Git 使用

全局配置

设置系统级别的权限

- git config -- system alias.st status
- git config -- global alias.st status

本用户级别的权限

- git config --global user.name ""
- git config -- global color.ui true
- git config -- global user.email ""
- git config --unset --global user.name
  - 删除配置

初始化

- git init
- 设置忽略文件
  - 设置所有人都要忽略的文件
    - Subtopic
  - 自身版本库要忽略的文件
- 添加文件
  - git add file.name
- 提交
  - git commit -m "这是注释"
- 文件目录结构
  - Subtopic
- 显示版本库.git 位置
  - git rev-parse --git-dir
- 显示当前目录到工作区的深度
  - git rev-parse -- show-cdup
- 当前版本库中的某一目录相对工作区根目录的相对目录
  - git rev-parse --show-prefix
- 暂存区
  - .git/index指向的就是该区
- 工作区
- 版本库
- git clone
  - git clone <repository> <directory>
    - 将repository 克隆到指定文件目录，该目录即作为工作目录
  - git clone --bare <repository> <directory>
    - 创建不包含工作区的版本库，只有版本库的内容
  - git clone --mirror <repository> <directory>
    - 创建不包含工作区的版本库，只有版本库的内容

注意事项

- 这里的所有remote-repo 均指的是仓库名
- 远程master分支
- origin/master
- origin 指代远程仓库别名
- 指的是HEAD 头指针指向了一个具体的提交而不是一个引用分支
- 头指针分离
- 如何处理头指针分离
  - 使用merge
  - 1 切换到要合并的分支
  - 2 找到分离的提交ID
  - 3 git merge acc2f43
- git ls-files --with-tree=HEAD^
  - 查看历史版本的文件列表
- git cat-file -p HEAD^:welcome.txt
  - 查看历史版本中文件内容
- git mv welcome.txt README
  - 更改文件名

remote 操作

push 提交本地到远程分支

- 将本地的master 分支推送到远程的master分支上(远程的origin 即为master)
  - git push origin master
- 将本地的test分支退送到远程的master分支
  - git push origin test:master
  - git push origin :<local-branch>:<remote-branch>
- 用空格将远程分支、即把一个local空分支提交到远程分支
  - git push origin :<remote-branch>
  - 删除远程分支

从远程版本库获取

- fetch 获取但不合并
  - git fetch -remote_repo>
    - 更新远程所有分支到本地
  - git fetch remote_repo remote_branch
    - 更新远程分支到本地
  - git fetch remote_repo remote_branch_name:local_branch_name
    - 更新并创建本地分支
- pull 获取并合并
  - 获取远程origin/next并与当前分支合并
  - git pull origin next_Branch
  - git pull origin <remote_branch>:<local_branch>

- git checkout -b myNewBranch origin/dragon
  - 检出远程分支
- git checkout -b [name] [remoteName]
  - 查看远程repo
- git remote -v
  - 查看远程repo
- git remote add <name> <url>
  - 添加远程repo
- git remote rm <repo>
  - 删除远程repo
- git pull <remote_repo> <local-repo>
  - 拉取远程repo
- git push <remote_repo> <local-repo>
  - 推送远程repo

常见操作

移动文件

```
$ mkdir lib
$ git mv hello.rb lib
$ git status
```

```
$ mkdir lib
$ mv hello.rb lib
$ git add lib/hello.rb
$ git rm hello.rb
```

查看远程库信息

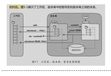
- git remote show origin
- Subtopic

显示工作区根目录

- git rev-parse --git-dir

文件修改 M 标记

- 第一列的M 代表版本库与暂存区比较有改动
- 第二列M 代表工作区和暂存区比较有改动



- git diff

- git对象关系图

恢复删除的文件

- 执行文件的删除并提交只是在最新的提价中删除了文件，历史提交中还存在
- git cat-file -p HEAD~1:welcome.txt > welcome.txt
- git checkout HEAD~1 --welcome.txt

