**今天内容： 两个软件的使用 - git 和 新浪云**

**需要网络**

**讲师的联系方式：**

**微信18800108022 小新老师**

**在ftp中 git\_sina 中下载今天内容的文档**

**如果是网页版本的ftp， 右键 -》另存为**

**先检查你电脑上是否有安装了git软件**

**检查方式：右键桌面 查看是否有 git gui here 栏目**

**如果没有安装：到官网下载**

<https://git-scm.com/downloads>

下载windows版本， 双击安装包 一直下一步即可安装完毕

网速慢， 也可以找项目经理要安装包， 42M

1. Git软件是什么

git是目前世界上最好的分布式版本管理系统，没有之一

2. 什么是分布式版本管理系统？

没有任何一门技术是无缘无故诞生的，都是在实际开发中

遇到了某些问题而诞生的解决方案。

实际开发中遇到的问题：

\* 老师写完的代码，同学照着写但是出现bug出不来。此时

就会拿着老师的代码 和 你自己的代码来对比不同的地方。

此时：能否有软件自动标注出不一样的地方？？

\* 大型项目书写时，往往新增的代码有可能导致项目出现bug

无法运行， 此时又找不到bug，我们希望能否回退到上一个

正确的时间节点。

\* 代码的保存：在单位工作时写的代码，回家以后也要，在家里

写完的代码第二天单位要用。此时如何移动代码，用U盘？

用网盘？ 无法记录修改的内容有哪些！ 记录每天修改了什么内容

\* 团队合作时：假设两个人合作同一个项目， A写一个页面，

B写另一个页面， 如何合并代码？ 需要一个软件能够自动合并

多个代码到一起。

git软件就是用来解决 代码的对比， 代码的存放， 代码的移动

代码的回退， 代码的修改记录， 代码的合并... 等等操作

版本控制软件 VCS version control system

有很多款：其中git就是最新的 2010年之后

git软件是一个专业的软件， 使用cmd命令行来进行使用。

今天要记住一些命令行

**打开命令行工具：win+R 搜索cmd**

**命令1：查看当前版本号**

**git version**

**目前最高是 2.23.0 只要是2.\* 以上版本即可**

**命令2：把一个项目采用git来进行管理，**

**随意找个位置创建一个文件夹， 名字叫 Demo**

**git软件就相当于一个物业公司，来管理某个文件夹下的所有内容**

**使用之前 必须先配置名字**

**git config --global user.name "你的名字"**

**git config --global user.email "你的邮箱"**

**例如：**

**git config --global user.name "xiaoxin"**

**git config --global user.email "736907613@qq.com"**

**查看用户名和邮箱地址：**

**$ git config user.name**

**$ git config user.email**

**shell 命令行工具有个理念：没有消息就是好消息，也就是说**

**只有出错的时候 才会给你必要的提示**

**查看是否配置成功**

**查看配置信息列表**

**git config -l**

**如果结果下方有： 按 下方向键 查看更多内容**

**结果是end时候， 代表到最后了 按Q 可以退出查看模式**

**下节课 10：00**

**多数错误是 不加空格导致的**

**把Demo文件夹变为git软件管理的文件夹**

**1. 把cmd命令行 在 Demo文件夹下打开**

**先到Demo文件夹下，在地址栏中输入cmd**

**回车即可打开命令行，路径就是demo文件夹**

**命令： git init**

**initialize缩写：init 初始化**

**在demo文件夹下会多出一个 .git 隐藏文件夹**

**首次修改：**

**在Demo文件夹下创建了 index.html**

**这属于新增文件的修改方式**

**立刻告诉git 此处有一个修改， 暂存起来**

**命令： git add 文件名**

**例如： git add index.html**

**查看是否成功暂存, status状态**

**$ git status**

**小技巧: 搜狗输入法 右键属性设置->中文时使用英文标点, 设置之后切换一次输入法,就可以在中文时使用英文标点了.