

How to implement SonarQube in CI/CD pipeline for Application pipeline

SonarQube : SonarQube is a popular tool used for continuous code quality inspection.

Code Quality Analysis: SonarQube analyzes code to detect issues related to bugs, vulnerabilities, and code smells. This helps improve the maintainability, reliability, and security of software.

Automated Code Review: It provides automated feedback during the development process, which can catch issues early before they become larger problems.

Technical Debt Management: SonarQube helps manage technical debt by identifying areas of the codebase that need improvement and offering suggestions for refactoring.

Standardization: It enforces coding standards and guidelines, ensuring consistency across a codebase, which is especially valuable in larger teams.

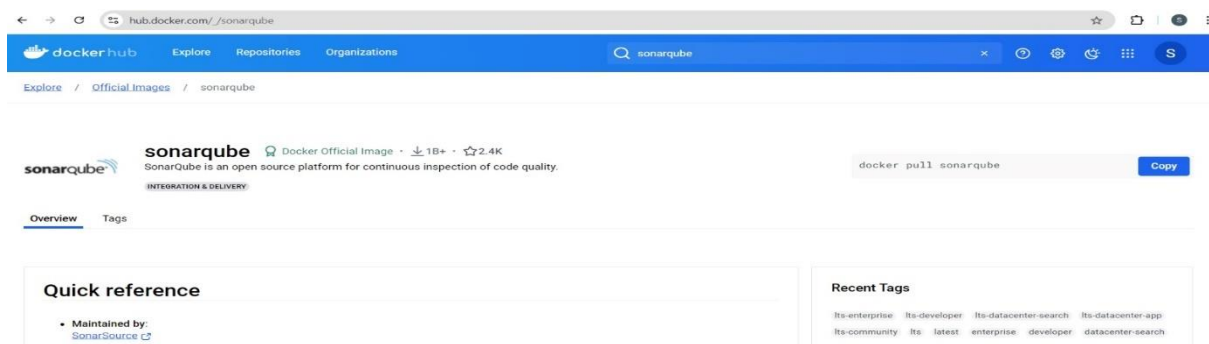
Integration: It integrates with CI/CD pipelines, allowing for continuous analysis of code as part of the build process, thus ensuring that code quality is maintained throughout the development lifecycle.

Historical Tracking: It tracks code quality over time, helping teams to monitor improvements or regressions and make informed decisions about codebase health.

Customizable Rules: Teams can customize rules and quality profiles to match their specific needs and industry standards.

Reporting: SonarQube provides detailed reports and dashboards that help teams visualize code quality metrics and track progress.

Step1: Go to docker hub and search SonarQube and then go inside SonarQube, copy the docker pull sonarqube command in your docker machine.(Need to login our docker machine, and then need to run docker pull command)



```

root@satish:~#
root@satish:~# docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59bed36c86
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest
root@satish:~#

```

Step2: Check docker images and then create container from this sonar Qube image.

```

root@satish:~# docker run -d -p 9000:9000 --name sonar sonarqube
fc656762fea46e74bb877658a347aa23dac3d272b141ea4ff41c2a75c5082db9
root@satish:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
fc656762fea4   sonarqube     "/opt/sonarqube/dock...  About a minute ago Up About
a minute      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp sonar
root@satish:~# docker ps

```

Docker ps

Docker run -d -p 9000:9000 --name sonar sonarQube

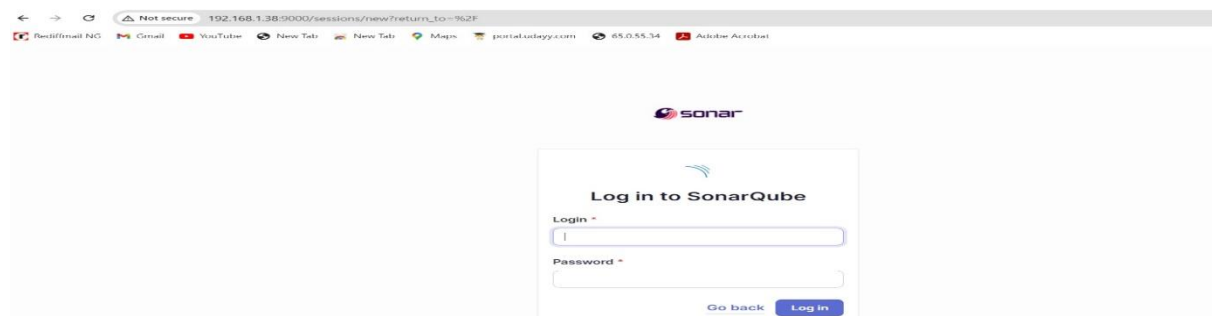
1st port no : host port, 2nd port no: container port, we can check the sonar Qube port details from docker hub

