# Git 命令行操作

[Git 命令行操作 1](#_Toc95079804)

[1.1本地库初始化 2](#_Toc95079805)

[1.2设置签名 2](#_Toc95079806)

[1.3基本操作 3](#_Toc95079807)

[1.3.1 状态查看 3](#_Toc95079808)

[1.3.2 添加 3](#_Toc95079809)

[1.3.3 提交 3](#_Toc95079810)

[1.3.4 查看历史记录 3](#_Toc95079811)

[1.3.5 前进后退 4](#_Toc95079812)

[1.3.6 reset 命令的三个参数对比 4](#_Toc95079813)

[1.3.7 删除文件并找回 4](#_Toc95079814)

[1.3.8 比较文件差异 4](#_Toc95079815)

[1.4 分支管理 5](#_Toc95079816)

[1.4.1 分支的好处 5](#_Toc95079817)

[1.4.2 分支操作 5](#_Toc95079818)

[1.4.3解决冲突 5](#_Toc95079819)

[1.5 版本控制 6](#_Toc95079820)

[1.6 GitHub 6](#_Toc95079821)

[1.6.1 Push项目 6](#_Toc95079822)

[1.6.2 克隆项目 7](#_Toc95079823)

[1.6.3 新增团队成员 8](#_Toc95079824)

[1.6.4 获取其他成员修改信息（pull = fetch + merge） 8](#_Toc95079825)

[1.6.5 解决冲突（只能在最新版本的基础上做修改） 8](#_Toc95079826)

[1.7 SSH登录 8](#_Toc95079827)

[1.8 Git工作流 9](#_Toc95079828)

[1.9 总结 上传项目到GitHub 10](#_Toc95079829)

## 1.1本地库初始化

 命令： git add

mkdir Demo 新建Demo目录

git init 初始化仓库

ls -lA查看隐藏信息

ll .git/查看文件夹内部信息

pwd 返回当前位置

git 目录中存放的是本地库相关的子目录和文件，不要删除，也不要胡

乱修改

## 1.2设置签名

用户名： tom

Email 地址： [goodMorning@atguigu.com](mailto:goodMorning@atguigu.com)

作用： 区分不同开发人员的身份

辨析： 这里设置的签名和登录远程库(代码托管中心)的账号、 密码没有任何关

系。

 命令

 项目级别/仓库级别： 仅在当前本地库范围内有效

 git config user.name tom\_pro

 git config user.email goodMorning\_pro@atguigu.com

 信息保存位置： ./.git/config 文件

 （通常）系统用户级别： 登录当前操作系统的用户范围

 git config --global user.name tom\_glb

 git config --global goodMorning\_pro@atguigu.com

 信息保存位置： ~/.gitconfig 文件

vim good.txt 在vim编辑器创建文件shift+z+z退出

:set nu 显示行号

:wq 退出

tail -n 3 good.txt 显示最后三行文件

## 1.3基本操作

### 1.3.1 状态查看

git status 查看工作区、 暂存区状态

### 1.3.2 添加

git add [file name]

将工作区的“新建/修改” **添加到暂存区**

**git rm –cached good.txt 从暂存区移除文件**

### 1.3.3 提交

git commit -m "commit message" [file name]

将暂存区的内容提交到本地库

### 1.3.4 查看历史记录

git log

多屏显示控制方式：

空格向下翻页

b 向上翻页

q 退出

git log --pretty=oneline 每行显示一个

git log –oneline（只显示当前版本及之前的版本）

**git reflog** 显示索引值的一小部分 到哪个版本需要多少步（所有版本都显示）

HEAD@{移动到当前版本需要多少步}

### 1.3.5 前进后退

 **基于索引值操作[推荐]**

 git reset --hard [局部索引值]

 git reset --hard a6ace91

### 1.3.6 reset 命令的三个参数对比

 --soft 参数

 仅仅在本地库移动 HEAD 指针

 --mixed 参数

 在本地库移动 HEAD 指针

 重置暂存区

 --hard 参数

 在本地库移动 HEAD 指针

 重置暂存区、重置工作区

### 1.3.7 删除文件并找回

 前提： 删除前， 文件存在时的状态**提交到了本地库**。

 操作： git reset --hard [指针位置]

 删除操作已经提交到本地库： 指针位置指向历史记录

 删除操作尚未提交到本地库： 指针位置使用 HEAD

### 1.3.8 比较文件差异

 git diff [文件名]

 将工作区中的文件和暂存区进行比较

 git diff [本地库中历史版本] [文件名]

