

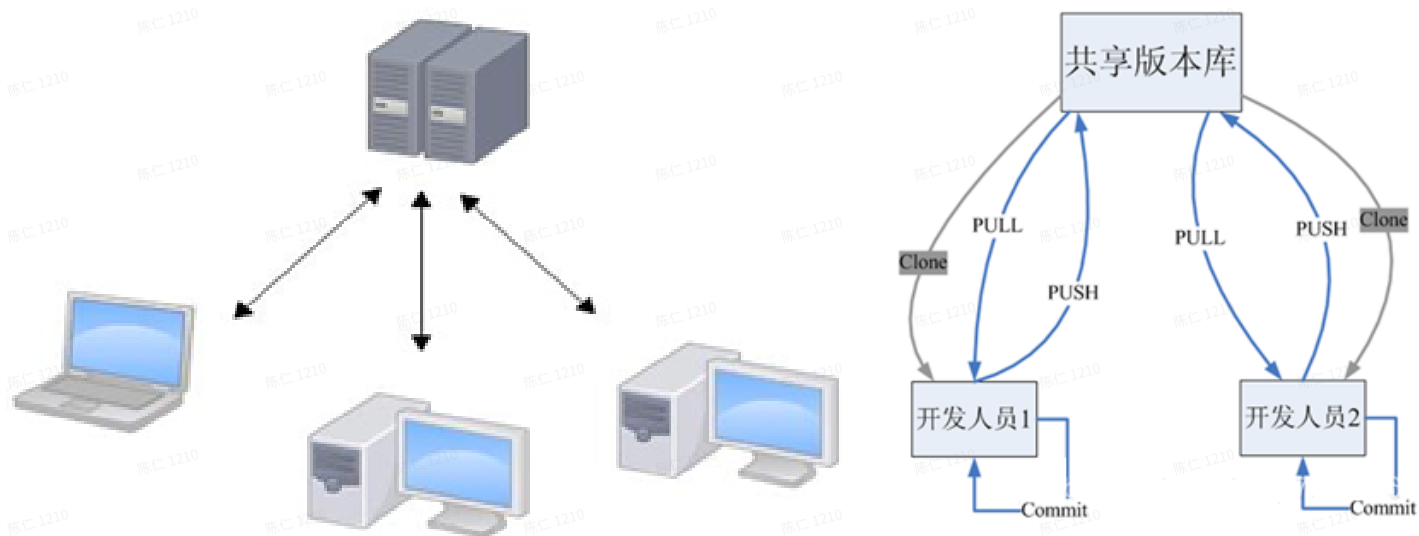


Git使用教程

1. 引言概述

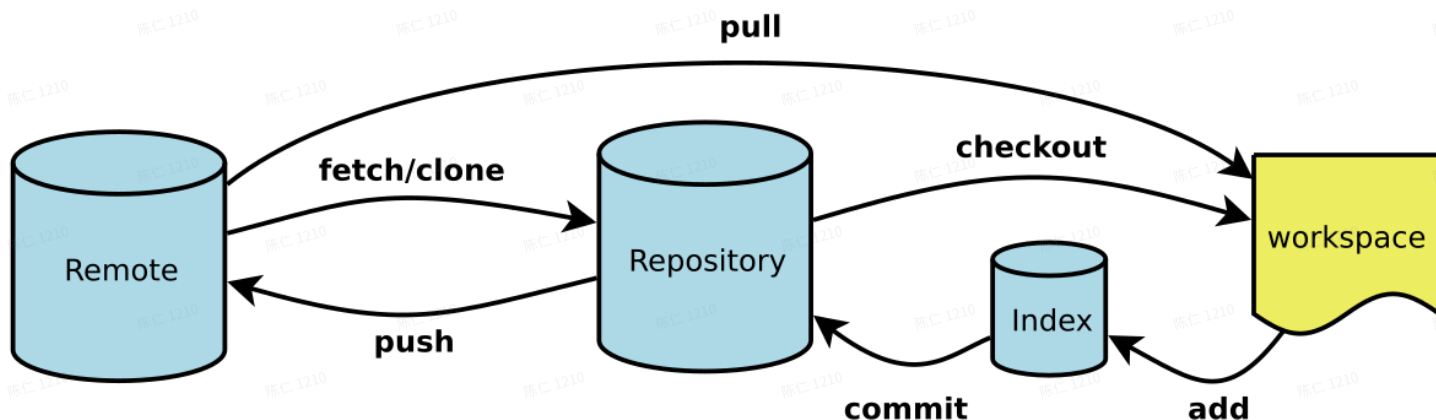
Git是目前世界上最先进的**分布式版本控制系统**（Distributed Version Control System，简称DVCS），SVN与Git的最主要的区别？SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就纳闷了。

