

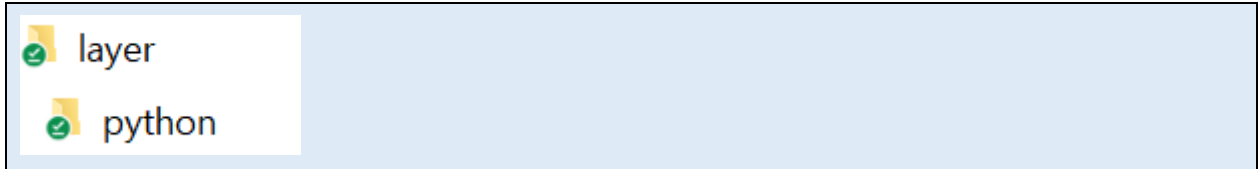
Programming of RDS using Lambda

1. Python용 Pymysql 패키지 Layer 추가
2. mysql에 연결하기
3. SELECT문으로 데이터 가져오기
4. insert / update / delete문 사용하기

1. pymysql 패키지 레이어에 추가하기

1) pymysql 다운로드 및 설치

① 아래와 같은 구조로 폴더를 생성한다.



② 관리자 권한으로 "명령 프롬프트"를 실행합니다.

③ CD 명령어를 사용하여 "python"폴더로 이동합니다.

```

관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.19044.2364]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd #
C:\>cd googledrive
C:\GoogleDrive>cd 01_study
C:\GoogleDrive\01_STUDY>cd 04_cloud
C:\GoogleDrive\01_STUDY\04_Cloud>cd 0402_developing on aws
C:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS>cd layer
C:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer>cd python
C:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer\python>
  
```

③ 파이썬용 pymysql 설치 및 다운로드.

```
C:\W...W...Wpython>pip install pymysql
```

④ 설치가 잘 되었는지 확인

```
C:\W...W...Wpython>pip list
```

⑤ 설치 및 다운로드를 통해 디렉토리(폴더)가 생성되었는지 확인

```
C:\W...W...Wpython>dir/w
```

```

c:\#GoogleDrive#01_STUDY#04_Cloud#0402_Developing On AWS#layer#python>pip install pymysql
Collecting pymysql
  Using cached PyMySQL-1.0.2-py3-none-any.whl (43 kB)
Installing collected packages: pymysql
Successfully installed pymysql-1.0.2

c:\#GoogleDrive#01_STUDY#04_Cloud#0402_Developing On AWS#layer#python>pip list
Package Version
-----
pip      22.3.1
PyMySQL  1.0.2
setuptools 65.5.0

c:\#GoogleDrive#01_STUDY#04_Cloud#0402_Developing On AWS#layer#python>dir/w
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 082D-12E3

c:\#GoogleDrive#01_STUDY#04_Cloud#0402_Developing On AWS#layer#python 디렉터리
[.]
                0개 파일                0 바이트 [pymysql] [PyMySQL-1.0.2.dist-info]
                4개 디렉터리 132,327,088,128 바이트 남음

c:\#GoogleDrive#01_STUDY#04_Cloud#0402_Developing On AWS#layer#python>
  
