Researching “the internet” (i.e. $DAYJOB) means having to deal with a ton of “unique” (I’m being kind) data formats. This is ultimately a tale of how I performed full-text searches across one of them.

It all started off innocently enough. This past week I need to be able to do full-text searches across metadata about who is using which parts of the internet. However, this time I needed *all* the resultant info and noticed an interesting foible on that full text search interface. To reproduce it. Enter something like “domino's” (for the record, I’m not researching Domino’s Pizza — nor would I ever consume it — but a Twitter ad happened to fly by for Domino’s and I just typed it for kicks) into the field and page around, keeping an eye on the results. I think they still use Solr for indexing/searching and aren’t passing in all they need to keep session context or something. Anyway, suffice it to say it was fairly useless (I filed a bug report, so I’m not just complaining, and I wish more sites had the same easy error reporting filing capability the RIPE folks do).

If it were just searching for precise data in one field, that’s not really an issue since we have ALL THE WHOIS IP THINGS in Parquet. But:

* I really *hate* giving Amazon money (even if it’s $WORK money) for Athena queries
* Full text search across all columns is not one of Parquet’s strengths
* This is a third bullet b/c I feel compelled to have a minimum of three points in bullet lists likely thanks to an overbearing middle-school English teacher

Since I have a modest analytics server setup at home, I figured I’d take the opportunity to re-brush-up on either Elasticsearch or Couchbase since [both are pretty great at free text searching JSON data](https://dzone.com/articles/searching-json-comparing-text-search-in-couchbase-3). Except…this isn’t JSON data, It’s records formatted like this:

as-block: AS7 - AS7

descr: RIPE NCC ASN block

remarks: These AS Numbers are assigned to network operators in the RIPE NCC service region.

mnt-by: RIPE-NCC-HM-MNT

created: 2018-11-22T15:27:05Z

last-modified: 2018-11-22T15:27:05Z

source: RIPE

remarks: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

remarks: \* THIS OBJECT IS MODIFIED

remarks: \* Please note that all data that is generally regarded as personal

remarks: \* data has been removed from this object.

remarks: \* To view the original object, please query the RIPE Database at:

remarks: \* http://www.ripe.net/whois

remarks: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

as-block: AS28 - AS28

descr: RIPE NCC ASN block

remarks: These AS Numbers are assigned to network operators in the RIPE NCC service region.

mnt-by: RIPE-NCC-HM-MNT

created: 2018-11-22T15:27:05Z

last-modified: 2018-11-22T15:27:05Z

source: RIPE

remarks: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

remarks: \* THIS OBJECT IS MODIFIED

remarks: \* Please note that all data that is generally regarded as personal

remarks: \* data has been removed from this object.

remarks: \* To view the original object, please query the RIPE Database at:

remarks: \* http://www.ripe.net/whois

remarks: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

They “keys” (the colon-ified line prefixes) vary and there are other record types (which I don’t need) that have other prefixes in them plus those #-prefixed comments are not necessarily only at the top. But, after judicious use of stringi::stri::stri\_enc\_toutf8(), stringi::stri\_split\_regex() and some vectorized record targeting they’re pretty easily converted to lovely ndjson data like this (random selection further in the conversion):

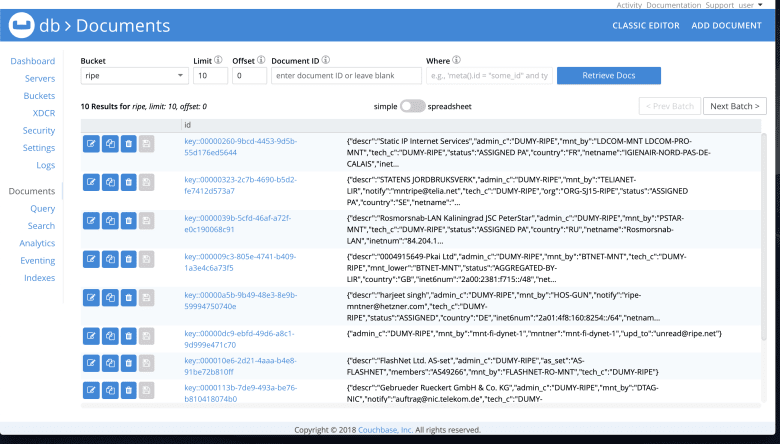
{"descr":"Reseau Teleinformatique de l'Education Nationale Educational and research network for Luxembourg","admin\_c":"DUMY-RIPE","as\_set":"AS-RESTENA","members":"AS2602, AS42909, AS51966, AS49624","mnt\_by":"AS2602-MNT","notify":"noc@restena.lu","tech\_c":"DUMY-RIPE"}

{"descr":"CWIX ASes announced to EBONE","admin\_c":"DUMY-RIPE","as\_set":"AS-TMPEBONECWIX","members":"AS3727, AS4445, AS4610, AS4624, AS4637, AS4654, AS4655, AS4656, AS4659 AS4681, AS4696, AS4714, AS4849, AS5089, AS5090, AS5532, AS5551, AS5559 AS5655, AS6081, AS6255, AS6292, AS6618, AS6639","mnt\_by":"EBONE-MNT","notify":"staff@ebone.net","tech\_c":"DUMY-RIPE"}

