Programming of RDS using Lambda

1. AWS RDS-MySQL 생성 및 MySQL Workbench로 작업하기

2. Python용 Pymysql 패키지 Layer 추가

3. mysql에 연결하기

4. SELECT문으로 데이터 가져오기

5. insert / update / delete문 사용하기

**1. AWS RDS-MySQL 생성 및 MySQL Workbench로 작업하기**

1) AWS RDS Service – MySQL 생성하기

① AWS RDS 🡺 데이터베이스 🡺 “데이터베이스 생성”버튼 클릭

단, 이때 주의 해야할 점은 간혹 리전이 변경되는 경우가 있으니 “서울”리전이 맞는지 확인 후 “데이터베이스 생성” 버튼을 클릭해야 함.

텍스트이(가) 표시된 사진

자동 생성된 설명

② 엔진 옵션, 템플릿, 가용성 및 내구성

|  |  |
| --- | --- |
|  |  |
| 엔진 옵션 | 템플릿과 가용성 및 내구성 |

③ 설정 및 인스턴스 구성

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |  |
| 설정 | 인스턴스 구성 |

④ 스토리지

텍스트이(가) 표시된 사진

자동 생성된 설명

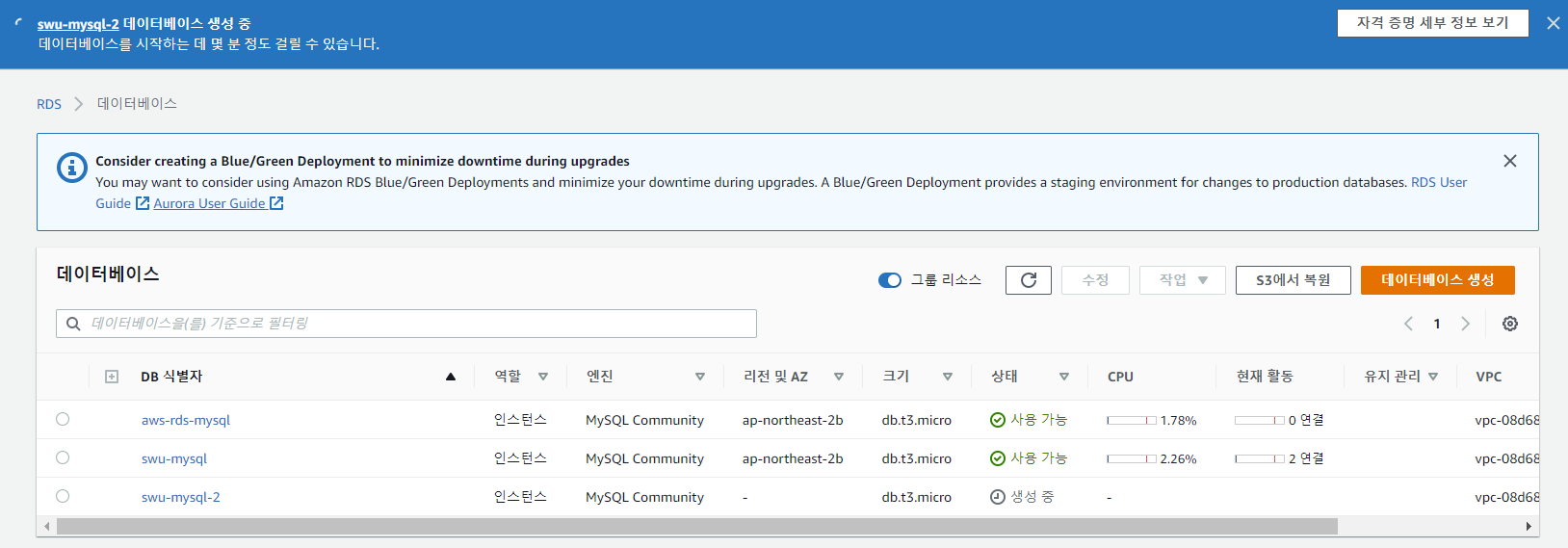
⑤ 연결

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |
| **[중요]** 퍼블릭 액세스 🡺 “예” 선택 | |

⑥ 기타 : 이후 변경할 사항은 없으며 “데이터베이스 생성” 버튼 클릭을 통해 마무리 한다.

