Project: Deploying Serverless Web Application
*(*서버리스 기반 웹앱 구축*)*

<Agenda>

➔ -S3 정적 호스팅 기능을 활용하여 웹 페이지 구현

➔ -DynamoDB 테이블 생성(파일 데이터 저장)

➔ -S3에 업로드 된 파일을 미리 서명된 URL로 가져오는 Lambda 생성

◆ API Gateway 기반의 REST API 생성

➔ 웹페이지에 REST API 연동

➔ 파일 업로드 및 코드로 다운로드 확인

Step1. DynamoDB 생성(생성 하기 전 Region이 Seoul로 선택되어있는지 확인)

- DynamoDB 생성 클릭
  - 테이블 이름 : filedata
  - 파티션 키 : code
  - 설정 : 설정 사용자 지정
  - 읽기/쓰기 용량 설정 : 온디맨드
  - DynamoDB 생성 완료

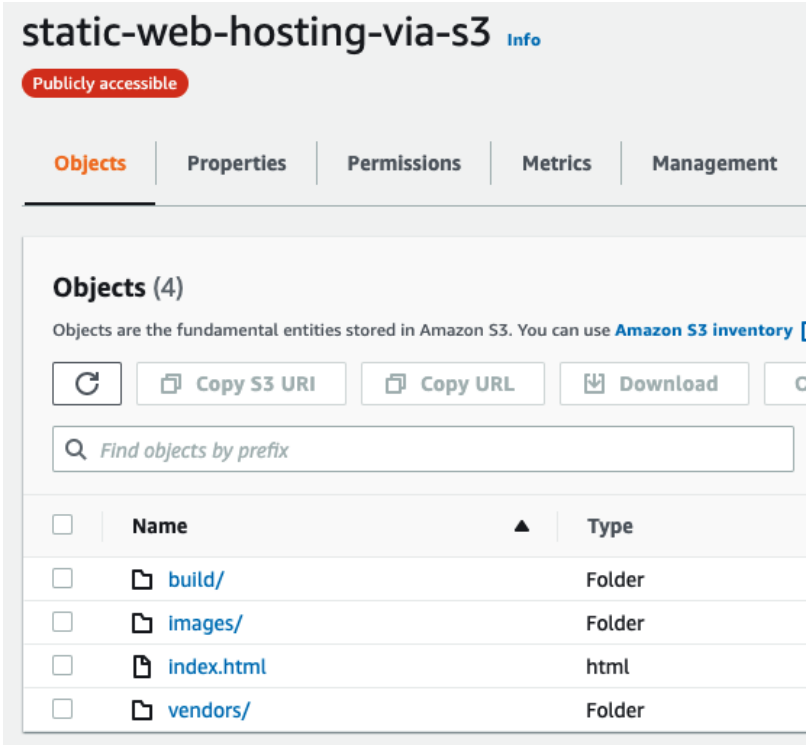
Step2. 정적 웹사이트 호스팅을 위한 S3 버킷 생성

- 버킷 만들기 클릭
  - 버킷 이름: 원하는 고유한 이름 지정(ex. static-web-hosting-via-s3)
  - 모든 퍼블릭 엑세스 차단 uncheck
  - 버킷 만들기 완료
  - 버킷 선택 후 속성
  - 정적 웹 사이트 호스팅 편집
    - 정적 웹 사이트 호스팅 활성화
    - 인덱스 문서: index.html
    - 변경 사항 저장
- 버킷 선택 후 권한
  - 버킷 정책 편집

