Pulp File Support Documentation

Release 1.1.0.dev

Pulp Project

Jun 02, 2020

Contents

This is the `pulp_file` Plugin for Pulp Project 3.0+. This plugin replaces the ISO support in the `pulp_rpm` plugin for Pulp 2.

CHAPTER 1

How to use these docs

The documentation here should be considered **the primary documentation for managing File content**. See How to navigate the pulpcore and plugin docs for a more thorough explanation.

All relevent workflows are covered here, with references to pulpcore supplemental docs. Users may also find pulpcore's conceptual docs helpful. Here, the documentation falls into two main categories:

1. *Workflows* show the **major features** of the File plugin, with links to reference docs.
2. REST API Docs are automatically generated and are responsible for containing thorough information for each **minor feature**, including all fields and options.

CHAPTER 2

Community

This plugin exists to serve the community. If we can do more for your use case, please let us know! Also, contributions are greatly appreciated in the form of:

1. Redmine Issues
2. Github Pull Requests
3. Helping other users

We can usually be found on freenode in *#pulp-dev* and *#pulp*.

CHAPTER 3

Table of Contents

3.1 User Setup

3.1.1 Ansible Installer (Recommended)

We recommend that you install *pulpcore* and *pulp-file* together using the Ansible installer. The remaining steps are all performed by the installer and are not needed if you use it.

3.1.2 Pip Install

This document assumes that you have installed pulpcore into a the virtual environment `pulpvenv`.

Users should install from **either** PyPI or source.

From PyPI

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

From Source

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

3.1.3 Make and Run Migrations

```
export DJANGO_SETTINGS_MODULE=pulpcore.app.settings
django-admin makemigrations file
django-admin migrate file
```

3.1.4 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
```

3.2 Workflows

This section will document the **major features** of *pulp-file* in a "quickstart" style. More detailed information (REST API Reference) is linked in each section.

If you have not yet installed Pulp and the File plugin, please follow our *User Setup*. These documents will assume you have the environment installed and ready to go.

httpie

The REST API examples here 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.

jq

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
```

3.2.1 Scripting

Each workflow renders bash scripts that allow the developers to ensure the continued correctness of the instructions. These scripts may also be helpful to users as a basis for their own scripts. All of the scripts can be found at https://github.com/pulp/pulp_file/tree/master/docs/_scripts/

The following scripts are used in conjunction with all the workflow scripts:

Base

```
#!/usr/bin/env bash
set -e

echo "Setting environment variables for default hostname/port for the API and the␣
↪Content app"
export BASE_ADDR=${BASE_ADDR:-http://localhost:24817}
export CONTENT_ADDR=${CONTENT_ADDR:-http://localhost:24816}

# Necessary for `django-admin`
export DJANGO_SETTINGS_MODULE=pulpcore.app.settings

# Poll a Pulp task until it is finished.
wait_until_task_finished() {
    echo "Polling the task until it has reached a final state."
    local task_url=$1
    while true
    do
        local response=$(http $task_url)
        local state=$(jq -r .state <<< ${response})
        jq . <<< "${response}"
        case ${state} in
            failed|canceled)
                echo "Task in final state: ${state}"
                exit 1
                ;;
            completed)
                echo "$task_url complete."
                break
                ;;
            *)
                echo "Still waiting..."
                sleep 1
                ;;
        esac
    done
}
```

Correctness Check (Destructive)

To check the correctness of the sync and publish workflow scripts, they can all be run together using:

```
#!/usr/bin/env bash

# This script will execute the component scripts and ensure that the documented␣
↪examples
# work as expected.

# From the _scripts directory, run with `source docs_check_sync_publish.sh` (source␣
↪to preserve the
# environment variables)
source base.sh

source repo.sh
source remote.sh
source sync.sh
```

(continues on next page)

(continued from previous page)

```
source publication.sh
source distribution.sh
source download_after_sync.sh
```

To check the correctness of the upload and publish workflow scripts, they can all be run together using: script.

```
#!/usr/bin/env bash

# This script will execute the component scripts and ensure that the documented␣
↪examples
# work as expected.

# NOTE: These scripts use httpie and requires a .netrc for authentication with Pulp

# From the _scripts directory, run with `source docs_check_upload_publish.sh` (source␣
↪to preserve
# the environment variables)
source base.sh

source repo.sh
source artifact.sh
source content.sh
source add_remove.sh

source publication.sh
source distribution.sh
source download_after_upload.sh
```

3.2.2 Synchronize a Repository

In this section, there is provided a basic workflow for synchronizing a remote repository. Bear in mind that the attached snippets utilize `httpie` and `jq`. Refer to *Workflows* to learn more about these utilities.

