dothebackup Documentation

Release 2.1.1

Marvin Steadfast

Jan 17, 2018

Contents

A small tool to run backups in different ways. Its pluggable.

Contents:

CHAPTER 1

Guides

1.1 Installation

There are several ways of installing and using dothebackup.

1.1.1 pex

You can download a pex-File for every release from the release page on GitHub. Read more about pex. Its a zipped up virtual environment, with all the dependencies contained, that you can run with just python installed.

1. Be sure you have Python >= 3.4 installed
2. Download pex file from release
3. Move it to a place in your `$PATH`. For examle `/usr/local/bin`
4. If you want you can renamse `dothebackup.pex` to `dothebackup`. This makes everything a little more neat

1.1.2 pip

Its also possible to use it with a normal pip command.

On Ubuntu for example:

1. `sudo apt-get install python3-pip`
2. `sudo pip3 install dothebackup`

1.1.3 pipenv

I use pipenv to develop. You can use it too to create and manage a virtualenv to run dothebackup in it.

1. Install `pipenv`

2. `git clone https://github.com/xsteadfastx/DoTheBackup.git`
3. `cd DoTheBackup`
4. `pipenv install --three`
5. `pipenv run bash -c "pip install -e ."`
6. `pipenv run dothebackup`

1.1.4 virtualenv

You can just pull the repo and install dothebackup in an virtualenv.

On Ubuntu for example:

1. `sudo apt-get install python3-virtualenv`
2. `git clone https://github.com/xsteadfastx/DoTheBackup.git`
3. `cd DoTheBackup`
4. `virtualenv env`
5. `source env/bin/activate`
6. `pip install -e .`

Now you should be able to use dothebackup.

1.2 Usage

```
Usage: dothebackup [OPTIONS] CONFIGFILE

  Commandline interface.

Options:
  -n, --name TEXT       Run a specific job from the config.
  -t, --test            Only prints the created commands that would be used.
  --debug [debug|info]  Debug or verbose messages.
  --version             Show the version and exit.
  --help                Show this message and exit.
```

First you have to create a config file. Its formatted in YAML. The required keys are:

- **logs**: The log configuration.
  - **dir**: The directory where the logfiles will be saved.
  - **keep**: The numbers of individual job logs to keep.
- **backup**: Here you define all the things you want to backup.

Be sure that you have the key `enabled` set to `yes`. Else that part of the config will be ignored on a normal run.

You also can use the option `--test` to get all commands print instead of running them.

There is a way to trigger a job only on specific days. The key `days` is needed for that. Its a list of days like:

```
days:
  - 01
  - 15
```

This would trigger the job on day `01` and `15` in the month. You need to be sure its a two diget number.

There is a way to trigger only one job from the commandline. Run `dothebackup` with the option `--name` and the name of the job. This would run this one job and ignore everything else in the config. You dont even need to set `enabled` to `yes`. Its handy if you do from time to time backups to external harddrives and want to define the way the backup should run.

1.2.1 Example

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  my_documents:
    type: rsync
    enabled: yes
    mode: month
    source: /home/myuser/documents
    destination: /media/backup/documents

  music:
    type: rsync
    enabled: yes
    mode: once
    source: /home/myuser/Music
    destination: /media/backup/Music

  repos:
    type: github
    enabled: yes
    username: xsteadfastx
    destination: /media/backup/repos
    days:
      - 01
      - 15
```

CHAPTER 2

Plugins

2.1 borg

This plugin will perform a borg backup in a specific repo. It will also run a `borg prune`-command to clean up old backups.

2.1.1 Dependencies

- **borg**

2.1.2 Preparation

Create a borg repository local or on a remote machine:

```
borg init -e none /media/backup/my_borg_backup_repo
```

2.1.3 Configuration

A list of required configuration keys:

- **source**: A list of sources to backup
- **destination** A `borg`-repository to backup to

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  my_borg_backup:
```

```
    type: borg
    enabled: yes
    source:
        - /etc
        - /home
        - /srv
    destination: /media/backup/my_borg_backup_repo
    exclude:
        - /home/user/Downloads
    keep:  # Backups to keep
        daily: 7  # last 7 days
        weekly: 4  # last 4 weeks
        monthly: 6  # last 6 months
    check: yes  # runs check command after each run on the repo
```