Git是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。



2. 工作流程

Workspace: 工作区; **Index/Stage**: 暂存区; **Repository**: 仓库区 (或本地仓库); **Remote**: 远程仓库

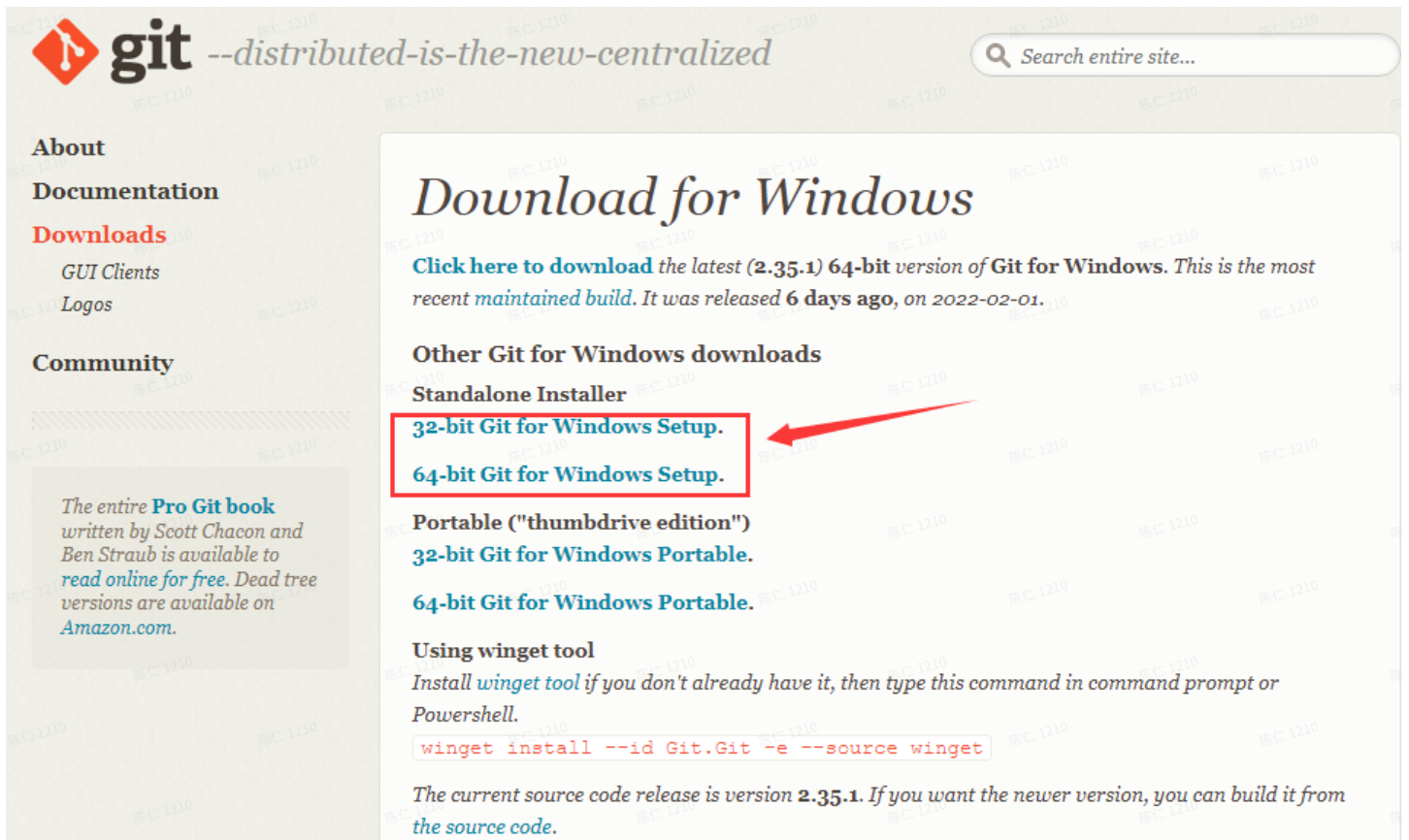


- 从远程仓库中克隆代码到本地仓库
- 从本地仓库中checkout代码然后进行代码修改
- 在提交前先将代码提交到暂存区
- 提交到本地仓库，本地仓库中保存修改的各个历史版本
- 修改完成后，需要和团队成员共享代码时，将代码push到远程仓库

