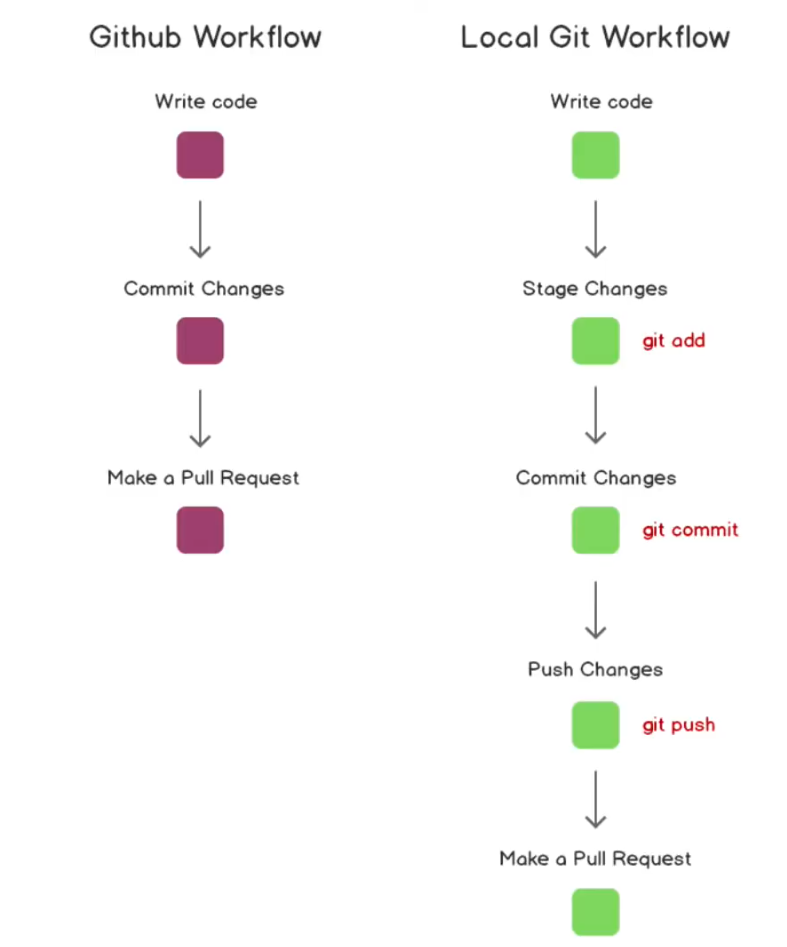
****

**CREDIT : FREECODECAMP.ORG**

**1-Create a GitHub account**

**A place to host your repository.**

**2-Install Git**

[**https://git-scm.com/downloads**](https://git-scm.com/downloads)

**3-Confirm Your Git Installation**

**git –version : learn your git version and if it is installed**

**git status : check status**

**4-Configure Your User Name and E-Mail**

**git config --global user.name "GitHubUserName"**

**git config --global user.email "GitHubMail"**

**git status : check status**

**\*extras**

**--system : applies to all users on the computer**

**--global : applies to only apply to your user**

**--local : applies to only apply to repository**

**git config –list : check the automatic settings**

**5-Clone a Repository**

**git clone** [**https://github.com/YourGitHubName/NameOfTheRepo**](https://github.com/YourGitHubName/NameOfTheRepo)

**: cloning the repository inside a folder**

**6-Change Directory**

**cd NameOfTheFolder**

**git status : check after cd command**

**7-Branching**

**git branch : list of local branches**

**git branch –all : list of all branches**

**git branch -a : list of all branches**

**git branch NewBranchName : create a new branch**

**git status : check status**

**\*Creating branches using SHA-1 codes of commit pointers**

**git log : shows SHA-1 codes**

**git branch “NewBrachName” SHA-1 :**

**first 6 digits of the SHA-1 or all of it**

**\*Create branches from other branches**

**git checkout -b feature2 feature**

**\*Switching to a branch that does not appear automatically**

**git pull origin NotAppearingBranch**

**git branch : the branch does not show up**

**git checkout NotAppearingBranch**

**git branch : the branch does shows up**

**git branch -d BranchName : delete branch**

**8-Checkout**

**git checkout TheNameOfTheBranchYouWantToJoin : switching branches**

**git status : check status**

**git checkout -b JustCreated : create brunch while checking out**

**9-Working Tree**

**untracked : use git add to track the file on Git**

**modified : the file has had changes**

**staged : file “add” ed to Git with “git add”**

**committed : “git commit -m” ,** **SHA-1 pointer(snapshot) created**

**\*Working Tree can be main branch. Generally you want to create a “features” branch and work there. Later you can merge these branches into main.**

