一、Git基本管理

1.1 Git的三大区域

- 工作区(正在操作的那个文件夹)
  - 文件透明:表明文件已经被git妥善管理(已生成版本)
  - 文件红色:表明是新增文件/修改过的文件
  - 使用git　add 命令使文件进入暂存区
- 暂存区
  - 文件绿色
  - 是一种缓冲
  - 使用git commit 命令可提交文件形成新的版本
- 版本库
  - 存放有提交过的历史版本

1.2 Git基本操作

```
git init                        初始化　git帮助我们管理当前文件夹
git status                      检测当前目录下文件夹的状态
git add                         文件名或者git add . 让git管理起来修改过的文件
git commit -m '描述信息'        提交一个新的版本
```

1.3 回滚

- 回滚至之后的版本

```
git log
git reset --hard 版本号
```

- 回滚至之前的版本

```
get reflog
git reset --hard 版本号
```

1.4 小结

```
git init
git add
git commit
git log
git reflog
 --
git reset --soft 版本号          将该版本从版本库转移到暂存区
git reset --HEAD .或文件名       将暂存区的文件回退为红色文件
git checkout -- 文件名           将已修改的红色文件回退为没修改前的模样
git reset --mix 版本号           将该版本从版本库直接移动到红色文件
```

二、修复Bug的思路

2.1 利用分支的思想

- 主干线叫master,其他分支可以自定义名字
    - 创建一个新的分支去修复bug,修复完成后将该分支合并到原分支master
    - 在有bug的那个版本的基础上开发的新功能的分支（如叫dev分支）合并到主分支上时,可能会出现冲突,此时应该手动修复代码

2.2 有关分支branch的命令

```
git branch           查询所有的分支
git branch name      创建一个名为name的分支
git checkout name    切换到name分支
git merge name       合并name分支到当前分支
git branch -d name   删除name分支
```

2.3 工作流

- 默认会有master主分支
- 应该至少再整出来一个dev(开发)分支
    - 例如在master主干上的v2.0版本时创建一个dev分支，那么dev分支是基于v2.0版本的；
    - 待到dev分支开发的新功能已经成熟稳定，再切换到master分支，将dev分支的新功能merge到主分支。
    - 一般是的顺序都是先进行commit形成版本之后才有可能进行版本合并。
    - 功能完善的dev分支合并到master后，可以选择删除该dev分支，然后再在master上开一个dev分支
    - 也可以不删除原先的dev分支，作为一个记录保留在仓库中；新建一个分支继续开发新功能。
- master用于记录稳定版本，dev分支用于记录测试版

三、Github

3.1 三个步骤

1. 注册账号
2. 创建仓库
3. 本地代码推送

3.2 命令

- 最初的上传代码

```
给远程仓库起别名
    git remote add origin https://github.com/DongDong-geeeeek/Bgics.git
向远程仓库推送代码(将本地分支推送到远程的指定分支)
    git push -u origin 分支名称
```

- 在另一个地方(比如公司)第一次获取线上的代码

```
克隆远程仓库代码：
clone用于本地没有任何线上仓库中的代码，运行命令会将线上仓库的代码全部拷贝到本地
    git clone 远程仓库地址
切换分支
    git checkout 分支名
```

- 在公司进行开发

```
切换到dev分支进行开发
    git checkout dev
把master分支合并到dev[仅进行一次即可]
    git merge master
修改代码
提交代码
    git add .
    git commit -m 'xx'
    git push origin dev
```

- 回到家中继续开发

```
切换到dev分支进行开发
    git checkout dev
拉取线上代码(将远程仓库的新的代码添加到本地)
    git pull origin dev
    git pull github 地址 分支名
继续开发

提交代码
    git add .
    git commit -m 'xx'  先本机的git提交新的代码版本
    git push origin dev 上传至github
```

