# Git 远程分支

## Git 向远程仓库推送新分支

### 步骤1：新建本地分支

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (master)

$ git branch -a

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/master

# 新建 develop 分支

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (master)

$ git checkout -b develop

Switched to a new branch 'develop'

# 在 develop 分支中新建文件，并提交

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ vim readme.txt

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git commit -a -m "dev 0"

[develop aef924b] dev 0

1 file changed, 1 insertion(+)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/master

# 在新建 develop 分支上，测试 push 和 push 命令

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ 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> develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (devlop)

$ git push

fatal: The current branch devlop has no upstream branch.

To push the current branch and set the remote as upstream, use

git push --set-upstream origin devlop

# 查看远程仓库 origin 的信息详情

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (devlop)

$ git remote show origin

\* remote origin

Fetch URL: D:/1111/repo/.git

Push URL: D:/1111/repo/.git

HEAD branch: master

Remote branch:

master tracked

Local branch configured for 'git pull':

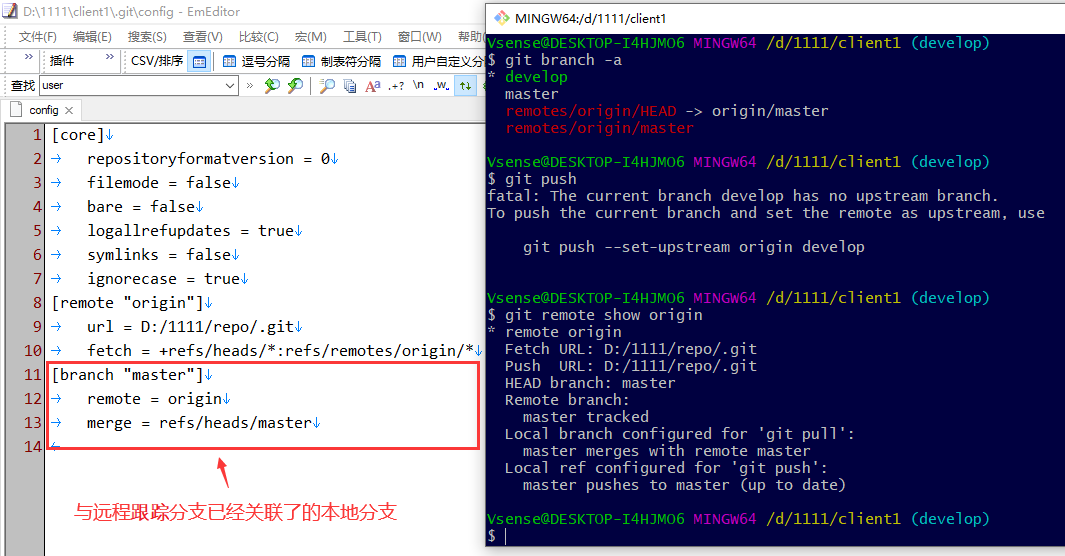
master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

此时，

* 通过 git branch –a 可以发现，本地分支中有新建的 develop 分支，远程跟踪分支中还没有 remotes/origin/develop 的存在；
* 查看本地仓库 /.git/config 文件，或者通过 git remote show origin，可以发现，与远程跟踪分支关联的本地分支只有 master 分支;

或者，打开仓库 .git 中的 config 文件中的分支信息：

### 步骤2：推送本地分支到远程仓库

如果，使用 git push origin develop，倒是确实可以将本地分支 develop 推送到远程仓库，并在本地仓库中创建 "remotes/origin/develop" 远程跟踪分支：

$ git push <remote-name> <branch-name>:<remote-branch-name>

示例：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git push origin develop

Counting objects: 3, done.

Delta compression using up to 4 threads.

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

Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0)

To D:/1111/repo/.git

\* [new branch] develop -> develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git push

fatal: The current branch develop has no upstream branch.

