**Git分布式开发**

版本的控制系统

Git的架构原理： 1.基本概念 1. 快照而不是比不同

2.几乎所有的操作都是本地的

3Git文件的三个状态

4.文件状态生命周期

安装git及helloworld 安装上官网

First time

开始前理解好需要理解的概念

Svn 必须学好

Git必须好好学下：分布式管理系统

下源代码

###### 什么选择git 什么svn

学会使用svn gui工具，理解svn架构，不要求对svn做很深入的理解，使用就好

学会使用Git工具，并且了解git的网络结构，使用就好

理解个版本控制工具的区别，以及git的优势所在

四大开源站点

熟练使用git来进行版本控制管理，但不要求成为git版本仓库管理员，掌握git常规操作，git的分支操作，git 的远程同步

问题：

Svn原子提交的理解

Git分布式本班控制概念理解

###### Git产生及特点：

Git是linux的创始人 linus 开发的开源和免费的版本管理系统，也是源代码系统scm，它的官方网站是 <http://git-scm.com>

2005年的时候linux核心开发小组和当时版本管理系统提供商出现分歧，就开发出来git。Git已经成为linux核心，Eclipse，andrioid，Ruby on Rails，jQuery等

Git的特点：

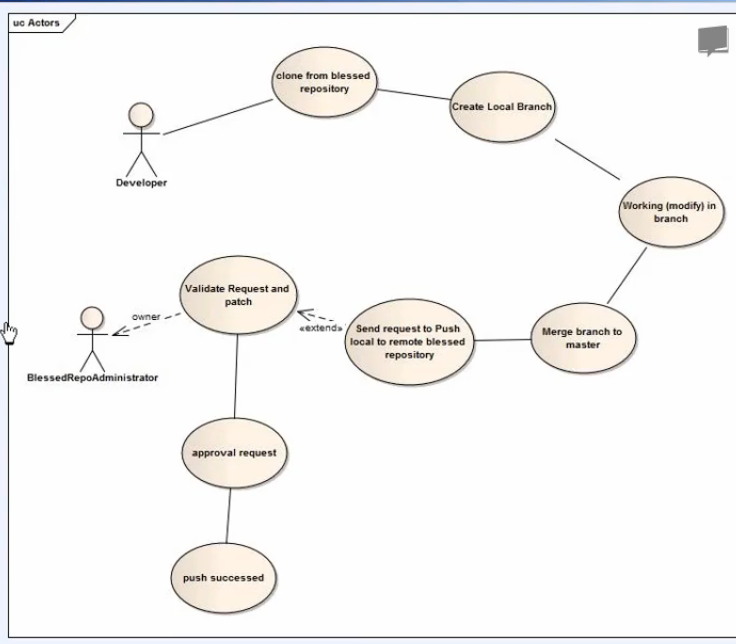
速度 c编写

设计简单

非线性开发

分布式开发？ 提交到本地

第五“能够管理大型项目



版本创库的管理员

每个都是独立，

当你远程

###### git开发一天

一般开发者，我就google source code

1. 从服务器上克隆数据库（包含源代码+版本信息）到单机上
2. 在自己的机器上创建分支，修改代码
3. 在单机上自己创建分支上提交代码
4. 在单机上合并分支
5. 新建一个分支，把服务器上最新版的代码fetch下来，然后跟自己的主支分支合并
6. 生成补丁（patch），把补丁发送给主开发者
7. 看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突，就会要求先解决冲突，然后再由其中一个人提交，如果主开发者可以自己解决，或就没有，就通过
8. 一般开发者之间解决冲突的方法，开发者之间可以使用pull命令解决冲突，解决完冲突之后再向主开发者提交补丁

主开发者的角度（假设主开发者不用开发代码）

1. 查看邮件或通过其他方式查看一般开发者的提交
2. 打上补丁，解决冲突

开始后了解git 系统环境变量：

/etc/gitconfig 存储为用户名等信息

1. 设置身份

$ git config –global user.name “Wuliang”

$ git config –global user.email [11@qq.com](mailto:11@qq.com)

