

\$

Lambda와 EventBridge는 여러분에게 날개를 달아줘요

이현재 · 한국외국어대학교



\$ 들어가면서

발표를 들으시면 좋을 분

- Lambda와 EventBridge의 존재는 알고 있으나 어디에 쓰면 좋을지 고민이신 분들
- 혹은 사용을 주저하시는 분들

발표를 들으시면 알게될 것들

- 서버리스에 대해서 알게 돼요
- 람다와 이벤트브릿지를 간단하게라도 사용할 수 있게 돼요
- 이 둘의 장점에 대해서 알게 돼요



소스코드

\$

1. 서버리스는 무엇이고 람다는 뭘까?

\$ 서버리스는 무엇이고 람다는 뭘까?

서버리스 = server + less = 서버가 없다?

\$ 서버리스는 무엇이고 람다는 뭘까?

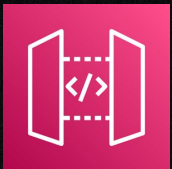
서버리스 = server + less = ~~서버가 없다?~~

서버는 있는데, 관리할 필요가 없다.

\$ 서버리스는 무엇이고 람다는 뭘까?

- 개발자가 관리할 서버가 없다.
- 사용한 만큼 비용을 지불한다.
- 고가용성, 그리고 유연한 확장

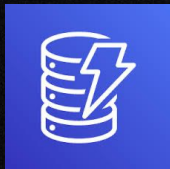
\$ 서버리스는 무엇이고 람다는 뭘까?



API gateway



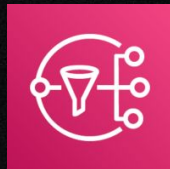
fargate



dynamoDB



S3



SNS



Lambda

etc..

\$ 서버리스는 무엇이고 람다는 뭘까?



Lambda

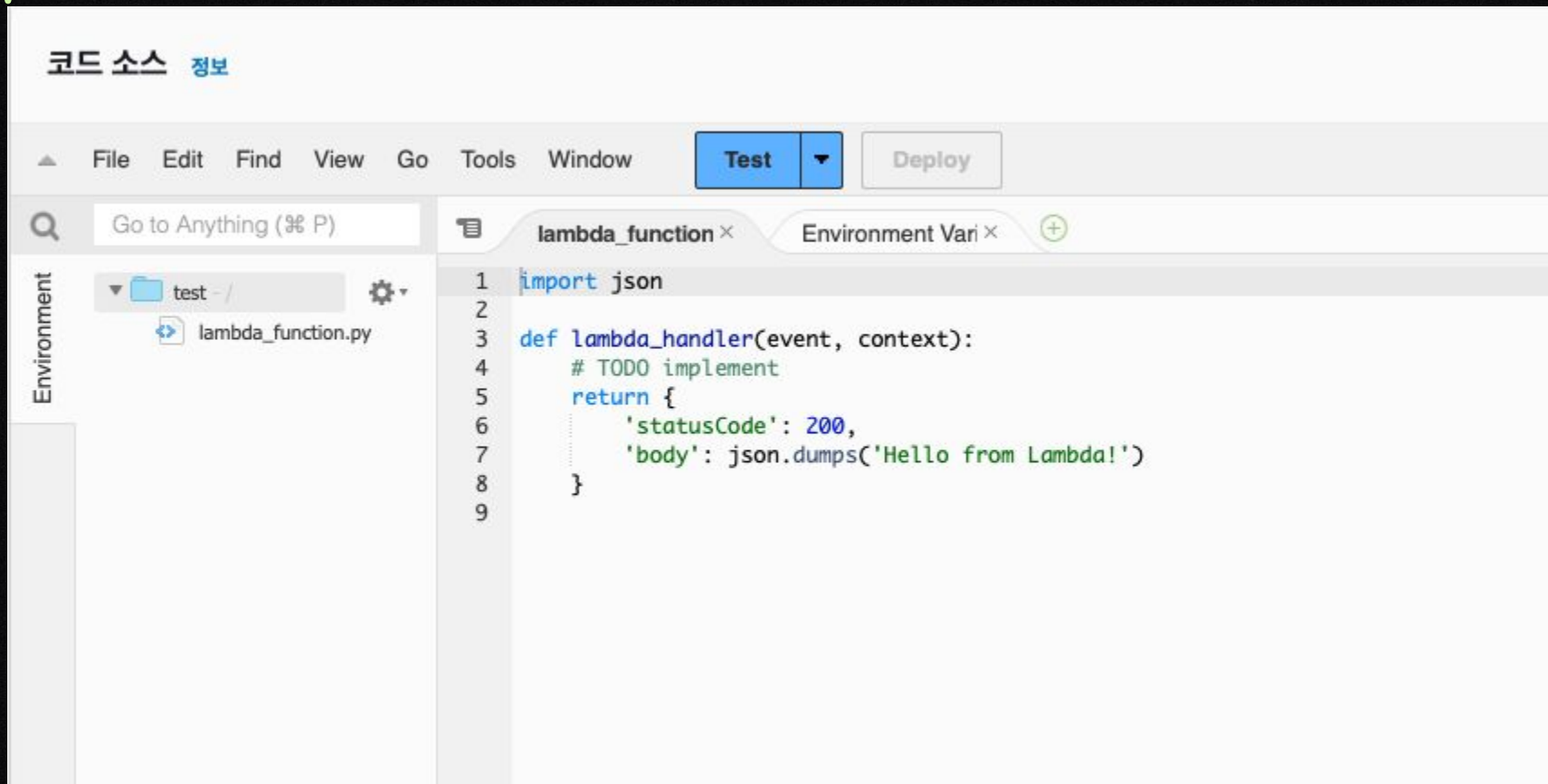
\$ 서버리스는 무엇이고 람다는 뭘까?



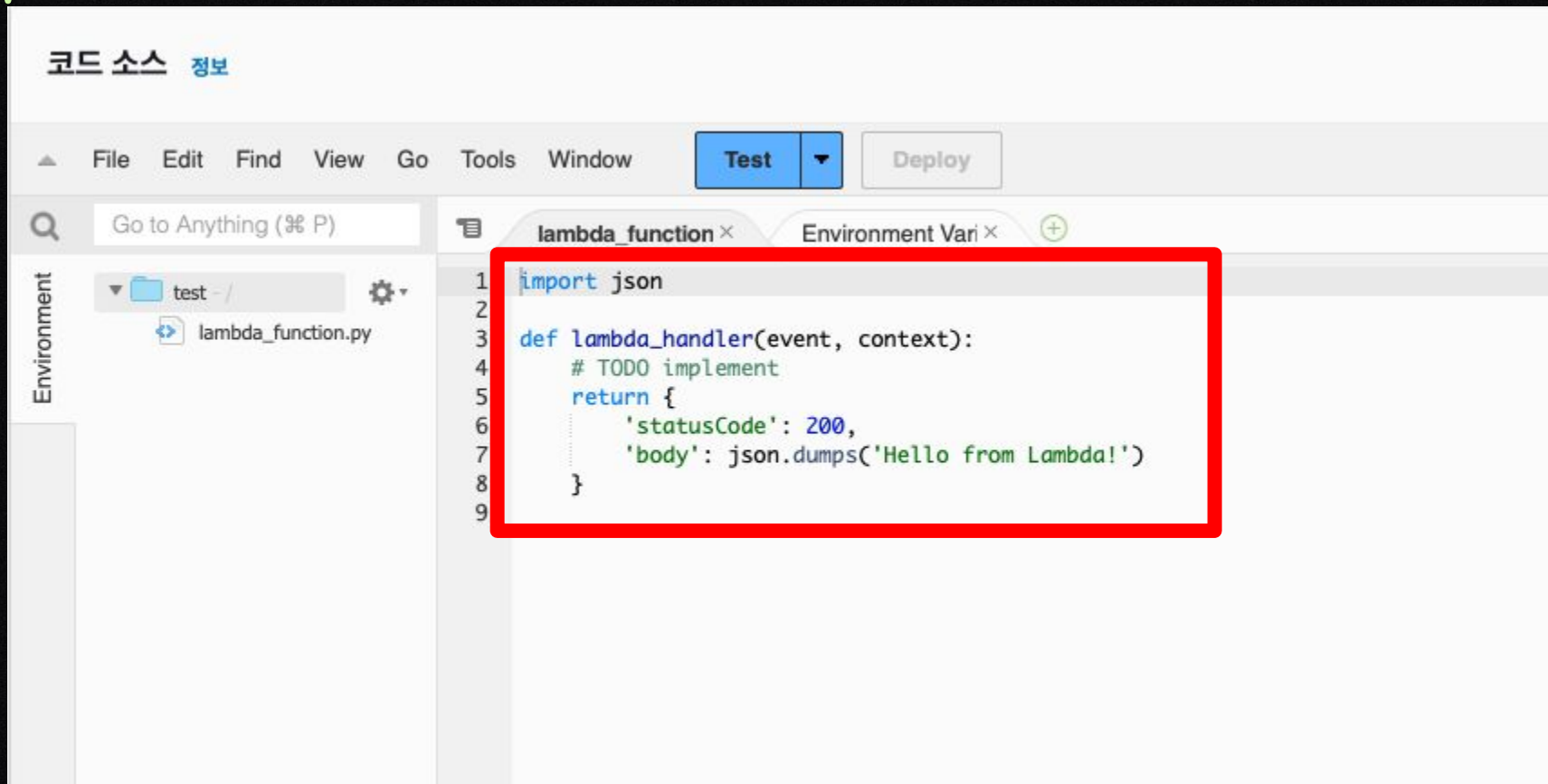
Lambda

- 대표적인 AWS의 서버리스 서비스
- python, go, node.js, java, ruby etc..
- 서버 구축 없이도 빠르고 쉽게 코드를 실행 가능
- 다양한 AWS 서비스와 연계가 좋다
- 요청 1백만 건당 0.20USD

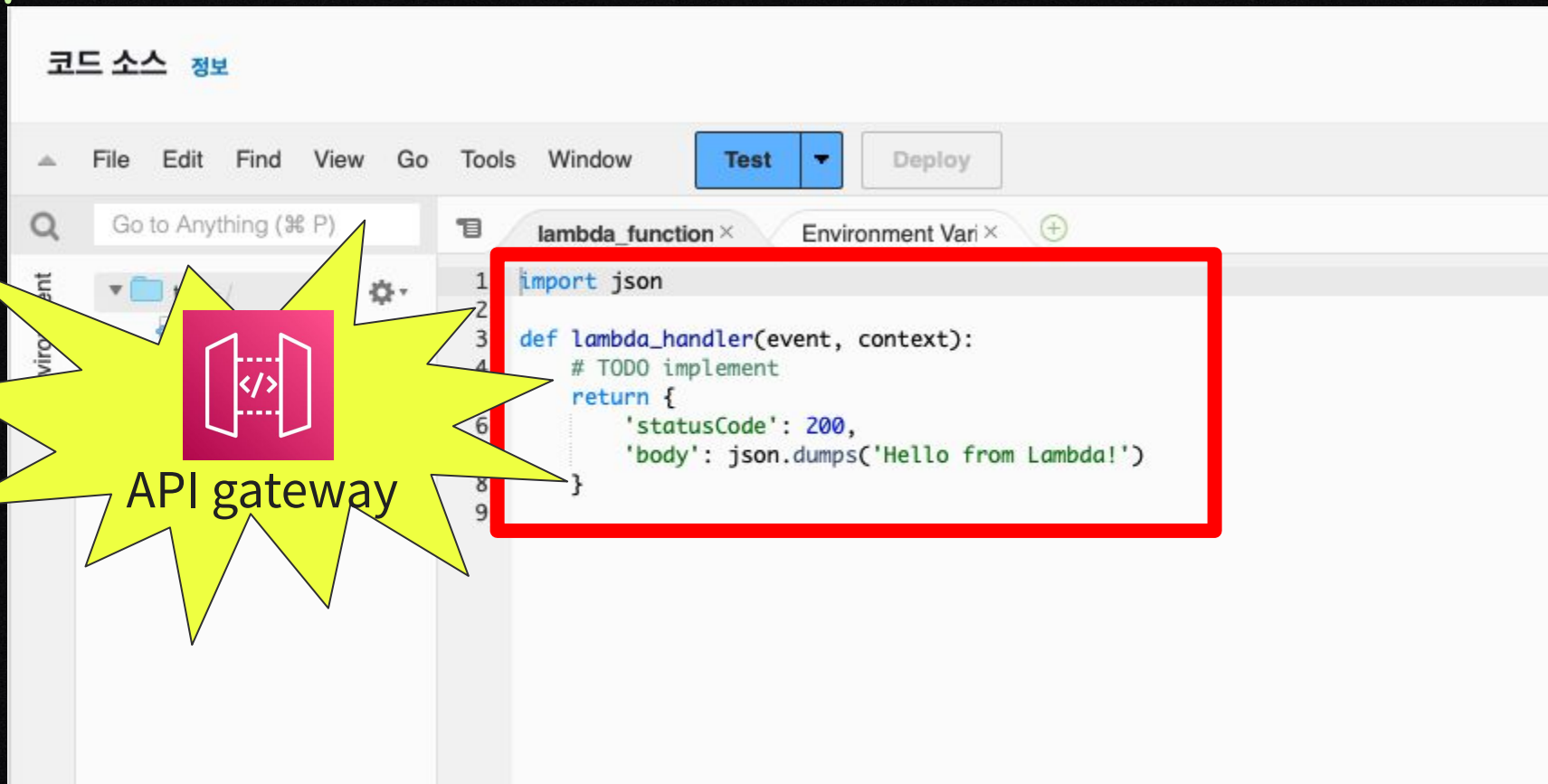
\$ 서버리스는 무엇이고 람다는 뭘까?



\$ 서버리스는 무엇이고 람다는 뭘까?



\$ 서버리스는 무엇이고 람다는 뭘까?



\$ 서버리스는 무엇이고 람다는 뭘까?



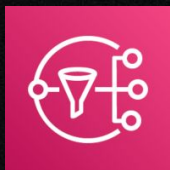
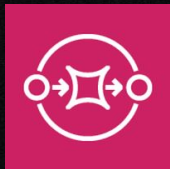
Lambda

- 대표적인 AWS의 서버리스 서비스
- python, go, node.js, java, ruby etc..
- 서버 없이도 빠르고 쉽게 코드를 실행 가능
- 다양한 AWS 서비스와 연계가 좋다
- 요청 1백만 건당 0.20USD

\$

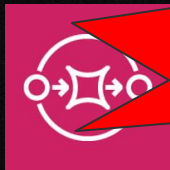
2. 람다의 트리거를 사용한 대표적인 사용 방법

\$ Lambda 트리거란?

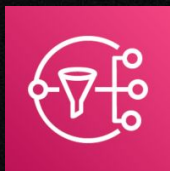


다양한 Trigger 소스들

\$ Lambda 트리거란?

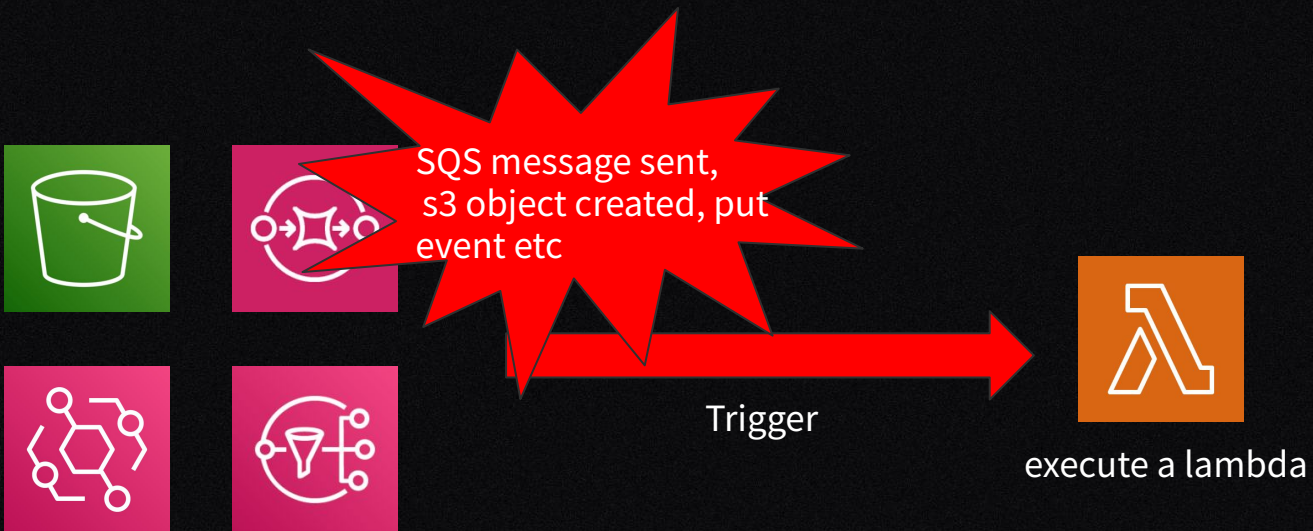


SQS message sent,
s3 object created, put
event etc...



다양한 Trigger 소스들

\$ Lambda 트리거란?



다양한 Trigger 소스들

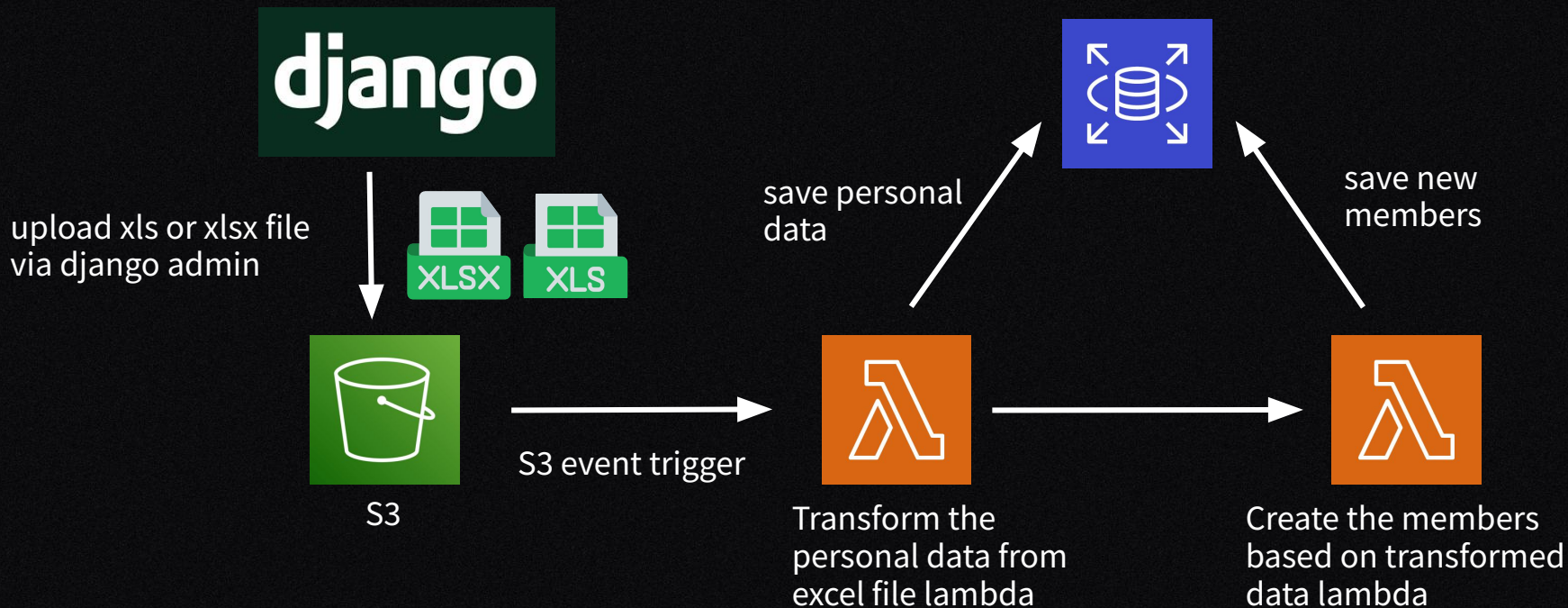
\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1




\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1


신규 입사자가 왔을 때,
개인정보가 갱신된 엑셀 파일을 사용해
회원 정보를 만드는 과정을 자동화 시켜야 한다.