To push the current branch and set the remote as upstream, use

git push --set-upstream origin develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git remote show origin

\* remote origin

Fetch URL: D:/1111/repo/.git

Push URL: D:/1111/repo/.git

HEAD branch: master

Remote branches:

develop tracked

master tracked

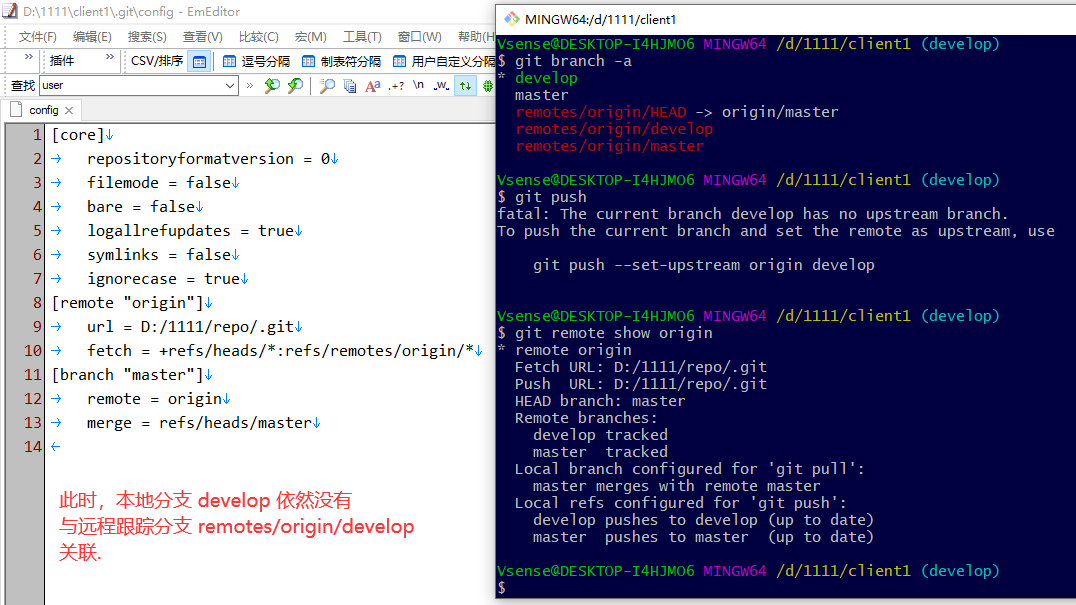
Local branch configured for 'git pull':

master merges with remote master

Local refs configured for 'git push':

develop pushes to develop (up to date)

master pushes to master (up to date)

但是，存在一个问题：本地分支 develop 并没有与远程跟踪分支 remotes/origin/develop 进行关联：

### 步骤3：将本地分支与远程跟踪分支关联

通过 git push --set-upstream origin develop 可以直接将本地分支推送到远程仓库，创建远程跟踪分支 remotes/origin/develop，并且将本地分支 develop 与对应的远程跟踪分支进行关联起来。

* git push –u origin develop 是 git push –set-upstream origin develop 的等价简写形式。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git push --set-upstream origin develop

Everything up-to-date

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git remote show origin

\* remote origin

Fetch URL: D:/1111/repo/.git

Push URL: D:/1111/repo/.git

HEAD branch: master

Remote branches:

develop tracked

master tracked

Local branches configured for 'git pull':

develop merges with remote develop

master merges with remote master

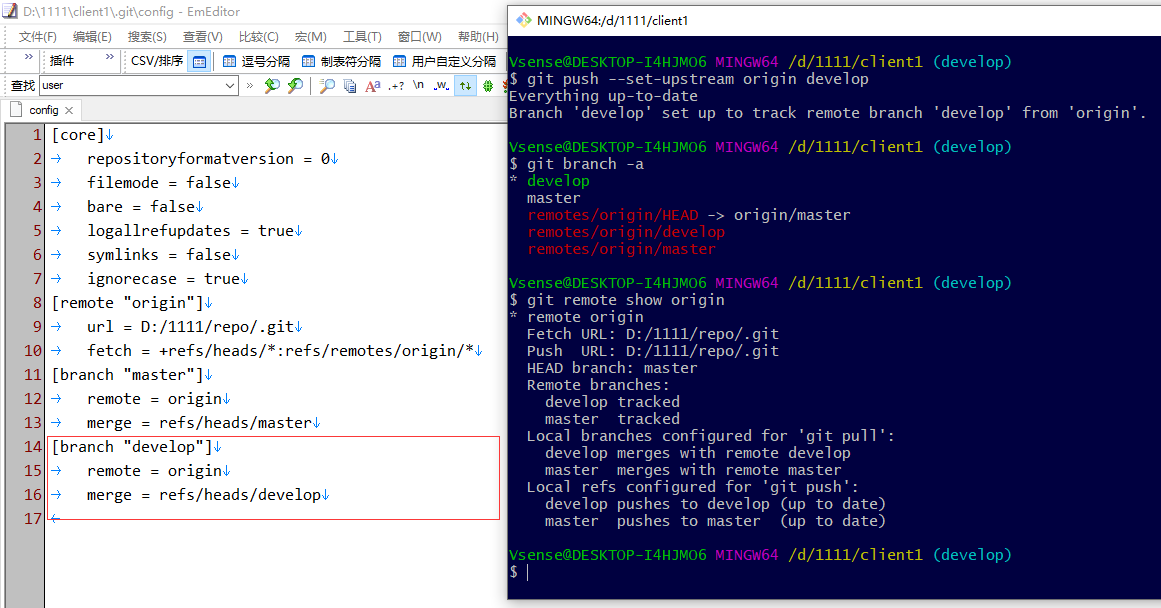
Local refs configured for 'git push':

