**步骤一：版本控制，安装Subversion服务器**

**1）YUM安装subversion软件**

[root@web1 ~]# yum -y install subversion

[root@web1 ~]# rpm -q subversion

**2)创建版本库**

[root@web1 ~]# mkdir /var/svn/

[root@web1 ~]# svnadmin create /var/svn/project

**3）本地导入初始化数据（使用SVN内置的认证机制可以有效的增强客户端访问版本库的安全性）**

[root@web1 ~]# cd /usr/lib/systemd/system/

[root@web1 ~]# svn import . file:///var/svn/project/ -m "Init Data"

#”随便命名”

**4）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。**

[root@web1 ~]# vim /var/svn/project/conf/svnserve.conf

[general]

### These options control access to the repository for unauthenticated

### and authenticated users. Valid values are "write", "read",

### and "none". The sample settings below are the defaults.

anon-access = none

//19行，匿名无任何权限

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

[root@web1 ~]# vim /var/svn/project/conf/passwd

… …

[users]

harry = 123456

//用户名和密码

tom = 123456

//用户名和密码

[root@web1 ~]# cat /var/svn/project/conf/authz

[/] //定义ACL访问控制，所有的子目录

harry = rw //用户对项目根路径可读可写

tom = rw

**5）启动服务**

[root@web1 ~]# svnserve -d -r /var/svn/project

[root@web1 ~]# netstat -nutlp |grep svnserve

# -d 是放到后台，-r 后面跟着共享路径，否则访问时后面就得打路径

**步骤二：客户端测试(192.168.2.200)**

**1）将服务器上的代码下载到本地**

[root@web2 ~]# cd /tmp

[root@web2 ~]# svn --username harry --password 123456 \

co svn://192.168.2.100/ code

#建立本地副本,从服务器192.168.2.100上co下载代码到本地code目录

#用户名harry,密码123456

#co 代表下载 ；code 下载路径，自动创建

Store password unencrypted (yes/no)? yes //提示是否保存密码

**2)提交修改数据，必须登陆工作目录**

[root@web2 ~]# cd /tmp/code

[root@web2 code]# ls

[root@web2 code]# vim user.slice //挑选任意文件修改其内容

[root@web2 code]# svn ci -m "modify user" //将本地修改的数据同步到服务器

**3）还原删掉的文件**

[root@web2 code]# svn update //将服务器上新的数据同步到本地

[root@web2 code]# svn info svn://192.168.2.100 //查看版本仓库基本信息

[root@web2 code]# svn log svn://192.168.2.100 //查看版本仓库的日志

**4)创建一个本地文件**

[root@web2 code]# echo "test" > test.sh //本地新建一个文件

[root@web2 code]# svn ci -m "new file" //提交失败，该文件不被svn管理

[root@web2 code]# svn add test.sh //将文件或目录加入版本控制

[root@web2 code]# svn ci -m "new file" //再次提交，成功

5)删除目录，但

[root@web2 code]# svn mkdir subdir //创建子目录

[root@web2 code]# svn rm timers.target //使用svn删除文件

[root@web2 code]# svn ci -m "xxx" //提交一次代码

6)

[root@web2 code]# vim umount.target //任意修改本地的一个文件

[root@web2 code]# svn diff //查看所有文件的差异

[root@web2 code]# svn diff umount.target //仅查看某一个文件的差异

[root@web2 code]# svn cat svn://192.168.2.100/reboot.target //查看服务器文件的内容

[root@web2 code]# sed -i 'd' tmp.mount

//删除文件所有内容，但未提交

[root@web2 code]# svn revert tmp.mount

//还原tmp.mount文件

[root@web2 code]# rm -rf \*.target

//任意删除若干文件

[root@web2 code]# svn update

//还原

[root@web2 code]# sed -i '1a #test###' tuned.service

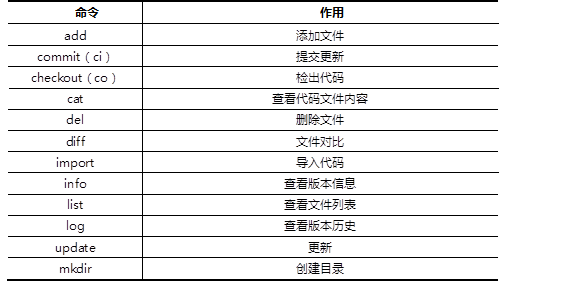
//修改本地副本中的代码文件

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

//提交代码sv

[root@web2 code]# svn merge -r7:2 tuned.service

//将文件从版本7还原到版本2



步骤一：多人协同工作

1）远程连接两个终端，每个人下载代码本地副本，注意web1(192.168.2.100)和web2（192.168.2.200）代表了两个不同的主机，看清楚操作是在哪一台计算机上执行！

