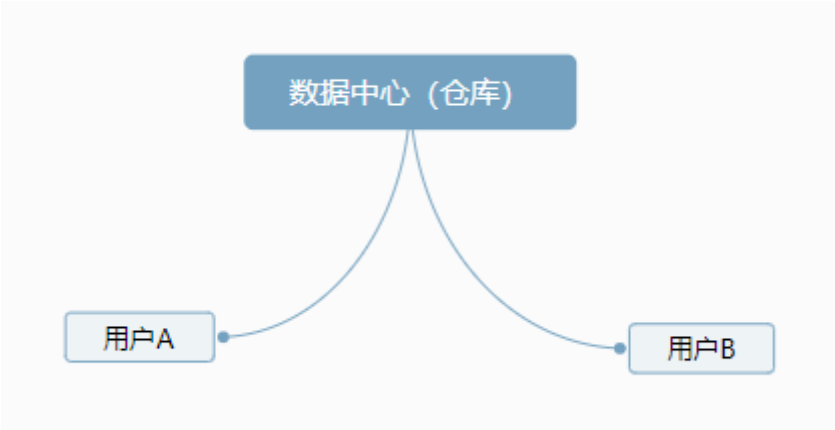
git学习总结

名称：分布式版本控制软件

历史进程：①版本复制 ②本地版本 ③集中版本 ④分布式版本



安装：

linux: sudo apt-get install git

windows: git --version ——>版本

第一章 基本命令

第一阶段：一人（自己写代码，版本控制）

版本控制——>git管理文件夹

①进入要管理的文件夹

②初始化（提名）

③管理文件与文件夹

④生成版本

git init——>进入仓库（初始化）

git status——>检测当前文件状态

三种状态：

①红色：新增文件或修改文件，git add 红色文件变成绿色

②绿色：git已经管理起来 git commit -m "描述信息"