\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1



\$

**make_dormitory_personal_info_table**

Layers (5)

S3

+ 트리거 추가

+ 대상 추가

마지막 수정
17일 전

함수 ARN
arn:aws:lambda:ap-northeast-2:313862985827:function:make_dormitory_personal_info_table

함수 URL [정보](#)
-

코드 | 테스트 | 모니터링 | **구성** | 별칭 | 버전

일반 구성

트리거

권한

대상

함수 URL

환경 변수

태그

VPC

모니터링 및 운영 도구

동시성

트리거 (1) [정보](#)

🔄

오류 수정

편집

삭제

트리거 추가

☐ 트리거

**S3: domtory-server**
arn:aws:s3:::domtory-server

▼ 세부 정보

Bucket: domtory-server

Event types: **s3:ObjectCreated:***

Notification: 9-992e-b9f00bc1a1fa

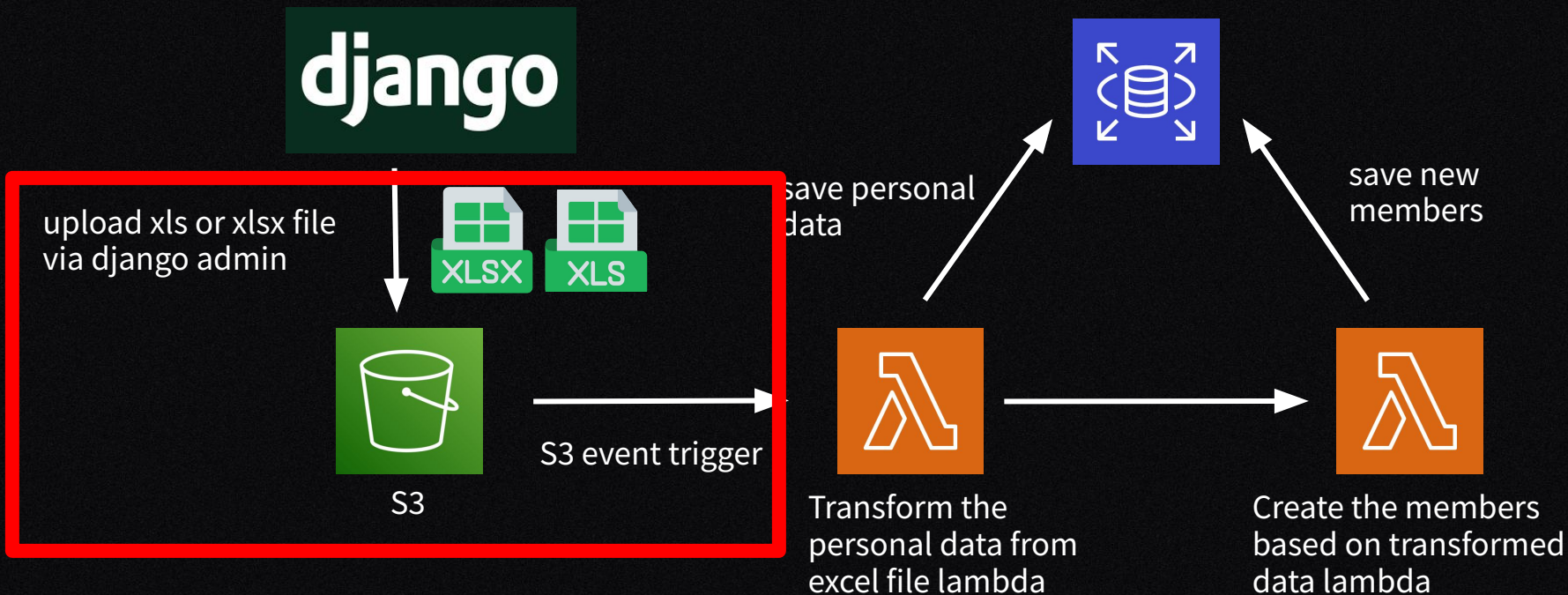
Prefix: personal_info_excel

Service principal: s3.amazonaws.com

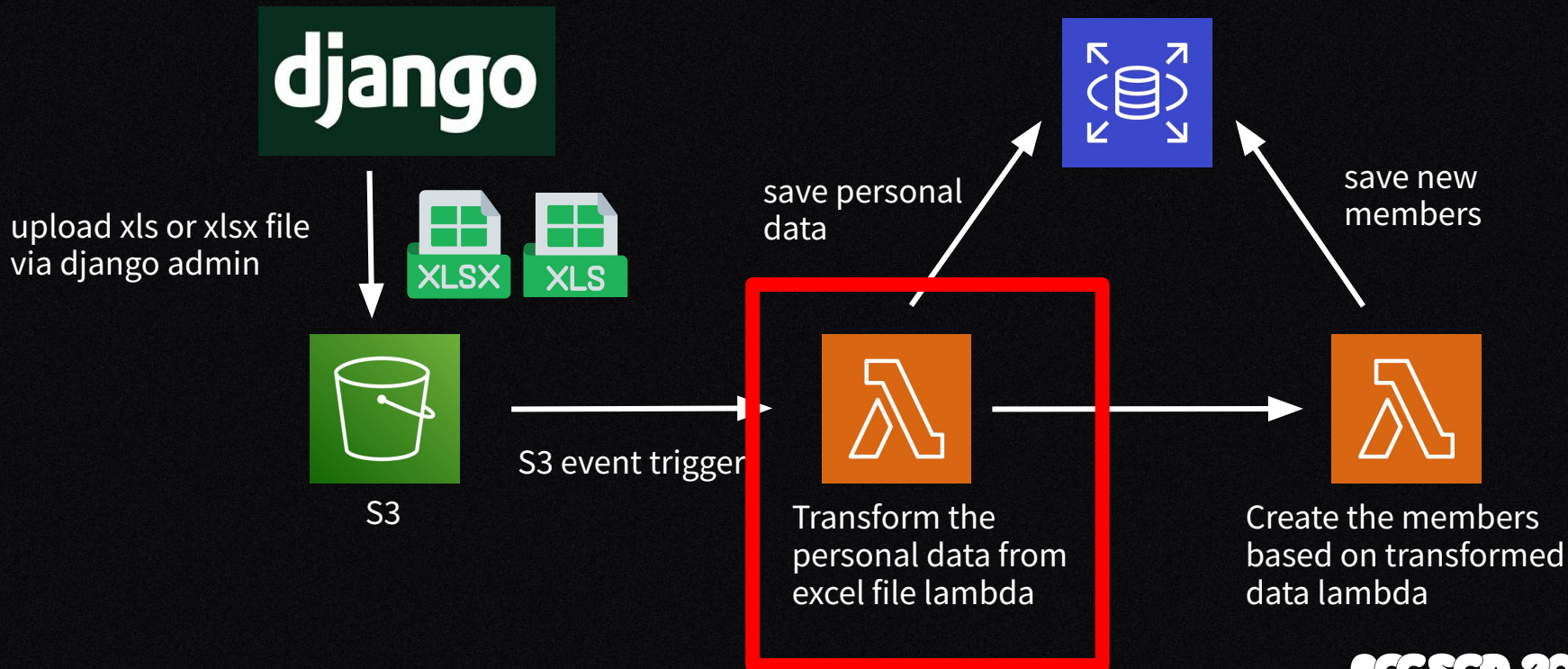
Source account: 313862985827

Statement ID: lambda-ed786139-88fd-4e35-876f-6a096265d80e

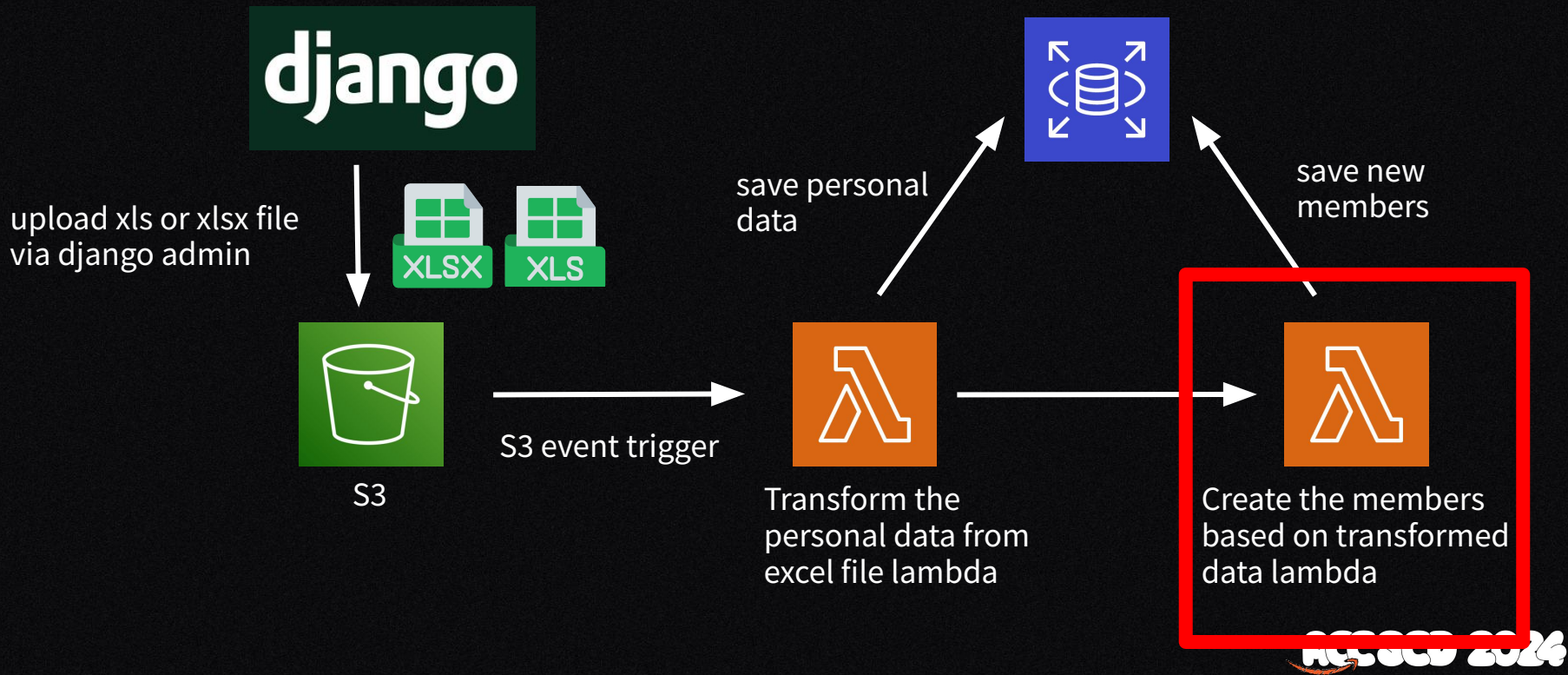
\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1



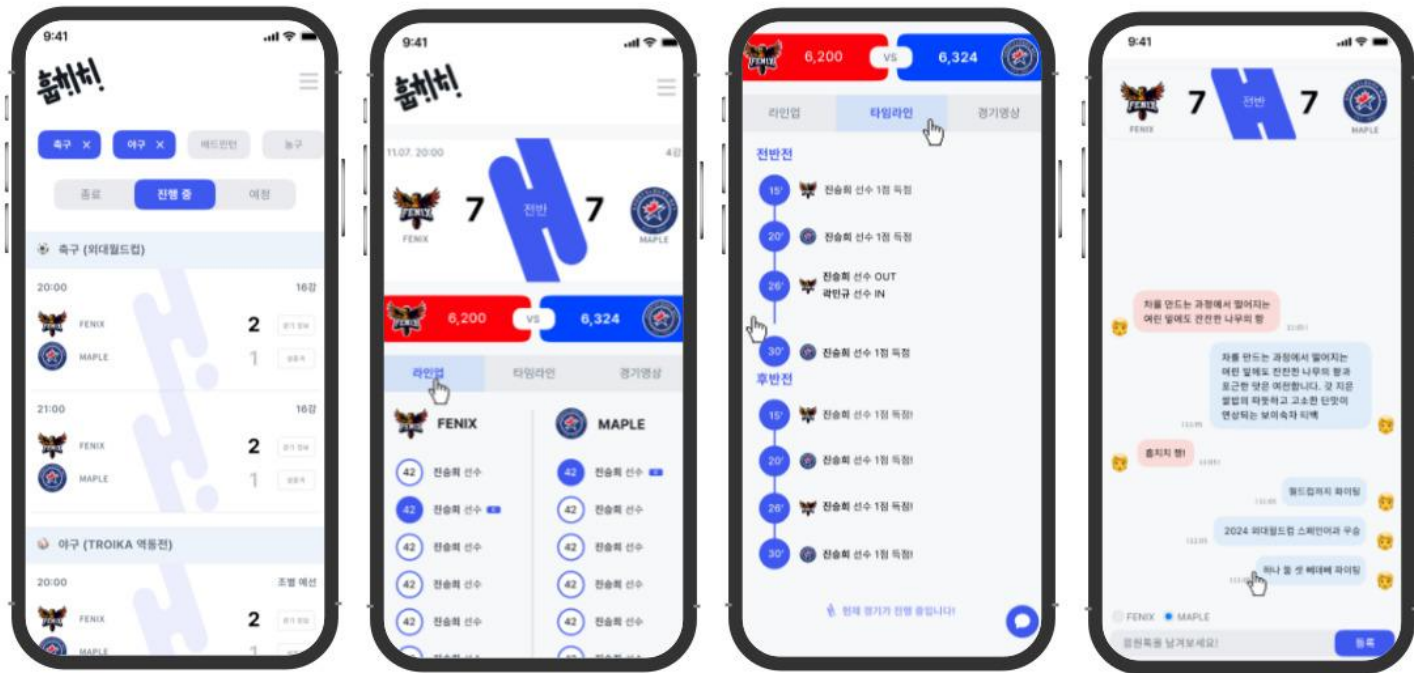
\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례1



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2

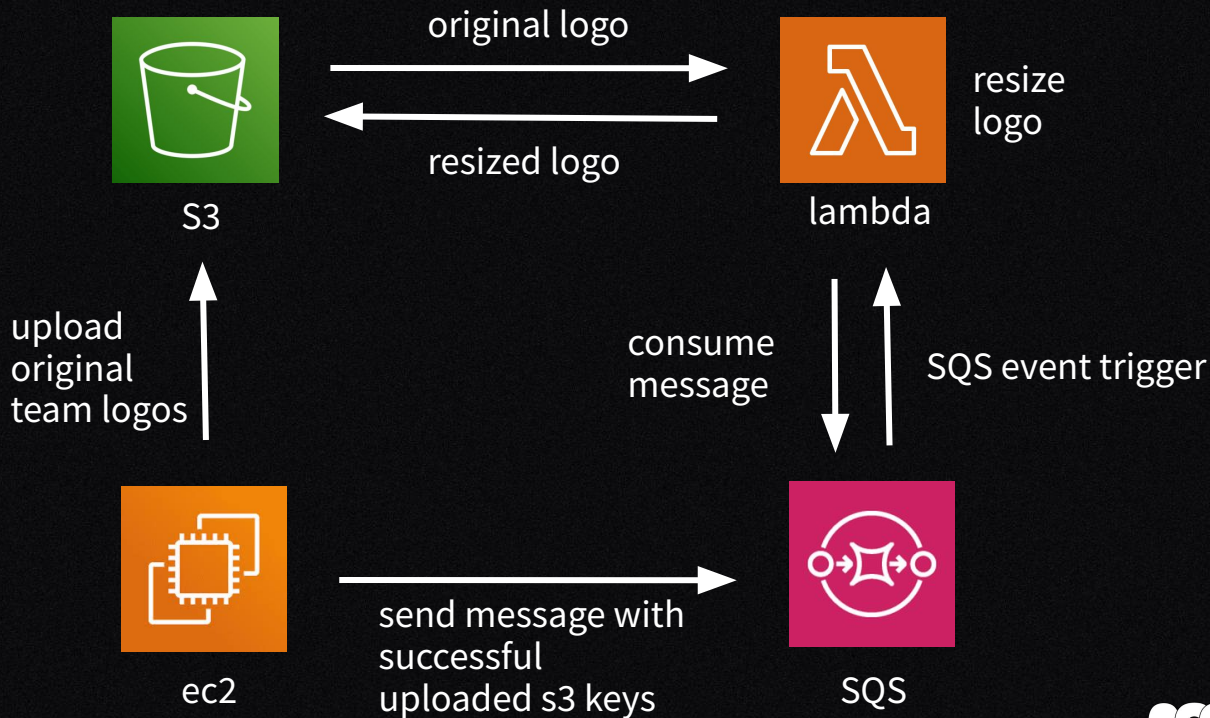
팀을 등록할 때 로고가 제각각이다.
확장자, 크기, 용량을 조정할 필요가 있다.
정상적으로 등록된 로고만 리사이징을 한다.

\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2


팀을 등록할 때 로고가 제각각이다.
확장자, 크기, 용량을 조정할 필요가 있다.
정상적으로 등록된 로고만 리사이징을 한다.


