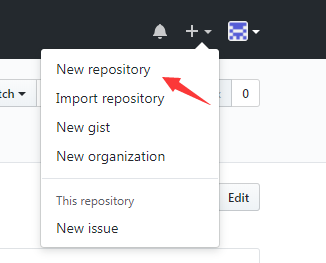
**GitHub使用指南**

**Create and use a repository/创建和使用库**

库通常用于组织单个项目。 存储库可以包含文件夹和文件，图像，视频，电子表格和数据集 - 您的项目需要的任何内容。 我们建议包括README或包含项目信息的文件。 GitHub可以在创建新存储库的同时轻松添加一个。 它还提供其他常见选项，例如许可证文件。

1. 在右上角，在您的头像或identicon旁边，单击，然后选择New repository。



2、将您的库命名为hello-world。

3、写一个简短的描述。

4、选择Initialize this repository with a README



**Start and manage a new branch/启动并管理新分支**

branch是处理不同版本的库的方法。

默认情况下，您的库有一个名为master的branch，它被认为是权威branch。 在将它们提交给master之前，我们使用branch进行实验并进行编辑。

当您从master branch创建branch时，您正在制作主服务器的副本或快照，就像在那个时间点那样。如果其他人在您的branch上工作时对master branch进行了更改，则可以引入这些更新。



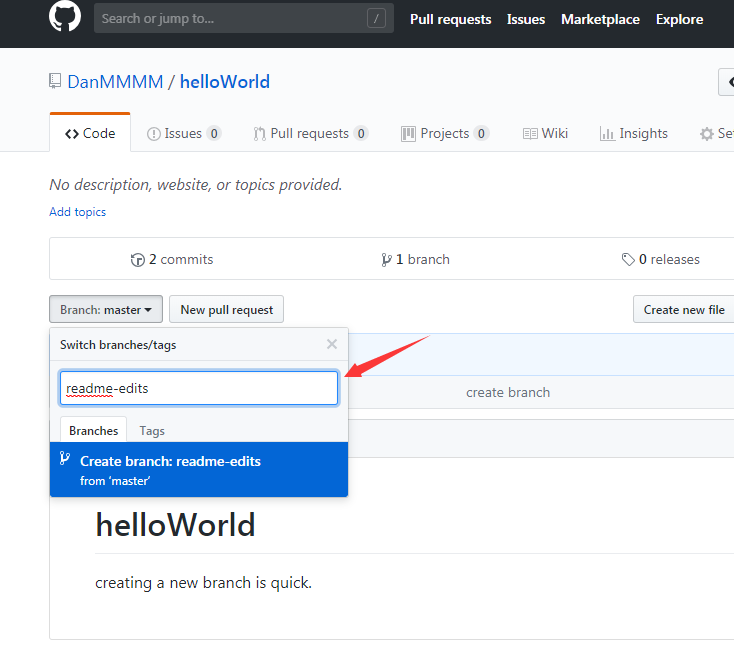
feature在合并到主服务器之前所经历的旅程

1、转到新的库hello-world。

2、单击文件列表顶部的下拉列表：master：master。

3、在新分支文本框中键入分支名称readme-edits。

4、选择蓝色的创建分支框或按键盘上的“Enter”。



Make changes to a file and push them to GitHub as commits/对文件进行更改并将其提交推送到GitHub

现在，您正在阅读readme-edits分支的代码视图，该branch是master的副本。 我们来做一些编辑。

在GitHub上，保存更改称为提交。 每个提交都有一个关联的提交消息，这也是为什么要进行特定更改的描述。 提交消息可捕获更改的历史记录，因此其他贡献者可以了解您已完成的操作以及原因。

