NAME: SHRAVANI RASAM
DISA 46

62 1

ADVANCE DEVOPS
ASSIGNMENT 2.

Create a REST API with the serverless framework

The serverless framework helps you deploy applications to cloud providers like AWS using a simplified configuration

1) Install Serverless Framework: Ensure you have Node.js installed, then install the serverless framework using npm install -g serverless

2) Set up AWS credentials: Serverless uses AWS Lambda and API Gateway so configure your AWS console aws configure

3) Create your AWS Access Key, Secret key region etc.

4) Create a New Service: Create a new project using a Node.js template serverless create --template aws-nodejs --path rest-api

This creates a basic serverless service with a structure including serverless.yml and handler.js file for code

5) Define the REST API in serverless.yml:
Edit the serverless.yml to define the REST API endpoints:
For eg: service: rest-api-services
provider:
name: aws
runtime: nodejs 14.x
functions:
hello:
handler: handler.hello
events:
- http:
path: hello
method: get

The handler.js file would contain the logic

6) Deploy the service: Deploy the API to AWS Lambda and API gateway using: serverless deploy

This deploys infrastructure and you'll get a URL to access API

Teacher's Sign.:

7) Testing : Use the URL provided to access your REST API. You can test it with tools like postman or simply via your browser.

Case Study for Sonarqube

SonarQube helps to automatically review your code for bugs, vulnerabilities and code smells. The steps below cover Java, Python and Node.js analysis

1) Create a SonarQube Profile:
Go to SonarQube cloud and create an account
You can link this account to your github profile to analyze code directly from repositories
Create a project in Sonar Cloud and connect it with your github repository

2) Analyze code on Sonar Cloud:
For your Github repository, configure it with Sonar cloud. This can be done using CI pipelines (Eg. Github Actions) or manually

Teacher's Sign.: _____

Example of Github Actions YAML for Sonar Cloud

```
name: Sonar Cloud
on:
  push:
    branches:
      - main
jobs:
  sonarcloud:
    runs-on : ubuntu-latest
    steps:
    - uses : actions/checkout@v1
    - name : SonarCloud Scan
      uses : SonarSource/sonarcloud-github-
                                action@v.14
      with:
        args:>
          -Dsonar.projectkey = my-project-key

          -Dsonar.organization = my-org

          -Dsonar.host.url = https://sonarcloud.io
```

Teacher's Sign.:

3. Sonarlint setup in Java IDE:
Install Sonarlint in intelliJ IDE or Eclipse from the plugin market place

Configure it to link with your sonar cloud account for continuous quality checks

Once installed SonarLint will analyze your Java code locally for issues

4. Python Project Analysis:

1) Create a Sonar-project.properties file in the root of your python project.

Sonar.projectKey = mypythonproject.
Sonar.organization = my-org
Sonar.sources = .

Run the analysis using SonarSource Scanner
Sonar-scanner

5) Nodejs Project Analysis:

For a Node.js project the steps are similar to python. configure a sonar-project.properties file and scan the project using sonar scanner

Teacher's Sign.: _____

Example: sonar - project - properties

sonar.projectKey = my-nodejs-project
sonar.sources =
sonar.exclusions= node-modules/**,**/*.test.js

Run sonar-scanner to analyze your node.js project.

Analyze Results:
Just like with Python, the results of the analysis will be uploaded to Sonar cloud. The report can be accessed from the dashboard which will highlight issues such as missing semicolons, unused variables and more.

Key features:

1) Sonar cloud allows you to integrate with Github easily and provides a cloud based dashboard for viewing the quality of your project.

2) Sonarlist helps developers fix issues in real time as they write code, making it easier to catch issues before they are committed

Teacher's Sign.:

3) For Python and Node.js the sonar-project properties file is essential for configuring the analysis and the sonar-scanner tool helps run the analysis locally

83: Let a large organization, your centralized operations team get many repetitive infrastructure request. You can use Terraform to built a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organizations practices. Terraform cloud can also integrate with ticketing systems like Service-now to automatically generate new infrastructure requests

ns- The goal of this task is to use Terraform to create a reusable infrastructure model enabling teams to deploy and manage their own resources without involving central operations

Teacher's Sign.: _____

1. Understand the self serve infrastructure model:

In large organizations product teams often requests infrastructure resources from a central ops team. This can be repetitive and time-consuming.

By using Terraform you can create reusable modules that product teams can use independently to deploy their own infrastructure based on organization standards.

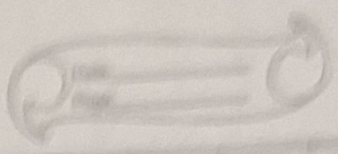
2. Creating Terraform Modules:

Modules allow you to reuse Terraform code- create a module that defines a common infrastructure component, such as an EC2 instance.

Teams can use this module in Terraform configurations

```
module "ec2" {
    source = "./ec2-instance"
    ami_id = "ami-0cbcd1234"
    instance_name = "team-app-instance"
}
```

Teacher's Sign.:



5) Automating with Terraform Cloud and Ticketing Systems

Terraform Cloud allows you to collaborate on infrastructure deployments. It can be integrated with a ticketing system like servie now to automate infrastructure requests

This integration ensures that infrastructure is deployed in compliance with the organization security and governance standards

Key Tools to Use:

1) Serverless framework: for deploying REST APIs

2) SonarQube / Sonar Cloud: for code quality analysis

3) SonarLint in IntelliJ / Eclipse for on-the-fly Java Analysis

4) Terraform: for infrastructure automation and self-service infrastructure.

Teacher's Sign.: _____