The Spec Will Do

Clojure Going Meta

JAN STĘPIEŃ

Senior Consultant
@janstepien

Bringing Balance To Parentheses

INNOQ

Syntax

or the lack thereof

sort

```
sort [3, 2, 1]
```

```
(sort [3, 2, 1])
```

```
(sort [3 2 1])
```

```
(sort [3 2 1])

 format
```

```
(sort [3 2 1])

 format "%s:%d", file, line
```

```
(sort [3 2 1])

(format "%s:%d", file, line)
```

```
(sort [3 2 1])

(format "%s:%d" file line)
```

+

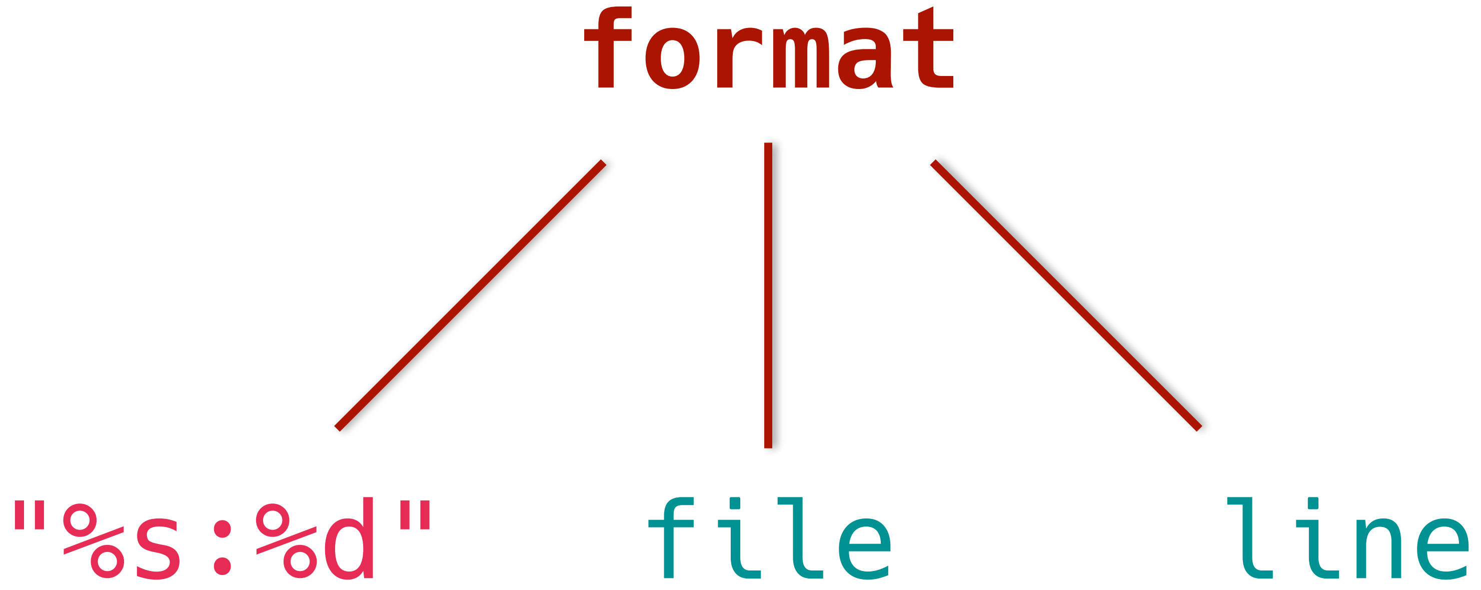
<=

```
(+ alpha beta gamma)

(<= 0 probability 1)
```

```
(request) => {
    log(request)
    validate(request)
    process(request)
}
```

```
(fn [request]
  (log request)
  (validate request)
  (process request))
```

```
(format "%s:%d" file line)
```


format
"%s:%d"
file
line

```clojure
(fn [input]
  (println input))
```

```clojure
(def log
  (fn [input]
    (println input)))
```

```clojure
(defn log
  [input]
  (println input)))
```

```clojure
(defn log
  [input]
  (println input))
```

$$\downarrow$$

```clojure
(def log
  (fn [input]
    (println input)))
```

Functions turn values
into values

Macros turn code
into code

```
(defmacro defn
```

```
(defn log
  [input]
  (println input)))
```

