APT 2.0 and other news

Have you mooed today?

Julian Andres Klode

July 22, 2019
what's been happening recently?
APT 1.9 code cleanup

- no more deprecated parts
- -3400 lines of C++
- far less includes
- systemd-wide triehash

APT 2.0 and other news

what's been happening recently?

1. APT 1.9 started with the removal of all deprecated API
2. About 3400 lines deleted in C++ code
3. API removals were easy to fix
4. Also cleaned up our includes and fixing this was more annoying.
5. Also removed our internal copy of TrieHash
what's been happening recently?

Ubuntu 14.04+ ESM and "never" pinning

- Ubuntu 14.04 ESM: public dists/, private pool/, requirements:
  - Want to calculate and show available updates
  - Only want to (try) install them if we are authorized

APT 2.0 and other news

- Ubuntu 14.04+ ESM and "never" pinning

1. Ubuntu ESM in trusty now has publicly available dists/, but requires auth for pool/
2. Want to show you available updates, but not (fail to) install them
3. Negative pinning: Good start, but does not cut some corner cases such as `install banana/trusty`
4. If you specify a pin in your preferences file, it might accidentally enable the ESM repo - and that would then cause errors.
5. Never pinning: Pinning a package to "never" pins is to -INT_MAX, and always takes precedence over any other pins
6. Unattended-upgrades currently has a lot of logic to force packages to be installed only from some origins
7. Plan: Pin unwanted origins to never instead.
8. Let’s see how it works out
Ubuntu 14.04 ESM and "never" pinning

- Ubuntu 14.04 ESM: public dists/, private pool/, requirements:
  - Want to calculate and show available updates
  - Only want to (try) install them if we are authorized
- Idea: Let tool that adds the ESM repo install a negative pin for the repository

- Ubuntu ESM in trusty now has publicly available dists/, but requires auth for pool/
  2. Want to show you available updates, but not (fail to) install them
  3. Negative pinning: Good start, but does not cut some corner cases such as install banana/trusty
  4. If you specify a pin in your preferences file, it might accidentally enable the ESM repo - and that would then cause errors.
  5. Never pinning: Pinning a package to "never" pins it to -INT_MAX, and always takes precedence over any other pins
  6. Unattended-upgrades currently has a lot of logic to force packages to be installed only from some origins
  7. Plan: Pin unwanted origins to never instead.
  8. Let’s see how it works out
Ubuntu 14.04+ ESM and "never" pinning

- Ubuntu 14.04 ESM: public dists/, private pool/, requirements:
  - Want to calculate and show available updates
  - Only want to (try) install them if we are authorized
- Idea: Let tool that adds the ESM repo install a negative pin for the repository
- Problem: Too broad pins may re-enable it or some packages (for example, just pinning ‘release trusty’ to priority 100 would also affect trusty ESM)

Solution: A new pin that always takes precedence - the 'never' pin

Usable elsewhere?

APT 2.0 and other news

what’s been happening recently?

Ubuntu 14.04+ ESM and "never" pinning

1. Ubuntu ESM in trusty now has publicly available dists/, but requires auth for pool/
2. Want to show you available updates, but not (fail to) install them
3. Negative pinning: Good start, but does not cut some corner cases such as install
   banana/trusty
4. If you specify a pin in your preferences file, it might accidentally enable the ESM repo -
   and that would then cause errors.
5. Never pinning: Pinning a package to "never" pins is to -INT, and always takes
   precedence over any other pins
6. Unattended-upgrades currently has a lot of logic to force packages to be installed only
   from some origins
7. Plan: Pin unwanted origins to never instead.
8. Let’s see how it works out
Ubuntu 14.04+ ESM and "never" pinning

- Ubuntu 14.04 ESM: public dists/, private pool/, requirements:
  - Want to calculate and show available updates
  - Only want to (try) install them if we are authorized
- Idea: Let tool that adds the ESM repo install a negative pin for the repository
- Problem: Too broad pins may re-enable it or some packages (for example, just pinning 'release trusty' to priority 100 would also affect trusty ESM)
- Solution: A new pin that always takes precedence - the 'never' pin

APT 2.0 and other news

- what’s been happening recently?

