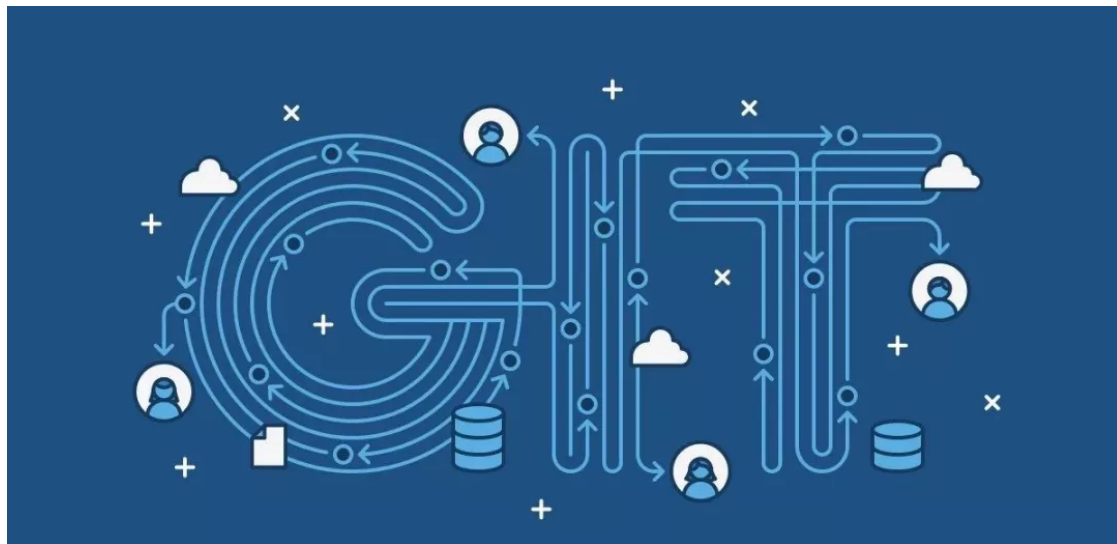


Git常用命令速查表

极客时间



根据官方文档和网络资源整理

名词

- master: 默认开发分支
- origin: 默认远程版本库
- Index / Stage: 暂存区
- Workspace: 工作区
- Repository: 仓库区（或本地仓库）
- Remote: 远程仓库

一、新建代码库

1. # 在当前目录新建一个Git代码库
2. `$ git init`
3. # 新建一个目录，将其初始化为Git代码库
4. `$ git init [project-name]`
5. # 下载一个项目和它的整个代码历史
6. `$ git clone [url]`

二、配置

Git的设置文件为.gitconfig，它可以在用户主目录下（全局配置），也可以在项目目录下（项目配置）。

1. # 显示当前的Git配置
2. \$ git config --list
3. # 编辑Git配置文件
4. \$ git config -e [--global]
5. # 设置提交代码时的用户信息
6. \$ git config [--global] user.name "[name]"
7. \$ git config [--global] user.email "[email address]"

三、增加/删除/修改文件

1. #查看状态
2. \$ git status
3. #查看变更内容
4. \$ git diff
5. # 添加指定文件到暂存区
6. \$ git add [file1] [file2] ...
7. # 添加指定目录到暂存区，包括子目录
8. \$ git add [dir]
9. # 添加当前目录的所有文件到暂存区
10. \$ git add .
11. # 添加每个变化前，都会要求确认
12. # 对于同一个文件的多处变化，可以实现分次提交
13. \$ git add -p
14. # 删除工作区文件，并且将这次删除放入暂存区
15. \$ git rm [file1] [file2] ...
16. # 停止追踪指定文件，但该文件会保留在工作区
17. \$ git rm --cached [file]
18. # 改名文件，并且将这个改名放入暂存区
19. \$ git mv [file-original] [file-renamed]

四、代码提交

1. # 提交暂存区到仓库区
2. \$ git commit -m [message]
3. # 提交暂存区的指定文件到仓库区
4. \$ git commit [file1] [file2] ... -m [message]

```
5. # 提交工作区自上次commit之后的变化，直接到仓库区
6. $ git commit -a
7. # 提交时显示所有diff信息
8. $ git commit -v
9. # 使用一次新的commit，替代上一次提交
10. # 如果代码没有任何新变化，则用来改写上一次commit的提交信息
11. $ git commit --amend -m [message]
12. # 重做上一次commit，并包括指定文件的新变化
13. $ git commit --amend [file1] [file2] ...
```

五、分支

```
1. #显示所有本地分支
2. $ git branch
3. # 列出所有远程分支
4. $ git branch -r
5. # 列出所有本地分支和远程分支
6. $ git branch -a
7. # 新建一个分支，但依然停留在当前分支
8. $ git branch [branch-name]
9. # 新建一个分支，与指定的远程分支建立追踪关系
10. $ git branch --track [branch] [remote-branch]
11. # 删除分支
12. $ git branch -d [branch-name]
13. # 删除远程分支
14. $ git push origin --delete [branch-name]
15. $ git branch -dr [remote/branch]
16. # 新建一个分支，并切换到该分支
17. $ git checkout -b [branch]

18. # 切换到指定分支，并更新工作区
19. $ git checkout [branch-name]
20. # 切换到上一个分支
21. $ git checkout -
22. # 建立追踪关系，在现有分支与指定的远程分支之间
23. $ git branch --set-upstream [branch] [remote-branch]
24. # 合并指定分支到当前分支
25. $ git merge [branch]
```

