

# git命令

---

## 配置用户名和邮箱

---

```
1 git config --global user.name "itheima"
2 git config --global user.email "itheima@itcast.cn"
```

- 

## 检查配置信息

---

```
1 # 查看所有的全局配置项
2 git config --list --global
3 # 查看指定的全局配置项
4 git config user.name
5 git config user.email
```

- 

## 在现有目录初始化仓库

---

如果自己有一个尚未进行版本控制的项目目录，想要用 Git 来控制它，需要执行如下两个步骤：

- ① 在项目目录中，通过鼠标右键打开“Git Bash”  
② 执行 `git init` 命令将当前的目录转化为 Git 仓库

## 显示文件状态

---

```
1 # 以精简的方式显示文件状态
2 git status -s
3 git status --short
```

- 

## 跟踪新文件

---

```
1 git add index.html
```

## 提交更新

---

- ```
1 git commit -m "新建了index.html文件"
```

## 跟踪多个新文件

---

- ```
1 git add .
```

## 取消暂存的文件

---

- ```
1 git reset HEAD 要移除的文件名称
```

## 跳过使用暂存区

---

- ```
1 git commit -a -m "描述消息"
```

## 移除文件

---

- ```
1 # 从 Git 仓库和工作区中同时移除 index.js 文件
2 git rm -f index.js
3 # 只从 Git 仓库中移除 index.css，但保留工作区中的 index.css 文件
4 git rm --cached index.css
```

## 忽略文件

---

```
1 # 忽略所有的 .a 文件
2 *.a
3
4 # 但跟踪所有的 lib.a, 即便你在前面忽略了 .a 文件
5 !lib.a
6
7 # 只忽略当前目录下的 TODO 文件, 而不忽略 subdir/TODO
8 /TODO
9
10 # 忽略任何目录下名为 build 的文件夹
11 build/
12
13 # 忽略 doc/notes.txt, 但不忽略 doc/server/arch.txt
14 doc/*.txt
15
16 # 忽略 doc/ 目录及其所有子目录下的 .pdf 文件
17 doc/**/*.*pdf
```

## 提交到Github

---

### HTTPS

---

## 4. 基于 HTTPS 将本地仓库上传到 Github

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** SSH `https://github.com/teacher-liu/project_02.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**1 本地没有现成的 Git 仓库**  
...or create a new repository on the command line

```
echo "# project 02" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/teacher-liu/project_02.git
git push -u origin master
```

1. 使用终端命令创建 README.md 文档，并写入初始内容为 #project 02  
2. 初始化本地 Git 仓库，并将文件的修改提交到本地的 Git 仓库中  
3. 将本地仓库和远程仓库进行关联，并把远程仓库命名为 origin  
4. 将本地仓库中的内容推送到远程的 origin 仓库中

**2 本地有现成的 Git 仓库**  
...or push an existing repository from the command line

```
git remote add origin https://github.com/teacher-liu/project_02.git
git push -u origin master
```

1. 将本地仓库和远程仓库进行关联，并把远程仓库命名为 origin  
2. 将本地仓库中的内容推送到远程的 origin 仓库中

## SSH

### 6. 生成 SSH key

- ① 打开 Git Bash
- ② 粘贴如下的命令，并将 `your_email@example.com` 替换为注册 Github 账号时填写的邮箱：
  - `ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`
- ③ 连续敲击 3 次回车，即可在 `C:\Users\用户名文件夹\.ssh` 目录中生成 `id_rsa` 和 `id_rsa.pub` 两个文件

### 7. 配置 SSH key

- ① 使用记事本打开 `id_rsa.pub` 文件，复制里面的文本内容
- ② 在浏览器中登录 Github，点击头像 -> Settings -> SSH and GPG Keys -> New SSH key
- ③ 将 `id_rsa.pub` 文件中的内容，粘贴到 Key 对应的文本框中
- ④ 在 Title 文本框中任意填写一个名称，来标识这个 Key 从何而来

