Name: RONAK S KATARIYA Class: T.E Div: D15A Roll No: 23

Subject: ADV. DEVOPS Topic: Ass - II Date: Page No: 1

Q.1. Create a REST API with the Serverless Framework.

⇨ * Serverless framework is an open source tool that simplifies the development & deployment of serverless applications. By leveraging this framework, you can focus on writing business logic without worrying about server management.

* Steps to create REST API using the Serverless framework:

- Install Node.js & NPM: Ensure you have Node.js & npm installed on your system by 'node -v' & 'npm -v'.

- Install the Serverless Framework: Use npm to install the Serverless Framework globally: 'npm install -g serverless'

- Create a new serverless project: ◦Run command ↵ 'serverless create --template aws-nodejs --path rest-api-serverless' ◦Then navigate to the project directory: cd rest-api-serverless'

- Configure AWS Credentials: ◦Configure your AWS credentials for the serverless framework:
'serverless config credentials --provider aws --key <your key> --secret <secret key>'

- Define API in serverless.yml file:
  ◦ Open serveless.yml file & define your Rest API endpoints under functions section. Eg.

```
service: rest-api-serverless
provider:
  name: aws
  runtime: nodejs14.x
functions:
  hello:
    handler: handler.hello
    events:
      - http:
```

AASMAN

path: hello
method: get

- Now comes the REST API Logic..
  - In handler.js file, write the logic for API. Eg.

```
module.exports.hello = async (event) => {
  return {
    statusCode: 200,
    body: JSON.stringify({
      message: "Hello from the Serverless Rest API!",
    }),
  };
};
```

- Deploy the API :
  - Use Serverless command to deploy your API to AWS lambda

`serverless deploy`

  - After deployment, the framework will give you a URL for your API. You can test it by making an HTTP request to that URL.

- Test the API :
  - Test your API using tools like Postman (used VS code extension)

https://your-api-id.execute-api.region.amazonaws.com/dev/hello

This is the whole process which will create a fully serverless REST API using AWS Lambda, API Gateway and the Serverless Framework.

Name: _____ Class: _____ Div: _____ Roll No: _____

Subject: _____ Topic: _____ Date: _____ Page No: _____

Q2. Case Study for SonarQube.

ED * SonarQube is a popular open-source tool used for continous inspection of code quality. It automatically reviews code to detect bugs, code smells & security vulnerabilities. It supports multiple programming languages & integrates seamlessly with CI/CD pipelines.

* Steps for SonarQube Analysis:

- Create your own profile in SonarQube:
  - Sign up for SonarQube or SonarCloud & create new profile.
  - Navigate to Quality Profiles under the Quality Gates menu to set up a profile that reflects your projects specific need

- Use SonarCloud to analyze your Github code:
  - Link your Github repo to SonarCloud.
  - SonarCloud will scan your repo & generate a report of code quality issues like bugs, vulnerabilities & technical debt.
  - You can check the analysis in the SonarCloud dashboard, where you'll get metrics like code coverage, duplications and security hotspots.

- Install SonarLint in IntelliJ or Eclipse for real-time feedback:
  - Install the Sonarlint plugin from the market place in IntelliJ IDEA or Eclipse.
  - Then after that link your SonarCloud account with the IDE.
  - SonarLint will provide immediate feedback about code quality as you write code.
  - You can then also fix issues in code in real-time & ensure adherence to your quality profile.

AASMAN

- Analyze a Python Project with SonarQube:

  - Set up a Python project in your SonarQube dashboard
  - Configure the sonar-projects.properties file in your Python project with the relevant details (project key, source directory).
  - Run the SonarQube scanner to analyse the Python Project:
    'sonar-scanner'
  - SonarQube will generate a detailed report showing bugs, code smells and security vulnerabilities in your Python code.

- Analyze a Node.js project with SonarQube:
  - Similar to python project, set up the sonar-project.properties file for your Node.js project.
  - Run the SonarQube scanner in Node.js project directory to analyze the code quality.
  - SonarQube will identify issues such as unhandled promises, performance issues and bad coding practices in JavaScript code.

So, SonarQube helps you ensure code quality across multiple projects and languages.
By setting up the correct quality gates and profiles, you can enforce coding standards & reduce technical debt.

Subject: _____ Class: ____ Div: ____ Roll No: ____

Topic: _____ Date: _____ Page No: _____

Q.3. Using Terraform for Infrastructure Automation.

⇒ * Terraform is an Infrastructure as Code (IaC) tool that allows you to define and provision infrastructure using code. It enables teams to automate infrastructure management, ensuring consistency & scalability.

- Scenario in a Large Organisation:
  - In large organisations, the centralized operations team receives frequent and repetitive infrastructure requests from various products teams.
  - So, instead of manually handling each request, you can create a self-serve model using Terraform.
  - This will allow product teams to manage & deploy their own infrastructure while ensuring compliance with organizational policies.

* Steps to set up Terraform for a self serve infrastructure model:

- Create Reusable Terraform Modules:
  - Define terraform modules that represent common infrastructure components like virtual machines, database networking configurations, etc.
  - These modules should follow organisation's best practices for security, networking and scalability.

Eg:

```
module "web_server" {
    source = "./modules/web_server"
    instance_type = "t2.micro"
    ami = "ami-123456"
}
```

AASMAN

- Allow teams to Manage Infrastructure Independently:
  - Each product team can use the predefined Terraform modules to spin up their infrastructure, avoiding the need for central approval for every change.
  - This accelerates the deployment process, ensuring teams can focus on delivering products faster.

- Integrate with Terraform Cloud for collaboration:
  - Used Terraform Cloud to centrally manage state files & run automated workflows for infrastructure changes.
  - So, teams can push their terraform configurations to a version-controlled apply the changes after a review process.
- Enforce Organizational Standards with policies:
  - Use Sentinel (policy-as-code framework in Terraform Cloud) to enforce compliance policies.
  - These policies are applied at the organization level, ensuring all teams follow the same infrastructure.
- Integrate with Ticketing Systems:
  - Terraform Cloud integrates with tools like ServiceNow to create automated infrastructure requests. When a product team needs new resources, terraform can automatically generate a ticket in ServiceNow, which triggers the infrastructure provisioning process.
  - This workflow minimizes manual intervention, reduces human error and ensure traceability for all infrastructure changes.

* Benefits of using Terraform:
- Scalability
- Consistency
- Automation.