Packaging and Distributing

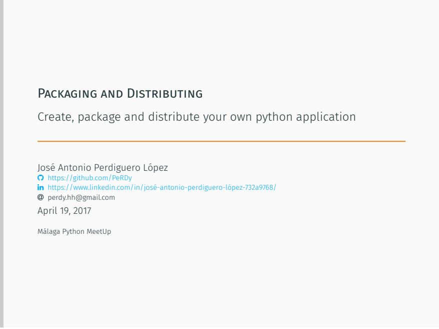
Create, package and distribute your own python application

José Antonio Perdiguero López
https://github.com/PeRDy
https://www.linkedin.com/in/josé-antonio-perdiguero-lópez-732a9768/
@ perdy.hh@gmail.com

April 19, 2017

Málaga Python MeetUp

2017-04-19



Packaging and Distributing

Create, package and distribute your own python application

José Antonio Perdiguero López
https://github.com/PeRDy
https://www.linkedin.com/in/josé-antonio-perdiguero-lópez-732a9768/
@ perdy.hh@gmail.com

April 19, 2017

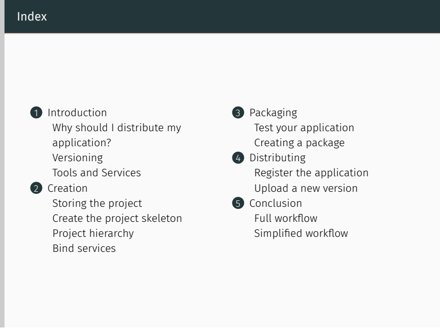
Málaga Python MeetUp

Index

1. Introduction
   - Why should I distribute my application?
   - Versioning
   - Tools and Services
2. Creation
   - Storing the project
   - Create the project skeleton
   - Project hierarchy
   - Bind services
3. Packaging
   - Test your application
   - Creating a package
4. Distributing
   - Register the application
   - Upload a new version
5. Conclusion
   - Full workflow
   - Simplified workflow

2017-04-19

Index


Index

1. Introduction
   - Why should I distribute my application?
   - Versioning
   - Tools and Services
2. Creation
   - Storing the project
   - Create the project skeleton
   - Project hierarchy
   - Bind services
3. Packaging
   - Test your application
   - Creating a package
4. Distributing
   - Register the application
   - Upload a new version
5. Conclusion
   - Full workflow
   - Simplified workflow

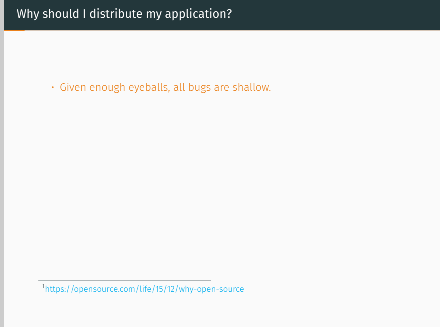
Introduction

Introduction

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.

[1] https://opensource.com/life/15/12/why-open-source


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.

[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

Show your code

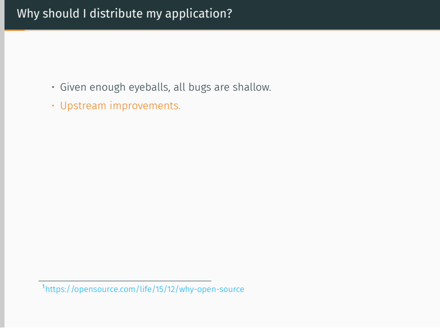
Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.

[1]https://opensource.com/life/15/12/why-open-source

2017-04-19

Packaging and Distributing
└ Introduction
  └ Why should I distribute my application?
    └ Why should I distribute my application?


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.

[1]https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

- **Force multiplier:** Diversity of ideas from exposing the problem.
- **Modular:** Open source projects tend to be more modularly architected.
- **Great advertising:** Maintainers of successful open source projects are often seen as industry leaders
- **Attract talent:** Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

- **Show your code**

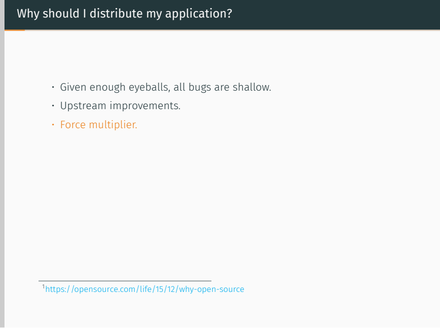
Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.

[1]https://opensource.com/life/15/12/why-open-source

2017-04-19

Packaging and Distributing
└ Introduction
  └ Why should I distribute my application?
    └ Why should I distribute my application?


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.

[1]https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

- **Force multiplier:** Diversity of ideas from exposing the problem.
- **Modular:** Open source projects tend to be more modularly architected.
- **Great advertising:** Maintainers of successful open source projects are often seen as industry leaders
- **Attract talent:** Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

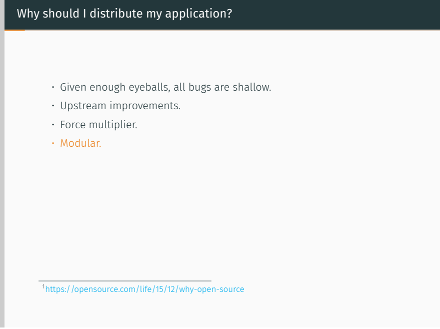
Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

- **Show your code**

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.

[1]https://opensource.com/life/15/12/why-open-source


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.

[1]https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

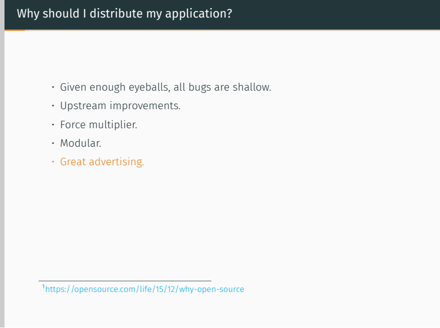
Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

Show your code

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.

[1] https://opensource.com/life/15/12/why-open-source



Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.

[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

Show your code

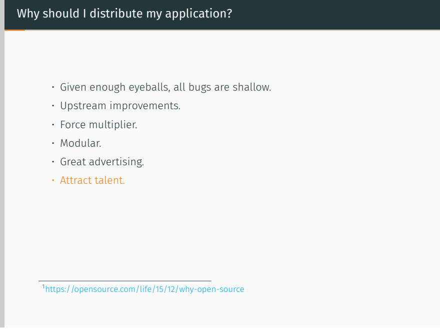
Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.

[1] https://opensource.com/life/15/12/why-open-source

2017-04-19

Packaging and Distributing
└ Introduction
  └ Why should I distribute my application?
    └ Why should I distribute my application?


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.

[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

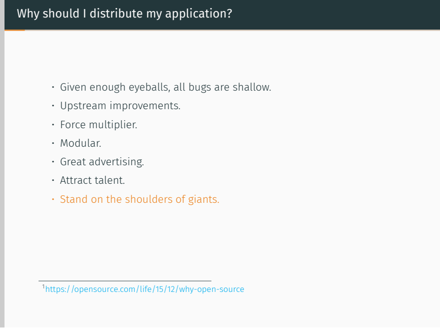
Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

Show your code

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.

[1] https://opensource.com/life/15/12/why-open-source


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.

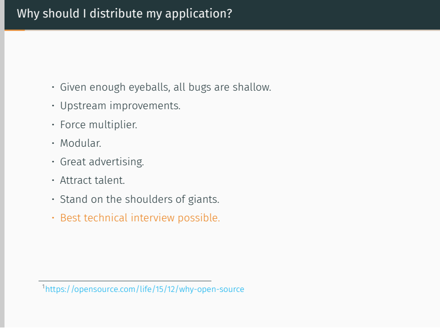
[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow	
Upstream improvements:	If you consume open source software, it’s in your best interest to contribute back.
Force multiplier:	Diversity of ideas from exposing the problem.
Modular:	Open source projects tend to be more modularly architected.
Great advertising:	Maintainers of successful open source projects are often seen as industry leaders
Attract talent:	Developers want to work on yet-unsolved problems.
Stand on the shoulders of giants	
Best technical interview possible:	You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.
Show your code	

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.
- Best technical interview possible.

[1] https://opensource.com/life/15/12/why-open-source



Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.
- Best technical interview possible.