追本溯源

- commit a52b8b57a9813d5eaeedbf6128d92874da5f47f93
  Author: sunjianfeng <Sun_224@sina.com>
  Date: Sun Jul 17 22:33:20 2016 +0800
  - 1 git log -1 HEAD
- .git/HEAD
  .git/logs/HEAD
  .git/logs/refs/heads/master
  .git/logs/refs/remotes/origin/master
  .git/refs/heads/master
  .git/refs/remotes/origin/master
  - 2 find .git –name HEAD –o –name master
- ref: refs/heads/master
  - 3 cat .git/HEAD
- a52b8b57a8613d5eaeedbf6128d92874da5147f93
  - 4 cat .git/refs/heads/master
- commit
  - git cat-file -t a52b
- tree d0535e44799dc56a50c3fde7a00053b5e84a3853
  parent c15193c70941bac149873a116d55b7fe1c95785843
  author sunjianfeng <Sun_224@sina.com> 1468766000 +0800
  committer sunjianfeng <Sun_224@sina.com> 1468766000 +0800
  - 2 git cat-file -p a52b

- Subtopic
  - 5 结论：master 指向一个commitID

冲突的合并

- git rev-parse <CommitID> 显示该id
- git mv welcome.txt README
  - 更改文件名
- git rev-parse
  - git rev-parse --git-dir 显示版本库位置
  - git rev-parse --symbolic --branches
    - 显示分支
  - git rev-parse --symbolic --tags
    - 显示里程碑
  - git rev-parse HEAD
    - 显示SHA1值
- git pull = git fetch + git merge
- git merge refs/remotes/origin/master
  - 与远程分支合并

100644 c21c3b5dafbddb1e3daff0aeae07e39137621568 0 Sun.txt
100644 6926997590728 32da4df6e56019cab6df906604f 0 add.txt
100644 03ec644e8c5a79d99f5033822f78bc1dbaeb980 0 add2.txt
100644 dc3f36eed300b300d2e71be6dc63de2b1097e07c 0 welcome.txt

- git ls-files -s
  - 列出冲突文件记录
- Subtopic
  - 编号为 1 代表冲突之前的文件，即双方共同的祖先版本
- Subtopic
  - 编号为2 的为当前分支修改的文件
- 编号为3 的为其他合并分支修改的文件

local 操作

查找工作区内容

- git grep "查找的内容"

commit 操作

- git commit --allow-empty -m "comment"
  - 允许没有任何更改的提交
- git commit -a -m ""
  - 提交所有已记录的修改
- git commit -m "" filename
  - 提交单个

add 操作

- git add.
  - 添加当前命令所在目录的所有文件
- git add -u
  - 添加本地所有修改过的文件
- git add -A
  - 添加所有本地文件，包括未跟踪的文件
- git add -i
  - 交互式添加

git rm操作

- git rm --fileName
  - 删除文件，并提交到暂存区

查看状态

- log日志
  - git log --stat
    - 查看每次提交的文件的变更统计
  - git log --pretty=oneline
    - 精简输出
  - git log --graph
  - git log --pretty=raw -1
- 每个分支的五后一次提交
  - git branch -v
- status 操作
  - git status
    - 查看当前工作目录暂与存区的状态
  - git status -s
    - 精简形式
  - 标示位含义
    - 第1列的M
      - 版本库和暂存区比较，文件有改动
    - 第2列的M
      - 工作区和暂存区比较，文件有改动

branch操作

- git branch
  - 查看本地分支
- git branch -r
  - 查看远程分支
- git branch <branch-name>
  - 创建本地分支，但不会切换到新的分支
- git checkout <branch-name>
  - 切换本地分支
- git branch -d <branch-name>
  - 删除本地分支
- git branch -m <branchName> <new_Branch>
  - 重命名分支
- git merge <branchName>
  - 合并当前分支和BranchName 分支

checkout 操作

- 检出分支
  - git checkout -b <newBranch> <start_point /commitId>
    - 基于某次提交创建分支，只包含该提交的内容
  - git checkout -b local-branchname origin/remote_branchname
    - 检出远程分支
  - git checkout -b <branch-name>
    - 检出本地分支并切换，会包含当前分支的所有改动(包括工作区)
- 撤销修改
  - git checkout <commit-id> --file
    - 用指定的commit撤销尚未提交的修改 用版本库中指定提交的文件直接覆盖暂存区和工作区
    - git checkout HEAD readme.txt
      todo.txt
    - git checkout HEAD *.txt
    - git checkout HEAD .
      - 撤销所有
    - 省略commitID 则从暂存区检出
  - git checkout .
    - 用暂存区的所有文件覆盖工作区，即取消本地所有的修改
  - git checkout --fileName
    - 检出暂存区的文件覆盖工作区
  - Subtopic
  - git checkout HEAD 使用版本库的文件覆盖暂存区和工作区

