GIT 版本管理工具轻松入门笔记

课时 1：01-git 介绍和软件安装及配置

Git 简介和软件安装

1. 简介

1. 版本控制软件提供完备的版本管理功能，用于存储、追踪目录（文件夹）和文件的修改历

史，是软件开发者的必备工具，是软件公司的基础设施。版本控制软件的最高目标，是支持软件公司的配置管理活动，追踪多个版本的开发和维护活动，及时发布软件。

2. 版本管理工具主要有两个作用

1. 代码版本管理

2. 多人协作开发

3. 版本管理是一个合格的程序员的必备的基础技能

版本常用工具一个是 git 一个是 svn

4. Git 是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。

Git GUI Here Git 提供的可视化界面来操作 git。

Git Bash Here 通过命令行的方式来操作 git

git 和 svn 有哪些区别?

区别：1、SVN 是集中化的版本控制系统，而 Git 是分布式版本控制系统；2、SVN 是按照原始文件存储的，体积较大，而 Git 是按照元数据方式存储的，体积很小；3、Git 的分支操作不会影响其他开发人员，而 SVN 会影响。

2. git 安装

1. 官网 https://git-scm.com/

2. 或者通过其他软件商店下载

3. 小乌龟 TortoiseGit 安装

https://tortoisegit.org/download/

1. 身份验证和凭据存储 记得选择使用 openssh

2. 安装完以后记得设置用户和邮箱

课时 2：02-本地仓库和 git 基本概念

Git 本地仓库创建和一些基本概念

1. 使用 git init 命令创建本地仓库

右键 > Git Bash Here

git init //创建本地仓库

在 Git 目录下的修改，提交之后，就会保存到.git 文件夹里

a. git init AA 会创建一个 AA 文件夹，.git 被创建在 AA 文件夹下

b. (use "git add <file>..." to include in what will be committed)
让仓库来跟踪这个文件

```
git add a.txt

git add . //把所有修改的文件添加到暂存区
```

c. git commit -m “创建 a.txt”

```
[master (root-commit) d131f17] 创建a.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
```

d131f17 版本号 前 7 个字符
-m “创建 a.txt” 提交消息，这次做了什么修改

d. 如果 git commit 后面不写 -m “创建 a.txt”
会跳出一个 VIM 程序，让你输入提交消息
按 I 插入消息，按 Esc 退出插入模式，按 Shift+ZZ 退出 vim 编辑器

e. git log 查看仓库版本变化的历史

2. 仓库区和工作区

.git 文件夹为仓库区，类似于一个数据库存储着每一次提交的变化。

(我们在工作区对代码操作完后，把它提交到本地仓库，最后 push 到远程仓库。本地仓库存储位置位于.git 的隐藏文件夹中。)

.git（文件夹？）所在目录以及子目录称为工作区，我们在这里创建项目,和其他文件

3. 使用 git add <文件名> 可以把文件添加到暂存区,暂存区存储将要被提交的**文件变化**

4. 使用 commit 命令提交暂存区存储的变化并生成一个新的版本

5. 使用 git status 命令查看状态, git log 命令查看日志 , git reflog 查看历史记录

使用 TortoiseGit

1. 右键 > 在这里创建版本库
右键 > 提交

右键 > tortoisegit > 显示日志

任务 3：03-git 服务器和远程仓库创建

Git 服务器和远程仓库创建

1. 可以搭建,自己搭建 git 服务器
2. 免费服务器
    1. github.com 全球最大开源项目托管平台
    2.gitee.com 国内知名开源项目托管平台 码云
3. 注册 gitee 账号,并创建仓库

码云使用,创建项目

1.码云新建仓库时如果想要把仓库名称作为 Unity 项目名，怎么办？

a. 先把路径名 改为 TestProject1，跟仓库名一样。
b. GitHubDeskTop 里 clone 仓库到本地
c. Unity 里创建项目，项目名随意，创建到 TestProject1 文件夹下
d. 把工程的 Assets 这一列的文件夹，全部剪切到 TestProject1 文件夹下
e. 最后把项目名文件夹删掉