Create a repository `foo`

```
#!/usr/bin/env bash
export REPO_NAME=$(head /dev/urandom | tr -dc a-z | head -c5)

echo "Creating a new repository named $REPO_NAME."
export REPO_HREF=$(http POST $BASE_ADDR/pulp/api/v3/repositories/file/file/ name=
↪$REPO_NAME \
  | jq -r '.pulp_href')

echo "Inspecting repository."
http $BASE_ADDR$REPO_HREF
```

Repository GET Response:

```
{
    "pulp_created": "2019-05-16T19:23:55.224096Z",
    "pulp_href": "/pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-
↪436b03285e4c/",
    "latest_version_href": null,
```

(continues on next page)

(continued from previous page)

```
    "versions_href": "/pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-
↪436b03285e4c/versions/",
    "description": "",
    "name": "foo"
}
```

Reference (pulpcore): Repository API Usage

Create a new remote `bar`

```
#!/usr/bin/env bash
echo "Creating a remote that points to an external source of files."
http POST $BASE_ADDR/pulp/api/v3/remotes/file/file/ \
    name='bar' \
    url='https://repos.fedorapeople.org/pulp/pulp/demo_repos/test_file_repo/PULP_
↪MANIFEST'

echo "Export an environment variable for the new remote URI."
export REMOTE_HREF=$(http $BASE_ADDR/pulp/api/v3/remotes/file/file/ | jq -r '.
↪results[] | select(.name == "bar") | .pulp_href')

echo "Inspecting new Remote."
http $BASE_ADDR$REMOTE_HREF
```

Remote GET Response:

```
{
    "pulp_created": "2019-05-16T19:23:56.771326Z",
    "pulp_href": "/pulp/api/v3/remotes/file/file/e682efef-3974-4366-aece-a333bfaec9f3/
↪",
    "pulp_last_updated": "2019-05-16T19:23:56.771341Z",
    "download_concurrency": 20,
    "name": "bar",
    "policy": "immediate",
    "proxy_url": "",
    "ssl_ca_certificate": null,
    "ssl_client_certificate": null,
    "ssl_client_key": null,
    "ssl_validation": true,
    "url": "https://repos.fedorapeople.org/pulp/pulp/demo_repos/test_file_repo/PULP_
↪MANIFEST",
    "validate": true
}
```

Reference: File Remote Usage

Sync repository `foo` using remote `bar`

```
#!/usr/bin/env bash

echo "Create a task to sync the repository using the remote."
export TASK_URL=$(http POST $BASE_ADDR$REPO_HREF'sync/' remote=$REMOTE_HREF␣
↪mirror=False \
  | jq -r '.task')
```

(continues on next page)

(continued from previous page)

```
# Poll the task (here we use a function defined in docs/_scripts/base.sh)
wait_until_task_finished $BASE_ADDR$TASK_URL

# After the task is complete, it gives us a new repository version
echo "Set REPOVERSION_HREF from finished task."
export REPOVERSION_HREF=$(http $BASE_ADDR$TASK_URL| jq -r '.created_resources | first
↪')

echo "Inspecting RepositoryVersion."
http $BASE_ADDR$REPOVERSION_HREF
```

