Git for beginners: The definitive practical guide

Asked 16 years ago Modified 12 years, 1 month ago Viewed 764k times

854

votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

Ok, after seeing this post by PJ Hyett, I have decided to skip to the end and go with Git.

So what I need is a beginner's **practical** guide to Git. "Beginner" being defined as someone who knows how to handle their compiler, understands to some level what a Makefile is, and has touched source control without understanding it very well.

"Practical" being defined as this person doesn't want to get into great detail regarding what Git is doing in the background, and doesn't even care (or know) that it's distributed. Your answers might hint at the possibilities, but try to aim for the beginner that wants to keep a 'main' repository on a 'server' which is backed up and secure, and treat their local repository as merely a 'client' resource.

So:

Installation/Setup

- How to install Git
- How do you set up Git? Try to cover Linux, Windows, Mac, think 'client/server' mindset.
    - Setup GIT Server with Msysgit on Windows
- How do you create a new project/repository?
- How do you configure it to ignore files (.obj, .user, etc) that are not really part of the codebase?

Working with the code

- How do you get the latest code?
- How do you check out code?
- How do you commit changes?
- How do you see what's uncommitted, or the status of your current codebase?
- How do you destroy unwanted commits?
- How do you compare two revisions of a file, or your current file and a previous revision?
- How do you see the history of revisions to a file?
- How do you handle binary files (visio docs, for instance, or compiler environments)?
- How do you merge files changed at the "same time"?
- How do you undo (revert or reset) a commit?

Tagging, branching, releases, baselines

- How do you 'mark' 'tag' or 'release' a particular set of revisions for a particular set of files so you can always pull that one later?
- How do you pull a particular 'release'?
- How do you branch?
- How do you merge branches?
- How do you resolve conflicts and complete the merge?
- How do you merge parts of one branch into another branch?
- What is rebasing?
- How do I track remote branches?
- How can I create a branch on a remote repository?
- How do I delete a branch on a remote repository?
- Git workflow examples

Other

- Describe and link to a good GUI, IDE plugin, etc. that makes Git a non-command line resource, but please list its limitations as well as its good.
    - msysgit - Cross platform, included with Git

    - gitk - Cross platform history viewer, included with Git
    - gitnub - Mac OS X
    - gitx - Mac OS X history viewer
    - smartgit - Cross platform, commercial, beta
    - tig - console GUI for Linux
    - qgit - GUI for Windows, Linux
    - Git Extensions - package for Windows, includes friendly GUI
- Any other common tasks a beginner should know?
    - Git Status tells you what you just did, what branch you have, and other useful information
- How do I work effectively with a subversion repository set as my source control source?

Other Git beginner's references

- Git guide
- Git book
- Git magic
- gitcasts
- GitHub guides
- Git tutorial
- Progit - book by Scott Chacon

- Git - SVN Crash Course
- Git from the bottom up
- Git ready
- gitref.org
- Git visual cheatsheet

Delving into Git

- Understanding Git conceptually
- Git for computer scientists (and another version)

I will go through the entries from time to time and 'tidy' them up so they have a consistent look/feel and it's easy to scan the list - feel free to follow a simple "header - brief explanation - list of instructions - gotchas and extra info" template. I'll also link to the entries from the bullet list above so it's easy to find them later.

git version-control

Share

edited May 23, 2017 at 12:34

community wiki
43 revs, 22 users 22%
Adam Davis

37 Answers

Sorted by: Highest score (default)

1 2 Next

118
votes

How do you create a new project/repository?

A git repository is simply a directory containing a special `.git` directory.

This is different from "centralised" version-control systems (like subversion), where a "repository" is hosted on a remote server, which you `checkout` into a "working copy" directory. With git, your working copy *is* the repository.

Simply run `git init` in the directory which contains the files you wish to track.

For example,

```
cd ~/code/project001/
git init
```

This creates a `.git` (hidden) folder in the current directory.

To make a new project, run `git init` with an additional argument (the name of the directory to be created):

```
git init project002

(This is equivalent to: mkdir project002 && cd
project002 && git init)
```

To check if the current current path is within a git repository, simply run `git status` - if it's not a repository, it will report "fatal: Not a git repository"

You could also list the `.git` directory, and check it contains files/directories similar to the following:

```
$ ls .git
HEAD          config        hooks/        objects/
branches/     description   info/         refs/
```

If for whatever reason you wish to "de-git" a repository (you wish to stop using git to track that project). Simply remove the `.git` directory at the base level of the repository.

```
cd ~/code/project001/
rm -rf .git/
```

Caution: This will destroy *all* revision history, *all* your tags, *everything* git has done. It will not touch the "current" files (the files you can currently see), but previous changes, deleted files and so on will be unrecoverable!

Share

edited Jun 20, 2020 at 9:12

Community Bot
1 ● 1

answered Nov 26, 2008 at 9:26

dbr
169k ● 69 ● 283 ● 347

3 Git makes its objects read-only, so you'll want `rm -rf .git` to obliterate git's database. – Josh Lee Oct 13, 2009 at 1:16

Normally a .gitignore file will need to be created during normal usage to specify files/trees to ignore in versioning, so to be complete about the last part on "de-gitting", besides removing .git you would also need to remove the .gitignore file. :) – Monoman Jul 22, 2010 at 19:41

How about bare repositories? They are somehow "centralized", and I think they're a good thing for many projects that need some sort of centralization (eg: projects developed by many people) – peoro Jan 18, 2011 at 8:28

WRT running `git status` to ensure you're within a repository: this has one gotcha: if you have the environment variable $GIT_DIR set in your current shell, git will ignore your current location and use the repository at $GIT_DIR. I should know, I lost an hour to that yesterday. – sanmiguel Mar 1, 2012 at 12:17

110 votes

GUIs for git

Git GUI

Included with git — Run `git gui` from the command line, and the Windows msysgit installer adds it to the Start menu.

