Cloud Computing Project Report

Title:

Serverless Personal To-Do List Application using AWS Lambda & DynamoDB

Submitted by:

ADITYA SAI PARISE

Internship Program:

1Stop Cloud Computing Internship – 2025

Institution:

Amrita Vishwa Vidyapeetham, Coimbatore.

ABSTRACT

This project focuses on the design and implementation of a Serverless Personal To-Do List Application using the robust suite of tools and services provided by Amazon Web Services (AWS). The main objective is to leverage cloud-native, serverless technologies to build a fully functioning backend system that eliminates the need for managing traditional server infrastructure. By using a serverless architecture, the application minimizes cost, simplifies scaling, and improves maintainability, making it highly suitable for individual developers and small teams.

The core computing component is AWS Lambda, which serves as the compute layer for executing backend logic in response to client requests. All operations — such as creating, reading, updating, and deleting to-do tasks (CRUD operations) — are handled through discrete Lambda functions, allowing fine-grained control over application behavior. Instead of setting up an entire server or backend framework, the business logic is encapsulated in lightweight and event-driven serverless functions.

Amazon DynamoDB, a fully managed NoSQL database service, is utilized for persistent data storage. It provides seamless scalability and high availability, ensuring that the to-do items are stored and retrieved efficiently regardless of the application's usage patterns. The data schema is kept intentionally simple, with each to-do item having a unique identifier and accompanying metadata such as task description and completion status. This design enables rapid development and avoids the overhead of schema migration, indexing, or server configuration.

One notable departure from traditional cloud tutorials is the intentional avoidance of AWS API Gateway. While API Gateway is a powerful tool for building RESTful APIs on AWS, it can introduce complexity in simpler projects. By excluding API Gateway, the project aims to demonstrate that it is still possible to build and deploy a fully functional serverless application using a streamlined architecture. Instead, the application is directly triggered and tested using AWS CLI, Lambda Console, and tools like Postman to simulate HTTP requests. This decision reflects a minimalist approach to backend development and encourages developers to focus on core functionality first.

The development process involved configuring the AWS Command Line Interface (CLI) for deploying and interacting with the AWS environment. The application also makes use of CloudWatch, AWS's monitoring and logging service, for debugging and performance monitoring. Any issues in Lambda execution, such as missing parameters or logical errors, are easily traceable through CloudWatch logs, which significantly improved the development feedback loop.

Throughout the internship project, emphasis was placed on following best practices in cloud development, such as secure key management, proper IAM role assignment, and tearing down resources to prevent unnecessary billing. The hands-on experience gained through this project contributes significantly to understanding the real-world applications of cloud platforms and the shift towards Infrastructure as Code (IaC) and Function as a Service (FaaS) models.

In conclusion, the Serverless To-Do List Application stands as a practical demonstration of using AWS to build lightweight, scalable, and cost-effective web applications without managing any physical servers. It not only showcases the technical skills acquired during the internship but also reflects a deepened understanding of modern software engineering paradigms in the cloud era.

OBJECTIVE

The primary objective of this internship project is to design, develop, and deploy a Serverless To-Do List Application using Amazon Web Services (AWS) with a special focus on AWS Lambda and Amazon DynamoDB. The project aims to provide hands-on experience in building a cloud-native application while adhering to best practices in serverless computing, cloud automation, and secure resource handling.

The following specific goals were established for the successful completion of this project:

1. To Develop a Serverless To-Do List Application using AWS Lambda and DynamoDB

The core goal is to build a fully functional backend system that performs basic CRUD (Create, Read, Update, Delete) operations on to-do items. This system is to be implemented entirely in a serverless environment, eliminating the need for provisioning or maintaining traditional servers. The backend logic is encapsulated within AWS Lambda functions, and the persistent storage layer is handled by Amazon DynamoDB, a scalable and fully managed NoSQL database.

2. To Manage Cloud Resources Effectively using AWS CLI

This project involves using the AWS Command Line Interface (CLI) to configure the working environment, create and manage AWS services, and deploy the application. Understanding and applying the AWS CLI promotes automation, scripting, and efficiency in handling repetitive cloud tasks. It also reinforces command-line proficiency, which is essential for any modern cloud developer or DevOps practitioner.

3. To Practice Secure Credential Management and Cost Monitoring

As part of AWS best practices, this project emphasizes the importance of securely managing access keys and IAM credentials. Learning how to generate, configure, and safely use these credentials is critical to maintaining application security. Additionally, constant monitoring of resource usage and implementing teardown strategies is practiced to avoid incurring unnecessary costs on the AWS free-tier or budget.

4. To Implement CRUD Functionality via Lambda without using API Gateway

Instead of relying on AWS API Gateway to expose Lambda functions as APIs, this project adopts a simplified architecture where CRUD operations are manually tested using CLI and Postman, thus demonstrating an alternative method of interaction. This approach eliminates the overhead of API Gateway configuration and helps in better understanding the inner workings of Lambda and DynamoDB without any abstraction.

5. To Use Tools like Postman for API Testing and CloudWatch for Debugging

Effective development also includes debugging and testing strategies. Tools like Postman are used to send HTTP requests directly to the deployed Lambda functions, simulating real-world usage. CloudWatch, AWS's built-in logging and monitoring service, is used to trace errors, log function executions, and ensure correct application behavior. This process improves overall observability and helps in building better problem-solving skills.

TOOLS & TECHNOLOGIES USED

The development and deployment of the Serverless Personal To-Do List Application involved a variety of cloud computing services, programming tools, and utilities. Each of these technologies played a crucial role in ensuring that the application was serverless, scalable, and easy to maintain. The selected tools also reflect modern industry practices in cloud development and DevOps.

1. Amazon Web Services (AWS)

Amazon Web Services (AWS) is the cloud platform used to deploy the entire application. It provided the infrastructure, compute services, and database functionalities required to implement the to-do list in a serverless manner.

2. AWS Lambda

AWS Lambda is the compute service used to run backend code without provisioning servers. It executes the application's logic for handling CRUD operations (Create, Read, Update, Delete) on to-do items. Each Lambda function is event-driven and automatically scales based on the number of incoming requests.

3. Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database used to store the to-do list items. It is schema-less, highly scalable, and integrates seamlessly with Lambda functions via the AWS SDK. The primary key used in this application is the ID of the to-do item.

4. AWS Command Line Interface (CLI)

The AWS CLI is used to configure credentials, set default regions, and interact with AWS services from the terminal. It allows for deploying Lambda functions, managing DynamoDB, and performing automated tasks without using the AWS Management Console.

5. Node.js (JavaScript Runtime)

The Lambda functions were written in Node.js, a JavaScript runtime built on Chrome's V8 engine. It is lightweight and well-supported within AWS Lambda, making it ideal for rapid development of serverless APIs.

6. Postman

Postman is used for API testing. Even though the architecture avoids using API Gateway, Postman helped test the Lambda endpoints directly (when invoked via AWS CLI or other triggers) and examine request/response behavior.

