현욱

About Me

안녕하세요!

> 저는 세명 컴퓨터 고등학교에 재학 중인 현욱이라고 합니다.
> 코딩을 좋아하며 현재는 클라우드 컴퓨팅을 공부하고 있습니다.

Skills

C

◆◆◆◇◇

Java

◆◆◆◆◇

Python

◆◆◆◆◇

Assembler

◆◇◇◇◇

SQL

◆◆◆◇◇

Linux

◆◆◆◇◇

AWS

◆◆◆◆◇

Project

Project

Aa 이름	STACK
AWS_Daily_Cost	AWS Python
AWS_Daily_Cost_2	AWS Shell

ETC

 **Instagram**

 **Tistory**

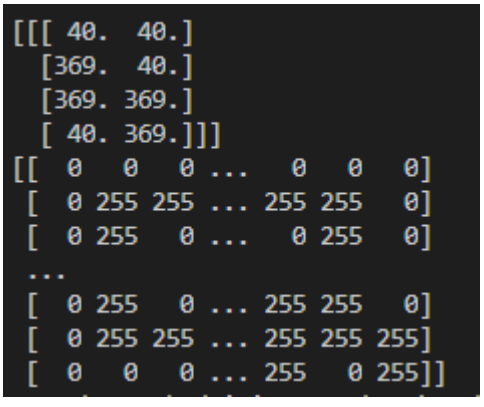
 **Notion**

 **GitHub**

11/02

👉 QR코드 읽어들이기 + 간단한 매크로

```
import qrcode #얘는 전에거랑 좀 다른 qrcode 만드는 애임
import cv2 #얘가 읽는 놈임

def read(open_img):
    detector = cv2.QRCodeDetector()
    data, bbox, _ = detector.detectAndDecode(open_img)
    if data:
        print(data)
        print(bbox)
        print(_)
filepath = "jobhealing.png"
cv_img1=cv2.imread(filepath)
read(cv_img1)
```



위치가 좌상 우상 좌하 우하 였나 그럴거임 ㅇㅇ 아마도

```
def read2(open_img):
    detector = cv2.QRCodeDetector()
    data, bbox, _ = detector.detectAndDecode(open_img)
    if data:
        print(data)
        print(bbox)

        lefttop = (int(bbox[0][0][0]), int(bbox[0][0][1]))
        rightbottom = (int(bbox[0][2][0]), int(bbox[0][2][1])
        cv2.rectangle(open_img, lefttop, rightbottom,(255,0,0
        cv2_imshow(open_img)

filepath = "jobhealing.png"
cv_img1=cv2.imread(filepath)
read2(cv_img1)
```

⛔ 두번 째 코드가 있긴한데 작동은 못해봄 모듈이 고장남

👉 이제는 qr아니고 간단한 매크로 만들기

⛔ 아나콘다 사용 실패(설치 이슈)

11/09

📢 Pyautogui 마우스 컨트롤

```
import pyautogui as pg

print(pg.size())

print(pg.position())

pg.mouseInfo() #유용한 앱? 뭔가를 열어 줌

pg.moveTo(520,986)

pg.moveTo(520,986,duration=3)

pg.move(520,986,duration=0.5)

pg.click()

pg.click(520,986)

pg.click(520,986,duration=1)

pg.click(clicks=2,interval=0.2,button='right')
#clicks = 횟수 interval = 간격 시간 button = 누르는 버튼
```

👉 기본적인 사용법

📢 화면 이미지 인식

```
import pyautogui as pg
import time


i = pg.locateOnScreen('num4.png') #화면에서 이미지의 좌표를 찾아서 
print(i)

q = pg.center(i) #찾은 좌표의 중앙 반환
#pg.locateCenterOnScreen() 함수로 이미지의 중앙 좌표 반환!
print(q)

pg.click(q)
```

```
import pyautogui as pg
import time


i = pg.position() #position으로 x,y 값을 i에 저장
x,y = i # x,y 라는 변수에 각각 x,y 값을 저장
pg.screenshot('num7.png',region=(x,y,50,50))
```

💡 이거 쓰면 좀 재밌는 매크로 만들 수 있을 듯

11/14

📢 pyautogui로 키보드 제어하기

```
import pyautogui as pg
import time
import pyperclip

pg.write("괄호안의 문자 타이핑")  # 한글은 이슈가 있어서 타이핑 안됨
pg.typewrite("문자열을 타이핑")
pg.typewrite(['enter'])  # 엔터키
pg.typewrite(['a','b','enter'])  # a, b, 입력 후 엔터
pg.typewrite('a','b', ['enter'])  # a, b, 입력 후 엔터

pyperclip.copy('한글')   #pyperclip으로 Copy해서 사용은 가능

pg.hotkey('enter')
pg.hotkey('ctrl','v')
```

✅ 기본적인 사용법

```
import pyautogui as pg

pg.hotkey('win','r')
pg.typewrite('cmd')
pg.hotkey('enter')
pg.typewrite('cd C:\\Users\\Administrator\\Desktop\\python')
```

```
# 파이썬 파일 경로
pg.hotkey('enter')
pg.typewrite('python practice.py')
pg.hotkey('enter')
```

💡 무한 반복 실행

📢 selenium 사용하기!

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

url = 'https://naver.com'

driver = webdriver.Chrome()

driver.get(url)

driver.find_element(By.CLASS_NAME,"search_input").send_keys("

time.sleep(1000)
```

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

url = 'https://naver.com'

driver = webdriver.Chrome()
```

```
driver.get(url)

driver.find_element(By.CLASS_NAME,"search_input").send_keys("

time.sleep(2)

driver.find_element(By.NAME,"query").clear()

time.sleep(2)

driver.find_element(By.NAME,"query").send_keys("파이썬이다"+Keys

time.sleep(2)

png = driver.get_screenshot_as_png()

open('search_result.png','wb').write(png)
```

📢 스샷 찍기

11/21

📢 selenium 이어서!

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

driver = webdriver.Chrome()
ele = driver.get('https://news.naver.com/')

driver.find_element(By.CLASS_NAME,"Nicon_search").click()
time.sleep(2)
driver.find_element(By.NAME,"query").send_keys("수학능력시험" +
time.sleep(100)
```

📢 CSS_SELECTOR로 요소 찾기

✅ 사용법: 찾은 요소 우클릭 COPY>COPY SELECTOR

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time
```

```
driver = webdriver.Chrome()

ele = driver.get('https://www.naver.com/')
time.sleep(2)

shortcut = driver.find_element(By.CSS_SELECTOR, "#shortcutAre
print(shortcut.text)

shortcut.click()    #click은 항상 정 가운데 부분을 클릭함
time.sleep(100)
```

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

driver = webdriver.Chrome()

ele = driver.get('https://www.google.co.kr/imghp?hl=ko')
time.sleep(2)

driver.find_element(By.CSS_SELECTOR,"q").send_keys("악뮤" + Ke
time.sleep(2)

shortcut = driver.find_element(By.CSS_SELECTOR,"#islrg > div.
shortcut.click()
time.sleep(100)
```

📢 보조도구 Navigation

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
```

```
import time

driver = webdriver.Chrome()

driver.get('https://www.google.co.kr')
time.sleep(2)
driver.get('https://www.naver.com')

#뒤로가기
driver.back()
time.sleep(2)

#앞으로 가기
driver.forward()
time.sleep(2)

#페이지 새로고침
driver.refresh()
time.sleep(2)
```

✅ 사용법

📢 브라우저 정보 가져오기

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

driver = webdriver.Chrome()

driver.get('https://www.google.co.kr')
```

```
#title
title = driver.title
print(title)

#url
url = driver.current_url
print(url)
```

✅ 사용법

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time

driver = webdriver.Chrome()

driver.get('https://mail.naver.com')

#url
url = driver.current_url
print(url)

login_url = "nid.naver.com"
if login_url in url:    # url안에 login_url이 있는가
    print("로그인 필요함!")
else:
    print("로그인 필요하지 않음!")
```

🛠️ 로그인 필요 여부 확인

📌 오늘 끄읕!

📘

11/23

📢 WebDriverWait 사용하기!

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as

driver = webdriver.Chrome()
driver.get('https:/www.naver.com')

css_selector = "#right-content-area > div.Layout-module__righ

WebDriverWait(driver, 10).until(EC.presence_of_all_elements_l
#WebDriverWait(크롬드라이버, 최대 시간).동안(EC.elements가 로딩될때((
print(driver.find_element(By.CSS_SELECTOR,css_selector).text)
```

✅ WebDriverWait를 사용하면 time.sleep()을 사용했을 때의 단점을 보완 할 수 있다.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
```

```
import time

driver = webdriver.Chrome()
keyword = input('검색할 제목을 적어주세요')
searchurl = 'https://www.youtube.com/results?search_query='
driver.get(searchurl+keyword)
for i in range(10):
    title = driver.find_elements(By.CSS_SELECTOR,'#video-titl
    url = title.get_attribute("href")
    print(f'{title.text}:{url}')
time.sleep(100)
```

✅ find_elements 는 여러개 가져옴 뒤에 리스트 형식이라서 뒤에 []배열 표시해야 함

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import time

driver = webdriver.Chrome()
query = 'rtx 4090'
page = 1
url = f'https://search.shopping.naver.com/search/all?pagingSi
target = '#content > div.style_content__xWg5l > div.basicList_

driver.get(url)
time.sleep(2)

for _ in range(2):
    driver.execute_script("window.scrollBy(0,10000);")
    time.sleep(1)
```

```
try:
    item = driver.find_element(By.CSS_SELECTOR,target)
    item_name = item.text
    item_code = item.get_attribute('data-i')
    data = item.get_attribute('data-nclick')
    rank = data.split(f'{item_code},r:')[1]

    print(f'{item_name}은 {query}키워드로 검색시 {page}페이지에서 검
except:
    print(f'{page}페이지에서 타켓 상품을 못 찾음')
```

📢 일단 오늘은 여기까지

11-30

📢 웹 크롤링 CONTINUE!

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import time

driver = webdriver.Chrome()
keyword = input('검색할 제목을 적어주세요')
searchurl = 'https://search.naver.com/search.naver?where=news
driver.get(searchurl+keyword)
items = driver.find_elements(By.CLASS_NAME,'news_tit')
for i in range(len(items)):
    url = items[i].get_attribute("href")
    print(f'{items[i].text}:{url}')
time.sleep(100)
```

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import time

driver = webdriver.Chrome()
keyword = input('검색할 제목을 적어주세요')
```

```
searchurl = 'https://www.google.com/imghp'
driver.get(searchurl)
search = driver.find_element(By.NAME,'q')
search.send_keys(keyword + Keys.ENTER)


for i in range(0,5):
    items = driver.find_element(By.XPATH,f'//div[@data-ri="{i
    time.sleep(1.5)
time.sleep(100)
```

✅ XPATH 엄청 편함

- `//` : 문서의 어떤 위치에서나 요소를 찾기 위해 사용합니다.
- `[@attribute='value']` : 특정 속성의 값으로 요소를 선택합니다.
- `/` : 직계 자식 요소를 나타냅니다.
- `[]` : 조건을 지정하여 요소를 선택합니다.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import time

driver = webdriver.Chrome()
keyword = input('검색할 제목을 적어주세요')
searchurl = 'https://www.google.com/imghp'
driver.get(searchurl)
search = driver.find_element(By.NAME,'q')
search.send_keys(keyword + Keys.ENTER)

items = driver.find_elements(By.XPATH,f'//div[@data-ri]')
elements = driver.find_elements(By.XPATH, f'//*[@data-ri]')
```

```
for i in elements:
    i.click()
    time.sleep(1.5)
time.sleep(100)
```

💡 //*[@data-ri]
뒤에 =’숫자’ 이거 빼면 그냥 속성만 있으면 작동함

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import time

driver = webdriver.Chrome()
sea = '김세정'.replace(' ','+')
url = f'https://www.google.com/search?q={sea}&tbm=isch'

driver.get(url)
time.sleep(1)

elements = driver.find_elements(By.XPATH, f'//*[@data-ri]')
flag = 0
for element in elements:
    ele = element.find_element(By.TAG_NAME,'img')
    element.click()
    print(ele.get_attribute('alt'))
    time.sleep(1.5)
time.sleep(100)
```

💡 data-ri가 상위 객체에 있어서 data-ri의 하위 요소를 검색하고 그 중에 alt를 가져옴

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import urllib.request
import time

driver = webdriver.Chrome()
sea = '김세정'.replace(' ','+')
url = f'https://www.google.com/search?q={sea}&tbm=isch'

driver.get(url)
time.sleep(1)
flag = 0
elements = driver.find_elements(By.XPATH, f'//*[@data-ri]')
for element in elements:
    ele = element.find_element(By.TAG_NAME,'img')
    print(ele.get_attribute('alt'))
    imgurl = ele.get_attribute('src')
    urllib.request.urlretrieve(imgurl,f"{sea}{flag}.jpg")
    element.click()
    flag += 1
    time.sleep(0.2)
time.sleep(100)
```

✅ urllib.request.urlretrieve(요소,"이름.jpg")

📢 일단 오늘은 여기까지

12/05

📢 pandas 사용하기

```
import pandas as pd
import time

df = pd.DataFrame({"col":[1,2,3,4]})
print(df)

df = pd.DataFrame([[1,2,3],[4,5,6],[7,8,9]])
print(df)
```

```
import pandas as pd
import time

mylist=[]

for i in range(100):
    mylist.append(i)

df = pd.DataFrame({"세로줄":mylist})
print(df)
print(df.head)
df.to_excel("엑셀연습.xlsx")
```

💡 pandas는 전에 해본적이 있어서 친숙한 느낌?

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
import urllib.request
import pandas as pd
import time

driver = webdriver.Chrome()
sea = input('검색어를 입력해주세요').replace(' ','+')
url = f'https://www.youtube.com/results?search_query={sea}'

driver.get(url)
time.sleep(1)
data_list =[]
for i in range(10):
    data = {}
    item = driver.find_elements(By.CSS_SELECTOR,'#video-title
    item_name = item.text
    url = item.get_attribute("herf")
    if url:
        data['title'] = item_name
        data['url'] = url
        data_list.append(data)
    else:
        continue

df = pd.DataFrame(data_list)
df.to_excel(f"youtube_{sea}_result.xlsx")
time.sleep(100)
```

💡 selenium + pandas

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time
import pandas as pd

driver = webdriver.Chrome()
search = input('검색어를 입력하세요: ').replace(' ', '+')

driver.get(f"https://search.naver.com/search.naver?where=news
data_list = []
time.sleep(1)

# 요소를 찾아서 카피 해옴. 웹브라우저 + 개발자 도구
selector = "news_tit"

# 찾아온 요소를 find_element로 가져오기
items = driver.find_elements(By.CLASS_NAME, selector)

time.sleep(1)

for i in range(len(items)):
    data = {}
    item_name = items[i].text
    url = items[i].get_attribute("href")
    if url:
        data['title'] = item_name
        data['url'] = url
        data_list.append(data)
        #print(f"{item_name} : {url}")
    else:
        continue

df = pd.DataFrame(data_list)
#print(data_list)
print(df)
```

```
# 엑셀로 저장
df.to_excel(f"naver_{search}_result.xlsx")
```

💡 SQL 써봐야지

```
MySQL 연결: conn = pymysql.connect(연결 옵션)
커서 생성하기: cur = conn.cursor()
테이블 만들기: cur.execute("CREATE TABLE 문장")
데이터 입력하기: cur.execute("INSERT 문장")
입력 데이터 저장: conn.commit()
MySQL 연결 종료: conn.close()
```

```
import pymysql
conn = pymysql.connect(host='localhost',user='root',password=
cur = conn.cursor()

sql = """CREATE TABLE stu(
    uid INT PRIMARY KEY AUTO_INCREMENT,
    nick VARCHAR(20) NOT NULL,
    pwd VARCHAR(20) NOT NULL,
    phon VARCHAR(20) NOT NULL,
    addr VARCHAR(20) NOT NULL
);
"""
cur.execute(sql)

conn.rollback() #위에서 실행한 쿼리를 취소

conn.commit() #위에서 실행한 쿼리를 반영(저장)

conn.close #연결 종료
```

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost',user='root',password=
cur = conn.cursor()
sql ="SELECT * FROM stu"
cur.execute(sql)
data = cur.fetchall() #
print(data)

data = pd.DataFrame(data)
print(data)
```

💡 pandas로 이쁘게

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost',user='root',password=
cur = conn.cursor()
sql ="SELECT * FROM Country"
cur.execute(sql)
data = cur.fetchall() #
print(data)

data = pd.DataFrame(data)
print(data)
```

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost',user='root',password=
cur = conn.cursor()
```

```
sql ="SELECT * FROM Country Limit 10;"
cur.execute(sql)
data = cur.fetchall()
cols = [desc[0] for desc in cur.description]
print(cols)
data = pd.DataFrame(data,columns=cols)
print(data)
```

💡 pandas columns로 칼럼 이름 추출 해서 바꾸기

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost',user='root',password=
cur = conn.cursor()

def tb_search(tb,num):
    sql = f"select * from {tb} limit {num};"
    cur.execute(sql)
    data = cur.fetchall()
    print(data)
```

💡 이런식으로 함수로 만들어서 편하게 접근도 가능

📢 오늘로 파이썬 끝!

어셈블리8086 대소문자 변환

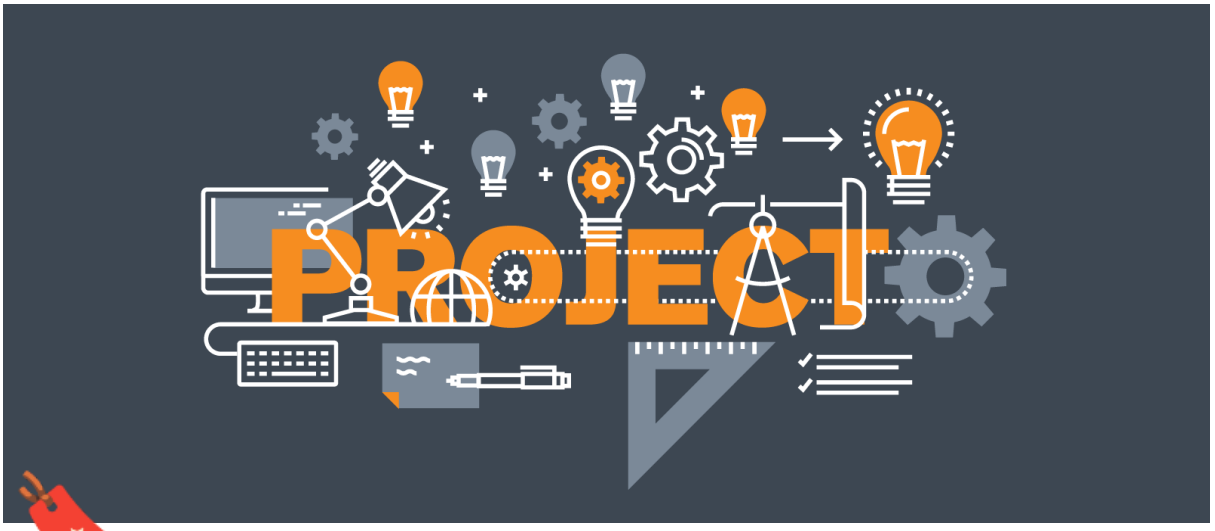
```
.model small
.code
main proc

    binput:
    mov ah,1    ;입력 서비스 번호 1을 호출(al에 저장)
    int 21h     ;인터럽트(입출력)

    cmp al,1ah  ;키보드 작동(ctrl+z)비교
    je end      ;ctrl+z누르면 end로 가서 프로시저를 종료함

    cmp al,'Z'  ;입력한값이 대문자알파벳인지 아스키코드로 범위 확인
    ja bprint   ;범위 밖이면 변환하지 않고 건너뜀

    cmp al,'A'  ;입력한값이 대문자알파벳인지 아스키코드확인 범위 확인
    jb bprint   ;범위 밖이면 변환하지 않고 건너뜀

    badd:
    add al,20h  ;입력 받은 값에 20h를 더해서 소문자로 변환

    bprint:
    mov dl,al   ;dl에서 출력 하기 때문에 입력받은 값이 저장되있는 al값
    mov ah,2    ;출력 서비스 번호 2를 호출(dl에 저장)
    int 21h     ;인터럽트(입출력)

    cmp dl,7Eh  ; ~ 입력 시 소문자 변환으로 이동
    je sinput

    cmp dl,1ah   ;무한 루프 돌리기용 아무거나
    jne binput
```

```
    sinput:
    mov ah,1    ;입력 서비스 번호 1을 호출(al에 저장)
    int 21h     ;인터럽트(입출력)

    cmp al,1ah  ;키보드 작동(ctrl+z)비교
    je end      ;ctrl+z누르면 end로 가서 프로시저를 종료함

    cmp al,'z'  ;입력한값이 소문자알파벳인지 아스키코드로 범위 확인
    ja sprint   ;범위 밖이면 변환하지 않고 건너뜀

    cmp al,'a'  ;입력한값이 소문자알파벳인지 아스키코드로 범위 확인
    jb sprint   ;범위 밖이면 변환하지 않고 건너뜀

    sadd:
    sub al,20h  ;입력 받은 값에 20h를 더해서 대문자로 변환

    sprint:
    mov dl,al   ;dl에서 출력 하기 때문에 입력받은 값이 저장되있는 al값
    mov ah,2    ;출력 서비스 번호 2를 호출(dl에 저장)
    int 21h     ;인터럽트(입출력)

    cmp dl,7Eh  ; ~ 입력 시 대문자 변환으로 이동
    je binput

    cmp dl,1ah   ;무한 루프 돌리기용 아무거나
    jne sinput

    end:

    main endp
end main
```

어셈블리어 암호

```
.model small
.data
msg1 db 'Input P.W.=>$'
msg2 db 0ah, 0dh,'OK!$'
.code
main proc

    mov ax,@data
    mov ds,ax

    mov ah,9
    lea dx,msg1
    int 21h

    re:
    mov ah,1
    int 21h

    cmp al,'s'
    jne beep

    int 21h

    cmp al,'m'
    jne beep

    int 21h

    cmp al,'c'
    jne beep
```

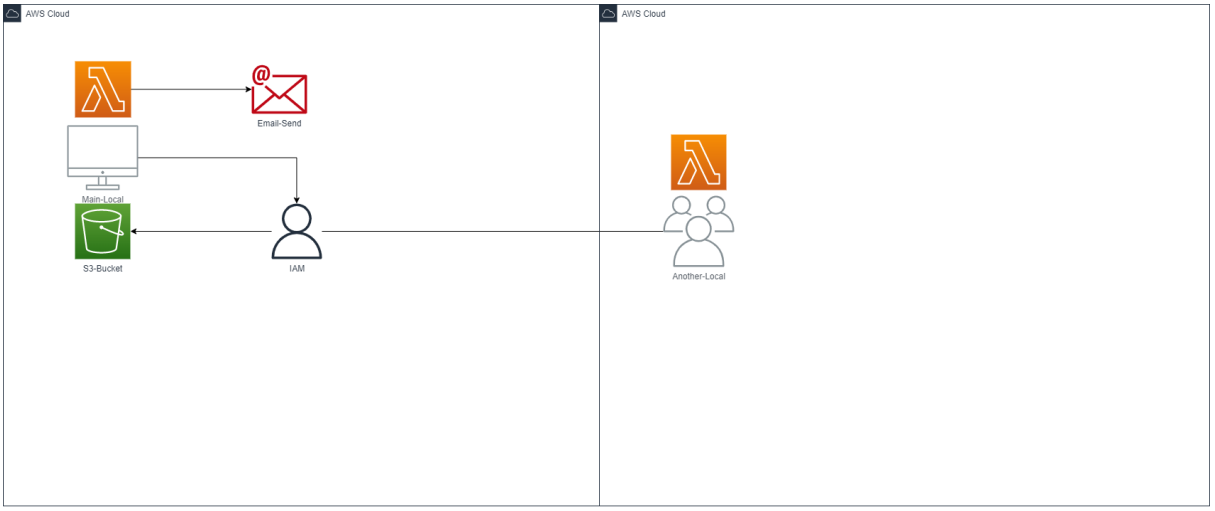
```
    mov ah,9
    lea dx,msg2
    int 21h

    mov ah,4ch
    int 21h

main endp

sound proc

    beep:
    mov ah,2
    mov dl,7
    int 21h
    jmp re

sound endp
end main
```



AWS_Daily_Cost

STACK AWS Python

1.구축?설계?



설명을 위해 자동으로 이메일을 보내는 Lambda함수를 설정할 계정을 “Main-Local” 이라고 지칭하고

그 이외의 다른 계정들은 “Another-Local” 이라고 지칭하겠습니다.

구성: Main-Local 에서는 Lambda를 통해 파일을 S3에 올립니다Another-Local은 Main-Local의 IAM 유저을 통하여 Main-Local의 S3에 업로드 합니다. 이후 Main-Local에서 파일을 모두 합치고 정렬하여 Email로 보냅니다. (모든 작업은 Lambda로 구성하였습니다.)

2.Main-Local 세팅

Main-Local에는 총 3개의 Lambda 함수와 버킷을 구성합니다.

2.1 Daily-Cost 파일 업로드 할 버킷 만들기

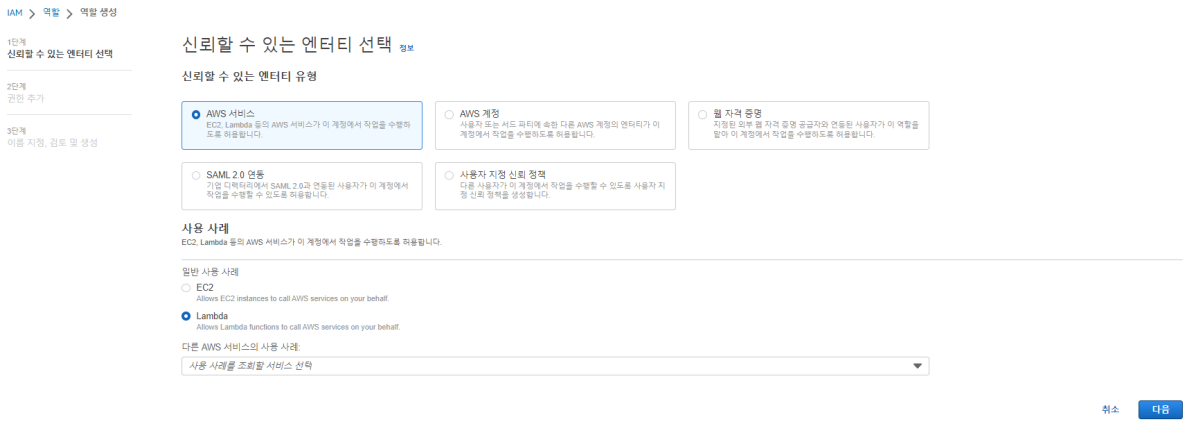
먼저 파일을 저장할 S3 버킷을 만들겠습니다.

S3 콘솔 → 버킷 → 버킷 만들기를 누릅니다.

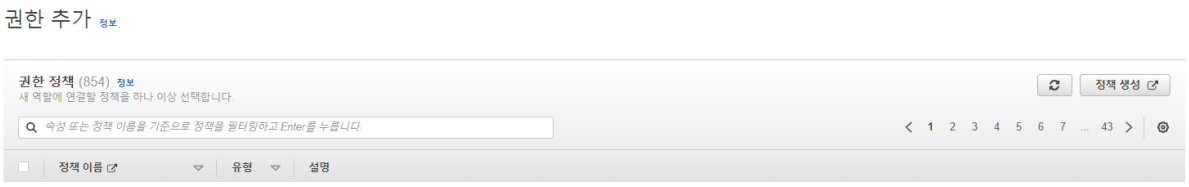
이름만 정하면 됩니다. 나머지는 다 기본으로 되있는 걸로 합니다. 코드에서 버킷 이름을 통해 파일을 업로드 하므로 “daily-cost-s3” 로 해주세요. (다른 이름으로 하면 나중에 Lambda 함수 코드를 고쳐야 합니다.)

2.2 Daily-Cost 역할 및 함수 만들기

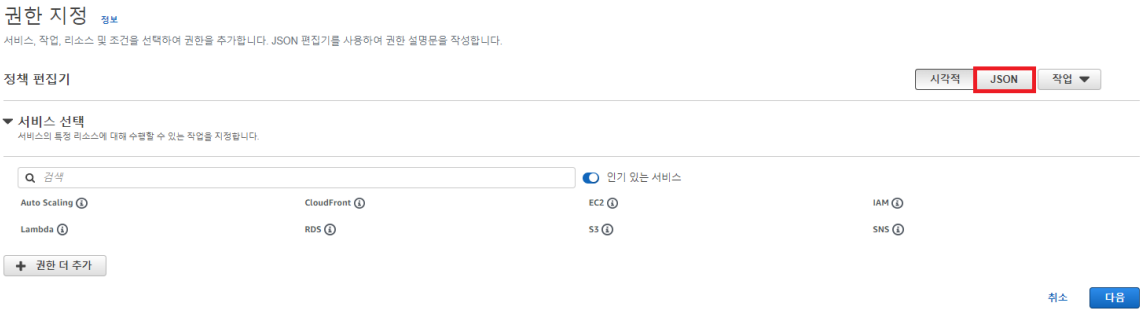
이제 파일을 저장할 S3 버킷이 만들어 졌으니 Lambda 함수를 실행할 IAM역할을 만들겠습니다.



