

# Big Data Analytics Report

**Project 1: Real-time Spark Processing: (Kappa):** Set up an Apache Kafka + Spark Streaming + Hadoop + Docker pipeline (use PySpark if needed) and demonstrate a variety of real-time analytical queries. You can also use SparkSql for SQL-based queries. Requires Front-End.

By Abeera Tariq – 13170 (2<sup>nd</sup> June 2022)

## DATA PREPARATION

## SET UP POSTMAN

This will allow testing of API.



## Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

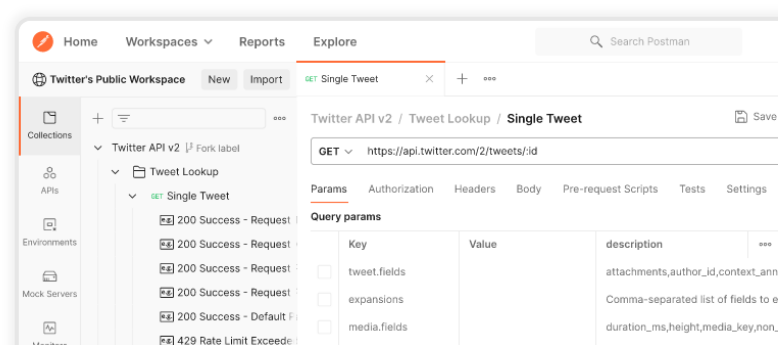


The ever-improving Postman app (a new release every week) gives you a full-featured Postman experience.

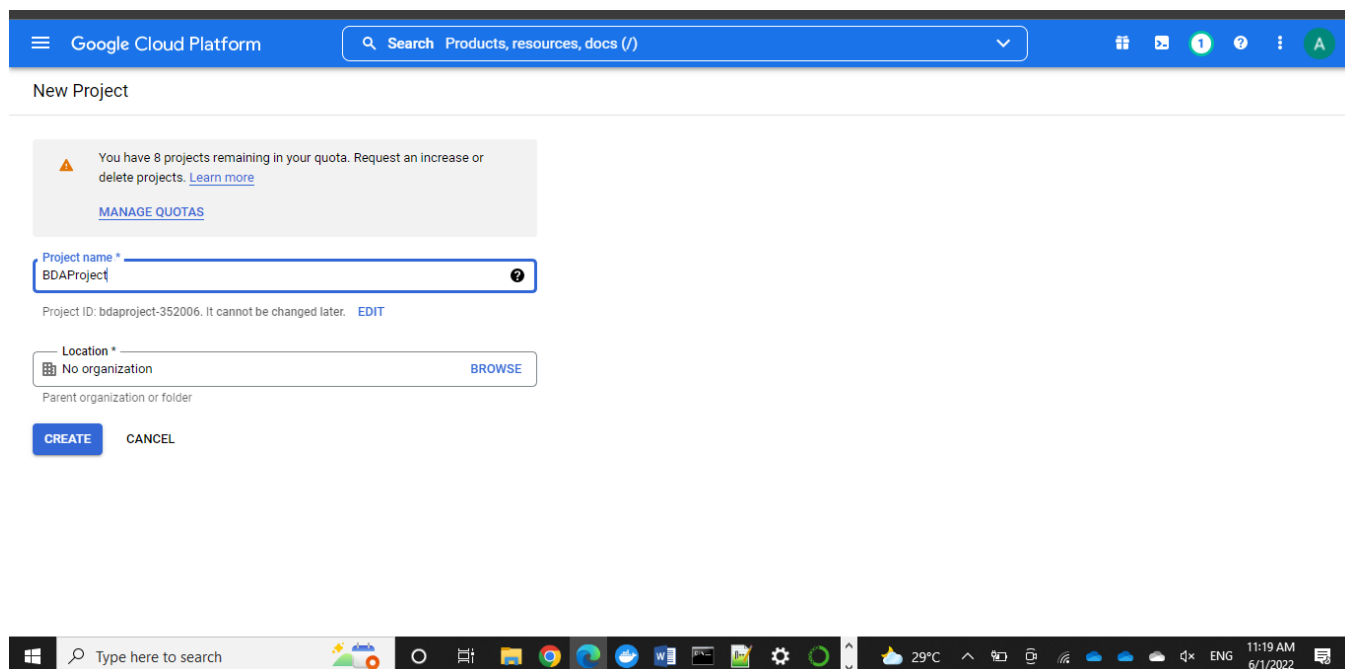


By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Version 9.19.3](#) - [Release Notes](#) - [Product Roadmap](#)



## GET API KEY



API key: AI\*\*\*\*\*

(hidden from the project)

To create your own Youtube Data API key, follow the steps as in [YouTube Data API Overview | Google Developers](#).

Once you have the key, create a file named api\_key.txt containing the key and follow the steps as below.

Scrapped data from Youtube using API.

```
(bda) E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Analytics\Project\my-document-streaming-project-main\my-document-streaming-project-main\client>python get-data.py
Writing US data to file...

(bda) E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Analytics\Project\my-document-streaming-project-main\my-document-streaming-project-main\client>python transformer.py

(bda) E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Analytics\Project\my-document-streaming-project-main\my-document-streaming-project-main\client>_
```

From the conda environment, execute the script get-data.py to fetch datafeeds from Youtube and run transformer.py to get the data as a text file in json format.

Here on, client/output.txt contains a json of Youtube trending videos data of US. This can be modified to have more countries with changes in the country\_codes file.

# DATASET

**Use Case:** Social Media Analysis – YOUTUBE Trending videos

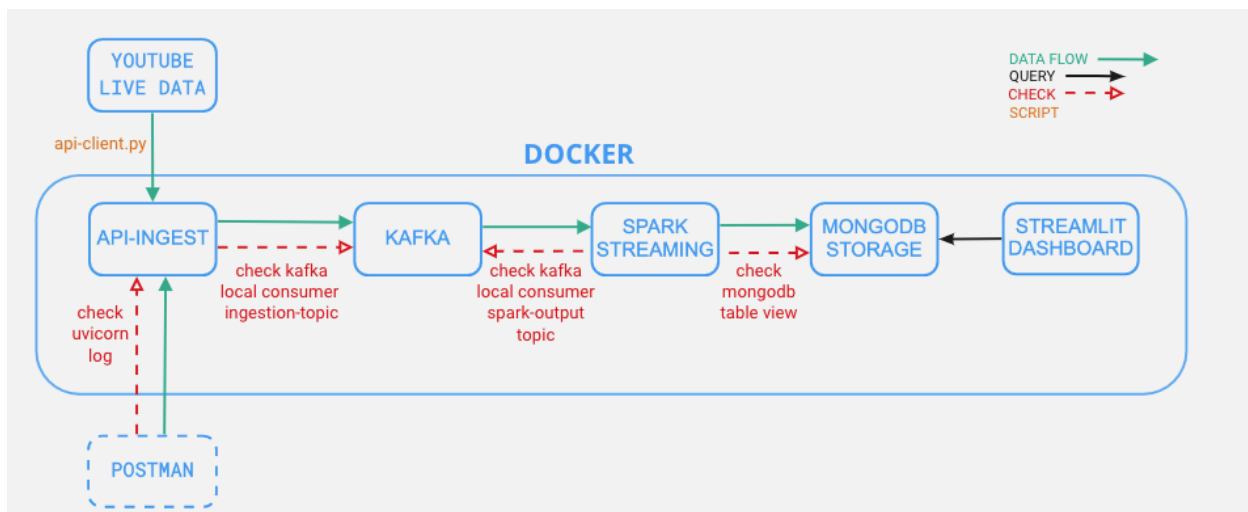
## Sample Data:

```
{
  "video_id": "5XWEVoI40sE",
  "title": "THE INSIDE OUTTAKES - Bo Burnham (4K)",
  "publish_time": "2022-05- 31T00:55:11Z",
  "channel_title": "UC81hVml5eEBIt3s3HQpJd_w",
  "channelTitle": "boburnham",
  "category_id": 23,
  "trending_date": "22.01.06",
  "views": 2235673,
  "likes": 0,
  "dislikes": 0,
  "comment_count": 14033
}
```

