AUSG

쌤(SAM)! 도와주세요!

@7기 이태현

https://github.com/0417taehyun/help-me-sam

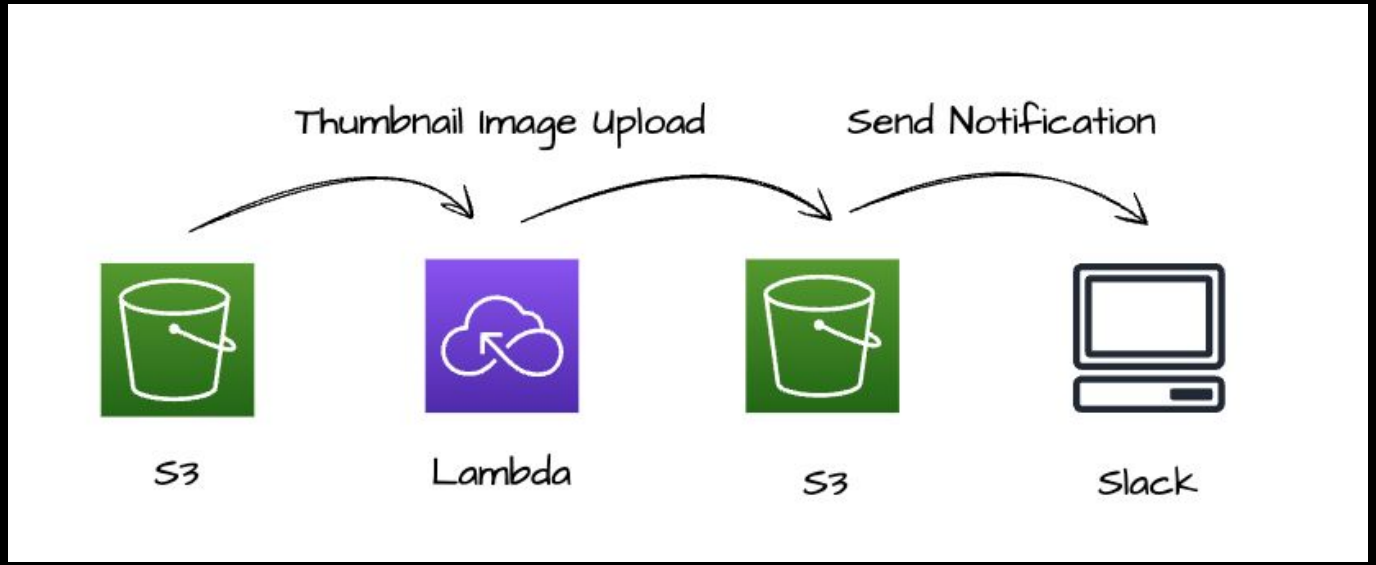
- AWS 쌤(SAM)! 도와주세요!

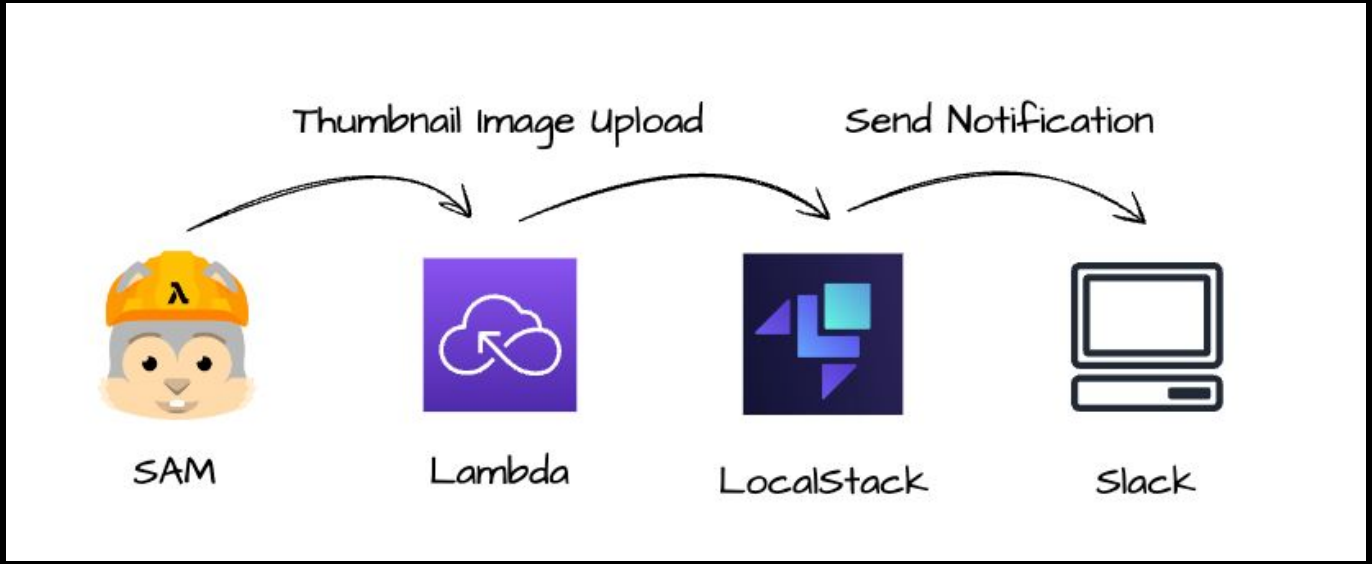
- AWS 쌤(SAM)! 도와주세요!
- AUSG 쌤! 도와주세요!

