|  |  |
| --- | --- |
| **2) Setup**  **02\_02\_\_Setup.txt**  In Kibana: <http://127.0.0.1:5601/app/dev_tools#/console>  # Cluster Health Check | |
| ## Overall Cluster Health GET /\_cat/health?v | epoch timestamp cluster status node.total node.data shards pri relo init unassign pending\_tasks max\_task\_wait\_time active\_shards\_percent 1681330358 20:12:38 elasticsearch yellow 1 1 25 25 0 0 6 0 - 80.6% |
| ## Node Health (define how many nodes - 1 node) GET /\_cat/nodes?v | ip heap.percent ram.percent cpu load\_1m load\_5m load\_15m node.role master name 127.0.0.1 32 84 13 cdfhilmrstw \* VLADIMIRP |
| ## List Indices GET \_cat/indices?v | health status index uuid pri rep docs.count docs.deleted store.size pri.store.size green open .geoip\_databases I3qMx8wcQte6eQvI1U1t0w 1 0 42 72 78.5mb 78.5mb green open .apm-custom-link -TTZSBhYTHqbuPQJAebUHA 1 0 0 0 208b 208b green open .kibana\_task\_manager\_1 ACTJZFBURV2TJDM77Q9wMA 1 0 5 67 86.6kb 86.6kb green open .apm-agent-configuration slJZiJAPTYCFDz\_ePtlUSw 1 0 0 0 208b 208b green open .kibana\_1 LBhfwFXEQrGjKXNWFNKH5Q 1 0 75 2 2.1mb 2.1mb green open .kibana-event-log-7.10.2-000004 Nj4GjdyiSGabMWXMljclxg 1 0 1 0 5.6kb 5.6kb |
| ## Create 'sales' Index (first index) PUT /sales | {  "acknowledged" : true,  "shards\_acknowledged" : true,  "index" : "sales" } |
| ## Add 'order' to 'sales' index PUT /sales/order/123 {  "orderID":"123",  "orderAmount":"500" } | {  "\_index" : "sales",  "\_type" : "order",  "\_id" : "123",  "\_version" : 1,  "result" : "created",  "\_shards" : {  "total" : 2,  "successful" : 1,  "failed" : 0  },  "\_seq\_no" : 0,  "\_primary\_term" : 1 } |
| ## Retrieve document GET /sales/order/123 | {  "\_index" : "sales",  "\_type" : "order",  "\_id" : "123",  "\_version" : 1,  "\_seq\_no" : 0,  "\_primary\_term" : 1,  "found" : true,  "\_source" : {  "orderID" : "123",  "orderAmount" : "500"  } } |
| ## Delete index DELETE sales | {  "acknowledged" : true } |
| ## List indices GET /\_cat/indices?v |  |
| **3) Loading Data** **03\_01.txt Bulk loading data** | |
| # Get ES endpoint # from https://cloud.elastic.co/home click gear icon next to cluster # copy endpoint next to "Elasticsearch" under Applications and create variable in terminal  ES\_HOST="<PASTE ENDPOINT HERE>"  # now create API key # click on security then in settings "kibana" link # scroll down to API Keys then "Create API Key" # create variable in terminal  ES\_API\_KEY="<PASTE API KEY HERE>"  # Create File with Requests (make sure to include new line at end of file) # open VI and create a file called "reqs" vi reqs  # type "i" then copy/paste this data into vi  { "index" : { "\_index" : "my-test", "\_id" : "1" } } { "col1" : "val1"} { "index" : { "\_index" : "my-test", "\_id" : "2" } } { "col1" : "val2"} { "index" : { "\_index" : "my-test", "\_id" : "3" } } { "col1" : "val3" }  # hit esc # type :wq to save the file and exit vi # type ls to see the file is there  # OR Create File 'reqs' with Requests (make sure to include new line at end of file) { "index" : { "\_index" : "my-test", "\_type" : "my-type", "\_id" : "1" } } { "col1" : "val1"} { "index" : { "\_index" : "my-test", "\_type" : "my-type", "\_id" : "2" } } { "col1" : "val2"} { "index" : { "\_index" : "my-test", "\_type" : "my-type", "\_id" : "3" } } { "col1" : "val3" }   # using curl, upload data file to cluster and create index  curl -XPOST -i -k \ -H "Content-Type: application/x-ndjson" \ -H "Authorization: ApiKey $ES\_API\_KEY" \ $ES\_HOST/\_bulk --data-binary "@reqs"; echo  # OR Load from CURL curl -H "Content-Type: application/x-ndjson" -XPOST -i -k localhost:9200/\_bulk --data-binary "@data/reqs"; echo  # Login to cluster in web portal, go to dev tools, look for indices and return our data GET /\_cat/indices?v | |
