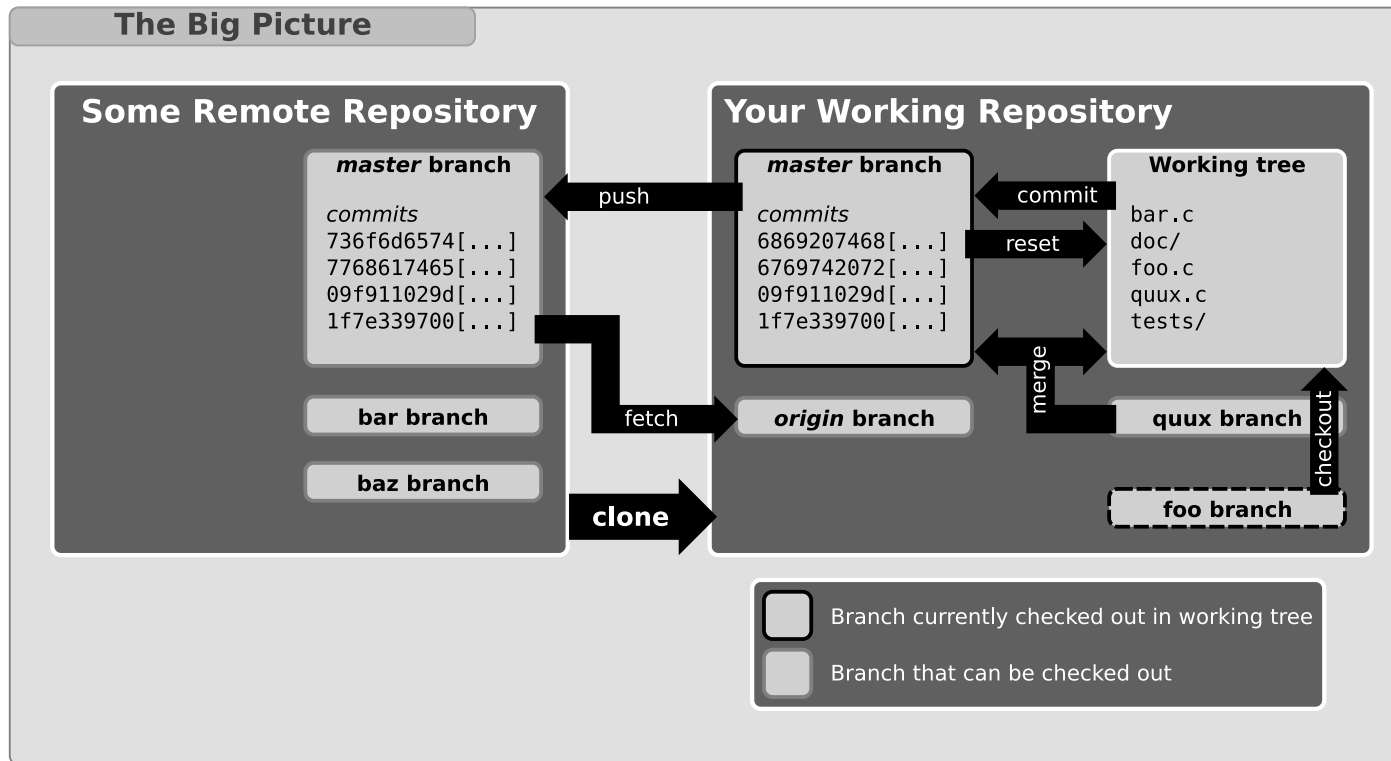


Git Cheat Sheet

Front – Overview & Concepts



Resources

Git website

<http://git.or.cz/>

Cheat Sheet website

<http://jan-krueger.net/git>

Terminology

Branch

A line of development to which changes can be made. *Merging* branches means that changes performed in one branch are transported into another. The most recent commit of a branch is called its *tip* and it can be referenced to a *head*. The default name of the development branch is *master*.

