Assignment 5 - APIGateway

Problem Statement

The task is to create a serverless REST API using AWS Lambda, Amazon DynamoDB, and Amazon API Gateway to perform CRUD operations on book items. The main component of the application is a REST API, backed by Amazon API Gateway and AWS Lambda. Amazon DynamoDB is used to persist the book collection. The AWS SAM CLI is responsible for development and deployment.

The following endpoints are available:

- GET /books/ - retrieves the list of all books stored.
- POST /books/ - creates a new book.
- GET /books/{id} - retrieves a specific book.
- DELETE /books/{id} - deletes a specific book.
- PUT /books/{id} - updates the state of a book.

Solution

Lambda function code to perform the CRUD operations on the DynamoDB table.

```
import json
import boto3
from botocore.exceptions import ClientError


tableName = "my-table"
dynamo = boto3.client('dynamodb')

def returnFun(status,body):
    return{
        "statusCode":status,
        "body":json.dumps(body)

    }
def specData(book_id):
    response = dynamo.get_item(
        TableName=tableName,
        Key={
            'id': {'S': book_id}

        })
    return(response)

def getData():
    response = dynamo.scan(TableName=tableName)
    return(response['Items'])

def putData(item):
    try:
        response = dynamo.put_item(TableName=tableName,Item=item)
        return(response['ResponseMetadata']['HTTPStatusCode'])
    except ClientError as e:
        return(400)
def deleteData(book_id):
    response = dynamo.delete_item(TableName=tableName,Key={'id': {'S': book_id}})
    return(response)

def updateData(book_id,ExpressionAttributeNames,ExpressionAttributeValues,UpdateExpression):
    response = dynamo.update_item(
        ExpressionAttributeNames=ExpressionAttributeNames,
        ExpressionAttributeValues=ExpressionAttributeValues,Key={'id': {'S': book_id}},
        UpdateExpression=UpdateExpression,
        TableName=tableName)
    return(response)
```

```
def lambda_handler(event, context):
    if(event['httpMethod']=='GET'):
        if event['pathParameters']!=None:
            book_id = event['pathParameters']['id']
            response = specData(book_id)
            if 'Item' in response:
                return(returnFun(200,response['Item']))
            else:
                return(returnFun(404,'Book not found'))
        else:
            data=getData()
            return(returnFun(200,data))
    elif(event['httpMethod']=='POST'):
        item=json.loads(event['body'])
        statusCode = putData(item)
        if(statusCode == 200):
            return(returnFun(200,"Succesfully Item updated"))
        else:
            return(returnFun(400,"Please send item with correct schema"))
    elif(event['httpMethod']=='DELETE'):
        if event['pathParameters']!=None:
            book_id = event['pathParameters']['id']
            response = deleteData(book_id)
            return(returnFun(200,response))
    elif(event['httpMethod']=='PUT'):
        print(event)
        if event['pathParameters']!=None:
            book_id = event['pathParameters']['id']
            body=json.loads(event['body'])
            ExpressionAttributeNames=body['ExpressionAttributeNames']
            ExpressionAttributeValues=body['ExpressionAttributeValues']
            UpdateExpression=body['UpdateExpression']
            response=updateData(book_id,ExpressionAttributeNames,ExpressionAttributeValues,UpdateExpression)
            return(returnFun(200,response))
    else:
        data="Unknown method"
        statusCode=200
        print(event)
        return{
            "statusCode":statusCode,
            "body":json.dumps(data)
        }
```

Sam template to create the lambda function, API Gateway and DynamoDB table.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  APIGateway