3. 平台安装

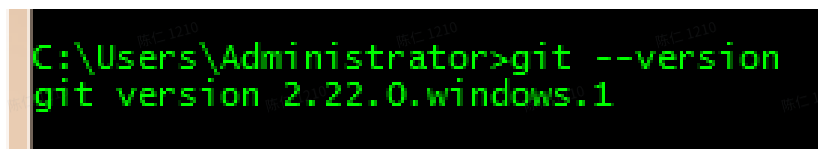
3.1 Windows安装

下载地址: <https://git-scm.com/download/win> 下载好后，双击进行下一步下一步即可



验证git安装是否成功

- 命令行输入：`git --version`，提示如下标识安装成功；同时鼠标右击如果看到有两个git单词则安装成功



3.2 Linux安装

在Linux上是有yum安装Git，非常简单，只需要一行命令 `yum -y install git`；输入 `git --version` 查看Git是否安装完成以及查看其版本号。备注：*yum安装git被安装在 `/usr/libexec/git-core` 目录下*

4. 使用方法

4.1 初始化配置

因为Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识，方式如下：

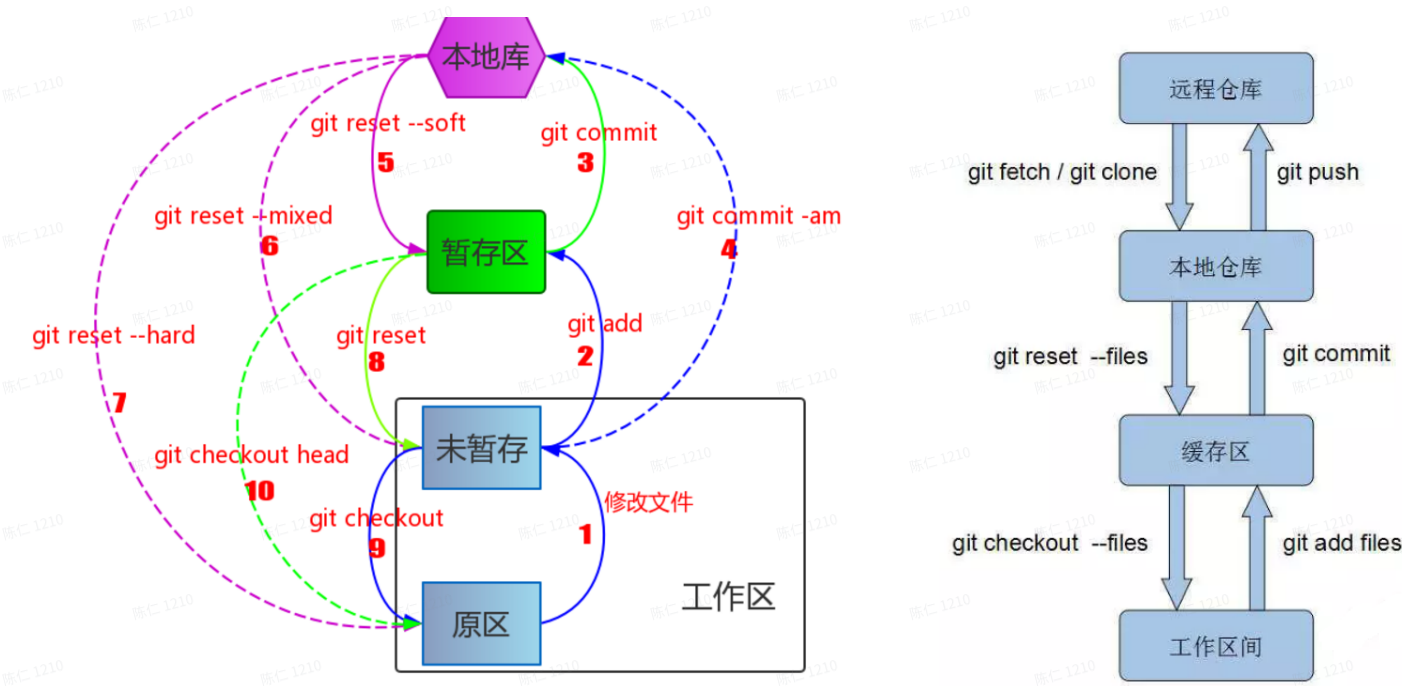
设置用户名：`$ git config --global user.name "<用户名>"`