7. CloudWatch

Amazon CloudWatch is used to monitor, debug, and log the execution of Lambda functions. It captures logs that help track function errors, input/output parameters, and provides visibility into the behavior of the deployed application.

8. Zip Utility (for Packaging Code)

Before deployment, the function code is zipped and uploaded via the Lambda console. This allows for modular uploads of index.js and related Node.js dependencies.

9. Visual Studio Code (VS Code)

VS Code served as the primary code editor. It provided features like syntax highlighting, linting, and extensions for AWS, making it easier to write, debug, and test the Lambda function locally before deployment.

These tools and technologies were selected based on their support for cloud-native development, serverless architectures, and ease of integration with each other. Together, they offered a robust foundation for building, deploying, and maintaining the application in a modern development workflow.

METHODOLOGY

Step-by-Step Workflow

Step 1: AWS Account Setup & Best Practices

To begin the development process, an AWS account was created by registering with a valid email address and setting up multi-factor authentication (MFA) for added security. The root user account was secured, and an IAM user with limited permissions was created for daily operations.

Once inside the AWS Management Console, time was spent understanding the interface and identifying services such as DynamoDB, Lambda, IAM, and CloudWatch. One of the key lessons was to manage billing by avoiding idle or unused services. Before initiating any task, the cost monitoring dashboard was checked to track active resources.

The instructor emphasized best practices like:

- Creating IAM users instead of using the root account
- Deleting unused Lambda functions and tables
- Avoiding idle EC2 or API Gateway services
- Using free-tier eligible services whenever possible

These best practices were followed consistently throughout the project to ensure efficient and cost-effective cloud usage.

Step 2: Installing AWS CLI

The AWS Command Line Interface (CLI) was installed to interact with AWS services programmatically. The installation was done using the official MSI installer for Windows.

After installation, the CLI was verified using the command aws --version.

To configure the CLI, the following command was executed:

```
aws configure
```

This prompted the user to enter the following details:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name (set to us-east-1)
- Default output format (set to json)

This step enabled secure and flexible access to AWS services directly from the terminal without depending entirely on the web console. It also made it easier to automate deployment and testing tasks.

Step 3: Generating Access Keys

To use the CLI securely, access keys were required. These were generated under the IAM console by navigating to “My Security Credentials.”

The following steps were followed:

- Opened IAM Management Console
- Selected the user account
- Chose the “Security Credentials” tab
- Clicked on “Create access key” under the “Access keys” section

Once generated, the Access Key ID and Secret Access Key were securely copied and used for CLI configuration. These credentials allow the AWS CLI to authenticate and perform actions on behalf of the user without exposing sensitive data unnecessarily.

Proper care was taken not to commit or share the keys publicly, and the credentials were rotated if there were any doubts of misuse.

Step 4: Creating DynamoDB Table

The next major component was setting up the database. A new table was created using the AWS Console under the DynamoDB service.

Details of the table configuration:

- Table Name: todo-table
- Primary Key (Partition Key): id (Data type: String)
- No Sort Key
- Billing Mode: On-Demand (to simplify throughput management)
- Auto-scaling: Enabled for read/write capacity

No Global Secondary Indexes (GSI) or Local Secondary Indexes (LSI) were used in this simple implementation. The table was intended to store each to-do item with attributes like task, status, and timestamp.

This step laid the foundation for persistent, serverless storage without managing database servers.

Step 5: Creating and Uploading Lambda Function

Once the database was ready, a Lambda function was created to handle the application's backend logic.

Details of the function:

- Function Name: todoLambda
- Runtime: Node.js 18.x
- Permissions: Lambda execution role with DynamoDB full access
- Deployment Method: Upload a zip file containing the code

The function's IAM role was configured to allow it to read and write to DynamoDB. The function was created in the AWS Lambda console, and a deployment package was prepared by zipping index.js.

The zipped code was uploaded manually during the creation process. After deployment, the Lambda function was ready to be tested with sample inputs through Postman or CLI.

Further testing, debugging, and iterations were done using CloudWatch Logs.

CODE IMPLEMENTATION

Lambda Function (index.js) – Node.js 18.x Runtime

```
const AWS = require('aws-sdk');

const dynamoDB = new AWS.DynamoDB.DocumentClient();

const TABLE_NAME =  "todoTableTwo";


exports.handler = async (event) => {

  const httpMethod = event.httpMethod;

  const body = event.body ? JSON.parse(event.body) : {};
  switch(httpMethod){
    case "POST":
      const newTask = {
        id:body.id,
        task:body.task,
        completed:false
      };
      await dynamoDB.put({TableName:TABLE_NAME, Item:newTask}).promise();
      return {statusCode:201,body:JSON.stringify(newTask)};
    case "GET":
      const tasks = await dynamoDB.scan({TableName:TABLE_NAME}).promise();
      return {statusCode:200,body:JSON.stringify(tasks.Items)};
    case "PUT":
      await dynamoDB.update({
        TableName : TABLE_NAME,
        Key : {id:body.id},
        UpdateExpression : "set task=:task, completed =:completed",
        ExpressionAttributevalues : {
          ":task":body.task,
          ":completed":body.completed
        }
      }).promise();
      return {statusCode:201,body:JSON.stringify({message:"Task Updated"})};
    case "DELETE":
      await dynamoDB.delete({TableName:TABLE_NAME,Key:{id:body.id}}).promise();
      return {statusCode:200,body:JSON.stringify({message : "Task Deleted"})};
    default:
      return {statusCode:400,body:JSON.stringify({message:"Invalid Request"})}
  }
}
```

Key Points:

- Method Detection: Lambda distinguishes between POST, GET, PUT, and DELETE methods based on the event.httpMethod field.
- DynamoDB Client: AWS.DynamoDB.DocumentClient() is used for high-level interactions with DynamoDB.
- Error Handling: Try-catch is used to handle unexpected issues gracefully and return HTTP 500 on failure.
- JSON Response: All outputs are returned as JSON to ensure compatibility with API clients like Postman.

