



COMMUNITY DAY

SEOUL



쌤(SAM)! 도와주세요!

이태현 | 당근, 한국외대 스칸디나비아어학과, AUSG



COMMUNITY DAY

SEOUL

발표 소개



발표 자료

<https://bit.ly/help-me-sam>



목표

어떻게 하면 로컬에서 AWS Lambda를 테스트해 볼 수 있을까?

어떻게 하면 분산된 AWS Lambda 함수를 잘 관리할 수 있을까?

목표

영상 업로드



썸네일 이미지 추출



이미지 업로드



목표

영상 업로드



썸네일 이미지 추출

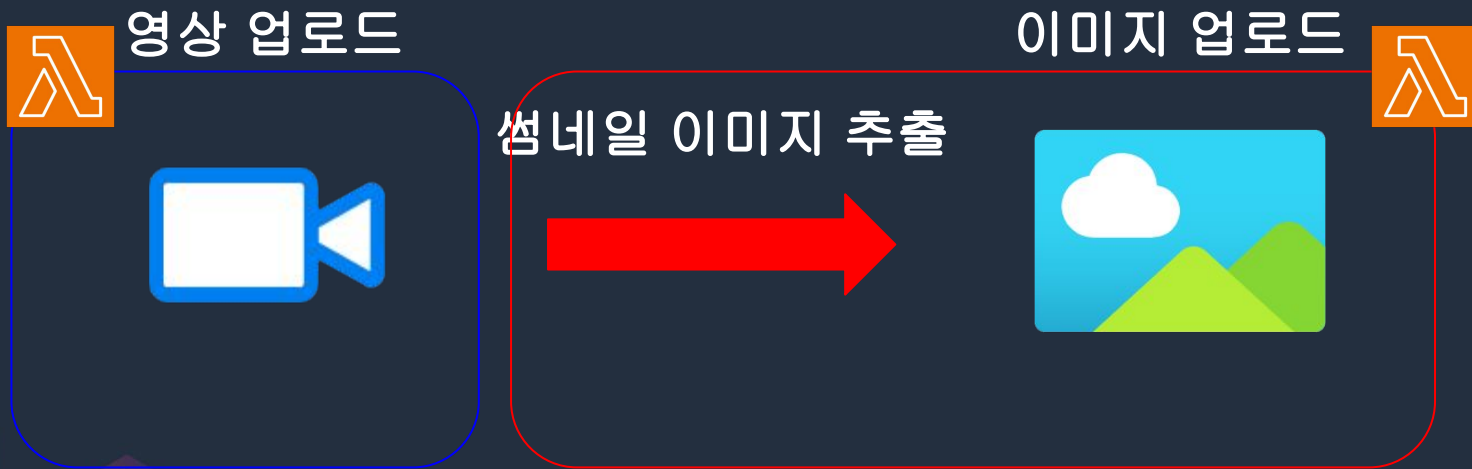


이미지 업로드



어차피 **간헐적으로 실행될 텐데 서버리스로 구현할 수는 없을까?**

목표



목표

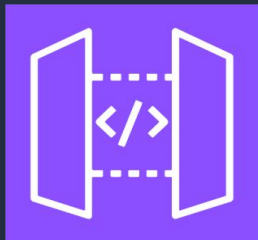
API Gateway

영상 업로드

S3

이미지 업로드

S3



작업의 흐름

목표

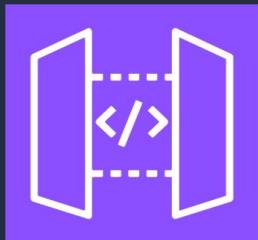
API Gateway

영상 업로드

S3

이미지 업로드

S3



소스 코드도 한 곳에서 관리하고, 배포 안 하고 테스트 해볼 수는 없나?

목표

어떻게 하면 로컬에서 AWS Lambda를 테스트해 볼 수 있을까?

어떻게 하면 분산된 AWS Lambda 함수를 잘 관리할 수 있을까?

목표

어떻게 하면 로컬에서 AWS Lambda를 테스트해 볼 수 있을까?

목표

- AWS SAM이 무엇인지 직접 **템플릿 YAML 파일**을 살펴보며 이해해요.

목표

- AWS SAM이 무엇인지 직접 템플릿 YAML 파일을 살펴보며 이해해요.
- 로컬에서 AWS API Gateway를 구동해보며, Zip 패키지 유형의 내부 작동을 이해해요.

목표

- AWS SAM이 무엇인지 직접 템플릿 YAML 파일을 살펴보며 이해해요.
- 로컬에서 AWS API Gateway를 구동해보며, Zip 패키지 유형의 내부 작동을 이해해요.
- AWS API Gateway를 로컬에서 사용할 때 유의해야 할 점을 이해해요.

목표

- AWS SAM이 무엇인지 직접 템플릿 YAML 파일을 살펴보며 이해해요.
- 로컬에서 AWS API Gateway를 구동해보며, Zip 패키지 유형의 내부 작동을 이해해요.
- AWS API Gateway를 로컬에서 사용할 때 유의해야 할 점을 이해해요.
- LocalStack을 사용하여 로컬에서 AWS S3를 구축하는 방법을 이해해요.

목표

- **AWS SAM**이 무엇인지 직접 템플릿 **YAML** 파일을 살펴보며 이해해요.
- 로컬에서 **AWS API Gateway**를 구동해보며, **Zip** 패키지 유형의 내부 작동을 이해해요.
- **AWS API Gateway**를 로컬에서 사용할 때 유의해야 할 점을 이해해요.
- **LocalStack**을 사용하여 로컬에서 **AWS S3**를 구축하는 방법을 이해해요.
- **RIE**(Runtime Interface Emulator), **RIC**(Runtime Interface Client)에 대해 이해해요.

목표

- **AWS SAM**이 무엇인지 직접 템플릿 **YAML** 파일을 살펴보며 이해해요.
- 로컬에서 **AWS API Gateway**를 구동해보며, **Zip** 패키지 유형의 내부 작동을 이해해요.
- **AWS API Gateway**를 로컬에서 사용할 때 유의해야 할 점을 이해해요.
- **LocalStack**을 사용하여 로컬에서 **AWS S3**를 구축하는 방법을 이해해요.
- **RIE(Runtime Interface Emulator)**, **RIC(Runtime Interface Client)**에 대해 이해해요.
- 로컬에서 **AWS S3** 이벤트를 직접 다루어보며, **Image** 패키지 유형의 내부 작동을

목표

- **AWS SAM**이 무엇인지 직접 템플릿 **YAML** 파일을 살펴보며 이해해요.
- 로컬에서 **AWS API Gateway**를 구동해보며, **Zip** 패키지 유형의 내부 작동을 이해해요.
- **AWS API Gateway**를 로컬에서 사용할 때 유의해야 할 점을 이해해요.
- **LocalStack**을 사용하여 로컬에서 **AWS S3**를 구축하는 방법을 이해해요.
- **RIE(Runtime Interface Emulator)**, **RIC(Runtime Interface Client)**에 대해 이해해요.
- 로컬에서 **AWS S3** 이벤트를 직접 다루어보며, **Image** 패키지 유형의 내부 작동을

목표

- **AWS SAM**이 무엇인지 직접 템플릿 **YAML** 파일을 살펴보며 이해해요.
- 로컬에서 **AWS API Gateway**를 구동해보며, **Zip** 패키지 유형의 내부 작동을 이해해요.
- **AWS API Gateway**를 로컬에서 사용할 때 유의해야 할 점을 이해해요.
- **LocalStack**을 사용하여 로컬에서 **AWS S3**를 구축하는 방법을 이해해요.
- **RIE**(Runtime Interface Emulator), **RIC**(Runtime Interface Client)에 대해 이해해요.
- 로컬에서 **AWS S3** 이벤트를 직접 다루어보며, **Image** 패키지 유형의 내부 작동을



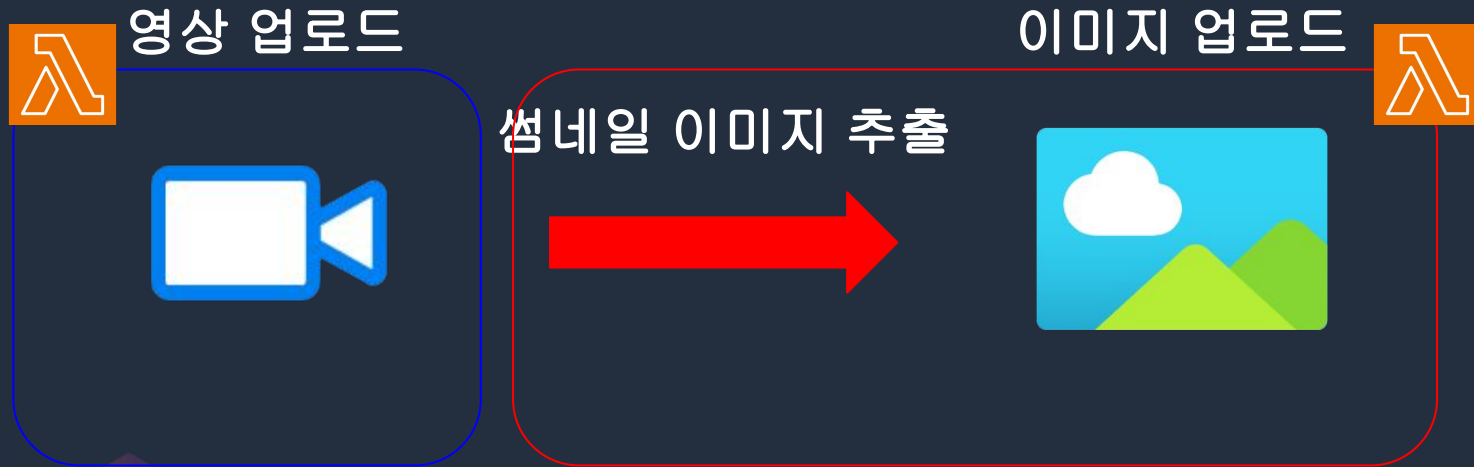
COMMUNITY DAY

SEOUL

프로젝트 개요



프로젝트 개요



프로젝트 개요

: 영상 업로드 애플리케이션



프로젝트 개요

: 영상 업로드 애플리케이션 로컬 환경



AWS SAM

GET /upload/url



Lambda

서명된 URL 생성



LocalStack S3

프로젝트 개요

: 영상 이미지 생성 애플리케이션



프로젝트 개요

: 영상 이미지 생성 애플리케이션 로컬 환경



AWS SAM

이벤트 트리거



Lambda

이미지 업로드



LocalStack S3



AWS CloudFormation과 SAM

CloudFormation



CloudFormation



laC(Infrastructure as Code)

: AWS 자원을 코드로 관리하게 도와주는
서비스

CloudFormation

template.yaml

Resources:

TestS3Bucket:

Type: **AWS::S3::Bucket**

Properties:

AccessControl: **Private**

VersioningConfiguration:

Status: **Enabled**



CloudFormation

template.yaml

Resources:

TestS3Bucket:

Type: **AWS::S3::Bucket**

Properties:

