Git的奇技淫巧🕸

Git常用命令集合, Fork于tips项目

Git是一个"分布式版本管理工具",简单的理解版本管理工具:大家在写东西的时候都用过"回撤"这个功能,但是回撤只能回撤几步,假如想要找回我三天之前的修改,光用"回撤"是找不回来的。而"版本管理工具"能记录每次的修改,只要提交到版本仓库,你就可以找到之前任何时刻的状态(文本状态)。

下面的内容就是列举了常用的 Git 命令和一些小技巧,可以通过 "页面内查找" 的方式进行快速查询: Ctr1/Command+f。

开卷必读

如果之前未使用过Git,可以学习Git 小白教程入门

- 1. 一定要先测试命令的效果后,再用于工作环境中,以防造成不能弥补的后果! 到时候别拿着砍刀来找我
- 2. 所有的命令都在 git version 2.7.4 (Apple Git-66) 下测试通过
- 3. 统一概念:
 - o 工作区: 改动(增删文件和内容)
 - o 暂存区:输入命令: git add 改动的文件名,此次改动就放到了'暂存区'
 - o 本地仓库(简称:本地):输入命令: git commit 此次修改的描述,此次改动就放到了'本地仓库',每个commit,我叫它为一个'版本'。
 - o 远程仓库(简称:远程):输入命令: git push 远程仓库,此次改动就放到了'远程仓库'(GitHub等)
 - o commit-id: 输出命令: git log, 最上面那行 commit xxxxxx, 后面的字符串就是 commit-id
- 4. 如果喜欢这个项目,欢迎 Star、提交 Pr、反馈问题 ②

目录

- 展示帮助信息
- 回到远程仓库的状态
- 重设第一个commit
- 展示工作区和暂存区的不同
- 展示暂存区和最近版本的不同
- 展示暂存区、工作区和最近版本的不同
- 快速切换分支
- 删除已经合并到 master 的分支
- 展示本地分支关联远程仓库的情况
- 关联远程分支
- 列出所有远程分支
- 列出本地和远程分支
- 创建并切换到本地分支
- 从远程分支中创建并切换到本地分支
- 删除本地分支
- 删除远程分支
- 重命名本地分支

- 查看标签
- 查看标签详细信息
- 本地创建标签
- 推送标签到远程仓库
- 删除本地标签
- 删除远程标签
- 切回到某个标签
- 放弃工作区的修改
- 恢复删除的文件
- 以新增一个 commit 的方式还原某一个 commit 的修改
- 回到某个 commit 的状态,并删除后面的 commit
- 修改上一个 commit 的描述
- 查看 commit 历史
- 显示本地更新过 HEAD 的 git 命令记录
- 修改作者名
- 修改远程仓库的 url
- 增加远程仓库
- 列出所有远程仓库
- 查看两个星期内的改动
- 把 A 分支的某一个 commit, 放到 B 分支上
- 给 git 命令起别名
- 存储当前的修改,但不用提交 commit
- 保存当前状态,包括 untracked 的文件
- 展示所有 stashes
- 回到某个 stash 的状态
- <u>回到最后一个 stash 的状态,并删除这个 stash</u>
- 删除所有的 stash
- 从 stash 中拿出某个文件的修改
- 展示所有 tracked 的文件
- 展示所有 untracked 的文件
- 展示所有忽略的文件
- 强制删除 untracked 的文件
- 强制删除 untracked 的目录
- 展示简化的 commit 历史
- 查看某段代码是谁写的
- 把某一个分支到导出成一个文件
- 从包中导入分支
- 执行 rebase 之前自动 stash
- 从远程仓库根据 ID, 拉下某一状态, 到本地分支
- 详细展示一行中的修改
- <u>清除 _gitignore</u> 文件中记录的文件
- 展示所有 alias 和 configs
- 展示忽略的文件
- commit 历史中显示 Branch1 有的,但是 Branch2 没有 commit
- <u>在 commit log 中显示 GPG 签名</u>
- 删除全局设置
- 新建并切换到新分支上,同时这个分支没有任何 commit
- 展示任意分支某一文件的内容

