Git简介

1、Git是目前世界上最先进的分布式版本控制系统（没有之一）

2、集中式和分布式版本控制系统有什么区别呢？

区别在于历史版本维护的位置：Git本地仓库包含代码库还有历史库，在本地的环境开发就可以记录历史；而SVN的历史库存在于中央仓库，每次对比与提交代码都必须连接到中央仓库才能进行。这样的好处在于：自己可以在脱机环境查看开发的版本历史；多人开发时如果充当中央仓库的Git仓库挂了，可以随时创建一个新的中央库然后同步就立刻恢复了中央库

安装Git

1、linux上安装Git

sudo apt-get install git

2、windows上安装Git

msysgit是Windows版的Git，从https://git-for-windows.github.io下载，然后按默认选项安装即可。

打开Git Bash既是

3、自报家门　　不报家门就不能推送到远程了

$ git config --global suer.name "pony"

$ git config --global user.email "pony@126.com"

创建版本库

1、创建方法：

创建一个版本库非常简单，首先，选择一个合适的地方，创建一个空目录（确保目录名不包含中文）

$ cd D:/doc/

$ mkdir learngit

$ cd learngit/

$ pwd

/d/doc/learngit

2、通过git init命令把这个目录变成Git可以管理的仓库

$ git init

Initialized empty Git repository in d:/doc/learngit/.git/

3、把文件添加到版本库中

※不要用windows默认编辑器，编码有问题，建议使用notepad++，而且默认编码设置为UTF-8 without BOM既可

任务：将readme.txt添加到Git

（1）readme.txt加到仓库（即放到项目的文件夹下 ）

（2） $ git add readme.txt

（3）$ git commit -m "wrote a redme　//-m是本次提交的说明

[master (root-commit) bd23402]

1 file changed, 2 insertions(

create mode 100644 readme.txt

4、为什么有add和commit？

因为可以add多个文件之后，再commit

时光机穿梭

查看状态

- 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。

zlm@ZLM-PC /d/Doc/learngit (master)

$ git status

On branch master

nothing to commit, working directory clean

zlm@ZLM-PC /d/Doc/learngit (master)

//修改了readme.txt后再次查看状态

$ git status

On branch master

Changes not staged for commit:

(use "git add ..." to update what will be committed)

(use "git checkout -- ..." to discard changes in working

modified:　readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

$ git diff readme.txt　　//查看到底修改了什么

WARNING: terminal is not fully functional

diff --git a/readme.txt b/readme.txt

index 084d42e..37bbe8d 100644

--- a/readme.txt

+++ b/readme.txt

@@ -1,2 +1,2 @@

-Git is a version control system

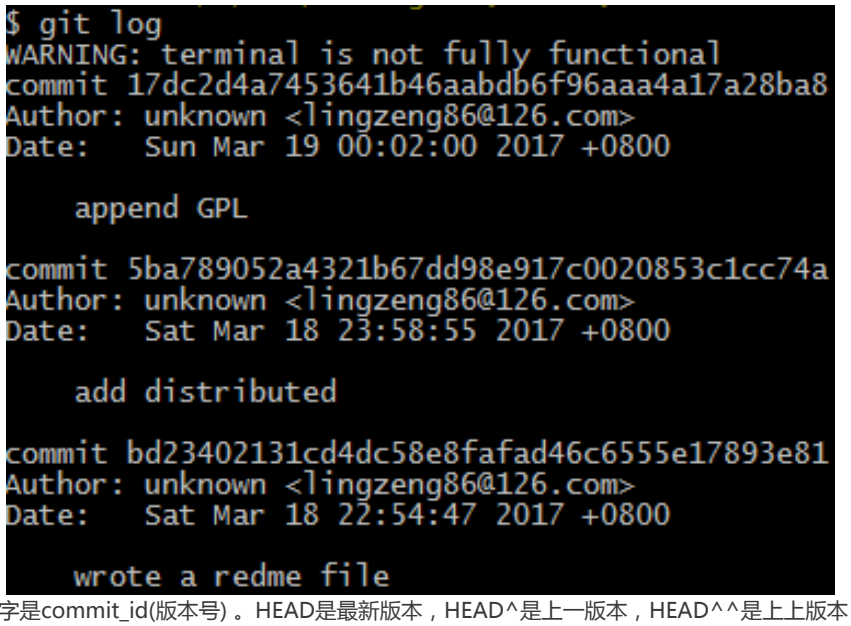
+Git is a distributed version control system