**10-Committing Changes**

**git status : status check of working tree**

**git add FileName.FileFormat : add files from Working Tree to Staging Area**

**git status : check if the changes applied**

**git commit -m “Comment” : SHA-1 pointer(snapshot) created**

**git status : one last check**

**11-Pushing(Uploading) Commits(Changes)**

**\*Changes are done to the files we Pulled (downloaded) from the repository earlier. Now it is time to Push (Code on the computer now becomes the repository on GitHub) changes to the repository.**

**git push –set-upstream origin NameOfTheRepo: “git push -u origin” is the same**

**upload the code and set origin, after this, we can shortly write git push**

**git push : short version**

**\*Make git remember your credentials:**

**Windows: git config --global credential.helper wincred**

**Mac: git config --global credential.helper osxkeychain**

**12-Merging**

**main or master : the main branch**

**features : the branch of main which has our commits**

**git checkout main : main or the name you gave it to**

**git status : check out the status**

**git merge features : features branch is merged with the branch we are on**

**git status: check status**

**14-Update Your Local Repository**

**git status : check status**

**git checkout main : checkout to master, after pull, it will be updated**

**git pull : pull repository, combination of commands(fetch and merge), updates**

**.md Shortcuts**

**# : used for a Header**

**\* or - : followed by a space, creates a bulleted list**

**\*\* BOLD TEXT \*\* : texts between the asterix are bold**

**- [ ] : creates a checklist**

**@mention : the way to mention someone**

**:emoji: : creates emojis, etc. smiley**

**Short Notes**

**\*git does what you command it to, no automatic action**

**\*README.md : Markdown Documentation. Basically, a user guide and explanation.**

**\*git config user.email "YourGitHubEmail" : when using another computer without a personal account. No –global in the code.**

**\*git log : shows SHA-1 code.**

**\*git branch –merged : see safe branches to delete**

**\*git pull –prune : delete remote tracking branches**

**\*git config –global fetch.prune true : pull request with deleting branches**

**\*creating aliases:**

**git config –global alias.cob “checkout -b”**

**git cob**

**\*mkdir NewDirectoryName : create new direction**

**\*clean branches:**

**git config alias.dlb '!git checkout <DEFAULT-BRANCH> && git pull --prune && git branch --merged | grep -v "\\* " | xargs -n 1 git branch -d'**

