_data_pipeline',
    default_args=default_args,
    description='간단한 데이터 파이프라인 예제',
    schedule_interval='@daily',
    start_date=datetime(2025, 5, 1),
    catchup=False,
) as dag:

    # 작업 1: 데이터 수집 (Bash 명령어 실행)
    collect_data = BashOperator(
        task_id='collect_data',
        bash_command='echo "데이터 수집 중..." > /tmp/collected_data.txt',
    )

    # 작업 2: 데이터 처리 (Python 함수 실행)
    def process_data(**kwargs):
        print("데이터 처리 중...")
        return "처리된 데이터"
```

```python
process_task = PythonOperator(
    task_id='process_data',
    python_callable=process_data,
)

# 작업 3: 결과 저장 (Bash 명령어 실행)
store_results = BashOperator(
    task_id='store_results',
    bash_command='echo "결과 저장 중..." && mkdir -p /tmp/results/',
)

# 작업 4: 완료 알림 (Python 함수 실행)
def send_notification(**kwargs):
    print("작업 완료 알림 전송 중...")
    return "알림 전송 완료"

notify = PythonOperator(
    task_id='send_notification',
    python_callable=send_notification,
)

# 작업 간 의존성 설정 (실행 순서 정의)
collect_data >> process_task >> store_results >> notify
```

Metaflow

- Netflix가 개발한 머신러닝 프레임워크, 2019년 오픈소스로 공개
- 클라우드 네이티브: AWS 서비스와의 원활한 서비스 제공
- 자동 버전 관리: 코드, 데이터 의존성을 자동으로 저장하여 재현성 보장
- 간결한 워크플로우: DAG 형태로 단계별 작업 정의

Conclusion

Considerations

- 조직 문화와 팀 구조의 변화 필요성
- 데이터 품질 관리의 중요성
- 규제 준수 및 윤리적 고려사항

Future

- AutoML과의 통합
- 멀티클라우드 환경에서의 MLOps
- 엣지 컴퓨팅과 MLOps의 결합

Thank
You