$\downarrow$

```
(def log
  (fn [input]
    (println input)))
```

```
(defmacro defn
  [symbol args body]
```

```
(defn log
  [input]
  (println input)))
```

```
       ↓
```

```
(def log
  (fn [input]
    (println input)))
```

```
(defmacro defn
  [symbol args body]
  (list 'def
        symbol
```

```
(defn log
  [input]
  (println input)))
```

```
(def log
  (fn [input]
    (println input)))
```

```
(defmacro defn
  [symbol args body]
  (list 'def
        symbol
        (list 'fn args body)))
```

=>

```
=> (defmacro defn
     [symbol args body]
     (list 'def
           symbol
           (list 'fn args body)))
```

```
=> (defmacro defn
     [symbol args body]
     (list 'def
           symbol
           (list 'fn args body)))
#'user/defn
=>
```

=>

```
=> (macroexpand '(defn log [val]
                      (println val)))
```

```
=> (macroexpand '(defn log [val]
                    (println val)))

(def log
  (fn [val] (println val)))

=>
```

=>

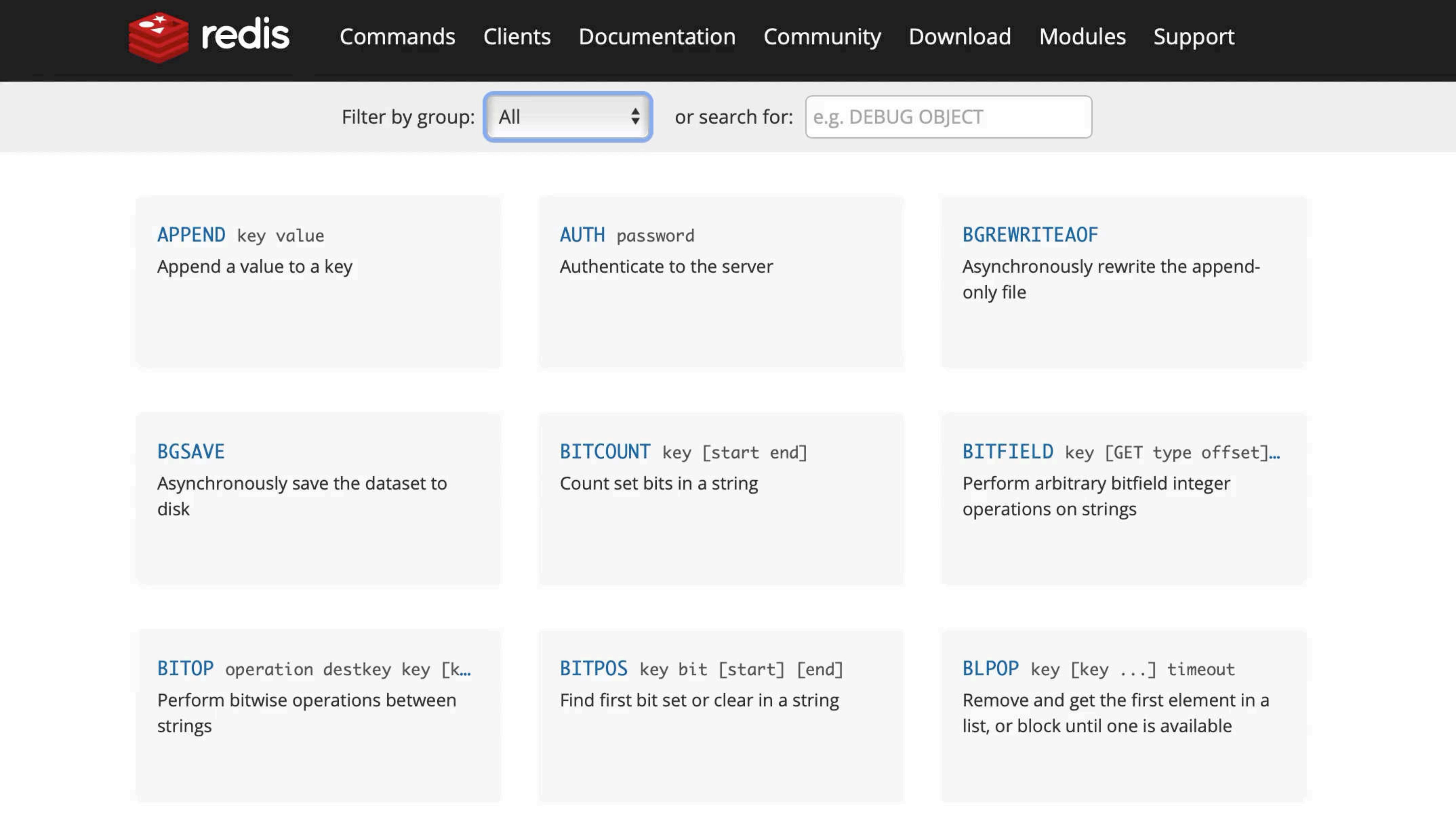
```
=> (defn log [val]
     (println val))

#'user/log

=>
```

```
=> (defn log [val]
     (println val))
#'user/log
=> (log "message")
message
=>
```