**\*merge conflics are <<<<<<<**, ======= , >>>>>>> **between these symbols**

**\*git init : initializes local repository in your pc**

**\*touch FileName.File : creates the file**

**\*HEAD : most recent commit on your branch**

**\*staging area : also called index**

**\*working tree : modified files, changes**

**Git Dictionary**

**\*git config –global user.name “GitHubUserName”**

**git config –global user.name : check name**

**\*git config –global user.email “GitHubEmail”**

**git config –global user.mail : check mail**

**\*git init project1 : creates a folder and initializes on Git**

**cd project1 : enters inside that folder, Change Direstory**

**\*touch TheFile.Extension :**

**creates file in the directory you are on**

**git status : check status**

**\*git add TheFile.Extension : adds file to staging area**

**git status : check status**

**git log : shows SHA-1 codes**

**\*git commit : commits changes and creates a pointer(SHA-1)**

**git status :**

**git log :**

**\*git push –set-upstream origin NameOfTheRepo: push changes**

**git push –u origin NameOfTheRepo : same and sets upstream so**

**git push : short version can be used later**

**\*git pull –prune : deletes branches while pulling**

\***git diff : shows changes working tree vs staging area(git add)**

**git diff –staged : staged vs commit**

**git diff HEAD : working tree vs last commit(HEAD)**

**git diff –color-words : colorizes words**

**git diff –word-diff : colorizes words**

**git diff –stat**

**\*git log –oneline : no SHA-1 ref number**

**git log –stat : shows added files**

**git log –patch : shows path**

**git log –patch –oneline : patch and oneline together**

**git log –graph : shows graph**

**git log –graph – all –decorate –oneline : shows better**

**\*git rm File.Ext : removes file**

**git add -u . : add deleted files as deleted**

**git rm –cached File.Ext : subtracts file from version control sys, not deleting**

**git mv header.jpg source/header.jpg : move file from current director to source**

**git add -A . : does the moving on git**

**\*git log –stat -M –follow – NameOfTheFile : check file history**

**\*git ignore**

**touch .gitignore**

**git add .gitignore**

**git commit -m “Ignore temporary files.”**

**(inside .gitignore)**

**.sass-cache : file to ignore**

**\*.log**

**DirectoryName/**

**SubDirectory\***

**git ls-files –others –ignored –exclude-standard : files inside .gitignore**

**\*branching**

**git branch BranchName : creates branch**

**git branch -d BranchName : for deleting a branch**

**git checkout BranchName : switching branches**

**git branch : shows our branch with \* and other branches**

**git status**

**\*checkout**

**git checkout SHA-1Code : check out to a commit**

**git status**

**git checkout – FileName.Ext : changes file to last commit**

**git checkout -b CheckingOutWhileBranching**

**\*merge**

**git checkout main**

**git branch : check and control branch name**

**git merge the BranchYouWant : merge the branch to main**

**\*merge conflict**

**<<<<<< HEAD**

**code on the HEAD : last commit on your branch**

**=======**

**code on the merged branch**

**>>>>>>>>**

**git add ConflictedFile : stage the file**

**git commit -m “Conflict solved”**

**git merge –abort : last commit of you current branch solves conflict**

**git merge –squash BranchName :**

**new commit with all changes happened on target branch**

**git branch -d BranchName : deleting unnecessary branch**

**\*networking**

**git remote add origin URL : url of the destination**

**git remote set-url origin NewURL : change remotes url**

**git remote -v : shows URLs**

**git branch -r : shows origin**

**git fetch origin : brings like pull**

**git pull origin : brings and merges**

**git push origin : pushes to git hub**

**\*reset**

**git reset HEAD : from staging are to working tree(undoes git add)**

**git reset –soft HEAD~5 : soft reset last 5 commit**

**git commit -m “Five new changes” : collect them in one commit**

**git reset –hard HEAD~3 : hard reset last 3 commit(pointer SHA-1 numbers)**

**git checkout 6DigitSHA-1 FileName.Extension:**

**takes the file from the commit you want**

**git reset : resets last “add” done to staging area**

**git reset FileName.Extension : resets the added file**

**git reset HEAD~1 : resets to last commit**

**git reset SHA-1Code : resets to a specific pointer**

**git reset –hard SHA-1Code : hard reset to pointer**

**\*git reflog**

**cd .git/logs : see hidden files**

**git reflog**

**git reset –hard 6DigitSHA-1 : reset to backup from reflog**

**\*rebase**

**git checkout TheFeatureYouWantToRebase**

**git rebase main**

**Local Git Workflow**

**1-Create a GitHub account**

**2-Download Git**

**git –version : check git version**

**\*to make it easy for you. select main as default branch name.**

**3-Configure your account**

**git config --global user.name "GitHubUserName"**

**git config --global user.name : check name**

**git config --global user.email "GitHubMail"**

**git config --global user.email : check email**

**4-Open Terminal or Git Bash**

**5-Initialize a directory(folder) to work**

**git init DirectoryName : creates directory and initializes**

**6-Change your directory**

**cd DirectoryName**

**7-See the files inside (empty if you did not put any file inside)**

**ls & ls -a : list files**

**git status : status check**

**8-Create and checkout to new branch to work on**

**\*If you don’t have any commits on main branch yet, when you check out to another branch, main will be lost. Because there isn’t any difference(commit) between main and your branch.**

**\*To overcome this issue; create a file. For example, README.md. git add . , git commit -m “Added README.md”. git status. git log.**

**\*So you are on main branch, you have your first commit, now you can create a new branch without losing the “main” branch.**

