서버리스 컴퓨팅을 활용한 리뷰 데이터 처리 파이프라인 구축

> ⚠️ **중요**
>
> AWS에서 **비용 이상징후 탐지-모니터 설정** 및 **예산 설정**을 꼭 하시기 바랍니다.
>
> 의도치 않은 지출이 발생하지 않도록 주의해주시고, **과제 완료 시 생성했던 리소스를 모두 삭제**해주세요.

English version

▸ FAQ

개요

- e-commerce 플랫폼에서 고객 리뷰 데이터를 효과적으로 처리하는 서버리스 컴퓨팅 파이프라인을 AWS 서비스를 활용해 설계하고 구현한다.
- 이벤트 기반 데이터 처리의 기본 개념을 학습하고, 퍼블릭 클라우드 플랫폼(AWS)의 다양한 서버리스 관련 서비스를 통합하여 실제 애플리케이션과 유사한 시스템을 구축하는 것을 목표로 한다.

구현 항목

API Gateway 생성 및 구성

- API 이름: `ReviewProcessingAPI-{student number}`
  - e.g., `ReviewProcessingAPI-2025320001`
- 리소스: `/reviews`
- 메서드: HTTP `POST` 메서드 생성 및 Lambda 함수와 통합
- 배포: API Gateway를 `prod` 스테이지로 배포
- 엔드포인트 URL: 배포 후 생성된 엔드포인트 URL을 활용

Lambda 함수 설정

- Lambda 함수 이름: `review-processing-{student number}`.

- 런타임: `Python 3.13`
- IAM 역할: 필요한 역할을 부여할 것
- **제공한 코드를 완성하고 필요 라이브러리와 함께 패키징하여 Lambda에 업로드**할 것
  - 필요 라이브러리: `TextBlob`, `boto3`

리뷰 데이터 생성 및 형식

- HTTP `POST` 요청으로 리뷰 데이터를 수집
- 데이터 형식: JSON
- 입력 예시:

```
{ "user_name": "Alice", "review": "This product is fantastic!" }
```

DynamoDB에 데이터 저장:

- 분석된 리뷰 데이터를 DynamoDB 테이블에 저장
- DynamoDB 테이블 스키마(schema):
  - 테이블 이름: `reviews-{student number}`
  - **Partition Key**: `user_name` (string)
  - **Sort Key**: `timestamp` (string)
- 저장 데이터 예시

```
{ "user_name": "Alice", "review": "This product is fantastic!", "senti
ment": "Positive", "timestamp": "2024-11-24T12:34:56Z" }
```

AWS Simple Email Service(SES)를 활용한 이메일 전송:

- 긍정적인 리뷰가 감지되면 관리자의 이메일로 알림을 전송
  - `“sentiment”: “Positive”` 인 경우

- 이메일 제목 및 본문 예시:

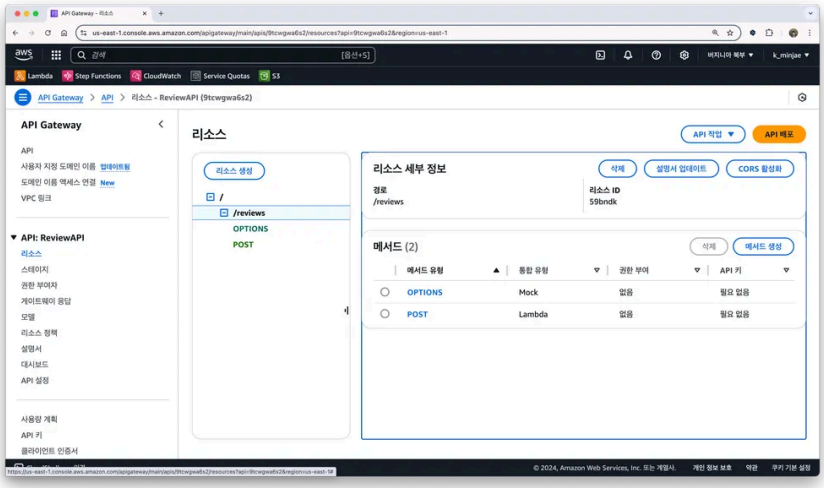
```
Subject: Positive Review from Alice Body: Alice wrote: "This product i
s fantastic!"
```

제출물

1. 코드:
   - **완성된** Lambda 함수 코드 + **라이브러리**
     → `{student number}.zip` 파일로 전송

2. 보고서(PDF only):
   → `{student number}_assignment4.pdf` 형식으로 제출

- 포함하는 자료:
  - **API gateway** 리소스 스크린샷 (제출물에는 좌측 패널에 학번이 표현되어야 함)



- 완성된 Lambda 함수 스크린샷
  - 학번이 포함된 **함수 이름**, **런타임 정보**가 드러나도록 할 것