**Elasticsearch and kibana**

**Introduction:**

* Data is stored as Documents which is a unit of information
* Documents refer to rows in RDBMS
* Elasticsearch is written in java, built on Apache Lucene, Easy to use & highly scalable. Search’s are lightning fast in millions of documents

**Elastic Stack**:

Elastic stack is a combination of different tools namely

* **Kibana** – An analytics and visualisation platform
* **Logstash –** Used to process logs from applications & send then to Elastic Search(A data processing Pipeline
* **X-pack** – Adds additional features to elasticsearch like security, monitoring, altering, Reporting, Machine Learning, Graph, Elastic SQL.
* **Beats –** Collets and process different types of data

**Installation:**

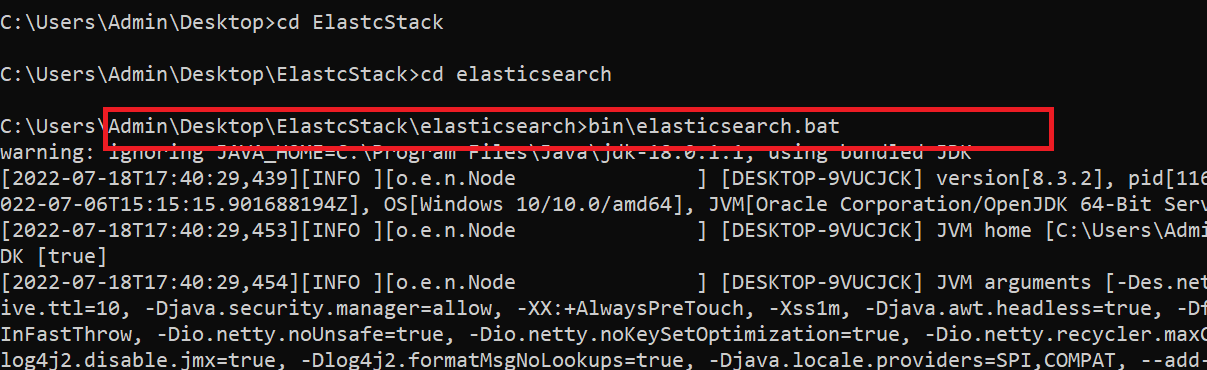
Download the Elasticsearch and kibana archives from the following links:

<https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.3.2-windows-x86_64.zip>

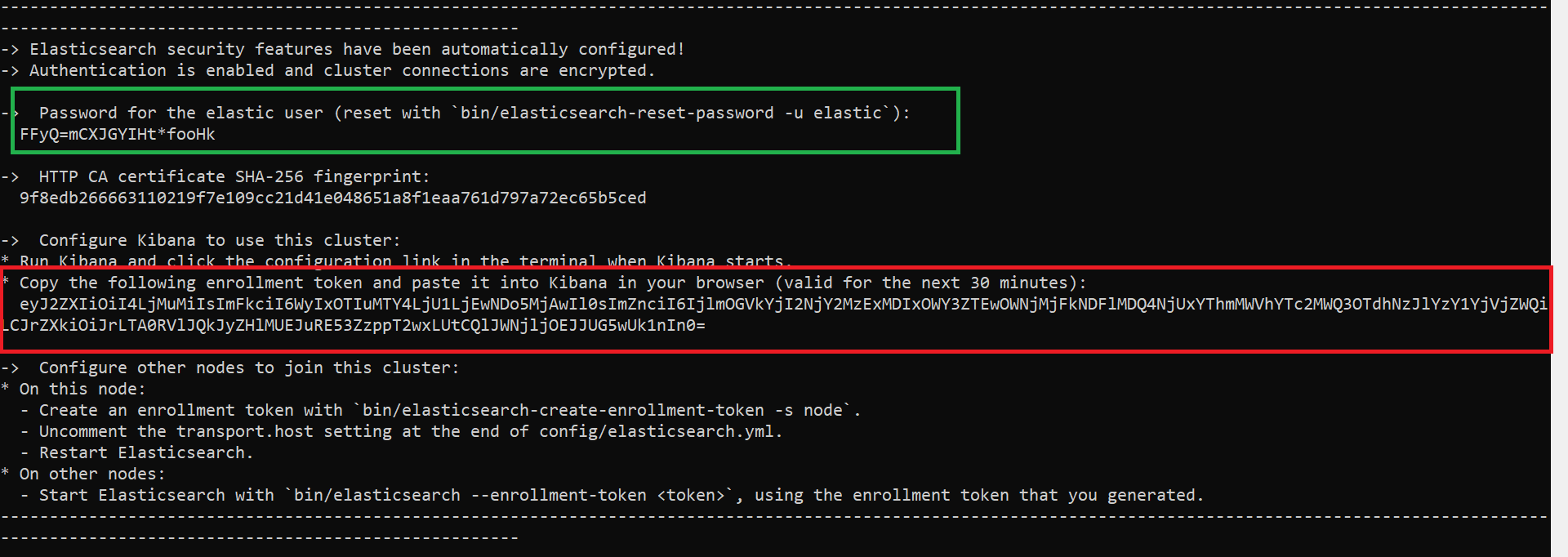
[https://artifacts.elastic.co/downloads/kibana/kibana-8.3.2-windows-x86\_64.zip](https://artifacts.elastic.co/downloads/kibana/kibana-8.3.2-windows-x86_64.zip%20)

Extract the archives using an Zip extractor

Go to the file location and open the file **elasticsearch.bat** in **bin** folder using command prompt.



