学习Git

命令中使用[]括起来的都是需要根据自己的实际情况更改的。

安装git

可以试一下非安装版

创建本地用户

在本地创建一个用户，如果名字中间有空格，需要使用英文双引号。

```
git config --global user.name UserName

git config --global user.email UserName@gmail.com
```

提高命令输出的可读性

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

初始化

将一个普通文件夹交给git管理，会生成一个.git文件夹

先cd到相应的目录

```
git init
```

连接远程仓库

刚刚init了一个仓库，但是没有与任何远程仓库建立连接，不能push到远程，所以需要使用remote add origin [远程仓库link]

```
git remote add origin [link]
```

clone

克隆一个项目

```
git clone https://gitee.com/wang_tao_zhi/test.git
```

```
git clone git@gitee.com:wang_tao_zhi/test.git
```

status

查看当前工作状态

```
git status
```

add

git不知道需要管理哪些文件，提交到暂存区的文件才是git真正要管理的文件。

将指定文件加入到暂存区

```
git add [filename]
```

将所有文件加入到暂存区

```
git add .
```

删除暂存区中的文件

当需要删除暂存区或分支上的文件，**同时工作区需要这个文件，但是不需要被版本控制**

先使用下面命令删除暂存区文件，后面添加进.gitignore文件中的文件可以使用这条命令解除版本控制的追踪，然后在commit忽略这个文件。

```
git rm --cache [fileName]

git rm -r --cache [文件夹]
```

commit

将暂存区中的文件提交到本地git中

```
git commit -m "备注"
```

使用参数-a可以省略add，-a -m=-am

```
git commit -a -m "备注"
```

push

将本地git中的内容push到github中——远程仓库。

push到与所在分支同名的远程分支

```
git push
```

push到指定名称的远程分支

```
git push origin [remote_branch_name]
```

将本地某分支push到远程某分支

```
git push origin [local_branch_name]:[remote_branch_name]
```

如果远程分支不存在则根据提示

```
 git push --set-upstream origin [new_remote_branch_name]
```

stash

如果手头上的工作还没完成，现在有一个紧急任务需要切换分支处理，但是又不想提交一个脏提交，所以可以使用stash命令。

> 在Git中，`stash` 命令可以将当前工作目录中的修改保存到一个临时区域，以便稍后恢复。这对于需要切换分支或暂时处理其他任务的开发人员非常有用。

将当前工作目录中的未提交的更改保存到一个新的堆栈项中，并清空工作目录：

```
git stash save "备注"
```

列出保存在堆栈中的所有项：》最后一个stash是stash@{0}

```
git stash list
```

查看stash的摘要

```
git stash show -p
```

恢复最近的堆栈项并将其应用于工作目录，但不会从堆栈中删除该项：

```
git stash apply
```

恢复最近的堆栈项并将其应用于工作目录，然后从堆栈中删除该项：

```
git stash pop
```

从堆栈中删除指定的堆栈项：

```
git stash drop
```

清除整个 stash：

```
git stash clear
```

fetch

远程分支同步到本地

当 git fetch 命令从服务器上抓取本地没有的数据时，它并不会修改工作目录中的内容。它只会获取数据然后让你自己合并。

```
git fetch origin [branch_name]
```

手动合并

```
git merge FETCH_HEAD
```

pull

由于 git pull 的魔法经常令人困惑所以通常单独显式地使用 fetch 与 merge 命令会更好一些。

git pull = git fetch + git merge

将与本地当前分支同名的远程分支 拉取到 本地当前分支上

```
git pull
```

将远程指定分支 拉取到 本地当前分支上

```
git pull origin [remote_branch_name]
```

将远程指定分支 拉取到 本地指定的分支上

```
git pull origin [remote_branch_name]:[local_branch_name]
```

分支

创建分支

```
git branch [new_branch_name]
```

切换分支

```
git checkout [branch_name]
```

切换到远程分支

```
git checkout origin/[branch_name]
```

创建并切换分支

```
git checkout -b [new_branch_name]
```

分支重命名

本地分支重命名(还没有推送到远程)

```
git branch -m [oldName] [newName]
```

远程分支重命名 (已经推送远程-假设本地分支和远程对应分支名称相同)

