Name :- Soham Satpute
Roll no :- 52/D15A.

AdDevops Assignment No: 2

Create RES API with the serverless Framework:

Here are the several steps to be followed to create REST API using serverless framework:

Set Up Environment:
• Ensure you have an AWS account and install node.js, npm and the serverless Framework globally

Initialize Project:
• Create a new project directory and initialize new serverless service using a template.

Define Configuration:
• ~~Eid~~ Edit the serverless.ymal file to specify the service name, provider details, runtime.

4] Create Lambda Functions:
• Implement the logic for each API endpoint in a separate file.

5] Configure Data store:
• Choose a ~~ba~~ database and set permissions [illegible] serverless.yml file if needed.

6] Deploy and test API:
- use the serverless framework to deploy your API to Aw.
- use tools like Postman or curl to test th API endpoints.

~~7] Test API:~~
~~• 88~~

7] Monitor and Updates:
- Monitor performance via Aws Cloudwatch update the API as needed.

Q2] Case Study for SonarQube:

→ ~~1A~~ Case Study: Quality Analysis of Sonar Qube:-

objectives:

1] Create a Sonarqube profile:
- Log in to sonarqube and navigate quality profiles.

~~2]~~• Click create, name your profile, and customize rules for your desired quality standards.

2] Analyze github code with sonarcloud:
- Sign in to SonarCloud with github and link your repository.

Set up a sonar-project.properties file in your repo, trigger analysis through CI/CD.

Install SonarLint For Java:
- Install solarlint in IntelliJ from the plugins
- Bind your project with sonarqube and sonarlint will flag issues with your java code and analyse it.

Analyze Python Project:
- Create a new project in sonarqube for python code.
- Add a sonar-project.properties file and run analysis using sonarscanner and check results.

Analyze Node.js Project:
- Create a new project in SonarQube for your Node.js code.
- ~~create~~ configure with a sonar-project.properties file and execute and analyse using Sonarscanner.

Conclusion:
This case study demonstrates how to use SonarQube effectively for code quality analysis across different programming languages and platforms.

Q3]

→ Implementing a self-serve infrastructur model using terraform can significantly streamli operations with large organisation. Here's a structur approach to achieve this:

1] Centralized Operations:
- Address repetitive infrastructure requests from product teams by using terraform.

2] Create Terraform Modules:
- Develop reusable modules that define stand for deploying and managing infrastructure

3] Empower product Teams:
- Set up a self-service portal where teams can initiate requests using predefined modules, enabling them to manage their infrastructure independently.

4] Integrate terraform Cloud:
- Connect terraform Cloud with ~~#~~ ticketing systems like ServiceNow to automate infrastruc requests. When a team submits a request, terraform can provision the needed resources automatically.

In tegrate with Ticketing system
Centralize state ma
Automate infrastructure requests through ticket system.
Streamline request and approval processes.

This approach helps centralize control while empowering product teams, creating a balance between agility and governance.