Git GUI can do a majority of what you'd need to do with git. Including stage changes, configure git and repositories, push changes, create/checkout/delete branches, merge, and many other things.

One of my favourite features is the "stage line" and "stage hunk" shortcuts in the right-click menu, which lets you commit specific parts of a file. You can achieve the same via `git add -i`, but I find it easier to use.

It isn't the prettiest application, but it works on almost all platforms (being based upon Tcl/Tk)

Screenshots | a screencast

GitK

Also included with git. It is a git history viewer, and lets you visualise a repository's history (including branches, when they are created, and merged). You can view and search commits.

Goes together nicely with git-gui.

Gitnub

Mac OS X application. Mainly an equivalent of `git log`, but has some integration with github (like the "Network view").

Looks pretty, and fits with Mac OS X. You can search repositories. The biggest critisism of Gitnub is that it shows history in a linear fashion (a single branch at a time) - it doesn't visualise branching and merging, which can be important with git, although this is a planned improvement.

Download links, change log and screenshots | git repository

GitX

Intends to be a "gitk clone for OS X".

It can visualise non-linear branching history, perform commits, view and search commits, and it has some other nice features like being able to "Quicklook" any file in any revision (press space in the file-list view), export any file (via drag and drop).

It is far better integrated into OS X than `git-gui`/`gitk`, and is fast and stable even with exceptionally large repositories.

The original git repository pieter has not updated recently (over a year at time of writing). A more actively maintained branch is available at brotherbard/gitx - it adds "sidebar, fetch, pull, push, add remote, merge, cherry-pick, rebase, clone, clone to"

Download | Screenshots | git repository | brotherbard fork | laullon fork

---

SmartGit

From the homepage:

> SmartGit is a front-end for the distributed version control system Git and runs on Windows, Mac OS X and Linux. SmartGit is intended for developers who prefer a graphical user interface over a command line client, to be even more productive with Git — the most powerful DVCS today.

You can download it from their website.

Download

TortoiseGit

TortoiseSVN Git version for Windows users.

> It is porting TortoiseSVN to TortoiseGit The latest release 1.2.1.0 This release can complete regular task, such commit, show log, diff two version, create branch and tag, Create patch and so on. See ReleaseNotes for detail. Welcome to contribute this project.

Download

QGit

> QGit is a git GUI viewer built on Qt/C++.
>
> With qgit you will be able to browse revisions history, view patch content and changed files, graphically following different development branches.

Download

---

gitg

> gitg is a git repository viewer targeting gtk+/GNOME. One of its main objectives is to provide a more unified user experience for git frontends across multiple desktops. It does this not be writing a cross-platform application, but by close collaboration with similar clients for other operating systems (like GitX for OS X).

Features

- Browse revision history.
- Handle large repositories (loads linux repository, 17000+ revisions, under 1 second).
- Commit changes.
- Stage/unstage individual hunks.
- Revert changes.

- Show colorized diff of changes in revisions.
- Browse tree for a given revision.
- Export parts of the tree of a given revision.
- Supply any refspec which a command such as 'git log' can understand to built the history.
- Show and switch between branches in the history view.

Download: releases or source

Gitbox

> Gitbox is a Mac OS X graphical interface for Git version control system. In a single window you see branches, history and working directory status.
>
> Everyday operations are easy: stage and unstage changes with a checkbox. Commit, pull, merge and push with a single click. Double-click a change to show a diff with FileMerge.app.

Download

Gity

The Gity website doesn't have much information, but from the screenshots on there it appears to be a feature rich open source OS X git gui.

Download or source

Meld

> Meld is a visual diff and merge tool. You can compare two or three files and edit them in place (diffs update dynamically). You can compare two or three folders and launch file comparisons. You can browse and view a working copy from popular version control systems such such as CVS, Subversion, Bazaar-ng and Mercurial [*and Git*].

Downloads

Katana

A Git GUIfor OSX by Steve Dekorte.

> At a glance, see which remote branches have changes to pull and local repos have changes to push. The git ops of add, commit, push, pull, tag and reset are supported as well as visual diffs and visual browsing of project hieracy that highlights local changes and additions.

Free for 1 repository, $25 for more.

Download

Sprout (formerly GitMac)

Focuses on making Git easy to use. Features a native Cocoa (mac-like) UI, fast repository browsing, cloning, push/pull, branching/merging, visual diff, remote branches, easy access to the Terminal, and more.

By making the most commonly used Git actions intuitive and easy to perform, Sprout (formerly GitMac) makes Git user-friendly. Compatible with most Git workflows, Sprout is great for designers and developers, team collaboration and advanced and novice users alike.

Download | Website

Tower

A feature-rich Git GUI for Mac OSX. 30-day free trial, $59USD for a single-user license.

Download | Website

EGit

> EGit is an Eclipse Team provider for the Git version control system. Git is a distributed SCM, which means every developer has a full copy of all

> history of every revision of the code, making queries against the history very fast and versatile.
>
> The EGit project is implementing Eclipse tooling on top of the JGit Java implementation of Git.

Download | Website

Git Extensions

Open Source for Windows - installs everything you need to work with Git in a single package, easy to use.

> Git Extensions is a toolkit to make working with Git on Windows more intuitive. The shell extension will intergrate in Windows Explorer and presents a context menu on files and directories. There is also a Visual Studio plugin to use git from Visual Studio.

Download

Big thanks to dbr for elaborating on the git gui stuff.

SourceTree

SourceTree is a *free* Mac client for Git, Mercurial and SVN. Built by Atlassian, the folks behind BitBucket, it seems to work equally well with any VC system, which allows you to master a single tool for use with all of your projects, however they're version-controlled. Feature-packed, and FREE.

> Expert-Ready & Feature-packed for both novice and advanced users:
>
> Review outgoing and incoming changesets. Cherry-pick between branches. Patch handling, rebase, stash / shelve and much more.