1. Ubuntu ESM in trusty now has publicly available dists/, but requires auth for pool/
2. Want to show you available updates, but not (fail to) install them
3. Negative pinning: Good start, but does not cut some corner cases such as install banana/trusty
4. If you specify a pin in your preferences file, it might accidentally enable the ESM repo - and that would then cause errors.
5. Never pinning: Pinning a package to "never" pins is to -INT_MAX, and always takes precedence over any other pins
6. Unattended-upgrades currently has a lot of logic to force packages to be installed only from some origins
7. Plan: Pin unwanted origins to never instead.
8. Let’s see how it works out

Usable elsewhere?

Julian Andres Klode

APT 2.0 and other news

July 22, 2019 4 / 13
what's been happening recently?

Ubuntu 14.04+ ESM and "never" pinning

- Ubuntu 14.04 ESM: public dists/, private pool/, requirements:
  - Want to calculate and show available updates
  - Only want to (try) install them if we are authorized
- Idea: Let tool that adds the ESM repo install a negative pin for the repository
- Problem: Too broad pins may re-enable it or some packages (for example, just pinning 'release trusty' to priority 100 would also affect trusty ESM)
- Solution: A new pin that always takes precedence - the 'never' pin
- Usable elsewhere?

1. Ubuntu ESM in trusty now has publicly available dists/, but requires auth for pool/
2. Want to show you available updates, but not (fail to) install them
3. Negative pinning: Good start, but does not cut some corner cases such as install banana/trusty
4. If you specify a pin in your preferences file, it might accidentally enable the ESM repo - and that would then cause errors.
5. Never pinning: Pinning a package to "never" pins is to -INT_MAX, and always takes precedence over any other pins
6. Unattended-upgrades currently has a lot of logic to force packages to be installed only from some origins
7. Plan: Pin unwanted origins to never instead.
8. Let’s see how it works out
what’s been happening recently?

Locking tales

Last year: Introduced the frontend lock
More issues detected and fixed, mostly in python-apt:
  - Never releasing lock for downloaded .deb after we are done installing them (#922416)
  - Releasing the dpkg lock when closing the cache
Result: Less crashes from unattended-upgrades in Ubuntu’s error tracker :-(

APT 2.0 and other news

1. Introduced the frontend lock for dpkg to have a lock that frontends can hold all the time while planning and executing dpkg.
2. solved a few race conditions between dpkg invocations where dpkg lock was released to run dpkg and someone else could steal it
3. Never releasing lock for downloaded .deb after we are done installing them (#922416)
4. dpkg lock should be kept as long as possible, but we released it when closing a cache
5. these things broke unattended-upgrades a lot
6. In Ubuntu, we now see significantly less crashes
what’s been happening recently?

(Not) shutting down

APT 1.8.1 and newer now hold a systemd inhibitor while installing packages
Prevents you from shutting down the system using systemctl and GUI
Does not work for sudo reboot and friends (systemd issue #2680)

APT 2.0 and other news

what’s been happening recently?

(Not) shutting down

1. APT 1.8.1 and newer now hold a systemd inhibitor while installing packages
2. Prevents you from shutting down the system using systemctl and GUI
3. Does not work for sudo reboot and friends (systemd issue #2680)
Breaking buster and testing

what’s been happening recently?

E: Repository ‘foo’ changed its ‘Suite’ value from ‘testing’ to ‘stable’
E: Repository ‘bar’ changed its ‘Codename’ value from ‘buster’ to ‘bullseye’

SORRY
What did we want to do?

- Security: Prevent people from giving you stable when you ask for stable-security
- Pinning: Prevent broken pinning situations

Can we do better? Some ideas...

1. Security: We already have a warning if distribution in repo does not match the one configured in sources.list. Make that an error?
2. Pinning: Can we warn people if they have pins that match something but they refer to it differently than in sources.list?


Julian Andres Klode
What did we want to do?

- **Security**: Prevent people from giving you stable when you ask for stable-security
- **Pinning**: Prevent broken pinning situations

Can we do better? Some ideas...

- **Security**: We already have a warning if distribution in repo does not match the one configured in sources.list. Make that an error?
- **Pinning**: Can we warn people if they have pins that match something but they refer to it differently than in sources.list?

what's been happening recently?

