Pulp cookbook Support

Release 0.1.0b10.dev

Feb 19, 2022

Contents

1 Table of Contents 3

2 Indices and tables 25

The `cookbook` plugin extends pulpcore to support hosting Chef cookbooks. This plugin assumes some familiarity with the pulpcore documentation.

Currently, it allows to import packaged cookbooks into cookbook repositories. When publishing a specific version of a cookbook repository, a universe endpoint will be created to allow berkshelf or Policyfile tooling to download cookbooks and resolve cookbook dependencies.

Not supported (yet):

- Only the `universe` endpoint of the Supermarket API is implemented. For example, the `knife supermarket` commands do not work currently.
- Setting cookbook version constraints on a remote (only filtering by cookbook name is supported)

If you are just getting started, we recommend getting to know the *basic workflows*.

The REST API documentation for `pulp_cookbook` is available here.

CHAPTER 1

Table of Contents

1.1 User Setup

1.1.1 Ansible Installer (Recommended)

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

1.1.2 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-cookbook
```

From Source

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

1.1.3 Run Migrations

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

1.1.4 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
```

1.2 Workflows

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

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 admin
```

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.

To make these workflows copy/pastable, we make use of environment variables. The first variable to set is the hostname and port:

```
$ export BASE_ADDR=http://<hostname>:24817
```

This documentation makes use of the jq tool 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/binary with:

```
$ sudo dnf install jq
```

1.2.1 Scripting

The workflows use bash functions that wait for the completion of a Pulp task if necessary. These are defined in `docs/_scripts/base.sh`:

```
: "${BASE_ADDR:=http://localhost}"
: "${CONTENT_ADDR:=http://localhost}"

# Poll a Pulp task until it is finished.
wait_until_task_finished() {
    local task_url=$1
    while true
```

(continues on next page)

(continued from previous page)

```
    do
        local response=$(http --pretty format "$task_url")
        local state=$(jq -r .state <<< "${response}")
        case ${state} in
            failed|canceled)
                cat <<< "${response}" >&2
                echo "Task in final state: ${state}" >&2
                break
                ;;
            completed)
                cat <<< "${response}"
                break
                ;;
            *)
                echo "Waiting for task completion. Task:" >&2
                cat <<< "${response}" >&2
                sleep 1
                ;;
        esac
    done
}

pulp_http() {
    local response=$(http --pretty format "$@")
    local task_url=$(jq -r '.task' <<< "$response")
    if [[ "$task_url" == "null" ]]; then
        cat <<< "$response"
    else
        wait_until_task_finished "$BASE_ADDR$task_url"
    fi
}
```

To execute the examples given in the workflows (and to develop own workflows), source `base.sh` first:

```
source docs/_script/base.sh
```

Synchronize a Repository

Users can populate their repositories with content from an external sources by syncing their repository.

Bear in mind that the attached snippets utilize `pulp_http`. Refer to *Workflows* to learn more about these utilities and the required tools (`httpie` and `jq`).

Create a Remote

In addition to *uploading content*, `pulp_cookbook` allows to synchronize a repository with an external content source (that has to provide a "universe" endpoint, like the Chef Supermarket). Creating a remote object informs Pulp about an external content source.

Since the Chef Supermarket is huge, let's mirror only the `pulp`, `qpid`, and `ubuntu` cookbooks into our repository by setting the `cookbooks` parameter. First, we have to create a remote:

```
pulp_http POST $BASE_ADDR/pulp/api/v3/remotes/cookbook/cookbook/ name='foo_remote'␣
↪url='https://supermarket.chef.io/' policy=immediate cookbooks:='{"pulp": "", "qpid
↪": "", "ubuntu": ""}'
export REMOTE_HREF=$(http $BASE_ADDR/pulp/api/v3/remotes/cookbook/cookbook/?name=foo_
↪remote | jq -r '.results[0].pulp_href')
```

Response:


```
{
    "ca_cert": null,
    "client_cert": null,
    "client_key": null,
    "cookbooks": {
        "pulp": "",
        "qpid": "",
        "ubuntu": ""
    },
    "download_concurrency": 10,
    "name": "foo_remote",
    "password": null,
    "policy": "immediate",
    "proxy_url": null,
    "pulp_created": "2020-08-31T18:55:53.000019Z",
    "pulp_href": "/pulp/api/v3/remotes/cookbook/cookbook/4fb67e84-1c87-454a-9cc9-
↪d70f747b78de/",
    "pulp_last_updated": "2020-08-31T18:55:53.000041Z",
    "tls_validation": true,
    "url": "https://supermarket.chef.io/",
    "username": null
}
```


Create a Repository

If you don't already have a repository called `foo`, create one. While it is possible to add the remote as a parameters to every sync operation, it is more convenient to store the default remote in the repository:

```
pulp_http --body POST $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/ name=foo␣
↪remote=$REMOTE_HREF
export REPO_HREF=$(http $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/?
↪name=foo | jq -r '.results[0].pulp_href')
```

Response:


```
{
    "description": null,
    "latest_version_href": "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-
↪4135-81c5-3ebb7a490e20/versions/0/",
    "name": "foo",
    "pulp_created": "2020-08-31T18:55:53.983398Z",
    "pulp_href": "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-
↪3ebb7a490e20/",
    "remote": "/pulp/api/v3/remotes/cookbook/cookbook/4fb67e84-1c87-454a-9cc9-
↪d70f747b78de/",
    "versions_href": "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-
↪81c5-3ebb7a490e20/versions/"
}
```

Sync repository `foo`

Use the repository object to kick off a synchronize task. You are telling pulp to fetch content from the default remote stored with the repository and add to the latest repository version (`mirror:=false`):

```
pulp_http POST $BASE_ADDR$REPO_HREF'sync/' mirror:=false
export LATEST_VERSION_HREF=$(http $BASE_ADDR$REPO_HREF | jq -r '.latest_version_href')
```

Response (finished task status):


```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-
↪3ebb7a490e20/versions/1/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:56:01.346161Z",
    "name": "pulp_cookbook.app.tasks.synchronizing.synchronize",
    "parent_task": null,
    "progress_reports": [
        {
            "code": "downloading.metadata",
            "done": 1,
            "message": "Downloading Metadata",
            "state": "completed",
            "suffix": null,
            "total": 1
        },
        {
            "code": "downloading.artifacts",
            "done": 18,
            "message": "Downloading Artifacts",
            "state": "completed",
            "suffix": null,
            "total": null
        },
        {
            "code": "associating.content",
            "done": 18,
            "message": "Associating Content",
            "state": "completed",
            "suffix": null,
            "total": null
        },
        {
            "code": "parsing.metadata",
            "done": 18,
            "message": "Parsing Metadata",
            "state": "completed",
            "suffix": null,
            "total": null
        }
    ],
    "pulp_created": "2020-08-31T18:55:54.958181Z",
    "pulp_href": "/pulp/api/v3/tasks/afc496dc-0862-4d9a-8bcc-24225cafbaf7/",
    "reserved_resources_record": [
```


(continues on next page)

(continued from previous page)

```
        "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-
↪3ebb7a490e20/",
        "/pulp/api/v3/remotes/cookbook/cookbook/4fb67e84-1c87-454a-9cc9-d70f747b78de/"
    ],
    "started_at": "2020-08-31T18:55:55.156590Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/7a1130eb-8483-418f-a370-cecf0e63f60b/"
}
```

When the synchronize task completes successfully, there are two possible outcomes:

1. If the sync changed the content, it creates a new version, which is specified in `created_resources`. Moreover, the created repository version will become the `latest_version` of the repository.
2. If the sync did not change the content (i.e. the remote contains no changes compared to the current repository version), no new version is created and `created_resources` is empty.

You can have a look at the latest repository version:

```
http "$BASE_ADDR$LATEST_VERSION_HREF"
```

Response:

```
{"pulp_href":"/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-
↪3ebb7a490e20/versions/1/","pulp_created":"2020-08-31T18:55:55.190535Z","number":1,
↪"base_version":null,"content_summary":{"added":{"cookbook.cookbook":{"count":18,
↪"href":"/pulp/api/v3/content/cookbook/cookbooks/?repository_version_added=/pulp/api/
↪v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-3ebb7a490e20/versions/1/"}
↪},"removed":{},"present":{"cookbook.cookbook":{"count":18,"href":"/pulp/api/v3/
↪content/cookbook/cookbooks/?repository_version=/pulp/api/v3/repositories/cookbook/
↪cookbook/84be8c38-0889-4135-81c5-3ebb7a490e20/versions/1/"}}}}
```

Upload and Manage Content

The section shows how to upload content to Pulp.

Bear in mind that the attached snippets utilize `pulp_http`. Refer to *Workflows* to learn more about these utilities and the required tools (`httpie` and `jq`).

Create a repository

If you don't already have a repository called `foo`, create one:

```
pulp_http --body POST $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/ name=foo
export REPO_HREF=$(http $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/?
↪name=foo | jq -r '.results[0].pulp_href')
```

Response:

```
{
    "description": null,
    "latest_version_href": "/pulp/api/v3/repositories/cookbook/cookbook/bfd18a84-3b55-
↪44c0-9566-7878fa73f67e/versions/0/",
    "name": "foo",
```

(continues on next page)

(continued from previous page)

```
    "pulp_created": "2020-08-31T18:55:45.422914Z",
    "pulp_href": "/pulp/api/v3/repositories/cookbook/cookbook/bfd18a84-3b55-44c0-9566-
↪7878fa73f67e/",
    "remote": null,
    "versions_href": "/pulp/api/v3/repositories/cookbook/cookbook/bfd18a84-3b55-44c0-
↪9566-7878fa73f67e/versions/"
}
```

Upload local cookbooks to Pulp

As a simple example, let's download two cookbooks from the Chef Supermarket and upload them into our `foo` repository.

Download 'ubuntu' and 'apt' cookbooks into the current directory (the 'ubuntu' cookbooks depends on the 'apt' cookbook):

```
curl -Lo ubuntu-2.0.1.tgz https://supermarket.chef.io:443/api/v1/cookbooks/ubuntu/
↪versions/2.0.1/download``
curl -Lo apt-7.0.0.tgz https://supermarket.chef.io:443/api/v1/cookbooks/apt/versions/
↪7.0.0/download``
```

Now, we create a cookbook content unit (which represent a cookbook in Pulp) for each file, respectively:

```
pulp_http --form POST $BASE_ADDR/pulp/api/v3/content/cookbook/cookbooks/ name="ubuntu
↪" file@ubuntu-2.0.1.tgz
export UBUNTU_CONTENT_HREF=$(http $BASE_ADDR/pulp/api/v3/content/cookbook/cookbooks/?
↪name=ubuntu | jq -r '.results[0].pulp_href')
```

Response (finished task status):

```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/content/cookbook/cookbooks/ba13f4e9-93fd-405e-aad2-dd37bf7c8f80/
↪"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:55:46.846807Z",
    "name": "pulpcore.app.tasks.base.general_create",
    "parent_task": null,
    "progress_reports": [],
    "pulp_created": "2020-08-31T18:55:46.507982Z",
    "pulp_href": "/pulp/api/v3/tasks/825345fc-e823-405a-8d0d-d970761fc99b/",
    "reserved_resources_record": [
        "/pulp/api/v3/artifacts/a91321d9-bd6f-408b-a094-f841925bf75b/"
    ],
    "started_at": "2020-08-31T18:55:46.747676Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```

And for the apt cookbook:

```
pulp_http --form POST $BASE_ADDR/pulp/api/v3/content/cookbook/cookbooks/ name="apt"␣
↪file@apt-7.0.0.tgz
export APT_CONTENT_HREF=$(http $BASE_ADDR/pulp/api/v3/content/cookbook/cookbooks/?
↪name=apt | jq -r '.results[0].pulp_href')
```

Response (finished task status):


```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/content/cookbook/cookbooks/5a7814d6-b3da-4363-a510-04ac097d4df2/
↪"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:55:48.556499Z",
    "name": "pulpcore.app.tasks.base.general_create",
    "parent_task": null,
    "progress_reports": [],
    "pulp_created": "2020-08-31T18:55:48.251165Z",
    "pulp_href": "/pulp/api/v3/tasks/935b9ae7-910d-4a54-8269-e71abd647f67/",
    "reserved_resources_record": [
        "/pulp/api/v3/artifacts/6690ae77-b88f-4586-a99a-03404535974e/"
    ],
    "started_at": "2020-08-31T18:55:48.456306Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```


Add content to repository `foo`

Once we have content unit(s), they can be added and removed and from to repositories, creating a new repository version (which will be the latest version):

```
pulp_http POST $BASE_ADDR$REPO_HREF'modify/' add_content_units:="[\"$UBUNTU_CONTENT_
↪HREF\",\"$APT_CONTENT_HREF\"]"
export LATEST_VERSION_HREF=$(http $BASE_ADDR$REPO_HREF | jq -r '.latest_version_href')
```

Response (finished task status):


