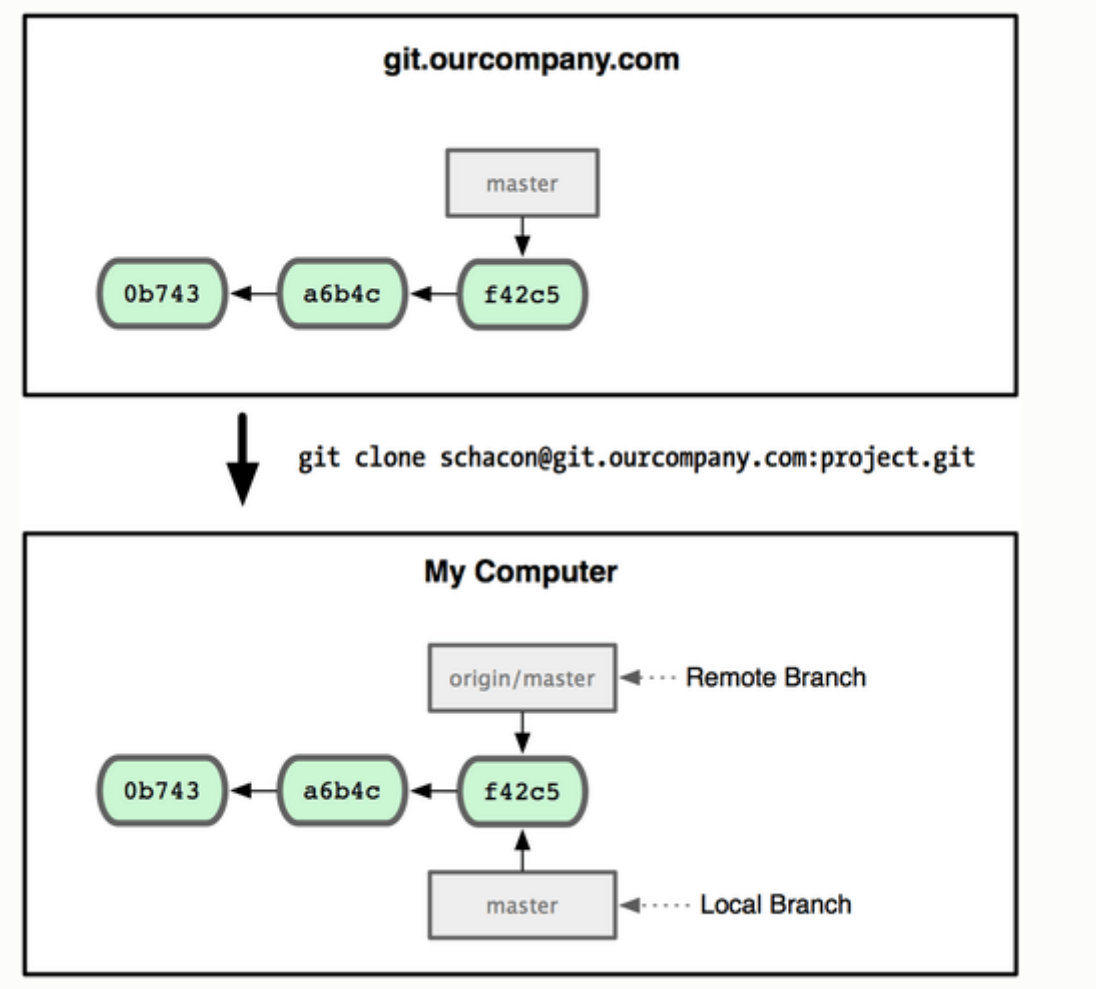
五：远程分支

- **远程分支**

远程分支（remote branch）是对远程仓库中的分支的索引。它们是一些无法移动的本地分支，只有在 Git 进行网络交互时才会更新。

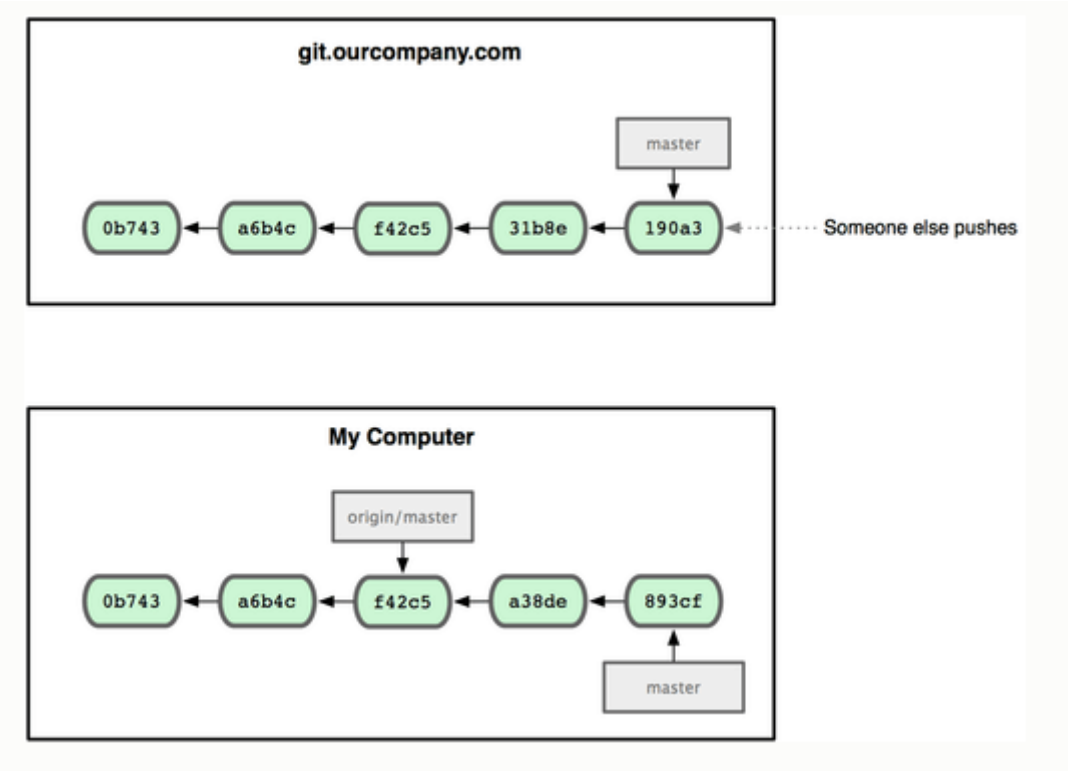
用 `<远程仓库名>/<分支名>` 这种形式表示远程分支。

几个例子：假如一个团队有个地址为 git.ourcompany.com 的git服务器。如果你从这里克隆，Git会自动为你将此远程仓库命名为 origin,并下载其中的所有数据，建立一个指向它的master分支的指针，本地命名为 origin/master ，但你无法在本地更改其数据，接着，git建立一个属于你的本地 master 分支，始于 origin上master 分支相同位置，接下来就可以展开工作了。

