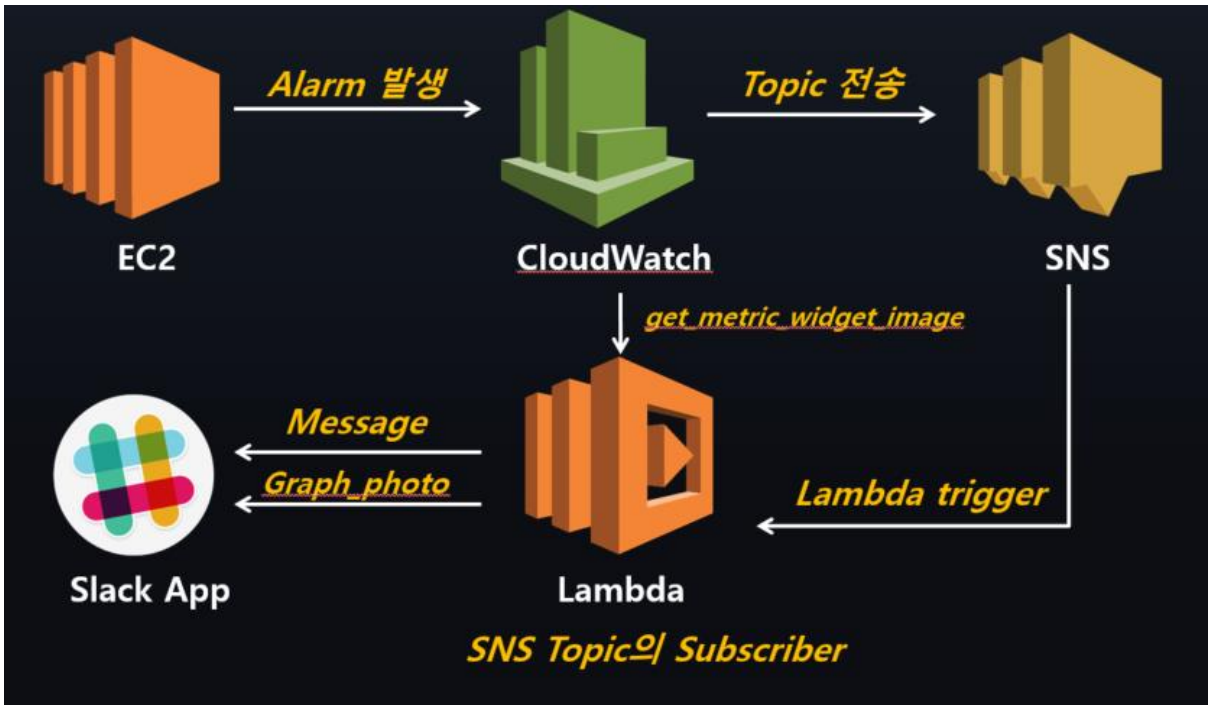
CloudWatch, SNS와 Lambda를 이용해서 Slack으로 알람받기(Python3 ver)