IAM → 역할 → 역할 생성



정책 생성 클릭



JSON 클릭 후 밑의 텍스트를 복사

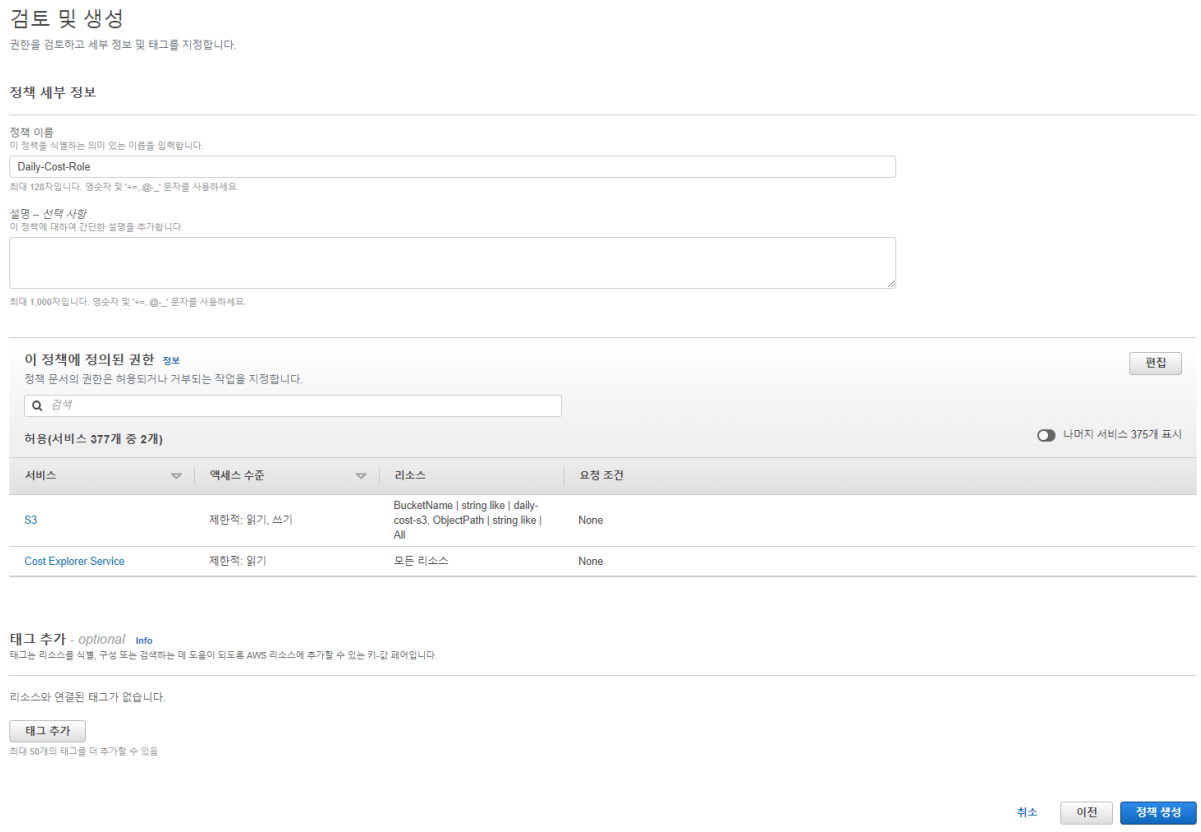
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ce:GetCostAndUsage"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::daily-cost-s3",
                "arn:aws:s3:::daily-cost-s3/*"
            ]
        }
    ]
}
```

만약 S3 버킷의 이름이 다른 경우 마지막 줄을 그에 맞는 이름으로 바꾸어야 합니다.

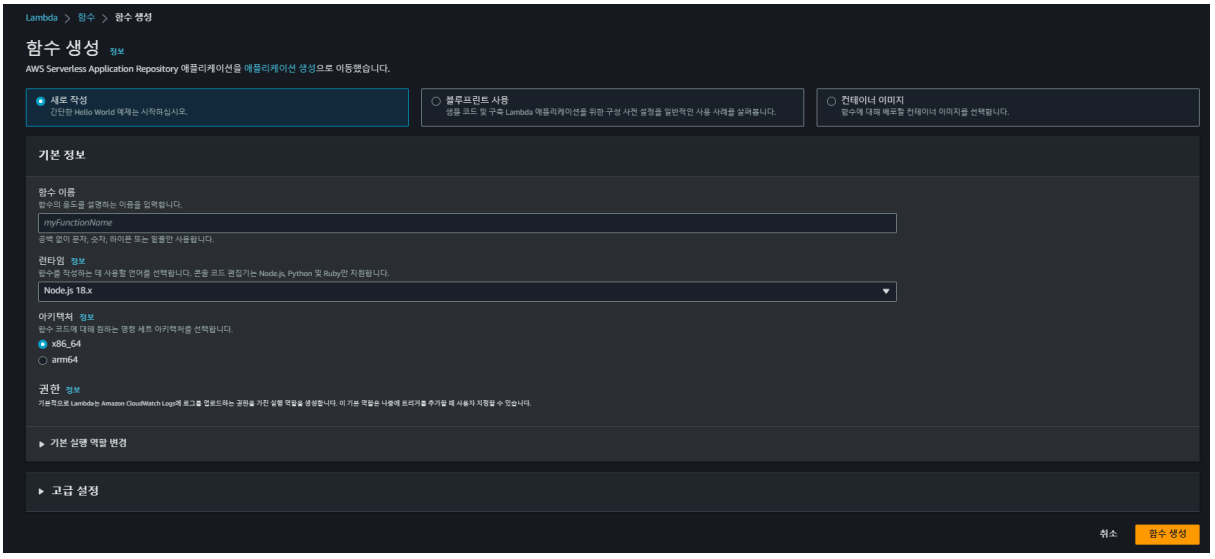
예시)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

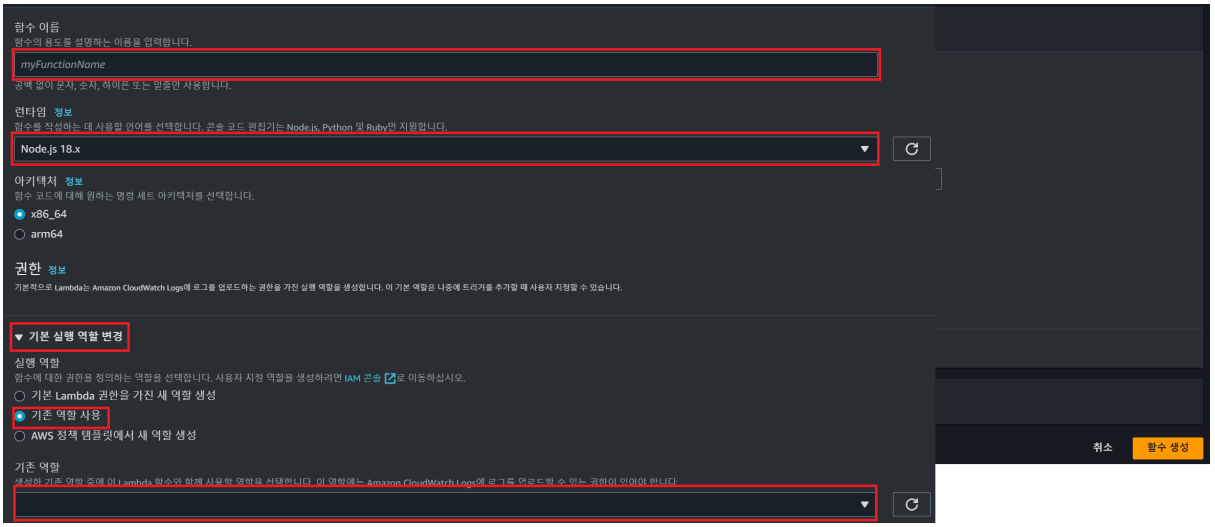
```
                "ce:GetCostAndUsage"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<버킷 이름>",
                "arn:aws:s3:::<버킷 이름>/*"
            ]
        }
    ]
}
```



이름을 지정해줍니다.(저는 알아보기 쉽게 Daily-Cost-Role 로 하였습니다.) 정책 생성을 눌러주면 끝입니다.



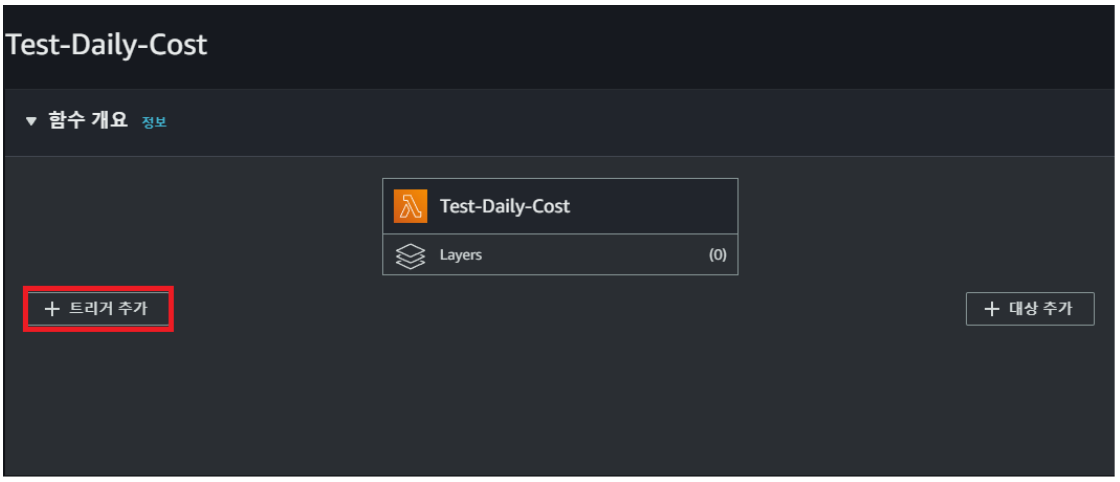
Lambda → 함수 → 함수 생성을 누릅니다.



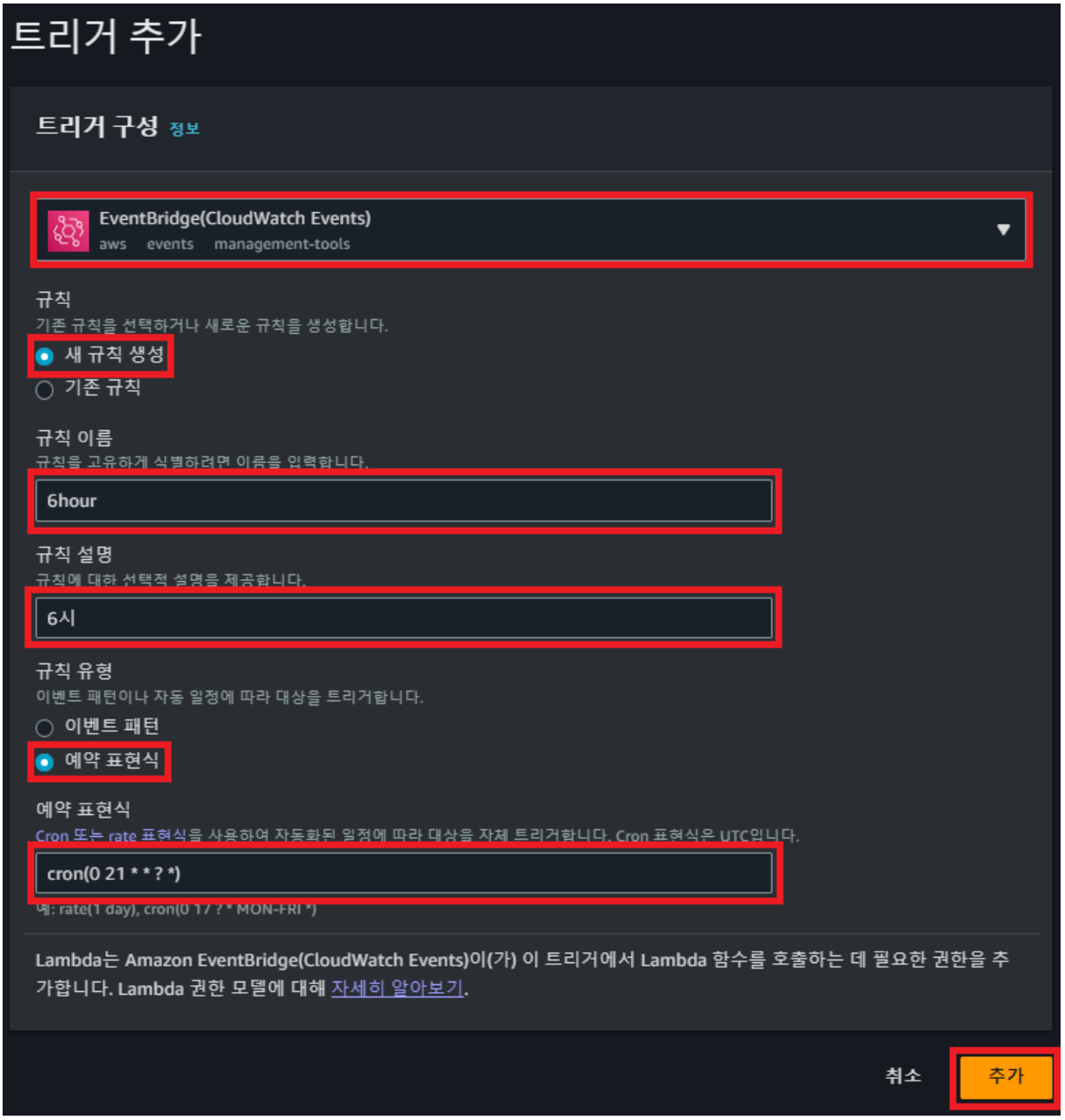
함수 이름을 설정해줍니다. (편하게 구분할 수 있게 **lambda_daily_cost** 로하겠습니다.)

런타임은 Python 3.9 로 지정해줍니다.

기본 실행 역할 변경 → 기존 역할 사용 → 전에 만든 역할 선택 → 함수 생성



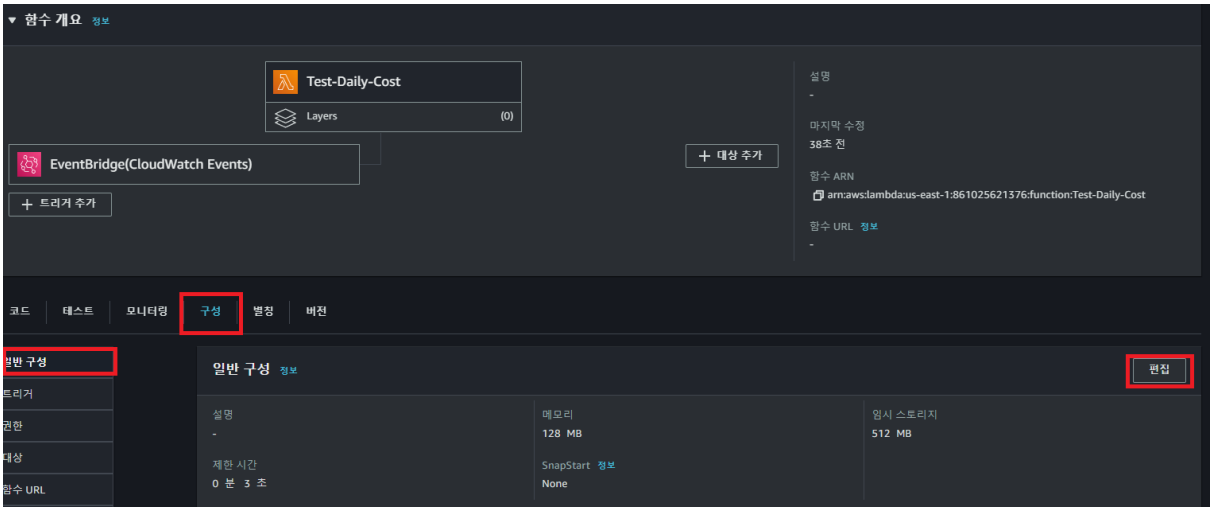
함수 → 트리거 추가 → EventBridge



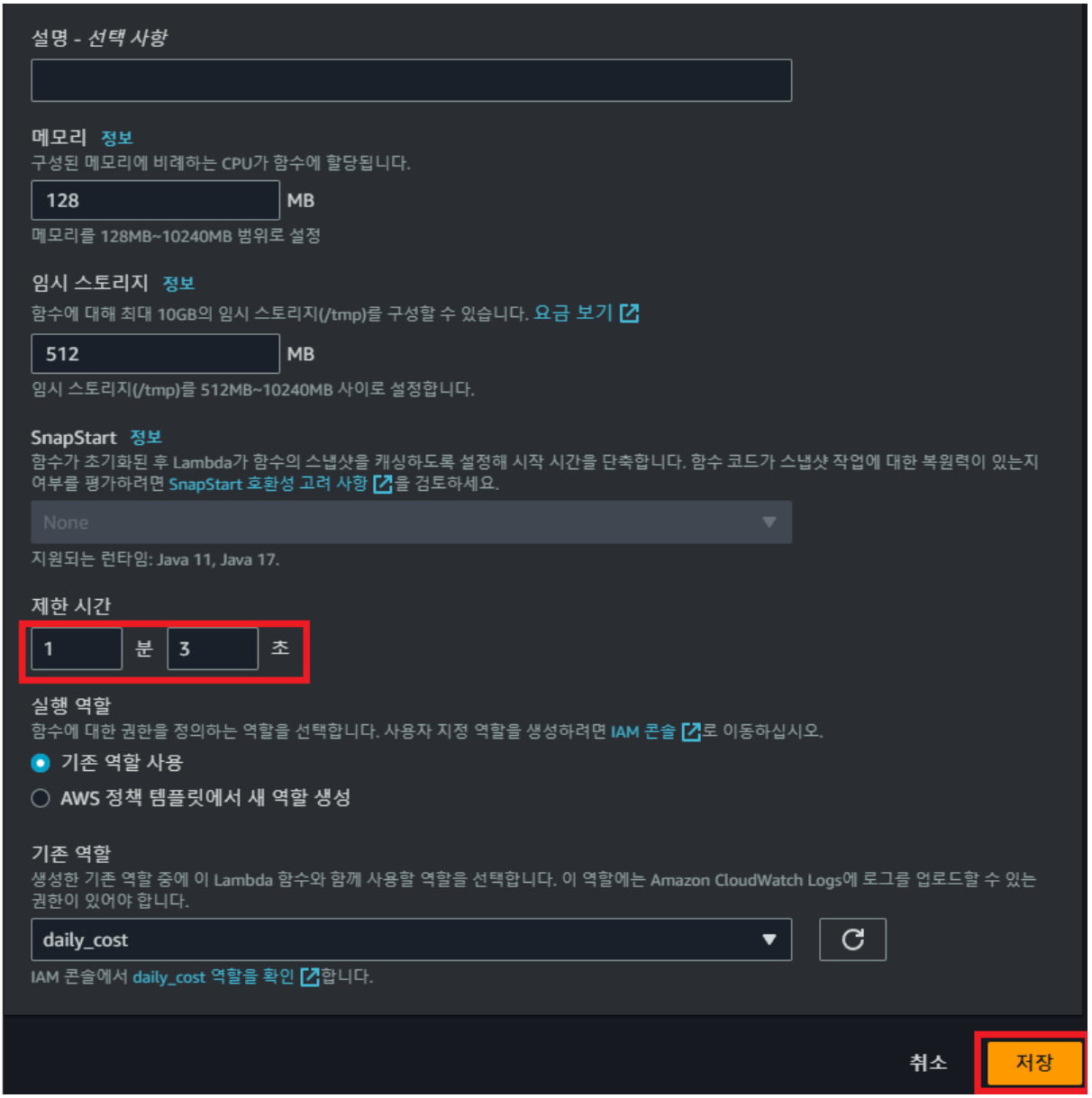

```
cron(0 21 ? * TUE-SAT *)
```

이 cron 식은 매주 화요일 부터 토요일 (화요일 부터 토요일 인 이유는 금액 정보가 어제 하루 기준이기 때문 입니다.) 까지 lambda 함수 지역시간 21 시에 실행되게 합니다.

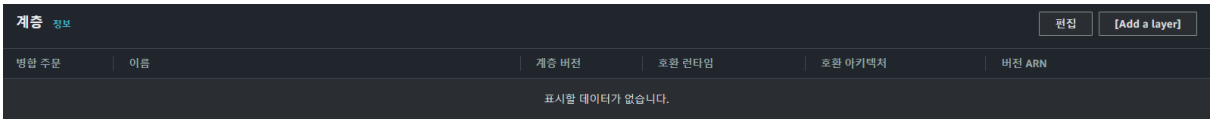
(lambda 함수를 만들 때 만든 지역 기준의 시간 입니다. 저는 버지니아 북부(us-east-1) 에 생성해서 지역시간 21시면 한국 시간 6시 입니다.)


함수 개요 정보
Test-Daily-Cost
Layers (0)
EventBridge(CloudWatch Events)
+ 트리거 추가
+ 대상 추가
설명
-
마지막 수정
38초 전
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:Test-Daily-Cost
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
일반 구성
트리거
권한
대상
함수 URL
일반 구성 정보
편집
설명
-
메모리
128 MB
임시 스토리지
512 MB
제한 시간
0 분 3 초
SnapStart 정보
None

구성 → 일반 구성 → 편집

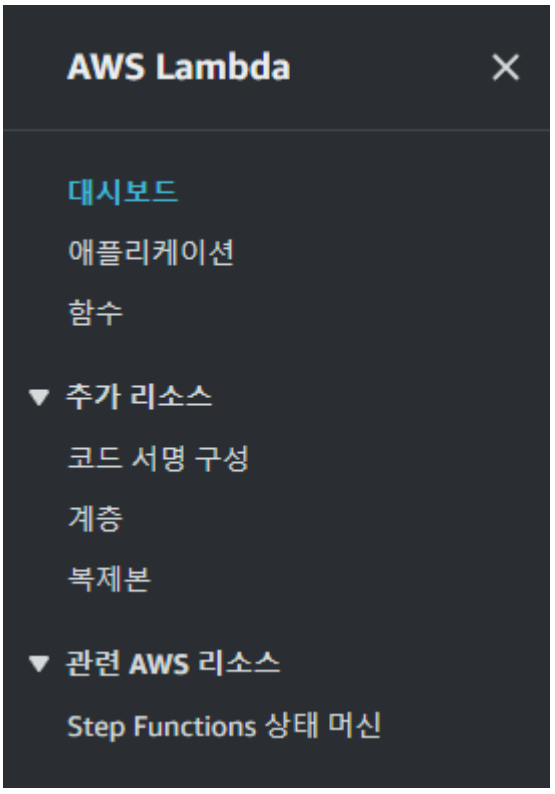


제한 시간 1 분 이상으로 늘리고 저장

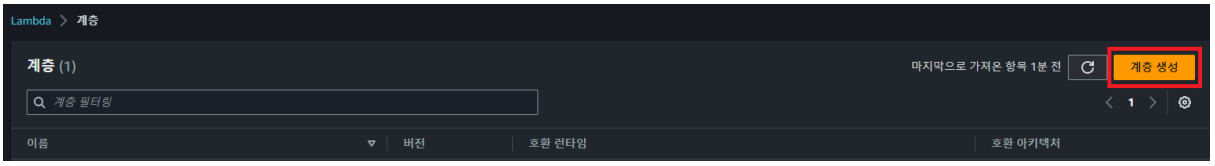


이제 계층을 추가 합니다. 이 함수에서 추가해야 하는 계층은 총 2개입니다.

먼저 사용자 지정 계층을 추가 하겠습니다.



Lambda 에서 계층 클릭



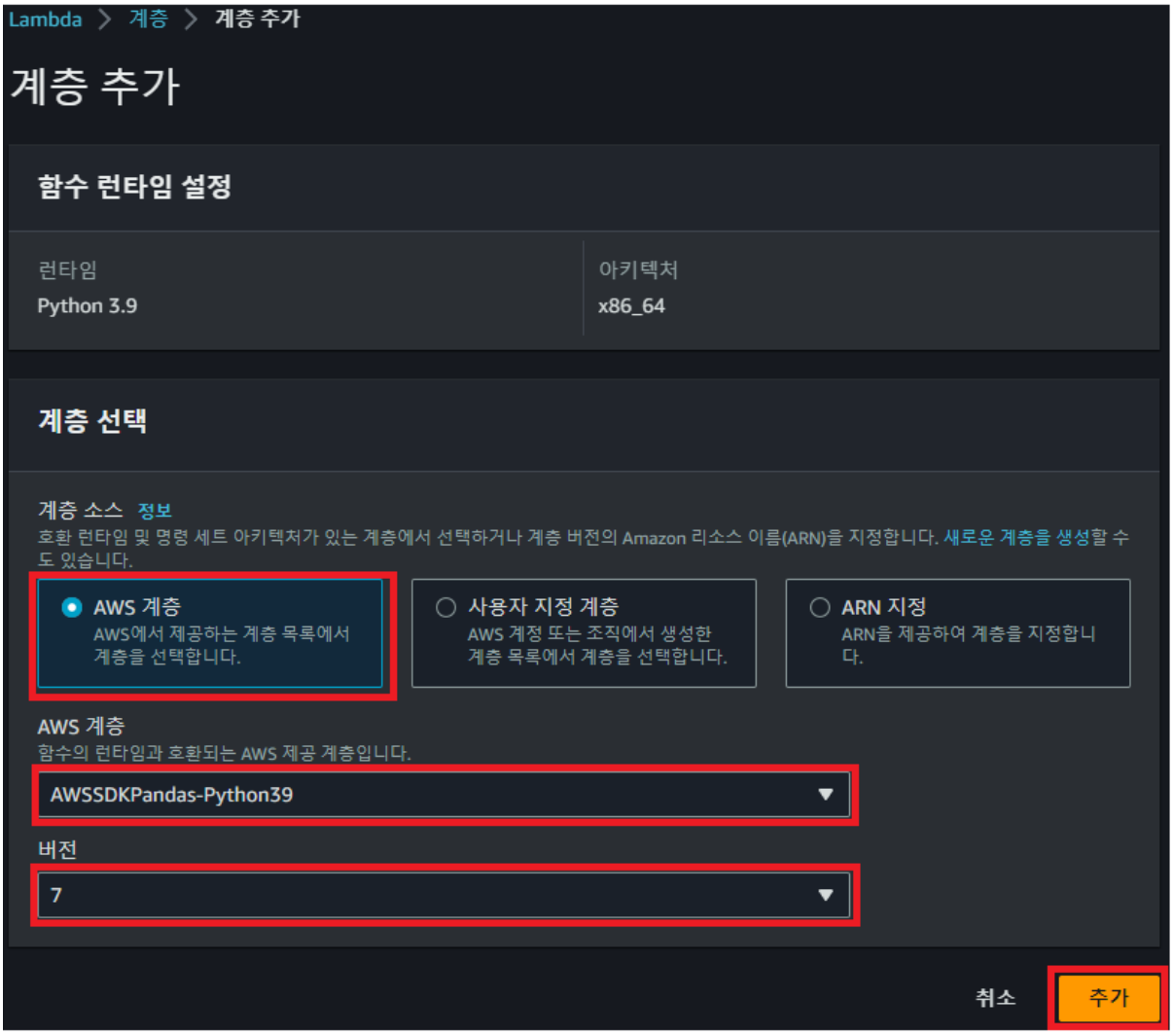
계층 생성

이름 지정 .zip 파일 업로드 → 업로드 파일(아래 파일 업로드) → 아키텍처 → 런타임 → 생성

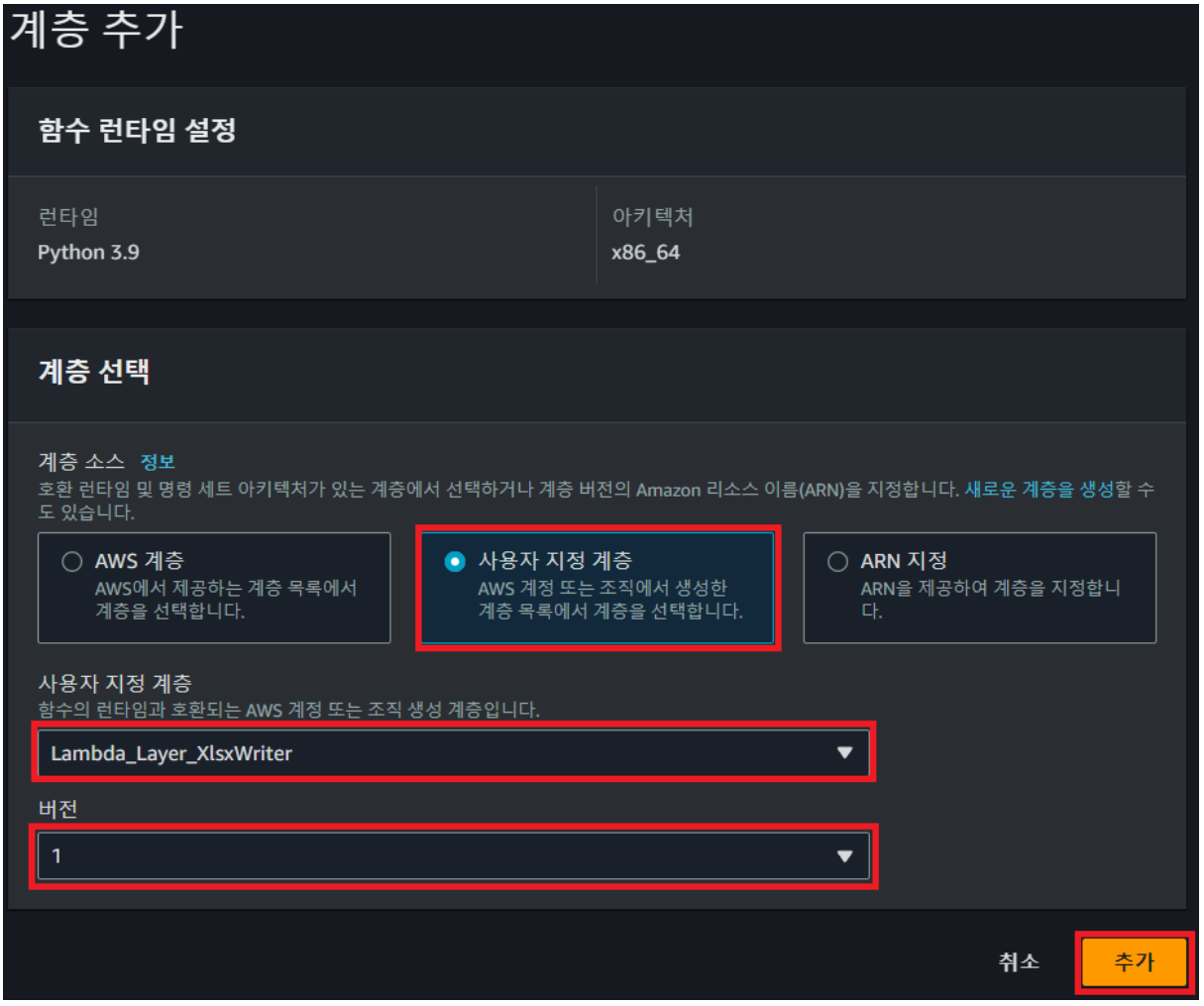
python.zip

다시 람다 함수로 돌아옵니다.

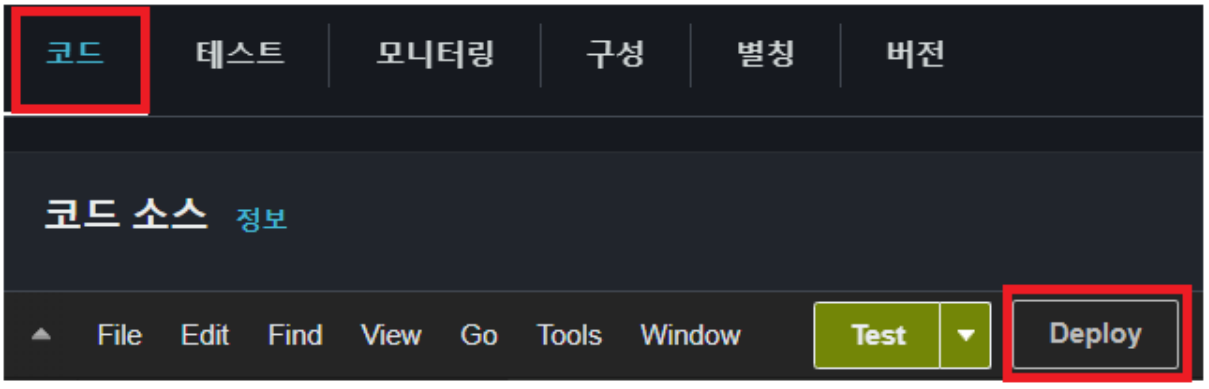
밑으로 쭉 스크롤 내리고 계층에서 Add a layer



aws 계층 → AWSSDKPandas-Python39 → 버전 → 추가



사용자 지정 계층 → 전에 만든 계층 → 버전 선택 → 추가



코드 → 코드 소스에 아래 코드 복사 붙여 넣기 → Deploy

