Package 'babelwhale'

October 3, 2019

Version 1.0.1

Title Talking to 'Docker' and 'Singularity' Containers

Description Provides a unified interface to interact with 'docker' and 'singularity' containers. You can execute a command inside a container, mount a volume or copy a file.

URL https://github.com/dynverse/babelwhale

BugReports https://github.com/dynverse/babelwhale/issues

License GPL-3

Depends R (>= 3.0.0)

Imports crayon, dplyr, dynutils, processx, purrr, utils

SystemRequirements Docker and/or Singularity (>=3.0)

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.1.1

Suggests testthat

NeedsCompilation no

Author Robrecht Cannoodt [aut] (<https://orcid.org/0000-0003-3641-729X>, rcannood),
Wouter Saelens [aut, cre] (<https://orcid.org/0000-0002-7114-6248>, zouter)

Maintainer Wouter Saelens <wouter.saelens@gmail.com>

Repository CRAN

Date/Publication 2019-10-03 13:20:03 UTC

R topics documented:

`babelwhale` *Talking to both Docker and Singularity containers from R*

Description

Talking to both Docker and Singularity containers from R

`copy_file` *Copy a file from a container to the host system*

Description

Copy a file from a container to the host system

Usage

```
copy_file(container_id, path_container, path_local)
```

Arguments

`container_id`	The name of the container, usually the repository name on dockerhub.
`path_container`	The path of the file inside the container
`path_local`	The path of the file on the host system

Examples

```
if (test_docker_installation()) {
  set_default_config(create_docker_config(), permanent = FALSE)
  copy_file("alpine", "/bin/date", tempfile())
}
```

create_config *Backend configuration for containerisation*

Description

It is advised to define the "BABELWHALE_BACKEND" environment variable as "docker" or "singularity".

When using singularity, also define the "SINGULARITY_CACHEDIR" environment variable, which is the folder where the singularity images will be cached. Each TI method will require about 1GB of space.

Alternatively, you can create a config and save it using set_default_config().

Usage

```
create_config(
  backend =
    get_env_or_null("BABELWHALE_BACKEND") %||%
    detect_backend()
)

create_docker_config()

create_singularity_config(
  cache_dir =
    get_env_or_null("SINGULARITY_CACHEDIR") %||%
    ".singularity/"
)

get_default_config()

set_default_config(config, permanent = TRUE)
```

Arguments

backend	Which backend to use. Can be either "docker" or "singularity".
cache_dir	A folder in which to store the singularity images. A container typically requires 100MB to 2GB.
config	A config to save as default.
permanent	Whether or not to save the config file permanently

Examples

```
config <- create_docker_config()
set_default_config(config, permanent = FALSE)

config <- create_singularity_config(
  # ideally, this would be set to a non-temporary directory
```

```
  cache_dir = tempdir()
)
set_default_config(config, permanent = FALSE)
```

`list_docker_images` *List docker containers*

Description

List docker containers

Usage

```
list_docker_images(container_id = NULL)
```

Arguments

`container_id`	An optional container id

Examples

```
if (test_docker_installation()) {
  set_default_config(create_docker_config(), permanent = FALSE)
  list_docker_images()
}
```

`pull_container` *Pull a container from dockerhub*

Description

Pull a container from dockerhub

Usage

```
pull_container(container_id)
```

Arguments

`container_id`	The name of the container, usually the repository name on dockerhub.

Examples

```
if (test_docker_installation()) {
  pull_container("alpine")
}
```

read_file *Read a file from a container*

Description

Read a file from a container

Usage

```
read_file(container_id, path_container)
```

Arguments

container_id	The name of the container, usually the repository name on dockerhub.
path_container	The path of the file inside the container

Examples

```
if (test_docker_installation()) {
  set_default_config(create_docker_config(), permanent = FALSE)
  read_file("alpine", "/etc/hosts")
}
```

run *Run a containerised command, and wait until finished*

Description

Run a containerised command, and wait until finished

Usage

```
run(container_id, command, args = NULL, volumes = NULL,
  workspace = NULL, environment_variables = NULL, debug = FALSE,
  verbose = FALSE)
```

Arguments

container_id	The name of the container, usually the repository name on dockerhub.
command	Character scalar, the command to run.
args	Character vector, arguments to the command.
volumes	Which volumes to be mounted. Format: a character vector, with each element containing the source path and container path concatenated with a ":". For example: `c("/source_folder:/container_folder")`.

workspace	Which working directory to run the command in.
environment_variables	A character vector of environment variables. Format: c("ENVVAR=VALUE").
debug	If TRUE, a command will be printed that the user can execute to enter the container.
verbose	Whether or not to print output

Examples

```
if (test_docker_installation()) {
  set_default_config(create_docker_config(), permanent = FALSE)

  # running a command
  run("alpine", "echo", c("hello"))

  # mounting a folder
  folder <- tempdir()
  write("i'm a mounted file", paste0(folder, "/file.txt"))
 run("alpine", "cat", c("/mounted_folder/file.txt"), volumes = paste0(folder, "/:/mounted_folder"))
}
```

---

singularity_image_path

Determine the cached path of singularity images

Description

Determine the cached path of singularity images

Usage

```
singularity_image_path(container_id)
```

Arguments

container_id	The name of the container, usually the repository name on dockerhub.

test_docker_installation

Tests whether docker is correctly installed and available

Description

Tests whether docker is correctly installed and available

Usage

```
test_docker_installation(detailed = FALSE)
```

Arguments

detailed	Whether top do a detailed check

Examples

```
test_docker_installation()

if (test_docker_installation()) {
  test_docker_installation(detailed = TRUE)
}
```

test_singularity_installation

Tests whether singularity is correctly installed and available

Description

Tests whether singularity is correctly installed and available

Usage

```
test_singularity_installation(detailed = FALSE)
```

Arguments

detailed	Whether top do a detailed check

Examples

```
test_singularity_installation()

if (test_singularity_installation()) {
  test_singularity_installation(detailed = TRUE)
}
```

Index