fpm Documentation

Release 1.9.0

Jordan Sissel

Sep 08, 2017

Contents

1 Backstory **3**

2 The Solution - FPM **5**

3 Things that should work **7**

4 Table of Contents **9**

4.1 What is FPM? 9
4.2 Installation 10
4.3 Use Cases 11
4.4 Packages 14
4.5 Want to contribute? Or need help? 24
4.6 Release Notes and Change Log 25

Note: The documentation here is a work-in-progress. If you want to contribute new docs or report problems, I invite you to do so on the project issue tracker.

The goal of fpm is to make it easy and quick to build packages such as rpms, debs, OSX packages, etc.

fpm, as a project, exists with the following principles in mind:

- If fpm is not helping you make packages easily, then there is a bug in fpm.
- If you are having a bad time with fpm, then there is a bug in fpm.
- If the documentation is confusing, then this is a bug in fpm.

If there is a bug in fpm, then we can work together to fix it. If you wish to report a bug/problem/whatever, I welcome you to do on the project issue tracker.

You can find out how to use fpm in the documentation.

You can learn how to install fpm on your platform in the installation guide.

CHAPTER 1

Backstory

Sometimes packaging is done wrong (because you can't do it right for all situations), but small tweaks can fix it.

And sometimes, there isn't a package available for the tool you need.

And sometimes if you ask "How do I get python 3 on CentOS 5?" some unhelpful trolls will tell you to "Use another distro"

Further, a job switches have me flipping between Ubuntu and CentOS. These use two totally different package systems with completely different packaging policies and support tools. Learning both was painful and confusing. I want to save myself (and you) that pain in the future.

It should be easy to say "here's my install dir and here's some dependencies; please make a package"

CHAPTER 2

The Solution - FPM

I wanted a simple way to create packages without needing to memorize too much.

I wanted a tool to help me deliver software with minimal steps or training.

The goal of FPM is to be able to easily build platform-native packages.

With fpm, you can do many things, including:

- Creating packages easily (deb, rpm, freebsd, etc)
- Tweaking existing packages (removing files, changing metadata/dependencies)
- Stripping pre/post/maintainer scripts from packages

CHAPTER 3

Things that should work

Sources:

- gem (even autodownloaded for you)
- python modules (autodownload for you)
- pear (also downloads for you)
- directories
- tar(.gz) archives
- rpm
- deb
- node packages (npm)
- pacman (ArchLinux) packages

Targets:

- deb
- rpm
- solaris
- freebsd
- tar
- directories
- Mac OS X *.pkg* files (*osxpkg*)
- pacman (ArchLinux) packages

CHAPTER 4

Table of Contents

What is FPM?

fpm is a command-line program designed to help you build packages.

Building a package might look something like this:

> fpm -s <source type> -t <target type> [list of sources]...

"Source type" is what your package is coming from; a directory (dir), a rubygem (gem), an rpm (rpm), a python package (python), a php pear module (pear), etc.

"Target type" is what your output package form should be. Most common are "rpm" and "deb" but others exist (solaris, etc)

You have a few options for learning to run FPM:

1. If you're impatient, just scan through *fpm –help*; you'll need various options, and we try to make them well-documented. Quick learning is totally welcome, and if you run into issues, you are welcome to ask questions in #fpm on freenode irc or on fpm-users@googlegroups.com!
2. The documentation has explanations and examples. If you run into problems, I welcome you to ask questions in #fpm on freenode irc or on fpm-users@googlegroups.com!

To give you an idea of what fpm can do, here's a few use cases:

Take a directory and turn it into an RPM:: fpm -s dir -t rpm ...

Convert a .deb into an rpm:: fpm -s deb -t rpm ...

Convert a rubygem into a deb package:: fpm -s gem -t deb ...

Convert a .tar.gz into an OSX .pkg file:: fpm -s tar -t osxpkg

Convert a .zip into an rpm:: fpm -s zip -t rpm ...

Change properties of an existing rpm:: fpm -s rpm -t rpm

Create an deb that automatically installs a service:: fpm -s pleaserun -t deb

Below is a 10-minute video demonstrating fpm's simplicity of use:

Now that you've seen a bit of what fpm can do, it's time to *install fpm*.

Installation

FPM requires a few things before you can use it. This document will show you how to install all the necessary things :)

Depending on what you want to do with FPM, you might need some extra things installed (like tooling to build rpms, or solaris packages, or something else), but for now, let's just get ruby so we can start using fpm!

Installing things FPM needs

> **Warning:** This section may be imperfect due to the inconsistencies across OS vendors

fpm is written in Ruby, you'll need to provide Ruby. Some operating systems, like OSX, come with Ruby already, but some do not. Depending on your operating system, you might need to run the following commands:

On OSX/macOS:

```
brew install gnu-tar
```

On Red Hat systems (Fedora 22 or older, CentOS, etc):

```
yum install ruby-devel gcc make rpm-build rubygems
```

On Fedora 23 or newer:

```
dnf install ruby-devel gcc make rpm-build libffi-devel
```

On Debian-derived systems (Debian, Ubuntu, etc):

```
apt-get install ruby ruby-dev rubygems build-essential
```

Installing FPM

You can install fpm with the *gem* tool:

```
gem install --no-ri --no-rdoc fpm
```

Note: *gem* is a command provided by a the Ruby packaging system called rubygems. This allows you to install, and later upgrade, fpm.

You should see output that looks like this:

```
% gem install --no-ri --no-rdoc fpm
Fetching: cabin-0.9.0.gem (100%)
Successfully installed cabin-0.9.0
Fetching: backports-3.6.8.gem (100%)
```

```
Successfully installed backports-3.6.8
Fetching: arr-pm-0.0.10.gem (100%)
Successfully installed arr-pm-0.0.10
Fetching: clamp-1.0.1.gem (100%)
Successfully installed clamp-1.0.1
Fetching: ffi-1.9.14.gem (100%)
Building native extensions.  This could take a while...
Successfully installed ffi-1.9.14
Fetching: childprocess-0.5.9.gem (100%)
Successfully installed childprocess-0.5.9
Fetching: archive-tar-minitar-0.5.2.gem (100%)
Successfully installed archive-tar-minitar-0.5.2
Fetching: io-like-0.3.0.gem (100%)
Successfully installed io-like-0.3.0
Fetching: ruby-xz-0.2.3.gem (100%)
Successfully installed ruby-xz-0.2.3
Fetching: dotenv-2.1.1.gem (100%)
Successfully installed dotenv-2.1.1
Fetching: insist-1.0.0.gem (100%)
Successfully installed insist-1.0.0
Fetching: mustache-0.99.8.gem (100%)
Successfully installed mustache-0.99.8
Fetching: stud-0.0.22.gem (100%)
Successfully installed stud-0.0.22
Fetching: pleaserun-0.0.27.gem (100%)
Successfully installed pleaserun-0.0.27
Fetching: fpm-1.6.3.gem (100%)
Successfully installed fpm-1.6.3
15 gems installed
```

Now you should be ready to use fpm!

To make sure fpm is installed correctly, try running the following command:

```
fpm --version
```

You should get some output like this, although the exact output will depend on which version of fpm you have installed.:

```
% fpm --version
1.6.3
```

Use Cases

Jenkins: Single-file package

For this example, you'll learn how to package hudson/jenkins which is a single-file download.

We'll use *make* to script the download, but *make* isn't required if you don't want it.

Makefile:

```
NAME=jenkins
VERSION=1.396

.PHONY: package
```

```
package:
  rm -f jenkins.war
  wget http://ftp.osuosl.org/pub/hudson/war/$(VERSION)/jenkins.war
  fpm -s dir -t deb -n $(NAME) -v $(VERSION) --prefix /opt/jenkins jenkins.war
```

Note: You'll need *wget* for this Makefile to work.

Running it:

```
% make
rm -f jenkins.war
wget http://ftp.osuosl.org/pub/hudson/war/1.396/jenkins.war
--2011-02-07 17:56:01--  http://ftp.osuosl.org/pub/hudson/war/1.396/jenkins.war
Resolving ftp.osuosl.org... 140.211.166.134
Connecting to ftp.osuosl.org|140.211.166.134|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 36665038 (35M) [text/plain]
Saving to: `jenkins.war'

100%[======================================================>] 36,665,038  3.88M/s   in␣
↪10s

2011-02-07 17:56:11 (3.37 MB/s) - `jenkins.war' saved [36665038/36665038]

% fpm -s dir -t deb -n jenkins -v 1.396 --prefix /opt/jenkins -d "sun-java6-jre (> 0)
↪" jenkins.war
Created .../jenkins-1.396-1.amd64.deb
```

Delicious.

nodejs and multiple packages

This example requires your *make install* support setting DESTDIR or otherwise allow you to install to a specific target directory.

Consider building nodejs. Sometimes you want to produce multiple packages from a single project. In this case, building three separate packages: nodejs, nodejs-dev, and nodejs-doc.

