pypiserver Documentation

Release latest

March 02, 2015

Contents

Authors Ralf Schmitt <ralf@systemexit.de>, Kostis Anagnostopoulos <ankostis@gmail.com>

Version 1.1.7

Date 2015-02-28

Source https://github.com/pypiserver/pypiserver

Download https://pypi.python.org/pypi/pypiserver#downloads

Table of Contents

pypiserver is a minimal PyPI compatible server. It can be used to upload and serve a set of packages, wheels and eggs to *pip* or *easy_install*.

CHAPTER 1

Installation and Usage/Quickstart

pypiserver will work with python 2.5 –> 2.7 and 3.2 –> 3.4. Python 3.0 and 3.1 may also work, but pypiserver is not being tested with these versions.

Run the following commands to get your PyPI server up and running:

```
## Installation.
pip install pypiserver
mkdir ~/packages          ## Copy packages/wheels/eggs to this directory.

## Start server.
pypi-server -p 8080 ~/packages &

## Install hosted packages.
pip install  --extra-index-url http://localhost:8080/simple/ ...
```

You can even install the latest *pypiserver* directly from github with this command, assuming you have *git* installed on your *PATH*:

```
pip install git+git://github.com/pypiserver/pypiserver.git
```

Note: The above commands do work on an unix like operating system with a posix shell. If you're using windows, you'll have to run their 'windows counterparts'. The same is true for the rest of this documentation.

CHAPTER 2

Alternative Installation as standalone script

The git repository contains a `pypi-server-standalone.py` script, which is a single python file ready to be executed without any other dependencies.

Run the following commands to download the script with *wget*:

```
wget https://raw.github.com/pypiserver/pypiserver/standalone/pypi-server-standalone.py
chmod +x pypi-server-standalone.py
```

or with *curl*:

```
curl -O https://raw.github.com/pypiserver/pypiserver/standalone/pypi-server-standalone.py
chmod +x pypi-server-standalone.py
```

The server can then be started with:

```
./pypi-server-standalone.py
```

Feel free to rename the script and move it into your $PATH.

CHAPTER 3

Running on heroku/dotcloud

https://github.com/dexterous/pypiserver-on-the-cloud contains instructions on how to run pypiserver on one of the supported cloud service providers.

CHAPTER 4

Detailed Usage

pypi-server -h will print a detailed usage message:

```
pypi-server [OPTIONS] [PACKAGES_DIRECTORY...]
  start PyPI compatible package server serving packages from
  PACKAGES_DIRECTORY. If PACKAGES_DIRECTORY is not given on the
  command line, it uses the default ~/packages.  pypiserver scans this
  directory recursively for packages. It skips packages and
  directories starting with a dot. Multiple package directories can be
  specified.

pypi-server understands the following options:

  -p, --port PORT
    listen on port PORT (default: 8080)

  -i, --interface INTERFACE
    listen on interface INTERFACE (default: 0.0.0.0, any interface)

  -a, --authenticate (UPDATE|download|list), ...
    comma-separated list of (case-insensitive) actions to authenticate
    (requires giving also the -P option). For example to password-protect
    package uploads & downloads while leaving listings public, give:
      -a update,download.
    If unspecified, only 'update' is password-protected.

  -P, --passwords PASSWORD_FILE
    use apache htpasswd file PASSWORD_FILE to set usernames & passwords
    used for authentication of certain actions (see -a option).

  --disable-fallback
    disable redirect to real PyPI index for packages not found in the
    local index

  --fallback-url FALLBACK_URL
    for packages not found in the local index, this URL will be used to
    redirect to (default: http://pypi.python.org/simple)

  --server METHOD
    use METHOD to run the server. Valid values include paste,
    cherrypy, twisted, gunicorn, gevent, wsgiref, auto. The
    default is to use "auto" which chooses one of paste, cherrypy,
    twisted or wsgiref.
```

```
  -r, --root PACKAGES_DIRECTORY
    [deprecated] serve packages from PACKAGES_DIRECTORY

  -o, --overwrite
    allow overwriting existing package files

  --welcome HTML_FILE
    uses the ASCII contents of HTML_FILE as welcome message response.

  -v
    enable INFO logging;  repeate for more verbosity.

  --log-conf <FILE>
    read logging configuration from FILE.
    By default, configuration is read from 'log.conf' if found in server's dir.

  --log-file <FILE>
    write logging info into this FILE.

  --log-frmt <FILE>
    the logging format-string.  (see 'logging.LogRecord' class from standard python library)
    [Default: %(asctime)s|%(levelname)s|%(thread)d|%(message)s]

  --log-req-frmt FORMAT
    a format-string selecting Http-Request properties to log; set to  '%s' to see them all.
    [Default: %(bottle.request)s]

  --log-res-frmt FORMAT
    a format-string selecting Http-Response properties to log; set to  '%s' to see them all.
    [Default: %(status)s]

  --log-err-frmt FORMAT
    a format-string selecting Http-Error properties to log; set to  '%s' to see them all.
    [Default: %(body)s: %(exception)s \n%(traceback)s]

pypi-server -h
pypi-server --help
  show this help message

pypi-server --version
  show pypi-server's version

pypi-server -U [OPTIONS] [PACKAGES_DIRECTORY...]
  update packages in PACKAGES_DIRECTORY. This command searches
  pypi.python.org for updates and shows a pip command line which
  updates the package.

The following additional options can be specified with -U:

  -x
    execute the pip commands instead of only showing them

  -d DOWNLOAD_DIRECTORY
    download package updates to this directory. The default is to use
    the directory which contains the latest version of the package to
    be updated.

  -u
```

```
    allow updating to unstable version (alpha, beta, rc, dev versions)

Visit https://pypi.python.org/pypi/pypiserver for more information.
```