## Column Names and Data types:

```
video_id: str
trending_date: str
title: str
channel_title: str
category_id: int
publish_time: str
views: int
likes: int
dislikes: int
comment_count: int
```

# Pipeline Schema



The application uses simple API to ingest YouTube trending video data, then use Kafka, Apache Spark and MongoDB for storage to create a streaming pipeline and Streamlit is utilized for viewing and creating dashboards.

## API Preparation

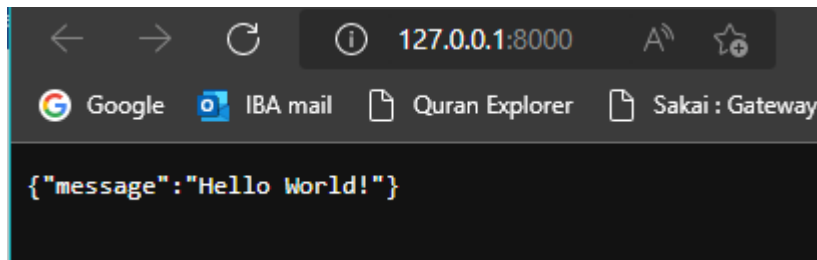
- Import the conda yaml file to have the required environment needed for this application.
- 2 methods are created.
  - GET: display message to ensure API is running
  - POST: convert trending data column to a standard date and post the record as a json string to Kafka

Go to my-document-streaming-project-main\API-Ingest\app

- In the conda terminal, run  
uvicorn main:app --reload

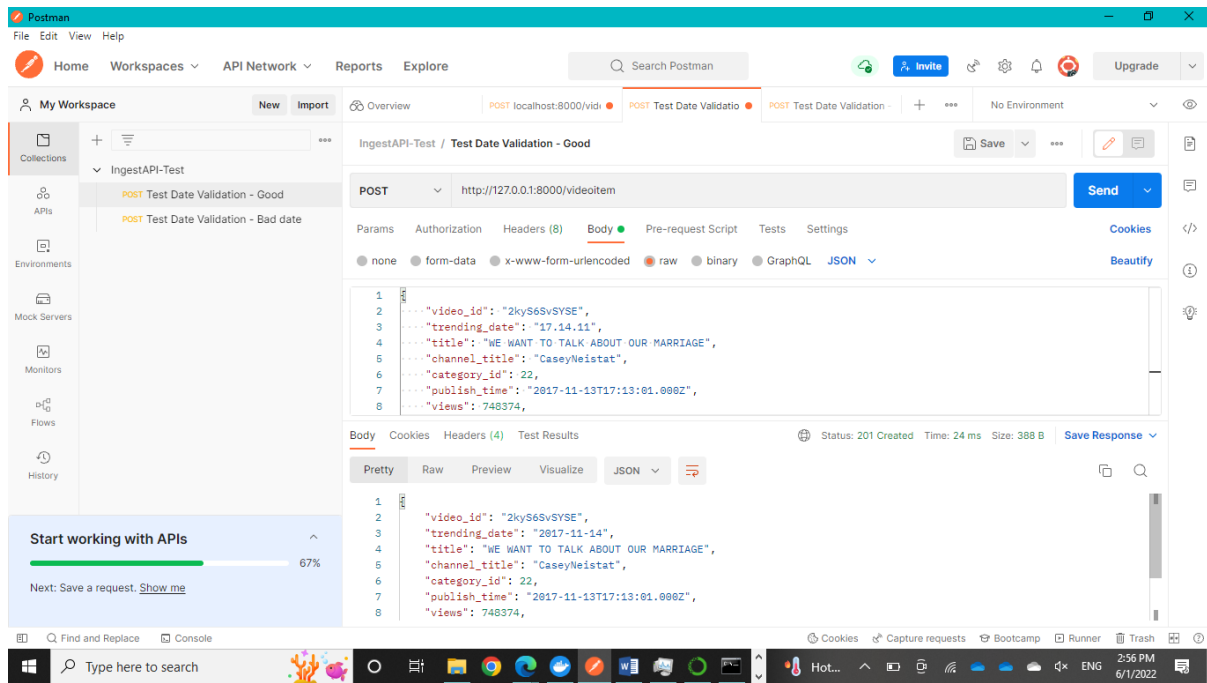
```
(bda) E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Analytics\Project\my-document-streaming-project-main\my-document-s
treaming-project-main\API-Ingest\app>uvicorn main:app --reload
+32mINFO+[0m: Will watch for changes in these directories: ['E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Ana
lytics\Project\my-document-streaming-project-main\my-document-streaming-project-main\API-Ingest\app']
+32mINFO+[0m: Uvicorn running on +1mhhttp://127.0.0.1:8000+[0m (Press CTRL+C to quit)
+32mINFO+[0m: Started reloader process [+36m+[1m36084+[0m using [+36m+[1mwatchgod+[0m
+33mWARNING+[0m: The --reload flag should not be used in production on Windows.
+32mINFO+[0m: Started server process [+36m36044+[0m]
+32mINFO+[0m: Waiting for application startup.
+32mINFO+[0m: Application startup complete.
```

At <http://127.0.0.1:8000> , the following page should be visible.



