**GIT BASH CHEAT SHEET**

created by Stanislav Minchev

**INSTALL GIT**

**Download Git:**

https://git-scm.com/downloads

**Installation**

Using default options

**COMMON GIT COMMANDS**

Show installation folder:

**$ which git**

Git version:

**$ git --version**

Git help

**$ git --help**

**f - forward**

**b - backward**

**q - quit**

**CONFIGURE GIT**

Configure user information for all local repositories

Set the name for the user:

**$ git config --global user.name "[name]"**

Set the email for the user:

**$ git config --global user.email "[mail@mail.mail]"**

Set the default editor for Windows to Notepad:

**$ git config --global core.editor "notepad.exe"**

Set the default editor for Unix to Mate (wait and start on the first line):

**$ git config --global core.editor "mate -wl1"**

Enable colors in git for better experience:

**$ git config --global color.ui true**

Adding aliases for fit commands(example alias.co "checkout"):

**$ git config --global alias.[command\_alias] "[command]"**

Ignoring files globally:

**$ git config --global core.excludefile [full\_path]/.gitignore\_global**

Show list with current configurations:

**$ git config --list**

Show list with current configurations with location:

**$ git config --list --show-origin**

Open .gitconfig (unix style):

**$ cat .gitconfig**

**CREATE REPOSITORY**

Start a new repository:

**$ git init [project\_name]**

Download repository from url:

**$ git clone [url]**

Download repository from url and make folder with the specified name:

**$ git clone [url] [dir\_name]**

**COMMIT CHANGES**

List the changes in files if any :

**$ git status**

Add changes for a single file to the staging index:

**$ git add [file\_name]**

Add changes for a all files to the staging index:

**$ git add .**

Add changes from staging index to repository with a message:

**$ git commit -m "[Message]"**

Add changes from working directory to repository with a message (do not include untracked files):

**$ git commit -am "[Message]"**

Change only the last commit:

**$ git commit --amend -m"[Message]"**

Shows changes to tracked file (between repository and working dir)

**$ git diff [file\_name]**

Shows changes between repository and staged index

**$ git diff --staged**

Comparing commits:

**$ git diff [SHA-1] [file\_name]**

**$ git diff [SHA-1]..[SHA-1]**

**$ git diff --stat --summary [SHA-1]..HEAD**

Comparing commits ignoring spaces:

**$ git diff --b -w [SHA-1]..HEAD**

Reviewing commit:

**$ git show [SHA-1]**

**COMMIT HISTORY**

Show all commits, starting with newest :

**$ git log**

Show only n numbers of commits:

**$ git log -n**

Show commits after the date:

**$ git log --since=YYYY-MM-DD**

Show commits until the date:

**$ git log --until=YYYY-MM-DD**

Show commits before the date:

**$ git log --before=YYYY-MM-DD**

Show commits since the date until other date:

**$ git log --since=2 weeks --until=3.days**

Show commits from the author:

**$ git log --author="[Name]"**

Show commits containing the text in the message:

**$ git log --grep="[Text]"**

List of one line log:

**$ git log --oneline**

List of one line log with n lines:

**$ git log --oneline [n]**

List of one line log with full SHA-1:

**$ git log --format=oneline**

Log with email format:

**$ git log --format=email**

Log with full format:

**$ git log --format=full**

Log with raw format:

**$ git log --format=raw**

Filtering the log by SHA-1:

**$ git log [SHA-1]..[SHA1] --oneline**

Show all commits for the file from the first commit:

**$ git log [SHA-1]..[file]**

Show changes over time for a specific file:

**$ git log -p [file]**

Show statistic what is changed in the commits:

**$ git log --stat -summary**

Create graph for the commit:

**$ git log --graph**

**$ git log --oneline --graph --all --decorate**

**RESET, UNDO, CLEAN, DELETE**

Undo changes on working directory("--" good practice for checkout everything without branch):

**$ git checkout -- [file\_name or directory\_name]**

Rollback changes to files on working directory from repository :

**$ git checkout HEAD [file\_name]**

**$ git checkout [SHA-1] [file\_name]**

Rollback to the old commit([SHA] - min 8-10 symbols):

**$ git checkout [SHA] -- [file\_name]**

Revert commit like checkout[SHA] -- [filename]:

**$ git revert [SHA]**

Unstage files from the staging index and working directory, and move the Head pointer to the previous commit:

**$ git reset HEAD [file\_name]**

Move the Head pointer to the provided commit and match staging index and working directory:

**$ git reset --hard HEAD [file\_name or SHA-1]**

Move the Head pointer to the provided commit and change the staging index to match the repo (do not change working directory, mix is default for reset)(deprecated):

**$ git reset --mix HEAD [file\_name or SHA-1]**

Move the Head pointer to the provided commit:

**$ git reset --soft HEAD [file\_name or SHA-1]**

Remove untracked files from working directory:

**$ git clean**

Only show that will remove untracked files from working directory :