Step3: Check Docker system IP and run in our browser:

```

root@satish:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:00:07:11:20:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.38/24 brd 192.168.1.255 scope global dynamic enp0s3
        valid_lft 89027sec preferred_lft 89027sec
    inet6 fe80::a00:27ff:fe1e:2939/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f5:8b:44:c1 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:f5ff:fe8b:44c1/64 scope link
        valid_lft forever preferred_lft forever
5: vethdc76e5e@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 9e:a9:05:67:f2:94 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::9ca9:5ff:fe67:f294/64 scope link
        valid_lft forever preferred_lft forever

```



First time login with admin and admin, then change the password and then it login with new password.

Step4: Now create one project for CI/CD implementation, and fill required attributes.

← → ↻ Not secure 192.168.1.38:9000/projects/create?mode=manual

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

1 of 2

Create a local project

Project display name *

Project key *

Main branch name *

The name of your project's default branch [Learn More](#)

Cancel Next

arqube Projects Issues Rules Quality Profiles Quality Gate

1 of 2

Create a local project

Project display name *

 ✓

Project key *

 ✓

Main branch name *

The name of your project's default branch [Learn More](#)

Cancel Next

← → ↻ Not secure 192.168.1.38:9000/projects/create?mode=manual&setncd=true

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. [Learn more: Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

Previous version

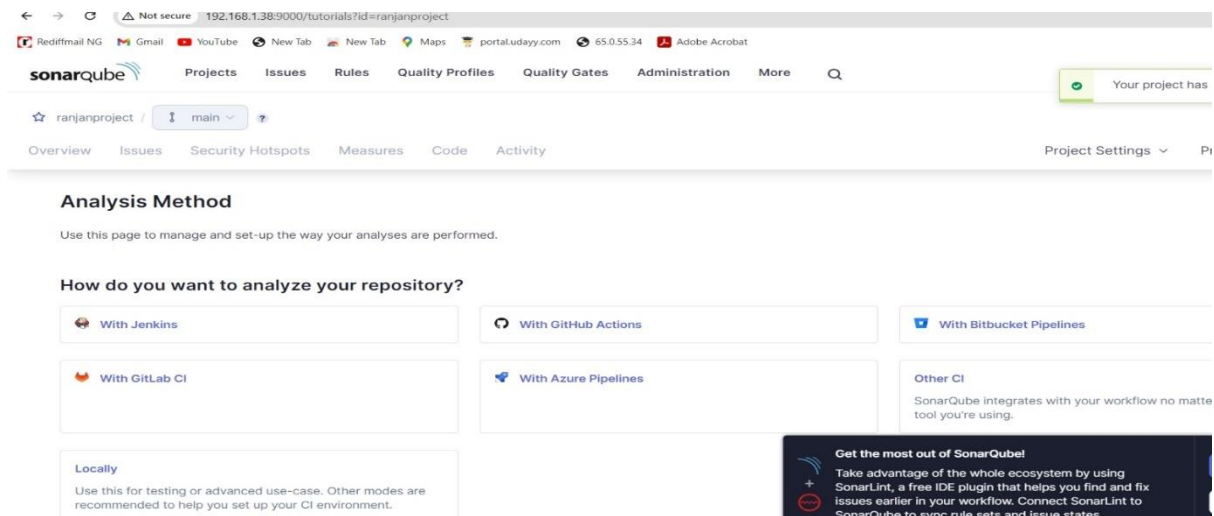
Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version

