# Git architecture



# Git terminal commands

Git is called as distributed because , we have both local and remote

Svn doesn’t have local git repo, it will have only remote repo, since git is distributed we have both local and remote

| **Centralized** | **Distributed** |
| --- | --- |
| You can keep changes only in the server | You can keep changes locally (commit) as well |
| Changes can be merged in the server (remote) alone | Changes can be merged locally as well as remotely |

* git clone: Get the complete project from remote to your local machine
* git pull origin <branch\_name>: Get the new changes from remote branch to local branch
* git push origin <branch\_name>: Send your local branch changes to the remote branch
* git remote add <name> <url>: Add a new remote repo link to your local repo
* git remote -v: List all the remote repo URLs linked to your local repo

## set the email after installing git

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

$ git config --global user.email "myemail@domain.com"

* **git init** adds .git folder and **initializes the current folder to track its changes**
* **git status** displays the current state of the staging area and the working directory, that is, which files are added/removed/modified
* **git diff** **shows the exact changes** with line and column number
* **git add** adds the changes to the staging area. If you have added a new file, this command **starts tracking** the file for modifications.
* **git commit** will **save all the changes** with a unique hash number in the local repository
* **git push** sends the changes to the remote repository (server)
* git log
* git show
* git diff

**HEAD** is a reference variable that always **points to the tip of your current branch, that is, recent commit of your current branch**.

**HEAD** can be used with the following symbols to refer to other commits:

* Tilde symbol (~): Used to point to the **previous commits from base HEAD**
* Caret symbol (^): Used to point to the **immediate parent commit** from the current referenced commit
* git log -2 displays the history of **last two commits**
* git log commit\_id shows the history **starting from commit\_id**
* git log filename displays the list of commits for the file
* git pull is the convenient shortcut key to fetch and merge the content.
  + git pull <remote\_name> <branch\_name>
* git fetch command downloads the remote content to your local repo, **without changing your code changes**.
  + git fetch <remote\_name> <branch\_name> fetches the content from that specific branch in remote to your current working area
* git merge command merges the fetched remote content to the local working tree.
  + git merge <remote\_name>/<branch\_name> merges the content to the specified branch.
* For example: git remote add origin https://github.com/play/repo.git

Note: Your local repository can be linked to multiple remote repositories as **git remote add**origin1**<url>**, **git remote add**origin2**<url>**

Delete a folder

Rm –rf <folder name>

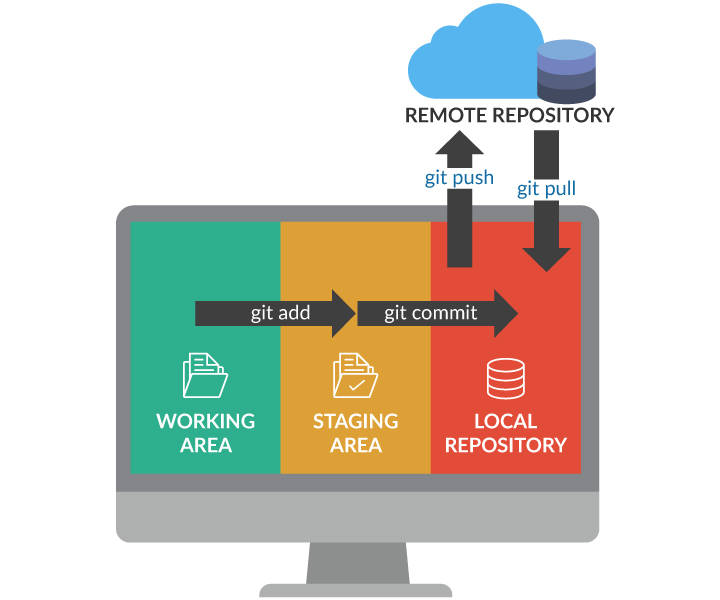
//here –r means –recursively, -f means force

Most used Git commands

**git status**

it will show the files which files are changed on the system

**git add**



All GIT commands

To initialise a local git repository

// This will create a local git repository

git init

// this will create a local empty git repo inside the specified folder

git init <folder name>

To open windows folder from current git command prompt

start . or explorer.exe .

here dot represents current directory

this start . will even work in windows application also

**Clone- pull or check out the code**

1. Best way is git clone 2) another way is git pull <remote url alias >

git clone <url ends with .git>

git clone <https://github.com/manideep-vv/SpringPaintBrushes-1.git>

This command **downloads the complete project,** all branches, commits and logs from the given remote URL (react repo here) to your local machine

