Git基础

1.初始设置

设置姓名和邮箱地址

```
$ git config --global user.name "userName"
$ git config --global user.email "userName@xxx.xxx"
```

提高命令输出可读性

```
$ git config --global color.ui auto
```

创建 SSH Key

```
$ ssh-keygen -t rsa -C "userName@xxx.xxx"
```

在 GitHub 添加公开密钥

```
$ cat ~/.ssh/id_rsa.pub   # 显示公开密钥，在 GitHub 页面输入即可
```

2.使用示例

clone 仓库

```
$ git clone ...
```

查看状态

```
$ git status
```

提交

```
$ git add yourFileName      # 将文件加入暂存区
$ git commit -m "Message"   # 提交
$ git push                  # push
```

查看提交日志

```
$ git log
```

```
admin@DESKTOP-7RH9HTC MINGW64 /e/code/learnGitHub/learnGit-GitHub (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        HelloWorld.cpp

nothing added to commit but untracked files present (use "git add" to track)

admin@DESKTOP-7RH9HTC MINGW64 /e/code/learnGitHub/learnGit-GitHub (main)
$ git add HelloWorld.cpp

admin@DESKTOP-7RH9HTC MINGW64 /e/code/learnGitHub/learnGit-GitHub (main)
$ git commit -m "Add hello world file by C++"
[main 8905b21] Add hello world file by C++
 1 file changed, 6 insertions(+)
 create mode 100644 HelloWorld.cpp

admin@DESKTOP-7RH9HTC MINGW64 /e/code/learnGitHub/learnGit-GitHub (main)
$ git log
commit 8905b214ffe12a6e1ba6a6598bea5207b106243e (HEAD -> main)
Author: UnpureRationalist <3295861919@qq.com>
Date:   Sat Feb 27 23:46:52 2021 +0800

    Add hello world file by C++

commit 66605df175d95f38682d3f4a95a7b8582aeaf01f (origin/main, origin/HEAD)
Author: arcsin2 <52957336+UnpureRationalist@users.noreply.github.com>
Date:   Sat Feb 27 23:34:16 2021 +0800

    Initial commit

admin@DESKTOP-7RH9HTC MINGW64 /e/code/learnGitHub/learnGit-GitHub (main)
$ git push
Warning: Permanently added the RSA host key for IP address '13.250.177.223' to the list of known hosts.
Enter passphrase for key '/c/Users/admin/.ssh/id_rsa':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 384 bytes | 384.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:UnpureRationalist/learnGit-GitHub.git
   66605df..8905b21  main -> main
```

3. 基本操作

初始化仓库

```
$ git init      # 生成 .git 文件夹，称之为“附属于该仓库的工作树”
```

查看仓库状态

```
$ git status
```

向暂存区添加文件

```
$ git add README.md
```

保存仓库的历史记录

```
$ git commit -m "First commit"      # 记录一行提交信息
```

```
$ git commit                        # 记录详细提交信息
```

查看提交日志

```
$ git log
$ git log --pretty=short            # 只显示提交信息的第一行
$ git log fileName/DirName          # 只显示指定文件、文件夹的日志
$ git log -p                        # 显示文件的改动
$ git log -p fileName               # 只显示指定文件提交日志和提交前后差别
```

查看更改前后的差别

```
$ git diff
```

查看工作树和最新提交的差别

```
$ git diff HEAD
```

4. 分支的操作

显示分支一览表

```
$ git branch
```

创建、切换分支

```
$ git checkout -b newBranchName     # 创建新分支 newBranchName 并切换， 相当于下面两条命令

$ git branch newBranchName          # 创建新分支
$ git checkout newBranchName        # 切换分支
```

切换回上一个分支

```
$ git checkout -
```

合并分支

```
$ git merge --no-ff newBranchName   # 合并分支，执行此命令会弹出 vim 窗口，输入“zz”退出
```

以图表形式查看分支

```
$ git log --graph
```

5. 更改提交的操作

回溯历史版本

```
$ git reset --hard 42810318e5cbbae39739cae41cc872eb12c5280c          # 最后字符串为目标时间节点的哈希值
```

查看当前仓库操作日志

```
$ git log           # 只能查看以当前状态为重点的历史日志
$ git reflog        # 查看当前仓库操作日志 一般用于恢复原先状态
```

修改上一条提交信息

```
$ git commit --amend        # 执行该命令后将启动 vim 编辑器
```

压缩历史

```
$ git rebase -i HEAD~n      # n 为一个正整数，表示将最新 n 个开始记录合并。此操作将打开编辑器 vim
```

6. 推送至远程仓库

添加至远程仓库

```
$ git remote add origin git@github.com:UnpureRationalist/git_tutorial.git
# origin 为标识符
```

推送至远程仓库

```
$ git push -u origin master
```

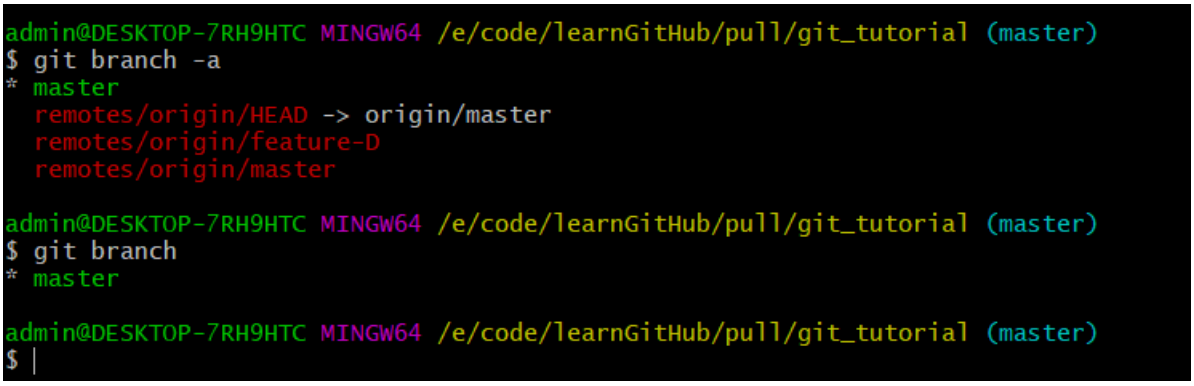
7. 从远程仓库获取

获取远程仓库

```
$ git clone git@github.com:UnpureRationalist/git_tutorial.git
```

查看当前分支信息

```
$ git branch -a     # -a 参数同时显示本地仓库和远程仓库的分支信息
```



获取远程的 feature-D 分支

```
$ git checkout -b feature-D origin/feature-D        # -b 参数的后面是本地仓库中新建分支的名称
# 上面指令指定了 origin/feature-D，就是说以名为 origin 的仓库（这里指 GitHub 端的仓库）的 feature-D 分支为来源，
# 在本地仓库中创建 feature-D 分支
```

向本地的 feature-D 分支提交更改

```
$ git commit -am "Add feature-D"
```

推送 feature-D 分支

```
$ git push
```

获取最新的远程仓库分支

```
$ git pull origin feature-D
```

8. 在线学习资料

- Pro Git
- LearnGitBranching
- tryGit