Package software for any distribution with upt

Cyril Roelandt

Introduction

- Cyril Roelandt
- OpenStack (Red Hat)
- GNU Guix (Do not miss Ludovic's talk at 4:30PM!)

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

What is packaging?

- Turn an “upstream” package into a “downstream” package

What is packaging?

- Turn an “upstream” package into a “downstream” package
- Debugging

What is packaging?

- Turn an “upstream” package into a “downstream” package
- Debugging
- Send/maintain patches

Is packaging hard?

- Many sources (PyPI, RubyGems. . .)

Is packaging hard?

- Many sources (PyPI, RubyGems...)
- Many targets (Debian, Fedora, Guix...)

Is packaging hard?

- Many sources (PyPI, RubyGems...)
- Many targets (Debian, Fedora, Guix...)
- Cannot be fully automated:

Is packaging hard?

- Many sources (PyPI, RubyGems. . .)
- Many targets (Debian, Fedora, Guix. . .)
- Cannot be fully automated:
  - Dependency hell

Is packaging hard?

- Many sources (PyPI, RubyGems. . .)
- Many targets (Debian, Fedora, Guix. . .)
- Cannot be fully automated:
  - Dependency hell
  - Debugging

Is packaging hard?

- Many sources (PyPI, RubyGems. . .)
- Many targets (Debian, Fedora, Guix. . .)
- Cannot be fully automated:
  - Dependency hell
  - Debugging
- Some tasks can be automated:

Is packaging hard?

- Many sources (PyPI, RubyGems...)
- Many targets (Debian, Fedora, Guix...)
- Cannot be fully automated:
  - Dependency hell
  - Debugging
- Some tasks can be automated:
  - Package layout

Is packaging hard?

- Many sources (PyPI, RubyGems. . .)
- Many targets (Debian, Fedora, Guix. . .)
- Cannot be fully automated:
  - Dependency hell
  - Debugging
- Some tasks can be automated:
  - Package layout
  - Metadata

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

Existing tools

	Debian	Fedora	Guix	FreeBSD	OpenBSD
CPAN	dh-make-perl	cpan2rpm	guix import	?	PortGen
NPM	npm2deb	npm2rpm	N/A	?	?
PyPI	pypi2deb	pyp2rpm	guix import	pytoport	PortGen
Ruby	gem2deb	gem2rpm	guix import	?	PortGen

Issues

- One CLI per tool

Issues

- One CLI per tool
- Different behaviours

Issues

- One CLI per tool
- Different behaviours
- Code duplication (read upstream archive)

Issues

- One CLI per tool
- Different behaviours
- Code duplication (read upstream archive)
- Code duplication (write downstream package)

Example: licenses (PyPI)

- SPDX identifiers?

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License
  - BSD2 or BSD3 or . . . ?

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License
  - BSD2 or BSD3 or . . . ?
  - https://github.com/pypa/warehouse/issues/2996

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License
  - BSD2 or BSD3 or . . . ?
  - https://github.com/pypa/warehouse/issues/2996
- “license” field (setup.{cfg,py})

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License
  - BSD2 or BSD3 or . . . ?
  - https://github.com/pypa/warehouse/issues/2996
- "license" field (setup.{cfg,py})
  - For values not available in the standard classifiers

Example: licenses (PyPI)

- SPDX identifiers?
- Classifiers:
  - License :: OSI Approved :: BSD License
  - BSD2 or BSD3 or . . . ?
  - https://github.com/pypa/warehouse/issues/2996
- "license" field (setup.{cfg,py})
  - For values not available in the standard classifiers
  - 'BSD2', 'BSD 2', 'BSD-2', 'BSD 2 Clause'. . .

Example: dependencies (PyPI)

pipenv

```
$ curl -s https://pypi.org/pypi/pipenv/json \
    | jq .info.requires_dist
[
  "ordereddict; python_version < \"2.7\"",
  "requests[security]; python_version < \"2.7\"",
  "virtualenv",
  "virtualenv-clone (>=0.2.5)",
  "setuptools (>=36.2.1)",
  "certifi",
  "pip (>=9.0.1)"
]
```

Example: dependencies (PyPI)

botocore

```
$ curl -s https://pypi.org/pypi/botocore/json \
    | jq .info.requires_dist
null
```

Example: dependencies (PyPI)

botocore

```
$ curl -s https://pypi.org/pypi/botocore/json \
    | jq .info.requires_dist
null
$ wget https://.../botocore-1.12.16-py2.py3-none-any.whl
$ unzip botocore-1.12.16-py2.py3-none-any.whl
$ cd botocore-1.12.16.dist-info/
$ jq .run_requires metadata.json
[
  {
    "requires": [
      "jmespath>=0.7.1,<1.0.0",
      "docutils>=0.10",
      "urllib3>=1.20,<1.24"
    ]
  },
  ...
```

