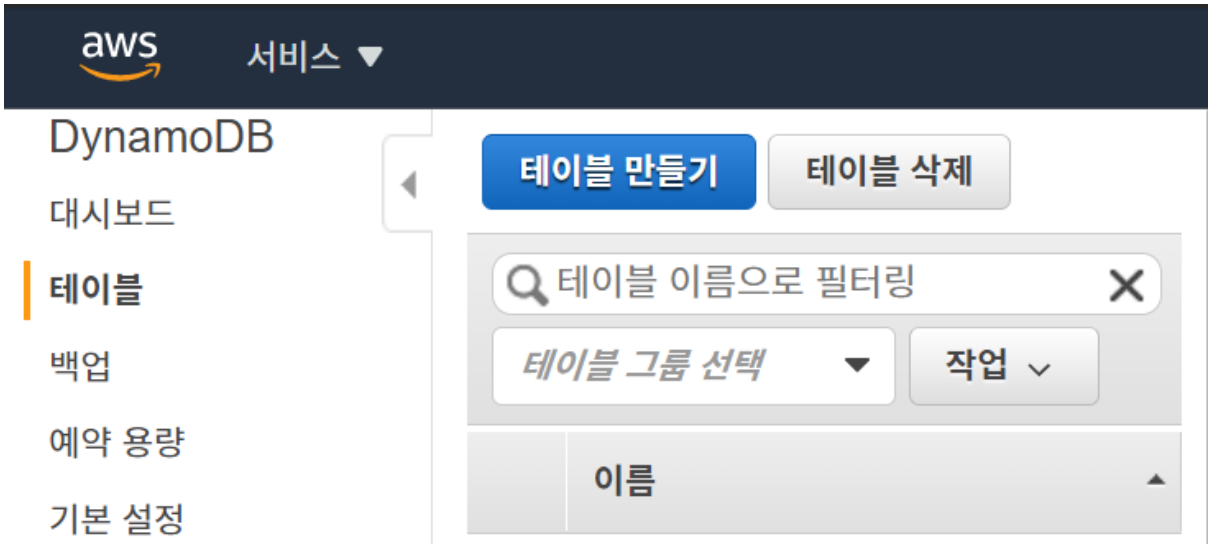
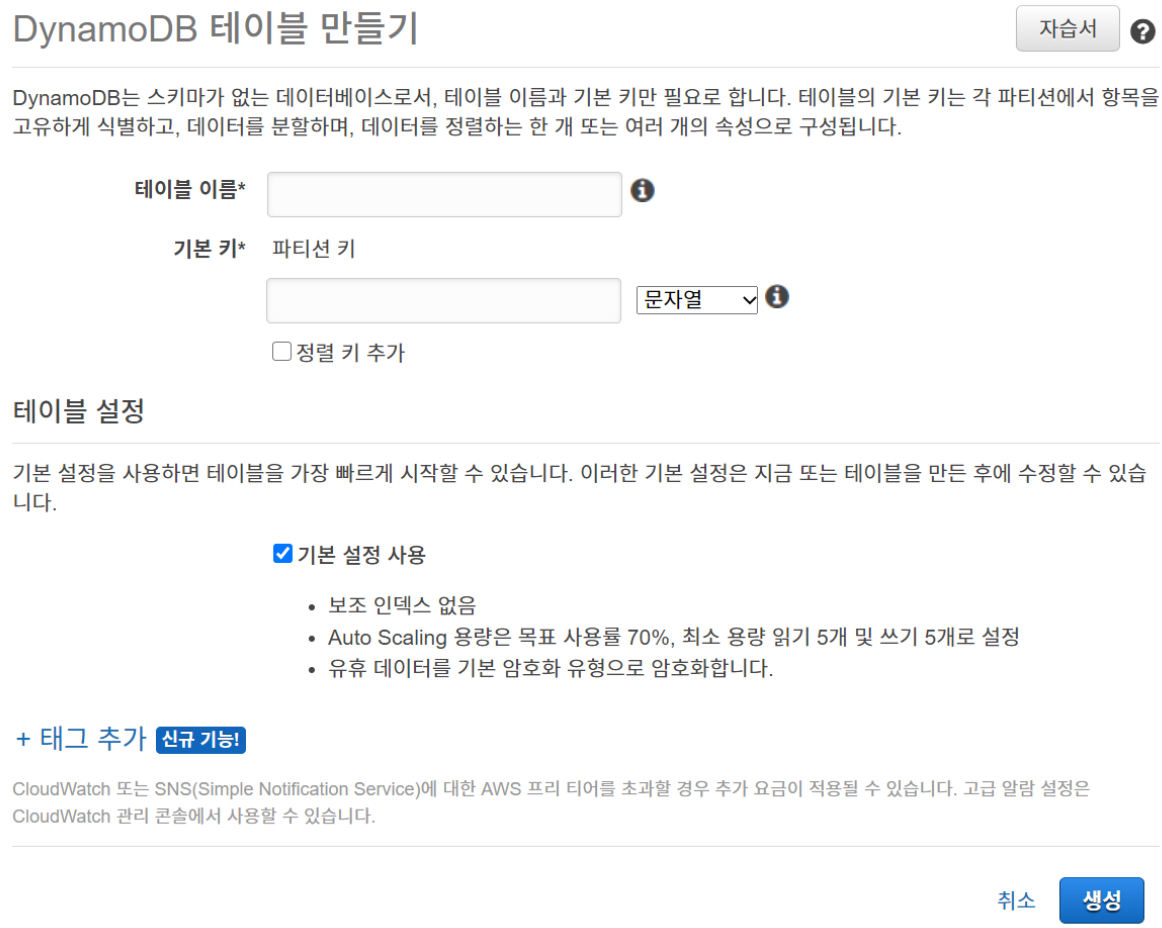
Lambda Chat, Dashboard, DynamoDB, MBTI 인수인계

