Package 'liftr'

October 13, 2022

Type Package

Title Containerize R Markdown Documents for Continuous Reproducibility

Version 0.9.2

Maintainer Nan Xiao <me@nanx.me>

Description Persistent reproducible reporting by containerization of R Markdown documents.

License GPL-3 | file LICENSE

SystemRequirements Docker (see <https://docs.docker.com/install/>)

VignetteBuilder knitr

URL https://nanx.me/liftr/, https://github.com/nanxstats/liftr

BugReports https://github.com/nanxstats/liftr/issues

Depends R (>= 3.0.2)

Imports yaml, knitr, rmarkdown, stringr, rstudioapi

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Author Nan Xiao [aut, cre] (<https://orcid.org/0000-0002-0250-5673>),
Miao-Zhu Li [ctb],
Teng-Fei Yin [ctb]

Repository CRAN

Date/Publication 2019-06-19 04:20:03 UTC

R topics documented:

check_docker_install 2
check_docker_running 2
install_docker 3
lift 3
prune_all_auto 4
prune_container 5

prune_container_auto 6
prune_image 6
prune_image_auto 7
render_docker 7

Index **10**

`check_docker_install` *Check if Docker was Installed*

Description

This function checks if Docker was properly installed and discoverable by R and liftr. If still not usable, please start Docker daemon

Usage

```
check_docker_install()
```

Value

`TRUE` if Docker was deteted, `FALSE` otherwise.

Examples

```
## Not run:
check_docker_install()
## End(Not run)
```

`check_docker_running` *Check if Docker Daemon is Running*

Description

This function checks if the Docker daemon is running.

Usage

```
check_docker_running()
```

Value

`TRUE` if Docker daemon is running, `FALSE` otherwise.

Examples

```
## Not run:
check_docker_running()
## End(Not run)
```

install_docker *Installation Helper for Docker Engine*

Description

This function guides you to install Docker (Engine).

Usage

```
install_docker()
```

References

https://docs.docker.com/engine/installation/

Examples

```
## Not run:
install_docker()
## End(Not run)
```

lift *Containerize R Markdown Documents*

Description

Containerize R Markdown documents. This function generates `Dockerfile` based on the liftr metadata in the RMD document.

Usage

```
lift(input = NULL, use_config = FALSE, config_file = "_liftr.yml",
  output_dir = NULL)
```

Arguments

`input`	Input (R Markdown) file.
`use_config`	If `TRUE`, will parse the liftr metadata from a YAML file, if `FALSE`, will parse such information from the metadata section in the R Markdown file. Default is `FALSE`.
`config_file`	Name of the YAML configuration file, under the same directory as the input file. Default is `"_liftr.yml"`.
`output_dir`	Directory to output `Dockerfile`. If not provided, will be the same directory as `input`.

Details

After running lift, run render_docker on the document to render the containerized R Markdown document using Docker containers. See `vignette('liftr-intro')` for details about the extended YAML front-matter metadata format used by liftr.

Value

`Dockerfile`.

Examples

```
# copy example file
dir_example = paste0(tempdir(), '/liftr-minimal/')
dir.create(dir_example)
file.copy(system.file("examples/liftr-minimal.Rmd", package = "liftr"), dir_example)

# containerization
input = paste0(dir_example, "liftr-minimal.Rmd")
lift(input)

## Not run:
# render the document with Docker
render_docker(input)

# view rendered document
browseURL(paste0(dir_example, "liftr-minimal.html"))

# purge the generated Docker image
purge_image(paste0(dir_example, "liftr-minimal.docker.yml"))
## End(Not run)
```

`prune_all_auto` *Remove Docker Containers, Images, and Networks*

Description

This function removes stopped containers, all networks not used by at least one container, all dangling images, and all build cache.

Usage

```
prune_all_auto(volumes = FALSE)
```

Arguments

`volumes`	Logical. Should we prune volumes? Default is `FALSE`.

Value

prune results

References

https://docs.docker.com/engine/admin/pruning/

Examples

```
## Not run:
prune_all_auto()
## End(Not run)
```

`prune_container` *Remove Specific Docker Containers*

Description

This function stops and removes the Docker containers used for rendering the R Markdown document based on the output YAML file from the (possibly unsuccessful) rendering process.

Usage

```
prune_container(input_yml)

purge_container()
```

Arguments

`input_yml`	The YAML file path (output of `render_docker`) when `prune_info = TRUE` which stores the information of the Docker container to be stopped and removed. If not specified, will prune all dangling containers.

