|  |  |  |  |
| --- | --- | --- | --- |
|  | **COMMANDS** |  | **COMMANDS** |
|  | **SETUP** |  | **INSPECT & COMPARE** |
| 1 | git config --global user.name “[firstname lastname]”  *set a name that is identifiable for credit when review version history* | 28 | git log branchB..branchA  *show the commits on branchA that are not on branchB* |
| 2 | git config --global user.email “[valid-email]”  *set an email address that will be associated with each history marker* | 29 | git log --follow [file]  *show the commits that changed file, even across renames* |
| 3 | git config --global color.ui auto  *set automatic command line coloring for Git for easy reviewing* | 30 | git log –summary  *View changes (detailed)* |
|  | **SETUP & INIT** | 31 | git log –oneline  *View changes (briefly)* |
| 4 | git init  *initialize an existing directory as a Git repository* | 32 | git diff branchB...branchA  *show the diff of what is in branchA that is not in branchB* |
| 5 | git clone [url]  *retrieve an entire repository from a hosted location via URL* | 33 | git show [SHA]  *show any object in Git in human-readable format* |
|  | **STAGE & SNAPSHOT** |  | **SHARE & UPDATE** |
| 6 | git status  *show modified files in working directory, staged for your next commit* | 34 | git remote add [alias] [url]  *add a git URL as an alias* |
| 7 | git add –A  *Add all new and changed files to the staging area* | 35 | git fetch [alias]  *fetch down all the branches from that Git remote* |
| 8 | git add [file]  *add a file as it looks now to your next commit (stage)* | 36 | git merge [alias]/[branch]  *merge a remote branch into your current branch to bring it up to date* |
| 9 | git reset [file]  *unstage a file while retaining the changes in working directory* | 37 | git push [alias] [branch]  *Transmit local branch commits to the remote repository branch* |
| 10 | git diff  *diff of what is changed but not staged* | 38 | git push origin [branch name]  *Push a branch to your remote repository* |
| 11 | git diff –staged  *diff of what is staged but not yet commited* | 39 | git push -u origin [branch name]  *Push changes to remote repository (and remember the branch)* |
| 12 | git diff [source branch] [target branch]  *Preview changes before merging* | 40 | git push  *Push changes to remote repository (remembered branch)* |
| 13 | git commit -m “[descriptive message]”  *commit your staged content as a new commit snapshot* | 41 | git pull origin [branch name]  *Pull changes from remote repository* |
|  | **BRANCH & MERGE** | 42 | git remote set-url origin ssh://git@github.com/[username]/[repository-name].git  *Set a repository's origin branch to SSH* |
| 14 | git branch –a  *List all branches (local and remote)* | 43 | git remote add origin ssh://git@github.com/[username]/[repository-name].git  *Add a remote repository* |
| 15 | git branch  *list your branches. a \* will appear next to the currently active branch* | 44 | git commit –amend  *to change your latest log message* |
| 16 | git branch [branch-name]  *create a new branch at the current commit* | 45 | git pull  *fetch and merge any commits from the tracking remote branch* |
| 17 | git branch -d [branch name]  *Delete a branch* |  | **TRACKING PATH CHANGES** |
| 18 | git push origin --delete [branch name]  *Delete a remote branch* | 46 | git rm [file]  *delete the file from project and stage the removal for commit* |
| 19 | git checkout [branch name]  *switch to another branch and check it out into your working directory* | 47 | git rm -r [file-name.txt]  *Remove a file (or folder)* |
| 20 | git checkout -b [branch name]  *Create a new branch and switch to it* | 48 | git mv [existing-path] [new-path]  *change an existing file path and stage the move* |
| 21 | git checkout -b [branch name] origin/[branch name]  *Clone a remote branch and switch to it* | 49 | git log --stat –M  *show all commit logs with indication of any paths that moved* |
| 22 | git branch -m [old branch name] [new branch name]  *Rename a local branch* |  | **REWRITE HISTORY** |
| 23 | git checkout –  *Switch to the branch last checked out* | 50 | git rebase [branch]  *apply any commits of current branch ahead of specified one* |
| 24 | git checkout -- [file-name.txt]  *Discard changes to a file* | 51 | git reset --hard [commit]  *clear staging area, rewrite working tree from specified commit* |
| 25 | git merge [branch]  *merge the specified branch’s history into the current one* |  | **TEMPORARY COMMITS** |
| 26 | git merge [source branch] [target branch]  *Merge a branch into a target branch* | 52 | git stash  *Save modified and staged changes* |
| 27 | git log  *show all commits in the current branch’s history* | 53 | git stash list  *list stack-order of stashed file changes* |
| 54 | git blame <filename>  *Show file line numbers annotated with author information can also see in git log* | 58 | git reset --soft HEAD^  *Undo the last commit, leaving changes in the index.* |
| 55 | git rebase -i HEAD~n  *command to display a list of the last n commits in your default text editor.* |  |  |
| 56 | git push --force *example-branch*  *command to force push over the old commit.* |  |  |
| 57 | git merge –abort  *stop the merge and return everything to its pre-merge state.* |  |  |