Vitalus Documentation

Release 0.4.1

F. Boulogne

Sep 27, 2017

Contents

Author François Boulogne

Download Stable version

Developer's corner github.com project

Generated Sep 27, 2017

Version 0.4.1

Vitalus is a rsync wrapper. Rsync is a good atomic tool, but it needs to be wrapped to have a real backup solution. Backup solutions are generally too basic or very difficult. This one fits my needs.

Contents:

CHAPTER 1

How to install?

Requirements

- python3
- rsync

Package manager

An AUR package AUR package is available.

PyPI

See Pypi

To install with pip:

```
pip install Vitalus
```

Manual installation

Download

```
python setup.py --root=/usr/local/bin
```

CHAPTER 2

Example

This is an example. To know more about the API, read the *Vitalus library* documentation.

```
#!/usr/bin/env python

# This file is an example
# It is designed to be run frequently
# by a cron job (e.g. each few hours)

import Vitalus.vitalus as vitalus

# Create an instance
# Default: log are in ~/.backup. You can change it with log_path=/foo
my_backup = vitalus.Vitalus()


# When I want to check my script, I set the log level to DEBUG.
# This give you a chance to understand what's going wrong.
# The default is INFO, so WARNING or CRITICAL messages are printed in logs
# Logs are storred in ~/.backup. They are rotated once a day.
my_backup.set_log_level('DEBUG')

# This is my external disk
my_backup.set_destination('/media/disk/backup')

# I add a job for 'my_documents'
# I want to keep increments (default: False)
my_backup.add_rsyncjob('my_documents', '/home/myself/documents', history=True)

# Copy data excepting *.html files.
# This is a rsync filter rule (man rsync to learn more)
# filter is a tuple. Don't forget the coma.
my_backup.add_rsyncjob('my_data', '/home/myself/data', history=True, filter=('- *.html
↪',))

# Another job
```

```
# minimal duration between two backups: 5 hours (default: 24h)
my_backup.add_rsyncjob('thunderbird', '/home/myself/.thunderbird', period=5,␣
↪history=False)


# Sync my home space on a server to my disk
# Keys, without password must be configured
my_backup.add_rsyncjob('server', 'myself@server.tld:.')


# Let's go!
my_backup.run()

# Read the log in ~/.backup
```

CHAPTER 3

Vitalus library

class vitalus.**Vitalus**(*log_path='~/.backup'*, *log_rotation=30*, *force=False*)

Bases: object

This is the main class to declare and run backup tasks.

Params log_path directory for logs and database

Params log_rotation How many days logs are kept

Parameters **force** – if True, do not perform timebase check.

All jobs will be run. :type force: bool

add_customjob(*name*, *job*, **args*)

Add a custom job.

Parameters

- **name** (*string*) – backup label
- **period** (*float*) – min time (hours) between backups
- **job** – object representing a job
- **args** – arguments to initialize the job

Note: This is particularly useful to do not use the *run()* function implemented in *RsyncJob*. For instance, a git-annex repository can be synchronized by writting a proper class to use git-annex features.

---

add_rsyncjob(*name*, *source*, *period=24*, *history=False*, *duration=50*, *keep=10*, *filter=None*)

Add a rsync job.

Parameters

- **name** (*string*) – backup label
- **source** (*string*) – backup from...
- **destination** (*string*) – backup to...

- **period** (*float*) – min time (hours) between backups
- **history** – Activate (True) or desactivate (False) snapshots

or perform a simple copy (None). :type history: bool or None :param duration: How many days snapshotss are kept :type duration: int :param keep: How many snapshots are (at least) kept :type keep: int :param filter: filters :type filter: tuple

Raises ValueError – if destination if not set

Note: Filter syntax is the same of rsync. See "FILTER RULES" section in the rsync man page.

history: if set to True or False, the date is written in the path. This is a feature. It leaves you the possibility to switch on/off the history. If you don't want the date in the path, set the value to None.

---

run()
Run all jobs

set_destination(*destination*, *guid=(None*, *None)*)
Set the destination of the backup if uid or gid are None, files owner are not changed

Parameters

- **destination** (*string*) – destination path
- **guid** (*tuple*) – (uid, gid) for destination

set_log_level(*level='INFO'*)
Set the logger level (INFO, CRITICAL, DEBUG, ERROR, FATAL)

Parameters **level** (*string*) – Loger level

class job.**Job**(*log_dir*, *destination*, *name*, *source*, *period*)
Bases: object

Class containing a job

Parameters

- **log_dir** (*string*) – Log directory path
- **destination** (*string*) – Destination path
- **name** (*string*) – Job name
- **source** (*string*) – Source path
- **period** (*float*) – Min duration between two backups (in seconds)

Note: —

---

run(*uid=None*, *gid=None*)
Run the job.

