```
=========
Git Hub
=========

=> It is version control software

-> Git Hub is a platform which is used to store project related files/code.

-> In git hub, we can create source code repository to store project code.

-> All the developers can connect to project repository to store all the source code
   (Code Integration will become easy)

-> Code will be available at one central location (easy access)

-> Git Hub repository will monitor all code changes

                  - who modified
                  - when modified
            - what modified
                  - why modified

===================
Environment Setup
===================

1) Create account in www.github.com (free of cost)

2) Download & Install git client software

                URL : https://git-scm.com/downloads

3) Open git bash tool and configure your name and email using below commands.

                $ git config --global user.name "Ashok IT"

                $ git config --global user.email "ashokitschool@gmail.com"

Note: Configuring name and email is just one time process.

================================
What is Git Hub Repository ?
================================

=> Repository is a place where we can store project source code / files.

=> For every project one repository will be created

=> We can create 2 types of repositories in git hub

                1) Public Repo (anybody can see & you choose who can commit)

                2) Private Repo (you choose who can see & commit)

## Project git repo url : https://github.com/ashokitschool/sbi_mobile_banking.git

=> Project team members will connect with git repository using its URL.


=========================================
What is the diff between git and git hub
=========================================
```

=> git is a client s/w which is used to communicate with git hub repositories.

==========
Git Architecture
==========

1) Working tree

2) Staging area

3) Local Repo

4) Central Repo (remote)

=> Working tree represents the place from where we want to perform git operations. Generally project directory will be considerd as working tree.

Note: To represent/initialize working tree we will execute 'git init' command.

=> Staging area represents which files are eligible for commit. To add files to staging area we will use 'git add' command.

```
git add <file-name>

git add *.txt

git add *.java

git add .
```

=> Local Repo represents the commits we have done using 'git commit' command.

```
git commit -m 'commit-msg'
```

Note: With 'git commit' only staged files will be commited to local repo.

=> Central Repo will be available in git hub. All the team members changes will be integrated in central repo.

Note: To send local commits to central repo we will use 'git push' command.

==========
Git commands
==========

git config : To configure name & email

git init : To initialize working tree

git status : To check working tree status (staged & un-staged)

git add : To add files to staging area

git commit : To commit files from staging to local repo

git push : To send files from local repo to remote repo

git restore :

1) To discard changes when the file is in unstaged state

2) To unstage the file when it is added to staging area

git log : To see commit history

git pull : To take latest changes from central repo to local repo

git clone : To clone remote repo to our local machine.

```
git clone <repo-url>
```

git rm : To remove files (rm + commit + push)

==========
What is .gitignore file ?
==========

=> .gitignore is used to exclude files & folders from our commits.

Ex-1: In maven project, we should n't commit "target folder" to git repository hence we can give this info to git using .gitignore file.

Ex-2 : In Angular app, we should n't commit "node_modules" folder.

==========
Git Branches
==========

=> In a project multiple development teams will work paralelly

1) Bug fixing team
2) Enhancements team
3) change request (CR) team
4) Research & Development team
6) Prod support team

=> When multiple teams work on single repository then it will become very difficult to manage the code.

Note: To overcome above problem, we will use "git branches" concept.

=> Branches are used to maintain multiple code bases in the single repository.

Note: We can create any no.of branches in single repository.

=> If we have branches in git repo then multiple teams can work paralelly without effecting other teams code.

```
# clone git repo (default main branch)
git clone https://github.com/ashokitschool/springboot_register_login_security.git

# clone git repo develop branch only
git clone -b develop <repo-url>

# swith branch
git checkout <branch-name>
```

==========
What is pull request (PR)
==========

=> It is used to merge changes from one branch to another branch.

PR

Ex : develop ----------> main

==========
what is git stash
==========

11:00 AM => Manager assigned task "AB-102"

2:00 PM => Completed half of the task (fully not completed)

3:00 PM => Manager called and informed "AB-103" is high priority. First complete 103 and then work on 102.

Note: In this situation we can't delete 102 changes from working tree because 3 hours we spent and we can't commit also because changes are not tested.

git stash : save working tree changes in temp area and make working tree clean.

git stash apply : Get stashed changes back to working tree.

==========
What is git fetch ?
==========

git pull : it will download latest changes from central repo to local repo and merge with working tree also.

Note: with pull command there is a chance of getting conflicts.

git fetch : It will download latest changes from central repo to local repo and it will not merge with working tree.

Note: To merge changes from local repo to working tree we need to execute 'git merge' command.

```
git pull = git fetch + git merge
```

==========
What is git revert
==========

=> Revert is used to remove our changes from central repository based on commit id.

```
git revert <commit-id>
```

Note: When we execute 'revert' command it will open editor for msg. Save and close it using "esc + :wq".

=> After revert command we need to execute "git push" to publish revert operation to central repo.

==========
What is git conflict
==========

=> When we execute "git pull" command there is a chance of getting git conflicts.

Note: If two team members working on same file and same line then we will get conflicts.

=> When we merge one branch changes with another branch then there is a chance of getting git conflicts.

Note: When we get conflicts then it is our responsibility to resolve those conflicts and commit to github without conflicts.

==========
What is git fork
==========

=> It allows you to create a personal copy of someone else's repository

Note: In general, we will use this forking concept for experiments on open source projects.

==========
How to remove git local commits
==========

```
# remove latest local repo commit and keep changes in working tree
git reset --soft HEAD~1

# remove latest local repo commit and discard changes from working tree
git reset --hard HEAD~1

# remove last 3 local commits
git reset --soft HEAD~3
```

==========
What is cherry-pick
==========

In Git, cherry-pick is a command used to apply the changes from a specific commit from one branch to another.

Ex:

=> In develop branch we have done 2 commits. If we create pull request to merge develop branch changes to main branch then it will merge all commits at once (full merge)

=> If we don't to perform full merge, we want to merge only particular commit from develop branch to main then use 'git cherry-pick' option

Ex : git cherry-pick <commit-id>

==========
What is git tag
==========

=> Tags in Git provide a convenient way to mark specific commits, usually for releases or important milestone.

```
# show all tags
git tag

# create a tag
git tag -a <tag-name> -m '<tag-msg>'

# push tags to central repo
git push origin <tag-name>

# push all tags to central repo
git push --tags

# delete tag
git tag -d <tag-name>

# We can checkout particular tag using its name
git checkout <tag-name>
```

Note: To comeout from tag stat we can use below command

```
git switch -
```

==========
git merge vs git rebase
==========

=> These commands are used to merge one branch commits to another branch

Ex :

=> We have committed changes to develop branch now we want to merge develop branch changes to main branch.

```
git switch main

git merge develop ( or ) git rebase develop
```

merge : Creates new commit and combines and preserves commit history.

rebase : Reapplies commits on top of our branch. It will not preserve commit history.

==========
Real-Time work flow
==========

=> As a devops engineer we are responsible to manage source code repositories in the project.

=> Development team will Create JIRA story for git repo creation and will assign to devops team with development manager approval.

=> Once request got approved, we should create git repo and we need to share git repo url with developmen team.

=> Development team will integrate code in git repo and they will create multiple branches.

Note: DevOps team will decide branching strategy. That means development code should be there in which branch and which branch code will be deployed to production.

=> DevOps team will use user permissions for git repo (RBAC)

- Read Permissions
- Read & Write Permissions

Note: Before production deployment we need to go for Code Freeze
(15 days before or 1 month before)