Git
Revision Control

Raymond E. Marcil
<marcilr@gmail.com>

Revision –revision– (September 29, 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 outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.[1]

[1] Git - https://git-scm.com/

Contents

Contents 2

List of Figures 3

List of Tables 3

List of Definitions and Abbreviations 4

Introduction 5
- Repositories . . . 6
- Workflow . . . 6

Branching & Tagging 7

Remotes 8
- GitHub . . . 9
  - Caching your GitHub password in Git . . . 10
  - Create a Github Repo from the Command Line . . . 10

Commands 14
- Add . . . 14
- Clone . . . 15
- Commit . . . 17
- Diff . . . 17
- Init . . . 17
- Push . . . 18
- Remote Commands . . . 18
  - Remote add . . . 19
  - Remote set-url . . . 20
  - Remote rename . . . 22
  - Remote rm . . . 23

Appendix 24

List of Figures

List of Tables

1 Commands 14

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").[2]
- **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]

[2] Git FAQ
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,[3] data integrity,[4] and support for distributed, non-linear workflows.[5] 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.[6]

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.[7] Like the Linux kernel, Git is free software distributed under the terms of the GNU General Public License version 2.[8]

[3] 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."

[4] Torvalds, Linus (2007-06-10). "Re: fatal: serious inflate inconsistency". git (Mailing list). A brief description of Git's data integrity design goals.

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

[6] "Eclipse Community Survey 2014 results — Ian Skerrett". Ianskerrett.wordpress.com. 2014-06-23. Retrieved 2014-06-23.

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

[8] Git (software), From Wikipedia, the free encyclopedia, 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.[9]

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.

[9] 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.[10]

[10] 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."[11]

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.[12]

Git associates a remote URL with a name, and your default remote is usually called `origin`.[13]

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

[11] Git Basics - Working with Remotes,
http://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes

[12] About remote repositories, https://help.github.com/articles/about-remote-repositories/

[13] Ibid.

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

[15] GitHub - Where software is built
https://github.com/

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

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.[17]

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

[17] About remote repositories, https://help.github.com/articles/about-remote-repositories/

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

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.[18]

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.[19]

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[20] for the same user (available for generation in your GitHub Settings > Applications), and `$repo_name` with your desired new Repository name.[21]

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.

[18] Generating SSH keys, https://help.github.com/articles/generating-ssh-keys/

[19] Caching your GitHub password in Git, https://help.github.com/articles/caching-your-github-password-in-git/

[20] 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

[21] 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.[22]

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>'
$
```

[22] 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 GetHub username and token to alleviate GutHub 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]

Commands

[FIXME: Need to update with commands (log, status, mv, rm, status, and commit from: http://getliner.com/qXYnW/]

Command	Description
`add`	Add file contents to the index
`apply`	Apply a patch to files and/or to the index
`clone`	Get a complete copy of a repository
`commit`	Record changes to the repository
`diff`	Show changes between commits, commit and working tree, etc
`init`	Initialize new repository
`pull`	Fetch from and integrate with another repository or a local branch
`push`	Update remote refs along with associated objects
`rebase`	Forward-port local commits to the updated upstream head
`status`	Show the working tree status. See what files have changed.

Table 1: Commands

Add

Add file contents to the index

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.[23]

Examples

Create and add `.repo/manifests/default.xml` file and `.repo/manifests/manifest.xml`

```
covellite:~/git/.repo/manifests$ git add default.xml
covellite:~/git/.repo/manifests$ git commit default.xml
covellite:~/git/.repo/manifests$ cd ..
covellite:~/git/.repo/$ ln -s manifests/default.xml manifest.xml
covellite:~/git/.repo/$ git add manifest.xml
covellite:~/git/.repo/$ git commit
```

[23] git-add - Add file contents to the index
https://git-scm.com/docs/git-add

Clone

To grab a complete copy of another user's repository, use git clone like this:

```
$ git clone https://github.com/USERNAME/REPOSITORY.git
# Clones a repository to your computer
```

When you run `git clone`, the following actions occur:
> A new folder called repo is made
> It is initialized as a Git repository
> A remote named origin is created, pointing to the URL you cloned from
> All of the repository's files and commits are downloaded there
> The default branch (usually called master) is checked out

For every branch `foo` in the remote repository, a corresponding remote-tracking branch `refs/remotes/origin/foo` is created in your local repository. You can usually abbreviate such remote-tracking branch names to `origin/foo`.[24]

Syntax

Clone the required package by executing the following command:[25]

