SOMA 멘티교육자료

AWS 클라우드 실습가이드 - 서버리스 DynamoDB

정철웅 멘토

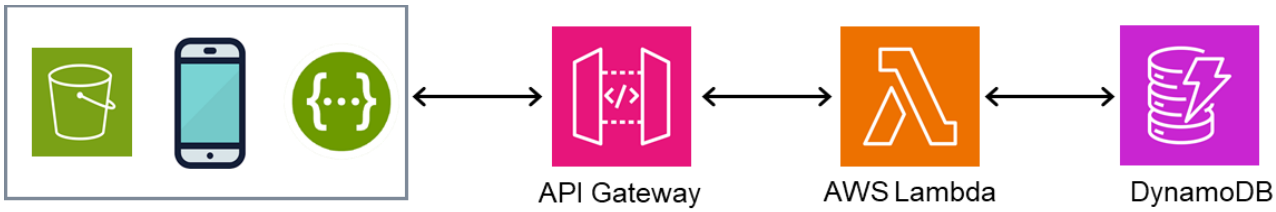
[cwjung123@gmail.com]

Version 2.8

서버리스 DynamoDB API 연동

전화번호부를 등록, 수정, 조회, 삭제하는 프로그램 (Servelss 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 화면캡쳐 진행 (일부 혼용)
- Lambda 에 사용된 실소스 및 교육자료 GitHub 링크 강의중 제공
- 강의중 생성된 자원은 첨부 3. 삭제편 확인후 삭제가능

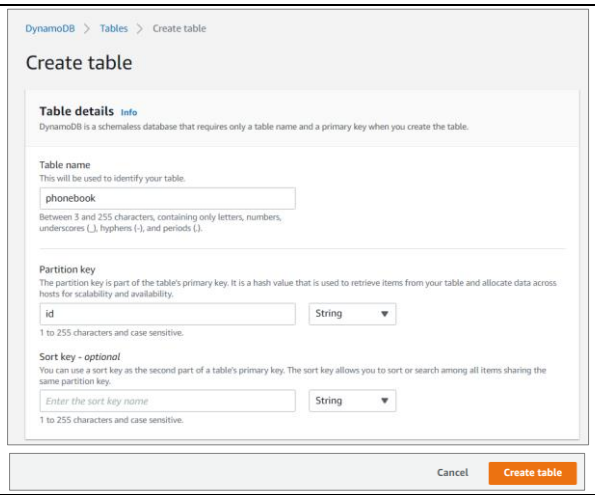
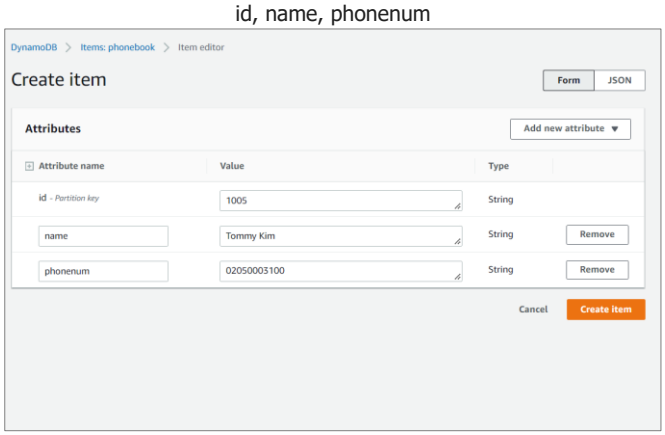
[1단계. DynamoDB 테이블 생성]

[방법 1. 콘솔]

▶ 콘솔 URL

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

▶ 테이블생성 --> 테이터입력

테이블명 : phonebook, 파티션키 : id	항목추가 : DynamoDB > 항목탐색 > phonebook id, name, phonenum
	

[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

비고. window는 인코딩 ansi로 되어야 함

aws dynamodb batch-execute-statement --statements file://item_partiql_ansi.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

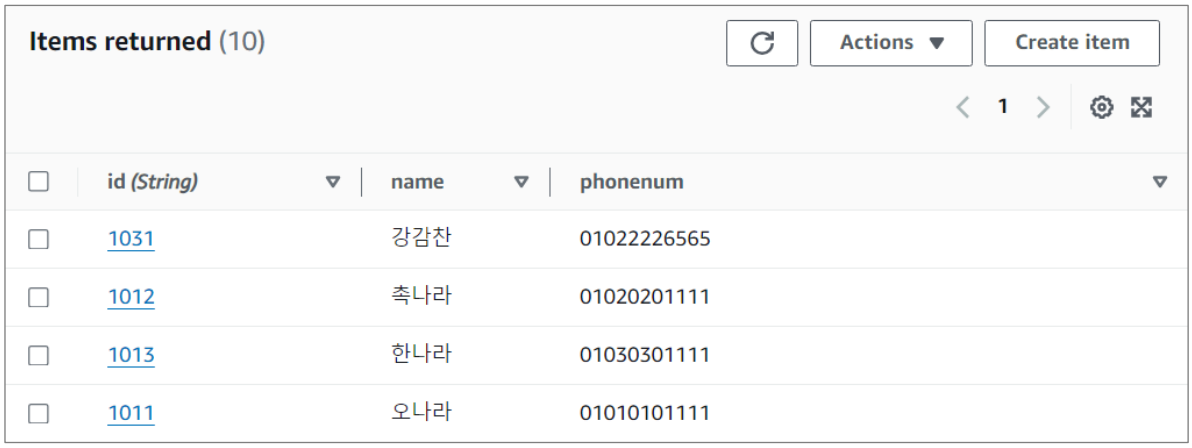
aws dynamodb batch-write-item --request-items file://item_list.json

비고. window는 인코딩 ansi로 되어야 함

aws dynamodb batch-write-item --request-items file://item_list_ansi.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
id (String)
name
phonenum
1031
강감찬
01022226565
1012
촉나라
01020201111
1013
한나라
01030301111
1011
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]

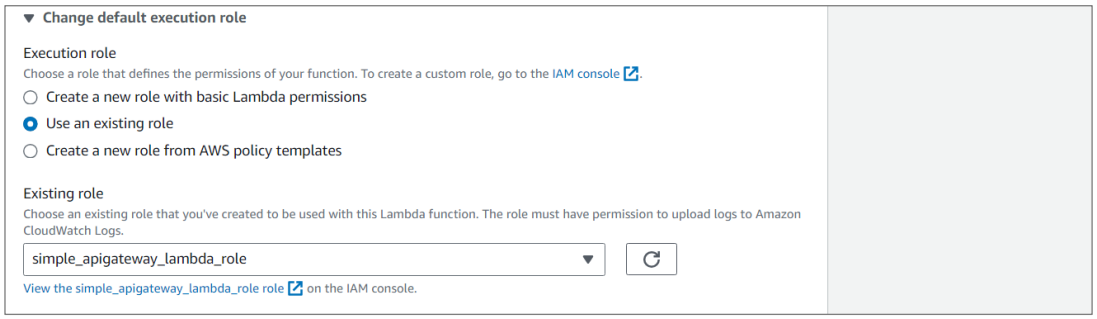
함수명 : func_phonebook_node 런타임 : Node.js 18.x ~

[Python선택]

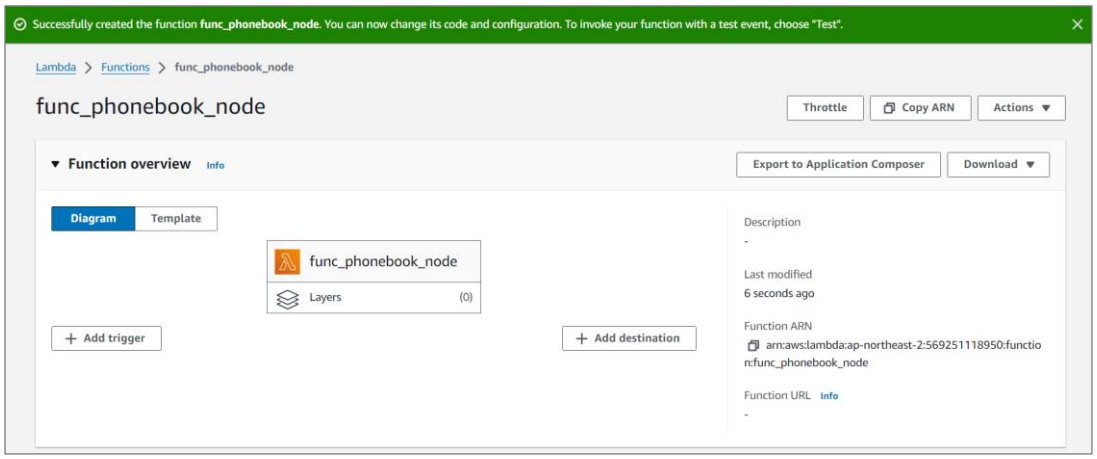
함수명 : func_phonebook_python 런타임 : Pythn3.13

