Adv DevOps Assignment 2

Create a Rest API with serverless framework.

Creating REST API with serverless framework is an efficient way to deploy serverless applications across various cloud providers that can scale automatically without managing services

(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 CLI globally using node packet manager (npm). This allows 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

Teacher's Sign.: _____

for AWS node.js microservices that will eventually deploy to AWS Lamda,

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 serveless.yml fil

5) With the 'sls deploy' commands serverless framework packages your applications, uploads necessary resources to AWS and set up the infrastructure.

6) Storing data in Dynamo DB : To store submitted candidate data, integrate AWS Dynamo DB.

7) Once deployed, you can test REST API using tools like curl or Postman by making post requests.

8) Adding more functionalities like list all candidates, get candidate by id.

9) AWS IAM Permissions: You need to ensure that the serverless framework is given right permissions to interact with AWS resources like Dynamo DB.

10) Monitoring and maintenance

Teacher's Sign.: _____

Case study for Sonarqube.

1. Create your own profile in sonarqube for testing project quality.

① Go to sonarCloud or set up a local instance of sonarqube.
② Once logged in, create a new project by linking your Github repository or local project.

2. Using sonarCloud to analyze your Github code.

① After linking your Github project, SonarCloud will run code quality checks for your repository.
② You will need to configure the sonar-project.properties file in your project root.

```
properties
sonar.projectKey = your_project_key
sonar.organization = your_organization_key
sonar.sources = src
sonar.host.url = http://sonarcloud.io
sonar.login = your_sonar_token
```

3. Install sonarlint in IntelliJ / Eclipse

① Open your IntelliJ IDE or Eclipse.
② Go to the plugins marketplace and search for Sonarlint. Install it.
③ After configuration, configure it by linking it to your Sonarqube profile.

Teacher's Sign.:

④ Analyze your Java project by running SonarLint to get immediate feedback on code quality.

- Analyzing Python, Node.js Projects with Sonar

① first, ensure sonarqube or sonarcloud is connected to your project.

② Analyzing Python.

Add a sonar-project.properties file to your Python Project:

```
sonar.projectKey = my-python-project.
sonar.sources = .
sonar.language = py.
sonar.python.version = 3.x
sonar.host.url = http://localhost:9000
sonar.login = your_sonar_token
```

Run the following command in the project directory:

```
sonar-scanner.
```

- Analyzing Node.js.

① Similarly, add the sonar-project.properties file

```
sonar.projectKey = my-nodejs_project.
sonar.sources = .
sonar.language = js
sonar.host.url = http://localhost:9000
sonar.login = sonar_token.
```

Teacher's Sign.: _____

Terraform "self-serve" Infrastructure Model.

① Terraform Modules for self-serve infrastructure

- Create Terraform modules that codify the standards for deploying common resources like VPCs, EC2 instances, and S3 buckets.

Example module 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
}
```

16/10/2024 00:32

Teacher's Sign:

Teams can now use this module to deploy EC2 instances with:

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

② Terraform Cloud Integration with Service Now.

- You can integrate Terraform Cloud with ServiceNow to automate the infrastructure request process.
- Using Terraform's API-driven approach, ServiceNow can trigger Terraform runs based on ticket approvals, automating resource deployment.

Example workflow:

① A product team submits a request in ServiceNow for new infrastructure.
② The request triggers a Terraform Cloud updates the ServiceNow ticket with the status and resource details.

Teacher's Sign.:

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 organizational standards.