



About

Documentation

- Reference
- Book
- Videos
- External Links

Blog

Downloads

Community

This book is available in [English](#).

Full translation available in [Deutsch](#), 简体中文, 繁體中文, [Français](#), 日本語, [Nederlands](#), [Русский](#), [한국어](#), [Português \(Brasil\)](#) and [Čeština](#).

Partial translations available in [Arabic](#), [Español](#), [Indonesian](#), [Italiano](#), [Suomi](#), [Македонски](#), [Ελληνικά](#), [Polski](#) and [Türkçe](#).

Translations started for [Azarbaycan dili](#), [Беларуская](#), [Català](#), [Esperanto](#), [Español \(Nicaragua\)](#), [فارسی](#), [हिन्दी](#), [Magyar](#), [Norwegian Bokmål](#), [Polski](#), [Română](#), [Српски](#), [ភាសាខ្មែរ](#), [Tagalog](#), [Tiếng Việt](#), [Українська](#) and [Ўзбекча](#).

The source of this book is [hosted on GitHub](#). Patches, suggestions and comments are welcome.

Chapters ▾ 1st Edition

.5 Git 基础 - 远程仓库的使用

远程仓库的使用

要参与任何一个 **Git** 项目的协作，必须要了解该如何管理远程仓库。远程仓库是指托管在网络上的项目仓库，可能会有好多个，其中有些你只能读，另外有些可以写。同他人协作开发某个项目时，需要管理这些远程仓库，以便推送或拉取数据，分享各自的工作进展。管理远程仓库的工作，包括添加远程库，移除废弃的远程库，管理各式远程库分支，定义是否跟踪这些分支，等等。本节我们将详细讨论远程库的管理和使用。

查看当前的远程库

要查看当前配置有哪些远程仓库，可以用 `git remote` 命令，它会列出每个远程库的简短名字。在克隆完某个项目后，至少可以看到一个名为 **origin** 的远程库，**Git** 默认使用这个名字来标识你所克隆的原始仓库：

```
$ git clone git://github.com/schacon/ticgit.git
Cloning into 'ticgit'...
remote: Reusing existing pack: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1857/1857), 374.35 KiB | 193.00 KiB/s, done.
Resolving deltas: 100% (772/772), done.
Checking connectivity... done.
$ cd ticgit
$ git remote
origin
```

也可以加上 `-v` 选项（译注：此为 `--verbose` 的简写，取首字母），显示对应的克隆地址：

```
$ git remote -v
origin  git://github.com/schacon/ticgit.git (fetch)
origin  git://github.com/schacon/ticgit.git (push)
```

如果有多个远程仓库，此命令将全部列出。比如在我的 **Grit** 项目中，可以看到：

```
$ cd grit
$ git remote -v
bakkdoor  git://github.com/bakkdoor/grit.git
cho45     git://github.com/cho45/grit.git
defunkt   git://github.com/defunkt/grit.git
koke      git://github.com/koke/grit.git
origin    git@github.com:mojombo/grit.git
```

这样一来，我就可以非常轻松地地从这些用户的仓库中，拉取他们的提交到本地。请注意，上面列出的地址只有 **origin** 用的是 **SSH URL** 链接，所以也只有这个仓库我能推送数据上去（我们会在第四章解释原因）。

添加远程仓库

要添加一个新的远程仓库，可以指定一个简单的名字，以便将来引用，运行 `git remote add [shortname] [url]`：

```
$ git remote
origin
$ git remote add pb git://github.com/paulboone/ticgit.git
$ git remote -v
origin  git://github.com/schacon/ticgit.git
pb      git://github.com/paulboone/ticgit.git
```

现在可以用字符串 **pb** 指代对应的仓库地址了。比如说，要抓取所有 **Paul** 有的，但本地仓库没有的信息，可以运行 `git fetch pb`：

```
$ git fetch pb
remote: Counting objects: 58, done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 44 (delta 24), reused 1 (delta 0)
Unpacking objects: 100% (44/44), done.
From git://github.com/paulboone/ticgit
 * [new branch]      master    -> pb/master
 * [new branch]      ticgit    -> pb/ticgit
```

现在，**Paul** 的主干分支（**master**）已经完全可以在本地访问了，对应的名字是 `pb/master`，你可以将它合并到自己的某个分支，或者切换到这个分支，看看有些什么有趣的更新。

