AWS서비스를 이용한 Daily Cost 메일로 받기!

널캉말캉세미나

현욱NULL4U

목차

목차

효과 와 장점

01. 편리하다!

금액을 확인하기 매우 불편함

AWS에서 한달치 금액을 확인 하는 것은 쉽지만
하루치를 확인하기에는 어려움

하지만 AWS API를 사용한다면

하루의 비용을 빠르게 볼 수 있고 어떤 서비스에서
많이 나왔는지 확인 할 수 있으며 매일 이메일로
자동으로 전송 된다!

02. 관짝행을 피할 수 있다

장례식

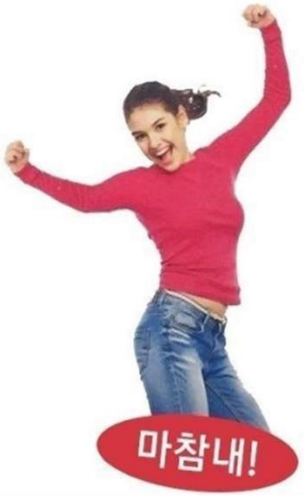
이미 NULL4U는 전설 적인 사건 두번으로 인해 장례식을 치렀으며 현재 개 같이 부활한 상태다

무빙

하지만 이걸 사용하면 같은 팀의 트롤을 방지 할 수 있다

편리함 + 관짝 무빙

마참내!



서비스 소개

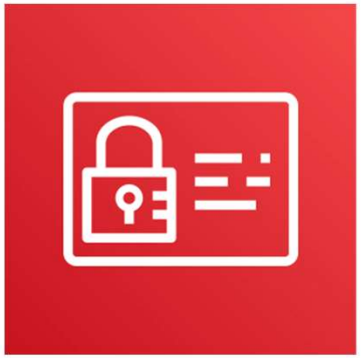
01. 서비스 소개

Lambda

서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스입니다.

S3

확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다.



IAM

AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 웹 서비스입니다.

SES

사용자의 이메일 주소와 도메인을 사용해 이메일을 보내고 받는 서비스입니다.

02. Lambda

Lambda는 AWS에서 제공하는 서버리스 컴퓨팅 플랫폼입니다.
Lambda는 아래와 같이 많은 장점이 있습니다.

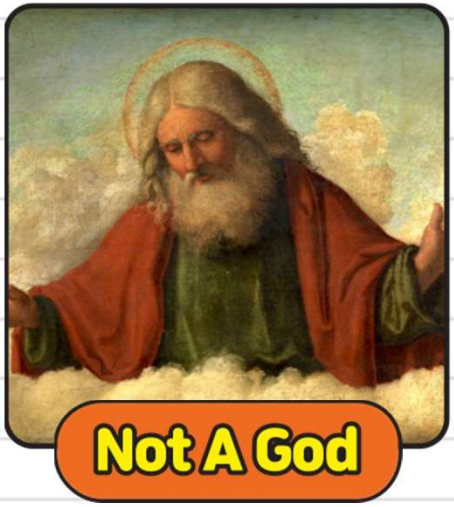
03. S3 버킷

S3 버킷은 평균 GB당 0.023 USD 라는 저렴한 가격에 서비스를 제공 하고 있습니다.

S3 버킷은 다른 AWS 서비스 혹은 AWS CLI 와 호환되어 간편하게 접근할 수 있습니다.

04. IAM

IAM은 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 서비스입니다.
IAM을 사용하면 사용자가 액세스할 수 있는 AWS 리소스를 제어하는 권한을 중앙에서 관리할 수 있습니다.



