

## VERSION CONTROL SYSTEM PART 1 :-

- also known as source control system or revision control.
- It is basically a management of document, programs, large web sites or other collection of information.
- tightly coupled with software development.

## VCS KIND OF DATABASE :-

- VCS options Git, Mercurial, SVN, Perforce.
- kind of database b/c it has snapshot of your project at time you want.

## - VCS IS INDEPENDENT :-

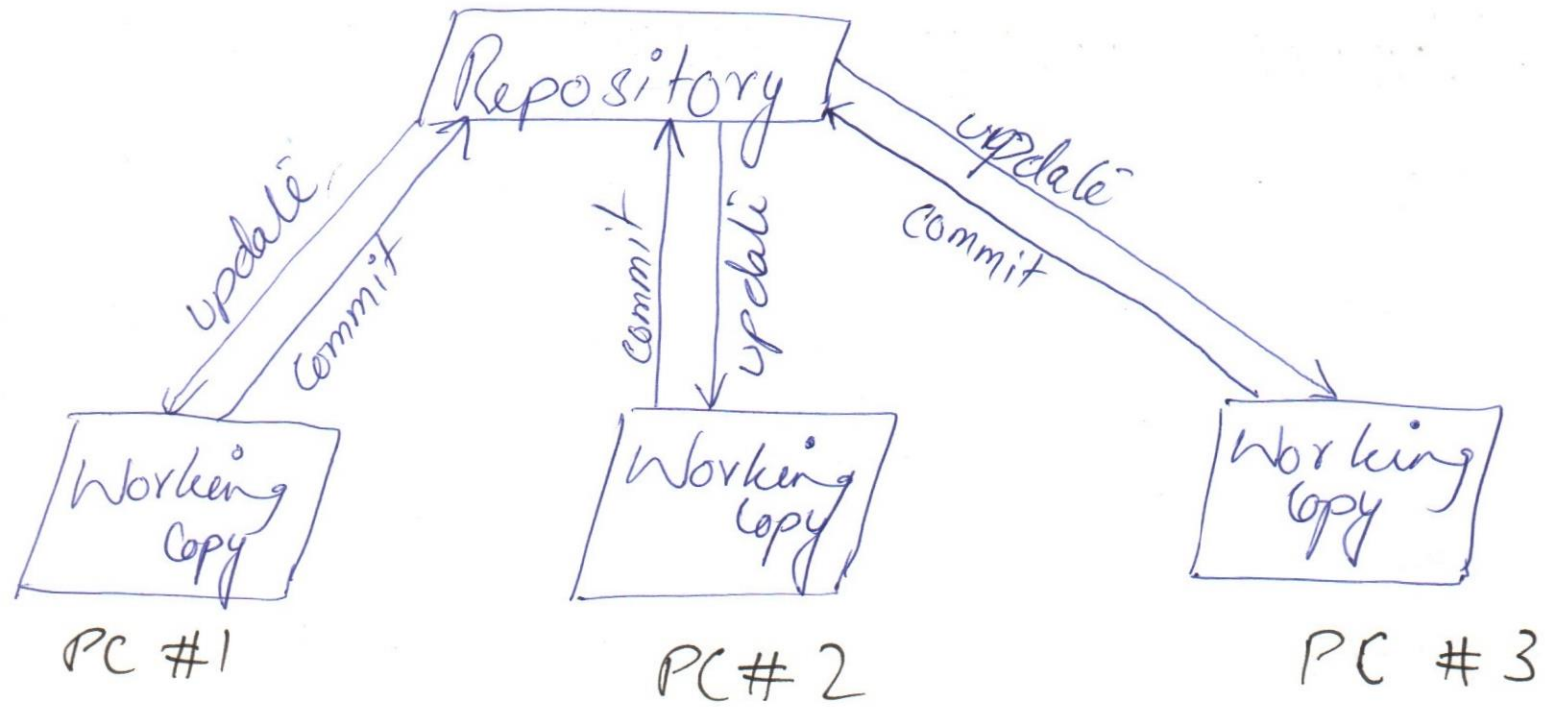
It is independent of kind of project / technology / framework / editor.

