Package ‘sergeant’

November 29, 2021

Title Tools to Transform and Query Data with Apache Drill

Version 0.9.1

Description Apache Drill is a low-latency distributed query engine designed to enable data exploration and analysis on both relational and non-relational data stores, scaling to petabytes of data. Methods are provided that enable working with Apache Drill instances via the REST API, DBI methods and using ‘dplyr'/dbplyr' idioms. Helper functions are included to facilitate using official Drill Docker images/containers.

Depends R (>= 3.6.0)

URL https://gitlab.com/hrbrmstr/sergeant

BugReports https://gitlab.com/hrbrmstr/sergeant/-/issues

License MIT + file LICENSE

Encoding UTF-8

Imports bit64 (>= 0.9-7), DBI (>= 0.7), dplyr (>= 0.8.0), dbplyr (>= 1.3.0), htr (>= 1.2.1), jsonlite (>= 1.5.0), htmltools (>= 0.3.6), readr (>= 1.1.1), purrr (>= 0.2.2), scales (>= 0.4.1), stringi, tibble, utils, methods, magrittr (>= 1.5)

Suggests DT (>= 0.5), stevedore, tinytest, covr (>= 3.0.0), DBItest

RoxygenNote 7.1.2

NeedsCompilation no

Author Bob Rudis [aut, cre] (<https://orcid.org/0000-0001-5670-2640>), Edward Visel [ctb], Andy Hine [ctb], Scott Came [ctb], David Severski [ctb] (<https://orcid.org/0000-0001-7867-0459>), James Lamb [ctb]

Maintainer Bob Rudis <bob@rud.is>

Repository CRAN

Date/Publication 2021-11-29 18:40:02 UTC
ctas_profile

Generate a Drill CTAS Statement from a Query

Description

When working with CSV[H] files in Drill 1.15.0+ everything comes back VARCHAR since that’s the way it should be. The old behaviour of sergeant to auto-type convert was kinda horribad wrong. However, it’s a royal pain to make CTAS queries from a giant list of VARCHAR field by hand. So, this is a helper function to do that, inspired by David Severski.
**Usage**

```
ctas_profile(x, new_table_name = "CHANGE____ME")
```

**Arguments**

- `x` a tbl
- `new_table_name` a new Drill data source spec (e.g. `dfs.xyz.`'a.parquet`)

**Note**

WIP!

**Examples**

```r
## Not run:
db <- src_drill("localhost")
# Test with bare data source
flt1 <- tbl(db, "dfs.d."/'flights.csv"')
cat(ctas_profile(flt1))

# Test with SELECT
flt2 <- tbl(db, sql("SELECT 'year', tailnum, time_hour FROM dfs.d."/'flights.csv"'))
cat(ctas_profile(flt2, "dfs.d."/flights.parquet"'))

## End(Not run)
```

---

**Description**

Drill `dbDataType`

**Usage**

```r
## S4 method for signature 'DrillConnection'
dbDataType(dbObj, obj, ...)
```

**Arguments**

- `dbObj` A `DrillDriver` object
- `obj` Any R object
- `...` Extra optional parameters
See Also

Other Drill REST DBI API: `DrillConnection-class, DrillDriver-class, DrillResult-class, Drill(), dbUnloadDriver, DrillDriver-method`

---

`dbGetInfo, DrillDriver-method`

**Metadata about database objects**

### Description

Metadata about database objects

### Usage

```r
## S4 method for signature 'DrillDriver'
dbGetInfo(dbObj)
```

```r
## S4 method for signature 'DrillConnection'
dbGetInfo(dbObj)
```

### Arguments

- `dbObj` A `DrillDriver` or `DrillConnection` object

---

`dbUnloadDriver, DrillDriver-method`

**Unload driver**

### Description

Unload driver

### Usage

```r
## S4 method for signature 'DrillDriver'
dbUnloadDriver(drv, ...)
```

### Arguments

- `drv` driver
- `...` Extra optional parameters

### See Also

Other Drill REST DBI API: `DrillConnection-class, DrillDriver-class, DrillResult-class, Drill(), dbDataType, DrillConnection-method`
Drill

