

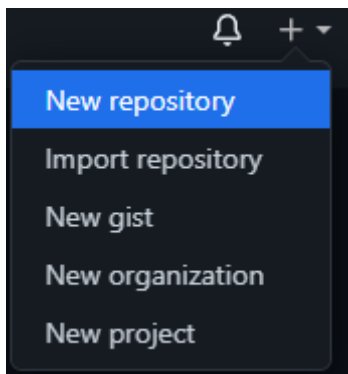
# git命令

---

## 1.git创建仓库和remote

---

第一步：再github.com页面找到+号



第二步：

# 创建新存储库

存储库包含所有项目文件，包括修订历史记录。已经在其他地方拥有项目存储库？[导入存储库](#)。

所有者 \*



element-ui

存储库名称 \*

/ note



仓库名称

伟大的存储库名称简短而令人难忘。需要灵感？[理想八花西兰花](#)怎么样？

描述 (可选)

存储自己记录的笔记



公共

互联网上的任何人都可以看到这个存储库。您可以选择谁可以提交。



私人

您可以选择谁可以查看并提交到此存储库。

使用以下命令初始化此存储库：

如果要导入现有存储库，请跳过此步骤。

☒ 添加自述文件

在这里，您可以为项目编写详细的描述。[了解更多信息](#)。

添加 .gitignore

从模板列表中选择不跟踪的文件。[了解更多信息](#)。

.gitignore template: 没有

选择许可证

许可证告诉其他人他们可以 and 不能用你的代码做什么。[了解更多信息](#)。

许可证: 没有

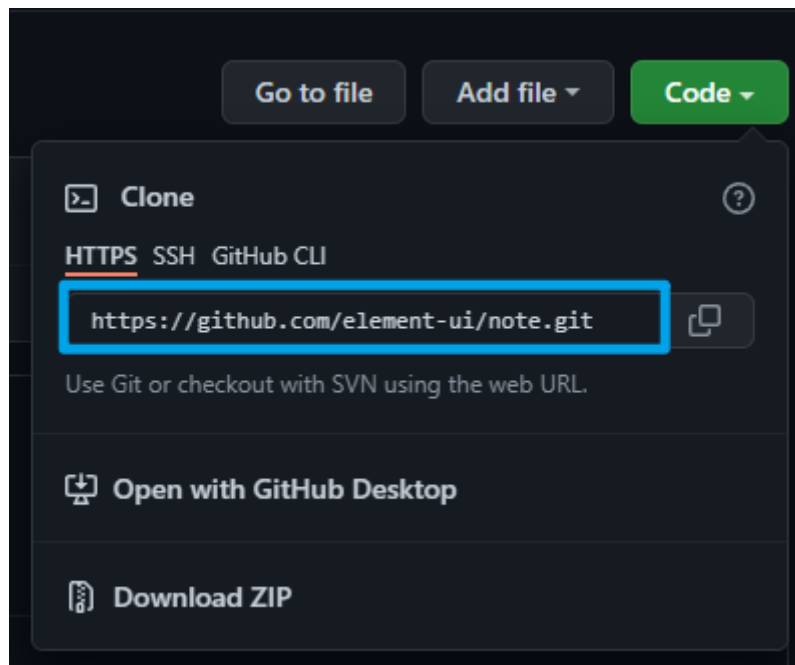
这将设置 主要 作为默认分支。更改设置中的默认名称。



您正在个人帐户中创建公共存储库。

创建存储库

第三步：复制仓库的地址



第四步: remote

```
1 | git remote add 名字 仓库地址
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git remote add giteenote https://gitee.com/cheng-chuyi/note.git
```

## 2.将资料推送到远程库中

```
1 | #查看状态
2 | git status
3 |
4 | #暂存区
5 | git add .
6 |
7 | #提交到本地仓库
8 | git commit -m "说明"
9 |
10 | #查看分支
11 | git branch
12 |
13 | #切换分支
14 | git checkout 分支名
15 |
16 | #合并分支
17 | git merge 分支名, #这里的分支名是要合并在被选中分支名上
18 |
19 | #查看当前所有远程地址别名
20 | git remote -v
21 |
22 | #提交到远程库
23 | git push -u remote名称 分支名
```

## 3.github的信息

---

### 3.1 注册邮箱

[github@ccy.com](mailto:github@ccy.com)

[2967911782@qq.com](mailto:2967911782@qq.com)

