# 前言

1、 rsync

如果你man一下sync的话，就会发现：sync-flush file system buffers，它是一个把缓冲区中的数据同步到文件系统中的一个命令；而rsync其实就是remote rsync，它是一个远程同步工具，兼具cp和scp的功能，rsync命令的使用几乎和scp是一样一样的。rsync是通过超级守护进程xinetd进行触发同步的。

1）优点：与cp和scp相比来说，rsync更快，更安全，支持增量备份。rsync在数据同步的过程中，不像cp一样全部都拷贝，而是先去比对特征码，只有不一样的才会去拷贝，如果一样了，就不需再做多余操作。通过使用rsync+crontab可以解决对实时性要求不是太高的场景。

2）缺点：在这个大数据年代，如果你的数据量非常大，你每做一次任务计划，rsync都会先去遍历目标目录，把所有数据做一次特征码比对，然后进行差量传输，这个过程会是很漫长的，对于那些要求实时性更新比较高的企业来说，无疑是一场噩梦。但是如果能够出现一个工具能够实时的去监控我们的文件系统，只有在数据改变时才会触发它去同步，那该有多好啊！为了满足广大需求者的心声rsync+inotify组合就出现了

2、inotify

Inotify 是一种强大的、细粒度的、异步的文件系统事件监控机制，linux内核从2.6.13起，加入了Inotify支持，通过Inotify可以监控文件系统中添加、删除，修改、移动等各种细微事件，利用这个内核接口，第三方软件就可以监控文件系统下文件的各种变化情况，而inotify-tools就是这样的一个第三方软件。

3、rsync+inotify 之推荐理由

1）服务器性能：rsync+crontab会定时去检查是否有文件更新，这势必会造成服务器性能下降；而rsync+inotify组合是触发式更新，只有在数据文件有变化时，才会去更新，因此相对前者而言，是提高了服务器性能