```

2) 라이브러리 압축

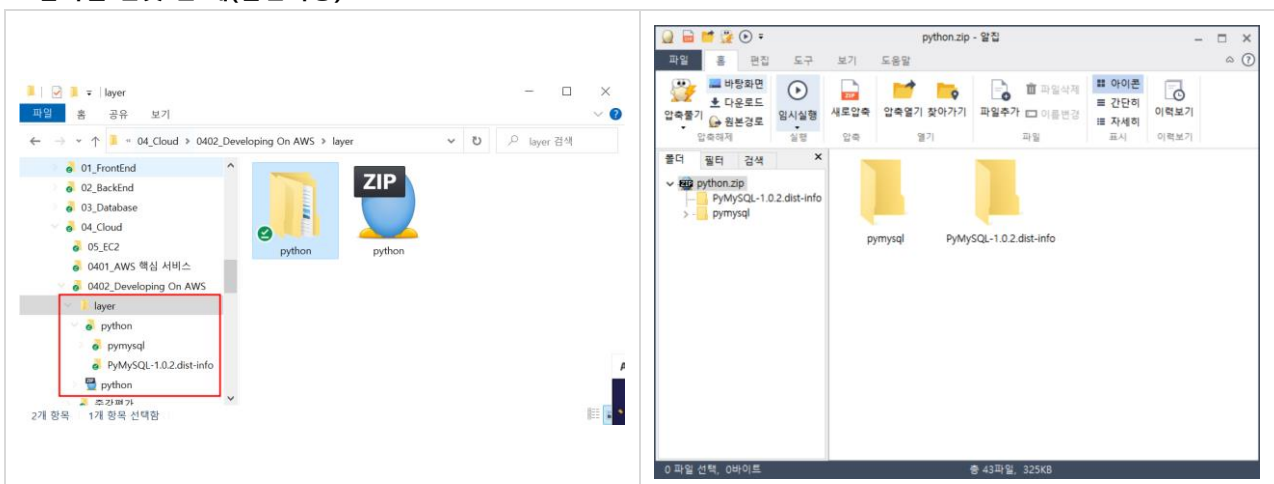
이 부분은 아래 이미지를 보고 동일한 형태로 압축이 되었는지 반드시 확인이 필요하다. 반드시 압축파일의 구조가 아래와 같이 생성이 되어있어야 한다.

• 디렉터리(폴더) 구조



① 압축 유틸리티를 사용하여 압축하는 방법

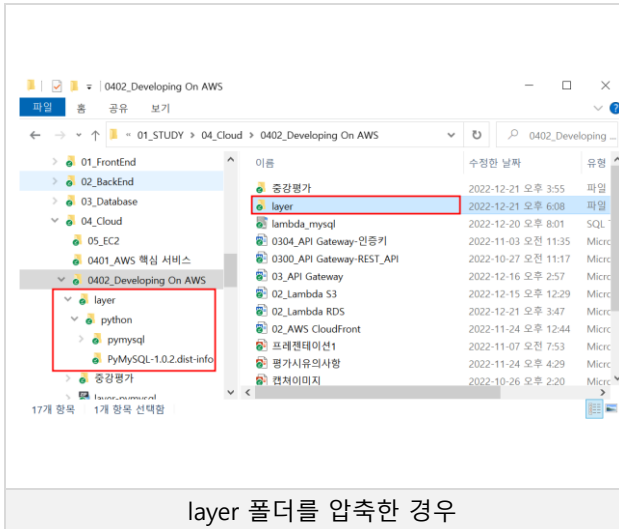
• 압축을 잘못 한 예(알집사용)



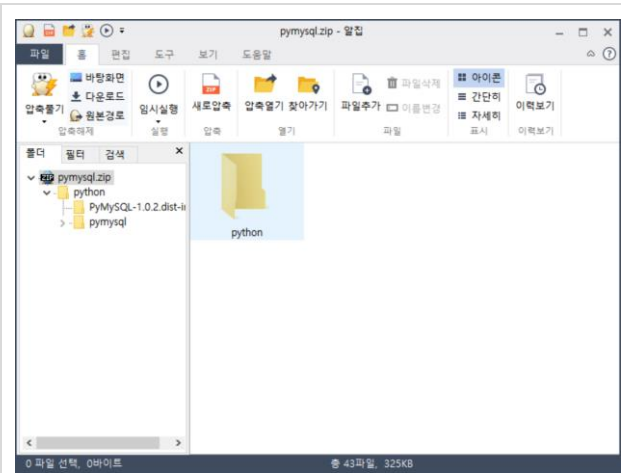
Python 폴더를 압축한 경우

압축파일의 내부구조에 python이 없음

• 올바른 압축 예(알집사용)



layer 폴더를 압축한 경우



압축파일의 내부구조에 python이 있음

② 명령 프롬프트를 사용하여 압축하는 방법

- 아래 사이트를 통해 zip, unzip 파일을 다운받는다.

<http://stahlworks.com/dev/?tool=zipunzip>

StahlWorks Technologies. Software Tools For Professionals.



