

## Assignment. 2

Shreya Sawant

DISA 54

Ad. devops

48

create a REST API with the serverless framework.

For developing API some prerequisites must be met as Node.js - installed on the local machine to manage dependencies and run scripts.

serverless framework - installed globally using npm to facilitate the creation and deployment of serverless application.

AWS account - required to access AWS Lambda and other services.

AWS CLI configuration - properly configured AWS credentials to allow the serverless framework to interact with aws services.

The process of creating REST API with the serverless framework consists of several key steps -

① setting up the environment -

The first step involves installing Node.js and the serverless framework, followed by configuring AWS credentials using AWS CLI.

② creating new serverless service -

A new service is created using the serverless CLI command.

```
serverless create --template aws-nodejs --path
```

This command initializes a new project structure.

③ Defining API endpoints -

The serverless.yml file is crucial for defining the service configuration and the API endpoints. Each endpoint corresponds to a function that can be invoked via HTTP requests.

Teacher's Sign: \_\_\_\_\_



### case study for sonarqube.

- create your own profile in sonarqube for testing project quality.
- Use sonarqube to analyze your github code.
- Install sonarlint in your java intelliJ ide or eclipse ide and analyze your java code.
- Analyze python project with sonarqube.
- Analyze nodejs project with sonarqube.

create your own profile in sonarqube for testing project quality.

- ① Download and install sonarqube from the official website. follow the installation instructions.
- ② start sonarqube using following command -  
`./bin/<your-os>/sonar.sh start`
- ③ Access sonarqube at <https://localhost:9000> in web browser.
- ④ Now create profile for that firstly login to the sonarqube. Then navigate to quality profile in the top menu.  
 click on create to make a new profile.  
 select the language you want to create profile for (python, java, javascript)  
 customize your rules by adding or removing rules according to your project's needs. Then save.



- Use sonarqube to analyze your github code.
- ① Go to the sonarqube and sign up using github account.
- ② After creating account create new repository and import it.
- ③ Follow the on screen instruction to configure analysis. This typically involves adding a configuration file to your repo.

```
sonar.projectkey = your-project-key  
sonar.organization = your-org  
sonar.sources =
```

- ④ Now run the analysis - create a workflow file .github/workflows/sonarcloud.yml.
- ⑤ view the results by navigating through sonarcloud project.

- Install sonarlint in IntelliJ IDEA or Eclipse IDE and analyze your java code.

- ① open your IntelliJ IDE or Eclipse.
- ② Go to the plugins marketplace and search sonarlint, Install it.
- ③ After configuration, configure it by linking it to your sonarqube profile.
- ④ Analyze your Java project by running sonar to get immediate feedback on code quality.



### • Analyzing python project with sonarqube.

① First, ensure sonarqube or sonarcloud is connected to your project.

### ② Analyzing python

add a sonar project properties file to your python project:

```
sonar.projectkey = my-python-project  
sonar.sources =  
sonar.language = py  
sonar.python.version = 3.x  
sonar.host.url = http://localhost:9000  
sonar.login = your-sonar-token.
```

Run the following command in the project directory:

```
sonar-scanner.
```

### • Analyzing Node.js

① similarly add the sonar-project properties file.

```
sonar.projectkey = my-node.js-project  
sonar.sources =  
sonar.language = js  
sonar.host.url = http://localhost:9000  
sonar.login = sonar-token.
```

```
ec2-module/output.tf
```

```
output "instance-id" {  
  value = aws-instance.example.id  
}
```

Teams can now use the module to display EC2 instance with

```
module "ec2" {  
  source = "../ec2-module"  
  instance-type = "t2.medium"  
}
```

- Terraform cloud integration with service now.

You can integrate terraform cloud with service now to automate the infrastructure request process.

Using terraform's API driven approach, servicenow can trigger ~~the~~ terraform runs based on ticket approval - automating resource deployment.

Example workflow -

1. A product team submits a request in service now for new infrastructure.

2. The request triggers a terraform cloud update, the servicenow ticket with the status and resource details.

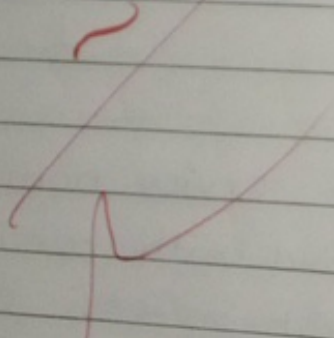


Date \_\_\_\_\_  
Page \_\_\_\_\_

- creating terraform modules for teams define reusable modules for commonly requested resources like

- ① Networking (VPC, subnets)
- ② compute (EC2, Autoscaling groups)
- ③ storage (S3, RDS)
- ④ IAM roles/policies

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





§ 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 standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organizational practices. Terraform cloud can also integrate with ticketing system like ServiceNow to automatically generate new infrastructure requests.

⇒ • Terraform modules for self serve infrastructure

① create terraform modules that codify the standards for deploying common resources  
VPCs, EC2 instances and S3 buckets.

② Example module for an EC2 instance-

```
ec2 - module / main - if
```

```
variable "instance_type" {
```

```
    default = "t2.micro"
```

```
}
```

```
resource "aws_instance" "example" {
```

```
    ami = "ami-12345678"
```

```
    instance_type = var.instance_type
```

```
    tags = {
```

```
        Name = "example-instance"
```

```
    }
```

```
}
```