## - WHY USE VCS: COLLABORATION

- Git is distributed VCS

# CENTRALISE

# VCS

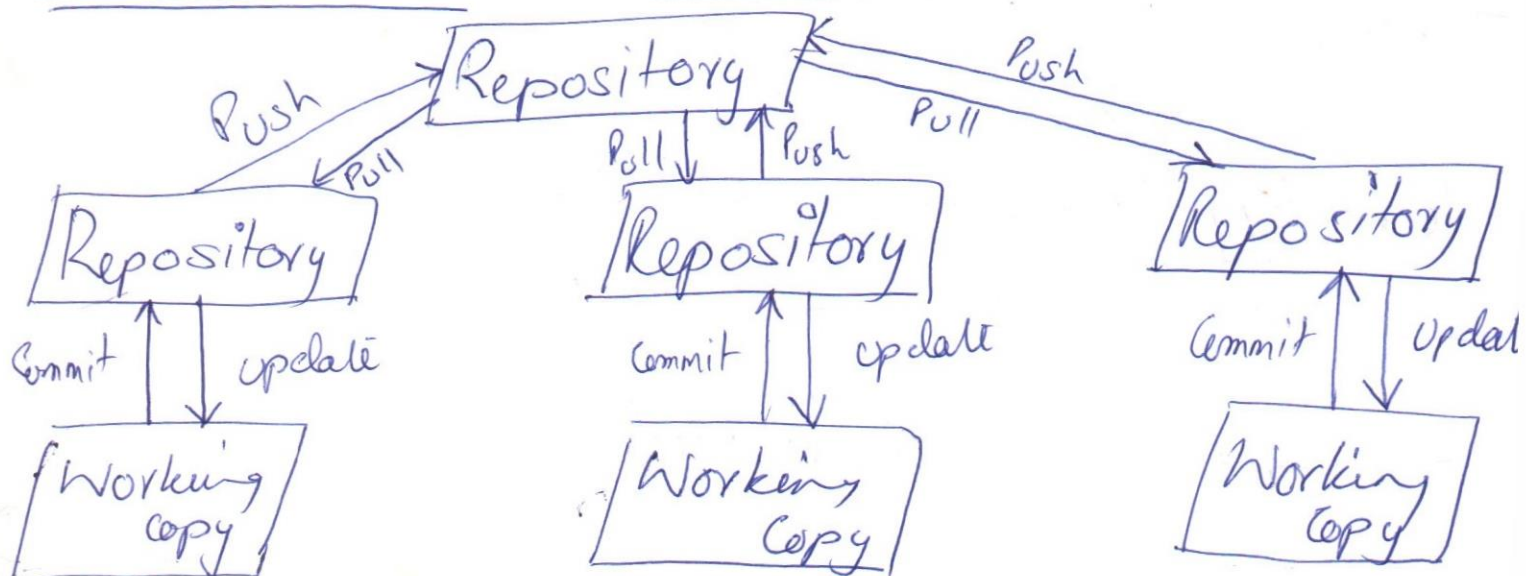


— Every ~~operation~~ operation is on central server.

## Drawbacks:

- Network or internet is required
- Server/machine lost mean data lost.

# DISTRIBUTED VERSION CONTROL SYSTEM





# DVCS ADVANTAGES :-

21-JUL-2020  
TUE 10:42 PM

- ① All operations are speedy since data is being accessed from hard drive not from remote server. (You do not need internet conn.)
- ② Changes if ~~set~~ are committed so you still have a ~~change~~ chance to push or revoke changes.
- ③ If central server crashes, so any user's copy can be utilized for a master copy.

## INSTALLATION AND BASIC WORKFLOW :-

- Go to [www.git-scm.com](http://www.git-scm.com) to download Git
- & [www.syntero.com](http://www.syntero.com) for Smart Git.

