## w3c git教程：<https://www.w3cschool.cn/git/git-tutorial.html>

## Git软件到底包含什么？

Git是一个分布式的代码托管系统。说白了一个服务端。而为了让更多的人包括小白，轻而易举的就能将自己的代码或文件保存到Git服务端上。Git肯定会提供一些客户端工具，这些客户端工具与服务端进行通信。用户只需要操作客户端，点几个按钮，就能将代码传到git服务端上或者从Git上下载自己的文件。当然GIt客户端也提供了命令方式。不管使用命令还是使用界面工具，都是客户端。因此，我们实际上要操作的就是在Git客户端和Gti服务端之间交互。

Git客户端安装在本机。所以我们的文件可以在本机创建仓库。

Git服务端在远程，所以我们需要创建账号，git为我们分配空间，我们可以创建仓库。

通过一定的交互，可以将本地仓库中的文件同步到Git的我们的远程仓库中。

那么git到底有什么作用呢？

* 云盘功能

这点，可以理解为何百度网盘一样。既然是网盘，那肯定上传文件、下载文件的功能。的确如此，但仍有一些区别。git一般作用存放代码、设计文档、素材等。而网盘则是什么资源都往里面怼。之所以强调这个，是因为git中的文件是以仓库(Repository)为单位进行管理，更多的情况下，我们把一个仓库完整下载进来，功能才会完善。因此文件越小，下载、上传速度会越快。而网盘可以看作是远程的文件系统。

* 版本控制

这一点是git区别于网盘的重要原因。版本更新迭代是软件开发中一个常见的行为习惯。如何更好的控制保存旧版本、更新新版本是版本的首要任务。版本控制软件提供了如下常用功能：提交新版本、回退到某个保本、合并到某个版本。

* 代码托管

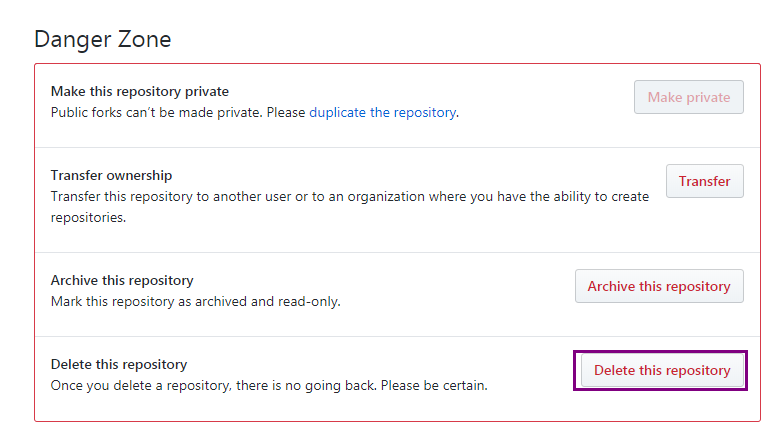
git的一大亮点是代码托管，开发者把项目放到git上的公共仓库之后。所有的用户都可以将浏览，对这个项目感兴趣的人希望会收藏或者复制一份代码到自己的git仓库，他可以fork这个仓库。fork是刀叉的意思，打个不恰当的比喻，你可以把项目当做一块牛排，用户对这个牛排感兴趣，就自己用刀叉叉起这个牛排，放到自己碗里。同时如果，其它用户发现了项目的bug，想要修改。他可以发送一个修改请求给项目拥有者。项目拥有者看到之后，既可以拒绝，也可以接收这个修改请求。如果接收，那么修改后的版本将会合并到最新版本。所有用户都会适时看到最新的版本状况。

