

# Git-安装

2018年4月29日 16:20

## *Linux*

```
sudo apt-get install git
```

## *Windows*

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email "email@example.com"
```

# Git-版本库

2018年4月29日 16:27

[1.创建文件夹并提交成版本库](#)

[2.使用notepad++读写文件](#)

在合适的地方创文件夹，将文件夹变成合适的Git可以管理的仓库

```
$ mkdir learngit
```

```
$ cd learngit
```

```
$ git init
```

来自 <<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013743256916071d599b3aed534aab22a0db6c4e07fd0000>>

## Note:

Windows自带记事本自动为文件开头添加0xeffbfb，会产生莫名其妙的错误。使用Notepad++代替记事本

1.NotePade++ 官网 <https://notepad-plus-plus.org/>

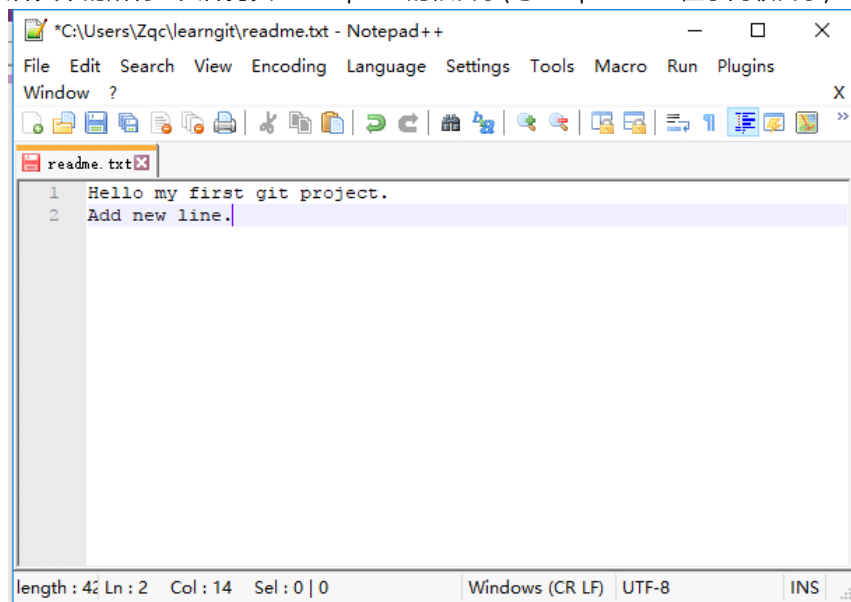
2.对于XML插件的安装:

(1) 通过网址<http://sourceforge.net/projects/npp-plugins/files/XML%20Tools/>下载XML Tools，本人下载的是Xml Tools 2.4.8 Unicode.zip

(2) 解压Xml Tools 2.4.8 Unicode.zip，关闭Notepad++

(3) 将xmltools子文件夹下的XMLTools.dll拷贝至Notepad++的plugins子文件夹

(4) 将ext\_libs子文件夹下的所有dll文件拷贝至Notepad++的根目录(与notepad++.exe位于同级目录)



# Git-版本回退

2018年4月29日 16:49

## 管理文件

- 用命令git add将文件添加到仓库  
git add readme.txt
- 使用git commit, 将文件提交到仓库  
git commit -m "information about your edit"  
-m后面输入的是本次提交的说明, 可以输入任意内容
- git status命令可以让我们时刻掌握仓库当前的状态
- git diff 用于查看尚未提交的修改。