DynamoDB

1. 테이블 만들기 클릭



2. 정보 입력



만들어야 할 테이블

Aa 테이블 이름	기본 키	추가 키
chat_connections	connectionId (문자열)	
chat_log	roomId (문자열)	requestTimeEpoch (번호)
MBTI-product	id (번호)	

RDS

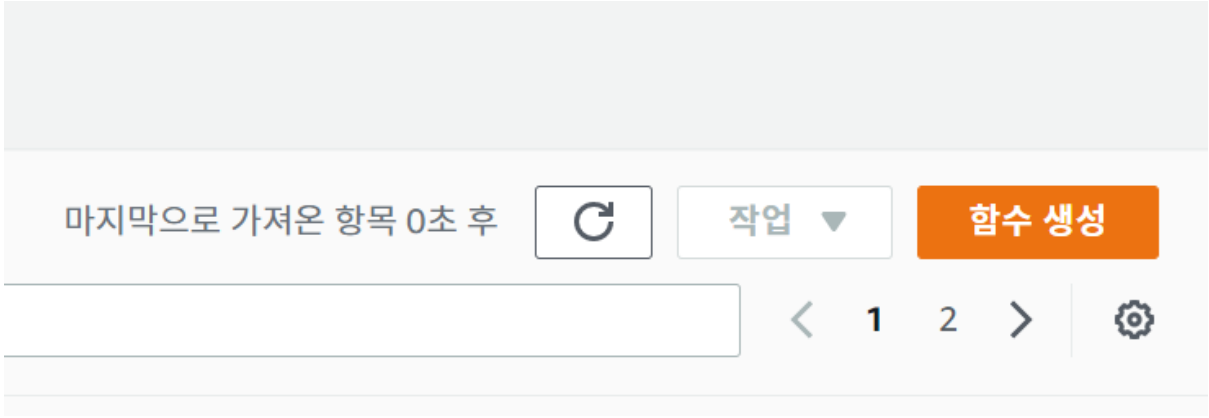
`exec/Lambda/LanguageGenerator` 의 `(1)RDS만들기및설정하기.pdf` 를 참고하여 만든다.

MySQL Workbench 같은 프로그램에서 `make_table.sql`, `mbti_relationship.sql`, `procedure.sql` 을 순서대로 실행한다.