develop pushes to develop (up to date)

master pushes to master (up to date)

到此，

* 通过 git branch –a 可以发现，本地分支存在 develop 分支，远程跟踪分支中也存在 remotes/origin/develop 分支；
* 查看本地仓库 /.git/config 文件，或者通过 git remote show origin，可以发现，与远程跟踪分支关联的本地分支也增加了 develop 分支;



### 案例A：推送本地分支到远程仓库并关联

目标：将本地新建的分支 develop 推送至远程仓库 origin/develop 并关联;

实现：

**方法 1：同步骤 2、3 所描述，先切换到 develop 分支，然后推送 develop**

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git checkout develop

Switched to branch 'develop'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (develop)

$ git push -u origin develop

Counting objects: 3, done.

Delta compression using up to 4 threads.

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

Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.

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

To D:/Zzzz/git/repo1/.git

\* [new branch] develop -> develop

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

**方法 2：在当前 master 分支，直接推送 develop**

可以不需要切换到指定分支 develop，在当前 master 分支上，也可以推送 develop 分支至远程仓库，其语法格式如下：

$ git push -u <remote-name> <branch-name>:<remote-branch-name>

注意，冒号前后不可以存在空格符！

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git branch -a

develop

\* master

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git branch -vv

develop 39458b2 增加 file2.txt

\* master 557a2dd [origin/master] 增加 file3.txt

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

# 错误，由于冒号前后存在空格

$ git push -u origin develop : develop

error: dst ref refs/heads/develop receives from more than one src.

error: failed to push some refs to 'D:/Zzzz/git/repo1/.git'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

# 错误，由于冒号前面存在空格

$ git push -u origin develop :develop

error: dst ref refs/heads/develop receives from more than one src.

error: failed to push some refs to 'D:/Zzzz/git/repo1/.git'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

# 成功，冒号前后均没有空格

$ git push -u origin develop:develop

Counting objects: 3, done.

Delta compression using up to 4 threads.

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

Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.

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

To D:/Zzzz/git/repo1/.git

\* [new branch] develop -> develop

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git branch -a

develop

\* master

remotes/origin/develop

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git branch -vv

develop 39458b2 [origin/develop] 增加 file2.txt

\* master 557a2dd [origin/master] 增加 file3.txt

## Git 克隆包含多个远程分支的远程仓库

### 步骤1：从克隆开始

通过 git clone 一个包含了多个分支的远程仓库时，默认情况下，Git 在生成的本地仓库中，只会创建与远程仓库的 HEAD 指针（即 remotes/origin/HEAD）所指向的远程跟踪分支对应的同名本地分支，并将两者关联。

例如：

通常，一个远程仓库的 HEAD 指针指向远程仓库的 master 分支，那么 git clone 完成后，本地仓库会创建本地 master 分支，并且与远程跟踪分支 remotes/origin/master 关联。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111

$ git clone D:/1111/repo/.git client2

Cloning into 'client2'...

done.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111

$ cd client2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -a

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git remote show origin

\* remote origin

Fetch URL: D:/1111/repo/.git

Push URL: D:/1111/repo/.git

HEAD branch: master

Remote branches:

develop tracked

master tracked

Local branch configured for 'git pull':

master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

在上面的 git branch –a 命令输出中，不难发现，存在远程跟踪分支 remotes/origin/develop；但是，git clone 时并没有自动创建本地分支 develop 与之关联。