Download | Website

Share

edited May 23, 2017 at 11:33

community wiki
20 revs, 8 users 68%
dylanfm

---

2 You have some good answers (especially gitcasts, and the push/pull answer), but could I recommend splitting it into a separate answers? The question'er requested that you "don't try to jam a bunch of information into one answer"! – dbr Nov 27, 2008 at 13:08

3 Maybe you should add TortoiseGit code.google.com/p/tortoisegit to your list, for Windows gitters... – Andrzej Undzillo Nov 19, 2009 at 10:33

1 Gity (macendeavor.com/gity) is an option, but is still in development (OS X) – Dave DeLong Jan 27, 2010 at 6:04

2 Tower ("The most powerful Git client for Mac") is a beautiful new client for Git. – rubiii Feb 14, 2011 at 1:37

59
votes

Well, despite the fact that you asked that we not "simply" link to other resources, it's pretty foolish when there already exists a community grown (and growing) resource that's really quite good: the Git Community Book. Seriously, this 20+ questions in a question is going to be anything but concise and consistent. The Git Community Book is available as both HTML and PDF and answers many of your questions with clear, well formatted and peer reviewed answers and in a format that allows you to jump straight to your problem at hand.

Alas, if my post really upsets you then I'll delete it. Just say so.

Share

edited Nov 25, 2008 at 1:17
Greg Hewgill
990k ● 191 ● 1.2k ● 1.3k

answered Nov 25, 2008 at 0:49
Pat Notz
214k ● 31 ● 94 ● 92

2 If you're not using git because it's a DVCS, why bother using git at all? This question is silly and diverts resources that could be spent on other things to satisfy a questionable goal.
– Randal Schwartz Mar 15, 2010 at 4:20

56
votes

How to configure it to ignore files:

The ability to have git ignore files you don't wish it to track is very useful.

To ignore a file or set of files you supply a pattern. The pattern syntax for git is fairly simple, but powerful. It is applicable to all three of the different files I will mention bellow.

- A blank line ignores no files, it is generally used as a separator.
- Lines staring with **#** serve as comments.
- The **!** prefix is optional and will negate the pattern. Any negated pattern that matches will override lower precedence patterns.
- Supports advanced expressions and wild cards
    - Ex: The pattern: ***.[oa]** will ignore all files in the repository ending in .o or .a (object and archive files)
- If a pattern has a directory ending with a slash git will only match this directory and paths underneath it. This excludes regular files and symbolic links from the match.
- A leading slash will match all files in that path name.
    - Ex: The pattern **/*.c** will match the file **foo.c** but not **bar/awesome.c**

Great Example from the gitignore(5) man page:

```
$ git status
[...]
# Untracked files:
[...]
#       Documentation/foo.html
#       Documentation/gitignore.html
#       file.o
#       lib.a
#       src/internal.o
[...]
$ cat .git/info/exclude
  # ignore objects and archives, anywhere in the
tree.
  *.[oa]
$ cat Documentation/.gitignore
# ignore generated html files,
*.html
# except foo.html which is maintained by hand
!foo.html
$ git status
[...]
# Untracked files:
[...]
#       Documentation/foo.html
[...]
```

Generally there are three different ways to ignore untracked files.

1) Ignore for all users of the repository:

Add a file named **.gitignore** to the root of your working copy.

Edit **.gitignore** to match your preferences for which files should/shouldn't be ignored.

```
git add .gitignore
```

and commit when you're done.

2) Ignore for only your copy of the repository:

Add/Edit the file **$GIT_DIR/info/exclude** in your working copy, with your preferred patterns.

Ex: My working copy is ~/src/project1 so I would edit **~/src/project1/.git/info/exclude**

You're done!

3) Ignore in all situations, on your system:

Global ignore patterns for your system can go in a file named what ever you wish.

Mine personally is called **~/.gitglobalignore**

I can then let git know of this file by editing my **~/.gitconfig** file with the following line:

```
core.excludesfile = ~/.gitglobalignore
```

You're done!

I find the gitignore man page to be the best resource for more information.

Share edited Nov 25, 2008 at 1:19

Greg Hewgill
990k ● 191 ● 1.2k ● 1.3k

answered Nov 25, 2008 at 1:04

Brian Gianforcaro
27.1k ● 11 ● 59 ● 78

Could somebody, please, add one minor but important detail to this post? This works only for files already not tracked by git. To 'untrack' file but leave it in filesystem, you need 'git rm --cached filename'. Thanks! – Nikita Rybak Jul 5, 2010 at 22:20

I just want to note that adding the core.excludesfile line didn't work for me. I had to [git config --global core.excludesfile ~/.gitglobalignore] to make it work. – Coding District Aug 23, 2010 at 6:32

There's now a project on Github called gitignore that has gitignore files for a variety of languages and development environments: github.com/github/gitignore – Ryan Lundy Mar 24, 2011 at 19:18

47 votes

How do you 'tag' a particular set of revisions

How do you 'mark' 'tag' or 'release' a particular set of revisions for a particular set of files so you can always pull that one later?

Using the `git tag` command.

