Pulp Python Support Documentation

Release 1.0.1

Pulp Project

October 20, 2015

Contents

Welcome to the Pulp Python Plugin documentation. These plugins extend the Pulp Project so that it is capable of handling Python packages. With these plugins, you can create Python repositories in Pulp, upload Python packages to those repositories, and use pip to install packages from client machines.

We plan to add support for more features in the future, and community contributions are welcome. Send us pull requests on our GitHub repository. See existing bugs in the Pulp's Redmine instance or file a new bug.

CHAPTER 1

Release Notes

1.1 1.0 Release Notes

1.1.1 1.0.1

This is a bugfix release. It contains two bugfixes:

- Add support for packages that use DOS line endings
- Add support for packages that contain more than one PKG-INFO file

These fixed were both provided to the Pulp community in Pull Request #40. Thanks to Graham Forest for the contributions!

1.1.2 1.0.0

The Pulp team is pleased to release version 1.0.0 of the Python plugins for Pulp.

Sync Feature

This release introduces the ability to *Synchronize Packages from PyPI*.

Upgrade

To upgrade, simply follow these steps (substituting for systemctl as appropriate, if you are not using systemd):

1. Stop all Pulp services on every machine that is part of the installation:

```
$ for s in {pulp_workers,pulp_resource_manager,pulp_celerybeat,httpd,goferd}; do sudo systemctl
```

2. Upgrade the Pulp packages on every machine:

```
$ sudo yum update
```

3. Apply database migrations:

```
$ sudo -u apache pulp-manage-db
```

4. Start the Pulp services:

```
    $ for s in {pulp_workers,pulp_resource_manager,pulp_celerybeat,httpd,goferd}; do sudo systemctl
```

Bugfixes

This release contains minor bugfixes. See the bugs fixed in 1.0.0.

CHAPTER 2

Administrator Documentation

2.1 Installation

2.1.1 Prerequisites

These instructions assume that you have a working Pulp installation first. If you have not yet installed Pulp, please follow the Pulp installation instructions, and then return to this document.

The command line examples included here are written for systems that use yum as their package manager, and systemd as their init system. Please season to taste if your system is different.

2.1.2 Server

Consider stopping httpd. If you need it to keep running other web apps, or if you need Pulp to continue serving static content, it is usually sufficient to disable access to Pulp's REST API. That will be left as an exercise for the reader. Otherwise, just stop the `httpd` service:

```
$ sudo systemctl stop httpd
```

Next, install the `pulp-python-plugins` package:

```
$ sudo yum install pulp-python-plugins
```

Then run `pulp-manage-db` to initialize the new types in Pulp's database. You must run this command as the same user that the web server uses when it runs Pulp:

```
$ sudo -u apache pulp-manage-db
```

Finally, restart `httpd`:

```
$ sudo systemctl restart httpd
```

2.1.3 Admin Client

Simply install the `pulp-python-admin-extensions` package:

```
$ sudo yum install pulp-python-admin-extensions
```

2.2 Configuration

2.2.1 Python Importer Configuration

The Python importer is configured by editing `/etc/pulp/server/plugins.conf.d/python_importer.json`. This file must be valid JSON.

The importer supports the settings documented in Pulp's importer config docs.

CHAPTER 3

User Documentation

3.1 Getting Started

If you have not yet installed the Python plugins on your Pulp installation, please follow our Installation. This document will assume you have the environment installed and ready to go. We will perform some simple tasks here to get you started by showing you how to create a repository, upload Python packages into it, publish it, and then use pip to install packages from it.

3.1.1 Create a Repository

We will start by making a Python repository:

```
$ pulp-admin python repo create --repo-id my_own_pypi
```

3.1.2 List Repositories

You can list existing Python repositories:

```
$ pulp-admin python repo list
+----------------------------------------------------------------------+
                          Python Repositories
+----------------------------------------------------------------------+

Id:                  my_own_pypi
Display Name:        my_own_pypi
Description:         None
Content Unit Counts:
```

3.1.3 Upload a Python Package

Now that we have a Python repository, we can upload a Python source package to it. Let's clone the pulp_python plugins package and build a source package suitable for uploading to Pulp:

```
$ cd /tmp
$ git clone https://github.com/pulp/pulp_python.git --branch 0.0-dev
$ cd pulp_python/plugins
$ ./setup.py sdist
<output snipped>
```

```
$ ls dist/
pulp_python_plugins-0.0.0.tar.gz
```