```
$ git diff
diff --git a/readme.txt b/readme.txt
index aef65b0..6d37473 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,2 @@
-Hello my first git project.
\ No newline at end of file
+Hello my first git project.
+Add new line.
\ No newline at end of file
```

Note:

将修改使用 add 和commit命令进行提交。

## 历史版本的选择

- git log (--pretty=oneline)  
git log命令显示从最近到最远的提交日志
- git reset --hard (HEAD or 版本ID)  
版本回滚 用HEAD表示当前版本, 上一个版本就是HEAD^, 多个版本使用HEAD~100。

git reset --hard HEAD^

- HEAD 指向的版本就是当前版本, 因此, Git允许我们在版本的历史之间穿梭, 使用命令git reset --hard commit\_id。
- 穿梭前, 用git log可以查看提交历史, 以便确定要回退到哪个版本。
- 要重返未来, 用git reflog查看命令历史, 以便确定要回到未来的哪个版本。

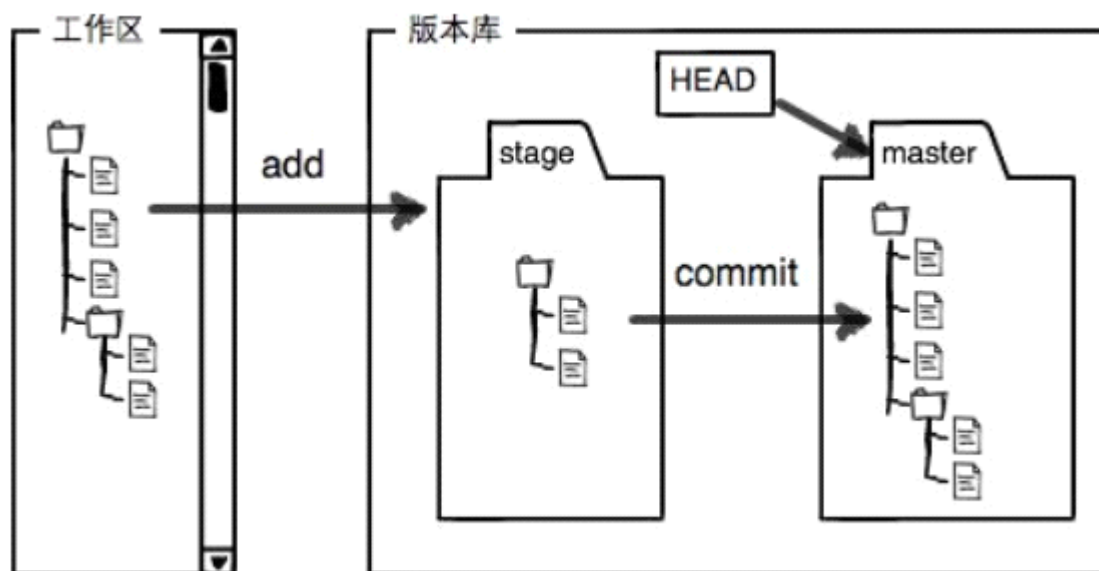
# Git-工作区和暂存区

2018年4月30日 16:51

## 针对修改进行更新版本，只提交暂存区的内容

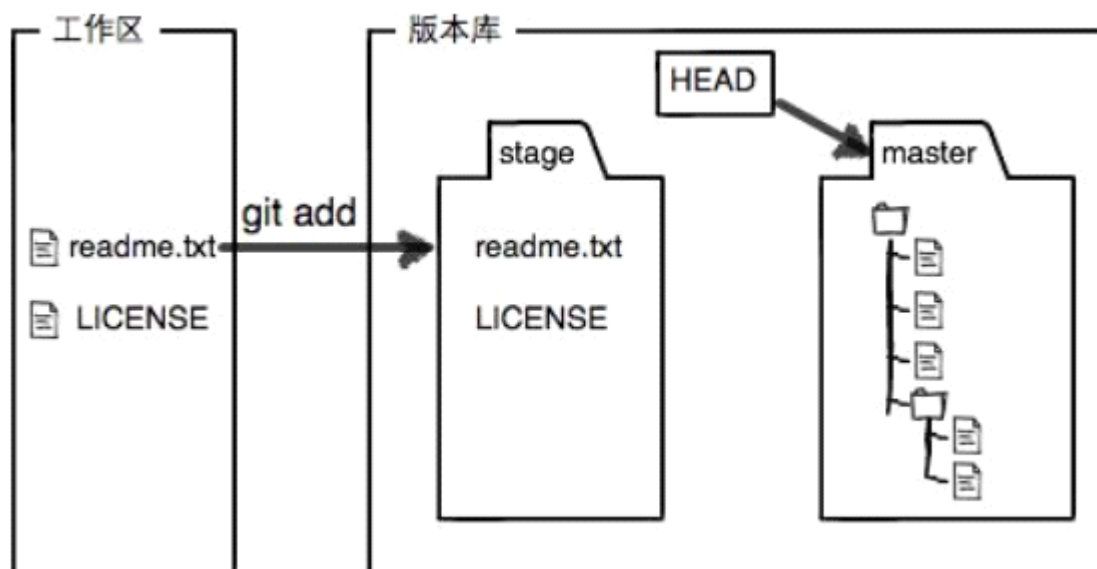
工作区 (working directory) 可视的，由用户使用git init name进行提交的。

