**Git topics**

1. Introduction

2. What is Version Control System (VCS) and types of VCS

a. Local Version Control System

b. Centralized Version Control System

c. Distributed Version Control System (GitHub, bitbucket,)

3. Git Software installation

4. Git project Architecture

5. Git commands or Operations

|  |  |  |  |
| --- | --- | --- | --- |
| git help | git commit | git fetch | git show |
| git init | git log | git stage | git revert |
| git config | git clone | git merge |  |
| git add | git push | git rebase |  |
| git status | git pull | git diff |  |
| git rm | git branch | git revert |  |
| git restore | git checkout | git mv |  |

6. Git commands execution

a. using command line (gitbash, gitcmd, gitgui)

b. using IDEs like Eclipse/STS (Spring Too lSuite)/IntelliJ

7. GitHub account creation

a. public repository creation

b. private repository creation

8. Git folder structure

9. Git Branching Strategy

1. developer branch

2. master branch

3. release branch

4. hofix branch

5. fork branch

10. Project Review Process (PR process), Code Reviews, Code Merge

11. Realtime problems with git and how to fix them

12. FAQ's

**Introduction**

Git is a popular Version Control system (VCS)

It was created by **Linus Torvalds** in 2005 and it is maintained by **Junio Hamano**

Git is used for

a. Tracking code changes

b. Tracking who made the changes like history of files

c. Coding Collaborations’

**What is Version Control System (VCS) and types of VCS?**

It is a system that records changes made to the file or set of files over the time,so that we can recall the specific version later.

ie, for every source code change in a file a new version will be created

eg: JDK1.0V, JDK1.1V, JDK1.2V, .........

Spring1.X, Spring2.X, Spring3.X, ......

**Types of Version Control Software (VCS)**

There are 3 types of VCS

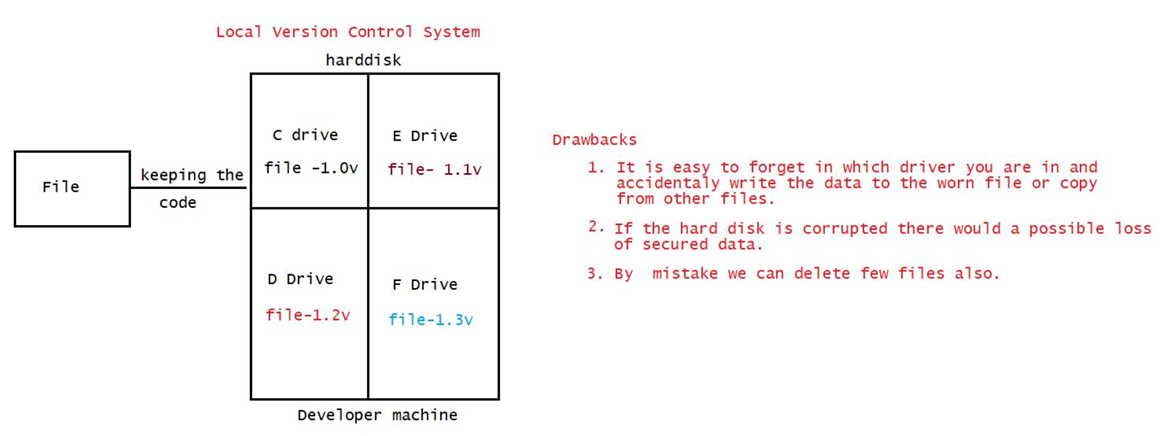
a. Local Version Control System

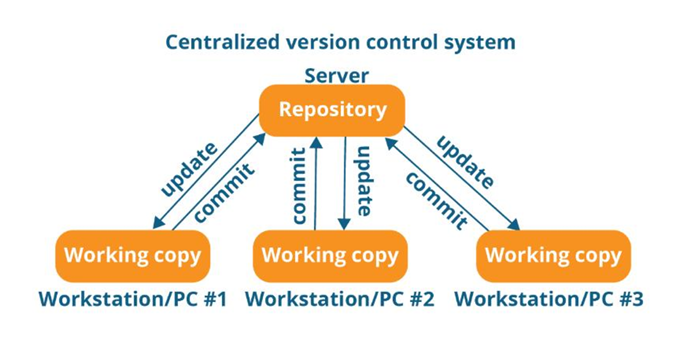
b. Centralized Version Control System

c. Distributed Version Control System

1. **Local Version Control System**

It is used to maintain the file version and retrieve the files based on the specific version refer:





To overcome the drawback of Local Version Control System we have "Centralized Version Control System".