### Testing with Postman

- Launch Postman
- Import my-document-streaming-project/API-Ingest/Postman/IngestAPI-Test.postman\_collection.json



- Ensure that line 64 in `\API-Ingest\app\main.py` is commented out for testing with test data. The request should be received.

```

Message received
Found the first timestamp: 2017-11-14 00:00:00
New trending date: 2017-11-14
{"video_id": "2kyS6SvSYSE", "trending_date": "2017-11-14", "title": "WE WANT TO TALK ABOUT OUR MARRIAGE", "channel_title": "CaseyNeistat", "category_id": 22, "publish_time": "2017-11-13T17:13:01.000Z", "views": 748374, "likes": 57527, "dislikes": 2966, "comment_count": 15954}
+ [32mINFO+ [0m: 127.0.0.1:54276 - "[1mPOST /videoitem HTTP/1.1+[0m" + [32m201 Created+[0m

```

## Kafka Container

### RUN in a terminal

```
docker-compose -f docker-compose-kafka.yml up
```

### Setting Kafka Topics

In the kafka terminal, execute

```
cd /opt/bitnami/kafka/bin
```

look for existing topics

```
./kafka-topics.sh --list --bootstrap-server localhost:9092
```

Create the ingestion topic

```
./kafka-topics.sh --create --topic ingestion-topic --bootstrap-server localhost:9092
```

```

docker exec -it bd5232ce78bfd8e59fd0b6699a426ef6ec5364153e6b...
$ cd /opt/bitnami/kafka/bin
$ ./kafka-topics.sh --list --bootstrap-server localhost:9092

$ ./kafka-topics.sh --create --topic ingestion-topic --bootstrap-server localhost:9092
Created topic ingestion-topic.
$

```

# Local Consumer

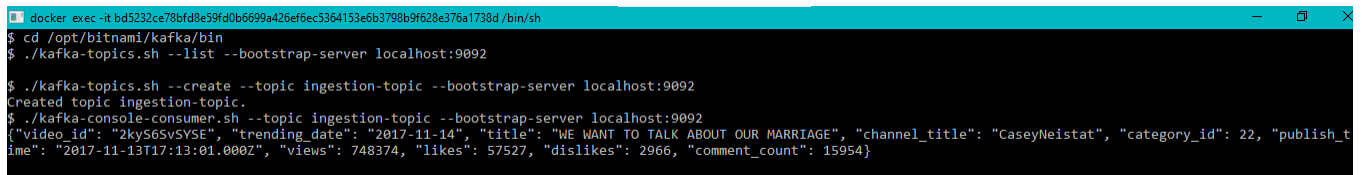
## Create Local Consumer

In the kafka terminal:

```
./kafka-console-consumer.sh --topic ingestion-topic --bootstrap-server localhost:9092
```

Meanwhile, uncomment line 64 in main.py and switch `bootstrap_servers='localhost:9093'`

Test with Postman: send the same record and see if both Postman and local consumer receive it successfully



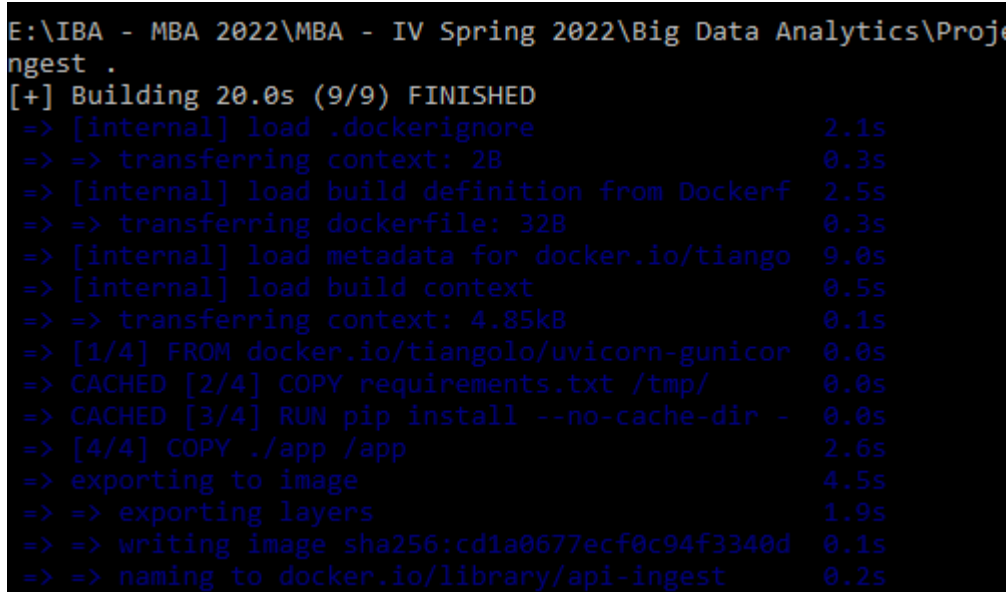
```
docker exec -it bd5232ce78bfd8e59fd0b6699a426ef6ec5364153e6b3798b9f628e376a1738d /bin/sh
$ cd /opt/bitnami/kafka/bin
$ ./kafka-topics.sh --list --bootstrap-server localhost:9092
$ ./kafka-topics.sh --create --topic ingestion-topic --bootstrap-server localhost:9092
Created topic ingestion-topic.
$ ./kafka-console-consumer.sh --topic ingestion-topic --bootstrap-server localhost:9092
{"video_id": "2kyS6SvSYSE", "trending_date": "2017-11-14", "title": "WE WANT TO TALK ABOUT OUR MARRIAGE", "channel_title": "CaseyNeistat", "category_id": 22, "publish_time": "2017-11-13T17:13:01.000Z", "views": 748374, "likes": 57527, "dislikes": 2966, "comment_count": 15954}
```

## Connecting with Kafka

With test passed, change `bootstrap_servers='kafka:9092'` in main.py as we are moving from external to client

Create an image for API app, use dockerfile and requirements, run

```
docker build -t api-ingest .
```



```
E:\IBA - MBA 2022\MBA - IV Spring 2022\Big Data Analytics\Project\api-ingest .
[+] Building 20.0s (9/9) FINISHED
=> [internal] load .dockerignore 2.1s
=> => transferring context: 2B 0.3s
=> [internal] load build definition from Dockerfile 2.5s
=> => transferring dockerfile: 32B 0.3s
=> [internal] load metadata for docker.io/tiangolo 9.0s
=> [internal] load build context 0.5s
=> => transferring context: 4.85kB 0.1s
=> [1/4] FROM docker.io/tiangolo/uvicorn-gunicorn 0.0s
=> CACHED [2/4] COPY requirements.txt /tmp/ 0.0s
=> CACHED [3/4] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> [4/4] COPY ./app /app 2.6s
=> exporting to image 4.5s
=> => exporting layers 1.9s
=> => writing image sha256:cd1a0677ecf0c94f3340d 0.1s
=> => naming to docker.io/library/api-ingest 0.2s
```