[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

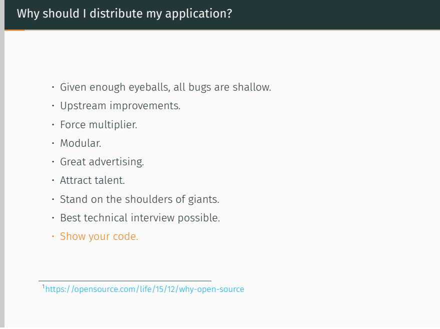
Show your code

Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.
- Best technical interview possible.
- Show your code.

[1] https://opensource.com/life/15/12/why-open-source

2017-04-19


Why should I distribute my application?

- Given enough eyeballs, all bugs are shallow.
- Upstream improvements.
- Force multiplier.
- Modular.
- Great advertising.
- Attract talent.
- Stand on the shoulders of giants.
- Best technical interview possible.
- Show your code.

[1] https://opensource.com/life/15/12/why-open-source

Given enough eyeballs, all bugs are shallow

Upstream improvements: If you consume open source software, it's in your best interest to contribute back.

Force multiplier: Diversity of ideas from exposing the problem.

Modular: Open source projects tend to be more modularly architected.

Great advertising: Maintainers of successful open source projects are often seen as industry leaders

Attract talent: Developers want to work on yet-unsolved problems.

Stand on the shoulders of giants

Best technical interview possible: You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work.

Show your code

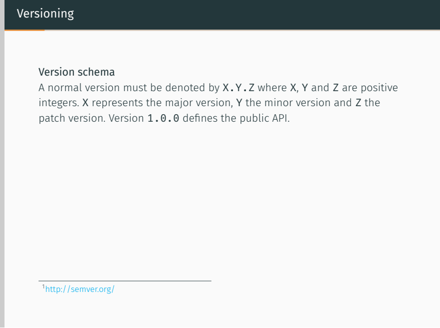
Versioning

Version schema

A normal version must be denoted by `X.Y.Z` where X, Y and Z are positive integers. X represents the major version, Y the minor version and Z the patch version. Version `1.0.0` defines the public API.

[1] http://semver.org/

2017-04-19


Versioning

Version schema

A normal version must be denoted by `X.Y.Z` where X, Y and Z are positive integers. X represents the major version, Y the minor version and Z the patch version. Version `1.0.0` defines the public API.

[1] http://semver.org/

Versioning

Version schema

A normal version must be denoted by `X.Y.Z` where X, Y and Z are positive integers. X represents the major version, Y the minor version and Z the patch version. Version `1.0.0` defines the public API.

Version upgrade

Given a version number, increment:

Major	version when you make incompatible API changes. Reset minor and patch version to `0`.
Minor	version when you add functionality in a backwards-compatible manner. Reset patch version to `0`.
Patch	version when you make backwards-compatible bug fixes.

[1] http://semver.org/

2017-04-19

Versioning

Version schema
A normal version must be denoted by X.Y.Z where X, Y and Z are positive integers. X represents the major version, Y the minor version and Z the patch version. Version 1.0.0 defines the public API.

Version upgrade
Given a version number, increment:
Major version when you make incompatible API changes. Reset minor and patch version to 0.
Minor version when you add functionality in a backwards-compatible manner. Reset patch version to 0.
Patch version when you make backwards-compatible bug fixes.

[1] http://semver.org/

Tools I

Prospector

Static code analysis using different tools.

https://github.com/landscapeio/prospector

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

2017-04-19

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

Bumpversion
Utility to upgrade your project version.
https://github.com/peritus/bumpversion

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

Bumpversion
Utility to upgrade your project version.
https://github.com/peritus/bumpversion

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

Bumpversion
Utility to upgrade your project version.
https://github.com/peritus/bumpversion

Pre-commit
Utility that does some checks before git commits.
https://github.com/pre-commit/pre-commit

2017-04-19

Tools I

Prospector
Static code analysis using different tools.
https://github.com/landscapeio/prospector

Sphinx
Create documentation for your project.
https://github.com/sphinx-doc/sphinx

Bumpversion
Utility to upgrade your project version.
https://github.com/peritus/bumpversion

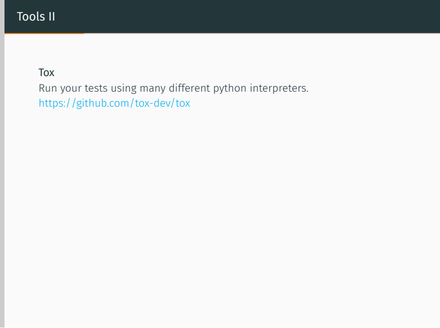
Pre-commit
Utility that does some checks before git commits.
https://github.com/pre-commit/pre-commit

Tools II

Tox

Run your tests using many different python interpreters.

https://github.com/tox-dev/tox


Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Cookiecutter reduces the task of creates the whole project structure.

Clinner simplifies the process of create a package and upload it.

Tox will ease the testing.

Clinner is the cornerstone around the rest of tools, the build script done as example in the doc allows to define a single entrypoint for the rest of tools. http://clinner.readthedocs.io/en/latest/examples.htmlbuilder-main.

A deeper explanation about the cookiecutter template and the project hierarchy will come in next sections.

Distutils: setup.py uses distutils.

Tools II

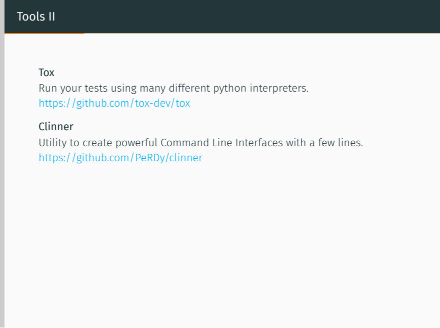
Tox

Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner

Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

2017-04-19



Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner
Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

Cookiecutter reduces the task of creates the whole project structure.

Clinner simplifies the process of create a package and upload it.

Tox will ease the testing.

Clinner is the cornerstone around the rest of tools, the build script done as example in the doc allows to define a single entrypoint for the rest of tools. http://clinner.readthedocs.io/en/latest/examples.htmlbuilder-main.

A deeper explanation about the cookiecutter template and the project hierarchy will come in next sections.

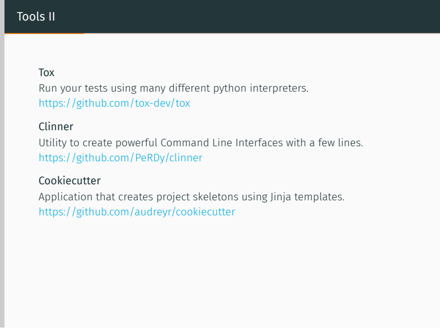
Distutils: setup.py uses distutils.

Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner
Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

Cookiecutter
Application that creates project skeletons using Jinja templates.
https://github.com/audreyr/cookiecutter


Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner
Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

Cookiecutter
Application that creates project skeletons using Jinja templates.
https://github.com/audreyr/cookiecutter

Cookiecutter reduces the task of creates the whole project structure.

Clinner simplifies the process of create a package and upload it.

Tox will ease the testing.

Clinner is the cornerstone around the rest of tools, the build script done as example in the doc allows to define a single entrypoint for the rest of tools. http://clinner.readthedocs.io/en/latest/examples.htmlbuilder-main.

A deeper explanation about the cookiecutter template and the project hierarchy will come in next sections.

Distutils: setup.py uses distutils.

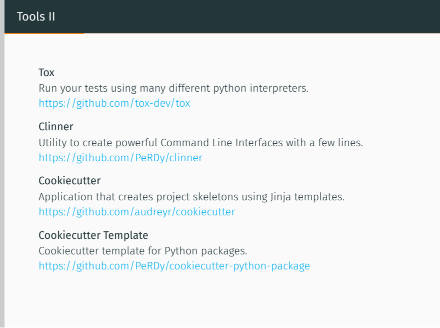
Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner
Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

Cookiecutter
Application that creates project skeletons using Jinja templates.
https://github.com/audreyr/cookiecutter

Cookiecutter Template
Cookiecutter template for Python packages.
https://github.com/PeRDy/cookiecutter-python-package


Tools II

Tox
Run your tests using many different python interpreters.
https://github.com/tox-dev/tox

Clinner
Utility to create powerful Command Line Interfaces with a few lines.
https://github.com/PeRDy/clinner

Cookiecutter
Application that creates project skeletons using Jinja templates.
https://github.com/audreyr/cookiecutter

Cookiecutter Template
Cookiecutter template for Python packages.
https://github.com/PeRDy/cookiecutter-python-package

Cookiecutter reduces the task of creates the whole project structure.