③白色：生成版本

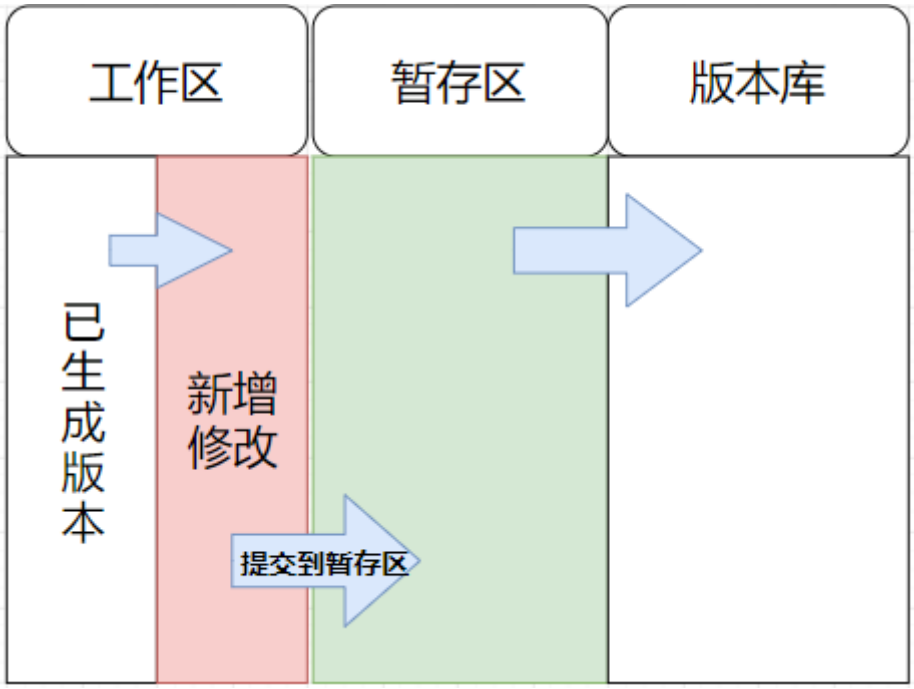
git add 文件名、文件夹——>管理文件

git add . ——>管理所有文件

个人信息配置：用户名、邮箱

git commit -m "" ——>生成版本

git log ——>查看版本记录



第二章 深入了解

2.1回滚：

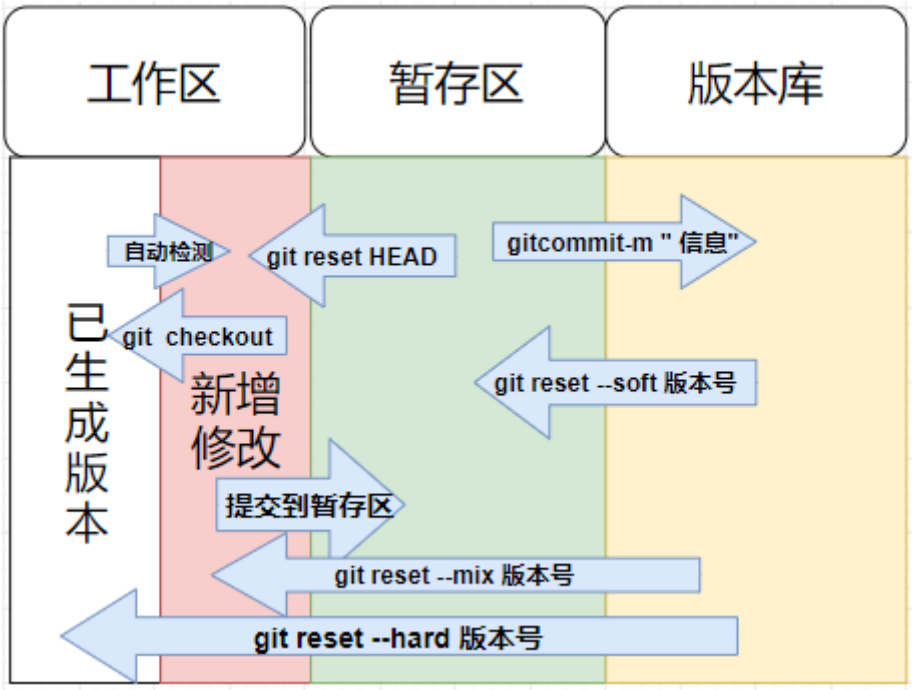
①git log ——>查看版本号

②git reset --hard 版本号

2.2反回滚：

①git reflog ——>查看版本号

②git reset --hard 版本号



2.3分支：线上bug修复，利用分支，版本合并，环境隔离

git branch

git branch dev ——>创建dev分支

git checkout dev ——>转跳到dev分支

git branch bug ——>创建bug分支

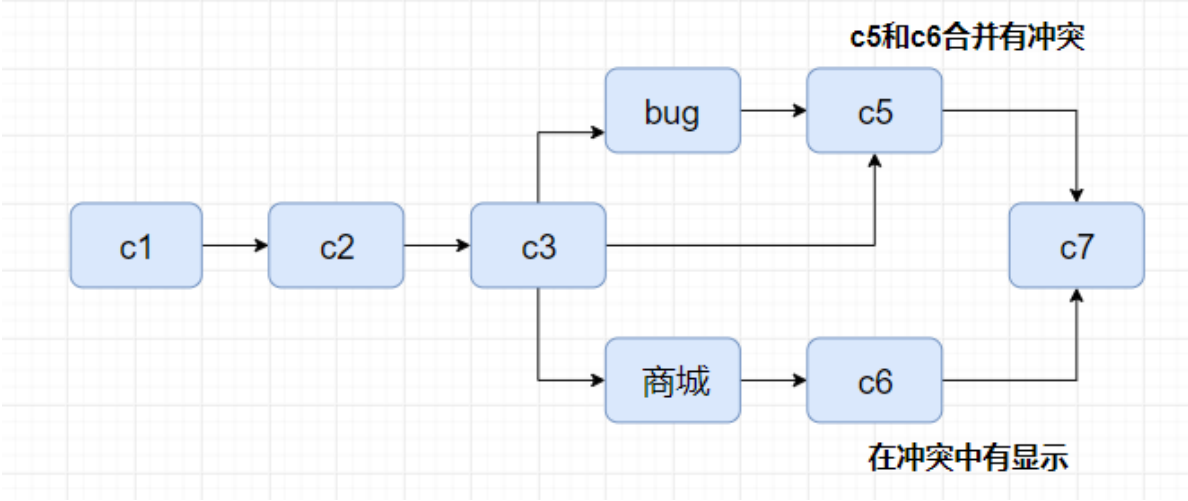
git checkout master——>转跳bug分支

