# Advance - DevOps
## Assignment - 2



05/05

Q1) Create a REST API with serverless framework.

→ Steps to Create a REST API with serverless framework.

1) Install serverless framework globally using the following command on the terminal

- npm install -g serveless

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 provider, including AWS.

2) Create a new service with AWS node.js template.

Serverless create --template aws-~~nagios~~ nodejs --path rest-api. This command initializes a new serverless service called rest-api. It create a folder containing basic files and a template specifically configured for building serverless application using node.js on AWS Lambda.

3) Navigate to the Project Directory:

cd rest-api

4) Initialize Node.js project and install dependencies

npm init -y

npm install express serverless-http

The express dependency build the REST Api and serverless-http integrate Express with AWS Lambda

5) Edit the serverless.yml file to include

```
server: 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 provider settings and defines the Lambda function with HTTP event trigger.

6) Edit settings of handler.js to add Express app.

```
const express = require ('express');
const serverless = require ('serverless-http');
const app = express();
app.get ('/helloworld', (req, res) => res.json ({ m
  'Hello AWSDevops' }));
module.exports.app = serverless (app);
```

This creates a simple Express app with a single /helloworld and exports it in a Lambda comp format

7) Deploy the service:

```
serverless deploy
```

Deploys the API to AWS setting up resources li Lambda and API Gateway. A URL is generated testing.

8) Test the deployed API.

```
curl https://<api-id>.execute-api.<region>.ama
aws.com/dev/helloworld
```

Using above command, it returns a JSON message
{ " message " : " Hello Advdevops " }.

9) Redeploy after updates.
serverless deploy
After the modification the code, redeploy it to update the API with our changes

10) Demove the service:
serveless, remove.
The above command removes all AWS resources associated with the API, ensuring that there are no changes for unused services

Q.2) Case study for Sonarqube:

- Create your own profile in Sonarqube for testing project quality
- Use SonarCloud to analyze your Github code.
- Install Sonarlint in your Java Intellij IDE or Eclipse IDE and analyse your Java Code.
- Analyse Python project with sonarqube
- Analyse Node.js project with sonarqube.

→ Create your own profile in SonarQube

1) Download and install sonarqube from the official website. Unzips 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
2) Log into the SonarQube using the default credentials (user-name: admin, password: admin). After logging in, change the password.
3) Navigate to Projects tab, click on 'Create New Project' assign a project key and name and generate a project token.

- Use SonarCloud to Analyze Github Code

1) Sign up for SonarCloud from the official website using your Github account
2) In SonarCloud under Projects > create project choose your Github repository and grant SonarCloud access to it
3) 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
4) Use SonarScanner to analyze the code by running the following command : sonar-scanner.

- Install SonarLint in your Java IntelliJ or Eclipse IDE and analyze Java code
1) Install SonarLint by going into IntelliJ or Eclipse, go to the Plugins / Marketplace and search for SonarLint. Install and restart your IDE
2) In the IDE, configure SonarLint by linking it to your SonarQube or SonarCloud project to sync the rules and profiles.
3) Open a Java project and use SonarLint to analyze it. It will display issues directly in the IDE while coding
4) Some of the errors are termed as Code smells, Code Reliability, Scalability etc
SonarLint provides real-time feedback on code quality based on sonarQube rules.

- Analyze python project with SonarQube:
1) Set up a Python code in a project and ensure that sonarQube is running locally
2) Download and configure Sonar scanner from its non-critical official website and in the sonar-project-properties file, edit to include the following
sonar.key : python project

sonar.language : pj
Sonar.sources: " ";

3) Run the analysis of the project by executi the following from the project directory.
sonar-scanner.
The results will be pushed to your local Sona server and the analysis is visible on dashboard

- Analyse Node.js project with SonarQube.

1) Set up a Node.js project
2) In SonarQube ensure that all JavaScript Type script plugins have been installed. Plugin can be installed from the Marketplace tab in SonarQube
3) Create a sonar-project properties file in yo project root and include the following in it
sonar.projectkey: node-project
sonar.language: js
Sonar.sources: ,
4) Run the analysis of the project by executi the sonar-scanner command
SonarQube will analyze the Node.js proje and show results on the dashboard, highli code quality, bugs and vulnerabilities

3) 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. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization's practices. Terraform Cloud can also integrate with ticketing systems like ServiceNow to automatically generate new infrastructure requests.

1) Self-Serve Infrastructure with Terraform:

Terraform is widely-used infrastructure as code (IaC) tool, offers a solution to this problem by enabling a self-serve infrastructure model. In this model, the product teams are empowered to provision and manage their own infrastructure without needing to go through the central operations team for every request. Instead the teams can provide Terraform modules that codify the organizations infrastructure standards. These modules act as reusable templates that product teams can use to deploy infrastructure that is consistent, secure and compliant with organizational policies.

2) Terraform modules for standardization:

Terraform modules are reusable templates that simplify the process of deploying infrastructure by codifying the standards for managing...

resources (eg: servers, databases, networking). Product teams can leverage these module to deploy resources that comply with the organizational policies ensuring consistency and compliance across different environmen

- In such large organization, the operation team can create standardized modules for co infrastructure patterns, such as deploying web servers, databases or kubernetes clusters
- This helps the team to maintain security performance and cost-efficiency, as teams a not manually configuring infrastructure from scratch

3) Integration with Ticketing systems like Servicen

- Terraform cloud or Enterprise can integrate wit Servicenow tools, automating the process of generating new infrastructure requests.
- For this, the integration between ServiceNow & Terraform cloud allows ServiceNow to act as the central hub for managing infrastructure provisions
- Requests are sent via routes that tells through the approval workflows based on the policies.
- Service Now is updated in real-time with the provisioning status while ensuring compliance with standards. Service Now is updated on real-time with the provisioning track progress. This model optimizes the efficiency by delegating infrastructure management to product teams.