1. 设置编译器（可选）

$ git config –global core editor emacs

1. 设置你的比较工具

$ git config –global merge tool vimdiff

1. 检查你的配置 $ git config –list
2. 帮助 $ git help 命令 / $ git 命令 help

###### 四大仓库

Blessed（remote）repository 远程仓库

Local repository 本地仓库

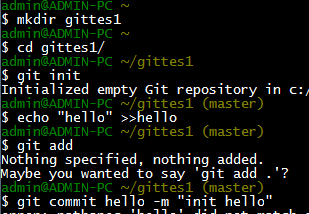
Stage area 临时工作区

Work area 工作区

在分布式开发中，不再是树形结构，可能是网状结构，远程服务器最终发布的代码

，电信机房或公司机房，同时本地也有一个版本local，local中有一个工作区work area ，而其中有一个临时缓冲区就stage area

###### 本地创建项目和四个区域分别在哪？



1. Mkdir gittes1 创建文件夹，只会在创建C:\Users\admin\gites1 目录或在D:\Git\Git\gites1文件夹，是空文件
2. Git init /wu 创建 .git文件

$ git init /wu

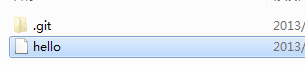
Initialized empty Git repository in D:/Git/Git/wu/.git/ 文件夹

**产生的.git这就是我们的版本仓库**

**同时也是工作区**

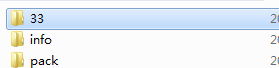
存档的东西是放在object内

1. $ echo “hell0” >>hello 将”hell0” 写入到hello文件中去，说明在在.git工作区域



**Hello文件就是 work area**

1. $ git add . 注意 “.” 包含所有，提交到index



没有真正提交到我们的版本仓库，只是index临时版本仓库，

add . 时候创建**.git/index 临时文件夹**  也创建了 **.git/objects/33文件夹为本地文件夹**，都是二进制文件

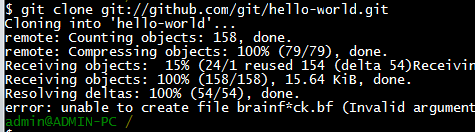
1. **$ git commit -m “init hello” 提交 和 增加日志文件**

**会在本地文件中加入文件,提交到本地区**

###### 远程使用git项目步骤：





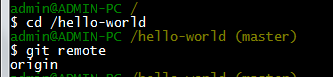


1. $ git clone git://github.com/git/hello-world.git 克隆远程仓库代码：

下载远程，在d/git中

现在就有本地仓库与远程仓库的联系

1. $ git remote remote仓库的信息



显示出远程文件程序名



显示文件地址

添加命令： ls \*.sundy 为查询出所有的后缀名为sundy的文件

Git add \*.c 为增加所有后缀名为c的文件

git add README 增加README文件

git commit –m “helll” 提交

git status 查看状态，及时跟踪文件状态

<https://github.com/anpingxiong/ourHr>

我们想忽略一些系统文件夹（bin，tmp等）：

1. Echo “bin” > .gitignore 创建文件.gitignore 并写入文bin
2. 将.gitignore 放入文件中.gitignore，只能手动放，而命令放入，只能放入一个，不知为啥
3. Git add .
4. Git status 可以查看到只有bin1.dll，而没有bin.dll

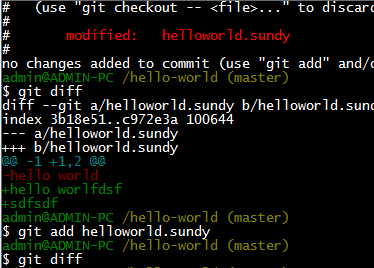
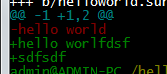
**注意：验证过，所以当使用java是，bin上面的东西就不必提交到远程服务器中去**

**比较区域之间不同：diff：**

$ git diff –staged :比较workpace VS staged

$ git diff –cached staged VS locat

1. $ git diff 命令默认为比较工作区与临时区的不同
2. 看不出来不同时，需修改数据，比如在 helloword.sundy vim hellowrd,sundy,写入，改变，“：wq”退出
3. 再$git diff ，就出现

可以通过查看状态来查询，显示出modifie，说明没提交到staged就是临时区，只有 “$git add helloworld.sundy” 才会提交到staged（临时区）

1. $ git diff –-cached 有出现，上右图
2. $ git commit helloword.sundy 才从staged（临时区）提交到（local）本地，才真正diff不显示效果