➡ S3 트리거를 사용한다면 팀 저장 실패시 **람다 리소스 낭비**


\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



\$

 make-light-logo

 Layers (1)

 SQS

+ 대상 추가


+ 트리거 추가

코드 | 테스트 | 모니터링 | **구성** | 별칭 | 버전

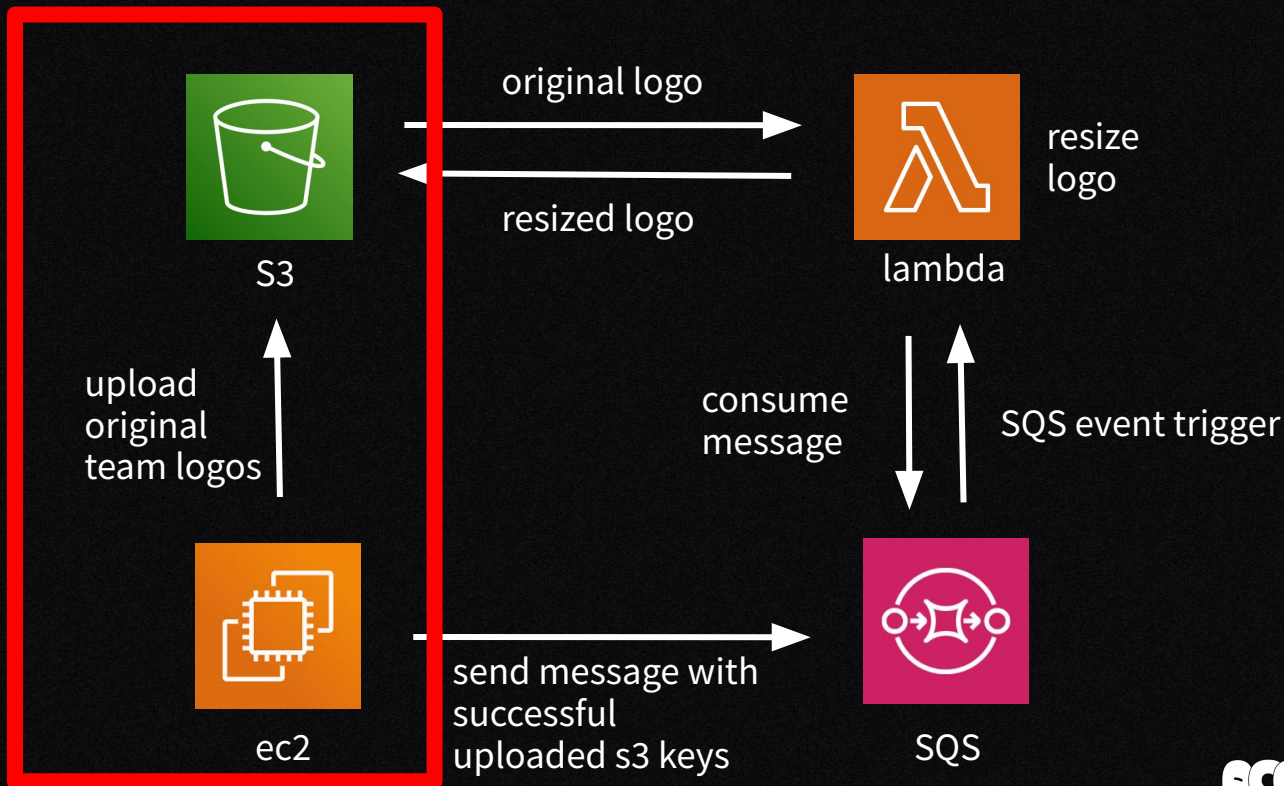
일반 구성 | **트리거** | 권한 | 대상 | 함수 URL | 환경 변수

트리거 (1) 정보

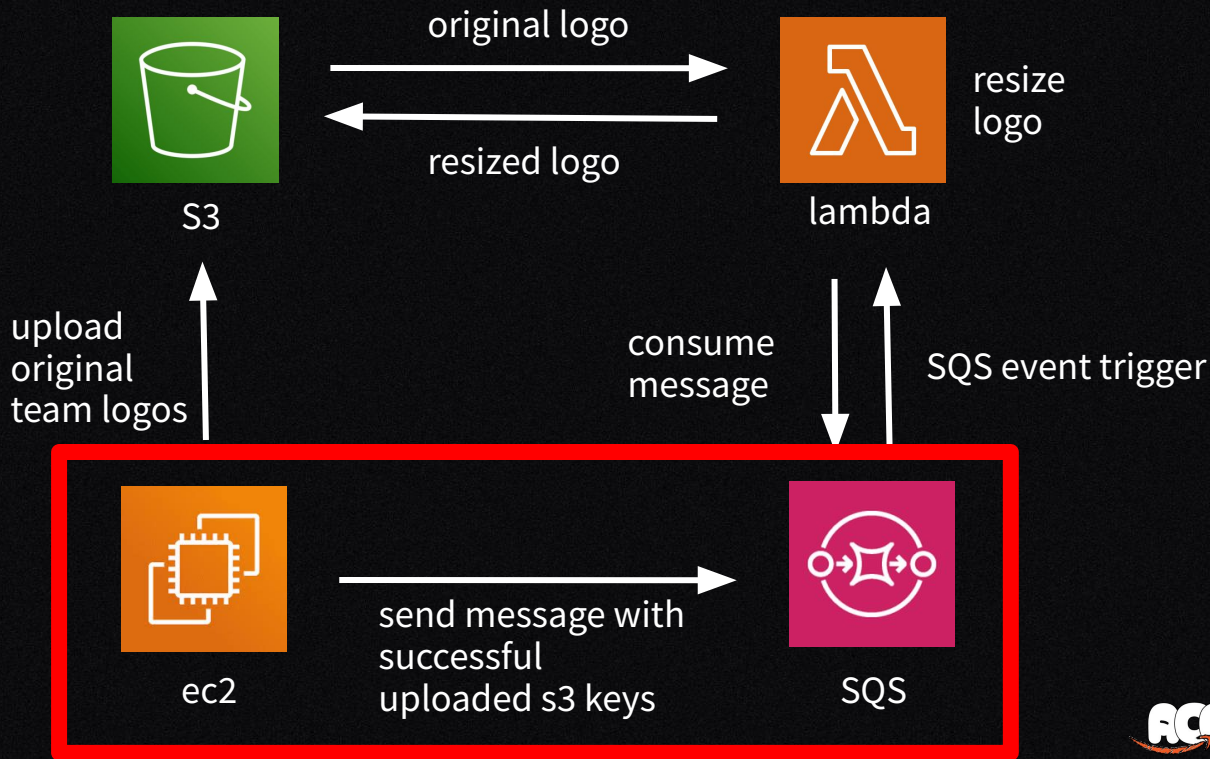
☐

 **SQS: [LogoLambdaQueue](#)**
arn:aws:sqs:ap-northeast-2:550581268183:LogoLambdaQueue
state: **Enabled**
▶ 세부 정보

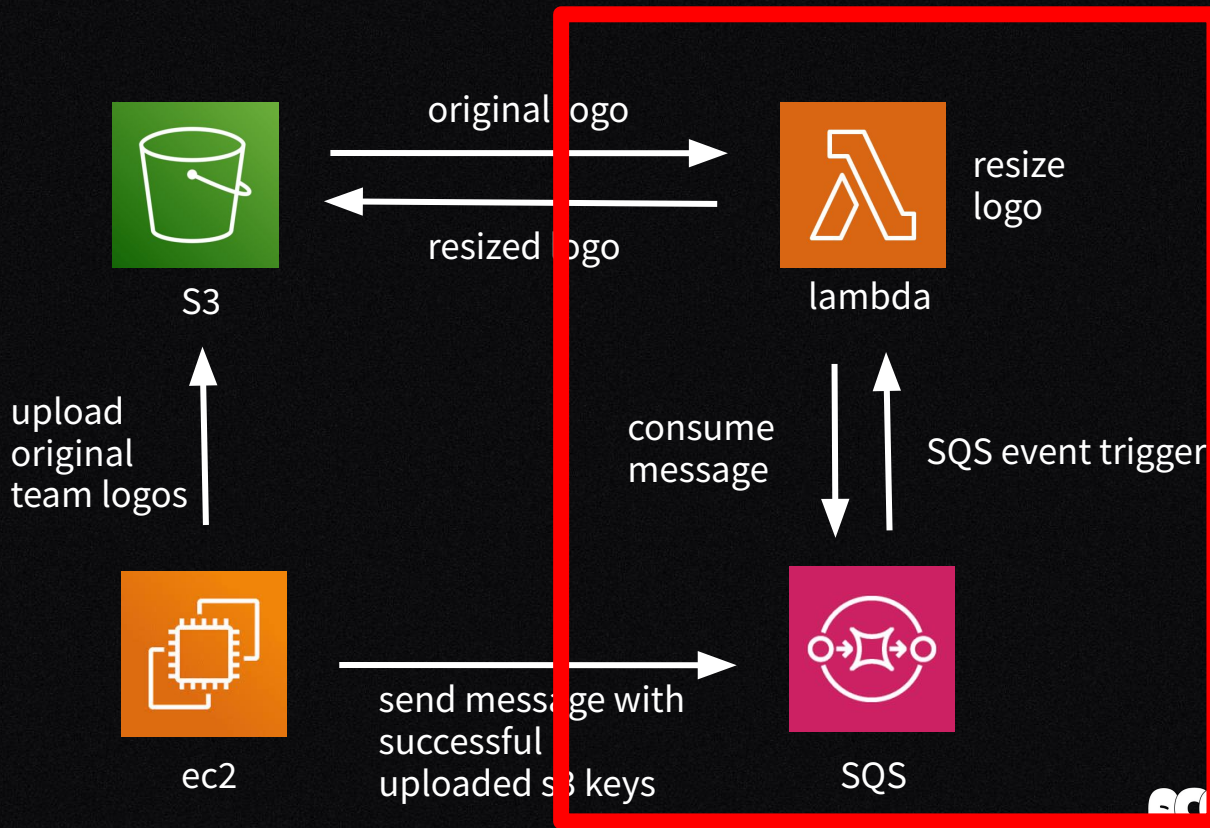
\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



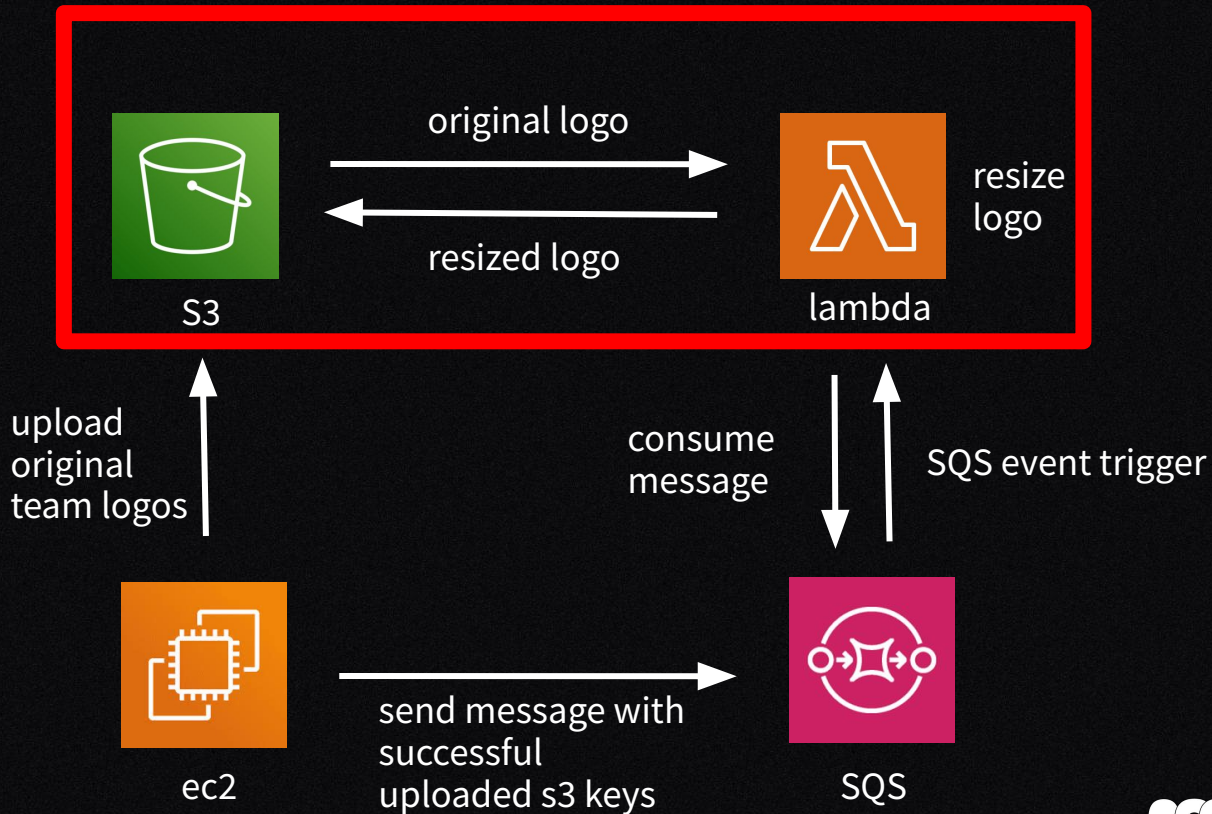
\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2

```
def lambda_handler(event, context):
    for record in event['Records']:
        s3_key = record['body']
        s3 = S3Connect()
        try:
            original_logo = s3.get_s3_image_object(s3_key)
            original_format = original_logo.format

            new_logo = process_logo(original_logo, original_format)
            out_img = io.BytesIO()
            new_logo.save(out_img, format=original_format, quality=70)
            out_img.seek(0)
            s3.put_s3_image_object(out_img, s3_key)
        except Exception as e:
            print("오류:{}, S3키:{}".format(e, s3_key))
    return {
        'statusCode': 200
    }
```

\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2

```
def lambda_handler(event, context):  
    for record in event['Records']:  
        s3_key = record['body']  
        s3 = S3Connect()  
        try:  
            original_logo = s3.get_s3_image_object(s3_key)  
            original_format = original_logo.format  
  
            new_logo = process_logo(original_logo, original_format)  
            out_img = io.BytesIO()  
            new_logo.save(out_img, format=original_format, quality=70)  
            out_img.seek(0)  
            s3.put_s3_image_object(out_img, s3_key)  
        except Exception as e:  
            print("오류:{}, S3키:{}".format(e, s3_key))  
    return {  
        'statusCode': 200  
    }
```

\$ 람다의 트리거를 사용한 대표적인 사용 방법: 사례2



용량: 211KB -> 128KB
크기: 500x500 -> 400x400



용량: 4.8M -> 17KB
크기: 5217x3418 -> 400x400



Photo by @canmandawe on Unsplash



용량: 28KB -> 13KB
크기: 512x768 -> 400x400



\$

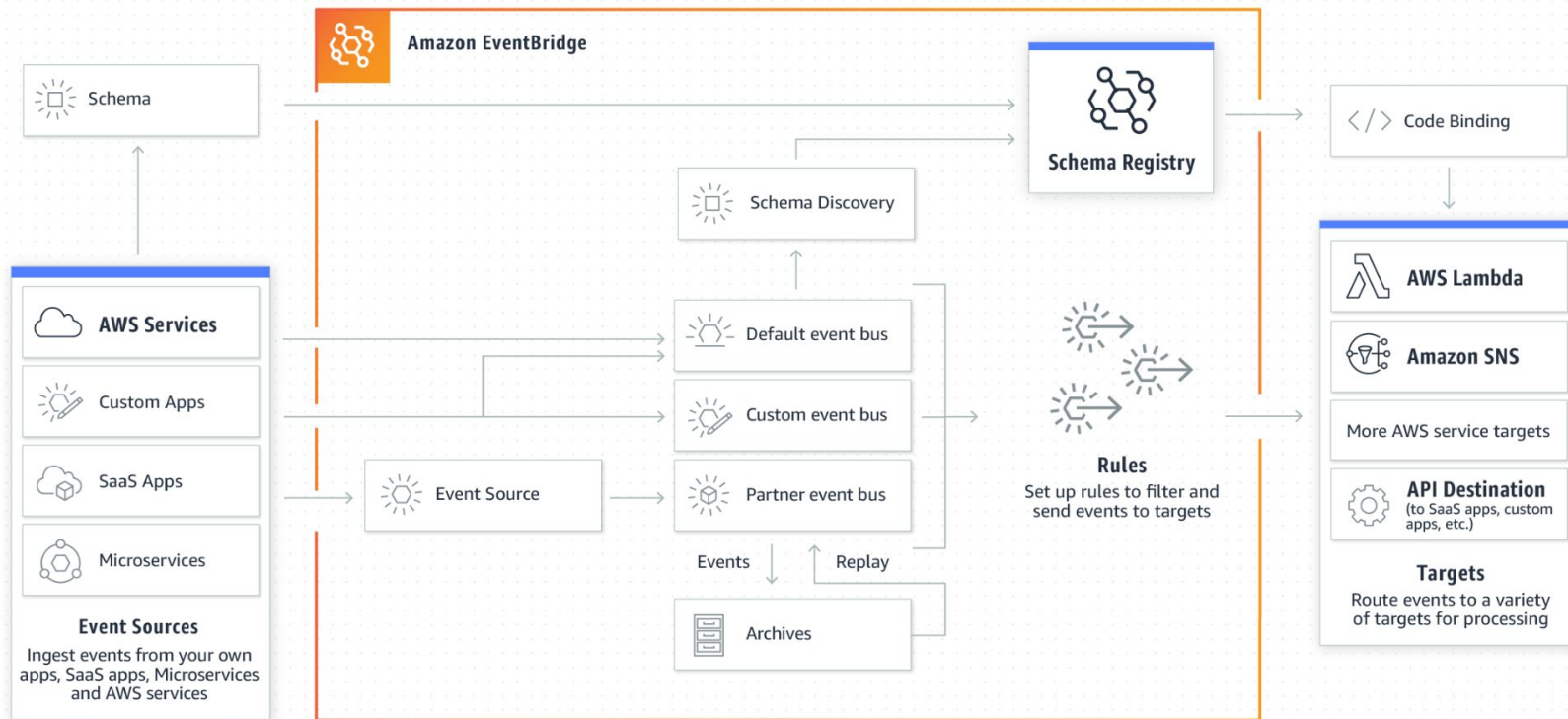
3. EventBridge는 어디에 쓰이는 걸까?

\$ EventBridge는 어디에 쓰이는 걸까?