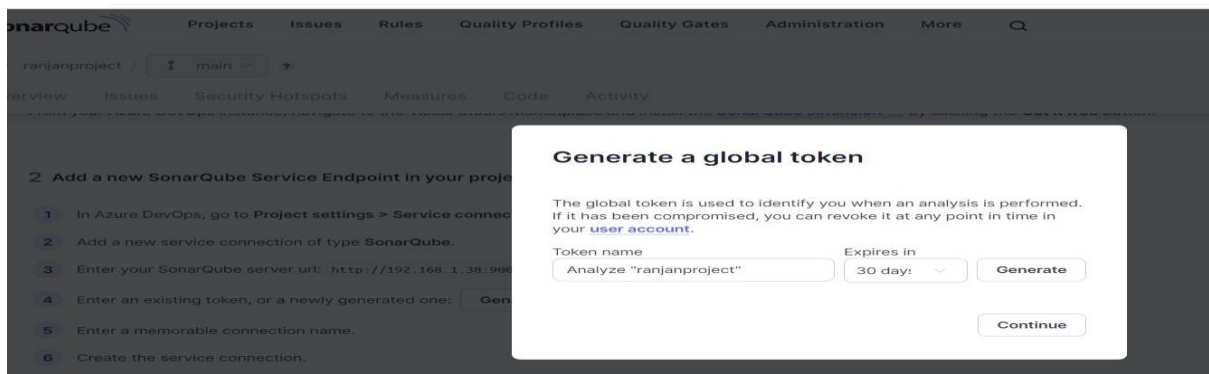
Step5: Choose our repository for generate credential to be used in to create service principle in CICD pipeline azure devOps portal.



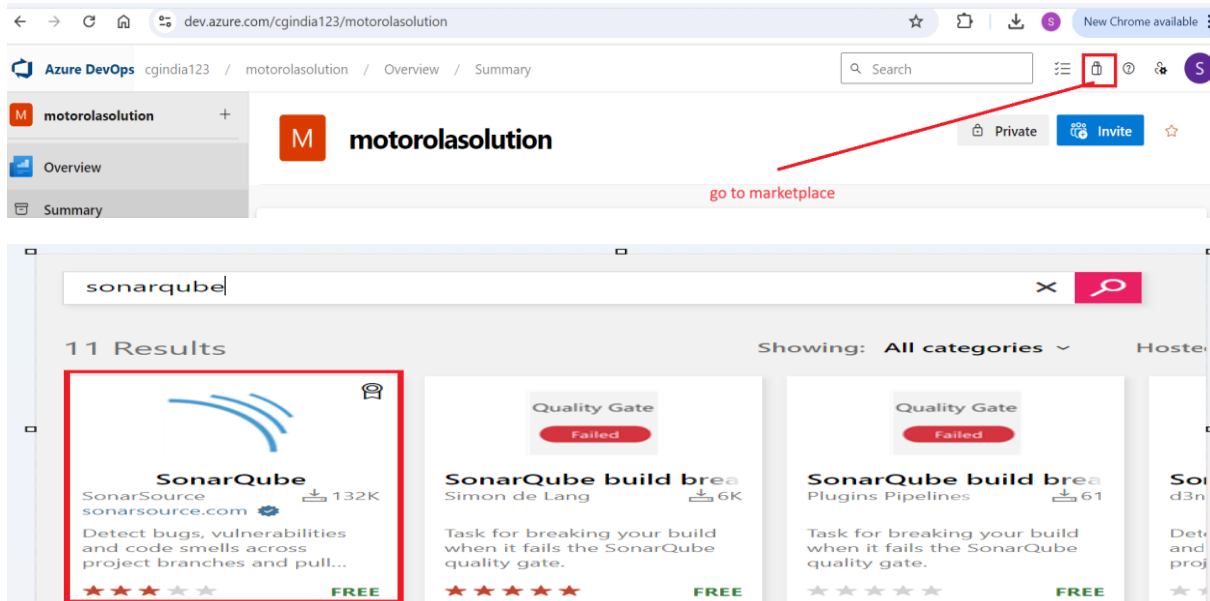
2 Add a new SonarQube Service Endpoint in your project

- 1 In Azure DevOps, go to **Project settings** > **Service connections**.
- 2 Add a new service connection of type **SonarQube**.
- 3 Enter your SonarQube server url: `http://192.168.1.38:9000`
- 4 Enter an existing token, or a newly generated one: [Generate a token](#)
- 5 Enter a memorable connection name.
- 6 Create the service connection.

