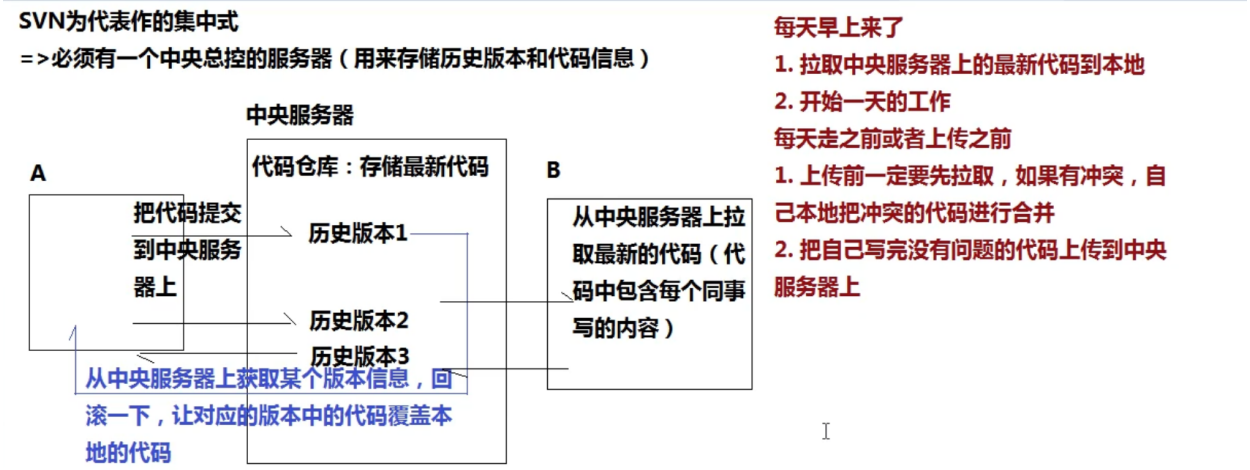
我的git

- 用户名 dayu-jiayou
- 密码 qq123214@

GIT版本控制系统

> 版本控制系统

svn



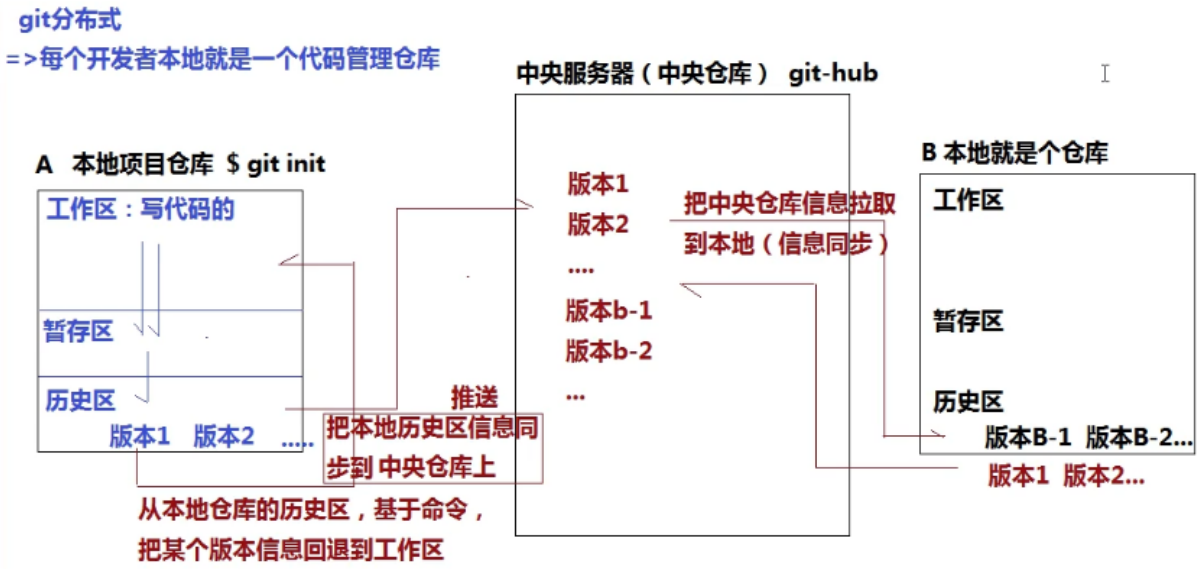
每天一到先干什么

1. 拉取中央服务器上的最新代码到本地
2. 开始一天的工作 每天走之前或者上传之前
3. 上传一定要先拉取，如果有冲突，自己本地把冲突的文件的代码进行合并
4. 把自己写完没有问题的代码上传到中央服务器上