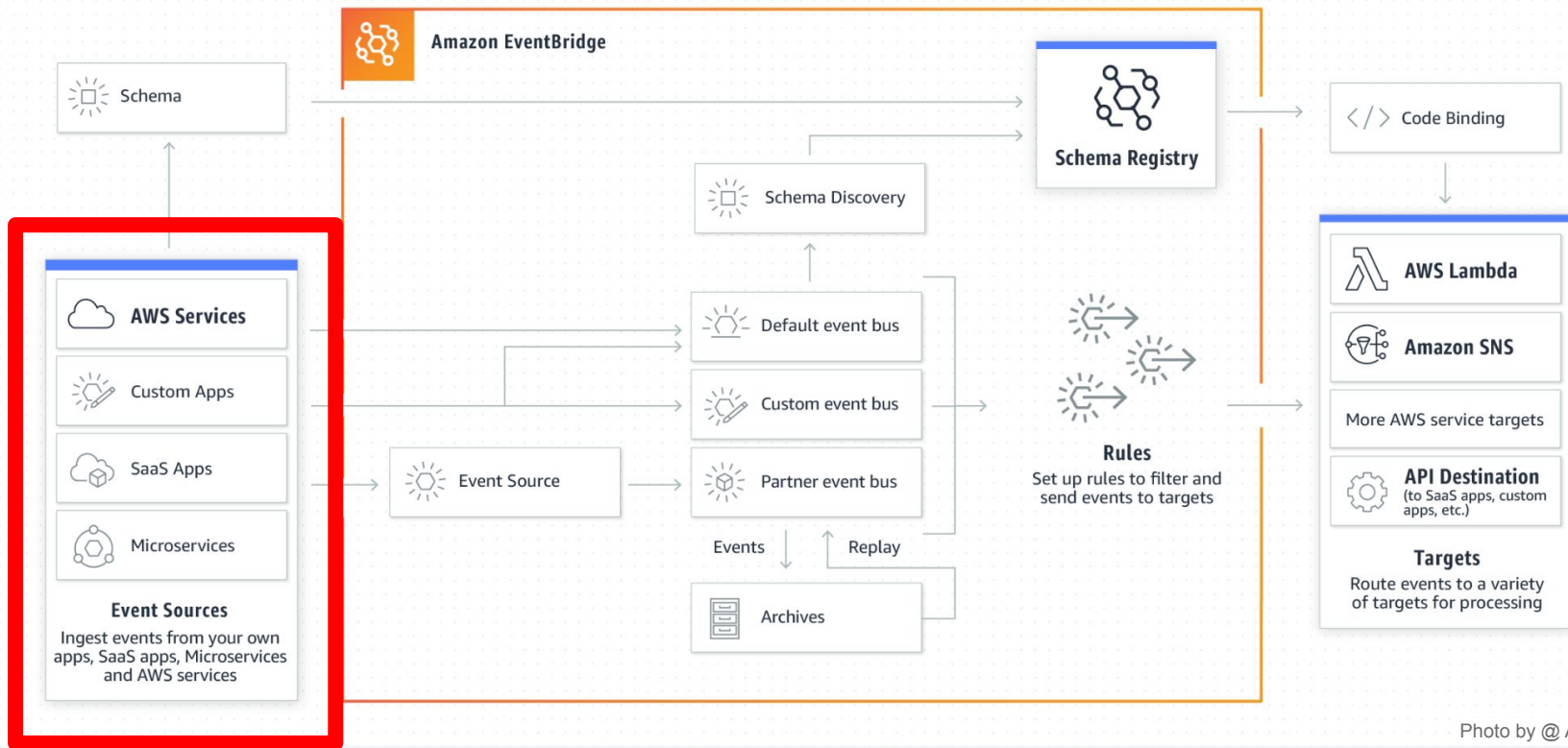


이벤트를 사용하여 애플리케이션 구성 요소를
서로 연결하는 서버리스 서비스

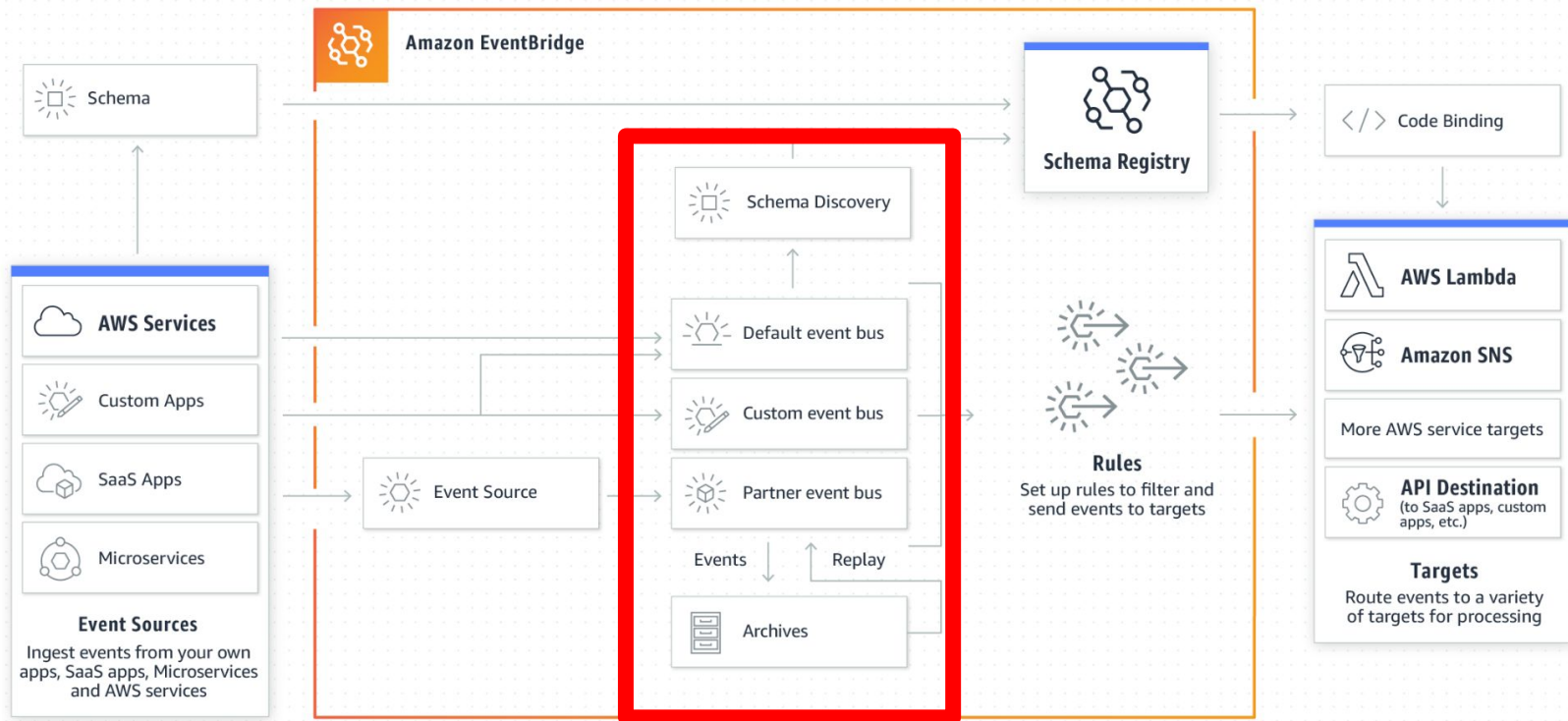
EventBridge는 어디에 쓰이는 걸까?



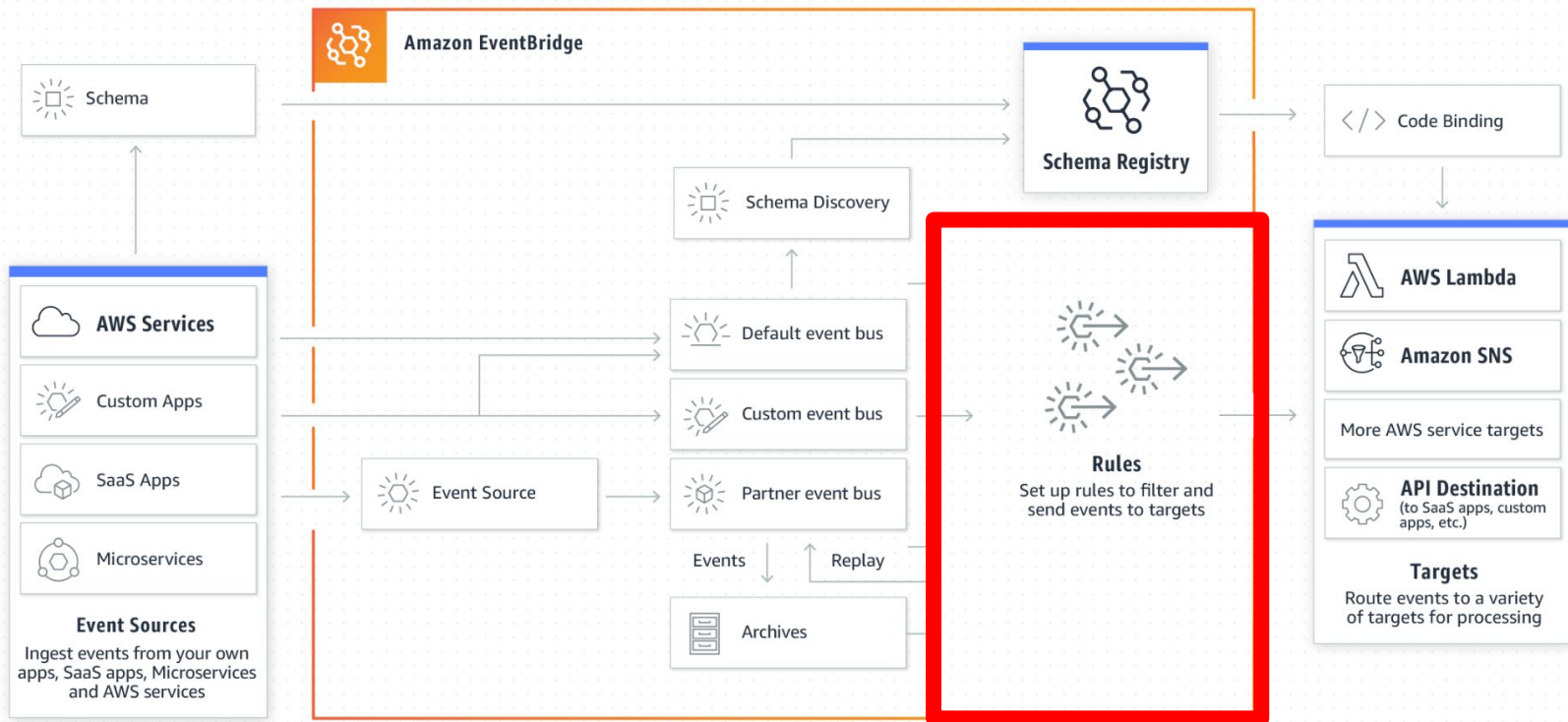
\$ EventBridge는 어디에 쓰이는 걸까?



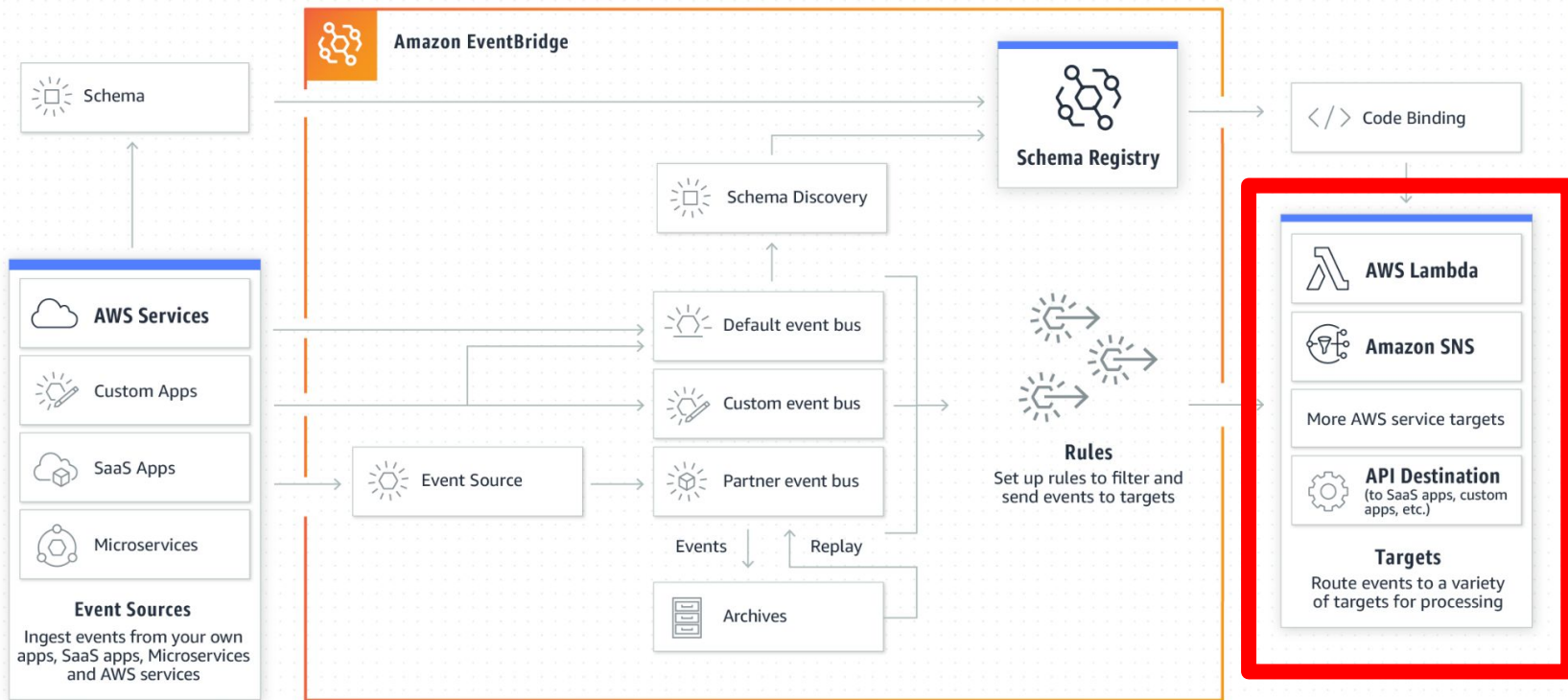
\$ EventBridge는 어디에 쓰이는 걸까?



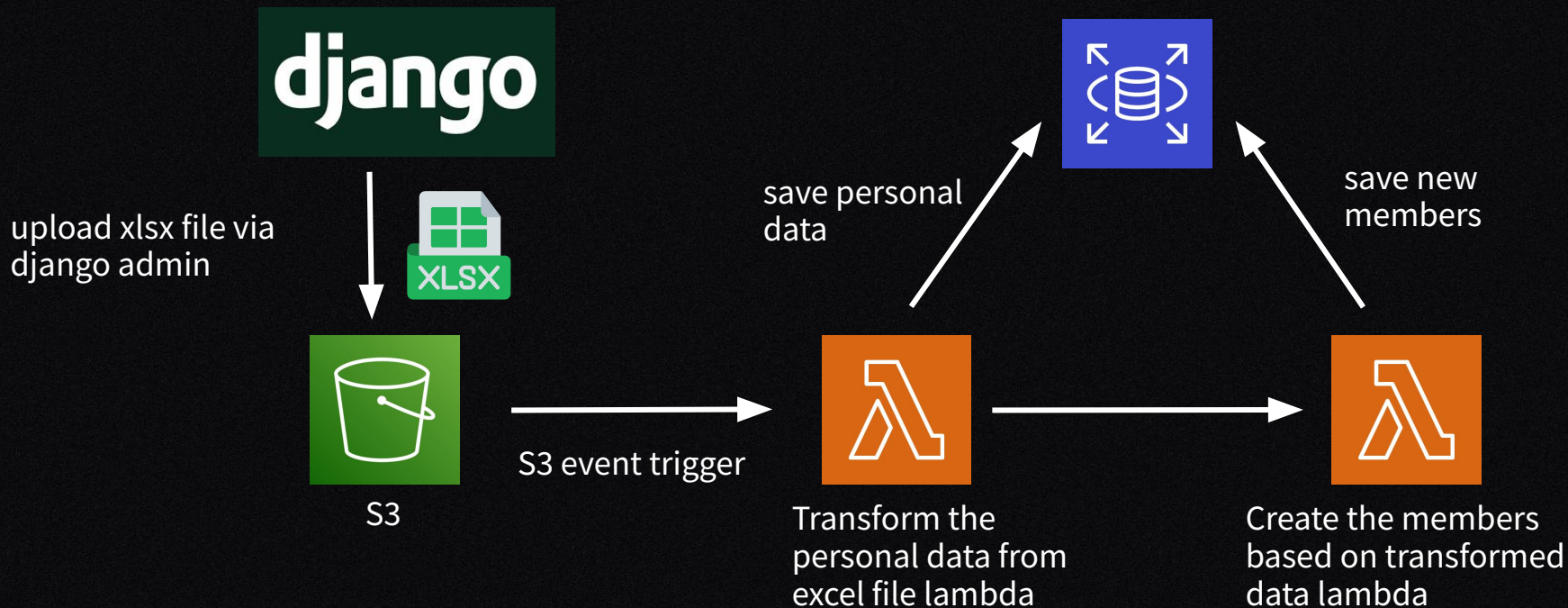
\$ EventBridge는 어디에 쓰이는 걸까?



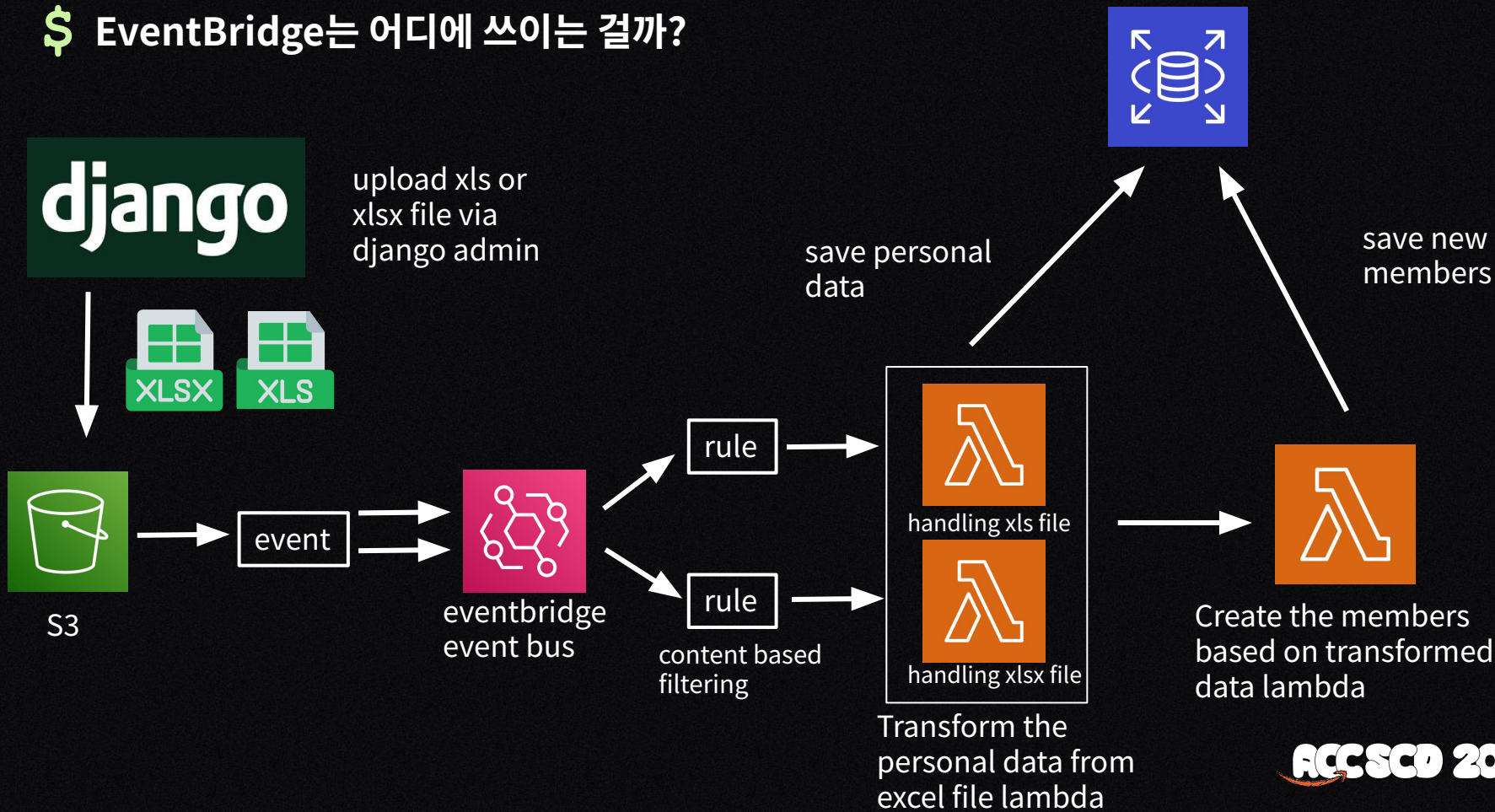
\$ EventBridge는 어디에 쓰이는 걸까?



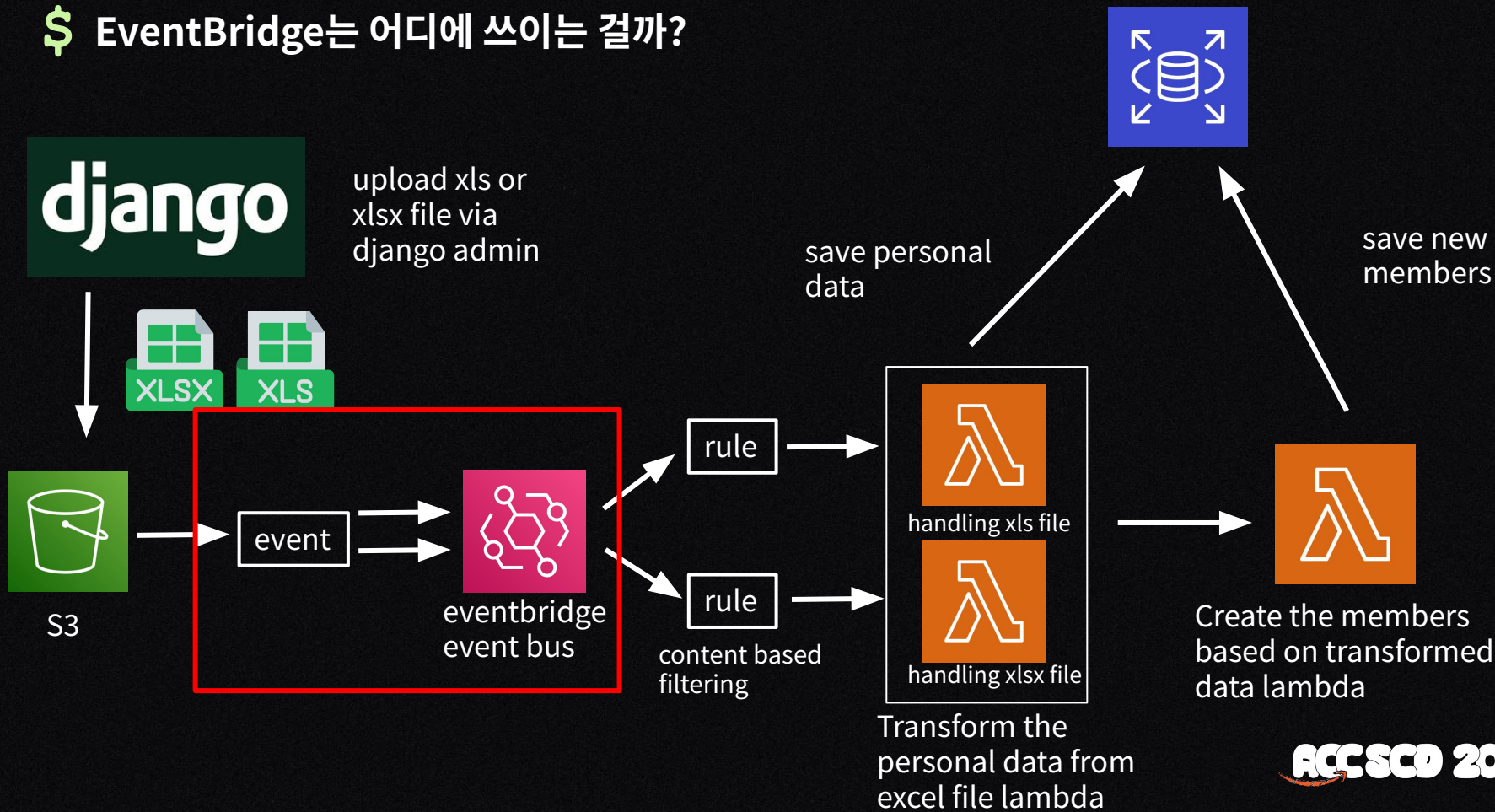
\$ EventBridge는 어디에 쓰이는 걸까?



