

## Create Easy Data Backups with Rsnapshot back-up tool on GNU / Linux

Author : admin



Backing up information on *Linux servers* is essential part of routine system administrator job. Thus I decided to write for those interested in how one can easily create backups of important data through a tiny tool called **rsnapshot** which I prior used to **make periodic data incremental backups on few of Debian Linux servers I manage**. In case you wonder why use rsnapshot and not just *rsync* - the reasons are 2.

- a. *Rsnapshot is very easy to configure and use* and you don't need to have deep understanding on *rsync* numerous options to use it.
- b. *Rsnapshot does support incremental data backups* - saving a lot of disk space on backup host.

Mentioning **incremental data backups** for some those term might be a news so I will in short explain here **what is Incremental Data Backups?**

*Incremental Data Backups* are such backups which only create new backup of system scheduled files to backup only whether there are changes in files to backup or new ones are added to directory/directories set to be routinely backed up. Incremental backups are often desirable as they consume minimum storage space and are quicker to perform than normal periodic whole data archiving (*differential backups*). *rsync* has also support for incremental backups but configuring it to do so takes time and requires extra time on reading and understanding how they work, so I personally prefer simplicity *rsnapshot* brings.

### 1. Installing rsnapshot with apt-get

Here is rsnapshot debian package description;

```
debian:~# apt-cache show rsnapshot|grep -i description -A 5
```

Description: local and remote filesystem snapshot utility  
rsnapshot is an rsync-based filesystem snapshot utility. It can take incremental backups of local and remote filesystems for any number of machines. rsnapshot makes extensive use of hard links, so disk space is only used when absolutely necessary.  
Homepage: <http://www.rsnapshot.org/>

As you can read from description, *rsnapshot* is a frontend command using rsync to make data backups.

Install of *rsnapshot* is done through;

**debian:~# apt-get install --yes rsnapshot**

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  rsnapshot
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/140 kB of archives.
After this operation, 598 kB of additional disk space will be used.
Selecting previously deselected package rsnapshot.
(Reading database ... 87026 files and directories currently installed.)
Unpacking rsnapshot (from .../rsnapshot_1.3.1-1_all.deb) ...
Processing triggers for man-db ...
Setting up rsnapshot (1.3.1-1) ...
```

## 2. Rsnapshot package content and Documentation

Once installed here is file *content of rsnapshot deb package*;

**debian:~# dpkg -L rsnapshot**

```
/.  
/usr  
/usr/share  
/usr/share/doc-base  
/usr/share/doc-base/rsnapshot  
/usr/share/doc  
/usr/share/doc/rsnapshot  
/usr/share/doc/rsnapshot/TODO  
/usr/share/doc/rsnapshot/changelog.gz  
/usr/share/doc/rsnapshot/Upgrading_from_1.1.gz  
/usr/share/doc/rsnapshot/examples  
/usr/share/doc/rsnapshot/examples/rsnapshot.conf.default.gz  
/usr/share/doc/rsnapshot/examples/utils  
/usr/share/doc/rsnapshot/examples/utils/backup_mysql.sh  
/usr/share/doc/rsnapshot/examples/utils/mysqlbackup.pl  
/usr/share/doc/rsnapshot/examples/utils/random_file_verify.sh  
/usr/share/doc/rsnapshot/examples/utils/rsnapreport.pl.gz  
/usr/share/doc/rsnapshot/examples/utils/make_cvs_snapshot.sh  
/usr/share/doc/rsnapshot/examples/utils/backup_pgsq.sh  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/CHANGES.txt  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/rsnapshotDB.pl.gz  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/INSTALL.txt  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/TODO.txt  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/rsnapshotDB.xsd  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/rsnapshotDB.conf.sample  
/usr/share/doc/rsnapshot/examples/utils/rsnapshotdb/README.txt  
/usr/share/doc/rsnapshot/examples/utils/rsnapshot-copy  
/usr/share/doc/rsnapshot/examples/utils/backup_rsnsnapshot_cvsroot.sh  
/usr/share/doc/rsnapshot/examples/utils/backup_dpkg.sh  
/usr/share/doc/rsnapshot/examples/utils/sign_packages.sh  
/usr/share/doc/rsnapshot/examples/utils/mkmakefile.sh  
/usr/share/doc/rsnapshot/examples/utils/rsnaptar  
/usr/share/doc/rsnapshot/examples/utils/rsnapshot_invert.sh  
/usr/share/doc/rsnapshot/examples/utils/rsnapshot_if_mounted.sh  
/usr/share/doc/rsnapshot/examples/utils/README  
/usr/share/doc/rsnapshot/examples/utils/debug_moving_files.sh
```

