AWS를 이용한 Serverless 호텔 예약 웹페이지 제작

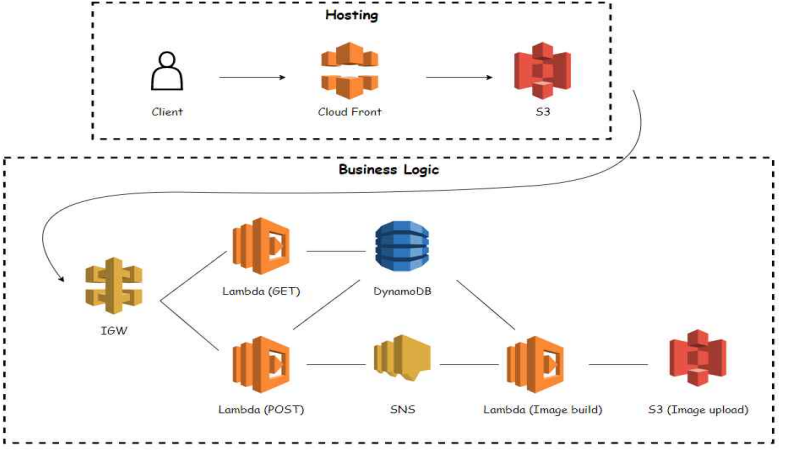
안민규

Index.

Part 1

1. Scenario

프로젝트 시나리오


Hosting
Client
Cloud Front
S3
Business Logic
IGW
Lambda (GET)
DynamoDB
Lambda (POST)
SNS
Lambda (Image build)
S3 (Image upload)

Part 2

2. Business Logic

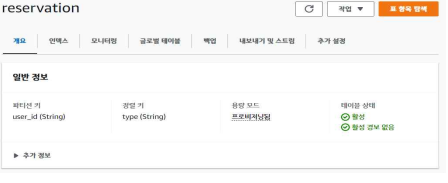
1. DynamoDB 생성

데이터를 저장할 DynamoDB를 우선적으로 생성

user_id를 파티션 키, type을
정렬 키로 설정하여 db를 구축

get, post, Image make 함수와
연동되어 데이터를 스캔 및 생성할 수 있게 설정

연결을 위해 각 lambda 함수에서는
boto3 api를 이용해 DynamoDB를 호출



user_id	type	end_date	phone_num...	start_date	user_name
Testuser	Room1	2022-12-31	01011111111	2022-12-30	user1
안민규	Room1	2022-12-30	01012345678	2022-12-29	안민규
user3	Room1	2022-12-30	01012345678	2022-12-30	user3

2. S3

QR code 제작을 위한 font와 logo 이미지를 저장할 s3 버킷 생성

또한 make 함수를 통해 만들어진 image를 저장

후에는 frontend를 s3 버킷에 저장하고 s3 버킷의 url로 접속해서 이용 가능하게 설정 - Public access가 되야하므로 권한으로 설정

이름	유형
font/	폴더
images/	폴더
index.css	css
index.html	html
index.js	js
qrcodes/	폴더

3. Lambda 함수 구현

1) Get 함수 구현

DynamoDB에 저장된 정보를 가져오는 역할을 하는 함수

IGW로부터 event가 발생하면 event에 맞는 정보를 db에서 scan하고 스캔한 정보를 출력

웹페이지 상에선 예약한 자들의 정보를 확인하거나 qr code를 확인할 때 사용


```
"statusCode": 200,
"items": [
  {
    "user_id": "Testuser",
    "end_date": "2022-12-31",
    "start_date": "2022-12-30",
    "phone_number": "01011111111",
    "user_name": "user1",
    "type": "Room1"
  },
  {
    "user_id": "안민규",
    "end_date": "2022-12-30",
    "start_date": "2022-12-29",
    "phone_number": "01012345678",
    "user_name": "안민규",
    "type": "Room1"
  },
```

