day-7 (Sonar, Pipeline as a code, email, security)
Docker introduction

* Sonar Qube → Provide very good Quality gate than pmd & checkstyle.

1. install Sonar Qube
2. install Sonar plugin jenkins
3. Sonar Scanner - on jenkins (sends the data to Sonar) - this is an Agent.
4. Configure Sonar Config
5. Setup job to send

[Lab1 ... 1 Sonar Qube-setup.txt)

→ it doesn't run as root user then so you can create another user.

- Install Sonar-Qube-runner.

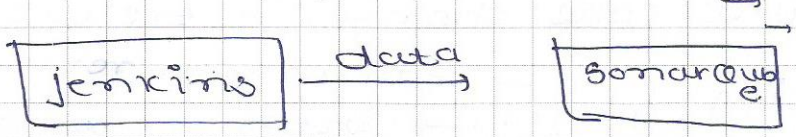
> sudo su devops
password ...

> 10pt1 ... sonarqube, sonar-runner-2.4

sonarqube .. 1 bin 1 linux-86- > ./sonar.sh start.

* 192.168.0.109:9000
  - admin
  - admin

Quality Gate [Passed]



- manage jenkins → configure system (third party tool) another server soft

* Sonar Qube Server
  - name: Sonar
  - URL: 192.168.0.109:9000
  - token from.

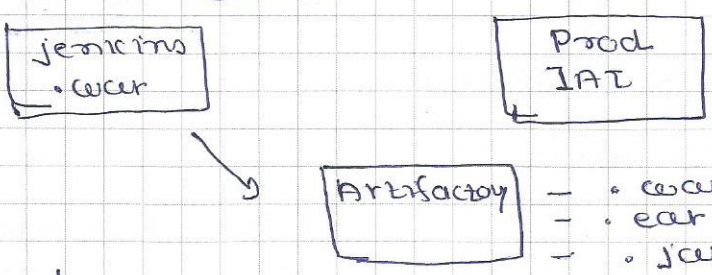
Global tool config. Sonar Qube Scanner (Agent)
- location of the software
10pt1 Sonar-runner-2.4

job → configure → Build → maven → Goal: compile
Execute SonarQube scanner.
sonar: sonar

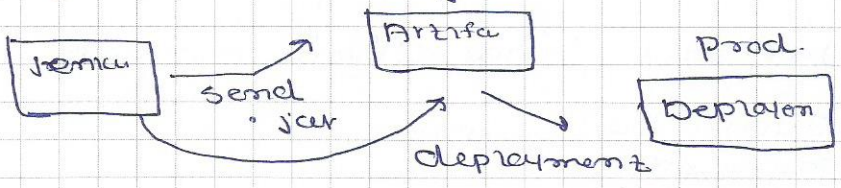
> Create Quality Gate: (set as a default)
↳ Add conditions – New Vulnerabi: W E
is greater than 2 ↳

> Go Admin: Quality profile

compile send data to SonarQube. Server.



* Nexus / jFrog setup. (docker container)
→ plugin : 'Artifactory'



* Pipeline as a Code.
(Automatically - compile, build, Sonar)
Jenkins file (as pipeline code)
Jenkinsfile → [Pipeline plugin]
→ support create. ↑ github inside (part of source-code)
Source code change. – execute jenkins file
→ compile, code-review.

Node: ↳ Stages ↳ Steps ([redacted],

```
pipeline {
    Agent any (slave name)
    stages {
        stage ('Build') {
            steps {
            }
        }
    }
}
```

← declarative pipeline

→ Pipeline → ([redacted] (groovy scripts)

```
stage ('compile') {
    steps { echo 'compiling'
            git url: '__'
            sh script: __
    }
}
('package') {
    steps {
    }
}
```

* Pipeline Example
  ↳ Pipline ((Pipeline plugin)
    ↳ Build Trigger (poll SCM..)
    ↳ Pipeline
      Pipeline Script
      Script 1.
      SCM : git
        ↳ Repository

Script path : Jenkinsfile-linux
Lightweight checkout.

- Stage View

Declarative
Checkout SCM

→ Pipeline Syntax: generate Script automatically.
  ↳ Script (Snippet generator)
  ↙ triggers ↘
  generate Script → (learn groovy Scripting)

* Email Setting
  ↳ manage jenkins → config System
    ↳ Extended E-mail config.

SMTP Server: smtp.gmail.com
Suffix: leadldevops

Jobs → confige → Post-build → Email Notification
Attach Build log : Attach build log.
trigger : Before Build, Fixed... Always.
Build Now. [Email Formate]
↙
Job: Build #3 : Success
↳ Attached log.

* Security: → Admin - Admin
  ↳ Authentication : [who are you?]
  ↳ Authorization : [what you are allowed to do?]

→ Add more User → manage jenkins
  ↳ manage Users
    ↳ create User

* Authentication + Authorization (Manage jenkins) (4)
(4)

↳ Configure Global Security

↳ Jenkins' own user database (user database

↳ LDAP

(right weight Accept pro.)

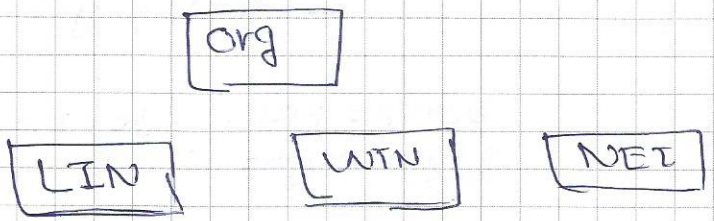
→ user data storage.

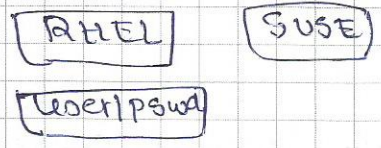
↓

third party Server

(data in database

LDAP Registory





↳ WIFI
↳ Window
↳ LINUX
↳ NET

} LDAP password

↳ Authorization

- matrix based Security

↳ Add user → Kalyan. (what permission)

Kalyan ☐ Read ☐ Read ☐ Read (view)

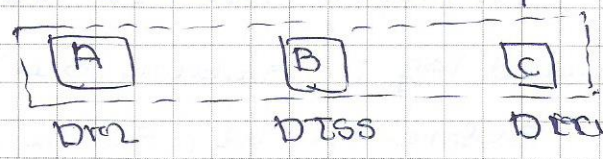
↳ Run the job

↳ jenkins: → Run & Edit the jobs

↳ LDAP - group based Authorization (Enterprise level)

↳ Realm - Registory for user storage.

* Projects based Security



(certain team can access certain Project)

↳ Compile - job

↳ (Add your own User)