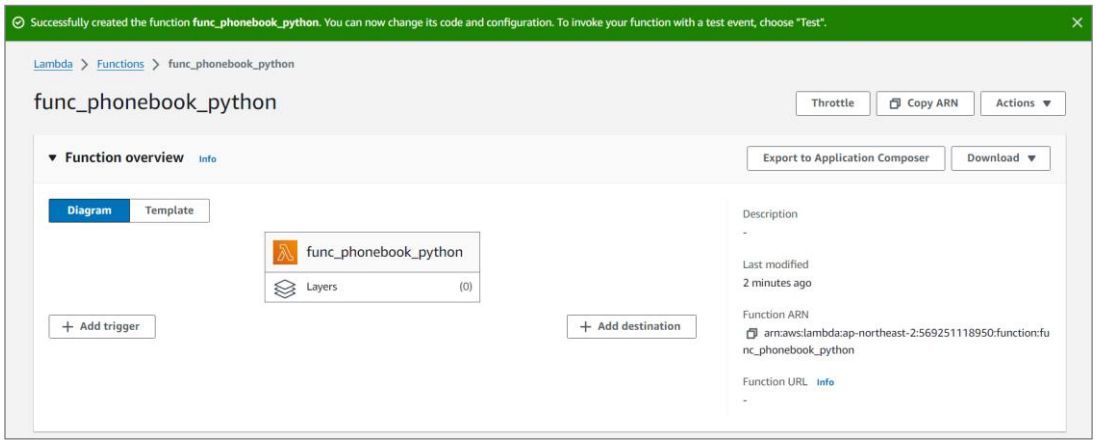
▶ 역할설정

기본 람다 역할을 선택해도 되나 기존에 생성한 역할로 대체해서 진행



▶ 함수생성 완료화면





소스가 입력지지 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!')
    }
```

[2 단계-B. Lambda 함수생성 - CLI]

1. NodeJS 22.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 : phonebook2_node

```
aws lambda create-function --function-name phonebook2_node_function --zip-file
 fileb://myfunction.zip --handler index.handler --architectures arm64  --runtime
 nodejs22.x --role arn:aws:iam::111122223333:role/simple_apigateway_lambda_role
```

2. Python V3.13

- 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  phonebook2_python_function --zip-file
 fileb://myfunction.zip --handler lambda_function.lambda_handler  --architectures
 arm64 --runtime python3.13 --role
 arn:aws:iam::111122223333:role/simple_apigateway_lambda_role
```

※ 함수 내용변경

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

※ 함수 역할변경

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

[3 단계. API 생성]

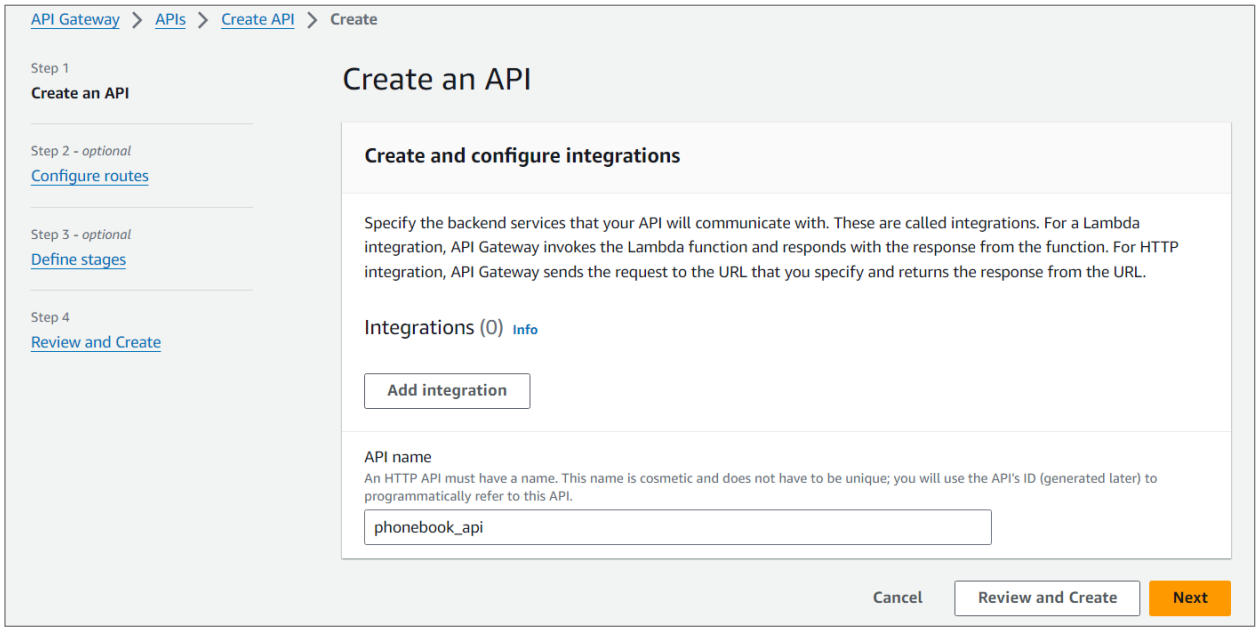
========== A-S1. 생성 ==========

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

유형선택

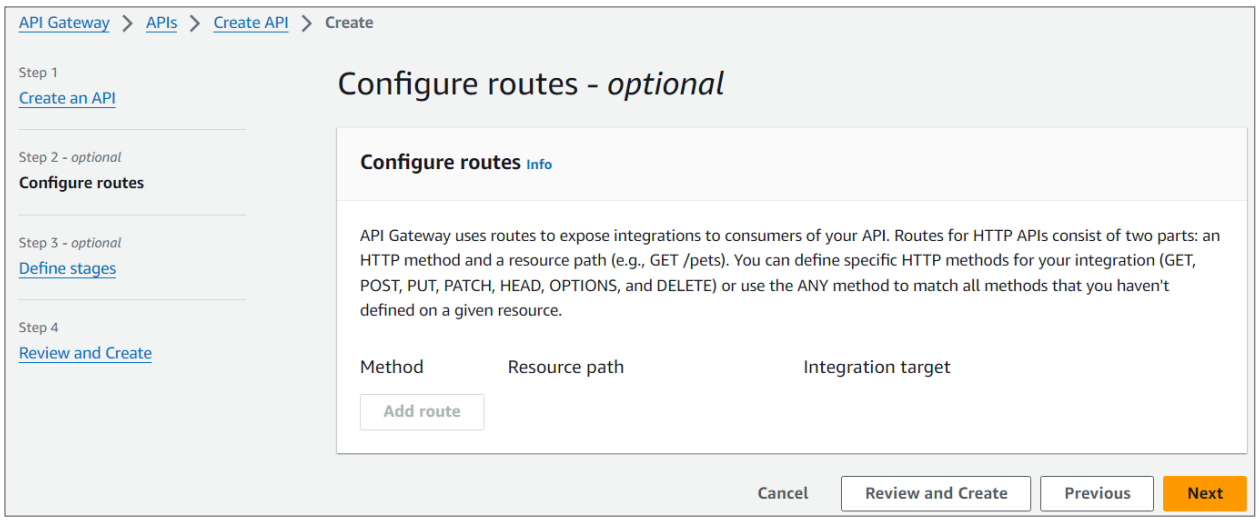
HTTP-API 는 비용과 성능측면에서 유리
- HTTP-APIs (effective REST APIs) : $1/1M
- REST APIs : $3.5/1M
- HTTP-APIs are 14~16% faster than REST APIs