Clinner simplifies the process of create a package and upload it.

Tox will ease the testing.

Clinner is the cornerstone around the rest of tools, the build script done as example in the doc allows to define a single entrypoint for the rest of tools. http://clinner.readthedocs.io/en/latest/examples.htmlbuilder-main.

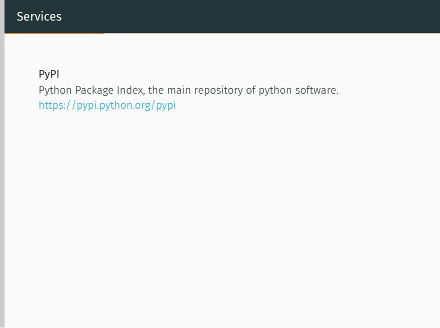
A deeper explanation about the cookiecutter template and the project hierarchy will come in next sections.

Distutils: setup.py uses distutils.

Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

2017-04-19


Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

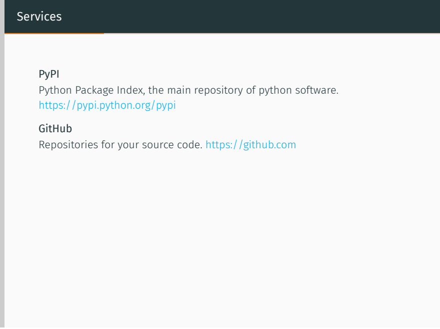
Alternatives to GitHub: *Bitbucket*, *Gitlab*.

Travis and *Coveralls* are free for your open source projects.

Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com


Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Alternatives to GitHub: *Bitbucket*, *Gitlab*.

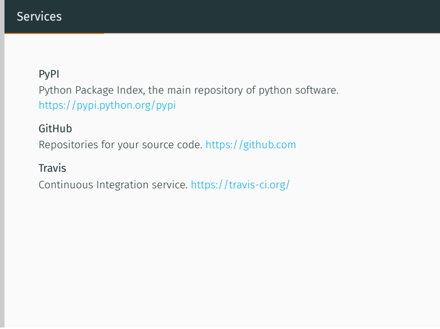
Travis and *Coveralls* are free for your open source projects.

Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/


Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/

Alternatives to GitHub: *Bitbucket*, *Gitlab*.

Travis and *Coveralls* are free for your open source projects.

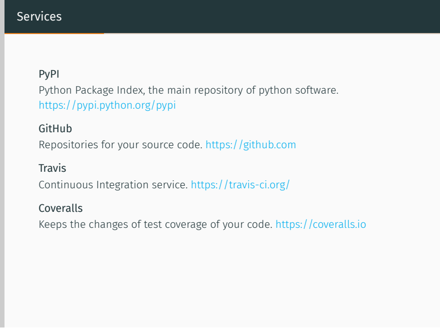
Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/

Coveralls
Keeps the changes of test coverage of your code. https://coveralls.io


Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/

Coveralls
Keeps the changes of test coverage of your code. https://coveralls.io

Alternatives to GitHub: *Bitbucket*, *Gitlab*.

Travis and *Coveralls* are free for your open source projects.

Services

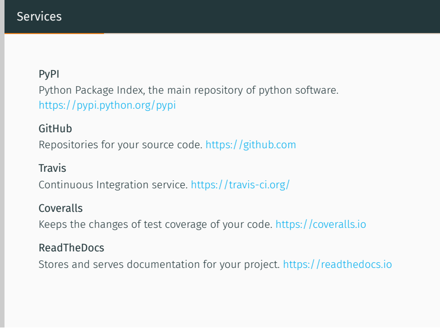
PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/

Coveralls
Keeps the changes of test coverage of your code. https://coveralls.io

ReadTheDocs
Stores and serves documentation for your project. https://readthedocs.io


Services

PyPI
Python Package Index, the main repository of python software.
https://pypi.python.org/pypi

GitHub
Repositories for your source code. https://github.com

Travis
Continuous Integration service. https://travis-ci.org/

Coveralls
Keeps the changes of test coverage of your code. https://coveralls.io

ReadTheDocs
Stores and serves documentation for your project. https://readthedocs.io

Alternatives to GitHub: *Bitbucket*, *Gitlab*.

Travis and *Coveralls* are free for your open source projects.

Creation

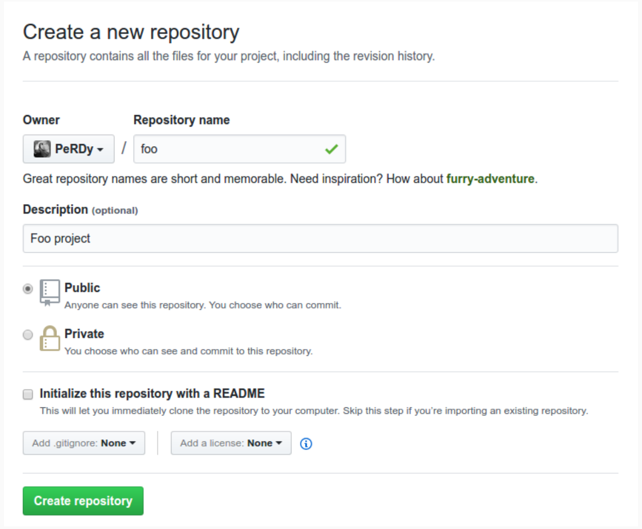
Packaging and Distributing

└─ Creation

2017-04-19

Creation

Storing the project


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: PeRDy / Repository name: foo

Great repository names are short and memorable. Need inspiration? How about **furry-adventure**.

Description (optional)

Foo project

[✓ **Public**] Anyone can see this repository. You choose who can commit.

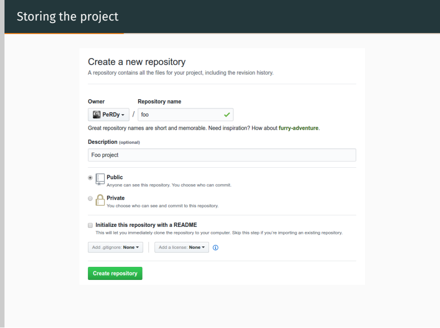
Private You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

2017-04-19

Packaging and Distributing
└ Creation
  └ Storing the project
    └ Storing the project


Storing the project

This speech is focused on open source, so working on GitHub.

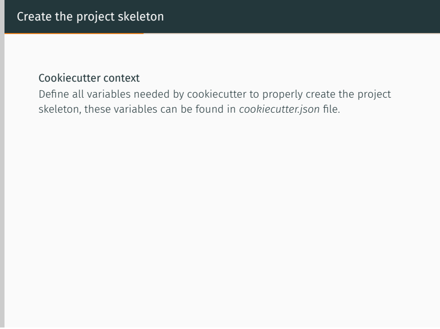
Create the project skeleton

Cookiecutter context

Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

2017-04-19

- Packaging and Distributing
  - Creation
    - Create the project skeleton
      - Create the project skeleton



Create the project skeleton

Cookiecutter context

Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

Create the project skeleton

Cookiecutter context
Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

Create skeleton
Execute cookiecutter with previously defined context to create the project skeleton.

Create the project skeleton

Cookiecutter context
Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

Create skeleton
Execute cookiecutter with previously defined context to create the project skeleton.

Create the project skeleton

Cookiecutter context

Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

Create skeleton

Execute cookiecutter with previously defined context to create the project skeleton.

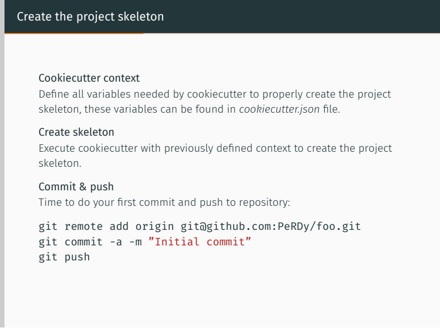
Commit & push

Time to do your first commit and push to repository:

```
git remote add origin git@github.com:PeRDy/foo.git
git commit -a -m "Initial commit"
git push
```

2017-04-19

Packaging and Distributing
└─Creation
  └─Create the project skeleton
    └─Create the project skeleton


Create the project skeleton

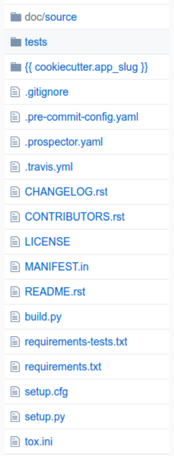
Cookiecutter context
Define all variables needed by cookiecutter to properly create the project skeleton, these variables can be found in *cookiecutter.json* file.