添加开源许可证：一般选择 MIT
勾选 使用 Readme 文件初始化这个仓库

Readme 文件是对仓库的说明文件，介绍项目主要是做什么的，怎么来使用这个项目

任务 4：04-从服务器克隆仓库

Git 克隆

1. https 方式

1. 第一次克隆需要我们输入账号密码

命令行的方式：

在要克隆仓库的文件夹下 右键> git bash here

git clone 仓库的 https 地址 （右键粘贴 或者 shift+insert）

图形界面的方式：

右键 > Git 克隆

2. ssh 方式

1. ssh 是一种开源非对称加密通信协议，

2. ssh 只是一种协议,有开源实现也有商业实现,git 默认使用开源实现的 openssh

3. ssh 通信需要一对密钥(公和私一对),私钥留在自己电脑上,公钥给其他的电脑

4. 使用 GitGUI 生成密钥,并把公钥放在服务器上

5. 安装完 git 第一次克隆会请求是否允许使用 ssh,需要我们输入一个"yes".或者点击 ok

3. 使用 GitGUI 生成密钥,并把公钥放在服务器上

使用 GitGUI 生成密钥步骤:

a. 右键》Git GUI Here

b. Help>show SSH key

c. Generate key Your key is in: ~/.ssh/id_rsa.pub ~代表用户目录

C:\Users\Administrator\.ssh 秘钥的目录

id_rsa 私钥

id_rsa.pub 公钥 把公钥放在服务器上

把公钥放在服务器上步骤:

a.码云 点击头像, 设置

b.SSH 公钥

c.粘贴公钥,取个标题名

4. 使用 ssh 方式克隆仓库

a.码云 克隆/下载 复制 SSH

b.右键》git 克隆 自动填写复制的 SSH 地址

如果 git 克隆没出现,需要将隐藏的.git 文件删除掉

5.秘钥作用?

a.用于加密通信

b.用来做用户的权限认证

6. 非对称加密

https://baike.baidu.com/item/%E9%9D%9E%E5%AF%B9%E7%A7%B0%E5%8A%A0%E5%AF%86/9874417?fr=aladdin

对称加密算法在加密和解密时使用的是同一个秘钥；而非对称加密算法需要两个密钥来进行加密和解密，这两个密钥是公开密钥（public key，简称公钥）和私有密钥（private key，简称私钥）。

定义

1976 年，美国学者 Dime 和 Henman 为解决信息公开传送和密钥管理问题，提出一种新的密钥交换协议，允许在不安全的媒体上的通讯双方交换信息，安全地达成一致的密钥，这就是“公开密钥系统”。

与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。

数据的加密和解密过程是通过密码体制和密钥来控制的。密码体制的安全性依赖于密钥的安全性，现代密码学不追求加密算法的保密

性，而是追求加密算法的完备，即：使攻击者在不知道密钥的情况下，没有办法从算法找到突破口。根据加解密算法所使用的密钥是否相同，或能否由加(解)密密钥简单地求得解(加)密密钥。密码体制可分为对称密码体制和非对称密码体制。

非对称密码体制也叫公钥加密技术，该技术是针对私钥密码体制(对称加密算法)的缺陷被提出来的。与对称密码体制不同，（非对称密码体制）公钥加密系统中，加密和解密是相对独立的，加密和解密会使用两把不同的密钥，加密密钥(公开密钥)向公众公开，谁都可以使用，解密密钥(秘密密钥)只有解密人自己知道，非法使用者根据公开的加密密钥无法推算出解密密钥，这样就大大加强了信息保护的力度。公钥密码体制不仅解决了密钥分配的问题，它还为签名和认证提供了手段。

非对称密码算法有很多，其中比较典型的是 RSA 算法，它的数学原理是大素数的分解。

课时 5：05-推送和拉取

添加,提交,推送,拉取

1. 添加 add (如果使用小乌龟 勾选即为 add)
2. 提交 commit
    1. 提交只提交到本地仓库,需要推送才会把变化更新到服务器仓库
3. 推送 push