It's also possible to use a remote destination:

```
destination: ssh://user@remotebox:/media/backup/my_borg_backup_repo
```

2.1.4 API

Borg.

dothebackup.plugs.borg.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippt for this plugin

Return type List[List[str]]

Returns Commands to create the backup

2.2 git

This plugin clones and pulls a git repository to the backup machine. First it will check if a repository is already cloned to `destination` and if not it will clone and do a `git pull` afterwards. On the next runs it will just do a `git pull`.

2.2.1 Dependencies

- **git**

2.2.2 Configuration

A list of required configuration keys:

- **source**: A repository url to clone from.
- **destination**

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  my_repository:
    type: git
    enabled: yes
    source: https://github.com/xsteadfastx/DoTheBackup.git
    destination: /media/backup/repos/dothebackup
```

2.2.3 API

Git.

dothebackup.plugs.git.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

2.3 github

This plugin gets a users public repositories from the github API and clones them if needed. Else it will go through all cloned repositories and do a `git pull`.

2.3.1 Dependencies

- **git**

2.3.2 Configuration

A list of required configuration keys:

- **username**: Needs to be a valid github username
- **destination**

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  mygithubrepos:
    type: github
    enabled: yes
    username: xsteadfastx
    destination: /media/backup/github
```

2.3.3 API

Github.

dothebackup.plugs.github.**get_repos**(*username*)
Create a GET request on the github api to get all repos from a user.

Parameters **username** (str) – github username

Return type List[Dict[str, Union[int, str, Dict[~KT, ~VT]]]]

Returns Full JSON dictionary of user repos from github

dothebackup.plugs.github.**main**(*config*)
Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

2.4 mysql

With this plugin you can backup MySQL-Databases. You also can commit the changes to a git repository.

2.4.1 Dependencies

- **git**
- **mysqldump**

2.4.2 Configuration

A list of required configuration keys:

- **server**: The hostname or ip of the MySQL server.
- **username**: A username that is allowed to login and backup stuff. The recommended way is to create a special backup user. Consider *Create MySQL BACKUPUSER*.
- **password**: Password for the MySQL user.
- **database**: Name of the database to backup.
- **destination**: The folder where the backup is going to be saved to. Its just the name of the folder. The dump itself will be called `<databasename>.sql`.

Its also needed to define a `mode`. Here is a list:

- **once**: This mode will run the `mysqldump` command once and will overwrite a dump that is already there.
- **git**: This mode will commit every change to the dump after the dump got created. This will save every change of it. For a large database with alot of changes this can be a little heavy. But for small database this may be the best solution.

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  mydatabase:
    enabled: yes
    type: mysql
    mode: git
    server: localhost
    username: backupuser
    password: backuppassword
    database: mydatabase
    destination: /media/backup/mydatabase
```

2.4.3 Create MySQL BACKUPUSER

If you want to create a user that can backup all databases:

```
# /usr/bin/mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or g.
Your MySQL connection id is 253404
Server version: 5.1.54-1ubuntu4 (Ubuntu)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or 'h' for help. Type 'c' to clear the current input statement.

mysql> GRANT LOCK TABLES, SELECT ON *.* TO 'BACKUPUSER'@'%' IDENTIFIED BY 'PASSWORD';
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> Bye
```

If you only want the BACKUPUSER to have permissions to one database, you can modify the command:

```
GRANT LOCK TABLES, SELECT ON DATABASE.* TO 'BACKUPUSER'@'%' IDENTIFIED BY 'PASSWORD';
```

Thanks to Ben Cane for that information.

2.4.4 API

MYSQL.

dothebackup.plugs.mysql.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

2.5 rsync

This plugin uses rsync to sync directories local and remote.

2.5.1 Dependencies

- **rsync**
- **ssh**: This is needed for remote syncs.

2.5.2 Configuration

A list of required configuration keys:

- **source**: A directory that should be synced.
- **destination**: The directory the backup should be synced to.

Its also needed to define a `mode`. Here is a list:

- **once**: Syncs one to one. It also deleted files on destination that are not there anymore.
- **week**: Keeps one week. It stores the files in a numbered day directory and uses hardlinks to link to the files that are not changed from the day before.
- **month**: Keeps one month. it stores the files for one month in a day numbered directory and works with hardlinks just like the week mode.

It supports rsync `exclude` and `include` patterns as list.

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  my_documents:
    type: rsync
    enabled: yes
    mode: month
    source: /home/myuser/documents
    destination: /media/backup/documents
    exclude:
      - foo
      - bar
    include:
      - importantdir
```

This would save the source to `/media/backup/documents/27` for example and rotate for one month with day numbers.

It supports ssh for transfering data to remote hosts. For example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  my_documents:
    type: rsync
    enabled: yes
    mode: once
    source: /home/myuser/documents
    destination: ssh://foo@remotehost:/media/backup/documents
```

Be sure that you are allowed to connect and ssh-keys are exchanged.

2.5.3 API

Rsync.

dothebackup.plugs.rsync.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

dothebackup.plugs.rsync.**normalize_path**(*path*)

Returns a normalized path.

Rsync needs a normalized path and special handing if its a ssh path.

Parameters **path** (str) – Path to normalize

Return type str

Returns Normalized path

2.6 slapcat

With this plugin you can backup LDAP-Databases. You can commit the changes to a git repository

2.6.1 Dependencies

- **git**
- **slapcat**

2.6.2 Configuration

A list of required configuration keys:

- **destination**: The folder where the bacup is going to be saved to. Its just the name of the folder. The export itself will be called backup.ldif.

Its also needed to define a mode. Here is a list:

- **once**: This mode will run the `slapcat` command once and will overwrite a export that is already there.
- **git**: This mode will commit every change to the export after the export got created. This will save every change of it.

Here's a example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  myldap:
    enabled: yes
    type: slapcat
    mode: git
    destination: /media/backup/myldap
```

2.6.3 Restore

Example:

```
slapadd -l /media/backup/myldap/backup.ldif
```

2.6.4 API

Slapcat.

dothebackup.plugs.slapcat.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

2.7 tar

This plugin takes a list of sources and puts them into a tarfile.

2.7.1 Dependencies

For simple usage just the normal tar binary is needed. If you want to use some kind of compression, you need to install additional packages.

- **tar**: The main command needed for uncompressed archives.
- **xz-utils**: For XZ compressed tarfiles.
- **lbzip2**: For BZIP2 compressed tarfiles.

2.7.2 Configuration

A list of required configuration keys:

- **source**: A list of sources to be included in the archive.
- **destination**: The destination file. The extension tells the plugin if a compression is wanted and which or none. Example: `foo.tar`, `foo.tar.gz`, `foo.tar.bzip2`, `foo.tar.xz`