  Sample SAM Template for APIGateway
Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: DynamoDB_CRUD/
      Handler: app.lambda_handler
      Runtime: python3.9
      Architectures:
        - x86_64
      Events:
        BooksGet:
          Type: Api
          Properties:
            Path: /books
            Method: get
        BooksGetSpec:
          Type: Api
          Properties:
            Path: /books/{id}
            Method: get
```

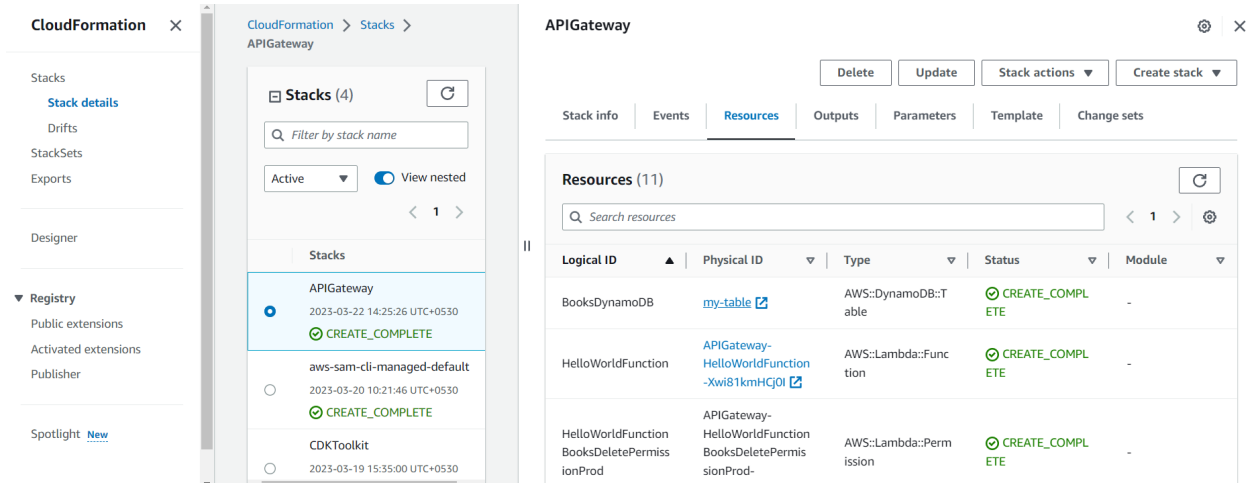
```yaml
        BooksPost:
          Type: Api
          Properties:
            Path: /books
            Method: post
        BooksPut:
          Type: Api
          Properties:
            Path: /books/{id}
            Method: put
        BooksDelete:
          Type: Api
          Properties:
            Path: /books/{id}
            Method: delete
      Policies:
        - DynamoDBCrudPolicy:
            TableName: !Ref BooksDynamoDB
  BooksDynamoDB:
    Type: AWS::Serverless::SimpleTable
    Properties:
      TableName: my-table
      PrimaryKey:
          Name: id
          Type: String
      ProvisionedThroughput:
          ReadCapacityUnits: 5
          WriteCapacityUnits: 5
Outputs:
  HelloWorldApi:
    Description: "API Gateway endpoint URL for Prod stage for Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/books/"
```

Sam Deploy

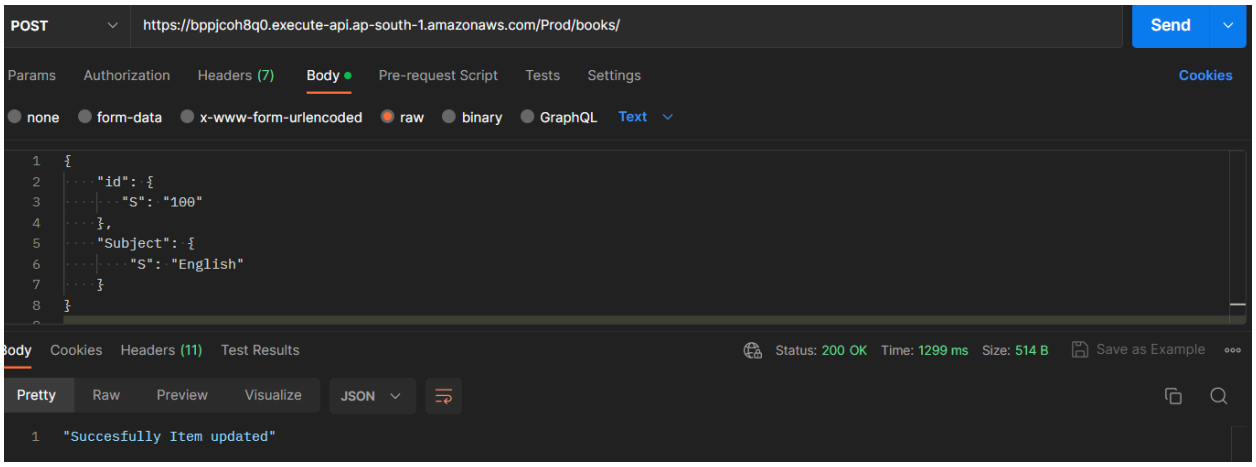
```
[root@ip-172-31-0-37 APIGateway]# ls
DynamoDB_CRUD  events  README.md  samconfig.toml  template.yaml
[root@ip-172-31-0-37 APIGateway]# sam deploy

                Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-1nq2jloczupzn
                A different default S3 bucket can be set in samconfig.toml
                Or by specifying --s3-bucket explicitly.
        Uploading to eb8ba006cc86e6fda30090b0d3ffa70a  1016 / 1016  (100.00%)

        Deploying with following values
        ===============================
        Stack name                   : APIGateway
        Region                       : ap-south-1
        Confirm changeset            : True
        Disable rollback             : False
        Deployment s3 bucket         : aws-sam-cli-managed-default-samclisourcebucket-1nq2jloczupzn
        Capabilities                 : ["CAPABILITY_IAM"]
        Parameter overrides          : {}
        Signing Profiles             : {}

Initiating deployment
=====================
```

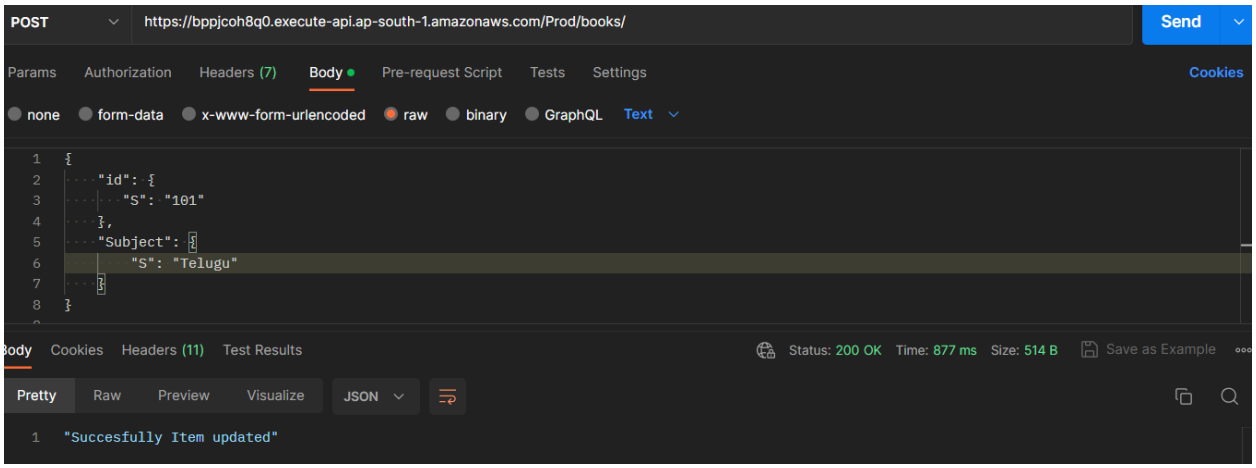
```
CloudFormation stack changeset
-------------------------------------------------------------------------------------------------------------------------------------
Operation                     LogicalResourceId                        ResourceType                   Replacement
