

SOMA 멘티교육자료

**AWS 클라우드 실습가이드 - 서버리스 DynamoDB**

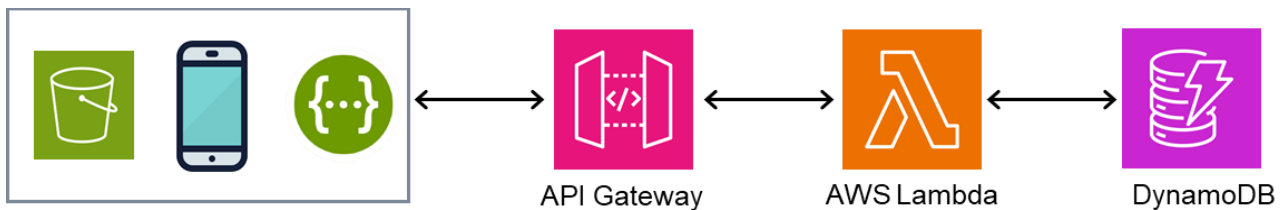
정철웅 멘토

[ [cwjung123@gmail.com](mailto:cwjung123@gmail.com) ]

## 서버리스 DynamoDB API 연동

전화번호부를 등록, 수정, 조회, 삭제하는 프로그램 ( Serverless 기반: Lambda + DynamoDB + API Gateway)

### ■ 목표시스템 구성



### ■ 사전준비 사항

- AWS 계정생성
- AWS IAM 사용자 및 접근키생성
- AWS CLI 설정

### ■ 설치도구

- NoSQL Workbench for DynamoDB  
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.html>

### ■ 활용사이트

- CRUD 예제  
<https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html>

- API Test  
<https://www.postman.com>  
<https://swagger.io>  
<https://resttesttest.com>  
Advanced REST client (<https://install.advancedrestclient.com/>)

### ■ 기타

- 영어 (English US)기반으로 화면캡처 진행 ( 일부 혼용 )
- Lambda 에 사용된 실소스 및 교육자료 GitHub 링크 강의중 제공
- 강의중 생성된 자료는 첨부 3. 삭제편 확인후 삭제가능

## [ 1 단계. DynamoDB 테이블 생성 ]

### [ 방법 1. 콘솔]

#### ▶ 콘솔 URL

<https://ap-northeast-2.console.aws.amazon.com/dynamodbv2/home?region=ap-northeast-2#tables>

#### ▶ 테이블생성 --> 테이블입력

테이블명 : phonebook, 파티션키 : id	항목추가 : DynamoDB > 항목탐색 > phonebook																
<p>DynamoDB &gt; Tables &gt; Create table</p> <p><b>Create table</b></p> <p><b>Table details</b> <small>info</small></p> <p>DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.</p> <p><b>Table name</b> This will be used to identify your table. phonebook <small>Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).</small></p> <p><b>Partition key</b> The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability. id String <small>1 to 255 characters and case sensitive.</small></p> <p><b>Sort key - optional</b> You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key. Enter the sort key name: String <small>1 to 255 characters and case sensitive.</small></p> <p>Cancel Create table</p>	<p>DynamoDB &gt; Items: phonebook &gt; Item editor</p> <p><b>Create item</b> <span>Form JSON</span></p> <p><b>Attributes</b> <span>Add new attribute ▼</span></p> <table border="1"> <thead> <tr> <th>Attribute name</th> <th>Value</th> <th>Type</th> <th></th> </tr> </thead> <tbody> <tr> <td>id - Partition key</td> <td>1005</td> <td>String</td> <td></td> </tr> <tr> <td>name</td> <td>Tommy Kim</td> <td>String</td> <td>Remove</td> </tr> <tr> <td>phonenum</td> <td>02050003100</td> <td>String</td> <td>Remove</td> </tr> </tbody> </table> <p>Cancel Create item</p>	Attribute name	Value	Type		id - Partition key	1005	String		name	Tommy Kim	String	Remove	phonenum	02050003100	String	Remove
Attribute name	Value	Type															
id - Partition key	1005	String															
name	Tommy Kim	String	Remove														
phonenum	02050003100	String	Remove														

### [ 방법 2. CLI ] : DynamoDB 이용 권한을 가진 IAM 키 적용된 상태에서 진행

- CLI 테스트 : `aws dynamodb list-tables` (테이블 목록출력)
- 테이블명: phonebook (파티션키)
- 속성 json 파일 작성 ( `create-table-phonebook.json` ) (파일위치 `~/apistart/reference/dynamo` )

```
{
  "TableName": "phonebook",
  "KeySchema": [
    { "AttributeName": "id", "KeyType": "HASH" }
  ],
  "AttributeDefinitions": [
    { "AttributeName": "id", "AttributeType": "S" }
  ],
  "BillingMode": "PAY_PER_REQUEST"
}
```

#### - 테이블 생성 CLI

`aws dynamodb create-table --cli-input-json file://create-table-phonebook.json`

#### - 테이블 삭제 CLI

`aws dynamodb delete-table --table-name phonebook`

- 데이터등록 1 (partiQL 실행)

**aws dynamodb batch-execute-statement --statements file://item\_partiql.json**

===== item\_partiql.json내용 =====

```
[
  {
    "Statement": "INSERT INTO phonebook VALUE {'id' : '1011', 'name' :
'오나라', 'phonenum' : '01010101111' }"
  },
  ----- 중간 생략 -----
  {
    "Statement": "INSERT INTO phonebook VALUE {'id' : '1015', 'name' :
'한나라', 'phonenum' : '01030301111' }"
  }
]
```




- 데이터등록2 (put item방식) --> 간편함

**aws dynamodb batch-write-item --request-items file://item\_list.json**

===== item\_list.json내용 =====

```
{
  "phonebook": [
    {
      "PutRequest": {
        "Item": { "id": {"S": "1031"}, "name": {"S": "강감찬"},
        "phonenum": {"S": "01022226565"}
      }
    },
    ----- 이하 생략 -----
  ]
}
```

- DynamoDB 콘솔에서 데이터 확인

Items returned (10)					Actions ▼	Create item
				< 1 >  		
<input type="checkbox"/>	id (String) ▼	name ▼	phonenum ▼			
<input type="checkbox"/>	<a href="#">1031</a>	강감찬	01022226565			
<input type="checkbox"/>	<a href="#">1012</a>	촉나라	01020201111			
<input type="checkbox"/>	<a href="#">1013</a>	한나라	01030301111			
<input type="checkbox"/>	<a href="#">1011</a>	오나라	01010101111			

## [ 2 단계-A. Lambda 함수생성 - 콘솔 ]

### ▶ 콘솔 URL

<https://ap-northeast-2.console.aws.amazon.com/lambda/home?region=ap-northeast-2#/functions>

### ▶ 함수생성

#### [ Node 선택]

함수명 : func\_phonebook\_node 런타임 : Node.js 16.x

Lambda > Functions > Create function

### Create function Info

Choose one of the following options to create your function.