Package up the nodejs runtime

Normal build steps:

```
# Normal build steps.
% wget http://nodejs.org/dist/v0.6.0/node-v0.6.0.tar.gz
% tar -zxf node-v0.6.0.tar.gz
% cd node-v0.6.0
% ./configure --prefix=/usr
% make
```

Now install it to a temporary directory:

```
# Install to a separate directory for capture.
% mkdir /tmp/installdir
% make install DESTDIR=/tmp/installdir
```

Now make the 'nodejs' package:

```
# Create a nodejs deb with only bin and lib directories:
# The 'VERSION' and 'ARCH' strings are automatically filled in for you
# based on the other arguments given.
% fpm -s dir -t deb -n nodejs -v 0.6.0 -C /tmp/installdir \
  -p nodejs_VERSION_ARCH.deb \
  -d "libssl0.9.8 > 0" \
  -d "libstdc++6 >= 4.4.3" \
  usr/bin usr/lib
```

Install the package, test it out:

```
# 'fpm' just produced us a nodejs deb:
% file nodejs_0.6.0-1_amd64.deb
nodejs_0.6.0-1_amd64.deb: Debian binary package (format 2.0)
% sudo dpkg -i nodejs_0.6.0-1_amd64.deb

% /usr/bin/node --version
v0.6.0
```

Package up the manpages (create nodejs-doc)

Now, create a package for the node manpage:

```
# Create a package of the node manpage
% fpm -s dir -t deb -p nodejs-doc_VERSION_ARCH.deb -n nodejs-doc -v 0.6.0 -C /tmp/
↪installdir usr/share/man
```

Look in the nodejs-doc package:

```
% dpkg -c nodejs-doc_0.6.0-1_amd64.deb| grep node.1
-rw-r--r-- root/root       945 2011-01-02 18:35 usr/share/man/man1/node.1
```

Package up the headers (create nodejs-dev)

Lastly, package the headers for development:

Package up the headers via:

```
% fpm -s dir -t deb -p nodejs-dev_VERSION_ARCH.deb -n nodejs-dev -v 0.6.0 -C /tmp/
↪installdir usr/include
% dpkg -c nodejs-dev_0.6.0-1_amd64.deb | grep -F .h
-rw-r--r-- root/root     14359 2011-01-02 18:33 usr/include/node/eio.h
-rw-r--r-- root/root      1118 2011-01-02 18:33 usr/include/node/node_version.h
-rw-r--r-- root/root     25318 2011-01-02 18:33 usr/include/node/ev.h
...
```

Yay!

Packages

Sources

Sources are the things you input to fpm.

dir - Directories

Synopsis:

```
fpm -s dir [other flags] path1 [path2 ...]
```

The 'dir' source will package up one or more directories for you.

Path mapping

Note: Path mapping was added in fpm version 0.4.40

Some times you want to take a path and copy it into a package but under a different location. fpm can use the = directive to mark that:

```
fpm [...] -s dir ./example/foo=/usr/bin
```

This will put the file *foo* in the /usr/bin directory inside the package.

A simple example of this can be shown with redis. Redis has a config file (redis.conf) and an executable (redis-server). Let's put the executable in /usr/bin and the config file in /etc/redis:

```
% ls src/redis-server redis.conf
src/redis-server
redis.conf

# install src/redis-server into /usr/bin/
# install redis.conf into /etc/redis/
% fpm -s dir -t deb -n redis --config-files /etc/redis/redis.conf -v 2.6.10 \
  src/redis-server=/usr/bin/ \
  redis.conf=/etc/redis/
Created deb package {:path=>"redis_2.6.10_amd64.deb"}

% dpkg -c redis_2.6.10_amd64.deb
drwx------ jls/jls           0 2013-07-11 23:49 ./
drwxrwxr-x jls/jls           0 2013-07-11 23:49 ./etc/
drwxrwxr-x jls/jls           0 2013-07-11 23:49 ./etc/redis/
-rw-rw-r-- jls/jls       24475 2013-02-11 04:24 ./etc/redis/redis.conf
drwxrwxr-x jls/jls           0 2013-07-11 23:49 ./usr/
drwxrwxr-x jls/jls           0 2013-07-11 23:49 ./usr/bin/
-rwxrwxr-x jls/jls     3566152 2013-02-14 11:19 ./usr/bin/redis-server

# Did the conffiles setting work? Yep!
% dpkg-deb -e redis_2.6.10_amd64.deb  .
% cat conffiles
/etc/redis/redis.conf
```

Voila!

gem - RubyGems

Simplest invocation

Here's a command that will fetch the latest *json* gem and convert it to a .deb package:

```
% cd /tmp
% fpm -s gem -t deb json
...
Created /tmp/rubygem-json-1.4.6-1.amd64.deb
```

This will download the latest 'json' rubygem from rubygems.org and convert it to a .deb. It will create a package named 'rubygem-json-VERSION_ARCH.deb' with appropriate version/arch in place.

Check the package:

```
% dpkg --info rubygem-json-1.4.6-1.amd64.deb
 new debian package, version 2.0.
 size 1004040 bytes: control archive= 335 bytes.
     275 bytes,    10 lines      control
       5 bytes,     1 lines      md5sums
 Package: rubygem-json
 Version: 1.4.6-1
 Architecture: amd64
 Maintainer: Florian Frank
 Standards-Version: 3.9.1
 Section: Languages/Development/Ruby
 Priority: extra
 Homepage: http://flori.github.com/json
 Description: JSON Implementation for Ruby
   JSON Implementation for Ruby
```

From the above, you can see that fpm automatically picked the package name, version, maintainer, section, homepage, and description all from the rubygem itself. Nothing for you to worry about :)

Specifying a version

You can ask for a specific version with '-v <VERSION>'. It will also handle dependencies. How about an older gem like rails 2.2.2:

```
% fpm -s gem -t deb -v 2.2.2 rails
Trying to download rails (version=2.2.2)
...
Created .../rubygem-rails-2.2.2-1.amd64.deb
```

Now observe the package created:

% dpkg –info ./rubygem-rails-2.2.2-1.amd64.deb new debian package, version 2.0. size 2452008 bytes: control archive= 445 bytes.

575 bytes, 11 lines control 6 bytes, 1 lines md5sums

Package: rubygem-rails Version: 2.2.2-1 Architecture: amd64 Maintainer: David Heinemeier Hansson Depends: rubygem-rake (>= 0.8.3), rubygem-activesupport (= 2.2.2),

rubygem-activerecord (= 2.2.2), rubygem-actionpack (= 2.2.2), rubygem-actionmailer (= 2.2.2), rubygem-activeresource (= 2.2.2)

Standards-Version: 3.9.1 Section: Languages/Development/Ruby Priority: extra Homepage: http://www.rubyonrails.org Description: Web-application framework with template engine, control-flow layer, and ORM.

Web-application framework with template engine, control-flow layer, and ORM.

Noticei how the *Depends* entry for this debian package lists all the dependencies that *rails* has?

Let's see what the package installs:

```
% dpkg -c ./rubygem-rails-2.2.2-1.amd64.deb
...
drwxr-xr-x root/root         0 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/
drwxr-xr-x root/root         0 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/
-rw-r--r-- root/root      3639 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/source_annotation_extractor.rb
-rw-r--r-- root/root       198 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/performance_test_help.rb
drwxr-xr-x root/root         0 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/tasks/
-rw-r--r-- root/root       204 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/tasks/log.rake
-rw-r--r-- root/root      2695 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/tasks/gems.rake
-rw-r--r-- root/root      4858 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/tasks/testing.rake
-rw-r--r-- root/root     17727 2011-01-20 17:00 ./usr/lib/ruby/gems/1.8/gems/rails-2.
↪2.2/lib/tasks/databases.rake
```

Packaging individual dependencies

A frequently-asked question is how to get a rubygem and all its dependencies converted. Let's take a look.

First we'll have to download the gem and its deps. The easiest way to do this is to stage the installation in a temporary directory, like this:

```
% mkdir /tmp/gems
% gem install --no-ri --no-rdoc --install-dir /tmp/gems cucumber
<output trimmed>

Successfully installed json-1.4.6
Successfully installed gherkin-2.3.3
Successfully installed term-ansicolor-1.0.5
Successfully installed builder-3.0.0
Successfully installed diff-lcs-1.1.2
Successfully installed cucumber-0.10.0
6 gems installed
```

Now you've got everything cucumber requires to run (just as a normal 'gem install' would.)

gem saves gems to the cache directory in the gem install dir, so check it out:

```
% ls /tmp/gems/cache
builder-3.0.0.gem    diff-lcs-1.1.2.gem  json-1.4.6.gem
cucumber-0.10.0.gem  gherkin-2.3.3.gem   term-ansicolor-1.0.5.gem
```

(by the way, under normal installation situations, gem would keep the cache in a location like /usr/lib/ruby/gems/1.8/cache, see 'gem env | grep INSTALL')

Let's convert all these gems to debs (output trimmed for sanity):

```
% find /tmp/gems/cache -name '*.gem' | xargs -rn1 fpm -d ruby -d rubygems --prefix
↪$(gem environment gemdir) -s gem -t deb
...
Created /tmp/gems/rubygem-json-1.4.6-1.amd64.deb
...
Created /tmp/gems/rubygem-builder-3.0.0-1.amd64.deb
...
Created /tmp/gems/rubygem-gherkin-2.3.3-1.amd64.deb
...
Created /tmp/gems/rubygem-diff-lcs-1.1.2-1.amd64.deb
...
Created /tmp/gems/rubygem-term-ansicolor-1.0.5-1.amd64.deb
...
Created /tmp/gems/rubygem-cucumber-0.10.0-1.amd64.deb

% ls *.deb
rubygem-builder-3.0.0-1.amd64.deb    rubygem-gherkin-2.3.3-1.amd64.deb
rubygem-cucumber-0.10.0-1.amd64.deb  rubygem-json-1.4.6-1.amd64.deb
rubygem-diff-lcs-1.1.2-1.amd64.deb   rubygem-term-ansicolor-1.0.5-1.amd64.deb
```