Description

Drill

Connect to Drill

Usage

Drill()

```r
## S4 method for signature 'DrillDriver'
dbConnect(
  drv,
  host = "localhost",
  port = 8047L,
  ssl = FALSE,
  username = NULL,
  password = NULL,
  ...
)
```

Arguments

- `drv` An object created by `Drill()`
- `host` host
- `port` port
- `ssl` use ssl?
- `username, password` credentials
- `...` Extra optional parameters

See Also

Other Drill REST DBI API: `DrillConnection-class, DrillDriver-class, DrillResult-class, dbDataType, DrillConnection-method, dbUnloadDriver, DrillDriver-method`

Other Drill REST DBI API: `DrillConnection-class, DrillDriver-class, DrillResult-class, dbDataType, DrillConnection-method, dbUnloadDriver, DrillDriver-method`
### drill_active

**Test whether Drill HTTP Drill direct REST API Interface server is up**

**Description**

This is a very simple test (performs `HEAD /` request on the Drill server/cluster)

**Usage**

```r
drill_active(drill_con)
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`

**See Also**

Other Drill direct REST API Interface: `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```r
## Not run:
drill_connection() %>% drill_active()

## End(Not run)
```

### drill_cancel

**Cancel the query that has the given queryid**

**Description**

Cancel the query that has the given queryid

**Usage**

```r
drill_cancel(drill_con, query_id)
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`
- `query_id`: the UUID of the query in standard UUID format that Drill assigns to each query.
drill_connection

References

Drill documentation

See Also

Other Drill direct REST API Interface: drill_active(), drill_connection(), drill_functions(), drill_metrics(), drill_options(), drill_opts(), drill_profiles(), drill_profile(), drill_query(), drill_settings_reset(), drill_set(), drill_stats(), drill_status(), drill_storage(), drill_system_reset(), drill_threads(), drill_version()

---

### drill_connection

#### Setup a Drill connection

**Description**

Setup a Drill connection

**Usage**