```
import boto3
import datetime
```

```
import io
import pandas as pd

def lambda_handler(event, context):
    # AWS 청구서 클라이언트 생성
    client = boto3.client('ce')

    # 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 그
            },
            {
                'Type': 'DIMENSION',
                'Key': 'LINKED_ACCOUNT'  # 링크된 어카운트별 비용을
            }
        ]
    }

    # AWS 청구서에 요청하여 결과 가져오기
    response = client.get_cost_and_usage(**params)

    # 결과를 DataFrame으로 변환하기
    data = []
    total_cost = 0.0  # 총합 비용 초기화
    for result_by_time in response['ResultsByTime']:
```

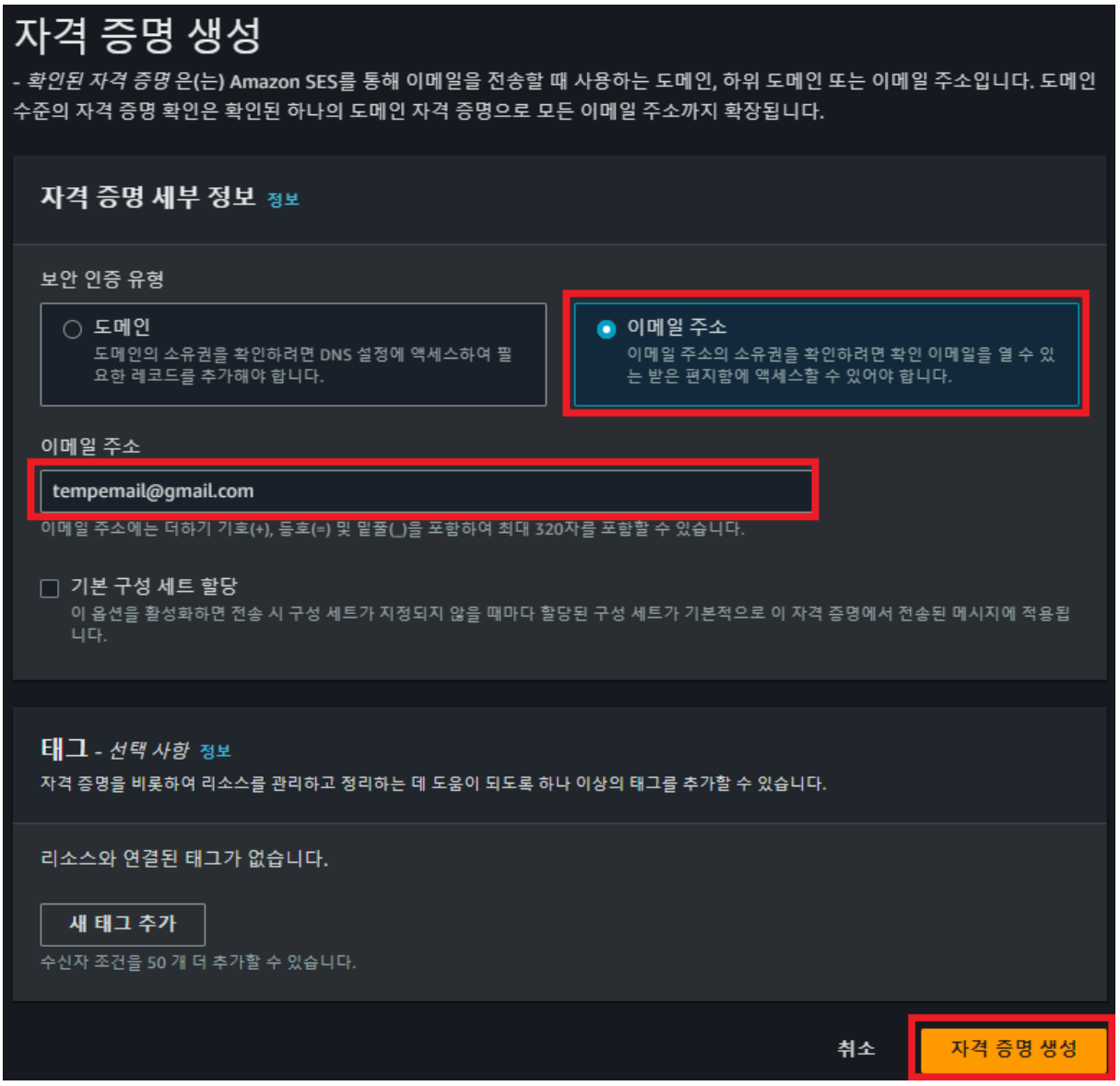
```
        date = result_by_time['TimePeriod']['Start']
        for group in result_by_time['Groups']:
            linked_account = group['Keys'][1]  # 링크된 어카운트
            service = group['Keys'][0]
            cost = group['Metrics']['UnblendedCost']['Amount'
            total_cost += float(cost)  # 비용을 누적하여 총합 비용
            data.append([linked_account, date, service, cost]
    df = pd.DataFrame(data, columns=['Linked Account', 'Date'

    # 총합 비용 행 추가
    total_row = [linked_account, date, 'Total', str(total_cos
    total_df = pd.DataFrame([total_row], columns=['Linked Acc
    df = pd.concat([df, total_df], ignore_index=True)

    # DataFrame을 Excel 파일로 저장하기
    output = io.BytesIO()
    with pd.ExcelWriter(output, engine='xlsxwriter') as write
        df.to_excel(writer, index=False, sheet_name='Sheet1')
    output.seek(0)

    # S3 버킷 객체 전부 삭제
    s3 = boto3.resource('s3')
    bucket_name = 'daily-cost-s3'
    bucket = s3.Bucket(bucket_name)
    for obj in bucket.objects.all():
        obj.delete()


    # Excel 파일을 S3 버킷에 업로드하기
    filename = 'aws_cost_{}.xlsx'.format(datetime.datetime.no
    s3.Object(bucket_name, filename).put(Body=output.getvalue
```

Deploy가 끝나면 첫 번째 함수는 끝입니다.

2.3 Send-Mail 역할 및 함수 만들기

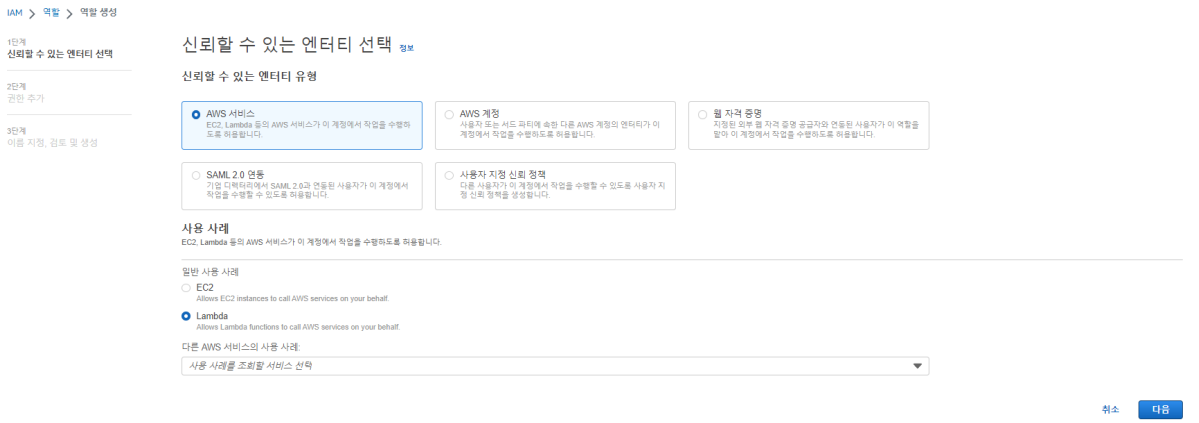
먼저 메일을 전송, 전송 받을 메일을 SES에 등록하겠습니다.

SES 콘솔 → 자격 증명 생성

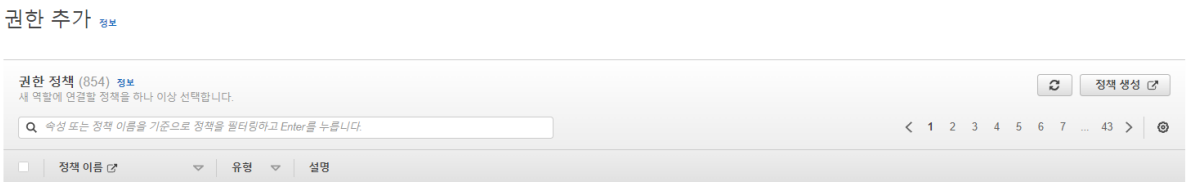


이메일을 적고 자격 증명 생성 한 개만 만들면 됩니다.

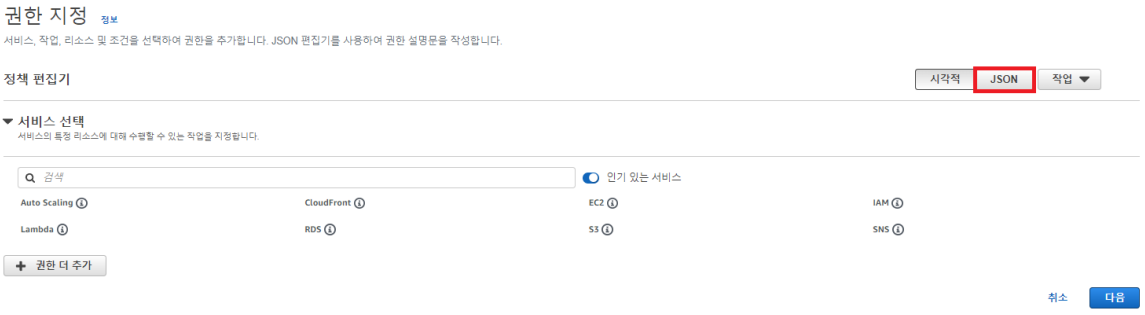
두 번째로 람다 함수를 실행 할 IAM 역할을 만들겠습니다.



IAM → 역할 → 역할 생성



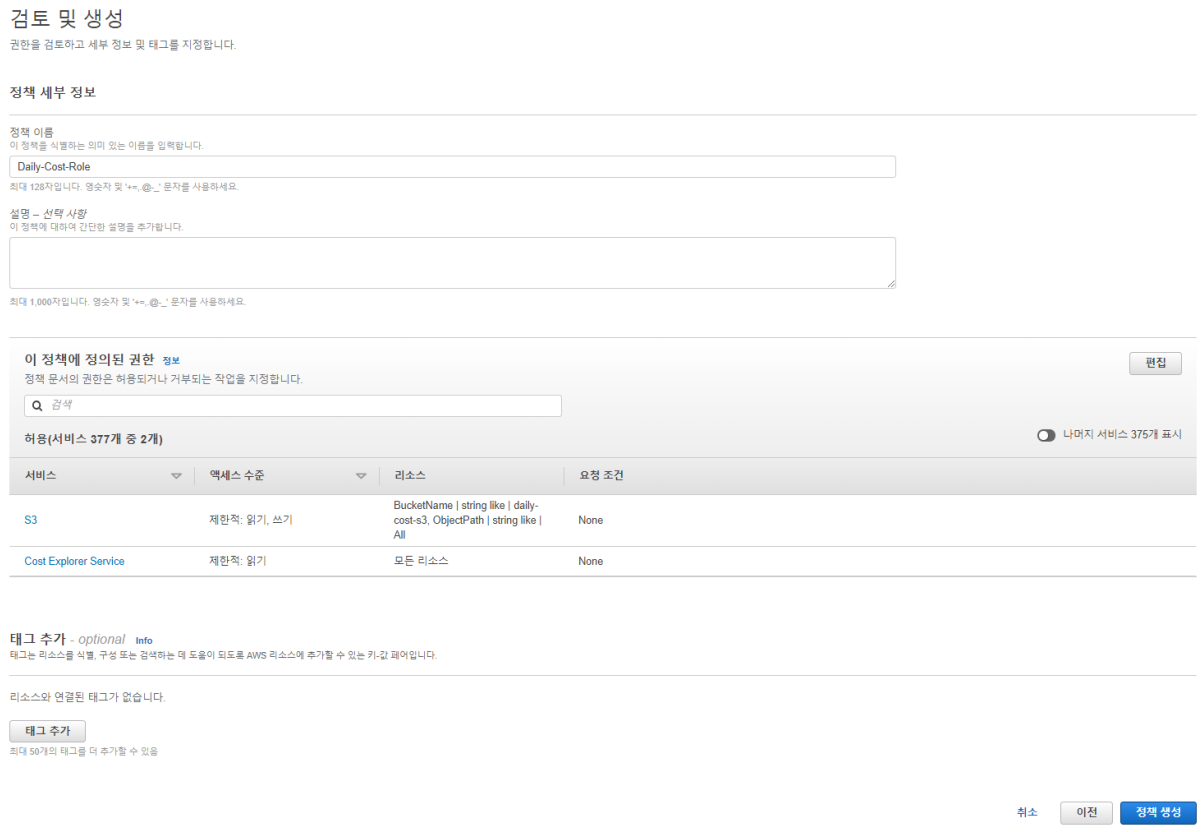
정책 생성 클릭



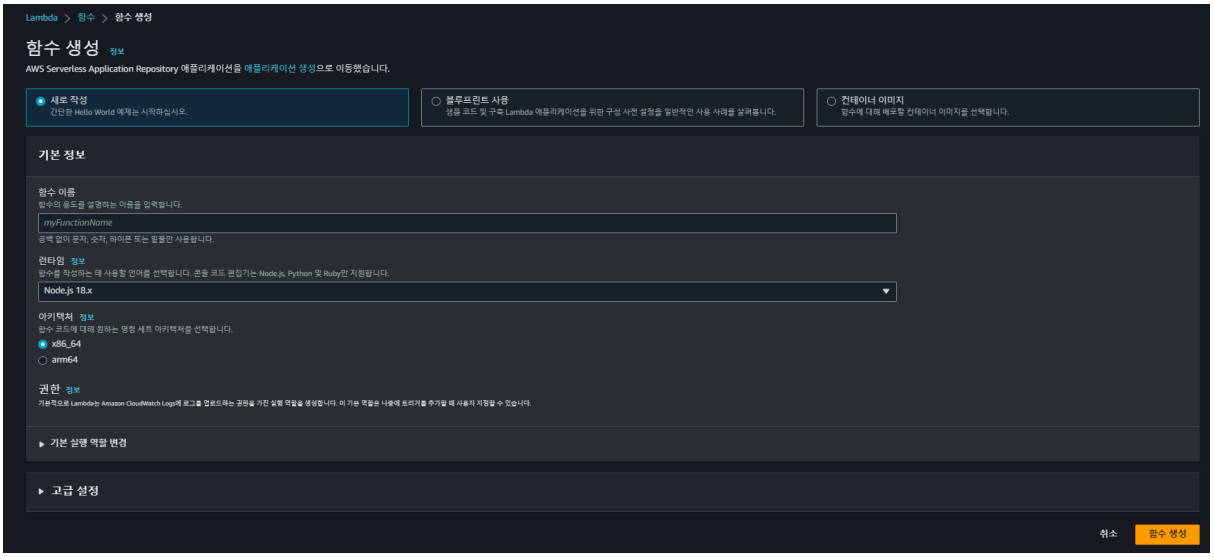
JSON 클릭 후 밑의 텍스트를 복사

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::daily-cost-s3",
                "arn:aws:s3:::daily-cost-s3/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:SendRawEmail"
            ],
            "Resource": "*"
        }
    ]
}
```

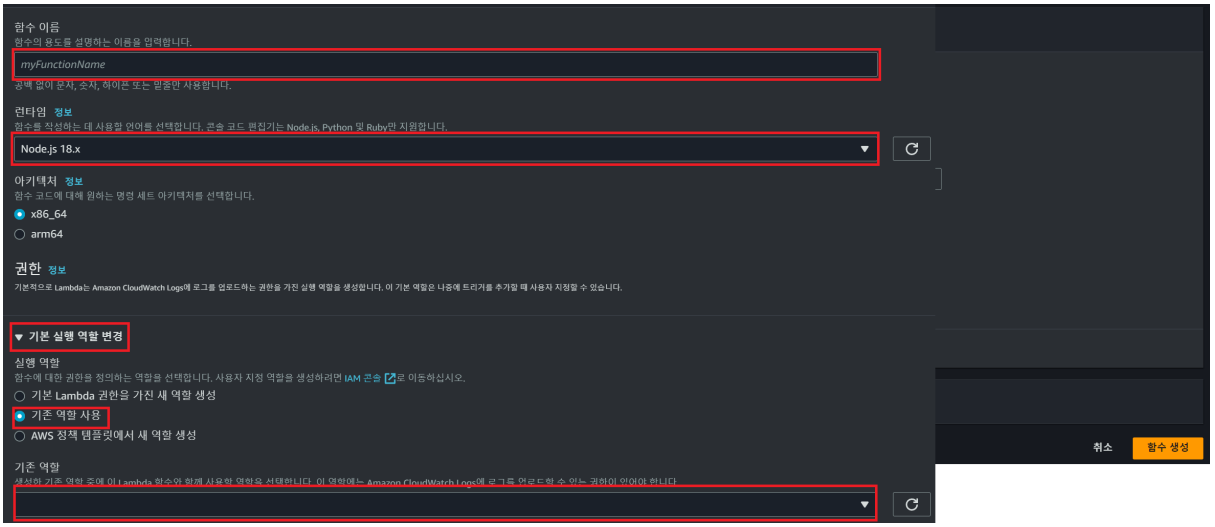
위에서 말했듯이 버킷 이름이 다르면 바꿔줘야 합니다.



이름을 지정해줍니다.(저는 알아보기 쉽게 Send-Mail-Role 로 하였습니다.) 정책 생성을 눌러주면 끝입니다.



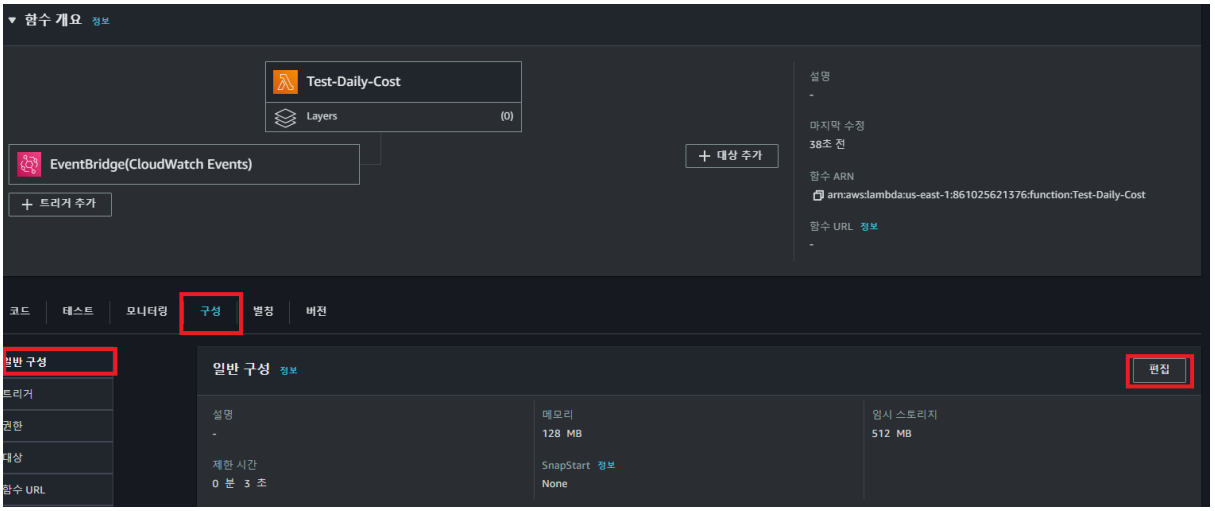
Lambda → 함수 → 함수 생성을 누릅니다.



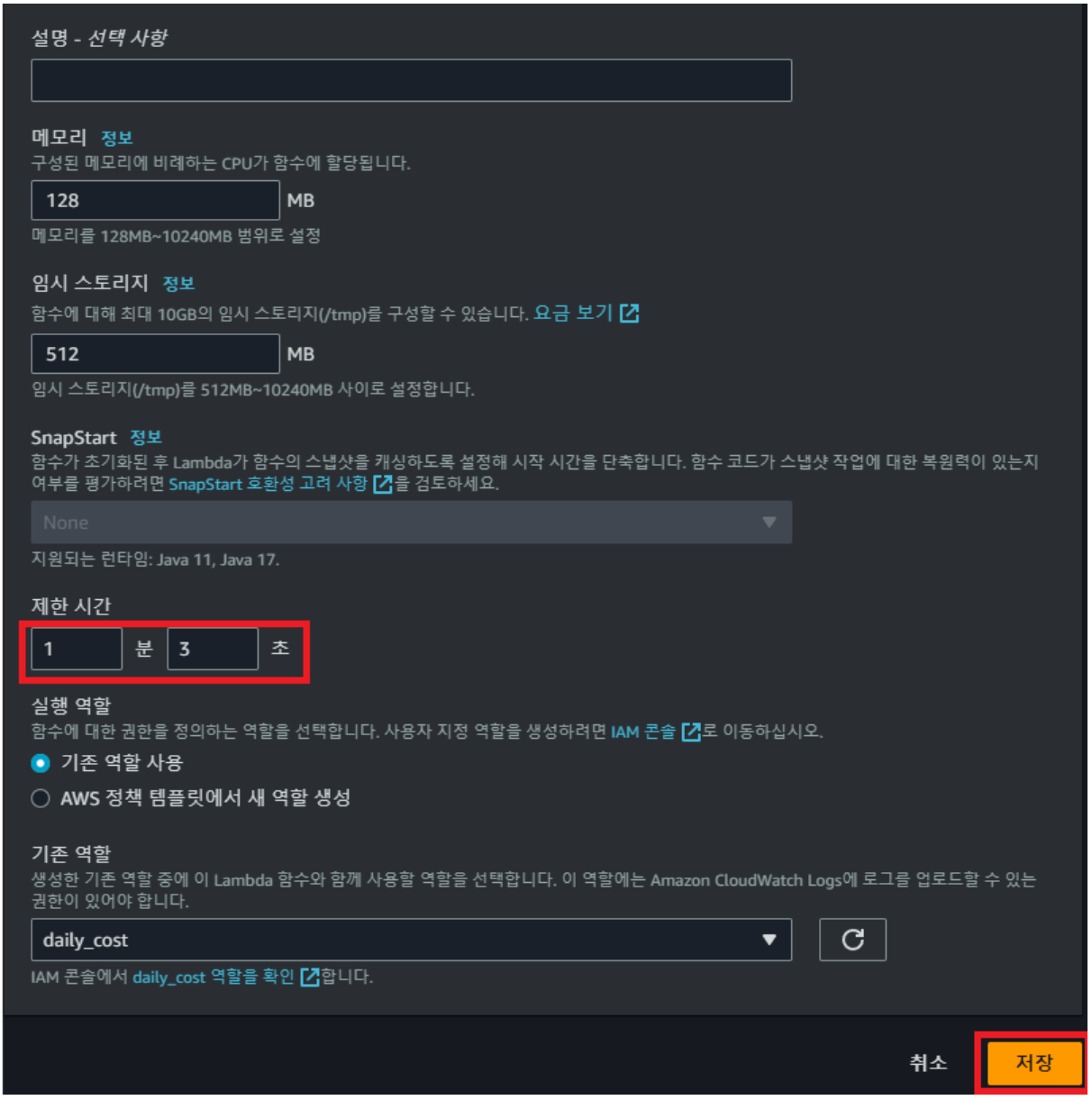
함수 이름을 설정해줍니다. (편하게 구분할 수 있게 **lambda_send_mail** 로하겠습니다.)

런타임은 Python 3.9 로 지정해줍니다.

