Git 托管服务

如果不想经历自己架设 Git 服务器的麻烦，网络上有几个专业的仓库托管服务可供选择。这样做有几大优点：托管账户的建立通常比较省时，方便项目的启动，而且不涉及服务器的维护和监控。即使内部创建并运行着自己的服务器，同时为开源项目提供一个公共托管站点还是有好处的 — 让开源社区更方便地找到该项目，并给予帮助。

目前，可供选择的托管服务数量繁多，各有利弊。在 Git 官方 wiki 上的 Githosting 页面有一个最新的托管服务列表：

https://git.wiki.kernel.org/index.php/GitHosting

由于本书无法全部一一介绍，而本人（译注：指本书作者 Scott Chacon。）刚好在其中一家公司工作，所以接下来我们将会介绍如何在 GitHub 上建立新账户并启动项目。至于其他托管服务大体也是这么一个过程，基本的想法都是差不多的。

GitHub 是目前为止最大的开源 Git 托管服务，并且还是少数同时提供公共代码和私有代码托管服务的站点之一，所以你可以在上面同时保存开源和商业代码。事实上，本书就是放在 GitHub 上合作编著的。（译注：本书的翻译也是放在 GitHub 上广泛协作的。）

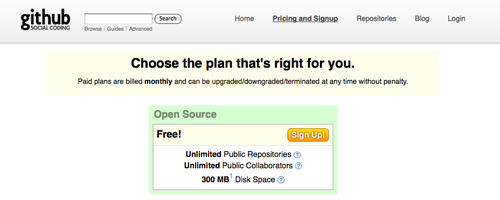
GitHub

GitHub 和大多数的代码托管站点在处理项目命名空间的方式上略有不同。GitHub 的设计更侧重于用户，而不是完全基于项目。也就是说，如果我在 GitHub 上托管一个名为 grit 的项目的话，它的地址不会是github.com/grit，而是按在用户底下 github.com/shacon/grit （译注：本书作者 Scott Chacon 在 GitHub 上的用户名是 shacon。）。不存在所谓某个项目的官方版本，所以假如第一作者放弃了某个项目，它可以无缝转移到其它用户的名下。

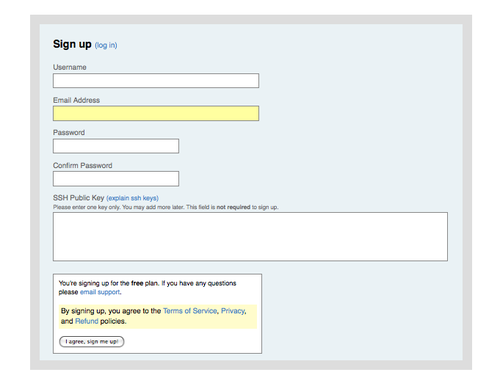
GitHub 同时也是一个向使用私有仓库的用户收取费用的商业公司，但任何人都可以方便快捷地申请到一个免费账户，并在上面托管数量不限的开源项目。接下来我们快速介绍一下 GitHub 的基本使用。

建立新账户

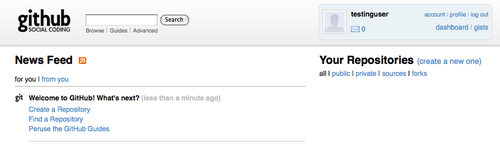
首先注册一个免费账户。访问 "Plans and pricing" 页面 https://github.com/pricing 并点击 Free acount 里的 Sign Up 按钮（见图 4-2），进入注册页面。

  
图 4-2. GitHub 服务简介页面

选择一个系统中尚未使用的用户名，提供一个与之相关联的电邮地址，并输入密码（见图 4-3）：

  
图 4-3. GitHub 用户注册表单

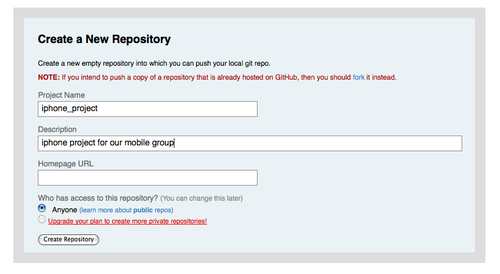
如果方便，现在就可以提供你的 SSH 公钥。我们在前文的"小型安装" 一节介绍过生成新公钥的方法。把新生成的公钥复制粘贴到 SSH Public Key 文本框中即可。要是对生成公钥的步骤不太清楚，也可以点击 "explain ssh keys" 链接，会显示各个主流操作系统上完成该步骤的介绍。 点击 "I agree，sign me up" 按钮完成用户注册，并转到该用户的 dashboard 页面（见图 4-4）:

  
图 4-4. GitHub 的用户面板

接下来就可以建立新仓库了。

建立新仓库

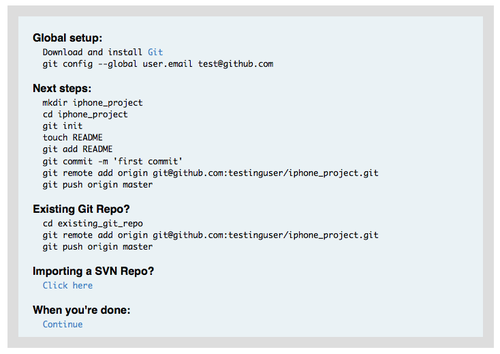
点击用户面板上仓库旁边的 "create a new one" 链接，显示 Create a New Repository 的表单（见图 4-5）：

  
图 4-5. 在 GitHub 上建立新仓库

当然，项目名称是必不可少的，此外也可以适当描述一下项目的情况或者给出官方站点的地址。然后点击 "Create Repository" 按钮，新仓库就建立起来了（见图 4-6）：

  
图 4-6. GitHub 上各个项目的概要信息

由于尚未提交代码，点击项目地址后 GitHub 会显示一个简要的指南，告诉你如何新建一个项目并推送上来，如何从现有项目推送，以及如何从一个公共的 Subversion 仓库导入项目（见图 4-7）：

  
图 4-7. 新仓库指南

该指南和本书前文介绍的类似，对于新的项目，需要先在本地初始化为 Git 项目，添加要管理的文件并作首次提交：

$ git init

$ git add .

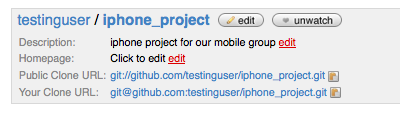
$ git commit -m 'initial commit'

然后在这个本地仓库内把 GitHub 添加为远程仓库，并推送 master 分支上来：

$ git remote add origin git@github.com:testinguser/iphone\_project.git

$ git push origin master

现在该项目就托管在 GitHub 上了。你可以把它的 URL 分享给每位对此项目感兴趣的人。本例的 URL 是http://github.com/testinguser/iphone\_project。而在项目页面的摘要部分，你会发现有两个 Git URL 地址（见图 4-8）：

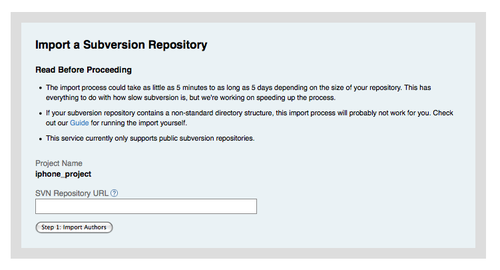
  
图 4-8. 项目摘要中的公共 URL 和私有 URL

Public Clone URL 是一个公开的，只读的 Git URL，任何人都可以通过它克隆该项目。可以随意散播这个 URL，比如发布到个人网站之类的地方等等。

Your Clone URL 是一个基于 SSH 协议的可读可写 URL，只有使用与上传的 SSH 公钥对应的密钥来连接时，才能通过它进行读写操作。其他用户访问该项目页面时只能看到之前那个公共的 URL，看不到这个私有的 URL。

从 Subversion 导入项目

如果想把某个公共 Subversion 项目导入 Git，GitHub 可以帮忙。在指南的最后有一个指向导入 Subversion 页面的链接。点击它会看到一个表单，包含有关导入流程的信息以及一个用来粘贴公共 Subversion 项目连接的文本框（见图 4-9）：

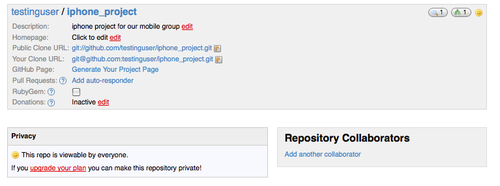
  
图 4-9. Subversion 导入界面

如果项目很大，采用非标准结构，或者是私有的，那就无法借助该工具实现导入。到第 7 章，我们会介绍如何手工导入复杂工程的具体方法。

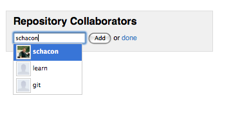
添加协作开发者

现在把团队里的其他人也加进来。如果 John，Josie 和 Jessica 都在 GitHub 注册了账户，要赋予他们对该仓库的推送权限，可以把他们加为项目协作者。这样他们就可以通过各自的公钥访问我的这个仓库了。