其实就是先把本地分支名称改了，然后删掉远程分支，然后把重命名后的分支再push到远程仓库里，最后将本地分支与远程分支关联

1. 重命名远程分支对应的本地分支

```
git branch -m oldName [newName]
```

2. 删除远程分支

```
git push --delete origin [oldName]
```

3. 上传新命名的本地分支（会自动在远程仓库创建一个同名分支）

```
git push origin [newName]
```

4. 把修改后的本地分支与远程分支关联

```
git branch --set-upstream-to origin/[newName]

git branch --set-upstream-to=origin/[remote_branch] [newName]
```

删除本地分支

```
git branch -d [branch_Name]
# 强制删除 -D
git branch -D [branch_Name]
```

删除远程分支

```
git push origin --delete [branch_name]
```

删除远程跟踪分支

```
git branch -dr origin/[branch]
```

意思就是将当前分支与远程分支解除关系，如果再次git push当前分支由于找不到跟踪的远程分支会提示git push --set-upstream origin [branch_name] 意思就是将当前分支与远程的某个分支建立跟踪关系，有三种方法可以push。

```
git push --set-upstream origin [branch_name]
```

或者（这种方法下次git push的时候还是会提醒未与远程分支建立跟踪关系）

```
git push origin [local_branch_name]:[remote_branch_name]
```

或者（这种方法下次git push的时候还是会提醒未与远程分支建立跟踪关系）

```
git push origin [remote_branch_name]
```

合并分支

将指定分支合并到当前分支

```
git merge [branch_name]
```

如果两个分支有冲突——两个分支都对同一文件进行了修改，分支名称变为(branch_name|MERGING)，git不知道你需要哪个版本，所以这时候就需要手动去打开冲突文件分析需要保留哪个版本，把不需要的内容删掉后再commit。

仓库重命名与移除

> 是的，Git可以修改远程仓库的名称。您可以使用 `git remote rename` 命令来更改现有远程仓库的名称。例如，如果要将名为 `old-origin` 的远程仓库重命名为 `new-origin`，可以使用以下命令：

```
git remote rename [old-origin] [new-origin]
```

> 这将在本地Git仓库中将现有的 `old-origin` 远程仓库重命名为 `new-origin`。请注意，这不会影响远程仓库本身，只是将其在本地Git仓库中的名称更改为新名称。

例如将 pb 重命名为 paul

```
git remote rename [pb] [paul]
```

删库跑路

> 是的，Git可以删除远程仓库。您可以使用 `git remote rm` 命令来删除已经添加的远程仓库。例如，如果要删除名为 `origin` 的远程仓库，可以使用以下命令：

```
git remote rm [origin]
```

> 请注意，这将从本地Git仓库中删除指定的远程仓库，并且不会影响远程仓库本身。如果您想要完全删除远程仓库，请联系相应的Git托管服务（如GitHub、GitLab等）以获取更多帮助。

```
git remote remove [paul]
```

删除本地仓库

```
# 先删除.git文件夹
rm -rf .git

cd ..
rm -rf [rep_name]
```

遇到的一些问题

pull 分支报错

```
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false  # merge
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
```

分析：这是由于你拉取pull分支前，进行过merge合并更新分支操作，而其他人在你之前已经push过一个版本，导致版本不一致

第一种解决方法：比较简单

- 执行`git config pull.rebase false`
- 默认将pull下来的代码与现有改动的代码进行合并
- 但是可能会造成代码冲突，需要处理下这个问题，代码冲突如果2个人都改了同一个文件，需要联系之前push的同学，看看这块代码怎么保存

fatal: refusing to merge unrelated histories

这里的问题的关键在于：`fatal: refusing to merge unrelated histories`
你可能会在`git pull`或者`git push`中都有可能会遇到

在你操作命令后面加`--allow-unrelated-histories`

```
git pull --allow-unrelated-histories
```