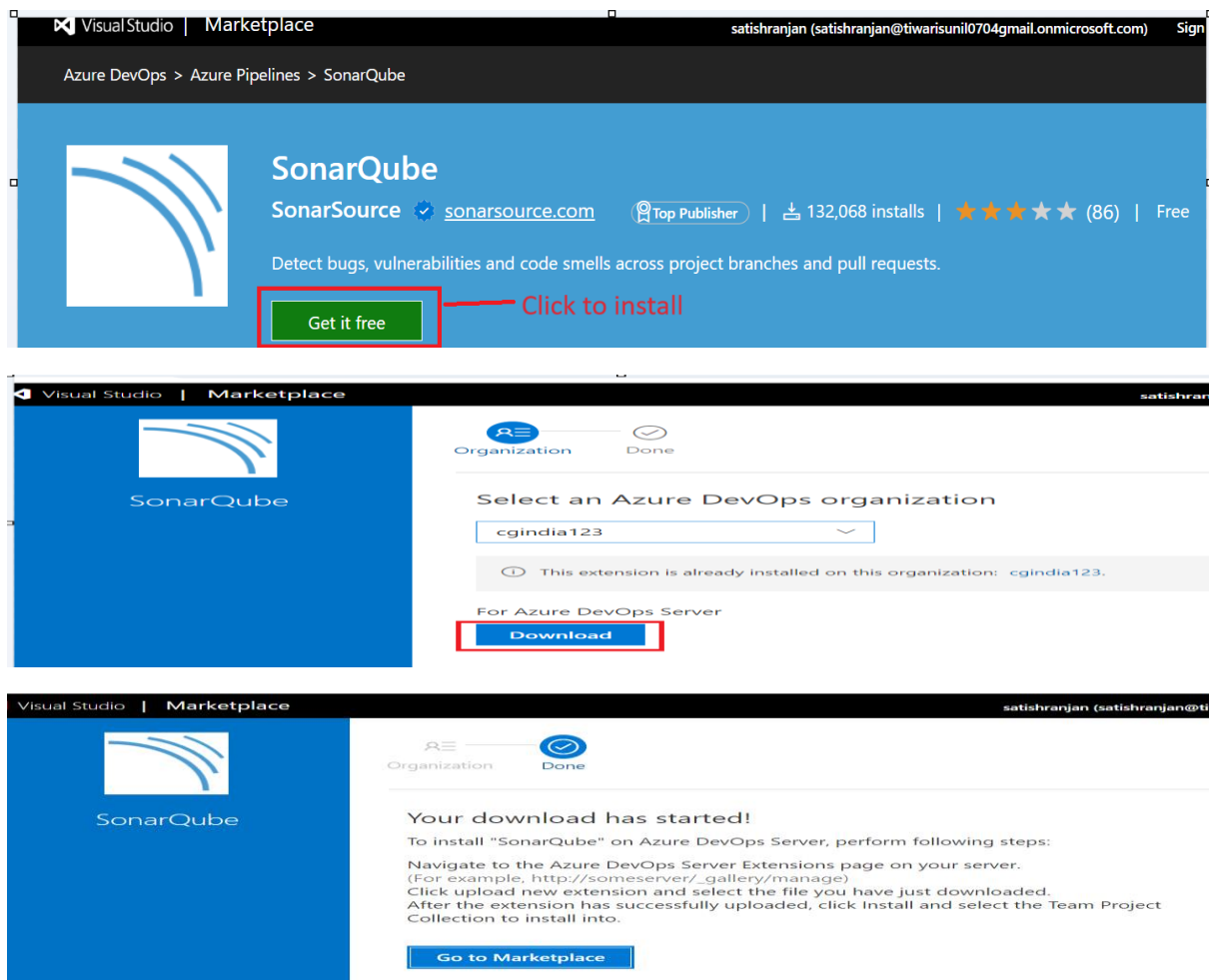
Use this for creating service principal in Azure DevOps CICD pipeline



Step6: Login to dev.azure.com portal, go to our project and then clicked on marketplace to search SonarQube.

