

Github

Hi all,

Below are some useful git and GitHub (not the same thing, by the way) resources that helped me greatly;

- <https://learngitbranching.js.org/>
- <https://www.codecademy.com/learn/learn-git>

GitHub Actions CI (advanced);

- <https://lo-victoria.com/github-actions-101-creating-your-first-workflow>
- <https://www.codeproject.com/Articles/5265628/Writing-CI-Pipeline-using-GitHub-Actions-to-Build>

Tools and others;

- <https://www.toptal.com/developers/gitignore>
- <https://www.gitkraken.com/>
- <https://education.github.com/pack>

Regards,

Jan-Hendrik

Reply

See this post in context

如何初始化一开始的仓库

不要用readme初始化仓库,否则这个仓库就为master.

初始化仓库的时候一定要选择仓库里什么都没有.

新建文件夹

初始化

添加远程仓库

add commit

git push origin master //这个是本地分支与远程分支不关联的情况下

git push //这个是本地分支与远程分支关联的情况下

[如何关联](#)

然后再用githubdesk打开

1、master是主分支，还可以建一些其他的分支用于开发。

2、git push origin master的意思就是上传本地当前分支代码到master分支。git push是上传本地所有分支代码到远程对应的分支上。

```
git init
git remote add origin website
git add .
git commit -m "init"
git push origin master
```

git clone

#clone

clone只克隆主分支,要想其分支下拉到本地

1. 创建这个本地分支并切换过去
2. 下拉这个分支

状态获取

git status 获取你当前本地仓库有没有东西需要提交
如果是直接克隆的还会告诉你是不是与远端同步
如果是下拉的则不会

git fetch获取本地仓库是不是与远端同步
同步则不会有任何提示 不同步则会有提示

创建分支

#创建分支

很简单，记录一下给有缘人看

- 首先克隆 `git clone url.git`
- 然后查看当前所有分支 `git branch -a`
- 新建本地分支 `git branch my_branch` , 然后切换到本地分支 `git checkout my_branch`
- 将我们的项目代码提交 `git add .` , `git commit -m "new add files"`
- 最后将本地分支推到远程分支 `git push origin my_branch`
- 完结

添加换绑远程

```
//第一次添加远程库  
git remote add origin git@github.com:michaelliao/learngit.git
```

第一次需要先添加远程库,再通过 `git push remote_name`推送

拉取 克隆远程库

拉取远程库的意思就是将本地文件和远程文件同步

即 远程文件修改了,我现在要让本地文件和他一样.

`git pull` 远程库名 这是拉取主分支 pull就是同步

注意克隆分支与主分支是不同的操作

克隆分支

这个操作是创建本地分支并且与远端分支相关联.

`git checkout -b yulong origin/yulong`

[git clone远程仓库的指定分支 51CTO博客](#) [git clone 指定远程分支](#)

克隆主分支

`git clone` 直接https的连接 克隆主分支

推送/下拉分支

切换到该分支直接推

`git pull origin branch_name`

`git push origin branch_name`

在每次推送时一定要确认是否是在需要推送的分支,而不是在错误的分支推送了.

Premise

所有的git 操作都需要在git仓库文件夹下进行, 否则就会出现错误。

[添加远程库 - 廖雪峰的官方网站 \(liaoxuefeng.com\)](#)

[Git心得](#)

`git clone [url]` 克隆远端工程

`git remote -v` 查看目前git连接的哪台远端服务器

git push [origin] [master] 本地推向远端

git checkout branch_name 切换分支

git branch -a 查看所有分支包括远端分支

git branch branch_name remotes/origin/test12 克隆分支 创建分支然后clone

git pull origin branch_name 这是拉取分支 不过本地要先有这个分支并切换过去 origin就是远程的意思

git remote add origin git@github.com:michaelliao/learngit.git 第一次初始化仓库都需要添加远程库

关联远程库后

git pull origin remote_name 拉取

git push 推送

拉取

git pull origin remote_name 拉取远程分支

<https://z.itpub.net/article/detail/F83C6A58E0296138376A73FE4A1ADF61>

使本地分支与远程分支创建联系

git push --set-upstream origin remote_name

git branch -vv 命令可以查看本地分支关联的远程分支的对应关系

操作步骤

Git websites

[Git是什么？可以用来做什么？如何使用？ - 知乎 \(zhihu.com\)](#)