AUSG

SAM

Serverless Application Model


Thumbnail Image Upload
Send Notification
S3
Lambda
S3
Slack


Thumbnail Image Upload
Send Notification
SAM
Lambda
LocalStack
Slack

- 단위 테스트 (Unit Test)
- 통합 테스트 (Integration Test)

AWS SAM을 언제 사용해야 할까?

- 단위 테스트 (Unit Test)
- 통합 테스트 (Integration Test)

AUSG

템플릿

환경 변수

이벤트

```
$ brew install aws-sam-cli
$ sam init

Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Data processing
```

```
$ brew install aws-sam-cli
$ sam init

Which template source would you like to use?
    1 - AWS Quick Start Templates
    2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Data processing
```

```
$ tree .

├── README.md
├── events
│   └── event.json
├── hello_world
│   ├── __init__.py
│   ├── app.py
│   └── requirements.txt
├── samconfig.toml
└── template.yaml
```

```
$ tree .

├── README.md
├── events
│   └── event.json
├── hello_world
│   ├── __init__.py
│   ├── app.py
│   └── requirements.txt
├── samconfig.toml
└── template.yaml
```

template.yaml

```
Transform: AWS::Serverless-2016-10-31
Description: 영상 썸네일 추출 애플리케이션
```

```
Resources:
    ThumbnailGenerator:
        Type: AWS::Serverless::Function

    Metadata:
        Dockerfile: ./dockerfiles/Dockerfile
        DockerContext: ../
```

```
Properties:
    Architectures:
        - arm64
    FunctionName: ThumbnailGenerator
    Runtime: python:3.9
    PackageType: Image
    Timeout: 600
    MemorySize: 1024
    Environment:
        Variables:
            LEVEL: LEVEL
```

```
Properties:
    Architectures:
        - arm64
    FunctionName: ThumbnailGenerator
    Runtime: python:3.9
    PackageType: Image
    Timeout: 600
    MemorySize: 1024
    Environment:
        Variables:
            LEVEL: LEVEL
```

Makefile

```
architecture ?= x86

.PHONY: build
build:
    @poetry run sam build \
        -t ../templates/template.local.$(architecture).yaml
```

.aws-sam/template.yaml

```yaml
Metadata:
    DockerContext: help-me-sam/thumbnail-generator
    Dockerfile: ./dockerfiles/Dockerfile
    SamResourceId: ThumbnailGenerator
Properties:
    ImageUri: thumbnailgenerator:latest
```

```
.PHONY: invoke
invoke:
    @poetry run sam local invoke \
        -t .aws-sam/build/template.yaml \
        -e ../events/event.json \
        --env-vars ../environments/env.local.json \
```

event.json

```
{
    "Records": [
        "s3": {
            "bucket": {"name": "example-bucket"},
            "object": {"key": "test.mp4"}
        }
    ]
}
```

```
$ sam local generate-event -h
```

Examples:

Generate event S3 sends to local Lambda function:

```
$sam local generate-event s3 [put/delete]
```

env.json

```
{
    "Parameters": {
        "AWS_REGION": "YOUR_AWS_REGION",
        "AWS_ACCESS_KEY_ID": "YOUR_AWS_ACCESS_KEY_ID",
        "AWS_SECRET_ACCESS_KEY": "YOUR_AWS_SECRET_ACCESS_EY",
        "AWS_ENDPOINT_URL": "YOUR_AWS_ENDPOINT_URL",
    }
}
```