| # Check Kibana GET /my-test | {  "my-test" : {  "aliases" : { },  "mappings" : {  "properties" : {  "col1" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword",  "ignore\_above" : 256  }  }  }  }  },  "settings" : {  "index" : {  "routing" : {  "allocation" : {  "include" : {  "\_tier\_preference" : "data\_content"  }  }  },  "number\_of\_shards" : "1",  "provided\_name" : "my-test",  "creation\_date" : "1681415211880",  "number\_of\_replicas" : "1",  "uuid" : "GF\_EIrw6TRSudfRiO2FxFw",  "version" : {  "created" : "7150099"  }  }  }  }  } |
| GET /my-test/my-type/1 GET /my-test/\_doc/1 если "\_type" : "my-type" был не указан | {  "\_index" : "my-test",  "\_type" : "\_doc",  "\_id" : "1",  "\_version" : 2,  "\_seq\_no" : 3,  "\_primary\_term" : 1,  "found" : true,  "\_source" : {  "col1" : "val1"  }  } |
| b) # Load from Console POST \_bulk { "index" : { "\_index" : "my-test-console", "\_type" : "my-type", "\_id" : "1" } } { "col1" : "val1" } { "index" : { "\_index" : "my-test-console", "\_type" : "my-type", "\_id" : "2" } } { "col1" : "val2"} { "index" : { "\_index" : "my-test-console", "\_type" : "my-type", "\_id" : "3" } } { "col1" : "val3" }  # Check Kibana GET /my-test-console GET /my-test-console/my-type/1 | |
| **03\_02.txt Loading sample data**  # Download from Elastic or use Exercise Files # If necessary unzip the accounts.zip file # https://download.elastic.co/demos/kibana/gettingstarted/accounts.zip  # change to downloaded folder cd <path to folder>  # Inspect Accounts.json head accounts.json  # Load via curl, notice the endpoint which creates the 'bank' index curl -XPOST -i -k \ -H "Content-Type: application/x-ndjson" \ -H "Authorization: ApiKey $ES\_API\_KEY" \ $ES\_HOST/bank/\_bulk?pretty --data-binary @accounts.json; echo  # OR Load via curl, notice the endpoint and type curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/bank/account/\_bulk?pretty' --data-binary @data/accounts.json | |
| # check inside ES GET /\_cat/indices GET /bank | {  "bank" : {  "aliases" : { },  "mappings" : {  "properties" : {  "account\_number" : {  "type" : "long"  },  "address" : {  "type" : "text",  "fields" : {  "keyword" : {  "type" : "keyword",  "ignore\_above" : 256  }  }  },  "age" : {  "type" : "long"  }, |
| GET /bank/\_doc/1 | {  "\_index" : "bank",  "\_type" : "\_doc",  "\_id" : "1",  "\_version" : 2,  "\_seq\_no" : 1000,  "\_primary\_term" : 11,  "found" : true,  "\_source" : {  "account\_number" : 1,  "balance" : 39225,  "firstname" : "Amber",  "lastname" : "Duke",  "age" : 32,  "gender" : "M",  "address" : "880 Holmes Lane",  "employer" : "Pyrami",  "email" : "amberduke@pyrami.com",  "city" : "Brogan",  "state" : "IL"  }  } |
| # ES8 set index pattern in Kibana Management > Stack Management > Kibana > Data Views > Create data view > bank  # ES7 Index pattern in Kibana http://localhost:5601/app/management Management -> Kibana -> Index Patterns write name: bank Index pattern name: bank\* uncheck 'Index contains time-based events'. click 'create' | |
| **03\_03.txt Setting data types**  # Download sample logs from Elastic or use Exercise Files # If necessary unzip/tar logs.jsonl.gz # <https://download.elastic.co/demos/kibana/gettingstarted/logs.jsonl.gz>  # Add mapping for lat/lon geo properties for logs PUT /logstash-2015.05.18 {  "mappings": {  "properties": {  "geo": {  "properties": {  "coordinates": {  "type": "geo\_point"  }  }  }  }  } } the same PUT /logstash-2015.05.19 ... PUT /logstash-2015.05.20 ...  # Check out structure of log data head logs.jsonl  # Import log files curl -XPOST -i -k \ -H "Content-Type: application/x-ndjson" \ -H "Authorization: ApiKey $ES\_API\_KEY" \ $ES\_HOST/\_bulk?pretty --data-binary @logs.jsonl; echo  # OR Import log files curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/\_bulk?pretty' --data-binary @data/logs.jsonl | |
| # Check ES for data (просмотреть все 3 индекса) GET /\_cat/indices/logstash-\* | yellow open logstash-2015.05.20 zSWeMvZ\_ToKqJdcbxOeq\_g 1 1 4750 0 12.2mb 12.2mb  yellow open logstash-2015.05.18 qIpqyDbsQcqkKD21I9YBlQ 1 1 4634 0 12mb 12mb  yellow open logstash-2015.05.19 vqPMvF-vRa-00ZmMUy6GuQ 1 1 4624 0 12mb 12mb |
