

Package ‘tugboat’

January 20, 2026

Title Build a Docker Image from a Directory or Project

Version 0.1.6

Description Simple utilities to generate a Dockerfile from a directory or project, build the corresponding Docker image, push the image to DockerHub, and publicly share the project via Binder.

Imports here, renv (>= 1.0.0)

Suggests gert, usethis, yaml, rmarkdown, testthat (>= 3.0.0)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://www.dmolitor.com/tugboat/>

Config/testthat/edition 3

Config/testthat/start-first create

Language en-US

NeedsCompilation no

Author Daniel Molitor [aut, cph, cre]

Maintainer Daniel Molitor <molitdj97@gmail.com>

Repository CRAN

Date/Publication 2026-01-20 08:30:02 UTC

Contents

binderize	2
build	3
create	5

Index

7

binderize	<i>Prepare project for Binder</i>
-----------	-----------------------------------

Description

The `binderize()` function converts an existing tugboat project into a [Binder](#)-compatible project by creating a Dockerfile that launches RStudio Server via the `rocker/binder` base image. Optionally, it can add a Binder launch badge to the project's README.

Usage

```
binderize(  
  dockerfile = here::here("Dockerfile"),  
  branch = "main",  
  hub = "mybinder.org",  
  urlpath = "rstudio",  
  add_readme_badge = TRUE,  
  overwrite = TRUE  
)
```

Arguments

<code>dockerfile</code>	Path to the tugboat-generated Dockerfile.
<code>branch</code>	Character string specifying the Git branch, tag, or commit hash to build. Defaults to "main".
<code>hub</code>	The Binder hub to use. Currently only "mybinder.org" is supported.
<code>urlpath</code>	The URL path to open inside the Binder instance. Defaults to "rstudio", which opens an RStudio Server session.
<code>add_readme_badge</code>	Logical. Whether to add a Binder launch badge to the README. Defaults to TRUE.
<code>overwrite</code>	Logical. Whether to overwrite an existing Binder Dockerfile. Defaults to TRUE.

Details

This enables one-click, cloud-based execution of your R analysis environment directly from GitHub using Binder.

Currently only GitHub repositories are supported. If `add_readme_badge = TRUE`, a Binder badge will be appended to the README file, linking to the live Binder instance.

Value

Invisibly returns `NULL`. Called primarily for its side effects of creating Binder-related files and optionally committing them.

Note

Binder can only build from the remote GitHub repository. The `.binder/Dockerfile` and `README` changes must be committed and pushed before launching Binder; otherwise, the build will not reflect local modifications.

See Also

- [create\(\)](#) — Generates a Dockerfile from an analysis directory.
- [build\(\)](#) — Builds the corresponding Docker image locally.

Examples

```
## Not run:  
binderize(  
  dockerfile = here::here("Dockerfile"),  
  branch = "main",  
  add_readme_badge = TRUE  
)  
  
## End(Not run)
```

build*Build a Docker image*

Description

A simple utility to quickly build a Docker image from a Dockerfile.

Usage

```
build(  
  dockerfile = here::here("Dockerfile"),  
  image_name = "tugboat",  
  tag = "latest",  
  platforms = c("linux/amd64", "linux/arm64"),  
  build_args = NULL,  
  build_context = here::here(),  
  push = FALSE,  
  dh_username = NULL,  
  dh_password = NULL,  
  verbose = FALSE  
)
```