右键》TortoiseGit 》 推送

把本地的变化推送到服务器

1. 如果服务器版本比我们（本地）新则不会推送成功,需要先拉取

4. 拉取 pull

右键》TortoiseGit 》 拉取

拉取服务器上的其他同事的更新变化

1. 拉取时应确保工作区整洁（文件都是绿色的勾，没有新文件，也没有红色修改过的文件）(先提交本地再拉取服务器的变化)

Github 图形软件里

Fetch origin 从服务器取来更新变化

Push origin 把本地的变化推送到服务器

课时 6：06-分支创建与合并

分支

1. master 是仓库的主分支,为了避免开发过程中程序员之间相互影响,我们一般选择创建一个新的分支来 开发新功能

2. 创建分支

右键 》TortoiseGit 》 创建分支

显示日志，可以在任意一个版本上创建分支

3. 切换分支

两种方式：

a.右键 》TortoiseGit 》切换/检出

b. 右键 》TortoiseGit 》版本分支图　　右击》切换/检出

4. 合并分支

先切换到 Master 主分支，再去合并

右键 》TortoiseGit 》合并

从其他分支合并到当前分支（Master 主分支）

5.删除分支

右键 》TortoiseGit 》版本分支图　找到要删除的分支，右击，删除

课时 7：07-解决冲突

冲突

1. 冲突如何产生的

1. 两个分支修改了同一个文件,合并的时候会发生冲突

2. 如何解决冲突

1.协商修改冲突位置,并重新提交

提交的时候，发现还是标识文件是冲突的，右键》选择 解决

将日志信息里的 #相关的内容 去掉

2.点解决　出现冲突文件列表　双击文件进去修改

3. 如何减少冲突

1. 先 pull（拉取）在修改

2. 确保自己正在修改的文件是最新版本的

3. 各自开发各自的模块,如果要修改公共文件,最好先确认有没有人正在修改

4. 不要擅自修改同事的代码

任务 8：08-忽略

忽略

.gitignore 文件　　是忽略文件

右击》Git Bash Here

使用 touch .gitignore 命令来创建一个忽略文件。

空文件夹会被自动忽略

Unity 的.gitignore 文件

#代表注释，类似 c# //

/[Ll]ibrary/

/[Tt]emp/

表示要忽略的文件夹，可以写成：

（忽略大小写，无论大写还是小写的文件夹都会忽略）

a. #按文件夹名忽略所有的文件夹

文件夹名/ （[Aa]忽略大小写）

Temp/ 忽略所有的 Temp 文件夹

b. #按路径忽略文件夹

/文件夹名(路径)/

/Temp/ 只忽略当前目录下的 Temp 文件夹

*.csproj

*.unityproj

*.sln

表示要忽略的文件

C.#忽略文件

*.后缀名

*.sln 忽略所有的.sln 文件

d. #反忽略(不忽略),对文件和文件夹都有效

!*.文件名

!A/

!*.mp3

不忽略当前目录及其所有子目录的.mp3 文件

!temp/

不忽略当前目录及其所有子目录的 temp 文件夹

!*.mp3 为什么不说 不忽略所有的.mp3 文件？可以看下面的例子。

在最外层的.gitignore 文件里 *.mp3 忽略所有的.mp3 文件,如果有个文件夹下的.mp3 文件不想被忽略该怎么办?

在这个文件夹下， 使用 touch .gitignore 命令来创建一个忽略文件，写入 !*.mp3 ，这样就不会忽略当前目录及其所有子目录的.mp3 文件

课时 9：09-版本回退

版本回退

在仓库下面右键》TortoiseGit》显示日志 》 选择一个节点右击 》 重置 Master 到此版本

在版本日志界面选择一个节点,使用重置(Reset)命令

Head 当前分支的引用指针(重置就是移动 Head 这个指针)