### 3.2 账号

element-ui

### 3.3 密码

15119402282qq

### 3.4 token值

ghp\_pWv6eBbMY8IUig8B3pUmnYQGawI8UU02LqtL

## 4.gitee的信息

---

### 4.1 注册邮箱

[2967911782@qq.com](mailto:2967911782@qq.com)

### 4.2 账号

18476765783

### 4.3 密码

15119402282qq

## 5.遇到的问题

---

### 5.1 git status

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

解决办法:

```
1 | git init # 初始化仓库
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data
$ git init
Initialized empty Git repository in D:/data/data/.git/
```

## 5.2 git commit

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git commit -m "第一次提交笔记"
Author identity unknown

*** Please tell me who you are.

Run

    git config --global user.email "you@example.com"
    git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got '陈楚义@LAPTOP-JBBD2536.(none)')
```

解决办法:

```
1 | git config --global user.email "邮箱"
2 | git config --global user.name "账号"
```

## 5.3 git push

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git push -u origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

解决办法:

```
1 | git remote add remote名字 远程仓库地址
2 | git push -u remote名字 分支名
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git remote add note https://github.com/element-ui/note.git~
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git push -u note master
fatal: unable to access 'https://github.com/element-ui/note.git~/: The requested
URL returned error: 400
```

解决办法:

```
1 | 重新设置远程地址名
2 | git remote add remote名字 远程仓库地址
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git remote add origin https://github.com/element-ui/note.git
```

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/data (master)
$ git push -u origin master
warning: ----- SECURITY WARNING -----
warning: | TLS certificate verification has been disabled! |
warning: -----
warning: HTTPS connections may not be secure. See https://aka.ms/gcm/tlsverify f
or more information.
```

```
F:\Vue项目实战\my_project_01>git push -u github rights
Username for 'https://github.com': 2967911782@qq.com
Password for 'https://2967911782@qq.com@github.com': 密码失效了, 用token替代密码
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: unable to access 'https://github.com/element-ui/my_github_project_01.git/': The requested URL returned error: 403
```

采用https方式进行上传代码:

这里是引用

Username for 'https://gitee.com': 自己的邮箱

Password for 'https://自己的邮箱@gitee.com': 密码{密码填了不提示, 保护隐私而已}

fatal: Authentication failed for 'https://gitee.com/???g/experience.git/'{账户或密码错误}

解决办法:

- 1 密码用token值登录就行了
- 2 token值: ghp\_pwv6eBbMY8IUig8B3pumnyQGaw18UU02Lqtl

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/dddd (master)
$ git push gitee master
To https://gitee.com/chen-chuyi/note.git
! [rejected] master -> master (fetch first)
error: failed to push some refs to 'https://gitee.com/chen-chuyi/note.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

解决办法:

- 1 # 将不同的资料推送到同一个仓库中时, 需要先拉取下来
- 2 git pull 远程地址名 分支名
- 3 # 再进行推送
- 4 git push 远程地址名 分支名

```
陈楚义@LAPTOP-JBBD2536 MINGW64 /d/data/dddd (master)
$ git pull giteenote master
remote: Enumerating objects: 190, done.
remote: Counting objects: 100% (190/190), done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 190 (delta 1), reused 162 (delta 0), pack-reused 0
```

## 6.git remote 命令

- 1 git remote add [shortname] [url] # 创建远程地址名
- 2 git remote rm name # 删除远程仓库地址名
- 3 git remote rename old\_name new\_name # 修改远程地址名

## 7. git clone

---

```
1 | git clone git://github.com/schacon/grit.git mygrit
```

## 8. git push

---

```
1 | git push <远程主机名> <本地分支名> # 推送到远程仓库中  
2 | git push --force origin master # 强行推送
```

## 9. git pull

---

```
1 | # 将不同的资料推送到同一个仓库中时，需要先拉取下来  
2 | git pull 远程地址名 分支名  
3 | # 再进行推送  
4 | git push 远程地址名 分支名
```

## 10. 分支管理

---

创建分支命令：

```
1 | git branch (branchname)
```

切换分支命令：

```
1 | git checkout (branchname)
```

合并分支：

```
1 | git merge 分支名, #这里的分支名是要合并在被选中分支名上
```

## 快速查找命令符

---

# Git 常用命令速查表

master :默认开发分支      Head :默认开发分支  
origin :默认远程版本库      Head^ :Head 的父提交

## 创建版本库

```
$ git clone <url>          #克隆远程版本库
$ git init                 #初始化本地版本库
```

## 修改和提交

```
$ git status              #查看状态
$ git diff                #查看变更内容
$ git add .               #跟踪所有改动过的文件
$ git add <file>          #跟踪指定的文件
$ git mv <old> <new>      #文件改名
$ git rm <file>           #删除文件
$ git rm --cached <file>  #停止跟踪文件但不删除
$ git commit -m "commit message"
                             #提交所有更新过的文件
$ git commit --amend      #修改最后一次提交
```