Nice, eh? Now, let's show what happens after these packages are installed:

```
# Show it's not install yet:
% gem list cucumber

*** LOCAL GEMS ***


# Now install the .deb packages:
% sudo dpkg -i rubygem-builder-3.0.0-1.amd64.deb \
  rubygem-cucumber-0.10.0-1.amd64.deb rubygem-diff-lcs-1.1.2-1.amd64.deb \
  rubygem-gherkin-2.3.3-1.amd64.deb rubygem-json-1.4.6-1.amd64.deb \
  rubygem-term-ansicolor-1.0.5-1.amd64.deb
...
Setting up rubygem-builder (3.0.0-1) ...
Setting up rubygem-diff-lcs (1.1.2-1) ...
Setting up rubygem-json (1.4.6-1) ...
Setting up rubygem-term-ansicolor (1.0.5-1) ...
Setting up rubygem-gherkin (2.3.3-1) ...
Setting up rubygem-cucumber (0.10.0-1) ...

# Is it installed?
% gem list cucumber

*** LOCAL GEMS ***

cucumber (0.10.0)

# Does it work?
```

```
% dpkg -L rubygem-cucumber | grep bin
/usr/lib/ruby/gems/1.8/gems/cucumber-0.10.0/bin
/usr/lib/ruby/gems/1.8/gems/cucumber-0.10.0/bin/cucumber
/usr/lib/ruby/gems/1.8/bin
/usr/lib/ruby/gems/1.8/bin/cucumber

% /usr/lib/ruby/gems/1.8/bin/cucumber --help
Usage: cucumber [options] [ [FILE|DIR|URL][:LINE[:LINE]*] ]+
...
```

You can put these .deb files in your apt repo (assuming you have a local apt repo, right?) and easily install them with 'apt-get' like: 'apt-get install rubygem-cucumber' and expect dependencies to work nicely.

Deterministic output

If convert a gem to a deb twice, you'll get different output even though the inputs didn't change:

> % fpm -s gem -t deb json % mkdir run1; mv *.deb run1 % sleep 1 % fpm -s gem -t deb json % mkdir run2; mv *.deb run2 % cmp run1/*.deb run2/*.deb run1/rubygem-json_2.1.0_amd64.deb run2/rubygem-json_2.1.0_amd64.deb differ: byte 124, line 4

This can be a pain if you're uploading packages to an apt repository which refuses reuploads that differ in content, or if you're trying to verify that packages have not been infected. There are several sources of nondeterminism; use 'diffoscope run1/*.deb run2/*.deb' if you want the gory details. See http://reproducible-builds.org for the whole story.

To remove nondeterminism due to differing timestamps, use the option –source-date-epoch-from-changelog; that will use the timestamp from the gem's changelog.

In case the gem doesn't have a standard changelog (and most don't, alas), use –source-date-epoch-default to set a default integer Unix timestamp. (This will also be read from the environment variable SOURCE_DATE_EPOCH if set.)

Gems that include native extensions may have nondeterministic output because of how the extensions get built (at least until fpm and compilers finish implementing the reproducible-builds.org recommendations). If this happens, use the option –gem-stagingdir=/tmp/foo.

For instance, picking the timestamp 1234 seconds after the Unix epoch:

> % fpm -s gem -t deb –source-date-epoch-default=1234 –gem-stagingdir=/tmp/foo json % mkdir run1; mv *.deb run1 % sleep 1 % fpm -s gem -t deb –source-date-epoch-default=1234 –gem-stagingdir=/tmp/foo json % mkdir run2; mv *.deb run2 % cmp run1/*.deb run2/*.deb % dpkg-deb -c run1/*.deb ... -rw-rw-r– 0/0 17572 1969-12-31 16:20 ./var/lib/gems/2.3.0/gems/json-2.1.0/CHANGES.md % date –date @1234 Wed Dec 31 16:20:34 PST 1969

If after using those three options, the files are still different, you may have found a bug; we might not have plugged all the sources of nondeterminism yet. As of this writing, these options are only implemented for reading gems and writing debs, and only verified to produce identical output when run twice on the same Linux system.

cpan - Perl packages

A basic example:

fpm -t deb -s cpan Fennec

The above will download Fennec from CPAN and build a Debian package of the Fennec Perl module locally.

By default, fpm believes the following to be true:

1. That your local Perl lib path will be the target Perl lib path
2. That you want the package name to be prefixed with the word perl
3. That the dependencies from CPAN are valid and that the naming scheme for those dependencies are prefixed with perl

If you wish to avoid any of those issues you can try:

fpm -t deb -s cpan –cpan-perl-lib-path /usr/share/perl5 Fennec

This will change the target path to where perl will be. Your local perl install may be /opt/usr/share/perl5.10 but the package will be constructed so that the module will be installed to /usr/share/perl5

fpm -t deb -s cpan –cpan-package-name-prefix fubar Fennec This will replace the perl default prefix with fubar. The resulting package will be named in the scheme of fubar-fennec-2.10.deb

fpm -t -deb -s cpan –no-depends Fennec This will remove omit dependencies from being added to the package metadata.

A full list of available options for CPAN are listed here:

```
--cpan-perl-bin PERL_EXECUTABLE (cpan only) The path to the perl executable you wish␣
↪to run. (default: "perl")
--cpan-cpanm-bin CPANM_EXECUTABLE (cpan only) The path to the cpanm executable you␣
↪wish to run. (default: "cpanm")
--cpan-mirror CPAN_MIRROR     (cpan only) The CPAN mirror to use instead of the␣
↪default.
--[no-]cpan-mirror-only       (cpan only) Only use the specified mirror for metadata.␣
↪(default: false)
--cpan-package-name-prefix NAME_PREFIX (cpan only) Name to prefix the package name␣
↪with. (default: "perl")
--[no-]cpan-test              (cpan only) Run the tests before packaging? (default:␣
↪true)
--cpan-perl-lib-path PERL_LIB_PATH (cpan only) Path of target Perl Libraries
--[no-]cpan-sandbox-non-core  (cpan only) Sandbox all non-core modules, even if they
↪'re already installed (default: true)
--[no-]cpan-cpanm-force       (cpan only) Pass the --force parameter to cpanm␣
↪(default: false)
```

pleaserun - Please, run!

Synopsis:

```
fpm -s pleaserun [other flags] program [args...]
```

The *pleaserun* source uses the pleaserun project to help you build a package that installs a service.

pleaserun supports the following service managers:

- sysv, /etc/init.d/whatever
- upstart
- systemd
- runit
- launchd (OS X)

Automatic Platform Detection

Targeting multiple platforms with a single package is hard. What init system is used? Can you predict?

fpm+pleaserun can detect this at installation-time!

One Package for All Platforms

The following is an example which creates an rpm that makes *redis* service available:

```
fpm -s pleaserun -t rpm -n redis-service /usr/bin/redis-server
```

The output looks like this:

```
Created package {:path=>"redis-service-1.0-1.x86_64.rpm"}
```

Note: Your package will detect the service platform (systemd, upstart, etc) automatically upon installation :)

Let's see what happens when I install this on Fedora 25 (which uses systemd):

```
% sudo rpm -i redis-service-1.0-1.x86_64.rpm
Platform systemd (default) detected. Installing service.
To start this service, use: systemctl start redis-server
```

And checking on our service:

```
% systemctl status redis-server
 redis-server.service - redis-server
   Loaded: loaded (/etc/systemd/system/redis-server.service; disabled; vendor pr
   Active: inactive (dead)
```

(It is inactive and disabled because fpm does not start it by default)

As you can see in the above example, *fpm* added an after-install script which detects the service manager during installation. In this case, *systemd* was detected.

The above example shows installing on Fedora 25, which uses systemd. You can use this same rpm package on CentOS 6, which uses upstart, and it will still work:

```
% sudo rpm -i redis-service-1.0-1.x86_64.rpm
Platform upstart (0.6.5) detected. Installing service.
To start this service, use: initctl start redis-server
```

And checking on our service:

```
% initctl status redis-server
redis-server stop/waiting
```

Hurray! We now have a single rpm that installs this *redis-service* service on most systems.

Questions You May Have

How does the package know whether to use systemd, upstart, sysv, or something else?

fpm creates a package that does a platform check when the package is installed

Does this mean I need ruby and pleaserun installed on the target system?

> Fortunately, no! fpm creates a package that consists only of the install scripts and the service files. The install scripts are written in bourne shell */bin/sh*.
>
> Here's an example:

```
% fpm -s pleaserun -t rpm -n example /usr/bin/example
% rpm -qlp example-1.0-1.x86_64.rpm
/usr/share/pleaserun/example/generate-cleanup.sh
/usr/share/pleaserun/example/install-path.sh
/usr/share/pleaserun/example/install.sh
/usr/share/pleaserun/example/launchd/10.9/files/Library/LaunchDaemons/
↪example.plist
/usr/share/pleaserun/example/launchd/10.9/install_actions.sh
/usr/share/pleaserun/example/systemd/default/files/etc/default/example
/usr/share/pleaserun/example/systemd/default/files/etc/systemd/system/
↪example.service
/usr/share/pleaserun/example/systemd/default/install_actions.sh
/usr/share/pleaserun/example/sysv/lsb-3.1/files/etc/default/example
/usr/share/pleaserun/example/sysv/lsb-3.1/files/etc/init.d/example
/usr/share/pleaserun/example/upstart/0.6.5/files/etc/default/example
/usr/share/pleaserun/example/upstart/0.6.5/files/etc/init/example.conf
/usr/share/pleaserun/example/upstart/1.5/files/etc/default/example
/usr/share/pleaserun/example/upstart/1.5/files/etc/init/example.conf
```

> The package includes service definitions for your specific service that can target systemd, a few versions of upstart, launchd, and sysv.
>
> Upon install, the *install.sh* script is run which detects the correct service definition to install.

