Git 常用命令

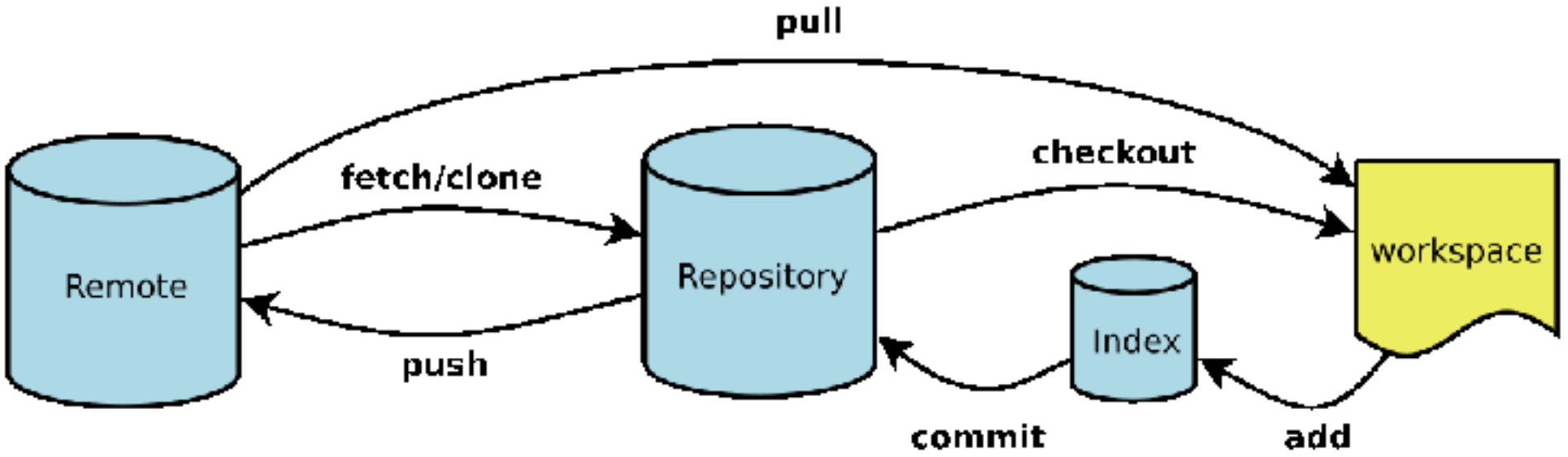
罗流毅

xluoly@gmail.com

2018-7-26

Overview

- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库



新建代码库

- 在当前目录新建一个 Git 代码库

```
$ git init
```

- 新建一个目录，将其初始化为 Git 代码库

```
$ git init <project-name>
```

- 克隆一仓库，并检出 master 分支到工作区

```
$ git clone <repo-url>
```

基本配置

- 显示当前的 Git 配置

```
$ git config --list
```

- 编辑 Git 配置文件

```
$ git config -e [--global]
```

- 设置提交代码时的用户信息

```
$ git config [--global] user.name "<name>"
$ git config [--global] user.email "<email-address>"
```

将文件放入暂存区

- 添加指定文件到暂存区

```
$ git add <file1> [file2] ...
```

- 添加指定目录及其子目录的所有文件到暂存区

```
$ git add <dir>
```

- 添加当前目录以及子目录的所有文件到暂存区

```
$ git add .
```

删除、移动文件

- 删除工作区文件，并且将这次删除放入暂存区

```
$ git rm <file1> [file2] ...
```

- 停止追踪指定文件，但该文件会保留在工作区

```
$ git rm --cached <file>
```

- 改名或剪切文件，并且将这次改名或剪切放入暂存区

```
$ git mv <file-origin> <file-renamed>
```

代码提交

- 提交暂存区文件到仓库区

```
$ git commit -m "<message>"
```

- 提交暂存区的指定文件到仓库区

```
$ git commit <file1> [file2] ... -m "<message>"
```

- 提交工作区自上次 commit 之后的变化，直接到仓库区

```
$ git commit -a
```

- 提交时显示所有 diff 信息

```
$ git commit -v
```

- 改写上一次 commit 的提交信息

```
$ git commit --amend -m "<message>"
```

- 重做上一次 commit，并包括指定文件的新变化

```
$ git commit --amend <file1> [file2] ...
```

查看分支

- 列出所有本地分支

```
$ git branch
```

- 列出所有远程分支

```
$ git branch -r
```

- 列出所有本地分支和远程分支

```
$ git branch -a
```

创建分支

- 新建一个分支，但依然停留在当前分支

```
$ git branch <branch-name>
```

- 新建一个分支，并切换到该分支

```
$ git checkout -b <branch>
```

- 新建一个分支，指向指定 commit

```
$ git branch <branch> <commit>
```

- 新建一个分支，与指定的远程分支建立追踪关系

```
$ git branch --track <branch> <remote-branch>
```

- 切换到指定分支，并更新工作区

```
$ git checkout <branch-name>
```

- 在现有分支与指定的远程分支之间建立追踪关系

```
$ git branch --set-upstream <branch> <remote-branch>
```

