

Lambda Layer

목표

- Lambda Layer란?
- Lambda Layer 생성
- Lambda Layer 적용

AWS Lambda Layer란?

- Python library import를 하고 싶다면?
 - PC : pip install ~~~
 - AWS는?
- pandas, selenium을 AWS Lambda에서 사용하고 싶다면?

Response

```
{
  "errorMessage": "Unable to import module 'lambda_function': No module named 'selenium'",
  "errorType": "Runtime.ImportModuleError",
  "requestId": "bb5579cf-ad85-4149-ac32-22e4d5bd6ba7",
  "stackTrace": []
}
```

emoji 라이브러리 실습

람다함수 생성 : sgu-계정-layer-test
런타임 : python 3.13
권한 : SafetRoleForUser-계정

```
import emoji

def lambda_handler(event, context):
    text = "AWS Lambda is awesome! :rocket:"
    result = emoji.emojize(text, language='alias')
    print(result)
    return {
        'statusCode': 200,
        'body': result
    }
```

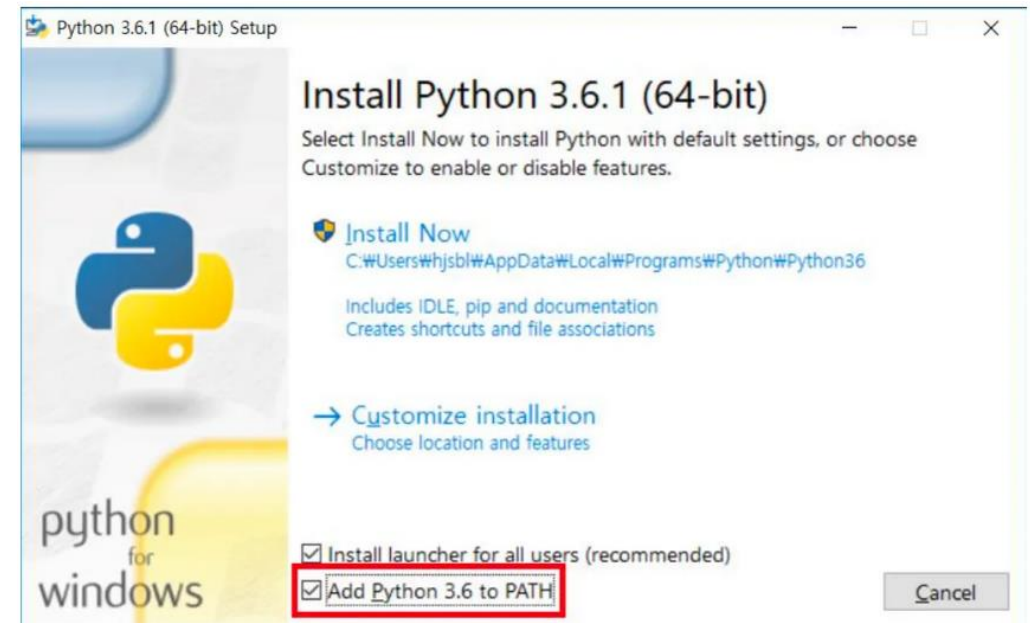
```
{
  "errorMessage": "Unable to import module 'lambda_function': No module named 'emoji'",
  "errorType": "Runtime.ImportModuleError",
  "requestId": "",
  "stackTrace": []
}
```

실습실PC) emoji 라이브러리 설치

- 디렉토리 만들기 예) D:\aws_3c\python
- 파이썬 버전 확인 : 3.13이 아닐 경우, 파이썬 3.13 설치!

```
D:\python\requests-layer>python -V  
Python 3.12.2
```

- 꼭 설치시 path 체크 하기!



실습실PC) emoji 라이브러리 설치

- CMD창에서 D:\aws_3c 가기!
- `py -3.13 -m pip install emoji -t python/`
명령어 실행

```
D:\aws_3c>py -3.13 -m pip install emoji -t python/
```

```
Collecting emoji
```

```
  Downloading emoji-2.14.1-py3-none-any.whl (100.0 kB)
```