```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/repositories/cookbook/cookbook/bfd18a84-3b55-44c0-9566-
↪7878fa73f67e/versions/1/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:55:50.281814Z",
    "name": "pulpcore.app.tasks.repository.add_and_remove",
    "parent_task": null,
    "progress_reports": [],
    "pulp_created": "2020-08-31T18:55:49.950397Z",
    "pulp_href": "/pulp/api/v3/tasks/1abcec8b-a38b-4133-a033-94ca73971aa9/",
    "reserved_resources_record": [
        "/pulp/api/v3/repositories/cookbook/cookbook/bfd18a84-3b55-44c0-9566-
↪7878fa73f67e/"
```


(continues on next page)

(continued from previous page)

```
    ],
    "started_at": "2020-08-31T18:55:50.201119Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```

One Shot upload

TODO

Publish and Host

This section assumes that you have a repository called `foo` with content in it. To do this, see the *Synchronize a Repository* or *Upload and Manage Content* documentation.

Create a Publication

Create a publication from the latest repository version. This prepares the meta-data needed to distribute the content of the repository on a `/universe` endpoint. However, creating a publication distributes no content to the outside. This has to be done in a separate step.

```
export REPO_HREF=$(http $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/?
↪name=foo | jq -r '.results[0].pulp_href')
export LATEST_VERSION_HREF=$(http $BASE_ADDR$REPO_HREF | jq -r '.latest_version_href')
task_result=$(pulp_http POST $BASE_ADDR/pulp/api/v3/publications/cookbook/cookbook/␣
↪repository_version=$LATEST_VERSION_HREF)
echo "$task_result"
export PUBLICATION_HREF=$(echo "$task_result" | jq -r '.created_resources[0]')
```

Response:

```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/publications/cookbook/cookbook/a477ab9e-79e1-4fe8-a56c-
↪dbd31e629afe/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:56:04.875990Z",
    "name": "pulp_cookbook.app.tasks.publishing.publish",
    "parent_task": null,
    "progress_reports": [
        {
            "code": "publishing.content",
            "done": 18,
            "message": "Publishing Content",
            "state": "completed",
            "suffix": null,
            "total": null
        }
    ],
```

(continues on next page)

(continued from previous page)

```
    "pulp_created": "2020-08-31T18:56:04.575918Z",
    "pulp_href": "/pulp/api/v3/tasks/9bd500b9-3a5e-4a80-a8f1-77b086f48fa6/",
    "reserved_resources_record": [
        "/pulp/api/v3/repositories/cookbook/cookbook/84be8c38-0889-4135-81c5-
↪3ebb7a490e20/"
    ],
    "started_at": "2020-08-31T18:56:04.775986Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/7a1130eb-8483-418f-a370-cecf0e63f60b/"
}
```

Create a Distribution `foo` for the Publication

Actually distribute the publication created in the last step at a specific path. The base path is `foo` in this case. The publication will be distributed at `/pulp_cookbook/content/foo/universe` on the content endpoint (which is on port 24816 in our example).

```
pulp_http POST $BASE_ADDR/pulp/api/v3/distributions/cookbook/cookbook/ name='foo'␣
↪base_path='foo' publication=$PUBLICATION_HREF
export DISTRIBUTION_HREF=$(http $BASE_ADDR/pulp/api/v3/distributions/cookbook/
↪cookbook/?name=foo | jq -r '.results[0].pulp_href')
```

Response:

```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/distributions/cookbook/cookbook/4bbbb105-f466-4ba5-ba6a-
↪fbe057628534/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:56:06.166432Z",
    "name": "pulpcore.app.tasks.base.general_create",
    "parent_task": null,
    "progress_reports": [],
    "pulp_created": "2020-08-31T18:56:05.823671Z",
    "pulp_href": "/pulp/api/v3/tasks/3b12f649-46f0-471c-be40-7596fd2c9b04/",
    "reserved_resources_record": [
        "/api/v3/distributions/"
    ],
    "started_at": "2020-08-31T18:56:06.032185Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```

You can have a look at the published "universe" metadata now:

```
pulp_http $CONTENT_ADDR/pulp_cookbook/content/foo/universe
```

Response:

```
{
    "pulp": {
```

(continues on next page)

(continued from previous page)


