**Git学习笔记**

**安装Git**

    配置用户信息，git config命令的--global参数，用了这个参数，表示这台机器上的所有git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址

1. $ git config --global user.name 'Wendaojiao'
2. $ git config --global user.email '412113232@qq.com'

**创建版本库**

    初始化一个Git仓库，使用git init命令(windows系统，为避免遇到各种莫名其妙的问题，确保目录名包括父目录不包含中文)

    添加文件到Git仓库，分两步:

        第一步，使用命令git add <file>,注意可以多次使用，添加多个文件；

        第二步，使用命令git commit，完成。(git commit命令，-m后面输入的是本次提交的说明)

1. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code
2. $ mkdir learngit
3. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code
4. $ dir
5. ace gbk.txt index.php learngit python vue web yii yungou 原型图
6. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code
7. $ cd learngit/
8. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit
9. $ pwd
10. /E/code/learngit
11. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit
12. $ dir
13. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit
14. $ git init
15. Initialized empty Git repository in E:/code/learngit/.git/
16. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
17. $ dir
18. readme.txt
19. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
20. $ git add readme.txt
21. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
22. $ git commit -m 'wrote a readme file'
23. [master (root-commit) f3563bb] wrote a readme file
24. 1 file changed, 2 insertions(+)
25. create mode 100644 readme.txt
26. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)

**时光穿梭机**

    随时掌握工作区的状态，使用git status命令

    如果git status告诉有文件被修改过，用git diff可以查看修改内容

1. $ git status
2. On branch master
3. Changes not staged for commit:
4. (use "git add <file>..." to update what will be committed)
5. (use "git checkout -- <file>..." to discard changes in working directory)
6. modified: readme.txt
7. no changes added to commit (use "git add" and/or "git commit -a")
8. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
9. $ git diff readme.txt
10. diff --git a/readme.txt b/readme.txt
11. index 629f6b7..8afedca 100644
12. --- a/readme.txt
13. +++ b/readme.txt
14. @@ -1,2 +1,2 @@
15. -Git is a version control system
16. +Git is a distribute version control system
17. Git is free software
18. \ No newline at end of file
19. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
20. $ git add readme.txt
21. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
22. $ git status
23. On branch master
24. Changes to be committed:
25. (use "git reset HEAD <file>..." to unstage)
26. modified: readme.txt
27. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
28. $ git commit -m 'add distribute'
29. [master 491f31f] add distribute
30. 1 file changed, 1 insertion(+), 1 deletion(-)
31. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
32. $ git status
33. On branch master
34. nothing to commit, working tree clean

**版本回退**

        HEAD指向版本的当前版本，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id(HEAD表示当前版本，上一个版本是HEAD^，上上个版本是HEAD^^,若往上100个版本，可以 写HEAD~100)

        穿梭前，用git log可以查看提交历史，以便确定回退到哪个版本

        重返未来，用git reflog查看命令历史，以便确定回到未来的哪个版本

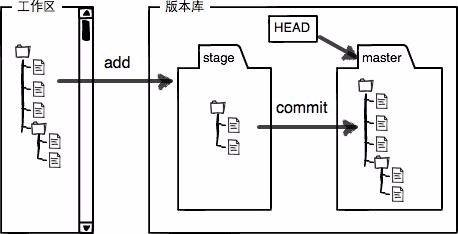
1. $ git log
2. commit df0f838df8960d71f8076e47d4ce265f4f041d55
3. Author: Wendaojiao <412113232@qq.com>
4. Date: Wed Dec 28 11:23:28 2016 +0800
5. append GPL
6. commit 491f31ffa7b2b7d4f483987e5ad6a1b3294bc356
7. Author: Wendaojiao <412113232@qq.com>
8. Date: Wed Dec 28 11:16:25 2016 +0800
9. add distribute
10. commit f3563bb79380af6c7812a361413d54e778bfe369
11. Author: Wendaojiao <412113232@qq.com>
12. Date: Wed Dec 28 11:01:11 2016 +0800
13. wrote a readme file
14. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
15. $ git log --pretty=oneline
16. df0f838df8960d71f8076e47d4ce265f4f041d55 append GPL
17. 491f31ffa7b2b7d4f483987e5ad6a1b3294bc356 add distribute
18. f3563bb79380af6c7812a361413d54e778bfe369 wrote a readme file
19. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
20. $ git reset --hard HEAD^
21. HEAD is now at 491f31f add distribute
22. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
23. $ cat readme.txt
24. Git is a distribute version control system
25. Git is free software
26. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
27. $ git log
28. commit 491f31ffa7b2b7d4f483987e5ad6a1b3294bc356
29. Author: Wendaojiao <412113232@qq.com>
30. Date: Wed Dec 28 11:16:25 2016 +0800
31. add distribute
32. commit f3563bb79380af6c7812a361413d54e778bfe369
33. Author: Wendaojiao <412113232@qq.com>
34. Date: Wed Dec 28 11:01:11 2016 +0800
35. wrote a readme file
36. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
37. $ git reset --hard df0f838df896
38. HEAD is now at df0f838 append GPL
39. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
40. $ cat readme.txt
41. Git is a distributed version control system
42. Git is free software distributed under the GPL.
43. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)
44. $ git reflog
45. df0f838 HEAD@{0}: reset: moving to df0f838df896
46. 491f31f HEAD@{1}: reset: moving to HEAD^
47. df0f838 HEAD@{2}: commit: append GPL
48. 491f31f HEAD@{3}: commit: add distribute
49. f3563bb HEAD@{4}: commit (initial): wrote a readme file
50. Administrator@CFCS5MVGYNS3SAX MINGW64 /E/code/learngit (master)