26. # 衍合指定分支到当前分支
27. \$ git rebase <branch>
28. # 选择一个commit, 合并进当前分支
29. \$ git cherry-pick [commit]

六、标签

1. \$ git tag # 列出所有本地标签
2. \$ git tag <tagname> # 基于最新提交创建标签
3. \$ git tag -d <tagname> # 删除标签

1. # 删除远程tag
2. \$ git push origin :refs/tags/[tagName]
3. # 查看tag信息
4. \$ git show [tag]
5. # 提交指定tag
6. \$ git push [remote] [tag]
7. # 提交所有tag
8. \$ git push [remote] --tags
9. # 新建一个分支, 指向某个tag
10. \$ git checkout -b [branch] [tag]

七、查看信息

1. # 显示有变更的文件
2. \$ git status
3. # 显示当前分支的版本历史
4. \$ git log
5. # 显示commit历史, 以及每次commit发生变更的文件
6. \$ git log --stat
7. # 搜索提交历史, 根据关键词
8. \$ git log -S [keyword]
9. # 显示某个commit之后的所有变动, 每个commit占据一行
10. \$ git log [tag] HEAD --pretty=format:%s
11. # 显示某个commit之后的所有变动, 其"提交说明"必须符合搜索条件
12. \$ git log [tag] HEAD --grep feature

13. # 显示某个文件的版本历史，包括文件改名
14. \$ git log --follow [file]
15. \$ git whatchanged [file]
16. # 显示指定文件相关的每一次diff
17. \$ git log -p [file]
18. # 显示过去5次提交
19. \$ git log -5 --pretty --oneline
20. # 显示所有提交过的用户，按提交次数排序
21. \$ git shortlog -sn
22. # 显示指定文件是什么人在什么时间修改过
23. \$ git blame [file]
24. # 显示暂存区和工作区的差异
25. \$ git diff
26. # 显示暂存区和上一个commit的差异
27. \$ git diff --cached [file]
28. # 显示工作区与当前分支最新commit之间的差异
29. \$ git diff HEAD
30. # 显示两次提交之间的差异
31. \$ git diff [first-branch]...[second-branch]
32. # 显示今天你写了多少行代码
33. \$ git diff --shortstat "@{0 day ago}"
34. # 显示某次提交的元数据和内容变化
35. \$ git show [commit]
36. # 显示某次提交发生变化的文件
37. \$ git show --name-only [commit]
38. # 显示某次提交时，某个文件的内容
39. \$ git show [commit]:[filename]
40. # 显示当前分支的最近几次提交
41. \$ git reflog

八、远程操作

- 1.
2. # 下载远程仓库的所有变动
3. \$ git fetch [remote]
4. # 取回远程仓库的变化，并与本地分支合并
5. \$ git pull [remote] [branch]

```
6. # 显示所有远程仓库
7. $ git remote -v
8. # 显示某个远程仓库的信息
9. $ git remote show [remote]
10. # 增加一个新的远程仓库，并命名
11. $ git remote add [shortname] [url]
12. # 上传本地指定分支到远程仓库
13. $ git push [remote] [branch]
14. # 强行推送当前分支到远程仓库，即使有冲突
15. $ git push [remote] --force
16. # 推送所有分支到远程仓库
17. $ git push [remote] --all
```

```
1. $ git push <remote> :<branch/tag-name> #删除远程分支或标签
2. $ git push --tags #上传所有标签
```

九、撤销

```
1. $ git reset --hard HEAD #撤销工作目录中所有未提交文件的修改内容
2. $ git checkout HEAD <file> #撤销指定的未提交文件的修改内容
3. $ git revert <commit> #撤销指定的提交
4. $ git log --before="1 days" #退回到之前1天的版本
```

```
1. # 恢复暂存区的指定文件到工作区
2. $ git checkout [file]
3. # 恢复某个commit的指定文件到暂存区和工作区
4. $ git checkout [commit] [file]
5. # 恢复暂存区的所有文件到工作区
6. $ git checkout .
7. # 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
8. $ git reset [file]
9. # 重置暂存区与工作区，与上一次commit保持一致
10. $ git reset --hard
11. # 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
12. $ git reset [commit]
13. # 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
14. $ git reset --hard [commit]
```

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

15. # 重置当前HEAD为指定commit，但保持暂存区和工作区不变

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

17. # 新建一个commit，用来撤销指定commit

18. # 后者的所有变化都将被前者抵消，并且应用到当前分支

```
19. $ git revert [commit]
```

20. # 暂时将未提交的变化移除，稍后再移入

```
21. $ git stash
```

```
22. $ git stash pop
```

十、其他

生成一个可供发布的压缩包

```
1. $ git archive
```