Package: pracpac (via r-universe)

September 12, 2024

Title Practical 'R' Packaging in 'Docker'

Version 0.2.0

Description Streamline the creation of 'Docker' images with 'R' packages and dependencies embedded. The 'pracpac' package provides a 'usethis'-like interface to creating Dockerfiles with dependencies managed by 'renv'. The 'pracpac' functionality is described in Nagraj and Turner (2023) <doi:10.48550/arXiv.2303.07876>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports magrittr, glue, fs, rprojroot, renv, pkgbuild

Depends R (>= 2.10)

Suggests rmarkdown, knitr, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://signaturescience.github.io/pracpac/, https://github.com/signaturescience/pracpac/

BugReports https://github.com/signaturescience/pracpac/issues

Repository https://stephenturner.r-universe.dev

RemoteUrl https://github.com/signaturescience/pracpac

RemoteRef HEAD

RemoteSha 5e426dce72ebfd129dec0ed2b0b7eefc273576a7

Contents

---

add_assets   *Add assets for the specified use case*

Description

Add template assets for the use case specified in add_dockerfile or use_docker.

Usage

```
add_assets(
  pkg_path = ".",
  img_path = NULL,
  use_case = "default",
  overwrite = TRUE
)
```

Arguments

pkg_path	Path to the package directory. Default is "." for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
use_case	Name of the use case. Defaults to "default", which only uses the base boilerplate.
overwrite	Logical; should existing assets should be overwritten? Default is TRUE.

Details

Example #1: the "shiny" use case requires than an app.R file moved into /srv/shiny-server/ in the container image. Using add_assets(use_case="shiny") (or when using the "shiny" use case in add_dockerfile or use_docker) will create a placeholder assets/app.R in the docker/ directory. The Dockerfile for the "shiny" use case will place COPY assets/app.R/srv/shiny-server into the Dockerfile.

Example #2: the "pipeline" use case creates boilerplate for moving pre- and post-processing R and shell scripts into the container at add_assets(use_case="pipeline") (or when using the

"pipeline" use case in add_dockerfile or use_docker) will create a placeholder assets/pre.R, assets/post.R, and assets/run.sh into the docker/assets directory. The Dockerfile for the "pipeline" use case will place COPY assets/run.sh /run.sh into the Dockerfile.

This function is run as part of use_docker but can be used on its own.

See vignette("use-cases", package="pracpac") for details on use cases.

Value

Invisibly returns assets per handle_use_case. Called primarily for its side effects.

Examples

```
## Not run:

# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Add assets for shiny use case
add_assets(pkg_path = file.path(tempdir(), "hellow"), use_case="shiny")
# Add assets for pipeline use case
add_assets(pkg_path = file.path(tempdir(), "hellow"), use_case="pipeline")

## End(Not run)
```

---

add_dockerfile *Add a Dockerfile to the docker directory*

Description

Adds a Dockerfile to the docker directory created by create_docker_dir. Allows for specification of several preset use cases, whether or not use use renv to manage dependencies, and optional overriding the base image.

Usage

```
add_dockerfile(
  pkg_path = ".",
  img_path = NULL,
  use_renv = TRUE,
  use_case = "default",
  base_image = NULL,
  repos = NULL
)
```

Arguments

pkg_path	Path to the package directory. Default is "." for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
use_renv	Logical; use renv? Defaults to TRUE. If FALSE, package dependencies are scraped from the DESCRIPTION file and the most recent versions will be installed in the image.
use_case	Name of the use case. Defaults to "default", which only uses the base boiler-plate. See vignette("use-cases", package="pracpac") for other use cases (e.g., shiny, rstudio, pipeline).
base_image	Name of the base image to start FROM. Default is NULL and the base image will be derived based on use_case. Optionally override this by setting the name of the base image (including tag if desired).
repos	Option to override the repos used for installing packages with renv by passing name of repository. Only used if use_renv = TRUE. Default is NULL meaning that the repos specified in renv lockfile will remain as-is and not be overridden.

