Package: tarchives (via r-universe)

July 13, 2026

Title Make Your 'targets' Pipelines into a Package

Version 0.2.0.9000

Description Runs 'targets' pipelines bundled inside a package and caches the results in the R user cache directory, so that users of the package do not need to rerun the pipeline themselves. Package authors can update the cached results at any time by releasing a new package version.

License MIT + file LICENSE

Encoding UTF-8

Imports callr, cli, fs, rlang (>= 1.1.0), targets, withr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.0.0)

Roxygen list(markdown = TRUE)

URL https://github.com/UchidaMizuki/tarchives, https://uchidamizuki.github.io/tarchives/

BugReports https://github.com/UchidaMizuki/tarchives/issues

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

Config/pak/sysreqs cmake libglpk-dev make libuv1-dev libxml2-dev

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

Date/Publication 2026-07-12 12:12:36 UTC

RemoteUrl https://github.com/uchidamizuki/tarchives

RemoteRef HEAD

RemoteSha abfd7ca4b37c2ffd6e191543f2f38c53dd17ad0c

Contents

tar_archive . . . 2
tar_archive_pipelines . . . 3
tar_archive_script . . . 4
tar_archive_store . . . 5
tar_destroy_archive . . . 6
tar_load_archive . . . 7
tar_make_archive . . . 9
tar_manifest_archive . . . 12
tar_meta_archive . . . 14
tar_read_archive . . . 16
tar_source_archive . . . 17
tar_target_archive . . . 18
use_tarchives . . . 24

Index **25**

`tar_archive` *Function factory for archived targets*

Description

Function factory for archived targets

Usage

```
tar_archive(
  f,
  package,
  pipeline,
  envir = parent.frame(),
  script = targets::tar_config_get("script"),
  store = targets::tar_config_get("store")
)
```

Arguments

`f`	A function of targets package.
`package`	A scalar character of the package name.
`pipeline`	A scalar character of the pipeline name.
`envir`	An environment, where to run the target R script (default: `_targets.R`) if `callr_function` is `NULL`. Ignored if `callr_function` is anything other than `NULL`. `callr_function` should only be `NULL` for debugging and testing purposes, not for serious runs of a pipeline, etc. The `envir` argument of `tar_make()` and related functions always overrides the current value of `tar_option_get("envir")` in the current R session just before

running the target script file, so whenever you need to set an alternative `envir`, you should always set it with `tar_option_set()` from within the target script file. In other words, if you call `tar_option_set(envir = envir1)` in an interactive session and then `tar_make(envir = envir2, callr_function = NULL)`, then `envir2` will be used.

`script` Character of length 1, path to the target script file. Defaults to `tar_config_get("script")`, which in turn defaults to `_targets.R`. When you set this argument, the value of `tar_config_get("script")` is temporarily changed for the current function call. See `tar_script()`, `tar_config_get()`, and `tar_config_set()` for details about the target script file and how to set it persistently for a project.

`store` Character of length 1, path to the `targets` data store. Defaults to `tar_config_get("store")`, which in turn defaults to `_targets/`. When you set this argument, the value of `tar_config_get("store")` is temporarily changed for the current function call. See `tar_config_get()` and `tar_config_set()` for details about how to set the data store path persistently for a project.

Value

A function.

Examples

```
tar_outdated_archive <- tar_archive(
  targets::tar_outdated,
  package = "tarchives",
  pipeline = "example-model"
)


withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  tar_outdated_archive()
)
```

---

tar_archive_pipelines *List the archived pipelines in a package*

Description

Returns the names of the pipelines bundled in a package's `inst/tarchives` directory. Use it to discover which pipelines are available before running them with `tar_make_archive()` or inspecting their targets with `tar_manifest_archive()`.

Usage

```
tar_archive_pipelines(
  package,
  envir = parent.frame(),
  script = targets::tar_config_get("script")
)
```

Arguments

package	A scalar character of the package name.
envir	An environment used to resolve package, so that a package currently loaded with pkgload::load_all() (e.g. during interactive package development) is resolved to its source directory instead of its installed one. Defaults to the calling environment.
script	Character of length 1, path to the target script file. Defaults to tar_config_get("script"), which in turn defaults to _targets.R. When you set this argument, the value of tar_config_get("script") is temporarily changed for the current function call. See tar_script(), tar_config_get(), and tar_config_set() for details about the target script file and how to set it persistently for a project.

Value

A character vector of pipeline names. A pipeline is a directory in inst/tarchives that contains a target script file (script), so the shared R/ helper directory is not included.

See Also

tar_manifest_archive() to list the targets within a pipeline.

Examples

```
tar_archive_pipelines(package = "tarchives")
```

---

tar_archive_script *Path to the archived target script file*

Description

Path to the archived target script file

Usage

```
tar_archive_script(
  package,
  pipeline,
  envir = parent.frame(),
  script = targets::tar_config_get("script")
)
```

Arguments

package	A scalar character of the package name.
pipeline	A scalar character of the pipeline name.
envir	An environment used to resolve package, so that a package currently loaded with pkgload::load_all() (e.g. during interactive package development) is resolved to its source directory instead of its installed one. Defaults to the calling environment.
script	Character of length 1, path to the target script file. Defaults to tar_config_get("script"), which in turn defaults to _targets.R. When you set this argument, the value of tar_config_get("script") is temporarily changed for the current function call. See tar_script(), tar_config_get(), and tar_config_set() for details about the target script file and how to set it persistently for a project.

