Package: s3pe (via r-universe)

November 8, 2026

Title A Small, Dependable Client for S3 Object Stores

Version 0.3.1

Description Read and write data on any S3-compatible object store ('AWS S3', 'MinIO', 'Garage', 'Ceph', ...). read_s3() and write_s3() move data frames, lists and files, with the format chosen by the key's extension (CSV, JSON, Parquet, RDS, GeoJSON); list, inspect and delete objects with the same small vocabulary. Requests are signed with AWS Signature Version 4 over the real hash of the payload, transient failures are retried with exponential backoff, and every transfer is checked against the server's ETag and content length. Endpoint and credentials come from the environment and are never written to disk.

License MIT + file LICENSE

URL https://github.com/StrategicProjects/s3pe, https://strategicprojects.github.io/s3pe/r/

BugReports https://github.com/StrategicProjects/s3pe/issues

Depends R (>= 4.1)

Imports cli, httr2 (>= 1.0.0), jsonlite, openssl, readr, rlang (>= 1.1.0), tibble

Suggests knitr, nanoparquet, readxl, rmarkdown, sf, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate, pkgdown

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libssl-dev libx11-dev

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

Date/Publication 2026-11-07 18:11:49 UTC

RemoteUrl https://github.com/strategicprojects/s3pe

RemoteRef HEAD

RemoteSha 53de6c8798d5bc76706f21b9c8b4ee0486b17e2c

RemoteSubdir r

Contents

buckets_s3 . . . 2
copy_s3 . . . 3
delete_s3 . . . 3
download_s3 . . . 4
exists_s3 . . . 5
head_s3 . . . 5
list_s3 . . . 6
presign_s3 . . . 7
read_s3 . . . 7
s3_config . . . 9
sync_up_s3 . . . 10
upload_s3 . . . 11
use_bucket . . . 12
write_s3 . . . 13

Index **15**

`buckets_s3` *List the buckets the credentials can see*

Description

List the buckets the credentials can see

Usage

```
buckets_s3(config = s3_config())
```

Arguments

`config`	An `s3_config()`.

Value

A tibble with `name` and `created` (UTC).

Examples

```
buckets_s3()
```

`copy_s3` *Copy an object inside the store*

Description

A server-side `CopyObject`: nothing is downloaded. The copy keeps the source's content type and metadata.

Usage

```
copy_s3(from, to, bucket = NULL, config = s3_config())
```

Arguments

`from`, `to`	Keys, or `s3://bucket/key` URLs (the two may name different buckets).
`bucket`	Bucket; defaults to the config's bucket (`S3PE_BUCKET`).
`config`	An `s3_config()`.

Value

Invisibly, a one-row tibble: `bucket`, `key`, `size`, `etag`, `elapsed` of the copy.

Examples

```
copy_s3("etl/sales.parquet", "archive/sales-2026.parquet", bucket = "my-bucket")
copy_s3("s3://staging/report.pdf", "s3://public/report.pdf")
```

`delete_s3` *Delete objects*

Description

One key is a `DELETE`; several keys go in batches of 1000 (`DeleteObjects`). Deleting an object that does not exist is not an error; an object the store refuses to delete is.

Usage

```
delete_s3(key, bucket = NULL, config = s3_config())
```

Arguments

key	One or more keys (or s3://bucket/key URLs).
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
config	An s3_config().

Value

The number of keys deleted, invisibly.

Examples

```
delete_s3("report/draft.csv", bucket = "my-bucket")
delete_s3(list_s3("tmp/", bucket = "my-bucket")$key, bucket = "my-bucket")
```

---

download_s3 *Download an object to a file*

Description

Streams the object to disk. It is written next to its final name with the suffix .s3pe-part and renamed when complete and verified, so an interrupted download leaves nothing at path.

Usage

```
download_s3(key, path, bucket = NULL, overwrite = FALSE, config = s3_config())
```

Arguments

key	Object key inside bucket, or an s3://bucket/key URL.
path	Where to write. Its directory is created if needed.
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
overwrite	Replace path if it exists?
config	An s3_config().

Value

Invisibly, a one-row tibble: bucket, key, size, etag, elapsed.

Examples

```
download_s3("data/regions.geojson", "regions.geojson", bucket = "my-bucket")
```

exists_s3 *Does an object exist?*

Description

Does an object exist?

Usage

```
exists_s3(key, bucket = NULL, config = s3_config())
```

Arguments

key	Object key inside bucket, or an s3://bucket/key URL.
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
config	An s3_config().

Value

TRUE or FALSE. Anything but a plain "not found" (no credentials, access denied, server down) is an error.

Examples

```
exists_s3("data/manifest.json", bucket = "my-bucket")
```

head_s3 *Metadata of an object*

Description

A HEAD request: size, content type, modification time, ETag and user metadata, without downloading the object.

