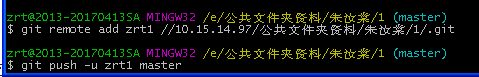
**情景1：**

**git push时出现错误refusing to update checked out branch: refs/heads/master**

客户端已经把修改commit了，然后输入一下两条指令：



出现一下报错：

Counting objects: 3, done.

Writing objects: 100% (3/3), 262 bytes | 0 bytes/s, done.

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

remote: error: refusing to update checked out branch: refs/heads/master

remote: error: By default, updating the current branch in a non-bare repository

remote: error: is denied, because it will make the index and work tree inconsistent

remote: error: with what you pushed, and will require 'git reset --hard' to match

remote: error: the work tree to HEAD.

remote: error:

remote: error: You can set 'receive.denyCurrentBranch' configuration variable t

remote: error: 'ignore' or 'warn' in the remote repository to allow pushing int

remote: error: its current branch; however, this is not recommended unless you

remote: error: arranged to update its work tree to match what you pushed in som

remote: error: other way.

remote: error:

remote: error: To squelch this message and still keep the default behaviour, se

remote: error: 'receive.denyCurrentBranch' configuration variable to 'refuse'.

To //10.15.14.97/公共文件夹资料/朱汝棠/1/.git

! [remote rejected] master -> master (branch is currently checked out)

error: failed to push some refs to '//10.15.14.97/公共文件夹资料/朱汝棠/1/.git'

## 上面原因是因为远程仓库创建的时候没有使用—bare选项

### 第一种方法:

解决这个冲突需要将远程仓库当前分支切换到其他分支（如谁也不会使用的分支），

### 第二种方法：

这是由于git默认拒绝了push操作，需要进行设置，在客户端修改.git/config添加如下代码：

[receive]

denyCurrentBranch = ignore

或者按照提示内容，设置denyCurrentBranch = ignore

在服务器上运行(其实发现只要在服务器上运行这条就可以了，客户端不用修改)：

git config receive.denyCurrentBranch ignore

然后就能把客户端的信息push到服务器上了：

$ git push zrt1 master

Counting objects: 3, done.

Writing objects: 100% (3/3), 275 bytes | 0 bytes/s, done.

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

To //10.15.14.97/公共文件夹资料/朱汝棠/1/.git

ea43a84..6dfece2 master -> master

**上面设置后你可以push上去但是在服务器上目录里看不到push进来的文件，也就是文件已经修改了，但是在服务器的log中看不到。**

## 错误解析：

在初始化远程仓库时最好使用 git--bare init  ，而不要使用：git init

如果使用了gitinit初始化，则远程仓库的目录下，也包含worktree，当本地仓库向远程仓库push时, 如果远程仓库正在push的分支上（如果当时不在push的分支，就没有问题）, 那么push后的结果不会反应在worktree上,  也即在远程仓库的目录下对应的文件还是之前的内容，必须得使用git reset--hard才能看到push后的内容.

## 下面记录下创建仓库时的区别

git init  和 git init –bare 的区别---------

使用命令"git init --bare"(bare汉语意思是:裸,裸的)初始化的版本库(暂且称为bare repository)只会生成一类文件:用于记录版本库历史记录的.git目录下面的文件;而不会包含实际项目源文件的拷贝;所以该版本库不能称为工作目录(working tree);

如果你进入版本目录,就会发现只有.git目录下的文件,而没有其它文件;就是说,这个版本库里面的文件都是.git目录下面的文件,把原本在.git目录里面的文件放在版本库的根目录下面;

换句话说,不使用--bare选项时,就会生成.git目录以及其下的版本历史记录文件,这些版本历史记录文件就存放在.git目录下;

而使用--bare选项时,不再生成.git目录,而是只生成.git目录下面的版本历史记录文件,这些版本历史记录文件也不再存放在.git目录下面,而是直接存放在版本库的根目录下面

用"git init"初始化的版本库用户也可以在该目录下执行所有git方面的操作。但别的用户在将更新push上来的时候容易出现冲突。

比如有用户在该目录（就称为远端仓库）下执行git操作，且有两个分支(master 和 b1)，当前在master分支下。另一个用户想把自己在本地仓库（就称为本地仓库）的master分支的更新提交到远端仓库的master分支，他就想当然的敲了

git push origin master:master

于是乎出现因为远端仓库的用户正在master的分支上操作，而你又要把更新提交到这个master分支上，当然就出错了。

但如果是往远端仓库中空闲的分支上提交还是可以的，比如

git push origin master:b1

还是可以成功的

解决办法就是使用”git init –bare”方法创建一个所谓的裸仓库，之所以叫裸仓库是因为这个仓库只保存git历史提交的版本信息，而不允许用户在上面进行各种git操作，如果你硬要操作的话，只会得到下面的错误（”This operation must be run in a work tree”）

这个就是最好把远端仓库初始化成bare仓库的原因。

下面我将：

1. 使用“普通库”代指用"git init"命令创建的GIT库；
2. 使用“裸库”代指用"git init --bare"命令创建的GIT库；

当你创建一个普通库时，在工作目录下，除了.git目录之外，你还可以看到库中所包含的所有源文件。你拥有了一个可以进行浏览和修改（add, commit, delete等）的本地库。  
当你创建一个裸库时，在工作目录下，只有一个.git目录，而没有类似于本地库那样的文件结构可供你直接进行浏览和修改。但是你仍旧可以用git show命令来进行浏览，举个例子（参数为某个commit的SHA1值）：

# git show 921dc435a3acd46e48e3d1e54880da62dac18fe0

一般来说，一个裸库往往被创建用于作为大家一起工作的共享库，每一个人都可以往里面push自己的本地修改。一个惯用的命名方式是在库名后加上.git，举个例子：

# mkdir example.git

# cd example.git

# git init --bare .

这样你便拥有了一个叫做example的共享库(在服务器上)。在你自己的本地机器上，你可以用git remote add命令做初始化check-in：

// assume there're some initial files you want to push to the bare repo you just created,

// which are placed under example directory

# cd example

# git init

# git add \*

# git commit -m "My initial commit message"

# git remote add origin git@example.com:example.git

# git push -u origin master

项目团队里面的每个人都可以clone这个库，然后完成本地修改之后，往这个库中push自己的代码。

# git clone git@example.com:example.git

# cd example

**情景2：**

在github上创建项目，然后本地[Git](http://lib.csdn.net/base/git) init

然后没有[git](http://lib.csdn.net/base/git) pull -f --all

然后git add .  | git commit -am "init"

输入命令

git push -u origin master

导致github上的版本里有readme文件和本地版本冲突。

解决方法：

用 git push -u origin master -f 命令，但会把服务器上的版本覆盖,更新到本地版本一样的版本，也就是本地版本覆盖服务器上的版本，这个要慎重使用。

**情景3：**

$ git push -u user1 master

Counting objects: 3, done.

Delta compression using up to 4 threads.

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

Writing objects: 100% (3/3), 244 bytes | 0 bytes/s, done.

Total 3 (delta 1), reused 2 (delta 0)

remote: error: insufficient permission for adding an object to repository database ./objects

remote: fatal: failed to write object

error: unpack failed: unpack-objects abnormal exit

To //10.15.14.97/test/project/backup.git/

! [remote rejected] master -> master (unpacker error)

error: failed to push some refs to '//10.15.14.97/test/project/backup.git/'

原因是权限不够，虽然共享文件夹里显示是所有用户都可访问，但最后把自己的用户名加上，而且要确定自己对共享文件夹可读写。