 Git is a free software

\ No newline at end of file

版本回退

- 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。



commit后面的一长串数字是commit_id(版本号) 。HEAD是最新版本，HEAD^是上一版本，HEAD^^是上上版本，HEAD~100是前100个版本

返回上一版本：

$ git reset --hard HEAD^

HEAD is now at 5ba7890 add distributed

返回某一个指定版本，只需要写版本号前几位就行，git会自动找到

$ git reset --hard bd2340

HEAD is now at bd23402 wrote a redme file

$ git reflog　　//用来记录每一次命令

WARNING: terminal is not fully functional

bd23402 HEAD@{0}: reset: moving to bd2340

5ba7890 HEAD@{1}: reset: moving to HEAD^

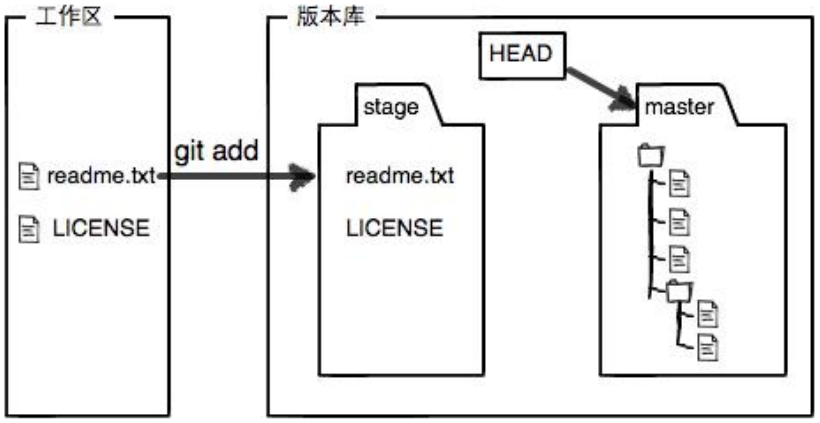
17dc2d4 HEAD@{2}: commit: append GPL

5ba7890 HEAD@{3}: commit: add distributed

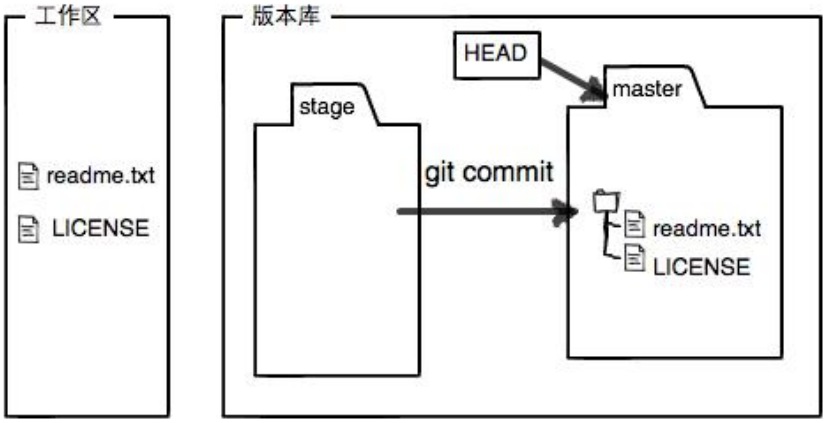
bd23402 HEAD@{4}: commit (initial): wrote a redme file

工作区和缓存区

git add之后：



git commit之后：



管理修改

每次修改，如果不add到暂存区，那就不会加入到commit中

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

撤销修改

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。

删除文件

从版本库恢复文件：　　不想要本地的修改了

```
$ git checkout -- test.txt
```

删除版本库中的文件：

```
$ git rm test.txtrm 'test.txt'$ git commit -m "remove test.txt"[master d17efd8] remove test.txt 1 file changed, 1 deletion(-) delete mode 100644 test.txt
```

远程仓库

GitHub

1、注册一个github账号（https://github.com/ ），就可以获得git远程仓库

C:\Users\Administrator\.ssh

2、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"lingzeng86@126.com"
```

注册github的邮箱地址

然后一路回车，使用默认值即可，由于这个Key也不是用于军事目的，所以也无需设置密码。

如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id_rsa和id_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id_rsa是私钥，不能泄露出去，id_rsa.pub是公钥，可以放心地告诉任何人。

3、登陆GitHub，打开“Account settings”，“SSH Keys”页面，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，就可以了。

添加远程库　　使得本地库的东西可以远程发送到网上去　　git remote -v 查看已关联的远程库

1、在GitHub添加一个仓库名字为“learngit”

2、本地库与GitHub做关联

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

后面这一句是要去github的工程通过ssh密钥连接取得地址

origin是固定用法，表示远程仓库

3、本地库所有内容推送到远程库

在第三步之前，必须有git pull origin master

```
fatal:refused to merge unrelated ...
```

出现这个问题的最主要原因还是在于本地仓库和远程仓库实际上是独立的两个仓库。假如我之前是直接clone的方式在本地建立起远程github仓库的克隆本地仓库就不会有这问题了。

或者是 git pull origin master --allow-unrelated-histories 强制允许融合独立的库

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

```
The authenticity of host 'github.com (192.30.253.113)' can't be established.

RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'github.com,192.30.253.113' (RSA) to the list of know

n hosts.

Counting objects: 20, done.
```

Delta compression using up to 4 threads.

Compressing objects: 100% (15/15), done.

Writing objects: 100% (20/20), 1.65 KiB | 0 bytes/s, done.

Total 20 (delta 4), reused 0 (delta 0)

remote: Resolving deltas: 100% (4/4), done.

To git@github.com:zenglingming/learngit.git

 * [new branch]   master -> master

Branch master set up to track remote branch master from origin.

把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

从现在起，只要本地作了提交，就可以通过命令：

```
$ git push origin master
```

把本地master分支的最新修改推送至GitHub，现在，你就拥有了真正的分布式版本库！

※上图中的SSH警告只会出现一次，是因为需要吧GitHub的Key添加到本机的一个信任列表里，以后不会再有。

从远程库克隆

1、在GitHub创建一个新仓库，名字叫gitskills，可以勾选上“Initialize this repository with a README”

2、git clone git@github.com:zenglingming/gitskills.git

```
zlm@ZLM-PC /d/doc
$ git clone git@github.com:zenglingming/gitskills.git
Cloning into 'gitskills'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
```

当然也可以用https://github.com/zenglingming/gitskills 的htpps协议也可以clone，但除了速度慢之外，每次推送还需要输入口令。而ssh支持的原生git协议是最快的。

分支管理

Git的创建分、切换和分支速度很快，1秒钟内就能完成！

创建与合并分支

http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/001375840038939c291467cc7c

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch

切换分支：git checkout

创建+切换分支：git checkout -b

```
git checkout -b dev
equals to git branch dev
          git checkout dev
```

切换到另一个分支dev修改工作区文件，并且commit提交上去了，如果切换回原来的分支master，那么文件并不是修改的，需要git merge dev，才有效果

合并某分支到当前分支：git merge

删除分支：git branch -d

解决冲突

http://www.liaoxuefeng.com/wiki/001373951630592960dd18361248578c67b8067c8c017b000/001375840202368c74be33fbd8

当branch和master都有新的提交，branch合并到master时，会有冲突，这就需要手动解决。

用带参数的git log也可以看到分支的合并情况：

```
$ git log --graph --pretty=oneline --abbrev-commit*   59bc1cb conflict fixed|\| * 75a857c AND simple* | 400b400 &
simple|/* fec145a branch test...
```

分支管理策略

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，而默认的fast forward合并就看不出来曾经做过合并。

1、创建并切换到dev分支：git checkout -b dev

2、修改readme.txt，提交

$ git add readme.txt

$ git **commit** -m "add merge"

3、切换回master：$ git checkout master

4、普通模式合并dev分支

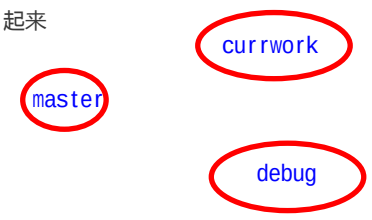
$ git merge --no-ff -m "merge with no-ff" dev

5、查看分支历史

$ git log --graph --pretty=oneline --abbrev-commit

Bug分支

1、在dev正常开发中，有一个bug要解决，先用stash功能吧当前工作现场“存储”起来



$ git stash

2、用git status查看工作区，就是干净的（除非有没有被Git管理的文件）

3、从master拉取bug分支，如：issue-101，修改之后，commit

master有bug的话 你必须从currwork（里面代码还有好多没写完，不提交先）先git stash存储工作内容，然后跳到master创建一个分支并且改bug提交上去，再跳回到master进行debug改过的代码合并
git merge --no-ff -m "merged bug fix 101" debug 并且删除debug分支
然后跳到currwork发现源代码不见了，需要git stash pop之前的工作代码才会回来

4、切换到master，合并分支issue-101，然后删除分支issue-101

5、切换到dev分支，查看保存的工作现场

```
$ git stash list
```

```
stash@{0}:WIP on dev:6224937 add merge
```

6、恢复工作现场

方法1：`git stash apply`恢复，但是恢复后，stash内容并不删除，你需要用`git stash drop`来删除

方法2：`git stash pop`，恢复的同时把stash内容也删了

※查看保存的工作现场，然后可以恢复指定的stash

```
$ git stash list$ git stash apply stash@{0}
```

null

FEATURE 分支

原则：创建新分支
和 bug分支合并一样，但如果提交以后老大不想要这个功能，要就地销毁，先跳到feature之前的那个分支，
`git branch -D <name>`