Value

A scalar character of the path to the archived target script file.

Examples

```
tar_archive_script(package = "tarchives", pipeline = "example-model")
```

---

tar_archive_store *Path to the archived target store directory*

Description

Path to the archived target store directory

Usage

```
tar_archive_store(package, pipeline, store = targets::tar_config_get("store"))
```

Arguments

package	A scalar character of the package name.
pipeline	A scalar character of the pipeline name.
store	Character of length 1, path to the targets data store. Defaults to tar_config_get("store"), which in turn defaults to _targets/. When you set this argument, the value of tar_config_get("store") is temporarily changed for the current function call. See tar_config_get() and tar_config_set() for details about how to set the data store path persistently for a project.

Value

A scalar character of the path to the archived target store directory.

Examples

```
tar_archive_store(package = "tarchives", pipeline = "example-model")
```

`tar_destroy_archive` *Destroy an archived pipeline's storage*

Description

tarchives version of `targets::tar_destroy()`. Removes the cached `targets` store of an archived pipeline from the R user cache directory (`tools::R_user_dir("tarchives", "cache")`).

Usage

```
tar_destroy_archive(
  package,
  pipeline,
  destroy = "all",
  batch_size = 1000L,
  verbose = TRUE,
  ask = NULL,
  store = targets::tar_config_get("store")
)
```

Arguments

`package`	A scalar character of the package name.
`pipeline`	A scalar character of the pipeline name.
`destroy`	Character of length 1, what to destroy. Choices: • `"all"`: entire data store (default: `_targets/`) including cloud data, as well as download/upload scratch files. • `"cloud"`: cloud data, including metadata, target definition object data from targets with `tar_target(..., repository = "aws")`, and workspace files saved on the cloud. Also deletes temporary staging files in `file.path(tempdir(), "targets")` that may have been accidentally left over from incomplete uploads or downloads. • `"local"`: all the local files in the data store but nothing on the cloud. • `"meta"`: metadata file at `meta/meta` in the data store, which invalidates all the targets but keeps the data. • `"process"`: progress data file at `meta/process` in the data store, which resets the metadata of the main process. • `"progress"`: progress data file at `meta/progress` in the data store, which resets the progress tracking info. • `"objects"`: all the target return values in `objects/` in the data store but keep progress and metadata. Dynamic files are not deleted this way.

- "scratch": temporary files in saved during tar_make() that should automatically get deleted except if R crashed.
- "workspaces": compressed lightweight files locally saved to the workspaces/ folder in the data store with the saved workspaces of targets. Does not delete workspace files on the cloud. For that, consider destroy = "all" or destroy = "cloud". See tar_workspace() for details.
- "user": custom user-supplied files in the user/ folder in the data store.

batch_size — Positive integer between 1 and 1000, number of target definition objects to delete from the cloud with each HTTP API request. Currently only supported for AWS. Cannot be more than 1000.

verbose — Logical of length 1, whether to print console messages to show progress when deleting each batch of targets from each cloud bucket. Batched deletion with verbosity is currently only supported for AWS.

ask — Logical of length 1, whether to pause with a menu prompt before deleting files. To disable this menu, set the TAR_ASK environment variable to "false". usethis::edit_r_environ() can help set environment variables.

store — Character of length 1, path to the targets data store. Defaults to tar_config_get("store"), which in turn defaults to _targets/. When you set this argument, the value of tar_config_get("store") is temporarily changed for the current function call. See tar_config_get() and tar_config_set() for details about how to set the data store path persistently for a project.

Value

NULL (invisibly).

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  {
    tar_make_archive(package = "tarchives", pipeline = "example-model")
    tar_destroy_archive(package = "tarchives", pipeline = "example-model", ask = FALSE)
  }
)
```

---

tar_load_archive *Load a target's value from archive storage*

Description

tarchives version of targets::tar_load(). Reads one or more targets from an archived pipeline's store and assigns them into an environment.

Usage

```
tar_load_archive(
  names,
  package,
  pipeline,
  branches = NULL,
  meta = NULL,
  strict = TRUE,
  silent = FALSE,
  envir = parent.frame(),
  store = targets::tar_config_get("store")
)
```

Arguments

names	Names of the targets to load. tar_load() uses non-standard evaluation in the names argument (example: tar_load(names = everything())), whereas tar_load_raw() uses standard evaluation for names (example: tar_load_raw(names = quote(everything()))). The object supplied to names should be a tidyselect expression like any_of() or starts_with() from tidyselect itself, or tar_described_as() to select target names based on their descriptions.
package	A scalar character of the package name.
pipeline	A scalar character of the pipeline name.
branches	Integer of indices of the branches to load for any targets that are patterns.
meta	Data frame of target metadata from tar_meta().
strict	Logical of length 1, whether to error out if one of the selected targets is in the metadata but cannot be loaded. Set to FALSE to just load the targets in the metadata that can be loaded and skip the others.
silent	Logical of length 1. Only relevant when strict is FALSE. If silent is FALSE and strict is FALSE, then a message will be printed if a target is in the metadata but cannot be loaded. However, load failures will not stop other targets from being loaded.
envir	R environment in which to load target return values.
store	Character of length 1, directory path to the data store of the pipeline.

Value

Nothing.

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  {
    tar_make_archive(package = "tarchives", pipeline = "example-model")
```

```
    tar_load_archive(model, package = "tarchives", pipeline = "example-model")
    model
  }
)
```

