.==========
Source Code Repository Tools (or) Version Control Softwares
==========

-> Multiple developers will work for project development

-> Developers will be working from multiple locations

-> All developers code should be stored at one place (Code Integration Should Happen)

-> To integrate all the developers source code at one place we will use Source Code Repository Softwares

==========
Advantages with Source code repository sofwares
==========

1) All the developers can connect to repository server and can integrate the code

2) Code Integration will become easy

3) Repository server will provide monitored access

- Who
- When
- Why
- What

==========
Repository Tools
==========

SVN (outdated)

Git Hub

BitBucket

==========
Environment Setup to work with Git Hub
==========

1) Create Github account (www.github.com)

2) Download and install Git Client software (https://git-scm.com/downloads)

3) Once installation completed, right click on the mouse and verify git options display (If git options displaying our git client installation completed successfully)

==========

Working with GitHub
==========

-> Login into github account with your credentials

-> Create Repository in github

Note: Repository is used to store project source code. Every Project will have one repository

-> When we create a repository, unique URL will be generated with Repository Name (i.e Repo URL)

Ex: https://github.com/ashokitschool/hdfc_bank_app.git

-> All the developers will connect to repository using Repository URL

-> We can create 2 types of Repositories in Git Hub

1) public repository

2) private repository

-> Public Repository means everybody can access but we can choose who can modify our repository

-> Private Repository means we will choose who can access and who can modify

Repo URL : https://github.com/ashokitschool/01-devops-app.git

==========
Working with Git Bash
==========

-> Git Bash we can use as Git Client software to perform Git Operations

-> Download and install git client (https://git-scm.com/downloads)

-> Right Click on Mouse and choose "Open Git Bash Here"

git help : It will display frequently used git commands

git help <cmd-name> : It will open documentation for given command

==========
Configure Your Email and Name in Git Bash with Commands
==========

$ git config --global user.email "youremail@yourdomain.com"

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

Note: email and name we will configure only for first time.

$ git init : To initialize our folder as git working tree folder

$ git clone : To clone git repository to our machine from github.com

Syntax : $ git clone <project-repo-url>

$ git status : It will display staged , un-staged and un-tracked files

Syntax : $ git status

Staged Files : The files which are added for commit

Un-Staged Files : The files which are modified but not added for commit

Un-tracked files : Newly created files

Note: To commit a file(s), we should add to staging area first

$ git add : It is used to add file(s) to staging area

Syntax : $ git add <file-name>

$ git add .

$ git commit : It is used to commit staged files to git local repository

Syntax : $ git commit -m 'reason for commit'

$ git push : To push changes from git local repository to git central repository

Syntax : $ git push

$ git rm : To remove file(s) from repository

==========
Steps to push code to github central repository
==========

1) Create one public repository in git hub (take github repo url)

2) Clone github repository using 'git clone' command

$ git clone 'repo-url'

3) Navigate to repository folder

4) Create one file in repository folder

$ touch Demo.java

5) Check status of the file using 'git status' command

$ git status (It will display as untracked file)

6) Add file to staging area using 'git add' command

$ git add .

7) Commit file to git local repository

$ git commit -m 'commit-msg'

8) Push file from git local repository to git central repository using 'git push' command

$ git push

Note: If you are doing 'git push' for first time it will ask to enter your github account password.

Note: Git bash will ask our password only for first time. It will save our git credentials in Credential Manager in Windows machine.

-> Go to Credential Manager -> Windows Credentials -> Select Github -> We can modify and delete saved credentials from here
---

-> When we do git commit then it will geneate a commit-id with 40 characters length

-> From this commit-id it will display first 7 characters in git hub central repository

-> We can check commit history using 'git log' command

==========
Steps to commit Maven Project to Github Repository
==========

1) Create Maven Project

2) Create GitHub Repository

Note: After creating git repository, it will display set of commands to execute

3) Open gitbash from project folder and execute below commands

$ git init
$ git add .
$ git commit -m 'commit-msg'
$ git branch -M main
$ git remote add origin <repo-url>

$ git push -u origin master

When we are working on one task suddenly we may get some other priority task.

Usecase
++++++++
-> ager assigned task id : 101
-> I am working on that task (i am in middle of the task)
-> Managed told that stop the work for 101 and complete 102 on priority.
-> Once 102 is completed then resume your work on 101

-> When manager asked me to start 102 task, i have already done few changes for 101 (partially completed)

-> We can't push partial changes to repository because with our partial changes existing functionality may break.

-> We can't delete our changes because we have spent few hours of time to implement those changes

********** In this scenario we will go for 'git stash' option **********

=> Git stash is used to save working tree changes to temporary location and make working tree clean.

-> After priority work completed we can get stashed changes back using 'gitstash apply'

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

-> Branches are used to maintain seperate code bases for our project

-> In GiT repository we can create multiple branches

main
develop
qa
release
research

-> development team will integrate the code in 'develop' branch

-> bug-fixing team will integrate the code in 'qa' branch

-> R & D team will integrate the code in 'research' branch

-> In github we can create branches

-> To clone particular branch in git repo we will use below command

$ git clone -b <branch-name> <repo-url>

==========
Branch Merging
==========

=> The process of merging changes from one branch to another branch is called as Branch merging

=> We will use Pull Request for Branch Merging

==========
Steps to do branch merging
==========

1) Create feature branch from main branch

2) clone feature branch

3) create new file in feature branch then commit and push to central repo

4) Go to central repository then create pull request to merge feature branch changes to main branch

Note: Once feature branch changes are merged to main branch then we can delete feature branch (if required)

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

-> This .gitignore is used to configure the files or folders which we want to ignore from our commits

-> The files and folders which are not required to commit to central repository those things we can configure in .gitignore file

Ex: In maven project, 'target' folder will be available which is not required to commit to central repository. This we can configure in .gitignore file.

----------.gitignore----------
HELP.md
target/
!.mvn/wrapper/maven-wrapper.jar
!**/src/main/**/target/
!**/src/test/**/target/

STS
.apt_generated
.classpath
.factorypath

```
.project
.settings
.springBeans
.sts4-cache

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/
build/
!**/src/main/**/build/
!**/src/test/**/build/

### VS Code ###
.vscode/
```

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

=> These commands are used to merge changes from one branch to another branch

-> git merge will preserve commit history

-> git rebase will not preserve that rebase history

-> When we are working on particular sprint and we want to merge changes from one branch to another branch then we will use 'git merge' command

-> Once sprint-1 is delivered then we want to take latest code of sprint-1 to start sprint-2 development. In this scenario we don't need commit history so we will use 'git rebase' command.

==========
What is git pull command
==========

-> pull command is used to take latest changes from repository to local

-> When we want to make some changes to code, it is always recommended to take git pull first then start your changes.

Note: When we execute 'git pull' there is a chance of getting conflicts. We need to resolve the conflict and we should push the code without conflicts.

==========
git diff : It is used to compare the files (local file & repository file)

git squash : It is used to combine multiple commits into single commit

git bisect : It is used to identify the bad commit which introduced bug.

==========

==========
Git Hub Lab Task
==========
1) Create Maven Web Application
2) Add 'Spring-Core' dependency in project pom.xml file (www.mvnrepository.com)
3) Package maven project as war file using maven goal
4) Create Git repository in github / bitbucket (public repo)
5) Push maven project into github repo using gitbash
(target folder shouldn't be commited, add this in .gitignore file)
6) Make changes in pom.xml and push changes to github repo using git bash
7) Create 'feature' branch in git repo from main branch
8) Switch to feature branch from git bash
9) Make changes in 'feature' branch pom.xml file
10) Push changes to feature branch
11) Create pull request and merge 'feature' branch changes to 'main' branch

==========
Git Hub Class Summary
==========
1) What is Source Code repository
2) Why we need source code repository
3) What are the source code repositories available
4) What is git hub
5) What is git
6) What is Repository
7) Public Repository vs Private Repository
8) Cloning Repository
9) Staged vs Unstaged vs Untracked File
10) Adding Files to Stating Area
11) Unstaging the files from staging
12) Discarding local changes
13) What is working tree
14) What is Local Repostiory
15) What is Central / Remote Repository
16) Commit from working tree to local repo (git commit -m 'msg')
17) push from local repo to central repo (git push)
18) Taking latest code changes
19) push vs pull
20) What is conflict
21) How to resolve conflicts
22) What is branch in git hub
23) How to create branches
24) How to clone particular branch (git clone -b <branch-name> <repo-url>)

25) how to switch to particular branch (git checkout <branch-name>)
26) How to merge branches
27) What is pull request
28) git merge vs rebase
29) what is .gitignore
30) Working with Bitbucket

git init
git help
git config
git clone
git status
git add .
git add <file-name>
git restore
git commit
git push
git pull
git log
git rm
git revert
git branch
git checkout
git fetch
git merge
git rebase
git stash
git squash

=========

BitBucket

-> Version control software

-> Developed By Atlasian Company

-> BitBucket / GitHub internally uses git as version control