CHAPTER 5

Configuring pip/easy_install

Always specifying the the pypi url on the command line is a bit cumbersome. Since pypi-server redirects pip/easy_install to the pypi.python.org index if it doesn't have a requested package, it's a good idea to configure them to always use your local pypi index.

5.1 *pip*

For *pip* this can be done by setting the environment variable *PIP_EXTRA_INDEX_URL* in your *.bashrc*/*.profile*/*.zshrc*:

```
export PIP_EXTRA_INDEX_URL=http://localhost:8080/simple/
```

or by adding the following lines to ~/.pip/pip.conf:

```
[global]
extra-index-url = http://localhost:8080/simple/
```

Note: If you have installed *pypi-server* on a remote url without *https* you wil receive an "untrusted" warning from *pip*, urging you to append the '–trusted-host' option. You can include this option permanently in your configuration-files or environment variables.

5.2 *easy_install*

For *easy_install* it can be configured with the following setting in *~/.pydistutils.cfg*:

```
[easy_install]
index_url = http://localhost:8080/simple/
```

CHAPTER 6

Uploads via *setup.py* upload

Uploading packages via `python setup.py upload` is also possible. First make sure you have the *passlib* module installed:

```
pip install passlib
```

Then create a apache *htpassword* file with:

```
htpasswd -sc .htaccess myusername
```

You'll be prompted for a password. You'll need to restart the server with the *-P* option:

```
pypi-server -p 8080 -P /path/to/.htaccess /path/to/private_pypi_folder/
```

Edit or create a *~/.pypirc* file with the following content:

```
[distutils]
index-servers =
  pypi
  internal

[pypi]
username:pypiusername
password:pypipasswd

[internal]
repository: http://127.0.0.1:8080
username: myusername
password: mypasswd
```

Uploading then works by running:

```
python setup.py sdist upload -r internal
```

CHAPTER 7

Managing the package directory

The *pypi-server* command has the *-U* option that searches for updates of available packages. It scans the package directory for available packages and searches on pypi.python.org for updates. Without further options `pypi-server -U` will just print a list of commands which must be run in order to get the latest version of each package. Output looks like:

```
 $ ./pypi-server -U
checking 106 packages for newer version

.........u.e..........e..u.............
.....e.............................e...
.........................

no releases found on pypi for PyXML, Pymacs, mercurial, setuptools

# update raven from 1.4.3 to 1.4.4
pip -q install --no-deps  --extra-index-url http://pypi.python.org/simple -d /home/ralf/packages/mirr

# update greenlet from 0.3.3 to 0.3.4
pip -q install --no-deps  --extra-index-url http://pypi.python.org/simple -d /home/ralf/packages/mirr
```

It first prints for each package a single character after checking the available versions on pypi. A dot(.) means the package is up-to-date, *u* means the package can be updated and *e* means the list of releases on pypi is empty. After that it shows a *pip* command line which can be used to update a one package. Either copy and paste that or run `pypi-server -Ux` in order to really execute those commands. You need to have *pip* installed for that to work however.

Specifying an additional *-u* option will also allow alpha, beta and release candidates to be downloaded. Without this option these releases won't be considered.

