**版本控制器**

**简介**

Git---linus[linux基金会]---分布式版本控制

每次更新所有的数据都更新，所有数据在所有的电脑都是一样的。

版本控制软件subversion，git 开源的版本控制系统

在subversion管理下，文件和目录可以超越时空

SVN允许你恢复到早期版本

或者检查数据修改的里历史

许多人将版本控制系统当作i一种神奇的“时间机器”

允许多人协作写作文档崩跟踪所做的修改

访问：

命令行，图形

本地访问，SVN服务器，WEB服务器

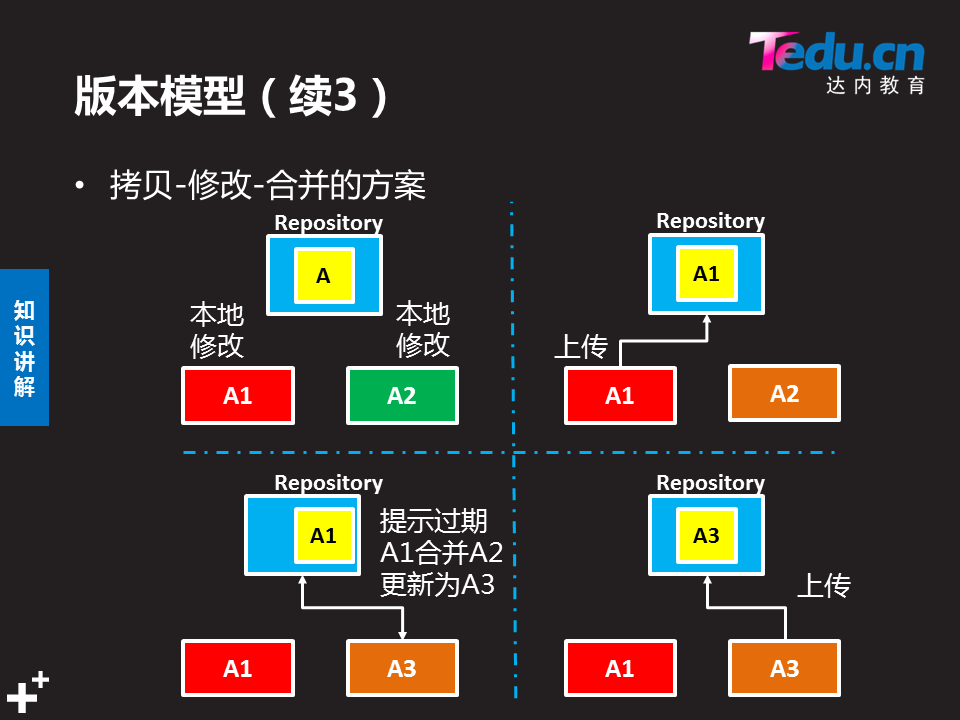
**版本库** C/S 客户端/服务器

版本库是版本控制的核心，

任意数量客户端

客户端通过写数据库分享代码

特点：记录每一次更改



Subversion控制系统使用拷贝-修改-合并模型

每个客户连接版本库，并建立个人工作副本

用户并性工作，修改自己的副本

最终合并版本，解决个别冲突问题，需要认为手动解决

流程：

A拷贝副本，B拷贝副本

A修改c1，B修改c2

A上传c1后，B上传c2，会提示B的文件已过期

B更新文件（合并）后上传新的c3

**搭建subversion**

yum -y install subversion

mkdir /usr/svn

svnadmin create /usr/svn/project //新irdkm建仓库

svn import /usr/lib/systemd/system/ [file:///var/svn/project](file://var\\svn\\project) -m “First “

//提交文档到仓库，-m为注释

**修改配置文件**

创建账户与密码,所有配置文件，要求顶头写，开头不要有空格

vim **/var/svn/project/conf/svnserver.conf**

anon-access = none //19行，匿名无任何权限,anon-->anonymous

auth-access = write //20行，有效账户可写

password-db = passwd //27行，密码文件

authz-db = authz //34行，ACL访问控制列表文件

vim **/var/svn/project/conf/passwd**

[users]

harry = 123456 //用户名和密码

tom = 123456 //用户名和密码

vim **/var/svn/project/conf/authz**

[/]//定义ACL访问控制,根是相对当前仓库而言的,仓库下的文档自动继承根的访问权限

harry = rw                        //用户对项目根路径可读可写

tom = rw

启动服务

svnserve -d -r /var/svn/project // -d 以守护进程方式运行svnserve

-r 指定一个虚拟路径，客户端访问svnserve服务

时以这个路径进行访问

启动服务也可以使用svnserve -d启动，但客户端访问时需要指定绝对路径

(svn://服务器ip/var/svn/project)