从远程仓库抓取数据

正如之前所看到的，可以用下面的命令从远程仓库抓取数据到本地：

```
$ git fetch [remote-name]
```

此命令会到远程仓库中拉取所有你本地仓库中还没有的数据。运行完成后，你就可以在本地访问该远程仓库中的所有分支，将其中某个分支合并到本地，或者只是取出某个分支，一探究竟。（我们会在第三章详细讨论关于分支的概念和操作。）

如果是克隆了一个仓库，此命令会自动将远程仓库归于 **origin** 名下。所以，`git fetch origin` 会抓取从你上次克隆以来别人上传到此远程仓库中的所有更新（或是上次 **fetch** 以来别人提交的更新）。有一点很重要，需要记住，**fetch** 命令只是将远端的数据拉到本地仓库，并不自动合并到当前工作分支，只有当你确实准备好了，才能手工合并。

如果设置了某个分支用于跟踪某个远端仓库的分支（参见下节及第三章的内容），可以使用 `git pull` 命令自动抓取数据下来，然后将远端分支自动合并到本地仓库中当前分支。在日常工作中我们经常这么用，既快且好。实际上，默认情况下 `git clone` 命令本质上就是自动创建了本地的 **master** 分支用于跟踪远程仓库中的 **master** 分支（假设远程仓库确实有 **master** 分支）。所以一般我们运行 `git pull`，目的都是要从原始克隆的远端仓库中抓取数据后，合并到工作目录中的当前分支。

推送数据到远程仓库

项目进行到一个阶段，要同别人分享目前的成果，可以将本地仓库中的数据推送到远程仓库。实现这个任务的命令很简单：`git push [remote-name] [branch-name]`。如果要把本地的 **master** 分支推送到 **origin** 服务器上（再次说明下，克隆操作会自动使用默认的 **master** 和 **origin** 名字），可以运行下面的命令：

```
$ git push origin master
```

只有在所克隆的服务器上有写权限，或者同一时刻没有其他人在推数据，这条命令才会如期完成任务。如果你推数据前，已经有其他人推送了若干更新，那你的推送操作就会被驳回。你必须先把他们的更新抓取到本地，合并到自己的项目中，然后才可以再次推送。有关推送数据到远程仓库的详细内容见第三章。

查看远程仓库信息

我们可以通过命令 `git remote show [remote-name]` 查看某个远程仓库的详细信息，比如要看所克隆的 **origin** 仓库，可以运行：

```
$ git remote show origin
* remote origin
  URL: git://github.com/schacon/ticgit.git
  Remote branch merged with 'git pull' while on branch master
    master
  Tracked remote branches
    master
    ticgit
```

除了对应的克隆地址外，它还给出了许多额外的信息。它友善地告诉你如果是在 **master** 分支，就可以用 `git pull` 命令抓取数据合并到本地。另外还列出了所有处于跟踪状态中的远端分支。

上面的例子非常简单，而随着使用 **Git** 的深入，`git remote show` 给出的信息可能会像这样：

```
$ git remote show origin
* remote origin
  URL: git@github.com:defunkt/github.git
  Remote branch merged with 'git pull' while on branch issues
    issues
  Remote branch merged with 'git pull' while on branch master
    master
  New remote branches (next fetch will store in remotes/origin)
    caching
  Stale tracking branches (use 'git remote prune')
    libwalker
    walker2
  Tracked remote branches
    acl
    apiv2
    dashboard2
    issues
    master
    postgres
  Local branch pushed with 'git push'
    master:master
```

它告诉我们，运行 `git push` 时缺省推送的分支是什么（译注：最后两行）。它还显示了有哪些远端分支还没有同步到本地（译注：第六行的 **caching** 分支），哪些已同步到本地的远端分支在远端服务器上已被删除（译注：**Stale tracking branches** 下面的两个分支），以及运行 `git pull` 时将自动合并哪些分支（译注：前四行中列出的 **issues** 和 **master** 分支）。

远程仓库的删除和重命名

在新版 **Git** 中可以用 `git remote rename` 命令修改某个远程仓库在本地的简称，比如想把 **pb** 改成 **paul**，可以这么运行：

```
$ git remote rename pb paul
$ git remote
origin
paul
```

注意，对远程仓库的重命名，也会使对应的分支名称发生变化，原来的 `pb/master` 分支现在成了 `paul/master`。

碰到远端仓库服务器迁移，或者原来的克隆镜像不再使用，又或者某个参与者不再贡献代码，那么需要移除对应的远端仓库，可以运行 `git remote rm` 命令：

```
$ git remote rm paul
$ git remote
origin
```

[prev](#) | [next](#)