vbackup Documentation

Release 1.0.0

Stefanos Harhalakis

February 06, 2016

Contents

VBackup is a modular, command-line, backup utility, aimed at automated backup procedures, mostly for remote systems.

CHAPTER 1

VBackup - A modular backup utility

vbackup is an extensible CLI backup tool. Ultimately it is a collection of scripts that perform backup of systems. It is based on bash and should support most of the unices out there (if not then please report it as a bug).

From a user's point of view, vbackup simplifies the backup process by using just a config file for everything that otherwise would require a custom made script.

Vbackup can backup:

- Postgresql databases
- MySQL databases
- Filesystems using GNU tar
- XFS filesystems using xfsdump
- dpkg and RPM databases
- MBR, Partition tables, MD and LVM information
- OpenLDAP database

The backup procedure is completely customizable and extendable.

vbackup supports multiple backup strategies and each one can supports multiple levels. For example, one can have a backup strategy for filesystem backups and another for VM backups or for testing

CHAPTER 2

Installation

If you are not installing from packages then you most probably need to do something like this:

```
# ./configure --sysconfdir=/etc
# make
# make install
```

I.e. standard autoconf installation.

2.1 Requirements

- GNU find
- GNU tar - for the tar method
- xfsdump - for the xfsdump method
- RPM - for the rpm method
- dpkg - for the dpkg method
- openldap - for the openldap method
- openssl - for the x509 method
- mdadm and lvm - for the mbr method

CHAPTER 3

Overview

3.1 Usage

For usage run:

```
# vbackup --help
```

For a list of backup methods (scripts) run:

```
# vbackup --list
```

To get some information about a module run:

```
# vbackup --help <module>
```

example:

```
# vbackup --help xfsdump
```

Once setup, you can perform level 0 and level 5 backups with:

```
# vbackup 0
# vbackup 5
```

If you use multiple backup strategies then you can perform a backup for each one of them with:

```
# vbackup strategy1 0
# vbackup strategy2 0
```

3.2 High level overview

vbackup runs by scanning a configuration directory for configuration files. This directory has a vbackup.conf file and a series of config files that dictate what vbackup should do. For example, one may indicate that a PostgreSQL database should be backed up, the next one can indicate that /home should be backed up and a final one would instruct vbackup to copy the backups to a remote system using scp.

Each strategy's configuration is stored under /etc/vbackup:

- The default strategy (i.e. the one with no name) is the one stored at /etc/vbackup/rc.d
- Each named strategy is stored under /etc/vbackup/rc.<strategy>.d

See How it works for more details.

CHAPTER 4

Setup

4.1 Setup - the beginner's way

There is a wizard script that will help you create basic configuration files. Most probably you will want to use it for the first time. For more advanced configuration and fine-tuning you will have to adjust the configuration files yourself. Read the rest of this file to fully understand how you can use vbackup.

To launch the wizard run:

```
# vbackup-wizard
```

and answer its questions. The wizard also serves as an introduction.

Note that this will only create basic backups, even though that may be all that you need. For more advanced stuff you'll need to use the CLI or the manual method

4.2 Setup - the CLI way

4.2.1 A simple backup

Let's assume you want to:

- Store backups under /mnt/backup/
- Backup /home
- Backup /etc
- Backup your partition information in case disaster hits

Run the following and answer the questions properly. You change your config later by editing config files by hand.

```
# vbackup --rc --init
    # Enter "/mnt/backup/%L1%-%D%", without the quotes, for DESTDIR0
# vbackup --rc --add 10-exist.exist     # Just press enter when asked
# vbackup --rc --add 20-mbr.mbr         # Press enter on all questions
# vbackup --rc --add 30-home.tar
    # Enter "/home /etc", without the quotes, for DIRS
    # Press enter for everything else
```

And you're done. This initialized the default strategy under /etc/vbackup/rc.d and added 3 config files:

- One to ensure that /mnt/backup is present

- One to backup MBR and LVM data
- One to backup /home and /etc using tar

You can verify that all is good by running:

```
# vbackup --check
```

You can perform backups using:

```
# vbackup 0
- or, for less verbose output -
# vbackup -d 4 0
```

You can perform higher level (incremental) backups by passing a different parameter to vbackup:

```
# vbackup 5
```

this will base the backup on a previous level. Since the only lower level backup is 0, it will be based on that. All methods that support incremental backups will honor this. In this case this is tar. The mbr method will still perform full backups, regardless of the level, as it doesn't support incremental backups.

