# 1 R语言

## R client for the Elasticsearch HTTP API

代码1

|  |
| --- |
| #https://github.com/ropensci/elastic  install.packages("elastic")  install.packages("devtools")  devtools::install\_github("ropensci/elastic")  library('elastic')  connect(es\_host = "172.28.11.167", es\_user="elastic", es\_pwd = "changeme", es\_port = 9200)  Search(index = "dcsid-2017.08.03",size = 1)$hits$hits |

代码2

|  |
| --- |
| #Stable version from CRAN  install.packages("elastic")  #还安装相依关系‘mime’, ‘openssl’, ‘R6’, ‘httr’, ‘curl’, ‘jsonlite’  #Development version from GitHub  install.packages("devtools")  #还安装相依关系‘memoise’, ‘whisker’, ‘digest’, ‘rstudioapi’, ‘git2r’, ‘withr’  devtools::install\_github("ropensci/elastic")  一.  1.链接  connect(es\_port = 9200)  2.使用x-pack  connect(es\_host = "172.28.11.167", es\_path = "", es\_user="elastic", es\_pwd = "changeme", es\_port = 9200, es\_transport\_schema = "https")  #修改1  connect(es\_host = "172.28.11.167", es\_user="elastic", es\_pwd = "changeme", es\_port = 9200)  Search(index = "dcsid-2017.08.03",size = 1)$hits$hits |

## elasticsearchr

|  |
| --- |
| #https://cran.r-project.org/web/packages/elasticsearchr/vignettes/quick\_start.html  devtools::install\_github("alexioannides/elasticsearchr")  es <- elastic("http://172.28.11.167:9200", "elastic", "changeme")  for\_everything <- query('{  "match\_all": {}  }') |

# Perl API

## 安装

|  |
| --- |
| <https://metacpan.org/pod/Search::Elasticsearch>  ppm install Search::Elasticsearch  # Generating HTML for Search-Elasticsearch-5.01...done  # Generating HTML for Any-URI-Escape-0.01...done  # Generating HTML for namespace-clean-0.27...done  # Generating HTML for Moo-2.003002...done  # Generating HTML for Log-Any-1.049...done  # Generating HTML for B-Hooks-EndOfScope-0.21...done  # Generating HTML for Role-Tiny-2.000005...done  # Generating HTML for Sub-Quote-2.004000...done  # Generating HTML for Class-Method-Modifiers-2.12...done  # Generating HTML for Variable-Magic-0.61...done  # Updating files in site area...done |

## Demo

|  |
| --- |
| use Search::Elasticsearch;  # Round-robin between two nodes:  my $e1 = Search::Elasticsearch->new(  nodes => [  '192.168.11.73:9200',  '192.168.11.74:9200',  '192.168.11.75:9200',  '192.168.11.76:9200',  '192.168.11.77:9200'  ]  );  # Connect to cluster at 192.168.11.73:9200, sniff all nodes and round-robin between them:  my $e = Search::Elasticsearch->new(  nodes => '192.168.11.73:9200',  cxn\_pool => 'Sniff'  );  # Index a document:  $e->index(  index => 'my\_app',  type => 'blog\_post',  id => 1,  body => {  title => 'Elasticsearch clients',  content => 'Interesting content...',  date => '2013-09-24'  }  );  # Get the document:  my $doc = $e->get(  index => 'my\_app',  type => 'blog\_post',  id => 1  );  # Search:  my $results = $e->search(  index => 'my\_app',  body => {  query => {  match => { title => 'elasticsearch' }  }  }  );  # Cluster requests:  my $health = $e->cluster->health;  my $state = $e->cluster->state;  # Index requests:  $e->indices->create(index=>'my\_index');  $e->indices->delete(index=>'my\_index');  $e->indices->delete(index=>'my\_app');  foreach $key (keys %$health)  {  print "$key=>$health->{$key}\n";  }  # foreach $key (keys %$results)  # {  # $value=$results->{$key};  # if (ref($value) eq 'HASH') {  # print "$key=>$value\n";  # print "\t\*\*\*\*\*\*\*\*\*\*\*begin\*\*\*\*\*\*\*\*\* \n";  # foreach $subkey (keys %$value){  # print "\t$subkey=>$value->{$subkey}\n";  # }  # print "\t\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\* \n";  # }  # else{  # print "$key=>$value\n";  # }  # }  foreach $key (keys %$state)  {  $value=$state->{$key};  if (ref($value) eq 'HASH') {  print "$key=>$value\n";  print "\t\*\*\*\*\*\*\*\*\*\*\*begin\*\*\*\*\*\*\*\*\* \n";  foreach $subkey (keys %$value){  $subvalue=$value->{$subkey};  if (ref($subvalue) eq 'HASH') {  print "$subkey=>$value->{$subkey}\n";  print "\t\t\*\*\*\*\*\*\*\*\*\*\*sub\_begin\*\*\*\*\*\*\*\*\* \n";  foreach $sub\_subkey (keys %$subvalue){  print "\t\t$sub\_subkey=>$subvalue->{$sub\_subkey}\n";  }  print "\tt\*\*\*\*\*\*\*\*\*\*\*sub\_end\*\*\*\*\*\*\*\*\* \n";  }  else{  print "\t$subkey=>$value->{$subkey}\n";  }  }  print "\t\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\* \n";  }  else{  print "$key=>$value\n";  }  } |

