推送至远程仓库

Git 是分散型版本管理系统，但是我们前面所学习的，都是针对单一本地仓库的操作。下面，我们将开始接触远在网络另一头的远程仓库。远程仓库顾名思义，是与我们本地仓库相对独立的另一个仓库。让我们现在 GitHub 上创建一个仓库，并将其设置为本地仓库的远程仓库。

为防止与其他仓库混淆，仓库名请与本地仓库保持一致，即 hello。创建时请不要勾选 Initialize this repository with a README 选项。因为一旦勾选该选项，GitHub 一侧的仓库就会自动生成 README 文件，从创建之初便与本地仓库失去了整合性。虽然到时也可以强制覆盖，但为防止这一情况发生还是建议不要勾选该选项，直接点击 Create repository 创建仓库。

（1）git remote add——添加远程仓库

在 GitHub 上创建的仓库路径为“git@github.com:用户名/hello.git”。现在我们用 git remote add 命令将它设置为本地仓库的远程仓库。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git remote add origin git@github.com:ZitingLou/hello.git
```

按照上述格式执行 git remote add 命令之后，Git 会自动将“git@github.com:用户名/hello.git”远程仓库的名称设置为 origin（标识符）。

（2）git push——推送至远程仓库

①推送至 master 分支

如果想将当前分支下本地仓库中的内容推送给远程仓库，需要用到 git push 命令。现在假定我们在 master 分支下进行操作。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git push -u origin master
Enter passphrase for key '/c/Users/LZT/.ssh/id_rsa':
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 544 bytes | 272.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:ZitingLou/hello.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

像这样执行 git push 命令，当前分支的内容就会被推送给远程仓库 origin 的 master 分支。-u 参数可以在推送的同时，将 origin 仓库的 master 分支设置为本地仓库当前分支的 upstream

（上游）。添加了这个参数，将来运行 git pull 命令从远程仓库获取内容时，本地仓库的这个分支就可以直接从 origin 的 master 分支获取内容，省去了另外添加参数的麻烦。

执行该操作后，当前本地仓库 master 分支的内容将会被推送到 Github 的远程仓库中。在 Github 上也可以确认远程 master 分支的内容和本地 master 分支相同。

②推送至 master 以外的分支

除了 master 分支之外，远程仓库也可以创建其他仓库。举个例子，我们在本地仓库中创建 feature-D 分支，并将它以同名形式 push 至远程仓库。

```
LZT@LZT-PC MINGW64 ~/hello (master)
$ git checkout -b feature-D
Switched to a new branch 'feature-D'
```

我们在本地仓库中创建了 feature-D 分支，现在将它 push 给远程仓库并保持分支名称不变。

```
LZT@LZT-PC MINGW64 ~/hello (feature-D)
$ git push -u origin feature-D
Enter passphrase for key '/c/Users/LZT/.ssh/id_rsa':
Total 0 (delta 0), reused 0 (delta 0)
To github.com:ZitingLou/hello.git
 * [new branch]      feature-D -> feature-D
Branch feature-D set up to track remote branch feature-D from origin.
```

现在，在远程仓库的 GitHub 页面就可以查看到 feature-D 分支了。

从远程仓库获取

上一节中我们把在 GitHub 上新建的仓库设置成了远程仓库，并向这个仓库 push 了 feature-D 分支。现在，所有能够访问这个远程仓库的人都可以获取 feature-D 分支并加以修改。

本节中我们从实际开发者的角度出发，在另一个目录下新建一个本地仓库，学习从远程仓库获取内容的相关操作。这就相当于我们刚刚执行过 push 操作的目标仓库又有了另一名新开发者来共同开发。

（1）git clone——获取远程仓库

①获取远程仓库

首先我们换到其他目录下，将 GitHub 上的仓库 clone 到本地。注意不要与之前操作的仓库在同一目录下。

```
LZT@LZT-PC MINGW64 ~/Desktop (master)
$ git clone git@github.com:ZitingLou/hello.git
Cloning into 'hello'...
Enter passphrase for key '/c/Users/LZT/.ssh/id_rsa':
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 7 (delta 1), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
```

执行 git clone 命令后我们会默认处于 master 分支下，同时系统会自动将 origin 设置成该远程仓库的标识符。也就是说，当前本地仓库的 master 分支与 GitHub 端远程仓库（origin）的 master 分支在内容上是完全相同的。