To simply "tag" the current revision, you would just run..

```
git tag -a thetagname
git tag -a 0.1
git tag -a 2.6.1-rc1 -m 'Released on 01/02/03'
```

To list the current tags, simply run `git tag` with no arguments, or `-l` (lower case L):

```
$ git tag -a thetagname # and enter a message, or
use -m 'My tag annotation'
$ git tag -l
thetagname
```

To delete a tag, you use the `-d` flag:

```
$ git tag -d thetagname
Deleted tag 'thetagname'
$ git tag
[no output]
```

To tag a specific (previous) commit, you simply do..

```
git tag [tag name] [revision SHA1 hash]
```

For example:

```
git tag 1.1.1
81b15a68c6c3e71f72e766931df4e6499990385b
```

Note: by default, git creates a "lightweight" tag (basically a reference to a specific revision). The "right" way is to use the `-a` flag. This will launch your editor asking for a tag message (identical to asking for a commit message, you can also use the `-m` flag to supply the tag message on the command line). Using an annotated tag creates an object with its own ID, date, tagger (author), and optionally a GPG signature (using the `-s` tag). *For further information on this, see this post*

```
git tag mytagwithmsg -a -m 'This is a tag, with
message'
```

And to list the tags with annotations, use the `-n1` flag to show 1 line of each tag message (`-n245` to show the first 245 lines of each annotation, and so on):

```
$ git tag -l -n1
mytagwithmsg    This is a tag, with message
```

For more information, see the git-tag(1) Manual Page

Share

edited Apr 6, 2009 at 13:43

answered Nov 27, 2008 at 4:58

dbr
169k ● 69 ● 283 ● 347

git tag *does not* create tags by default, just lightweight references. You must use either -a or -s to create a tag object (which things like describe will use): rockstarprogrammer.org/post/2008/oct/16/… – Dustin Dec 1, 2008 at 8:48

Ah, interesting. Thanks, I've updated the answer to reflect this – dbr Dec 1, 2008 at 13:00

And how do you tag a previously committed revision? (sorry it's too long so I skimmed through, did I miss something?) – hasen Apr 6, 2009 at 4:03

hasen j: Added info to answer, basically `git tag tagname revision_SHA1` – dbr Apr 6, 2009 at 13:44

1 To push tags into the remote repo, add --tags when using git push (info from github help area). – Héctor Ramos Dec 23, 2009 at 12:07

---

46 votes

Workflow example with GIT.

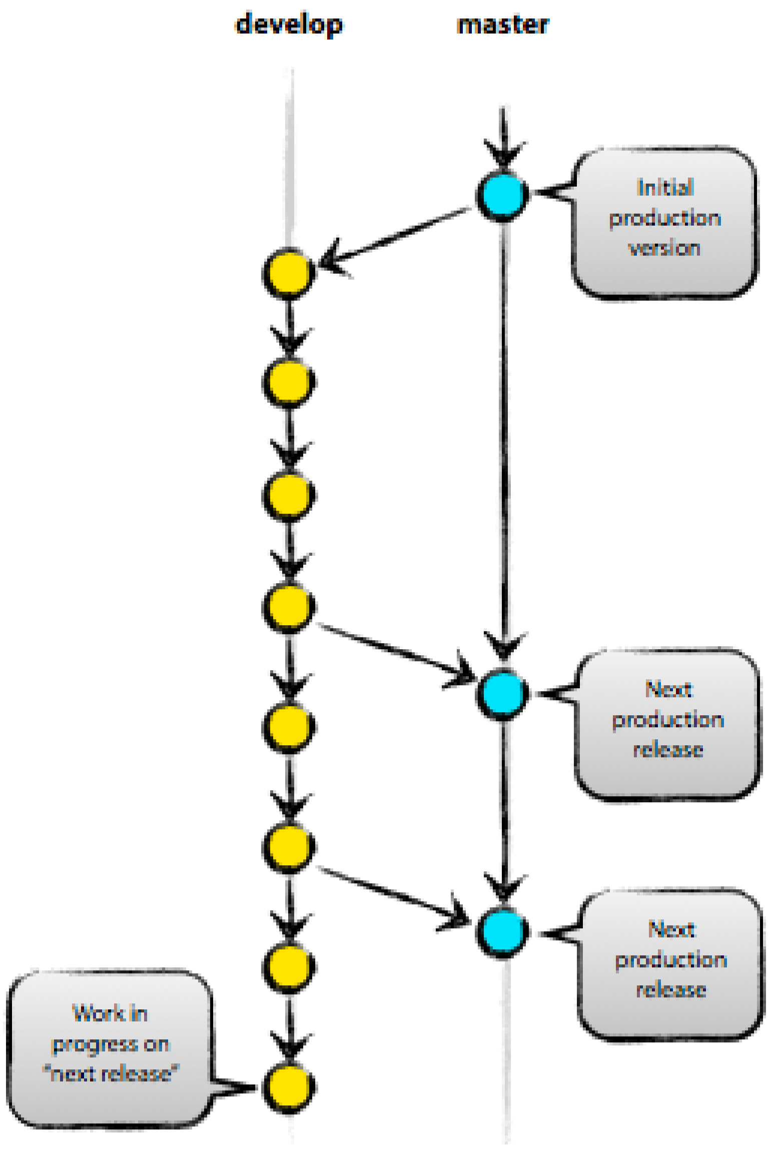
Git is extremely flexible and adapts good to any workflow, but not enforcing a particular workflow might have the negative effect of making it hard to understand what you can do with git beyond the linear "backup" workflow, and how useful branching can be for example.

This blog post explains nicely a very simple but effective workflow that is really easy to setup using git.

quoting from the blog post: We consider origin/master to be the main branch where the source code of HEAD always reflects a production-ready state:

The workflow has become popular enough to have made a project that implements this workflow: git-flow