AccessControl: Private

VersioningConfiguration:

Status: Enabled



SAM

Serverless Application Model



SAM



laC(Infrastructure as Code)

: 서버리스 애플리케이션 관리를 도와주는 서비스

SAM



laC(Infrastructure as Code)

: 서버리스 애플리케이션 관리를 도와주는
서비스

SAM



CloudFormation을 활용하여
로컬 환경에서의 구동과 애플리케이션 관리
및 배포

SAM

template.yaml

Resources:

VideoUploader:

Type: `AWS::Serverless::Function`

Properties:

Runtime: `python3.9`

PackageType: `Image`

Timeout: `600`

MemorySize: `1024`



SAM

template.yaml

Resources:

VideoUploader:

Type: **AWS::Serverless::Function**

Properties:

Runtime: python3.9

PackageType: Image

Timeout: 600

MemorySize: 1024





Zip 패키지 유형의 영상 업로드 애플리케이션



Zip 패키지 유형의 영상 업로드 애플리케이션



Zip 패키지 유형의 영상 업로드 애플리케이션



AWS SAM

GET /upload/url



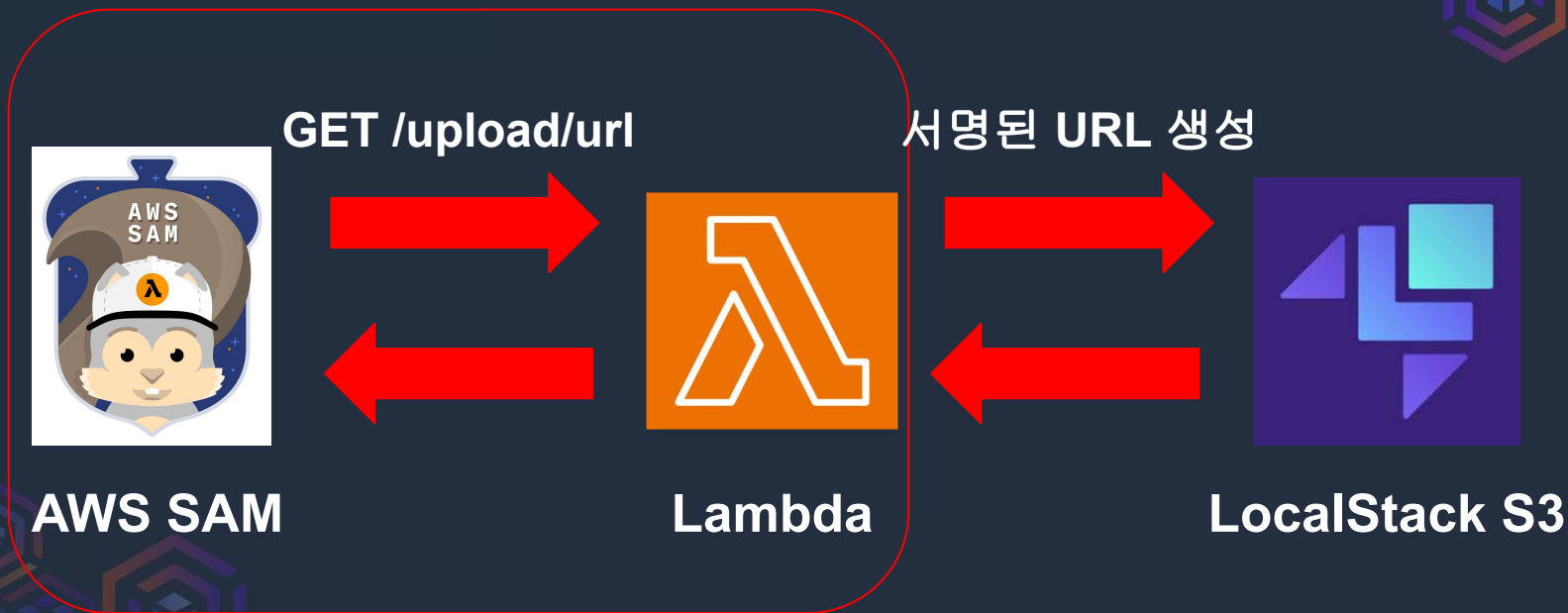
Lambda

서명된 URL 생성



LocalStack S3

Zip 패키지 유형의 영상 업로드 애플리케이션



Lambda 함수 및 API Gateway 정의



AWS SAM 템플릿 정의

AWS Lambda 함수 정의



Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: python3.9

PackageType: Zip

CodeUri: ../app

Handler: src.main.lambda_handler

Timeout: 600

MemorySize: 1024

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: python3.9

PackageType: Zip

CodeUri: ../app

Handler: src.main.lambda_handler

Timeout: 600

MemorySize: 1024

프로그래밍 언어 및
환경

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: python3.9

PackageType: Zip

CodeUri: ../app

Handler: src.main.lambda_handler

Timeout: 600

MemorySize: 1024

Lambda 함수 패키지

유형



Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: `python3.9`

PackageType: `Zip`

CodeUri: `../app`

Handler: `src.main.lambda_handler`

Timeout: `600`

MemorySize: `1024`



실행 대상 소스 코드 위치 및
핸들러 함수 (**Lambda** 함수
진입점)

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: python3.9

PackageType: Zip

CodeUri: ../app

Handler: src.main.lambda_handler

Timeout: 600

MemorySize: 1024



최대 실행시간 및 메모리 크기

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties

Properties:

Runtime: python3.9

PackageType: Zip

CodeUri: ../app

Handler: src.main.lambda_handler

Timeout: 600

MemorySize: 1024



CodeUri 및 Handler는
Zip 패키지일 때 필수 속성



AWS SAM 템플릿 정의

AWS API Gateway 정의



Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties의 Events

Properties:

...

Events:

UploadVideo:

Type: **HttpApi**

Properties:

Path: **/upload/url**

Method: **get**

Apild: **!Ref** UploadVideoAPI

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties의 Events

Properties:

...

Events:

UploadVideo:

Type: **HttpApi**



API Gateway 종류 중 HTTP

Properties:

API

Path: /upload/url

Method: get

Apild: **!Ref** UploadVideoAPI

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties의 Events

Properties:

...

Events:

UploadVideo:

Type: HttpApi

Properties:

Path: /upload/url



엔드포인트

Method: get



HTTP 메서드

ApiId: !Ref UploadVideoAPI

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 Properties의 Events

Properties:

...

Events:

UploadVideo:

Type: HttpApi

Properties:

Path: /upload/url

Method: get

ApiId: !Ref UploadVideoAPI



YAML 내 정의된 리소스

참조

Zip 패키지 유형의 영상 업로드 애플리케이션

template.yaml 중 UploadVideoAPI

UploadVideoAPI:  YAML 내 정의된 리소스

Type: AWS::Serverless::HttpApi

Properties:

StageName: test

AllowOrigins:

- "*"

AllowMethods:

- GET

Zip 패키지 유형의 영상 업로드 애플리케이션



AWS SAM

GET /upload/url



Lambda

서명된 URL 생성



LocalStack S3

업로드를 위한 로컬 환경의 S3 정의

Docker를 활용한 LocalStack 정의

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 s3-local

s3-local:

image: localstack/localstack

container_name: localstack

ports:

- "4566:4566"

environment:

- SERVICES=s3

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 s3-local

s3-local:

image: localstack/localstack



LocalStack 공식 이미지

container_name: localstack

ports:

- "4566:4566"

environment:

- SERVICES=s3



사용할 LocalStack 서비스

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 s3-local

s3-local:

image: localstack/localstack

container_name: localstack

ports:

- "4566:4566"

environment:

- SERVICES=s3

→ 컨테이너 이름이면서 동시에
내부 통신에서 사용될 호스트
이름

→ 내부 통신에서 사용될 포트

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 s3-local

s3-local:

image: localstack/localstack

container_name: localstack

ports:

- "4566:4566"

environment:

- SERVICES=s3

컨테이너 이름이면서 동시에
내부 통신에서 사용될 호스트
이름

내부 통신에서 사용될 포트

http://localstack:4566

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 aws-cli

```
aws-cli:
```

```
  build:
```

```
    context: ..
```

```
    dockerfile: dockerfiles/Dockerfile.local
```

```
  container_name: aws-cli
```

```
  depends_on:
```

```
    - s3-local
```

Zip 패키지 유형의 영상 업로드 애플리케이션

docker-compose.yaml 중 Services의 aws-cli

```
aws-cli:
```

```
  build:
```


```
    context: ..
```

```
    dockerfile: dockerfiles/Dockerfile.local
```

```
  container_name: aws-cli
```

```
  depends_on:
```

```
    - s3-local
```



LocalStack 통한 S3 생성 이후
해당 S3에 버킷을 생성하기 위한
의존성

Zip 패키지 유형의 영상 업로드 애플리케이션

로컬 S3 내 버킷을 생성하는 local_aws_s3_setup.sh 파일

```
#!/bin/sh
```

```
aws s3 mb s3://help-me-sam --endpoint-url=http://localstack:4566
```

Zip 패키지 유형의 영상 업로드 애플리케이션

로컬 S3 내 버킷을 생성하는 local_aws_s3_setup.sh 파일

```
#!/bin/sh
```

```
aws s3 mb s3://help-me-sam --endpoint-url=http://localhost:4566
```

