# **Git Instructions**

|  |  |
| --- | --- |
| **Setting up Git** | |
| **How to check what directory you are in: pwd** | **How to** **create a new folder:** mkdir git-fast |
| **How to go back one step out of a folder/directory: cd ..** | **How to delete a folder/directory:** rm -rf demo-merge |
| **How to clear the screen: clear** | **How to check configurations like name, email etc:** vi ~/.gitconfig |
| **How to jump to different directory:**  cd git-fast | **How to create a username and e-mail:** git config --global user.email "conor.nugent9@gmail.com"  git config --global user.name "Conor Nugent" |
| **Notes:** A file is known as untracked if it has not been git added yet | We must git init each new directory after copying it in. So MyRepoFromExistingSource must be initialized even though it is contained within the already initialized git-fast directory |
| **Working with fFles & Folders in Git** | |
| **How to create a text file:** vi/subl <filename> | **How to view contents of a file:** cat .gitconfig |
| **Initializing an empty repository**: git init | **List all objects: Is-al / ls** |
| **Create file and contents in one go without the editor:** echo "this is my first file in empty repository" >> firstFileUsingEcho.txt | **Copying a file to folder:** cp < f/Bibhash/Downloads/ filename> for example  cp /c/Users/conor/Downloads/initializr-verekia-4.0.zip . |
| **Unzipping a folder: Unzip <filename>** | **Removing a copied file:** rm initializr-verekia-4.0.zip |
| **To fork another git repository to our local machine: git clone <addressofrepository> while in the directory you want the files to be forked into** | **When you see # after the master, it indicates you have just git initialized the folder. When you see the + symbol, it means you have just added files to the staging area. The absence of any symbol indicates a clean working directory. The star symbol indicates there is a modification in the working directory.** |
| **Renaming a file: mv originalname newname** | **Steps to create version control for pre-existing file: 1. Copy in file 2. Unzip it 3. Delete original folder 4. Rename unzipped folder 5. Initialize this folder** |
| **Moving a file from one directory to another: 1. Log into folder containing file/folder 2. Type “git mv filename\_to\_move directory\_name”** | **How to check repo status: git status** |
| **How to list configurations for current user:** git config --global --list | **Getting general help:** git help |
| **Access various git guides: Git help -g** | **Access git glossary: git help glossary** |
| **Access help for a particular command: Git help init** | **How to exit vi editor: Press “:” and then “q” or “wq”** |
| **Copying a github repository to your directory: 1. Cd into the folder you want 2. git clone <addressofrepository> 3. Type ls -l to confirm everything worked** | **Add your change from working directory to staging area: git add <filename> or git add . for multiple files** |
| **Removing file from staging area back to working directory: git rm –cached <filename>** | **Committing changes from staging area to repository for good: git commit -m “our first commit”** |
| **Adding and committing a file: 1. Git status 2. Git add <filename> 3. Git commit -m “ our first commit in this course”** | **Check commit history: Either git log (for entire history) or git log –oneline (condensed view)** |
| **Only show commits for a specific file: git log <filename>** | **Show only commits that occur between <since> and <until>:** <since> or <until> can be a branch, name, head etc : git log <since>..<until> |
| **Branching** | |
| **To see all the branches: git branch -a** | **To create a branch: git branch <branchname>** |
| **How to switch to a branch and thus make it active: git checkout <branchname>**  **Note: A demo branch will inherit the commit history of the master branch but commits to this demobranch will not go back to the master** | **To rename a branch: git branch -m <original name><new name>** |
| **To delete a branch: git branch -d <name offeature>** | **Create a branch that contains all the same contents as a specific branch (as opposed to the master branch: git checkout -b <branchtocreate> <branchtocopy>** |
| **Checkout** | |
| **Note: Can be used to revert back to the status of say commit 3, if all the changes after commit 3 were not working correctly and you want to alter things: git checkout <commit number>**  **Note: Whatever changes you make here, do not affect the rest of the project. This is because you are working from a detached head state.** | **To go back to previous (master) commit state: git checkout master** |