1. **Centralized Version Computer**
2. Developers can collaborate the code in one repository and do the change. eg of Centralized Version software’s: SVN, Subversion, Perforce, ......
3. Centralized Version server will have single server that contains all the version files
4. For many years this has been the standard version control system
5. More no of developer’s would connect to CVS to checkout the files

Note: Checkout -> taking the code from repository to local machine.

push -> sending the code from local machine to repository(CVS)

Advantage

1. Everyone know to certain degree what everyone else on the project is doing.
2. Administrator will have full control over which can do what and it is easier to manage.

Drawback

1. Single point of Failure (SPF) would represent the Centralized system.
2. If the server goes down due to network traffic, during that hour nobody can collaborate at all or save changes to the server.
3. If the hard disk of the centralized system gets corrupted and proper backup haven't been taken then there is every possibility of loss of data.

Note: In LVCS ad in CVCS getting up the complete history of changes is not possible.

It is possible to get only the latest version, but not the entire history.

eg: SVN push will not happen w.r.t version rather push will happen only with the latest change.

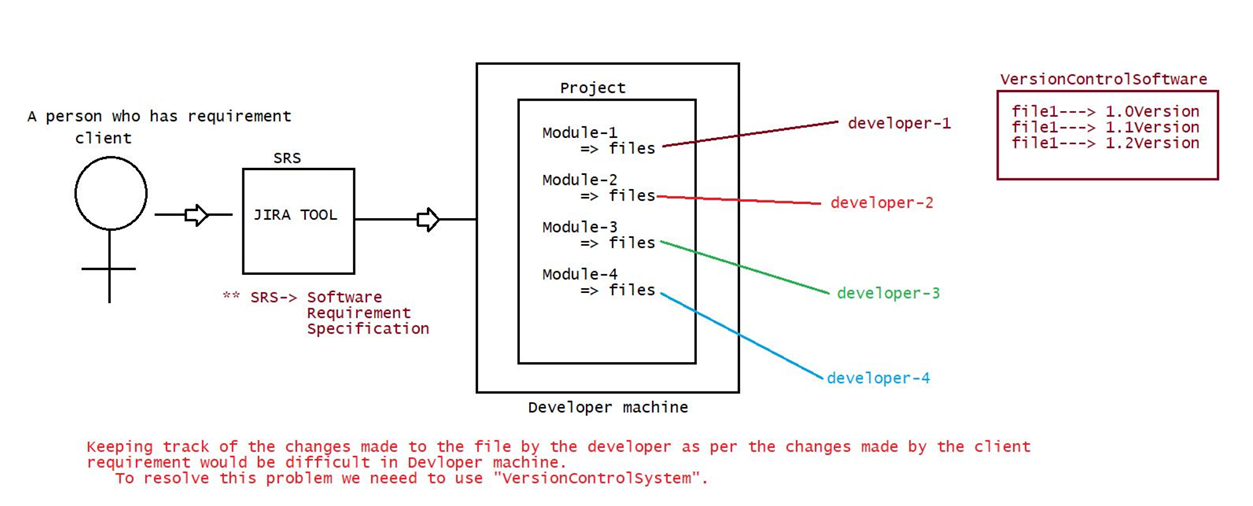
Version history

file -> 1.0V

file-> 1.1V

file-> 1.2V

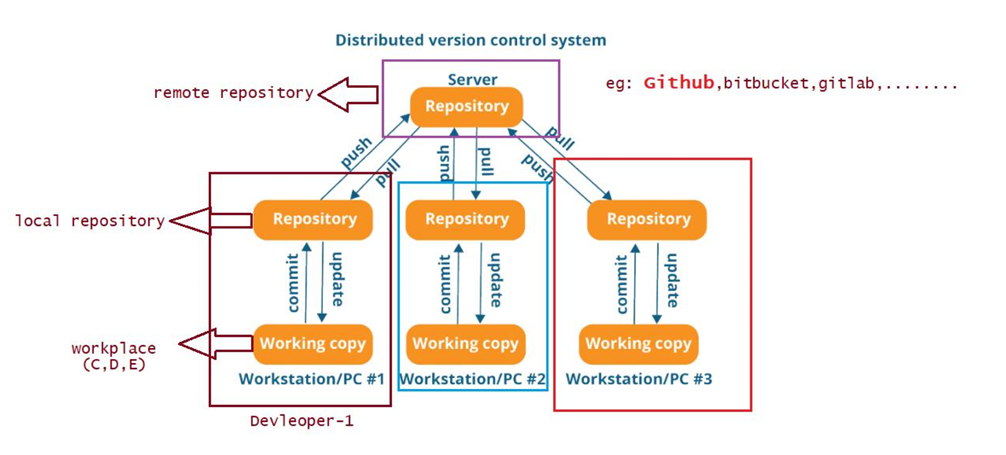
file-> 1.3V

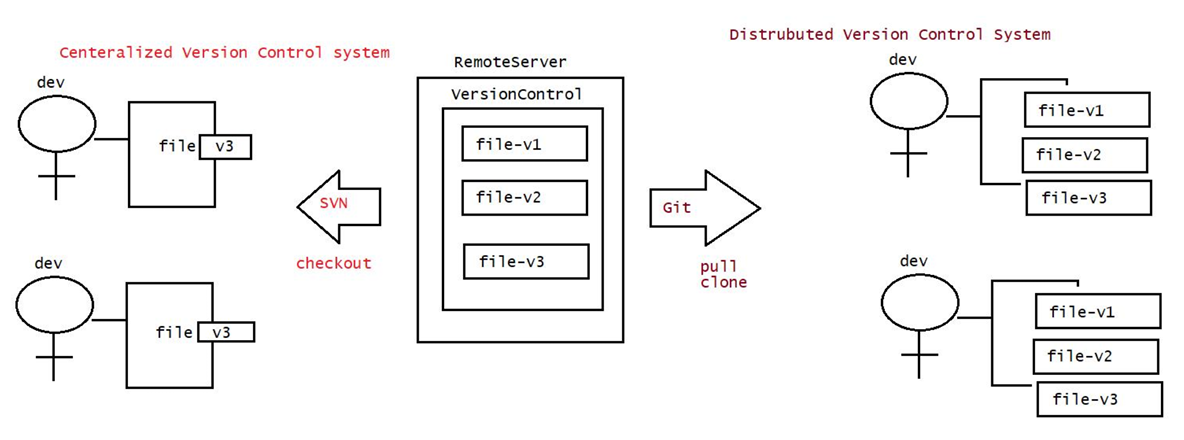


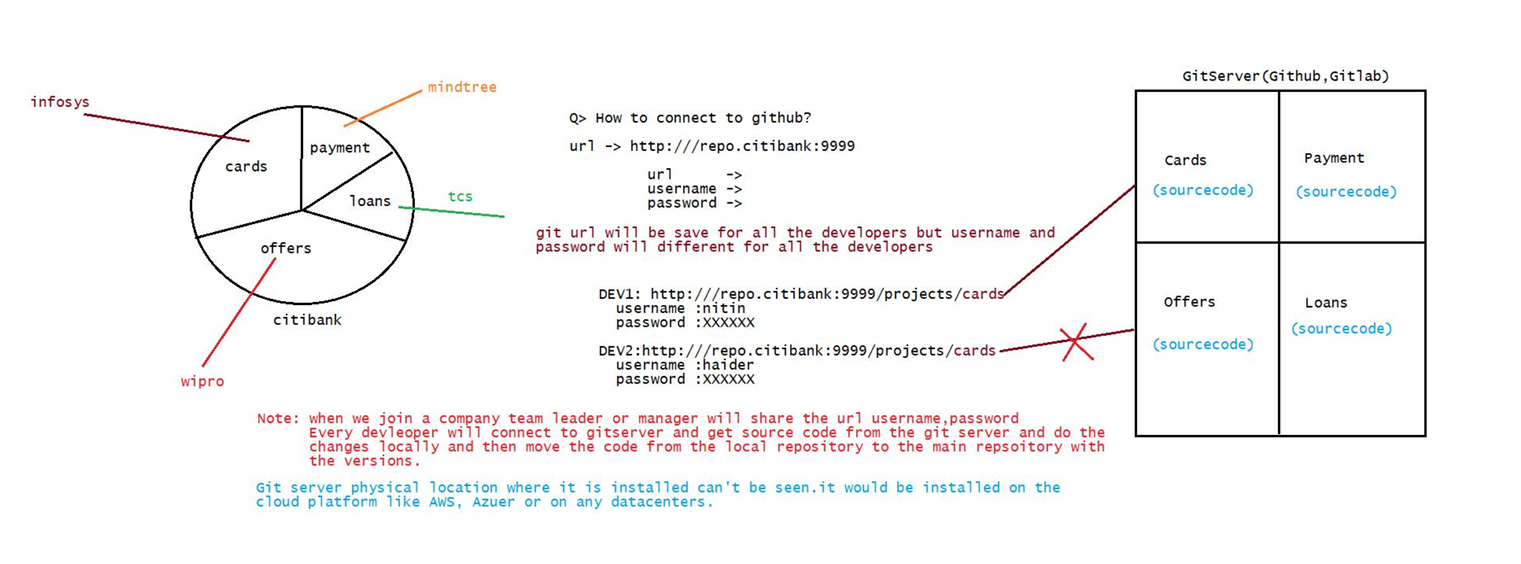
**3.Distrubuted Version Control System**

Eg: Git, Mercurial, Darcs, Baazar, etc,....

1. Developers will not only get the latest version but also the compile history of the files
2. Push will not only happen with latest snapshot of the files rather they will push the old files also.
3. If the main sever goes off, still there is a local repository which would have maintained the copy of the repository where the entire code is available (history of versions).
4. If the remote repository is down, then developer can do changes in the local repository and when the main repository is up the code can be pushed to remote repository from local repository.