Functions turn values into values

Macros turn code into code

redis



Commands Clients Documentation Community Download Modules Support

Filter by group: All or search for: e.g. DEBUG OBJECT

`APPEND key value`
Append a value to a key

`AUTH password`
Authenticate to the server

`BGREWRITEAOF`
Asynchronously rewrite the append-only file

`BGSAVE`
Asynchronously save the dataset to disk

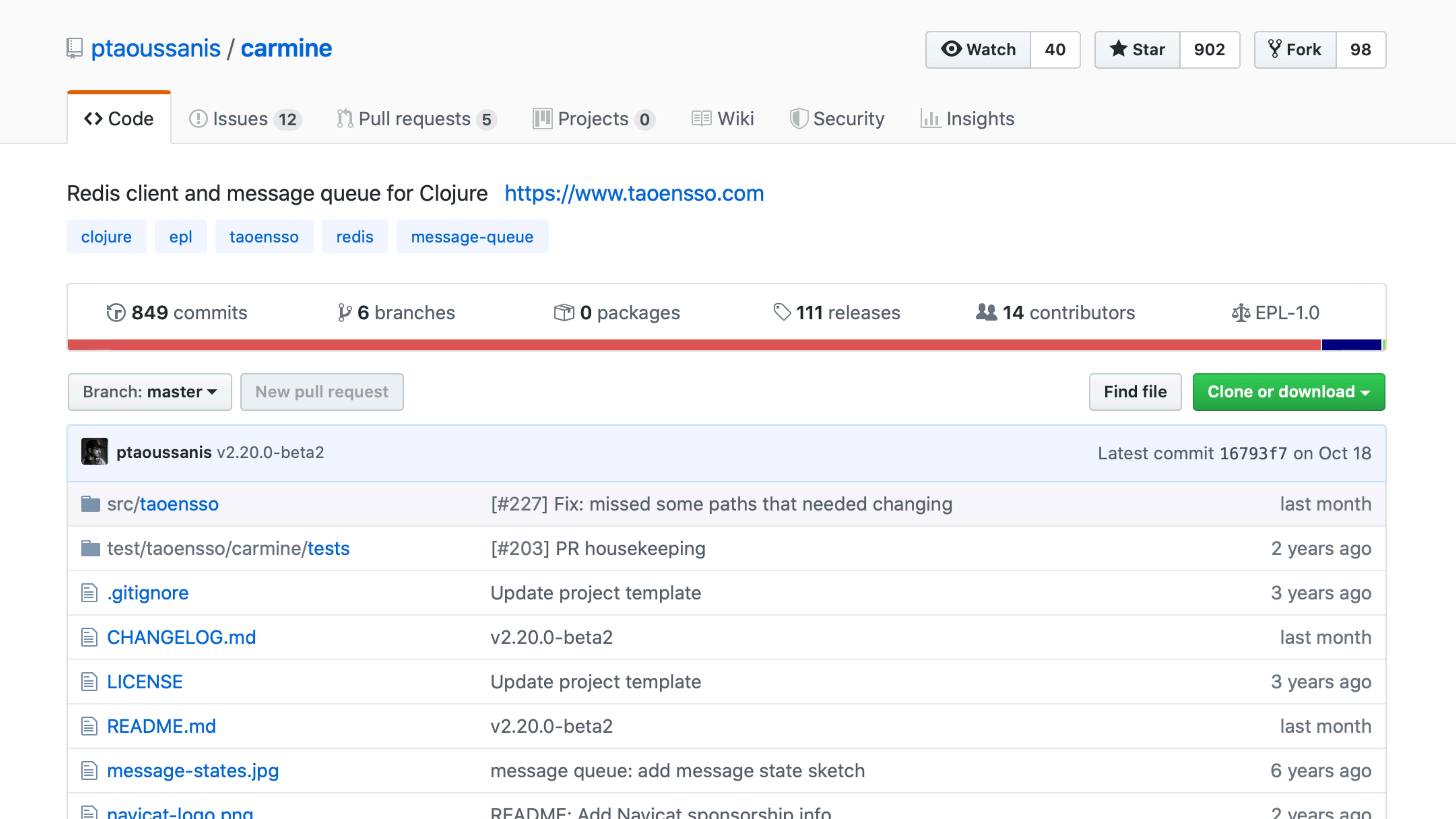
`BITCOUNT key [start end]`
Count set bits in a string

