Pulp RPM Support Documentation

Release 3.0.0b8.dev

Pulp Team

Nov 13, 2019

Contents

The `pulp_rpm` plugin extends pulpcore to support hosting RPM family content types.

CHAPTER 1

Features

- *Create, Sync and Publish a Repository* with "RPM Content" including RPMs and Errata
- *Versioned Repositories* so every operation is a restorable snapshot
- *Download content on-demand* when requested by clients to reduce disk space.
- Upload local RPM content *easily*
- Copy and organize RPM content into various repositories
- De-duplication of all saved content
- Host content either locally or on S3

CHAPTER 2

Get Started

To get started, check the *installation docs* and take a look at the *basic workflows*.

Community contributions are encouraged.

- Send us pull requests on our GitHub repository.
- View and file issues in the Redmine Tracker.

CHAPTER 3

REST API

REST API documentation for the RPM plugin can be found here

CHAPTER 4

Table of Contents

4.1 User setup

4.1.1 Install `pulpcore`

Follow the installation instructions provided with pulpcore.

4.1.2 Install `pulp_rpm`

Users should install from **either** PyPI or source or use ansible-pulp installer.

Install with Ansible-pulp

With the use of the `ansible-pulp` installer you need to download a supportive role before you run installer.

Only Fedora 29+ and CentOS 7 (with epel repository and python36) are supported.

```
git clone https://github.com/pulp/ansible-pulp.git
cd ansible-pulp
ansible-galaxy install pulp.pulp_rpm_prerequisites -p ./roles/
```

Then use role you downloaded **before** ansible-pulp installer roles. Do not forget to set `pulp_use_system_wide_pkgs` to `true`.

```
---
- hosts: all
  vars:
    pulp_secret_key: secret
    pulp_default_admin_password: password
    pulp_use_system_wide_pkgs: true
    pulp_install_plugins:
```

(continues on next page)

(continued from previous page)

```
      pulp-rpm:
        prereq_role: "pulp.pulp_rpm_prerequisites"
  roles:
    - pulp-database
    - pulp-workers
    - pulp-resource-manager
    - pulp-webserver
    - pulp-content
  environment:
    DJANGO_SETTINGS_MODULE: pulpcore.app.settings
```

Now you can run installer against your desired host following instructions in the ansible-pulp installer.

Install `createrepo_c` from source

`pulp_rpm` depends on a Python package named `createrepo_c`, which is compiled from a C library. Unfortunately, this package is currently only available as a Python "source distribution", meaning that it must be compiled on your own machine. But, luckily, you won't have to do that yourself! Simply install the build dependencies on your machine and the build process itself will happen behind the scenes when you install the package.

Caveat: Unfortunately, a fully-featured `createrepo_c` can only be built on Red Hat based distros, as not all of build dependencies are available on Debian-based platforms.

If you are on Fedora, install the build dependencies with this command:

```
sudo dnf install -y gcc make cmake bzip2-devel expat-devel file-devel glib2-devel␣
↪libcurl-devel libmodulemd-devel libxml2-devel python3-devel rpm-devel openssl-devel␣
↪sqlite-devel xz-devel zchunk-devel zlib-devel
```

If on CentOS or RHEL, use this command:

```
sudo yum install -y gcc make cmake bzip2-devel expat-devel file-devel glib2-devel␣
↪libcurl-devel libmodulemd-devel libxml2-devel python36-devel rpm-devel openssl-
↪devel sqlite-devel xz-devel zchunk-devel zlib-devel
```

Install `pulp_rpm` from source

```
sudo -u pulp -i
source ~/pulpvenv/bin/activate
git clone https://github.com/pulp/pulp_rpm.git
cd pulp_rpm
pip install -e .
```

Install `pulp-rpm` From PyPI

```
sudo -u pulp -i
source ~/pulpvenv/bin/activate
pip install pulp-rpm
```