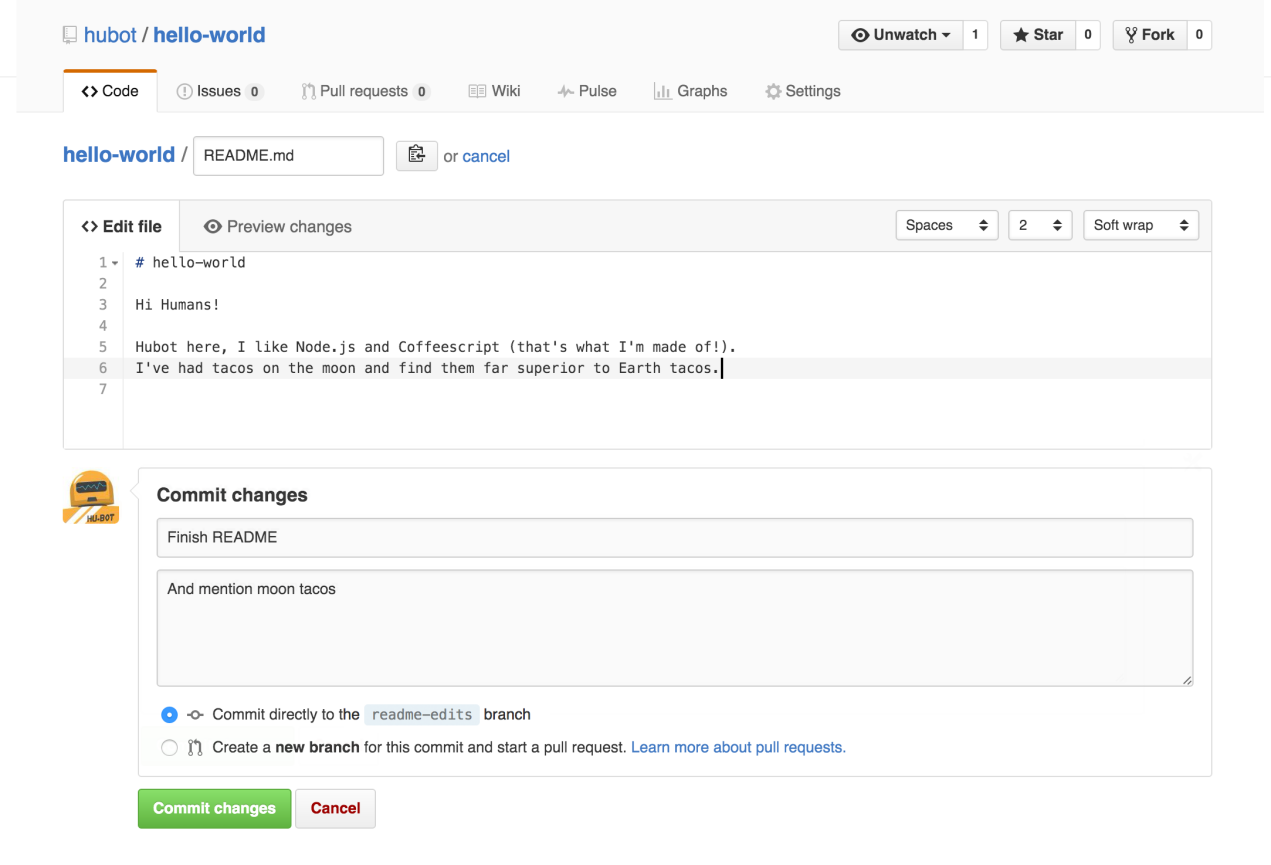
1、单击README.md文件。

2、单击文件视图右上角的铅笔图标进行编辑。

3、在编辑器中，写一点关于你自己。

4、编写描述更改的提交消息。

5、单击提交更改按钮。



Open and merge a pull request/打开,合并请求

现在您在master branch中有更改，您可以打开pull request。

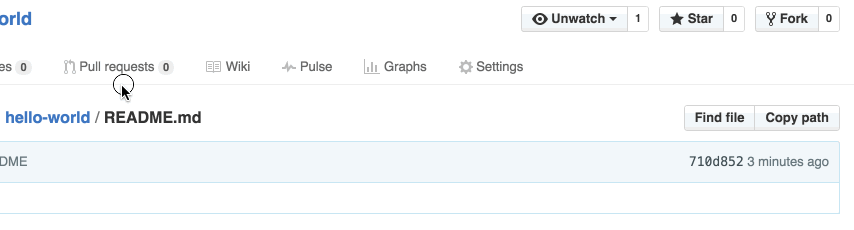
Pull Requests是GitHub上合作的核心。 当您打开拉取请求时，您提出了更改并请求某人审核并提取您的贡献并将其合并到他们的分支中。 Pull requests显示来自两个分支的内容的差异或差异。 更改，添加和减少以绿色和红色显示。