### 步骤2：创建本地分支并与远程跟踪分支关联

Git 提供了两种命令方式：

* $ git checkout -b develop remotes/origin/develop

或者，

$ git checkout -b develop origin/develop

* $ git checkout --track remotes/origin/develop

或者，

$ git checkout --track origin/develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git checkout -b develop remotes/origin/develop

Switched to a new branch 'develop'

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (develop)

$ git push

Everything up-to-date

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (develop)

$ git pull

Already up to date.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (develop)

$ git remote show origin

\* remote origin

Fetch URL: D:/1111/repo/.git

Push URL: D:/1111/repo/.git

HEAD branch: master

Remote branches:

develop tracked

master tracked

Local branches configured for 'git pull':

develop merges with remote develop

master merges with remote master

Local refs configured for 'git push':

develop pushes to develop (up to date)

master pushes to master (up to date)

## Git 获取远程仓库中分支的更新

### 注意：在本地仓库中获取当前分支之外的其它分支的更新

# 测试背景：

# 1. 本地分支 issue 与 远程分支 origin/issue 已经建立了关联;

# 2. 远程分支 origin/issue 有了新的提交；

# 3. 当前本地仓库位于 master 分支上（非 issue 分支）

# 如果直接在 master 上执行 git pull，

# 那么本地 issue 分支并没有更新至 origin/issue 分支

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git pull

remote: Counting objects: 3, done.

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

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

Unpacking objects: 100% (3/3), done.

From file:///d/Zzzz/git/repo1/

801722f..1bda9b1 issue -> origin/issue

Already up to date.

# 此时切换到 issue 分支时，注意 git 的提示信息：

# (use "git pull" to update your local branch)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (master)

$ git checkout issue

Switched to branch 'issue'

Your branch is behind 'origin/issue' by 1 commit, and can be fast-forwarded.

(use "git pull" to update your local branch)

# 需要在 issue 分支再次执行 git pull 完成 fast-forward 合并

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/Zzzz/git/client1 (issue)

$ git pull

Updating 801722f..1bda9b1

Fast-forward

issue2.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 issue2.txt

## Git 获取远程仓库中的新分支

目前，在前面的测试案例中，有两个本地仓库 client1 和 client2，他们分别都完成了与远程仓库的同步，都同样包含了连个分支 master 和 develop 分支。

为了模拟这一节内容的情形案例，我们先回到本地仓库 client1 中再新增一个分支 hotfix 并将其推送到远程仓库中。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (develop)

$ git checkout -b hotfix

Switched to a new branch 'hotfix'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (hotfix)

$ vim readme.md

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (hotfix)

$ git commit -a -m "C11"

[hotfix 638b4bc] C11

1 file changed, 1 insertion(+)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (hotfix)

$ git push

fatal: The current branch hotfix has no upstream branch.

To push the current branch and set the remote as upstream, use

git push --set-upstream origin hotfix

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (hotfix)

$ git push --set-upstream origin hotfix

Counting objects: 3, done.

Writing objects: 100% (3/3), 258 bytes | 129.00 KiB/s, done.

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

To D:/1111/repo/.git

\* [new branch] hotfix -> hotfix

Branch 'hotfix' set up to track remote branch 'hotfix' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client1 (hotfix)

$ git branch -a

develop

\* hotfix

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/hotfix

remotes/origin/master

### 步骤1：先拉取或抓取远程仓库同步到本地仓库

回到本地仓库 client2 中，在 master 分支下执行 git pull 或者 git fetch：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull

remote: Counting objects: 3, done.

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

Unpacking objects: 100% (3/3), done.

From D:/1111/repo/

\* [new branch] hotfix -> origin/hotfix

Already up to date.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -a

develop

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/hotfix

remotes/origin/master

### 步骤2：创建本地分支并与远程跟踪分支关联

Git 提供了两种命令方式：

* $ git checkout -b develop remotes/origin/develop

或者，

$ git checkout -b develop origin/develop

* $ git checkout --track remotes/origin/develop

或者，

$ git checkout --track origin/develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git checkout --track origin/hotfix

Switched to a new branch 'hotfix'

Branch 'hotfix' set up to track remote branch 'hotfix' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (hotfix)

