**TO-DO Application**

**Project Documentation**

**Project Overview**

This project is a simple Node.js application to provide a simple API for managing to-do items. It also includes infrastructure as code (IaC) using Terraform to deploy resources such as API Gateway, IAM roles, Lambda functions, S3 buckets, and SSM parameters.

**Setup Instructions**

**Prerequisites**

* Node.js (v18.x or later)
* npm (v6.x or later)
* Terraform (v0.12.x or later)

**Installation**

1. Clone the repository using:

**git clone** [**https://github.com/toxxxx.git/**](https://github.com/toxxxx.git/)

**cd project-directory**

1. Install Node.js dependencies:

**npm install**

1. Initialize Terraform:

**terraform init**

**Usage**

1. Running the Application:

**npm build**

**Deploying Infrastructure**

1. Apply the Terraform configuration:

**terraform apply**

**Directory and File Descriptions**

1. **project-directory/dist/**: The npm build command will compile the TypeScript code and create the dist/ directory along with the compiled JavaScript code.
   * **dist/controllers/todoController.js**: This is a controller file which contains multiple controller functions.
   * **dist/dto/todoModel.js**: This directory contains the preferred model structure for the to-do application.
   * **dist/service/service.js**: This is the service layer directory which includes the business logic for the data exchanged.
   * **dist/utils/utils.js**: Contains utility functions (if any).
2. **project-directory/src/**: This directory contains the TypeScript source code.
3. **project-directory/modules/**: This directory includes all the deployment and IaC codes.
   * **modules/api\_gateway/**: Terraform configuration for API Gateway.
     + **main.tf**: Defines the infrastructure resources for API Gateway.
     + **outputs.tf**: Outputs for API Gateway resources.
     + **variables.tf**: Variables for API Gateway resources.
   * **modules/iam/**: Terraform configuration for IAM roles.
     + **main.tf**: Defines the infrastructure resources for IAM roles.
     + **outputs.tf**: Outputs for IAM resources.
     + **variables.tf**: Variables for IAM resources.
   * **modules/lambda/**: Terraform configuration for Lambda functions.
     + **main.tf**: Defines the infrastructure resources for Lambda functions.
     + **outputs.tf**: Outputs for Lambda resources.
     + **variables.tf**: Variables for Lambda resources.
   * **modules/s3/**: Terraform configuration for S3 buckets.
     + **main.tf**: Defines the infrastructure resources for S3 buckets.
     + **outputs.tf**: Outputs for S3 resources.
     + **variables.tf**: Variables for S3 resources.
   * **modules/ssm\_parameters/**: Terraform configuration for SSM parameters.
     + **main.tf**: Defines the infrastructure resources for SSM parameters.
     + **outputs.tf**: Outputs for SSM resources.
     + **variables.tf**: Variables for SSM resources.
4. **Root Directory Files**:
   * **main.tf**: Main Terraform configuration file for defining infrastructure resources.
   * **outputs.tf**: Terraform outputs file for outputting values after infrastructure deployment.
   * **package.json**: Contains metadata about the Node.js project and its dependencies.
   * **parameters. json**: JSON file containing project parameters.
   * **tsconfig. json**: TypeScript configuration file.
   * **variables.tf**: Terraform variables file for defining input variables.

**Configuring Deployment**

1. **Module – lambda**

**Overview**

This Terraform module prepares, packages, and deploys an AWS Lambda function. It consists of three main components:

1. **Preparing the Lambda function**: Running build commands and installing dependencies.
2. **Creating a ZIP file**: Packaging the Lambda function and its dependencies into a .zip file.
3. **Deploying the Lambda function**: Uploading the .zip file to AWS Lambda.
   1. **Input Variables**

function\_name: - Name of the Lambda function

lambda\_role\_arn: - ARN of the IAM role for Lambda

handler: - Handler for the Lambda function

runtime: - Runtime for the Lambda function

environment\_variables: - Environment variables for the Lambda function

* 1. **Output Variables**

lambda\_function\_name: - Name of the Lambda function

lambda\_invoke\_arn: - Invocation ARN of the Lambda function

* 1. **Using lambda module in project-directory/main.tf**

module "lambda" {

source = "./modules/lambda"

function\_name = "my\_lambda\_function"

lambda\_role\_arn = // specify the arn

handler = "index.handler"

runtime = "nodejs14.x"

environment\_variables = {

"KEY1" = "value1"

