RPM Support - Story #1877

Clean up the RPM models

04/28/2016 08:58 PM - jcline@redhat.com

Status:	CLOSED - WONTFIX	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:			
Platform Release:		**Tags:**	Pulp 2
Groomed:	No	**Sprint:**	
Sprint Candidate:	No	**Quarter:**	

Description

This story was inspired by https://pulp.plan.io/issues/1618

Our RPM models are confusing (to me). This story might be worth splitting up into two stories, but for now I've included them both here.

Excessive Fields

Our (S)RPM model is confusing. It contains the following fields:

- name
- epoch
- version
- release
- arch
- build_time
- buildhost
- vendor
- size
- base_url
- filename
- relative_url_path
- relativepath
- repodata
- group
- provides
- files
- description
- header_range
- sourcerpm
- license
- changelog
- url
- summary
- time
- requires

Now, this is not quite everything required to construct the metadata entries in primary.xml, filelists.xml, and other.xml, but it's almost everything. I really don't see people searching for RPMs by ``header_range`` or by ``changelog`` so the only purpose I can see for these fields is publishing. However, when we publish a repository we construct these metadata files using snippets of XML stored as strings in the database (the ``repodata`` field) and don't make use of the other database fields at all.

Therefore, we need to pick a direction (either we use the XML snippets and toss the database fields or we use the fields on the model and toss the snippets) while considering the following:

- We need all the information necessary to publish an RPM repository in the database *somewhere*. We cannot simply pull it out of the RPM itself at publish time because during lazy syncs, we don't have the packages available.

- We need to be able to change the repository metadata based on the configuration of the distributor that's publishing the repository. For now the only known example of this is the checksum and its type. If we use the snippets, we need to modify them in some way (or keep many copies) at publish time.

- There is a library, createrepo_c[0], which is designed to create these metadata files. We already use it to generate sqlite databases, if requested. It's written in C and has Python bindings so it's very performant[1]. It offers features like re-using metadata from previous repositories to update the metadata quickly (something we currently implement ourselves). It appears to be maintained by Red Hat so it'd be nice to de-duplicate our effort.

This may or may not get decided as part of issue #1618.

Unit Key

The unit key for RPM is (name, epoch, version, release, arch, checksum, checksum_type). An example would be:

```
name: "389-db-base"
epoch: "0"
version: "1.3.1.6"
release: "25.el7"
arch: "x86_64"
checksum: "sha1"
checksum_type: "46fb0312eca3ab9b94799ccc296dd16c4e3f9ef2"
```

From what I've gathered, the reason for including the checksum and its type was to differentiate the packages in RHEL from its various clones (CentOS packages have the same name, epoch, version, release, and arch as the RHEL packages). However, it would probably be more appropriate to include the ``vendor`` field, the ``packager`` field, or potentially the GPG key fingerprint that signed the package (a little iffy since I don't think it's in the yum metadata and is only available inside the RPM itself, which is a no-go with lazy syncing - we'd need to arrive at a solution with projects creating this content).

The disadvantage of using the checksum and checksum type is that it allows a vendor (like Red Hat) to re-use the NEVRA for a new build, which I believe is a bad thing. Generally, tools don't allow this (see Koji, for example).

[0] https://github.com/rpm-software-management/createrepo_c

[1] Working with upstream will probably be necessary if we want additional APIs to optimize our performance and/or minimize the ways we generate the metadata. One option is to fork out to createrepo_c and let it handle everything, but it will generate all the checksums (which means reading the entirety of every RPM). Another option is to use the Python bindings to create the repository, but this doesn't perform as well because the native option creates worker threads to perform I/O and calculate metadata in parallel. However the second option is one we need to do in cases where we don't have the RPM locally.

History

#1 - 04/12/2019 10:17 PM - bmbouter

- Status changed from NEW to CLOSED - WONTFIX

Pulp 2 is approaching maintenance mode, and this Pulp 2 ticket is not being actively worked on. As such, it is being closed as WONTFIX. Pulp 2 is still accepting contributions though, so if you want to contribute a fix for this ticket, please reopen or comment on it. If you don't have permissions to reopen this ticket, or you want to discuss an issue, please reach out via the developer mailing list.

#2 - 04/15/2019 10:31 PM - bmbouter

- Tags Pulp 2 added