设置用户邮箱：`$ git config --global user.email "<电子邮件>"`

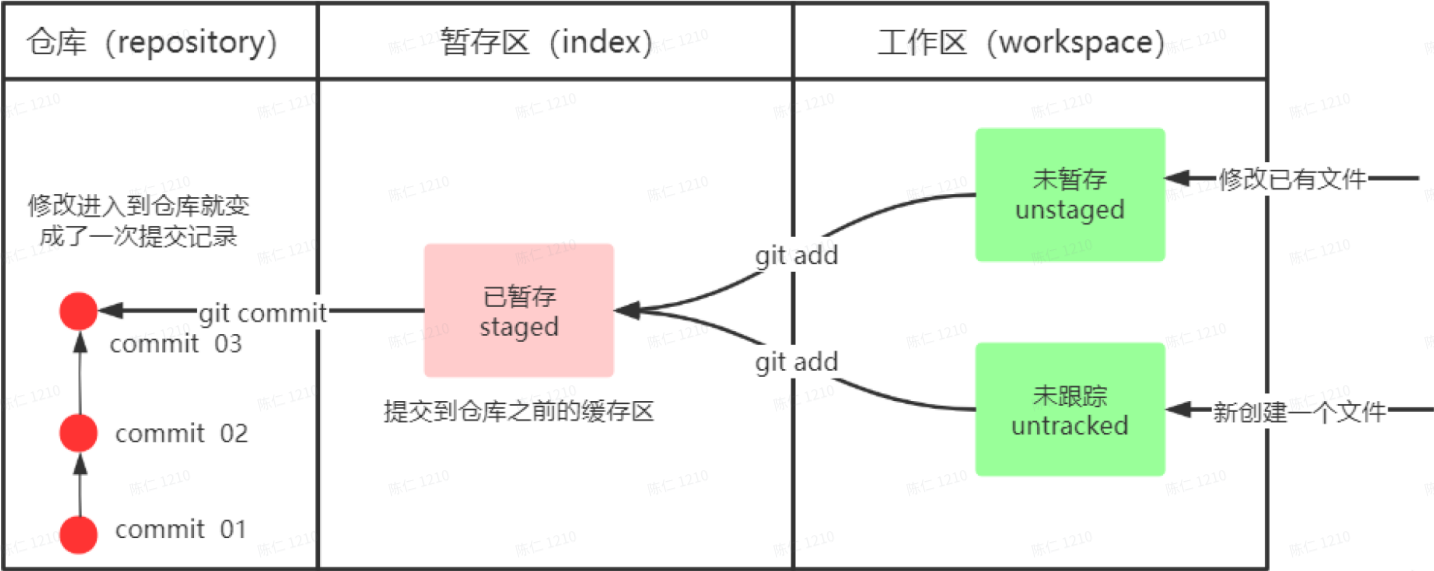
检查是否配置成功：`$ git config --list`

该配置会在github主页上显示谁提交了该文件，需注意git config --global 参数，有了这个参数表示你这台机器上所有的git仓库都会使用这个配置，当然你也可以对某个仓库指定不同的用户名和邮箱

4.2 详细流程图



4.3 暂存区/工作区



前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

第一步是用 `git add` 把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用 `git commit` 提交更改，实际上就是把暂存区的所有内容提交到当前分支。

