Upload source distributions of your requirements to your PyPI server.
The problem

If you are using a custom PyPI server as a proxy and want to upload some packages there, it’s not easy. For each package, you need to download its source and upload it using `setup.py` script:

```bash
# Uploading requests==2.0.0
$ git clone https://github.com/kennethreitz/requests
$ cd requests
$ git checkout v2.0.0
# Assuming you have ‘internal’ index-server configured in your ‘~/.pypirc’.
$ python setup.py sdist upload -r internal
# Uploading coverage==3.5
$ cd ..
$ git clone https://github.com/nedbat/coveragepy
$ cd coveragepy
$ git checkout coverage-3.5
$ python setup.py sdist upload -r internal
```

You could also download the packages directly into the PyPI’s index directory.

```bash
$ ssh pypi-mirror.yourdomain.com
$ pip install requests==2.0.0 coverage==3.5 -d ~/.packages
```

If there’s more than one package, you could use a requirements file.

```bash
$ scp requirements.txt pypi-mirror.yourdomain.com:.
$ ssh pypi-mirror.yourdomain.com
$ pip install -r requirements.txt -d ~/.packages
```

But it’s still too much. You should be able to do it with one command. And what if you don’t have SSH access to the PyPI server host?
The solution

One command for download and upload.

2.1 Upload packages by name

$ pypiupload packages mock==1.0.1 requests==2.2.1 -i internal

2.2 Upload packages from requirements file

$ pypiupload requirements requirements.txt -i internal

2.3 Upload source distribution files

$ pypiupload files packages/mock-1.0.1.tar.gz packages/requests-2.2.1-py2.py3-none-any.whl -i internal

2.4 More options

$ pypiupload --help
$ pypiupload <command> --help
CHAPTER 3

Supported PyPI servers

Tested only on pypiserver.
Installation

Install from PyPI:

$ pip install pypi-uploader

Or go to the root directory with setup.py script and install it:

$ python setup.py install
Documentation

Documentation is available at https://pypi-uploader.readthedocs.org
Source

Source is available at https://github.com/ignacysokolowski/pypi-uploader
License

PyPI Uploader is licensed under the MIT license.
8.1 Version 1.0.0

- Added `--no-use-wheel` option

8.2 Version 0.1.0

First release
Issues and contributing

Please report any issues on GitHub at https://github.com/ignacysokolowski/pypi-uploader/issues

Or contribute by submitting a pull request with your changes following these rules:

- Follow PEP 8 rules
- Follow PEP 257 rules
- Follow The Zen of Python
- Test your commits
- Write meaningful commit messages
- Keep the documentation up-to-date

To run tests and build the docs, you have to install additional packages:

$ python setup.py develop
$ pip install -r requirements_dev.txt

Running tests:

$ tox

Building documentation:

$ cd docs
$ make html
10.1 pypiuploader Package

Upload source distributions of your requirements to your PyPI server.

10.1.1 pypiuploader.commands Module

Commands for pypiupload program.

main() is installed as a console script during package setup.

Examples:

```bash
$ pypiupload requirements requirements.txt -i internal
$ pypiupload packages mock==1.2.1 requests==2.0.0 -i internal
$ pypiupload files packages/mock-1.2.1.tar.gz -i internal
$ pypiupload requirements requirements.txt
   \ -i http://localhost:8000 -u user -p password -d packages_download_dir
$ pypiupload requirements requirements.txt
   \ -i http://localhost:8000 --no-use-wheel
```

```python
main (argv=None, stdout=None)

Run the Command.

Parameters

- **argv** – A list of arguments to parse, defaults to sys.argv.
- **stdout** – Standard output file, defaults to sys.stdout.
```

class Command (options, stdout=None)

Runs a pypiupload command.

Available commands:

- requirements
- packages
- files

For files command, upload the given source distribution files.
PyPI Uploader, Release 1.0.0

For **requirements** and **packages**, first download the packages and then upload them. **download_dir** option (-d or --download-dir) can specify the directory to which the files will be downloaded. If not given, will create a temporary directory. With **no_use_wheel** option (--no-use-wheel), will not find and prefer wheel archives.

**index** option (-i or --index-url) specifies the PyPI host or name from the ~/.pypirc config file. If **username** (-u or --username) and **password** (-p or --password) are not given, will try to find them in the rc file.

**Parameters**

- **options** – Command arguments, argparse.Namespace instance parsed by the `parse_args()`.
- **stdout** – Standard output file, defaults to sys.stdout.

**run()**

Run the command.

- Make the upload.PackageUploader instance. If ~/.pypirc file exists, parse it using pypirc.RCParser
- Define the files to upload – download them if necessary, using download.PackageDownloader
- Upload the packages

**parse_args(argv)**

Parse arguments for the commands.

Return a argparse.Namespace instance.

**Parameters argv** – A list of arguments to parse.

### 10.1.2 pypiuploader.exceptions Module

Custom exception classes.

**exception** ConfigFileError

Bases: exceptions.Exception

Raised when a PyPI config file does not exist.

**exception** PackageConflictError

Bases: exceptions.Exception

Raised when the version of the package was already uploaded.

The PyPI server responds with HTTP 409 and the upload.PackageUploader raises this exception instead.

### 10.1.3 pypiuploader.download Module

Downloading packages to a directory.

**class** PackageDownloader (download_path=None)

Bases: object

Downloads source distributions from PyPI to a directory.

