MUSKAN CHANDIRAMANI
ROLL NO - 5
DIS C

# ASSIGMENT : 2

03/05

Q1 Create a REST API with serverless framwork.

→ It is an efficient way to deploy serverless application that can scale automatically without managing server

Steps For creating :

1) Install serverless framework
you start by installing CLI globally using node packages manager. This allow you to manage serverless application directly from your terminal

2) Creating a Node.js serverless project
A directory is created for your project, where you will intialize a severless service, this service will house all your lamba function, config, & cloud services.

3) Project structure
The project creates essential files like handler.js & serverless.yml.

4) Create a REST API resource
In the serverless.yml fille you define function that handles post requests of HTTP

5) Deploy the service
with the 'sls deploy' commands serverless frame work packages your applications

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
you need to integrate AWS DynamoDB as Database.

8) Adding more functionalities
like 'list all candidates, get candidates by ID'

9) AWS IAM Permissions
you need to ensure that serverless framwork is given right permission to interact with AWS resources like DynamoDB.

10) Monitoring & Maintanance
After deployment serverless framework provides service information like deployed endpoints, API keys, log streams.

Q2 Case study for sonarqube.

→ SonarQube is an open source platform used for continuous insertion. It detects bugs, code smells & security vulnerabilities in project across various prog languages.

1) Profile creation in SonarQube.
Quality profile are essential configuration that define rules applied during code analysis. Each project has a quality profile for every supported language. Custom profile can be created by copying or extenting existing ones Copying creates an independent profile, while extending inherit rules from parent profile.

Permissions to manage quality profile are restricted to users with administrative privelage. It allows for comparison of two profile to check difference

2) Using Sonarcloud to analyze GitHub code.
It is a cloud based counterport of sonarQube that integrate directly with Github; Bitbucket, Azure & Gitlab repositories. To get started with sonarcloud via Github signup via product page & connect your Github organization to personal account. Once connect, setup will be done with each project corresponding to repository, while CI based analysis integrate with your build process once the analysis is complete results can be viewed in both sonarcloud & Github

3) Sonarlint in Java IDE.
Sonarlint is an IDE that performs on-the-fly code analysis as you write code. It helps developer detect bogs, security & code smells directly in development environment such as intell, Idea or Eclipse. To set it up, install the sonarlint plugin, configure the connection with sonarqube as sonarcube & select the project profile to analyze Java code This approche ensures immediate feeback on code quality. & maintain 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 port
To enable coverage adjust your blind process so that tool runs before & ensure, file is saved in different tools path.
For setup, you can use FOX, PyTest to configure & run test. In you tox.ini include config for report in XML format. The build process can also be automated using Github actions.

5) Analyzing Node.js projects with SonarQube
For Node.js project sonarqube can analyze Javascript and TS code. similar to python setup, you can configure sonarqube to analyze node.Js project be installing the plugins & using sonarscanner to scan the projects. It will check the code.

Q3 Terraform "self-serve" Infrastructure Model.
→ ① Create Terraform modules that codify the standards for deploying common resources like VPCs, EC2, S3 buckets
Ex for an EC2 instance:
ec2-module/main.tf:
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
  }
```

② Terraform cloud Integration with Service now

- you can integrate Terraform cloud with service now to automata the infrastructure request process
- Using approach it runs based on ticket approval, automating resource deployment.

Example workflow: 1) A product team submit a request in service Now

② The request triggers & updates it with status & resource details

③ Creating Terraform Modules for teams define reusable modules for commonly requested resources like.

1) Networking (VPC, subnets)
2) Compute (EC2, Autoscaling Groups)
3) Storage (S3, RDS)
4) IAM Roles / Policies

By doing this, teams can manage their own infrastructure while maintining compliance with organizational standards.