CODE IMPLEMENTATION

Package.json

```
{
  "name": "lambda_1",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "aws-sdk": "^2.1692.0"
  }
}
```

Package- lock.jason :-

```
{
 "name": "lambda_1",
 "version": "1.0.0",
 "lockfileVersion": 3,
 "requires": true,
 "packages": {
  "": {
   "name": "lambda_1",
   "version": "1.0.0",
   "license": "ISC",
   "dependencies": {
    "aws-sdk": "^2.1692.0"
   }
  },
  "node_modules/available-typed-arrays": {
   "version": "1.0.7",
   "resolved": "https://registry.npmjs.org/available-typed-arrays/-/available-typed-arrays-
1.0.7.tgz",
   "integrity": "sha512-
wvUjBtSGN7+7SjNpq/9M2Tg350UZD3q62IFZLbRAR1bSMlCo1ZaeW+BJ+D090e4hIIZL
BcTDWe4Mh4jvUDajzQ==",
   "license": "MIT",
   "dependencies": {
    "possible-typed-array-names": "^1.0.0"
   },
   "engines": {
    "node": ">= 0.4"
   },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/aws-sdk": {
    "version": "2.1692.0",
    "resolved": "https://registry.npmjs.org/aws-sdk/-/aws-sdk-2.1692.0.tgz",
    "integrity": "sha512-
x511uiJ/57FIsbgUe5csJ13k3uzu25uWQE+XqfBis/sB0SFoiElJWXRkgEAUh0U6n40eT3ay5
Ue4oPkRMu1LYw==",
    "hasInstallScript": true,
    "license": "Apache-2.0",
    "dependencies": {
      "buffer": "4.9.2",
      "events": "1.1.1",
      "ieee754": "1.1.13",
      "jmespath": "0.16.0",
      "querystring": "0.2.0",
      "sax": "1.2.1",
      "url": "0.10.3",
      "util": "^0.12.4",
      "uuid": "8.0.0",
      "xml2js": "0.6.2"
    },
    "engines": {
      "node": ">= 10.0.0"
    }
  },
  "node_modules/base64-js": {
    "version": "1.5.1",
    "resolved": "https://registry.npmjs.org/base64-js/-/base64-js-1.5.1.tgz",
      "integrity": "sha512-
AKpaYlHn8t4SVbOHCy+b5+KKgvR4vrsD8vbvrbiQJps7fKDTkjkDry6ji0rUJjC0kzbNePL
wzxq8iypo41qeWA==",
      "funding": [
        {
          "type": "github",
          "url": "https://github.com/sponsors/feross"
        },
        {
          "type": "patreon",
          "url": "https://www.patreon.com/feross"
        },
        {
          "type": "consulting",
          "url": "https://feross.org/support"
        }
      ],
      "license": "MIT"
    },
    "node_modules/buffer": {
      "version": "4.9.2",
      "resolved": "https://registry.npmjs.org/buffer/-/buffer-4.9.2.tgz",
      "integrity": "sha512-
xq+q3SRMOxGivLhBNaUdC64hDTQwejJ+H0T/NB1XMtTVEwNTrfFF3gAxiyW0Bu/xW
EGhjVKgUcMhCrUy2+uCWg==",
      "license": "MIT",
      "dependencies": {
        "base64-js": "^1.0.2",
        "ieee754": "^1.1.4",
        "isarray": "^1.0.0"
      }
    },
    "node_modules/call-bind": {
      "version": "1.0.8",
      "resolved": "https://registry.npmjs.org/call-bind/-/call-bind-1.0.8.tgz",
      "integrity": "sha512-
oKlSFMcMwpUg2ednkhQ454wfWiU/ul3CkJe/PEHcTKuiX6RpbehUiFMXu13HalGZxfUw
CQzZG747YXBn1im9ww==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.0",
        "es-define-property": "^1.0.0",
        "get-intrinsic": "^1.2.4",
        "set-function-length": "^1.2.2"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/call-bind-apply-helpers": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/call-bind-apply-helpers/-/call-bind-apply-helpers-
1.0.2.tgz",
      "integrity": "sha512-
Sp1ablJ0ivDkSzjcaJdxEunN5/XvksFJ2sMBFfq6x0ryhQV/2b/KwFe21cMpmHtPOSij8K99/
wSfoEuTObmuMQ==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0",
        "function-bind": "^1.1.2"
      },
    "engines": {
      "node": ">= 0.4"
    }
  },
  "node_modules/call-bound": {
    "version": "1.0.4",
    "resolved": "https://registry.npmjs.org/call-bound/-/call-bound-1.0.4.tgz",
    "integrity": "sha512-
+ys997U96po4Kx/ABpBCqhA9EuxJaQWDQg7295H4hBphv3IZg0boBKuwYpt4YXp6MZ5
AmZQnU/tyMTlRpaSejg==",
    "license": "MIT",
    "dependencies": {
      "call-bind-apply-helpers": "^1.0.2",
      "get-intrinsic": "^1.3.0"
    },
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/define-data-property": {
    "version": "1.1.4",
    "resolved": "https://registry.npmjs.org/define-data-property/-/define-data-property-
1.1.4.tgz",
    "integrity": "sha512-
rBMvIzlpA8v6E+SJZoo++HAYqsLrkg7MSfIinMPFhmkorw7X+dOXVJQs+QT69zGkzMyf
DnIMN2Wid1+NbL3T+A==",
    "license": "MIT",
    "dependencies": {
      "es-define-property": "^1.0.0",
        "es-errors": "^1.3.0",
        "gopd": "^1.0.1"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/dunder-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/dunder-proto/-/dunder-proto-1.0.1.tgz",
      "integrity": "sha512-
KIN/nDJBQRcXw0MLVhZE9iQHmG68qAVIBg9CqmUYjmQIhgij9U5MFvrqkUL5FbtyyzZuOeOt0zdeRe4UY7ct+A==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.1",
        "es-errors": "^1.3.0",
        "gopd": "^1.2.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-define-property": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/es-define-property/-/es-define-property-1.0.1.tgz",
      "integrity": "sha512-
e3nRfgfUZ4rNGL232gUgX06QNyyez04KdjFrF+LTRoOXmrOgFKDg4BCdsjW8EnT69eqdYGmRpJwiPVYNrCaW3g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-errors": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/es-errors/-/es-errors-1.3.0.tgz",
      "integrity": "sha512-
Zf5H2Kxt2xjTvbJvP2ZWLEICxA6j+hAmMzIlypy4xcBg1vKVnx89Wy0GbS+kf5cwCVFFz
dCFh2XSCFNULS6csw==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-object-atoms": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/es-object-atoms/-/es-object-atoms-1.1.1.tgz",
      "integrity": "sha512-
FGgH2h8zKNim9ljj7dankFPcICIK9Cp5bm+c2gQSYePhpaG5+esrLODihIorn+Pe6FGJzWh
XQotPv73jTaldXA==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/events": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/events/-/events-1.1.1.tgz",
      "integrity": "sha512-
kEcvvCBByWXGnZy6JUlgAp2gBIUjfCAV6P6TgT1/aaQKcmuAEC4OZTV1I4EWQLz2gx
Zw76atuVyvHhTxvi0Flw==",
      "license": "MIT",
      "engines": {
        "node": ">=0.4.x"
      }
    },
    "node_modules/for-each": {
      "version": "0.3.5",