Run Migrations

```
django-admin migrate rpm
```

4.1.3 Run Services

```
django-admin runserver 24817
gunicorn pulpcore.content:server --bind 'localhost:24816' --worker-class 'aiohttp.
↪GunicornWebWorker' -w 2
sudo systemctl restart pulpcore-resource-manager
sudo systemctl restart pulpcore-worker@1
sudo systemctl restart pulpcore-worker@2
```

4.2 Quickstart

4.2.1 Run Pulplift

Use pulplift to try out Pulp. From their quickstart pulplift uses Vagrant so you'll need to have that installed.

```
git clone --recurse-submodules https://github.com/pulp/pulplift.git
cd pulplift
```

Create your pulp_rpm.yaml playbook with these contents:

```
---
- hosts: all
  vars:
    pulp_secret_key: secret
    pulp_default_admin_password: password
    pulp_install_plugins:
      pulp-rpm:
        prereq_role: "pulp.pulp_rpm_prerequisites"
  roles:
    - pulp-database
    - pulp-workers
    - pulp-resource-manager
    - pulp-webserver
    - pulp-content
  environment:
    DJANGO_SETTINGS_MODULE: pulpcore.app.settings
```

Install the dependency bootstrapping role `pulp.pulp_rpm_prerequisites`:

ansible-galaxy install pulp.pulp_rpm_prerequisites -p ./roles/

Start your box, run ansible on it, ssh to your box:

```
vagrant up fedora30
ansible-playbook pulp_rpm.yaml -l fedora30
vagrant ssh fedora30
```

4.2.2 Check Pulp's Status

Check the status API using `httpie`:

```
sudo dnf install httpie -y
http :24817/pulp/api/v3/status/  # This should show the status API
```

4.2.3 Next Steps

- Sync rpms from the zoo repo using the *Create, Sync and Publish a Repository*.
- Upload or mirror content using the *rpm one-shot uploader*.
- Configure a client to *download packages* from a RPM repository hosted by Pulp.

4.3 Workflows

All REST API examples below use httpie to perform the requests. The `httpie` commands below assume that the user executing the commands has a `.netrc` file in the home directory. The `.netrc` should have the following configuration:

```
machine localhost
login admin
password password
```

If you configured the `admin` user with a different password, adjust the configuration accordingly. If you prefer to specify the username and password with each request, please see `httpie` documentation on how to do that.

This documentation makes use of the jq library to parse the json received from requests, in order to get the unique urls generated when objects are created. To follow this documentation as-is please install the jq library with:

```
$ sudo dnf install jq
```

4.3.1 Create, Sync and Publish a Repository

One of the most common workflows is a fetching content from a remote source and making it available for users.

Create an RPM repository `foo`

```
$ http POST http://localhost:24817/pulp/api/v3/repositories/rpm/rpm/ name=foo
```

```
{
    "pulp_href": "/pulp/api/v3/repositories/rpm/rpm/5eeabc0b-3b86-4264-bb3a-
↪5889530a6f5b/",
}
```

```
$ export REPO_HREF=$(http :24817/pulp/api/v3/repositories/rpm/rpm/ | jq -r
'.results[] | select( .name == "foo") | .pulp_href')
```

Create a new remote `bar`

By default `policy='immediate` which means that all the content is downloaded right away. Specify `policy='on_demand'` to make synchronization of a repository faster and only to download RPMs whenever they are requested by clients.

```
$ http POST http://localhost:24817/pulp/api/v3/remotes/rpm/rpm/ name='bar'
url='https://repos.fedorapeople.org/pulp/pulp/fixtures/rpm-unsigned/'
policy='on_demand'
```

```
{
    "pulp_href": "/pulp/api/v3/remotes/rpm/rpm/378711cd-1bee-4adc-8d9b-fe3bceaba39f/",
}
```

```
$ export REMOTE_HREF=$(http :24817/pulp/api/v3/remotes/rpm/rpm/ | jq -r '.
results[] | select(.name == "bar") | .pulp_href')
```