版本库 (Repository) ，是Git的版本库。stage为暂存区，master为Git自动创建的分支。

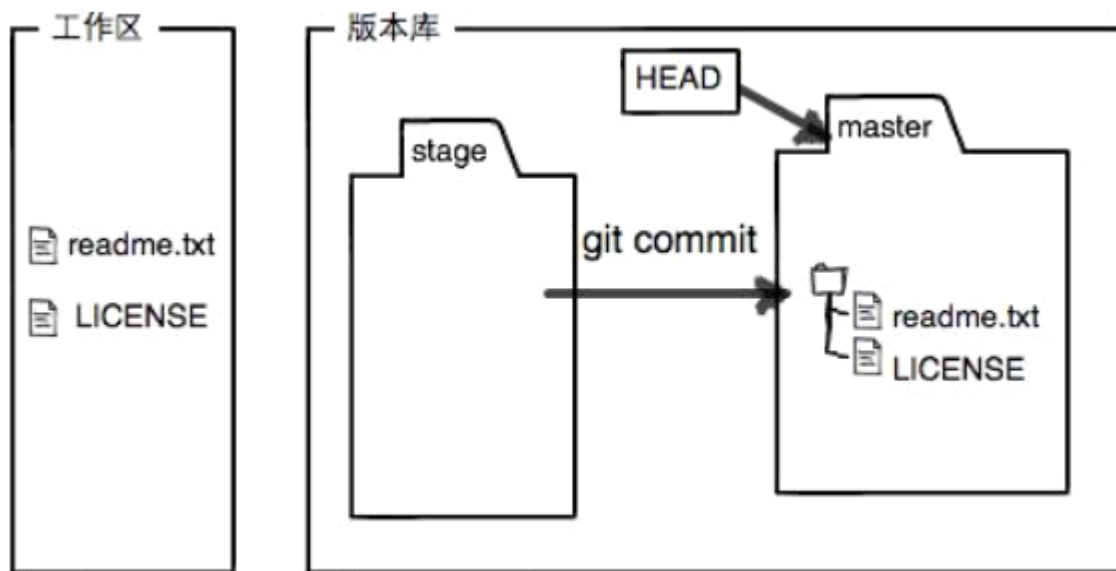


前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

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；  
(对readme.txt进行了修改 和 LICENSE进行了添加)



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。



用 `git diff HEAD -- readme.txt` 命令可以查看工作区和版本库里面最新版本的区别

# Git-撤销修改&删除

2018年4月30日 20:17

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。

当你使用 `rm filename`，在文件管理器中删除时；

- ❖ 删除有效时，需要在Git版本库中将提交的文件删除  
`git rm filename`
- ❖ 误删除时，使用 `git checkout -- filename`，回滚到上一个版本，即使用版本库里的版本替换工作区的版本。

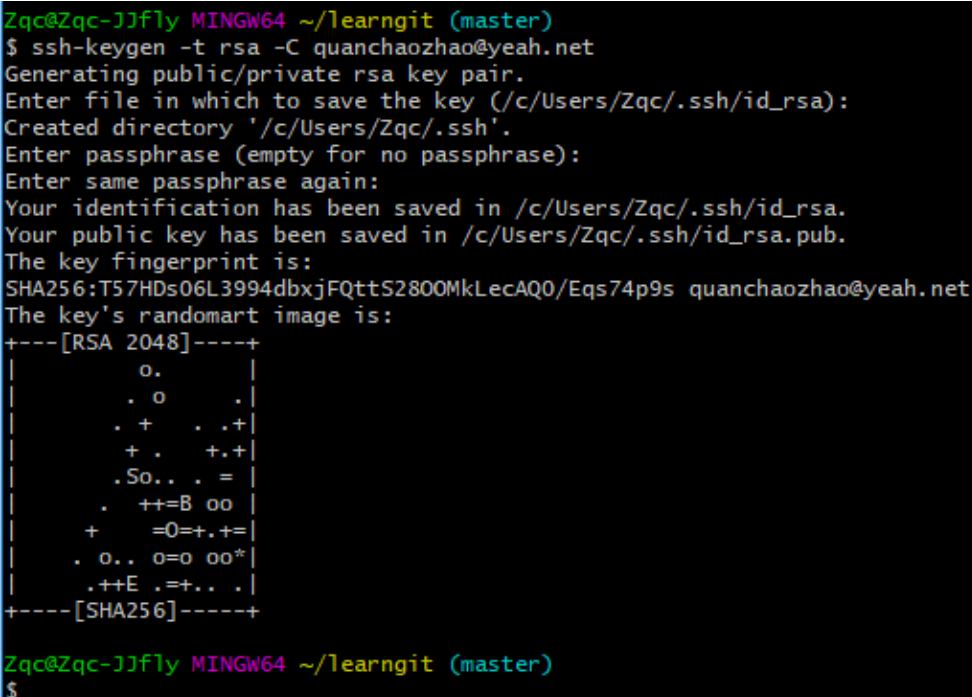
# Git-远程仓库

2018年4月30日 20:29

## ✓ 创建ssh公钥

第1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，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

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