提交后，即使在代码完成之前，您也可以打开拉取请求并开始讨论。

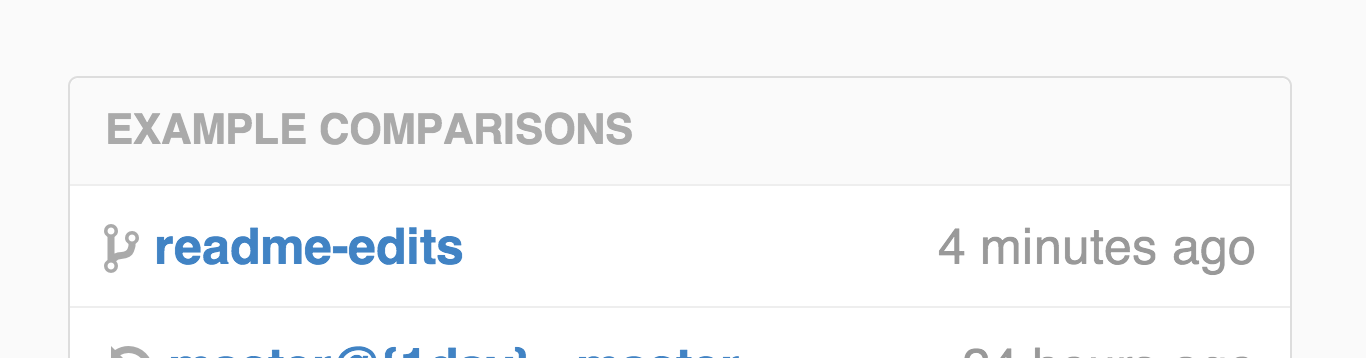
通过在pull request消息中使用GitHub的@mention系统，无论他们是在大厅还是10个时区之外，您可以询问特定人员或团队的反馈。

您甚至可以在自己的库中打开pull request并自行合并。 在开展大型项目之前，这是学习GitHub Flow的好方法。

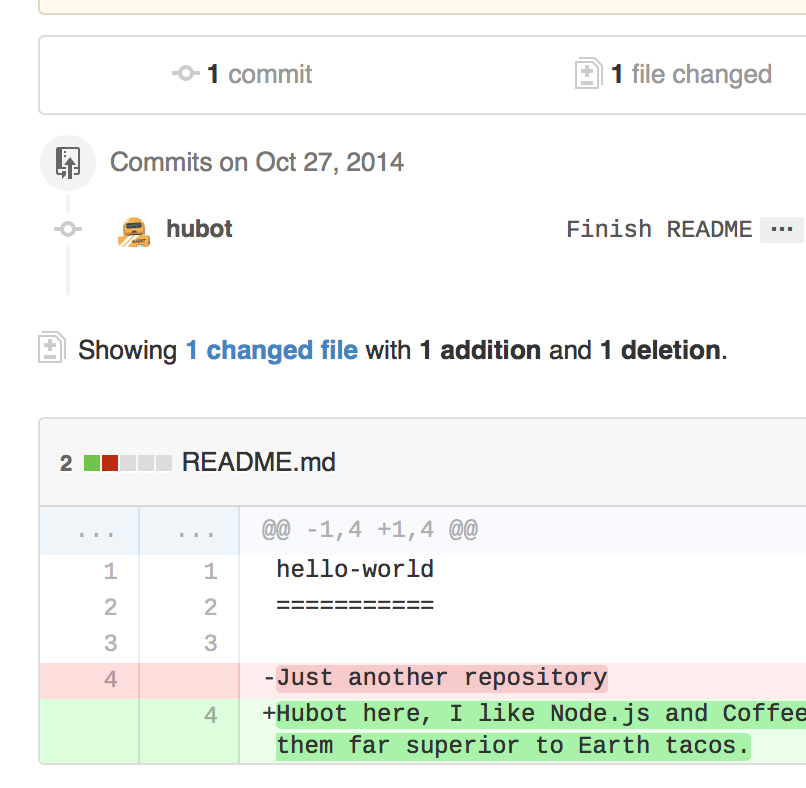
1、单击Pull Request选项卡，然后从Pull Request页面中，单击绿色New pull request按钮。