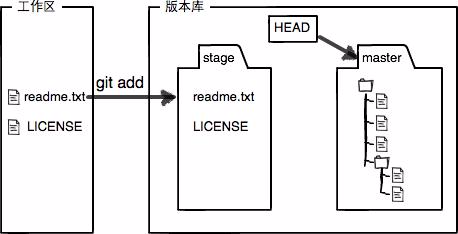
**工作区和暂存区**

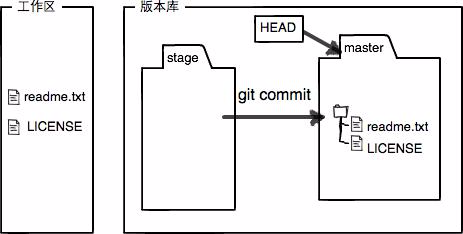
        把文件往Git版本库里添加的时候是分两步执行的：

        第一步是用git add把文件添加进去，实际上是吧文件修改添加到暂存区；

        第二步是用git commit提交修改，实际上是吧暂存区的所有内容提交到当前分支







**管理修改**

        操作： 第一次修改->git add ->第二次修改 ->gie commit

        git add命令后，工作区的第一次修改被放入暂缓区，但工作区的第二次修改并没有放入暂存区，所以，git commit只负责把暂存区的修改提交了，也就是第一次提交，第二次没有提交。要提交第二次，可以继续git add再git commit

**撤销修改**

        命令git checkout -- readme.txt意思是把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

            一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本一模一样的状态

            一种是readme.txt已经添加到暂存区，又做了修改，现在撤销修改回到添加到暂存区后的状态

            总之，就是让文件回到最近一次git commit或git add时的状态。

        git checkout -- file 命令中的--很重要，没有--，就变成了‘切换到另一个分支’的命令。

        git reset HEAD file可以把暂存区的修改撤销掉，重新放回工作区

        git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。

        场景1：当你改乱了工作区的某个文件的内容，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的某个文件内容，还添加到了暂存区，想丢弃修改，分两步，第一步用命令git reset HEAD file,回到场景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，想要撤销本次修改，参考版本回退

**删除文件**

        若本地删除了某个文件：

            若要确实从版本库中删除该文件，则用命令git rm <file>删除，并且git commit

            若是删错了，可以用命令git checkout -- <file>把误删文件恢复到最新版本

**远程仓库**

**添加远程库**

        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，每次提交后，只要有必要，都可以用命令git push origin master推送最新修改；

**从远程库克隆**

        克隆一个仓库，先知道仓库的地址，然后使用git clone命令克隆。git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。例：$ git clone git@github.com:michaelliao/gitskills.git

**分支管理**

**创建与合并分支**

        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>

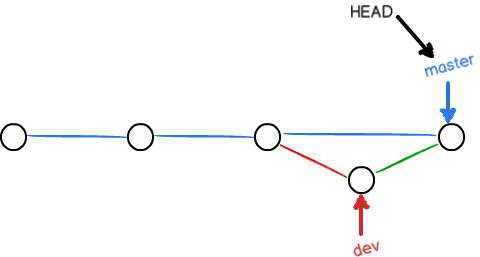
**解决冲突**

        当Git无法自动合并分支时，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。

**分支管理策略**

        通常合并分支，Git会用Fast forward模式，但这种模式，删除分支后，会丢掉分支信息。如果要强制禁用Fast forward模式，Git会在merge是生成一个新的commit,这样在分支历史上可以看出分支信息。

