Storing logs with Elastic

- logstash -e “input { stdin { } } output { elasticsearch { hosts => localhost } }”

- After starting type some message

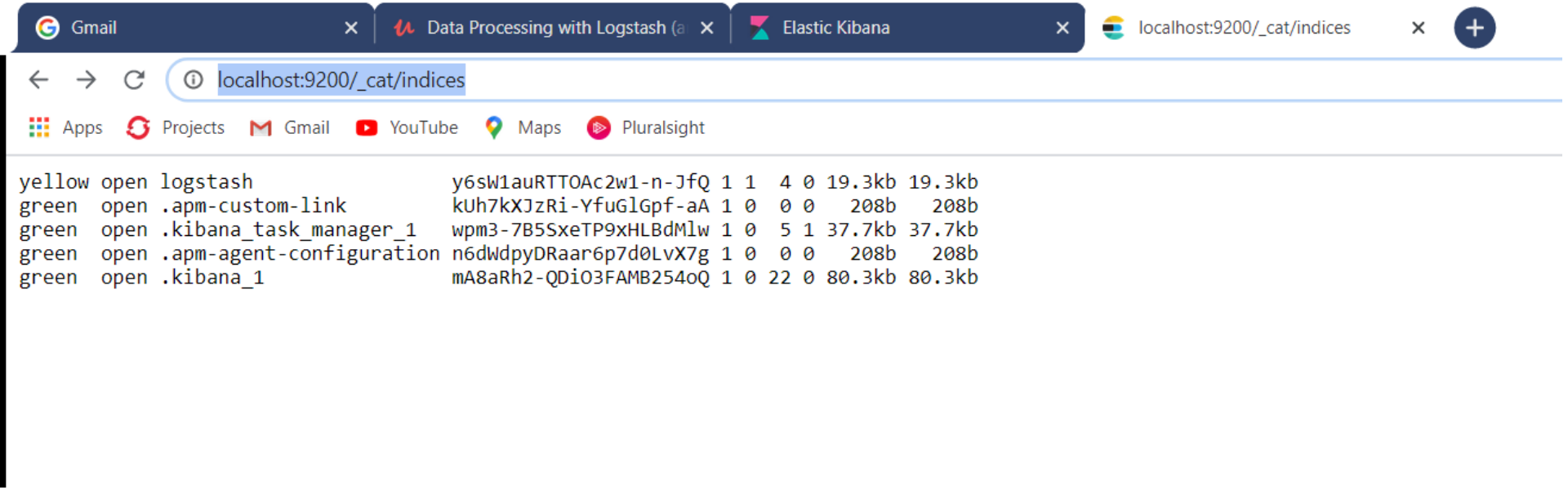
Storing logs with Elastic

```
refresh_interval"=>"5s"}, "mappings"=>{"_default_"=>{"dynamic_templates"=>[{"message_field
"=>{"path_match"=>"message", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "
norms"=>false}}}, {"string_fields"=>{"match"=>"*", "match_mapping_type"=>"string", "mappin
g"=>{"type"=>"text", "norms"=>false, "fields"=>{"keyword"=>{"type"=>"keyword", "ignore_abo
ve"=>256}}}}}], "properties"=>{"@timestamp"=>{"type"=>"date"}, "@version"=>{"type"=>"keywo
rd"}, "geoip"=>{"dynamic"=>true, "properties"=>{"ip"=>{"type"=>"ip"}, "location"=>{"type"=
>"geo_point"}, "latitude"=>{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}
}}}
[2018-12-16T21:19:35,813][INFO ][logstash.outputs.elasticsearch] Installing elasticsearch
template to _template/logstash
[2018-12-16T21:19:38,135][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output
{:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["//localhost"]}
[2018-12-16T21:19:38,255][INFO ][logstash.pipeline        ] Pipeline started successfully
{:pipeline_id=>"main", :thread=>"#<Thread:0xf28c60 run>"}
The stdin plugin is now waiting for input:
[2018-12-16T21:19:38,348][INFO ][logstash.agent           ] Pipelines running {:count=>1,
:running_pipelines=>[:main], :non_running_pipelines=>[]}
[2018-12-16T21:19:38,630][INFO ][logstash.agent           ] Successfully started Logstash
API endpoint {:port=>9600}
Hi
How are you
```

Storing logs with Elastic

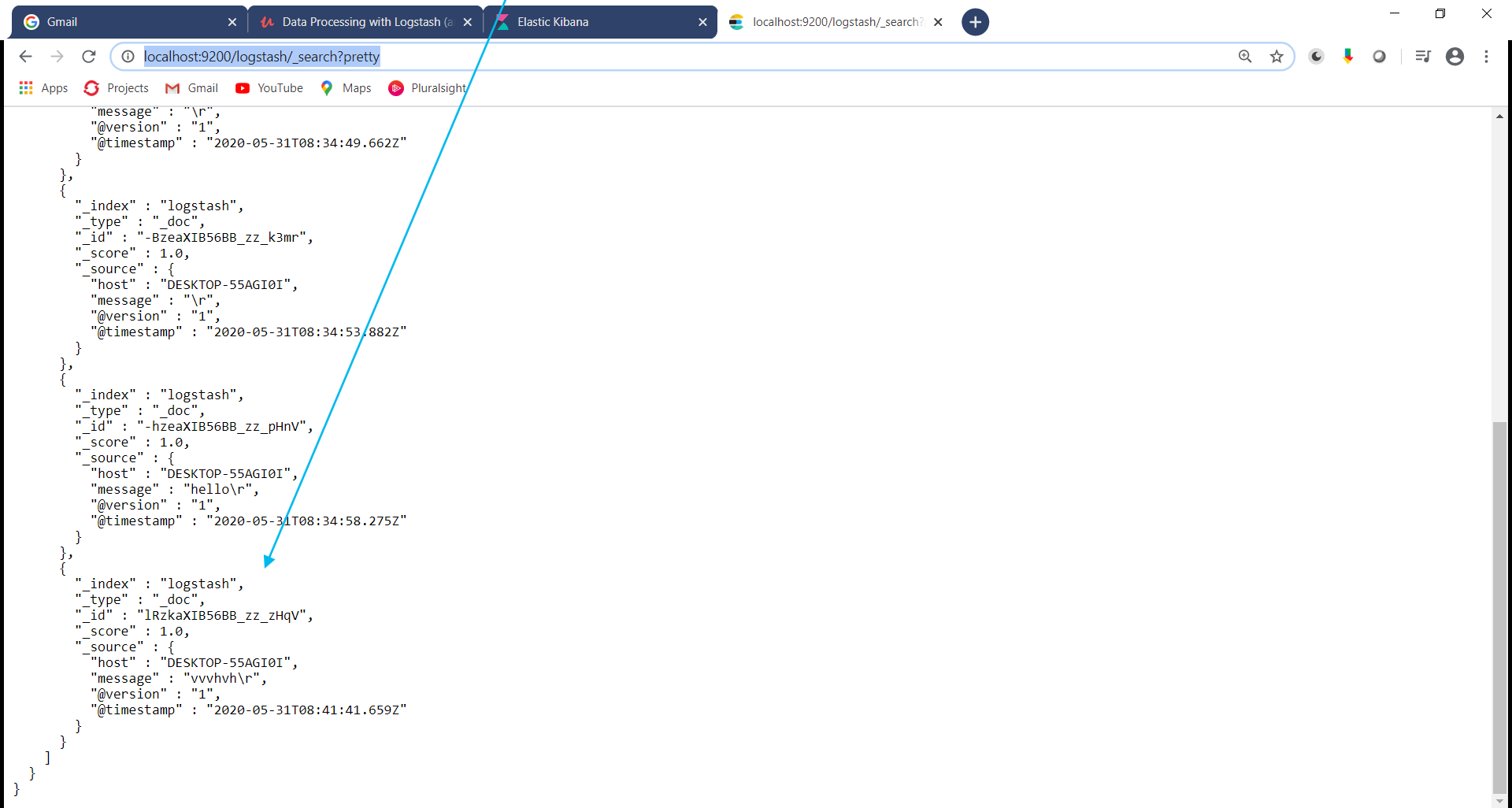
http://localhost:9200/_cat/indices

http://localhost:9200/_cat/indices?v&health=yellow



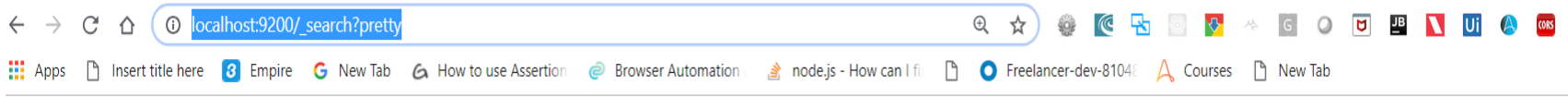
Storing logs with Elastic

http://localhost:9200/logstash/_search?pretty



Storing logs with Elastic

http://localhost:9200/_search?pretty




```
{
  "took" : 58,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "logstash-2018.12.16",
        "_type" : "doc",
        "_id" : "ZZG2t2cBNWJlpFD3QMCG",
        "_score" : 1.0,
        "_source" : {
          "message" : "Hi\r",
          "@timestamp" : "2018-12-16T15:50:31.020Z",
          "host" : "DESKTOP-55AGI0I",
          "@version" : "1"
        }
      }
    ]
  }
}
```

Delete All from Elastic

- curl -X DELETE http://localhost:9200/_all


```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 0,
    "successful" : 0,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : 0.0,
    "hits" : [ ]
  }
}
```

Capturing Tomcat logs

```
D:\logstash-6.3.0\bin>logstash --verbose -f "D:\logstash configurations\pipeline.conf"
Sending Logstash's logs to D:/logstash-6.3.0/logs which is now configured via log4j2.prope
rties
[2018-12-16T21:52:36,947][WARN ][logstash.config.source.multilocal] Ignoring the 'pipeline
s.yml' file because modules or command line options are specified
[2018-12-16T21:52:37,388][INFO ][logstash.runner          ] Starting Logstash {"logstash.v
ersion"=>"6.3.0"}
[2018-12-16T21:52:38,921][INFO ][logstash.pipeline        ] Starting pipeline {:pipeline_i
d=>"main", "pipeline.workers"=>8, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50}
[2018-12-16T21:52:39,472][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs u
pdated {:changes=>{:removed=>[], :added=>[http://localhost:9200/]}}
[2018-12-16T21:52:39,481][INFO ][logstash.outputs.elasticsearch] Running health check to s
ee if an Elasticsearch connection is working {:healthcheck_url=>http://localhost:9200/, :p
ath=>"/"}
[2018-12-16T21:52:39,669][WARN ][logstash.outputs.elasticsearch] Restored connection to ES
 instance {:url=>"http://localhost:9200/"}
[2018-12-16T21:52:39,729][INFO ][logstash.outputs.elasticsearch] ES Output version determi
ned {:es_version=>6}
[2018-12-16T21:52:39,733][WARN ][logstash.outputs.elasticsearch] Detected a 6.x and above
cluster: the `type` event field won't be used to determine the document _type {:es_version
=>6}
```

Capturing Tomcat logs


```
{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 46,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "logstash-2018.12.16",
        "_type" : "doc",
        "_id" : "MjHdt2cBiVdjzqqkDY82",
        "_score" : 1.0,
        "_source" : {
          "@version" : "1",
          "@timestamp" : "2018-12-16T16:32:56.520Z",
          "message" : "How are you\r",
          "host" : "DESKTOP-55AGI0I"
        }
      },
      {
        "_index" : "logstash-2018.12.16",
        "_type" : "doc",
        "_id" : "ODHet2cBiVdjzqqkBY_S",
        "_score" : 1.0,
        "_source" : {
          "@version" : "1",
          "@timestamp" : "2018-12-16T16:34:00.032Z",
          "message" : "28-Apr-2018 04:41:26.094 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:            C:\\Program Files\\Java\\jre1.8.0_161\r",
          "path" : "C:/Program Files/Apache Software Foundation/Tomcat 9.0/logs/catalina.2018-04-28.log",
          "host" : "DESKTOP-55AGI0I"
        }
      },
      {
        "_index" : "logstash-2018.12.16",
        "_type" : "doc",
        "_id" : "OTHet2cBiVdjzqqkBY_S",
```

Shipping, Filtering, and Parsing Events with Logstash

- Sample Dataset
- This dataset can be downloaded from the following location:
- http://cdiac.ornl.gov/ftp/us_recordtemps/sta424/tmax_serial/CA_6719tmax.txt

Shipping, Filtering, and Parsing Events with Logstash

- The sample data has the following fields:
- Station Number (i.e., 046719: two-digit state code, followed by
- four-digit station code)
- Day of Year (1-365)
- Year
- Month
- Day of Month
- Tmax (Maximum Temperature)

Shipping, Filtering, and Parsing Events with Logstash

Station Number	Day of Year	Year	Month	Day of Month	Maximum Temperature
046719	1	1986	1	1	62
046719	2	1986	1	2	63
046719	3	1986	1	3	65
046719	4	1986	1	4	66
046719	5	1986	1	5	66
046719	6	1986	1	6	73
046719	7	1986	1	7	74
046719	8	1986	1	8	72
046719	9	1986	1	9	76
046719	10	1986	1	10	82

Shipping, Filtering, and Parsing Events with Logstash

- 1,1986-01-01,62
- 2,1986-01-02,63
- 3,1986-01-03,65
- 4,1986-01-04,66
- 5,1986-01-05,66
- 6,1986-01-06,73
- 7,1986-01-07,74
- 8,1986-01-08,72
- 9,1986-01-09,76

Logstash Configuration

- Before proceeding with log analysis, Logstash has to be configured to accept input from a particular source and in a particular format.
- In order to read, parse, and filter different types of data, Logstash enables you to specify different types of inputs, outputs, and filters.
- This is facilitated by a diverse set of plugins. In order to read data from a file, the **file** plugin can be used.

Logstash Configuration

- Each line in the source file is treated as a separate event and streamed by the file input plugin.
- In running systems, typically the log files rotate. The file input plugin has the ability to detect file rotation and handle it accordingly.
- This it does by maintaining the last read location.
- New data is automatically detected if the correct configuration is done..

Logstash Configuration

- input {
- file {
- path => #String (path of the files) (required)
- start_position => #String (optional, default "end")
- tags => #array (optional)
- type => #string (optional)
- }
- }

Logstash Configuration

- **path**: It is the only mandatory field in the file input plugin and is used to specify the path of the file from where input events have to be received and processed.
- **start_position**: Logstash can start reading from any point in the source file and this point is specified by **“start_position”**.
- It can take the value of **“beginning”** or **“end”**. In order to read live streams, specify the default value of “end”.
- Only if you want to read any historical data do you need to specify a value as “beginning”.

Logstash Configuration

- **tags**: This field helps in filtering and processing events. Any number of filter strings can be specified as an array for this purpose.
- **type**: In order to mark a specific type of events, you can categorize a specific type of events by using this field. Type is added to a document that is indexed in Elasticsearch.
- It can be later viewed in Kibana under the **_type** field. For example, you can assign type as **“critical”** or **“warning”**.

Logstash Configuration

- The configuration varies according to the plugin type. Often there are cases where a plugin may require the value to be of a certain type such as string or array. The following value types are supported:
- **Array**: If you want to specify multiple values, then use the array type. Specifying the same setting multiple times appends to the array.
- For Example:
- path => ["/var/log/*.log", "/var/log/postgresql/*.log"]
- path => "/var/log/apache/*.log"
- This example specifies **path** to be an array with an element for each of the strings.

Logstash Configuration

- **Boolean**: The value of a Boolean type can be either **true** or **false**. Take care that the true and false keywords are not within quotes.
- For example,
- ssl_enable => false
- **Bytes**: This field is a string field representing a valid unit of bytes.
- It provides a convenient way to use specific sizes in plugin options. It support both **base-1000 SI** (k M G T P E Z Y) and **base-1024 Binary** (Ki Mi Gi Ti Pi Ei Zi Yi) units. Not only are these fields case-insensitive but they also accept spaces between value and unit. If no unit is specified, the integer string stands for the number of bytes.

Logstash Configuration

- **Codec**: This field represents name of the Logstash codec being used for input or output.
- **Input** codecs facilitate decoding data before actual processing.
- **Output** codecs facilitate encoding data before outputting it. By using an input or output codec, you eliminate the need for using a separate filter. See the following for an example:
- codec => "plain"
- **Hash**: A collection of **key value pairs** in the form “field1” => “value1”. The comma separator is used to separate multiple key value entries. See the following for an example:
- match => {
- "key1" => "value1"
- "key2" => "value2"
- ...
- }

Logstash Configuration

- **Codec**: This field represents name of the Logstash codec being used for input or output.
- **Input** codecs facilitate decoding data before actual processing.
- **Output** codecs facilitate encoding data before outputting it. By using an input or output codec, you eliminate the need for using a separate filter. See the following for an example:
- codec => "plain"
- **Hash**: A collection of **key value pairs** in the form “field1” => “value1”. The comma separator is used to separate multiple key value entries. See the following for an example:
- match => {
- "key1" => "value1"
- "key2" => "value2"
- ...
- }

Logstash Configuration

- **Number**: It represents valid **numeric** values (floating point or integer). For example, num_descriptor => 25
- • **Password**: Represents a string that will be neither **logged** nor **printed**. For example,
- admin_password => "password"
- • **Path**: Used to specify a valid system path. For example, log_path => /var/opt/log
- **String**: String values are single character sequences enclosed in **quotes** (double or single). You need to use the **backslash** to escape literal quotes if they are of the same kind as the string delimiter. You need to escape both double quotes within a double-quoted string and single quotes within a single-quoted string. For example,
- name => "Good Bye"
- name => 'It\'s a hot afternoon'
- name => "I like \"red\" shirts

Logstash Configuration

- Comments
- You can put comments in the configuration file in the same way as is done in Perl, Ruby, and Python. You can start the comment with a # **character**, and it can be in any position in a line, like so:
- # Comment from start of the line
- input { # Comment at the end of line
- # ...
- }

Logstash Configuration

- Configuring for Events
- The Logstash event pipeline consists of three stages: **inputs**, **filters**, and **outputs**. Events are generated by inputs and modified by filters. Events are shipped by outputs.
- The event properties are referred to as **fields** by Logstash. For example, an HTTP request has an HTTP verb like GET or PUT. Event-specific configuration can be done in Logstash.
- Since events are generated by inputs, the event-specific configuration applies only after the input phase. The event-specific configuration works only within filter and output blocks.

Logstash Configuration

- Field References
- It can be more intuitive to refer to a field by name, and this is exactly what the Logstash field reference syntax achieves. You can access a field by **[fieldname]**.
- For **top-level fields**, just **omit** the [] and simply use fieldname. For **nested fields**, specify the full path to that field: **[top-level-field][nested field]**. For example, the following event has two nested fields and one top level field:

{

"network": {

"ip": ["192.168.1.21"],

"timeout": 20,

},

"path": "/var/log/syslog"

}

- To reference the **timeout** field, you specify **[network][timeout]**. To reference a toplevel
- field such as **path**, just specify the field name.

Logstash Configuration

- sprintf Format
- You can use field reference format in sprintf format also. This way you can refer to field values from within other strings.
- For example, the statsd output can increment field values like timeout.

```
output {
statsd {
increment => "syslog.%{[response][status]}"
}
```

Logstash Configuration

- Conditionals
- In certain scenarios, you may want to filter or output an event under certain conditions.
- This is where you can use a conditional. Logstash conditionals are pretty similar to their programming language counterpart. Logstash supports if, else, and else statements, which can be nested also.
- The syntax looks like the following:

```
if EXPRESSION {

...

} else if EXPRESSION {

...

} else {

...

}
```

Logstash Configuration

- Comparison uses Boolean logic to arrive at the correct path. The following
- comparison operators can be used:
- The **equality** operators are ==, !=, <, >, <=, >=.
- The **regexp** operators of =~, !~ check a pattern on the right-hand side against a string value on the left-hand side.
- The **inclusion** operators are in and not in.

You can use the following binary operators:

- **and, or, xor, nand**
- You can use the following unary operator:
- **!**

Logstash Configuration

- There are lots of permutations and combinations possible with expressions. They can include other expressions or group few expressions using parenthesis (…).
- In the following expression, a conditional check is used to take an action, which in this case is to **drop** events that contain DEBUG or INFO level log information:

```
filter {
#Rest of the processing
if [type] == "linux-syslog" and [messagetype] in ["DEBUG", "INFO"]
{
drop {}
}
}
```

Logstash Configuration

- Multiple expressions can be combined in a single condition.

```
output {
# Send production errors to stdout
if [loglevel] == "ERROR" and [type] == "apache-error" {
stdout { codec => rubydebug }
}
}
```

Logstash Configuration

- Metadata
- From Logstash 1.5 onwards, you can specify metadata with events. It is a neat way to extend and build event fields with field references and sprintf formatting.
- The metadata information is specified by using **@metadata** field.
- A common use for metadata tag is to handle logs of different applications running on the same machine.
- Each application emits its own logs in a separate log file.
- The local Logstash reads all the messages, processes, and forwards ahead to the central Logstash server or Elasticsearch server.
- It can be challenging to ensure that the correct filters and output run on the logs.

Logstash Configuration

- Adding the “tags” field to the input and checking for it in filters and output is a common practice. This requires some discipline in removing the tag in the output.
- More often than not, the tag stays in the output and leads to unexpected results.
- This can now be avoided by intelligently adding metadata to events.
- In fact, a metadata tag can be used to form an independent Logstash pipeline for each application running on the same system without the need of running multiple instances of Logstash.

Logstash Configuration

- The following example shows how to use metadata tags for RabbitMQ logs.
- On reading logs from a RabbitMQ topic and processing them, each type of log is dumped into its own RabbitMQ topic (based on the type field of the event).
- Refer Separate-logs.conf

Logstash Configuration

- Filtering Events
- Before proceeding with log analysis, Logstash has to be configured to accept input from a particular source and in a particular format.
- In order to read, parse, and filter different types of data, Logstash lets you specify different types of inputs, outputs, and filters.
- This is facilitated by a diverse set of plugins. In order to read data from a file, the **file** plugin can be used.

Logstash Configuration

- After configuring the input file, the appropriate filter needs to be applied on the input so that only useful fields are picked and analyzed.
- For this purpose, a **filter** plugin can be used to perform intermediate processing on the input event. This filter can be applied on selective fields based on conditions.
- Since your input file is in a CSV format, it is best to use the **csv** filter. On receiving the input event, the csv filter parses it and stores its individual fields. Besides the comma, it can parse data with other separators also.

Logstash Configuration

- Generally, the csv filter looks like following:
- filter {
- csv {
- columns => #Array of column names.
- separator => #String ; default -","
- }
- }
- Optionally, the attribute columns can be used to specify the name of fields in an input csv file. The default nomenclature would be column1, column2, and so on.
- The separator attribute specifies the character to be used to separate the different columns in the file.

Logstash Configuration

- For your example, let's use the following csv filter:

```
filter {
csv {
columns =>
["day_of_year","date_of_record","max_temp"]
separator => ","
}
}
```

Logstash Configuration

- There is a specific **date filter** in Logstash and it looks as following:

```
filter {
date {
match => # array (optional), default: []
target => # string (optional), default: "@timestamp"
timezone => # string (optional)
}
}
```

Logstash Configuration

- The match attribute is associated with an array in the format [field, formats].
- It is followed by a set of time formats which can be applied to the field.
- If the input events have multiple formats, the following code can be used:

match => ["date_field", "MMM dd YYY HH:mm:ss",

"ISO8601", "MMddYYYY", "MMM d YYY HH:mm:ss"]

Logstash Configuration

- Based on the input event date format, the date filter would be the following:

```
date{
match => ["date_of_record", "yyyy-MM-dd"]
target => "@timestamp"
}
```

Logstash Configuration

- For conversion of fields to a specific data type, the **mutate** filter can be used.
- This filter performs general mutations such as modification of data types, renaming, replacing fields, and removing fields.
- It can also perform other advanced functions like merging two fields, performing uppercase and lowercase conversion, split and strip fields, and so on.
- Generally, a **mutate** filter looks like following:

Logstash Configuration

```
filter {
mutate {
convert => # hash of field and data type (optional)
join => # hash of fields to be joined (optional)
lowercase => # array of fields to be converted (optional)
merge => # hash of fields to be merged (optional)
rename => # hash of original and rename field (optional)
replace => # hash of fields to replaced with (optional)
split => # hash of fields to be split (optional):
```

Logstash Configuration

- strip => # array of fields (optional)
- uppercase => # array of fields (optional)
- }
- }
- The mutate filter in your case looks like the following:
- mutate {
- convert => ["max_temp","integer"]
- }
- The convert functionality is being used to convert max_temp (maximum temperature) to an integer.

Logstash Configuration

- Shipping Events
- After transforming data into a CSV format, configuring Logstash to accept data from a CSV file, and process it based on the specified data type, you are all set to ship the events.
- In your example, Logstash will fetch the data from the CSV file and ship it to **Elasticsearch**, where the different fields can be indexed.
- This will facilitate the visualization of data using the **Kibana** interface.
- The **output** plugin of Logstash can be used to get output in a form acceptable by Elasticsearch.

Logstash Configuration

- output

```
{
elasticsearch {
action => # string (optional), default: "index"
hosts => # array
document_id => # string (optional), default: nil
index => # string (optional), default: "logstash-%{+YYYY.MM.dd}"
path => # string (optional), default: "/”
timeout => # number
}
}
```

Logstash Configuration

- **action**: The action to take on incoming documents. The default action is “**index**” which can be changed to “delete”. For indexing a document, use the “index” value; for deleting a document, use the “delete” value.
- **hosts**: IP address or hostname(s) of the node(s) where Elasticsearch is running. If multiple hosts are specified, requests will be load balanced.
- For example, a single host can be specified as “127.0.0.1” and multiple hosts can be specified as [“127.0.0.1:9200”, “127.0.0.2:9200”].

Logstash Configuration

- **document_id**: Document id of the index; useful to delete or overwrite the existing entries.
- **index**: The index name where incoming events have to be written.
- The default action is to index based on each day and name it as “logstash-%{+YYYY.MM.dd}”. The timestamp value is based on the filter criteria (event capture time or event raising time).
- **path**: HTTP path at which Elasticsearch is accessible.
- **timeout**: The timeout value for network requests and requests send to Elasticsearch.