****

**Git installation**

There are 2 types of Git software

1. Git Server
2. Git Client

**Git Server**

It is a repository

It is the largest host of source code in the world.

It is used to store/manage the source code of the project.

Some of the Git server tools are: Github, BitBucket , GitLab , .......

refer: gitserverarchitecture.png

Where should we provide url, username and password?

To type these details, we need git client

**Git Client**

Installation of git software

1. Download a git software from the following

link https://git-scm.com/download/win

It is a tool which is used to connect to our git server.

if we install git client (git s/w) we get the following tools for free

1. git bash => Linux commands are required
2. git gui => Graphical user interface where all the actions will be done through clicks
3. git cmd => command line tools where developer should provide URL, username and password

Note:

Git client is a .exe file which can installed with just few clicks.

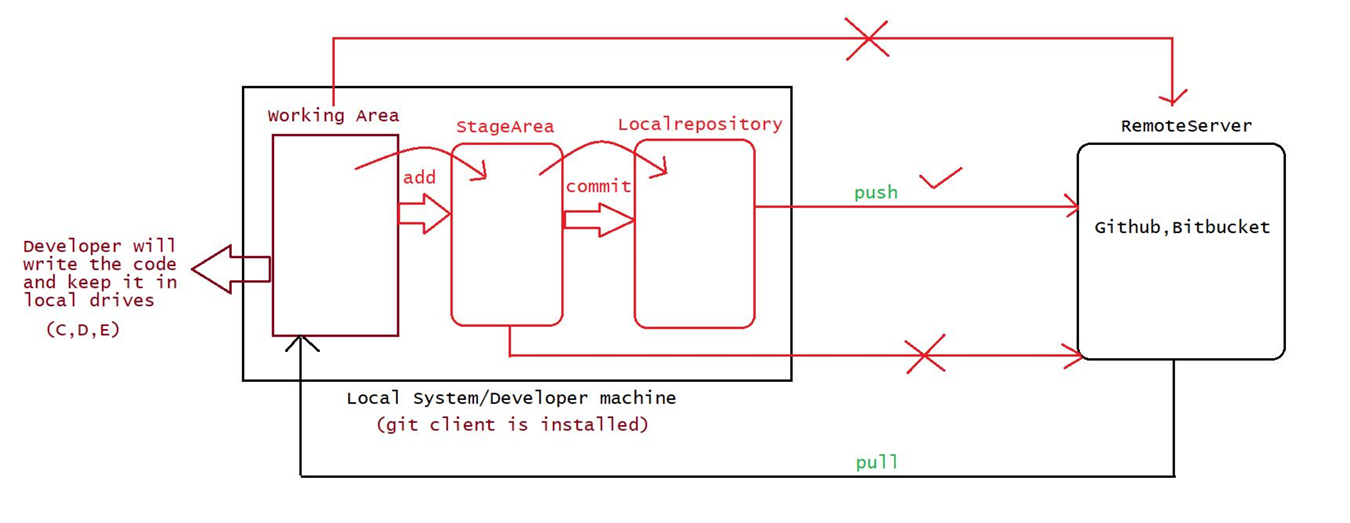
git -> client tool where the client should provide url, username and password

