6CS030 WORKSHOP 09

# SECTION 1

1. What is an ELK stack ? Explain it using its architecture.

ELK Stack stands for:

* Elasticsearch (search and analytics engine)
* Logstash (data processing pipeline)
* Kibana (visualization and dashboard tool)

Architecture Flow:

* Logstash: Collects and transforms data from multiple sources.
* Elasticsearch: Stores, indexes, and allows fast search over the data.
* Kibana: Connects to Elasticsearch and visualizes data with dashboards, graphs, and charts.

1. What is Shard in ElasticSearch ?

A shard is:

* A basic unit of storage and search in Elasticsearch.
* Each shard is an independent and self-contained index.
* Elasticsearch splits large datasets into smaller shards for better scalability, performance, and fault tolerance.
* Primary Shards store original data; Replica Shards are copies for backup and high availability.

1. What are the 3 major components of Logstash ?

Input:

* Where data enters Logstash (e.g., logs, metrics, events).
* Examples: Beats, Syslog, Kafka, etc.

Filter:

* Processes and transforms the data.
* Examples: grok (parsing logs), mutate, drop, json.

Output:

* Sends the processed data to another system.
* Typically Elasticsearch, but could also be files, emails, or other outputs.

1. Perform a case study on the possible usage of ELK stack by anyone of the following companies leveraging big data.

1. Netflix
2. LinkedIn
3. Medium

Netflix is one of the heaviest users of the ELK stack. Because of

Netflix uses ELK:

* Massive Scale: Thousands of microservices, millions of users worldwide.
* Real-time Monitoring: Need to monitor millions of logs/events in real-time.
* Operational Intelligence: Detect service failures, latency issues, or bad deployments instantly.

How they use it:

* Log Collection: Logstash collects logs from hundreds of services.
* Storage and Search: Elasticsearch indexes logs for instant search and analysis.
* Visualization: Kibana dashboards visualize service health, system metrics, error rates, etc.
* Alerting: Custom alerts for anomalies detected via search queries.

Benefits:

* Faster root cause analysis.
* Proactive monitoring reduces downtime.
* Easier debugging of distributed services.

1. Explore Fuzzy Search in ELK.

Fuzzy Search is:

* A search technique that finds matches even when words are misspelled or similar.
* Useful for human errors, typos, or partial inputs.

How it works in Elasticsearch:

* Based on the Levenshtein edit distance (number of edits needed to change one word into another).

Applications:

* Autocomplete
* Search engines
* Error-tolerant queries

# SECTION 2

For this, you need to install the ELK stack in your machine.

Here are some resources you can consider referring to for installation :

👉 [week 09](https://drive.google.com/drive/folders/1SydBgMwH85XpYeX32dA5UwJKqngOB7um)

👉 [ELK Stack on Windows : The Complete Installation Guide](https://adamtheautomator.com/elk-stack-on-windows/)

# 

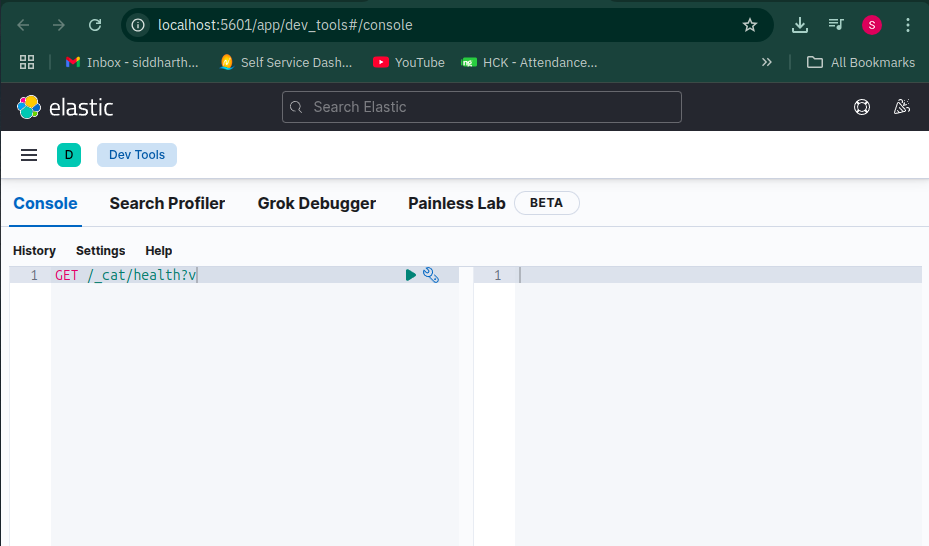
# SECTION 3

As part of building primary familiarity with ELK lets try exploring the provided elastic apis for data manipulation using the Kibana interface.