Example: dependencies (PyPI)

MarkupSafe

```
$ curl -s https://pypi.org/pypi/MarkupSafe/json \
    | jq .info.requires_dist
null
```

Example: dependencies (PyPI)

MarkupSafe

```
$ curl -s https://pypi.org/pypi/MarkupSafe/json \
    | jq .info.requires_dist
null
$ pip download --only-binary :all: MarkupSafe
Collecting MarkupSafe
  Could not find a version that satisfies the \
  requirement MarkupSafe (from versions: )
No matching distribution found for MarkupSafe
```

Example: dependencies (PyPI)

MarkupSafe

```
$ curl -s https://pypi.org/pypi/MarkupSafe/json \
    | jq .info.requires_dist
null
$ pip download --only-binary :all: MarkupSafe
Collecting MarkupSafe
  Could not find a version that satisfies the \
  requirement MarkupSafe (from versions: )
No matching distribution found for MarkupSafe
$ cat setup.py
...
extras_require={
    'dev': ...
    'docs': ...
}
```

Example: dependencies (PyPI)

- Read the JSON file provided by PyPI
- Read the wheel
- Read the source code
- Install the package, run “pip list”
- What about test dependencies?
- There might be even more ideas!

What we really need

- Unified CLI

What we really need

- Unified CLI
- Unified behaviour

What we really need

- Unified CLI
- Unified behaviour
- Modular

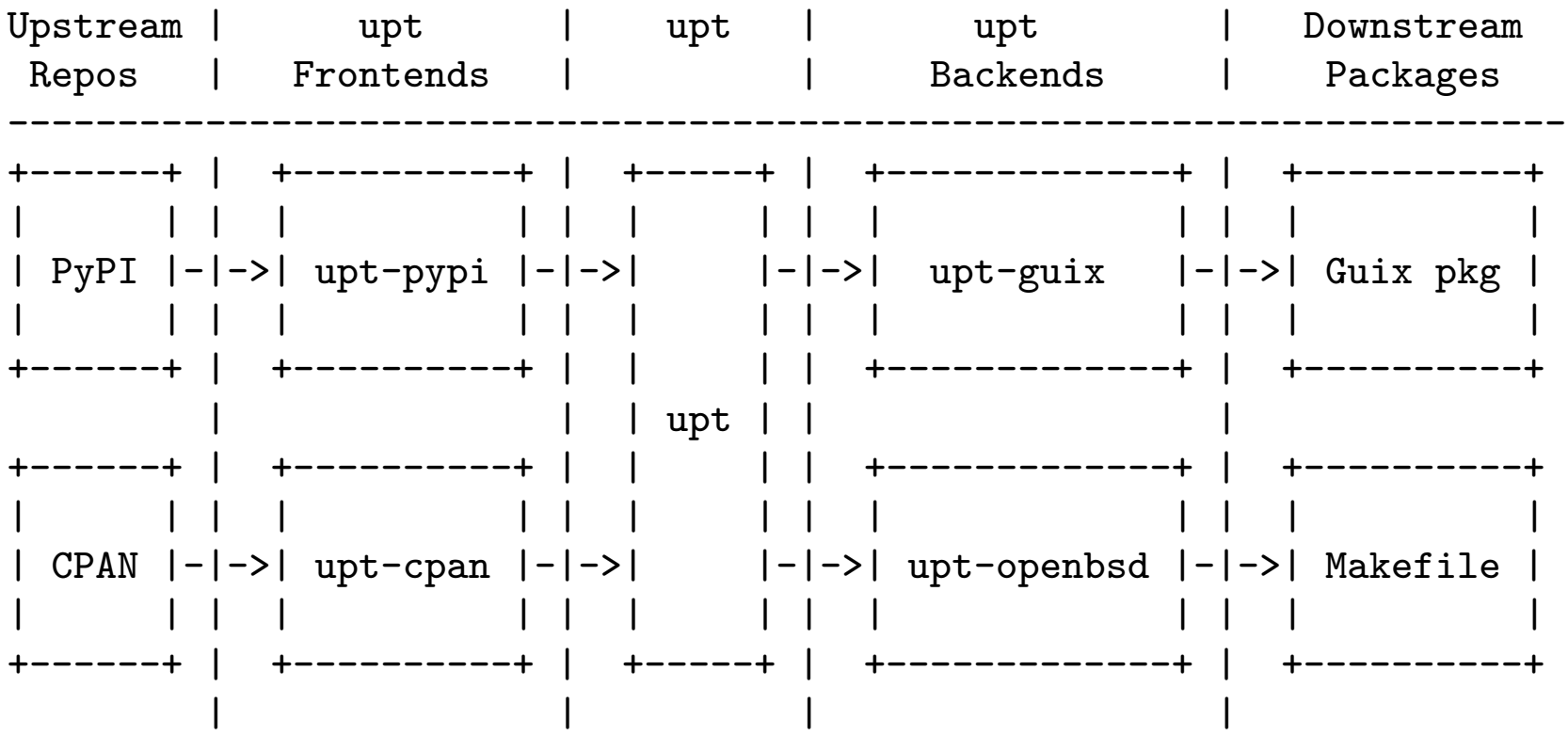
What we really need

- Unified CLI
- Unified behaviour
- Modular
- Simple

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

Architecture



Usage

Install (with modules)

```
$ pip install upt[frontends,backends]
```

Usage

Install (with modules)

```
$ pip install upt[frontends,backends]
```

Available frontends

```
$ upt list-frontends
cpan
pypi
rubygems
```

Usage

Install (with modules)

```
$ pip install upt[frontends,backends]
```

Available frontends

```
$ upt list-frontends
cpan
pypi
rubygems
```

Available backends

```
$ upt list-backends
fedora
freebsd
guix
nix
openbsd
```

Usage

Package upt for OpenBSD

```
$ upt package -f pypi -b openbsd -o /tmp/py-upt upt
[INFO    ] [Backend] Creating the directory \
                     structure in /tmp/py-upt
[INFO    ] [Backend] Creating the Makefile
[INFO    ] [Backend] Creating pkg/DESCR
[INFO    ] [Backend] You still need to create \
                     /tmp/py-upt/distinfo
[INFO    ] [Backend] You still need to create \
                     /tmp/py-upt/pkg/PLIST
```

Usage

$ less /tmp/py-upt/Makefile

```
# $OpenBSD$
COMMENT=                    package software from any package m
MODPY_EGG_VERSION=          0.4.1
DISTNAME=                   upt-${MODPY_EGG_VERSION}
PKGNAME=                    py-upt-${MODPY_EGG_VERSION}
CATEGORIES=                 XXX
HOMEPAGE=                   https://framagit.org/upt/upt
MAINTAINER=                 XXX <xxx@xxx.xxx>