Create skeleton
Execute cookiecutter with previously defined context to create the project skeleton.

Commit & push
Time to do your first commit and push to repository:

git remote add origin git@github.com:PeRDy/foo.git
git commit -a -m "Initial commit"
git push

Project hierarchy

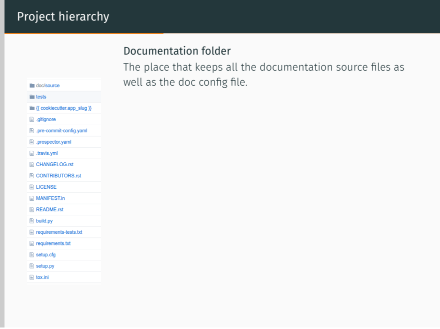

doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder

The place that keeps all the documentation source files as well as the doc config file.

2017-04-19

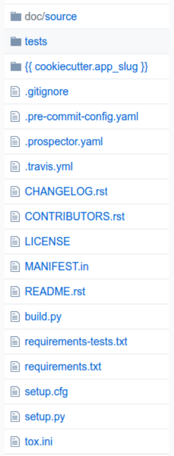
Packaging and Distributing
└─Creation
  └─Project hierarchy
    └─Project hierarchy


Project hierarchy

Documentation folder
The place that keeps all the documentation source files as well as the doc config file.

Tools configuration	Prospector, Pre-commit, Git.
Services configuration	Travis.
Build scripts	`build.py`, `setup.py`, `tox.ini`.
Metadata	Readme, manifest, contributors, changelog, requirements, `setup.cfg`.

Project hierarchy

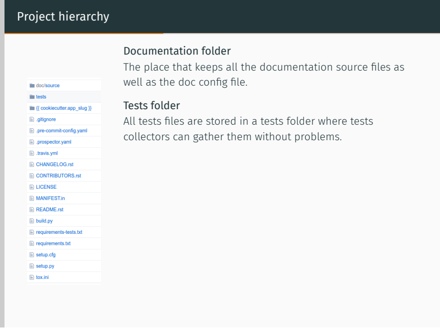

doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder

The place that keeps all the documentation source files as well as the doc config file.

Tests folder

All tests files are stored in a tests folder where tests collectors can gather them without problems.


Project hierarchy

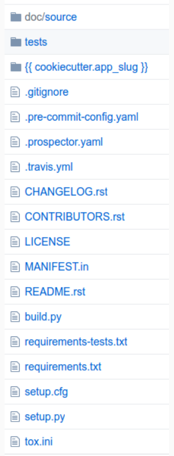
doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder
The place that keeps all the documentation source files as well as the doc config file.

Tests folder
All tests files are stored in a tests folder where tests collectors can gather them without problems.

Tools configuration	Prospector, Pre-commit, Git.
Services configuration	Travis.
Build scripts	`build.py`, `setup.py`, `tox.ini`.
Metadata	Readme, manifest, contributors, changelog, requirements, `setup.cfg`.

Project hierarchy


doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder

The place that keeps all the documentation source files as well as the doc config file.

Tests folder

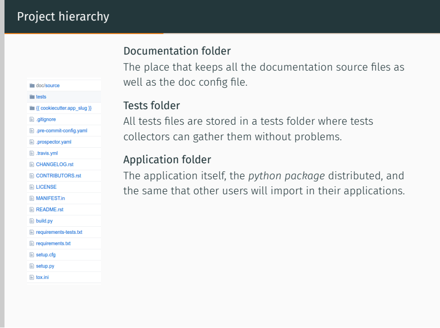
All tests files are stored in a tests folder where tests collectors can gather them without problems.

Application folder

The application itself, the *python package* distributed, and the same that other users will import in their applications.

2017-04-19

Packaging and Distributing
└Creation
  └Project hierarchy
    └Project hierarchy


Project hierarchy

doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

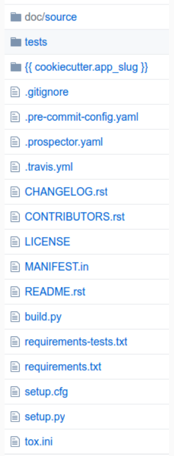
Documentation folder
The place that keeps all the documentation source files as well as the doc config file.

Tests folder
All tests files are stored in a tests folder where tests collectors can gather them without problems.

Application folder
The application itself, the *python package* distributed, and the same that other users will import in their applications.

Tools configuration	Prospector, Pre-commit, Git.
Services configuration	Travis.
Build scripts	`build.py`, `setup.py`, `tox.ini`.
Metadata	Readme, manifest, contributors, changelog, requirements, `setup.cfg`.

Project hierarchy


doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder
The place that keeps all the documentation source files as well as the doc config file.

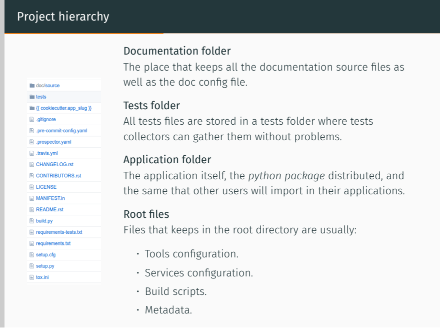
Tests folder
All tests files are stored in a tests folder where tests collectors can gather them without problems.

Application folder
The application itself, the *python package* distributed, and the same that other users will import in their applications.

Root files
Files that keeps in the root directory are usually:

- Tools configuration.
- Services configuration.
- Build scripts.
- Metadata.

2017-04-19


Project hierarchy

doc/source
tests
{{ cookiecutter.app_slug }}
.gitignore
.pre-commit-config.yaml
.prospector.yaml
.travis.yml
CHANGELOG.rst
CONTRIBUTORS.rst
LICENSE
MANIFEST.in
README.rst
build.py
requirements-tests.txt
requirements.txt
setup.cfg
setup.py
tox.ini

Documentation folder
The place that keeps all the documentation source files as well as the doc config file.

Tests folder
All tests files are stored in a tests folder where tests collectors can gather them without problems.

Application folder
The application itself, the *python package* distributed, and the same that other users will import in their applications.

Root files
Files that keeps in the root directory are usually:

- Tools configuration.
- Services configuration.
- Build scripts.
- Metadata.

Tools configuration	Prospector, Pre-commit, Git.
Services configuration	Travis.
Build scripts	`build.py`, `setup.py`, `tox.ini`.
Metadata	Readme, manifest, contributors, changelog, requirements, `setup.cfg`.

Relevant files I

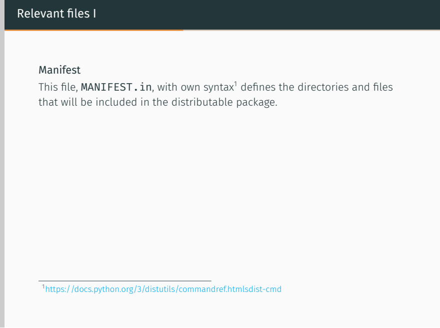
Manifest

This file, `MANIFEST.in`, with own syntax[1] defines the directories and files that will be included in the distributable package.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd

2017-04-19

Packaging and Distributing
└ Creation
  └ Project hierarchy
    └ Relevant files I


Relevant files I

Manifest

This file, MANIFEST.in, with own syntax[1] defines the directories and files that will be included in the distributable package.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd

Readme can be written in rst or md, and will be the front page of the project in GitHub.

Relevant files I

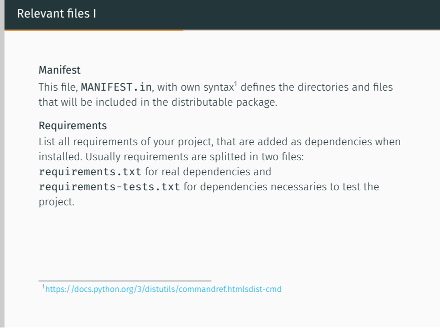
Manifest

This file, `MANIFEST.in`, with own syntax[1] defines the directories and files that will be included in the distributable package.

Requirements

List all requirements of your project, that are added as dependencies when installed. Usually requirements are splitted in two files: `requirements.txt` for real dependencies and `requirements-tests.txt` for dependencies necessaries to test the project.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd



Relevant files I

Manifest
This file, `MANIFEST.in`, with own syntax[1] defines the directories and files that will be included in the distributable package.