CHAPTER 8

Optional dependencies

pypiserver relies on the *passlib* module for parsing apache htpasswd files. You need to install it, when using the *-P*, *–passwords* option. The following command will do that:

```
pip install passlib
```

CHAPTER 9

Using a different WSGI server

- *pypiserver* ships with it's own copy of bottle. It's possible to use bottle with different WSGI servers.
- *pypiserver* chooses any of the following *paste*, *cherrypy*, *twisted*, *wsgiref* (part of python) if available.
- If none of the above servers matches your needs, pypiserver also exposes an API to get the internal WSGI app, which you can then run under any WSGI server you like. *pypiserver.app* has the following interface:

  ```
  def app(root=None,
          redirect_to_fallback=True,
          fallback_url="http://pypi.python.org/simple")
  ```

  and returns the WSGI application. *root* is the package directory, *redirect_to_fallback* specifies whether to redirect to *fallback_url* when a package is missing.

9.1 gunicorn

The following command uses *gunicorn* to start *pypiserver*:

```
gunicorn -w4 'pypiserver:app("/home/ralf/packages")'
```

or when using multiple roots:

```
gunicorn -w4 'pypiserver:app(["/home/ralf/packages", "/home/ralf/experimental"])'
```

9.2 apache/mod_wsgi

In case you're using *apache2* with *mod_wsgi*, the following config-file (contributed by Thomas Waldmann) can be used:

```
# An example pypiserver.wsgi for use with apache2 and mod_wsgi, edit as necessary.
#
# apache virtualhost configuration for mod_wsgi daemon mode:
#    Alias /robots.txt /srv/yoursite/htdocs/robots.txt
#    WSGIPassAuthorization On
#    WSGIScriptAlias /     /srv/yoursite/cfg/pypiserver.wsgi
#    WSGIDaemonProcess     pypisrv user=pypisrv group=pypisrv processes=1 threads=5 maximum-requests=
#    WSGIProcessGroup      pypisrv

PACKAGES = "/srv/yoursite/packages"
```

```
HTPASSWD = "/srv/yoursite/htpasswd"
import pypiserver
application = pypiserver.app(PACKAGES, redirect_to_fallback=True, password_file=HTPASSWD)
```

9.3 paste/pastedeploy

paste allows to run multiple WSGI applications under different URL paths. Therefore it's possible to serve different set of packages on different paths.

The following example *paste.ini* could be used to serve stable and unstable packages on different paths:

```
[composite:main]
use = egg:Paste#urlmap
/unstable/ = unstable
/ = stable

[app:stable]
use = egg:pypiserver#main
root = ~/stable-packages

[app:unstable]
use = egg:pypiserver#main
root = ~/stable-packages
       ~/unstable-packages

[server:main]
use = egg:gunicorn#main
host = 0.0.0.0
port = 9000
workers = 5
accesslog = -
```

Note: You need to install some more dependencies for this to work, e.g. run:

```
pip install paste pastedeploy gunicorn pypiserver
```

The server can then be started with:

```
gunicorn_paster paste.ini
```

CHAPTER 10

Sources

Source releases can be downloaded from https://pypi.python.org/pypi/pypiserver

https://github.com/pypiserver/pypiserver carries a git repository of the in-development version.

Use:

```
git clone https://github.com/pypiserver/pypiserver.git
```

to create a copy of the repository, then:

```
git pull
```

inside the copy to receive the latest version.

CHAPTER 11

Bugs

pypiserver does not implement the full API as seen on PyPI. It implements just enough to make *easy_install* and *pip* install work.

The following limitations are known:

- It doesn't implement the XMLRPC interface: pip search will not work.
- It doesn't implement the json based '/pypi' interface.
- It accepts documentation uploads but does not save them to disk (see https://github.com/pypiserver/pypiserver/issues/47 for a discussion)
- It does not handle misspelled packages as pypi-repo does, therefore it is suggested to use it with *–extra-index-url* instead of *–index-url* (see discussion at https://github.com/pypiserver/pypiserver/issues/38)

Please use github's bugtracker if you find any other bugs.

CHAPTER 12

License

pypiserver contains a copy of bottle which is available under the *MIT* license:

```
Copyright (c) 2012, Marcel Hellkamp.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

The remaining part is distributed under the *zlib/libpng* license:

```
Copyright (c) 2011-2014 Ralf Schmitt

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.

3. This notice may not be removed or altered from any source
```

```
distribution.
```

CHAPTER 13

Similar Projects

There are lots of other projects, which allow you to run your own PyPI server. If *pypiserver* doesn't work for you, the following are among the most popular alternatives:

devpi-server **(https://pypi.python.org/pypi/devpi-server)** easy-to-use caching proxy server

proxypypi **(https://pypi.python.org/pypi/proxypypi)** a PyPI caching proxy