Repository Version GET Response (when complete):


```
{
    "pulp_created": "2019-05-16T19:23:58.230896Z",
    "pulp_href": "/pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-
↪436b03285e4c/versions/1/",
    "base_version": null,
    "content_summary": {
        "added": {
            "file.file": {
                "count": 3,
                "href": "/pulp/api/v3/content/file/files/?repository_version_added=/
↪pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-436b03285e4c/versions/1/"
            }
        },
        "present": {
            "file.file": {
                "count": 3,
                "href": "/pulp/api/v3/content/file/files/?repository_version=/pulp/
↪api/v3/repositories/file/file/680f18e7-0513-461f-b067-436b03285e4c/versions/1/"
            }
        },
        "removed": {}
    },
    "number": 1
}
```


Reference: File Sync Usage

Reference (pulpcore): Repository Version API Usage

3.2.3 Upload Content

The section shows how to upload content to Pulp. The attached snippets use the utilities `httpie` and `jq`. Learn more about them at *Workflows*.

Upload a file to Pulp (Create an Artifact)

```
#!/usr/bin/env bash

echo "Creating a dummy file at path FILE_CONTENT to upload."
export FILE_CONTENT=$(head /dev/urandom | tr -dc a-z | head -c10)
```

(continues on next page)

(continued from previous page)

```
echo $FILE_CONTENT > test_upload.txt

echo "Uploading the file to Pulp, creating an artifact, storing ARTIFACT_HREF."
export ARTIFACT_HREF=$(http --form POST $BASE_ADDR/pulp/api/v3/artifacts/ \
    file@./test_upload.txt \
    | jq -r '.pulp_href')

echo "Inspecting new artifact."
http $BASE_ADDR$ARTIFACT_HREF
```

Artifact GET Response:

```
{
    "pulp_created": "2019-05-16T20:07:48.066089Z",
    "pulp_href": "/pulp/api/v3/artifacts/cff8078a-826f-4f7e-930d-422c2f134a07/",
    "file": "artifact/97/
↪144ab16c9aa0e6072d471d6aebe7c21083e21359137e676445bfeb4051ba25",
    "md5": "5148c996f375ed5aab94ef6993df90a0",
    "sha1": "a7bd2bcaf1d68505f3e8b2cfe3505d01b31db306",
    "sha224": "18a167922b68a3fb8f2d9a71fa78f9776f5402dce4b3d97d5cea2559",
    "sha256": "97144ab16c9aa0e6072d471d6aebe7c21083e21359137e676445bfeb4051ba25",
    "sha384":
↪"4cd006bfac7f2e41baa8c411536579b134daeb3ad666310d21463f384a7020360703fc5538b4eca724033498d514e144
↪",
    "sha512":
↪"e1aae6bbc6fd24cf890b82ffa824629518e6e93935935a0b7c008fbd9fa59f08aa32a7d8580b31a65b21caa0f48e737d8e
↪",
    "size": 11
}
```

Reference (pulpcore): Artifact API Usage

Create `file` content from an Artifact

```
#!/usr/bin/env bash

echo 'Create File Content from the artifact and save as environment variable'
export TASK_URL=$(http POST $BASE_ADDR/pulp/api/v3/content/file/files/ \
    relative_path="test_upload.txt" \
    artifact=$ARTIFACT_HREF \
    | jq -r '.task')

# Poll the task (here we use a function defined in docs/_scripts/base.sh)
wait_until_task_finished $BASE_ADDR$TASK_URL

# After the task is complete, it gives us a new content
echo "Set CONTENT_HREF from finished task."
export CONTENT_HREF=$(http $BASE_ADDR$TASK_URL| jq -r '.created_resources | first')

echo "Inspecting new file content"
http $BASE_ADDR$CONTENT_HREF
```

Content GET Response:

```
{
    "_artifact": "/pulp/api/v3/artifacts/cff8078a-826f-4f7e-930d-422c2f134a07/",
    "pulp_created": "2019-05-16T20:07:48.929374Z",
    "pulp_href": "/pulp/api/v3/content/file/files/c23def43-44bc-45f4-8a6f-
↪0310285f5339/",
    "md5": "3b4fd267e71a1a8e8746893fcc91e5b5",
    "relative_path": "test_upload.txt",
    "sha1": "6fd76062f3680be44de9ed6a4b80bdce512dd620",
    "sha224": "3b518d0e428c4e5996ec6861960a7640770bc8bbbe16775b1dfc1e81",
    "sha256": "b671500c402128babf4f4e51afc552584df3db501bb1a0bd3ee96dc121228a9c",
    "sha384":
↪"7f90e3b612defd1d85d51c3a0efca932fb1a6cdee4a11dd532edd8302dfe7860d3ce4d50b2ed73d984a83e6c6265e54b
↪",
    "sha512":
↪"91c823e2d547e4073d4c47f73f36122acf60a722100d6e592a68aae0b6ba8ee12cd40dbed75cade7ad1a1f7e197a06ed2a
↪"
}
```