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**  
Select a container image to deploy for your function.

---

#### Basic information

**Function name**  
Enter a name that describes the purpose of your function.

func\_phonebook\_node

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** Info  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 16.x

**Architecture** Info  
Choose the instruction set architecture you want for your function code.

☐ x86\_64

☒ arm64

#### [ Python선택]

함수명 : func\_phonebook\_python 런타임 : Python3.12

Lambda > Functions > Create function

### Create function Info

Choose one of the following options to create your function.

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**  
Select a container image to deploy for your function.

---

#### Basic information

**Function name**  
Enter a name that describes the purpose of your function.

func\_phonebook\_python

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** Info  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.12

**Architecture** Info  
Choose the instruction set architecture you want for your function code.

☐ x86\_64

☒ arm64

## ▶ 역할설정

기본 람다 역할을 선택해도 되나 기존에 생성한 역할로 대체해서 진행

▼ Change default execution role

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

simple\_apigateway\_lambda\_role

↻

[View the simple\\_apigateway\\_lambda\\_role role](#) on the IAM console.

## ▶ 함수생성 완료화면

Successfully created the function **func\_phonebook\_node**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > func\_phonebook\_node

func\_phonebook\_node

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

func\_phonebook\_node

Layers (0)

+ Add trigger + Add destination

Description -

Last modified 6 seconds ago

Function ARN [arn:aws:lambda:ap-northeast-2:569251118950:function:func\\_phonebook\\_node](#)

Function URL [Info](#)

Successfully created the function **func\_phonebook\_python**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > func\_phonebook\_python

func\_phonebook\_python

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

func\_phonebook\_python

Layers (0)

+ Add trigger + Add destination

Description -

Last modified 2 minutes ago

Function ARN [arn:aws:lambda:ap-northeast-2:569251118950:function:func\\_phonebook\\_python](#)

Function URL [Info](#)

소스가 입력되지 않은 상태이므로 소스를 편집하고 트리거를 설정한 이후에 함수가 동작함.

(각 언어 타입에 맞는 초기구동파일과 코드가 출력됨 : 아래는 파이썬 )

----- 아 래 -----

```
import json
def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

-----

5 / 25

## [ 2 단계-B. Lambda 함수생성 - CLI ]

### 1. NodeJS 20.X (SDK v3) 기준

- 소스경로로 이동 ( ex : ~/apistart/reference/lambda/phonebook\_node/phonebook2\_clean )

- 모든파일과 디렉토리 압축

```
zip myfunction.zip -r *
```

- Lambda CLI 동작

소스압축본 myfunction.zip(명칭임의) 생성된 것 확인후에 짚애

함수명 : func\_phonebook2\_node

```
aws lambda create-function --function-name func_phonebook2_node --zip-file
fileb://myfunction.zip --handler index.handler --architectures arm64 --runtime
nodejs20.x --role arn:aws:iam::111122223333:role/simple_apigateway_lambda_role
```

### 2. Python V3.12

- 소스경로로 이동 ( ex : ~/apistart/reference/lambda/phonebook\_python)

```
cd ~/apistart/reference/lambda/phonebook_python
```

- 모든파일과 디렉토리 압축

```
zip myfunction.zip -r *
```

- Lambda CLI 동작 ( myfunction.zip 생성된 것 확인후에 ) 함수명 : phonebook2\_python\_function

```
aws lambda create-function --function-name func_phonebook2_python --zip-file
fileb://myfunction.zip --handler lambda_function.lambda_handler --architectures
arm64 --runtime python3.12 --role
arn:aws:iam::111122223333:role/simple_apigateway_lambda_role
```

※ 함수 내용변경

```
aws lambda update-function-code --function-name functionName --zip-file
fileb://depoymntFile.zip
```

※ 함수 역할변경