```r
drill_connection(
  host = Sys.getenv("DRILL_HOST", "localhost"),
  port = Sys.getenv("DRILL_PORT", 8047),
  ssl = FALSE,
  user = Sys.getenv("DRILL_USER", ""),
  password = Sys.getenv("DRILL_PASSWORD", "")
)
```

**Arguments**

- **host** Drill host (will pick up the value from DRILL_HOST env var)
- **port** Drill port (will pick up the value from DRILL_PORT env var)
- **ssl** use ssl?
- **user, password** (will pick up the values from DRILL_USER/DRILL_PASSWORD env vars)

**Note**

If user/password are set this function will make a POST to the REST interface immediately to prime the cookie-jar with the session id.

**See Also**

Other Drill direct REST API Interface: drill_active(), drill_cancel(), drill_functions(), drill_metrics(), drill_options(), drill_opts(), drill_profiles(), drill_profile(), drill_query(), drill_settings_reset(), drill_set(), drill_stats(), drill_status(), drill_storage(), drill_system_reset(), drill_threads(), drill_version()
Examples

dc <- drill_connection()

---

drill_custom_functions

*Drill expressions / custom functions dplyr translations*

Description

One benefit of `dplyr` is that it provide a nice DSL over database ops but that means there needs to be knowlege of functions supported by the host database and then a translation layer so they can be used in R.

Details

Similarly, there are functions like `grepl()` in R that don’t directly exist in databases. Yet, one can create a translation for `grepl()` that maps to a [Drill custom function](#) so you don’t have to think differently or rewrite your pipes when switching from core tidyverse ops and database ops.

Many functions translate on their own, but it’s handy to provide explicit ones, especially when you want to use parameters in a different order.

If you want a particular custom function mapped, file a PR or issue request in the link found in the DESCRIPTION file.

- `as.character(x)`: `CAST( x AS CHARACTER )`
- `as.integer64(x)`: `CAST( x AS BIGINT )`
- `as.date(x)`: `CAST( x AS DATE )`
- `as.logical(x)`: `CAST( x AS BOOLEAN)`
- `as.numeric(x)`: `CAST( x AS DOUBLE )`
- `as.posixct(x)`: `CAST( x AS TIMESTAMP )`
- `binary_string(x)`: `BINARY_STRING( x )`
- `cbrt(x)`: `CBRT( x )`
- `char_to_timestamp(x,y)`: `TO_TIMESTAMP( x,y )`
- `grepl(y,x)`: `CONTAINS( x,y )`
- `contains(x,y)`: `CONTAINS( x,y )`
- `convert_to(x,y)`: `CONVERT_TO( x,y )`
- `convert_from(x,y)`: `CONVERT_FROM( x,y )`
- `degrees(x)`: `DEGREES( x )`
- `lshift(x,y)`: `DEGREES( x,y )`
- `negative(x)`: `NEGATIVE( x )`
- `pow(x,y)`: `MOD( x,y )`
- `sql_prefix(x,y)`: `POW( x,y )`
- `string_binary(x)` \: `STRING_BINARY(x)`
- `radians(x)` \: `RADIANS(x)`
- `rshift(x)` \: `RSHIFT(x)`
- `to_char(x,y)` \: `TO_CHAR(x,y)`
- `to_date(x,y)` \: `TO_DATE(x,y)`
- `to_number(x,y)` \: `TO_NUMBER(x,y)`
- `trunc(x)` \: `TRUNC(x)`
- `double_to_timestamp(x)` = `TO_TIMESTAMP(x)`
- `char_length(x)` = `CHAR_LENGTH(x)`
- `flattten(x)` = `FLATTEN(x)`
- `kvgen(x)` = `KVGEN(x)`
- `repeated_count(x)` = `REPEATED_COUNT(x)`
- `repeated_contains(x)` = `REPEATED_CONTAINS(x)`
- `ilike(x,y)` = `ILIKE(x,y)`
- `init_cap(x)` = `INIT_CAP(x)`
- `length(x)` = `LENGTH(x)`
- `lower(x)` = `LOWER(x)`
- `tolower(x)` = `LOWER(x)`
- `ltrim(x,y)` = `LTRIM(x,y)`
- `nullif(x,y)` = `NULLIF(x,y)`
- `position(x,y)` = `POSITION(x IN y)`
- `gsub(x,y,z)` = `REGEXP_REPLACE(z,x,y)`
- `regexp_replace(x,y,z)` = `REGEXP_REPLACE(x,y,z)`
- `rtrim(x,y)` = `RTRIM(x,y)`
- `rpad(x,y)` = `RPAD(x,y)`
- `rpad_with(x,y,z)` = `RPAD(x,y,z)`
- `lpad(x,y)` = `LPAD(x,y)`
- `lpad_with(x,y,z)` = `LPAD(x,y,z)`
- `strpos(x,y)` = `STRPOS(x,y)`
- `substr(x,y,z)` = `SUBSTR(x,y,z)`
- `upper(x)` = `UPPER(x)`
- `toupper(x)` = `UPPER(x)`

You can get a compact list of these with:
```
sql_translate_env(src_drill()$con)
```
as well.

**See Also**

Other Drill REST API (dplyr): `src_drill()`, `src_tbls.src_drill()`
**drill_functions**

Show all the available Drill built-in functions & UDFs

**Usage**

```r
drill_functions(drill_con, browse = FALSE)
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`
- `browse`: if TRUE display an HTML interactive HTML widget with the functions as well as return the data frame with the functions. Default if FALSE.

**Value**

data frame

**Note**

You must be using Drill 1.15.0+ to use this function

**References**

Drill documentation

**See Also**

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```r
## Not run:
drill_connection() %>% drill_functions()
## End(Not run)
```
### drill_metrics

**Get the current memory metrics**

**Description**

Get the current memory metrics

**Usage**

```
drill_metrics(drill_con)
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`

**See Also**

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```
## Not run:
drill_connection() %>% drill_metrics()

## End(Not run)
```

### drill_options

**List the name, default, and data type of the system and session options**

**Description**

List the name, default, and data type of the system and session options

**Usage**

```
drill_options(drill_con, pattern = NULL)
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`
- `pattern`: pattern to filter results by
References

Drill documentation

See Also

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`,
`drill_functions()`, `drill_metrics()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`,
`drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`,
`drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

Examples

```r
## Not run:
drill_connection() %>% drill_options()
## End(Not run)
```

---

### `drill_opts`

Show all the available Drill options

**Usage**

```r
drill_opts(drill_con, browse = FALSE)
```

**Arguments**

- `drill_con` drill server connection object setup by `drill_connection()`
- `browse` if TRUE display an HTML interactive HTML widget with the options as well as return the data frame with the options Default if FALSE.