1. First install Git.

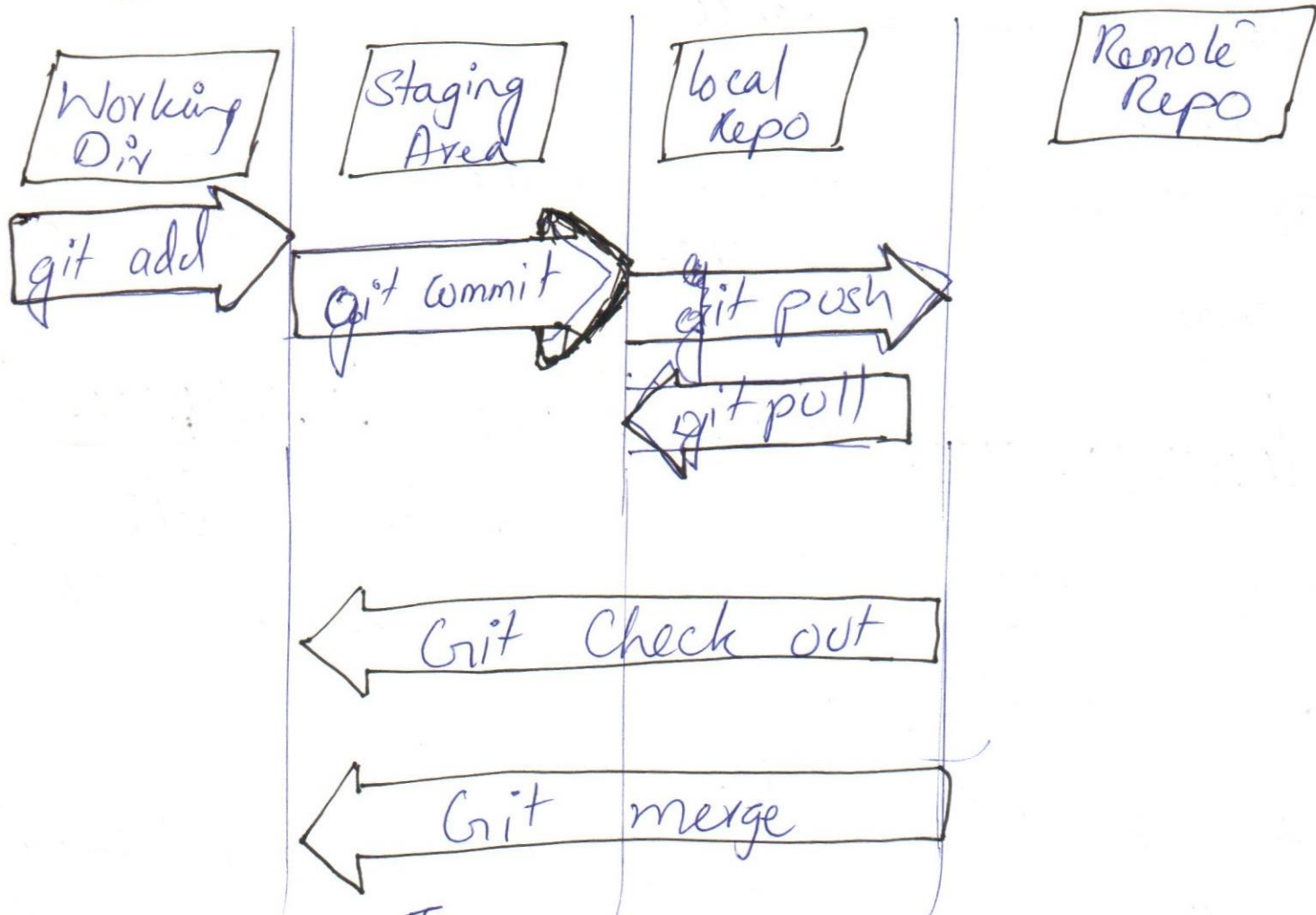
- ~~#~~ git config -- global user.name "Zohaib"
- # git config -- global user.email "Email"

SMART GIT:

- Select non-commercial user only.

# Git OPERATIONS :-

- 1) Initialization.
- 2) Add
- 3) Commit
- 4) Pull
- 5) Push



## IMPORTANT TERMS

- ① Repository.
- ② Working directory.
- ③ File status. (Tracked/Untracked)



23-7-2020

8:38 PM

THURSDAY

# GIT INSTALLATION

- Download GIT:-

- go to <https://git-scm.com/download/linux>

For GUI:-

[www.syntexo.com/smartgit](http://www.syntexo.com/smartgit)

Installation of git:-

Next, Next... next.