2）数据实时性：rsync+crontab是周期性任务计划，不能保证数据的实时性；rsync+inotify组合是触发式更新，只要有数据变化，就立刻同步更新

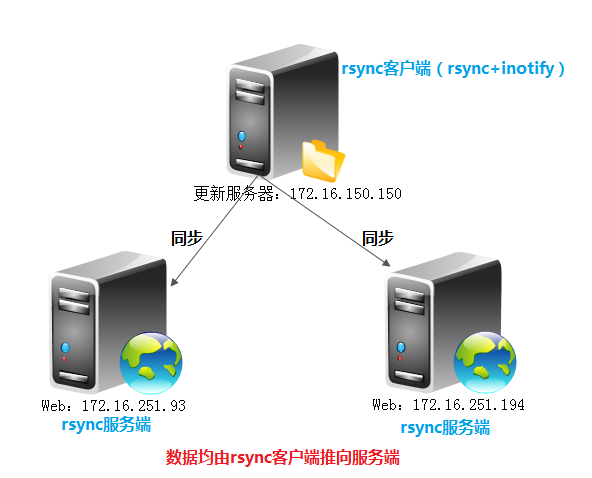
# 二、环境

1、操作系统：CentOS6.4-X86\_64

2、rsync客户端（rsync+inotify）：192.168.254.11

3、rsync服务端：192.168.254.9，192.168.254.10

# 三、拓扑



客户端：需要把文件同步到另一台服务器的。Systemctl start rsyncd启动

服务端：把别人文件同步过来的。rsync –daemon启动

# 四、配置rsync服务端

## 1.rsync服务器配置192.168.254.9

1）检查系统是否已经安装rsync和xinetd超级守护进程，CentOS6.4默认已经安装rsync，但未安装xinetd





2）为rsync服务提供配置文件

vim /etc/rsyncd.conf

#Global Settings 全局配置

port = 10001 #运行端口号

02.uid = web                          #运行rsync的用户

03.gid = web                          #运行rsync的用户组

04.use chroot = no                      #是否让进程离开工作目录

05.max connections = 5                  #最大并发连接数，0为不限制

06.timeout = 600                        #超时时间

07.pid file = /var/run/rsyncd.pid       #指定rsync的pid存放路径

08.lockfile = /var/run/rsyncd.lock      #指定rsync的锁文件存放路径

09.log file = /var/log/rsyncd.log       #指定rsync的日志存放路径

10.#模块配置

11.[web1] #认证的模块名,在client端需要指定

12.path = /var/www/data                 #需要同步的目录

13.ignore errors = yes                  #忽略一些无关的I/O错误

14.read only = no                       #如果为只读，那么将不能进行写同步，所以必须关闭

15.write only = no                      #客户端是否有写权限（PUSH）

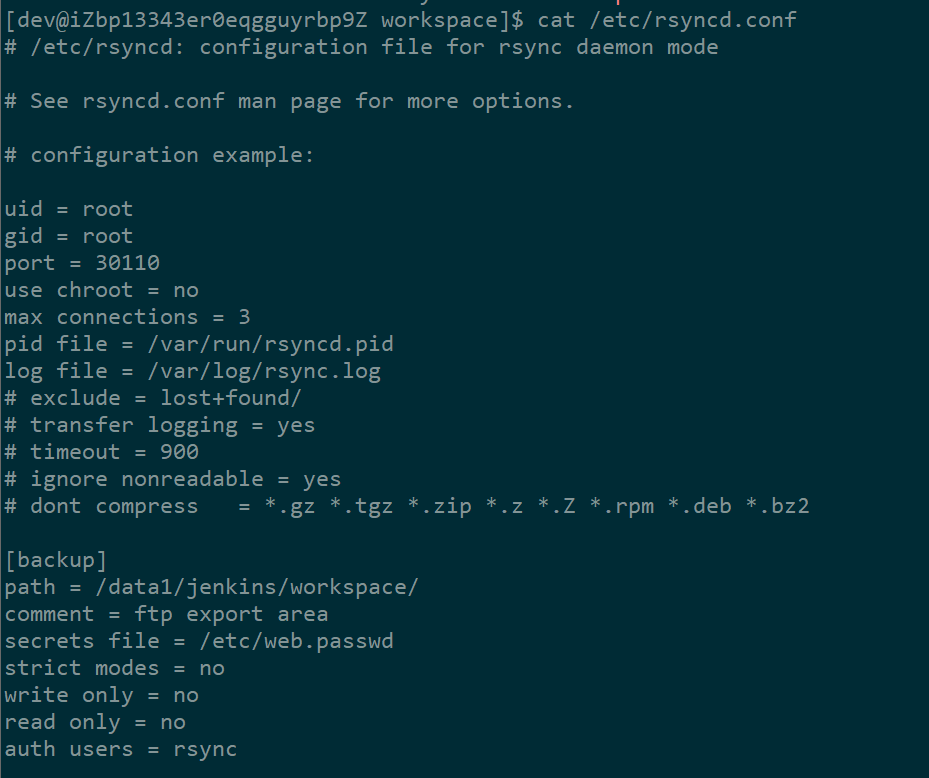
16.hosts allow = 172.16.150.150         #白名单，可以访问此模块的主机