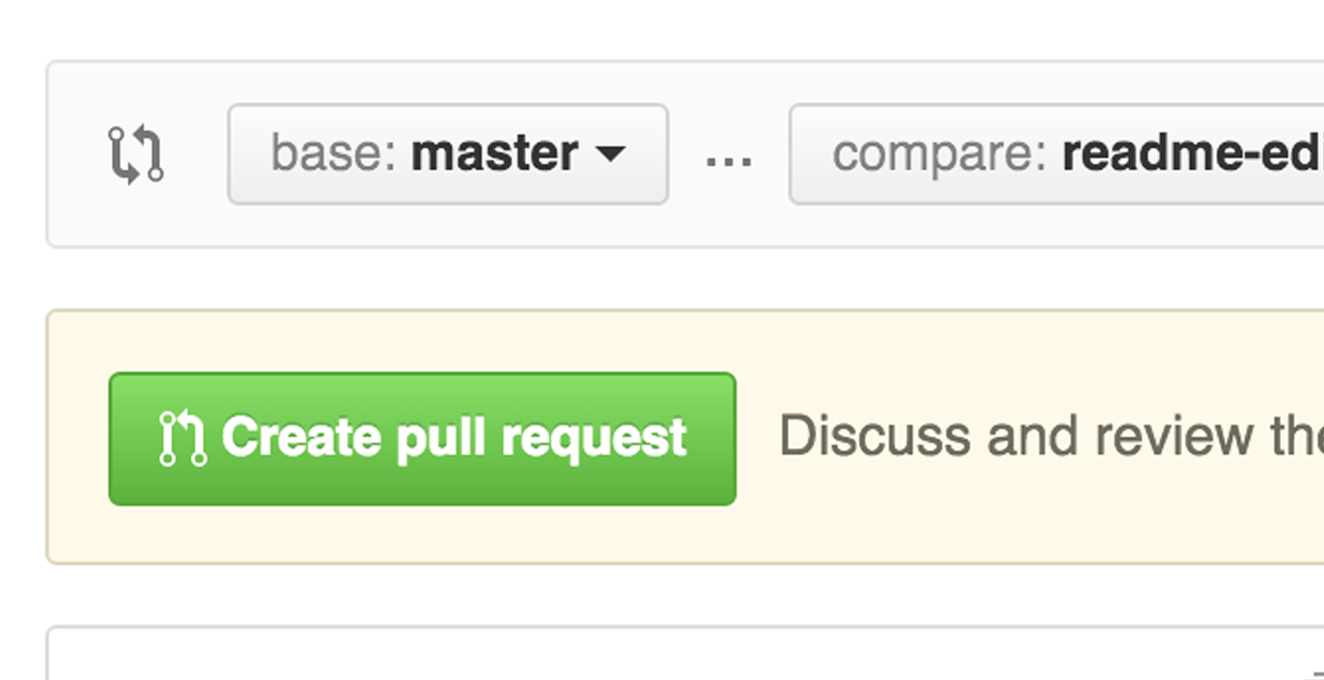
2、在 Example Comparisons中，选择您创建的branch，自述编辑，以与master（原始）进行比较。