- 在公司继续开发

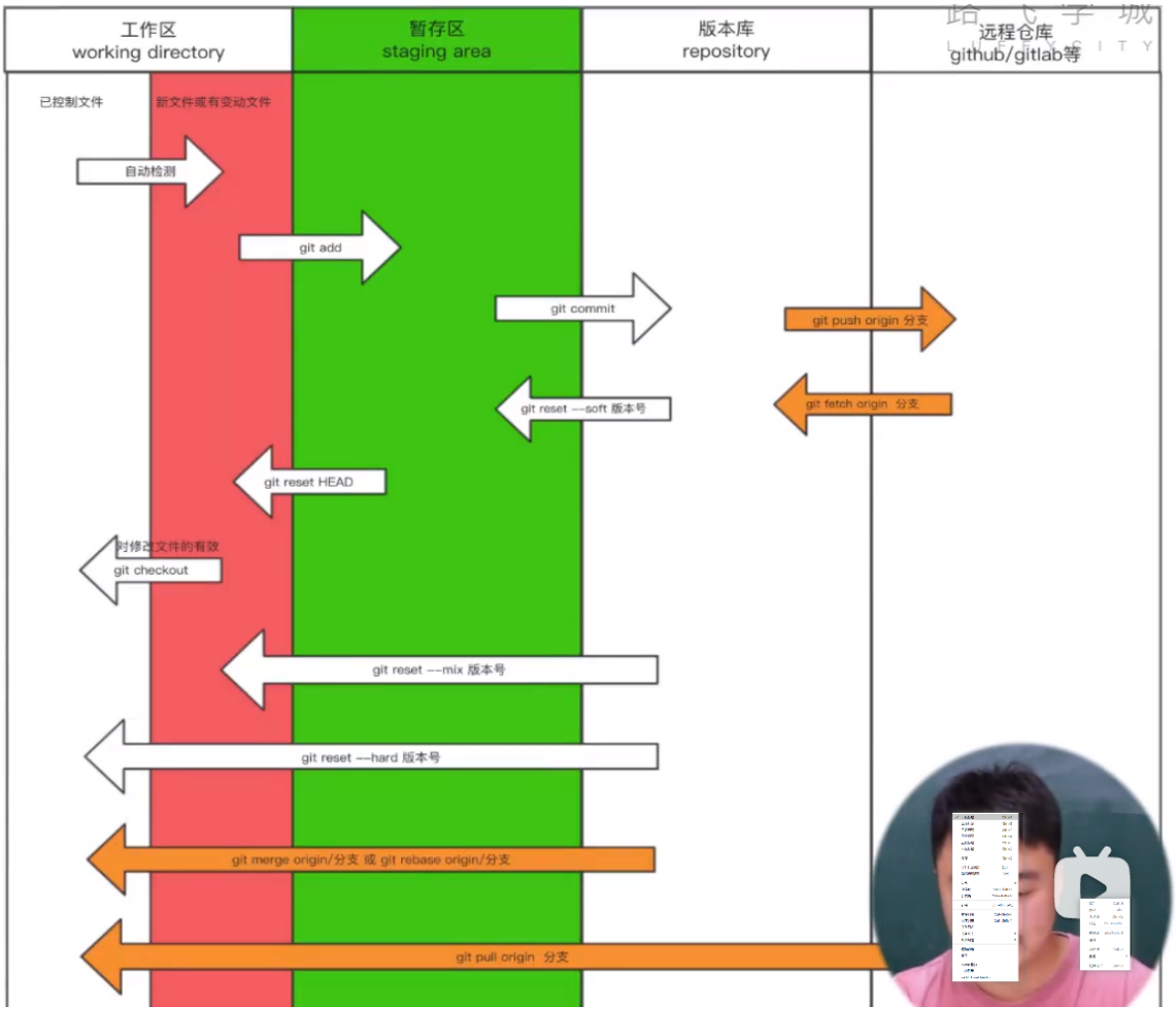
```
切换到dev分支进行开发
    git checkout dev
把master分支合并到dev[仅进行一次即可]
    git merge master
修改代码
提交代码
    git add .
    git commit -m 'xx'
    git push origin dev
```

- 若dev 开 发完毕，要上线（大概的流程）

```
切换到master分支
    git checkout master
合并dev到master
    git merge dev
上线到github仓库
    git push origin master
```