删除分支

- 删除本地分支

```
$ git branch -d <branch-name>
```

- 如果该分支未合并回原分支，会提示删除失败，强制删除可用如下命令

```
$ git branch -D <branch-name>
```

- 删除远程分支

```
$ git push origin --delete [branch-name]
$ git branch -dr [remote/branch]
```

合并

- 合并指定分支到当前分支

```
$ git merge <branch>
```

- 选择一个 commit，合并进当前分支

```
$ git cherry-pick [commit]
```

查看标签

- 列出所有 tag

```
$ git tag
```

- 查看 tag 信息

```
$ git show <tag-name>
```

新建和删除标签

- 新建一个 tag 在当前 commit

```
$ git tag <tag-name>
```

- 新建一个 tag 在指定 commit

```
$ git tag <tag-name> <commit-hash>
```

- 删除本地 tag

```
$ git tag -d <tag-name>
```

- 删除远程 tag

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

提交和检出标签

- 提交指定 tag

```
$ git push <remote> <tag-name>
```

- 提交所有 tag

```
$ git push <remote> --tags
```

- 新建一个分支，指向某个 tag

```
$ git checkout -b <branch-name> <tag-name>
```

查看状态

- 显示工作空间所有变更的文件

```
$ git status
```

查看 log

- 查看当前分支的版本历史

```
$ git log
```

- 查看 commit 历史，以及每次 commit 发生变更的文件

```
$ git log --stat
```

- 查看某个 tag 之后的所有变动，每个 commit 占据一行

```
$ git log <tag-name> HEAD --pretty=format:%s
```

- 查看某个文件的版本历史，包括文件改名

```
$ git log --follow <file>
```

- 查看指定文件相关的每一次 diff

```
$ git log -p <file>
```

查看文件每行的最后修改信息

- 显示指定文件的每一行被什么人在什么时间修改过

```
$ git blame <file>
```

比较差异

- 显示暂存区和工作区的差异

```
$ git diff
```

- 显示暂存区和上一个 commit 的差异

```
$ git diff --cached [file]
```

- 显示工作区与当前分支最新 commit 之间的差异

```
$ git diff HEAD
```

- 显示两次提交之间的差异

```
$ git diff [first-branch]...[second-branch]
```

查看提交的详细信息

- 显示某次提交的元数据和内容变化

```
$ git show <commit>
```

- 显示某次提交发生变化的文件

```
$ git show --name-only <commit>
```

- 显示某次提交时，某个文件的内容

```
$ git show <commit>:<filename>
```

- 显示当前分支的最近几次提交

```
$ git reflog
```

远程仓库操作

- 显示所有远程仓库

```
$ git remote -v
```

- 显示某个远程仓库的信息

```
$ git remote show [remote]
```

- 增加一个新的远程仓库，并命名

```
$ git remote add [shortname] [url]
```

同步远程仓库

- 下载远程仓库的所有变动

```
$ git fetch [remote]
```

- 取回远程仓库的变化，并与本地分支合并

```
$ git pull [remote] [branch]
```

- 上传本地指定分支到远程仓库

```
$ git push [remote] [branch]
```

- 强行推送当前分支到远程仓库，即使有冲突

```
$ git push [remote] --force
```

- 推送所有分支到远程仓库

```
$ git push [remote] --all
```

撤销未提交的修改

- 恢复暂存区的指定文件到工作区

```
$ git checkout [file]
```

- 恢复某个 commit 的指定文件到工作区

```
$ git checkout [commit] [file]
```

- 恢复上一个 commit 的所有文件到工作区

```
$ git checkout .
```

重置暂存区

- 重置暂存区的指定文件，与上一次 commit 保持一致，但工作区不变

```
$ git reset [file]
```

- 重置暂存区与工作区，与上一次 commit 保持一致

```
$ git reset --hard
```

- 重置当前分支的指针为指定 commit，同时重置暂存区，但工作区不变

```
$ git reset [commit]
```

- 重置当前分支的 HEAD 为指定 commit，同时重置暂存区和工作区，与指定 commit 一致

```
$ git reset --hard [commit]
```

- 重置当前 HEAD 为指定 commit，但保持暂存区和工作区不变

```
$ git reset --keep [commit]
```

回滚

- 新建一个 commit，用来撤销指定 commit 后者的所有变化都将被前者抵消，并且应用到当前分支

```
$ git revert [commit]
```

储藏未提交的修改

- 储藏

```
$ git stash
```

- 显示/查看

```
$ git stash list
```

- 取出某次储藏

```
$ git stash apply stash@{1}
```

- 取消最近一次储藏

```
$ git stash pop
```

- 从储藏中创建分支

```
$ git stash branch testchanges
```

- 清空

```
$ git stash clear
```

打包

- 导出最新的版本，生成一个可供发布的压缩包

```
$ git archive HEAD -o last.zip
```

- 导出为 tar.gz 格式

```
$ git archive gzip > last.tar.gz
```

The End