# Package 'dockerfiler'

September 3, 2021

**Title** Easy Dockerfile Creation from R

**Version** 0.1.4

**Description** Build a Dockerfile straight from your R session.
'dockerfiler' allows you to create step by step a Dockerfile, and
provide convenient tools to wrap R code inside this Dockerfile.

**License** MIT + file LICENSE

**URL** https://github.com/ColinFay/dockerfiler

**BugReports** https://github.com/ColinFay/dockerfiler/issues

**Imports** attempt (>= 0.3.1), glue, R6, utils, fs, cli, remotes, desc,
usethis, jsonlite, pkgbuild

**Suggests** covr (>= 3.5.1), knitr (>= 1.31), rmarkdown (>= 2.6),
testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Colin Fay [cre, aut] (<https://orcid.org/0000-0001-7343-1846>),
Vincent Guyader [aut] (<https://orcid.org/0000-0003-0671-9270>)

**Maintainer** Colin Fay <contact@colinfay.me>

**Repository** CRAN

**Date/Publication** 2021-09-03 12:40:06 UTC

## R topics documented:

`Dockerfile` *Build a Dockerfile*

## Description

Build a Dockerfile

Build a Dockerfile

## Details

`Dockerfile$new()` will create an object that can be used to build a Dockerfile using a series of commands. Once the Dockerfile is completed, you can save it to the disk.

## Value

A dockerfile template

## Public fields

`Dockerfile` The content of the Dockerfile

## Methods

### Public methods:

- `Dockerfile$new()`
- `Dockerfile$RUN()`
- `Dockerfile$ADD()`
- `Dockerfile$COPY()`
- `Dockerfile$WORKDIR()`
- `Dockerfile$EXPOSE()`
- `Dockerfile$VOLUME()`
- `Dockerfile$CMD()`
- `Dockerfile$LABEL()`
- `Dockerfile$ENV()`
- `Dockerfile$ENTRYPOINT()`
- `Dockerfile$USER()`
- `Dockerfile$ARG()`
- `Dockerfile$ONBUILD()`
- `Dockerfile$STOPSIGNAL()`
- `Dockerfile$HEALTHCHECK()`
- `Dockerfile$SHELL()`
- `Dockerfile$MAINTAINER()`
- `Dockerfile$custom()`
- `Dockerfile$print()`

- Dockerfile$write()
- Dockerfile$switch_cmd()
- Dockerfile$remove_cmd()
- Dockerfile$add_after()
- Dockerfile$clone()

**Method** `new()`**:** Initialize a Dockerfile

*Usage:*

```
Dockerfile$new(FROM = "rocker/r-base", AS = NULL)
```

*Arguments:*

`FROM` base image

`AS` of the image

**Method** `RUN()`**:** Add a RUN command

*Usage:*

```
Dockerfile$RUN(cmd)
```

*Arguments:*

`cmd` The command to add

**Method** `ADD()`**:** Add ADD command

*Usage:*

```
Dockerfile$ADD(from, to, force = TRUE)
```

*Arguments:*

`from` Base file

`to` Dest file

`force` Boolean, should the ADD be forced?

**Method** `COPY()`**:** Add a COPY command

*Usage:*

```
Dockerfile$COPY(from, to, force = TRUE)
```

*Arguments:*

`from` Base file

`to` Dest file

`force` Boolean, should the ADD be forced?

**Method** `WORKDIR()`**:** Add a WORKDIR command

*Usage:*

```
Dockerfile$WORKDIR(where)
```

*Arguments:*

`where` Location of the WORKDIR

**Method** `EXPOSE()`**:** Add a EXPOSE command

*Usage:*

```
Dockerfile$EXPOSE(port)
```

*Arguments:*

`port` The port to expose.

**Method** `VOLUME()`**:** Add a VOLUME command

*Usage:*

```
Dockerfile$VOLUME(volume)
```

*Arguments:*

`volume` The volume to add.

**Method** `CMD()`**:** Add a CMD command

*Usage:*

```
Dockerfile$CMD(cmd)
```

*Arguments:*

`cmd` The command to add

**Method** `LABEL()`**:** Add a LABEL command

*Usage:*

```
Dockerfile$LABEL(key, value)
```

*Arguments:*

`key, value` The key value pair of the Label.

**Method** `ENV()`**:** Add a ENV command

*Usage:*

```
Dockerfile$ENV(key, value)
```

*Arguments:*

`key, value` The key value pair of the Env.

**Method** `ENTRYPOINT()`**:** Add a ENTRYPOINT command

*Usage:*

```
Dockerfile$ENTRYPOINT(cmd)
```

*Arguments:*

`cmd` The command to launch as an entrypoint.