 LocalStack S3를 엔드포인트로 설정

Zip 패키지 유형의 영상 업로드 애플리케이션



LocalStack S3



Bucket

s3-local 통해 S3 생성

aws-cli 통해 버킷 생성

Zip 패키지 유형의 영상 업로드 애플리케이션



LocalStack S3

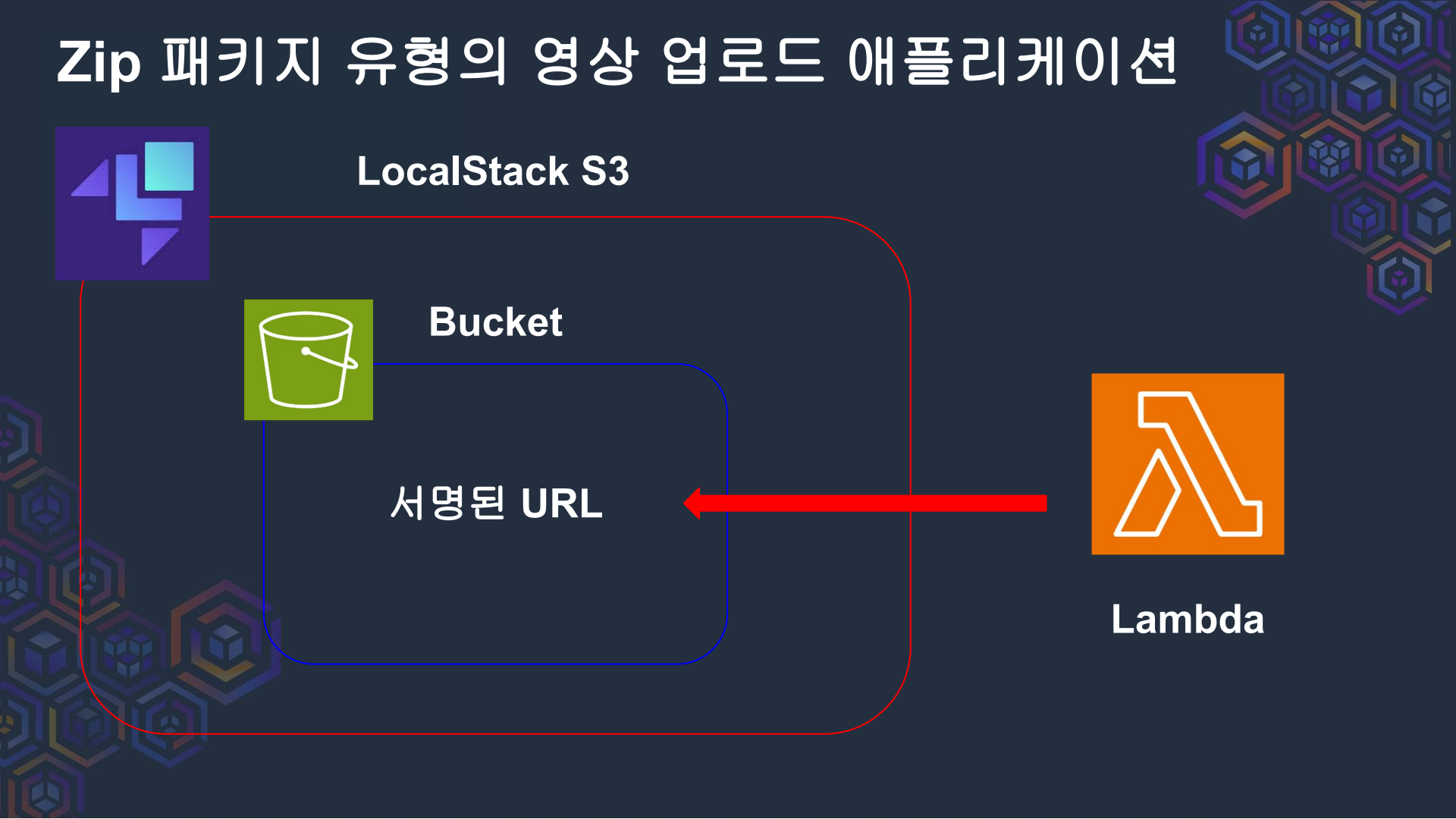


Bucket

서명된 URL



Lambda



영상을 직접 업로드 하면 안 되나?

HTTP API quotas

The following quotas apply to configuring and running an HTTP API in API Gateway.

| Resource or operation | Default quota | Can be increased |
|-------------------------------------------------------|---------------|------------------|
| Routes per API | 300 | Yes |
| Integrations per API | 300 | No |
| Maximum integration timeout | 30 seconds | No |
| Stages per API | 10 | Yes |
| Multi-level API mappings per domain | 200 | No |
| Tags per stage | 50 | No |
| Total combined size of request line and header values | 10240 bytes | No |
| Payload size | 10 MB | No |
| Custom domains per account per Region | 120 | Yes |
| Access log template size | 3 KB | No |
| Amazon CloudWatch Logs log entry | 1 MB | No |
| Authorizers per API | 10 | Yes |
| Audiences per authorizer | 50 | No |
| Scopes per route | 10 | No |
| Timeout for JSON Web Key Set endpoint | 1500 ms | No |
| Response size from JSON Web Key Set endpoint | 150000 bytes | No |

Function configuration, deployment, and execution

The following quotas apply to function configuration, deployment, and execution. Except as noted, they can't be changed.

Note

The Lambda documentation, log messages, and console use the abbreviation MB (rather than MiB) to refer to 1024 KB.

| Resource | Quota |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function memory allocation | 128 MB to 10,240 MB, in 1-MB increments. Note: Lambda allocates CPU power in proportion to the amount of memory configured. You can increase or decrease the memory and CPU power allocated to your function using the Memory (MB) setting. At 1,769 MB, a function has the equivalent of one vCPU. |
| Function timeout | 900 seconds (15 minutes) |
| Function environment variables | 4 KB, for all environment variables associated with the function, in aggregate |
| Function resource-based policy | 20 KB |
| Function layers | five layers |
| Function burst concurrency | 500 - 3000 (varies per Region) Can be increased from default values. Service Quotas do not support changes in burst limits at this time. Contact AWS Support to inquire further . |
| Invocation payload (request and response) | 6 MB each for request and response (synchronous) 20 MB for each streamed response (Synchronous. The payload size for streamed responses can be increased from default values. Contact AWS Support to inquire further.) 256 KB (asynchronous) |
| Bandwidth for streamed responses | Uncapped for the first 6 MB of your function's response For responses larger than 6 MB, 2MBps for the remainder of the response |
| Deployment package (.zip file) | 50 MB (zipped, for direct upload) |

Zip 패키지 유형의 영상 업로드 애플리케이션



6MB보다 작은 페이로드(Payload)

: 6MB보다 큰 파일의 경우 S3 서명된 URL
사용

로컬에서의 AWS Lambda 실행

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam build -t template.yaml
```

Build Succeeded

Built Artifacts : .aws-sam/build

Built Template : .aws-sam/build/template.yaml

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam build -t template.yaml
```

Build Succeeded

Built Artifacts : .aws-sam/build

Built Template : .aws-sam/build/template.yaml

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam build -t template.yaml
```

Build Succeeded

Built Artifacts : [.aws-sam/build](#)

Built Template : [.aws-sam/build/template.yaml](#)

Zip 패키지 유형의 영상 업로드 애플리케이션

.aws-sam

build

└─ VideoUploader

└─ boto3

└─ src

└─ template.yaml

Zip 패키지 유형의 영상 업로드 애플리케이션

.aws-sam 디렉토리

build

└─ VideoUploader

└─ boto3

└─ src

└─ template.yaml



설치된 외부 패키지 및
실제 **AWS Lambda** 함수 소스
코드

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam local start-api -t .aws-sam/build/template.yaml
```

Mounting VideoUploader at <http://127.0.0.1:3000/upload/url>

[GET, OPTIONS]

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam local start-api -t .aws-sam/build/template.yaml
```

Mounting VideoUploader at <http://127.0.0.1:3000/upload/url>

[GET, OPTIONS]

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ sam local start-api -t .aws-sam/build/template.yaml
```

Mounting VideoUploader at <http://127.0.0.1:3000/upload/url>

[GET, OPTIONS]



내장된 Flask 프레임워크 통해 생성된
엔드포인트

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/apigw/local_apigw_service.py에서 생성되는 Flask 객체

```
def create(self):  
    self._app = Flask(  
        __name__,  
        static_url_path="",  
        static_folder=self.static_dir,  
    )  
    ...
```


Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/apigw/local_apigw_service.py에서 생성되는 Flask 객체

```
def create(self):
```

```
    self._app = Flask(
```

→ 내부적으로 Flask 객체 생성

```
        __name__,
```

```
        static_url_path="",
```

```
        static_folder=self.static_dir,
```

```
    )
```

```
    ...
```

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ curl http://localhost:3000/video/upload -X GET
```

```
Invoking src.main.lambda_handler (python3.9)
```

```
Building image ...
```

```
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
```

Zip 패키지 유형의 영상 업로드 애플리케이션