- 注意

```
注意:
git pull origin dev
等价于下面的两条命令
git fetch origin dev     将远程仓库的版本拿到本地版本库
git merge origin/dev     将已拿到本地版本库的版本合并到工作区
```



四、变基

- 作用：使git记录变得简洁

4.1 将多个log简化为一个log

- 注意：合并记录时不要合并那些已经push到Github的版本记录

```
git rebase -i 版本号          表示将当前所在版本依次到版本号指定的版本合并成一个log记录
git rebase -i HEAD~n          表示从当前所在版本依次合并n个版本的log记录
```

- 进入vi的编辑模式修改后怎么保存退出?

```
按Esc--->光标跳转到最后
输入:wq--->保存退出
```

- 此时应该编辑合并信息

```
在insert模式下可以编辑
修改添加想要的合并信息
保存退出即可
```

4.2 将分支和master的log整理成清晰的commit线

- 获取log的图形表示

```
git log --graph
git log --graph --pretty=format:"%h %s"
```

- 将dev上所有的commit，重新在新的master的HEAD上commit一遍，有冲突要手动解决。

```
git checkout dev     切换到dev分支
git rebase master
git checkout master
git merge dev
git log --graph      获得清晰的commit线性流程
```

4.3 若本地有新代码(未上传Github),线上也有新代码(本地没有)

- 方法1：会产生分叉

```
git checkout dev
git pull origin dev
    等价于下面的两条命令
    git fetch origin dev     将远程仓库的版本拿到本地版本库
    git merge origin/dev     将已拿到本地版本库的版本合并到工作区
```

- 方法2：不会产生分叉

```
git checkout dev
git fetch origin dev     将远程仓库的版本拿到本地版本库
git rebase origin/dev
```

4.4 Beyond Compare

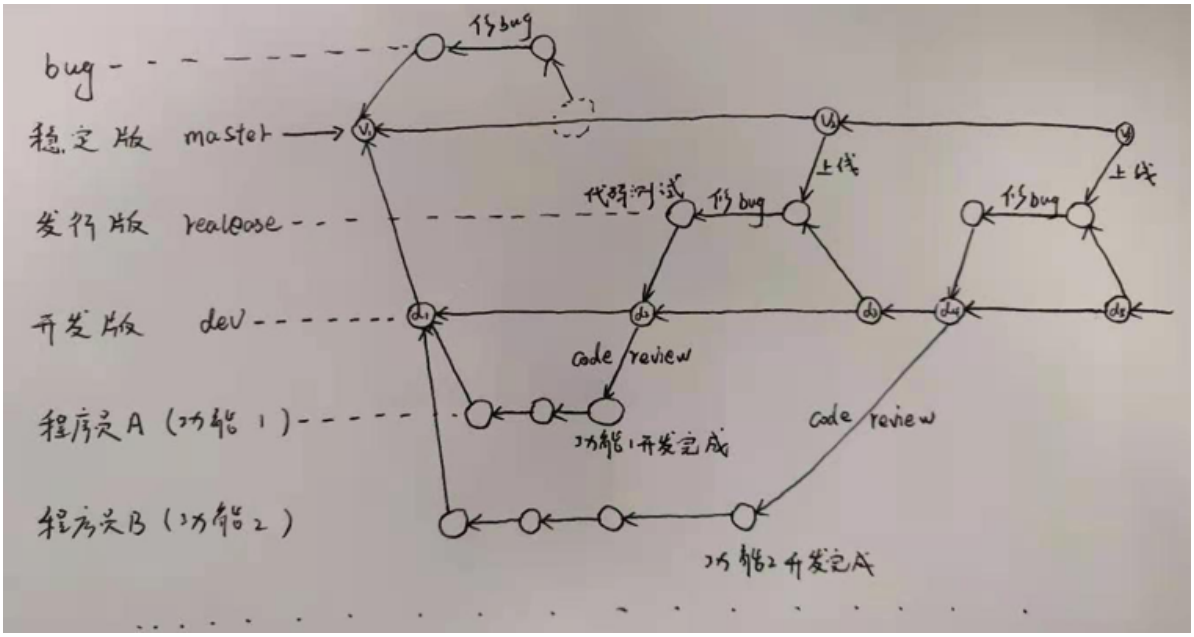
1. 安装 Beyond Compare
2. 在git中配置

```
git config --local merge.tool bc3
git config --local mergetool.path 'D:\BeyondCompare\install_Path\Beyond
Compare 4'
git config --local mergetool.keepBackup false
```