Usage

```
head_s3(key, bucket = NULL, config = s3_config())
```

Arguments

key	Object key inside bucket, or an s3://bucket/key URL.
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
config	An s3_config().

Value

A one-row tibble (`key`, `size`, `content_type`, `last_modified`, `etag`, and `metadata`, a list column holding the named list of user metadata), or `NULL` when the object does not exist.

Examples

```
head_s3("data/manifest.json", bucket = "my-bucket")
```

---

list_s3 *List the objects under a prefix*

Description

Every object whose key starts with `prefix`, across all pages of the listing. Reading a bucket's manifest (`metadata.json`, `manifest.json`) is usually faster than listing it, when the publisher writes one last: it says what is there, and it is there only when everything else is.

Usage

```
list_s3(prefix = NULL, bucket = NULL, delimiter = NULL, config = s3_config())
```

Arguments

`prefix`	Key prefix, `""` or `NULL` for the whole bucket, or an `s3://bucket/prefix` URL.
`bucket`	Bucket; defaults to the config's bucket.
`delimiter`	`NULL` lists every object under the prefix. `"/"` lists one level: the objects directly under the prefix and the common prefixes ("folders") below it, which come back as rows whose `key` ends with the delimiter and whose `size`, `last_modified` and `etag` are `NA`.
`config`	An `s3_config()`.

Value

A tibble with one row per object: `key`, `size` (bytes), `last_modified` (UTC) and `etag`. Zero rows when nothing matches.

Examples

```
list_s3("data/", bucket = "my-bucket")
list_s3("s3://my-bucket/series/")
list_s3("series/", bucket = "my-bucket", delimiter = "/")   # one level
```

presign_s3 *A pre-signed URL*

Description

A URL anyone can use for `expires` seconds, without credentials, to read (`GET`) or write (`PUT`) the object. The signature is in the query string (AWS Signature Version 4).

Usage

```
presign_s3(
  key,
  bucket = NULL,
  expires = 3600,
  method = c("GET", "PUT"),
  config = s3_config()
)
```

Arguments

`key`	Object key inside `bucket`, or an `s3://bucket/key` URL.
`bucket`	Bucket; defaults to the config's bucket (`S3PE_BUCKET`).
`expires`	Validity in seconds, from 1 to 604800 (seven days).
`method`	`"GET"` to share a download, `"PUT"` to accept an upload.
`config`	An `s3_config()`.

Value

The URL, a string.

Examples

```
presign_s3("report/report.pdf", bucket = "my-bucket", expires = 3600)
```

read_s3 *Read an object into R*

Description

Downloads an object and parses it according to its extension (or `format`):

Usage

```
read_s3(key, bucket = NULL, ..., format = NULL, config = s3_config())
```

Arguments

key	Object key inside bucket, or an s3://bucket/key URL.
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
...	Passed to the reader (for example col_types for CSV, simplifyVector for JSON).
format	One of "csv", "tsv", "json", "geojson", "parquet", "rds", "xlsx", "text", "raw"; overrides the extension.
config	An s3_config().

Details

Extension	Reader	Result
.csv, .csv.gz	readr::read_csv()	tibble
.tsv, .tsv.gz	readr::read_tsv()	tibble
.json	jsonlite::fromJSON()	list or data frame
.geojson	sf::st_read() if sf is installed, else jsonlite	sf or list
.parquet	nanoparquet::read_parquet()	data frame
.rds	readRDS()	the saved object
.xlsx	readxl::read_excel()	tibble
.txt, .md, .html, .xml, .svg, ...		character (one string)
anything else		raw vector

The byte count is checked against the server's Content-Length and, when the ETag is an MD5, against the MD5 of what arrived.

Value

The parsed object, as in the table.

Examples

```
meta <- read_s3("data/manifest.json", bucket = "my-bucket")
meta$ultimo_mes

sales <- read_s3("s3://my-bucket/series/sales.csv.gz")

# the bytes, whatever the extension
png <- read_s3("report/figure1.png", bucket = "my-bucket", format = "raw")
```

`s3_config` *Configure the connection to the store*

Description

Collects endpoint, region, credentials and default bucket. Anything not given as an argument comes from an option, then from the environment, then from the default:

Usage

```
s3_config(
  endpoint = NULL,
  region = NULL,
  access_key = NULL,
  secret_key = NULL,
  session_token = NULL,
  bucket = NULL,
  timeout = NULL,
  max_tries = NULL,
  multipart_threshold = NULL,
  part_size = NULL
)
```

Arguments

`endpoint`	`scheme://host[:port]`, no path. Missing endpoint is an error when a request is made, not here.
`region`	Signing region; `us-east-1` is what most self-hosted stores (MinIO, Garage, Ceph) expect.
`access_key`, `secret_key`	Credentials. Missing credentials are an error when a request is made, not here.
`session_token`	Session token of temporary credentials.
`bucket`	Bucket used when a key is not an `s3://bucket/key` URL.
`timeout`	Time limit of one call, in seconds.
`max_tries`	Attempts per call (the first one included).
`multipart_threshold`	Bodies larger than this many bytes are uploaded in parts (`write_s3()`, `upload_s3()`).
`part_size`	Size of each part, in bytes (at least 5 MiB).

Details

Setting	Option	Environment	Default
`endpoint`	`s3pe.endpoint`	`S3PE_ENDPOINT`, `AWS_ENDPOINT_URL`	required
`region`	`s3pe.region`	`AWS_REGION`, `AWS_DEFAULT_REGION`	`us-east-1`

access_key		AWS_ACCESS_KEY_ID	required
secret_key		AWS_SECRET_ACCESS_KEY	required
session_token		AWS_SESSION_TOKEN	none
bucket	s3pe.bucket	S3PE_BUCKET	none
timeout	s3pe.timeout		300 s
max_tries	s3pe.max_tries		3
multipart_threshold	s3pe.multipart_threshold	S3PE_MULTIPART_THRESHOLD	64 MiB
part_size	s3pe.part_size	S3PE_PART_SIZE	16 MiB (at least 5 MiB)

Keep credentials in `~/.Renviron` (R reads it at start-up), never in scripts. The endpoint has no default: set it once per machine (`S3PE_ENDPOINT`) or per session (`use_endpoint()`). URLs are always path-style (`endpoint/bucket/key`), which every S3-compatible store accepts.

Value

An object of class `s3pe_config`. Printing it never shows the secret key.

Examples

```
s3_config()
cfg <- s3_config(endpoint = "https://s3.example.org", bucket = "my-bucket")
list_s3("data/", config = cfg)
```

`sync_up_s3` *Mirror a directory to a prefix, or a prefix to a directory*

Description

`sync_up_s3()` uploads every file under `dir` that is missing from the prefix or differs from it; `sync_down_s3()` downloads every object under the prefix that is missing locally or differs. "Differs" is decided by size and ETag: the local file's MD5, or its multipart ETag when it is larger than the config's `multipart_threshold` (computed with the same `part_size`, so files this package uploaded compare equal).

Usage

```
sync_up_s3(
  dir,
  prefix,
  bucket = NULL,
  delete = FALSE,
  last = NULL,
  config = s3_config()
)

sync_down_s3(prefix, dir, bucket = NULL, delete = FALSE, config = s3_config())
```

Arguments

dir	Local directory (created by sync_down_s3() if needed).
prefix	Key prefix, or an s3://bucket/prefix URL. A trailing / is added if missing.
bucket	Bucket; defaults to the config's bucket.
delete	Remove what the source does not have?
last	Relative path of the file to upload last (sync_up_s3() only).
config	An s3_config().

Details

With delete = TRUE, what exists on the destination but not on the source is removed (remote objects in one batch; local files with unlink()). last names the file to upload after every other, so a manifest is there only when the batch is complete.

Value

Invisibly, a list of class s3pe_sync with transferred, skipped and deleted (relative paths) and elapsed (seconds).

Examples

```
sync_up_s3("output/", "reports/2026/", bucket = "my-bucket", last = "manifest.json")
sync_down_s3("s3://my-bucket/reports/2026/", "reports-2026/")
```

---

upload_s3 *Upload a file*

Description

Streams a file to the store (two passes over the file: one to hash it, one to send it). The server's ETag is compared with the file's MD5. Files larger than the config's multipart_threshold (64 MiB by default) go up as a multipart upload, read part by part, aborted on any failure.

Usage

```
upload_s3(
  path,
  key = NULL,
  bucket = NULL,
  content_type = NULL,
  metadata = NULL,
  headers = NULL,
  config = s3_config()
)
```

Arguments

path	File to upload.
key	Object key, or an s3://bucket/key URL. NULL, "" or a key ending in / gets the file's name appended.
bucket	Bucket; defaults to the config's bucket (S3PE_BUCKET).
content_type	Content type to store with the object.
metadata	Named character vector of user metadata, stored with the object as x-amz-meta-<name> and returned by head_s3().
headers	Named character vector of extra headers to send with the upload: Cache-Control, Content-Disposition, Content-Encoding, x-amz-storage-class, x-amz-server-side-encryptio ... (x-amz-* headers are signed).
config	An s3_config().

Value

Invisibly, a one-row tibble: bucket, key, size, etag, elapsed.

Examples

```
upload_s3("output/panel.parquet", "etl/", bucket = "my-bucket")
upload_s3("figure1.png", "s3://my-bucket/report/figure1.png")
upload_s3("report.pdf", "public/", bucket = "my-bucket",
          headers = c("Content-Disposition" = "inline"))
```

---

use_bucket *Set the default bucket or endpoint for the session*

Description

Shortcuts for options(s3pe.bucket =) and options(s3pe.endpoint =): after use_bucket("my-bucket") every verb accepts bare keys (read_s3("data/manifest.json")); after use_endpoint("https://s3.example.org") requests go to that store for the rest of the session. NULL clears the setting.

Usage

```
use_bucket(bucket)

use_endpoint(endpoint)
```

Arguments

bucket	Bucket name, or NULL to clear.
endpoint	scheme://host[:port], or NULL to go back to the environment's S3PE_ENDPOINT.

Value

The previous value, invisibly.

Examples

```
use_endpoint("https://s3.example.org")
use_bucket("my-bucket")
read_s3("data/manifest.json")
```

`write_s3` *Write an R object to the store*

Description

Serialises `x` according to the key's extension (or `format`) and uploads it:

Usage

```
write_s3(
  x,
  key,
  bucket = NULL,
  ...,
  format = NULL,
  content_type = NULL,
  metadata = NULL,
  headers = NULL,
  config = s3_config()
)
```

Arguments

`x`	The object to write.
`key`	Object key inside `bucket`, or an `s3://bucket/key` URL.
`bucket`	Bucket; defaults to the config's bucket (`S3PE_BUCKET`).
`...`	Passed to the writer (for example `pretty = TRUE` for JSON, `na = ""` for CSV).
`format`	One of `"csv"`, `"tsv"`, `"json"`, `"geojson"`, `"parquet"`, `"rds"`, `"xlsx"`, `"text"`, `"raw"`; overrides the extension.
`content_type`	Content type to store with the object.
`metadata`	Named character vector of user metadata, stored with the object as `x-amz-meta-<name>` and returned by `head_s3()`.
`headers`	Named character vector of extra headers to send with the upload: `Cache-Control`, `Content-Disposition`, `Content-Encoding`, `x-amz-storage-class`, `x-amz-server-side-encryptio ...` (`x-amz-*` headers are signed).
`config`	An `s3_config()`.

Details

Extension	Writer	Accepts
`.csv`, `.csv.gz`	`readr::write_csv()`	data frame
`.tsv`, `.tsv.gz`	`readr::write_tsv()`	data frame
`.json`	`jsonlite::toJSON()` (`auto_unbox = TRUE`, `digits = NA`)	anything jsonlite can encode
`.geojson`	`sf::st_write()` for sf objects, else jsonlite	sf or list
`.parquet`	`nanoparquet::write_parquet()`	data frame
`.rds`	`saveRDS()`	anything
`.txt`, `.md`, `.html`, ...		character (joined by newlines)
anything else		raw vector, or character as text

The content type is set from the extension unless `content_type` is given. After the upload the server's ETag is compared with the MD5 of what was sent. Bodies larger than the config's `multipart_threshold` (64 MiB by default) go up as a multipart upload, part by part, aborted on any failure.

Value

Invisibly, a one-row tibble: `bucket`, `key`, `size` (bytes), `etag`, `elapsed` (seconds).

Examples

```
write_s3(mtcars, "report/mtcars.csv", bucket = "my-bucket")
write_s3(list(generated_at = Sys.time()), "report/manifest.json", bucket = "my-bucket")
write_s3(readBin("figure1.png", "raw", 1e7), "s3://my-bucket/report/figure1.png")
write_s3(mtcars, "report/mtcars.csv", bucket = "my-bucket",
         metadata = c(author = "ana"), headers = c("Cache-Control" = "max-age=3600"))
```

Index

buckets_s3, 2

copy_s3, 3

delete_s3, 3
download_s3, 4

exists_s3, 5

head_s3, 5
head_s3(), *12, 13*

jsonlite::fromJSON(), *8*
jsonlite::toJSON(), *14*

list_s3, 6

presign_s3, 7

read_s3, 7
readr::read_csv(), *8*
readr::read_tsv(), *8*
readr::write_csv(), *14*
readr::write_tsv(), *14*
readRDS(), *8*

s3_config, 9
s3_config(), *2–8*, *11–13*
saveRDS(), *14*
sync_down_s3 *(*sync_up_s3*)*, 10
sync_up_s3, 10

unlink(), *11*
upload_s3, 11
upload_s3(), *9*
use_bucket, 12
use_endpoint *(*use_bucket*)*, 12
use_endpoint(), *10*

write_s3, 13
write_s3(), *9*