***读《Git版本控制管理》***

是一款免费、开源的分布式版本控制系统，最早由LinilusTorvalds创建，用于管理Linux内核开发，现已成为分布式版本控制的主流工具。

1. 背景

VCS：版本控制系统

SCM：源代码管理器

RCS：修订控制系统

目的：开发以及维护开发出来的代码、方便读取代码的历史版本、记录所有的修改。

SCCS：源代码控制系统

端点对端点的模型

散列指纹

1. Cygwin版本的Git

下载地址：http://cygwin.com/

msysGit独立的git

下载地址：https://git-for-windows.github.io/

关于Git的配置问题：

(1)git config --global user.name ”你的名字”

(2)git config --global user.email “你的邮箱”

其中，global是指全部推送的名字，如果在某个项目下可以去掉global，单独设置你的名字和邮箱。

1. ssh-keygen -t rsa;

创建密钥，所有的选项都设置为空，可以在用户文件夹下看到ras文件

1. 首先，你要有自己的github的账号，然后，将你的密钥添加到账号的ssh中。（设置-ssh）
2. Git命令行
3. Git使用快速入门

两种版本库的基础技术：（1）从头开始创建：

1. 复制一个已有的版本库：

创建初始版本库：

（1）创建：mkdir 文件

（2）进入文件：cd 文件

（3）初始化：将任何目录转化为Git版本库：git init

将文件添加到版本库中：

1. git add 文件（只是暂存staged）
2. git status 显示中间状态
3. git commit -m “......”，其中“......”是指:必须提供日志消息和作者
4. Bash：export GIT\_EDITOR=vim，选择设置你喜欢的编辑器

Tcsh：setenv GIT\_EDITOR=emacs

git status 命令显示没有突出的、暂存的变更需要提交

配置提交作者：在对版本做多次提交之前，你应该建立一些基本环境和配置选项。最为基本的是，Git必须知道你的名字和email地址。

还可以使用：GIT\_AUTHOR\_NAME和GIT\_AUTHOR\_EMAIL环境变量来实现。

再次提交：

1. git log
2. git show
3. git show-branch --more=10

查看提交差异：

1. git diff

版本库内文件的删除和重命名：（添加和删除都要跟两步）

git rm 文件名

git commit -m“Remove a 文件名”

创建版本库副本：

cd

git clone public\_html my\_website

配置文件：

1. ：基本的Git概念

4.1基本概念

4.1.1版本库

Git版本库只是一个简单的数据库，其中包含所有用来维护与管理项目的修订版本和历史的信息。

Git版本库提供所有文件的完整副本，还提供版本库本身的副本。

Git在每个版本库里维护一组配置值。

Git维护两个主要的数据结构：对象库和索引。

所有这些版本库数据存放在工作根目录下一个名为.git的隐藏子目录中。

4.1.2 Git对象类型

对象库是Git版本库实现的心脏。它包含你的原始数据文件和所有日志信息、作者信息、日，以及其他用来重建项目任意版本或分支的信息。

对象库的对象只有4个类型

Blob：块，

Tree：目录树，

Commit：提交，

Tag：标签，

随着时间的推移，Git把对象压缩并存储在打包文件里，这些文件也在对象库中。

4.1.3 索引

是一个临时的、动态的二进制文件，它描述整个版本库的目录结构。索引捕获项目在某个时刻的整个结构的一个版本。

Git的关键特色之一就是它允许你用有条理的、定义好的步骤来改变索引的内容。

4.1.4 可寻址内容名称

Git对象库被组织及实现成一个内容寻址的存储系统。

对象库中的每个队形都有一个唯一的名称。向对象的内容应用SHA1得到SHA1散列值。160位数，通常表示为40为16进制的数。

4.1.5 Git追踪内容

主要表现为两种关键的方式

1. 基于对象内容的散列计算的值，git追踪的是内容而不是文件。
2. 当文件从一个版本变到下一个版本的时候，git的内部数据库有效地存储每个文件的每个版本，而不是他们的差异。

4.1.6 路径名与内容

Git仅仅记录每个路径名，并且确保能通过它的内容精确地重建文件和目录，这些是由散列值来索引的。

4.1.7 打包文件

Git使用了一种叫做“打包文件（pack file）”的更有效的存储机制。

4.2 对象库图示

4.3 Git在工作时的概念

4.3.1 进入.git目录

使用 git init来初始化一个空的版本库

# 列出当前目录中的所有文件

find .

