Package 'minioclient'

May 8, 2026

Title Interface to the 'MinIO' Client

Version 0.0.6

Description An R interface to the 'MinIO' Client. The 'MinIO' Client ('mc') provides a modern alternative to UNIX commands like 'ls', 'cat', 'cp', 'mirror', 'diff', 'find' etc. It supports 'filesystems' and Amazon ``S3'' compatible cloud storage service (``AWS'' Signature v2 and v4). This package provides convenience functions for installing the 'MinIO' client and running any operations, as described in the official documentation, <https://min.io/docs/minio/linux/reference/minio-mc.html?ref=docs-redirect>. This package provides a flexible and high-performance alternative to 'aws.s3'.

License MIT + file LICENSE

URL https://github.com/cboettig/minioclient, https://cboettig.github.io/minioclient/

BugReports https://github.com/cboettig/minioclient/issues

Encoding UTF-8

RoxygenNote 7.2.3

Suggests spelling, testthat (>= 3.0.0), curl

Config/testthat/edition 3

Imports fs, glue, tools, utils, processx, jsonlite

Language en-US

NeedsCompilation no

Author Carl Boettiger [aut, cre] (ORCID: <https://orcid.org/0000-0002-1642-628X>), Markus Skyttner [ctb]

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2023-11-07 10:10:02 UTC

Contents

`install_mc` *install the mc client*

Description

install the mc client

Usage

```
install_mc(
  os = system_os(),
  arch = system_arch(),
  path = minio_path(),
  force = FALSE
)
```

Arguments

`os`	operating system
`arch`	architecture
`path`	destination where binary is installed.
`force`	install even if binary is already found. Can be used to force upgrade.

Details

This function is just a convenience wrapper for prebuilt MINIO binaries, from https://dl.min.io/client/mc/release/. Should support Windows, Mac, and Linux on both Intel/AMD (amd64) and ARM architectures. For details, see official MINIO docs for your operating system, e.g. https://min.io/docs/minio/macos/index.html.

NOTE: If you want to install to other than the default location, simply set the option "minioclient.dir", to the appropriate location of the directory containing your "mc" binary, e.g. `options("minioclient.dir" = "~/.mc")`. This is also used as the location of the config directory. Note that this package will not automatically use MINIO available on $PATH (to promote security and portability in design).

Value

path to the minio binary (invisibly)

Examples

```
install_mc()

# Force upgrade
install_mc(force=TRUE)
```

`mc` *mc*

Description

The MINIO Client

Usage

```
mc(command, ..., path = minio_path(), verbose = interactive())
```

Arguments

`command`	space-delimited text string of an mc command (starting after the mc ...)
`...`	additional arguments to `processx::run()`
`path`	location where mc executable will be installed. By default will use the OS-appropriate storage location.
`verbose`	print output?

Details

This function forms the basis for all other available commands. This utility can run any `mc` command supported by the official minio client, see https://min.io/docs/minio/linux/reference/minio-mc.html. The R package provides wrappers only for the most common use cases, which provide a more natural R syntax and native documentation.

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

`mc_alias_ls` *List all configured aliases*

Description

List all configured aliases

Usage

```
mc_alias_ls(alias = "")
```

Arguments

`alias`	optional argument, display only specified alias

Details

Note that all available

Value

Configured aliases, including secret keys!

See Also

mc

`mc_alias_set` *mc alias set*

Description

Set a new alias for the minio client, possibly using env var defaults.

Usage

```
mc_alias_set(
  alias = "minio",
  endpoint = Sys.getenv("AWS_S3_ENDPOINT", "s3.amazonaws.com"),
  access_key = Sys.getenv("AWS_ACCESS_KEY_ID"),
  secret_key = Sys.getenv("AWS_SECRET_ACCESS_KEY"),
  scheme = "https"
)
```

Arguments

alias	a short name for this endpoint, default is minio
endpoint	the endpoint domain name
access_key	access key (user), reads from AWS env vars by default
secret_key	secret access key, reads from AWS env vars by default
scheme	https or http (e.g. for local machine only)

Value

Returns the list from processx::run(), with components status, stdout, stderr, and timeout; invisibly.

References

https://min.io/docs/minio/linux/reference/minio-mc.html. Note that keys can be omitted for anonymous use.

Examples

```
mc_alias_set()
```

---

mc_anonymous_set *Set anonymous access policy*

Description

This function uses the mc command to set the anonymous access policy for a specified target.

Usage

```
mc_anonymous_set(
  target,
  policy = c("download", "upload", "public", "private"),
  verbose = interactive()
)
```

Arguments

target	Character string specifying the target cloud storage bucket or object
policy	Character string specifying the anonymous access policy. Must be one of "download", "upload", "public" (upload and download), or "private".
verbose	print output?

