Ryjuta Medhi
D15A - 28

A+ 05/05

Assignment - 2.

Q1 Create a REST API with serverless framework

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

i) Serverless framework: A powerful tool that simplifies deployment of 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 caring about underlying infrastructure, enabling focus on code and business logic.

iii) Rest API: Representational state transfer is architectural style for designing network applications.

Steps for creating REST API for server-less framework:-

1] Install serverless framework:
You can start by installing serverless framework CLI globally using npm.

2] Create a Node.js serverless project:
A directory is created for your project, where you initialise Serverless service. This service will have all lambda functions, configurations and cloud resources.

3] Project structure :
The project scaffold creates essential files like handler.js

4] Create Rest API resource :
In the serverless.yml file you define function. to handle HTTP.

5] Deploy the service :
deploy using 'S/S deploy' command

6] Testing the API :
Once deployed, Test APS using curl or Postman.

7] Add more functionalities :
Add functionalities like 'List all candidates'.

8] Monitoring and maintenance
After deployment serverless framework provide service information like deployed endpoints.

Q 2. Case ~~Study~~ for Sonarqube
• Create your own profile in sonarqube for testing project quality.
• Use sonar cloud to analyze your GitHub code
• Install sonarlint in your Java Intellij IDE. or
• Analyze python project
• Analyze node js project with sonarqube

→ • Create the sonarqube profile for testing project quality.
• Open intelly setting, find Tools > Sonarlint entry and select + to open connection wizard
• Enter name for this connection, select sonarcloud or sonarqube
• choose authentication method :
a) Generatic token on sonarqube or sonarcloud
b) Username + Password. This can be used on sonarqube connection only
• For sonarcloud only select organization that you want to connect
• Sonarqube and sonarcloud can push notifications to developers.
• Validate the connection creating by selecting finishing at the end of the wizard.
• save the connection in global setting by clicking OK.

Bind Python Project to Sonarqube
• Select Sonarlint > Bind Project to Sonarqube.
• choose the correct project from sonarqube
Analyze the project (Python Project)
• Trigger an analysis by going to code > Analyze code

Analyze Node.js project
Make sure your Node.js project is properly configured with sonar-qube properties file or equivalent for the analysis to run.

Q3 At large organization your centralized operations team may get many repetitive infrastructure requests. You can use Terraform modules to build "self-serve" infra-structure model that lets product teams manage their own infrastructure independently. Terraform cloud can also integrate with ticketing system like service now to automatically generate new infrastructure requests.

Self-serve infrastructure model with Terraform modules:
At a large organization, implementation a self-serve infrastructure model using Terraform can significantly streamline process of managing infrastructure across differend teams. This approach allows users to manage their own infrastructure independently while adhering to organizational standards.
Key aspects are :-

a) Standardization through Terraform Modules by creating and utilizing Terraform modules, organizations can codify their infrastructure deployment and management standards.

b) Efficient Deployment: Product teams on leverage these standardized modules to quickly deploy services without needing to reinvent the wheel or wait for the centralized operation team to handle every request.

c) Compilance: By using predefined modules, teams, ensure that their deployment comply with organisations

established practices and security guidelines
d] Automation :
The use of Terraform modules promotes automation, reducing manual intervention and potential human errors in infrastructure management.
e] Version control :
With modules stored in version control system like Git, teams can track changes, collaborate on improvements and maintain a history of infrastructure configuration.

Integration with Ticketing Systems:
Terraform cloud offer integration capabilities that further enhance the self-serve model:
a) Automatic infrastructure requests :
Terraform cloud can integrate with ticketing systems like servicenow to automatically generate new infrastructure requests.
b) Centralised Management :
By centralizing infrastructure management through Terraform cloud organisation can maintain better control over who can request and approve infrastructure changes.
c) Governance ::
The integration with ticketing systems allow for better governance of infrastructure requests.

Collaborative Infrastructure Management

a) Team Based performance permissions
b) State and run history.
c) Sensitive Information protection.
d) Module Registry.

By implementing these features and promises large organisations can effectively leverage Terraform to build a robust, scalable and complaint infrastructure management system.