## 查看提交历史

```
$ git log                 #查看提交历史
$ git log -p <file>       #查看指定文件的提交历史
$ git blame <file>        #以列表方式查看指定文件的提交历史
```

## 撤销

```
$ git reset --hard HEAD   #撤销工作目录中所有未提交文件的修改内容
$ git checkout HEAD <file> #撤销指定的未提交文件的修改内容
$ git revert <commit>     #撤销指定的提交
```

## 分支与标签

```
$ git branch              #显示所有本地分支
$ git checkout <branch/tag> #切换到指定分支或标签
$ git branch <new-branch> #创建新分支
$ git branch -d <branch>  #删除本地分支
$ git tag                 #列出所有本地标签
$ git tag <tagname>       #基于最新提交创建标签
$ git tag -d <tagname>    #删除标签
```

## 合并与衍合

```
$ git merge <branch>      #合并指定分支到当前分支
$ git rebase <branch>     #衍合指定分支到当前分支
```

## 远程操作

```
$ git remote -v           #查看远程版本库信息
$ git remote show <remote> #查看指定远程版本库信息
$ git remote add <remote> <url>
                             #添加远程版本库
$ git fetch <remote>       #从远程库获取代码
$ git pull <remote> <branch> #下载代码及快速合并
$ git push <remote> <branch> #上传代码及快速合并
$ git push <remote> :<branch/tag-name>
                             #删除远程分支或标签
$ git push --tags          #上传所有标签
```

## -----师兄整理的git笔记-----

### 基础linux 命令

```
1 clear : 清除屏幕
2
3 echo 'test content'**: 往控制台输出信息 **echo 'test content' > test.txt ll : 将
  当前目录下的 子文件&子目录平铺在控制台
4
5 find 目录名: 将对应目录下的子孙文件&子孙目录平铺在控制台
6
7 find 目录名 -type f : 将对应目录下的文件平铺在控制台
8
9 rm 文件名 : 删除文件
10
11 mv 源文件 重命名文件: 重命名
12
13 cat 文件的url : 查看对应文件的内容
14
15 #### vim 文件的 url(在英文模式下)
16
17 按 i 进插入模式 进行文件的编辑按 esc 键&按:键 进行命令的执行
18
19 q! 强制退出(不保存)
20
21 wq 保存退出
22
23 set nu 设置行号
```



```
1 git对象
2     git hash-object -w fileurl : 生成一个key(hash值):val(压缩后的文件内容)键值对存
   到.git/objects
3 tree对象
4     git update-index --add --cacheinfo 100644 hash test.txt : 往暂存区添加一条
   记录(让git对象 对应 上文件名)存到.git/index
5     git write-tree : 生成树对象存到.git/objects
6 commit对象
7     echo 'first commit' | git commit-tree treehash : 生成一个提交对象存
   到.git/objects
8 对以上对象的查询
9     git cat-file -p hash          : 拿对应对象的内容
10    git cat-file -t hash          : 拿对应对象的类型
```

## 查看暂存区

```
1 git ls-files -s
```

安装

```
1 git --version
```

## 重点①初始化仓库

```
1 git init 初始化在所在的文件，作为仓库！
```

## 重点②初始化配置

```
1     git config user.name tom_pro (仓库级别)
2     git config user.email goodMorning_pro@atguigu.com
3
4 ②设置签名
5     git config --global user.name 名字
6     git config --global user.email 邮箱
7     git config --list
8 ③查看签名配置
9     cat ~/.gitconfig
```

## 重点④远程协作的基本流程

```
1 第一步：项目经理创建一个空的远程仓库
2 第二步：
3 第三步：为远程仓库配别名 配完用户名 邮箱
4 第四步：在本地仓库中初始化代码 提交代码
5 第五步：推送
6 第六步：邀请成员
7 第七步：成员克隆远程仓库
8 第八步：成员做出修改
9 第九步：成员推送自己的修改
10 第十步：项目经理拉取成员的修改
```