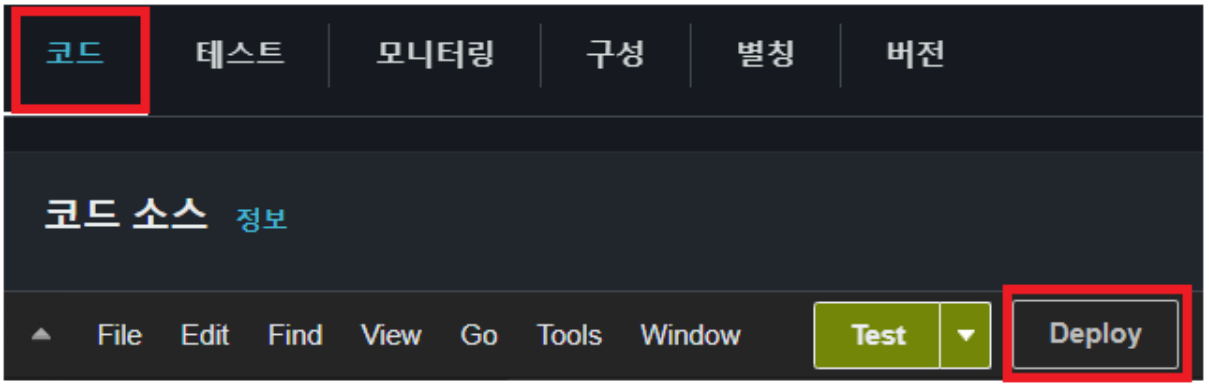
기본 실행 역할 변경 → 기존 역할 사용 → 전에 만든 역할 선택 → 함수 생성



구성 → 일반 구성 → 편집



제한 시간 1 분 이상으로 늘리고 저장



코드 → 코드 소스에 아래 코드 복사 붙여 넣기 (보내는 이메일과 받는 이메일 지정) → Deploy

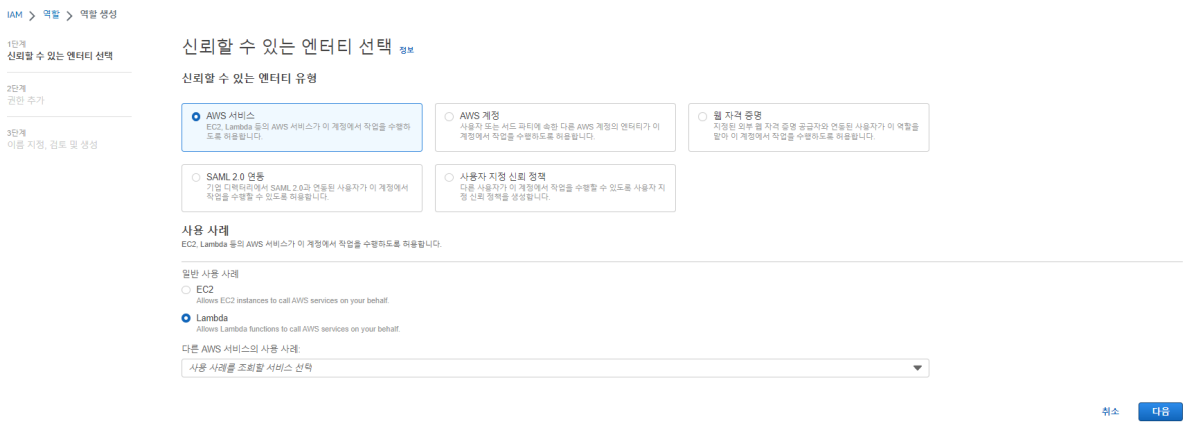
```
import boto3
import os
import email.mime.multipart
import email.mime.base
from botocore.exceptions import ClientError

# AWS SES 클라이언트 생성
ses = boto3.client('ses')

def lambda_handler(event, context):
    # S3 클라이언트 생성
    s3 = boto3.client('s3')

    # S3 버킷에서 파일 가져오기
    bucket_name = 'daily-cost-s3' #버킷 이름 다르면 바꾸어야 합니다
    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' #제목
    msg['From'] = '<보내는 이메일>' #인증 받은 메일만 가능
    msg['To'] = '<받는 이메일>' #인증 받은 메일만 가능

    # 파일 첨부
    attachment = email.mime.base.MIMEBase('application', 'oct
    attachment.set_payload(file_content)
    email.encoders.encode_base64(attachment)
    attachment.add_header('Content-Disposition', 'attachment'
    msg.attach(attachment)

    # 이메일 보내기
    try:
        response = ses.send_raw_email(
```

```
            Source=msg['From'],
            Destinations=[msg['To']],
            RawMessage={'Data': msg.as_string()}
        )
        print('Email sent')
    except ClientError as e:
        print(e.response['Error']['Message'])

    # S3 버킷 객체 전부 삭제
    s3_resource = boto3.resource('s3')
    bucket = s3_resource.Bucket(bucket_name)
    bucket.objects.delete()
```

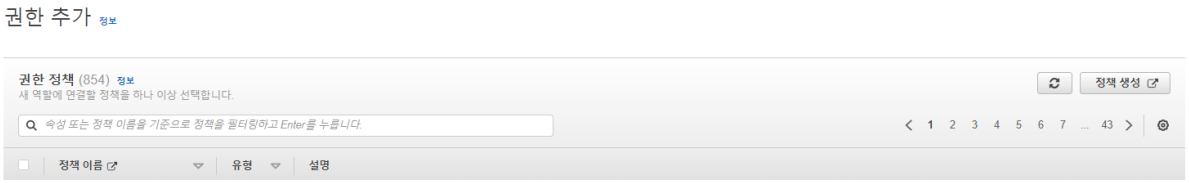
Deploy가 끝나면 두 번째 함수도 끝입니다.

2.4 Merge-File 역할 및 함수 생

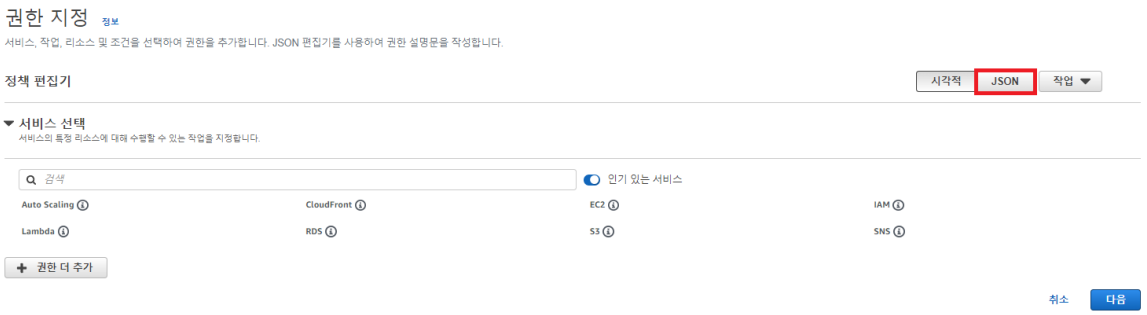
Lambda 함수를 실행할 IAM역할을 만들겠습니다.



IAM → 역할 → 역할 생성



정책 생성 클릭



JSON 클릭 후 밑의 텍스트를 복사

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ce:GetCostAndUsage"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
```

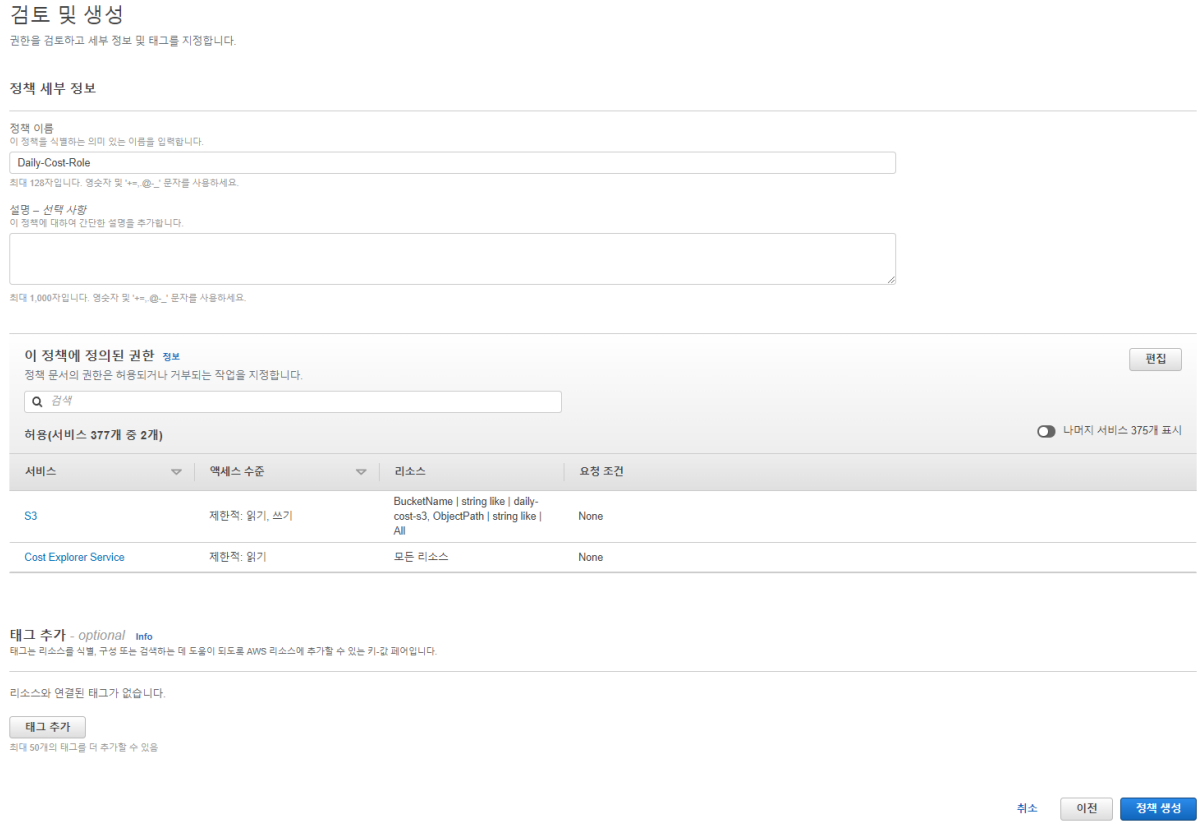
```
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::daily-cost-s3",
                "arn:aws:s3:::daily-cost-s3/*"
            ]
        }
    ]
}
```

만약 S3 버킷의 이름이 다른 경우 마지막 줄을 그에 맞는 이름으로 바꾸어야 합니다.

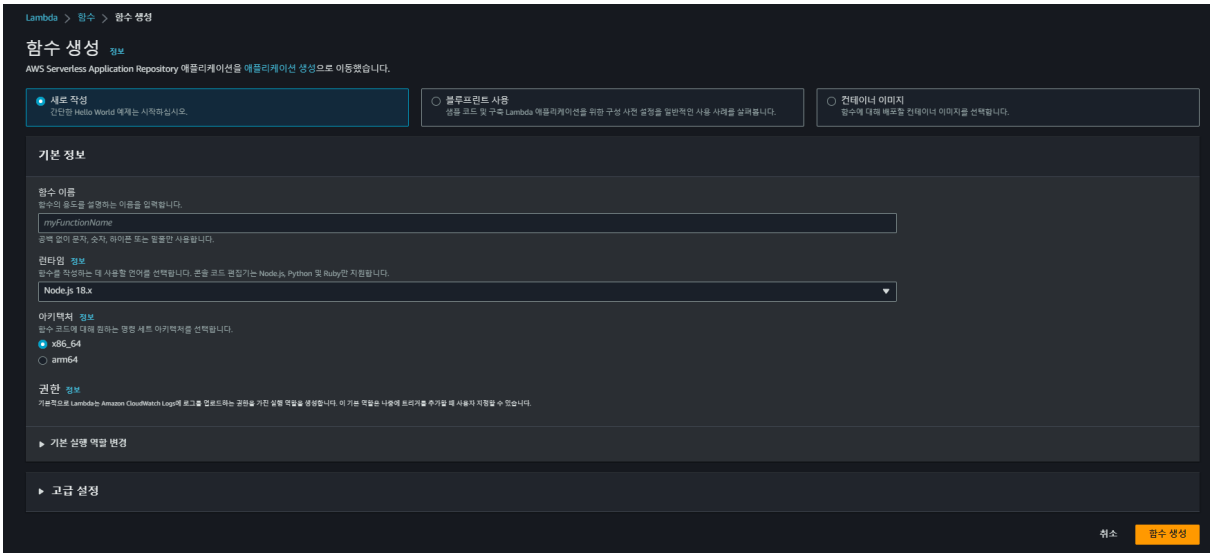
예시)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ce:GetCostAndUsage"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<버킷 이름>",
                "arn:aws:s3:::<버킷 이름>/*"
            ]
        }
```

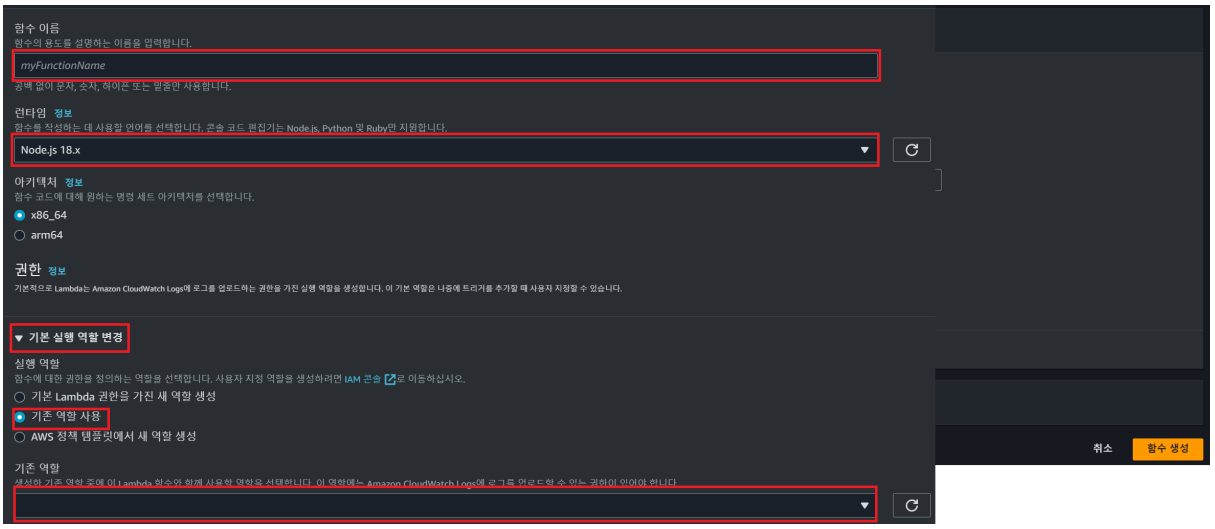
```
        ]
}
```



이름을 지정해줍니다.(저는 알아보기 쉽게 Daily-Cost-Role 로 하였습니다.) 정책 생성을 눌러주면 끝입니다.



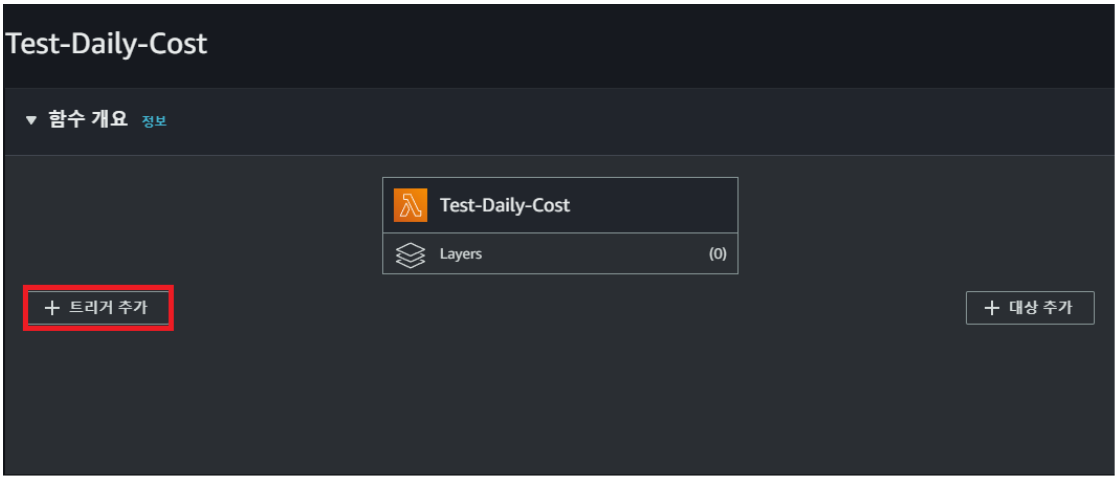
Lambda → 함수 → 함수 생성을 누릅니다.



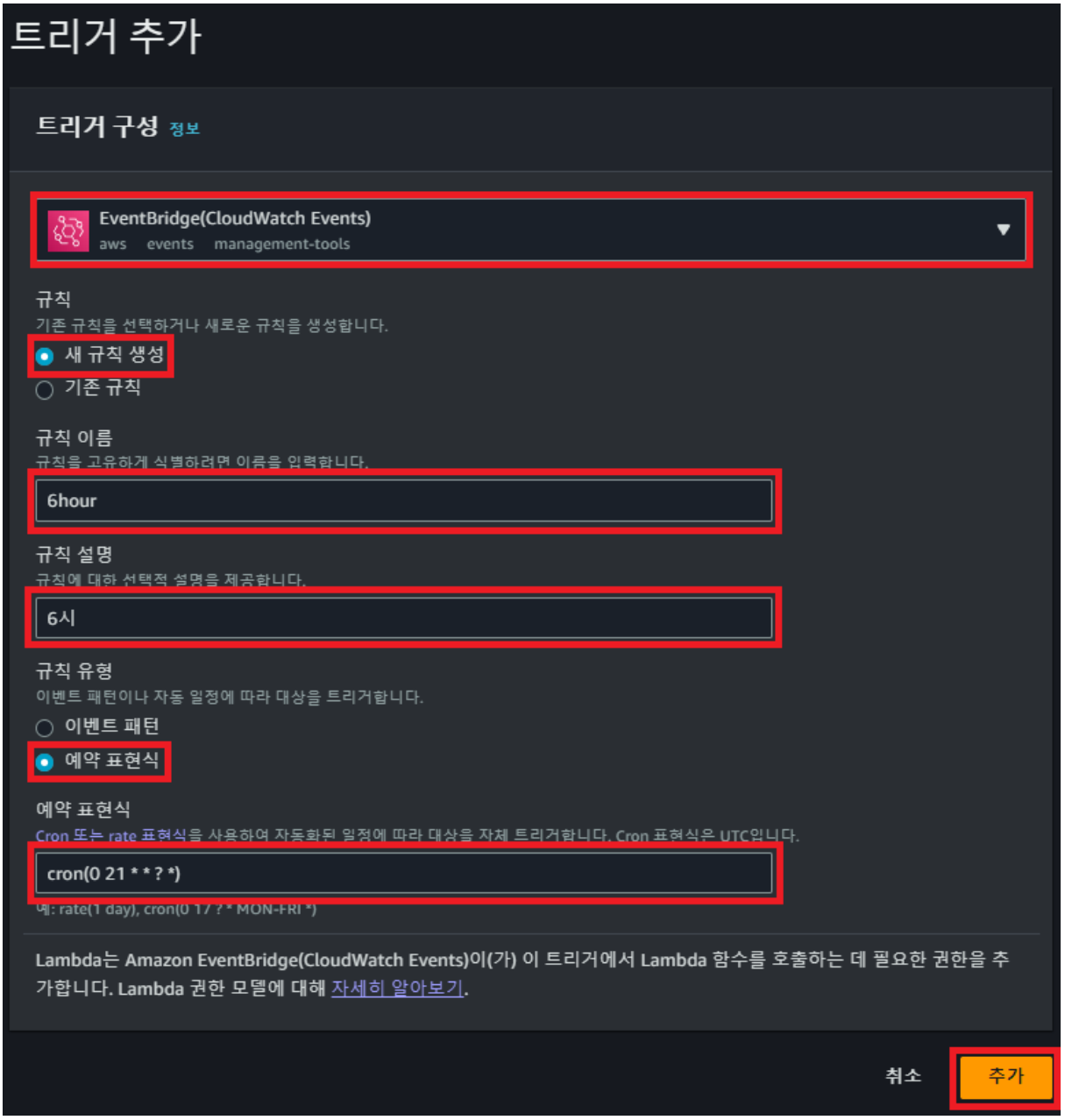
함수 이름을 설정해줍니다. (편하게 구분할 수 있게 **lambda_daily_cost** 로하겠습니다.)

런타임은 Python 3.9 로 지정해줍니다.

기본 실행 역할 변경 → 기존 역할 사용 → 전에 만든 역할 선택 → 함수 생성



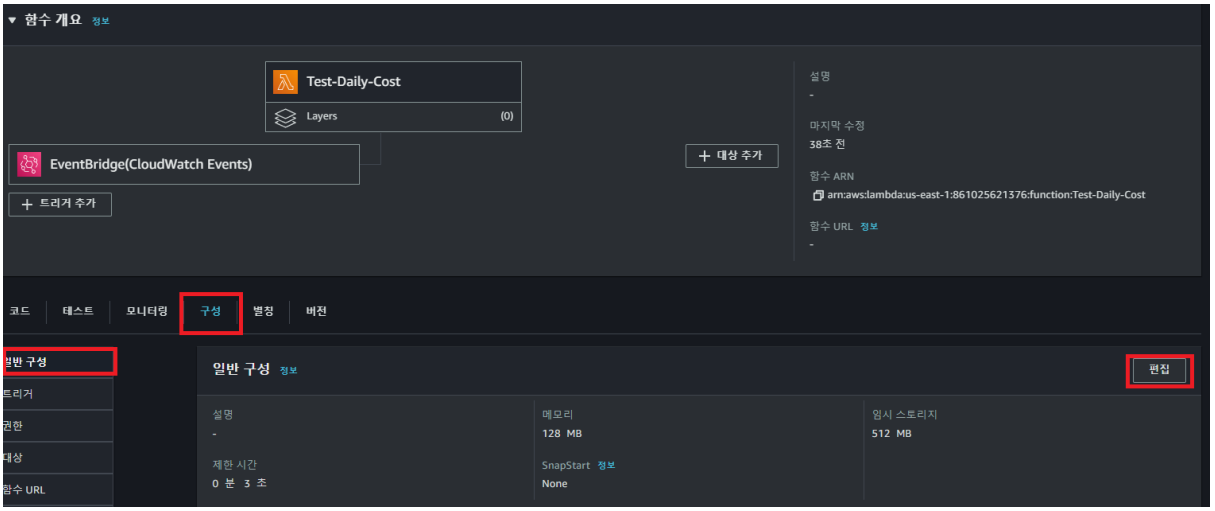
함수 → 트리거 추가 → EventBridge



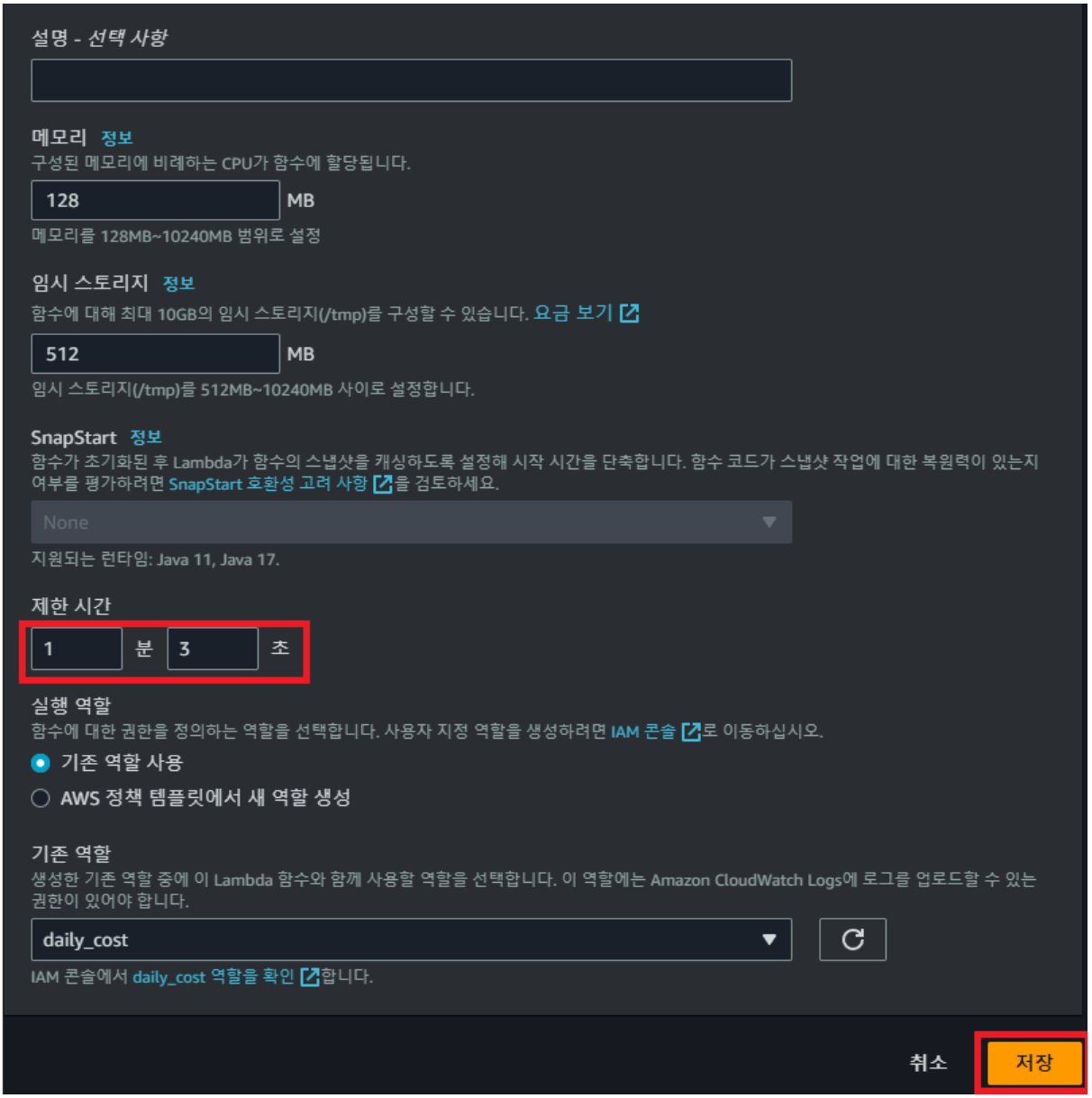

```
cron(5 21 ? * TUE-SAT *)
```

이 cron 식은 매주 화요일 부터 토요일 (화요일 부터 토요일 인 이유는 금액 정보가 어제 하루 기준이기 때문 입니다.) 까지 lambda 함수 지역시간 21 시 5분에 실행되게 합니다.

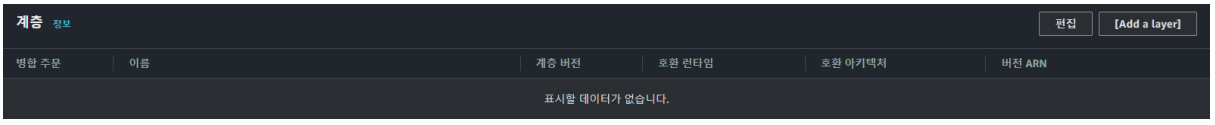
(lambda 함수를 만들 때 만든 지역 기준의 시간 입니다. 저는 버지니아 북부(us-east-1) 에 생성해서 지역시간 21시면 한국 시간 6시 입니다.)