Value

prune results

Examples

```
## Not run:
prune_container()
## End(Not run)
```

prune_container_auto *Remove Dangling Docker Containers*

Description

This function prunes all dangling Docker containers automatically.

Usage

```
prune_container_auto()
```

Value

prune results

References

https://docs.docker.com/engine/admin/pruning/

Examples

```
## Not run:
prune_container_auto()
## End(Not run)
```

prune_image *Remove Specific Docker Images*

Description

This function removes the Docker images used for rendering the R Markdown document based on the output YAML file from the (possibly unsuccessful) rendering process.

Usage

```
prune_image(input_yml)

purge_image()
```

Arguments

input_yml	The YAML file path (output of render_docker) when prune_info = TRUE which stores the information of the Docker image to be removed. If not specified, will prune all dangling images.

Value

prune results

Examples

```
## Not run:
prune_image()
## End(Not run)
```

---

prune_image_auto *Remove Dangling Docker Images*

Description

This function prunes all dangling Docker images automatically.

Usage

```
prune_image_auto()
```

Value

prune results

References

https://docs.docker.com/engine/admin/pruning/

Examples

```
## Not run:
prune_image_auto()
## End(Not run)
```

---

render_docker *Render Containerized R Markdown Documents*

Description

Render R Markdown documents using Docker.

Usage

```
render_docker(input = NULL, tag = NULL, container_name = NULL,
  cache = TRUE, build_args = NULL, run_args = NULL, prune = TRUE,
  prune_info = TRUE, dry_run = FALSE, ...)

drender(...)
```

Arguments

input	Input file to render in Docker container.
tag	Docker image name to build, sent as docker argument -t. If not specified, it will use the same name as the input file.
container_name	Docker container name to run. If not specified, will use a randomly generated name.
cache	Logical. Controls the --no-cache argument in docker run. Setting this to be TRUE can accelerate the rendering speed substantially for repeated/interactive rendering since the Docker image layers will be cached, with only the changed (knitr related) image layer being updated. Default is TRUE.
build_args	A character string specifying additional docker build arguments. For example, --pull=true -m="1024m" --memory-swap="-1".
run_args	A character string specifying additional docker run arguments. For example, --privileged=true.
prune	Logical. Should we clean up all dangling containers, volumes, networks, and images in case the rendering was not successful? Default is TRUE.
prune_info	Logical. Should we save the Docker container and image information to a YAML file (name ended with .docker.yml) for manual pruning or inspections later? Default is TRUE.
dry_run	Preview the Docker commands but do not run them? Useful for debugging purposes. Default is FALSE.
...	Additional arguments passed to render.

Details

Before using this function, please run lift on the RMD document first to generate the Dockerfile.

After a successful rendering, you will be able to clean up the Docker image with prune_image.

Please see vignette('liftr-intro') for details of the extended YAML metadata format and system requirements for writing and rendering containerized R Markdown documents.

Value

- A list containing the image name, container name, and Docker commands will be returned.
- An YAML file ending with .docker.yml storing the image name, container name, and Docker commands for rendering this document will be written to the directory of the input file.
- The rendered output will be written to the directory of the input file.

Examples

```
# copy example file
dir_example = paste0(tempdir(), "/liftr-tidyverse/")
dir.create(dir_example)
file.copy(system.file("examples/liftr-tidyverse.Rmd", package = "liftr"), dir_example)

# containerization
```

```
input = paste0(dir_example, "liftr-tidyverse.Rmd")
lift(input)

## Not run:
# print the Docker commands first
render_docker(input, dry_run = TRUE)

# render the document with Docker
render_docker(input)

# view rendered document
browseURL(paste0(dir_example, "liftr-tidyverse.pdf"))

# remove the generated Docker image
prune_image(paste0(dir_example, "liftr-tidyverse.docker.yml"))
## End(Not run)
```

Index

check_docker_install, 2
check_docker_running, 2

drender *(*render_docker*)*, 7

install_docker, 3

lift, 3, *4*, *8*

prune_all_auto, 4
prune_container, 5
prune_container_auto, 6
prune_image, 6, *8*
prune_image_auto, 7
purge_container *(*prune_container*)*, 5
purge_image *(*prune_image*)*, 6

render, *8*
render_docker, *4–6*, 7