## 重点⑤创建远程库地址别名

```
1  git remote -v 查看当前所有远程地址别名、
2
3  git remote add [别名] [远程地址]
4
5  如: git remote add bieming https://github.com/xiaobai975/JUNSHI.git
6  推送, 一般先拉取最新的文件在修改提交push
7
8  git push [别名] [分支名]: [分支名]
9
10 如: git push junshi_ssh master:master
11 克隆
12 git clone [远程地址]
13 stat
14 拉取!
15 git pull + 用户别名 +分支名
16
17 若仓库有东西先
18 git pull --rebase junshi_ssh mastergit
```

## 重点⑥SSH登陆

```
1  进入当前用户的家目录
2  $ cd ~
3  删除.ssh 目录
4  $ rm -rvf .ssh
5  运行命令生成.ssh 密钥目录
6  $ ssh-keygen -t rsa -C junshi975@aliyun.com
7  [注意: 这里-C 这个参数是大写的 C]
8  进入.ssh 目录查看文件列表
9  $ cd .ssh
10 $ ls -lF
11 查看 id_rsa.pub 文件内容
12 $ cat id_rsa.pub
13 复制 id_rsa.pub 文件内容, 登录 GitHub, 点击用户头像→Settings→SSH and GPG keys
14 New SSH Key
15 输入复制的密钥信息
16 回到 Git bash 创建远程地址别名
17 git remote add origin_ssh git@github.com:atguigu2018ybuq/huashan.git
18 推送文件进行测试
```

```
1  拉取! git pull + 用户别名 +分支名
2  推送, 一般先拉取最新的文件在修改提交push
3
4  git push [别名] [分支名]: [分支名]
5  合并分支: 先切换到需要合并的分支里, git merge + 在修改过东西的分支名
6  合并分支 : git merge branchname
7      快进合并 --> 不会产生冲突
8      典型合并 --> 有机会产生冲突
9      解决冲突 --> 打开冲突的文件 进行修改 add commit
10
11 查看分支列表 : git branch
```

```
12 查看合并到当前分支的分支列表: git branch --merged
13    一旦出现在这个列表中 就应该删除
14 查看没有合并到当前分支的分支列表: git branch --no-merged
15    一旦出现在这个列表中 就应该观察一下是否需要合并
16 创建分支          : git branch branchname
17 切换分支          : git checkout branchname
18 创建&切换分支      : git checkout -b branchname
19 版本穿梭(时光机) : git branch branchname commitHash
20 普通删除分支      : git branch -d branchname
21 强制删除分支      : git branch -D branchname
```

## 做跟踪

```
1 克隆仓库时 会自动为master做跟踪
2 本地没有分支
3    git checkout --track 远程跟踪分支(remote/分支名)
4 本地已经创建了分支
5    git branch -u 远程跟踪分支(remote/分支名)
```

## C(新增)

```
1 在工作目录中新增文件
2 git status
3 git add ./
4 git commit -m "msg"
```

## U(修改)

```
1 在工作目录中修改文件
2 git status
3 git add ./
4 git commit -m "msg"
```

## D(删除 & 重命名)

```
1    git rm 要删除的文件      git mv 老文件 新文件
2    git status                git status
3    git commit -m "msg"      git commit -m "msg"
```

## R(查询)

```
1    git status    : 查看工作目录中文件的状态(已跟踪(已提交 已暂存 已修改) 未跟踪)
2    git diff      : 查看未暂存的修改
3    git diff --cache : 查看未提交的暂存
4    git log --oneline : 查看提交记录
```

# 分支

```
1 分支的本质其实就是一个提交对象!!!
2 HEAD:
3     是一个指针 它默认指向master分支 切换分支时其实就是让HEAD指向不同的分支
4     每次有新的提交时 HEAD都会带着当前指向的分支 一起往前移动
5 git log --oneline --decorate --graph --all : 查看整个项目的分支图
6 git branch : 查看分支列表
7 git branch -v: 查看分支指向的最新的提交
8 git branch name : 在当前提交对象上创建新的分支
9 git branch name commithash: 在指定的提交对象上创建新的分支
10 git checkout name : 切换分支
11 git branch -d name : 删除空的分支 删除已经被合并的分支
12 git branch -D name : 强制删除分支
```

