

---

# Git Revision Control

Raymond E. Marcil  
<marcilr@gmail.com>

Revision: 425 (ce8f129)  
September 30, 2016

## Abstract

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It out-classes SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.<sup>1</sup>

---

<sup>1</sup>Git - <https://git-scm.com/>

# Contents

<b>Contents</b>	<b>2</b>
<b>List of Figures</b>	<b>3</b>
<b>List of Tables</b>	<b>3</b>
<b>List of Definitions and Abbreviations</b>	<b>4</b>
<b>Introduction</b>	<b>5</b>
Repositories . . . . .	6
Workflow . . . . .	6
<b>Branching &amp; Tagging</b>	<b>7</b>
<b>Remotes</b>	<b>8</b>
Bitbucket . . . . .	8
GitHub . . . . .	8
Caching your GitHub password in Git . . . . .	9
Create a Github Repo from the Command Line . . . . .	10
<b>Appendix</b>	<b>14</b>

**List of Figures**

**List of Tables**

## List of Definitions and Abbreviations

- **Branch** - [FIXME: Need data]
- **Git** - Quoting Linus: I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'. ('git' is British slang for "pig headed, think they are always correct, argumentative").<sup>2</sup>
- **Repo** - "The multiple repository tool. Repo is a tool that we built on top of Git. Repo helps us manage the many Git repositories, does the uploads to our revision control system, and automates parts of the Android development workflow. Repo is not meant to replace Git, only to make it easier to work with Git in the context of Android. The repo command is an executable Python script that you can put anywhere in your path."  
<https://code.google.com/p/git-repo/>
- **Tag** - [FIXME: Need data]

---

<sup>2</sup>Git FAQ

[https://git.wiki.kernel.org/index.php/GitFaq#Why\\_the\\_.27Git.27\\_name.3F](https://git.wiki.kernel.org/index.php/GitFaq#Why_the_.27Git.27_name.3F)

# Introduction

[FIXME: Update with great git architecture details from:  
<https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>]

Git is a distributed revision control system with an emphasis on speed,<sup>3</sup> data integrity,<sup>4</sup> and support for distributed, non-linear workflows.<sup>5</sup> Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become one of the most widely adopted version control systems for software development.<sup>6</sup>

As with most other distributed revision control systems, and unlike most clientserver systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.<sup>7</sup> Like the Linux kernel, Git is free software distributed under the terms of the GNU General Public License version 2.<sup>8</sup>

---

<sup>3</sup> Torvalds, Linus (2005-04-07). “Re: Kernel SCM saga...” linux-kernel (Mailing list). “So I’m writing some scripts to try to track things a whole lot faster.”

<sup>4</sup> Torvalds, Linus (2007-06-10). “Re: fatal: serious inflate inconsistency”. git (Mailing list). A brief description of Git’s data integrity design goals.

<sup>5</sup>Linus Torvalds (2007-05-03). [Google tech talk: Linus Torvalds on git](#). Event occurs at 02:30. Retrieved 2007-05-16.

<sup>6</sup> “[Eclipse Community Survey 2014 results — Ian Skerrett](#)”. ianskerrett.wordpress.com. 2014-06-23. Retrieved 2014-06-23.

<sup>7</sup>Chacon, Scott (24 December 2014). [Pro Git](#) (2nd ed.). New York, NY: Apress. pp. 2930. ISBN 978-1484200773.

<sup>8</sup>Git (software), From Wikipedia, the free encyclopedia, [https://en.wikipedia.org/wiki/Git\\_\(software\)](https://en.wikipedia.org/wiki/Git_(software))

## Repositories

The purpose of Git is to manage a project, or a set of files, as they change over time. Git stores this information in a data structure called a repository.<sup>9</sup>

A git repository contains, among other things, the following:

- A set of commit objects.
- A set of references to commit objects, called heads.

The Git repository is stored in the same directory as the project itself, in a subdirectory called `.git`. Note differences from central-repository systems like CVS or Subversion:

- There is only one `.git` directory, in the root directory of the project.
- The repository is stored in files alongside the project. There is no central server repository.