AUSG

로컬 환경에서
AWS 리소스를 어떻게 구축할까?

docker-compose.yaml

```yaml
s3-local:
    image: localstack/localstack
    container_name: localstack
    ports:
        - "4566:4566"
    environment:
        - SERVICES=s3
        - DEFAULT_REGION=ap-northeast-2
        - DATA_DIR=/tmp/localstack/data
```

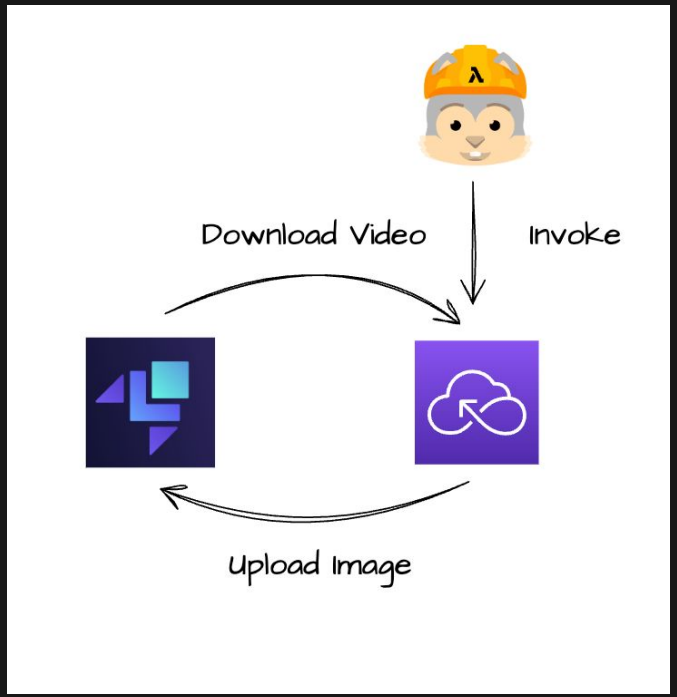
```
aws-cli:
    build:
        context: ..
        dockerfile: dockerfiles/Dockerfile.local
    container_name: aws-cli
    environment:
        AWS_DEFAULT_REGION: ap-northeast-2
        AWS_ACCESS_KEY_ID: test
        AWS_SECRET_ACCESS_KEY: test
    depends_on:
        - s3-local
```

