Assignment-2

Q.1 Create a REST API with serverless Framework.
Following are the steps.

1) Install serverless framework globally using :-
   install -g serverless.

2) Create a new service with AWS node js :-
   serverless create --template aws-nodejs --path rest

3) Navigate to the project directory & install project dependencies.
   cd rest-api
   npm init -y
   npm install express aws-sdk
   or npm install express serverless-http

4) Modify the ~~s~~ yml ~~(se~~ file (serverless.yml)
   include :-

```
service: rest-api
provider:
  name: aws
  runtime: nodejs18.x
  region: us-east-1
functions:
  getItem:
  createItem.
```

— This configuration specifies the service na
AWS provider setting & defines lambda function with
HTTP events.

Teacher's Sign.:

6) Edit handler.js to add the express app

```
const express requires (express)
const serverless requires (serverless:http)
app.get (:hello world) (req-res) => region
    ({message: 'Hello World'})
```

7) Deploy the service:

```
serverless the service
```

8) Test the deployed API

```
curl http://<api-id>.execute-api(region).amazo
                                        com
                                        hell
```

9) Redeploy after update:

```
Serverless deploy
```

10) Remove the service: [once the job is done]

```
serverless remove
```

Case study For sonarqube:

As organizations scale their software development processes, maintaining high-quality code becomes critical, that's when Sonarqube comes in play. It is an open-source platform that continuously inspects code quality by detecting bugs, vulnerabilities, code smells & security issues.

* Objectives:

1) Improvement of overall code quality.
2) Automate the detection of bugs, code smells, vulnerabilities in the early stage.
3) Standardize code quality.
4) Reduce technical debt & establish a sustainable development process.

Sonarcloud for analyzing github code:

* Create your own profile:

1) Download & install sonarqube from its official website, unzip & configure.
2) Login with default credentials:
   username/password: admin/admin.
3) Click on create new project, assign a name & generate a project token thru:-
   profile → security → generate user token.

* Sonarcloud to analyze github code:

1) Signup/Login in sonarcloud.
2) On Sonarcloud, under project → analyze new project → choose your github account & select a repo.
click on setup & proceed with the default settings

Teacher's Sign.: _____

3) In the repo, add a workflow yaml file (sonarqu
4) Generate a security token. Then settings
→ secrets & variables → actions. Add a new secre
called 'sonarcloud'
5) Push changes to your repo.
The results can be seen by navigating to
the project dashboard.

* Sonarlint in eclipse IDE & Java Intellij.
1) Install sonarlint by going to plugins (marketp
2) Configure it by linking it to sonarqube/ sonarc
3) To trigger a full project analysis:
Open the java project in eclipse. Sonarlint
will automatically start code analysis as you
edit your file.
4) Sonarlint then displays issues like bugs, co
smells etc.

* Analyse python project with sonarqube:
1) Download sonarscanner (in Sonarqube) & add
path to your system path.
2) In Sonarqube, create a new project & gene
a security token.
3) Add properties to root of your python projec
Create a sonar-project.properties.
4) add project key, url, login source & language.
5) Run Sonar-scanner.

Teacher's Sign.:

Analyze node js project:
1) Install sonarqube & sonarscanner.
2) In the root of node.js project, create a file sonar-project.properties.
3) Results can be viewed from sonarqube project dashboard.

Terraform 'self service' infrastructure model.
Terraform modules for "self-service" infra.
- Create Terraform modules that codify the standards for deploying common resources like VPCs, EC2 instances & S3 buckets.

Example module for an EC2 instance.

ec2-module/main.tf

```
variable "instance_type" {
    default = "t2.micro"
}
resource "aws_instance" "example" {
    ami = "ami-12345678"
    instance_type = var.instance_type
    tags = {
        Name = "example-instance"
    }
}
```

ec2-module/outputs.tf

```
output "instance_id" {
    value = aws_instance.example.id
}
```

Teams can now use this to deploy EC2 instances

Teacher's Sign.:

(2) Terraform Cloud integration with service Now:-
- You can integrate terraform cloud with service
to automate the infrastructure request process.
- Using Terraform's API-driven approach, service N
can trigger Terraform runs based on ticket
approvals automating resource deployment.

Example workflow
(1) A team submits a request in ServiceNow
for new infrastructure.
(2) The request triggers Terraform Cloud updates
ServiceNow ticket with the status & resource detail
(3) Creating Terraform module for teams.
Define reusable modules for commonly requested
resources like
1) Networking (VPC, subnets)
2) Compute (EC2)
3) Storage (S3)
4) IAM Roles/ Policies.

— By doing this, teams can manage their own
infrastructure while maintaining compliance with
organizational standards.