Git & Github

集中式版本控制软件：SVN

分布式版本控制软件：Git（优点：避免单点故障）

**1.** Git功能：

(1)协同修改

(2)数据备份（保存每一个提交的历史状态）

(3)版本管理（不保存重复数据，SVN：增量式保存；Git：文件系统快照）

(4)权限控制（Git可接受非授权人士的代码并审核，SVN不能）

(5)保存历史记录

(6)分支管理

**2.** Git优势：

(1)大部分在本地完成（在本地有本地库）

(2)完整保存性（hash）

(3)与Linux命令全面兼容

**3.** Git本地结构：

本地库(历史版本)

git commit

暂存区(临时储存)

git add

工作区（代码）

**4.** Git代码托管中心（维护远程库）：

(1)局域网：GitLab服务器

(2)外网：github、码云

**5.** 本地库和远程库

* 团队内协作

代码托管中心

远程库

pull push clone push(加入团队)

本地库

本地库

* 跨团队协作

代码托管中心

**merge** **审核** **pull request**

**fork**

远程库

远程库

pull

push pull clone

本地库

本地库

本地库

(非团队人员)

**6.** Git命令行操作：

(1)本地库初始化：**git init**

(2)设置签名：

* 形式：
* 用户名：tom
* Email地址：….@....com
* 作用：区分不同开发人员的身份(这里的签名与登录远程库的账号和密码没有关系)
* 命令：
* 项目级别/仓库级别：仅在当前本地库范围内有效

**git config**

**(eg. git config user.name tom)**

信息保存位置：./.git/config （查看命令:cat .git/config）

* 系统用户级别：登录当前操作系统的用户范围

**git config –global**

信息保存位置：~/.gitconfig

(3)状态查看：**git status**

(4)将工作区新建或修改添加到暂存区：**git add [file name]**

从暂存区移除文件：**git rm –cached [file name]**

(5)将暂存区内容提交到本地库：**git commit [file name]**（此时会进入到vim编辑器）

**git commit -m “commit message” [file name]**

(6)查看历史记录：（空格下翻页，b上翻页，q退出）

只能显示历史版本：

**git log**

**git log –pretty=oneline**

**git log –oneline**

可显示所有版本：

**git reflog**

(7)版本前进后退：

* 基本索引值操作：**git reset –hard [局部索引值]**
* 使用^符号：
* 命令：**git reset –hard HEAD^^**
* 说明：a. 异或符号数量即表示版本后退数目

b. 只能版本后退，不能版本前进

* 使用~符号：
* 命令：**git reset –hard HEAD~3**
* 说明：a. 波浪线后的数即为版本后退数目

b. 只能版本后退，不能版本前进

* reset命令的三个参数：

**--soft** 仅仅在本地库移动HEAD指针

**--mixed** 在本地库移动HEAD指针；重置暂存区

**--hard** 在本地库移动HEAD指针；重置暂存区；重置工作区

**7.** 文件比较

将工作区文件与暂存区比较：**git diff [文件名]**

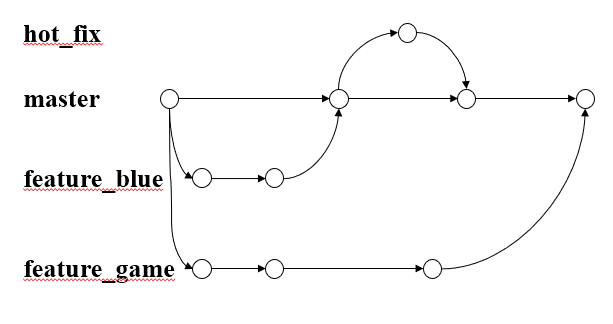
将工作区文件与本地库记录比较：**git diff [本地库指针] [文件名]**

不加文件名则比较所有文件

**8.** 分支管理

(1)分支：在版本控制中，使用多条线同时推进多个任务

分支结构：



(2)好处：并行推进多个功能开发；各分支独立开发

(3)创建新分支：**git branch [分支名]**

查看所有分支：**git branch -v**

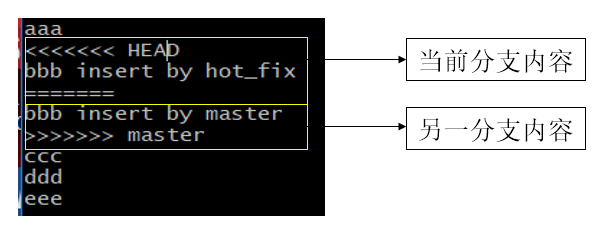
切换分支：**git checkout [分支名]**

合并分支（必须先切换到接受修改的分支上）：**git merge [另一个分支名]**

(4)冲突解决：

* 冲突出现：合并时相同文件的同一行，内容不同

冲突表现：



* 冲突解决：
* **编辑文件，删除特殊符号，并修改文件**
* **git add [文件名]**
* **git commit -m “commit message”**（不能带文件名）