`BITFIELD key [GET type offset]…`
Perform arbitrary bitfield integer operations on strings

`BITOP operation destkey key [k…`
Perform bitwise operations between strings

`BITPOS key bit [start] [end]`
Find first bit set or clear in a string

`BLPOP key [key ...] timeout`
Remove and get the first element in a list, or block until one is available



ptaoussanis / carmine

Watch 40 | Star 902 | Fork 98

Code | Issues 12 | Pull requests 5 | Projects 0 | Wiki | Security | Insights

Redis client and message queue for Clojure https://www.taoensso.com

clojure | epl | taoensso | redis | message-queue

849 commits | 6 branches | 0 packages | 111 releases | 14 contributors | EPL-1.0

Branch: master | New pull request | Find file | Clone or download

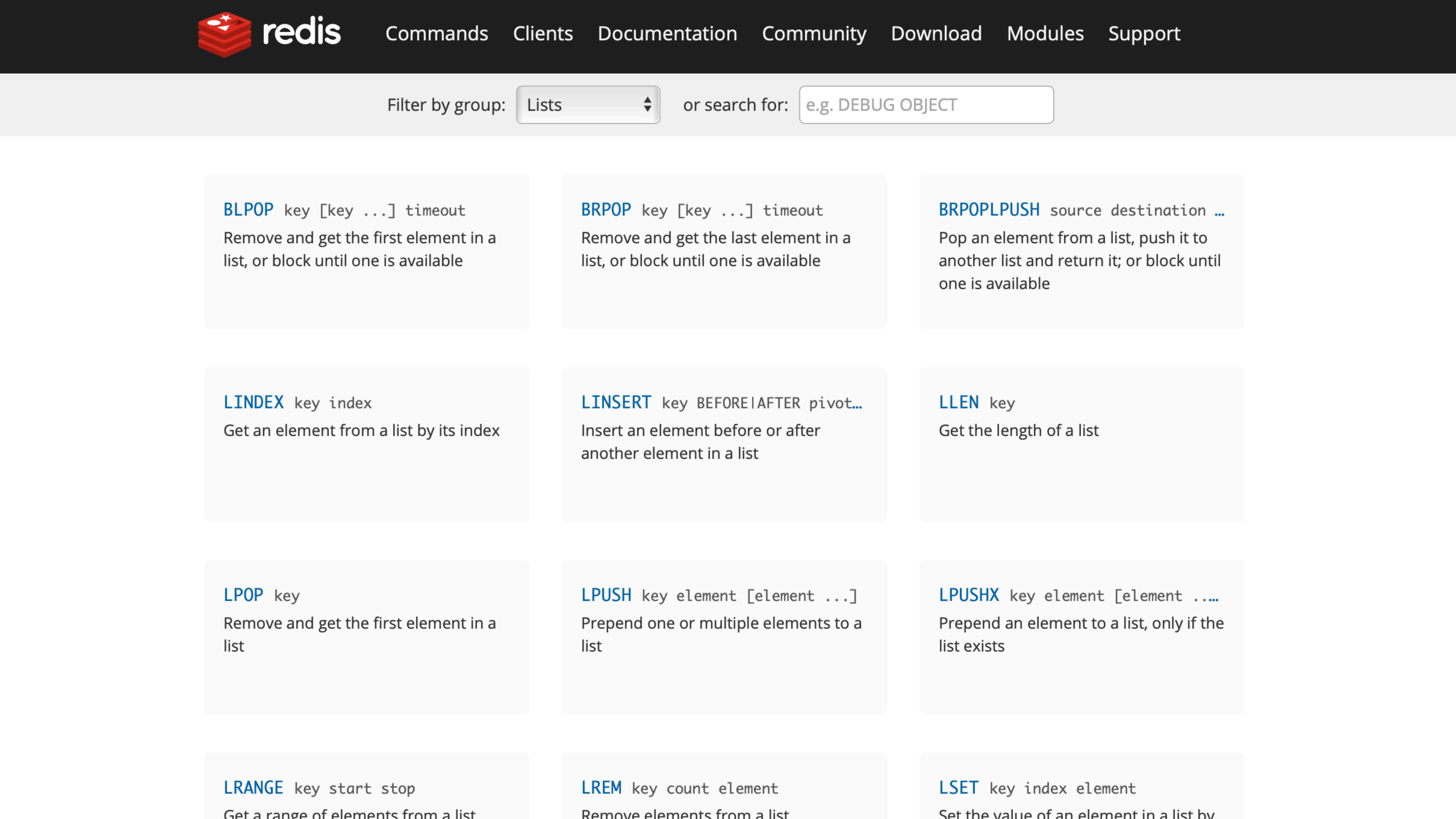
ptaoussanis v2.20.0-beta2 — Latest commit 16793f7 on Oct 18

src/taoensso	[#227] Fix: missed some paths that needed changing	last month
test/taoensso/carmine/tests	[#203] PR housekeeping	2 years ago
.gitignore	Update project template	3 years ago
CHANGELOG.md	v2.20.0-beta2	last month
LICENSE	Update project template	3 years ago
README.md	v2.20.0-beta2	last month
message-states.jpg	message queue: add message state sketch	6 years ago
navicat-logo.png	README: Add Navicat sponsorship info	2 years ago

```clojure
(def conn
  {:spec {:host "127.0.0.1"
          :port 6379}})

(car/wcar conn
          (car/ping)
          (car/set "foo" "bar")
          (car/get "foo"))
```

```clojure
(enc/defalias raw            protocol/raw)
(enc/defalias with-thaw-opts nippy-tools/with-thaw-opts)
(enc/defalias freeze         nippy-tools/wrap-for-freezing
  "Forces argument of any type (incl. keywords, simple numbers, and binary types)
  to be subject to automatic de/serialization with Nippy.")

(enc/defalias return protocol/return)
(comment (wcar {} (return :foo) (ping) (return :bar))
         (wcar {} (parse name (return :foo)) (ping) (return :bar)))

;;;; Standard commands

(commands/defcommands) ; Defines 190+ Redis command fns as per official spec

(defn redis-call
  "Sends low-level requests to Redis. Useful for DSLs, certain kinds of command
  composition, and for executing commands that haven't yet been added to the
  official `commands.json` spec.
  (redis-call [:set \"foo\" \"bar\"] [:get \"foo\"])"
  [& requests]
  (enc/run!
    (fn [[cmd & args]]
      (let [cmd-parts (-> cmd name str/upper-case (str/split #"-"))
            request   (into (vec cmd-parts) args)]
        (commands/enqueue-request (count cmd-parts) request)))
```

```
(commands/defcommands) ; Defines 190+ Redis command
                       ; fns as per official spec
```



Filter by group: Lists or search for: e.g. DEBUG OBJECT

`BLPOP key [key ...] timeout`
Remove and get the first element in a list, or block until one is available

`BRPOP key [key ...] timeout`
Remove and get the last element in a list, or block until one is available

`BRPOPLPUSH source destination ...`
Pop an element from a list, push it to another list and return it; or block until one is available

`LINDEX key index`
Get an element from a list by its index

`LINSERT key BEFORE|AFTER pivot...`
Insert an element before or after another element in a list

