

GO Lambda 연동 매뉴얼

작성자: 김영훈

작성일: 2023.11.15

버전: 0.1

목차

GO 설명	3
GO란?	3
GO의 특징.....	3
GO의 단점.....	3
GO 설치	4
코드 설치	6
Lambda 함수 생성 및 설정.....	7
API Gateway 생성 및 설정.....	10
GO Lambda 함수 업로드.....	14
파일 압축.....	14
Lambda 함수 페이지에서 zip파일 업로드	14
AWS 커맨드 업로드	15
Postman 테스트.....	15
POST 메소드	15
GET 메소드	16
DELETE 메소드	16

GO 설명

GO란?

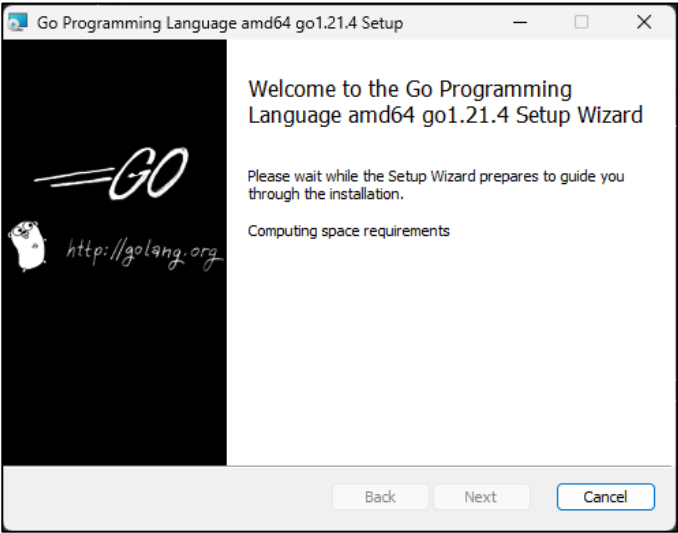
Google에서 개발한 오픈 소스 프로그래밍 언어로, 간결하고 빠르며 안전한 특성을 갖추고 있다. Go는 C 스타일의 정적 타입 언어이며, 특히 병행성 및 병렬성을 강조하는 특징이 있다. Go 언어는 간단한 문법, 효율적인 컴파일, 가비지 컬렉션과 같은 기능들로 알려져 있다.

GO의 특징

1. **간결한 문법:** 간결하고 가독성이 좋은 문법을 갖추고 있다.
2. **컴파일 속도와 실행 속도** 컴파일 속도가 빠르며, 실행 속도도 빠른 편에 속한다.
3. **병행성 지원:** 경량 스레드인 고루틴(Goroutine)을 제공하여 병행성을 효과적으로 다룰 수 있다.
4. **자동 메모리 관리:** 가비지 컬렉션을 통해 메모리 관리를 자동으로 처리한다.
5. **풍부한 표준 라이브러리** 풍부한 표준 라이브러리를 내장하고 있어, 다양한 작업을 간편하게 처리할 수 있다.
6. **정적 타입 언어:** 정적 타입 언어로, 변수의 타입을 선언하여 컴파일 시에 타입 안정성을 보장한다.
7. **포인터 지원:** 포인터를 지원하며, 메모리 주소에 직접 접근할 수 있다.
8. **크로스 플랫폼 지원:** 크로스 플랫폼 개발을 지원하여, 여러 운영 체제에서 동일한 코드를 실행할 수 있다.

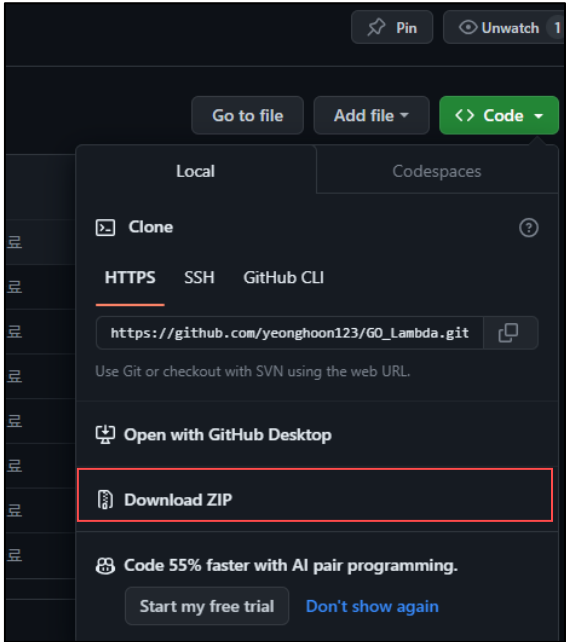
GO의 단점

1. **제한된 라이브러리 생태계:** 비교적 새로운 언어이기 때문에 다른 언어에 비해 라이브러리 생태계가 다소 제한적일 수 있다. 하지만 최근에는 커뮤니티와 라이브러리 생태계가 확장되고 있다.
2. **제한된 제네릭 프로그래밍 지원:** 현재로서는 제네릭 프로그래밍을 지원하지 않거나 지원이 제한적이다. 이로 인해 코드의 재사용성이 일부 제한될 수 있다. 하지만 이에 대한 개선이 Go 버전 업데이트에서 계획되어 있다.