If u want to see what are all the branches it pulled “git branch –r” (here r stands for recursive)

1. Initialize an empty git repository using (git init) and do

git pull <https://github.com/manideep-vv/Forked-Junit-5-samples.git>

1. With out using url everytime , set a remote and create a alias for above url

|  |  |
| --- | --- |
| Initialise an empty git repo | git init |
| Configure the remote url to our local repo and create alias name for that url | git remote add maniorigin https://github.com/manideep-vv/Forked-Junit-5-samples.git |
| Pull specific branch | git pull maniorigin b1  // pull from that url and mention that specific branch |

Connect local repo to remote repo

#### Setting a remote -aliasing

Lets say u initialized a empty git repo using “git init” and If u want to connect that local repo to remote git repository then follow

1. Add a remote to the local repo

-First u should have created an empty git repo using “git init”

We should point the local repo to remote repo by creating a remote

& create an alias name for the remote git URL and point our local repo to remote git url

//pointing to a remote git URL

git remote add <alias name for the url> <remote git url>

git add remote origin <https://github.com/manideep-vv/Forked-Junit-5-samples.git>

// here we have set origin as an alias name for the git url, so for everytime for git pull and push we can use that alias name instead of git url

|  |  |
| --- | --- |
| git pull origin <branch-name> | No need of  git pull <git url> <branch-name> |
| git push origin <branch-name> | No need of  git push <git url> <branch-name> |

Generally if u want to pull remote changes to local,

1st best way is git clone <git url> // this will fetches or downloads all branches

2nd option is create a empty git repo and add a remote and pull using

#### To see git remote repo

git remote

git remote –v

// this –v will give alias name and remote repository pointing url

1. Pulling a specific branch

I have only 2 branches called b1,b2

git pull <alias name of git url> <branch name>

git pull origin b1

1. Changing the alias name or Renaming a remote

git remote rename <current alias name> <new alias name>

git remote rename origin Canada

after that u can pull as

git pull Canada b2



Branches

1. If u want to see what are all the branches available in local then type “git branch –r” (here r stands for recursive)
2. If u want to see on which branch u are then type git branch

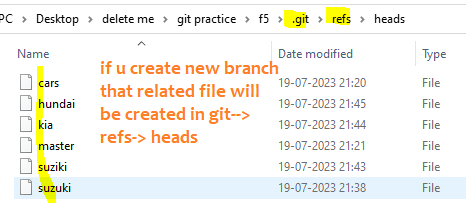
3) To switch to another branch git switch -c b1

1. Pulling a specific branch

git pull <alias name of git url> <branch name>

git pull origin b1

|  |  |
| --- | --- |
| If u want to know on which branch you are | git status |
| To see the list of all branches | git branch |
| Create a branch from HEAD | git branch <branch name>  git branch suzuki |
|  |  |
| Switch to another branch | git switch <existing-branch-name>  git switch cars |
| Create new branch from HEAD and switch to new branch | git switch -c <desired new branch name >  git switch -c hundai |
| Checkout- this is also used to switch to another branch= checkout means switch & many other functionalities | git checkout <existing-branch-name>  git checkout cars |
| Delete branch | git branch –d <branch-name> |
| If u want to see all commits and their names.  If u see the commit names- later u can checkout that particular commit | git log --oneline |
| checkout particular commit -- then head will be detached (because at that particular commit, many branches could be there, many branches would have merged at that point) | git checkout <particular-name>  // if u want to checkout latest-1  git checkout HEAD~1 |
| If u want to re-attach head | git switch - |
| Re-attach head-when head is detached then again u have to switch to reattach the head, some other branch –when u checkout a particular commit –then ur head will be detached | git switch master |



Git help

|  |  |
| --- | --- |
| To get imp commands | git help |
| To get all commands | git help –a  Or  git help –all  give down all to go through all the list,  press q to exit |
| If u want to specific command in detail  // here the commands will be shown in console | git <command> help  git init help  Or  git remote help |
| To see the documentation for that command | git help <command name>  git help merge  git help push |

To check for modified files

git status

To see all changes in a tool

Git difftool HEAD

To discard all the changes in the working directory

Git checkout -- <file name>

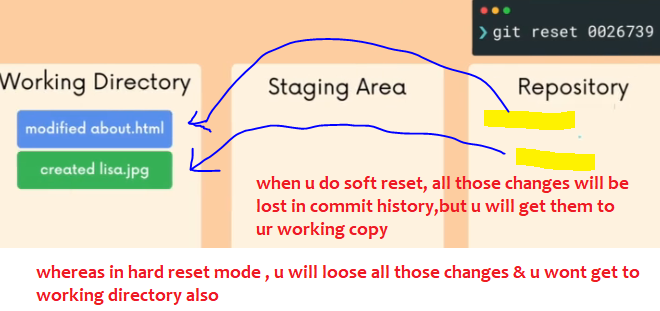
The above commands will work only for staging area,

I mean it will revert those files present in staging area.

