当你从远程仓库克隆时，实际上 Git 自动把本地的 `master` 分支和远程的 `master` 分支对应起来了，并且，远程仓库的默认名称是 `origin`。

要查看远程库的信息，用 `git remote`：

```
$ git remote

origin
```

或者，用 `git remote -v` 显示更详细的信息：

```
$ git remote -v

origin  git@github.com:michaelliao/learngit.git (fetch)

origin  git@github.com:michaelliao/learngit.git (push)
```

上面显示了可以抓取和推送的 `origin` 的地址。如果没有推送权限，就看不到 push 的地址。

推送分支

推送分支，就是把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，这样，Git 就会把该分支推送到远程库对应的远程分支上：

```
$ git push origin master
```

如果要推送其他分支，比如 `dev`，就改成：

```
$ git push origin dev
```

但是，并不是一定要把本地分支往远程推送，那么，哪些分支需要推送，哪些不需要呢？

- `master` 分支是主分支，因此要时刻与远程同步；
- `dev` 分支是开发分支，团队所有成员都需要在上面工作，所以也需要与远程同步；
- bug 分支只用于在本地修复 bug，就没必要推到远程了，除非老板要看看你每周到底修复了几个 bug；
- feature 分支是否推到远程，取决于你是否和你的小伙伴合作在上面开发。

总之，就是在 Git 中，分支完全可以在本地自己藏着玩，是否推送，视你的心情而定！

抓取分支

多人协作时，大家都会往 `master` 和 `dev` 分支上推送各自的修改。

现在，模拟一个你的小伙伴，可以在另一台电脑（注意要把 SSH Key 添加到 GitHub）或者同一台电脑的另一个目录下克隆：

```
$ git clone git@github.com:michaelliao/learngit.git

Cloning into 'learngit'...

remote: Counting objects: 40, done.

remote: Compressing objects: 100% (21/21), done.

remote: Total 40 (delta 14), reused 40 (delta 14), pack-reused 0

Receiving objects: 100% (40/40), done.

Resolving deltas: 100% (14/14), done.
```

当你的小伙伴从远程库 clone 时，默认情况下，你的小伙伴只能看到本地的 `master` 分支。不信可以用 `git branch` 命令看看：

```
$ git branch

* master
```

现在，你的小伙伴要在 `dev` 分支上开发，就必须创建远程 `origin` 的 `dev` 分支到本地，于是他用这个命令创建本地 `dev` 分支：

```
$ git checkout -b dev origin/dev
```

现在，他就可以在 `dev` 上继续修改，然后，时不时地把 `dev` 分支 `push` 到远程：

```
$ git add env.txt



$ git commit -m "add env"

[dev 7a5e5dd] add env

 1 file changed, 1 insertion(+)

 create mode 100644 env.txt
```

```
$ git push origin dev

Counting objects: 3, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To github.com:michaelliao/learngit.git

   f52c633..7a5e5dd  dev -> dev
```

你的小伙伴已经向`origin/dev`分支推送了他的提交，而碰巧你也对同样的文件作了修改，并试图推送：

```
$ cat env.txt

env


$ git add env.txt


$ git commit -m "add new env"

[dev 7bd91f1] add new env

 1 file changed, 1 insertion(+)

 create mode 100644 env.txt


$ git push origin dev

To github.com:michaelliao/learngit.git

 ! [rejected]        dev -> dev (non-fast-forward)

error: failed to push some refs to 'git@github.com:michaelliao/learngit.git'
```

```
hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Integrate the remote changes (e.g.

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

推送失败，因为你的小伙伴的最新提交和你试图推送的提交有冲突，解决办法也很简单，Git 已经提示我们，先用 `git pull` 把最新的提交从 `origin/dev` 抓下来，然后，在本地合并，解决冲突，再推送：

```
$ git pull

There is no tracking information for the current branch.

Please specify which branch you want to merge with.

See git-pull(1) for details.


    git pull <remote> <branch>


If you wish to set tracking information for this branch you can do so with:


    git branch --set-upstream-to=origin/<branch> dev
```

`git pull` 也失败了，原因是没有指定本地 `dev` 分支与远程 `origin/dev` 分支的链接，根据提示，设置 `dev` 和 `origin/dev` 的链接：

```
$ git branch --set-upstream-to=origin/dev dev

Branch 'dev' set up to track remote branch 'dev' from 'origin'.
```

再 pull：

```
$ git pull

Auto-merging env.txt

CONFLICT (add/add): Merge conflict in env.txt
```

```
Automatic merge failed; fix conflicts and then commit the result.
```

这回 `git pull` 成功，但是合并有冲突，需要手动解决，解决的方法和分支管理中的解决冲突完全一样。解决后，提交，再 push：

```
$ git commit -m "fix env conflict"

[dev 57c53ab] fix env conflict


$ git push origin dev

Counting objects: 6, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (4/4), done.

Writing objects: 100% (6/6), 621 bytes | 621.00 KiB/s, done.

Total 6 (delta 0), reused 0 (delta 0)

To github.com:michaelliao/learngit.git

   7a5e5dd..57c53ab  dev -> dev
```

因此，多人协作的工作模式通常是这样：

1. 首先，可以试图用 `git push origin <branch-name>` 推送自己的修改；
2. 如果推送失败，则因为远程分支比你的本地更新，需要先用 `git pull` 试图合并；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 没有冲突或者解决掉冲突后，再用 `git push origin <branch-name>` 推送就能成功！

   如果 `git pull` 提示 `no tracking information`，则说明本地分支和远程分支的链接关系没有创建，用命令 `git branch --set-upstream-to <branch-name> origin/<branch-name>`。

   这就是多人协作的工作模式，一旦熟悉了，就非常简单。

小结

- 查看远程库信息，使用 `git remote -v`；
- 本地新建的分支如果不推送到远程，对其他人就是不可见的；

- 从本地推送分支，使用 `git push origin branch-name`，如果推送失败，先用 `git pull` 抓取远程的新提交；
- 在本地创建和远程分支对应的分支，使用 `git checkout -b branch-name origin/branch-name`，本地和远程分支的名称最好一致；
- 建立本地分支和远程分支的关联，使用 `git branch --set-upstream branch-name origin/branch-name`；
- 从远程抓取分支，使用 `git pull`，如果有冲突，要先处理冲突。