API생성 (명칭 : phonebook_api / 통합은 초기에 추가 혹은 나중에 변경가능)



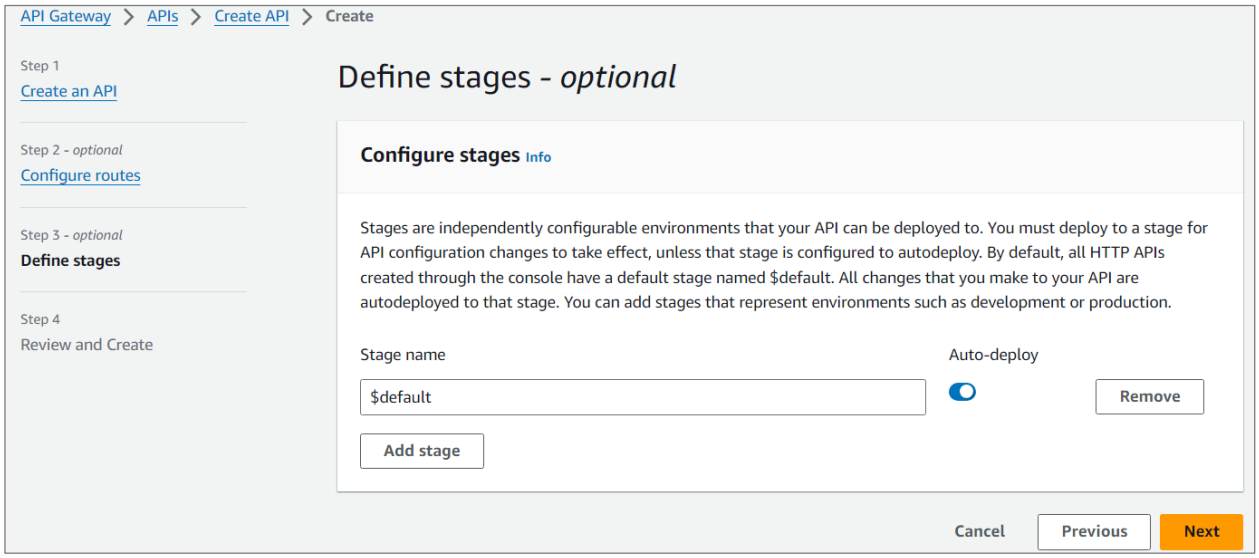
(Next선택)

경로구성

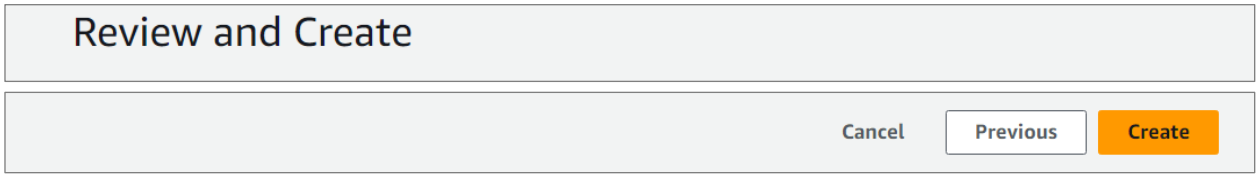


(Next선택)

스테이지 정의



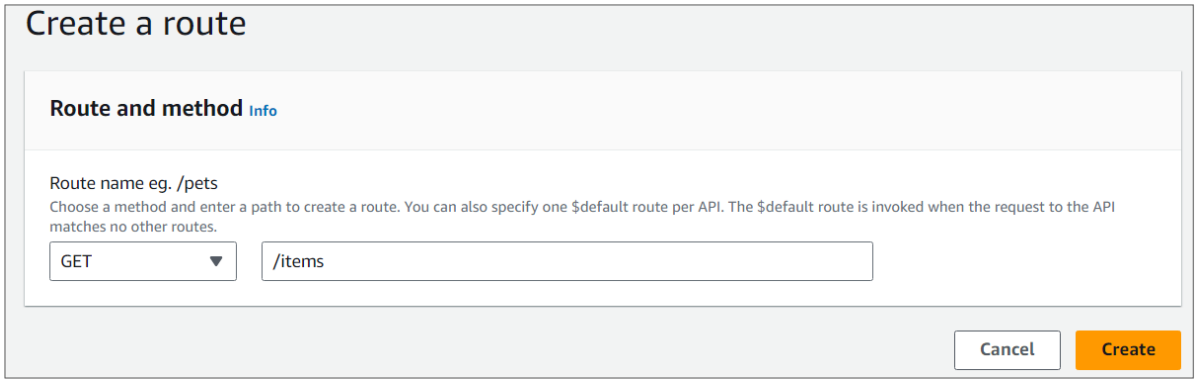
(Next선택)



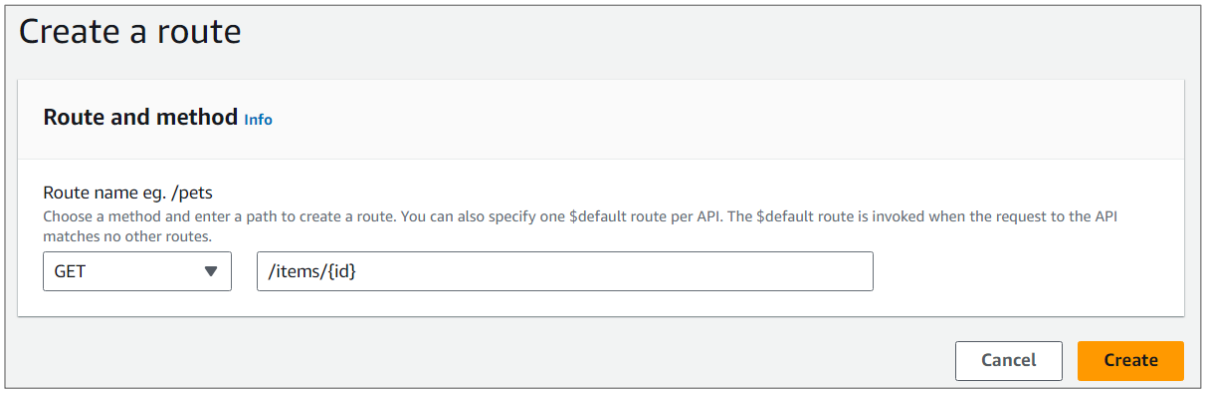
[생성완료됨!]