[添加远程库 - 廖雪峰的官方网站 \(liaoxuefeng.com\)](#)

问题

疑难解答

Q: 输入 `git add readme.txt`, 得到错误: `fatal: not a git repository (or any of the parent directories)`。

A: Git命令必须在Git仓库目录内执行 (`git init` 除外), 在仓库目录外执行是没有意义的。

Q: 输入 `git add readme.txt`, 得到错误 `fatal: pathspec 'readme.txt' did not match any files`。

A: 添加某个文件时, 该文件必须在当前目录下存在, 用 `ls` 或者 `dir` 命令查看当前目录的文件, 看看文件是否存在, 或者是否写错了文件名。

小结

第一次使用 git

首先使用git 命令是在git.exe的窗口中,而不是在其他的窗口中。

1. 首先需要初始化 `git init`
2. 然后设置名字

Run

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

to set your account's default identity.

Omit `--global` to set the identity only in this repository.

```
fatal: unable to auto-detect email address (got '冉帅@RamShy.(none)')
```

```
D:\Softwares\Git_Test>git config --global user.email "theshy1011@163.com"
```

```
D:\Softwares\Git_Test>git config --global user.name "RanShy"
```

3. 修改
4. 修改后使用 `git add file` 添加提交
5. `git commit -m "messages"` 正式提交

Commands

- 要随时掌握工作区的状态, 使用 `git status` 命令。
- 如果 `git status` 告诉你有文件被修改过, 用 `git diff` 可以查看修改内容。
- `git log` 日志查询

git中回退

[git 回退](#)

[版本回退的强行推送](#)

GIT

```
git reset --hard dde8c25694f34acf8971f0782b1a676f39bf0a46
```

回退到指定的commit号

GIT

```
$ git reset --hard HEAD^
```

C

```
$ git reset --hard 1094a
```

git reflog 用来记录你的每一次提交号

git log 用来查看提交历史

版本回退的强行推送

GIT

```
git push -f origin <branch name>
```

别人提交或拉取代码可能会因为我们回退了版本而出现错误或者无法拉取，解决命令如下：

GIT

```
git fetch --all
```

关于git于github

git是一种版本管理工具，是可以离线使用的。

github算是远程仓库

查看工作区和版本库里面最新版本的区别

```
git diff HEAD -- readme.txt
```

撤销工作区修改

又到了小结时间。

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令 `git checkout -- file`。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令 `git reset HEAD <file>`，就回到了场景1，第二步按场景1操作。

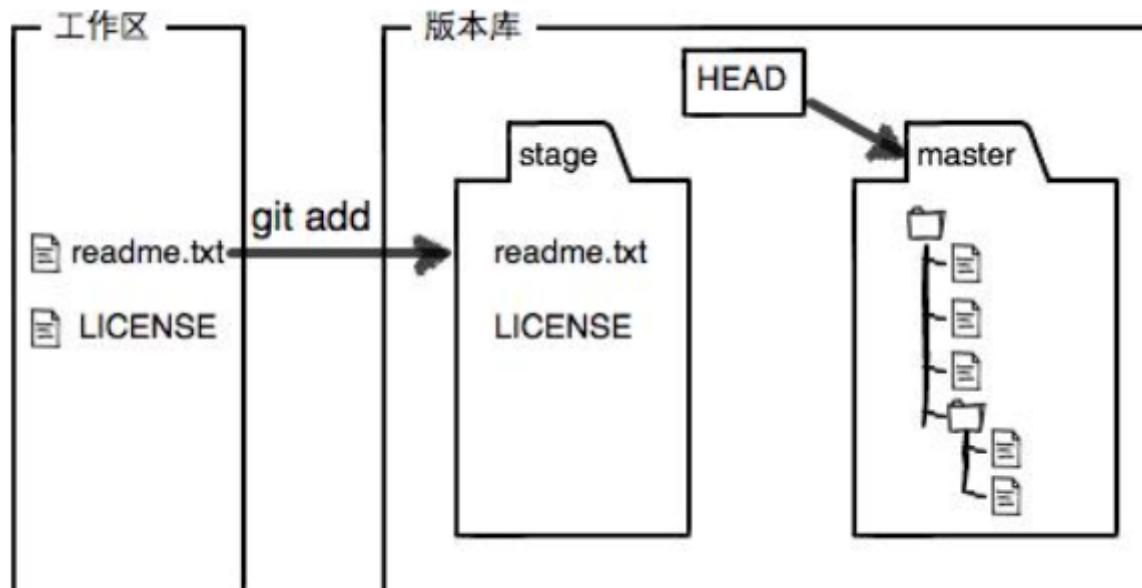
场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](#)一节，不过前提是没有推送到远程库。