3.引用beyond compare 解决冲突

```
git mergetool
```

五、多人协同工作流



5.1 利用Github多人协作

1. 创建一个普通项目然后邀请其他人参与进来
2. 创建一个组织，在该组织中创建项目，邀请其他人进入项目，以后该组织中的所有项目，组织中的成员皆可参与

- 为当前HEAD指向的版本创建标签，并附加“描述信息”

```
git tag -a 标签 -m "描述信息"
```

- 将标签更新到线上

```
git push origin --tags
```

- 创建新的分支并直接切换到该分支

```
git checkout -b 新标签名
```

5.2 给开源项目贡献代码

1. fork源代码
    - 将别人源代码拷贝到我自己的远程仓库
2. 从远程仓库下载到本地仓库
    - 进行修改代码
    - 然后再上传到我的远程仓库
3. 给源代码作者提交我修改的代码(pull request)

六、补充知识

6.1 配置

- 当前项目配置文件：项目 \ .git \ config

```
git config --local user.name 'dzc'
git config --local user.email 'dzc@163.com'
```

- 全局配置文件：C:\Users\18221\ .gitconfig
    - 所有的项目都可使用

```
git config --global user.name 'dzc'
git config --global user.email 'dzc@163.com'
```

- 系统配置文件：\etc \ .gitconfig (在我的电脑上没有找到)

```
git config --system user.name 'dzc'
git config --system user.email 'dzc@163.com'

注意:需要有root权限
```

应用场景:

```
git config --global user.name 'dzc'
git config --global user.email 'dzc@163.com'

git config --local merge.tool bc3
git config --local mergetool.path 'D:\BeyondCompare\install_Path\Beyond Compare 4'
git config --local mergetool.keepBackup false

git remote add origin URL  默认添加地址到本地配置文件中,也即仅仅本项目可以用origin代表该URL
```

6.2 免密登录

- URL实现

```
原来的地址: https://github.com/DongDong-geeeeek/FiraCode.git
修改的地址: https://用户名:密码@github.com/DongDong-geeeeek/FiraCode.git

git remote add origin https://用户名:密码@github.com/DongDong-geeeeek/FiraCode.git
git push origin master
```

- SSH实现

```
1. 生成公钥和私钥(默认放在 ~\.ssh目录下,id_rsa.pub公钥、id_rsa私钥)
    ssh-keygen
2. 拷贝公钥的内容,并设置到github中
    vim id_rsa.pub  使用vim编辑器打开公钥
3. 在git本地中设置ssh地址
    git remote add origin git@github.com:DongDong-geeeeek/FiraCode.git

4. 以后使用就可以不用密码
    git push origin master
```

- git自动管理凭证(现在个人用户用的最多的)

6.3 .gitignore文件

- 可以让git忽略到某些文件

```
vim .gitignore  编辑.gitignore文件,写入的文件类型将被git忽略
```

- .gitignore

```
a.h          忽略a.h文件
*.h          忽略所有.h后缀的文件
!a.h         特别的,不忽略a.h
.gitignore   忽略.gitignore文件
files/       忽略文件夹files
```

- Github上有专门的某种编程语言的.gitignore推荐文档
- 更多参考：https://github.com/DongDong-geeeeek/gitignore

6.4 任务管理相关

- issues
  - 相当于项目的讨论区，可以提问
- wiki
  - 对项目的描述
  - 一些小百科

附录：常用的Linux命令

```
pwd          查看当前文件夹路径
F:           进入F盘
D:           直接从当前盘进入F盘
cd 名称       打开文件夹\文件\软件
cd..         返回上一级目录
dir          查看当前文件夹中的所有文件
color 数字    改变字体颜色
touch 文件名  创建一个xx文件
ls           列出当前文件夹的所有文件
```