      "resolved": "https://registry.npmjs.org/for-each/-/for-each-0.3.5.tgz",
      "integrity": "sha512-
dKx12eRCVIzqCxFGplyFKJMPvLEWgmNtUrpTiJIR5u97zEhRG8ySrtboPHZXx7daLxQVr
l643cTzbab2tkQjxg==",
      "license": "MIT",
      "dependencies": {
        "is-callable": "^1.2.7"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/function-bind": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/function-bind/-/function-bind-1.1.2.tgz",
      "integrity": "sha512-
7XHNxH7qX9xG5mIwxkhumTox/MIRNcOgDrxWsMt2pAr23WHp6MrRlN7FBSFpCpr+o
VO0F744iUgR82nJMfG2SA==",
      "license": "MIT",
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/get-intrinsic": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/get-intrinsic/-/get-intrinsic-1.3.0.tgz",
      "integrity": "sha512-
9fSjSaos/fRIVIp+xSJlE6lfwhES7LNtKaCBIamHsjr2na1BiABJPo0mOjjz8GJDURarmCPGq
aiVg5mfjb98CQ==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.2",
        "es-define-property": "^1.0.1",
        "es-errors": "^1.3.0",
        "es-object-atoms": "^1.1.1",
        "function-bind": "^1.1.2",
        "get-proto": "^1.0.1",
        "gopd": "^1.2.0",
        "has-symbols": "^1.1.0",
        "hasown": "^2.0.2",
        "math-intrinsics": "^1.1.0"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/get-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/get-proto/-/get-proto-1.0.1.tgz",
      "integrity": "sha512-
sTSfBjoXBp89JvIKIefqw7U2CCebsc74kiY6awiGogKtoSGbgjYE/G/+l9sF3MWFPNc9IcoO
C4ODfKHfxFmp0g==",
      "license": "MIT",
      "dependencies": {
        "dunder-proto": "^1.0.1",
        "es-object-atoms": "^1.0.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/gopd": {
      "version": "1.2.0",
      "resolved": "https://registry.npmjs.org/gopd/-/gopd-1.2.0.tgz",
      "integrity": "sha512-
ZUKRh6/kUFoAiTAtTYPZJ3hw9wNxx+BIBOijnlG9PnrJsCcSjs1wyyD6vJpaYtgnzDrKYR
Sqf3OO6Rfa93xsRg==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/has-property-descriptors": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/has-property-descriptors/-/has-property-
descriptors-1.0.2.tgz",
      "integrity": "sha512-
55JNKuIW+vq4Ke1BjOTjM2YctQIvCT7GFzHwmfZPGo5wnrgkid0YQtnAleFSqumZm4az
3n2BS+erby5ipJdgrg==",
      "license": "MIT",
      "dependencies": {
        "es-define-property": "^1.0.0"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/has-symbols": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/has-symbols/-/has-symbols-1.1.0.tgz",
      "integrity": "sha512-
1cDNdwJ2Jaohmb3sg4OmKaMBwuC48sYni5HUw2DvsC8LjGTLK9h+eb1X6RyuOHe4hT
0ULCW68iomhjUoKUqlPQ==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/has-tostringtag": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/has-tostringtag/-/has-tostringtag-1.0.2.tgz",
      "integrity": "sha512-
NqADB8VjPFLM2V0VvHUewwwsw0ZWBaIdgo+ieHtK3hasLz4qeCRjYcqfB6AQrBggRK
ppKF8L52/VqdVsO47Dlw==",
      "license": "MIT",
      "dependencies": {
        "has-symbols": "^1.0.3"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/hasown": {
      "version": "2.0.2",
      "resolved": "https://registry.npmjs.org/hasown/-/hasown-2.0.2.tgz",
      "integrity": "sha512-
0hJU9SCPvmMzIBdZFqNPXWa6dqh7WdH0cII9y+CyS8rG3nL48Bclra9HmKhVVUHyPW
NH5Y7xDwAB7bfgSjkUMQ==",
      "license": "MIT",
      "dependencies": {
        "function-bind": "^1.1.2"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/ieee754": {
      "version": "1.1.13",
      "resolved": "https://registry.npmjs.org/ieee754/-/ieee754-1.1.13.tgz",
      "integrity": "sha512-
4vf7I2LYV/HaWerSo3XmlMkp5eZ83i+/CDluXi/IGTs/O1sejBNhTtnxzmRZfvOUqj7lZjqHk
eTvpgSFDlWZTg==",
      "license": "BSD-3-Clause"
```

```
  },
  "node_modules/inherits": {
    "version": "2.0.4",
    "resolved": "https://registry.npmjs.org/inherits/-/inherits-2.0.4.tgz",
    "integrity": "sha512-
k/vGaX4/Yla3WzyMCvTQOXYeIHvqOKtnqBduzTHpzpQZzAskKMhZ2K+EnBiSM9zGSo
IFeMpXKxa4dYeZIQqewQ==",
    "license": "ISC"
  },
  "node_modules/is-arguments": {
    "version": "1.2.0",
    "resolved": "https://registry.npmjs.org/is-arguments/-/is-arguments-1.2.0.tgz",
    "integrity": "sha512-
7bVbi0huj/wrIAOzb8U1aszg9kdi3KN/CyU19CTI7tAoZYEZoL9yCDXpbXN+uPsuWnP02c
yug1gleqq+TU+YCA==",
    "license": "MIT",
    "dependencies": {
      "call-bound": "^1.0.2",
      "has-tostringtag": "^1.0.2"
    },
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/is-callable": {
    "version": "1.2.7",
    "resolved": "https://registry.npmjs.org/is-callable/-/is-callable-1.2.7.tgz",
      "integrity": "sha512-
1BC0BVFhS/p0qtw6enp8e+8OD0UrK0oFLztSjNzhcKA3WDuJxxAPXzPuPtKkjEY9UUoE
WlX/8fgKeu2S8i9JTA==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/is-generator-function": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/is-generator-function/-/is-generator-function-
1.1.0.tgz",
      "integrity": "sha512-
nPUB5km40q9e8UfN/Zc24eLlzdSf9OfKByBw9CIdw4H1giPMeA0OIJvbchsCu4npfI2QcM
VBsGEBHKZ7wLTWmQ==",
      "license": "MIT",
      "dependencies": {
        "call-bound": "^1.0.3",
        "get-proto": "^1.0.0",
        "has-tostringtag": "^1.0.2",
        "safe-regex-test": "^1.1.0"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/is-regex": {
      "version": "1.2.1",
      "resolved": "https://registry.npmjs.org/is-regex/-/is-regex-1.2.1.tgz",
      "integrity": "sha512-
MjYsKHO5O7mCsmRGxWcLWheFqN9DJ/2TmngvjKXihe6efViPqc274+Fx/4fYj/r03+ESv
BdTXK0V6tA3rgez1g==",
      "license": "MIT",
      "dependencies": {
        "call-bound": "^1.0.2",
        "gopd": "^1.2.0",
        "has-tostringtag": "^1.0.2",
        "hasown": "^2.0.2"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/is-typed-array": {
      "version": "1.1.15",
      "resolved": "https://registry.npmjs.org/is-typed-array/-/is-typed-array-1.1.15.tgz",
      "integrity": "sha512-
p3EcsicXjit7SaskXHs1hA91QxgTw46Fv6EFKKGS5DRFLD8yKnohjF3hxoju94b/OcMZoQ
ukzpPpBE9uLVKzgQ==",
      "license": "MIT",