There you will find username & password and an enrolment key for kibana



The default user will be **elastic.**

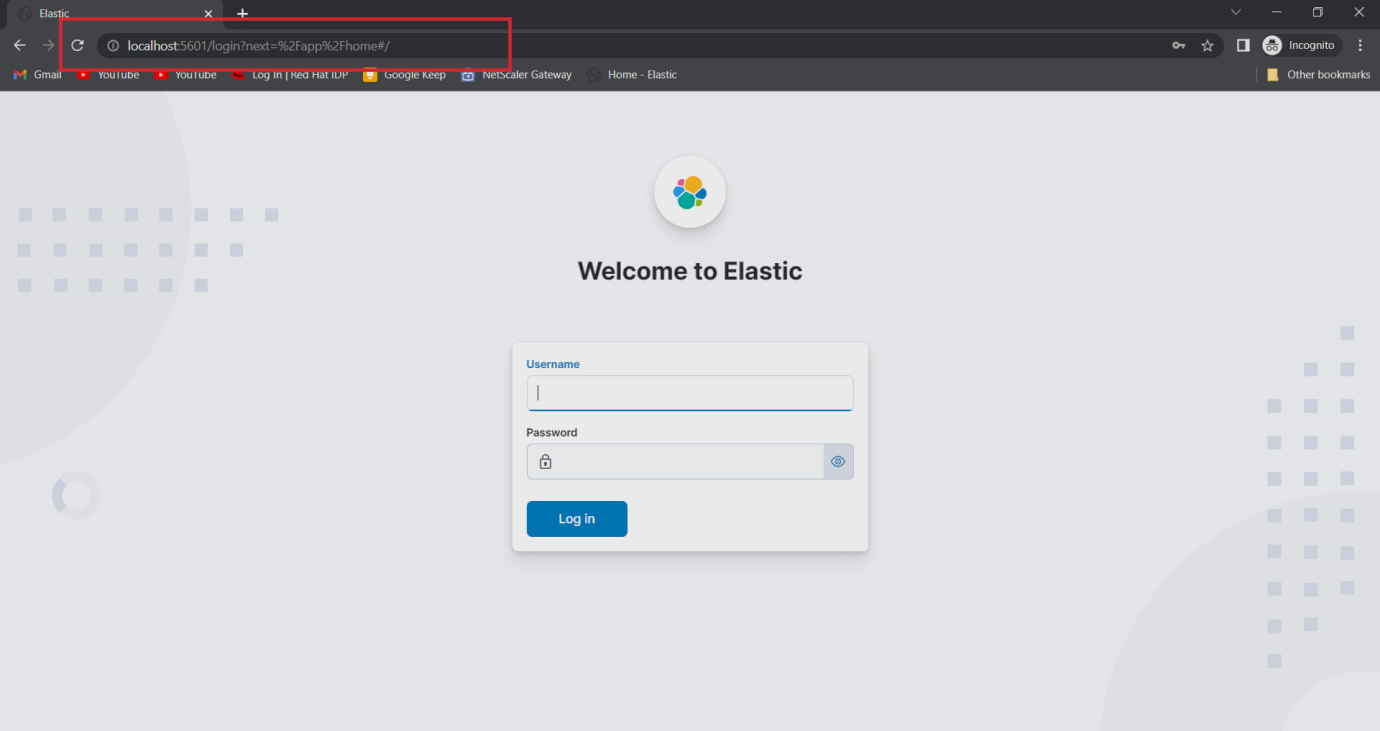
If you want to reset the password using command

bin/elasticsearch-reset-password -u elastic

To setup kibana open kibana.bat file in bin folder in kibana home file.

Follow the link generated there after opening the file It will direct you to the login page and you should enter the encryption key and enter the login credentials and login.

Login page looks like this.



**Architecture:**

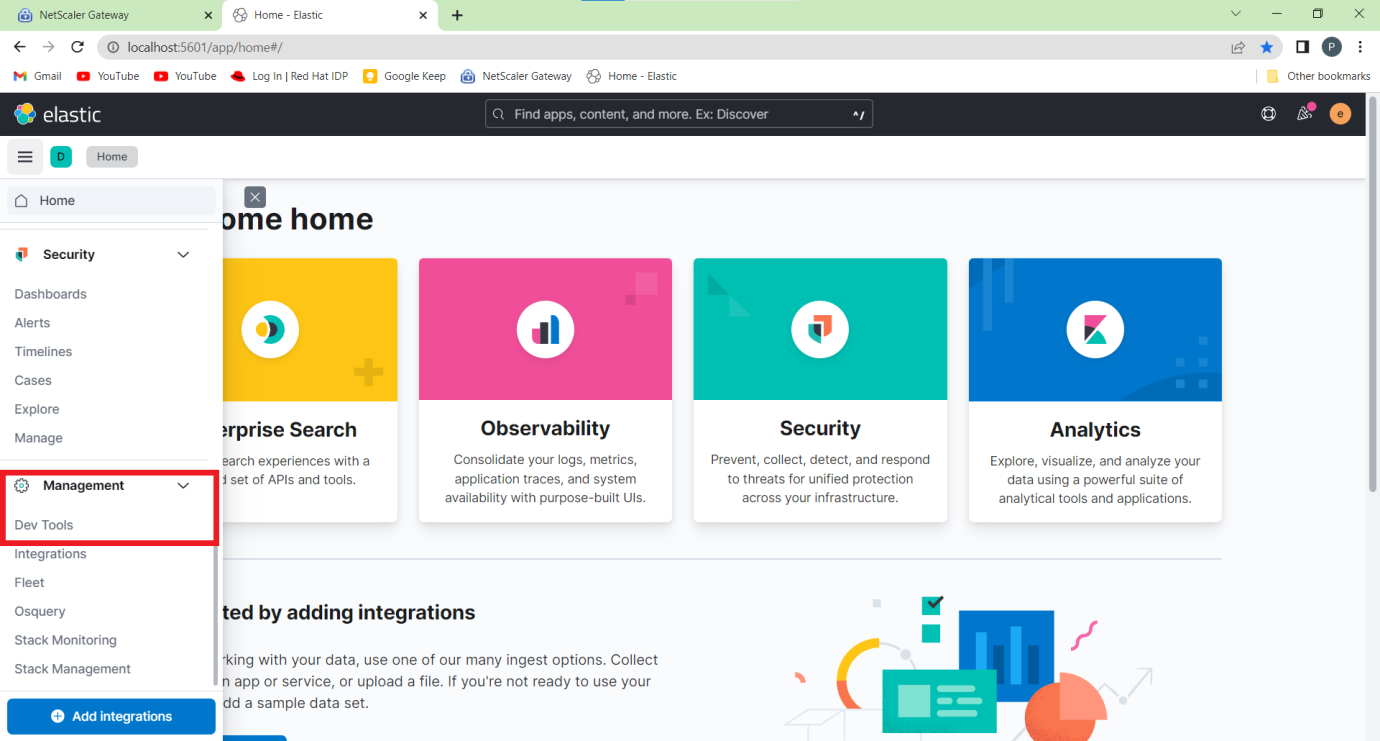
Node: A node refers to an instance of Elasticsearch. we can run any number of nodes in the machine

Cluster: A cluster is a collection of related nodes that together contain all of our data. clusters are completely independent of each other.