3. Lambda 함수 구현

2) Post 함수 구현

DynamoDB에 발생한 event 정보를 저장하는 역할을 하는 함수

새로운 예약자가 생겼을 때 그 정보를 db에 저장

또한 image 함수에 event 발생 여부를 알려주기 위해 sns의 publisher로써 동작

```
"statusCode": 200,
"event": {
  "type": "room2",
  "user_id": "mingyu",
  "user_name": "안민규",
  "start_date": "2022-12-30",
  "end_date": "2022-12-31"
},
```

3. Lambda 함수 구현

3) make 함수 구현

sns에서 발행된 post 함수에서 발생한 event를 subscribe하고 그에 따른 qr code 이미지를 생성하는 함수

이미지를 생성할 때 파이썬 라이브러리인 Pillow, qrcode가 필요한데 두 라이브러리는 기본적으로 탑재된 라이브러리가 아니므로 layer를 생성해서 라이브러리를 사용할 수 있게 설정

ec2를 이용해 만든 가상환경이나 로컬환경에서 필요한 파이썬 라이브러리 파일을 받아 계층에 등록해서 사용

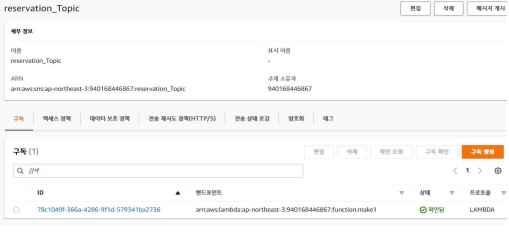
발생한 event를 records에 저장하고 records에 맞는 데이터를 db로부터 가져와 db에 저장된 다른 데이터들을 가져와 이를 통해 user_name, start-end date, qr code가 포함된 이미지를 생성

4. SNS 생성

SNS란 Simple Notification Service의 준말로 Publisher와 Subscriber를 가지고 Publisher에서 발행된 정보를 Subscriber에게 알림 메세지를 전송하는 서비스

앞서 생성한 Post 함수를 SNS의 Publisher로 설정하고 Make 함수를 Subscriber로 설정

웹페이지에서 정보를 기입하고 submit 버튼을 누르면 IGW를 통해 Post 함수에 event가 발생하고 이를 Make함수에게 알리는 역할을 수행



5. IGW 생성

Get함수와 Post함수와 Hosting 영역을 연결하기 위해 Gateway 생성

REST API로 Gateway를 생성하고 앞서 생성한 db에 따라 리소스를 생성 - async service를 사용하기 위함

생성한 리소스에 Get, Post 메소드를 연결

연결한 get 함수에 통합 요청에 user_id, type에 대한 매핑 탬플릿을 생성하고, post 함수의 경우 올바른 데이터가 입력되지 않았을 때 데이터가 저장되는 것을 방지하기 위해 모델을 생성하고, 생성한 모델을 요청 본문에 추가

Part 3

3. Hosting

1. Frontend 구현 - index.html

header에선 여러 stylesheet와 만들어 놓은 css 파일을 불러와 사용, body는 header class, body class, footer class 3가지로 나누어 작업

header class의 경우 이름이 들어가는 구간과 예약, 예약내역을 확인할 수 있는 구간을 따로 설정하고, 예약 관련 구간은 fixed-tab으로 클릭할 수 있게 설정

body class에서 type은 선택지에서 선택할 수 있게 select 형태로 제작, 나머지 정보에 대해서는 직접 기입할 수 있게 설정, start, end date는 달력에서 선택할 수 있게 설정

마지막으로 submit 버튼을 만들고, 이 버튼을 통해 event가 발생하게끔 설정

history로 들어오면 유저 정보가 나타나고, 원하는 유저 정보의 print 버튼을 누를 시 나타나는 print와 close 버튼 제작

1. Frontend 구현 - index.js

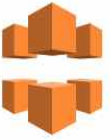
발생하는 event 및 함수에 대한 작업을 진행

Endpoint와 cf에 s3 버킷 도메인, cloudfront 도메인을 입력

달력 작업 진행 및 document 클릭 시 qr code를 보여주게 설정

get 함수와 post 함수의 역할은 ajax 비동기식 call을 사용해 정보를 가져오거나 보내주어 앞서 구현한 기능들이 정상적으로 작동하게 제작