## Workflow

The usual git workflow looks like:

1. Do some programming.
2. `git status` to see what files I changed.
3. `git diff [file]` to see exactly what I modified.
4. `git commit -a -m [message]` to commit.

---

<sup>9</sup>Understanding Git: Repositories,  
<http://getliner.com/qXYnW/>

# Branching & Tagging

In short: Best practice is branch out, merge often and keep always in sync.

There are pretty clear conventions about keeping your code in a separate branches from master branch:

1. You are about to make an implementation of major or disruptive change
2. You are about to make some changes that might not be used
3. You want to experiment on something that you are not sure it will work
4. When you are told to branch out, others might have something they need to do in master

Rule of thumb is after branching out, you should keep in sync with the master branch. Because eventually you need to merge it back to master. In order to avoid a huge complicated mess of conflicts when merging back, you should commit often, merge often.<sup>10</sup>

---

<sup>10</sup>Git branching and tagging best practices  
<http://programmers.stackexchange.com/questions/165725/git-branching-and-tagging-best-practices>

## Remotes

“To be able to collaborate on any Git project, you need to know how to manage your remote repositories. Remote repositories are versions of your project that are hosted on the Internet or network somewhere. You can have several of them, each of which generally is either read-only or read/write for you. Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work. Managing remote repositories includes knowing how to add remote repositories, remove remotes that are no longer valid, manage various remote branches and define them as being tracked or not, and more. In this section, we’ll cover some of these remote-management skills.”<sup>11</sup>

A remote URL is Git’s fancy way of saying “the place where your code is stored.” That URL could be your repository on GitHub, or another user’s fork, or even on a completely different server.<sup>12</sup>

Git associates a remote URL with a name, and your default remote is usually called `origin`.<sup>13</sup>

A remote repository provides easy access from distributed locations and alleviates backup issues. Candidates for a remote repositories include Bitbucket,<sup>14</sup> GitHub,<sup>15</sup> or Google Code.<sup>16</sup>

## Bitbucket

[FIXME: Need data here.]

## GitHub

[FIXME: Still need cli list, rename, and delete functionality.]

GitHub’s collaborative approach to development depends on publishing commits from your local repository for other people to view, fetch, and update.<sup>17</sup>

---

<sup>11</sup>Git Basics - Working with Remotes,  
<http://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes>

<sup>12</sup>About remote repositories, <https://help.github.com/articles/about-remote-repositories/>

<sup>13</sup>Ibid.

<sup>14</sup>Bitbucket - Code, Manage, Collaborate, Bitbucket is the Git solution for professional teams  
<https://bitbucket.org/>

<sup>15</sup>GitHub - Where software is built  
<https://github.com/>

<sup>16</sup>Google Code - Provides a free collaborative development environment for open source projects.  
<https://code.google.com/>

<sup>17</sup>About remote repositories, <https://help.github.com/articles/about-remote-repositories/>



## Pushing from local repository to GitHub hosted remote

You push your local repository to the remote repository using the `git push` command after first establishing a relationship between the two with the `git remote add [alias] [url]` command. If you visit your Github repository, it will show you the URL to use for pushing. You'll first enter something like:

```
git remote add origin git@github.com:username/reponame.git
```

Unless you started by running `git clone` against the remote repository, in which case this step has been done for you already.

And after that, you'll type:

```
git push origin master
```

After your first push, you can simply type:

```
git push
```

when you want to update the remote repository in the future.

edited Jul 10 '14 at 14:17 by thomio

answered May 13 '12 at 18:01 by larsks

...

Subversion implicitly has the remote repository associated with it at all times. Git, on the other hand, allows many “remotes”, each of which represents a single remote place you can push to or pull from.

You need to add a remote for the GitHub repository to your local repository, then use `git push <REMOTE>` or `git pull <REMOTE>` to push and pull respectively - or the GUI equivalents.

...

Once you have associated the two you will be able to push or pull branches.

answered May 13 '12 at 18:01

by Daniel Pittman

## Caching your GitHub password in Git

If you're cloning GitHub repositories using HTTPS, you can use a credential helper to tell Git to remember your GitHub username and password every time it talks to GitHub.

If you clone GitHub repositories using SSH, then you authenticate using SSH keys instead of a username and password. For help setting up an SSH connection, see Generating SSH Keys.<sup>18</sup>

---

<sup>17</sup>Pushing from local repository to GitHub hosted remote,  
<http://stackoverflow.com/questions/10573957/pushing-from-local-repository-to-github-hosted-remote>

<sup>18</sup>Generating SSH keys, <https://help.github.com/articles/generating-ssh-keys/>

Turn on the credential helper so that Git will save your password in memory for some time. By default, Git will cache your password for 15 minutes.<sup>19</sup>

1. In Terminal, enter the following:

```
$ git config --global credential.helper cache
# Set git to use the credential memory cache
```

2. To change the default password cache timeout, enter the following:

```
$ git config --global credential.helper 'cache --timeout=3600'
# Set the cache to timeout after 1 hour (setting is in seconds)
```

[FIXME: The GitHub user and pass got saved by alternate means. How was that done?]

## Create a Github Repo from the Command Line

Creating a GitHub repository from the command line is incredibly convenient.

Googled up some simple shell script to create GitHub repo via command line:

```
"curl -u $username:$token" https://api.github.com/user/repos \
-d '{"name":"' $repo_name '"}'
```

To use, you could simply replace `$username` with your GitHub username, `$token` with a Personal Access Token<sup>20</sup> for the same user (available for generation in your GitHub Settings > Applications), and `$repo_name` with your desired new Repository name.<sup>21</sup>

Creating a repo from the command line is definitely faster than going to Github and using the web app to get the job done, but in order to truly make this task speedy, we need some Bash programming.

---

<sup>19</sup>Caching your GitHub password in Git, <https://help.github.com/articles/caching-your-github-password-in-git/>

<sup>20</sup>GitHub supports Personal access tokens, under Settings, click Personal access tokens. Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication. I set mine to the usual: <https://github.com/settings/tokens>

<sup>21</sup>Create a Github Repo from the Command Line, by Eli Fatsi - Jan 29, 2014, <https://viget.com/extend/create-a-github-repo-from-the-command-line>

```

github-create() {
    repo_name=$1

    dir_name='basename $(pwd)'

    if [ "$repo_name" = "" ]; then
        echo "Repo name (hit enter to use '$dir_name')?"
        read repo_name
    fi

    if [ "$repo_name" = "" ]; then
        repo_name=$dir_name
    fi

    username='git config github.user'
    if [ "$username" = "" ]; then
        echo "Could not find username, run 'git config \
        --global github.user <username>'"
        invalid_credentials=1
    fi

    token='git config github.token'
    if [ "$token" = "" ]; then
        echo "Could not find token, run 'git config \
        --global github.token <token>'"
        invalid_credentials=1
    fi

    if [ "$invalid_credentials" == "1" ]; then
        return 1
    fi

    echo -n "Creating Github repository '$repo_name' ..."
    curl -u "$username:$token" https://api.github.com/user/repos \
    -d '{"name":"'$repo_name'"}' > /dev/null 2>&1
    echo " done."

    echo -n "Pushing local code to remote ..."
    git remote add origin git@github.com:$username/$repo_name.git \
    > /dev/null 2>&1
    git push -u origin master > /dev/null 2>&1
    echo " done."
}

```

Plop this function into your `~/.bash_profile`, open a new Terminal window or source `~/.bash_profile`, and the function will be loaded up and ready for use.

Then while in an existing git project, running `github-create` will create the repo and push your master branch up in one shot. You will need to set some github config variables (instructions will be spit out if you don't have them). Heres an example:

```
BASH:projects $ rails new my_new_project
..... (a whole lot of generated things)
BASH:projects $ cd my_new_project/
BASH:my_new_project $ git init && git add . && git commit \
-m 'Initial commit'
..... (a whole lot of git additions)
BASH:my_new_project $ github-create
Repo name (hit enter to use 'my_new_project')?

Creating Github repository 'my_new_project' ... done.
Pushing local code to remote ... done.
```

Had I called the function with an argument `github-create my_project` then it would have used the argument and skipped the Repo name question.<sup>22</sup>

On GCI Network Services, OSS `covellite` Debian `jessie` 8.2 workstation the `gtihub-create` did not execute. Put `github-create()` into a standalone `~/gtihub-create` script with call to `gtihub-create()` under main.