To start with this, open up Kibana on a web browser and go to :

👉

This should lead you to the Console that will allow you to kick start the task. The GUI looks like the following.

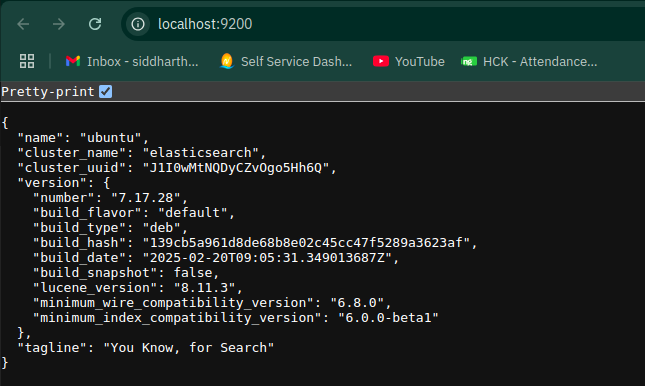


## 

Make sure you have elasticsearch enabled too.

👉 <http://localhost:9200/>

If Elasticsearch is running, you should see a JSON response similar to this:



## Checking Cluster Health

Your task first is to try out a way to make a cluster health check via an Elastic API call.

Consider trying the command

|  |
| --- |
| GET /\_cat/health?v |

For command: GET /\_cat/health?v

#! 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/7.17/security-minimal-setup.html to enable security.

epoch timestamp cluster status node.total node.data shards pri relo init unassign pending\_tasks max\_task\_wait\_time active\_shards\_percent

1745741074 08:04:34 elasticsearch green 1 1 9 9 0 0 0 0 - 100.0%

For Command: GET /\_cat/health

#! 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/7.17/security-minimal-setup.html to enable security.

1745741207 08:06:47 elasticsearch green 1 1 9 9 0 0 0 0 - 100.0%

The difference between GET /\_cat/health?v and GET /\_cat/health lies mainly in the way the output is presented. When you add the ?v parameter (which stands for "verbose") in GET /\_cat/health?v, Elasticsearch returns the health status of the cluster along with headers, meaning it clearly labels each column such as timestamp, cluster, status, node.total, and so on. This makes the output much more human-readable and easy to interpret. In contrast, when you use GET /\_cat/health without ?v, the output omits the headers and simply shows the raw values. This format is less readable for humans because you have to know the order of the columns beforehand, but it can be useful for automated scripts or tools that already understand the data structure. In both cases, the underlying information about the cluster’s health is the same, but using ?v adds clarity by explicitly naming the fields.

Exercise To do

1. *Observe and write down your understanding of the result. Also, try sending a call request without the ?v, note down the difference if present.*
2. *Make an api call to check the node health.*