## Deploy API

Start Kafka, find the name of network with the container

```
docker run --rm --network my-document-streaming-project-main_default -  
-name my-api-ingest -p 81:80 api-ingest
```

```
#!/usr/bin/env bash  
  
# Let the DB start  
sleep 10;  
# Run migrations  
alembic upgrade head  
  
[2022-06-01 10:26:39 +0000] [1] [INFO] Starting gunicorn 20.1.0  
[2022-06-01 10:26:39 +0000] [1] [INFO] Listening at: http://0.0.0.0:80 (1)  
[2022-06-01 10:26:39 +0000] [1] [INFO] Using worker: uvicorn.workers.UvicornWorker  
  
[2022-06-01 10:26:39 +0000] [9] [INFO] Booting worker with pid: 9  
[2022-06-01 10:26:39 +0000] [10] [INFO] Booting worker with pid: 10  
[2022-06-01 10:26:39 +0000] [11] [INFO] Booting worker with pid: 11  
[2022-06-01 10:26:39 +0000] [12] [INFO] Booting worker with pid: 12  
[2022-06-01 10:26:45 +0000] [11] [INFO] Started server process [11]  
[2022-06-01 10:26:45 +0000] [10] [INFO] Started server process [10]  
[2022-06-01 10:26:45 +0000] [12] [INFO] Started server process [12]  
[2022-06-01 10:26:45 +0000] [10] [INFO] Waiting for application startup.  
[2022-06-01 10:26:45 +0000] [12] [INFO] Waiting for application startup.  
[2022-06-01 10:26:45 +0000] [11] [INFO] Waiting for application startup.  
[2022-06-01 10:26:45 +0000] [9] [INFO] Started server process [9]  
[2022-06-01 10:26:45 +0000] [9] [INFO] Waiting for application startup.  
[2022-06-01 10:26:45 +0000] [9] [INFO] Application startup complete.  
[2022-06-01 10:26:45 +0000] [11] [INFO] Application startup complete.  
[2022-06-01 10:26:45 +0000] [12] [INFO] Application startup complete.  
[2022-06-01 10:26:45 +0000] [10] [INFO] Application startup complete.
```

### Test API container with Kafka


Start Local Consumer again


```
./kafka-console-consumer.sh --topic ingestion-topic --bootstrap-server  
localhost:9092
```








Change port number to 81, and send record





## IngestAPI-Test / Test Date Validation - Good



**POST**  http://127.0.0.1:81/videoitem

Params Authorization Headers (8) **Body**  Pre-request Script Tests Settings

 none  form-data  x-www-form-urlencoded  **raw**  binary  GraphQL **JSON** 

```
1 {}
2   .... "video_id": "2kyS6SvSYSE",
3   .... "trending_date": "17.14.11",
4   .... "title": "WE WANT TO TALK ABOUT OUR MARRIAGE",
5   .... "channel_title": "CaseyNeistat",
6   .... "category_id": 22,
7   .... "publish_time": "2017-11-13T17:13:01.000Z",
8   .... "views": 748374,
```

**Body** Cookies Headers (4) Test Results  Status: 201 Created 

Pretty Raw Preview Visualize **JSON**  

```
1 {}
2   "video_id": "2kyS6SvSYSE",
3   "trending_date": "2017-11-14",
4   "title": "WE WANT TO TALK ABOUT OUR MARRIAGE",
5   "channel_title": "CaseyNeistat",
6   "category_id": 22,
7   "publish_time": "2017-11-13T17:13:01.000Z",
8   "views": 748374,
```

```
Message received
Found the first timestamp: 2017-11-14 00:00:00
New trending date: 2017-11-14
{"video_id": "2kyS6SvSYSE", "trending_date": "2017-11-14", "title": "WE WA
NT TO TALK ABOUT OUR MARRIAGE", "channel_title": "CaseyNeistat", "category
_id": 22, "publish_time": "2017-11-13T17:13:01.000Z", "views": 748374, "li
kes": 57527, "dislikes": 2966, "comment_count": 15954}
172.18.0.1:57356 - "POST /videoitem HTTP/1.1" 201
```

```
$ ./kafka-console-consumer.sh --topic ingestion-topic --bootstrap-server localhost:9092
{"video_id": "2kyS6SvSYSE", "trending_date": "2017-11-14", "title": "WE WANT TO TALK ABOUT OUR MARRIAGE", "channel_title": "CaseyNeistat", "category_id": 22, "publish_t
ime": "2017-11-13T17:13:01.000Z", "views": 748374, "likes": 57527, "dislikes": 2966, "comment_count": 15954}
```

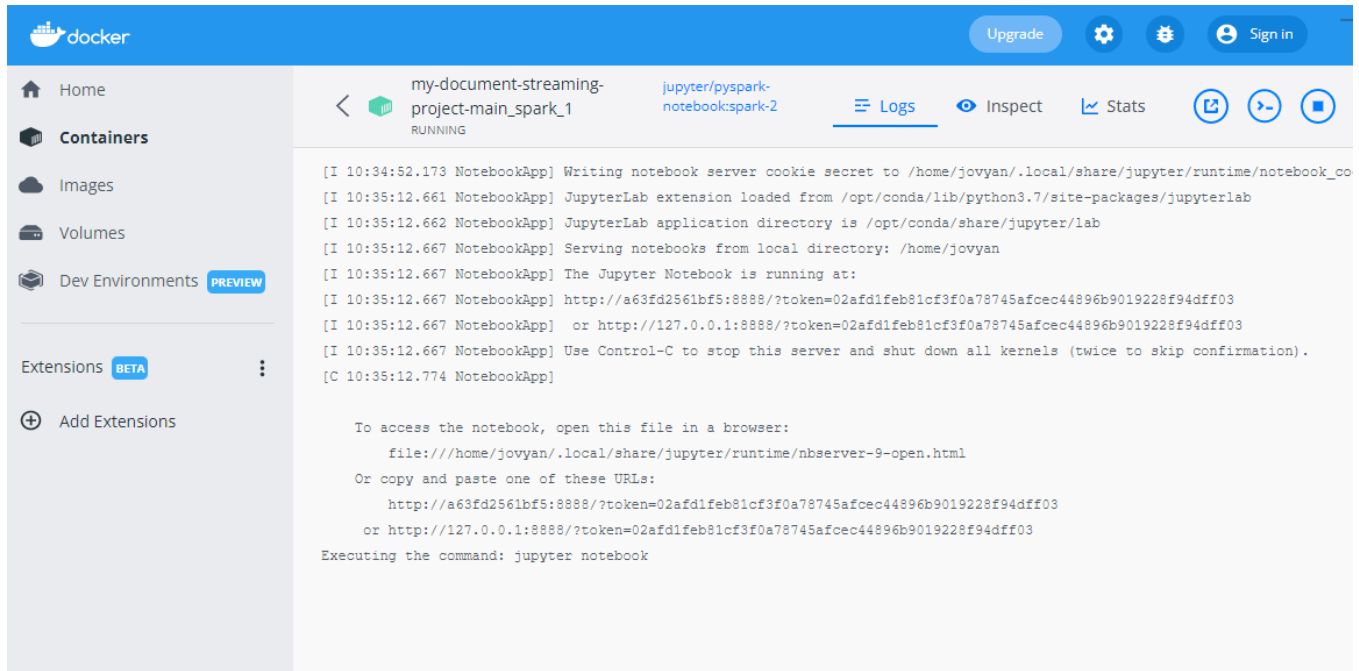
Both consumer and postman receive same message successfully.

Stop Container.

