Name: Sanket Hore
class: D15A
Roll No: 30

05/05

Advanced Dev-Ops Assignment - 2

Q1.) Create a REST API with serverless framework.

Creating a REST API using the serverless framework involves setting up your development environment, creating a new serverless project & configuring AWS Lambda functions to handle your API routes

Prerequisites:-
- AWS Account to deploy the API
- Ensure you have Node.js installed
- Make sure you have installed serverless globally via npm
- Install AWS CLI to easily configure AWS credentials.

Step:1 Open a bash terminal & create a new project for your REST API. The serverless framework will generate a starter Template for you.

```
serverless create --template aws-nodejs --path
my-rest-api
cd my-rest-api
```

This will create a new directory 'my-rest-api' with a few basic configuration

Step:2 Configure serverless.yml file which is the core configuration file where you'll define your Lambda functions, API Gateway events & other AWS services.
The provider file specifies the AWS as the cloud provider & sets the Lambda runtime to Node.js 18.

functions field defines Lambda functions & their corresponding API Gateway HTTP events.
Plugins field Adds the serverless-offline plugin for local testing

Step:3 Creating Lambda function handlers, you need to write the logic for each lambda function inside the handler.js file.

Step:4 Install the necessary plugins, including serverless-offline for local testing
npm init -y
npm install serverless-offline --save-dev

step:5 You can test your API locally using serverless-offline. This plugin emulate API Gateway & AWS Lambda on your local machine.
serverless offline.
Visit http://localhost:3000 ; to test the routes locally.

step 6: Once satisfied with the local testing, deploy the REST API to AWS
serverless deploy
This will create AWS lambda functions & set up an API gateway to route requests to the appropriate Lambda functions.

step 7: After deployment, serverless will provide the API gateway endpoint in the terminal. You can test the deployed API using curl.

curl https://<api-id>.execute-api.us-east-1.amazonaws.com/dev/user/

step8: You can use AWS console to monitor Lambda execution tracks API usage using logs. Serverless also provides the commands for logs.
serverless logs -f <name of api>

Case study for sonarqube

i) Creating a Profile in SonarQube for Testing Project Quality :-
SonarQube is an open-source platform used for continous inspection of code quality. It supports multiple programming languages & can detect code vulnerabilities & other issues. To create your own profile in SonarQube for testing project quality :-

- Install SonarQube : If you are working locally, download & install SonarQube from the official website or run it using Docker.

- Create a SonarQube Profile :-
Once SonarQube is running, log in to the dashboard (default credentials are admin/admin). Go to Quality Profiles. Click on 'Create' to build a custom profile for your project. Assign this profile to your project by navigating to the project settings & choosing your custom profile for analysis.

- Run a Quality Analysis :- Use SonarQube Scanner to run the analysis by configuring the sonar-project.properties file in your project. Execute the command 'sonar-scanner' to perform the analysis. Results will be seen on dashboard

ii) Using SonarCloud to Analyze Github code :-
SonarCloud is the cloud-base version of SonarQube, which can directly integrate with Github for seamless code analysis.

- Create an Account on SonarCloud using your Github credentials.
- Navigate to 'My Projects' & click on 'Analyze New Project'.
Select the repo you want to analyze from your Github & give SonarCloud necessary permissions.

· Setup Github Actions:
Sonar Cloud integrates well with Github Actions for continuous
Configure a .github/workflows/sonarcloud.yml file in your pro
automatically analyze code on every push or pull request.

· Run the Analysis:
The analysis will run automatically when code is pushed to Github
can view the results directly in Sonar Cloud.

iii.) Installing SonarLint in Java IDE to Analyze Java Code:-
SonarLint is an IDE plugin that helps detect issues & enforce co
as you write code. Here's how to use it:
· Install SonarLint: In IntelliJ IDEA, go to Plugins/
Marketplace & search for SonarLint. Install the plugin & restart
IDE.
· Analyze Java Code: Once installed, SonarLint will automatically
analyze your Java code in real-time as you type. To manually trig
an analysis, click on individual file & select 'Analyze with SonarLi
SonarLint will highlight any issues or code smells directly in
editor with suggestions for improvement.

iv.) Analyzing a Python Project with SonarQube:-
SonarQube can also be used to analyze Python projects. Follow these s
· Install SonarQube Scanner for Python:
Ensure Python is installed on your system. Add Python-specific ana
support in SonarQube by navigating to Admin → Marketplace &
installing the Python Plugin.