Each unit of data is document are JSON objects containing whatever data you desire.

Every document within Elasticsearch is stored within an index. Documents are grouped together with indices.

Now go to devtools in management

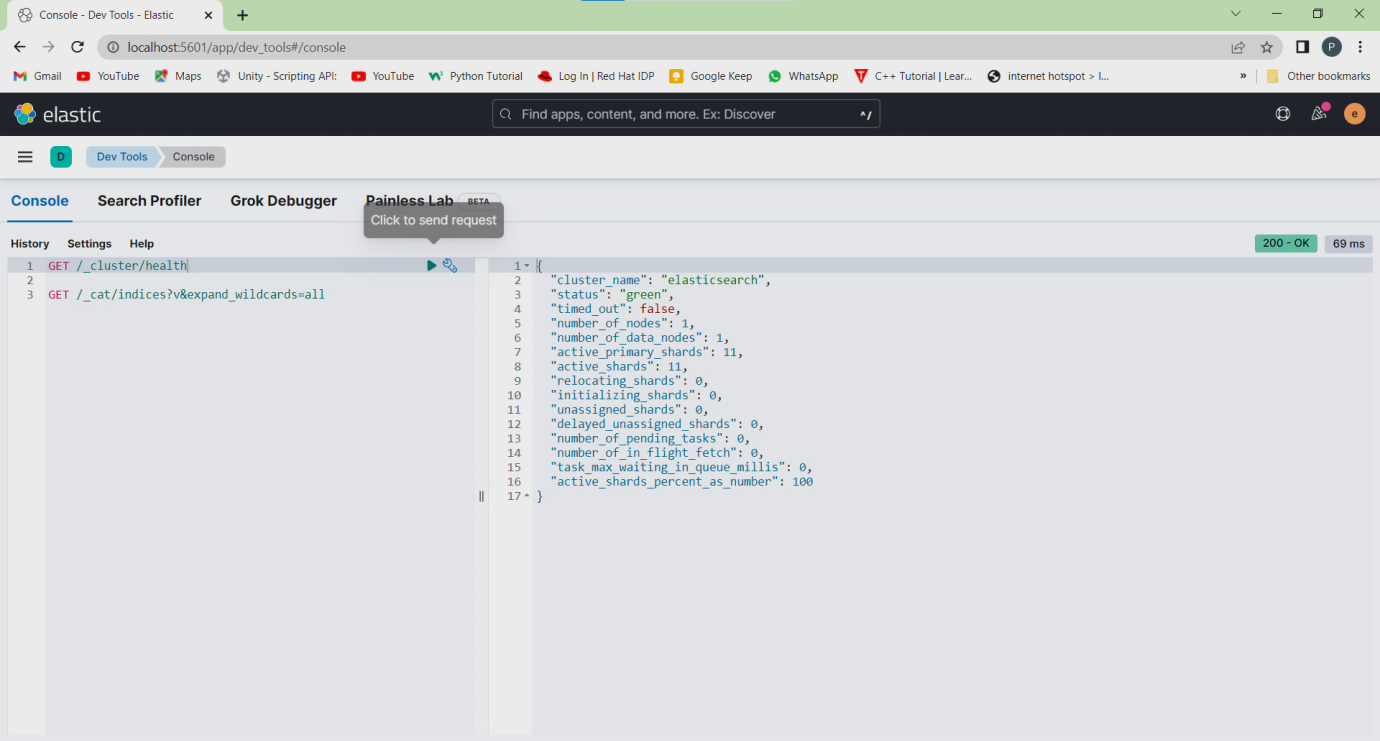


You can check your cluster health in the cluster API

By using HTTP verb GET

Command -> **GET /\_cluster/health**

API starts with an underscore(\_)



It shows all about the cluster and its status. The status green specifies that the cluster is healthy.

Similarly we have CAT API a human readable format

**GET /\_cat/nodes?v**

The above command shows about nodes information like IP address, name as well as some performance measures.

**GET /\_cat/indices?v**

Shows present indices at starting it shows empty when no indices are created.

**GET /\_cat/indices?v&expand\_wildcards=all**

This shows system indices which are hidden by default.

**SHARDING AND SCALABILITY :**

Sharding is a way to divide indices into smaller pieces each referred as shard and it is done at index level. Main purpose of sharding is to horizontally scale the data volume.

In each index contains a single shard by default.

Increase the number of shards with Split API.

Reduce the number of shards with the shrink API.

**REPLICATION :**

Elasticsearch supports replication of shards.

Primary shard and replica shard both called as replication group.

Replication is configured at the index level.

\*\* Replica shards are never in the same node.

Elasticsearch supports Snapshots but replication works on live data.

NOW create an index using the command

**PUT /pages**

Default 1 Primary shard and & one replica shard is created. But the primary shard will be only in working. Replica shard will be assigned to a new node when it is created.