Does the package clean up after itself when I remove it?

> It should. When installing, the package generates a manifest of what service files were installed, and it uses that manifest to clean up when the package is uninstalled or removed.

python - Python packages

Minimal example

Here's a simple example to download the *pyramid* python package and convert it to an rpm:

```
% fpm -s python -t rpm pyramid
Trying to download pyramid (using easy_install)
Searching for pyramid
Reading http://pypi.python.org/simple/pyramid/
Reading http://docs.pylonshq.com
Reading http://docs.pylonsproject.org
Best match: pyramid 1.0
...
Created /home/jls/python-pyramid-1.0.noarch.rpm
```

This will download the latest 'pyramid' python module using easy_install and convert it to an rpm. It will create a package named 'python-pyramid-VERSION_ARCH.rpm' with appropriate version/arch in place.

Check the package:

```
% rpm -qip python-pyramid-1.0.noarch.rpm
Name        : python-pyramid               Relocations: (not relocatable)
Version     : 1.0                               Vendor: (none)
Release     : 1                            Build Date: Mon 16 May 2011 06:41:16 PM␣
↪PDT
Install Date: (not installed)              Build Host: snack.home
Group       : default                      Source RPM: python-pyramid-1.0-1.src.rpm
Size        : 2766900                         License: BSD-derived (http://www.
↪repoze.org/LICENSE.txt)
Signature   : (none)
URL         : http://docs.pylonsproject.org
Summary     : The Pyramid web application framework, a Pylons project
Description :
The Pyramid web application framework, a Pylons project
```

From the above, you can see that fpm automatically picked the package name, version, maintainer, homepage, and description all from the python package itself. Nothing for you to worry about :)

Looking at the dependencies:

```
% rpm -qRp python-pyramid-1.0.noarch.rpm
python-Chameleon >= 1.2.3
python-Mako >= 0.3.6
python-Paste > 1.7
python-PasteDeploy >= 0
python-PasteScript >= 0
python-WebOb >= 1.0
python-repoze.lru >= 0
python-setuptools >= 0
python-zope.component >= 3.6.0
python-zope.configuration >= 0
python-zope.deprecation >= 0
python-zope.interface >= 3.5.1
python-venusian >= 0.5
python-translationstring >= 0
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
```

Packaging for multiple pythons

Some systems package python with packages named 'python24' and 'python26' etc.

You can build packages like this with fpm using the *–python-package-name-prefix* flag:

```
% ruby bin/fpm -s python -t rpm --python-package-name-prefix python26 pyramid
...
Created /home/jls/projects/fpm/python26-pyramid-1.0.noarch.rpm

% rpm -qRp python26-pyramid-1.0.noarch.rpm
python26-Chameleon >= 1.2.3
python26-Mako >= 0.3.6
python26-Paste > 1.7
python26-PasteDeploy >= 0
<remainder of output trimmed... you get the idea>
```

You can ask for a specific version with '-v <VERSION>'. It will also handle dependencies. Here's an example converting an older package like pysqlite version 2.5.6:

```
% fpm -s python -t rpm --python-package-name-prefix python26 -v 2.5.6 'pysqlite'
Trying to download pysqlite (using easy_install)
Searching for pysqlite==2.5.6
Reading http://pypi.python.org/simple/pysqlite/
Reading http://pysqlite.googlecode.com/
< ... output cut ... >
Created /home/jls/projects/fpm/python26-pysqlite-2.5.6.x86_64.rpm
```

Local python sources

If you are the developer of a python package, or you already have the local package downloaded and unpacked.

In this scenario, you can tell fpm to use the *setup.py*:

```
% ls pyramid/setup.py
pyramid/setup.py

% fpm -s python -t rpm pyramid/setup.py
...
Created /tmp/python-pyramid-1.0.noarch.rpm
```

rpm - RPM Packages

Synopsis:

```
fpm -s rpm [other options] path-to-rpm
```

Using 'rpm' as a source lets you treat an existing package as a source for building a new one. This can be useful for converting packages between formats or for "editing" upstream packages.

Strip out docs under */usr/share/doc*:

```
fpm -t rpm -s rpm --exclude /usr/share/doc ruby-2.0.0.x86_64.rpm`
```

Rename a package and assign different version:

```
fpm -t rpm -s rpm --name myruby --version $(date +%S) ruby-2.0.0.x86_64.rpm
```

Convert an rpm in to a deb:

```
fpm -t deb -s rpm fpm-1.63.x86_64.rpm
```

virtualenv - Python virtual environments

Synopsis:

```
fpm -s virtualenv [other options] EGG_SPEC|requirements.txt
```

FPM has support for building packages that provide a python virtualenv from a single egg or from a *requirements.txt* file. This lets you bundle up a set of python dependencies separate from system python that you can then distribute.

Note: *virtualenv* support requires that you have *virtualenv* and the *virtualenv-tools* binary on your path. This can usually be achieved with *pip install virtualenv virtualenv-tools*.

Example uses:

Build an rpm package for ansible:

```
fpm -s virtualenv -t deb ansible
yum install virtualenv-ansible*.rpm
which ansible # /usr/share/python/ansible/bin/ansible
```

Create a debian package for your project's python dependencies under */opt*:

```
echo 'glade' >> requirements.txt
echo 'paramiko' >> requirements.txt
echo 'SQLAlchemy' >> requirements.txt
fpm -s virtualenv -t deb --name myapp-python-libs \
  --prefix /opt/myapp/virtualenv requirements.txt
```

Create a debian package from a version 0.9 of an egg kept in your internal pypi repository, along with it's external dependencies:

```
fpm -s virtualenv -t deb \
  --virtualenv-pypi-extra-url=https://office-pypi.lan/ \
  proprietary-magic=0.9
```

Targets

Targets are the output of fpm.

Want to contribute? Or need help?

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms. See the Code of Conduct for details.

All contributions are welcome: ideas, patches, documentation, bug reports, complaints, and even something you drew up on a napkin :)

It is more important that you are able to contribute and get help if you need it than it is how you contribute or get help.

That said, some points to get started:

- Have a problem you want fpm to solve for you? You can email the mailing list, or join the IRC channel #fpm on irc.freenode.org, or email me personally (jls@semicomplete.com)
- Have an idea or a feature request? File a ticket on github, or email the mailing list, or email me personally (jls@semicomplete.com) if that is more comfortable.
- If you think you found a bug, it probably is a bug. File it on github or send details to the mailing list.
- If you want to send patches, best way is to fork this repo and send me a pull request. If you don't know git, I also accept diff(1) formatted patches - whatever is most comfortable for you.

- Want to lurk about and see what others are doing? IRC (#fpm on irc.freenode.org) is a good place for this as is the mailing list.

Contributing changes by forking from GitHub

First, create a GitHub account if you do not already have one. Log in to GitHub and go to [the main fpm GitHub page](https://github.com/jordansissel/fpm).

At the top right, click on the button labeled "Fork". This will put a forked copy of the main fpm repo into your account. Next, clone your account's GitHub repo of fpm. For example:

$ git clone git@github.com:yourusername/fpm.git

Development Environment

If you don't already have the bundler gem installed, install it now:

$ gem install bundler

Now change to the root of the fpm repo and run:

$ bundle install

This will install all of the dependencies required for running fpm from source. Most importantly, you should see the following output from the bundle command when it lists the fpm gem:

... Using json (1.8.1) Using fpm (0.4.42) from source at . Using hitimes (1.2.1) ...

If your system doesn't have *bsdtar* by default, make sure to install it or some tests will fail:

apt-get install bsdtar

yum install bsdtar

Next, run make in root of the fpm repo. If there are any problems (such as missing dependencies) you should receive an error

At this point, the fpm command should run directly from the code in your cloned repo. Now simply make whatever changes you want, commit the code, and push your commit back to master.

If you think your changes are ready to be merged back to the main fpm repo, you can generate a pull request on the GitHub website for your repo and send it in for review.

Problems running bundle install?

If you are installing on Mac OS 10.9 (Mavericks) you will need to make sure that you have the standalone command line tools separate from Xcode:

$ xcode-select –install

Finally, click the install button on the prompt that appears.

Release Notes and Change Log

1.9.2 (July 29, 2017)

- rpm: Fix *–config-files* handling (#1390, #1391; Jordan Sissel)

1.9.1 (July 28, 2017) happy sysadmin day!