**Value**

data frame

**Note**

You must be using Drill 1.15.0+ to use this function

**References**

Drill documentation
drill_profile

See Also

Other Drill direct REST API Interface: drill_active(), drill_cancel(), drill_connection(), drill_functions(), drill_metrics(), drill_options(), drill_profiles(), drill_profile(), drill_query(), drill_settings_reset(), drill_set(), drill_stats(), drill_status(), drill_storage(), drill_system_reset(), drill_threads(), drill_version()

Examples

```r
## Not run:
drill_connection() %>% drill_opts()

## End(Not run)
```

---

### drill_profile

Get the profile of the query that has the given queryid

**Description**

Get the profile of the query that has the given queryid

**Usage**

`drill_profile(drill_con, query_id)`

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`
- `query_id`: UUID of the query in standard UUID format that Drill assigns to each query

**References**

Drill documentation

**See Also**

Other Drill direct REST API Interface: drill_active(), drill_cancel(), drill_connection(), drill_functions(), drill_metrics(), drill_options(), drill_opts(), drill_profiles(), drill_query(), drill_settings_reset(), drill_set(), drill_stats(), drill_status(), drill_storage(), drill_system_reset(), drill_threads(), drill_version()
drill_profiles

Get the profiles of running and completed queries

Description
Get the profiles of running and completed queries

Usage
```
drill_profiles(drill_con)
```

Arguments
```
drill_con      drill server connection object setup by drill_connection()
```

References
Drill documentation

See Also
Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

Examples
```
## Not run:
  drill_connection() %>% drill_profiles()

## End(Not run)
```

---

drill_query

Submit a query and return results

Description
This function can handle REST API connections or JDBC connections. There is a benefit to calling this function for JDBC connections vs a straight call to `dbGetQuery()` in that the function result is a `tbl_df` vs a plain `data.frame` so you get better default printing (which can be helpful if you accidentally execute a query and the result set is huge).

Usage
```
drill_query(drill_con, query, uplift = TRUE, .progress = interactive())
```
**Arguments**

- **drill_con**: drill server connection object setup by `drill_connection()` or `drill_jdbc()`
- **query**: query to run
- **uplift**: automatically run `drill_uplift()` on the result? (default: `TRUE`, ignored if `drill_con` is a JDBCConnection created by `drill_jdbc()`)
- **.progress**: if `TRUE` (default if in an interactive session) then ask `httr::RETRY` to display a progress bar

**References**

Drill documentation

**See Also**

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```r
try(
  drill_connection() %>%
  drill_query("SELECT * FROM cp.'employee.json' limit 5")
), silent=TRUE)
```

---

**drill_set**

*Set Drill SYSTEM or SESSION options*

**Description**

Helper function to make it more R-like to set Drill SESSION or SYSTEM options. It handles the conversion of R types (like `TRUE`) to SQL types and automatically quotes parameter values (when necessary).

**Usage**