git merge bug ——>合并分支（合并bug分支） 在主分支上合并分支

git branch -d bug 删除分支

git checkout master ——>转跳分支

git merge dev ——>报出冲突，修复冲突，原因是此时的master 和原来在商城（dev）的master不相同



2.4命令总结：

查看分支

```
git branch
```

创建分支

```
git branch 分支名称
```

切换分支

```
git checkout 分支名称
```

分支合并（可能产生冲突）

```
git merge 要合并的分支
注意：切换分支再合并
```

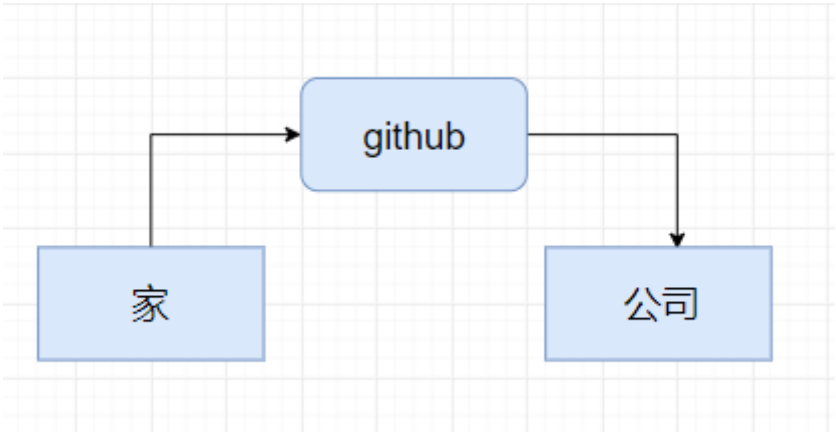
删除分支

```
git branch -d 分支名称
```

2.5工作流

正式版在master上，开发版在dev分支上进行

第三章 github



①注册账号

②创建仓库

③本地创建代码

“家”操作连接：

```
git remote add origin xxx(仓库网址)
```

```
git push origin master(dev) //上传分支
```