2. Cloudfront 구현



앞서 생성한 s3 bucket을 원본 도메인으로 설정

구현한 Frontend 파일인 index.html을 기본값 루트 객체로 설정 후 배포

기본값 루트 객체 설정은 루트 url을 요청할 때 반환할 객체를 설정

원본 도메인

AWS 원본을 선택하거나 사용자 원본의 도메인 이름을 입력합니다.

msacloud3.s3.ap-northeast-3.amazonaws.com

원본 경로 - *선택 사항* 정보

기본값 루트 객체 - *선택 사항*

뷰어가 특정 객체 대신 루트 URL(/)을 요청할 때 반환할 객체(파일 이름)입니다.

index.html

Part 4

4. Front Hosting

Front Hosting

CORS(Cross-Origin Resource Sharing) 구조로 deploy를 진행

CORS란 웹 페이지 상의 제한된 리소스를 최초 자원이 서비스된 도메인 밖의 다른 도메인으로부터 요청할 수 있게 허용하는 구조

생성된 S3 bucket을 통해 유저와 연결하기에 S3 bucket의 public access를 활성화

Dev stage를 생성한 후 dev stage url을 미리 생성한 index.js의 endpoint에 입력하고, 생성한 Cloud Front의 domain name을 CF endpoint에 입력

Frontend 파일이 저장된 s3 bucket에서 인덱스 문서를 index.html로 설정하고, 정적 웹 사이트 호스팅을 활성화

Part 5

5. Result

Result

Room
Room2

User ID
안민규

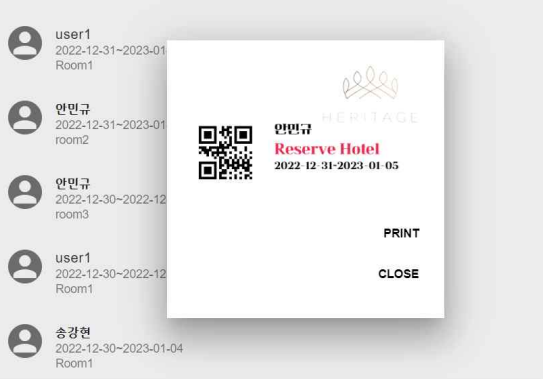
Name
안민규

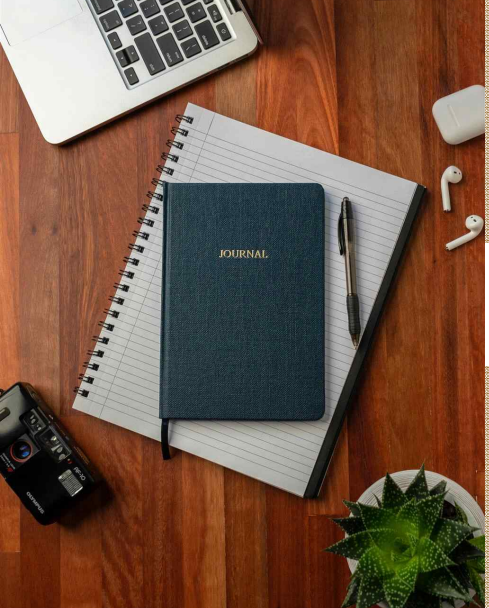
Phone Number
010123456789

start_date
2022-12-31

end_date
2023-01-05

SUBMIT





Part 6

6. Review & Role

Review

Strong

Serverless에 대한 세계의 관심도가 커져가는데 Serverless 환경을 제작 및 운영해봄으로써 세상의 트렌드에 맞는 능력을 기를 수 있었다.

Weak

환경 제작에 있어 아직 미숙한 점이 많아 그 전에 진행한 실습에서 크게 발전하지 못한 점

프론트엔드와 백엔드에 대한 기반 지식들이 아직 부족하여 원하는 것들을 구현하는데 어려움이 있었다.

Develop

추후에 시장 조사를 더 진행해 필요한 시스템에 대한 구축 진행 및 사람들의 needs에 맞는 서비스 구현

서비스를 구현하기는 했지만 실제로 운영될 수 있는 정도 수준은 아니기 때문에 지식을 더 습득하여 실제 서비스 구현

감사합니다