**Method** `USER()`**:** Add a USER command

*Usage:*

```
Dockerfile$USER(user)
```

*Arguments:*

`user` The user name

**Method** `ARG()`**:** Add a ARG command.

*Usage:*

```
Dockerfile$ARG(arg, ahead = FALSE)
```

*Arguments:*

`arg` The ARG to add.

`ahead` Should the arg be put ahead of the Dockerfile?

**Method** `ONBUILD()`**:** Add a ONBUILD command

*Usage:*

```
Dockerfile$ONBUILD(cmd)
```

*Arguments:*

`cmd` The command to launch onbuild.

**Method** `STOPSIGNAL()`**:** Add a STOPSIGNAL command

*Usage:*

```
Dockerfile$STOPSIGNAL(signal)
```

*Arguments:*

`signal` The signal.

**Method** `HEALTHCHECK()`**:** Add a HEALTHCHECK command

*Usage:*

```
Dockerfile$HEALTHCHECK(check)
```

*Arguments:*

`check` The check.

**Method** `SHELL()`**:** Add a SHELL command

*Usage:*

```
Dockerfile$SHELL(shell)
```

*Arguments:*

`shell` The shell.

**Method** `MAINTAINER()`**:** Add a MAINTAINER command

*Usage:*

```
Dockerfile$MAINTAINER(name, email)
```

*Arguments:*

`name, email` The maintainer mail and email.

**Method** `custom()`**:** Add a custom command (you need to provide the verb)

*Usage:*

```
Dockerfile$custom(base, cmd)
```

*Arguments:*

`base` The verb.

`cmd` The content of the command.

**Method** `print()`**:** Print the Dockerfile.

*Usage:*

```
Dockerfile$print()
```

**Method** `write()`**:** Print the Dockerfile.

*Usage:*

```
Dockerfile$write(as = "Dockerfile")
```

*Arguments:*

`as` The full path of the Dockerfile.

**Method** `switch_cmd()`**:** Switch two lines.

*Usage:*

```
Dockerfile$switch_cmd(a, b)
```

*Arguments:*

`a, b` The two lines to switch.

**Method** `remove_cmd()`**:** Remove a line.

*Usage:*

```
Dockerfile$remove_cmd(where)
```

*Arguments:*

`where` The line number.

**Method** `add_after()`**:** Add a cmd after a given line.

*Usage:*

```
Dockerfile$add_after(cmd, after)
```

*Arguments:*

`cmd` The cmd to add.

`after` The line number.

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
Dockerfile$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
my_dock <- Dockerfile$new()
```

## docker_ignore_add *Create a dockerignore file*

### Description

Create a dockerignore file

### Usage

```
docker_ignore_add(path)
```

### Arguments

| | |
|---|---|
| path | Where to write the file |

### Value

The path to the `.dockerignore` file, invisibly.

### Examples

```
if (interactive()) {
  docker_ignore_add()
}
```

## dock_from_desc *Create a Dockerfile from a DESCRIPTION*

### Description

Create a Dockerfile from a DESCRIPTION

### Usage

```
dock_from_desc(
  path = "DESCRIPTION",
  FROM = paste0("rocker/r-ver:", R.Version()$major, ".", R.Version()$minor),
  AS = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  update_tar_gz = TRUE,
  build_from_source = TRUE,
  extra_sysreqs = NULL
)
```

### Arguments

| | |
|---|---|
| path | path to the DESCRIPTION file to use as an input. |
| FROM | The FROM of the Dockerfile. Default is FROM rocker/r-ver:R.Version()$major.R.Version()$minor. |
| 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. |
| update_tar_gz | boolean. If TRUE and build_from_source is also TRUE, an updated tar.gz is created. |
| build_from_source | boolean. If TRUE no tar.gz is created and the Dockerfile directly mount the source folder. |
| extra_sysreqs | character vector. Extra debian system requirements. Will be installed with apt-get install. |

---

## get_sysreqs *Get system requirements*

---

### Description

This function retrieves information about the system requirements using the https://sysreqs.r-hub.io API.

### Usage

```
get_sysreqs(packages, quiet = TRUE, batch_n = 30)
```

### Arguments

| | |
|---|---|
| packages | character vector. Packages names. |
| quiet | Boolean. If TRUE the function is quiet. |
| batch_n | numeric. Number of simultaneous packages to ask. |

### Value

A vector of system requirements.

r *Turn an R call into an Unix call*

## Description

Turn an R call into an Unix call

## Usage

```
r(code)
```

## Arguments

| | |
|---|---|
| code | the function to call |

## Value

an unix R call

## Examples

```
r(print("yeay"))
r(install.packages("plumber", repo = "http://cran.irsn.fr/"))
```

# Index