Logstash Configuration

- logstash –f “D:\logstash configurations/tmax.conf”


```
{
  "took" : 7,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 9,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "tmax-data",
        "_type" : "doc",
        "_id" : "L1vDvGcB1da7Pa9bD8Dw",
        "_score" : 1.0,
        "_source" : {
          "message" : "5,05/01/1986,66\r",
          "path" : "D:/logstash configurations/tmax.csv",
          "host" : "DESKTOP-55AGI0I",
          "day_of_year" : "5",
          "max_temp" : 66,
          "date_of_record" : "05/01/1986",
          "@timestamp" : "1986-01-04T18:30:00.000Z",
          "@version" : "1"
        }
      },
      {
        "_index" : "tmax-data",
        "_type" : "doc",
        "_id" : "LFvDvGcB1da7Pa9bD8Dv",
        "_score" : 1.0,
        "_source" : {
          "message" : "1,01/01/1986,62\r",
          "path" : "D:/logstash configurations/tmax.csv",
          "host" : "DESKTOP-55AGI0I",
          "day_of_year" : "1",
          "max_temp" : 62,
          "date_of_record" : "01/01/1986",
          "@timestamp" : "1985-12-31T18:30:00.000Z",
          "@version" : "1"
        }
```

Logstash Configuration

- To list all indices
- http://localhost:9200/_cat/indices?v
- To search by index
- curl http://localhost:9200/tmax-data/_search?pretty*
- To search by type
- http://localhost:9200/_all/doc/_search?pretty
- To delete by index
- curl -XDELETE “http://localhost:9200/.monitoring-es-6-2018.12.17”

Logstash Configuration Ruby filter

```
filter {
csv {
      separator => ","
#Date,Open,High,Low,Close,Volume (BTC),Volume (Currency),Weighted Price
      columns => ["Date","Open","High","Low","Close","Volume (BTC)", "Volume (Currency)" ,"Weighted Price"]
  }

  ruby {
    code => "
      wanted_fields = ['High', 'Low']
      event.to_hash.keys.each { |k|
        event.remove(k) unless wanted_fields.include? k
      }
    "
  }
}
```

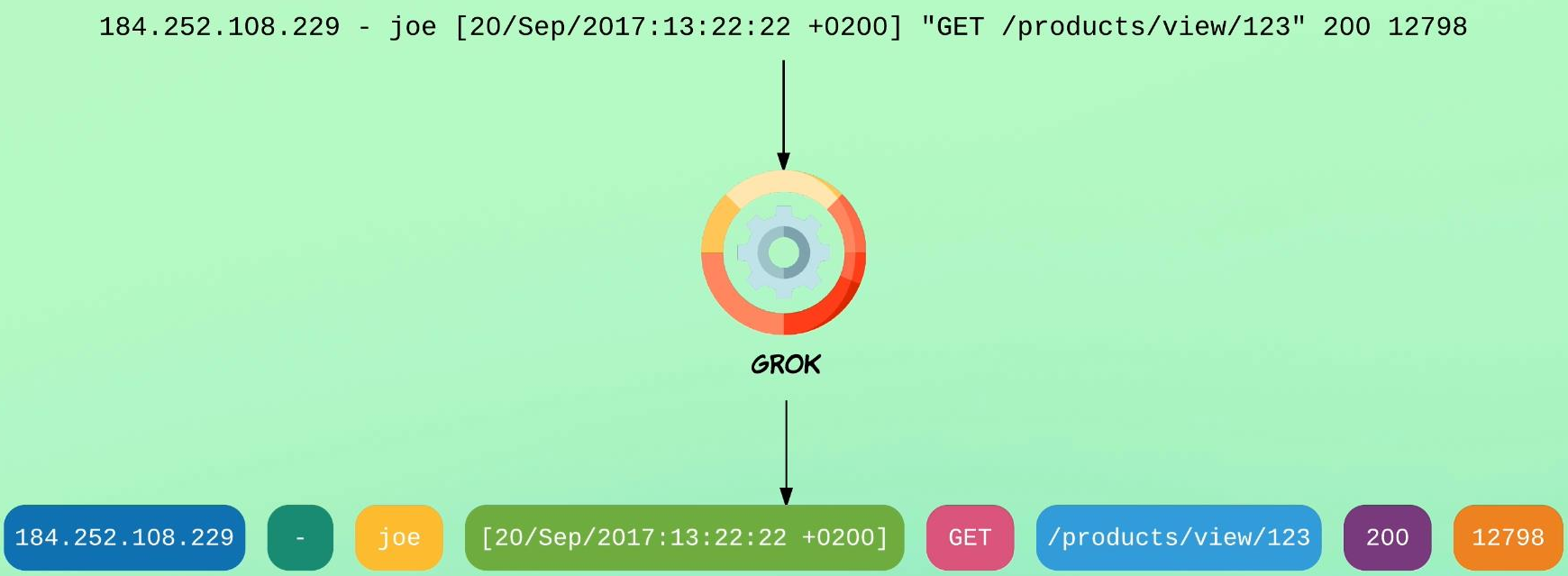
Logstash Configuration Ruby filter


```
{
  "took" : 27,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 1321,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "bitcoinruby-prices",
        "_type" : "doc",
        "_id" : "XCEJwmcBk_dKDbM7Zl8H",
        "_score" : 1.0,
        "_source" : {
          "High" : "973",
          "Low" : "912.02001"
        }
      },
      {
        "_index" : "bitcoinruby-prices",
        "_type" : "doc",
        "_id" : "WCEJwmcBk_dKDbM7ZmAL",
        "_score" : 1.0,
        "_source" : {
          "High" : "973.2",
          "Low" : "915.15"
```

Logstash Configuration grok

- Grok is a great way to parse unstructured log data into something structured and queryable.
- GROK (Graphical Representation of Knowledge)
- This tool is perfect for syslog logs, apache and other webserver logs, mysql logs, and in general, any log format that is generally written for humans and not computer consumption.
- Logstash ships with about 120 patterns by default.
- Grokpatterns [click here]

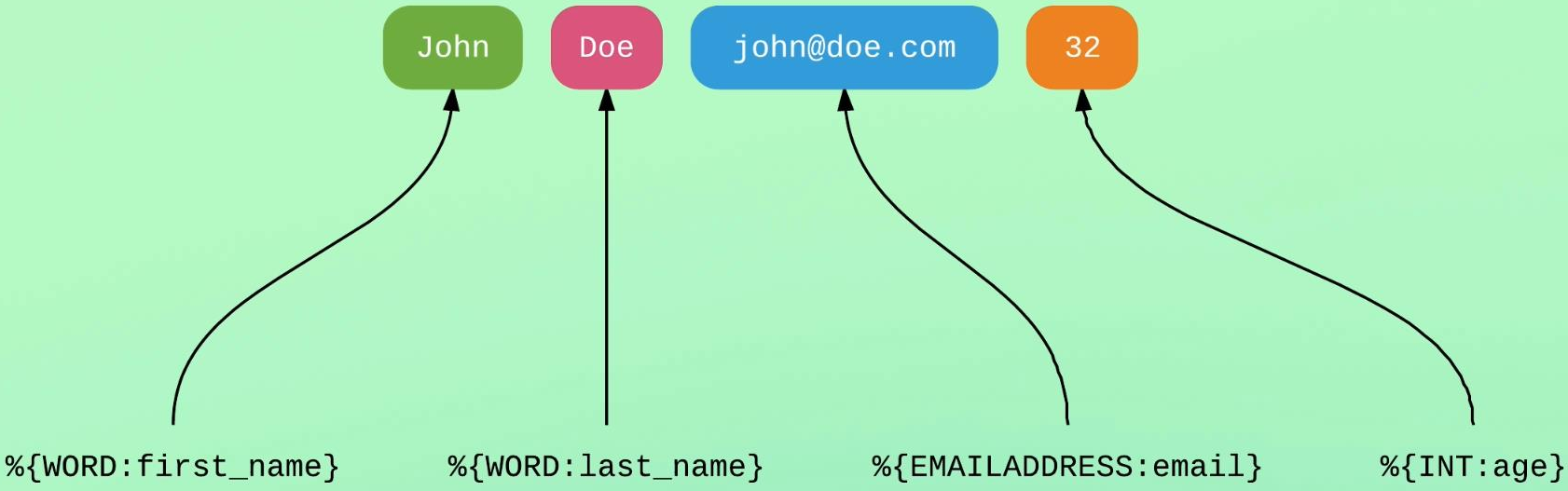
Parsing Requests using grok



Parsing Requests using grok

```
%{SYNTAX:SEMANTIC}
```

Parsing Requests using grok



Parsing file request using grok

- https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

- https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

- Refer ngnixgrok.conf

Date grok(dategrok.conf)

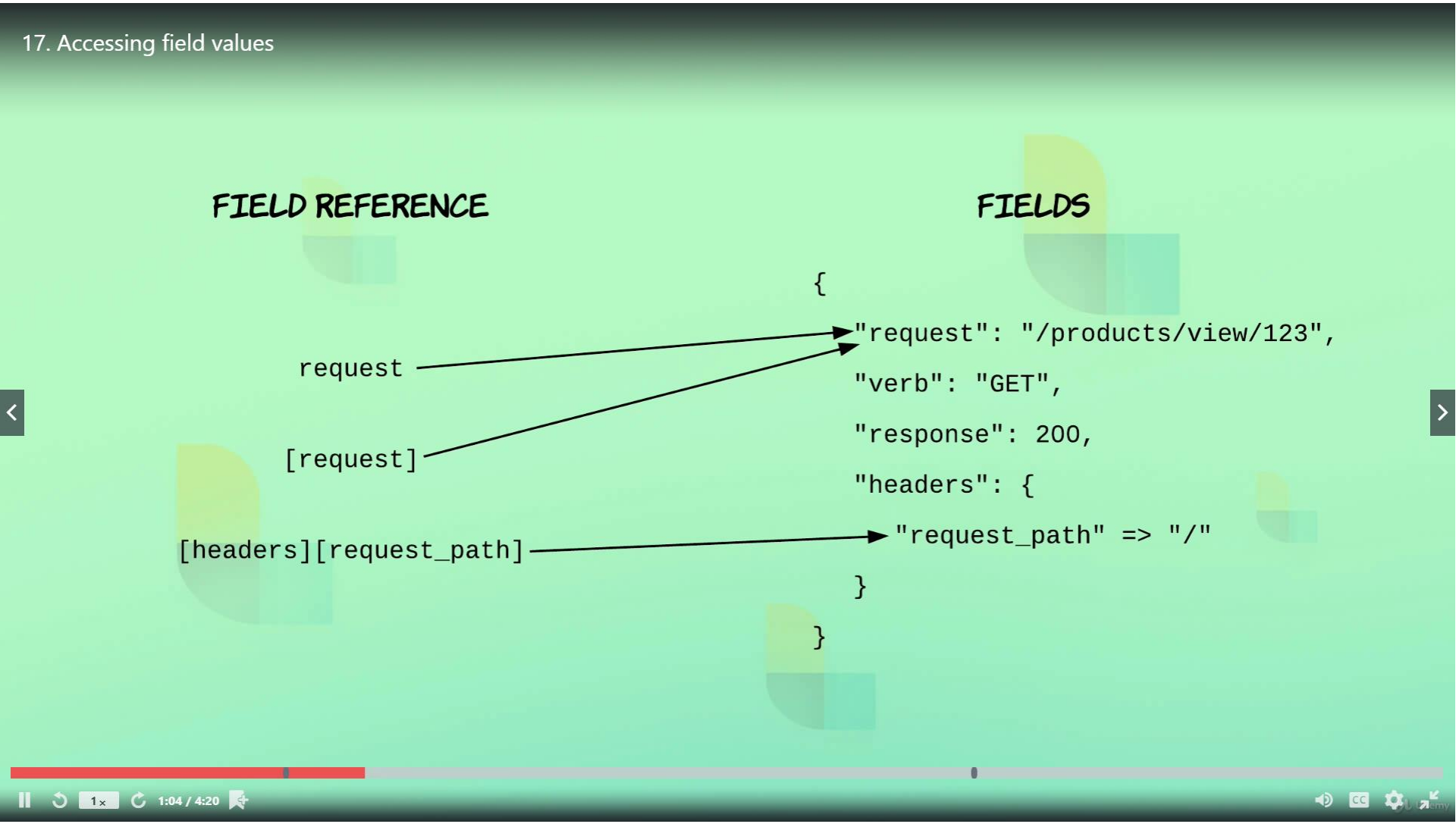
- 127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] "GET /xampp/status.php HTTP/1.1" 200 3891 "http://cadenza/xampp/navi.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0"

Date grok(dategrok.conf)

Administrator: Command Prompt - logstash --verbose -f "D:\logstash configurations\date_grok.conf" --config.reload.automatic

```
          "bytes" => "3891",
      "timestamp" => "11/Dec/2013:00:01:45 -0800",
       "@version" => "1",
     "@timestamp" => 2013-12-11T08:01:45.000Z,
       "clientip" => "127.0.0.1",
        "message" => "127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] \"GET /xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"\r",
        "request" => "/xampp/status.php",
       "referrer" => "\"http://cadenza/xampp/navi.php\"",
           "verb" => "GET",
           "host" => "DESKTOP-55AGI0I"
}
127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] "GET /xampp/status.php HTTP/1.1" 200 3891 "http://cadenza/xampp/navi.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0"
{
          "ident" => "-",
           "auth" => "-",
          "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"",
       "response" => "200",
    "httpversion" => "1.1",
          "bytes" => "3891",
      "timestamp" => "11/Dec/2013:00:01:45 -0800",
       "@version" => "1",
     "@timestamp" => 2013-12-11T08:01:45.000Z,
       "clientip" => "127.0.0.1",
        "message" => "127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] \"GET /xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"\r",
        "request" => "/xampp/status.php",
       "referrer" => "\"http://cadenza/xampp/navi.php\"",
           "verb" => "GET",
           "host" => "DESKTOP-55AGI0I"
}
```

Access Field Value



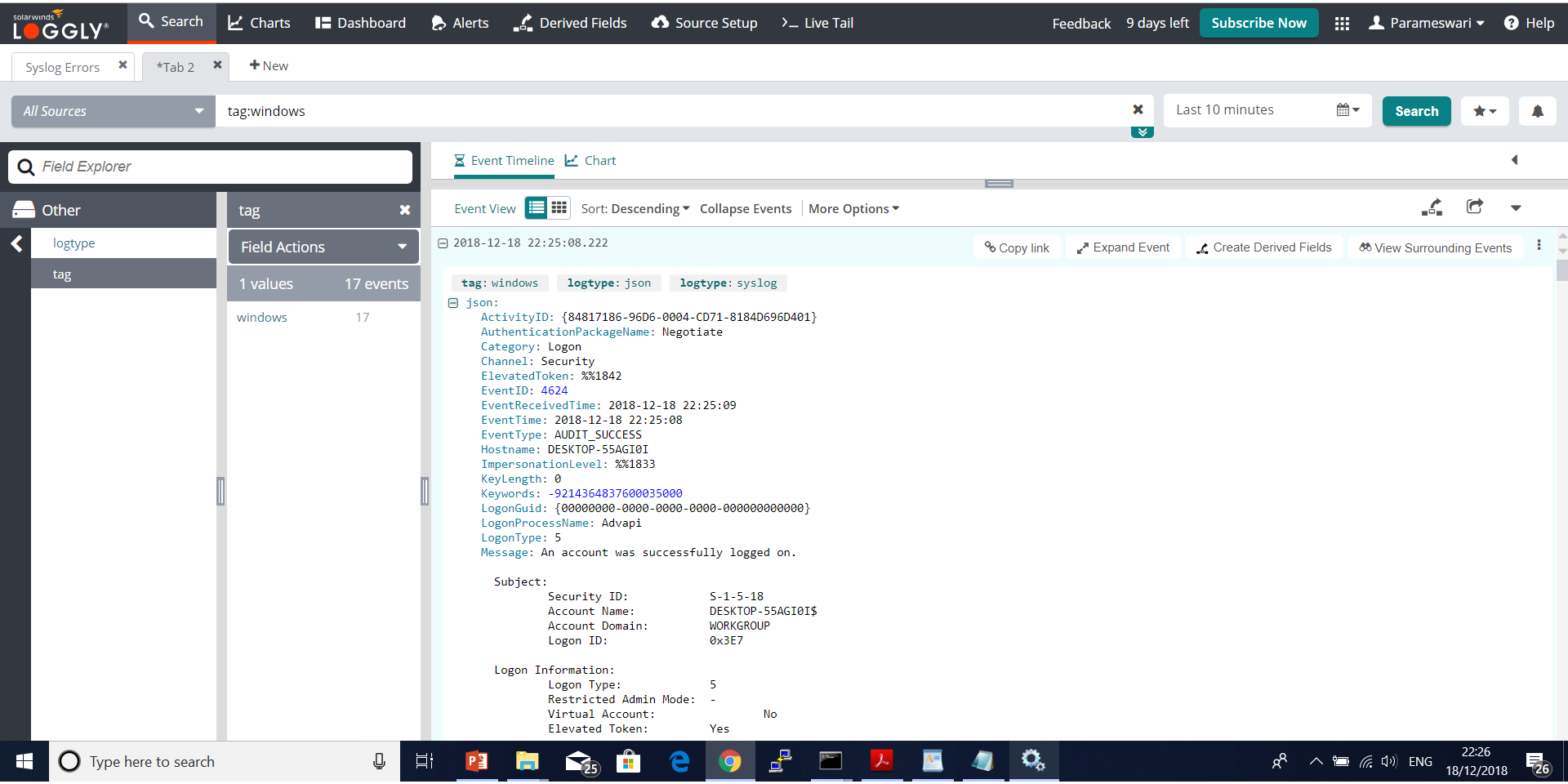
Syslog with grok

- Download nxlog
- https://nxlog.co/products/nxlog-community-edition/download
- Install nxlog msi
- **Copy the Configuration**
- **Replace** the above variables:
- CUSTOMER_TOKEN: Replace with your own customer token
- ROOT and ROOT_STRING: If you are in 32-bit Windows, uncomment the top root path on lines 8 and 9 to use the 32-bit program files folder then comment the two below.

Syslog with grok

- Verify
- Verify it shows up in Loggly by doing a search for the windows tag over the past hour.
- tag:windows

Syslog with grok



Geoip logstash

Administrator: RabbitMQ Command Prompt (sbin dir) - logstash -f "D:\logstash configurations\geotest.conf"

```
{
    "@timestamp" => 2018-12-18T19:17:29.840Z,
          "geoip" => {
            "region_name" => "Victoria",
                     "ip" => "1.1.1.1",
         "continent_code" => "OC",
              "city_name" => "Research",
          "country_code3" => "AU",
            "postal_code" => "3095",
              "longitude" => 145.15,
               "location" => {
              "lat" => -37.7333,
              "lon" => 145.15
         },
          "country_code2" => "AU",
               "timezone" => "Australia/Melbourne",
           "country_name" => "Australia",
              "latitude" => -37.7333,
           "region_code" => "VIC"
    },
          "host" => "DESKTOP-55AGI0I",
      "@version" => "1",
```

Extending Logstash

- Plugin Management
- Logstash has an extensive collection of plugins (**inputs, filters, outputs, codecs**).
- These plugins are developed by Elastic search but also get significant contributions from the community.
- The biggest forte of Logstash is the ease of availability of plugins and the flexibility of adding new ones to provide more features.
- In fact, about **200** plugins are available for you to choose from and work with.

Extending Logstash

- Plugins in Logstash are not part of the core package.
- Ruby is used to develop Logstash plugins.
- The Ruby programming language comes with a package manager called RubyGems.
- This package manger specifies a common format for distributing programs and libraries built in Ruby.
- It manages the installation and distribution of gems.
- RubyGems is used to make Logstash plugins available as separate self-contained packages.
- RubyGems provides the support required for the release of plugin updates independently from Logstash releases.

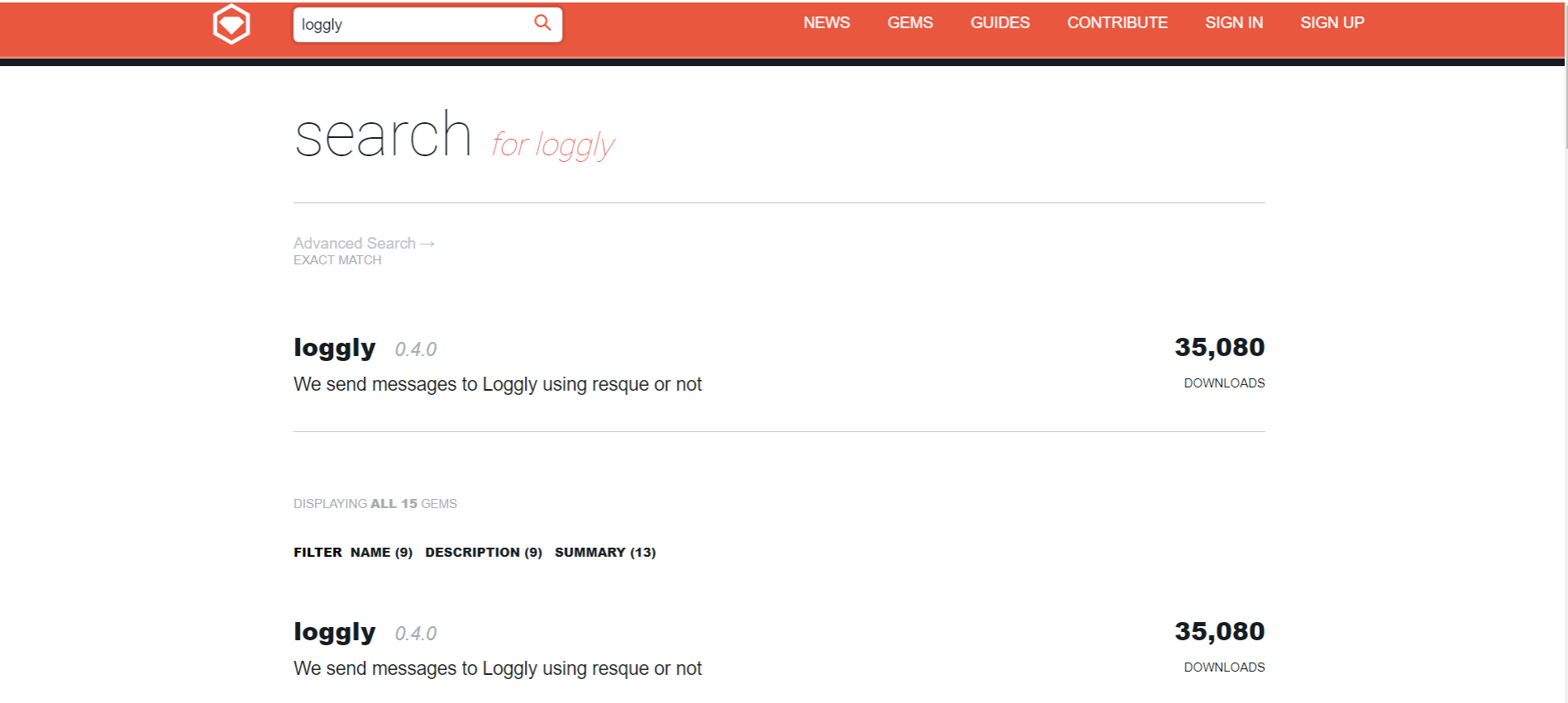
Extending Logstash

- The key advantages of moving plugins away from the Logstash core are as follows:
- The Logstash release can be independent of plugin updates.
- Developers can release new features and bug fixes for plugins without being tied to the Logstash release plan.
- It offers easy-to-describe external dependencies.
- It means a leaner core Logstash distribution package.

Extending Logstash

- Logstash plugins, both core and community ones, can be downloaded from
- https://rubygems.org/.
- The GitHub repository is another place where you can access all of the Logstash plugins:
- https://github.com/logstash-plugins.

Extending Logstash



Plugin Installation

- Let's say you want to install the logstash-output-mongodb plugin.
- You can issue the following command from the Logstash installation folder:
- **logstash-plugin install logstash-output-mongodb**
- This command will install the **logstash-output-mongodb** plugin to the Logstash installation.
- If you want to install a particular version, you can specify the –version parameter.
- logstash-plugin list

Plugin Installation

- Updating a Plugin
- Whenever a new version of a plugin is released, you may want to upgrade the existing installation of that plugin.
- This can be done by using the following command:
- logstash-plugin update logstash-output-mongodb
- This command will update the logstash-output-mongodb plugin to its latest version.

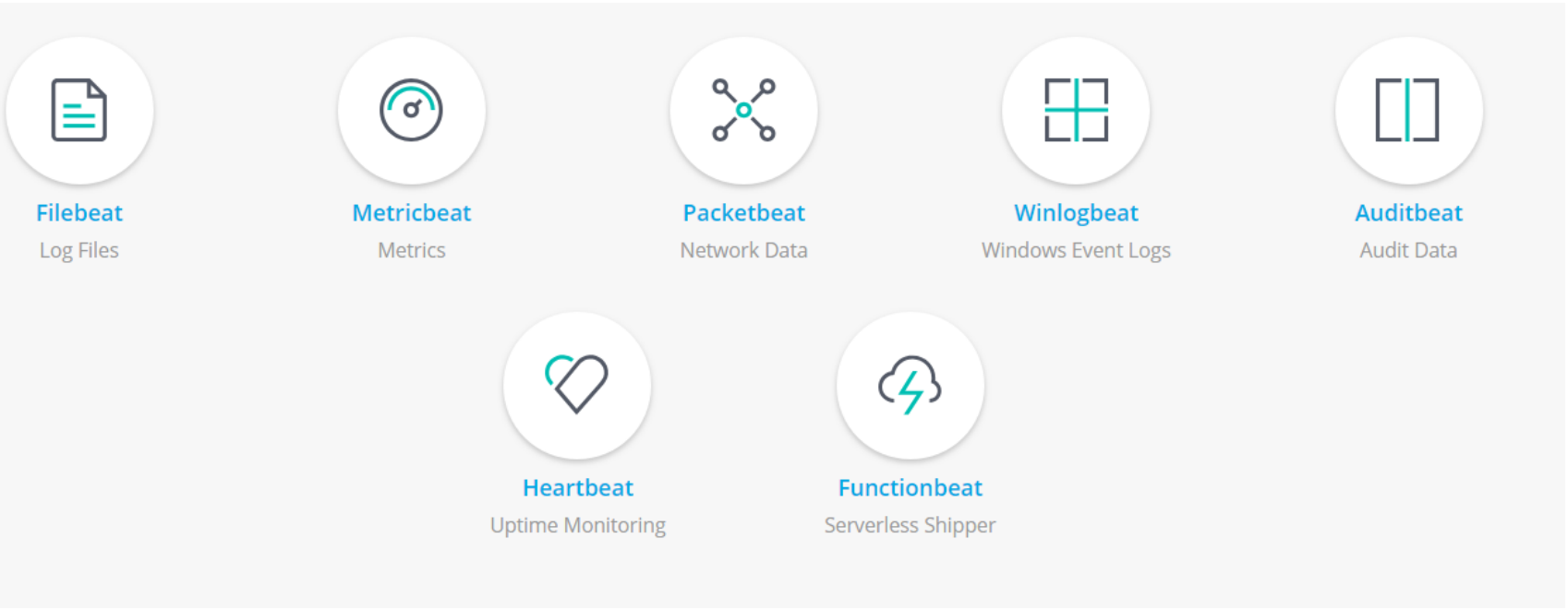
Plugin Installation

- Uninstallation
- If you no longer wish to use a particular plugin, you can uninstall it in the following manner:
- logstash-plugin uninstall logstash-output-mongodb
- This command will **uninstall** the logstash-event plugin from the Logstash installation..

Lightweight Data Shippers

- Beats is the platform for single-purpose data shippers.
- They send data from hundreds or thousands of machines and systems to Logstash or Elasticsearch.

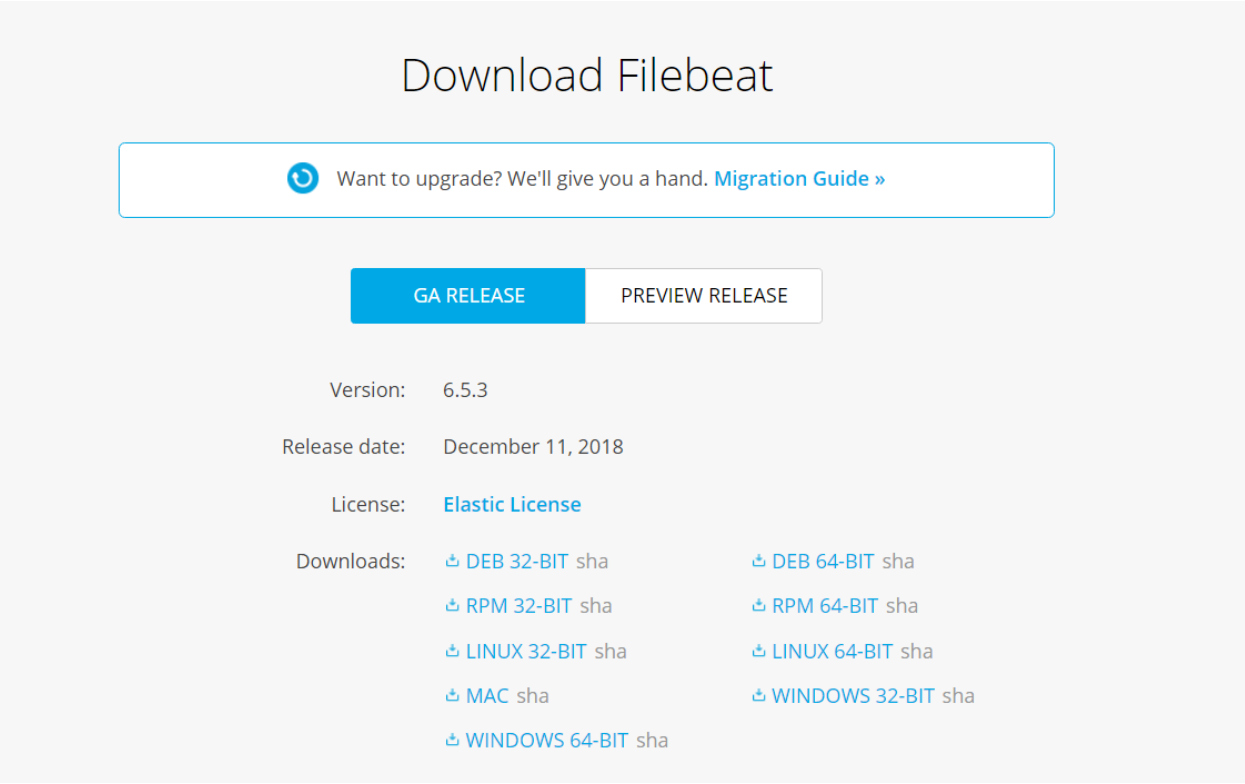
Lightweight Data Shippers



Lightweight Data Shippers

- Filebeat: collects and ships log files.
- Metricbeat: collects metrics from your systems and services.
- Packetbeat: collects and analyzes network data.
- Winlogbeat: collects Windows event logs.
- Auditbeat: collects Linux audit framework data and monitors file integrity.
- Heartbeat: monitors services for their availability with active probing.

Install FileBeat



Install FileBeat

Installation Steps

1. Download and unzip Filebeat

   Filebeat can also be installed from our package repositories using apt or yum. See *Repositories* in the Guide.

