iRODS Build and Packaging: 2024 Update

Markus Kitsinger
Software Developer and Build Engineer
iRODS Consortium

May 28-31, 2024
iRODS User Group Meeting 2024
Amsterdam, Netherlands

Overview

- Versioning changes
- Recap of plans
- Current progress
- Externals
  - `libstdc++`
  - New Features
  - Version Freezes
- Other things that happened
- Notable yaks in need of shaving
- Other considerations

Versioning changes - iRODS 4.4 is now 5.0

- Following the 4.3.x series, major releases of iRODS will be versioned 5.0, 6.0, 7.0, etc
  - Minor releases will be versioned 5.1, 5.2, 5.3, etc
- Gnome-style development versioning - between releases, code in repository will have development version numbers
  - x.9# for major releases/main branch (ex: 4.90 is development version of 5.0.0)
  - x.y.9# for minor releases/stable branches (ex: 5.1.92 is development version of 5.2.0)
- Custom CMake package version file
  - Takes into account development versions
  - Takes into account the versioning change for 5.0
  - More info in issue #7532

Recap of Plans

- We will shift to using the standard tools (`dpkg-buildpackage` and `rpmbuild`) for packaging
  - `git-buildpackage` will be used to maintain debian packages, Salsa-style
    - Possibly rpm packages as well, still investigating
  - We will not provide an externals package if the distribution already provides a usable package
  - Debian and rpm source packages will be provided in our repositories
  - We will follow established patterns for setting up service accounts
  - We will install our libraries in the normal locations
  - We will provide default systemd unit(s)
- We will build against `libstdc++`
- We will decouple the iRODS buildsystem from externals packaging implementation details

Current Progress

- We decided to transition to `libstdc++` before moving away from `fpm`/CPack.
  - This will allow us to reduce the number of externals packages we provide, which will facilitate the transition to standardized packaging for externals.
  - This also meant putting a lot more effort into the current externals system than I really wanted to. More on this in a bit.
- iRODS 5.0 will be built against `libstdc++`.
- Some externals are now used in CMake via `find_package`.
- `mungefs` buildsystem completely decoupled from externals.

Externals - `libstdc++`

- In order to build iRODS against `libstdc++`, externals needs to build against `libstdc++` as well.

- iRODS 4.3.x is still built against `libc++`, so we have to support both until 4.3 is EOL.

- Some packages now have two variants, one for `libstdc+++` and one for `libc++`.

Externals - New and Removed Packages

- New package - `jwt-cpp` - Added for HTTP API provider client
- Several packages removed
  - `cpr` - no single version compatible with all versions of curl we must support
  - `elasticlient` - built on top of `cpr`
  - `jansson` - replaced by nlohmann-json (`json`)
  - `pistache` - only used by one project, which has now been sunset
  - `libs3` - all relevant code has been merged into the s3 resource plugin
  - `aws-sdk-cpp` - originally added for s3 resource plugin, but was never used
- Exploring removing more
  - `libarchive` - distro-provided packages should be sufficient once EL7 is dropped (#7286)
  - `redis` - distro-provided packages should be sufficient once EL7 is dropped (#7478)
  - `zeromq4-1` - distro-provided packages should be sufficient once EL7 is dropped (#7479)
  - `json` - distro-provided packages may be sufficient already (#7726)

Externals - New Features

- Each package now declares dependencies per-distribution per-version.
- New source patch system
  - Initially added for Ubuntu 24.04 support, as `clang` required changes a little too complex for basic shell scripting to handle.
  - We now pull in some patches from distribution packages.
- Package revisions are now properly supported, allowing for in-place upgrades.

Externals - Version Freezes

We want to transition to distro-provided packages where possible. In order to facilitate this, we have implemented a soft version freeze on most of our externals.

- Bumping the version of an externals package that has a distro-provided equivalent (or is likely to have one in the future) needs to be sufficiently justified.
- Externals packages unlikely to ever have a distro-provided equivalent (such as `jwt-cpp`) are free from this restriction.
- This presents its own challenges we have had to overcome:
  - Our `clang` externals package supports C++ coroutines, but we cannot use them due to an incompatibility with `libstdc++`.
  - Our `cmake` externals package does not support the newer versions of Python used on some distributions.

Other Things That Happened

- s3 resource plugin has absorbed `libs3`.
  - s3 resource plugin has been relicensed to LGPLv3+/GPLv2+.
- Python rule engine plugin build has been properly parallelized.
- Development environment `Dockerfile`s use new `Dockerfile` syntax.
- Development environment core builder now uses `ccache`.
- We now leverage CMake object library targets to improve our buildsystems in a number of ways.

Notable Yaks In Need of Shaving (that we know about)

- Non-package installation - `make install` should be enough
  - Side-by-side database plugin installation (#5999)
  - URI json schema IDs (#6283)
- File/directory ownership
- File/directory location
- Default configurations (or something along those lines)
- Unprivileged builds in CI and development environment
- CMake target names
- Removal of externals plumbing in CMake

Other Considerations

- How will this affect development workflows?
  - How will we document this?
- How will CI need to change?
- When are cleanroom builds needed and how will we support them?
- How will we verify our dependency minimum versions?
  - How often should we do this?
- Presently, a lot of functionality for standing up and cleaning up after iRODS is handled by Python scripts. How much of this should be migrated into iRODS proper, and how?
- How long should we maintain legacy CMake target aliases?
- How will the upgrade process be affected?

Thank you!

Questions?