| # Count in exact index GET logstash-2015.05.18/\_count | {  "count" : 4634,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  } } |
| # Change default index pattern in Kibana  # Load Shakespeare data (download from elastic or find in the Exercise Files) # Check out shakespeare.json # https://download.elastic.co/demos/kibana/gettingstarted/shakespeare\_6.0.json  head shakespeare.json  # Shakespeare Schema {  "line\_id": INT,  "play\_name": "String",  "speech\_number": INT,  "line\_number": "String",  "speaker": "String",  "text\_entry": "String", }  # Create Shakespeare index with data types PUT /shakespeare {  "mappings" : {  "properties" : {  "speaker" : {"type": "keyword" },  "play\_name" : {"type": "keyword" },  "line\_id" : { "type" : "integer" },  "speech\_number" : { "type" : "integer" }  }  } }  # Load Shakespeare data curl -XPOST -i -k \ -H "Content-Type: application/x-ndjson" \ -H "Authorization: ApiKey $ES\_API\_KEY" \ $ES\_HOST/shakespeare/\_bulk?pretty --data-binary @shakespeare.json; echo  # OR $ curl -XPOST -i -k -H "Content-Type: application/x-ndjson" localhost:9200/shakespeare/\_bulk?pretty --data-binary @data/shakespeare.json; echo | |
| # Check out index in ES GET /shakespeare GET /\_cat/indices GET shakespeare/\_doc/2 | {  "\_index" : "shakespeare",  "\_type" : "\_doc",  "\_id" : "2",  "\_version" : 1,  "\_seq\_no" : 2,  "\_primary\_term" : 1,  "found" : true,  "\_source" : {  "type" : "line",  "line\_id" : 3,  "play\_name" : "Henry IV",  "speech\_number" : "",  "line\_number" : "",  "speaker" : "",  "text\_entry" : "Enter KING HENRY, LORD JOHN OF LANCASTER, the EARL of WESTMORELAND, SIR WALTER BLUNT, and others"  }  } |
| **4) Querying Data**  **04\_01\_\_QueryingData.txt Simple queries** |  |
| # show me everything GET bank/\_search | {  "took" : 8,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : 1.0,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "1",  "\_score" : 1.0,  "\_source" : {  "account\_number" : 1,  "balance" : 39225,  "firstname" : "Amber",  "lastname" : "Duke",  "age" : 32, ... |
| # find CA accounts only GET bank/\_search {  "query": {  "match": {  "state": "CA"  }  } } | {  "took" : 6,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 17,  "relation" : "eq"  },  "max\_score" : 4.046554,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "68",  "\_score" : 4.046554,  "\_source" : {  "account\_number" : 68,  "balance" : 44214,  "firstname" : "Hall",  "lastname" : "Key",  "age" : 25,  "gender" : "F",  "address" : "927 Bay Parkway",  "employer" : "Eventex",  "email" : "hallkey@eventex.com",  "city" : "Shawmut",  "state" : "CA"  }  },  ... |
| # find "Techade" accounts in CA only GET bank/\_search {  "query": {  "bool": {  "must": [  { "match": {"state": "CA"} },  { "match": {"employer": "Techade"}}  ]  }  } } | {  "took" : 71,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 1,  "relation" : "eq"  },  "max\_score" : 10.552503,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "413",  "\_score" : 10.552503,  "\_source" : {  "account\_number" : 413,  "balance" : 15631,  "firstname" : "Pugh",  "lastname" : "Hamilton",  "age" : 39,  "gender" : "F",  "address" : "124 Euclid Avenue",  "employer" : "Techade",  "email" : "pughhamilton@techade.com",  "city" : "Beaulieu",  "state" : "CA"  }  }  ... |
| # find non "Techade" accounts outside of CA (find opposite result) GET bank/\_search {  "query": {  "bool": {  "must\_not": [  { "match": {"state": "CA"} },  { "match": {"employer": "Techade"}}  ]  }  } } | {  "took" : 26,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 983,  "relation" : "eq"  },  "max\_score" : 0.0,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "1",  "\_score" : 0.0,  "\_source" : {  "account\_number" : 1,  "balance" : 39225,  "firstname" : "Amber",  "lastname" : "Duke",  "age" : 32,  "gender" : "M",  "address" : "880 Holmes Lane",  "employer" : "Pyrami",  "email" : "amberduke@pyrami.com",  "city" : "Brogan",  "state" : "IL"  }  }, |