1. 진행방식



2. 슬랙 설정

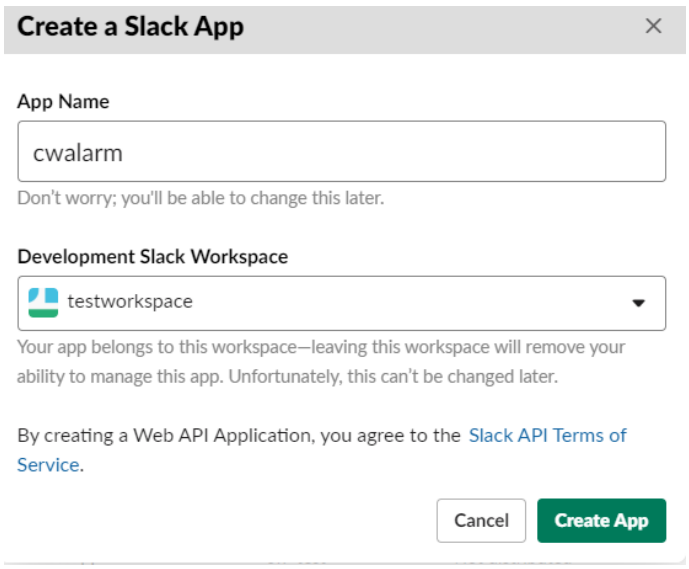
1) 새로운 workspace 생성

생성한 workspace명 : testworkspace

2) 새로운 App 생성

생성한 app명 : cwalarm

https://api.slack.com/apps -> create new app -> deploy할 workspace 연결



3) Bot 기능 추가

add features and functionality에서 bots 선택 -> OAuth & Permissions 이동

3-1) bot 토큰을 할당받기 위한 scope 할당작업 review Scopes to Add 선택

추가 할 scopes : chat : write / (chat: write.customize) / files:write

3-2) install app to workspace 선택 : 현재 생성한 app을 해당 workspace에 설치

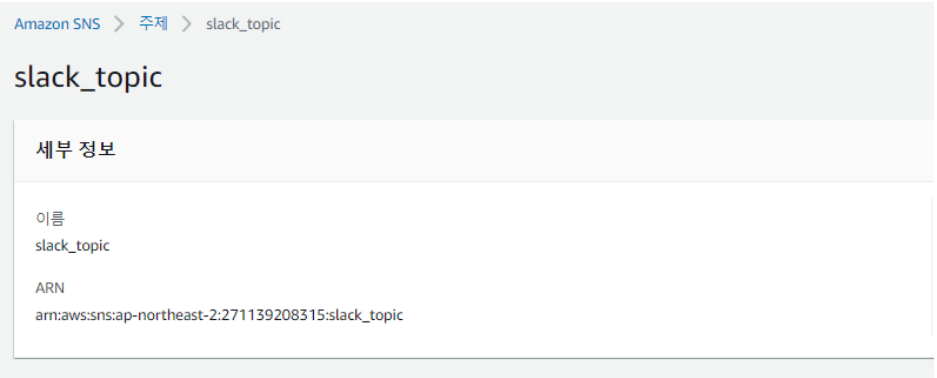
3-3) 설치가 완료 되면 bot user OAuth Access 토큰 발급(xoxob-)