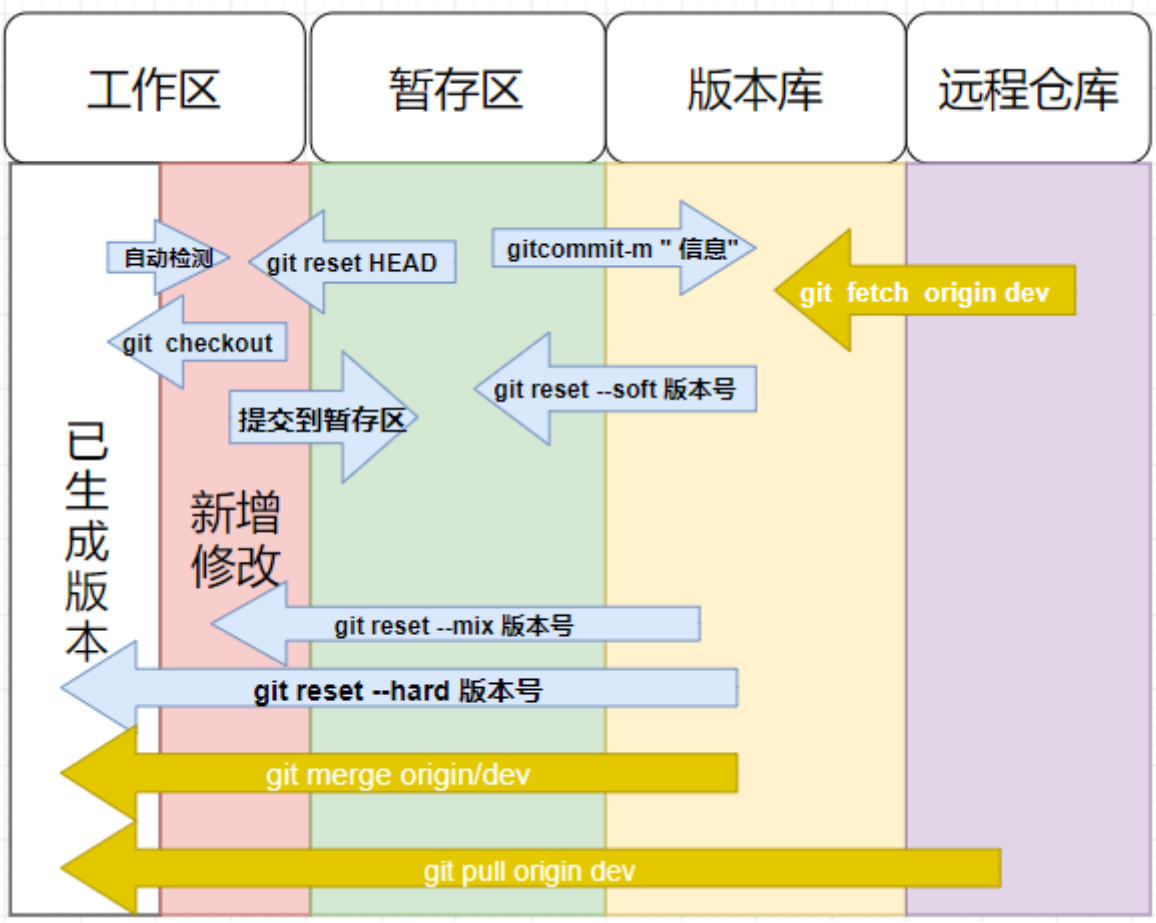
```
vim .git/config //修改配置资料
```

第一次“公司”下载：

```
git clone XXX(仓库网址)
```

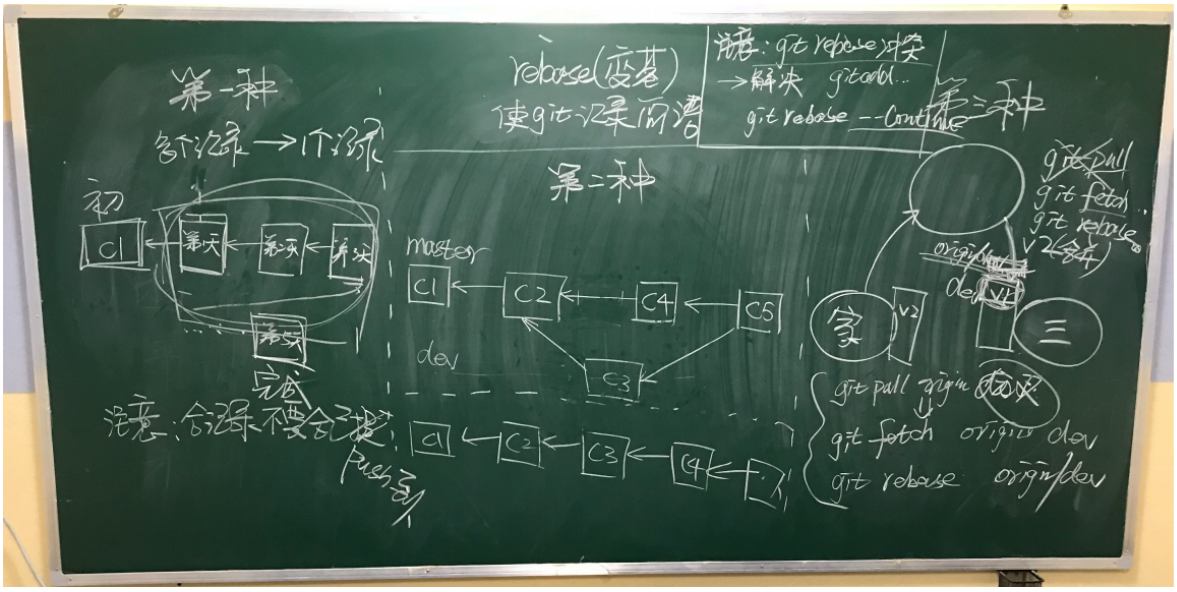
“在家”下拉

```
git pull origin 分支名称
等价于下面两个命令
git fetch origin dev
git merge origin/dev
```