删除本地文件后从版本库中恢复文件

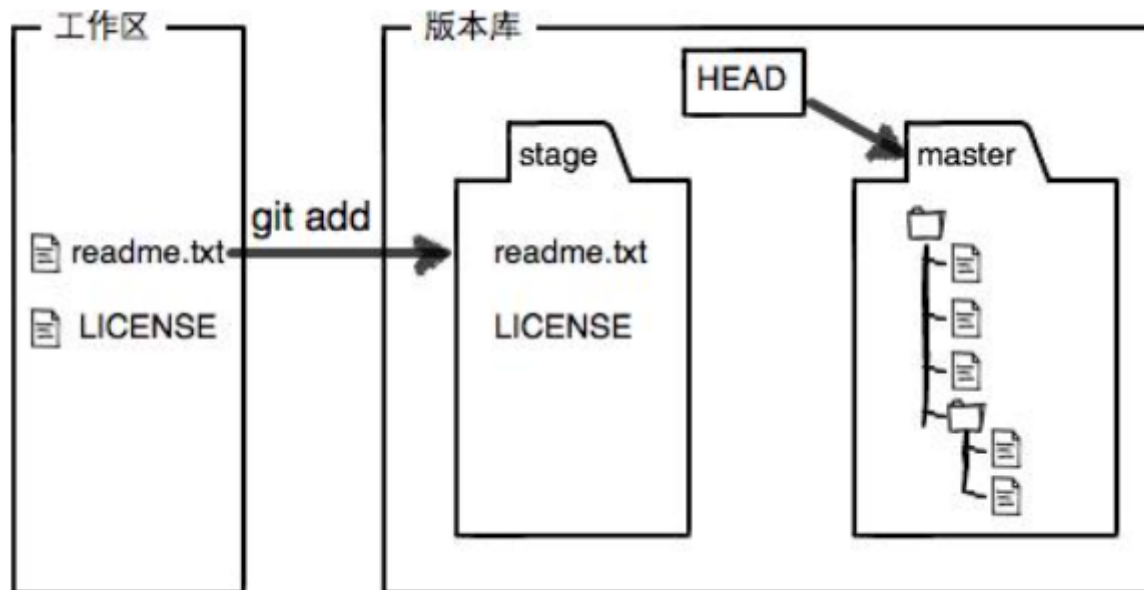
```
git checkout -- text.md
```

工作区 暂存区 版本控制区

使用 `git add` 把文件放到stage中后



使用 `git commit` 把文件放到 `master` 分支后



为什么需要SSH Key密钥对

为什么GitHub需要SSH Key呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。

当然，GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了。

最后友情提示，在GitHub上免费托管的Git仓库，任何人都可以看到喔（但只有你自己才能改）。所以，不要把敏感信息放进去。

如果你不想让别人看到Git库，有两个办法，一个是交点保护费，让GitHub把公开的仓库变成私有的，这样别人就看不见了（不可读更不可写）。另一个办法是自己动手，搭一个Git服务器，因为是你自己的Git服务器，所以别人也是看不见的。这个方法我们后面会讲到的，相当简单，公司内部开发必备。