```
/usr/share/doc/rsnapshot/examples/utils/backup_smb_share.sh
/usr/share/doc/rsnapshot/README.gz
/usr/share/doc/rsnapshot/changelog.Debian.gz
/usr/share/doc/rsnapshot/copyright
/usr/share/doc/rsnapshot/README.Debian
/usr/share/doc/rsnapshot/html
/usr/share/doc/rsnapshot/html/rsnapshot-HOWTO.en.html
/usr/share/doc/rsnapshot/NEWS.Debian.gz
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/rsnapshot
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/rsnapshot.1.gz
/usr/share/man/man1/rsnapshot-diff.1.gz
/usr/bin
/usr/bin/rsnapshot-diff
/usr/bin/rsnapshot
/var
/var/cache
/var/cache/rsnapshot
/etc
/etc/cron.d
/etc/cron.d/rsnapshot
/etc/rsnapshot.conf
/etc/logrotate.d
/etc/logrotate.d/rsnapshot
```

To get basic idea, on rsnapshot and how it can be configured and run manually as well as how it can be set-up to run periodic via a cronjob *README* shipped with package is a good start point.

```
debian:~# zless /usr/share/doc/rsnapshot/README.gz
```

....

It is also useful to check program documentation in HTML, whether you have some text browser installed - i.e. lynx or links:

```
debian:~# links /usr/share/doc/rsnapshot/html/rsnapshot-HOWTO.en.html
```

Note that many of information in **rsnapshot-HOWTO** is related to how **rsnapshot** is installed manually from source, so for Deb based distro users reading these sections can be safely skipped. For Debian users hence it is useful to read howto from section 4.A onwards. **man rsnapshot's** Examle section is very good reading too as it gives a lot of use scenarios necessary in more complicated backup situations.

### 3. Configuring Rsnapshot - Setting Data Directories to Backup

Configuration of *Rsnapshot* is done through **/etc/rsnapshot.conf** file. There is plenty of comments in file, so opening in text editor and taking few minutes to read commented lines is necessary. Configuration options just like with most Linux tool config files is done through config directives, not commented.

```
debian:~# cat /etc/rsnapshot.conf |grep -v '"#"'|uniq
```

```
config_version 1.2
```

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

```
cmd_rm /bin/rm
```

```
cmd_rsync /usr/bin/rsync
```

```
cmd_logger /usr/bin/logger
```

```
interval hourly 6
```

```
interval daily 7
```

```
interval weekly 4
```

**verbose     2**

**loglevel   3**

**lockfile   /var/run/rsnapshot.pid**

**backup   /home/     localhost/**

**backup   /etc/     localhost/**

**backup   /usr/local/   localhost/**

Above config options are clear to understand, there is interval of backups to set (hourly, daily, weekly), verbose level of *rsnapshot* backup operation log file, lockfile which will be used by *rsnapshot* to prevent duplicate *rsnapshot* runs and last backup directive in which you need to specify what needs to be backed up. In config file there is also commented variable for creating *rsnapshot* backup once a month

**#interval   monthly 3**

If you need to create backups once a month uncomment it.

In backup directive add all directories from filesystem which need to have routine backup, for example I keep my Apache Web server files in */var/www/*, store various install software in

*/root/*

and keep backup of Qmail (Vpopmail) old emails kept in

*/var/vpopmail* .

To make *rsnapshot* backup those I add after rest of backup directives:

**backup   /var/www/   localhost/**

**backup   /var/vpopmail/ localhost/**

**backup   /root/   localhost/**

It is good practice to change `snapshot_root` directive to `/root/.backups` or whether you prefer to keep `snapshot_root` to default `/var/cache/rsnapshot` at least link with `ln` command `/root/.backups` to `-> /root/.backups`.