{"descr":"ASs accepted by DFN from the University of Cologne","admin\_c":"DUMY-RIPE","as\_set":"AS-DFNFROMCOLOGNE","members":"AS5520 AS6733","mnt\_by":"DFN-MNT","tech\_c":"DUMY-RIPE"}

{"descr":"NetMatters UK","admin\_c":"DUMY-RIPE","as\_set":"AS-NETMATTERS","members":"AS6765 AS3344","mnt\_by":"AS8407-MNT","tech\_c":"DUMY-RIPE"}

I went with Couchbase. Plus, Couchbase has been around long enough that it generally installs without pain and has a fairly decent web admin panel. Here’s a snapshot of the final import:

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2019/04/cb-screen.png?ssl=1)

and here’s the config for the “all” full text index:

{

"type": "fulltext-index",

"name": "all",

"uuid": "481bc7ed642dddfb",

"sourceType": "couchbase",

"sourceName": "ripe",

"sourceUUID": "3ffbbe0c0923f233ffe0fc96c652262d",

"planParams": {

"maxPartitionsPerPIndex": 171

},

"params": {

"doc\_config": {

"docid\_prefix\_delim": "",

"docid\_regexp": "",

"mode": "type\_field",

"type\_field": "type"

},

"mapping": {

"analysis": {},

"default\_analyzer": "standard",

"default\_datetime\_parser": "dateTimeOptional",

"default\_field": "\_all",

"default\_mapping": {

"dynamic": true,

"enabled": true

},

"default\_type": "\_default",

"docvalues\_dynamic": true,

"index\_dynamic": true,

"store\_dynamic": false,

"type\_field": "\_type"

},

"store": {

"indexType": "scorch",

"kvStoreName": ""

}

},

"sourceParams": {}

}

**You Said This Is A Post With R Code**

Very true! We’ll get to that in a minute.

Going with Couchbase introduced a different problem: there’s almost no R support for Couchbase. Sure, Couchbase has a gnarly, two-year old, raw httr::-prefixed but that’s not really as cool as if there were a library(couchbase).

If you were expecting a big reveal, right now, that I’ve got a feature-packed, full R Couchbase package ready to roll…you didn’t actually read the title of the post. What I do have— given server/connection metadata, a bucket, a full text index, and a query — will return all matching documents (I still do not like that term for “record”) for said set of parameters:

Function Code:

#' Return all record specified by document ids

#'

#' @param keys character vector of couchbase document ids

#' @param bucket the bucket where the documents live

#' @param cb\_svc\_url\_prefix,cb\_username,cb\_password Your couchbase connection info

cb\_get\_records\_from\_keys <- function(keys,

bucket,

cb\_svc\_url\_prefix = Sys.getenv("COUCHBASE\_SVC\_URL\_PREFIX"),

cb\_username = Sys.getenv("COUCHBASE\_USERNAME"),

cb\_password = Sys.getenv("COUCHBASE\_PASSWORD")) {

keys <- jsonlite::toJSON(keys)

httr::POST(

url = glue::glue("{cb\_svc\_url\_prefix}/query/service"),

httr::authenticate(cb\_username, cb\_password),

body = glue::glue('SELECT \* FROM {bucket} USE KEYS {keys}'),

encode = "raw"

) -> res

out <- httr::content(res, as = "text", encoding = "UTF-8")

out <- jsonlite::fromJSON(out)

out <- out[["results"]][[unique(names(out[["results"]]))]]

tibble::as\_tibble(out)

}

.cb\_fts\_q <- function(query,

fts\_index,

from = 0,

cb\_fts\_url\_prefix = Sys.getenv("COUCHBASE\_FTS\_URL\_PREFIX"),

cb\_username = Sys.getenv("COUCHBASE\_USERNAME"),

cb\_password = Sys.getenv("COUCHBASE\_PASSWORD")) {

httr::POST(

url = glue::glue("{cb\_fts\_url\_prefix}/api/index/{fts\_index}/query"),

httr::authenticate(cb\_username, cb\_password),

body = list(

query = list(

query = query,

from = from

)

),

encode = "json"

) -> res

httr::stop\_for\_status(res)

out <- httr::content(res, as = "text", encoding = "UTF-8")

out <- jsonlite::fromJSON(out)

out

}

#' Perform a full text search and receive matching documents

#'

#' @param query your [free text query](https://docs.couchbase.com/server/6.0/fts/full-text-intro.html)

#' @param index the full text [query index](https://docs.couchbase.com/server/6.0/fts/fts-creating-indexes.html)