## git分支本质

```
1 分支本质是一个提交对象,所有的分支都会有机会被HEAD所引用(HEAD一个时刻只会指向一个分支)
2 当我们有新的提交的时候 HEAD会携带当前持有的分支往前移动
```

## 重点④git分支命令

```
1 创建分支 : git branch branchname
2 切换分支 : git checkout branchname
3 创建&切换分支 : git checkout -b branchname
4 版本穿梭(时光机) : git branch branchname commithash
5 普通删除分支 : git branch -d branchname
6 强制删除分支 : git branch -D branchname
7
8 合并分支: 先切换到需要合并的分支里, git merge + 在修改过东西的分支名
9 合并分支 : git merge branchname
10 快进合并 --> 不会产生冲突
11 典型合并 --> 有机会产生冲突
12 解决冲突 --> 打开冲突的文件 进行修改 add commit
13
14 查看分支列表 : git branch
15 查看合并到当前分支的分支列表: git branch --merged
16 一旦出现在这个列表中 就应该删除
17 查看没有合并到当前分支的分支列表: git branch --no-merged
18 一旦出现在这个列表中 就应该观察一下是否需要合并
```

## ★git分支的注意事项

```
1 在切换的时候 一定要保证当前分支是干净的!!!
2 允许切换分支:
3     分支上所有的内容处于 已提交状态
4     (避免)分支上的内容是初始化创建 处于未跟踪状态
5     (避免)分支上的内容是初始化创建 第一次处于已暂存状态
6 不允许切分支:
7     分支上所有的内容处于 已修改状态 或 第二次以后的已暂存状态
8
9 在分支上的工作做到一半时 如果有切换分支的需求, 我们应该将现有的工作存储起来
10 git stash : 会将当前分支上的工作推到一个栈中
11 分支切换 进行其他工作 完成其他工作后 切回原分支
```

```
12 | git stash apply : 将栈顶的工作内容还原 但不让任何内容出栈
13 | git stash drop  : 取出栈顶的工作内容后 就应该将其删除(出栈)
14 | git stash pop   :      git stash apply + git stash drop
15 | git stash list  : 查看存储
```

## 后悔药

```
1 | 撤销工作目录的修改 : git checkout -- filename
2 | 撤销暂存区的修改   : git reset HEAD filename
3 | 撤销提交           : git commit --amend
```

## reset三部曲

```
1 | git reset --soft commithash ---> 用commithash的内容重置HEAD内容
2 | git reset [--mixed] commithash ---> 用commithash的内容重置HEAD内容 重置暂存区
3 | git reset --hard commithash ---> 用commithash的内容重置HEAD内容 重置暂存区 重置
   | 工作目录
```

## 路径reset

```
1 | 所有的路径reset都要省略第一步!!!
2 | 第一步是重置HEAD内容 我们知道HEAD本质指向一个分支 分支的本质是一个提交对象
3 | 提交对象 指向一个树对象 树对象又很有可能指向多个git对象 一个git对象代表一个文件!!!
4 | HEAD可以代表一系列文件的状态!!!!
5 | git reset [--mixed] commithash filename
6 | 用commithash中filename的内容重置暂存区
```

## checkout深入理解

```
1 | git checkout branchname 跟 git reset --hard commithash特别像
2 | 共同点
3 | 都需要重置 HEAD 暂存区 工作目录
4 | 区别
5 | checkout对工作目录是安全的 reset --hard是强制覆盖
6 | checkout动HEAD时不会带着分支走而是切换分支
7 | reset --hard时是带着分支走
8 |
9 | checkout + 路径
10 | git checkout commithash filename
11 | 重置暂存区
12 | 重置工作目录
13 | git checkout -- filename
14 | 重置工作目录
```

eslint

```
1 | js代码的检查工具
2 | 下载: npm i eslint -D
3 | 使用:
4 |     生成配置文件 npx eslint --init
5 |     检查js文件    npx eslint 目录名
6 |     命中的规则:
7 |         字符串必须使用单引号
8 |         语句结尾不能有分号
9 |         文件的最后必须要有换行
```

## eslint结合git

```
1 | husky: 哈士奇, 为Git仓库设置钩子程序
2 | 使用
3 |     在仓库初始化完毕之后 再去安装哈士奇
4 |     在package.json文件写配置
5 |         "husky": {
6 |             "hooks": {
7 |                 "pre-commit": "npm run lint"
8 |                 //在git commit之前一定要通过npm run lint的检查
9 |                 // 只有npm run lint不报错时 commit才能真正的运行
10 |             }
11 |         }
```

三个必须懂得概念

```
1 | 本地分支
2 | 远程跟踪分支(remote/分支名)
3 | 远程分支
```

## 推送

```
1 | git push + 用户别名 +分支名
```

## 拉取

```
1 | git pull + 用户别名 +分支名
```

## pull request

```
1 | 让第三方人员参与到项目中 fork
```

## 使用频率最高的五个命令

```
1 | git status
2 | git add
3 | git commit
4 | git push
5 | git pull
```