해당 토큰은 이 bot을 사용하기 위한 유니크 키값으로 람다에서 사용하는 환경변수

3. AWS 설정 (텔레그램 방식과 동일)

1) SNS Topic 생성

Amazon SNS – Topic – Create Topic



2) CloudWatch 경보(alarm)생성

2-1) 인스턴스 및 지표선택

Amazon CloudWatch – Alarm – Create Alarm

알람명 : slack-cw 3t-bastion

네임스페이스 : AWS/EC2

인스턴스명 : 3T2-B

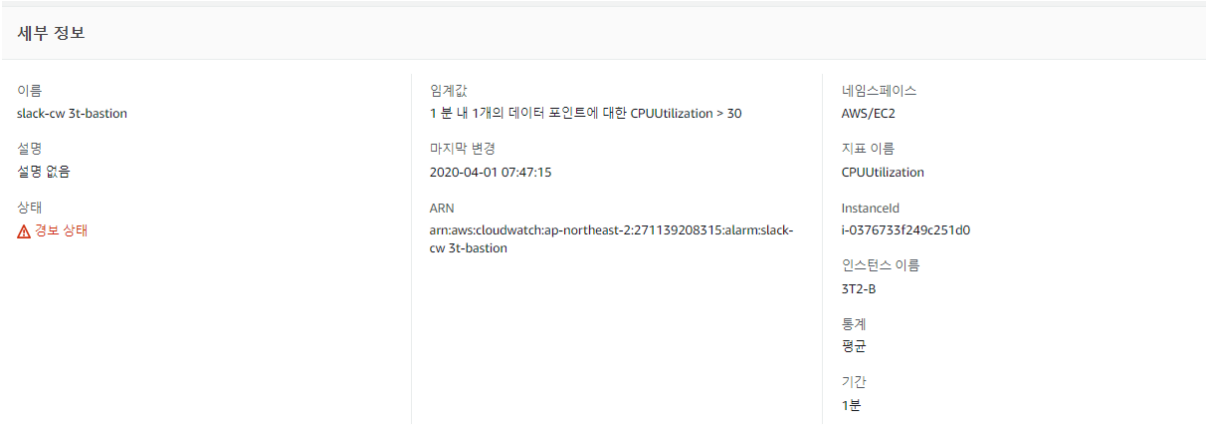
지표이름 : CPU Utilization

통계 :평균

기간 : 1분

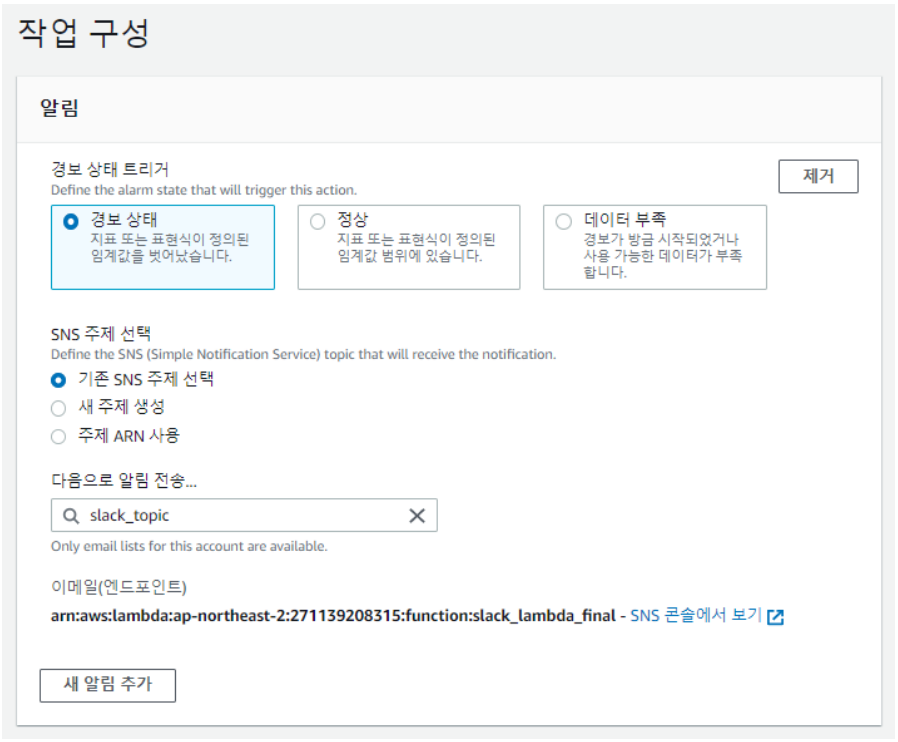
2-2) 경보알림 조건선택

CPUUtilization >= 0.01



2-3) 경보일 때, 알림 전송할 트리거 지정

SNS topic : slack_topic선택



4. Lambda 설정

4-1) Lambda 함수 생성(Python 3.8)

4-2) 전체zip 파일(py파일+packages파일)로 업로드

4-3) 환경변수 설정

CUSTOMER : 고객명

TOKEN : 슬랙에서 발급받은 bot token값(xoxob-)

UID : 슬랙에서 만든 채널명

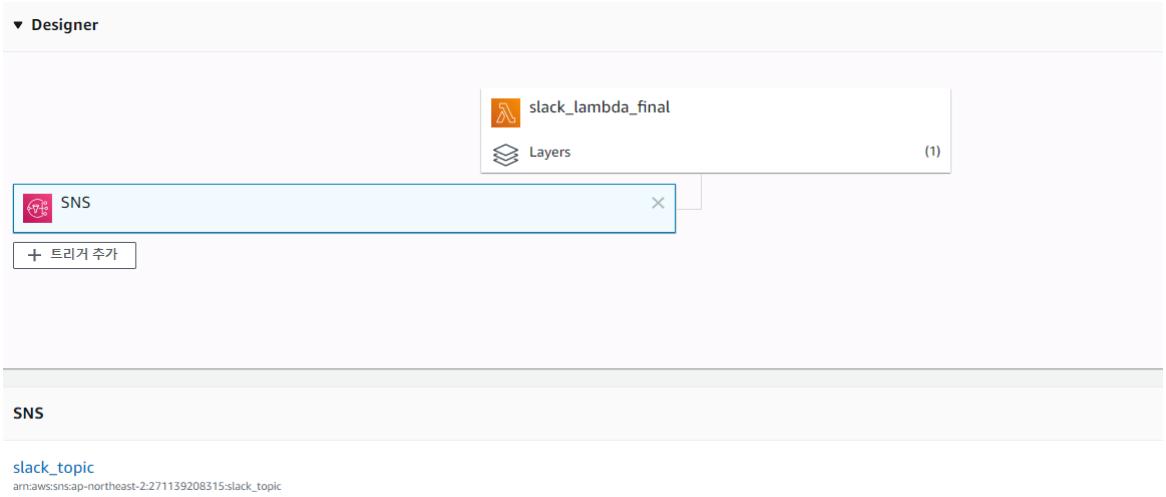
4-4) Role추가

람다를 생성할 때 만든 Role : AWSLambdaBasicExecutionRole-868cc1bc-33b2-4947-b242-0209da381ed3