Details

This function is run as part of [use_docker](#) but can be used on its own.

See vignette("use-cases", package="pracpac") for details on use cases.

Value

Invisibly returns a list of package info returned by [pkg_info](#). Primarily called for side-effect to create Dockerfile.

Examples

```
## Not run:

# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Default: FROM rocker/r-ver:latest with no additional template
# By default add_dockerfile requires you either to specify use_renv = FALSE
# Or run renv_deps() prior to add_dockerfile()
# The use_docker() wrapper runs these sequentially, and is recommended for most usage
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), use_renv = FALSE)
# Specify tidyverse base image
renv_deps(pkg_path = file.path(tempdir(), "hellow"))
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), base_image="rocker/tidyverse:4.2.2")
# Specify different default repo
```

```
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), repos="https://cran.wustl.edu/")
# RStudio template
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), use_case="rstudio")
# Shiny template
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), use_case = "shiny")
# Pipeline template
add_dockerfile(pkg_path = file.path(tempdir(), "hellow"), use_case="pipeline")

## End(Not run)
```

---

build_image	*Build a Docker image*

Description

Builds a Docker image created by use_docker or add_dockerfile. This function is run as part of use_docker when build = TRUE is set, but can be used on its own.

Usage

```
build_image(
  pkg_path = ".",
  img_path = NULL,
  cache = TRUE,
  tag = NULL,
  build = TRUE
)
```

Arguments

pkg_path	Path to the package directory. Default is "." for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
cache	Logical; should caching be used? Default TRUE. Set to FALSE to use --no-cache in docker build.
tag	Image tag to use; default is NULL and the image will be tagged with package name version from pkg_info.
build	Logical as to whether or not the image should be built. Default is TRUE, and if FALSE the docker build command will be messaged. Setting build=FALSE could be useful if additional docker build options or different tags are desired. In either case the docker build command will be returned invisibly.

Value

Invisibly returns the docker build command. Primarily called for its side effects, which runs the docker build as a system command.

Examples

```
## Not run:
# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Run use_docker to create Docker directory and assets for the example package
use_docker(pkg_path = file.path(tempdir(), "hellow"))

# Build the image
build_image(pkg_path = file.path(tempdir(), "hellow"))
# Or construct the image build command without building
build_cmd <- build_image(pkg_path = file.path(tempdir(), "hellow"), build=FALSE)
build_cmd

## End(Not run)
```

---

build_pkg	*Build a package tar.gz*

Description

Builds a package source tar.gz using [pkgbuild::build](#) and moves it into a user-specified location (default docker/).

Usage

```
build_pkg(pkg_path = ".", img_path = NULL, ...)
```

Arguments

pkg_path	Path to the package directory. Default is "." for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
...	Additional optional arguments passed to [pkgbuild::build](#).

Value

Invisibly returns a list of package info returned by [pkg_info](#), tar.gz source and destination file paths.

Examples

```
## Not run:
# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Build the example package from tempdir()
build_pkg(pkg = file.path(tempdir(), "hellow"))

## End(Not run)
```

---

create_docker_dir   *Create Docker directory*

Description

Creates a docker/ directory for a given package. By default, assumes that docker/ should be a
subdirectory of the specified package path.

Usage

```
create_docker_dir(pkg_path = ".", img_path = NULL)
```

Arguments

pkg_path	Path to the package directory. Default is "." for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.

Details

This function is run as part of use_docker but can be used on its own.

Value

Invisibly returns a list of package info returned by pkg_info. Primarily called for side-effect to
create docker directory.

Examples

```
## Not run:
# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Assuming default behavior then docker/ will be created under source root
create_docker_dir(pkg_path = file.path(tempdir(), "hellow"))

# Alternatively you can specify another directory above, below, or beside package source
create_docker_dir(pkg_path = file.path(tempdir(), "hellow"), img_path = file.path(tempdir(), "img"))

## End(Not run)
```

---

handle_use_case   *Handle the use case*

Description

This unexported helper function internally handles the provided use case.

