pulp_deb past, present, future

tell us what pulp_deb development should focus on next

Quirin Pamp, Tobias Grigo

February 6th, 2023

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

THE Linux & Open Source Company!

Consulting

Engineering

Support

Training

AlmaLinux · amazon web services · ANSIBLE · Azure · CentOS · debian · docker · FOREMAN · git · Google Cloud Platform

Istio · kafka · KATELLO · kubernetes · KVM · ORACLE Linux · oVirt · PROXMOX · puppet · RANCHER

Red Hat · RED HAT OPENSHIFT · Rocky Linux · SALTSTACK · SUSE · Terraform · Ubuntu · vmware · Windows

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Past

- Focus on pulp2_deb feature parity
  - Here, feature parity is defined by the needs of Katello
- Focus on successful 2to3 migration

Present

- pulp_deb continues to be part of every Katello release
- orcharhino downstream product uses Katello 4.3, pulpcore 3.16, pulp_deb 2.16.2 in production
- Focus on test coverage and maintainability
- Focus on “the structured content problem”

Future

- Focus on community contributions
- Focus on performance
  - optmize sync with mirror on sync
- Missing APT repo features, e.g:
  - Translation files
  - Package by UUID

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Testing - Goals

- Improve the workflow for contributors to add new tests
- Follow the pulp trend to switch to the pytest framework

Testing - Goals

- Improve the workflow for contributors to add new tests
- Follow the pulp trend to switch to the pytest framework
  - Improving Pulp's Testing, PulpCon 2021
  - Pytest Revolution, PulpCon 2022

Testing - Goals

- Improve the workflow for contributors to add new tests
- Follow the pulp trend to switch to the pytest framework
  - Improving Pulp's Testing, PulpCon 2021 $\rightarrow link$
  - Pytest Revolution, PulpCon 2022 $\rightarrow link$

Testing - Status

- Slow but gradual conversion of old tests
- Fixed the long broken nightly CI
- A proof of concept for locally hosted test data

Testing - Status

- Slow but gradual conversion of old tests → #600 #572 #544 #540
- Fixed the long broken nightly CI
- A proof of concept for locally hosted test data

Testing - Status

- Slow but gradual conversion of old tests → #600 #572 #544 #540
- Fixed the long broken nightly CI → #540
- A proof of concept for locally hosted test data

Testing - Status

- Slow but gradual conversion of old tests → #600 #572 #544 #540
- Fixed the long broken nightly CI → #540
- A proof of concept for locally hosted test data → #704

Testing - Next steps

- Convert the remaining tests
- Increase test coverage
- Speed up testing by making as many tests as possible run in parallel

Local test data fixtures - Status

- Stores pre-compiled deb packages in the repository
- As well a TON of repository metadata files

Local test data fixtures - Next steps

- Simplify the process
- Provide pytest fixtures that will generate repository metadata
- Finally autogenerate the deb packages

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Design Challenge: The structured content problem

https://github.com/pulp/pulp_deb/issues/599

- Has been with us since the start of Pulp 3
- Blocking several proposed features (including community submissions)
- Difficult to fix

Background: APT repo structure (1/4)

- Ignore all obsolete repo formats
- Gloss over some complexities

Background: APT repo structure (2/4)

Example `/etc/apt/sources.list` entry:

```
deb http://ftp.de.debian.org/debian/ bullseye contrib main
    <repo_base_url>            <distribution> <components>
```

- APT repos must define one or more entry points
- Each entry point is defined by a `Release` and/or `InRelease` file
- To find the entry point we need to know the distribution:

```
<repo_base_url>/dists/<distribution>/Release
<repo_base_url>/dists/<distribution>/InRelease
```

Background: APT repo structure (3/4)

- Each release file must define one or more `Components`
- Each release file must declare one or more supported `Architectures`
- Each component must contain a "package index" (aka `Packages` file) for each supported architecture
- Each Package index contain a list of package paragraphs
  - The paragraphs include the path to the `.deb` package file itself

Background: APT repo structure (4/4)

In conclusion…

- Packages are not just "in the repository"
- Packages are associated with a component within a release/distribution

How does pulp_deb store this structure information

We have several “structure” content types:

- `Release`
- `ReleaseComponent`
- `PackageReleaseComponent`

The problem

- The `Release` content currently has the following fields
  - `distribution`
  - `codename`
  - `suite`
- All three fields are part of the uniqueness constraints
- Problem: The `distribution` and only the `distribution` contains the structure information
- What happens is that multiple Releases with the same distribution can collide!
- Other Example: Advanced Copy

Solution 1: Embrace the chaos

Every feature in pulp_deb that encounteres colliding distributions needs to “handle” the situation.