 将工作区中的文件和本地库历史记录比较

 不带文件名比较多个文件

## 1.4 分支管理

### 1.4.1 分支的好处

 同时并行推进多个功能开发， 提高开发效率

 各个分支在开发过程中， 如果某一个分支开发失败， 不会对其他分支有任

何影响。 失败的分支删除重新开始即可。

### 1.4.2 分支操作

 创建分支：git branch [分支名]

 查看分支：git branch -v

 切换分支：git checkout [分支名]

 合并分支

 第一步： 切换到接受修改的分支（被合并， 增加新内容） 上

git checkout [被合并分支名]

 第二步： 执行 merge 命令

git merge [有新内容分支名]

### 1.4.3解决冲突

 冲突的解决

 第一步： 编辑文件， 删除特殊符号

 第二步： 把文件修改到满意的程度， 保存退出

 第三步： git add [文件名]

 第四步： git commit -m "日志信息"

 注意： 此时 commit 一定不能带具体文件名

## 1.5 版本控制

Git的文件管理机制：快照流

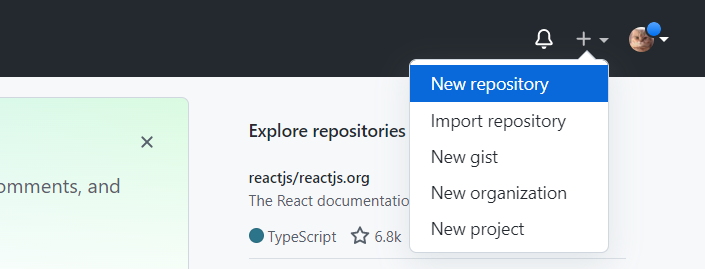
Git 把数据看作是小型文件系统的一组快照。 每次提交更新时 Git 都会对当前的全部文件制作一个快照并保存这个快照的索引。 为了高效， 如果文件没有修改，Git 不再重新存储该文件， 而是只保留一个链接指向之前存储的文件。

版本管理就是一个链条，在链条的基础上，创建分支就是创建一个新的指针，切换分支就是就是切换HEAD指针的位置

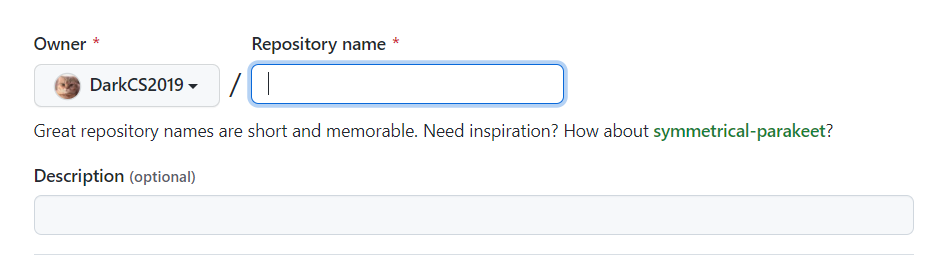
## 1.6 GitHub

### 1.6.1 Push项目

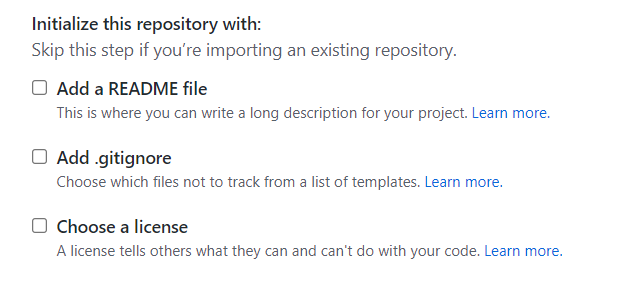
1、创建远程库



2、填写仓库名称、描述

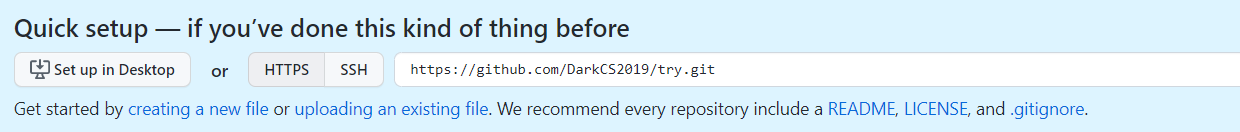


是否使用readme文件初始化该远程库



3、将本地库的内容push到远程库中

显示该仓库的http地址：



在git本地保存该地址(以后在本地用origin表示该地址)