**

**在git中, 凡是做了小修改, 并且没有bug**

**就可以立刻 输入**

**$ git add 文件名 来暂存修改**

**再使用**

**$ git status 查看状态**

**新的命令: 查看相对于之前暂存的文件 新增了哪些修改? different 不同**

**$ git diff index.html**

**再次新建多个文件**

**a.txt**

**b.txt**

**c.txt**

**d.txt**

**然后都暂存起来(git add xx), 然后查看暂存的状态(git status)**

**新的命令**

**$ git add .**

**.代表所有修改,都会被暂存**

**暂存: 暂时存储,临时性质的. 用于小修改**

**当暂存的修改达到很大的数量,或者当前代码已经完善到一定程度了, 可以保存起来形成一个版本.**

**新的命令:把所有暂存的修改保存起来,形成一个永久的记录**

**$ git commit -m "备注"**

**例如: commit**提交

**$ git commit -m "version 1.0"**

**用来看暂存的修改, 暂存提交以后就清空了**

**$ git status**

**看提交记录, 看版本记录**

**$ git log**

**练习:**

**新建 news.html, 换成 detail.html 再试一次**

**然后暂存 $ git add .**

**然后查看暂存的状态 $ git status**

**修改 news.html 中的内容**

**再暂存**

**再次修改 news.html中的内容**

**查看新修改的内容是什么: 看的是上一次暂存和当前改动的不同**

**$ git diff 文件名**

**再次暂存 -> 提交新版本 version 2.0 ->查看日志 $ git log**

**提交第五个版本:**

**新建 happy.txt**

**$ git add .**

**$ git commit -m "version 5.0"**

**$ git log**

**看简洁log的命令**

**$ git log --pretty=oneline**

**git log 记录中的HEAD-> 指向的是当前操作的版本**

**回退到上一个版本**

**$ git reset --hard "HEAD^"**

**^代表上一个版本, 上两个版本HEAD^^**

**上100个版本 HEAD~100**

**此时:要前进到5版本怎么办?**

**必须依赖于唯一标识**

**可以通过命令**

**$ git reflog 来查看所有版本的唯一标识**

**使用唯一标识来移动HEAD**

**$ git reset --hard 唯一标识**

**执行之后,查看happy.txt 是否出现了**

**再使用 git log 看HEAD的指向**

**----------------------------------**

**git也支持从暂存文件进行恢复**

**操作:**

**\* 在happy.txt 随便写点内容, 暂存起来**

**\* 然后在happy.txt中再新增一些内容**

**$ git checkout -- 文件名**

**作用:恢复指定文件到最近的暂存点**

**回顾:**

**git的基本操作:**

**先暂存->提交到版本**

**学习的命令有:**

**查看git版本号**

**$ git version**

**设置git用户的 姓名和邮箱**

**$ git config --global user.name "姓名"**

**$ git config --global user.email "邮箱"**

**查看配置信息列表**

**$ git config -l**

**把某个文件夹变为git管理的仓库, 在文件夹下会自动创建 .git 隐藏文件夹**

**$ git init**

**暂存修改**

**$ git add . 或 文件名**

**查看暂存的修改**

**$ git status**

**对比代码 和 暂存有什么不同**

**$ git diff 文件名**

**回退文件到暂存的状态**

**$ git checkout -- 文件名**

**版本的提交,必须双引号  
$ git commit -m "备注信息"**