Git的代码托管使得多人并行开发项目变得轻而易举，提高了开发效率。尤其对于大型项目，例如：linux，更是急中生智智慧的一个伟大工程。

## 那就开始吧

## 如何删除远程仓库？

进入仓库，找到settings，拉倒最底部。



## Project和Repository区别？

A repository contains all project files，including the revision history。

## 在本地将一个写好的java工程提交到我的某个git仓库中。

Step1:创建一个远程仓库。例如名字叫做MiniHttp 。

Step2:创建或初始化一个本地仓库MiniHttp。可以将java工程目录转化为本地仓库。

Step3: 使用git add . 将本地仓库的所有文件添加到工作缓存区。

Step4：使用 git commit –m “备注” 提交事务。

Step5: 使用 git remote add origin将本地仓库和远程仓库关联 。

其用法： git remote add origin 仓库的ssh或https地址



或者

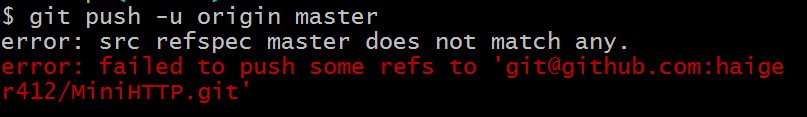


Step6:使用git push –u origin master 命令，将本地仓库的内容推送到远程仓库。

此操作，将本地仓库的master推送给远程仓库的master分支。

-u参数表示将本地仓库的master分支和远程仓库的master分支关联起来。

如果执行过程中出现以下错误：表明SSH配置有无。

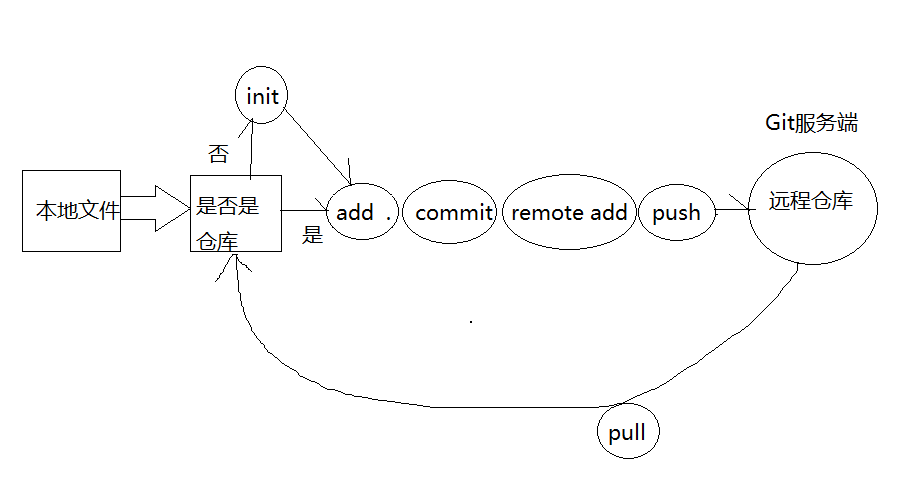


可以选择用Https方式提交，用HTTPS时需要输入用户名和密码。

请注意：一定要先init add commit remote push这5个步骤。

少了add，操作，表明你没有新文件要提交。

## 如何在远程主机上获取项目，更新后再提交？



## 本机创建一个仓库

可以新建一个文件夹，也可以在某个已存在的文件夹目录下打开cmd或者git Bash

输入： git init 即可。

## 使用Git客户端配置

* **配置用户名和和邮箱**

$ git config --global user.name "haiger412"

$ git config --global user.email "312525647@qq.com"

