Package: containr (via r-universe)

May 20, 2026

Title Containerize Your 'R' Project

Version 0.1.3.9000

Description Provides tools for containerizing 'R' projects. Reads an 'renv' lock file, generates a ready-to-use 'Dockerfile', builds the container image, and pushes it to a registry. Generated 'COPY' instructions preserve the local directory structure inside the container. Cross-platform builds are supported via 'Docker' buildx for targeting 'linux/amd64' from ARM hosts. Designed to help researchers build portable, reproducible workflows that can be reliably shared, archived, and rerun across systems. See R Core Team (2025) <https://www.R-project.org/>, Ushey et al. (2025) <https://CRAN.R-project.org/package=renv>, and Docker Inc. (2025) <https://www.docker.com/>.

License Apache License (>= 2)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

URL https://github.com/erwinlares/containr, https://erwinlares.github.io/containr/

BugReports https://github.com/erwinlares/containr/issues

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0), withr

Imports cli, fs, glue, httr2, jsonlite, purrr, readr, remotes, renv

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

Repository https://erwinlares.r-universe.dev

Date/Publication 2026-05-20 18:29:41 UTC

RemoteUrl https://github.com/erwinlares/containr

RemoteRef HEAD

RemoteSha bbc36e80d190126aa377f0c87eaeea480a0602a5

Contents

`build_image` *Build a container image from a Dockerfile*

Description

`build_image()` builds a container image from a `Dockerfile` using either `podman` or `docker`. It auto-detects which tool is available on the system unless `tool` is specified explicitly. Use `dry_run = TRUE` to preview the exact command that would be run without executing it.

Usage

```
build_image(
  dockerfile = "Dockerfile",
  tag = NULL,
  platform = "linux/amd64",
  tool = NULL,
  dry_run = FALSE,
  verbose = FALSE,
  comments = FALSE
)
```

Arguments

`dockerfile`	A character string. Path to the `Dockerfile` to build from. Defaults to `"Dockerfile"` in the current working directory.
`tag`	A character string or `NULL`. The full image tag to assign to the built image, including the registry prefix, e.g. `"registry.doit.wisc.edu/netid/myimage"`. If `NULL`, no tag is applied and the image is identified only by its image ID. Defaults to `NULL`.
`platform`	A character string or `NULL`. The target platform for the container image. Defaults to `"linux/amd64"`, which is the architecture used by most HPC and HTC clusters. Set to `"linux/arm64"` for ARM-based systems (Apple Silicon, AWS Graviton). Set to `NULL` to let the container tool build for the host architecture.
`tool`	A character string or `NULL`. The container tool to use for building. One of `"podman"` or `"docker"`. If `NULL` (the default), the function auto-detects which tool is available, preferring `podman` if both are found.
`dry_run`	Logical. If `TRUE`, prints the command that would be run without executing it. Useful for verifying the command before committing to a potentially slow build. Defaults to `FALSE`.

verbose	Logical. If TRUE, prints progress messages at each step. Defaults to FALSE.
comments	Logical. If TRUE, prints explanatory context before each step – what the command does, why it is needed, and common pitfalls. Useful for first-time users learning the container build workflow. Defaults to FALSE.

Details

When the target platform differs from the host architecture (e.g. building linux/amd64 on an Apple Silicon Mac), build_image() automatically uses docker buildx build instead of docker build, and includes --load to ensure the image is available in the local store. For podman, --platform is passed directly to podman build.

Value

Called for its side effects. Returns invisible(NULL).

Prerequisites

Before calling build_image(), ensure the following are in place:

1. A Dockerfile exists at dockerfile. Use generate_dockerfile() to create one if needed.
2. An renv.lock file is present in the build context – the generated Dockerfile uses it to restore the R package environment inside the container.
3. Either podman or docker is installed and the daemon (for docker) or the Podman service is running. Verify with podman info or docker info in a terminal.

Cross-platform builds

Building for a different architecture than the host requires emulation. On Apple Silicon Macs, building linux/amd64 images uses QEMU emulation under Podman, which can be slow and unstable. Docker Desktop handles cross-platform builds more reliably via buildx and Rosetta 2.

If builds fail with QEMU segfaults, consider:

- Using Docker Desktop instead of Podman (tool = "docker")
- Building on a native x86_64 machine (e.g. via GitHub Actions)
- Building directly on the target cluster if it supports container builds

Tagging convention for CHTC

For UW-Madison CHTC, the full tag format is: registry.doit.wisc.edu/<netid>/<image-name>:<version>

For example: registry.doit.wisc.edu/erwin.lares/my-analysis:1.0.0

The version tag defaults to latest if omitted. Using explicit version tags (e.g. 1.0.0) is recommended for reproducibility – latest will be overwritten each time you push.

