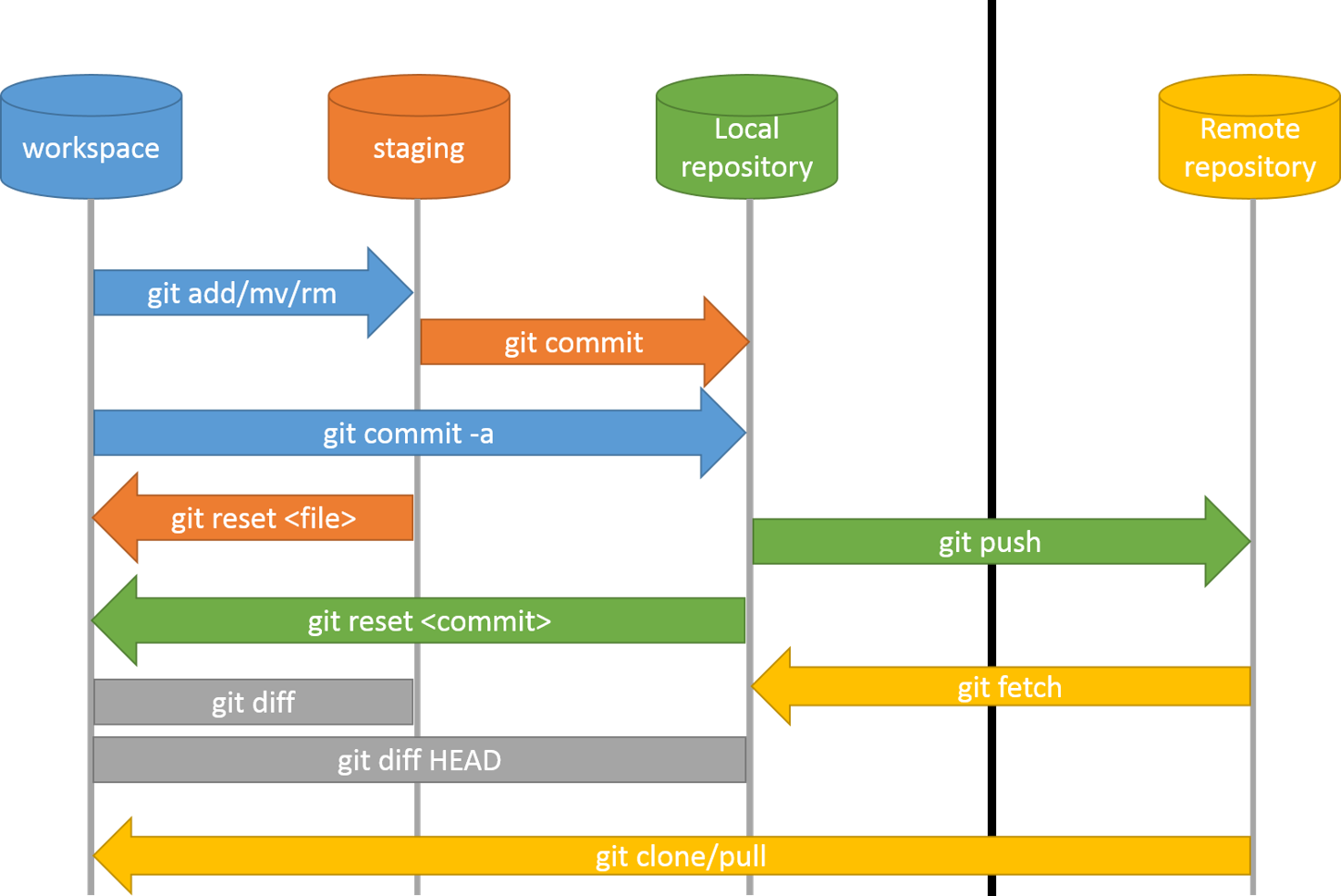
My usual Commands

|  |  |
| --- | --- |
| git init | Initialize git file system in the current directory [a hidden .git folder will be created] |
| git status | Query the changes in the local file system [will show the untracked files] |
| git add . | Add to the current filesystem |
| git add –A . | If you have deleted many files then after git add . git status still shows untracked files then do it its like telling git filesystem to accept the current files at its filesystem forcefully. |
| git commit –m “comment your current state” | Comment the changes you have made |
| git pull origin [branch\_name] | Pull the origin and update your branch [git pull is actually git fetch followed by a git merge] always use git pull |
| git push origin [branch\_name] | Push the current file system to the remote origin |
| git checkout [branch\_name] | Switch to a different branch of your local repo from the existing one |
| git reset - - hard [commitNo] | Go back to a previous commit [go back means absolute go back files changed or added after this commit wont remain in your local repo anymore] |
|  |  |
| git branch –d [branch\_name] | delete the local branch from your machine |
| git push origin –delete [remote branch] | to remove a remote branch from the server |

Following tutorial from <https://try.github.io> and this picture



HEAD is a reference to the last commit in the currently checked-out branch.

|  |  |
| --- | --- |
| git clone https://github.com/hassin23ayz/test.git | Initialize a git repository The repository is a hidden directory where Git operates by Cloning [at first respiratory was created in github] |
| git status | For checking whether repository has changed |
|  | New file is added |
| git status | For checking whether repository has changed |
|  | Untracked files : files list |
| git add . | Add all the untracked files from workspace to the staging area |
| git status | For checking whether repository has changed |
|  | Changes to be committed: files list. The files listed here are in the Staging Area, and they are not in our repository yet. We could add or remove files from the stage before we store them in the repository |
| git commit –m “First file added” | Store the staged changes to the local repo commit all the files in the index |
| git add “\*.txt” | Add many files of the same type from workspace to staging area |
| git log | Prints all the changes we have committed so far |
| git push origin master | Push local repo changes to the origin’s master branch [the name of our remote is origin] |
|  | Now clone the repo at another directory and make change on the file then push it |
| git pull origin master | Pull the remote repo changes in the workspace |
|  | Changes are shown in the command line |
| git diff HEAD | Look what is different between the workspace and the local repo **files**. [i.e. the last commit : where the HEAD is now]  Difference will be checked only between the common files of both the workspace and the HEAD |
|  | no output |
|  | Now firstFile is modified |
| git diff HEAD | Changes are shown TODO: can I sync winmerge here? |
| git add . | Stage the untracked/modified files from workspace to the staging area |
| git diff --staged | Look the difference between the staged area & the local Repo |
|  | Changes are shown as local repo is not yet updated by the staged area |
|  | Now you can unstage also and make staged area files equal to |
| git reset firstFile.txt | Staged is unstaged for this particular file |
| git checkout – firstFile.txt | To discard changes in the working directory |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* winmerge install \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) Open .gitconfig file. It's located at your home directory: c:\users\username\.gitconfig

2) Add the lines below. Pay attention to the single quotes wrapping the path to winmerge:

[diff]

tool = winmerge

[difftool "winmerge"]

cmd = "'C:/Program Files (x86)/WinMerge/WinMergeU.exe'" -e "$LOCAL" "$REMOTE"

<https://coderwall.com/p/76wmzq/winmerge-as-git-difftool-on-windows>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* difference finding \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Make sure winmerge is installed and integrated with git

To check the differences of 1 folder of 2 branches run the following command:

git difftool branch1:folderPath branch2:folderPath

example: git difftool branch\_ayz:comm\_module\_for\_pilot\_project/firmware/commModuleBetaAyzSub old-state:commModuleBetaAyzSub

then winmerge will single handedly pop up options to diff

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*