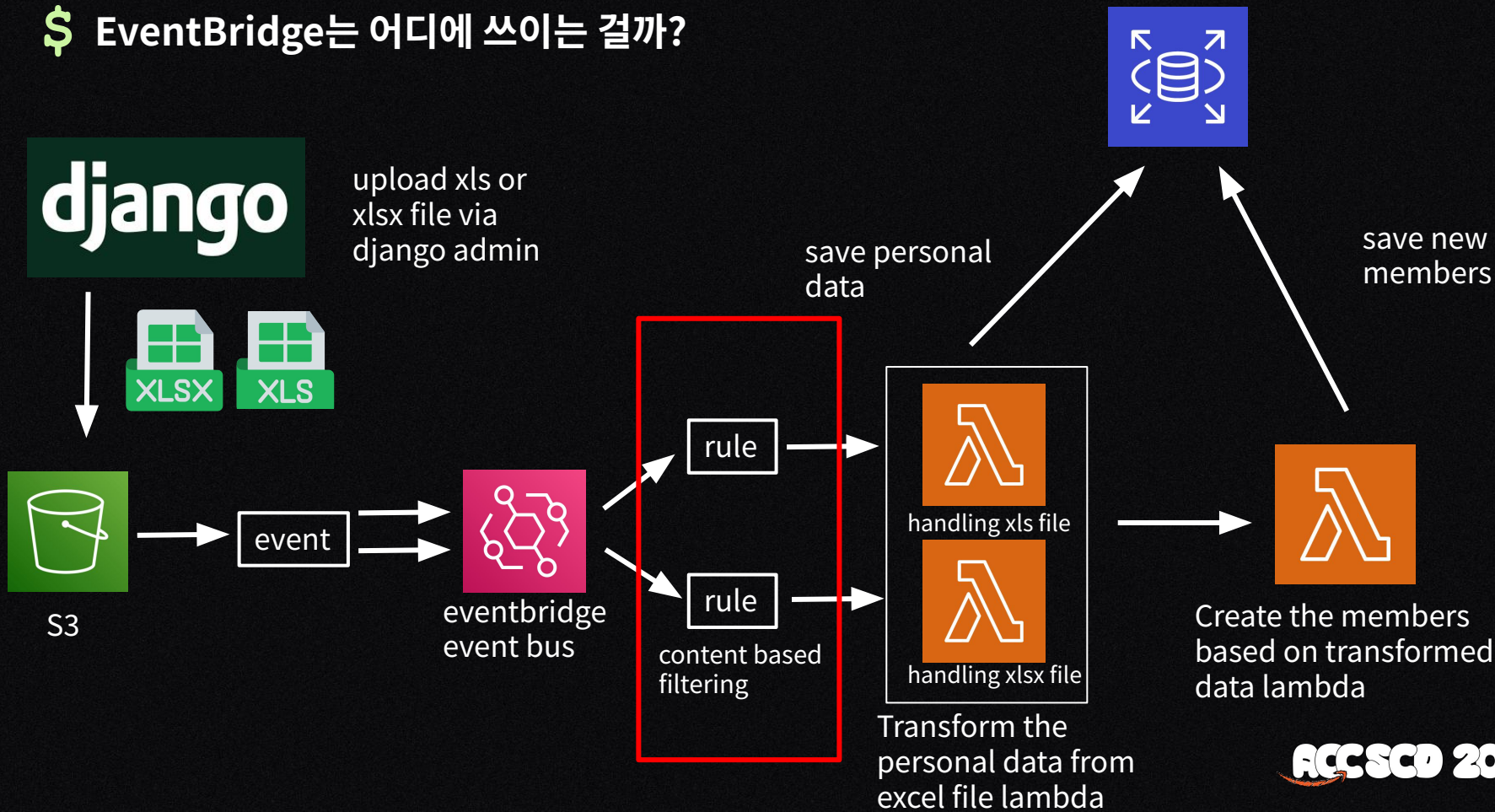
\$ EventBridge는 어디에 쓰이는 걸까?



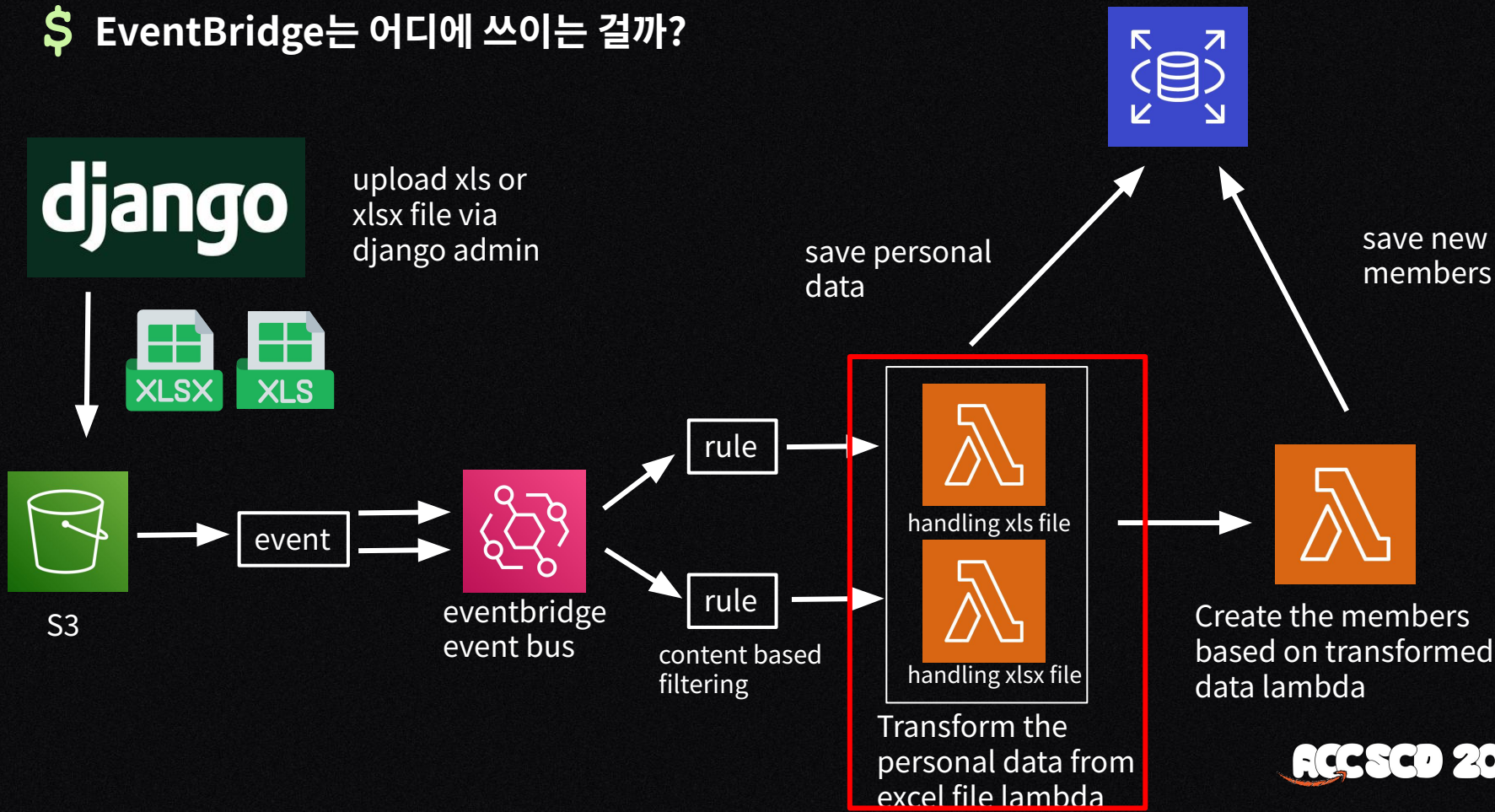
\$ EventBridge는 어디에 쓰이는 걸까?



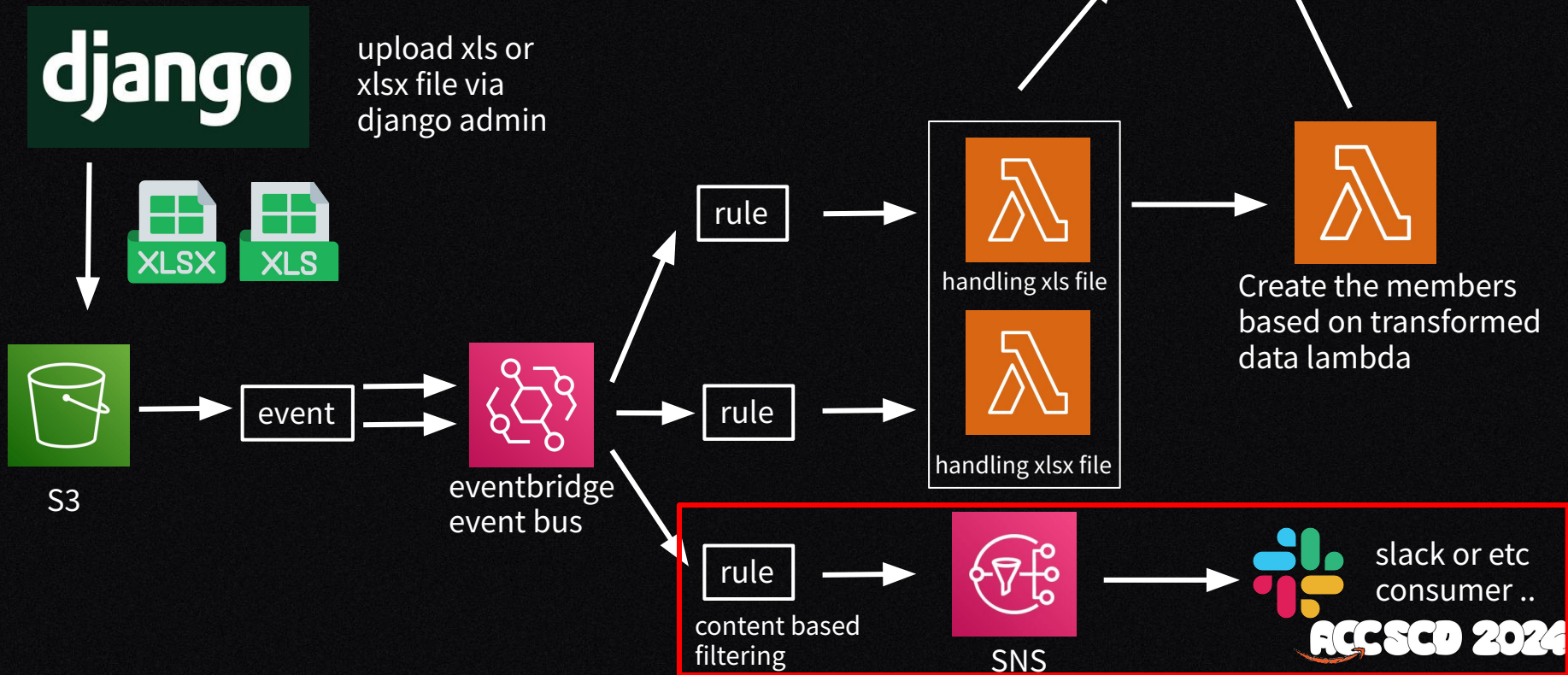
\$ EventBridge는 어디에 쓰이는 걸까?



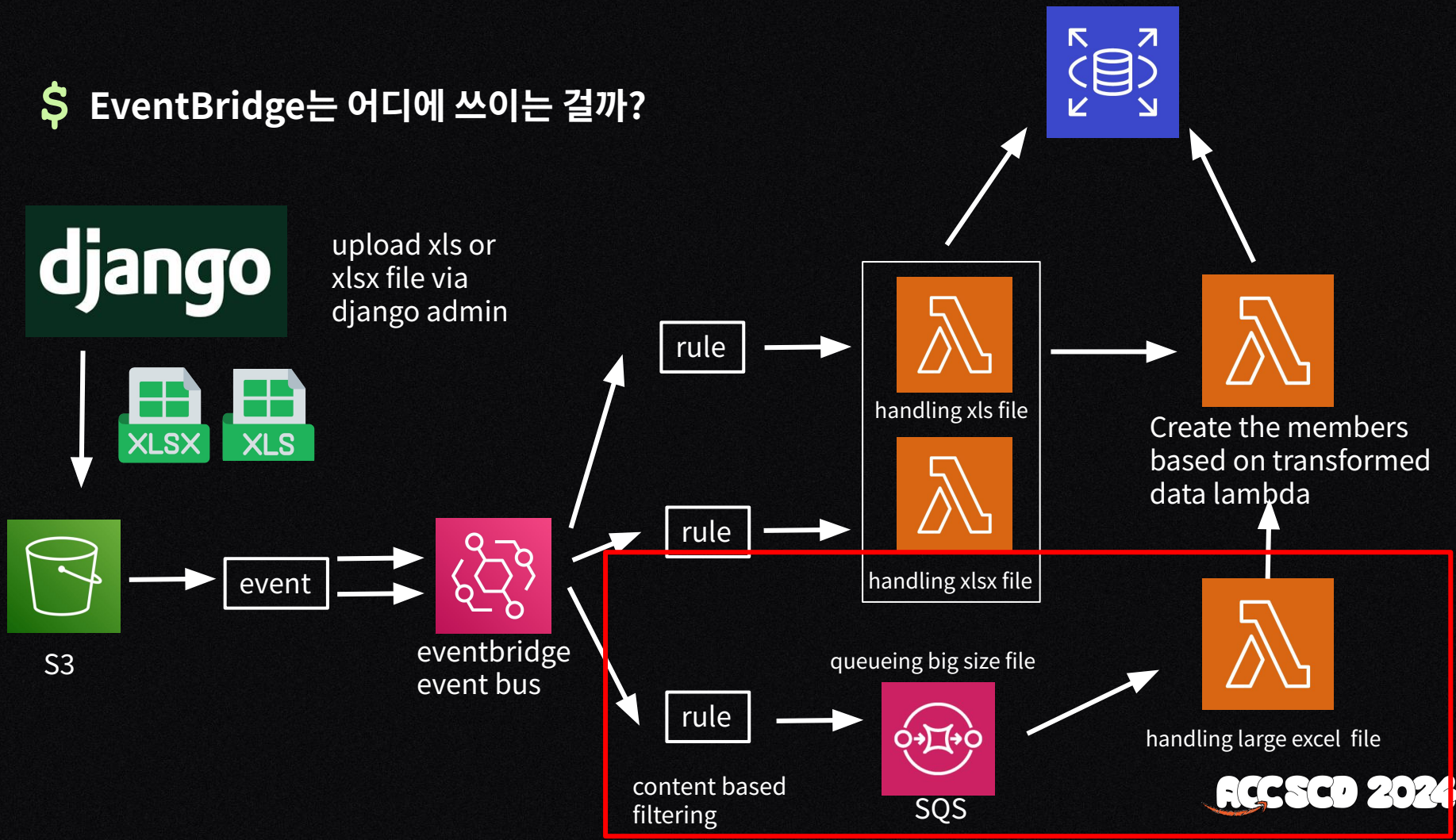
\$ EventBridge는 어디에 쓰이는 걸까?



\$ EventBridge는 어디에 쓰이는 걸까?



\$ EventBridge는 어디에 쓰이는 걸까?



\$ EventBridge는 어디에 쓰이는 걸까?



EventBridge Scheduler

\$ EventBridge는 어디에 쓰이는 걸까?



EventBridge Scheduler

- 다양한 AWS 서비스들과 연동 가능한 스케줄러
- Cron 기반 or 특정 주기로 호출

\$ EventBridge는 어디에 쓰이는 걸까?

▼ 버스

이벤트 버스

규칙

글로벌 엔드포인트

아카이브

재생

▼ 파이프

파이프

▼ Scheduler

일정

일정 그룹

▼ 통합

파트너 이벤트 소스

API 대상

▼ 스키마 레지스트리

스키마

일정 패턴

발생 정보

일회성 또는 반복 일정을 정의할 수 있습니다.

☐ 일회성 일정

☒ 반복 일정

시간대

일정의 시간대입니다.

(UTC+09:00) Asia/Seoul

일정 유형

요구 사항에 가장 잘 맞는 일정 유형을 선택합니다.

☒ Cron 기반 일정

매월 첫 번째 월요일 오전 8시(PST)와 같이 특정 시간에 실행되는 Cron 표현식을 사용하여 설정된 일정입니다.

☐ Rate 기반 일정

10분마다와 같이 일정한 빈도로 실행되는 일정입니다.

Cron 표현식 정보

일정에 대한 cron 표현식 정의

복사

지우기

cron

(

분

시간

일

월

요일

연도

)

\$ EventBridge는 어디에 쓰이는 걸까?

일정 유형

이 화면에 가장 잘 맞는 일정 유형을 선택합니다.

☐ Cron 기반 일정

매월 첫 번째 월요일 오전 8시(PST)와 같이 특정 시간에 실행되는 Cron 표현식을 사용하여 설정된 일정입니다.

☒ Rate 기반 일정

10분마다와 같이 일정한 빈도로 실행되는 일정입니다.

Rate 표현식 [정보](#)

일정을 실행할 값과 시간 단위를 입력합니다.

rate ()

값 단위

유연한 기간

유연한 기간을 선택하면 Scheduler는 지정한 시간 범위 내에서 일정을 호출합니다. 예를 들어 15분을 선택하면 스케줄이 시작 시간으로부터 15분 이내에 실행됩니다.

\$ EventBridge는

2/4단계

대상 선택

대상 세부 정보

대상 API 정보

일정의 대상으로 호출할 API를 선택합니다.

☒ 템플릿 형식의 대상

☐ 모든 API



CodeBuild



StartBuild



CodePipeline



StartPipel...



Amazon ECS



RunTask



Amazon EventBridge



PutEvents



Kinesis Data Firehose



PutRecord



Amazon Inspector V1



StartAssess...



Kinesis Data Streams



PutRecord



AWS Lambda



Invoke



SageMaker



StartPipel...



AWS Step Functions



StartExecuti...



Amazon SNS



Publish



Amazon SQS



SendMessage

대상 세부 정보

대상 API 정보

일정의 대상으로 호출할 API를 선택합니다.

☐ 템플릿 형식의 대상

☒ 모든 API

모든 AWS 서비스

🔍 서비스 찾기

< 1 2 3 4 5 6 7 8 ... >



Amazon A2I (3)



API Gateway
V1 (74)



API Gateway
V2 (46)



AWS Account
Management (5)



AmazonConnectCa
mpaignService
(15)



AWS Amplify (22)



Amplify
Backend (22)



Amplify UI
Builder (16)



AWS App
Mesh (23)

\$ EventBridge는

2/4단계

대상 선택

대상 세부 정보

대상 API 정보

일정의 대상으로 호출할 API를 선택합니다.