17.hosts deny = \*                       #黑名单，\*表示任何主机

18.list = yes                           #客户端请求是否可以列出模块列表

19.auth users = web                     #登录用户名，不需要系统用户中有，虚拟的就行

20.secrets file = /etc/web.passwd       #指定存放“用户名：密码”格式的文件



3）创建同步目录并更改目录属主

[root@localhost ~]# mkdir /var/www/data

[root@localhost ~]# chown web.web /var/www/data

4）配置rsync认证文件/etc/web.passwd



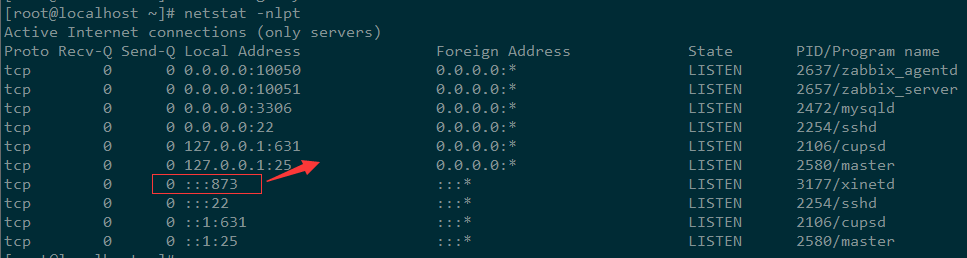
5)修改/etc/web.passwd的权限为600



6）配置服务能够启动

rsync --daemon

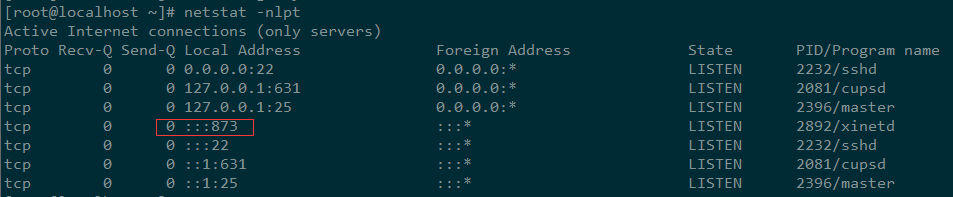
7）检查873端口是否成功监听



## 2.rsync服务器配置192.168.254.10

步骤同上

最后查看873端口是否被监听，确保已安装配置完成



# 五、配置rsync客户端（rsync+inotify）

配置rsync客户端：192.168.254.11

1. 设置 rsync 客户端的密码文件，客户端只需要设置 rsync 同步的密码即可，不用设置用户名（默认已经安装了rsync）



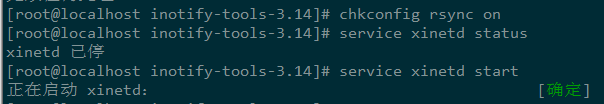
1. 修改/etc/rsyncd.passwd的权限为600



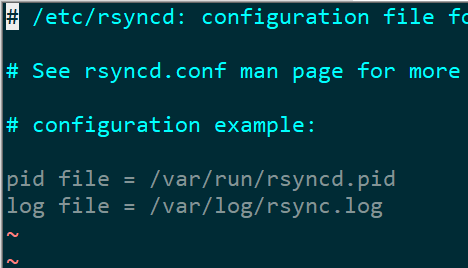
1. 安装rsync客户端软件

yum install  xinetd

1. 启动rsync



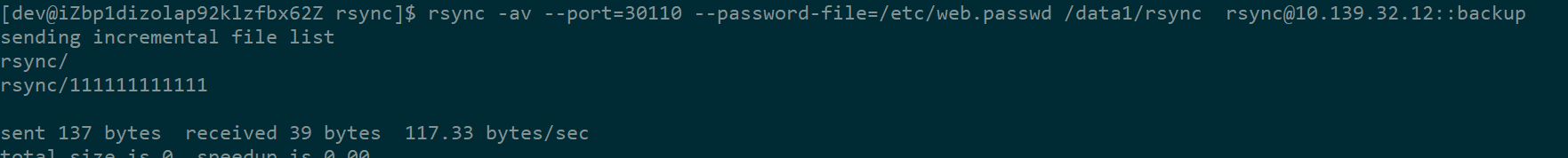
1. 客户端配置文件不需要指定其他东西，只需要写上日志路径 pid路径即可



此时已可以手动进行文件同步了

rsync -av --port=30110 --password-file=/etc/web.passwd /data1/rsync [rsync@10.139.32.12::backup](mailto:rsync@10.139.32.12::backup)

把客户端/data1/rsync 目录文件同步到服务端，已rsync账号认证，端口30110，同步路径在backup里配置



如需要自动同步需按下面要求安装inotify

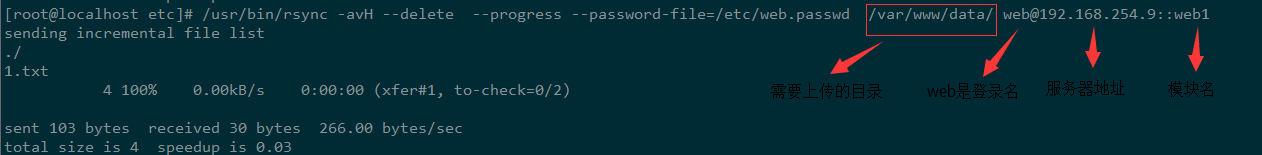
1. 安装inotify-tools
2. # tar xf inotify-tools-3.13.tar.gz
3. # cd inotify-tools-3.13
4. # ./configure --prefix=/usr/local/inotify
5. # make && make install
6. 在192.168.254.11上测试下上传文件到192.168.254.9服务端

/usr/bin/rsync -avH --delete --progress --password-file=/etc/web.passwd /var/www/data/ [web@192.168.254.9::web1](mailto:web@192.168.254.9::web1)

其中登录名 模块名是在服务端rsyncd.conf配置好的

/var/www/data/ 是需要上传的目录

[web@192.168.254.9::web1](mailto:web@192.168.254.9::web1) web是用户名 192.168.254.9是上传服务器地址，web1是模块名