Reference: File Content API Usage

Create a repository `foo`

```
#!/usr/bin/env bash
export REPO_NAME=$(head /dev/urandom | tr -dc a-z | head -c5)

echo "Creating a new repository named $REPO_NAME."
export REPO_HREF=$(http POST $BASE_ADDR/pulp/api/v3/repositories/file/file/ name=
↪$REPO_NAME \
  | jq -r '.pulp_href')

echo "Inspecting repository."
http $BASE_ADDR$REPO_HREF
```

Repository GET Response:

```
{
    "pulp_created": "2019-05-16T19:23:55.224096Z",
    "pulp_href": "/pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-
↪436b03285e4c/",
    "latest_version_href": null,
    "versions_href": "/pulp/api/v3/repositories/file/file/680f18e7-0513-461f-b067-
↪436b03285e4c/versions/",
    "description": null,
    "name": "foo"
}
```

Reference (pulpcore): Repository API Usage

Add content to repository `foo`

```
#!/usr/bin/env bash

echo "Kick off a task to add content to a repository, storing TASK_URL env variable"
export TASK_URL=$(http POST $BASE_ADDR$REPO_HREF'modify/' \
```

(continues on next page)

(continued from previous page)

```
    add_content_units:="[\"$CONTENT_HREF\"]" \
    | jq -r '.task')

# Poll the task (here we use a function defined in docs/_scripts/base.sh)
wait_until_task_finished $BASE_ADDR$TASK_URL

echo "Retrieving REPOVERSION_HREF from task"
export REPOVERSION_HREF=$(http $BASE_ADDR$TASK_URL| jq -r '.created_resources | first
↪')

echo "Inspecting repository version."
http $BASE_ADDR$REPOVERSION_HREF
```

Repository Version GET Response:


```
{
    "pulp_created": "2019-05-16T20:07:50.363735Z",
    "pulp_href": "/pulp/api/v3/repositories/file/file/0d908664-e300-4223-869b-
↪fc5d2cef285f/versions/1/",
    "base_version": null,
    "content_summary": {
        "added": {
            "file.file": {
                "count": 1,
                "href": "/pulp/api/v3/content/file/files/?repository_version_added=/
↪pulp/api/v3/repositories/file/file/0d908664-e300-4223-869b-fc5d2cef285f/versions/1/"
            }
        },
        "present": {
            "file.file": {
                "count": 1,
                "href": "/pulp/api/v3/content/file/files/?repository_version=/pulp/
↪api/v3/repositories/file/file/0d908664-e300-4223-869b-fc5d2cef285f/versions/1/"
            }
        },
        "removed": {}
    },
    "number": 1
}
```


Reference (pulpcore): Repository Version Creation API Usage

3.2.4 Publish and Host

This section assumes that you have a repository with content in it (a repository version). To do this, see the *Synchronize a Repository* or *Upload Content* documentation.

Create a Publication

```
#!/usr/bin/env bash
echo "Create a new publication specifying the repository_version."
# Alternatively, you can specify the repository, and Pulp will assume the latest␣
↪version.
export TASK_URL=$(http POST $BASE_ADDR/pulp/api/v3/publications/file/file/ \
```

(continues on next page)

(continued from previous page)

```
  repository_version=$REPOVERSION_HREF | jq -r '.task')

# Poll the task (here we use a function defined in docs/_scripts/base.sh)
wait_until_task_finished $BASE_ADDR$TASK_URL

echo "Setting PUBLICATION_HREF from the completed task."
export PUBLICATION_HREF=$(http $BASE_ADDR$TASK_URL| jq -r '.created_resources | first
↪')

echo "Inspecting Publication."
http $BASE_ADDR$PUBLICATION_HREF
```