ss -ntulp |grep svn //查看服务启动情况，**默认3690端口**

测试：

Web2:

svn --username harry --password 123456 checkout svn://192.168.2.100/ code

vim code/httpd.service

cd code

svn ci -m "xxx" // 上传，只上传修改过的文件httpd.serice ci-->commit

rm -rf code/\*.server

svn update      //将服务器上新的数据同步到本地

svn info     svn://192.168.2.100    //查看版本仓库基本信息

svn log     svn://192.168.2.100    //查看版本仓库的日志

echo "test" > test.sh        //本地新建一个文件

svn ci -m "new file"        //提交失败，该文件不被svn管理

svn add test.sh                //将文件或目录加入版本控制

svn ci -m "new file"            //再次提交，成功

svn mkdir subdir                //创建子目录

svn rm timers.target            //使用svn删除文件

svn ci -m "xxx"                //提交一次代码

vim umount.target            //任意修改本地的一个文件

svn diff                     //查看所有文件的差异

svn diff umount.target        //仅查看某一个文件的差异

svn cat svn://192.168.2.100/reboot.target    //查看服务器文件的内容

svn **-r6** cat svn://192.168.2.100/reboot.target //查看服务器文件第6版本的内容

sed -i 'd' tmp.mount            //删除文件所有内容，但未提交

svn revert tmp.mount         //还原tmp.mount文件

rm -rf \*.target                //任意删除若干文件

svn update                    //还原

sed -i '1a #test###' tuned.service //修改本地副本中的代码文件

svn ci -m "xxx" //提交代码

svn merge -r5:2    tuned.service     //将文件从版本7还原到版本2

**多人协同工作**

Web1:

cd /tmp

svn --username tom --password 123456 co svn://192.168.2.100/ code

cd code

Web2:

cd /tmp

svn --username harry --password 123456 co svn://192.168.2.100/ code

cd code

harry和tom修改不同的文件

[root@web1 code]# sed -i "3a ###harry modify#####" tmp.mount

[root@web1 code]# svn ci -m "has modified"

[root@web2 code]# sed -i "3a ###tom modify#####" umount.target

[root@web2 code]# svn ci -m "has modified"

[root@web2 code]# svn update //将对方修改的内容更新到自己的文件中

[root@web1 code]# svn update

harry和tom修改相同文件的不同行

[root@web1 mycode]# sed -i "3a ###harry modify#####" user.slice

[root@web1 mycode]# svn ci -m "modified"

[root@web2 mycode]# sed -i "6a ###tom modify#####" user.slice

[root@web2 mycode]# svn ci -m "modified"        //提交失败

Sending svnserve

Transmitting file data .svn: Commit failed (details follow):

svn: File '/user.slice' is out of date（过期）

[root@web2 mycode]# svn update                    //提示失败后，先更新再提交即可

[root@web2 mycode]# svn ci -m "modified"        //提交成功

Sending user.slice

Transmitting file data .

**harry和tom修改相同文件的相同行**

[root@web1 mycode]# sed -i '1c [UNIT]' tuned.service

[root@web1 mycode]# svn ci -m "modified"

[root@web2 mycode]# sed -i '1c [unit]' tuned.service

[root@web2 mycode]# svn ci -m "modified"

Sending tuned.service

Transmitting file data .svn: Commit failed (details follow):

svn: File '/tuned.service' is out of date(过期)

[root@web2 mycode]# svn update                    //出现冲突，需要解决

Conflict discovered in 'tuned.service'.

Select: (p) postpone, (df) diff-full, (e) edit,

(mc) mine-conflict, (tc) theirs-conflict,

(s) show all options:p                    //选择先标记p，随后解决

[root@web2 mycode]# ls

tuned.service tuned.service.mine        tuned.service.r10    tuned.service.r9

[root@web2 mycode]# mv tuned.service.mine tuned.service

[root@web2 mycode]# rm -rf tuned.service.r10 tuned.service.r9

[root@web2 mycode]# svn ci -m "modified"    //解决冲突

**备份版本库数据**

[root@web1 ~]# svnadmin dump /var/svn/project > project.bak //备份

[root@web1 ~]# svnadmin create /var/svn/project2 //新建空仓库

[root@web1 ~]# svnadmin load /var/svn/project2 < project.bak     //还原