```r
drill_set(drill_con, ..., type = c("session", "system"))
```

**Arguments**

- **drill_con**: drill server connection object setup by `drill_connection()`
- **...**: named parameters to be sent to ALTER SYSTEM or ALTER SESSION
- **type**: set the session or system parameter
drill_settings_reset

Details
If any query errors result, error messages will be presented to the console.

Value
a tbl (invisibly) with the ALTER queries sent and results, including errors.

References
Drill documentation

See Also
Other Drill direct REST API Interface: drill_active(), drill_cancel(), drill_connection(), drill_functions(), drill_metrics(), drill_options(), drill_opts(), drill_profiles(), drill_profile(), drill_query(), drill_settings_reset(), drill_stats(), drill_status(), drill_storage(), drill_system_reset(), drill_threads(), drill_version()

Examples
```r
## Not run:
drill_connection() %>%
  drill_set(exec.errors_VERBOSE=TRUE, store.format="parquet", web.logs.max_lines=20000)
## End(Not run)
```

drill_settings_reset  Changes (optionally, all) session settings back to system defaults

Description
Changes (optionally, all) session settings back to system defaults

Usage
drill_settings_reset(drill_con, ...)

Arguments
drill_con  drill server connection object setup by drill_connection()
...  bare name of system options to reset

References
Drill documentation
**See Also**

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```r
## Not run:
drill_connection() %>% drill_settings_reset(exec.errors.verbose)

## End(Not run)
```

---

**drill_show_files**  
*Show files in a file system schema.*

**Description**

Show files in a file system schema.

**Usage**

```r
drill_show_files(drill_con, schema_spec, .progress = interactive())
```

**Arguments**

- `drill_con`: drill server connection object setup by `drill_connection()`
- `schema_spec`: properly quoted "filesystem.directory_name" reference path
- `.progress`: if TRUE (default if in an interactive session) then ask `httr::RETRY` to display a progress bar

**References**

Drill documentation

**See Also**

Other Dill direct REST API Interface: `drill_show_schemas()`, `drill_use()`

**Examples**

```r
try({
    drill_connection() %>%
    drill_show_files("dfs.tmp")
}, silent=TRUE)
```
Finally, the `drill_stats` function provides information about Drillbit, such as port numbers.

**Description**

Get Drillbit information, such as ports numbers

**Usage**

```
drill_stats(drill_con)
```

**Arguments**

- `drill_con` drill server connection object setup by `drill_connection()`

**References**

Drill documentation
## drill_status

Get the status of Drill

### Description

Get the status of Drill

### Usage

```r
drill_status(drill_con)
```

### Arguments

- `drill_con` drill server connection object setup by `drill_connection()`

### Note

The output of this is in a "viewer" window

### See Also

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

### Examples

```r
## Not run:
drill_connection() %>% drill_stats()

## End(Not run)
```


**Description**

Retrieve, modify or remove storage plugins from a Drill instance. If you intend to modify an existing configuration it is suggested that you use the "list" or "raw" values to the as parameter to make it easier to modify them.

**Usage**

```r
drill_storage(drill_con, plugin = NULL, as = c("tbl", "list", "raw"))
drill_mod_storage(drill_con, name, config)
drill_rm_storage(drill_con, name)
```

**Arguments**

- `drill_con` drill server connection object setup by `drill_connection()`
- `plugin` the assigned name in the storage plugin definition.
- `as` one of "tbl" or "list" or "raw". The latter two are useful if you want modify an existing storage plugin (e.g. add a workspace) via `drill_mod_storage()`.
- `name` name of the storage plugin configuration to create/update/remove
- `config` a raw 1-element character vector containing valid JSON of a complete storage spec

**References**

Drill documentation

**See Also**

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_system_reset()`, `drill_threads()`, `drill_version()`

**Examples**

```r
## Not run:
drill_connection() %>% drill_storage()

drill_connection() %>%
  drill_mod_storage(
    name = "drilldat",
    config = '
```
### drill_system_reset

Changes (optionally, all) system settings back to system defaults

#### Description

Changes (optionally, all) system settings back to system defaults

#### Usage

```r
drill_system_reset(drill_con, ..., all = FALSE)
```

#### Arguments

- `drill_con`: drill server connection object setup by `drill_connection()`
- `...`: bare name of system options to reset
- `all`: if TRUE, all parameters are reset (... is ignored)

#### References

Drill documentation

#### See Also

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_threads()`, `drill_version()`
## drill_threads

### Examples

```r
## Not run:
drill_connection() %>% drill_system_reset(all=TRUE)

## End(Not run)
```

### drill_threads

Get information about threads

#### Usage

```r
drill_threads(drill_con)
```

#### Arguments

- `drill_con`: drill server connection object setup by `drill_connection()`

#### Note

The output of this is in a "viewer" window

#### See Also

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_version()`

#### Examples

```r
## Not run:
drill_connection() %>% drill_threads()

## End(Not run)
```
Start a Dockerized Drill Instance

**Description**

This is a "get you up and running quickly" helper function as it only runs a standalone mode Drill instance and is optionally removed after the container is stopped. You should customize your own Drill containers based on the one at [Drill’s Docker Hub](https://hub.docker.com/r/drill/drill/).