Usage

```
handle_use_case(use_case)
```

Arguments

use_case   The specified use case.

Value

List of parsed information for the use case including, the name of the use case, path to Dockerfile template, base image, and path to assets (delimited by ; if there are multiple and NA if there are none).

---

pkg_info   *Get information about the current package*

Description

Returns information about the current package in a list which can be passed to other functions.

Usage

```
pkg_info(pkg_path = ".", ...)
```

Arguments

pkg_path   Path to the package directory. Default is `"."` for the current working directory,
                which assumes developer is working in R package root. However, this can be
                set to another path as needed.

...   Arguments passed to rprojroot::find_package_root_file.

Value

A list of information about the package.

- `pkgroot`: Root directory of the package.

- `pkgdeps`: Package dependencies from `Imports` in the `DESCRIPTION`.

- `descfile`: File path to the `DESCRIPTION` file.

- `pkgname`: Package name.

- `pkgver`: Package version.

Examples

```
## Not run:
# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# This will succeed if this is a package
pkg_info(pkg_path = file.path(tempdir(), "hellow"))
# This will fail if this is not a package location
pkg_info(pkg_path = tempdir())

## End(Not run)
```

---

pkg_root　　　　　　　　　　　*Find package root*

Description

Unexported helper to find the root of the R package. Returns an error if the path specified is not an R package.

Usage

```
pkg_root(pkg_path = ".", ...)
```

Arguments

pkg_path	Path to the package directory. Default is `"."` for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
...	Arguments passed to rprojroot::find_package_root_file.

Value

A file path of the package root. If no package is found at the root then the function will `stop` with an error message.

---

renv_deps　　　　　　　　　*Get dependencies using renv*

Description

Get dependencies using renv. This function will inspect your package specified at `pkg_path` (default is current working directory, `.`), and create an renv lock file (`renv.lock`) in the `docker/` directory. More information about the `renv` implementation is provided in the Details section.

Usage

```
renv_deps(
  pkg_path = ".",
  img_path = NULL,
  other_packages = NULL,
  overwrite = TRUE,
  consent_renv = TRUE
)
```

Arguments

pkg_path	Path to the package directory. Default is `"."` for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.
img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
other_packages	Vector of other packages to be included in `renv` lock file; default is NULL.
overwrite	Logical; should an existing lock file should be overwritten? Default is TRUE.
consent_renv	Logical; give renv consent in this session with `options(renv.consent = TRUE)`? Default is TRUE. See renv::consent for details.

Details

The `renv.lock` file will capture all your package's dependencies (and all their dependencies) at the current version installed on your system at the time this function is run. When using the default use_renv=TRUE in use_docker or add_dockerfile, the resulting Dockerfile will install packages from this `renv.lock` file using renv::restore. This ensures that versions of dependencies in the image mirror what is installed on your system at the time of image creation, rather than potentially newer versions on package repositories like CRAN or Bioconductor, which may come with breaking changes that you are unaware of at the time of package development.

If there are additional R packages that may be useful for the Docker image you plan to build (but may not be captured under your package dependencies), then you can add these packages to the renv procedure with the "other_packages" argument.

This function is run as part of use_docker but can be used on its own.

Value

Invisibly returns a list of package info returned by pkg_info. Primarily called for side effect. Writes an `renv` lock file to the docker/ directory.

Examples

```
## Not run:
# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Run using defaults; only gets current package dependencies
renv_deps(pkg_path = file.path(tempdir(), "hellow"))
# Add additional packages not explicitly required by your package
renv_deps(pkg_path = file.path(tempdir(), "hellow"), other_packages=c("shiny", "knitr"))

## End(Not run)
```

---

use_docker   *Use docker packaging tools*

Description

Wrapper function around other `pracpac` functions. See help for the functions linked below for detail on individual functions. All arguments to `use_docker()` are passed to downstream functions. `use_docker()` will sequentially run:

1. pkg_info to get information about the current R package.