Tested with:

```
$ mkdir ~/quux
$ cd ~/quux
$ git init
Initialized empty Git repository in /home/marcilr/quux/.git/
$ github-create
Repo name (hit enter to use 'quux')?

Could not find username, run 'git config --global github.user <username>'
Could not find token, run 'git config --global github.token <token>'
$
```

---

<sup>22</sup>Create a Github Repo from the Command Line, by Eli Fatsi - Jan 29, 2014,  
<https://viget.com/extend/create-a-github-repo-from-the-command-line>

Configured the GitHub username and token to alleviate GitHub credential errors:

```
$ git config --global github.user marcilr  
$ git config --global github.token <token>
```

Was then able to run `~/github-create` successfully:

```
$ cd ~/quux/  
$ github-create  
Repo name (hit enter to use 'quux')? <enter>  
  
Creating Github repository 'quux' ... done.  
Pushing local code to remote ... done.  
$
```

Checking GitHub via online access I found the new `quux` repo.

[FIXME: Need Bitbucket vs. GitHub section]

# Appendix

A successful Git branching model

by Vincent Driessen on Tuesday, January 05, 2010

Fine branching diagram here.

<http://nvie.com/posts/a-successful-git-branching-model/>

Bitbucket vs. GitHub: Which project host has the most?

The right choice boils down to a number of factors – you might even consider using both

<http://www.infoworld.com/article/2611771/application-development/application-development-bitbucket-vs-github-which-project-host-has-the-most->

Developing

Has Repo and Gerrit details with syntax and examples.

<http://source.android.com/source/developing.html>

Fetching a remote

```
> git clone
```

```
> git fetch
```

```
> git merge
```

```
> git pull
```

<https://help.github.com/articles/fetching-a-remote/>

Git

<https://git-scm.com/>

Git (software)

From Wikipedia, the free encyclopedia

[https://en.wikipedia.org/wiki/Git\\_\(software\)](https://en.wikipedia.org/wiki/Git_(software))

Git About

<https://git-scm.com/about>

Git branching and tagging best practices

Excellent details and semantics.

<http://programmers.stackexchange.com/questions/165725/git-branching-and-tagging-best-practices>

Git FAQ

<https://git.wiki.kernel.org/index.php/GitFaq>

Git on the Server - The Protocols, The SSH Protocol

The Git Book

<https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols>

Git (software)  
From Wikipedia, the free encyclopedia  
[https://en.wikipedia.org/wiki/Git\\_\(software\)](https://en.wikipedia.org/wiki/Git_(software))

Git repositories on Gerrit  
<https://gerrit.googlesource.com/>

GitHub  
Project host  
<https://github.com/>

How does the Android repo manifest repository work?  
<http://stackoverflow.com/questions/6149725/how-does-the-android-repo-manifest-repository-work>

Installing Repo  
<http://source.android.com/source/downloading.html#installing-repo>

Managing Remotes  
<https://help.github.com/categories/managing-remotes/>

Manifest Format for repo  
<https://gerrit.googlesource.com/git-repo/+master/docs/manifest-format.txt>

Pro Git (the git book)  
Available as [pdf](#), [epub](#), [mobi](#), and [html](#).  
<http://git-scm.com/book/en/v2/>

Re: repo + private repositories in github  
Details on manifest for google repo use.  
<https://groups.google.com/forum/embed/?#!topic/repo-discuss/kCXO-NdFvj4>

Repo Command Reference  
Using Repo and Git - very useful details here.  
<http://source.android.com/source/using-repo.html>

Repo: Tips & Tricks  
<http://xda-university.com/as-a-developer/repo-tips-tricks>

repo - The multiple repository tool  
<https://code.google.com/p/git-repo/>

Set Up Git  
>Creating a repository  
>Forking a repository

>Being social

<https://help.github.com/articles/set-up-git/>