**Usage**

```r
drill_up(
  image = "drill/apache-drill:1.16.0",
  container_name = "drill",
  data_dir = getwd(),
  remove = TRUE
)
```

**Arguments**

- **image**: Drill image to use. Must be a valid image from [Drill’s Docker Hub](https://hub.docker.com/r/drill/drill/). Defaults to most recent Drill docker image.
- **container_name**: Name for the container. Defaults to "drill".
- **data_dir**: valid path to a place where your data is stored; defaults to the value of `getwd()`. This will be `path.expand()`ed and mapped to /data in the container. This will be mapped to the dfs storage plugin as the dfs.d workspace.
- **remove**: remove the Drill container instance after it's stopped? Defaults to TRUE since you shouldn’t be relying on this in production.
- **id**: the id of the Drill container

**Details**

The path specified in `data_dir` will be mapped inside the container as /data and a new dfs storage workspace will created (dfs.d) that maps to /data and is writable.

Use `drill_down()` to stop a running Drill container by container id (full or partial).

**Value**

A stevedore docker object (invisibly) which you are responsible for killing with the `$stop()` function or from the Docker command line (in interactive mode the docker container ID is printed as well).

**Note**

This requires a working Docker setup on your system and it is highly suggested you docker pull it yourself before running this function.
See Also

Other Drill Docker functions: `killall_drill()`, `showall_drill()`

Examples

```r
## Not run:
drill_up(data_dir = "~/Data")
## End(Not run)
```

---

### `drill_uplift`

**Turn columnar query results into a type-converted tbl**

**Description**

If you know the result of `drill_query()` will be a data frame, then you can pipe it to this function to pull out rows and automatically type-convert it.

**Usage**

```r
drill_uplift(query_result)
```

**Arguments**

- `query_result` the result of a call to `drill_query()`

**Details**

Not really intended to be called directly, but useful if you accidentally ran `drill_query()` without `uplift=TRUE` but want to then convert the structure.

**References**

- Drill documentation
drill_use

Change to a particular schema.

Description

Change to a particular schema.

Usage

```r
drill_use(drill_con, schema_name, .progress = interactive())
```

Arguments

- **drill_con**: drill server connection object setup by `drill_connection()`.
- **schema_name**: A unique name for a Drill schema. A schema in Drill is a configured storage plugin, such as hive, or a storage plugin and workspace.
- **.progress**: if TRUE (default if in an interactive session) then ask `httr::RETRY` to display a progress bar.

References

Drill documentation

See Also

Other Dill direct REST API Interface: `drill_show_files()`, `drill_show_schemas()`

---

drill_version

Identify the version of Drill running

Description

Identify the version of Drill running

Usage

```r
drill_version(drill_con)
```

Arguments

- **drill_con**: drill server connection object setup by `drill_connection()`

References

Drill documentation
See Also

Other Drill direct REST API Interface: `drill_active()`, `drill_cancel()`, `drill_connection()`, `drill_functions()`, `drill_metrics()`, `drill_options()`, `drill_opts()`, `drill_profiles()`, `drill_profile()`, `drill_query()`, `drill_settings_reset()`, `drill_set()`, `drill_stats()`, `drill_status()`, `drill_storage()`, `drill_system_reset()`, `drill_threads()`

Examples

```r
## Not run:
drill_connection() %>% drill_version()
## End(Not run)
```

---

**format.DrillConnection**

*A concise character representation (label) for a DrillConnection*

---

**Description**

A concise character representation (label) for a DrillConnection

**Usage**

```r
## S3 method for class 'DrillConnection'
format(x, ...)
```

**Arguments**

- `x` a `DrillConnection`
- `...` ignored

---

**killall_drill**

*Prune all dead and running Drill Docker containers*

---

**Description**

*This is a destructive function.* It will stop any Docker container that is based on an image matching a runtime command of "bin/drill-embedded". It’s best used when you had a session forcefully interrupted and had been using the R helper functions to start/stop the Drill Docker container. You may want to consider using the Docker command-line interface to perform this work manually.