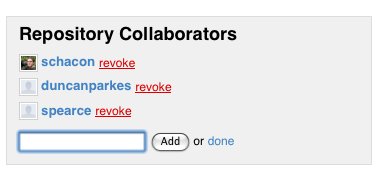
点击项目页面上方的 "edit" 按钮或者顶部的 Admin 标签，进入该项目的管理页面（见图 4-10）：

  
图 4-10. GitHub 的项目管理页面

为了给另一个用户添加项目的写权限，点击 "Add another collaborator" 链接，出现一个用于输入用户名的表单。在输入的同时，它会自动跳出一个符合条件的候选名单。找到正确用户名之后，点 Add 按钮，把该用户设为项目协作者（见图 4-11）：

  
图 4-11. 为项目添加协作者

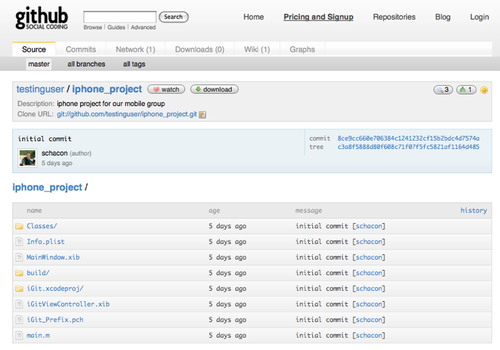
添加完协作者之后，就可以在 Repository Collaborators 区域看到他们的名单（见图 4-12）：

  
图 4-12. 项目协作者名单

如果要取消某人的访问权，点击 "revoke" 即可取消他的推送权限。对于将来的项目，你可以从现有项目复制协作者名单，或者直接借用协作者群组。

项目页面

在推送或从 Subversion 导入项目之后，你会看到一个类似图 4-13 的项目主页：

  
图 4-13. GitHub 上的项目主页

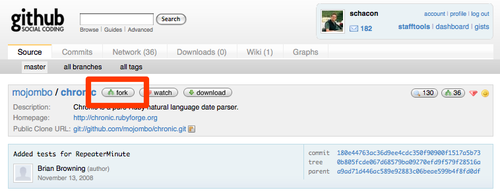
别人访问你的项目时看到的就是这个页面。它有若干导航标签，Commits 标签用于显示提交历史，最新的提交位于最上方，这和 git log 命令的输出类似。Network 标签展示所有派生了该项目并做出贡献的用户的关系图谱。Downloads 标签允许你上传项目的二进制文件，提供下载该项目各个版本的 tar/zip 包。Wiki 标签提供了一个用于撰写文档或其他项目相关信息的 wiki 站点。Graphs 标签包含了一些可视化的项目信息与数据。默认打开的 Source 标签页面，则列出了该项目的目录结构和概要信息，并在下方自动展示 README 文件的内容（如果该文件存在的话），此外还会显示最近一次提交的相关信息。

派生项目

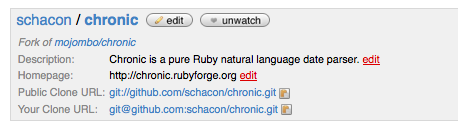
如果要为一个自己没有推送权限的项目贡献代码，GitHub 鼓励使用派生（fork）。到那个感兴趣的项目主页上，点击页面上方的 "fork" 按钮，GitHub 就会为你复制一份该项目的副本到你的仓库中，这样你就可以向自己的这个副本推送数据了。

采取这种办法的好处是，项目拥有者不必忙于应付赋予他人推送权限的工作。随便谁都可以通过派生得到一个项目副本并在其中展开工作，事后只需要项目维护者将这些副本仓库加为远程仓库，然后提取更新合并即可。

要派生一个项目，到原始项目的页面（本例中是 mojombo/chronic）点击 "fork" 按钮（见图 4-14）：

  
图 4-14. 点击 "fork" 按钮获得任意项目的可写副本

几秒钟之后，你将进入新建的项目页面，会显示该项目派生自哪一个项目（见图 4-15）：

  
图 4-15. 派生后得到的项目副本

GitHub 小结

关于 GitHub 就先介绍这么多，能够快速达成这些事情非常重要（译注：门槛的降低和完成基本任务的简单高效，对于推动开源项目的协作发展有着举足轻重的意义。）。短短几分钟内，你就能创建一个新账户，添加一个项目并开始推送。如果项目是开源的，整个庞大的开发者社区都可以立即访问它，提供各式各样的帮助和贡献。最起码，这也是一种 Git 新手立即体验尝试 Git 的捷径。