Examples

```
## Not run:
# Build for linux/amd64 (default) with auto-detected tool
build_image()

# Build and tag for CHTC registry
build_image(tag = "registry.doit.wisc.edu/netid/my-analysis:1.0.0")

# Build for the host architecture (no --platform flag)
build_image(platform = NULL)

# Build for ARM64 (e.g. local use on Apple Silicon)
build_image(platform = "linux/arm64")

# Preview the build command without running it
build_image(
  tag     = "registry.doit.wisc.edu/netid/my-analysis:1.0.0",
  dry_run = TRUE
)

# Guided build for first-time users
build_image(
  tag      = "registry.doit.wisc.edu/netid/my-analysis:1.0.0",
  verbose  = TRUE,
  comments = TRUE
)

## End(Not run)
```

`generate_dockerfile` *Generate a reproducible Dockerfile for an R project*

Description

`generate_dockerfile()` inspects an R project's dependencies via an `renv` lockfile and writes a ready-to-use `Dockerfile` to the specified output directory. It supports multiple Rocker base images, automatic system library detection, Quarto installation, file copying, user creation, and inline documentation comments.

Usage

```
generate_dockerfile(
  r_version = "current",
  r_mode = "base",
  auto_syslibs = TRUE,
  install_syslibs = NULL,
  output = tempdir(),
  data_file = NULL,
  code_file = NULL,
```

```
  misc_file = NULL,
  add_user = NULL,
  home_dir = "/home",
  expose_port = "8787",
  install_quarto = FALSE,
  comments = FALSE,
  verbose = FALSE
)
```

Arguments

r_version	A character string specifying the R version to use, e.g. "4.3.0". Defaults to "current", which resolves to the version of R running in the current session.
r_mode	A character string selecting the Rocker base image. Inspired by the Rocker Project. One of "base" for plain R, "tidyverse" for R with the tidyverse, "rstudio" for RStudio Server, or "tidystudio" for tidyverse plus TeX Live and publishing-related packages. Defaults to "base".
auto_syslibs	Logical. If TRUE (the default), reads renv.lock from the current working directory, queries the Posit Package Manager sysreqs database via remotes::system_requirements(), and automatically includes the system libraries required by all packages in the lock file. Warns and continues without auto-detection if the lookup fails. Set to FALSE to skip auto-detection entirely.
install_syslibs	A character vector or NULL. Additional system libraries to install beyond those auto-detected from renv.lock. Each element should be a valid apt package name, e.g. c("libuv1-dev", "libwebp-dev"). Defaults to NULL.
output	A character string. Directory path where the Dockerfile will be written. Defaults to tempdir().
data_file	A character string or NULL. Path to a data file to copy into the container. The local directory structure is preserved under /home/ – e.g. "data-raw/sample.csv" becomes /home/data-raw/sample.csv inside the container. The file must be inside the current working directory (the build context). Defaults to NULL.
code_file	A character string or NULL. Path to a script file (e.g. .R, .qmd, .rmd) to copy into the container. The local directory structure is preserved under /home/. The file must be inside the current working directory. Defaults to NULL.
misc_file	A character string or NULL. Path to a miscellaneous file (e.g. an image or shell script) to copy into the container. The local directory structure is preserved under /home/. The file must be inside the current working directory. Defaults to NULL.
add_user	A character string. Name of a Linux user to create inside the container with sudo access. Defaults to NULL.
home_dir	A character string. The working directory set inside the container via WORKDIR. Defaults to "/home".
expose_port	A character string. The port to expose when r_mode is "rstudio". Defaults to "8787". Ignored when r_mode is not "rstudio".
install_quarto	Logical. If TRUE, downloads and installs the Quarto CLI inside the container. Defaults to FALSE.

comments	Logical. If TRUE, annotates each Dockerfile instruction with an explanatory comment. Useful for learning or sharing. Defaults to FALSE.
verbose	Logical. If TRUE, prints progress messages as each section of the Dockerfile is written. Defaults to FALSE.

Value

Called for its side effects. Writes a Dockerfile to output. Returns invisible(NULL).

Prerequisites

generate_dockerfile() requires an renv.lock file in the current working directory. Create one with renv::snapshot() before calling this function. If the lock file is out of sync with your project library, a warning is issued – run renv::snapshot() to update it before building the image.

Examples

```
## Not run:
# Requires renv.lock in the current working directory.
# Run renv::snapshot() first if you don't have one.

# Generate a minimal Dockerfile using a pinned R version
generate_dockerfile(r_version = "4.4.0", output = tempdir())

# Pin a specific R version with the tidyverse image
generate_dockerfile(r_version = "4.3.0", r_mode = "tidyverse",
                    output = tempdir())

# Add extra system libraries on top of auto-detected ones
generate_dockerfile(
  r_version       = "4.4.0",
  install_syslibs = c("libuv1-dev", "libwebp-dev"),
  output          = "."
)

# Include a data file -- directory structure is preserved in the container
generate_dockerfile(
  r_version = "4.3.0",
  data_file = "data-raw/penguins.csv",
  code_file = "analysis.R",
  comments  = TRUE,
  output    = "."
)

## End(Not run)
```

`list_images` *List locally available container images*

Description

`list_images()` returns a data frame of container images currently stored in the local image store, as reported by `podman image ls` or `docker image ls`. This is useful for finding the image ID to pass to `push_image()` after building an image with `build_image()`.