2. Edit the filebeat.yml configuration file

3. Start the daemon by running `sudo ./filebeat -e -c filebeat.yml`

4. Dive into the getting started guide and video.

File Beat Configuration

- Open the Filebeat configuration file:
- filebeat/filebeat.yml
- Filebeat supports numerous outputs, but you'll usually only send events directly to Elasticsearch or to Logstash for additional processing.
- Here we'll use Logstash to perform additional processing on the data collected by Filebeat.
- Filebeat will not need to send any data directly to Elasticsearch, so let's disable that output.
- To do so, find the output.elasticsearch section and comment out the following lines by preceding them with a #:

File Beat Configuration

- Then, configure the output.logstash section.
- Uncomment the lines output.logstash: and hosts: ["localhost:5044"] by removing the #.
- This will configure Filebeat to connect to Logstash on your Elastic Stack server at port 5044, the port for which we specified a Logstash input earlier:

File Beat Configuration

- The functionality of Filebeat can be extended with Filebeat modules.
- Here we will use the system module, which collects and parses logs created by the system logging service of common Linux distributions.
- Let's enable it:
- filebeat modules enable system
- You can see a list of enabled and disabled modules by running:
- filebeat modules list

File Beat Configuration

```
Module system is already enabled

D:\filebeat-6.3.0-windows-x86_64>filebeat modules list
Enabled:
system

Disabled:
apache2
auditd
icinga
iis
kafka
logstash
mongodb
mysql
nginx
osquery
postgresql
redis
traefik

D:\filebeat-6.3.0-windows-x86_64>
```

File Beat Configuration

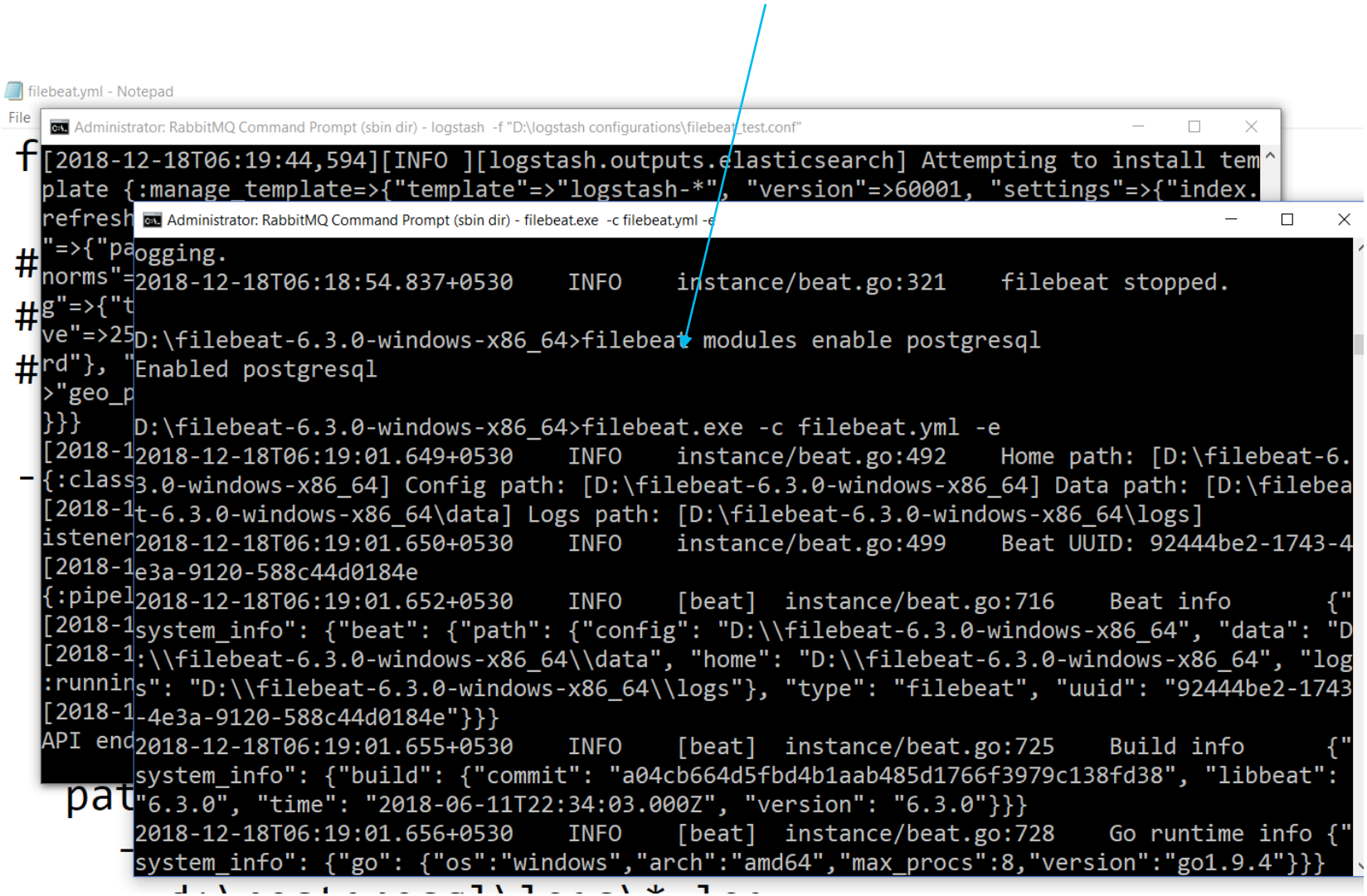
```
traefik

D:\filebeat-6.3.0-windows-x86_64>filebeat modules enable mysql
Enabled mysql

D:\filebeat-6.3.0-windows-x86_64>filebeat modules list
Enabled:
mysql
system

Disabled:
apache2
auditd
icinga
iis
kafka
logstash
mongodb
nginx
osquery
postgresql
redis
```

File Beat Configuration



Running Filebeat

- filebeat.exe –modules mongodb -c filebeat.yml –e
- filebeat.exe -c filebeat.yml -e -d "*"

Running Filebeat

C:\Windows\System32\cmd.e

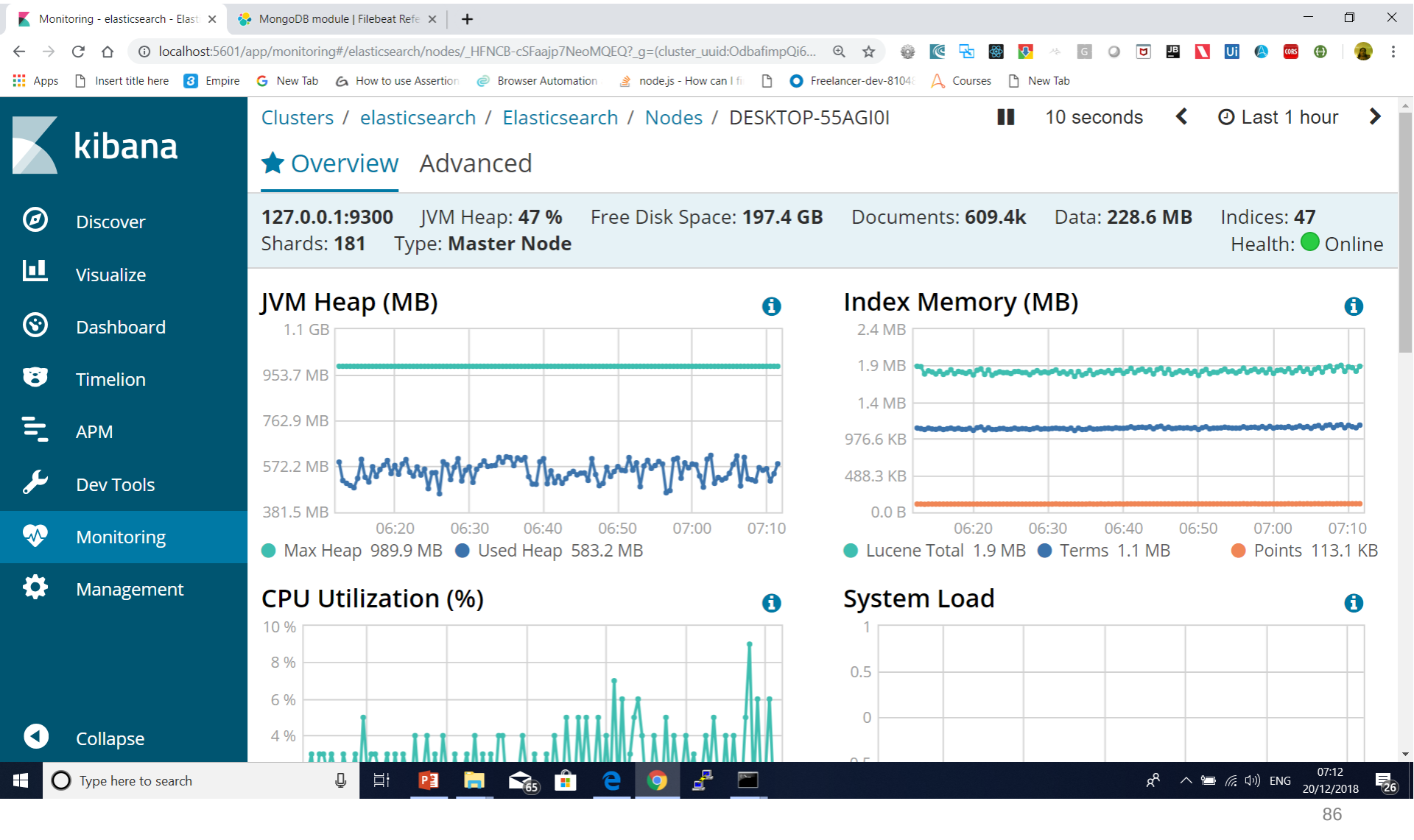
```
      "192.168.1.35",
      "fe80::a600:cd58:e118:16ab",
      "169.254.47.59",
      "fe80::e98e:5ee5:5ca1:83a7",
      "172.22.0.1"
    ],
    "name": "DESKTOP-55AGI0I",
    "mac": [
      "58:8a:5a:02:6b:0a",
      "f8:34:41:ac:d4:70",
      "fa:34:41:ac:d4:6f",
      "00:ff:28:1b:32:b7",
      "f8:34:41:ac:d4:6f",
      "f8:34:41:ac:d4:73",
      "00:15:5d:49:e9:80"
    ]
  },
  "agent": {
    "id": "58aa5e0b-ef1b-4704-9105-8479072bd79c",
    "name": "DESKTOP-55AGI0I",
    "type": "filebeat",
    "version": "7.15.0",
    "hostname": "DESKTOP-55AGI0I",
    "ephemeral_id": "55de2138-dd3a-4ba7-9a44-1b398b9bd6d4"
  },
  "log": {
    "file": {
      "path": "E:\\software\\A08\\file\\apache-activemq-5.14.0-bin\\apache-activemq-5.14.0\\data\\activemq.log"
    },
    "offset": 128741
  }
}
2023-06-17T12:10:37.342+0530    DEBUG   [processors]    processing/processors.go:203    Publish event: {
  "@timestamp": "2023-06-17T06:40:37.341Z",
  "@metadata": {
    "beat": "filebeat",
    "type": "_doc",
    "version": "7.15.0"
  },
  "agent": {
```

Running Filebeat and logstash filebeat_test.conf

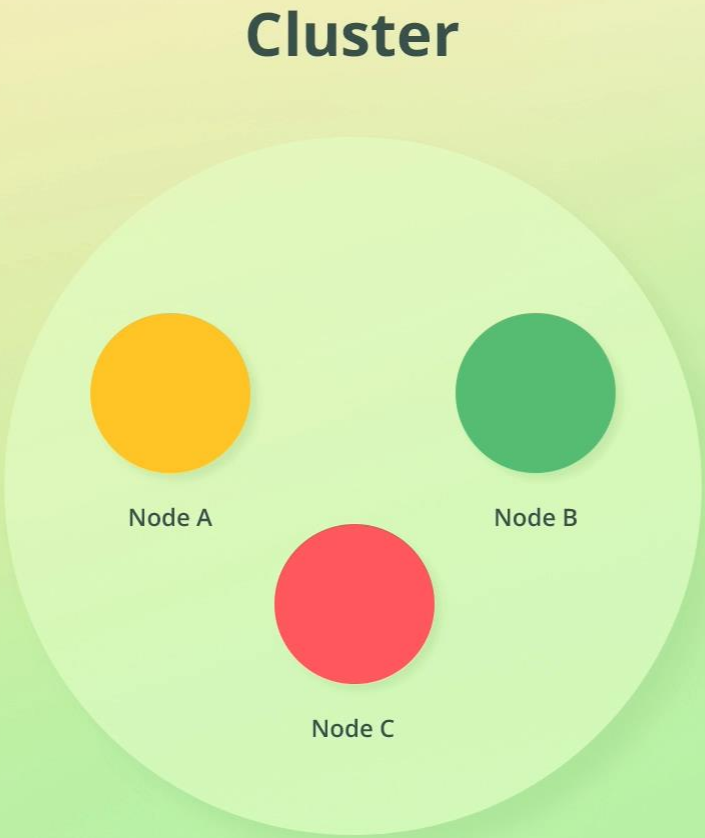
C:\Windows\System32\cmd.e

```
              "id" => "5c06b0e1-6f67-4761-b106-2718ea98589a",
             "mac" => [
        [0] "58:8a:5a:02:6b:0a",
        [1] "f8:34:41:ac:d4:70",
        [2] "fa:34:41:ac:d4:6f",
        [3] "00:ff:28:1b:32:b7",
        [4] "f8:34:41:ac:d4:6f",
        [5] "f8:34:41:ac:d4:73",
        [6] "00:15:5d:49:e9:80"
    ]
},
     "agent" => {
    "ephemeral_id" => "55de2138-dd3a-4ba7-9a44-1b398b9bd6d4",
        "hostname" => "DESKTOP-55AGI0I",
            "name" => "DESKTOP-55AGI0I",
         "version" => "7.15.0",
            "type" => "filebeat",
              "id" => "58aa5e0b-ef1b-4704-9105-8479072bd79c"
},
"@timestamp" => 2023-06-17T06:40:37.138Z,
       "log" => {
      "file" => {
        "path" => "E:\\software\\A08\\file\\apache-activemq-5.14.0-bin\\apache-activemq-5.14.0\\data\\activemq.log"
    },
    "offset" => 115137
},
   "message" => "2018-03-12 05:53:01,197 | INFO  | Connector openwire started | org.apache.activemq.broker.TransportConnector | WrapperSimpleAppMain",
     "input" => {
    "type" => "filestream"
},
  "@version" => "1",
       "ecs" => {
    "version" => "1.11.0"
},
    "fields" => {
    "review" => 1,
     "level" => "debug"
},
      "tags" => [
    [0] "beats_input_codec_plain_applied"
```

Metric Beat


Monitoring - elasticsearch - Elast
MongoDB module | Filebeat Refe
localhost:5601/app/monitoring#/elasticsearch/nodes/_HFNCB-cSFaajp7NeoMQEQ?_g=(cluster_uuid:OdbafimpQi6...
Apps
Insert title here
Empire
New Tab
How to use Assertion
Browser Automation
node.js - How can I fi
Freelancer-dev-8104
Courses
New Tab
kibana
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
Clusters / elasticsearch / Elasticsearch / Nodes / DESKTOP-55AGI0I
10 seconds
Last 1 hour
Overview
Advanced
127.0.0.1:9300 JVM Heap: 47 % Free Disk Space: 197.4 GB Documents: 609.4k Data: 228.6 MB Indices: 47
Shards: 181 Type: Master Node
Health: Online
JVM Heap (MB)
1.1 GB
953.7 MB
762.9 MB
572.2 MB
381.5 MB
06:20
06:30
06:40
06:50
07:00
07:10
Max Heap 989.9 MB
Used Heap 583.2 MB
Index Memory (MB)
2.4 MB
1.9 MB
1.4 MB
976.6 KB
488.3 KB
0.0 B
06:20
06:30
06:40
06:50
07:00
07:10
Lucene Total 1.9 MB
Terms 1.1 MB
Points 113.1 KB
CPU Utilization (%)
10 %
8 %
6 %
4 %
System Load
1
0.5
0
Type here to search
ENG
07:12
20/12/2018

Elastic Cluster



Document

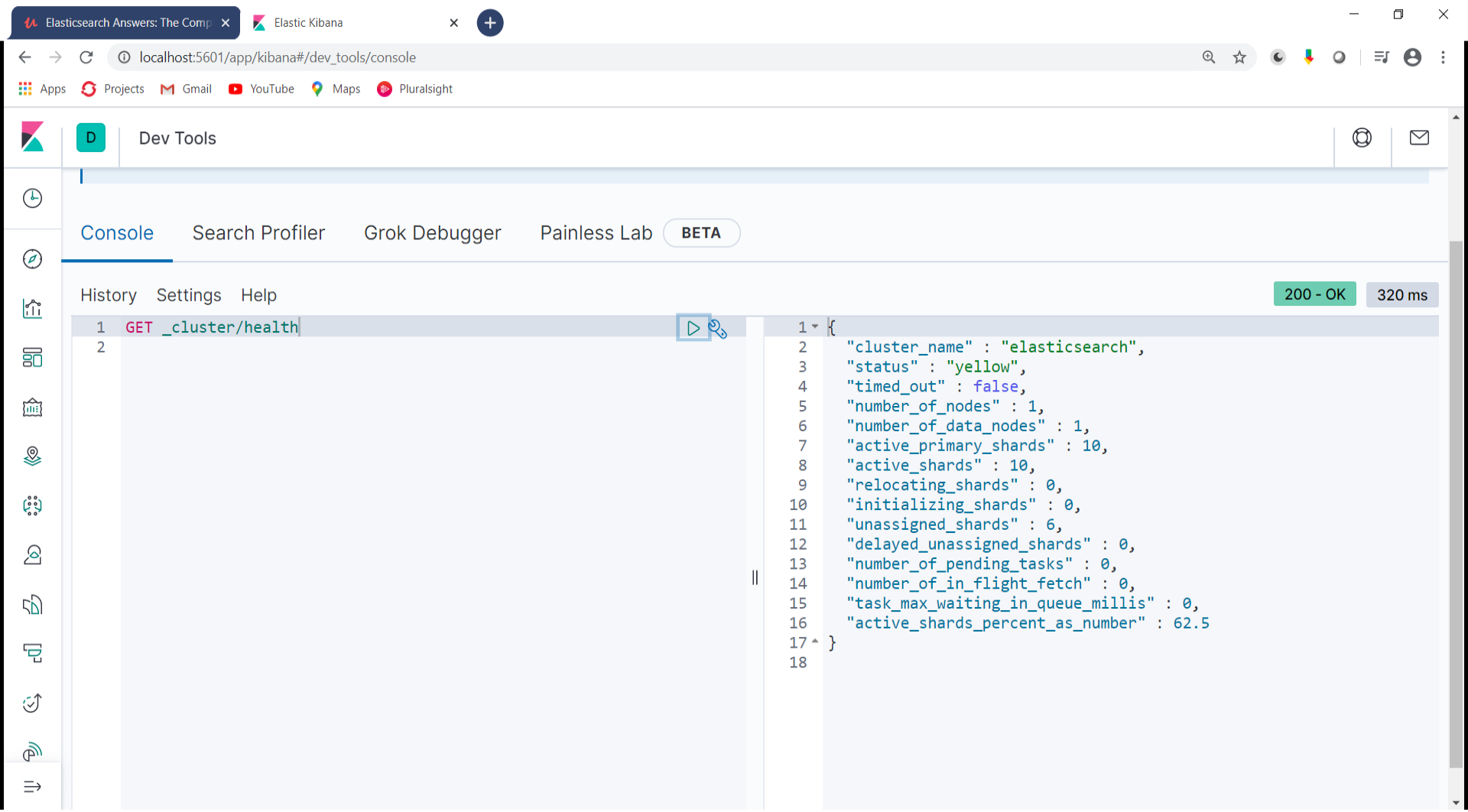
```
{
  "name": "Bo Andersen",
  "country": "Denmark"
}
```

is stored as

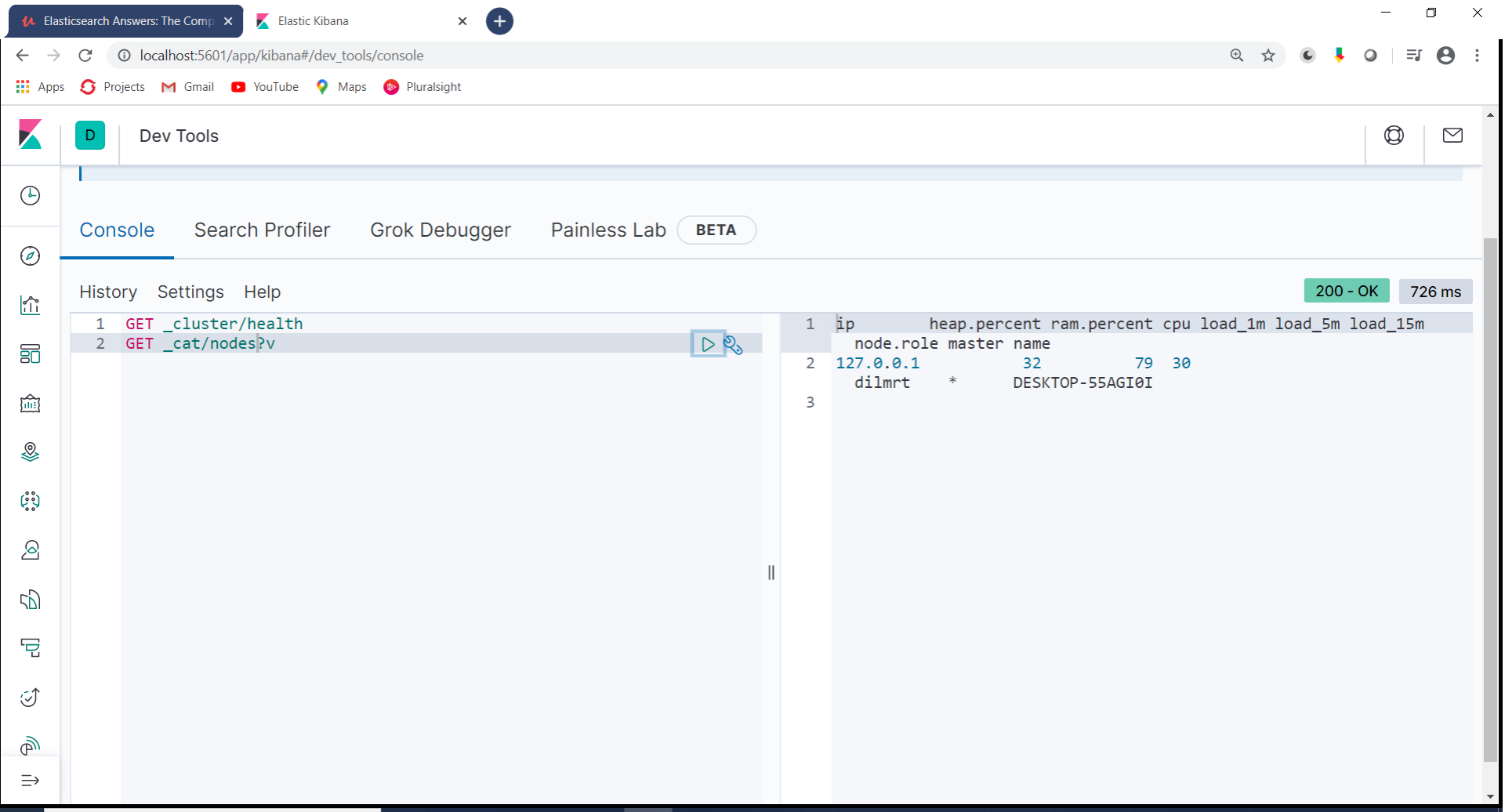
```
{
  "_index": "people",
  "_type": "_doc",
  "_id": "123",
  "_version": 1,
  "_seq_no": 0,
  "_primary_term": 1,
  "_source": {
    "name": "Bo Andersen",
    "country": "Denmark"
  }
}
```

Document and Indexes

GET _cluster/health



GET _cat/nodes?v



Elasticsearch Split Brain Problem

- Make sure nodes can communicate with each other quickly enough and that you won’t have a split brain.
- Split brain happens when two parts of the cluster that can’t communicate and think the other part dropped out.
- Suppose you have 2 nodes — Node 1 and Node 2 and you have just one index deployed.
- Node 1 stores the primary shard and Node 2 stores the replica shard. Node 1 gets elected as a master during cluster start-up.

Elasticsearch Split Brain Problem

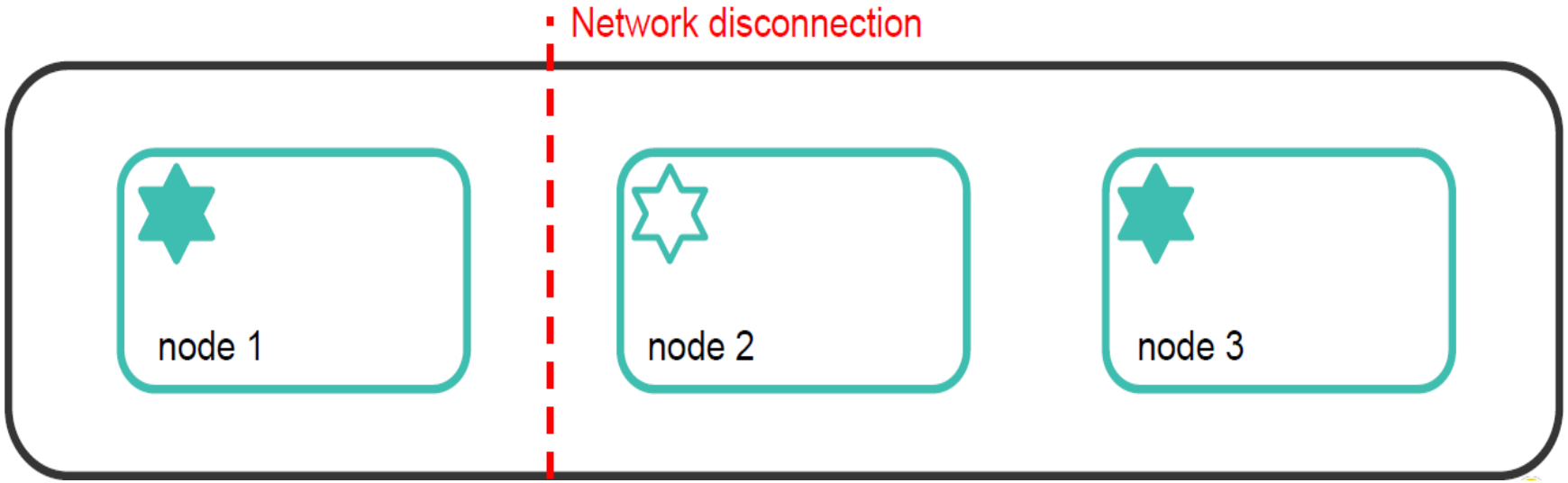
- Then suppose there is communication failure between the two nodes.
- Now, each of the nodes are in dark about the status of the other node and hence, they believe that the other has failed.
- Node 1 being the master will do nothing because it thinks it is up while the slave is down, so no issues.
- But Node 2 thinks that master has gone down and so is the primary shard, so it will automatically elect itself as master and promote the replica shard to a primary shard..

Elasticsearch Split Brain Problem

- Now, the cluster gets into a confused zone and can result in an inconsistent state.
- Indexing requests that will hit Node 1 will index data in its copy of the primary shard, while the requests that go to Node 2 will fill the second copy of the shard.
- This can result in situations like when searching for data: depending on the node the search request hits, results will differ.

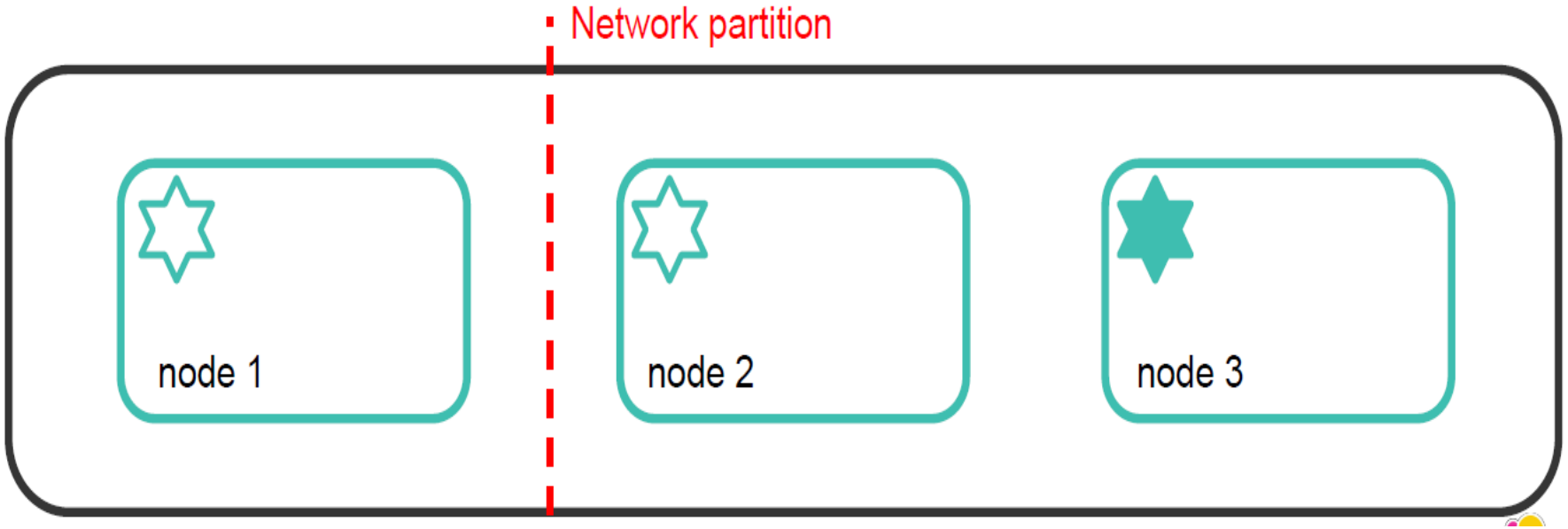
Split Brain

- Cluster with 3 master eligible nodes
- Concern if network becomes partitioned
- The cluster would inadvertently elect two masters, which is referred to “split brain”