SETTING UP GIT:-

```
# git config --global user.name="Zohaib Attar"
```

```
# git config --global user.email="Zohaib4986@gmail.com"
```

SETTING UP SMART GIT:-

- Non commercial

GIT OPERATION:-

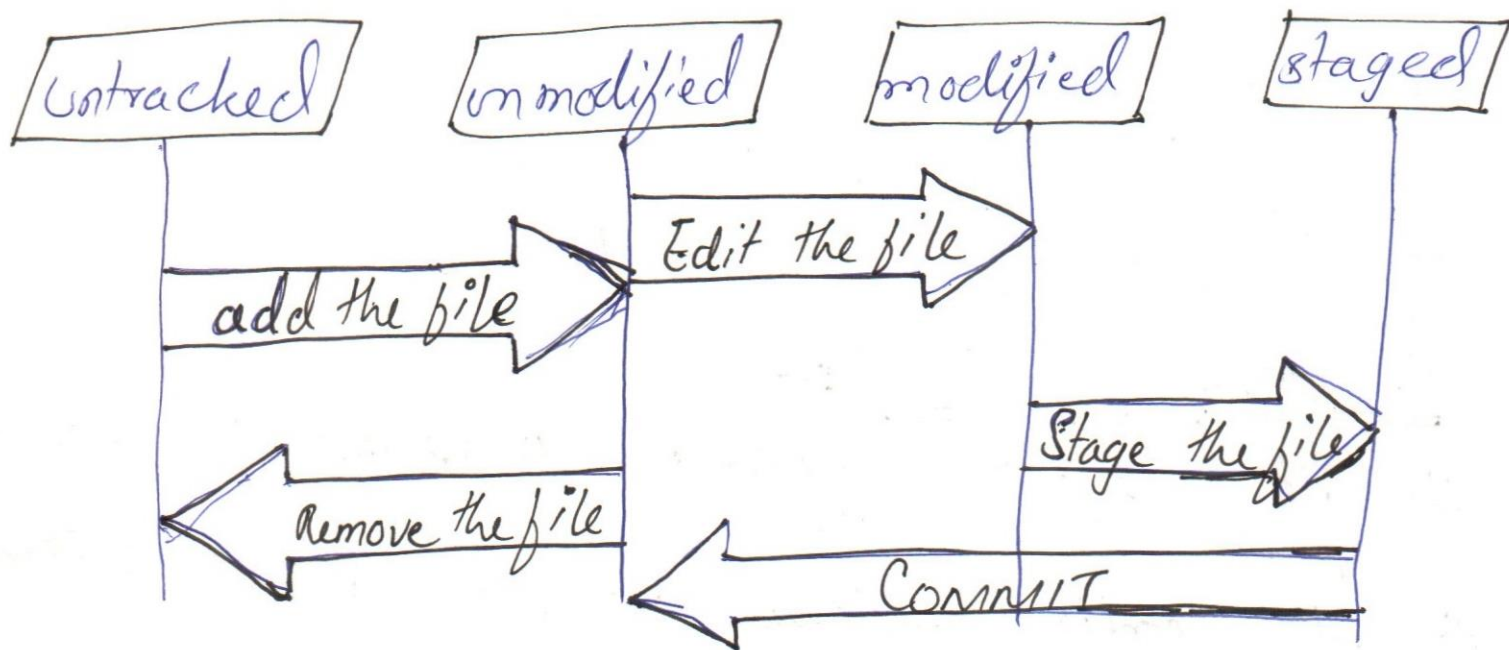
- Initialize
- Add
- Commit
- Pull
- Push

ADVANCE OPERATIONS:-

1. Branching
2. Merging.
3. Rebasing

## IMPORTANT TERMS :-

- 1) Repository.
- 2) Working directory
- 3) File status.



## 4) Staging Area

## BASIC WORKFLOW :-

- Open terminal and create directory.  
C: \Repo \ my project
- # git init (to initialize repo in directory)  
This will create .git hidden folder
- To add file
- # git add first.txt second.txt



• Commit Message:

# git commit -m "implemented new feature"

• Git history

# git log

# Git status: - (# git status)

To check changes

To ADD:-

# git add.

OR

# git add \*.txt.

OR

# git add first.txt

COMMIT: PAST, PRESENT & FUTURE

- If C1, C2 are the commits then C2 will be HEAD, current represents Branch (master), and next commit will future commits.

COMMIT HASH:-

Every commit provided with a unique identifier of 40 bits.

## UNSTAGE FILE

By git reset command, it will remove from staging area along with changes if not committed.

# git reset.

# git reset --hard (remove from staging area and discard)

## \* IGNORING FILES :-

Some files or folder if exist in Git repo then you may ignore them.

