Package: relic (via r-universe)

August 25, 2024

Title Objects from History

Version 0.1.0.9000

Description The 'relic' package provides tools for extracting files and objects from the revision history, including local and remote git repositories and S3 buckets. It is a high-level interface designed to enable comparison of objects in reproducible research workflows, especially pipelines that use the 'targets' package.

License MIT + file LICENSE

URL https://ecohealthalliance.github.io/relic, https://ecohealthalliance.r-universe.dev/relic

BugReports https://github.com/ecohealthalliance/relic/issues

Imports fs, git2r, gh, rlang

Suggests callr, glue, knitr, lintr, minioclient, paws.storage, openssl, rmarkdown, rprojroot, spelling, styler, targets, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Remotes cboettig/minioclient#14, ropensci/git2r

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE, roclets = c(``collate", ``rd", ``namespace", ``devtag::dev_roclet"))

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/ecohealthalliance/relic

RemoteRef HEAD

RemoteSha 69cada8b9565ee09eba3d3e67322c39e16659f25

Contents

`create_example_repo` *Set up a testing and example repository*

Description

This function generates a repository with a commit history that can be used for testing and examples.

Usage

```
create_example_repo(
  dir = fs::file_temp("relic_example_"),
  reporter = "silent",
  s3 = TRUE,
  overwrite = TRUE
)
```

Arguments

`dir`	Path to the directory where the repository should be created
`reporter`	the reporter to use when building targets with `targets::tar_make()`. Defaults to "silent".
`s3`	Whether the repository should use S3 storage for targets. Note that the S3 endpoint and bucket must already be available.
`overwrite`	Whether to overwrite the directory if it already exists.

Value

The path to the created repository

Examples

```
example_repo <- create_example_repo(s3 = FALSE)
fs::dir_ls(example_repo, all = TRUE)
dir_ls_version(".", "initial-target-file", repo = example_repo)
```

`get_file_version` *Get in files from versioned repository*

Description

Fetches the contents of a file from a versioned repository - a local git repository, a GitHub repository, or an S3 bucket. Always fetches the file to a local cache and returns the path to it.

Usage

```
get_file_version(
  path,
  ref = "HEAD",
  repo = ".",
  endpoint = NULL,
  region = NULL
)
```

Arguments

`path`	Path of a file relative to the git directory or bucket
`ref`	A git commit SHA, tag, branch or other revision string: such as `"HEAD~1"`, `"main@{2023-02-26 18:30:00}"`, or `"branch@{yesterday}"`. Can also be a `git2r::commit()` object. For S3 buckets, a VersionID. Defaults to "HEAD", which also means the latest version in an S3 bucket.
`repo`	The repository to get the file from. This can be a local git directory, a GitHub repository (URL or `"owner/repo"` string), or an S3 bucket indicated by "s3://bucket-name". Defaults to the current working directory.
`endpoint, region`	For S3 buckets, the endpoint and region of the bucket. If NULL, the default endpoint and region in local config or environment variables are used. (Usually `us-east-1` and `s3.amazonaws.com`.)

Value

A path to the file in the local cache.

`relic_cache` *The relic cache*

Description

The relic cache directory stores files that have been retrieved from both local and remote repositories to avoid repeated extractions or downloads. Its location can be set with the environment variable `RELIC_CACHE_DIR` or `options("relic.cache.dir")`, and it defaults to the user cache directory. The cache is cleaned up regularly at package startup, but can also be cleaned up manually with `relic_cache_cleanup()` or cleared entirely with `relic_cache_delete()`.