경로구성

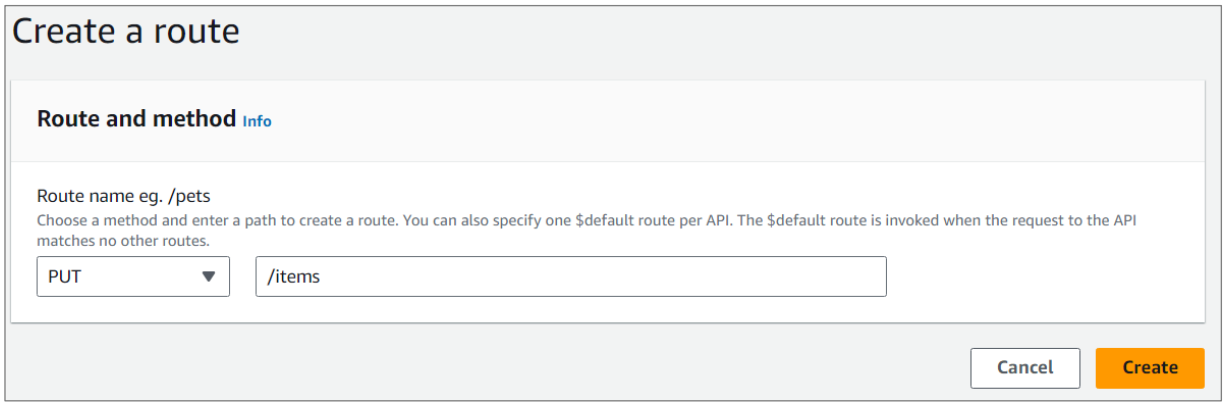
- GET /items : 전체가져오기



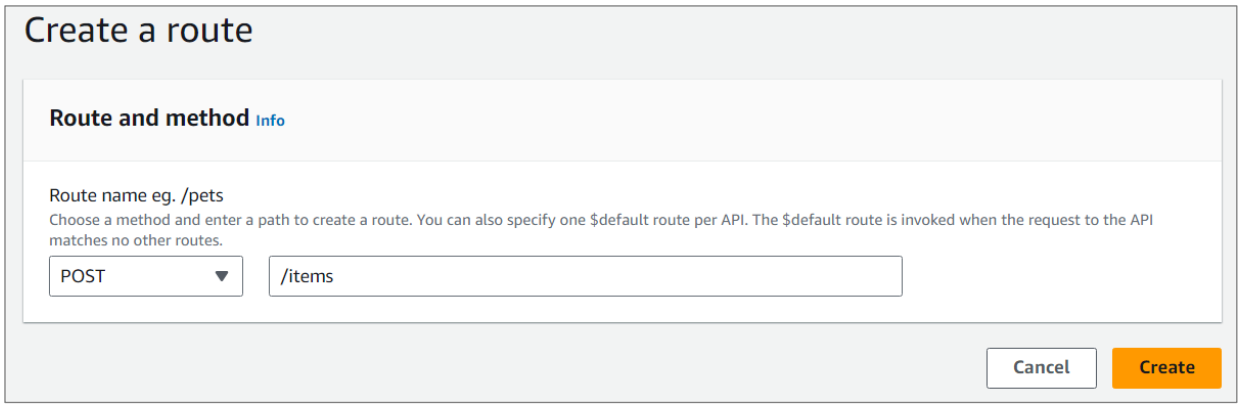
- GET /items/{id} : 단항목가져오기



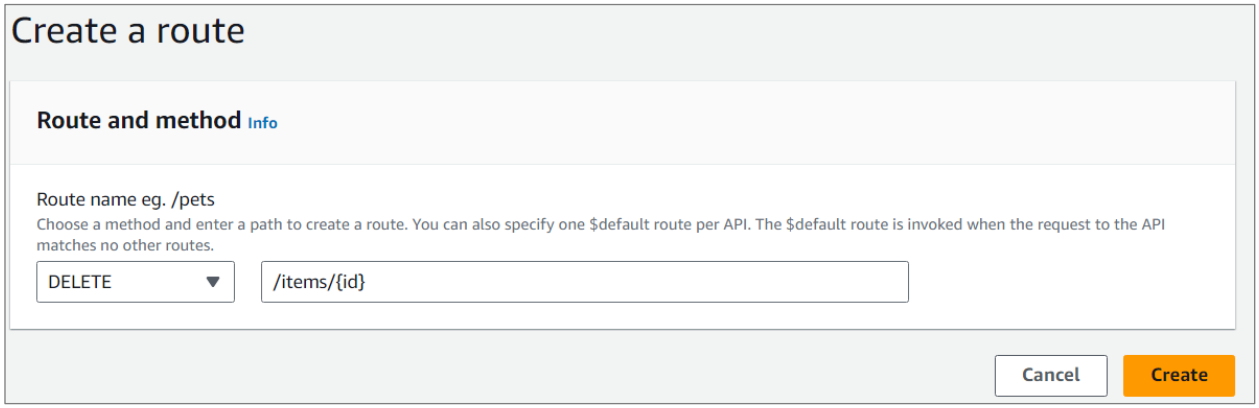
- PUT /items



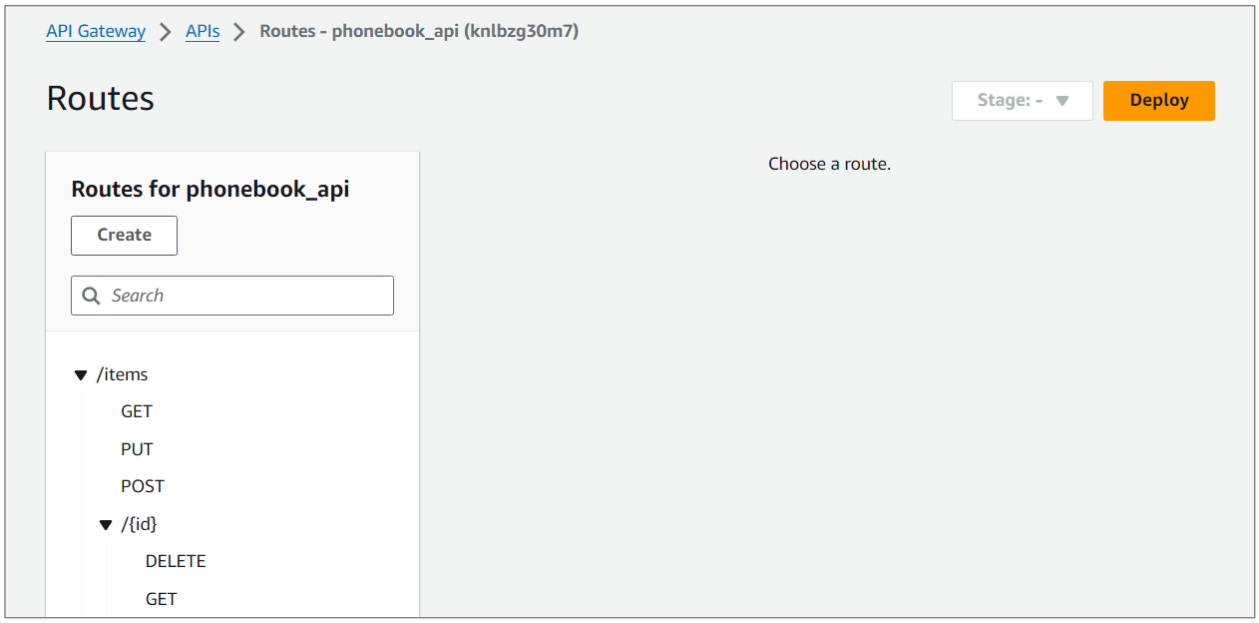
- POST /items



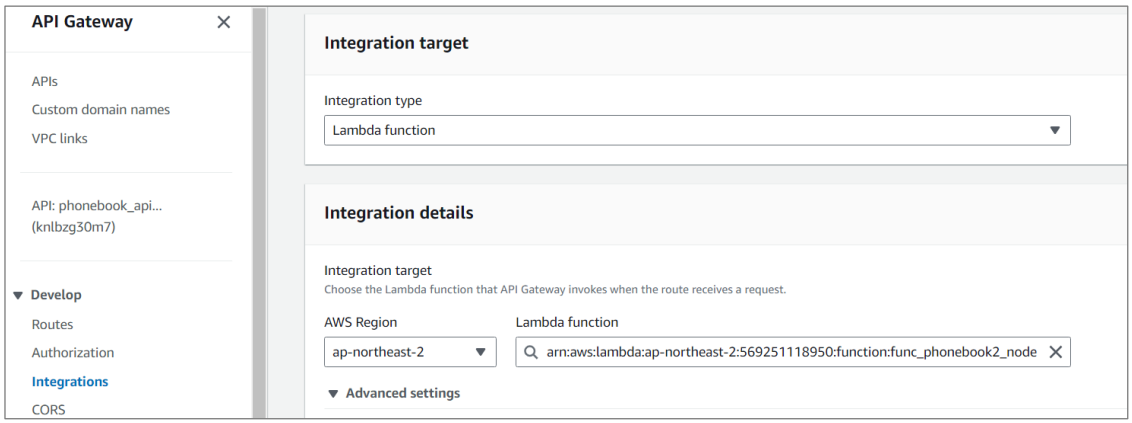
- DELETE /items/{id}



최종경로설정이 완료된 화면

