Backups con Rsnapshot

– Juan Gabriel Covas. 2015-2016

Mis notas en inglés acerca de este paquete de backup incluido en las distros más populares de Linux.

rsnapshot is a filesystem snapshot utility. It can take incremental snapshots of local and remote filesystems for any number of machines.

Local filesystem snapshots are handled with **rsync**. Secure remote connections are handled with **rsync over ssh**.

Since I've worked a lot with "rsync over ssh" for fast, custom mirroring and backup solutions, this old, established program looked perfect to me. I also liked the concept of rapidly having a **"snapshot server"** that simply pulls files from other boxes and easily allows you to access historical, incremental backups for them. It's a "standard package" on both Ubuntu and RedHat/CentOS distros and many others. Backend is mainly Perl, rsync and other standard tools (cp, ssh), so it's pretty KISS.

Allows for very simple restore operations and simple historical backups (i.e. automatic rotation for hourly, daily, weekly, monthly), saving a lot of disk space by using hard links.

```
RH/CentOS # yum install rsnapshot
Ubuntu    # aptitude install rsnapshot
```

The manpage for rsnapshot is pretty straightforward. Anyway, I had a taste of the features by looking at:

- http://www.cyberciti.biz/faq/redhat-cetos-linux-remote-backup-snapshot-server/
- https://www.howtoforge.com/set-up-rsnapshot-archiving-of-snapshots-and-backup-of-mysql-databases-on-debian

Things I like:

- Very easy to setup if you're used to passwordless ssh connections with pubkey authentication
- You can prepare "backup_scripts" to integrate backup-issues like database backups (dumps).
- You can automate the creation of "tar" packages to make real physical backups to i.e. DVDs.

My rsnapshot walktrough

Preparing a host to allow passwordless login by the snapshot server (so it can pull the files)

Add the public key to the file /home/user/.ssh/authorized_keys

```
$ if [ ! -d .ssh ]; then mkdir .ssh ; chmod 700 .ssh ; fi
$ echo "ssh-rsa XXXXXXXXXXXXXXXXXXXXXXX" >>~/.ssh/authorized_keys
$ chmod 600 .ssh/authorized_keys
```

Change at `/etc/ssh/sshd_config`

```
PermitRootLogin without-password (restart sshd!!!!)
```

TEST the connection so you bypass the known host check and obviously test that everything works

Preparing a mysql database to allow remote login (so the snapshot server can pull the dumps)

Make sure firewall allows TCP port 3306

Update CSF Firewall, file: `/etc/csf/csf.conf`

```
TCP_IN = "20,21,22,25,53,80,110,143,443,465,587,993,995,9876,3000:3050,3306"
TCP_OUT = "20,21,22,25,53,80,110,113,443,9876,3000:3050,3306"
```

Creating a new user "remotebackup" with "all privileges" that can connect from anywhere:

```
mysql> CREATE USER 'remotebackup'@'%' IDENTIFIED BY 'xxx';
mysql> GRANT ALL PRIVILEGES ON * . * TO 'remotebackup'@'%';
mysql> FLUSH PRIVILEGES;
```

Make sure `skip-networking` is commented at mysql server (so you allow networking)

`/etc/my.cnf.d/server.cnf` (requires mysql restart)

```
#skip-networking
```

Configuration changes done at: `/etc/rsnapshot.conf`

```
# This file requires TABS between elements
#
# Directories require a trailing slash:
#   right: /home/
#   wrong: /home
```

You can later test configuration using:

```
$ rsnapshot configtest
```

```
# All snapshots will be stored under this root directory.
#
snapshot_root   /.snapshots/
```

Define snapshot root:

```
snapshot_root /var/cache/rsnapshot/
snapshot_root /backupdrive/rsnapshot/
```

Uncomment cmd_ssh to use rsync over ssh:

```
#cmd_ssh      /usr/bin/ssh
cmd_ssh       /usr/bin/ssh
```

Comment hourly retain since we don't want it:

```
retain                hourly  6
#retain               hourly  6
```

Uncomment logfile so we get logs:

```
#logfile      /var/log/rsnapshot.log
logfile       /var/log/rsnapshot.log
```

Uncomment ssh_args so we can set a custom port, key, etc. for remote server(s): Note: we could have different configs by calling rsnapshot using "-c configgile"

```
#ssh_args     -p 22
ssh_args      -p 9876 -i /root/.ssh/snapshot_rsa
```

Changes done at /etc/cron.d/rsnapshot

I commented the hourly cron since I don't want it (also commented the hourly retain at .conf)

```
# This is a sample cron file for rsnapshot.
# The values used correspond to the examples in /etc/rsnapshot.conf.
# There you can also set the backup points and many other things.
#
# To activate this cron file you have to uncomment the lines below.
# Feel free to adapt it to your needs.

# 0 */4         * * *           root    /usr/bin/rsnapshot hourly
30 3    * * *           root    /usr/bin/rsnapshot daily
0  3    * * 1           root    /usr/bin/rsnapshot weekly
30 2    1 * *           root    /usr/bin/rsnapshot monthly
```

Official rsnapshot HOWTO

http://rsnapshot.org/rsnapshot/docs/docbook/rest.html

Pass arguments to conf backup/backup_script directive

Use the fourth column of a "backup". This is how you specify per-backup-point options to over-ride global settings. This extra parameter can take several options, separated by commas.

```
I.e. ssh_args=-p456
```

TESTS

Syntax test for config file:

```
# rsnapshot configtest
```

Run rsnapshot in test mode. This will print out a verbose list of the things it will do, without actually doing them. To do a test run, run this command:

```
# rsnapshot -t daily [hourly]
```

To see the sum total of all space used, try:

```
rsnapshot du
```

My database backup script for “backup_script” at rsnapshot.conf

I have written a “rsnapshot-mysql.sh” bash script to be able to pull all mysql databases from local or remote hosts. It writes each database on a directory and dumps tables separately to be more rsnapshot friendly. Since this complicates the db restore process, it automatically creates restore scripts too.

~~DISCUSSION|Comentarios~~