**Usage**

```r
killall_drill()
```

**See Also**

Other Drill Docker functions: `drill_up()`, `showall_drill()`
**print.drill_conn**

*Print function for drill_conn objects*

**Description**

Print function for drill_conn objects

**Usage**

```r
## S3 method for class 'drill_conn'
print(x, ...)
```

**Arguments**

- `x`: a drill_conn object made with `drill_connection()`
- `...`: unused

**sergeant-exports**

*sergeant exported operators*

**Description**

The following functions are imported and then re-exported from the sergeant package to enable use of the magrittr pipe operator with no additional library calls

**showall_drill**

*Show all dead and running Drill Docker containers*

**Description**

This function will show all Docker containers that are based on an image matching a runtime command of "bin/drill-embedded".

**Usage**

`showall_drill()`

**See Also**

Other Drill Docker functions: `drill_up()`, `killall_drill()`
Description

Use `src_drill()` to connect to a Drill cluster and `tbl()` to connect to a fully-qualified "table reference". The vast majority of Drill SQL functions have also been made available to the `dplyr` interface. If you have custom Drill SQL functions that need to be implemented please file an issue on GitHub.

Usage

```r
src_drill(
  host = Sys.getenv("DRILL_HOST", "localhost"),
  port = as.integer(Sys.getenv("DRILL_PORT", 8047L)),
  ssl = FALSE,
  username = NULL,
  password = NULL
)
```

## S3 method for class 'src_drill'

`tbl(src, from, ...)`

Arguments

- `host`: Drill host (will pick up the value from `DRILL_HOST` env var)
- `port`: Drill port (will pick up the value from `DRILL_PORT` env var)
- `ssl`: use ssl?
- `username, password`: if not NULL the credentials for the Drill service.
- `src`: A Drill "src" created with `src_drill()`
- `from`: A Drill view or table specification
- `...`: Extra parameters

Note

This is a DBI wrapper around the Drill REST API.

See Also

Other Drill REST API (dplyr): `drill_custom_functions, src_tbls.src_drill()`
Other Drill REST API (dplyr): `drill_custom_functions, src_tbls.src_drill()`
Examples

```r
try({
  db <- src_drill("localhost", 8047L)

  print(db)
  ## src: DrillConnection
  ## tbls: INFORMATION_SCHEMA, cp.default, dfs.default, dfs.root, dfs.tmp, sys

  emp <- tbl(db, "cp.employee.json")

  count(emp, gender, marital_status)
  ## Source: lazy query [?? x 3]
  ## Database: DrillConnection
  ## Groups: gender
  ## marital_status gender n
  ## <chr> <chr> <int>
  ## 1 S F 297
  ## 2 M M 278
  ## 3 S M 276

  # Drill-specific SQL functions are also available
  select(emp, full_name) %>%
    mutate(      
      loc = strpos(full_name, "a"),
      first_three = substr(full_name, 1L, 3L),
      len = length(full_name),
      rx = regexp_replace(full_name, "[aeiouAEIOU]", "*"),
      rnd = rand(),
      pos = position("en", full_name),
      rpd = rpad(full_name, 20L),
      rpdw = rpad_with(full_name, 20L, "*"))
  ## Source: lazy query [?? x 9]
  ## Database: DrillConnection
  ## loc full_name len rpdw pos rx
  ## <int> <chr> <int> <chr> <int> <chr>
  ## 1 0 Sheri Nowmer 12 Sheri Nowmer******** 0 Sh*r* N*wmer
  ## 2 0 Derrick Whelply 15 Derrick Whelply***** 0 D*rr*ck Wh*lply
  ## 3 5 Michael Spence 14 Michael Spence***** 11 M*ch**l Sp*nce
  ## 4 2 Maya Gutierrez 14 Maya Gutierrez****** 0 M*ya G*t**zz
  ## 5 7 Roberta Damstra 15 Roberta Damstra***** 0 R*b*t*r* D*mstr*
  ## 6 7 Rebecca Kanagaki 16 Rebecca Kanagaki****** 0 R*e*cka* K*n*g*k*
  ## 7 0 Kim Brunner 11 Kim Brunner********** 0 K*m Br*nrr
  ## 8 6 Brenda Blumberg 15 Brenda Blumberg***** 3 Br*nd* Bl*mbrrg
  ## 9 2 Darren Stanz 12 Darren Stanz********* 5 D*rr*n St*nz
  ## 10 4 Jonathan Murraiin 17 Jonathan Murraiin*** 0 J*n*t*n M*r**n
  ## ... with more rows, and 3 more variables: rpd <chr>, rnd <dbl>, first_three <chr>
}, silent=TRUE)
```

