编 程 学 习 笔 记

Git基础操作

Code

作 者 姓 名： 颜佳

目 录

[1 修改配置 1](#_Toc76239928)

[2 新建本地仓库 1](#_Toc76239929)

[2.1 初始化一个空目录 1](#_Toc76239930)

[2.2 从远程仓库复制到本地仓库 1](#_Toc76239931)

[3 把本地仓库推送到远程 1](#_Toc76239932)

[3.1 生成SSH公钥与私钥对 1](#_Toc76239933)

[3.2 本地关联远程仓库 1](#_Toc76239934)

[3.3 推送到远程仓库 2](#_Toc76239935)

[4 查看文件状态 2](#_Toc76239936)

[5 分支 2](#_Toc76239937)

[5.1 查看分支 2](#_Toc76239938)

[5.2 创建分支 2](#_Toc76239939)

[5.3 切换分支 2](#_Toc76239940)

[5.4 合并分支 2](#_Toc76239941)

[6 标签 2](#_Toc76239942)

[6.1 查看标签 3](#_Toc76239943)

[6.2 创建标签 3](#_Toc76239944)

[6.3 共享标签 3](#_Toc76239945)

[6.4 删除标签 3](#_Toc76239946)

[6.5 检出标签 3](#_Toc76239947)

[7 撤销操作 3](#_Toc76239948)

[7.1 合并两次提交 3](#_Toc76239949)

[7.2 取消暂存的文件 4](#_Toc76239950)

[7.3 撤消对文件的修改 4](#_Toc76239951)

[8 查看日志 4](#_Toc76239952)

[9 忽略文件 4](#_Toc76239953)

[10 比较差异 5](#_Toc76239954)

[11 删除文件，不再追踪 5](#_Toc76239955)

[12 重命名文件 5](#_Toc76239956)

[13 查看远程服务器 5](#_Toc76239957)

[13.1 列出指定的每一个远程服务器的简写 5](#_Toc76239958)

[13.2 列出需要读写远程仓库使用的 Git 保存的简写与其对应的 URL 5](#_Toc76239959)

[13.3 查看某一个远程仓库的更多信息 5](#_Toc76239960)

[14 拉取远程仓库 6](#_Toc76239961)

[14.1 git fetch origin 6](#_Toc76239962)

[14.2 git pull 6](#_Toc76239963)

[15 远程仓库简写名的重命名 6](#_Toc76239964)

[16 远程仓库的移除 6](#_Toc76239965)

# 修改配置

git config --global user.name "YanJiaRZGN"

git config --global user.mail "yanjiargzn@126.com

# 从github上下代码

ssh-keygen -t rsa -C [yanjiasolstice@email.com](mailto:yanjiasolstice@email.com)

# 新建本地仓库

## 初始化一个空目录

git init

git add main.c

git commit -m "第一次提交"

以下命令可以跳过git add命令

git commit -a -m "第一次提交"

## 从远程仓库复制到本地仓库

git clone git@github.com:YanJiaRGZN/Git-and-GitHub.git

# 把本地仓库推送到远程

## 生成SSH公钥与私钥对

ssh-keygen -t rsa

## 本地关联远程仓库

git remote add **标识名(简写)** git@github.com:YanJiaRGZN/Git-and-GitHub.git

git remote add **origin** [git@github.com:YanJiaRGZN/Git-and-GitHub.git](mailto:git@github.com:YanJiaRGZN/Git-and-GitHub.git)

origin 是Git给你克隆的**仓库服务器**的默认名字

## 推送到远程仓库

git push **标识名** master

git push **origin** master

# 查看文件状态

git status

git status -s

git status --short

# 分支

## 查看分支

git branch

## 创建分支

git branch dev

## 切换分支

git checkout dev

## 合并分支

git checkout master

git merge dev

快速合并：有一个分支没有改变；

三方合并:

合并冲突：

# 标签

给仓库历史中的某一个提交打上标签

## 查看标签

git tag

git tag -l

git tag –list

## 创建标签

git tag –a v1.4 –m ‘my version v1.4’

git show v1.4 显示标签详细信息（打标签者的信息、打标签的日期时间、附注信息，然后显示具体的提交信息）

## 共享标签

git push origin v1.5

必须显示地推送到服务器上

## 删除标签

git tag -d <tagname>

git push <remote>:refs/tags/<tagname>

必须显式地删除远程服务器上的标签

git push origin --delete <tagname>

## 检出标签

git checkout v1.4

查看某个标签所指向的文件版本。

# 撤销操作

## 合并两次提交

$ git commit -m 'initial commit'

$ git add forgotten\_file

$ git commit --amend

最终你只会有一个提交——第二次提交将代替第一次提交的结果。修补提交最明显的价值是可以稍微改进你最后的提交，而不会让“啊，忘了添加一个文件”或

者 “小修补，修正笔误”这种提交信息弄乱你的仓库历史。

## 取消暂存的文件

git reset HEAD <file>

## 撤消对文件的修改

git checkout -- <file>

# 查看日志

git log

git log --oneline

git log --oneline --graph

# 忽略文件

.gitignore

文件 .gitignore 的格式规范如下：

• 所有空行或者以 # 开头的行都会被 Git 忽略。

• 可以使用标准的 glob 模式匹配，它会递归地应用在整个工作区中。

• 匹配模式可以以（/）开头防止递归。

• 匹配模式可以以（/）结尾指定目录。

• 要忽略指定模式以外的文件或目录，可以在模式前加上叹号（!）取反。

# 比较差异

git diff

比较的是工作目录中当前文件和暂存区域快照之间的差异

git diff --staged

这条命令将比对已暂存

文件与最后一次提交的文件差异

# 删除文件，不再追踪

git rm --cached README

# 重命名文件

git mv file\_from file\_to

# 查看远程服务器

## 列出指定的每一个远程服务器的简写

git remote

## 列出需要读写远程仓库使用的 Git 保存的简写与其对应的 URL

git remote –v

## 查看某一个远程仓库的更多信息

git remote show origin

# 拉取远程仓库

## git fetch origin

访问远程仓库，从中拉取所有你还没有的数据，只会将数据下载到你的本地仓库——它并不会自动合并或修改你当前的工作。

## git pull

如果你的当前分支设置了**跟踪远程分支**，该命令会自动抓取后合并该远程分支到当前分支。默认情况下，git clone 命令会自动设置本地 master 分支跟踪克隆的远程仓库的 master 分支。运行 git pull 通常会从最初克隆的服务器上抓取数据并自动尝试合并到当前所在的分支。

# 远程仓库简写名的重命名

git remote rename origin origin2

# 远程仓库的移除

git remote remove origin

git remote rm