Git Cheat-sheet

Initial set-up:

Initial configuration of a new Git installation:

`git config` Set-ups Git for the first time

`git config user.name "your name"` Configures git to recognize you Locally

`git config --global user.name "your name"` Configures git to recognize you Globally

`git config --global user.email "your email"` “

`git --list` Lists all the configuration values

Checkings:

`git --version` Checks if git has been installed

`git config user.name` Shows who it is configured to

`git config user.email` Shows the email associated to git

Help:

`git help` Shows the 21 most common git commands or

`git help <command>` Give more specific help about <command>

`git <command> --help` “

e.g.: `git help init`
`git config --help`

List / Dir:

If see a hidden dir “.git” & file “.gitignore”, it’s already a repo!

`ls` Shows all files & subdirs in the directory

`ls -la` Shows everything including hidden (If see a hidden dir “.git” & file “.gitignore”, it’s already a repo)

Create Repo:

`git init` Execute this in the project directory xx

`git init xx` Creates a new Git repo xx

Status:

`git status` Checks the current state of repo
`git status <file>` Checks state of specific file

Delete Repo:

`rm -rf .git` Removes the git repository

Running this inside a repository removes “.git” and makes the directory un-track-able (un-git). Win-users can equivalently delete “.git” using file-explorer.

Stage-ing:

`git add <file>` Adds <file> to "*staging area*"

(Staged files are ready to be committed.)

e.g.: `git add *.txt`

`git add -A .` Adds everything in and beneath

(Important: use capital A)

Erasing, etc.

Add gitignore file:

`touch .gitignore` to create a "*.gitignore*" txt file This file can be edited and to each line we can specify the (type of) files that we do not want the to stage. (See Appendix B)

Reset:

`git reset <file>` removes file(s) from the staging area & brings it back to the working-area

`git reset` resets every modified file in working-space to its latest commit.*(you may lose all the changes.)*

Unstage:

`git rm -cached index.html` (to un-stage the file index.html)

`git checkout -- index.html` (to discard all the changes in index.html file)

Differences / Comparing

Differences:

`git diff` shows differences: Working-Directory <–vs–> Staging-Area

`git diff index.html` `git diff --staged` Shows differences:

Just-Staged <–vs.–> Last-Commits

(a.k.a. "Head")

`git diff HEAD` Shows differences:

Working-Area <—vs.—> Last-Commit

Committing

Commit:

`git commit -a`

`git commit` commits all the file in the staged area and asks for the comment

`git commit -m "Message goes here."` Switch '-m' adds a message to the commit

Logging/History:

`git log` creates a log of history

Create Branch

Branch:

`git branch -a`

Shows both remote & local branches

`git branch <new-branch-name>`

Create a new (local) branch

`git checkout <branch-name>`

Switches to <branch-name>

`git checkout -b <new-branch-name>`

Creates a new-branch & checkouts

`git reflog` Views the history of checkouts

`git branch -d <branch-name>`

Delete this branch, This do not delete if branch has unmerged changes.

`git branch -D <branch-name>`

Force delete this branch, even if it has unmerged changes.

`git branch -m <branch-new-name>`

Rename the current branch to branch-new-name.

Remote Repository (Git-Hub)

- log-in to: https://github.com
- Create a remote repository in https://github.com/YourGit/Git-cmdref.git
- `git remote add origin https://github.com/YourGit/Git-cmdref.git`

Removing Remote URL:

`git remote -v` views the current remote
`git remote rm` removes a remote URL from your repository
`git remote rm master`

Pushing:

`git push -u origin master`

Sends local changes to remote repository (*origin*)

Pulling:

`git pull origin master` Pull down any new changes (by collaborators etc.) from the remote repo

Branch:

`git branch -a`

Shows both remote & local branches

`git branch -r` Shows remote branches

`git checkout origin`

`git checkout <remotebranch>`

Summary

Example:

```
echo "# Git-Help-LaTeX" » README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/BehN/Git-Help-LaTeX.git
git push -u origin master
```

APPENDIX A
SAMPLE OF HOW TO NAVIGATE:
CLONE FROM GITHUB: → EDIT → COMMIT → PUSH TO GITHUB

How to colone::

- `go to GitHub`
- `go to Repositories` → `click on "personalWebPage"`
- `click on Clone-or-download` → `Copy the web URL`

e.g.: `''https://github.com/personalWebPage.git''`

- `go to "Z:\code"`
- `right click` → `click on: "Git Bash here!"`
- `git clone https://github.com/personalWebPage.git`

As a result, the diectory "personalWebPage" is created containing a copy of git-repository. This new directory is not a git repo yet!