☒ 템플릿 형식의 대상

☐ 모든 API



CodeBuild



StartBuild



CodePipeline



StartPipel...



Amazon ECS



RunTask



Amazon EventBridge



PutEvents



Kinesis Data Firehose



PutRecord



Amazon Inspector V1



StartAssess...



Kinesis Data Streams



PutRecord



AWS Lambda



Invoke



SageMaker



StartPipel...



AWS Step Functions



StartExecuti...



Amazon SNS



Publish



Amazon SQS



SendMessage

\$ EventBridge는 어디에 쓰이는 걸까?: 사례1

24.03.17 (일)

조식
잡곡밥, 소고기미역국, 새우까스*소스, 푸실리카찜조림, 갯손나물볶음, 배추김치, 모닝빵*말기잼*버터, 우유*시리얼

중식
잡곡밥, 차돌면장찌개, 코다리양념조림, 계란맛살전, 시금치나물무침, 배추김치, 과일

석식
잡곡밥, 육개장, 돈시태떡볶, 열대과일샐러드, 참나물무침, 배추김치

24.03.18 (월)

조식
잡곡밥, 계란파국, 소불고기, 어묵

중식
잡곡밥, 감자미역국, 돈수육, 반무

석식
잡곡밥, 사례기된장국, 닭볶음탕

공지사항

• 2024년 1학기 학습동아리 신청	2024-03-12
• 3월 사설점검 실시	2024-03-08
• 3월 사설 및 공동구역 방역소독 실시	2024-03-06
• 심폐소생술 교육	2024-03-06



동토리

3.17(SUN)
아침

잡곡밥
소고기미역국
새우까스*소스
푸실리카찜조림
갯손나물볶음
배추김치
모닝빵*말기잼*버터
우유*시리얼

동토리

간략히 보기 X

오늘의 동토리 저녁 메뉴예요.... 어제 오후 6:23
잡곡밥, 물만두국, 우체피망볶음, 꽃빵,
청경채겉절이, 배추김치, 과일

오늘의 동토리 점심 메뉴예요.... 어제 오전 11:38
잡곡밥, 북어무국, 마나돈까스샐러드, 궁중떡볶이,
도토리묵무침, 배추김치

학사 공지사항

더 보기 >

2024-03-12
2024년 1학기 학습동아리 신청

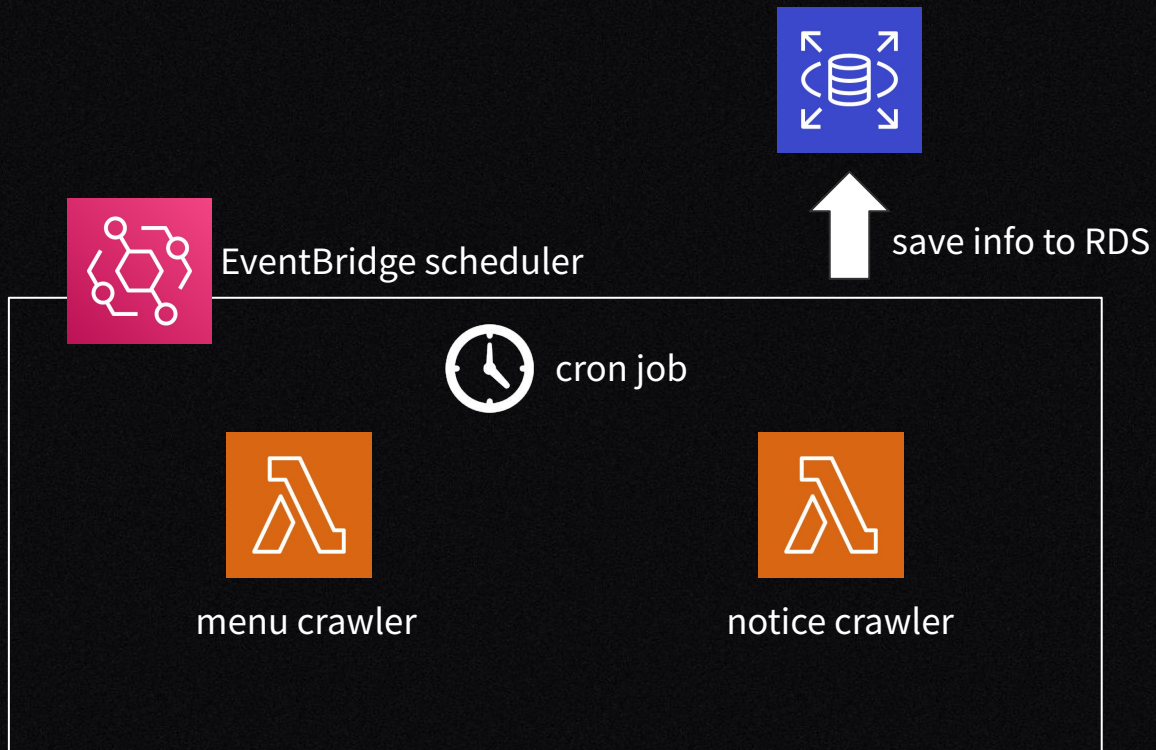
2024-03-08
3월 사설점검 실시

2024-03-06
3월 사설 및 공동구역 방역소독 실시

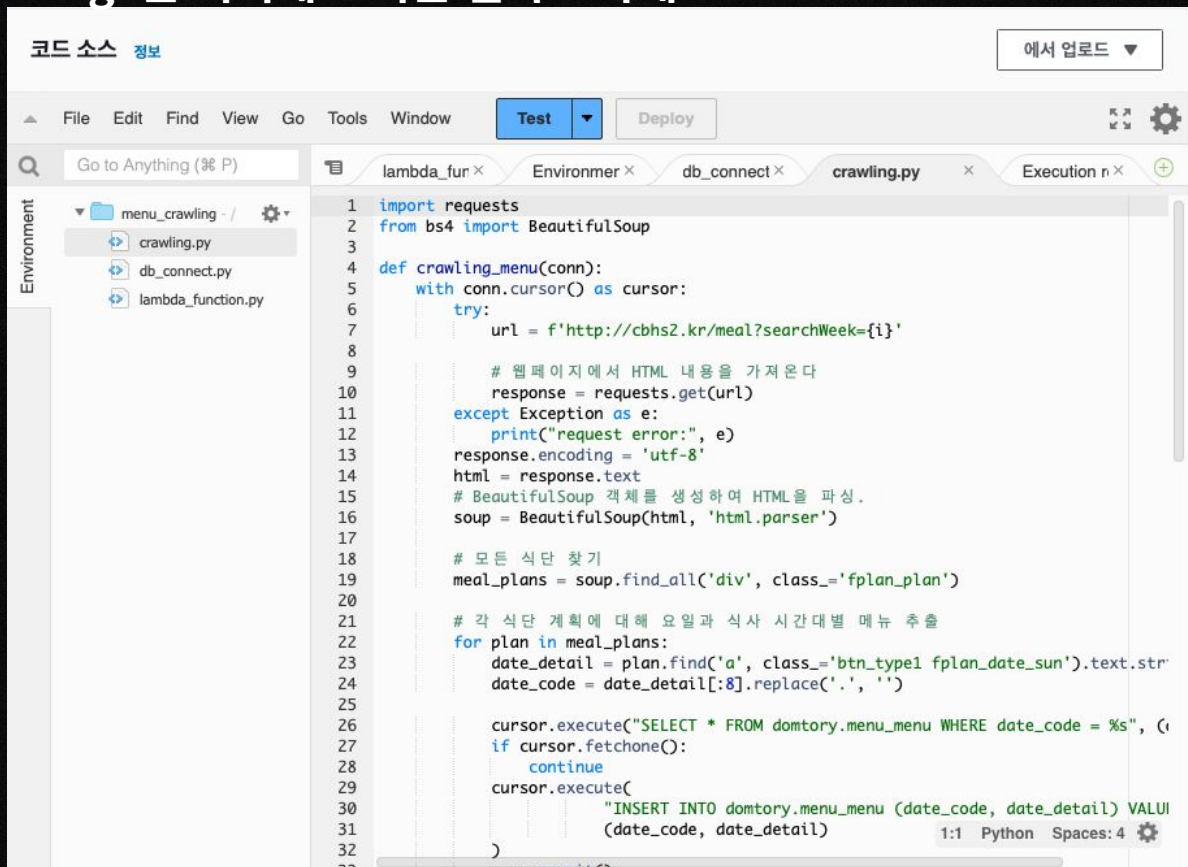
식단 메뉴와 공지사항을

충북학사 홈페이지에서 크롤링해와야 한다.

\$ EventBridge는 어디에 쓰이는 걸까?: 사례1



\$ EventBridge는 어디에 쓰이는 걸까?: 사례1



The screenshot shows a code editor interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'menu_crawling' with files 'crawling.py', 'db_connect.py', and 'lambda_function.py'. The code editor shows the 'crawling.py' file with the following Python code:

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 def crawling_menu(conn):
5     with conn.cursor() as cursor:
6         try:
7             url = f'http://cbhs2.kr/meal?searchWeek={i}'
8
9             # 웹 페이지에서 HTML 내용을 가져온다
10            response = requests.get(url)
11            except Exception as e:
12                print("request error:", e)
13            response.encoding = 'utf-8'
14            html = response.text
15            # BeautifulSoup 객체를 생성하여 HTML을 파싱.
16            soup = BeautifulSoup(html, 'html.parser')
17
18            # 모든 식단 찾기
19            meal_plans = soup.find_all('div', class_='fplan_plan')
20
21            # 각 식단 계획에 대해 요일과 식사 시간대별 메뉴 추출
22            for plan in meal_plans:
23                date_detail = plan.find('a', class_='btn_type1 fplan_date_sun').text.strip()
24                date_code = date_detail[:8].replace('.', '')
25
26                cursor.execute("SELECT * FROM domtory.menu_menu WHERE date_code = %s", (
27                    if cursor.fetchone():
28                        continue
29                    cursor.execute(
30                        "INSERT INTO domtory.menu_menu (date_code, date_detail) VALUES
31                        (date_code, date_detail)
32                )
33            conn.commit()
```

\$ Event

menu_crawling

일정 세부 정보

일정 이름
menu_crawling

설명
-

일정 그룹 이름
default

상태
🟢 **활성**

일정 ARN
arn:aws:scheduler:ap-northeast-2:313862985827:schedule/default/menu_crawling

완료 후 작업
NONE

일정

대상

제시도 정책

DLQ(Dead Letter Queue)

암호화

일정

Cron 표현식 정보

40 3 ? * * *
분 시간 일 월 요일 연도

Cron 표현식 복사

다음 10개의 트리거 날짜

날짜와 시간은 '2022년 11월 9일 수요일 09:00(UTC - 08:00)'처럼 선택

▼ 개발자 리소스

학습

샌드박스

Quick Starts

▼ 버스

이벤트 버스

규칙

글로벌 엔드포인트

아카이브

재생

▼ 파이프

파이프

▼ Scheduler

일정

일정 그룹

▼ 통합

파트너 이벤트 소스

API 대상

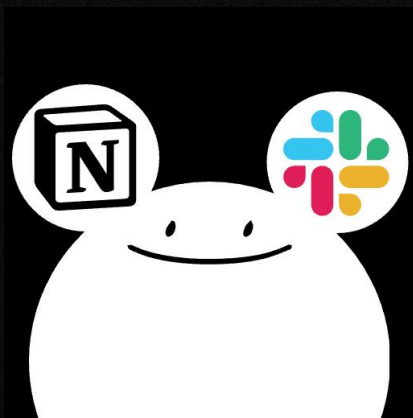
▼ 스키마 레지스트리


스키마

설명서 

\$ EventBridge는 어디에 쓰이는 걸까?: 사례2


회의록					
Aa 이름	📅 날짜	⌵ 확정여부	⌵ 참여 파트	⌵ 종류	⌵ 참여자
👤 정기 스크럼	2024년 3월 8일 오후 10:00	확정	전체	비대면	👤 [redacted]
👤 정기 스크럼	2024년 3월 1일 오후 10:00	확정	전체	비대면	👤 [redacted]




**회의봇** 🗓️ 오후 7:00

🔔 1월 2일 오후 7시 0분에 "첫 3차 릴리즈 회의"이/가 예정되어있어요! 잊지 마세요.
회의 타입: **대면**
회의 노션페이지: [바로가기](#)
참여자: @ [redacted] @이현재

1월 2일 화요일 ▾

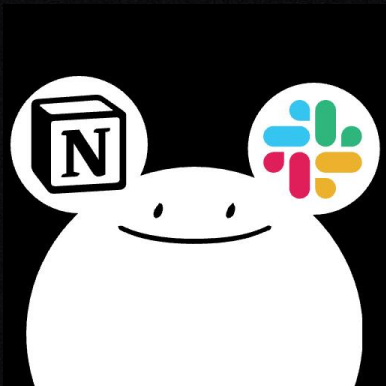
**회의봇** 🗓️ 오후 2:00

🔔 오늘 "첫 3차 릴리즈 회의"이/가 있다는 거 잊지 않으셨죠? 잊지 말아주세요!
회의 타입: **대면**
회의 노션페이지: [바로가기](#)
참여자: @ [redacted] @이현재

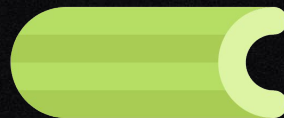
**회의봇** 🗓️ 오후 6:50

!!! 곧 10분 뒤에 "첫 3차 릴리즈 회의" 가 시작돼요! 모두 준비해주세요.
회의 노션페이지: [바로가기](#)
참여자: @ [redacted] @이현재

\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

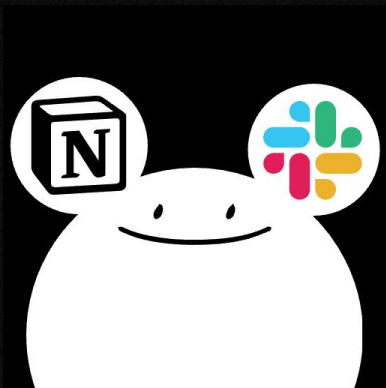


fastapi

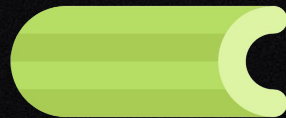


celery

\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



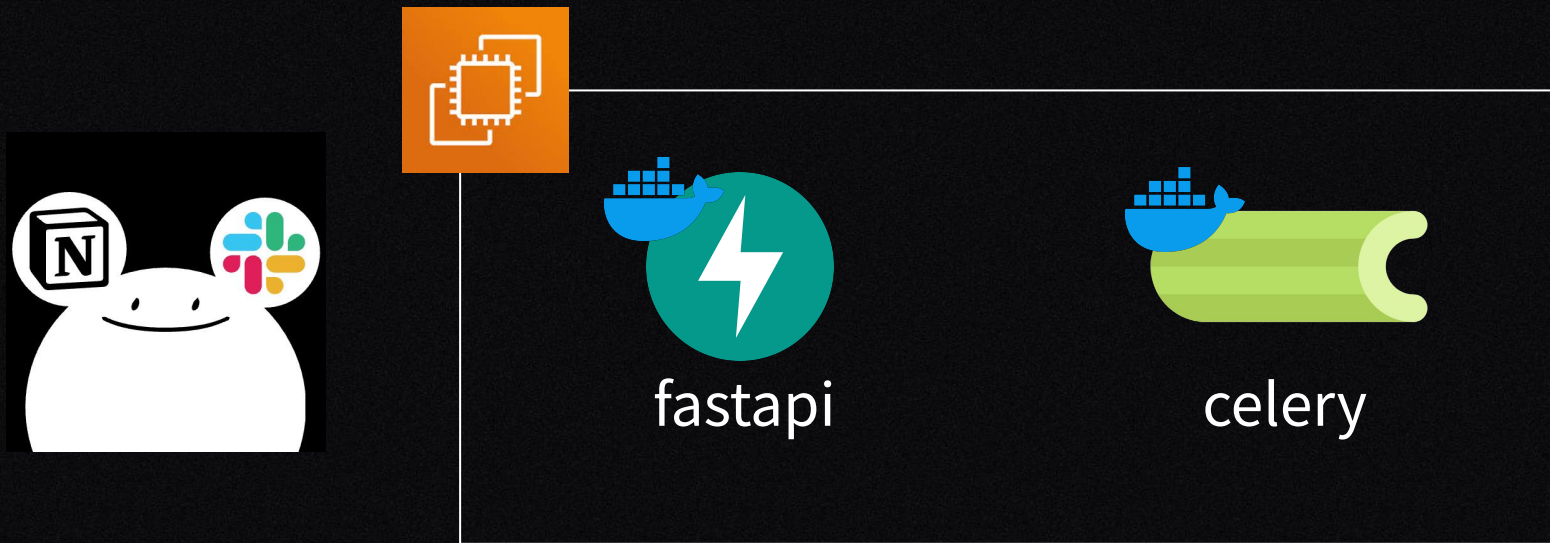
fastapi



celery

매주 예정된 회의 페이지를 노선에 자동으로 추가해야 한다.
단, 회의 시간이나 템플릿의 변화에 유연해야 한다.

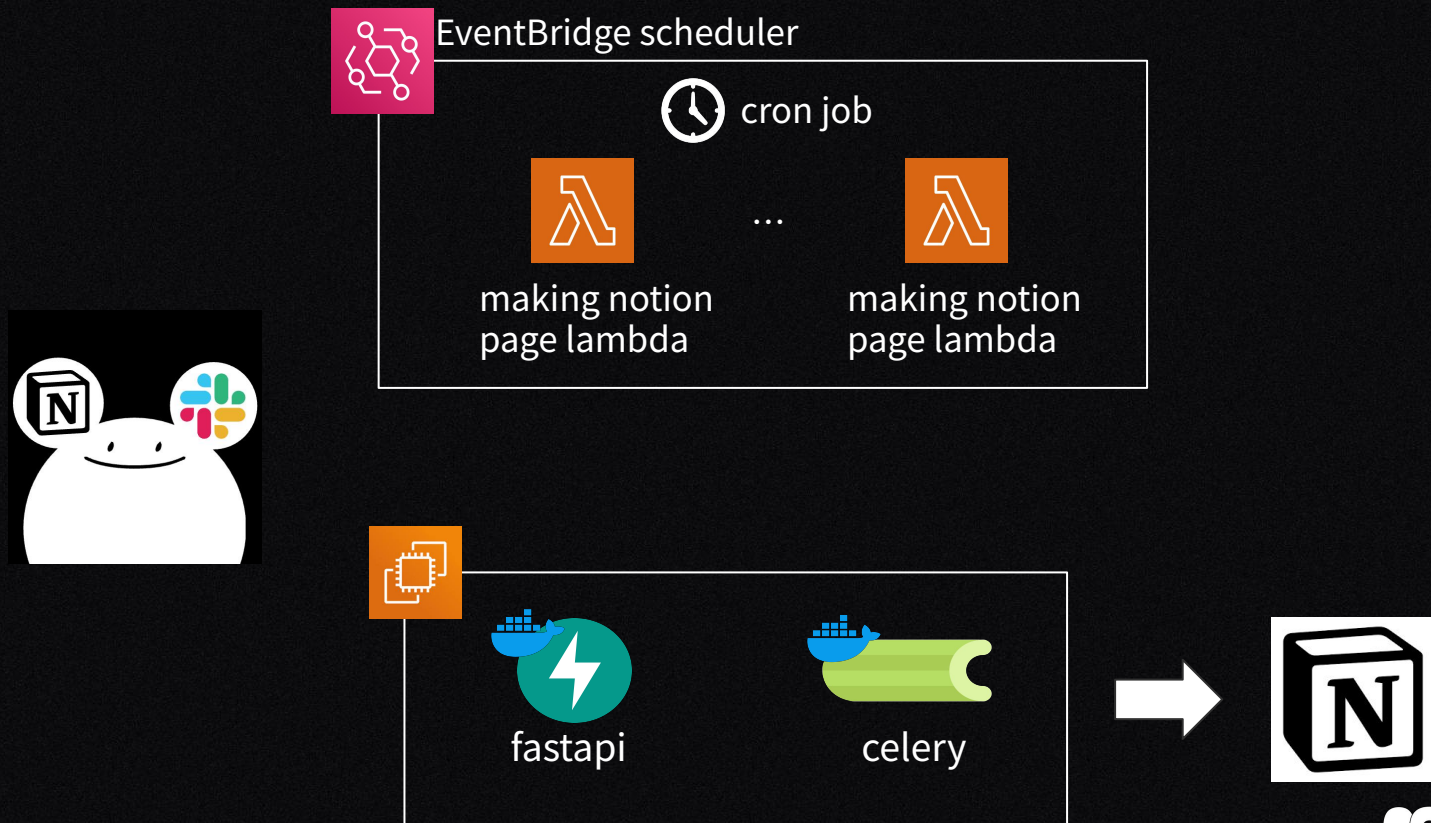
\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



회의 시간 변경이나 페이지 템플릿이 변화되면 **재배포** 해야되는 불편함

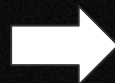
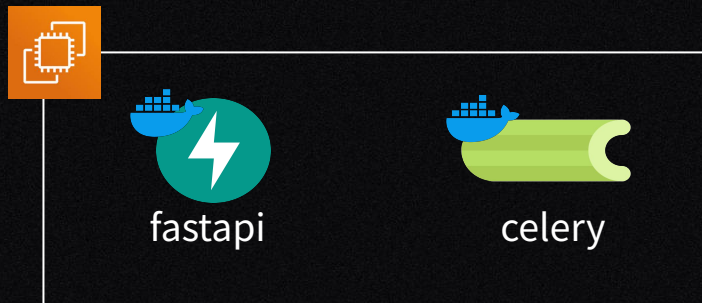
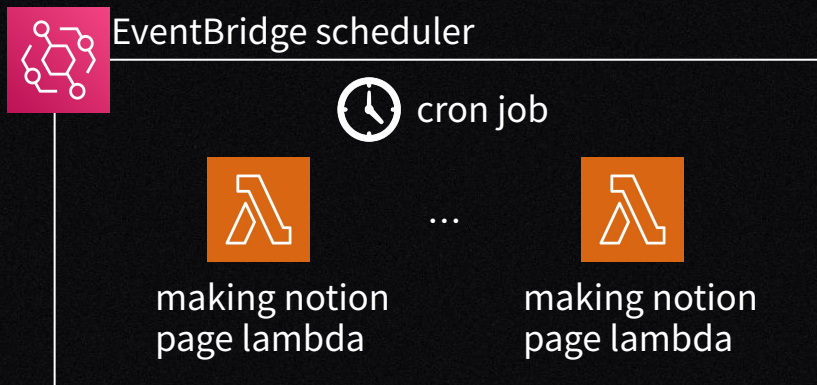
→ 유연한 대처 불가능

\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

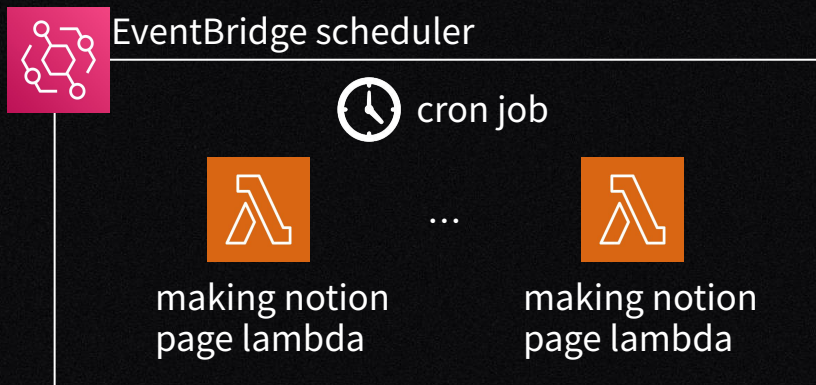


\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

노션 페이지를 만드는 역할



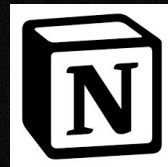
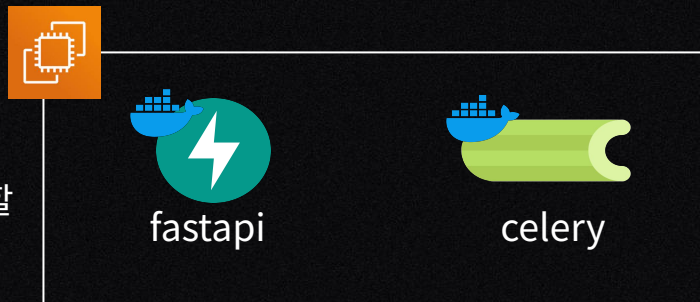
\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



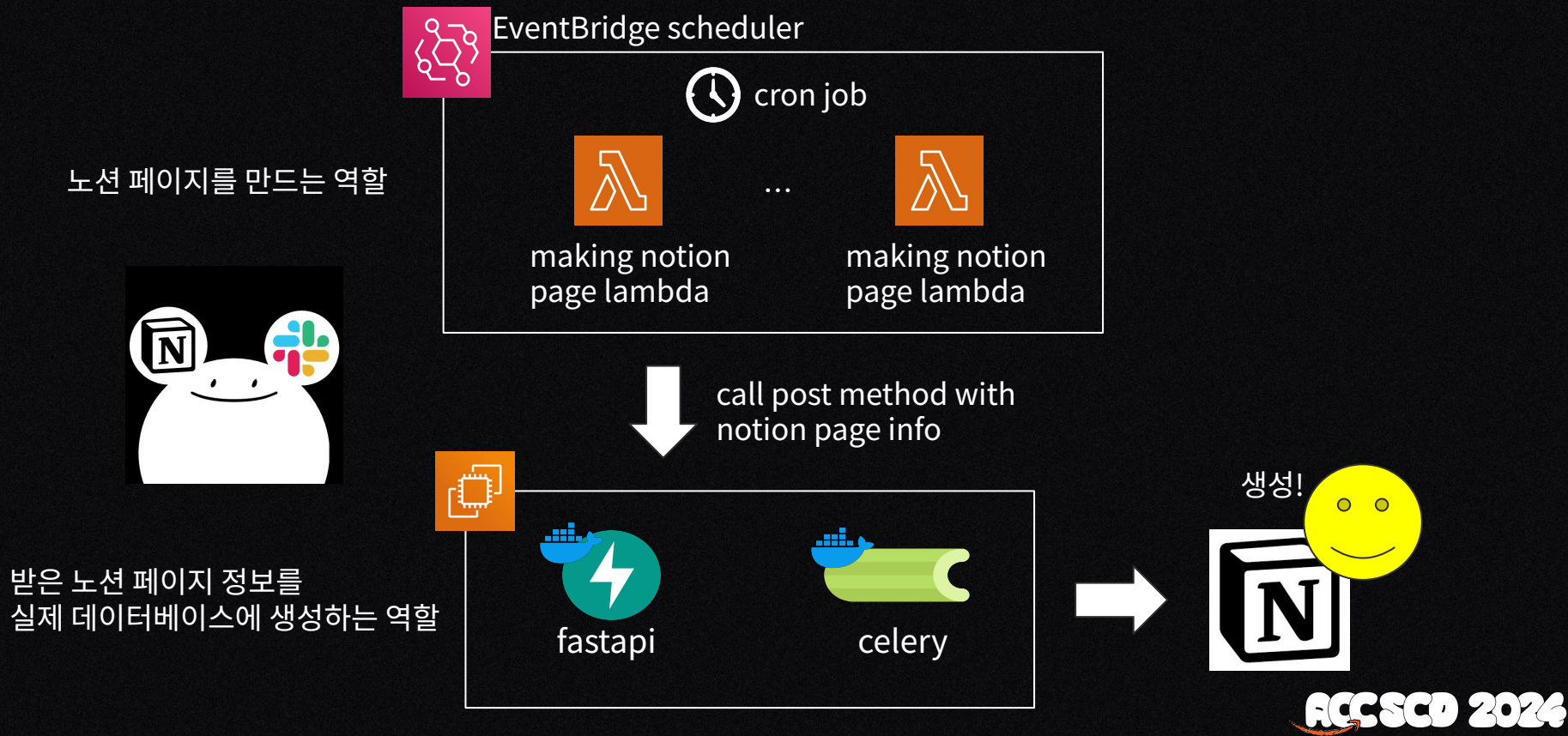
노션 페이지를 만드는 역할



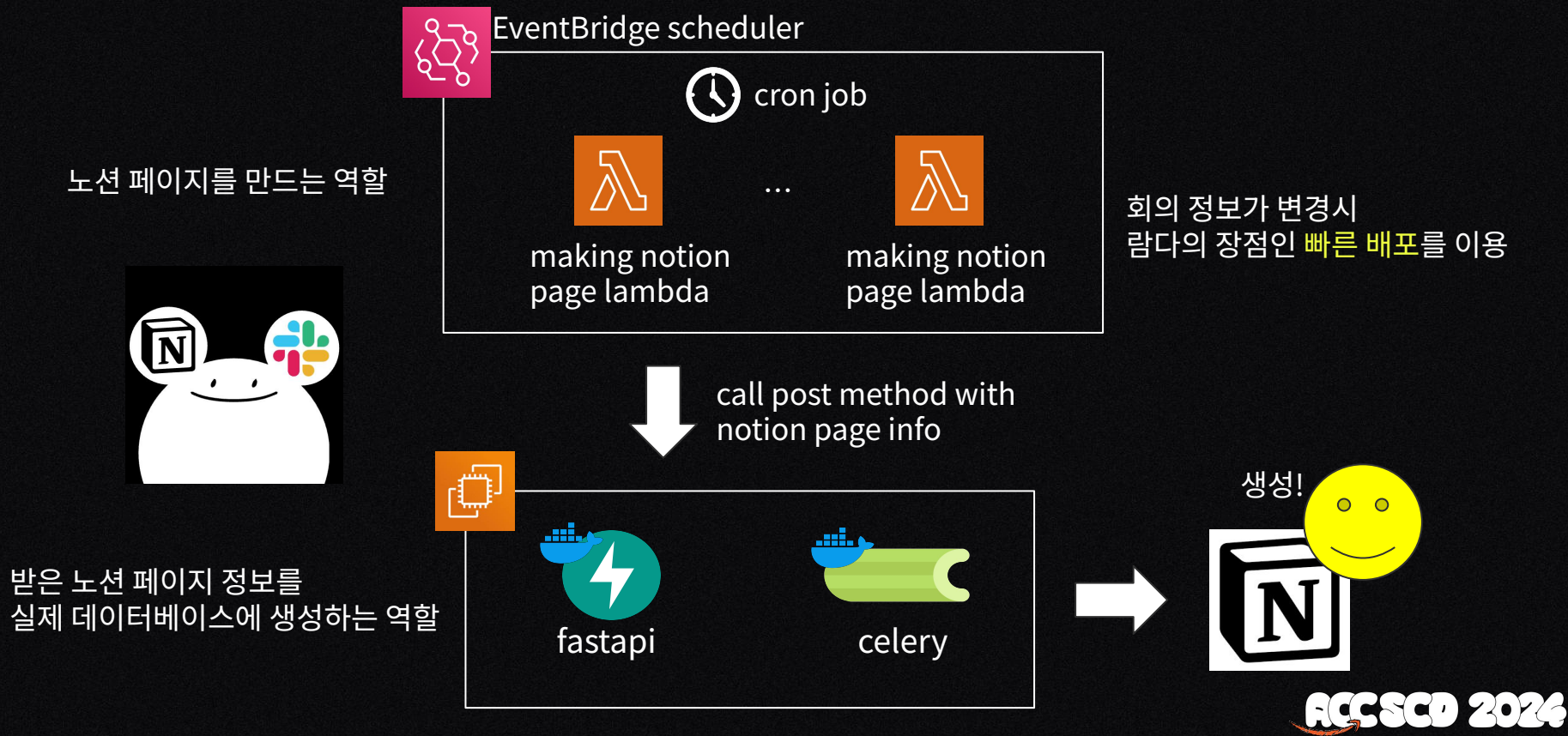
받은 노션 페이지 정보를
실제 데이터베이스에 생성하는 역할



\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

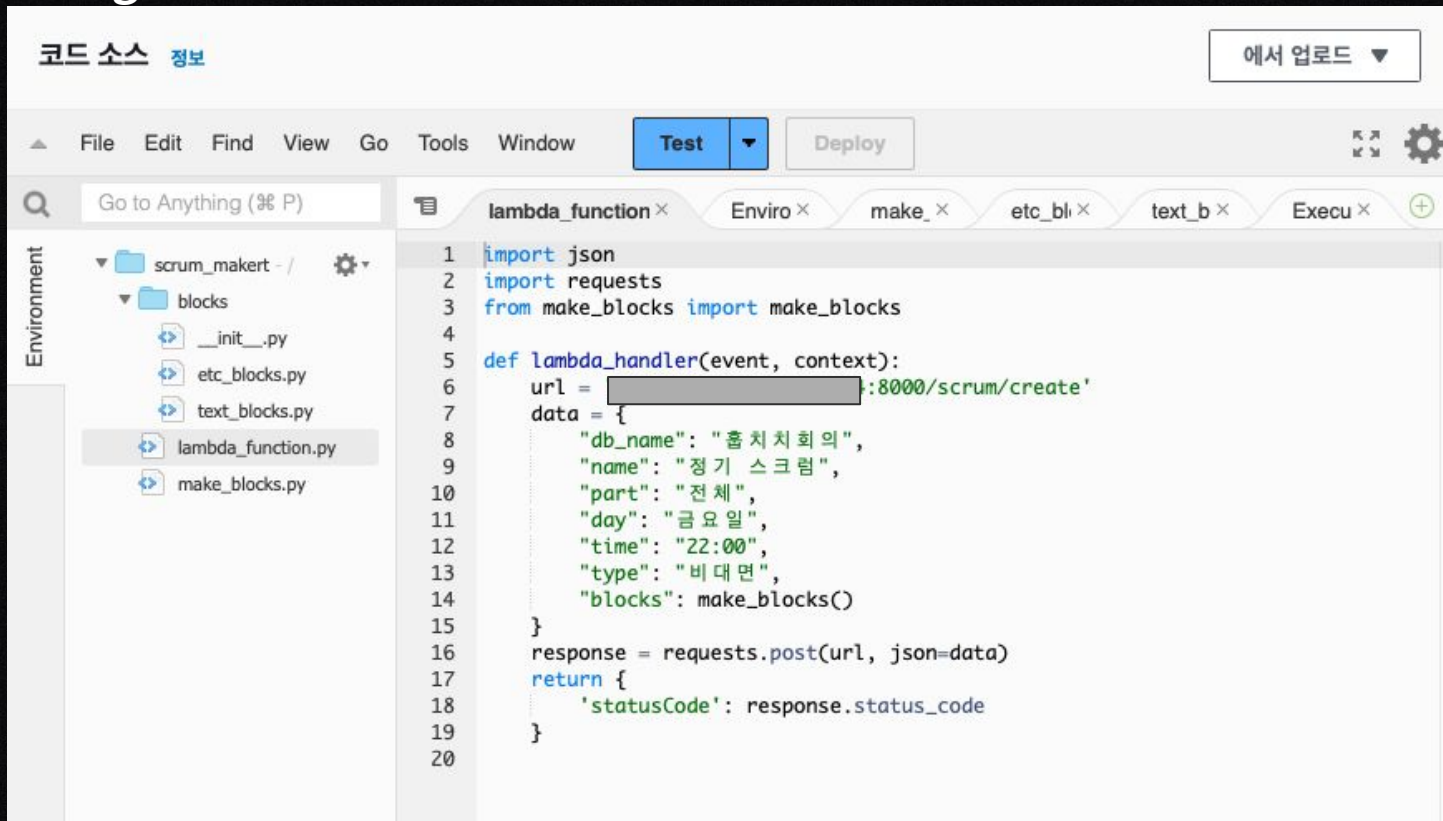


\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

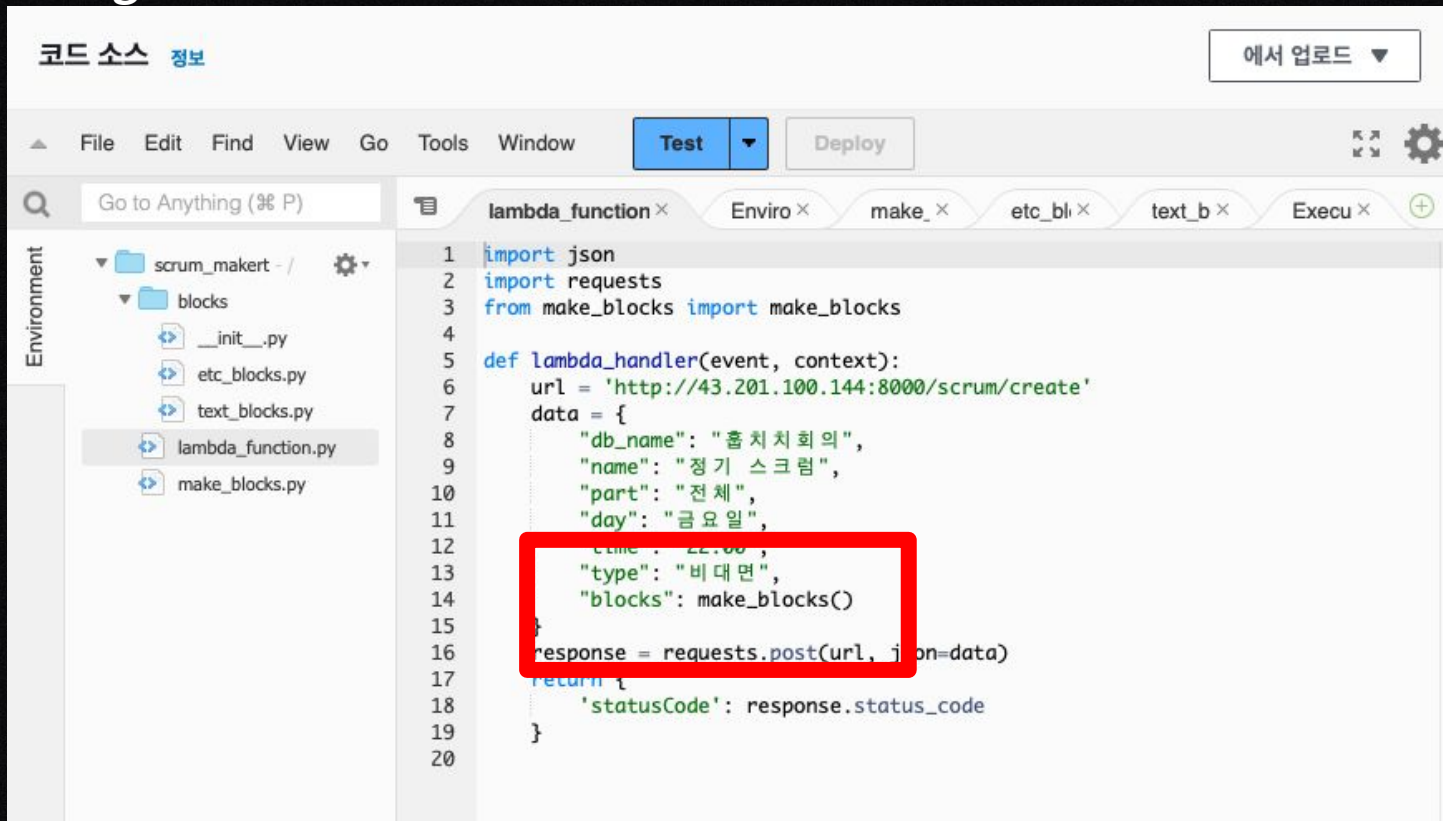