Solution 1: Embrace the chaos

Every feature in pulp_deb that encounteres colliding distributions needs to “handle” the situation.
=> Ever escalating implementation complexity with each new feature and edge chase.
=> Endless potential for bugs
=> Endless test cases needed
=> This is definitely a no-go!

Solution 2: Keep the design, fix the root cause

- We "simply" drop `codename` and `suite` from the `Release` content model.
- Main problem: Requires a very complex one time DB migration.

Solution 2: Keep the design, fix the root cause

- We "simply" drop `codename` and `suite` from the `Release` content model.
- Main problem: Requires a very complex one time DB migration.

=> This is not impossible
=> We have had big and complex DB migrations before

Solution 3: Fix the design

- Maybe using “content” to represent repo structure the way we are doing is the “wrong” way to represent reality?
- What if Pulp had a concept of “sub repositories”?

Solution 3: Fix the design

- Maybe using “content” to represent repo structure the way we are doing is the “wrong” way to represent reality?
- What if Pulp had a concept of “sub repositories”?

=> Package content would NOT be stored in a particular Pulp repository (version)
=> Content would be stored in a Pulp “sub repository”
=> The Pulp reposiotry (version) would reference some sub repository (versions)
=> pulp_deb repositories would have one sub repository for each “ReleaseComponent”

Solution 3: Benefits

- Efficiency: Elimenates about half the “content” per synced repo!
- Ease of use: querrying for packages.
- Ease of use: less content types.
- Could become a pulpcore feature that might be of interest to other plugins?

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Goals

- Speed up sync speeds by not re-syncing packages that have not been changed
- Provide this feature to be on par with `pulp_rpm`
- So katello users can expect same procedure for both plugins

Problems

- Different repository structure deb/rpm
- Previous sync options needed to be stored
- Currently not working with the `mirror` option

Status

- Feature got implemented in `pulp_deb 2.20.0`
- A fix for the `mirror` policy is planned

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Open Community Submissions

https://github.com/pulp/pulp_deb/pull/

- Source package support:
- Flexible release file field handling:
- Signing service improvements:
- New querry filter:
- Bugfixes:
- Documentation:

Open Community Submissions

https://github.com/pulp/pulp_deb/pull/

- Source package support: → #295
- Flexible release file field handling: → #656
- Signing service improvements: → #689 #683
- New querry filter: → #647
- Bugfixes: → #706
- Documentation: → #595

Agenda

1. Introduction: Past, Present, Future
2. Improving testing in pulp_deb
3. Design Challenge: The structured content problem
4. Feature Spotlight: optimize sync mode
5. Open Community Submissions
6. Summary: What is the future of pulp_deb?

Where do we see the future of pulp_deb?

- We want test coverage to guarantee for stability
- We want the capacity to get community submissions in
- We need it to be easy to extend the test coverage
- We need to fix our structure design problem

Where do we see the future of pulp_deb?

- We want test coverage to guarantee for stability
- We want the capacity to get community submissions in
- We need it to be easy to extend the test coverage
- We need to fix our structure design problem

=> Once we have this all capacity can go towards features

Questions: Where do you see the future of pulp_deb?

- What do the people in this room want to see?
- More workflow documentation?

Contact

- Tobias Grigo
  - Email: grigo[at]atix.de
  - GitHub, Matrix: hstct
- Quirin Pamp
  - Email: pamp[at]atix.de
  - GitHub, Matrix: quba42
- Matrix channel: `#pulp_debian`
- Slides will be available at: https://atix.de/config-management-camp-2023/
- Feel three to visit our booth!