|  |  |
| --- | --- |
| git fetch (pull + merge)  remotes  git remote add <name> <remote-url>  git remote -v | Pull is (fetch + merge)  -location where you project is hosted (hub/lab  upstream: original project  origin: your forked |
| Peer to Peer (P2P)  Porcelain/Plumbing  Fast Forward | -means each computer can act as a server/client  -decentralized communications model where each party has the same capabilities  High-level/Low-level  -if no additional work on main, Git can perform a FF merge  -like working on the main the entire time |
| Detached HEAD  git checkout <commit> | -when Head is not point to a branch but a commit  Checkout a commit -> make multiple commits-> to save them, create a branch on detached HEAD otherwise GC will delete them |
| Rebase  git pull --rebase  -If someone rebases a shared commits on main  -It will look at the reflog, for each commit to find the common ancestor | -the branch you are on, gets “detached” and put on top of the branch you are git rebase <branch>  -In reality, makes new commits with new hashes because commits are DB objects (immutable)  -GC will remove unreachable commits |
| Rebase vs Tradeoff  \*Never rebase shared commits!  Scenarios:  -clean up your local history before sharing branch  -pull changes from a branch into your branch without performing a merge | -Refactors history to look cleaner but a lie  Merge  -preserve history as exactly as it happened but can look messy  -if conflicts, merge commit would include fixes to the conflict |
| git push -f | -we can lose commits by GC  -history will be conflicting  -doesn’t solve conflicts, just puts the burden on other users |
| git push -u origin <branch>  no -u the second time pushing?  git remote -v  git ls-remote  git fetch origin <branch>  git branch -a  git checkout --track origin/<branch> | -Push the branch to remote and set the origin as upstream branch  (-u) sets the origin as upstream flag  -List remotes and URLs  -Notice the remote branch is not listed  -List branches on remote  -Fetch the branch from remote to local but still need to track  -List both remote-tracking and local branches  -Set up a local branch to track remote branch |
| git merge-base ticket1 main  git rebase -i <use return commit above>  pick <commit>  squash <commit>  …  Then commit message  git rebase main  git checkout <target>  git merge <source>  Example:  git checkout main  git merge ticket1 | -get the original base of ticket1 branch from main  -start the rebase from the commit sha  PICK the first one  Squash the commits you want to combine into a single commit  -pull the current changes from main  Target: refers to branch you want to modify  Source: refers to the branch that has changes you want |
| git branch <branch>  git branch -m <current> <new>  Shortcut: git branch -m <new>  git branch -d <branch>  git branch -D <branch> | -Creates the branch  -Rename a branch  Must switch to a different branch when deleting  -delete a branch  -delete a branch that has not yet merged |
| git diff <branch1> <branch2> <file>  git diff  git diff –cached  git diff HEAD  git diff -w | -Shows committed changes between the tips  -only looks at unstaged changes by default  -see only staged changes  -staged and unstaged changes  -ignore whitespace differences |
| \* ? ! / [a-zA-Z]  \*\*/bin  \*.zip  /bin  git rm --cached <filename>  git rm <filename>  /git/info/exclude | Anything, one character, negator, directory separator, range  -\*\* matches any directory in repository  -\* matches any file in repository  -relative to .gitignore directory  -Delete file from Git repo or  Delete from repo and local filesystem  -Ignore patterns for your system only |
|  |  |
|  |  |
|  |  |
|  |  |