`LLEN key`
Get the length of a list

`LPOP key`
Remove and get the first element in a list

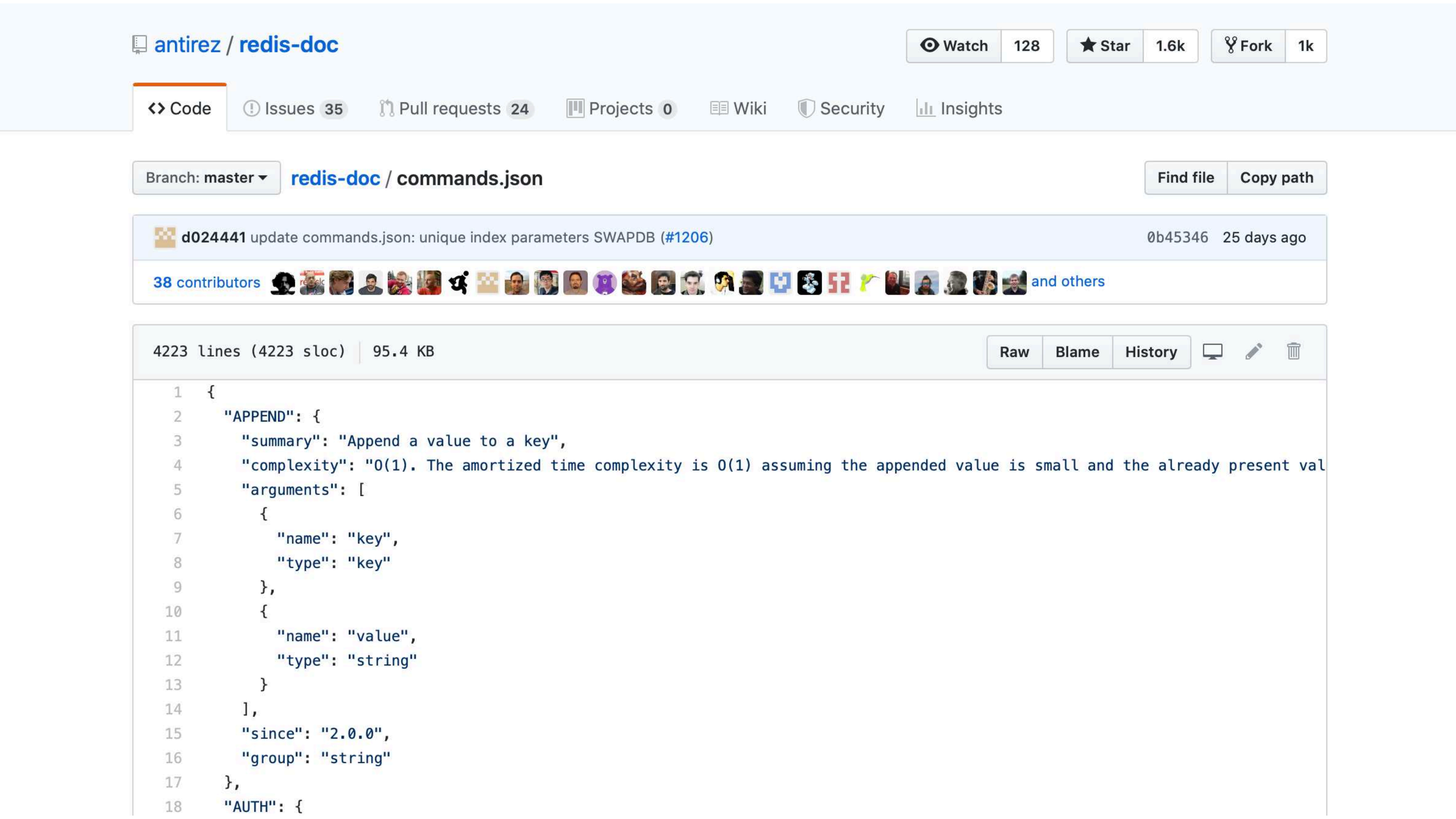
`LPUSH key element [element ...]`
Prepend one or multiple elements to a list

`LPUSHX key element [element ....`
Prepend an element to a list, only if the list exists

`LRANGE key start stop`
Get a range of elements from a list

`LREM key count element`
Remove elements from a list

`LSET key index element`
Set the value of an element in a list by


antirez / redis-doc

Watch 128 | Star 1.6k | Fork 1k

Code | Issues 35 | Pull requests 24 | Projects 0 | Wiki | Security | Insights

Branch: master | redis-doc / commands.json | Find file | Copy path

d024441 update commands.json: unique index parameters SWAPDB (#1206) 0b45346 25 days ago