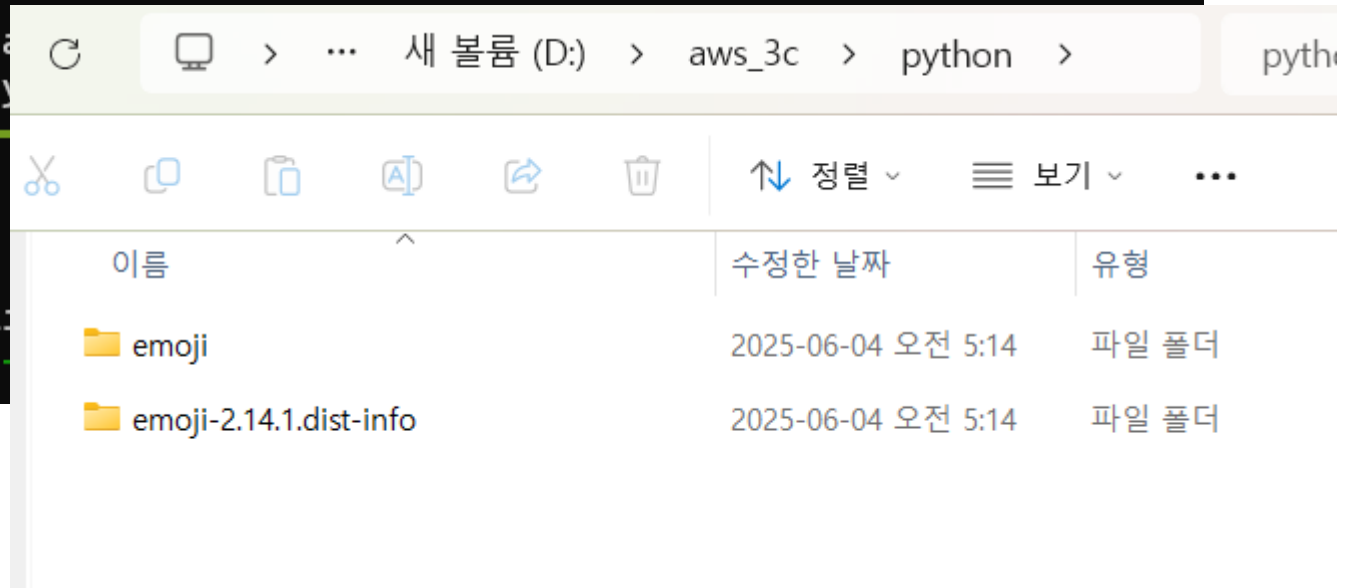
```
Downloading emoji-2.14.1-py3-none-any.whl (100.0 kB)
```

```
Installing collected packages: emoji
```

```
Successfully installed emoji-2.14.1
```

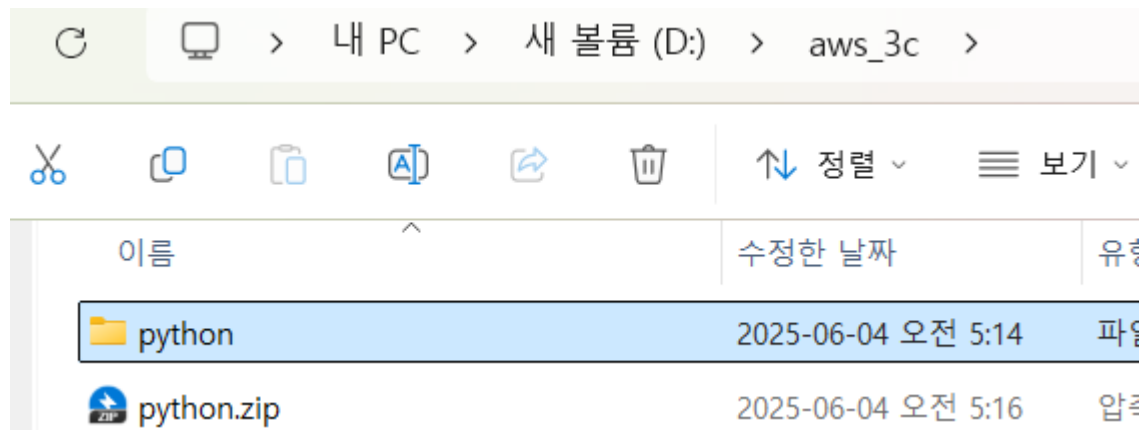
```
[notice] A new release of pip is available: 24.0 -> 24.3.1
```

```
[notice] To update, run: python.exe -m pip install --upgrade pip
```



실습실PC) emoji 라이브러리 설치

- python 디렉터리 zip으로 압축하기!



aws console

- 람다 → 추가리소스 → 계층 → 계층생성
- 이름 : sgu-계정-layer / 설명 : emoji install / .zip 파일 업로드 하기
- 호환런타임 : python 3.13--> 생성

Lambda
대시보드
애플리케이션
함수
▼ 추가 리소스
코드 서명 구성
이벤트 소스 매핑
계층
복제본

계층 구성

< 이름

sgu-202500-layer

호환 런타임 - 선택 사항 | 정보
최대 15개의 런타임을 선택합니다.