```
aws lambda update-function-configuration --function-name functionName --role
arn:aws:iam::111122223333:role/roleName
```

## [ 3 단계. API 생성 ]

===== A-S1. 생성 =====

순서 : API-Gateway생성 --> 경로(route)생성 --> 통합(Integration)생성 --> 통합붙이기 --> CORS설정

### 유형선택

HTTP-API 는 비용과 성능측면에서 유리

- HTTP-APIs (effective REST APIs) : \$1/1M
- REST APIs : \$3.5/1M
- HTTP-APIs are 14~16% faster than REST APIs

API생성 (명칭 : phonebook\_api / 통합은 초기에 추가 혹은 나중에 변경가능)

API Gateway > APIs > Create API > Create

Step 1  
Create an API

Step 2 - optional  
[Configure routes](#)

Step 3 - optional  
[Define stages](#)

Step 4  
[Review and Create](#)

### Create an API

#### Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations (0) [Info](#)

[Add integration](#)

API name  
An HTTP API must have a name. This name is cosmetic and does not have to be unique; you will use the API's ID (generated later) to programmatically refer to this API.

phonebook\_api

Cancel [Review and Create](#) [Next](#)

(Next선택)

### 경로구성

API Gateway > APIs > Create API > Create

Step 1  
[Create an API](#)

Step 2 - optional  
**Configure routes**

Step 3 - optional  
[Define stages](#)

Step 4  
[Review and Create](#)

### Configure routes - optional

#### Configure routes [Info](#)

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path	Integration target
<a href="#">Add route</a>		

Cancel [Review and Create](#) [Previous](#) [Next](#)

(Next선택)



## 스태이지 정의

[API Gateway](#) > [APIs](#) > [Create API](#) > Create

Step 1  
[Create an API](#)

Step 2 - optional  
[Configure routes](#)

Step 3 - optional  
**Define stages**

Step 4  
Review and Create

### Define stages - optional

**Configure stages** [Info](#)

Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named \$default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name  
\$default

Auto-deploy  
☒

Remove

Add stage

CancelPreviousNext

(Next선택)

Review and Create

CancelPreviousCreate

[ 생성완료됨! ]

☑ Successfully created API phonebook\_api (knlbzg30m7).

[API Gateway](#) > [APIs](#) > Routes - phonebook\_api (knlbzg30m7)

### Routes

Stage: - Deploy

Choose a route.

Routes for phonebook\_api Create

Q Search

## 경로구성

- GET /items : 전체가져오기

### Create a route

**Route and method** [Info](#)

Route name eg. /pets  
Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

GET ▼

/items

Cancel

Create

- GET /items/{id} : 단항목가져오기

### Create a route

**Route and method** [Info](#)

Route name eg. /pets  
Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

GET ▼

/items/{id}

Cancel

Create

- PUT /items

### Create a route

**Route and method** [Info](#)

Route name eg. /pets  
Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

PUT ▼

/items

Cancel

Create

- POST /items

### Create a route

**Route and method** [Info](#)

Route name eg. /pets  
Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

POST ▼

/items

Cancel

Create

- DELETE /items/{id}

## Create a route

**Route and method** [Info](#)

Route name eg. /pets  
Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes.

DELETE

최종경로설정이 완료된 화면

[API Gateway](#) > [APIs](#) > Routes - phonebook\_api (knlbzg30m7)

## Routes

Stage: -

Choose a route.

**Routes for phonebook\_api**

- ▼ /items
  - GET
  - PUT
  - POST
- ▼ /{id}
  - DELETE
  - GET

## 통합생성

API Gateway

APIs

Custom domain names

VPC links

API: phonebook\_api...  
(knlbzg30m7)

▼ Develop

Routes

Authorization

Integrations

CORS

Integration target

Integration type

Lambda function

Integration details

Integration target

Choose the Lambda function that API Gateway invokes when the route receives a request.

AWS Region

ap-northeast-2

Lambda function

arn:aws:lambda:ap-northeast-2:569251118950:function:func\_phonebook2\_node

▼ Advanced settings

통합은 복수개가 생성되어 각 경로별로 붙을 수도 있음

경로별로 통합을 붙임

API Gateway > APIs > phonebook\_api (knlbzg30m7) > Integrations

Integrations

Stage: - Deploy

Attach integrations to routes Manage integrations

Routes for phonebook\_api

Search

▼ /items

GET

PUT

POST

▼ /{id}

DELETE

GET

Integration details for route

GET /items (ID: cdxrhx7)

No integration is attached to this route. You can either attach an existing integration, or create and attach a new one.

Create and attach an integration

- or -

Choose an existing integration

Search

func\_phonebook2\_node (mmhm292)

Attach integration

[최종통합 완료화면]

Attach integrations to routes Manage integrations

Routes for phonebook\_api

Search

▼ /items

GET AWS Lambda

PUT AWS Lambda

POST AWS Lambda

▼ /{id}

DELETE AWS Lambda

GET AWS Lambda

Integration details for route

GET /items/{id} (gg513d4)

Detach integration Manage integration

Lambda function

func\_phonebook2\_node (ap-northeast-2)

Integration ID

mmhm292

Description

-

Payload format version

The parsing algorithm for the payload sent to and returned from your Lambda function. [Learn more.](#)

2.0 (interpreted response format)

Invoke permissions

The resource policy of the Lambda function determines if API Gateway can invoke it. You can run the AWS CLI command snippet below to give API Gateway permission to invoke your AWS Lambda function.

▶ Example policy statement

## CORS설정

아래와 3값에 대한 세부처리 혹은 \* 입력처리

Access-Control-Allow-Origin / Access-Control-Allow-Methods / Access-Control-Allow-Headers

### Cross-Origin Resource Sharing

#### Configure CORS [Info](#)

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our [CORS documentation](#) for more details.

Access-Control-Allow-Origin

Add

\* X

Access-Control-Allow-Headers

Add

\* X

Access-Control-Allow-Methods

Choose Allowed Methods

▼

\* X

Access-Control-Expose-Headers

Add

Access-Control-Max-Age

Access-Control-Allow-Credentials

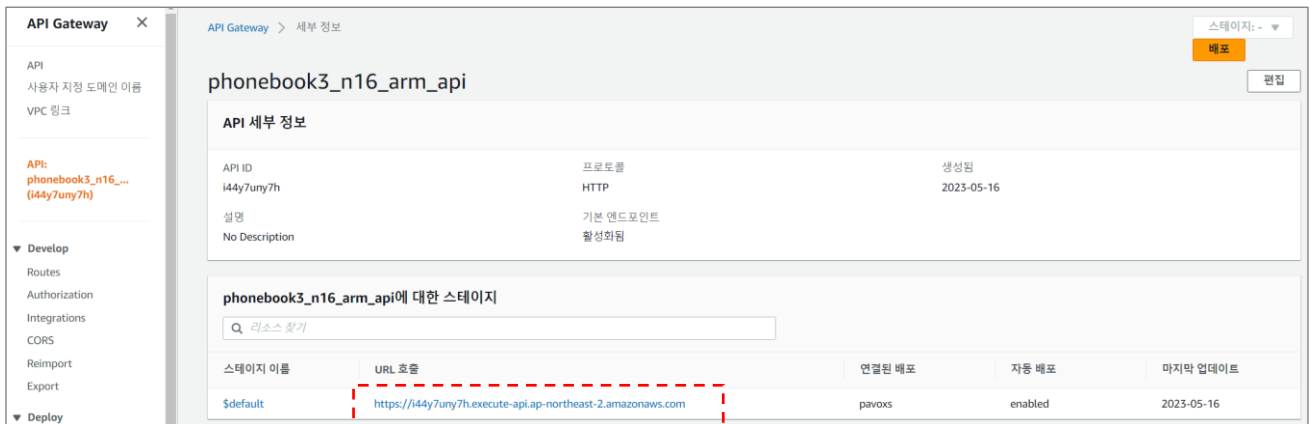
☐ NO

Cancel

Save

## [ 호출 테스트 ]

### ===== F1. API EndPoint 확인 =====



API 세부 정보		
API ID: i44y7uny7h	프로토콜: HTTP	생성됨: 2023-05-16
설명: No Description	기본 엔드포인트: 기본 엔드포인트	활성화됨

스테이지 이름	URL 호출	연결된 배포	자동 배포	마지막 업데이트
\$default	https://i44y7uny7h.execute-api.ap-northeast-2.amazonaws.com	pavoxs	enabled	2023-05-16

### 붉은색 박스 URL

<https://dl1s0lzvyh.execute-api.ap-northeast-2.amazonaws.com>

### ===== F2. 호출방법선택 =====

방법1) EC2 혹은 노트북에서 curl로 진행

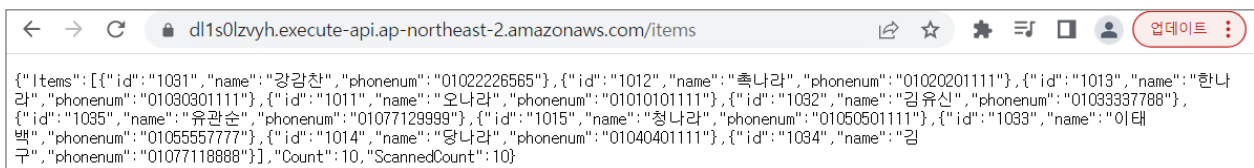
방법2) 웹호출 <https://reqbin.com/curl> (웹브라우저 secret mode)

방법3) Postman 등 API호출 도구사용

### ===== F3. API-호출테스트 진행 =====

#### ■ 전체목록 호출 GET /items

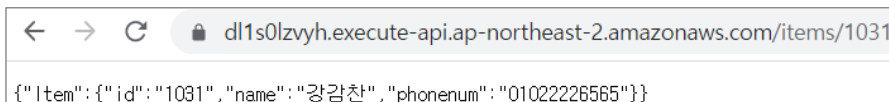
<https://dl1s0lzvyh.execute-api.ap-northeast-2.amazonaws.com/items>