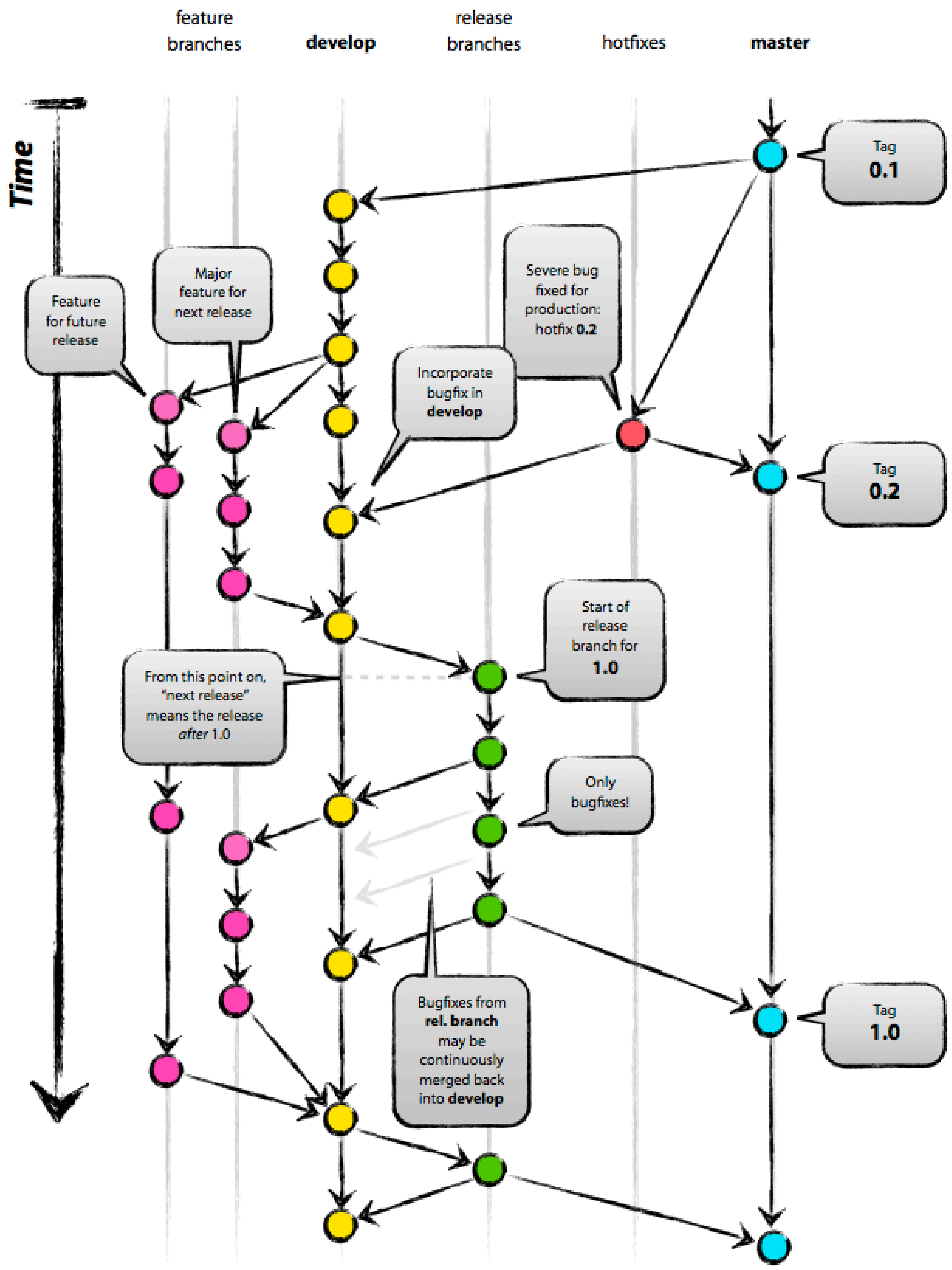
Nice illustration of a simple workflow, where you make all your changes in develop, and only push to master when the code is in a production state:


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

Now let's say you want to work on a new feature, or on refactoring a module. You could create a new branch, what we could call a "feature" branch, something that will take some time and might break some code. Once your feature is "stable enough" and want to move it "closer" to production, you merge your feature branch into develop. When all the bugs are sorted out after the merge and your code passes all tests rock solid, you push your changes into master.

During all this process, you find a terrible security bug, that has to be fixed right away. You could have a branch called hotfixes, that make changes that are pushed quicker back into production than the normal "develop" branch.

Here you have an illustration of how this feature/hotfix/develop/production workflow might look like (well explained in the blog post, and I repeat, the blog post explains the whole process in a lot more detail and a lot better than I do.


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Share

edited May 13, 2011 at 9:37

community wiki
4 revs
ashwoods

I am a git newbie, and this diagram makes it *more* confusing for me. – finnw May 11, 2011 at 22:14

Which one, the first one, or the last one? I didn't really want to make the post too long, but I'll add a small explanation of both diagrams later. – ashwoods May 12, 2011 at 7:59

Read the full article. I got confused by this diagram as well, but the blog post is very well written nvie.com/posts/a-successful-git-branching-model – Felipe Sabino May 12, 2011 at 11:55

is it better now? i only wanted to qive a rough overview, not repost the whole blog post here :) – ashwoods May 13, 2011 at 9:01

---

39 votes

Here's a copy of PJ Hyett's post, as it is not available anymore:

> ## Git Isn't Hard
>
> Nov 23, 2008
>
> When we tell people why they should use Git over Subversion, the go-to line is, "Git does Subversion better than Subversion, but it does a lot more than that."
>
> The "lot more" is comprised of a bunch of stuff that makes Git really shine, but it can be pretty overwhelming for those coming from other SCM's like Subversion.

That said, there's nothing stopping you from using Git just like you use Subversion while you're making the transition.

Assuming you've installed the necessary software and have a remote repository somewhere, this is how you would grab the code and push your changes back with Subversion:

```
$ svn checkout svn://foo.googlecode.com/svn/trunk foo
# make your changes
$ svn commit -m "my first commit"
```

And how would you do it in Git:

```
$ git clone git@github.com:pjhyett/foo.git
# make your changes
$ git commit -a -m "my first commit"
$ git push
```

One more command to make it happen in Git. That extra command has large implications, but for the purposes of this post, that's all we're talking about, one extra command.

See, it really isn't that hard.

Update: I'd be remiss to not also mention that the equivalent of updating your local copy in Subversion compared to Git is `svn update` and

> `git pull`, respectively. Only one command in both cases.