## 8. 检测 Github 的 SSH key 是否配置成功

打开 Git Bash，输入如下的命令并回车执行：

```
1 ssh -T git@github.com
```

上述的命令执行成功后，可能会看到如下的提示消息：

```
1 The authenticity of host 'github.com (IP ADDRESS)' can't be established.
2 RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
3 Are you sure you want to continue connecting (yes/no)?
```

输入 yes 之后，如果能看到类似于下面的提示消息，证明 SSH key 已经配置成功了：

```
1 Hi username! You've successfully authenticated, but GitHub does not
2 provide shell access.
```

## 9. 基于 SSH 将本地仓库上传到 Github

Quick setup — if you've <sup>①</sup> done this kind of thing before

☐ Set up in Desktop or ☐ HTTPS ☒ SSH git@github.com:teacher-liu/project\_03.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# project_03" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:teacher-liu/project_03.git
git push -u origin master
```

将本地现成的仓库推送到 Github

...or push an existing repository from the command line

```
git remote add origin git@github.com:teacher-liu/project_03.git
git push -u origin master
```

② 1. 将本地仓库和远程仓库进行关联，并把远程仓库命名为 origin  
2. 将本地仓库中的内容推送到远程的 origin 仓库中

## 克隆

```
1 git clone 远程仓库的地址
```

## 查看分支列表

- 1 `git branch`

## 创建分支

---

- 1 `git branch` 分支名称

## 切换分支

---

- 1 `git checkout login`

## 分支快速创建与切换

---

- 1 # `-b` 表示创建一个新分支
- 2 # `checkout` 表示切换到刚才新建的分支上
- 3 `git checkout -b` 分支名称

## 合并分支

---

- 1 # 1. 切换到 `master` 分支
- 2 `git checkout master`
- 3 # 2. 在 `master` 分支上运行 `git merge` 命令，将 `login` 分支的代码合并到 `master` 分支
- 4 `git merge login`

## 删除分支

---

- 1 `git branch -d` 分支名称

## 遇到冲突时合并分支

---

```
1 # 假设：在把 reg 分支合并到 master 分支期间，代码发生了冲突
2 git checkout master
3 git merge reg
4
5 # 打开包含冲突的文件，手动解决冲突之后，再执行如下的命令
6 git add .
7 git commit -m "解决了分支合并冲突的问题"
```

## 将本地分支推送给远程仓库

```
1 # -u 表示把本地分支和远程分支进行关联，只在第一次推送的时候需要带 -u 参数
2 git push -u 远程仓库的别名 本地分支名称:远程分支名称
3
4 # 实际案例：
5 git push -u origin payment:pay
6
7 # 如果希望远程分支的名称和本地分支名称保持一致，可以对命令进行简化：
8 git push -u origin payment
```

## 查看远程仓库分支列表

```
1 git remote show 远程仓库名称
```

## 跟踪分支

```
1 # 从远程仓库中，把对应的远程分支下载到本地仓库，保持本地分支和远程分支名称相同
2 git checkout 远程分支的名称
3 # 示例：
4 git checkout pay
5
6 # 从远程仓库中，把对应的远程分支下载到本地仓库，并把下载的本地分支进行重命名
7 git checkout -b 本地分支名称 远程仓库名称/远程分支名称
8 # 示例：
9 git checkout -b payment origin/pay
```

## 拉取远程分支的最新的代码

---

- ```
1 # 从远程仓库，拉取当前分支最新的代码，保持当前分支的代码和远程分支代码一致
2 git pull
```

## 删除远程分支

---

- ```
1 # 删除远程仓库中，指定名称的远程分支
2 git push 远程仓库名称 --delete 远程分支名称
3 # 示例:
4 git push origin --delete pay
```

## 总结

---



## ① 能够掌握 Git 中基本命令的使用

- git init
- git add .
- git commit -m "提交消息"
- git status 和 git status -s

## ② 能够使用 Github 创建和维护远程仓库

- 能够配置 Github 的 SSH 访问
- 能够将本地仓库上传到 Github

## ③ 能够掌握 Git 分支的基本使用

- git checkout -b 新分支名称
- git push -u origin 新分支名称
- git checkout 分支名称
- git branch