**\*Work on your new branch. Create a commit after words.**

**\*When you check out to main you will see the changes disappear because changes are on another branch.**

**git checkout -b BranchName**

**git branch : see the branch you are on(\*)**

**9-Add files from Working Tree to Staging Area**

**touch example.txt : touch command creates file**

**git add : adds files from Working Tree to Staging Area**

**git add . : adds all, git add example.txt : only adds emaple.txt to staging area**

**git status : check status (new file axample.txt)**

**10-Commit changes from staging area (thus create new HEAD)**

**git commit -m “Your comment”**

**git status : (nothing to commit, working tree clean)**

**\*changes stayed at the branch. no change on the main yet.**

**11-Checkout to main (or master)**

**git checkout main : if there is no main, git checkout -b main**

**git branch : see branches. \* is the branch you are on**

**\*clear : cleans Git history**

**12- Add origin**

**git remote add origin** [**https://github.com/YourGitHubName/NameOfTheRepo**](https://github.com/YourGitHubName/NameOfTheRepo)

**git remote -v : shows remote repository names(fetch-pull, push)**

**13-Upload (Push) to GitHub repository**

**git push -u origin main : -u is for upstream. git push will be enough for future.**

**\*Commits pushed to GitHub repository.**

**git push : we have set upstream, now we can push using this shortcut.**

**\*You must push branches separately.**

**\*Now you need to update your main before pushing your branches (If you work with other). After you update your main with git pull (while on main branch), you can checkout to your branch and you can merge updated main branch to your branch(committing your work branch before merge is a good idea).**

**14-Pull changes to the main branch**

**git checkout main : check out to main**

**git status: status check**

**git pull : pull the latest changes to the main branch**

**15-Merge if the work is done**

**\*You can merge main to your branch if you work with others to see if changes to main affects your work, before moving forward. If you work solo. you can merge your branch to main.**

**git checkout main : check out to main**

**git status: status check**

**git pull : pull the latest changes to the main branch**

**git branch : see branches**

**git merge example : merge example branch to main**

**\* rm : removes files**

**\*git branch -d BranchName : deletes branch on git**

**\*git push origin --delete NameOfTheBranch : deletes branch on GitHub**

**Git Clone Workflow**

**1-Create a director and clone the repository inside.**

**\*Same steps for first usage and setting up. (1-8)**

**git clone** [**https://github.com/YourGitHubName/NameOfTheRepo**](https://github.com/YourGitHubName/NameOfTheRepo)

**: clone the repository inside a new folder**

**2-Change your directory to the repository directory**

**cd DirectoryName**

**3-Create a new branch**

**\*Different than the first set up of local git flow. Main branch does not dis appear because you already have a commit before.**

**git branch : check branches**

**git log : see commits that have been created before**

**\*you have to push branches separately!**

**4-Make changes and commit**

**git add : or git add . , add from Working Tree to Staging Area**

**git commit -m “Your Comment” -m “Your comment description”**

**: commit your changes**

**5-Pushing changes**

**git push -u origin NameOfTheBranchYouAreOn**

**: pushes to remote repository and sets it as upstream**

**\*Since we have set an upstream, we can use the short version now (git push).**

**6-Pull request**

**git checkout main : or the branch you want**

**git pull : pull the latest changes to the repository**

**\*git branch -d BranchName : delete the branch if you are done with it**

**\*git push origin --delete NameOfTheBranch : deletes branch on GitHub**

**\*You don’t want to get behind the main branch as you go. So, you create a pull request while on main. After the pull request, create a new branch and merge main to your new branch to check for any conflict.**

**Sources:**

1. [**https://git-scm.com/doc**](https://git-scm.com/doc)
2. [**https://training.github.com/downloads/tr/github-git-cheat-sheet/**](https://training.github.com/downloads/tr/github-git-cheat-sheet/)
3. [**https://training.github.com/**](https://training.github.com/)
4. [**https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent**](https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)
5. [**https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-email-preferences/setting-your-commit-email-address**](https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/managing-email-preferences/setting-your-commit-email-address)
6. [**https://guides.github.com/introduction/flow/**](https://guides.github.com/introduction/flow/)
7. [**https://www.youtube.com/c/GitHubGuides/playlists**](https://www.youtube.com/c/GitHubGuides/playlists)
8. [**https://www.varonis.com/blog/git-branching/**](https://www.varonis.com/blog/git-branching/)