`tar_make_archive` *Run an archived pipeline of targets*

Description

Run an archived pipeline of targets

Usage

```
tar_make_archive(
  package,
  pipeline,
  names = NULL,
  shortcut = targets::tar_config_get("shortcut"),
  reporter = "silent",
  seconds_meta_append = targets::tar_config_get("seconds_meta_append"),
  seconds_meta_upload = targets::tar_config_get("seconds_meta_upload"),
  seconds_reporter = targets::tar_config_get("seconds_reporter"),
  seconds_interval = targets::tar_config_get("seconds_interval"),
  callr_function = callr::r,
  callr_arguments = targets::tar_callr_args_default(callr_function, reporter),
  envir = parent.frame(),
  script = targets::tar_config_get("script"),
  store = targets::tar_config_get("store"),
  garbage_collection = NULL,
  use_crew = targets::tar_config_get("use_crew"),
  terminate_controller = TRUE,
  as_job = targets::tar_config_get("as_job")
)
```

Arguments

`package`	A scalar character of the package name.
`pipeline`	A scalar character of the pipeline name.
`names`	Names of the targets to run or check. Set to `NULL` to check/run all the targets (default). The object supplied to `names` should be a `tidyselect` expression like `any_of()` or `starts_with()` from `tidyselect` itself, or `tar_described_as()` to select target names based on their descriptions.

shortcut
: Logical of length 1, how to interpret the names argument. If shortcut is FALSE (default) then the function checks all targets upstream of names as far back as the dependency graph goes. shortcut = TRUE increases speed if there are a lot of up-to-date targets, but it assumes all the dependencies are up to date, so please use with caution. It relies on stored metadata for information about upstream dependencies. shortcut = TRUE only works if you set names.

reporter
: A scalar character of the reporter type. By default, "silent". See targets::tar_make() for more options.

seconds_meta_append
: Positive numeric of length 1 with the minimum number of seconds between saves to the local metadata and progress files in the data store. his is an aggressive optimization setting not recommended for most users: higher values generally make the pipeline run faster, but unsaved work (in the event of a crash) is not up to date. When the pipeline ends, all the metadata and progress data is saved immediately, regardless of seconds_meta_append.

  When the pipeline is just skipping targets, the actual interval between saves is max(1, seconds_meta_append) to reduce overhead.

seconds_meta_upload
: Positive numeric of length 1 with the minimum number of seconds between uploads of the metadata and progress data to the cloud (see https://books.ropensci.org/targets/cloud-storage.html). Higher values generally make the pipeline run faster, but unsaved work (in the event of a crash) may not be backed up to the cloud. When the pipeline ends, all the metadata and progress data is uploaded immediately, regardless of seconds_meta_upload.

seconds_reporter
: Deprecated on 2025-03-31 (targets version 1.10.1.9010).

seconds_interval
: Deprecated on 2023-08-24 (targets version 1.2.2.9001). Use seconds_meta_append and seconds_meta_upload instead.

callr_function
: A function from callr to start a fresh clean R process to do the work. Set to NULL to run in the current session instead of an external process (but restart your R session just before you do in order to clear debris out of the global environment). callr_function needs to be NULL for interactive debugging, e.g. tar_option_set(debug = "your_target"). However, callr_function should not be NULL for serious reproducible work.

callr_arguments
: A list of arguments to callr_function.

envir
: An environment, where to run the target R script (default: _targets.R) if callr_function is NULL. Ignored if callr_function is anything other than NULL. callr_function should only be NULL for debugging and testing purposes, not for serious runs of a pipeline, etc.

  The envir argument of tar_make() and related functions always overrides the current value of tar_option_get("envir") in the current R session just before running the target script file, so whenever you need to set an alternative envir, you should always set it with tar_option_set() from within the target script

file. In other words, if you call `tar_option_set(envir = envir1)` in an interactive session and then `tar_make(envir = envir2, callr_function = NULL)`, then `envir2` will be used.

`script` Character of length 1, path to the target script file. Defaults to `tar_config_get("script")`, which in turn defaults to `_targets.R`. When you set this argument, the value of `tar_config_get("script")` is temporarily changed for the current function call. See `tar_script()`, `tar_config_get()`, and `tar_config_set()` for details about the target script file and how to set it persistently for a project.

`store` Character of length 1, path to the `targets` data store. Defaults to `tar_config_get("store")`, which in turn defaults to `_targets/`. When you set this argument, the value of `tar_config_get("store")` is temporarily changed for the current function call. See `tar_config_get()` and `tar_config_set()` for details about how to set the data store path persistently for a project.