람다에서 Cloudwatch 리소스에 접근하고자 하는 Role : CloudWatchReadOnlyAccess

4-5) 트리거 등록

subscribe 할 topic 선택하여 트리거 활성화



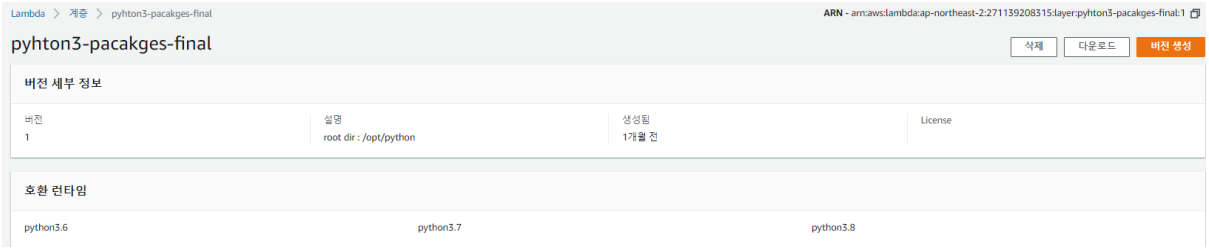
참고) packages파일을 따로 Layer로 추가하는 경우 필요한 작업

Layer : Python의 Package 모듈을 따로 import하기 위해서 사용

1) 현재 생성한 Layer : python3-packages-final

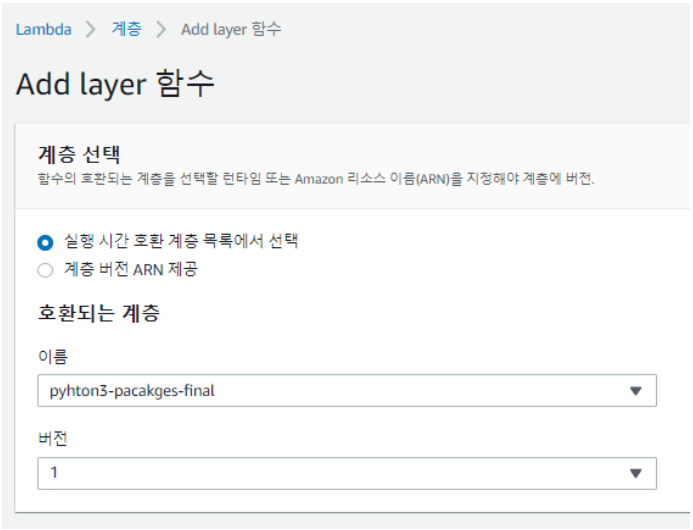
Lambda함수가 run하기 위해 필요한 모듈파일들 (request,urllib 외 5개모듈)