Usage

```
relic_cache()

relic_cache_delete()

relic_cache_cleanup(
  max_age = relic_cache_max_age(),
  max_size = relic_cache_max_size()
)

relic_cache_regular_cleanup(cleanup_time = relic_cache_cleanup_time())

relic_cache_max_size()

relic_cache_max_age()

relic_cache_cleanup_time()
```

Arguments

`max_age`	The maximum age of files to keep in the cache, as a difftime object. Files older than this will be deleted. Defaults to `Inf`. Can be set with the environment variable `RELIC_CACHE_MAX_AGE` or `options("relic.cache.max.age")`, which take numeric time in days or a string with units, e.g., "1 day" or "2 weeks".
`max_size`	The maximum size of the cache, as a string that can be parsed by `fs::fs_bytes()`. Defaults to "20 MB". Can be set with the environment variable `RELIC_CACHE_MAX_SIZE` or `options("relic.cache.max.size")`. Cached files will be deleted from oldest to youngest until the cache size is under this limit.
`cleanup_time`	The time between cache cleanups, as a difftime object. Defaults to 1 day. Can be set with the environment variable `RELIC_CACHE_CLEANUP_TIME` or `options("relic.cache.cleanup.ti` which take numeric time in days or a string with units, e.g., "1 day" or "2 weeks". If set to "Inf", no cleanup will be performed at startup.

Value

The path to the relic cache directory

Examples

```
relic_cache()
```

`tar_meta_version` *Read a target project's metadata from repository history*

Description

This function extracts targets metadata from versioned history. In most cases this is a git repository, but it can also be an S3 cloud bucket for a project using cloud versioning and storing the metadata file in the cloud. (See `repository_meta` in `targets::tar_option_set()`).

Usage

```
tar_meta_version(
  ref = "HEAD",
  repo = ".",
  store = NULL,
  endpoint = NULL,
  region = NULL,
  ...
)
```

Arguments

`ref`	A git commit SHA, tag, branch or other revision string: such as `"HEAD~1"`, `"main@{2023-02-26 18:30:00}"`, or `"branch@{yesterday}"`. Can also be a `git2r::commit()` object. For S3 buckets, a VersionID. Defaults to "HEAD", which also means the latest version in an S3 bucket.
`repo`	The repository to get the file from. This can be a local git directory, a GitHub repository (URL or `"owner/repo"` string), or an S3 bucket indicated by "s3://bucket-name". Defaults to the current working directory.
`store`	Path to the targets store within the project. Defaults to `"_targets`, or the current project's store name if `repo = "."`.
`endpoint, region`	For S3 buckets, the endpoint and region of the bucket.
`...`	Arguments passed to `targets::tar_meta()`

Value

A data frame with one row per target/object. See `targets::tar_meta()` for details.

`tar_read_version` *Read a target's value from a git repository*

Description

Reads the content of targets from a git repository. Target metadata and local target files are extracted into a temporary directory before being read in by `tar_read()`. For targets of type "file", the files are also extracted and the paths to the extracted files are returned.

Usage

```
tar_read_version(name, ref = "HEAD", repo = ".", store = NULL)

tar_read_raw_version(name, ref = "HEAD", repo = ".", store = NULL)
```

Arguments

`name`	Name of the target. `tar_read_version()` can take a symbol, `tar_read_raw_version()` requires a character.
`ref`	A git commit SHA, tag, branch or other revision string: such as `"HEAD~1"`, `"main@{2023-02-26 18:30:00}"`, or `"branch@{yesterday}"`. Can also be a `git2r::commit()` object. For S3 buckets, a VersionID. Defaults to "HEAD", which also means the latest version in an S3 bucket.
`repo`	The repository to get the file from. This can be a local git directory, a GitHub repository (URL or `"owner/repo"` string), or an S3 bucket indicated by "s3://bucket-name". Defaults to the current working directory.
`store`	Path to the targets store within the project. Defaults to `"_targets`, or the current project's store name if `repo = "."`.

Details

For cloud targets, the target metadata is read from git history and then this metadata is used to download the target from the cloud. For this to work, cloud storage must be set up with versioning. Note that targets metadata includes the bucket, endpoint, and region of a S3-stored target, but you must still provide an AWS access key and secret as environment variables. If these differ from the credentials used for your current project or environment, can use `withr::with_envvar()` to temporarily set the credentials.

Value

The target's value. If the target is of `format = "file"`, this will be the path to the file in the relic cache.

Index