

Package ‘shiny2docker’

June 28, 2025

Title Generate Dockerfiles for 'Shiny' Applications

Version 0.0.3

Description Automates the creation of Dockerfiles for deploying 'Shiny' applications.
By integrating with 'renv' for dependency management and leveraging Docker-based solutions, it simplifies the process of containerizing 'Shiny' apps, ensuring reproducibility and consistency across different environments. Additionally, it facilitates the setup of CI/CD pipelines for building Docker images on both GitLab and GitHub.

License MIT + file LICENSE

Imports attachment (>= 0.4.3), cli, dockerfiler, here, yesno

Encoding UTF-8

RoxygenNote 7.3.2

URL <https://github.com/VincentGuyader/shiny2docker>

BugReports <https://github.com/VincentGuyader/shiny2docker/issues>

Suggests shiny, renv, testthat (>= 3.0.0), knitr, rmarkdown, rstudioapi

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Vincent Guyader [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-0671-9270>>)

Maintainer Vincent Guyader <vincent@thinkr.fr>

Repository CRAN

Date/Publication 2025-06-28 21:10:02 UTC

Contents

set_github_action	2
set_gitlab_ci	2
shiny2docker	3

Index	6
--------------	----------

set_github_action	<i>Configure GitHub Action pipeline for Docker builds</i>
-------------------	---

Description

Copies the docker-build.yml file provided by the shiny2docker package into the .github/workflows/ directory within the specified base directory. This GitHub Action configuration is designed to build a Docker image and push the created image to the GitHub Container Registry.

Usage

```
set_github_action(path)
```

Arguments

path	A character string specifying the base directory where the .github/workflows/ folder will be created and the docker-build.yml file copied. If missing, the user will be prompted to use the current directory.
------	--

Value

A logical value indicating whether the file was successfully copied (TRUE) or not (FALSE).

Examples

```
# Copy the docker-build.yml file to the .github/workflows/ directory in a temporary folder
set_github_action(path = tempdir())
```

set_gitlab_ci	<i>Configure GitLab CI pipeline for Docker builds</i>
---------------	---

Description

Copies the .gitlab-ci.yml file provided by the shiny2docker package into the specified directory. The GitLab CI configuration is designed to build a Docker image and push the created image to the GitLab container registry.

Usage

```
set_gitlab_ci(path, tags = NULL)
```

Arguments

path	A character string specifying the directory where the .gitlab-ci.yml file will be copied. If missing, the user will be prompted to use the current directory.
tags	Optional character vector of GitLab runner tags. If provided, the function will add these tags to the generated CI job so the appropriate runner is selected. You can provide multiple tags.

Value

A logical value indicating whether the file was successfully copied (TRUE) or not (FALSE).

Examples

```
# Copy the .gitlab-ci.yml file to a temporary directory
set_gitlab_ci(path = tempdir())

# Copy the file and specify runner tags
set_gitlab_ci(path = tempdir(), tags = c("shiny_build", "prod"))
```

shiny2docker

*shiny2docker***Description**

Generate a Dockerfile for a Shiny Application

Usage

```
shiny2docker(
  path = ".",
  lockfile = file.path(path, "renv.lock"),
  output = file.path(path, "Dockerfile"),
  FROM = "rocker/geospatial",
  AS = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  extra_sysreqs = NULL,
  use_pak = FALSE,
  user = NULL,
  dependencies = NA,
  sysreqs_platform = "ubuntu",
  folder_to_exclude = c("renv")
)
```

Arguments

<code>path</code>	Character. Path to the folder containing the Shiny application (e.g., <code>app.R</code> or <code>ui.R</code> and <code>server.R</code>) along with any other necessary files.
<code>lockfile</code>	Character. Path to the <code>renv.lock</code> file that specifies the R package dependencies. If the <code>renv.lock</code> file does not exist, it will be created for production using the <code>attachment::create_renv_for_prod</code> function.
<code>output</code>	Character. Path to the generated Dockerfile. Defaults to <code>"Dockerfile"</code> .
<code>FROM</code>	Docker image to start FROM. Default is <code>rocker/geospatial</code>

AS	The AS of the Dockerfile. Default it NULL.
sysreqs	boolean. If TRUE, the Dockerfile will contain sysreq installation.
repos	character. The URL(s) of the repositories to use for options("repos").
expand	boolean. If TRUE each system requirement will have its own RUN line.
extra_sysreqs	character vector. Extra debian system requirements. Will be installed with apt-get install.
use_pak	boolean. If TRUE use pak to deal with dependencies during <code>renv::restore()</code> . FALSE by default
user	Name of the user to specify in the Dockerfile with the USER instruction. Default is NULL, in which case the user from the FROM image is used.
dependencies	What kinds of dependencies to install. Most commonly one of the following values: <ul style="list-style-type: none"> • NA: only required (hard) dependencies, • TRUE: required dependencies plus optional and development dependencies, • FALSE: do not install any dependencies. (You might end up with a non-working package, and/or the installation might fail.)
sysreqs_platform	System requirements platform.ubuntu by default. If NULL, then the current platform is used. Can be : "ubuntu-22.04" if needed to fit with the FROM Operating System. Only debian or ubuntu based images are supported
folder_to_exclude	Folder to exclude during scan to detect packages

Details

Automate the creation of a Dockerfile tailored for deploying Shiny applications. It manages R dependencies using `renv`, generates a `.dockerignore` file to optimize the Docker build process, and leverages the `dockerfiler` package to allow further customization of the Dockerfile object before writing it to disk.

Value

An object of class `dockerfiler`, representing the generated Dockerfile. This object can be further manipulated using `dockerfiler` functions before being written to disk.

Examples

```
temp_dir <- tempfile("shiny2docker_example_")
dir.create(temp_dir)
example_app <- system.file("dummy_app", package = "shiny2docker")
file.copy(example_app, temp_dir, recursive = TRUE)

app_path <- file.path(temp_dir, "dummy_app")
if (requireNamespace("rstudioapi", quietly = TRUE) &&
    rstudioapi::isAvailable()) {
  rstudioapi::filesPaneNavigate(app_path)
}
```

```
docker_obj <- shiny2docker::shiny2docker(path = app_path)

print(list.files(app_path, all.files = TRUE, no.. = TRUE))

# Further manipulate the Dockerfile object
docker_obj$add_after(
  cmd = "ENV ENV \'MY_ENV_VAR\'=\'value\'",
  after = 3
)
docker_obj$write(file.path(app_path, "Dockerfile"))
```

Index

`set_github_action`, [2](#)
`set_gitlab_ci`, [2](#)
`shiny2docker`, [3](#)