4.2.2 A more complex setup

Let's assume you want to

- Backup to an NFS mounted space that will be mounted on demand under /mnt/nfsbackup
- Perform backup of /home using xfsdump
- Perform backup of /boot and /etc using tar
- Perform mysql and postgresql backup
- Name this backup strategy "big"

Run the following:

```
# vbackup --rc --init big
    # Enter "/mnt/nfsbackup/%D1%-%L%", without the quotes, for DESTDIR0
# vbackup --rc --add big 00-remote.nfsmount
    # Enter the NFS URI to mount (e.g. mynfs.local:/mnt/backup) in NFSDIR
    # Enter the mount point (/mnt/nfsbackup) in MOUNTPOINT
# vbackup --rc --add big 10-full.mysql
    # Answer as needed. For Debian systems you can just press enter
    # on all options
# vbackup --rc --add big 10-full.pgsql
    # Same as above
# vbackup --rc --add big 20-home.xfsdump
    # Enter /home for PARTITIONS
# vbackup --rc --add big 30-system.tar
    # Enter "/etc /boot" for DIRS
# vbackup --rc --add 99-remote.umount
    # Enter /mnt/nfsbackup for MOUNTPOINT
```

Now, see what you have created and check the configuration:

```
# vbackup --rc --list big
# vbackup --check big
```

Level 0 vackups are performed with:

```
# vbackup big 0
```

Level 5 backups are performed with:

```
# vbackup big 5
```

4.3 Setup - by hand

Let's assume the complex scenario above

1. Go to /etc/vbackup and create a folder named rc.big.d. This will hold all the configuration scripts.
2. Copy from the samples directory ($prefix/share/vbackup/samples) the following scripts to /etc/vbackup/rc.big.d:
   - sample.nfsmount
   - sample.umount
   - sample.xfsdump
   - sample.mysql
   - sample.pgsql
   - sample.umount
   - vbackup.conf.sample
3. Rename sample.nfsmount to 00-remote.nfsmount (any name with the extension .nfsmount will do) and edit it. Change the values of the variables to fit your needs.
4. Rename sample.umount to 99-remote.umount and edit it. Just set the mountpoint to umount.
5. Rename sample.mysql to 10-full.mysql and edit it. Read the note at the top of the script and set variables as needed.
6. Rename sample.pgsql to 10-full.pgsql and edit it.
7. Rename sample.xfsdump to 20-home.xfsdump and edit it.
8. Rename sample.tar to 30-system.tar an edit it.
9. Rename vbackup.conf.sample to vbackup.conf and edit it
10. Check the configuration using:

```
    # vbackup --check big 0
    # vbackup --check big 5
```

To perform level 0 backup run:

```
# vbackup big 0
```

To perform level 5 backup run:

```
# vbackup big 5
```

CHAPTER 5

How it works

When you run:

```
# vbackup 0
```

- The script enters directory /etc/vbackup/rc.d and searches for files named XX-name.module, where <XX> is a number and <module> is a backup module/method.
- All configurations are examined in an alphabetical order, so that the <XX> is used to indicate the order. These files apply to all backup levels.
- The script also checks for files named XX-name-level.module, where <level> is a certain level. These files apply only to that level and are ignored in other levels.
- For each configuration file, the corresponding module (<module>) is executed using the parameters of the configuration file. A module may be used from zero to many times. E.g. there can be many .tar files in there

The verbosity of the backup procedure may be controlled using the -d parameter. Level 4 should let you know of any special cases while keeping everything else for itself

Backups that fail will not stop the backup procedure. There are however some errors that will stop everything (ex: when failing to mount a filesystem). However, backups that fail will cause vbackup to exit with non 0 code.