确保你拥有一个GitHub账号后，我们就即将开始远程仓库的学习。

如何加速国内浏览github的速度

在浏览器上安装github加速器插件

如何关联远程库并推送到远程库

小结

要关联一个远程库，使用命令 `git remote add origin git@server-name:path/repo-name.git`；

关联一个远程库时必须给远程库指定一个名字，`origin` 是默认习惯命名；

关联后，使用命令 `git push -u origin master` 第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令 `git push origin master` 推送最新修改；

分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

远程仓库与本地仓库

本地仓库可在不联网时编辑

只需要在联网时

分支的重要性

分支在实际中有什么用呢？假设你准备开发一个新功能，但是需要两周才能完成，第一周你写了50%的代码，如果立刻提交，由于代码还没写完，不完整的代码库会导致别人不能干活了。如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。

现在有了分支，就不用怕了。你创建了一个属于你自己的分支，别人看不到，还继续在原来的分支上正常工作，而你在自己的分支上干活，想提交就提交，直到开发完毕后，再一次性合并到原来的分支上，这样，既安全，又不影响别人工作。

其他版本控制系统如SVN等都有分支管理，但是用过之后你会发现，这些版本控制系统创建和切换分支比蜗牛还慢，简直让人无法忍受，结果分支功能成了摆设，大家都不去用。

但Git的分支是与众不同的，无论创建、切换和删除分支，Git在1秒钟之内就能完成！无论你的版本库是1个文件还是1万个文件。

Git Add

`git add xx` 命令可以将xx文件添加到暂存区

如果有很多改动可以通过 `git add -A .` 来一次添加所有改变的文件。注意 `-A` 选项后面还有一个句点。

**** **** `git add -A` 表示添加所有内容

**** **** `git add .` 表示添加新文件和编辑过的文件不包括删除的文件；

`git add -u` 表示添加编辑或者删除的文件，不包括新添加的文件

Git

git取消与远程仓库的连接

连接远程仓库 `git remote add origin 仓库地址`

查看[远程连接](#) `git remote -v`

git取消与远程仓库的连接 `git remote remove origin`

git与远程库建立联系并推入

使用git将本地项目推送到远程仓库[github - 简书 \(jianshu.com\)](#).

github分支的创建与合并

点击create直接创建

pull request中选择两个分支，然后选择create pull request创建就可以开始合并了

注意箭头所指的表示最终能够留下来的。另一个被合并了

How to Push Project in Vscode to Github

[Push](#)

问题

[Git报错解决: OpenSSL SSL_read: Connection was reset, errno 10054 错误解决](#)

重新开始

更换账号推送失败

解决办法

直接在sourcetree中删除账户再重新绑定一次

注意点

两个账号来回切换是没问题的,但是有时候在windows上容易出错,建议只用一个账号

是我自己理解错了,我没有在上面加上两个github账号

remote: Permission to 1RanShy/1.git denied to RanShy1011.

fatal: unable to access '<https://github.com/1RanShy/1.git/>': The requested URL returned error: 403

两个原因:

1. 有多个账号的凭据 这个其实是不关系的
2. 主要是我选择的是ssh但是用的确实ssh

sourcetree 顺序

首先在仓库里 git init

然后再sourcetree里添加这个库

再在设置里填写上远端库的url 地址

Windows



出现这个问题是因为老的账号的凭据没有删除,要删除老帐号的凭据,添加新账号的凭据才行.

新账号: 1RanShy

老帐号:RanShy1011

老帐号的凭据一直留在电脑里,导致git登录时一直推送给老帐号,但是推送的仓库地址却是新账号的,发生错误.

凭证更改后就出现了这个错误

```
PS C:\Users\RanShy\Desktop\Git> git push https://github.com/RanShy1011/test.git
remote: Permission to RanShy1011/test.git denied to 1RanShy.
fatal: unable to access 'https://github.com/RanShy1011/test.git/': The requested URL returned error: 403
PS C:\Users\RanShy\Desktop\Git> |
```