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::[버킷명]/*"
```

```
            }
        ]
    }
```

- 버킷명에 해당하는 부분에 버킷 이름: 원하는 고유한 이름 지정 (ex. static-web-hosting-via-s3)으로 변경
- 변경 후 붉은 글씨로 'Publicly accessible'이라는 문구 확인
- frontend 폴더로 들어간 후 3개의 하위 폴더와 1개의 파일을 생성한 S3 버킷에 업로드



- 버킷 선택 후 속성
  - 맨 밑의 정적 웹 사이트 호스팅탭에 가서 버킷 웹 사이트 엔드포인트 주소 복사 후 주소창에 URL 입력, 웹 사이트가 뜨는지 확인

Step3. Backend에 활용할 Lambda를 위한 IAM Role 생성

- IAM 선택
  - 역할 -> 역할 만들기 -> 사용 사례(Lambda 선택)
  - 권한 추가(아래 리스트에 있는 권한 추가
    - AmazonS3FullAccess
    - AmazonDynamoDBFullAccess
    - CloudwatchFullAccess
  - 역할 이름: IAM-Role-for-Lambda

Step 2: Add permissions

Permissions policy summary

Policy name	Type
CloudWatchFullAccess	AWS managed
AmazonDynamoDBFullAccess	AWS managed
AmazonS3FullAccess	AWS managed

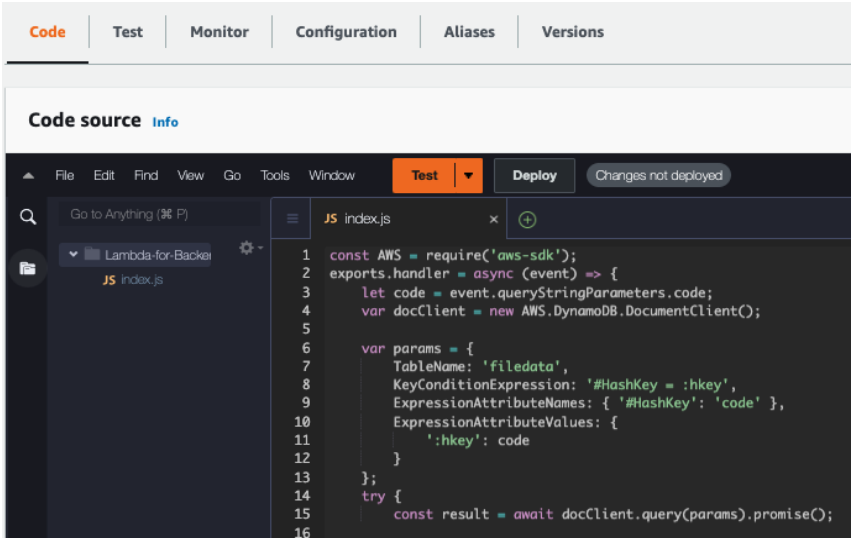
Step4. Backend에 활용할 Lambda 생성(생성 하기 전 Region이 Seoul로 선택되어있는지 확인)

- 함수 생성
  - 새로 작성
  - 함수 이름: Lambda-for-Backend
  - 런타임: Node.js 12.x
  - 권한 -> 기본 실행 역할 변경 -> 기존 역할 사용 -> IAM-Role-for-Lambda -> 함수 생성
- 함수 생성 확인

/* Lambda 함수에 2가지 방법으로 소스 코드를 업로드 가능