# Python

## 集成python API

|  |
| --- |
| # Official low-level client for Elasticsearch  # elasticsearch-py  # http://elasticsearch-py.rtfd.org/  # C:\Users\unicom>pip install elasticsearch  # Successfully installed elasticsearch-5.4.0 urllib3-1.22  from datetime import datetime  from elasticsearch import Elasticsearch  # Thread safety:By default we allow urllib3 to open up to 10 connections to each node,  # if your application calls for more parallelism, use the maxsize parameter to raise the limit:  # allow up to 25 connections to each node  es = Elasticsearch([  '192.168.11.73:9200',  '192.168.11.74:9200',  '192.168.11.75:9200',  '192.168.11.76:9200',  '192.168.11.77:9200'  ],  maxsize=25)  # you can specify to sniff on startup to inspect the cluster and load  # balance across all nodes  es2 = Elasticsearch([  '192.168.11.73:9200',  '192.168.11.74:9200',  '192.168.11.75:9200',  '192.168.11.76:9200',  '192.168.11.77:9200'  ],  sniff\_on\_start=True)  # you can also sniff periodically and/or after failure:  es3 = Elasticsearch([  '192.168.11.73:9200',  '192.168.11.74:9200',  '192.168.11.75:9200',  '192.168.11.76:9200',  '192.168.11.77:9200'],  sniff\_on\_start=True,  sniff\_on\_connection\_fail=True,  sniffer\_timeout=60)  # SSL client authentication using client\_cert and client\_key  es4 = Elasticsearch(  ['192.168.11.73', '192.168.11.74'],  http\_auth=('user', 'secret'),  # port=443,  # use\_ssl=True,  # ca\_certs='/path/to/cacert.pem',  # client\_cert='/path/to/client\_cert.pem',  # client\_key='/path/to/client\_key.pem',  )  doc = {  'author': 'kimchy',  'text': 'Elasticsearch: cool. bonsai cool.',  'timestamp': datetime.now(),  }  res = es.index(index="test-index", doc\_type='tweet', id=1, body=doc)  print(res['created'])  res = es.get(index="test-index", doc\_type='tweet', id=1)  print(res['\_source'])  es.indices.refresh(index="test-index")  res = es.search(index="test-index", body={"query": {"match\_all": {}}})  print("Got %d Hits:" % res['hits']['total'])  for hit in res['hits']['hits']:  print("%(timestamp)s %(author)s: %(text)s" % hit["\_source"])  es.indices.create(index='test-index', ignore=400)  # ignore 404 and 400  es.indices.delete(index='test-index', ignore=[400, 404]) |

## 自定义API的demo

|  |
| --- |
| #-\*-coding:utf8-\*-  """  date:20170602  """  import json  import urllib2  import urllib  class HttpElasticSearch(object):  def \_\_init\_\_(self, url):  self.url = url    def search\_one\_id(self, id\_in):  """  从es查某id  """  fix\_url = self.url + "\_search"  data ={"query":{}}  data["query"]["match"] = {}  data["query"]["match"]["id"] = id\_in  data = json.dumps(data)  req = urllib2.Request(fix\_url, data)  req.get\_method = lambda:'POST'  out = json.loads(urllib2.urlopen(req, timeout=1000).read().strip())  return out  def search\_title\_regx(self, title\_regx):  """  根据title的正则表达式  查出相应的id  """  fix\_url = self.url + "\_search"  data = {"query":{}}  data["query"]["wildcard"] = {}  data["query"]["wildcard"]["title"] = title\_regx  data["\_source"] = ["id", "title"]  data = json.dumps(data)  req = urllib2.Request(fix\_url, data)  req.get\_method = lambda:'POST'  out = json.loads(urllib2.urlopen(req, timeout=1000).read().strip())  return out    def update\_one\_doc(self, id\_in, update\_body):  """  update\_body is a dict:  like {"viewCount":"5800"}  """  res = obj.search\_one\_id(id\_in)  if "hits" not in res or "hits" not in res["hits"] or "\_type" not in res["hits"]["hits"][0]:  return False  type\_t = str(res["hits"]["hits"][0]["\_type"])  fix\_url = self.url + type\_t + "/" + id\_in + "/\_update"  data = {"doc":update\_body}  data = json.dumps(data)  req = urllib2.Request(fix\_url, data)  req.get\_method = lambda:'POST'  out = json.loads(urllib2.urlopen(req, timeout=1000).read().strip())  if "\_shards" not in out or int(out["\_shards"]["successful"]) != int(out["\_shards"]["total"]):  return False  return True  if \_\_name\_\_ == "\_\_main\_\_":  obj = HttpElasticSearch("host:port/你自己的索引/")  #res = obj.search\_one\_id("docid")  #print obj.search\_title\_regx("Best Surprise\*")  if obj.update\_one\_doc("d5b39859b8f4cd0fe01f0116af01a733",{"viewCount":"5900"}):  print "hihi" |