# BSD-3-Clause
PERMIT_PACKAGE_CDROM =      Yes
MODULES=                    lang/python
MODPY_SETUPTOOLS=           Yes
MODPY_PI=                   Yes
RUN_DEPENDS=                xxx/py-spdx-lookup${MODPY_FLAVOR}
.include <bsd.port.mk>
```

Usage

$ upt package -f pypi -b nix upt

```
{ stdenv, buildPythonPackage, fetchPypi
, spdx-lookup }:
buildPythonPackage rec {
  name = "${pname}-${version}";
  version = "0.4.1";
  pname = "upt";
  src = fetchPypi {
    inherit pname version;
    sha256 = "TODO";
  };
  propagatedBuildInputs = [ spdx-lookup ];
  meta = with stdenv.lib; {
    description = ...
    homepage = https://framagit.org/upt/upt;
    license = [ licenses.bsd3 ];
  }
```

Modularity

- One module (backend or frontend) -> one Git repo

Modularity

- One module (backend or frontend) -> one Git repo
- Easier for “small” projects

Modularity

- One module (backend or frontend) -> one Git repo
- Easier for “small” projects
- More independence

Add a backend or frontend

$ cookiecutter https://framagit.org/upt/cookiecutter-upt

```
project_name []: upt-archlinux
project_slug [upt_archlinux]:
author []: John Doe
email []: john@doe.org
short_description []: Arch Linux backend for upt.
url [https://framagit.org/upt/upt-archlinux]:
Select license:
1 - BSD-3
2 - other
Choose from 1, 2 (1, 2) [1]: 1
Select kind:
1 - backend
2 - frontend
Choose from 1, 2 (1, 2) [1]: 1
Initialized empty Git repository in /tmp/upt-archlinux/.git
```

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

A different approcah

- Ruby

A different approcah

- Ruby
- Modules are distributed along with the core

A different approcah

- Ruby
- Modules are distributed along with the core
- Generates a .rpm instead of a .spec (or .deb instead of a debian/ folder)

A different approcah

- Ruby
- Modules are distributed along with the core
- Generates a .rpm instead of a .spec (or .deb instead of a debian/ folder)
- Useful for users who need to use their distro's package manager

A different approcah

- Ruby
- Modules are distributed along with the core
- Generates a .rpm instead of a .spec (or .deb instead of a debian/ folder)
- Useful for users who need to use their distro's package manager
- Not the best tool for package maintainers?

Outline

1. Packaging
2. Automation
3. The Universal Packaging Tool
4. fpm
5. Conclusion

Future work

- More modules (written by you?)

Future work

- More modules (written by you?)
- Type hints

Future work

- More modules (written by you?)
- Type hints
- Need a standard format for all upstream package archives

Join me!

- Code: https://framagit.org/upt
- Emails: cyril.roelandt@aquilenet.fr, cyril@redhat.com
- Freenode: #upt-packaging, Steap
- Questions?