$ git branch -a

develop

\* hotfix

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/hotfix

remotes/origin/master

## Git 将本地已存在的本地分支与远程跟踪分支关联

### 查看所有本地分支的关联状态

如果想要查看所有本地分支哪些已经关联远程跟踪分支，哪些没有关联，可以使用 git branch 的 -vv 选项。

$ git branch -vv

  iss53 7e424c3 [origin/iss53: ahead 2] forgot the brackets

  master 1ae2a45 [origin/master] deploying index fix

\* serverfix f8674d9 [teamone/server-fix-good: ahead 3, behind 1] this

should do it

  testing 5ea463a trying something new

这里可以看到 iss53 分支正在跟踪 origin/iss53 并且 “ahead” 是 2，意味着本地有两个提交还没有推送到服务器上。 也能看到 master 分支正在跟踪 origin/master 分支并且是最新的。 接下来可以看到serverfix 分支正在跟踪 teamone 服务器上的 server-fix-good 分支并且领先 3 落后 1， 意味着服务器上有一次提交还没有合并入同时本地有三次提交还没有推送。 最后看到 testing 分支并没有跟踪任何远程分支。

需要重点注意的一点是这些数字的值来自于你从每个服务器上最后一次抓取的数据。 这个命令并没有连接服务器，它只会告诉你关于本地缓存的服务器数据。 如果想要统计最新的领先与落后数字，需要在运行此命令前抓取所有的远程仓库。 可以像这样做：

$ git fetch --all; git branch -vv

### 将远程跟踪分支与已存在分支进行关联

#### 添加远程仓库

有时候，一个本地仓库可能存在了多个远程仓库；比如，你的本地仓库可以既推送到 github 上，也可以又推送到 Gitee 上；或者，既推送到 github 上，又推送到可移动 U 盘中。

为了模拟这一节内容的情形案例，将上面的案例中的“远程仓库”（路径：/d/1111/repo/.git）复制一份，于是就有了新的“远程仓库”，其路径是：/d/111/repo2/.git

回到本地仓库 client2 中，为其再添加一个远程仓库：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git remote -v

origin D:/1111/repo/.git (fetch)

origin D:/1111/repo/.git (push)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git remote add team1 /d/1111/repo2/.git

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git remote -v

origin D:/1111/repo/.git (fetch)

origin D:/1111/repo/.git (push)

team1 D:/1111/repo2/.git (fetch)

team1 D:/1111/repo2/.git (push)

可以看到，远程仓库 repo2 的引用 team1 已成功添加到当前本地仓库中；不过，该新增加的远程仓库 team1 的内容并没有下载到本地仓库中，本地仓库中内容也没有上传到该远程仓库；此时，你可以理解为目前还只是在本地仓库中建立了远程仓库 repo2/.git 的一个引用 team1。

使用 git branch –a 可以发现，远程仓库 team1 的远程跟踪分支还没有创建：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -a

develop

hotfix

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/hotfix

remotes/origin/master

接下来，使用 git pull team1 先从远程仓库 team1 上下载数据：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull team1

From D:/1111/repo2/

\* [new branch] develop -> team1/develop

\* [new branch] hotfix -> team1/hotfix

You asked to pull from the remote 'team1', but did not specify

a branch. Because this is not the default configured remote

for your current branch, you must specify a branch on the command line.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -a

develop

hotfix

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/hotfix

remotes/origin/master

remotes/team1/develop

remotes/team1/hotfix

remotes/team1/master

在执行了 git pull team1 之后，通过 git branch –a 可以看到，远程仓库 team1 在本地仓库的远程跟踪分支也已经生成（remotes/team1/master 等等）。

但是，在上面的 git pull team1 执行的输出提示了：

You asked to pull from the remote 'team1', but did not specify

a branch. Because this is not the default configured remote

for your current branch, you must specify a branch on the command line.

这句话的意思是：

当前拉取（pull）的远程仓库 team1 中的分支，与当前本地仓库的当前分支（master）并没有建立关联，原因是远程仓库 tem1 并不是当前本地分支 master 默认配置关联的远程仓库（当前本地分支 master 默认配置关联的远程跟踪分支是 remotes/origin/master）。

因此，上面 git pull team1 的执行效果其实只是执行了 git fetch team1，并没有执行 git merge team1/master 操作，即并没有将远程跟踪分支 remotes/team1/master 与当前本地分支 master 进行合并操作。

解决这个问题的方式有多种，

简单的方式是按照执行 git pull team1 时反馈的提示信息：

you must specify a branch on the command line.