Share

answered Apr 20, 2010 at 20:02

community wiki
Adam Davis

33
votes

How to install Git

On Windows:

Install msysgit

There are several downloads:

- **Git:** Use this unless you specifically need one of the other options below.
- **PortableGit:** Use this if you want to run Git on a PC without installing on that PC (e.g. running Git from a USB drive)
- **msysGit:** Use this if you want to develop Git itself. If you just want to use Git for *your* source code, but don't want to edit *Git's* source code, you don't need this.

This also installs a Cygwin bash shell, so you can use the `git` in a nicer shell (than cmd.exe), and also includes git-gui (accessible via `git gui` command, or the `Start > All Programs > Git` menu)

Mac OS X

Use the git-osx-installer, or you can also install from source

Via a package manager

Install `git` using your native package manager. For example, on Debian (or Ubuntu):

```
apt-get install git-core
```

Or on Mac OS X, via MacPorts:

```
sudo port install git-core+bash_completion+doc
```

…or fink:

```
fink install git
```

…or Homebrew:

```
brew install git
```

On Red Hat based distributions, such as Fedora:

```
yum install git
```

In Cygwin the Git package can be found under the "devel" section

From source (Mac OS X/Linux/BSD/etc.)

In Mac OS X, if you have the Developer Tools installed, you can compile Git from source very easily. Download the latest version of Git as a `.tar.bz` or `.tar.gz` from http://git-scm.com/, and extract it (double click in Finder)

On Linux/BSD/etc. it should be much the same. For example, in Debian (and Ubuntu), you need to install the `build-essential` package via `apt`.

Then in a Terminal, `cd` to where you extracted the files (Running `cd ~/Downloads/git*/` should work), and then run..

```
./configure && make && sudo make install
```

This will install Git into the default place (`/usr/local` - so `git` will be in `/usr/local/bin/git`)

It will prompt you to enter your password (for `sudo`), this is so it can write to the `/usr/local/` directory, which can only be accessed by the "root" user so sudo is required!

If you with to install it somewhere separate (so Git's files aren't mixed in with other tools), use `--prefix` with the configure command:

```
./configure --prefix=/usr/local/gitpath
make
sudo make install
```

This will install the `git` binary into `/usr/local/bin/gitpath/bin/git` - so you don't have to type that every time you, you should add into your `$PATH` by adding the following line into your `~/.profile`:

```
export PATH="${PATH}:/usr/local/bin/gitpath/bin/"
```

If you do not have sudo access, you can use `--prefix=/Users/myusername/bin` and install into your home directory. Remember to add `~/bin/` to `$PATH`

The script x-git-update-to-latest-version automates a lot of this:

> This script updates my local clone of the git repo (localy at `~/work/track/git`), and then configures, installs (at `/usr/local/git`-`git describe`) and updates the `/usr/local/git` symlink.
>
> This way, I can have `/usr/local/git/bin` in my `PATH` and I'm always using the latest version.

> The latest version of this script also installs the man pages. You need to tweak your `MANPATH` to include the `/usr/local/git/share/man` directory.

Share

edited Mar 24, 2011 at 19:26

Ryan Lundy
210k ● 41 ● 184 ● 216

answered Nov 27, 2008 at 13:25

dbr
169k ● 69 ● 283 ● 347

5 On Fedora: `yum install git`. For the GUI run `yum install git-gui`. – Cristian Ciupitu Sep 12, 2009 at 2:17

2 On Mac, `sudo port install git-core+bash_completion+doc` – Singletoned Oct 9, 2009 at 11:08

I downloaded fink for mac, but running fink install git gives me error: "Failed: no package found for specification 'git'!" – quano Dec 17, 2009 at 1:17

@quano It should be there, pdb.finkproject.org/pdb/package.php/git - check Fink is properly updated - I think running `fink self-update` should help – dbr Dec 20, 2009 at 17:35

32 votes

Git Reset

Say you make a pull, merge it into your code, and decide you don't like it. Use git-log, or tig, and find the hash of wherever you want to go back to (probably your last commit before the pull/merge) copy the hash, and do:

```
# Revert to a previous commit by hash:
git-reset --hard <hash>
```

Instead of the hash, you can use **HEAD^** as a shortcut for the previous commit.

```
# Revert to previous commit:
git-reset --hard HEAD^
```

Share

edited Mar 17, 2010 at 7:22

answered Nov 27, 2008 at 14:25

Dean Rather

32.3k ● 15 ● 68 ● 72

4 This is the analog to a revert in most other centralized version control systems. – Jeremy Wall Aug 29, 2009 at 1:52

"$ git-reset --hard HEAD^" should be shorthand for the parent of head (i.e. the previous state before last commit). – Ben Page Dec 16, 2009 at 0:42

6 just a plain old `git reset` should unstage accidental `git add` – slf Mar 26, 2010 at 2:43

31
votes

How do you set up a shared team repository?

How to set up a *normal* repository is described here -- but how do you set up a team repository that everybody can pull and push from and to?

Using a shared NFS file system

Assuming your team already has for instance a shared group membership that can be used.

```
mkdir /your/share/folder/project.git
cd /your/share/folder/project.git
newgrp yourteamgroup # if necessary
git init --bare --shared
```

To start using this repository the easiest thing to do is start from a local repository you already have been using:

```
cd your/local/workspace/project
git remote add origin
/your/share/folder/project.git
git push origin master
```

Others can now clone this and start working:

```
cd your/local/workspace
git clone /your/share/folder/project.git
```

Using SSH

Set up a user account on the target server. Whether you use an account with no password, an account with a password, or use `authorized_keys` really depend on your required level of security. Take a look at Configuring Git over SSH for some more information.

If all developers use the same account for accessing this shared repository, you do not need to use the `--shared` option as above.

After initing the repository in the same way as above, you do the initial push like this:

```
cd your/local/workspace/project
git remote add origin
user@server:/path/to/project.git
git push origin master
```

See the similarity with the above? The only thing that might happen in addition is SSH asking for a password if the account has a password. If you get this prompt on an account without a password the SSH server probably has disabled `PermitEmptyPasswords`.