```
{
  "Items": [
    {
      "id": "1031",
      "name": "강감찬",
      "phonenum": "01022226565"
    },
    {
      "id": "1012",
      "name": "속나라",
      "phonenum": "01020201111"
    },
    {
      "id": "1013",
      "name": "한나라",
      "phonenum": "01030301111"
    },
    {
      "id": "1011",
      "name": "오나라",
      "phonenum": "01010101111"
    },
    {
      "id": "1032",
      "name": "김유신",
      "phonenum": "01033337788"
    },
    {
      "id": "1035",
      "name": "유관순",
      "phonenum": "01077129999"
    },
    {
      "id": "1015",
      "name": "청나라",
      "phonenum": "01050501111"
    },
    {
      "id": "1033",
      "name": "이태백",
      "phonenum": "01055557777"
    },
    {
      "id": "1014",
      "name": "당나라",
      "phonenum": "01040401111"
    },
    {
      "id": "1034",
      "name": "김구",
      "phonenum": "01077118888"
    }
  ],
  "Count": 10,
  "ScannedCount": 10
}
```

#### ■ 단일호출 GET /items/{id}

<https://dl1s0lzvyh.execute-api.ap-northeast-2.amazonaws.com/items/1031>



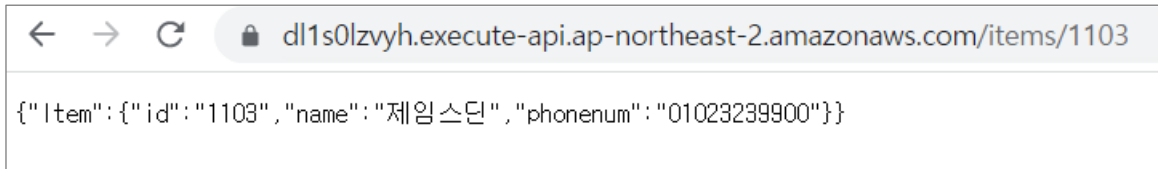
```
{
  "Item": {
    "id": "1031",
    "name": "강감찬",
    "phonenum": "01022226565"
  }
}
```

### ■ 데이터등록 PUT /items

```
curl -X PUT https://d11s0lzyyh.execute-api.ap-northeast-2.amazonaws.com/items -H "Content-Type: application/json" -d '{ "id": "1103", "name": "제임스딘", "phonenum": "01023239900" }'
```

(출력) "Put item 1104"

### 등록확인 (GET)



### ■ 데이터삭제 DELETE /items/{id}

```
curl -X DELETE https://d11s0lzyyh.execute-api.ap-northeast-2.amazonaws.com/items/1103
```

(출력) "Delete item 1103"

[ 오류메시지 ]

Lambda내부 로직오류

```
{"message": "Internal Server Error"}
```

--> CloudWatch에서 로그를 통해서 오류 내용을 확인후 조치 (권한, 문법오류, 연동오류 등)

CORS 오류 : 브라우저 개발자콘솔에서 확인

Access to fetch at 'https://i44y7uny7h.execute-api.ap-northeast-2.amazonaws.com/items' from origin 'http://app.myservice.com' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.



## [ 정적페이지에서 API-Gateway 호출 ]

### 버킷생성

- 방식 1 : 콘솔에서 생성 (버킷명을 실습예제에서는 s3://계정번호 12 자리숫자-front-phonebook )
- 방식 2 : CLI 로 생성

```
aws s3 mb s3://111122223333-front-phonebook
```

### 정적페이지 및 권한설정

- 정적웹으로 설정
- Public 허용 및 정책붙임

----- 정책내용 -----

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3PublicPart",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:*"],
      "Resource": [
        "arn:aws:s3:::111122223333-front-phonebook",
        "arn:aws:s3:::111122223333-front-phonebook/*"
      ]
    }
  ]
}
```