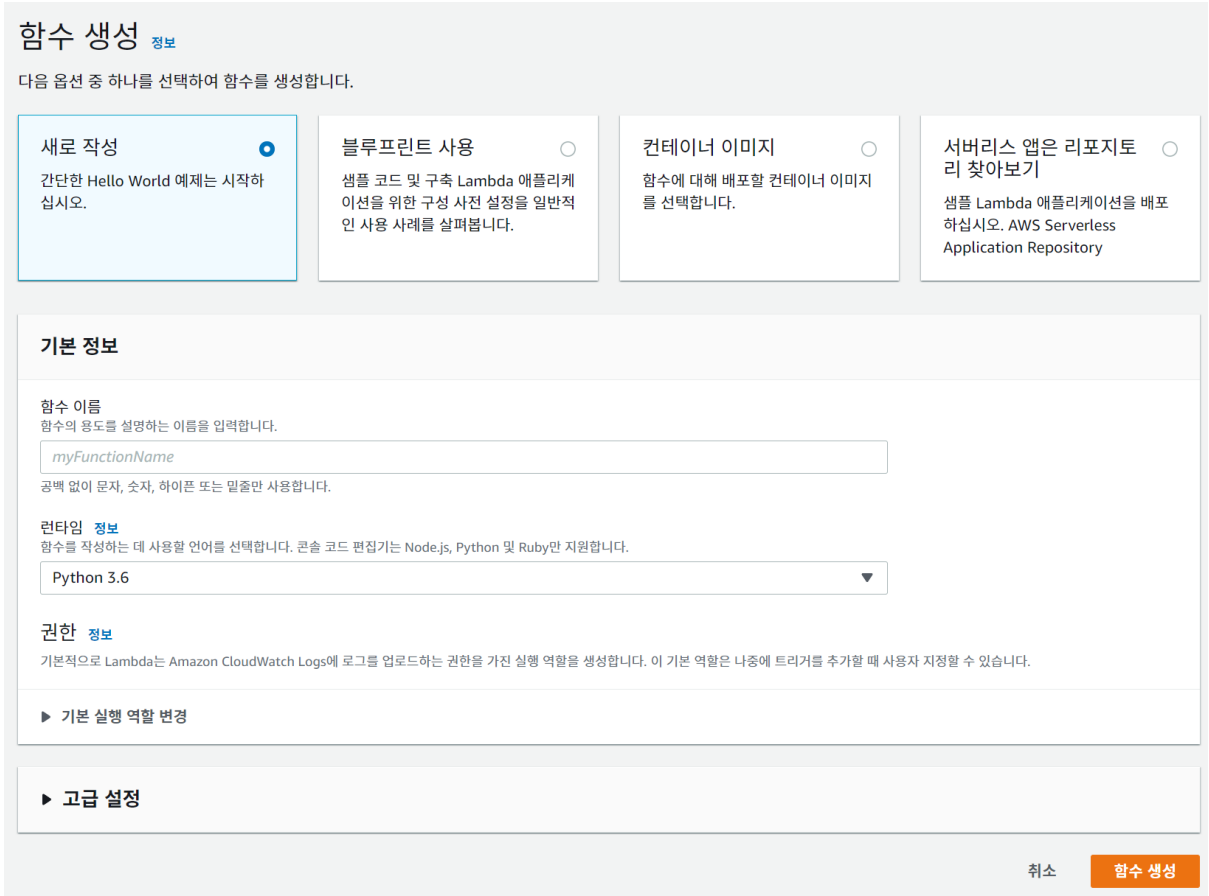
AWS Lambda

만드는법

1. 함수 생성 클릭

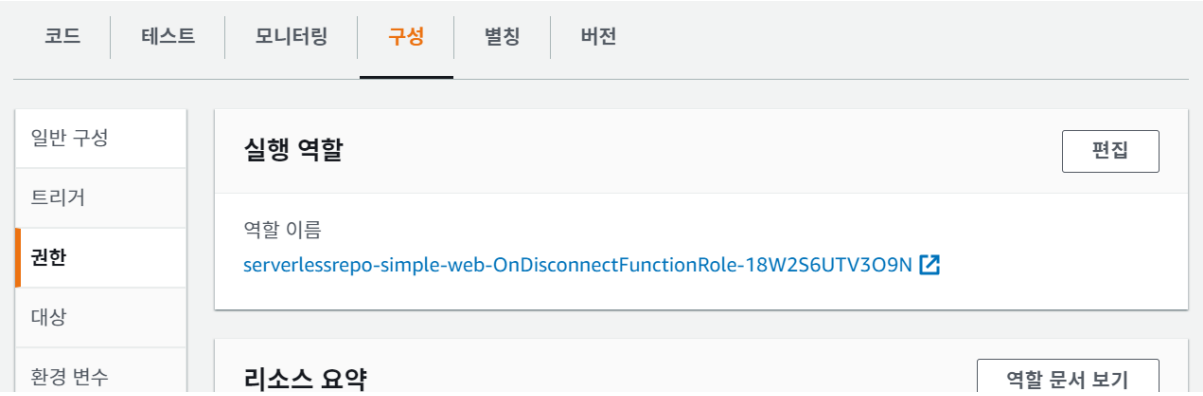


2. 함수 이름, 런타임 설정 후 함수 생성 클릭



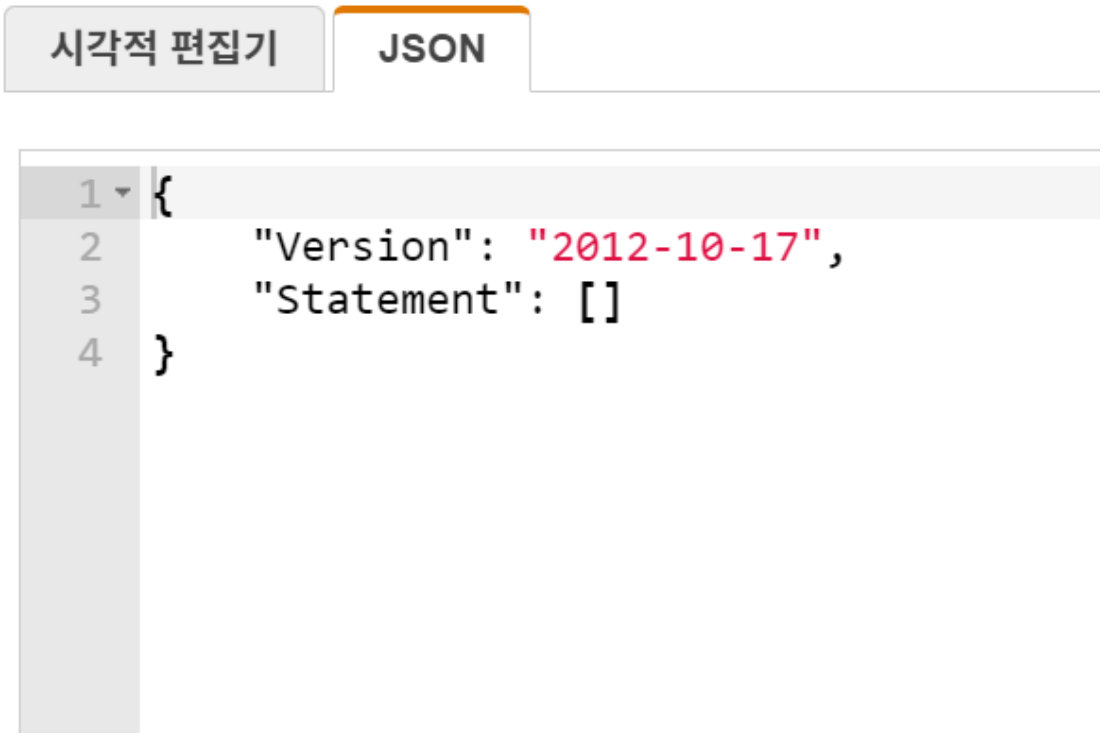
IAM 등록 방법