#' @param bucket the bucket where the documents live

#' @param cb\_svc\_url\_prefix,cb\_username,cb\_password Your couchbase connection info

cb\_fts <- function(query,

index,

bucket,

cb\_fts\_url\_prefix = Sys.getenv("COUCHBASE\_FTS\_URL\_PREFIX"),

cb\_svc\_url\_prefix = Sys.getenv("COUCHBASE\_SVC\_URL\_PREFIX"),

cb\_username = Sys.getenv("COUCHBASE\_USERNAME"),

cb\_password = Sys.getenv("COUCHBASE\_PASSWORD")) {

# get initial result set

.cb\_fts\_q(

query = query,

fts\_index = index,

from = 0,

cb\_fts\_url\_prefix = cb\_fts\_url\_prefix,

cb\_username = cb\_username,

cb\_password = cb\_password

) -> rs

if (rs[["total\_hits"]] == 0) return(NULL) # bad/0-result query

if (rs[["total\_hits"]] > 10) { # if we need to paginate

times <- seq(10, rs[["total\_hits"]], 10) # figure out how many times

more <- purrr::map(times, ~.cb\_fts\_q( # do the thing

query = query,

fts\_index = index,

from = .x,

cb\_fts\_url\_prefix = cb\_fts\_url\_prefix,

cb\_username = cb\_username,

cb\_password = cb\_password

))

rs <- c(list(rs), more) # add back the original query and leave for futher processing

} else {

rs <- list(rs) # encapsulate to match ^^ format and leave for futher processing

}

purrr::flatten\_chr(

purrr::map(rs, ~.x[["hits"]][["id"]]) # get the record IDs

) -> keys

cb\_get\_records\_from\_keys( # retrieve the records that match

keys = keys,

bucket = bucket,

cb\_svc\_url\_prefix = cb\_svc\_url\_prefix,

cb\_username = cb\_username,

cb\_password = cb\_password

)

}

cb\_fts("domino's", "all", "ripe")

## # A tibble: 120 x 9

## admin\_c country descr inetnum mnt\_by netname status tech\_c notify

##

## 1 DUMY-RIPE FR OPEN IP DOMINO'S PIZZA 79.141.8.44 - 79.141.8.… ALPHALINK-… OPEN-IP ASSIGNED… DUMY-R… NA

## 2 DUMY-RIPE NL Domino's Pizza TILBURG 62.21.176.160 - 62.21.1… AS286-MNT OTS2634… ASSIGNED… DUMY-R… ip-reg@kpn.net

## 3 DUMY-RIPE NL Domino's Pizza EINDHOVEN 62.132.252.168 - 62.132… AS286-MNT OTS2270… ASSIGNED… DUMY-R… ip-reg@kpn.net

## 4 DUMY-RIPE NL Domino's Pizza SPYKENISSE 194.123.233.232 - 194.1… AS286-MNT OTS69259 ASSIGNED… DUMY-R… ip-reg@kpn.net

## 5 DUMY-RIPE NL Domino's AMSTERDAM 37.74.38.188 - 37.74.38… AS286-MNT OTS6103… ASSIGNED… DUMY-R… kpn-ip-office@…

## 6 DUMY-RIPE NL Domino's Pizza VOORSCHOTEN 92.66.116.136 - 92.66.1… AS286-MNT OTS1914… ASSIGNED… DUMY-R… ip-reg@kpn.net

## 7 DUMY-RIPE NL Domino's Pizza Doetinchem… 212.241.42.136 - 212.24… AS286-MNT OTS2301… ASSIGNED… DUMY-R… ip-reg@kpn.net

## 8 DUMY-RIPE NL Domino's Pizza AMSTERDAM 194.120.45.224 - 194.12… AS286-MNT OTS82906 ASSIGNED… DUMY-R… ip-reg@kpn.net

## 9 DUMY-RIPE NL Domino's Pizza [Woerden] … 62.41.228.80 - 62.41.22… AS286-MNT OTS2024… ASSIGNED… DUMY-R… ip-reg@kpn.net

## 10 DUMY-RIPE NL Domino's Pizza GRONINGEN 188.203.128.0 - 188.203… AS286-MNT OTS3767… ASSIGNED… DUMY-R… kpn-ip-office@…

## # … with 110 more rows

It’s not fancy.

It’s meets the needs of a narrow use-case.

It’s not in a standalone package (which is triggering my R code OCD something fierce).

But, it’s seriously fast, got me back to “work mode” with a minimum of hassle, and now there’s some google-able Couchbase R code that isn’t just bare httr calls that may help someone else who’s on a quest for how to work with Couchbase in R.

The first primary function – cb\_fts() — uses the /api/index/{index-name}/query API endpoint to paginate through results of the full text search and retrieves all matching record doc id keys, then calls the last primary function — cb\_get\_records\_from\_keys() — which uses the /query/service API endpoint, issues a SELECT \* FROM {bucket} USE KEYS {keys} query with all the found document (record) key ids and returns the result set. Nothing more fancy than that.

**FIN**

While I do not have these functions in a standalone, Couchbase-focused package I *do* have them in the package associated with this particular project. If you do know of a Couchbase R package (please don’t link to JDBC/ODBC drivers as I’m not going to buy) please link to them in the comments.

If you have other strategies for how to deal with these “un-packages”, please blog about it and post a link as well! I’m curious how others balance the package/not-a-package/un-package tension, especially when you may need to depend on a series of functions across projects.