Name: Aditya Adinath Kirtane.
Roll no. 26, Division: DISC

Page No.
Date

Advance devops - Assignment No. 2.

creating a rest API using the serverless framework is a great way to build a scalable and efficient API with minimal infrastructure management.

Prerequisites:-
AWS account
Node.js installed.
Serverless Framework installed (npm install -g Serverless.

step 1:- install the serverless framework.
npm install -g serverless.

step 2:- create a new service.
create a new service by running following command.

Serverless create --template aws-nodejs --path rest-api-serverless
this create a directory called rest-api-serverless with some basic configuration files.

step 3:- Update Serverless.yml.
The Serverless.yml is the configuration file that defines your service, including function, event and resources.

```
service: rest-api-serverless

provider:
  name: aws
  runtime: nodejs18.x
  stage: dev
  region: us-east-1

functions:
  createUser:
    handler: handler.createUser
    events:
      - http:
          path: users
          method: post

  getUser:
    handler: handler.getUser
    events:
      - http:
          path: users/{id}
          method: get
  updateUser
    handler: updateUser
    events:
      - http:
          path: users/{id}
          method: put
BillingMode: PAY_PER_REQUEST
```

Page No.
Date

ep 4:- Create the Handler functions
In the handler.js file, implement your lambda functions.

```
onst AWS = require('aws-sdk');
onst dynamoDb = new AWS.DynamoDB.
ocumentClient();

y {
await dynamoDb.delete(params).promise();
return {
    statusCode: 200,
    body: JSON.stringfy({message:
        'User deleted Successfully'}),
};
} catch(error) {
    return {
    statusCode: 500,
    body: JSON.stringfy("could not
    delete user"),
    };
};
```

Step 5:- Deploy the Service
Serverless deploy

Step 6:- Test the API.
After deployment, the Serverless Framework will return the API Gateway endpoint.

step 7:- Monitor and manage.
- To view logs:- Serverless logs -f created
- To remove the service when done.
  Serverless remove.

Q.2. Case Study for Sonarqube.
- create ur own profile in Sonarq for testing project quality..
- use Sonarcloud to analyze your github code.
- install Sonarlint.
- Analyze python project with Sonarqube.
- Analyze node.js project with Sonarqube.

Step1 Create your own profile in Sonarqube for testing project quality.

(1.) Install SonarQube
- Download and install SonarQube from Sonarqube downloads.
- start the SonarQube Server.
- open the dashboard.

(2.) Login and Set up profile.
- Log in to SonarQube.
- Go to Quality Profiles from top menu.
- Click create to make a profile.

Page No.
Date

3) Run analysis on your project.
Install and configure SonarQube Scanner on your local machine configure the Sonar project properties.

4) Review results.
After the scan complete, go to the SonarQube dashboard.
checks the detected issues, bugs, and vulnerabilities based on custom profile.

step2. Use Sonarcloud to analyze your Github code.
objectives:- analyze your Github repository.

procedure:-
1. Set up Sonarcloud.
2. Link a Github repository.
3. Integrate Sonarcloud.
4. Run analysis

step3:- Install Sonarlint in IntelliJ or Eclipse IDE for Java code
1. Install Sonarlint plugin.
2. configure Sonarlint:-
3. Analyze code

step 4:- Analyze a python Proje
with SonarQube
(1.). Prepare python project
(2.). Run SonarQube Scanner.
(3). Review Results

Step5:- Analyze nodejs project.
with SonarQube.
(1) Prepare NodeJs Project.
(2) Run SonarQube Scanner.
(3) Review Results.

Q.3 ~~Ans~~ At a large organization your
centralized operation team may
get many repetitive infrastrutare
request.

lets product team manage their
own infrastructure independently
terraform modules that codify the
standards
organization, allowing team to
efficiently deploy service.
ServiceNow to automatically
generate new infrastructure reques

Building a Self Serve infrastructure
model with terraform for large
organization.

Page No.
Date

key concepts:
self-serve infrastructure
terraform modules
terraform cloud.
terraform integration.

Step 1:- Designing Terraform Module for Self Service.
1) create Standard terraform module
2) Encapsulate Standards
3) Version control for modules

Step 2:- Implementing terraform cloud for collaboration.
1. Set up workspaces
2. Set up permission.
3 Automate runs

Step 3:-
1. Setup Service Now.
2. Service-Now terraform cloud Integration.
3. Automate request fulfilment

Step 4:- Deploying and managing infrastructure by teams

Step 5:- monitoring and continuous compliance