Shivam R Prajapati D15C 41

05/05

06/10/24 Adv DevOps Assignment No: 2 DATE: 06/10/24

Q.1 Create a rest api with serverless framework.

Soln:-

① Install serverless node package using npm install -g serverless

② Create a serverless.yml configuration file with following code and save it.

```
service: my-rest-api
provider:
  name: aws
  runtime: nodejs18.x
  region: us-east-1
function:
  serverInfo:
    handler: handler.serviceInfo
    events:
      - http:
          path: serverInfo
          method: get
```

Here name of my function is serverInfo.

③ Create a new file in the samedirectory as handler.js with the Rest API code that the server will return on calling it.

④ In the same directory where the configuration file was created, initialize the serverless using command serverless. select Login/Register. A new browser window will open asking for us to sign in (Create account if it does not exist)
After setting up, type command serverless again to initialize project.

DATE:

5. Select create a new app.
6. Name your app.
7. Connect your AWS to serverless framework. You will be shown multiple options, you can select any. Here I am using Create AWS IAM Role (Easy and recommended) method to connect. In this a URL will be shown, you have to open it into your browser and sign in with AWS account. Scroll down and click on create stack
8. Now our serverless service is ready to deploy. to it use serverless deploy. After successfull deployment, we will get a URL of serverless REST Api endpoint
9. Visit the provided URL. As we can see, that the server returns the system information.

Q.2 a) Create your own profile in sonarqube for testing project quality.

Soln
1. Create a local project in Sonarqube dashboard
2. Create quality profile. Select language and name of the quality Profile.
3. Activate security checks according to your project
4. Assign created quality profile to project
   Go to the project section and select the project we just created. Then go to project settings situated at right top corner and select quality profiles. Add language of your project and select the created quality profile
5. Set up Jenkins. Open Jenkins dashboard and create new item.

(6) Name the project and select item type as pipeline.
(7) Go to the pipeline script section and paste the following code and save it. Here we are analyzing the Python project named as requests, a simple HTTP Library
(8) Click on Build Now.
As we can see our analysis is completed in jenkins dashboard.
Our analysis is passed in Sonarqube also in accordance with the specified quality profiles which we created earlier.

Q.2 b) Use Sonarcloud to analyze your github code.

Soln:

(1) Sign up on Sonarcloud.
(2) Create a new project in sonar cloud.
(3) Fork the ~~repo~~ repository of your choice into your github account.
(4) Click on "import an organization from github".
(5) Give access permission to repository you want to analyze.
(6) A summary will be shown, choose free plan and click on "create organization".
(7) Select the repositories under created organizat" and set up them.
(8) Select analysis method
(9) Analysis will be started. Wait for sometime until analysis gets completed. As we can see our analysis is completed successfully with list of issues shown in

DATE:

the dashboard. We can see any issue in detail by clicking on it.

(10) In our example, one of the maintainability issues is shown.

Q.2 c) Install ~~sonarlit~~ Sonarlint in your Java Intellij ide or eclipse ide and analyze your Java code.

Soln:-

(1) Open Eclipse IDE
(2) Go to the help menu and select eclipse marketplace
(3) In the eclipse marketplace, search for ~~Sonarlit~~ Sonarlint
(4) In the search results, find Sonarlint and ~~check~~ click the install button, Restart eclipse IDE after installing Sonarlint.
(5) Create a new Java Project.
(6) Name your project and finish setting up.
(7) Create a new sample java file (eg calculator.java) in the newly created project.
(8) Run manual analysis. You can trigger a manual code analysis by right-clicking on the project, folder, or file in the Project explorer and selecting SonarLint → Analyze.
(9) See results of analysis in bottom Tab. Here we got a error from sonarlint
(10) Create a new package com.myapp.calculator
(11) Here we are getting warning to solve this add package com to the top of our code in Calculator.java file and move Calculator.java to correct location such as /src/com/calculator.java

