=========
Git Hub
=========

-> Git Hub is a cloud platform which is used to store all the developers project source code at one place using repositories concept.

=> To communicate with git hub repositories we will use 'git' as client software.

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

-> All the project team members can connect to git hub repository for code integration.

-> Using Git Hub repository we can monitor/track 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 in our system

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

3) Open Git bash and configure your name and your email using below commands

```
git config --global user.name "your-name"

git config --global user.email "your-email"
```

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

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

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

=> For every project one git hub 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 & who can commit)

@@ Project Git Repo URL : https://github.com/ashokitschool/01-SBI-Loans-App.git

Note: In real-time DevOps team is responsible to create git repositories required for the project.

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

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

1) Working Tree

2) Staging Area

3) Local Repo

4) Central Repo

Working Tree :: This is your current workspace where files are created, edited, and deleted.

Staging Area :: A middle layer between your working directory and the local repository. It Represents which files eligible for commit.

Local Repo :: The place where your commits are permanently stored on your local machine. For each commit unique commit id will be generated.

Central Repo :: Hosted on www.GitHub.com and used for collaboration among multiple developers.

Working Tree â†’ (git add) â†’ Staging Area â†’ (git commit) â†’ Local Repo â†’ (git push) â†’ Remote Repo

==========
Git Bash Commands
==========

git clone : to download central repo to local system

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

git status : display working tree status (staged and un-staged)

```
$ git status
```

git add : Add files to staging area

```
$ git add <file-name>

$ git add .
```

git commit : Commit staged files to local repo

```
$ git commit -m <msg>
```

git push : publish local commits to central repo

git rm : To remove file

```
$ git rm <file-name>
```

Note: After 'git rm' execution we need to execute commit and push to delete file from central repo.

git log : to display commit history

git restore :: We can discard file changes + We can unstage the file

git pull : download latest changes from central repo to our working tree.

==========
Q) how to remove git local commits
==========

=> Using 'git reset' command we can remove our local commits

=> Git Reset we can do in 2 ways

a) soft reset

b) hard reset

=> soft reset means only local commit will be deleted and file changes will be there in staging area.

```
$ git reset --soft <commit-id>
```

=> hard reset means local commit will be deleted and file changes also will be deleted from working tree.

```
$ git reset --hard <commit-id>
```

==========
Q) how to revert git central repo commits
==========

=> Using 'git revert' command we can revert code changes from central repo.

Ex : git revert <commit-id>

Note: After git revert execution we need execute git push.

==========
Q) What is git stash
==========

you are watching a movie... you are in middle of the movie... still watching..

mother told to bring one milk packet from shop.. immediatley (high priority)..

Then we will pause the movie and we will bring milk packet from the shop then we will continue watching movie..

9:00 AM : JIRA-101 story assigned to me . work started... few changes completed... i m in middle of that story by *8:00 PM i will complete it....

@11:00 AM: JIRA-102 assigned and it is very crictical now... high priority.. first i should complete JIRA-102 story and i need to push only 102 task changes to central repo.

=> To work on JIRA-102 story changes, first we need to stash 101 story related changes.

```
$ git stash
```

Note: When we use git stash, it will save uncommitted chanages to Working tree as backup.

=> After pushing JIRA-102 changes we can get JIRA-101 changes back to working tree using 'git stash apply' command.

```
$ git stash list

$ git stash show stash@{0}

$ git stash apply

$ git stash apply stash@{1}

$ git stash apply stash@{2}
```

```
                $ git stash drop stash@{1}
```

Definition : git stash is used to temporarily save (or "stash") your uncommitted changes in a working directory so you can work on something else without committing those changes.

```
============================
Q) What is merge conflict ?
============================
```

git pull :: Download latest changes from central repo to working tree directley.

Note : When we use git pull there is a chance of getting merge conflicts.

A Git conflict happens when Git canâ€™t automatically merge changes in working tree.

Scenario : If two developers modifyting same file and same line of code then we will face this issue.

Note: When conflicts occurs we must manually edit that file and resolve conflict issues and push changes to central repo.

```
=========================
Q) Git pull vs git fetch
=========================
```

git pull : Download latest changes from central repo to working tree directley.

git fetch : download from central repo to local repo only.

note : To merge changes from local repo to working tree we can use 'git merge' command.

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

```
================
What is Git Gui
================
```

=> It is a desktop tool to perform git operations

=> Execute below command in working tree to open git gui

```
                $ git gui
```

```
==============
Git Branches
==============
```

20 developers available in the team

on going development team => 5 ppl

research & dev team => 5 ppl

bug fixing team => 5 ppl

prod support => 5 ppl

=> When multiple teams working on same repo then project delivery will become difficult.

=> As all team members are integrating code in single repo we can't differentiate team wise work to deliver particular team work to client.

Ex: We need to delivery only R & D team code changes to client today.

=> To make our project delivery process simple, we need to maintain multiple branches in git hub repo.

Note: For every team one git branch is recommended.

=> By using branches, multiple teams can work paralelly.

Note: Our DevOps team will decide branching strategy (which team should use which branch).

main branch (default)

develop

feature

research

release

=========
Lab Task
=========

1) Go to git hub repository and create develop branch from main branch

2) Clone git repo (by default main branch will come)

```
$ git clone <url>
```

3) Switch from main to develop

```
$ git checkout develop
```

4) create a file and push to develop branch

5) Create pull request and merge develop branch changes to main branch.

========================
What is pull request ?
========================

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

ex: merge develop branch changes to main branch

==

```
# clone main branch by defult
git clone https://github.com/ashokitschool/01-SBI-Loans-App.git


# clone develop branch
git clone -b develop https://github.com/ashokitschool/01-SBI-Loans-App.git
```

==

==========================

Q) what is git cherry-pick
==========

=> When we use 'git merge' command by default all commits will merge from one branch to another branch.

=> If we want to merge specific commit from one branch to another branch then we can use 'git cherry-pick'.

Ex :

$ git cherry-pick <commit-id-1> <commit-id-2> <commit-id-3>

$ git cherry-pick <start_commit>^..<end_commit>

Note: If we use pull-request it will merge all commits from one branch to another branch. You cannot directly merge specific commits using a GitHub Pull Request.

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

git merge :: Merges the content of one branch into another and creates a merge commit.

$ git checkout main
$ git merge develop

Note: This creates a new commit that joins the histories of main and develop.

1) Preserves complete commit history
2) Safe for shared branches
3) Shows when and how branches diverged

git rebase :: Reapplies commits from one branch on top of another branch â€” rewrites commit history.

$ git checkout main
$ git rebase develop

=> This moves the base of develop branch to the tip of main, reapplying the commits

1) Clean, linear commit history
2) Easier to understand with git log

==========
Q) What is git fork ?
==========

=> It is used to copy somebody's git repo into our git hub account.

Note: When we want to work with open source projects then we can use git forking option.

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

=> It is used to specify files and folder to skip from git operations.

ex:

```
.settings
.classpath
.project

target/
```

==========
How to push entire project to git hub ?
==========

Step-1 :: Create new github repository. It will show git commands like below

```
echo "# hello-app" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ashokitschool/hello-app.git
git push -u origin main
```

Note: Instead of adding only 'git add README.md' file to staging area we can add all files by using 'git add .'

Step-2 :: Execute above all commands in project directory then our project files will be pushed to github central repo.

==========

Note-1 : Git hub repos will be created by devops team.

Note-2 : DevOps team will manage users access for the repos (read & write access).

Note-3 : We can create task based branches based on our requirement.

Note-4 : Branching strategy will be defined by DevOps team.

Note-5 : Code Freeze will be there when we have production deployment.

==========