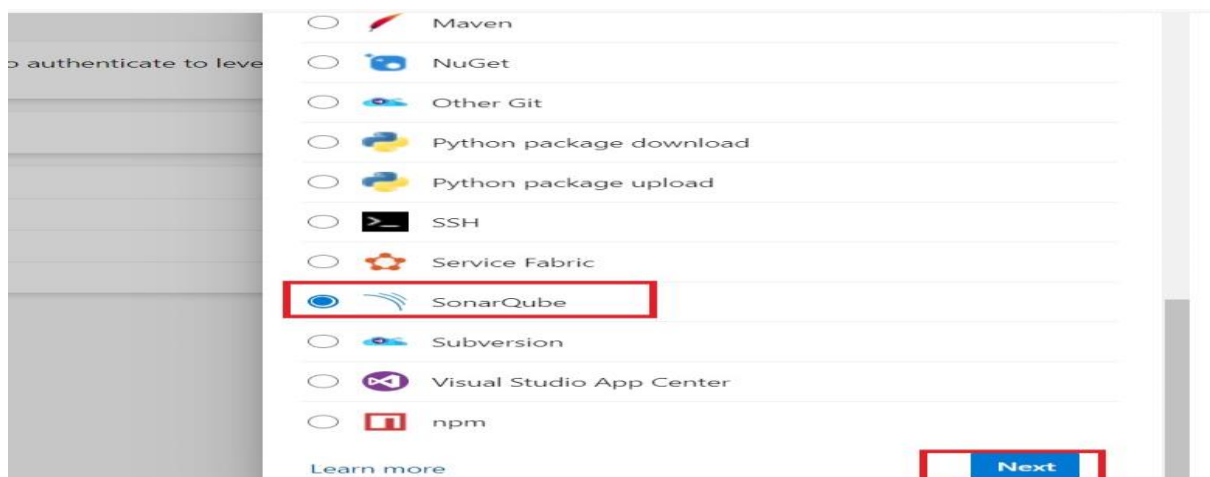
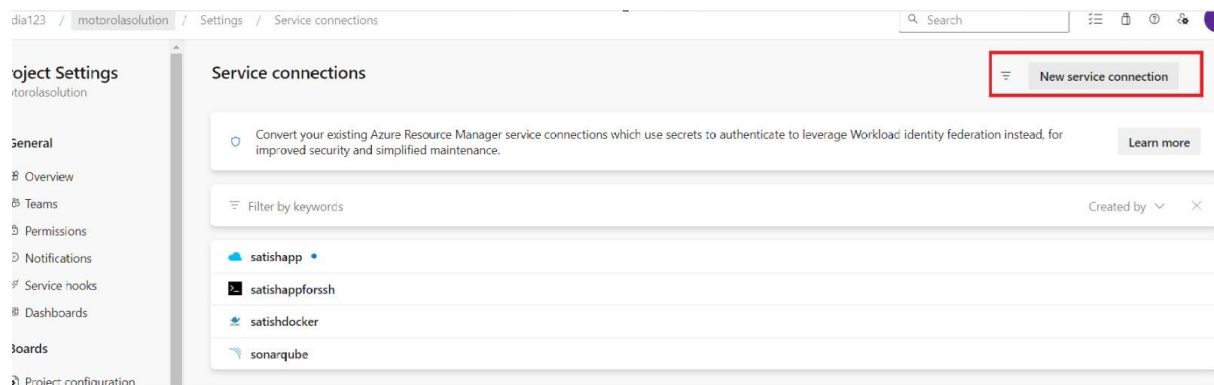
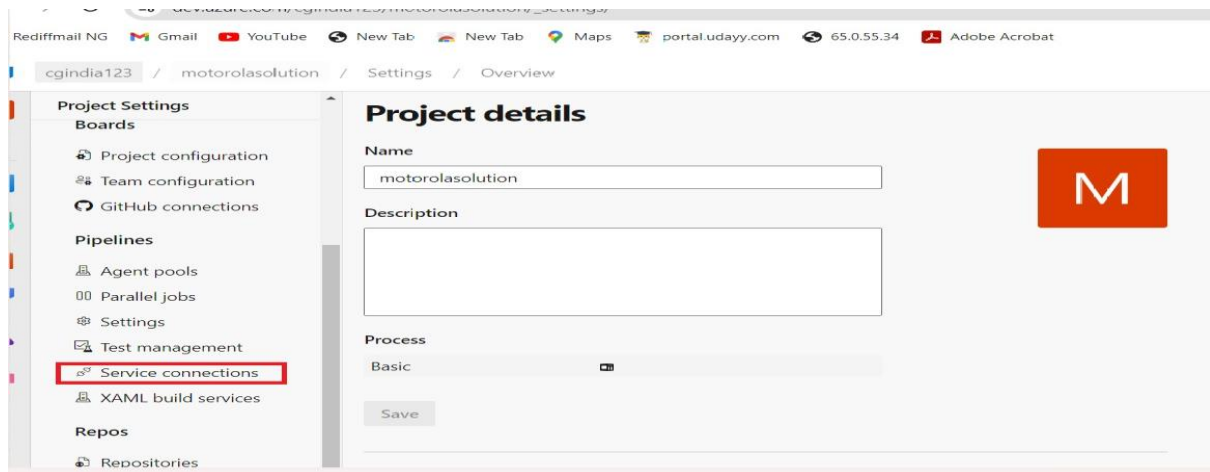


