PyVolume Documentation

Release 0.1.2

Raghavendra Prabhu

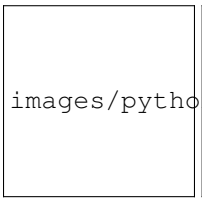
Jul 24, 2022

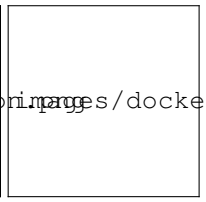
Contents

Contents:

========== PyVolume ==========

|Updates| |Python 3|





Python Docker Volume driver.

Supports pluggable implementations, currently there are three written.

Implements: * '/Plugin.Activate' * '/VolumeDriver.Create' * '/VolumeDriver.Remove' * '/VolumeDriver.List' * '/VolumeDriver.Path' * '/VolumeDriver.Mount' * '/VolumeDriver.Unmount' * '/VolumeDriver.Get' * '/VolumeDriver.Capabilities'

for Docker Volume.

and * '/' * '/shutdown'

for management.

The volume manager (common to all drivers) uses Flask for routing and handles multiple invocations of Mount and Unmount for same volume as per docker specifications. It also passes options passed through API to the drivers. Cleanup is also handled on shutdown.

CHAPTER 1

Current Implementations

- Ephemeral FileSystem
- SSHFS FileSystem
- Zookeeper FileSystem
  - This uses docker-zkfuse for using zkfuse and mounts zkfuse from container to host through shared mounting of volume from host to container.

1.1 Installing

1. Install the package.

```
pip install -r requirements.txt --user
python2 setup.py install --prefix=/usr/local
```

After this pyvolume should be available as /usr/local/bin/pyvolume.

2. Copy the pyvolume.json to /etc/docker/plugins/

CHAPTER 2

Dependencies:

CHAPTER 3

Installation

1. Python 2.7 and python related dependencies - pip, virtualenv etc.
2. sshfs for SSHFileSystem (default).
1. ssh-add and sshfs.
2. curl
3. util-linux (for mount etc.)
4. Python-related tools such as virtualenv.

3.1 Usage

3.1.1 Permissions

- Make sure you have write permissions for base mount directory which is /mnt by default.
  - If not, make sure to chown as the user you run pyvolume as.
- For ZookeeperFileSystem, make sure the pyvolume's user can do docker run without sudo.
  - If not, add the pyvolume's user to docker group.

3.1.2 For SSH FileSystem

Arguments for docker volume create:

- remote_path: as host:directory (Required)
- ssh_config: path to ssh config directory if it is not default.
- sshfs_options: any options to pass to sshfs.

After Installing,

1. Make sure the ssh keys are available through the ssh-agent.

```
ssh-add -l
```

2. Start the pyvolume server.

```
$ /usr/local/bin/pyvolume
INFO:werkzeug: * Running on http://0.0.0.0:1331/ (Press CTRL+C to quit)
```

3. Create a docker volume.

```
docker volume create -d pyvolume --name myvolume2 -o 'remote_path=server:/home/user' -
↪o 'ssh_config=/home/rprabhu/.ssh.bkp/config.server'
```

4. Run docker as usual, providing the newly created volume name.

```
docker run -it -v myvolume2:/data  busybox:latest sh
```

5. PROFIT!

3.1.3 For ZooKeeper FileSystem

Arguments for docker volume create:

- zookeeper_string - host:port (or a list of tuples) to running zookeeper instance. (Required)
- docker_opt - any options to pass to docker.

1. Start the pyvolume server.

```
$ /usr/local/bin/pyvolume -t zookeeper
INFO:werkzeug: * Running on http://0.0.0.0:1331/ (Press CTRL+C to quit)
```

3. Create a docker volume.

```
docker volume create -d pyvolume --name zoo -o 'zookeeper_string=0.0.0.0:2181' -o
↪'docker_opt=--net=host'
```

This assumes that you have a local zookeeper running on host at 0.0.0.0:2181. Since it is running on host, you need '–net=host' as well.

Otherwise, if you have zookeeper running on *host* :port, following will do:

```
docker volume create -d pyvolume --name zoo -o "zookeeper_string=$host:$port"
```

4. Run docker as usual, providing the newly created volume name.

```
docker run -it -v zoo:/data  busybox:latest sh
```

and you can access the zookeeper znodes through /data/ after this.

3.2 Local Installation and Running

1. Look above for dependencies. 2.

```
$ make devenv
$ source devenv/bin/activate

$ ./devenv/bin/pyvolume --help

usage: pyvolume [-h] [-t {sshfs,ephemeral}] [-H HOST] [-p PORT] [-m BASE]

Arguments to volume router

optional arguments:
-h, --help            show this help message and exit
-t {sshfs,ephemeral}, --driver {sshfs,ephemeral}
                        Type of driver to use
-H HOST, --host HOST  Host to listen on
-p PORT, --port PORT  Port to listen on
-m BASE, --base BASE  Base directory to mount over

$ ./devenv/bin/pyvolume
INFO:werkzeug: * Running on http://0.0.0.0:1331/ (Press CTRL+C to quit)
```

3.3 Testing

CHAPTER 4

Integration test.

1. Set the required environment variables.

```
a. export SSH_CONFIG=/home/rprabhu/.ssh.bkp/config.server
b. export REMOTE_PATH='server:/home/user'
c. make itest
```

2. itest log - https://gist.github.com/ronin13/83d99b801202e63f07523c1c5b2be450

CHAPTER 5

Unit test.

1. make test

```
make test
tox2 -e py27
GLOB sdist-make: /home/rprabhu/repo/pyvolume/setup.py
py27 create: /home/rprabhu/repo/pyvolume/.tox/py27
py27 installdeps: -r/home/rprabhu/repo/pyvolume/requirements_dev.txt
py27 inst: /home/rprabhu/repo/pyvolume/.tox/dist/pyvolume-0.1.0.zip
py27 installed: You are using pip version 8.1.2, however version 9.0.0 is available.,
 ↪You should consider upgrading via the 'pip install --upgrade pip' command.,
 ↪alabaster==0.7.9,argh==0.26.2,Babel==2.3.4,bumpversion==0.5.3,cffi==1.8.3,click==6.
 ↪6,coverage==4.1,cryptography==1.4,docutils==0.12,enum34==1.1.6,flake8==2.6.0,
 ↪Flask==0.11.1,idna==2.1,imagesize==0.7.1,ipaddress==1.0.17,itsdangerous==0.24,
 ↪Jinja2==2.8,MarkupSafe==0.23,mccabe==0.5.2,pathtools==0.1.2,pluggy==0.3.1,
 ↪plumbum==1.6.2,py==1.4.31,pyasn1==0.1.9,pycodestyle==2.0.0,pycparser==2.17,
 ↪pyflakes==1.2.3,Pygments==2.1.3,pytest==2.9.2,pytz==2016.7,pyvolume==0.1.0,
 ↪PyYAML==3.11,six==1.10.0,snowballstemmer==1.2.1,Sphinx==1.4.8,tox==2.3.1,
 ↪virtualenv==15.0.3,watchdog==0.8.3,Werkzeug==0.11.11
py27 runtests: PYTHONHASHSEED='2628874551'
py27 runtests: commands[0] | pip install -U pip
Collecting pip
Using cached pip-9.0.0-py2.py3-none-any.whl
Installing collected packages: pip
Found existing installation: pip 8.1.2
    Uninstalling pip-8.1.2:
    Successfully uninstalled pip-8.1.2
Successfully installed pip-9.0.0
py27 runtests: commands[1] | py.test
============================================================================================================
 ↪test session starts␣
 ↪============================================================================================================
platform linux2 -- Python 2.7.12, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/rprabhu/repo/pyvolume, inifile: tox.ini
collected 3 items
```

(continues on next page)

(continued from previous page)

```
test_pyvolume_sshfs.py ...

=============================================================================================================
↪3 passed in 0.09 seconds
↪=============================================================================================================
_____________________________________________________________________________________________________
↪_____________________ summary _____________________________________________________
↪_____________________________________________________
py27: commands succeeded
congratulations :)
```

5.1 License

- Free software: MIT license

5.2 Credits

This package was created with Cookiecutter_ and the 'audreyr/cookiecutter-pypackage' project template.

- Cookiecutter: https://github.com/audreyr/cookiecutter
- audreyr/cookiecutter-pypackage: https://github.com/audreyr/cookiecutter-pypackage

CHAPTER 6

Installation

6.1 Stable release

To install PyVolume, run this command in your terminal:

```
$ pip install pyvolume
```

This is the preferred method to install PyVolume, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

6.2 From sources

The sources for PyVolume can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/ronin13/pyvolume
```

Or download the tarball:

```
$ curl  -OL https://github.com/ronin13/pyvolume/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 7

Usage

To use PyVolume in a project:

```
import pyvolume
```

CHAPTER 8

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

8.1 Types of Contributions

8.1.1 Report Bugs

Report bugs at https://github.com/ronin13/pyvolume/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

8.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

8.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

8.1.4 Write Documentation

PyVolume could always use more documentation, whether as part of the official PyVolume docs, in docstrings, or even on the web in blog posts, articles, and such.

8.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ronin13/pyvolume/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

8.2 Get Started!

[Check README.md for details on testing and development.] Ready to contribute? Here's how to set up *pyvolume* for local development.

1. Fork the *pyvolume* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyvolume.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pyvolume
$ cd pyvolume/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass black and the tests, including testing other Python versions with tox:

```
$ black pyvolume tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

8.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.md.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/ronin13/pyvolume/pull_requests and make sure that the tests pass for all supported Python versions.

8.4 Tips

To run a subset of tests:

```
$ py.test tests.test_pyvolume
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search