- Documentation improvements: #1291; Pablo Castellano. #1321; ge-fa. #1309; jesusbagpuss. #1349; Perry Stole. #1352, Jordan Sissel. #1384; Justin Kolberg.
- Testing improvements: #1320; Rob Young. #1266; Ryan Parman. #1374; Thiago Figueiró.
- Fix bug so fpm can now copy symlinks correctly (#1348; ServiusHack)
- apk: Improve performance (#1358; Jan Delgado)
- cpan: Fix crash when CPAN query returns a version value that was a number and fpm was expecting a string. (#1344, #1343; liger1978)
- cpan: Fix MetaCPAN searches to use v1 of MetaCPAN's API. The v0 API is no longer provided by MetaCPAN. (#1341, *#1339*; Bob Bell)
- cpan: Have perl modules implicitly "provide" (*–provides*) capabilities. (#1340; Bob Bell. #1345; liger1978)
- cpan: Now transforms perl version values like "5.008001" to "5.8.1" (#1342; Bob Bell)
- cpan: Use >= ("this version or newer") for package dependencies instead of = ("exactly this version"). (#1338; Bob Bell)
- deb: Add *–deb-after-purge* flag for running a script after *apt-get purge* is run. (Alexander Weidinger)
- deb: fix bug when using *–deb-upstart* would use the wrong file name (#1325, #1287; vbakayev)
- deb: New flags *–deb-interest-noawait* and *–deb-activate-nowait*. (#1225, #1359; Philippe Poilbarbe)
- dir: Remove a debug statement that would put fpm into a debug prompt (#1293, #1259; Joseph Anthony Pasquale Holsten)
- dir: When using **'path mapping'_** (*a=b* syntax), and *a* is a symlink, use the path *b* as the symlink, not *b/a* (#1253, Nemanja Boric)
- gem: Can now make **reproducible_builds_** when building a deb (*-s gem -t deb*). See the **'Deterministic output'_** docs.
- gem: Add *–gem-embed-dependencies* flag to include in the output package all dependent gems of the target. For example, *fpm -s gem -t rpm –gem-embed-dependencies rails* will create a single *rails* rpm that includes active_support, active_record, etc.
- pleaserun: Add more flags (*–pleaserun-chdir*, *–pleaserun-user*, etc) to allow more customization of pleaserun services. (#1311; Paulo Sousa)
- python: Add *–python-setup-py-arguments* flag for passing arbitrary flags to *python setup.py install* (#1120, #1376; Ward Vandewege, Joseph Anthony Pasquale Holsten)
- rpm: –config-files can now copy files from outside of the package source. This means you can do things like *fpm -s gem -t rpm –config-files etc/my/config* and have *etc/my/config* come from the local filesystem. (#860, #1379; jakerobinson, Joseph Anthony Pasquale Holsten)
- tar: Only create *.scripts* directory if there are scripts to include (#1123, #1374; Thiago Figueiró)
- virtualenv: Add *–virtualenv-find-links* flag which appends *–find-links* to the *pip install* command.
- virtualenv: documentation improvements (Nick Griffiths)
- virtualenv: Make *–prefix* useful and deprecate *–virtualenv-install-location* (#1262; Nick Griffiths)
- zip: fix bug in output where the temporary directory would be included in the file listing (#1313, #1314; Bob Vincent)
- Other: Remove unused archive-tar-minitar as a dependency of fpm (#1355; Diego Martins)

- Other: Add stud as a runtime dependency (#1354; Elan Ruusamäe)

1.9.0 (July 28, 2017)

Yanked offline. I forgot some dependency changes. Hi.

1.8.1 (February 7, 2017)

- Pin archive-tar-minitar library to version 0.5.2 to work around a problem breaking *gem install fpm*

1.8.0 (December 28, 2016)

- virtualenv: Add *–virtualenv-setup-install* flag to run *setup.py install* after pip finishes installing things. (#1218; John Stowers)
- virtualenv: Add *–virtualenv-system-site-package* flag which creates the virtualenv in a way that allows it to use the system python packages. (#1218; John Stowers)
- cpan: Fix bug preventing some perl modules from being installed (#1236, #1241; Richard Grainger)
- rpm: Documentation improvements (#1242; Nick Griffiths)

1.7.0 (November 28, 2016)

- virtualenv: Fix a bug where *pip* might be run incorrectly (#1210; Nico Griffiths)
- FreeBSD: –architecture (-a) flag now sets FreeBSD package ABI (#1196; Matt Sharpe)
- perl/cpan: Fix bug and now local modules can be packaged (#1202, #1203; liger1978)
- perl/cpan: Add support for *http_proxy* environment variable and improve how fpm queries CPAN for package information. (#1206, #1208; liger1978)
- Fix crash for some users (#1231, #1148; Jose Diaz-Gonzalez)
- Documentation now published on fpm.readthedocs.io. This is a work-in progress. Contributions welcome! <3 (#1237, Jordan Sissel)
- deb: Can now read bz2-compressed debian packages. (#1213; shalq)
- pleaserun: New flag –pleaserun-chdir for setting the working directory of a service. (#1235; Claus F. Strasburger)

1.6.3 (September 15, 2016)

- Fix bug in fpm's release that accidentally included a few *.pyc* files (#1191)

1.6.2 (July 1, 2016)

- Reduce *json* dependency version to avoid requiring Ruby 2.0 (#1146, #1147; patch by Matt Hoffman)
- pacman: skip automatic dependencies if –no-auto-depends is given (Leo P)
- rpm: Fix bug where –rpm-tag was accidentally ignored (#1134, Michal Mach)

- deb: Omit certain fields from control file if (Breaks, Depends, Recommends, etc) if there are no values to put in that field. (#1113, TomyLobo)
- rpm: remove trailing slash from Prefix for rpm packages (#819, luto)
- virtualenv: Now supports being given a requirements.txt as the input. (Nick Griffiths)

1.6.1 (June 10, 2016)

- freebsd: Only load xz support if we are doing a freebsd output. (#1132, #1090, Ketan Padegaonkar)

1.6.0 (May 25, 2016)

- New source: pleaserun. This lets you create packages that will install a system service. An after-install script is used in the package to determine which service platform to target (systemd, upstart, etc). Originated from Aaron Mildenstein's work on solving this problem for Logstash. (#1119, #1112)
- New target: Alpine Linux "apk" packages. (#1054, George Lester)
- deb: don't append *.conf* to an upstart file if the file name already ends in *.conf*. (#1115, josegonzalez)
- freebsd: fix bug where –package flag was ignored. (#1093, Paweł Tomulik)
- Improvements to the fpm rake tasks (#1101, Evan Gilman)

1.5.0 (April 12, 2016)

- Arch package support is now available via -s pacman and -t pacman. (#916; wonderful community effort making this happen!)
- FreeBSD packages can now be built *-t freebsd* (#1073; huge community effort making this happen!)
- You can now set fpm flags and arguments with the FPMOPTS environment variable (#977, mildred)
- Using –exclude-file no longer causes a crash. Yay! (#982, wyaeld)
- A new rake task is available for folks who want to invoke fpm from rake (#756, pstengel)
- On FreeBSD, when tarring, gtar is now used. (#1008, liv3d)
- virtualenv: Add –virtualenv-pypi-extra-url flag to specify additional PyPI locations to use when searching for packages (#1012, Paul Krohn)
- deb: Init scripts, etc/default, and upstart files are automatically added as config files in a debian package. Disable this behavior with —deb-auto-config-files
- deb: Small changes to make lintian complain less about our resulting debs.
- deb: New flag –deb-systemd lets you specify a systemd service file to include in your package. (#952, Jens Peter Schroer)
- cpan: Add –[no-]cpan-cpanm-force flag to pass –force to cpanm.
- rpm: File names with both spaces and symbols should now be packageable. (#946, iwonbigbro)
- cpan: Now queries MetaCPAN for package info if we can't find any in the cpan archive we just downloaded. (#849, BaxterStockman)
- rpm: You can now specify custom rpm tags at the command line. Be careful, as no validation is done on this before sending to rpmbuild. (#687, vStone)

- cpan: Install if the package name given is a local file (#986, mdom)
- sh: Metadata now available as env vars for post-install scripts (#1006, Ed Healy)
- rpm: No more warning if you don't set an epoch. (#1053, Joseph Frazier)

1.4.0 (July 26, 2015)

- Solaris 11 IPS packages 'p5p' now supported *-t p5p*. (Jonathan Craig)
- Python Virtualenv is now supported *-t virtualenv* (#930, Simone Margaritelli and Daniel Haskin)
- deb: Files in /etc are now by default marked as config files. (#877, Vincent Bernat)
- *fpm –help* output now includes a list of supported package types (#896, Daniel Haskin)
- cpan: –[no-]cpan-sandbox-non-core flag to make non-core module sandboxing optional during packaging (#752, Matt Sharpe)
- rpm: Add –rpm-dist flag for specifically setting the target distribution of an rpm. (Adam Lamar)
- rpm: Fix a crash if –before-upgrade or –after-upgrade were used. (#822, Dave Anderson)
- deb: Ensure maintainer scripts have shebang lines (#836, Wesley Spikes)
- deb: Fix bug in maintainer scripts where sometimes we would write an empty shell function. Empty functions aren't valid in shell. (Wesley Spikes)
- Fix symlink copying bug (#863, Pete Fritchman)
- python: Default to https for pypi queries (Timothy Sutton)
- New flag –exclude-file for providing a file containing line-delimited exclusions (Jamie Lawrence)
- python: new flag –python-disable-dependency to disable specific python dependencies (Ward Vandewege)
- python: ensure we avoid wheel packages for now until fpm better supports them. (#885, Matt Callaway)
- deb: Add support for installation states "abort-remove" and "abort-install" (#887, Daniel Haskin)
- If PATH isn't set, and we need it, tell the user (#886, Ranjib Dey)
- cpan: –[no-]cpan-test now works correctly (#853, Matt Schreiber)
- deb-to-rpm: some improved support for config file knowledge passing from deb to rpm packages (Daniel Haskin)

1.3.3 (December 11, 2014)

- The fpm project now uses Contributor Covenant. You can read more about this on the website: http://contributor-covenant.org/
- npm: Fix bug causing all *-s npm* attempts to fail due to a missing method. This bug was introduced in 1.3.0. (#800, #806; Jordan Sissel)
- rpm: fix bug in rpm input causing a crash if the input rpm did not have any triggers (#801, #802; Ted Elwartowski)

1.3.2 (November 4, 2014)

- deb: conversion from another deb will automatically use any changelog found in the source deb (Jordan Sissel)