| **To make any changes you made in the detached head state permanent, you have to create a newbranch out of these changes: git checkout -b <newbranch>** | **To change previous commit state to current state, for good: git checkout <commitnumber> filename** |
| **To undo this change: git checkout HEAD filename** | **To confirm this commit: git commit -m “message”** |
| **Remove file from staging area but do not undo change made to file: git reset filename** | **Remove file from staging area and undo any changes you made to these files: git reset --hard** |
| **To show differences between head and a particular commit point: git difftool 2342343 HEAD or git difftool HEAD~1 HEAD or git difftool HEAD^ HEAD** | **How to show the differences for a particular file: git difftool --major-milestones** |
| **Reset to a particular commit point: git reset 23423423** | **Reset to a particular commit point and destroy anything after that point: git reset --hard 23423423** |
| **Show which files are going to be removed without actually doing it: git clean -n** | **Remove untracked file: git clean -f** |
| **Checkout – creates a new temporary branch, with all the contents of the HEAD copied to this branch. Does not move the head.** | **To edit the git config file: git config –global --edit** |
| **Delete all files that have not been added to the staging area (including .gitignore): git clean -xf** | **Push a repository to GitHub: git push origin master** |
| **Pull a repository from Github: git pull master origin** | **To add a group of files to a git repository: 1. Copy address of repository onto clipboard 2. git remote add origin “**<https://github.com/josemourinho/business-venture.git>**” 3. To push the files after establishing the above connection “git push origin master”** |
| **To display differences between feature branch and master branch: git difftool feature\_branch master** | **To display differences between local and remote repositories: git diff master origin/master** |
| **To Set up a Public and Private Key** | |
| **To generate an ssh key: ssh-keygen -t rsa -b 2048 - C "home machine"** | **Start the ssh agent in the background: eval "$(ssh-agent -s)"** |
| **Add your SSH key to the ssh-agent :** ssh-add ~/.ssh/id\_rsa.pub | **Verify SSH connection; here 'git" is the user name; note that you cannot use your github user name such as 'william':** ssh -T git@github.com |
| **To set connection to password:** git remote set-url origin https://github.com/josemourinho/sample-java-project.git | **To set connection to ssh: git remote set-url origin https://github.com/josemourinho/sample-java-project.git** |
| **Set/Change an ssh password for the private key: ssh-keygen -p** | **To check what kind of connection we have (password or ssh): git remote -v** |
| **Checking our GitHub SSH Key: $ssh -T git@github.com** | **Copying the private key to clipboard: $clip < ~/.ssh/id\_rsa.pub** |
| **Checking if SSH keys are already installed: $ls -l ~/.ssh** |  |
| **Installing Sublime as the Default Text Editor** | |
| **Installing and Configuring text editor (Sublime Text) for Git on Windows: git config --global core.editor "'c:/program files/sublime text 3/subl.exe' -w"** | **To display the difference between the HEAD (last commit) and the commit before that: git diff <second last commit number> HEAD** |
| **Diff and Merge Tool for Git** | |
| **To display difference between working directory and staging area/commit area etc: git diff or git difftool (for visual interface)** | **To display the differences that have occurred between one commit and another commit: gitdiff tool <one commit numbder> <another commit number>** |
| **To display the changes is a single file: git diff --<name of file>** | **To set difftool to p4merge: First type “git config –global diff.tool p4merge” and then set path with “git config –global difftool.p4merge.path “C:/Program Files/Perforce/p4merge.exe”** |
| **Prevent needing to launch confirmation each time launch difftool: git config –global difftool.prompt false** | **To set mergetool to p4merge: First type “git config –global merge.tool p4merge” and then set path with “git config –global mergetool.p4merge.path “C:/Program Files/Perforce/p4merge.exe”** |
| **Prevent needing to launch confirmation each time launch mergetool: git config –global mergetool.prompt false** |  |
| **Editing GitBash Interface** | |
| **Change Conor@desktop prompt to hello etc: PS1 ="Hello! \W $ "**  **Note: This must be placed in the vi .bash\_profile file as “source . ~/bashrc” to be permanent** | **Access bashrc: vi ~/.bashrc** |
| **Access Source bashrc: source ~/.bashrc** | **Final code to be typed into bashrc:**  **GIT\_PS1\_SHOWDIRTYSTATE=1**  **export PS1 = ‘\[\e[1;32m\]Hello! You are in \[\e[1;33m\]\W\[\e[1;34m\]$(\_\_git\_ps1)’$’\[\e[1;35m\]\n$\[\e[0m\]’** |
| **Git Log In-Depth** | |