第四章 rebase(变基)

功能：使git提交记录变得简洁



第一种情况：多个记录整合成一个记录

```
git rebase -i HEAD~次数
```

注意：合并的次数之前不要上传到仓库

第二种情况：合并分叉记录

```
git log --graph //图像化显示
```

第一步：切换到合并分支

```
git checkout 合并分支
```

第二步：合并主分支

```
git checkout master
git rebase master
```

第三步：最后合并分支

```
git merge 合并分支
```

第三种情况：公司中忘记上传代码，在家上传代码后，出现冲突

```
git fetch origin dev
```

```
git rebase origin/bev //可能产生冲突
```

第五章 快速解决冲突

1.安装beyond compare

2.在git中配置

```
git config --local merge.tool bc3 //软件
git config --local mergetool.patt '/user/local/bin/bcomp' //安装地址
git config --local mergetool.keepBackup false //备份
```

3.应用beyond compare 解决冲突

```
git mergetool
```

前五章总结

1.添加远程连接（别名）

```
git remote add origin 地址
```

2.推送代码

```
git push origin dev
```

3.下载代码

```
git clone 地址
```

4.拉取代码

```
git pull origin dev(分支)
等价于
git fetch origin dev
git merge origin/dev
```

5.保持代码提交整洁（变基）

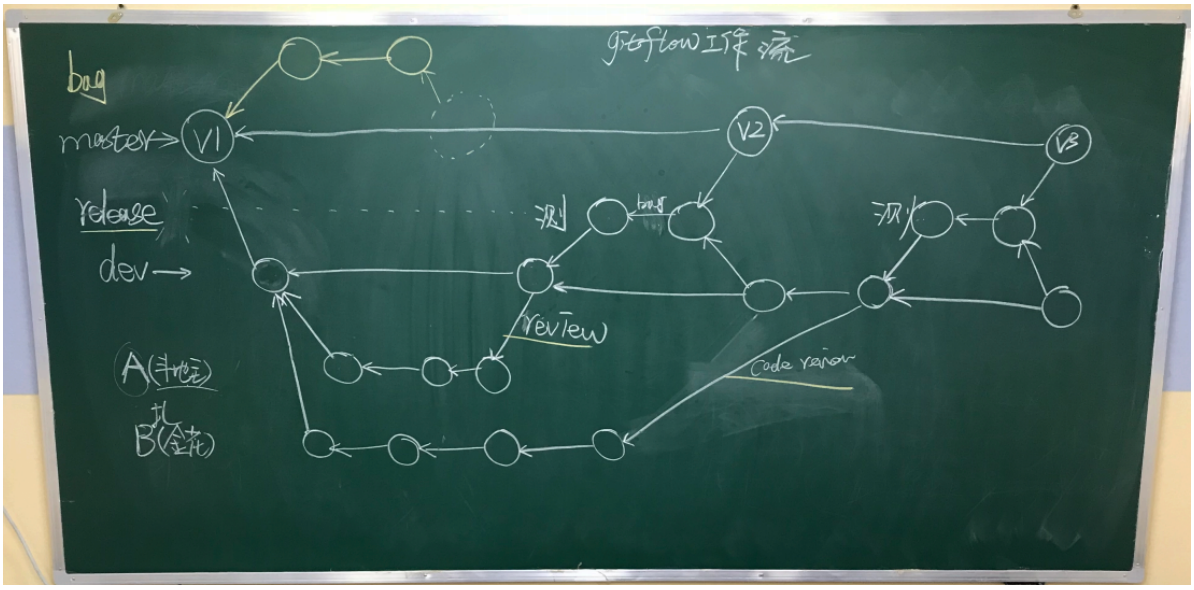
```
git rebase 分支
```

记录图形展示

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

第六章 多人协同开发

协同开发gitflow工作流



一、创建第一版并打上标签：

首先在github上新建一个组织的仓库，并且新建仓库——>公司企业级

建立新的项目与github进行连接

```
git init
vim 123.py
git add .
git commit -m "基本功能"
git remote add origin 网址
git push -u origin master
git log
git tag -a v1(第一版) -m "第一版"
```

此时在github的仓库也能看见标签

二、创建分支

```
git checkout -b dev  //创建dev分支，并且切换到dev分支
```

成员在dev分支上进行创建新的分支，并且进行新的开发。

三、进行代码review

github上进行设置rule,然后请求账号进行代码review，设置pull request

四、进行release

小规模代码修改

五、合并master分支

进行上线合并，gitflow工作流程

第七章 给开源软件贡献代码

1、kork源代码，将别人的源代码拷贝到我自己的远程仓库

2、本地连接远程，进行代码clone

3、进行添加修改代码，进行push代码

4、给原代码作者进行提交修复bug的申请（pull request)

5、点击确定creat pull request