```
        "0.1.1": {
            "dependencies": {
                "qpid": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/pulp/0_1_1/pulp-0.1.1.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/pulp/0_1_1/pulp-0.1.1.tar.gz",
            "location_type": "uri"
        }
    },
    "qpid": {
        "0.1.3": {
            "dependencies": {
                "yum": ">= 0.0.0",
                "yum-epel": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/qpid/0_1_3/qpid-0.1.3.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/qpid/0_1_3/qpid-0.1.3.tar.gz",
            "location_type": "uri"
        }
    },
    "ubuntu": {
        "0.7.0": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/0_7_0/ubuntu-0.7.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/0_7_0/ubuntu-0.7.0.tar.gz",
            "location_type": "uri"
        },
        "0.99.0": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/0_99_0/ubuntu-0.99.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/0_99_0/ubuntu-0.99.0.tar.gz",
            "location_type": "uri"
        },
        "1.0.0": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_0_0/ubuntu-1.0.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_0_0/ubuntu-1.0.0.tar.gz",
            "location_type": "uri"
        },
        "1.1.0": {
            "dependencies": {
```


(continues on next page)

(continued from previous page)


```
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_0/ubuntu-1.1.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_0/ubuntu-1.1.0.tar.gz",
            "location_type": "uri"
        },
        "1.1.2": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_2/ubuntu-1.1.2.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_2/ubuntu-1.1.2.tar.gz",
            "location_type": "uri"
        },
        "1.1.4": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_4/ubuntu-1.1.4.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_4/ubuntu-1.1.4.tar.gz",
            "location_type": "uri"
        },
        "1.1.6": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_6/ubuntu-1.1.6.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_6/ubuntu-1.1.6.tar.gz",
            "location_type": "uri"
        },
        "1.1.8": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_8/ubuntu-1.1.8.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_1_8/ubuntu-1.1.8.tar.gz",
            "location_type": "uri"
        },
        "1.2.0": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_2_0/ubuntu-1.2.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_2_0/ubuntu-1.2.0.tar.gz",
            "location_type": "uri"
```


(continues on next page)

(continued from previous page)


```
        },
        "1.2.1": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_2_1/ubuntu-1.2.1.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/1_2_1/ubuntu-1.2.1.tar.gz",
            "location_type": "uri"
        },
        "2.0.0": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/2_0_0/ubuntu-2.0.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/2_0_0/ubuntu-2.0.0.tar.gz",
            "location_type": "uri"
        },
        "2.0.1": {
            "dependencies": {
                "apt": ">= 0.0.0"
            },
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/2_0_1/ubuntu-2.0.1.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/2_0_1/ubuntu-2.0.1.tar.gz",
            "location_type": "uri"
        },
        "3.0.0": {
            "dependencies": {},
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_0/ubuntu-3.0.0.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_0/ubuntu-3.0.0.tar.gz",
            "location_type": "uri"
        },
        "3.0.1": {
            "dependencies": {},
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_1/ubuntu-3.0.1.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_1/ubuntu-3.0.1.tar.gz",
            "location_type": "uri"
        },
        "3.0.2": {
            "dependencies": {},
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_2/ubuntu-3.0.2.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_2/ubuntu-3.0.2.tar.gz",
            "location_type": "uri"
        },
        "3.0.3": {
            "dependencies": {},
```


(continues on next page)

(continued from previous page)

```
            "download_url": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_3/ubuntu-3.0.3.tar.gz",
            "location_path": "http://pulp3-source-fedora32.gandalf.example.com/pulp_
↪cookbook/content/foo/cookbook_files/ubuntu/3_0_3/ubuntu-3.0.3.tar.gz",
            "location_type": "uri"
        }
    }
}
```

Use Berkshelf with the published repo

Create a `Berksfile` with the following content:

```
source 'http://localhost/pulp_cookbook/content/foo/'

cookbook 'ubuntu'
```

And run:

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching cookbook index from http://localhost:24816/pulp_cookbook/content/foo/...
Installing apt (7.0.0) from http://localhost:24816/pulp_cookbook/content/foo/ ([uri]␣
↪http://localhost:24816/pulp_cookbook/content/foo/cookbook_files/apt/7_0_0/apt-7.0.0.
↪tar.gz)
Installing ubuntu (2.0.1) from http://localhost:24816/pulp_cookbook/content/foo/␣
↪([uri] http://localhost:24816/pulp_cookbook/content/foo/cookbook_files/ubuntu/2_0_1/
↪ubuntu-2.0.1.tar.gz)
```

Snapshot of Chef Supermarket

Using the 'on_demand' policy on a remote allows to create snapshots of a large repository like the Chef Supermarket effectively. In "on_demand" mode, only the meta-data will be synchronized. Actual cookbooks are not downloaded at sync time, but only when requested from a distribution.