Commit (a.k.a.: revision, version)

A specific state in the branch's history. Each commit can be identified by a SHA hash and contains the hashes of its parents, i.e. the commit(s) it is based on, along with author information, a timestamp, and similar things.

As a verb: record specific changes made to the working tree in the associated branch as a new state.

Merge

Transport changes in a branch into the current one. To merge from a remote branch, a copy of it must first be *fetched*. The combination of *fetch* and *merge* is called *pull*.

Origin

Indicates the default *upstream* repository, i.e. the (possibly remote) repository you *cloned* your local repository from. (This is actually called *origin*, i.e. no capital "o").

Push

Transport local changes to a remote repository

Repository

A combination of a working tree (not usually accessible from the outside) and a set of branches, some of which may be copies of remote branches. On a physical level, a repository is a directory containing a `.git` directory with repository metadata, and the files you are currently working on.

Tag

A name for a specific commit that never changes. This can be used to mark interesting versions of a branch, e.g. releases.

Getting started

New to the trade:

1. This cheat sheet is not a tutorial. Read one!
2. Since I already mentioned tutorials: the Git website has a lot of documentation.

Switching from another system:

3. Interoperability tools exist: Arch, CVS, SVN
4. *add/commit* work differently than in most other SCM systems: *add* schedules changes for committing, *commit* records them. *commit -a* does both.
5. Every working tree contains a full repository, unlike as in CVS or SVN.

Useful Tools

git

Has all the standard operations as subcommands, e.g. *branch*, *checkout*, *clone*, *commit*, *fetch*, *merge* and so on.

git-gui

A graphical user interface for Git (Tk). Offers commands to commit, branch, merge etc.

gitk

Git's standard repository browser. Visualizes commits and such.

git-web

A web interface for viewing a Git repository. Ships with Git.

Create

```
From existing files
git init
git add .
git commit

From remote repository
git clone ../old ../new
git clone git://...
git clone ssh://...
```

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)
```

Configuration

Change options using `git config [--global] varname value`. The following variable names are useful:

- `core.bare`
True for repositories without a working tree (usually public repositories).
- `core.sharedRepository`
Set to group or all to make the repository contents writeable for the file group or everybody.
- `core.compression`
A zlib compression level for objects (0-9, 9 = best compression) or -1 to use zlib's default.
- `color.branch`
Color-code list of branches (true = always, auto = only when outputting to a terminal)
- `color.diff`
Color-code diffs (true, auto)
- `color.status`
Color-code output of `git status` (true, auto).
- `user.email`
Your e-mail address (used in commits).
- `user.name`
Your name (used in commits).

Browse

```
git status
git diff oldref newref
git log [-p] [file|dir]
git blame file
git show ref[:file]
git branch (shows list, * = current)
git tag -l (shows list)
```

Record

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

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

Object refs

master	default devel branch
origin	default upstream branch
HEAD	current branch
HEAD^	parent of HEAD
HEAD~4	great-great grandp. of HEAD
foo..bar	from ref foo to ref bar

Other Useful Commands

```
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)
```

Commit Messages

Some of Git's viewing tools need commit messages in the following format:

A brief one-line summary
<blank line>
Details about the commit

There's a little bit of room for your own notes here. This is your chance to customize this cheat sheet!

Change

Track Files

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

Revert

In Git, reverting usually means adding a commit that undos changes in previous commits.

```
git reset --hard (NO UNDO)
(throw away all pending changes)
git revert ref
git commit -a --amend
(replace previous commit)
git checkout ref file
```

Update

```
git fetch (from default upstream)
git fetch ref
git pull (= fetch + merge)
git am -3 patch.mbox
git apply patch.diff
```

Publish

```
git push
git push remote
git format-patch origin
(create set of diffs)
```

Resolve Conflicts

Use add to mark files as resolved.

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

Explanation of Syntax

```
[foo]  foo is optional
...    You can get creative here
foo    foo is a placeholder for
       something you need to fill in

ref    An object hash or name
       (see "Object Refs" for
       standard names)
```