底层：plumbing

4.3.2 对象、散列和blob

当创建一个对象的时候，git不关心文件名，只关心文件里面的内容。

4.3.3 文件和树

Git通过另一种叫做tree目录树的对象来跟踪文件的路径名。

使用 git add的时，不会马上为树创建一个对象，相反索引更新了。为与.git/index中，它跟踪路径名和blob。每次执行命令的时候，都会使用心得路径名和blob信息来更新索引。

4.3.4 对Git使用SHA1的一点说明

1、每次对相同的索引计算一个树对象，它们的SHA1散列值是完全一样的。

只有两个不同的对象产生一个相同的散列值时才算碰撞。

1. 散列就好似对象的可靠标签或名称

它通过原始提交的散列值唯一标识整个数据结构在提交时的状态。

散列函数的强大应用：它提供了一种有效的方法来比较两个对象，甚至是两个非常大二复杂的数据结构，而且并不需要完全传输。

4.3.5 树层次结构

4.3.6 提交

4.3.7 标签

尽管git已经实现了一种标签。但是有两种基本的标签类型

轻量级的：lightweight

带附注的：annotated

轻量级标签只是一个提交对象的引用，视为私有，不存在与版本库中。

带附注的标签更加充实，并且会创建一个对象。

1. 文件管理和索引

5.1关于索引的一切

5.2 Git中的文件分类

3类：已追踪的、被忽略的以及未追踪的

5.3 使用git add

5.4 使用git commit 的一些注意事项

5.4.1 使用git commit --all

会导致执行提交之前自动暂存所有未暂存的和未追踪的文件变化，包括从工作副本中删除已追踪的文件。一个全新的目录，目录下没有任何文件名或路径，-all不能将其提交。

5.4.2 编写提交日志消息

5.5 使用 git rm

5.6使用 git mv

Mv 文件名 新文件名

Git rm 文件名

Git add 新文件名

等同于：

Git mv 文件名 新文件名

Git log --follow 新文件名

会让git 在日志中回溯并找到内容相关的整个历史记录。

5.7 追踪重命名注解

5.8 .gitignore文件

1. 提交

6.1 原子变更集

在底层对象模型方面，原子性是有意义的：一张提交快照代表所有文件和目录的变更，它代表一棵树的状态，而两张提交快照之间的变更集就代表一个完整的树到树的转换。

Git实现原子性操作的关键原因之一：它允许你根据一些最佳实践建议来设计你的提交。

6.2 识别提交

6.2.1 绝对提交名

散列ID是全局唯一的

6.2.2 引用和符号引用

消除二义性的启发式算法

6.2.3 相对提交名

分支的头

6.3 提交历史记录

6.3.1查看旧提交

6.3.2 提交图

6.3.3 提交范围

6.4 查找提交

6.4.1 使用git bisect

一般基于任意搜索条件查找特定的错误提交

唯一搜索要求就是，在给定版本库的一个检出状态时，你能够确定它是否符合你的搜索需求。

在使用git bisect 命令时，你需要首先确定一个好的提交和一个坏的提交。

运行git bisect命令通常为了找出某个导致版本库产生倒退或bug的特殊提交。

至关重要的是你要从一个干净的工作目录中启动 git bisect。

6.4.2 使用git blame

6.4.3 使用Pickaxe

1. 分支

7.1 使用分支的原因