Usage

```
list_images(tool = NULL, verbose = FALSE)
```

Arguments

`tool`	A character string or `NULL`. The container tool to use. One of `"podman"` or `"docker"`. If `NULL` (the default), the function auto-detects which tool is available, preferring `podman`.
`verbose`	Logical. If `TRUE`, prints a progress message before querying the local image store. Defaults to `FALSE`.

Value

A data frame with five columns: `repository`, `tag`, `image_id`, `created`, and `size`. Rows where both `repository` and `tag` are `<none>` correspond to untagged images produced by `build_image()` when no `tag` argument was supplied. The data frame is also printed to the console. Returns an empty data frame if no images are found.

Finding your image ID

After calling `build_image()`, run `list_images()` to find the image ID of the image you just built. Untagged images appear with `<none>` in the `repository` and `tag` columns – the `image_id` column contains the hash you need to pass to `push_image()`:

```
imgs <- list_images()
push_image(
  image_id = imgs$image_id[1],
  netid    = "erwin.lares",
  project  = "container-registry"
)
```

Examples

```
## Not run:
# List all local images
list_images()

# Capture the result for programmatic use
```

```
imgs <- list_images()
imgs$image_id[1]

## End(Not run)
```

---

push_image *Tag and push a container image to a registry*

Description

push_image() tags a locally built container image with a full registry path and pushes it to a container registry. It handles both the podman tag and podman push steps in a single call. Auto-detects which container tool is available unless tool is specified explicitly. Use dry_run = TRUE to preview the exact commands without executing them. The format for the is registry.doit.wisc.edu//:

Usage

```
push_image(
  image_id = NULL,
  netid = NULL,
  project = NULL,
  tag = "latest",
  registry = "registry.doit.wisc.edu",
  tool = NULL,
  check_login = TRUE,
  dry_run = FALSE,
  verbose = FALSE,
  comments = FALSE
)
```

Arguments

image_id	A character string. The local image ID or name to push, as shown in podman image ls or docker image ls. This is typically a 12-character hash (e.g. "974123909a36") or a locally assigned name if the image was built with a tag via build_image().
netid	A character string. Your UW-Madison NetID, used to construct the full registry path, e.g. "erwin.lares".
project	A character string. The GitLab project name that hosts the container registry, e.g. "container-registry".
tag	A character string. The version tag to assign to the image. Defaults to "latest". Using explicit version tags (e.g. "1.0.0") is recommended for reproducibility – "latest" is overwritten on every push.
registry	A character string. The registry hostname. Defaults to "registry.doit.wisc.edu" (UW-Madison CHTC).
tool	A character string or NULL. The container tool to use. One of "podman" or "docker". If NULL (the default), the function auto-detects which tool is available, preferring podman.

check_login	Logical. If TRUE (the default), verifies that you are logged in to registry before attempting the push. If not logged in, the function errors with instructions on how to authenticate.
dry_run	Logical. If TRUE, prints the commands that would be run without executing them. Defaults to FALSE.
verbose	Logical. If TRUE, prints progress messages at each step. Defaults to FALSE.
comments	Logical. If TRUE, prints explanatory context before each step – what the command does, why it is needed, and common pitfalls. Useful for first-time users learning the container push workflow. Defaults to FALSE.

Value

Called for its side effects. Returns invisible(NULL).

Prerequisites

Before calling push_image(), ensure the following are in place:

1. The image has been built locally with build_image(). Run podman image ls to find the image ID.
2. You have a GitLab account at git.doit.wisc.edu and a project with the container registry enabled.
3. You have a Personal Access Token (PAT) with read_registry and write_registry scopes. Create one at: https://git.doit.wisc.edu/-/user_settings/personal_access_tokens
4. You are logged in to the registry. Authenticate once in a terminal: podman login registry.doit.wisc.edu Enter your NetID as the username and your PAT as the password.

Authentication

The GitLab container registry requires authentication before pushing. Use a Personal Access Token (PAT) rather than your NetID password – PATs can be scoped to registry access only and revoked independently. Authentication is cached by podman or docker after the first login, so you only need to run podman login once per machine per session.

Note that GitLab Self-Managed authentication tokens expire after five minutes by default. If you see an unauthorized: authentication required error mid-push on a large image, re-authenticate and push again.

Examples

```
## Not run:
# Tag and push an image to the CHTC registry
push_image(
  image_id = "974123909a36",
  netid    = "erwin.lares",
  project  = "container-registry"
)

# Push with an explicit version tag
```

```
push_image(
  image_id = "974123909a36",
  netid    = "erwin.lares",
  project  = "container-registry",
  tag      = "1.0.0"
)

# Preview the commands without running them
push_image(
  image_id = "974123909a36",
  netid    = "erwin.lares",
  project  = "container-registry",
  dry_run  = TRUE
)

# Guided push for first-time users
push_image(
  image_id = "974123909a36",
  netid    = "erwin.lares",
  project  = "container-registry",
  verbose  = TRUE,
  comments = TRUE
)

## End(Not run)
```

Index