1. 구성 → 권한 → 역할 이름 클릭



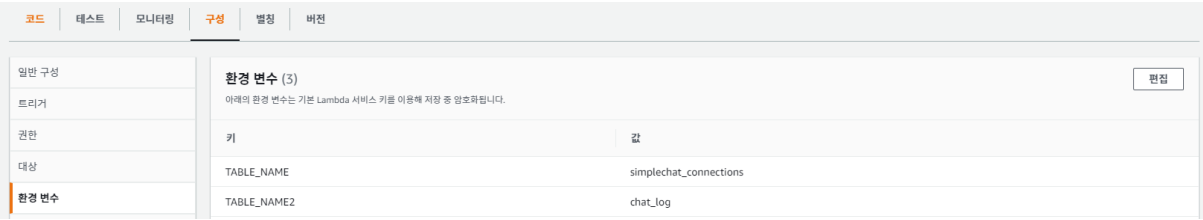
2. 인라인 정책 추가 클릭

3. JSON 클릭 후 IAM 추가

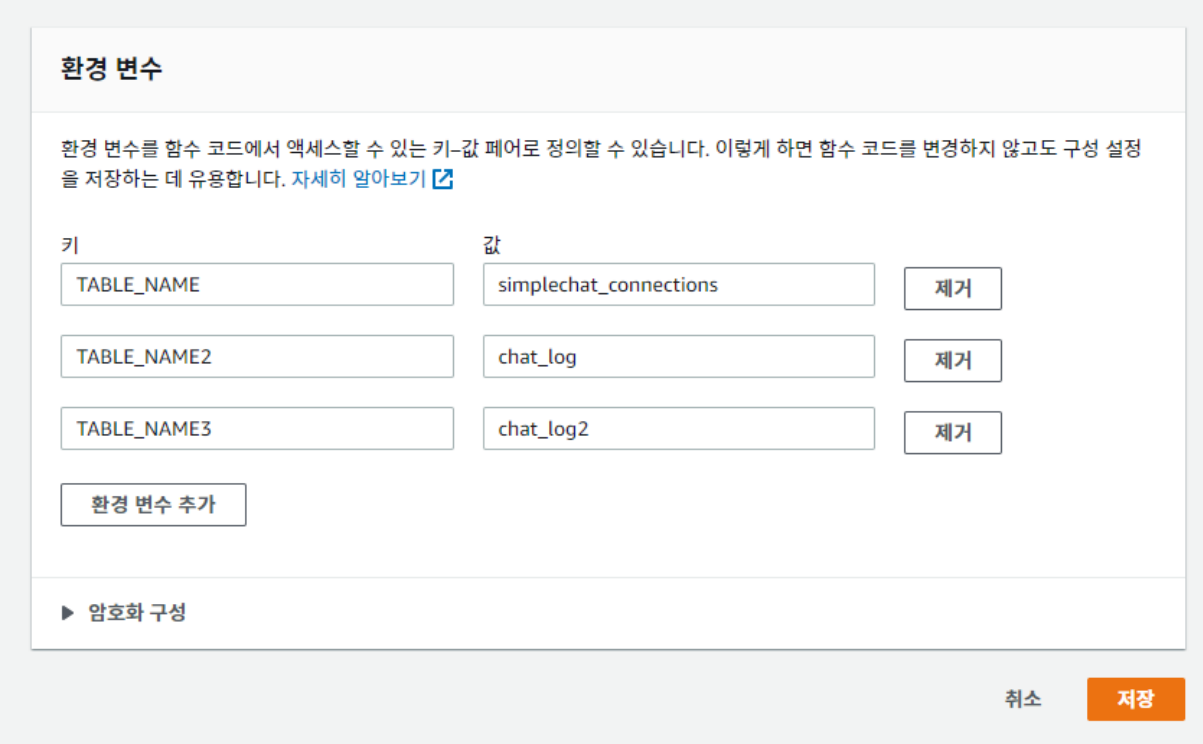


환경 변수 등록 방법

1. 구성 → 환경 변수 → 편집 클릭(환경 변수 필요 시)