```
$ curl http://localhost:3000/video/upload -X GET
```

Invoking src.main.lambda_handler (python3.9)

Building image ...

Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.



AWS Lambda 이미지 생성

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



실제 소스 코드



내부

통신

볼륨 마운트

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



실제 소스 코드



내부

통신

볼륨 마운트

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



실제 소스 코드



내부

통신

볼륨 마운트

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator

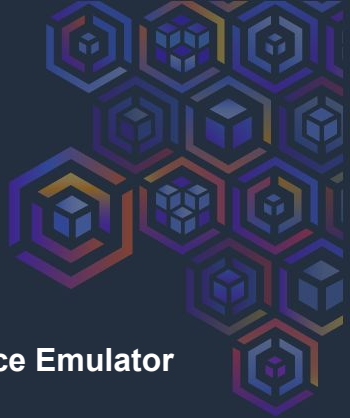


실제 소스 코드

RIC

Runtime Interface Client

볼륨 마운트



Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



실제 소스 코드



내부

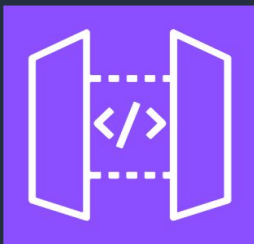
통신

블록 마운트

Zip 패키지 유형의 영상 업로드 애플리케이션



User



API Gateway



Lambda

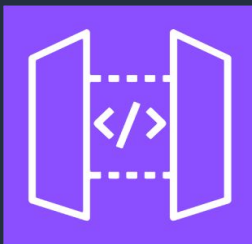


Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



API Gateway



Lambda

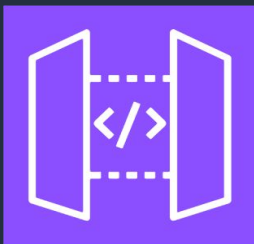


Function

Zip 패키지 유형의 영상 업로드 애플리케이션



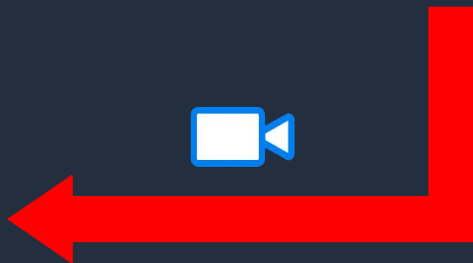
User



API Gateway



Lambda

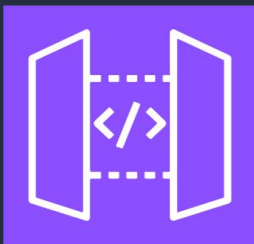


Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



API Gateway



Lambda



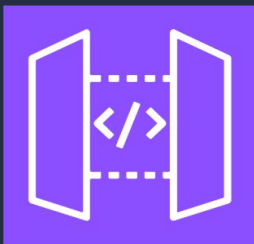
Function



Zip 패키지 유형의 영상 업로드 애플리케이션



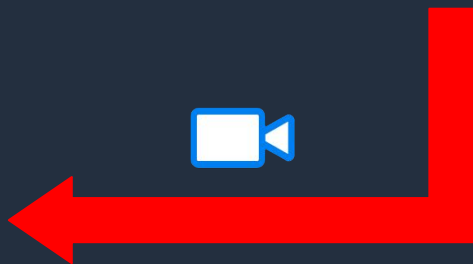
User



API Gateway



Lambda



Runtime API



Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



API Gateway



RIE



RIC



Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



Flask



RIE



Function

RIC

Zip 패키지 유형의 영상 업로드 애플리케이션



User



Flask



RIE



RIC



Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



Flask



RIE



RIC



Function

Zip 패키지 유형의 영상 업로드 애플리케이션



User



Flask



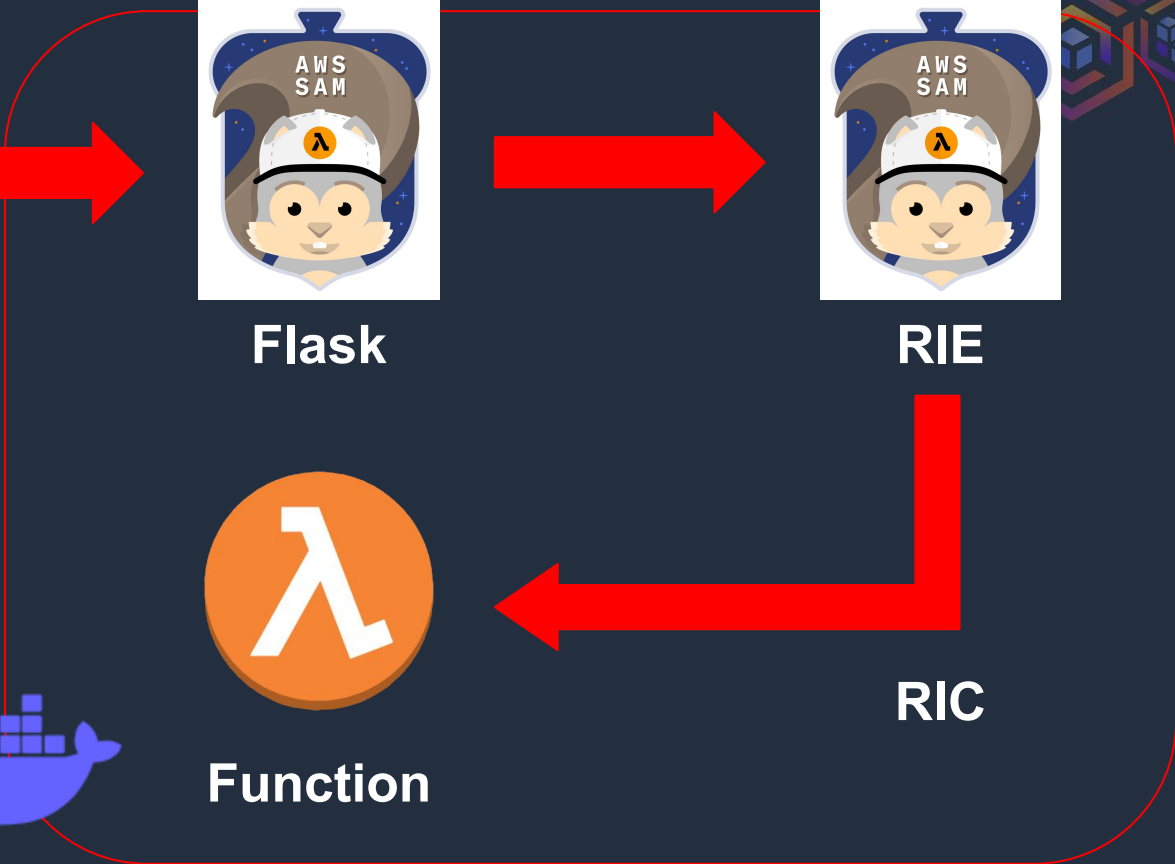
RIE



RIC



Function





RIE(Runtime Interface Emulator)

: AWS Lambda 서비스를 모방한 실행 파일



RIE(Runtime Interface Emulator)


: AWS Lambda 서비스를 모방한 실행 파일

RIC(Runtime Interface Client)


: AWS Lambda 서비스와 함수가 통신하는 API

RIC(Runtime Interface Client)

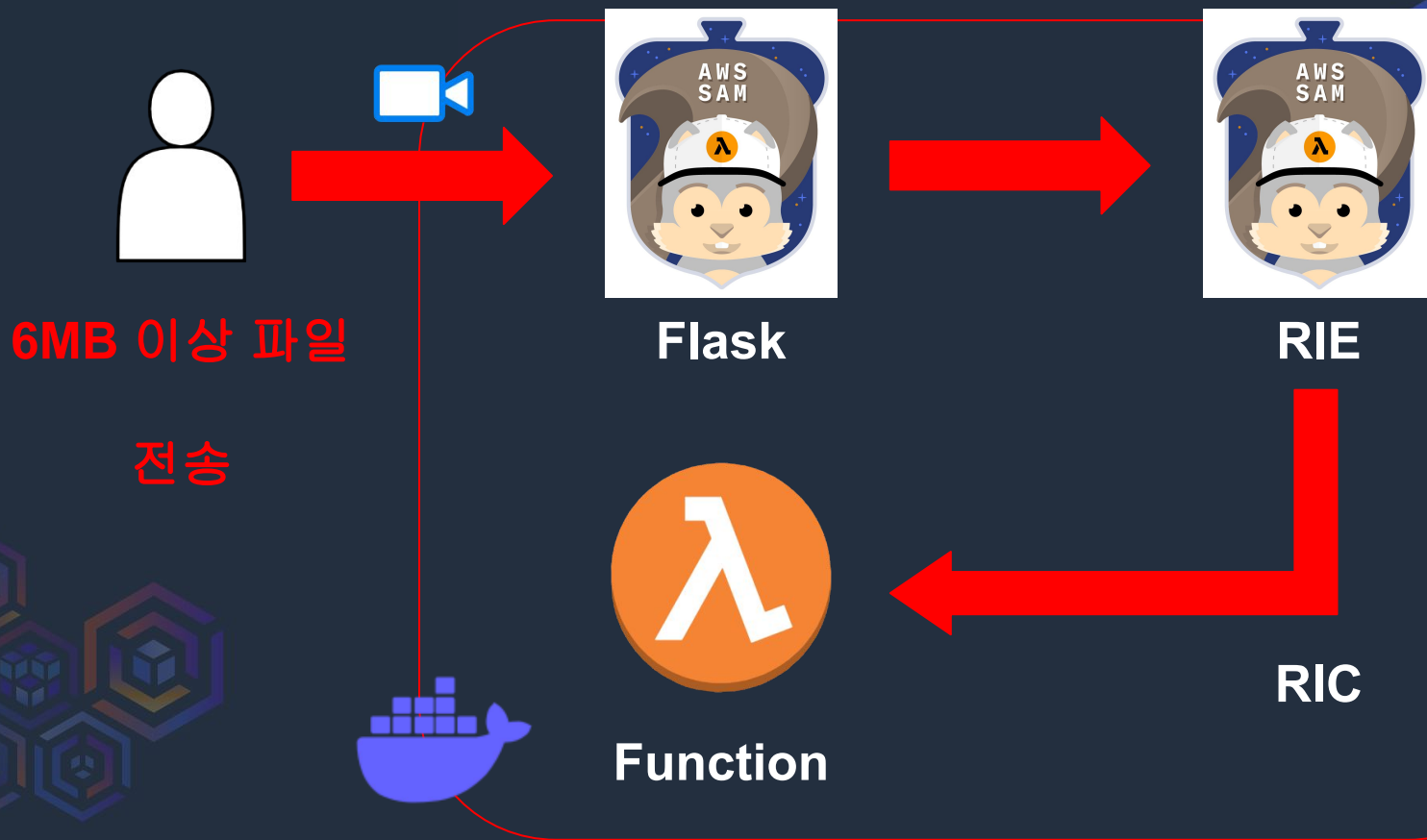
: AWS Lambda 서비스와 함수가 통신하는 API



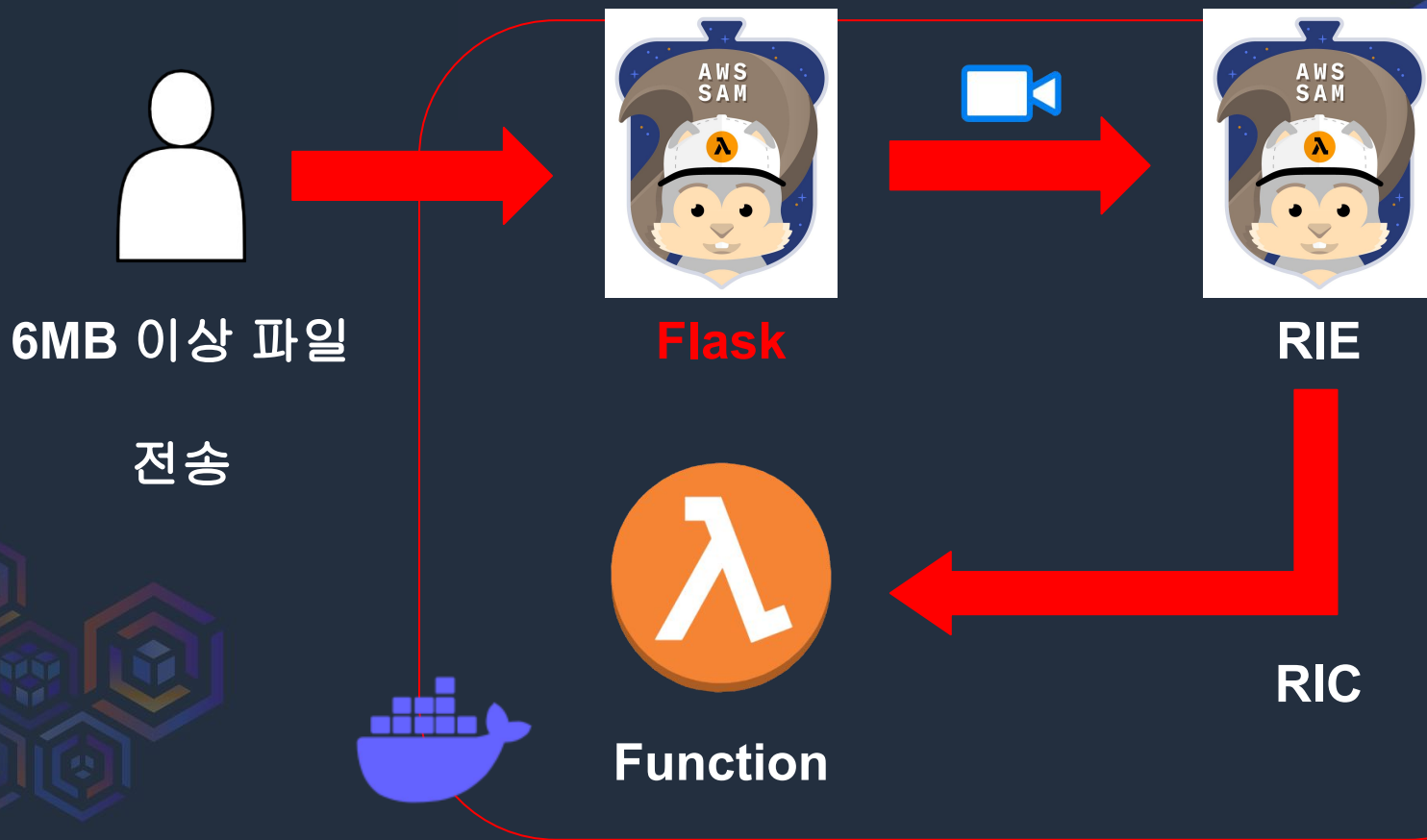
6MB 이상의 데이터 전송 때 주의해야 할 사항



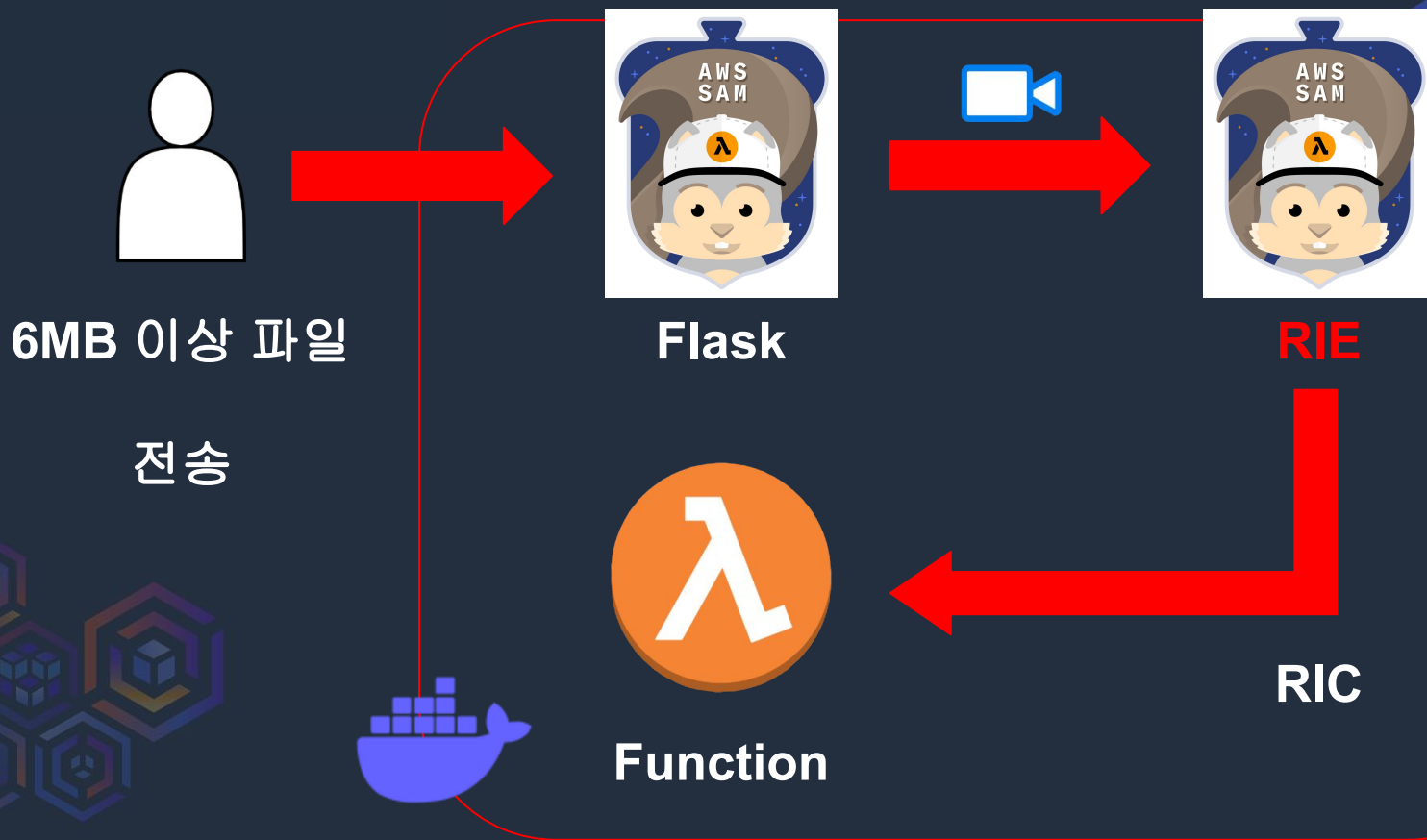
Zip 패키지 유형의 영상 업로드 애플리케이션



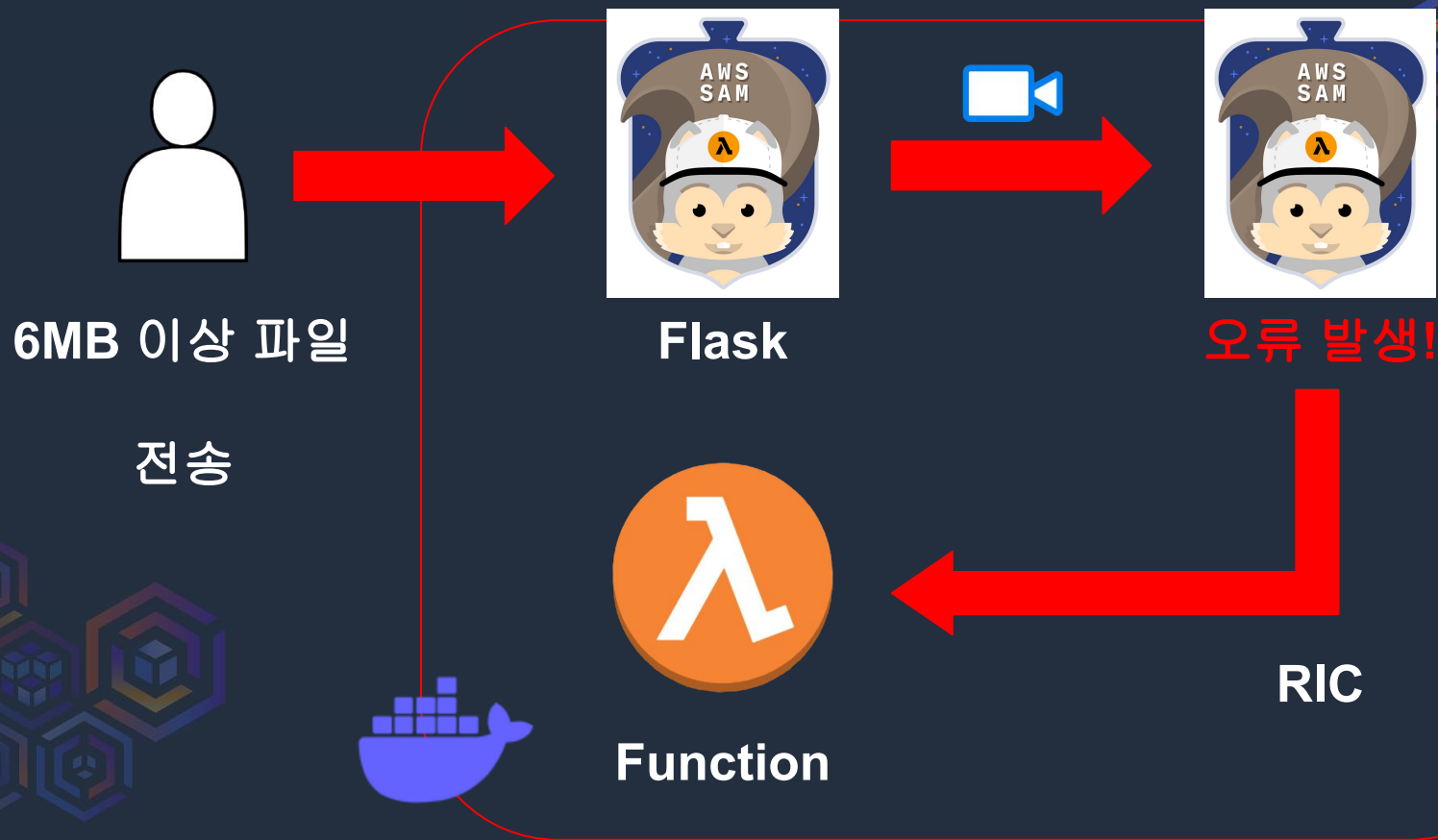
Zip 패키지 유형의 영상 업로드 애플리케이션




Zip 패키지 유형의 영상 업로드 애플리케이션




Zip 패키지 유형의 영상 업로드 애플리케이션





실제로 RIE에서 페이로드 크기를 다루는 방법



Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 `lambda/interop/model.go` 파일에서 최대 페이로드 크기 정의 부분

