Git Cheat-sheet

Initial Configuration of a New Git Installation

Set-up:

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

`git config user.name "John Doe"`

Configures git to recognize you Locally

`git config --global user.name "John Doe"`

`git config --global user.email "jd@x.ca"`

Configures git to recognize you Globally

`git --list` Lists all the configuration values

Check:

`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

`git help <command>` or

`git <command> --help`

Give more specific help about <command>

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

Create a Local Repository (Repo)

List / Dir:

`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 a Local Repo

Remove-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.

Staging

Stage:

`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)

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 A)

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 <file>` to un-stage

Differences / Comparing

Differences:

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

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.' e.g.:
git commit -m 'initial project version'
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
```

Appendices

Appendix A
Appendix: 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%
```