svn弊端：

1. 需要联网才能回退或查看历史版本信息
2. 中央服务器毁坏了，一切over
3. 所有的上传和下载都是基于文件的传输方式完成的，速度会慢。

git分布式



git好处：

1. 无需联网也能记录和查看历史版本信息
2. 无需过多依赖中央仓库，每个人本地也有全部的信息
3. 向中央仓库传输内容依托的是文件流传输，速度比SVN快N倍

1.GIT] C作原理

- 工作区:我们能看到的,并且用来写代码的区域
- 暂存区:临时存储用的
- 历史区:生成历史版本

第一次全局配置

第一次安装完成git后 ,我们在全局环境下配置基本信息:我是谁?

```
$ git config -l
查看配置信息
$ git config --g1oba1 -l 查看全局配置信息

配置全局信息:用户名和邮箱
$ git config --global user.name 'dayu'
$ git config --global user.email ' xxx@xx. xx'
```

```
$ git init
//=>会生成一个隐藏文件夹 ".git" (这个文件夹千万不要删，因为暂存区和
历史区还有一些其它的信息都在这里， 删了就不是一个完整的git仓库)
在本地编写完成代码后(在工作区) ,把- -些文件提交到暂存区
$ git add xXX把某一个文件或者文件夹提交到暂存区
$ git add . 把当前仓库中所有最新修改的文件都提交到暂存区
$ git add -u 包含修改和删除的，但是不包含新增的
$ git add -A 同时具备.和-u 的特征
$ git status 查看当前文件的状态(红色代表在工作区，绿色代表在暂存
```

```
区，看不见东西证明所有修改的信息都已经提交到历史区）
```

把暂存区内容提交到历史区

```
$ git commit -m'描述信息:本次提交内容的一个描述'

查看历史版本信息(历史记录)
$ git log
$ git reflog 包含回滚的信息
```

1. Settings用户设置

- Profile 修改自己的基本信息
- Account 可以修改用户名
- Security 可以修改自己的密码
- Emails 邮箱(必须进行邮箱校验) .....

2. 创建仓库 new repository -> 填写信息-> Create repository

- public公共仓库作为开源的项目
- private 私有仓库作为内部团队协作管理的项目
- Settings ->删除仓库Delete this repository ->Collaborators设置协作开发的人员 Code可以查看历史版本信息和分支信息

3. 把本地仓库信息提交到远程仓库

```
//=>建立本地仓库和远程仓库的链接
查看本地仓库和哪些远程仓库保持链接
$ git remote -V
让本地仓库和远程仓库新建一个链接origin是随便起的一个链接名(可以改成自己想要的，只不过一般都用这个名字)
$ git remote add origin [GIT远程 仓库地址]
删除关联信息
$ git remote rm origin
提交之前最好先拉取
$ git pu11 origin master
把本地代码提交到远程仓库(需要输入github的用户名密码)
$ git push origin master
```

$ git clone [远程仓库git地址][别名：可以不设置，默认是仓库名]

真实的项目开发流程

1. 组长或者是负责人先创建中央仓库
2. 小组成员基于$ git clone 把远程仓库默认的内容克隆到本地一份（解决了三个事情：初始化了一个本地仓库‘git init’ ）和对应的远程仓库也保持了关联‘git remote add’把远程仓库默认内容拉取到本地‘git pull’
3. 每个组员写完自己的程序之后，基于‘git add/git commit’把自己修改的内容存放到历史区，然后通过‘git pull/git push’把本地信息和远程仓库信息保持同步即可(可能涉及及冲突的处理)