Index 就是缓存区（暂存区）

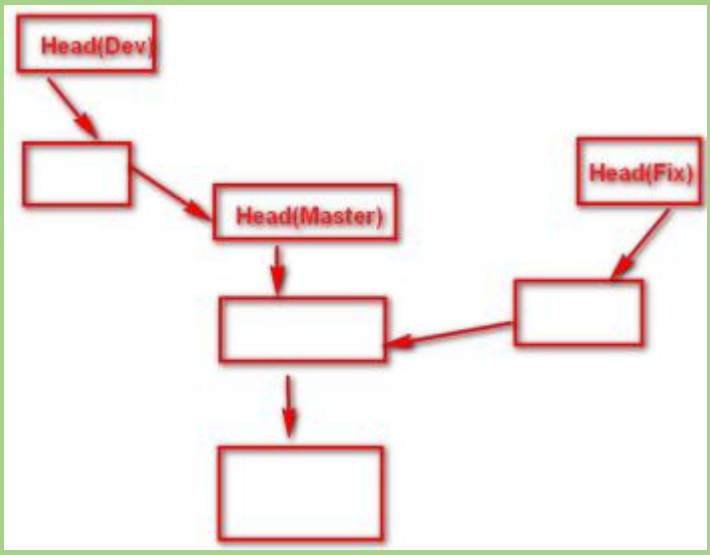
(存储？)被 Add 的(文件),也可以说是(存储？)将要被提交的,文件的快照/更改记录/变化。

将要被提交到仓库区。

工作区 .git 所在的目录以及子目录

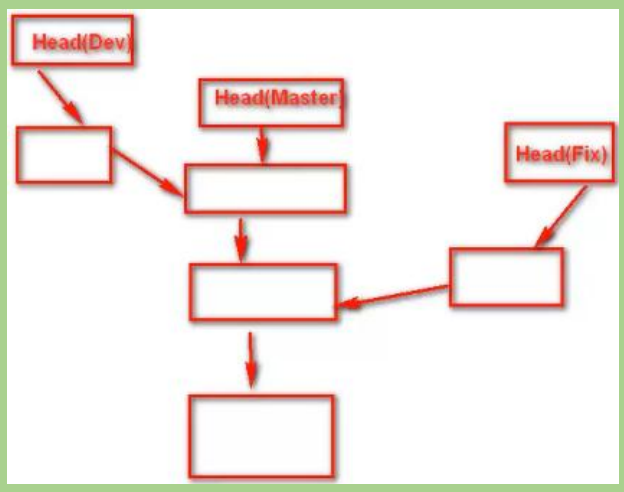
git add . //把所有修改的文件添加到暂存区

使用 git add <文件名> 可以把文件添加到暂存区,暂存区存储将要被提交的文件变化(文件的快照)