Here's an example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  foo:
    type: tar
    enabled: yes
    source:
      - /home
      - /src
    destination: /media/backup/archives/foo.tar.gz
```

2.7.3 API

Tar.

dothebackup.plugs.tar.**main**(*config*)

Command builder.

Parameters **config** (Dict[str, Union[str, List[str], bool]]) – config snippet for this plugin

Return type List[List[str]]

Returns Commands to create the backup

CHAPTER 3

API

3.1 runner

Runner.

dothebackup.runner.**builder**(*config*, *name*)
Builds a dict of commands.

Parameters

- **config** (Dict[str, Dict[str, Dict[~KT, ~VT]]]) – YAML config
- **name** (Optional[str]) – Name of a specific job to run

Return type Dict[str, List[List[str]]]

Returns A dict with all commands needed commands

dothebackup.runner.**check_config_keys**(*config*, *key_list*)
Aborts if keys are not set in config.

Parameters

- **config** (*dict*) – Config
- **key_list** (*list*) – List of used keys

Return type None

dothebackup.runner.**check_if_already_running**()
Aborts if other DoTheBackup process is running.

Return type None

dothebackup.runner.**check_plugin**(*name*)
Aborts and throw an error if plugin is not there as defined as type in config.

Parameters **name** (*str*) – Name of plugin that is defined in the config

Return type None

dothebackup.runner.**get_started**(*configfile*, *name*, *test*)
The entrypoint for the UI.

This is used to get everything started up. It will read the config, check the keys, build the command dictionary and run them.

Parameters

- **configfile** (IO[AnyStr]) – The config file
- **name** (str) – A name of a specific job
- **test** (bool) – Switch for only printing the commands

Return type None

dothebackup.runner.**parse_config**(*configfile*)
Read config file.

Parameters **configfile** (*_io.TextIOWrapper*) – YAML config file

Returns loaded configfile

Return type dict

dothebackup.runner.**print_commands**(*commands*)
Prints the commands that would be used.

Parameters **commands** (Dict[str, List[List[str]]]) – Command dictionary

Return type None

dothebackup.runner.**run_commands**(*commands*, *test*, *log_dir*, *log_keep*)
Running the commands.

The actual runner. It will take the commands dictionary and run it one after another. There is also a test key. With this enabled it will only print the commands it would run.

Parameters

- **commands** (Dict[str, List[List[str]]]) – Commands dictionary
- **test** (bool) – If test the commands only will be printed
- **log_dir** (str) – Dictionary for logfiles
- **log_keep** (int) – How many logs to keep from one job

Return type None

3.2 logger

Logger.

class dothebackup.logger.**Logger**(*log_dir*, *name*, *keep*)
A logger for handling stdout logging.

Parameters

- **log_dir** (str) – Directory to store logfiles in
- **name** (str) – Name of the backup job
- **keep** (int) – Number of logs to keep

create_log_dir()
Create logdir if its not there.

Return type None

logfile()
A logfile handler.

This is used in a context to write to a logfile:

```
with logger.logfile() as logfile:
    logfile.write('foobar')
```

Yields Opened logfile

Return type Iterator[IO[str]]

rotate()
A logfile rotator.

This function moves old logfiles around.

Example:

```
foo.log
```

gets:

```
foo.log.0001
```

and so on.

Return type None

3.3 plugins

Plugins.

dothebackup.plugins.**load_plugins**()
Load plugins from plugin directory.

This function reads the plugs directory and loads all plugins.

Return type Dict[str, Callable]

dothebackup.plugins.**required_executables**(*dep_list*)
Decorator to check required executables.

Parameters **dep_list** (List[str]) – Dependency list

Return type Callable

Returns Decorated function

dothebackup.plugins.**required_keys**(*key_list*)
Decorator to check against key list.

Parameters **key_list** (List[str]) – List of keys that needs to be in the config

Return type Callable

Returns Decorated function

3.4 utils

Utils.

dothebackup.utils.**absolutenormpath**(*path*)
Returns a absolute normalized path.

Parameters **path** (str) – Path

Return type str

Returns Absolute path

dothebackup.utils.**git_cloned_yet**(*path*)
Returns if path contains a git repository.

Parameters **path** (str) – A path

Return type bool

Returns If path contains a git repositoy

dothebackup.utils.**git_something_to_commit**(*path*)
Returns if path has something to commit.

Parameters **path** (str) – A path

Return type bool

Returns If path has something to commit

dothebackup.utils.**pidfile**()
Contextmanager to create a Pidfile for the DoTheBackup process and removes it afterwards.

Return type Iterator[None]

dothebackup.utils.**return_code**(*exitcodes*)
Get overalls exitcode out of a list of exitcodes.

Parameters **exitcodes** (List[int]) – List of exitcodes

Return type int

Returns Return exit code 1 if something different than 0 is in the list

dothebackup.utils.**today**()
Returns todays day string.

Return type str

Returns Today day string

CHAPTER 4

Changelog

4.1 v2.1.0 (2017-11-14)

- Added *BORG_HOSTNAME_IS_UNIQUE=yes* to borg. [Marvin Steadfast]

4.2 v2.0.0 (2017-11-09)

4.2.1 Fix

- "enabled" missing in config throws exit code 1. [Marvin Steadfast]

4.2.2 Other

- Travis deploy only if $TOXENV is pex. [Marvin Steadfast]
- Creates a pidfile. [Marvin Steadfast]

  This prevents *dothebackup* from running more then once parallel.
- Moving package to src directory. [Marvin Steadfast]
- Using pipenv for development environment and testing. [Marvin Steadfast]
- Python 3.3 no longer supported. [Marvin Steadfast]
- Raising coverage. [Marvin Steadfast]
- Testing debug levels of the UI. [Marvin Steadfast]
- Added test for ui. [Marvin Steadfast]

4.3 v1.1.1 (2017-06-16)

4.3.1 Fix

- Add typing as dependency. [Marvin Steadfast]

4.4 v1.1.0 (2017-06-16)

- Print stdout from commands in debug more. [Marvin Steadfast]
- Added borg check command. [Marvin Steadfast]
- Dothebackup return itself an error exit code. [Marvin Steadfast]

  If one job failed it will not just write this to the job logs it also will exit with a different exit code.
- Fixing import order in tests. [Marvin Steadfast]
- Added typing and plugin for borg backup. [Marvin Steadfast]
- Switched from `arrow` to `pendulum` [Marvin Steadfast]

4.5 v1.0.2 (2017-01-04)

- Replace invalid characters in logging. [Philipp Weißmann]
- Working on Jenkinsfile. [Marvin Steadfast]
- Added Jenkinsfile. [Marvin Steadfast]

4.6 v1.0.1 (2016-08-18)

4.6.1 Fix

- Logs finally rotate. [Marvin Steadfast]

  forgot to call the rotate function in the runner and fixed a bug caused by the log numbering order. logs are now numbered like:

```
foo.log
foo.log.0001
foo.log.0002
```

4.7 v1.0.0 (2016-08-18)

- Testing and no installations for old python versions. [Marvin Steadfast]

  added more python versions for local tox testing. installations on python versions < python 3.3 will be aborted in setup.py.

- Logger module. [Marvin Steadfast]

  added logger module for better logging of the jobs stdout. it keeps track on rotating old logs and writing to the right logfile.

  the logging part of the config changed. a example:

```
logs:
  dir: /var/log/dothebackup
  keep: 10
backup:
  ...
  ...
  ...
```

  keep defines how many log files to keep.

4.8 v0.2.1 (2016-08-15)

- Adds log for startup and finishing dothebackup. [Philipp Weißmann]
- New changelog. [Marvin Steadfast]

4.9 v0.2.0 (2016-08-08)

- More docs and logging. [Marvin Steadfast]
- Better docs and more debug messages. [Marvin Steadfast]
- The runner has debugging messages now. [Marvin Steadfast]
- Test for load_plugins. [Marvin Steadfast]
- Plugin loader not in __init__ and added logging. [Marvin Steadfast]

4.10 v0.1.9 (2016-07-29)

- 0.1.9 release. [Marvin Steadfast]
- Adds newline seperators to log file. [Philipp Weißmann]

4.11 v0.1.8 (2016-07-28)

- 0.1.8 release. [Marvin Steadfast]
- Fixes pep violations (line length) [Philipp Weißmann]
- Adds finishing date and total runtime to log. [Philipp Weißmann]
- Fixed typo. [Marvin Steadfast]
- Fixes typo in Readme. [Philipp Weißmann]

4.12 v0.1.7 (2016-04-13)

- 0.1.7 release. [Marvin Steadfast]
  - Fixes a bug where git something to commit detection fails if git is not initialised
- Added forgotten enabled in examples. [Marvin Steadfast]
- Removed support for python 3.2. [Marvin Steadfast]

4.13 v0.1.6 (2016-04-12)

- Fixed typo in docs. [Marvin Steadfast]
- 0.1.6 release. [Marvin Steadfast]
  - Added slapcat plugin.
- Fixed doc. [Marvin Steadfast]

4.14 v0.1.5 (2015-11-12)

- 0.1.5 release. [Marvin Steadfast]
  - Added mysql plugin.
  - Added some git tools.
- Fixing travis python 3.5 job. [Marvin Steadfast]

4.15 v0.1.4 (2015-11-02)

- 0.1.4 release. [Marvin Steadfast]
  - Restructured code. Splitted the ui and runner parts.
  - Testing also against Python versions 3.2, 3.3 and 3.5.
  - Added `name` option to command line for running only a specific job. Even if its not enabled.
  - The config file takes a `days` list for a job. Before running it will check the day its running and if its in the list. Else it will skip it.
- Added –test to the docs. [Marvin Steadfast]
- Removed stuff from docs. [Marvin Steadfast]
- Added test_tar fixture. [Marvin Steadfast]

4.16 v0.1.3 (2015-10-22)

- Added tar plugin, Python 3 only, docs. [Marvin Steadfast]

  Added a plugin that creates tar archives from a list of source directories. Dropped Python 2 support because of the UnicodeDecodeErrors i dont want to deal with no more. Python 3 should make this more futureproof and robust. Also added docs.

- Fix README. [Marvin Steadfast]

4.17 v0.1.2 (2015-10-20)

- Added github plugin. [Marvin Steadfast]

  Its a plugin to get a users public repositories through the GitHub Api, clone them (if not done before) and pulls the changes on every run.

- Fix readme tabs. [Marvin Steadfast]

4.18 v0.1.1 (2015-10-07)

- Added git plugin. [Marvin Steadfast]

  A simple git plugin to clone a git repo to a destination and run a git pull afterwards.

- Using click.File for reading configfile. [Marvin Steadfast]
- Removed old config dist file. [Marvin Steadfast]

4.19 v0.1 (2015-10-06)

- Added pypi badge to readme. [Marvin Steadfast]
- Added tests for the ui. [Marvin Steadfast]
- Added tests for exclude key. [Marvin Steadfast]
- Moved to codecov. [Marvin Steadfast]
- Removed support for python 3.2. [Marvin Steadfast]
- Rebased everything. [Marvin Steadfast]

  Its now installable through pip. Also it uses plugins now. All you need is a plugin that returns a list if commands that get executed. Right now only the rsync plugin is there.

- Adds option to keep backups for a week (additional to a month) [Philipp Weißmann]
- Still tweaking tox.ini to run also on jenkins smooth. [Marvin Steadfast]
- Ignore coverage.xml. [Marvin Steadfast]
- Tests are more verbose now to make jenkins happy. [Marvin Steadfast]
- Changed TOXENV. [Marvin Steadfast]
- Forgot to readd coveralls command. [Marvin Steadfast]

- Test against more python versions. [Marvin Steadfast]
- Moved coverage from .travis.yml to tox.ini to make it simpler and cleaner. [Marvin Steadfast]
- Moved test to tests. [Marvin Steadfast]
- Moved from nose to py.test. [Marvin Steadfast]
- Fixed some test and did some refactoring of the tests. [Marvin Steadfast]
- Fixed readme layout. [Marvin Steadfast]
- Tests rsync commands. [Marvin Steadfast]
- Subprocess arguments gets tested. [Marvin Steadfast]
- Install rsync for travis testing. [Marvin Steadfast]
- Added .coveragerc. [Marvin Steadfast]
- Better tests through tox and travis. [Marvin Steadfast]
- Some pep8 fix up. [Marvin Steadfast]
- Fixed a bug with the paths when running the tests from a different location. [Marvin Steadfast]
- Added first tests. [Marvin Steadfast]
- Almost rewrote everything and added git_mysql type. [Marvin Steadfast]
- Added ssh support. [Marvin Steadfast]
- Added cron shell script. [Marvin Steadfast]
- Complete rewrite. [Marvin Steadfast]
- Fixd readme. [Marvin Steadfast]
- First working version. [Marvin Steadfast]
- Initial commit. [xsteadfastx]

Python Module Index

d

Index