Sync repository `foo` using remote `bar`

```
$ http POST :24817${REPO_HREF}sync/ remote=$REMOTE_HREF
```

Look at the new Repository Version created

```
$ http GET :24817${REPO_HREF}versions/1/
```

```
{
    "_added_href": "/pulp/api/v3/repositories/rpm/rpm/5eeabc0b-3b86-4264-bb3a-
↪5889530a6f5b/versions/1/added_content/",
    "_content_href": "/pulp/api/v3/repositories/5eeabc0b-3b86-4264-bb3a-5889530a6f5b/
↪rpm/rpm/versions/1/content/",
    "pulp_href": "/pulp/api/v3/repositories/5eeabc0b-3b86-4264-bb3a-5889530a6f5b/rpm/
↪rpm/versions/1/",
    "_removed_href": "/pulp/api/v3/repositories/5eeabc0b-3b86-4264-bb3a-5889530a6f5b/
↪rpm/rpm/versions/1/removed_content/",
    "content_summary": {
        "package": 35,
        "advisory": 4
    },
    "created": "2018-02-23T20:29:54.499055Z",
    "number": 1
}
```

Create a Publication

```
$ http POST :24817/pulp/api/v3/publications/rpm/rpm/ repository=$REPO_HREF
```

```
[
    {
        "pulp_href": "/pulp/api/v3/tasks/fd4cbecd-6c6a-4197-9cbe-4e45b0516309/",
        "task_id": "fd4cbecd-6c6a-4197-9cbe-4e45b0516309"
    }
]
```

```
$ export PUBLICATION_HREF=$(http :24817/pulp/api/v3/publications/rpm/rpm/
| jq -r '.results[] | select(.repository_version|test("'$REPO_HREF'.")) | .
pulp_href')
```

Create a Distribution for the Publication

```
$ http POST http://localhost:24817/pulp/api/v3/distributions/rpm/rpm/
name='baz' base_path='foo' publication=$PUBLICATION_HREF
```

```
{
    "pulp_href": "/pulp/api/v3/distributions/8f394d20-f6fb-49dd-af0e-778225d79442/",
}
```

4.3.2 Upload and Manage Content

Content can be added to a repository not only by synchronizing from a remote source but also by uploading.

Upload `foo.rpm`

Create an Artifact by uploading the file to Pulp.

```
$ http --form POST http://localhost:24817/pulp/api/v3/artifacts/ file@./foo-4.
1-1.noarch.rpm
```

```
{
    "pulp_href": "/pulp/api/v3/artifacts/d1dd56aa-c236-414a-894f-b3d9334d2e73/",
}
```

Create `rpm` content from an Artifact

Create a content unit and point it to your artifact

```
$ http POST http://localhost:24817/pulp/api/v3/content/rpm/packages/
artifact="/pulp/api/v3/artifacts/d1dd56aa-c236-414a-894f-b3d9334d2e73/"
relative_path=foo-4.1-1.noarch.rpm
```

```
{
    "pulp_href": "/pulp/api/v3/content/rpm/packages/2df123b2-0d38-4a43-9b21-
↪a3e830ea1324/",
    "artifact": "/pulp/api/v3/artifacts/d1dd56aa-c236-414a-894f-b3d9334d2e73/",
}
```

```
$ export CONTENT_HREF=$(http :24817/pulp/api/v3/content/rpm/packages/ | jq -r
'.results[] | select( .location_href == "foo-4.1-1.noarch.rpm") | .pulp_href')
```

Add content to repository `foo`

```
$ http POST :24817${REPO_HREF}modify/ add_content_units:="[\"$CONTENT_HREF\"]"
```

One shot upload

You can use one shot uploader to upload one rpm and optionally create new repository version with rpm you uploaded. With this call you can substitute previous two (or three) steps (create artifact, content from artifact and optionally add content to repo).

```
http --form POST http://localhost:24817/pulp/api/v3/rpm/upload/ file@./foo-1.
0-1.noarch.rpm repository=${REPO_HREF}
```

```
{
   "pulp_href": "/pulp/api/v3/tasks/f2b525e3-8d8f-4246-adab-2fabd2b089a8/",
   "created_resources": [
       "/pulp/api/v3/content/rpm/packages/1edf8d4e-4293-4b66-93cd-8e913731c87a/",
       "/pulp/api/v3/repositories/rpm/rpm/64bdeb44-c6d3-4ed7-9c5a-94b264a6b7b5/
↪versions/2/"
   ],
}
```

4.3.3 Get content from Pulp

When content is published and distributed, one can download packages directly from Pulp or point client tools to a repo distributed by Pulp.