```
async def create_scrum_in_notion(self, request_data: ScrumRequestDto):
    headers, database_id = self._make_header(request_data.db_name)
    page_data = self._make_post(request_data, database_id)
    async with ClientSession() as session:
        async with session.post('https://api.notion.com/v1/pages', headers=headers, data=json.dumps(page_data)) as response:
            return await response.json()
```

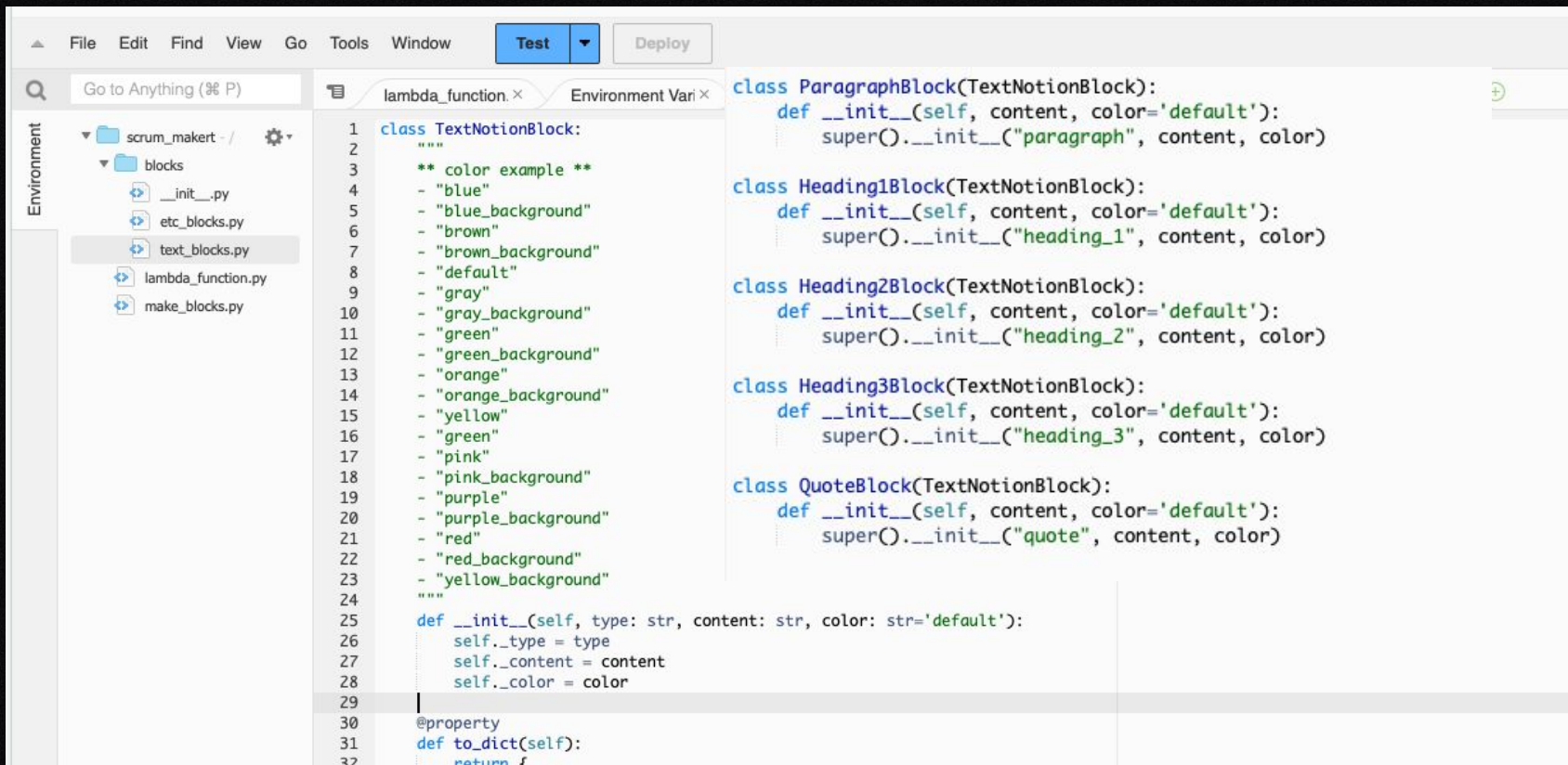
\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



\$ EventBridge는 어디에 쓰이는 걸까?: 사례2



```
File Edit Find View Go Tools Window Test Deploy
Go to Anything (% P)
Environment
  scrum_makert /
    blocks
      __init__.py
      etc_blocks.py
      text_blocks.py
      lambda_function.py
      make_blocks.py
lambda_function.py
1 class TextNotionBlock:
2     """
3     ** color example **
4     - "blue"
5     - "blue_background"
6     - "brown"
7     - "brown_background"
8     - "default"
9     - "gray"
10    - "gray_background"
11    - "green"
12    - "green_background"
13    - "orange"
14    - "orange_background"
15    - "yellow"
16    - "green"
17    - "pink"
18    - "pink_background"
19    - "purple"
20    - "purple_background"
21    - "red"
22    - "red_background"
23    - "yellow_background"
24    """
25    def __init__(self, type: str, content: str, color: str='default'):
26        self._type = type
27        self._content = content
28        self._color = color
29
30    @property
31    def to_dict(self):
32        return {
class ParagraphBlock(TextNotionBlock):
    def __init__(self, content, color='default'):
        super().__init__("paragraph", content, color)

class Heading1Block(TextNotionBlock):
    def __init__(self, content, color='default'):
        super().__init__("heading_1", content, color)

class Heading2Block(TextNotionBlock):
    def __init__(self, content, color='default'):
        super().__init__("heading_2", content, color)

class Heading3Block(TextNotionBlock):
    def __init__(self, content, color='default'):
        super().__init__("heading_3", content, color)

class QuoteBlock(TextNotionBlock):
    def __init__(self, content, color='default'):
        super().__init__("quote", content, color)
```




Go to Anything (% P)

Environment

- ▼ scrum_makert - /
 - ▼ blocks
 - __init__.py
 - etc_blocks.py
 - text_blocks.py
 - lambda_function.py
 - make_blocks.py

```
3
4 def make_blocks():
5     parts = ('기획', '관객 서버', '매니저 서버', '클라이언트', '디자인')
6     colors = ('orange_background', 'yellow_background', 'green_background', 'blue_background', 'purple_background')
7     예정_현황_blocks = [
8         Heading3Block(content).to_dict
9         for content in ('예정', '현황')
10    ]
11    # parts_block을 구성하는 부분 수정
12    parts_block = []
13    for part, color in zip(parts, colors):
14        parts_block.append(Heading2Block(part, color=color).to_dict) # Heading2Block 추가
15        # 예정, 현황에 대한 Heading3Block를 추가
16        for content in ('예정', '현황'):
17            parts_block.append(Heading3Block(content).to_dict)
18
19    original_parts_block = [
20        Heading2Block(part, color=color).to_dict
21        for part, color in zip(parts, colors)
22    ]
23
24    result = [
25        Heading1Block('파트별 진행 상황 공유').to_dict
26    ] + parts_block + [
27        Heading1Block('회고').to_dict
28    ] + [
29        Heading1Block('다음 스프린트 목표').to_dict
30    ] + original_parts_block
31    return result
```

\$ EventBridge는 어디에 쓰이는 걸까?: 사례2

The screenshot displays the Amazon EventBridge console interface. On the left is a navigation sidebar with categories like '개발자 리소스' (Developer Resources), '버스' (Buses), '파이프' (Pipelines), 'Scheduler', '통합' (Integrations), and '스키마 레지스트리' (Schema Registry). The 'Scheduler' section is expanded, showing '일정' (Schedule) as the selected option. The main panel is titled 'scrum_make_schedule' and contains a table of '일정 세부 정보' (Schedule Details). Below this is a tabbed interface with '일정' (Schedule) selected, showing the 'Cron 표현식 정보' (Cron Expression Info) as '0 4 ? * 2 *' with labels for '분' (Minute), '시간' (Hour), '일' (Day), '월' (Month), '요일' (Day of Week), and '연도' (Year).

일정 세부 정보		
일정 이름 scrum_make_schedule	상태 🟢 활성	일정 시작 시간 Jan 22, 2024, 04:00:00 (UTC+09:00)
설명 -	일정 ARN arn:aws:scheduler:ap-northeast-2:279381197488:schedule/default/scrum_make_schedule	일정 종료 시간 -
일정 그룹 이름 default	완료 후 작업 NONE	실행 시간대 Asia/Seoul

일정 | 대상 | 재시도 정책 | DLQ(Dead Letter Queue) | 암호화

일정

Cron 표현식 정보

0 4 ? * 2 *
분 시간 일 월 요일 연도

정기 스크럼

📅 날짜 2024년 3월 22일 오후 10:00

≡ 종류 비데먼

≡ 참여 파트 전체

≡ 참여자 비어 있음

≡ 확정여부 미정

+ 속성 추가

💬 댓글 추가

파트별 진행 상황 공유

기획

예정

현황

관객 서버

예정

현황

매니저 서버

예정

\$

4. 결론

\$ 결론



Lambda

서버 없이도 빠르고 쉽게 코드를 실행 가능

ex) 간단한 크롤링시 복잡하게 서버 구축할 필요 X

\$ 결론



Lambda

다양한 AWS 서비스와 연계가 좋다

EventBridge와 연계로 스케줄링

S3, SQS 등등의 트리거 활용

이외 무궁무진한 사용처 -> serverless architecture

\$ 결



Happy new year



2024

나의 2024년 운세는?!

이름을 입력해주세요.

확인

만든이: 이현제 [Github](#)

나의 2024년 운세는?!

이름을 입력해주세요.

이현제님, 2024년에는 행운이 가득할 것이며 새로운 기회가 찾아올 것입니다. 하지만 자만하지 말고 겸손하고 부지런히 노력하는 것이 중요해요. 올 해에는 적은 돈을 쓰는 것이 좋을 것 같아요. 번아웃을 피하기 위해서는 적절한 휴식과 취미 활동을 꾸준히 유지해야 해요.

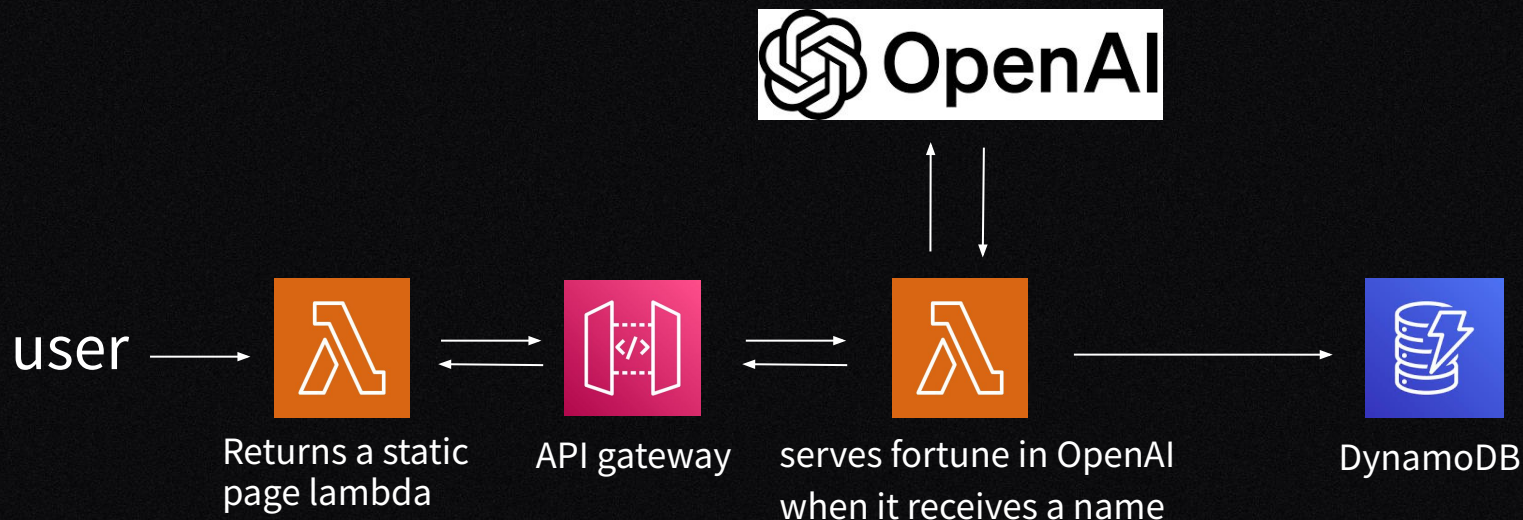
운세복사

공유하기

<http://what2024.kro.kr>

ACCSCD 2024

\$ 결론



반환된 항목 (181)



작업 ▼

항목 생성

< 1 2 3 4 > ⚙️ 🔗

<input type="checkbox"/>	name (문자열) ▼	detail (문자열) ▼
<input type="checkbox"/>		씨, 2024년은 뜻밖의 행운이 찾아올 것입니다. 하지만 주의해야할 점은 조심스럽게 행동하고 결정을 내리는 것입니다. 충동적인 선택은 피해주세요.
<input type="checkbox"/>		님은 2024년에 행운이 가득할 것이니까 기대해도 좋아요. 다만, 조심해야할 점은 남의 말에 대해 너무 민감하게 받아들이지 않는 것이 좋을 것 같아요. 상대방의 ...
<input type="checkbox"/>		씨, 2024년에는 행운이 가득한 한 해가 될 것입니다. 하지만 조금 더 신중하게 선택하고 결정하는 것이 좋을 것 같아요.
<input type="checkbox"/>		님, 2024년에는 행운에 빛나는 한 해가 될 것입니다. 힘들었던 시기를 뒤로 하고 새로운 시작을 맞이할 준비가 되어 있으십니다. 하지만 주의해야 할 점은 무모한...
<input type="checkbox"/>		씨, 2024년은 당신에게 행운이 넘치는 해가 될 것입니다. 그러나 조심해야 할 것은 과도한 낙관적인 마음가짐으로 인해 조심하셔야 합니다. 균형을 잃지 않고 현...
<input type="checkbox"/>		씨, 2024년에는 행운과 성공의 기회가 여러분을 기다리고 있어요. 하지만 조심해야할 점은 너무 과도한 욕심을 가져서 자제하는 것이예요. 절제하지 않으면 의미...
<input type="checkbox"/>		씨, 2024년의 운세를 전해드리자면, 행운이 가득한 한 해가 될 것입니다. 좋은 기회와 성공이 당신을 기다릴 것이며, 자신의 잠재력을 발휘할 수 있는 좋은 시기...
<input type="checkbox"/>		씨, 2024년에는 행운과 성공이 가득한 한 해가 될 것입니다. 하지만 조심해야 할 점은 욕심과 과도한 자만심입니다. 겸손을 잃지 않고 자신의 성취를 축하하는 시...
<input type="checkbox"/>		님, 2024년에는 행운이 가득한 한 해가 될 것입니다. 하지만 조심해야 할 것은 과도한 낙관적인 태도입니다. 현실적인 목표를 설정하고 꾸준한 노력으로 성취해 ...
<input type="checkbox"/>		님의 2024년 운세는 밝고 행운이 가득한 해입니다. 지금까지의 노력과 끈기가 결실을 맺어 큰 성공을 얻을 수 있을 것입니다. 하지만 조심해야 할 점은 오만함에...
<input type="checkbox"/>		님, 2024년은 여러가지 기회가 찾아올 예정입니다. 자신의 잠재력을 믿고 도전하는 태도를 가지세요. 하지만 조심해야 할 점은 조급함에 빠지지 않는 것입니다. ...
<input type="checkbox"/>		님, 2024년 운세는 풍요롭고 성공적인 한 해가 될 것입니다. 하지만 조심해야 할 것은 저조한 분위기에서 너무 안주하지 말고 항상 노력을 게을리하지 않아야 함...

\$ 결론

AWS 완전 관리형 서비스
개발자가 쉽게 API를 생성, 게시,
유지 관리, 모니터링, 보호 가능



\$

```
588 // Submit 버튼을 누르면 overlay를 보여줍니다.  
589 document.getElementById('overlay').style.display = "block";  
590  
591 fetch('https://zqzifv0n38.execute-api.ap-northeast-2.amazonaws.com/default', {  
592   method: 'POST',  
593   headers: {  
594     'Content-Type': 'application/json'  
595   },  
596   body: JSON.stringify({ name: name })  
597 })
```

☰ /

POST

ARN
arn:aws:execute-api:ap-northeast-2:550581268183:zqzifv0n38/* /POST/

리소스 ID
5xskr1c462

클라이언트

→

메서드 요청

→

통합 요청

→

Lambda 통합

←

통합 응답

←

메서드 응답

←

클라이언트

메서드 요청

통합 요청

통합 응답

메서드 응답

테스트

통합 요청 설정

편집

통합 유형 정보
Lambda

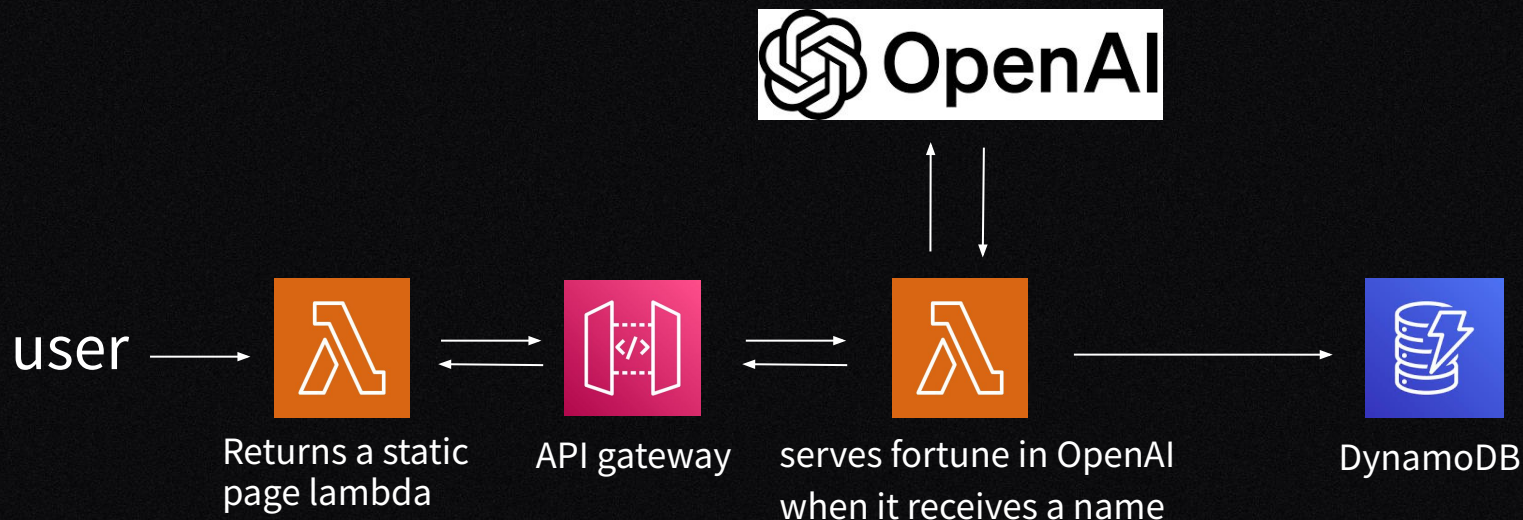
리전
ap-northeast-2

Lambda 프록시 통합 정보
False

Lambda 함수
fortuneapi

SCD 2024

\$ 결론



\$ 결론



Lambda

무엇보다 장점은 **본 서버와의 분리**

저렴한 비용으로

간헐적으로 처리를 해야하는 프로세스들 및
잦은 변경으로 재배포가 부담스러운 기능을 분리

\$

감사합니다

Break time 집현전 Track

Q&A Slido



다음 세션을 들으면서 궁금한 점이 생긴다면
위의 Slido로 들어가서 자유롭게 질문을
남겨주세요.

세션 만족도 조사



더 나은 ACC SCD를 위한 세션별 설문 조사를
진행하고 있습니다. 아래의 QR 코드를 통해
여러분의 소중한 의견을 공유해주세요.