exception job.**TARGETError**(*message=''*)
Bases: Exception

Exception for target validity

Parameters **message** (*string*) – Message

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class job.**Target**(*target*)
Bases: object

A target is a source or a destination. It can be a local directory or a remote one (SSH)

Parameters **target** (*string*) – a target

check_availability()
Check if the target is available For SSH host, it means it's reachable

Raises TARGETError – if not available

is_local()
Check if the target is a directory

Returns bool – True if it is a directory

is_ssh()
Check if the target is a 'SSH' host

Returns bool – True if it is a SSH host

class rsyncjob.**RsyncJob**(*log_dir*, *destination*, *name*, *source*, *period*, *snapshot*, *duration*, *keep*, *force*, *guid*, *filter*)
Bases: Vitalus.job.Job

Class containing a rsync job

Parameters

- **log_dir** (*string*) – Log directory path
- **destination** (*string*) – Destination path
- **name** (*string*) – Job name
- **source** (*string*) – Source path
- **period** (*float*) – Min duration between two backups (in seconds)
- **snapshot** (*bool or None*) – Activate (True) or desactivate (False) snapshots or simple (None) copy
- **duration** (*int*) – How many days snapshots are kept
- **keep** (*int*) – How many snapshots are (at least) kept
- **force** (*bool*) – overide the timebase check, no min. duration.
- **guid** (*tuple*) – (uid, gid) for destination
- **filter** (*list*) – Rsync filters

Note: Source and destination path can be either real path or a ssh login joined to the path by a : character.

if uid or gid are None, files owner are not changed

---

run(*uid=None*, *gid=None*)
Run the job.

CHAPTER 4

Logwatch

Goal

If you have very sensitive data, desktops, servers, I would advice to monitor your machines. Logwatch is a simple set of script which reads your logs and send you tidy and readable emails. The goal of this page is to setup logwatch for Vitalus.

Setup

- In services/vitalus.log

```
# this is in the format of <name> = <value>.  Whitespace at the beginning
# and end of the lines is removed.  Whitespace before and after the = sign
# is removed.  Everything is case *insensitive*.

# Yes = True  = On  = 1
# No  = False = Off = 0

Title = "Vitalus"

# Which logfile group...
LogFile = vitalus
```

- In logfiles/vitalus.conf (don't forget to adapt the paths)

```
# What actual file?  Defaults to LogPath if not absolute path....
#Solaris is /var/cron/log -mgt
LogFile = /root/.backup/backup.log

# If the archives are searched, here is one or more line
# (optionally containing wildcards) that tell where they are...
Archive = /root/.backup/backup.log.*
```

- In scripts/services/, put the script available in Vitalus sources in logwatch/scripts/services/.

CHAPTER 5

API

`Vitalus.vitalus` — Main class

class `vitalus.`**`Vitalus`**(*log_path='~/.backup'*, *log_rotation=30*, *force=False*)

This is the main class to declare and run backup tasks.

Params log_path directory for logs and database

Params log_rotation How many days logs are kept

Parameters **`force`** – if True, do not perform timebase check.

All jobs will be run. :type force: bool

`add_customjob`(*name*, *job*, **args*)

Add a custom job.

Parameters

- **`name`** (*`string`*) – backup label
- **`period`** (*`float`*) – min time (hours) between backups
- **`job`** – object representing a job
- **`args`** – arguments to initialize the job

Note: This is particularly useful to do not use the *run()* function implemented in *RsyncJob*. For instance, a git-annex repository can be synchronized by writting a proper class to use git-annex features.

`add_rsyncjob`(*name*, *source*, *period=24*, *history=False*, *duration=50*, *keep=10*, *filter=None*)

Add a rsync job.

Parameters

- **`name`** (*`string`*) – backup label
- **`source`** (*`string`*) – backup from...

- **destination** (*string*) – backup to...
- **period** (*float*) – min time (hours) between backups
- **history** – Activate (True) or desactivate (False) snapshots

or perform a simple copy (None). :type history: bool or None :param duration: How many days snapshotss are kept :type duration: int :param keep: How many snapshots are (at least) kept :type keep: int :param filter: filters :type filter: tuple

Raises ValueError – if destination if not set

Note: Filter syntax is the same of rsync. See "FILTER RULES" section in the rsync man page.

history: if set to True or False, the date is written in the path. This is a feature. It leaves you the possibility to switch on/off the history. If you don't want the date in the path, set the value to None.