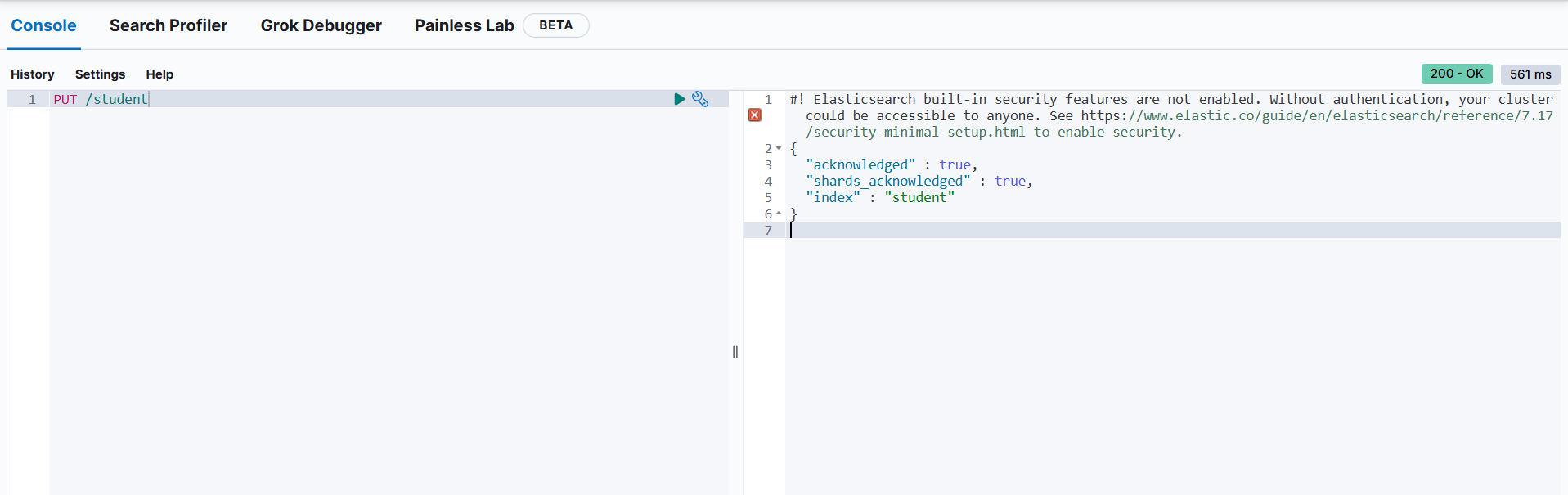
## Creating a Resource

Now, lets try adding in data to our **nosql database** which in current case is ElasticSearch.

In Elasticsearch, a resource is called an **Index.** Let’s create a new one using the below command :

|  |
| --- |
| PUT /resourcename |

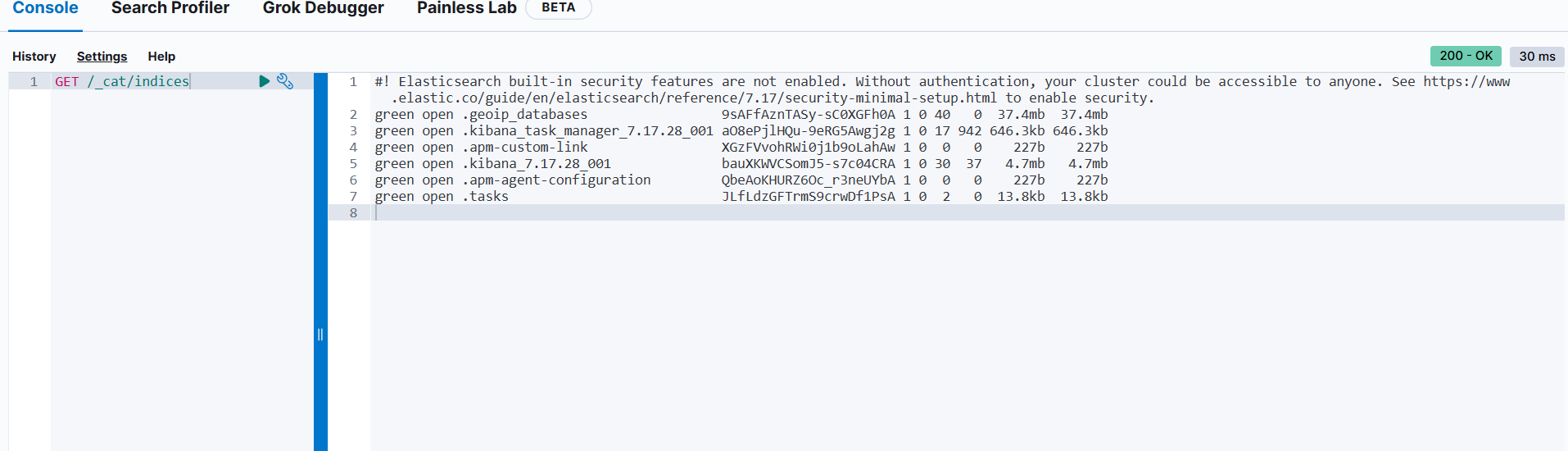
*Replace* ***resourcename*** *with any name you want for your index (e.g.,students).*

