RPM Support - Story #3740

Implement modularity content dependency solving

06/07/2018 02:53 PM - milan

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	milan	**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:	2.19.0		
Platform Release:	2.19.0	**Tags:**	Pulp 2
Groomed:	No	**Sprint:**	Sprint 47
Sprint Candidate:	No	**Quarter:**	

Description

Motivation

Pulp lacks the ability to perform dependency solving at a module level and to copy dependent modules and their artifacts during copy operation.

Pulp (@2.18) ignores module dependencies when copying modules recursively.
Pulp doesn't distinguish between modular and non-modular artifacts when recursively copying modules to a repository. Pulp however always copies all module artifacts, including the artifacts rpm dependencies.
A potential problem and a rare case:
- a mixture of modular and non-modular packages with the same NEVRA are present in a repo
- unlucky coincidence with versions
- as a result - a "bad" repo (client will likely have a problem with a module which doesn't have all its **modular** rpms in a repo, since some of the copied rpms were non-modular ones).

Proposed solution

It's important for modules to have all their artifacts present in a repo.
Regardless of the depsolving strategy for RPMs, **all** module artifacts and related modules should be copied to a target repo. E.g if the target repository already contains units newer than a module requires, the module-required units need to be copied in the older version, in addition to the already present units.

Modules may depend on other modules, either for the build- or the run-time. Pulp should perform dependency solving based on runtime dependencies only. They are already available on the Modulemd model.

To support these usecases, a fake libsolv solvable can be created for each module unit. This will allow tracking dependencies between the modules. To prevent non-modular content from satisfying dependencies in target repository, it seems exposing the pool->considered bitmap from the libsolv library to its Python binding will be required. This is how DNF deals with solving modular dependencies when installing and upgrading content. This effort is tracked in the sub-task #3741.

References

- the base pulp_modularity support tracker; Issue #3206
- the Modularity project docs page
- the depsolving library investigation #3528 POC
- libsolv
- modular errata
- A cross-team, modular-depsolving document

Subtasks:

Task # 3746: Investigate, whether exposing the @pool->considered@ bitmap is necessary f...	**CLOSED - COMPLETE**

Related issues:

Related to RPM Support - Story #4058: As a user, I can calculate applicabilit...	**CLOSED - CURRENTRELEASE**
Related to RPM Support - Issue #4152: Regression Pulp 2.17.1: recursive copy ...	**CLOSED - CURRENTRELEASE**
Related to RPM Support - Story #4162: As a user, I have dependency solving wh...	**CLOSED - CURRENTRELEASE**
Related to RPM Support - Test #4543: Test newly described definitions of recu...	**CLOSED - COMPLETE**

Copied to RPM Support - Test #4364: Implement modularity content dependency s...	**CLOSED - COMPLETE**

Associated revisions

Revision 62bb34a6 - 12/17/2018 10:04 PM - milan

Introduce modular deps processing

Modulemd and modulemd_defaults are now exposed to the libsolv solver. This allows for the (inter-)modular dependencies to be processed when copying a unit. Bot the relaxed and conservative dependency processing algorithms work with the functionality this patch introduces.

Fixes #3740 https://pulp.plan.io/issues/3740

History

#1 - 06/07/2018 04:38 PM - milan

- Description updated

#2 - 06/08/2018 12:02 PM - milan

- Description updated

#3 - 06/08/2018 12:30 PM - milan

- Description updated

#4 - 07/09/2018 06:29 PM - rchan

- Sprint/Milestone set to 2.17.0

Adding to 2.17.0 milestone. This is one of the required deliverables.

#5 - 07/30/2018 06:25 PM - ipanova@redhat.com

- Sprint/Milestone deleted (2.17.0)

#6 - 08/16/2018 06:13 PM - milan

- Description updated

#7 - 08/16/2018 06:22 PM - milan

- Description updated

#8 - 08/16/2018 06:28 PM - milan

- Description updated

#9 - 08/28/2018 12:58 PM - milan

- Description updated

#10 - 09/04/2018 03:21 PM - ipanova@redhat.com

inter-modular dependency support will lead to a model change of Modulemd. A new field should be added. called `dependencies`

```
dependencies:
- buildrequires:
    httpd: [2.4]
    platform: [el8]
  requires:
    httpd: [2.4]
    nginx: [1.14]
```

```
      platform: [el8]
```

This is an example of how the deps could look. As per spec [0], there might be multiple instances of Modulemd.Dependencies() object, where each of them will describe different way of build with varying build time and runtime.

During recursive copy we do not really care if those modules are required during build time or runtime, I suggest to make a union between `buildrequires` and `requires` per instance of Modulemd.Dependencies() object.

I suggest the following data structure, where each dict will be a translated Modulemd.Dependencies() object.

```
dependencies: [
                              {module-name1:  [stream1, stream2, stream3],
                               module-name2: [stream1, stream2]
                              },
                              {module-name2: [stream3],
                              module-name3: [stream1, stream2]
                              }
                    ]
```

Note** there might me inclusive and exclusive lists of stream, where exclusive is marked with a minus sign. We need to preserve the minus for later conflicts solving.

Note** the `platform` module is synthesized at runtime by dnf, we can ignore it and not include in the deps info at all.

[0] https://github.com/fedora-modularity/libmodulemd/blob/master/spec.v2.yaml#L78

#11 - 09/04/2018 06:58 PM - milan

A new field should be added. called `dependencies`

Another option could be a list of flat dictionaries like:

```
[
    {
```

```
    "name": "foo",
    "streams": ["stable"],
    "type": "runtime"
  },
  {
    "name": "bar",
    "streams": ["master"],
    "type": "build"
  },
  }
    "name": "foo",
    "streams": ["stable"],
    "type": "build"
  }
]
```

This would allow to keep track of the build- vs. run-time dependencies. Plus they say "flat's better than nested".