Index

* Dill direct REST API Interface
  drill_show_files, 17
  drill_show_schemas, 18
  drill_use, 25

* Drill Docker functions
  drill_up, 23
  killall_drill, 26
  showall_drill, 27

* Drill REST API (dplyr)
  drill_custom_functions, 8
  src_drill, 28

* Drill REST DBI API
  dbDataType, DrillConnection-method, 3
  dbGetInfo, DrillConnection-method
    (dbGetInfo, DrillDriver-method), 4
  dbGetInfo, DrillDriver-method, 4
  dbUnloadDriver, DrillDriver-method, 4
  Drill, 4, 5
  drill_active, 6, 7, 10–17, 19–22, 26
  drill_cancel, 6, 6, 7, 10–17, 19–22, 26
  drill_connection, 6, 7, 7, 10–17, 19–22, 26
  drill_connection(), 27
  drill_custom_functions, 8, 28
  drill_down (drill_up), 23
  drill_down(), 23
  drill_functions, 6, 7, 10, 11–17, 19–22, 26
  drill_metrics, 6, 7, 10, 11, 12–17, 19–22, 26
  drill_mod_storage (drill_storage), 20
  drill_mod_storage(), 20
  drill_options, 6, 7, 10, 11, 11, 13–17,
    19–22, 26
  drill_opts, 6, 7, 10–12, 12, 13–17, 19–22, 26
  drill_profile, 6, 7, 10–13, 13, 14–17,
    19–22, 26
  drill_profiles, 6, 7, 10–13, 14, 15–17,
    19–22, 26
  drill_query, 6, 7, 10–14, 14, 16, 17, 19–22,
    26
  drill_rm_storage (drill_storage), 20
  drill_set, 6, 7, 10–15, 15, 17, 19–22, 26
  drill_settings_reset, 6, 7, 10–16, 16,
    19–22, 26
  drill_show_files, 17, 18, 25
  drill_show_schemas, 17, 18, 25
  drill_stats, 6, 7, 10–17, 18, 19–22, 26
  drill_status, 6, 7, 10–17, 19, 19, 20–22, 26
  drill_storage, 6, 7, 10–17, 19, 20, 21, 22, 26
  drill_system_reset, 6, 7, 10–17, 19, 20, 21, 22, 26
  drill_threads, 6, 7, 10–17, 19–21, 22, 26
  drill_version, 25

ctas_profile, 2

dbConnect, DrillDriver-method (Drill), 5
dbDataType, DrillConnection-method, 3
INDEX

drill_up, 23, 26, 27
drill_uplift, 24
drill_use, 17, 18, 25
drill_version, 6, 7, 10–17, 19–22, 25
DrillConnection, 4
DrillDriver, 3, 4

format.DrillConnection, 26

getwd(), 23

killall_drill, 24, 26, 27

path.expand(), 23
print.drill_conn, 27

sergeant-exports, 27
showall_drill, 24, 26, 27
src_drill, 9, 28
src_tbls.src_drill, 9, 28

tbl (sergeant-exports), 27
tbl.src_drill (src_drill), 28