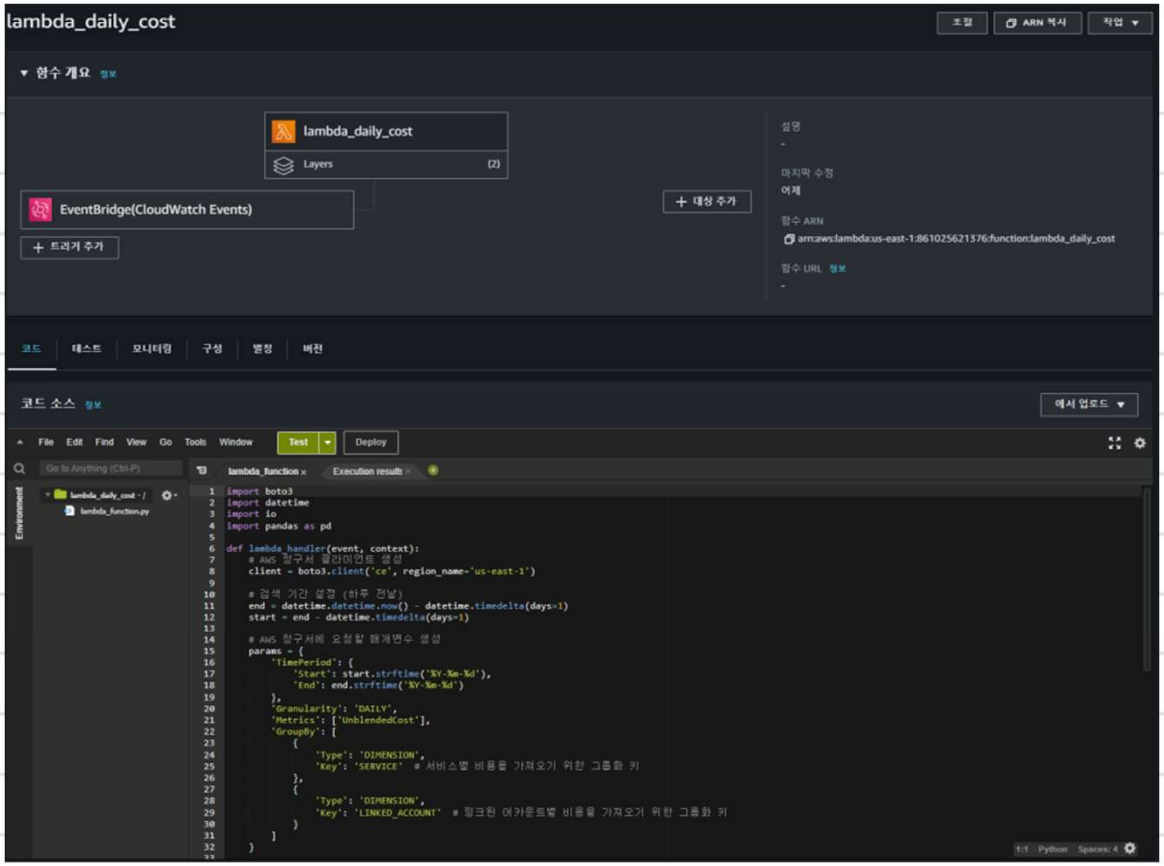
05. SES

SES는 사용자의 이메일 주소와 도메인을 사용해 이메일을 보내고 받기 손쉬운 방법을 제공하는 이메일 플랫폼입니다.