1.3.1 (November 4, 2014)

- deb: fix md5sums generation such that *dpkg -V* now works (#799, Matteo Panella)
- rpm: Use maximum compression when choosing xz (#797, Ashish Kulkarni)

1.3.0 (October 25, 2014)

- Fixed a bunch of Ruby 1.8.7-related bugs. (Jordan Sissel)
- cpan: Fix bug in author handling (#744, Leon Weidauer)
- cpan: Better removal of perllocal.pod (#763, #443, #510, Mathias Lafeldt)
- rpm: Use lstat calls instead of stat, so we don't follow symlinks (#765, Shrijeet Paliwal)
- rpm and deb: Now supports script actions on upgrades. This adds two new flags: –before-upgrade and –after-upgrade. (#772, #661; Daniel Haskin)
- rpm: Package triggers are now supported. New flags: –rpm-trigger-before-install, –rpm-trigger-after-install, –rpm-trigger-before-uninstall, –rpm-trigger-after-target-uninstall. (#626, Maxime Caumartin)
- rpm: Add –rpm-init flag; similar to –deb-init. (Josh Dolitsky)
- sh: Skip installation if already installed for the given version. If forced, the old installation is renamed. (#776, Chris Gerber)
- deb: Allow Vendor field to be omitted now by specifying *–vendor ""* (#778, Nate Brown)
- general: Add –log=level flag for setting log level. Levels are error, warn, info, debug. (Jordan SIssel)
- cpan: Check for Build.PL first before Makefile.PL (#787, Daniel Jay Haskin)
- dir: Don't follow symlinks when copying files (#658, Jordan Sissel)
- deb: Automatically provide a 'changes' file in debs because lintian complains if they are missing. (#784, Jordan Sissel)
- deb: Fix and warn for package names that have spaces (#779, Grantlyk)
- npm: Automatically set the prefix to *npm prefix -g* (#758, Brady Wetherington and Jordan Sissel)

1.2.0 (July 25, 2014)

- rpm: Add –rpm-verifyscript for adding a custom rpm verify script to your package. (Remi Hakim)
- Allow the -p flag to target a directory for writing the output package (#656, Jordan Sissel)
- Add –debug-workspace which skips any workspace cleanup to let users debug things if they break. (#720, #734; Jordan Sissel)
- rpm: Add –rpm-attr for controlling attribute settings per file. This setting will likely be removed in the future once rpmbuild is no longer needed. (#719)
- deb: Add –deb-meta-file to add arbitrary files to the control dir (#599, Dan Brown)
- deb: Add –deb-interest and –deb-activate for adding package triggers (#595, Dan Brown)
- cpan: Fix small bug in handling empty metadata fields (#712, Mathias Lafeldt)
- rpm: Fix bug when specifying both –architecture and –rpm-os (#707, #716; Alan Ivey)
- gem: Fix bug where –gem-version-bins is given but package has no bins (#688, Jan Vansteenkiste)

- deb: Set permissions correct on the package's internals. Makes lintian happier. (Jan Vansteenkiste)
- rpm: rpmbuild's _tmppath now respects –workdir (#714, Jordan Sissel)
- gem/rpm: Add –rpm-verbatim-gem-dependencies to use old-style (fpm 0.4.x) rpm gem dependencies (#724, Jordan Sissel)
- gem/rpm: Fix bug for gem pessimistic constraints when converting to rpm (Tom Duckering)
- python: Fix small bug with pip invocations (#727, Dane Knecht)

1.1.0 (April 23, 2014)

- New package type: zip, for converting to and from zip files (Jordan Sissel)
- New package type: sh, a self-extracting package installation shell archive. (#651, Chris Gerber)
- 'fpm –version' will now emit the version of fpm.
- rpm: supports packaging fifo files (Adam Stephens)
- deb: Add –deb-use-file-permissions (Adam Stephens)
- cpan: Improve how fpm tries to find cpan artifacts for download (#614, Tim Nicholas)
- gem: Add –gem-disable-dependency for removing one or more specific rubygem dependencies from the automatically-generated list (#598, Derek Olsen)
- python: Add –python-scripts-executable for setting a custom interpreter to use for the hashbang line at the top of may python package scripts. (#628, Vladimir Rutsky)
- Allow absolute paths with –directories even when –prefix is used (Vladimir Rutsky)
- dir: Now correctly identifies hardlinked files and creates a package correctly with that knowledge (#365, #623, #659; Vladimir Rutsky)
- rpm: Add –rpm-auto-add-exclude-directories for excluding directories from the –rpm-auto-add-directories behavior (#640, Vladimir Rutsky)
- general: –config-files now accepts directories and will recursively mark any files within as config files inside the package (#642, Vladimir Rutsky)
- general: If you specify a –config-files path that doesn't exist, you will now get an error. (#654, Alan Franzoni)
- python: Support –python-pypi when using –python-pip (#652, David Lindquist)
- deb: Tests now try to make packages ensure we don't upset lintian (#648, Sam Crang)
- rpm: Fix architecture targeting (#676, Rob Kinyon)
- rpm: Allow –rpm-user and –rpm-group to override the user/group even if –rpm-use-file-permissions is enabled. (#679, Jordan Sissel)
- gem: Add –gem-version-bins for appending the gem version to the file name of executable scripts a rubygem may install. (Jan Vansteenkiste)
- python: Attempt to provide better error messages for known issues in python environments (#664, Jordan Sissel)

1.0.2 (January 10, 2013)

- rpm: No longer converts - to _ in dependency strings (#603, Bulat Shakirzyanov)
- Handle Darwin/OSX tar invocations (now tries 'gnutar' and 'gtar'). (Jordan Sissel)

- Process $HOME/.fpm, and $PWD/.fpm in the correct order and allow CLI flags to override fpm config file settings. (#615, Jordan Sissel)
- Don't leave empty gem bin paths in packages that don't need them (#612, Jordan Sissel)
- deb: Make –deb-compression=gz work correctly (#616, #617; Evan Krall, Jason Yan)

1.0.1 (December 7, 2013)

- deb: Correctly handle –config-files given with a leading / (Jordan Sissel)

1.0.0 (December 5, 2013)

- Config file of flags is now supported. Searches for $HOME/.fpm and $PWD/.fpm. If both exist, $HOME is loaded first so $PWD can override. (Pranay Kanwar)
- pkgin: Basic support for SmartOS/pkgsrc's pkgin format. (#567, Brian Akins)
- cpan: catch more cases of perllocal.pod and delete them (#510, Jordan Sissel)
- cpan: Correctly support module version selection (#518, Matt Sharpe)
- cpan: include builddeps in PERL5LIB when running cpan tests (#500, Matt Sharpe)
- cpan: Avoid old system perl modules when doing module builds (#442, #513; Matt Sharpe)
- python: safer gathering of python module dependencies.
- python: better handling of unicode strings in python package metadata (#575, Bruno Renié)
- cpan: Support 'http_proxy' env var. (#491, Patrick Cable)
- deb: –deb-user and –deb-group both default to 'root' now (#504, Pranay Kanwar)
- deb: convert '>' to '>>' in deb version constraints (#503, #439, Pranay Kanwar)
- deb: Warn if epoch is set. Just so you know what's going on, since the default filename doesn't include the epoch. (#502, Pranay Kanwar)
- deb,rpm: –config-files is now recursive if you give it a directory. This seems to be the most expected behavior by users. (#171, #506; Pranay Kanwar)
- dir: Respect -C when using path mapping (#498, #507; Pranay Kanwar)
- rpm: Add –rpm-ignore-iteration-in-dependencies to let you to depend on any release (aka iteration) of the same version of a package. (#364, #508; Pranay Kanwar)
- dir: Handle copying of special files when possible (#347, #511, #539, #561; Pranay Kanwar)
- rpm: Don't mistake symlinks as actual directories (#521, Nathan Huff)
- npm: Choose an alternate npm registry with –npm-registry (#445, #524; Matt Sharpe)
- cpan: Choose an alternate cpan server with –cpan-mirror. Additionally, you can use –cpan-mirror-only to only use this mirror for metadata queries. (#524, Matt Sharpe)
- deb: Fix broken –deb-changelog flag (#543, #544; Tray Torrance)
- deb: When –deb-upstart is given, automatically create an upstart-sysv symlink /etc/init.d/<name> to /lib/init/upstart-job (#545, Igor Galić)
- rpm: Fix bug when generating spec file listings on files with strange characters in the names. (#547, Chris Chandler)

- dir: Fix bug where the new directory mapping feature would cause you not to be able to select files with '=' in the name for packaging. (#556, #554; Pranay Kanwar)
- python: Fix some unicode string issues in package metadata (#575, Bruno Renié)
- gem-rpm: Now respects the –gem-package-name-prefix when generating the 'rubygem(name)' provides statement (#585, Stepan Stipl)
- deb: Downcase and replace underscores with dashes in 'provides' list. (#591, Eric Connell)
- deb: Fix a lintian complaint about md5sums permissions (#593, Sam Crang)
- cpan: Modules with 'MYMETA' files are now supported (#573, Michael Donlon)

0.4.42 (July 23, 2013)

- dir: make source=destination mappings behave the same way 'rsync -a' does with respect to source and destination paths.