P -> represents primary shards

R -> represents replica shards

**Creating Multiple nodes** :

Again extract the archive two times to create two new nodes

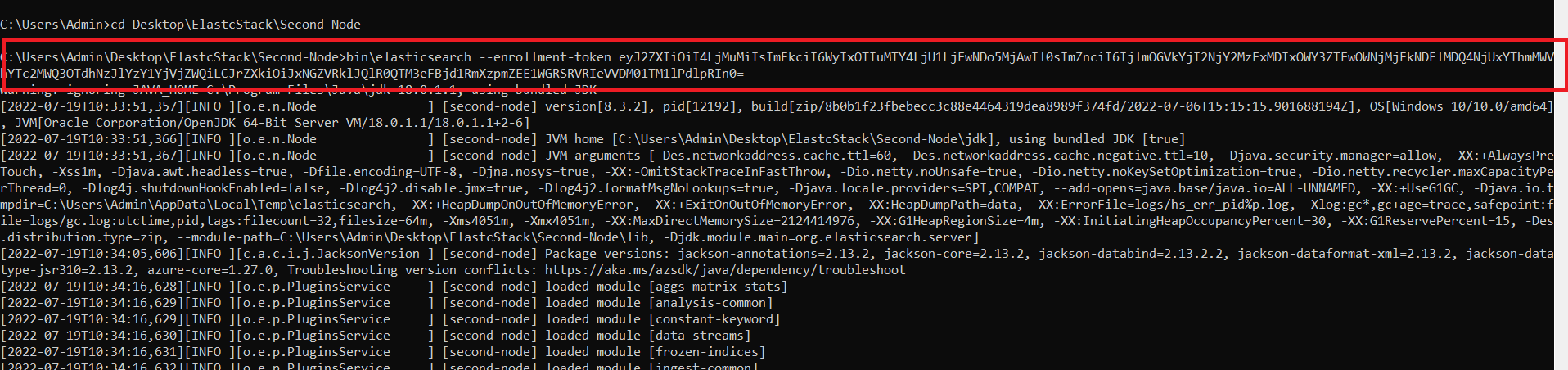
You can change the name of the node in **/config/elaticsearch.yml** file

We need an enrolment token to link it go to the present working directory and type

**bin/elastisearch-create-enrollment-token –scope node**

Now go to home directory and enter this command

**bin/elasticsearch –enrollment-token (token here)**

****

After this when third node is created the elastic search divides the replica shards evenly.

Even the one nose is shutdown the shards are shared evenly.

19/07/2022

**NODE ROLES**

**Master Eligible:**

node.master

The Node is elected as the master node ,and it is responsible for creating & deleting indices among them.

Master is elected by voting process. It is selected in such a way that the master node is stable.

**Data:**

node.data

Enables to store data, storing performance queries, search queries.

**Ingest:**

node.ingest

Enables a node to run ingest pipelines. A simplified way of writing Logstash, diectly within elasticsearch. This role  is mainly useful for having=g dedicated ingest nodes.

**Machine Learning:**

node.ml -> identifies a node as an ML node. This enables node to run machine learning jobs.

xpack.ml.enabled -> Enables/Disables the ML API for the node.

**Coordination:**

This node is responsible for  processing a request and handling the delegation of work needed in that process.(large clusters) can be used as a load balancer.

**Voting-role:**

Node only participating in voting. Used in large clusters.

**ADDING & DELETING INDICES:**

DELETE /pages

To delete a index named pages

PUT /products

{

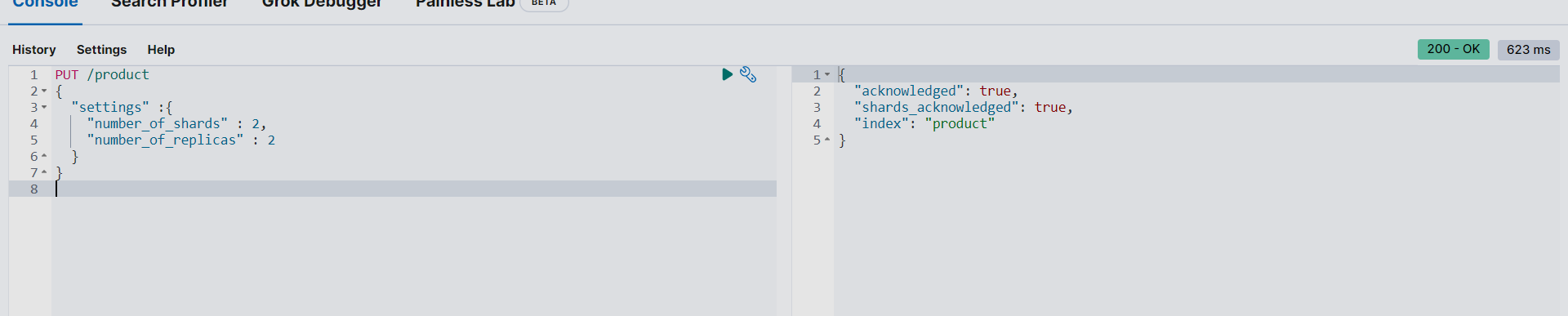
"settings" :{

"number\_of\_shards" : 2,

"number\_of\_replicas" : 2

}

}



The above query creates an index with 2 shards and 2 replica shards.

**INSERTING DOCUMENTS:**

POST /products/\_doc

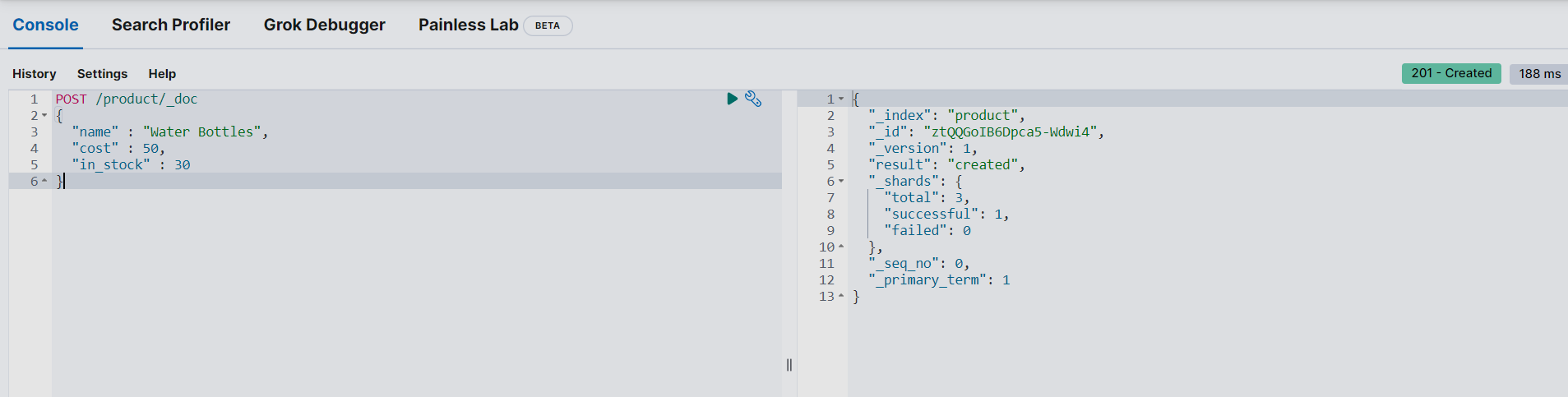
{

“name” : “Water bottles,

“price” : 50,

“in\_stock” : 12

}



This creates a document. And the default id will be assigned to the document.