That tarball in the `dist/` folder is the package that Pulp expects with its upload command. Let's upload it to Pulp now:

```
$ pulp-admin python repo upload --repo-id my_own_pypi -f dist/pulp_python_plugins-0.0.0.tar.gz
```

And now we can see that there is one Python package in our repository:

```
$ pulp-admin python repo list
+----------------------------------------------------------------------+
                          Python Repositories
+----------------------------------------------------------------------+

Id:                  my_own_pypi
Display Name:        my_own_pypi
Description:         None
Content Unit Counts:
  Python Package: 1
```

3.1.4 Query Packages in a Repository

You can also query the packages in a repository:

```
$ pulp-admin python repo packages --repo-id my_own_pypi --match name=pulp-python-plugins
Name:         pulp-python-plugins
Version:      0.0.0
Author:       Pulp Team
Author Email: pulp-list@redhat.com
Description:  UNKNOWN
Home Page:    http://www.pulpproject.org
License:      GPLv2+
Platform:     UNKNOWN
Summary:      plugins for python support in pulp
```

3.1.5 Publish a Python Repository

The next thing we might want to do once our repository has some content in it is to publish it so that clients can install the package from Pulp:

```
$ pulp-admin python repo publish run --repo-id my_own_pypi
```

3.1.6 Install a Package From a Pulp Hosted Python Repository

We will now install our package on another machine using pip:

```
$ pip install -i http://pulp.example.com/pulp/python/web/my_own_pypi/simple/ pulp-python-plugins
Downloading/unpacking pulp-python-plugins
  Downloading pulp-python-plugins-0.0.0.tar.gz
  Running setup.py egg_info for package pulp-python-plugins

Installing collected packages: pulp-python-plugins
  Running setup.py install for pulp-python-plugins
```

```
Successfully installed pulp-python-plugins
Cleaning up...
```

3.1.7 Remove Python Packages From a Pulp Python Repository

Occasionally, we may want to remove uploaded packages from the repository:

```
$ pulp-admin python repo remove --repo-id my_own_pypi --str-eq="name=pulp-python-plugins"
This command may be exited via ctrl+c without affecting the request.


[\]
Running...

Units Removed:
  pulp-python-plugins-0.0.0
```

Note that this only removes the association of given packages with the repository. Uploaded packages still exist on the server. Python packages which are not associated with any repositories can be removed from the server using *pulp-admin orphan remove –type python_package* command.

3.1.8 Synchronize Packages from PyPI

It is possible to synchronize packages from the Python Package Index. In order to do this, you must specify the feed URL as well as a comma separated list of package names you wish to sync:

```
$ pulp-admin python repo create --repo-id pypi --feed https://pypi.python.org/ --package-names numpy,
Repository [pypi] successfully created

$ pulp-admin python repo sync run --repo-id pypi
+----------------------------------------------------------------------+
                    Synchronizing Repository [pypi]
+----------------------------------------------------------------------+

This command may be exited via ctrl+c without affecting the request.


Downloading and processing metadata.
[-]
... completed

Downloading and processing Python packages.
[==================================================] 100%
30 of 30 items
... completed


Task Succeeded
```

CHAPTER 4

Reference Documentation

4.1 Importer Reference

The Python importer supports the standard Pulp importer keys, as well as one custom config key:

package_names: This key is a comma separated list of the names of the packages that should be synchronized from the feed URL.

4.2 Python Type

The Python plugins come with a database type for Python packages. This type's id is `python_package`, and it has the following attributes:

4.2.1 Unit Key

The Python type's unit key is an ordered list of the following attributes:

Name	Description
name	The name of the package
version	The version of the package

4.2.2 Other Attributes

The Python package type has these additional attributes that are all taken from the package's PKG-INFO file:

Name	Description
summary	A brief summary
home_page	The package's home page URL
author	The author's name
author_email	The author's e-mail address
license	The package's licence type
description	A long description of the package
platform	The platforms that the package is intended to work in

CHAPTER 5

Indices and tables

- genindex
- modindex
- search