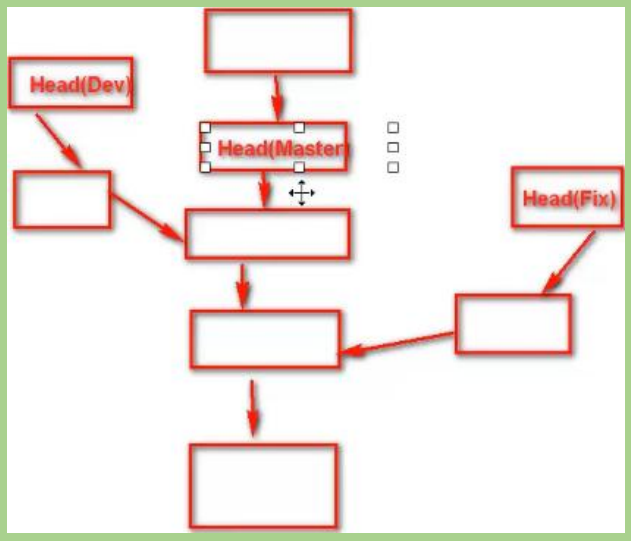


中间主分支，右边修 bug 的分支，左边开发新功能的分支。
每一个分支都有一个 head
切换分支的时候，其实就是切换 head 所在的节点（版本？）

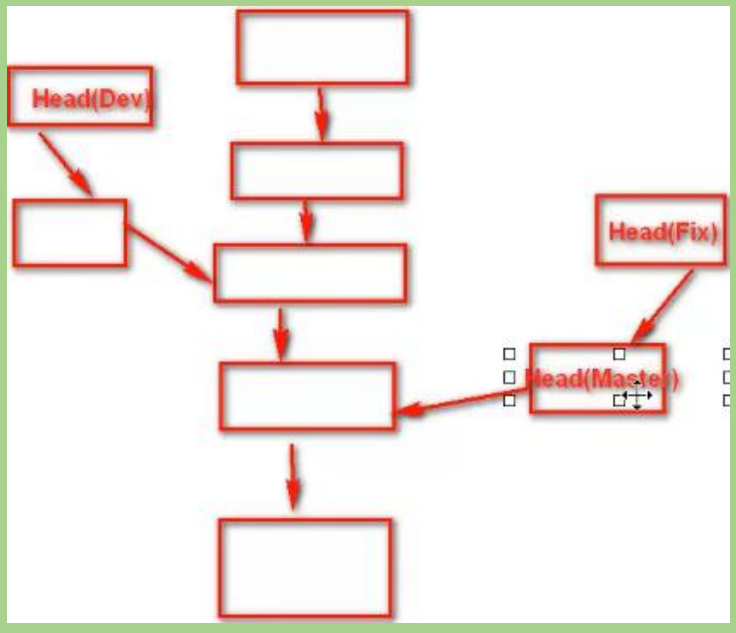
主分支 Master 上有一次新的提交，形成了一个新的版本。新版本会指向之前的版本。Head 会移动到新版本的位置



重置就是把 head 移到之前提交过的某个节点（版本） 如下图。



或者可把 head 移动到其他分支的节点上，这样 head（Master）就变到了另外一个分支上面去了。（切换分支？）这个分支就变成了主分支。



重置当前分支从 master 到：

分支　　某个分支

标签　　对某个版本打的标签

提交　　就是我们当前选中的版本

重置类型（三种重置）

软重置(S): 不更改工作区和索引

混合(M):　保持工作区不变，重置索引文件

硬重置(H): 重置工作区和索引(丢弃所有本地变更)

不改变工作区文件内容。

索引文件，指的是暂存区里的文件。

重置索引文件：清空暂存区里的内容

重置工作区：把工作区里的文件全部还原到这个版本的状态

如何重置版本？

显示日志，选中提交的版本，右击，重置 master 到此版本。

一般设置里选择，重置当前分支从 master 到 提交。重置类型为硬重置。

但是这种方式有一个缺点，如果重置错了，无法把重置掉的内容恢复。

可以在重置之前，先给想要重置的版本打个标签，这样想要撤销时就

很方便。

如何打标签？

在版本上 右击 在此版本上创建标签。 右击 删除，可以删除标签。

如果想要恢复打了标签的版本,在任意一个版本上,右击,重置 master 到此版本。选择对应的标签。硬重置。

课时 10 : 10-子模块

子模块就是一个 git 仓库包含其他 git 仓库

当我们需要把其他开源项目作为本项目的一个库,并可可以随时拉取更新的时候

或者我们的项目需要分模块独立开发的时候可以使用子模块

克隆有子模块的仓库,需要选择递归

TortoiseGit 》 添加子模块

版本库里可以填 仓库的 https 或 ssh

路径可以为默认，或者自己复制绝对路径进去

添加子模块后会出现.gitmodules 文件，里面记录了子模块的名称，路径，仓库地址

推送的时候不会把子模块推送上去。

课时 11：11-Issue 和 PullRequest

git 非常灵活可以根据自己的需要构思出来不同的工作流(团队开发的方式和流程)

issue 讨论。用于跟踪待办事项、bug、功能需求等。

PullRequest 可以帮助您与他人协作编写代码。拉取请求,为项目贡献代码,请求项目管理者合并自己的代码

Fork 把别人的仓库复制到自己的账号里

其他账号如何 PullRequest?

就当是自己创建分支然后开发完毕,然后合并到主分支再推送到服务器的 newfunction

新建 PullRequest，创建

仓库作者，可以对 PullRequest，审查，测试，合并分支，接受 PullRequest

Github 图形软件里

Fetch origin　　从服务器取来更新变化

Push origin　　把本地的变化推送到服务器