# here 100 is the id

PUT /products/\_doc/100

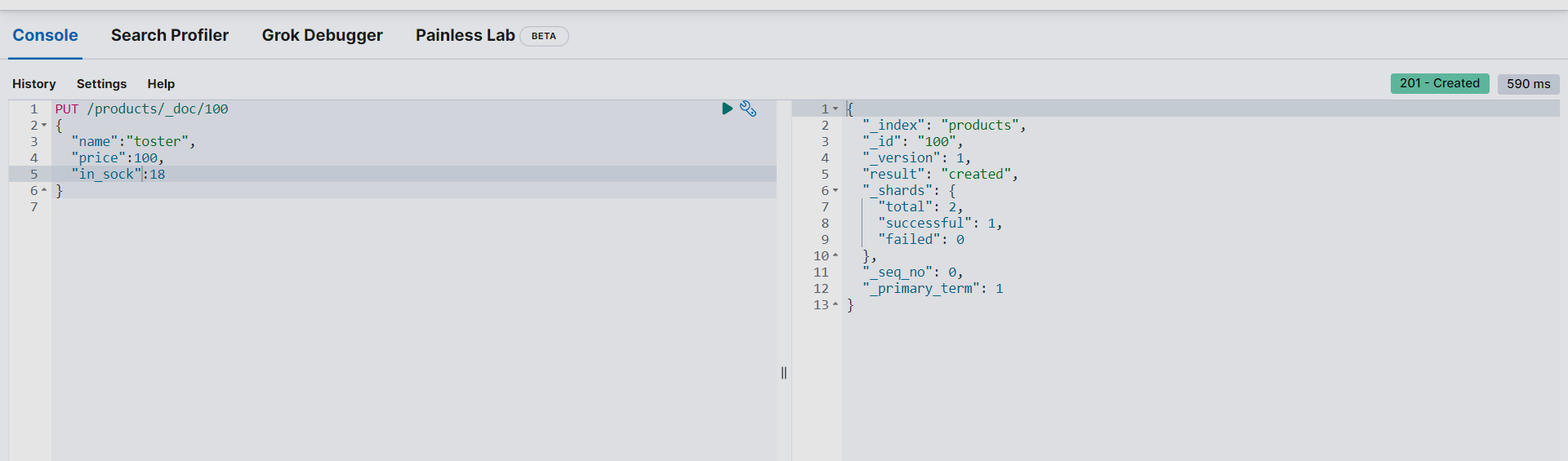
{

“name”:”toster”,

“price”:100,

“in\_sock”:18

}



Retrieving Documents through ID:

GET /products/\_doc/100



**UPDATING DOCUMENTS**:

POST /products/\_update/100

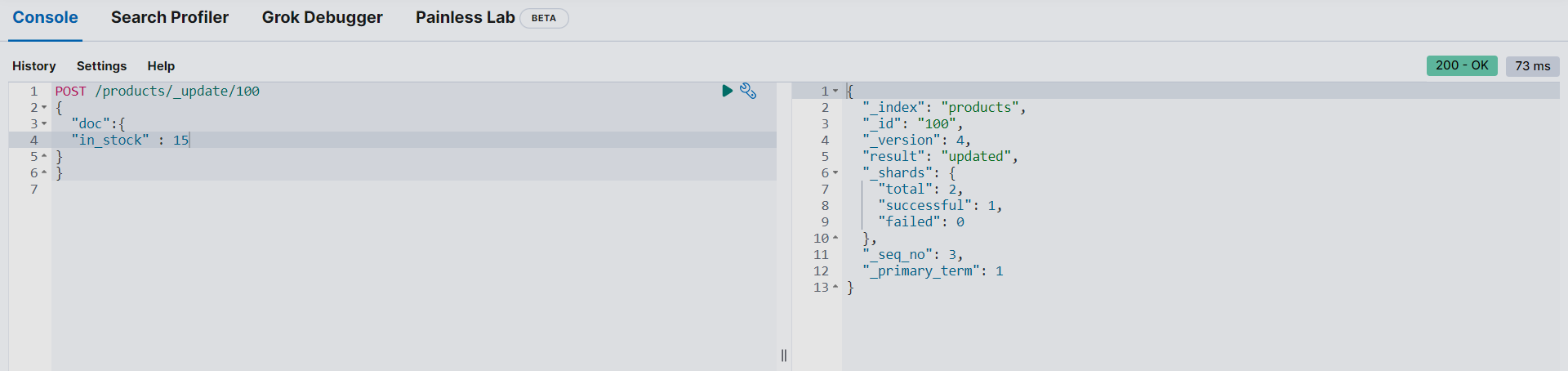
{

“doc”:{

“in\_stock” : 15

}

}



In update we can add more fields also

POST /products/\_update/100

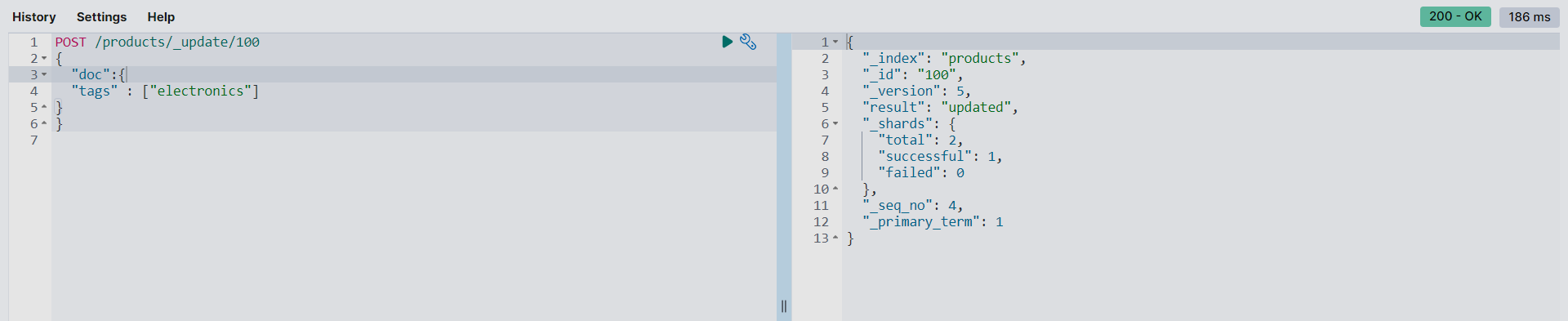
{

“doc”:{

“tags” : [“electronics”]

}

}



**WRITING DATA IN ELASTIC SEARCH**:

Write operations are sent to primary shards

The primary shards forwards the operation to replica shards.

Primary numbers and sequence numbers are useful to recover from failures.

Global and local checkpoints speedup the recovery process.

Primary terms and sequence numbers are available within responses.

**20/07/2022**

**DOCUMENT VERSIONING:**

Elasticsearch only stores the most recent version of the document.

Elasticsearch stores an **\_version** metadata filed with every document.

* + The value is an integer.
  + It is incremented by one when modifying a document.
  + When a document is deleted if we assign new document within 60 sec then \_version will be incremented or else it will be reset.

Configured with the index.gc\_deletes setting

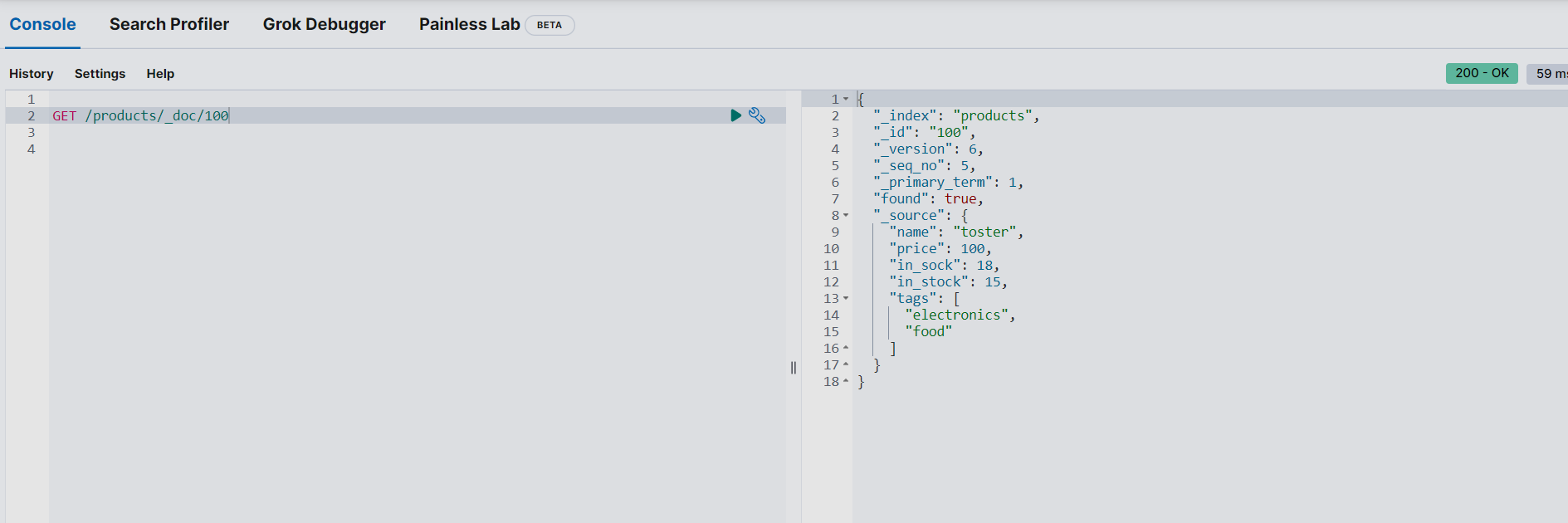
This is default type of versioning termed as internal versioning.

There is also external type of versioning

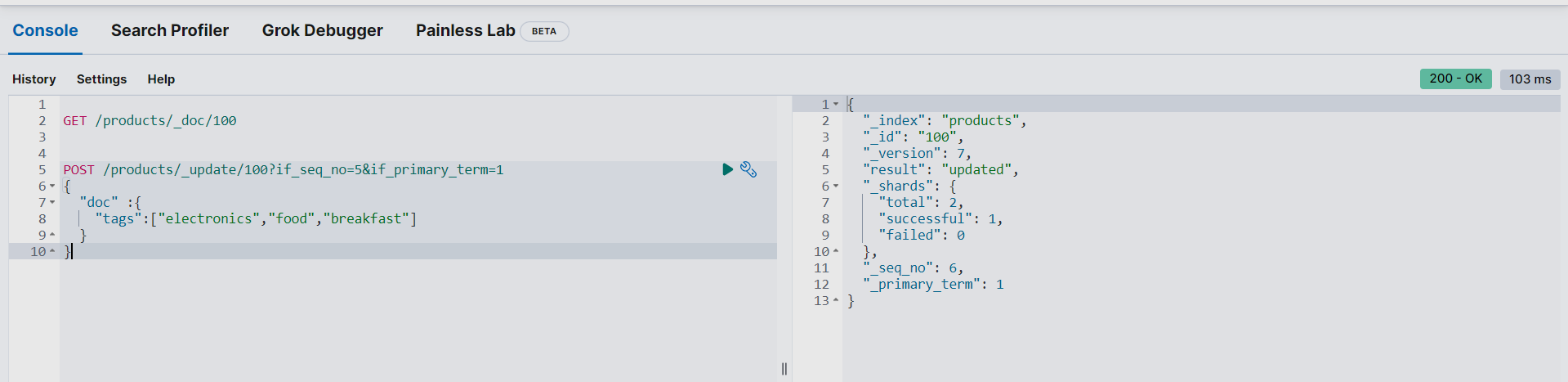
* Useful when versions is maintained outside the elasticsearch
* When documents are also stored in RDBMS
* To use external versioning you specify both version(number) and version type

**OPTIMISTIC CONCURRENCY CONTROL:**

Sending write request concurrently may overwrite changes made by other concurrent process.