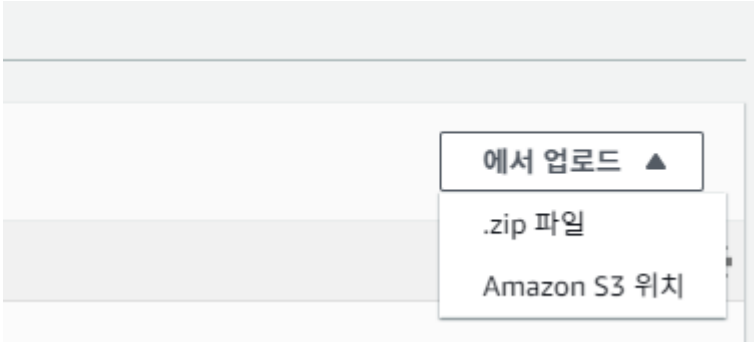


2. 환경 변수 키 / 값 작성 후 저장

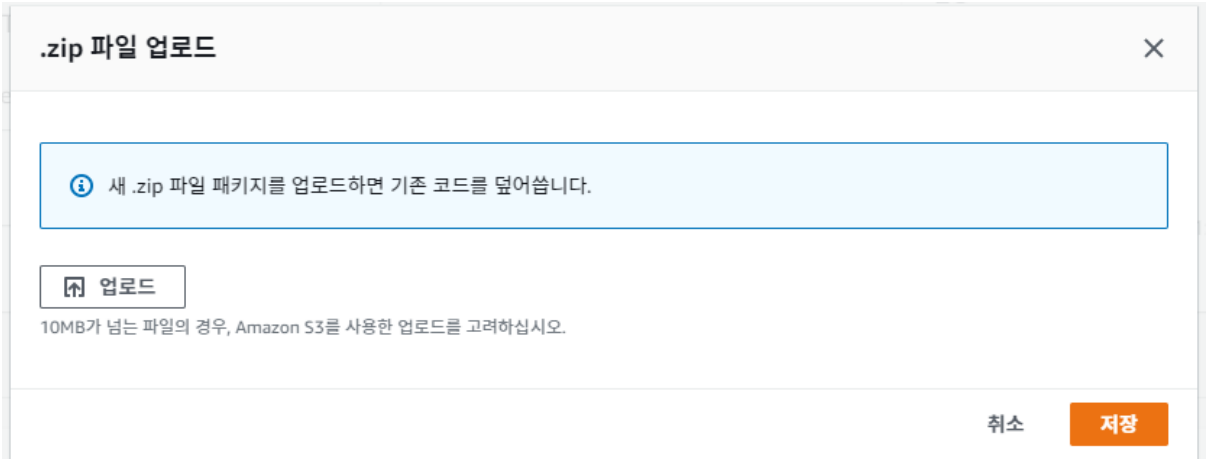


zip 파일로 업로드 방법

1. zip 파일에서 업로드 클릭



2. 업로드할 zip 파일 찾아서 등록 후 저장



MBTI 함수

1. MBTI_landing_page.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.6
    - requirements: pymysql
    - MBTI 성격 유형 테스트의 메인 페이지에서 각 성격 유형의 검사 결과 비율과 총 검사 결과 count를 반환하는 함수
    - `line 6` 의 `host` 에 AWS RDS 엔드포인트 주소, `user` 에 유저명, `password` 에 비밀번호, `db` 에 스키마명 넣기
2. MBTI_result.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.6
    - requirements: pymysql
    - MBTI 성격 유형을 input으로 받으면 해당하는 유형의 정보를 output으로 반환하는 함수
    - `line 7` 의 `host` 에 AWS RDS 엔드포인트 주소, `user` 에 유저명, `password` 에 비밀번호, `db` 에 스키마명 넣기
3. MBTI_update_count.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.6
    - requirements: pymysql
    - MBTI 성격 유형 테스트의 마지막 질문에 대한 응답을 하고 나면 데이터베이스에 해당하는 유형의 count를 +1 하는 함수
    - `line 8` 의 `host` 에 AWS RDS 엔드포인트 주소, `user` 에 유저명, `password` 에 비밀번호, `db` 에 스키마명 넣기

4. MBTI_randProduct.py
    - 런타임: Python 3.6
    - 삼성전자 제품 중 다섯 개를 랜덤으로 추출해서 반환하는 함수
    - IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

DynamoDB 함수

1. DynamoDB_upload_product.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.8
    - requirements: boto3
    - DynamoDB에 삼성 제품을 무작위 순서대로 입력하는 함수. 압축파일 내의 json 파일의 데이터를 db에 입력한다.
    - IAM

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
```

```
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Chat 함수

1. Chat_load_prevChat.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.6
    - requirements: boto3
    - MBTI 성격 유형 별 결과 페이지에서 채팅 버튼을 눌렀을 때 이전 채팅 기록을 불러오는 함수
    - IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

2. Chat_on_connect.js
    - 런타임: Node.js 12.x
    - 환경변수
        - `TABLE_NAME` : `chat_connections`
        - `TABLE_NAME2` : `chat_log`
    - 채팅을 위해 웹소켓 연결을 시작하는 함수
    - IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

3. Chat_enter_room.js
    - 런타임: Node.js 12.x
    - 환경변수
        - `TABLE_NAME` : `chat_connections`
    - MBTI 성격 유형 별 채팅방에 입장하는 함수
    - IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Chat_send_message.js
    - 런타임: Node.js 12.x
    - 환경변수
        - `TABLE_NAME` : `chat_connections`
    - 채팅방에 메시지를 전달하는 함수
    - IAM

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:ConditionCheckItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

