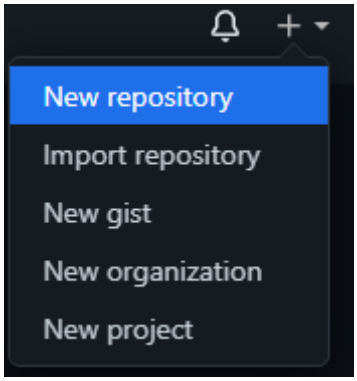
git命令

1.git创建仓库和remote

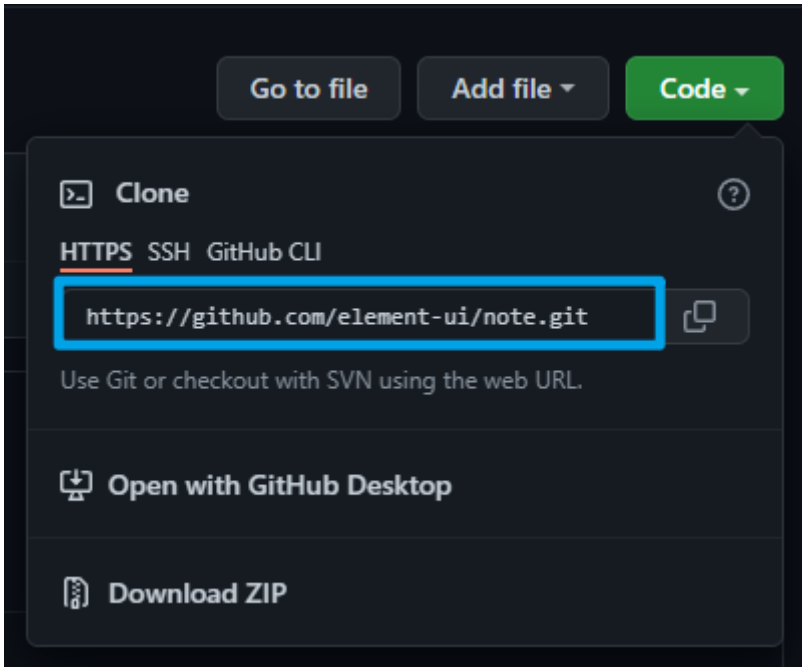
第一步：再github.com页面找到+号



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。
仓库名称
所有者 *
存储库名称 *
element-ui
/
note
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和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

```
git remote add 名字 仓库地址
```

2.将资料推送到远程库中

```
#查看状态
git status

#暂存区
git add .

#提交到本地仓库
git commit -m "说明"

#查看分支
git branch

#切换分支
git chekout 分支名

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

#查看当前所有远程地址别名
git remote -v

#提交到远程库
git push -u remote名称 分支名
```

3.github的信息

3.1 注册邮箱

github@ccy.com

2967911782@qq.com

3.2 账号

element-ui

3.3 密码

15119402282qq

3.4 token值

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

4.gitee的信息

4.1 注册邮箱

2967911782@qq.com

4.2 账号

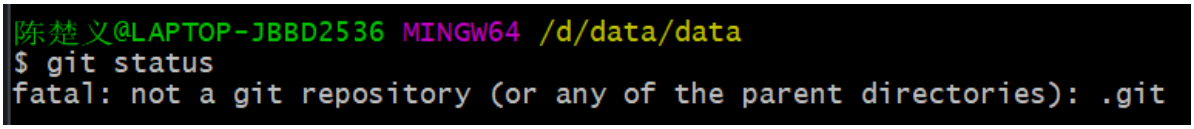
18476765783

4.3 密码

15119402282qq

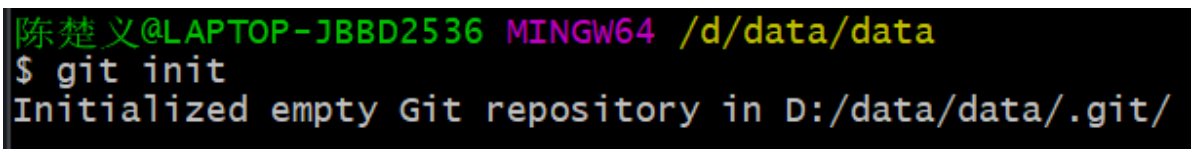
5.遇到的问题

5.1 git status



解决办法:

```
git init # 初始化仓库
```



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:

```
git config --global user.email "邮箱"
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:

```
git remote add remote名字 远程仓库地址
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:

```
重新设置远程地址名
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':
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
```

密码失效了，用token替代密码

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_pWv6eBbMY8IUig8B3pUmnYQGaWl8UUO2LqtL
```

```
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 一进去git界面就先执行git pull命令，而且要上传的资料需与上一次上传到远程库中的资料的路径要一致
2 # 将不同的资料推送到同一个仓库中时，需要先拉取下来
3 git pull 远程地址名 分支名
4 # 再进行推送
5 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

```
git remote add [shortname] [url] # 创建远程地址名
git remote rm name  # 删除远程仓库地址名
git remote rename old_name new_name  # 修改远程地址名
```

7. git clone

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

8. git push

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

9. git pull

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

10. 分支管理

创建分支命令：

```
git branch (branchname)
```

切换分支命令:

```
git checkout (branchname)
```

合并分支：

```
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

-----师兄整理的git笔记------

基础linux 命令