함수 개요 정보
Test-Daily-Cost
Layers (0)
EventBridge(CloudWatch Events)
+ 트리거 추가
+ 대상 추가
설명
-
마지막 수정
38초 전
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:Test-Daily-Cost
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
일반 구성
트리거
권한
대상
함수 URL
일반 구성 정보
편집
설명
-
메모리
128 MB
임시 스토리지
512 MB
제한 시간
0 분 3 초
SnapStart 정보
None

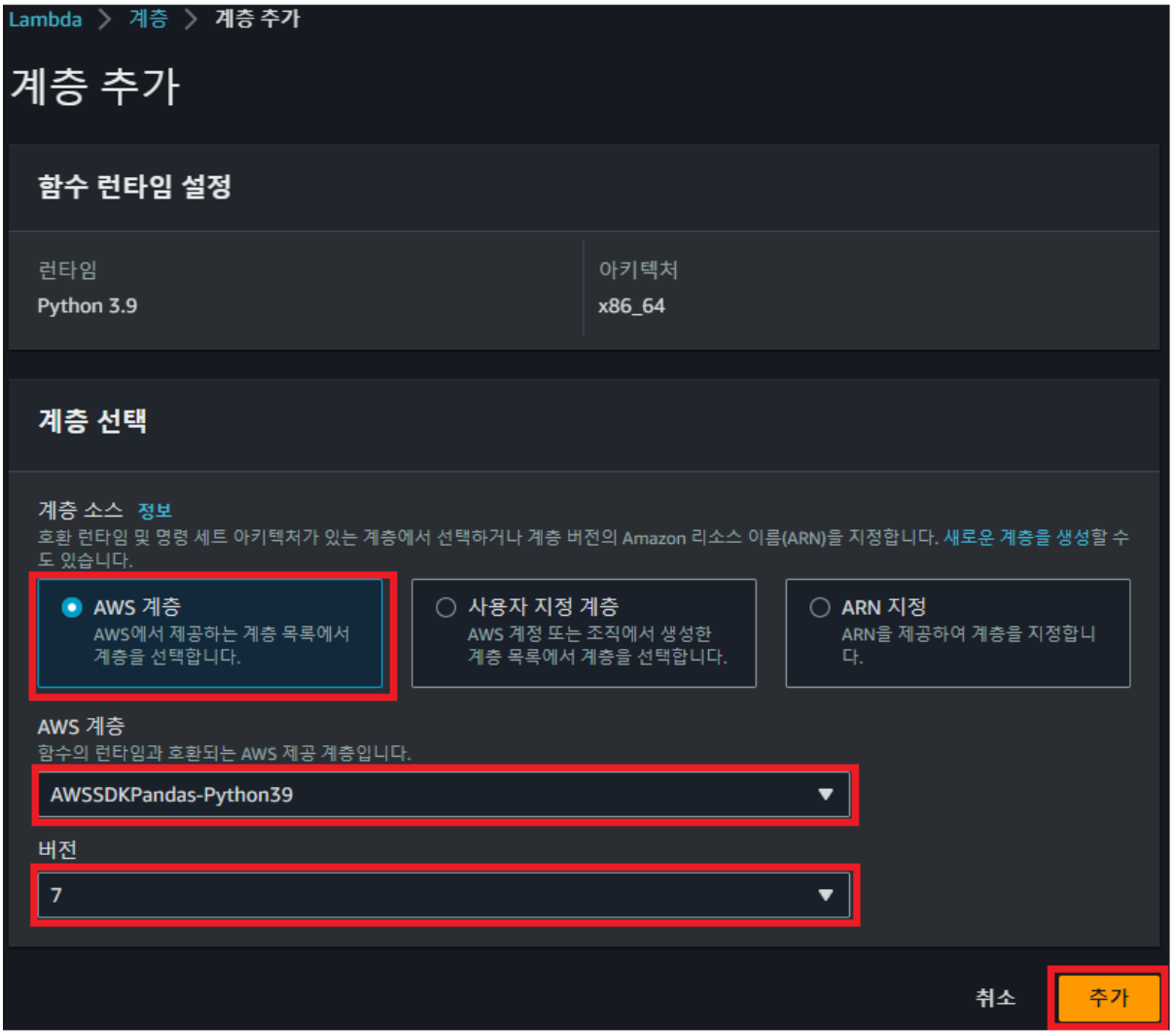
구성 → 일반 구성 → 편집



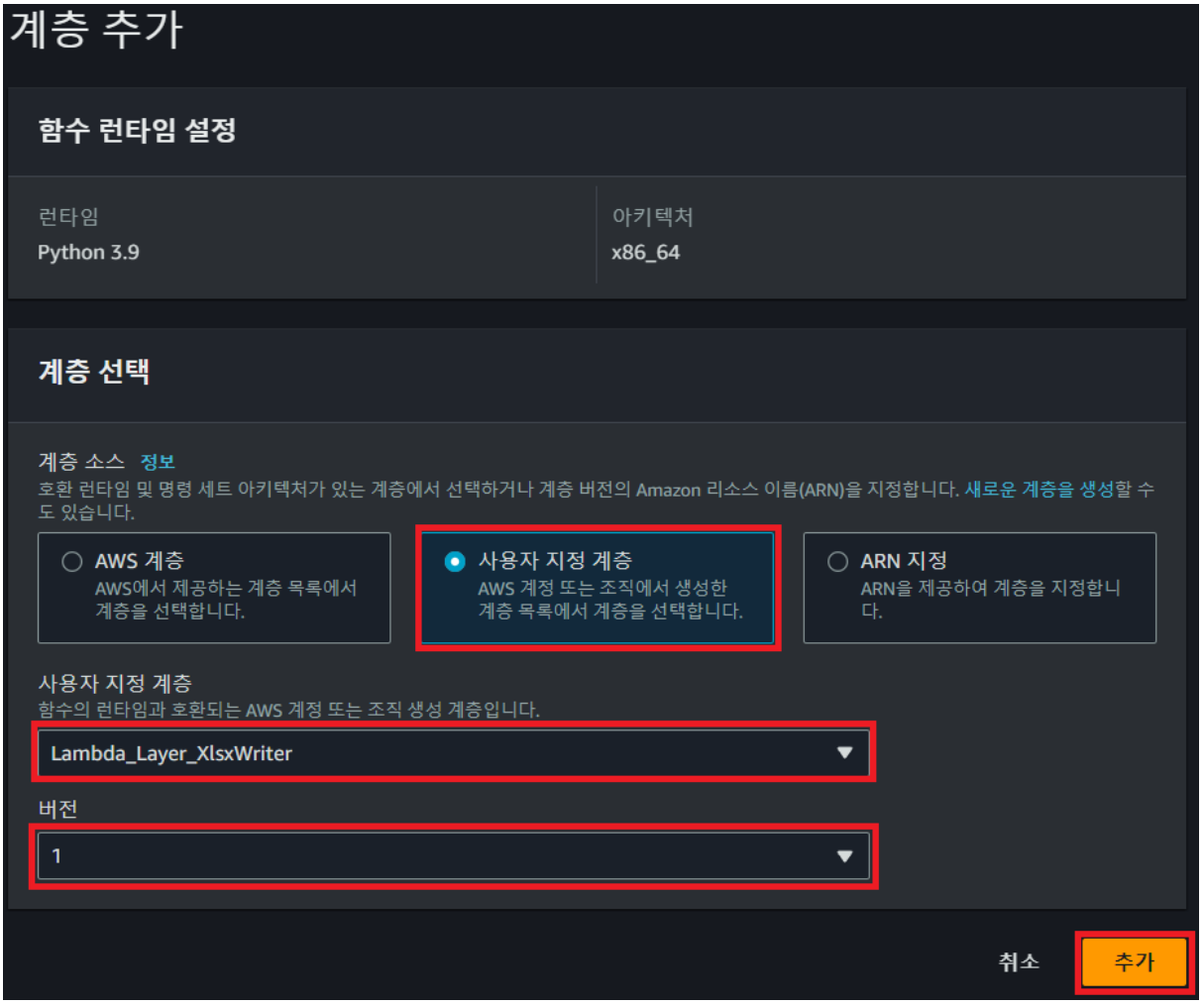
제한 시간 1 분 이상으로 늘리고 저장



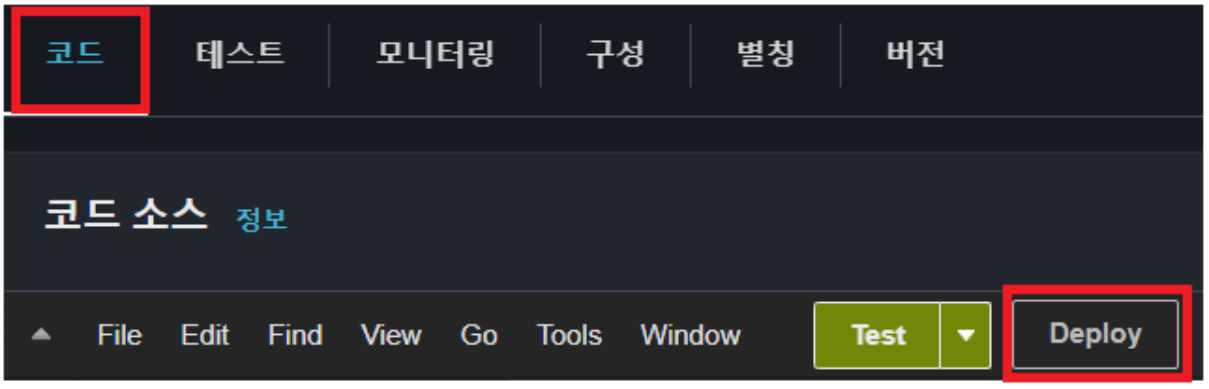
이제 계층을 추가 합니다. 이 함수에서 추가해야 하는 계층은 총 2개입니다.



aws 계층 → AWSSDKPandas-Python39 → 버전 → 추가



사용자 지정 계층 → 전에 만든 계층 → 버전 선택 → 추가



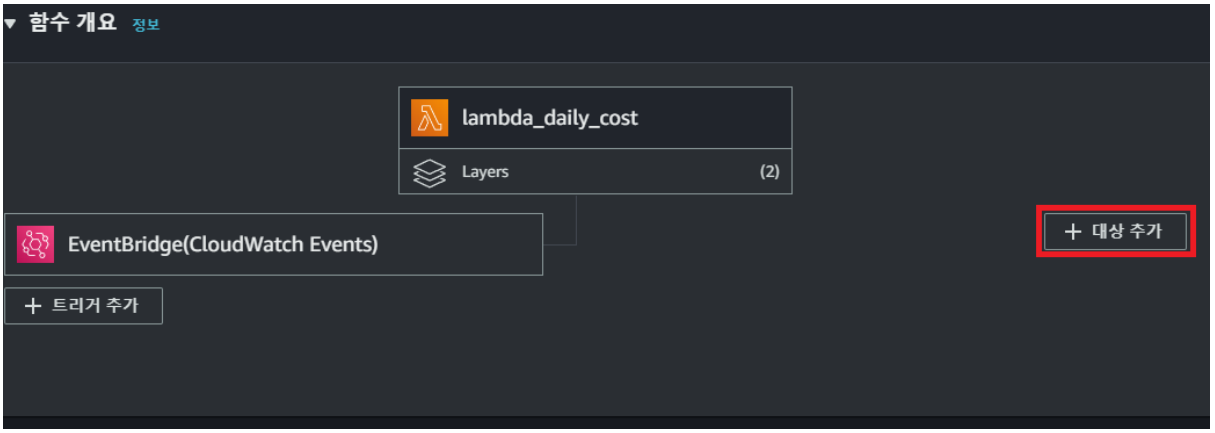
코드 → 코드 소스에 아래 코드 복사 붙여 넣기 → Deploy

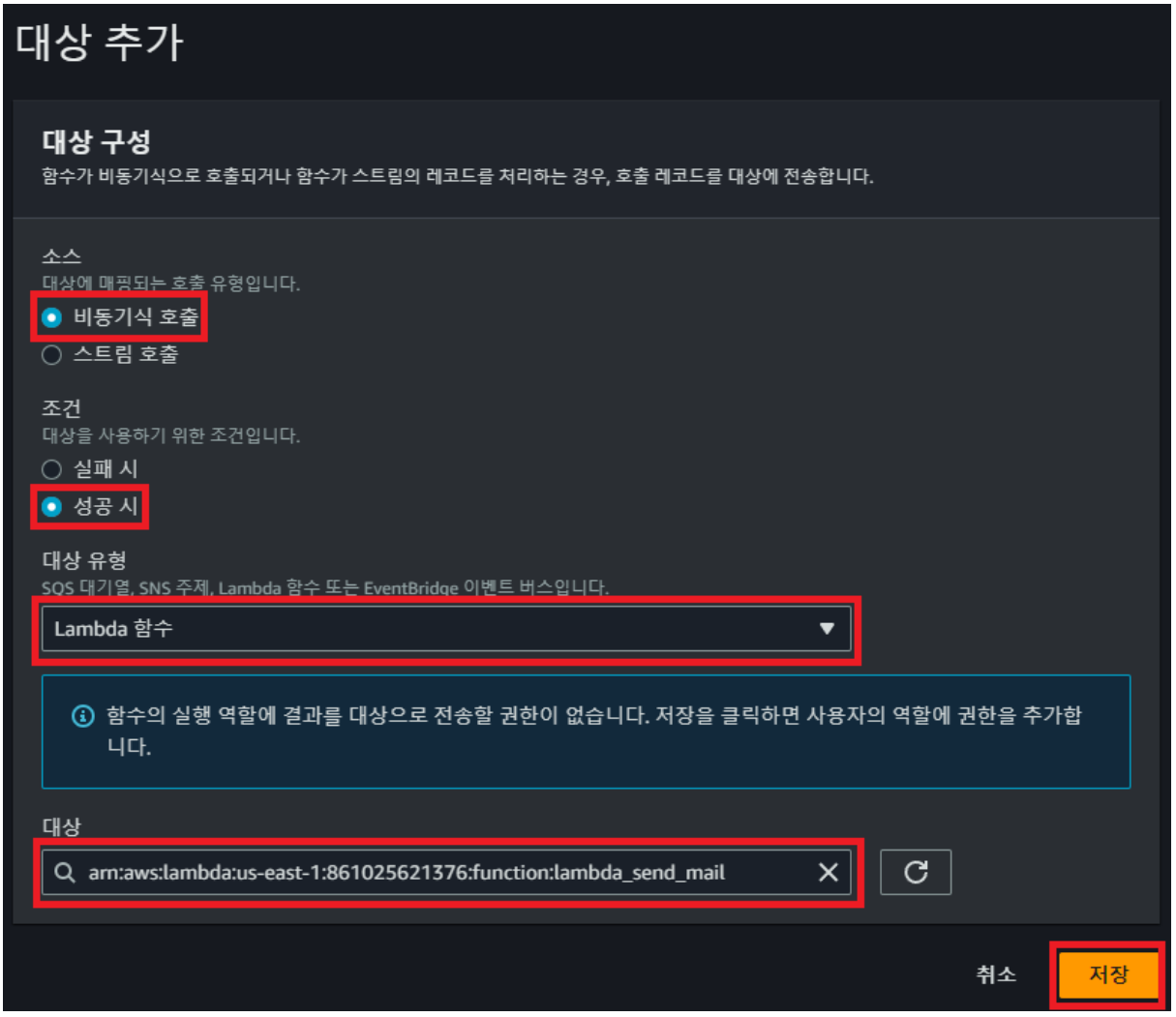
```
import boto3
import pandas as pd
```

```
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)['
    excel_files = [obj['Key'] for obj in bucket_objects if ob

    # 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', '

    # 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
        for column in worksheet.columns:
            max_length = 0
            column = column[0].column_letter  # 열 이름 가져오기
            for cell in worksheet[column]:
                try:
                    if len(str(cell.value)) > max_length:
                        max_length = len(cell.value)
                except TypeError:
```

```
                    pass
            adjusted_width = (max_length + 2) * 1.2
            worksheet.column_dimensions[column].width = adjus

        # 헤더 스타일 지정
        header_font = openpyxl.styles.Font(bold=True)
        for cell in worksheet[1]:
            cell.font = header_font

    output.seek(0)

    # Excel 파일을 S3에 업로드
    s3.upload_fileobj(output, bucket_name, output_filename)


def lambda_handler(event, context):
    bucket_name = 'daily-cost-s3'
    output_filename = 'merged_cost.xlsx'
    merge_excel_files(bucket_name, output_filename)
```

이제 마지막으로 대상을 추가 해줍니다.



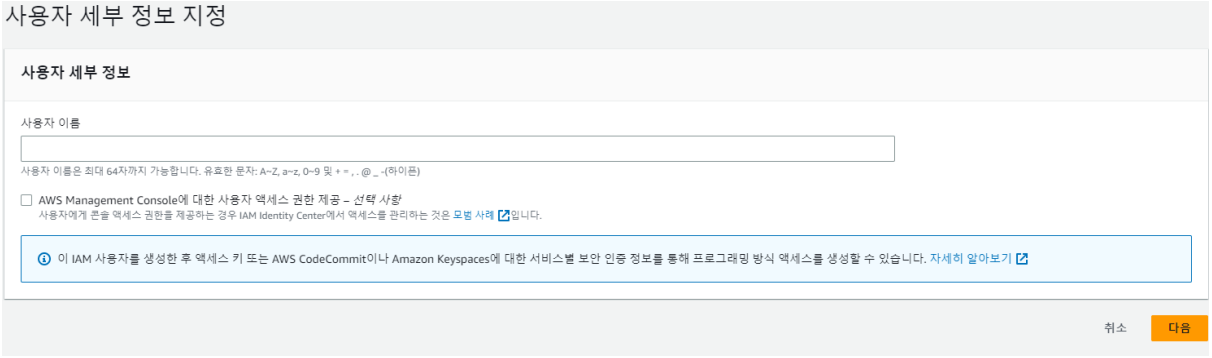


비동기식 호출 → 성공 시 → Lambda 함수 → 대상 추가(전에 만든 send-mail 함수 입니다) → 저장

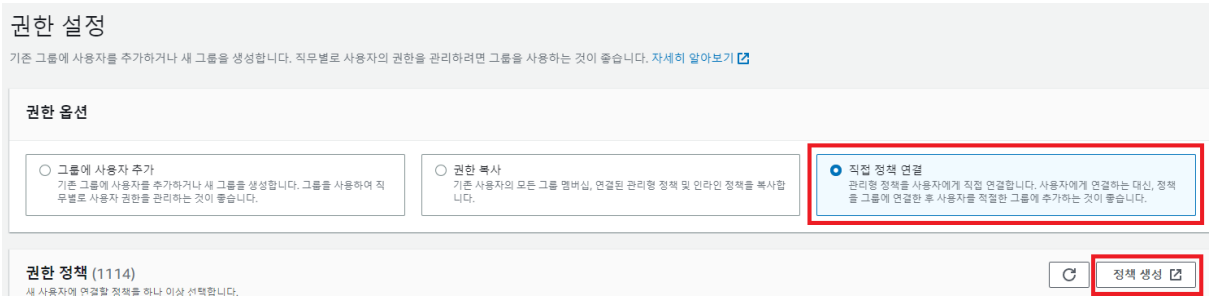
마지막으로 다른 계정에서 접속할 IAM 계정을 생성합니다.

2.5 IAM 계정 생성

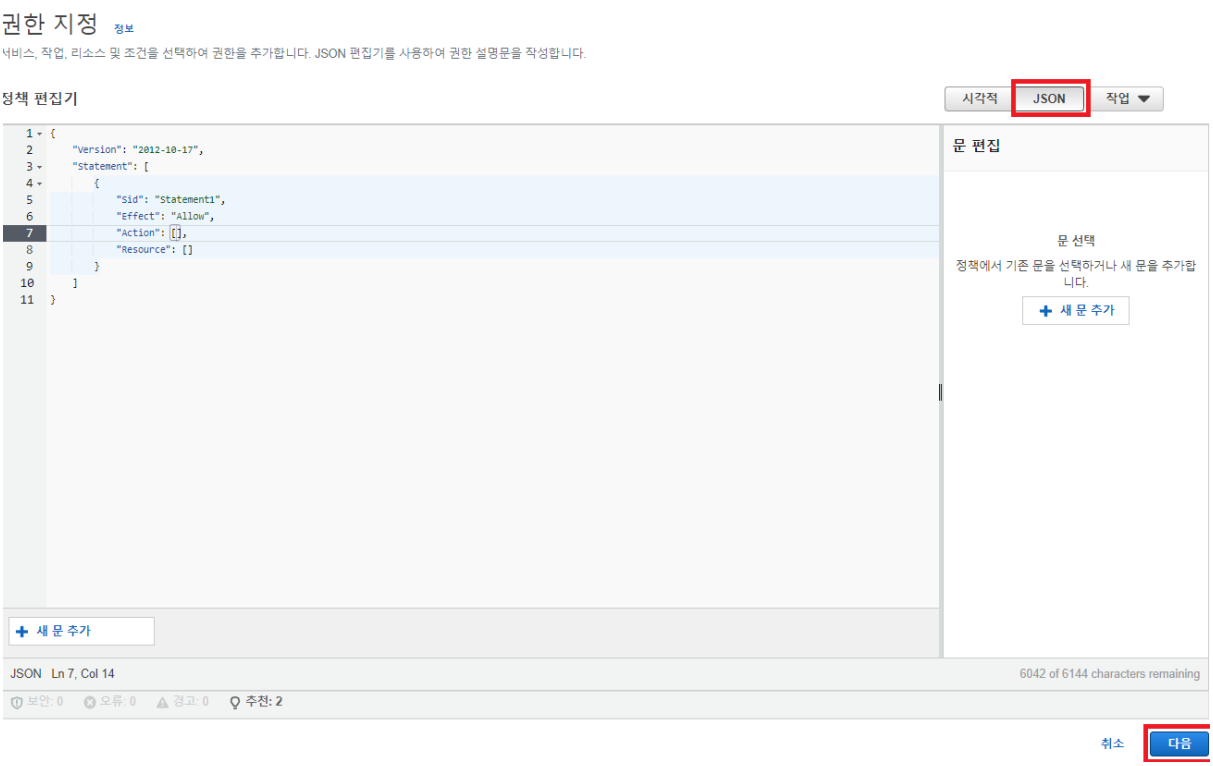
IAM → 사용자 → 사용자 추가



사용자 이름(아무거나) → 다음



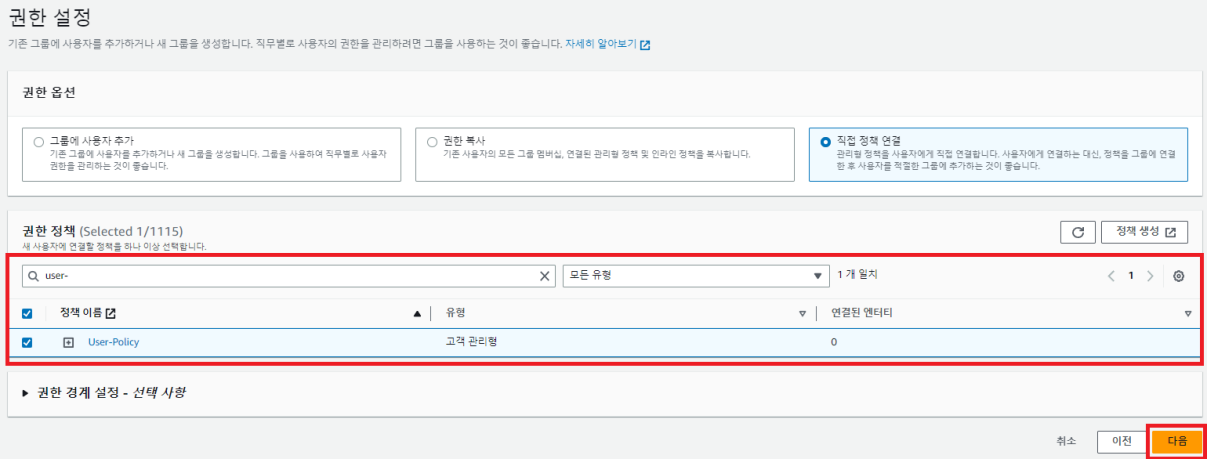
직접 정책 연결 → 정책 생성



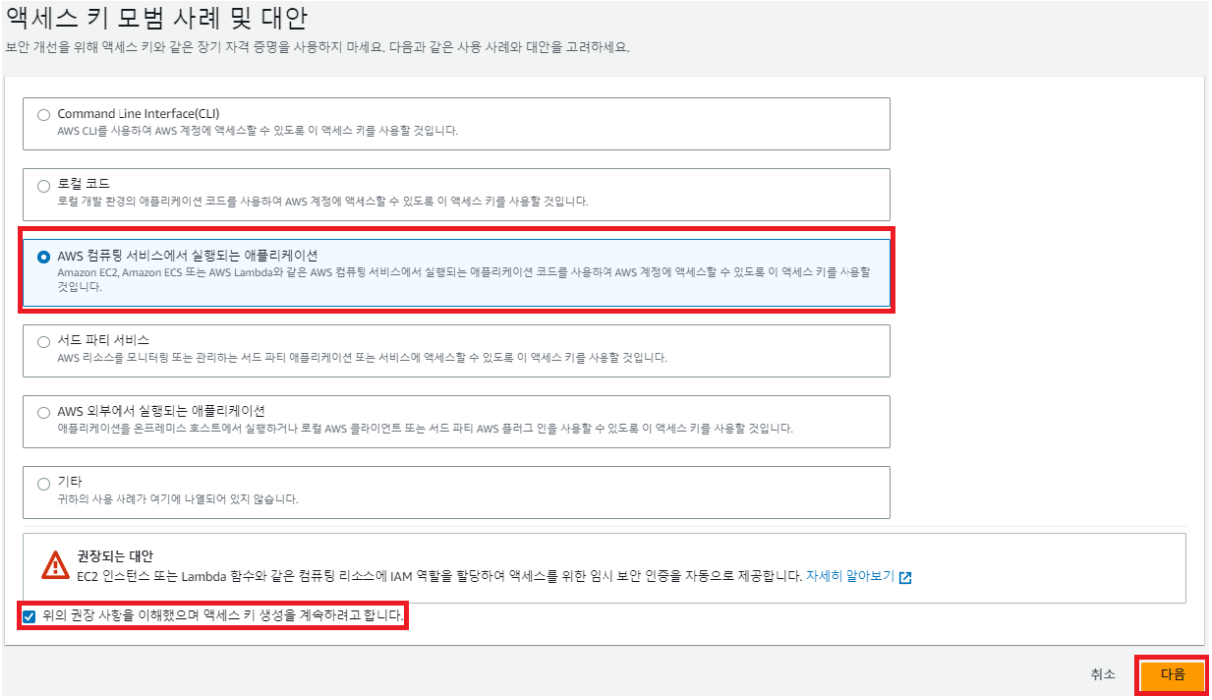
JSON 클릭 후 아래 텍스트 복사 붙여 넣기(s3 버킷 이름 다르면 바꿔줘야 합니다) → 다음