- 다음 경로에 다운로드 받은 zip.exe 파일과 unzip.exe파일을 복사한다.

C:\Windows\System32

- 명령 프롬프트를 실행하고 "cd"명령을 이용하여 python폴더의 부모 폴더로 이동한다. "dir/w"명령을 이용하여 "python" 폴더가 아래 그림과 같이 확인이 되어야 한다.

```

C:\WINDOWS\system32\cmd.exe

c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer>cd #
c:\>cd c:\googledrive
c:\GoogleDrive>cd 01_study
c:\GoogleDrive\01_STUDY>cd 04_cloud
c:\GoogleDrive\01_STUDY\04_Cloud>cd 0402_Developing on aws
c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS>cd layer
c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer>dir/w
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 082D-12E3

c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer 디렉터리
[.]      [..]      [python]
0개 파일              0 바이트
3개 디렉터리 132,248,911,872 바이트 남음

c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer>
  
```

- zip명령을 이용하여 "python"폴더를 압축한다.

zip [option] [압축결과파일명] [압축대상]

[option]

압축률 : 1~9, 높을수록 압축률이 좋음

v : 압축되는 진행상황을 보여줌

m : 압축 대상 파일들을 압축파일로 이동(거의 사용 안됨)

r : 하위 디렉토리(폴더)의 파일들까지 모두 묶어 압축

Zip -9vr pymysql.zip ./python

```

C:\WINDOWS\system32\cmd.exe

c:\GoogleDrive\01_STUDY\04_Cloud\0402_Developing On AWS\layer>zip -9vr pymysql.zip ./python
adding: python/ (192 bytes security) (in=0) (out=0) (stored 0%)
adding: python/pymysql/ (164 bytes security) (in=0) (out=0) (stored 0%)
adding: python/pymysql/charset.py (164 bytes security) (in=10293) (out=1778) (deflated 83%)
adding: python/pymysql/connections.py (164 bytes security) (in=51251) (out=12512) (deflated 76%)
adding: python/pymysql/constants/ (164 bytes security) (in=0) (out=0) (stored 0%)
adding: python/pymysql/constants/CLIENT.py (164 bytes security) (in=878) (out=430) (deflated 51%)
adding: python/pymysql/constants/COMMAND.py (164 bytes security) (in=679) (out=292) (deflated 57%)
adding: python/pymysql/constants/CR.py (164 bytes security) (in=1927) (out=695) (deflated 64%)
adding: python/pymysql/constants/ER.py (164 bytes security) (in=12296) (out=4595) (deflated 63%)
adding: python/pymysql/constants/FIELD_TYPE.py (164 bytes security) (in=370) (out=204) (deflated 45%)
adding: python/pymysql/constants/FLAG.py (164 bytes security) (in=214) (out=147) (deflated 31%)
adding: python/pymysql/constants/SERVER_STATUS.py (164 bytes security) (in=333) (out=165) (deflated 50%)
adding: python/pymysql/constants/_init_.py (172 bytes security) (in=0) (out=0) (stored 0%)
adding: python/pymysql/constants/_pycache_/ (164 bytes security) (in=0) (out=0) (stored 0%)
adding: python/pymysql/constants/_pycache_/CLIENT.cpython-311.pyc (164 bytes security) (in=1041) (out=764) (deflated 27%)
adding: python/pymysql/constants/_pycache_/COMMAND.cpython-311.pyc (164 bytes security) (in=1136) (out=708) (deflated 38%)
adding: python/pymysql/constants/_pycache_/CR.cpython-311.pyc (164 bytes security) (in=2646) (out=1365) (deflated 48%)
adding: python/pymysql/constants/_pycache_/ER.cpython-311.pyc (164 bytes security) (in=17369) (out=7899) (deflated 55%)
adding: python/pymysql/constants/_pycache_/FIELD_TYPE.cpython-311.pyc (164 bytes security) (in=850) (out=596) (deflated 30%)
adding: python/pymysql/constants/_pycache_/FLAG.cpython-311.pyc (164 bytes security) (in=542) (out=434) (deflated 20%)
adding: python/pymysql/constants/_pycache_/SERVER_STATUS.cpython-311.pyc (164 bytes security) (in=642) (out=447) (deflated 30%)
adding: python/pymysql/constants/_pycache_/__init__.cpython-311.pyc (164 bytes security) (in=203) (out=167) (deflated 18%)
  