-소스코드를 index.js에 복사 붙여넣기(소스코드 위치:Backend -> src -> lambda -> get.js)
-Backend -> src폴더를 압축해서 zip 파일을 업로드
*/

- 소스코드를 index.js에 복사 붙여넣기한 예시

- 소스코드를 복사 붙여넣기한 후 Deploy 클릭

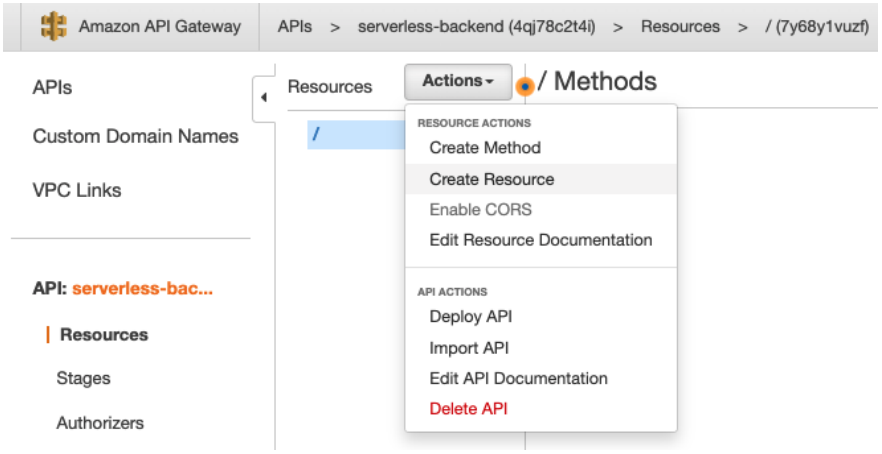


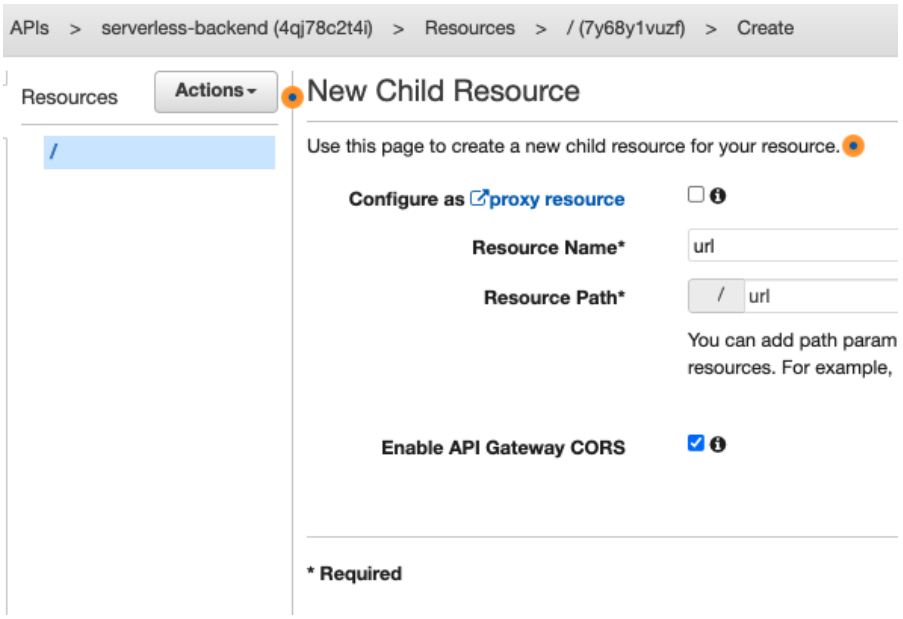
Step5. API Gateway 생성(생성 하기 전 Region이 Seoul로 선택되어있는지 확인)

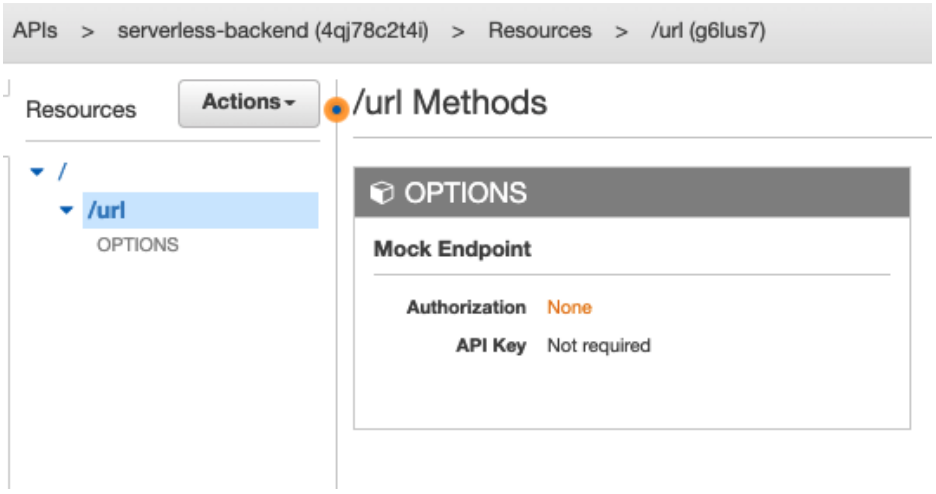
- API 생성
  - Rest API 선택
  - 프로토콜: REST

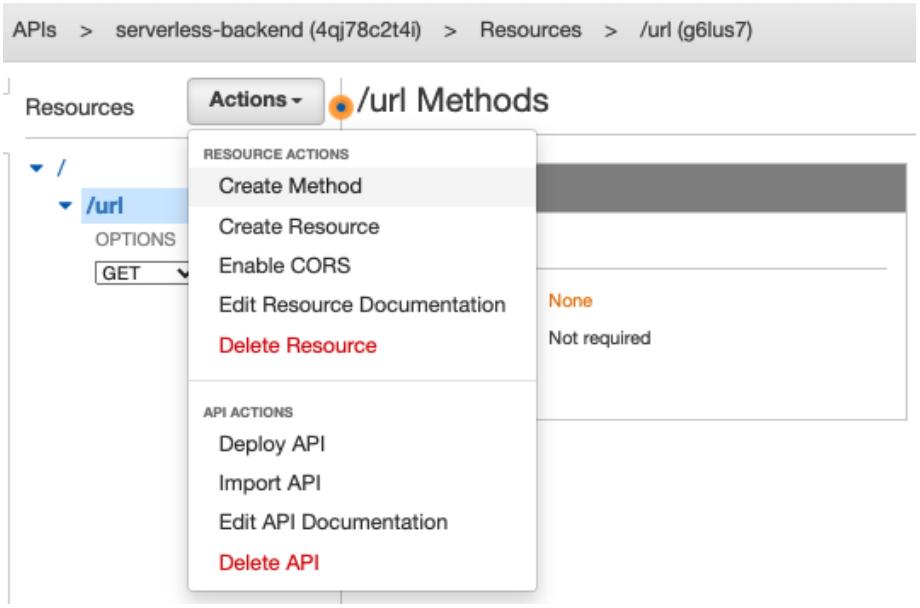
- 새 API 생성: 새 API
- 설정
  - API 이름: serverless-backend

