Name: Vaishnal Mali
Div: DISA
Roll No.: 27
04/05
Adv- devops ~~Exp~~ Assignment-2

Q.1 Create a REST API with Serverless framework.

→ Creating REST API with Serverless framework is an efficient way to deploy serverless applications that can scale automatica-lly without managing servers.

(i) Serverless Framework: A powerful tool that simplifies the deployment of serverless applications across various cloud providers such as AWS, Azure and Google cloud.

(ii) Serverless Architecture: This design model Allows developers to build applications without worrying about underlying infrastructure, enabling focus on code and business logic.

(iii) REST API: Representational state transfer is orchitectural style for designing network applications.

Steps for Creating REST API for serverless framework:

1) 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) Create a Node.js Serverless project:
A directory is created for your project, where you initialize a Serverless service (project). This service will house all your lambda functions, configurations and cloud resources. Using the command 'serverless create', you set up a template for AWS Node.js microservices that will eventually deploy to AWS Lambda.

3) Project Structure:
The project scaffold creates essential files like handler.js

(which contains code for lambda function) and serverless.yml.

4) Create a REST API Resource
In the serverless.yml file you define function that handles HTTP POST requests.

5) Deploy the service:
With the 'sls deploy' command, serverless framework packages your applications, uploads necessary resources to Aws and set up the infrastructure.

6) Testing the API:
Once deployed you can test REST API using tools like curl or Postman by making POST requests to genrated API.

7) Storing data in Dynamo DB
To store submitted Candidate data, you integrate Aws Dynamo DB as a database.

8) Adding more functionalities:
Adding functialities like 'List all candidates, Get candidates by ID'.

9) Aws IAM Permissions
You need to ensure that serverless framework is given right permissions to interact with Aws resources like dyanamo DB.

10) Monitoring and Maintenance
After deployment serverless framework provides service information like deployed endpoints, API key, log streams.

Q.2 Case study for Sonarqube:

Creating ur own profile in Sonarqube for testing project quality.

Use Sonarcloud to analyze your Github code.

Install Sonarlint in ur Java intellj ide and analyze Java code

Analze python project with Sonarqube

Analze node js project with Sonarqube

→ SonarQube is an open-source platform used for continous inspection of code quality. It detects bugs, code smells and security vulnerabilities in project across various programming languages.

1) Profile Creation in SonarQube:

Quality profiles in SonarQube are essential configurations that define rules applied during code analysis. Each project has a quality profile for every supported language with default being 'Sonar way' profile comes built-in for all languages. Custom profiles can be created by copying or extending existing ones. Copying creates an independent profile, while extending inherits rules from parent profile and reflects future changes automatically. You can activate or deactivate rules, prioritize certain rules and configure parameters to tailor profile to specific projects. Permissions to manage quality profile are restricted to users with adminstrative privileges. SonarQube allows for the comparison of two profiles to check for diffrences in activated rules and users can track changes via event log. Quality profiles can also be imported from other instances via backup and restore. To ensure profiles include new rules it's important to check against updated built in profiles or use SonarQube rules page.

2) Using SonarCloud to Analyze GitHub Code:
Sonarcloud is cloud-based counterpart of SonarQube that integrates directly with Github, Bitbucket, Azure and Gitlab repositories. To get started with Sonarcloud via Github Sign up via Sonarcloud product page and connect your Github organization or personal account. Once connected, Sonar Cloud mirrors your Github setup with each project corresponding to Github repository. After setting up the organization choose subscription plan (free for public repo). Next, import repositories into your Sonarcloud organization where each Github repo becomes a sonarcloud project. Define 'new code' to focus on recent changes and choose between automatic analysis or CI-based analysis. Automatic Analysis happens directly in Sonarcloud, while CI-based analysis integrates with your build process. Once analysis is complete results can be viewed in both Sonar-cloud and Github including security import issue.

3) 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 enviorment such as IntelliJ Idea or Eclipse. To set it up, install the SonarLint plugin, configure the connection with SonarQube or Sonarcloud and select the project profile to analyze Java code. This approach ensures immediate feedback on code quality, promoting clean and maintainable code from beginning.

4) Analyzing Python projects with SonarQube:
SonarQube supports python test coverage reporting but it requires third party tool like Coverage.py to genrate the Covarage port. To enable Covarage adjust your blind process so that coverage tool runs before Sonar scanner and ensures report file is saved in defined path.
For setup you can use Tox, PYtest and Coverage.py to configure and run tests. In your tox.ini include configurations for pytest and coverage to genrate coverage report in XML format. The build process can be automated using Github Actions, which install dependencies runs, tests and invoke SonarQube scan. Ensure report in Cobeurata XML format and place where scanner can access it for proper analysis.

5) Analyzing Node.js Projects with SonarQube:
for Node.js project SonarQube can analyze Javascript and Typescript code. Similar to the Python setup, you can configure SonarQube to analyze node.js projects by installing the appropriate plugin and using Sonar Scanner to scan the project. SonarQube will check the code against industry standered rules and best practices, flagging issues related to security vulnerabilities bugs and performence optimization.

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 services in your organization. Terraform cloud can also integrate with ticketing System like servicenow to automatically generate new infrastructure requests.

→ Implementing a self-serve infrastructure model using Terraform can transform how large organization manage their infrastructure independently, organization can enhance efficiently reduce bottlenecks and ensure compilance with established needs.

(i) The need for self-service infrastructure:

In large organizations, centralized operations teams often face an overwhelming number of repetative requests. This can lead to dealys in service delivery and frustration among product teams to provision and manage their infrastructure without relying on the operation team for every request.

Benefits of using Terraform

1) Modularity and Reusability:

Terraform modules encapsulate standard configurations for various infrastructure components. (eg network, database) Teams can reuse these modules across diffrent projects, reducing reduconces and minimizing risk of errors.

2) Standerdizations

By defining best practices within modules organizations can ensure that all deployments comply with internal

policies and standards. This consistency helps maintain security and operational integrity across organization.

3) Increased Efficiency

Product teams can integrate with ticketing systems like servicenow to automate genration of infrastructure requests. This integration streamlines workflows by allowing teams to initiate requests directly from their ticketing platforms reducing manual invention.

Implementation of steps:

1) Identify Infrastructure Components

2) Develop Terraform modules

3) Establish foravances and Best Practices

4) Testing and validation

Best practices for Module managment:

1) Utilize teraform registry
2) Version control
3) Documentation
4) Encourage collabration

This approch not only streamlines process but also enhance agility in reponding to changing business needs. Ultimately it leads to more responsive IT enviorment that supports innovation and growth within organization.