因为创建Git版本库时，Git自动创建了唯一 `master` 分支，所以，`git commit` 就是往 `master` 分支上提交更改。

可以简单理解为，需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

俗话说，实践出真知。下面4.4章节进行Demo演示。

Git工作目录下的文件存在两种状态：

untracked 未跟踪（未被纳入版本控制）

tracked 已跟踪（被纳入版本控制）

Unmodified 未修改状态

Modified 已修改状态

Staged 已暂存状态

4.4 Demo演示

4.4.1 初始化仓库

cd 进入D盘，mkdir TestGit，通过命令 `git init` 把这个目录变成git可以管理的仓库；这时TestGit目录下会多了一个.git的目录，它是Git来跟踪管理版本的，千万不要手动乱改这个目录里面的文件，否则，会把git仓库给破坏。

```
$ pwd
/d/TestGit
陈仁@cd-chenren MINGW64 /d/TestGit
$ git init
Initialized empty Git repository in D:/TestGit/.git/
```



4.4.2 添加到版本库

第1步：在版本库TestGit目录下新建1个记事本文件 readme.txt，内容111111

第2步：查看仓库状态 `git status`

第3步：使用命令 `git add readme.txt` 添加到暂存区里面去

第4步：用命令 `git commit` 告诉Git，把文件提交到仓库，其中“提交”是注释

```
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git commit -m 'readme.txt提交'
[master 46056fb] readme.txt提交
1 file changed, 1 insertion(+), 1 deletion(-)
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git status
On branch master
nothing to commit, working tree clean
```

第5步：修改文件后可以使用 `git diff` 查看文件修改前后对比


```

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt
no changes added to commit (use "git add" and/or "git commit -a")

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git diff readme.txt
diff --git a/readme.txt b/readme.txt
index b23c1c5..b72ce18 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1 +1,2 @@
-111111
\ No newline at end of file
+111111
+222222
\ No newline at end of file

```

→ 修改后查看状态, 显示未提交

→ git diff 查看做了什么修改

```

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git add readme.txt
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.txt

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git commit -m "文件增加222222内容"
[master ad624d5] 文件增加222222内容
1 file changed, 2 insertions(+), 1 deletion(-)

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git status
On branch master
nothing to commit, working tree clean

```

→ 提交到暂存区

→ 查看状态, 修改已提交暂存

→ 提交到仓库

→ 检查状态

第6步: 查看提交日志 `git log`

```

$ git log
commit ad624d589391ef6a3da4d2560a578a9cd2c54010 (HEAD -> master)
Author: chen.ren <1757467616@qq.com>
Date: Thu Nov 9 10:15:16 2023 +0800

    文件增加222222内容

commit 46056fb38f30bc788d183690ad51f663615442b3
Author: chen.ren <1757467616@qq.com>
Date: Thu Nov 9 10:00:00 2023 +0800

    readme.txt提交

commit 3c6e8647cf6ea9086c582a3658511f48e624b420
Author: chen.ren <1757467616@qq.com>
Date: Wed Nov 8 14:08:47 2023 +0800

    readme.txt提交

```

→ 版本号

→ 最近一次提交

→ 前次提交

→ 大前次提交

4.4.3 版本回退

1) 第一种: `git reset --hard HEAD^`

如果要回退到上上个版本只需把`HEAD^`改成`HEAD^^`以此类推。那如果要回退到前100个版本的话, 使用上面的方法肯定不方便, 可以使用下面的简便命令操作: `git reset --hard HEAD~100` 即可。**log也会发生变化**

```

$ git reset --hard HEAD^
HEAD is now at 46056fb readme.txt提交

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ cat readme.txt
111111

```

```

$ git log
commit 46056fb38f30bc788d183690ad51f663615442b3 (HEAD -> master)
Author: chen.ren <1757467616@qq.com>
Date: Thu Nov 9 10:00:00 2023 +0800

    readme.txt提交

commit 3c6e8647cf6ea9086c582a3658511f48e624b420
Author: chen.ren <1757467616@qq.com>
Date: Wed Nov 8 14:08:47 2023 +0800

    readme.txt提交

```

2) 第二种: `git reset --hard 版本号`

在刚才的基础上, 我们已经恢复到最近一次提交版本, 那现在又想回复回去, 但log上看不到, 该怎么办, 如下操作

```

$ git reflog
46056fb (HEAD -> master) HEAD@{0}: reset: moving to HEAD^
ad624d5 HEAD@{1}: commit: 文件增加222222内容
46056fb (HEAD -> master) HEAD@{2}: commit: readme.txt提交
3c6e864 HEAD@{3}: commit (initial): readme.txt提交

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git reset --hard ad624d5
HEAD is now at ad624d5 文件增加222222内容