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

Examples

```
# create a test bucket on the 'play' server
mc_mb("play/minioclient-test")

# Set anonymous access policy to download
mc_anonymous_set("play/minioclient-test/file.txt", policy = "download")

# Set anonymous access policy to upload
mc_anonymous_set("play/minioclient-test/directory", policy = "upload")

# Set anonymous access policy to public
mc_anonymous_set("play/minioclient-test/file.txt", policy = "public")

# Set anonymous access policy to private (default policy for new buckets)
mc_anonymous_set("play/minioclient-test/directory", policy = "private")

mc_rb("play/minioclient-test")
```

---

mc_cat *Display object contents*

Description

The cat command returns the contents of the object as a string. This can be useful when reading smaller files (without first downloading to disk).

Usage

```
mc_cat(target, offset = 0, tail = 0, flags = "")
```

Arguments

`target`	character string specifying the target directory path.
`offset`	start offset, default 0 if not specified
`tail`	tail number of bytes at ending of file, default 0 if not specified
`flags`	additional flags to be passed to the `cat` command. Default is an empty string.

Value

a character string with the contents of the file

Examples

```
# upload a file to a bucket and read it back
install_mc()
mc_mb("play/mcr")
mc_cp(system.file(package = "minioclient", "DESCRIPTION"), "play/mcr/DESCRIPTION")
mc_cat("play/mcr/DESCRIPTION")
```

`mc_config_set` *mc_config_set*

Description

Edit the config files, e.g. to add a sessionToken

Usage

```
mc_config_set(alias, key, value, json = file.path(minio_path(), "config.json"))
```

Arguments

`alias`	A configured alias, see `mc_alias_set()`
`key`	the parameter name, e.g. `sessionToken`
`value`	the value to set the parameter to
`json`	path to the config

Value

updates configuration and returns silently (`NULL`).

Examples

```
mc_config_set("play", key="sessionToken", value="MyTmpSessionToken")
```

mc_cp *Copy files or directories between servers*

Description

Most commonly used to upload and download files between local filesystem and remote S3 store.

Usage

```
mc_cp(from, to = "", recursive = FALSE, flags = "", verbose = FALSE)
```

Arguments

from	Character string specifying the source file or directory path. Can accept a vector of file paths as well.
to	Character string specifying the destination path.
recursive	Logical indicating whether to recursively copy directories. Default is FALSE.
flags	any additional flags to cp
verbose	Logical indicating whether to report files copied. Default is FALSE.

Details

see `mc("cp -h")` for details.

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

See Also

`mc_mirror`

Examples

```
# Copy a file
mc_cp("local/path/to/file.txt", "alias/bucket/path/file.txt")

# Copy a directory recursively
mc_cp("local/directory", "alias/bucket/path/to/directory", recursive = TRUE)
```

mc_du *Show disk usage for a target path*

Description

Show disk usage for a target path

Usage

```
mc_du(target, flags = "")
```

Arguments

target	alias/bucket to list
flags	optional additional flags

Details

for more help, run `mc_du("-h")`

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

Examples

```
# create a new bucket
mc_mb("play/minioclient-test")

# no disk usage on new bucket
mc_du("play/minioclient-test")

# clean up
mc_rb("play/minioclient-test")
```

mc_head *Display first few lines of an object*

Description

The head command returns the first n lines of the object as a string. This can be useful when inspecting the content of a large file (without first having to download and store it on disk locally).

Usage

```
mc_head(target, n = 10, flags = "")
```

Arguments

target	character string specifying the target directory path.
n	integer number of lines to read from the beginning, by default 10
flags	additional flags to be passed to the cat command. Default is an empty string.

Value

a character string with the contents of the file

Examples

```
# upload a CSV file
install_mc()
tf <- tempfile()
write.csv(iris, tf, row.names = FALSE)
mc_mb("play/iris")
mc_cp(tf, "play/iris/iris.csv")

# read first 13 lines from the CSV (header + 12 rows of data)
read.csv(text = mc_head("play/iris/iris.csv", n = 13))
```

mc_ls *List files and directories using mc command*

Description

This function uses the mc command to list files and directories at the specified target location.

Usage

```
mc_ls(target, recursive = FALSE, details = FALSE)
```

Arguments

target	character vector specifying the target directory path(s).
recursive	Logical indicating whether to recursively list directories. Default is FALSE.
details	logical, by default FALSE; if TRUE a data frame with details for the directory listing is returned.

Value

a vector of file or directory names ("keys" in minio parlance) or, if details is TRUE, a data.frame with the directory listing information

Examples

```
# list all buckets on play server
mc_ls("play/")
mc_ls("play", details = TRUE)
```

---

mc_mb *Create a new S3 bucket using mc command*

Description

Create a new S3 bucket using mc command

Usage

```
mc_mb(bucket, ignore_existing = TRUE, flags = "", verbose = TRUE)
```

Arguments

bucket	Character string specifying the name of the bucket to create.
ignore_existing	do not error if bucket already exists
flags	additional flags, see mc_mb("-h") for details.
verbose	print output?

