Prajjwal Pandey
DISA - 33

33

04/05

Advance DevOps Assignment 2

1. Create a REST API with the serverless framework.

S: The serverless framework simplifies the deploymen-t and management of serverless applications. It allows developers to build and deploy application ~~from~~ on cloud platform like AWS without having to manage the underlying infrastructure.

Step1: Prerequisites
1. Node Js and npm installed on your local machine.
2. Serverless framework installed globally using npm.
command: npm install -g services

Step 2: Install the serverless framework first ensure you have Node JS and npm installed. Then ensure to install the serverless framework globally.

Step 3: Create a new serverless service serverless create --template aws-nodejs --path my-service
cd my-service.

Step 4: Define function in serverless.yml open serverless.yml file in project directory. In this file we define our service configurations including the functions and their triggers/events. This file contains service, provider, its name and runtime environment. It also contains functions which contains, create, read, update and delet methods.

Step 5: Write lambda functions (Handlers)
Open the handler.js file, and write the logic for the API endpoints.
handler.js contains the logic for :
- Handling a simple GET request
- Handling POST request to create a new item

Step 6: Deploy the service
To deploy the service and lambda functio
command: serverless deploy
with the 'sls deploy' command, serverless framework packages your applications, uploads necessary resources to AWS and set up the infrastructure.

Step 7: Testing the API
Once deployed you can test REST API using tools like curl or postman by making POST requests to generated API

Step 8: Storing data in Dynamo DB.
To store submitted candidate data, you integrate AWS Dynamo DB as a database.

Step 8 : AWS IAM Permissions
you need to ensure that serverless framework is given right permissions to interact with AWS resources like dynamo DB.

This is the whole process which will create a fully serverless REST API using AWS lambda, API Gateway the serverless framework.

Q2. Case study for sonar qube.
Creating your own profile in sonarqube for testing project quality. Use sonarqube to analyze your github ~~pr~~ code. Install sonarlint in your java intellij IDE and analyze java code. Analyze python project with sonarqube.

A: Sonarqube is an opensource platform used for continuous 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 defines 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 inherited rules from parent profile and reflects future changes automatically. You can activate or deactivate rules, prioritize certain rules and configure parameter to tailor profile to specific projects. Permissions to manage quality profiles are restricted to users with admin priveledges. SonarQube allows for the comparison of two profiles to check for differences 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, important to check against updated built in profiles or use sonarQube rules page.

(d) Using SonarCloud to analyze GitHub Code:
SonarCloud is Cloud-based counterpart of SonarQ that integrates directly with GitHub, BitBucket, Azure and Gitlab repositories. To get started with SonarCloud via GitHub signup via sonar Cloud pr page and ~~content~~ connect your GitHub organization personal account.
Once connected, sonarcloud mirrors your github setup with each project corresponding to Git repository. After setting up the organization ch subscription plan (free for public repo). Next, import repositories into your sonarcloud organ -ion where each GitHub repo becomes a SonarClo project. Define 'new code' to focus on recent chang and choose between automatic analysis or CI-ba analysis. Automatic analysis happens directly in sonarcloud, while CI based analysis interacts w your build process once the analysis is complete results can be viewed ~~by~~ in both SonarCloud 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 environmen-t such as IntelliJ IDE 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 maintable code from beginning.

④ Analyzing python projects with SonarQube:
SonarQube supports python test coverage, reporting but it requires third party too like coverage.py to generate the coverage porti. To enable coverage adjust your ~~build~~ blind 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.py to configure 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 installs dependencies, runs, tests and invoke SonarQube scan. Ensure report in cobeurata xml format & place where scanner can access it.

(5) Applying node JS project with SonarQube. :
for NodeJS project sonarQube can analyze Javascript and Typescript code. Similar to the p setup, you can configure sonarqube to analyze nodejs projects by installing the appropriate plug and using sonarscanner to scan the projects. SonarQube will check the code against industry standard rules and best practices, flagging issu related to security vulnerabilities bugs ~~are~~ and performance optimization.

Q3. At a large organization your centralized operation 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 lik services to automatically generate new infrastructur requests.

Sol: Implementing a self-serve infrastructure model usi Terraform can transform now large organization mange their infrastructure independently, organizati can enhance efficiency, reduce bottlenecks and ensu compliance with established needs.

(1) The need for self-service infrastructure:
In large organizations, centralized operations tear often face an overwhelming number of repetative requests. This can lead to ~~deploy~~ delays in service

Sundaram

FOR EDUCATIONAL USE

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 encapsulates standard configurations for various infrastructure components. (e.g. network, database)
Teams can reuse these modules across different projects, reducing reducances and minimizing risks of errors.
2. Standardizations
By defining best practices within modules organizations can ensure that ~~a~~ deployments comply with internal policies and standards. The consistency helps maintain security and operational integrity across organization.
3. Increases efficiency
Product teams can integrate with ticketing systems like service now to automate generation 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 best practices.
4) Testing and validation.

Best practice for module management:
1) Utilize terraform registry
2) Version control
3) Documentation
4) Encourage collaboration.

This approach not only streamlines process but al
enhance agility in responding to changing
business needs. Ultimately it leads to more
responsive IT environment that supports innovati
and growth within organization.