```

→ 查看历史版本记录

→ 版本号, 恢复到最新

```

$ git log
commit ad624d589391ef6a3da4d2560a578a9cd2c54010 (HEAD -> master)
Author: chen.ren <1757467616@qq.com>
Date: Thu Nov 9 10:15:16 2023 +0800

    文件增加222222内容

commit 46056fb38f30bc788d183690ad51f663615442b3
Author: chen.ren <1757467616@qq.com>
Date: Thu Nov 9 10:00:00 2023 +0800

    readme.txt提交

commit 3c6e8647cf6ea9086c582a3658511f48e624b420
Author: chen.ren <1757467616@qq.com>
Date: Wed Nov 8 14:08:47 2023 +0800

    readme.txt提交

陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ cat readme.txt
111111
222222

```

→ 当前版本log已恢复

→ 文件内容, 增加222222

1) 撤销修改

`git checkout -- readme.txt`，把readme.txt文件在工作区做的修改全部撤销，这里有2种情况。

- ## 2) 删除文件

```
$ cat readme.txt
111111
222222
# 当前文件内容
$ git commit -m "测试提交" -p
$ cat readme.txt
111111
222222
333333
# 提交内容增加333333
$ git checkout -- readme.txt
# 撤销修改
$ cat readme.txt
111111
222222
```

```
$ rm readme.txt
```

→ 删除文件，但库里还有

```
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ ls
```

```
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ git checkout -- readme.txt
```

→ 撤销删除，并恢复文件

```
陈仁@cd-chenren MINGW64 /d/TestGit (master)
$ ls
readme.txt
```

5. 远程仓库

5.1 注册账号并创建SSH

首先注册github账号，由于本地Git仓库和github仓库之间的传输是通过SSH加密的，**第一步**：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id_rsa和id_rsa.pub这两个文件，如果有的话，直接跳过此如下命令，如果没有的话，输入如下命令。id_rsa是私钥，不能泄露出去，id_rsa.pub是公钥。

```
1 ssh-keygen -t rsa -C "youremail@example.com"
```

2 详见: https://blog.csdn.net/qq_35427589/article/details/123276929

```

$ ssh-keygen -t rsa -C "root@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c:/Users/1351231210/.ssh/id_rsa):
Created directory /c:/Users/1351231210/.ssh.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c:/Users/1351231210/.ssh/id_rsa
Your public key has been saved in /c:/Users/1351231210/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:13vgeZ3mH90ov0t8K6b72W4in085KWgde root@qq.com
The key's randomart image is:
[SHA256]

```



5.2 Github ssh key配置

登录github,打开” settings” 中的SSH Keys页面,然后点击 “Add SSH Key”,填上任意title,在Key文本框里黏贴id_rsa.pub文件的内容。