- Configure SonarQube for Python:

Create a 'sonar-project.properties file in the root of your Python project:

```
sonar.projectKey = python_project
sonar.sources = .
sonar.language = py
sonar.python.version = 3.x
```

- Run the Analysis

use the 'sonar-scanner' command to analyze the project. The results will show issues like coding standards violation in the Python code

Analyzing a Node.js Project with SonarQube:-

Similar to Python, SonarQube can analyze Node.js projects for code quality.

- Install Javascript plugin from the SonarQube Marketplace.
- Configure Your Node.js Project by adding a properties file in the root of your Node.js project.

```
sonar.projectKey = nodejs_project
sonar.sources = .
sonar.language = js
```

- Ensure your Node.js project has the proper files & run sonar-scanner to initiate analysis.

Q3) At a large organization, your centralized operations team may get many repetitive infrastructure requests. You can use Terraform to build a "self-serve" infrastructure model that lets product teams manage their own infrastructure independently.

In a large organization, managing repetitive infrastructure requests can become a time-consuming task for centralized operations teams. Using Terraform, a powerful infrastructure-as-code (IaC) tool, enables the creation of a self-serve infrastructure model, allowing product teams to manage their own infrastructure independently while still adhering to organizational standards.

Benefits of Self-Serve Infrastructure with Terraform:

1) Decentralized Management: By creating a modular infrastructure, teams can deploy & manage their services without relying on the centralized ops team.

2) Consistency: Terraform modules can encapsulate best practices & standards for deploying services, ensuring that all teams follow policies & security guidelines.

3) Efficiency & Scalability: Product teams no longer need to submit requests for every infrastructure need. Instead they can reuse pre-approved Terraform modules to quickly launch compliant infrastructure, reducing time to market.

4) Automating & Integration: By integrating with ticketing systems like ServiceNow, Terraform Cloud can automatically trigger workflows to provision new infrastructure requests without manual intervention.

Steps to Implement Self-Serve Infrastructure with Terraform Modules:

Codify Infrastructure Standards with Terraform Modules:

- Develop Terraform modules that encapsulate your organization's standards for infrastructure. For eg:, modules for deploying AWS services such as EC2 instances, S3 buckets, or VPCs can be built with pre-defined security config. naming conventions & resource limits.

- Example of a Terraform module for deploying an EC2 instance:

```
module "ec2_instance" {
  source        = "path_to_ec2_module"
  instance_type = "t2.micro"
  key_name      = "org-approved-key"
  ami           = "ami-12345678"
  tags = {
    Name = "compliant-instance"
  }
}
```

Enable Teams to use Modules Independently:

- Provide product teams with access to the Terraform modules repository.

- Teams can configure their own deployments by passing specific parameters to the modules while still adhering to the core standards set in the module. This allows them flexibility in deployment while ensuring security & consistency

3.) Integrate Terraform Cloud with Ticketing Systems :-

- Use Terraform cloud to manage the state & execution of infrastr
code. Terraform Cloud provides features like version control, au
approvals, & integration with other tools.

- Integration with ServiceNow or similar ticketing systems can stre
the process. ServiceNow triggers a Terraform Cloud workspace to run
appropriate Terraform plan. This reduces human intervention & speeding up

4.) Monitor & Maintain Compliance :-

- Use Terraform 'policy as code framework, Sentinel, to enforce polic
infrastructure is provisioned. This ensures teams remain compliant with o
zational standards & security requirements.

- Sentinel Policies can ensure that :
  * Resources like databases are encrypted
  * Proper tags are applied for cost allocation
  * Instances are deployed only in approved regions.