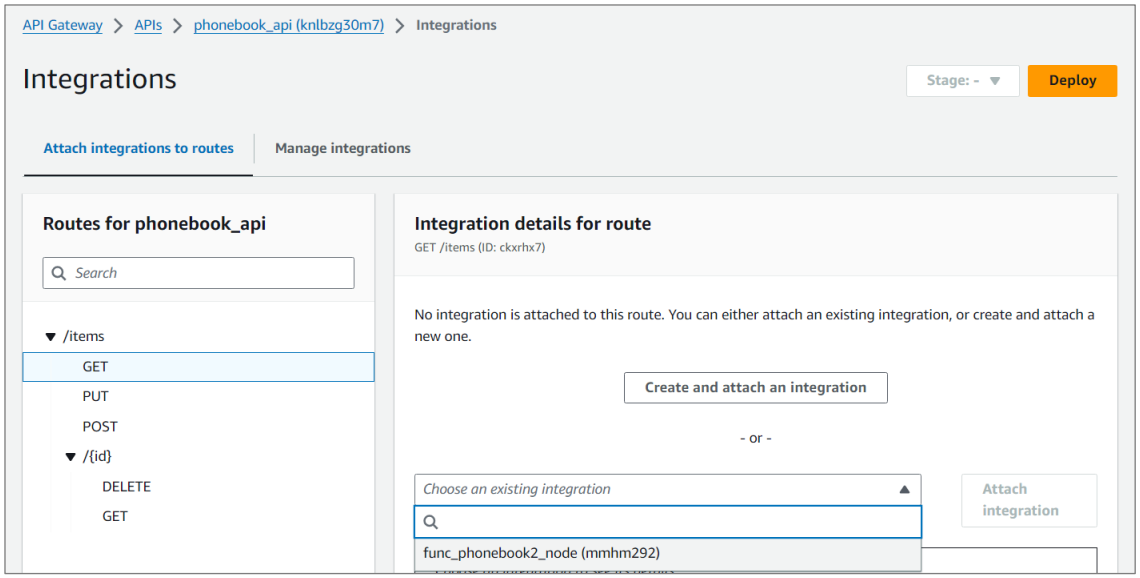


통합생성

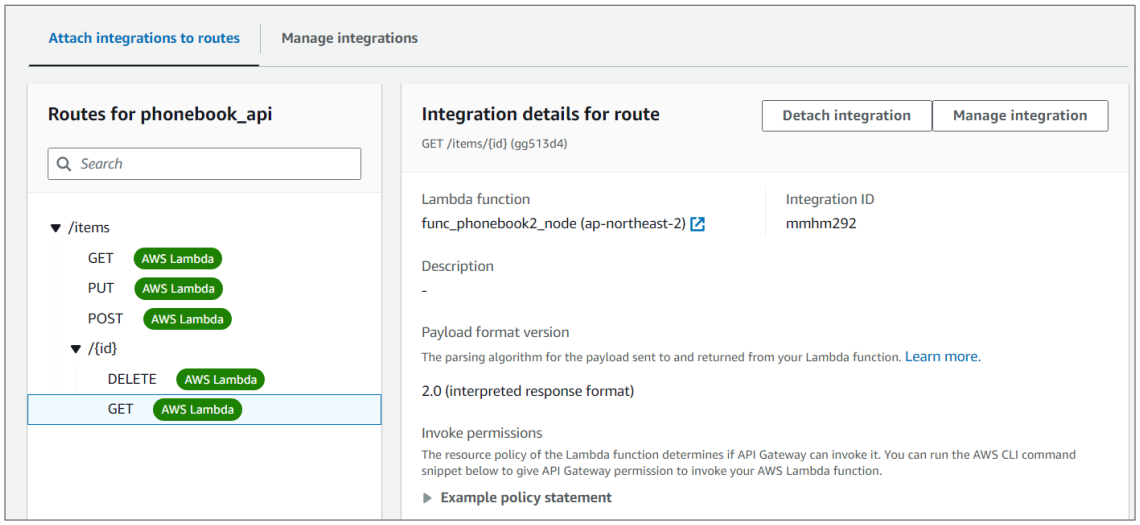


통합은 복수개가 생성되어 각 경로별로 붙을 수도 있음

경로별로 통합을 붙임



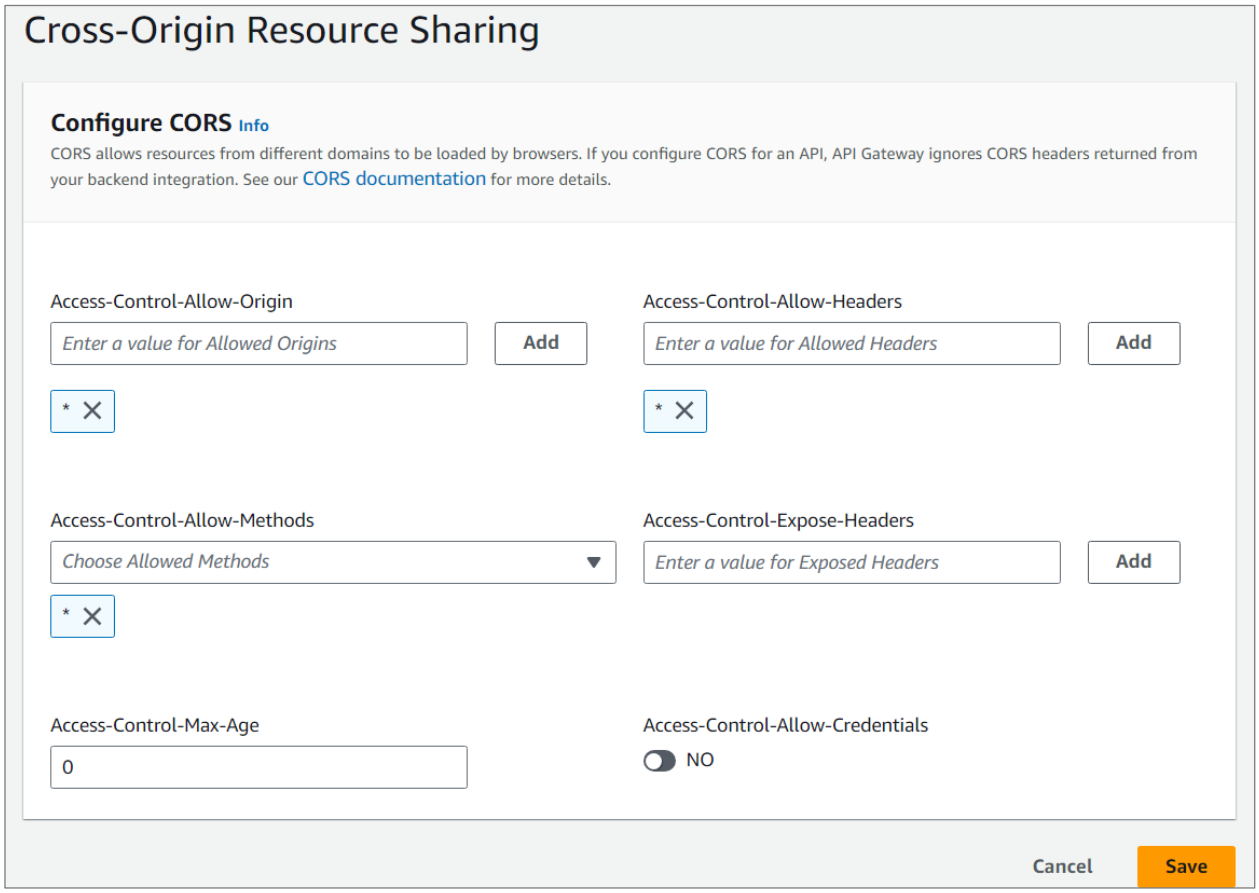
[최종통합 완료화면]



CORS설정

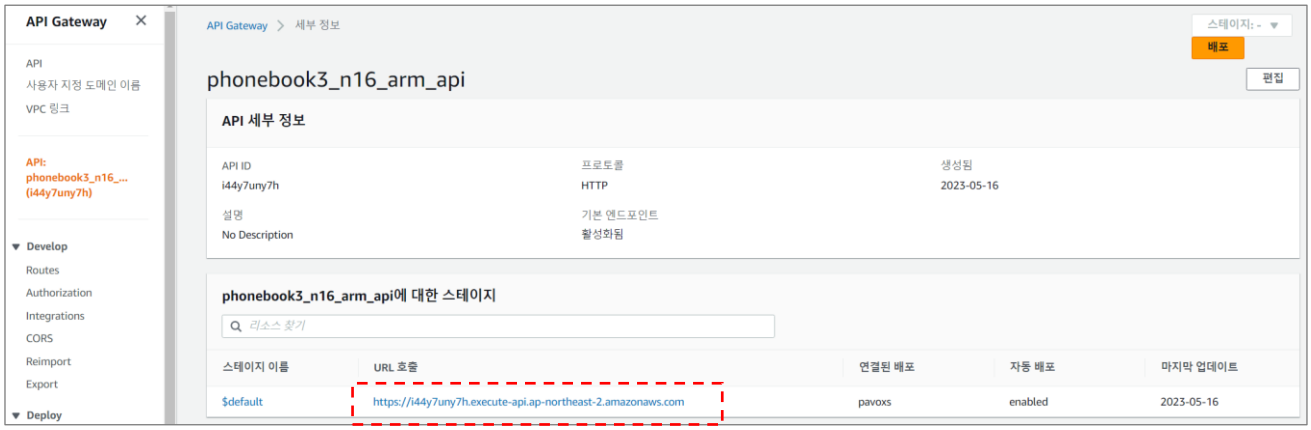
아래와 3값에 대한 세부처리 혹은 * 입력처리

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



[호출 테스트]

