나만의 어학사전

201635904 컴퓨터공학과 강유민

1. 마이크로 서비스, Lambda, S3

- 마이크로 서비스 : 소프트웨어를 구축하기 위한 아키텍처이다. 애플리케이션을 상호 독립적인 최소 구성 요소로 분할하고, 모든 요소가 독립적이며 동일한 작업을 수행하기 위해 함께 작동시킨다.
  - 편리한 엑세스 : 큰 애플리케이션을 작은 조각으로 분할해서 업데이트 하며 개선 편리
  - 향상된 개방성 : 다중 언어 지원 API사용
  - 간단한 배포 : 모듈화 되어있고 작아서 배포에 따른 우려사항이 낮아짐

- AWS Lambda : 서버리스 함수
  - 완전 관리형 서비스 : 서버리스 컴퓨팅이기때문에 실행할 런타임만 정하면 이용 가능
  - 유연한 확장성 : 다른 AWS서비스들을 호출하여 자신만의 서비스 만듬
  - 고가용성
  - 유휴 용량 없음 : 요청이 올 때만 프로비저닝되어 작동하기 때문에 비용이 청구안됨.
  - 마이크로 서비스 호환성 : 독립적인 코드 단위로 개발 가능, 클라우드 환경에서 효율적

1. 마이크로 서비스, Lambda, S3

- Simple Storage Service(S3) : 클라우드 공급자가 저장 공간을 서비스로 관리하고 운영하는 클라우드 스토리지이며 객체 스토리지의 한 종류이다. 버킷을 만들고 오브젝트를 저장한 후 접근 권한을 만들고 공개한다. 주의할 점은 버킷의 이름은 중복되면 안된다.
  - 높은 내구성
  - 손쉬운 확장성 : 사용량 예측이 어려운 서비스에 효율적, 파일 수 제한 없음
  - 보안성과 편리성 : HTTPS라는 보안 프로토콜 제공, IAM을 통해 인증된 사용자만 데이터에 접근 가능, 콘솔화면만으로 관리 가능하고 버저닝 가능
  - 관리 유연성 : 스토리지 관리자는 데이터 사용 추세를 분류, 보고 및 시각화 하여 비용 줄임, 각 스토리지 사용 및 비용을 개별적으로 확인 가능
  - S3 스토리지 클래스 -S3 standard, S3-IA, S3 One-Zone-IA, 글래이셔

- 출처 : (책) 당신이 지금 알아야 할 AWS (이영호, 한동수)

2. 구현 방법 (S3, dictionary.html)

- 버킷 : parser-yuum
- 리전 : 아시아 태평양
- 퍼블릭 엑세스 차단 : 비활성
- 객체 : dictionary.html -> 정적 웹 사이트 호스팅

```
fetch ("https://rfjpjh5elc.execute-api.ap-
northeast-
2.amazonaws.com/default/parser_Lamda", {
        method : "POST",
        body : JSON.stringify({
            text : input.value
        })
    }).then(function(response){
        return response.text().then(function(text
    ) {
            result.innerHTML = text;
        }
```

람다 함수와 연결

- Post 방식으로 데이터 전송
- 사용자가 입력한 단어(ex-안녕)을 {....., "body": "{\"text\":\" 안녕\"}"}의 형식의 json으로 Lambda 함수에 전송

람다 함수가 응답한 결과를 text형식으로 받아서 html 창에 띄우기

2. 구현 방법 (Lambda, lambda_function.py)

- 람다 함수 이름 : parser_Lamda
- 런타임 : Python 3.6
- 핸들러 : lambda_function.lambda_handler
- 파이썬 라이브러리 : Beautifulsoup4
  - 크롤링을 위한 라이브러리, 람다 함수에 zip파일을 업로드 하면 됨
- API게이트 웨이 : S3 연결을 위한 endpoint 생성
- Lambda_function.py

```
s3_client = boto3.client('s3')
```

→ S3에 연결 (boto3사용)

```
response = event['body']
response = json.loads(response)
response = response['text']
```

→ S3에서 보낸 json을 response에 넣고 분해해서 검색할 단어만 추출하는 코드

```
return {    'statusCode' : 200,
            'headers' : {"Access-Control-Allow-Origin" : "*",
                         "Access-Control-Allow-Credentials" : 1        },
            'body' : json.dumps(result_list, ensure_ascii=False) }
```

S3으로 return 하는 코드
- Headers : S3에 엑세스할 수 있는 권한 부여
- Body : json형식으로 크롤링한 단어의 뜻을 리스트에 넣어서 보냄

*파이썬에서 유니코드 형식으로 보내기 때문에 ensure_ascii=False 을 넣어준다.

