rpm-versiontracker Documentation

Release 0.1

David Moreau Simard

Sep 27, 2017

Contents

rpm-versiontracker is a small, generic, Flask web application that leverages the DNF API to display the list of packages available from different repositories and compare them.

It provides a REST API to query it's settings and return package lists so you can build and integrate around it.

`DNF` is short for `Dandified Yum`, the next generation of package management for Red Hat based distributions.

CHAPTER 1

Table of Contents

Installing

Dependencies

CentOS 7/RHEL7:

```
yum install dnf python34 python3-pip
pip3 install -r requirements.txt
```

Fedora 22+:

```
dnf install python3 python3-pip
pip3 install -r requirements.txt
```

rpm-versiontracker

`rpm-versiontracker` is not yet packaged. Clone the repository wherever you will be running it from:

```
git clone https://github.com/dmsimard/rpm-versiontracker.git
```

Read on the configuration documentation to see how to configure `rpm-versiontracker`.

Configuring

`rpm-versiontracker` ships with a default configuration that shows different Openstack repositories.

You can override the default configuration by editing the local_settings.py file.

Settings

- TMPDIR is a path that defines where DNF will store it's repository cache.
- REPOSITORIES defines which repositories can be queried. Configured repositories will automatically show up in the top menu.
- TAGS is a string matcher. It will automatically create a link in the Compare tab of the top menu for comparing repositories that match the tag string.
- PACKAGE_PROPERTIES defines which properties are pulled from DNF and made available through the API. The interface does not display all package properties in the detailed and comparison tables.
- SHOW_SOURCE_RPM is a toggle to display (or not) the source RPM packages alongside the various architecture packages.

Read on the running documentation to see how to get rpm-versiontracker to run.

Running

Quickstart: Standalone

Flask provides a built-in webserver and it is sufficient for self hosting rpm-versiontracker and personal usage.

To start the standalone webserver:

```
$ ./run.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

You will be able to access the application on http://127.0.0.1:5000/.

Running behind an application and web server

If you are expecting any kind of real traffic, you probably want to run rpm-versiontracker from an application server and put a web server or reverse-proxy in front of it.

There's many ways to get a Flask application running. Flask has great documentation on the different deployment options.

Perhaps your favorite setup is with Apache and mod_wsgi or you're used to gunicorn and nginx in front instead. Flask is very flexible in this regard.

What you need to know is that the application to start is run:app where run references run.py and app the actual Flask application.

The REST API is currently unrestricted. It is highly recommended to secure access to it with the help of a web server or other means as it could fairly easily be abused or used in amplification attacks.

REST API

rpm-versiontracker provides a REST API in order to build and integrate around it.

The REST API is currently unrestricted. It is highly recommended to secure access to it with the help of a web server or other means as it could fairly easily be abused or used in amplification attacks.

Settings

Repositories

GET /settings/repositories/ (**str:** *repository*) **/**

str: *param* Retrieve all repositories and their properties or, a repository and it's properties or, a specific property from a repository

Retrieve all repositories and their properties

Example request:

```
GET /settings/repositories HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "liberty-master": {
    "url": "http:\/\/trunk.rdoproject.org\/centos7-liberty\/current\/
↪delorean.repo",
    "name": "liberty-master",
    "friendly_name": "Liberty (master)"
  },
  "kilo-master": {
    "url": "http:\/\/trunk.rdoproject.org\/centos7-kilo\/current\/delorean-
↪kilo.repo",
    "name": "kilo-master",
    "friendly_name": "Kilo (master)"
  }
}
```

Retrieve the properties of a single repository

Example request:

```
GET /settings/repositories/liberty-master HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "url": "http:\/\/trunk.rdoproject.org\/centos7-liberty\/current\/delorean.
↪repo",
  "name": "liberty-master",
  "friendly_name": "Liberty (master)"
}
```

Retrieve a specific property from a single repository

Example request:

```
GET /settings/repositories/liberty-master/url HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

"http://trunk.rdoproject.org/centos7-liberty/current/delorean.repo"
```

Tags

GET /settings/tags/(str: *tag*)/

str: *param* Retrieve all tags and their properties or, a tag and it's properties or, a specific property from a tag

Retrieve all tags and their properties

Example request:

```
GET /settings/tags HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "kilo": {
    "name": "kilo",
    "friendly_name": "Kilo repositories"
  },
  "liberty": {
    "name": "liberty",
    "friendly_name": "Liberty repositories"
  }
}
```

Retrieve the properties of a single tag

Example request:

```
GET /settings/tags/liberty HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "name": "liberty",
  "friendly_name": "Liberty repositories"
}
```

Retrieve a specific property from a single tag

Example request:

```
GET /settings/tags/liberty/friendly_name HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

"Liberty repositories"
```

Package Properties

GET /settings/packageproperties

Returns the list of properties that is pulled from DNF and made available through the API when retrieving packages.

Retrieve the list of properties pulled from DNF

Example request:

```
GET /settings/packageproperties HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

[
  "arch",
  "buildtime",
  "downloadsize",
  "epoch",
  "files",
  "installtime",
  "installsize",
  "name",
  "release",
  "sourcerpm",
  "version"
]
```

Showing source packages

GET /settings/showsourcerpm
Returns `true` or `false` to show or hide source packages, respectively. Note: This setting only impacts the web interface for the time being.

Retrieve the setting to know if source RPMs are hidden

Example request:

```
GET /settings/showsourcerpm HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

false
```

Packages

GET /packages/(**str:** *repository*)**/**
str: *package***/str:** *property* Retrieve all packages and their properties from a specified repository or, a package and it's properties or, a specific property from a package

Retrieve all packages and their properties

Example request:

```
GET /packages/liberty-master HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "python-saharaclient": {
    "arch": "src",
    "sourcerpm": null,
    "release": "dev22.el7.centos",
    "version": "0.11.1",
    "name": "python-saharaclient",
    "buildtime": 1443451578
  },
  "python-glanceclient": {
    "arch": "src",
    "sourcerpm": null,
    "release": "dev10.el7.centos",
    "version": "1.1.1",
    "name": "python-glanceclient",
    "buildtime": 1443441984
  },
  "python-keystone": {
    "arch": "noarch",
    "sourcerpm": "openstack-keystone-9.0.0-dev27.el7.centos.src.rpm",
    "release": "dev27.el7.centos",
    "version": "9.0.0",
    "name": "python-keystone",
    "buildtime": 1443474407
  },
  [...]
}
```

Retrieve the properties of a single package

Example request:

```
GET /packages/liberty-master/python-keystone HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "arch": "noarch",
  "sourcerpm": "openstack-keystone-9.0.0-dev27.el7.centos.src.rpm",
  "release": "dev27.el7.centos",
  "version": "9.0.0",
  "name": "python-keystone",
  "buildtime": 1443474407
}
```

Retrieve a specific property from a single package

Example request:

```
GET /packages/liberty-master/python-keystone/sourcerpm HTTP/1.1
Host: example.org
Accept: application/json
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

"openstack-keystone-9.0.0-dev27.el7.centos.src.rpm"
```

HTTP Routing Table