Requirements
List all requirements of your project, that are added as dependencies when installed. Usually requirements are splitted in two files: `requirements.txt` for real dependencies and `requirements-tests.txt` for dependencies necessaries to test the project.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd

Readme can be written in rst or md, and will be the front page of the project in GitHub.

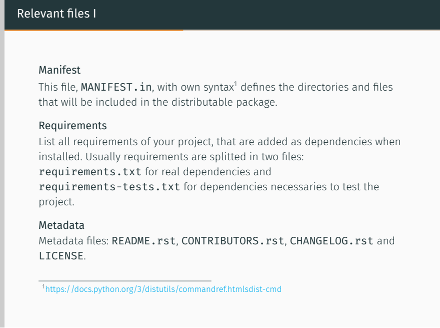
Relevant files I

Manifest
This file, `MANIFEST.in`, with own syntax[1] defines the directories and files that will be included in the distributable package.

Requirements
List all requirements of your project, that are added as dependencies when installed. Usually requirements are splitted in two files:
`requirements.txt` for real dependencies and
`requirements-tests.txt` for dependencies necessaries to test the project.

Metadata
Metadata files: `README.rst`, `CONTRIBUTORS.rst`, `CHANGELOG.rst` and `LICENSE`.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd


Relevant files I

Manifest
This file, `MANIFEST.in`, with own syntax[1] defines the directories and files that will be included in the distributable package.

Requirements
List all requirements of your project, that are added as dependencies when installed. Usually requirements are splitted in two files:
`requirements.txt` for real dependencies and
`requirements-tests.txt` for dependencies necessaries to test the project.

Metadata
Metadata files: `README.rst`, `CONTRIBUTORS.rst`, `CHANGELOG.rst` and `LICENSE`.

[1] https://docs.python.org/3/distutils/commandref.htmlsdist-cmd

Readme can be written in rst or md, and will be the front page of the project in GitHub.

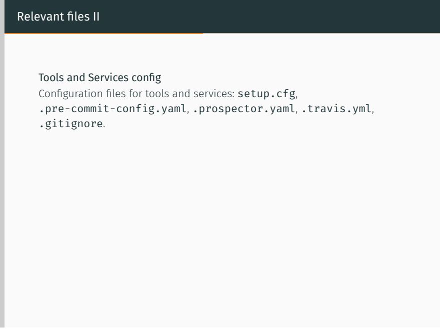
Relevant files II

Tools and Services config

Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

2017-04-19

Packaging and Distributing
└─Creation
  └─Project hierarchy
    └─Relevant files II


Relevant files II

Tools and Services config
Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

Main configuration file: `setup.cfg`. Ini-style file with sections for configuration of different tools, like *bumpversion*, *nose* and *coverage*.

Setup file keeps the requirements list, the current version, application name...

Tox is integrated with travis.

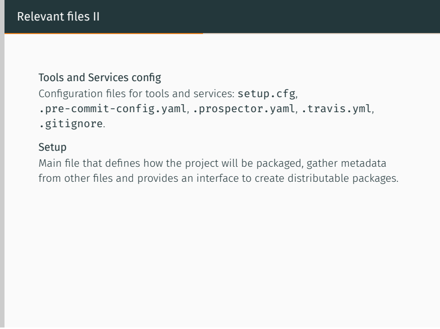
Relevant files II

Tools and Services config

Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

Setup

Main file that defines how the project will be packaged, gather metadata from other files and provides an interface to create distributable packages.



Relevant files II

Tools and Services config
Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

Setup
Main file that defines how the project will be packaged, gather metadata from other files and provides an interface to create distributable packages.

Main configuration file: `setup.cfg`. Ini-style file with sections for configuration of different tools, like *bumpversion*, *nose* and *coverage*.

Setup file keeps the requirements list, the current version, application name...

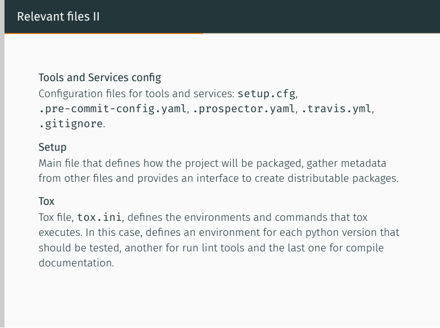
Tox is integrated with travis.

Relevant files II

Tools and Services config
Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

Setup
Main file that defines how the project will be packaged, gather metadata from other files and provides an interface to create distributable packages.

Tox
Tox file, `tox.ini`, defines the environments and commands that tox executes. In this case, defines an environment for each python version that should be tested, another for run lint tools and the last one for compile documentation.


Relevant files II

Tools and Services config
Configuration files for tools and services: `setup.cfg`, `.pre-commit-config.yaml`, `.prospector.yaml`, `.travis.yml`, `.gitignore`.

Setup
Main file that defines how the project will be packaged, gather metadata from other files and provides an interface to create distributable packages.

Tox
Tox file, `tox.ini`, defines the environments and commands that tox executes. In this case, defines an environment for each python version that should be tested, another for run lint tools and the last one for compile documentation.

Main configuration file: `setup.cfg`. Ini-style file with sections for configuration of different tools, like *bumpversion*, *nose* and *coverage*.

Setup file keeps the requirements list, the current version, application name...

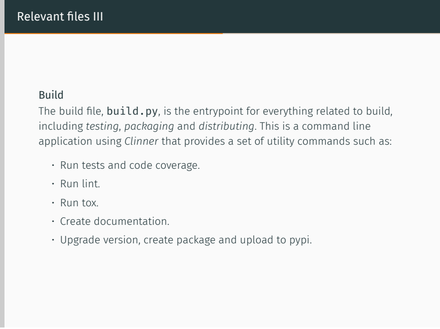
Tox is integrated with travis.

Relevant files III

Build

The build file, `build.py`, is the entrypoint for everything related to build, including *testing*, *packaging* and *distributing*. This is a command line application using *Clinner* that provides a set of utility commands such as:

- Run tests and code coverage.
- Run lint.
- Run tox.
- Create documentation.
- Upgrade version, create package and upload to pypi.

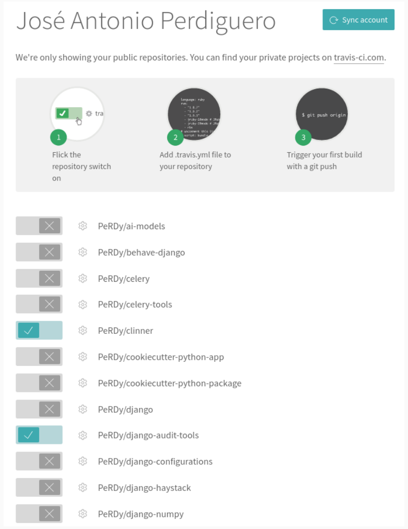


Relevant files III

Build
The build file, `build.py`, is the entrypoint for everything related to build, including *testing*, *packaging* and *distributing*. This is a command line application using *Clinner* that provides a set of utility commands such as:

- Run tests and code coverage.
- Run lint.
- Run tox.
- Create documentation.
- Upgrade version, create package and upload to pypi.

Bind Travis



José Antonio Perdiguero

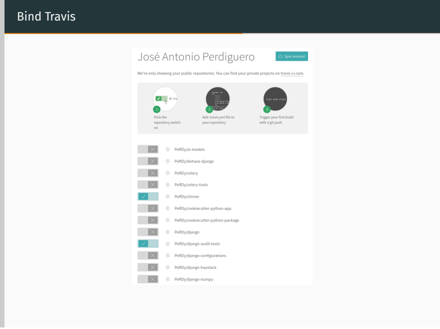
We're only showing your public repositories. You can find your private projects on travis-ci.com.