```
Zqc@Zqc-JJfly MINGW64 ~/learngit (master)
$ ssh-keygen -t rsa -C quanchaozhao@yeah.net
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Zqc/.ssh/id_rsa):
Created directory '/c/Users/Zqc/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Zqc/.ssh/id_rsa.
Your public key has been saved in /c/Users/Zqc/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:T57HDs06L3994dbxjFQttS2800MkLecAQ0/Eqs74p9s quanchaozhao@yeah.net
The key's randomart image is:
+---[RSA 2048]---+
|           o.           |
|          . o .         |
|         . + . . +      |
|        + . . +. +     |
|       .So.. . =       |
|      . ++=B oo       |
|     +   =O=+.+=      |
|    . o.. o=o oo*     |
|   .++E .+=.. .      |
+-----[SHA256]-----+
Zqc@Zqc-JJfly MINGW64 ~/learngit (master)
$
```

第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容：

## ✓ 使用两种方法建立连接

```
git remote set-url origin https://github.com/zqclinux/kkk.git
```

(http连接需要账号密码)

```
git remote add origin git@github.com:zqclinux/learngit.git
```

(ssh 连接)

```
git push (-u第一次推送) origin master
```

git remote -v：查看所有的本地库

git remote rm (remove repository) Name :删除本地库

## ✓ 克隆远程库

使用 git clone git@github.com:zqclinux/gitskills.git

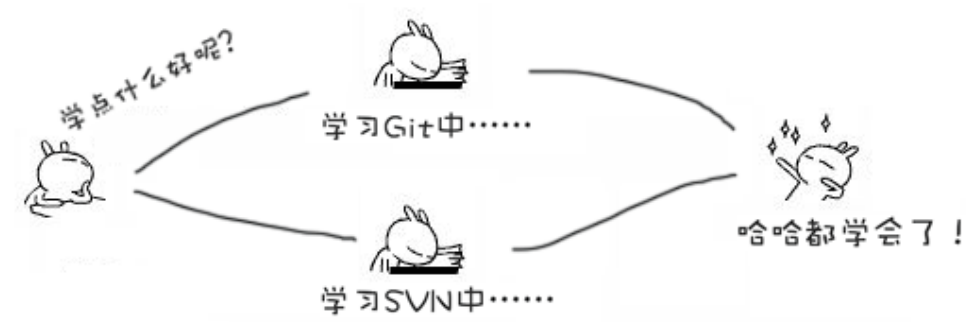
或