2. 구현 방법 (lambda_function.py-크롤링)

```
word = parse.quote(response)
```

검색할 단어를 유니코드로 바꿔서 word에 저장

```
url
=f"https://dict.naver.com/search.nhn?dicQuery={word}₩&query={word}

₩&target=dic₩&ie=utf8₩&query_utf=₩&isOnlyViewEE="
```

크롤링 할 주소를 url에 저장 : 네이버 사전 주소로 dicQuery={word}\&query={word}의 word부분에 검색할 단어를 유니코드로 넣으면 그 단어가 검색된 페이지가 나옴

```
soup = BeautifulSoup(urllib.request.urlopen(url).read(), "html.parser")
```

단어가 검색 된 페이지 전체를 크롤링해서 저장

```
a = soup.select_one("#content > div")

find_word = a.select_one("ul > li> p > a > span.c_b > strong")

result_list.append(find_word.text)

mean = a.select("ul > li> p")[1].text

mean = " ".join(mean.split())

result_list.append(mean)
```

네이버 사전 페이지를 selector로 구분해서 원하는 부분인 검색된 단어와 그 단어의 의미 부분만 뽑아서(파싱) result_list에 저장

2. 구현 방법(IAM 정책 연결)

- 람다 실행 역할 이름 : parser_Lamda-role-vd16feow
- 연결된 정책들 :
  - AWSLambdaBasicExecutionRole :
    - 관리형 정책
    - CloudWatch Logs (서비스, 제한-쓰기)
  - AWSLambdaLambdaFunctionDestinationExecutionRole :
    - 관리형 정책
    - Lambda (서비스)
  - AWSLambdaExecute :
    - AWS 관리형 정책
    - CloudWatch Logs (서비스, 제한-목록, 읽기, 쓰기)
    - S3 (서비스, 제한-읽기, 쓰기)

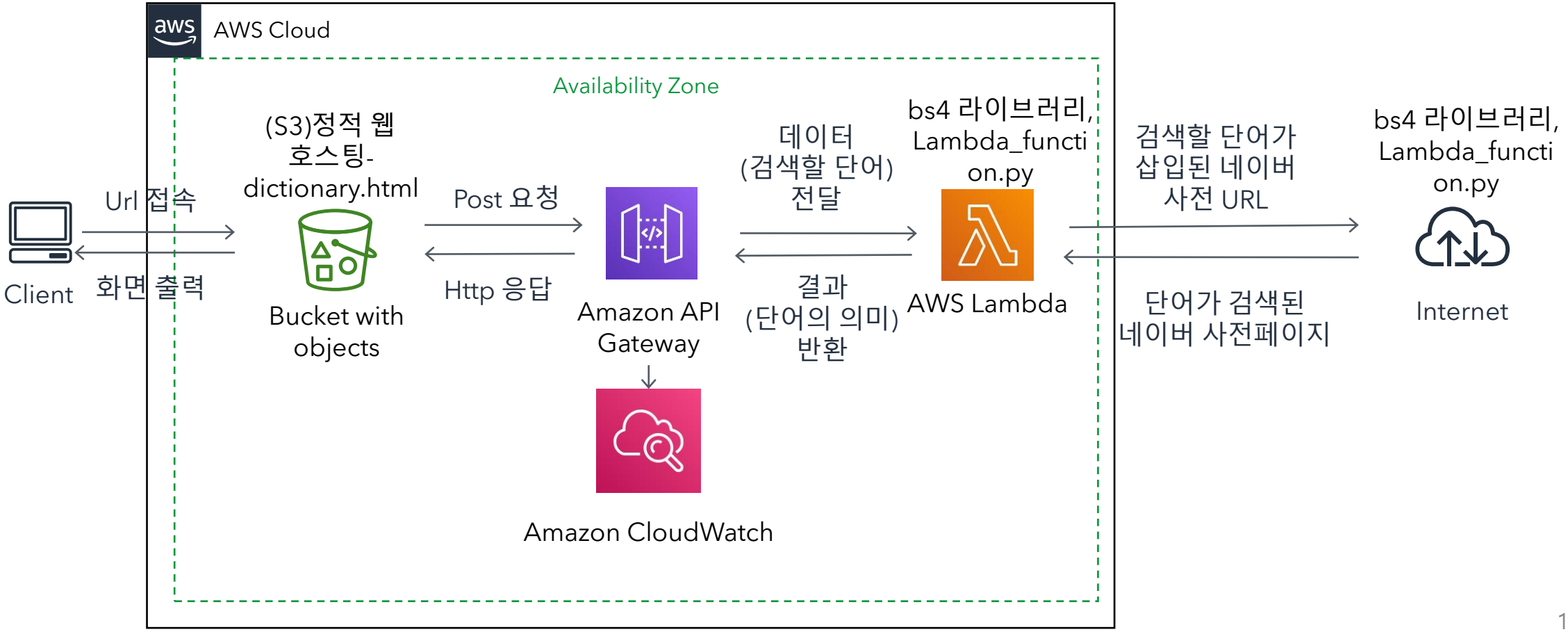
3. Lambda 전체 코드 lambda_function.py

```
# -*- coding: utf-8 -*-
import boto3
import json
import urllib.request
from urllib import parse
from bs4 import BeautifulSoup

def lambda_handler(event, context):

    s3_client = boto3.client('s3')
    response = event['body']
    response = json.loads(response)
    response = response['text']
    result_list = []
    if response == "":
        result_list.append("단어를 입력해주세요!")
    else:
        word = parse.quote(response)
        url = f"https://dict.naver.com/search.nhn?dicQuery={word}\&query={word
        }\&target=dic\&ie=utf8\&query_utf=\&isOnlyViewEE="
        soup = BeautifulSoup(urllib.request.urlopen(url).read(), "html.parser")
        a = soup.select_one("#content > div")

        if str(a['class']) == "['search_result']":
            result_list.append("없는 단어 입니다! 다시 입력해주세요")
        elif str(a['class']) == "['kr_dic_section', 'search_result', 'dic_kr_e
ntry']":
            find_word = a.select_one("ul > li> p > a > span.c_b > strong")
            result_list.append(find_word.text)
            mean = a.select("ul > li> p")[1].text
            mean = " ".join(mean.split())
            result_list.append(mean)
        else:
            find_word = a.select_one("dl > dt > a > span.c_b > strong")
            result_list.append(find_word.text)
            mean = a.select_one("dl > dd").text
            mean = " ".join(mean.split())
            result_list.append(mean)
    return {
        'statusCode' : 200,
        'headers' : {
            "Access-Control-Allow-Origin" : "*",
            "Access-Control-Allow-Credentials" : 1
        },
        'body' : json.dumps(result_list, ensure_ascii=False)
    }
```