# 六、配置脚本，让文件自动同步

#!/bin/sh

srcdir=/var/www/data/

rsyncuser=web

password=/etc/web.passwd

host="192.168.254.9 192.168.254.10"

for ip in $host

do

rsync -avH --progress --delete --password-file=${password} ${srcdir} ${rsyncuser}@$ip::web1

done

/usr/local/inotify/bin/inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f%e' -e close\_write,modify,delete,create,attrib,move $srcdir | while read file

do

for ip in $host

do

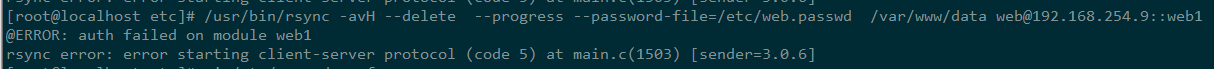
rsync -avH --progress --delete --password-file=${password} ${srcdir} ${rsyncuser}@$ip::web1

echo " ${file} was rsynced" >> /var/log/rsync.log 2>&1

done

done

# 七、排错

1. 如果出现@ERROR: auth failed on module web1错误，可能是rsyncd.conf编码问题，把里面汉字删掉
2. @ERROR: access denied to web1 from unknown (192.168.254.11)

rsync error: error starting client-server protocol (code 5) at main.c(1503) [sender=3.0.6]，

原因：是配置选项host allow的问题，初学者喜欢一个允许段做成一个配置，然后模块又是同一个，致使导致。

# 八、命令

### rsync参数的具体解释如下：

-v, --verbose 详细模式输出  
-q, --quiet 精简输出模式  
-c, --checksum 打开校验开关，强制对文件传输进行校验  
-a, --archive 归档模式，表示以递归方式传输文件，并保持所有文件属性，等于-rlptgoD  
-r, --recursive 对子目录以递归模式处理  
-R, --relative 使用相对路径信息  
-b, --backup 创建备份，也就是对于目的已经存在有同样的文件名时，将老的文件重新命名为~filename。可以使用--suffix选项来指定不同的备份文件前缀。  
--backup-dir 将备份文件(如~filename)存放在在目录下。  
-suffix=SUFFIX 定义备份文件前缀  
-u, --update 仅仅进行更新，也就是跳过所有已经存在于DST，并且文件时间晚于要备份的文件。(不覆盖更新的文件)  
-l, --links 保留软链结  
-L, --copy-links 想对待常规文件一样处理软链结  
--copy-unsafe-links 仅仅拷贝指向SRC路径目录树以外的链结  
--safe-links 忽略指向SRC路径目录树以外的链结  
-H, --hard-links 保留硬链结  
-p, --perms 保持文件权限  
-o, --owner 保持文件属主信息  
-g, --group 保持文件属组信息  
-D, --devices 保持设备文件信息  
-t, --times 保持文件时间信息  
-S, --sparse 对稀疏文件进行特殊处理以节省DST的空间  
-n, --dry-run现实哪些文件将被传输  
-W, --whole-file 拷贝文件，不进行增量检测  
-x, --one-file-system 不要跨越文件系统边界  
-B, --block-size=SIZE 检验算法使用的块尺寸，默认是700字节  
-e, --rsh=COMMAND 指定使用rsh、ssh方式进行数据同步  
--rsync-path=PATH 指定远程服务器上的rsync命令所在路径信息  
-C, --cvs-exclude 使用和CVS一样的方法自动忽略文件，用来排除那些不希望传输的文件  
--existing 仅仅更新那些已经存在于DST的文件，而不备份那些新创建的文件  
--delete 删除那些DST中SRC没有的文件 //如果加上这一参数，删除rsync客户端监控的目录中的文件，则对应的也会在服务端进行删除；不加的话，服务端只会备份文件，不会因为客户端删除文件而删除  
--delete-excluded 同样删除接收端那些被该选项指定排除的文件  
--delete-after 传输结束以后再删除  
--ignore-errors 及时出现IO错误也进行删除  
--max-delete=NUM 最多删除NUM个文件  
--partial 保留那些因故没有完全传输的文件，以是加快随后的再次传输  
--force 强制删除目录，即使不为空  
--numeric-ids 不将数字的用户和组ID匹配为用户名和组名  
--timeout=TIME IP超时时间，单位为秒  
-I, --ignore-times 不跳过那些有同样的时间和长度的文件  
--size-only 当决定是否要备份文件时，仅仅察看文件大小而不考虑文件时间  
--modify-window=NUM 决定文件是否时间相同时使用的时间戳窗口，默认为0  
-T --temp-dir=DIR 在DIR中创建临时文件  
--compare-dest=DIR 同样比较DIR中的文件来决定是否需要备份  
-P 等同于 --partial  
--progress 显示备份过程  
-z, --compress 对备份的文件在传输时进行压缩处理  
--exclude=PATTERN 指定排除不需要传输的文件模式  
--include=PATTERN 指定不排除而需要传输的文件模式  
--exclude-from=FILE 排除FILE中指定模式的文件  
--include-from=FILE 不排除FILE指定模式匹配的文件  
--version 打印版本信息  
--address 绑定到特定的地址  
--config=FILE 指定其他的配置文件，不使用默认的rsyncd.conf文件  
--port=PORT 指定其他的rsync服务端口  
--blocking-io 对远程shell使用阻塞IO  
-stats 给出某些文件的传输状态  
--progress 在传输时实现传输过程  
--log-format=formAT 指定日志文件格式  
--password-file=FILE 从FILE中得到密码  
--bwlimit=KBPS 限制I/O带宽，KBytes per second  
-h, --help 显示帮助信息