**git remote add origin** [**https://github.com/DarkCS2019/try.git**](https://github.com/DarkCS2019/try.git)

4、将master分支上的文件推送到上述仓库（远程库）中

**git push origin master**

### 1.6.2 克隆项目

**git clone** [**https://github.com/DarkCS2019/try.git**](https://github.com/DarkCS2019/try.git)

效果

 完整的把远程库下载到本地

 创建 origin 远程地址别名、初始化本地库

### 1.6.3 新增团队成员

项目中：setting-collaborators-add people-出现一个网址

被邀请人打开网址加入团队，则可以push项目上去

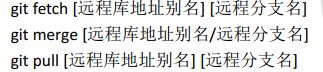
### 1.6.4 获取其他成员修改信息（pull = fetch + merge）

① 抓取远程库的master,把远程库的内容下载到本地

git fetch origin master

② 把远程库中的master合并到本地的master

git merge origin/master



### 1.6.5 解决冲突（只能在最新版本的基础上做修改）

如果不是基于 GitHub 远程库的最新版所做的修改，不能推送， 必须先拉取。拉取下来后如果进入冲突状态， 则按照“分支冲突解决” 操作解决即可

## 1.7 SSH登录

① 进入当前用户的家目录 $ cd ~

② 删除.ssh 目录 $ rm -rvf .ssh

③ 运行命令生成.ssh 密钥目录

$ ssh-keygen -t rsa -C atguigu2018ybuq@aliyun.com

[注意： 这里-C 这个参数是大写的 C]

④ 进入.ssh 目录查看文件列表

$ cd .ssh $ ls -lF

⑤ 查看 id\_rsa.pub 文件内容 $ cat id\_rsa.pub

⑥ 复制 id\_rsa.pub 文件内容， 登录 GitHub， 点击用户头像

→Settings→SSH and GPG keys

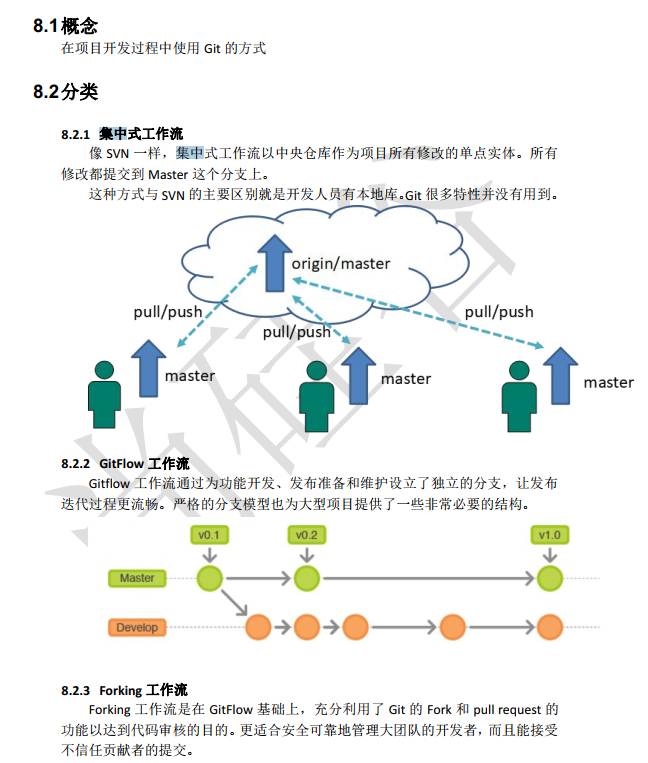
 New SSH Key、输入复制的密钥信息

⑦ 回到 Git bash 创建远程地址别名（用origin\_ssh代替地址）

git remote add origin\_ssh [git@github.com:atguigu2018ybuq/huashan](mailto:git@github.com:atguigu2018ybuq/huashan).git

 ⑧ 推送文件进行测试

## 1.8 Git工作流



## 1.9 总结

### 1.9.1 上传项目到GitHub

① 到项目文件夹中初始化项目仓库 git init

② 将文件添加至暂存区 git add Git命令行操作.docx

③ 将文件添加至本地库 git commit -m "note" Git命令行操作.docx

④ 在GitHub中创建远程库 主页右上角“+”-> new repository -> 填写仓库名称

⑤ 获得仓库的HTTP地址，并复制

⑥ 在git本地保存该地址(以后在本地用origin表示该地址)

git remote add origin <https://github.com/DarkCS2019/Git_learning.git>

⑦ 将master分支上的文件推送到上述仓库（远程库）中

git push origin master

### 1.9.2 克隆别人的项目

① 到项目文件夹中初始化项目仓库 git init

② 在GitHub上拿到地址

③ git clone <https://github.com/DarkCS2019/Git_learning.git>