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

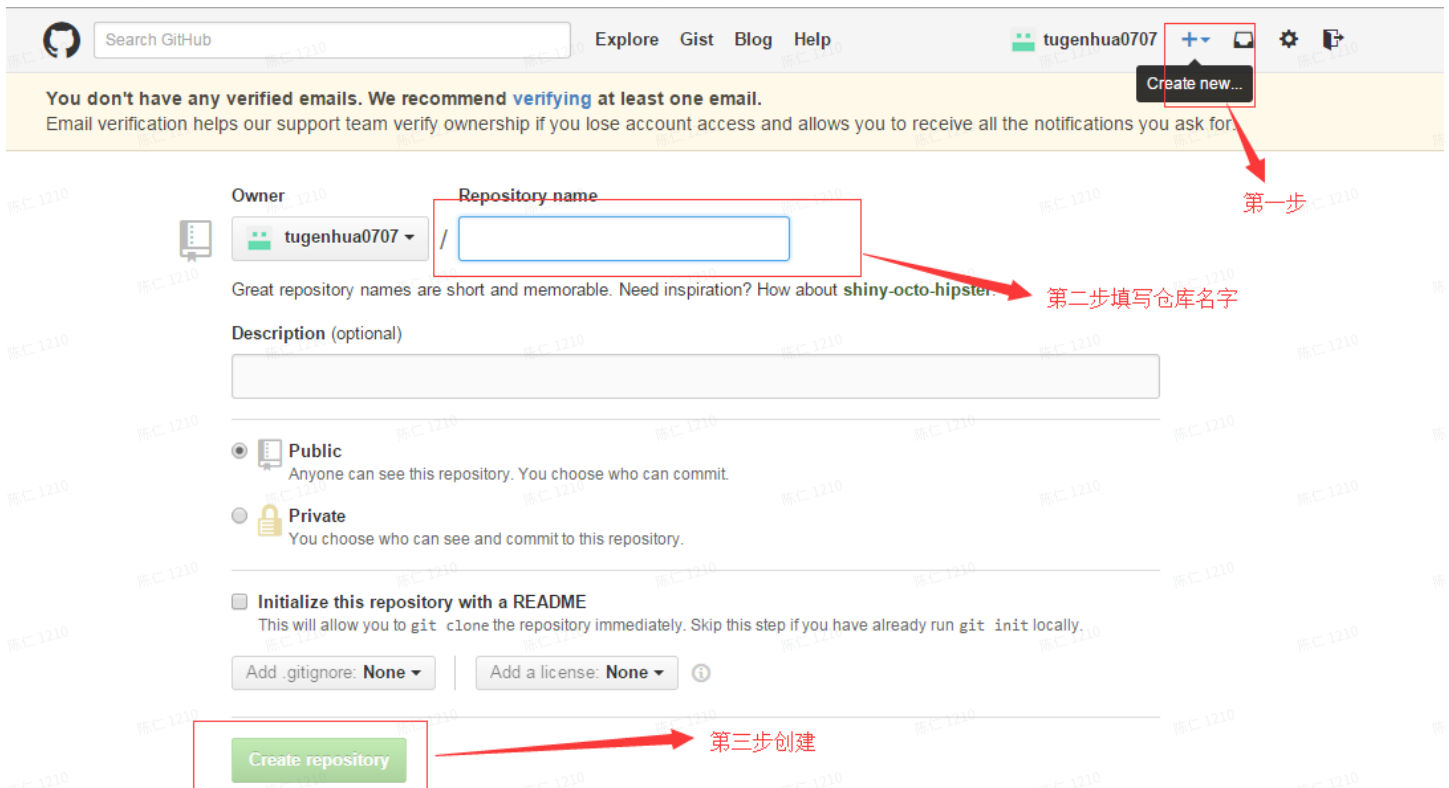


工作Demo
SHA256: [redacted] B6Kb7ZwHin085KwgdETMuQtM6a0
Added on Nov 9, 2023
Never used — Read/write

Delete

5.3 添加远程库并关联

如何添加远程库？现在的情景是：我们已经在本地创建了一个Git仓库后，又想在github创建一个Git仓库，并且希望这两个仓库进行远程同步，这样github的仓库可以作为备份，又可以其他人通过该仓库来协作。首先，登录github上，然后在右上角找到“create a new repo”创建一个新的仓库。



Owner: tugenhua0707

Repository name:

Description (optional):

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: | Add a license:

Create repository

第一步

第二步填写仓库名字

第三步创建

目前，在GitHub上的这个testgit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。下面我们进行关联。

- 1 `git remote add origin https://github.com/xxxxx/Work-Notes.git`
- 2 `git push -u origin master(或者其他branch name)`


```
$ git remote add origin https://github.com/chenren1210/Work-Notes.git
error: remote origin already exists.
```

```
陈仁@cd-chenren MINGW64 /d/TestGit (master)
```

```
$ git push -u origin master
```

```
Enumerating objects: 9, done.
```

```
Counting objects: 100% (9/9), done.
```

```
Delta compression using up to 8 threads
```

```
Compressing objects: 100% (3/3), done.
```

```
Writing objects: 100% (9/9), 685 bytes | 342.00 KiB/s, done.
```

```
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
remote:
```

```
remote: Create a pull request for 'master' on GitHub by visiting:
```

```
remote: https://github.com/chenren1210/Work-Notes/pull/new/master
```

```
remote:
```

```
To https://github.com/chenren1210/Work-Notes.git
```