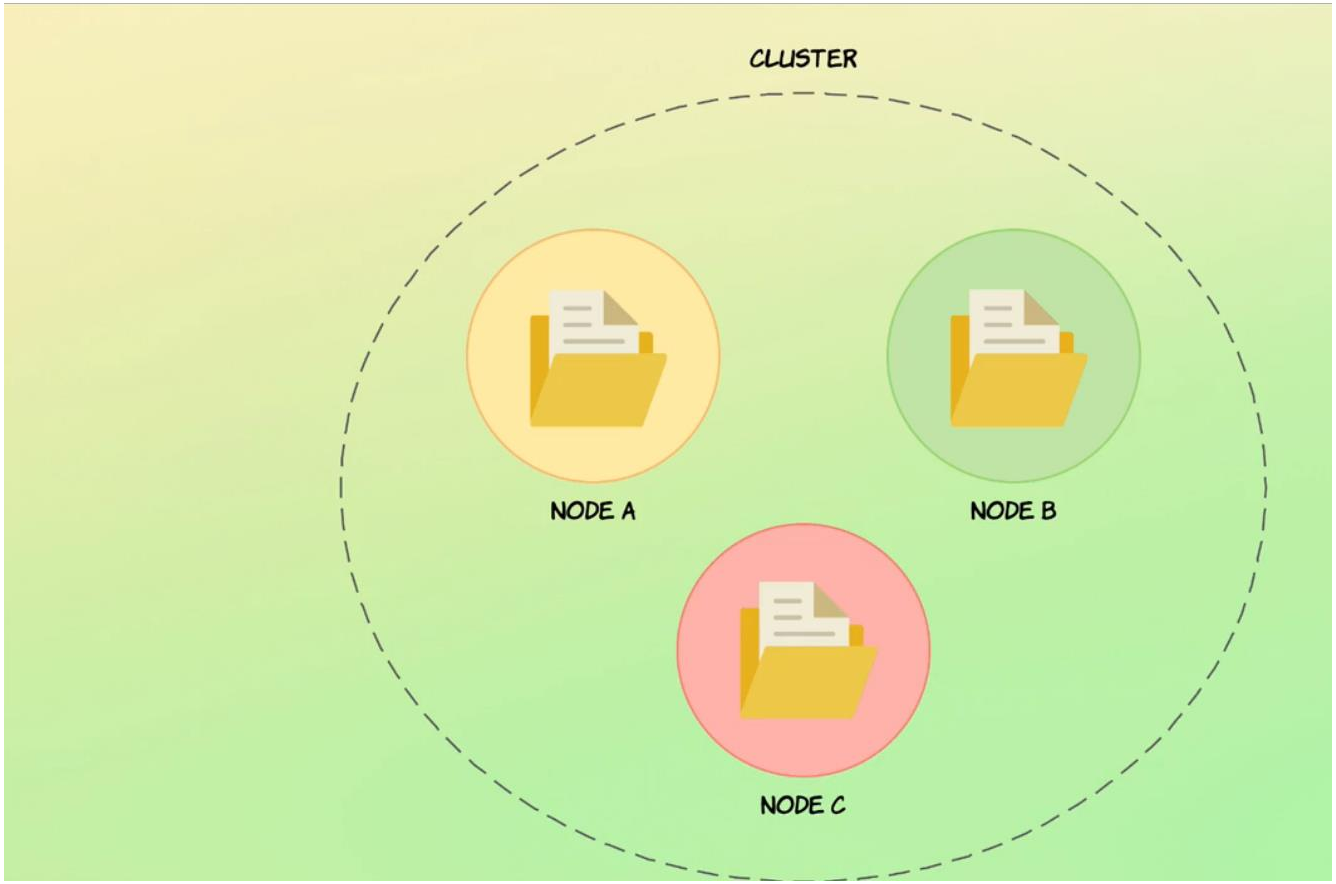


Avoiding Split Brain

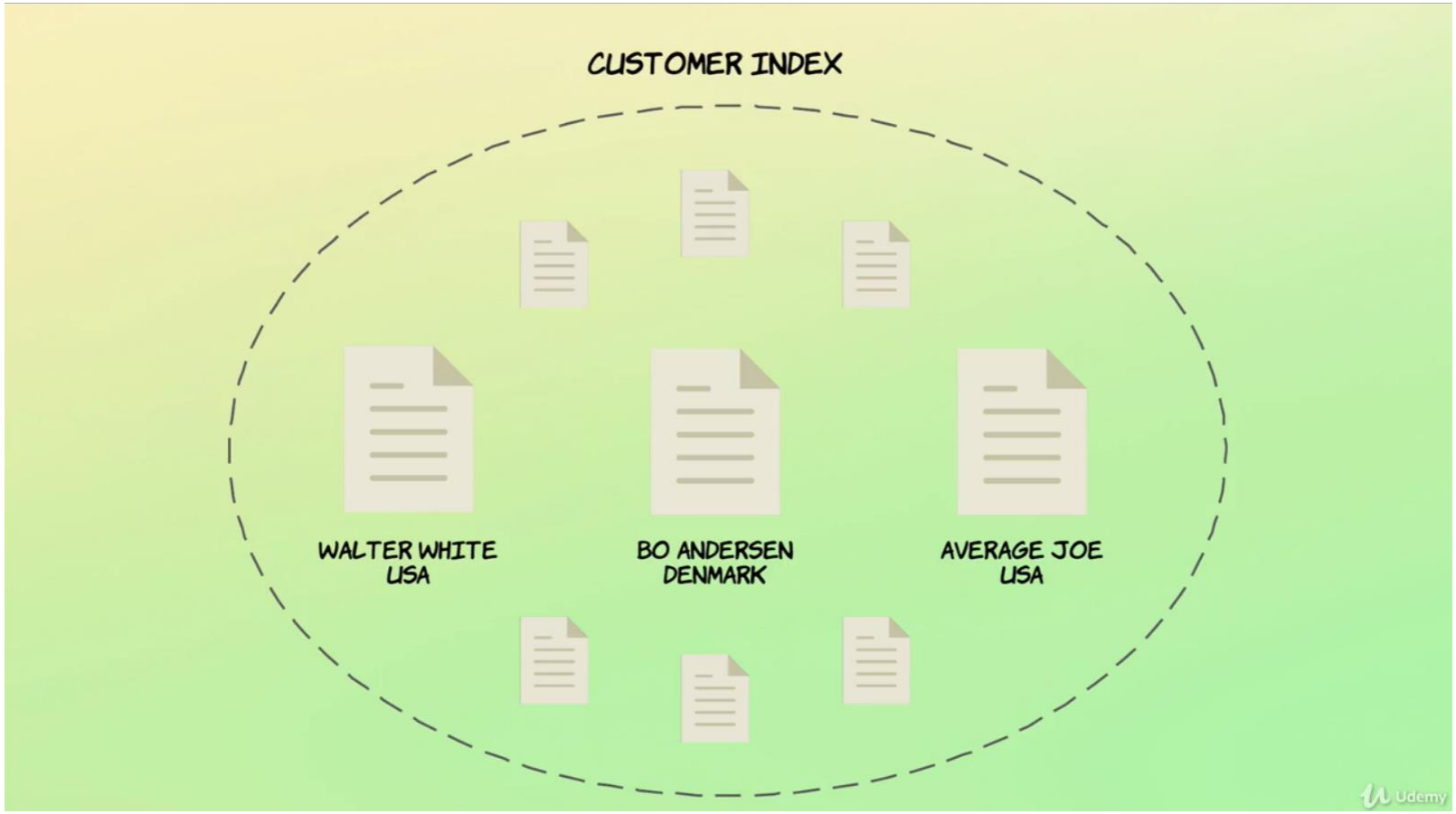
- A master eligible node needs at least minimum_master_nodes votes to win an election
  - Setting it to a quorum prevents the split brain scenario
- Recommendation for production clusters is to have 3 dedicated master eligible nodes
  - with the setting `minimum_master_nodes = 2`



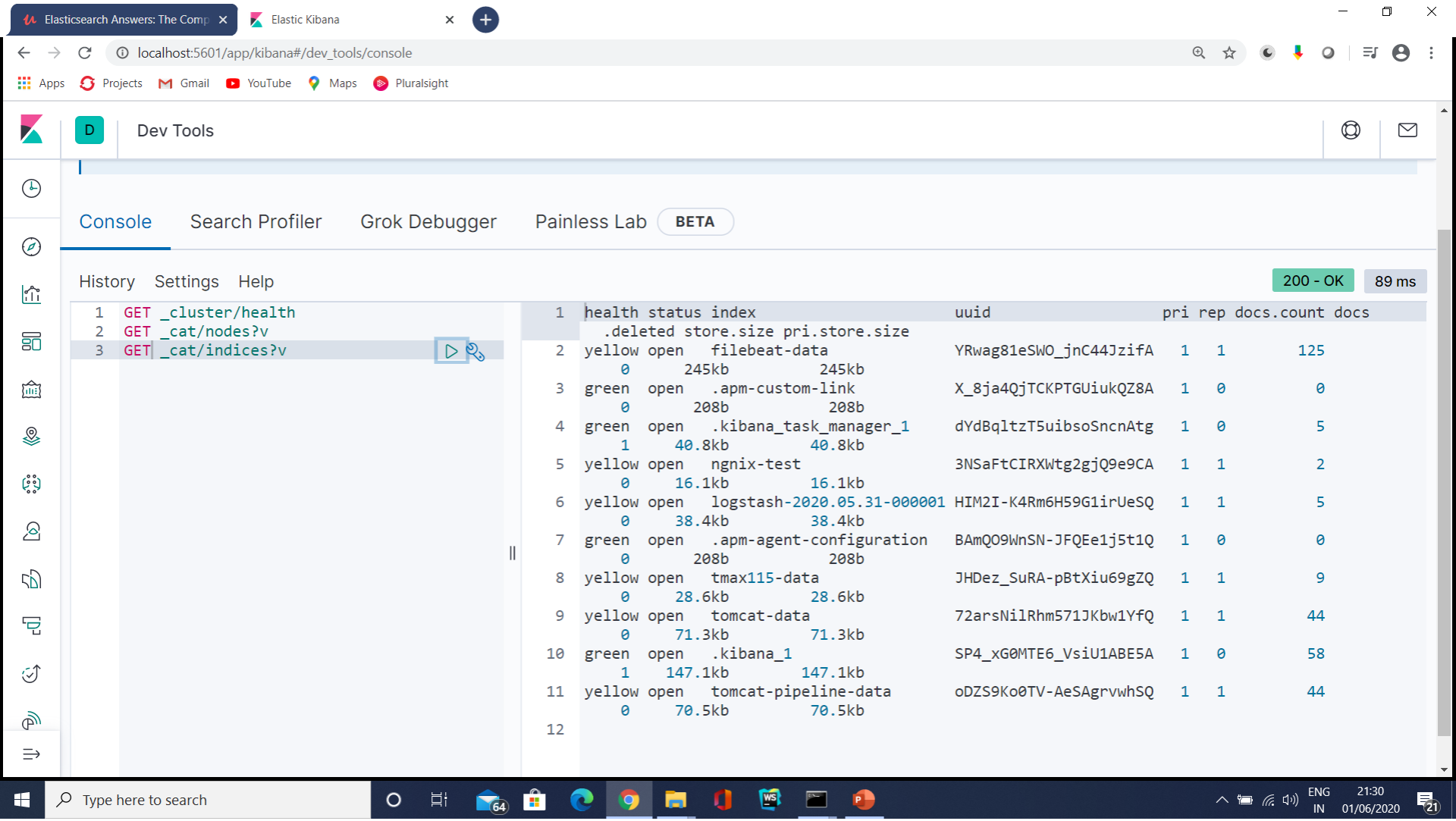
Nodes and Clusters



Nodes and Clusters



GET _cat/indices?v



Nodes and Clusters

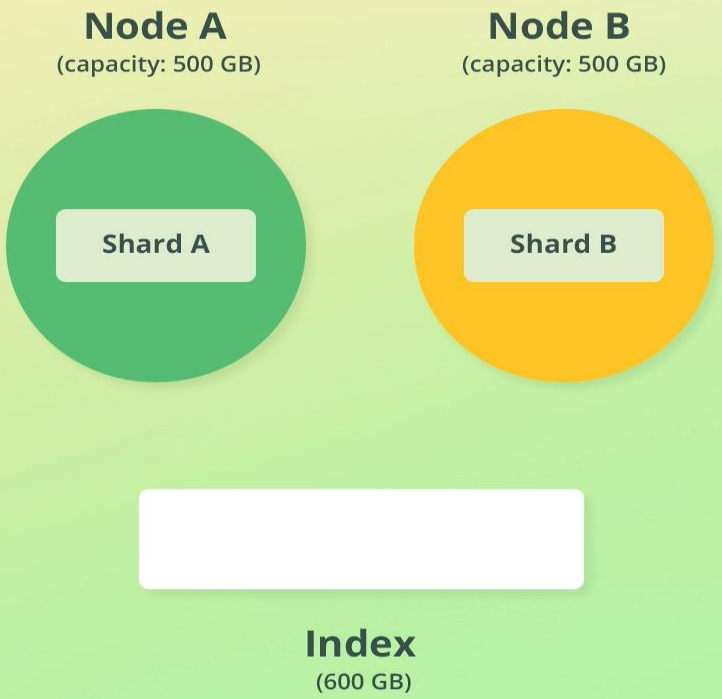
- What is an index?
- An index is stored in a set of shards, which are themselves Lucene indices.
- This already gives you a glimpse of the limits of using a new index all the time: Lucene indices have a small yet fixed overhead in terms of disk space, memory usage and file descriptors used.
- For that reason, a single large index is more efficient than several small indices: the fixed cost of the Lucene index is better amortized across many documents.

What is shard

- Sharding is a way to divide indices into smaller pieces
- Each piece is referred to as a *shard*
- Sharding is done at the index level
- The main purpose is to horizontally scale the data volume

What is shard

Index needs 600 GB space. Divide that to two different shards



What is shard

- A shard is an independent index... *kind of*
- Each shard is an Apache Lucene index
- An Elasticsearch index consists of one or more Lucene indices
- A shard has no predefined size; it grows as documents are added to it
- A shard may store up to about two billion documents

Purpose of Sharding

- Mainly to be able to store more documents
- To easier fit large indices onto nodes
- Improved performance
    - Parallelization of queries increases the throughput of an index

Configuring number of shards

- An index contains a single shard by default
- Indices in Elasticsearch < 7.0.0 were created with five shards
  - This often led to *over-sharding*
- Increase the number of shards with the Split API
- Reduce the number of shards with the Shrink API

How many shards are optimal

- There is no formula that will yield a number for us 😕
- There are many factors involved, so it *depends*
- Factors include the # of nodes and their capacity, the # of indices and their sizes, the # of queries, etc.
- Anticipate millions of documents? Consider adding a couple of shards
- Need to store some thousand documents? Stick to the default settings

How many shards are optimal

- There is no formula that will yield a number for us 😕
- There are many factors involved, so it *depends*
- Factors include the # of nodes and their capacity, the # of indices and their sizes, the # of queries, etc.
- Anticipate millions of documents? Consider adding a couple of shards
- Need to store some thousand documents? Stick to the default settings

Nodes and Clusters

- **How to search data.**
- While each shard is searched independently, Elasticsearch needs to merge results from all the searched shards.
- For instance, if we search across 10 indices that have 5 shards each, the node that coordinates the execution of a search request will need to merge 5x10=50 shard results.
- If there are too many shard results to merge and/or if you ran a heavy request that produces large shard responses (which can easily happen with aggregations), the task of merging all these shard results can **become very resource-intensive, both in terms of CPU and memory**.

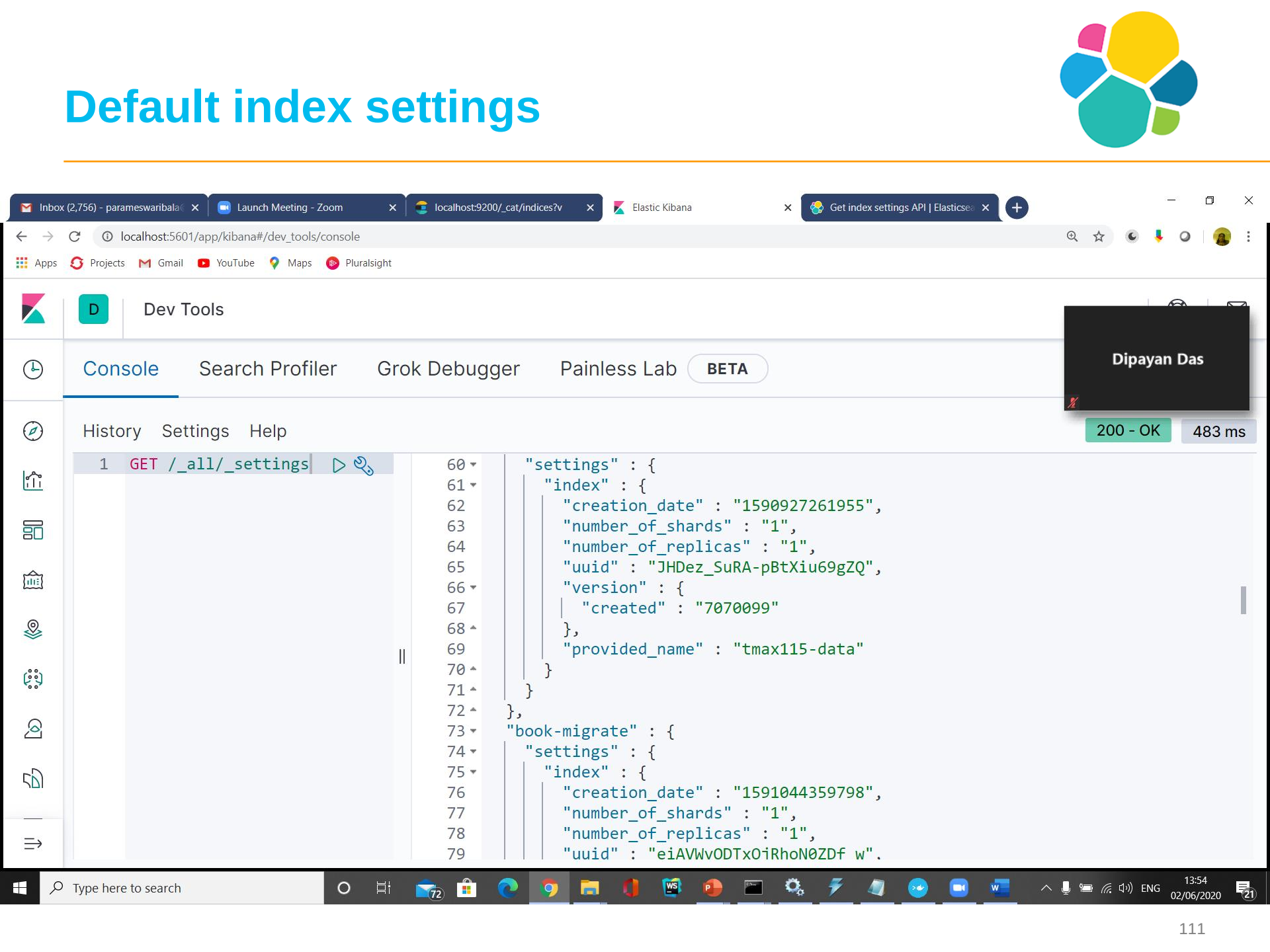
Shards

- The shard is the actual physical area where documents are stored.
- The index is just a logical namespace that references one or more shards.
- Applications need not be bothered with the details of shards, and they can perform all operations using the index.

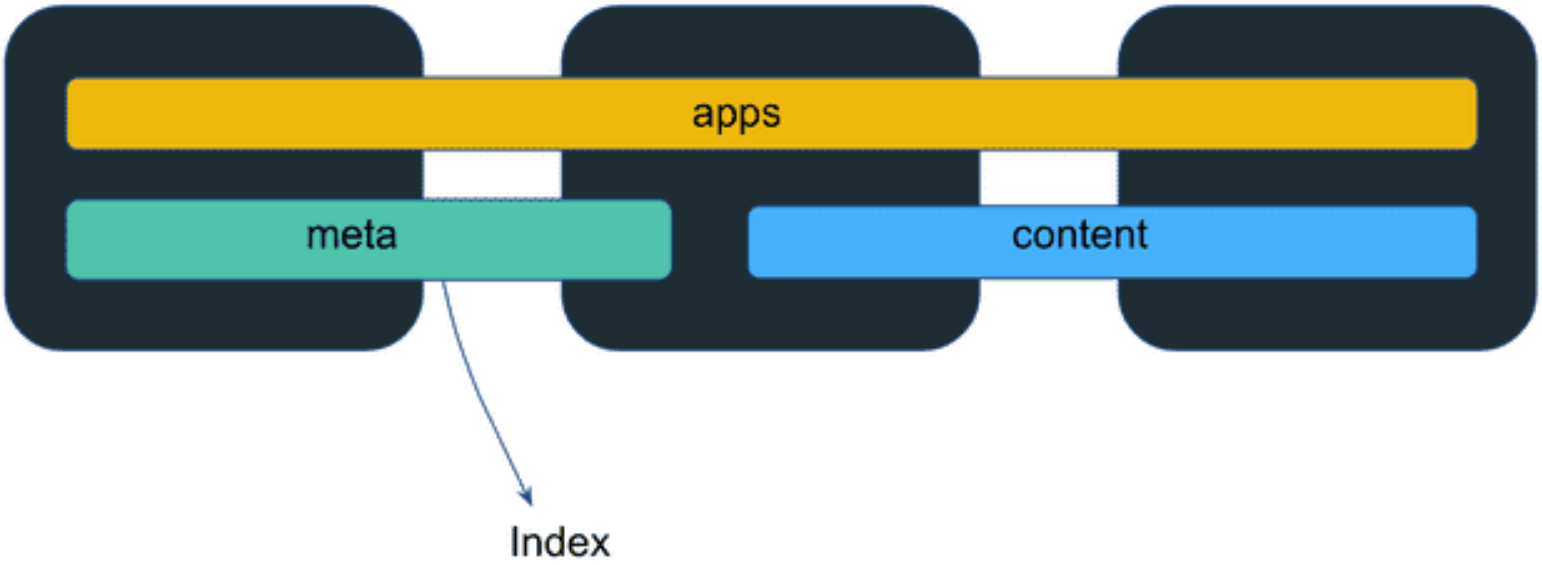
Shards(Primary and Replica)

- In order to maintain the availability of data in an Elasticsearch cluster, all documents are stored on one **primary** shard and multiple **replica** shards.
- When a document is indexed, it is first stored in its primary shard and then on corresponding replica shards.
- The default number of primary shards is **5** and it can be configured as per your needs.
- Replica shards generally reside on a node different from the primary shard.
- Replica shards load balance requests to take care of a high load and play a key role in case of failover.

Default index settings

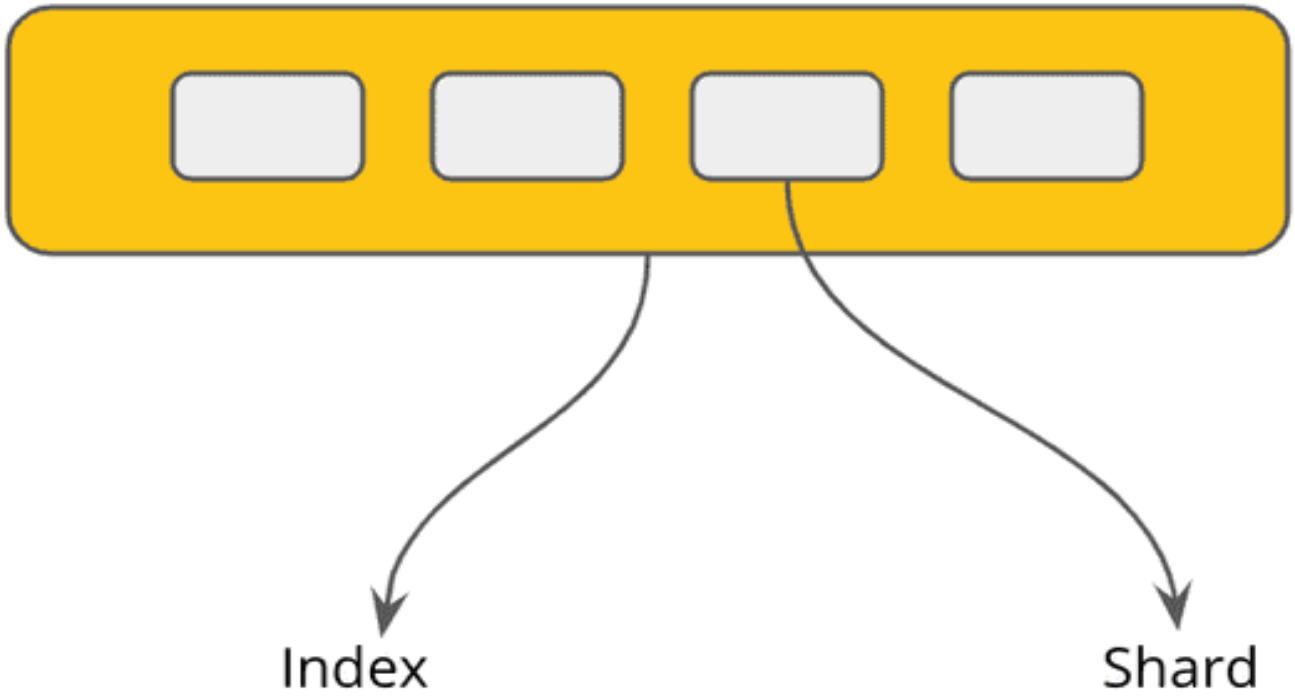


Index



Elasticsearch is a distributed Search engine. The data is stored and spread across various nodes. Indexes are the ones that hold that data logically (physical data is in shards).

Index



Index

RDBMS with SQL

```
select * from emp-index
```

Elasticsearch with QueryDSL

Index name

```
GET emp-index/_search
{
    "query": {
        "match_all": {}
    }
}
```

Index Naming Rules

- Lowercase only
- Cannot include \, /, ``, ?, ", <, >, |, (space character), ,, #
- Indices prior to 7.0 could contain a colon (:), but that has been deprecated and will not be supported in 7.0+
- Cannot start with ``, _, +
- Cannot be . or ..
- Cannot be longer than 255 bytes (note it is bytes, so multi-byte characters will count towards the 255 limit faster)

Index Naming Strategy

- For time-series data such as logs, metrics, traces., Elastic's Beats or Logstash, write data into Elasticsearch with default index names like filebeat-0001 or logstash-%{+yyyy.MM.dd}.
- It automatically increments the index name after a specific data limit.
- These defaults also help in managing data with other features like Index/Snapshot Lifecycle Management Policies.
- If one has a strong opinion about creating indices with naming strategies, go ahead to create your own, but think about the scaling strategy.
- Keeping with defaults helps one can correlating the data (ECS) easily and leverage all the inbuilt features.

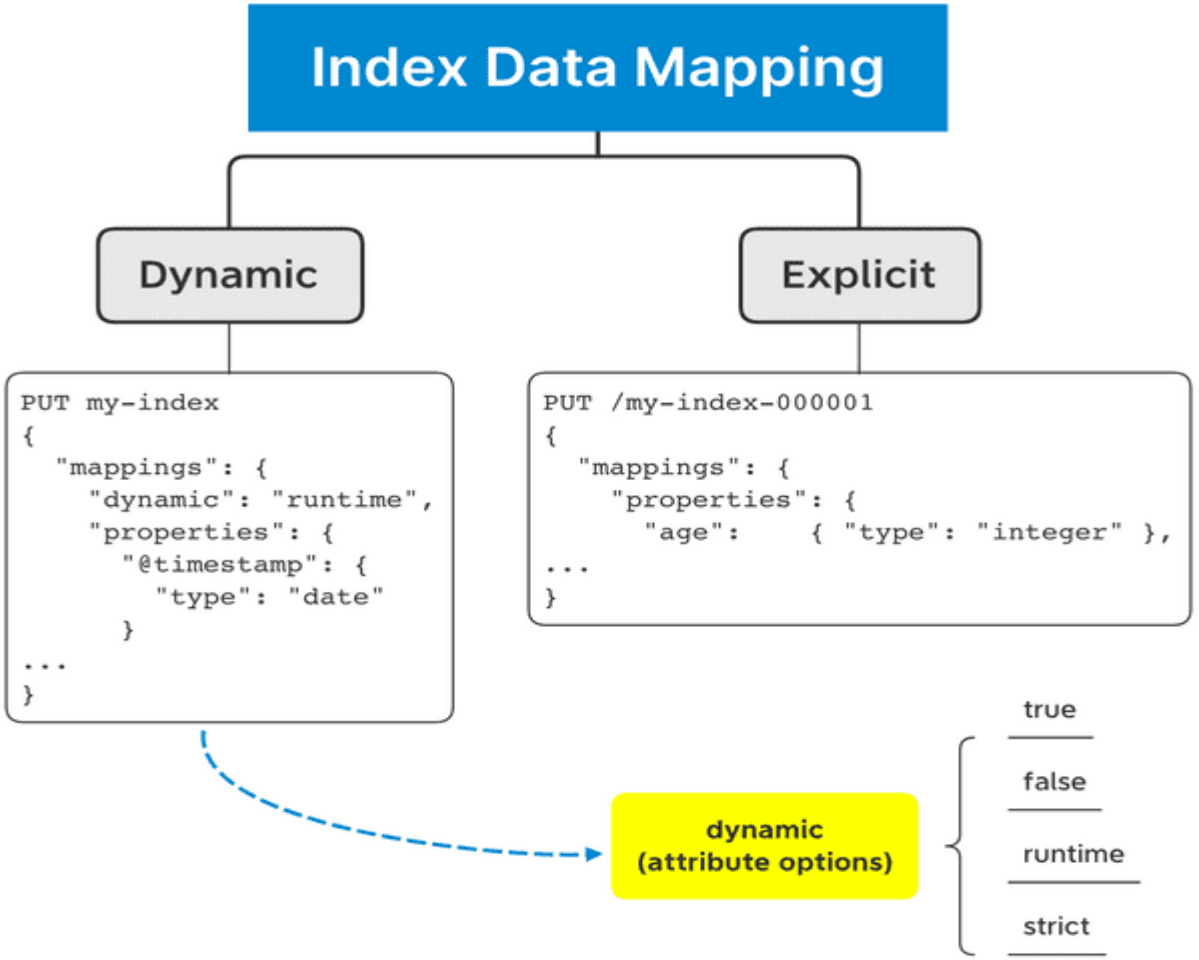
Categories of Data Stored in indexes

- Indices store different data categories, although, remember, each data has a different volume, tenure.
  - Time series (Data indexed in the order of time)
  - Logs
  - Metrics
  - Traces
  - Security Events
  - Content
  - PDFs, Spreadsheets, PPT's etc
  - Data from RDBMS

Index Data Mapping

- Elasticsearch prepares the data for search/query operations while ingesting the data.
- It is also called "Schema on Write."
- It is like creating the schema dynamically/explicitly while defining inserting the record in a SQL Table.
- One can also do "Schema on Read," which means preparing or fetching the data while querying with Runtime fields.

Index Data Mapping



Dynamic

- It happens on the fly as and when we ingest the data. It can be set to false, true, runtime, strict.
- It means one can enable or disable the dynamic mapping even make it strict to follow a specific set of schema, otherwise reject the client's documents for ingestion.

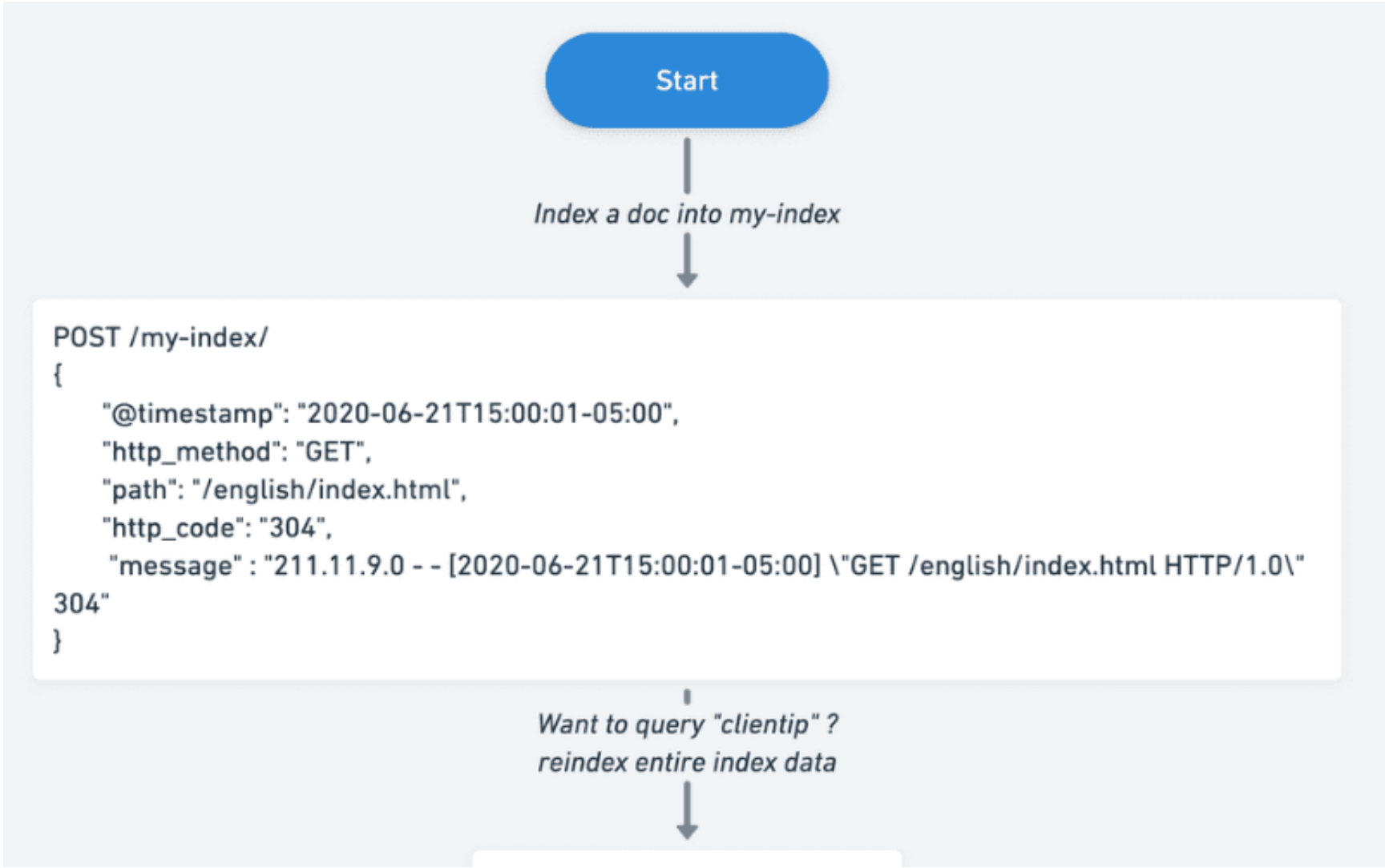
Explicit

- Can be set like an RDBMS Table while creating the index.
- Once defined, it can only be changed based on the mapping parameters.
- If needed to change, data needs to be reindexed.
- Method 1: One could use an alias for a field name and swap to new fields, deprecating the documents' old field.
- Method 2: Create another index with the desired field data type + existing fields in the document, field names, then reindex the data.
- Elasticsearch attempts to index all the fields client sends, but sometimes if we don't want specific fields to be indexed, set enabled field to false and those fields will not be prepared for querying (can also be called not-indexed, unindexed).