所以说，修改的文件或信息都需要add commit 才真正修改了

$ Git reset 追溯到前面：软恢复

##### 文件删除与复制：rm,checkout

1. $ echo “bin1” >> bin1.dll 创建文件bin1.dll，并写入bin1

**,创建到work area（工作区），删除不到，必须在.git文件内才能被**

**git rm bin1.dll,不然就不行了**

1. **$ git add bin1.dll 提交到index（临时区），用git rm bin1.dll删除不了，而是用可以rm bin1.dll删除，并且可以用$ git checkout –bin1.dll，回滚出来**
2. **$git commit –m “deteled bin1” 提交到（local）本地区**
3. $ git rm bin1.dll 删除了，当你删除时,不需要checkout了，回滚不了

**git checkout**：Git的checkout有两个作用，其一是在不同的branch之间进行切换，例如 'git checkout new\_branch'就会切换到new\_branch的分支上去;另一个功能是还原代码的作用，例如'git checkout app/model/user.rb'就会将user.rb文件从上一个已提交的版本中更新回来，未提交的内容全部会回滚

###### **注意**：

1. 命令加git与不加git的区别？

例如：cat bin1.dll 查看（work area区）中文件，而git cat bin1.dll，git中没有cat的方法，而且 加git是关于.git文件夹操作（同时作用工作区中的文件），不加git一般是工作区中操作

**2**. $ rm bin1.dll 与 $ git rm bin1.dll 区别

**其中$ rm bin1**.dll 是删除不用提交上 .git文件中的工作区中的文件，而$ git rm bin1.dll必须提交才能本删除的命令，并且把工作区中文件也删除了

###### **文件的移动与复制 mv**

$ mv 移动文件 移动后的文件（可以不存在）

Git log 查看日志

Git whatchanged 查看比较详细

##### 远程：

1. 如何远程服务：(一般远程仓库)

克隆 clone版本仓库

Pull 就是当地与远程合并（git pull origin）

Fetch 不合并，建立新的分支(git fetch origin)

Push (git push origin master（分支名）) 失败，没有权限

**$ git format-patch origin/master 在文件中加补丁文件，这个补丁打完后在工作区中，把补丁文件用email发送到管理员，管理员通过才能**

Remote：

列出远程仓库

Git remote

Git remote –v 显示远程地址

添加远程仓库

Git remote add pb git：//github.com/sundyhome/ticgit.git

Git remote rename pb paul

Git remote rm paul

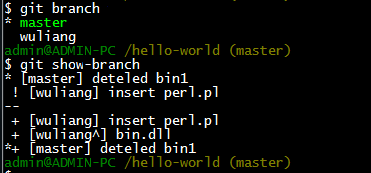
分支管理与合并：

1. 创建分支：$ git branch 分支名（wuliang）
2. 进入分支：$ git checkout wuliang 就进入了分支wuliang

在其中更改文件的时候，例如：bin.dll 其中修改了，在两个分支都可以看到改变，？？？？？

这个是由于没有提交，所以分支改变是在工作区（work aera）中,所以分支是一样的，区域，而其中git commit perl.pl，才会分支有不同

1. $ git branch 显示分支目录
2. $ git show-branch 显示出分支的差异



1. $ git diff master wuliang 显示两个分支的不同

Merge：

1. Git merge “merge branch1 to master” HEAD branch1 （git merge “merge” HEAD wuliang） 合并分支

HEAD 代表最新版本，合并到最新版本

其中需要当两个分支都修改同一个文件时，其中 merge 而导致合并不成功，当vim

修改这个文件，把不需要删除，才重新提交才可以

另一种： git checkout master

Git pull . branch1

<http://www.codeplex.com/> 用的是水星和微软 开源

<https://code.google.com/intl/zh-CN/>

<https://github.com/>

在win7下使用git特殊性：

## git did not exit cleanly (exit code 128)

My colleague solved the problem. Steps:

1. Right click -> TortoiseGit -> Settings -> Network
2. SSH client was pointing to C:\Program Files\TortoiseGit\bin\TortoisePlink.exe
3. Changed path to C:\Program Files (x86)\Git\bin\ssh.exe

Ssh.exe

SSH Secure Shell 安全外壳协议  
首先你装完LINUX系统之后一定会接触这样一个术语:SSH,他的全称是"（SSH: Secure Shell Protocol）" SSH：安全外壳协议.  
通过使用SSH，你可以把所有传输的数据进行加密，这样“中间人”这种攻击方式就不可能实现了，而且也能够防止DNS和IP欺骗。还有一个额外的好处就是传输的数据是经过压缩的，所以可以加快传输的速度。SSH有很多功能，它既可以代替telnet，又可以为ftp、pop、甚至ppp提供一个安全的“通道”。