git pull命令的作用是，取回远程主机某个分支的更新，再与本地的指定分支合并。它的完整格式稍稍有点复杂。

```
$ git pull <远程主机名> <远程分支名>:<本地分支名>
```

比如，取回origin主机的next分支，与本地的master分支合并，需要写成下面这样。

```
$ git pull origin next:master
```

如果远程分支是与当前分支合并，则冒号后面的部分可以省略。

```
$ git pull origin next
```

上面命令表示，取回origin/next分支，再与当前分支合并。实质上，这等同于先做git fetch，再做git merge。

```
$ git fetch origin
$ git merge origin/next
```

在某些场合，Git会自动在本地分支与远程分支之间，建立一种追踪关系（tracking）。比如，在git clone的时候，所有本地分支默认与远程主机的同名分支，建立追踪关系，也就是说，本地的master分支自动"追踪"origin/master分支。

Git也允许手动建立追踪关系。

```
git branch --set-upstream master origin/next
```

上面命令指定master分支追踪origin/next分支。

如果当前分支与远程分支存在追踪关系，git pull就可以省略远程分支名。

```
$ git pull origin
```

上面命令表示，本地的当前分支自动与对应的origin主机"追踪分支"（remote-tracking branch）进行合并。
如果当前分支只有一个追踪分支，连远程主机名都可以省略。

```
$ git pull
```

上面命令表示，当前分支自动与唯一一个追踪分支进行合并。
如果合并需要采用rebase模式，可以使用--rebase选项。

```
$ git pull --rebase <远程主机名> <远程分支名>:<本地分支名>
```

如果远程主机删除了某个分支，默认情况下，`git pull` 不会在拉取远程分支的时候，删除对应的本地分支。这是为了防止，由于其他人操作了远程主机，导致`git pull`不知不觉删除了本地分支。
但是，你可以改变这个行为，加上参数 `-p` 就会在本地删除远程已经删除的分支。

```
$ git pull -p
# 等同于下面的命令
$ git fetch --prune origin
$ git fetch -p
```