5.1 DESTDIR0

DESTDIR0 holds the destination directory. The name should be an absolute path. Certain character sequences are translated during runtime:

%L% is replaced with the backup level, as specified in the command line. E.g. 1

%D0% is replaced by a number for each day of the week. Monday is 1, Sunday is 7

%D1% is replaced by YYMMDD where YY are the two last digits of the year, MM is the two-digit month number (e.g. 05 for May) and DD is the two-digit day of the month number

%D2% is replaced by YYMMDDHH where YYMMDD is as above and HH is the two digit 24 hour-of-the-day number

%D3% is replaced by YYMMDDHHmm where YYMMDDHH is as above and mm is the two digit minutes number within the hour

For example, /mnt/backup/%D1%-%L% while create a separate backup directory per day, per level, like this:

```
drwxr-xr-x 8 v13 v13 70 Jun 28  2015 150628-1
drwxr-xr-x 8 v13 v13 70 Jul 20  2015 150720-5
drwxr-xr-x 8 v13 v13 70 Aug 18 01:23 150818-5
```

CHAPTER 6

Available modules

The following modules are available:

dpkg:	Backup debian package lists
exec:	Execute some shell commands
exist:	Check whether a file/dir exists
ftar:	Filesystem backup using tar and find
gpg:	Encrypt a file or a directory using gpg
mbr:	Backup MBRs, Partition Tables, MD and LVM information
mount:	Mount a partition
mysql:	Dump mysql databases
nfsmount:	Mount an NFS share
off:	Don't do anything
openldap:	Backup OpenLDAP database
pgsql:	Dump pgsql databases
rm:	Remove a directory recursively
rpm:	Backup RPM database
scp:	Remote file copy using scp
tar:	Perform filesystem backup using GNU tar
umount:	Umount a partition
x509:	Encrypt a file or a directory using x509 certificates
xfsdump:	Perform filesystem backup using xfsdump

CHAPTER 7

Known issues

The xfsdump method cannot possibly support multi-level backups for multiple strategies. This is because xfsdump stores the state under /var/lib/xfsdump/inventory with no option to override this. This means that multiple strategies wills hare the same state information, meaning that a level 5 backup of strategy A may end up depending on the level 0 backup of strategy B.

I.e: Do not use xfsdump with multiple strategies and multiple levels or else you *will* lose data. If you have a good idea on how to solve this, please send a bug report.

CHAPTER 8

License and Copyright

8.1 Copyright

8.2 License

8.3 Additional

vbackup uses the rtd (readthedocs) theme, which is distributed under a combination of:

- MIT License (Expat)
- BSD 2-Clause License
- OFL-1.1
- Apache-2.0 License

CHAPTER 9

Extending vbackup

If you're interested in extending vbackup then read below and have a look at the existing modules (I recommend mount and tar/xfsdump). A template for new modules is available with the name 'skel' under the scripts/ directory of the source code.

If you believe you've created something that will be useful for others too, I'll be glad to include it in the distribution (preserving your copyright of course)

9.1 File tree

The installation layout should comply with the latest FHS (2.3 as of Nov 2006). Everything different than that should be reported as bug.

```
${bindir}/                      The vbackup program

${sysconfdir}/
       vbackup/
               rc.d/            Backup configurations
               backup.daily/    Links to appropriate rc.d scripts
                       vbackup.conf    Per backup global configuration file
               backup.weekly/   ....
               backup.monthly/  ....

${datadir}/
       vbackup/
               bin/             Core scripts (run) used by vbackup
                                (They are standalone scripts)
               helpers/         Helper scripts (not standalone)
               scripts/         The backup scripts
               wizard/          Wizard configuration files
               samples/         Sample configurations. Also used for --rc

${docdir}/
       vbackup/                 Documentation
```

9.2 Backup scripts

Each backup script must provide:

9.2.1 Functions

do_help() A function to display help and return

do_check_conf() Check whether all configuration variables are ok

Return: 0: OK, 1: Error

May display error messages

do_run() Do the backup

9.2.2 Variables

NAME	The script name (“psql”)
VERSION	The script version (“1.0.0”)
DESC	A short description (“Backup a postgresql database”)
COPYRIGHT	Copyright (“Copyright (c) 2006 Harhalakis Stefanos”)
LICENCE	The license of the script (“GPLv2”)
CONTACT	A contact email for bugreports etc (“v13@v13.gr”)

9.2.3 It may expect

- If the configuration file includes a DESTDIR options, it will be prepended with the DESTDIR0 prefix. It will also be transformed by changing %X% variables (for example %D1%)
- A variable named ABORT is set to 1 if a script previously exited with error code 2 (see below). Most scripts should abort with exit code 0 when they detect this. For example: a script that checks for free space may exit with error code 2. The ABORT variable will be set automatically. All backup scripts must detect this and do nothing. The umount script will ignore this and actually try to umount the directory.
- A variable named LEVEL that indicates the backup level as passed to the vbackup command.
- A variable named STRATEGY that indicates the strategy as passed to the vbackup command. This will be empty if no strategy was specified.
- A variable named STATEDIR that points to a script-specific directory to store state. Note that the directory may not exist. If needed, the script should use h_ensuredir to create the directory

