Name: Ishan Kiran Joshi
Roll no: 21 DISC Div:

Advance DevOps Assignment 2

Create a REST API with the serverless Framework

Steps to create a REST API with serverless Framework

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 including AWS.

Create a new service with AWS Node.js template:-

serverless create --template aws-nodejs --path rest-api

This command initializes a new serverless service called rest-api. It creates a folder containing basic files, and a template specifically configured for building serverless applications using Node.js on AWS Lambda.

Navigate to the project directory:-

Cd rest-api

This command changes directory into the newly created project directory to manage files and configurations specific to our service.

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.

Edit the serverless.yml file to include:

service: rest-api

provider:

name: aws

```
  runtime: nodejs 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 provi
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)

7] Deploy the service

```
serverless deploy
```

Deploys the API to AWS setting up resources like Lamb
and API Gateway. A URL is generated for testing.

Test the deployed API:

Curl https://<api-id>.execute-api.<region>.amazonaws.com/dev/helloworld

Using above command it returns a JSON message.

{"message": "Hello World"}.

Redeploy after updates:

serverless deploy

After modifying the code, redeploy it to update the API with our changes.

Remove the Service.

serverless remove

The above command removes all AWS resources associated with the API, ensuring that there are no charges for unused services.

Case Study for Sonarqube.
Create your own profile in Sonarqube for testing project quality.
Use Sonar Cloud to analyse your Github code
Install Sonarlist in your Java Intellij IDE or Eclipse IDE and analyse your Java code.
Analyse Python project with Sonarqube.
Analyse Node.js project with Sonarqube

1] Create your own profile in Sonarqube! -
Download and install Sonarqube from the official website
Unzip the file and start the server by running:
./bin/windows-x86-64/StartSonar.bat
This launches Sonarqube locally and can be accessed at http://localhost:9000
Log in to Sonarqube using the default credentials (username: admin; password: admin). After logging in, change the password.
Navigate to Projects tab, click on 'Create New Project' assign an project Key and name and generate a project token.

2] Use Sonar Cloud to analyze your Github code:-
Signup For Sonar Cloud from the official website. Using your Github account.
In Sonar Cloud under Projects > Create Project, choose your Github repository and grant Sonar Cloud access to it.
Add a Sonar-project.properties file in the root of your repository with the following code:-

sonar.projectKey = <your_project_key>
sonar.organization = <your_organization>
sonar.host.url = https://sonarcloud.io

d) Use SonarScanner to analyse the code by running the following command: sonar-scanner
This uploads analysis results from your local project to SonarCloud.

3) Install SonarLint in your Java IntelliJ or Eclipse IDE and analyse your Java code.

a) Install SonarLint by going onto IntelliJ or Eclipse going to plugins/Marketplace and search for SonarLint. Install and restart your IDE.

b) In the IDE, configure SonarLint by linking it to your Sonarqube or SonarCloud project to sync rules and profiles.

c) Open a Java project and use SonarLint to analyse. It will display issues directly in the IDE while coding. SonarLint provides real-time feedback on code quality based on Sonarqube rules.

4) Analyse python project with Sonarqube:-
Set up a python code in a project and ensure that SonarQube is running locally.
Download and configure SonarScanner from its official website and in the sonar-project.properties file, edit to include the following:-
Sonar.projectKey=python_project
Sonar.language=py
Sonar.sources=

Run the analysis of the project by executing the following from the project directory, sonar-scanner. The results will be pushed to your local SonarQube server and the analysis is visible on the dashboard.

5] Analyse Node.js project with SonarQube:-

Set up a Node.js project

In SonarQube, ensure that all JavaScript / TypeScript plugins have been installed. Pluging can be installed from the 'Market place tab' in SonarQube.

Create a Sonar project, properties file in your project root and include the following in it:-

Sonar project Key = node_project

Sonar. ~~project~~ language = js

Sonar Sources = .

Run the analysis of the project by executing the 'Sonar-Scanner' command.

SonarQube will analyse the Node.js project and show results on the dashboard, highlighting code quality, bugs and vulnerabilities.

At a large organisation your centralised operations team may get many repetitive infrastructure requests. You can see Terraform to build a self-service 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 organisation, allowing teams to efficiently deploy services in compliance with your organisation's practices. Terraform Cloud can also integrate with ticketing systems like ServiceNow to automatically generate new infrastructure requests.

Creating a self-service infrastructure model using Terraform for a large organisation involves the following steps.

Step 1: Define Infrastructure Standards:-

Establish clear standards and best practices for infrastructure deployment, including naming conventions, resource types, tagging policies and security compliance. This foundation ensures consistency across the organisation.

Step 2: Create Terraform Modules:-

Develop reusable Terraform modules based on your organisation's standards. Each module defines resources and configurations, allowing teams to deploy infrastructure efficiently and consistently.

Step 3: Set-up Terraform Cloud or Enterprise:-

Use Terraform Cloud or Enterprise for centralised management of configuration and state files, enabling collaboration and access control for infrastructure changes.

Step 4: Configure Version Control
Integrate Terraform modules with version (e.g. Github). This tracks changes, facilitates collaboration and ensures proper versioning for updates and compatibility.

Step 5: Integrate with ServiceNow or other tic systems.
Integrate with systems like ServiceNow to au infrastructure requests. This triggers Terraform streamlining the process for teams.

Step 6: Provide Documentation and Training
Create documentation and training for using Ter modules and submitting requests, helping teams understand and follow best practices.

Step 7: Monitor and Support:
Monitor the usage of the self-service model and provide ongoing support to users. Gathering feedback helps identify pain points and areas for improvement ensuring that the infrastructur remains compliant and efficient.

Step 8: Iterate and Improve:
Regularly review and update Terraform modules, documentation and policies based on feedback an changing organisational needs continuous iteration enhances efficiency, security and compliance.

By following above steps, organisations can enable product teams to manage their own infrastructure through a standardised Terraform approach.