使用 git pull team1 master : master 代替 git pull team1 明确指定与当前本地分支 master 合并。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull team1 master : master

From D:/1111/repo2/

\* branch master -> FETCH\_HEAD

\* branch HEAD -> FETCH\_HEAD

\* branch master -> FETCH\_HEAD

Already up to date.

另一个方式，就是“[关联已存在本地分支](#_关联已存在的本地分支)”，将 remotes/team1/master 与本地分支 master 进行关联，之后再执行 git pull team1 时，就不需要明确指定分支名称。

#### 关联已存在的本地分支

设置已有的本地分支跟踪一个刚刚拉取下来的远程分支，或者想要修改正在跟踪的上游分支， 你可以在任意时间使用 -u 或 --set-upstream-to 选项运行 git branch 来显式地设置。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull team1

You asked to pull from the remote 'team1', but did not specify

a branch. Because this is not the default configured remote

for your current branch, you must specify a branch on the command line.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -u team1/master

Branch 'master' set up to track remote branch 'master' from 'team1'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull team1

Already up to date.

* 上游分支快捷方式

当设置好跟踪分支后，可以通过简写 @{upstream} 或 @{u} 来引用它的上游分支。 所以在master 分支时并且它正在跟踪 origin/master 时，如果愿意的话可以使用 git merge @{u} 来取代 git merge origin/master。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git fetch team1

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git merge @{u}

Already up to date.

#### 一个本地分支只能与一个远程跟踪分支关联

注意，在前面的案例中，原本本地分支 master 是与 remotes/origin/master 关联的；

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -vv

develop 4abd1c2 [origin/develop] C2

hotfix 638b4bc [origin/hotfix] C11

\* master 58a9fac [origin/master] C1

当执行了 git branch –u team1/master 之后，本地分支 master 变成与 remote/team1/master 相关联：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -u team1/master

Branch 'master' set up to track remote branch 'master' from 'team1'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -vv

develop 4abd1c2 [origin/develop] C2

hotfix 638b4bc [origin/hotfix] C11

\* master 58a9fac [team1/master] C1

这样一来，如果想从远程跟踪分支remotes/origin/master 拉取到本地 master 分支，或者将本地master 分支推送到远程跟踪分支remotes/origin/master，也只好在命令中明确指定分支名称。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git pull origin

You asked to pull from the remote 'origin', but did not specify

a branch. Because this is not the default configured remote

for your current branch, you must specify a branch on the command line.

或者，重新将本地 master 分支与远程跟踪分支 remotes/origin/master 关联（意味着本地 master 分支会与原本的 remotes/team1/master 断开关联）：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -u origin/master

Branch 'master' set up to track remote branch 'master' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/1111/client2 (master)

$ git branch -vv

develop 4abd1c2 [origin/develop] C2

hotfix 638b4bc [origin/hotfix] C11

\* master 58a9fac [origin/master] C1

总之，一个本地分支只能与一个远程跟踪分支相关联，一个本地分支不能同时与多个远程跟踪分支相关联。

### 取消对远程跟踪分支的关联

git branch --unset-upstream <remote>/<r-branch>

测试失败，原因未知：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [origin/develop] dev c1

master 49f5d3c [origin/master] m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch --unset-upstream origin/develop

fatal: Branch 'origin/develop' has no upstream information

## Git 删除远程仓库中的远程分支

### 删除远程分支

如果想要从服务器上删除 serverfix 分支，运行下面的命令：

* 方式1：$ git push origin --delete serverfix
* 方式2：$ git branch –d –r origin/serverfix

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git push team1 --delete develop

To D:/2222/repo2/.git

- [deleted] develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [team1/develop: gone] dev c1

master 49f5d3c [origin/master] m line 2

Git 删除远程仓库中的远程分支，会保留对应的本地分支，即使本地分支与被删除的远程分支进行了关联。注意上面 git branch –vv 中的信息：

$ git branch -vv

\* develop 13677bd [team1/develop: gone] dev c1

表示本地分支 develop 所关联的远程跟踪分支 team1/develop 处于 gone 状态，因为我们将它删除了。

因此，通常当删除了某个远程分支之后，如果当前本地分支与其有关联，建议将其取消关联：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [team1/develop: gone] dev c1

master 49f5d3c [origin/master] m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