Runtime Fields:

- Elasticsearch uses the "Schema on Write" methodology, which enables faster querying, better visualizations.
- However, if unindexed data in the _source needs to be queried later, you need to reindex to bring that data into a field.

Runtime Fields:



Runtime Fields:

Runtime Fields:

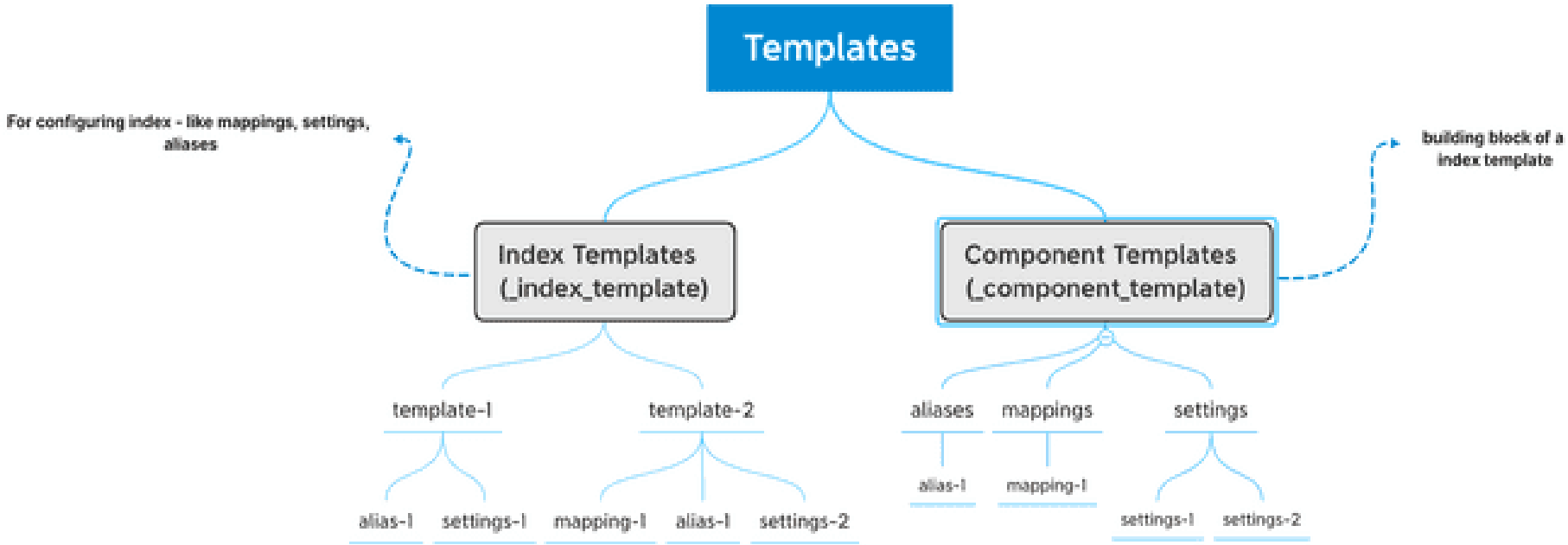
```
GET my-index/_search
{
  "size": 1,
  "query": {
    "match": {
      "clientip": "211.11.9.0"
    }
  },
  "fields" : ["*"]
}
```

Templates

- There are two types of Templates in Elasticsearch: Index Templates, Component Templates.

- Index templates are settings that get applied when an index with a specific pattern gets created, for example, logs-* or app-*.

Templates


Templates
For configuring index - like mappings, settings, aliases
building block of a index template
Index Templates (_index_template)
Component Templates (_component_template)
template-1
template-2
aliases
mappings
settings
alias-1
settings-1
mapping-1
alias-1
settings-2
alias-1
mapping-1
settings-1
settings-2

Templates

- Index templates contain settings like how many shards and replicas the index should initialize with, what mapping settings, aliases to use.
- One can also assign priority to the index template.
- 100 is the default priority.
- Generally, indexes in the hot phase should have the highest value and indexes in the cold phase should have the lowest values.
- For example: 100 for the hot phase, 50 for the warm phase, and 0 for the cold phase.
- Indices that don’t set this value have a default priority of 1.

Templates

- Component templates are nothing but building blocks for index templates.
- Create an alias component template, settings component template, or a mapping component template and use them via the "composed_of" parameter.
- Elastic has built-in index templates which maps to index/datastream patterns logs-*-*,metrics-*-*, synthetics-*-*, which have default priority of 100.
- To create index template's which overrides the built-in index templates but use the same patterns, assign a priority that is above 100.
- To disable the built-in index and component templates, set the stack.templates.enabled as false.

Templates

```
GET /_template

GET /_template/metric_beat

PUT /_index_template/rps_template
{
  "index_patterns" : ["rps*", "virt*"],
  "priority" : 1,
  "template": {
    "settings" : {
        "number_of_shards" : 2,
        "number_of_replicas": 0
    }
  },
  "version": 1.0
}

GET /_index_template/rps_template
```

Templates

```
{
    "my_index_template": {
        "order": 0,
        "index_patterns": [
            "my_index-"
        ],
        "settings": {
            "index": {
                "lifecycle": {
                    "name": "my_index_policy",
                    "rollover_alias": "my_index"
                },
                "codec": "best_compression",
                "refresh_interval": "30s",
                "number_of_shards": "40",
                "number_of_replicas": "1"
            }
        },
        "mappings": {
            "properties": {
                "field1": {
                    "type": "keyword"
                },
                "field2": {
                    "type": "date"
                }
            }
        },
        "aliases": {
            "my_index": {}
        }
    }
}
```

Index Management

- Data in Elasticsearch is stored in one or more indices.
- Elasticsearch typically deal with large volumes of data, data in an index is partitioned across shards to make storage more manageable.
- An index may be too large to fit on a single disk, but shards are smaller and can be allocated across different nodes as needed.
- Another benefit of proper sharding is that searches can be run across different shards in parallel, speeding up query processing.
- The number of shards in an index is decided upon index creation and cannot be changed later.

Index Management

- Sharding an index is useful, still only a single copy of each document in the index, which means there is no protection against data loss.
- We can set up replication.
- Each shard may have several replicas, which are configured upon index creation and may be changed later.
- The primary shard is the main shard that handles the indexing of documents and can also handle processing of queries.
- The replica shards process queries but do not index documents directly.
- They are always allocated to a different node from the primary shard, and, in the event of the primary shard failing, a replica shard can be promoted to take its place.

Index Management

- While more replicas provide higher levels of availability in case of failures, it is also important not to have too many replicas.
- Each shard has a state that needs to be kept in memory for fast access.
- The more shards we use, the more overhead can build up and affect resource usage and performance.

Index Lifecycle Management

- Index Lifecycle management helps manage all data in the indices, leverage the underlying hardware capabilities.
- Time-series Data has different tenures, and only a sizeable chunk needs to be actively queried all the time

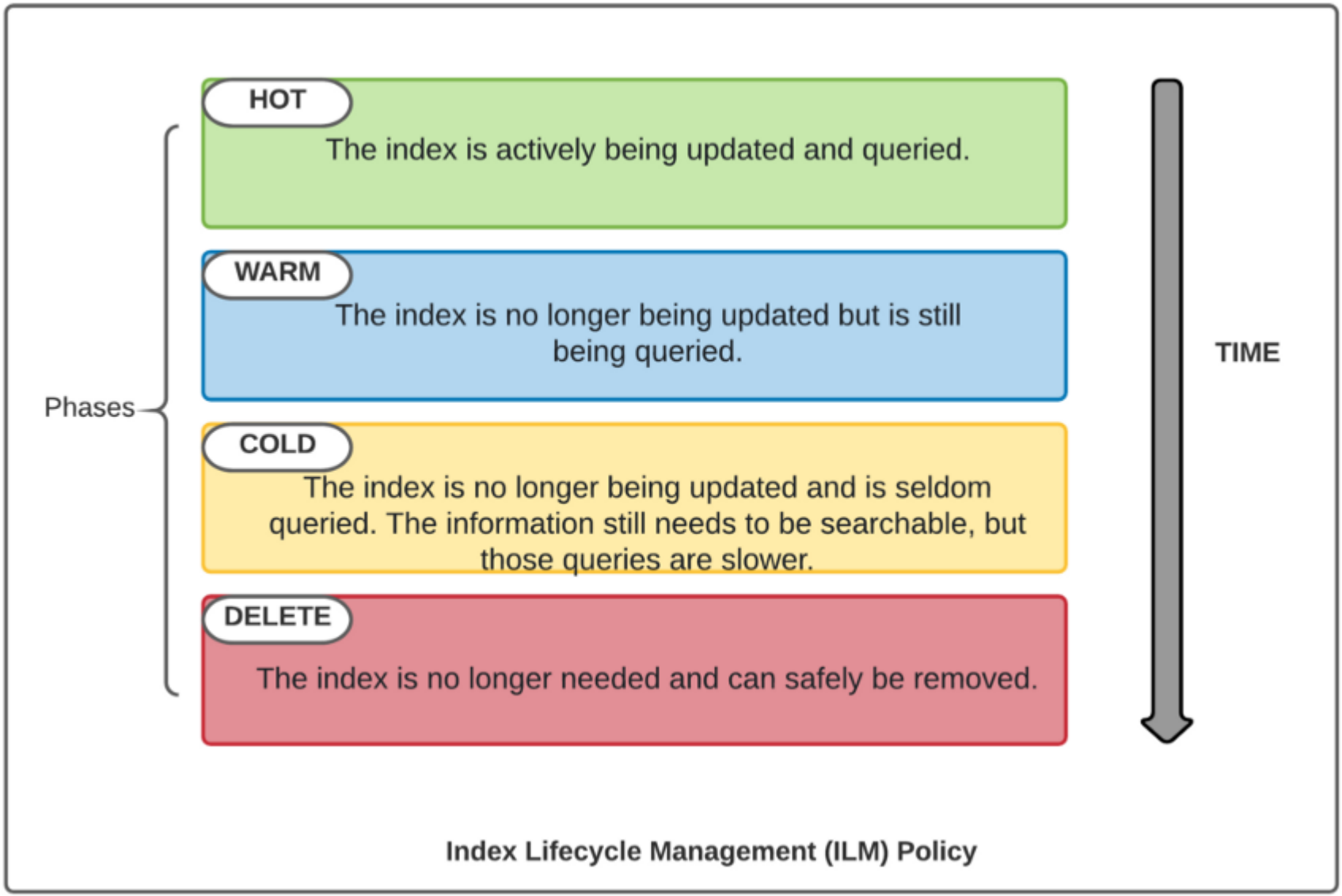
Index Lifecycle Management

- Elasticsearch ILM easily manages indexes in hot-warm-cold phase architectures which is the efficient way of working with time series data.
- We can specify the condition to rollover to a new index in the following ways:
- By specifying the number of documents, maximum shard size, or age at which we want to roll over
- The point at which the index is no longer being used or queried and we can reduce the number of primary shards
- The point at which to force a merge to remove documents permanently which are marked for deletion

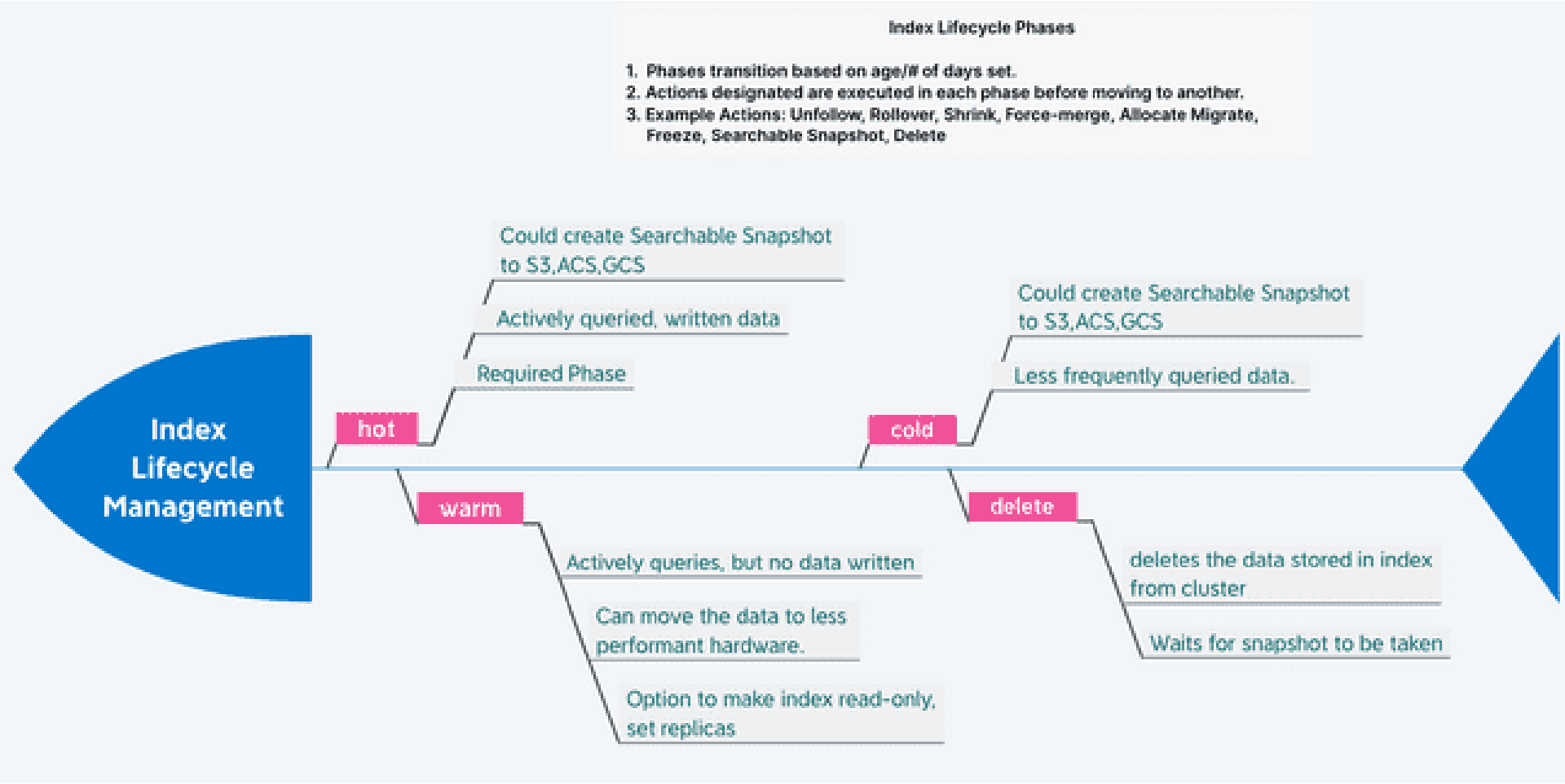
Index Lifecycle Management

- The point at which an index can be moved to a less expensive storage
- The point at which the number of replicas can be reduced and the availability is not as critical
- The point at which an index is not required and can be deleted safely

Index Lifecycle Management



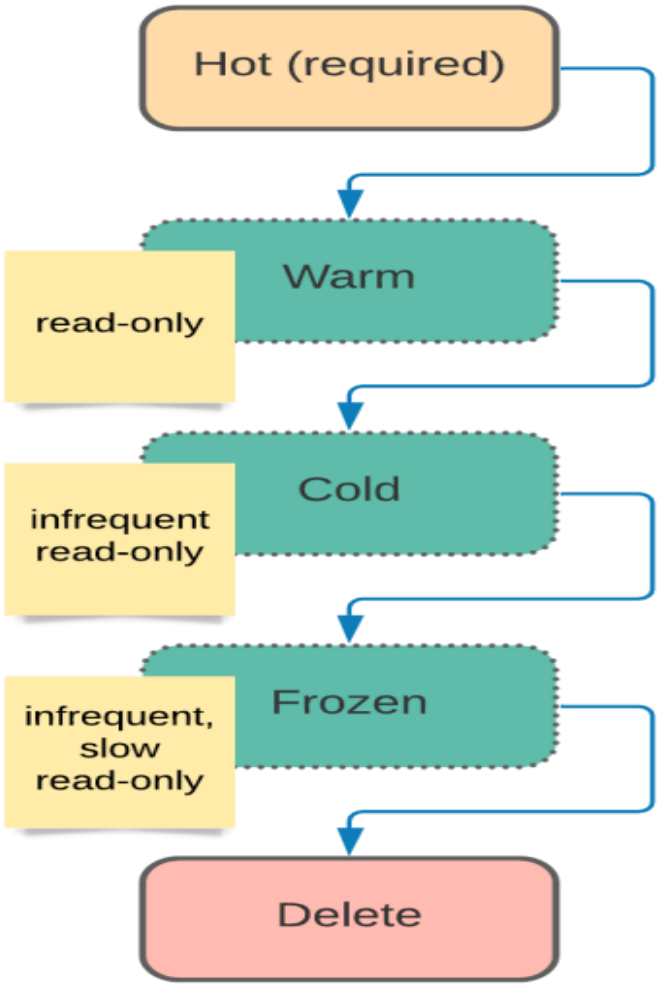
Index Lifecycle Management



Index Lifecycle Management

- To check ILM status
- GET _ilm/policy
- GET my-index-ilm/_ilm/explain

Index Lifecycle Management




```
PUT _ilm/policy/ilm_aken
{
  "policy": {
    "phases": {
      "hot": {
        "min_age": "0ms",
        "actions": {
          "rollover": {
            "max_size": "50gb",
            "max_age": "1m"
          },
          "set_priority": {
            "priority": 100
          }
        }
      },
      "warm": {
        "actions": {}
      },
      "cold": {
        "min_age": "2m",
        "actions": {}
      },
      "delete": {
        "min_age": "3m",
        "actions": {}
      }
    }
  }
}
```

Index Lifecycle Management

- ILM has policies that are instructions configured to execute on indices containing data.
- These instructions are like triggers that create, rollover, deletes data based on set conditions.
- There can be multiple ILM policies with different priorities set.
- For example, if we are ingesting 1 GB, logs, metrics, and traces data from the apps and infrastructure.
- By the end of the month, it becomes 30 GB, quarter: 90 GB, An year: 360+ GB.

Index Lifecycle Management

- We might not need entire raw data to gather insights, as it does not change with time, but you might be interested in comparing trends.
- With ILM, you could gather critical insights and rollover the raw data to less expensive hardware-based nodes or delete the cluster's unneeded data.
- Besides, using the data tiers (hot, warm, cold), you could move the index data to designated nodes and have faster hardware for faster performance.
- ILM automates this entire process of moving data to different types of nodes.

Index Lifecycle Management

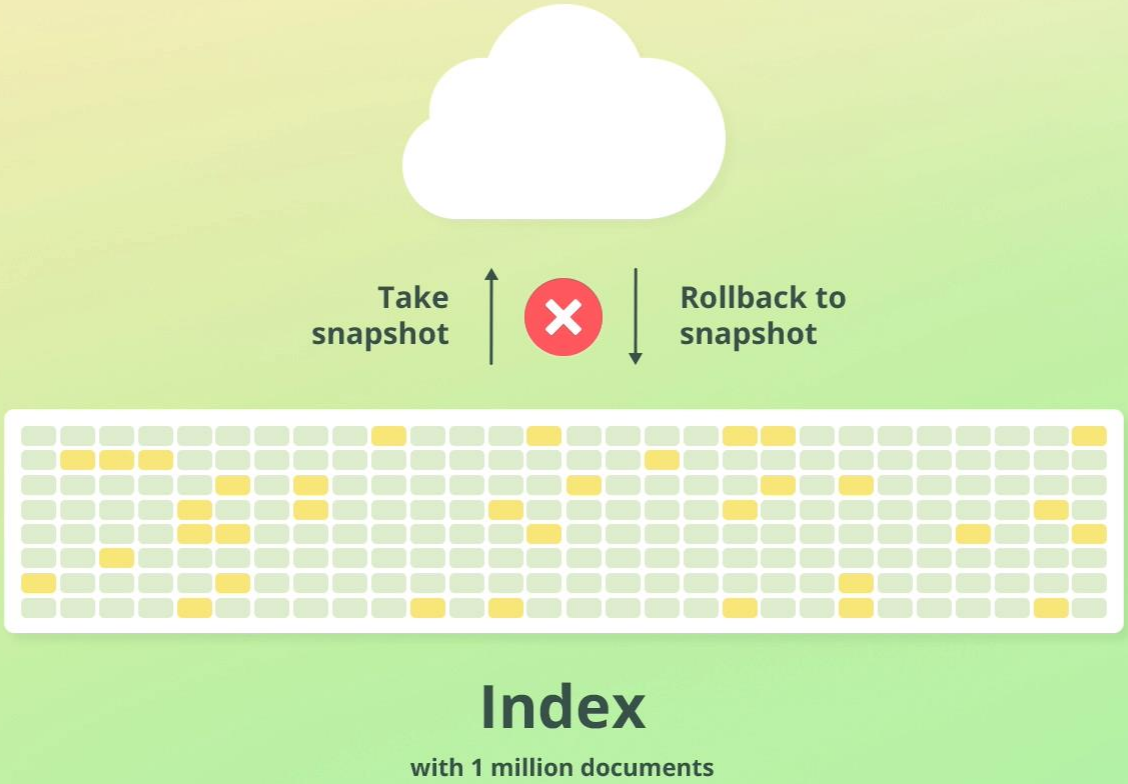
```
#create policy

PUT _ilm/policy/datastream_policy
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "50GB",
            "max_age": "30d"
          }
        }
      },
      "delete": {
        "min_age": "90d",
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

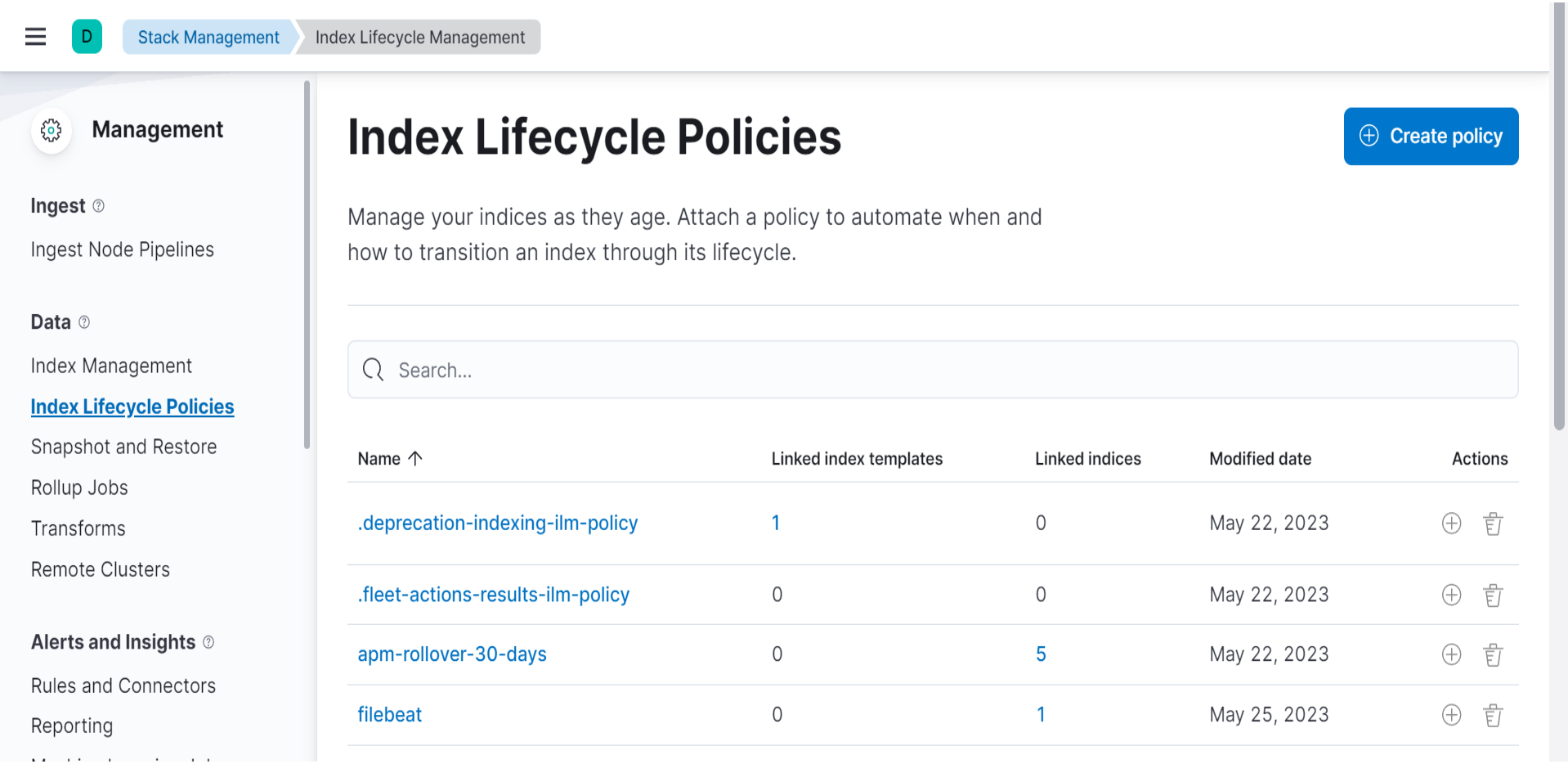
Index Lifecycle Management