第八章 其他

1. 配置文件: 三种方式

```
git config --local user.name '李阳齐'
git config --local user.email 'liyangqi@xx.com'

git config --global user.name '李阳齐'
git config --global user.email 'liyangqi@xx.com'

git config --system user.name '李阳齐'
git config --system user.email 'liyangqi@xx.com'
```

三个配置文件：

①当前项目、本地配置文件：.git/config

```
git config --local user.name '李阳齐'
git config --local user.email 'liyangqi@xx.com'
```

②全局配置文件：当前用户 ~/.git/config

```
git config --global user.name '李阳齐'
git config --global user.email 'liyangqi@xx.com'
```

③系统配置文件：/etc/.git/config

```
git config --system user.name '李阳齐'
git config --system user.email 'liyangqi@xx.com'
注意：需要有root权限
```

应用场景：

```
git config --local user.name '李阳齐'
git config --local user.email 'liyangqi@xx.com'

git config --local merge.tool bc3 //软件
git config --local mergetool.patt '/user/local/bin/bcomp' //安装地址
git config --local mergetool.keepBackup false //备份

git remote add origin 地址 ， 默认添加在本地配置文件中。
```

2. 免密登入：

1.URL 中体现

```
原来的地址：https://github.com/4794-sourse/tornado.git
修改的地址：https://用户名：密码@github.com/4794-sourse/tornado.git

git remote add origin https://用户名：密码@github.com/4794-sourse/tornado.git
git push origin master
```

2 .ssh实现

```
1.生成公钥和私钥（默认放在~/.ssh/  id_rsa私钥、id_rsa.pub公钥）
    ssh-keygen //回车
2.拷贝公钥内容，并设置在github

3.在git本地配置项目ssh地址
    git remote add origin git@github.com:4794-sourse/tornado.git
4.以后使用
    git push origin master
```

3.git自动管理凭证

3.git忽略文件：

```
vim .gitignore
将文件名添加在.gitignore中

*.h
!a.h
files/
*.py[c|a|d]
```

参考地址：https://github.com/github/gitignore

4.任务管理相关：

① issues:文档以及任务管理

② wiki：项目相关资料