- `git config -global user.name "John Doe"`
- `git git config -global user.email you@email.com`

To be done only once for your git installation!

How to make a branch::

- `ls`
- `cd personalWebPage`
- `git checkout -b dev` (makes a new local branch, 'dev')
- `git branch` (to show the existing branches, the active one is in green)
- `git checkout dev` (switched git to the new local branch 'dev')

After applying required edittings,

How to stage and commit::

- `git add --a`
- `git status` (check status on branch dev)
- `git commit -m "your commit note goes here!"`

How to push the branch to GitHub::

- `go to Git-Hub` → `create pull request`
- `git push origin dev` (push 'dev' to the remote, it is created if is not exist)

APPENDIX B
BATCH-FILE (*.CMD) TO CREATE .GITIGNORE

This is a simple windows script (batch file) that can be used to generate a sample (and fairly complete) `.gitignore`. Both the following script and `.gitignore` from it can be edited to customize it with your need. WrtGitIgnore.cmd:

```
@echo off

set CurrDir=%CD%
::cd /d %~dp0\..\..

set FN=.gitignore
attrib -h -r %FN%
del /s/q %FN%
::-----------------------

::WIN:
echo .dropbox>%FN%
echo desktop.ini>>%FN%
echo .tmp>>%FN%
echo nul*>>%FN%

::PDF
echo *.pdf>>%FN%

::Matlab:
echo *.asv>>%FN%

::Graphics
echo *.eps>>%FN%
echo *.png>>%FN%

::Hspice:
echo *.log>>%FN%
echo MIL.*>>%FN%
echo sxcmd.*>>%FN%
echo *.sx>>%FN%
echo *.lis>>%FN%
echo *.fsdef>>%FN%
echo *.str>>%FN%
echo *.ic0>>%FN%
echo *.st0>>%FN%
echo *.pa0>>%FN%
echo *.sw0>>%FN%
echo *.tr0>>%FN%
echo *.ac0>>%FN%

::TexnicCenter:
echo *.out>>%FN%
echo *.aux>>%FN%
echo *.blg>>%FN%
echo *.bbl>>%FN%
echo *.toc>>%FN%
echo *.dvi>>%FN%
echo *.bak>>%FN%
echo *.prj>>%FN%
echo *.ppl>>%FN%
echo *.lot>>%FN%
echo *.lof>>%FN%
echo *.tps>>%FN%
echo *.synctex>>%FN%
echo *.tmp>>%FN%
echo *.tps>>%FN%
echo *.pdfsync>>%FN%
echo *.ps>>%FN%
echo *.undo>>%FN%
echo *.tex~>>%FN%
echo *.tex.backup>>%FN%

::Vim:
```

```
echo *.project.vim>>%FN%
echo *.glg>>%FN%
echo *.glo>>%FN%
echo *.gls>>%FN%
echo *.ist>>%FN%
echo *.dcl>>%FN%

::TeXStudio/TeXMaker:
echo *.gz>>%FN%
echo *.spl>>%FN%
echo *.fls>>%FN%
echo *.brf>>%FN%
echo *.xml>>%FN%
echo *.bcf>>%FN%

::Beamer:
echo *.nav>>%FN%
echo *.snm>>%FN%

:: XHTML:
echo *.idx>>%FN%
echo *.css>>%FN%
echo *.ilg>>%FN%
echo *.ind>>%FN%

::Others:
echo *._*>>%FN%
echo *.ini>>%FN%
echo *.fdb*>>%FN%


attrib +h +r %FN%
::------------------------
start notepad++ %FN%
```