Publication GET Response (after task is complete):

```
{
    "pulp_created": "2019-05-16T19:28:42.971611Z",
    "pulp_href": "/pulp/api/v3/publications/file/file/7d5440f6-202c-4e71-ace2-
↪14c534f6df9e/",
    "distributions": [],
    "publisher": null,
    "repository": "/pulp/api/v3/repositories/e242c556-bf46-4330-9c81-0be5432e55ba/
↪file/file/",
    "repository_version": "/pulp/api/v3/repositories/e242c556-bf46-4330-9c81-
↪0be5432e55ba/file/file/versions/1/"
}
```

Reference: File Publication Usage

Create a Distribution for the Publication

```
#!/usr/bin/env bash

export DIST_NAME=$(head /dev/urandom | tr -dc a-z | head -c5)
export DIST_BASE_PATH=$(head /dev/urandom | tr -dc a-z | head -c5)

# Distributions are created asynchronously.
echo "Creating distribution \
  (name=$DIST_NAME, base_path=$DIST_BASE_PATH publication=$PUBLICATION_HREF)."
export TASK_URL=$(http POST $BASE_ADDR/pulp/api/v3/distributions/file/file/ \
  name=$DIST_NAME \
  base_path=$DIST_BASE_PATH \
  publication=$PUBLICATION_HREF | jq -r '.task')

# Poll the task (here we use a function defined in docs/_scripts/base.sh)
wait_until_task_finished $BASE_ADDR$TASK_URL

echo "Setting DISTRIBUTION_HREF from the completed task."
# DISTRIBUTION_HREF is the pulp-api HREF, not the content app href
export DISTRIBUTION_HREF=$(http $BASE_ADDR$TASK_URL| jq -r '.created_resources | first
↪')

echo "Inspecting Distribution."
http $BASE_ADDR$DISTRIBUTION_HREF
```

Distribution GET Response (after task is complete):

```
{
    "pulp_created": "2019-05-16T19:28:45.135868Z",
    "pulp_href": "/pulp/api/v3/distributions/file/file/9e9e07cb-b30f-41c5-a98b-
↪583185f907e2/",
    "base_path": "foo",
    "base_url": "localhost:24816/pulp/content/foo",
    "content_guard": null,
    "name": "baz",
    "publication": "/pulp/api/v3/publications/file/file/7d5440f6-202c-4e71-ace2-
↪14c534f6df9e/"
}
```

Reference: File Distribution Usage

Download `test.iso` from Pulp

If you created your repository version using the *Synchronize a Repository* workflow:

```
#!/usr/bin/env bash

# The distribution will return a url that can be used by http clients
echo "Setting DISTRIBUTION_BASE_URL, which is used to retrieve content from the␣
↪content app."
export DISTRIBUTION_BASE_URL=$(http $BASE_ADDR$DISTRIBUTION_HREF | jq -r '.base_url')
# If Pulp was installed without CONTENT_HOST set, it's just the path.
# And httpie will default to localhost:80
if [[ "${DISTRIBUTION_BASE_URL:0:1}" = "/" ]]; then
    DISTRIBUTION_BASE_URL=$CONTENT_ADDR$DISTRIBUTION_BASE_URL
fi

# Next we download a file from the distribution
# This will default to http://
http -d $DISTRIBUTION_BASE_URL/test.iso
```

If you created your repository version using the *Upload Content* workflow:

```
#!/usr/bin/env bash

# The distribution will return a url that can be used by http clients
echo "Setting DISTRIBUTION_BASE_URL, which is used to retrieve content from the␣
↪content app."
export DISTRIBUTION_BASE_URL=$(http $BASE_ADDR$DISTRIBUTION_HREF | jq -r '.base_url')
# If Pulp was installed without CONTENT_HOST set, it's just the path.
# And httpie will default to localhost:80
if [[ "${DISTRIBUTION_BASE_URL:0:1}" = "/" ]]; then
    DISTRIBUTION_BASE_URL=$CONTENT_ADDR$DISTRIBUTION_BASE_URL
fi

echo "Downloading file from Distribution via the content app."
# This will default to http://
http -d $DISTRIBUTION_BASE_URL/$ARTIFACT_RELATIVE_PATH
```

3.3 REST API

Pulpcore Reference: pulpcore REST documentation.