| **Git log --oneline --decorate --graph –all: The graph function here depicts the branches present. Decorate will show up the branch names. When you do not include all, git filters the info that is shown.** | **Display which files were altered and the number of lines that were added and deleted from them: Git log –stat** |
| **Display a patch representing each commit in great detail: Git log -p** | **Show the history for a particular directory: Git log <nameoffolder>** |
| **Searches for all the commits that contain a particular keyword: Git log –oneline grep=”searchtopic”:** |  |
| **Git Branching and Merging Techniques** | |
| **To carry out a three way merge: git merge 3wm-branch -m "our first 3-way merge"** | **To merge a branch into the master: git merge <name of branch you wish to be merged into the master>** |
| **To stop the generation of any temporary merge files: git config --global mergetool.keepBackup false** | **To deal with merge conflicts: git mergetool** |
| **Commit History Rewriting** | |
| **How to add a new file/change to the previous commit but keep the old commit name in place:** git commit –amend –no-edit | **How to add to a previous commit and change the commit name: git commit --amend** |
| **To stop rebasing: git rebase --abort** | **To bring local repository up to date with remote repository: git fetch <name of repository>** |
| **To pull smaller changes from the remote repository (and prevent your local repository becoming too cluttered): git fetch** | **To create a remote repository that reflects a directory on your local machine: Follow the steps in video “Git pull with rebase – Part 1” or 1. Create a repository on github that is called the same as your local folder 2. While in the local folder in git bash, type in “git remote add origin** [**https://github.com/josemourinho/demo-pull-rebase-2.git**](https://github.com/josemourinho/demo-pull-rebase-2.git)**” 3. Then type “git push -u origin master”. Now you have created a connection between your local folder and the remote repository.** |
| **Git pull without rebase, is equivalent to fetching changes from the remote repository and merging in these changes.** | **It is ok to use this if we are pulling in changes from a large feature, but if we are only importing small changes using git pull, then our local commit history might get cluttered with a lot of merges. When pulling in small changes like this it is better to do a git pull with rebase** |
| **To view a certain point in the reflog: git show HEAD@{relevantnumber}** | **To see a branch at one hour ago: git show master@{2.days.ago}** |
| **Use reflog to revert to a previous state: 1. Git reflog 2. Git checkout HEAD@{1} 3. Git checkout -b restore-branch** |  |
| **Git Tagging** | |
| **To create a tag: git tag <nameoftag>**  **Note: https://stackoverflow.com/questions/11514075/what-is-the-difference-between-an-annotated-and-unannotated-tag** | **To list tags: git tag or git tag --list** |
| **To make an annotated tag: git tag -a v-1.9-rc2 -m "release version 1.9"** | **To review the details related to a tag: git show v-1.9.1** |
| **To search for a tag: git tag -l "v-1.9\*"** | **To see the difference between two tags:** **git difftool v-3.1.0-alpha v-3.1.0-beta** |
| **How to associate a tag with a commit ID: git tag -a v-3.1.0-beta --force e52f2ea** | **How to delete a tag: git tag v-1.8-rc1 --delete** |
| **How to push tags to a local repository: git push origin --tags** | **How to delete a tag from remote repository: git push origin :v-1.9-rc2**  **Note: To delete multiple tags just name them all after the :** |
| **How to push all annotated tags to remote repository: git push origin --follow-tags**  **Note: Lightweight tags should be kept for local development while annotated tags can be pushed to the remote server. This avoids tag clashes** | **To push annotated tags to remote repository by default: git config --global push.followTags true** |
| **To convert a lightweight tag to an annotated tag: git tag -a -f <tagname> <tagname>** |  |
| **Git Stash** | |
| **To save things temporarily without committing them to the remote repository: git stash** | **To stash save with a message attached for easy identification: git stash save “message”** |
| **To show what file a stash is associated with (in case you forgot to save it with a message previously): git stash show <stashname>** | **To stash untracked files: git stash -u** |
| **To drop a stash: Git stash drop <name of stash>** | **To reactivate/apply a stash: git stash apply** |
| **To erase all stashes: git stash clear** | **To stash both tracked and untracked files: git stash -u** |
| **To transfer (and subsequently delete) stashes from master to a branch (demobranch): git stash branch demobranch** |  |

