Adv Devops Assignment-2

Q) Create Rest API with the serverless framework

Steps to Create a REST API with serverless framework

1) Install Serverless framework globally using the following command on the terminal:-
npm install -g Serverless
This command installs the Serverless framework on your machine globally using npm. It allows you to create, manage and deploy serverless applications across various Cloud providers using AWS.

2) Create a new service with AWS Node.js --path rest-api This command initialises a new Serverless service called rest-api. It creates a folder containing basic files and a template specifically configured for building Serverless application using node.js on AWS Lambda

3) Navigate to project directory:
cd rest-api
This command changes directory into the newly created project directory to manage files and configurations specific to our service.

4) Initialise Node.js project and install dependencies
npm init -y
npm install express serverless-http
The express dependency builds the REST API and serverless-http integrates Express with AWS Lambda

5) Edit the serverless.yml file to include:

```
service: rest-api
Provider:
  name: aws
  runtime: node.js 14.x
  stage: dev
  region: us-east-1
functions:
  app:
    handler: handler.app
    events:
      - http:
          path: /
          method: any
```

This configuration specifies the service name. AWS provider settings and defines the lambda function with HTTP event trigger.

6) Edit handler.js to add the Express app:

```
Const express = require('express');
Const serverless = require('serverless-http');
Const app = express();
app.get('/helloworld', (req, res) => res.json({
message: 'Hello world' }));
module.exports.app = serverless(app);
```

This creates a simple Express app with a single route /helloworld and exports it in a lambda compatible format

Deploy the service:
serverless deploy
Deploys the API to AWS setting up resources like lambda and API Gateway. A URL is generated for testing.

Case study for SonarQube

→ SonarQube is an open source platform used for continuous inspection. 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 creates an independent profile, while extending inherit rules from parent profile. permissions to manage quality profile are restricted to users with administrative privilege. It allows for comparison of two profile to check difference

2) using SonarCloud to analyze Github code
It is a cloud based counterpart of SonarQube that integrate directly with Github, Bit Bucket, Azur & Gitlab repositories. To get started with SonarCloud via Github Signup 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

3) SonarQube in Java IDE

SonarLint is an IDE that performs on the fly code analysis as you write code. It helps developer detect bugs, security & Code Smells directly in development environment such as Intell, Idea or Eclipse. To set it up, install the SolarLint plugin; Configure the connection with SonarQube as SonarCloud & select the project Profile to analyze Java Code This approach gives immediate feedback on code quality & maintain code from beginning

4) Analyzing python projects with SonarQube

SonarQube supports python test average, reporting but it requires third party too like coverage.py to generate the Coverage port adjust your blind process so that tol runs before & ensure, file is saved in different tools path for setup you can use for, pyTest to Configure & run test. In your 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 plugin using SonarScanner to Scan the Projects. It will check the Code

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 automate the infrastructure request process using approach it was 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 maintaining compliance with organization standards.