```
const (  
    MaxPayloadSize = 6*1024*1024 + 100 // 6 MiB + 100 bytes  
    ...  
)
```

Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 `lambda/interop/model.go` 파일에서 최대 페이로드 크기 정의 부분

```
const (  
    MaxPayloadSize = 6*1024*1024 + 100 // 6 MiB + 100 bytes  
    ...  
)
```

Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 `lambda/rip/rendering.go` 파일에서 페이로드를 자르는 부분

```
func (s *InvokeRenderer) bufferInvokeRequest() error {  
    ...  
    if nil == s.requestBuffer {  
        reader := io.LimitReader(  
            s.invoke.Payload,  
            interop.MaxPayloadSize  
        )  
        ...  
    }  
}
```

Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 `lambda/rip/rendering.go` 파일에서 페이로드를 자르는 부분

```
func (s *InvokeRenderer) bufferInvokeRequest() error {
```

```
...
```

```
if nil == s.requestBuffer {
```

```
    reader := io.LimitReader(  
        s.invoke.Payload,  
        interop.MaxPayloadSize
```

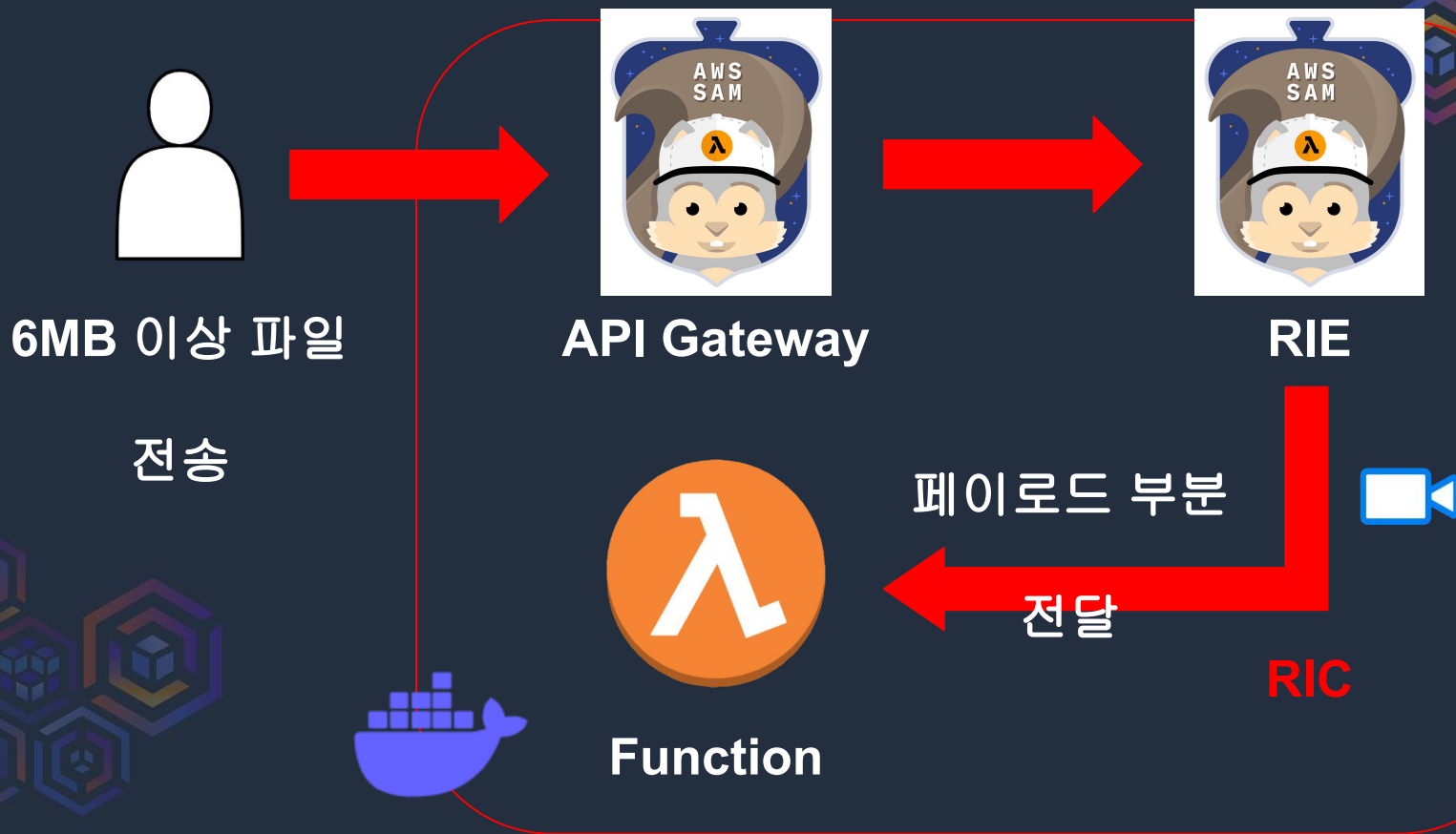
```
...
```