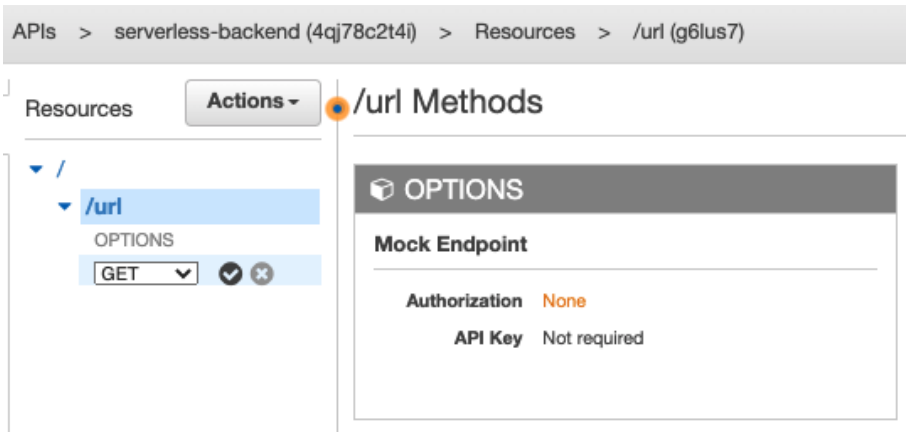
- 생성 완료 후(작업 -> 리소스 생성)



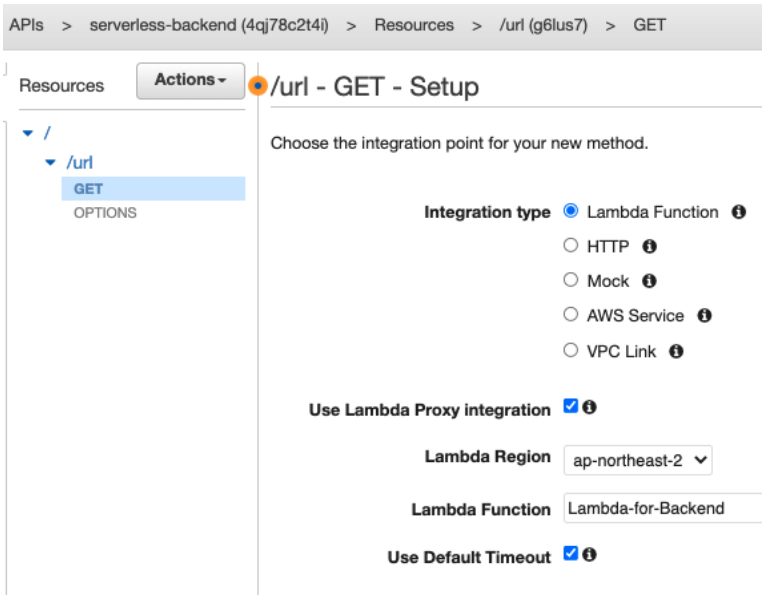




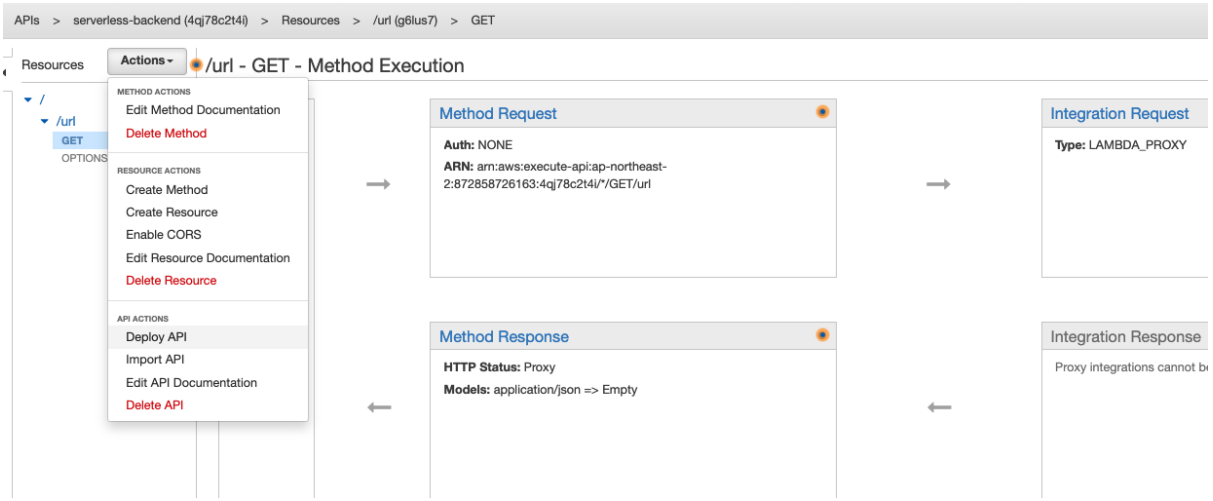




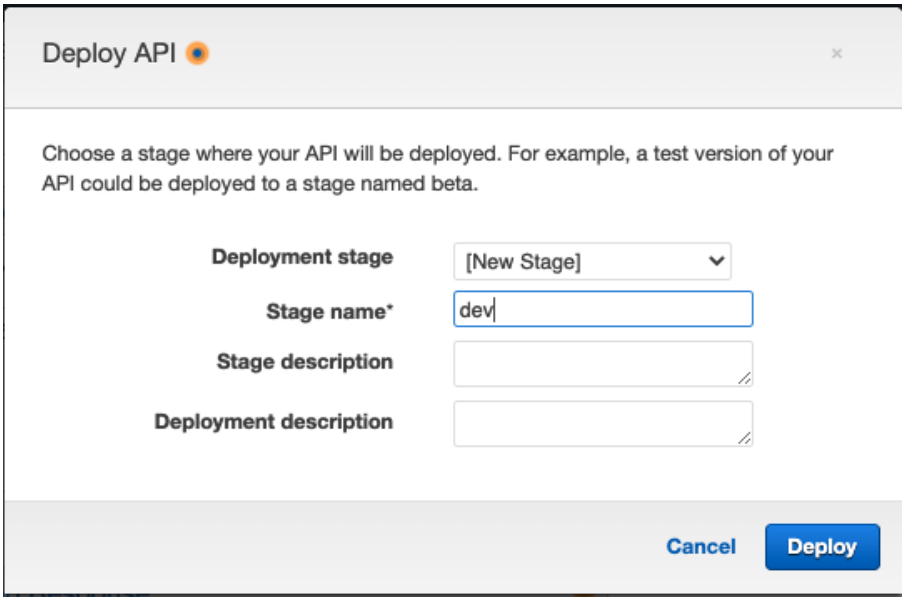
- 작업 -> 메서드 생성 -> GET 선택 -> 체크 버튼 클릭



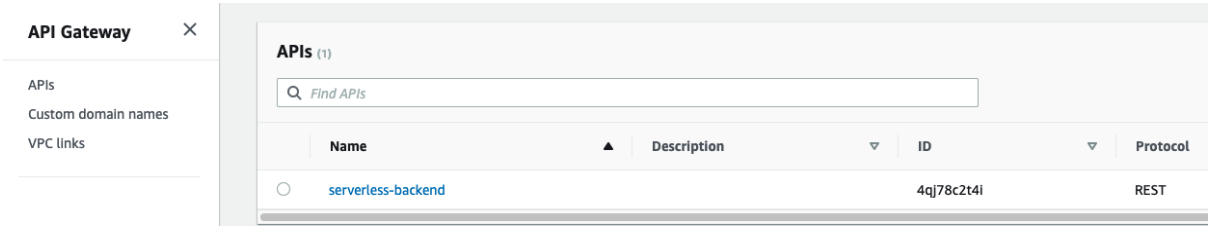
- Lambda 함수는 Lambda-for-Backend 선택, Lambda 함수에 대한 권한 추가 알림이 뜨면 확인 클릭



- 작업 -> API 배포



- 새 스테이지 -> 스테이지 명: dev -> 배포



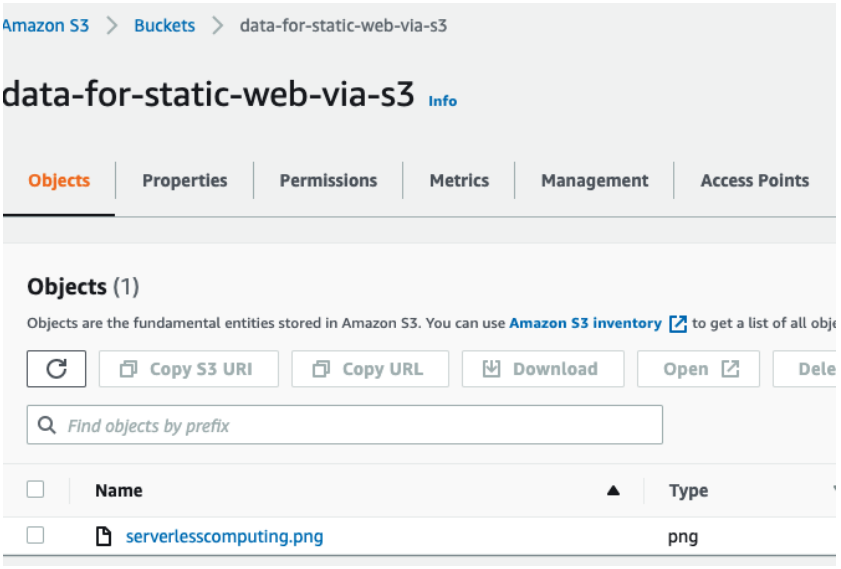
- 생성된 API 확인 및 API ID 확인
  - 이후 Frontend 폴더의 index.html에 들어가 21번째 줄 var api_gateway_id를 본인의 API ID로 변경