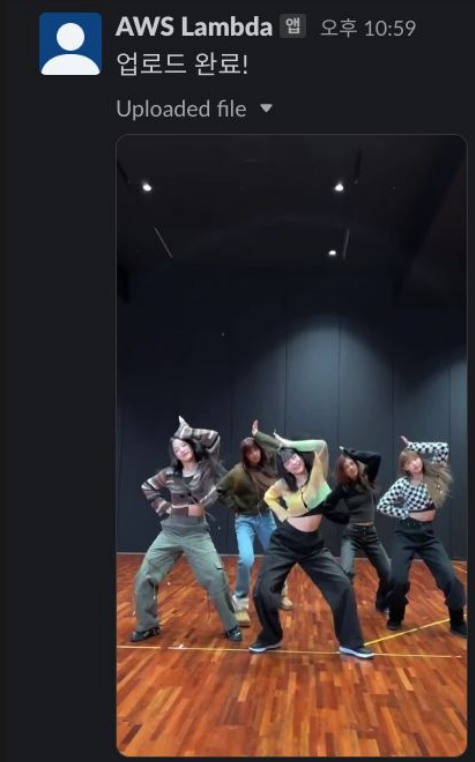
local_aws_s3.setup.sh

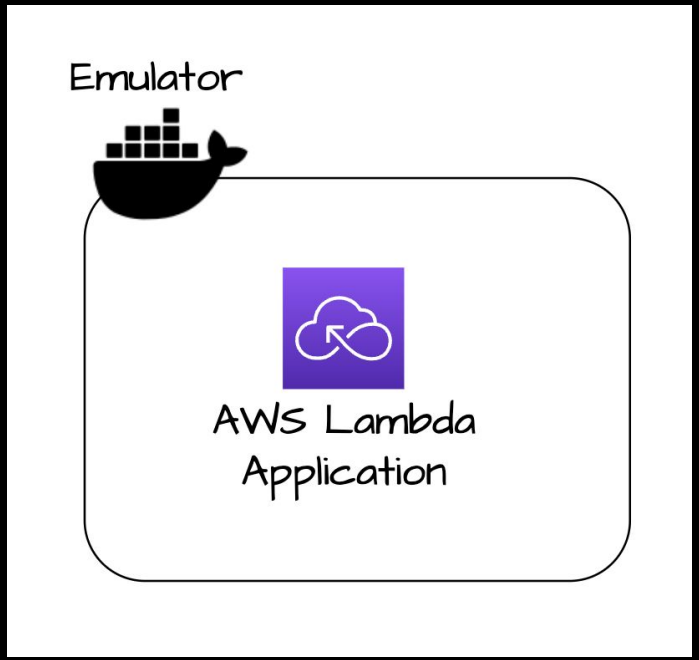
```
#!/bin/sh
aws s3 mb s3://example-bucket \
    --endpoint-url=http://localstack:4566

aws s3 cp /data/test.mp4 s3://example-bucket/test.mp4 \
    --endpoint-url=http://localstack:4566
```

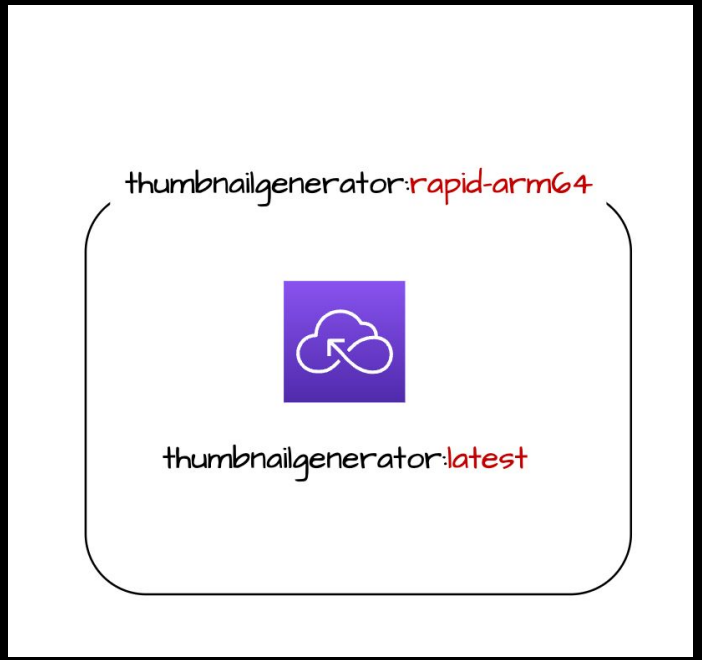
local_aws_s3.setup.sh

```
#!/bin/sh
aws s3 mb s3://example-bucket \
    --endpoint-url=http://localstack:4566

aws s3 cp /data/test.mp4 s3://example-bucket/test.mp4 \
    --endpoint-url=http://localstack:4566
```


Download Video
Invoke
Upload Image


AWS Lambda 앱 오후 10:59
업로드 완료!
Uploaded file


Emulator
AWS Lambda
Application


thumbnailgenerator:rapid-arm64
thumbnailgenerator:latest

local/docker/lambda_image.py

```
FROM thumbnailgenerator:latest
ADD aws-lambda-rie-x86_64 /var/rapid/
RUN mv /var/rapid/aws-lambda-rie-x86_64 /var/rapid/aws-lambda-rie \
    && chmod +x /var/rapid/aws-lambda-rie
```

```
if (
    self.force_image_build
    or image_not_found
    or any(
        layer.is_defined_within_template for layer
        in downloaded_layers
    )
    or not runtime
):
    self._build_image(...)
```

```
if (
    self.force_image_build
    or image_not_found
    or any(
        layer.is_defined_within_template for layer
        in downloaded_layers
    )
    or not runtime
):
    self._build_image(...)
```

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

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

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

```python
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
```

```
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
```

왜 비교를 건너 뛰는 거지?

```
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
```

```
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
```

```python
def is_base_image_current(self, image_name: str) -> bool:
    return (
        self.get_local_image_digest(image_name)
        ==
        self.get_remote_image_digest(image_name)
    )
```