5. Chat_on_disconnect.js
    - 런타임: Node.js 12.x
    - 환경변수
        - `TABLE_NAME` : `chat_connections`
    - 연결이 끊어지면 데이터베이스에서 connectionId를 삭제하는 함수
    - IAM

```
{
    "Statement": [
        {
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:DeleteItem",
                "dynamodb:PutItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:BatchGetItem",
                "dynamodb:DescribeTable",
                "dynamodb:ConditionCheckItem"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Dashboard

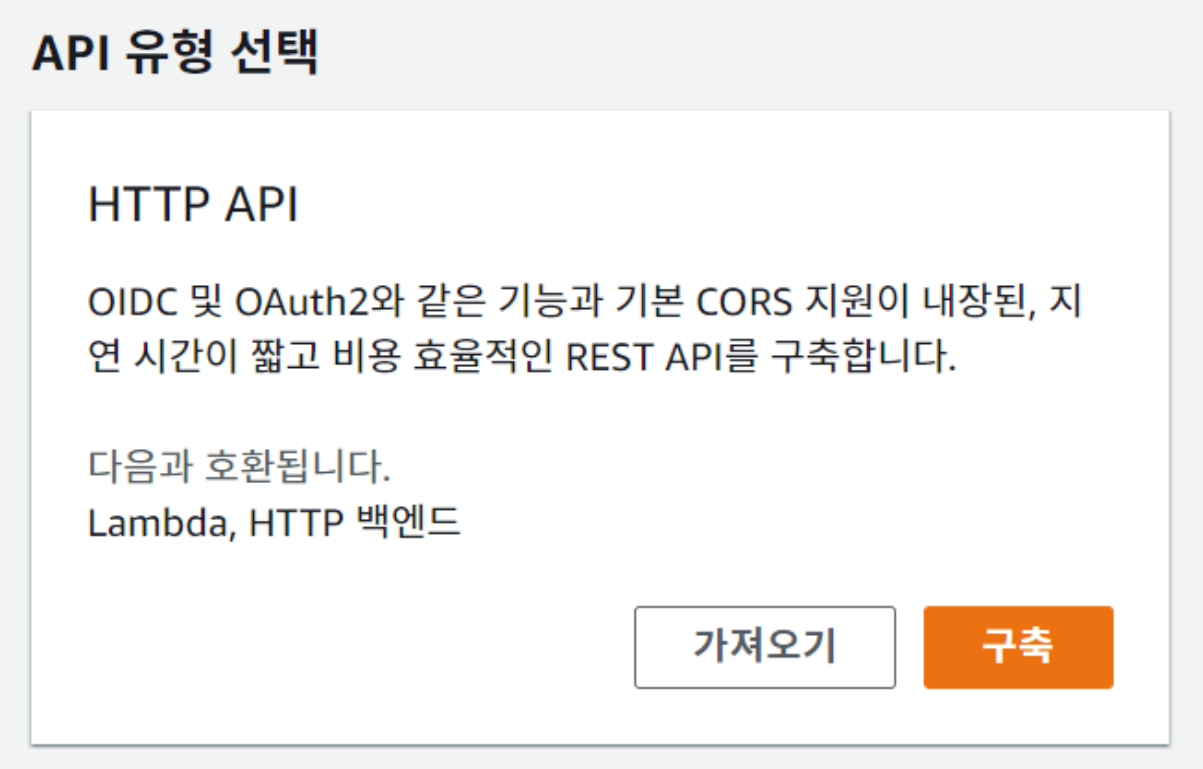
1. Dashboard_CW_API.zip (zip 파일로 업로드하기)
    - 런타임: Python 3.6
    - requirements: boto3

- Dashboard에 보여줄 lambda 함수 관련 cloudwatch log 결과를 이미지를 반환하는 함수

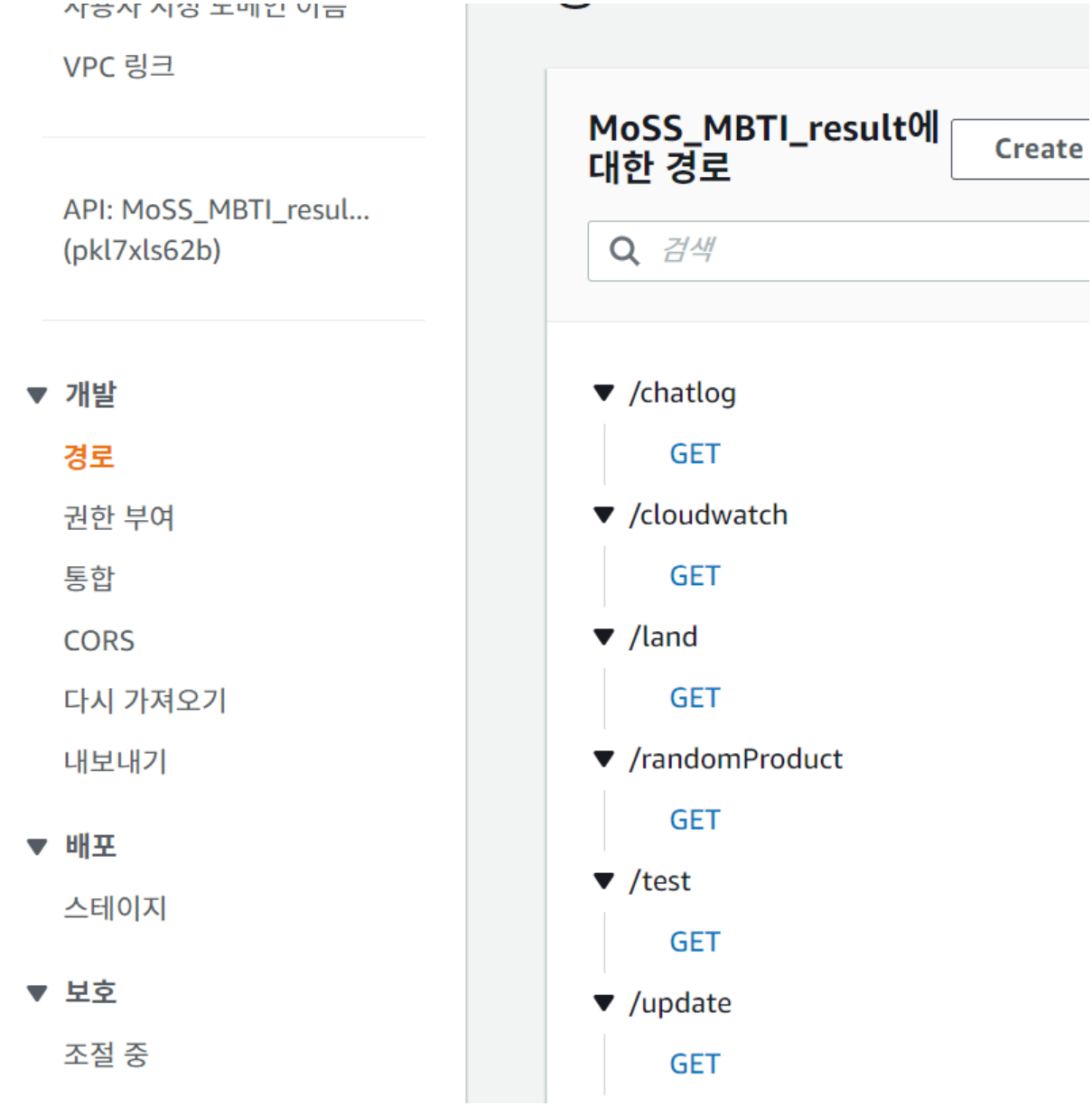
API Gateway

MBTI API

1. HTTP API 구축



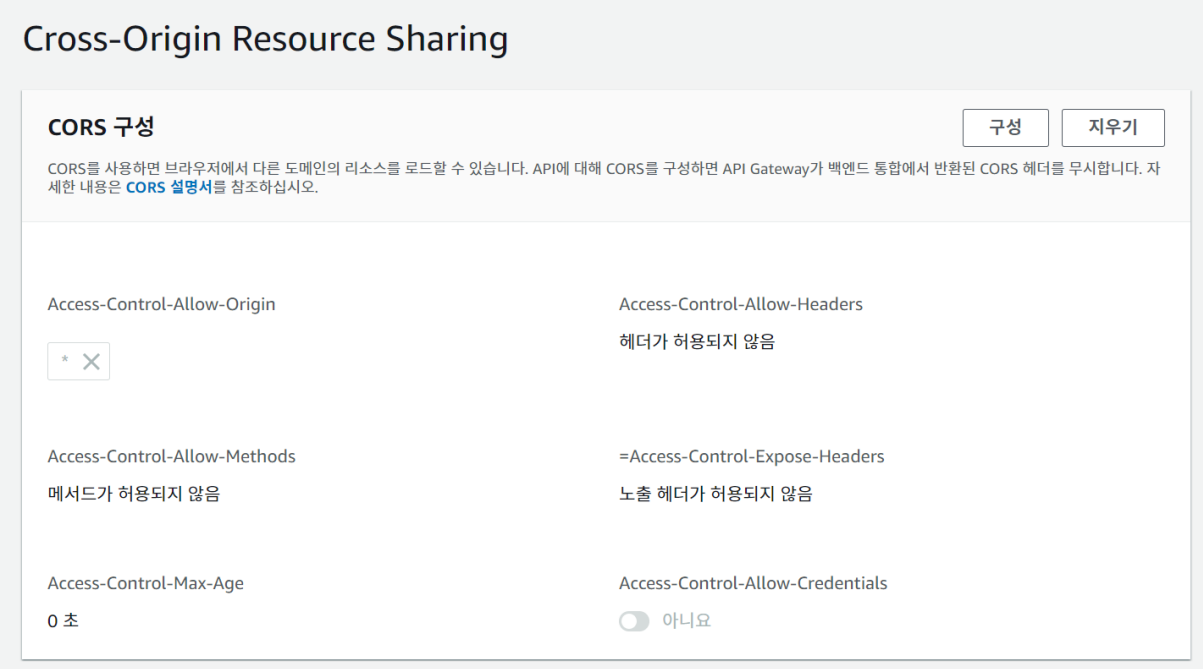
2. 경로 생성



MBTI API - lambda 연결

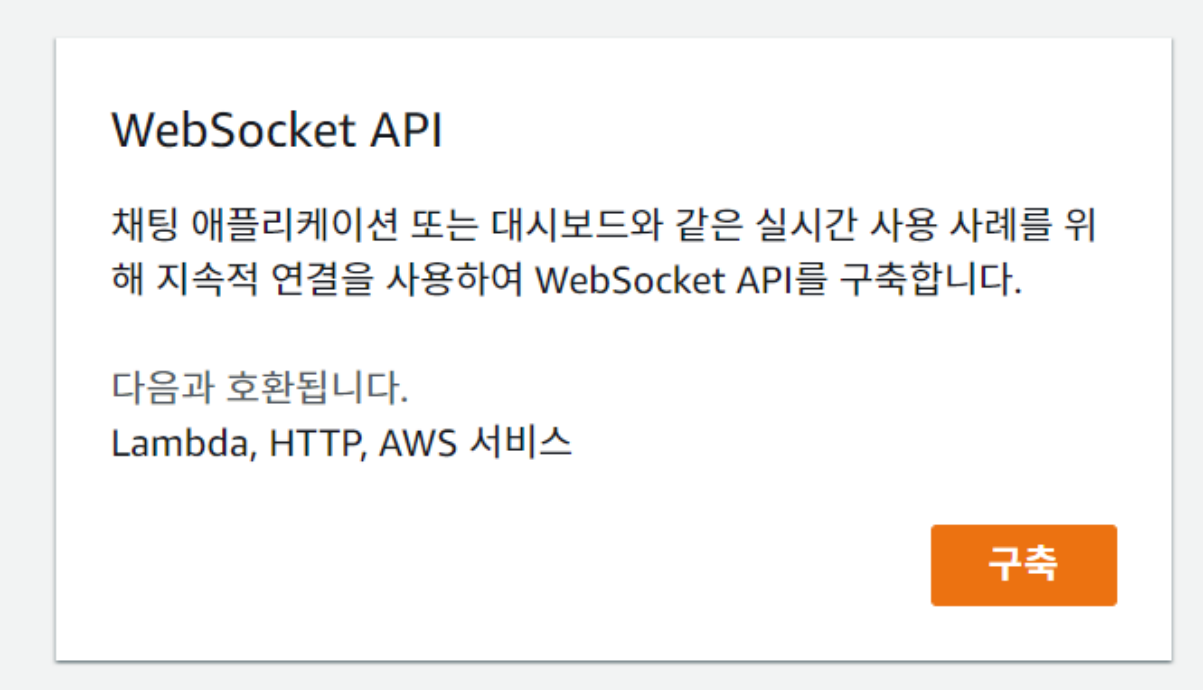
Aa API 경로	Lambda function
/chatlog	Chat_load_prevChat
/cloudwatch	Dashboard_CW_API
/land	MBTI_landing_page