可以通过

git config –global查看配置选项。

通过`

git config --global –l查看已配置的

* **配置SSH**

Step1:配置好邮箱和用户名。

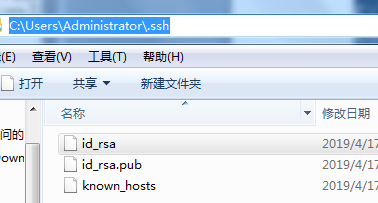
Step2:生成公钥和秘钥。

ssh-keygen -t rsa -C 邮箱



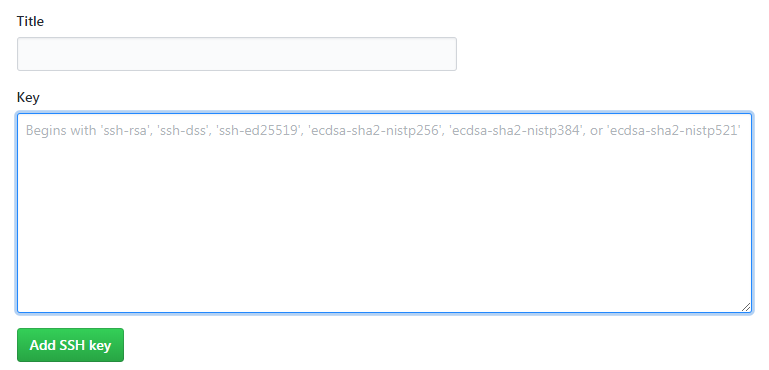
Step3: 3次回车。

完了只会，会在默认目录生成三个文件。



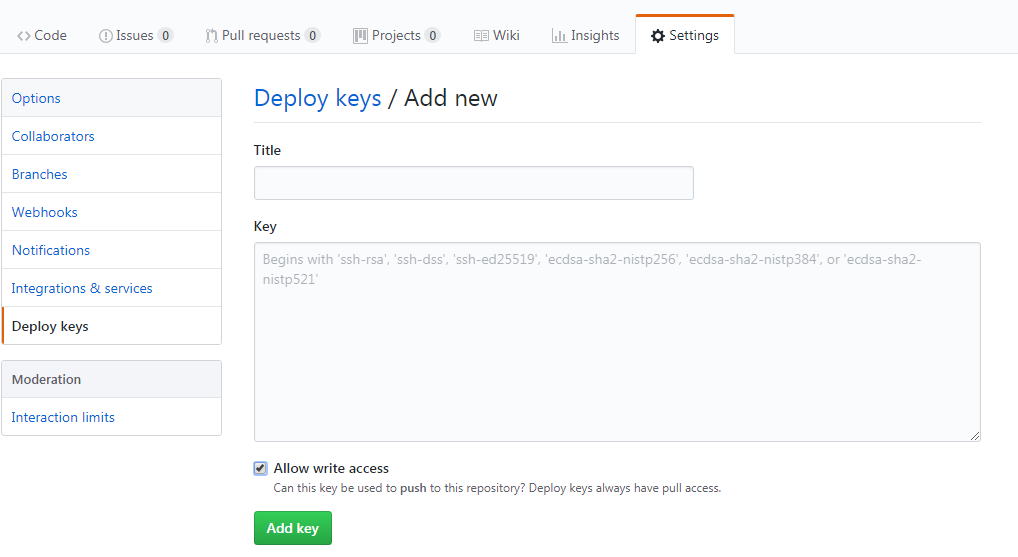
Step4：登陆git用户账号，settings-> SSH and GPG keys->new SSH key