Runs pip install command, e.g.:

```python
>>> downloader = PackageDownloader('/~.packages')
>>> downloader.download(['mock', 'requests==1.2.1'])
```
Would run:

$ pip install -d ~/.packages mock requests==1.2.1

And this:

```python
>>> downloader = PackageDownloader('~/.packages')
>>> downloader.download(requirements_file='requirements.txt')
```

Would run:

$ pip install -d ~/.packages -r requirements.txt

**Parameters**

- **download_path** – Optional path to directory where the packages should be downloaded, will be set as `download_path`. If it doesn’t exist, will be created on `download()`. If not given, will create a temporary directory and update `download_path` on `download()`.

- **download_path** = `None`

  Path to directory where the packages should be downloaded.

- **download** (**requirements**=`None`, **requirements_file**=`None`, **no_use_wheel**=`False`)

  Download the packages using `pip install` command.

  Either `requirements` or `requirements_file` must be given, otherwise raise `ValueError`.

  Return a generator yielding full paths to the downloaded packages.

  **Parameters**

  - **requirements** – Optional list of packages names to download.
  
  - **requirements_file** – Optional path to a requirements file.
  
  - **no_use_wheel** – Do not find and prefer wheel archives, default to `False`. Corresponds to `--no-use-wheel` option from `pip install`.

### 10.1.4 pypiuploader.pypirc Module

Reading `.pypirc` file.

**class** **RCParser** ((**config_parser**))

- **Bases** : `object`

  Parser for the `~/.pypirc` file.

  Parses the file to find a specified repository’s config:

  - repository URL
  
  - username
  
  - password

  Example:

  ```python
  >>> parser = RCParse.from_file()
  >>> parser.get_repository_config('internal')
  {'repository': 'http://localhost/', 'username': 'foo', 'password': 'bar'}
  ```

10.1. **pypiuploader Package**
Parameters **config_parser** – `configparser.ConfigParser` instance with the `.pypirc` content loaded.

```python
config = None
configparser.ConfigParser instance to parse.
```

```python
classmethod from_file(path=None)
Read a config file and instantiate the RCParser.
Create new `configparser.ConfigParser` for the given `path` and instantiate the RCParser with the ConfigParser as `config` attribute.
If the `path` doesn’t exist, raise `exceptions.ConfigFileError`. Otherwise return a new RCParser instance.

Parameters `path` – Optional path to the config file to parse. If not given, use ‘~/.pypirc’.
```

```python
get_repository_config(repository)
Get config dictionary for the given repository.
If the repository section is not found in the config file, return None. If the file is invalid, raise`configparser.Error`. Otherwise return a dictionary with:

• `'repository’ – the repository URL
• `'username’ – username for authentication
• `'password’ – password for authentication

Parameters `repository` – Name or URL of the repository to find in the `.pypirc` file. The repository section must be defined in the config file.
```

```python
read_config(path)
Make a config parser for the given config file.
Return a `configparser.ConfigParser` instance with the given file loaded.

Parameters `path` – Path to the config file to read.
```

## 10.1.5 `pypuploader.upload` Module

Packages uploading.

```python
class PackageUploader (host, username=None, password=None)
Bases: object

Uploads source distributions to a PyPI server.

Example:
```python
>>> uploader = PackageUploader('http://localhost:8000', 'user', 'pass')
>>> uploader.upload('amqp-1.4.4.tar.gz')
<Response [200]>
>>> uploader.upload('amqp-1.4.4.tar.gz')
Traceback (most recent call last):
...
PackageConflictError: Package amqp-1.4.4.tar.gz already uploaded.
```

Parameters
• **host** – The host of the PyPI server, e.g. ‘http://localhost:8000’.
• **username** – Optional username for HTTP authentication.
• **password** – Optional password for HTTP authentication.

**classmethod from rc file** *(repository, username=None, password=None, config_path=None)*

Instantiate the uploader using configuration from .pypirc file.

Read the rc file using `pypirc.RCParse`. Read the rc file using `pypirc.RCParse`.

Use the `repository`’s authentication config from the rc file, if the file exists and the repository section is defined. The **username** and **password** defined in the file can be overridden by the given arguments.

If the rc file or the section for the `repository` doesn’t exist, use the `repository` argument as the server host.

Return a new instance of `PackageUploader`.

**Parameters**

- **repository** – PyPI server name or URL.
- **username** – Optional username for authentication.
- **password** – Optional password for authentication.
- **config_path** – Optional config path to use instead of ~/.pypirc.

**classmethod from repository config** *(repo_config, username=None, password=None)*

Instantiate the uploader using repository configuration dictionary.

Examples:

```python
>>> repo_config = {
...     'repository': 'http://localhost:8000',
...     'username': 'foo',
...     'password': 'bar',
... }
>>> uploader = PackageUploader.from_repository_config(repo_config)
>>> uploader.host, uploader.username, uploader.password
('http://localhost:8000', 'foo', 'bar')
>>> uploader = PackageUploader.from_repository_config(
...     repo_config, username='bar')
>>> uploader.host, uploader.username, uploader.password
('http://localhost:8000', 'bar', 'bar')
```

Return a new instance of `PackageUploader`.

**Parameters**

- **repo_config** – A dictionary with the repository configuration, must contain three keys:
  - repository,
  - username
  - password
- **username** – Optional username to override the one from the `repo_config`.
- **password** – Optional password to override the one from the `repo_config`.

**upload** *(filepath)*

Upload a package under the given path.

Return `requests.Response` from the PyPI server.
If the package is already uploaded, raise `exceptions.PackageConflictError`. On other errors raise `requests.exceptions.HTTPError`.

**Parameters**

- `filepath` – A path to the package file you want to upload.
Indices and tables

- genindex
- modindex
- search
p
pypiuploader, ??
pypiuploader.commands, ??
pypiuploader.download, ??
pypiuploader.exceptions, ??
pypiuploader.pypirc, ??
pypiuploader.upload, ??