**

## Listing Indices

Now, try out the list indices call to observe any changes in the resources using :

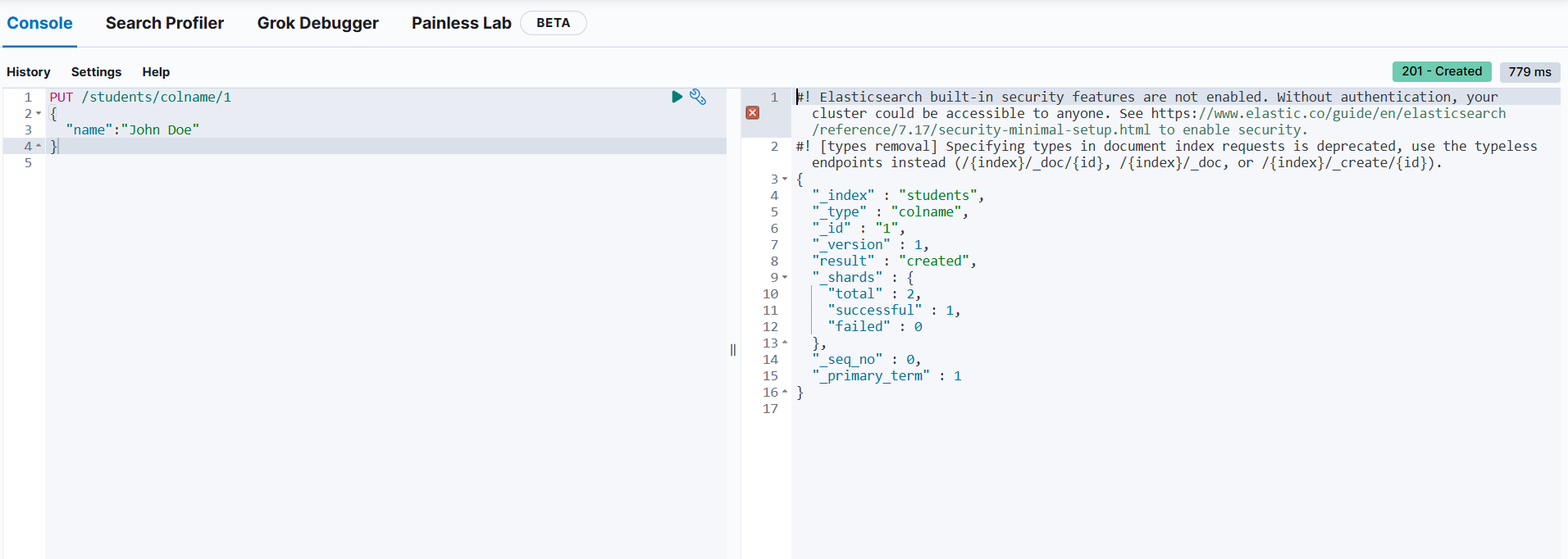
|  |
| --- |
| GET /\_cat/indices |



Next on, let's move ahead with addition of data into the resource we just created,