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

**Bug分支**

        修复bug

            1,dev分支上开发工作先把当前dev分支的工作现场存储起来。git stash

            2,在修复bug的分支上创建临时分支，假定在master上修复bug

                git checkout master    #切换到master分支

                git checkout -b issue-101    #创建bug修复临时分支并切换

                git add readme.txt

                git commit -m 'fix bug 101'

                git checkoout master    #切换到master分支

                git merger --no -ff -m 'merged bug fix 101' issue-101    #master分支上合并bug修复临时分支

                git checkout dev    #切换到开发分支

                git stash list    #查看开发分支的工作现在

                一是用 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；

                二是用git stash pop 恢复的同时把stash内容也删了；

**Featur分支**

        开发一个新feature，最好新建一个分支；

        如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除

**多人协作**

        master分支是主分支，因此要时刻与远程同步；

        dev分支是开发分支，团队所有成员在上面工作，所以也需要与远程同步；

        bug分支只用于本地修改bug，

        feature分支是否推送到远程，取决于你是否和你的小伙伴合作在上面开发；

         多人协作工作模式：

        1，首先，试图用git push origin branch-name推送自己的修改

        2，如果推送失败，则因为远程分支比本地更新，需要先用git pull 试图合并；

        3，如果合并有冲突，则解决冲突，并在本地提交；

        4，没有冲突或解决冲突后，再用git push origin branch-name推送到远程

        如果git pull 提示‘no tracking information’，则说明本地分支和远程分支的链接关系没有创建，用命令

        git branch --set-upstream branch-name origin/branch-name

        查看远程库信息使用 git remote -v;

        本地新建分支如果不推送到远程，对其他人是不可见；

        本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程新的提交；

        在本地分支和远程分支对应的分支。使用git checkout -b branch-name origin/branch-name,本地和远程的分支名称最后一致；

        建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name

        从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

**标签管理**

    Git的标签是版本库的快照，其实它是指向某个commit的指针(跟分支很像，分支可以移动，标签不能移动)

**创建标签**

         git tag 查看所有标签

         git show <tagname>查看标签信息

         git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id

         git tag -a <tagname> -m 'blablabla...'可以指定标签信息 -a 指定标签名 -m 指定说明文字；

         git tag -s <tagname> -m 'blablabla...'可以用PGP签名标签；

**操作标签**

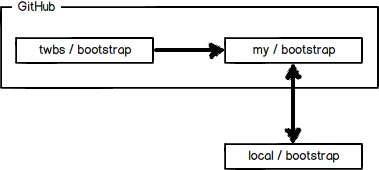
        命令git push origin <tagname>可以推送一个本地标签

        命令git push origin --tags 可以推送全部未推送过的本地标签

        命令git tag -d <tagname>可以删除一个本地标签

        命令git push origin :refs/tags/<tagname>可以删除一个远程标签

**使用GitHub**



**自定义Git**

    让Git显示颜色 git config --global color.ui true

**忽略特殊文件**

        git add -f App.class 可以用-f强制添加到Git

        git check-ignore -v App.class 可以用git check-ignore命令检查.gitignore是否写得有正确

        忽略某些文件时，需要编写.gitignore

        .gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理

**配置别名**

        --global参数是全局参数，针对当前用户，如不加，只针对当前的仓库起作用

        $ git config --global alias.co checkout

        $ git config --global alias.ci commit

        $ git config --global alias.br branch

**搭建Git服务器**

        第一步，安装git

            $ sudo apt-get install git

     第二步，创建一个git用户，用来运行git服务

       $ sudo adduser git

     第三步，创建证书登录

            收集所有需要登录的用户的公钥，也就是他们自己的id\_rsa.pub文件，把所有公钥导入到/home/git/.ssh/authorized\_keys

        第四步，初始化Git仓库

            先选定一个目录作为Git仓库，假定是/srv/sample.git,在/srv目录下输入命令

            sudo git init --bare sample.git

        sudo chown -R git:git sample.git

        第五步，禁用shell登录

            git用户可以正常通过ssh使用git，但无法登录shell

            git:x:1001:1001:,,,:/home/git:/bin/bash

        改为

        git:x:1001:1001:,,,:/home/git:/usr/bin/git-shell

         第六步，克隆远程仓库

              git clone git@server:/srv/sample.git