========== F1. API EndPoint 확인 ========



붉은색 박스 URL

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

========== F2. 호출방법선택 ========

방법1) EC2 혹은 노트북에서 curl로 진행

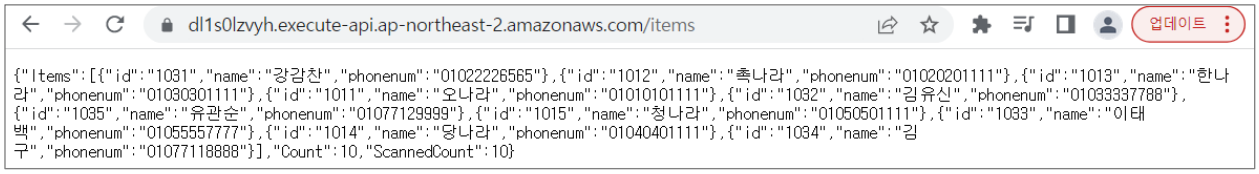
방법2) 웹호출 https://reqbin.com/curl (웹브라우저 secret mode)

방법3) Postman 등 API호출 도구사용

========== F3. API-호출테스트 진행 ========

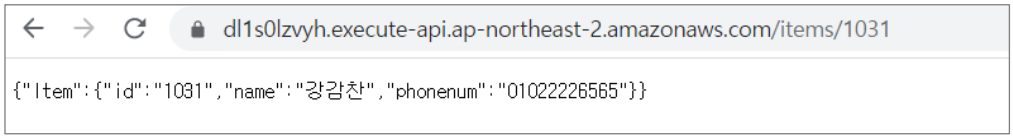
■ 전체목록 호출 GET /items

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



■ 단일호출 GET /items/{id}

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

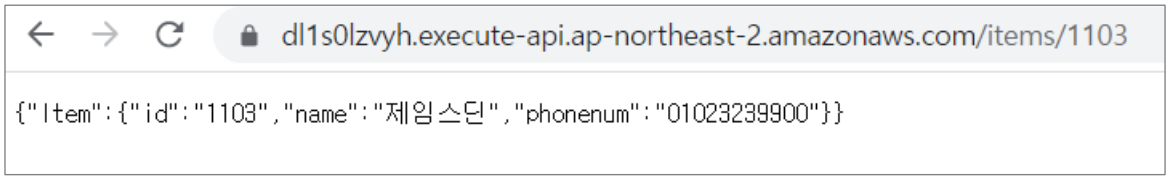


■ 데이터등록 PUT /items

```
curl -X PUT https://dl1s0lzvyh.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://dl1s0lzvyh.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 : 브라우져 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:DeleteObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::111122223333-front-phonebook",
                "arn:aws:s3:::111122223333-front-phonebook/*"
            ]
        }
    ]
}
```

정적페이지호출 (using JavaScript in S3)

~/apiclass : 디렉토리를 S3로 올려서 정적웹주소로 확인

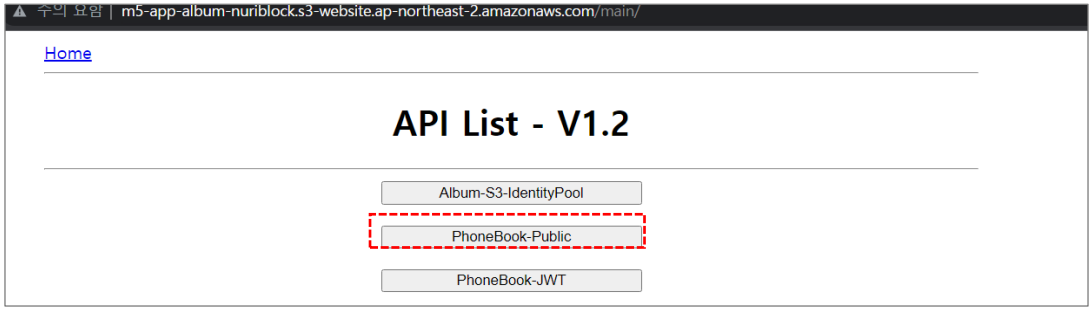
$ vi ./apistart/main/phonebook/main.js : 아래의 밑줄 API엔드포인트(호출주소) 변경

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

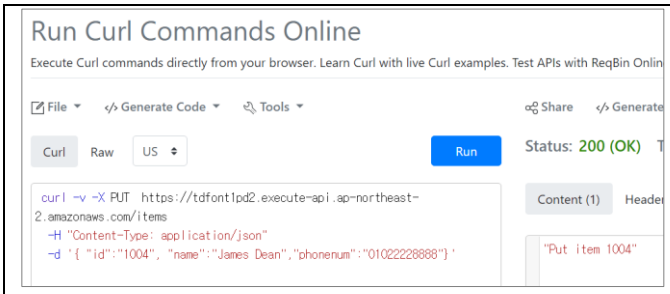
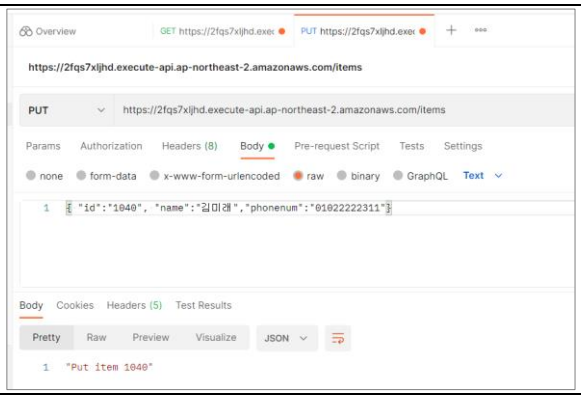
PUT /items	GET /item/{id}
	

※ API Test

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

(1) PUT 데이터입력

명령	출력
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"}' ※ 유의사항 : 경로확인 유의 (예제에서는 /items) (Invoke URL + API Path)	"Put item 1004"

	
(테스트 – 비회원) https://reqbin.com/curl	(테스트 – 포스트맨 다운로드 혹은 회원가입) https://www.postman.com/

(2) GET 데이터 단항목 가져오기

명령	출력

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

(3) GET 데이터 전체 가져오기

명령	출력

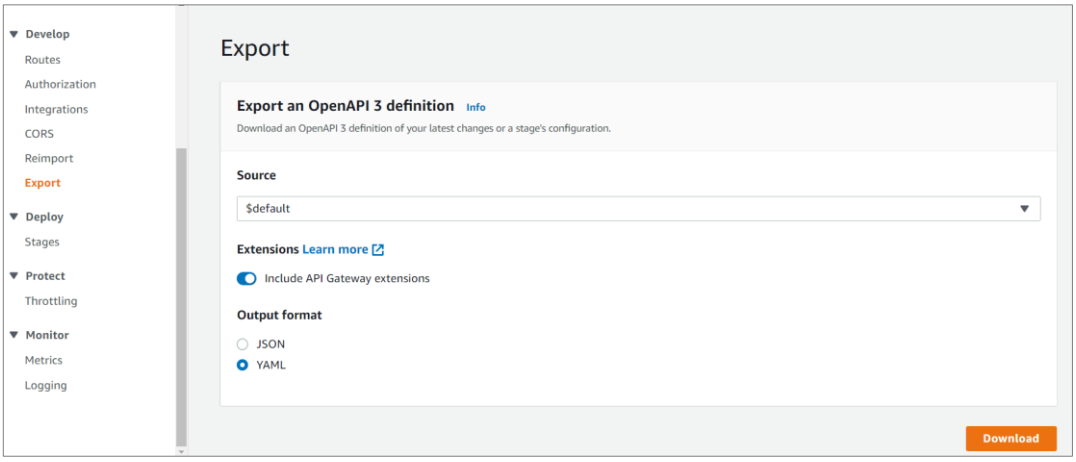
curl https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items
 	
※ Path : /items
	{
	"Items": [{
	"id": "1004",
	"name": "James Dean",
	"phonenum": "01022228888"
	}, {
	"id": "1020",
	"name": "임꺽정",
	"phonenum": "01020003000"
	}, {
	"id": "1040",
	"name": "홍삼정",
	"phonenum": "01020004000"
	},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

(4) DELETE 데이터삭제

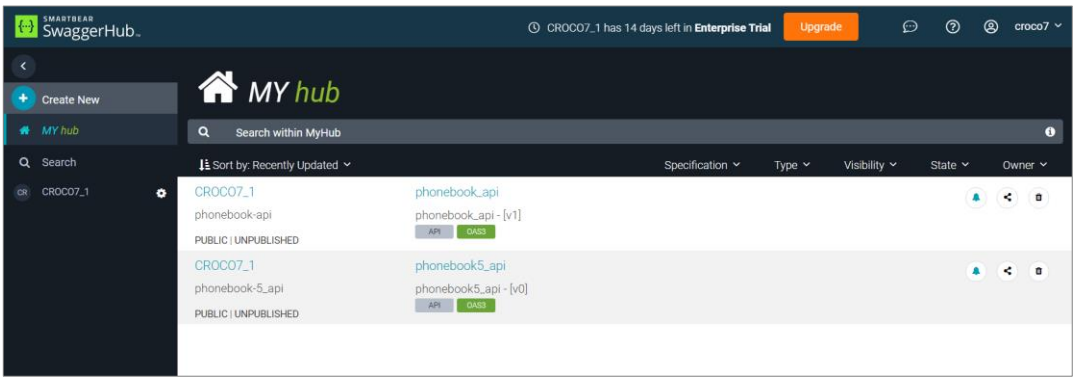
명령	출력
curl -X DELETE curl -v https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/items/1004 ※ Path : /items/{id}	"Deleted item 1004"

[첨부 1. OpenAPI 3 - SWAGGER 내보내기]

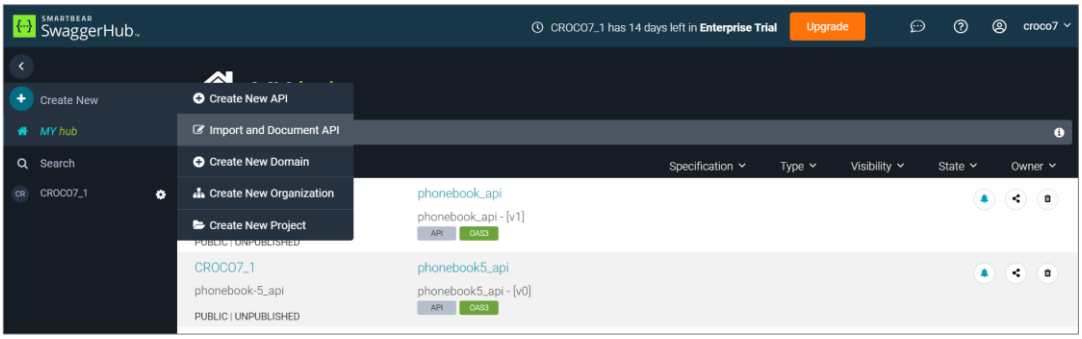
- swgger.io 서비스 이용하거나 swagger 구현
- API 내보내기 (AWS > API-Gateway)

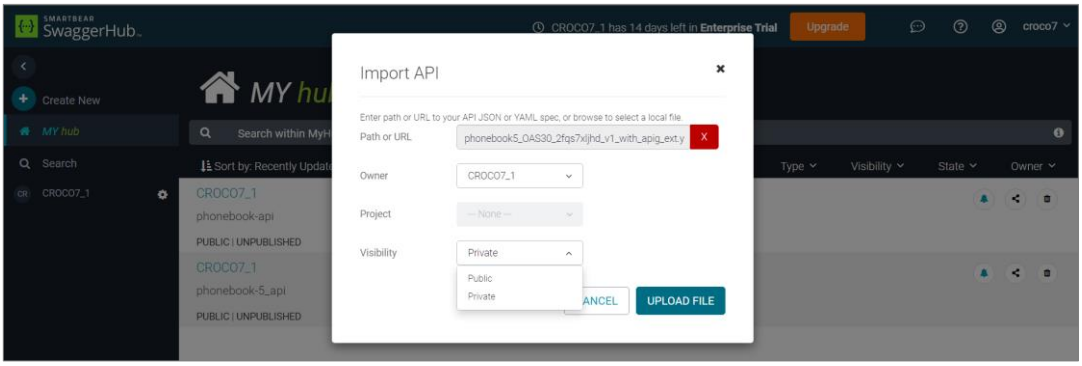


- API 가져오기 (SWAGGER)

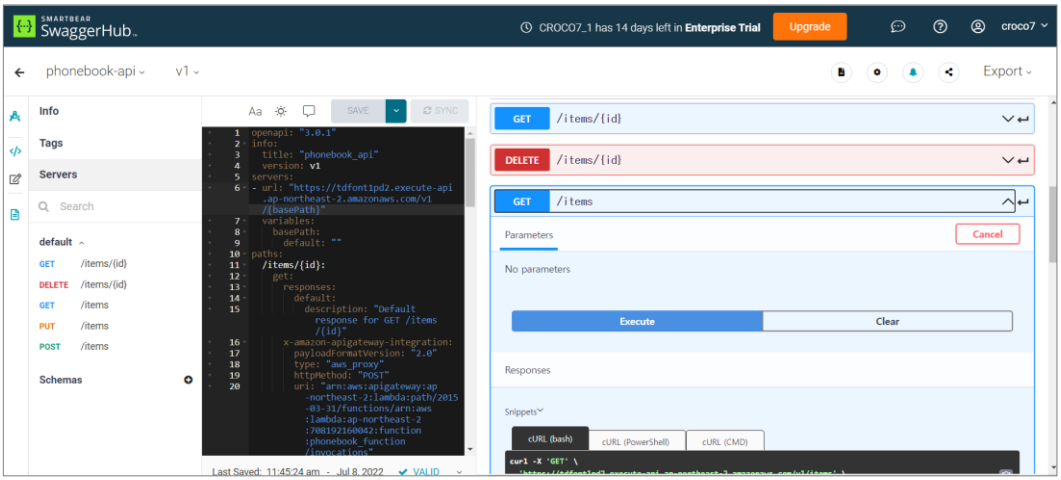


--> Import and Document API





- 실행


SwaggerHub
CROCO7_1 has 14 days left in Enterprise Trial
Upgrade
croco7
phonebook-api
v1
Export
Info
Tags
Servers
Search
default
GET /items/{id}
DELETE /items/{id}
GET /items
PUT /items
POST /items
Schemas
SAVE
SYNC
openapi: "3.0.1"
info:
title: "phonebook_api"
version: v1
servers:
- url: "https://tdfont1pd2.execute-api.ap-northeast-2.amazonaws.com/v1/{basePath}"
variables:
basePath:
default: ""
paths:
/items/{id}:
get:
responses:
default:
description: "Default response for GET /items/{id}"
x-amazon-apigateway-integration:
payloadFormatVersion: "2.0"
type: "aws_proxy"
httpMethod: "POST"
uri: "arn:aws:apigateway:ap-northeast-2:lambda:path/2015-03-31/functions/arn:aws:lambda:ap-northeast-2:708192160842:function:phonebook_function/invocations"
GET /items/{id}
DELETE /items/{id}
GET /items
Parameters
Cancel
No parameters
Execute
Clear
Responses
Snippets
cURL (bash)
cURL (PowerShell)
cURL (CMD)
curl -X 'GET' \
Last Saved: 11:45:24 am - Jul 8, 2022
VALID

[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

policy_svc_log_create_record.json

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Effect": "Allow",
        "Action": [
           "logs:CreateLogGroup",
           "logs:CreateLogStream",
           "logs:PutLogEvents"
        ],
        "Resource": "*"
     }
  ]
}
```

(2) Lambda 실행

policy_svc_lambda_invoke.json

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "lambda:InvokeFunction",
         "Resource": "*"
      }
   ]
}
```

(3) 다이나모 DB

policy_svc_dynamodb_rw.json

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "dynamodb:Query",
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:Scan",
            "dynamodb:UpdateItem"
         ],
         "Resource": "*"
      }
   ]
}
```

API-Gateway 실행

policy_svc_apigateway_invoke.json

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "execute-api:Invoke",
            "execute-api:ManageConnections"
         ],
         "Resource": "arn:aws:execute-api:*:*:*"
      }
   ]
}
```

※ 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": "ANPA2JY4KZEVM5HIX4NHM",
      "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": "AROA2JY4KZEVDFLPVOC5P",
      "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)

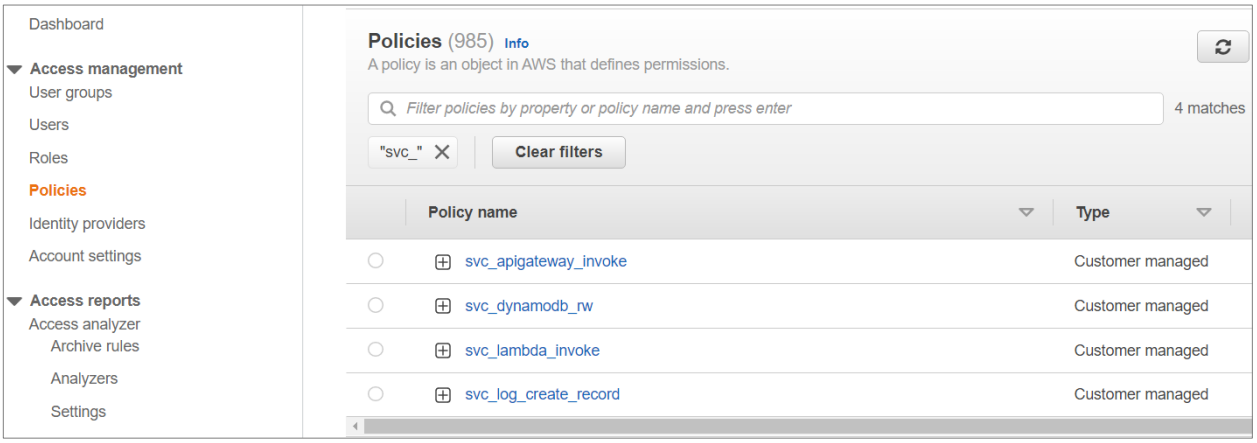
aws iam attach-role-policy --role-name example-role --policy-arn "arn:aws:iam::aws:policy/AmazonRDSReadOnlyAccess"

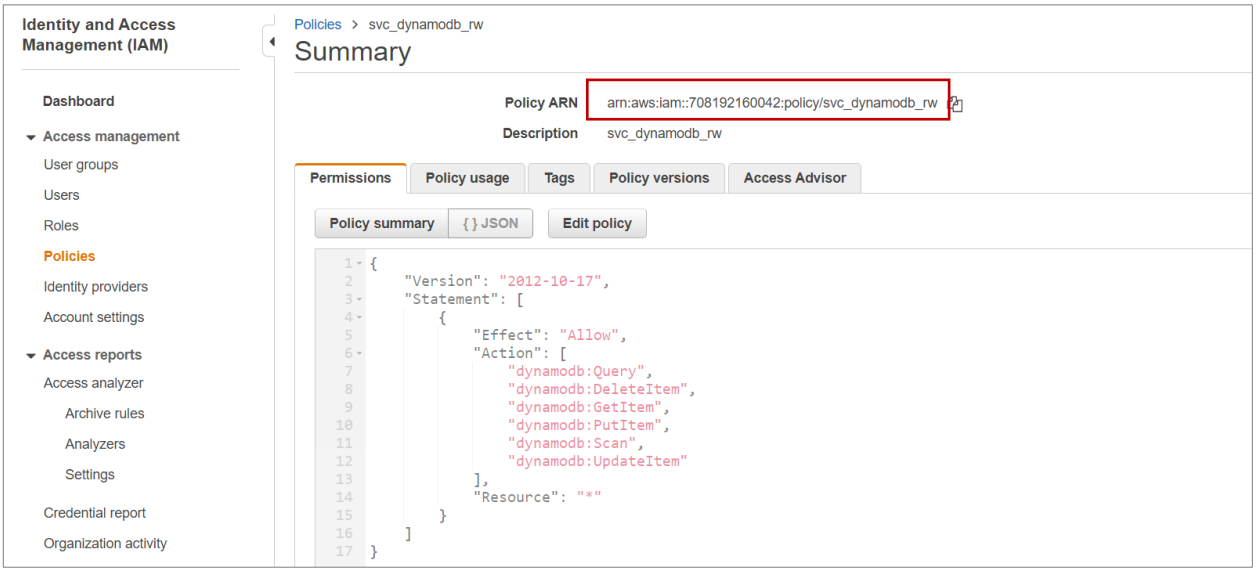
(정책확인 CLI 명령어)

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)

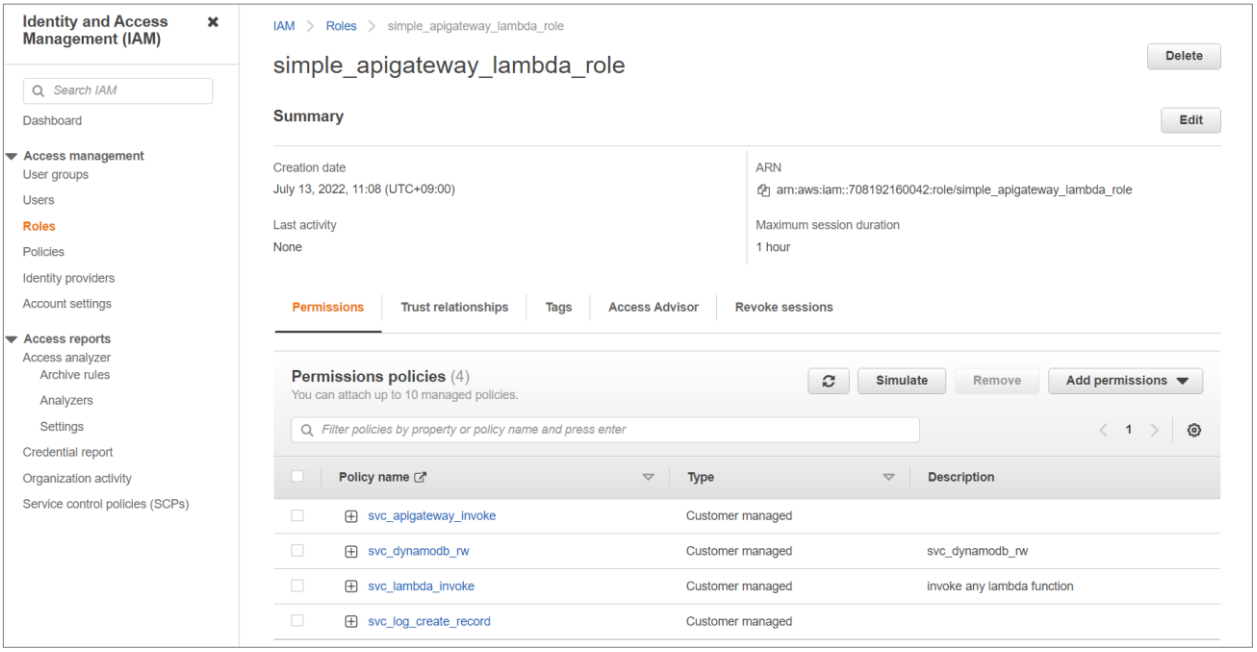
```
aws iam attach-role-policy --role-name simple_apigateway_lambda_role  --policy-arn
arn:aws:iam::111122223333:policy/svc_apigateway_invoke

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



[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 -