| # let's combine them to search for non "Techade" accounts inside CA GET bank/\_search {  "query": {  "bool": {  "must": [  { "match": { "state": "CA" }}  ],   "must\_not": [  { "match": {"employer": "Techade"}}  ]  }  } } | {  "took" : 5,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 16,  "relation" : "eq"  },  "max\_score" : 4.046554,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "68",  "\_score" : 4.046554,  "\_source" : {  "account\_number" : 68,  "balance" : 44214,  "firstname" : "Hall",  "lastname" : "Key",  "age" : 25,  "gender" : "F",  "address" : "927 Bay Parkway",  "employer" : "Eventex",  "email" : "hallkey@eventex.com",  "city" : "Shawmut",  "state" : "CA"  }  }, |
| # Boost results for Smith GET bank/\_search {  "query": {  "bool": {  "should": [  { "match": {"state": "CA"} },  { "match": {  "lastname": {  "query": "Smith",  "boost": 3  }  }  }  ]  }  } } | {  "took" : 31,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 18,  "relation" : "eq"  },  "max\_score" : 19.509869,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "516",  "\_score" : 19.509869,  "\_source" : {  "account\_number" : 516,  "balance" : 44940,  "firstname" : "Roy",  **"lastname" : "Smith",**  "age" : 37,  "gender" : "M",  "address" : "770 Cherry Street",  "employer" : "Parleynet",  "email" : "roysmith@parleynet.com",  "city" : "Carrsville",  **"state" : "RI"**  }  }, |
| **04\_02\_\_QueryingData.txt Term-level queries** | |
| # Term Query (for numeric values, key words) GET bank/\_search {  "query": {  "term": {  "account\_number": 516  }  } }  # Or  GET bank/account/\_search  {  "query": {  "term": {  "account\_number": 516  }  }  } | {  "took" : 2,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 1,  "relation" : "eq"  },  "max\_score" : 1.0,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "516",  "\_score" : 1.0,  "\_source" : {  **"account\_number" : 516,**  "balance" : 44940,  "firstname" : "Roy",  "lastname" : "Smith",  "age" : 37,  "gender" : "M",  "address" : "770 Cherry Street",  "employer" : "Parleynet",  "email" : "roysmith@parleynet.com",  "city" : "Carrsville",  "state" : "RI"  }  }  ]  }  } |
| # Returns null because "state" is a text field (hence not an exact match) GET bank/\_search {  "query": {  "term": {  "state": "RI"  }  } }  # OR  GET bank/account/\_search ... | {  "took" : 1,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  **"value" : 0,**  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  }  } |
| # This works because it uses the "analysis" process GET bank/\_search {  "query": {  "match": {  "state": "RI"  }  } }  # OR  GET bank/account/\_search  ... | "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "49",  "\_score" : 3.990984,  "\_source" : {  "account\_number" : 49,  "balance" : 29104,  "firstname" : "Fulton",  "lastname" : "Holt",  "age" : 23,  "gender" : "F",  "address" : "451 Humboldt Street",  "employer" : "Anocha",  "email" : "fultonholt@anocha.com",  "city" : "Sunriver",  "state" : "RI"  }  }, |
| # Terms can return multiple results (terms позволяют делать range запросы) GET bank/\_search {  "query": {  "terms": {  "account\_number": [516,851]  }  } }  #OR  GET bank/account/\_search  ... | "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "516",  "\_score" : 1.0,  "\_source" : {  **"account\_number" : 516,**  "balance" : 44940,  "firstname" : "Roy",  "lastname" : "Smith",  "age" : 37,  "gender" : "M",  "address" : "770 Cherry Street",  "employer" : "Parleynet",  "email" : "roysmith@parleynet.com",  "city" : "Carrsville",  "state" : "RI"  }  },  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "851",  "\_score" : 1.0,  "\_source" : {  **"account\_number" : 851,**  "balance" : 22026, |
| # Range Queries ## gte = Greater-than or equal to ## gt = Greater-than ## lte = Less-than or equal to ## lt = Less-than | |
