Github-了解开源相关的概念

1. 常见的5种开源许可协议

- BSD(Berkeley Software Distribution)
- Apache Licence 2.0
- **GPL**(GNU General Public Licence)
- LGPL(GNU Lesser General Public License)
- **MIT**(Massachusetts Institute of Technology,MIT)

GPL:具有传染性的一种开源协议，不允许修改后和衍生的代码作为闭源的商业软件和销售
使用GPL最著名的软件是Linux

MIT：是目前限制最少的一种协议，唯一的条件是：在修改之后的代码或者发行包中，必须包含原作者的许可信息
使用MIT的软件项目有：jQuery NOde.js

Github-远程仓库的使用

1. 远程仓库的两种调用方式

Github 上的远程仓库，有两种访问方式，分别是 HTTPS和SSH。他们的区别是：

- HTTPS：零配置；但是每次访问仓库时，需要重复输入Github账号才能访问成功
- SSH：需要进行额外的配置，但是配置成功之后，每次访问仓库时，不需要重复输入Github的账号和密码

2. 基于HTTPS将本地仓库上传到Github

第一次推送时使用以下代码：

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

之后推送可以使用：

```
git push
```

3. 生成SSH key

- 打开Git Bash
- 粘贴如下的命令，替换成自己注册时的邮箱

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- 连续敲击三次回车，即可在C:\users\用户名.ssh目录中生成id_rsa和id_rsa.pub两个文件

Git分支-本地分支操作

1. master主分支

在初始化本地Git仓库时，Git默认已经帮我们创建了一个名字叫做master的分支。通常我们把这个master分支叫做主分支

在实际工作中，master主分支的作用是：用来保存和记录整个项目已经完成的功能代码
因此，不允许程序员直接在master分支上修改代码，因为这样做的风险较高，容易导致整个项目崩溃

2. 功能分支

由于程序员不能直接在master分支上进行功能的开发，所以就有了功能分支的概念

功能分支指的是专门用来开发新功能的分支，它是临时从master主分支上分叉出去的，当新功能开发且测试完毕后，最终合并到master主分支上

3. 查看分支列表

使用如下的命令，可以查看当前Git仓库中所有的分支列表：

```
git branch
```

注意：分支名字前面的*号表示当前所处的分支

4. 创建新的分支

使用如下命令，可以基于当前分支，创建一个新的分支，此时，新分支中的代码和当前分支完全一样：

```
git branch 分支名称
```

注意：执行完创建分支命令之后，用户当前所处的分支还是master主分支

5. 切换分支

```
git checkout 分支名称
```

6. 分支的快速创建和切换

使用如下的命令，可以创建指定名称的新分支，并立即切换到新分支上：

```
git checkgout -b 分支名称
```

7. 分支合并

功能分支的代码开发测试完成之后，可以使用如下命令，将完成后的代码合并到master主分支上

(1)切换到master分支

```
git checkout master
```

(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

```
git merge login
```

8. 删除分支

当把功能分支的代码合并到master主分支上以后，就可以使用如下的命令，删除对应的功能分支：

```
git branch -d 分支名称
```

注意：不能删除当前所在的分支，必须在其他分支上时才能删除当前分支

9. 遇到冲突时的分支合并

如果唉两个不同的分支中，对同一个文件进行了不同的修改，Git就没法干净的合并他们。此时，我们需要打开这些包含冲突的文件然后**手动解决冲突**

假设：在把reg分支合并到master分支期间，代码发生了冲突

```
git checkout master
git merge reg
```

打开包含冲突的文件，手动解决冲突后，再执行如下的命令：

```
git add .
git commit -m "解决了分支合并冲突的问题"
```

Git分支-远程分支操作

1. 将本地分支推送到远程仓库

如果是第一次将本地分支推送到远程仓库，只在第一次推送的时候需要带 -u 参数

```
git push -u 远程仓库的别名 本地分支名称：远程分支名称
```

实际案例：

```
git push -u origin payment:pay
```

如果希望远程分支的名称和本地分支的名称保持一致，可以对命令进行简化：

```
git push -u origin payment
```

注意：第一次推送分支需要带`-u`参数，此后可以直接使用`git push`推送代码到远程分支

2. 查看远程仓库中所有的分支列表

通过如下命令，可以查看远程仓库中，所有的分支列表的信息：

```
git remote show 远程仓库名称
```

远程仓库名称默认都是`origin`

3. 跟踪分支

跟踪分支指的是：从远程仓库中，把远程分支下载到本地仓库中，需要运行的命令如下：

- 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

```
git checkout 远程分支的名称
```

- 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

```
git checkout -b 本地分支名称 远程仓库名称/远程分支名称
```

例如：

```
git checkout -b payment origin/pay
```

4. 拉取远程分支的最新代码

可以使用如下的命令，把远程分支最新的代码下载到本地对应的分支中：

```
git pull
```

5. 删除远程分支

可以使用如下的命令，删除远程仓库中指定的分支：

```
git push 远程仓库名称 --delete 远程分支名称
```

例如：

```
git push origin --delete pay
```

应该与删除本地分支有所区别

```
git branch -d 本地分支名字
```