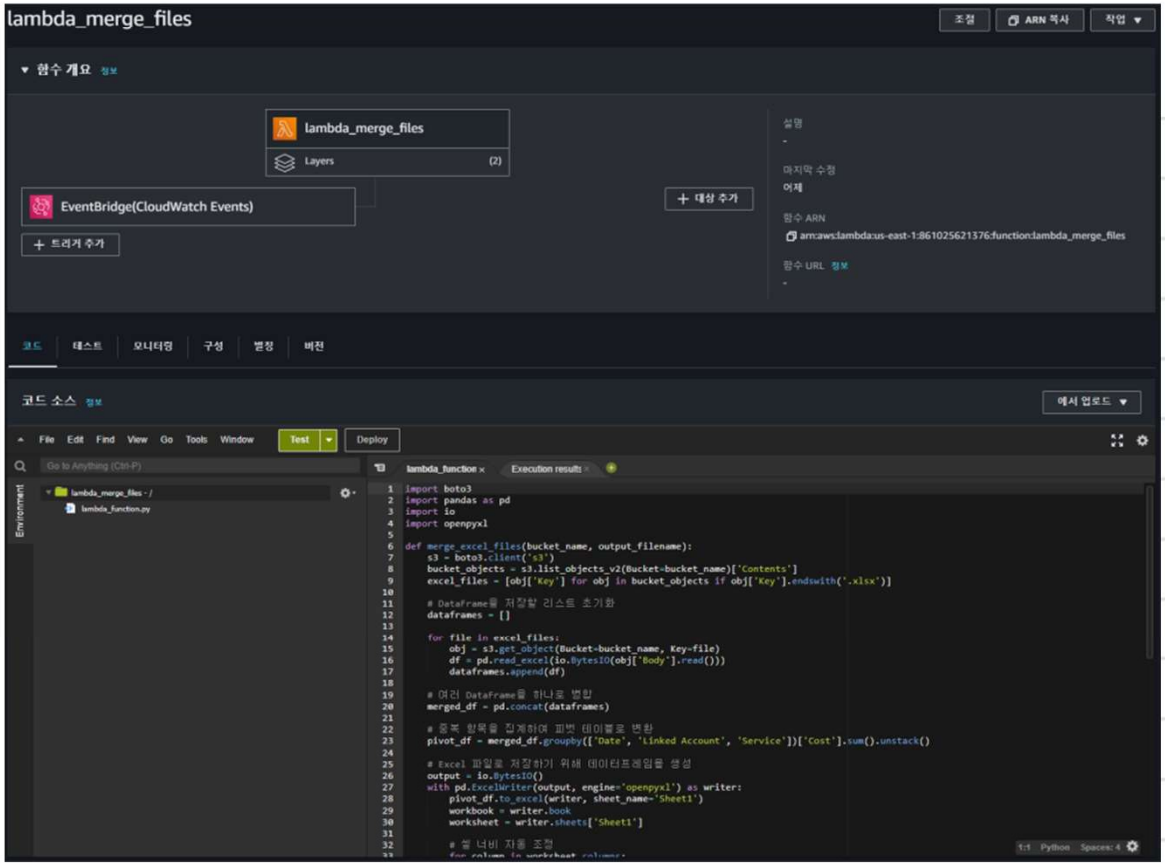
서비스 구성

01. 서비스 구성

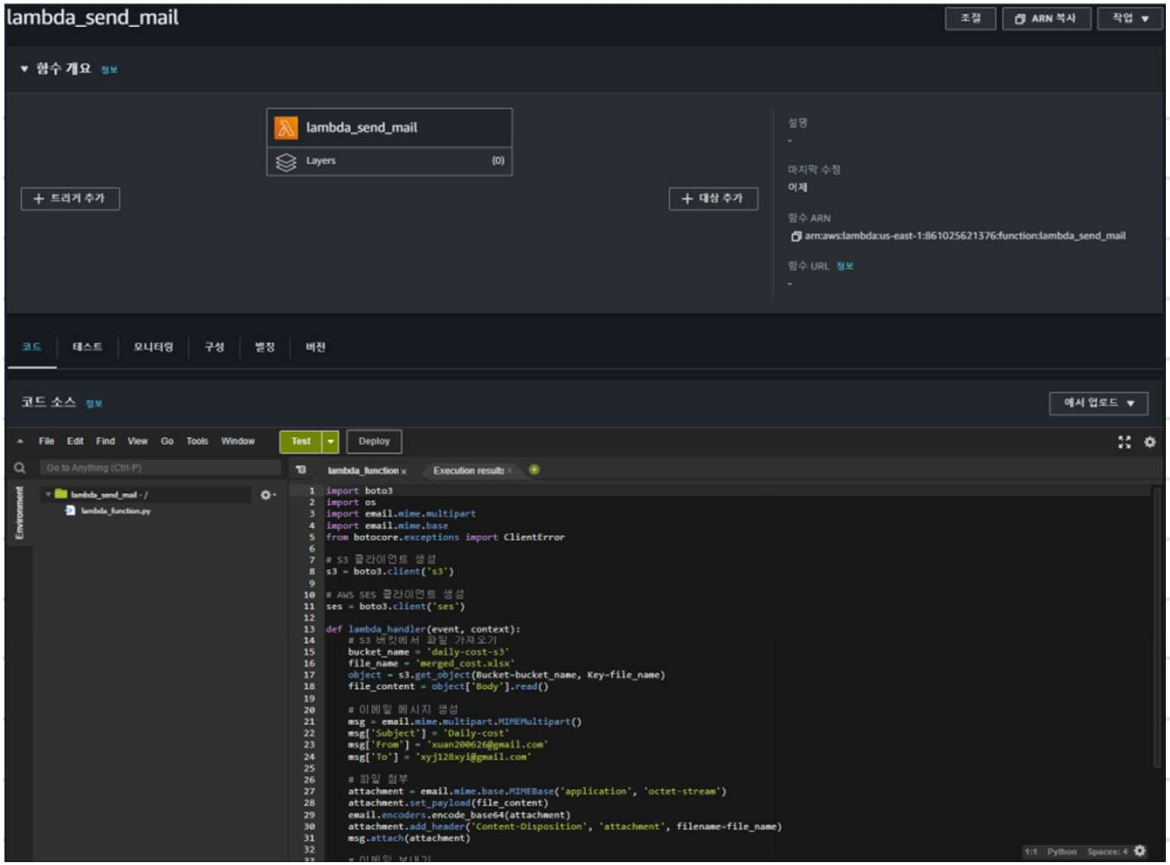
02. 실제 구축 및 결과


lambda_daily_cost
조절
ARN 복사
작업
함수 개요 정보
lambda_daily_cost
Layers (2)
EventBridge(CloudWatch Events)
+ 대상 추가
+ 트리거 추가
설명
-
마지막 수정
어제
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:lambda_daily_cost
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
코드 소스 정보
에서 업로드
File Edit Find View Go Tools Window
Test
Deploy
lambda_function
Execution result
lambda_daily_cost - /
lambda_function.py
Environment
import boto3
import datetime
import io
import pandas as pd
def lambda_handler(event, context):
# AWS 청구서 클라이언트 생성
client = boto3.client('ce', region_name='us-east-1')
# 검색 기간 설정 (하루 전날)
end = datetime.datetime.now() - datetime.timedelta(days=1)
start = end - datetime.timedelta(days=1)
# AWS 청구서에 요청할 매개변수 생성
params = {
'TimePeriod': {
'Start': start.strftime('%Y-%m-%d'),
'End': end.strftime('%Y-%m-%d')
},
'Granularity': 'DAILY',
'Metrics': ['UnblendedCost'],
'GroupBy': [
{
'Type': 'DIMENSION',
'Key': 'SERVICE' # 서비스별 비용을 가져오기 위한 그룹화 키
},
{
'Type': 'DIMENSION',
'Key': 'LINKED_ACCOUNT' # 링크된 어카운트별 비용을 가져오기 위한 그룹화 키
}
]
}
1:1 Python Spaces: 4

02. 실제 구축 및 결과


lambda_merge_files
조절
ARN 복사
작업
함수 개요 정보
lambda_merge_files
Layers (2)
EventBridge(CloudWatch Events)
+ 트리거 추가
+ 대상 추가
설명
-
마지막 수정
어제
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:lambda_merge_files
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
코드 소스 정보
에서 업로드
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_merge_files - /
lambda_function.py
Environment
lambda_function
Execution results
import boto3
import pandas as pd
import io
import openpyxl
def merge_excel_files(bucket_name, output_filename):
s3 = boto3.client('s3')
bucket_objects = s3.list_objects_v2(Bucket=bucket_name)['Contents']
excel_files = [obj['Key'] for obj in bucket_objects if obj['Key'].endswith('.xlsx')]
# DataFrame을 저장할 리스트 초기화
dataframes = []
for file in excel_files:
obj = s3.get_object(Bucket=bucket_name, Key=file)
df = pd.read_excel(io.BytesIO(obj['Body'].read()))
