



# **iRODS Build and Packaging: 2025 Update**

Markus Kitsinger  
Software Developer and Build Engineer  
iRODS Consortium

June 17-20, 2025  
iRODS User Group Meeting 2025  
Durham, NC

- Milestones
- iRODS 5.0
- Versioning changes
- New iCommands installation location
- Toolchain update - Clang 16
- `distro_distill` - New Python module
- Other things that happened
- Up next
- Other considerations

# Milestones

- Improve iRODS's integration with the system
  - Libraries in the normal locations\*
  - ~~systemd service manager communication~~
  - Multi-instance support
  - Default systemd unit(s)\*
  - Default configuration(?)
  - Less server functionality in Python scripts\*
  - ~~Build against default C++ standard library~~
  - No more shoving everything in `/var/lib/irods`
  - Fewer files/directories owned by service account
  - Side-by-side installable database plugins
- Improve iRODS's buildsystem
  - Fewer opinions and assumptions in CMake
  - Fewer overridden defaults in CMake
  - No reliance on externals packaging implementation
  - CMake components in CMake configuration
  - Proper CMake target names
  - `pkg-config` support(?)
- Migrate from CPack to standard tools for packaging (`dpkg-buildpackage` and `rpmbuild`)
  - No externals package for a dependency if the distribution already provides a usable package
  - Debian and RPM source packages provided in our repositories
  - Service accounts created following established patterns
  - Debian (and possibly RPM) packages maintained with `git-buildpackage`, Salsa-style

# What's new with build and packaging in iRODS 5.0?

- Building against `libstdc++`.
- systemd support.
- Libraries in proper location (for now only on Enterprise Linux).
- Server has absorbed some functionality previously handled by Python scripts.
- New version numbering.
- New iCommands installation location - symlinks in original location.
- Updated toolchain to Clang 16.
- Reduced reliance on externals:
  - Some externals migrated to system packages:
    - CMake
    - nlohmann-json
    - `libarchive`
    - fmtlib
    - `spdlog`
    - Catch2
  - Some externals eliminated entirely:
    - avro
    - ZeroMQ
    - `cppzmq`

## Versioning changes - Recap

- Starting with iRODS 5.0, major releases of iRODS are versioned 5.0, 6.0, 7.0, etc.
  - Minor releases will be versioned 5.1, 5.2, 5.3, etc.
  - Patch releases will be versioned 5.0.1, 5.0.2, 5.0.3, etc.
- Gnome-style development versioning - between releases, code in repository has development version numbers:
  - x.9# for major releases/main branch (ex: 5.90 is development version of 6.0.0).
  - 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 prior to 5.0.
  - More info in issue [#7532](#).

- Still migrating to new system in some areas.
- Patch releases will be API and ABI compatible, barring exceptional circumstances.
  - Starting with 5.1, library sonames will only include major and minor version ([#8543](#)).
- Some sister projects (plugins, clients) now decoupled from iRODS release cadence.
  - iRODS version number at build-time stamped on package revision strings.
  - Eliminates the need for tagged release for every iRODS update.
  - Eventually, rebuilds will not be needed for iRODS patch releases ([#8544](#)).

## New iCommands installation location

- iCommand executable names collide with other projects, causing package conflicts, such as with `slleuthkit`, which also has an `ils` executable.
  - Debian and Ubuntu can work around this with `dpkg-divert`, but no solution for EL.
- iCommand executables now installed by default in `LIBDIR/irods/clients/icommands`.
  - Installation location configurable via CMake variable.
  - Symbolic links provided in original locations (toggleable by CMake variable).
- Starting with 5.1, the symbolic links will be split into a new package.
  - Original package will include source-able shell scripts for common shells to add iCommands installation directory to `PATH`.
  - We may also provide module files for `modules/Lmod`.
- More info in issues [#8361](#) and [#8496](#).

# Toolchain update - Clang 16

- iRODS 4.3 was built with Clang 13.
  - Released in 2021.
  - Sometimes crashes when compiling C code containing VLAs.
  - `constexpr` is broken.
  - Incompatible with `libstdc++`'s coroutines.
  - Incompatible with large swaths of newer versions of `libstdc++` ([llvm/llvm-project#44178](https://llvm.org/issues/44178)).
- iRODS 5.0 built with Clang 16.
  - Released in 2023.
  - Many bug fixes.
    - `constexpr` works!
  - Better `libstdc++` compatibility - can be built against distro's default `libstdc++`.
  - Should ease transition to distro-provided compilers.

## `distro_distill` - New Python module

- Companion to the `distro` module.
- Provides additional information about the current Linux distribution.
  - Mainly information about the upstream distribution, if any, on which the current distribution is based (Ubuntu for Mint, RHEL for AlmaLinux, etc).
- Will be executable, like `lsb_release`.
- Still very much a work in progress.
- Started as `distro_info.py` in the externals repository.
- Currently lives at `scripts/irods/distro_distill.py` in the main iRODS repository.
- Will eventually be its own project and package.

# Other Things That Happened

- Support for Ubuntu 20.04, Debian 11, and EL8 dropped for iRODS 5.0+.
  - 4.3 will continue to support these distributions.
- CI workflows migrated to Ubuntu 24.04.
- More databases added to testing environment:
  - Postgres 16
  - MariaDB 10.6
  - MariaDB 11.4
  - MySQL 8.4
  - MariaDB 10.11
- Python Rule Engine Plugin updated for Python 3.12.
- Python 2 support code removed from Python Rule Engine Plugin.
- RPM packages signed with `sha384` digest algorithm.
- Implemented support for systemd as entrypoint for development environment runner containers.
- CMake minimum version requirements raised to 3.12.
- Migrated all Rocky Linux containers to new base image location.

## Up next (tentative, non-exhaustive)

- EL10 support.
- Remove unused externals packages:
  - CMake
  - spdlog
  - Catch2
  - fmtlib
  - nlohmann-json
  - Redis
- Update `mungefs` for new CMake versions.
- Use distribution compilers for building externals, where possible.
- Userspace packager update.
- Externals packager migration from `fpm` to `nFPM`.
- Libraries in the normal locations on Debian/Ubuntu.
- `init.d` script for systemd-less environments.
- `main()` in plugins and libraries to provide build/runtime information.
- Proper CMake target names.
- More splitting up of `/var/lib/irods`.
- Side-by-side installable database plugins.
- Headers in CMake target sources lists.
- Testing environment improvements.

## Other considerations

- How will buildsystem/packaging changes affect development workflows?
  - How will we document this?
- How will CI need to change?
  - How should we handle CI for sister projects?
- 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?
- How long should we maintain legacy CMake target aliases?
- How will the upgrade process be affected by packaging and system integration changes?
- How should our package repositories be maintained? Is our current system sufficient?

Thank you!

Questions?