| # Show all accounts between 516 and 851, boosting the importance GET bank/\_search {  "query": {  "range": {  "account\_number": {  "gte": 516,  "lte": 851,  "boost": 2  }  }  } } | "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "518",  "\_score" : 2.0,  "\_source" : {  **"account\_number" : 518,**  "balance" : 48954,  "firstname" : "Finch",  "lastname" : "Curtis",  "age" : 29,  "gender" : "F",  "address" : "137 Ryder Street",  "employer" : "Viagrand",  "email" : "finchcurtis@viagrand.com",  "city" : "Riverton",  "state" : "MO"  }  },  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "520",  "\_score" : 2.0,  "\_source" : {  **"account\_number" : 520,**  "balance" : 27987,  "firstname" : "Brandy",  "lastname" : "Calhoun",  "age" : 32,  "gender" : "M",  "address" : "818 Harden Street",  "employer" : "Maxemia",  "email" : "brandycalhoun@maxemia.com",  "city" : "Sidman",  "state" : "OR"  } |
| # Show all account holders older than 35 GET bank/\_search {  "query": {  "range": {  "age": {  "gt": 35  }  }  } } | "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "6",  "\_score" : 1.0,  "\_source" : {  "account\_number" : 6,  "balance" : 5686,  "firstname" : "Hattie",  "lastname" : "Bond",  **"age" : 36,**  "gender" : "M",  "address" : "671 Bristol Street",  "employer" : "Netagy",  "email" : "hattiebond@netagy.com",  "city" : "Dante",  "state" : "TN"  }  }, |
| **04\_03\_\_QueryingData.txt Analysis and tokenization** | |
| # Basic Example GET bank/\_analyze {  "tokenizer" : "standard",  "text" : "The Moon is Made of Cheese Some Say" } | {  "tokens" : [  {  "token" : "The",  "start\_offset" : 0,  "end\_offset" : 3,  "type" : "<ALPHANUM>",  "position" : 0  },  {  "token" : "Moon",  "start\_offset" : 4,  "end\_offset" : 8,  "type" : "<ALPHANUM>",  "position" : 1  },  {  "token" : "is",  "start\_offset" : 9,  "end\_offset" : 11,  "type" : "<ALPHANUM>",  "position" : 2  }, |
| # Mixed String GET bank/\_analyze {  "tokenizer" : "standard",  "text" : "The Moon-is-Made of Cheese.Some Say$" } | {  "tokens" : [  ...  {  "token" : "Cheese.Some",  "start\_offset" : 20,  "end\_offset" : 31,  "type" : "<ALPHANUM>",  "position" : 5  },  {  "token" : "Say",  "start\_offset" : 32,  "end\_offset" : 35,  "type" : "<ALPHANUM>",  "position" : 6  }  ]  } |
| # Uset the letter tokenizer GET bank/\_analyze {  "tokenizer" : "letter",  "text" : "The Moon-is-Made of Cheese.Some Say$" } | ...  {  "token" : "Cheese",  "start\_offset" : 20,  "end\_offset" : 26,  **"type" : "word",**  "position" : 5  },  {  "token" : "Some",  "start\_offset" : 27,  "end\_offset" : 31,  **"type" : "word",**  "position" : 6  },  {  "token" : "Say",  "start\_offset" : 32,  "end\_offset" : 35,  **"type" : "word",**  "position" : 7  }  ]  } |
| # How about a URL GET bank/\_analyze {  "tokenizer": "uax\_url\_email",  "text": "you@example.com login at https://bensullins.com attempt" } | {  "tokens" : [  {  **"token" : "you@example.com",**  "start\_offset" : 0,  "end\_offset" : 15,  **"type" : "<EMAIL>",**  "position" : 0  },  {  **"token" : "login",**  "start\_offset" : 16,  "end\_offset" : 21,  **"type" : "<ALPHANUM>",**  "position" : 1  },  {  **"token" : "https://bensullins.com",**  "start\_offset" : 25,  "end\_offset" : 47,  **"type" : "<URL>",**  "position" : 3  }  ... |
| # Where it breaks, two fields with diff analyzers PUT /idx1 {  "mappings": {  "properties": {  "title": {  "type": "text",  "analyzer" : "standard"  },  "english\_title": {  "type": "text",  "analyzer": "english"  }  }  } } |  |
| GET idx1 | {  "idx1" : {  "aliases" : { },  "mappings" : {  "properties" : {  **"english\_title"** : {  "type" : "text",  **"analyzer" : "english"**  },  **"title"** : {  "type" : "text",  **"analyzer" : "standard"**  }  }  ... |
| GET idx1/\_analyze {  **"field": "title",**  "text": "Bears" } | {  "tokens" : [  {  **"token" : "bears",**  "start\_offset" : 0,  "end\_offset" : 5,  "type" : "<ALPHANUM>",  "position" : 0  }  ]  } |
| GET idx1/\_analyze  {  **"field": "english\_title",**  "text": "Bears"  } | {  "tokens" : [  {  **"token" : "bear",**  "start\_offset" : 0,  "end\_offset" : 5,  "type" : "<ALPHANUM>",  "position" : 0  }  ]  } |
