2025-03-21

R package binaries for Linux - Community Edition

Dr. Patrick Schratz
devXY GmbH

Signatories

Josiah Perry has acknowledged the usefulness lately and even donated for it.

Sharing the project in the `rstats` subreddit resulted in some very positive reactions.

As my social media accounts are not having the greatest reach and Twitter/X is basically dead, the reactions there were limited.

A blog post shared on my website has been shared in rweekly in December 2024.

Last, a variety of business clients of mine have shared their enthuisiams with me personally, hoping to see a positive development of the overall project idea.

Project team

Patrick Schratz, CEO devXY.

More than 10 years of R experience, PhD in applied machine learning, devOps Engineer. Architecting and implementing R-based data science environemnts for corporate teams. Deep understanding of cloud architectures, UNIX system administration and how to use R in the most performant ways.

Contributors

None right now.

Consulted

None so far.

The Problem

Linux binary packages

Binary Packages are essential for efficient workflows in R. Currently, CRAN is building and publishing binary packages for Windows and macOS. Linux binaries are missing, even though most (corporate) team environments run on Linux. In addition, the share of private R users using Linux as their host OS becomes larger every year.

Without binaries for Linux, package installations can take many minutes, sometimes even up to hours (e.g. when packages like {duckdb} or {Rfast} are involved). To my knowledge (spanning mostly from community discussions), CRAN does not seem to have plans building binary packages for Linux. Especially after Posit launched their public Package Manager Service in 2020, providing Linux binaries for different distributions on the `x86_64` architecture, the motivation to do so might been reduced even further.

Public Posit Package Manager

The PPM has been a valuable resource for the R community since its launch. However, it has the following drawbacks:

- The build process is untransparent / not public
- The download speed is at best "acceptable"
- The usage of the binaries comes with a TOS agreement. This can be problematic for specific use cases and poses a general risk to users, as it is often overlooked and might cause (costly) architecture adjustments once it is realized
- Binaries for the arm64 architecture are missing

Alpine Linux

All of the solutions above are missing out on Alpine Linux. This Linux distribution is the de-factor standard for any CI/CD builds, whether running containerized or not, due to the small size of its OS and related system libraries.

The distribution is using a different C library (MUSL) instead of the canonical default GLIBC. This leads to issues for some packages wrapping C-code and authors needs to adjust for it. Not many do, unless they have a focus on Alpine, also because CRAN does not perform any MUSL-based checks for package submissions.

In a recent correspondence with the CRAN team in January 2025, adding such checks are also not of interest for the CRAN team right now. The lack of Alpine R package binaries makes R a second-class citizen in the modern CI/CD world.

Architecture: arm64

R package binaries for the `arm64` architecture are missing completely right now. Even Posit has not done any efforts yet in building for this architecture, even though they stated that they want to start doing so in 2025 (in a correspondence from 2024).

The `arm64` architecture has been becoming increasingly popular in recent years and is well supported on many cloud providers. Servers with this architecture provide cost-saving opportunities compared to their `x86` equivalents and also often outperform them in CPU benchmarks.

The lack of R package binaries (and interpreter binaries) makes R a second-class citizen in this space.

R Universe

R Universe, an existing R-consortium funded project, is a first step to an alternative packaging system for the R community.

However, as of today, R universe only builds binary packages for the latest LTS release of Ubuntu (filtered on Linux in general). Packages for other architectures, distributions or architectures are missing.

In addition, the build process heavily relies on GitHub Actions. While this might seem a positive aspect for some on the first look, I'd argue that it is effectively a downside for the following reasons:

- GitHub's public runners are rather slow compared to evenly-sized cloud VMs
- GitHubs default free build minutes are quite limited and the costs for adding additonal ones are quite high, compared with the alternative of providing private runners
- GitHub did not have arm64 support until recently, and the support itself is in an relatively early stage with limited build capacities

All of this could be done much more efficiently, both in terms of costs and resource efficiency.

Another lacking point is that the engine behind the package builds, its infrastructure and build process is somewhat abstract. Yes, every package has its own repository executing the package builds in the public on GitHub Actions, yet the underlying code and process is unclear and not easily visible. Neither can uses re-use the logic to build and publish binaries themselves, they fully need to rely on the magic done by R Universe in the background.

Creating custom repositories

All of the above do not help if teams want to create and maintain their own private repository containing a selected suite of (internal) R packages. These days, some tools exist that allow achieving this: {minicran} and drat. However, these do not provide the option to (easily) build binary packages or manage the packages in S3 buckets.