### 정적페이지호출 ( using JavaScript in S3 )

~/apiclass : 디렉토리를 S3로 올려서 정적웹주소로 확인

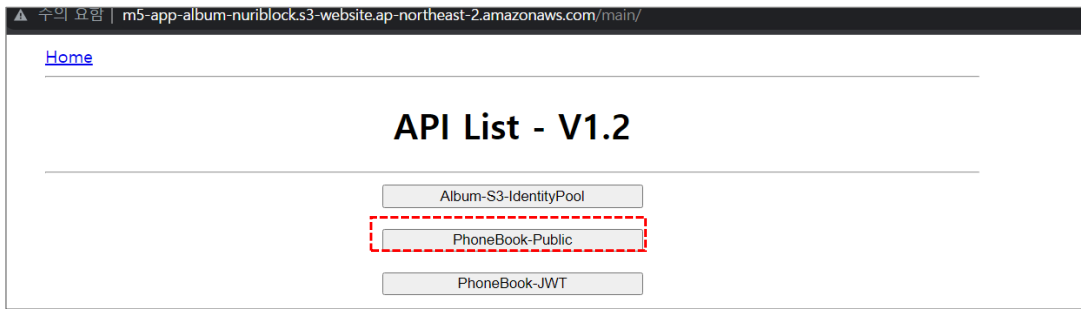
\$ vi ./apistart/main/phonebook/main.js : 아래의 밑줄 API엔드포인트(호출주소) 변경

```
const apiInvokeBaseURL = "https://i1vfadfady2.execute-api.ap-northeast-2.amazonaws.com";
```

(모두 업로드)

\$ cd ./apistart

```
aws s3 cp --recursive . s3://111122223333-front-phonebook --recursive --exclude
".git/*" --exclude "reference/*" --exclude ".github/*" --exclude ".gitignore"
```



GET /items

전화번호부 목록		
<a href="#">추가</a>		
ID	Name	Phone
1003	미켈란젤로	01022990088
1011	오나라	01010101111
1012	축나라	01020201111
1013	한나라	01030301111

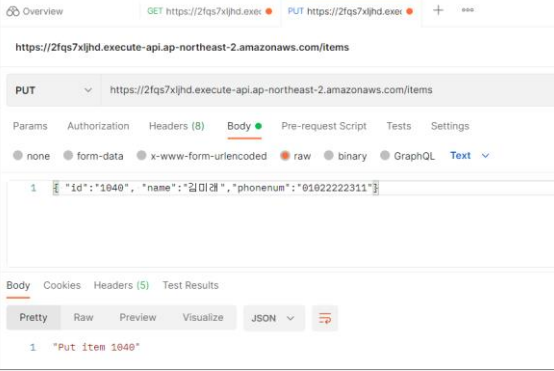
PUT /items	GET /item/{id}
<p><b>데이터 수정</b></p> <p>ID : <input type="text" value="1032"/></p> <p>Name : <input type="text" value="김유신"/></p> <p>Phone : <input type="text" value="01033337788"/></p> <p><a href="#">확인</a></p>	<p><b>전화번호부 Information</b></p> <p>Id : 1032</p> <p>Name : 김유신</p> <p>Phone : 01033337788</p> <p><a href="#">수정</a> <a href="#">삭제</a></p>

※ API Test

- 도구 : <https://resttesttest.com> / postman 기타 curl 명령전송가능 shell

(1) PUT 데이터입력

명령	출력
<pre>curl -v -X PUT https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items -H "Content-Type: application/json" -d '{"id":"1004", "name":"James Dean", "phonenum":"01022228888"}'</pre> <p>※ 유의사항 : 경로확인 유의 ( 예제에서는 /items ) ( Invoke URL + API Path )</p>	<pre>"Put item 1004"</pre>

<p><b>Run Curl Commands Online</b></p> <p>Execute Curl commands directly from your browser. Learn Curl with live Curl examples. Test APIs with ReqBin Online</p> <p>File Generate Code Tools Share Generate</p> <p>Curl Raw US Run Status: 200 (OK)</p> <pre>curl -v -X PUT https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items -H "Content-Type: application/json" -d '{ "id": "1004", "name": "James Dean", "phonenum": "01022228888" }'</pre> <p>Content (1) Header</p> <p>"Put item 1004"</p>	
(테스트 - 비회원 ) <a href="https://reqbin.com/curl">https://reqbin.com/curl</a>	(테스트 - 포스트맨 다운로드 혹은 회원가입 ) <a href="https://www.postman.com/">https://www.postman.com/</a>

## (2) GET 데이터 단항목 가져오기

명령	출력
<pre>curl -v https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items/1005</pre> <p>※ Path : /items/{id}</p>	<pre>{   "Item": {     "id": "1005",     "name": "Tommy Kim",     "phonenum": "02050003100"   } }</pre>

