Niraj S. Kothawade
D15A - 24

05/05

Adv. DevOps Assignment 2

Q1. Create a REST API with Serverless framework.

→

Creating REST API with serverless framework is an effecient way to deploy serverless applications that can scale automatically without managing servers.

1. Serverless framework : A powerfull tool that simplifies the deployment of serverless applications across various cloud providers such as AWS, Azure and Google cloud.
2. Serverless Architecture : This design model allows developers to build applications without worrying about underlying infrastructure, enabling focus on code and business logic.
3. REST API : Representational state transfer is architectural 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. This allow you to manage serverless applications directly from your terminal.

2) Create a Node.js serverless project, where you initialize a serverless servic. 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 'serverless.yml' file you define function that handles HTTP POST requests.

5) Deploy the service:
With the S/s 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 Postman by making POST requests to generated API.

7) Storing data in DyanmoDB:
To store submitted candidate data, you integrate AWS Dynamo DB as a database.

8) Adding more functionalities:
Adding functionalities 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.

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

Q2. Case study for Sonarqube.

Creating your own profile in Sonarqube for testing project quality. Use sonarcloud to analyze your Github code. Install Sonarlint in your Java IntelliJ and analyze Java code. Analyze python project with Sonarqube. Analyze node js project with sonarqube.

→

1. Profile creation in SonarQube:
Quality profiles in sonarQube are essential configuration that define rules applied during code analysis. Each project has a quality profile for every supported language with default being Sonarqube 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 indepent 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 also be imported from other instances via backup and restore. To ensure profiles include new rules its 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 signup via SonarCloud product page and connect your Github organization or personal account. Once connected SonarCloud mirrors your Github setup with each project corresponding to gitHub repository. After setting up the organisation choose subscription plan (free for public repo. b) 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 impart 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 environment 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.

Q3. At a large organisation, your centralized operations team may get many repetitive requests, you can use terraform to build a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. You can use terraform to build a modules that codify the standards for deploying and managing services in your organization allowing teams teams to efficiently deploy services in compilance with your organizations practices. Terraform Cloud can also integrate with ticketing system like service now to automatically generate new infrastructure requests.

→

In a large organization, centralized operations teams handle infrastructure provisioning for different departments or teams. However, as the demand for infrastructure increases, manually fulfilling these requests becomes inefficient. Terraform helps solve this problem by enabling a "self-serve" infrastructure model where teams can manage their own infrastructure

The Self-Serve Infrastructure Model:

1. Standardization with Terraform Modules:
   - Create Terraform modules that encapsulate best practices for provisioning infrastructure like EC2 instances, databases, VPC's or networking configurations.
   - These modules can be shared with different teams, ensuring that they follow a common set of standards such as security policies, tagging and resources

management.

2. Self Serve infrastructure:
- Product teams can use these modules to create their infrastructure independently. The operations team only needs to define the modules and product teams can deploy the resources as needed.
- This allows for greater agility, as teams do not have to wait for manual approval or intervention from central operations team.

3. Use of Terraform Cloud:
- Terraform Cloud allows for state management, policy enforcement and collaboration across teams.
- Using terraform cloud, teams can automate the deployment process and perform infrastructure updates.

4. Integration with Ticketing Systems.
- Terraform Cloud can be integrated with systems like ServiceNow, automating the process of generating new infrastructure requests.

Benefits of Self Serve Infrastructure model:
1. Agility
2. Standardization
3. Resource efficiency
4. Automation.