目 录

初始化配置 2

查看、添加、提交、删除、找回，重置修改文件 2

查看文件diff 2

查看提交记录 3

取得Git仓库 3

提交你的修改 3

查看、切换、创建和删除分支 4

分支合并和rebase 4

Git补丁管理(方便在多台机器上开发同步时用) 4

Git暂存管理 4

Git远程分支管理 4

基本的分支管理 5

Git远程仓库管理 5

创建远程仓库 5

**初始化配置**

1. #配置使用git仓库的人员姓名
2. git config --global user.name "Your Name Comes Here"
3. #配置使用git仓库的人员email
4. git config --global user.email you@yourdomain.example.com
5. #配置到缓存 默认15分钟
6. git config --global credential.helper cache
7. #修改缓存时间
8. git config --global credential.helper 'cache --timeout=3600'
9. git config --global color.ui **true**
10. git config --global alias.co checkout
11. git config --global alias.ci commit
12. git config --global alias.st status
13. git config --global alias.br branch
14. git config --global core.editor "mate -w"    # 设置Editor使用textmate
15. git config -1 #列举所有配置
16. #用户的git配置文件~/.gitconfig

## ****查看、添加、提交、删除、找回，重置修改文件****

1. git help <command>  # 显示command的help
2. git show            # 显示某次提交的内容
3. git show $id
4. git co  -- <file>   # 抛弃工作区修改
5. git co  .           # 抛弃工作区修改
6. git add <file>      # 将工作文件修改提交到本地暂存区
7. git add .           # 将所有修改过的工作文件提交暂存区
8. git rm <file>       # 从版本库中删除文件
9. git rm <file> --cached  # 从版本库中删除文件，但不删除文件
10. git reset <file>    # 从暂存区恢复到工作文件
11. git reset -- .      # 从暂存区恢复到工作文件
12. git reset --hard    # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改
13. git ci <file>
14. git ci .
15. git ci -a           # 将git add, git rm和git ci等操作都合并在一起做
16. git ci -am "some comments"
17. git ci --amend      # 修改最后一次提交记录
18. git revert <$id>    # 恢复某次提交的状态，恢复动作本身也创建了一次提交对象
19. git revert HEAD     # 恢复最后一次提交的状态

## ****查看文件diff****

1. git diff <file>     # 比较当前文件和暂存区文件差异
2. git diff
3. git diff <$id1> <$id2>   # 比较两次提交之间的差异
4. git diff <branch1>..<branch2> # 在两个分支之间比较
5. git diff --staged   # 比较暂存区和版本库差异
6. git diff --cached   # 比较暂存区和版本库差异
7. git diff --stat     # 仅仅比较统计信息

## ****查看提交记录****

1. git log
2. git log <file>      # 查看该文件每次提交记录
3. git log -p <file>   # 查看每次详细修改内容的diff
4. git log -p -2       # 查看最近两次详细修改内容的diff
5. git log --stat      #查看提交统计信息

**tig**

Mac上可以使用tig代替diff和log，brew install tig

## 取得Git仓库

1. #初始化一个版本仓库
2. git init
3. #Clone远程版本库
4. git clone git@xbc.me:wordpress.git
5. #添加远程版本库origin，语法为 git remote add [shortname] [url]
6. git remote add origin git@xbc.me:wordpress.git
7. #查看远程仓库
8. git remote -v

## 提交你的修改

1. #添加当前修改的文件到暂存区
2. git add .
3. #如果你自动追踪文件，包括你已经手动删除的，状态为Deleted的文件
4. git add -u
5. #提交你的修改
6. git commit –m "你的注释"
7. #推送你的更新到远程服务器,语法为 git push [远程名] [本地分支]:[远程分支]
8. git push origin master
9. #查看文件状态
10. git status
11. #跟踪新文件
12. git add readme.txt
13. #从当前跟踪列表移除文件，并完全删除
14. git rm readme.txt
15. #仅在暂存区删除，保留文件在当前目录，不再跟踪
16. git rm –cached readme.txt
17. #重命名文件
18. git mv reademe.txt readme
19. #查看提交的历史记录
20. git log
21. #修改最后一次提交注释的，利用–amend参数
22. git commit --amend
23. #忘记提交某些修改，下面的三条命令只会得到一个提交。
24. git commit –m &quot;add readme.txt&quot;
25. git add readme\_forgotten
26. git commit –amend
27. #假设你已经使用git add .，将修改过的文件a、b加到暂存区
28. #现在你只想提交a文件，不想提交b文件，应该这样
29. git reset HEAD b
30. #取消对文件的修改
31. git checkout –- readme.txt