## (3) GET 데이터 전체 가져오기

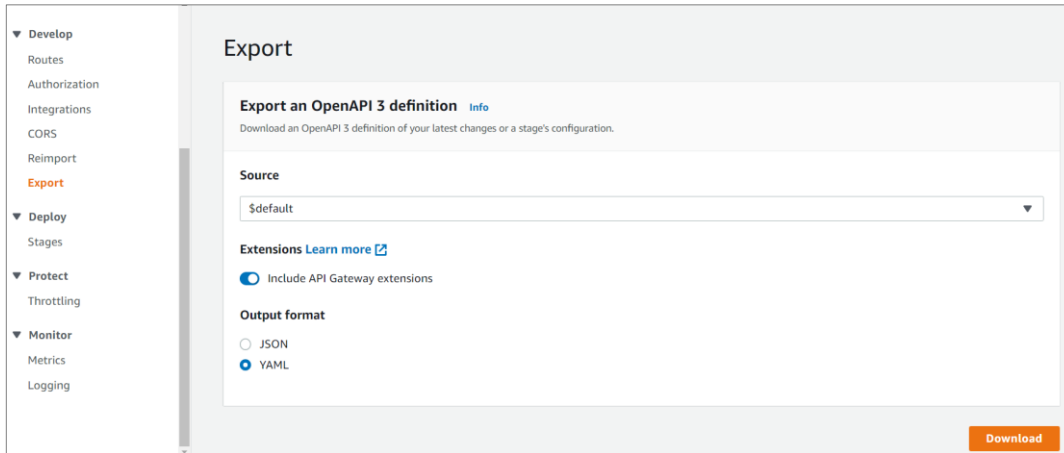
명령	출력
<pre>curl https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items</pre> <p>※ Path : /items</p>	<pre>{   "Items": [     {       "id": "1004",       "name": "James Dean",       "phonenum": "01022228888"     },     {       "id": "1020",       "name": "임격정",       "phonenum": "01020003000"     },     {       "id": "1040",       "name": "홍삼정",       "phonenum": "01020004000"     },     {       "id": "1000",       "name": "홍길동",       "phonenum": "01010001122"     },     {       "id": "1005",       "name": "Tommy Kim",       "phonenum": "02050003100"     }   ],   "Count": 5,   "ScannedCount": 5 }</pre>

## (4) DELETE 데이터삭제

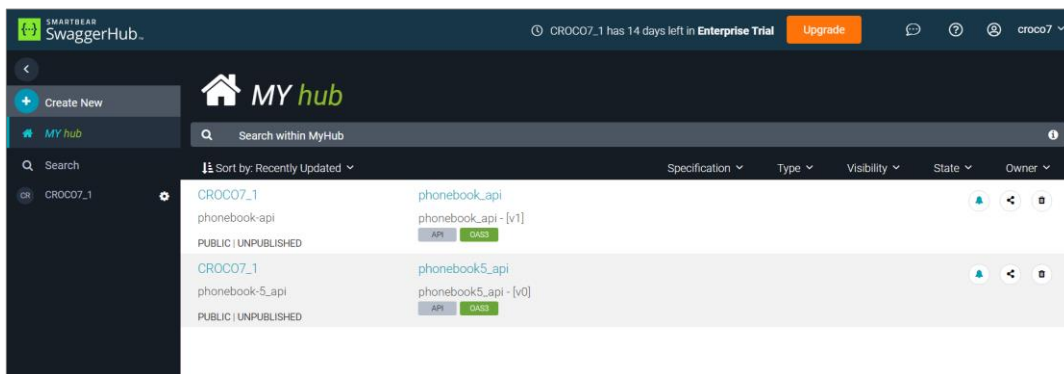
명령	출력
<pre>curl -X DELETE curl -v https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items/1004</pre> <p>※ Path : /items/{id}</p>	<p>"Deleted item 1004"</p>

## [ 첨부 1. OpenAPI 3 - SWAGGER 내보내기 ]

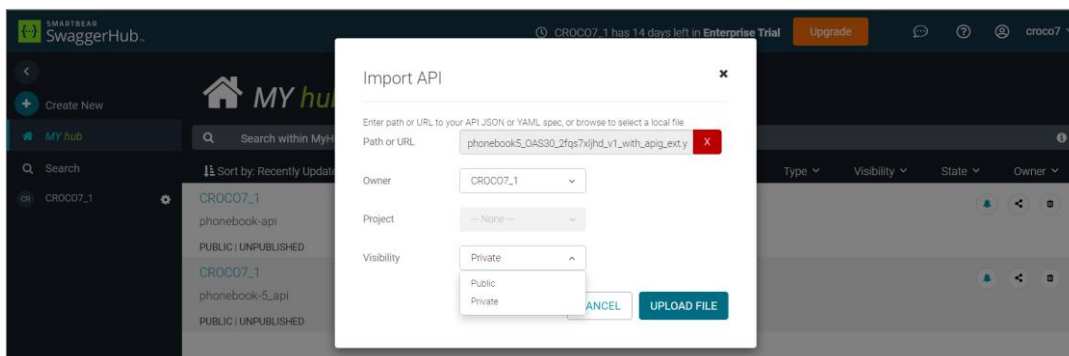
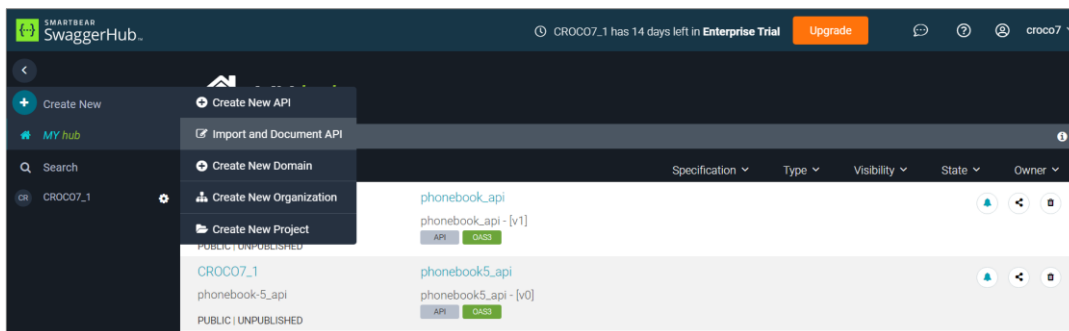
- swagger.io 서비스 이용하거나 swagger 구현
- API 내보내기 ( AWS > API-Gateway )



- API 가져오기 ( SWAGGER )



--> Import and Document API



## - 실행

The image shows the SwaggerHub web interface for an API named 'phonebook-api' at version 'v1'. The interface is divided into several sections:

- Info:** Displays the API title 'phonebook-api', version 'v1', and the base URL 'https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/v1'. It also shows the API is valid.
- Tags:** A list of tags for the API.
- Servers:** A list of servers for the API.
- default:** A list of API endpoints with their methods and paths:
  - GET /items/{id}
  - DELETE /items/{id}
  - GET /items
  - PUT /items
  - POST /items
- Schemas:** A list of schemas for the API.