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCostExplorerAPIAccess",
            "Effect": "Allow",
            "Action": [
                "ce:GetCostAndUsage"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::daily-cost-s3",
                "arn:aws:s3:::daily-cost-s3/*"
            ]
        }
    ]
}
```

이름 설정 후 → 다음 → 다시 권한 설정 창으로 돌아와서 전에 만든 정책 선택 → 다음 → 사용자 생성

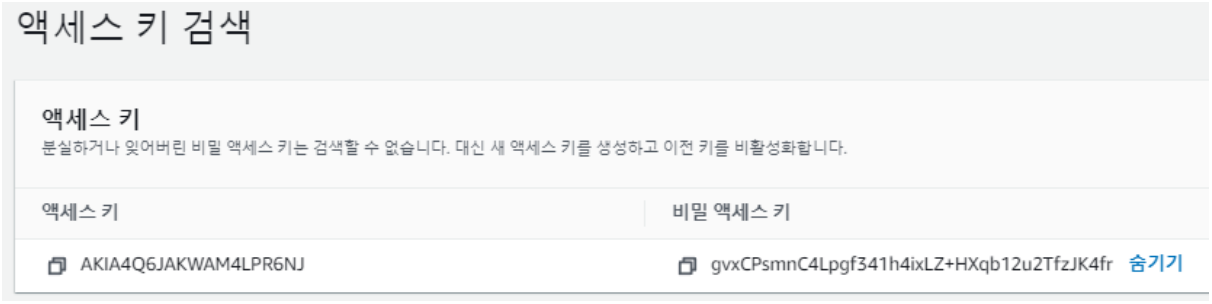


이제 만든 사용자 클릭 → 보안 자격 증명 → 액세스 키 만들기



AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 → 체크 박스 체크 → 다음 → 액세스 키 만들기

완료되면

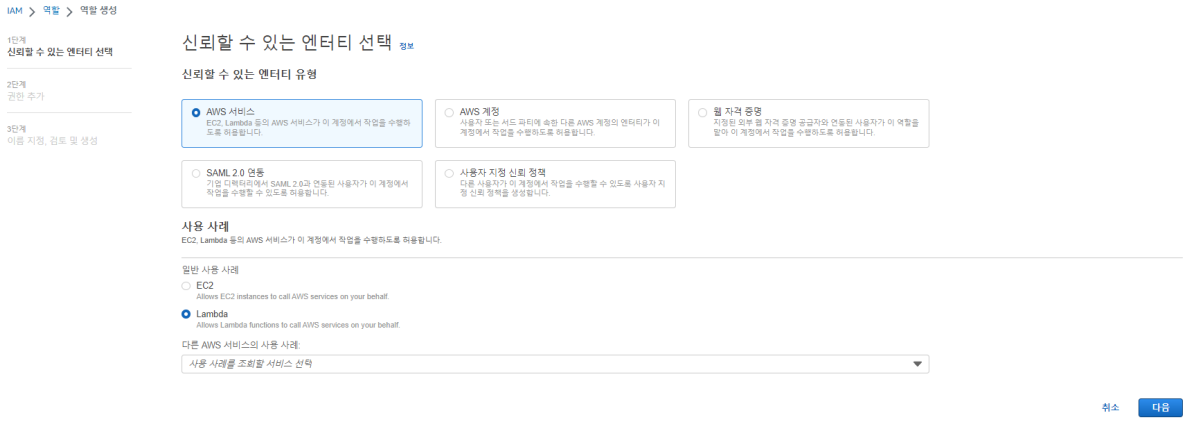


액세스 키와 비밀 액세스 키를 다른 곳에 저장 해주세요

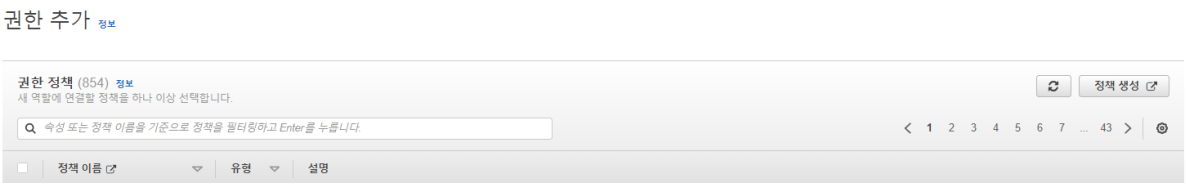
3. Another-Local 세팅

이제 다른 계정에서 Lambda 함수를 만들면 끝입니다.

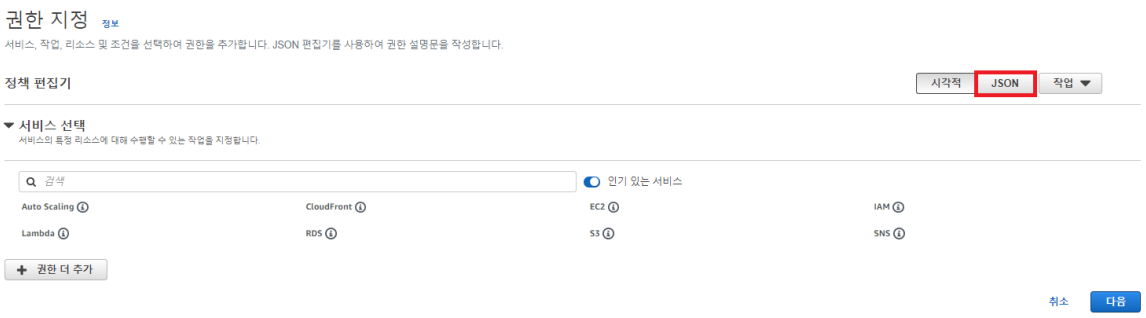
먼저 람다 함수를 실행할 IAM 역할을 생성 하겠습니다.



IAM → 역할 → 역할 생성

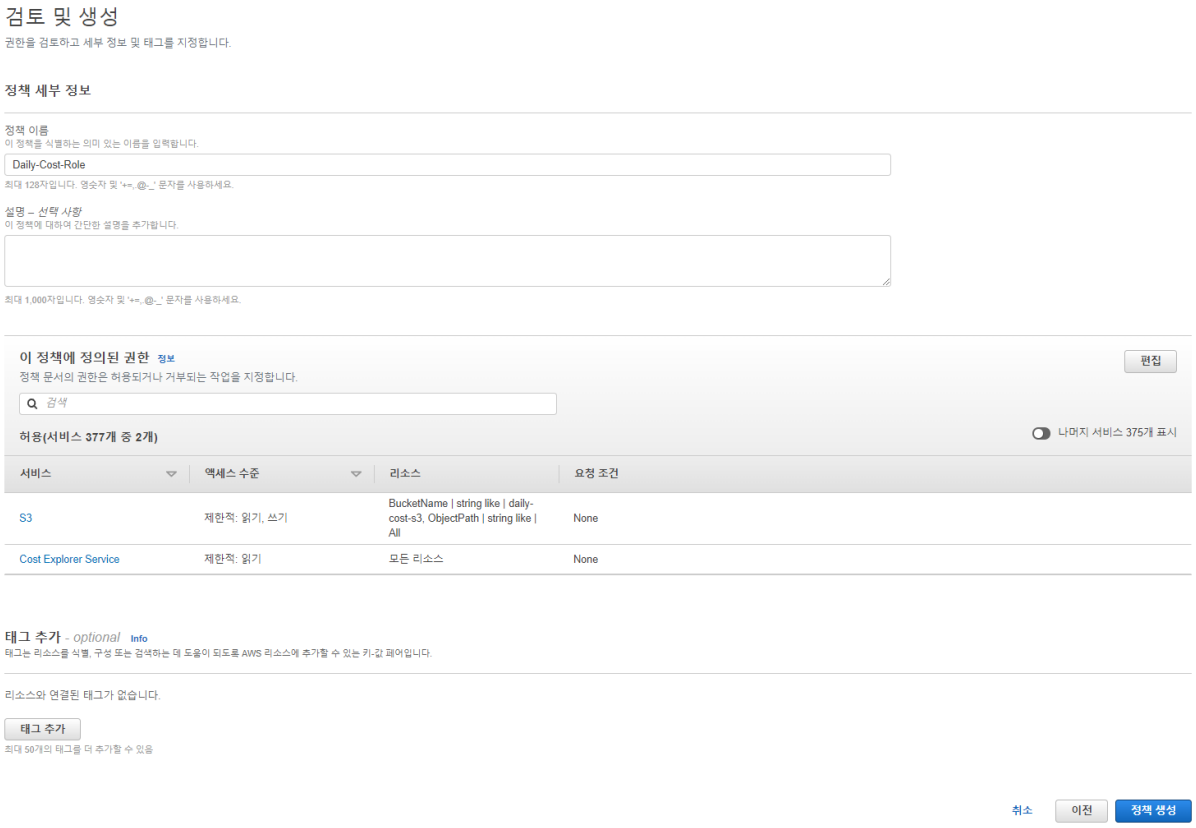


정책 생성 클릭

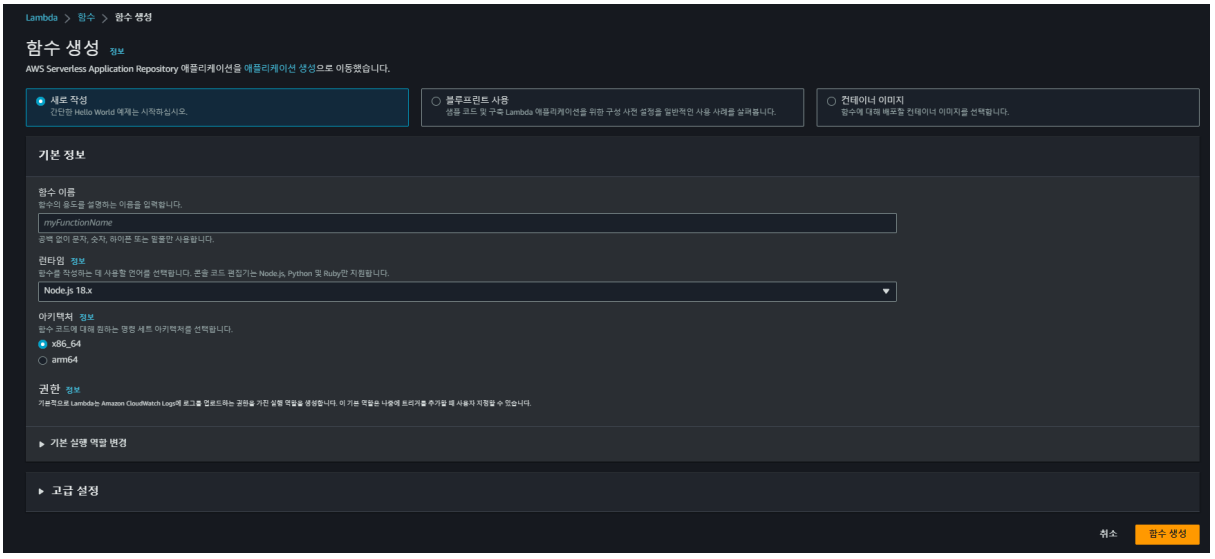


JSON 클릭 후 밑의 텍스트를 복사

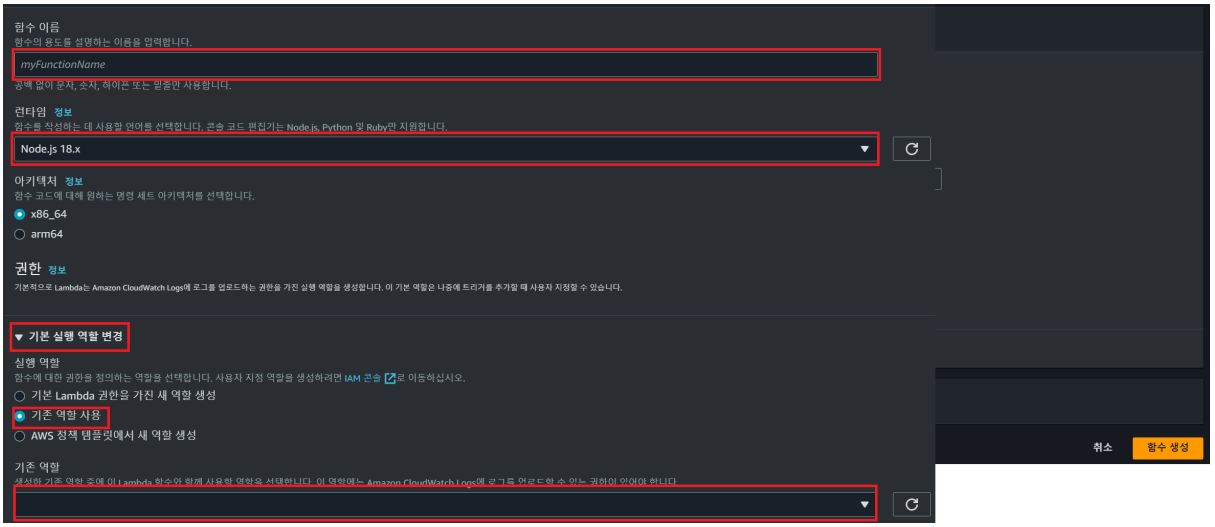
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCostExplorerAPIAccess",
      "Effect": "Allow",
      "Action": [
        "ce:GetCostAndUsage"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowS3ObjectPutAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::daily-cost-s3/*"
    }
  ]
}
```



이름을 지정해줍니다.(저는 알아보기 쉽게 Daily-Cost-Role 로 하였습니다.) 정책 생성을 눌러주면 끝입니다.



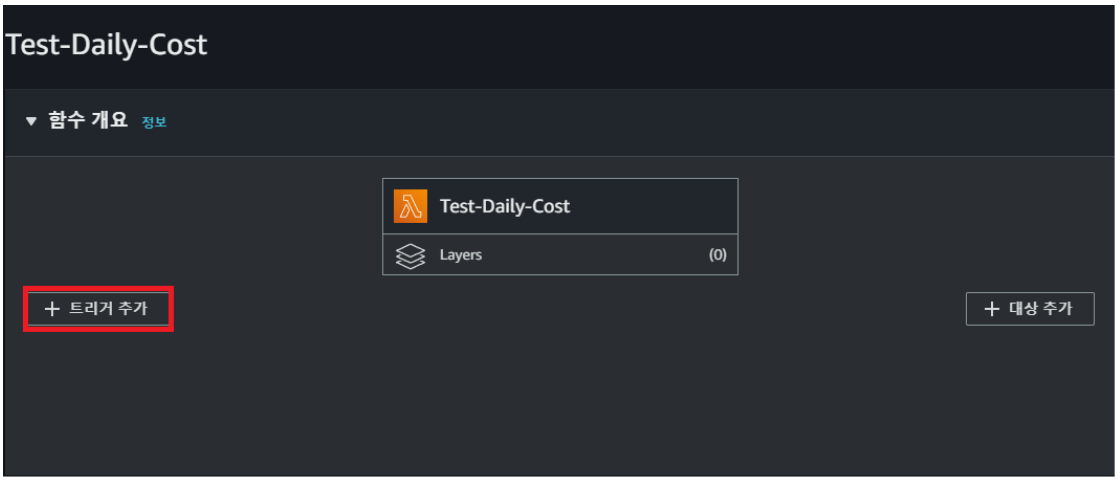
Lambda → 함수 → 함수 생성을 누릅니다.



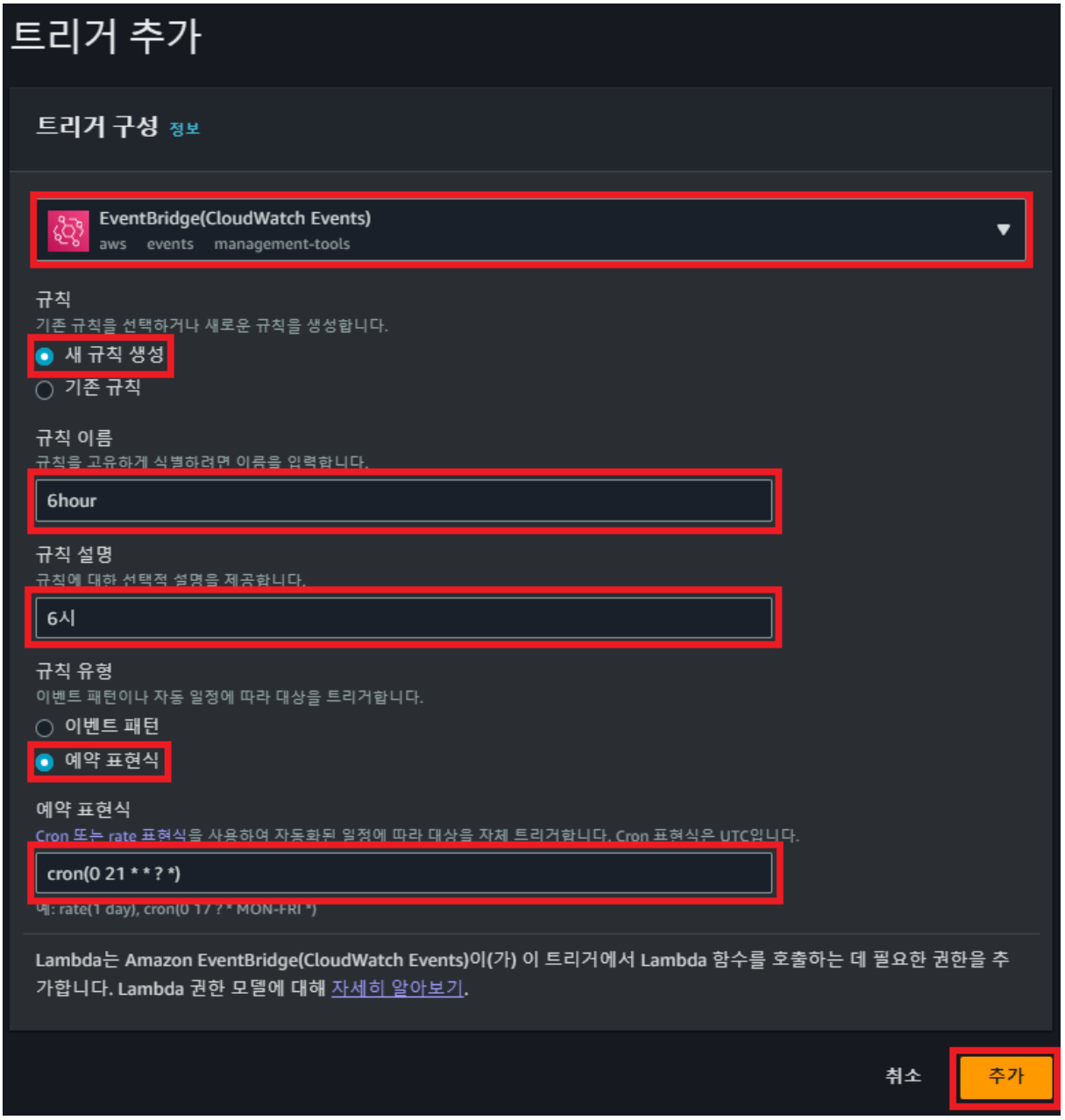
함수 이름을 설정해줍니다. (편하게 구분할 수 있게 **lambda_daily_cost** 로하겠습니다.)

런타임은 Python 3.9 로 지정해줍니다.

기본 실행 역할 변경 → 기존 역할 사용 → 전에 만든 역할 선택 → 함수 생성



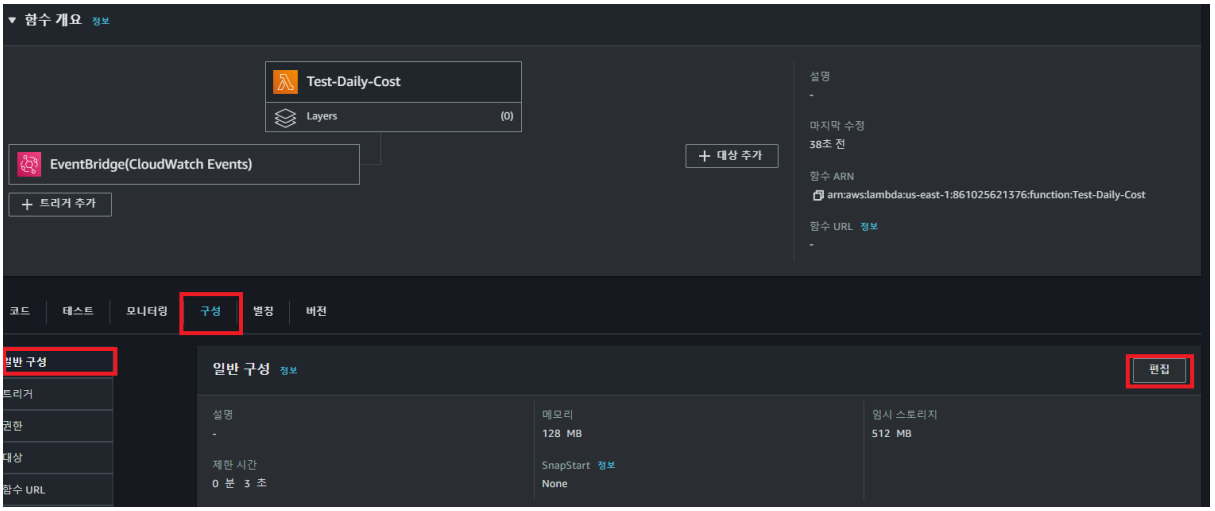
함수 → 트리거 추가 → EventBridge



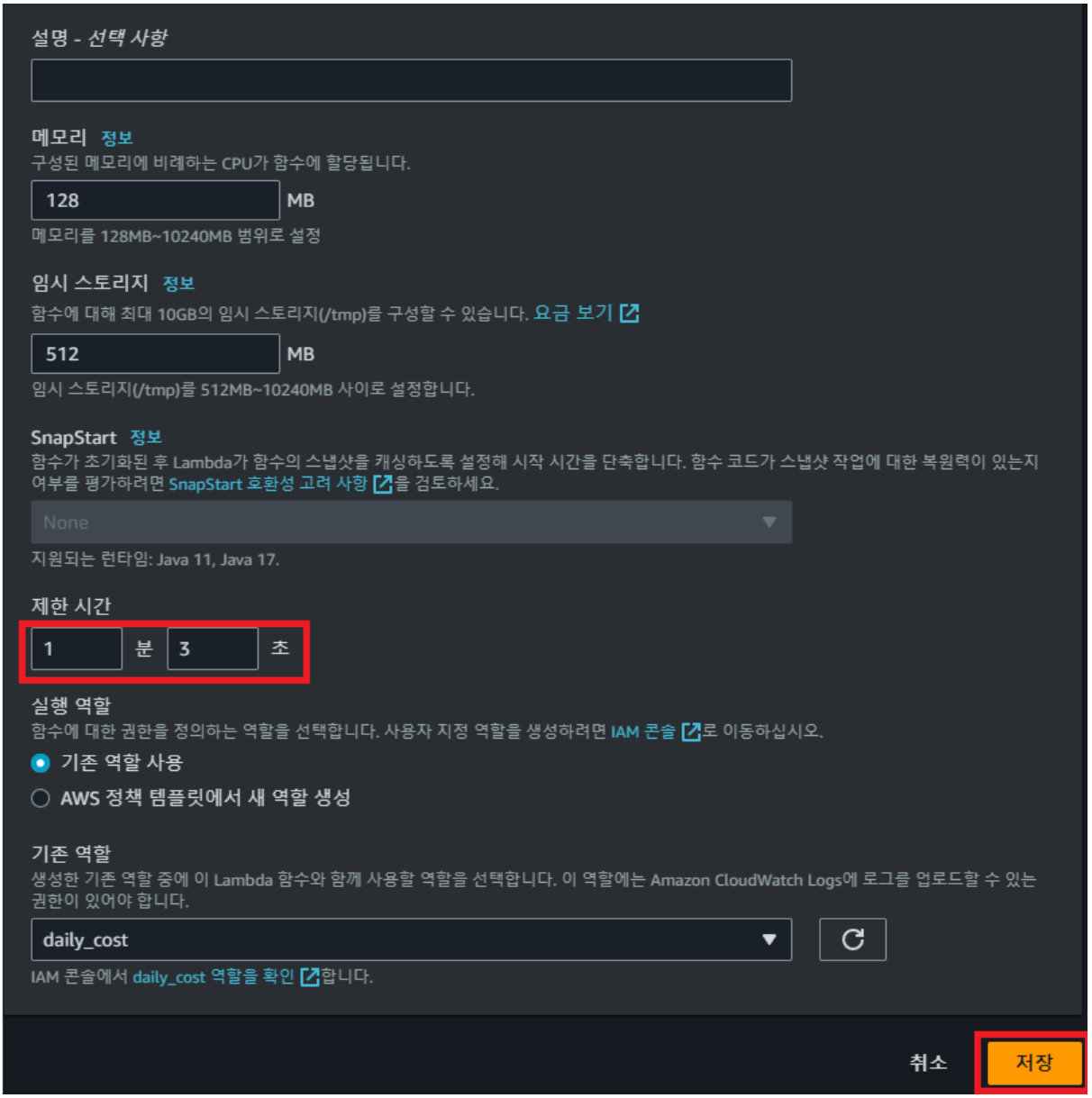

```
cron(0 21 ? * TUE-SAT *)
```

이 cron 식은 매주 화요일 부터 토요일 (화요일 부터 토요일 인 이유는 금액 정보가 어제 하루 기준이기 때문 입니다.) 까지 lambda 함수 지역시간 21 시에 실행되게 합니다.

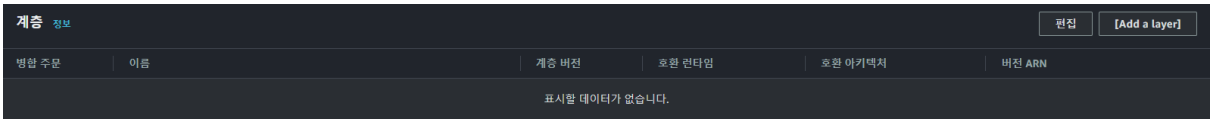
(lambda 함수를 만들 때 만든 지역 기준의 시간 입니다. 저는 버지니아 북부(us-east-1) 에 생성해서 지역시간 21시면 한국 시간 6시 입니다.)


함수 개요 정보
Test-Daily-Cost
Layers (0)
EventBridge(CloudWatch Events)
+ 트리거 추가
+ 대상 추가
설명
-
마지막 수정
38초 전
함수 ARN
arn:aws:lambda:us-east-1:861025621376:function:Test-Daily-Cost
함수 URL 정보
-
코드
테스트
모니터링
구성
별칭
버전
일반 구성
트리거
권한
대상
함수 URL
일반 구성 정보
편집
설명
-
메모리
128 MB
임시 스토리지
512 MB
제한 시간
0 분 3 초
SnapStart 정보
None

구성 → 일반 구성 → 편집

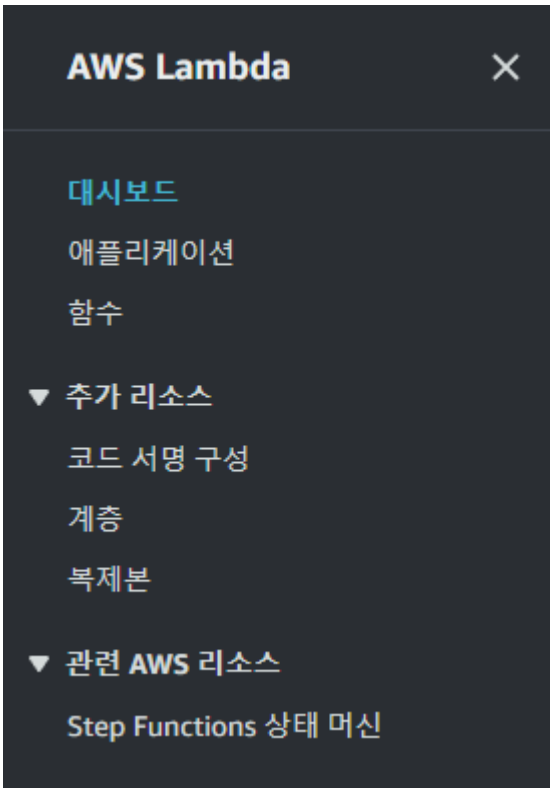


제한 시간 1 분 이상으로 늘리고 저장

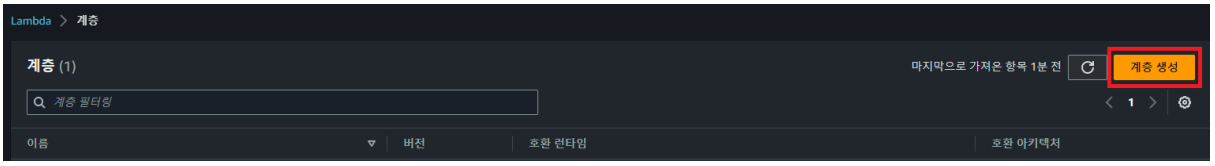


이제 계층을 추가 합니다. 이 함수에서 추가해야 하는 계층은 총 3개입니다.

먼저 사용자 지정 계층을 추가 하겠습니다.



Lambda 에서 계층 클릭

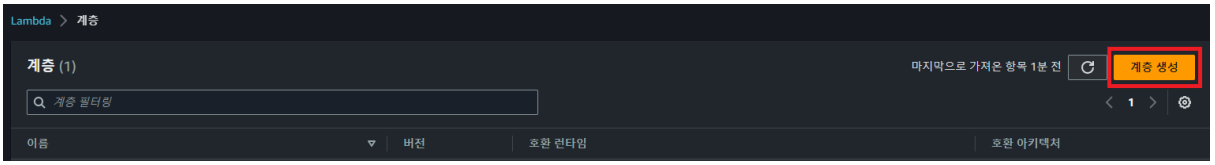


계층 생성

이름 지정 .zip 파일 업로드 → 업로드 파일(아래 파일 업로드) → 아키텍처 → 런타임 → 생성

python.zip

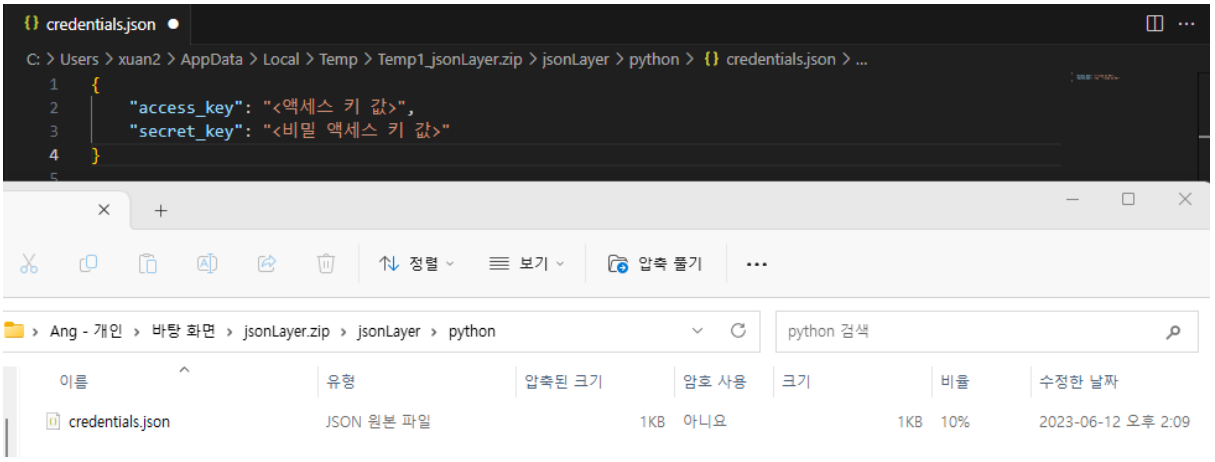
이젠 키 값을 저장 할 Json 형식 파일을 계층에 추가 할 겁니다.



