Name : Vedant Sanap
Class : D15A
Roll No : 48

56

A

04/03

DSSignment-2

Create a Rest API with serverless framework.

Creating REST API with serverless framework is an efficient way to deploy serverless application that can scale automatically without ma managing server.

Serverless framework : A powerful tool that deployment of services and serverless applica across various cloud providers such as AWS, AZURE and Google cloud.

Serverless application architecture : this design mode allows developers to build application without worrying about underlying infrastructure, enabling focu on code and business logic.

REST API : Representational State transfer is architecture style for designing network applications

Steps for creating REST API for serverless framework :

Install serverless framework :

You start by installing serverless framework CLI globally using node package manager. (npm). This allow you to manage serverless applications directly from your terminal.

2) Creating a Node.js serverless project:
A directory is created for your project where you will initialize a Serverless service (project). The service will have all your lambda functions, config and cloud resources. Using the command create you set up a template for your Node.js microservices that will eventually deploy to AWS lambda.

3) Project Structure:
The project creates essential files like handler.js (which contains code for lambda functions) and serverless.yml

4) Create a REST API Resource.
In the serverless.yml file you define functions that handles post requests of HTTP.

5) Deploy the service:
With the 'sls deploy' command serverless framework packages your applications, uploads necessary files to AWS and setup the infrastructure.

6) Testing the API: Once deployed you can test your API using tools like curl or Postman by making post requests to generated API.

7. Storing data in DynamoDB: To store submitted candidate data you integrate AWS DynamoDB as a DB.

1. AWS IAM Permissions:
You need to ensure that Serverless framework is given right permissions to interact with AWS resources like DynamoDB.

0. Monitoring and Maintenance:
After deployment serverless framework ~~is given~~ provides source information like deployed endpoints, API key, log streams.

2. Case Study for SonarQube.
Creating your own profile in sonarqube for testing project quality. Use Sonarqube to analyze your Github code. Install Sonarlint in your Java intellij ide and analyze java code. Analyze Python project with sonarqube.
→ Sonarqube is an open source platform used for continuous integration of code quality. It detects bugs, code smell and security vulnerabilities in project across various programming language.

Profile Creation in Sonarqube:
Quality profiles in sonarqube are essential config that define rules applied during code analysis. Each project has a quality profile for every

Supported language with default being Sonar profile comes built in for all languages. Cu profiles can be created by copying or exte existing ones. Copying creates an independent profile, while extending existing ones. Copying an independent profile, while extending inh rules from parent profile and reflects fu changes automatically. Sonarqube allows for comparison of two profiles to check for di in activated rules and users can track c via event log.

e) Using Sonarcloud to github code:
Sonarqube is cloud-based counterpart of Son that integrates directly with Github, Bitbu Azure and Gitlab repos. To get started with cloud product page and connect your Github a Next import repo into your Sonarcloud. organ where each Github repo becomes a Sonarcloud Define 'newcode' to focus on recent changes an choose btwn Automatic analysis or CI-b analysis. Automatic analysis happens direct in Sonarcloud, while CI based analysis integ with your build process and the analysis is completed results can be viewed in both sonar and github including security import issue.

IT Auto.

Sonarlint in Java IDE:
Sonarlint is an IDE that performs on-the-fly code analysis as you write code. It helps developers detect bugs, security vulnerabilities and code smells directly in the development environment such as IntelliJ Idea or Eclipse. This approach ensures immediate feedback on code quality, promoting clean and maintainable code from beginning.

Analyzing Python Projects with Sonarqube:
Sonarqube supports Python test coverage reporting but it requires third party tool like coverage.py to generate the coverage report. To enable coverage adjust your build process so that coverage tool runs before Sonar Scanner and ensures report file is saved in different path.
For setup you can use Tox, PyTest and coverage to configure & run tests. In your tox.ini include config for pytest and coverage report in XML format. The build process can also be automated using github action which installs dependencies, runs tests and invoke SonarQube scan.

Analyzing Node.js project with SonarQube:
For Nodejs project SonarQube can analyze Javascript and Typescript code. Similar to Python setup, you can configure SonarScanner to scan

the code against Industry Standard rule best practices, flagging issues related to sec vulnerabilities bugs & performance optimization

Q3) In a large organization, your centralized ops team may get repetative infra requests. You can use Terraform to build a self service infrastructure.

→ Terraforms self-serve infrastructure provides a powerful use case in larger organizations

i) Self serve infra: By using Terraform modules you can create reusable and standardized configs. Module creation in Terraform, main.tf, variable.tf and output.tf.
Also after module creation its standardization is equally important.

(ii) Enabling self-service for product teams:
Create a self service or version control access and stream products teams and the most important RBAC for preventing unauthorized Access.

(iii) Automate Infrastruck Request via ticketing System:
Integrate terraform cloud or Terraform Enterprise Connect terraform with the ticketing system automate approval workflows and monitor & log requests.

Workspaces Setup for environment Segregation :

To manage different environments, Terraform workspaces were setup. This ensured that teams could deploy the same infrastructure code in different environments without overlap.