The right-hand side of the interface shows the details for the selected endpoint, 'GET /items'. It includes a 'Parameters' section with 'No parameters' and an 'Execute' button. Below the 'Execute' button, there are tabs for 'cURL (bash)', 'cURL (PowerShell)', and 'cURL (CMD)'. The 'cURL (bash)' tab is selected, showing the command: `curl -X 'GET' 'https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items/{id}'`.

## [ 첨부 2. CLI 활용 정책 및 역할추가 ]

API-Gateway + Lambda + DynamoDB 사용 Role 작성

( cd ~/apistart/reference/iam\_ref 에 관련파일 모두 있음 )

최종작성 RoleName : simple\_apigateway\_lambda\_role

### 1) 정책추가

- 로그쓰기 정책 : posvc\_log\_create\_record

(1) Log 기록	( 2 ) Lambda 실행
<b>policy_svc_log_create_record.json</b>	<b>policy_svc_lambda_invoke.json</b>
<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "logs:CreateLogGroup",         "logs:CreateLogStream",         "logs:PutLogEvents"       ],       "Resource": "*"     }   ] }</pre>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": "lambda:InvokeFunction",       "Resource": "*"     }   ] }</pre>
(3) 다이나모 DB	API-Gateway 실행
<b>policy_svc_dynamodb_rw.json</b>	<b>policy_svc_apigateway_invoke.json</b>
<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "dynamodb:Query",         "dynamodb:DeleteItem",         "dynamodb:GetItem",         "dynamodb:PutItem",         "dynamodb:Scan",         "dynamodb:UpdateItem"       ],       "Resource": "*"     }   ] }</pre>	<pre>{   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": [         "execute-api:Invoke",         "execute-api:ManageConnections"       ],       "Resource": "arn:aws:execute-api:*:*:*"     }   ] }</pre>

※ 비고) 위의 **policy\_svc\_apigateway\_invoke.json** 내용은 AmazonAPIGatewayInvokeFullAccess 권한과 동일함.

(정책생성 CLI)

aws iam create-policy --policy-name my-policy --policy-document file://policy

위의 정책 생성 파일을 모두 CLI 명령어로 실행

aws iam create-policy --policy-name svc\_log\_create\_record --policy-document file://policy\_svc\_log\_create\_record.json

aws iam create-policy --policy-name svc\_lambda\_invoke --policy-document file://policy\_svc\_lambda\_invoke.json

aws iam create-policy --policy-name svc\_dynamodb\_rw --policy-document file://policy\_svc\_dynamodb\_rw.json

aws iam create-policy --policy-name svc\_apigateway\_invoke --policy-document file://policy\_svc\_apigateway\_invoke.json

[ 실행결과 ]

```
{
  "Policy": {
    "PolicyName": "svc_apigateway_invoke",
    "PolicyId": "ANPA2JY4KZEVMSHIX4NHM",
    "Arn": "arn:aws:iam::708192160042:policy/svc_apigateway_invoke",
    "Path": "/",

```

```

        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2022-07-13T01:54:26+00:00",
        "UpdateDate": "2022-07-13T01:54:26+00:00"
    }
}

```

## 2) 역할(Role)생성 ( 신뢰관계추가 + 정책붙이기 )

- 신뢰관계 정의파일 : **trust\_apigateway\_lambda.json**

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "apigateway.amazonaws.com",
          "lambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- 신뢰관계 설정 - Role 생성시 진행

( CLI 명령어 )

```
aws iam create-role --role-name example-role --assume-role-policy-document file://example-role-trust-policy.json
```

(실행)

```
aws iam create-role --role-name simple_apigateway_lambda_role --assume-role-policy-document file://trust_apigateway_lambda.json
```

[ 실행결과 ]

```

{
  "Role": {
    "Path": "/",
    "RoleName": "simple_apigateway_lambda_role",
    "RoleId": "AR0A2JY4KZEVDPLPV0C5P",
    "Arn": "arn:aws:iam::708192160042:role/simple_apigateway_lambda_role",
    "CreateDate": "2022-07-13T02:08:20+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": [
              "apigateway.amazonaws.com",
              "lambda.amazonaws.com"
            ]
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}

```

- 정책추가

( 정책추가 CLI 명령어 )