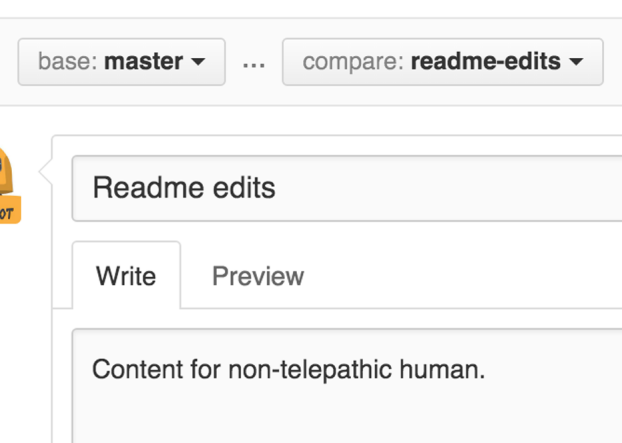
1. 在比较页面上查看差异中的差异，确保它们是您要提交的内容



3、如果您对要提交的更改感到满意，请单击绿色的Create pull request。



1. 为您的拉取请求提供标题，并写下您的更改的简要说明。

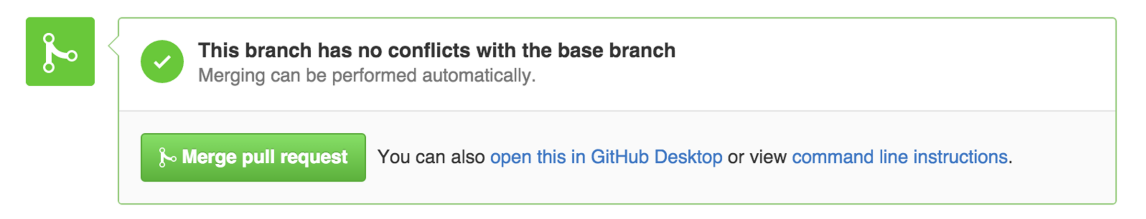


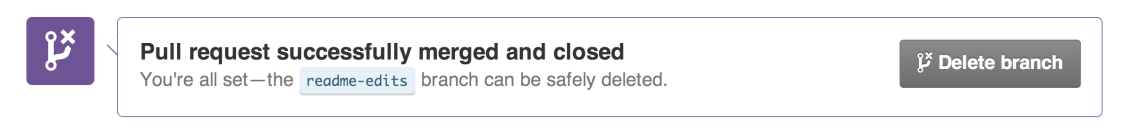
在最后一步中，是时候将您的更改结合在一起 - 将您的readme-edits branch合并到master branch中。

1、单击绿色合并拉取请求按钮以将更改合并到主文件中。

2、单击确认合并。

3、继续删除分支，因为它的更改已合并，紫色框中的删除分支按钮。





Git的简单使用

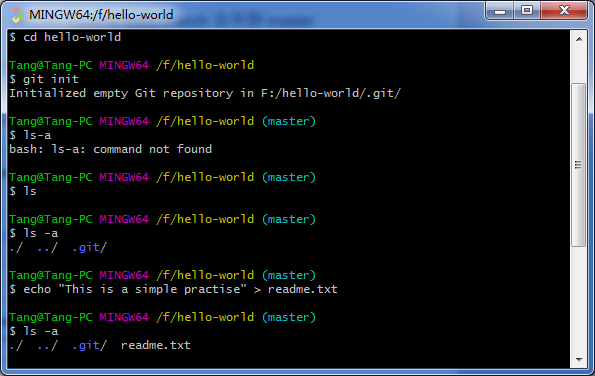
Git Bash：在你想要下载代码的路径，点击鼠标右键，选择Git Bash here。注意，你的代码路径也应是纯英文的。Git Bash使用的是MinGW。

创建一个本地hello-world仓库

1. 在命令行输入mkdir hello-word，创建一个新文件夹。ls命令：查看当前目录下有哪些文件和文件夹。

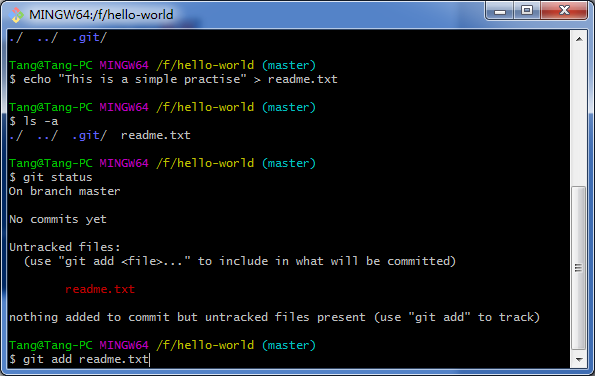
2. 输入cd hello-world进入新文件夹，注意在输入命令时，你可以用Tab键来自动补全。

3. 输入git init初始化Git仓库。ls -a：查看当前目录，可以看到多了一个.git/的文件夹，此文件夹保存了版本控制的所有相关信息。



4. 输入echo "This is a simple practise" > readme.txt，创建一个readme.txt文件。

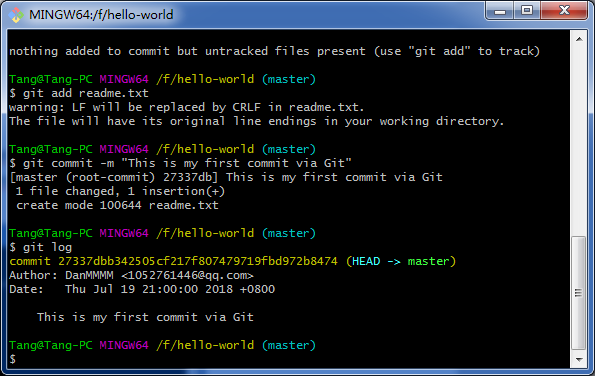
5. 输入git status查看当前版本库状态，在Untracked files(未跟踪文件)下，会出现红色的readme.txt，代表此文件还未被Git所管理。