1. Flick the repository switch on
2. Add .travis.yml file to your repository
3. Trigger your first build with a git push

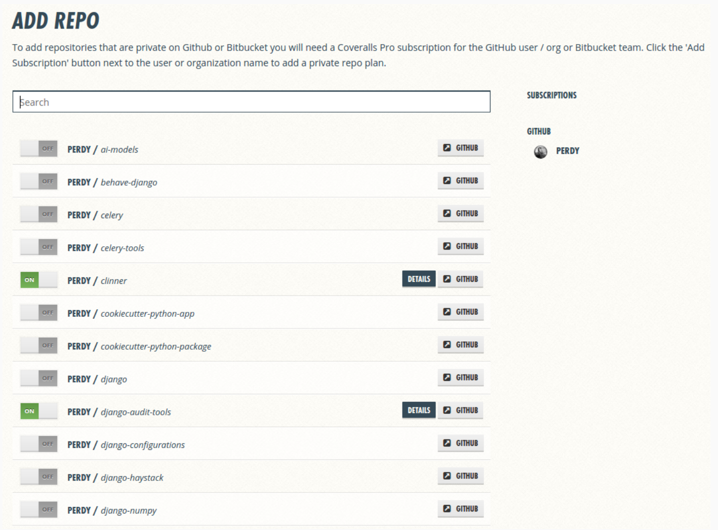
✗	PeRDy/ai-models
✗	PeRDy/behave-django
✗	PeRDy/celery
✗	PeRDy/celery-tools
✓	PeRDy/clinner
✗	PeRDy/cookiecutter-python-app
✗	PeRDy/cookiecutter-python-package
✗	PeRDy/django
✓	PeRDy/django-audit-tools
✗	PeRDy/django-configurations
✗	PeRDy/django-haystack
✗	PeRDy/django-numpy

2017-04-19

Packaging and Distributing
└ Creation
  └ Bind services
    └ Bind Travis



Bind Coveralls



ADD REPO

To add repositories that are private on Github or Bitbucket you will need a Coveralls Pro subscription for the GitHub user / org or Bitbucket team. Click the 'Add Subscription' button next to the user or organization name to add a private repo plan.

Search

OFF	PERDY / *ai-models*		GITHUB
OFF	PERDY / *behave-django*		GITHUB
OFF	PERDY / *celery*		GITHUB
OFF	PERDY / *celery-tools*		GITHUB
ON	PERDY / *clinner*	DETAILS	GITHUB
OFF	PERDY / *cookiecutter-python-app*		GITHUB
OFF	PERDY / *cookiecutter-python-package*		GITHUB
OFF	PERDY / *django*		GITHUB
ON	PERDY / *django-audit-tools*	DETAILS	GITHUB
OFF	PERDY / *django-configurations*		GITHUB
OFF	PERDY / *django-haystack*		GITHUB
OFF	PERDY / *django-numpy*		GITHUB

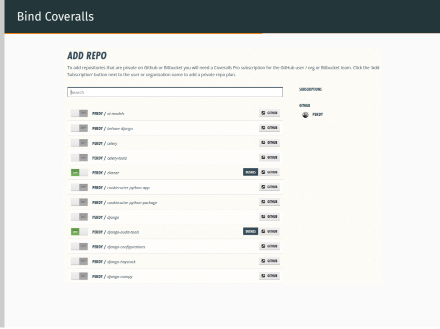
SUBSCRIPTIONS

GITHUB

PERDY

2017-04-19

Packaging and Distributing
└ Creation
  └ Bind services
    └ Bind Coveralls

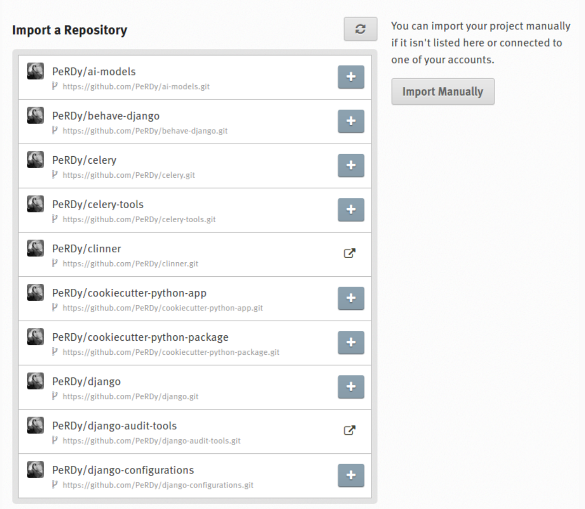


Bind Coveralls

ADD REPO

To add repositories that are private on Github or Bitbucket you will need a Coveralls Pro subscription for the GitHub user / org or Bitbucket team. Click the 'Add Subscription' button next to the user or organization name to add a private repo plan.

Bind ReadTheDocs

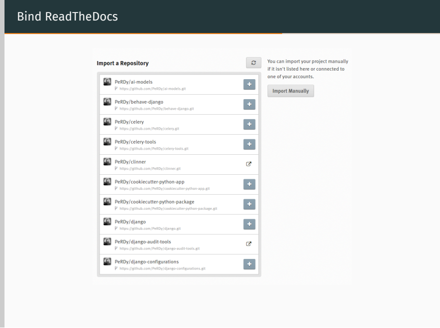

Import a Repository

- PeRDy/ai-models
  https://github.com/PeRDy/ai-models.git
- PeRDy/behave-django
  https://github.com/PeRDy/behave-django.git
- PeRDy/celery
  https://github.com/PeRDy/celery.git
- PeRDy/celery-tools
  https://github.com/PeRDy/celery-tools.git
- PeRDy/clinner
  https://github.com/PeRDy/clinner.git
- PeRDy/cookiecutter-python-app
  https://github.com/PeRDy/cookiecutter-python-app.git
- PeRDy/cookiecutter-python-package
  https://github.com/PeRDy/cookiecutter-python-package.git
- PeRDy/django
  https://github.com/PeRDy/django.git
- PeRDy/django-audit-tools
  https://github.com/PeRDy/django-audit-tools.git
- PeRDy/django-configurations
  https://github.com/PeRDy/django-configurations.git

You can import your project manually if it isn't listed here or connected to one of your accounts.

2017-04-19

Packaging and Distributing
└ Creation
  └ Bind services
    └ Bind ReadTheDocs



Packaging

Packaging and Distributing
└ Packaging

2017-04-19

Packaging

Test your application

Development

Run tests while developing using nose.

```
python build.py test
```

Test your application

Development

Run tests while developing using nose.

python build.py test

Test your application

Development

Run tests while developing using nose.

```
python build.py test
```

Multi-environment

Once development is done, run tests against all different interpreters supported to check compatibility.

```
python build.py tox
```

2017-04-19

Test your application

Development
Run tests while developing using nose.

python build.py test

Multi-environment
Once development is done, run tests against all different interpreters supported to check compatibility.

python build.py tox

Test your application

Development

Run tests while developing using nose.

```
python build.py test
```

Multi-environment

Once development is done, run tests against all different interpreters supported to check compatibility.

```
python build.py tox
```

Continuous Integration

Development is done, tests pass in every environment, so code can be uploaded to repository safely. Once a commit is done:

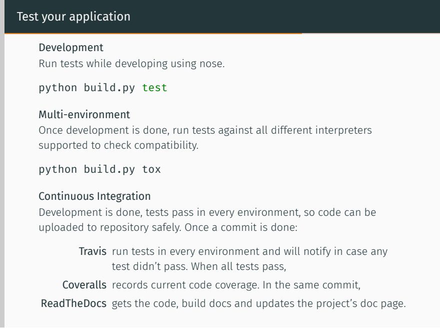
Travis run tests in every environment and will notify in case any test didn't pass. When all tests pass,

Coveralls records current code coverage. In the same commit,

ReadTheDocs gets the code, build docs and updates the project's doc page.

2017-04-19

Packaging and Distributing
└Packaging
  └Test your application
    └Test your application


Test your application

Development
Run tests while developing using nose.

python build.py test

Multi-environment
Once development is done, run tests against all different interpreters supported to check compatibility.

python build.py tox

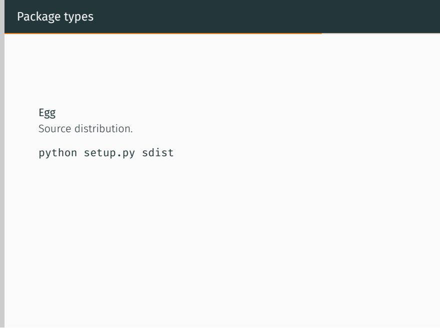
Continuous Integration
Development is done, tests pass in every environment, so code can be uploaded to repository safely. Once a commit is done:

Travis run tests in every environment and will notify in case any test didn't pass. When all tests pass,
Coveralls records current code coverage. In the same commit,
ReadTheDocs gets the code, build docs and updates the project's doc page.

Package types

Egg
Source distribution.

```
python setup.py sdist
```


Package types

Egg
Source distribution.

python setup.py sdist

There is two types of python packages: *egg* and *wheel*. Source distribution needs a build step, built or binary distributions only need to move the package in the right path. Case of *numpy*, *scipy* and *pandas*.