- clone 下来指定的单一分支
- 忽略某个文件的改动
- 忽略文件的权限变化
- 以最后提交的顺序列出所有 Git 分支
- 在 commit log 中查找相关内容
- 把暂存区的指定 file 放到工作区中
- 强制推送
- 一图详解
- 联系我

展示帮助信息

```
git help -g
```

The command output as below:

The common Git guides are:

attributes Defining attributes per path

cli Git command-line interface and conventions

core-tutorial A Git core tutorial for developers

cvs-migration Git for CVS users diffcore Tweaking diff output

everyday A useful minimum set of commands for Everyday Git

glossary A Git Glossary hooks Hooks used by Git

ignore Specifies intentionally untracked files to ignore

modules Defining submodule properties

namespaces Git namespaces

repository-layout Git Repository Layout

revisions Specifying revisions and ranges for Git

tutorial A tutorial introduction to Git

tutorial-2 A tutorial introduction to Git: part two
workflows An overview of recommended workflows with Git

'git help -a' and 'git help -g' list available subcommands and some concept guides. See 'git help <command>' or 'git help <concept>' to read about a specific subcommand or concept.

回到远程仓库的状态

抛弃本地所有的修改, 回到远程仓库的状态。

```
git fetch --all && git reset --hard origin/master
```

重设第一个 commit

也就是把所有的改动都重新放回工作区,并清空所有的 commit,这样就可以重新提交第一个 commit 了

```
git update-ref -d HEAD
```

展示工作区和暂存区的不同

输出工作区和暂存区的 different (不同)。

```
git diff
```

还可以展示本地仓库中任意两个 commit 之间的文件变动:

```
git diff <commit-id> <commit-id>
```

展示暂存区和最近版本的不同

输出暂存区和本地最近的版本 (commit) 的 different (不同)。

```
git diff --cached
```

展示暂存区、工作区和最近版本的不同

输出工作区、暂存区和本地最近的版本 (commit)的 different (不同)。

```
git diff HEAD
```

快速切换分支上一个分支

```
git checkout -
```

删除已经合并到 master 的分支

```
git branch --merged master | grep -v '^* | master' | xargs -n 1 git branch -d
```

展示本地分支关联远程仓库的情况

```
git branch -vv
```

关联远程分支

关联之后,[git branch -vv] 就可以展示关联的远程分支名了,同时推送到远程仓库直接: [git push],不需要指定远程仓库了。

```
git branch -u origin/mybranch
```

```
git push origin/mybranch -u
```

列出所有远程分支

-r 参数相当于: remote

```
git branch -r
```

列出本地和远程分支

-a 参数相当于: all

```
git branch -a
```

创建并切换到本地分支

```
git checkout -b <br/>branch-name>
```

从远程分支中创建并切换到本地分支

```
git checkout -b <branch-name> origin/<branch-name>
```

删除本地分支

```
git branch -d <local-branchname>
```

删除远程分支

```
git push origin --delete <remote-branchname>
```

或者

```
git push origin :<remote-branchname>
```

重命名本地分支

```
git branch -m <new-branch-name>
```

查看标签

git tag

展示当前分支的最近的 tag

```
git describe --tags --abbrev=0
```

查看标签详细信息

git tag -ln

本地创建标签

git tag <version-number>

默认 tag 是打在最近的一次 commit 上,如果需要指定 commit 打 tag:

```
$ git tag -a <version-number> -m "v1.0 发布(描述)" <commit-id>
```

推送标签到远程仓库

首先要保证本地创建好了标签才可以推送标签到远程仓库:

git push origin <local-version-number>

一次性推送所有标签,同步到远程仓库:

git push origin --tags

删除本地标签

git tag -d <tag-name>

删除远程标签

删除远程标签需要先删除本地标签,再执行下面的命令:

git push origin :refs/tags/<tag-name>

切回到某个标签

一般上线之前都会打 tag, 就是为了防止上线后出现问题,方便快速回退到上一版本。下面的命令是回到某一标签下的状态:

```
git checkout -b branch_name tag_name
```

放弃工作区的修改

git checkout <file-name>

放弃所有修改:

git checkout .