reset 撤销提交操作

- git reset --hard <commit>
  - 变更引用，使用版本库覆盖暂存区和工作区
  - git reset --hard HEAD~2
- git reset -- soft <commit>
  - 变更版本库index,不影响工作区和暂存区
    现有内容，将commitID 之后的变更转为暂存区状态，撤销版本库commitID之后的提交至暂存区，不影响工作区和暂存区的内容
  - 只改变版本库
- git rest --mixed <commit>
  - 撤销版本库和暂存区的提交至工作区，变为未提交状态，不影响工作区的内容
  - 只改变版本库和暂存区
- git reset HEAD -- fileName
  - 将提交撤出暂存区,并不会影响并修改本地的内容
  - 只改变暂存区，工作区不变
- git reset --fileName
  - 将文件撤出暂存区，相当于git add 的反向操作，不影响工作区
- git reset = git reset HEAD
  - 工作区不受影响，相当于将git add之后更新到暂存区的内容撤出暂存区
  - 只改变暂存区
  - 相当于git add 的反向操作

diff 操作

- git diff <cm1> <cm2>
  - 比较两次提交的差异
- git diff <branch> <branch2>
- git diff
  - 比较工作区和暂存区
- git diff --cached/--staged
  - 比较暂存区和版本库的差异
- git diff HEAD <file>
  - 比较工作区和版本库的差异<head 指向的分支库>
- git diff -- stat
  - 仅仅比较统计信息
- git diff localRepo origin/ remoteRepo
  - 比较远程和本地

cat-file 操作

- git cat-file -p
  - 查看git对象内容
- git cat-file -t

revert 操作

- git revert HEAD^
  - HEAD之前的提交的反转提交，不会影响工作区，只是针对版本库

stash 操作

- 保存和恢复工作进度
  - 需要切换到另一分支前，保存当前工作进度，回来后可以恢复 git stash pop
- 只有被版本跟踪的文件才会得到stash保存 也即是说，新文件需要add 到暂存区
- apply 只会读取暂存区的数据，通过 apply 后，暂存区的数据仍然存在；
  而 pop 是取出最新的一次暂存数据，取出后，这次数据就不会存在于暂存区中了。
- 使用git stash apply @{x} ，可以将编号x的缓存释放出来，但是该缓存还存在于stash中
  而 git stash apply，会将当前分支的最后一次缓存的内容释放出来，但是刚才的记录还存在stash中
  而 git stash pop，也会将当前分支的最后一次缓存的内容释放出来，但是刚才的记录不存在stash中
- 我们在本地工作区删除所有文件，在从暂存区恢复
  - rm *.txt
  - git ls-files 查看暂存区的文件

rebase 操作

- 原理
  - 现有分支mywork,origin 想要将origin 合并到mywork，step1. 将mywork 的提交存为临时补丁放入到.git/rebase目录，然后将origin的最新提交更新到mywork，然后再把保存的补丁拼接到mywork

- 和merge的区别
  - 举例：整理衣柜抽屉 merge：一股脑全部塞进抽屉 rebase: 有顺序一件一件放

- 应用场景
  - 假设目前有两个分支origin,mywork，origin 分支和 mywork 都并行进行了多次的commit, 想要将origin 分支合并到mywork分支
- git rebase other_branch
  - 将other_branch 分支合并到当前分支
- git rebase -- continue
- git rebase --abort

tag 操作

- 创建标签
  - 为当前分支的最近一次提交创建标签
    - git tag t1.0
  - 为分支的最近一次提交创建标签btag1.0
    - git tag btag1.0 newBranchName
  - 为某次历史提交创建标签 t1.0
    - git tag t1.0 45435d
- 显示标签列表
  - git tag
- 检出标签
  - git checkout t1.0
- 根据标签创建分支
  - git branch b1.0 t1.0
  - git branch -b b1.0 t1.0
- 删除标签
  - git tag -d t1.0