**查看提交的版本记录,当前HEAD之前的版本**

**$ git log**

**$ git log --pretty=oneline**

**查看所有版本**

**$ git reflog**

**回退到上几个版本, HEAD^^ HEAD~100**

**$ git reset --hard "HEAD^"**

**前往到指定版本**

**$ git reset --hard 版本标识**

**大练习:**

**1. 再次设置你的用户名和邮箱, 稍作变动**

**2. 查看设置是否成功**

**3. 创建新的文件夹 Project01**

**4. 把 Project01 变为git管理的仓库**

**5. 在 Project01 中新增 a.txt 暂存+提交为第一个版本**

**6. 按照 第5步, 分别新增 b.txt c.txt d.txt e.txt 并提交对应的4个版本**

**7. 修改a.txt 中的内容, 暂存修改. 再次修改a.txt内容, 查看修改记录, 对比暂存的内容.**

**8. 查看当前暂存状态**

**9. 恢复a.txt 到暂存状态**

**10. 查看已提交的所有版本日志**

**11. 回退到3个版本之前, 查看文件夹的内容变化**

**12. 查看所有提交的版本log, 找到最新版本唯一标识**

**13. 前往到最新的版本, 查看文件夹内容的变化**

**重复这些操作: 熟练上午的所有命令**

**提前申请 新浪微博的账号, 下载一个微博手机APP 登录账号**

**---------------------------------**

**git最重要的功能: 分支**

**分支功能是 淘汰掉其它竞品软件的重要功能.**

**实际的问题:**

**在多人协作开发项目时:**

**假设A制作 新闻功能.**

**B同时制作 产品功能.**

**在制作的时候, 互相会有干扰. 如果同时修改同一个代码,则互相影响,无法正常执行.**

**解决办法: 分支功能**

**分支功能类似于 克隆操作, 就是把代码克隆一个出来, A在克隆体上进行修改, 此修改不会影响到 原样本.**

**B也克隆一个进行修改, 在克隆体上修改不影响其他的 克隆体和原样本.**

**当修改完毕,功能完善之后, 合并克隆体的修改到 原样本上. 之后删除克隆体即可.**

**切换到今天的 Demo文件夹下, 执行命令行**

**查看目前已有分支列表**

**$ git branch**

**默认主分支的名字为 master**

**创建分支命令:**

**$ git branch 分支名**

**例如 git branch skt**

**再次查看分支列表 $ git branch**

**切换当前操作的分支到 skt**

**$ git checkout 分支名**

**例如**

**$ git checkout skt**

**$ git branch**

**此时操作的demo文件夹就是 skt 分支的文件夹**

**创建一个 skt.txt**

**暂存修改 $ git add .**

**提交版本 $ git commit -m "xxxx"**

**切换回到 master 原样本: 查看skt.txt**

**$ git checkout master**

**练习:**

**1. 再次切换到 skt**

**2. 创建skt1.txt**

**3. 暂存+提交**

**4. 切换回到 master**

**-----------------------------**

**合并 分支 skt 的内容到主分支master上**

**切换到mater分支上**

**$ git checkout master**

**合并skt分支内容到当前分支**

**$ git merge skt**

**合并分支之后,删除分支**

**$ git branch -d 分支名**

**例如: git branch -d skt**

**再次查看分支列表: git branch**

**练习:**

**1. 查看当前分支列表**

**2. 创建新的分支: ig**

**3. 切换到ig 分支上**

**4. 再次查看分支列表**

**5. 在demo文件夹下创建 ig.txt**

**6. 暂存+提交版本**

**7. 切换回到 master分支,查看ig.txt是否存在**

**8. 合并ig分支到master分支上**

**9. 删除ig分支**

**10. 再次查看分支列表**

**分支总结:**

**分支解决的问题是开发新功能时, 在没有开发完毕之前 不完善的代码会导致项目无法执行.**

**此时可以克隆一份主分支的代码,形成一个分支**

**在分支上修改代码不会影响主分支, 当分支功能制作完毕之后.