2 D) Analyze python project with Sonarqube

In:- ① Clone the project repository of your choice
② Create the sonar-project.properties File in the root of the Flask project directory
③ Install dependencies Before running the analysis, ④ install the required dependencies of the flask project. The requirements-dev.txt file contains development dependencies, including testing tools
pip install -r requirements/dev.txt
④ Now we have to create a ~~local~~ global analysis token, to do so, go to My Account Go to security tab. Name your token and select type as Global analysis token and click on generate token. A new token will be generated, copy it ~~is~~ in clipboard
⑤ Now we would use Sonar Scanner ~~GIT~~ CLI to analyze our project using below command Note: Replace -D sonar.login with your own generated token.
⑥ Open the project dashboard using provided ~~to~~ url in output or look in projects section of Sonarqube dashboard. As we can see our analysis is completed.

.2 E) Analyze node.js project with Sonarqube.

oln: ① Clone the Node.js project you want to analyze. For this ~~expto~~ example let us use the Express repository

② In the root directory of your Node.js, create a sonar-project.properties file to configure the analysis.

(3) Now we have to create a global analysis token, to do so, go to My Account. Go to security Tab
(4) Name your token and select type as Global analysis token and click on generate token.
(5) A new token will be generated, copy it in clipboard
(6) Now we would use Sonar Scanner CLI to analyze our project using ~~bat~~ command.
(7) Open the project dashboard using provided url in output or look in project section of SonarQube dashboard
(8) As Now we can see our analysis is completed.

Q 3 At a large organization, your centralized ~~op~~ operations team may get many repetitive infrastructure request. You can use Terraform to build a "self service" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organization, allowing teams to efficiently deploy services in compliance with your organization's practices. Terraform cloud can also integrate with ticketing systems like service Now to automatically generate new infrastructure requests.

Soln:- Setting up S3 bucket using Terraform scripts and modules.

(1) Create a parent directory terraform_modules, inside it create S3- bucket directory.

(2) Create 4 files named main.tf, variables.tf, output.tf and terraform.tfvars

(3) After creating above files inside S3-bucket directory, the module structure should look like this taking base directory as terraform-modules. After creating all directories and files, it should look like this

```
— terraform-modules
  └— S3-bucket
       ├— main.tf
       ├— outputs.tf
       ├— terraform.tfvars
       └— variables.tf
```

(4) Now initialize git repository in S3-bucket directory

(5) Create empty repository on Github and push the changes to it.

(6) Create a Terraform cloud account on app.terraform.io.

(7) Create organization with organization name.

(8) In your organization, create a new workspace that will be linked to your Terraform code repository. Select Github and version as Github.com.

(9) Authorize Terraform cloud.

(10) Install Terraform cloud as Github app so it can see and manage changes in github repository.

(11) select repository from which terraform config files will be looked from.

(12) Review final settings and click on create.

(13) Give tag name and bucket name which would be created in your aws account and click on save variables and then click on start new plan.

(14) Give run name and select run type. We got error as we had not provided credentials of our AWS account.

(15) Generate access keys for AWS account. Search users and select users. Create new user or select existing one if you already have. Create access key while creating new user.

(16) Go to settings → variable sets and click on create variable set. Name the variable set and set scope to Apply globally. Select variable category as environmental variable. Enter Key name as AWS_access-Key-id and Value as your secret which you have get from AWS account and mark it as sensitive. Similarly add AWS-SECRET_ACCESS-Key and mark it as sensitive. After adding all keys click on create variable set.

(17) Go to workspaces on Terraform cloud and select S3 bucket-Terraform and click on New Run situated at top right ~~click~~ corner

(18) click on confirm and apply to apply the plan.

(19) Add comment to confirm applying pan. Our plan finished applying

~~20~~

(20) Visit AWS S3 bucket dashboard to see newly created bucket. Our bucket in visible on AWS.

***