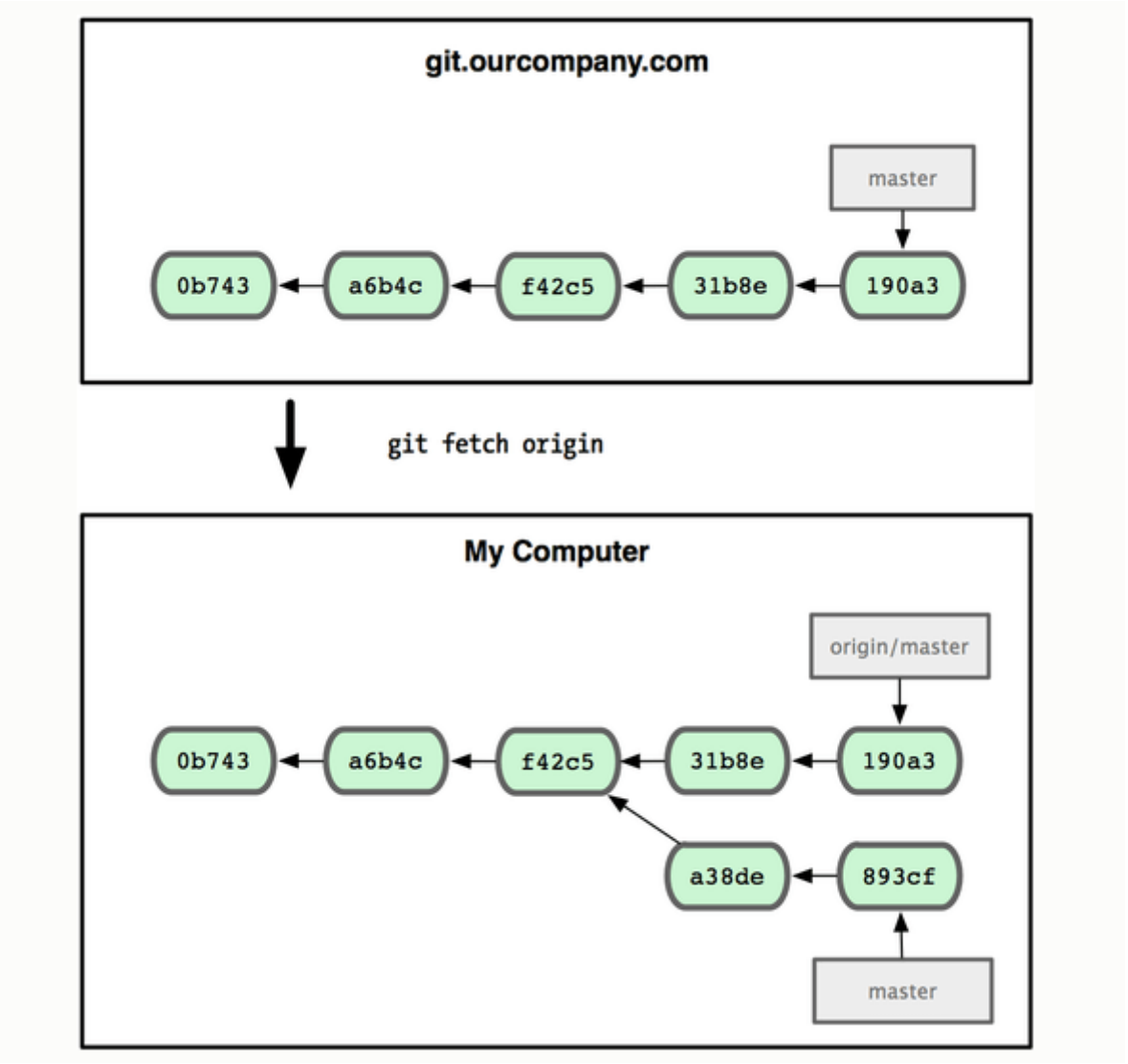


> 注意： git 的一次 clone 操作会创建 本地分支 master 和远程分支 origin/master , 并且将他们都指向 origin 上的master 分支。

如果你在本地 master 分支上做了些改动，与此同时，其他人向 git.ourcompany.com 推送了他们的更新，那么服务器上的 master分支就会向前移动，而与此同时，你的本地历史正朝不同方向发展，只要你不和服务器通讯，你的 origin/master 指针仍然保持原位不移动，如图：

