**Day 7 Github**

* Maven
* Version control ( git, GitHub)
* Local repo, remote repo

VERSION CONTROL

* A version control system is a repository of assets. Often the files for the source code of computer programs, with manages access control.
* Tracks:
  + Who made changes
  + What the difference is (previous version)
  + Any metadata provided by the author as part of the change (check-in comment)
* GIT:
  + Manage local repositories.
  + In distributed development.
  + Each developer has a local copy of the development history.
  + Changes can be pulled or pushes from repository to repository copies.
  + Servers can be replicated and distributed around the world if necessary.

(2)

git————> gitlab

(1) / \

/ \

sonar jenkins

(3) (4)

1. Git
2. Gitlab:
   * Client is git
   * Server is gitlab
3. Sonar - code review, analysis, report, statistics
4. Jenkins - for continuous integration and deployment

- internal testing

- creates a url (build) (installed phase)

- with every update/ patch, new build are made and sent to the clients

Github life cycle:

* Workspace (system hard disk)
* Staging area
* Local repository
* Remote repository (server)

<https://github.com/pushpankq/Git-Commands->

GIT IMP COMMANDS

* Git init -> to create empty local repository
* Git add . -> all files -> file added to staging area
* Git add file\_name
* Fit status -> check status of working directory
* Git help
* Git commit -m “commit Message”

GIT BRANCHING

* Merge new branch to master branch
* First checkout to master branch
  + Git checkout master
* Git merge branchName
* Git push -u origin master

1. Create file
2. Git init ( to initialise the local directory)
3. Git add ( add the files to the local directory)
4. Git commit -m “comment” ( commit the files to the local repo)
5. Create the repo in github
6. Copy the url of the created remote repo
7. Git remote add <remote\_repo\_name> <url> ( connect the local repo to remote repo)
   1. To verify use ( git remote -v) command
8. Git push -u <remote\_repo> <local\_repo>

Git clone <url\_of\_repo\_to\_be\_cloned>

**Branching:**

1. Go to the local repo
2. To create branch ( git branch <branch\_name>
3. Switch to a branch ( git checkout <branch\_name>)
   1. We are in the new branch now
   2. Make change, add files, etc
   3. Now add file to the local repo ( git add <file\_name>)
   4. Commit
   5. Switch to the main branch ( git checkout <main OR main\_branch\_name>)
4. To merge the branches:
   1. Git merge <branch\_name\_which\_has\_to\_be\_merged>
5. Now push the local repo(main branch) to remote repo:
   1. Git push -u <remote\_repo\_name> <local\_repo/main\_branch\_name>

Files added or deleted have to be committed in the main branch before pushing

**MAVEN**

* Build automation: build automation does everything mentioned below except the coding part. Coding has to be done by the developer using business logic
  + Developer job:
    - Code
    - Compile
    - Test ( test cases have to be made and tested against the program coded )
    - Package
    - Deploy
  + War file (web archive)
  + Jar file (java archive)
* Build life cycle:

1. Validate

2. Compile (conversion to .class files)

3. Test (test the logic of the .class files created). (Create a test class containing test cases comprising all the possibilities) (Junit, mokito - unit testing framework)

4. Package (take the compiled code and package ii in its distributable format. (JAP)

5. Verify ( integration testing)

6. Install ( created jar file, etc have to be added to the local repository) ( install packages into the local repo)

7. Deploy

* Into
* Environment setup
* Sample maven app
* Repositories

Build tools : apache ant, apache maven, gradle, sbt