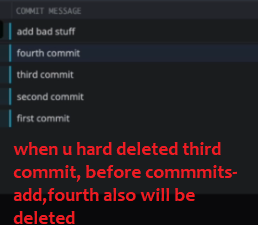
Commit related

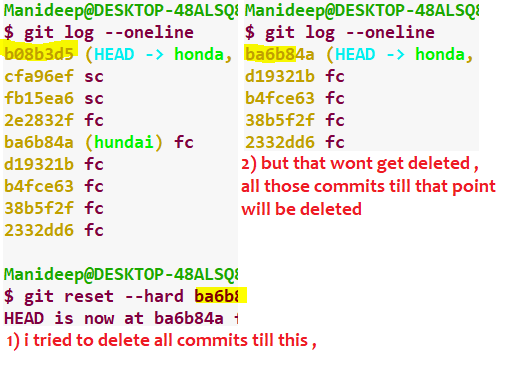
|  |  |
| --- | --- |
| If u want to see the commit history, with commit hashes | git log ---oneline |
| To see what all are staged and un-staged changes | git status |
| To discard all changes made to that file | git restore <filename.txt> (this might work only when file is not in staging area)  --way-1  git checkout -- kiaengine.txt  git checkout –- <filename.txt>  or –way -2  git checkout HEAD <file-name>  git checkout HEAD kiaengine.txt |
| To revert all the local files or reset all local changes | git reset –-hard |
| Add all files to staging area | git add .  // to add single file to stage area  git add <file-name> |
| Commit | git commit –m “fc”  //here –m means commit message  git commit –a –m “fc”  //to add all files present in stage area(not clear) & commit at once |
| Un-staging file | git restore –-staged <file-name>  git restore --staged vehicle.txt  // command says restore the staged file and move back to un-staged area |
| ***If u want to reset that particular commit in soft way*** – when u mistakenly committed then u can clear that commit in remote repo- if u reset those changes will be lost in commit history, but still those changes u will get into ur working directory | git reset <commit-hash>  U can commit hash using git log –oneline  //it removes the commit from log, but it won’t remove the changes from working directory |
| ***Delete the commits and delete the changes from working directory*** | git reset –-hard <commit-hash name>  //if u delete it ,this commit won’t be deleted , till that commit all previous commit will get deleted |

Git soft reset- here commits will be cleaned but changes will still be there



***Delete all commits and delete changes from working directory***





#### Add single file to staging area

//when u are adding means u are adding that file to staging area.

git add < file name abcd.txt >



if u want to pull back the file from staging area and keep in working area then

git restore –-staged <staged-file-name.txt>

#### To check whether file added to stage area or not

It will simply give the status of the branch

If u want to check what file are added to stage area, and which files are not in stage area.

git status

#### Add all files to staging area

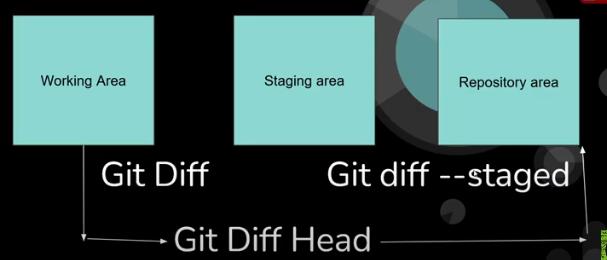
git add .

here “.” Means current directory

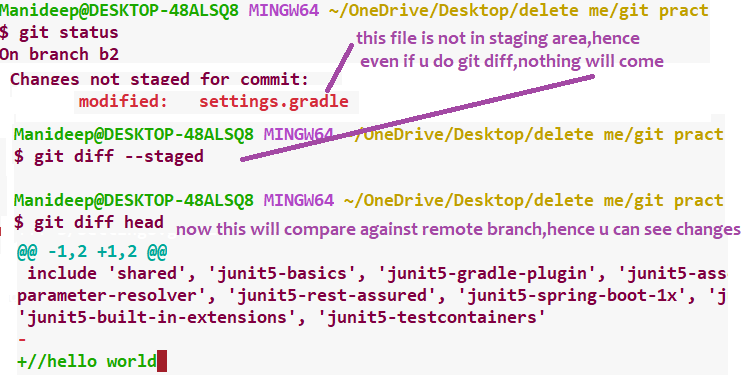
To remove a file from staging area

git restore --staged "1.unstaged file.txt"

#### **Diff**



If that file is not present in staging area, then it will simply it will not show anything



#### **Commit**

|  |  |
| --- | --- |
| To add files into stage area and commit at once | git commit -a -m "commit message"  //here a stands for all , -m stands for commit message  git commit -a -m "sc" |
|  |  |

git commit -m "first commit" or **git commit -m m1**

// here double quotes are not mandatory, “m” stands for commit message

git commit -m “Initital commit message”

it will commit the files only present in staging area.