GitHub-> server software where repositories/projects will be maintained

**Git Architecture**

There are 3 regions

1. workplace => It is a place where developers maintain their source code
2. stage area => Once the code is ready, then it will be added to stage area (indication to git software)
3. local repository => Once the code is in stage area, we commit it to the local repository with some standard message, from local repository we "push" to main repository by providing url, username and password



**Yesterday, Topics**

1. What is the need of VCS?

2. How many types of VCS?

a. LVCS

b. CVCS

c. DVCS

eg: SVN eg: GitHub,bitbucket,gitlab,.....

3. Architecture of git

a. working area => devleoper workspace(source code)

b. Stage area => Before adding the code to the localrepository, we keep the code in stage area

c. Local repository => Before pushing the code to remote repository, we keep the code in "Local repository".

4. What is git and github?

git -> It is a client tool where the user will enter url,username and password of github repository.

github-> It is a server where we keep repositories/projects which would be used for colloboration.

git commands(case sensitive)

1. git version

2. git help

3. git config

4. git init

5. git clone

6. git add

7. git status

8. git rm

9. git restore

10. git commit

11. git log

12. git show

13.git push

14. git pull

15. git branch

16. git checkout

17. git stash

Note: git merge, git rebase

1. **git version** - This command is used to check the version of git

syntax: git version or git --version

nitin@DESKTOP-1N5U4UJ MINGW64 ~

$ git version

git version 2.37.0.windows.1

nitin@DESKTOP-1N5U4UJ MINGW64 ~

$ git --version

git version 2.37.0.windows.1 2.

1. **git help** - If we want to see the list of commands then we can use git help

