Manav Punjabi
D15A - 45 Batch - C

AdvDevOps Assignment - 2

1.] Create a REST API with Serverless framework

Ans Creating REST API with Serverless framework is an efficient way to deploy Serverless applications that can scale automatically without managing servers.

i) Serverless framework: A powerful tool that automates deployment of services and Serverless applications across various cloud providers such as AWS, Azure and Google cloud.

ii) Serverless architecture: This design model allows developers to build application without worrying about underlying infrastructure, enabling focus on code and business logic.

iii) REST API:- Representational state transfer is architecture style for designing network applications.

Steps for creating REST API for Serviced framework:

1) Install Serverless framework:
You start by installing Serverless framework globally using node package managers. This allows you to manage Serverless applications directly from your terminal.

2) Creating a Node.js Serverless Project:
A directory is created by your Project where you will initialize a Serverless service. This service will house all your lambda function, configuration, and cloud resources.

3) Project Structure:
The Project staffing 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 function that handles Post request of HTTP

5) Deploy the service:
with the 'sls deploy' command serverless framework packages your application, uploads necessary resources

6) Testing the API: once deployed you can test Rest API using tools like curl or Postman.

7) Storing data in Dynamic: To store the submitted candidate data you integrated AWS.

8) Adding more functionalities: Adding functionalities like 'list all candidate, get candidate by ID'

9) AWS IAM permissions
you need to ensure that serverless framework is given right permissions to interact with AWS resources like Dynamic.

10) Monitoring and maintaince
After deployment serverless framework is given right permissions to interact with AWS resources like Dynamic.

8.2) Case study for sonarable
creating your own Profile in sonarable for testing Project quality. we sonarable to analyse your Github code. Install sonarlin in your Java intellij ide and analyze java code. Analyze Python Project with sonarable.

→ sonarable is an open source platform used for continuous inspection of quality. It detects bugs, code smells and security vulnerability in project across various programming languages.

1) Profile creation in sonarable.
Quality Profiles in sonarable are essential configuration 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 inroast races fr
Parent Profile and rejects future change
automatically you can activate or deact
rules, Prioritize certain rules and
configure Parameters to tailor Profile t
specific Profiles. Permissions to manage
quality Profile are restricted to users
with administrative Privileges. Sonar
allows for the comparision of two
Profiles to check for differences in
activated rules and users can track
changes via event tag.

2) using sonarcloud to analyze Github
Sonarcloud is cloud-based code Post c
Sonarqube that integrates directly with
Github - BitBucket, Azure and Github
repositories. To get started with
sonarcloud via Github signup via son
cloud Product page and Connect
your Github organization or Perso
Account. once Connect, sonarcloud
mirror 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

including security import issue.

3) Sonarlint in Java IDE:
Sonarlint is an IDE that performs on-the-fly code analysis as your write code. It helps developers detect bugs, security vulnerabilities and code smells directly in the development environment such as IntelliJ IDEA or Eclipse. To set it up, install the Sonarlint plugin, configure the connection with Sonarcloud or Sonarqube and select the project profile to analyze Java code. This approach offers immediate feedback in code quality, promoting clean and maintainable code from beginning.

4) Analyzing Python Projects with Sonarcloud:
Sonarcloud support 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 that ensures report file is saved in different path.
For setup you can use Tox, PyTest and coverage and run test. In your tox.ini include configurations for Pytest and coverage to generate coverage report in xml format. The build process can also be automated using Github Actions, which install dependencies, runs tests and invokes Sonarcloud scan.

5) Analyzing Node.JS Projects with Sonarlable. 3.

For Node.JS Project Sonarsoube can analyze Javascript and Typescript code. Similar the Python setup, you can configure Sonarlouble analyze Node.JS Projects by installing the appropriate Pulgins and using Sonar Scanner to Scan the Projects. Sonarsoube will check the code against Industry standard rules and best Parties.

Q.3] In large organizations, centralized operation teams often face many repetitive infrastructure requests, causing delays. Using a self service infrastructure model with Terraform can help Product means teams manage the infrastructure model with Terraform can help Product team manage independently.

key benefits of terefom:-

1. Modularity & Reusability: Terraform modules encapsulate standard configurations (database, compute resources) for reuse across Projects.

2. Standardization: Best Practices are built into modules, enforcing compliance with organizational policies.

3. Version control:- versioning helps manage dependencies and reduce disruptions.

4. Documentation: Comprehensive documentation promotes collaboration and understanding.

Implementation steps:-

1. Identify Infrastructure components: Determine which components (security grps, load balancers) to modularize.

2. Develop Reusable module:- Define configuration with input variables and output for integration.

3. Governance and Best Practices:- Set guidelines for module usage, versioning, and encourage team contributions.

4. Testing and validation: Implement testing frameworks to validate module functionality.

Terraform can also integrate with ticketing system like ServiceNow to automate infrastructure requests, improving workflows and reducing manual interactions. This self-service model increases agility.