호환 런타임 : python 3.6/ python 3.7/ python 3.8



2) Layers 람다 함수에 적용

Add Layers 함수



Layer 또한 zip파일처럼 적용되서 내부적으로 지정된 Python의 path대로 라이브러리 import

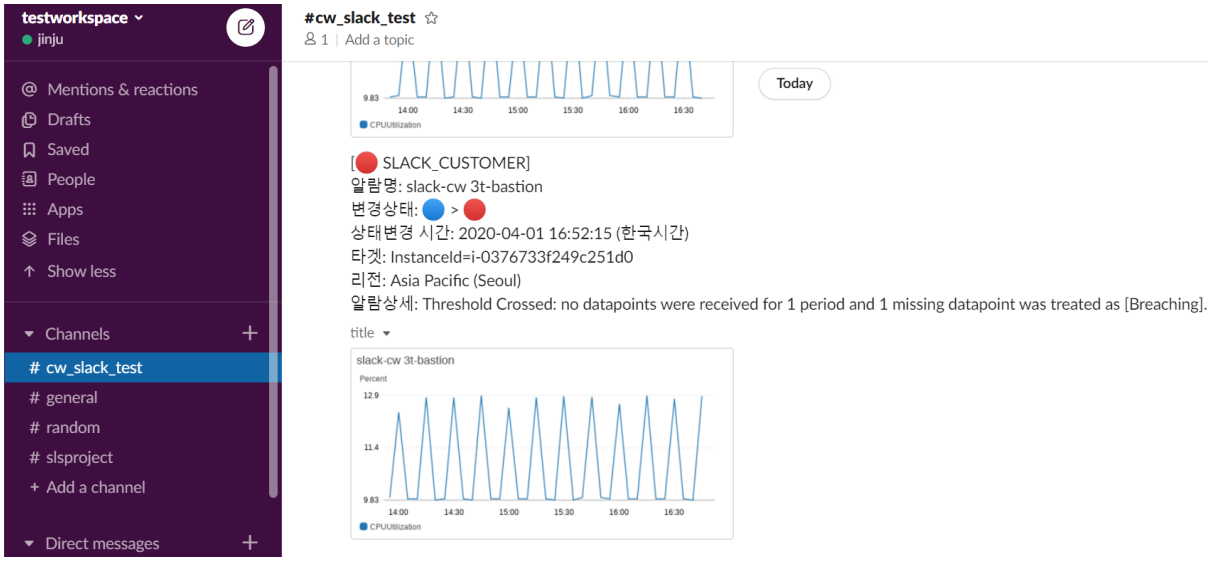
현재 layer zip파일의 상위디렉토리 /python

+ 만약 Layer의 상위 디렉토리가 없다면 패키지가 unzip되는 루트 디렉토리는 /opt 추가

sys.path.append('/opt')

```
9  sys.path.append('/opt')
10 import requests
```

5. 최종 테스트 결과(슬랙)



6. Slack API

1. incoming webhook API 방식 : Slack 자체에서 생성한 App 이 메세지를 보냄
2. WEB API 방식(텔레그램과 같은 방식) : Slack 안에 설치한 새로운 App 이 메세지를 보냄

위에서 사용한 방식은 WEB API 방식
슬랙자체에서 request 와 response 형식을 지정해 줌
request 형식 : 메소드에 따른 URL
response 형식 : webpage 에 뿌려주는 내용

사용한 method : https://api.slack.com/methods

이 중에서 사용한 method (이벤트 요구사항)
1. chat.postMessage
- documentation: https://api.slack.com/methods/chat.postMessage
Method URL : https://slack.com/api/chat.postMessage
HTTP Method : POST
Content-types : application/x-www-form-urlencoded, application/json
bot 에게 필요한 scope : chat:write

즉, 위의 URL 로 POST 요청이 들어오면(Content-types 을 header 에 넣어서), slack API 가 'message' 를 보내준다.

- test request URL : https://slack.com/api/chat.postMessage?token=xoxb-953060682726-1035855460085 wrOf9O4DKd7cjFRQtaRiIMXw&channel=cw_slack_test&text=tttttt&pretty=1
(위의 test URL 은 slack API 에서 정의한 request URL 로
https://api.slack.com/methods/chat.postMessage/test 에서 생성)

- 이때 보내는 필수 매개변수들
token : bot token 명 / channel : 채널명 / test : 보낼 메시지

- test response : test request URL 로 접속시 webpage 에 뿌려주는 내용

2. files.upload
- documentation: https://api.slack.com/methods/files.upload
Method URL : https://slack.com/api/files.upload
HTTP Method : POST
Content-tyes : application/x-www-form-urlencoded, multipart/form-data
bot 에게 필요한 scope : files:write

즉, 위의 URL 로 POST 요청이 들어오면(Content-types 을 header 에 넣어서), slack API 가 'message' 를 보내준다.

- test URL : https://slack.com/api/files.upload?token=xoxb-953060682726-1035855460085-wrOf9O4DKd7cjFRQtaRiIMXw&channels=cw_slack_test&pretty=1

(위의 test URL 은 slack API 에서 정의한 request URL 로
https://api.slack.com/methods/files.upload/test 에서 생성)

즉, 위의 URL 로 POST 요청이 들어오면, slack API 가 'files' 를 보내준다.

- 이때 보내는 필수 매개변수들
token : bot token 명 / channel : 채널명 / file : 파일 / (filename : 파일명)

- test response : test request URL 로 접속시 webpage 에 뿌려주는 내용