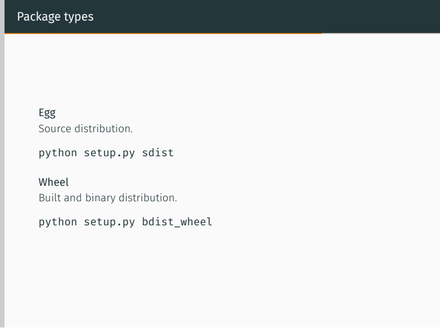
Package types

Egg
Source distribution.

```
python setup.py sdist
```

Wheel
Built and binary distribution.

```
python setup.py bdist_wheel
```


Package types

Egg
Source distribution.

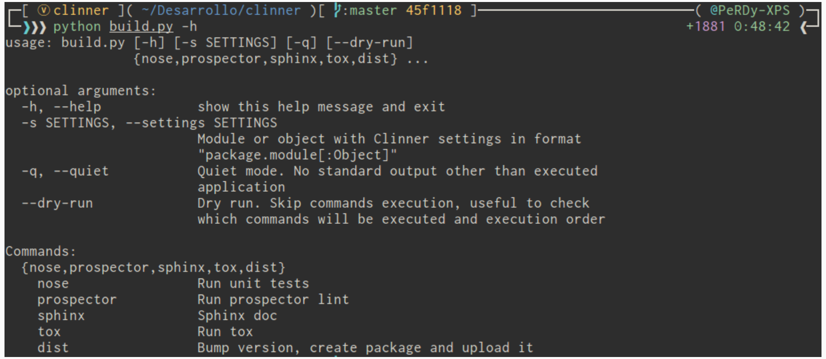
python setup.py sdist

Wheel
Built and binary distribution.

python setup.py bdist_wheel

There is two types of python packages: *egg* and *wheel*. Source distribution needs a build step, built or binary distributions only need to move the package in the right path. Case of *numpy*, *scipy* and *pandas*.

Builder



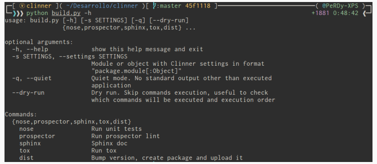
```
┌[ ⓥclinner ]( ~/Desarrollo/clinner )[ :master 45f1118 ]────────────────( @PeRDy-XPS )┐
└❯❯❯ python build.py -h                                              +1881 0:48:42 ⮐┘
usage: build.py [-h] [-s SETTINGS] [-q] [--dry-run]
                {nose,prospector,sphinx,tox,dist} ...

optional arguments:
  -h, --help            show this help message and exit
  -s SETTINGS, --settings SETTINGS
                        Module or object with Clinner settings in format
                        "package.module[:Object]"
  -q, --quiet           Quiet mode. No standard output other than executed
                        application
  --dry-run             Dry run. Skip commands execution, useful to check
                        which commands will be executed and execution order

Commands:
  {nose,prospector,sphinx,tox,dist}
    nose                Run unit tests
    prospector          Run prospector lint
    sphinx              Sphinx doc
    tox                 Run tox
    dist                Bump version, create package and upload it
```

2017-04-19

Builder



Distributing

Packaging and Distributing

Distributing

2017-04-19

Distributing

Register the application

PyPI account

Create a PyPI[1] account. Configure `.pypirc` file with PyPI credentials.

[1] https://pypi.python.org/pypi
[2] https://github.com/pypa/twine

Register the application

PyPI account

Create a PyPI[1] account. Configure `.pypirc` file with PyPI credentials.

[1] https://pypi.python.org/pypi
[2] https://github.com/pypa/twine

Register the application

PyPI account

Create a PyPI[1] account. Configure `.pypirc` file with PyPI credentials.

Application register

Use twine[2] with a package of your application to register it in PyPI:

```
twine register dist/project-name.whl
```

[1] https://pypi.python.org/pypi

[2] https://github.com/pypa/twine

2017-04-19

```
Packaging and Distributing
└─Distributing
   └─Register the application
      └─Register the application
```

Register the application

PyPI account
Create a PyPI[1] account. Configure .pypirc file with PyPI credentials.
Application register
Use twine[2] with a package of your application to register it in PyPI:
twine register dist/project-name.whl

[1] https://pypi.python.org/pypi
[2] https://github.com/pypa/twine

Upload a new version

Upload packages

Use twine again to upload all packages to PyPI:

```
twine upload dist/project-name.whl
twine upload dist/project-name.tar.gz
```

2017-04-19

Packaging and Distributing
- Distributing
  - Upload a new version
    - Upload a new version

Upload a new version

Upload packages

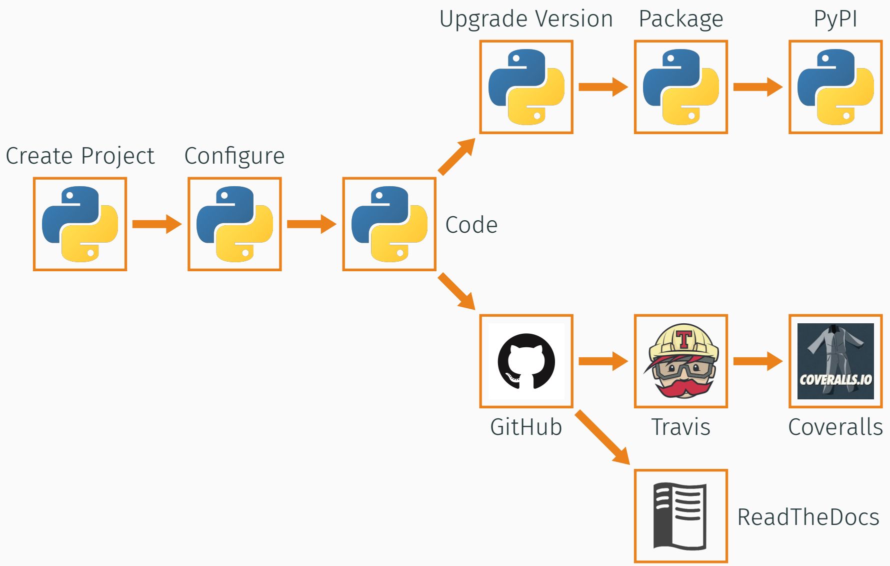
Use twine again to upload all packages to PyPI:

twine upload dist/project-name.whl
twine upload dist/project-name.tar.gz

Conclusion

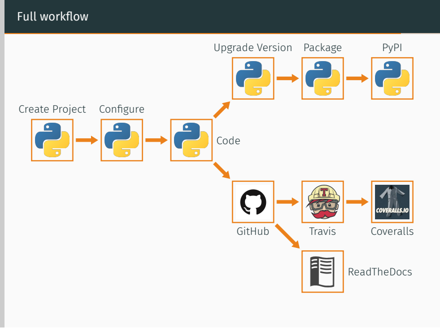
Conclusion

Full workflow

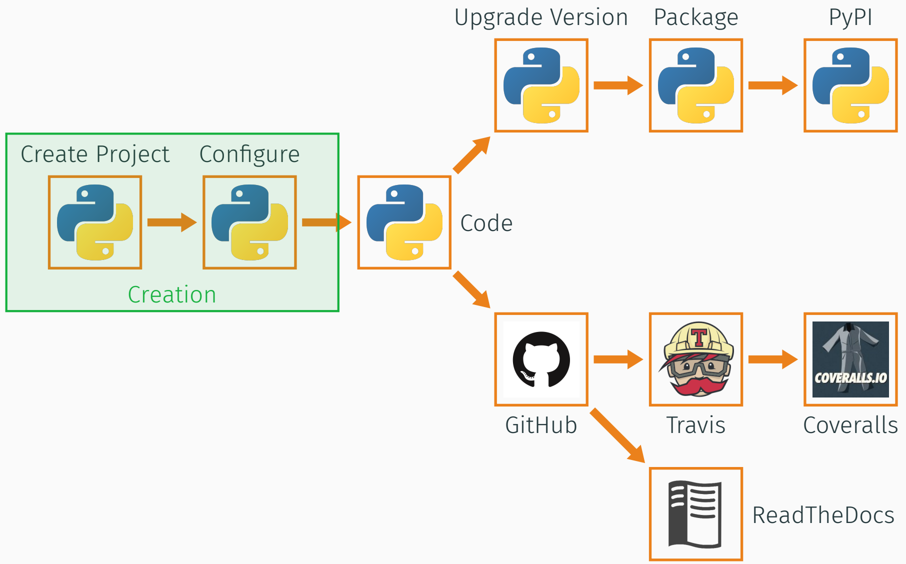


2017-04-19

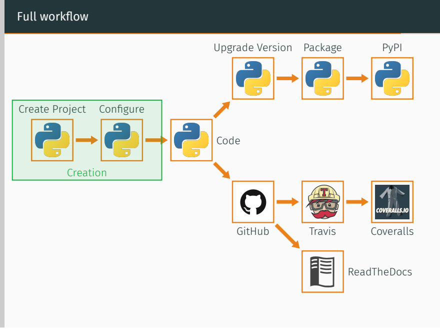
Packaging and Distributing
- Conclusion
  - Full workflow
    - Full workflow