Arguments

<code>dockerfile</code>	The path to the Dockerfile. The default value is a file named <code>Dockerfile</code> in the project directory surfaced by <code>here::here</code> .
<code>image_name</code>	A string specifying the Docker image name. Default is <code>tugboat</code> .
<code>tag</code>	A string specifying the image tag. Default is <code>latest</code> .
<code>platforms</code>	A vector of strings. Which platforms to build images for. Default is both <code>linux/amd64</code> and <code>linux/arm64</code> .
<code>build_args</code>	A vector of strings specifying additional build arguments to pass to the <code>docker buildx build</code> command. Optional.
<code>build_context</code>	The directory that is the build context for the image(s). Default value is the directory returned by <code>here::here</code> .
<code>push</code>	A boolean indicating whether to push to DockerHub.
<code>dh_username</code>	A string specifying the DockerHub username. Only necessary if <code>push == TRUE</code> .
<code>dh_password</code>	A string specifying the DockerHub password. Only necessary if <code>push == TRUE</code> .
<code>verbose</code>	A boolean. Whether to print the actual Docker build command or not. Defaults to <code>FALSE</code> .

Value

The name of the built Docker image as a string.

Examples

```
## Not run:
dock <- create(
  project = here::here(),
  FROM = "rstudio/r-base:devel-bookworm",
  exclude = c("/data", "/examples")
)

image_name <- build(
  dockerfile = here::here("Dockerfile"),
  image_name = "awesome_analysis",
  push = TRUE,
  dh_username = Sys.getenv("DH_USERNAME"),
  dh_password = Sys.getenv("DH_PASSWORD")
)

## End(Not run)
```

`create`*Create a Dockerfile*

Description

This function will crawl all files in the current project/directory and (attempt to) detect all R packages and store these in a lockfile. From this lockfile, it will create a corresponding Dockerfile. It will also copy the full contents of the current directory/project into the Docker image. The directory in the Docker container containing the current directory contents will be /current-directory-name. For example if your analysis directory is named `incredible_analysis`, the corresponding location in the generated Docker image will be `/incredible_analysis`.

Usage

```
create(  
  project = here::here(),  
  as = file.path(project, "Dockerfile"),  
  FROM = NULL,  
  ...,  
  exclude = NULL,  
  verbose = FALSE,  
  optimize_pak = TRUE  
)
```

Arguments

<code>project</code>	The project directory. If no project directory is provided, by default, the <code>here</code> package will be used to determine the active project. If no project is currently active, then <code>here</code> defaults to the working directory where initially called.
<code>as</code>	The file path to write to. The default value is <code>file.path(project, "Dockerfile")</code> .
<code>FROM</code>	Docker image to start <code>FROM</code> . Default is <code>FROM r-base:R.version</code> .
<code>...</code>	Additional arguments which are passed directly to <code>renv::snapshot</code> . Please see the documentation for that function for all relevant details.
<code>exclude</code>	A vector of strings specifying all paths (files or directories) that should NOT be included in the Docker image. By default, all files in the directory will be included. NOTE: the file and directory paths should be relative to the project directory. They do NOT need to be absolute paths.
<code>verbose</code>	A boolean indicating whether or not to print the resulting Dockerfile to the console. Default value is <code>FALSE</code> .
<code>optimize_pak</code>	A boolean indicating whether or not to try to optimize package installations with <code>pak</code> . Defaults to <code>TRUE</code> . This should rarely be changed from its default value. However, sometimes this optimization may cause build failures. When encountering a build error, a good first step can be to set <code>optimize_pak = FALSE</code> and see if the error persists.

Value

The Dockerfile contained as a string vector. Each vector element corresponds to a line in the Dockerfile.

See Also

[here::here](#); this will be used by default to determine the current project directory.

[renv::snapshot](#) which this function relies on to find all R dependencies and create a corresponding lockfile.

Examples

```
## Not run:  
# Create a Dockerfile based on the rocker/rstudio image.  
# Write the Dockerfile locally to here::here("Dockerfile").  
# Copy all files except the /data and /examples directories.  
dock <- create(  
  project = here::here(),  
  FROM = "rocker/rstudio",  
  exclude = c("/data", "/examples")  
)  
  
## End(Not run)
```

Index

`binderize`, [2](#)

`build`, [3](#)

`build()`, [3](#)

`create`, [5](#)

`create()`, [3](#)

`here::here`, [4](#), [6](#)

`renv::snapshot`, [5](#), [6](#)