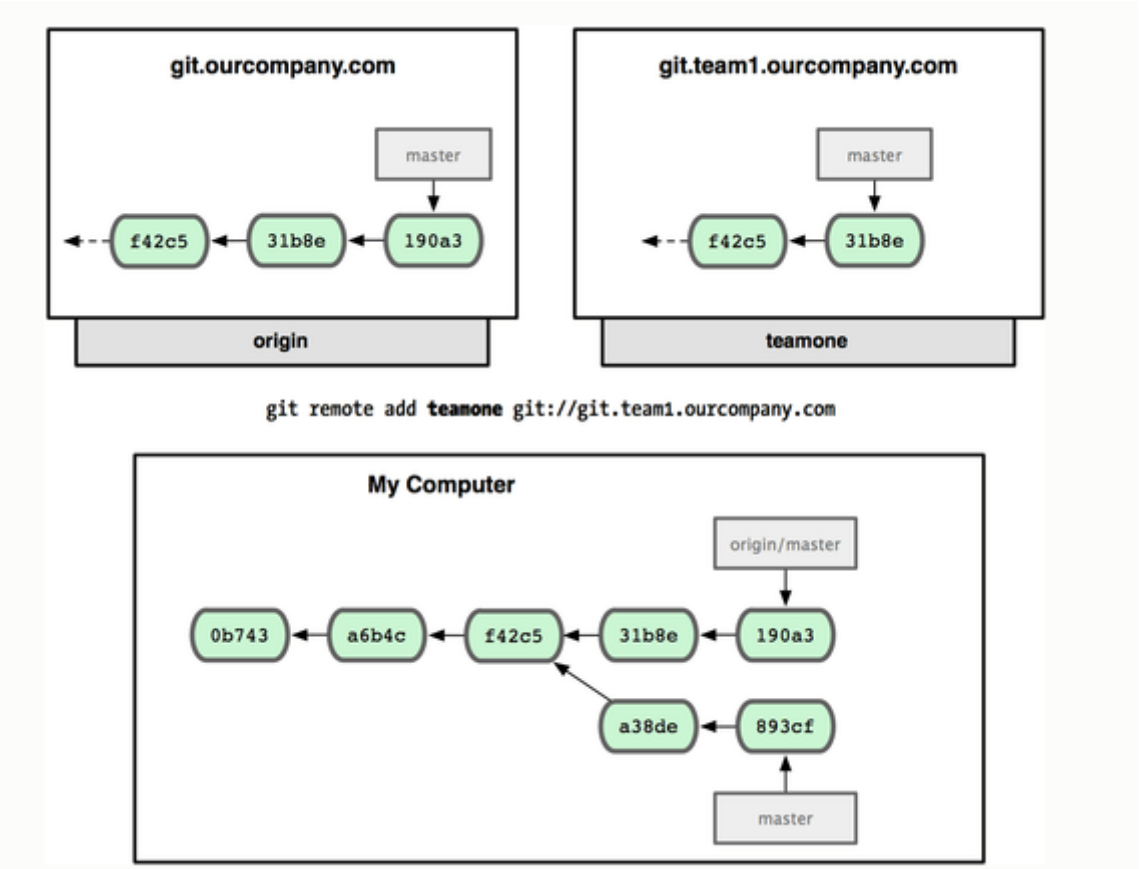


运行 git fetch origin 来同步远程服务器上的数据到本地，该命令首先找到 orgin 是那个服务器（git.ourcompany.com）,从上面获取你尚未拥有的数据，更新你的数据库，然后把 origin/master 的指针移到它最新的位置上，如图：

