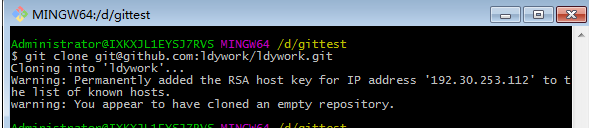
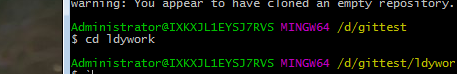
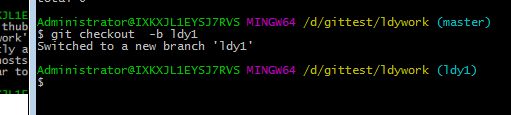
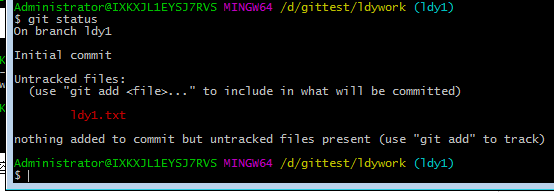
做项目之前先pul代码提交时候再pull代码

1. 克隆项目

git clone （git地址）

1. 
2. 进入项目文件夹
3. 
4. 创建并且换到新分支
5. 
6. 添加文件修改该分支内容。
7. 查看该分支文件的修改git status
8. 
9. 将修改文件存到暂存区git add .
10. 提交到本地仓库git commit –m “修改说明”
11. 切换分支git checkout 已有的目标分支
12. Git push origin 本地分支名称推送到远程
13. 合并目标分支到当前分支git merge 分支名称（分支）
14. 查看本地分支git branch 查看远程分支git branch –a
15. Git还原本地修改git checkout – 要还原的文件名称还原本地修改文件（未add之前）
16. 全部还原git checkout .
17. 还原代码先查看日志git log 查看操作还原命令git reset –hard node

返回上一个版本$ git re**set** --hard HEAD^

1. git reflog用来记录你的每一次命令 防止版本号忘记找不回来

$ git checkout -- readme.txt

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

现在，看看readme.txt的文件内容：

一般情况下，你通常直接在文件管理器中把没用的文件删了，或者用rm命令删了：

$ rm test.txt

$ git rm test.txt

rm 'test.txt'

$ git **commit** -m "remove test.txt"

[master d17efd8] remove test.txt

1 file changed, 1 deletion(-)

**delete** mode 100644 test.txt

另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

$ git checkout -- test.txt

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：

$ git clone git@github.com:michaelliao/gitskills.git

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

Switched to a new branch 'dev'

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

然后，用git branch命令查看当前分支：

现在，我们把dev分支的工作成果合并到master分支上：

$ git merge dev

Updating d17efd8..fec145a

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

git merge命令用于合并指定分支到当前分支。

合并完成后，就可以放心地删除dev分支了：

$ git branch -d dev

Deleted branch dev (was fec145a).

推送分支

推送分支，就是把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上：

$ git push origin master

如果要推送其他分支，比如dev，就改成：

$ git push origin dev

推送失败，因为你的小伙伴的最新提交和你试图推送的提交有冲突，解决办法也很简单，Git已经提示我们，先用git pull把最新的提交从origin/dev抓下来，然后，在本地合并，解决冲突，再推送：

$ git pull

git pull也失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：

$ git branch *--set-upstream dev origin/dev*

git fetch origin QC\_online;  
git merge --no-ff origin/QC\_online  
  
git push origin QC1807:QC\_online

git add . 添加进暂存区

git commit -m "备注" 提交到本地仓库

git push origin branch1:branch1 将本地仓库的branch1 分支推到 远程branch1 分支

git fetch origin branch1 将远端的branch1 分支代码 fetch到本地 origin/下

git merge --no-ff origin/branch1 将fetch的branch1 分支的代码merge到当前分支中

git mergetool 冲突处理

pull = fetch + merge、

git gc 如果有问题 ， 会有提示

git reset -hard 退回到最近提交的版本

git clean -df 清除untrack的文件

添加documentation目录及其子目录下所有\*.txt文件的内容：

$ git add documentation/\*.txt

Shell

注意，在这个例子中，星号\*是从shell引用的; 这允许命令包含来自 Documentation/目录和子目录的文件。

将所有 git-\*.sh 脚本内容添加：

$ git add git-\*.sh

Shell

因为这个例子让shell扩展星号(即明确列出文件)，所以它不考虑子目录中的文件，如：subdir/git-foo.sh这样的文件不会被添加。

**基本用法**

git add <path>

Shell

通常是通过git add <path>的形式把<path>添加到索引库中，<path>可以是文件也可以是目录。

git不仅能判断出<path>中，修改(不包括已删除)的文件，还能判断出新添的文件，并把它们的信息添加到索引库中。





