Contents

1 Installation 3
2 Usage 5
3 API Reference 7
4 Additional Notes 9
Python Module Index 13
Index 15
Python interface to your NPM and package.json.

Further documentation is available on https://pynpm.readthedocs.io/.
CHAPTER 1

Installation

PyNPM is on PyPI so all you need is:

```
$ pip install pynpm
```
First point PyNPM to your `package.json`:

```python
from pynpm import NPMPackage
pkg = NPMPackage('path/to/package.json')
```

Now you can run e.g. `npm install` from within Python:

```python
pkg.install()
```

Arguments are also support so you can run e.g. `npm run build --report`:

```python
pkg.run_script('build', '--report')
```

Want to use `yarn` instead?

```python
from pynpm import YarnPackage
pkg = YarnPackage('path/to/package.json')
pkg.install()
```

By default NPM output is piped through and the function call will wait for NPM to finish. If you want to silence the output or interact with process pass `wait=False` and you will get a subprocess.POpen object back:

```python
p = pkg.install(wait=False)
p.wait()
```

By default you can run the following NPM commands:

- build
- init
- install
- link
- run-script
• start
• stop
• test

You can also run other NPM commands or restrict which commands you can run:

```python
pkg = NPMPackage('path/to/package.json', commands=['install'])
```
PyNPM is a small API to help you invoke NPM from inside Python.

```python
class pynpm.NPMPackage(filepath, npm_bin='npm', commands=None)
    API to an NPM package.json.

    Parameters
    • filepath – Path to package.json or directory containing the file.
    • npm_bin – Path to NPM binary. Defaults to npm.
    • commands – List of allowed NPM commands to invoke.
```

Initialize package.

```python
package_json
    Read package.json contents.

package_json_path
    Get package.json file path.
```

```python
class pynpm.YarnPackage(filepath, yarn_bin='yarn', commands=None)
    Yarn package.

    Initialize package.
```
Notes on how to contribute, legal information and changes are here for the interested.

### 4.1 Contributing

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

#### 4.1.1 Types of Contributions

**Report Bugs**


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.

**Fix Bugs**

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

**Implement Features**

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.
Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/pynpm/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 :)

4.1.2 Get Started!

Ready to contribute? Here’s how to set up pynpm for local development.

1. Fork the inveniosoftware/pynpm repo on GitHub.

2. Clone your fork locally:

   ```bash
   $ git clone git@github.com:your_name_here/pynpm.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:

   ```bash
   $ mkvirtualenv pynpm
   $ cd pynpm/
   $ pip install -e .[all]
   ```

4. Create a branch for local development:

   ```bash
   $ 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 tests:

   ```bash
   $ ./run-tests.sh
   ```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

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

   ```bash
   $ git add .
   $ git commit -s
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.
4.1.3 Pull Request Guidelines

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

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.org/inveniosoftware/pynpm/pull_requests and make sure that the tests pass for all supported Python versions.

4.2 Changes

Version 0.1.2 (released 2020-05-06)
- Deprecated Python versions lower than 3.6.0. Now supporting 3.6.0 and 3.7.0.

Version 0.1.1 (released 2017-05-16)
- Fix problem with testing click CLI output.

Version 0.1.0 (released 2017-05-12)
- Initial public release.

4.3 License

PyNPM is free software; you can redistribute it and/or modify it under the terms of the Revised BSD License quoted below.

Copyright (C) 2017 CERN.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.
4.4 Authors

Python interface to your NPM and package.json.

- Lars Holm Nielsen <lars.holm.nielsen@cern.ch>
Python Module Index

p
  pynpm, 7
Index

N
NPMPackage (class in pynpm), 7

P
package_json (pynpm.NPMPackage attribute), 7
package_json_path (pynpm.NPMPackage attribute), 7
pynpm (module), 7

Y
YarnPackage (class in pynpm), 7