Cloning now looks like this:

```
cd your/local/workspace
git clone user@server:/path/to/project.git
```

Share

edited May 23, 2017 at 12:03

community wiki
3 revs
Asgeir S. Nilsen

besides NFS - how do you set up git server to work over ssh? - Like a tiny-scale instance of github.com? – Dafydd Rees Jun 3, 2010 at 11:34

Is it necessary to have a group sticky bit set on the relevant directories, or does git take care of all that? If the latter, how does git know what group to use on the permissions for the Unix files? – Norman Ramsey Jun 3, 2010 at 14:38

I've added a section on SSH as requested. The sticky bit is needed if not all developers have the shared group as their primary group. If any of the users have a different primary group they would by default create files with this group ownership. This happens below git and is thus not always within git's control. – Asgeir S. Nilsen Jun 3, 2010 at 20:48

what *git repo-config core.sharedRepository group* is useful for? – systempuntoout Jul 6, 2010 at 22:11

28
votes

`git status` is your friend, use it often. Good for answering questions like:

- What did that command just do?
- What branch am I on?
- What changes am I about to commit, and have I forgotten anything?

- Was I in the middle of something last time I worked on this project (days, weeks, or perhaps months ago)?

Unlike, say `svn status`, `git status` runs nigh-instantly even on large projects. I often found it reassuring while learning git to use it frequently, to make sure my mental model of what was going on was accurate. Now I mostly just use it to remind myself what I've changed since my last commit.

Obviously, it's much more useful if your .gitignore is sanely configured.

Share

edited Feb 17, 2009 at 0:30

answered Nov 26, 2008 at 1:20

Peter Burns
45.3k ● 7 ● 40 ● 56

27 votes

Commit Changes

Once you've edited a file, you need to commit your changes to git. When you execute this command it will ask for a commit message - which is just a simple bit of text that tells everyone what you've changed.

```
$ git commit source/main.c
```

Will commit the file main.c in the directory ./source/

```
$ git commit -a # the -a flag pulls in all modified
files
```

will commit all changed files (but not new files, those need to be added to the index with git-add). If you want to commit only certain files then you will need to stage them first with git-add and then commit without the -a flag.

Commiting only changes your local repository though not the remote repositories. If you want to send the commits to the remote repository then you will need to do a push.

```
$ git push <remote> <branch> # push new commits to
the <branch> on the <remote> repository
```

For someone coming from CVS or SVN this is a change since the commit to the central repository now requires two steps.

Share

edited Aug 29, 2009 at 1:40

community wiki
4 revs, 4 users 47%
Adam Davis

27
votes

How do you branch?

The default branch in a git repository is called `master`.

To create a new branch use

```
git branch <branch-name>
```

To see a list of all branches in the current repository type

```
git branch
```

If you want to switch to another branch you can use

```
git checkout <branch-name>
```

To create a new branch and switch to it in one step

```
git checkout -b <branch-name>
```

To delete a branch, use

```
git branch -d <branch-name>
```

To create a branch with the changes from the current branch, do

```
git stash
git stash branch <branch-name>
```

Share

edited May 15, 2010 at 9:49

answered May 3, 2009 at 8:56

Markus Dulghier
1,580 ● 1 ● 17 ● 20

11 you should mention the shortcut git checkout -b <branch-name> which creates a branch and switches to it in one step. It's probably the most common use case for a beginner and even advanced git user. – Jeremy Wall Aug 29, 2009 at 1:46

21 votes

Getting the latest Code

```
$ git pull <remote> <branch> # fetches the code and merges it into
                             # your working directory
$ git fetch <remote> <branch> # fetches the code but does not merge
                              # it into your working directory

$ git pull --tag <remote> <branch> # same as above but fetch tags as well
$ git fetch --tag <remote> <branch> # you get the idea
```

That pretty much covers every case for getting the latest copy of the code from the remote repository.

Share

edited Aug 29, 2009 at 1:41

answered Aug 29, 2009 at 1:33

Jeremy Wall
25.2k ● 5 ● 57 ● 75

20 votes

The Pro Git free book is definitely my favorite, especially for beginners.

Share

edited Aug 29, 2010 at 0:13

community wiki
2 revs, 2 users 67%
Peter Mortensen

18 votes

Git Magic is all you'll ever need. Guaranteed or your money back!

Share

answered Nov 25, 2008 at 1:02

Andrew
1,217 ● 10 ● 17

14 Sigh, I want my money back. Buggy software (msysGit) with an incomplete tutorial (GitMagic) == hours of work, which is hardly free – SamGoody Apr 23, 2009 at 14:30

16 votes

How do you merge branches?

If you want to merge a branch (e.g. `master` to `release`), make sure your current branch is the target branch you'd like to merge into (use `git branch` or `git status` to see your current branch).

Then use

```
git merge master
```

(where `master` is the name of the branch you want to merge with the current branch).

If there are any conflicts, you can use

```
git diff
```

to see pending conflicts you have to resolve.

Share

answered May 3, 2009 at 9:08

Markus Dulghier
1,580 ● 1 ● 17 ● 20

2 There is git mergetool which does a three-way-diff with your favourite tool (gvimdiff, kdiff3 or some more) – Dave Vogt Oct 26, 2009 at 13:04

16
votes

I've also found [Git Internals](#) to be very useful. It is written by Scott Chacon (author of Pro Git, and maintainer of the Git Community Book). What I like about Git Internals is it [focuses on the concepts first and then the commands](#), and being that it is ~100 small pages it is quickly digestible.