```
}
```

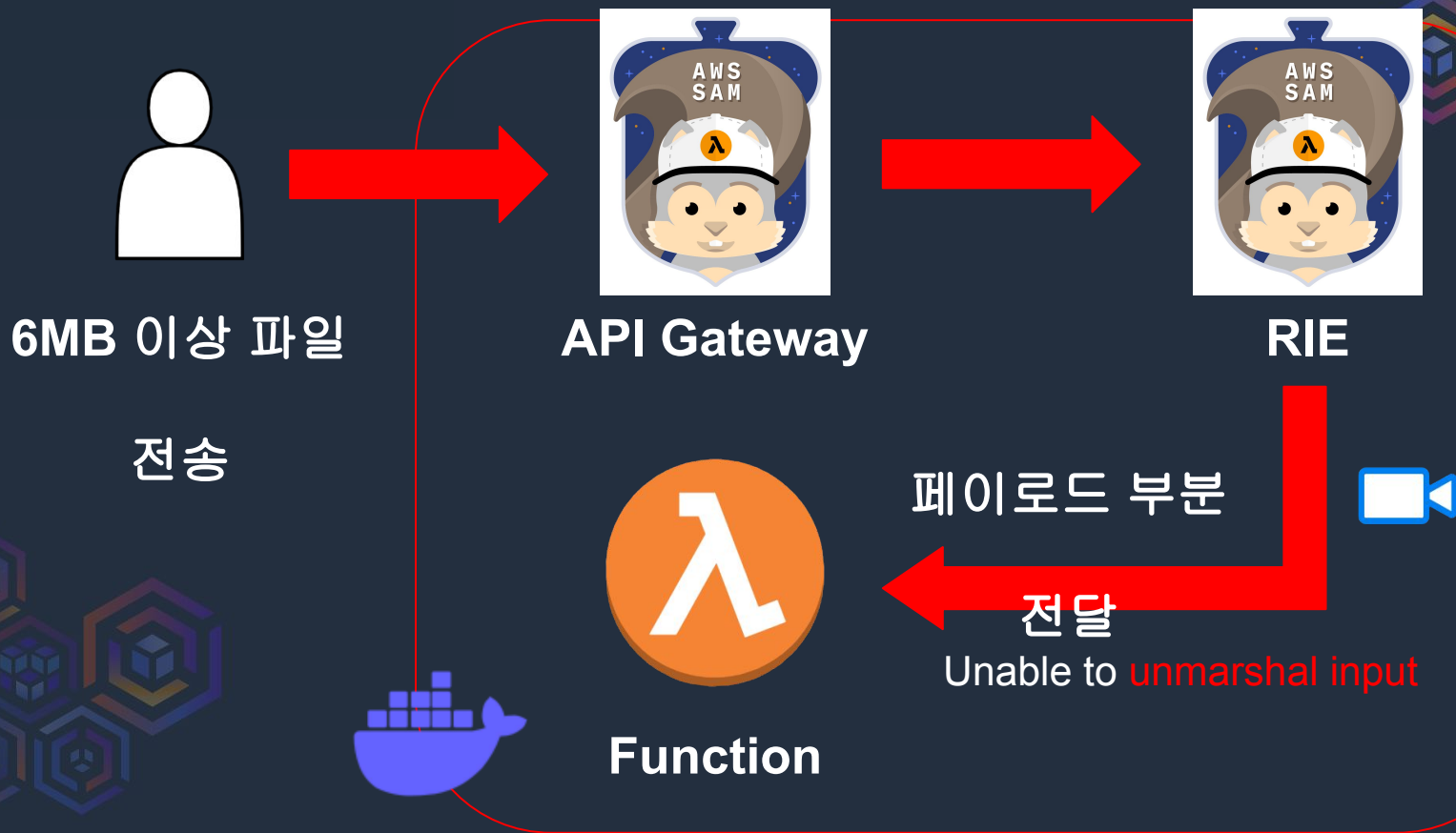


LimitReader 메서드를 통해
MaxPayloadSize를 넘지 않게
페이로드를 자르는 부분

Zip 패키지 유형의 영상 업로드 애플리케이션



Zip 패키지 유형의 영상 업로드 애플리케이션



RIC에서 오류를 처리하는 방법

Zip 패키지 유형의 영상 업로드 애플리케이션

RIC를 구성하는 `lambda_runtime_unmarshaller.py` 파일에서 오류를 반환하는 부분

```
def unmarshal_request(...):  
    ...  
    try:  
        return json.loads(request)  
    except Exception as e:  
        raise FaultException(  
            "Unable to unmarshal input: {}".format(str(e)),  
        )
```

Zip 패키지 유형의 영상 업로드 애플리케이션

RIC를 구성하는 `lambda_runtime_unmarshaller.py` 파일에서 오류를 반환하는 부분

```
def unmarshal_request(...):
```

```
...
```

```
try:
```

```
    return json.loads(request)
```

```
except Exception as e:
```


```
    raise FaultException(
```

```
        "Unable to unmarshal input: {}".format(str(e)),
```


```
    )
```



일부만 전달된 페이로드로 인해
역직렬화 과정에서 오류 발생



오류 메시지에 unmarshal 존재할 경우
페이로드 크기 의심 필요



정확히 6MB면 문제가 없을까?

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/apigw/event_constructor.py에서 실행되는 Base64 인코딩

```
def construct_v2_event_http(...) -> Dict[str, Any]:  
    ...  
    if is_base_64:  
        request_data = base64.b64encode(request_data)  
    ...
```


Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/apigw/event_constructor.py에서 실행되는 Base64 인코딩

```
def construct_v2_event_http(...) -> Dict[str, Any]:
```

```
...
```

```
if is_base_64:
```

```
    request_data = base64.b64encode(request_data)
```

```
...
```



바이너리 데이터를 문자열로 다루기 위한 **Base64**
인코딩

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 `local/apigw/event_constructor.py`에서 실행되는 Base64 인코딩

```
def construct_v2_event_http(...) -> Dict[str, Any]:
```

```
...
```

```
if is_base_64:
```

```
    request_data = base64.b64encode(request_data)
```

```
...
```

- 바이너리 데이터를 문자열로 다루기 위한 **Base64** 인코딩
- 6 Bit 단위로 잘라서 8 Bit 단위 **ASCII** 문자로 변환

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 `local/apigw/event_constructor.py`에서 실행되는 Base64 인코딩

```
def construct_v2_event_http(...) -> Dict[str, Any]:
```


```
...
```

```
if is_base_64:
```


```
    request_data = base64.b64encode(request_data)
```

```
...
```

- 바이너리 데이터를 문자열로 다루기 위한 **Base64** 인코딩
- 6 Bit 단위로 잘라서 8 Bit 단위 **ASCII** 문자로 변환
- 기존 파일 용량보다 증가



바이너리 데이터로 이루어진 페이로드의
경우 Base64 인코딩 고려 필요



Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



실제 소스 코드

내부
통신

볼륨 마운트

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지 → rapid-arm64

AWS Lambda 에뮬레이터 → RIE
Runtime Interface Emulator



실제 소스 코드

↕
내부
통신

볼륨 마운트

Docker 컨테이너 이미지가 만들어지는 방법



Zip 패키지 유형의 영상 업로드 애플리케이션



AWS SAM 내장 `local/docker/lambda_image.py`가 생성하는 Dockerfile 파일

```
FROM public.ecr.aws/lambda/python:3.9-x86_64
```

```
ADD aws-lambda-rie-x86_64 /var/rapid/
```



Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 `local/docker/lambda_image.py`가 생성하는 Dockerfile 파일

```
FROM public.ecr.aws/lambda/python:3.9-x86_64
ADD aws-lambda-rie-x86_64 /var/rapid/
```



**RIC가 포함된
Lambda 베이스 이미지
사용**

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 `local/docker/lambda_image.py`가 생성하는 Dockerfile 파일

```
FROM public.ecr.aws/lambda/python:3.9-x86_64
```

```
ADD aws-lambda-rie-x86_64 /var/rapid/
```



RIE 추가

Zip 패키지 유형의 영상 업로드 애플리케이션



User



Flask

통신



RIE



RIC



Function



AWS SAM이 내부적으로 RIE를 사용하는 방법

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/docker/container.py에서 이루어지는 RIE 통신

```
def wait_for_http_response(self, name, event, stdout) -> ...:
    resp = requests.post(
        self.URL.format(...),
        data=event.encode("utf-8"),
        ...
    )
    ...
```

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/docker/container.py에서 이루어지는 RIE 통신

```
def wait_for_http_response(self, name, event, stdout) -> ...:
```

```
    resp = requests.post(
```

```
        self.URL.format(...),
```

```
        data=event.encode("utf-8"),
```

```
        ...
```

```
    )
```

```
    ...
```

모방된 API Gateway에서 호출하는 RIE
엔드포인트

<http://localhost:9000/2015-03-31/functions/...>

Zip 패키지 유형의 영상 업로드 애플리케이션

AWS SAM 내장 local/docker/container.py에서 이루어지는 RIE 통신