# 取消当前分支 develop 的远程关联

$ git branch --unset-upstream

# 如果当前分支不是 develop, 也可以指定取消关联的分支

$ git branch –unset-upstream develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd dev c1 # 注意，这里没有了关联

master 49f5d3c [origin/master] m line 2

或者，修改为关联到其它远程分支：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -u origin/develop

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [origin/develop] dev c1

master 49f5d3c [origin/master] m line 2

### 恢复被删除的远程分支

#### 如果本地分支还存在

当某个远程分支被 git push <remote> --delete <branch> 删除了，如果本地仓库中还存在相应的本地分支 <branch>，那么：

如果此前已经进行了关联，可以直接将本地分支推送到该远程仓库中：

$ git push <remote> <local-branch> : <remote-branch>

如果希望将本地分支与远程分支关联，那么，可以使用：

$ git push –u <remote> <local-branch> : <remote-branch>

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git push team1 --delete develop

To D:/2222/repo2/.git

- [deleted] develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git push team1 develop

Counting objects: 6, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (6/6), 490 bytes | 490.00 KiB/s, done.

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

To D:/2222/repo2/.git

\* [new branch] develop -> develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

#### 如果本地分支不存在

当本地分支也不存在时，先通过 git reflog --date=iso 找到被删除的本地分支的最后 commit 的 sha1 值，然后 git checkout –b <分支名称> sha1 新建分支；最后再将恢复的本地分支推送到远程仓库。

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -r

origin/HEAD -> origin/master

origin/develop

origin/master

team1/develop

team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git push team1 --delete develop

To D:/2222/repo2/.git

- [deleted] develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -r

origin/HEAD -> origin/master

origin/develop

origin/master

team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git checkout master

Switched to branch 'master'

Your branch is up to date with 'origin/master'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -d develop

warning: deleting branch 'develop' that has been merged to

'refs/remotes/origin/develop', but not yet merged to HEAD.

Deleted branch develop (was 13677bd).

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -a

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git reflog

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{0}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{1}: checkout: moving from master to develop

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{2}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{3}: checkout: moving from master to develop

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{4}: commit: m line 2

67c7b75 HEAD@{5}: merge @{u}: Fast-forward

ae9cef5 HEAD@{6}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{7}: pull: Fast-forward

3b7d900 HEAD@{8}: checkout: moving from master to develop

ae9cef5 HEAD@{9}: pull: Fast-forward

267331d HEAD@{10}: clone: from D:/2222/repo/.git

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git reflog --date=iso

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{2022-01-08 18:17:13 +0800}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{2022-01-08 18:05:34 +0800}: checkout: moving from master to develop

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{2022-01-08 17:43:43 +0800}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{2022-01-08 17:25:03 +0800}: checkout: moving from master to develop

49f5d3c (HEAD -> master, team1/master, origin/master, origin/HEAD) HEAD@{2022-01-08 16:02:30 +0800}: commit: m line 2

67c7b75 HEAD@{2022-01-08 11:53:21 +0800}: merge @{u}: Fast-forward

ae9cef5 HEAD@{2022-01-08 11:52:31 +0800}: checkout: moving from develop to master

13677bd (origin/develop) HEAD@{2022-01-08 11:51:29 +0800}: pull: Fast-forward

3b7d900 HEAD@{2022-01-08 01:50:14 +0800}: checkout: moving from master to develop

ae9cef5 HEAD@{2022-01-08 01:48:44 +0800}: pull: Fast-forward

267331d HEAD@{2022-01-08 01:45:38 +0800}: clone: from D:/2222/repo/.git

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git checkout -b develop 13677bd

Switched to a new branch 'develop'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git push team1 develop

Counting objects: 6, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (6/6), 490 bytes | 490.00 KiB/s, done.