```
git clone https://github.com/zqclinux/python.git
```

# Git-分支管理

2018年5月1日 11:41

git分支管理通过改变指针完成，速度很快





# Git-创建合并分支

2018年5月1日 16:15

## 创建和合并分支

### 创建和合并命令

查看分支: `git branch`

创建分支: `git branch <name>`

切换分支: `git checkout <name>`

创建+切换分支: `git checkout -b <name>`

合并某分支到当前分支: `git merge <name>`

删除分支: `git branch -d <name>`

强制删除分支: `git branch -D <name>`

## 解决冲突

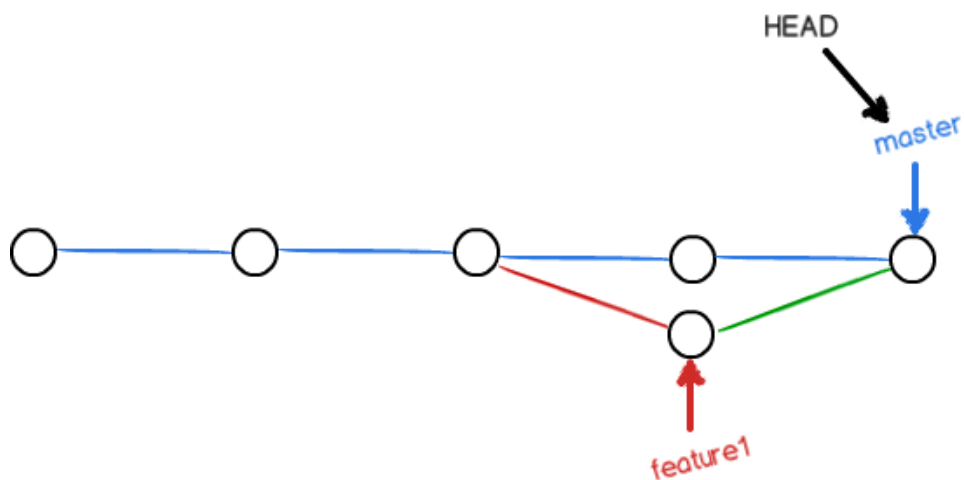
当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

用`git log --graph`命令可以看到分支合并图。

对于无法自动合并的分支，需要人工进行修改。不同的分支用=====分割

```
Zqc@Zqc-JJfly MINGW64 ~/learngit (master|MERGING)
$ cat readme.txt
Hello my first git project.
Add new line.
<<<<<<< HEAD
First change.
In the master.
=====
First change.
Second change.
In the feature.
>>>>>>> feature1
```

合并的后的分支线:



## 分支管理

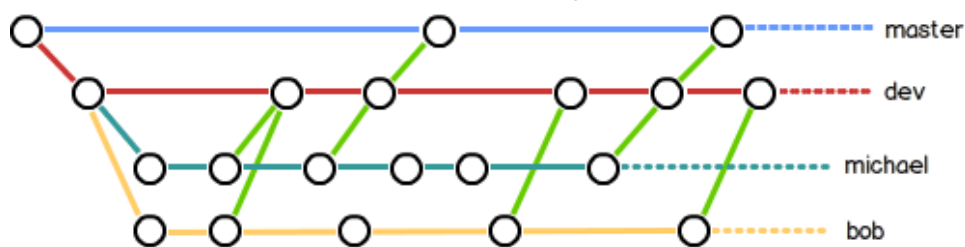
在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

Git分支十分强大，在团队开发中应该充分应用。

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。



# Git-bug修复

2018年5月1日 19:24

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；  
当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

# 多人协作

2018年5月1日 23:23

[多人协作](#)