      "dependencies": {
        "which-typed-array": "^1.1.16"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/isarray": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/isarray/-/isarray-1.0.0.tgz",
      "integrity": "sha512-VLghIWNM6ELQzo7zwmcg0NmTVyWKYjvIeM83yjp0wRDTmUnrM678fQbcKBo6n2CJEF0szoG//ytg+TKla89ALQ==",
      "license": "MIT"
    },
    "node_modules/jmespath": {
      "version": "0.16.0",
      "resolved": "https://registry.npmjs.org/jmespath/-/jmespath-0.16.0.tgz",
      "integrity": "sha512-9FzQjJ7MATs1tSpnco1K6ayiYE3figslrXA72G2HQ/n76RzvYlofyi5QM+iX4YRs/pu3yzxlVQSST23+dMDknw==",
      "license": "Apache-2.0",
      "engines": {
        "node": ">= 0.6.0"
      }
    },
    "node_modules/math-intrinsics": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/math-intrinsics/-/math-intrinsics-1.1.0.tgz",
      "integrity": "sha512-/IXtbwEk5HTPyEwyKX6hGkYXxM9nbj64B+ilVJnC/R6B0pH5G4V3b0pVbL7DBj4tkhBAppbQUlf6F6Xl9LHu1g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/possible-typed-array-names": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/possible-typed-array-names/-/possible-typed-array-
names-1.1.0.tgz",
      "integrity": "sha512-
/+5VFTchJDoVj3bhoqi6UeymcD00DAwb1nJwamzPvHEszJ4FpF6SNNbUbOS8yI56qHzdV
8eK0qEfOSiodkTdxg==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/punycode": {
      "version": "1.3.2",
      "resolved": "https://registry.npmjs.org/punycode/-/punycode-1.3.2.tgz",
      "integrity": "sha512-
RofWgt/7fL5wP1Y7fxE7/EmTLzQVnB0ycyibJ0OOHIlJqTNzglYFxVwETOcIoJqJmpDXJ9
xImDv+Fq34F/d4Dw==",
      "license": "MIT"
    },
    "node_modules/querystring": {
      "version": "0.2.0",
      "resolved": "https://registry.npmjs.org/querystring/-/querystring-0.2.0.tgz",
      "integrity": "sha512-
X/xY82scca2tau62i9mDyU9K+I+djTMUsvwf7xnUX5GLvVzgJybOJf4Y6o9Zx3oJK/LSXg
5tTZBjwzqVPaPO2g==",
      "deprecated": "The querystring API is considered Legacy. new code should use the
URLSearchParams API instead.",
      "engines": {
        "node": ">=0.4.x"
    }
  },
  "node_modules/safe-regex-test": {
    "version": "1.1.0",
    "resolved": "https://registry.npmjs.org/safe-regex-test/-/safe-regex-test-1.1.0.tgz",
    "integrity": "sha512-
x/+Cz4YrimQxQccJf5mKEbIa1NzeCRNI5Ecl/ekmlYaampdNLPalVyIcCZNNH3MvmqBug
V5TMYZXv0ljslUlaw==",
    "license": "MIT",
    "dependencies": {
      "call-bound": "^1.0.2",
      "es-errors": "^1.3.0",
      "is-regex": "^1.2.1"
    },
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/sax": {
    "version": "1.2.1",
    "resolved": "https://registry.npmjs.org/sax/-/sax-1.2.1.tgz",
    "integrity": "sha512-
8I2a3LovHTOpm7NV5yOyO8IHqgVsfK4+UuySrXU8YXkSRX7k6hCV9b3HrkKCr3nMpg
j+0bmocaJJWpvp1oc7ZA==",
    "license": "ISC"
  },
  "node_modules/set-function-length": {
    "version": "1.2.2",
    "resolved": "https://registry.npmjs.org/set-function-length/-/set-function-length-1.2.2.tgz",
      "integrity": "sha512-
pgRc4hJ4/sNjWCSS9AmnS40x3bNMDTknHgL5UaMBTMyJnU90EgWh1Rz+MC9eFu4Bu
N/UwZjKQuY/1v3rM7HMfg==",
      "license": "MIT",
      "dependencies": {
        "define-data-property": "^1.1.4",
        "es-errors": "^1.3.0",
        "function-bind": "^1.1.2",
        "get-intrinsic": "^1.2.4",
        "gopd": "^1.0.1",
        "has-property-descriptors": "^1.0.2"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/url": {
      "version": "0.10.3",
      "resolved": "https://registry.npmjs.org/url/-/url-0.10.3.tgz",
      "integrity": "sha512-
hzSUW2q06EqL1gKM/a+obYHLIO6ct2hwPuviqTTOcfFVc61UbfJ2Q32+uGL/HCPxKqrd
GB5QUwIe7UqlDgwsOQ==",
      "license": "MIT",
      "dependencies": {
        "punycode": "1.3.2",
        "querystring": "0.2.0"
      }
    },
    "node_modules/util": {
      "version": "0.12.5",
      "resolved": "https://registry.npmjs.org/util/-/util-0.12.5.tgz",
      "integrity": "sha512-
kZf/K6hEIrWHI6XqOFUiiMa+79wE/D8Q+NCNAWclkyg3b4d2k7s0QGepNjiABc+aR3N1
PAyHL7p6UcLY6LmrnA==",
      "license": "MIT",
      "dependencies": {
        "inherits": "^2.0.3",
        "is-arguments": "^1.0.4",
        "is-generator-function": "^1.0.7",
        "is-typed-array": "^1.1.3",
        "which-typed-array": "^1.1.2"
      }
    },
    "node_modules/uuid": {
      "version": "8.0.0",
      "resolved": "https://registry.npmjs.org/uuid/-/uuid-8.0.0.tgz",
      "integrity": "sha512-
jOXGuXZAWdsTH7eZLtyXMqUb9EcWMGZNbL9YcGBJl4MH4nrxHmZJhEHvyLFrkxo+
28uLb/NYRcStH48fnD0Vzw==",
      "license": "MIT",
      "bin": {
        "uuid": "dist/bin/uuid"
      }
    },
    "node_modules/which-typed-array": {
      "version": "1.1.19",
      "resolved": "https://registry.npmjs.org/which-typed-array/-/which-typed-array-1.1.19.tgz",
      "integrity": "sha512-
rEvr90Bck4WZt9HHFC4DJMsjvu7x+r6bImz0/BrbWb7A2djJ8hnZMrWnHo9F8ssv0OMEra
sDhftrfROTyqSDrw==",
      "license": "MIT",
      "dependencies": {
        "available-typed-arrays": "^1.0.7",
      "call-bind": "^1.0.8",
      "call-bound": "^1.0.4",
      "for-each": "^0.3.5",
      "get-proto": "^1.0.1",
      "gopd": "^1.2.0",
      "has-tostringtag": "^1.0.2"
    },
    "engines": {
      "node": ">= 0.4"
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/xml2js": {
    "version": "0.6.2",
    "resolved": "https://registry.npmjs.org/xml2js/-/xml2js-0.6.2.tgz",
    "integrity": "sha512-
T4rieHaC1EXcES0Kxxj4JWgaUQHDk+qwHcYOCFHfiwKz7tOVPLq7Hjq9dM1WCMhylq
MEfP7hMcOIChvotiZegA==",
    "license": "MIT",
    "dependencies": {
      "sax": ">=0.6.0",
      "xmlbuilder": "~11.0.0"
    },
    "engines": {
      "node": ">=4.0.0"
    }
  },
  "node_modules/xmlbuilder": {
    "version": "11.0.1",
      "resolved": "https://registry.npmjs.org/xmlbuilder/-/xmlbuilder-11.0.1.tgz",
      "integrity": "sha512-
fDlsI/kFEx7gLvbecc0/ohLG50fugQp8ryHzMTuW9vSa1GJ0XYWKnhsUx7oie3G98+r56aT
QIUB4kht42R3JvA==",
      "license": "MIT",
      "engines": {
        "node": ">=4.0"
      }
    }
  }
}
```

