```
===========================
Version Control Softwares
===========================

=> Multiple Team members will be there for project development

=> Team members will be working from different locations


Problem-1 : How to integrate all the developers code at one place

Problem-2 : How monitor code changes (who/why/what/when)


=> We can get solution for above two problems using version control s/ws

=> We have several version control softwares in the market

        a) SVN (outdated)
        b) Git Hub
        c) BitBucket


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

=> Git Hub is a cloud platform which is used to maintain project source code repositories

=> We can use trail version of git hub for practice

Note: In companies we will use licenced version of git


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

=> Create account in www.github.com

=> Download & install git client software (www.git-scm.com)

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

=> For every project one git repository will be created in github.com

=> Repository is a place where we can store our project source code

=> Every Repository will have its own URL

=> We can connect with git hub repository using its URL.

                Repo URL : https://github.com/ashokitschool/ashokit_orkut_app.git

=> In git hub we can create 2 types of repositories

                a) public
                b) private


=========================
Git Bash Configuration
=========================

=> Open gitbash cli and execute below two commands

# Configure name & email (one time process)

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

# Verify configuration
git config --list



1) Created Account in www.github.com
2) Created Git Hub Repository
3) Downloaded & Installed Git Client software
4) Configured Name & Email in git bash


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

=> Working tree

=> Staging Area

=> Local Repository

=> Remote Repository

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

git config : To configure our name & email

git init : To intialize working tree

        $ git init

git add : To add files to staging area

        $ git add <filename>
        $ git add --a
        $ git add .

git status : To check git working tree satus

git commit : To send files from staging area to local repo

        $ git commit -m <msg>

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

git restore : To discard changes & to unstage the file

        $ git restore filename

        $ git restore --staged filename

git reset : To undo a local commit
```

```
        $ git reset HEAD~1

git revert : To revert (remove) git changes from remote repo based on commit id

        $ git revert <commit-id>


git log : To display commit history

git rm : To remove files from remote repo

git clone : To download remote repo to local using repo url

        $ git clone  <repo-url>

git pull : To download latest changes from remote to local

        $ git pull


===========================
what is .gitignore file?
===========================

=> .gitignore is a simple text file whithout any extension

        Filename : .gitignore

=> This is used to configure files & folders which we want to ignore from our working tree so that
those files & folders will not consider for commit.

        Ex: target, .classpath, .settings, .idea, .vscode etc....


Note: If .gitignore is not working then delete git locals cache using below command

        $ git rm -r --cached .


=====================
What is git stash ?
====================

=> It is used to save working tree changes to temporary area and makes working tree clean.

Usecase ::
-----------
9:00 AM :: Manager assigned Task-101 : I am working on this task (i am in middle of it)

11:00 AM :: Manager Assigned Task-102 and told that first complete 102 task and work on 101.

=> In this scenario we can stash changes made for task-101 and we can work on task-102. After task-
102 changes pushed then we can get back task-101 changes from stash area

# Create git stash with working tree changes
$ git stash

# To display all stashed
$ git stash list

# get stashed changes back to working tree
$ git stash apply


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

=> In one project, multiple development teams will be available like

        a) Enhancements Team
        b) Bug Fixing Team
        c) Reasearch & Development Team

=> If 3 team are working paralelly then managing code integrations will become very difficult because
we can't track which team is making which code change.

############## To overcome this problem we will use Git branches #############

=> Branches are used to maintain multiple code bases under one repository

=> Using branches multiple teams can work paralelly

=> In git repo we will have branches like below

        a) main
        b) develop
        c) qa
        d) feature
        e) release

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

=> We can create branches in www.github.com directley

# check our working tree pointint to which branch
$ git branch

# branch switching
$ git checkout <branch>

Note: If branch is not reflecting in local, then take git pull

=======================
What is Pull Request ?
=======================

=> Pull request is used to merge changes from one branch to another branch

Ex:

=> New enhancement related code pushed to develop branch. After testing completed we need to merge
develop branch changes to main branch for that we can use Pull Request.


====================================
How to clone particular branch ?
====================================

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

Note: If we don't specify branch name then it will clone default branch which is main.


================
Git conflicts
================

=> When two developers are modifying same file then there is a chance of getting conflict
```

=> If we have some local changes and if we take pull from remote then conflict may occur

=> When conflict occurs we need to resolve that conflict and we need to push the code without any conflicts.

Note: Its recommended to discuss with other developer who pushed code to repo to resolve conflict

========================================
## How to perform GIT Operations from IDE
========================================

=> Clone Git repo using git bash

=> Import Project into IDE

=> Right Click on Project  => Team => Git Options we can see and we can use them

Note: Instead of password use git token

=> We can generate git token in settings

Profile => Settings => Developer Settings => Classic Tokens => Generate Token

==================
## What is BitBucket
==================

=> BitBucket software developed by Atlasian Company

=> We can use this as Version Control Software

=> To perform operations in bitbucket repository we can use git bash commands

=========================
## Realtime Working Process
==========================

=> Development team will send request to devops team to create git repository

=> DevOps team will create git repo and will share URL to development team

=> All the developers will connect with git repo and will push their code into remote repo for code integration

=> Developers can create git branches & can merge merges using pull request

=================
## Git Hub Summary
=================

1) What is Version Control Software
2) Version Control Softwares available in market
3) Git Hub Introduction
4) Git Hub & Git Setup
5) What is source code repository
6) Types of Repositories
7) Git Branches
8) Branch merging with Pull Request
9) Git Conflicts
10) What is .gitignore file
11) Realtime working process with github

```
git config
git config --list
git init
git status
git add
git restore
git restore --staged
git commit -m
git push
git log
git rm
git branch
git remote add
git stash
git stash apply
git reset
git revert
git clone
git clone -b
git checkout
git pull

git fetch
git merge
git rebase


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