填写title和key，点击addSSH KEY。完事。其中title随意，key来自于Step3生成的id\_rsa.pub文件的所有内容。



如果希望多人提交到同一个用户的仓库中，如何配置的？

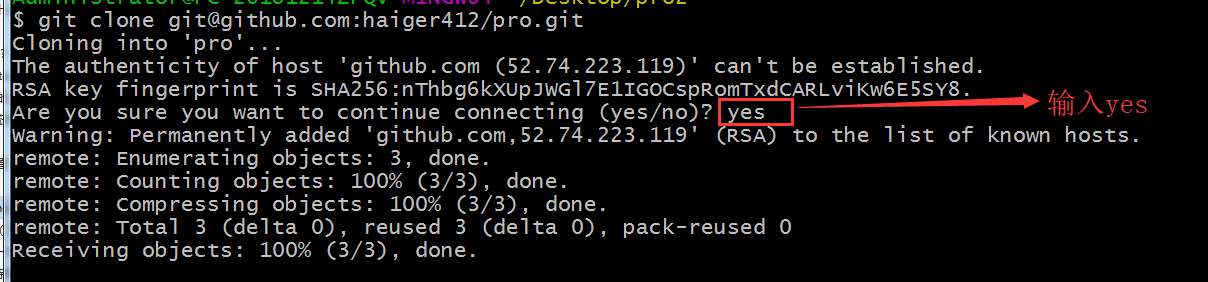
方法二，在某个git账户的某个仓库的setting中，配置deploy key



## 缺少.ssh目录。

重新生成ssk之后，该目录会生成。

## .ssh目录下没有know\_host文件，导致用ssh地址克隆时有警告。



## 使用Git下载文件（克隆仓库）

* 创建一个仓库之后，git会为用户的仓库分配一个地址。改地址可以是http的url地址，也可以是ssh的地址。
* 
* 

Git提供了两种方式下载：https和ssh。

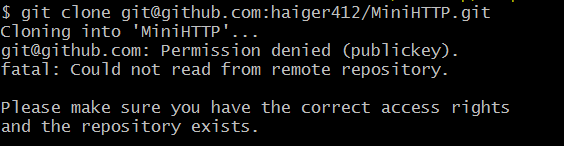
可以使用浏览器下载，当然也可以使用git客户端，借助一些命令来下载。下载命令为，git clone 仓库地址。Git把下载的过程成为clone，所以一般我们说克隆仓库，而不说下载仓库。

使用方法：在某个文件夹目录下执行，git clone 仓库地址

使用ssh方式克隆： git clone [git@github.com:haiger412/MiniHTTP.git](mailto:git@github.com:haiger412/MiniHTTP.git)

使用https方式克隆： git clone <https://github.com/haiger412/MiniHTTP.git>

对于https来说，直接下载就能成功。但对于ssh方式来说，如果未做任何配置，会出现以下错误：



需要配置好ssh即可。详情见<<为客户端配置ssh>>

## 使用HTTPS和SSH区别

很多操作需要用到仓库地址，https和ssh一些使用上的区别如下：

https方式每次fetch和push操作都需要输入账号和密码，比较麻烦。

ssh方式则无需如此，但需要提前配置好ssh key。并且只有项目的拥有者才能使用ssh url方式去克隆和提交。ssh url采用公钥 + 秘钥方式方式验证。

更多ssh验证原理，查看<https://baijiahao.baidu.com/s?id=1612411213158569988&wfr=spider&for=pc>

## Git服务器搭建（待完善）

搭建Git服务器的意义在于，团队协作时，可以基于同一项目进行并行开发，提高开发效率。所以搭建一个公司或团队内部的git服务器，迫在眉睫。

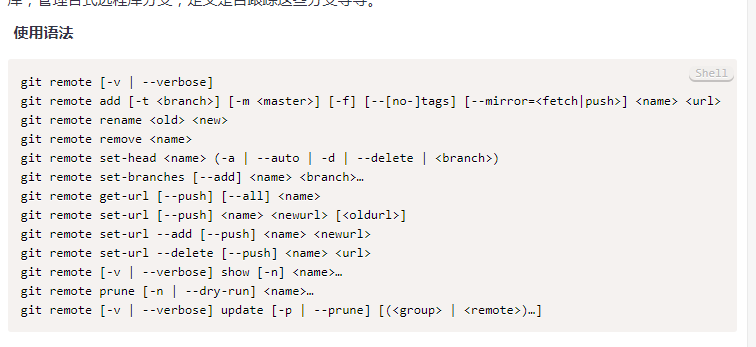
其过程就是，每个人都针对服务端的同一个仓库的代码进行修改，每个人都向服务端提交自己的修改，每个人都从服务端获取最新的版本。这样就可以确保，所有人的代码能够合并在一起，并且每个人都不是单枪匹马，而是在别人的基础上进行修改。

使用git服务器时，我们只能配置一个公钥。其它用户无法通过ssh的方式提交到我们的仓库中。**因此在搭建内部git服务器时，多人提交意味着需要在服务端配置多个用户的公钥。**

