Git

Stores files, history, config

**Hidden Repository (.git)**

Contains older versions of working code in the working directory.

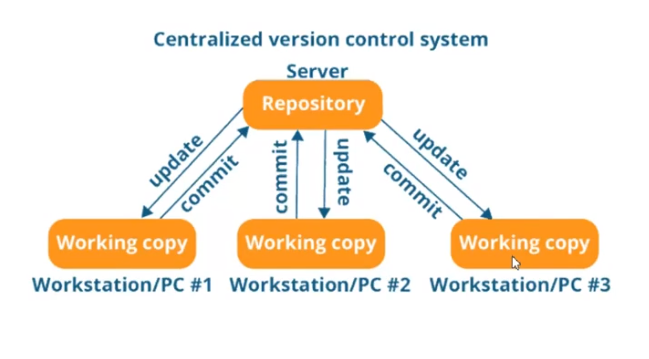
3 stages

1. Working directory
2. Staging area (committed changes but not pushed to the remote)
3. Git repository

Remote Repository (github or server)

Master Branch (repository where you are committing)

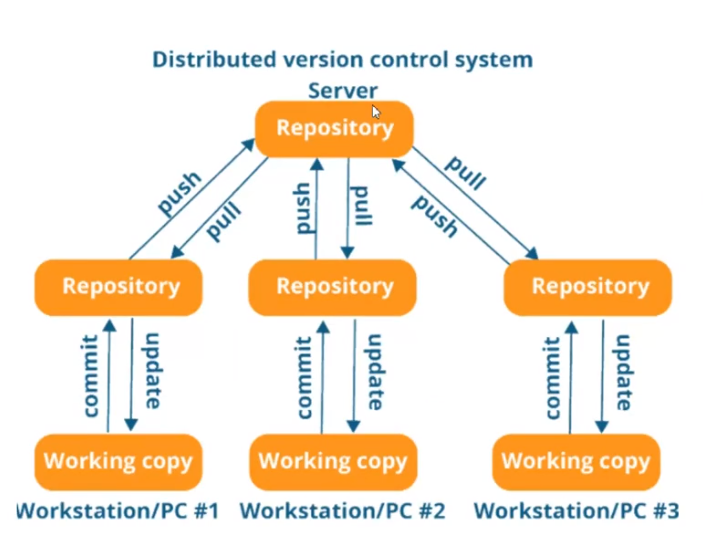
**Centralized Version Control System**



Drawbacks:

Only one can work on the centralized repository at a time.

Distributed Version Control System



Git is Distributed Version Control System

**Configuring a User**

git config

List all possible configurations

Important ones

git config –global user.name “Venkataramana, Jeevan”

git config –global user.email “jeevan.venkataramana@gmail.com”

git config –list (list all details configured)

**git help**

git help add – complete details about add

**Initialize the repository**

git init

**Status of the Repository**

Return information about last commit and changes done after that.

git status

red marked files are not added into the staging area

**Add Changes**

Adds the changes before committing

git add .

add only a certain file

git add <filename>

**Commit Changes**

Commit changes in to the staging directory

git commit -m “<status here>”

**Logs or Commit history**

History of commits

git log

commits done specific to a user

git log –author=”<name>”

commits of a certain file

git log -- <filename>

**Difference in text file**

git diff

difference between the staged repository and working repository

git diff –staged

**Delete a file**

git rm <document name>

**Adding Remote Repository (GITHUB)**

git remote add origin “<copy link here>”

**Pulling Remote Repository from Github**

git pull origin <branch name or master>

**Push Changes to Remote Repository**

git push origin <branch name or master>

**Branches**

Master branch – can be the main code

Branches – different teams can create their own branches and commit to their local branch.

**Creating Branch**

git branch <new branch name>

master branch code will be copied to new branch. (not a duplicate copy but a new pointer link to memory)

changes done on 1 branch will not reflected on master branch unless merged.

**Switching to different branches**

git checkout <branch name>

when you switch only files in that branch will be visible in that folder.

**List all branches**

git branch

Merging branch with master branch

Checkout to master branch git checkout master

Merge the branch git merge <branch name>

It will automatically be committed.

**Un Staging**

git reset HEAD <filename> -- after green

Changes done but not staged. You have to revert back to code that was there after previous commit.

git checkout -- <filename> -- when in red

**Revert a Commit in GIT**

Revert to last commit

git reset HEAD~

Revert with commit id

git revert <commit id> -- reverts the changes done on that commit

**Fork an existing project on GITHUB**

Copy some ones repository

Fork on GUI

Clone into local Machine

git clone <url>

**Instructions to push**

git init

git add .

git status

git commit -m “<memo>”

git remote add origin <repo url>

git push origin <master or specific branch name>

**Instructions to pull**

git pull origin master

git push origin master

**Creating Alias**

When a long command is used again and again

Suppose the below command has to be shortened

git log --all --graph --decorate –oneline 🡺 git <shortened name>

git config --global alias.history “log --all --graph --decorate –oneline”

git history

**Excluding files from working directory**

Create a .gitignore file

nano .gitignore

write the file names which needs to be excluded into it

ignoring text files: \*.txt

ignoring files in a directory: <directory name>/