```

3) 계층 생성

AWS 콘솔 로그인 → AWS Lambda → 추가 리소스 → 계층 → 계층 생성

계층 구성

이름

설명 - 선택 사항

☒ .zip 파일 업로드

☐ Amazon S3에서 파일 업로드

pymysql.zip (130.8kB)

10MB가 넘는 파일의 경우, Amazon S3를 사용한 업로드를 고려하십시오.

호환 아키텍처 - 선택 사항 정보

계층에 대해 호환 명령 세트 아키텍처를 선택합니다.

☒ x86_64

☐ arm64

호환 런타임 - 선택 사항 정보

최대 15개의 런타임을 선택합니다.

실행 시간

Python 3.9 X

라이선스 - 선택 사항 정보

취소

생성

4) Lambda 함수에 계층 추가

AWS 콘솔 로그인 → AWS Lambda → 함수 → 함수 선택 → 페이지 하단 계층 → [Add a layer]

계층 추가

함수 런타임 설정

런타임
Python 3.9

아키텍처
x86_64

계층 선택

계층 소스 정보

호환 런타임 및 명령 세트 아키텍처가 있는 계층에서 선택하거나 계층 버전의 Amazon 리소스 이름(ARN)을 지정합니다. 새로운 계층을 생성할 수도 있습니다.

☐ AWS 계층

AWS에서 제공하는 계층 목록에서 계층을 선택합니다.

☒ 사용자 지정 계층

AWS 계정 또는 조직에서 생성한 계층 목록에서 계층을 선택합니다.

☐ ARN 지정

ARN을 제공하여 계층을 지정합니다.

사용자 지정 계층

함수가 런타임과 호환되는 AWS 계정 또는 조직 생성 계층입니다.

lambda-layer-python39-pymysql

버전

1

취소

추가

▼ 함수 개요 정보



ex-rds-mysql-connect



Layers

(1)

+ 트리거 추가

+ 대상 추가

2. pymysql 주요 객체 및 메서드

➔ 깃허브에 업로드한 "PyMySQL-0.7_Documentation.pdf"참고

3. SELECT문을 이용한 데이터 추출

1) 절차

데이터베이스 연결 → 커서 생성 → 쿼리 실행 → Fetch → 연결 종료

2) 기본 Lambda Function

함수명 : ex-rds-mysql-select

```
import json
import pymysql

def lambda_handler(event, context):
    conn = pymysql.connect(
        host="aws-rds-mysql. xxxxxxxxxxxx.ap-northeast-2.rds.amazonaws.com",
        user="admin", password="XXXX", database=" db_name", port=3306, charset="utf8"
    )
    cursor = conn.cursor(pymysql.cursors.DictCursor)
    cursor.execute("select fid, fpass, fname, femail, fphone, faddr1, faddr2, fbirthday, fgender, fdate
from TMEMBER")
    rows=cursor.fetchall()

    arr_rows = []
    for row in rows:
        json_row = {}
        for key in row:
            json_row[key]=row[key]
        arr_rows.append(json_row)
        print(json.dumps(json_row, default=str, indent=2))

    conn.close()
    return {
        'statusCode': 200,
        "body": json.dumps(arr_rows, default=str, indent=2)
    }
```

3) 코드 주요 내용

- **pymysql.cursors.DictCursor**

컬럼 인덱스가 아닌 컬럼 이름으로 데이터 추출하기 위한 옵션

- **json.dumps(arr_rows, default=str, indent=2)**

Json에 삽입된 데이터 중 날짜 형식의 데이터가 있을 경우 문자열로 변환이 되지 않는다. 이를 허용해주기 위한 옵션으로 default=str, indent=2 를 추가해주면 된다.