```
debian:~# ln -sf /var/cache/rsnapshot /root/.backups
```

If you change `snapshot_root` to `/root/.backups`, don't forget to create `/root/.backups` and set `chmod` dir permissions only readable to owner, i.e.:

```
debian:~# mkdir /root/.rsnapshot
debian:~# chmod -R 700 /root/.backups
```

Note that, it is important to use tab delimiters, everywhere in `/etc/rsnapshot.conf`, if you use space key delimiter instead of Tab you will end up with errors preventing `rsnapshot` to run.

#### 4. Testing `rsnapshot` configuration and launching it first time

I will say it once again use Tab key for delimiters in config. It was my mistake on first time `Rsnapshot` launch to use spaces to delimiter my config options, thus testing my configuration, `rsnapshot` print an error and failed:

```
debian:~# rsnapshot configtest
```

```
-----
rsnapshot encountered an error! The program was invoked with these options:
/usr/bin/rsnapshot configtest -----
ERROR: /etc/rsnapshot.conf on line 199: ERROR: backup /var/www/
localhost/
ERROR: -----
ERROR: Errors were found in /etc/rsnapshot.conf, ERROR: rsnapshot can not
continue. If you think an entry looks right, make
ERROR: sure you don't have spaces where only tabs should be.
```

After changing, Space delimiters with Tabs and re-running *rsnapshot configtest* if all fine you get:

```
debian:~# rsnapshot configtest  
Syntax OK
```

Once all good with config to launch Rsnapshot do its first complete incremental data backup, to display what rsnapshot will backup and what exact rsync invocations will it use type:

```
debian:~# rsnapshot -t hourly
```

```
echo 5644 > /var/run/rsnapshot.pid  
mv /var/cache/rsnapshot/hourly.2/ /var/cache/rsnapshot/hourly.3/  
mv /var/cache/rsnapshot/hourly.1/ /var/cache/rsnapshot/hourly.2/  
native_cp_al("/var/cache/rsnapshot/hourly.0", \  
    "/var/cache/rsnapshot/hourly.1")  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded /home \  
    /var/cache/rsnapshot/hourly.0/localhost/  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded /etc \  
    /var/cache/rsnapshot/hourly.0/localhost/  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded \  
    /usr/local /var/cache/rsnapshot/hourly.0/localhost/  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded \  
    /var/www /var/cache/rsnapshot/hourly.0/localhost/  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded \  
    /var/vpopmail /var/cache/rsnapshot/hourly.0/localhost/  
/usr/bin/rsync -a --delete --numeric-ids --relative --delete-excluded /root \  
    /var/cache/rsnapshot/hourly.0/localhost/  
touch /var/cache/rsnapshot/hourly.0/
```

To launch backup first time manually:

```
debian:~# rsnapshot hourly
```

Depending on backedup data (*Mega/Giga/Terabytes*) size and the number of files



which had to be backed up, backup takes from minutes to hours.

Note that it is always good idea to create backups on separate hard disk configured in some kind of *RAID array*, preferably (RAID 1 or RAID 5). Creating backups on separate hard disk has numerous advantages, the most important one is it doesn't put too much Input / Output (I/O) stress on hard disk and thus will not create server downtimes on *High traffic - Busy servers slow old Hard Disks* or *servers with Big amount of I/O HDD read/writes* .

## 5. Enabling Rsnapshot to create backups via scheduled cron job

On package install Rsnapshot creates a skele file for running via cronjob in `/etc/cron.d/rsnapshot`.