Download `foo.rpm` from Pulp

```
$ http GET http://localhost:24816/pulp/content/foo/foo.rpm
```

Install a package from Pulp

Open /etc/yum.repos.d/foo.repo and add the following:

```
[foo]
name = foo
baseurl = http://localhost:24816/pulp/content/foo/
gpgcheck = 0
```

Now use dnf to install a package:

```
$ sudo dnf install walrus
```

List and Install applicable Advisories

Make sure Pulp repo is configured in /etc/yum.repos.d/, then use dnf to work with Advisory content.

List applicable Advisories:

```
$ dnf list-sec
```

Install a specific advisory:

```
sudo dnf update --advisory XXXX-XXXX:XXXX
```

4.3.4 Copy RPM content between repositories

If you want to copy RPM content from one repository into another repository, you can do so.

Copy workflow

You can use the modify endpoint on a repository to copy all content present in one repository version to another by specifying a base version.

```
http POST http://localhost:24817${REPO_HREF}modify/ base_version=${BASE_REPO_HREF}
```

```
{
    "created_resources": [
        "/pulp/api/v3/repositories/rpm/rpm/5fc8a78e-068a-47c0-b728-a5475042573a/
↪versions/2/"
    ],
}
```

4.4 Client Bindings

4.4.1 Python Client

The pulp-rpm-client package on PyPI provides bindings for all API calls in the pulp_rpm API documentation. It is currently published daily and with every RC.

The pulpcore-client package on PyPI provides bindings for all API calls in the pulpcore API documentation. It is currently published daily and with every RC.

4.4.2 Ruby Client

The pulp_rpm_client Ruby Gem on rubygems.org provides bindings for all API calls in the pulp_rpm API documentation. It is currently published daily and with every RC.

The pulpcore_client Ruby Gem on rubygems.org provides bindings for all API calls in the pulpcore API documentation. It is currently published daily and with every RC.

4.4.3 Client in a language of your choice

A client can be generated using Pulp's OpenAPI schema and any of the available generators.

Generating a client is a two step process:

1) Download the OpenAPI schema for pulpcore and all installed plugins:

```
curl -o api.json http://<pulp-hostname>:24817/pulp/api/v3/docs/api.json
```

The OpenAPI schema for a specific plugin can be downloaded by specifying the plugin's module name as a GET parameter. For example for `pulp_rpm` only endpoints use a query like this:

```
curl -o api.json http://<pulp-hostname>:24817/pulp/api/v3/docs/api.json?plugin=pulp_
↪rpm
```

2) Generate a client using openapi-generator.

The schema can then be used as input to the openapi-generator-cli. The documentation on getting started with openapi-generator-cli is available on openapi-generator.tech.

4.5 Changelog

4.5.1 3.0.0b7 (2019-10-16)

Features

- Convert all the TextFields which store JSON content into Django JSONFields. #5215

Improved Documentation

- Change the prefix of Pulp services from pulp-* to pulpcore-* #4554
- Docs update to use *pulp_use_system_wide_pkgs*. #5488

Deprecations and Removals

- Change *_id*, *_created*, *_last_updated*, *_href* to *pulp_id*, *pulp_created*, *pulp_last_updated*, *pulp_href* #5457
- Removing *repository* from *Addon*/*Variant* serializers. #5516
- Moved endpoints for distribution trees and repo metadata files to /pulp/api/v3/content/rpm/distribution_trees/ and /pulp/api/v3/content/rpm/repo_metadata_files/ respectively. #5535
- Remove "_" from *_versions_href*, *_latest_version_href* #5548