[root@web1 ~]# cd /tmp

[root@web1 ~]# svn --username tom --password 123456 \

> co svn://192.168.2.100/ code

[root@web2 ~]# cd /tmp

[root@web2 ~]# svn --username harry --password 123456 \

> co svn://192.168.2.100/ code

[root@web1 ~]# cd code

[root@web2 ~]# cd code

2) harry和tom修改不同的文件

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

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

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

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

[root@web2 mycode]# svn update

[root@web1 mycode]# svn update

3）harry和tom修改相同文件的不同行

[root@srv5 ~]# cd harry

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

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

[root@web2 mycode]# sed -i "6a ###harry 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

4) 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(冲突) 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

步骤二：使用dump指令备份版本库数据

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

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

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

**步骤三：注册使用Github**

1.登陆网站https://github.com，点击Sign up（注册）

[root@pc001 ~]# yum -y install git

[root@pc001 ~]# git clone https://github.com/账户名称/仓库名称

#clone指令用于将服务器仓库中的资料打包下载到本地

[root@pc001 ~]# cd 仓库名称

[root@pc001 ~]# 任意修改文件，或新建文件

[root@pc001 ~]# git add .

#add添加新文件

[root@pc001 ~]# git commit -m "test"

[root@pc001 ~]# git push

#commit和push实现提交代码的功能

[root@pc001 ~]# git pull

#pull更新，类似于svn update

**案例3：软件打包，制作nginx的RPM包**

1）安装rpm-build软件包

[root@web1 ~]# yum -y install rpm-build

2）生成rpmbuild目录结构

[root@web1 ~]# rpmbuild -ba nginx.spec //会报错，没有文件或目录

[root@web1 ~]# ls /root/rpmbuild //自动生成的目录结构

3）准备工作，将源码软件复制到SOURCES目录

[root@web1 ~]# cp nginx-1.12.2.tar.gz /root/rpmbuild/SOURCES/

4）创建并修改SPEC配置文件

[root@web1 ~]# vim /root/rpmbuild/SPECS/nginx.spec 制作nginx的

Name:nginx #tar包版本数字前面的名字

Version:1.12.2

Release: 10

Summary: Nginx is a web server software.

Source0:nginx-1.12.2.tar.gz

./configure --prefix=/usr/local/nginx

/usr/local/nginx/\* //对哪些文件与目录打包

步骤二：使用配置文件创建RPM包

1）安装依赖软件包

[root@web1 ~]# yum -y install gcc pcre-devel zlib-devel openssl-devel

2）rpmbuild创建RPM软件包

[root@web1 ~]# rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

[root@web1 ~]# ls /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

[root@web1 ~]# rpm -qpi RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

步骤三：安装、卸载软件

[root@web1 ~]# rpm -ivh RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

[root@web1 ~]# rpm -qa |grep nginx

[root@web1 ~]# /usr/local/nginx/sbin/nginx

[root@web1 ~]# curl http://127.0.0.1/

**总结：**

**Subversion（svn） 允许数据恢复到早期版本，允许多人一起修改文件，并记录每一次的改变**

1.安装Subversion（后端服务器192.168.2.100）

yum -y install subversion

rpm -q subversion #查看

1. 创建版本库

mkdir /var/svn

svnadmin create /var/svn/project

1. 导入项目代码

cd /usr/lib/systemd/system

svn import . [file:///var/svn/project](file://var\\svn\\project) -m “xx” # “.” 是导入”project”当前路径