可以看出确实是由于凭证的原因导致的,而不是与什么ssh

修改电脑的git账户，git的账号配置

原创 精哥哥yxy5558 于 2019-06-19 15:15:01 发布 13255 收藏 22

版权

git 专栏收录该内容

0 订阅 20 篇文章 订阅专栏

账户配置

接受一部老电脑，进行代码开发，需要把原来主人的账号（例如：github账号）改掉自己的

1.进入控制面板——用户账户



2.管理Windows凭据



3.删除对应的账号



可以全部删除，到时候需要密码，会弹出窗口，让你填的。

git的账号配置

git用户信息

第一个要配置的是你个人的用户名和电子邮件地址。这两条配置很重要，每次 Git 提交时都会引用这两条信息，说明是谁提交了更新，所以会随更新内容一起被永久纳入历史记录：

```
1 $ git config --global user.name "John Doe"
2 $ git config --global user.email "johndoe@example.com"
```

📖 文章知识点与官方知识档案匹配，可进一步学习相关知识

Git技能树 > 首页 > 概览 3505 人正在系统学习中

精哥哥yxy5558

码龄6年 企业员工

207

1万+

134万+

51万+

原创

周排名

总排名

访问

等级

6059

62

99

44

233

积分

粉丝

获赞

评论

收藏

私信

关注

显示推荐内容

Mac

也是像Mac一样,不过是在 启动台的 钥匙串里更改. 注意不要删错了,如果删错就需要重新加载账号[在sourcetree中],获得github的授权.

每次切换都需要删除所有关于github的信息

在sourectree中添加新的账号,关联guthub即可

和windows不一样,windows只需改一条

如何关掉已经git追踪的文件 文件

```
git rm --cached *.json
```

然后再gitignore中写下这个需要忽略的文件就算完成

文件夹

```
git rm -r --cached .obsidian
```

命令行推送失败

unable to access '<https://github.com/1RanShy/Pet-Automatic-Feeder.git/>': The requested URL returned error: 403

