Bhushan malpani
D15C
33

Page No. 03
Date 03/9

Assignment -2

Q) Create a REST API with serverless framework.

⇒ • Set up the environment
  • Install Node.js, AWS CLI and serverless framework.
  • Configure AWS credentials using aws configure.
• Create a new Serverless service.
  • Create a new directory for the project
  • Use the serverless CLI to generate a new service with serverless create --template aws-nodejs.

• Configure the serverless.yml file
  ∘ Define the service, provider [AWS] and functions [API endpoints] in serverless.yml.
  • Set up the HTTP methods [GET, POST, PUT, DELETE] for your REST API paths.

∘ With the API logic:
  • In handler.js implement functions for handling requests [e.g fetching, creating, updating & deleting data].
  • Handle different HTTP methods & their corresponding responses.

∘ Deploy the service:
  • Deploy the service to AWS using serverless deploy.
  • Take note of the API endpoints provided in the deployment output.

• TEST the API:
  • Use tools like POSTMAN to test the endpoint [e.g sending GET, POST, PUT and DELETE requests]

• Remove the service:
  • After testing, remove the service from AWS using serverless remove to clean up resource.

These steps will guide you through the process of creating, deploying & managing a serverless REST API

Q2> 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 sonarqube Java Intellij ide and analyze java code. Analyze python project with sonarqube.

⇒ Sonarqube is an open source platform used for continuous inspection of code quality. It detects bugs, code smalls & security vulnerabilities in project across various programming language.

i> 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 'sonarway' profile comes built in for all language. Custom profiles can be created by copying or extending existing ones. Copying creates an independent profile, while extending existing ones. Copying creates an independent profile, while extending inherit rules from parent profile & 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 & restore. To ensure profiles include new rules its important to check against updated built in profiles or use sonarqube rules page

Using sonarcloud to analyze Github code:
Sonarcloud is cloud-based counterport of sonarqube that integrates directly with Github, BitBucket, Azure and Gitlab repositories. To get started with sonarqube via Github signup via sonarcloud product page & connect your Github organization or personal product page and connect your Github repository. After setting up the organization choose subscription plan [free for public repo). Next, import respositories into your sonarcloud organization where each Github repo becomes a sonarCloud project. Define 'new code' to focus on recent changes & choosen betn 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 results can be viewed in both sonar-cloud and Github including security import issue.

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 & code smells directly

in the developement environment such as Intellij Idea or Eclipse. To set it up, install the sonarlint plugin, configure the connection with sonarqube or Sonarcloud & select the project profile to analyze Java Code. This approach ensures immediates feedback on code quality, promoting clean & maintainable code from beginning.

4) Analyzing python projects with sonarqube:
Sonarqube supports python test coverage, reporting but it requires third party to like coverage.py to generate the coverage tool runs before sonar scanner & ensures report file is saved in different path.

5) Analyzing Node.js projects with sonarqube
For node.js project sonarqube can analyze javascript & typescript code. Similar to the python setup, you can configure sonarqube to analyze node.js projects by installing the appropriate plugins & using sonarscanner to scan the projects. Sonarqube will check the code against Industry standard rules, best practices, flagging issues related to security vulnerabilities bugs & 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 independently. You can create & use terraform modules that codify the standards for deploying & managing services in compliance with your organisation's practices. Terraform cloud can also integrate with ticketing system like ~~environment~~ service now to automatically generate new infrastructure requests.

Implementing a 'self-service' infrastructure model using Terraform can transform how large organizations manage their infrastructure independently, organization can enhance efficiently, reduce bottlenecks & ensure compliance with established needs.

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

• Benifits of using Terraform:
1> Modularity & Reusability:
• Terraform modules encapsulate standard configurations for various components
• Teams can reuse these modules across different projects, reducing redundancy & minimizing the risk of errors.

2> Standardizations:
• By defining best practices with modules, organisation can ensure that all deployments comply with iternal policies & standard.
• This consistency helps maintain security & operational integrity across the organization.

3> Increased Efficiency:
• Product teams can deploy services quickly by ~~si~~ using pre-defined modules, significantly reducing the time spent on infrastructure setup.

Page No.
Date

4> Integration with Ticketing systems:
• Terraform cloud can integrate with ~~creati~~ ticketing system like service now 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 with components of your infras-tructure can be modularized.
2> Develop Terraform moduls:
• Create reusable modules that define the desired configurations & resources.
3> Establish Governance & Best Practices:
• Define guidelines for module usuage, versioning & documentation to ensure clarity & maintainbility.
4> Testing & validation:
• Implement a testing framework to validate module Functionality before deployment.

* Best practices for module management:
• Utilize the terraform register.
• Version control: Implement versioning for your modules to track changes overtime. This helps manage dependencies effectively & minimize distruption during updates.

This approach not many streamline processes but also enhances agility in responding to changing business needs. Ultimately, it leads to a move responsive IT environment that supports innovation & growth within

10.1

the organization.

→ Terraform "self-serve" infrastructure model
→ i) Terraform modules for self-serve infrastructure
• create terraform modules that codify the standard for deploying common resources like VPC's, EC2 instan & S3 bucket.

Ex:
```
variable "instance-type" {
        default = "t2.micro"
  }
 resource "aws_instance" "example" {
   ami = "ami-12345678"
  instance_type = var.instance_type
  tags={
        Name: "example-instance"
     }
 }
 ec2-module/outputs.tf:
 output "instance_id" {
      value = aws_instance.example.id
 }
```

Teams can now use the module to deploy with

```
module "ec2" {
     source = "./ec2-module"
     instance_type = "t2.medium"
  }
```

Terraform can now use thes module to deploy EC2 instances with

• You can integrate terraform cloud with service Now to automate the infrastructure request process.

• Using terraform's API driven approach, service Now can trigger Terraform runs based on ticket approvals, automating resource deployment.

Example workFlow:

1) A product team submits a request in service Now for new infrastructure

2) The request triggers a Terraform Cloud updates the service Now ticket with the status, & resource details.

3) Creating Terraform modules for ~~stat~~ teams define reusable modules for commonly requested resource like

1) Networking [VPC, Subnets]

2) Compute [EC2, Autoscaling Groups

3) Storage [S3, RPS]

4) IAM Roles/ Policies

By doing this, teams can manage their own infrastructure while maintaining compliance with organizational standard.

[illegible]