```
def wait_for_http_response(self, name, event, stdout) -> ...:
```

```
    resp = requests.post(
```

```
        self.URL.format(...),
```

```
        data=event.encode("utf-8"),
```

```
        ...
```

```
    )
```

```
    ...
```

모방된 API Gateway에서 호출하는 RIE
엔드포인트

<http://localhost:9000/2015-03-31/functions/...>

Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 cmd/aws-lambda-rie/http.go 파일에서 HTTP 통신 생성 부분

```
func startHTTPServer(...) {  
    ...  
    http.HandleFunc(  
        "/2015-03-31/functions/function/invocations",  
        func(w http.ResponseWriter, r *http.Request) {...}  
    )  
    ...  
}
```


Zip 패키지 유형의 영상 업로드 애플리케이션

RIE를 구성하는 cmd/aws-lambda-rie/http.go 파일에서 HTTP 통신 생성 부분

```
func startHTTPServer(...) {  
    ...  
    http.HandleFunc(  
        "/2015-03-31/functions/function/invocations",  
        func(w http.ResponseWriter, r *http.Request) {...}  
    )  
    ...  
}
```

Zip 패키지 유형의 영상 업로드 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator





실제 소스 코드

내부
통신





블록 마운트

Zip 패키지 유형의 소스 코드를 다루는 방법

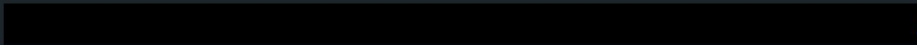

Zip 패키지 유형의 영상 업로드 애플리케이션

boring_lumiere
public.ecr.aws/lambda/python:3.9-rapid-x86_64
4643371cbe87 
[8753:8080](#) 

STATUS
Running (7 seconds ago)





LogsInspectBind mountsExecFilesStats





| Source (Host) | Destination (Container) |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
|  /app/.aws-sam/build/VideoUploader  | /var/task |

런타임 시점에 마운트된 실제 소스 코드

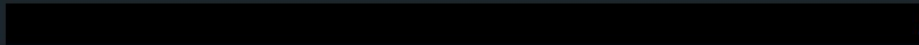

Zip 패키지 유형의 영상 업로드 애플리케이션

boring_lumiere
public.ecr.aws/lambda/python:3.9-rapid-x86_64
4643371cbe87 
[8753:8080](#) 

STATUS
Running (7 seconds ago)



LogsInspectBind mountsExecFilesStats

| Source (Host) | Destination (Container) |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
|  /app/.aws-sam/build/VideoUploader  | /var/task |



컨테이너 내 소스 코드의
위치

Docker 컨테이너 이미지의 디렉터리 구조

Zip 패키지 유형의 영상 업로드 애플리케이션

Docker 컨테이너 이미지의 var 디렉토리 구조

var

└─ task

└─ boto3

└─ src

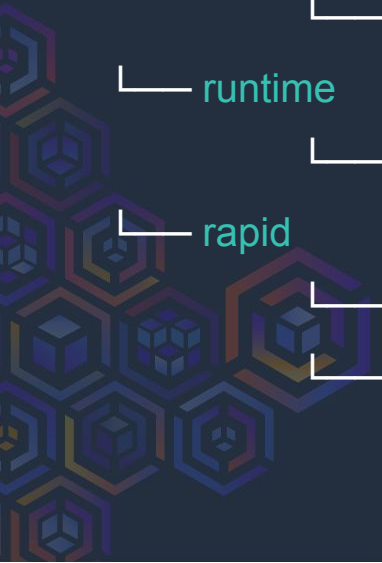
└─ runtime

└─ awslambdarc

└─ rapid

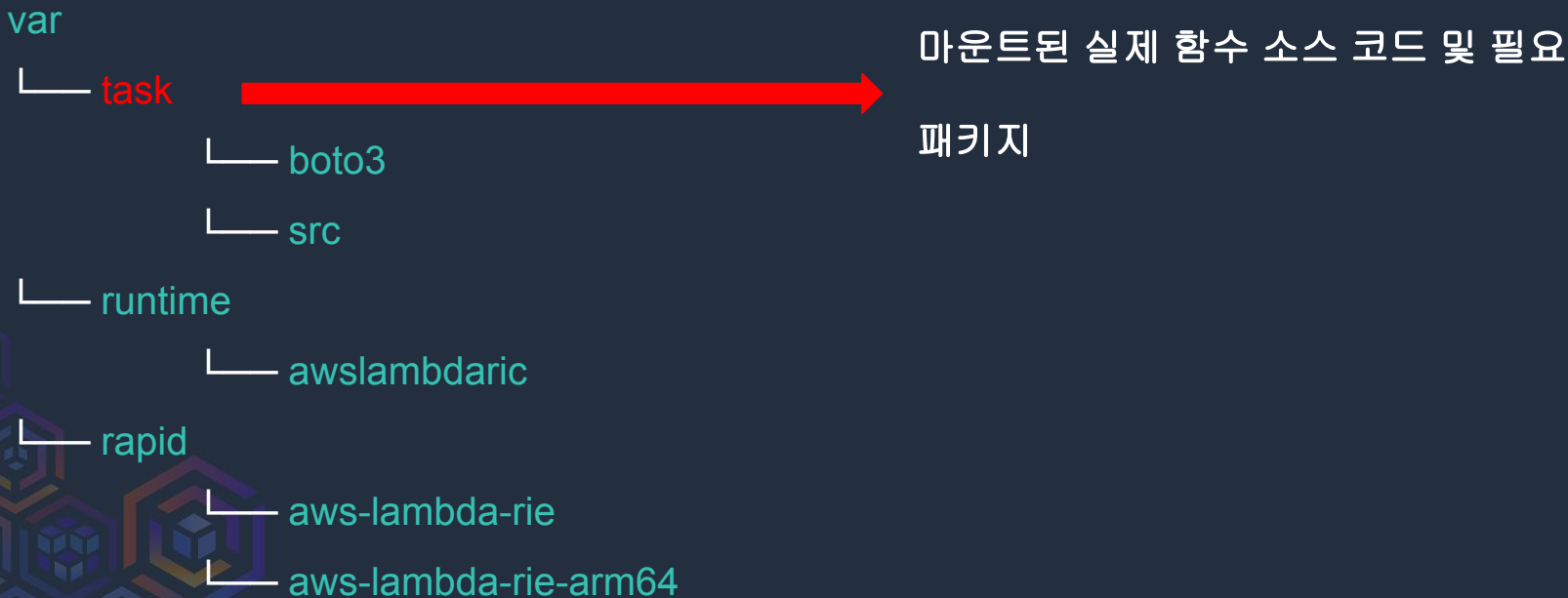
└─ aws-lambda-rie

└─ aws-lambda-rie-arm64



Zip 패키지 유형의 영상 업로드 애플리케이션

Docker 컨테이너 이미지의 var 디렉토리 구조



Zip 패키지 유형의 영상 업로드 애플리케이션

Docker 컨테이너 이미지의 var 디렉토리 구조

var

└─ task

└─ boto3

└─ src

└─ runtime



RIC 관련 패키지

└─ awslambdatic

└─ rapid

└─ aws-lambda-rie

└─ aws-lambda-rie-arm64

Zip 패키지 유형의 영상 업로드 애플리케이션

Docker 컨테이너 이미지의 var 디렉토리 구조

var

└─ task

└─ boto3

└─ src

└─ runtime

└─ awslambdarc

└─ rapid

└─ aws-lambda-rie

└─ aws-lambda-rie-arm64

RIE 실행 파일

AWS SAM CLI 명령어에 따른 정리

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

```
build
├── VideoUploader
│   ├── boto3
│   └── src
└── template.yaml
```

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

```
build
├── VideoUploader
│   ├── boto3
│   └── src
└── template.yaml
```



패키지와 함께 함수 소스 코드

생성

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

```
build
├── VideoUploader
│   ├── boto3
│   └── src
└── template.yaml
```

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**



API Gateway 모방을 위한
웹 애플리케이션 생성 및 실행

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

```
build
├── VideoUploader
│   ├── boto3
│   └── src
└── template.yaml
```

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**



런타임 시점에 동적으로
가짜 Lambda 환경 생성

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

```
build
├── VideoUploader
│   ├── boto3
│   └── src
└── template.yaml
```

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**



런타임 시점에 동적으로
가짜 Lambda 환경 생성

런타임 시점에 소스 코드 변화를 감지하는 방법

Zip 패키지 유형의 영상 업로드 애플리케이션

CodeUri 지정

Properties:

PackageType: Zip

CodeUri: ../app

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**

Zip 패키지 유형의 영상 업로드 애플리케이션

CodeUri 지정

Properties:

PackageType: Zip

CodeUri: ../app

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**



-t .aws-sam/build/template.yaml 대신
-t template.yaml 사용

Zip 패키지 유형의 영상 업로드 애플리케이션

CodeUri 지정

Properties:

PackageType: Zip

CodeUri: ../app

sam local start-api

Flask 객체 생성

curl

**Docker 컨테이너
생성**

매번 빌드하지 않고 변화

감지



Image 패키지 유형의 영상 이미지 생성 애플리케이션



프로젝트 개요

: 영상 이미지 생성 애플리케이션



프로젝트 개요

: 영상 이미지 생성 애플리케이션 로컬 환경



AWS SAM

이벤트 트리거



Lambda

이미지 업로드



LocalStack S3

프로젝트 개요

: 영상 이미지 생성 애플리케이션 로컬 환경



Lambda 함수 정의



AWS SAM 템플릿 정의

AWS Lambda 함수 정의



Image 패키지 유형의 영상 이미지 생성 애플리케이션

template.yaml 중 Properties와 Metadata

Metadata:

Dockerfile: `./dockerfiles/Dockerfile`

DockerContext: `../`

Properties:

Runtime: `python3.9`

PackageType: `Image`

Image 패키지 유형의 영상 이미지 생성 애플리케이션

template.yaml 중 Properties와 Metadata

Metadata:

Dockerfile: ./dockerfiles/Dockerfile

DockerContext: ../



Dockerfile 파일의 경로 및
이미지 빌드 컨텍스트 경로

Properties:

Runtime: python3.9

PackageType: Image

Dockerfile 파일 정의



Image 패키지 유형의 영상 이미지 생성 애플리케이션

Dockerfile

```
FROM public.ecr.aws/lambda/python:3.9
```

```
...
```

```
CMD [ "main.lambda_handler" ]
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

Dockerfile

```
FROM public.ecr.aws/lambda/python:3.9
```

```
...
```

```
CMD [ "main.lambda_handler" ]
```

→ AWS에서 제공하는 Lambda 베이스

이미지

→ Lambda 핸들러 선언

Image 패키지 유형의 영상 이미지 생성 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



Docker Image

내부
통신

사용자가 정의하고 빌드한
이미지

Image 패키지 유형의 영상 이미지 생성 애플리케이션



Docker 컨테이너 이미지

rapid-arm64

AWS Lambda 에뮬레이터

RIE

Runtime Interface Emulator



Docker Image

내부

통신

사용자가 정의하고 빌드한

이미지

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 `local/docker/lambda_image.py`가 생성하는 Dockerfile 파일

```
FROM thumbnailgenerator:latest
```

```
ADD aws-lambda-rie-x86_64 /var/rapid/
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py가 생성하는 Dockerfile 파일

```
FROM thumbnailgenerator:latest  
ADD aws-lambda-rie-x86_64 /var/rapid/
```



사용자가 만든 이미지를 베이스로
사용

Image 패키지 유형의 영상 이미지 생성 애플리케이션


AWS SAM 내장 local/docker/lambda_image.py가 생성하는 Dockerfile 파일

```
FROM thumbnailgenerator:latest
```


```
ADD aws-lambda-rie-x86_64 /var/rapid/
```



RIE 추가



왜 S3는 따로 템플릿에 정의하지 않는
걸까?



프로젝트 개요

: 영상 이미지 생성 애플리케이션 로컬 환경



AWS SAM

이벤트 트리거



Lambda

이미지 업로드



LocalStack S3

LocalStack을 통해 생성된 S3

로컬에서의 AWS Lambda 실행

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam build -t template.yaml
```

Building image for ThumbnailGenerator function

...