`garbage_collection`
Deprecated. Use the `garbage_collection` argument of `tar_option_set()` instead to run garbage collection at regular intervals in a pipeline, or use the argument of the same name in `tar_target()` to activate garbage collection for a specific target.

`use_crew` Logical of length 1, whether to use `crew` if the `controller` option is set in `tar_option_set()` in the target script (`_targets.R`). See https://books.ropensci.org/targets/crew.html for details.

`terminate_controller`
Logical of length 1. For a `crew`-integrated pipeline, whether to terminate the controller after stopping or finishing the pipeline. This should almost always be set to `TRUE`, but `FALSE` combined with `callr_function = NULL` will allow you to get the running controller using `tar_option_get("controller")` for debugging purposes. For example, `tar_option_get("controller")$summary()` produces a worker-by-worker summary of the work assigned and completed, `tar_option_get("controller")$queue` is the list of unresolved tasks, and `tar_option_get("controller")$results` is the list of tasks that completed but were not collected with `pop()`. You can manually terminate the controller with `tar_option_get("controller")$summary()` to close down the dispatcher and worker processes.

`as_job` `TRUE` to run as an RStudio IDE / Posit Workbench job, if running on RStudio IDE / Posit Workbench. `FALSE` to run as a `callr` process in the main R session (depending on the `callr_function` argument). If `as_job` is `TRUE`, then the `rstudioapi` package must be installed.

Value

`NULL` except if `callr_function = callr::r_bg()`, in which case a handle to the `callr` background process is returned. Either way, the value is invisibly returned.

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  tar_make_archive(package = "tarchives", pipeline = "example-model")
```

```
)
```

`tar_manifest_archive` *List the targets of an archived pipeline*

Description

tarchives version of `targets::tar_manifest()`. Returns a data frame of the targets defined in an archived pipeline's target script file.

Usage

```
tar_manifest_archive(
  package,
  pipeline,
  names = NULL,
  fields = NULL,
  drop_missing = TRUE,
  callr_function = callr::r,
  callr_arguments = targets::tar_callr_args_default(callr_function),
  envir = parent.frame(),
  script = targets::tar_config_get("script")
)
```

Arguments

`package`	A scalar character of the package name.
`pipeline`	A scalar character of the pipeline name.
`names`	Names of the targets to show. Set to `NULL` to show all the targets (default). Otherwise, the object supplied to `names` should be a `tidyselect` expression like `any_of()` or `starts_with()` from `tidyselect` itself, or `tar_described_as()` to select target names based on their descriptions.
`fields`	Names of the fields, or columns, to show. Set to `NULL` to show all the fields (default). Otherwise, the value of `fields` should be a `tidyselect` expression like `starts_with()` to select the columns to show in the output. Possible fields are below. All of them can be set in `tar_target()`, `tar_target_raw()`, or `tar_option_set()`.

- `name`: Name of the target.
- `command`: the R command that runs when the target runs.
- `description`: custom free-form text description of the target, if available.
- `pattern`: branching pattern of the target, if applicable.
- `format`: Storage format.
- `repository`: Storage repository.

- `iteration`: Iteration mode for branching.
- `error`: Error mode, what to do when the target fails.
- `memory`: Memory mode, when to keep targets in memory.
- `storage`: Storage mode for high-performance computing scenarios.
- `retrieval`: Retrieval mode for high-performance computing scenarios.
- `deployment`: Where/whether to deploy the target in high-performance computing scenarios.
- `priority`: Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get dispatched earlier (and polled earlier in `tar_make_future()`).
- `resources`: A list of target-specific resource requirements for `tar_make_future()`.
- `cue_mode`: Cue mode from `tar_cue()`.
- `cue_depend`: Depend cue from `tar_cue()`.
- `cue_expr`: Command cue from `tar_cue()`.
- `cue_file`: File cue from `tar_cue()`.
- `cue_format`: Format cue from `tar_cue()`.
- `cue_repository`: Repository cue from `tar_cue()`.
- `cue_iteration`: Iteration cue from `tar_cue()`.
- `packages`: List columns of packages loaded before running the target.
- `library`: List column of library paths to load the packages.

`drop_missing` Logical of length 1, whether to automatically omit empty columns and columns with all missing values.

`callr_function` A function from `callr` to start a fresh clean R process to do the work. Set to `NULL` to run in the current session instead of an external process (but restart your R session just before you do in order to clear debris out of the global environment). `callr_function` needs to be `NULL` for interactive debugging, e.g. `tar_option_set(debug = "your_target")`. However, `callr_function` should not be `NULL` for serious reproducible work.

`callr_arguments`

A list of arguments to `callr_function`.

`envir` An environment, where to run the target R script (default: `_targets.R`) if `callr_function` is `NULL`. Ignored if `callr_function` is anything other than `NULL`. `callr_function` should only be `NULL` for debugging and testing purposes, not for serious runs of a pipeline, etc.

The `envir` argument of `tar_make()` and related functions always overrides the current value of `tar_option_get("envir")` in the current R session just before running the target script file, so whenever you need to set an alternative `envir`, you should always set it with `tar_option_set()` from within the target script file. In other words, if you call `tar_option_set(envir = envir1)` in an interactive session and then `tar_make(envir = envir2, callr_function = NULL)`, then `envir2` will be used.

`script` Character of length 1, path to the target script file. Defaults to `tar_config_get("script")`, which in turn defaults to `_targets.R`. When you set this argument, the value of `tar_config_get("script")` is temporarily changed for the current function

call. See tar_script(), tar_config_get(), and tar_config_set() for details about the target script file and how to set it persistently for a project.

Value

A data frame of information about the targets in the pipeline. Rows appear in topological order (the order they will run without any influence from parallel computing or priorities).

See Also

tar_archive_pipelines() to list the pipelines in a package.

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  tar_manifest_archive(package = "tarchives", pipeline = "example-model")
)
```