# Apache Spark

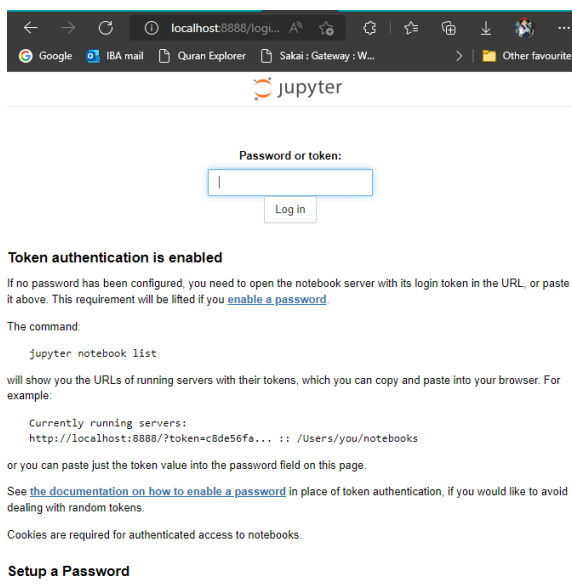
```
docker-compose -f docker-compose-kafka-spark.yml up
```

Once the container is running, go to *view logs* to get the token



token=02afd1feb81cf3f0a78745afcec44896b9019228f94dff03

Go to *localhost:8888* to enter the token and open jupyter notebook (this was configured in the docker compose yaml file), you should see 2 notebooks there already




Select items to perform actions on them.

Upload

New ▾



0 ▾ / work		Name ▾	Last Modified	File size
..			seconds ago	
<input type="checkbox"/>	 01-streaming-kafka-src-dst.ipynb		2 months ago	11.2 kB
<input type="checkbox"/>	 02-streaming-kafka-src-dst-mongodb.ipynb		2 months ago	5.43 kB

## Spark Streaming

Using python notebooks, a spark session will be set which will listen to messages from Kafka on one topic and receive it then send back to Kafka using a second topic – spark-output for testing.

- Create a new topic *spark-output*: `./kafka-topics.sh --create --topic spark-output --bootstrap-server localhost:9092`
- Check to see if both topics are there: `./kafka-topics.sh --list --bootstrap-server localhost:9092`

Create a local consumer: `./kafka-console-consumer.sh --topic spark-output --bootstrap-server localhost:9092`

```
$ cd /opt/bitnami/kafka/bin
$ ./kafka-topics.sh --create --topic spark-output --bootstrap-server localhost:9092
Created topic spark-output.
$ ./kafka-topics.sh --list --bootstrap-server localhost:9092
spark-output
$ ./kafka-topics.sh --create --topic ingestion-topic --bootstrap-server localhost:9092
Created topic ingestion-topic.
$ ./kafka-topics.sh --list --bootstrap-server localhost:9092
ingestion-topic
spark-output
$
```

- Execute the notebook then go to Postman and send the same record again
- The message should appear in local consumer and you should see the confirmation in the terminal

```
$ ./kafka-console-consumer.sh --topic spark-output --bootstrap-server localhost:9092
{"video_id": "2kyS6SvSYSE", "trending_date": "2017-11-14", "title": "WE WANT TO TALK ABOUT OUR MARRIAGE", "channel_title": "CaseyNeistat", "category_id": 22, "publish_time": "2017-11-13T17:13:01.000Z", "views": 748374, "likes": 57527, "dislikes": 2966, "comment_count": 15954}
```

POST http://127.0.0.1:81/videoitem

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL

```

1 [
2   "video_id": "2kyS6SvSYSE",
3   "trending_date": "17.14.11",
4   "title": "WE WANT TO TALK ABOUT OUR MARRIAGE",
5   "channel_title": "CaseyNeistat",
6   "category_id": 22,
7   "publish_time": "2017-11-13T17:13:01.000Z",
8   "views": 748374,

```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```

1 [
2   "video_id": "2kyS6SvSYSE",
3   "trending_date": "2017-11-14",
4   "title": "WE WANT TO TALK ABOUT OUR MARRIAGE",
5   "channel_title": "CaseyNeistat",
6   "category_id": 22,
7   "publish_time": "2017-11-13T17:13:01.000Z",
8   "views": 748374,

```

- You can also check out the Spark cluster interface at *localhost:4040*

localhost:4040/jobs/

Spark 2.4.5 Jobs Stages Storage Environment Executors SQL kafka-streaming application UI

### Spark Jobs (?)

User: joyvan  
Total Uptime: 8.1 min  
Scheduling Mode: FIFO  
Completed Jobs: 2

Event Timeline

Completed Jobs (2)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5 (8a42d9bb-998a-462d-93c2-625b65fbfe0e)	Id = 01af3b27-4455-4799-bb64-be28f4291a8e runId = 8a42d9bb-998a-462d-93c2-625b65fbfe0e batch = 1 start at NativeMethodAccessorImpl.java:0	2022/06/01 11:26:48	27 s	1/1	1/1
4 (54f7aa9c-472b-4a8e-b15c-72d473133751)	Id = 464a8377-78e6-412f-afd5-e7efe9ecc951 runId = 54f7aa9c-472b-4a8e-b15c-72d473133751 batch = 1 start at NativeMethodAccessorImpl.java:0	2022/06/01 11:26:43	31 s	1/1	1/1

Type here to search

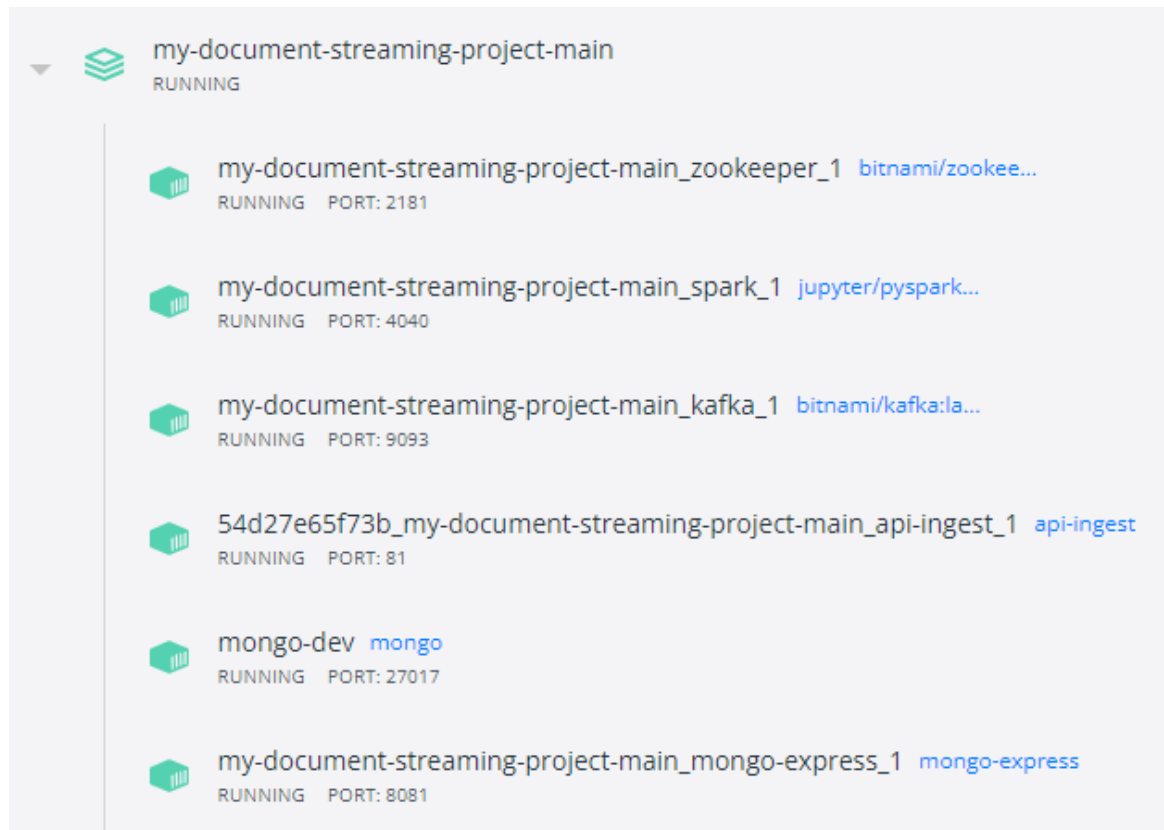
35°C 4:29 PM 6/1/2022

- Stop the container

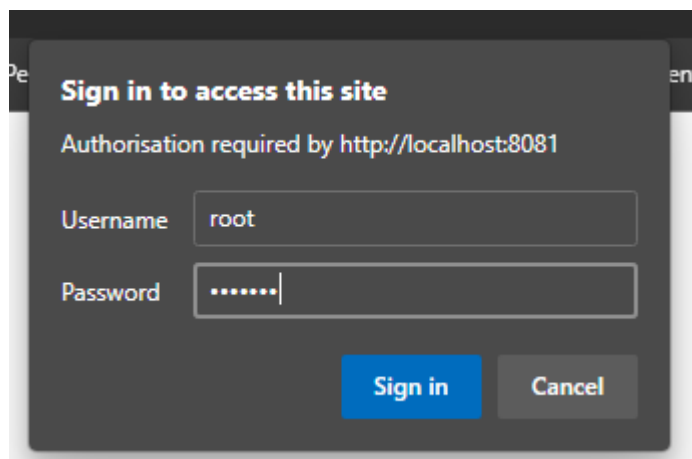
# MongoDB

It is essential to store the data and query it. For this project, I have chosen MongoDB.