0.4.41 (July 17, 2013)

- cpan: handle cases where modules don't specify a license
- deb: support multiple init scripts (#487, patch by Kristian Glass)

0.4.40 (July 12, 2013)

- dir: supports mapping one path to another. You set mappings by using 'source=destination' syntax. For example: % fpm -s dir -t deb -n example /home/jls/.zshrc=/etc/skel/ The key above is the '=' symbol. The result of the above will be a package containing only /etc/skel/.zshrc For more, see https://github.com/jordansissel/fpm/wiki/Source:-dir#mapping
- python: the default scripts location is now chosen by python itself. The previous default was "/usr/bin" and was not a good default. (#480)
- rpm: config files should have attributes (#484, patch by adamcstephens)
- python: correctly log the python setup.py exit code (#481, patch by Derek Ludwig)

0.4.39 (June 27, 2013)

- cpan: support more complex dependency specifications (reported by Mabi Knittel)

0.4.38 (June 24, 2013)

- cpan: fpm's cpan code now works under ruby 1.8.7
- python: fix a bug in dependency handling (#461, Pranay Kanwar)
- pear: Added –pear-data-dir flag (#465, Zsolt Takacs)
- cpan: fix a bug with some clean up on certain 64bit systems
- gem: improve detection of the gem bin install path (#476, Tray Torrance)
- rpm: fix bug when calling using –rpm-use-file-permissions (#464, Rich Horwood)

0.4.37 (May 30, 2013)

- deb: fix creation failures on OS X (#450, patch by Anthony Scalisi and Matthew M. Boedicker)
- deb: you can now set –deb-build-depends. This is generally for extremely rare use cases. (#451, patch by torrancew)
- perl: add –cpan-perl-lib-path for a custom perl library installation path (#447, patch by Brett Gailey)

0.4.36 (May 15, 2013)

- pear: only do channel-discover if necessary (#438, patch by Hatt)
- cpan: now supports cpan modules that use Module::Build
- cpan: –no-cpan-test now skips tests for build/configure dependencies
- rpm: Add –rpm-defattrfile and –rpm-defattrdir flags (#428, patch by phrawzty)

0.4.35 – was not announced

0.4.34 (May 7, 2013)

- Now supports CPAN - Perl mongers rejoice! For example: 'fpm -s cpan -t deb DBI'
- deb: fixed some additional complaints by lintian (#420, patch by Pranay Kanwar)
- rpm: add flags –rpm-autoreqprov, –rpm-autoreq, and –rpm-autoprov to tell rpm to enable that feature in the rpm spec. (#416, patch by Adam Stephens)

0.4.33 (April 9, 2013)

- Now supports npm, the node package manager. For example: 'fpm -s npm -t deb express'

0.4.32 (April 9, 2013)

- COMPATIBILITY WARNING: rpm: The default epoch is now nothing because this aligns more closely with typical rpm packages in the real world. This decision was reached in #381. If you need the previous behavior, you must now specify '–epoch 1' (#388, patch by Pranay Kanwar)
- python: new flag –python-obey-requirements-txt which makes a requirements.txt file from the python package used for the package dependencies instead of the usual setup.py dependencies. The default behavior without this flag is to respect setup.py. (#384)
- deb: new flag –deb-shlibs to specify the content of the 'shlibs' file in the debian package (#405, patch by Aman Gupta)
- deb: fixed a few lintian errors (empty conffiles, md5sums on symlinks, etc)
- Add '-f' / '–force' flag to force overwriting an existing package output path (#385, Timothy Sutton)
- New flag: –no-auto-depends flag to skip any automatic dependencies that would normally be added by gem, python, deb, and rpms input packages. (#386, #374; patch by Pranay Kanwar)
- gem: Use 'gem' command to download gems and read gem package information. (#389, #394, #378, #233; patches by Pranay Kanwar and Chris Roberts)

- rpm: dashes are now replaced with underscores in rpm version strings (#395, #393, #399; patches by Jeff Terrace and Richard Guest)
- python: Only use the first line of a license; some python packages (like 'requests') embed their full license copy into the license field. For the sake of sanity and function with most packaging systems, fpm only uses the first line of that license.
- rpm: Add new 'none' option to –rpm-compression to disable compression entirely. (#398, patch by Richard Guest)
- deb: Make dependencies using the '!=' operator represented as "Breaks" in the deb package (previously used "Conflicts"). (#400)
- deb: Add md5sums to the debian packages which improves correctness of the package. (#403, #401; patch by Pranay Kanwar)
- rpm: Convert all '!=' dependency operators to 'Conflicts'. Previously, this only applied to packages converting from python to rpm. (#404, #396; patch by Pranay Kanwar)

0.4.31 (March 21, 2013)

- rpm: new flag –rpm-use-file-permissions which try to create an rpm that has file ownership/modes that exactly mirror how they are on the filesystem at package time. (#377, patch by Paul Rhodes)
- general: remove empty directories only when they match the exclude pattern (#323, patch by Pranay Kanwar)

0.4.30 (March 21, 2013)

- Solaris: –solaris-user and –solaris-group flags to specify the owner of files in a package. (#342, patch by Derek Olsen)
- rpm: (bug fix) epoch of 0 is permitted now (#343, patch by Ben Hughes)
- pear: add flags –pear-bin-dir –pear-php-bin –pear-php-dir (#358, patch by Zsolt Takacs)
- New 'source' type: empty. Allows you to create packages without any files in them (sometimes called 'meta packages'). Useful when you want to have one package be simply dependencies or when you want to spoof a package you don't want installed, etc. (#359, 349; patch by Pranay Kanwar)
- solaris: Add –solaris-user and –solaris-group flags (#342, Patch by Derek Olsen)
- gem: new flag –env-shebang; default true (disable with –no-env-shebang). Lets you disable #! (shebang) mangling done by gem installation. (#363, patch by Grier Johnson)
- deb: fix bug on changelog handling (#376, patch by mbakke)
- rpm: fix –rpm-rpmbuild-define (#383, patch by Eric Merritt)

0.4.29 (January 22, 2013)

- Copy links literally, not what they point at (#337, patch by Dane Knecht)

0.4.28 (January 21, 2013)

- Fix a dependency on the 'cabin' gem. (#344, reported by Jay Buffington)

0.4.27 (January 16, 2013)

- Make all fpm output go through the logger (#329; patch by jaybuff)
- New package type: osxpkg, for building packages installable on OS X. (#332, patch by Timothy Sutton)
- Fix crash bug when converting rpms to something else (#316, #325; patch by rtucker-mozilla)
- deb: Add –deb-field for setting a custom field in the control file. For more information on this setting, see section 5.7 "User-defined fields" of the debian policy manual: http://www.debian.org/doc/debian-policy/ch-controlfields.html#s5.7
- deb: Add –deb-recommends and –deb-suggests (#285, #310; patch by Pranay Kanwar)
- python to rpm: convert "!=" dependency operators in python to "Conflicts" in rpm. (#263, #312; patch by Pranay Kanwar)
- python: fix bug - ignore blank lines in requirements.txt (#312, patch by Pranay Kanwar)

0.4.26 (December 27, 2012)

- rpm: add –rpm-sign flag to sign packages using the 'rpmbuild –sign' flag. (#311, Patch by Pranay Kanwar)
- rpm: fix flag ordering when calling rpmbuild (#309, #315, patch by Trotter Cashion)
- deb: re-enable "Predepends" support (#319, #320, patch by Pranay Kanwar)
- rpm: fix default 'rpm os' value (#321, 314, 309)

0.4.25 (December 7, 2012)

- Added –deb-changelog and –rpm-changelog support flags. Both take a path to a changelog file. Both must be valid changelog formats for their respective package types. (#300, patch by Pranay Kanwar)
- deb: Multiple "provides" are now supported. (#301, patch by Pranay Kanwar)
- rpm: Added –rpm-os flag to set the OS target for the rpm. This lets you build rpms for linux on OS X and other platforms (with –rpm-os linux). (#309)
- rpm: Avoid platform-dependent commands in the %install phase (#309, fixes 'cp -d' on OSX)
- python: ignore comments in requirements.txt (#304, patch by Pranay Kanwar)
- Fixed warning 'already initialized constant' (#274)

0.4.24 (November 30, 2012)

- Don't include an empty url in rpm spec (#296, #276; patch by Pranay Kanwar)
- Don't require extra parameters if you use –inputs (#278, #297; Patch by Pranay Kanwar)
- python: supports requirements.txt now for dependency information.
- python: supports pip now. Use '–python-pip path/to/pip' to have fpm use it instead of easy_install.
- solaris: package building works again (#216, #299, patch by Pierre-Yves Ritschard)

0.4.23 (November 26, 2012)

- The –directories flag is now recursive when the output package is rpm. This makes all directories under a given path as owned by the package so they'll be removed when the package is uninstalled (#245, #293, #294, patch by Justin Ellison)
- Add fpm version info to '–help' output (#281)
- gem to rpm: Use 'rubygem(gemname)' for dependencies (#284, patch by Jan Vansteenkiste)
- Fix a bug in gem version mangling (#292, #291; patch by Pranay Kanwar)
- Fix compatibility with Python 2.5 (#279, patch by Denis Bilenko)

0.4.22 (November 15, 2012)

- Add –no-depends flag for creating packages with no dependencies listed (#289, patch by Brett Gailey)
- Fix a bug where blank lines were present in a debian control file. (#288, patch by Andrew Bunday)

0.4.21 (November 8, 2012)

- gem: remove restriction on expected gem names (#287)
- add –directory flag; lets you mark a directory as being owned by a package. (#260, #245, patch by ajf8)
- deb: don't include a version in the Provides field (#280)
- gem: if the version is '1.1' make it imply '1.1.0' (#269, patch by Radim Marek)

0.4.20 (October 5, 2012)

- python: only specify –install-{scripts,lib,data} flags to setup.py if they were given on the command line to fpm. Fixes #273.