<p>설치한 프로그램 파일 실행</p>	
<p>cmd창을 열어 커맨드 입력</p>	<p>Command: go version</p> 

코드 설치

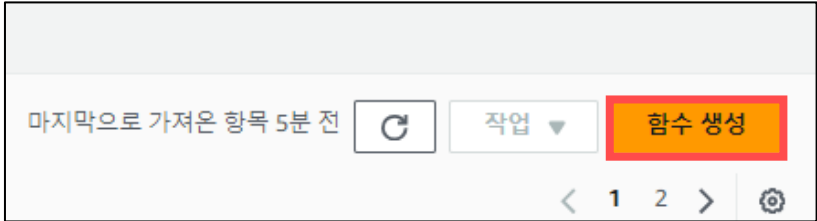
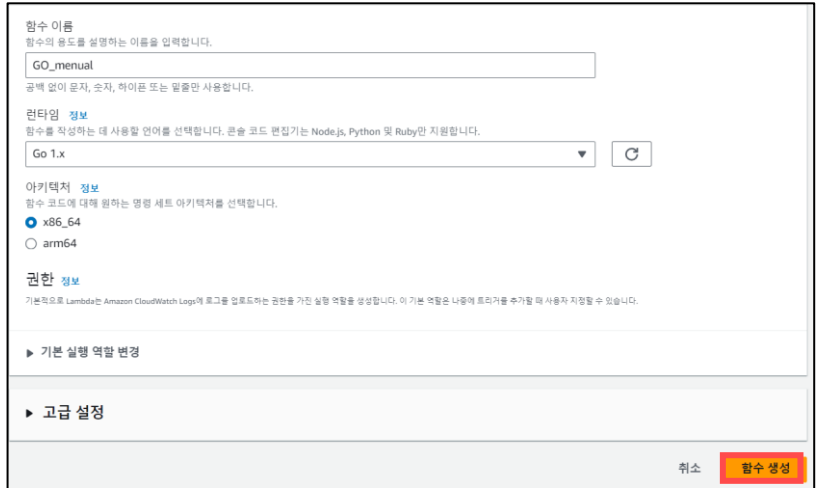
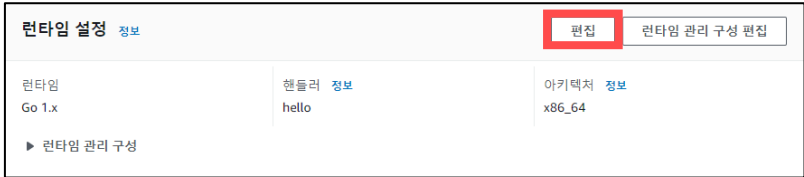
GO_Lambda repository에서 코드를 설치한다.


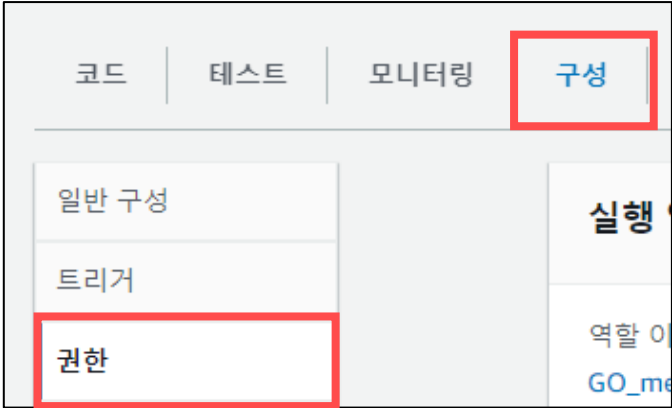
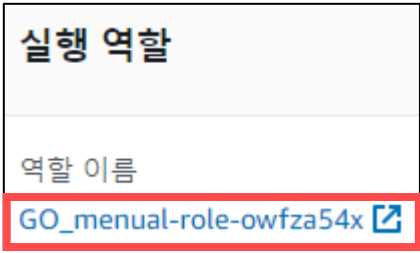
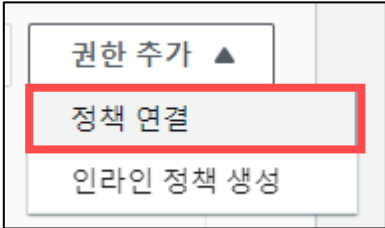
Git 사용	설치할 디렉토리에 커맨드 입력 Command: git clone https://github.com/yeonghoon123/GO_Lambda.git
zip파일 다운	링크 접속 후 Code 탭 선택 후 Download Zip 선택 링크: https://github.com/yeonghoon123/GO_Lambda 