6. 使用git add readme.txt，将该文件加入缓冲区，如果你确定所有的修改都需要提交，可以使用git add .来加入所有修改。现在用git status查看，将看到文件名变为绿色。

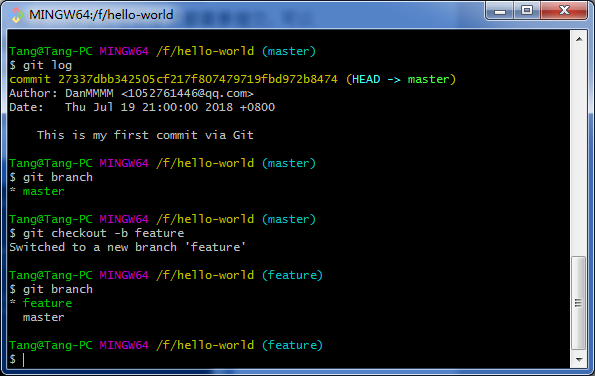
7. 使用git commit -m "This is my first commit via Git!"来提交修改，-m后面所带的参数是本次提交信息，一般用来记录本次提交的主要意图。

8. 提交成功后，可以用git log查看历史提交记录。每个记录都会有提交id，作者和提交日期。



9. 你可以用git branch查看当前有哪些分支，当然，因为我们没有创建任何分支，目前只会有一个master分支。

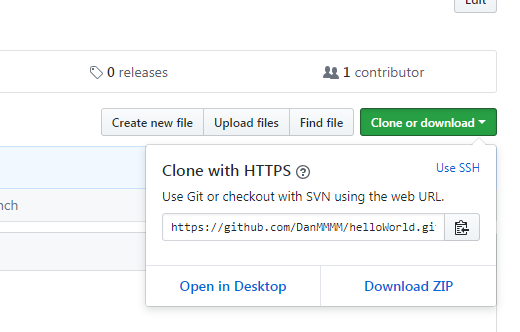
10. 使用git checkout -b feature创建一个名为feature的分支，再用git branch查看一下。



**项目的下载，查看和修改**

第一步. 从GitHub上下载我们的项目代码。

1. 以[Hello-World](https://github.com/feicui-android/Hello-World" \t "https://blog.csdn.net/javaandroid730/article/details/_blank)项目为例，点击绿色按钮Clone or download，然后在弹出窗口中点击剪切板图标，复制仓库的URL。



第二步. 查看版本历史

1. cd到项目文件夹下，使用git log能看到我们的历史提交记录。

2. 要回到某一历史版本，可以使用git checkout commitId，看完后要回到最新代码，使用git checkout master。

第三步. 本地修改代码

你可以在我们的代码基线上任意修改，但为了下载新代码时不出现冲突，请遵循以下步骤：

1. 下载新代码：git pull。

2. 从master出捡出一个新的分支：git checkout -b feature。feature是分支名称，你可以随意取名，但请用英文。

3. 在feature分支上随意修改，改完后你可以提交你的修改：git commit -m "some message"。

4. 此时要同步代码，请先切回主分支：git checkout master，然后更新git pull。

5. 如果想删除自己建立的分支，使用git branch -D feature，注意执行此命令后分支被强制删除，无法恢复。

下载新代码：git pull

查看当前分支：git branch

新建分支：git checkout -b feature，feature是分支名称

查看当前版本库状态：git status

切换分支：git checkout feature

将文件加入缓冲区：git add readme.txt

提交修改：git commit -m "This is my first commit via Git!"，m后面所带的参数是本次提交信息，一般用来记录本次提交的主要意图。

合并分支：git merge newbranch

push代码：git push -u origin master

查看历史提交记录：git log

创建一个新文件夹：mkdir hello-word

查看当前目录下文件和文件夹：ls

进入新文件夹：cd hello-world，(Tab键来自动补全)

初始化Git仓库：git init

将本地仓库和远程仓库连接起来：git remote add origin git@github.com:yourname/仓库名.git