**Software Configuration Management (SCM) Tools - GitHub**

Is to practice discipline of tracking and controlling changes in software, part of the broader field of configuration management.

**Functions of SCM** Includes:

* Version Control(VCS)
  + Git, Mercurial, SVN (Sub version)
  + Atlassian stash, IBM ClearCase
* Change management
  + Review/approval, etc process through tools like Jira
* Build management
  + CI/CD pipelines with static code, code completion, coverage tools integrated and automated. Ex: Jenkins, GitHub Actions, Maven, Gradle, Make, etc
  + Static Code analyser – SonarQube, JSLint for JS
  + WhiteSource(Mend) for OpenSource Security, License mgmt., etc
  + Blackduck for Governance and Compliance
  + JaCoCo – for Java C
* Release
  + QA/UAT, sign off, etc., integrated in a planned release
* Configuration
  + Blue/green deployments
  + Env specific config files like config.yaml
  + Docker config
  + Dependency config
* Audit and Reporting
  + SecCheck, Dashboards, Logs, Reports on code/GDPR compliance, etc

**GitHub:**

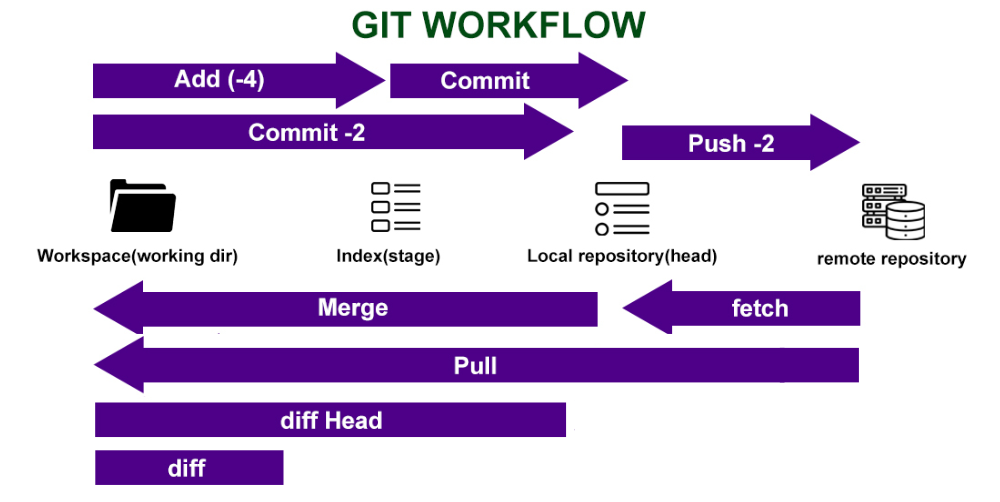
* built on top of Git, a free and open source VCS
* Either we can use browser based tool (cockpit, web interface/dashboard) or installed tools like GitHub Desktop

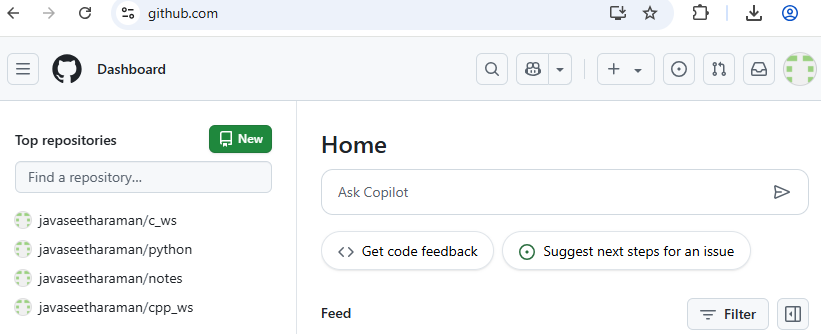
**Basic Git workflow:**

1. **Clone** a repository (copy it locally).
2. **Make changes** to files.
3. **Stage** changes (git add).
4. **Commit** changes with a message (git commit).
5. **Push** commits to a remote repository (git push).
6. **Pull** updates others have made (git pull).
7. **.gitignore** to exclude files