-------------------------------------------------------------------------------------------------------------------------------------
+ Add                         BooksDynamoDB                            AWS::DynamoDB::Table           N/A
+ Add                         HelloWorldFunctionBooksDeletePermissionP AWS::Lambda::Permission        N/A
                              rod
+ Add                         HelloWorldFunctionBooksGetPermissionProd AWS::Lambda::Permission        N/A
+ Add                         HelloWorldFunctionBooksGetSpecPermission AWS::Lambda::Permission        N/A
                              Prod
+ Add                         HelloWorldFunctionBooksPostPermissionPro AWS::Lambda::Permission        N/A
                              d
+ Add                         HelloWorldFunctionBooksPutPermissionProd AWS::Lambda::Permission        N/A
+ Add                         HelloWorldFunctionRole                   AWS::IAM::Role                 N/A
+ Add                         HelloWorldFunction                       AWS::Lambda::Function          N/A
+ Add                         ServerlessRestApiDeployment761c8d86ff    AWS::ApiGateway::Deployment    N/A
+ Add                         ServerlessRestApiProdStage               AWS::ApiGateway::Stage         N/A
+ Add                         ServerlessRestApi                        AWS::ApiGateway::RestApi       N/A
-------------------------------------------------------------------------------------------------------------------------------------
```

Resources created successfully with the cloudformation stack

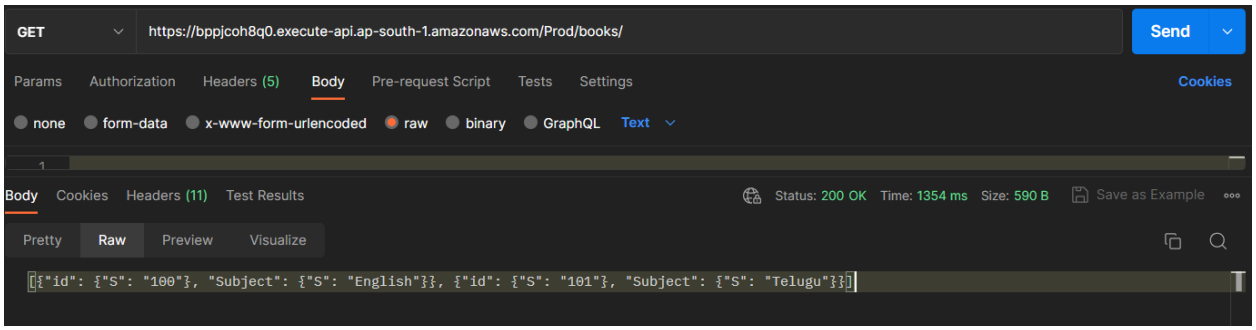


Performing the POST operation on the DynamoDB table with the API Gateway endpoint.

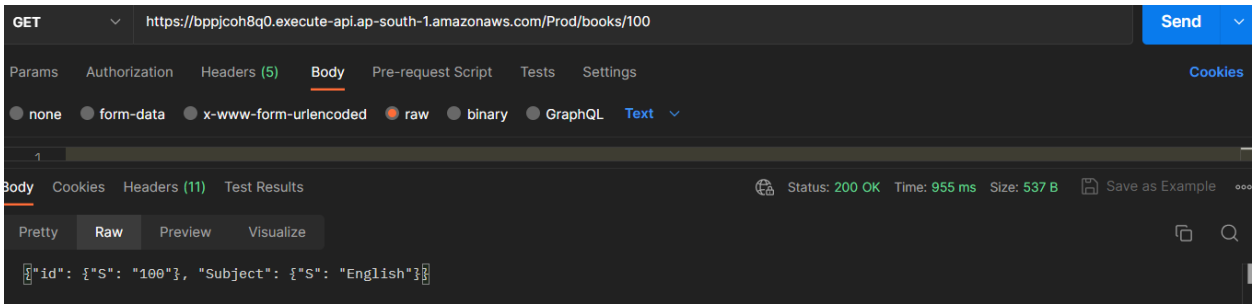




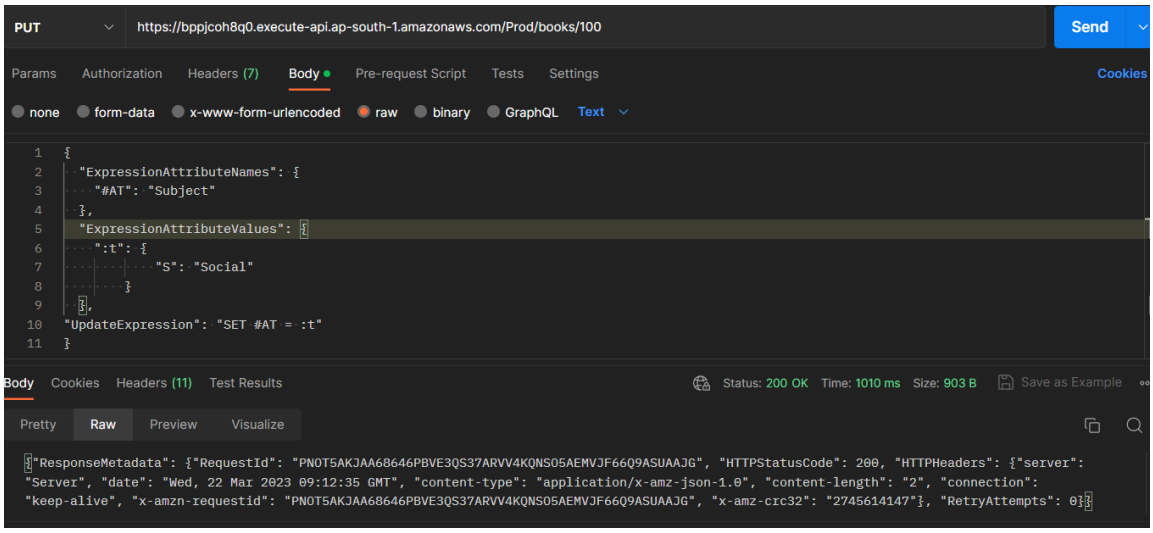
Performing the GET operation on the DynamoDB table with the API Gateway endpoint.
Retrieved all items from the DynamoDB table.



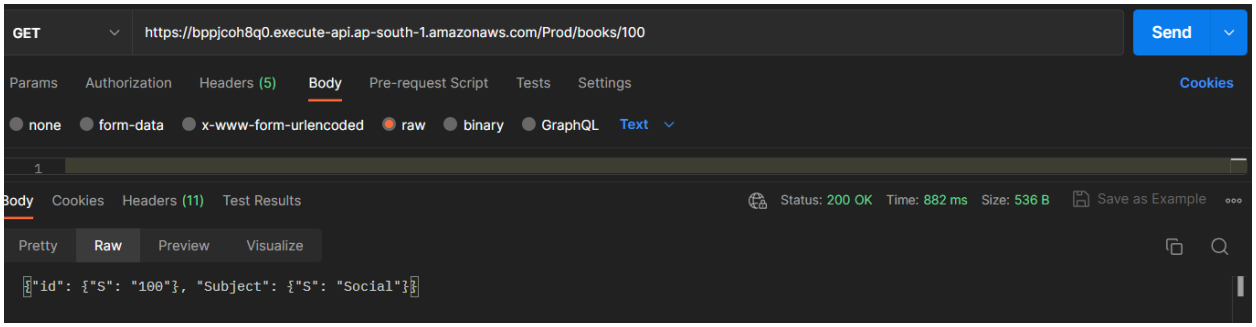
Retrieving the specific item from the DynamoDB table.



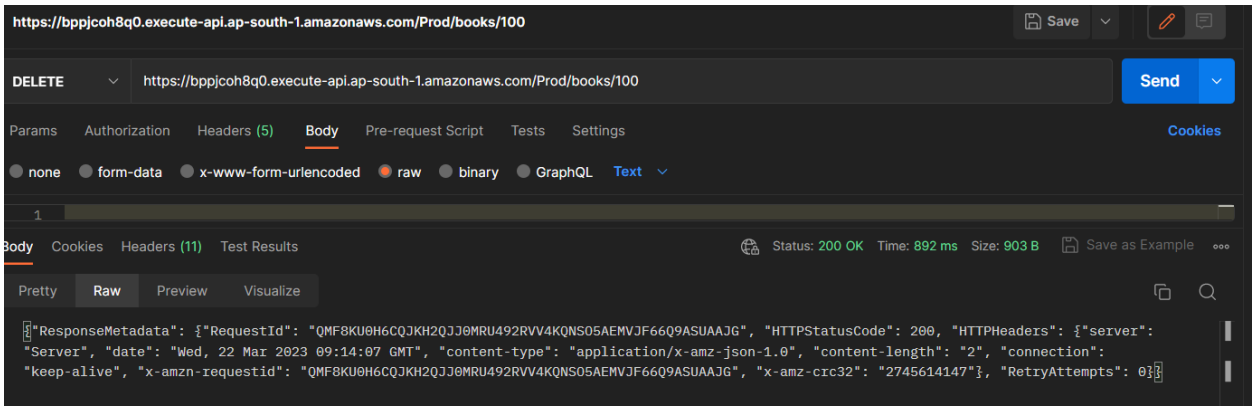
Updating the item in the DynamoDB table.



Checking the update operation



Deleting the item in the DynamoDB table.



Checking the delete operation