恢复删除的文件

```
git rev-list -n 1 HEAD -- <file_path> #得到 deleting_commit
```

git checkout <deleting_commit>^ -- <file_path> #回到删除文件 deleting_commit 之前的状态

以新增一个 commit 的方式还原某一个 commit 的修改

git revert <commit-id>

回到某个 commit 的状态,并删除后面的 commit

和 revert 的区别: reset 命令会抹去某个 commit id 之后的所有 commit

git reset <commit-id> #默认就是-mixed参数。

git reset -mixed HEAD^ #回退至上个版本,它将重置HEAD到另外一个commit,并且重置暂存区以便和HEAD相匹配,但是也到此为止。工作区不会被更改。

git reset -soft HEAD~3 #回退至三个版本之前,只回退了commit的信息,暂存区和工作区与回退之前保持一致。如果还要提交,直接commit即可

git reset -hard <commit-id> #彻底回退到指定commit-id的状态,暂存区和工作区也会变为指定commit-id版本的内容

修改上一个 commit 的描述

git commit --amend

查看 commit 历史

git log

查看某段代码是谁写的

blame 的意思为'责怪', 你懂的。

git blame <file-name>

显示本地更新过 HEAD 的 git 命令记录

每次更新了 HEAD 的 git 命令比如 commint、amend、cherry-pick、reset、revert 等都会被记录下来(不限分支),就像 shell 的 history 一样。 这样你可以 reset 到任何一次更新了 HEAD 的操作之后,而不仅仅是回到当前分支下的某个 commit 之后的状态。

git reflog

修改作者名

```
git commit --amend --author='Author Name <email@address.com>'
```

修改远程仓库的 url

git remote set-url origin <URL>

增加远程仓库

git remote add origin <remote-url>

列出所有远程仓库

git remote

查看两个星期内的改动

git whatchanged --since='2 weeks ago'

把 A 分支的某一个 commit, 放到 B 分支上

这个过程需要 cherry-pick 命令,参考

git checkout <branch-name> && git cherry-pick <commit-id>

给 git 命令起别名

```
git config --global alias.<handle> <command>
比如: git status 改成 git st, 这样可以简化命令
git config --global alias.st status
```

存储当前的修改,但不用提交 commit

详解可以参考廖雪峰老师的 git 教程

git stash

保存当前状态,包括 untracked 的文件

untracked 文件: 新建的文件

git stash -u

展示所有 stashes

git stash list

回到某个 stash 的状态

git stash apply <stash@{n}>

回到最后一个 stash 的状态,并删除这个 stash

git stash pop

删除所有的 stash

git stash clear

从 stash 中拿出某个文件的修改

git checkout <stash@{n}> -- <file-path>

展示所有 tracked 的文件

```
git 1s-files -t
```

展示所有 untracked 的文件

```
git 1s-files --others
```

展示所有忽略的文件

```
git ls-files --others -i --exclude-standard
```

强制删除 untracked 的文件

可以用来删除新建的文件。如果不指定文件文件名,则清空所有工作的 untracked 文件。 clean 命令,注意两点:

- 1. clean 后,删除的文件无法找回
- 2. 不会影响 tracked 的文件的改动,只会删除 untracked 的文件

```
git clean <file-name> -f
```

强制删除 untracked 的目录

可以用来删除新建的目录,注意:这个命令也可以用来删除 untracked 的文件。详情见上一条

```
git clean <directory-name> -df
```

展示简化的 commit 历史

```
git log --pretty=oneline --graph --decorate --all
```

把某一个分支到导出成一个文件

git bundle create <file> <branch-name>

从包中导入分支

新建一个分支,分支内容就是上面 git bundle create 命令导出的内容

```
git clone repo.bundle <repo-dir> -b <br/> <br/>branch-name>
```