설명 - 선택 사항

emoji install

Python 3.13 X

라이선스 - 선택 사항 | 정보

☒ .zip 파일 업로드

☐ Amazon S3에서 파일

[↗ 파일 선택](#)


python.zip
616.81 KB

[취소](#) [생성](#)


레이어 생성 확인!

sgu-202500-layer

[삭제](#) [다운로드](#) [버전 생성](#)

버전 세부 정보			
버전 1	버전 ARN  arn:aws:lambda:ap-northeast-2:443370697536:layer:sgu-202500-layer:1	설명 emoji install	
생성함 29초 전	라이선스 -	호환 런타임 -	
호환 아키텍처 -			

[버전](#) | 이 버전을 사용하는 함수

모든 버전 (1)			마지막으로 가져온 항목 30초 전 
<input type="text" value="속성별 필터 또는 키워드별 검색"/>			
버전 ▼	버전 ARN	설명	
1	arn:aws:lambda:ap-northeast-2:443370697536:layer:sgu-202500-layer:1	emoji install	

람다 레이어 적용!



람다 레이어 적용!

사용자지정계층 → 생성한 레이어 --> 버전 1 --> 추가 클릭!

계층 선택

계층 소스

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

Choose a layer from a list of layers created by your AWS account.

사용자 지정 계층

Layers created by your AWS account that are compatible with your function's runtime.

sgu-202500-layer

람다 레이어 테스트!

PROBLEMS OUTPUT CODE REFERENCE LOG TERMINAL

Status: Succeeded

Test Event Name: hello-world

Response:

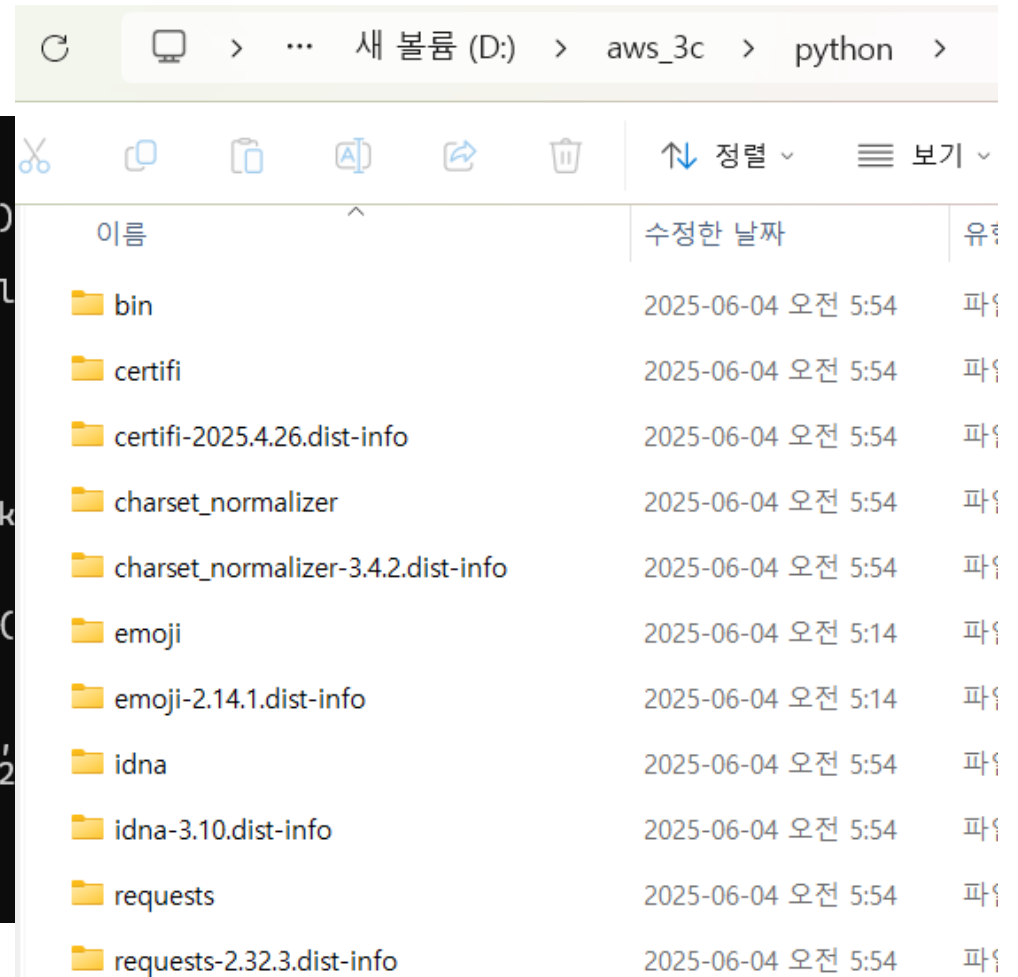
```
{  
  "statusCode": 200,  
  "body": "AWS Lambda is awesome! 🚀"  
}
```

라이브러리를 추가하고 싶은 경우?

