**Git Cheat sheet**

git explained best by this guy <https://www.youtube.com/playlist?list=PLu0W_9lII9agwhy658ZPA0MTStKUJTWPi> :)

**Adding credentials**

* **git config --global user.name <user\_name> ← adds username**
* **git config --global user.email <user\_email> ← adds email id**

**local - only within that repository**

**global - across all repositories within the linux user**

**system - across all repositories and all users in the system**

* **git config --global -l ← lists all configurations made**

**Initializing and clone**

* **git init ← Initializes git**
* **git clone <repo\_url> <name\_you\_want\_to\_give\_it> ← clones repo from remote git repo (github/ bitbucket/ gitlab)**

**Basic Commands**

* **git add . ← adds all files to tracked**
* **git status ← get the status of the file**
* **git commit -m “<comment>” ← committing the tracked files**
* git commit -a -m “<comment>” ← skipping the ‘git add .’ the second time (not recommended)

**.gitignore file**

* **vim .gitignore ← adding filenames or directories which should be skipped while staging**
* Inside .gitignore,
  + if **/**<dirname> ← skips only the /<dirname> located on the root of **git repo**
  + if <dirname> ← skips all the <dirname> located anywhere in the **git repo**
* Changes made to the file after it is included in .gitignore will result in modified status even after including the filename. This is because the git is still tracking the file. To completely ignore the file, we have to run

**git rm --cached <.gitignored\_filename>**

**Diff**

* **git diff ← shows the difference b/w staged and unstaged changes**
* **git diff --staged ← shows the difference b/w staged and committed changes**

**rm and mv in git**

* git rm <filename> ← removes file
* git mv <old\_filename> <new\_filename> ← renames file

**Log**

* **git log ← shows all committed logs**
* **git log -p ← shows what is deleted or added along with committed logs**
* **git log -p -<number> ← shows the specified number of recent logs**
* git log --oneline ← all logs in one line
* git log --pretty=oneline ← all logs in a prettified format
* git log --pretty=short ← all logs in short format
* git log --pretty=full ← all logs with some more info of logs
* git log --pretty=format:”<git scm>” ← filter out logs

e.g. git log --pretty=format:”%h -- %an” ← shows commit\_hash and author name (refer git-log docs for more info)

* git log --since=<number>.<days/months/years> ← shows logs from that days/months/years onward

e.g. git log --since=2.months

**Restoring files**

* **git restore <file\_name> ← to discard changes in a file**
* **git restore --staged <filename> ← to unstage a file**
* git checkout -- <filename> ← to restore the previous committed changes

**(Use this command carefully)**

It is ineffective against .gitignore

* git checkout -f ← restores all the previous committed changes to all files

**git remote**

* **git remote ← gives remote name**
* git remote -v ← git remote name with additional fetch and pull url
* **git remote add origin <repo\_ssh\_url> ← adds the remote path to ‘origin’ (we can name origin with something else too)**
* Refer <https://docs.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

on how to generate ssh key and add public key toh github ssh-gpg keys section

* **git remote remove origin ← removes the remote path**
* **git push origin master ← push it to remote git repo as master**
* **git push origin <feature\_branch\_name> ← push its feature branches to git repo**
* git push origin <feature\_branch\_name>:<new\_name> ← sets a new name for the specified branch only on the remote repo
* **git push -d origin <branch\_name> ← deletes the specified branch in remote repo**

**Setting alias**

* **git config --global alias.<alias\_you\_want> <command>**

e.g. git config --global alias.**st** status

→ git **st**

git config --global alias.**unstage** **‘restore --staged’**

→ git **unstage** abc.txt

**Branching commands**

* **git branch ← shows the list of branches as well as which branch you are in**
* git branch -v ← shows the list of branches as well as its committed hash and message
* **git branch <branch\_name> ← creates new branch with the given branch name**
* git branch --merged ← shows all the merged branches
* git branch --no-merged ← shows all the unmerged branches
* **git checkout <branch\_name> ← switches to that branch**
* git checkout -b <branch\_name> ← creates the branch with the specified branch name as well as switch to that branch
* **git branch -d <branch\_name> ← deletes the specified branch (switch to master first)**

**Gives error if branch is not merged**

* git branch -D <branch\_name> ← forcefully deletes the branch

**Merge**

* Switch to master and,

**git merge <branch\_you\_want\_to\_merge> ← merges to master branch**