Day 16. Sonarqube Setup And Practical.
(End-to End Deployment).

1) Will do entire setup using EC2, GitHub Actions (CI), Jenkins (CD) and Sonarqube

→ Launch an EC2 instance (t2-medium).
(20 GB of storage) (Terminate after the Practical).

a) Docker install.

b) Again configure / install Docker Compose

c) Vi docker-compose.yml file (instructions to launch Sonarqube).
(One is DB (Postgres), second one is Sonarqube).

d) Start Docker Engine / Daemon.

e) docker-compose up -d.
(Will create both the containers mentioned in the step (c)).

f) docker images.
docker volume ls
docker ps.

g) Copy IP Address and Login to Sonarqube.

h) Create project.
(myproj).
branch name.
All global setting.

(5) Pipeline (GitHub Actions).

j:> Set Sonar Token and URL (Imp).

Setting → Secret & Variables → Sonar Token → Sonar Host URL.

Generate Token

[All this in GitHub].

k:> Workflow file. (yml) file.

. Need to Trigger the Pipeline.

Change in Repo, then will be automatically triggered.

l:> Check Actions for workflow pipeline and Sonarqube as well.

Quality Gate tell the signal whether entire Scan was successfully done or not.

⇒ Added some threshold values.

⇒ Can create custom Quality Gate as well. As per our company rules can create rules to make the build pass / failed.

⇒ By Default there is build-in Quality Gate.

(eg) gfg qualitygate.

⇒ Have to make a new commit in order to use the New Quality Gate.

Quality Profiles How Sonarqube Knows about the Coding languages, this is because of rules, And they have around 321 overall rules.

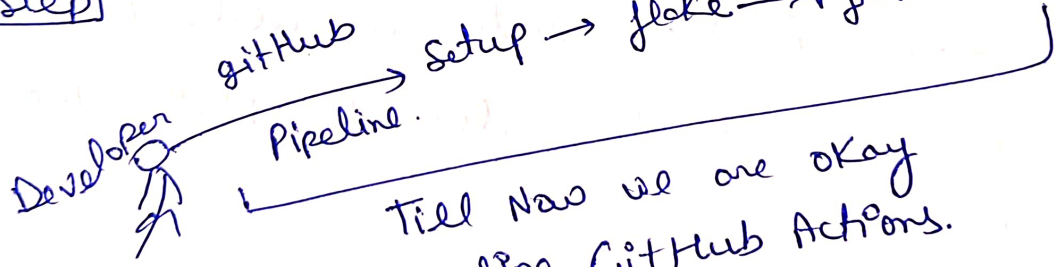
Python ⇒ validate 251 rules.
And for Docker ⇒ 40 rules.

→ Can create our own Quality Profiles.

Which language → Quality Profile Name → for Docker

* Select rules out of 44 → Change the Priority → Activate.
* Just validate new rules from Custom Profile, attach it to the project where we need to Validate this. (Not Recommended)

7th Step



Till Now we are okay using GitHub Actions.

Create a Dockerfile, Build Image from it

↓

Already Have this (Upload this to Docker Hub).

(11). In workflow file create 3rd Job to Build Docker Image.

Actions → Right side MarketPlace (Docker build search this)
On Git Hub

↓
Docker setup.

(Search in internet Docker build push Action)

↳ . login to Docker Hub (as push the image).
. action used to Build / Push Docker image.

(Secrets. DOCKK)

⇓

Go Back to GitHub → Setting → Add one more for DockHub.

| Password | Token |

Q:> Now once Docker Image is pushed to Docker Hub.

Till now we have done CI (Continuous Integration Pipeline).

GitHub Action can be used for both CI & CD

P:> . Now will use Jenkins for CD part.

. Go to EC2 instance, Configure Here Jenkins.

↳ Steps we already know how to configure.

Q:> launch one more EC2 instance (This will be Worker Node for Jenkins (15 GB).

⇓

Don't have do do this in core of GitHub Actions.

↳ Configure this as Worker Node.

4> Create a new (Worker) Node.

Name ⇒ gfg-Python.

→ Directory Name ⇒ gfg (Worker EC2).

→ Labels: gfg-Python

→ Save (To make it Active install the jdk javafile).
do make Worker node Active.
(Jdk gets Installed). (Agent is connected)

5> Create a job

→ gfg-Pipeline.
(Select Pipeline option).

→ Pipeline Script from SCM.
⇓
Git code location.
⇓
main (Branch Name).
⇓
Script Path. (Save) and not Triggered.
(Will Have the Info about CD).

```
pipeline
  {
  agent:
      {
       label 'gfgpython'

  }
   stages { stage ('Deployment')
  }.          steps {  sh 'docker pull — — !
                    } sh 'docker rm -f gfgwebos'
                      sh 'docker run -d -- name gfgwebos
                                -p 80:80 — 9mage N
```

4th step Manually click on Build Now in Jenkins after the Build is complete from GitHub Actions.

* Install Docker in Worker Node *

Need to Install git also in Worker Node
. (To need Jenkins file and it will get from repo).

Note If I made changes in Jenkins file, then no need to build other jobs, for that we can add conditions as well.

Standard Process.

1. Create a feature branch.
   ↳ if changes made in this no build/CI was triggered as in workflow file triggered is added on main branch.

   ↳ Create a PR from/to the Main branch.
   Imp

   ↳ Code testing should be done before pushing to main branch.

   ↳ Test cases should be validated also. Change in the feature branch test app.py file

```
Push:
    branches: ["main"]
```
] commented as not required.

⇒ . Jenkins Manually Triggered the job.

General → Triggers → Poll SCM (every 1 min (wastage of Resources)).

(Source code Management).

. Go to GitHub.

* * * * * (every 1 minute).

GitHub hook trigger. (who will be the first person to know).
(select this) → Save.

. GitHub → push event → Triggered Jenkins.

↳ GitHub → Setting → Webhooks

Allow external services to be notified when events happed.

↳ Add webhook (Jenkins URL).

IP_address-port/github-webhook/

↳ Select event

(Let me select individual events).

↳ (Just the push event).

↳ GitHub now integrated with Jenkins.

↳ End to End Automated Pipeline.