After git commit, a unique hash is created and the changes are saved.

#### To check whether committed or not

git log

// this will show u the commit history

Push to upstream

Remember when u push, always u should mention the url + branch name

Here origin was an alias name given to the url(while checking out the code, this would have automatically happenned)

Ex:- git remote add origin <git url>

//here with this origin was set as an alias name our git remote url

git push <git remote url> <branch name>

git push <git remote url alias name(usually origin )> <branch name>

git push origin <branch name>

git push origin master

* origin will contain the remote URL
* master is the branch that is pushed (We shall discuss branches later in this course)

To revert all modified files in staging area

git reset –hard

it will revert all the files in staging area only and

It will not impact to any un staged file ,

How to create a git text file using cmd

touch .gitignore.txt

the above will create a text file

to exit from command prompt

:q

To Link local repo to remote repo

* git remote add origin https://github.com/play/repo.git

git remote add origin git@github.com:StephenGrider/docker-react.git git push-u origin master

### Pull fetch

* git pull is the convenient shortcut key to fetch and merge the content.
  + git pull <remote\_name> <branch\_name>

git fetch command downloads the remote content to your local repo, **without changing your code changes**.

*Merge*

* git merge <branch-name> command merges the fetched remote content to the local working tree.

Merges the specified branch code into current branch

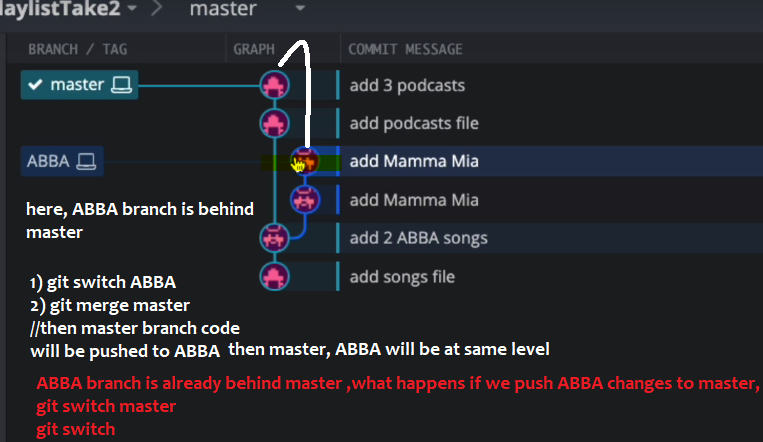
* + git merge <branch\_name> merges the current branch with specified branch.

Ur friends would have committed to the remote branch, if u want to have all those changes locally then use git merge

Ex:- git switch chennai // switches to branch name chennai

git merge <branch name>// means new branch code will be pushed to current branch

git merge hyd // hyd branch code will be pushed to current branch/Chennai



Ex:-

git switch master

git merge bugfix // here bugfix code will be pushed to master branch

Stash

Stash means save something in some secret area.

|  |  |
| --- | --- |
| Create a stash(it will take both staged and un-staged changes & staged changes from working copy and creates a stash) | git stash save |
| Apply the stashed changes to current branch(it will remove the stashed changes and apply to working copy) | git stash pop |
| Apply the stashed changes without removing from stash memory | git stash apply  //when u use git stash pop, the content will be removed from the stashed are and it will be applied to current branch  Whereas when u used git stash apply the changes will not be removed from stash area and will be applied to current working directory |
| To see the list of all stashed list | git stash list |
| Applying a particular stash | git stash apply <stash name>  ex:-git stash apply stash@{0} |
| To drop / delete the stashed version | git stash drop <stashed-name>  git stash drop stash@{3}  // this will delete stashed version present at index 3  // if u are using pop, then stash will be automatically removed from the list, when u used git stash apply then stash won’t be removed |
| To delete all stashed versions | git stash clear |

# Some questions

1. staging area means

only the files moved to staging area will be committed to local git repo

1. see which email ocnfigured

git config user.email