Lambda 함수 생성 및 설정

Lambda 서비스 링크 접속

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

함수 생성 버튼 선택	
데이터 입력 후 함수 생성 버튼 선택	<p>런타임: go 1.x</p> 
코드 탭에 런타임 설정에 편집 선택	

<p>핸들러 변경 후 저장 선택</p>	<p>핸들러: main</p> 
<p>구성에 권한 탭으로 이동</p>	
<p>역할 이름 선택</p>	
<p>권한 추가 탭에 정책 연결 선택</p>	

권한 선택 후 권한 추가

AmazonAPIGatewayInvokeFullAccess,
AmazonDynamoDBFullAccess 선택

정책 이름

☒

AmazonDynamoDBFullAccess

☐

AmazonDynamoDBReadOnlyAccess

☐

AWSLambdaDynamoDBExecutionRole

☐

AWSLambdaInvocation-DynamoDB

☐

DynamoDbAccess

☐

svc_dynamodb_rw


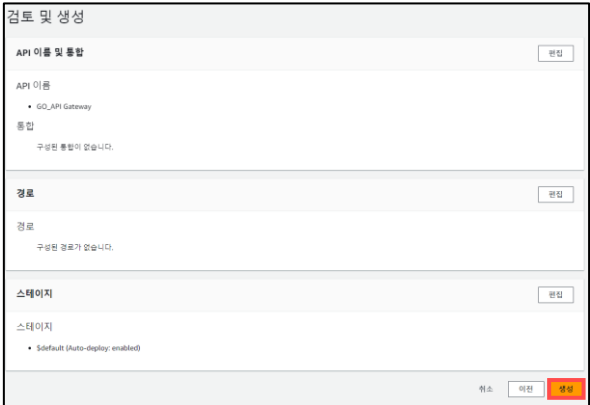
취소

권한 추가

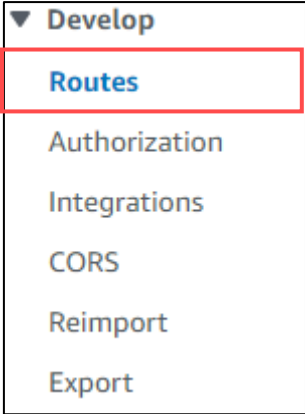
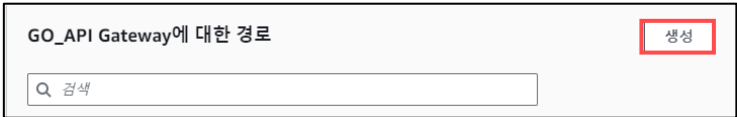
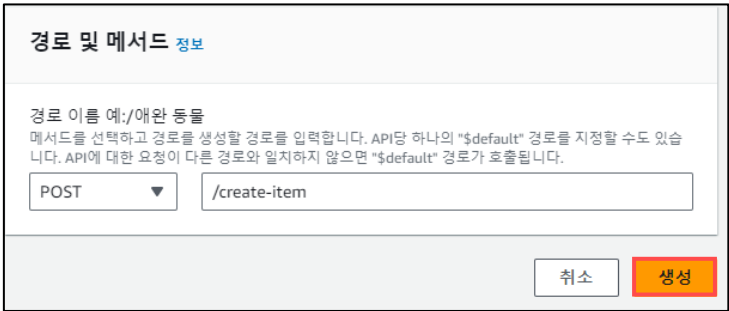
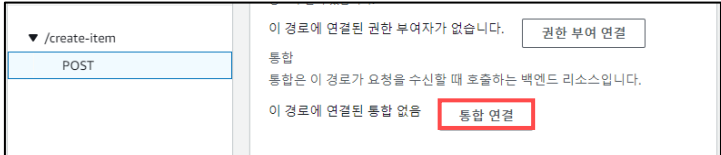

API Gateway 생성 및 설정

API Gateway 서비스 접속

<https://ap-northeast-2.console.aws.amazon.com/apigateway/main/apis?region=ap-northeast-2#>