**

**通过merge直接瞬间合并到主分支上, 分支的所有代码都会复制一份到主分支上.**

**然后分支如果没用了, 需要手动删除掉.**

**------------------------------------------------**

**分支合并时产生的冲突问题:**

**来到demo文件夹下:**

**\* 创建 fpx.txt 文件, 暂存+提交版本**

**\* 创建 fpx 分支 切换到fpx分支, 修改fpx.txt第一行内容, 随便写点就行, 暂存+提交版本**

**\* 切换到master分支, 修改fpx.txt文件第一行内容, 随便写点就行, 暂存+提交版本**

**\* 合并 fpx分支到 master分支**

**\* 查看 fpx.txt 的内容**

**问题: 冲突**

**master分支改了第一行, fpx分支也改了第一行**

**合并时 git无法自动解决冲突, 则会保留两行代码 要求程序员人工手动判定如何合并.**

**删除没用的, 留下正确的 即可**

**删掉以后 暂存+提交即可**

**---------------------------------------------------**

**git网络操作方式:**

**我们通常会把代码存放在网络git服务器上, 这样就可以在任何地方来修改你自己的代码了.**

**免费的git服务器:**

**码云: 中国的,速度快, 全中文界面 个人免费**

**Github: 全球的,在中国速度稍慢, 英文界面, 个人免费. 有名**

**github的官网: https://github.com**

**先注册账号: 右上角的 sign up**

**username: 用户名, 可以填写邮箱 或者 你的姓名拼音等**

**email: 邮箱**

**password: 密码 要求15位以上 或者 8位以上的含有 小写字母和数字的**

**验证码需要把图片调整到站立状态, 点击next**

**---------------------------------------**

**码云: https://gitee.com**

**记住你的账号和密码, 一会要用到**

**做法:**

**1. 在网络git服务器上新建一个 仓库 : 相当于在网络上创建了一个文件夹,并且默认使用git来管理**

**2. 把这个网络创建的git管理的文件夹下载到本地**

**3. 在本地对文件夹进行各种操作**

**4. 推送修改到网上即可**

**找到你项目的 https 地址,**

**在任意位置开启命令行工具, 可以clone网络文件夹到你的指定目录下**

**克隆命令:**

**$ git clone 你的项目地址**

**此时有可能会:**

**\* 弹出框 要求填入你的码云账号和密码**

**\* 可能不提示弹出框 直接显示失败**

**\* 也有可能是另一种方式提示输入账号密码**

**此处必须 参考 ftp上 git\_sina 中的 git.pdf 文件**

**来解决各种各样的问题**

**重置账号密码命令 必须通过管理员模式打开 cmd 命令行工具才可以. 在开始菜单->windows系统->右键打开cmd, 选择管理员模式打开**

**重置账号密码命令**

**git config --system --unset credential.helper**

**成功时 在本地文件夹中会下载对应的网络文件夹**

**来到硬盘上,找到你克隆下来的文件夹**

**在文件夹下随便创建些文件**

**然后使用命令行 在文件夹下打开**

**执行暂存+提交版本命令**

**$ git add .**

**$ git commit -m "xxxx"**

**推送本地版本到网络服务器**

**$ git push**

**如果要求输入账号密码, 就输入以下即可.**

**然后到网页上, 刷新你的仓库, 查看是否多出来了文件**

**更新操作: 把网络git服务器上的信息下载更新到本地**

**$ git pull**

**pull 拉取**

**问题1:**

**假设你回家以后, 要把单位的代码下载到你的电脑上, 用哪个命令?**

**$ git clone xxx**

**问题2:**

**你在家里对代码进行了修改, 在睡觉之前要上传到网络git服务器**

**要执行哪些命令?**

**\* 先暂存所有修改 $ git add .