Ideally, it should be as easy as running a single function (or two) which inits the remote storage, builds packages from a local source or remote URL and returns the final repository URL for download in the end.

The motivation behind such a solutin has also increased in the past year after Posit has drastically increased the pricing for its Package Manager product. Before, the product was reasonably priced and providing a convenient way to set up and manage internal R repositories. After the price change, the application became unaffordable for many companies considering its scope, leading many teams to reconsider their choice of tools (a conclusion drawn from my personal experience in daily consulting practice).

The proposal

Establish an **open-source** build system for R package binaries. Not only the source code should be public, but also the actual build processes (running in CI) should be publicly accesible.

The build system should be able to build packages for (common) Linux distributions and multiple architectures (for the start, `x86` and `arm64`, `riscv64` in the forseeable future). As R is the best tool

for building its own packages, thanks to projects like {pak}, {cranlike}, {pkgdepends}, {cranberries} and others, the core part will be done in R itself. This also allows for possible contributions from the community itself, as the codebase should be familiar to them.

The engine should be written in a generic way, so that it can be used to build binaries from any R package source, being it a local or remote one. It should allow publishing to custom repositories, so that the community has a "go-to" tool for building and creating their own private repositories.

Users should optionally be allowed to store the build metadata in a database (SQLite, PG, MariaDB), allowing for optional statistical analysis and other post-hoc analysis.

The main storage for the binaries itself is S3 (as the only option for now). S3 allows for flexible data storage and replication while keeping costs small. Currently, no existing build tools allow using S3 directly, i.e. binaries must be stored on hard disks. Doing so increases storage costs by many factors and becomes a problem for medium-large sized repositories.

Storing packages in S3 allows for direct distribution of such to users. Yet often enough, the raw S3 URLs are not pretty enough to use them in production environments. A simple CNAME entry can often help to resolve this and allows for a simple, custom-domain access for users. Optionally, S3 can serve as the source for a Content Delivery Network (CDN), which allows for accelerated download speed.

As of today, the current system in place provides binaries for

- Ubuntu 24.04
- Ubuntu 22.04
- RHEL8
- RHEL9
- Alpine 3.20
- Alpine 3.21

for both `x86_64` and `arm64` architectures. The download speed is 5-10x faster than when downloading binaries from the Posit Package Manager. The build process is fully transparent (https://ci.devxy.io/repos/7).

Project plan

Start-up phase

The project is already up and running with respect to the core components and infrastructure.

Technical delivery

An up-and-running global package binary download service which processes daily incoming changes of CRAN packages (additions / removals / updates).

For the core engine behind, continuous releases and improvements.

Other aspects

The project would profit from more public promotion and user testing/use.

Presentations at common R conferences are planned and are important for establishing the project within the community.

Requirements

General

- Support for continuous development of the code engine
- Cost coverage for the public running service building CRAN packages for various distributions and architectures

People

Additional people which ensure an upkeep of the public package build service and the underlying servers.

The project can be run by a single person technically, more people would be helpful to improve the project code and robustify the service maintenance.

Tools & Tech

- For the public CRAN build service, a cloud infrastructure (already existing) which provides native `x86_64` and `arm64` runners that are able to natively build the respective packages. The servers need decent CPU power to minimize build times and being able to process all existing packages, in scenarios of major OS version updates, in a reasonable time.
- A (professionally) managed database storing the build metadata (already in place).
- S3 storage in the cloud (currently using Hetzner, 1TB/m = 5 EUR).
- UI frontend showing available packages (optional). Currently, a self-built Shiny App Dashboard is in use (https://app.devxy.io/app/r-package-binaries-dashboard)