```
#apply apolicy
PUT _template/datastream_template
{
  "index_patterns": ["datastream-*"],
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1,
    "index.lifecycle.name": "datastream_policy",
    "index.lifecycle.rollover_alias": "datastream"
  }
}
```

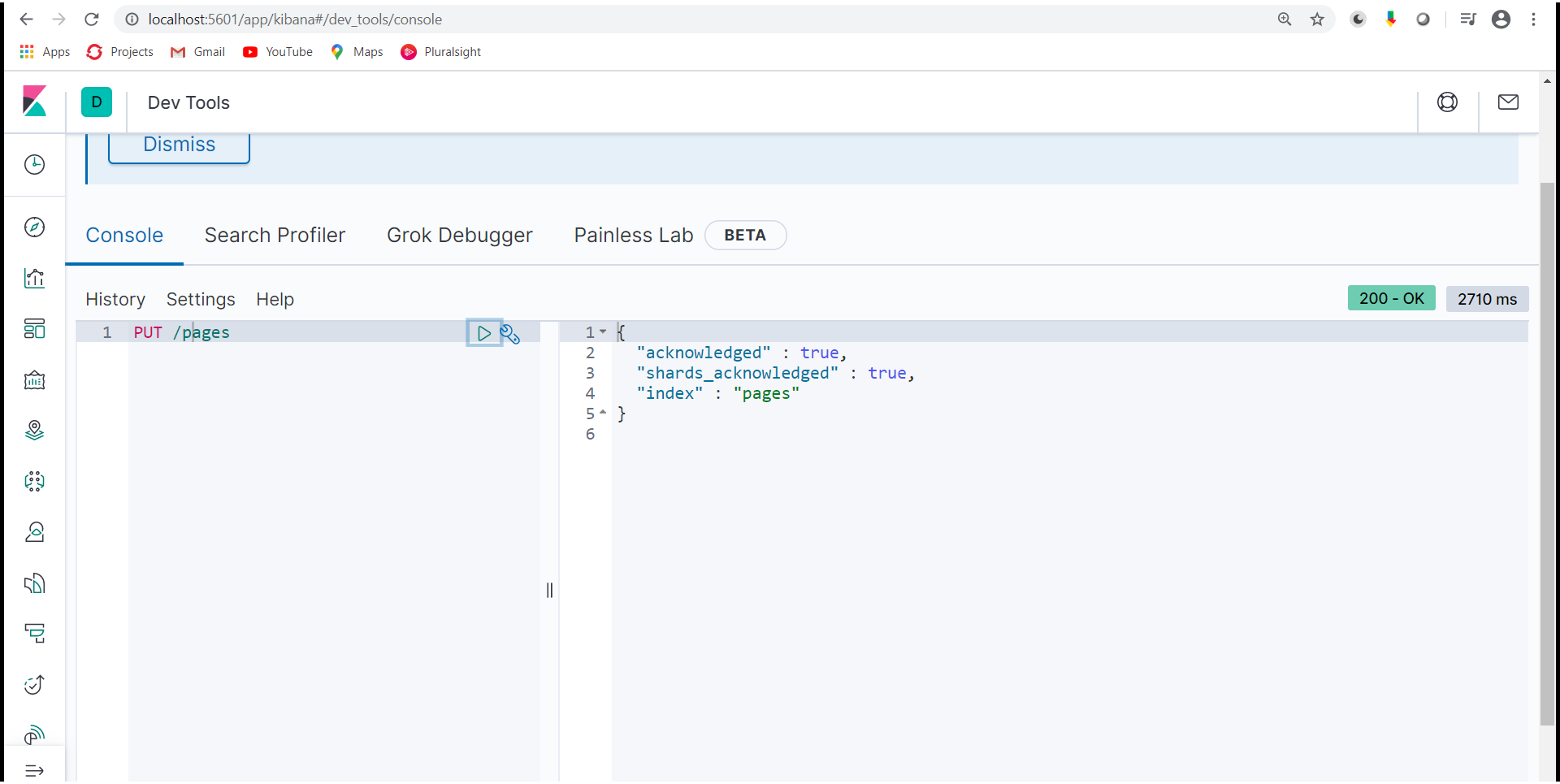
Index


Take
snapshot
Rollback to
snapshot
Index
with 1 million documents

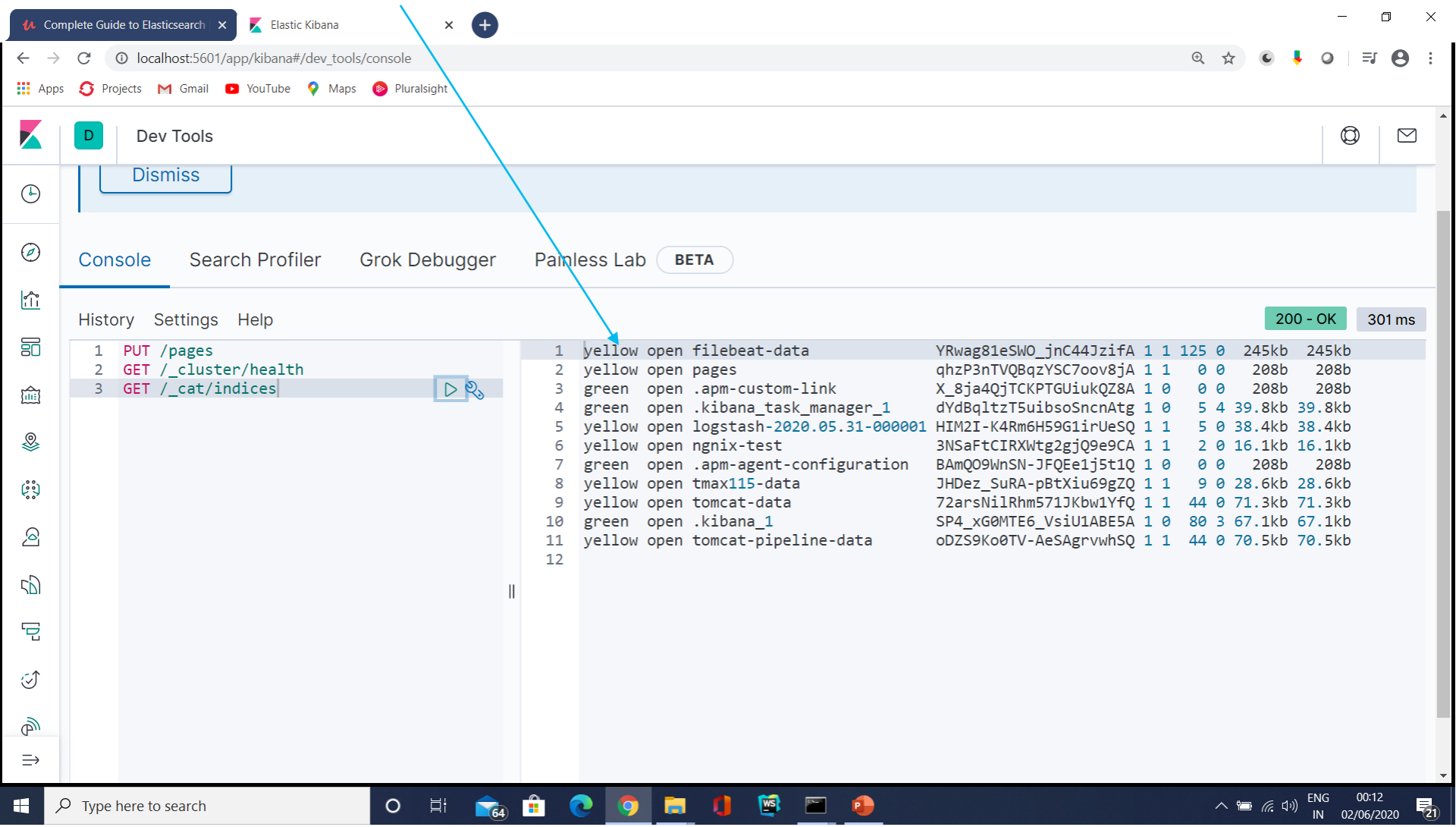
Index Lifecycle Management



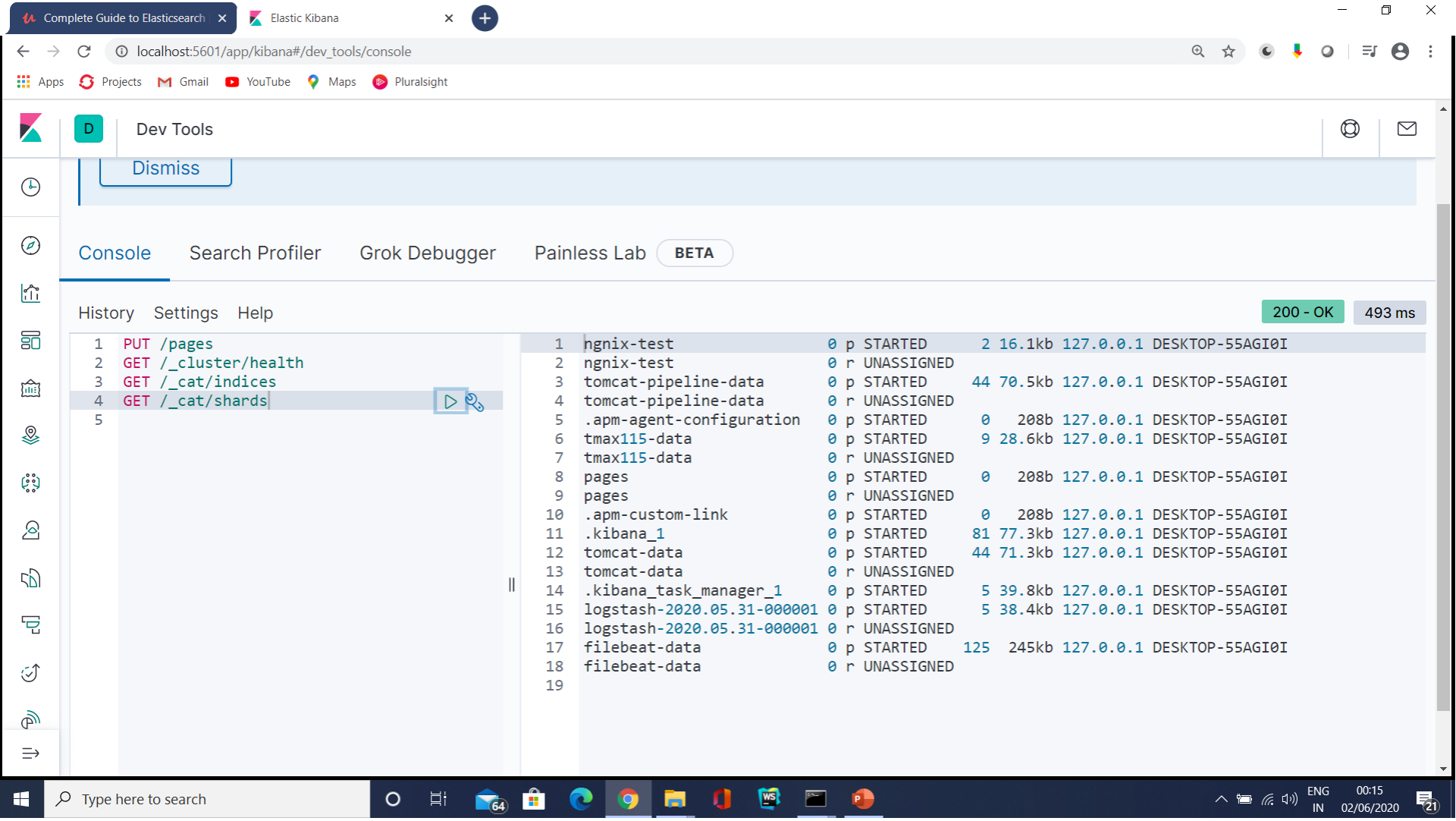
Managing Documents – Create Index



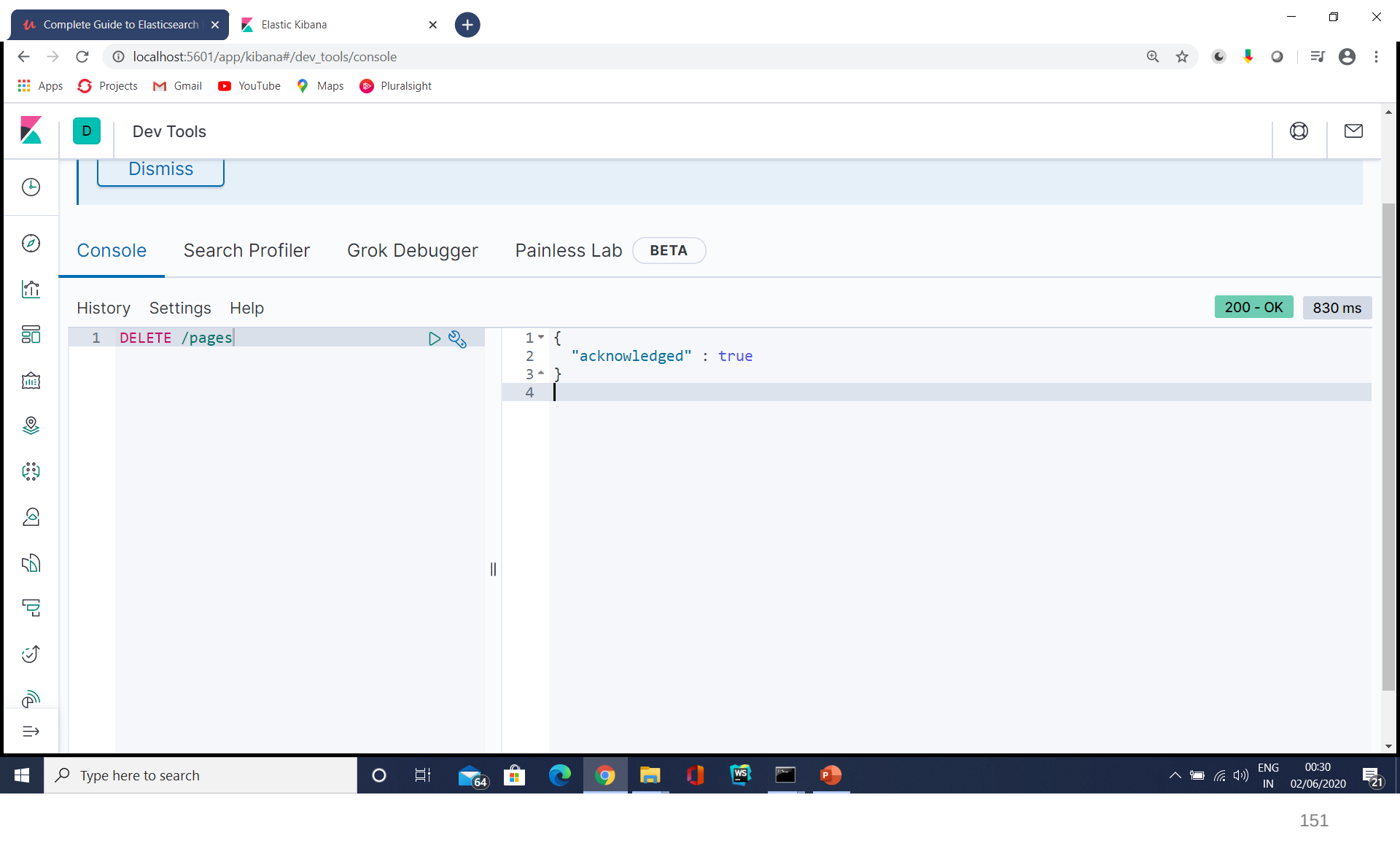
Managing Documents – Create Index



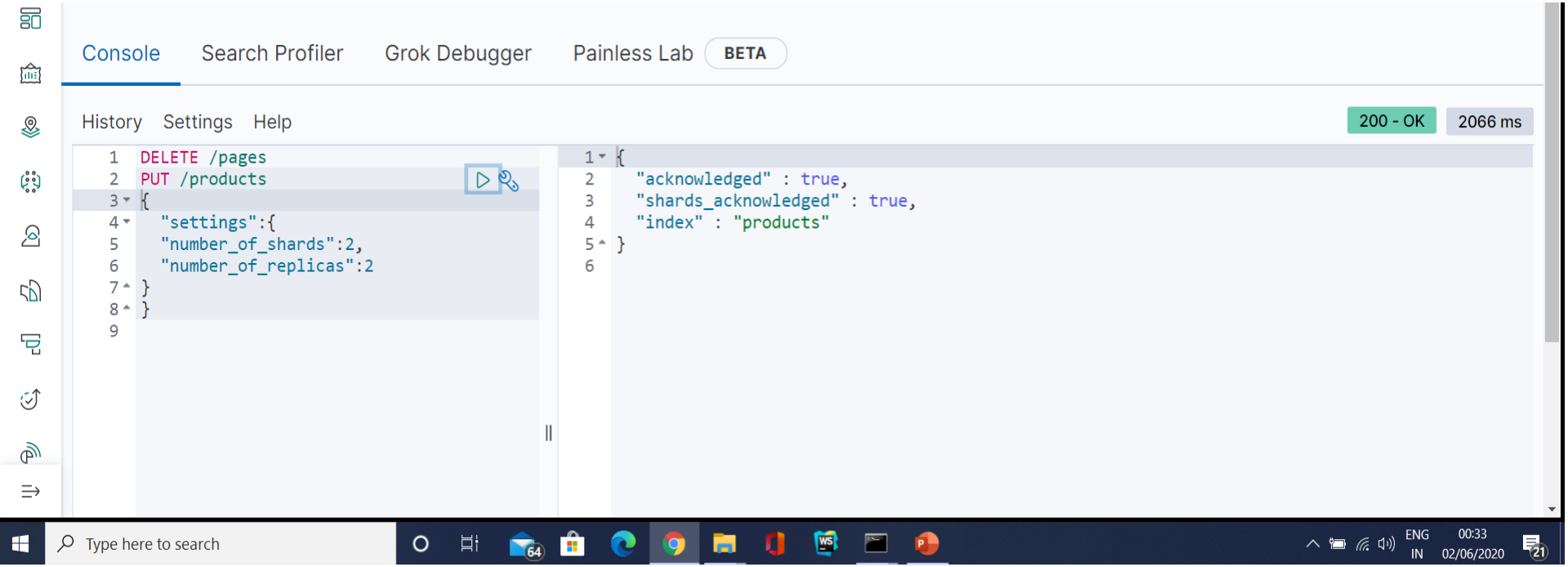
Managing Documents – Create Index



Creating and Deleting indices



Creating and Deleting indices



Alias Indexing

- Add an alias
- The following request adds the alias1 alias to the test1 index.
- POST /_aliases
- {
- "actions" : [
- { "add" : { "index" : "test1", "alias" : "alias1" } }
-]
- }

Index Template

- An Elasticsearch index template is a method used to instruct Elasticsearch to configure indices upon creation.
- For example, an index template used on a data stream configures the stream’s backing indices upon creation.
- An index template is created manually before index creation.
- When creating an index, the template applies the configuration settings for the index.

Index Template

- Index templates define settings and mappings that you can automatically apply when creating new indices.
- Elasticsearch applies templates to new indices based on an index pattern that matches the index name.
- Index templates are only applied during index creation.
- Changes to index templates do not affect existing indices.
- Settings and mappings specified in create index API requests override any settings or mappings specified in an index template.

How can I view the current templates?

Console | Search Profiler | Grok Debugger | Painless Lab BETA

History | Settings | Help

200 - OK | 65 ms

```
GET _template
```

```
#! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.15/security-minimal-setup.html to enable security.
{
  ".kibana-event-log-7.15.0-template" : {
    "order" : 0,
    "index_patterns" : [
      ".kibana-event-log-7.15.0-*"
    ],
    "settings" : {
      "index" : {
        "lifecycle" : {
          "name" : "kibana-event-log-policy",
          "rollover_alias" : ".kibana-event-log-7.15.0"
        },
        "number_of_shards" : "1",
        "auto_expand_replicas" : "0-1"
      }
```

Search for specific template

- GET _template/filebeat-*

How to Create an Index Template

```
PUT _index_template/template_1
{
  /* Define the index pattern */
  "index_patterns" : ["te*"],
  "priority" : 1,
  /* Define settings for the indices*/
  "template": {
    "settings" : {
      "number_of_shards" : 2
    }
  }
}
```

Body of Index Template

- Template: The template property (object) defines which template to be applied; it can include aliases, mappings, and settings—this an optional parameter.
- Composed_of: This property defines a list of names for component templates. Once defined, component templates get compounded in their specification order. That means the last component template defined takes the highest precedence.

Body of Index Template

- Priority: The priority property defines the precedence of the index template when creating an index. If any precedence has the highest value, it gets higher precedence compared to lower values. The priority value is not required and is of type integer. 0 is the default value for non-specified templates.
- Version: The version parameter specifies the index template version, which helps to manage the templates.

Index Template with Version

```
PUT /_index_template/template_2
{
  "index_patterns" : ["remp*", "re*"],
  "priority" : 1,
  "template": {
    "settings" : {
        "number_of_shards" : 2,
        "number_of_replicas": 0
    }
  },
  "version": 1.0
}
```

Alias

- An alias is a secondary name for a group of data streams or indices.
- Most Elasticsearch APIs accept an alias in place of a data stream or index name.
- You can change the data streams or indices of an alias at any time.
- If you use aliases in your application’s Elasticsearch requests, you can reindex data with no downtime or changes to your app’s code.

Alias

- Alias types
- There are two types of aliases:
  - A data stream alias points to one or more data streams.
  - An index alias points to one or more indices.
  - An alias cannot point to both data streams and indices. You also cannot add a data stream’s backing index to an index alias.

Add Alias

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "logs-nginx.access-prod",
        "alias": "logs"
      }
    }
  ]
}
```

Add Alias

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "logs-*",
        "alias": "logs"
      }
    }
  ]
}
```

Add or Remove Alias

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "logs-nginx.access-prod",
        "alias": "logs"
      }
    },
    {
      "add": {
        "index": "logs-my_app-default",
        "alias": "logs"
      }
    }
  ]
}
```

Reindexing

- Reindex is the concept of copying existing data from a source index to a destination index which can be inside the same or a different cluster.
- Elasticsearch has a dedicated endpoint _reindex for this purpose.
- A reindexing is mostly required for updating mapping or settings.

Reindexing in the same cluster

- POST /_reindex?pretty
- {
- "source": {
- "index": "news"
- },
- "dest": {
- "index": "news_v2"
- }
- }

Reindex Steps

- Step 1
- Get all existing index data and note down the docs count of the index which is going to reindex.

- curl -X GET \
- 'http://localhost:9200/_cat/indices/%2A?v=&s=index:desc' \

Reindex Steps

- Create a new index with newly updated mapping and versioning. We shall call it companydatabase_v2
- curl -X PUT \
- http://localhost:9200/companydatabase_v2 \
- -H 'Content-Type: application/json' \
- -d '{
- "mappings": {
- "properties": {
- "FirstName": {
- "type": "text"
- },
- "Gender": {
- "type": "text"
- },
- "LastName": {
- "type": "text"
- },
- "Salary": {
- "type": "text"
- }
- }
- }
- }'

Reindex Steps

- Create a new index with newly updated mapping and versioning. We shall call it companydatabase_v2
- curl -X PUT \
- http://localhost:9200/companydatabase_v2 \
- -H 'Content-Type: application/json' \
- -d '{
- "mappings": {
- "properties": {
- "FirstName": {
- "type": "text"
- },
- "Gender": {
- "type": "text"
- },
- "LastName": {
- "type": "text"
- },
- "Salary": {
- "type": "text"
- }
- }
- }
- }'

Reindex Steps

- Step 3
- Now we shall transfer data of old index into the newly created index with _reindex command. In our case, we are transferring data from companydatabase to companydatabase_v2
- curl -X POST \
- http://localhost:9200/_reindex \
- -H 'Content-Type: application/json' \
- -d '{
- "source": {
- "index": "companydatabase"
- },
- "dest": {
- "index": "companydatabase_v2"
- }
- }'

Reindex Steps

- Step 4
- Always double-check that the newly created index should have the exact number of documents of the old index. We have 50,000 documents in our old index companydatabase which are copied to the newly created index companydatabase_v2.
- Confirm newly created index has an equal number of documents with the old index using the curl request shown below.
- curl -X GET \
- 'http://localhost:9200/companydatabase_v2/_search?scroll=10m&size=50' \
- -H 'Content-Type: application/json' \
- -d '{
- "query" : {
- "match_all" : {}
- }
- }'

Reindex Steps

- Once it is confirmed that all both the index has equal number of documents, it is safe to delete older index.
- curl -X DELETE \
- http://localhost:9200/companydatabase

Using Indices

RESTful API

Elasticsearch fundamentally works via HTTP requests and JSON data. Any language or tool that can handle HTTP can use Elasticsearch.

Client API's

Most languages have specialized Elasticsearch libraries to make it even easier.

Analytic Tools

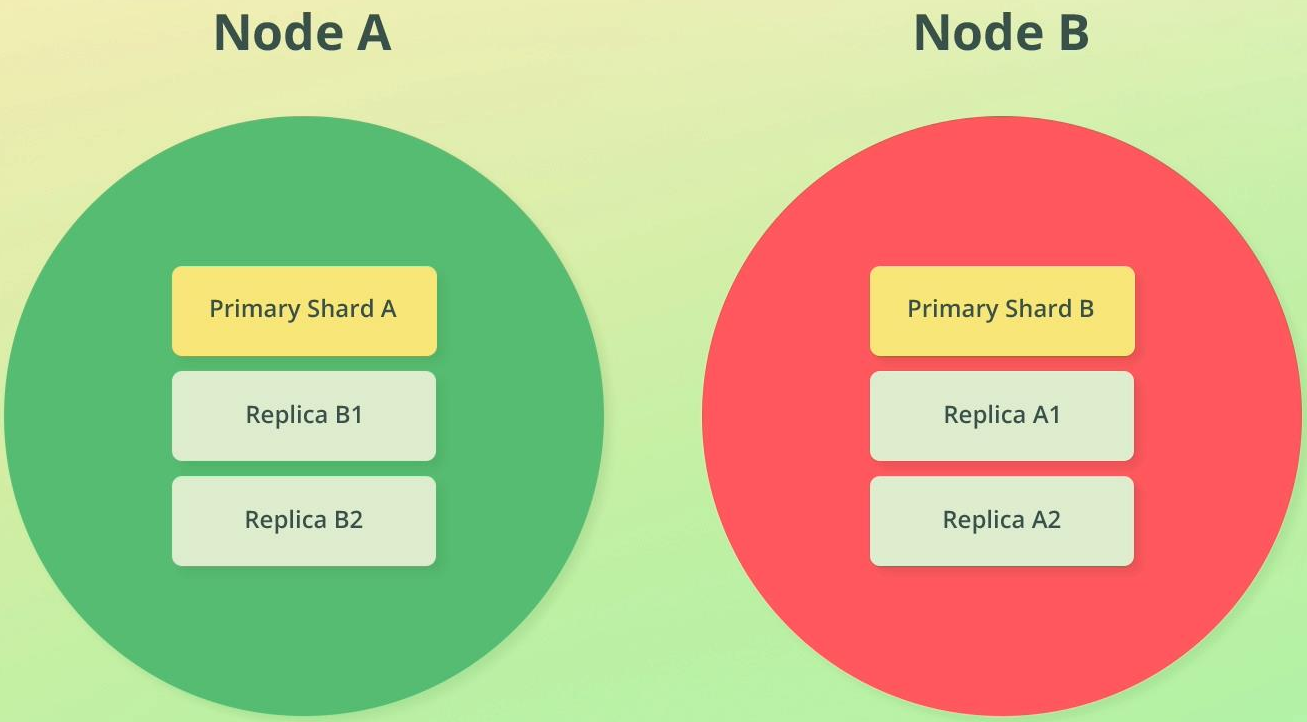
Web-based graphical UI's such as Kibana let you interact with your indices and explore them without writing code.

How does replication work

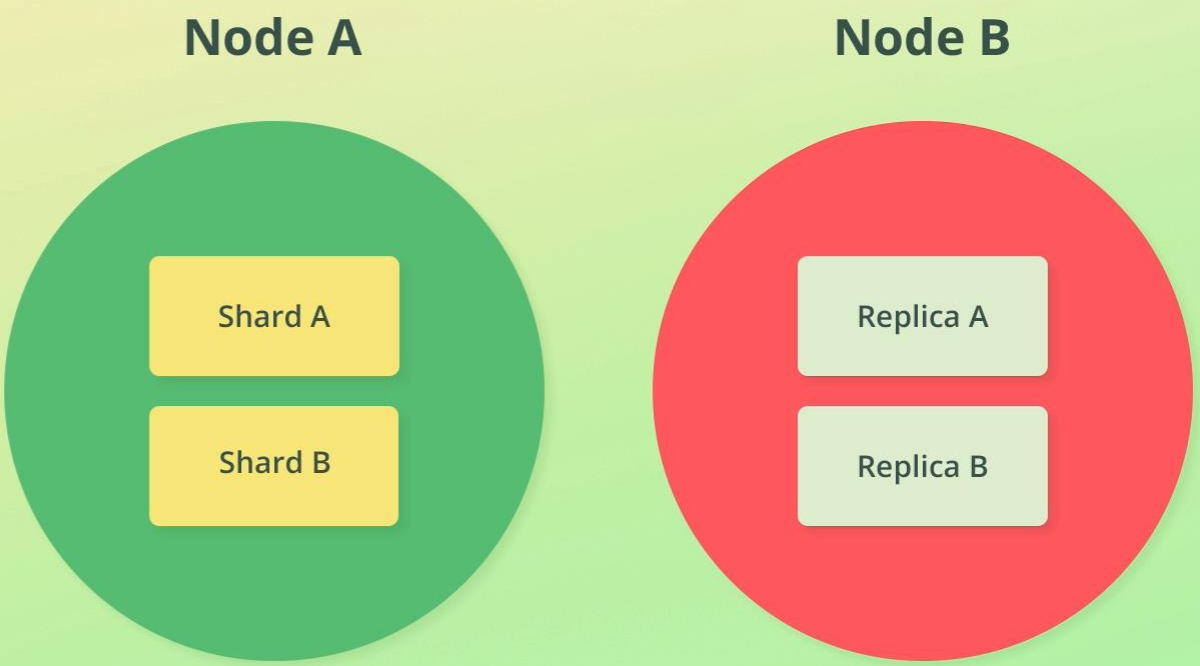
- Replication is configured at the index level
- Replication works by creating copies of shards, referred to as *replica shards*
- A shard that has been replicated, is called a *primary shard*
- A primary shard and its replica shards are referred to as a *replication group*
- Replica shards are a complete copy of a shard
- A replica shard can serve search requests, exactly like its primary shard
- The number of replicas can be configured at index creation

How does replication work

How does replication work



How does replication work



Choosing number of replica shards

- How many replica shards are ideal, depends on the use case
- E.g. is the data stored elsewhere, such as in a RDBMS?
- Is it OK for data to be unavailable while you restore it?
- For mission critical systems, downtime is not acceptable
- Replicate shards *once* if data loss is not a disaster
- For critical systems, data should be replicated *at least* twice

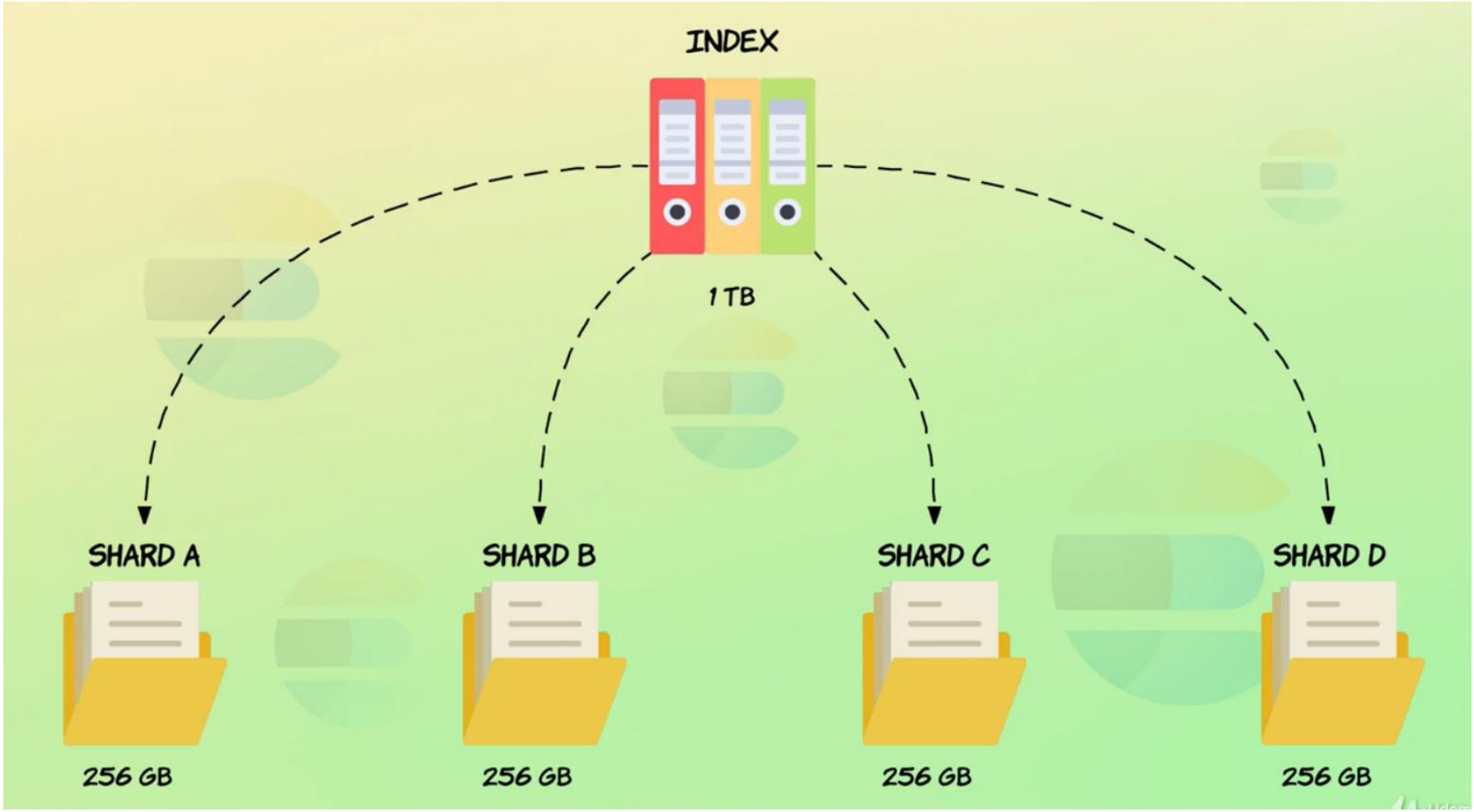
Shards

- Elasticsearch supports taking snapshots as backups
- Snapshots can be used to restore to a given point in time
- Snapshots can be taken at the index level, or for the entire cluster
- Use snapshots for backups, and replication for high availability (and performance)