0.4.19 (September 26, 2012)

- Escape '%' characters in file names (#266, #222. Patch by John Wittkoski)

0.4.18 (September 25, 2012)

- Fix regression in rpm building where the epoch in was missing in the rpm, but prior fpm versions defaulted it to 1. This caused rpms built with newer fpms to appear "older" than older rpms. Tests added to ensure this regression is caught prior to future releases! (Reported by eliklein)

0.4.17 (September 12, 2012)

- Remove accidental JSON warning when using '-s python'

0.4.16 (September 6, 2012)

- Fix compatibility with Ruby 1.8.7 (broken in 0.4.15)

0.4.15 (September 6, 2012)

- pear: support custom channels with –pear-channel <channel> (#207) Example: fpm -s pear -t deb –pear-channel pear.drush.org drush
- permit literal 'n' in –description, fpm will replace with a newline character. Example: fpm –description "line onenline two" (#251)
- improve error messaging when trying to output a package to a directory that doesn't exist (#244)
- deb: convert '>' and '<' dependency operators to the correct '>>' and '<<' debian version operators (#250, patch by Thomas Meson).
- deb: add –deb-priority flag (#232) for setting the debian 'priority' value for your package.
- add –template-value. Used to expose arbitrary values to script templates. If you do –template-value hello=world, in your template you can do <%= hello %> to get 'world' to show up in your maintainer scripts.
- python: add –python-install-data flag to set the –install-data option to setup.py (#255, patch by Thomas Meson)
- Reject bad dependency flags (ones containing commas) and offer alternative. (#257)
- Try to copy a file if hardlinking fails with permission problems (#253, patch by Jacek Lach)
- Make –exclude, if a directory, include itself and any children, recursive. (#248)

0.4.14 (August 24, 2012)

- rpm: Replace newlines with space in any license setting. (#252)

0.4.13 (August 14, 2012)

- Make –exclude accept path prefixes as well. If you have a files in 'usr/share/man' in your package, you can now exclude all of a subdir by doing '–exclude usr/share/man'

0.4.12 (August 10, 2012)

- Fix a major bug introduced in 0.4.11 that caused all deb packages to contain empty maintainer scripts if not otherwise specified, which made apt/dpkg quite unhappy

0.4.11 (August 7, 2012)

- Fix some symlink handling to prevent links from being followed during cleanup (#228, patch by sbuss)
- rpm: 'vendor' in rpm spec is now omitted if empty or nil. This fixes a bug where rpmbuild fails due to empty 'Vendor' tag if you convert rpm to rpm.
- internal: remove empty directories marked by –exclude (#205, patch by jimbrowne)
- dir: don't try to set utime on symlinks (#234, #240, patch by ctgswallow)
- rpm: relocatable rpms now supported when using the '–prefix' flag. Example: fpm -s dir -t rpm –prefix /usr/local -n example /etc/motd (patch by jkoppe)
- deb: –deb-compression flag: Support different compression methods. Default continues to be gzip.
- new flag: –template-scripts. This lets you write script templates for –after-install, etc. Templates are ERB, so you can do things like '<%= name %>' to get the package name in the script, etc.

- warn on command invocations that appear to have stray flags to try and help users who have complex command lines that are failling.

0.4.10 (May 25, 2012)

- Fix python package support for python3 (#212, patch by Slezhuk Evgeniy)
- Preserve file metadata (time, owner, etc) when copying with the dir package. (#217, patch by Marshall T. Vandegrift)
- Missing executables will now error more readably in fpm.
- Fix gem and python 'version' selection (#215, #204)
- Dependencies using '!=' will now map to 'Conflicts' in deb packages. (#221, patch by Sven Fischer)
- Allow setting default user/group for files in rpm packages (#208, patch by Jason Rogers). Note: This adds –user and –group flags to effect this. These flags may go away in the future, but if they do, they will be
- In python packages set 'install-data' correctly. (#223, patch by Jamie Scheinblum)

0.4.9 (April 25, 2012)

- Fix –prefix support when building gems (#213, patch by Jan Vansteenkiste)

0.4.8 (April 25, 2012)

- RPM: use 'noreplace' option for config files (#194, patch by Steve Lum)
- Python: Fix bug around exact dependency versions (#206, patch by Lars van de Kerkhof)
- Gem->RPM: Make 'provides' "rubygem(thegemname)" instead of "rubygem-thegemname"
- Fix oddity where Ruby would complain about constant redefinition (#198, patch by Marcus Vinicius Ferreira)

0.4.7 skipped.

0.4.6 (April 10, 2012)

- Work around more problems in RPM with respect to file listing (#202)

0.4.5 (April 3, 2012)

- Fix gem->rpm conversion where the '~>' rubygem version operator (#193, patch by antoncohen)
- Escape filenames RPM install process (permits files with spaces, dollar signs, etc) (#196, reported by pspiertz)

0.4.4 (March 30, 2012)

- Fix a bug in gem bin_dir handling (Calen Pennington)
- The –config-files flag should work again (Brian Akins)
- Fix syntax error when using –deb-pre-depends (Andrew Bennett)

- Make –exclude work again (#185, #186) (Calen Pennington)
- Fix file listing so that rpm packages don't declare ownership on / and /usr, etc.
- make –deb-custom-control to work again (Tor Arne Vestbø)
- Add –rpm-digest flag to allow selection of the rpm 'file name' digest algorithm. Default is 'md5' since it works on the most rpm systems.
- Reimplement old behavior assuming "." as the input when using '-s dir' and also setting -C (#187)
- Set BuildRoot on rpm to work around an rpmbuild bug(?) on CentOS 5 (#191)
- Add –rpm-compression flag to allow selection of the rpm payload compression. Default is 'gzip' since it works on the most rpm systems
- Specs now pass on ubuntu/32bit systems (found by travis-ci.org's test runner)
- Improve default values of iteration and epoch (#190)
- Make FPM::Package#files list only 'leaf' nodes (files, empty directories, symlinks, etc).

0.4.3 (March 21, 2012)

- Fix bug in python packaging when invoked with a relative path to a setup.py (Reported by Thomas Meson, https://github.com/jordansissel/fpm/pull/180)

0.4.2 (March 21, 2012)

- Set default temporary directory to /tmp (https://github.com/jordansissel/fpm/issues/174)
- Improve symlink handling (patch by Aleix Conchillo Flaqué, pull/177))
- Python package support changes (thanks to input by Luke Macken):
  - New flag: –python-install-bin. Sets the location for python package scripts (default: /usr/bin)
  - New flag: –python-install-lib. Sets the location for the python package to install libs to, default varies by system. Usually something like /usr/lib/python2.7/site-packages.
  - Fix up –prefix support
  - Improve staged package installation

0.4.1 (March 19, 2012)

- Fix fpm so it works in ruby 1.8 again. Tests run, and passing: rvm 1.8.7,1.9.2,1.9.3 do bundle exec rspec

0.4.0 (March 18, 2012)

- Complete rewrite of pretty much everything.
  - Otherwise, the 'fpm' command functionality should be the same
  - Please let me know if something broke!
- Now has an API (see examples/api directory)
- Also has a proper test suite

- Updated the rpm spec generator to disable all the ways I've found rpmbuild to be weird about packages. This means that fpm-generated rpms will no longer strip libraries, move files around, randomly mutate jar files, etc.
- Add –license and –vendor settings (via Pieter Loubser)
- python support: try to name python packages sanely. Some pypi packages are literally called 'python-foo' so make sure we generate packages named 'python-foo' and not 'python-python-foo' (via Thomas Meson)
- rpm support: Add –rpm-rpmbuild-define for passing a –define flag to rpmbuild (via Naresh V)
- PHP pear source support (fpm -s pear ...) (via Andrew Gaffney)

0.3.10 (Oct 10, 2011)

- Allow taking a list of files/inputs on stdin with '-' or with the –inputs flag. (Matt Patterson)
- (python) pass -U to easy_install (Khalid Goudeaux)
- (debian) quote paths in md5sum calls (Matt Patterson)
- (debian) quiet stderr from dpkg –print-architecture

0.3.9 (Sep 8, 2011)

- Fix bug in 'dir' source that breaks full paths
- Added a bunch of tests (yaay)

0.3.8 and earlier: I have not kept this file up to date very well... Sorry :(

0.2.29 (May 20, 2011)

- Add 'tar' source support. Useful for binary releases to repackage as rpms and debs. Example:

```
fpm -s tar -t rpm -n firefox -v 4.0.1 --prefix /opt/firefox/4.0.1 firefox-4.0.1.
↪tar.bz2
```

0.2.28 (May 18, 2011)

- Use –replaces as "Obsoletes" in rpms.

0.2.27 (May 18, 2011)

- If present, DEBEMAIL and DEBFULLNAME environment variables will be used as the default maintainer. Previously the default was simply <$user@$hostname> https://github.com/jordansissel/fpm/issues/37
- Add '–replaces' flag for specifying packages replaced by the one you are building. This only functions in .deb packages now until I find a suitable synonym in RPM.
- Add –python-bin and –python-easyinstall flags. This lets you choose specific python and easy_install tools to use when building. Default is simply 'python' and 'easy_install' respectively.
- Add support for ~/.fpmrc - The format of this file is the same as the flags. One flag per line. https://github.com/jordansissel/fpm/issues/38. Example:

```
--python-bin=/usr/bin/python2.7
--python-easyinstall=/usr/bin/easy_install2.7
```

0.2.26 and earlier

No changelist tracked. My bad, yo.