Aryan Patankar
DISA
34

Advance Devops Assignment 2

Q.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 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 applications 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 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) Creating a Node.js serverless project:
A directory is created for your project, where you will initialize a serverless service (project). This service will house all your lamba 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 functions) and serverless.yml.

4) Create a REST API Resource:

In the serverless.yml file you define function that handles post requests of HTTP

5) Deploy the service:
With the 'sls deploy' commands 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 generated API.

7) Storing data in DynamoDB: To store submitted candidate data you integrate AWS DynamoDB as a database

8) ~~Add~~ 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 like DynamoDB

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 your own profile in sonarqube for testing project quality. Use sonarqube to analyze your github code. Install sonarlint in your Java intelly Ide and analyze java code. Analyze python project with sonarqube.

→ SonarQube is an open source platform used for continuous inspection of co

quality. It detects bugs, code smells and security vulnerabilities in project across various programming languages.

1) Profile creation in SonarQube:
Quality profile 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 inherit 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 profile are restricted to users with administrative privileges. 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 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 connect, SonarCloud mirrors your GitHub setup with each project corresponding to GitHub repository. After setting up the organization choose subscription plan (free for public repo). Next, impo 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 the analysis is complete result 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 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. 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 generate the coverage port. To enable coverage adjust your 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 install dependencies, runs, tests and invoke SonarQube scan. Ensure report in. coberwrata XML format and path where scanner can access it.

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 plugins and using SonarScanner to scan the projects. SonarQube will check the code against Industry standard rules and best practices, flagging issues related to security vulnerabilities bugs and performance optimization.

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 their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organization's practices. Terraform Cloud can also integrate with ticketing system like ServiceNow to automatically generate new infrastructure requests.

Ans. Implementing a "self-serve" infrastructure model using Terraform can transform how large organizations manage their infrastructure independently, organizations can enhance efficiency, reduce bottlenecks, and ensure compliance with established needs.

- The Need for self-service infrastructure:
In large organizations, centralized operations teams often face an overwhelming number of repetitive requests. This can lead to delays in service delivery and frustration among product teams who need to move quickly. A self-service model allows teams to provision and manage their infrastructure without relying on the operations team for every request

- Benefits of using Terraform

1. Modularity and Reusability:
- Terraform modules encapsulate standard configurations for various infrastructure components (eg. networks, databases, compute resources).
- Teams can reuse these modules across different projects, reducing redundancy and minimizing the risk of errors.

2. Standardizations
- 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 the organization

3. Increased Efficiency
- Product teams can deploy services quickly by using pre-defined modules, significantly reducing the time spent on infrastructure setup.
- This allows team to focus on developing features rather than managing infrastructure

4. Integration with Ticketing Systems
- Terraform Cloud can integrate with ticketing systems like ServiceNow to automate the generation of infrastructure requests.
- This integration streamlines workflows by allowing teams to initiate requests directly from their ticketing platform, reducing manual intervention

→ Implementation steps

1. Identify Infrastructure components
- Begin by identifying which components of your infrastructure can be modularized (eg. VPCs, security groups, load balancers)

2. Develop Terraform modules.
- Create reusable modules that define the desired configurations and

resources.
- Ensure each module includes input variables for customization and outputs for integration with other modules.

3. Establish Governance and Best Practices:
- Define guidelines for module usage, versioning, and documentation to ensure clarity and maintainability.
- Encourage teams to contribute to module development and share improvements.

4. Testing and validation
- Implement a testing framework to validate module functionality before deployment.
- Use tools like terraform plan to preview changes and catch potential issues early.

- Best Practices for module management
- Utilize the terraform registry:
- Leverage existing community modules from the Terraform Registry to avoid reinventing solutions and ensure adherence to best practices.
- Version control: Implement versioning for your modules to track changes over time. This helps manage dependencies effectively and minimize disruptions during updates.
- Documentation: Maintain comprehensive documentation for each module including usage examples, input/output descriptions and any dependencies.
- Encourage collaboration: Foster a culture of collaboration by sharing modules across teams. This promotes consistency in deployments and facilitates knowledge within the organization.

By adopting a self-service infrastructure model with Terraform, organizations can empower product teams to efficiently manage their own infrastructure while ensuring compliance with established standards

This approach not only streamlines processes but also enhances agility in responding to changing business needs. Ultimately, it leads to a more responsive IT environment that supports innovation and growth within the organization.