```
debian:~# cat /etc/cron.d/rsnapshot
```

```
# 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
```

To make *hourly*, *daily*, *weekly*, *monthly* backup uncomment one of above 4 lines. For paranoid admins scared to loose even a bit of data, hourly data is a good solution. For me personally I prefer configuring weekly backups for the reason I routinely monitor servers - keeping an eye regularly on `dmesg` and [checking Linux smartd / smartmontools logs to find out whether a hard disk or RAID has bad blocks](#)

## 6. Checking backup size / backup difference and backup structure

Checking size of backups can be done by using standard `du` command on backup directory:

```
debian:~# du -hsc /var/cache/rsnapshot/*
4.3G /var/cache/rsnapshot/hourly.0
4.5M /var/cache/rsnapshot/hourly.1
68M /var/cache/rsnapshot/hourly.2
4.4G total
```

*rsnapshot* also has *du* argument via which backup size can be viewed:

```
debian:~# rsnapshot du 4.3G /var/cache/rsnapshot/hourly.0/
4.5M /var/cache/rsnapshot/hourly.1/
68M /var/cache/rsnapshot/hourly.2/
4.4G total
```

As you can see each new incremental backup is with new number after *hourly{0,1,2}* etc.

To check difference between two different backups:

```
debian:~# rsnapshot diff /var/cache/rsnapshot/hourly.0/
/var/cache/rsnapshot/hourly.1/
Comparing /var/cache/rsnapshot/hourly.1 to /var/cache/rsnapshot/hourly.0
Between /var/cache/rsnapshot/hourly.1 and /var/cache/rsnapshot/hourly.0:
660 were added, taking 3728377727 bytes;
492 were removed, saving 17623 bytes;
```

Structure of backed up files is identical to normal copy of files without any compression:

```
debian:~# cd /root/.backups/hourly.0/localhost/
debian:~/.backups/hourly.0/localhost# ls
etc/ home/ root/ usr/ var/
```

## **7. Restoring files or directory from rsnapshot backup**

To restore let's say `/var` directory *cd* into it:

```
debian:~/.backups/hourly.0/localhost# cd var
debian:~/.backups/hourly.0/localhost/var#
```

Then use *rsync* as follows:

```
debian:~/.backups/hourly.0/localhost/var# rsync -avr * /
```

## **8. Creating rsnapshot backups from remote server via SSH protocol**

In `/etc/rsnapshot.conf` you should have set SSH port on which remote server is accepting SSH connections. Standard port is 22, however it is wise to configure on backup server SSH to listen to some other non standard port.

In config variables to look on are:

```
ssh_args -p 22
```

and

Onwards to enable remote login via ssh uncomment in `/etc/rsnapshot.conf` :

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

to

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

Before starting rsnapshot to create backups on remote host2 you need to [Configure automatic SSH passwordless login by generating DSA or RSA key pair](#) between *host1* and *host2*. Where *host1* is machine on which rsnapshot is run and to which backups will be copied from *host2*

Once *passwordless ssh* to remote host is active, to force rsnapshot create backups from *host1* you will need to add near end of */etc/rsnapshot.conf* .

```
backup root@host2.com:/root/ host2.com/
```

The same way you can add a number of remote hosts from which periodic backups will be created to central *host1*. Only condition is on each node - *host3*, *host4*, *host5*.

```
backup root@host3.com:/root/root/ host3.com
```

```
backup root@host4.com:/home/ host4.com
```

```
backup root@host4.com:/var/ host4.com
```

To create on *host1* public key (*id\_dsa.pub*) file with command:

```
debian:~# ssh-keygen -t dsa
```

```
...
```

```
....
```

```
debian:~# ssh-copy-id -i ~/.ssh/id_dsa.pub root@host3
```

Once all hosts that needs to get backed up to central backup host - *host1*. To test if backups gets uploaded manually issue:

```
debian:~# rsnapshot -v hourly
```

```
...
```

Rsnapshot has a number of other scripts which can be easily integrated with it in */usr/share/doc/rsnapshot/examples/utils*.

Inside you can find example scripts on how to create MySQL / PostgreSQL database backup, Samba Share backups, backup CVS repositories and so on. The scripts can be easily modified and work with mostly any data or protocol with a bit of tweaking. Short description of each of example scripts can be found in */usr/share/doc/rsnapshot/examples/utils/README*