---

tar_meta_archive *Read metadata from archive storage*

Description

tarchives version of targets::tar_meta(). Returns the metadata of an archived pipeline's store.

Usage

```
tar_meta_archive(
  package,
  pipeline,
  names = NULL,
  fields = NULL,
  targets_only = FALSE,
  complete_only = FALSE,
  store = targets::tar_config_get("store")
)
```

Arguments

package	A scalar character of the package name.
pipeline	A scalar character of the pipeline name.
names	Optional, names of the targets. If supplied, tar_meta() only returns metadata on these targets. You can supply symbols or tidyselect helpers like any_of() and starts_with(). If NULL, all names are selected.

`fields` Optional, names of columns/fields to select. If supplied, `tar_meta()` only returns the selected metadata columns. If `NULL`, all fields are selected. You can supply symbols or `tidyselect` helpers like `any_of()` and `starts_with()`. The `name` column is always included first no matter what you select. Choices:

- `name`: name of the target or global object.
- `type`: type of the object: either `"function"` or `"object"` for global objects, and `"stem"`, `"branch"`, `"map"`, or `"cross"` for targets.
- `data`: hash of the output data.
- `command`: hash of the target's deparsed command.
- `depend`: hash of the immediate upstream dependencies of the target.
- `seed`: random number generator seed with which the target ran. A target's random number generator seed is a deterministic function of its name. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with `tar_meta(your_target, seed)` and run `tar_seed_set()` on the result to locally recreate the target's initial RNG state.
- `path`: A list column of paths to target data. Usually, each element is a single path, but there could be multiple paths per target for file targets (i.e. `tar_target(format = "file")`).
- `time`: `POSIXct` object with the time the target's data in storage was last modified. If the target stores no local file, then the time stamp corresponds to the time the target last ran successfully. Only targets that run commands have time stamps: just non-branching targets and individual dynamic branches. Displayed in the current time zone of the system. If there are multiple outputs for that target, as with file targets, then the maximum time is shown.
- `size`: hash of the sum of all the bytes of the files at `path`.
- `bytes`: total file size in bytes of all files in `path`.
- `format`: character, one of the admissible data storage formats. See the `format` argument in the `tar_target()` help file for details.
- `iteration`: character, either `"list"` or `"vector"` to describe the iteration and aggregation mode of the target. See the `iteration` argument in the `tar_target()` help file for details.
- `parent`: for branches, name of the parent pattern.
- `children`: list column, names of the children of targets that have them. These include buds of stems and branches of patterns.
- `seconds`: number of seconds it took to run the target.
- `warnings`: character string of warning messages from the last run of the target. Only the first 50 warnings are available, and only the first 2048 characters of the concatenated warning messages.
- `error`: character string of the error message if the target errored.

`targets_only` Logical, whether to just show information about targets or also return metadata on functions and other global objects.

complete_only	Logical, whether to return only complete rows (no NA values).
store	Character of length 1, path to the targets data store. Defaults to tar_config_get("store"), which in turn defaults to _targets/. When you set this argument, the value of tar_config_get("store") is temporarily changed for the current function call. See tar_config_get() and tar_config_set() for details about how to set the data store path persistently for a project.

Value

A data frame with one row per target/object and the selected fields.

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  {
    tar_make_archive(package = "tarchives", pipeline = "example-model")
    tar_meta_archive(package = "tarchives", pipeline = "example-model")
  }
)
```

---

tar_read_archive *Read a target's value from archive storage*

Description

Read a target's value from archive storage

Usage

```
tar_read_archive(
  name,
  package,
  pipeline,
  branches = NULL,
  meta = NULL,
  store = targets::tar_config_get("store")
)

tar_read_archive_raw(
  name,
  package,
  pipeline,
  branches = NULL,
  meta = NULL,
  store = targets::tar_config_get("store")
)
```

Arguments

name	Name of the target to read. tar_read() expects an unevaluated symbol for the name argument, whereas tar_read_raw() expects a character string.
package	A scalar character of the package name.
pipeline	A scalar character of the pipeline name.
branches	Integer of indices of the branches to load if the target is a pattern.
meta	Data frame of metadata from tar_meta(). tar_read() with the default arguments can be inefficient for large pipelines because all the metadata is stored in a single file. However, if you call tar_meta() beforehand and supply it to the meta argument, then successive calls to tar_read() may run much faster.
store	Character of length 1, path to the targets data store. Defaults to tar_config_get("store"), which in turn defaults to _targets/. When you set this argument, the value of tar_config_get("store") is temporarily changed for the current function call. See tar_config_get() and tar_config_set() for details about how to set the data store path persistently for a project.

Details

tar_read_archive() captures name with non-standard evaluation, whereas tar_read_archive_raw() takes it as a character string.

Value

The target's return value from its file in _targets/objects/, or the paths to the custom files and directories if format = "file" was set.

Examples

```
withr::with_envvar(
  c(R_USER_CACHE_DIR = tempfile()),
  {
    tar_make_archive(package = "tarchives", pipeline = "example-model")
    tar_read_archive(model, package = "tarchives", pipeline = "example-model")
  }
)
```

---

tar_source_archive *Run archived R scripts*

Description

Run archived R scripts

Usage

```
tar_source_archive(
  package,
  files = "R",
  envir = targets::tar_option_get("envir"),
  change_directory = FALSE
)
```

Arguments

package	A scalar character of the package name.
files	Character vector of file and directory paths to look for R scripts to run. Paths must either be absolute paths or must be relative to the current working directory just before the function call.
envir	Environment to run the scripts. Defaults to `tar_option_get("envir")`, the environment of the pipeline.
change_directory	Logical, whether to temporarily change the working directory to the directory of each R script before running it.

Value

`NULL` (invisibly)

Examples

```
tar_source_archive(package = "tarchives")
```

---

tar_target_archive *Declare a target to read an archive*

Description

Declare a target to read an archive

Usage

```
tar_target_archive(
  name,
  package,
  pipeline,
  name_archive = NULL,
  ...,
  pattern = NULL,
  packages = targets::tar_option_get("packages"),
```

```
  library = targets::tar_option_get("library"),
  deps = NULL,
  string = NULL,
  format = targets::tar_option_get("format"),
  repository = targets::tar_option_get("repository"),
  iteration = targets::tar_option_get("iteration"),
  error = targets::tar_option_get("error"),
  memory = targets::tar_option_get("memory"),
  garbage_collection = isTRUE(targets::tar_option_get("garbage_collection")),
  deployment = targets::tar_option_get("deployment"),
  priority = targets::tar_option_get("priority"),
  resources = targets::tar_option_get("resources"),
  storage = targets::tar_option_get("storage"),
  retrieval = targets::tar_option_get("retrieval"),
  cue = targets::tar_option_get("cue"),
  description = targets::tar_option_get("description")
)

tar_target_archive_raw(
  name,
  package,
  pipeline,
  name_archive = name,
  ...,
  pattern = NULL,
  packages = targets::tar_option_get("packages"),
  library = targets::tar_option_get("library"),
  deps = NULL,
  string = NULL,
  format = targets::tar_option_get("format"),
  repository = targets::tar_option_get("repository"),
  iteration = targets::tar_option_get("iteration"),
  error = targets::tar_option_get("error"),
  memory = targets::tar_option_get("memory"),
  garbage_collection = isTRUE(targets::tar_option_get("garbage_collection")),
  deployment = targets::tar_option_get("deployment"),
  priority = targets::tar_option_get("priority"),
  resources = targets::tar_option_get("resources"),
  storage = targets::tar_option_get("storage"),
  retrieval = targets::tar_option_get("retrieval"),
  cue = targets::tar_option_get("cue"),
  description = targets::tar_option_get("description")
)
```

Arguments

name	Symbol, name of the target. In tar_target(), name is an unevaluated symbol, e.g. tar_target(name = data). In tar_target_raw(), name is a character string, e.g. tar_target_raw(name = "data").

A target name must be a valid name for a symbol in R, and it must not start with a dot. Subsequent targets can refer to this name symbolically to induce a dependency relationship: e.g. `tar_target(downstream_target, f(upstream_target))` is a target named `downstream_target` which depends on a target `upstream_target` and a function `f()`.

In most cases, The target name is the name of its local data file in storage. Some file systems are not case sensitive, which means converting a name to a different case may overwrite a different target. Please ensure all target names have unique names when converted to lower case.

In addition, a target's name determines its random number generator seed. In this way, each target runs with a reproducible seed so someone else running the same pipeline should get the same results, and no two targets in the same pipeline share the same seed. (Even dynamic branches have different names and thus different seeds.) You can recover the seed of a completed target with `tar_meta(your_target, seed)` and run `tar_seed_set()` on the result to locally recreate the target's initial RNG state.

`package` A scalar character of the package name.

`pipeline` A scalar character of the pipeline name.

`name_archive` Symbol, name of the archived target. If `NULL`, the name of the target is used. By default, `NULL`.

`...` Arguments to pass to `tar_make_archive()` or `tar_read_archive_raw()`.

`pattern` Code to define a dynamic branching branching for a target. In `tar_target()`, `pattern` is an unevaluated expression, e.g. `tar_target(pattern = map(data))`. In `tar_target_raw()`, `command` is an evaluated expression, e.g. `tar_target_raw(pattern = quote(map(data)))`.

To demonstrate dynamic branching patterns, suppose we have a pipeline with numeric vector targets `x` and `y`. Then, `tar_target(z, x + y, pattern = map(x, y))` implicitly defines branches of `z` that each compute `x[1] + y[1]`, `x[2] + y[2]`, and so on. See the user manual for details.

`packages` Character vector of packages to load right before the target runs or the output data is reloaded for downstream targets. Use `tar_option_set()` to set packages globally for all subsequent targets you define.

`library` Character vector of library paths to try when loading `packages`.

`deps` Optional character vector of the adjacent upstream dependencies of the target, including targets and global objects. If `NULL`, dependencies are resolved automatically as usual. The `deps` argument is only for developers of extension packages such as `tarchetypes`, not for end users, and it should almost never be used at all. In scenarios that at first appear to requires `deps`, there is almost always a simpler and more robust workaround that avoids setting `deps`.

`string` Optional string representation of the command. Internally, the string gets hashed to check if the command changed since last run, which helps `targets` decide whether the target is up to date. External interfaces can take control of `string` to ignore changes in certain parts of the command. If `NULL`, the strings is just deparsed from `command` (default).

`format` Optional storage format for the target's return value. With the exception of `format = "file"`, each target gets a file in `_targets/objects`, and each format is a different way to save and load this file. See the "Storage formats" section for a detailed list of possible data storage formats.

`repository` Character of length 1, remote repository for target storage. Choices:

- `"local"`: file system of the local machine.
- `"aws"`: Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the `endpoint` argument of `tar_resources_aws()`, but versioning capabilities may be lost in doing so. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.
- `"gcp"`: Google Cloud Platform storage bucket. See the cloud storage section of https://books.ropensci.org/targets/data.html for details for instructions.
- A character string from `tar_repository_cas()` for content-addressable storage.

Note: if `repository` is not `"local"` and `format` is `"file"` then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. As of `targets` version 1.11.0 and higher, the local file is no longer deleted after the target runs.

`iteration` Character of length 1, name of the iteration mode of the target. Choices:

- `"vector"`: branching happens with `vctrs::vec_slice()` and aggregation happens with `vctrs::vec_c()`.
- `"list"`, branching happens with `[[]]` and aggregation happens with `list()`.
- `"group"`: `dplyr::group_by()`-like functionality to branch over subsets of a non-dynamic data frame. For `iteration = "group"`, the target must not by dynamic (the `pattern` argument of `tar_target()` must be left `NULL`). The target's return value must be a data frame with a special `tar_group` column of consecutive integers from 1 through the number of groups. Each integer designates a group, and a branch is created for each collection of rows in a group. See the `tar_group()` function to see how you can create the special `tar_group` column with `dplyr::group_by()`.

`error` Character of length 1, what to do if the target stops and throws an error. Options:

- `"stop"`: the whole pipeline stops and throws an error.
- `"continue"`: the whole pipeline keeps going.
- `"null"`: The errored target continues and returns `NULL`. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline. In addition, as of `targets` version 1.8.0.9011, a value of `NULL` is given to upstream dependencies with `error = "null"` if loading fails.
- `"abridge"`: any currently running targets keep running, but no new targets launch after that.
- `"trim"`: all currently running targets stay running. A queued target is allowed to start if:
  1. It is not downstream of the error, and

2. It is not a sibling branch from the same tar_target() call (if the error happened in a dynamic branch).

The idea is to avoid starting any new work that the immediate error impacts. error = "trim" is just like error = "abridge", but it allows potentially healthy regions of the dependency graph to begin running. (Visit https://books.ropensci.org/targets/debugging.html to learn how to debug targets using saved workspaces.)

memory

Character of length 1, memory strategy. Possible values:

- "auto" (default): equivalent to memory = "transient" in almost all cases. But to avoid superfluous reads from disk, memory = "auto" is equivalent to memory = "persistent" for for non-dynamically-branched targets that other targets dynamically branch over. For example: if your pipeline has tar_target(name = y, command = x, pattern = map(x)), then tar_target(name = x, command = f(), memory = "auto") will use persistent memory for x in order to avoid rereading all of x for every branch of y.
- "transient": the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value.
- "persistent": the target stays in memory until the end of the pipeline (unless storage is "worker", in which case targets unloads the value from memory right after storing it in order to avoid sending copious data over a network).

For cloud-based file targets (e.g. format = "file" with repository = "aws"), the memory option applies to the temporary local copy of the file: "persistent" means it remains until the end of the pipeline and is then deleted, and "transient" means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

garbage_collection

Logical: TRUE to run base::gc() just before the target runs, in whatever R process it is about to run (which could be a parallel worker). FALSE to omit garbage collection. Numeric values get converted to FALSE. The garbage_collection option in tar_option_set() is independent of the argument of the same name in tar_target().

deployment

Character of length 1. If deployment is "main", then the target will run on the central controlling R process. Otherwise, if deployment is "worker" and you set up the pipeline with distributed/parallel computing, then the target runs on a parallel worker. For more on distributed/parallel computing in targets, please visit https://books.ropensci.org/targets/crew.html.

priority

Deprecated on 2025-04-08 (targets version 1.10.1.9013). targets has moved to a more efficient scheduling algorithm (https://github.com/ropensci/targets/issues/1458) which cannot support priorities. The priority argument of tar_target() no longer has a reliable effect on execution order.

resources

Object returned by tar_resources() with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See tar_resources() for details.

storage — Character string to control when the output of the target is saved to storage. Only relevant when using targets with parallel workers (https://books.ropensci.org/targets/crew.html). Must be one of the following values:

- "worker" (default): the worker saves/uploads the value.
- "main": the target's return value is sent back to the host machine and saved/uploaded locally.
- "none": targets makes no attempt to save the result of the target to storage in the location where targets expects it to be. Saving to storage is the responsibility of the user. Use with caution.

retrieval — Character string to control when the current target loads its dependencies into memory before running. (Here, a "dependency" is another target upstream that the current one depends on.) Only relevant when using targets with parallel workers (https://books.ropensci.org/targets/crew.html). Must be one of the following values:

- "auto" (default): equivalent to retrieval = "worker" in almost all cases. But to avoid unnecessary reads from disk, retrieval = "auto" is equivalent to retrieval = "main" for dynamic branches that branch over non-dynamic targets. For example: if your pipeline has tar_target(x, command = f()), then tar_target(y, command = x, pattern = map(x), retrieval = "auto") will use "main" retrieval in order to avoid rereading all of x for every branch of y.
- "worker": the worker loads the target's dependencies.
- "main": the target's dependencies are loaded on the host machine and sent to the worker before the target runs.
- "none": targets makes no attempt to load its dependencies. With retrieval = "none", loading dependencies is the responsibility of the user. Use with caution.

cue — An optional object from tar_cue() to customize the rules that decide whether the target is up to date.

description — Character of length 1, a custom free-form human-readable text description of the target. Descriptions appear as target labels in functions like tar_manifest() and tar_visnetwork(), and they let you select subsets of targets for the names argument of functions like tar_make(). For example, tar_manifest(names = tar_described_as(starts_with("survival model"))) lists all the targets whose descriptions start with the character string "survival model".

Details

tar_target_archive() captures name and name_archive with non-standard evaluation, whereas tar_target_archive_raw() takes them as character strings.

The archive is built (if outdated) and read when the target runs, not when the target script is sourced, so inspecting the pipeline with targets::tar_manifest() or targets::tar_visnetwork() does not trigger a build. The target tracks the installed version of package, so it reruns and refreshes the data when a new version of the package providing the archive is installed, and is skipped otherwise. Downstream targets still only rebuild when the value actually changes.

Value

A target definition object. Users should not modify these directly, just feed them to `list()` in your target script file (default: `_targets.R`).

Examples

```
tar_target_archive(
  model,
  package = "tarchives",
  pipeline = "example-model"
)
```

`use_tarchives` *Use tarchives*

Description

Set up tarchives for an existing package.

Usage

```
use_tarchives(store = targets::tar_config_get("store"))
```

Arguments

`store`	Character of length 1, path to the `targets` data store. Defaults to `tar_config_get("store")`, which in turn defaults to `_targets/`. When you set this argument, the value of `tar_config_get("store")` is temporarily changed for the current function call. See `tar_config_get()` and `tar_config_set()` for details about how to set the data store path persistently for a project.

Value

No return value, called for side effects.

Examples

```
withr::with_tempdir(
  use_tarchives()
)
```

Index

any_of(), *8, 9*, *12*, *14, 15*

list(), *24*

starts_with(), *8, 9*, *12*, *14, 15*

tar_archive, 2
tar_archive_pipelines, 3
tar_archive_pipelines(), *14*
tar_archive_script, 4
tar_archive_store, 5
tar_config_get(), *3–5*, *7*, *11*, *14*, *16, 17*, *24*
tar_config_set(), *3–5*, *7*, *11*, *14*, *16, 17*, *24*
tar_cue(), *13*
tar_described_as(), *8, 9*, *12*
tar_destroy_archive, 6
tar_group(), *21*
tar_load(), *8*
tar_load_archive, 7
tar_load_raw(), *8*
tar_make(), *2*, *7*, *10*, *13*, *23*
tar_make_archive, 9
tar_make_archive(), *3*, *20*
tar_make_future(), *13*
tar_manifest(), *23*
tar_manifest_archive, 12
tar_manifest_archive(), *3, 4*
tar_meta(), *8*, *17*
tar_meta_archive, 14
tar_option_set(), *11, 12*, *22*
tar_read(), *17*
tar_read_archive, 16
tar_read_archive_raw
(tar_read_archive), 16
tar_read_archive_raw(), *20*
tar_read_raw(), *17*
tar_repository_cas(), *21*
tar_resources_aws(), *21*
tar_script(), *3–5*, *11*, *14*
tar_seed_set(), *15*, *20*
tar_source_archive, 17
tar_target(), *11, 12*, *15*, *19–22*
tar_target_archive, 18
tar_target_archive_raw
(tar_target_archive), 18
tar_target_raw(), *12*, *19, 20*
tar_visnetwork(), *23*
tar_workspace(), *7*
targets::tar_destroy(), *6*
targets::tar_load(), *7*
targets::tar_make(), *10*
targets::tar_manifest(), *12*, *23*
targets::tar_meta(), *14*
targets::tar_visnetwork(), *23*

use_tarchives, 24