38 contributors and others

4223 lines (4223 sloc) | 95.4 KB | Raw | Blame | History

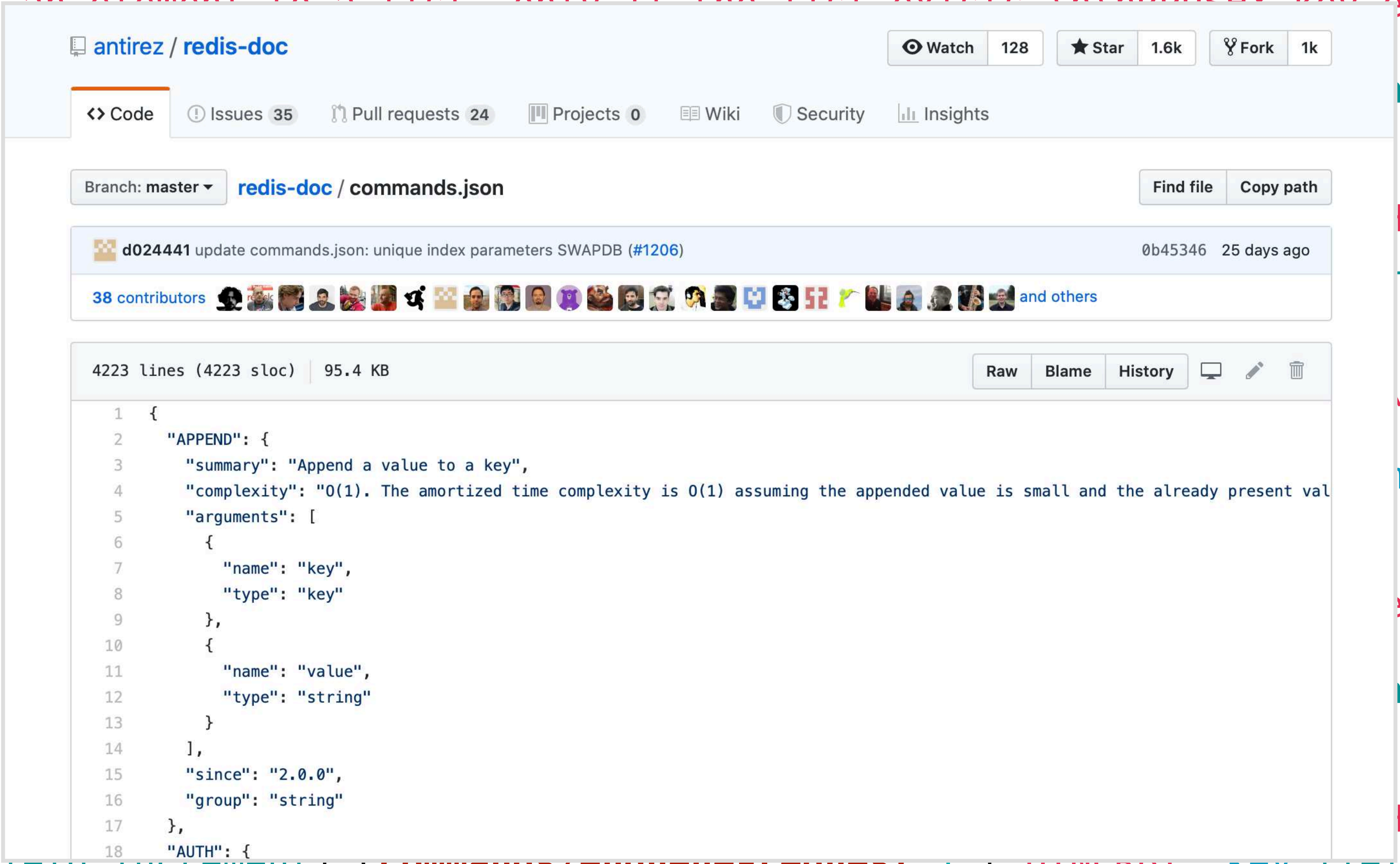
```
{
  "APPEND": {
    "summary": "Append a value to a key",
    "complexity": "O(1). The amortized time complexity is O(1) assuming the appended value is small and the already present val
    "arguments": [
      {
        "name": "key",
        "type": "key"
      },
      {
        "name": "value",
        "type": "string"
      }
    ],
    "since": "2.0.0",
    "group": "string"
  },
  "AUTH": {
```

```clojure
(defn get-command-spec
  "Given a `clojure/slurp`-able Redis commands.json location (e.g. URL),
  reads the json and returns a cleaned-up Redis command spec map."
  ([] (get-command-spec
        (java.net.URL. "https://raw.githubusercontent.com/ ↵
                        antirez/redis-doc/master/commands.json")))
  ([json]
   (try
     (let [m (clojure.data.json/read-str (slurp json) :key-fn keyword)]
       (persistent!
         (reduce-kv
           (fn [m k v]
             (let [cmd-name (name k)
                   fn-name
                   (-> cmd-name str/lower-case (str/replace #" " "-"))

                   {:keys [arguments summary since complexity]} v

                   [fn-params-fixed
                    fn-params-more?]
```

```clojure
(defonce ^:private command-spec
  (if-let [edn (enc/slurp-resource "taoensso/carmine/commands.edn")]
    (try
      (enc/read-edn edn)
      (catch Exception e
        (throw (ex-info "Failed to read Carmine commands edn" {} e))))
    (throw (ex-info "Failed to find Carmine commands edn" {}))))

(defmacro defcommands []
  `(do ~@(map (fn [[k v]] `(defcommand ~k ~v)) command-spec)))
```

```
(defn rpushx
  "Append an element to a list, only if the list exists.\n\nRPUSHX key element [ele
  ([key element] (commands/enqueue-request 1 ["RPUSHX" key element]))
  ([key element & args] (commands/enqueue-request 1 ["RPUSHX" key element] args)))

(defn hincrby
  "Increment the integer value of a hash field by the given number.\n\nHINCRBY key
   [key field increment] (commands/enqueue-request 1 ["HINCRBY" key field incremen

(defn zrangebylex
  "Return a range of members in a sorted set, by lexicographical range.\n\nZRANGEBY
  ([key min max] (commands/enqueue-request 1 ["ZRANGEBYLEX" key min max]))
  ([key min max & args] (commands/enqueue-request 1 ["ZRANGEBYLEX" key min max] arg