```
19
20    <script>
21        var api_gateway_id = "n3ugy3khg4";
22    </script>
23
```

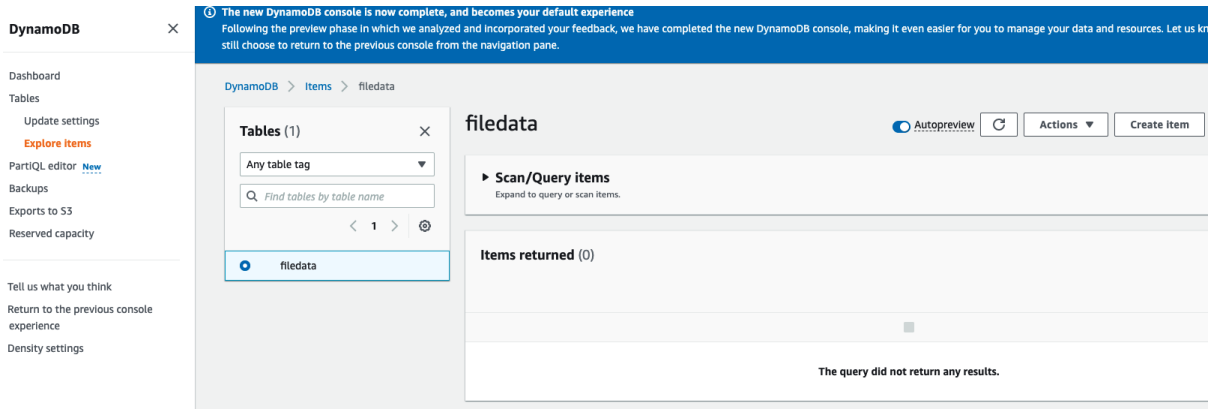
  - 변경 후 index.html을 버킷에 다시 업로드

Step6. 정적 웹 사이트 데이터를 담기 위한 S3 생성

- 버킷은 전부 기본 값으로 설정해놓고 이름만 원하는 이름으로 설정해서 생성
- 유저들이 코드를 입력 후 다운할 파일 업로드



Step7. DynamoDB에 key, value 추가



- 항목 생성 클릭

- bucket에는 정적 웹 호스팅을 위한 s3 버킷 이름이 아닌 그 뒤에 만들었던 s3 버킷
- key는 s3 버킷 안에 있는 파일명
- 속성을 추가할 때는 새 속성 추가 버튼 선택
- 마지막으로 웹 접속 후, code 부분에 test 입력 시 해당 이미지가 보이면 성공
- 이미지뿐만 아니라 다른 확장자의 파일도 가능(ex. pdf)