Key Insights

- HTTP Method Routing: Handles POST, GET, PUT, DELETE, returning 200 on success or 400 for unsupported methods.
- DynamoDB Interaction: Uses DocumentClient for simplified data operations.
- Robust Error Handling: try-catch blocks ensure clean error responses.
- JSON I/O: Ensures responses are client-friendly and interoperable.

Sample Postman Outputs

POST /todoLambda

Body: {"id":"1","task":"Buy textbooks"}

Response: 200 OK

{"id":"1","task":"Buy textbooks","completed":false}

[Insert Screenshot – "POST to-do item"]

GET /todoLambda

Response: 200 OK

[{"id":"1","task":"Buy textbooks","completed":false}]

[Insert Screenshot – "GET all items"]

PUT /todoLambda

Body: {"id":"1","completed":true}

Response: 200 OK

{"Attributes":{"completed":true}}

[Insert Screenshot – "PUT update item"]

POSTMAN TESTING OUTPUT

Postman Testing Results

POST Request

- Sent body:

```
{
  "id": "1",
  "task": "Buy Milk"
}
```

- Response: 200 OK
  Task Created Successfully

GET Request

- Called endpoint to fetch all tasks.
- Response: 200 OK

json

CopyEdit

```
[
  {
    "id": "1",
    "task": "Buy Milk",
    "completed": false
  }
]
```

POSTMAN PUT & DELETE + CLOUDWATCH DEBUGGING

PUT Request

- Initial body:

```
{
  "id": "1",
  "completed": true
}
```

- Response: Validation Error
  Issue: Incorrect Update Expression or value binding.
- After fixing code and redeploying:
  - Response: 200 OK
    Task Updated Successfully

DELETE Request

- Sent body:

```
{
  "id": "1"
}
```

- Response: 200 OK
  Task Deleted Successfully

CLOUDWATCH LOGS + DEBUGGING LEARNINGS

CloudWatch Logs Insights

- CloudWatch helped track:

    - JSON parsing errors
    - Missing expression values
    - Logical errors in handler methods
    - Invalid HTTP methods used

Debugging Learnings

Redeployment Process

1. Edit index.js locally
2. Zip updated file
3. Upload via AWS Lambda Console
4. Click Deploy
5. Retest each operation in Postman

This cycle ensured quick testing and validation of fixes without needing API Gateway.

CHALLENGES FACED

During the development of the Serverless To-Do List Application using AWS Lambda and DynamoDB, several challenges were encountered that enhanced the learning process:

1. Handling AWS IAM Permissions

Configuring the correct IAM role for the Lambda function was crucial. Initially, there were permission errors when trying to access DynamoDB. This was resolved by attaching a policy with dynamodb:* access to the Lambda execution role.

2. Lambda Cold Starts

Occasionally, Lambda functions took longer to respond, especially after periods of inactivity. These cold starts added a few seconds of delay, affecting perceived performance during initial testing.

3. JSON Parsing Errors

There were multiple instances where improperly formatted request bodies led to SyntaxError: Unexpected token errors. Ensuring event.body was parsed only if it existed was a key fix.

4. Incorrect UpdateExpression Syntax

The PUT method required an exact syntax for DynamoDB's UpdateExpression. Mistakes like forgetting ExpressionAttributeValues or incorrect placeholders (e.g., :c instead of actual values) resulted in validation errors.

5. AWS CLI Configuration Issues

Misconfigured AWS CLI settings, such as wrong region (us-east-1 vs us-west-2) or expired credentials, led to authentication failures. Re-running aws configure helped restore proper access.

Conclusion

Through this project, I successfully designed and deployed a scalable, cost-efficient, and serverless to-do list application on Amazon Web Services (AWS). By leveraging Lambda functions for backend logic and DynamoDB for data storage, I eliminated the need for managing traditional servers or infrastructure.

One of the key decisions in this project was intentionally avoiding API Gateway, which encouraged me to think more critically about cloud architecture design and explore alternative communication strategies. This simplification proved beneficial for learning core AWS services deeply without getting overwhelmed by additional layers.

Hands-on usage of AWS CLI, Lambda Console, Postman, and CloudWatch Logs helped reinforce essential cloud engineering practices, especially around debugging, resource provisioning, and access management. Each challenge—whether it was IAM configuration, parsing errors, or Lambda redeployment—contributed to a deeper understanding of real-world cloud workflows.

Future Improvements

To further enhance the project, the following improvements are proposed:

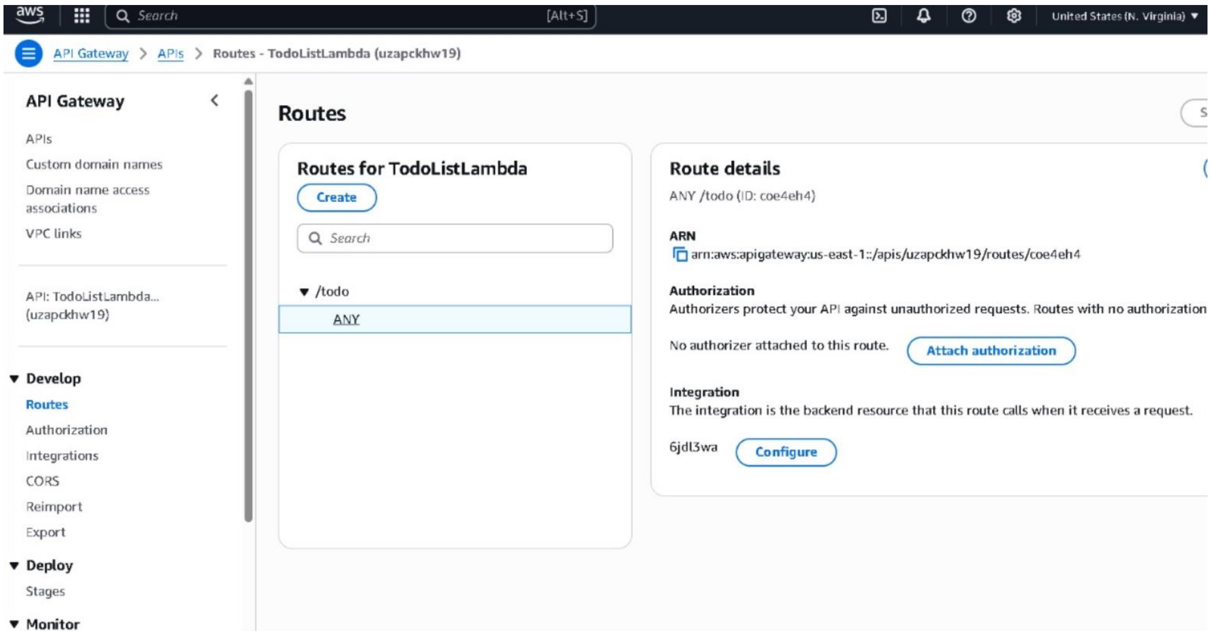
1. User Authentication
   Secure user-level access using AWS Cognito or third-party authentication systems.
2. Web-Based Frontend
   Create a responsive web interface using HTML, CSS, JavaScript, or modern frameworks such as React.
3. Task Analytics
   Track task completion statistics, identify usage patterns, and visualize data using AWS CloudWatch Metrics or external dashboard tools.

References

1. AWS Lambda Documentation
   https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
2. Amazon DynamoDB Developer Guide
   https://docs.aws.amazon.com/amazondynamodb/latest/developerguide
3. AWS CLI Installation and Configuration
   https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
4. Amazon CloudWatch Logs Documentation
   https://docs.aws.amazon.com/AmazonCloudWatch
5. YouTube Tutorial by [Instructor Name]
   (Insert actual name if available)
   https://www.youtube.com
6. Serverless Framework Official Website
   https://www.serverless.com/

Screenshots :-

Api Gateways:-



Routes

Routes for TodoListLambda

▼ /todo
ANY

Route details
ANY /todo (ID: coe4eh4)

ARN
arn:aws:apigateway:us-east-1::/apis/uzapckhw19/routes/coe4eh4

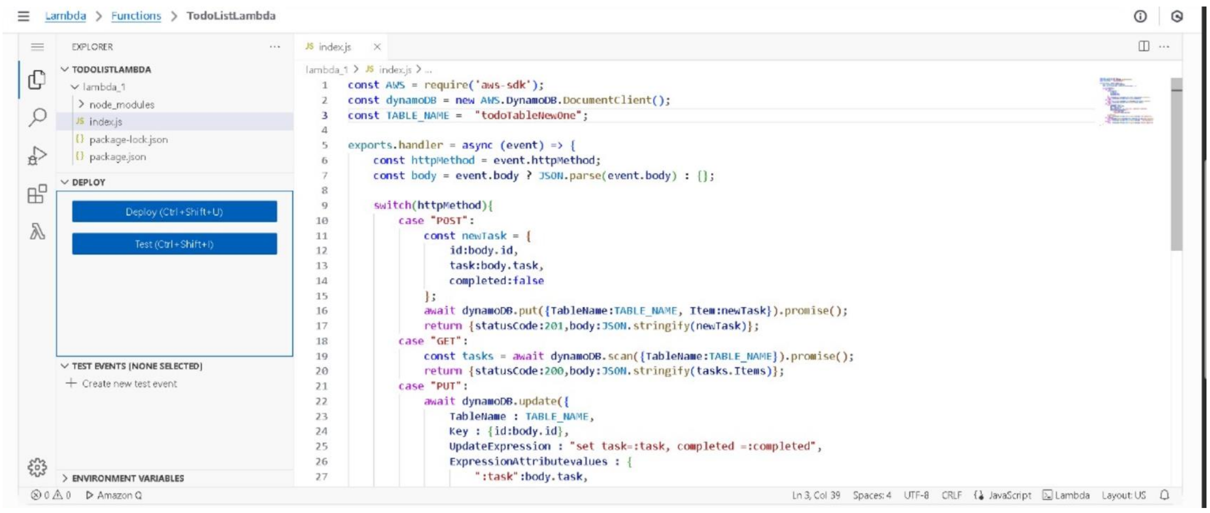
Authorization
Authorizers protect your API against unauthorized requests. Routes with no authorization

No authorizer attached to this route.

Integration
The integration is the backend resource that this route calls when it receives a request.