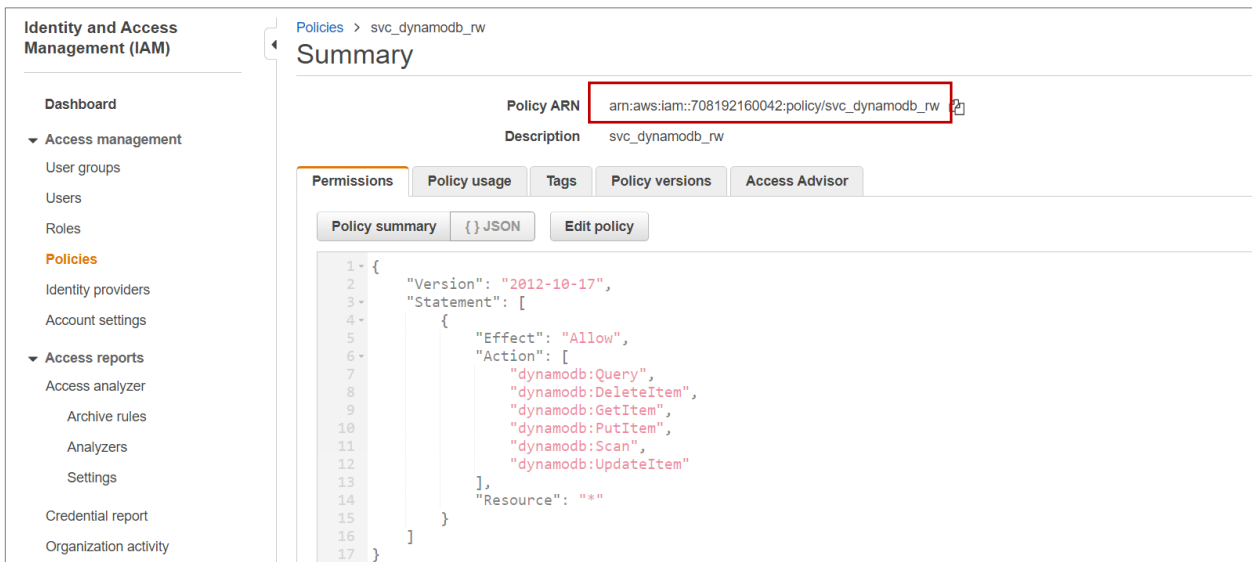
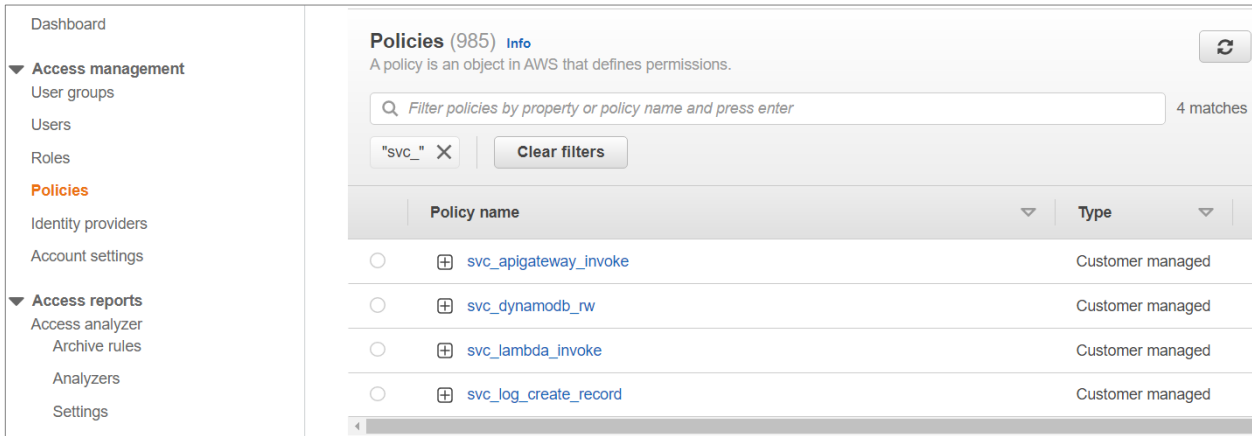
```
aws iam attach-role-policy --role-name example-role --policy-arn "arn:aws:iam::aws:policy/AmazonRDSReadOnlyAccess"
```

( 정책확인 CLI 명령어)

```
aws iam list-attached-role-policies --role-name example-role
```

- 정책추가 진행

Console > IAM > Policies > 필터링 "svc\_" : 정책의 ARN 을 확인할 수 있음.



(명령어 생성 ) 111122223333 : AWS 계정번호에 해당 (각자 변경)

```
aws iam attach-role-policy --role-name simple_apigateway_lambda_role --policy-arn arn:aws:iam::111122223333:policy/svc_apigateway_invoke
```

```
aws iam attach-role-policy --role-name simple_apigateway_lambda_role --policy-arn arn:aws:iam::111122223333:policy/svc_dynamodb_rw
```

```
aws iam attach-role-policy --role-name simple_apigateway_lambda_role --policy-arn arn:aws:iam::111122223333:policy/svc_lambda_invoke
```

```
aws iam attach-role-policy --role-name simple_apigateway_lambda_role --policy-arn arn:aws:iam::111122223333:policy/svc_log_create_record
```

- 최종적으로 추가된 역할(Role) **simple\_apigateway\_lambda\_role** 을 확인함 .

권한(Permissions)와 신뢰관계(Trust relationships) 확인



Role 의 ARN 은 자주 사용됨 - CLI 템플릿 및 명령어

여제 ) `arn:aws:iam::708192160042:role/simple_apigateway_lambda_role`

The screenshot shows the AWS IAM console interface for a role named 'simple\_apigateway\_lambda\_role'. The left sidebar contains navigation links for 'Access management' (User groups, Users, Roles, Policies, Identity providers, Account settings) and 'Access reports' (Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, Service control policies (SCPs)). The main content area shows the role's summary, including its creation date (July 13, 2022, 11:08 UTC+09:00), last activity (None), and ARN (arn:aws:iam::708192160042:role/simple\_apigateway\_lambda\_role). Below the summary, there are tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is active, showing a list of four attached policies: 'svc\_apigateway\_invoke', 'svc\_dynamodb\_rw', 'svc\_lambda\_invoke', and 'svc\_log\_create\_record'. Each policy is marked as 'Customer managed'.

Policy name	Type	Description
svc_apigateway_invoke	Customer managed	
svc_dynamodb_rw	Customer managed	svc_dynamodb_rw
svc_lambda_invoke	Customer managed	Invoke any lambda function
svc_log_create_record	Customer managed	

[ Lambda 실행기본 역할 : Authorizer 등에서 사용 ]

#### 1) 역할생성

```
aws iam create-role --role-name simple_lambda_role --assume-role-policy-document  
file://trust_lambda.json
```

#### 2) 정책붙이기

```
aws iam attach-role-policy --role-name simple_lambda_role --policy-arn  
arn:aws:iam::111122223333:policy/svc_lambda_invoke  
  
aws iam attach-role-policy --role-name simple_lambda_role --policy-arn  
arn:aws:iam::111122223333:policy/svc_log_create_record
```

### [ 첨부 3. 사용자원 삭제 ]

- lambda 삭제

[node]

```
aws lambda delete-function --function-name func_phonebook_node
```

```
aws lambda delete-function --function-name func_phonebook2_node
```

[python]

```
aws lambda delete-function --function-name func_phonebook_python
```

```
aws lambda delete-function --function-name func_phonebook2_python
```

- s3 삭제

```
aws s3 rm s3://111122223333-front-phonebook --recursive
```

```
aws s3 rb s3://111122223333-front-phonebook
```

- dynamodb 삭제

```
aws dynamodb delete-table --table-name phonebook
```

- apigateway 삭제 (콘솔에서 api-id 확인후 진행)

```
aws apigatewayv2 delete-api --api-id abcd111e
```

- 참조소스 삭제

```
rm -rf ~/apistart
```

- 끝 -