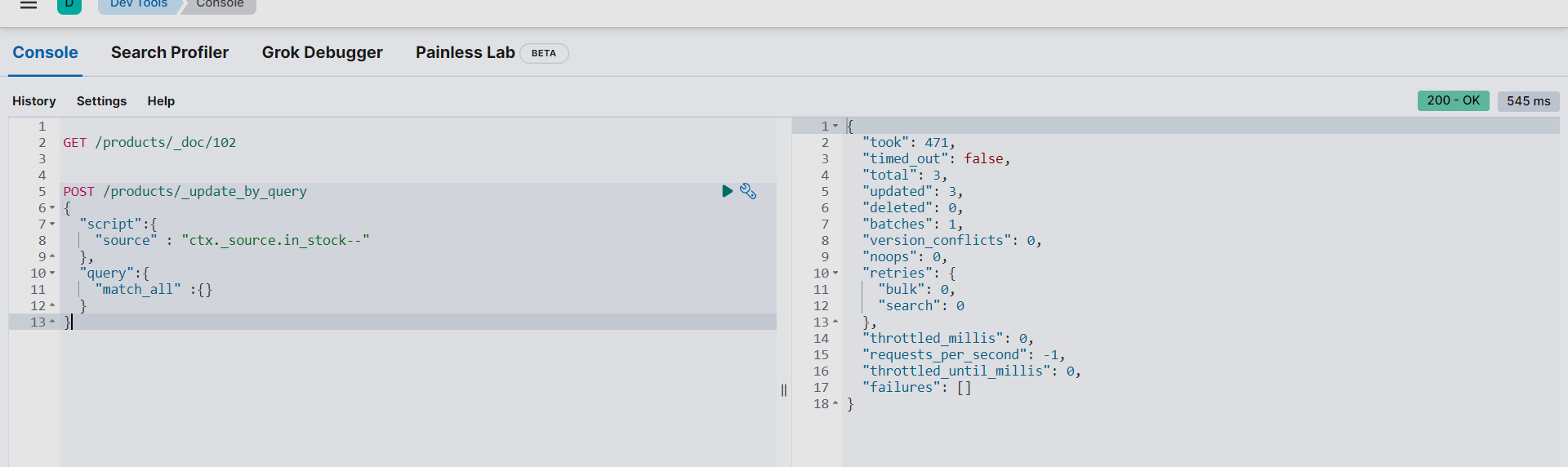


Using the \_seq\_num and \_primary\_term and mention them if only these values matches the data will be updated.

Therefore this technique will prevent concurrency issue.

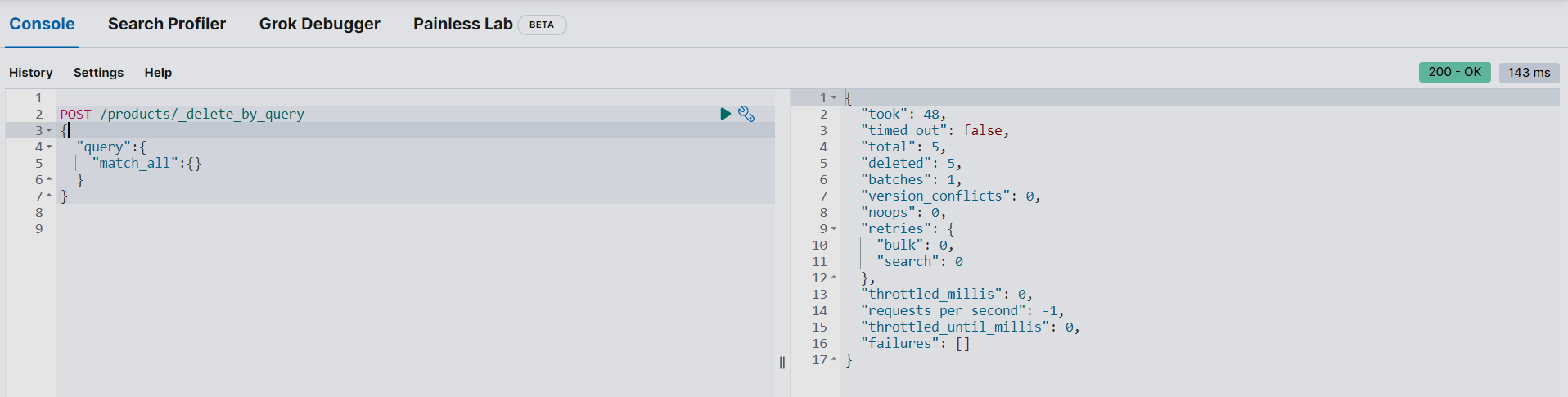
**UPDATING MULTIPLE DOCUMENTS BY SINGLE QUERY:**