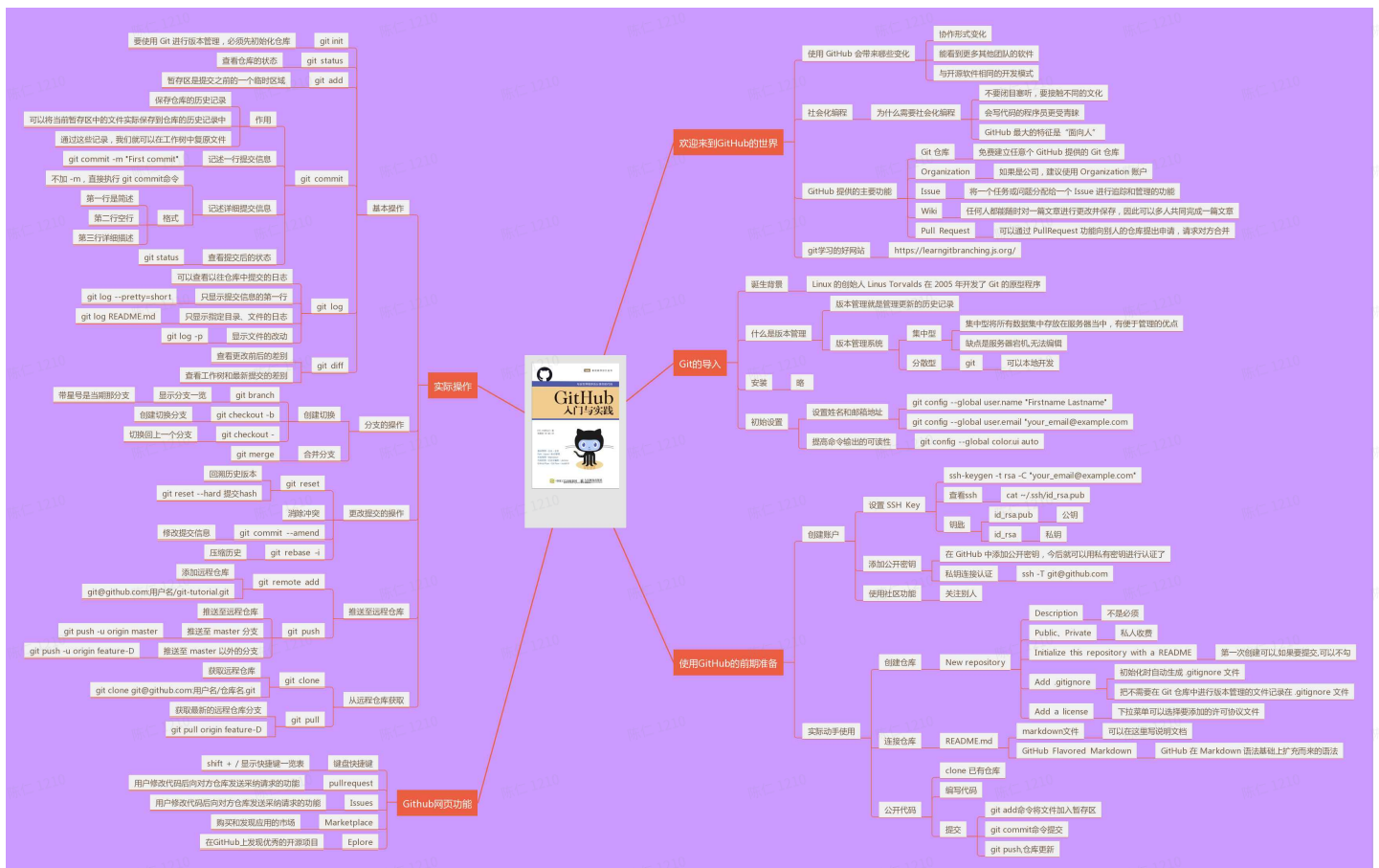
```
* [new branch] master -> master
```

```
branch 'master' set up to track 'origin/master'.
```

5.4 版本分支管理

<https://zhuanlan.zhihu.com/p/30044692>

6. Git常用命令图



[hah]:<https://www.cnblogs.com/upstudy/p/15868898.html>