CIS5900 Term Project Tutorial

Authors: Aakanksha Tasgaonkar; Amogh Mahesh; Louis Tan; Nanjesh Gowda; Tanvi Gidwani

Instructor: Jongwook Woo

Date: 08/10/2019

Lab Tutorial

Aakanksha Tasgaonkar (atasgao@calstatela.edu)

Amogh Mahesh (amahesh3@calstatela.edu)

Louis Tan (ltan3@calstatela.edu)

Nanjesh Gowda (nmandya@calstatela.edu)

Tanvi Gidwani (tgawade@calstatela.edu)

08/10/2016

Trending YouTube Video Data Analysis using Elasticsearch and Kibana

---

Objectives

YouTube (the world-famous video sharing website) maintains a list of the top trending videos on the platform. According to Variety magazine, "To determine the year's top-trending videos, YouTube uses a combination of factors including measuring users' interactions (number of views, shares, comments and likes). This dataset is a daily record of the top trending YouTube videos. The dataset includes several months (and counting) of data on daily trending YouTube videos. Data is included for the US, GB, DE, CA, and FR regions (USA, Great Britain, Germany, Canada, and France, respectively), with up to 200 listed trending videos per day. We would analyze this data to get insights into YouTube trending videos, to see

what is common between these videos. Those insights might also be used by people who want to increase popularity of their videos. Goal of this analysis is to include an insight into YouTube statistics such as the trending videos, the most liked/viewed categories, trending YouTube channels based on these categories. This also includes a well-planned and mapped statistical analysis of the data over a given period. We have used cutting edge technology like Elastic Search and Kibana for the visualization of the procured data.

- Download Logstash.exe, Elasticsearch, Kibana on local.
- Download and map the dataset into Elastic Search with appropriate mapping types.
- Configure Logstash.conf file to ingest the dataset in the form of CSV to Elastic Search.
- Verify successful loading at API console of Elasticsearch.
- Define Index Patterns in Kibana.

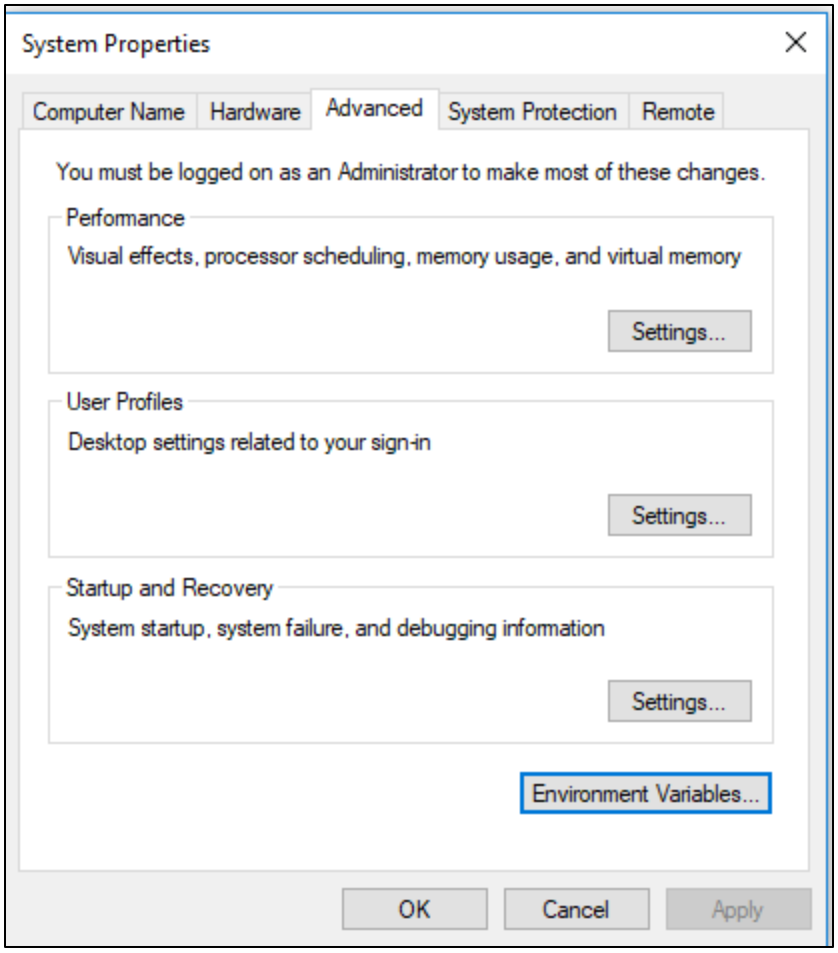
Platform Spec

- Elasticsearch Logstash Kibana
- Server's OS: Windows OS
- Memory Size: Elastic search- 931.5 GB; Kibana- 1.4 GB
- CPU Speed: 1.9 Ghz
- Total Memory Size: Elastic search- 931.5 GB; Kibana- 1.4 GB

Step 1: Download and Install Java Development Kit (AdoptOpenJDK)

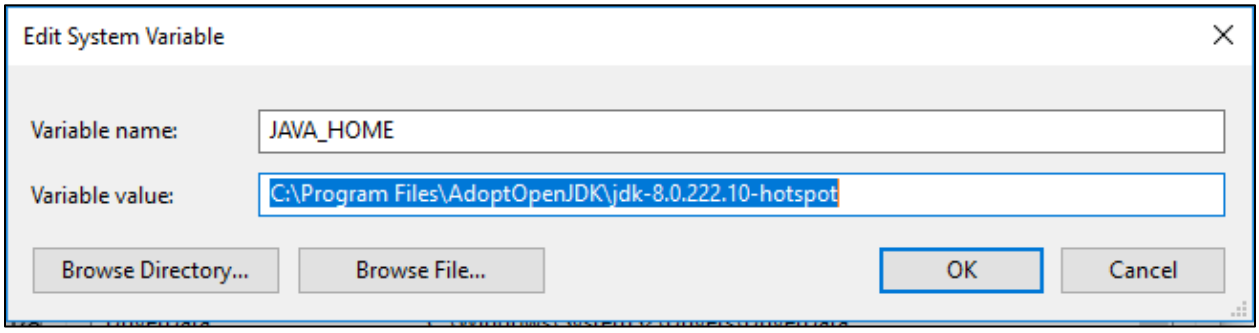
Elasticsearch is built using Java and includes a bundled version of OpenJDK from the JDK maintainers (GPLv2+CE) within each distribution. The bundled JVM is the recommended JVM and is located within the JDK directory of the Elasticsearch home directory.

1. Download JDK kit on local if you don't have one using the below given link:
   i) https://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Once downloaded and installed we need to set the environment variable shown in steps below.
   i) System Properties → Environment Variables



ii) New System Variable → Variable Name: JAVA_HOME & variable value : C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot (Path to your JDK file)

Click ok and apply changes



3. Verify JAVA_HOME is set by using command in Windows command line interface:

   echo %JAVA_HOME%

```
C:\Users\kunal>echo %JAVA_HOME%
C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot

C:\Users\kunal>_
```

Step 2: Download and Extract ELK Stack on Local

To start with the lab, we need to download and extract Elasticsearch, Logstash and Kibana on local.

1. Download Elastic search, Logstash, Kibana from the below given links:
   i) https://www.elastic.co/downloads/elasticsearch
   ii) https://www.elastic.co/downloads/kibana
   iii) https://www.elastic.co/downloads/logstash

2. Extract Logstash using the following commands on Windows Command Line Interface.

   i) Unzip xvf file.gz

```
Command Prompt
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/blocks.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/database_wait_statistics.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/errors.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/metric.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/querystore_runtime_statistics.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/querystore_wait_statistics.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/sql_insights.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/sql_diagnostics/timeouts.json
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.advanced.yml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.advanced_invalid1.yml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.advanced_invalid2.yml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.basic.yml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.basic_invalid1.yml
x logstash-7.2.1/x-pack/spec/modules/azure/samples/yaml/logstash.basic_invalid2.yml
x logstash-7.2.1/x-pack/spec/modules/module_license_checker_spec.rb
x logstash-7.2.1/x-pack/spec/monitoring
x logstash-7.2.1/x-pack/spec/monitoring/inputs
x logstash-7.2.1/x-pack/spec/monitoring/inputs/metrics
x logstash-7.2.1/x-pack/spec/monitoring/inputs/metrics/state_event
x logstash-7.2.1/x-pack/spec/monitoring/inputs/metrics/state_event/lir_serializer_spec.rb
x logstash-7.2.1/x-pack/spec/monitoring/inputs/metrics_spec.rb
x logstash-7.2.1/x-pack/spec/monitoring/inputs/timer_task_logger_spec.rb
x logstash-7.2.1/x-pack/spec/monitoring/internal_pipeline_source_spec.rb
x logstash-7.2.1/x-pack/spec/monitoring/schemas
x logstash-7.2.1/x-pack/spec/monitoring/schemas/monitoring_document_schema.json
x logstash-7.2.1/x-pack/spec/monitoring/schemas/states_document_schema.json
x logstash-7.2.1/x-pack/spec/spec_helper.rb
x logstash-7.2.1/x-pack/spec/support
x logstash-7.2.1/x-pack/spec/support/helpers.rb
x logstash-7.2.1/x-pack/spec/support/matchers.rb
x logstash-7.2.1/x-pack/src
x logstash-7.2.1/x-pack/src/test
x logstash-7.2.1/x-pack/src/test/java
x logstash-7.2.1/x-pack/src/test/java/org
x logstash-7.2.1/x-pack/src/test/java/org/logstash
x logstash-7.2.1/x-pack/src/test/java/org/logstash/xpack
x logstash-7.2.1/x-pack/src/test/java/org/logstash/xpack/test
x logstash-7.2.1/x-pack/src/test/java/org/logstash/xpack/test/RSpecIntegrationTests.java
x logstash-7.2.1/x-pack/src/test/java/org/logstash/xpack/test/RSpecTests.java
x logstash-7.2.1/LICENSE.txt
x logstash-7.2.1/logstash-core/lib/logstash/build.rb

D:\Summer 2019 Course>
Type here to search
5:12 PM
7/30/2019
```

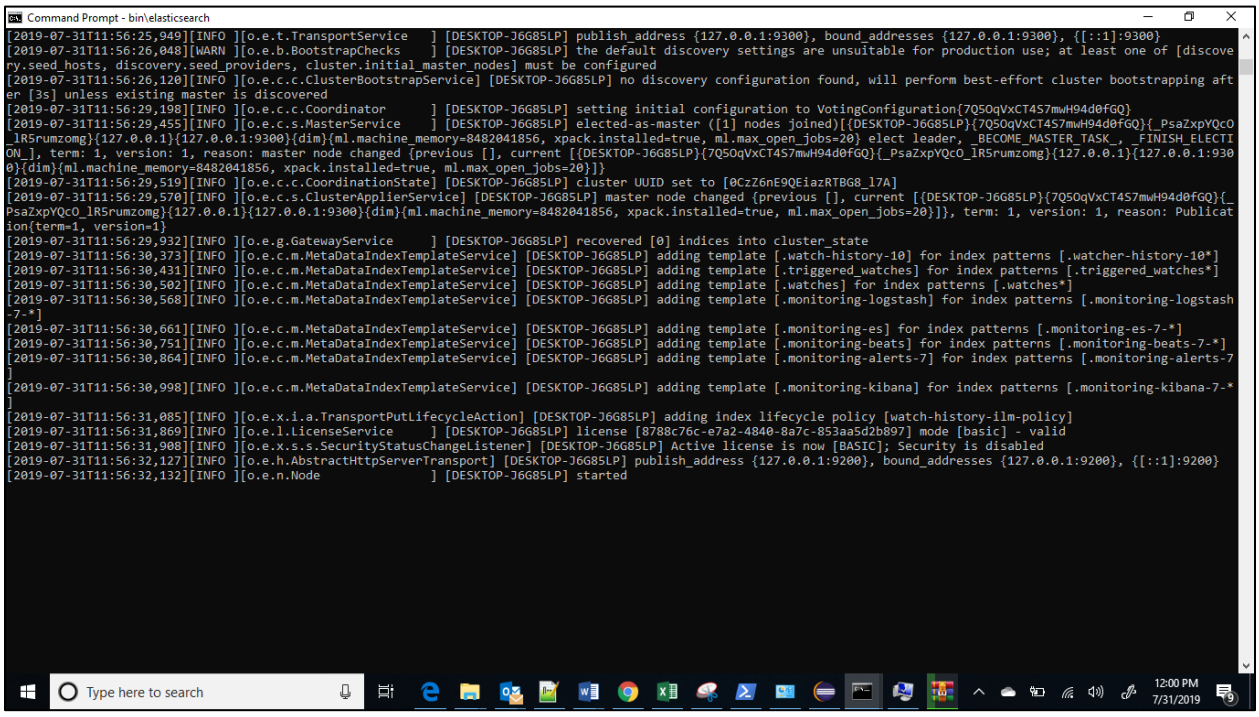
3. Similarly extract Elasticsearch and Kibana using the command line interface.

4. Once the ELK stack is downloaded, we need to start Elastic as well as Kibana in order to ingest the configuration file through Logstash and set up the mappings in Elasticsearch.

5. Initiate Elasticsearch through Windows command line interface by using the following commands:

   i) Go to your elastic folder on local: cd elasticsearch-7.3.0

   ii) bin\elasticsearch



6. Similarly, initiate Kibana through Windows command line interface by using the following commands:

   i) Go to your kibana folder on local: cd kibana

   ii) bin\kibana

```
Command Prompt - bin\kibana
  log   [19:07:14.199] [info][status][plugin:oss_telemetry@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:14.204] [info][status][plugin:file_upload@7.3.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
  log   [19:07:14.210] [warning][encrypted_saved_objects] Generating a random key for xpack.encrypted_saved_objects.encryptionKey. To be able to decrypt encrypted saved
 objects attributes after restart, please set xpack.encrypted_saved_objects.encryptionKey in kibana.yml
  log   [19:07:14.212] [info][status][plugin:encrypted_saved_objects@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:14.225] [info][status][plugin:snapshot_restore@7.3.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
  log   [19:07:14.242] [info][status][plugin:actions@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:14.258] [info][status][plugin:alerting@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:14.265] [info][status][plugin:data@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:15.383] [info][status][plugin:timelion@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:15.392] [info][status][plugin:ui_metric@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:15.397] [info][status][plugin:visualizations@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:19.128] [info][status][plugin:elasticsearch@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.489] [info][license][xpack] Imported license information from Elasticsearch for the [data] cluster: mode: basic | status: active
  log   [19:07:19.499] [info][status][plugin:xpack_main@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.500] [info][status][plugin:graph@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.504] [info][status][plugin:searchprofiler@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.507] [info][status][plugin:ml@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.509] [info][status][plugin:tilemap@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.510] [info][status][plugin:watcher@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.511] [info][status][plugin:grokdebugger@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.515] [info][status][plugin:logstash@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.518] [info][status][plugin:beats_management@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.520] [info][status][plugin:index_management@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.521] [info][status][plugin:index_lifecycle_management@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.523] [info][status][plugin:rollup@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.527] [info][status][plugin:remote_clusters@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.530] [info][status][plugin:cross_cluster_replication@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.530] [info][status][plugin:file_upload@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.532] [info][status][plugin:snapshot_restore@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:19.534] [info][kibana-monitoring][monitoring] Starting monitoring stats collection
  log   [19:07:19.551] [info][status][plugin:maps@7.3.0] Status changed from yellow to green - Ready
  log   [19:07:23.750] [warning][reporting] Generating a random key for xpack.reporting.encryptionKey. To prevent pending reports from failing on restart, please set xp
ack.reporting.encryptionKey in kibana.yml
  log   [19:07:23.763] [info][status][plugin:reporting@7.3.0] Status changed from uninitialized to green - Ready
  log   [19:07:23.910] [info][task_manager] Installing .kibana_task_manager index template version: 7030099.
  log   [19:07:23.989] [info][task_manager] Installed .kibana_task_manager index template: version 7030099 (API version 1)
  log   [19:07:25.580] [info][migrations] Creating index .kibana_1.
  log   [19:07:25.981] [info][migrations] Pointing alias .kibana to .kibana_1.
  log   [19:07:26.066] [info][migrations] Finished in 498ms.
  log   [19:07:26.072] [info][listening] Server running at http://localhost:5601
  log   [19:07:26.114] [info][server][Kibana][http] http server running
  log   [19:07:26.868] [info][status][plugin:spaces@7.3.0] Status changed from yellow to green - Ready
Type here to search
12:07 PM
7/31/2019
```

7. Now we need to navigate to our Kibana host URL as shown in the above screenshot to access Kibana through web browser on local:

   i) log [19:07:26.072] [info][listening] Server running at http://localhost:5601

```
log   [19:07:26.066] [info][migrations] Finished in 498ms.
log   [19:07:26.072] [info][listening] Server running at http://localhost:5601
log   [19:07:26.114] [info][server][Kibana][http] http server running
log   [19:07:26.868] [info][status][plugin:spaces@7.3.0] Status changed from yellow to green - Ready
```

Step 3: Download and Map the Datasets

This tutorial requires Trending YouTube Video Statistics data set to be downloaded on to your local. You can download the dataset from Kaggle on the below given link:

   i) Dataset link: https://www.kaggle.com/datasnaek/youtube-new

Before you load the data set, you must set up mappings for the fields. Mappings divide the documents in the index into logical groups and specify the characteristics of the fields. These characteristics include the searchability of the field and whether it's tokenized or broken up into separate words.

In Kibana Dev Tools > Console, set up a mapping for the YouTube data set by copying and pasting the following. Then, select Play button:

```
PUT global/
{
  "mappings": {
    "properties": {
      "location": {
        "type": "geo_point"
      },
      "title": {
        "type": "keyword"
      },
      "channel_title": {
        "type": "keyword"
      },
      "category_id": {
        "type": "keyword"
      },
      "tags": {
        "type": "text"
      },
      "Country": {
        "type": "keyword"
      },
      "trending_date": {
        "type": "date",
```

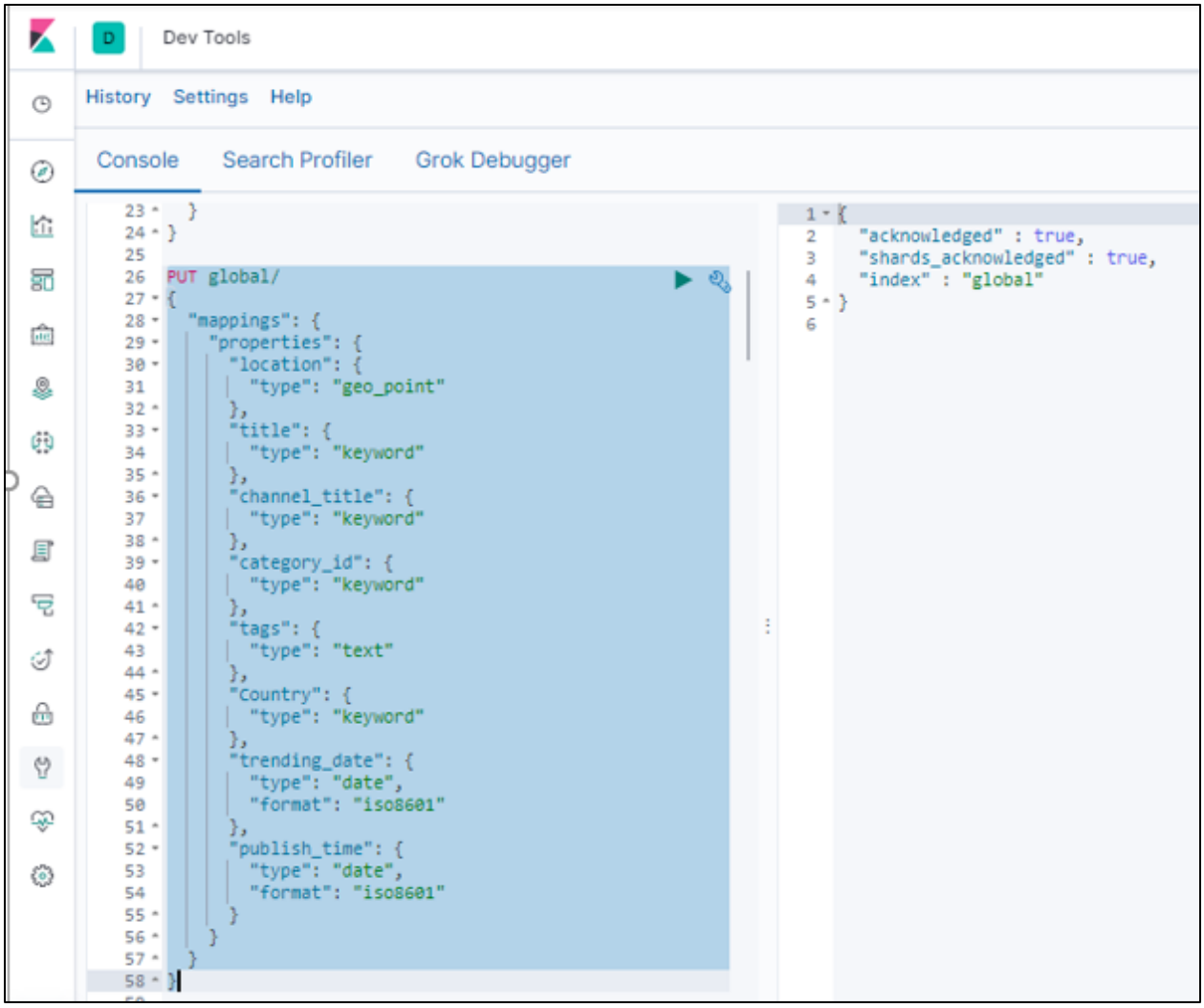
```
        "format": "iso8601"
      },
      "publish_time": {
        "type": "date",
        "format": "iso8601"
      }
    }
  }
}
```

This mapping specifies field characteristics for the data set:

i) The title, channel_title, category_id and Country fields are keyword fields. These fields are not analyzed. The strings are treated as a single unit even if they contain multiple words.

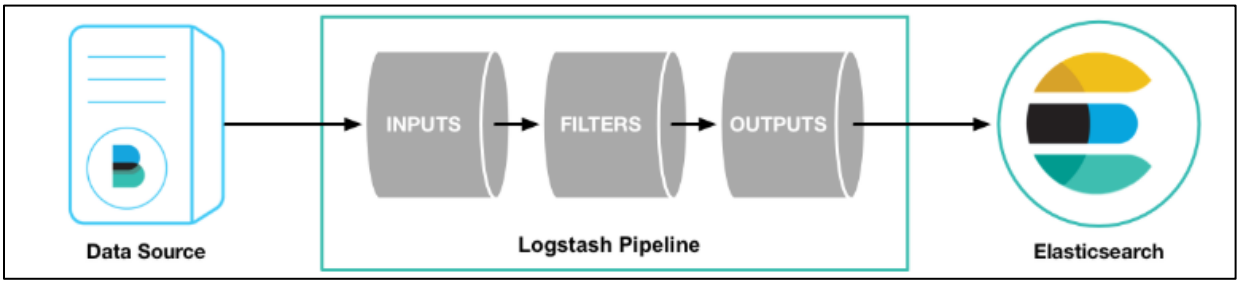
ii) The location is mapped in the form of geo_point to accept latitude-longitude pairs and to aggregate documents geographically.

The trending_date and publish_time are the date fields in iso8601 format.



Step 4: Parsing Logs with Logstash

We now need to ingest our data from the dataset into Elasticsearch for visualizations through Logstash which is a primary part of ELK stack that collects, parses and transforms our data into Elasticsearch cluster. Processing of Logstash as given below in pictorial representation:



i) We need to define the input and filters in the form of Logstash configuration file. Open notepad++ and define the code as given below and save it as .conf file:

```
input{
      file{
            path => "D:/Summer_2019_Course/Project/global_youtube.csv"
            start_position => "beginning"
            sincedb_path => "NUL"
      }
}
filter {
      csv {
            separator => ","
            columns    =>    ["video_id",    "trending_date",    "title",
"channel_title", "category_id", "publish_time", "tags", "views", "likes",
      "dislikes",     "comment_count",     "latitude"     ,"longitude",
"Country"]
            }
            mutate {convert => ["views", "integer"] }
            mutate {convert => ["likes", "integer"] }
            mutate {convert => ["dislikes", "integer"] }
            mutate {convert => ["comment_count", "integer"] }
            mutate { convert => {"latitude" => "float"} }
            mutate { convert => {"longitude" => "float"} }
            mutate { rename => {"latitude" => "[location][lat]"} }
            mutate { rename => {"longitude" => "[location][lon]"} }
            date {match => [ "trending_date","ISO8601" ]
```

```
            timezone => "America/Chicago"}
            date {match => [ "publish_time","ISO8601" ]
            timezone => "America/Chicago"}
            if [message] =~ "^?video_id" { drop {} }
            }
output {
      elasticsearch {
            hosts => "localhost"
            index => "publish_time_global"
            document_type => "_doc"
            }
            stdout {}
      }
```

ii) The configuration file would take the input i.e. YouTube dataset which is in the form of a comma separated file, define its columns and transform data into integer, location and match the date in its correct format.

iii) Now go to the Logstash folder through windows command line:

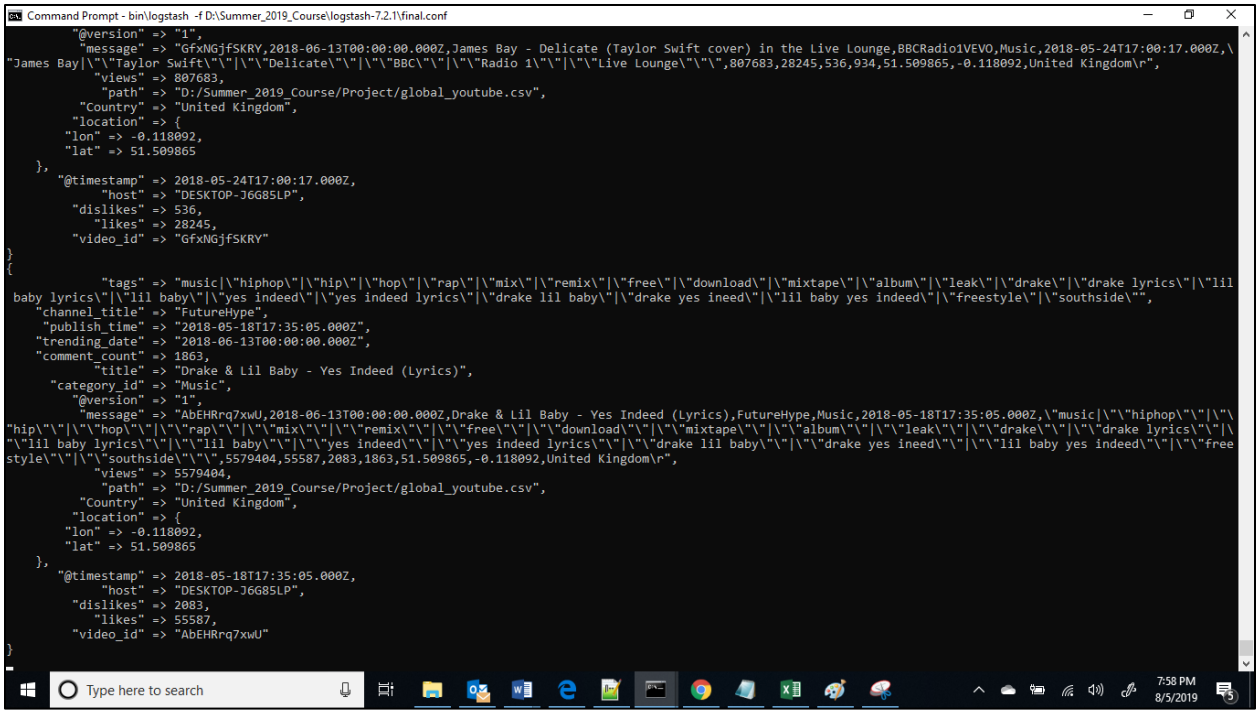
cd logstash-7.2.1

iv) Execute the below command in order to ingest our Final.conf i.e. Logstash configuration file into our Elasticsearch cluster

bin\logstash -f D:\Summer_2019_Course\logstash-7.2.1\final.conf

```
D:\Summer_2019_Course\logstash-7.2.1>bin\logstash -f D:\Summer_2019_Course\logstash-7.2.1\final.conf
```

v) Once executed the data from the dataset will start getting loaded into our Elasticsearch cluster as shown below:



vi) In Kibana Dev Tools > Console, verify that the data from the dataset has been ingested into our Elasticsearch cluster by using the API console in Kibana

Command will give you the count of documents uploaded in Elasticsearch.

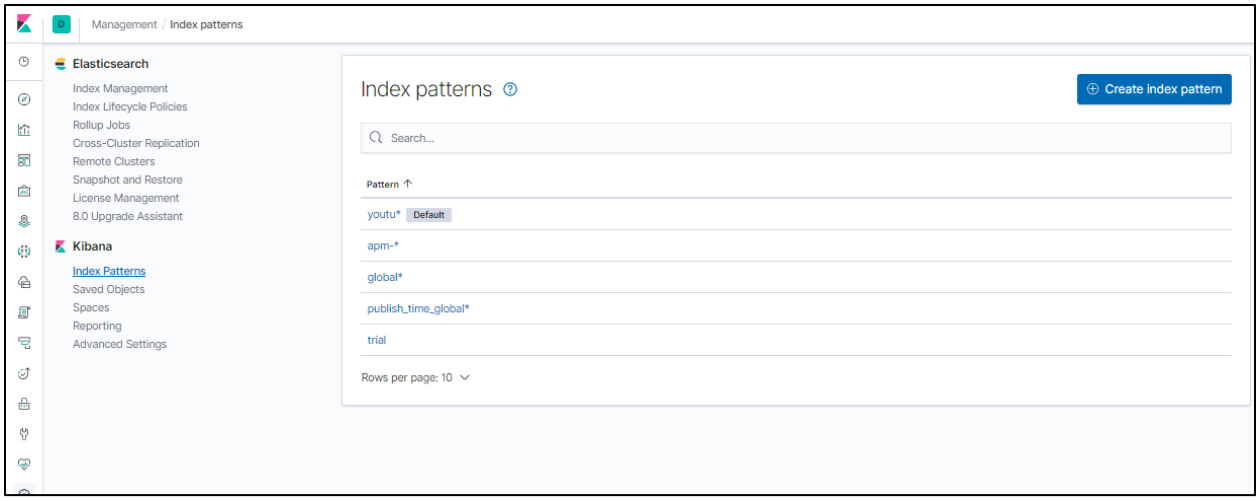
Step 4: Defining your index patterns

Index patterns tell Kibana which Elasticsearch indices you want to explore. An index pattern can match the name of a single index or include a wildcard (*) to match multiple indices.

You'll create patterns for the YouTube data set, which has an index named global

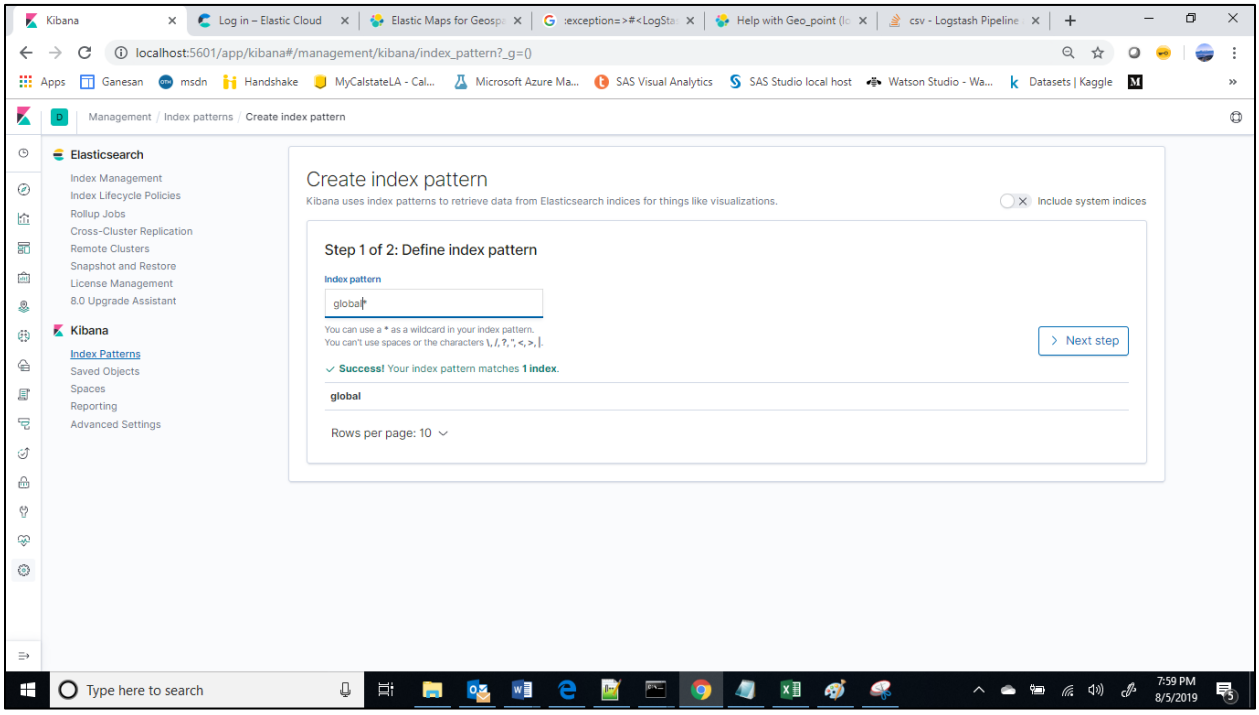
This data sets contain time-series data.

i) In Kibana, open Management, and then click Index Patterns.



ii) If this is your first index pattern, the Create index pattern page opens automatically. Otherwise, click Create index pattern in the upper left.

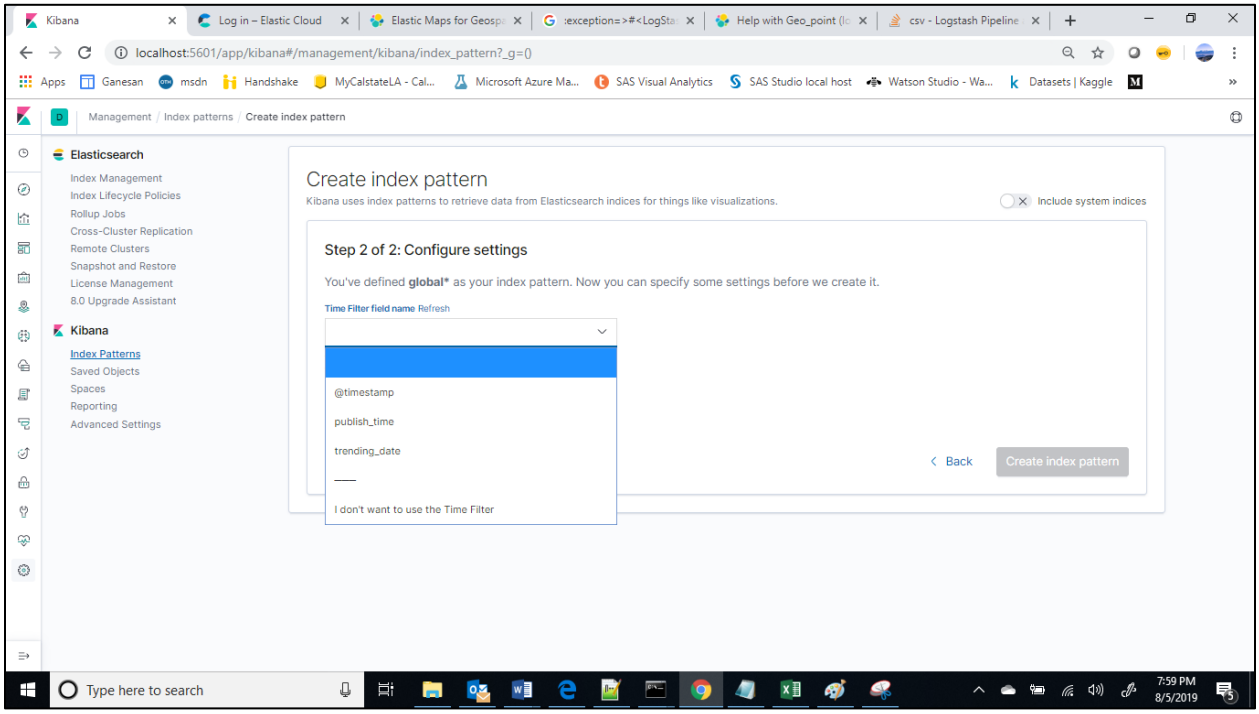
iii) Enter global* in the Index pattern field, which should show “Success!”. If not, your data uploading has not worked.



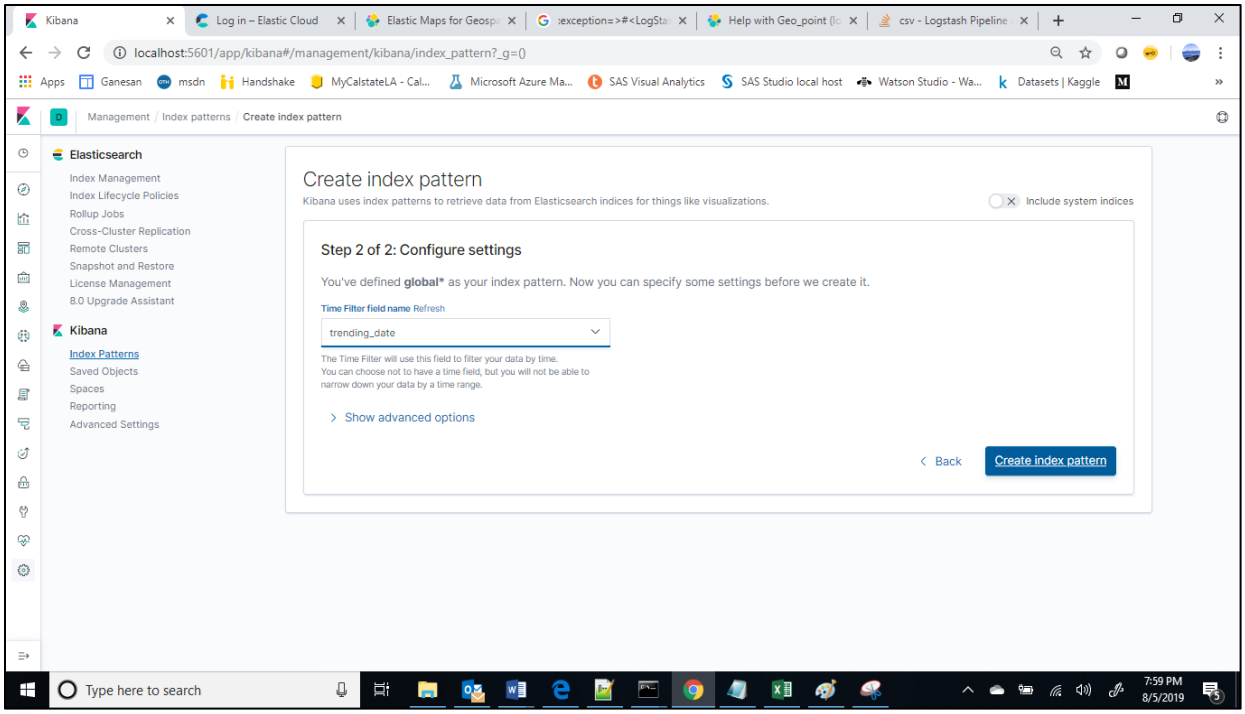
iv) Click Next step.

v) In Configure settings, click Create index pattern. For this pattern, you don't need to configure any settings.

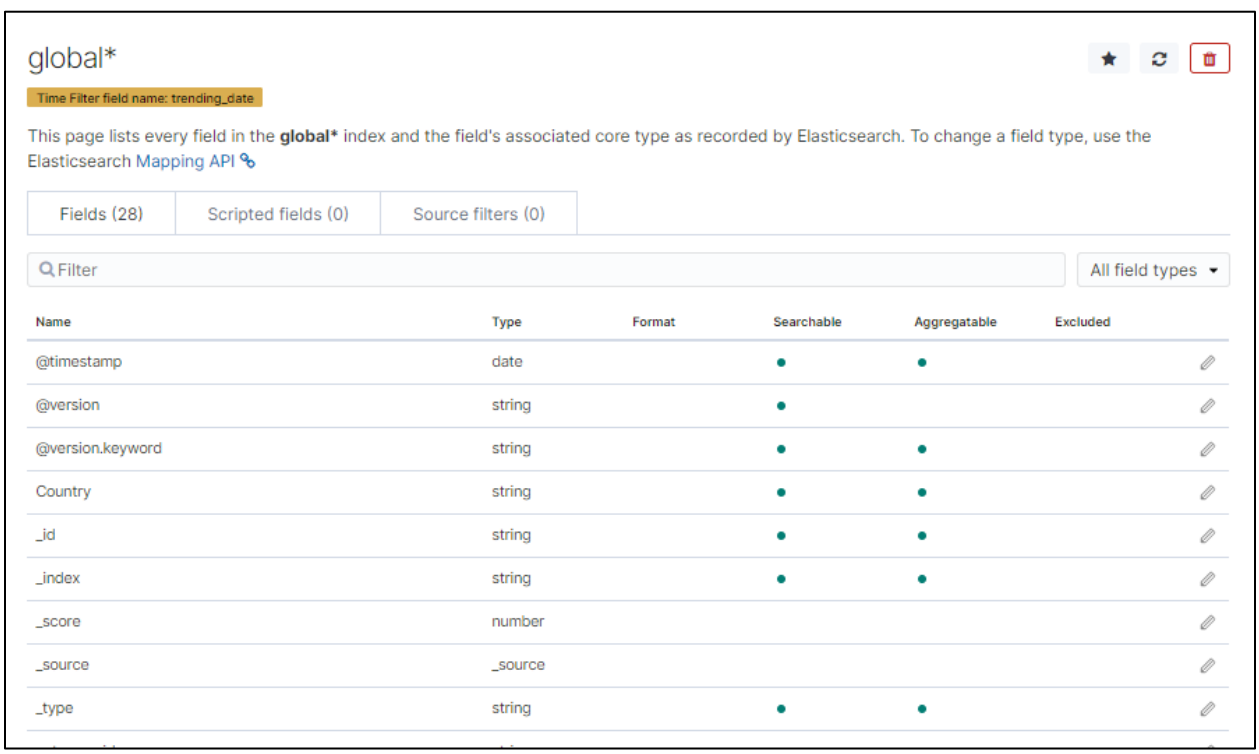
vi) This data set contains time-series data. In Configure settings, select @timestamp in the Time Filter field name dropdown menu. Click create index pattern.



vii) Click create index pattern.



viii) You will see that Timestamp field is indexed as Time/date field. It will show the following fields that are indexed with the data types you defined at Mapping:

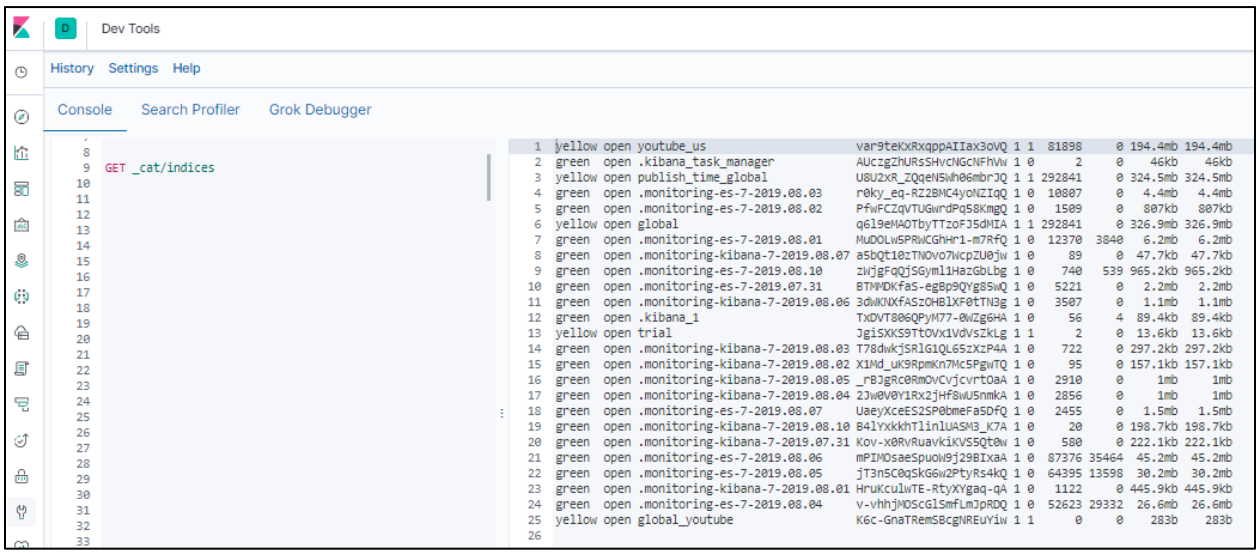


Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		•	•	
@version	string		•		
@version.keyword	string		•	•	
Country	string		•	•	
_id	string		•	•	
_index	string		•	•	
_score	number				
_source	_source				
_type	string		•	•	

ix) Similarly create one more index to define the 2nd timeline that YouTube dataset has i.e. Publish_time with the index pattern name as publish_time_global.

x) When you define an index pattern, the indices that match that pattern must exist in Elasticsearch and they must contain data.

xi) To check which indices are available, go to Dev Tools > Console and enter GET _cat/indices.



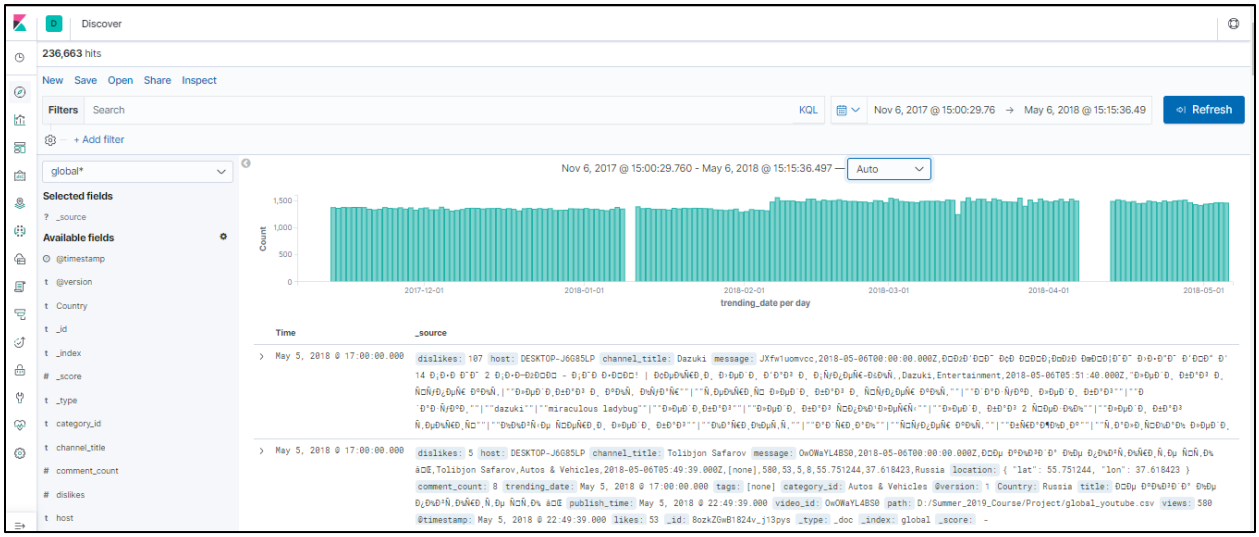
Step 5: Discovering your data

Using the Discover application, you can enter an Elasticsearch query to search your data and filter the results.

i) Open Discover. The current index pattern appears below the filter bar, in this case global*. You might need to click New in the menu bar to refresh the data.

ii) In the Time filter field, enter the following dates and select Update/Refresh button: Nov 6, 2017 – May 6, 2018.

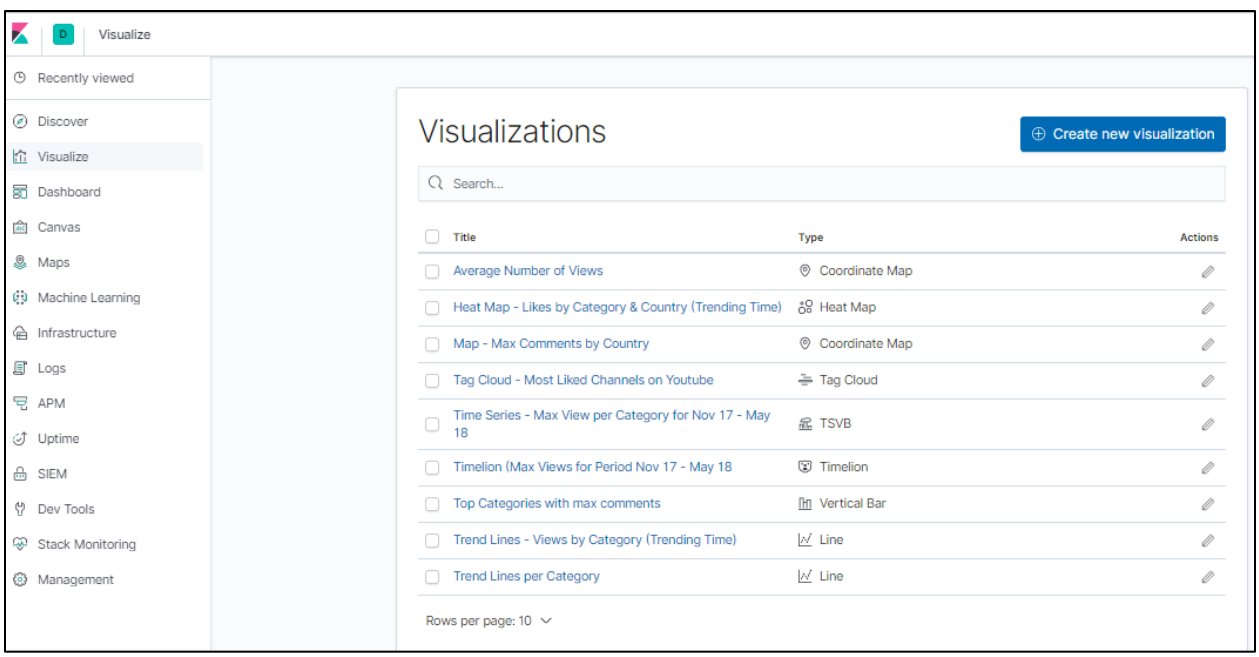
iii) The search returns all YouTube documents between the above given dates.



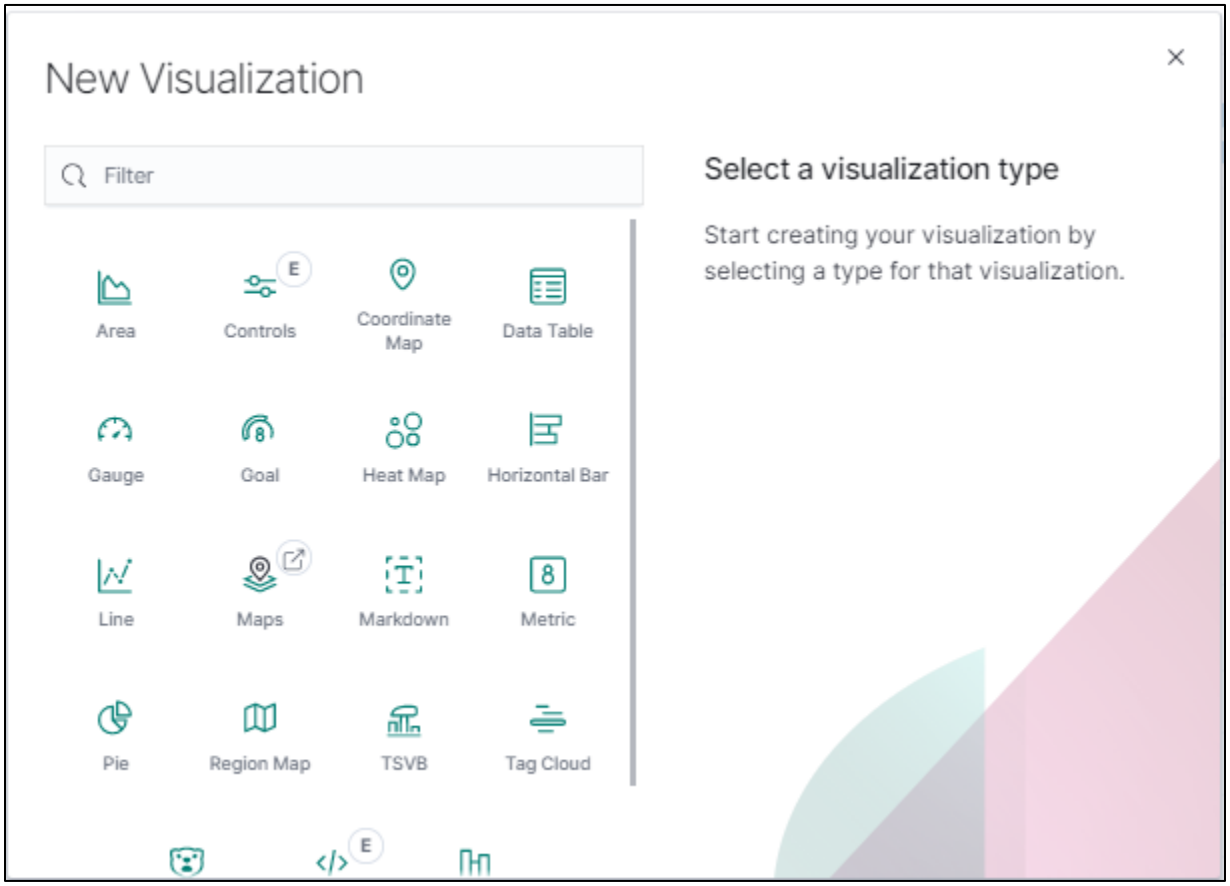
Step 5: Visualizing your data

In the Visualize application, you can shape your data using a variety of charts, tables, and maps, and more. You'll create six visualizations in the lab: a timelion chart, time- series chart, trend line chart, tag cloud, heat map and geo spatial chart.

i) In Kibana page, open Visualize.



ii) Click Create a visualization or the + button. You'll see all the visualization types in Kibana.



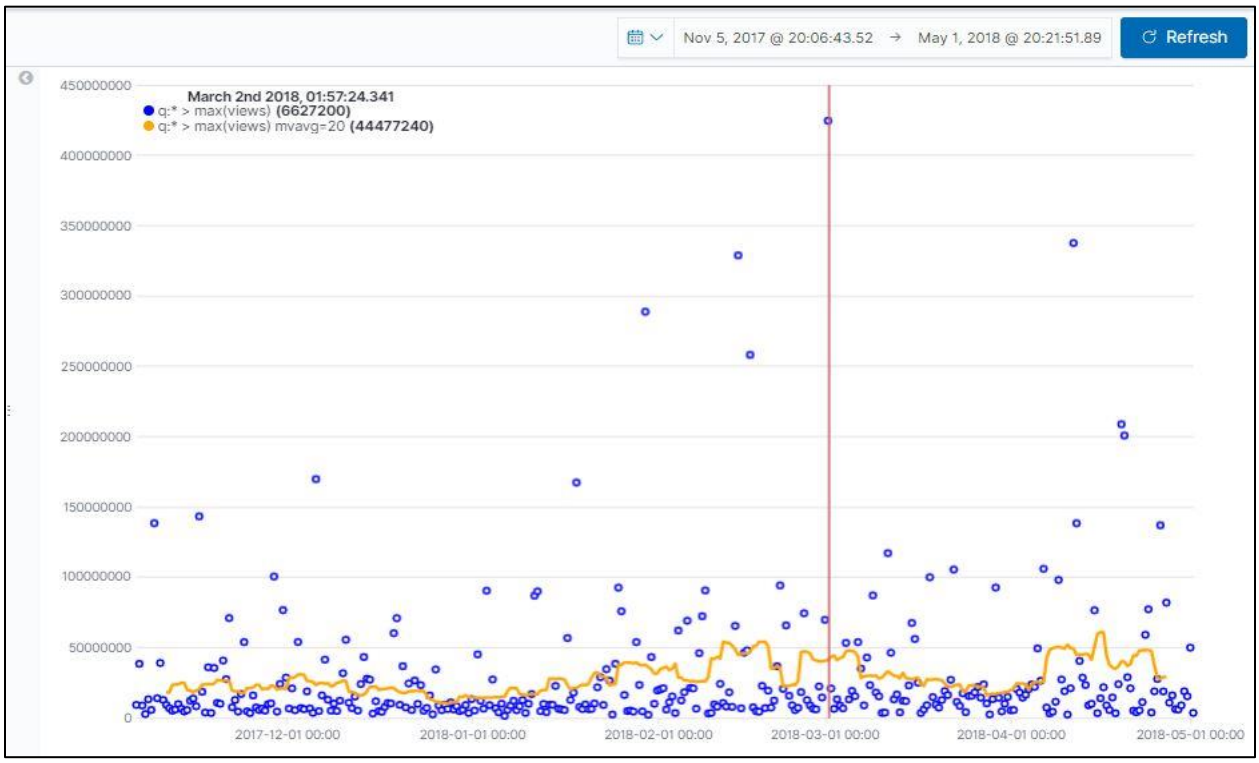
iii) In New Search, select the global* index pattern. You'll use Timelion chart to gain insight into the time line of the YouTube views.

Timelion

Timelion is a time series data visualizer that enables you to combine totally independent data sources within a single visualization. It's driven by a simple expression language you use to retrieve time series data, perform calculations to tease out the answers to complex questions, and visualize the results. We would now visualize to see the time series for maximum Views of YouTube users over the time period – Nov 2017 to May 2018

Code:

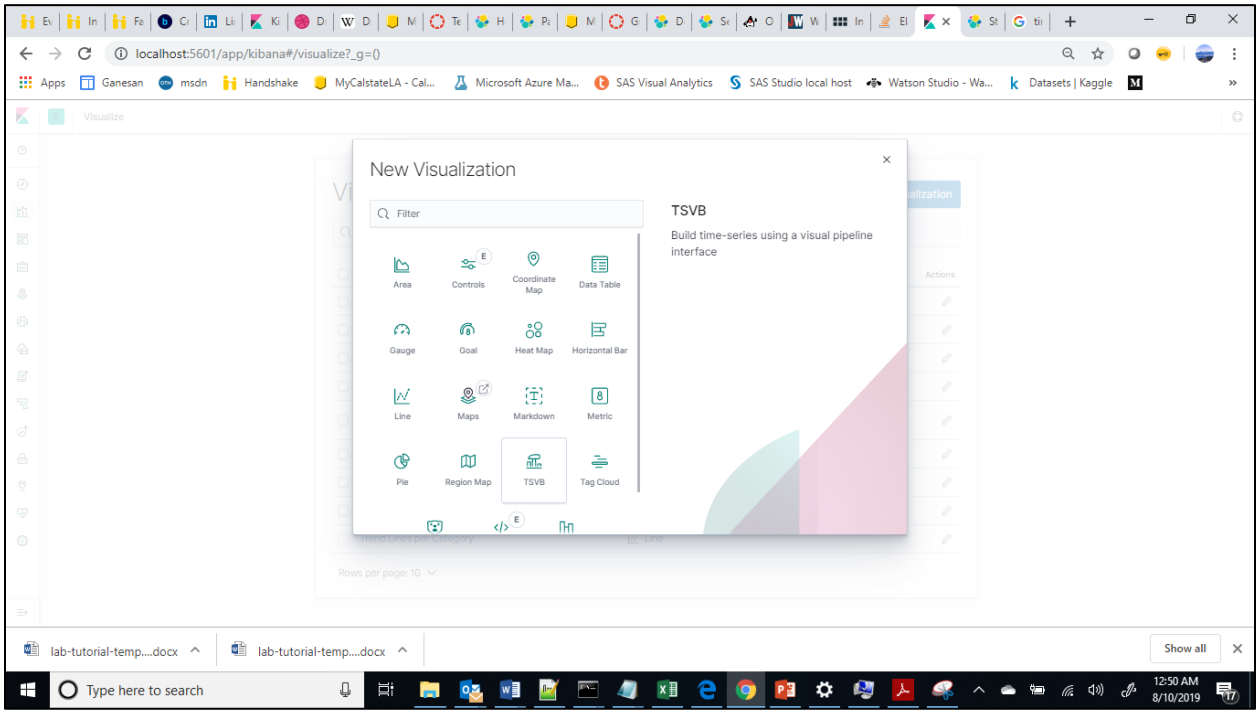
```
.es(index=publish_time_global*,metric=max:views).points().color(blue),
.es(index=publish_time_global*,metric=max:views).mvavg(20).color(orang
e)
```



The above graph shows the time series graph analysis of the pattern of YouTube videos viewed by public on a monthly basis. With the help of this analysis we can understand the pattern as to when the audience are viewing videos and how many users are actively viewing YouTube channels in a particular time frame. From the graph it is evident that maximum viewership was clocked by YouTube in the months of March and April.

TSVB

TSVB is a time series data visualizer that allows you to use the full power of the Elasticsearch aggregation framework.



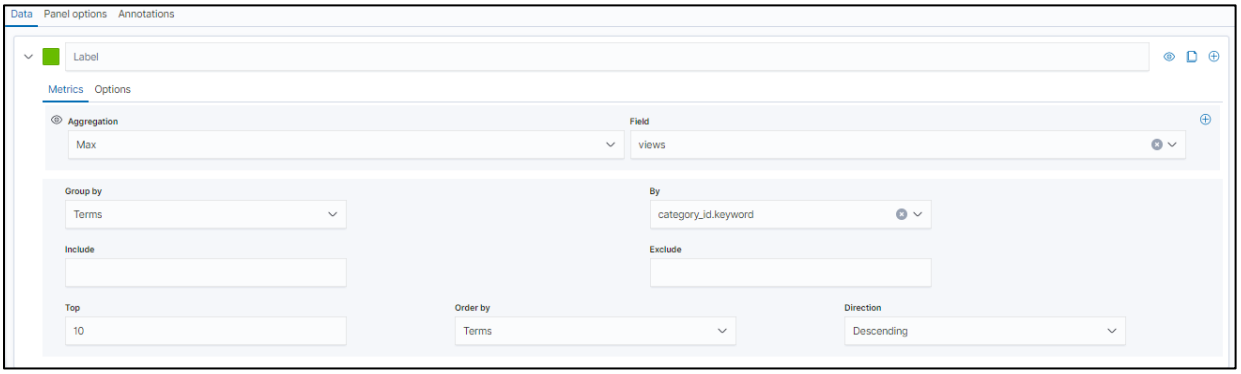
Aggregation

The aggregations framework helps provide aggregated data based on a search query. It is based on simple building blocks called aggregations, that can be composed in order to build complex summaries of the data.

Settings

- Aggregations: Max

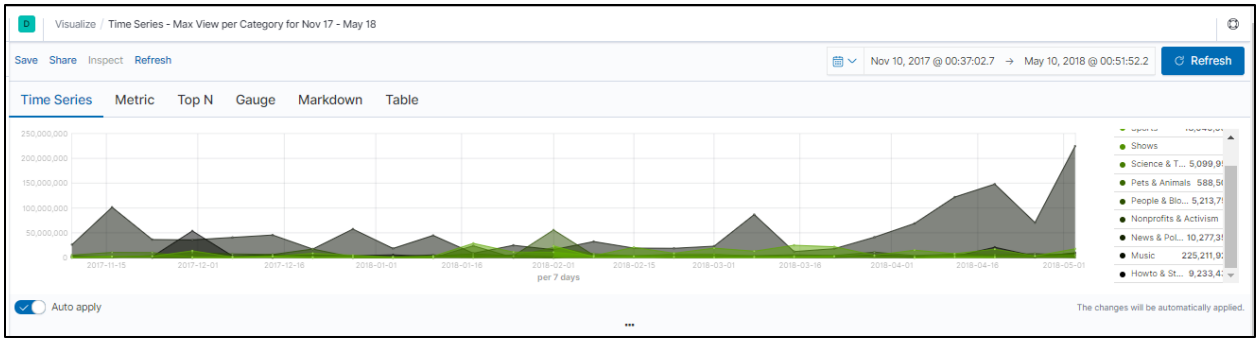
- Field: views
- Group: terms
- By: category_id.keyword
- Top: 10
- Order by: Terms
- Direction: Descending



- Panel Options: Interval → 7d

This will represent out data on weekly basis.

Visualization

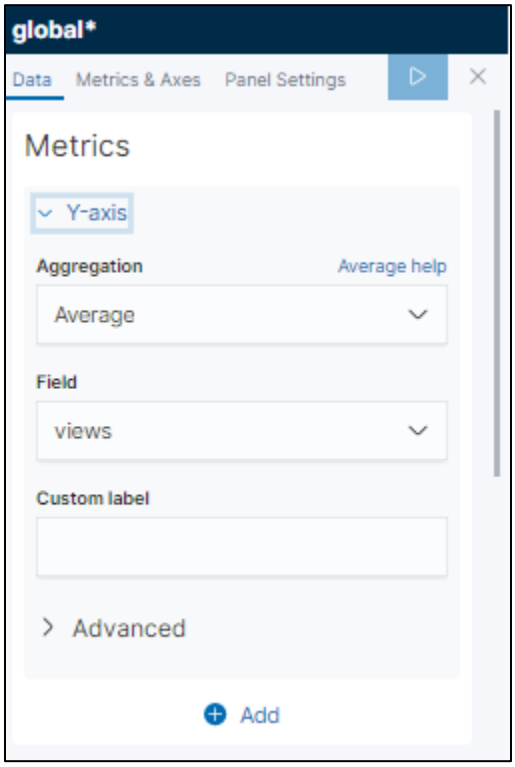


Trend Lines

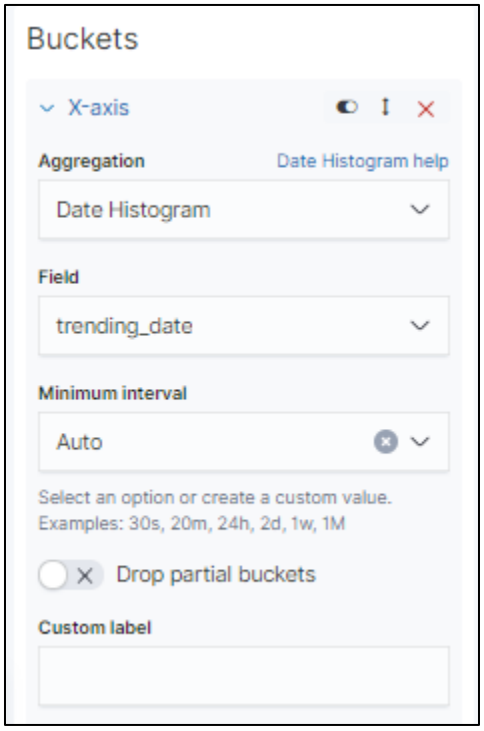
Trend lines are used to predict the continuation of a certain trend of a variable. It also helps to identify the correlation between two variables by observing the trend in both simultaneously.

Settings:

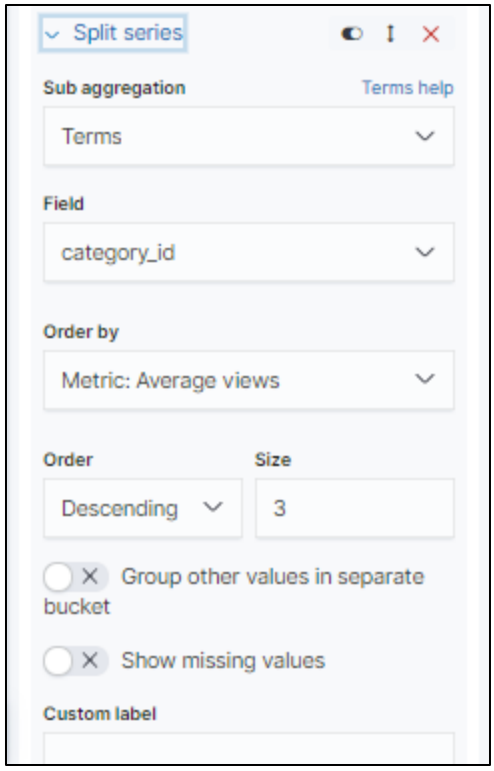
Metric



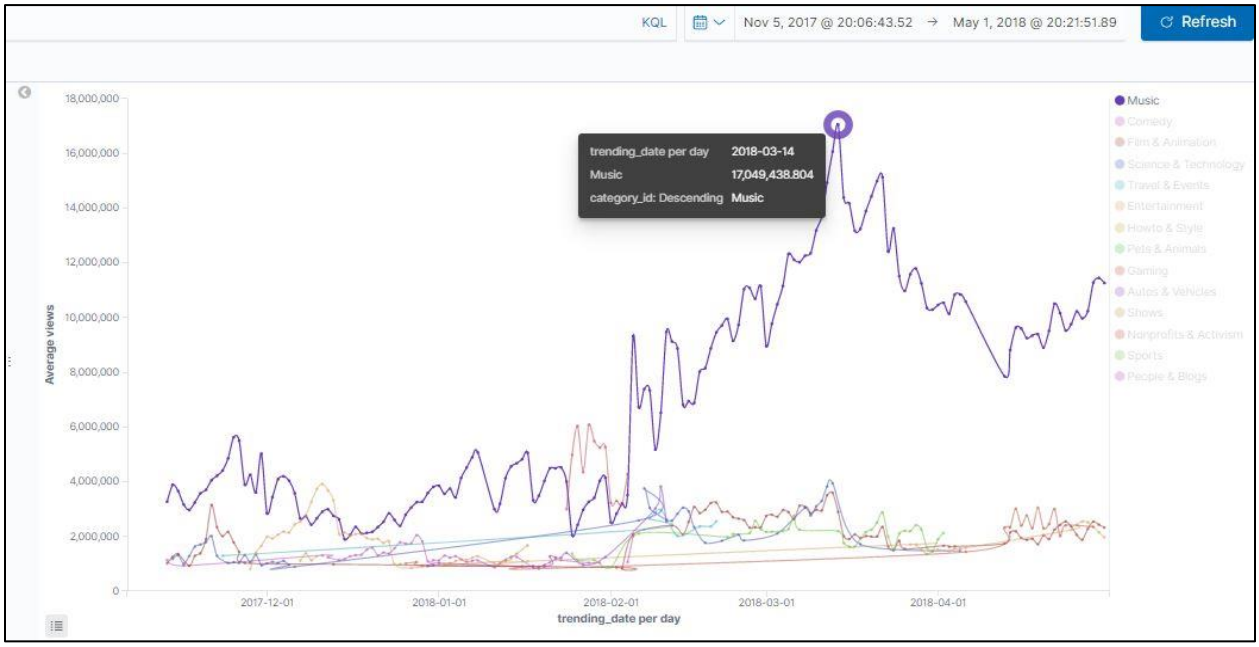
Buckets



Split Series



Visualization

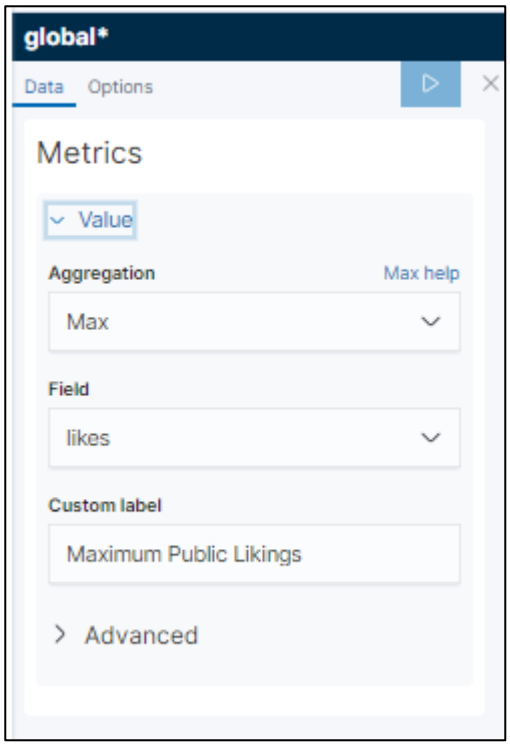


This graphical representation gives insight about the categories that are trending daily. The above analysis depicts that Music is the most preferred YouTube category which is viewed daily by many audiences.