**9.** Git基本原理：

* 哈希算法（加密算法）

特点：(1)不论输入数据多大，输入同一个哈希算法，得到的加密结果长度固定。

(2)哈希算法确定，同一个数据的加密结果相同。

(3)哈希算法确定，输入数据有变化，输出一定有变化，且一般变化较大。

(4)哈希算法不可逆。

* Git底层采用SHA-1算法。
* Git分支管理的本质是创建和移动指针。

**10.** Git远程：

(1)新建远程地址别名：**git remote add origin [远程库地址]**

(2)查看远程地址别名：**git remote -v**

(3)推送本地库到远程：**git push [远程库地址别名] [本地库分支名]:[远程仓库名]**

(4)克隆：**git clone [远程地址]**

克隆效果：完整的把远程库下载到本地；创建远程地址别名；初始化本地库

(5)拉取：pull = fetch + merge

**git fetch [远程库地址别名] [远程分支名]**

**git merge [远程库地址别名/远程地址名]**

**git pull [远程库地址别名] [远程分支名]**

(6)冲突解决：如果不是基于Github远程库最新版本所做修改，团队协作时需先拉取最新版本，否则不能推送。

(7)移除remote： **git remote rm origin**

**11.** SSH登录（只能为一个账号创建）：

* 操作：

(1)进入家目录：cd ~

(2)创建ssh目录：**ssh-keygen -t rsa -C [email]**（之后一直打回车，保持默认设置）

(3)进入ssh目录：cd .ssh/

(4)cat id\_rsa.pub 复制其中内容

(5)复制到Github上SSH and GPG keys（New SSH key→title随意填[如：mykey]，将内容粘贴到key→Add SSH key）

(6)**git remote add origin\_ssh [SSH远程地址]**

(7)推送测试

* 其他功能：

删除已有ssh目录：rm -r .ssh/

**12.** 给其他软件添加忽略文件：

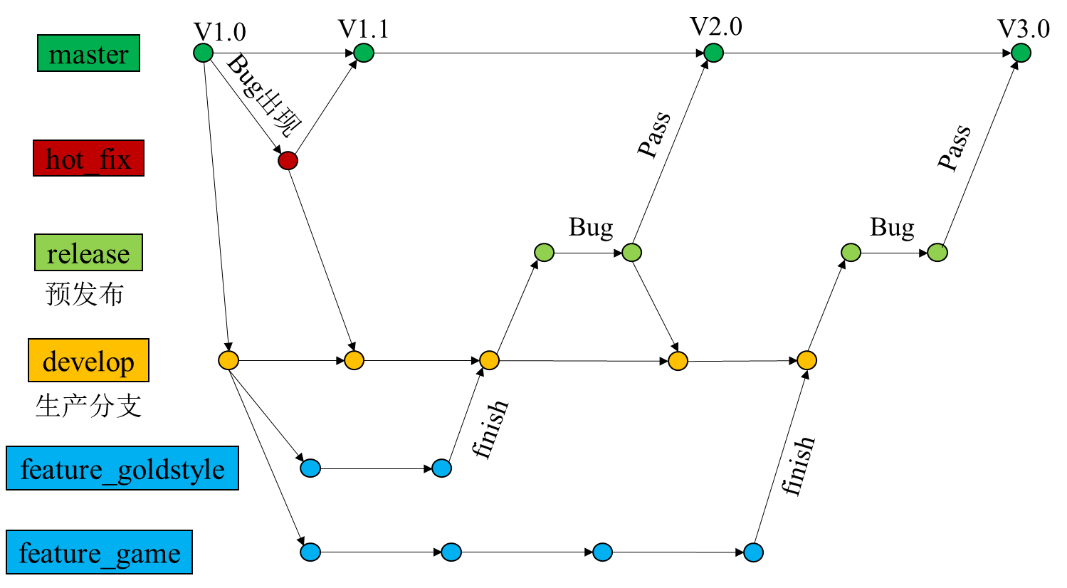
* 操作：

1. 进到[http://github.com/github/gitignore](http://github.com/github/gitignore选择)
2. 在网站里找到所需要忽略的类型文件
3. 与.gitconfig同位置下创建上述文件，并在其中添加自己额外的忽略文件类型
4. 在.gitconfig文件中添加如下代码：

|  |
| --- |
| [core]  excludesfile=[上述文件的绝对路径]（路径中斜线必须使用正斜线“/”） |

**13.** Git工作流

* 集中式工作流（无分支，很少用）
* GitFlow工作流（多用）



* Forking工作流（虚拟团队多用）

14.