# 多机部署之使用rsync+inotify实时双向同步代码

## 参考路径

https://blog.csdn.net/qq\_39014761/article/details/85159674

https://www.cnblogs.com/jackyyou/p/5681126.html

## Rsync和inotify

**rsync可以实现触发式的文件同步，但是通过crontab守护进程方式进行触发，同步的数据和实际数据会有差异，而inotify可以监控文件系统的各种变化，当文件有任何变动时，就触发rsync同步，这样刚好解决了同步数据的实时性问题。**

## 安装rsync

### 1.安装

yum install -y rsync

### 2.创建配置文件vi /etc/rsyncd.conf

#rsync通用配置文件,配置的注释不要写在配置后面，否则会有问题

uid = root

gid = root

use chroot = 0

port = 873

#允许ip访问设置，请根据实际需要进行配置,这里为了方便设为全网段 \*,生产环境下为了安全请指定ip或ip段

# hosts allow = 192.168.0.1/255.255.255.0 198.162.145.1 10.0.1.0/255.255.255.0

hosts allow = \*

max connections = 0

timeout = 300

pid file = /var/run/rsyncd.pid

lock file = /var/run/rsyncd.lock

log file = /var/log/rsyncd.log

log format = %t %a %m %f %b

transfer logging = yes

syslog facility = local3

#方括号中为模块声明，对应命名,这里master\_web对应了主web机配置,从服务器可都为[slave\_web]，方便inotify脚本配置

[master\_web]

#指定当前模块在rsync服务器上的同步路径，该参数是必须指定的

path = /home/wwwroot/

#注释，可以同模块名一样,从服务器可都为slave\_web

comment = master\_web

ignore errors

#是否允许客户端上传文件

read only = no

list = no

#指定由空格或逗号分隔的用户名列表，只有这些用户才允许连接该模块

auth users = rsync

#保存密码和用户名文件，需要自己生成

secrets file = /etc/rsyncd.passwd

### 创建或者修改用户名密码

vi /etc/rsyncd.passwd和vi /root/rsyncd.passwd

#/etc/rsyncd.passwd中内容格式 用户名:密码 ，多个用户名密码则每添加多行，建议从web机都是一样的用户名和密码

rsync:123456

#/root/rsyncd.passwd中内容只需要填写从服务器的密码，例如这里从服务器配的用户名密码都是rsync:123456，这里主服务器则写123456一个就可以了

123456

### 4.给密码文件赋权限

chmod 600 /etc/rsyncd.passwd

chmod 600 /root/rsyncd.passwd

### 5.以守护进程方式启动rsync服务

/usr/bin/rsync --daemon --config=/etc/rsyncd.conf

停止服务：rm -rf /var/run/rsyncd.pid;

### 6.添加开机自启动

echo "/usr/bin/rsync --daemon --config=/etc/rsyncd.conf">>/etc/rc.local

### 7.验证代码是否同步

# 主web服务器推送代码到从web服务器，命令中的

# rsync@192.168.10.230::slave\_web# 192.168.10.230 为从web服务器的ip,

# slave\_web 为从web服务器的 rsyncd.conf 中配置的模块名，

# rsync 为为从web服务器的中 rsyncd.passwd 中配置的用户名

rsync -vzrtopg --progress /home/publicWebapps/ROOT/ rsync@192.168.10.107::master\_web --password-file=/root/rsyncd.passwd

rsync -vzrtopg --progress /home/publicWebapps/ROOT/ rsync@192.168.10.72::master\_web --password-file=/root/rsyncd.passwd

rsync -vzrtopg --progress /home/publicWebapps/ROOT/ rsync@192.168.10.70::master\_web --password-file=/root/rsyncd.passwd

### 同步时报没有权限

1. 备份的文件是否有权限（755）
2. 判断系统是否为Enforcing，是修改为Permissive

[root@test01 etc]# getenforce

Enforcing

果然是enforce模式，然后

[root@test01 etc]# setenforce 0

关闭enforce模式

[root@test01 etc]# getenforce

Permissive

## 安装inotify

### 安装

wget http://github.com/downloads/rvoicilas/inotify-tools/inotify-tools-3.14.tar.gz --no-check-certificate

tar -zxvf inotify-tools-3.14.tar.gz

makedir /usr/local/inotify

cd inotify-tools-3.14.tar.gz

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

make && make install

### 查看是否安装成功

ls -alh /usr/local/inotify/bin/inotify\*

### 配置inotify\_back.sh同步监控脚本

vi /usr/local/inotify/inotify\_back.sh

脚本内容：

#!/bin/bash

src=/data/ # 需要同步的源路径

des=data # 目标服务器上 rsync --daemon 发布的名称，rsync --daemon 上面rsyncd.conf中配置的comment的值

rsync\_passwd\_file=/root/rsyncd.passwd # rsync验证的密码文件

ip1=192.168.0.18 # 目标服务器1

ip2=192.168.0.19 # 目标服务器2

user=root # rsync --daemon定义的验证用户名

cd ${src} # 此方法中，由于rsync同步的特性，这里必须要先cd到源目录，inotify再监听 ./ 才能rsync同步后目录结构一致，有兴趣的同学可以进行各种尝试观看其效果

