SAIRAM.V
D15C
27

# ASSIGNMENT - 2

05/05

8

1] Create a Rest API with the serverless framework.

Step 1:- Install serverless framework using the following command on the terminal

npm install -g serverless

This command installs the serverless framework on your machine. It allows you to create, manage and deploy serverless applications across various cloud providers, including AWS.

Step 2: Create a new service with AWS Node.js template:

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

This command initialises a new serverless service called rest-api. It creates a folder containing basic file and template specifically configured for building serverless application.

Step 3: Navigate to project directory.

cd rest-api

Step 4: Initialise node.js project and install dependencies.

npm init -y.

npm install express serverless-http.

Express dependency builds REST API and serverless-http integrates Express with AWS Lambda.

Step 5: Edit the serverless.yml file

```
service: rest-api
provider:
  name: aws
  runtime: nodejs 18.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.

Step 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.

Step 7:- Deploy the service

```
serverless deploy
```

Deploys the API to AWS setting up resources like lambda and API Gateway.

Step 8:- Test the deployed API:

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

Using the above command, it returns a JSON message `{"message": "Hello world"}`.

Step 9] Replay after updates:

serverless deploy

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

Step 10: 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.

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 Intelli IDE or Eclipse IDE and analyze your Java code.
- Analyze Python project with Sonarqube
- Analyze Node.js project with Sonarqube

To start, you can either install official Sonarqube on your system by going to their official website or by pulling and initiating a Docker image of the same.

→ Using docker image:

1) docker pull sonarqube

→ this will install the image of sonarqube in your docker container

2) docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube

-> this command will run the sonarqube server on our local system at localhost:9000.
Login using 'admin' username 'admin' password then change the password.
In this way, a profile is created on sonarqube.
Now, navigate to 'Projects', create a new project. Assign a project key and name and generate a project token
Now, to analyze your Github code, go to the 'Projects' tab. There, click on the 'setup' of Import Github. This prompts to set up a configuration.
To analyze you Github code, we can use Jenkins a middleman software. After running sonarqube, make the necessary configurations on Jenkins.
1) Add sonarqube server.
2) Give the name, URL and token of project.
Later, set up your repository by making sure t configure a webhook to notify Jenkins with cha
In Jenkins, create a pipeline project and add a script that will monitor the changes. After saving the project configuration, we run the Jenkins project. This will trigger sonarQube analy based on the rules and thresholds in sonarQube.

For using sonarlint on a Java IDE, inital step is to install a Java IDE. Here it is IntelliJ IDEA Navigate File, then Settings and Plugins. Under Marketplace s Sonarlint. Click on install and restart the IDE. Right click on the Java project which you want

get analyzed. Right click on this folder, select Analyze with SonarLint. It will display the results directly in SonarLint tool window, highlight the code issues.

To analyze a Python project, set up Sonarqube as mentioned before. Also, you need to download SonarScanner locally on your system. To make it accessible on your terminal, add the bin directory to your PATH environment variable. Now, to make your Python project being able to be analyzed, a ~~sonar-properties~~ sonar-project.properties file is required. Create this file in the root directory of the project. Some of the information needed for this file are projectKey, projectVersion, etc. After saving this, open the terminal in the project path and run command sonar-scanner. This will analyze and submit the report of analysis to the Sonarqube server.

A similar setup is required for analyzing a Node.js project. Using the same ~~sonar~~ Scanner that was installed, the project can be analyzed. Again, a sonar-project.properties file is required to be created at the root directory containing the projectKey, projectVersion, etc. Optionally, you can install and configure ESLint and with SonarQube. To run the analysis, run the command 'sonar-scanner' in a terminal opened on the path of your project. After the analysis is complete, the report of the analysis will be available on a SonarQube server.

Q.3] In organizations, managing repetitive infrastructure requests strain centralized operations teams, slowing down pro
Adopting a self-serve infrastructure model using Terraform decentralizes this responsibility, empowering pr
team to manage their own infrastructure.

⇒ Terraform, a leading Infrastructure as a code (IAC) enables organizations to automate and manage infrastr
using declarative configuration files which reduces mans
errors, improves operational efficiency and making it sca
central to a service self-serve model are reusable,
version-controlled Terraform modules, which allows to
standardize infrastructure deployments.
Terraform Cloud offers secure, centralized state
management, preventing from overwriting each other's
changes. Sentinel, Terraform Cloud's policy-as-code
framework, enables organization to enforce governanc
policies and integrate them into CI/CD pipelines. Terra
Cloud's Integration with ticketing system like Service
streamlines infrastruce provisioning by automating wo
and approvals, reducing manual intervention while
maintaining compliance. It also supports automated
workflows like GitLab cost estimation features
allow teams to forecast expenses, while multi-clo
and hybrid-cloud support ensures scalability acros
diverse environments. Security is embedded IAM po
state encryption and integrations with secrets
management tools, ensuring adherence to framewo
like SOC2 and HIPAA.

Terraform's drift detection ensures that infrastructure remains aligned with desired state, automatically identifying and correcting any manual changes. Implementing a self-serve infrastructure model with Terraform enables large organizations to manage their infrastructure with greater autonomy, efficiency and control. By leveraging Terraform, teams can deploy infrastructure that adheres to organizations security with operational bottlenecks thus making Terraform an ideal tool for modern infrastructure management in complex organizations.