9.2.4 It must return (from each function)

0	Everything were OK
1	Error occurred - continue backups
2	Error occurred - abort backup sequence
3	Error occurred - force abort of backup sequence

Notes:

- When a script returns 2, the flag ABORT is set to 1. This is checked by backup scripts and they will not do anything at all. Scripts like umount may ignore this and umount the directory
- When a script returns 3, the backup is immediately aborted. This should be used by scripts like mount

9.3 Configuration files:

Some configuration files may include a DESTDIR directive. For example, '%D1%' which will be replaced by YYMMDD of the current date E.g.: /tmp/backups/filesystem/%D1% will be converted to: /tmp/backups/filesystem/061117 (as of 17 Nov 2006)

The following special sequences are recognised:

%D0%	Day of the week (1-7). 1 is Monday
%D1%	YYMMDD of the current date
%D2%	YYMMDDHH
%D3%	YYMMDDHHMM
%L%	The backup level

9.4 Configuration directives:

Configuration directives should preserve the same name across scripts when possible. Common directives are (using regexp):

DESTDIR	Where to backup to
DATABASES	The databases to backup, for DB backups
PASSWORD	To hold a password
.*USER	To hold a username (examples: PGUSER, MYUSER etc..)

NOTE! The directive USER should never be used. I repeat NEVER USE THE USER DIRECTIVE! Same thing applies for other well known directives that may be set by sh/bash.

9.5 Common configuration variables:

Common configuration variables are variables that are set in vbackup.conf or in other configuration files. The vbackup.conf entry is the main one but it may be overridden by individual configuration files.

COMPRESS	Whether to compress the backup or not (yes/no, on/off, 1/0)

9.6 Sample files:

Sample configuration files are parsed by vbackup when "–rc –add" is used. The configuration files are parsed using the following logic:

- Ignore all comments lines from the top of the file until an empty line is found
- If in the above block of comments there's a line '## No autoconfig' then the '–rc –add' will just copy the file and not try to autoconfigure it. This is useful for example in the case of the exec plugin.
- All files of the sample configuration file end up in the target configuration file. This means that all comments and empty lines will be copyied over there.
- Look for blocks of directives. All blocks are assumed to have some comments followed by one or more configuration directives.
- All the comments of a block will be displayed to the user. Only lines that start with '# ' (hash and a space) are assumed to be comments.
- When a line that doesn't start with '# ' is found then it's interpreted as a configuration directive.

- If the line start with # (e.g: #DESTDIR="test") then it is assumed not to have a default value. However, a line of text will be displayed to the user saying: "Sample value: test".
- If the line doesn't start with # (e.g: DESTDIR="test") then it is assumed that the value is the default value. It will be presented both as a sample line and as default value. If the user presses enter then he will use that value.

9.7 Wizard:

Inside the wizard dir, each script may place a configuration file. This file will describe a wizard step for initial configuration creation.

Each wizard configuration file includes:

```
NAME="the name of the configuration script"
PRI="Priority of configuration file"
```

w_do_ask() A function that will be called to ask for configuration parameters. This function may be called multiple times to answer different questions. The first argument mandates the question to me asked Initialy, this function is called with "ENABLE" as the first argument. All capital letter names should be reserved for vbackup usage. This function must set the variable ASK_NEXT to "" whenever the series of questions is finished, or to a name that will be passed as the first argument to the next function call.

The last function call must return 0 to indicate that this method is active or 1 to indicate that it is not

Example call tree:

- do_ask ENABLE ==> Set ASK_NEXT=path
- do_ask path ==> Set ASK_NEXT=level
- do_ask level ==> Set ASK_NEXT="", return code=0
- finished asking questions, method is enabled

This function may also write to the temporary file $CFG:

```
    echo 'DIRS="/tmp"' >> $CFG
    echo '# Comment' >> $CFG
```

This file will be available when w_get_config() is called. Typically, this should contain the majority of the configuration options except maybe from static options.

w_get_config() A function that should print to its stdout the configuration file.

If you would like to suggest or contribute additional modules, please do.