```
$ git clone [-b <Branch>] ssh://<Username>@review.tizen.org:\
29418/<Gerrit_Project> [<Local_Project>]
```

or

```
$ git clone ssh://<Username>@review.tizen.org:29418/pkgs/a/avsystem
```

Examples

To clone repository named `git` from GitHub to local `covellite` workstation:

```
covellite:~$ git clone https://github.com/marcilr/git.git
Cloning into 'git'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
covellite:~$
```

[24] Fetching a remote, `git clone`, `git fetch`, `git merge`, `git pull`, https://help.github.com/articles/fetching-a-remote/

[25] Cloning Tizen Source, https://source.tizen.org/documentation/developer-guide/getting-started-guide/cloning-tizen-source

Clone `.repo` repository into `git/` directory:

```
covellite:~/git$ git clone https://github.com/marcilr/.repo
Cloning into '.repo'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
covellite:~/git$
```

To clone a Git repository over SSH, you can specify ssh:// URL like this:

```
$ git clone ssh://user@server/project.git
```

Or you can use the shorter scp-like syntax for the SSH protocol:

```
$ git clone user@server:project.git
```

You can also not specify a user, and Git assumes the user you're currently logged in as.[26]

[FIXME: Need more commands here.]

[26] Git on the Server - The Protocols, The SSH Protocol,
https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols

Commit

Record changes to the repository

Stores the current contents of the index in a new commit along with a log message from the user describing the changes.[27]

Examples

Commit `spectracom.txt` file with message "`Added GCI Network Services OSS comments.`"

```
$ git commit -m "Added GCI Network Services OSS comments." spectracom.txt
[master 5267495] Added GCI Network Services OSS comments.
 1 file changed, 13 insertions(+)
$
```

Diff

Show changes between commits, commit and working tree, etc.

`git diff [file]` to see exactly what I modified.

Show changes between the working tree and the index or a tree, changes between the index and a tree, changes between two trees, changes between two blob objects, or changes between two files on disk.[28]

Synopsis

```
git diff [options] [<commit>] [--] [<path>...]
git diff [options] --cached [<commit>] [--] [<path>...]
git diff [options] <commit> <commit> [--] [<path>...]
git diff [options] <blob> <blob>
git diff [options] [--no-index] [--] <path> <path>
```

Init

Initialize new repository

The `git init` command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new empty repository. Most

[27] git-commit - Record changes to the repository,
https://git-scm.com/docs/git-commit

[28] git-diff - Show changes between commits, commit and working tree, etc
https://git-scm.com/docs/git-diff

of the other Git commands are not available outside of an initialized repository, so this is usually the first command youll run in a new project.

Executing `git init` creates a `.git` subdirectory in the project root, which contains all of the necessary metadata for the repo. Aside from the .git directory, an existing project remains unaltered (unlike SVN, Git doesn't require a `.git` folder in every subdirectory).[29]

Examples

```
$ mkdir wti
$ cd wti/
$ git init
Initialized empty Git repository in /home/marcilr/wti/.git/
$
```

Push

Update remote refs along with associated objects

[FIXME: Need for data here]

You can only push to two types of URL addresses:[30]

> An HTTPS URL like `https://github.com/user/repo.git`
> An SSH URL, like `git@github.com:user/repo.git`

Remote Commands

[29] Setting up a repository, git init, https://www.atlassian.com/git/tutorials/setting-up-a-repository

[30] About remote repositories, https://help.github.com/articles/about-remote-repositories/

Remote add

To add a new remote, use the `git remote add` command on the terminal, in the directory your repository is stored at.[31]

The `git remote add` command takes two arguments:
`$ git remote add <NAME> <REMOTE_URL>`

Where:

`<NAME>` - A remote name, for example, `origin`
`<REMOTE_URL>` - A remote URL, for example, `https://github.com/user/repo.git`

Git associates a remote URL with a name, and your default remote is usually called origin.[32]

Examples

```
# This associates the name origin with SSH URL for repo.git repository.
$ git remote add origin git@github.com:user/repo.git
```

Alternatively using https syntax:

```
$ git remote add origin https://github.com/user/repo.git
# Set a new remote

$ git remote -v
# Verify new remote
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)
```

[31] Adding a remote, https://help.github.com/articles/adding-a-remote/
[32] About remote repositories, https://help.github.com/articles/about-remote-repositories/

Remote set-url

The `git remote set-url` command changes an existing remote repository URL.[33]

The git remote set-url command takes two arguments:
> An existing remote name. For example, `origin` or `upstream` are two common choices.
> A new URL for the remote. For example:

> If you're updating to use HTTPS, your URL might look like:
`https://github.com/USERNAME/OTHERREPOSITORY.git`

> If you're updating to use SSH, your URL might look like:
`git@github.com:USERNAME/OTHERREPOSITORY.git`

Examples

Switching remote URLs from SSH to HTTPS

1. Open Terminal (for Mac and Linux users) or the command prompt (for Windows users).[34]
2. Change the current working directory to your local project.
3. List your existing remotes in order to get the name of the remote you want to change.

```
$ git remote -v
# origin  git@github.com:USERNAME/REPOSITORY.git (fetch)
# origin  git@github.com:USERNAME/REPOSITORY.git (push)
```

4. Change your remote's URL from SSH to HTTPS with the git remote set-url command.

```
$ git remote set-url origin \
https://github.com/USERNAME/OTHERREPOSITORY.git
```

5. Verify that the remote URL has changed.

```
$ git remote -v
# Verify new remote URL
# origin  https://github.com/USERNAME/OTHERREPOSITORY.git (fetch)
# origin  https://github.com/USERNAME/OTHERREPOSITORY.git (push)
```

[33] Changing a remote's URL, https://help.github.com/articles/changing-a-remote-s-url/
[34] Switching remote URLs from HTTPS to SSH, https://help.github.com/articles/changing-a-remote-s-url/

Switching remote URLs from HTTPS to SSH

1. Open Terminal (for Mac and Linux users) or the command prompt (for Windows users).[35]

2. Change the current working directory to your local project.

3. List your existing remotes in order to get the name of the remote you want to change.

```
$ git remote -v
origin  https://github.com/USERNAME/REPOSITORY.git (fetch)
origin  https://github.com/USERNAME/REPOSITORY.git (push)
```

4. Change your remote's URL from HTTPS to SSH with the git remote set-url command.

```
$ git remote set-url origin \
git@github.com:USERNAME/OTHERREPOSITORY.git
```

5. Verify that the remote URL has changed.

```
$ git remote -v
# Verify new remote URL
origin  git@github.com:USERNAME/OTHERREPOSITORY.git (fetch)
origin  git@github.com:USERNAME/OTHERREPOSITORY.git (push)
```

[35] Switching remote URLs from HTTPS to SSH, https://help.github.com/articles/changing-a-remote-s-url/

Remote rename

Use the git remote rename command to rename an existing remote.[36]

The git remote rename command takes two arguments:

> An existing remote name, for example, origin A new name for the
> remote, for example, destination

Example

The examples below assume you're cloning using HTTPS, which is recommended.

```
$ git remote -v
# View existing remotes
origin  https://github.com/OWNER/REPOSITORY.git (fetch)
origin  https://github.com/OWNER/REPOSITORY.git (push)

$ git remote rename origin destination
# Change remote name from 'origin' to 'destination'

$ git remote -v
# Verify remote's new name
destination  https://github.com/OWNER/REPOSITORY.git (fetch)
destination  https://github.com/OWNER/REPOSITORY.git (push)
```

[36]Renaming a remote, https://help.github.com/articles/renaming-a-remote/

Remote rm

Use the `git remote rm` command to remove a remote URL from your repository.[37]

The `git remote rm` command takes one argument:

> A remote name, for example, destination

Example

The examples below assume you're cloning using HTTPS, which is recommended.

```
$ git remote -v
# View current remotes
origin  https://github.com/OWNER/REPOSITORY.git (fetch)
origin  https://github.com/OWNER/REPOSITORY.git (push)
destination  https://github.com/FORKER/REPOSITORY.git (fetch)
destination  https://github.com/FORKER/REPOSITORY.git (push)

$ git remote rm destination
# Remove remote
$ git remote -v
# Verify it's gone
origin  https://github.com/OWNER/REPOSITORY.git (fetch)
origin  https://github.com/OWNER/REPOSITORY.git (push)
```

Note: `git remote rm` does not delete the remote repository from the server. It simply removes the remote and its references from your local repository.

[37] Removing a remote, https://help.github.com/articles/removing-a-remote/

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-v

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)

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)

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/