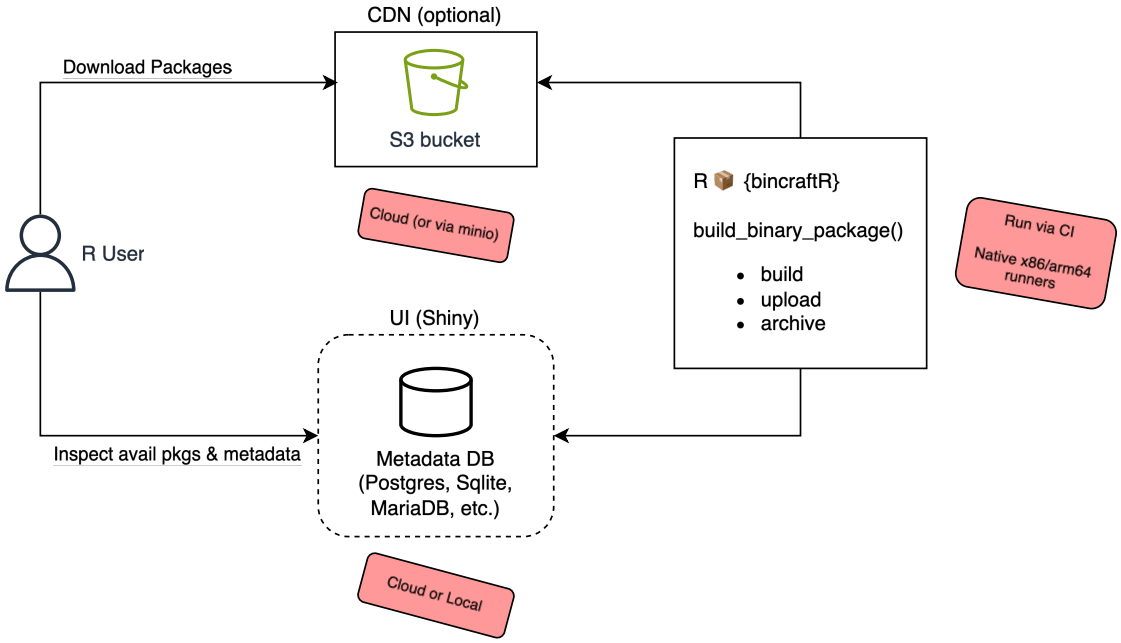

Figure 1: High level architecture diagram

Funding

There are two main types of funding:

1. General maintenance and improvement of the "engine" and its related documentation
2. Coverage of cloud costs for the running service providing binaries for CRAN packages

For (1), I do not know if there are hard-coded policies in place for human workforce costs per project/hour. It would surely be great to have financial support for this, but overall, (2) is more important in my opinion.

For (2), I'll outline a cloud cost overview which will be able to efficiently provide the technical resources to run the overall CRAN build service for the forseeable future:

Name	~ Cost/month	~ Costs / year	Notes
`x86_64` Server (8 cores, 16 threads, 64 GB RAM)	54 Euro	648 Euro	Used to build `x86_64` binaries, bare-metal root server
x2 `arm64` Server (16 cores, 32 GB RAM)	52 Euro	624 Euro	Used to build `arm64` binaries, shared Cloud Server, price for two units
Database Servers (HA)	12 Euro	144 Euro	HA Database for build metadata

Name	~ Cost/month	~ Costs / year	Notes
S3 storage costs	15 Euro	180 Euro	Costs for 3 TB, current storage is at 2.14 TB
CDN costs	?? Euro	?? Euro	CDN bandwith/transfer costs. 10$/1 TB bandwith, + local storage zones around the world

While the core setup (servers, S3, DB) should stay quite static at ~ 150 / month (1800 / year), the bandwith costs for fast downloads around the world is dynamic. If the service is picked up by the community on a large scale, costs of multiple hundreds per month could occur, when with a cost-efficient CDN provider.

Summary

The project would benefit from workforce costs for continous development of the core engine and coverage of the running cloud costs. Increasing the bus-factor for people with access to the core components of the infrastructure would make the project more robust against failures or human absences.

Success

Definition of done

1. A running, public build service providing R package binaries for Linux to the community which has caused minimal interruptions for users across a timespan covering multiple OS updates.
2. A framework/engine for building binary packages, written in R, with extensive documentation around its usage and optional extensions (e.g. metadata database), allowing users to easily host and use their own package repository.

Measuring success

- Download counts
- Tracking missing/corrupted packages in the public service
- Gathering user feedback on the overall process of creating custom user repositories

Future work

Community contribution should be encouraged and can help to sustain for the project for a long time. Overall, it would be super cool to have an open, free and professional tool which allows building and maintaining custom user repositories while also providing a highly performant binary package repository of CRAN packages in the first place.

Key risks

The following possible risks are associated with the project:

- Overall acceptance in the community and trust of the service and build process

- Competition from large players in the field (e.g. Posit) offering proprietary software
- Bus-factor: multiple persons must have access to all required components of the infrastructure to avoid a single-person bottleneck
- Lack of (long-term coverage) of the costs of the running build service (especially the bandwith costs in case there will be a large community adaption).