The [sergeant](https://rud.is/b/2018/09/09/driving-drill-dynamically-with-docker-and-updating-storage-configurations-on-the-fly-with-sergeant/gitlab.com/hrbrmstr/sergeant) package has a minor update that adds REST API coverage for [two “new” storage endpoints](https://drill.apache.org/docs/rest-api-introduction/) that make it possible to add, update and remove storage configurations on-the-fly without using the GUI or manually updating a config file.

This is an especially handy feature when paired with Drill’s new, official [Docker container](https://drill.apache.org/docs/running-drill-on-docker/) since that means we can:

* fire up a clean Drill instance
* modify the storage configuration (to, say, point to a local file system directory)
* execute SQL ops
* destroy the Drill instance

all from within R.

This is even more handy for those of us who prefer handling JSON data in Drill than in R directly or with sparklyr.

**Quick Example**

In a few weeks most of the following verbose-code-snippets will have a more diminutive and user-friendly interface within sergeant, but for now we’ll perform the above bulleted steps with some data that was used in [a recent new package](https://gitlab.com/hrbrmstr/dnshelpers/) which was also generated by another [recent new package](https://gitlab.com/hrbrmstr/zdnsr). The zdnsr::zdns\_exec() function ultimately generates a deeply nested JSON file that I really prefer working with in Drill before shunting it into R. Said file is stored, say, in the ~/drilldat directory.

Now, I have Drill running all the time on almost every system I use, but we’ll pretend I don’t for this example. I’ve run zdns\_exec() and generated the JSON file and it’s in the aforementioned directory. Let’s fire up an instance and connect to it:

library(sergeant) # git[hu|la]b:hrbrmstr/sergeant

library(dplyr)

docker <- Sys.which("docker") # you do need docker. it's a big dependency, but worth it IMO

(system2(

command = docker,

args = c(

"run", "-i",

"--name", "drill-1.14.0",

"-p", "8047:8047",

"-v", paste0(c(path.expand("~/drilldat"), "/drilldat"), collapse=":"),

"--detach",

"-t", "drill/apache-drill:1.14.0",

"/bin/bash"

),

stdout = TRUE

) -> drill\_container)

## [1] "d6bc79548fa073d3bfbd32528a12669d753e7a19a6258e1be310e1db378f0e0d"

The above snippet fires up a Drill Docker container (downloads it, too, if not already local) and wires up a virtual directory to it.

We should wait a couple seconds and make sure we can connect to it:

drill\_connection() %>%

drill\_active()

## [1] TRUE

Now, we need to add a storage configuration so we can access our virtual directory. Rather than modify dfs we’ll add a drilldat plugin that will work with the local filesystem just like dfs does:

drill\_connection() %>%

drill\_mod\_storage(

name = "drilldat",

config = '

{

"config" : {

"connection" : "file:///",

"enabled" : true,

"formats" : null,

"type" : "file",

"workspaces" : {

"root" : {

"location" : "/drilldat",

"writable" : true,

"defaultInputFormat": null

}

}

},

"name" : "drilldat"

}

')

## $result

## [1] "success"

Now, we can perform all the Drill ops sergeant has to offer, including ones like this:

(db <- src\_drill("localhost"))

## src: DrillConnection

## tbls: cp.default, dfs.default, dfs.root, dfs.tmp, drilldat.default, drilldat.root,

## INFORMATION\_SCHEMA, sys

tbl(db, "drilldat.root.`/\*.json`")

## # Source: table [?? x 10]

## # Database: DrillConnection

## data name error class status timestamp

##

## 1 "{\"authorities\":[{\"ttl\":180,\"type\":\"SOA\"… \_dmar… NA IN NOERR… 2018-09-09 13:18:07

## 2 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 3 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 4 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 5 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 6 "{\"authorities\":[{\"ttl\":1799,\"type\":\"SOA\… \_dmar… NA IN NOERR… 2018-09-09 13:18:07

## 7 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 8 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 9 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NXDOM… 2018-09-09 13:18:07

## 10 "{\"authorities\":[],\"protocol\":\"udp\",\"flag… \_dmar… NA IN NOERR… 2018-09-09 13:18:07

## # ... with more rows

(tbl(db, "(

SELECT

b.answers.name AS question,

b.answers.answer AS answer

FROM (

SELECT

FLATTEN(a.data.answers) AS answers

FROM

drilldat.root.`/\*.json` a

WHERE

(a.status = 'NOERROR')

) b

)") %>%

collect() -> dmarc\_recs)

## # A tibble: 1,250 x 2

## question answer

## \*

## 1 \_dmarc.washjeff.edu v=DMARC1; p=none

## 2 \_dmarc.barry.edu v=DMARC1; p=none; rua=mailto:dmpost@barry.edu,mailto:7cc566d7@mxtoolbox.d…

## 3 \_dmarc.yhc.edu v=DMARC1; pct=100; p=none

## 4 \_dmarc.aacc.edu v=DMARC1;p=none; rua=mailto:DKIM\_DMARC@aacc.edu;ruf=mailto:DKIM\_DMARC@aac…

## 5 \_dmarc.sagu.edu v=DMARC1; p=none; rua=mailto:Office365contact@sagu.edu; ruf=mailto:Office…

## 6 \_dmarc.colostate.edu v=DMARC1; p=none; pct=100; rua=mailto:re+anahykughvo@dmarc.postmarkapp.co…

## 7 \_dmarc.wne.edu v=DMARC1;p=quarantine;sp=none;fo=1;ri=86400;pct=50;rua=mailto:dmarcreply@…

## 8 \_dmarc.csuglobal.edu v=DMARC1; p=none;

## 9 \_dmarc.devry.edu v=DMARC1; p=none; pct=100; rua=mailto:devry@rua.agari.com; ruf=mailto:dev…

## 10 \_dmarc.sullivan.edu v=DMARC1; p=none; rua=mailto:mcambron@sullivan.edu; ruf=mailto:mcambron@s…

## # ... with 1,240 more rows

Finally (when done), we can terminate the Drill container:

system2(

command = "docker",

args = c("rm", "-f", drill\_container)

)

**FIN**

Those system2() calls are hard on the and a pain to type/remember, so they’ll be wrapped in some sergeant utility functions (I’m hesitant to add a reticulate dependency to sergeant which is necessary to use the docker package, hence the system call wrapper approach).

Check your favorite repository for more sergeant updates and file issues if you have suggestions for how you’d like this Docker API for Drill to be controlled.