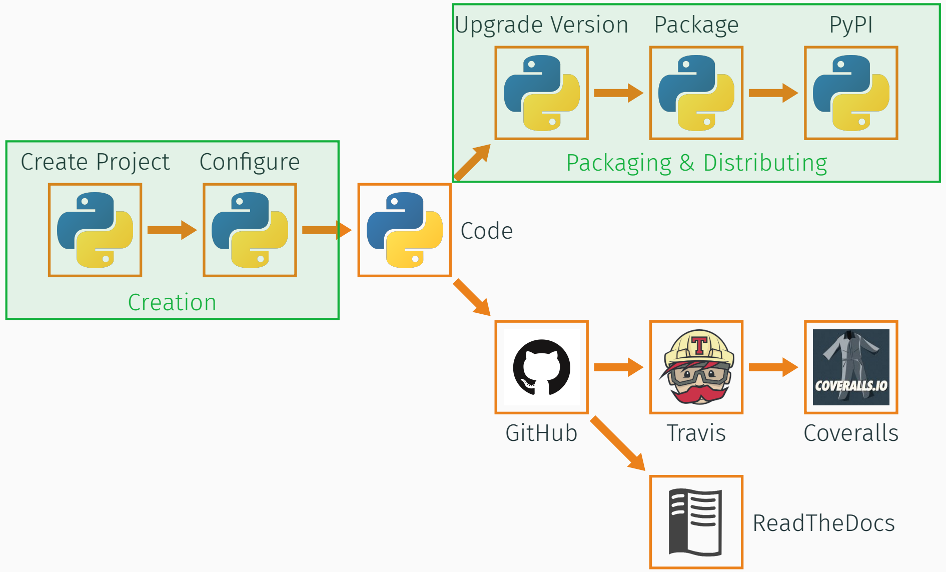
Full workflow



2017-04-19

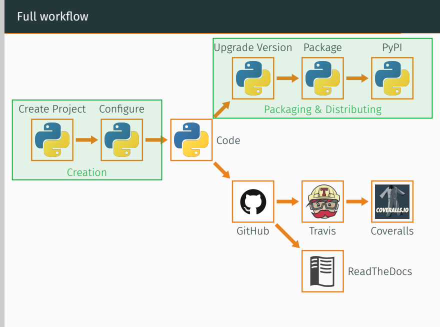


Full workflow



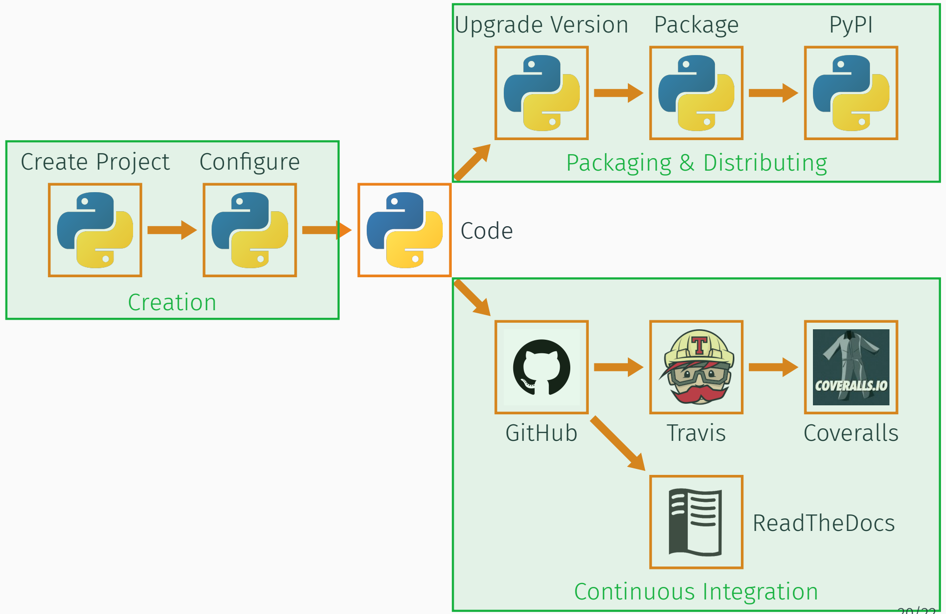
2017-04-19

Packaging and Distributing
Conclusion
Full workflow
Full workflow

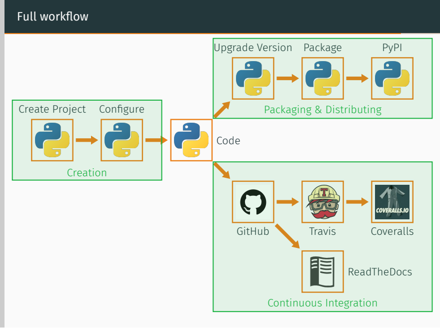


Full workflow

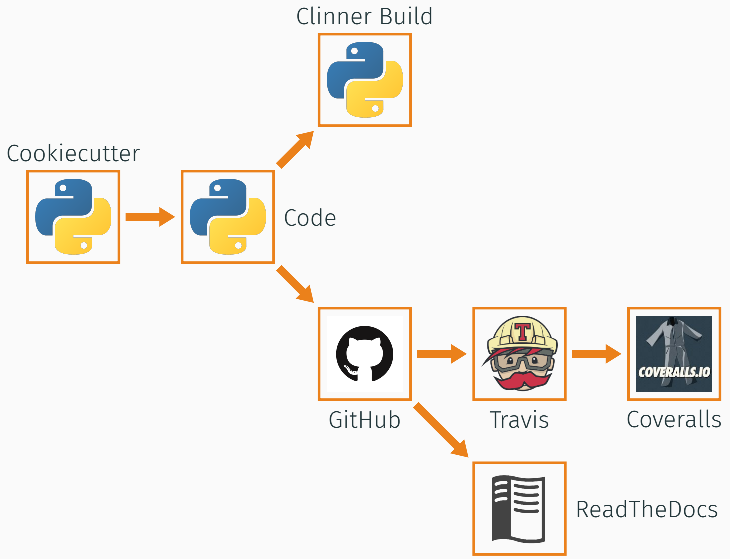
Full workflow



2017-04-19

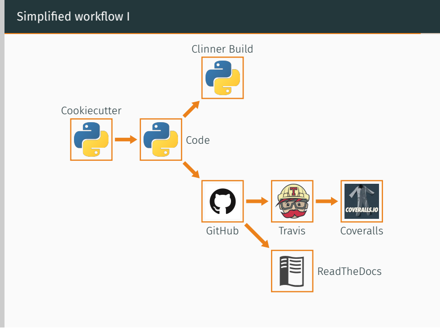


Simplified workflow I



2017-04-19

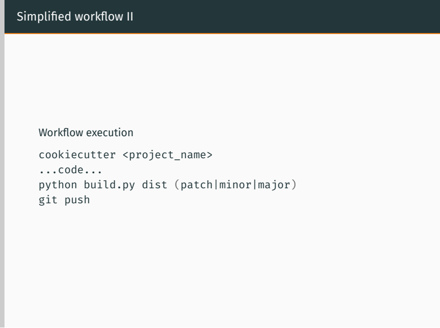
- Packaging and Distributing
  - Conclusion
    - Simplified workflow
      - Simplified workflow I



Simplified workflow II

Workflow execution

```
cookiecutter <project_name>
...code...
python build.py dist (patch|minor|major)
git push
```



Simplified workflow II

Workflow execution

```
cookiecutter <project_name>
...code...
python build.py dist (patch|minor|major)
git push
```

Open source your code !

Open source your code !