4. S3 dictionary.html 전체 코드

```
<body><br>

<h1>◆나만의 어학사전◆</h1>

<h3>찾고 싶은 단어를 입력하세요!</h3>

<h4>모든 언어로 가능(한자 제외!)</h5>

<textarea class="container" style="width: 400px; height: 100px; f
ont-size : 40px; border: mediumpurple solid;”></textarea><br>

<button onclick=“sendRequest()” style=“font-
size: 30px; background-
color: mediumslateblue; color: white;” class=“btn btn-primary”>검
색하기</button>

<h2 class="result" style="width: 600px; border: mediumorchid dott
ed;"></h2>

</body>

<script type="text/javascript">

        var input = document.querySelector(".container")

        var result = document.querySelector(".result")

            function sendRequest() {

                result.innerHTML="로딩중입니다! 잠시만 기다려 주세요~^^"

                fetch ("https://rfjpjh5elc.execute-api.ap-northeast-
2.amazonaws.com/default/parser_Lamda", {

                    method : "POST",

                    body : JSON.stringify({

                        text : input.value

                    })

                }).then(function(response){

                    return response.text().then(function(text) {

                        result.innerHTML = text;

                    })})}

    </script>
```

5. 전체 아키텍처


AWS Cloud
Availability Zone
Client
Url 접속
화면 출력
(S3)정적 웹 호스팅-dictionary.html
Bucket with objects
Post 요청
Http 응답
Amazon API Gateway
Amazon CloudWatch
데이터 (검색할 단어) 전달
결과 (단어의 의미) 반환
bs4 라이브러리, Lambda_function.py
AWS Lambda
검색할 단어가 삽입된 네이버 사전 URL
단어가 검색된 네이버 사전페이지
bs4 라이브러리, Lambda_function.py
Internet

6. End-point 및 실행 화면

- End-point : https://parser-yuum.s3.ap-northeast-2.amazonaws.com/dictionary.html
- 크롤링하는 시간이 다소 걸릴 수 있어서 조금만 기다려 주세요.