MLOps 및 자동화 QA

A. 모델 배포 및 버전 관리

Q1. 모델 배포는 어떤 방식으로 이루어지나요?
A1.
1. 일반적으로 REST API 형태로 모델을 서비스화하며, Flask/FastAPI + Docker를 조합하여 배포합니다.
2. 배포 대상에 따라 Batch 예측, Streaming 예측, Real-time API 예측 중 적절한 방식을 선택합니다.
3. 클라우드 환경(GCP/AWS/Vertex AI/SageMaker 등)을 활용하면 Auto-scaling 및 CI/CD 연동이 쉬워집니다.

Q2. 모델 버전 관리는 어떻게 하나요?
A2.
1. MLflow Model Registry, DVC(Data Version Control), 또는 Weights & Biases를 사용해 모델과 코드, 하이퍼파라미터, 메트릭을 함께 버전 관리합니다.
2. 모델 버전은 학습일자, 데이터셋 버전, 실험 조건 등을 기준으로 명명하고, 프로덕션 등록 여부(staging/production/archive)를 명확히 구분합니다.
3. CI/CD 파이프라인에 모델 승격 조건(예: AUC > 기존 모델)을 넣어 자동으로 배포되도록 할 수 있습니다.

Q3. 여러 모델 중 어떤 버전을 서비스해야 할지 어떻게 결정하나요?
A3.
1. **실험 메트릭 기반 비교(A/B 테스트)**로 기존 모델과의 성능 차이를 검증합니다.
2. 일반적으로 테스트 기간을 설정하고 전환율, 정확도, 지연시간 등 실전 지표를 모니터링합니다.
3. 가장 성능이 우수하고 안정적인 모델을 Production에 할당하고, 나머지는 Archive 또는 Shadow 모드로 보관합니다.

B. 모델 성능 모니터링

Q4. 모델 성능 저하는 어떻게 감지하나요?
A4.
1. **예측값 분포, 입력 데이터 통계, 메트릭(AUC, RMSE 등)**을 실시간 또는 주기적으로 수집해 모니터링합니다.
2. **데이터 드리프트(Data Drift)**나 컨셉 드리프트(Concept Drift) 감지 도구(Evidently, WhyLabs 등)를 활용할 수 있습니다.

3. 기준보다 성능이 떨어지면 자동 알림(Slack, email, PagerDuty)과 재학습 트리거 설정이 가능합니다.

Q5. 입력 데이터가 바뀌었을 때 성능 영향은 어떻게 판단하나요?
A5.
1. 데이터 통계 비교(KS-statistics, PSI, JS divergence) 등으로 입력 분포의 변화 감지
2. 예측 결과의 분포나 confidence score 변화도 함께 모니터링
3. 필요 시 재학습 이전에 Shadow Deployment로 성능 비교 후 운영 반영 여부 결정

Q6. 실시간 예측 시스템에서는 어떻게 모니터링하나요?
A6.
1. Prometheus + Grafana 같은 APM 도구를 통해 지연시간, 오류율, 트래픽 등을 시각화
2. 로그 기반으로 예측 입력/출력 저장 → 하루 단위로 메트릭 추출 및 대시보드 구성
3. FastAPI 또는 Flask에서 직접 요청 메타데이터 로그를 설정하고, 에러 핸들링 포함

C. 자동화 파이프라인 및 스케줄링
Q7. 모델 학습 파이프라인은 어떻게 자동화하나요?
A7.
1. 데이터 수집 → 전처리 → 학습 → 검증 → 배포 → 모니터링까지의 단계별 파이프라인을 구성합니다.
2. Airflow, Kubeflow, Prefect 등을 활용해 DAG 기반 스케줄링을 구현합니다.
3. 각 태스크를 Docker 컨테이너 단위로 구성하면 유지보수와 이식성이 높아집니다.

Q8. Airflow에서 모델 학습을 자동화할 때 어떤 점을 주의해야 하나요?
A8.
1. 각 Task 간 의존 관계(DAG)를 명확히 설정하고, 실패 시 재시도 또는 fallback 전략을 정의합니다.
2. 학습 결과(모델 파일, 메트릭)는 S3, GCS, DB 등에 저장하고 후속 Task에서 접근 가능하게 합니다.
3. 데이터 크기가 크면 XCom 대신 공유 스토리지를 통한 전달 방식이 효율적입니다.

Q9. 재학습은 어떤 주기로 자동화하는 것이 좋나요?
A9.
1. 일반적으로 주간/월간 재학습 또는 성능 기준 도달 시 트리거 방식을 병행합니다.
2. 데이터 양 증가 또는 드리프트 발생 시 자동으로 학습 파이프라인 실행되도록 설정합니다.

3. 특정 요일, 시간대에 실행되도록 Airflow, Cloud Scheduler로 조절할 수 있습니다.

D. 컨테이너 및 배포 인프라

Q10. Docker는 어떤 경우에 사용하나요?

A10.

1. 모델 환경(라이브러리, 설정 등)을 컨테이너로 패키징해 어디서든 동일한 실행 환경을 보장합니다.
2. MLOps에서는 학습용, 예측용 이미지로 나누어 관리하며, CI/CD 시 빠른 테스트와 배포가 가능합니다.
3. Kubernetes와 연동하여 스케일링, 롤링 업데이트, 장애 복구 등을 구현할 수 있습니다.

Q11. 모델 API는 어떻게 운영 환경에 올리나요?

A11.

1. Docker 이미지로 만든 모델 API를 Kubernetes, AWS ECS, GCP Cloud Run 등으로 배포
2. CI/CD 도구(GitHub Actions, Jenkins 등)와 연결해 자동 빌드 및 배포 파이프라인 구성
3. 로깅, 모니터링, 헬스체크(Liveness/Readiness Probe) 설정은 필수입니다.

Q12. MLOps 도입이 꼭 필요한 상황은 어떤 경우인가요?

A12.

1. 모델이 자주 재학습되거나, 여러 모델을 동시에 운영하는 경우
2. 모델 품질에 따라 사업 영향이 큰 경우(예: 금융, 의료, 마케팅 자동화 등)
3. 모델 성능 추적 및 품질 관리, 재현 가능한 실험 환경이 필요한 경우
4. 팀 단위 협업이 필요한 대규모 프로젝트에서는 필수 요소로 간주됩니다.