#12 - 09/04/2018 07:18 PM - milan

milan wrote:

> A new field should be added. called `dependencies`
>
> Another option could be a list of flat dictionaries like:
>
> [...]
>
> This would allow to keep track of the build- vs. run-time dependencies. Plus they say "flat's better than nested".

Actually, it won't work. It would loose the "inner build environments links" which exist so that build environments can have mutually exclusive dependencies and I guess the solver would freak out. Thanks Ina for pointing this out!

#13 - 10/03/2018 01:55 PM - ipanova@redhat.com

ttereshc, now that i look at my comment which i wrote month ago, i cannot recall a reasonable explanation why we would care about buildrequires, unless we build modules which we do not.

+1 to drop buildrequires and include in the proposed structure just the requires.

#14 - 10/03/2018 02:07 PM - ipanova@redhat.com

we should also consider possibly using https://github.com/fedora-modularity/fus

#15 - 11/10/2018 06:32 PM - milan

ipanova@redhat.com wrote:

> ttereshc, now that i look at my comment which i wrote month ago, i cannot recall a reasonable explanation why we would care about buildrequires, unless we build modules which we do not.
>
> +1 to drop buildrequires and include in the proposed structure just the requires.

Thing is, the consumers of our published content might not be able to rebuild the modules themselves i.e we might create an installable but un-buildable repo. This isn't the case for for ursine content now, though we don't process SRPM deps recursively either.

Fus on the other hand considers the build-time dependencies.

#16 - 11/10/2018 06:32 PM - milan

- Related to Story #4058: As a user, I can calculate applicability for modular content added

#17 - 11/10/2018 06:38 PM - ttereshc

To be clear for a quick reader, the relation with #4058 is weak.
#4058 requires only info of the first level module dependencies, if they are enabled on a consumer or not.

List of dependencies (requires only, not buildrequires) is added to the Modulemd model with this commit:
https://github.com/pulp/pulp_rpm/commit/990f65a9bba8e3f9faf2f89165f2a88015dad1c1

#18 - 11/10/2018 06:51 PM - milan

ipanova@redhat.com wrote:

> we should also consider possibly using https://github.com/fedora-modularity/fus

There are some trade-offs with Fus to consider:

- it is a command line tool that we'd have to integrate the way Pungi does
- we'd have to *semi-publish* (all) the source and destination repositories metadata

- this might have performance impact but we can afford experimenting with this idea

On the other hand, reimplementing some parts of both the Pungi and Fus tools seems like introducing code duplicity.
Still, the upside would be more control over how we resolve the dependencies.
There's an (undecided) Fus intgration e-mail discussion that can be queried for further information.

#19 - 11/13/2018 09:27 PM - milan

- Related to Issue #4152: Regression Pulp 2.17.1: recursive copy of RPMs does not copy partially resolvable dependencies added

#20 - 11/15/2018 04:21 PM - milan

- Related to Story #4162: As a user, I have dependency solving when copying modules added

#21 - 12/13/2018 05:15 PM - milan

- Status changed from NEW to POST

Linking the PR https://github.com/pulp/pulp_rpm/pull/1237

#22 - 12/13/2018 05:15 PM - milan

- Checklist item [x] associating a module content unit supports the recursive behavior set to Done
Checklist item [x] associating a module always considers all installation profiles set to Done
Checklist item [x] associating a module considers other modular dependencies set to Done

#23 - 01/07/2019 05:51 PM - ttereshc

- Checklist item deleted (associating a module adhers to the modular upgrade semantics)
Checklist item deleted (file a pulp-smash issue to automate testing recursive module copies)
Checklist item [ ] file a test issue to automate recursive copy of modules and their dependencies added

- Description updated

#24 - 01/07/2019 05:56 PM - ttereshc

- Description updated

#25 - 01/07/2019 06:00 PM - ttereshc

- Description updated

#26 - 01/18/2019 05:34 PM - dalley

- Description updated

#27 - 01/21/2019 08:45 PM - ttereshc

- Description updated

#28 - 01/22/2019 06:19 PM - milan

- Status changed from POST to MODIFIED

- % Done changed from 50 to 100

Applied in changeset 62bb34a6f1290db6f07c372c1a0c3ee9702c3e91.

#29 - 01/25/2019 03:36 PM - rchan

- Sprint set to Sprint 47

Adding to Sprint 47 since this work was done during this time.

#30 - 01/25/2019 03:44 PM - rchan

- Sprint/Milestone set to 2.19.0

Add 2.19.0 milestone since this is a new feature

#32 - 01/29/2019 03:26 PM - amacdona@redhat.com

- Assignee set to milan

#33 - 01/30/2019 03:50 PM - bherring

- Copied to Test #4364: Implement modularity content dependency solving added

#34 - 01/30/2019 03:50 PM - bherring

- Checklist item [x] file a test issue to automate recursive copy of modules and their dependencies set to Done

#35 - 03/14/2019 03:22 PM - bherring

- Related to Test #4543: Test newly described definitions of recursive and recursive_conservative flags added

#36 - 03/14/2019 05:16 PM - ttereshc

- Platform Release set to 2.19.0

#37 - 03/20/2019 10:02 AM - ttereshc

- Status changed from MODIFIED to 5

#38 - 04/02/2019 10:52 PM - ttereshc

- Status changed from 5 to CLOSED - CURRENTRELEASE

#39 - 04/15/2019 10:10 PM - bmbouter

- Tags Pulp 2 added