3.3.1 Pulp File Endpoints

Pulp File Reference pulp-file REST documentation

3.4 Changelog

3.4.1 1.0.0 (2020-05-27)

Misc

- #6514, #6708, #6730, #6761

3.4.2 0.3.0 (2020-04-16)

Features

- Added history for filesystem exports at `/exporters/file/filesystem/<uuid>/exports/`. #6328
- Add support for import/export processing #6472

Deprecations and Removals

- The fileystem exporter endpoint has been moved from `/exporters/file/file/` to `/exporters/file/filesystem/` and the export endpoint is now at POST `/exporters/file/filesystem/<uuid>/exports/`. Additionally, the table is being dropped and recreated due to a data structure change in core so users will lose any filesystem exporter data on upgrade. #6328

Misc

- #6155, #6300, #6362, #6392

3.4.3 0.2.0 (2020-02-26)

Deprecations and Removals

- Renamed the filter for the field 'digest' to 'sha256' to correspond to field name in API and other plugins. #5965

Misc

- #5567

3.4.4 0.1.1 (2020-01-31)

Bugfixes

- Adjusts setup.py classifier to show 0.1.0 as Production/Stable. #5897

Misc

- #5867, #5872, #5967, #6016

3.4.5 0.1.0 (2019-12-12)

Improved Documentation

- Labeling Exporters as tech preview. #5563

Misc

- #5701

3.4.6 0.1.0rc2 (2019-12-03)

Features

- Add checking for path overlapping for RepositoryVersions and Publications. #5559

Misc

- #5757

3.4.7 0.1.0rc1 (2019-11-14)

Features

- Sync, Upload, and Modify now have added content with the same *relative_path* as existing content will remove the existing content. #3541
- Change *relative_path* from *CharField* to *TextField* #4544
- Added support for exporting file publications to the filesystem. #5086

Deprecations and Removals

- Sync is no longer available at the {remote_href}/sync/ repository={repo_href} endpoint. Instead, use POST {repo_href}/sync/ remote={remote_href}.

  Creating / listing / editing / deleting file repositories is now performed on /pulp/api/v3/file/file/ instead of /pulp/api/v3/repositories/. Only file content can be present in a file repository, and only a file repository can hold file content. #5625

Misc

- #3308, #5458, #5580, #5629

3.4.8 0.1.0b4 (2019-10-15)

Bugfixes

- New RepositoryVersions will remove an existing unit at the same *relative_path*. This is true for both *sync* and *upload*, and is per Repository. #4028

Improved Documentation

- Change the prefix of Pulp services from pulp-* to pulpcore-* #4554

Deprecations and Removals

- Change *_id*, *_created*, *_last_updated*, *_href* to *pulp_id*, *pulp_created*, *pulp_last_updated*, *pulp_href* #5457
- Remove "_" from *_versions_href*, *_latest_version_href* #5548
- Removing base field: *_type* . #5550

3.4.9 0.1.0b3 (2019-09-30)

Features

- Setting *code* on *ProgressBar*. #5184
- Add upload functionality to the file content endpoint. #5403

Deprecations and Removals

- Adjust FileContentSerializer to upstream change. #5428

Misc

- #5304, #5444

3.4.10 0.1.0b2 (2019-09-11)

Improved Documentation

- Fix the code snippet provided in the example for creating a file content #5094

Misc

- #4681

3.4.11 0.1.0b1 (2019-07-09)

Features

- Override the Remote's serializer to allow policy='on_demand' and policy='streamed'. #4990

Improved Documentation

- Switch to using towncrier for better release notes. #4875

3.5 Contributing

To contribute to the `pulp_file` 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_file` clone
6. Make pull request from github page for your clone against master branch

3.5.1 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 4

Indices and tables

- genindex
- modindex
- search