4. INSERT문을 이용한 데이터 추가

1) 절차

데이터베이스 연결 → 커서 생성 → 쿼리 실행 → 커밋 → 연결 종료

2) 기본 Lambda Function

함수명 : ex-rds-mysql-insert

```
import json
import pymysql

def lambda_handler(event, context):
    conn = pymysql.connect(
        host="aws-rds-mysql. xxxxxxxxxxxx.ap-northeast-2.rds.amazonaws.com",
        user="admin", password="XXXX", database=" db_name", port=3306, charset="utf8"
    )
    # 쿼리를 실행할 커서 생성
    curs = conn.cursor()
    sql = """INSERT INTO TMEMBER VALUE (%s, %s, %s, %s, %s, %s, %s, %s,%s, now())"""
    result = curs.execute(sql,(
        "abc2222", "1111", "최영", "cy@naver.com", "010-4444-4444",
        "경기도 화성시 와우리", "수원대학교", "1991-01-01", "남"
    ))
    conn.commit()
    conn.close()
    return {
        'statusCode': 200,
        'body': json.dumps(result)
    }
```

3) 응용 Lambda Function

위 함수를 API Gateway를 통해 데이터를 입력 받아 실행 할 수 있도록 수정

5. UPDATE문을 이용한 데이터 수정

1) 절차

데이터베이스 연결 → 커서 생성 → 쿼리 실행 → 커밋 → 연결 종료

2) 기본 Lambda Function

함수명 : ex-rds-mysql-update

```
import json
import pymysql
def lambda_handler(event, context):
    conn = pymysql.connect(
        host="swu-mysql.xxxxxxxx.ap-northeast-2.rds.amazonaws.com", #AWS RDS MySQL Server의 End Point
        user="admin", password="mzc2023!", database="STUDY", port=3306
    )
    curs = conn.cursor()
    uid = event["queryStringParameters"]["uid"]
    passwd = event["queryStringParameters"]["upasswd"]
    uname = event["queryStringParameters"]["uname"]
    email = event["queryStringParameters"]["email"]
    phone = event["queryStringParameters"]["phone"]
    addr1 = event["queryStringParameters"]["addr1"]
    addr2 = event["queryStringParameters"]["addr2"]
    birthday = event["queryStringParameters"]["birthday"]
    gender = event["queryStringParameters"]["gender"]
    sql = "UPDATE TMEMBER SET fname=%s, femail=%s, fphone=%s, faddr1=%s, faddr2=%s, fbirthday=%s, fgender=%s "
    sql = sql + " WHERE fid = %s AND fpass = %s "
    result = curs.execute(sql, (uname, email, phone, addr1, addr2, birthday, gender, uid, passwd))
    conn.commit()
    return {
        'statusCode': 200,
        'body': json.dumps(result)
    }
```

3) 테스트 이벤트의 이벤트 JSON

```
{
  "queryStringParameters": {
    "uid": "ccc-2",
    "upasswd": "111",
    "uname": "장동건",
    "email": "shj@gmail.com",
    "phone": "010-8888-8888",
    "addr1": "제주도 제주시 효행로",
    "addr2": "222동 3333호",
    "birthday": "2004-12-31",
    "gender": "남"
  }
}
```

6. DELETE문을 이용한 행 삭제

1) 절차

데이터베이스 연결 → 커서 생성 → 쿼리 실행 → 커밋 → 연결 종료

2) 기본 Lambda Function

함수명 : ex-rds-mysql-delete

```
import json
import pymysql

def lambda_handler(event, context):
    conn = pymysql.connect(
        host="swu-mysql.xxxxxxxxxx.ap-northeast-2.rds.amazonaws.com", #AWS RDS MySQL Server의 End Point
        user="admin", password="mzc2023!", database="STUDY", port=3306
    )
    curs = conn.cursor()
    uid = event["queryStringParameters"]["uid"]
    passwd = event["queryStringParameters"]["upasswd"]
    sql = "DELETE FROM TMEMBER WHERE fid = %s AND fpass = %s "
    result = curs.execute(sql, (uid, passwd))
    conn.commit()

    return {
        'statusCode': 200,
        'body': json.dumps(result)
    }
```

3) 테스트 이벤트의 이벤트 JSON

```
{
  "queryStringParameters": {
    "uid": "aaa2",
    "upasswd": "1111"
  }
}
```