



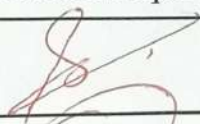
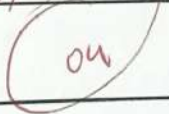
Vivekanand Education Society's Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Advance DevOps Lab

Experiment No.	Assignment-2
Title.	Create REST API
Roll No.	13
Name	Granesh Gupta
Class	D15 C
Subject	Advance DevOps
Lab Outcome	LO6: To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework
Signature:	
Grade:	

Assignment - 2.

04/08

Create a REST API with the serverless framework

Step 1:- To create 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 globally using npm. It allows you to various cloud providers, including AWS.

Create new service with AWS Node.js template

```
Serverless create --template aws-nodejs --path rest-api
```

This command initialises new serverless service called rest-api. It creates a folder containing basis file & a template specifically. configures application

- Initialize node.js project & install dependencies

- `npm init -y`

- `npm install express serverless-http`

Edit serverless-http integrate building the REST with AWS Lambda.

- Edit the `serverless.yml` file to include service rest-api

provider:

name: aws

runtime: nodejs 24.26

stage: dev

region: us-east-1

function:

app:

handler: app

events:

FOR EDUCATIONAL USE

- http

path: /

method: any

This configuration specifies the service on AWS provider settings & define their lambda functions with http event trigger.

- Edit handler.js to add express app.

- `const express = require("express");`

`const app = express();`

`app.get("/hello", (req, res) => res.json({msg: "hello"}));`

This creates a simple express app with single route

Deploy the app & the API gateway. A virtual lab is deployed for testing.

- Test the deployed API.

- curl `https://<apid>execute-api<region>amazon.com/dev/hello`.

Using the above returns json response `{ "msg": "hello" }`.

Case Study for Sonarqube.

i) Setting up your profile for sonarqube. Creating profile in sonarqube allows devops to analyze the quality of their projects & track improvement over time.

Steps: Install sonarqube & set it up locally

- Once logged in create a new project in sonarqube by providing a project name & key.
- Add the sonarqube properties file to the root directory of the project which contains necessary configurations
- Use sonarqube scanner to analyze project & uploads the result to the dashboard

ii) Using Sonarqube to analyze github code.

Steps: Signup & connect your github repository

- Setup github actions to run sonarqube scans whenever code is pushed into the repository. This ensures continuous code analysis.
- create sonar project property files with the necessary configuration in the root directory of the project. Sonarcloud scanner is triggered everywhere

iii) Sonarlist for realtime code analysis in IDEs: Sonarlist is a plugin for intelliJ IDEA & eclipse to perform code analysis. Steps: Install the plugin for intelliJ IDEA, go to files > settings > plugins find sonarlist & install it can be linked with sonarqube instance to sync rules & quality profiles

- SonarLint runs automatically as you write & flag issues directly in the editor

- iv) Analysing Node.js project with SonarQube
- Steps: verify javascript plugin available SonarQube instance.
 - configure project by adding properties in sonar project files.
 - you can also combine SonarLint with SonarQube for a more comprehensive Java analysis.
 - Use Sonar-Scanner to analyze the project

At a large organization your centralized operations team may get infrastructure sequences you can terraform to build self-service infrastructure model that lets product teams manage infrastructure independently. Create & use terraform modules that codify the standard for deploying & managing services in your organization. Terraform cloud can also integrate with ticketing options. like service now to automatically generate new infrastructure requests.

- Step 1:- Define Infrastructure standard establish class standards & best practices for infrastructure deployment including resource types, tagging policies & security compliances.

- Step 2:- Create terraform module based on the organizations standards. Deploy a common resource using EC2 bucket instance & S3 buckets.
eg:- variable "instance type" {
default = "t2.micro"

```
resource "aws_instance" "example" {  
  ami = "ami-12us678"  
  instance_type = var.instance_type  
  tags = {  
    name = "example-instance"  
  }  
}
```

```
output "instance-id" {  
  value = instance.example.id  
}
```


Terraform cloud integration to automate infrastructure request process can run on ticket approval automating & resource deployment

Step 3: Creating terraform modules for team
Define reusable modules for commonly used modules such as

- networking VPC subnets
- compute
- storage
- IAM roles

By this team can manage their own infrastructure while maintaining compliance with internal organization standards.