大体操作如下：

安装git、创建仓库 、收集多个用户的公钥。

## 如何添加、更新远程仓库关联



## 解决冲突

要想解决冲突，首先要了解造成冲突的原因有哪些。

情况1：张三和李四在7:00同时克隆一个仓库，将一个文件内容进行修改，张三先push，李四后push。提示push失败。

情况2：张三和李四在7:00同时克隆一个仓库，删除了一个文件，张三先push，李四后push。提示push失败。

情况3：张三和李四在7:00同时克隆一个仓库，添加了一个文件，张三先push，李四后push。提示push失败。

可以看出，冲突的主要原因就是，一方在编辑的同时，另一方已经做了push。而push之后的变化，有可能增、删、改，统称为修改。有可能修改的文件是这方正在编辑的文件，也有可能不是。但是不管如何，只要提交时，版本有最新的。都需要进行先拉取，再合并。执行git pull过程会尝试解决冲突，如果冲突无法解决。原因可能就是你正在提交的内容，别人也已经修改了，所以git会把别人修改的和你自己修改的部分用特殊的标记表明出来。让你自己选择是选择哪一部分进行。选择好之后，重新add，commit，push就没问题了。

永远记住，git会尝试解决冲突。但针对于同一个文件变更的内容，git不可能知道用户到底如何操作。所以只有通过将变化标记出来的方式，供用户选择。

## 删除文件

1 从远程仓库克隆下来，本地删除一个文件，并同步到远程仓库。

在文件系统中删除，并且在版本库中也删除：git rm hello3.txt

再commit、最后push。

2 本地误删恢复，版本库未删除情况下。如果版本库删除了，只能重新可克隆。

git checkout -- test.txt

## Pull

什么时候需要pull，张三和李四同时克隆一个仓库，同时在编辑。但是张三第一天编辑完了，提交并推送到远程仓库了。李四第二天编辑完，需要push到远程仓库。此时，李四需要pull。因为李四需要将在张三修改完成的基础上，添加自己的代码。而不再是基于前两天的代码上添加。此时可能会出现一些冲突问题。

## push

## add

## commit

## Fetch

Git fetch origin

## Merge

|  |
| --- |
| **git merge用途**  1、用于git-pull中，来整合另一代码仓库中的变化（即：git pull = git fetch + git merge）  2、用于从一个分支到另一个分支的合并  假设下图中的历史节点存在，并且当前所在的分支为“master”：  IMG_256  那么git merge topic命令将会把在master分支上二者共同的节点（E节点）之后分离的节点（即topic分支的A B C节点）重现在master分支上，直到topic分支当前的commit节点（C节点），并位于master分支的顶部。并且沿着master分支和topic分支创建一个记录合并结果的新节点，该节点带有用户描述合并变化的信息。  即下图中的H节点，C节点和G节点都是H节点的父节点。  IMG_257  **git merge <msg> HEAD <commit>...命令**  该命令的存在是由于历史原因，在新版本中不应该使用它，应该使用git merge -m <msg> <commit>....进行替代  **git merge --abort命令**  该命令仅仅在合并后导致冲突时才使用。git merge --abort将会抛弃合并过程并且尝试重建合并前的状态。但是，当合并开始时如果存在未commit的文件，git merge --abort在某些情况下将无法重现合并前的状态。（特别是这些未commit的文件在合并的过程中将会被修改时）  警告：  运行git-merge时含有大量的未commit文件很容易让你陷入困境，这将使你在冲突中难以回退。因此非常不鼓励在使用git-merge时存在未commit的文件，建议使用git-stash命令将这些未commit文件暂存起来，并在解决冲突以后使用git stash pop把这些未commit文件还原出来。 |

## 分支

git branch -r 查看分支

git branch -a 查看所有分支：本地、远程仓库的分支。

## Clone

ssh方式克隆 git clone [git@github.com:jelly-dong/Jelly\_dong.git](mailto:git@github.com:jelly-dong/Jelly_dong.git)

https方式克隆 git clone https:/xxxxx/Jelly\_dong.git

当前目录克隆 [git@github.com:jelly-dong/Jelly\_dong.git](mailto:git@github.com:jelly-dong/Jelly_dong.git) ./

## 恢复