2. create_docker_dir to create the `docker/` directory in the specified location, if it doesn't already exist.

3. renv_deps (if `use_renv=TRUE`, the default) to capture package dependencies with renv and create an `renv.lock` file

4. add_dockerfile to create a Dockerfile using template specified by `use_case`

5. add_assets depending on the `use_case`

6. build_pkg to build the current R package source .tar.gz, and place it into the `docker/` directory

7. build_image optional, default `FALSE`; if TRUE, will build the Docker image.

The default `build=FALSE` means that everything up to `build_image()` is run, but the image is not actually built. Instead, `use_docker()` will message the `docker build` command, and return that string in `$buildcmd` in the invisibly returned output.

See `vignette("use-cases", package="pracpac")` for details on use cases.

Usage

```
use_docker(
  pkg_path = ".",
  img_path = NULL,
  use_renv = TRUE,
  use_case = "default",
  base_image = NULL,
  other_packages = NULL,
  build = FALSE,
  repos = NULL,
  overwrite_assets = TRUE,
  overwrite_renv = TRUE,
  consent_renv = TRUE
)
```

Arguments

pkg_path   Path to the package directory. Default is `"."` for the current working directory, which assumes developer is working in R package root. However, this can be set to another path as needed.

img_path	Path to the write the docker image definition contents. The default NULL will use docker/ as a subdirectory of the pkg_path.
use_renv	Logical; use renv? Defaults to TRUE. If FALSE, package dependencies are scraped from the DESCRIPTION file without version information.
use_case	Name of the use case. Defaults to "default", which only uses the base boiler-plate.
base_image	Name of the base image to start FROM. Default is NULL and the base image will be derived based on use_case. Optionally override this by setting the name of the base image (including tag if desired).
other_packages	Vector of other packages to be included in renv lock file; default is NULL.
build	Logical as to whether or not the image should be built. Default is TRUE, and if FALSE the docker build command will be messaged. Setting build=FALSE could be useful if additional docker build options or different tags are desired. In either case the docker build command will be returned invisibly.
repos	Option to override the repos used for installing packages with renv by passing name of repository. Only used if use_renv = TRUE. Default is NULL meaning that the repos specified in renv lockfile will remain as-is and not be overridden.
overwrite_assets	
	Logical; should existing asset files should be overwritten? Default is TRUE.
overwrite_renv	Logical; should an existing lock file should be overwritten? Default is TRUE; ignored if use_renv = TRUE.
consent_renv	Logical; give renv consent in this session with options(renv.consent = TRUE)? Default is TRUE. See [renv::consent](#) for details.

Value

Invisibly returns a list with information about the package ($info) and the docker build command ($buildcmd). Primarily called for side effect. Creates docker/ directory, identifies renv dependencies and creates lock file (if use_renv = TRUE), writes Dockerfile, builds package tar.gz, moves all relevant assets to the docker/ directory, and builds Docker image (if build = TRUE).

Examples

```
## Not run:

# Specify path to example package source and copy to tempdir()
# Note that in practice you do not need to copy to a tempdir()
# And in fact it may be easiest to use pracpac relative to your package directory root
ex_pkg_src <- system.file("hellow", package = "pracpac", mustWork = TRUE)
file.copy(from = ex_pkg_src, to = tempdir(), recursive = TRUE)

# Run use_docker to create Docker directory and assets for the example package
use_docker(pkg_path = file.path(tempdir(), "hellow"))
# To not use renv
use_docker(pkg_path = file.path(tempdir(), "hellow"), use_renv=FALSE)
# To specify a use case
use_docker(pkg_path = file.path(tempdir(), "hellow"), use_case="pipeline")
# To overwrite the default base image
```

```
use_docker(pkg_path = file.path(tempdir(), "hellow"), base_image="alpine:latest")

## End(Not run)
```

Index