/randomProduct	MBTI_random_product
/test	MBTI_result
/update	MBTI_update_count

3. CORS 설정

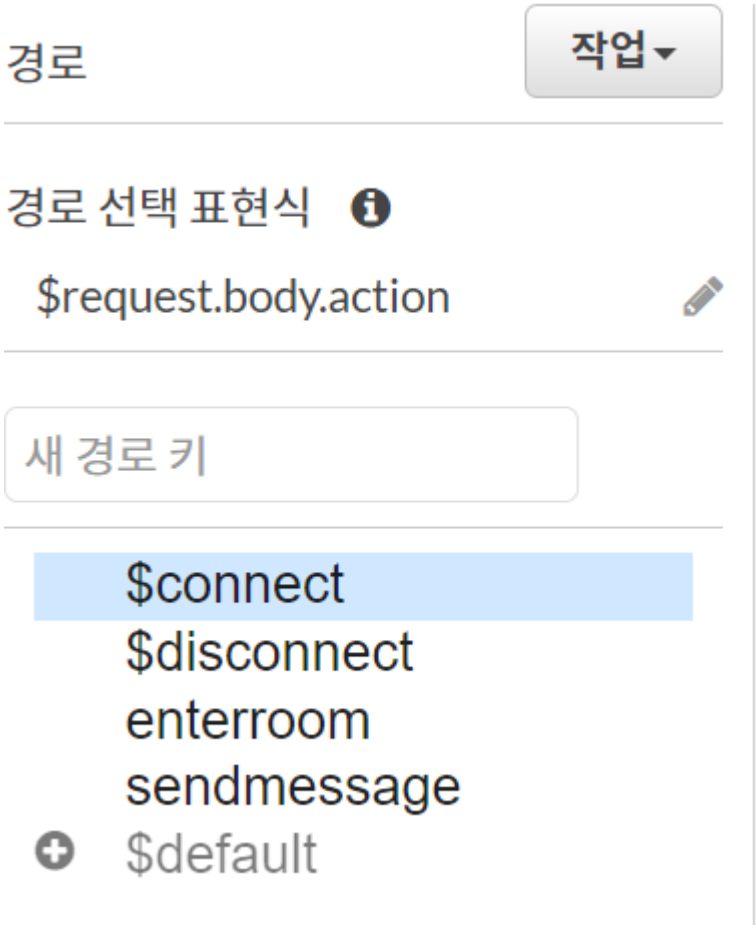


Chat API

1. WebSocket API 구축



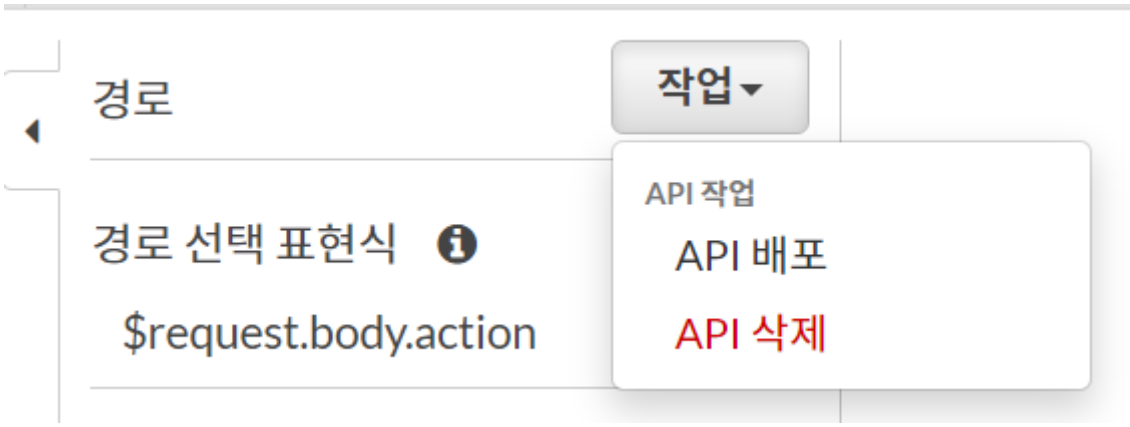
2. 각 경로에 lambda function 연결



websocket 경로 - lambda

Aa websocket 경로	lambda
$connect	Chat_on_connect
$disconnect	Chat_on_disconnect
enterroom	Chat_enter_room
sendmessage	Chat_send_message

3. API 배포 (안 하면 변경 사항 적용이 안 됨)


경로
작업
API 작업
API 배포
API 삭제
경로 선택 표현식
$request.body.action