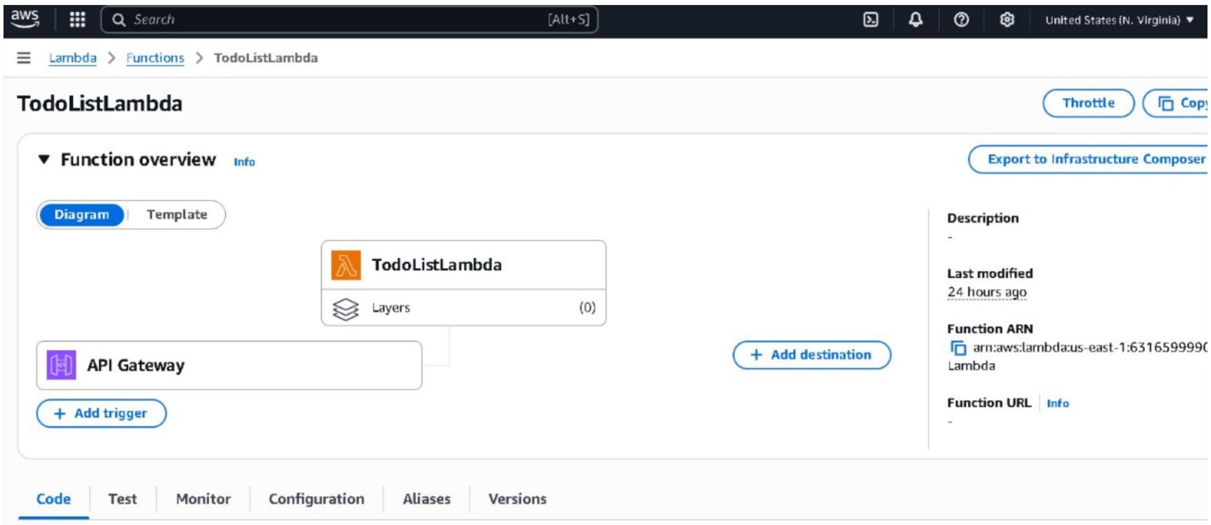
6jdl3wa

Index.js



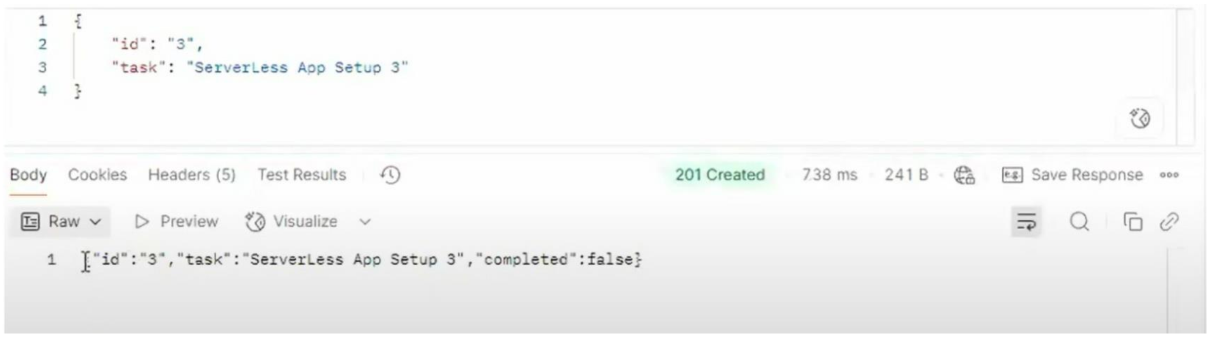
```
const AWS = require('aws-sdk');
const dynamoDB = new AWS.DynamoDB.DocumentClient();
const TABLE_NAME =  "todoTableNewOne";

exports.handler = async (event) => {
    const httpMethod = event.httpMethod;
    const body = event.body ? JSON.parse(event.body) : {};

    switch(httpMethod){
        case "POST":
            const newTask = {
                id:body.id,
                task:body.task,
                completed:false
            };
            await dynamoDB.put({TableName:TABLE_NAME, Item:newTask}).promise();
            return {statusCode:201,body:JSON.stringify(newTask)};
        case "GET":
            const tasks = await dynamoDB.scan({TableName:TABLE_NAME}).promise();
            return {statusCode:200,body:JSON.stringify(tasks.Items)};
        case "PUT":
            await dynamoDB.update({
                TableName : TABLE_NAME,
                Key : {id:body.id},
                UpdateExpression : "set task=:task, completed =:completed",
                ExpressionAttributevalues : {
                    ":task":body.task,
```

Lambda Function:-



Description
-
Last modified
24 hours ago
Function ARN
arn:aws:lambda:us-east-1:6316599990
Lambda
Function URL Info
-

Post :-

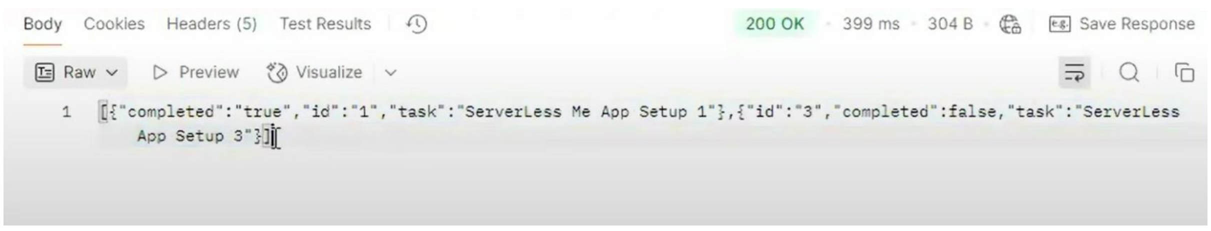


```
{
    "id": "3",
    "task": "ServerLess App Setup 3"
}
```

201 Created 738 ms 241 B

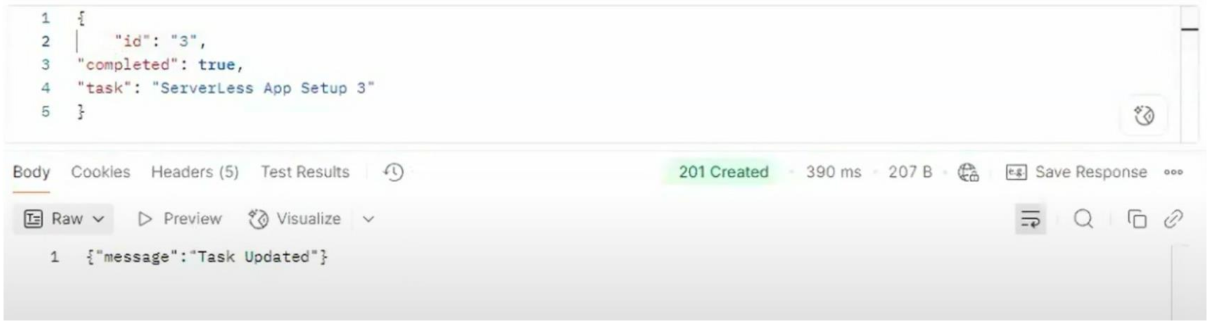
```
{"id":"3","task":"ServerLess App Setup 3","completed":false}
```

Get :-



```
[{"completed":"true","id":"1","task":"ServerLess Me App Setup 1"},{"id":"3","completed":false,"task":"ServerLess
App Setup 3"}]
```

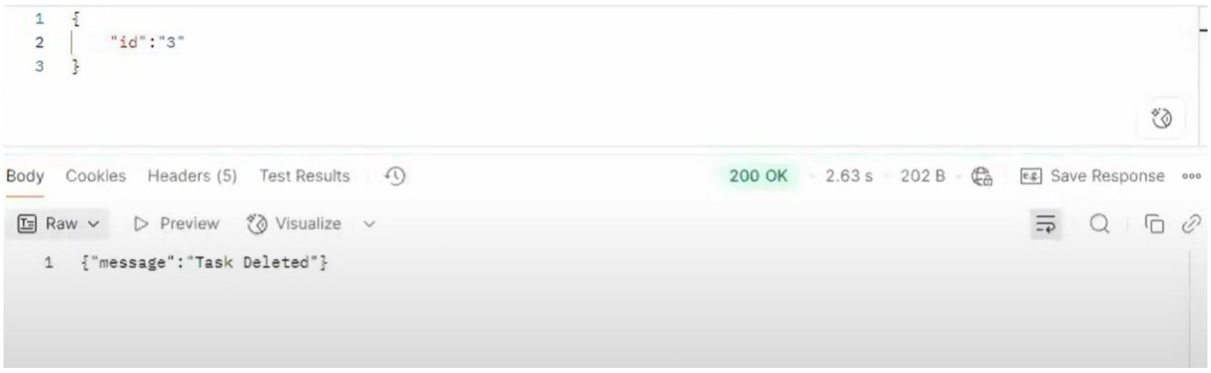
Put:-



```
{
    "id": "3",
"completed": true,
"task": "ServerLess App Setup 3"
}
```

```
{"message":"Task Updated"}
```

Delete:-



```
{
    "id":"3"
}
```

```
{"message":"Task Deleted"}
```