To do this we need to use \_update\_by\_query API



**DELETE BY QUERY:**

To do this we need to use \_delete\_by\_query API



**BATCH PROCESSING:**

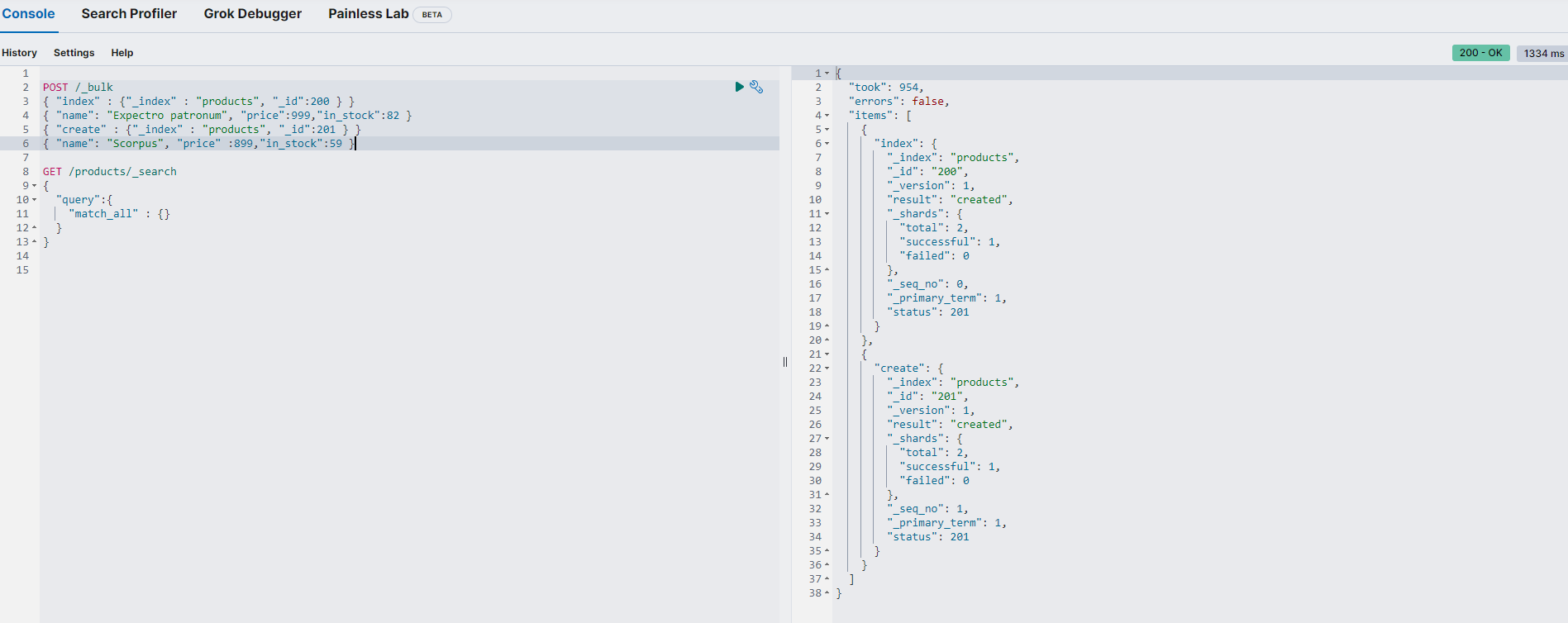
Indexing, updating, deleting, are done by single query on many documents.

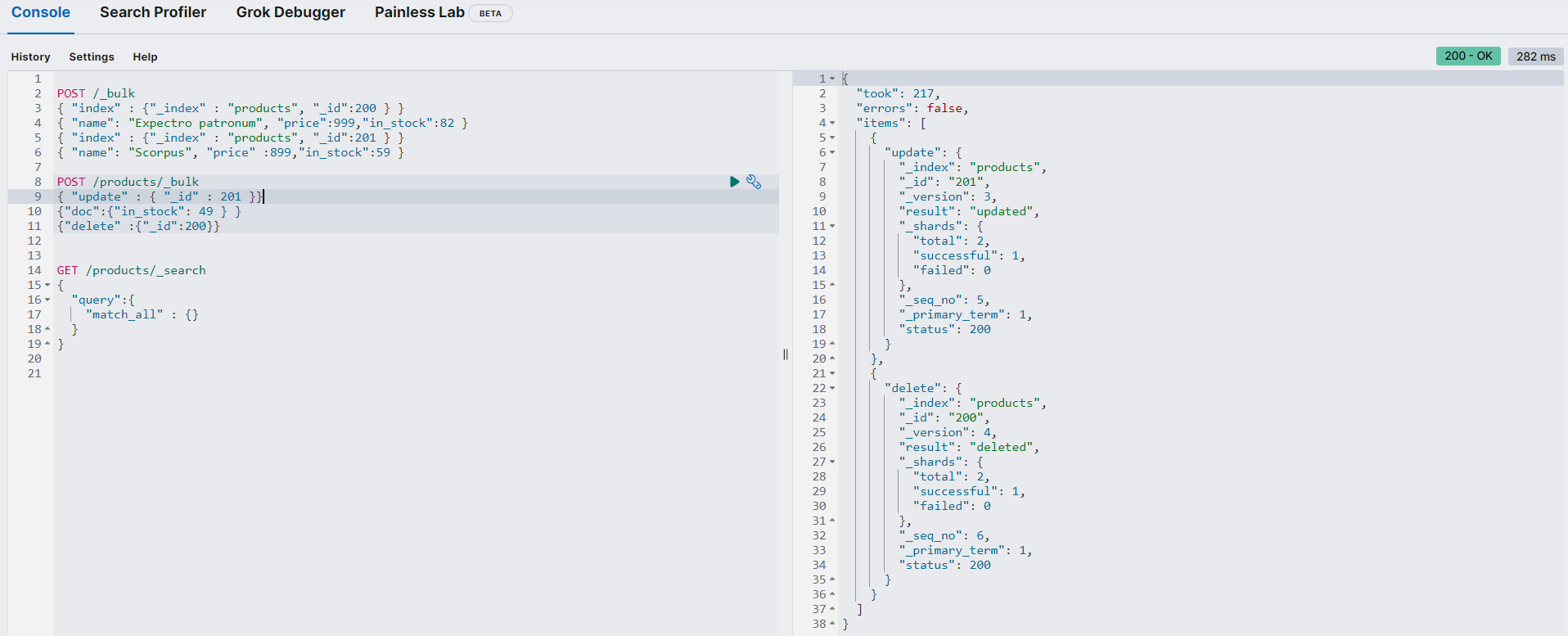
That is done with Bulk API.

Bulk API expects data formatted using the NDJSON specification.

Create documents using **create** and **index** actions

These two are same but create action fails if the document already exists, in index action the document will be replaced with the new one.





To use NDJSON format set the content type as follows

“Content-Type: application/x-ndjson”

In bulk API failed actions will not affect other actions

Added 1000 documents using the bulk API using cURL

