



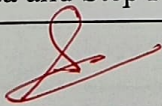
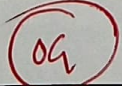
Vivekanand Education Society's Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Advance DevOps Lab

Experiment No.	Assignment-2
Title.	
Roll No.	03
Name	Shiven Bansal
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:	

Advance DevOps Assignment-2

1) Create a REST API with the Serverless Framework.

The Serverless Framework is a powerful tool designed to streamline the development and deployment of serverless applications. It automates the process of setting up cloud services like AWS Lambda and API Gateway, allowing developers to focus more on writing application logic rather than managing infrastructure.

• Key Components of Serverless Architecture-

- AWS Lambda - Executes your Application logic without needing to provision servers.

- API Gateway - Serves as the interface for exposing the Lambda functions as HTTP APIs.

- CloudWatch - Used for logging and monitoring the performance of Lambda functions.

• Steps :-

1) Installed Required Tools

- Node.js - The Serverless framework is built on Node.js

- AWS CLI - Interacts with AWS services, helping to manage deployments.

- Serverless Framework - Install it globally using npm.

Set up a new Serverless Project

Create a new project by running a command like `serverless create`. This generates a template with default configurations.

The core configuration happens in `serverless.yml` file, where Lambda functions and HTTP routes are defined.

- 3.) Define API endpoints - In `serverless.yml`, you can map HTTP methods like GET, POST, PUT and DELETE to specific Lambda functions. These functions handle the business logic for creating, reading, updating and deleting resources in a REST API.
- 4.) Deploy the API - Deploying the REST API is straightforward with the Serverless Framework. It automates the setup of all necessary AWS services, such as Lambda and API Gateway. After deploying, a public URL is provided by API Gateway, allowing users to interact with the API.
- 5.) Testing & Monitoring - After deployment, test the API using tools like Postman or curl. AWS CloudWatch provides logs for your Lambda function, allowing you to monitor performance & errors.

• Benefits

- 1.) Simplified Deployment
Automatic Cloud Infrastructure setup reducing complexity of deploying.
- 2.) Automatic Scaling - AWS Lambda automatically scales based on demand, handling fluctuations in traffic seamlessly.
- 3.) Cost Efficiency - You only pay for compute time used by your Lambda, making it ideal for applications with varying traffic.
- 4.) No infrastructure management - AWS handles the maintaining of servers part.

Case Study for SonarQube

Setting up your profile for SonarQube - Creating a profile in SonarQube allows developers to analyse the quality of their projects and track improvements over time.

Steps: Install SonarQube and set it up locally

- Once logged in create a new project in SonarQube by providing a project name & key.

- Add the sonar-project.properties file to the root directory of the project, which contains necessary configurations.

- Use SonarQube Scanner to analyse your project and uploads the result to the dashboard.

Using SonarQube to analyse Github code

Steps - Sign up and connect your Github repository.

- Setup Github actions to run SonarCloud scans whenever code is pushed into the repository. This ensures ~~continuous~~ continuous code quality analysis.

- Create Sonar project properties file with the necessary configurations in the root directory of the project. SonarCloud Scanner is triggered everytime.

Sonarlint for real time code analysis in IDEs - Sonarlint is for IntelliJ IDEA and Eclipse that performs code analysis

Steps: Install the plugin for IntelliJ IDEA, go to files > Settings > plugins, find Sonarlint & install.

Sonarlint can be linked to SonarQube instance to sync rules & quality profiles.

- SonarLint runs automatically as you ~~to~~ write code and flag issues directly in the editor.

iv.) Analyzing python project with SonarQube

- Steps : Ensure SonarQube is running - Configure SonarQube for python (specify the source files).
- Execute Sonar Scanner from the root of your project
- Analysis result will be uploaded to SonarQube.

v.) Analysing Node.js project with SonarQube.

- Steps - Verify javascript plugin is available in your SonarQube instance.
- Configure project by adding properties in the Sonarproject file.
- You can also combine SonarLint with SonarQube for a more comprehensive javascript analysis.
- ~~Use~~ Use Sonar-Scanner to analyse the project.

Q3.) At a large organisation, your centralised operations team may get many repetitive infrastructure requests. You can ~~use~~ use terraform to build a "Self service" infrastructure model that lets product team manage their own infrastructure independently. You can create and use Terraform modules that codify the standard for ^{deploying} ~~deployment~~ 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.

ps - 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 : Setup Terraform Cloud or Enterprise :-

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

- Step - 4 : Configure Version Control

Integrate Terraform modules with version control (eg - Github). This tracks changes, facilitates collaboration and ensures proper versioning for updates and compatibility.

- Step-5 : Integrate with Service Now or other ticketing systems :-

Integrate with systems like ServiceNow to automate infrastructure requests. This triggers Terraform runs, streamlining the process for teams.

- Step-6 : Provide documentation and Training :
Create documentation and training for using Terraform modules and submitting requests, helping teams understand and follow best practices.
- Step-7: Monitor and Support
Monitor the usage of self service model and provide ongoing support to users. Gathering feedback helps identify ~~pro~~ pain points and areas for improvements ensuring that the infrastructure remains compliant and efficient.
- Step-8 : Iterate and improve
Regularly review and update Terraform modules documentation and policies based on feedback and changing organisational needs. Continuous iteration enhances efficiency security and compliance.

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