执行 rebase 之前自动 stash

```
git rebase --autostash
```

从远程仓库根据 ID, 拉下某一状态, 到本地分支

git fetch origin pull/<id>/head:<branch-name>

详细展示一行中的修改

```
git diff --word-diff
```

清除 gitignore 文件中记录的文件

```
git clean -x -f
```

展示所有 alias 和 configs

注意: config 分为: 当前目录(local)和全局(golbal)的 config, 默认为当前目录的 config

```
git config --local --list (当前目录)
git config --global --list (全局)
```

展示忽略的文件

```
git status --ignored
```

commit 历史中显示 Branch1 有的,但是 Branch2 没有commit

git log Branch1 \Branch2

在 commit log 中显示 GPG 签名

```
git log --show-signature
```

删除全局设置

```
git config --global --unset <entry-name>
```

新建并切换到新分支上,同时这个分支没有任何 commit

相当于保存修改,但是重写 commit 历史

```
git checkout --orphan <br/> <br/> tranch-name>
```

展示任意分支某一文件的内容

git show <branch-name>:<file-name>

clone 下来指定的单一分支

git clone -b

branch-name> --single-branch https://github.com/user/repo.git

忽略某个文件的改动

关闭 track 指定文件的改动,也就是 Git 将不会在记录这个文件的改动

```
git update-index --assume-unchanged path/to/file
```

恢复 track 指定文件的改动

git update-index --no-assume-unchanged path/to/file

忽略文件的权限变化

不再将文件的权限变化视作改动

git config core.fileMode false

以最后提交的顺序列出所有 Git 分支

最新的放在最上面

```
git for-each-ref --sort=-committerdate --format='%(refname:short)' refs/heads/
```