- requests library를 실습실 PC에 설치 해 보자!
- 동일한 경로! → 다시 zip 생성

```
D:\aws_3c>py -3.13 -m pip install requests -t python/
Collecting requests
  Using cached requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Using cached charset_normalizer-3.4.2-cp313-cp313-win_amd64.whl
Collecting idna<4,>=2.5 (from requests)
  Using cached idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Using cached urllib3-2.4.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Using cached certifi-2025.4.26-py3-none-any.whl.metadata (2.5 kB)
Using cached requests-2.32.3-py3-none-any.whl (64 kB)
Using cached certifi-2025.4.26-py3-none-any.whl (159 kB)
Using cached charset_normalizer-3.4.2-cp313-cp313-win_amd64.whl (C
Using cached idna-3.10-py3-none-any.whl (70 kB)
Using cached urllib3-2.4.0-py3-none-any.whl (128 kB)
Installing collected packages: urllib3, idna, charset-normalizer,
Successfully installed certifi-2025.4.26 charset-normalizer-3.4.2

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```



이름	수정한 날짜	유형
bin	2025-06-04 오전 5:54	폴더
certifi	2025-06-04 오전 5:54	파일
certifi-2025.4.26.dist-info	2025-06-04 오전 5:54	파일
charset_normalizer	2025-06-04 오전 5:54	파일
charset_normalizer-3.4.2.dist-info	2025-06-04 오전 5:54	파일
emoji	2025-06-04 오전 5:14	파일
emoji-2.14.1.dist-info	2025-06-04 오전 5:14	파일
idna	2025-06-04 오전 5:54	파일
idna-3.10.dist-info	2025-06-04 오전 5:54	파일
requests	2025-06-04 오전 5:54	파일
requests-2.32.3.dist-info	2025-06-04 오전 5:54	파일

requests 라이브러리란?

- Python 용 HTTP 라이브러리
- 웹 서버로 HTTP GET, POST, PUT, DELETE 등의 다양한 요청을 보내고, 서버로부터 응답 데이터를 받을 수 있음
- 이번 실습에서는 <https://httpbin.org/post> 사용
- get 방식 : 쿼리스트링
- post 방식 : JSON데이터 Body에 담아 전송
- AWS API gateway는 주로 POST 방식
 - 왜? JSON Body로 데이터를 보내고 구조화된 데이터 전달에 용이 해서!

생성하였던 레이어 버전 생성

- 생성한 레이어 sgu-202500-layer에 버전을 생성!

sgu-202500-layer

삭제

다운로드

버전 생성

버전 세부 정보

버전

버전 ARN

설명

버전 생성

함수 런타임 설정

런타임
Python 3.13

아키텍처
x86_64

계층 선택

계층 소스

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
Choose a layer from a list of layers created by your AWS account.

사용자 지정 계층

Layers created by your AWS account that are compatible with your function's runtime.

sgu-202500-layer

버전

3

실습1) requests- get

requests라는 라이브러리를 사용해서 테스트용 API 사이트인 HTTPBin에 GET 요청을 보내보고 응답을 출력

```
import requests

def lambda_handler(event, context):
    url = "https://httpbin.org/get"
    response = requests.get(url)
    data = response.json()

    print("API 응답:", data)

    return {
        'statusCode': 200,
        'body': str(data)
    }
```

실습 2) request-post

requests 라이브러리를 사용해 POST 방식으로 HTTPBin에 요청 보냄

```
import requests
import json

def lambda_handler(event, context):
    url = "https://httpbin.org/post"
    payload = {"name": "test"}
    response = requests.post(url, json=payload)
    data = response.json()

    print("API 응답:", data)

    return {
        'statusCode': 200,
        'body': str(data)
    }
```

실습 2) request-post

- `payload = {"name": "test"} // POST 요청 Body에 담아 보낼 JSON 데이터`
- `response = requests.post(url, json=payload)`
 - `requests.post()`로 POST 요청 전송
 - `json=payload`
 - `payload`(딕셔너리)를 JSON 문자열로 자동 직렬화해서 전송
 - HTTP 요청 헤더에 `Content-Type: application/json`도 자동으로 붙여 줌
- `data = response.json() // 응답 객체(response)에서 JSON 형식의 응답 Body를 파싱해 Python 딕셔너리로 변환`