| **5) Analyzing Your Data**  **05\_01\_AnalyzingYourData.txt Basic aggregations** | |
| # Count of Accounts by State # Must be keyword field GET bank/\_search {  "size": 0,  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  }  }  } } | {  "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30  },  {  "key" : "MD",  "doc\_count" : 28  }, |
| # Add average balance in each state # Nesting the metric inside the agg (агрегация внутри другой агрегации) (взяли штат TX в нем 30 документов, в этих 30 – avg\_balance - 26073.3) GET bank/\_search {  "size": 0,  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  },  "aggs": {  "avg\_bal": {  "avg": {  "field": "balance"  }  }  }  }  } } | {  "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30,  **"avg\_bal" : {**  **"value" : 26073.3**  **}**  },  {  "key" : "MD",  "doc\_count" : 28,  "avg\_bal" : {  "value" : 26161.535714285714  }  }, |
| # Breakdown further with Nesting GET bank/\_search {  "size": 0,  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  },  "aggs": {  "avg\_bal": {  "avg": {  "field": "balance"  }  },  "age":{  "terms": {  "field": "age"  }  }  }  }  } } | "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30,  "avg\_bal" : {  "value" : 26073.3  },  "age" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 4,  "buckets" : [  **{**  **"key" : 21,**  **"doc\_count" : 5**  **},**  {  "key" : 24,  "doc\_count" : 3  }, |
| # Add avg\_price metric to lowest level GET bank/\_search {  "size": 0,  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  },  "aggs": {  "avg\_bal": {  "avg": {  "field": "balance"  }  },  "age":{  "terms": {  "field": "age"  },  "aggs": {"avg\_bal": {"avg": {"field": "balance"} }  }  }  }  }  } } | "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30,  "avg\_bal" : {  "value" : 26073.3  },  "age" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 4,  "buckets" : [  {  "key" : 21,  "doc\_count" : 5,  **"avg\_bal" : {**  **"value" : 17251.2**  **}**  },  {  "key" : 24,  "doc\_count" : 3,  "avg\_bal" : {  "value" : 37200.333333333336  }  }, |
| ## Get stats about bank balances ## Size=1 to omit search results (для примера покажет один результат) GET bank/\_search {  "size": 1,  "aggs": {  "balance-stats": {  "stats": {  "field": "balance"  }  }  } } | {  "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : 1.0,  "hits" : [  {  "\_index" : "bank",  "\_type" : "account",  "\_id" : "1",  "\_score" : 1.0,  "\_source" : {  "account\_number" : 1,  "balance" : 39225,  "firstname" : "Amber",  "lastname" : "Duke",  "age" : 32,  "gender" : "M",  "address" : "880 Holmes Lane",  "employer" : "Pyrami",  "email" : "amberduke@pyrami.com",  "city" : "Brogan",  "state" : "IL"  }  }  ]  },  "aggregations" : {  **"balance-stats" : {**  "count" : 1000,  "min" : 1011.0,  "max" : 49989.0,  "avg" : 25714.837,  "sum" : 2.5714837E7  }  }  } |
| **05\_02\_AnalyzingYourData.txt Filtering aggregations** | |
| # Count of Accounts by State # Must be keyword field GET bank/\_search {  "size": 0,  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  }  }  } }  # This is the equivalent of using match\_all GET bank/\_search {  "size": 0,  "query": {  "match\_all": {}  },  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  }  }  } } | "hits" : {  "total" : {  **"value" : 1000,**  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30  },  {  "key" : "MD",  "doc\_count" : 28  }, |
| # Aggs work in the context of the query, so apply a filter like normal GET bank/\_search {  "size": 0,  "query": {  "match": {  "state.keyword": "CA"  }  },  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  }  }  } } | "hits" : {  "total" : {  **"value" : 17,**  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 0,  "buckets" : [  {  "key" : "CA",  "doc\_count" : 17  }  ]  }  } |