4.认证与授权

1） vim /var/svn/project/conf/svnserve.conf

anon-access = none #匿名无任何权限 19

auth-access = write #有效账户可以写 20

password-db = passwd #密码文件 27

authz-db = authz #ACL访问控制列表文件 34

1. vim /var/svn/project/conf/passwd

harry = 123456 #用户名和密码

tom = 123456

1. vim /var/svn/project/conf/authz

[/] #定义ACL访问控制

harry = rw #用户对项目根路径可读可写

tom = rw

1. 启动服务

svnserve -d -r /var/svn/project

ss -anputl | grep svnserve #查看端口是否启用

1. 客户端测试(192.168.2.200)安装subversion

1)将服务器上的代码下载到本地

cd /tmp

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

#建立本地副本，从服务器上下载代码到本地目录code，后面提示是否保存密码yes

1. 对文件内容的修改及操作

cd /tmp/code

vim user.slice # 修改任意文件修改内容

svn ci -m “xxx” # 将本地修改的数据同步到服务器

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,谁修改了哪些文件

echo “test” > test.sh #本地新建一个文件

svn add test.sh #将文件或目录加入版本控制

svn mkdir aa #创建目录

svn ci -m “xx” #再次提交给服务器

svn rm test.sh #删除文件

svn diff #查看所有文件的差异

svn diff 文件名 #查看某一个文件的差异

svn cat svn://19.168.2.100/test.sh #查看服务器文件的内容

sed -i ‘d’ test.sh #删除文件所有内容，但未提交

svn revert test.sh #还原test.sh 文件

rm -rf \*.target #删除任意文件

svn update #还原

svn merge -r7:2 文件名 #将某文件从版本7还原到版本2

**使用Subversion协同工作**

1. web1和web2都下载代码本地副本

cd /tmp

web1: svn --username tom --password 123456 co svn://192.168.2.100/ code

web2: svn --username harry --password 123456 co svn://19.168.2.100/ code

cd code

1. harry 和tom修改不同文件
2. 修改相同文件，后提交svn ci -m “dd” 的会失败，先更新，再提交

svn update #更新到最新版本

svn ci -m “xx”

1. harry 和tom修改相同文件相同行 test.sh，后提交会冲突，需要人当面交流解决

可以先标p ,随后解决

ls test.\* #会出现四个文件，一个自己修改后的，一个对方修改的还有当前版本

和上版本

#解决问题后就就把多与的删除掉，只剩正确的，文件名修改为原文件一样

mv test.sh.mine test.sh

rm -rf test.sh.r2 test.sh.r4

svn ci -m “xx”

**使用dump备份版本库数据**

svnadmin dump /var/svn/project > project.bak #备份

svnadmin create /var/svn/project2 #新建空仓库

svnadmin load /var/svn/project2 < project.bak #还原

**二．注册使用github**

网址：<https://github.com>

账户：githuangzhuang

将网站上的资料下载到本地

1. 安装git

yum -y install git

git clone <https://github.com/账户名/仓库名称>

cd 仓库名称

git add #添加新文件

**软件打包，制作nginx的RPM包**

1. 准备源码软件 2.安装rpm-build 3.编写编译配置文件 4.编译RPM包

yum -y install rpm-buile

rpmbuild -ba test.spec #报错是正常的，生成rpmbuild目录

ls rpmbuild #生成六个目录

1. 将源码包复制到rpmbuild子目录

cp nginx-1.12.2.tar.gz rpmbuild/SOURCES

1. 新建SPEC文件

vim /root/rpmbuild/SPECS/nginx.spec #后缀必须是.spec

name:nginx

version:1.12.2

release: 2 #版本几，随便写

summary: #对软件的描述可以是中文，随便写

Group: #软件组，可以删掉

license: #协议，随便写

url： #网址，随便写

source:nginx-1.12.2.tar.gz #必须写对

./configure --prefix=/usr/local/nginx

/usr/local/nginx/\* #对哪些文件与目录打包

步骤二：使用配置文件创建RPM包

1）安装依赖软件包

yum -y install gcc pcre-devel zlib-devel openssl-devel

2）rpmbuild创建RPM软件包

rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

rpm -qpi RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

步骤三：安装、卸载软件

rpm -ivh RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm