Elasticsearch and Kibana

Investigation



Table of Contents

Contents

[1 elasticsearch 3](#_Toc120096985)

[1.1 Introduction: 3](#_Toc120096986)

[1.2 Main Components of Elastic Search: 4](#_Toc120096987)

[1.3 Primary Use Cases: 6](#_Toc120096988)

[1.4 Why Elasticsearch? 6](#_Toc120096989)

[1.5 How it works: 7](#_Toc120096990)

[1.6 When to use elastic search: 7](#_Toc120096991)

[1.7 Elastic Search vs SQL: 8](#_Toc120096992)

[1.8 Can it be used as a primary database? 9](#_Toc120096993)

[1.9 Best Practice: 10](#_Toc120096994)

[1.10 Pricing: 10](#_Toc120096995)

[2 kibana 11](#_Toc120096996)

[2.1 Introduction: 11](#_Toc120096997)

[2.2 What Kibana can do: 12](#_Toc120096998)

[2.3 Kibana pricing: 14](#_Toc120096999)

[2.4 Some examples over a dataset: 17](#_Toc120097000)

[3 installation guide 20](#_Toc120097001)

[3.1 Install Elasticsearch: 20](#_Toc120097002)

[3.2 Install Kibana: 22](#_Toc120097003)

[3.3 Install Logstash: 23](#_Toc120097004)

[4 elasticsearch in angular 24](#_Toc120097005)

[4.1 Searchbase: 24](#_Toc120097006)

[4.2 ReactiveSearch: 24](#_Toc120097007)

[4.3 Building a search UI with angular: 28](#_Toc120097008)

[5 Conclusion 32](#_Toc120097009)

[6 references 33](#_Toc120097010)

# elasticsearch



## Introduction:

* Elasticsearch (ES) is the most common open-source distributed search engine. It's based on Lucene, an information-retrieval library, and provides powerful search and query capabilities.
* It is a distributed search and analytics engine, recently it has become the most popular search engine worldwide.
* Commonly used for log analytics, full-text search, business analytics, and more.
* It is a No SQL database that lets you create indices and perform API requests on it
* It is scalable open source multitenant-capable, full-text, distributed search, and analytics engine
* Uses unstructured or semi-structured data Document Oriented Database

## Main Components of Elastic Search:

1. **Index:**

* It is like a Table in Database
* It stores complex data structure in the form of JSON Document,
* Logical data organization mechanism where each document has a field with key and value pair
* Created using PUT/<index>

1. **Shard:**

* Pieces of data where each piece contains n number of entire document as per the configuration of shard in the index

1. **Replica:**

* Copy of information of the shards

1. **Node:**

* Some collection of shards and replicas, starting an instance of elastic search will initiate a node

1. **Cluster:**

* Collection of nodes

Graphical user interface, text, application

Description automatically generated

Example of creating an index with its configurations

Graphical user interface, website

Description automatically generated

## Primary Use Cases:

* Application Search
* Website search
* Enterprise Search
* Logging and Log Analytics

## Why Elasticsearch?

Products with huge databases are facing issues as very long time in retrieving data because relational database work comparatively slow when it comes to huge data and fetching search results through database queries which leads to poor user experience

Simple REST API HTTP Interface

It uses schema-free JSON document

Can process large data and process them in parallel

Instead of API, it can be used from a tool *-Kibana-* to:

* Visualize data
* Load data into elastic search’s cluster

Supports many languages such as Java, Ruby, JavaScript, PHP, and more

## How it works:

* It can work in two ways:
  + Sending the data in the form of JSON document using API.
  + Using ingestion tools that visualize the data and build interactive dashboards such as *Kibana*, *Logstash, and Amazon Kinesis Firehose.*
* Both ways enable elastic search to automatically stores the document, add a searchable reference to the document in the cluster’s index, and let you search or retrieve data and find the best matches
* It makes a table with words in documents and documents itself and check whether this word is in that document or no
* When it searches for something, it just looks at that table
* When a document has a greater number of words that you are searching for it will have higher relevance (appear at the top)

## When to use elastic search:

* There are 2 main cases of elastic search:
  + 1. Text Search (Search Engine)

When you are searching with a word or part of a word in a text (not selection something like category or something like that)

2. Log Analysis

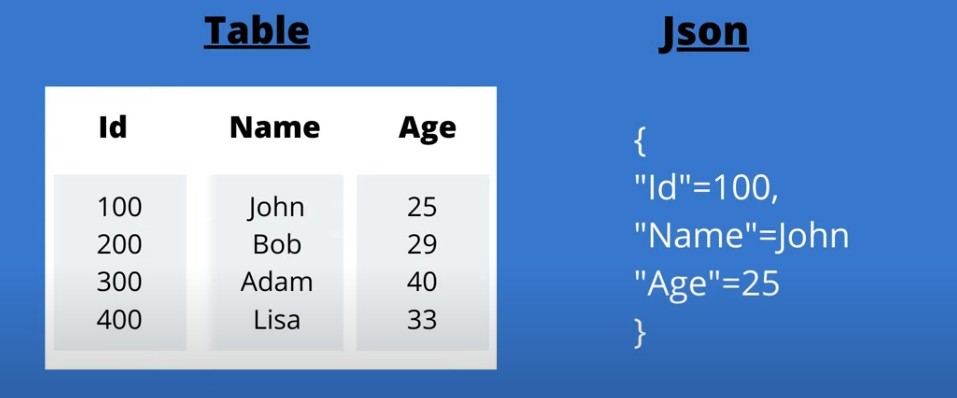
Kibana which is used on top of elastic search will give us the ability to analyze our logs efficiently.

## Elastic Search vs SQL:

Elastic Search is way better in terms of a large amount of data being indexed and searched while SQL is better in terms of transactions and complex queries

ES supports unstructured or semi-structured data while SQL supports structured data

|  |  |
| --- | --- |
| SQL | Elastic Search |
| Column | Field |
| Row | Document |
| Table | Index |
| Schema | Implicit |
| Database | Cluster |



💡

How Data is stored in Table in SQL, and in JSON format in elastic

Search Document:

## Can it be used as a primary database?

Not a good idea due to:

* + - Elasticsearch index sizes need to be pre-determined. Schema/Mapping changes require re-indexing. If the data grows or evolves and cannot be managed with original sharding or mapping strategies, must migrate data into newer indexes. Now, the application must both serve the incoming traffic and do migrations.
    - Performance is going to be a problem if all data queries need to be served out of Elasticsearch especially if volume of data is huge and all data is being indexed without specific attention paid to the query patterns being used.
    - Elasticsearch won’t give you “atomic” blocks of transactions, where you say, “do all these edits, or if there is any error - do no changes at all”. That’s a big deal for financial actions, like buying stuff, where you must update a couple of tables in sync (if you need something like this u need to architect a solution yourself)-won’t support joining also

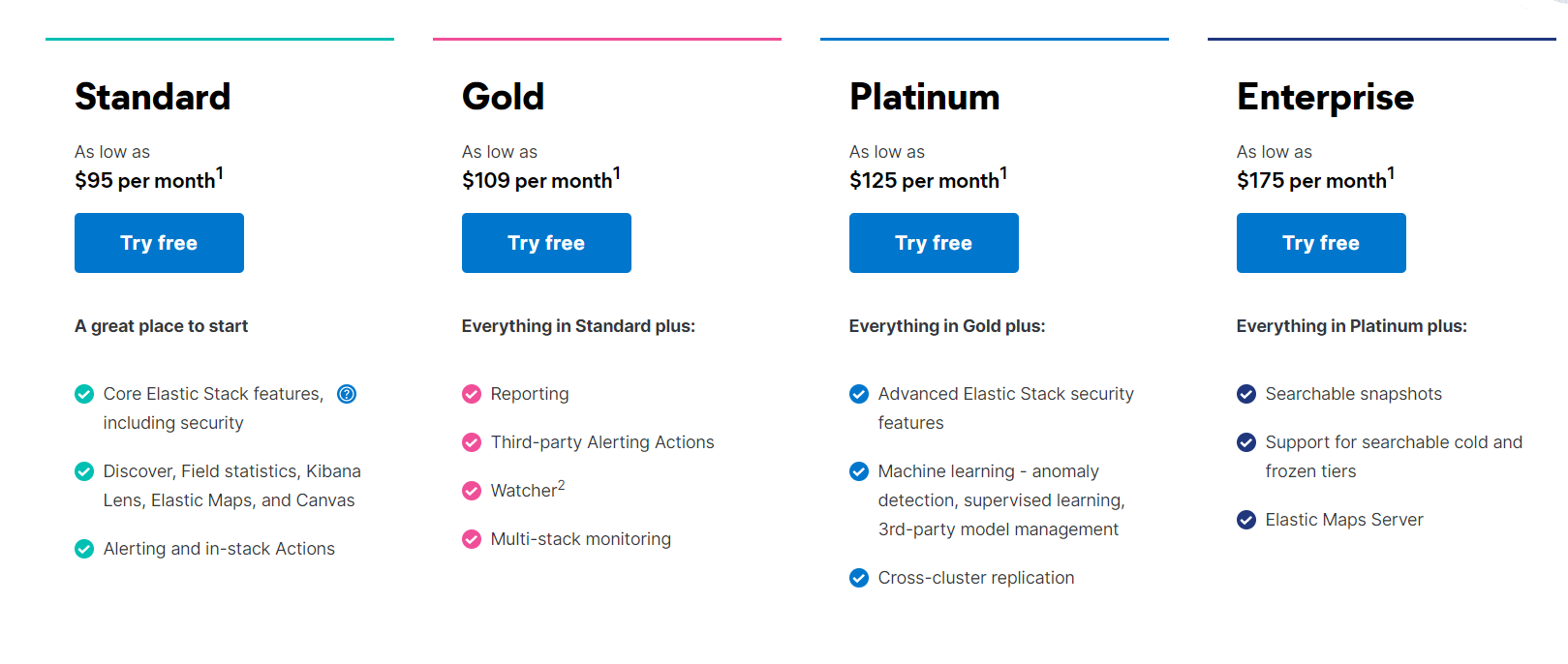
## Best Practice:

The typical use case and the most widely used scenario is that Elasticsearch is a sink in a data pipeline and with another system/database mastering the data. In case of data loss, there is a way to replay data from upstream. So, your system/database would be a single source of truth and you could always flatten the tables and Index that data to ES and for search related to queries, you query ES and use system/database as is for usual application-DB transactions

## Pricing:

There are 2 options:

* Managed hosting from elastic.co, they charge you according to several variables. You can find the pricing here: <https://www.elastic.co/cloud/elasticsearch-service/pricing>



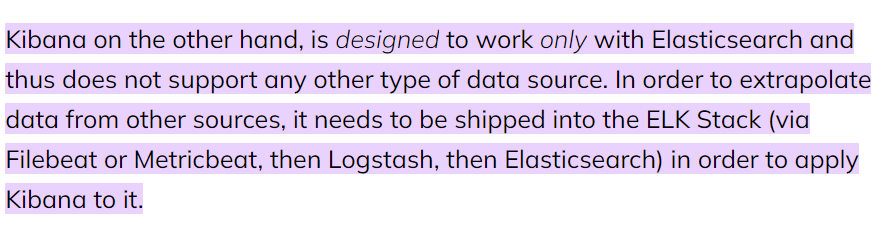
* Use the open-source version, stand up your own servers and manage your own deployment, the code is at no cost and can be found here: <https://github.com/elastic/elasticsearch>

# kibana

## Introduction:

* Kibana is a free and open frontend application that sits on top of the Elastic Stack, providing search and data visualization capabilities for data indexed in Elasticsearch.





## What Kibana can do:

Graphical user interface, application

Description automatically generated



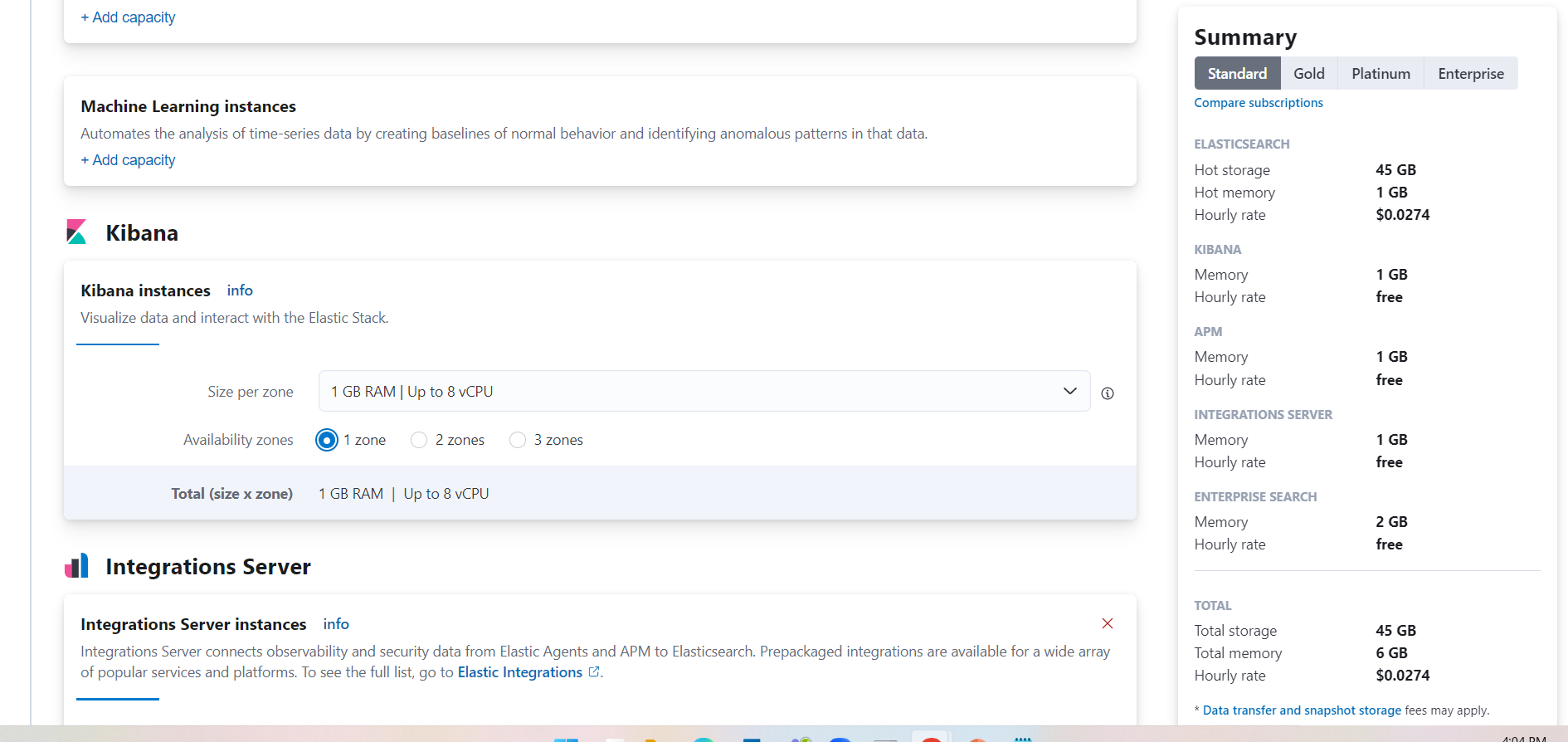


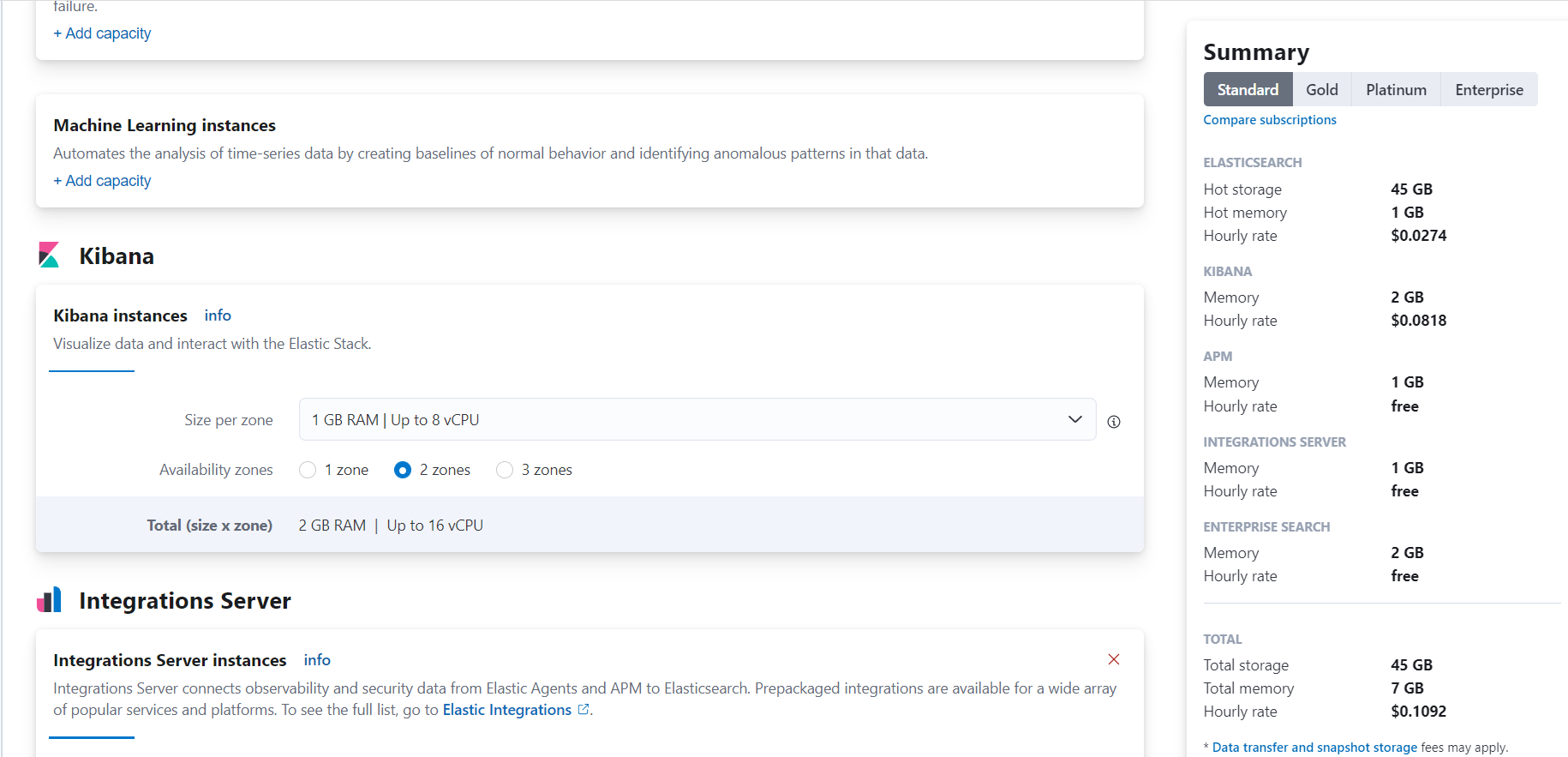
Graphical user interface, application, website

Description automatically generated

## Kibana pricing:

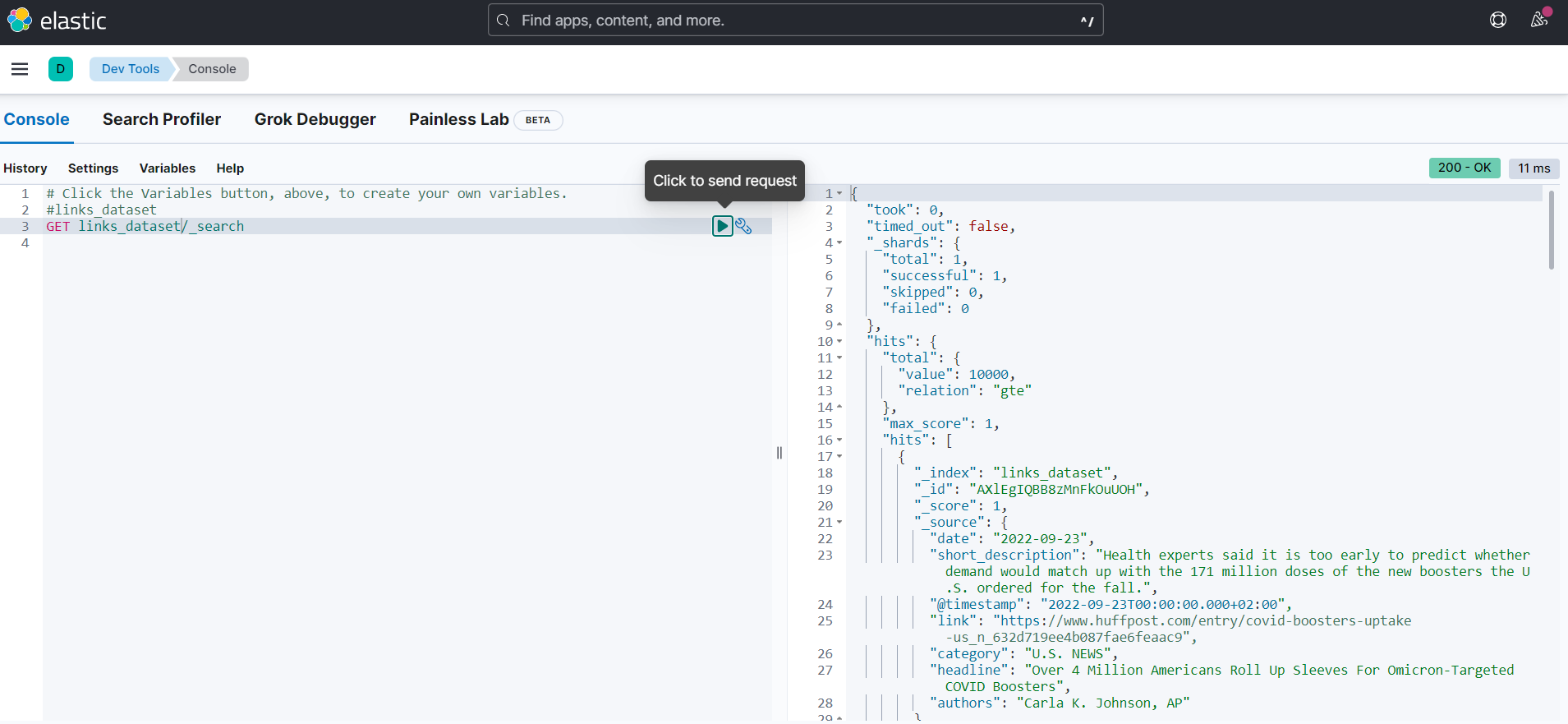
As for Kibana pricing it is included in Elasticsearch pricing calculator



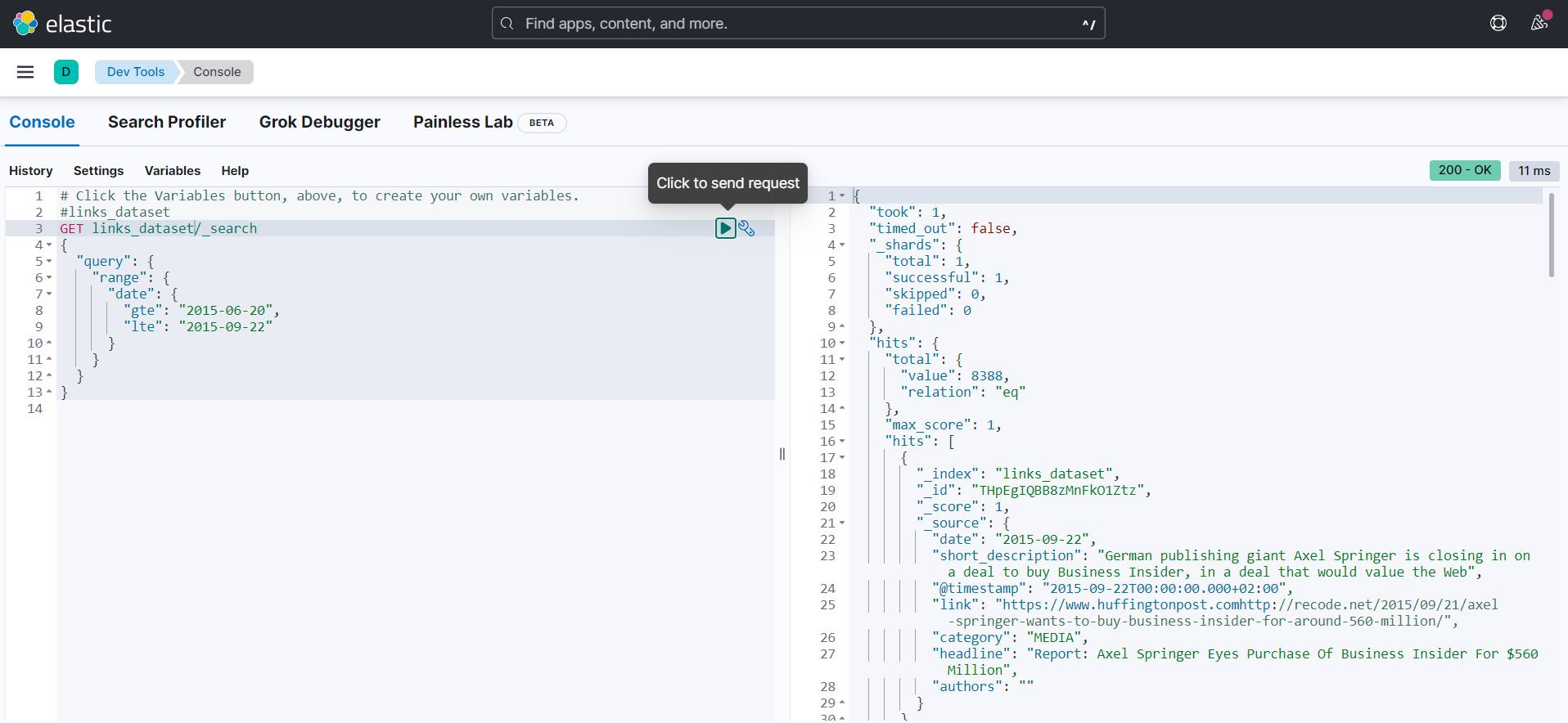


## Some examples over a dataset:

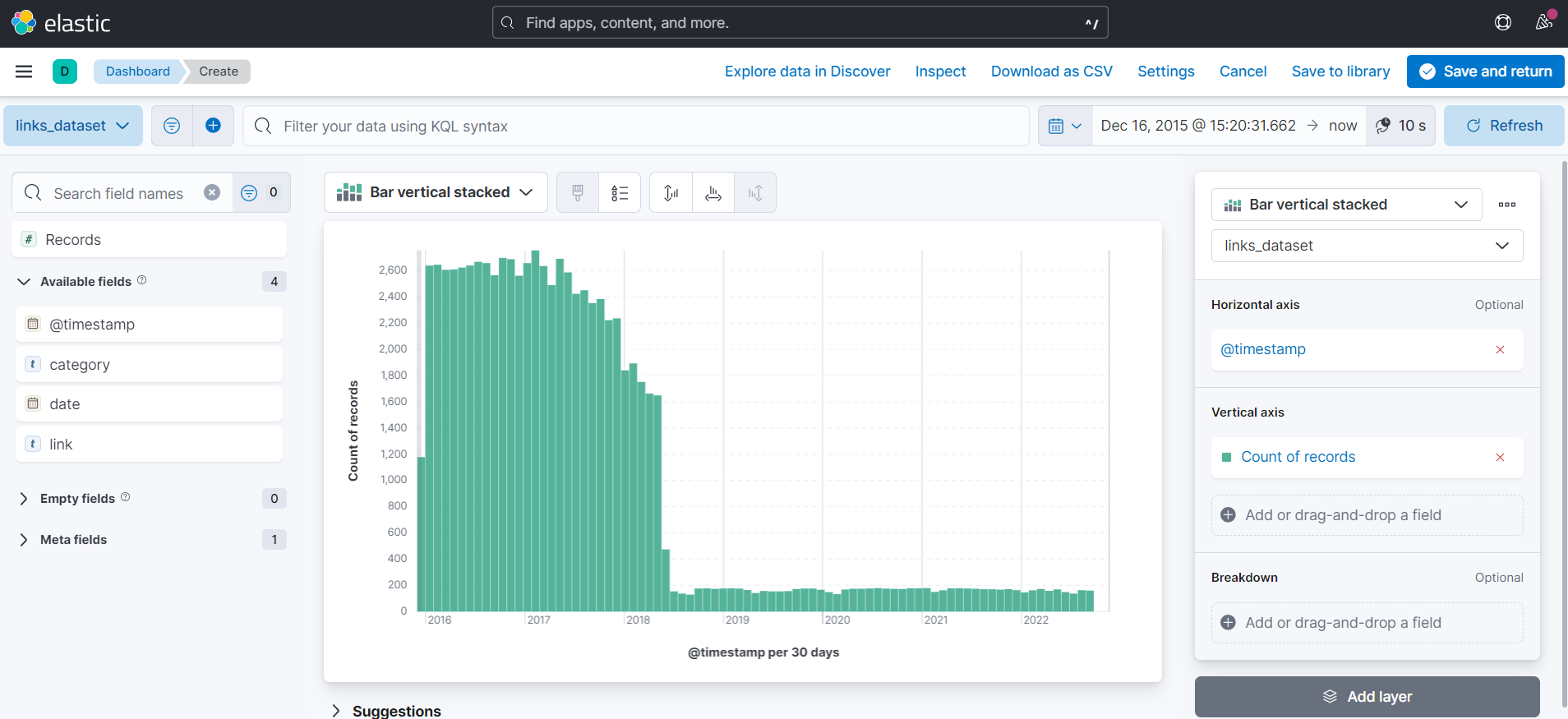
**Using console queries -> bring all data**



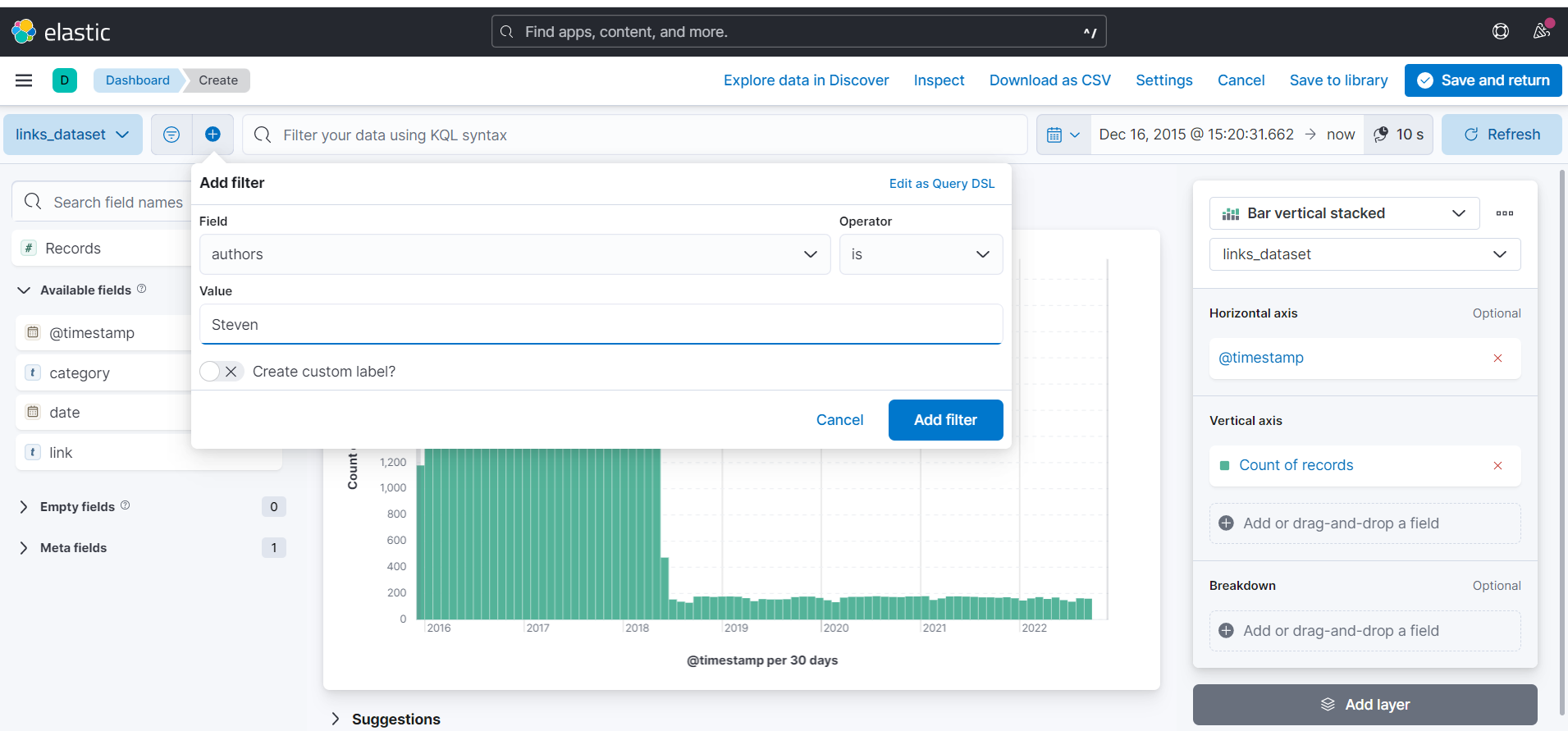
**-> search in a range**



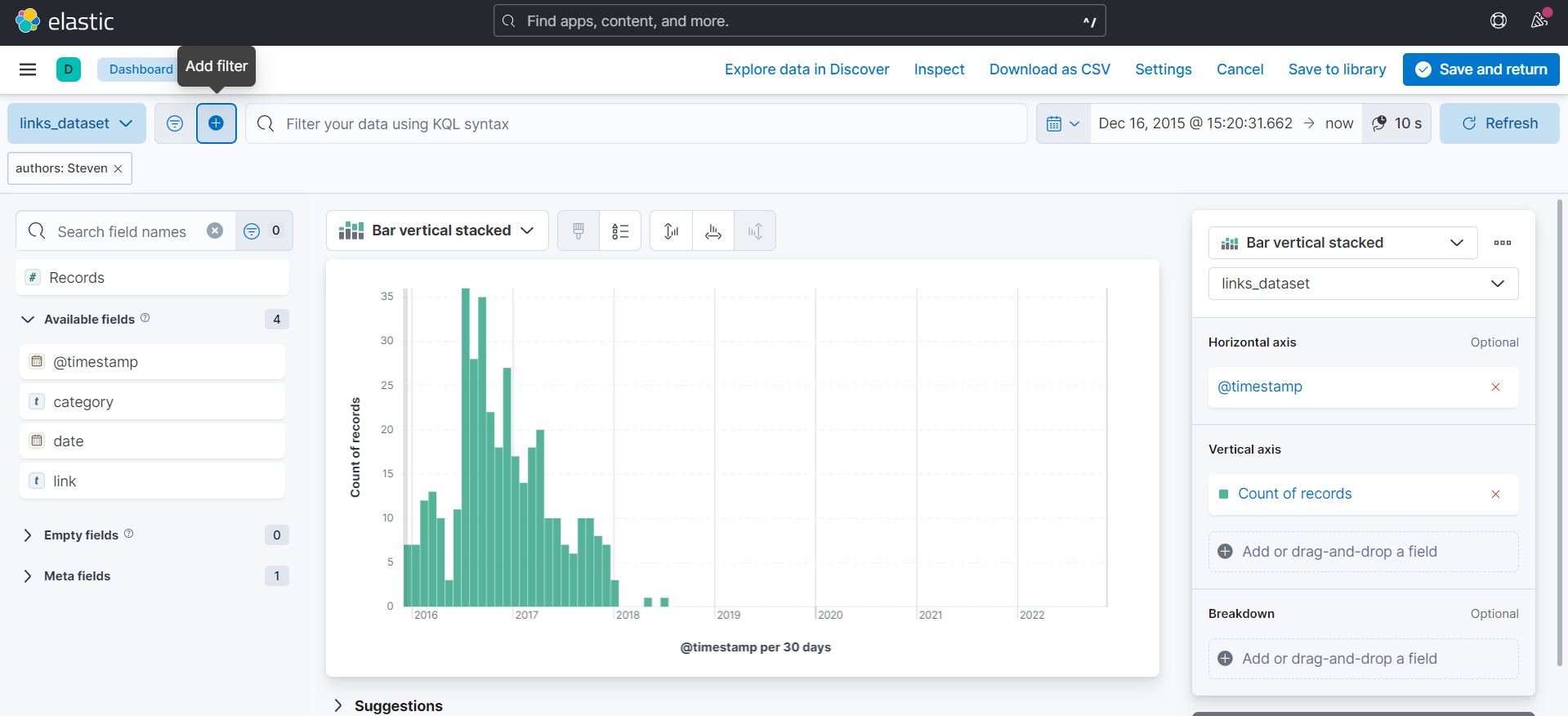
**Using dashboard ->the distribution of records**



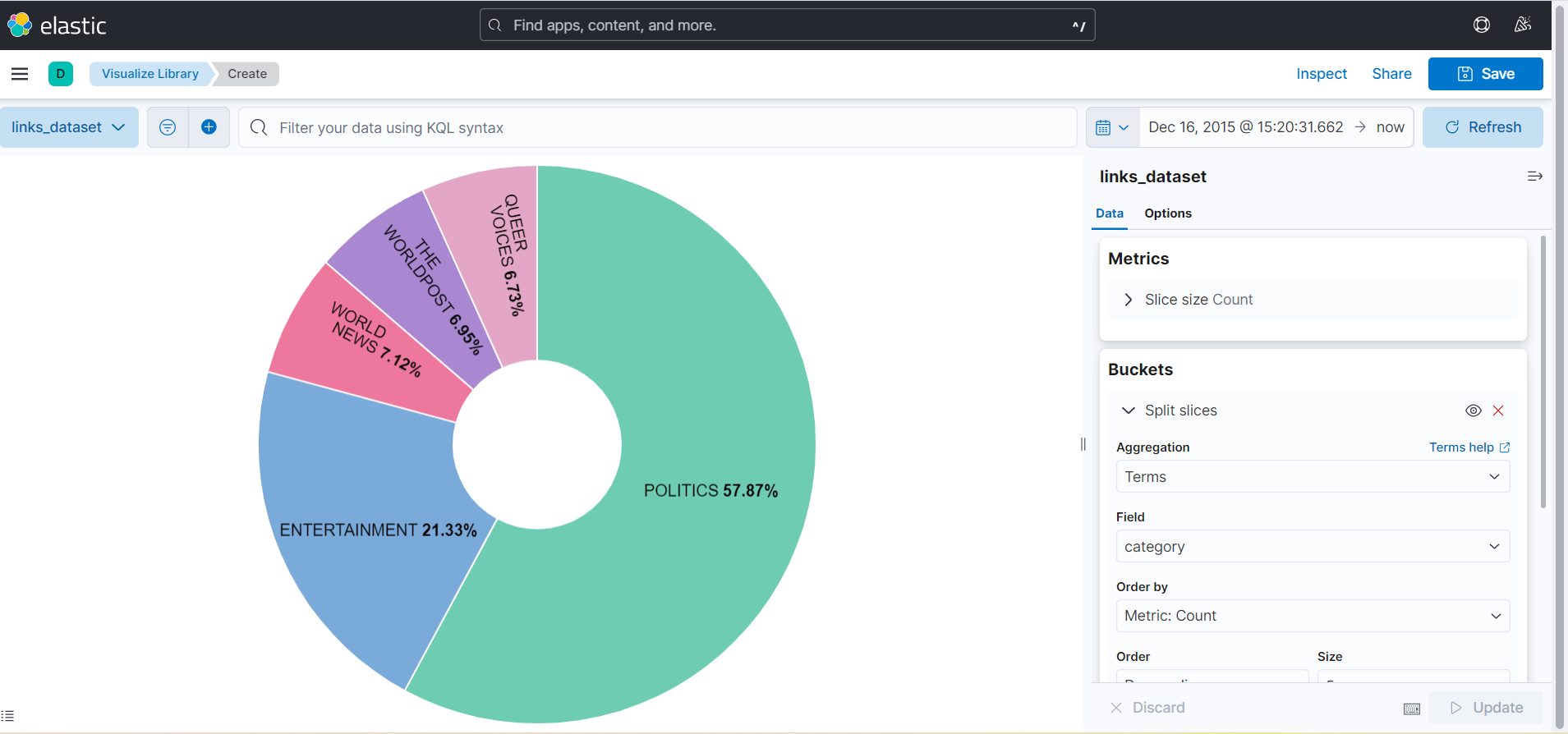
**To add filter**



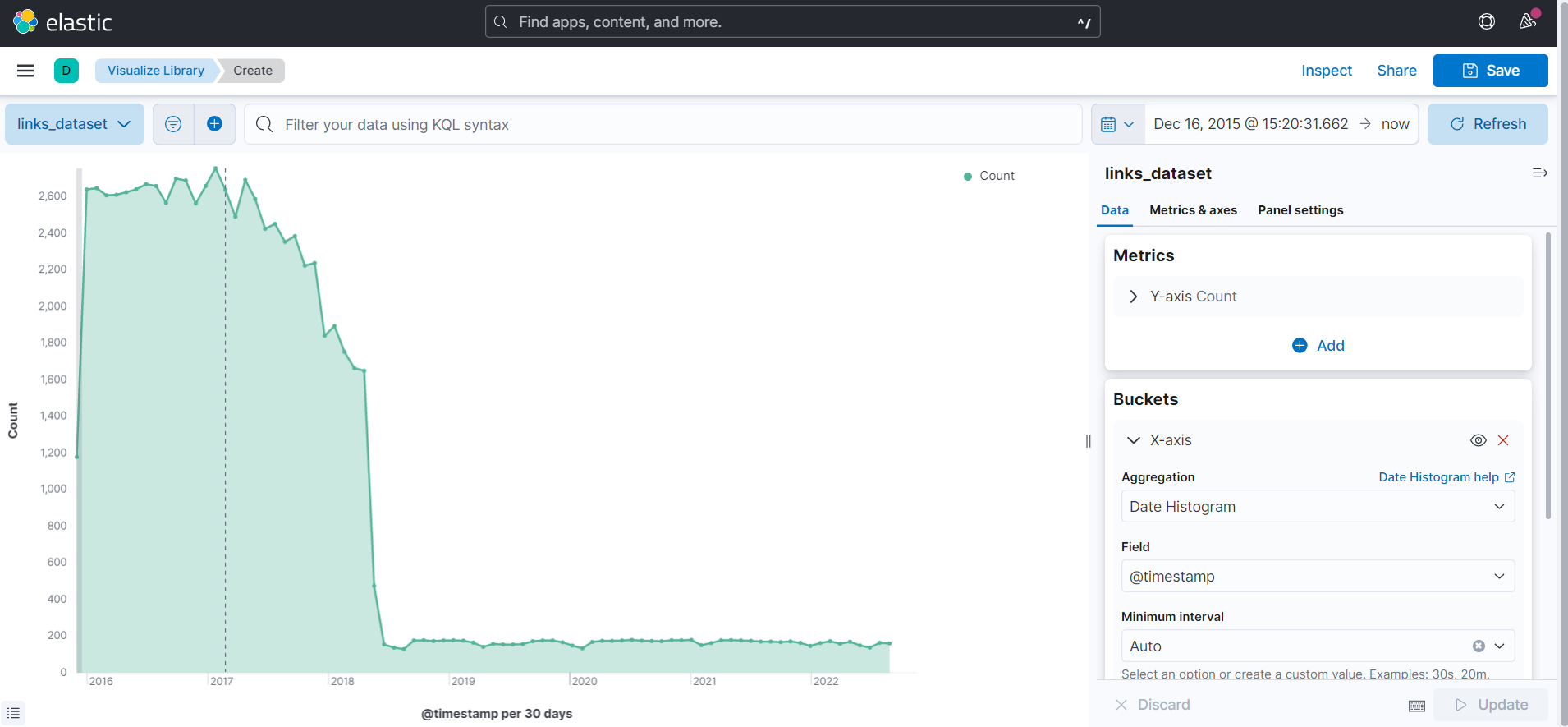
**After adding filter**



**Using a pie chart -> upon category**



**Using a histogram -> upon date**



# installation guide

## Install Elasticsearch:

* + 1. Download & unzip elasticsearch-8.3.2
    2. Add this line in “elasticsearch/config/elasticsearch.yml” file:

action.auto\_create\_index: .monitoring\*,.watches,.triggered\_watches,.watcher-history\*,.ml\*

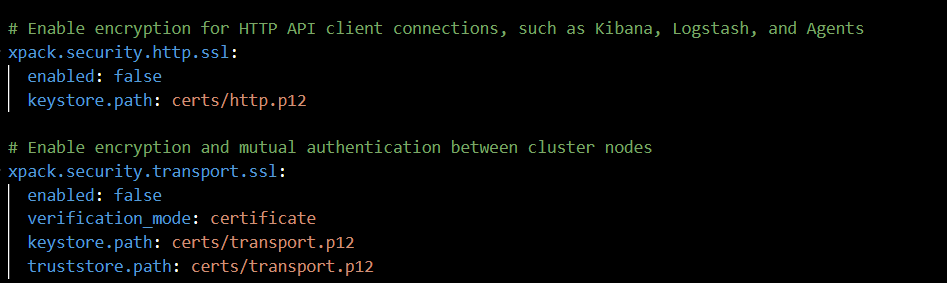
* + 1. Start the Elastic search at “elasticsearch/bin” folder
    - **elasticsearch.bat**
    1. In case of error according to native memory allocation:
* Add the following file at folder “elasticsearch/config/jvm.options.d”

[**jvm.options**](https://drive.google.com/file/d/1iJbjTv551DHDiv4qAwYJU570EfM68elA/view?usp=sharing)

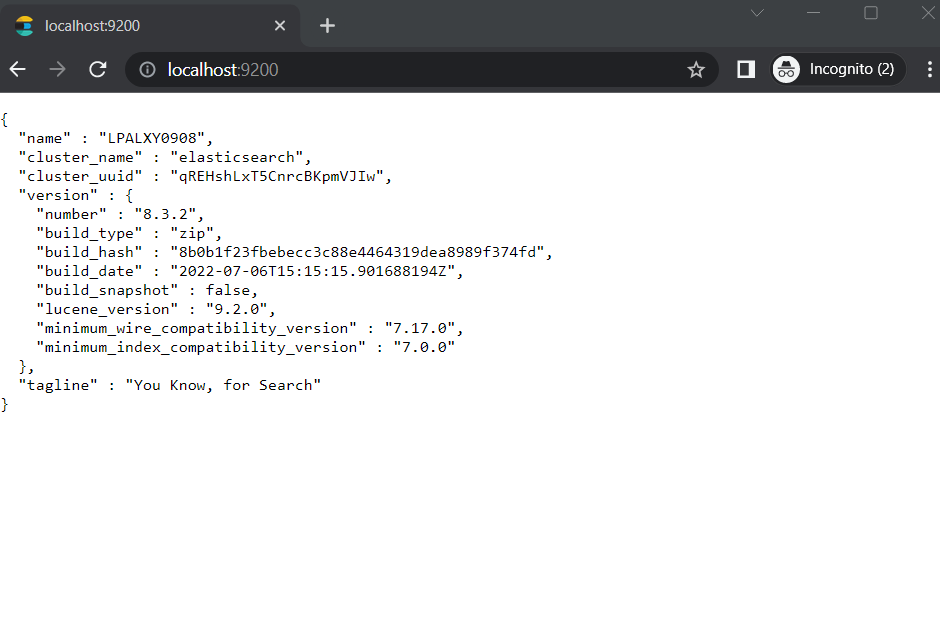
* Start the elastic search again

**elasticsearch.bat**

* + 1. Once you see Token press CTR + C and stop it and copy Paste TOKEN
    2. Update the following to false in “elasticsearch/config/elasticsearch.yml” file:

****

* + 1. Start the Elastic search at folder “elasticsearch/bin”
* **elasticsearch.bat** 
  + 1. Open localhost:9200 in incognito window
* username: elastic
* password: password for the elastic user from token obtained in step 5
  + 1. Now, Elasticsearch is installed successfully

x. To reset password of elastic user any time at “elasticsearch/bin” folder:

* **elasticsearch-reset-password -u elastic**

## Install Kibana:

1. Download & unzip kibana-8.3.2
2. Run this line in “elasticsearch/bin” folder:

elasticsearch-reset-password -u kibana\_system

1. Copy and paste the token to use it later
2. Uncomment the following lines at file “kibana/config/kibana.yml”
   * + server.port: 5601
     + server.host: "localhost"
     + elasticsearch.hosts: ["http://localhost:9200"]
     + elasticsearch.username: "kibana\_system"
     + elasticsearch.password: ""(write the password obtained from step 3)
3. Start kibana at folder “kibana/bin”

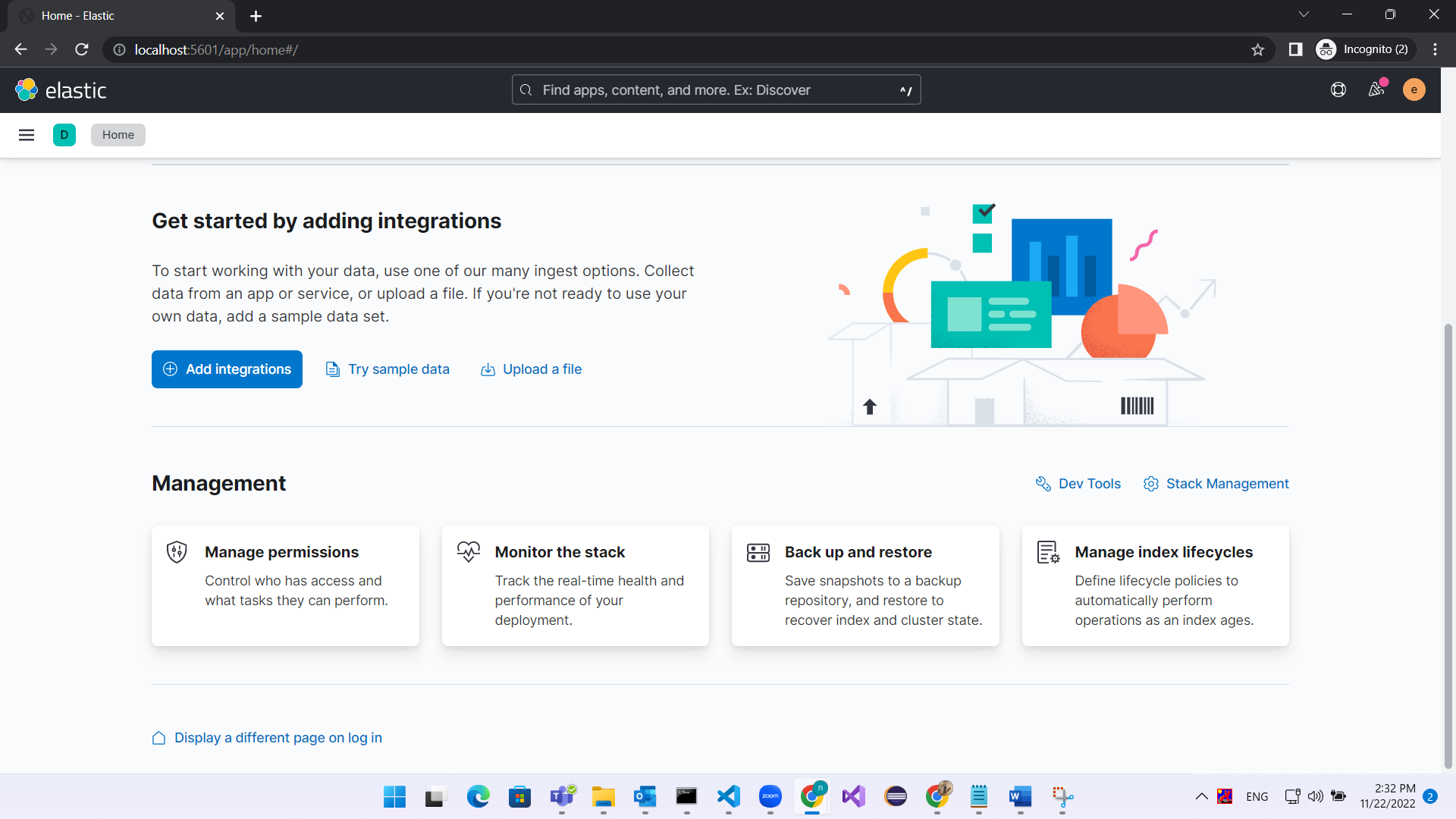
kibana.bat

1. Open localhost:5601 in incognito window

username: elastic

password: password obtained in elastic installation

1. Now, Kibana is installed successfully



1. To reset password of kibana-system at “elasticsearch/bin” folder:

elasticsearch-reset-password -u kibana\_system

## Install Logstash:

1. Download & unzip logstash-8.3.2
2. Add the following file at folder “logstash/config”

[learn.conf](https://drive.google.com/file/d/1AR4Y9VWL7jQKRblnSx4pWwv5EAk06gzU/view?usp=sharing)

update password with the one obtained in elastic installation

1. Start the Logstash

logstash -f .\config\learn.conf --config.reload.automatic

# elasticsearch in angular

## Searchbase:

* + - [Searchbase](https://github.com/appbaseio/searchbox/tree/master/packages/searchbase) is a lightweight and platform agnostic library that provides scaffolding to create search experiences powered by Elasticsearch.
    - Why Searchbase:
      * Offers ability to query Elasticsearch declaratively
      * Lightweight and performance focused
      * Can be easily plugged with any framework like React, Angular, Vue or Svelte as it’s written in pure Javascript
      * Lets you subscribe to state changes which can be binded with various UI component
      * Open source
    - Searchbase installation:
* Install yarn to your project:

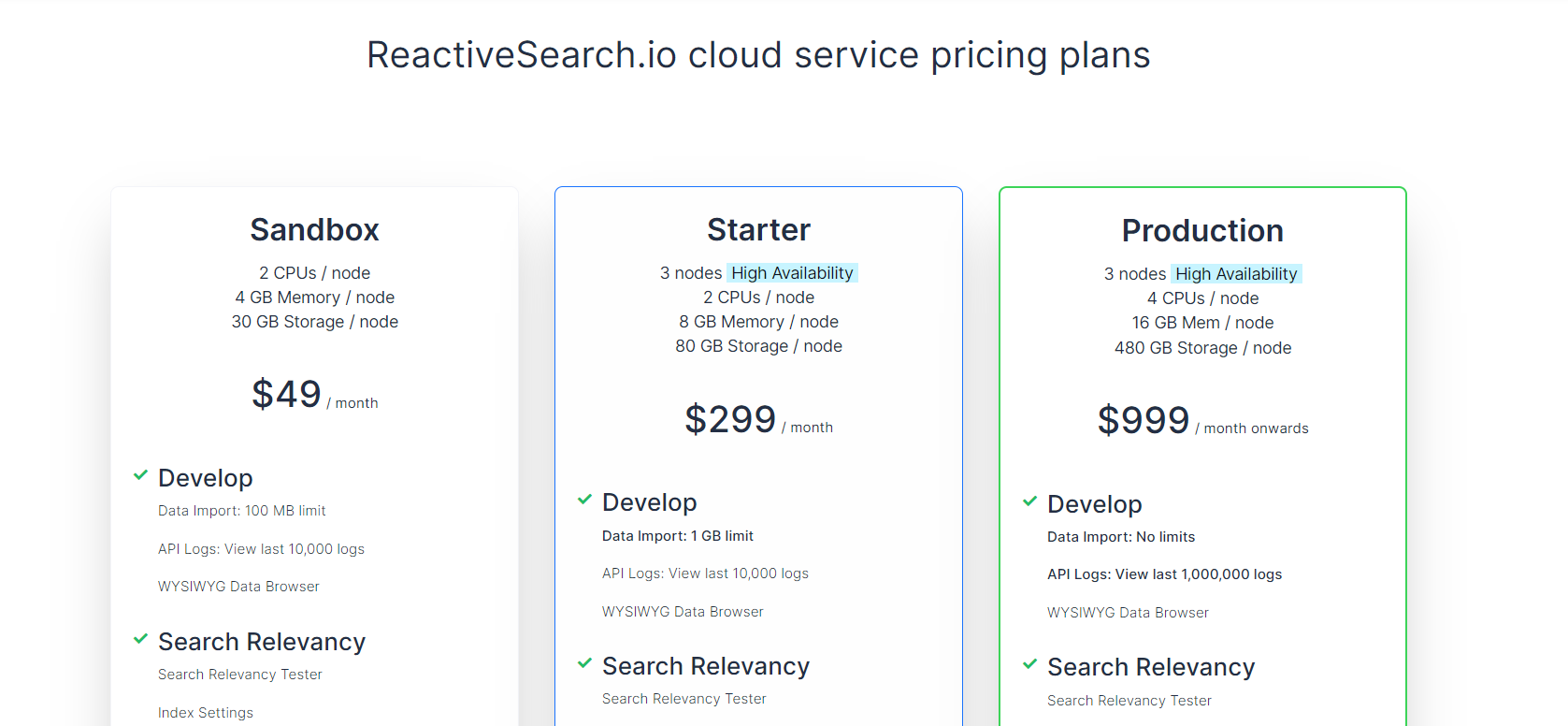
npm install -g yarn

* Install searchbase to your project:

yarn add @appbaseio/searchbase

## ReactiveSearch:

* Reactive search is the evaluation form of appbase.io. It’s a search UI dev tool for consuming JSON data streams with ElasticSearch. It allows developers to build reactive apps with data streams; and stream document updates, search results, and filters.
* Pricing plan is as following:

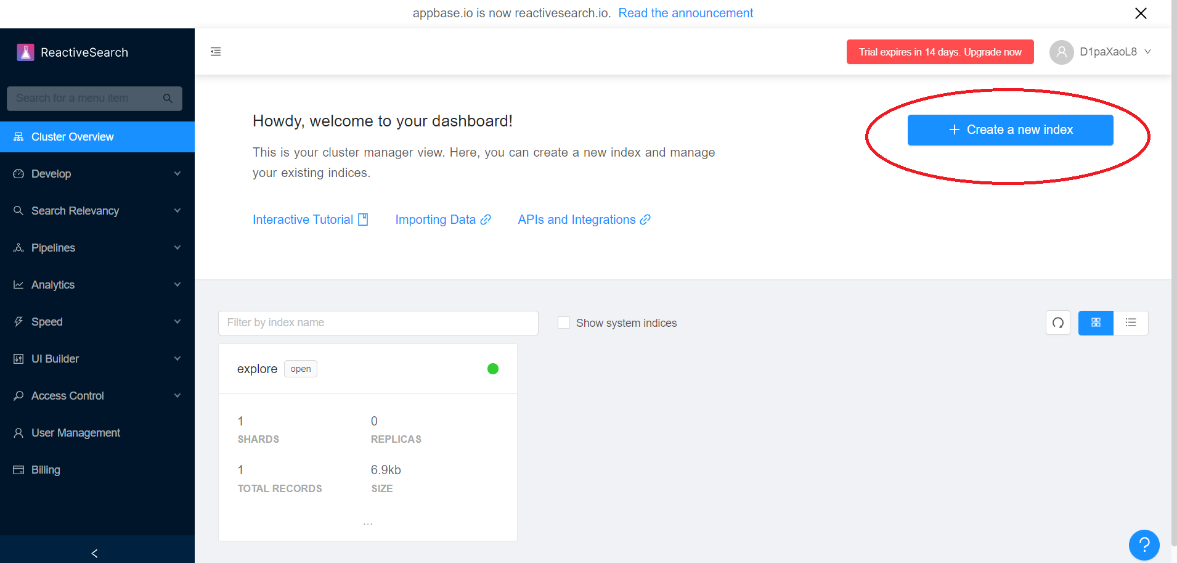


* Create a cluster

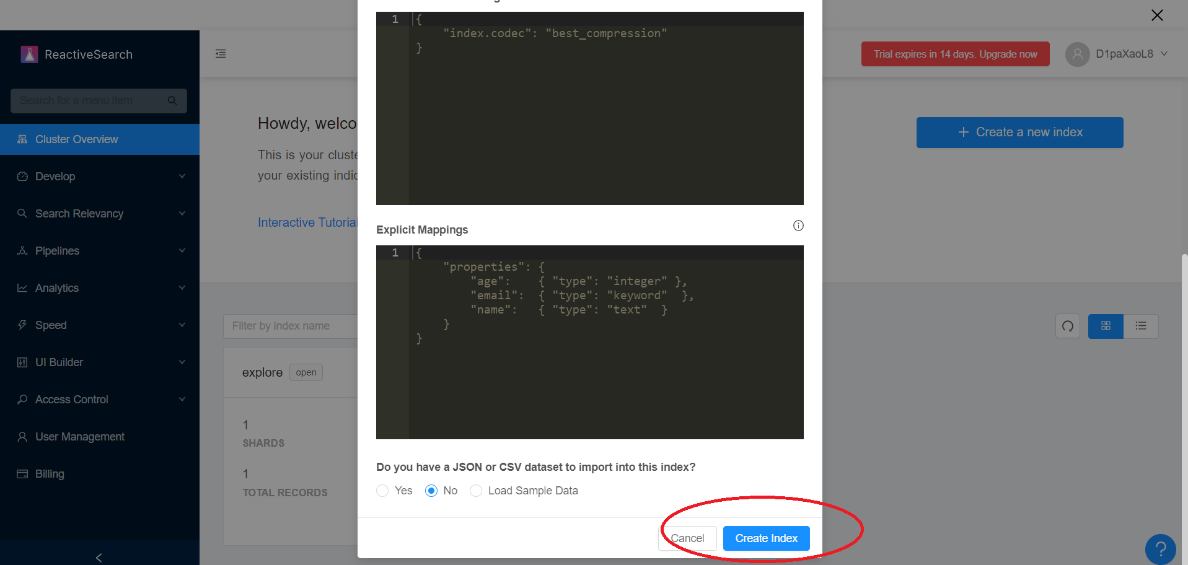
<https://www.youtube.com/watch?v=RtxQDMIVK2g>

* Steps to create an index:

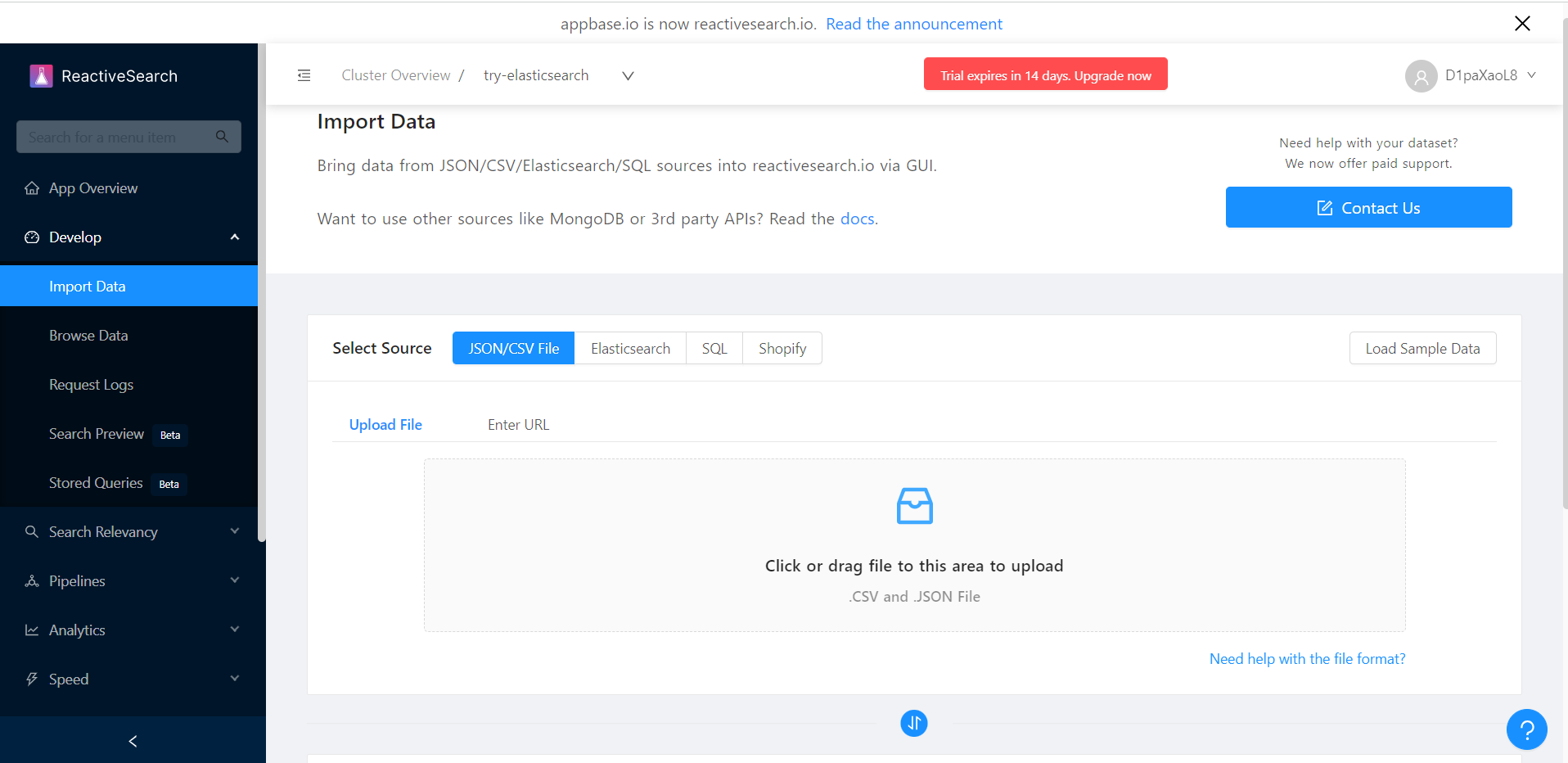
1. Create index:





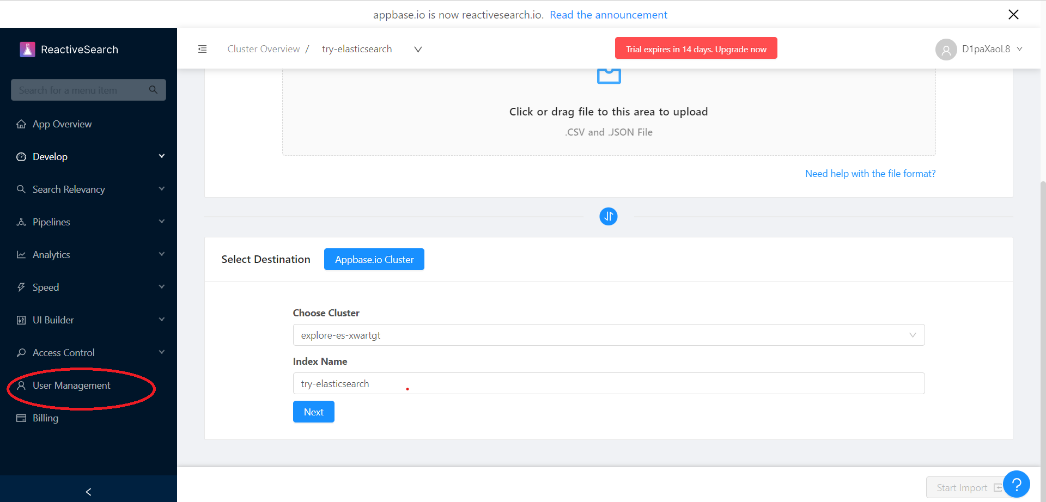


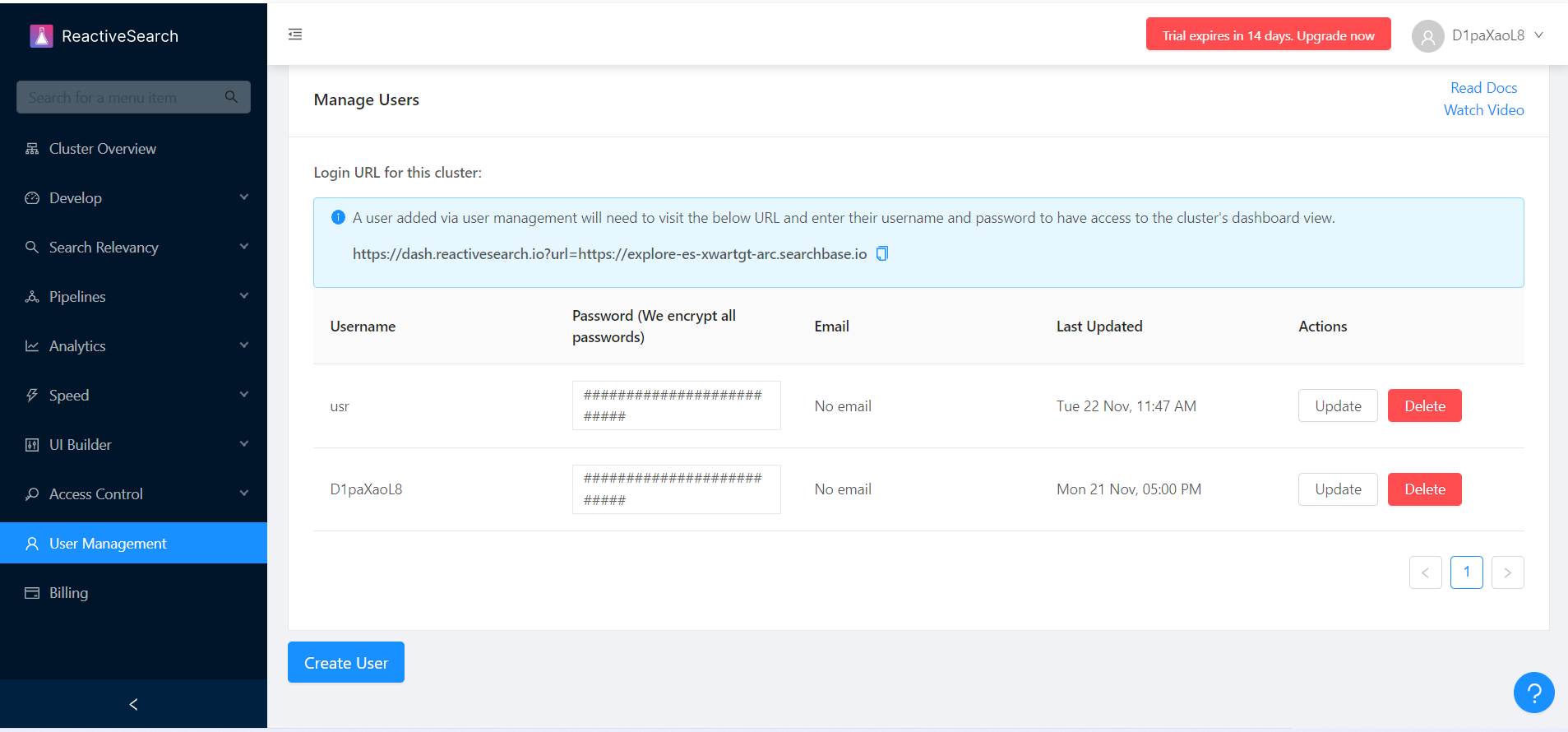




1. You can import data by uploading json file or url of online database

* Steps to create user: (save the username and password to be used later)





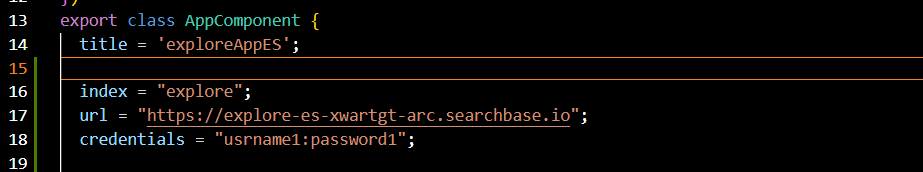
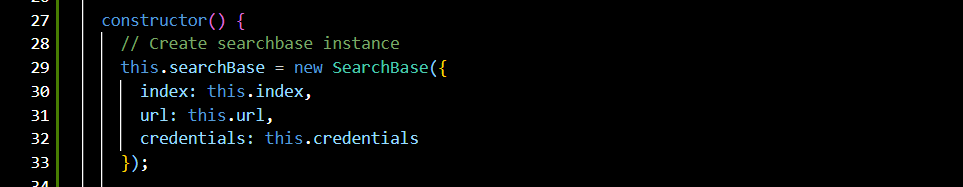
## Building a search UI with angular:

Main reference: [SEARCH UI](https://medium.appbase.io/building-a-search-ui-with-angular-and-elasticsearch-d8a101ace1d)

1. Import the following data to your index in ReactiveSearch:

<https://gist.github.com/siddharthlatest/53d1523e617504b6b731c278b5ec7904>

1. import the @appbaseio/searchbase library in “src/app/app.component.ts” file to create the search controllers.
2. Connect the controller via searchbase class:

* + Credentials:“username:password”e.g“user1:password1”
  + Index: your index name
  + URL: url to reach the index in reactive search

1. Build the search UI





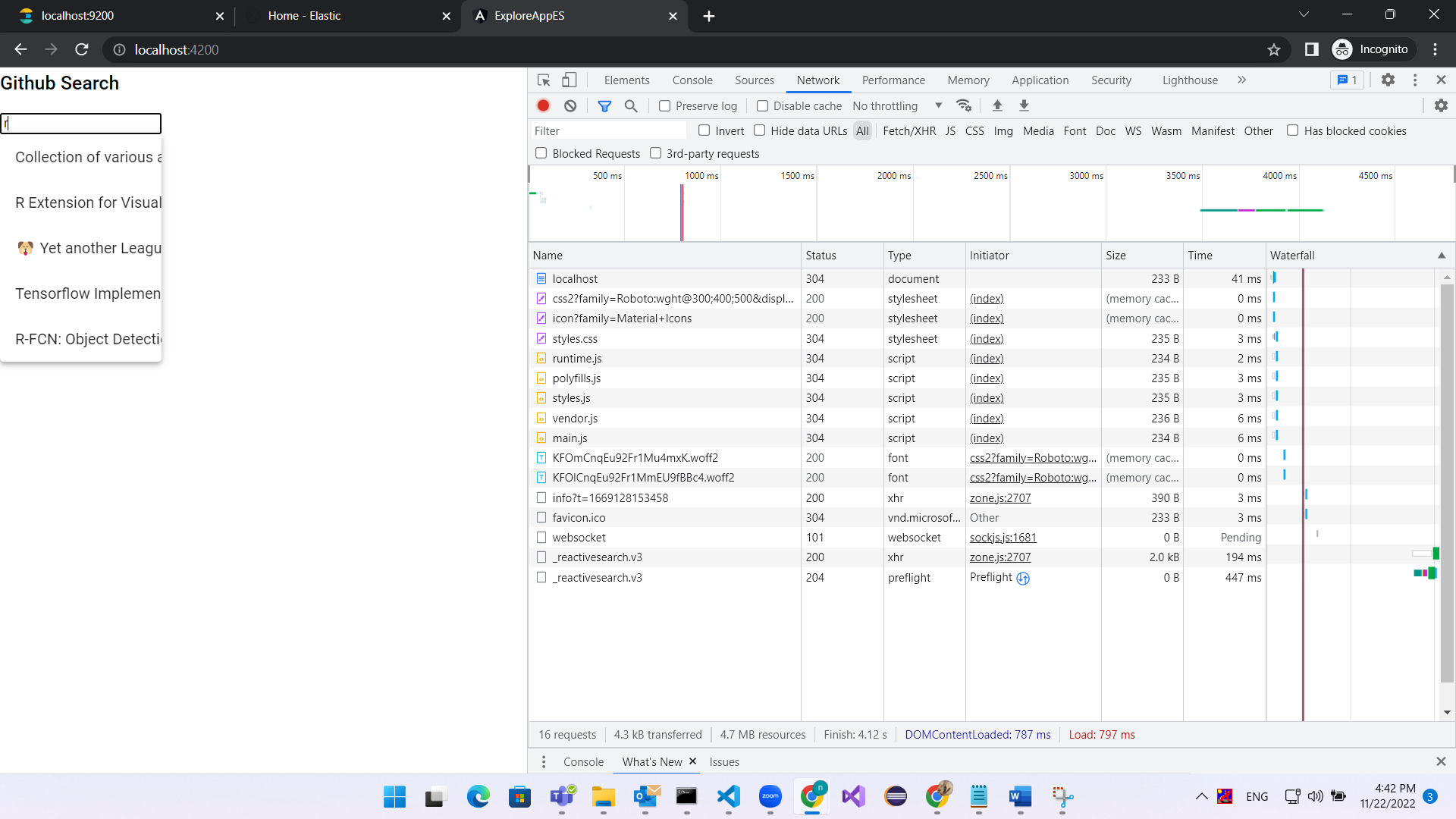
* Register a new controller with a unique identifier search-component
* The dataField property values are the name of the fields on which we want to apply our search
* The size represents the number of suggestions to display
* To improve the search relevancy, we're using multiple data fields with weights, field weights allow weighted search for the index fields.

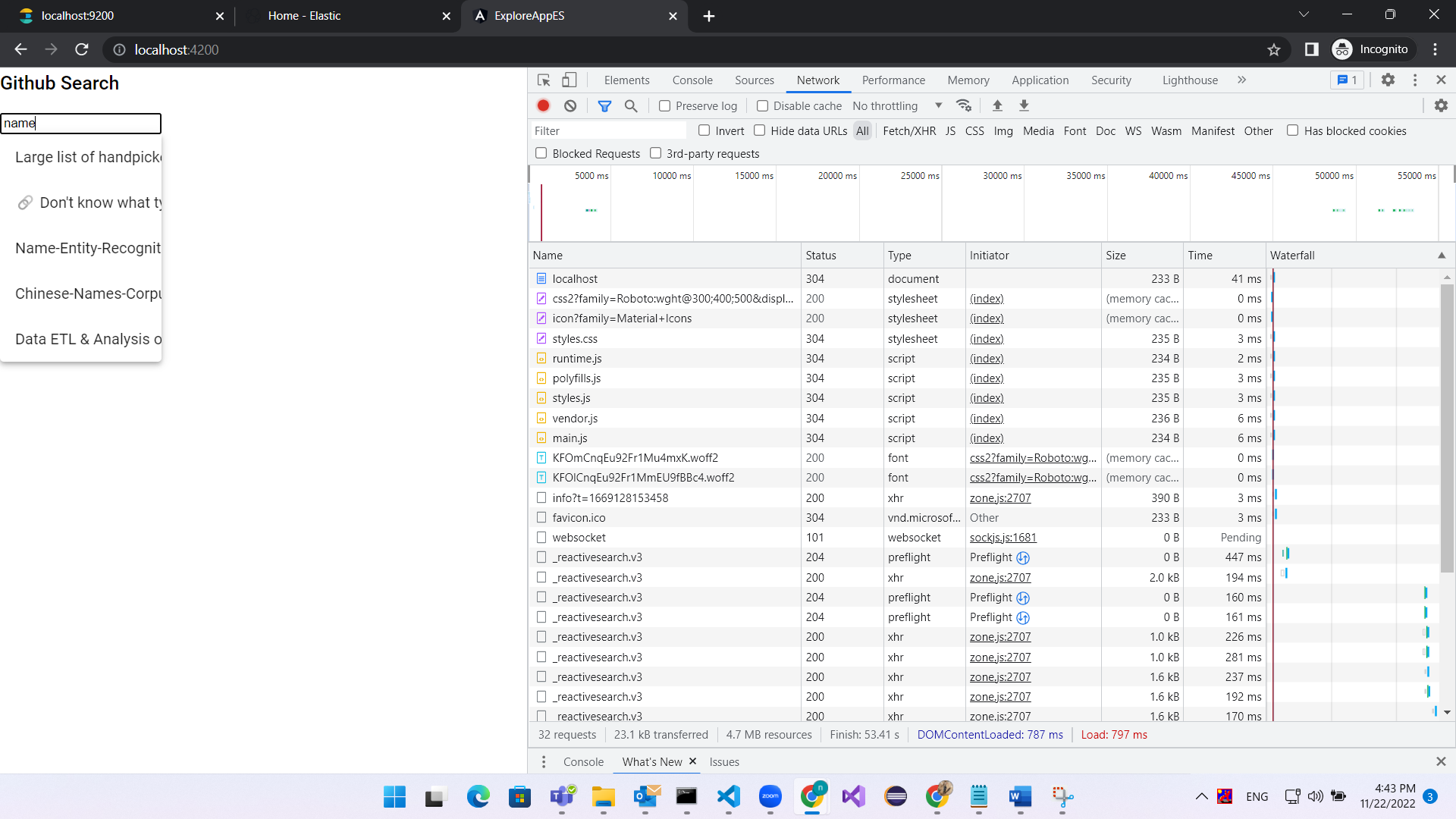
1. Build autosuggestion UI

* ng add @angular/material
* Import the autocomplete module to app.module.ts
* We are using the ngModel directive to bind the input value to the searchComponent controller. We have also defined an event(input) to call handleInput method to get suggestions whenever text changes. Let's define the handleInput method in the app.component.ts file.
* The setSuggestions method calls the setValue method of search controller to set the search controller value. The triggerDefaultQuery method fetches the default query of the controller (to fetch the suggestions). The suggestions property of the search controller would return the parsed suggestions from Elasticsearch.
* Html file looks like:



1. Snipping shots:





# Conclusion

Elasticsearch is a powerful search engine that we can use in audit and logging mechanism but need to check more its limitations as it is not recommended to be used in complex queries.

Steps Done till now:

1- We have successfully been able to call ES from angular directly using sample data from the internet using:

* Reactivesearch developing tool (evaluation form of appbase.io) consuming JSON data streams with Elasticsearch. So, we connected to Elasticsearch using Reactivesearch (by url, index and credentials)
* Searchbase library that provides scaffolding to create search experiences powered by Elasticsearch.

2- We have learned that Kibana is only a visualization tool (wasn’t used in step number 1)

Next Step:

* Learn How to import data from database table to an Elasticsearch index, (search if Kafka can be used, this link might help: [Kafka Elasticsearch Connector Tutorial with Examples (confluent.io)](https://www.confluent.io/blog/kafka-elasticsearch-connector-tutorial/) )
* Here will be two tracks:

1. **Track 1:**
   * 1. Use data from database as Reactivesearch imports data in form of (JSON, CSV) files.
     2. Call the imported data from angular and record the search time.
     3. Call Elasticsearch in angular project using current layout of Ejada.

**b. Track 2 (Recommended Option):** Develop a microservice in Spring Boot calling ES ( This link might help: [ES in spring-boot](https://medium.com/shoutloudz/what-is-elastic-search-how-to-use-it-with-spring-boot-36c6bb32e22f#%3A~%3Atext%3DThere%20are%20two%20ways%20to%2Cit%20creates%20elastic%2Dbased%20queries)), and then make a comparison between the response time of ES and Database to proof that ES is much faster

# references

* [**https://medium.com/@merrinkurian/elasticsearch-as-the-primary-database-5e41b2a0189d**](https://medium.com/@merrinkurian/elasticsearch-as-the-primary-database-5e41b2a0189d)
* [**https://cloud.elastic.co/pricing**](https://cloud.elastic.co/pricing)
* [**https://www.youtube.com/watch?v=DVgKDPf7hOU&t=1836s**](https://www.youtube.com/watch?v=DVgKDPf7hOU&t=1836s)
* [**https://logz.io/blog/grafana-vs-kibana/#:~:text=of%20data%20source.-,Kibana%20on%20the%20other%20hand%2C%20is%20designed%20to%20work%20only,to%20apply%20Kibana%20to%20it**](https://logz.io/blog/grafana-vs-kibana/#:~:text=of%20data%20source.-,Kibana%20on%20the%20other%20hand%2C%20is%20designed%20to%20work%20only,to%20apply%20Kibana%20to%20it)**.**
* [**https://docs.reactivesearch.io/docs/reactivesearch/searchbase/overview/QuickStart/**](https://docs.reactivesearch.io/docs/reactivesearch/searchbase/overview/QuickStart/)
* [**https://medium.appbase.io/building-a-search-ui-with-angular-and-elasticsearch-d8a101ace1d**](https://medium.appbase.io/building-a-search-ui-with-angular-and-elasticsearch-d8a101ace1d)
* [**https://blog.reactivesearch.io/appbaseio-is-reactivesearch**](https://blog.reactivesearch.io/appbaseio-is-reactivesearch)
* [**https://www.loom.com/share/9d794bcd10b049dcad44be56bcccb2c2**](https://www.loom.com/share/9d794bcd10b049dcad44be56bcccb2c2)
* [**example of search UI**](https://6672v.csb.app/)
* [**code of search UI**](https://codesandbox.io/s/6672v?file=/src/app/app.component.ts:635-798)
* **https://github.com/soumilshah1995/learning-logstash-and-elastic-search-plugins/blob/main/ELKStack8.3/readme.md**
* [**https://www.elastic.co/guide/en/elasticsearch/reference/8.3/install-elasticsearch.html**](https://www.elastic.co/guide/en/elasticsearch/reference/8.3/install-elasticsearch.html)
* [**https://github.com/skylot/jadx/issues/1566**](https://github.com/skylot/jadx/issues/1566)
* [**https://stackoverflow.com/questions/48592602/native-memory-allocation-mmap-failed-to-map**](https://stackoverflow.com/questions/48592602/native-memory-allocation-mmap-failed-to-map)
* 🡺**the next one is for installing steps only (till min 8)  
  https://www.youtube.com/watch?v=C-JKcMM6IXE&t=437s**