syntax : git help

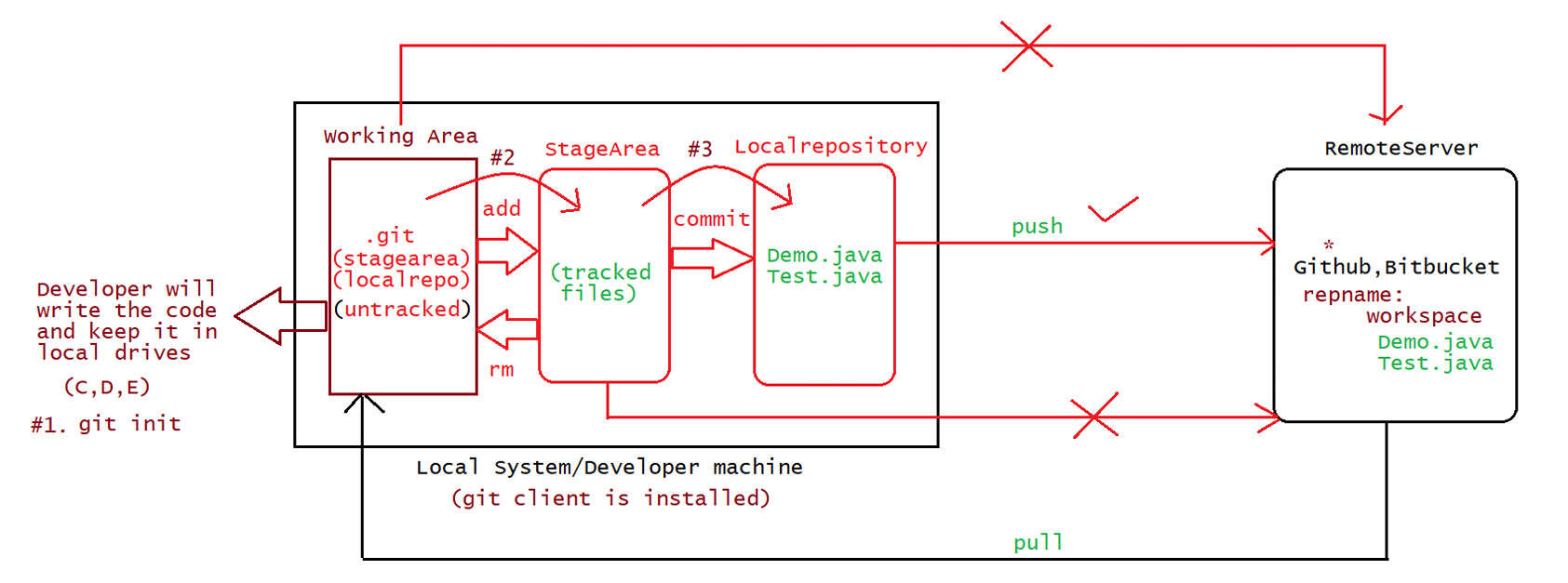
Note: This command is useful to get the documentation of any command

eg: git help

1. **git config** - It is used when the git software is used for the first time.

The command will set the devleoper identity like name,emailid,....

This configuration information will be used by git software for every push operation encountered.



1. git config --list //this command is used to provide the list of configuration

**$ git config --list**

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

**$ git config --list –show-origin**

//to set the username and email

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

**$ git config --global user.email** [**devavrat.wadekar@gmail.com**](mailto:devavrat.wadekar@gmail.com)

**$ git config --list**

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

user.name=devavratwadekar

user.email=devavrat.wadekar@gmail.com

global => it indicates the user can work with git commands from different drives of computer.

Note: git config --list --show-origin //display the location of git configuration holded by git software

**Important operations associated with git**

**git init**

1. Normally a folder will be created in the developers works place and inside the folder the source code would be place
2. Normally this is the first command which we execute to set up the git for operations like clone,push,pull,....
3. This command internally creates one folder called .git
4. .git is used by git software to identify the folder which should participate in pushing to "local" and "remote" repositories.

syntax : git init

Note: To change the directory use the following command cd .....

To check in which directory we are currently present we use pwd command

**git status**

1. This command is used to check the status of the working directory

$ git status

D:\gitsession\Workspace-1(master)

|=> Demo.java

On branch master

No commits yet

Untracked files: (use "git add ..." to include in what will be committed)

Demo.java

nothing added to commit but untracked files present (use "git add" to track)

git status

normally will give outputs in the following ways

a. untracked files(red color)

It means the files are present still in working area and these files can't be commited to "local repository" nor to "remote repository"

b. tracked files (green color)

It means the files are moved from working area to stage area so these files can be commited to "local repository" and to "remote repository".

* 1. c. modified files (red color) => it means the files are present still in working area and these files can be staged or it can also be restored back to the normal phase.

1. To send the code from workspace to stagearea we use the following command

syntax : git add

1. If we want to push all the files from workspace to stage area,we use the following command

syntax: git add .

git add --a

1. It is also possible to unstage the files from stagged area to workspace,using the following command syntax: git rm --cached <file-name>
2. To restore the old file, we use the following command

syntax: git restore <file-name>

1. The files which are ready for commit should be in stage area, to perform commit operation we use the following command
2. syntax: git commit -m
3. eg#1. git commit -m "first commit"

//This file commit all the files present in stage area

1. eg#2. git commit -m "second commit" filename

//This will commit only that file into local repository

**Steps followed to create a remote repository and push it to remote repository**