```
docker-compose -f docker-compose-kafka-spark-mongodb.yml up
```



Go to localhost:8081



Username: admin P/w: tribes

Create a DB called docstreaming and create a collection *videos* to store all the records from Spark streaming

**Viewing Database: docstreaming**

**Collections**

Collection Name	View	Export	[JSON]	Import	Del
Videos					
delete_me					

**Database Stats**

Stat	Value
Collections (incl. system.namespaces)	2
Data Size	0 Byte
Storage Size	8.19 KB
Avg Obj Size #	0 Byte
Indexes #	2
Index Size	8.19 KB

## Connect Spark to MongoDB

- Go to 02-streaming-kafka-src-dst-mongodb.ipynb
- Note code block 6 adds a dataframe transformation to set the output table in MongoDB
- Execute the notebook then go to Postman and post a record. Check to see that it appeared in the MongoDB collection properly

**Viewing Collection: videos**

**Simple**

Key: Value String Find

Delete all 200 documents retrieved

First Prev Next Last

_id	video_id	title	channel_title	category_id	publish_time	trending_date	views	likes	dislikes	comment_count
629758c2410df0030659b3b	5XWEV0l40sE	THE INSIDE OUTTAKES - Bo Burnham (4K)	boburnham	23	2022-05-31T00:55:11Z	2022-06-01	2235673	0	0	14033

## API Client Writes Data

- Clean up MongoDB table if you sent test record earlier
- Use [api-client.py](#) to send the most recent trending video records
- Multiple batches with all the records are stored in MongoDB

Mongo Express Database: docstreaming Collection: videos

### Viewing Collection: videos

Simple Advanced

Key Value String Find

Delete all 1 documents retrieved

_id	video_id	title	channel_title	category_id	publish_time	trending_date	views	likes	dislikes	comment_count
62975291410df80030659b37	2kyS6SvSYSE	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	2017-11-13T17:13:01.000Z	2017-11-14	748374	57527	2966	15954

```
C:\Windows\System32\cmd.exe - docker-compose -f docker-compose-kafka-spark-mongodb.yml up
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | {"video_id": "OZrftoPOxfY", "trending_date": "2022-06-01", "title": "Testing Scary Minecraft Myths To Prove Them Wrong", "channel_title": "Eystreem", "category_id": 20, "publish_time": "2022-05-28T02:00:05Z", "views": 1627778, "likes": 0, "dislikes": 0, "comment_count": 2772}
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | 172.21.0.1:59628 - "POST /videoitem HTTP/1.1" 201
spark_1 | Batch: 58
spark_1 | +-----+
spark_1 | | key | | value |
spark_1 | +-----+
spark_1 | |null|{"video_id": "70F...|
spark_1 | |null|{"video_id": "y0P...|
spark_1 | |null|{"video_id": "N-t...|
spark_1 | +-----+
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Message received
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Found the first timestamp: 2022-06-01 00:00:00
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | New trending date: 2022-06-01
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | {"video_id": "rimu-boifX8", "trending_date": "2022-06-01", "title": "The World's most Extreme Theme Park! Xavage!", "channel_title": "The Ninja Fam!", "category_id": 19, "publish_time": "2022-05-27T16:30:04Z", "views": 891952, "likes": 0, "dislikes": 0, "comment_count": 739}
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | 172.21.0.1:59636 - "POST /videoitem HTTP/1.1" 201
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Message received
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Found the first timestamp: 2022-06-01 00:00:00
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | New trending date: 2022-06-01
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | {"video_id": "lqyxJ8WYQic", "trending_date": "2022-06-01", "title": "Millyz ft. Fivio Foreign - Opt Out (Official Video)", "channel_title": "Millyz", "category_id": 10, "publish_time": "2022-05-26T22:59:14Z", "views": 590120, "likes": 0, "dislikes": 0, "comment_count": 4010}
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | 172.21.0.1:59644 - "POST /videoitem HTTP/1.1" 201
spark_1 | Batch: 59
spark_1 | +-----+
spark_1 | | key | | value |
spark_1 | +-----+
spark_1 | |null|{"video_id": "OZr...|
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Message received
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | Found the first timestamp: 2022-06-01 00:00:00
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | New trending date: 2022-06-01
54d27e65f73b_my-document-streaming-project-main_api-ingest_1 | {"video_id": "Iynpoh8t1tE", "trending_date": "2022-06-01", "title": "What if Goku was Locked in the Time Chamber for millennia and Betrayed? Part 6", "channel_title": "Daiku Theories Dbs", "category_id": 22, "publish_time": "2022-05-28T00:15:09Z", "views": 506298, "likes": 0, "dislikes": 0, "comment_count": 1774}
```

Mongo Express Database: docstreaming Collection: videos

← First Prev Next → Last →

_id	video_id	title	channel_title	category_id	publish_time	trending_date	views	likes	dislikes	comment_count
629758c2410df80030659b3b	5XWEVol40sE	THE INSIDE OUTTAKES - Bo Burnham (4K)	boburnham	23	2022-05-31T00:55:11Z	2022-06-01	2235673	0	0	14033
629758c4410df80030659b3c	2vNVGZGIUok	Pinocchio   Teaser Trailer   Disney+	Walt Disney Studios	1	2022-05-31T12:45:36Z	2022-06-01	1318493	0	0	5403
629758c4410df80030659b3d	x97mC5LnuLk	Grupo Firme - Banda El Recodo - El Reemplazo - (...)	Grupo Firme	10	2022-05-31T17:00:10Z	2022-06-01	1121940	0	0	2184
629758c4410df80030659b3e	BS9YAI9EP8	IGN First - Sonic Frontiers Teaser	Sonic the Hedgehog	20	2022-05-31T16:00:26Z	2022-06-01	1221380	0	0	11181
629758c4410df80030659b3f	g-x0Y0QTKcs	'Stranger Things' Millie Bobby Brown & Noah	ELLE	24	2022-05-31T16:00:35Z	2022-06-01	1291647	0	0	1338

## Interface with Streamlit

To visualize the data output in table view and perform real-time analytics.

streamlitapp - Streamlit

localhost:8501

Visualization

Exact Video ID:

Regex for Video ID:

Exact Title:

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

Made with Streamlit

# Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

The streamlit app was modified to accept more queries.

Analytical queries were run and executed as follows.



## Query by exact Video ID:

```
if video_id:
    myquery = {"video_id": video_id}
    # only includes or excludes
    mydoc = mycol.find( myquery , { "category_id": 0, "_id": 0})

    # create dataframe from resulting documents to use drop_duplicates
    df = DataFrame(mydoc)

    # drop duplicates, but keep the first one
    df.drop_duplicates(subset ="channel_title", keep = 'first', inplace = True)

    # Add the table with a headline
    st.header("Output Videos by ID")
    table2 = st.dataframe(data=df)
```

streamlitapp - Streamlit

localhost:8501

Visualization

Exact Video ID:

5XWEVoI40sE

Regex for Video ID:

Exact Title:

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

## Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

### Output Videos by ID

	video_id	title	channel_title	publish_time	trendi
0	5XWEVoI40sE	THE INSIDE OUTTAKES - ...	boburnham	2022-05-31T00:55:11Z	2022-4

Made with Streamlit

## Query by regex:

```
if regex_id:
    myquery = {"video_id": {"$regex": regex_id}}
    # only includes or excludes
    mydoc = mycol.find(myquery, { "_id": 0, "trending_date": 0, "views": 0, "likes": 0, "dislikes": 0, "comment_count": 0 })

    # create dataframe from resulting documents to use drop_duplicates
    df = DataFrame(mydoc)
    print(df)
    # drop duplicates, but keep the first one
    df.drop_duplicates(subset ="video_id", keep = 'first', inplace = True)