Successfully tagged thumbnailgenerator:latest

Build Succeeded

Built Artifacts : .aws-sam/build

Built Template : .aws-sam/build/template.yaml

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam build -t template.yaml
```

Building image for ThumbnailGenerator function

...

Successfully tagged thumbnailgenerator:latest

Build Succeeded

Built Artifacts : .aws-sam/build

Built Template : .aws-sam/build/template.yaml

→ 컨테이너 이미지 생성

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local invoke -e event.json --force-image-build -t ...
```

```
Invoking src.main.lambda_handler (python3.9)
```

```
Building image ...
```

```
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local invoke -e event.json --force-image-build -t ...
```

→ .aws-sam 디렉터리

```
Invoking src.main.lambda_handler (python3.9)
```

```
Building image ...
```

```
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local invoke -e event.json --force-image-build -t ...
```

➡ S3 업로드 이벤트

```
Invoking src.main.lambda_handler (python3.9)
```

```
Building image ...
```

```
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local generate event s3 put
```

```
{  
  "Records": [{  
    "s3": {  
      "bucket": { "name": "help-me-sam" }  
      "object": { "key": "test.mp4" }  
    }  
  }]  
}
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local generate event s3 put
```

```
{  
  "Records": [{  
    "s3": {  
      "bucket": { "name": "help-me-sam" }  
      "object": { "key": "test.mp4" }  
    }  
  }  
}
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local generate event s3 put
```

```
{  
  "Records": [{  
    "s3": {  
      "bucket": { "name": "help-me-sam" }  
      "object": { "key": "test.mp4" }  
    }  
  }  
}
```

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

함수에 대한 도커
컨테이너 이미지
생성

sam invoke

정의된 이벤트를
바탕으로 함수 실행

Zip 패키지 유형의 영상 업로드 애플리케이션

sam build

함수에 대한 도커
컨테이너 이미지
생성

sam invoke

정의된 이벤트를
바탕으로 함수 실행



가짜 Lambda 환경 생성

Image 패키지 유형의 영상 이미지 생성 애플리케이션

```
$ sam local invoke -e event.json --force-image-build -t ...
```

```
Invoking src.main.lambda_handler (python3.9)
```

```
Building image ...
```

```
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.
```

어째서 빌드를 강제화 하지?

Image 패키지 유형의 영상 이미지 생성 애플리케이션



Docker 컨테이너 이미지

AWS Lambda 에뮬레이터



컨테이너 이미지로

빌드된 함수 소스 코드



변경
발생

Image 패키지 유형의 영상 이미지 생성 애플리케이션



Docker 컨테이너 이미지



전체 이미지 재빌드 진행으로
예상

AWS Lambda 에뮬레이터

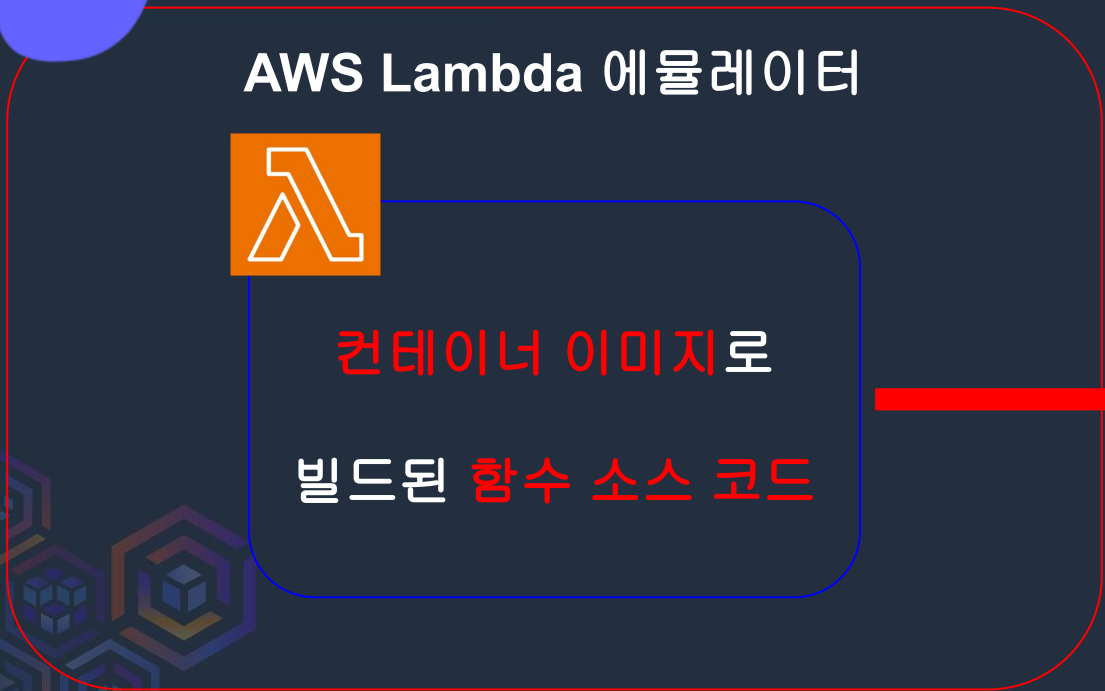


컨테이너 이미지로

빌드된 함수 소스 코드



변경
발생



내부적으로 이미지 빌드 조건을 다루는 방법

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
if image:
    self.skip_pull_image = True
...
try:
    self.docker_client.images.get(rapid_image)
    self._check_base_image_is_current(base_image)
...

```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
if image:
    self.skip_pull_image = True
...
try:
    self.docker_client.images.get(rapid_image)
    self._check_base_image_is_current(base_image)
...
```

패키지 유형이 이미지일 경우
skip_pull_image 값 True 할당

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
if image:
    self.skip_pull_image = True
...
try:
    self.docker_client.images.get(rapid_image)
    self._check_base_image_is_current(base_image)
...

```

베이스 이미지 변경

확인

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
def _check_base_image_is_current(self, image_name: str) -> None:
    if self.skip_pull_image or self.force_image_build:
        return

    if self.is_base_image_current(image_name):
        self.skip_pull_image = True
    else:
        self.force_image_build = True
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
def _check_base_image_is_current(self, image_name: str) -> None:
    if self.skip_pull_image or self.force_image_build:
        return

    if self.is_base_image_current(image_name):
        self.skip_pull_image = True
    else:
        self.force_image_build = True
```

→ 베이스 이미지 변경이 있을 경우
재빌드 플래그 설정

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
def _check_base_image_is_current(self, image_name: str) -> None:
    if self.skip_pull_image or self.force_image_build:
        return

    if self.is_base_image_current(image_name):
        self.skip_pull_image = True
    else:
        self.force_image_build = True
```

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
def _check_base_image_is_current(self, image_name: str) -> None:
```

```
    if self.skip_pull_image or self.force_image_build:
```

```
        return
```



앞서 skip_pull_image 값이 True 할당 되었기
때문에 베이스 이미지 변경 여부 검사 생략

Image 패키지 유형의 영상 이미지 생성 애플리케이션

AWS SAM 내장 local/docker/lambda_image.py

```
if (  
    self.force_image_build  
    or  
    ...  
):  
    self._build_image(...)
```

재빌드 플래그가 없기 때문에 빌드
생략



COMMUNITY DAY

SEOUL

결론



결론

- CloudFormation **YAML 문법을 활용해 Lambda 함수를 정의**할 수 있어요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE을 통해 로컬에서 Lambda 함수를 실행할 수 있어요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.
- SAM은 내부적으로 Flask 프레임워크를 사용해 API Gateway를 모방해요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.
- SAM은 내부적으로 Flask 프레임워크를 사용해 API Gateway를 모방해요.
- Lambda 함수가 수용할 수 있는 **페이로드 크기의 한계와 오류 메시지에 유의**해요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.
- SAM은 내부적으로 Flask 프레임워크를 사용해 API Gateway를 모방해요.
- Lambda 함수가 수용할 수 있는 페이로드 크기의 한계와 오류 메시지에 유의해요.
- Lambda 베이스 이미지를 활용하여 Dockerfile 파일로 Lambda 함수를 만들 수 있어요.

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.
- SAM은 내부적으로 Flask 프레임워크를 사용해 API Gateway를 모방해요.
- Lambda 함수가 수용할 수 있는 페이로드 크기의 한계와 오류 메시지에 유의해요.
- Lambda 베이스 이미지를 활용하여 Dockerfile 파일로 Lambda 함수를 만들 수 있어요.
- **JSON 객체 형식의 S3 이벤트를 정의**하고 이를 통해 Lambda 함수를 실행할 수

결론

- CloudFormation YAML 문법을 활용해 Lambda 함수를 정의할 수 있어요.
- SAM을 활용하여 RIE를 통해 로컬에서 Lambda 함수를 실행할 수 있어요.
- LocalStack을 활용하여 로컬 환경에 S3를 구축하고 사용할 수 있어요.
- SAM은 내부적으로 Flask 프레임워크를 사용해 API Gateway를 모방해요.
- Lambda 함수가 수용할 수 있는 페이로드 크기의 한계와 오류 메시지에 유의해요.
- Lambda 베이스 이미지를 활용하여 Dockerfile 파일로 Lambda 함수를 만들 수 있어요.
- JSON 객체 형식의 S3 이벤트를 정의하고 이를 통해 Lambda 함수를 실행할 수

결론

- CloudFormation **YAML 문법을 활용해 Lambda 함수를 정의**할 수 있어요.
- SAM을 활용하여 **RIE을 통해 로컬에서 Lambda 함수를 실행**할 수 있어요.
- **LocalStack을 활용하여 로컬 환경에 S3를 구축**하고 사용할 수 있어요.
- SAM은 내부적으로 **Flask 프레임워크를 사용해 API Gateway를 모방**해요.
- Lambda 함수가 수용할 수 있는 **페이로드 크기의 한계와 오류 메시지에 유의**해요.
- **Lambda 베이스 이미지를 활용하여 Dockerfile 파일로 Lambda 함수를 만들 수**
있어요.
- **JSON 객체 형식의 S3 이벤트를 정의**하고 이를 통해 Lambda 함수를 실행할 수