**Basic GitHub Flow:**

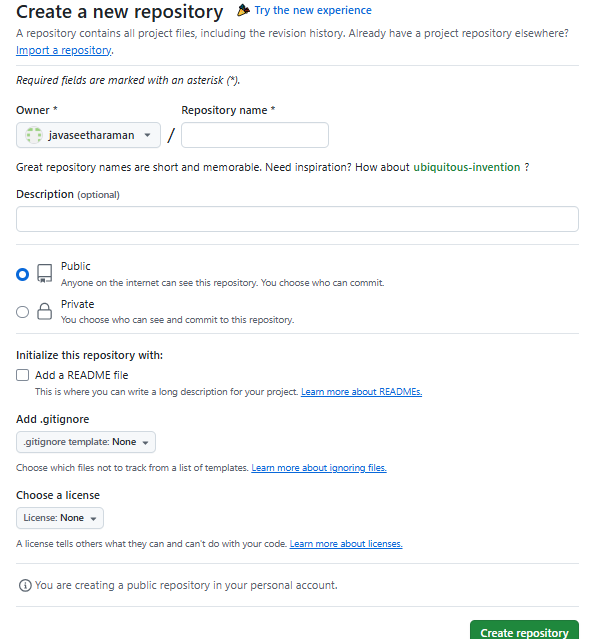
* Register with mailId in <https://github.com/>
* Login
* Repo actions



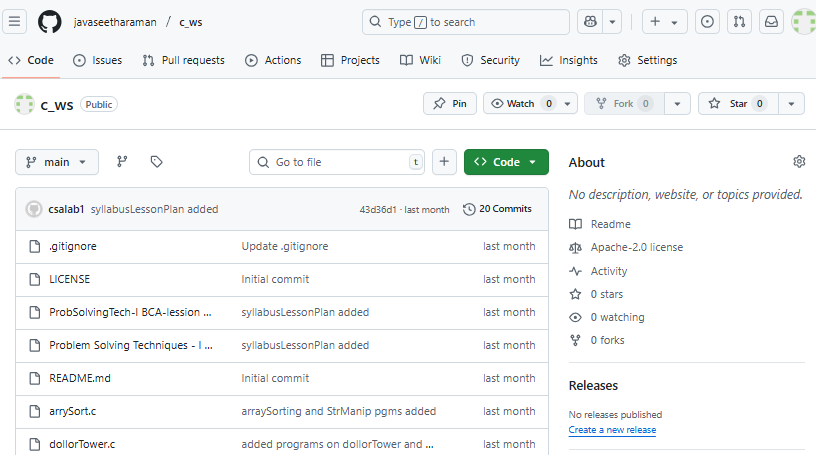


* Search repositories, issues, users.
* Create Repo/Issues
* Look my issues
* PRs I created, for my review
* Profile management
* Licensed like none(private), BSD, MIT, Apache, GNU, Mozilla

**Create New Repo:**



**Repository Summary View:**



**Steps to add Collaborators to our Repo:**

1. Go to your repository on GitHub.
2. Click on the "Settings" tab (⚙️) at the top of the repo.
3. In the left sidebar, click "Collaborators and teams" (or "Manage access").
4. Click “Invite a collaborator”.
5. Enter the person’s GitHub username or email.
6. Select their role (default: Read-only for viewers).
7. Click Add or Send Invitation.

* Once accepted, they can view (and optionally contribute to) the repository based on their permission level.
* Without adding, anyone can view our public repo
* Reviewers are allowed only for PR on branch merge

**Frequent Commands cheat sheet:**

1. Setup/Init

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

1. Start New/Existing Project

git init - Initialize a new Git repository in the current folder.

git clone <https://github.com/username/repo.git> - Clone a remote GitHub repo to your local machine.

1. git status - See current changes and staged files.
2. Stage files to include in the next commit.

git add filename

git add . # Add all changes

1. Commit staged changes.

git commit -m "Your commit message"

1. Push your commits to the remote GitHub repository.

git push origin branch-name

1. Fetch and merge changes from the remote repo.

git pull origin branch-name

1. Download changes without merging.

git fetch

**Branching:**

1. List all branches

git branch

1. Create new Branch

git branch new-branch

1. Switch to a different branch.

git checkout new-branch

1. Create and switch to a new branch.

git checkout -b feature/new-feature

1. Merge another branch into your current branch.

git merge branch-name

git rm, git log, git diff,