Bhushan Mukund Kor D15C

28

Page No.
Date 05/03

Assignment No. 2

Q.1) Create a REST API with the Serverless Framework.

→ Prerequisite:-

Install Node.js and NPM from https://nodejs.org.

Steps for creating a REST API with the serverless frame work:

Step 1:- Install serverless Framework

The serverless Framework is a CLI tool that helps deploy serverless applications, managing resources like Lambda, API Gateway, and more.

Command: npm install -g serverless

This installs the serverless framework globally on your machine. You need this to manage your serverless services and automate deployments.

Step 2: Create a New Serverless Project

The serverless Framework provides templates to scaffold a new serverless service. This will generate the folder structure and basic configuration.

Command:- serverless create --template aws-nodejs --path my-rest-api

cd my-rest-api

This creates a new service using the aws-nodejs template, which is optimized for AWS Lambda with Node.js as the runtime. The --path option creates a new directory called my-rest-api for your project. After that, you change into the project directory.

Step 3: Configure AWS Credentials

The serverless framework requires AWS credentials to deploy your functions and resources (Lambda, API Gateway, etc.) on AWS

command: aws configure

This command sets up the AWS CLI on your local machine by asking for your AWS access key, secret key, default region, and output format. This allows serverless Framework to authenticate and manage AWS resources on your behalf.

Step 4: Define API Routes and Resources in serverless.yml

The serverless.yml file defines your entire application - functions, events (such as HTTP endpoints), and resources. It's the core configuration file for the serveless Framework.

Command :- Modify serverless.yml as follows:

```
service : my-rest-api

provider:
  name : aws
  runtime : nodejs14.x
  region : us-east-1
  stage : dev

functions:
  createItem:
    handler : handler.createItem
    events :
      - http :
          path : item
          method : post

  getItem:
    handler : handler.getItem
    events :
      - http:
          path : item/{id}
          method : get

  updateItem:
    handler : handler.updateItem
    events :
      - http:
          path : item/{id}
          method : put
```

```
deleteItem:
  handler: handler.deleteItem
  events:
    - http:
        path: item/{id}
        method: delete.
```

- This file defines four Lambda functions: createItem, getItem, updateItem, and deleteItem.
- Each function has an associated HTTP method (GET, POST, PUT, DELETE), which is tied to a specific path in the API Gateway.
- Paths like /item/{id} mean that id is a dynamic parameter passed in the URL.

Step 5: Write Lambda function in handler.js

Lambda function handle the logic for each API route. They get triggered when a request is made to the API Gateway.

command: In handler.js, write functions like the following:

```
'use strict';

module.exports.createItem = async (event) => {
  const body = JSON.parse(event.body);

  return {
```

```
    statusCode: 200,
    body: JSON.stringify({
      message: "Item created successfully!",
      item: body
    }),
  };
};

module.exports.getItem = async (event) => {
  const id = event.pathParameters.id;

  return {
    statusCode: 200,
    body: JSON.stringify({
      message: "Item retrieved successfully!",
      itemId: id
    }),
  };
};
// similarly, define updateItem and deleteItem functions
```

- Each function is an asynchronous handler that will process the HTTP request.
- The event object contains the details of the HTTP request, such as parameters, headers, and body.
- The function responds with a JSON object containing the result.

Page No.
Date

Step 6:- Deploy the REST API

Once your configuration and code are set up, deploy the service to AWS:

serverless deploy

Step 7:- Test the API

You can use tools like Postman or curl to test your REST API.

command:- curl -X POST https://<your-api-url>/dev/item -d '{"name":"Book"}'

curl https://<your-api-url>/dev/item/1

Step 8:- Monitor and Logs

It's important to monitor your Lambda functions for debugging and performance analysis

command:-

serverless logs -f createItem -t

That's it! You've set up a basic REST API using the serverless Framework and AWS Lambda.

Page No.
Date

Q.2) Case Study for Sonarqube.

- Create ur own profile in sonarqube for testing project quality.
- Use sonarcloud to analyze your GitHub code.
- Install sonarlint in ur Java intellij ide or eclipse ide and analyze your java code.
- Analyze python project with sonarqube.
- Analyze node js project with sonarqube.