Share

answered Oct 31, 2009 at 17:21

community wiki
Jordan

13
votes

How do you see the history of revisions to a file?

```
git log -- filename
```

Share

answered Jan 22, 2010 at 3:59

community wiki
Pierre-Antoine LaFayette

12
votes

How to track remote branches

Assuming there is a remote repository that you cloned your local repository from and also assuming that there is a branch named 'some_branch' on that remote repository, here is how to track it locally:

```
# list remote branches
git branch -r

# start tracking one remote branch
```

```
git branch --track some_branch origin/some_branch

# change to the branch locally
git checkout some_branch

# make changes and commit them locally
....

# push your changes to the remote repository:
git push
```

Share

answered Oct 19, 2009 at 20:02

community wiki
innaM

It seems that in git 1.7 remote branches get automatically tracked when you make a local branch from them. I don't know in which version this behavior started. – Doppelganger Jul 29, 2010 at 19:50

Actually, you can list all remote branches by using `git remote show REMOTENAME` – Felipe Sabino May 13, 2011 at 10:47

11 votes

A real good paper for understanding how Git works is The Git Parable. Very recommended!

Share

edited Aug 9, 2010 at 14:10

community wiki

2 revs, 2 users 75%
EricSchaefer

10
votes

How do you compare two revisions of a file, or your current file and a previous revision?

Compare command is `git diff`.

To compare 2 revisions of a file:

```
$ git diff <commit1> <commit2> <file_name>
```

That diffs commit1 against commit2; if you change order then files are diffed the other way round, which may not be what you expect...

To compare current staged file against the repository:

```
$ git diff --staged <file_name>
```

To compare current unstaged file against the repository:

```
$ git diff <file_name>
```

Share

answered Nov 19, 2009 at 11:21

community wiki
Andrzej Undzillo

9
votes

Why yet another howto? There are really good ones on the net, like the git guide which is perfect to begin. It has good links including the git book to which one can contribute (hosted on git hub) and which is perfect for this collective task.

On stackoverflow, I would really prefer to see your favorite tricks !

Mine, which I discovered only lately, is `git stash`, explained here, which enables you to save your current job and go to another branch

EDIT: as the previous post, if you really prefer stackoverlow format with posts as a wiki I will delete this answer

Share

answered Nov 25, 2008 at 0:52
Piotr Lesnicki
9,730 ● 2 ● 30 ● 26

No, don't delete. Your answer is perfectly valid - and pointing others to good resources isn't a bad thing. I would also like the most common operations listed here, but it's a bit of work and I don't *expect* others to do it. I'll do it over time as I learn and this'll be a reference for me. – Adam Davis Nov 25, 2008 at 1:02

9
votes

Console UI - Tig

Installation:

```
apt-get install tig
```

Usage

While inside a git repo, type 'tig', to view an interactive log, hit 'enter' on any log to see more information about it. **h** for help, which lists the basic functionality.

Trivia

"Tig" is "Git" backwards.

Share

edited Nov 27, 2008 at 14:21

answered Nov 27, 2008 at 5:23

Dean Rather
32.3k ● 15 ● 68 ● 72

Shouldn't it be a "Console UI", since "console" and "graphical" are a bit.. contradictory? – dbr Nov 27, 2008 at 13:29

it's a lot more graphical than git-log... however, it is a lot lot more interfaceable... – Dean Rather Nov 27, 2008 at 14:21

8
votes

How can I create a branch on a remote repository?

Assuming that you have cloned your remote repository from some single remote repository.

```
# create a new branch locally
git branch name_of_branch
git checkout name_of_branch
# edit/add/remove files
# ...
# Commit your changes locally
git add fileName
git commit -m Message
# push changes and new branch to remote repository:
git push origin name_of_branch:name_of_branch
```

Share

answered Oct 19, 2009 at 20:05

community wiki
innaM

11 why name_of_branch:name_of_branch ? – Seun Osewa Mar 15, 2010 at 13:18

Yes, why? As far as I know you only need `git push origin name_of_branch` and the branch will already be created in your remote – Felipe Sabino May 13, 2011 at 10:51

the first `name_of_branch` is the local name, the second is the (desired) remote branch name, so it could be `local_name_of_branch:remote_name_of_branch` if you want the names to differ. If you want them the same, you still have to specify it like this b/c git doesn't make the assumption that you want the name to be the same unless you tell it so (there are other methods to do so as well, however) – johnny Aug 18, 2011 at 16:00

8 votes

I got started with the official Git tutorial. I think it's practical enough for beginners (I was, and still am, a beginner, by your definition! I barely grasp makefiles, I've only played a bit with Apache Subversion, etc.).

Share

edited Aug 29, 2010 at 0:15
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Apr 6, 2009 at 3:56
hasen
166k ● 66 ● 196 ● 233

8 votes

How do I delete a branch on a remote repository?

Perform a push in your remote using `:` before the name of the branch

```
git push origin :mybranchname
```

being `origin` the name of your remote and `mybranchname` the name of the branch about to be deleted

http://help.github.com/remotes/

Share

answered May 12, 2011 at 11:58

community wiki
Felipe Sabino

7
votes

Push and pull changes

In an simplified way, just do `git push` and `git pull`. Changes are merged and if there's a conflict git will let you know and you can resolve it manually.

When you first push to a remote repository you need to do a `git push origin master` (master being the master branch). From then on you just do the `git push`.

Push tags with `git push --tags`.

Share

answered Nov 27, 2008 at 13:21

dylanfm
6,334 ● 5 ● 30 ● 29

7
votes

Checking Out Code

First go to an empty dir, use "git init" to make it a repository, then clone the remote repo into your own.

```
git clone user@host.com:/dir/to/repo
```

Wherever you initially clone from is where "git pull" will pull from by default.

Share

answered Nov 27, 2008 at 14:27

Dean Rather
32.3k ● 15 ● 68 ● 72

7 I think clone does the init step for you removing the need to run init first. git init is really mostly for creating the first repository or for special configurations with multiple remotes that you want to set up different than a standard clone. – Jeremy Wall Aug 29, 2009 at 1:51

7
votes

Gity: http://macendeavor.com/gity

Share

answered Mar 27, 2010 at 23:18

community wiki
gngrwzrd