**dpkg**

News:
- Trigger loops might finally be over
- Rootless builds
- start-stop-daemon learned the new systemd readiness protocol
- Dpkg perl modules are on CPAN now
- cross-compilation support improved

Other stuff:
- /var/lib/dpkg is private
- Blocks https://wiki.debian.org/Teams/Dpkg/Spec/MetadataTracking

APT 2.0 and other news

1. Trigger loops might finally be over. They have plagued a lot of places, and also a lot of Ubuntu upgrades failed
2. Rootless builds are now possible with Rules-require-root: no
3. start-stop-daemon learned the new systemd readiness protocol
4. Dpkg perl modules are on CPAN now
5. and cross-compilation support improved
6. The dpkg database should be considered a private implementation detail
7. Do not access the database directly, use provided interfaces
8. Users of dpkg database will break if db format changes
9. this is blocking for several improvements, one being metadata tracking for installed files
APT 2.0, what to want?
APT 2.0, what to want?

Patterns? I want to `apt purge ?and(?installed,?config-files)`

Pinning by patterns?

1. aptitude has this neat feature where you can specify patterns to match against packages
2. for example, you could purge all packages in config-files state
3. Want the same thing for APT, and allow pinning on that
4. Started working on (re-)constructing the syntax tree
5. Probably won’t have all the features
6. Should this be extensible so you can provide additional pattern providers as plugins?
7. Want to build a lookup structure into an apt cache so we can find packages based on their source package
8. Can be used for pinning, or for upgrading by source package
9. Started working last Summer, got stuck or well, lost interest
10. Difficult to preserve some greediness aspects in global optimization
APT 2.0, what to want?

Can I haz . . .?

Patterns? I want to `apt purge` and installed, config-files.

Pinning by patterns?

Pin and upgrade by source package?

---

1. **Patterns?** I want to `apt purge` and installed, config-files.
2. **Pinning by patterns?**
3. **Pin and upgrade by source package?**

---

1. `aptitude` has this neat feature where you can specify patterns to match against packages.
2. For example, you could purge all packages in config-files state.
3. Want the same thing for `APT`, and allow pinning on that.
4. Started working on (re-)constructing the syntax tree.
5. Probably won’t have all the features.
6. Should this be extensible so you can provide additional pattern providers as plugins?
7. Want to build a lookup structure into an apt cache so we can find packages based on their source package.
8. Can be used for pinning, or for upgrading by source package.
9. Started working last summer, got stuck or well, lost interest.
10. Difficult to preserve some greediness aspects in global optimization.
APT 2.0, what to want?

Can I haz . . .?

Patterns? I want to `apt purge ?and(?installed,?config-files)`
Pinning by patterns?
Pin and upgrade by source package?
A better solver?

1. aptitude has this neat feature where you can specify patterns to match against packages
2. for example, you could purge all packages in config-files state
3. Want the same thing for APT, and allow pinning on that
4. Started working on (re-)constructing the syntax tree
5. Probably won’t have all the features
6. Should this be extensible so you can provide additional pattern providers as plugins?
7. Want to build a lookup structure into an apt cache so we can find packages based on their source package
8. Can be used for pinning, or for upgrading by source package
9. Started working last Summer, got stuck or well, lost interest
10. Difficult to preserve some greediness aspects in global optimization
Mandatory InRelease

- Recently proposed to make InRelease files mandatory, not using Release and Release.gpg anymore
- Questions:
  - What about unsigned repositories (especially w/o a Release file)? Do we still care about them? Should we?
  - Making (In)Release mandatory as a first step means we require signatures for all non-Release files - no need to guess which files exist
  - What about bugs in the signed → unsigned fallback? Can we be sure we don’t accidentally fall back to unsigned?
- Not sure yet how to proceed

APT 2.0 and other news

- Recently proposed to make InRelease files mandatory, not using Release and Release.gpg anymore
- Questions:
  - What about unsigned repositories (especially w/o a Release file)? Do we still care about them? Should we?
  - Making (In)Release mandatory as a first step means we require signatures for all non-Release files - no need to guess which files exist
  - What about bugs in the signed → unsigned fallback? Can we be sure we don’t accidentally fall back to unsigned?
- Not sure yet how to proceed