4.5.2 3.0.0b6 (2019-09-30)

Features

- Add upload functionality to the rpm contents endpoints. #5453
- Synchronize and publish modular content. #5493

Bugfixes

- Add url prefix to plugin custom urls. #5330

Deprecations and Removals

- Removing *pulp/api/v3/rpm/upload/* #5453

Misc

- #5172, #5304, #5408, #5421, #5469, #5493

4.5.3 3.0.0b5 (2019-09-17)

Features

- Setting *code* on *ProgressBar*. #5184
- Sync and Publish kickstart trees. #5206
- Sync and Publish custom/unknown repository metadata. #5432

Bugfixes

- Use the field relative_path instead of filename in the API calls while creating a content from an artifact #4987
- Fixing sync task failure. #5285

Misc

- #4681, #5201, #5202, #5331, #5430, #5431, #5438

4.5.4 3.0.0b4 (2019-07-03)

Features

- Add total counts to the sync progress report. #4503
- Greatly speed up publishing of a repository. #4591
- Add ability to copy content between repositories, content type(s) can be specified. #4716
- Renamed Errata/Update content to Advisory to better match the terminology of the RPM/DNF ecosystem. #4902
- Python bindings are now published nightly and with each release as pulp-rpm-client. Also Ruby bindings are published similarly to rubygems.org as pulp_rpm_client. #4960
- Override the Remote's serializer to allow policy='on_demand' and policy='streamed'. #5065

Bugfixes

- Require relative_path at the content unit creation time. #4835
- Fix migraitons failure by making models compatible with MariaDB. #4909
- Fix unique index length issue for MariaDB. #4916

Improved Documentation

- Switch to using towncrier for better release notes. #4875
- Add a docs page about the Python and Ruby bindings. #4960

Misc

- #4117, #4567, #4574, #5064

4.5.5 3.0.0b3

- Improve documentation, including *ansible installation* and REST API docs
- Improve support of Errata content
- Bug fixes
- Compatibility with pulpcore-plugin-0.1.0rc2

Comprehensive list of changes and bugfixes for beta 3.

4.5.6 3.0.0b2

- Add support for on_demand sync
- Add one-shot upload
- Performance improvements and bug fixes
- Compatibility with pulpcore-plugin-0.1.0rc1

Comprehensive list of changes and bugfixes for beta 2.

4.5.7 3.0.0b1

- Add support for basic sync/copy/publish of RPM packages and Errata content.

  Check our documentation for the *basic workflows*.

4.6 Contributing

To contribute to the `pulp_rpm` package follow this process:

1. Clone the GitHub repo
2. Make a change
3. Make sure all tests passed
4. Add a file into CHANGES folder (Changelog update).
5. Commit changes to own `pulp_rpm` clone
6. Make pull request from github page for your clone against master branch

4.6.1 Changelog update

The CHANGES.rst file is managed using the towncrier tool and all non trivial changes must be accompanied by a news entry.

To add an entry to the news file, you first need an issue in pulp.plan.io describing the change you want to make. Once you have an issue, take its number and create a file inside of the `CHANGES/` directory named after that issue number with an extension of .feature, .bugfix, .doc, .removal, or .misc. So if your issue is 3543 and it fixes a bug, you would create the file `CHANGES/3543.bugfix`.

PRs can span multiple categories by creating multiple files (for instance, if you added a feature and deprecated an old feature at the same time, you would create CHANGES/NNNN.feature and CHANGES/NNNN.removal). Likewise if a PR touches multiple issues/PRs you may create a file for each of them with the exact same contents and Towncrier will deduplicate them.

The contents of this file are reStructuredText formatted text that will be used as the content of the news file entry. You do not need to reference the issue or PR numbers here as towncrier will automatically add a reference to all of the affected issues when rendering the news file.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search