```
LZT@LZT-PC MINGW64 ~/Desktop (master)
$ cd hello

LZT@LZT-PC MINGW64 ~/Desktop/hello (master)
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/feature-D
  remotes/origin/master
```

我们用 git branch –a 命令查看当前分支的相关信息。添加-a 参数可以同时显示本地仓库和远程仓库的分支信息。

结果中显示了 remotes/origin/feature-D，证明我们的远程仓库中已经有了 feature-D 分支。

②获取远程的 feature-D 分支

我们试着将 feature-D 分支获取到本地仓库。

```
LZT@LZT-PC MINGW64 ~/Desktop/hello (master)
$ git checkout -b feature-D origin/feature-D
Switched to a new branch 'feature-D'
Branch feature-D set up to track remote branch feature-D from origin.
```

-b 参数的后面是本地仓库中新建分支的名称。为了便于理解，我们仍将其命名为 feature-D，让它与远程仓库的对应分支保持同名。新建分支名称后面是获取来源的分支名称。例子中指定了 origin/feature-D，就是说以名为 origin 的仓（这里指 GitHub 端的仓库）的 feature-D 分支为来源，在本地仓库中创建 feature-D 分支。

③向本地的 feature-D 分支提交更改

现在假定我们是另一名开发者，要做一个新的提交。在 README.md 文件中添加一行文字，查看更改。

```
#hello

-feature-C
-feature-D
```

```
LZT@LZT-PC MINGW64 ~/Desktop/hello (feature-D)
$ git diff
diff --git a/README.md b/README.md
index 0f2e074..d2a5f1b 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,4 @@
 #hello

--feature-C
\ No newline at end of file
+-feature-C
+-feature-D
\ No newline at end of file
```

按照之前学过的方式进行提交。

```
LZT@LZT-PC MINGW64 ~/Desktop/hello (feature-D)
$ git commit -am"Add feature-D"
[feature-D ec8e79d] Add feature-D
 1 file changed, 2 insertions(+), 1 deletion(-)
```

④推送 feature-D 分支

从远程仓库获取 feature-D 分支，在本地仓库中提交更改，再将 feature-D 分支推送回远程仓库，通过这一系列操作，就可以与其他开发者相互合作，共同培育 feature-D 分支，实现某些功能。

```
LZT@LZT-PC MINGW64 ~/Desktop/hello (feature-D)
$ git push
Enter passphrase for key '/c/Users/LZT/.ssh/id_rsa':
Counting objects: 3, done.
Writing objects: 100% (3/3), 264 bytes | 264.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
```

（2）git pull——获取最新的远程仓库分支

现在我们放下刚刚操作的目录，回到原先的那个目录下。这边的本地仓库中只创建了 feature-D 分支，并没有在 feature-D 分支中进行任何提交。然而远程仓库的 feature-D 分支中已经有了我们刚刚推送的提交。这时我们就可以使用 git pull 命令，将本地的 feature-D 分支更新到最新状态。当前分支为 feature-D 分支。

```
#hello

-feature-C
```

```
LZT@LZT-PC MINGW64 ~/hello (feature-D)
$ git pull origin feature-D
Enter passphrase for key '/c/Users/LZT/.ssh/id_rsa':
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:ZitingLou/hello
 * branch            feature-D  -> FETCH_HEAD
   32f0430..ec8e79d  feature-D  -> origin/feature-D
Updating 32f0430..ec8e79d
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
#hello

-feature-C
-feature-D
```

GitHub 端远程仓库中的 feature-D 分支是最新状态，所以本地仓库中的 feature-D 分支就得到了更新。今后只需要像平时一样在本地进行提交再 push 给远程仓库，就可以与其他开发者同时在同一个分支中进行作业，不断给 feature-D 增加新功能。

如果两人同时修改了同一部分的源代码，push 时就很容易发生冲突。所以多名开发者在同一分支中进行作业时，为减少冲突情况的发生，建议更频繁地进行 push 和 pull 操作。