계층 생성

이름 지정 .zip 파일 업로드 → 업로드 파일(아래 파일 업로드) → 아키텍처 → 런타임 → 생성

아래 파일을 다운로드 후 열어서 JSON 파일을 열어서

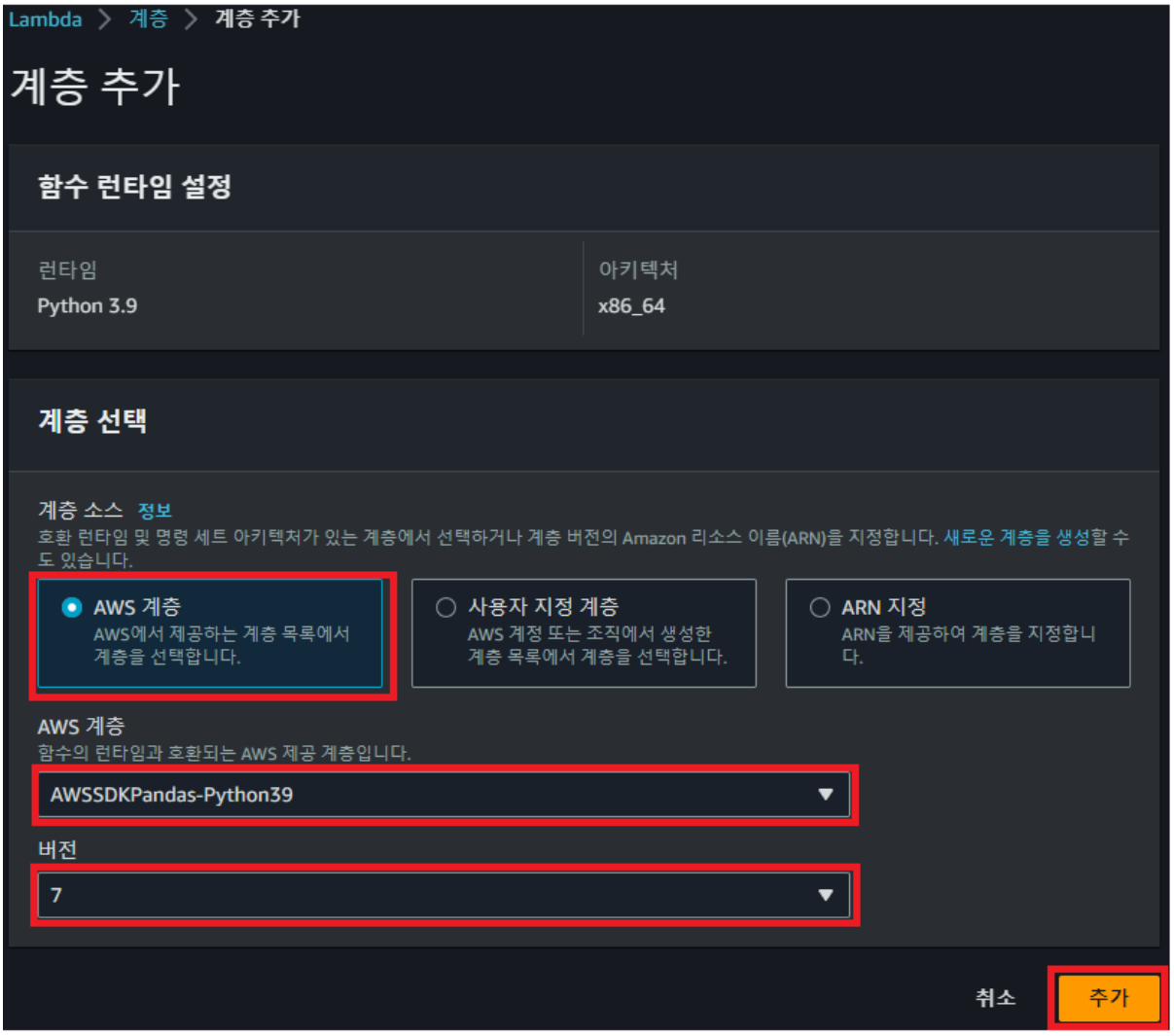


위에서 생성한 액세스 키들을 입력 해줍니다.

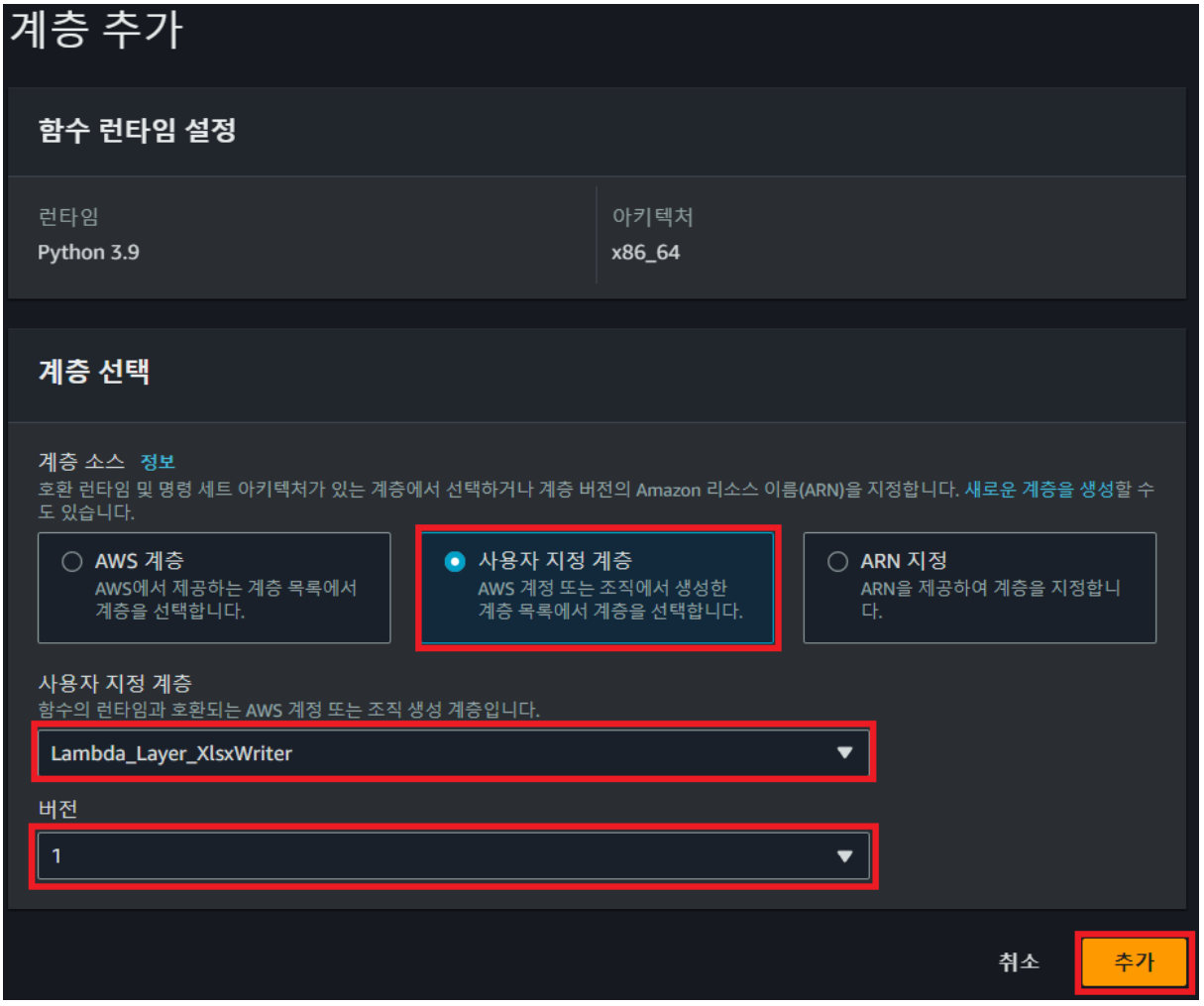
jsonLayer.zip

다시 람다 함수로 돌아옵니다.

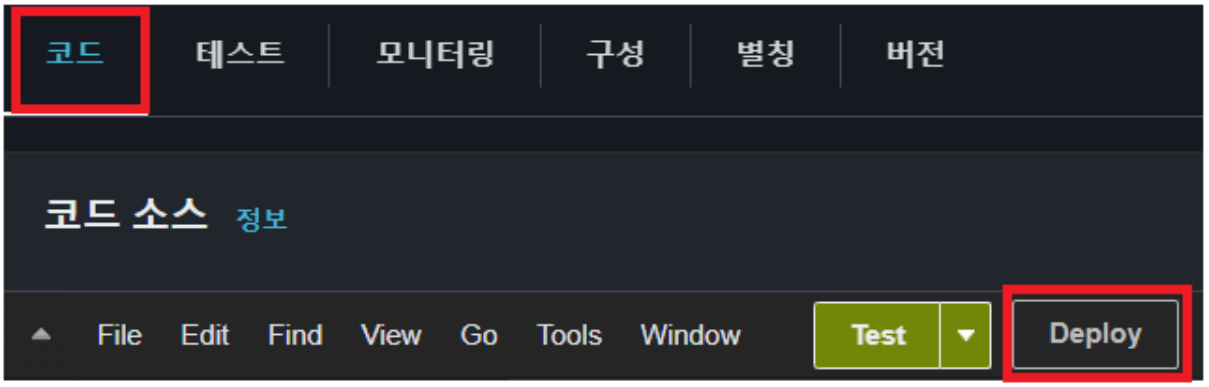
밑으로 쭉 스크롤 내리고 계층에서 Add a layer



aws 계층 → AWSSDKPandas-Python39 → 버전 → 추가



사용자 지정 계층 → 전에 만든 계층 → 버전 선택 → 추가



코드 → 코드 소스에 아래 코드 복사 붙여 넣기 → Deploy

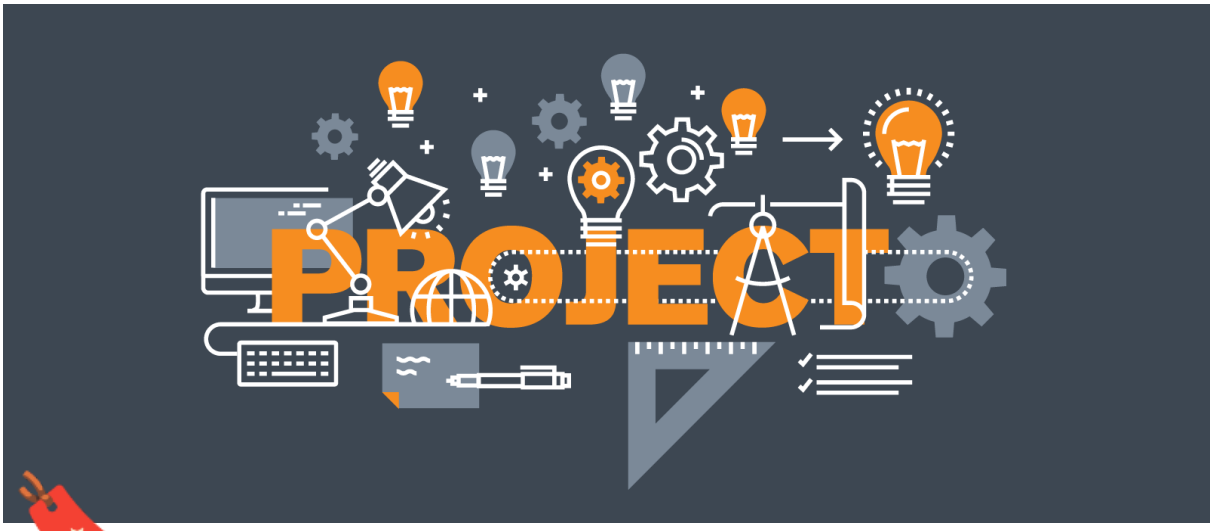
```
import boto3
import datetime
import io
```

```
import json
import pandas as pd

def lambda_handler(event, context):
    # 계층에 추가된 JSON 파일의 경로
    credentials_file = '/opt/jsonLayer/python/credentials.jso

    # JSON 파일에서 액세스 키와 시크릿 키 읽어오기
    with open(credentials_file, 'r') as file:
        credentials = json.load(file)
    access_key = credentials['access_key']
    secret_key = credentials['secret_key']

    # AWS 청구서 클라이언트 생성
    client = boto3.client('ce')

    # 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 ㄷ
            },
            {
                'Type': 'DIMENSION',
                'Key': 'LINKED_ACCOUNT'  # 링크된 어카운트별 비용을
            }
        ]
```

```
    }

    # AWS 청구서에 요청하여 결과 가져오기
    response = client.get_cost_and_usage(**params)

   # 결과를 DataFrame으로 변환하기
    data = []
    total_cost = 0.0  # 총합 비용 초기화
    for result_by_time in response['ResultsByTime']:
        date = result_by_time['TimePeriod']['Start']
        for group in result_by_time['Groups']:
            linked_account = group['Keys'][1]  # 링크된 어카운트
            service = group['Keys'][0]
            cost = group['Metrics']['UnblendedCost']['Amount'
            total_cost += float(cost)  # 비용을 누적하여 총합 비용
            data.append([linked_account, date, service, cost]
    df = pd.DataFrame(data, columns=['Linked Account', 'Date'

    # 총합 비용 행 추가
    total_row = [linked_account, date, 'Total', str(total_cos
    total_df = pd.DataFrame([total_row], columns=['Linked Acc
    df = pd.concat([df, total_df], ignore_index=True)


    # DataFrame을 Excel 파일로 저장하기
    output = io.BytesIO()
    with pd.ExcelWriter(output, engine='xlsxwriter') as write
        df.to_excel(writer, index=False, sheet_name='Sheet1')
    output.seek(0)

    # Excel 파일을 S3 버킷에 업로드하기
    s3 = boto3.resource('s3', aws_access_key_id=access_key, au
    bucket_name = 'daily-cost-s3'
    filename = 'aws_cost_{}_user1.xlsx'.format(datetime.datet
    s3.Object(bucket_name, filename).put(Body=output.getvalue
```

!중요 filename은 모든 계정마다 서로 다르게 해줘야 합니다.

Deploy가 끝나면 끝입니다.



AWS_Daily_Cost_2

STACK AWS Shell

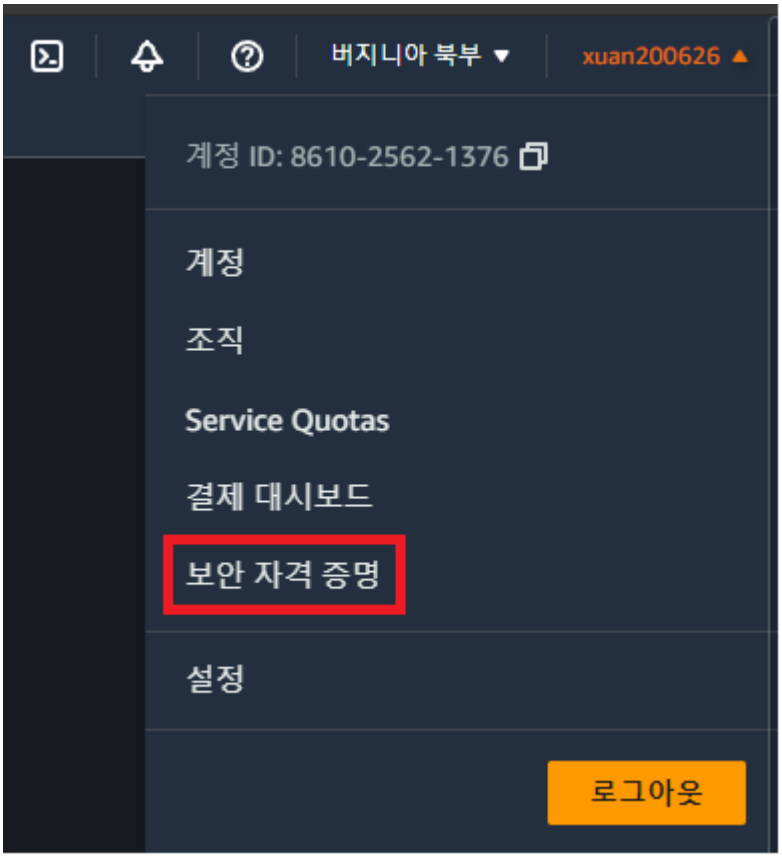
만든 목적:

1. 보다 쉽게 만들기 위해
2. 비용 발생을 줄이기 위해서

1. 사전 준비

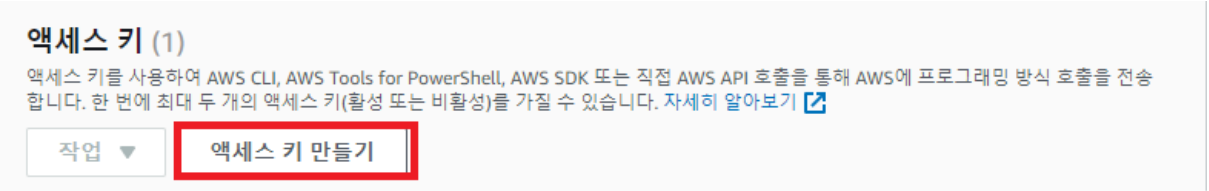
1.1 자격증명 만들기

aws configure를 통해 awscli로 비용 정보를 불러 올거기 때문에 엑세스 키가 필요



비용을 받고 싶은 계정에 로그인 후 보안 자격 증명으로 들어간다

이후 액세스 키를 하나 생성 해준다



만들어진 액세스 키와 비밀 액세스 키는 나중에 사용하므로 잘 저장해준다

1.2 aws configure

awscli를 설치해준다. (**설치 링크**)

cmd 나 powershell를 실행 후 aws configure를 입력한다.

```
C:\Users\xuan2>aws configure
AWS Access Key ID [****************QXMH]:
AWS Secret Access Key [****************rfl0]:
Default region name [None]:
Default output format [json]:
```

위와 같이 뜨면 차례대로 전에 저장 했던 액세스 키 와 비밀 액세스 키를 입력해준다. (그 이후는 다 넘겨도 된다.)

2. 스크립트 및 스케줄러 작성

2.1 스크립트 작성

메모장을 열고 아래 코드를 복사 해준다

4번째 줄의 경로 와 이름 (**출력 결과물을 저장할 파일 이름**) 을 정해준다 (**경로나 이름에 한글이 포함되면 에러가 발생할 수 있음**)

```
$StartDate = (Get-Date).AddDays(-2).ToString("yyyy-MM-dd")
$EndDate = (Get-Date).AddDays(-1).ToString("yyyy-MM-dd")
$today = Get-Date -Format "yyyy-MM-dd"
$output_file = "경로\이름_$today.json"

# Get cost and usage data from AWS Cost Explorer
$costData = aws ce get-cost-and-usage `
  --time-period Start=$StartDate,End=$EndDate `
  --granularity DAILY `
  --metrics BlendedCost UsageQuantity `
  --output json

# Output the raw JSON data to a file
$costData | Out-File -FilePath $output_file

# Process the data and categorize costs by service
$costByService = @{}
foreach ($line in $costData | ConvertFrom-Json) {
    $serviceNameDimension = $line.Dimensions | Where-Object {
```

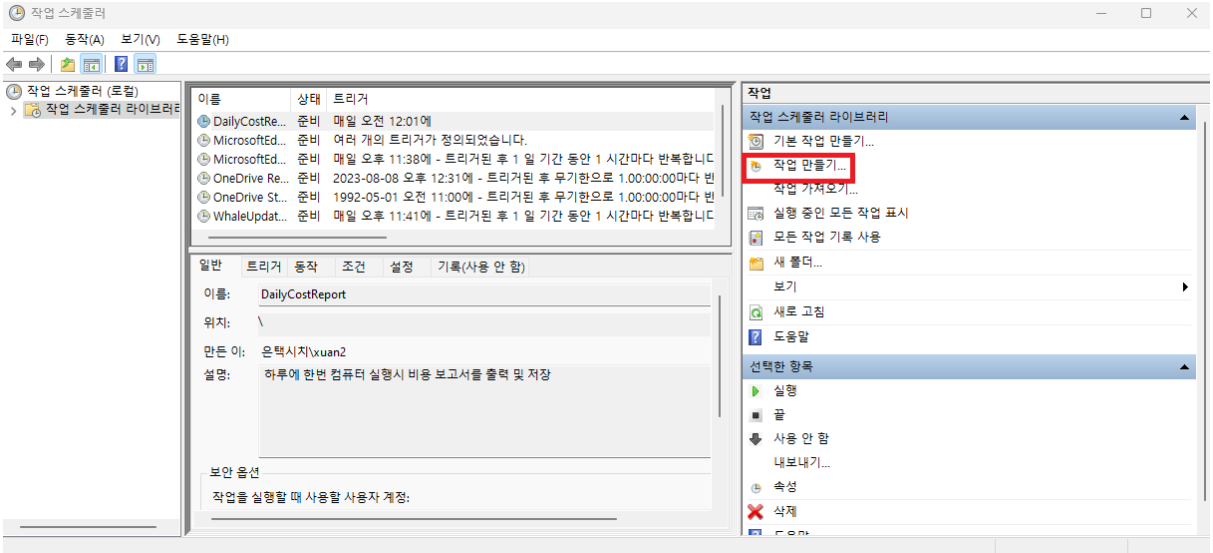
```
    if ($serviceNameDimension -ne $null) {
        $serviceName = $serviceNameDimension.Value
        $cost = $line.Total.BlendedCost.Amount
        if ($costByService.ContainsKey($serviceName)) {
            $costByService[$serviceName] += [decimal]$cost
        } else {
            $costByService[$serviceName] = [decimal]$cost
        }
    }
}
```

이후 파일 확장자명을 ps1로 바꾸어서 저장해준다. (밑에 **예시 파일**)

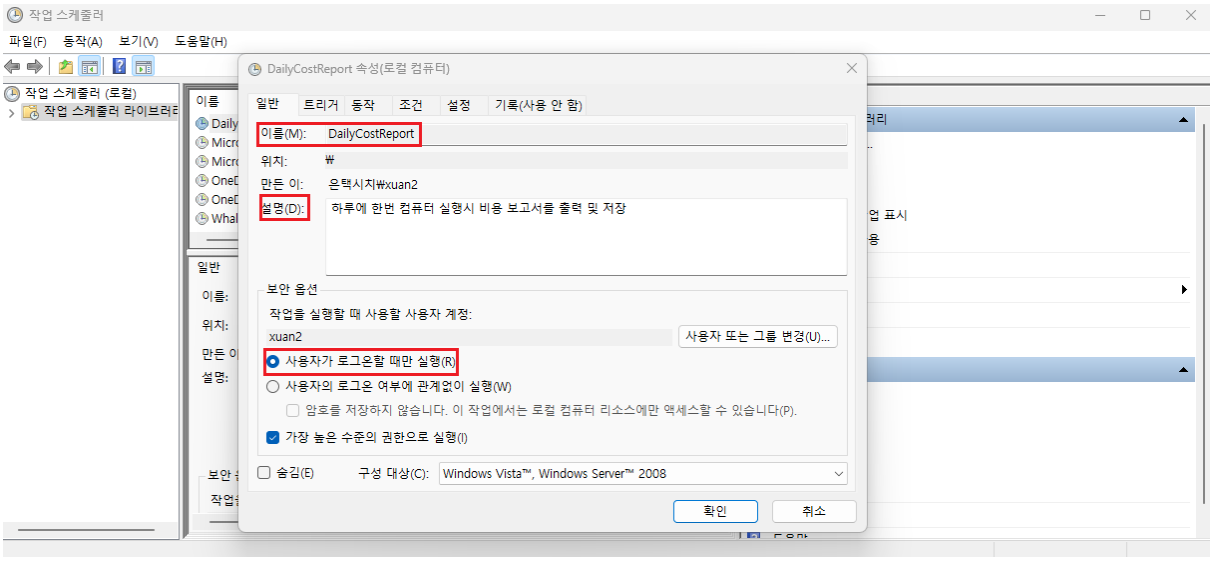
test.ps1

2.2 작업 스케줄러 설정

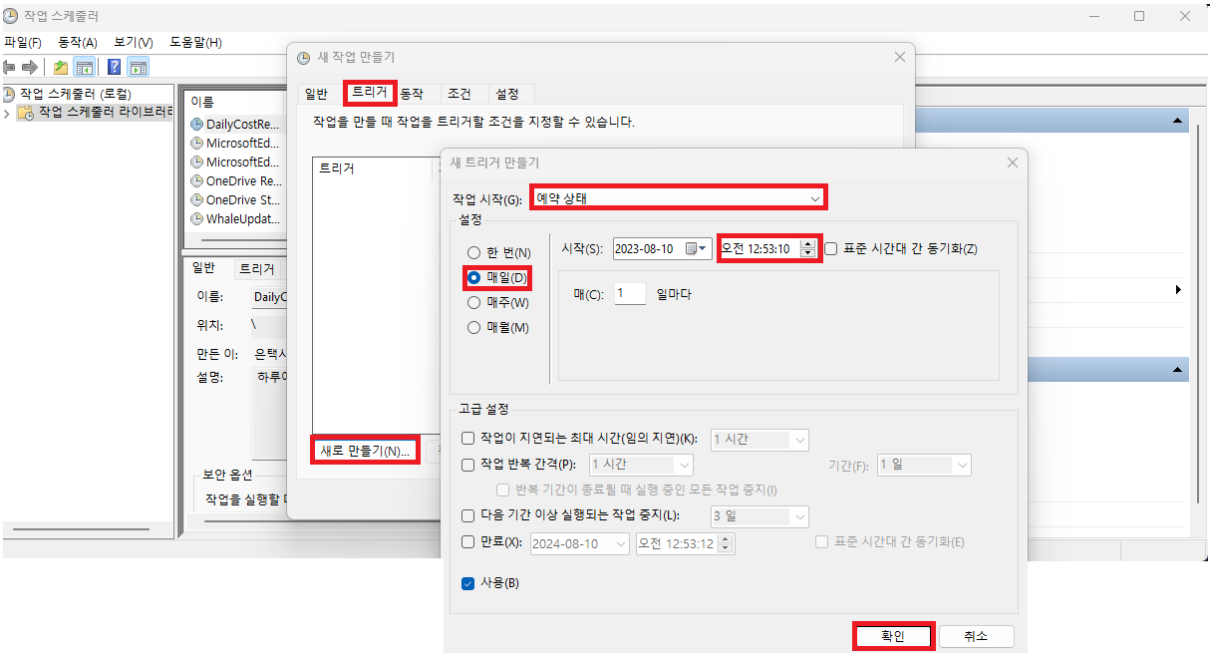
윈도우에서 작업 스케줄러를 열어준다.