The coobook versions distributed are the ones that were presents during the sync, i.e. this will create a frozen snapshot of the Chef Supermarket. Thanks to the repository versioning feature of Pulp 3, you can create a new repository version (i.e. a new snapshot) by syncing the Chef Supermarket at a later time. Note that this does neither influence existing repository versions nor existing distributions. This new repository version will become visible only after publishing and updating the distribution accordingly.

Create a Remote `supermarket`

Using *policy=on_demand* for the remote enables "lazy download": A sync using this remote will download the meta-data from the Chef Supermarket (i.e. information about all cookbooks and their respective versions), but no actual cookbook tar files.

Cookbooks will be downloaded only when requested from a distribution. After the first successful download, the cookbooks are stored locally for faster retrieval.

```
pulp_http POST $BASE_ADDR/pulp/api/v3/remotes/cookbook/cookbook/ name='supermarket'␣
↪url='https://supermarket.chef.io/' policy=on_demand
export REMOTE_HREF=$(http $BASE_ADDR/pulp/api/v3/remotes/cookbook/cookbook/?
↪name=supermarket | jq -r '.results[0].pulp_href')
```

Response:


```
{
    "ca_cert": null,
    "client_cert": null,
    "client_key": null,
    "cookbooks": null,
    "download_concurrency": 10,
    "name": "supermarket",
    "password": null,
    "policy": "on_demand",
    "proxy_url": null,
    "pulp_created": "2020-08-31T18:56:18.826267Z",
    "pulp_href": "/pulp/api/v3/remotes/cookbook/cookbook/5b43bd86-7c52-425c-a369-
↪5693cdea68c2/",
    "pulp_last_updated": "2020-08-31T18:56:18.826301Z",
    "tls_validation": true,
    "url": "https://supermarket.chef.io/",
    "username": null
}
```


Create a repository `supermarket`

If you don't already have a repository called `supermarket`, create one using the remote just created as its default remote:

```
pulp_http --body POST $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/␣
↪name=supermarket description="Snapshots of Supermarket (using lazy download)"␣
↪remote=$REMOTE_HREF
export REPO_HREF=$(http $BASE_ADDR/pulp/api/v3/repositories/cookbook/cookbook/?
↪name=supermarket | jq -r '.results[0].pulp_href')
```

Response:


```
{
    "description": "Snapshots of Supermarket (using lazy download)",
    "latest_version_href": "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-
↪4122-9b33-60b883b793c7/versions/0/",
    "name": "supermarket",
    "pulp_created": "2020-08-31T18:56:19.846680Z",
    "pulp_href": "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-4122-9b33-
↪60b883b793c7/",
    "remote": "/pulp/api/v3/remotes/cookbook/cookbook/5b43bd86-7c52-425c-a369-
↪5693cdea68c2/",
    "versions_href": "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-4122-
↪9b33-60b883b793c7/versions/"
}
```

Sync repository `supermarket`

Use the repository object to kick off a synchronize task. You are telling pulp to fetch content from the default remote stored in the repository and mirror the current content exactly (`mirror:=true`):

```
pulp_http POST $BASE_ADDR$REPO_HREF'sync/' mirror:=true
export LATEST_VERSION_HREF=$(http $BASE_ADDR$REPO_HREF | jq -r '.latest_version_href')
```

Response (finished task status):


```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-4122-9b33-
↪60b883b793c7/versions/1/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:59:03.915029Z",
    "name": "pulp_cookbook.app.tasks.synchronizing.synchronize",
    "parent_task": null,
    "progress_reports": [
        {
            "code": "downloading.metadata",
            "done": 1,
            "message": "Downloading Metadata",
            "state": "completed",
            "suffix": null,
            "total": 1
        },
        {
            "code": "parsing.metadata",
            "done": 25910,
            "message": "Parsing Metadata",
            "state": "completed",
            "suffix": null,
            "total": null
        },
        {
            "code": "associating.content",
            "done": 25910,
            "message": "Associating Content",
            "state": "completed",
            "suffix": null,
            "total": null
        },
        {
            "code": "unassociating.content",
            "done": 0,
            "message": "Un-Associating Content",
            "state": "completed",
            "suffix": null,
            "total": null
        }
    ],
    "pulp_created": "2020-08-31T18:56:20.899326Z",
    "pulp_href": "/pulp/api/v3/tasks/165479ae-a05f-4d2b-bff8-a2500877f9d7/",
    "reserved_resources_record": [
```