**

**\* 提交成一个本地版本 $ git commit -m "xxxx"**

**\* 推送到网络服务器上 $ git push**

**问题3:**

**第二天早上来到公司, 要把昨晚上的代码更新到你的单位电脑上**

**用什么命令?**

**$ git pull 更新操作**

**-----------------------------------------------**

**云服务器**

**项目的发布方式(原始)**

**我们做完的项目最终是要放在公网上给所有的人访问的.**

**1. 需要一台计算机安装服务器程序 nodejs apache Nginx**

**2. 需要购买一个域名**

**3. 在网络运营商购买服务, 然后把 计算机存放在网络运营商提供的指定机房中**

**4. 最后把域名和链接在计算机上的网线ip地址绑定在一起, 最终实现外网的访问**

**问题:**

**- 成本非常高:计算机的成本,网络运营商的成本,安全性的成本**

**此时云服务器就出现了.**

**大型公司(阿里,腾讯,百度,新浪,华为...) 批量从网络运营商购买服务, 然后这些公司再把服务器分包给不同的小公司.**

**大型公司的网络安全性, 服务器的性能会更好. 价格也低.**

**更适合中小型企业使用**

**这些服务器的提供商有:**

**\* 新浪云**

**\* 阿里云 : 最老牌**

**\* 百度云**

**\* 腾讯云**

**\* 华为云**

**...**

**做法原理都一样:**

**1. 在云服务器上先开通一个服务器**

**2. 把你的代码传上去 就可以了 (使用git来传递)**

**新浪云官网: www.sinacloud.com**

**使用手机微博app , 用微博扫码方式登录即可**

**------------------------------------------**

**两种网站:**

**1. 纯静态网站: 选择php**

**2. 使用nodejs服务器的, 选择 nodeJS**





**弹出框 输入 1 即可**



**clone这个仓库到你的文件夹下**

**$ git clone xxxxx**

**输入账号 密码克隆到本地**

**在克隆到本地的文件夹下 书写你的代码即可,**

**要求把 index.html 一定放在根目录下**

**使用cmd命令行在此文件夹下执行:**

**$ git add .**

**$ git commit -m "xxxx"**

**$ git push**

**可以把目录中的文件推送到新浪云服务器上**

**打开新浪云提供的网址, 就可以看到你的网站了**



**通过在线编辑可以看到在网上的代码对不对**



**在ftp 上, 1M大小的 webdemo1.zip 赶紧下载**

**这是一个node服务器的小demo**

**先解压缩, 然后使用cmd命令行在 webdemo1 文件夹下打开**

**运行 $ node start.js 来启动服务器**

**在浏览器输入 localhost:5050 来访问网站**

**---------------------------------**

**把nodejs服务器的项目传到新浪云上**



**1. 先clone仓库到本地**

**$ git clone xxxx**

**2. 拷贝webdemo1 中的所有文件到 克隆下来的目录中, 确保index.html 在根目录**

**3.** 根目录下必须包含package.json文件，HTTP的端口需要监听在5050

在目录中使用命令,生成package.json

$ npm init -y

**必须要修改package.json 添加一个脚本才可以**

**"scripts": {**

**"test": "echo \"Error: no test specified\" && exit 1",**

**"start": "node start.js"**

**},**

**在命令行中执行:**

**$ git add .**

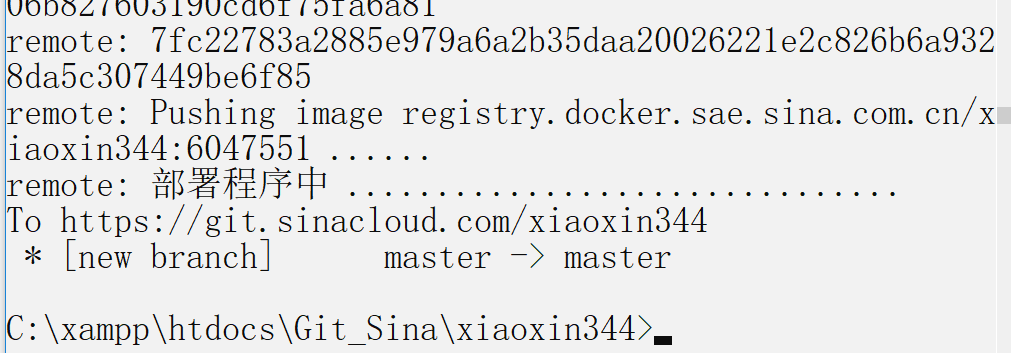
**$ git commit -m "xxx"**

**$ git push origin master**

**指定推送到 master分支的写法**

**弹出报错 就等会再试. 貌似是网络问题,也可能是网络里用的人太多导致的.**

**执行结束后**



**运行中即可! 如果运行失败, 则删除整个应用, 重新来一次整个步骤即可**



**通过网址来访问上传的网站**