| # You can also filter on terms GET bank/\_search {  "size": 0,  "query": {  "bool": {  "must": [  {"match": {"state.keyword": "CA"}},  {"range": {"age": {"gt": 35}}}  ]  }  },  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  }  }  } } | "hits" : {  "total" : {  **"value" : 6,**  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 0,  "buckets" : [  {  "key" : "CA",  "doc\_count" : 6  }  ]  }  } |
| # Lets add a metric back in GET bank/\_search {  "size": 0,  "query": {  "bool": {  "must": [  {"match": {"state.keyword": "CA"}},  {"range": {"age": {"gt": 35}}}  ]  }  },  "aggs": {  "states": {  "terms": {  "field": "state.keyword"  },  "aggs": {"avg\_bal": {"avg": {"field": "balance"} }}  }  } } | "hits" : {  "total" : {  "value" : 6,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "states" : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 0,  "buckets" : [  {  "key" : "CA",  "doc\_count" : 6,  **"avg\_bal" : {**  "value" : 22794.166666666668  }  }  ]  }  } |
| # You can also just filter the results GET bank/account/\_search {  "size": 0,  "query": {  "match": {"state.keyword": "CA"}  },  "aggs": {  "over35":{  "filter": {  "range": {"age": {"gt": 35}}  },  "aggs": {"avg\_bal": {"avg": {"field": "balance"} }}  }  } } | "hits" : {  "total" : {  "value" : 17,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "over35" : {  "doc\_count" : 6,  "avg\_bal" : {  "value" : 22794.166666666668  }  }  } |
| # Look at state avg and global average (parallel aggregation) GET bank/\_search {  "size": 0,  "aggs": {  "state\_avg": {  "terms": {  "field": "state.keyword"  },  "aggs": {"avg\_bal": {"avg": {"field": "balance"}}}  },  "global\_avg": {  **"global": {},**  "aggs": {"avg\_bal": {"avg": {"field": "balance"}}}  }  } } | "hits" : {  "total" : {  **"value" : 1000,**  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  **"global\_avg"** : {  **"doc\_count" : 1000,**  "avg\_bal" : {  "value" : 25714.837  }  },  **"state\_avg"** : {  "doc\_count\_error\_upper\_bound" : 0,  "sum\_other\_doc\_count" : 743,  "buckets" : [  {  "key" : "TX",  "doc\_count" : 30,  "avg\_bal" : {  "value" : 26073.3  }  },  {  "key" : "MD",  "doc\_count" : 28,  "avg\_bal" : {  "value" : 26161.535714285714  }  }, |
| **05\_03\_AnalyzingYourData.txt Percentiles and histograms** | |
| # Look at the percentiles for the balances GET bank/\_search {  "size": 0,  "aggs": {  "pct\_balances": {  "percentiles": {  "field": "balance",  "percents": [  1,  5,  25,  50,  75,  95,  99  ]  }  }  } } | "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  **"pct\_balances"** : {  "values" : {  "1.0" : 1455.0,  "5.0" : 3590.5,  "25.0" : 13713.214285714286,  "50.0" : 25999.375,  "75.0" : 38186.5,  "95.0" : 47601.5,  "99.0" : 49347.0  }  }  } |
| # Can also calculate High Dynamic Range (HDR) Historgram GET bank/account/\_search {  "size": 0,  "aggs": {  "pct\_balances": {  "percentiles": {  "field": "balance",  "percents": [  1,  5,  25,  50,  75,  95,  99  ],  "hdr": {  "number\_of\_significant\_value\_digits": 3  }  }  }  } } | "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "pct\_balances" : {  "values" : {  "1.0" : 1447.5,  "5.0" : 3589.5,  **"25.0" : 13679.5,**  "50.0" : 26015.5,  "75.0" : 38175.5,  "95.0" : 47551.5,  "99.0" : 49343.5  }  }  } |
| # We can use the percentile ranks agg for checking an individual values GET bank/\_search {  "size": 0,  "aggs": {  "bal\_outlier": {  "percentile\_ranks": {  "field": "balance",  "values": [35000,50000]  }  }  } } | "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  "bal\_outlier" : {  "values" : {  "35000.0" : 69.06507651615341,  "50000.0" : 100.0  }  }  } |
| # Similarly we can create a histogram  GET bank/\_search {  "size": 0,  "aggs": {  "bals": {  "histogram": {  "field": "balance",  "interval": 500  }  }  } } | "hits" : {  "total" : {  "value" : 1000,  "relation" : "eq"  },  "max\_score" : null,  "hits" : [ ]  },  "aggregations" : {  **"bals"** : {  "buckets" : [  {  "key" : 1000.0,  "doc\_count" : 13  },  {  "key" : 1500.0,  "doc\_count" : 6  },  {  "key" : 2000.0,  "doc\_count" : 6  },  {  "key" : 2500.0,  "doc\_count" : 7  },  {  "key" : 3000.0,  "doc\_count" : 16  }, |

