

# Git Cheat Sheet

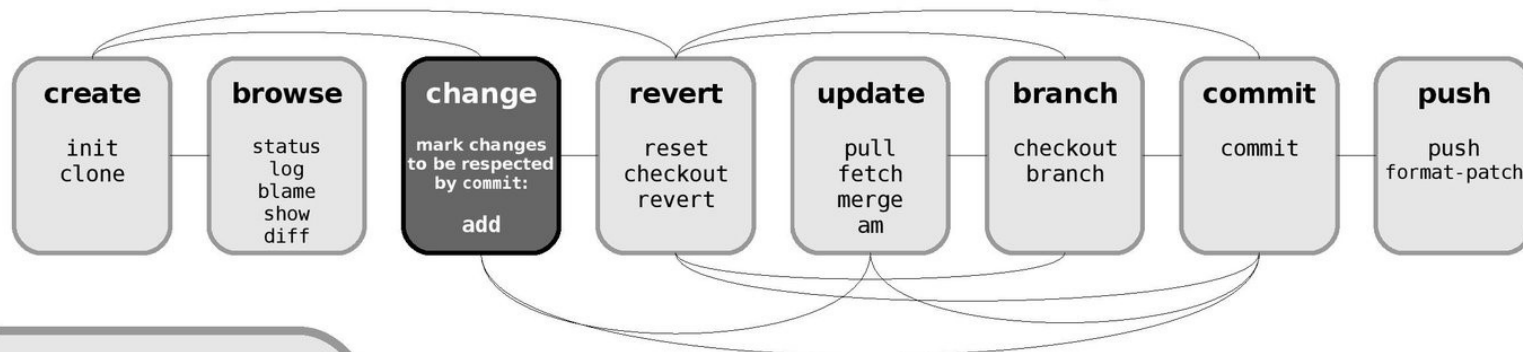
by Jan Krüger <jk@jk.gs>, <http://jan-krueger.net/git/>  
Based on work by Zack Rusin

## Basics

Use `git help [command]` if you're stuck.

master	default devel branch
origin	default upstream branch
HEAD	current branch
HEAD^	parent of HEAD
HEAD~4	great-great grandparent of HEAD
<code>foo..bar</code>	from branch <code>foo</code> to branch <code>bar</code>

## (left to right) Command Flow



## Create

### From existing files

```
git init
git add .
```

### From existing repository

```
git clone ~/old ~/new
git clone git://...
git clone ssh://...
```

## Publish

In Git, `commit` only respects changes that have been marked explicitly with `add`.

```
git commit [-a]
(-a: add changed files
automatically)
git format-patch origin
(create set of diffs)
git push remote
(push to origin or remote)
git tag foo
(mark current version)
```

## Useful Tools

```
git archive
Create release tarball
git bisect
Binary search for defects
git cherry-pick
Take single commit from elsewhere
git fsck
Check tree
git gc
Compress metadata (performance)
git rebase
Forward-port local changes to
remote branch
git remote add URL
Register a new remote repository
for this tree
git stash
Temporarily set aside changes
git tag
(there's more to it)
gitk
Tk GUI for Git
```

## Tracking Files

```
git add files
git mv old new
git rm files
git rm --cached files
(stop tracking but keep files in working dir)
```

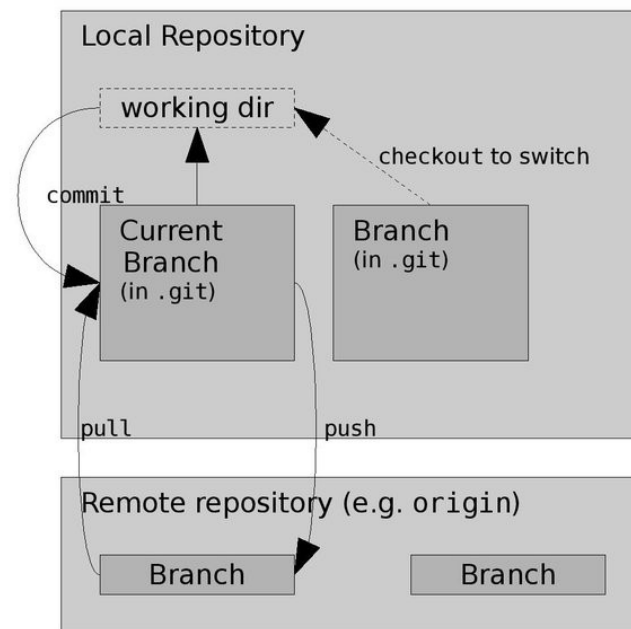
## View

```
git status
git diff [oldid newid]
git log [-p] [file|dir]
git blame file
git show id (meta data + diff)
git show id:file
git branch (shows list, * = current)
git tag -l (shows list)
```

## Update

```
git fetch (from def. upstream)
git fetch remote
git pull (= fetch & merge)
git am -3 patch.mbox
git apply patch.diff
```

## Structure Overview



## Revert

In Git, `revert` usually describes a new commit that undoes previous commits.

```
git reset --hard (NO UNDO)
(reset to last commit)
git revert branch
git commit -a --amend
(replaces prev. commit)
git checkout id file
```

## Branch

```
git checkout branch
(switch working dir to branch)
git merge branch
(merge into current)
git branch branch
(branch current)
git checkout -b new other
(branch new from other and
switch to it)
```

## Conflicts

Use `add` to mark files as resolved.

```
git diff [--base]
git diff --ours
git diff --theirs
git log --merge
gitk --merge
```