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

To D:/2222/repo2/.git

\* [new branch] develop -> develop

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

### 删除远程仓库（的引用）

要注意在本地仓库执行“删除远程仓库”的操作，只是将位于本地仓库中的对远程仓库的引用进行了删除；真实的远程仓库依然保存在原地。

使用 git remote remove <remote> 可以从本地仓库中移除对应的远程仓库的引用：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git remote -v

origin D:/2222/repo/.git (fetch)

origin D:/2222/repo/.git (push)

team1 D:/2222/repo2/.git (fetch)

team1 D:/2222/repo2/.git (push)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/HEAD -> origin/master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [origin/develop] dev c1

master 49f5d3c [origin/master] m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git remote remove origin

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd dev c1

master 49f5d3c m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -a

develop

\* master

remotes/team1/develop

remotes/team1/master

### 恢复被删除的远程仓库（的引用）

直接将远程仓库添加到当前本地仓库中，并根据需要添加关联即可：

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git remote add origin /d/2222/repo/.git

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git remote -v

origin D:/2222/repo/.git (fetch)

origin D:/2222/repo/.git (push)

team1 D:/2222/repo2/.git (fetch)

team1 D:/2222/repo2/.git (push)

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/team1/develop

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git fetch origin

From D:/2222/repo/

\* [new branch] develop -> origin/develop

\* [new branch] master -> origin/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -a

\* develop

master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd dev c1

master 49f5d3c m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -u origin/develop

Branch 'develop' set up to track remote branch 'develop' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git branch -vv

\* develop 13677bd [origin/develop] dev c1

master 49f5d3c m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (develop)

$ git checkout master

Switched to branch 'master'

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -u origin/master

Branch 'master' set up to track remote branch 'master' from 'origin'.

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -vv

develop 13677bd [origin/develop] dev c1

\* master 49f5d3c [origin/master] m line 2

Vsense@DESKTOP-I4HJMO6 MINGW64 /d/2222/client2 (master)

$ git branch -a

develop

\* master

remotes/origin/develop

remotes/origin/master

remotes/team1/develop

remotes/team1/master

## Git 同步远程仓库中已删除的远程分支

// 参考 https://www.cnblogs.com/saysmy/p/9166331.html

$ git remote show origin

$ git branch –a

$ git branch -vv

$ git remote prune origin

$ git branch –a

$ git branch -vv

$ git branch –D <branch-name>

## Git 强制推送并覆盖远程仓库中的远程分支

# 将本地分支推送到远程仓库的远程分支：

# git push <remote> <local-branch> : <remote-branch>

# 比如：

# 将本地分支 develop 推送到远程仓库 origin 的 develop 分支

$ git push origin develop

# 将本地分支 develop 推送到远程仓库 origin 的 develop 分支，与上面等价

$ git push origin develop : develop

# 将本地分支 develop 推送到远程仓库 origin 的 dev 分支

$ git push origin develop : dev

# 将本地分支推送到远程仓库的远程分支，并进行关联：

# git push -u <remote> <local-branch> : <remote-branch>

# 强制推送分支到远程仓库（覆盖远程分支）：

$ git push -f origin master

# 强制推送分支到远程仓库（覆盖远程分支），并进行关联：

$ git push -f -u origin master

# Git 远程标签

## [git 如何同步本地tag与远程tag](https://www.cnblogs.com/xiaouisme/p/10857149.html)

问题场景：  
同事A在本地创建tagA并push同步到了远程->同事B在本地拉取了远程tagA(git fetch)->同事A工作需要将远程标签tagA删除->同事B用git fetch同步远端信息，git tag后发现本地仍然记录有tagA

分析：对于远程repository中已经删除了的tag，即使使用git fetch --prune，甚至"git fetch --tags"确保下载所有tags，也不会让其在本地也将其删除的。而且，似乎git目前也没有提供一个直接的命令和参数选项可以删除本地的在远程已经不存在的tag（我目前是没找到有关这类tag问题的git命令~~，有知道的同学可以告知我下，互相进步）。  
解决方法：

git tag -l | xargs git tag -d #删除所有本地分支  
**git fetch origin --prune** #从远程拉取所有信息  
git branch -a --contains Tag\_V1.0.0 # 看看哪个分支包含这个tag/commit

#查询远程tags的命令如下：  
git ls-remote --tags origin

tag常用git命令：  
git tag #列出所有tag  
git tag -l v1.\* #列出符合条件的tag（筛选作用）  
git tag [tag名] #创建轻量tag（无-m标注信息）  
git tag -a [tag名] #创建含注解的tag

**git push origin --tags** #推送所有本地tag到远程  
git push origin [本地tag名] #推送指定本地tag到远程

git tag -d [本地tag名] #删除本地指定tag  
git push origin :refs/tags/[远程tag名] #删除远程指定tag

git fetch origin [远程tag名] #拉取远程指定tag  
git show [tag名] #显示指定tag详细信息