# linux 配置SVN小结

转载自：https://blog.csdn.net/championsuiyang/article/details/76147605

1 : 通过yum命令安装svnserve，命令如下：

yum -y install subversion

2 : 创建版本库目录（此时仅为目录，为后面创建版本库提供存放位置）

在根目录下随便找一个目录 这里我在 dev 新建一个目录 svn

再在SVN里新建一个ku(库的拼音)

3 ：创建svn版本库 ， 意味着 "ku" 这个普通目录 成为了SVN的库

svnadmin create /dev/svn/ku

可以进入 ku　里查看

ｋｕ这个目录里多了这些东西就对了

４　：配置修改

进入conf

conf目录下，一共存放三份重要的配置文件

authz：负责账号权限的管理，控制账号是否读写权限

passwd：负责账号和密码的用户名单管理

svnserve.conf：svn服务器配置文件

进入 passwd 在最后加上 用户名 = 密码 如下 用户名 mjj 密码 qqqqqq

保存退出 然后在 authz 里 最后一条 \* = r 的上面加一条

[/]

并 \* = r 改为 用户名 = rw

那么这个用户名就有了r 读 w 写的权限

保存退出 然后在 svnserve.conf 里

注释全部去掉

anon-access = none 这里要改为 none

auth-access = write

password-db = passwd

authz-db = authz

realm = /dev/svn/ku 这里要改为自己的目录

保存退出 然后 启动服务器

svnserve -d -r /dev/svn/ 通过svn:// ip/ ku/ 访问ku

svnserve -d -r /dev/svn/ku 通过 svn://ip/ 访问ku

# svn同步备份的方式有三种

转载自：https://www.cnblogs.com/xiaxiaosheng/p/6021616.html

1svnadmin dump

2)svnadmin hotcopy

3)svnsync.

优缺点分析

==============

第一种svnadmin dump是官方推荐的备份方式，优点是比较灵活，可以全量备份也可以增量备份，并提供了版本恢复机制。

缺点是：如果版本比较大，如版本数增长到数万、数十万，那么dump的过程将非常慢；备份耗时，恢复更耗时；不利于快速进行灾难恢复。

个人建议在版本数比较小的情况下使用这种备份方式。

第二种svnadmin hotcopy原设计目的估计不是用来备份的，只能进行全量拷贝，不能进行增量备份；

优点是：备份过程较快，灾难恢复也很快；如果备份机上已经搭建了svn服务，甚至不需要恢复，只需要进行简单配置即可切换到备份库上工作。

缺点是：比较耗费硬盘，需要有较大的硬盘支持（俺的备份机有1TB空间，呵呵）。

第三种svnsync实际上是制作2个镜像库，当一个坏了的时候，可以迅速切换到另一个。不过，必须svn1.4版本以上才支持这个功能。

优点是：当制作成2个镜像库的时候起到双机实时备份的作用；

以上不是自己观点，下面真实记录我的svn sync实验过程

两台机器

server1: 192.168.1.224

server2: 192.168.1.225

都是centos6.5环境

首先在sever1上搭建好了一个svn，然后模拟提交了一些东西

然后在sever2上搭建了一个一模一样的svn，保持空的

现在的目的是将server1同步备份到server2

在server1上直接运行:

svnsync init svn://192.168.1.225/ svn://192.168.1.224/（svnserve -d -r /dev/svn/ku 通过 svn://ip/ 访问ku）

svnsync init svn://192.168.1.225/ku svn://192.168.1.224/ku（以svnserve -d -r /dev/svn/ 通过svn:// ip/ ku/ 访问ku）

即svnsync init 目标svn链接 源svn链接，执行同步之前的初始化

这一步失败了，报如下错误：

svnsync: Repository has not been enabled to accept revision propchanges;

ask the administrator to create a pre-revprop-change hook

提示需要在hooks下面创建一个pre-revprop-change hook

简单解释下，hook类似于操作系统的勾子，svn会在收到一些操作请求的时候执行hooks目录下的对应的脚本，例如想要commit的时候做一些事情就可以在对应的脚本下面添加你要执行的命令，下一次在commit 的时候就会触发执行

一开始没明白，不知道应该在源机器上创建还是在目标机器上创建，其实是在目标机器上创建的

然后在目标机器上copy了一份pre-revprop-change.tmpl成pre-revprop-change

再次执行初始化命令

依然报错，这次的错误不一样了

svnsync: Revprop change blocked by pre-revprop-change hook (exit code 255) with no output.

大概就是同步的过程中刚刚创建的hook调用没有成功，然后尝试给pre-revprop-change添加可执行权限，依然失败

依然失败，依然失败，在stack overflow上看到有人提出的解决方法是把pre-revprop-change改成下面这个样子：

#!/bin/sh

exit 0

再次初始化，终于成功，提示先让你输入用户名密码，最后一步的输出如下：

后来经过验证，确实需要可执行权限，改了之后还是失败的原因是hooks执行的结果是失败的，确实执行了

这一步成功之后，下一步同步就直接成功了：

svnsync sync svn://192.168.1.225/

执行这个命令会把没有同步的版本都同步过去

然后为了让server1每次有更新之后都自动同步到server2，可以在server1的commit的hooks最后加上执行一下同步的命令：

svnsync sync svn://192.168.1.225/

这样就完美实现了实时备份，而且在server1出现问题的时候随时都可以直接切换到server2哟

最后有一点要注意的是，我尝试备份到一个不为空的svn，也就是目标svn已经存在要备份的repos的时候，是会失败的，因此，只能备份到一个空的svn

最后还有一个非常重要的要注意的问题就是，如果在直接在目标在做了修改的话，那么后面就没有办法同步了，都会失败，所以，禁止在备份svn上直接做任何操作，这里的花建议专门写一个脚本，在post-commit勾子里面调用，通过脚本来同步，然后判断一下是否同步成功，如果同步失败了需要及时处理，比如可以给相关人员发邮件通知及时处理，防止同步失败了导致server1一旦宕机之后突然发现server2早就没有同步了就晚了。