1. 用户一上传代码命令：

 git branch -vv    查看本地分支合远程分支的映射关系

1. 拉取https://github.com/city6789/springboot20191026.git（默认就是master分支），会生成一个本地分支（名字叫master），与远端的master分支建立关联：

**git clone https://github.com/city6789/springboot20191026.git**

( git branch -vv 可以验证映射关系 )

1. 以 master分支为基础新建一个本地分支 dev\_heshiyuan并切换到该分支:**git checkout -b dev\_heshiyuan**

3.向该空工程文件夹下 加如代码（比如 src/xxx,pom.xml）并推到远程的 dev\_heshiyuan:

3.1. **git push origin dev\_heshiyuan**  ( 这是 远端 已经有 dev\_heshiyuan分支可，github上可以验证，但是此时远端的dev\_heshiyuan分支并未和本地的dev\_heshiyuan分支映射 )

3.2. 将本地的分支dev\_heshiyuan和远端的分支dev\_heshiyuan映射：

**git branch -u origin/dev\_heshiyuan dev\_heshiyuan** ( ps: 映射的目的是执行 git push ,git pull 命令时不用指定目的和源分支，目的分支和源分支就是当前所在的映射分支

4，一个修改了代码（修改的是本地的dev\_heshiyuan分支的代码）并推送至远端dev\_heshiyuan分支的过程：

git add .

git commit -m "修改了啥啥啥..."

git push

如果想将本地分支xxx推送至远端 yyy分子（xxx和yyy并未建立映射），请百度 git push 需要带什么参数

[总结：在本地新建库，并同步到远端的步骤：

1，克隆（从远程库下载代码到本地） **git clone https://github.com/city6789/springboot20191026.git**

2，新建分支，以远程库里的代码为母本新建分支 **git checkout -b dev\_heshiyuan**

3，把新分支推到远端新建分支 **git push origin dev\_heshiyuan**

4，本地分支和远端分支 建立映射关系 **git branch -u origin/dev\_heshiyuan dev\_heshiyuan**

]

从git库下载代码

在G:\git\_workspace\eclipse\_workspace目录右键git bash here

git clone <https://github.com/city6789/springboot20191026.git>

（从远程url下载代码到本地）

（从远程url下载代码到本地，带参数指定clone到本地的分支（dev\_hsy），默认是master）

git clone -b dev\_hsy <https://github.com/city6789/springboot20191026.git>

      ---拉取指定分支

Cd springboot20191026 [cd 到该项目的路径下]

git branch -vv

场景1：本地不修改，远程修改了，更新远程到本地

--应先add commit到本地git库；然后再git pull

git pull一下就行了

场景2：本地修改，远程修改了，更新远程到本地（覆盖更新到本地，本地丢失）

--方案1 【add 且 commit后】

$ git fetch --all

$ git reset --hard origin/dev\_hsy

$ git pull

--方案2[未add时]

git checkout .

--方案3[add后，未commit 使用]

git status 先看一下add 中的文件

git reset HEAD 如果后面什么都不跟的话 就是上一次add 里面的全部撤销了  git reset HEAD XXX/XXX/XXX.java 就是对某个文件进行撤销了

git checkout .

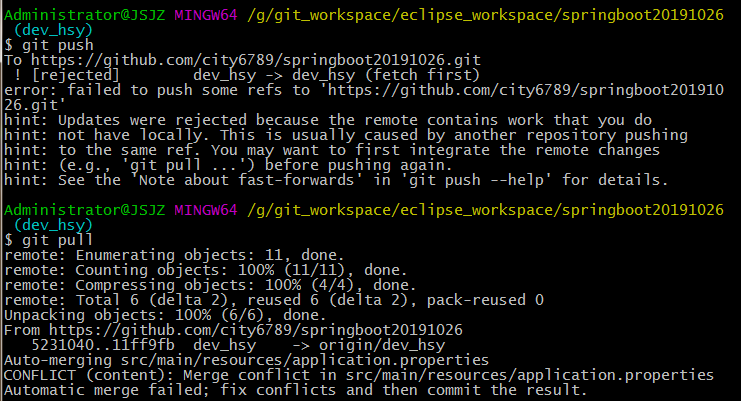
场景3：本地也修改，远程也修改，提交代码到远程。（合并更新到本地，本地不丢失）

Git add .

Git commit –m “”

Git push

--------冲突



处理冲突

先git pull 把冲突的部分在本地改好后，再add commit提交

Git add .

Git commit

Git push

.切换本地分支：  
    git checkout dev\_hsg

(将本地分支dev\_hsg 切换为本地当前命令行所在分支)



<https://www.cnblogs.com/wbl001/p/11495110.html>【记录】Git pull(拉取)，push(上传)命令整理（详细）

Production分支（主线分支用于发版，不会直接改）

Master分支，这个分支只能从其他分支合并，不能在这个分支直接修改

Develop分支（开发分支）

这个分支是我们是我们的主开发分支，包含所有要发布到下一个Release的代码，这个主要合并与其他分支，比如Feature分支

Feature分支（新功能分支）

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回Develop分支进入下一个Release

Release分支（偏向测试）

当你需要一个发布一个新Release的时候，我们基于Develop分支创建一个Release分支，完成Release后，我们合并到Master和Develop分支

Hotfix分支（紧急bug发布）

当我们在Production发现新的Bug时候，我们需要创建一个Hotfix, 完成Hotfix后，我们合并回Master和Develop分支，所以Hotfix的改动会进入下一个Release



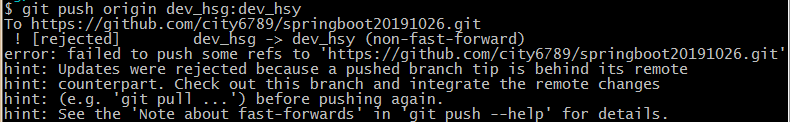
场景4：本地不修改，合并代码到远程（）切换分支

场景5：本地修改，合并代码到远程（）切换分支

本地dev\_hsg 提交到远程 dev\_hsy

git push <远程主机名> <本地分支名>:<远程分支名>

git push origin dev\_hsg:dev\_hsy



处理错误和冲突；先把远端dev\_hsy的pull到本地dev\_hsg，

在本地修改后，add commit push

然后再执行 git push origin dev\_hsg:dev\_hsy

附：操作命令集合：

git merge origin dev\_hsy

git pull origin dev\_hsy

git pull -r

git add .

git commit -m "处理冲突"

git push

git push origin dev\_hsg:dev\_hsy

# 场景6：Git Push的常规操作与Pull冲突解决

<https://blog.csdn.net/u014745069/article/details/81021809>

# GIT完成Eclipse到GitHub一条龙

<https://blog.csdn.net/u014745069/article/details/79838751>

git branch -vv

git status

$ git fetch --all

$ git reset --hard origin/release

$ git pull