API 생성 버튼 선택	
HTTP API 구축 버튼 선택	
API 이름 작성 후 검토 및 생성 선택	
생성 버튼 선택	

CORS 탭 선택	<div><div>▼ Develop</div><div>Routes</div><div>Authorization</div><div>Integrations</div><div>CORS</div><div>Reimport</div><div>Export</div></div>
구성 선택	<div><div>CORS 구성 정보</div><div>구성</div><div>지우기</div><p>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.</p></div>
값 기입 후 저장 버튼 선택	<p>Access-Control-Allow-Origin: *, Access-Control-Allow-Headers: * Access-Control-Allow-Methods: *, Access-Control-Max-Age: 60</p> <div><div><div>CORS 구성 정보</div><p>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.</p><div><div>Access-Control-Allow-Origin</div><div>허용되는 오리진 값 입력</div><div>추가</div><div>* X</div><div>Access-Control-Allow-Headers</div><div>허용되는 헤더 값 입력</div><div>추가</div><div>* X</div><div>Access-Control-Allow-Methods</div><div>허용되는 메서드 선택</div><div>≡ Access-Control-Expose-Headers</div><div>공개된 헤더 값 입력</div><div>추가</div><div>* X</div><div>Access-Control-Max-Age</div><div>60</div><div>Access-Control-Allow-Credentials</div><div>아니요</div><div>취소</div><div>저장</div></div></div></div>

Routes 탭 선택	
생성 버튼 선택	
경로와 메소드 입력 후 생성 버튼 선택	
생성된 메소드 선택 후 통합 연결 버튼 선택	
통합 생성 및 연결 선택	

<div>통합 유형과 세부 정보 입력 후 생성 버튼 선택</div>	<div>통합 유형 – Lambda 통합 세부정보 – 생성한 Lambda함수</div>
	<div><div>통합 대상</div><div>통합 유형 Lambda 함수</div><div>통합 세부 정보</div><div>통합 대상 경로가 요청을 수신할 때 API Gateway가 호출할 Lambda 함수를 선택합니다.</div><div><div>AWS 리전 ap-northeast-2</div><div>Lambda 함수 arn:aws:lambda:ap-northeast-2:141352286036:function:GO_manual</div></div><div>▶ 고급 설정</div><div>설명 - 선택 사항</div><div>호출 권한</div><div>API Gateway는 Lambda 함수의 리소스 정책에서 이 API 권한을 부여하여 Lambda 함수를 호출할 권한을 요청합니다. 리소스 정책을 수정하지 않으려는 경우 호출 역할을 대신 제공할 수 있습니다. API Gateway는 이 역할을 사용하여 Lambda 함수를 호출합니다.</div><div><input checked="" type="checkbox"/> API Gateway에 Lambda 함수를 호출할 권한 부여</div><div>취소 생성</div></div>

GO Lambda 함수 업로드

파일 압축

설치한 디렉토리 터미널 접속	
커맨드를 입력해 파일 빌드	Command: SET GOOS=linux& go build main.go
커맨드를 입력해 function.zip으로 생성	Command: zip function.zip main

Lambda 함수 페이지에서 zip파일 업로드

생성한 Lambda 함수 접속 후 에서 업로드 탭에서 .zip 파일 선택	
생성한 function.zip 업로드 후 저장 버튼 선택	

AWS 커맨드 업로드

- AWS 커맨드를 사용하려는 경우 AWS cli 설정이 완료되어 있어야 사용 가능

command: `aws lambda update-function-code --function-name my-function --zip-file fileb://function.zip`

```
D:\GO_Lambda>aws lambda update-function-code --function-name GO_manual --zip-file fileb://function.zip
{
  "FunctionName": "GO_manual",
```

Postman 테스트

POST 메소드

URL: `https://{my-apigateway-url}/{my-route-post-method}`,

Header: `{'Content-Type': 'application/json'}`,

Body: {

```
  "id": "",
  "sttText": "",
  "languageCode": "",
  "ttsBase64": {
    "ja": "",
    "ko": "",
    "em": ""
  },
  "translatorText": {
    "ja": "",
    "ko": "",
    "em": ""
  },
  "languageName": {
    "ja": "",
    "ko": "",
    "em": ""
  }
}
```

Result:

```
{"Status":true,"Message":"Create item success","Data":null}
```

GET 메소드

URL: https://{my-apigateway-url}/{my-route-get-method},

Result:

```
{
  "Status": true,
  "Message": "Scanning Data success",
  "Data": [
    {
      "Id": "test12312",
      "Stt_text": "test Stt Text",
      "Language_code": "test languageCode",
      "Tts_base64": {
        "ko": "ko ttsbase64",
        "en": "en ttsbase64",
        "ja": "ja ttsbase64"
      },
      "Translator_text": {
        "ko": "ko translatorText",
        "en": "en translatorText",
        "ja": "ja translatorText"
      },
      "Language_name": {
        "ko": "ko languageName",
        "en": "en languageName",
        "ja": "ja languageName"
      }
    }
  ]
}
```

DELETE 메소드

URL: https://{my-apigateway-url}/{my-route-delete-method},

Header: {'Content-Type': 'application/json'},

Body: { "Id": "" }

Result: {"Status":true,"Message":"Delete item success","Data":null}