username: 1RanShy

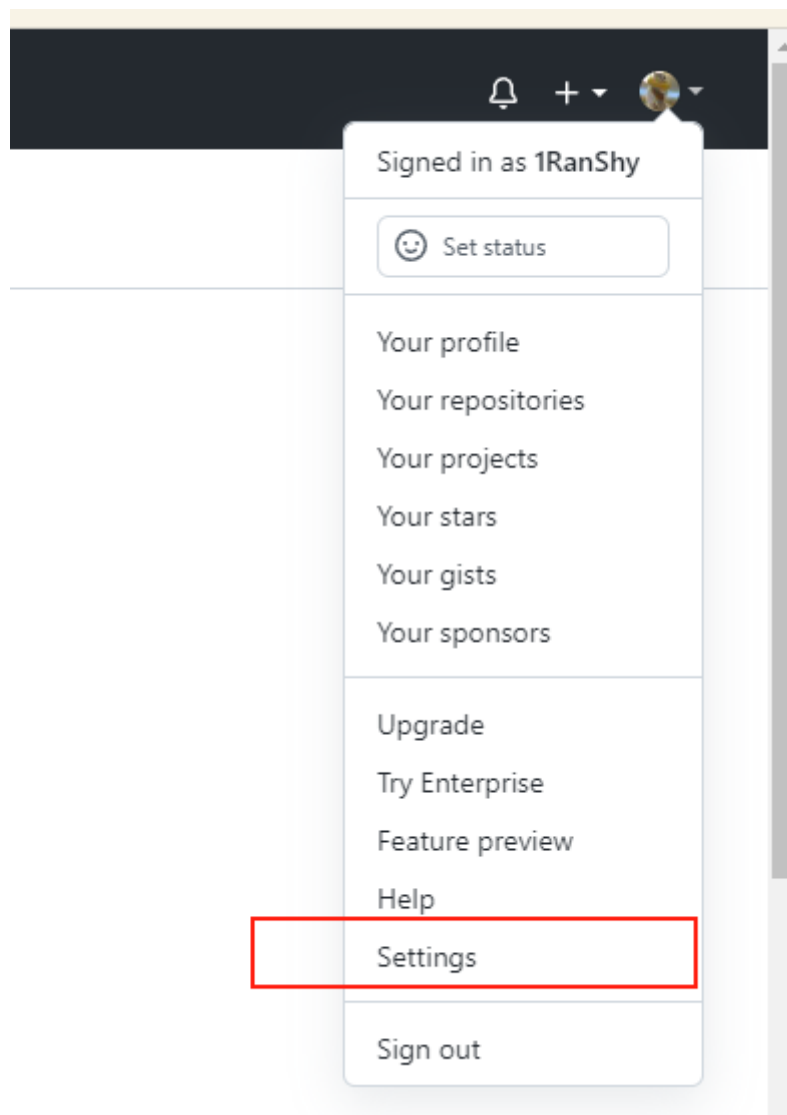
password: 直接输入密码 这是错误的








#token

```
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
```







这才是对的

如何生成密匙




-  Emails
 -  Password and authentication
 -  Sessions
 -  SSH and GPG keys
 -  Organizations
 -  Moderation 
-



Code, planning, and automation

-  Repositories
 -  Codespaces
 -  Packages
 -  Copilot
 -  Pages
 -  Saved replies
-



Security


-  Code security and analysis
-

Integrations

-  Applications
 -  Scheduled reminders
-

Archives

-  Security log
 -  Sponsorship log
-

 Developer settings

GitHub Apps OAuth Apps Personal access tokens ^

Fine-grained tokens (Beta) Tokens (classic)

Personal access tokens (classic)

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

test — admin:org, repo

Last used within the last week Delete

⚠ This token has no expiration date.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups

这两个要选上 否则生成的密匙只能访问不能修改

如何选择主分支为master而不是main

拉取 推送 关联远程分支

[git操作之pull拉取远程指定分支以及push推送到远程指定分支_51CTO博客_git pull 拉取远程分支](#)

二、push操作

1、将本地当前分支 推送到 远程指定分支上（注意：pull是远程在前本地在后，push相反）：

```
1. git push origin <本地分支名>:<远程分支名>
```

2、将本地当前分支 推送到 与本地当前分支同名的远程分支上（注意：pull是远程在前本地在后，push相反）：

```
1. git push origin <本地分支名>
```

3、将本地当前分支 推送到 与本地当前分支同名的远程分支上(需先关联远程分支，方法见文章末尾)

```
1. git push
```

同样的，推荐使用第2种方式，git push origin <远程同名分支名>

附：

// 将本地分支与远程同名分支相关联

```
1. git push --set-upstream origin <本地分支名>
```

Git如何查看本地分支状态

要查看远程分支是否有更新，你可以使用 `git fetch` 命令。这个命令会抓取所有远程仓库的最新提交，但并不会自动合并或重置本地分支。

如果你想要查看具体有哪些远程分支有更新，可以使用 `git branch -av` 命令。这个命令会列出所有本地和远程分支的名称，并且会标注出哪些分支是本地分支，哪些分支是远程分支。如果远程分支比本地分支新，它会在分支名称后面附加一个 `*` 号。

例如，如果你的本地仓库中有一个名为 `develop` 的本地分支，远程仓库中有一个名为 `origin/develop` 的远程分支，并且远程分支比本地分支新，那么输出的结果可能是这样的：

GIT

```
$ git branch -av
* develop                abcdefg  Fix some bugs
  remotes/origin/develop  hijklm  Add some features
```

复制代码

这里，`develop` 分支前面的 `*` 号表示它是当前正在使用的分支，而 `origin/develop` 分支后面的 `*` 号表示它是远程分支，并且比本地分支新。

希望这些信息对你有帮助。

如何缩小git文件的大小

不要克隆 因为克隆会包含你整个的git

新建初始化本地仓库下拉只会保存你现在的开始的git,会让你的git小不少,不过git status 不会给你 up to date提示

暂存提交合二为一

```
git commit -am "提交描述"
```

Pull 与Clone的区别

clone 是本地没有 repository 时, 将远程 repository 整个下载过来。

pull 是本地有 repository 时, 将远程 repository 里新的 [commit](#) 数据(如有的话)下载过来, 并且与本地代码merge。

删除所有提交记录使其成为新库

[彻底清除git所有历史提交记录使其为“新”库_RiskAI的博客-CSDN博客](#)