**$ git clean -n**

Remove non tracking files, but do not remove from staging index:

**$ git clean -f**

Delete the file from working dir:

**$ git rm [file\_name]**

Remove file from staged index, but left it inside working dir:

**$ git rm --cached [file\_name]**

**IGNORE FILE TRACKING**

Everything in .gitignore file will be ignored, and will not be tracked :

Lists of ignored files in this project :

**$ git ls-files --other --ignored --exclude-standard**

**BRANCH**

Lists of the branches :

**$ git branch**

Create new branch :

**$ git branch [branch\_name]**

Select branch :

**$ git checkout [branch\_name]**

Select branch :

**$ git checkout [branch\_name]**

Create new branch and Select the created branch :

**$ git checkout -b [branch\_name]**

Compare branches:

**$ git diff [branch\_name1]..[branch\_name2]**

List of the branches which are fully included in other branches :

**$ git branch --merged**

Rename branch :

**$ git branch -m [old\_branch\_name] [new\_branch\_name]**

Delete branch which is fully merged in its upstream branch:

**$ git branch -d [branch\_name]**

Delete branch no matter the its merged status :

**$ git branch -D [branch\_name]**

Give the current branch on the prompt (edit in .bash\_profile) for LINUX :

**echo $PS1='\W$(\_\_git\_ps1 "(%s)")>'**

Give the current branch on the prompt (edit in .bash\_profile) for WINDOWS :

**export PS1='\W$(\_\_git\_ps1 "(%s)")>'**

Merge branches:

**$ git merge [branch\_name]**

Merge branches without Fast-Forward merge :

**$ git branch --no-ff [branch\_name]**

Merge branches if only can performed Fast-Forward merge :

**$ git branch --ff-only [branch\_name]**

Abort merging:

**$ git merge --abort**

Merge tool:

**$ git mergetool --tool=[toolname]**

**STASH**

Put changes into stash :

**$ git stash save"[message]"**

Stash list:

**$ git stash list**

Show the diff between stash or the n-stash [stash@{n}] and the original parent :

**$ git stash show [stash@{n}]**

Show the diff:

**$ git stash show -p stash@{n}**

Put changes from stash into working dir and remove the changes from the stash:

**$ git stash pop stash@{n}**

Put changes from stash into working dir and rest the changes in the stash:

**$ git stash apply stash@{n}**

Delete from the stash :

**$ git stash drop stash@{x}**

Clear the stash :

**$ git stash clear**

**REMOTE REPOSITORY**

List of the remote repos:

**$ git remote**

Create new remote repository:

**$ git remote add [name] [url]**

Show remote url after name

**$ git remote -v**

Remove remote with the specified name:

**$ git remote rm [name]**

Show remote branches :

**$ git branch -r**

Show local and remote branches:

**$ git branch -a**

Create local repo :

**$ git clone [path]**

Create local repo project name [name]:

**$ git clone [path] [name]**

Create tracking branch in remote repository:

**$ git config branch.[name].remote [remote\_repo\_name]**

Create tracking branch in remote repository:

**$ git branch --set-upstream-to=[origin/remote\_repo\_name] [branch\_name]**

Remove tracking:

**$ git branch --unset-upstream**

Push changes to remote repo

**$ git push**

Push local branch to remote repository (-u create tracking branch):

**$ git push -u [remote\_repo\_name] [name\_local\_branch]**

Update from remote repo :

**$ git fetch**

Merging origin with master after fetch:

**$ git merge origin/master**

Update origin from remote repo and merge origin with master :

**$ git pull**

Merging unrelated files when pulling from remote repository :

**$ git pull origin [branch\_name] --allow-unrelated-histories**

Checking remote branch - create non tracking local branch from the remote branch

**$ git branch [name] [origin/non\_tracking or HEAD or SHA-1]**

Delete the non-tracking branch :

**$ git branch -d [branch\_name]**

Create new branch and select the newly created branch:

**$ git checkout -b [branch\_name] origin/[branch\_name]**

Deleting branch form server:

**$ git push origin :[branch]**

Deleting branch form server:

**$ git push origin --delete [branch]**

**ALIAS**

Example how to create alias to a command: alias "ex" command "example"

**$ git config --global alias.ex "example"**

Most common commands:

**$ git config --global alias.co checkout**

**$ git config --global alias.ci commit**

**$ git config --global alias.br branch**

**$ git config --global alias.df diff**

**$ git config --global alias.dfs "diff --staged"**

**$ git config --global alias.dfc "diff -cached"**

**$ git config --global alias.logg "log --graph --decorate --oneline --abbrev-commit --all"**

Modifying bash

- check current

echo $PS1

- create .bash\_profile

notepad -> Paste

export PS1='\[\033[33m\]\W\[\033[36m\] $(\_\_git\_ps1 "(%s)")\[\033[0m\]> '

-> Save As (Navigate to USER, save as .bash\_profile)

- Load bash\_profile without restarting cmd

source ~/.bash\_profile