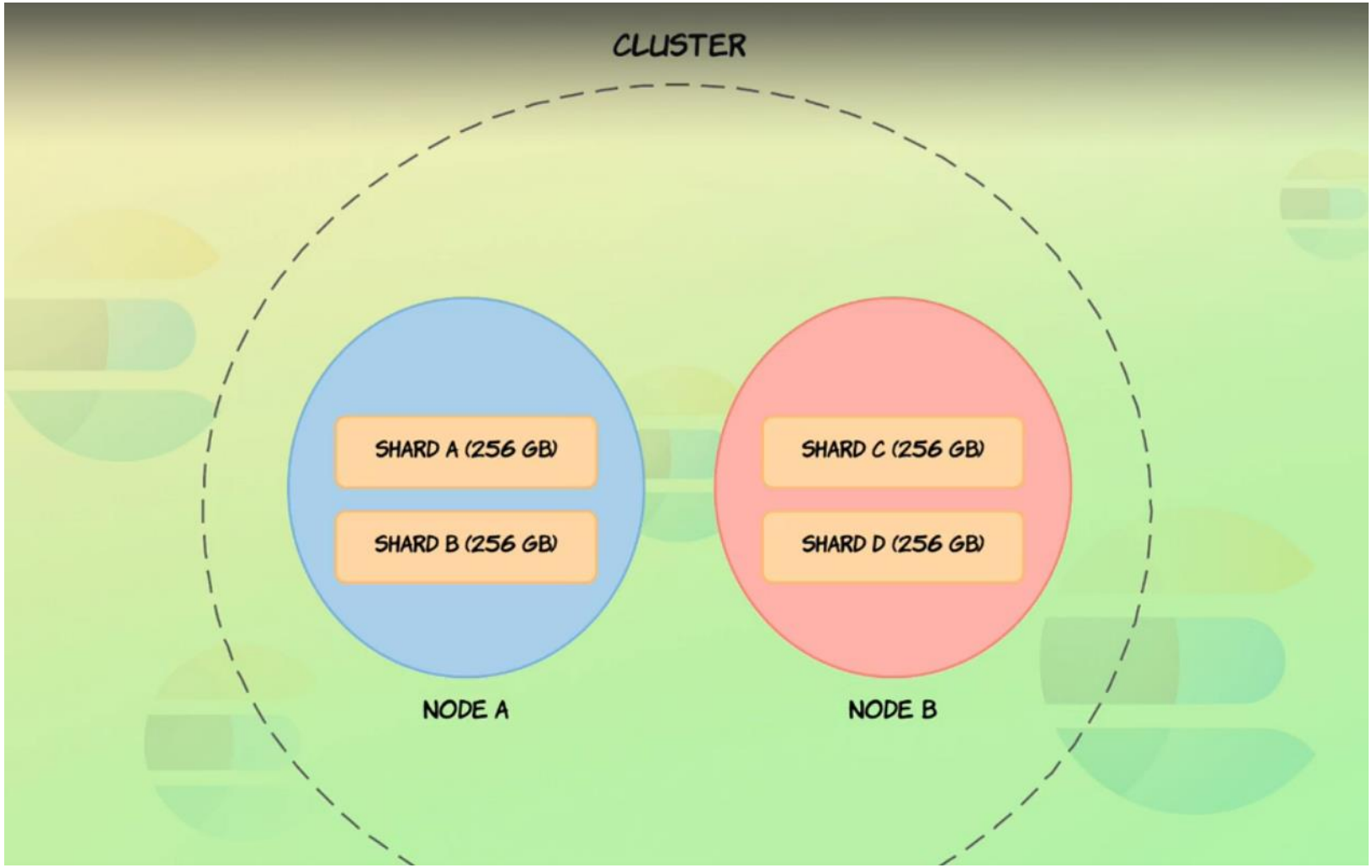
Increasing query throughput with replication

- Replica shards of a replication group can serve different search requests simultaneously
  - This increases the number of requests that can be handled at the same time
- Elasticsearch intelligently routes requests to the *best* shard (more on that later)
- CPU parallelization improves performance if multiple replica shards are stored on the same node

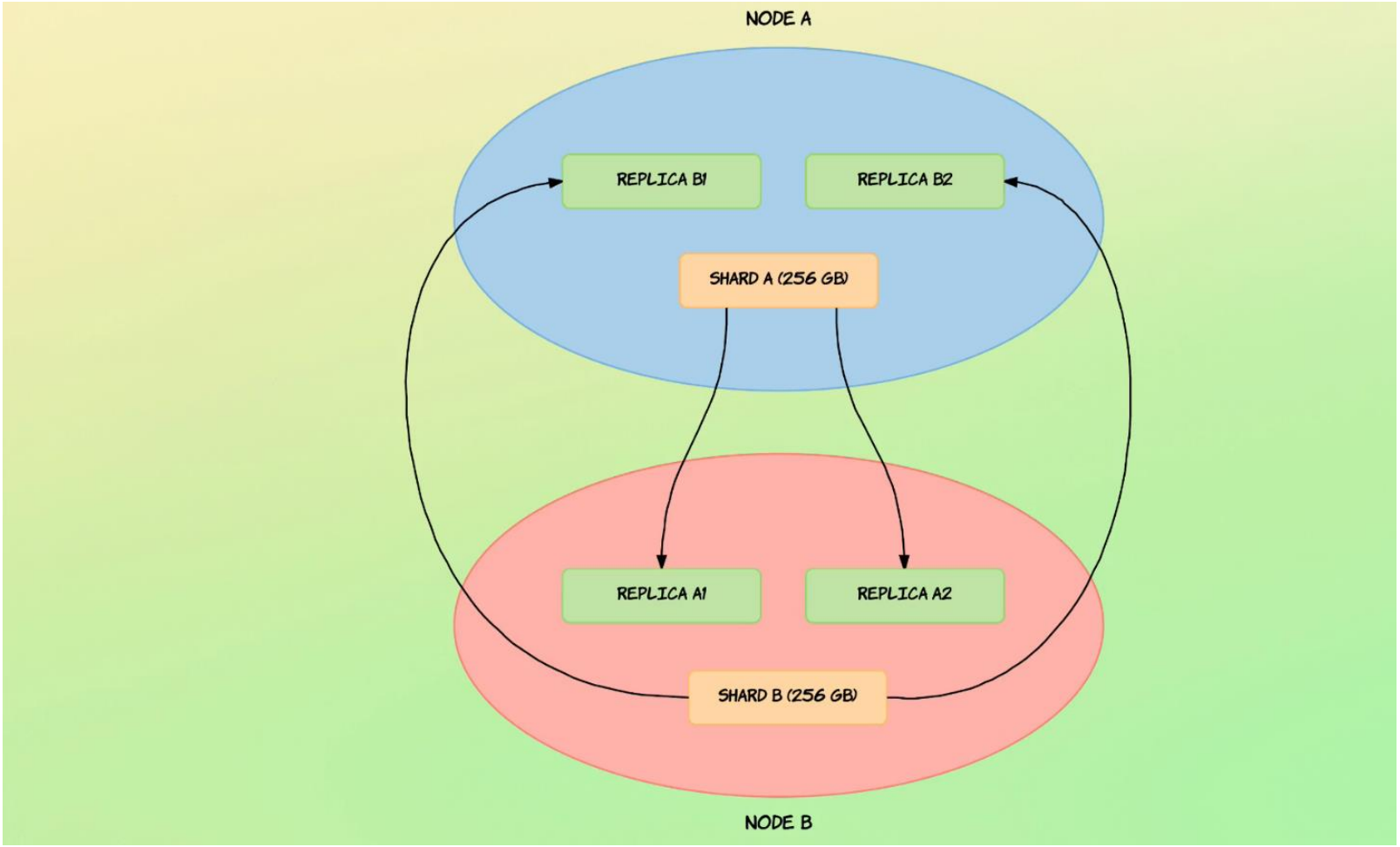
Sharding


INDEX
1 TB
SHARD A
SHARD B
SHARD C
SHARD D
256 GB
256 GB
256 GB
256 GB

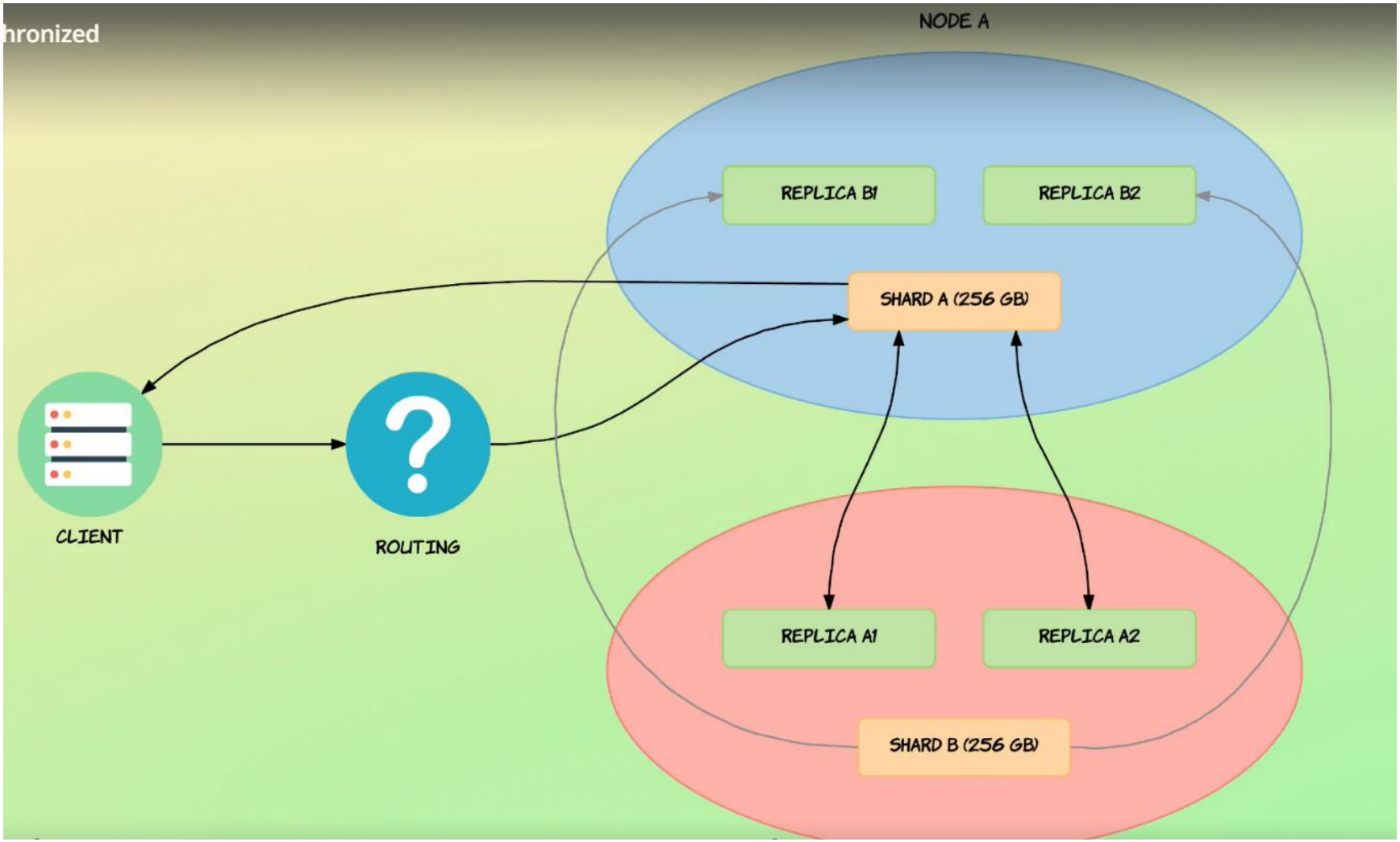
Sharding


CLUSTER
SHARD A (256 GB)
SHARD B (256 GB)
SHARD C (256 GB)
SHARD D (256 GB)
NODE A
NODE B

Replicas


NODE A
REPLICA B1
REPLICA B2
SHARD A (256 GB)
REPLICA A1
REPLICA A2
SHARD B (256 GB)
NODE B

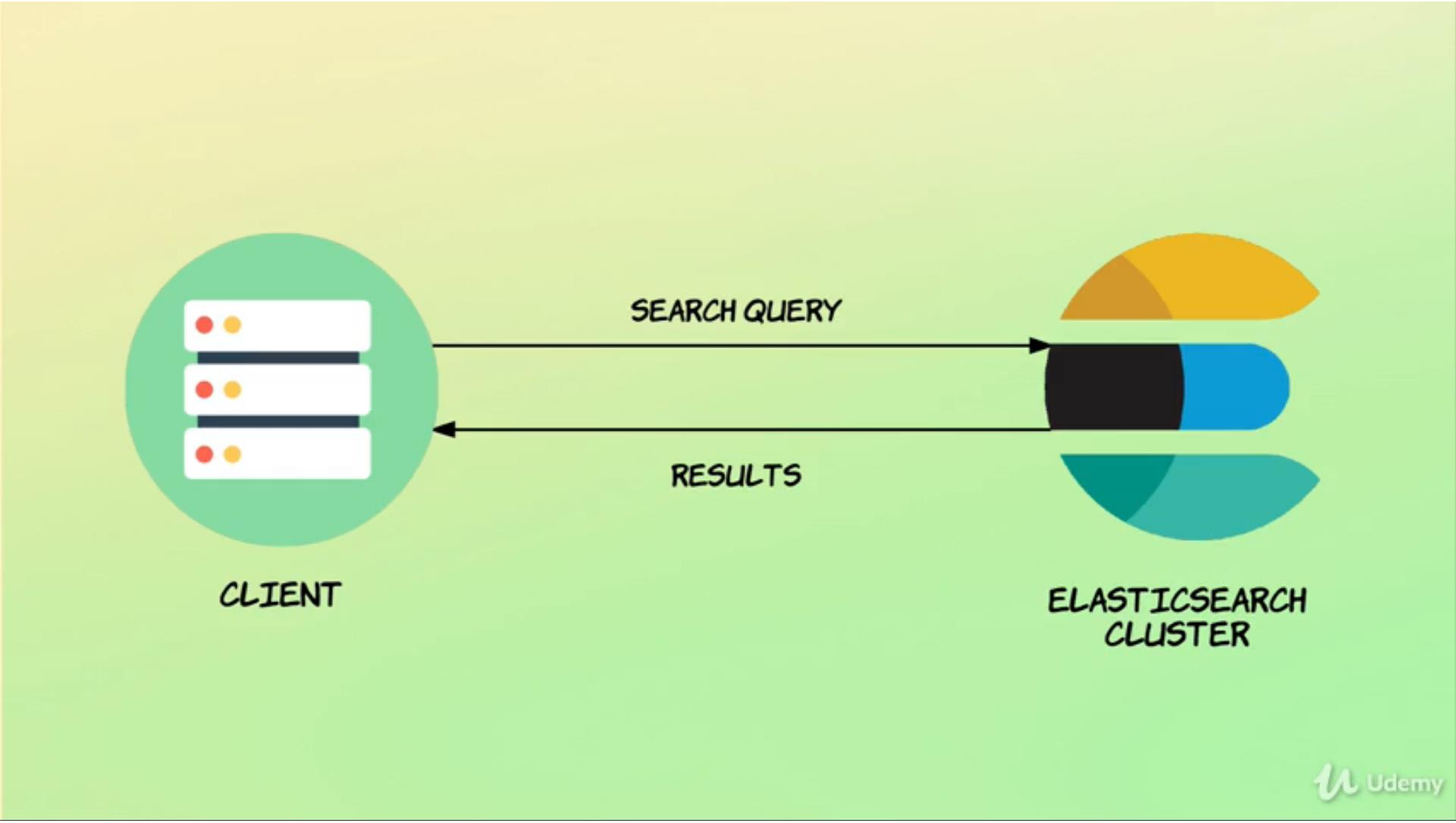
Replicas and Shards



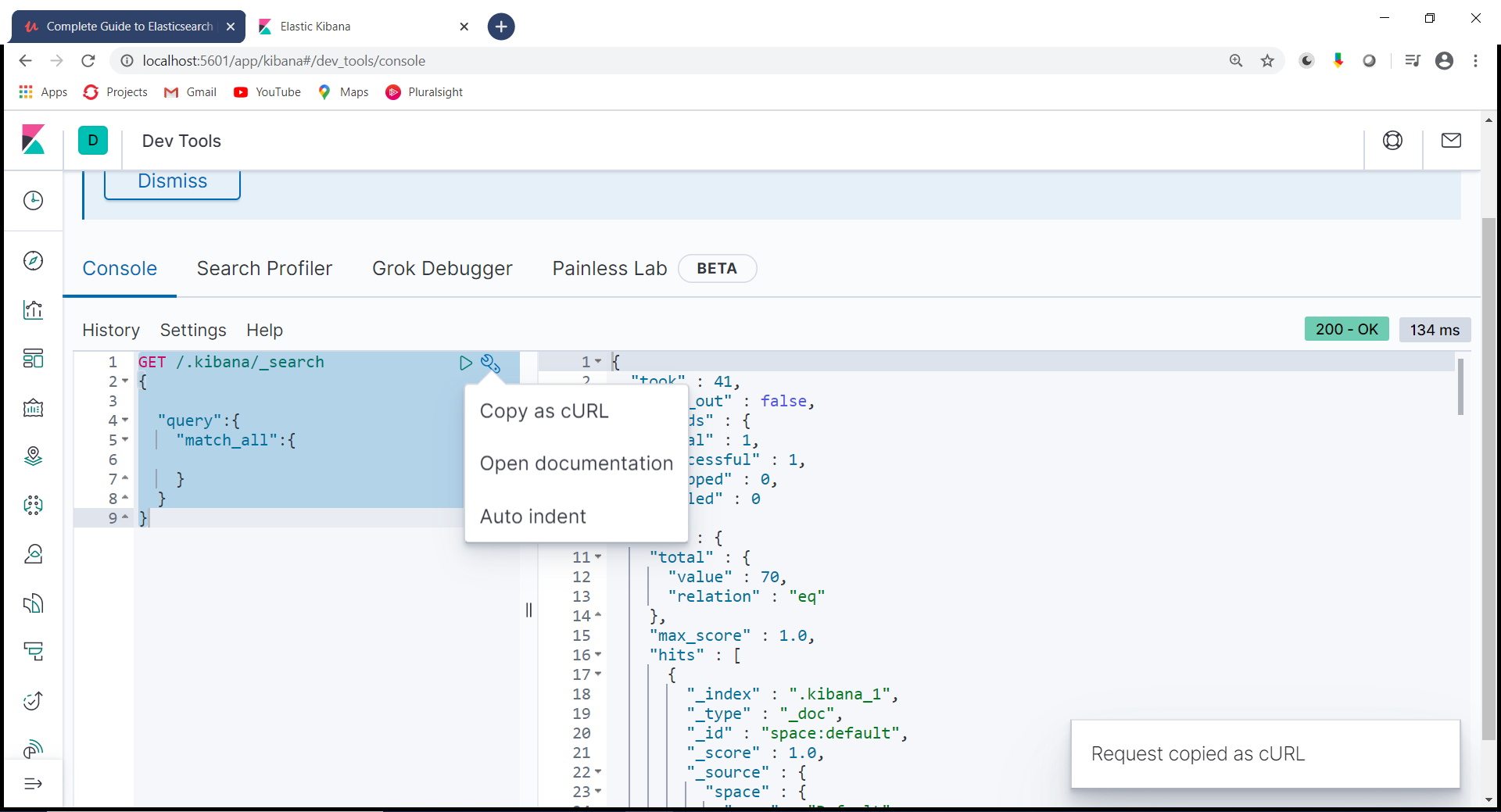
Elastic Health

- **Red**: Some or all primary shards are not ready to receive the requests.
- **Yellow**: All primary shards are allocated but some or all of the replicas are unallocated. For single node clusters, the normal status should be yellow because no other node is available for replication.
- **Green**: All shards and their replicas are well allocated and the cluster is working fine.

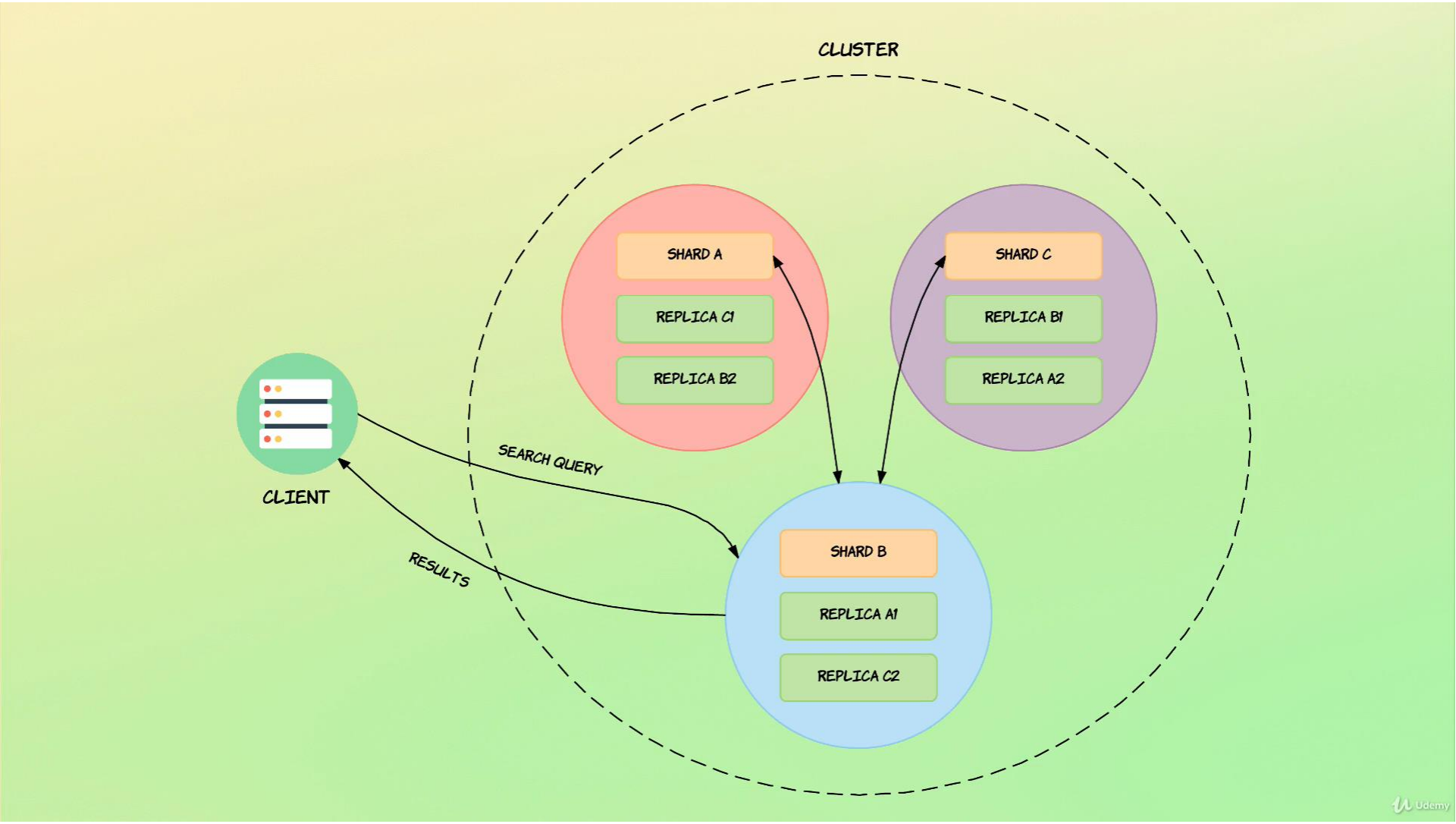
Search Query



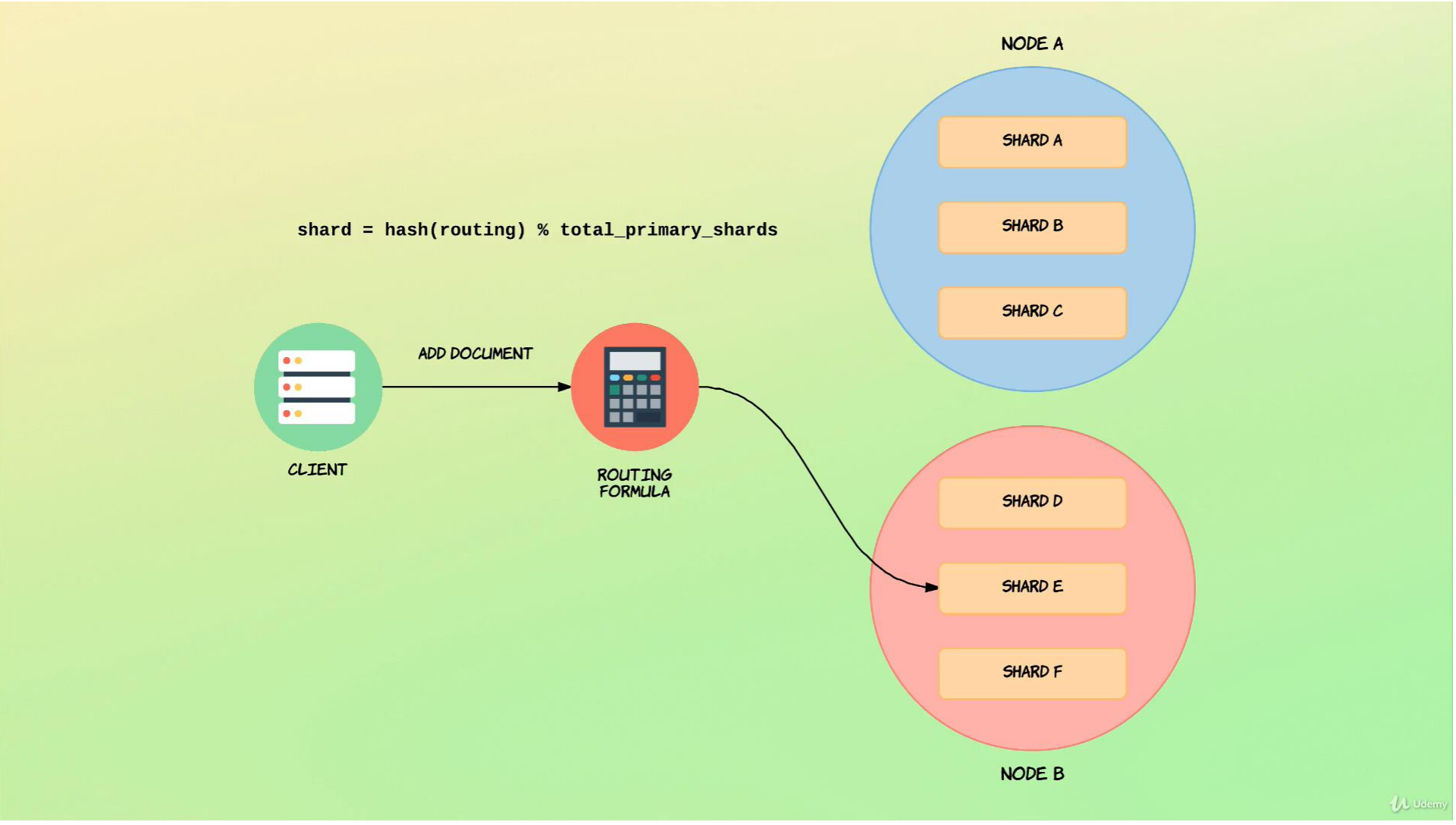
curl -XGET "http://localhost:9200/.kibana/_search" -H 'Content-Type: application/json' -d'{ "query":{ "match_all":{ } }}'