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |  |
| 데이터베이스 인증과 모니터링 | 데이터베이스 생성 |

⑦ 생성 중 화면 : 아직 완교가 되지 않아 사용할 수 없음



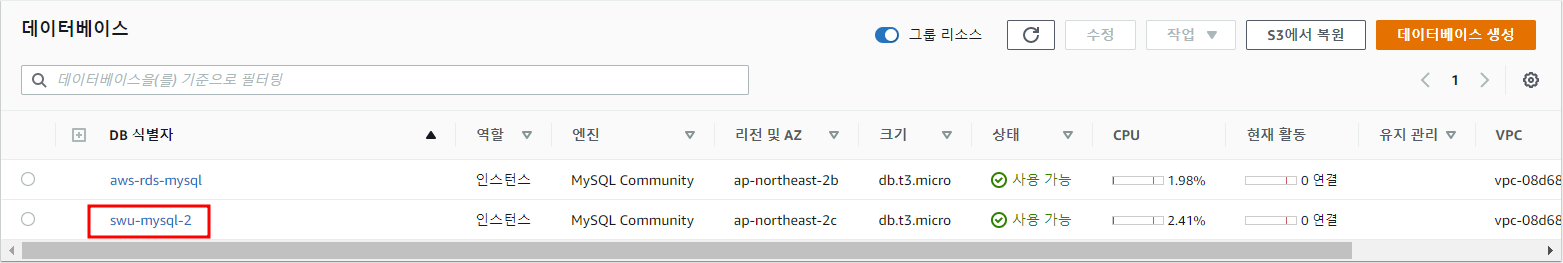
⑧ 생성 완료 상태 : 상태값이 “사용 가능”으로 표현되어야 이후부터 사용이 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

2) MySQL Workbench로 작업하기

① DB 식별자(이름) 클릭



② 데이터베이스 End Point 주소 복사

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

③ Workbench 실행 🡺 MySQL 연결 정보 등록

|  |  |
| --- | --- |
|  | 텍스트이(가) 표시된 사진  자동 생성된 설명 |
| MySQL 연결 정보 등록 | |

④ 데이터베이스 연결 설정

텍스트이(가) 표시된 사진

자동 생성된 설명

⑤ Test Connection 버튼을 통합 연결 확인 🡺 “OK”버튼 클릭

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |  |
|  |  |

**2. 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.

텍스트이(가) 표시된 사진

자동 생성된 설명

③ 파이썬용 pymysql 설치 및 다운로드.