**Git Cheatsheet**

**Reminder**

Press minus + shift + s and return to chop/fold long lines!

Show folder content: ls -la

**Notes**

Do not put (external) dependencies in version control!

**Setup**

See where Git is located: which git

Get the version of Git: git --version

Create an alias (shortcut) for git status: git config --global alias.st status

**Help**

Help: git help

**General**

Initialize Git: git init

Get everything ready to commit: git add .

Get custom file ready to commit: git add index.html

Commit changes: git commit -m "Message"

Add and commit in one step: git commit -am "Message"

Remove files from Git: git rm index.html

Update all changes: git add -u

Remove file but do not track anymore: git rm --cached index.html

Move or rename files: git mv index.html dir/index\_new.html

Undo modifications (restore files from latest commited version): git checkout -- index.html

Restore file from a custom commit (in current branch): git checkout 6eb715d -- index.html

**Reset**

Go back to commit: git revert 073791e7dd71b90daa853b2c5acc2c925f02dbc6

Soft reset (move HEAD only; neither staging nor working dir is changed): git reset --soft 073791e7dd71b90daa853b2c5acc2c925f02dbc6

Undo latest commit: git reset --soft HEAD~

Mixed reset (move HEAD and change staging to match repo; does not affect working dir): git reset --mixed 073791e7dd71b90daa853b2c5acc2c925f02dbc6

Hard reset (move HEAD and change staging dir and working dir to match repo): git reset --hard 073791e7dd71b90daa853b2c5acc2c925f02dbc6

**Update & Delete**

Test-Delete untracked files: git clean -n

Delete untracked files (not staging): git clean -f

Unstage (undo adds): git reset HEAD index.html

Commit to most recent commit: git commit --amend -m "Message"

Update most recent commit message: git commit --amend -m "New Message"

**Branch**

Show branches: git branch

Create branch: git branch branchname

Change to branch: git checkout branchname

Create and change to new branch: git checkout -b branchname

Rename branch: git branch -m branchname new\_branchname or: git branch --move branchname new\_branchname

Show all completely merged branches with current branch: git branch --merged

Delete merged branch (only possible if not HEAD): git branch -d branchname or: git branch --delete branchname

Delete not merged branch: git branch -D branch\_to\_delete

**Merge**

True merge (fast forward): git merge branchname

Merge to master (only if fast forward): git merge --ff-only branchname

Merge to master (force a new commit): git merge --no-ff branchname

Stop merge (in case of conflicts): git merge --abort

Stop merge (in case of conflicts): git reset --merge // prior to v1.7.4

Merge only one specific commit: git cherry-pick 073791e7

**Stash**

Put in stash: git stash save "Message"