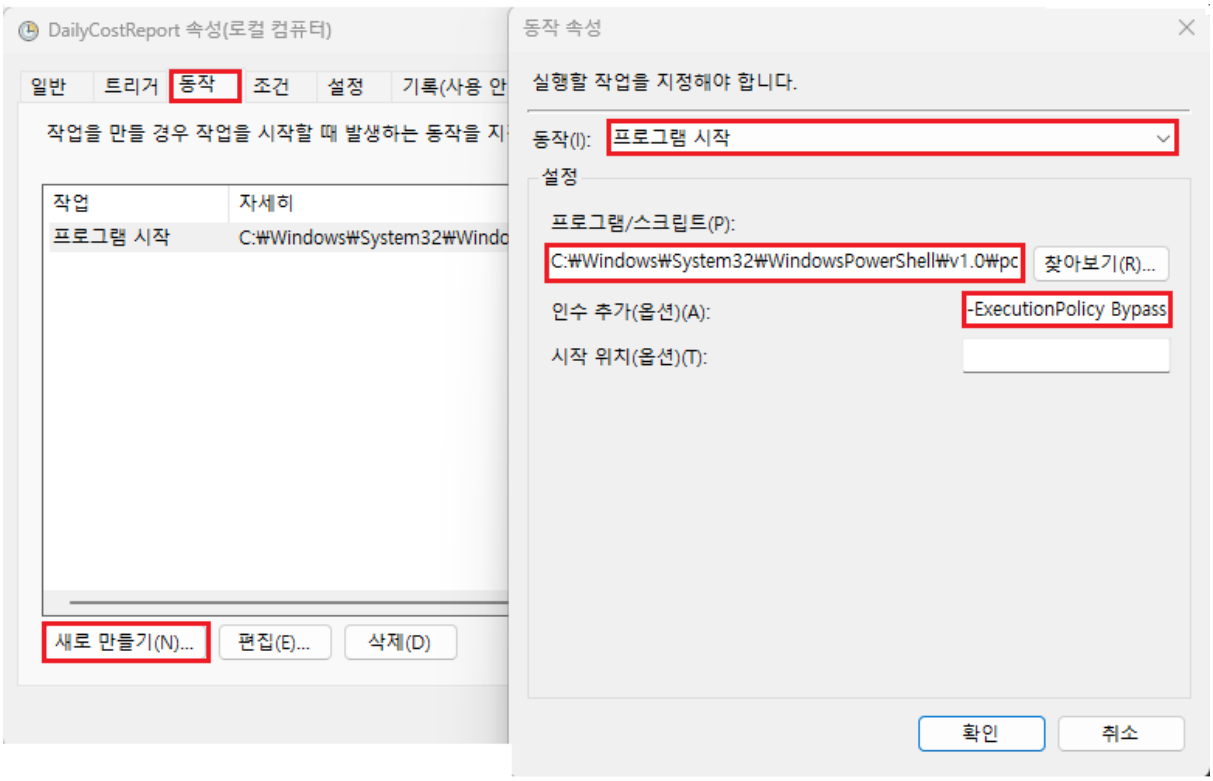
작업 스케줄러 라이브러리에서 작업 만들기를 클릭한다



이후 일반에서 이름과 설명을 원하는 대로 설정 해주고 사용자가 로드온할 때만 실행을 체크해준다



트리거 에서 새로 만들기 → 작업 시작은 예약 상태 → 설정은 매일 → 시작 시간은 원하는 시간대로 설정해준 후 확인

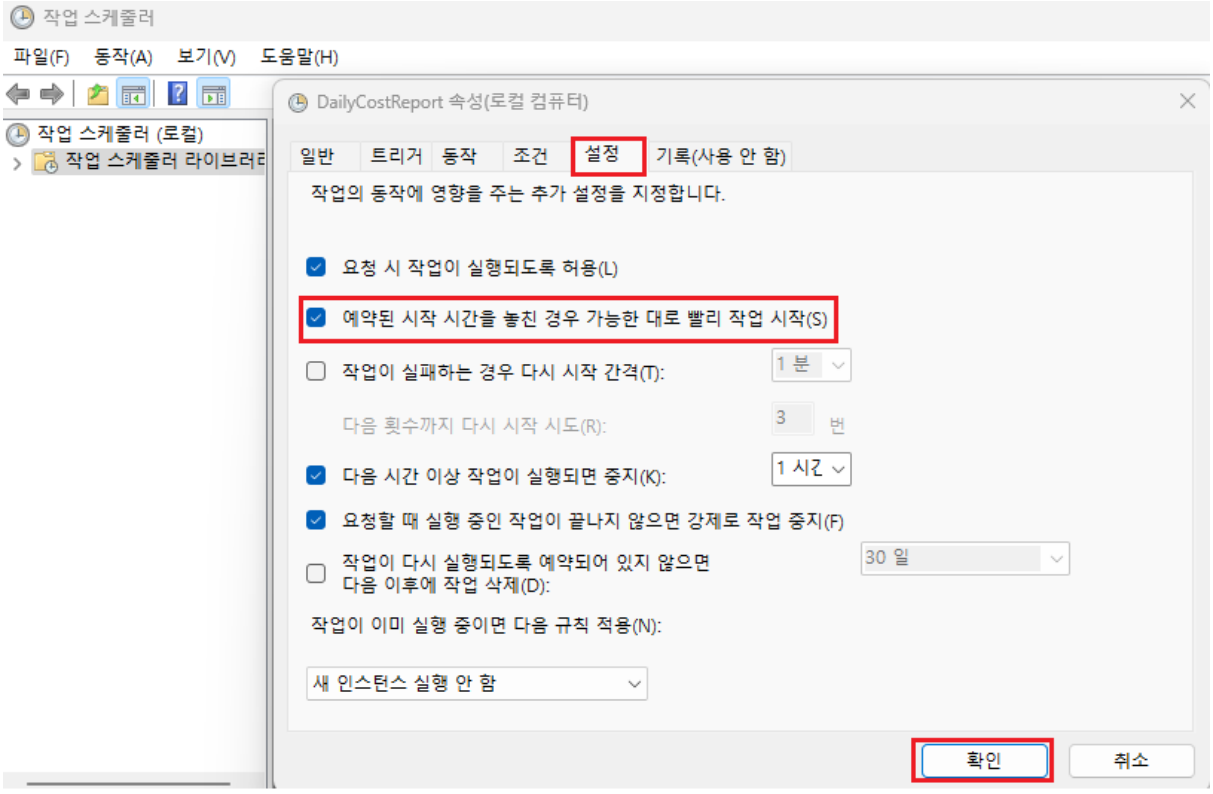


동작에서 새로 만들기 → 동작은 프로그램 시작 → 프로그램/스크립트 부분에 powershell 경로 입력

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

→ 인수 추가에

```
-ExecutionPolicy Bypass -File "스크립트 경로"
```



설정에 예약된 시작 시간을 놓친 경우 가능한 대로 빨리 작업 시작 체크 후 확인하면 끝이다.

C언어 성적처리

```
#include <stdio.h>
#include <windows.h>                                                    /
void init_ui();                                                         /
void print2DArray(int arr[][6], int col, int row);                      /
void Insert_value(int arr[][6], int col, int row);                      /
void Update_value(int arr[][6], int col, int row);                      /
int count =0;                                                           /

int main () {
    int ch;                                                             /
    int numArr[100][6] = {{0,},};                                       /
    int col = sizeof(numArr[0]) / sizeof(int);                          /
    int row = sizeof(numArr)/ sizeof(numArr[0]);                        /
    while(true){
        init_ui();                                                      /
        printf("\n원하는 메뉴를 선택해주세요.\n");
        scanf("%d",&ch);                                                /
        switch(ch){                                                     /
            case 1: Insert_value(numArr,col,row);
            case 2: print2DArray(numArr,col,row);
                    break;
            case 3: Update_value(numArr,col,row);
                    print2DArray(numArr,col,row);
            case 4: system("cls");                                      /
                    break;
            case 5: printf("프로그램을 종료합니다.\n");
                    break;
            defult: printf("입력을 잘못하셨습니다.");        // 다른
        }
        if(ch == 5) break;                                              /
    }
```

```
    return 0;
}
void init_ui(){                                                 /
    printf("____________________\n");                          /
    printf("__ 1. 성적입력    __\n");
    printf("__ 2. 성적조회    __\n");
    printf("__ 3. 성적수정    __\n");
    printf("__ 4. 초기화면으로__\n");
    printf("__ 5. 프로그램종료__\n");
    printf("____________________\n");
}



void print2DArray(int arr[][6], int col, int row){              /
    if (count ==0) printf("입력된 데이터가 없습니다.\n");    //할당
    for (int i =0; i < count; i++){                             /
        for (int j =0; j < col; j++)                            /
            printf("%5d",arr[i][j]);
        printf("\n");
    }
}

void Insert_value(int arr[][6], int col, int row){              /
    int cnt;                                                    /
    printf("몇명의 성적을 입력하시겠습니까?\n");
    scanf("%d",&cnt);                                           /
    for (int i = count; i <= count + cnt -1; i++){              /
        printf("학번을 입력해주세요.\n");
        scanf("%d",&arr[i][0]);
        printf("국어점수를 입력해주세요.\n");
        scanf("%d",&arr[i][1]);
        printf("수학점수를 입력해주세요.\n");
        scanf("%d",&arr[i][2]);
        printf("영어점수를 입력해주세요.\n");
        scanf("%d",&arr[i][3]);
        arr[i][4] = arr[i][1] + arr[i][2] + arr[i][3];          /
    }
    count += cnt;                                               /
```

```
    for (int i =0; i <= count -1; i++){                          /
        int rank =1;                                             /
        for (int j =0; j <= count -1; j++)                       /
            if (arr[i][4]<arr[j][4])                             /
                rank += 1;                                       /
        arr[i][5] = rank;                                        /
    }
}

void Update_value(int arr[][6], int col, int row){              /
    int hakbun, subject, jumsu;
    printf("\n수정을 원하는 학번을 입력하시오.\n");
    scanf("%d",&hakbun);                                         /
    while(true){
        printf("\n수정을 원하는 과목을 입력하시오.\n");
        printf("\n1.국어, 2.영어, 3.수학\n");
        scanf("%d",&subject);                                    /
        if(subject>3||subject<1)                                 /
            printf("\n과목명 입력이 잘못되었습니다.\n다시입력해 주십시
        else {                                                   /
            printf("\n수정을 원하는 점수를 입력하시오.\n");
        scanf("%d",&jumsu);                                      /
        break;
        }
    }

    for(int i =0; i<=count; i++){                                /
        if(arr[i][0]==hakbun){                                   /
            arr[i][subject]=jumsu;                               /
            arr[i][4] = arr[i][1] + arr[i][2] + arr[i][3];       /
            for(int k =0;k<=count-1;k++) {                       /
                int rank=1;                                      /
                for(int j=0;j<=count-1;j++)                      /
                    if(arr[k][4]<arr[j][4])                      /
                        rank+=1;                                 /
                    arr[k][5]=rank;                              /
            }
            printf("\n----수정된 데이터입니다.\n");
```

```
            break;                                                  /
        }
        if(i==count) printf("학번이 없습니다.");            //할당
    }
}
```

10/10

👉 기본적인 문법

```
#짝수 만으로 리스트 만들기
myList = [1,2,3,4,5,6,7,8,9,10]
even = []
for i in myList:
    if i&2 ==0:
        even.append(i)
print(even)

#한줄로 요약
even = [i for i in myList if i&2==0]
print(even)

#5보다 작은 정수
small = [i for i in myList if i<5]
print(small)

#변수값 가공도 가능
small = [i+10 for i in myList if i<5]
print(small, "변수 가공")

#문자열 쪼개기
a = 'This is an apple'
words = a.split(' ')
print(words)

a = 'This-is-an-apple'
temp = a.split('-')
```

```
print(temp[0])
print(temp[1])
print(temp[0]+temp[1])

#대소문자 전환
a= "ABCDEF"
print(a.lower())
print(a.upper())

#특정 문자열 시작/끝 부분 일치 여부

a= '01-smcimg.jpg'
b= '02-smcimg.png'
c= '03-smcimg.gif'
a.startswith('01')
a.endswith('png')

mylist = [a,b,c]
for file in mylist:
    if file.endswith('jpg'):
        print(file)

#특정 문자열 교체(replace)
a = '01-smcimg.jpg'
a.replace('jpg','bmp')
print(a.replace('jpg','bmp'))

#앞,뒤 공백 제거(strip)
a = '  kim semyeong' #앞 공백
b = 'kim semyeong'   #공백 없음
c = 'kim semyeong  ' #뒤 공백

a.strip()
c.strip()

#문자열 인덱싱
str = 'abcdef'
print(str[0])
```

```
print(str[3])
print(str[-1])
print(str[-2])

#문자열 슬라이싱
str = "Semyeong Computer Highschool"
print(str[:8]) #인덱스 ~7
print(str[9:17]) #인덱스 9~16
print(str[18:]) #인덱스 18~
```

👉 Turtle 라이브러리

Turtle

```
import ColabTurtle.Turtle as t
t.initializeTurtle()

#기본 사용법
t.forward(50)
t.left(45)
t.backward(50)
t.right(90)
t.forward(50)
t.color('blue')
t.left(90)
t.forward(100)
t.bgcolor('red')


#별
import ColabTurtle.Turtle as t
t.initializeTurtle()
for i in range(5):
  t.forward(100)
```

```
  t.left(145)

#광기
import ColabTurtle.Turtle as t
t.initializeTurtle()
t.speed(13)
for i in range(360):
  if(i%7==0):
    t.color('red')
  elif(i%7==1):
    t.color('orange')
  elif(i%7==2):
    t.color('yellow')
  elif(i%7==3):
    t.color('green')
  elif(i%7==4):
    t.color('blue')
  elif(i%7==5):
    t.color('Indigo')
  elif(i%7==6):
    t.color('purple')
  for j in range(6):
    t.forward(100)
    t.left(60)
  t.left(1)
```

👉 Turtle 신기하네

📘

10/12

👉 format 과 pyqrcode

```
#변수 출력 format

print('나는 %s 고등학교 %d학년 %d 반입니다'%('세명컴퓨터',2,3))
print('나는 {}고등학교 {}학년 {}반입니다'.format('세명컴퓨터',2,3))
print('%d는 16진수로 %x입니다.'%(30,30))

a = '세명컴고'
b = '홍길동'
print(f'안녕하세요 {a}에 다니는 {b}입니다.')
print(f'안녕하세요 {{{a}}}에 다니는 {{{b}}}입니다.')
```

```
#pyqrcode

import pyqrcode
qrcode = pyqrcode.create('https://www.naver.com',error="H",mo
qrcode.png("qr_naver.png", scale=8, module_color=[0,0,0,255],
qrcode.show()

import pyqrcode
data = "WIFI:S:senWiFi_Free;T:WPA;P:sem2021!wi:"
qrcode = pyqrcode.create(data, mode='binary')
qrcode.png('qr_wifi.png',scale=8, module_color=[10,10,50,255]


import pyqrcode
print("#### WIFI 접속 스크립트 ####")
```

```
ssid = input("와이파이 SSID를 입력하세요 :")
secu = input("암호화 방식을 입력하세요(WPA, WPA2, WPA2=PSK, TKIP 등
pwd = input("와이파이 패스워드를 입력하세요 :")

qrcode = pyqrcode.create(f"WIFI:S:{ssid};T:{secu};P:{pwd}:")
qrcode.png('qr_wifi2.png',scale=8,module_color=[0,0,0,255], b
    qrcode.show()
```

```
#명함 QR 만들기

import pyqrcode

data = "BEGIN:VCARD\n"
data += "VERSION:3.0\n"
data += "FN:김선생\n"
data += "TELL:TYPE=WORK : CELL:010 1234 1234\n"
data += "END:VCARD\n"

qrcode = pyqrcode.create(data, mode='binary', encoding='utf-8
qrcode.png('qr)namecard.png', scale=6, module_color=[10,10,10
```

```
#명함 큰 버전

import pyqrcode

data = "BEGIN:VCARD\n"
data += "VERSION:3.0\n"
data += "FN:김선생\n"
data += "TEL:TYPE=WORK ;CELL:010 1234 1234\n"
data += "END:VCARD\n"

qrcode = pyqrcode.create(data, mode='binary', encoding='utf-8
qrcode.png('qr_namecard.png', scale=6, module_color=[10,10,10
import pyqrcode

data = "BEGIN:VCARD\n"
```

```
data += "VERSION:3.0\n"
data += "FN:김선생\n"
data += "ORG:세명컴퓨터고등학교\n"
data += "TITLE:교사\n"
data += "TEL;TYPE=WORK ;CELL:010 1234 1234\n"
data += "TEL;TYPE=HOME ;CELL:010 4444 5555\n"
data += "EMAIL;TYPE=WORK:abcd@naver.com\n"
data += "URL;TYPE=WORK:https:smc.sen.go.kr\n"
data += "END:VCARD\n"

qrcode = pyqrcode.create(data, mode='binary', encoding='utf-8
qrcode.png('qr_namecarddddd.png', scale=6, module_color=[10,1
```

💡 vcard는 아이폰과 안드로이드에 따라 다름 ㅇㅇ

📕

11-28

📢 문제: `ONLINE_SALE` 테이블에서 동일한 회원이 동일한 상품을 재구매한 데이터를 구하여, 재구매한 회원 ID와 재구매한 상품 ID를 출력하는 SQL문을 작성해주세요. 결과는 회원 ID를 기준으로 오름차순 정렬해주시고 회원 ID가 같다면 상품 ID를 기준으로 내림차순 정렬해주세요.

👉 입력:

ONLINE_SALE_ID	USER_ID	PRODUCT_ID	SALES_AMOUNT	SALE
1	1	3	2	2022
2	1	4	1	2022
3	1	3	3	2022
4	2	4	2	2022
5	3	5	1	2022
6	2	4	1	2022
7	1	4	2	2022

👉 출력:

USER_ID	PRODUCT_ID
1	4

```
| 1          | 3            |
| 2          | 4            |
```

💡 풀이:

```
SELECT USER_ID, PRODUCT_ID FROM ONLINE_SALE
GROUP BY USER_ID, PRODUCT_ID HAVING COUNT(*) > 1 ORDER BY 1,
```

📌 해석:

```
HAVING절의 사용법은 여기에서 -> 여기

GROUP BY 로 두개를 지정해서 두개를 한 그룹으로 보고 그 그룹의 COUNT가 1
```

💡 문제 후기: SQL 쉽지 않다!

📕

11-29

📢 `ONLINE_SALE` 테이블과 `OFFLINE_SALE` 테이블에서 2022년 3월의 오프라인/온라인 상품 판매 데이터의 판매 날짜, 상품ID, 유저ID, 판매량을 출력하는 SQL문을 작성해주세요. `OFFLINE_SALE` 테이블의 판매 데이터의 `USER_ID` 값은 NULL 로 표시해주세요. 결과는 판매일을 기준으로 오름차순 정렬해주시고 판매일이 같다면 상품 ID를 기준으로 오름차순, 상품ID까지 같다면 유저 ID를 기준으로 오름차순 정렬해주세요.

👉 입력:

```
ONLINE_SALE TABLE
| ONLINE_SALE_ID | USER_ID | PRODUCT_ID | SALES_AMOUNT | SALES
|----------------|---------|------------|--------------|-----
| 1              | 1       | 3          | 2            | 2022
| 2              | 4       | 4          | 1            | 2022
| 3              | 2       | 2          | 2            | 2022
| 4              | 6       | 3          | 3            | 2022
| 5              | 5       | 5          | 1            | 2022
| 6              | 5       | 7          | 1            | 2022

OFFLINE_SALE TABLE
| OFFLINE_SALE_ID | PRODUCT_ID | SALES_AMOUNT | SALES_DATE  |
|-----------------|------------|--------------|-------------
| 1               | 1          | 2            | 2022-02-21
| 2               | 4          | 1            | 2022-03-01
| 3               | 3          | 3            | 2022-03-01
| 4               | 1          | 2            | 2022-03-01
```

```
| 5                | 2           | 1             | 2022-03-03
| 6                | 2           | 1             | 2022-04-01
```

👉 출력:

```
| SALES_DATE  | PRODUCT_ID | USER_ID | SALES_AMOUNT |
|-------------|------------|---------|--------------|
| 2022-03-01  | 1          | NULL    | 2            |
| 2022-03-01  | 3          | NULL    | 3            |
| 2022-03-01  | 4          | NULL    | 1            |
| 2022-03-01  | 4          | 4       | 1            |
| 2022-03-02  | 2          | 2       | 2            |
| 2022-03-02  | 3          | 6       | 3            |
| 2022-03-03  | 2          | NULL    | 1            |
| 2022-03-03  | 5          | 5       | 1            |
```

💡 풀이:

```
SELECT DATE_FORMAT(SALES_DATE, "%Y-%m-%d") SALES_DATE, PRODUC
FROM ONLINE_SALE WHERE SALES_DATE BETWEEN "2022-03-01" AND "2
UNION ALL
SELECT DATE_FORMAT(SALES_DATE, "%Y-%m-%d") SALES_DATE, PRODUC
FROM OFFLINE_SALE WHERE SALES_DATE BETWEEN "2022-03-01" AND "
ORDER BY SALES_DATE, PRODUCT_ID, USER_ID;
```

📌 해석:

```
UNION ALL절의 사용법은 여기에서 -> 여기
UNION ALL을 이용해서 두 테이블의 검색 값 두개를 합쳐서 나오게 함
```

없는 칼럼은 NULL로 바꾸기 위해 NULL 칼럼_이름

💡 문제 후기: SQL 뭐 이렇게 많냐

📕

11-30

📢 `ONLINE_SALE` 테이블과 `OFFLINE_SALE` 테이블에서 2022년 3월의 오프라인/온라인 상품 판매 데이터의 판매 날짜, 상품ID, 유저ID, 판매량을 출력하는 SQL문을 작성해주세요. `OFFLINE_SALE` 테이블의 판매 데이터의 `USER_ID` 값은 NULL 로 표시해주세요. 결과는 판매일을 기준으로 오름차순 정렬해주시고 판매일이 같다면 상품 ID를 기준으로 오름차순, 상품ID까지 같다면 유저 ID를 기준으로 오름차순 정렬해주세요.

👉 입력:

```
BOOK TABLE
| BOOK_ID | CATEGORY | AUTHOR_ID | PRICE | PUBLISHED_DATE |
|---------|----------|-----------|-------|-----------------|
| 1       | 인문     | 1         | 10000 | 2020-01-01      |
| 2       | 경제     | 1         | 9000  | 2021-04-11      |
| 3       | 경제     | 2         | 11000 | 2021-02-05      |

AUTHOR TABLE
| AUTHOR_ID | AUTHOR_NAME |
|-----------|-------------|
| 1         | 홍길동      |
| 2         | 김영호      |
```

👉 출력:

```
| BOOK_ID | AUTHOR_NAME    | PUBLISHED_DATE   |
|---------|----------------|------------------|
| 3       | 김영호         | 2021-02-05       |
| 2       | 홍길동         | 2021-04-11       |
```

💡 풀이:

```
SELECT b.BOOK_ID, a.AUTHOR_NAME, DATE_FORMAT(b.PUBLISHED_DATE
FROM BOOK AS b JOIN AUTHOR AS a ON b.AUTHOR_ID  = a.AUTHOR_ID
WHERE b.CATEGORY = '경제' ORDER BY PUBLISHED_DATE ;
```

📌 해석:

```
JOIN 문사용법은 여기
JOIN을 이용해서 두 테이블에서 원하는 칼럼을 지정하고 옵션 지정하고 출력
```

💡 문제 후기: JOIN 종류도 많네

📕

12-01

📢 `ONLINE_SALE` 테이블과 `OFFLINE_SALE` 테이블에서 2022년 3월의 오프라인/온라인 상품 판매 데이터의 판매 날짜, 상품ID, 유저ID, 판매량을 출력하는 SQL문을 작성해주세요. `OFFLINE_SALE` 테이블의 판매 데이터의 `USER_ID` 값은 NULL 로 표시해주세요. 결과는 판매일을 기준으로 오름차순 정렬해주시고 판매일이 같다면 상품 ID를 기준으로 오름차순, 상품ID까지 같다면 유저 ID를 기준으로 오름차순 정렬해주세요.

👉 입력:

```
| ANIMAL_ID | ANIMAL_TYPE |       DATETIME         | INTAKE_CO
|-----------|-------------|------------------------|----------
| A350276   | Cat         | 2017-08-13 13:50:00    | Normal
| A350375   | Cat         | 2017-03-06 15:01:00    | Normal
| A368930   | Dog         | 2014-06-08 13:20:00    | Normal
```

👉 출력:

```
| ANIMAL_ID | ANIMAL_TYPE | DATETIME               | INTAKE_CO
|-----------|-------------|------------------------|----------
| A350276   | Cat         | 2017-08-13 13:50:00    | Normal
| A350375   | Cat         | 2017-03-06 15:01:00    | Normal
| A368930   | Dog         | 2014-06-08 13:20:00    | Normal
```

💡 풀이:

```
SELECT ANIMAL_TYPE, IFNULL(NAME,'No name') AS NAME, SEX_UPON_
FROM ANIMAL_INS ORDER BY ANIMAL_ID;
```

📌 해석:

```
IFNULL 문사용법은 여기
IFNULL를 사용해서 출력하면 됩니다.
```

💡 문제 후기: 이번 문제는 쉬웠다

HAVING절

`HAVING` 절은 SQL 쿼리에서 그룹화된 결과에 조건을 적용하는 데 사용됩니다. `HAVING`은 `GROUP BY` 절과 함께 사용되어 그룹 함수 (예: `SUM`, `COUNT`, `AVG` 등)를 기반으로 한 조건을 지정합니다.

일반적인 구문은 다음과 같습니다:

```
sqlCopy code
SELECT column1, column2, aggregate_function(column3)
FROM table
GROUP BY column1, column2
HAVING condition;
```

여기서:

- `column1`, `column2`: 그룹화할 열입니다.
- `aggregate_function(column3)`: 그룹 함수를 사용하여 계산할 열입니다.
- `condition`: 그룹화된 결과에 대한 조건입니다.

예를 들어, `EMPLOYEES` 테이블에서 각 부서별로 평균 급여가 50000보다 큰 경우를 찾는 쿼리는 다음과 같습니다:

```
sqlCopy code
SELECT DEPARTMENT_ID, AVG(SALARY) AS AVERAGE_SALARY
FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING AVG(SALARY) > 50000;
```

이 쿼리에서 `GROUP BY DEPARTMENT_ID`는 각 부서별로 그룹화하고, `AVG(SALARY) AS AVERAGE_SALARY`는 각 부서의 평균 급여를 계산합니다. 그리고 `HAVING AVG(SALARY) > 50000`은 그룹화된 결과 중에서 평균 급여가 50000보다 큰 경우만 선택합니다.

즉, `HAVING`은 그룹화된 결과에 대한 필터링을 수행하는 역할을 합니다. 이는 `WHERE` 절이 개별 행에 대한 조건을 지정하는 데 사용되는 것과는 구별됩니다.

POWERED BY CHAT GPT

IFNULL문

`IFNULL` 함수는 주로 MySQL에서 사용되며, 특정 값이 `NULL`인 경우 대체값을 반환하는 데 사용됩니다. 다른 데이터베이스 시스템에서는 비슷한 기능을 수행하는 함수가 있을 수 있습니다. 아래는 MySQL에서의 `IFNULL` 사용법입니다:

```
sqlCopy code
SELECT IFNULL(column_name, alternative_value) AS new_column_n
FROM your_table;
```

여기서:

- `column_name`: 대체하기 전에 확인하려는 컬럼입니다.
- `alternative_value`: `column_name`이 `NULL`일 경우 대신 사용할 값입니다.
- `new_column_name`: 결과로 반환될 컬럼의 별칭입니다.

예제:

```
sqlCopy code
-- 'price' 컬럼이 NULL이면 0으로 대체
SELECT IFNULL(price, 0) AS adjusted_price
FROM products;
```

POWERED BY CHAT GPT

JOIN문

`JOIN`은 SQL에서 여러 테이블의 데이터를 결합하는 데 사용되는 연산자입니다. 여러 가지 종류의 `JOIN`이 있으며, 주요 종류로는 `INNER JOIN`, `LEFT JOIN` (또는 `LEFT OUTER JOIN`), `RIGHT JOIN` (또는 `RIGHT OUTER JOIN`), `FULL JOIN` (또는 `FULL OUTER JOIN`)이 있습니다. 아래에서 각각의 `JOIN`에 대한 설명과 사용법을 알려드리겠습니다.

1. INNER JOIN:

`INNER JOIN`은 두 테이블 간에 일치하는 행만을 반환합니다. `INNER JOIN`은 일반적으로 가장 많이 사용되는 `JOIN` 유형 중 하나입니다.

사용법:

```
sqlCopy code
SELECT *
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

2. LEFT JOIN (또는 LEFT OUTER JOIN):

`LEFT JOIN`은 왼쪽 테이블의 모든 행과 오른쪽 테이블에서 일치하는 행을 반환합니다. 오른쪽 테이블에 일치하는 값이 없는 경우에도 왼쪽 테이블의 모든 행이 결과에 포함됩니다.

사용법:

```
sqlCopy code
SELECT *
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

3. RIGHT JOIN (또는 RIGHT OUTER JOIN):

`RIGHT JOIN`은 오른쪽 테이블의 모든 행과 왼쪽 테이블에서 일치하는 행을 반환합니다. 왼쪽 테이블에 일치하는 값이 없는 경우에도 오른쪽 테이블의 모든 행이 결과에 포함됩니다.

사용법:

```
sqlCopy code
SELECT *
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

4. FULL JOIN (또는 FULL OUTER JOIN):

`FULL JOIN`은 양쪽 테이블 중 하나에만 있는 행을 포함하여 모든 행을 반환합니다. 어느 한 쪽이라도 일치하지 않는 경우에도 결과에 포함됩니다.

사용법:

```
sqlCopy code
SELECT *
FROM table1
FULL JOIN table2 ON table1.column_name = table2.column_name;
```

5. CROSS JOIN:

`CROSS JOIN`은 두 테이블 간의 모든 가능한 조합을 반환합니다. 이는 특별한 조건 없이 두 테이블의 모든 행을 결합하는 것입니다.

사용법:

```
sqlCopy code
SELECT *
FROM table1
CROSS JOIN table2;
```

`JOIN`은 데이터를 효과적으로 결합하는 데 사용되며, 테이블 간의 관계에 따라 적절한 `JOIN` 유형을 선택하는 것이 중요합니다.

POWERED BY CHAT GPT

UNION ALL

`UNION ALL`은 SQL에서 사용되는 연산자 중 하나로, 두 개 이상의 SELECT 문의 결과를 합칠 때 사용됩니다.

`UNION ALL`을 사용하면 중복된 행을 포함하여 모든 결과를 하나로 합칩니다.

기본적인 구문은 다음과 같습니다:

```
sqlCopy code
SELECT column1, column2, ...
FROM table1
UNION ALL
SELECT column1, column2, ...
FROM table2;
```

여기서 각 `SELECT` 문은 동일한 열 수와 유사한 데이터 유형을 가져야 합니다.

`UNION ALL`을 사용하면 중복된 행도 모두 결과에 포함됩니다.

예를 들어, `EMPLOYEES` 테이블과 `MANAGERS` 테이블에서 이름이나 ID 등의 열을 합칠 경우:

```
sqlCopy code
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
FROM EMPLOYEES
UNION ALL
SELECT MANAGER_ID, FIRST_NAME, LAST_NAME
FROM MANAGERS;
```

`UNION ALL`을 사용하면 중복된 행이 모두 결과에 포함되므로,

중복을 제거하려면 `UNION`을 사용할 수도 있습니다.

다만, `UNION`은 중복된 행을 하나로 합치는데 반해 `UNION ALL`은 중복된 행도 모두 포함시킨다는 차이가 있습니다.

POWERED BY CHAT GPT