E.g

- .DS\_store (MAC)

- Node modules

- Build

- logs

so you will create a file without name having extension .gitignore

E.g

- \* .doc

- \* .js

third.txt

build (folder)



# BRANCHING & MERGING

26-July-20

8:41 AM  
SUNDAY

Basically a concept of branch is to separate working space from another. like if you making project of <sup>new</sup> web site and suddenly you need change old (already deployed web site) then you change your branch; add, changes and submit changes and then revert back previously working branch.

## BRANCH COMMANDS

① show list of branches

# git branch

② Show list of branches with some details

# git branch -v

③ Create new branch with new name

# git branch new-dev

④ Switch to new 'new-dev' branch.

# git checkout new-dev

⑤ Merge branch into current active branch

# git merge new-dev

① Show commit difference in two branches

```
# git log new-dev -- master
```

OR

```
# git log master .. new-dev
```

## STASH :-

- A clipboard or temporary place where git files get save and these changes are not committed to repo.

- Later you can restore the changes from stash in your working copy and continue working where you left.

- you can create as many stashes as you want.

```
# git stash # git stash save <name>
```

```
# git stash apply stash-name stash@{1}
```

```
# git stash list # git stash pop
```



# REMOTE REPOSITORIES

## Online services

- Github
- bitbucket
- Gitlab
- Many other

So you don't need to maintain individual  
own central server,

## Cloning a repository

# git clone url

## COMMAND FOR REMOTE REPO :-

# git push  
↳ push changes to remote repository

# git fetch  
↳ Fetch changes from remote repository.

# git merge  
↳ Merge changes that was fetch by 'git  
fetch command.

Instead of git fetch & git merge we  
can use

# git pull

↳ Fetch and merge changes from remote repository.

# git remote -v

↳ Show remote urls

# git remote show origin.

↳ Show details of origin.

# git remote add myremote

↳ This command will add remote repo to local repository.

PUBLISH LOCAL REPO TO GITHUB

# git remote add origin url  
repo name

# git push -u origin master  
repo name branch

-u flag establishes a tracking connection  
b/w remote and our local



## COMMAND FOR REMOTE REPO

# git push 'remote repo name' 'branch'  
like

# git push origin master.

# git log remote/branch or origin/master.

## PUBLISH LOCAL BRANCHES

# git branch work  
branch name.

# git checkout work

# git push -u origin work

# ADVANCE GIT

8:25 AM TUE  
28-JULY-2020

## GIT WORKFLOW:-

It is possible while working with git that your code need to be tested and verified before going into production so git advance gives us a feature that in which a developer request the authority to push request and after review.

Pull request is a github feature

## Fork

If you donot have write access of repository then you fork means a copy of repository is create in your github account so that you can made changes ~~and~~ then like open source project.

- You can request pull request if so its on owner discretion that whether they accept you changes or reject.



## DELETE BRANCH :-

# git branch -d contact-form

For remote branch

# git branch -dr origin/contact-form.

To delete branch from git hub

# git push origin --delete test.

## UNDOING LOCAL CHANGES :-

# git checkout head <file to restore>

# git reset --hard Head

## UNDOING COMMITTED CHANGES

# git revert <commit hash>

# git reset --hard <commit hash>

ADVANCE GIT -- REBASEALTERNATE TO MERGE -- REBASE

- merging is the easiest and common way to integrate changes.
- Rebasing is quite a bit more complex than merging.

MERGE

- Fast forward: (simplest one) changes only one branch at a time
- Merge commit: create a additional commit of merger

REBASE :- Add changes in straight line.

# git rebase <branch name>

PITFALL / DISADV. :-

If a commit like C3 that is in b/w C2 & C4 will be added as new commit

BENEFITS :-

All the commit being done branch A will be seems for branch A.



# MARKDOWN

markup

- Markdown is a light weight language that you can use to add formatting elements to plaintext documents.
- Created by John gruber in 2004.
- often used for 'readme' files

GFM (Github flavoured Markdown)

Heading :-

It has 6 heading for big heading use '#'  
single hash or for smaller one use 6 '#'  
six hashes.

Styling Text :-

Bold      \*\* text \*\*      or      \_\_ text \_\_

Italic      \* text \*      or      \_ text \_

Quote text :-

Use > to quote

Quote code :-

Use ` text ` to format text in distinct block.

Links :-

use { text } to make it URL like  
[Github Pages](https://pages.github.com/).

List :-

Order list use '-' hyphen will convert into bullet

Nested :-

1. Hello  
    - World

Task list :-

use - [ ] or - [X] - checked