## Adding Data to a Resource

|  |
| --- |
| PUT /students/colname/1  {  "name":"John Doe"  } |



*Replace :*

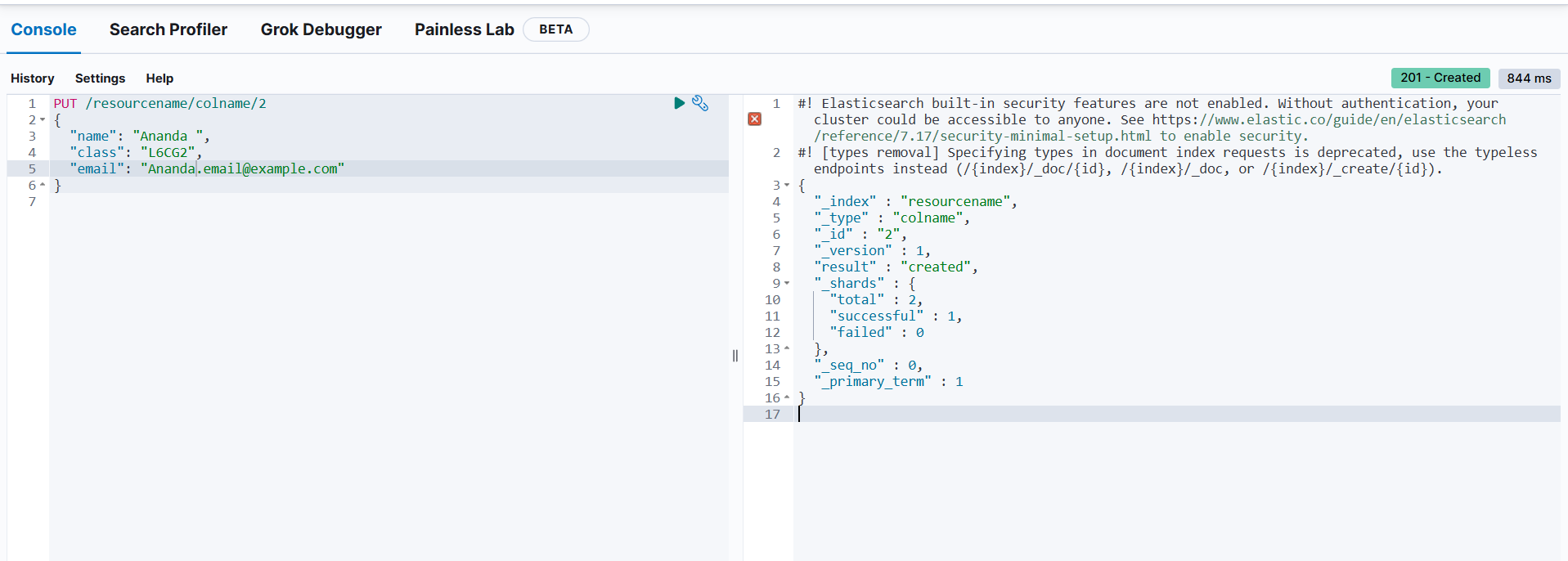
* **resourcename** with your index name (e.g., students)
* **colname** with a type (optional in newer Elasticsearch versions, but keep it for now).
* **1** with the document ID.

Exercise To do

I helped you add John Doe, it's your turn to add in your personal data (consider populating it with your name, class group and email) to the same resource.

|  |
| --- |
| PUT /resourcename/colname/2  {  "name": "Your Name",  "class": "Your Class Group",  "email": "your.email@example.com"  } |

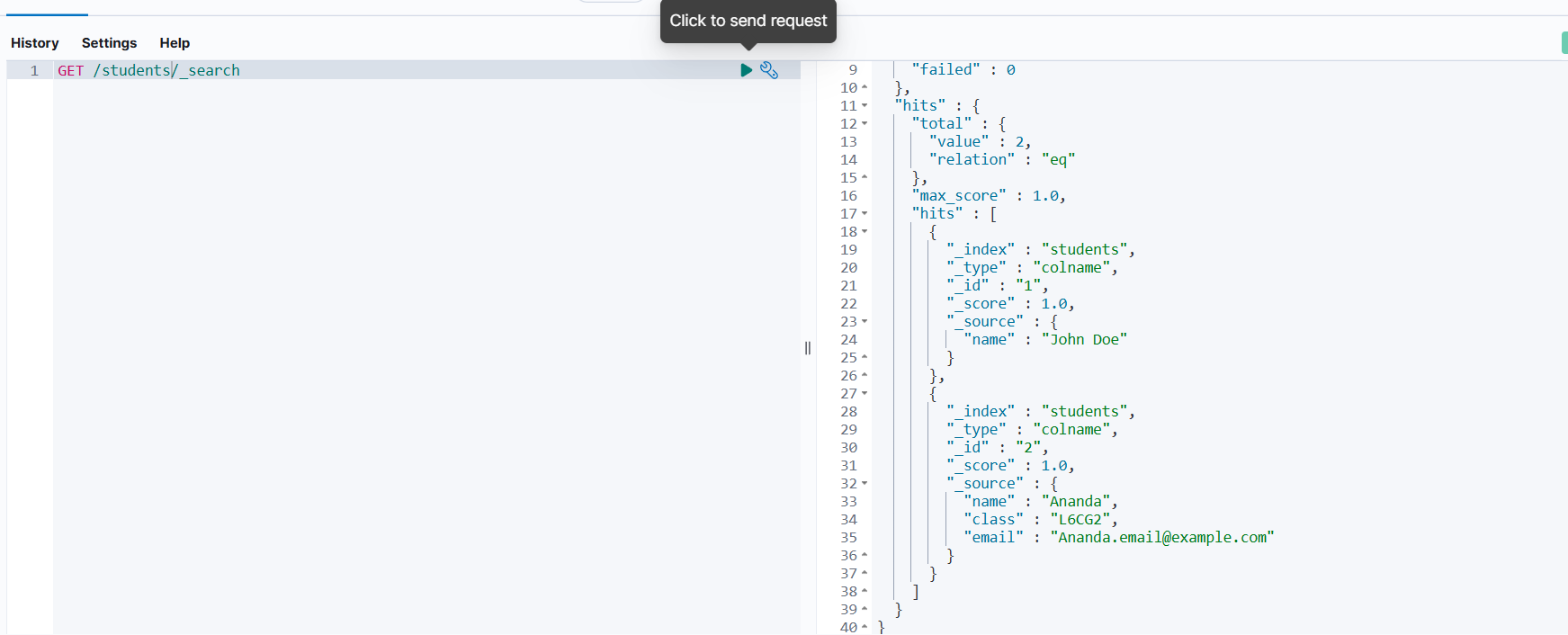
*Don’t forget to replace the* ***placeholder values*** *with* ***your actual details****.*