dataframes.append(df)
# 여러 DataFrame을 하나로 병합
merged_df = pd.concat(dataframes)
# 중복 항목을 집계하여 피벗 테이블로 변환
pivot_df = merged_df.groupby(['Date', 'Linked Account', 'Service'])['Cost'].sum().unstack()
# Excel 파일로 저장하기 위해 데이터프레임을 생성
output = io.BytesIO()
with pd.ExcelWriter(output, engine='openpyxl') as writer:
pivot_df.to_excel(writer, sheet_name='Sheet1')
workbook = writer.book
worksheet = writer.sheets['Sheet1']
# 셀 너비 자동 조정
1:1 Python Spaces: 4

02. 실제 구축 및 결과


lambda_send_mail
조절
ARN 복사
작업
함수 개요 정보
lambda_send_mail
Layers (0)
+ 트리거 추가
+ 대상 추가
설명
-
마지막 수정
어제
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:lambda_send_mail
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
코드 소스 정보
에서 업로드
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Execution results
lambda_send_mail - /
lambda_function.py
Environment
import boto3
import os
import email.mime.multipart
import email.mime.base
from botocore.exceptions import ClientError
# S3 클라이언트 생성
s3 = boto3.client('s3')
# AWS SES 클라이언트 생성
ses = boto3.client('ses')
def lambda_handler(event, context):
# S3 버킷에서 파일 가져오기
bucket_name = 'daily-cost-s3'
file_name = 'merged_cost.xlsx'
object = s3.get_object(Bucket=bucket_name, Key=file_name)
file_content = object['Body'].read()
# 이메일 메시지 생성
msg = email.mime.multipart.MIMEMultipart()
msg['Subject'] = 'Daily-cost'
msg['From'] = 'xuan200626@gmail.com'
msg['To'] = 'xyj128xyi@gmail.com'
# 파일 첨부
attachment = email.mime.base.MIMEBase('application', 'octet-stream')
attachment.set_payload(file_content)
email.encoders.encode_base64(attachment)
attachment.add_header('Content-Disposition', 'attachment', filename=file_name)
msg.attach(attachment)
1:1 Python Spaces: 4

02. 실제 구축 및 결과

H24

	A	B	C	D	E	F	G	H
1	Date	Linked Account	AWS Cost Explorer	AWS Lambda	Amazon Simple Storage Service	AmazonCloudWatch	EC2 - Other	Total
2	2023-05-27	8.61026E+11			0	0	0.008849704	0.00885
3	2023-05-28	8.61026E+11			0	0	0.004830047	0.00483
4	2023-05-29	8.61026E+11	0.05	1.0217E-06	0	0	0.004830047	0.05483
5	2023-06-04	8.61026E+11			0	0	0	0
6	2023-06-10	8.61026E+11			0	0	0	0

발표를 마치며

Q&A