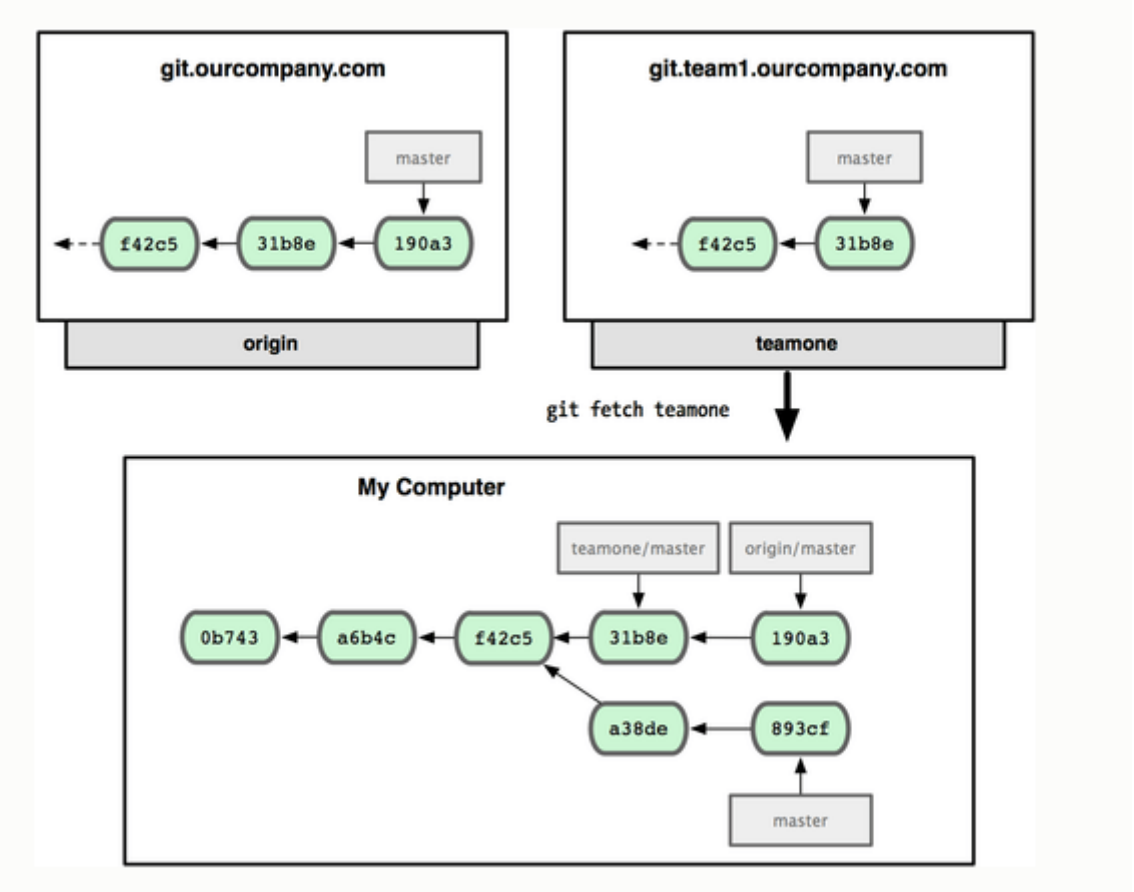


为了演示拥有多个远程分支（在不同的远程服务器上）的项目是如何工作的，假设你还有另一个仅供你的敏捷开发小组使用的内部服务器 git.team1.ourcompany.com。我们使用如下命令把它加为当前项目的远程分支之一。把他命名为 teamone

```
$ git remote add teamone git://git.team1.ourcompany.com
```



现在使用 git fetch teamone 来获取该小组服务器上你还没有的数据。由于当前服务器上的内容是你的 origin 服务器的子集， Git不会下载任何数据，而是简单创建一个名为 teamone/master 的远程分支，指向 teamone服务器上 master 分支所在的提交对象 31b8e



- **推送本地分支**

如果你有个叫 serverfix 的分支需要和其他人一起开发，可以运行 `git push <远程仓库名> <分支名>`

```
$ git push origin serverfix
Counting objects: 20, done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 1.74 KiB, done.
Total 15 (delta 5), reused 0 (delta 0)
To git@github.com:schacon/simplegit.git
* [new branch] serverfix -> serverfix
```

上面这个命令其实走了一点捷径，git自动把 serverfix 分支名扩展为 refs/heads/serverfix:refs/heads/serverfix 意为“取出本地的 serverfix 分支，推送到远程仓库的 serverfix 分支中去” 其实这个命令是 `git push [远程名] [本地分支]:[远程分支]`

> 关于 refs/heads/ 细节

接下来你的协作者再次从服务器获取数据时，他们将得到一个新的远程分支 origin/serverfix 并指向服务器上的 serverfix 所指向的版本：

```
$ git fetch origin
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 5), reused 0 (delta 0)
Unpacking objects: 100% (15/15), done.
From git@github.com:schacon/simplegit
* [new branch] serverfix -> origin/serverfix
```

需要注意的是，在 fetch 操作下载好新的远程分支之后，你仍然无法在本地编辑该远程仓库中的分支，你不会有一个新的serverfix分支，而只有一个无法移动的 origin/serverfix 指针。

如果要把该远程分支的内容合并到当前分支，可以运行 `git merge origin/serverfix`

如果想要一份自己的 serverfix 来开发，可以在远程分支的基础上分化出一个新的分支来：

```
$ git checkout -b serverfix origin/serverfix
Branch serverfix set up to track remote branch serverfix from origin.
Switched to a new branch 'serverfix'
```

这样会切换到 serverfix本地分支，其内容和远程分支 origin/serverfix 一致，你就可以继续开发了。

- **跟踪远程分支**

从远程分支 checkout 出来的本地分支，称为 跟踪分支（tracking branch）。跟踪分支是一种和某个远程分支有直接联系的本地分支。在跟踪分支里运行 git push , git 会自动推断应该向那个服务器的那个分支推送数据。git pull 会获取所有远程索引，并把它们的数据都合并到本地分支中来。

在克隆仓库时，git 会自动创建一个 master 分支来追踪 origin/master ，所以一开始就可以正常的 pull 和push

> 一般创建跟踪分支的命令是 `git checkout -b [分支名] [远程名]/[分支名]` 如果你是 1.6.2 版本以上的git，可以用 --track 简化命令 `$ git checkout --track origin/serverfix`

- **删除远程分支**

如果远程分支不需要了，可以使用这个命令来删除它 `git push [远程名] : [分支名]`

```
$ git push origin :serverfix
To git@github.com:schacon/simplegit.git
- [deleted] serverfix
```

> 方便记忆法： 上面有个命令 `git push [远程名] [本地分支]:[远程分支]` 语法，如果省略 [本地分支]，那就等于是在说“在这里提取空白然后把它变成[远程分支]”。