**

## Fetching the Record

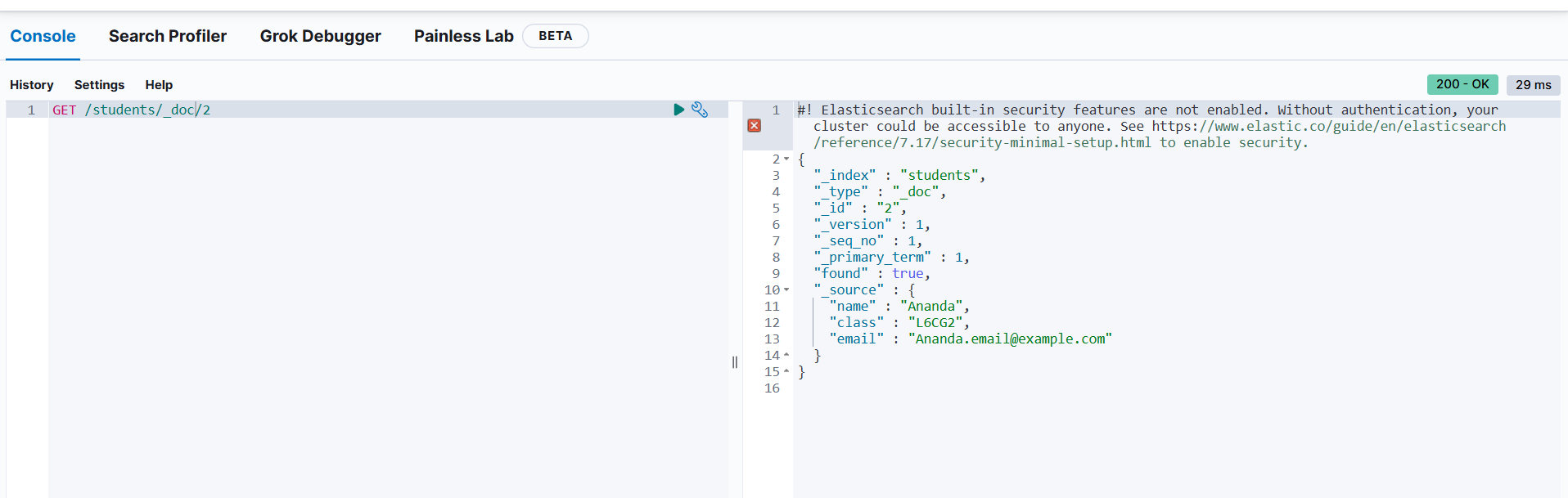
To view the data stored in the index, run a search query:

|  |
| --- |
| GET /resourcename/\_search |



Exercise To do

1. Write an api query call to fetch the second data record (that was your personal details).



1. Write a query to remove the data record of John Doe from the resource.

