Git Cheatsheet v0.1

# Create

Create a new local repository

$ git init

Clone an existing repository

$ git clone <remote>

Clone a specific branch

$ git clone -b <branch> <remote>

# LOCAL CHANGES

Changed files in your working directory

$ git status

Differences to all tracked files

$ git diff

Differences to specific tracked file

$ git diff <file>

Stage All (new, modified, deleted) files

$ git add –A

$ git add –all

Stage modified, deleted files

$ git add –u

$ git add --update

Stage specific <file> to the next commit

$ git add <file>

Commit previously staged changes and set commit message

$ git commit –m <msg>

Commit all local changes in tracked files and set commit message

$ git commit –am <msg>

Change the last commit

*Don’t amend published commits!*

$ git commit --amend

# COMMIT HISTORY

Show all commits, starting with newest

$ git log

Show all commits and added or deleted files

$ git log --summary

Show changes over time for a specific file

$ git log -p <file>

Differences with log to specific tracked file

$ git log –follow –p -- <file>

Who changed what and when in <file>

$ git blame <file>

# BRANCHES & TAGS

List all existing branches

$ git branch –av

List all existing refs

$ git show-ref

Switch HEAD branch

$ git checkout <branch>

Create & switch to branch

$ git checkout –b <branch>

Switch to remote-branch

git checkout <origin/branch>

Create a new branch based on your current HEAD

$ git branch <new-branch>

Rename current branch

$ git branch –m <new branchname>

Delete a local branch

$ git branch -d <branch>

Create a new tracking branch based on a remote branch

$ git checkout --track <remote/branch>

Mark the current commit with a tag

$ git tag <tag-name>

# UPDATE & PUBLISH

List all currently configured remotes

$ git remote -v

Show information about a remote repo

$ git remote show <url>

Add to a new remote repository

$ git remote add origin <url>

Download all changes from <remote>, but don’t integrate into HEAD

$ git fetch <remote>

Download changes and directly merge/integrate into HEAD

$ git pull <remote> <branch>

Publish local changes on a remote

$ git push <remote> <branch>

Publish local branch to remote repository

$ git push –u origin master

Delete a remote branch

$ git push origin --delete <branch> #git 1.7.0

$ git push origin :<branch> #older version

Delete a local branch

git branch --delete <branch>

git branch -d <branch> #short

git branch -D <branch> #force delete

Publish your tags

$ git push –tags

# MERGE & REBASE

Merge <branch> into your current HEAD

$ git merge <branch>

Rebase your current HEAD onto <branch>

*Don’t rebase published commits!*

$ git rebase <branch>

Abort a rebase

$ git rebase --abort

Continue a rebase after resolving conflicts

$ git rebase --continue

Use your configured merge tool to solve conflicts

$ git mergetool

Use your editor to manually solve conflicts and (after resolving) mark file as resolved

$ git add <resolved-file>

$ git rm <resolved-file>

# Remove & Rename

Remove all untracked files

$ git clean –df

Removing files or folder from tracking

$ git rm <file>

$ git rm -r <folder>

Rename files

$ git mv [-v] [-f] [-n] [-k] <src> <dest>

Options:

-f Force renaming/moving of a file

-k Skip move/rename action when error occur

-n Do nothing, only show what would happen

-v Report the names of files as they are moved

# Reset & Revert

Discard all local changes in your working directory

$ git reset --hard HEAD

Reset your HEAD pointer to a previous commit

…and discard all changes since then

$ git reset --hard <commit>

…and preserve all changes as unstaged

changes

$ git reset <commit>

…and preserve uncommitted local changes

$ git reset --keep <commit>

Revert a commit (by producing a new commit with contrary changes)

$ git revert <commit>

Discard local changes and reset a file

git checkout HEAD -- <file>

more: option –- means, treat every argument after this point as a file name, no matter what it looks like

# ARCHIVE and bundle

Create a zip file from specific branch without source control information. Format supported (.zip, .tar, .tar.gz)

$ git archive *branch* --format=zip --output=*file.zip*

Create a zip file from specific branch with source control information. Format supported (.zip, .tar, .tar.gz)

$ git bundle *branch* --format=zip --output=*file.zip*

# patch and APply patch

Create a standalone file that contains any changes that has been made.

$ git format-patch *branch* --stdout > *filename.patch*

Apply changes from a patch file.

# stash and apply stash

Save local changes into local git repository, but does not committed yet.

$ git stash

Apply the stash.

$ git stash apply

# Congif

Create shortcut command for git command.

$ git config --global alias.*sortcutAlias* “*git command*”

Config user name

$ git config --global user.name “Max Mustermann”

Config email address

$ git config --global user.email bar@foo.com

Config pretty command lines

$ git config --global color.ui true