    # Add the table with a headline
    st.header("Output Videos by Regex for Video ID")
    table3 = st.dataframe(data=df)
```

All videos whose id start with 5

streamlitapp - Streamlit

localhost:8501

Visualization

Exact Video ID:

Regex for Video ID:

Exact Title:

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

# Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

## Output Videos by Regex for Video ID

	video_id	title	channel_title	category_id	publish_time
0	5XWEVoI40sE	THE INSIDE OUTTAKES - ...	boburnham	23	2022-05-31T00:5

Made with Streamlit

Type here to search

30°C

10:19 PM 6/2/2022

### Query by exact title:

streamlitapp - Streamlit x +

localhost:8501

Visualization

Exact Video ID:

Regex for Video ID:

Exact Title:

What if Goku was Locked in the Time Ch.

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

# Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

## Output Videos by Title

	video_id	title	channel_title	publish_time
0	lynph8tITE	What if Goku was Locked ...	Daiku Theories Dbs	2022-05-28T00:15:09Z

Made with Streamlit

Type here to search

30°C

10:21 PM 6/2/2022

## Query by regex for title:

The screenshot shows a Streamlit web application running on a browser at localhost:8501. The application has a sidebar on the left with several input fields: 'Exact Video ID:', 'Regex for Video ID:', 'Exact Title:', 'Regex for Title:', 'Exact Channel Title:', and 'Regex for Channel Title:'. The 'Regex for Title:' field is highlighted with a red border and contains the text '^The'. The main area of the application displays a large heading 'Welcome to BDA Project' followed by 'by Abeera Tariq -13170'. Below this, it says 'Use the side-bar to query on the streaming data'. The main heading is 'Output Videos by Regex for Title'. Below this heading is a table with 5 columns: 'video\_id', 'title', 'channel\_title', 'category\_id', and 'publish\_t'. The table contains 10 rows of data. The Windows taskbar is visible at the bottom, showing the time as 10:25 PM on 6/2/2022.

	video_id	title	channel_title	category_id	publish_t
0	_DCsrRmQG...	The Legend of Zelda : Lin...	videogamedunkey	20	2022-05-3
1	EirBm-8240k	The Collider HIDE AND SE...	Nick Eh 30	20	2022-05-3
2	FfAueqEab30	The Summer I Turned Pre...	Prime Video	24	2022-05-3
3	FUU9EoZMU...	The Moses Ingram Situati...	Star Wars Theory	24	2022-05-3
4	La1wEk-24Ts	The NBA Playoffs Are Out ...	JxmyHighroller	17	2022-05-2
5	VbJ1EratsjA	The Pit Stop AS7 E03   Bo...	RuPaul's Drag Race	24	2022-05-2
6	_45zjnrfOA	The Cast of Stranger Thin...	BuzzFeed Celeb	24	2022-05-2
7	1QXxCEik51A	The ACTUAL Best Pokemo...	TheAMazing	1	2022-05-2
8	Y3uOQZolYQM	The Dark History of 2b2t's...	FitMC	20	2022-05-2
9	2h7nChCRvls	The Man Who Took Down...	Ask a Mortician	27	2022-05-2

## Query by exact Channel:

The screenshot shows the same Streamlit web application, but now the 'Exact Channel Title:' field in the sidebar is highlighted with a red border and contains the text 'Walt Disney Studios'. The main area of the application displays the same heading 'Welcome to BDA Project' followed by 'by Abeera Tariq -13170'. Below this, it says 'Use the side-bar to query on the streaming data'. The main heading is 'Output by Channel Title'. Below this heading is a table with 5 columns: 'video\_id', 'title', 'channel\_title', 'category\_id', and 'publish\_t'. The table contains 1 row of data. The Windows taskbar is visible at the bottom, showing the time as 10:26 PM on 6/2/2022.

	video_id	title	channel_title	category_id	publish_t
0	2vNVGZGIUok	Pinocchio   Teaser Trailer ...	Walt Disney Studios	1	2022-05-3

## Query by channel regex:

streamlitapp - Streamlit

localhost:8501

Exact Video ID:

Regex for Video ID:

Exact Title:

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

Go

# Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

## Output Videos by Regex for Channel Title

	video_id	title	channel_title	category_id	publish_
0	amMxvP00JBI	Early Release: Celia Muno...	America's Got Talent	24	2022-05-
1	jNlyzAqPK8k	GOLDEN BUZZER! Alesha ...	Britain's Got Talent	24	2022-05-

Made with Streamlit

Type here to search

30°C

10:31 PM 6/2/2022

## Visualizations for Dashboard:

streamlitapp - Streamlit

localhost:8501

Visualization

Exact Video ID:

Regex for Video ID:

Exact Title:

Regex for Title:

Exact Channel Title:

Regex for Channel Title:

# Welcome to BDA Project

by Abeera Tariq -13170

Use the side-bar to query on the streaming data

## Top 10 Trending viewed Youtube Videos

Index	Views
0	928637
1	939017
2	945484
3	949621
4	954515
5	957213
6	971419
7	985312
8	9914842
9	998543

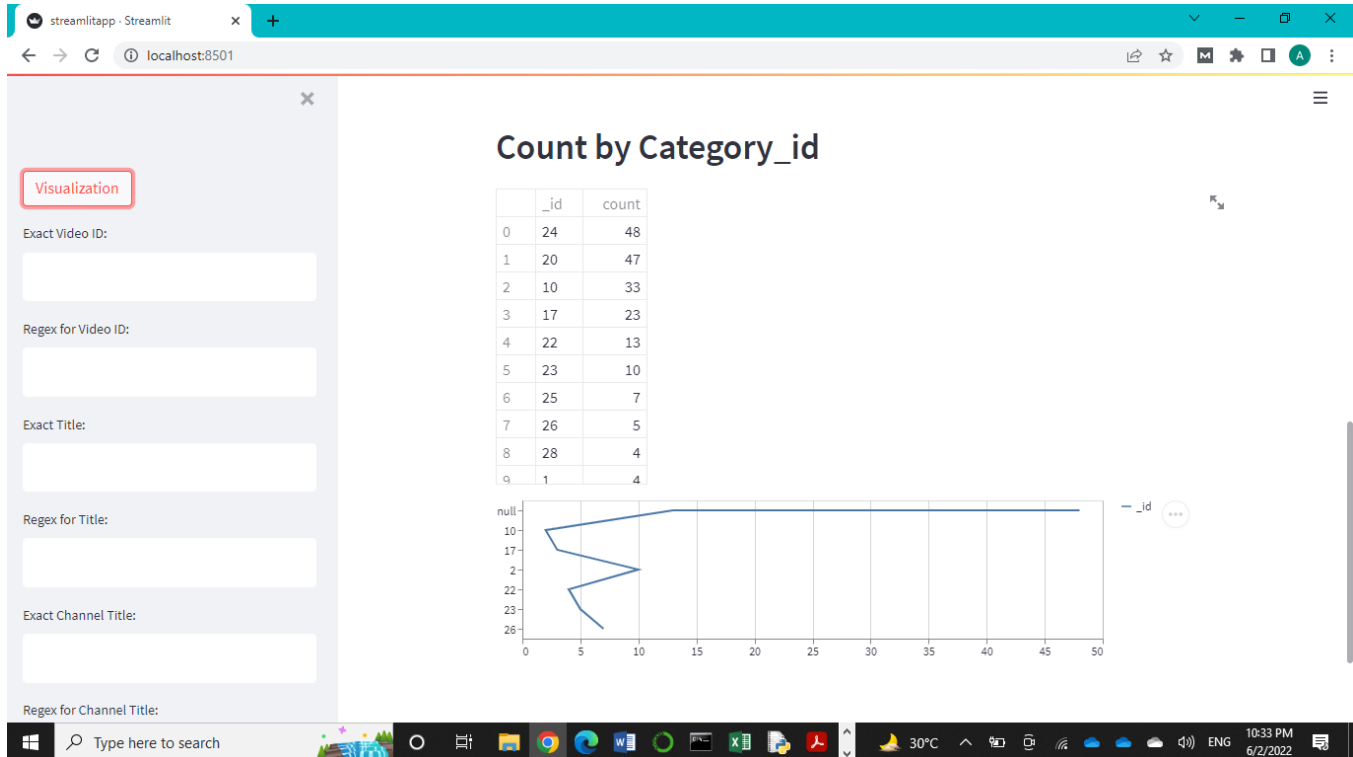
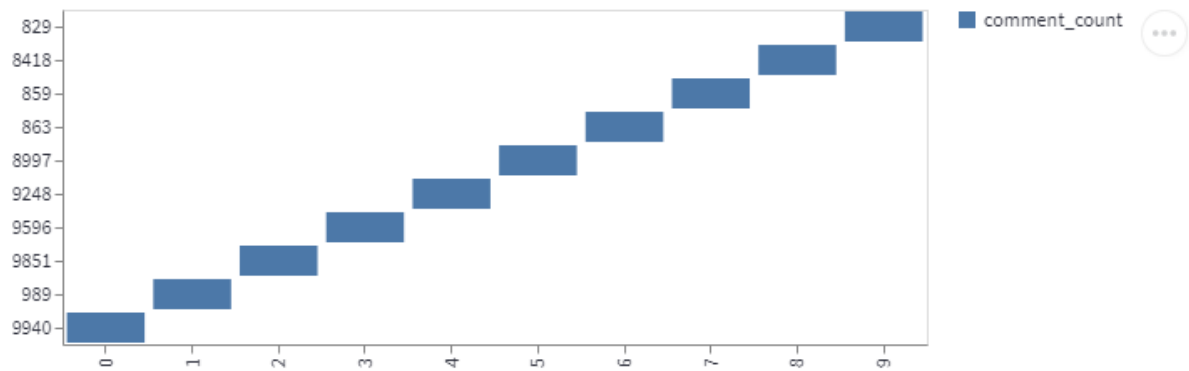
## Top 10 commented Trending Youtube Videos

Type here to search

30°C

10:32 PM 6/2/2022

# Top 10 commented Trending Youtube Videos



## Acknowledgement:

Special thanks to 2 GitHub repositories that served as great guidance to execute this project

[GitHub - anqizhao7/my-document-streaming-project](https://github.com/anqizhao7/my-document-streaming-project)

[GitHub - mitchelljy/Trending-YouTube-Scraper: Python script that scrapes the currently trending YouTube videos in a variety of countries](https://github.com/mitchelljy/Trending-YouTube-Scraper)