---

run()
Run all jobs

set_destination(*destination*, *guid=(None*, *None)*)
Set the destination of the backup if uid or gid are None, files owner are not changed

Parameters

- **destination** (*string*) – destination path
- **guid** (*tuple*) – (uid, gid) for destination

set_log_level(*level='INFO'*)
Set the logger level (INFO, CRITICAL, DEBUG, ERROR, FATAL)

Parameters **level** (*string*) – Loger level

Vitalus.history — functions to determine files to delete

history.**older**(*file_list*, *days=5*)
Return older files than "days"

Parameters

- **file_list** – list of files named in the format "%Y-%m-%d_%Hh%Mm%Ss"
- **days** – files older than this value are old

Returns a sorted list of old files

history.**older_keepmin**(*file_list*, *days=5*, *keep=10*)
Return older files in a list but keep a minium amount of files

Parameters

- **file_list** – list of files named in the format "%Y-%m-%d_%Hh%Mm%Ss"
- **days** – files older than this value are old
- **keep** – keep at least this number of files

Returns a sorted list of old files

Vitalus.rsyncjob —

class rsyncjob.**RsyncJob**(*log_dir*, *destination*, *name*, *source*, *period*, *snapshot*, *duration*, *keep*, *force*, *guid*, *filter*)

Class containing a rsync job

Parameters

- **log_dir** (*string*) – Log directory path
- **destination** (*string*) – Destination path
- **name** (*string*) – Job name
- **source** (*string*) – Source path
- **period** (*float*) – Min duration between two backups (in seconds)
- **snapshot** (*bool or None*) – Activate (True) or desactivate (False) snapshots or simple (None) copy
- **duration** (*int*) – How many days snapshots are kept
- **keep** (*int*) – How many snapshots are (at least) kept
- **force** (*bool*) – overide the timebase check, no min. duration.
- **guid** (*tuple*) – (uid, gid) for destination
- **filter** (*list*) – Rsync filters

Note: Source and destination path can be either real path or a ssh login joined to the path by a : character.

if uid or gid are None, files owner are not changed

---

run(*uid=None*, *gid=None*)

Run the job.

Vitalus.job —

class job.**Job**(*log_dir*, *destination*, *name*, *source*, *period*)

Class containing a job

Parameters

- **log_dir** (*string*) – Log directory path
- **destination** (*string*) – Destination path
- **name** (*string*) – Job name
- **source** (*string*) – Source path
- **period** (*float*) – Min duration between two backups (in seconds)

Note: —

---

run(*uid=None*, *gid=None*)

Run the job.

exception job.**TARGETError**(*message=''*)
Exception for target validity

Parameters **message** (*string*) – Message

class job.**Target**(*target*)
A target is a source or a destination. It can be a local directory or a remote one (SSH)

Parameters **target** (*string*) – a target

check_availability()
Check if the target is available For SSH host, it means it's reachable

Raises TARGETError – if not available

is_local()
Check if the target is a directory

Returns bool – True if it is a directory

is_ssh()
Check if the target is a 'SSH' host

Returns bool – True if it is a SSH host

Vitalus.utils —

utils.**compress**(*path*)
Compress the directory

utils.**get_folder_size**(*path*)
Get the size of the content in path

utils.**get_last_file**(*file_list*)
Return the more recent file in a list (in the format "%Y-%m-%d_%Hh%Mm%Ss")

Parameters **file_list** – list of files

Returns filename

utils.**get_older_files**(*file_list*, *days=5*, *keep=10*)
Deprecated. Use Vitalus.history.older_keepmin()

utils.**r_chmod**(*path*, *mode*)
Equivalent to chmod -R mod path

Parameters

- **path** – path
- **mode** – mode

utils.**r_chown**(*path*, *uid*, *gid*)
Equivalent to chown -R uid:gid path

Parameters

- **path** – path
- **uid** – user ID
- **gid** – group ID

CHAPTER 6

Philosophy

- I want to centralize my backup in a unique script to achieve many tasks.
- I want to backup my desktop on external disks.
- I want to backup external disks to other external disks.
- I want to backup my /home/user on hosts accessible via ssh to a local disk.
- I want to backup my destop to hosts accessible via ssh.
- I want to keep increment if I need it.
- I want to adjust the frequency of copies for each task. The script starts much more frequently.
- I want readable log files telling me if everything goes fine.
- ...

CHAPTER 7

Functionalities

This is just another way to express the philosophy :)

- Manage different tasks
- rsync from and to local disks
- rsync from SSH to local disk
- rsync from local to SSH almost supported
- Check disk space (local disks)
- Keep track of time evolution (increments done with hard links), not possible via SSH for the moment.
- Old increments deleted (keeping a minimal amount of increments)
- Rotated logs (general + one per task)

CHAPTER 8

How to install?

See *How to install?*.

CHAPTER 9

How to setup?

See *Example*.

In my use case, I have a cron job running every hour. IMHO, this is quite atomic. Then, the script decides which task has to be done.

About ssh

Keys must be configured with an empty passphrase. Add in your ~/.ssh/config, something like

> **Host sciunto.org** IdentityFile ~/.ssh/key-empty-passphrase

Source or destination must have the format: login@server:path

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index