在 commit log 中查找相关内容

通过 grep 查找, given-text: 所需要查找的字段

```
git log --all --grep='<given-text>'
```

把暂存区的指定 file 放到工作区中

不添加参数,默认是 -mixed

```
git reset <file-name>
```

强制推送

```
git push -f <remote-name> <branch-name>
```

一图详解

```
git config --global user. name "zahuifan"
                                                         git config --global user.email "zahuifan@163.com"
                                                         git config --global color.ui "always"
                                                         ● 将EmEditor 设置为Git 的解稿器

② 构设 EmEditor 设置为Git 的解稿器
② 始改 PATH 系统环境变量,参加FA工具模文档工具EmEditor
                                           解决 bash 控制台 Is 命令中文文件名乱码 在 Git\etc\git-completion bash 文件中追加一行 alias Is='Is --show-control-chars --color=auto'
                                                         解決 bash 控制台 git commit 无法输入中文注释 修改 GitletClinputr文件 set output-meta on set convert-meta off 解決中文乱刑问题 解決 gt log 命令中文注释乱码(只 bash 控制俗解用) 在 GitletClprofile 中道加一行: export LESSCHARSET=iso8859
                                                                      解表 gitk 显示中文注释乱码 git config --global i18n.commitencoding ISO-8859 git config --global i18n.logoutputencoding ISO-8859
                                 初始化
                                           初始化新版本库 git init 只会在根目录下创建一个名为。git 文件类,非常清爽!
                                           ① 在明目录部建一个名为,glignore 的文本文件 在命令行执行 echo *jpg-glignore 注意 > 号左右不要有空格
设置 忽略的文件 ② 将,glignore 文件加入版本库并提文
                                                          设置只有自己需要忽略的文件 修改 git/info/exclude 文件 可使用正则表达式 例如 * [oa] 等价于 * o 和 * a
                                                              添加单个文件 git add somefile.txt
                                           添加新文件到版本库 添加斯有txt文件 git add *.txt
                                                               添加所有文件 git add . ②包括子目录,但不包括空目录
                                           提交 ____ git commit -m "add all txt files"
                                                     提交所有修改 git commit -m "some msg" -a
                                             提交 提交单个文件 git commit -m "add msg to readme.txt" readme.txt
增计键文 git commit -C head -a --amend 不会产生额价键交历史记录
                                                         撤销、2个文件 git checkout head readme.txt todo.txt
數消為未提交的給款 服務所有以文件 git checkout head "txt
                                                                  觀询所有文件 git checkout head ...

反转確交 医转律交包不律交 例:反转截近一次推交 git revert -- no commit head 相当于推交最近的一次推交的仮操作
                                             撤销修改
                                                        列出本地分支 git branch
                                                      列出所有分支
                                                                 git branch -a
                                                      基于当前分支的末梢的建新分支 git branch <br/>obranchname>
                                                      检出分支
                                                               git checkout <branchname>
                     Local
                                                      基于当前分支的末梢创建新分支并检出分支 git checkout -b <br/> oranchname>
                                                     基于某次提交、分支或核签的健議分支 git branch emputy bfe57de0 用来查看某个历史断面侧方便 git branch emputy2 emputy
                                                               会并并建文 git merge <br/>
如果发生了中央,就不会自动推交,如果中央很多,不想立即解决它们,可以直接使用 git checkout head . 最销<br/>
音通合并 会并但不提交 git merge —no-commit 参数并未生效
                                             分支
                                                     当两个人合作开发一个新功能时,需要在一个分支上提交多次,开发完成之后再压合成一次提交
                                                              日常操作
                                                      git branch -d new2 如果分支沒有被合并会翻除失败 git branch -D new2 即使分支沒有被合并也腦附不误
                                                        冲突很少时  直接编辑有冲突的文件然后提交即可
                                                                         ① 会生成 BACKUP, BASE, LOCAL和 REMOTE四个文件
                                             解決冲突 / 神突比较负急时 git merge tool ②然后自动阐用中荣解之工具,比较稀酌给是如野安装,TortosieSVN 就会自动调用 TortosieMerge ①解去之后手动删除。crig 文件(中央解表之前的文件备份)
① 提交
                                                      显示标签列表 git tag
                                                    检出标签 git checkout 1.0 查看标签断面的银方便的方法,但是不能提交
msvsGit1.7.1
                                                     git branch b1.1 1.1
由标签创建分支
git checkout -b b1.0 1.0
                                                    删除标签 git tag -d 1.0
                                                       当前状态 git status
                                                                git log

    査看状态
    历史记录
    gtk
    窒暑当前分支历史记录

    gtk
    gtk <br/>・ gtk <br/>・ chranchname>
    蛮易等分支历史记录

                                                                         gitk --all 查看所有分支
                                                       每个分支最后的提交 git branch -v
                                             git archive --format=zip head>nb.zip
其它 导出版本库 git archive --format=zip --prefix=nb1.0/ head>nb.zip
                                                          git clone <url> 例: git clone d:\git\nb
                                             gt remote add < 姚念 < 远程能本席的URL>
示加远程能本库的例名
添加的名称本库的例名
添加的名称全自协会加 2 个 config remote origin uf=d \usb remote origin feltch=+refs/heads/* refs/remotes/origin/*
                                   初始化
                                                     删除远程库的别名和相关分支 git remote rm <别名>
                                             创建一个无本地分支的库 ♥ gt init -bare 当需要一个公用的中央库时,丰常适合把它建筑 bare 库公士 %社选程分支 gt branch -r

    列出运程分支
    git branch -r

    分支
    關係运程库中已经不存在的分支
    git remote prune origin

                                                              別: git fetch origin orgin 是远程能的數认別名
別: git fetch d/:git/lnb
                                                                                           等价于 git pull origin
                                  日常操作
                                                              git pull 無政策 branch master ne mote=origin和 branch master ne mote=origin和 branch master nerge=refs/heads/master 如果使用了git clone 克隆迈程库会自动添加这个概赏
                                                                                 git pull d:\git\nb master 由于不是从默认远程库获取,所以要指定获取哪个分支
                                               推入远程库 git push origin master 远程库的 master 不能是当前分支
                                                                      1) 启动 Git Bash 控制台 (git-bash.bat)
                                                                     生成 SSH Key 3 s.sh-keygen -t.rsa -C "zahuifan@163.com" #示 Enter passphrase 財要輸入密码
                      Remote
```



联系我

- 博客园: 削微寒
- 或者直接提 Pr, Issues

↑返回顶部