解决方法

1. 对于比较少分支的仓库-方法1

思路: 先查看远程分支, 然后在本地创建和远程仓库同名的分支。

关键命令: `git branch dev origin/dev`, 即新建一个本地分支来跟踪远程的某一分支, 创建该分支后, `dev origin/dev`, 创建分支, 并切换到该分支)

如何创建一个空分支并推送

[如何新建一个空的分支 - 飞翔的蜗牛~- 博客园](#)

问题场景:

需要一个空的分支(啥都没有), 用于存放我需要的某一个或几个文件;

问题分析:

在Git中创建分支, 是必须有一个父节点的, 也就是说必须在已有的分支上来创建新的分支, 如果你的工程已经进行了一段时间, 这个时候是无法创建空分支的。

解决方法:

1、使用 git checkout的--orphan参数:

```
git checkout --orphan empty_branch
```

该命令会生成一个叫 empty_branch 的分支，该分支会包含父分支的所有文件。但新的分支不会指向任何以前的提交，就是它没有历史，如果你提交当前内容，那么这次提交就是这个分支的首次提交。

2、删除所有文件：

我们想要空分支，所以我们需要把当前内容全部删除，用git命令

```
git rm -rf .
```

3、提交分支：

如果没有任何文件提交的话，分支是看不到的，所以我们需要创建一个新文件，然后提交则新建的branch就会显示出来。

```
echo '# new branch' >> README.md
```

```
git add README.md
```

```
`git commit -m 'new branch'
```

4、最后push到远程仓库，则新的空分支就创建成功了。

```
git push origin empty_branch
```

重命名分支并推送

[如何在 Git 中重命名本地或远程分支](#)

本地

重命名本地分支

```
git branch -m new-name
```

远端

删除远端分支名

```
git push origin --delete old-branch-name
```

重命名远端分支

```
git push origin -u new-branch-name
```

1. 旧分支：oldBranch
2. 新分支：newBranch

步骤：

1. 先将本地分支重命名

```
git branch -m oldBranch newBranch复制代码
```

2. 删除远程分支（远端无此分支则跳过该步骤）

```
git push --delete origin oldBranch复制代码
```

3. 将重命名后的分支推到远端

```
git push origin newBranch复制代码
```

4. 把修改后的本地分支与远程分支关联

```
git branch --set-upstream-to origin/newBranch
```

快速创建readme文件

```
echo '# new branch' >> README.md
```

不commit就会影响其他分支

问题描述：今天遇到一个git分支切换的问题，我在分支A上做了修改，然后切换到其他分支后发现其他分支上也存在A分支上的修改。（我记得之前碰到这种情况，是无法切换分支的，git会提醒当前A分支上有未提交的改动，这次虽然能切换了（当时就感觉奇怪），果然又碰到现在这个问题）

原因：如果当前分支所做的修改没有提交就切换去其他分支也会看到相同的修改，所以解决这个问题有两个办法。

解决办法：

用 git add和 git commit提交修改，只要用 git status 检查工作区和暂存区是干净的就可以了。那如果我当前分支上的工作还没做完，不能提交，但又想去其他分支，这时候可以把当前分支的

工作现场隐藏起来。用 `git stash` 隐藏当前工作现场，这个时候用 `git status` 查看工作区是干净的，所以就可以放心地去其他分支了。用 `git stash list` 可以查看隐藏起来的工作现场
恢复工作现场的两种方法：

用 `git stash apply` 恢复，但是恢复后，stash 内容并不删除，需要用 `git stash drop` 来删除；
用 `git stash pop`，恢复的同时把stash内容也删了，这时候用 `git stash list` 就看不到任何 stash 内容了

可以多次 stash，恢复的时候，先用 `git stash list` 查看，然后用 `git stash apply stash@{0}` 或者 `git stash pop stash@{0}` 恢复指定的stash

fetch 与 status

有时候status并不会提示你
update to origin/main
这是因为你的是下拉的,缺失了部分commit history

这时候你就可以使用fetch来看看是不是由文件需要跟新
如果什么都没有说明不需要,如果命令行出现了内容就说明需要更新.

Gitignore