```

```
(defn rpushx
  "Append an element to a list, only if the list exists.\n\nRPUSHX key element [ele
  ([key element] (commands/enqueue-request 1 ["RPUSHX" key element]))
  ([key element & args] (commands/enqueue-request 1 ["RPUSHX" key element] args)))

(defn hincrby
  "Increment the integer value of a hash field by the given number.\n\nHINCRBY key
   [key field increment] (commands/enqueue-request 1 ["HINCRBY" key field incremen
(defn rpushx
  "Append an element to a list, only if the list exists.\n\nRPUSHX key element [ele
  ([key e
  ([key e                                                                    t] args)))

(defn hin
  "Increm                                                                    INCRBY key
   [key f                                                                    d increment

(defn zra
  "Return                                                                    n\nZRANGEBY
  ([key m                                                                     ))
  ([key m                                                                    in max] arg

(defn rpu
  "Append                                                                    lement [ele
  ([key e
  ([key e                                                                    t] args)))

(defn hin
  "Increm                                                                    INCRBY key
   [key field increment] (commands/enqueue-request 1 ["HINCRBY" key field increment
```



antirez / redis-doc

Watch 128 · Star 1.6k · Fork 1k

Code · Issues 35 · Pull requests 24 · Projects 0 · Wiki · Security · Insights

Branch: master · redis-doc / commands.json · Find file · Copy path

d024441 update commands.json: unique index parameters SWAPDB (#1206) · 0b45346 25 days ago

38 contributors and others

4223 lines (4223 sloc) | 95.4 KB · Raw · Blame · History

```
{
  "APPEND": {
    "summary": "Append a value to a key",
    "complexity": "O(1). The amortized time complexity is O(1) assuming the appended value is small and the already present val
    "arguments": [
      {
        "name": "key",
        "type": "key"
      },
      {
        "name": "value",
        "type": "string"
      }
    ],
    "since": "2.0.0",
    "group": "string"
  },
  "AUTH": {
```

Don’t

at least in public

The Spec Will Do

Jan Stępień

janstepien.com