## ****查看、切换、创建和删除分支****

1. git br -r           # 查看远程分支
2. git br <new\_branch> # 创建新的分支
3. git br -v           # 查看各个分支最后提交信息
4. git br --merged     # 查看已经被合并到当前分支的分支
5. git br --no-merged  # 查看尚未被合并到当前分支的分支
6. git co <branch>     # 切换到某个分支
7. git co -b <new\_branch> # 创建新的分支，并且切换过去
8. git co -b <new\_branch> <branch>  # 基于branch创建新的new\_branch
9. git co $id          # 把某次历史提交记录checkout出来，但无分支信息，切换到其他分支会自动删除
10. git co $id -b <new\_branch>  # 把某次历史提交记录checkout出来，创建成一个分支
11. git br -d <branch>  # 删除某个分支
12. git br -D <branch>  # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)

## ****分支合并和rebase****

1. git merge <branch>               # 将branch分支合并到当前分支
2. git merge origin/master --no-ff  # 不要Fast-Foward合并，这样可以生成merge提交
3. git rebase master <branch>       # 将master rebase到branch，相当于：
4. git co <branch> && git rebase master && git co master && git merge <branch>

## ****Git补丁管理(方便在多台机器上开发同步时用)****

1. git diff > ../sync.patch         # 生成补丁
2. git apply ../sync.patch          # 打补丁
3. git apply --check ../sync.patch  #测试补丁能否成功

## ****Git暂存管理****

1. git stash                        # 暂存
2. git stash list                   # 列所有stash
3. git stash apply                  # 恢复暂存的内容
4. git stash drop                   # 删除暂存区

## ****Git远程分支管理****

1. git pull                         # 抓取远程仓库所有分支更新并合并到本地
2. git pull --no-ff                 # 抓取远程仓库所有分支更新并合并到本地，不要快进合并
3. git fetch origin                 # 抓取远程仓库更新
4. git merge origin/master          # 将远程主分支合并到本地当前分支
5. git co --track origin/branch     # 跟踪某个远程分支创建相应的本地分支
6. git co -b <local\_branch> origin/<remote\_branch>  # 基于远程分支创建本地分支，功能同上
7. git push                         # push所有分支
8. git push origin master           # 将本地主分支推到远程主分支
9. git push -u origin master        # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)
10. git push origin <local\_branch>   # 创建远程分支， origin是远程仓库名
11. git push origin <local\_branch>:<remote\_branch>  # 创建远程分支
12. git push origin :<remote\_branch>  #先删除本地分支(git br -d <branch>)，然后再push删除远程分支

## 基本的分支管理

1. #创建一个分支
2. git branch dev
3. #切换工作目录到dev
4. git chekcout dev
5. #将上面的命令合在一起，创建dev分支并切换到dev
6. git chekcout –b dev
7. #合并dev分支，当前工作目录为master
8. git merge dev
9. #合并完成后，没有出现冲突，删除dev分支
10. git branch –d dev
11. #拉去远程仓库的数据，语法为 git fetch [remote-name]
12. git fetch
13. #fetch 会拉去最新的远程仓库数据，但不会自动到当前目录下，要自动合并
14. git pull
15. #查看远程仓库的信息
16. git remote show origin
17. #建立本地的dev分支追踪远程仓库的develop分支
18. git checkout –b dev origin/develop

## ****Git远程仓库管理****

1. git remote -v                    # 查看远程服务器地址和仓库名称
2. git remote show origin           # 查看远程服务器仓库状态
3. git remote add origin git@ github:robbin/robbin\_site.git         # 添加远程仓库地址
4. git remote set-url origin git@ github.com:robbin/robbin\_site.git # 设置远程仓库地址(用于修改远程仓库地址)
5. git remote rm <repository>       # 删除远程仓库

## ****创建远程仓库****

1. git clone --bare robbin\_site robbin\_site.git  # 用带版本的项目创建纯版本仓库
2. scp -r my\_project.git git@ git.csdn.net:~      # 将纯仓库上传到服务器上
3. mkdir robbin\_site.git && cd robbin\_site.git && git --bare init # 在服务器创建纯仓库
4. git remote add origin git@ github.com:robbin/robbin\_site.git    # 设置远程仓库地址
5. git push -u origin master                                      # 客户端首次提交
6. git push -u origin develop  # 首次将本地develop分支提交到远程develop分支，并且track
7. git remote set-head origin master   # 设置远程仓库的HEAD指向master分支

也可以命令设置跟踪远程库和本地库

1. git branch --set-upstream master origin/master
2. git branch --set-upstream develop origin/develop