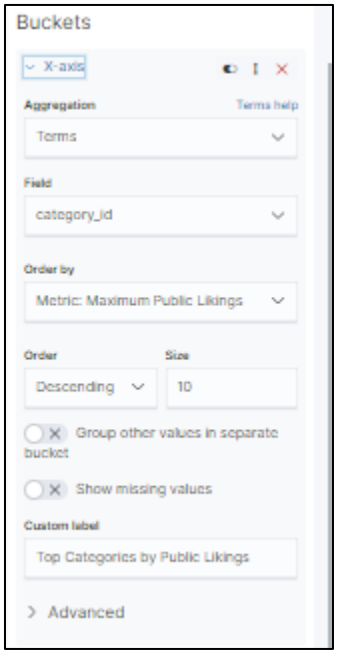
Heat Map

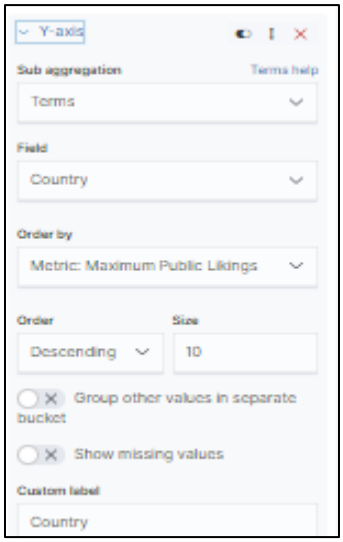
A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. The color for each matrix position is determined by the metrics aggregation.

Settings:



Bucket





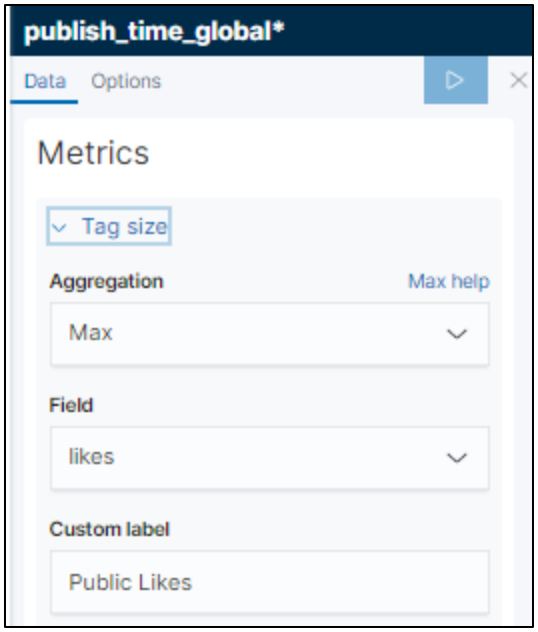
Visualization

We have also analyzed the country-wise preference of a YouTube category using the heat map. From the above analysis countries such as Europe and United States have a greater viewership for the category of Music which is followed by Entertainment.

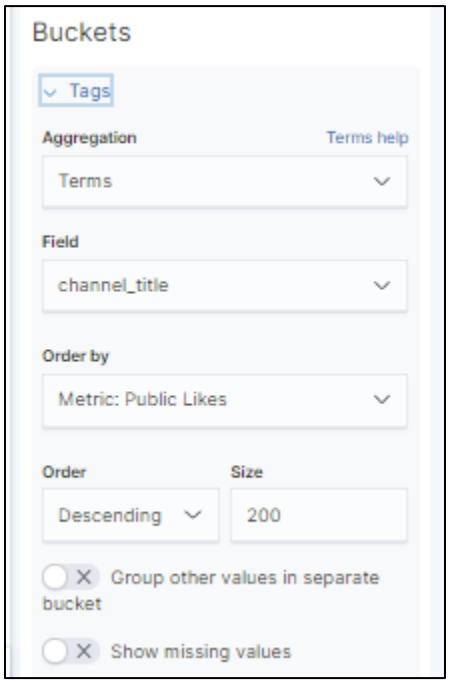
Tag Cloud

A tag cloud visualization is a visual representation of text data, typically used to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. The font size for each word is determined by the metrics aggregation.

Settings:



Buckets



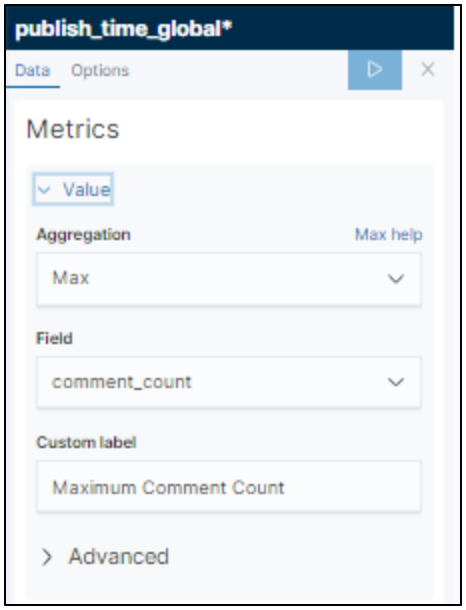
Visualization

The above pictorial representation shows the frequency of liking a YouTube Channels. The greater the size means most favorite. It can be observed that Childish Gambino Vevo is the most favorite YouTube channel among the users.

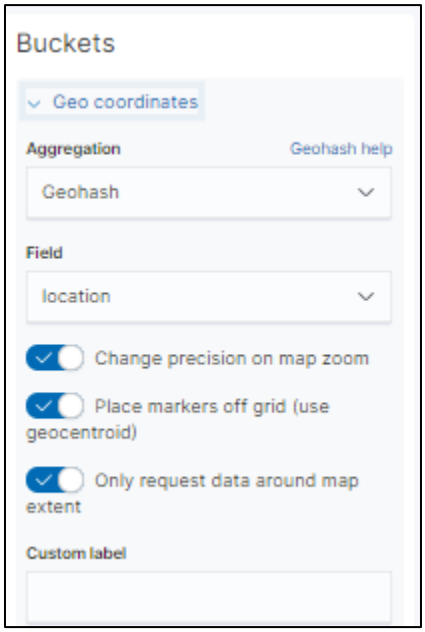
Coordinate Map

A coordinate map displays a geographic area overlaid with circles keyed to the data determined by the buckets you specify.

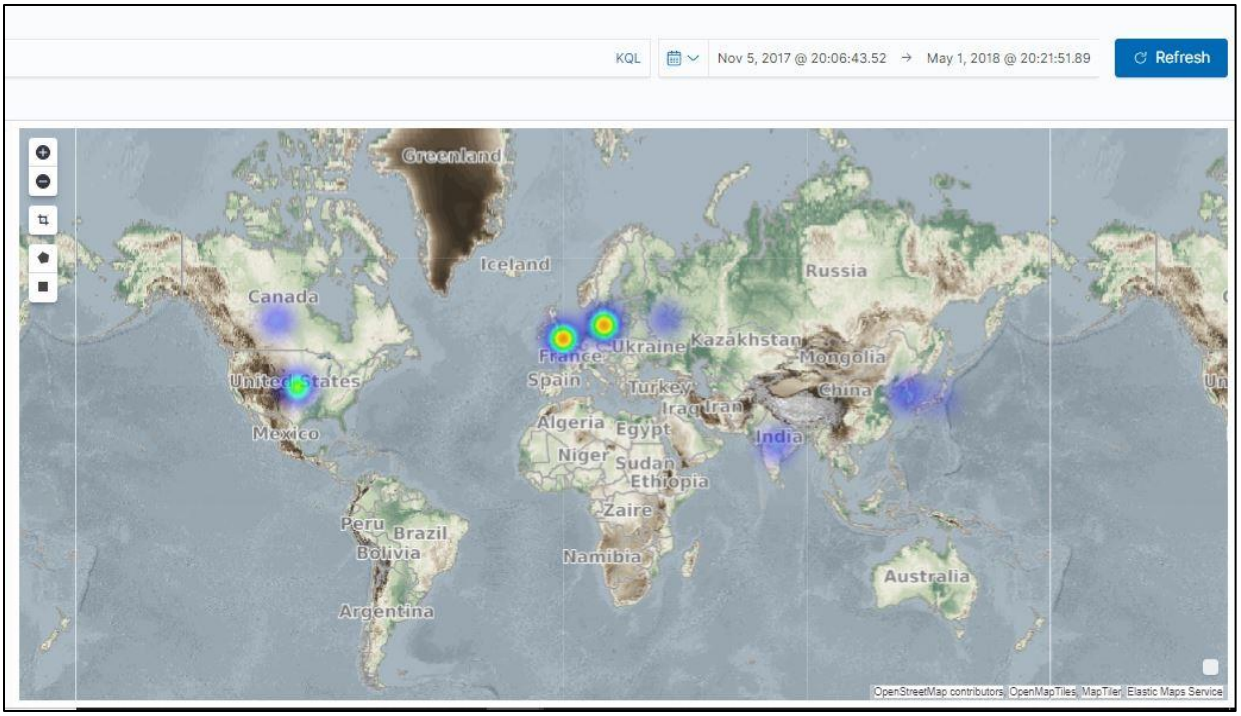
Settings:



Bucket



Visualization



The above geographical heat map represents the number of comments per location. From this representation we can find out the location having maximum number of users who comment on YouTube videos. And it is can be seen that in countries like United Kingdom, France and Germany the users have commented a greater number of times on the YouTube videos. Whereas the trend gets weaker as we come across countries like India and Japan.

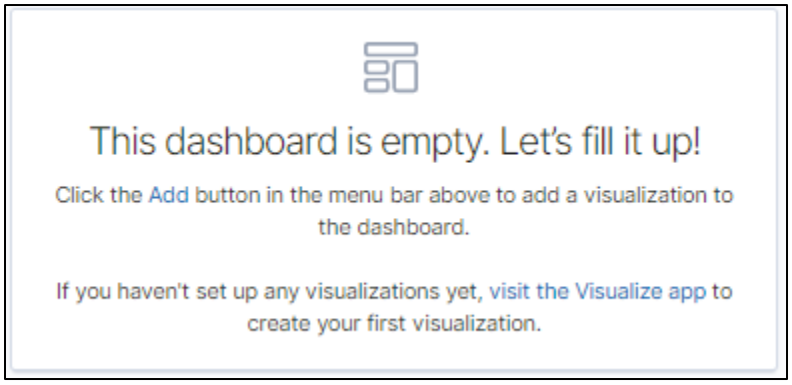
Dashboard

A Kibana dashboard is a collection of visualizations, searches, and maps, typically in real-time. Dashboards provide at-a-glance insights into your data and enable you to drill down into details.