一般都使用azv选项（如果windows与linux同步不建议使用这个参数）

### innotify参数说明

#### inotifywait

语法：  
inotifywait [-hcmrq] [-e ] [-t ] [--format ] [--timefmt ] [ ... ]  
参数：  
**-h,–help**  
输出帮助信息  
**@**  
排除不需要监视的文件，可以是相对路径，也可以是绝对路径。  
**–fromfile**  
从文件读取需要监视的文件或排除的文件，一个文件一行，排除的文件以@开头。  
**-m, –monitor**  
接收到一个事情而不退出，无限期地执行。默认的行为是接收到一个事情后立即退出。  
**-d, –daemon**  
跟–monitor一样，除了是在后台运行，需要指定–outfile把事情输出到一个文件。也意味着使用了–syslog。  
**-o, –outfile**  
输出事情到一个文件而不是标准输出。  
**-s, –syslog**  
输出错误信息到系统日志  
**-r, –recursive**  
监视一个目录下的所有子目录。  
**-q, –quiet**  
指定一次，不会输出详细信息，指定二次，除了致命错误，不会输出任何信息。  
**–exclude**  
正则匹配需要排除的文件，大小写敏感。  
**–excludei**  
正则匹配需要排除的文件，忽略大小写。  
**-t , –timeout**  
设置超时时间，如果为0，则无限期地执行下去。  
**-e , –event**  
指定监视的事件。  
**-c, –csv**  
输出csv格式。  
**–timefmt**  
指定时间格式，用于–format选项中的%T格式。  
**–format**  
指定输出格式。  
%w 表示发生事件的目录  
%f 表示发生事件的文件  
%e 表示发生的事件  
%Xe 事件以“X”分隔  
%T 使用由–timefmt定义的时间格式

#### inotifywatch

语法：  
inotifywatch [-hvzrqf] [-e ] [-t ] [-a ] [-d ] [ ... ]  
参数：  
**-h, –help**  
输出帮助信息  
**-v, –verbose**  
输出详细信息  
**@**  
排除不需要监视的文件，可以是相对路径，也可以是绝对路径。  
**–fromfile**  
从文件读取需要监视的文件或排除的文件，一个文件一行，排除的文件以@开头。  
**-z, –zero**  
输出表格的行和列，即使元素为空  
**–exclude**  
正则匹配需要排除的文件，大小写敏感。  
**–excludei**  
正则匹配需要排除的文件，忽略大小写。  
**-r, –recursive**  
监视一个目录下的所有子目录。  
**-t , –timeout**  
设置超时时间  
**-e , –event**  
只监听指定的事件。  
**-a , –ascending**  
以指定事件升序排列。  
**-d , –descending**  
以指定事件降序排列。

#### 可监听事件

|  |  |
| --- | --- |
| access | 文件读取 |
| modify | 文件更改。 |
| attrib | 文件属性更改，如权限，时间戳等。 |
| close\_write | 以可写模式打开的文件被关闭，不代表此文件一定已经写入数据。 |
| close\_nowrite | 以只读模式打开的文件被关闭。 |
| close | 文件被关闭，不管它是如何打开的。 |
| open | 文件打开。 |
| moved\_to | 一个文件或目录移动到监听的目录，即使是在同一目录内移动，此事件也触发。 |
| moved\_from | 一个文件或目录移出监听的目录，即使是在同一目录内移动，此事件也触发。 |
| move | 包括moved\_to和 moved\_from |
| move\_self | 文件或目录被移除，之后不再监听此文件或目录。 |
| create | 文件或目录创建 |
| delete | 文件或目录删除 |
| delete\_self | 文件或目录移除，之后不再监听此文件或目录 |
| unmount | 文件系统取消挂载，之后不再监听此文件系统。 |