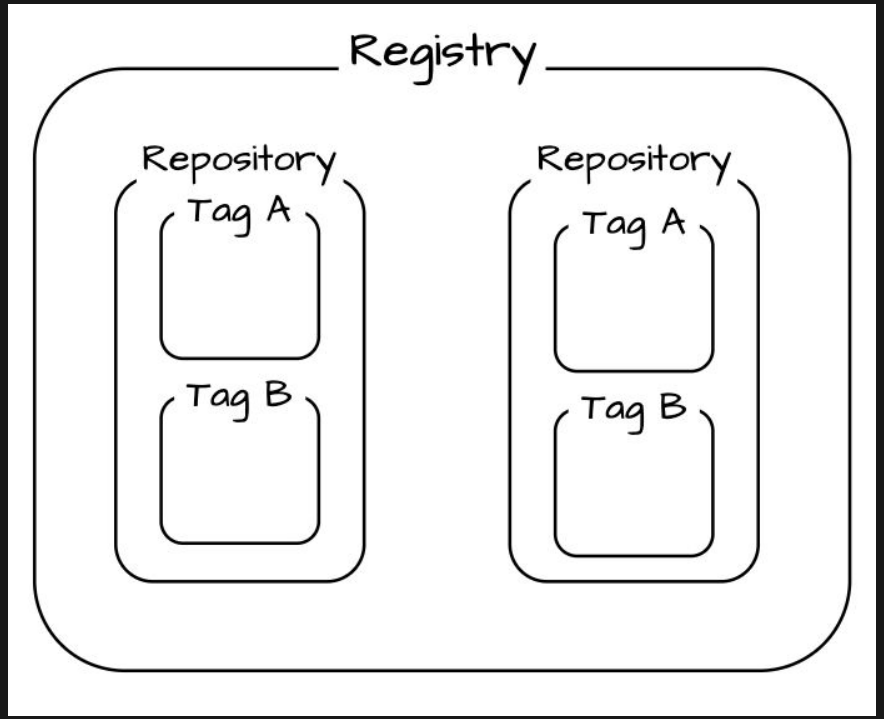
```python
def is_base_image_current(self, image_name: str) -> bool:
    return (
        self.get_local_image_digest(image_name)
        ==
        self.get_remote_image_digest(image_name)
    )
```

```python
def get_local_image_digest(self, image_name: str) -> Optional[str]:
    image_info = self.docker_client.images.get(image_name)
    full_digest: str = image_info.attrs.get("RepoDigests", [None])[0]
    try:
        return full_digest.split("@")[1]
    except (AttributeError, IndexError):
        return None
```

docker/api/image.py

```
def inspect_image(self, image):
    return self._result(...)
```

```python
def get_local_image_digest(self, image_name: str) -> Optional[str]:
    image_info = self.docker_client.images.get(image_name)
    full_digest: str = image_info.attrs.get("RepoDigests", [None])[0]
    try:
        return full_digest.split("@")[1]
    except (AttributeError, IndexError):
        return None
```



```python
def is_base_image_current(self, image_name: str) -> bool:
    return (
        self.get_local_image_digest(image_name)
        ==
        self.get_remote_image_digest(image_name)
    )
```

docker/api/image.py

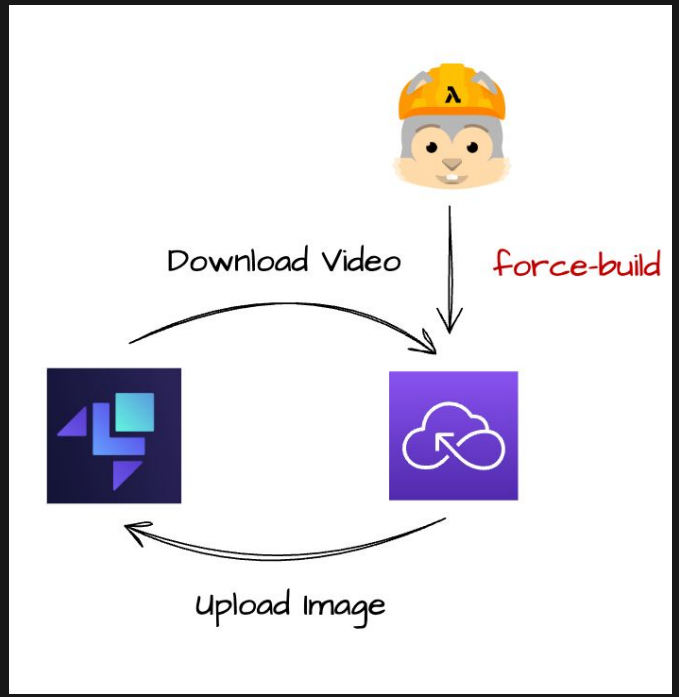
```
def inspect_distribution(self, image, auth_config=None):
    registry, _ = auth.resolve_repository_name(image)
    ...
    if auth_config is None:
        header = auth.get_config_header(self, registry)
    ...
    url = self._url("/distribution/{0}/json", image)

    return self._result(...)
```

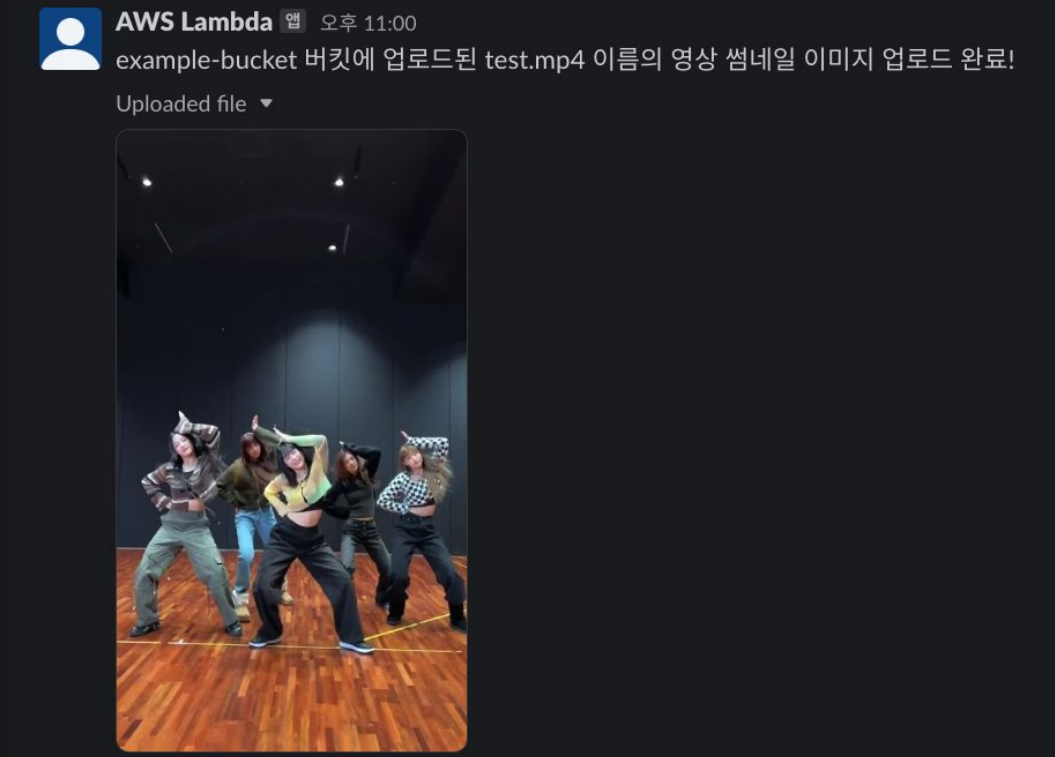
docker/api/image.py

```
def inspect_distribution(self, image, auth_config=None):
    registry, _ = auth.resolve_repository_name(image)
    ...
    if auth_config is None:
        header = auth.get_config_header(self, registry)
    ...
    url = self._url("/distribution/{0}/json", image)

    return self._result(...)
```

Makefile

```
.PHONY: invoke
invoke:
    @poetry run sam local invoke -t \
        .aws-sam/build/template.yaml \
        -e ../events/event.json \
        --env-vars ../environments/env.local.json \
        --force-image-build
```


Download Video
force-build
Upload Image


AWS Lambda 앱 오후 11:00
example-bucket 버킷에 업로드된 test.mp4 이름의 영상 썸네일 이미지 업로드 완료!
Uploaded file

AUSG

1. LocalStack은 로컬 환경에 AWS를 구축할 때 좋아요.
2. AWS SAM은 AWS Lambda를 활용한 통합 테스트 환경을 구축할 때 좋아요.
3. 로컬 환경에서 도커 이미지를 활용해 SAM을 구축할 때 주의해야 해요.