**6) Presenting Your Insights**

**a) Kibana overview and setup**

Graphical user interface, text, application

Description automatically generated

Then open <http://localhost:5601/app/discover#/>

Graphical user interface, text, application, email

Description automatically generated

Then select date range

Chart

Description automatically generated

**b) Creating visualizations in Kibana**

<http://localhost:5601/app/visualize#/>

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

How many visits site over time:

Chart, histogram

Description automatically generated

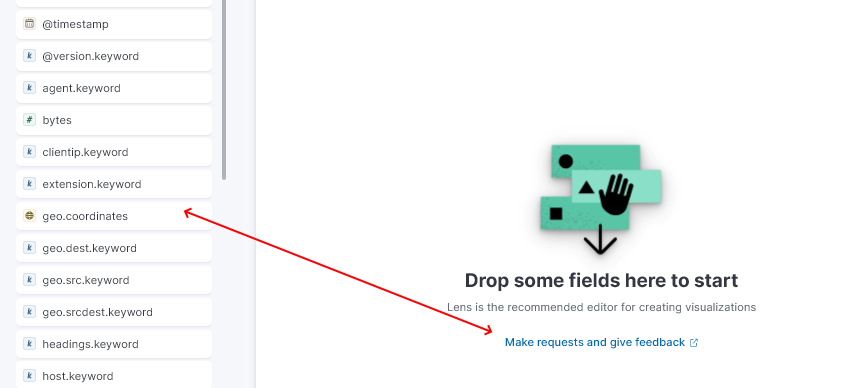
Chart

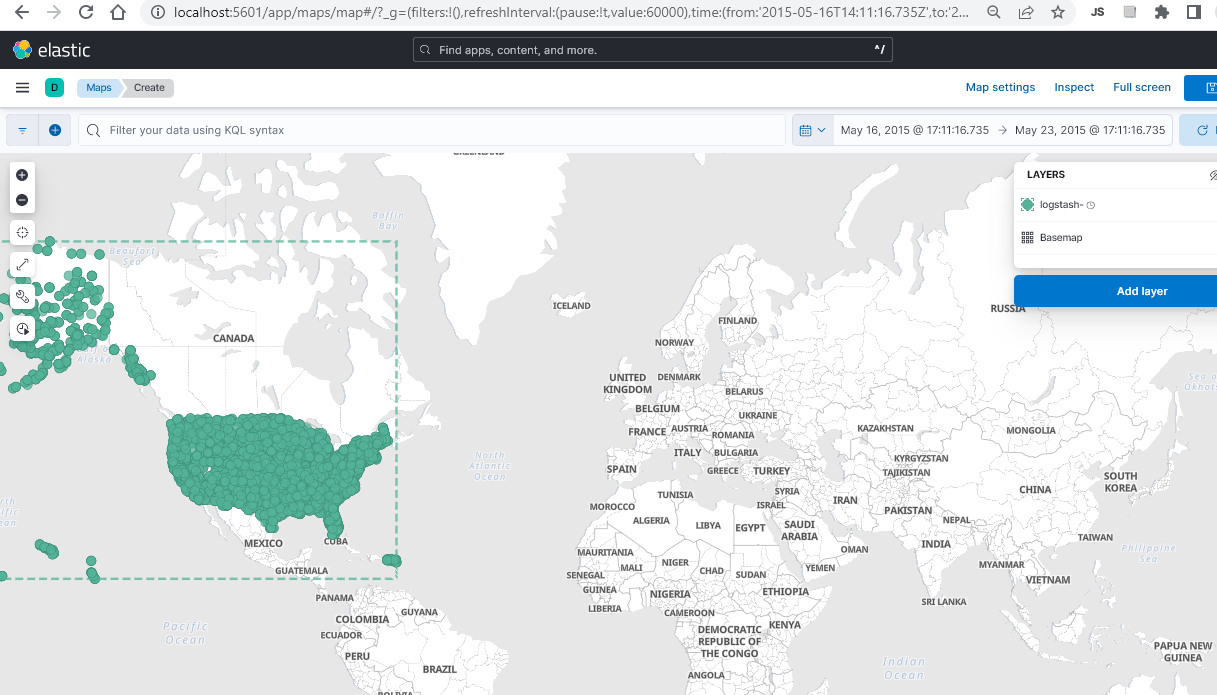
Description automatically generated

Graphical user interface, application

Description automatically generated

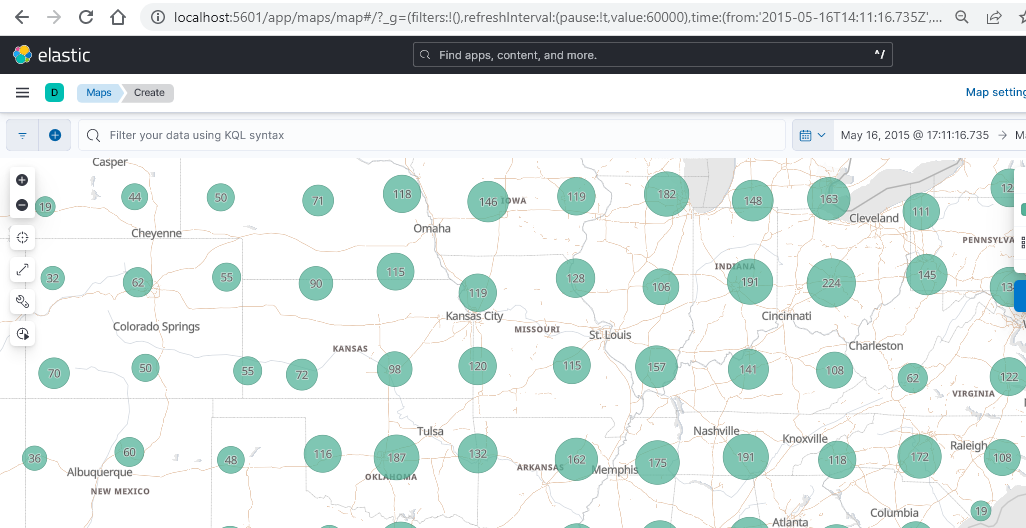
- show with map





Graphical user interface

Description automatically generated



* Shows records

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

Another metric:

Graphical user interface

Description automatically generated

Another metric:

Graphical user interface, application

Description automatically generated

Another metric:

Graphical user interface, text, application, email

Description automatically generated

**c) Creating dashboards in Kibana**

<http://localhost:5601/app/dashboards#/create>

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated