SVN：

SVN Server

SVN Client

Repository

Working directory

VISUALSVNSERVER

TortoiseSVN

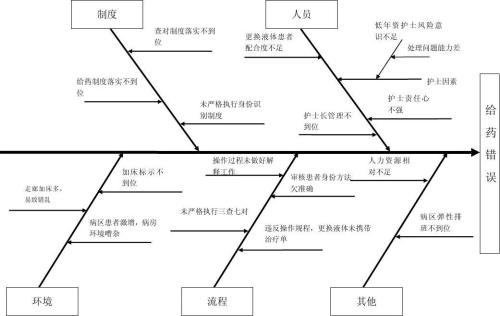
Checkout

.svn(包括服务器信息，子文件信息，保持和服务器沟通)

1. 建立SERVER和CLIENT之间checkout respository
2. 客户服务端操作，粘贴后添加add,commit；在空白处右键add,commit即可大批量操作
3. Checkout是第一次检出时用，往后用update,commit
4. 增删改
5. 一个新的文件夹单独加入服务器中（import）
6. 导出（EXPORT）不受版本控制，无.svn文件夹
7. 每个受控制的文件夹下都有.svn文件夹
8. Show log显示日志，查看版本
9. 版本是所有的文件都修改所造成的版本递增
10. 显示日志可以看到此文件的修改记录
11. Diff比较版本的不同，revision graph图形化显示
12. 还原到以前的版本：update to revision

Showlog

1. 文本值只保存增量
2. Repo-browser仓库浏览
3. 建立分支，正在开发中，A小组开发，在此软件加入其它特殊功能开发，B小组负责，A负责主线，鱼骨图



Branch，文件夹是不变的，merge合并分支

1. out of date超时，A，B修改不同行，A提交后,B修改后再次提交时A,B修改自动合并（merge）

A,B修改同一行，出现冲突（conflict），会出现冲突文件，mind（本地文件，更新前的文件），r8（第8个版本的文件），r9（服务器上最新的）文件，手动合并（在冲突文件中）的文件提交

1. （SVN备份，可以压缩）本地更新最新后，再执行EXPORT导出，不受版本控制，可以再次创建压缩包进行备份

导入（import）

导出（export）

检出（checkout）

检入（checkin）

修改（edit）

提交（commit）

更新（update）

新增目录和文件（add）

删除目录和文件（delete）

清理（clearup）

锁定（lock）

解锁（unlock）

1. SVN配置管理工具高级应用工具：ViewVC Mantis Trac

练习建立版本库：

URL: <https://DESKTOP-KNOFSUC/svn>

GIT:分布式版本控制系统

快速；容量大。。。bless远程的，一般说的是本地的版本库

一般开发者：

1. 从服务器上克隆数据库（代码和版本信息）到单机上
2. 在自己的机器上创建分支，修改代码
3. 在单机上自己创建的分支上提交代码（提交到本地）
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，然后跟自己夫人主分支合并
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，如果著开发者发现两个一般开发者之间又冲突（他们之间可以合作解决冲突），就要求他们先解决冲突，然后再由一人提交。如果主开发者自己可以解决，或者没有冲突，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（管理员）:

1. 查看邮件或者通过其他方式查看一般开发者的提交状态
2. 打上补丁，解决冲突（可以自己解决，也可以要求开发者之间解决以后再重新提交，如果是开源项目，还要决定哪些补丁有用，哪些补丁不用）
3. 向公共服务器提交结果，然后通知所有开发人员

Git几乎所有的操作都是本地的；git的三个状态：working directory正在工作区里，staging area临时存储空间，git directory（resopitory）本地版本仓库

GIT Bash命令行

GIT GUI（一般不用，用tortoi）

Blessed(remote) repository远程仓库

Local repository本地仓库，在文件夹里的object里

Stage area临时缓冲区（添加但未commit）

Work area工作区

分布式是网状结构，但也可以指定一个远程服务器blessed

作为git的个体工作者：1. 初始化和建立项目：init clone

2. add commit

3. status

4. lgnore file

5. diff

6. reset

7. rm，mv

Merge

将git的默认文件夹位置进行修改，可以在gitbash图标右键，属性，去掉--cd-to-home，修改“起始位置”为自定义的git 本地仓库的路径，如：F:\Workspace\KYGitLabel

有时需要验证身份，

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

1. 在本地建立一个仓库：在gutbush里输入cd learngit(建立一个版本库，并命名为learngit)

在本地找到learngit，在文件目录下添加一个readme.txt文档，返回gitbush，输入git add readme.txt（使readme文档加入到git中），再输入git commit –m”wrote a readme file”（将readme文档进行提交，-m后面为提交的注释），出现1 file changed, 2 insertions(+)两行命令，

一个文件被改动（新添加的readme.txt文件），插入了2行内容（readme.txt有2行内容）

1. git status命令用于显示工作目录和暂存区的状态。使用此命令能看到那些修改被暂存到了, 哪些没有, 哪些文件没有被Git tracked到。git status不显示已经commit到项目历史中去的信息。看项目历史的信息要使用git log.
2. 在learngit目录下新建一个start.txt文件有2种方法：一，在本地仓库直接进行建立。二，在gitbush里输入echo “learngit” >start.txt
3. 查看是否有未受控或未提交的文件使用git status,建议每次使用时都查看是否有未完成操作
4. 建立忽略文件时，采用echo “start.txt” >.gitignore表示start文件是要忽略的文件
5. 比较不同，diff staged（比较workspace和staged的不同）和diff cached（比较staged和local repo的不同）默认git diff
6. 在gitbush里直接编辑，可输入vim readme.txt,输入i编辑，esc退出编辑，shift+:wq返回，编辑后需要进行提交哦
7. Git reset回复（不常用）
8. 删除rm，移动mv，git rm test1.txt , 仓库检出git checkout test1.txt

删除但是未提交时，可以用checkout检出，步骤：git rm helloworld.txt(删除文件) git checkout "learngit" >helloworld.txt（检出），再次提交，使用命令：git commit –a –m “delete helloworld.txt ls可以查询目录

1. 命令rm \*.txt是删除所有.txt文件，ls可以检测目录下的文件，git status可以看到暂存区文件状态，执行git checkout – readme.txt可以恢复readme.txt文件
2. 使用cat readme.txt可以查看readme.txt文件内容
3. 从远程下载是clone，从本地发送给远程push，pull是把远程再推到当前的版本，clone是本地无远程，pull是本地有仓库，将远程与本地的仓库进行合并，fetch是将远程的下载，但是并不和本地的仓库进行合并，而是建立一个分支。
4. 使用git remote可以显示远程地址

Id\_rsa私钥：/c/Users/dvt/.ssh/id\_rsa.

Id\_rsa.pub公钥：/c/Users/dvt/.ssh/id\_rsa.pub.

将远程github的项目检出到你自己的指定的文件夹里，首先进入自己的文件夹，cd E:\mygit\_repo，其次，输入git clone <https://github.com/gaoyuh/learngitrepo.git>，则在指定的文件夹里出现远程的文件。在文件夹里出现后，需要再次加入提交

在github里建立版本库，建立分支，合并分支，将远程master加入到本地指定仓库，修改分支并合并，在本地拉取，在gitbash里提交。

1. 命令diff有2个，分别为diff –staged(比较)和diff –cached
2. 若在本地不小心删除文件，没有提交，想要找回，输入命令git reset HEAD test1.txt

回车后再次输入git checkout test1.txt

1. 使用cat test2.txt可以查看改变
2. 比较git的不同版本
3. 提交git push origin master
4. 分支管理及合并
5. Git里鼓励建立分支
6. SVN与GIT区别：

|  |  |
| --- | --- |
| SVN | Git |
| C/S结构 | 分布式 |
| 客户端只有工作区域 | 客户端有工作区域和本地仓库 |
| SVN不方便离线操作 | GIT可以一直离线操作 |
| SVN速度慢 | Git速度快 |

|  |  |
| --- | --- |
| SVN | Git |
| Checkout | Clone pull fetch |
| Svn branch建立新目录 | No |
| Update | Checkout |
| Delete | Remove |
| Commit | Push |

合并分支：

首先建立分支，在gitbush上加入