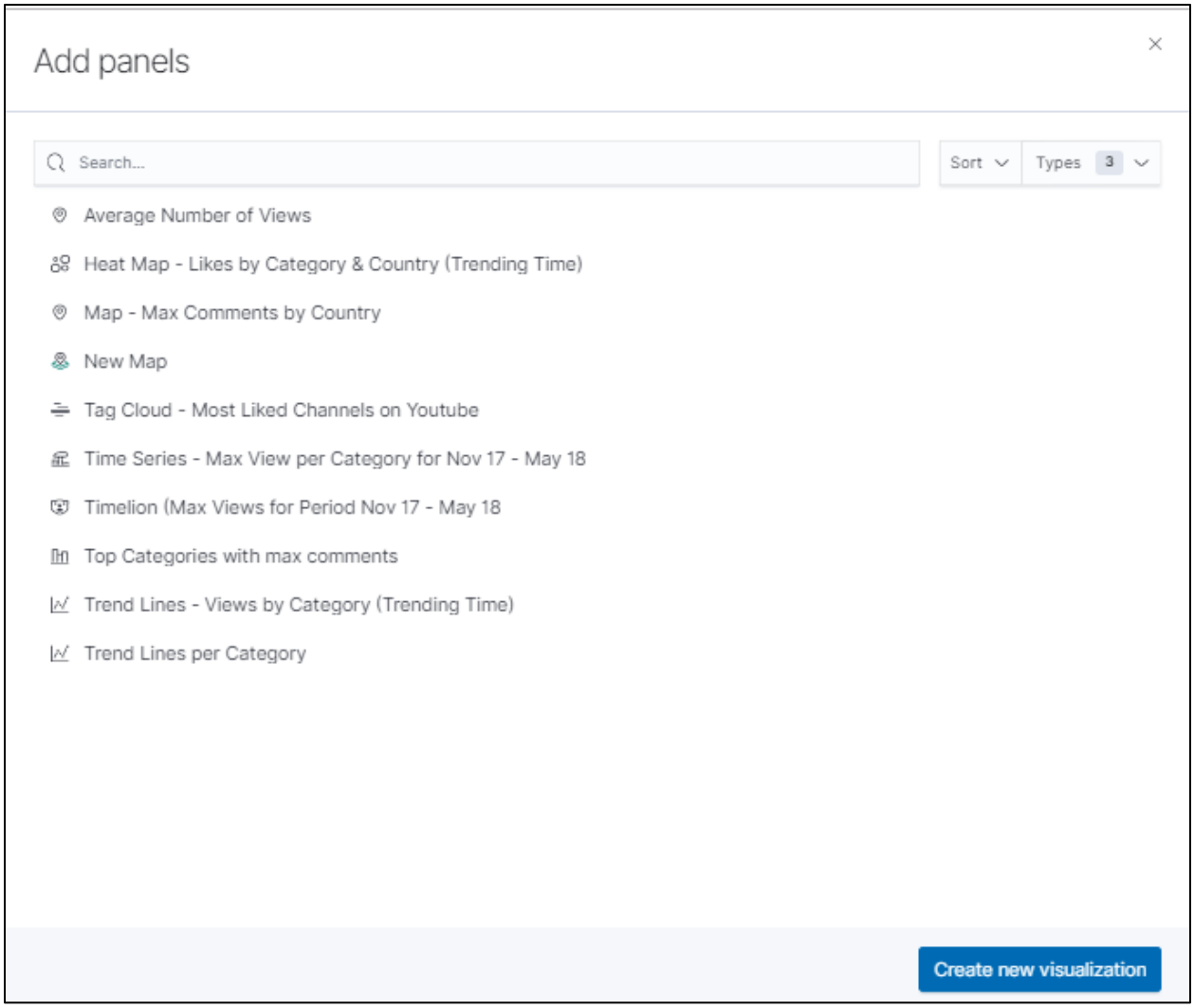
1. To start working with dashboards, click Dashboard in the side navigation.
2. In the side navigation, click Dashboard.

3. Click Create new dashboard.

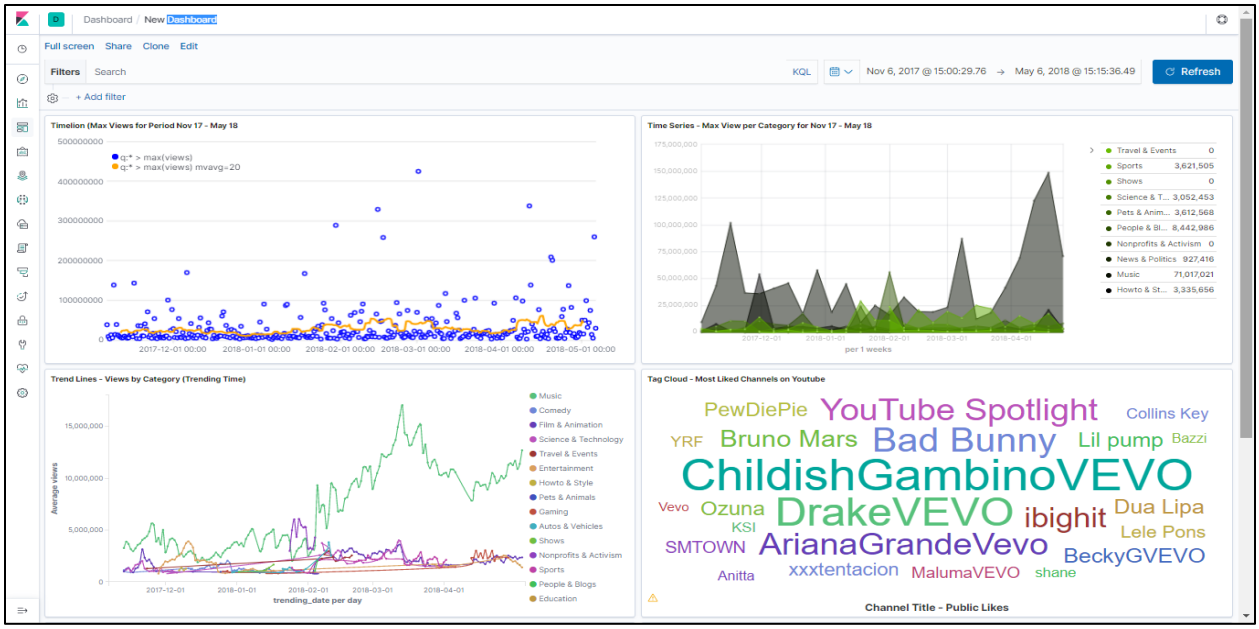
4. Click Add.

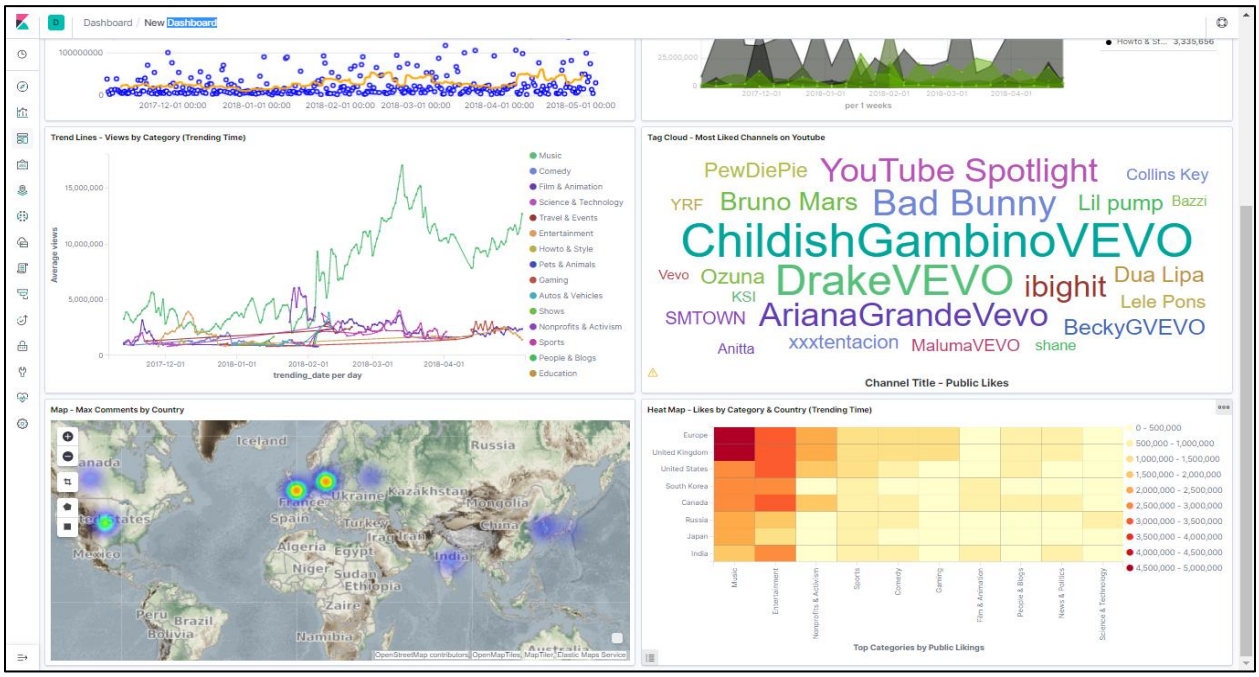


5. To add a visualization, select its name from the list of visualizations or click Add new visualization to create one. If you have many visualizations, you can filter the list.



6. To add a saved search, click the Saved Search tab, and then select a name from the list.
7. When you're finished adding and arranging the dashboard content, go to the menu bar, click Save, and enter a name. Optionally, you can store the time period specified in the time filter by selecting Store time with dashboard.


Dashboard / New Dashboard
Full screen Share Clone Edit
Filters Search
KQL
Nov 6, 2017 @ 15:00:29.76 → May 6, 2018 @ 15:15:36.49
Refresh
+ Add filter
Timelion (Max Views for Period Nov 17 - May 18
q:* > max(views)
q:* > max(views) mvavg=20
Time Series - Max View per Category for Nov 17 - May 18
Travel & Events 0
Sports 3,621,505
Shows 0
Science & T... 3,052,453
Pets & Anim... 3,612,568
People & Bl... 8,442,986
Nonprofits & Activism 0
News & Politics 927,416
Music 71,017,021
Howto & St... 3,335,656
per 1 weeks
Trend Lines - Views by Category (Trending Time)
Average views
trending_date per day
Music
Comedy
Film & Animation
Science & Technology
Travel & Events
Entertainment
Howto & Style
Pets & Animals
Gaming
Autos & Vehicles
Shows
Nonprofits & Activism
Sports
People & Blogs
Education
Tag Cloud - Most Liked Channels on Youtube
PewDiePie
YouTube Spotlight
Collins Key
YRF
Bruno Mars
Bad Bunny
Lil pump
Bazzi
ChildishGambinoVEVO
Vevo
Ozuna
DrakeVEVO
ibighit
Dua Lipa
KSI
Lele Pons
SMTOWN
ArianaGrandeVevo
BeckyGVEVO
Anitta
xxxtentacion
MalumaVEVO
shane
Channel Title - Public Likes


Dashboard / New Dashboard
Trend Lines - Views by Category (Trending Time)
Average views
trending_date per day
Tag Cloud - Most Liked Channels on Youtube
PewDiePie
YouTube Spotlight
Collins Key
YRF
Bruno Mars
Bad Bunny
Lil pump
Bazzi
ChildishGambinoVEVO
Vevo
Ozuna
DrakeVEVO
ibighit
Dua Lipa
KSI
Lele Pons
SMTOWN
ArianaGrandeVevo
BeckyGVEVO
Anitta
xxxtentacion
MalumaVEVO
shane
Channel Title - Public Likes
Map - Max Comments by Country
Iceland
Russia
Canada
France
Ukraine
Kazakhstan
Mongolia
Spain
Turkey
China
Mexico
Algeria
Egypt
India
Niger
Sudan
Ethiopia
Zaire
Peru
Brazil
Bolivia
Namibia
Heat Map - Likes by Category & Country (Trending Time)
Europe
United Kingdom
United States
South Korea
Canada
Russia
Japan
India
Music
Entertainment
Nonprofits & Activism
Sports
Comedy
Gaming
Film & Animation
People & Blogs
News & Politics
Science & Technology
0 - 500,000
500,000 - 1,000,000
1,000,000 - 1,500,000
1,500,000 - 2,000,000
2,000,000 - 2,500,000
2,500,000 - 3,000,000
3,000,000 - 3,500,000
3,500,000 - 4,000,000
4,000,000 - 4,500,000
4,500,000 - 5,000,000
Top Categories by Public Likings

References

1. URL of Data Source, https://www.kaggle.com/datasnaek/youtube-new
2. URL of your Github : https://github.com/tanvigawade/ElasticSearch-Kibana.git
3. https://www.elastic.co/guide/en/kibana/current/tutorial-visualizing.html
4. https://www.elastic.co/blog/getting-started-with-hosted-elasticsearch-and-a-sample-dataset
5. https://www.elastic.co/guide/en/kibana/current/tutorial-load-dataset.html
6. https://www.elastic.co/guide/en/kibana/current/tilemap.html
7. https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html
8. https://discuss.elastic.co/t/how-to-import-data-to-elasticsearch/58100/2
9. https://www.elastic.co/guide/en/logstash/current/advanced-pipeline.html