/usr/local/bin/inotifywait -mrq --format '%Xe %w%f' -e modify,create,delete,attrib,close\_write,move ./ | while read file # 把监控到有发生更改的"文件路径列表"循环

do

INO\_EVENT=$(echo $file | awk '{print $1}') # 把inotify输出切割 把事件类型部分赋值给INO\_EVENT

INO\_FILE=$(echo $file | awk '{print $2}') # 把inotify输出切割 把文件路径部分赋值给INO\_FILE

echo "-------------------------------$(date)------------------------------------"

echo $file

#增加、修改、写入完成、移动进事件

#增、改放在同一个判断，因为他们都肯定是针对文件的操作，即使是新建目录，要同步的也只是一个空目录，不会影响速度。

if [[ $INO\_EVENT =~ 'CREATE' ]] || [[ $INO\_EVENT =~ 'MODIFY' ]] || [[ $INO\_EVENT =~ 'CLOSE\_WRITE' ]] || [[ $INO\_EVENT =~ 'MOVED\_TO' ]] # 判断事件类型

then

echo 'CREATE or MODIFY or CLOSE\_WRITE or MOVED\_TO'

rsync -avzcR --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip1}::${des} && # INO\_FILE变量代表路径哦 -c校验文件内容

rsync -avzcR --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip2}::${des}

#仔细看 上面的rsync同步命令 源是用了$(dirname ${INO\_FILE})变量 即每次只针对性的同步发生改变的文件的目录(只同步目标文件的方法在生产环境的某些极端环境下会漏文件 现在可以在不漏文件下也有不错的速度 做到平衡) 然后用-R参数把源的目录结构递归到目标后面 保证目录结构一致性

fi

#删除、移动出事件

if [[ $INO\_EVENT =~ 'DELETE' ]] || [[ $INO\_EVENT =~ 'MOVED\_FROM' ]]

then

echo 'DELETE or MOVED\_FROM'

rsync -avzR --delete --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip1}::${des} &&

rsync -avzR --delete --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip2}::${des}

#看rsync命令 如果直接同步已删除的路径${INO\_FILE}会报no such or directory错误 所以这里同步的源是被删文件或目录的上一级路径，并加上--delete来删除目标上有而源中没有的文件，这里不能做到指定文件删除，如果删除的路径越靠近根，则同步的目录月多，同步删除的操作就越花时间。这里有更好方法的同学，欢迎交流。

fi

#修改属性事件 指 touch chgrp chmod chown等操作

if [[ $INO\_EVENT =~ 'ATTRIB' ]]

then

echo 'ATTRIB'

if [ ! -d "$INO\_FILE" ] # 如果修改属性的是目录 则不同步，因为同步目录会发生递归扫描，等此目录下的文件发生同步时，rsync会顺带更新此目录。

then

rsync -avzcR --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip1}::${des} &&

rsync -avzcR --password-file=${rsync\_passwd\_file} $(dirname ${INO\_FILE}) ${user}@${ip2}::${des}

fi

fi

done

### 给inotify\_back.sh脚本赋可执行权限

chmod +x /usr/local/inotify/inotify\_back.sh

### 运行inotify\_back.sh同步监控脚本和配置守护进程

nohup sh /usr/local/inotify/inotify\_back.sh > /usr/local/inotify/log.log 2&>1 &

nohup sh /usr/local/inotify/inotify\_back.sh > /dev/null 2&>>1 &

每隔一段时间监控一下

### 每两小时做1次全量同步

因为inotify只在启动时会监控目录，他没有启动期间的文件发生更改，他是不知道的，所以这里每2个小时做1次全量同步，防止各种意外遗漏，保证目录一致。

|  |  |
| --- | --- |
| 1  2 | crontab -e  \* \*/1 \* \* \* rsync -avz --password-file=/etc/rsync-client.pass /data/ root@192.168.0.18::data && rsync -avz --password-file=/etc/rsync-client.pass /data/ root@192.168.0.19::data |

报错：Please increase the amount of inotify watches allowed per user via `/proc/sys/fs/inotify/max\_user\_watches'.

解决：默认值是8192，echo 8192000 > /proc/sys/fs/inotify/max\_user\_watches