```
clear ：清除屏幕

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

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

find 目录名 -type f ：将对应目录下的文件平铺在控制台

rm   文件名 ： 删除文件

mv 源文件 重命名文件: 重命名

cat 文件的url : 查看对应文件的内容

#### vim 文件的 url(在英文模式下)

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

q!   强制退出（不保存）

wq  保存退出

set nu 设置行号
```

```
git对象
    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
tree对象
    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
    git write-tree : 生成树对象存到.git/objects
commit对象
    echo 'first commit' | git commit-tree treehash : 生成一个提交对象存到.git/objects
对以上对象的查询
    git cat-file -p hash       : 拿对应对象的内容
    git cat-file -t hash       : 拿对应对象的类型
```

查看暂存区

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

安装

```
git --version
```

重点①初始化仓库

```
git init 初始化在所在的文件，作为仓库！
```

重点②初始化配置

```
    git config user.name tom_pro（仓库级别）
    git config user.email goodMorning_pro@atguigu.com

②设置签名
git config --global user.name 名字
git config --global user.email 邮箱
git config --list
③查看签名配置
cat ~/.gitconfig
```

重点④远程协作的基本流程

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

```
git remote -v 查看当前所有远程地址别名、

git remote add [别名] [远程地址]

如：git remote add bieming https://github.com/xiaobai975/JUNSHI.git
推送，一般先拉取最新的文件在修改提交push

git push [别名] [分支名]：[分支名]

如：git push junshi_ssh master:master
克隆
git clone [远程地址]
stat
拉取!
git pull + 用户别名 +分支名

若仓库有东西先
git pull --rebase junshi_ssh mastergit
```

重点⑥SSH登陆

```
进入当前用户的家目录
$ cd ~
删除.ssh 目录
$ rm -rvf .ssh
运行命令生成.ssh 密钥目录
$ ssh-keygen -t rsa -C junshi975@aliyun.com
[注意：这里-C 这个参数是大写的 C]
进入.ssh 目录查看文件列表
$ cd .ssh
$ ls -lF
查看 id_rsa.pub 文件内容
$ cat id_rsa.pub
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
回到 Git bash 创建远程地址别名
git remote add origin_ssh git@github.com:atguigu2018ybuq/huashan.git
推送文件进行测试
```

```
拉取！ git pull + 用户别名 +分支名
推送，一般先拉取最新的文件在修改提交push

git push [别名] [分支名]：[分支名]
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
合并分支          : git merge branchname
    快进合并 --> 不会产生冲突
    典型合并 --> 有机会产生冲突
    解决冲突 --> 打开冲突的文件 进行修改 add commit

查看分支列表 : git branch
```

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

做跟踪

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

C(新增)

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

U(修改)

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

D(删除 & 重命名)

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

R(查询)

```
   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)
   git  diff     :  查看未暂存的修改
   git  diff --cache : 查看未提交的暂存
   git  log --oneline : 查看提交记录
```

分支

```
分支的本质其实就是一个提交对象!!!
HEAD:
    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
    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
git  log --oneline --decorate --graph --all : 查看整个项目的分支图
git branch : 查看分支列表
git branch -v: 查看分支指向的最新的提交
git branch name : 在当前提交对象上创建新的分支
git branch name commithash: 在指定的提交对象上创建新的分支
git checkout name :      切换分支
git branch -d name : 删除空的分支 删除已经被合并的分支
git branch -D name : 强制删除分支
```

git分支本质

```
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)
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

```
创建分支             : git branch branchname
切换分支            : git checkout  branchname
创建&切换分支      : git checkout -b branchname
版本穿梭(时光机) :  git branch branchname commitHash
普通删除分支       : git  branch -d branchname
强制删除分支       : git  branch -D branchname

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
合并分支          : git merge branchname
    快进合并 --> 不会产生冲突
    典型合并 --> 有机会产生冲突
    解决冲突 --> 打开冲突的文件 进行修改 add commit

查看分支列表 : git branch
查看合并到当前分支的分支列表: git branch --merged
    一旦出现在这个列表中 就应该删除
查看没有合并到当前分支的分支列表: git branch --no-merged
    一旦出现在这个列表中 就应该观察一下是否需要合并
```

★git分支的注意点

```
在切换的时候 一定要保证当前分支是干净的!!!
    允许切换分支:
        分支上所有的内容处于 已提交状态
        (避免)分支上的内容是初始化创建 处于未跟踪状态
        (避免)分支上的内容是初始化创建 第一次处于已暂存状态
    不允许切分支:
         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

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
    git stash : 会将当前分支上的工作推到一个栈中
    分支切换   进行其他工作 完成其他工作后 切回原分支
```

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

后悔药

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

reset三部曲

```
git reset --soft commithash    ---> 用commithash的内容重置HEAD内容
git reset [--mixed] commithash ---> 用commithash的内容重置HEAD内容 重置暂存区
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

```
所有的路径reset都要省略第一步!!!
    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
    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!!!
    HEAD可以代表一系列文件的状态!!!!
git reset [--mixed] commithash filename
     用commithash中filename的内容重置暂存区
```

checkout深入理解

```
git   checkout brancname  跟   git reset --hard commithash特别像
    共同点
        都需要重置 HEAD   暂存区   工作目录
    区别
         checkout对工作目录是安全的     reset --hard是强制覆盖
         checkout动HEAD时不会带着分支走而是切换分支
         reset --hard时是带着分支走

checkout + 路径
      git checkout commithash  filename
           重置暂存区
           重置工作目录
      git checkout -- filename
          重置工作目录
```

eslint

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

eslint结合git

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

三个必须懂得概念

```
本地分支
远程跟踪分支(remote/分支名)
远程分支
```

推送

```
git push + 用户别名 +分支名
```

拉取

```
git pull + 用户别名 +分支名
```

pull request

```
让第三方人员参与到项目中 fork
```

使用频率最高的五个命令

```
git status
git add
git commit
git push
git pull
```