

Git Cheat Sheet

Create

From existing data
`cd ~/projects/myproject`
`git init`
`git add .`

From existing repo
`git clone ~/existing/repo ~/new/repo`
`git clone you@host.org:dir/project.git`
default protocol is ssh

Remote repository for existing local data
`mkdir repo.git && cd repo.git`
`git init --bare [--shared=group]`
see [help](#) for info

Change

Files changed in working directory
`git status`

Changes to tracked files
`git diff`

Changes between ID1 and ID2
`git diff <ID1> <ID2>`

History of changes
`git log`

History of changes with files changed
`git whatchanged`

Who changed what and when in a file
`git blame <FILE>`

A commit identified by ID
`git show <ID>`

A specific file from a specific ID
`git diff <ID>:<FILE>`

All local branches
`git branch`
star "*" marks the current branch

Search for patterns
`git grep <PATTERN> [PATH]`

Branch/Tag

Switch to the BRANCH branch
`git checkout <BRANCH>`

Merge branch B1 into branch B2
`git checkout <B2>`
`git merge <B1>`

Create branch based on HEAD
`git branch <BRANCH>`

Create branch based on another
`git checkout <NEW> <BASE>`

Delete a branch
`git branch -d <BRANCH>`

Make a version or milestone
`git tag <VERSION_NAME>`
`git tag -a <VERSION_NAME> -m <MESSAGE>`
recommended

Revert

Return to the last committed state
`git checkout -f | git reset --hard`

Revert the last commit
`git revert HEAD`
Creates a new commit

Revert specific commit
`git revert <ID>`
Creates a new commit

Fix the last commit
`git commit -a --amend`
after editing the broken files

Checkout the ID version of a file
`git checkout <ID> <FILE>`

Rvert git reset --hard
`git reflog`
`git reset HEAD@{<N>}`

Update

Fetch latest changes from origin
`git fetch`
this does not merge them

Pull latest changes from origin
`git pull`
does a fetch followed by a merge

Apply a patch that someone sent you
`git am -3 patch.mbox`
In case of conflict, resolve the conflict and
`git am --resolve`

Commit

Commit all local changes
`git commit`
`git commit -m 'MESSAGE'`
`git commit -a -m 'MESSAGE'`

Changes last commit
`git commit --amend`

Publish

Prepare a patch for other developers
`git format-patch origin`

Push changes to origin
`git push <REMOTE> <BRANCH>`

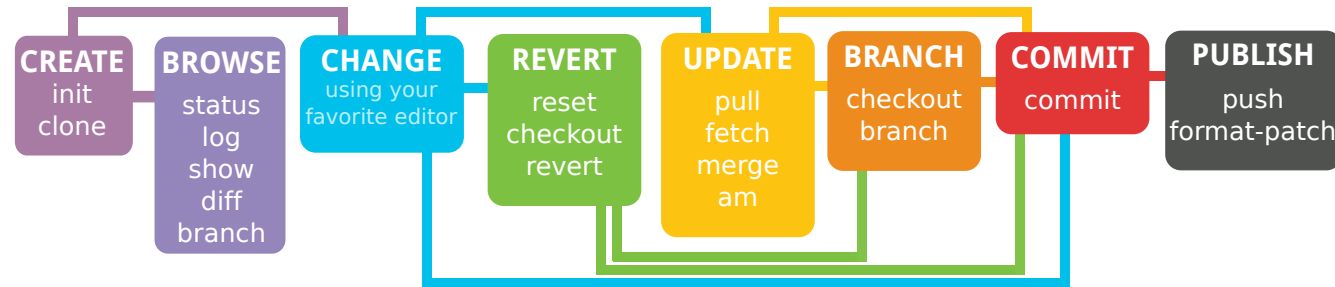
Fetch changes
`git fetch <REMOTE>`

Delete remote branch
`git push <REMOTE> :<BRANCH>`

Publish tags
`git push origin/upstream --tags`

List remotes
`git [REMOTE] -v`

Workflow



Legend
<required>
[optional]

Get Help!
`git help [command]`

Learn More!
<http://book.git-scm.com/>
<http://gitref.org/>
<http://learn.github.com/>
<http://help.github.com/>
<http://progit.org/book/>
<http://gitready.com/>

Useful tips

Get help!
`git help [command]`

Create empty branch
`git symbolic-ref HEAD refs/heads/newbranch`
`rm .git/index`
`git clean -fdx`
<DO WORK>
`git add your files`
`git commit -m 'Initial commit'`

Graphical log
`git log --graph`
`git log --graph --pretty=oneline --abbrev-commit`

Push branch to remote
`git push <REMOTE> <BRANCH>`

Semantic Versioning
`git tag <MAJOR.MINOR.PATCH>`
major API changes
minor add functionality
patch bug fixes

Merge conflicts

View merge conflicts
`git diff`

View merge conflicts against base file
`git diff --base <FILE>`

View merge conflicts against other changes
`git diff --theirs <FILE>`

View merge conflicts against your changes
`git diff --ours <FILE>`

Discard a conflicting patch
`git reset --hard`
`git rebase --skip`

After resolving conflicts, merge with
`git add <CONFLICTING_FILE>`
`git rebase --continue`

Configuration

`git config [--global]`
global is stored in ~/.gitconfig

user [user.name <NAME>
user.email <EMAIL>

color [color.ui auto true/false

github [github.user <USER>
github.token <TOKEN>

optimisation [pack.threads 0
diff.renamelimit 0
do not use on low memory pc

windows [core.autocrlf true

editor [core.editor <EDITOR>

merge [merge.tool <EDITOR>

master is the default development branch
origin is the default upstream repository
HEAD is the current branch

Successful branching model