"KEY2" = "value2"

}

}

1. **Module – api\_gateway**

**API Overview**

**Resources**

**Todos Resource**

* **Path: /todos**
* **Methods:**
  + **GET**
    - **Integration: AWS Lambda Function (var.get\_todos\_arn)**
  + **POST**
    - **Integration: AWS Lambda Function (var.add\_todo\_arn)**

**Todo Resource**

* **Path: /todos/{id}**
* **Methods:**
  + **PUT**
    - **Integration: AWS Lambda Function (var.update\_todo\_arn)**
  + **DELETE**
    - **Integration: AWS Lambda Function (var.delete\_todo\_arn)**
  1. **Input variables**

api\_name: Name of the API Gateway

api\_description: Description of the API Gateway

get\_todos\_arn: ARN of the Lambda function for getting to-do

add\_todo\_arn: ARN of the Lambda function for adding a to-do

update\_todo\_arn: ARN of the Lambda function for updating a to-do

delete\_todo\_arn: ARN of the Lambda function for deleting a to-do

* 1. **Output variables**

api\_id: ID of the API Gateway REST API

root\_resource\_id: Resource ID of the API's root

execution\_arn: Execution ARN part of the API Gateway stage

invoke\_url: URL to invoke the API pointing to the stage

* 1. **Using api\_gateway module in project-directory/main.tf**

module "api\_gateway" {

source = "Path to module"

api\_name = "Api Name goes here"

api\_description = "description goes here"

get\_todos\_arn = //specify arn here

add\_todo\_arn = //specify arn here

update\_todo\_arn = //specify arn here

delete\_todo\_arn = //specify arn here

}

1. **Module – S3**

**Overview**

This module defines the s3 bucket and file upload to the bucket

* 1. **Input variables**

**bucket\_name:** Name of the S3 bucket

**file\_key:** Key of the file in S3 bucket

**file\_source:** Source path of the file to upload

* 1. **Output variables**

**bucket\_id:** ID of the created S3 bucket

**bucket\_arn:** ARN of the created S3 bucket

**file\_key:** Key of the uploaded file

* 1. **Using s3 module in project-directory/main.tf**

module "s3" {

source = "path to module"

bucket\_name = // your bucket name

file\_key = // file name

file\_source = //path to your file

}

1. **Module – iam**
   1. **Input variables**

**s3\_bucket\_arn:** ARN of the S3 bucket

* 1. **Output variables**

**lambda\_role\_arn:** ARN of the IAM role for Lambda

* 1. **Usage**

**module** "iam" {

source = "Path to module"

s3\_bucket\_arn = //specify arn of bucket

}

**Configuring Source code**

1. **project-destination/src/dto/dto.ts**
   1. **Creating new dto structure**

export interface New\_dto{

// Define the structure

}

1. **project-destination/src/conreoller/todoController.ts**
   1. **Creating new controller**

import { TodoService } from '../service/service';

import { Todo } from '../dto/todoModel';

//Specify other imports here

//Extend the service class

const todoService **=** **new** TodoService()**;**

//specify your new controller here

*export* const deleteTodo **=** *async* (**event:** APIGatewayProxyEvent)**:** Promise<APIGatewayProxyResult> **=>** {

**try** {

**//your statemets**

}

catch(expression){

//Your catch statements

}

1. **project-destination/src/service/service.ts**
   1. **Creating new service**

*import* { S3Client**,** GetObjectCommand**,** PutObjectCommand**,** DeleteObjectCommand } *from* "@aws-sdk/client-s3"**;**

*import* { Todo } *from* '../dto/todoModel'**;**

*// Specify other imports here*

*export* class TodoService {

*// create private variables here*

*// create private variables here*

    constructor() {

//Add initilaization here

    }

*// Helper method to get the current list of Todos*

*private* *async* getTodos()**:** Promise<Todo[]> {

//Write todo logic here

    }

    }

// Write more methods

*public* *async* createTodo(**todo:** Todo)**:** Promise<*void*> {

//access todo here

    }

}

**Cleaning Up**

To remove all deployed resources, navigate to the terraform directory and run:

terraform destroy