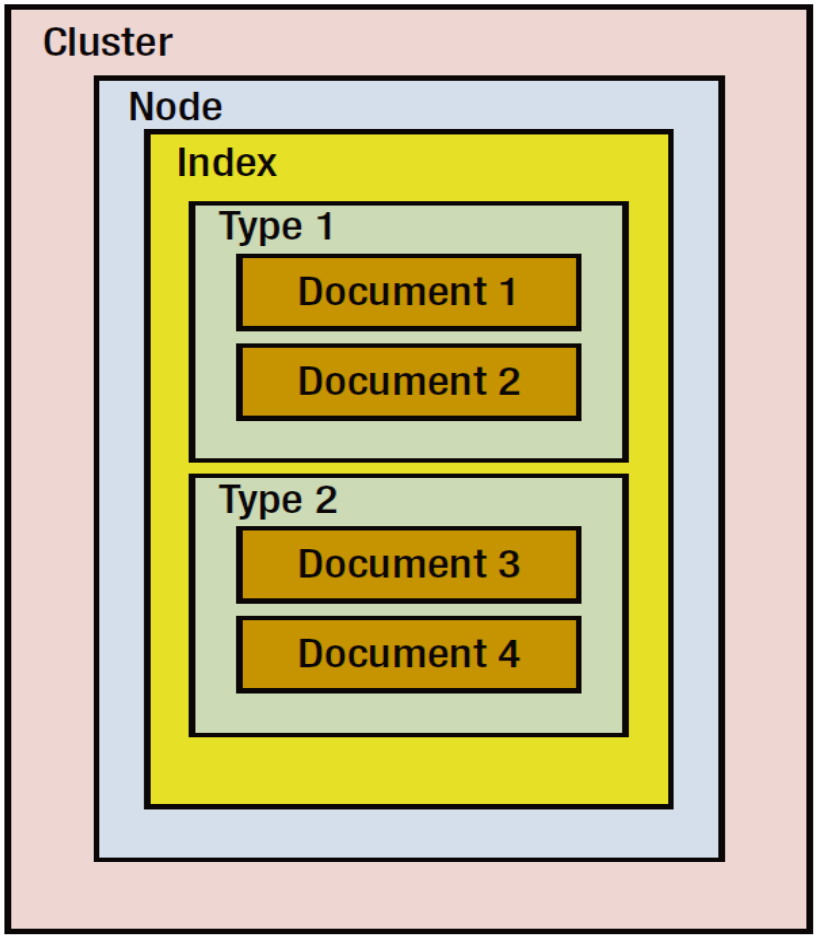
Search Query



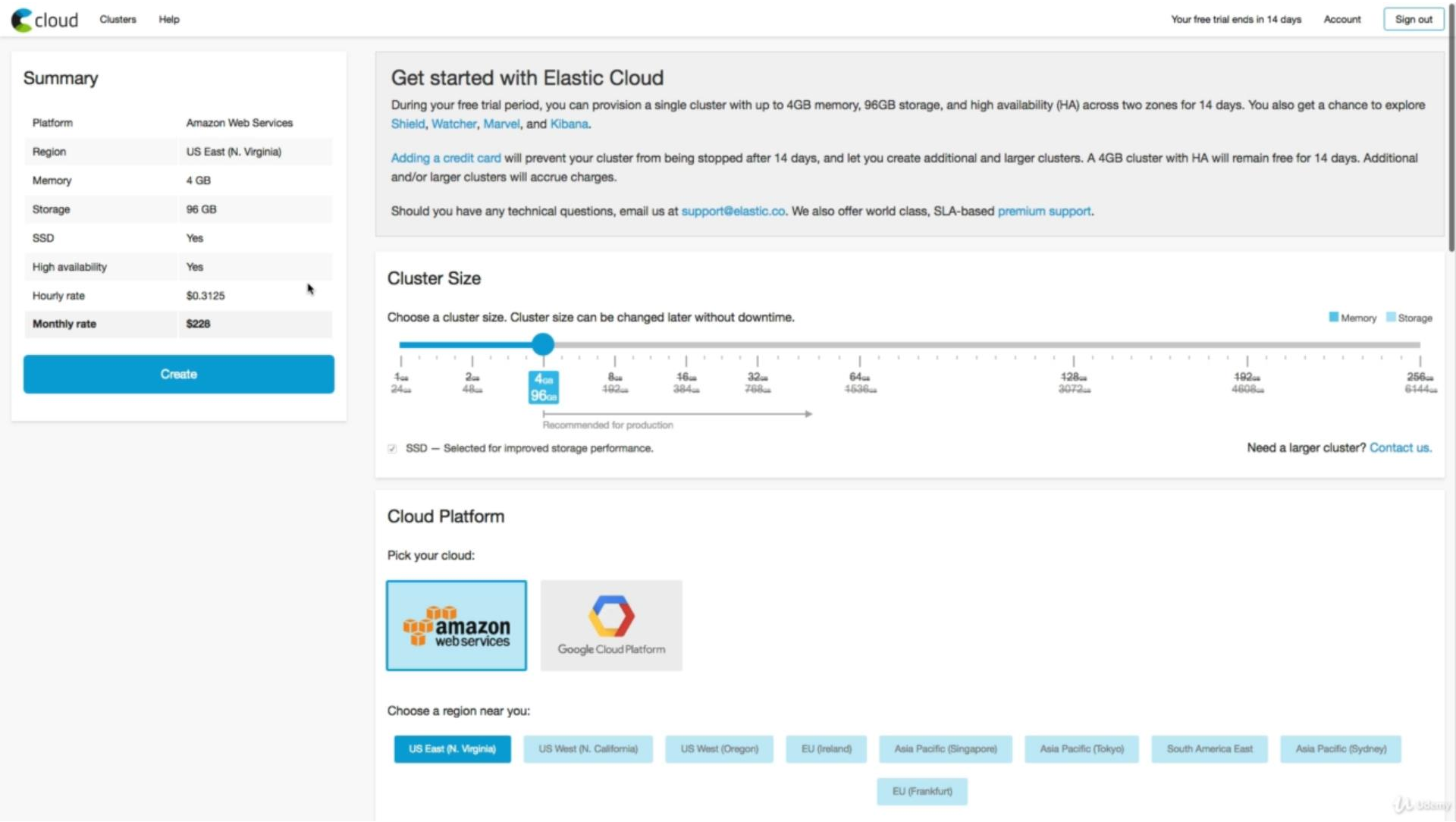
Search Query



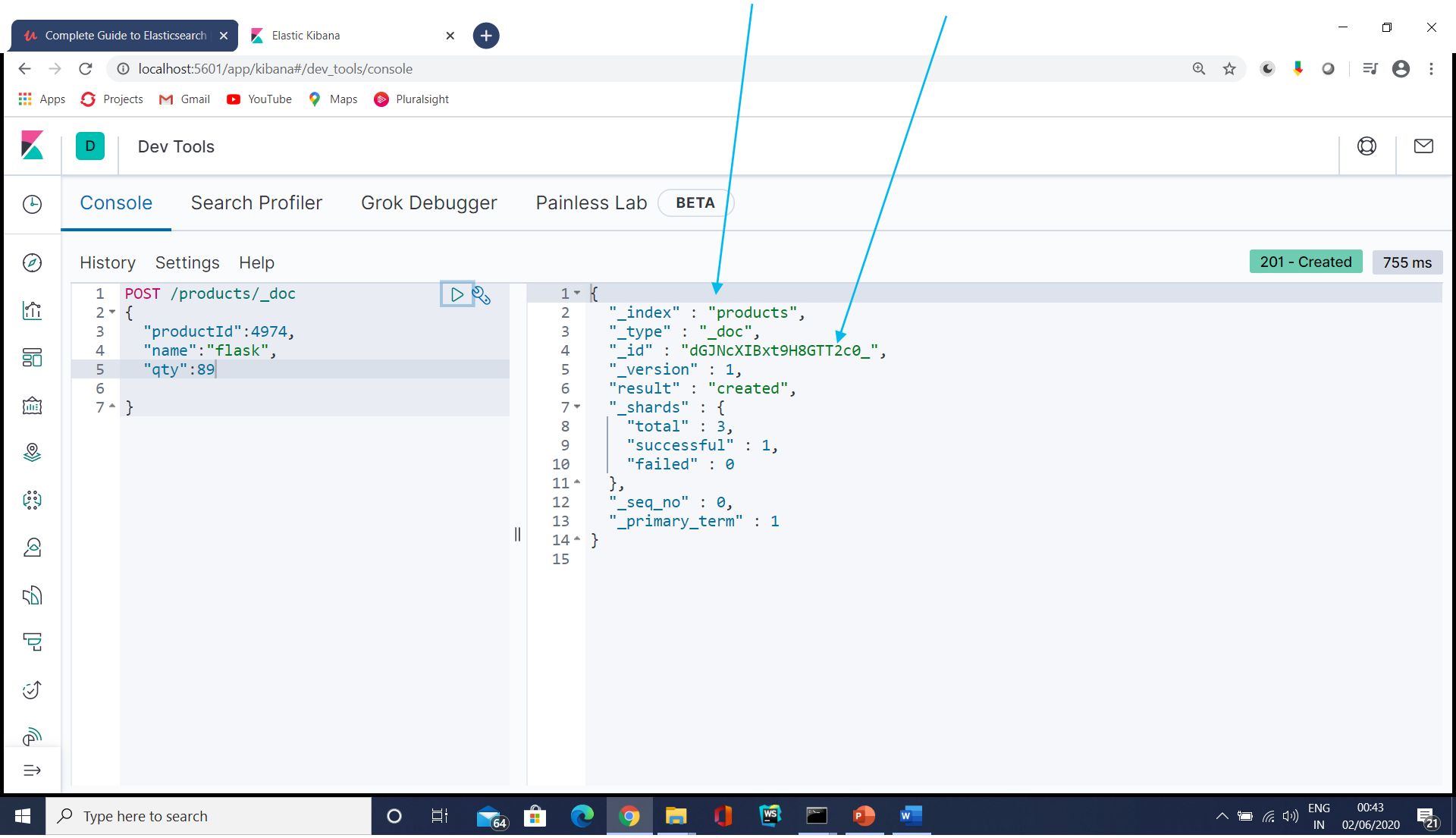
Search Query



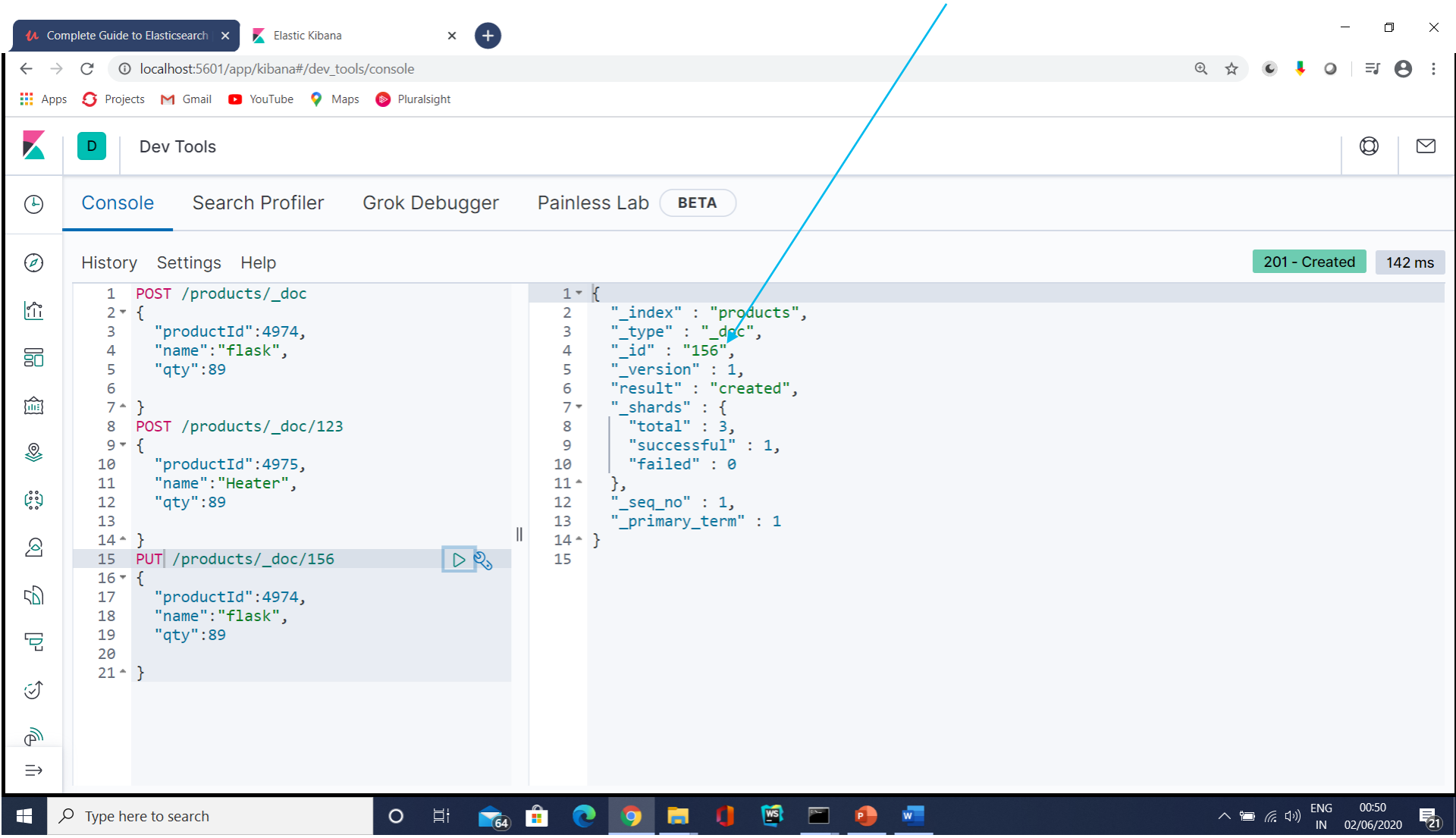
Elastic Cloud



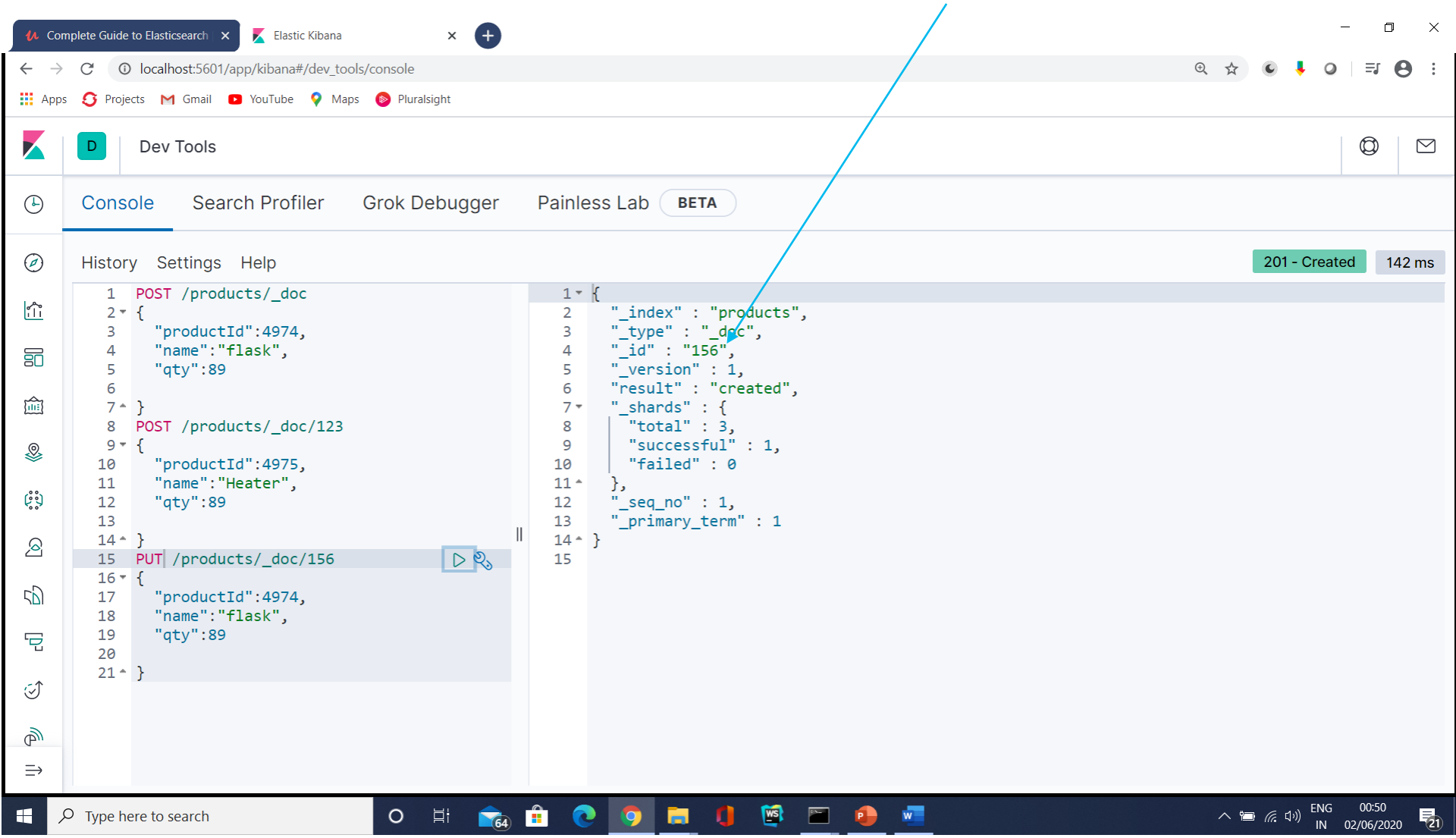
Create Indexing documents



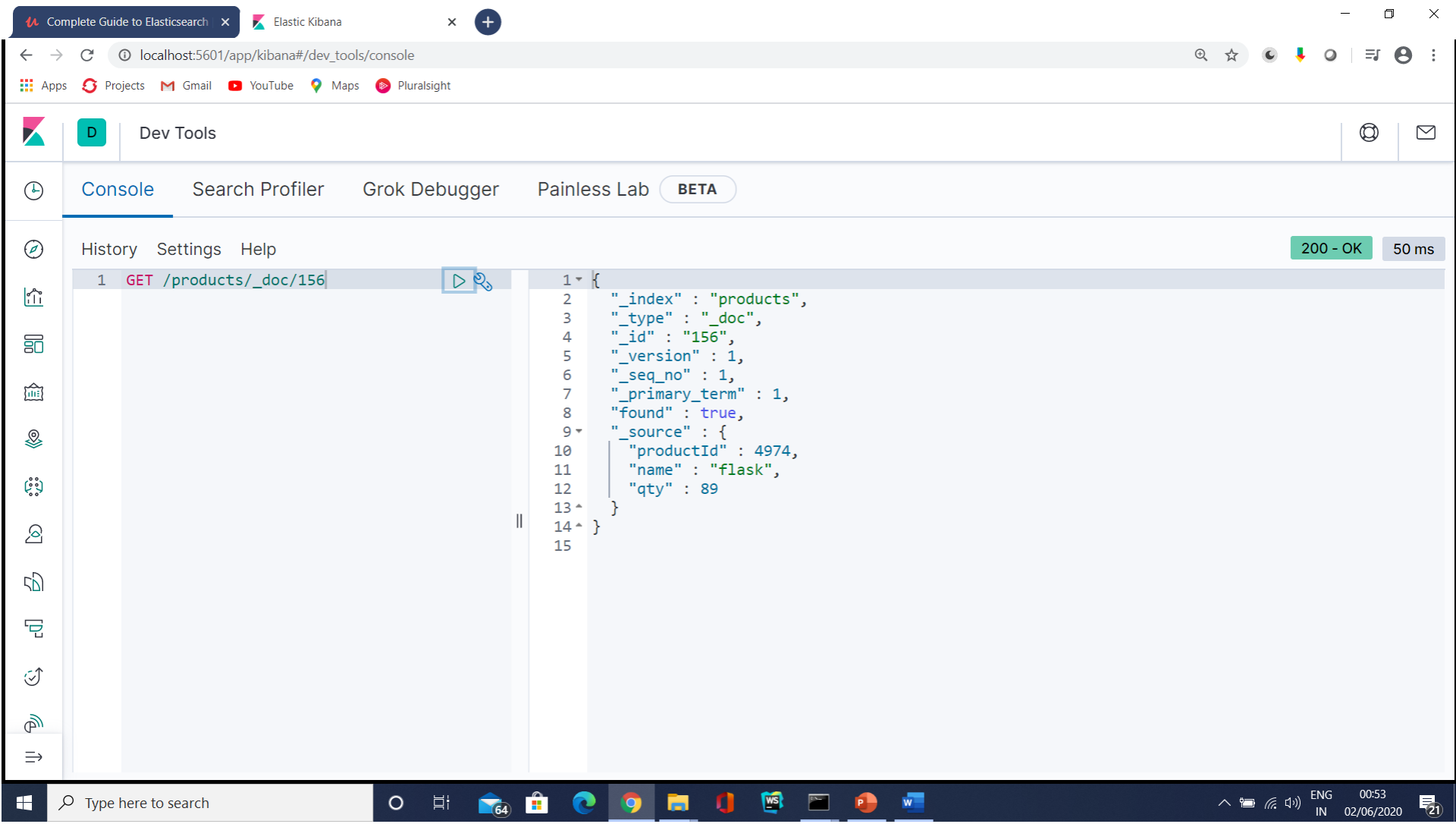
Create Indexing documents



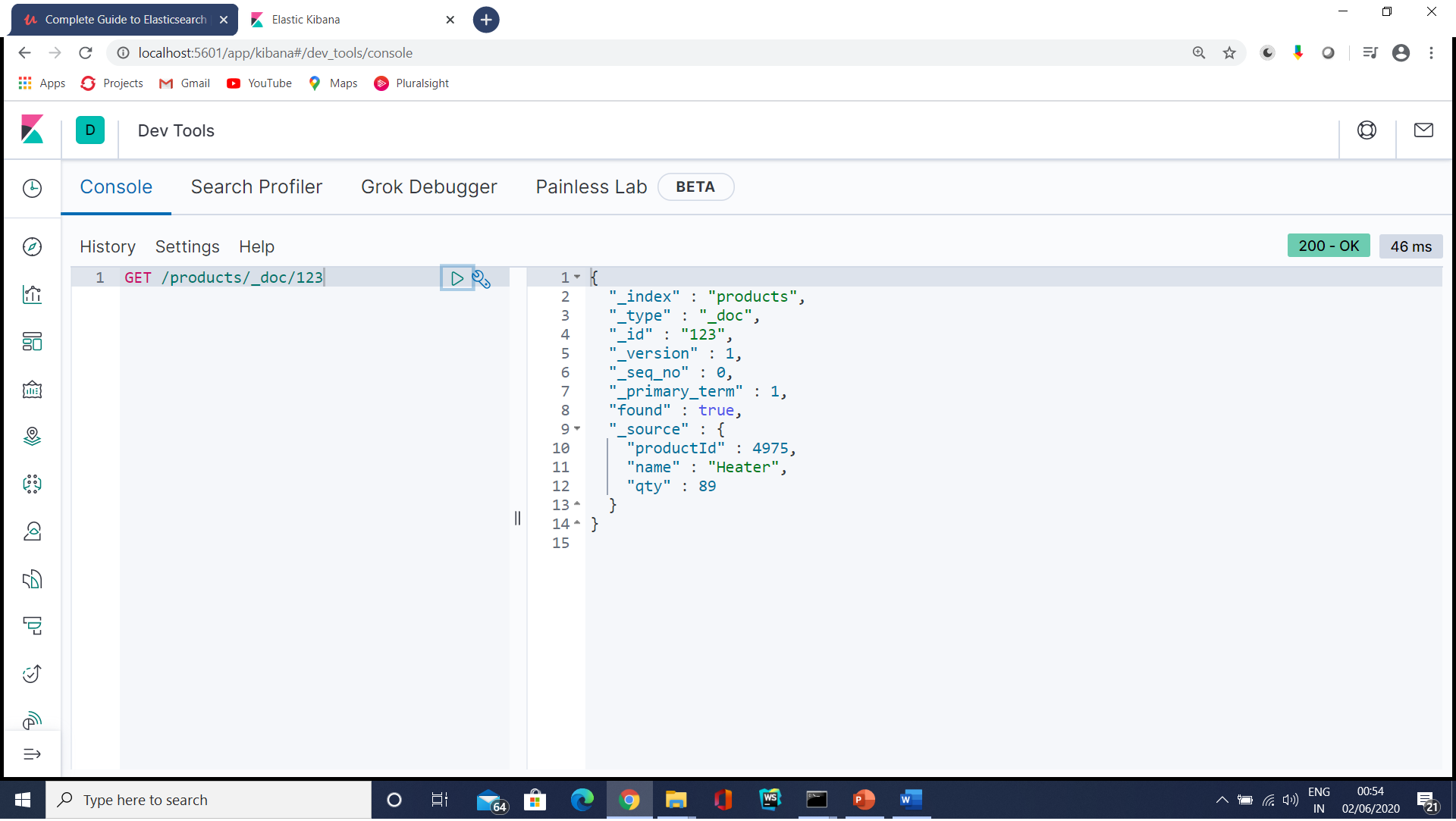
Create Indexing documents



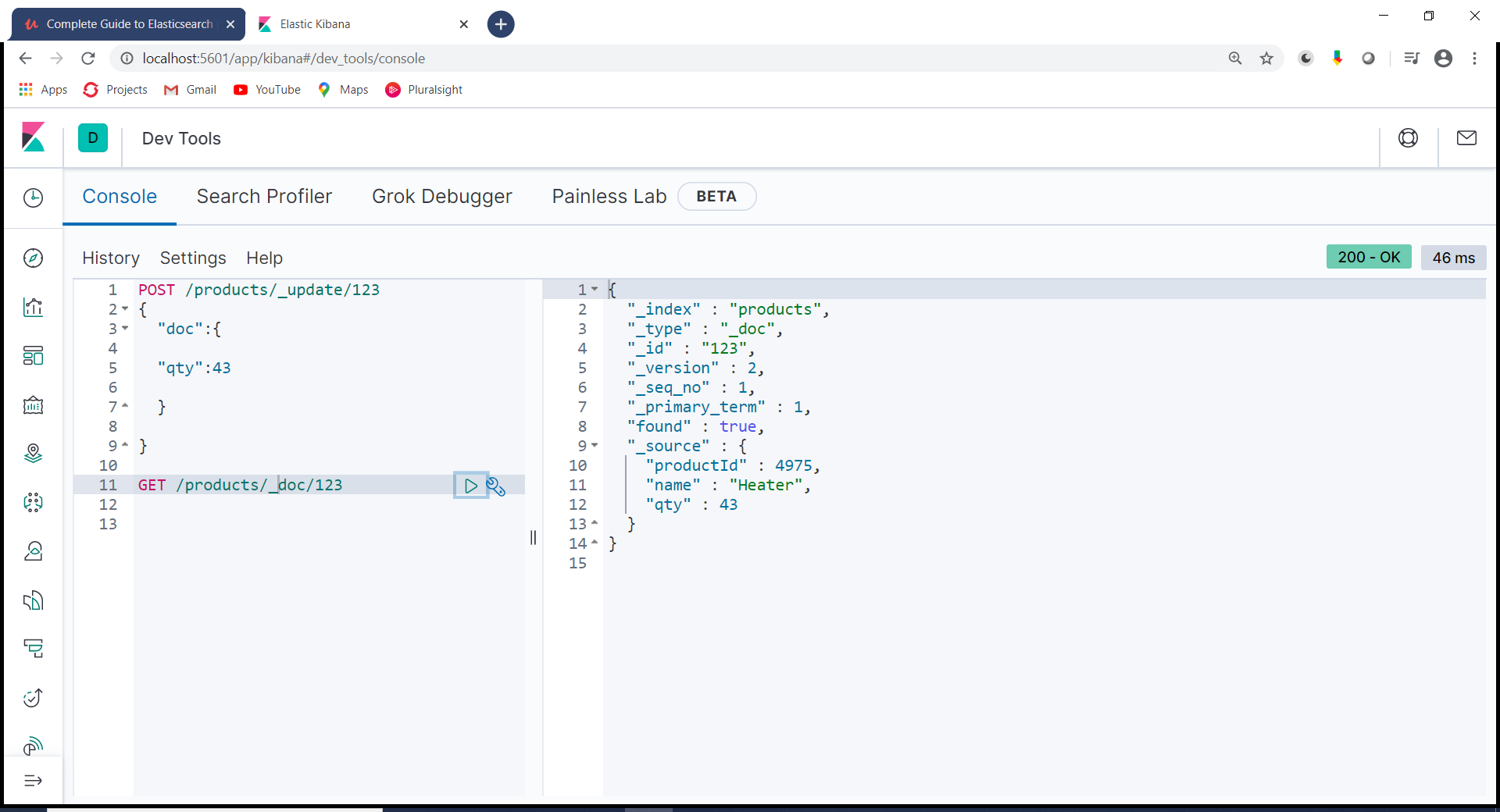
Retrieving Documents



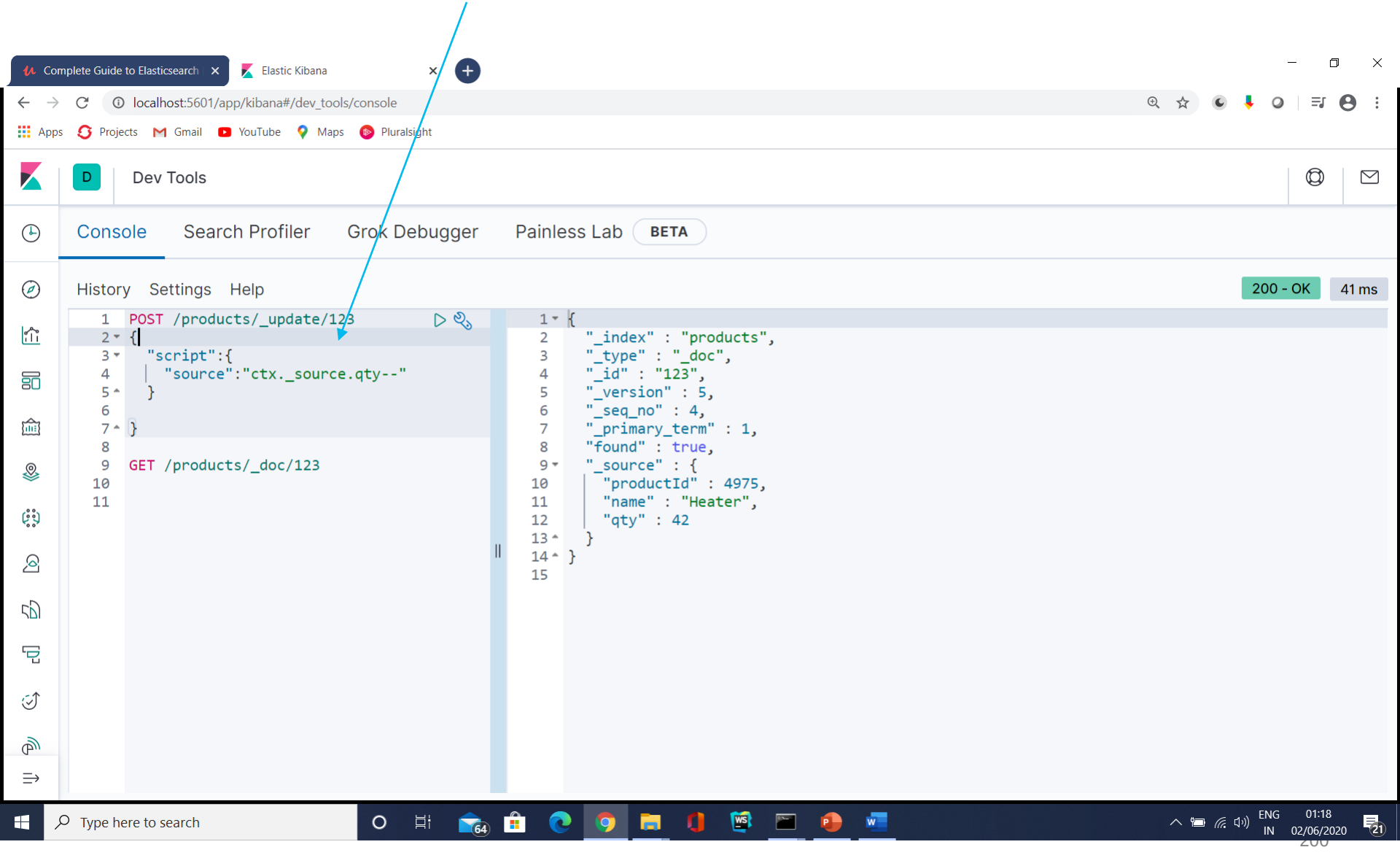
Retrieving Documents



Updating the Document



Scripting the document



Questions

Module Summary

- Spring Integration Framework.
- Message, Channel and Adapter
- Understood the different Component Integration
- Understood the Event-Driven Architecture