Value

Returns the list from processx::run(), with components status, stdout, stderr, and timeout; invisibly.

Examples

```
# Create a new bucket named "my-bucket"
mc_mb("play/my-bucket")
```

mc_mirror *Mirror files and directories using mc command*

Description

This function uses the mc command to mirror files and directories from one location to another.

Usage

```
mc_mirror(
  from,
  to,
  overwrite = FALSE,
  remove = FALSE,
  flags = "",
  verbose = FALSE
)
```

Arguments

from	Character string specifying the source file or directory path.
to	Character string specifying the destination path.
overwrite	Logical indicating whether to overwrite existing files. Default is FALSE.
remove	Logical indicating whether to remove extraneous files from the destination. Default is FALSE.
flags	Additional flags to be passed to the mirror command. Default is an empty string.
verbose	Logical indicating whether to display verbose output. Default is FALSE.

Value

Returns the list from processx::run(), with components status, stdout, stderr, and timeout; invisibly.

Examples

```
# Mirror files and directories from source to destination
mc_mirror("path/to/source", "path/to/destination")

# Mirror files and directories with overwrite and remove options
mc_mirror("path/to/source", "path/to/destination",
           overwrite = TRUE, remove = TRUE)

# Mirror files and directories with additional flags and verbose output
mc_mirror("path/to/source", "path/to/destination",
          flags = "--exclude '*.txt'", verbose = TRUE)
```

mc_mv *move or rename files or directories between servers*

Description

move or rename files or directories between servers

Usage

```
mc_mv(from, to, recursive = FALSE, flags = "", verbose = FALSE)
```

Arguments

from	Character string specifying the source file or directory path. Can accept a vector of file paths as well.
to	Character string specifying the destination path.
recursive	Logical indicating whether to recursively move directories. Default is FALSE.
flags	any additional flags to mv
verbose	Logical indicating whether to report files copied. Default is FALSE.

Details

see `mc("mv -h")` for details.

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

See Also

mc_cp

Examples

```
# move a file
mc_mv("local/path/to/file.txt", "alias/bucket/path/file.txt")

# move a directory recursively
mc_mv("local/directory", "alias/bucket/path/to/directory", recursive = TRUE)
```

mc_rb *Remove an S3 bucket using mc command*

Description

Remove an S3 bucket using mc command

Usage

```
mc_rb(bucket, force = FALSE)
```

Arguments

bucket	Character string specifying the name of the bucket to remove
force	Delete bucket without confirmation in non-interactive mode

Value

Returns the list from processx::run(), with components status, stdout, stderr, and timeout; invisibly.

Examples

```
# Create a new bucket named "my-bucket" on the "play" system
mc_mb("play/my-bucket")
mc_rb("play/my-bucket")
```

mc_rm *Remove files or directories*

Description

This function uses the mc command to remove files or directories at the specified target location.

Usage

```
mc_rm(target, recursive = FALSE, flags = "", verbose = FALSE)
```

Arguments

target	Character string specifying the target file or directory path to be removed.
recursive	Logical indicating whether to recursively remove directories. Default is FALSE.
flags	Additional flags to be passed to the rm command. Default is an empty string.
verbose	Logical indicating whether to list files removed. Default is FALSE.

Details

see `mc("rm -h")` for details.

Value

Returns the list from `processx::run()`, with components `status`, `stdout`, `stderr`, and `timeout`; invisibly.

Examples

```
# Remove a file
mc_rm("path/to/file.txt")

# Remove a directory recursively
mc_rm("path/to/directory", recursive = TRUE)
```

`mc_sql` *Use the S3 variant of SQL to query a minio object*

Description

The S3 Select API can be used against CSV and JSON objects stored in minio. If the minio server runs with MINIO_API_SELECT_PARQUET=on, also parquet files can be queried.

Usage

```
mc_sql(
  target,
  query = "select * from S3Object",
  recursive = TRUE,
  verbose = FALSE
)
```

Arguments

`target`	character alias or path specification at minio for the object (a .csv, .json or .parquet file)
`query`	character string with sql query, by default "select * from S3Object"
`recursive`	logical, by default TRUE, allowing a s3 select query to work across a minio ALIAS/PATH specification
`verbose`	logical, by default FALSE

Details

See https://min.io/docs/minio/linux/reference/minio-mc/mc-sql.html# and https://github.com/minio/minio/blob/master/docs/select/README.md

For example "select s.* from S3Object s limit 10" is valid syntax.

More examples of query syntax here: https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-select-sql-reference-select.html

Value

SQL query results as a `data.frame` of class `tbl_df`

Examples

```
install_mc()
# upload a CSV file
tf <- tempfile()
write.csv(iris, tf, row.names = FALSE)
mc_mb("play/iris")
mc_cp(tf, "play/iris/iris.csv")

# read first 12 lines from the CSV
mc_sql("play/iris/iris.csv", query = "select * from S3Object limit 12")
```

Index