|  |
| --- |
| C:\...\...\python>pip install pymysql |

④ 설치가 잘 되었늕 확인

|  |
| --- |
| C:\...\...\python>pip list |

⑤ 설치 및 다운로드를 통해 디렉토리(폴더)가 생성되었는지 확인

|  |
| --- |
| C:\...\...\python>dir/w |

텍스트이(가) 표시된 사진

자동 생성된 설명

**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** |
|  |

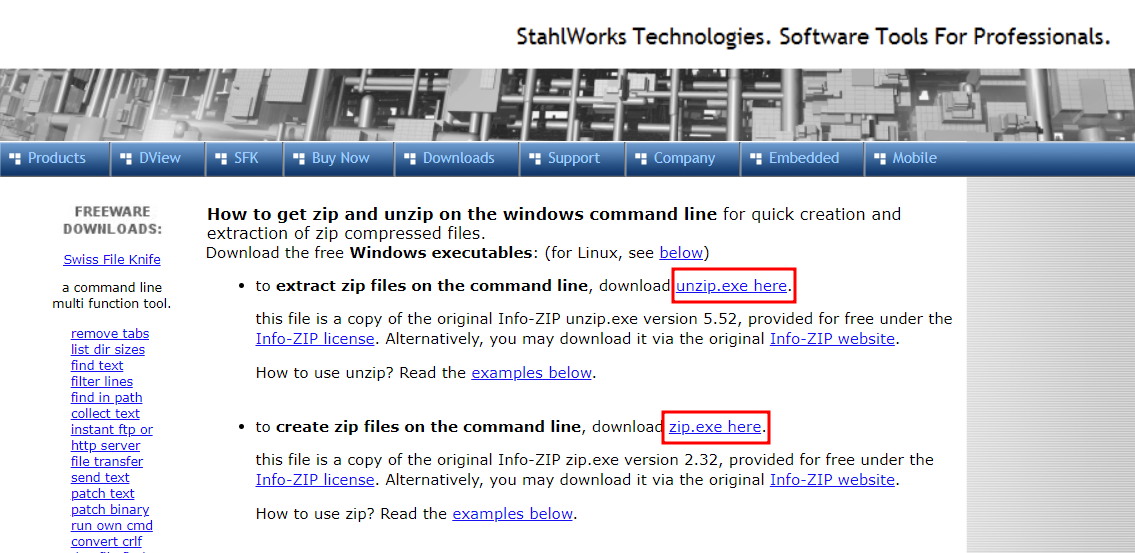
**① 압축 유틸리티를 사용하여 압축하는 방법**

|  |  |
| --- | --- |
| **🞘 압축을 잘못 한 예(알집사용)** | |
|  |  |
| Python 폴더를 압축한경우 | 압축파일의 내부구조에 python이 없음 |
| **🞘 옮바른 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 |



- 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.

텍스트이(가) 표시된 사진

자동 생성된 설명

- 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 |

텍스트이(가) 표시된 사진

자동 생성된 설명

**3) 계층 생성**

|  |
| --- |
| AWS 콘솔 로그인 🡺 AWS Lambda 🡺 추가 리소스 🡺 계층 🡺 계층 생성 |

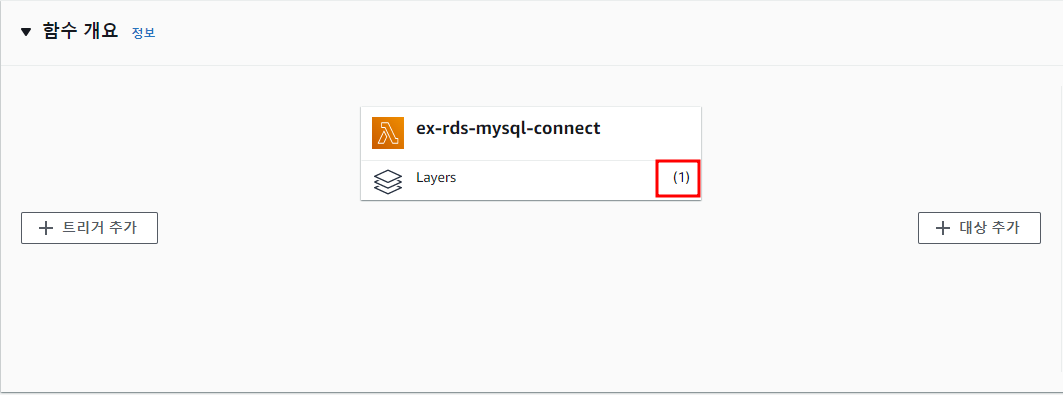


**4) Lambda 함수에 계층 추가**

|  |
| --- |
| AWS 콘솔 로그인 🡺 AWS Lambda 🡺 함수 🡺 함수 선택 🡺 페이지 하단 계층 🡺 [Add a layer] |

텍스트이(가) 표시된 사진

자동 생성된 설명



**3. pymysql 주요 객체 및 메서드**

🡺 깃허브에 업로드한 “PyMySQL-0.7\_Documentation.pdf”참고

**4. 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 옵현으로 default=str, indent=2 를 추가해주면 된다. |

**5. 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 |

**6. 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.xxxxxxxxx.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"  }  } |

**7. 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"  }  } |