Step7: Install SonarQube in CICD Pipeline.



Step8: Now create service principle for connectivity between SonarQube to our azure DevOps.

Go to project setting → click on service connection



New SonarQube service connection

Server Url

`http://192.168.1.38:9000`

URL for the SonarQube Server to connect to.

Authentication

Token

.....

Authentication token generated through SonarQube (go to My Account > Security > Generate Tokens)

Details

Service connection name

`ranjanapp`

Description (optional)

Security

☒ Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)


[Back](#) [Save](#)

← sonarqube [Edit](#)


Overview Usage history Approvals and checks

Details

Service connection type

 SonarQube using basic authentication

Creator

 satishranjan
satishranjan@tiwarisunil0704gmail.onmicrosoft.com

Step9: Now Implement this SonarQube in Application pipeline.

There are three step to implement SonarQube in application pipeline:

- Prepare analysis Configuration
- Run Code Analysis
- Publish Quality Gate Result

```

12  # sonarqube - Agents name - square3 or yanagene
13  stages:
14  - stage: codescan
15  - displayName: sonarqube-scan-code
16  - job:
17  - job:
18  - steps:
19  - task: SonarQubePrepare@6
20  - inputs:
21  - SonarQube: 'sonarqube'
22  - scannerMode: 'CLI'
23  - configMode: 'manual'
24  - cliProjectKey: 'satishsonarkube'
25  - cliProjectName: 'satishsonarkube'
26  - cliSources: '.'
27  - settings:
28  - task: SonarQubeAnalyze@6
29  - inputs:
30  - jdkversion: 'JAVA_HOME_17_X64'
31  - settings:
32  - task: SonarQubePublish@6
33  - inputs:
34  - pollingTimeoutSec: '300'
35  - # stage: terraforminstallation

```

1 Prepare Analysis Configuration
Prepare SonarQube analysis configuration

2 Publish Quality Gate Result
Publish SonarQube's Quality Gate result on the A...

3 Run Code Analysis
Run scanner and upload the results to the SonarQ...

SonarQube for MSBuild - Begin Analysis
[DEPRECATED] Fetch the Quality Profile from Son...

SonarQube for MSBuild - End Analysis
[DEPRECATED] Finish the analysis and upload the ...

Step10: Run Self hosted Agent and then run the pipeline.