→ Case Study: SonarQube for Code Quality Analysis

Introduction:- This case study explores the use of SonarQube for improving and maintaining code quality through continuous inspection. SonarQube helps developers identify issues like bugs, security vulnerabilities, and code smells. The case study covers creating a personal profile in SonarQube, using SonarCloud to analyze GitHub repositories, integrating sonarlint in a Java IDE, and analyzing Python and Node.js projects with SonarQube.

1) Creating Your Profile in SonarQube:

The first step in using SonarQube is creating a personal profile, which allows you to manage and monitor code quality across different projects.

Steps:

- Sign in to SonarQube and set up project preferences
- Connect your project to the profile for analysis.

Your profile will track code quality trends and improvements.

2) Using SonarCloud to Analyze GitHub Code

SonarCloud integrates with GitHub, offering continuous analysis for code hosted in repositories.

Steps:

- Link your GitHub repository to SonarCloud after signing up.
- SonarCloud automatically analyzes code, providing insights into bugs, security vulnerabilities, and maintainability.

For example, you can use SonarCloud to analyze a Node.js project and view the results directly on the dashboard.

Page No.
Date

3) Installing Sonarlint in Java IntelliJ or Eclipse IDE

Sonarlint is a plugin that offers real-time code quality checks while coding in an IDE.

Steps:

- In IntelliJ, install sonarlint from settings > Plugin
  In Eclipse, use Help > Eclipse Marketplace.
- Configure SonarLint to sync with SonarQube or SonarCloud.

As you write Java code, SonarLint will flag issues and suggest optimizations.

4) Analyzing a Python Project with SonarQube:

SonarQube supports Python projects. To analyze:

Steps:

- Set up the Python project with a sonar-project.properties file.
- Run the SonarScanner to analyze the code:
  sonar-scanner
- View the detailed report on SonarQube's dashboard.

5) Analyzing a Node.js Project with SonarQube

For Node.js analysis:

Steps:-

- Create a sonar-project.properties file in the project root.
- Use sonarscanner for analysis.
- Review the report on SonarQube, focusing on code security and optimization.

Conclusion:-

SonarQube provides comprehensive code analysis across different languages. With tools like SonarLint and SonarCloud, developers can ensure continuous improvement in code quality, covering Java, Python, Node.js and more.

Q. 3) At a large organization your centralized operations team may get many repetitive infrastructure requests. You can use Terraform to build a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing team to efficiently deploy services in compliance with your organizations practices. Terraform cloud can also integrate with ticketing systems like serviceNow to automatically generate new infrastructure requests.

→ This question highlights how Terraform can be leveraged to create a "self-serve" infrastructure model in a large organization, allowing product teams to manage their own infrastructure independently while adhering to the organization's standards.

1) Self-serve Infrastructure with Terraform:

- By using Terraform, you can build reusable modules that encapsulate the best practices and compliance standards of your organization. These modules can be made available to various product teams, empowering them to provision infrastructure themselves while following the pre-defined standards.

Page No.
Date

- This reduces the bottleneck that a centralized operations team may face, as the product team may face, as the product teams no longer need to wait for manual approval or setup. Instead, they can handle their own infrastructure needs within the boundaries set by the organization.

2) Terraform modules for standardization:-

- Terraform modules are reusable templates that simplify the process of deploying infrastructure by codifying the standards for managing resources (e.g. servers, databases, networking). Product teams can leverage these modules to deploy resources that comply with organizational policies, ensuring consistency and compliance across different environments.
- This helps maintain security, performance, and cost-efficiency, as teams are not manually configuring infrastructure from scratch.

3) Integration with Ticketing systems like ServiceNow:

- Terraform cloud or Enterprise can integrate with tools like ServiceNow, automating the process of generating new infrastructure requests.
- For instance, when a product team submits a ticket for infrastructure resources, Terraform Cloud can automatically handle provisioning

based on pre-approved templates, reducing the need for manual intervention from the operations team.

This model optimizes operational efficiency by delegating infrastructure management to product teams while still maintaining control over the infrastructure standards, and automating the request process through its integrations.