(continues on next page)

(continued from previous page)

```
        "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-4122-9b33-
↪60b883b793c7/",
        "/pulp/api/v3/remotes/cookbook/cookbook/5b43bd86-7c52-425c-a369-5693cdea68c2/"
    ],
    "started_at": "2020-08-31T18:56:21.075800Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```

Create a Publication

The sync created a new repository version which is reflected in the "latest_version_href" field of the repository.

You can kick off a publish task by specifying the repository version to publish. Alternatively, you can specify repository, which will publish the latest version.

The result of a publish is a publication, which contains all the information needed for an external package manager like `berkshelf` to use. Publications are not consumable until they are hosted by a distribution:

```
task_result=$(pulp_http POST $BASE_ADDR/pulp/api/v3/publications/cookbook/cookbook/␣
↪repository_version=$LATEST_VERSION_HREF)
echo "$task_result"
export PUBLICATION_HREF=$(echo "$task_result" | jq -r '.created_resources[0]')
```

Response (finished task status):

```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/publications/cookbook/cookbook/6f6ac342-6fdf-4ad1-8b93-
↪449786093d36/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:59:40.138601Z",
    "name": "pulp_cookbook.app.tasks.publishing.publish",
    "parent_task": null,
    "progress_reports": [
        {
            "code": "publishing.content",
            "done": 25910,
            "message": "Publishing Content",
            "state": "completed",
            "suffix": null,
            "total": null
        }
    ],
    "pulp_created": "2020-08-31T18:59:05.653566Z",
    "pulp_href": "/pulp/api/v3/tasks/992dc136-d38b-4a25-af05-98ca91b4f807/",
    "reserved_resources_record": [
        "/pulp/api/v3/repositories/cookbook/cookbook/cefae475-f693-4122-9b33-
↪60b883b793c7/"
    ],
    "started_at": "2020-08-31T18:59:05.871864Z",
    "state": "completed",
```

(continues on next page)

(continued from previous page)

```
    "task_group": null,
    "worker": "/pulp/api/v3/workers/714038d7-7077-43c5-b980-c9be814dcaa0/"
}
```

Create a Distribution at 'supermarket' for the Publication

To host a publication, you create a distribution which will serve the associated publication at `/pulp_cookbook/content/<distribution.base_path>`. In the example, `base_path` is set to `supermarket`.

```
pulp_http POST $BASE_ADDR/pulp/api/v3/distributions/cookbook/cookbook/ name=
↪'supermarket' base_path='supermarket' publication=$PUBLICATION_HREF
export DISTRIBUTION_HREF=$(http $BASE_ADDR/pulp/api/v3/distributions/cookbook/
↪cookbook/?name=supermarket | jq -r '.results[0].pulp_href')
```

Response (finished task status):

```
{
    "child_tasks": [],
    "created_resources": [
        "/pulp/api/v3/distributions/cookbook/cookbook/d3dad6b8-817b-42c2-b546-
↪addf72e92bf5/"
    ],
    "error": null,
    "finished_at": "2020-08-31T18:59:42.672067Z",
    "name": "pulpcore.app.tasks.base.general_create",
    "parent_task": null,
    "progress_reports": [],
    "pulp_created": "2020-08-31T18:59:42.253922Z",
    "pulp_href": "/pulp/api/v3/tasks/98dce068-16e5-4006-a1bb-836392335202/",
    "reserved_resources_record": [
        "/api/v3/distributions/"
    ],
    "started_at": "2020-08-31T18:59:42.522598Z",
    "state": "completed",
    "task_group": null,
    "worker": "/pulp/api/v3/workers/7a1130eb-8483-418f-a370-cecf0e63f60b/"
}
```

You can have a look at the published "universe" metadata now:

```
$ http localhost:24816/pulp_cookbook/content/supermarket/universe
```

In your `Berksfile`, you can use the following `source` to access the Supermarket snapshot:

```
source 'http://localhost/pulp_cookbook/content/supermarket/'
```

1.3 REST API

Pulpcore Reference: pulpcore REST documentation.

1.3.1 Pulp Cookbook Endpoints

Pulp Cookbook Reference pulp-cookbook REST documentation

1.4 pulp-cookbook Changelog

1.4.1 0.1.0b9 (2022-02-18)

Improved Documentation

- Follow documentation style of other Pulp plugins, documentation is available at https://pulp-cookbook.readthedocs.io/ #87

Misc

- #94, #95

1.4.2 0.1.0b8 (2020-08-29)

Features

- Added ability for users to add Remote to Repository and automatically use it when syncing. #84

1.4.3 0.1.0b7 (2020-04-03)

No significant changes.

1.4.4 0.1.0b6 (2020-02-09)

Features

- Speed up "create publication". PublishedArtifacts were created one by one. Use prefetching and bulk creation to speed up publishing. #75

1.4.5 0.1.0b5 (2019-12-13)

Features

- Adapt to pulpcore 3.0.0 release. #72

1.4.6 0.1.0b4 (2019-11-22)

Features

- Make repositories "typed". Repositories now live at a detail endpoint, i.e. `/pulp/api/v3/repositories/cookbook/cookbook/`. Sync is performed by POSTing to `{repo_href}/sync/ remote={remote_href}` instead of POSTing to the `{remote_href}/sync/ repository={repo_href}` endpoint. Creating a new repository version (adding & removing content) is performed by POSTing to `{repo_href}/modify/` #65
- Add validation to repository versions: A repository version must not have entries with duplicate repo_keys. Remove the current check done at publication time. #68

Deprecations and Removals

- Sync is no longer available at the `{remote_href}/sync/ repository={repo_href}` endpoint. #65

Misc

- #66

1.4.7 0.1.0b3 (2019-10-15)

Deprecations and Removals

- Change *_id*, *_created*, *_last_updated*, *_href* to *pulp_id*, *pulp_created*, *pulp_last_updated*, *pulp_href*. Remove *_type* field from content serializer. #56
- Remove "_" from *_versions_href*, *_latest_version_href* #58

1.4.8 0.1.0b2 (2019-10-04)

Features

- Use the new SingleArtifactContentUploadSerializer to implement the content upload function (single shot upload). #18

Deprecations and Removals

- Change *_artifact* field to *artifact* for cookbook content. #46

Misc

- #48

1.4.9 0.1.0b1 (2019-09-21)

Features

- Migrations are checked in now. #39

Bugfixes

- Fix breaking changes introduced by DRF 3.10 #36

1.4.10 Version 0.0.4b3

Released on May 20, 2019

- Removed the publisher, publications can be created directly
- Fix: distributions now return the correct `base_url`
- Adapt to pulpcore-plugin 0.1rc2

1.4.11 Version 0.0.4b2

Released on Mar 31, 2019

- Adapt to pulpcore-plugin 0.1rc1

1.4.12 Version 0.0.4b1

Released on Feb 5, 2019

- Support 'lazy' remote policies ('on_demand', 'streaming')
- live universe API migrated to new content app
- Implements "repo isolation". Content is shared between repos only if a cryptographic digest is known and is the same.
- Publish: Use repo key to check whether a repo version can be published without conflict.
- Adapt to pulpcore-plugin 0.1.0b18

1.4.13 Version 0.0.3a2

Released on Dec 21, 2018

- Adapt to pulpcore-plugin 0.1.0b16

1.4.14 Version 0.0.3a1

Released on Dec 19, 2018

- Adapt to pulpcore-plugin 0.1.0b15

1.4.15 Version 0.0.2a2

Released on Sep 14, 2018

- Initial version with sync and publish support (suitable for berkshelf).

1.5 Contributing

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

1. Clone the GitHub repo
2. Make a change
3. Make sure all tests passed
4. To add an entry to the change log add a file into CHANGES folder (see *Changelog update* below)
5. Commit changes to own `pulp_cookbook` clone
6. Make pull request from github page for your clone against master branch

1.5.1 Changelog update

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

To add an entry to the change log, you first need a pulp_cookbook issue 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 you may create a file for each of them with the exact same contents and Towncrier will deduplicate them.

The contents of the files are reStructuredText formatted text that will be used as the content of the change log 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 2

Indices and tables

- genindex
- modindex
- search