Show stash: git stash list

Show stash stats: git stash show stash@{0}

Show stash changes: git stash show -p stash@{0}

Use custom stash item and drop it: git stash pop stash@{0}

Use custom stash item and do not drop it: git stash apply stash@{0}

Delete custom stash item: git stash drop stash@{0}

Delete complete stash: git stash clear

**Gitignore & Gitkeep**

About: <https://help.github.com/articles/ignoring-files>

Useful templates: <https://github.com/github/gitignore>

Add or edit gitignore: nano .gitignore

Track empty dir: touch dir/.gitkeep

**Log**

Show commits: git log

Show oneline-summary of commits: git log --oneline

Show oneline-summary of commits with full SHA-1: git log --format=oneline

Show oneline-summary of the last three commits: git log --oneline -3

Show only custom commits: git log --author="Sven" git log --grep="Message" git log --until=2013-01-01 git log --since=2013-01-01

Show only custom data of commit: git log --format=short git log --format=full git log --format=fuller git log --format=email git log --format=raw

Show changes: git log -p

Show every commit since special commit for custom file only: git log 6eb715d.. index.html

Show changes of every commit since special commit for custom file only: git log -p 6eb715d.. index.html

Show stats and summary of commits: git log --stat --summary

Show history of commits as graph: git log --graph

Show history of commits as graph-summary: git log --oneline --graph --all --decorate

**Compare**

Compare modified files: git diff

Compare modified files and highlight changes only: git diff --color-words index.html

Compare modified files within the staging area: git diff --staged

Compare branches: git diff master..branchname

Compare branches like above: git diff --color-words master..branchname^

Compare commits: git diff 6eb715d git diff 6eb715d..HEAD git diff 6eb715d..537a09f

Compare commits of file: git diff 6eb715d index.html git diff 6eb715d..537a09f index.html

Compare without caring about spaces: git diff -b 6eb715d..HEAD or: git diff --ignore-space-change 6eb715d..HEAD

Compare without caring about all spaces: git diff -w 6eb715d..HEAD or: git diff --ignore-all-space 6eb715d..HEAD

Useful comparings: git diff --stat --summary 6eb715d..HEAD

Blame: git blame -L10,+1 index.html

**Releases & Version Tags**

Show all released versions: git tag

Show all released versions with comments: git tag -l -n1

Create release version: git tag v1.0.0

Create release version with comment: git tag -a v1.0.0 -m 'Message'

Checkout a specific release version: git checkout v1.0.0

**Collaborate**

Show remote: git remote

Show remote details: git remote -v

Add remote origin from GitHub project: git remote add origin https://github.com/user/project.git

Add remote origin from existing empty project on server: git remote add origin ssh://root@123.123.123.123/path/to/repository/.git

Remove origin: git remote rm origin

Show remote branches: git branch -r

Show all branches: git branch -a

Compare: git diff origin/master..master

Push (set default with -u): git push -u origin master

Push to default: git push origin master

Fetch: git fetch origin

Fetch a custom branch: git fetch origin branchname:local\_branchname

Pull: git pull

Pull specific branch: git pull origin branchname

Merge fetched commits: git merge origin/master

Clone to localhost: git clone https://github.com/user/project.git or: git clone ssh://user@domain.com/~/dir/.git

Clone to localhost folder: git clone https://github.com/user/project.git ~/dir/folder

Clone specific branch to localhost: git clone -b branchname https://github.com/user/project.git

Delete remote branch (push nothing): git push origin :branchname or: git push origin --delete branchname

**Archive**

Create a zip-archive: git archive --format zip --output filename.zip master

Export/write custom log to a file: git log --author=sven --all > log.txt

**Troubleshooting**

Ignore files that have already been committed to a Git repository: <http://stackoverflow.com/a/1139797/1815847>

**Security**

Hide Git on the web via .htaccess: RedirectMatch 404 /\.git (more info here: <http://stackoverflow.com/a/17916515/1815847>)

**Large File Storage**

Website: <https://git-lfs.github.com/>

Install: brew install git-lfs

Track \*.psd files: git lfs track "\*.psd" (init, add, commit and push as written above)