

# TECHNICAL PROJECT

ECE BIG DATA ECOSYSTEM CLASS

BUILDING A DISTRIBUTED DATA PROCESSING  
PIPELINE WITH APACHE SPARK, HADOOP, AND  
KIBANA ON AWS EMR ECOSYSTEM



# Summary

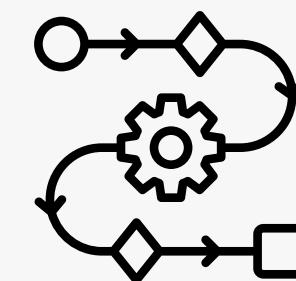
01 OVERVIEW



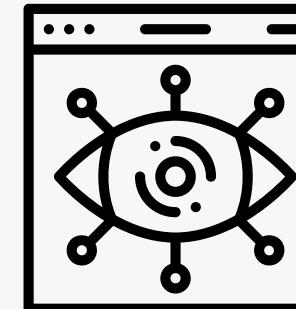
02 PROJECT ARCHITECTURE



03 PIPELINE WORKFLOW



04 KIBANA DASHBOARDS



05 CHALLENGES ENCOUNTERED



06 CONCLUSION



# 01 OVERVIEW



In this project, we designed and implemented a scalable data processing pipeline using open-source distributed systems:

Apache Spark

Apache Hadoop

Kibana and Elasticsearch

AWS s3 / Ec2 / EMR

While AWS EMR is neither open-source nor free for large-scale projects, we were eager to use it for **two key reasons**:

## CONCEPTS STUDIED IN CLASS

---

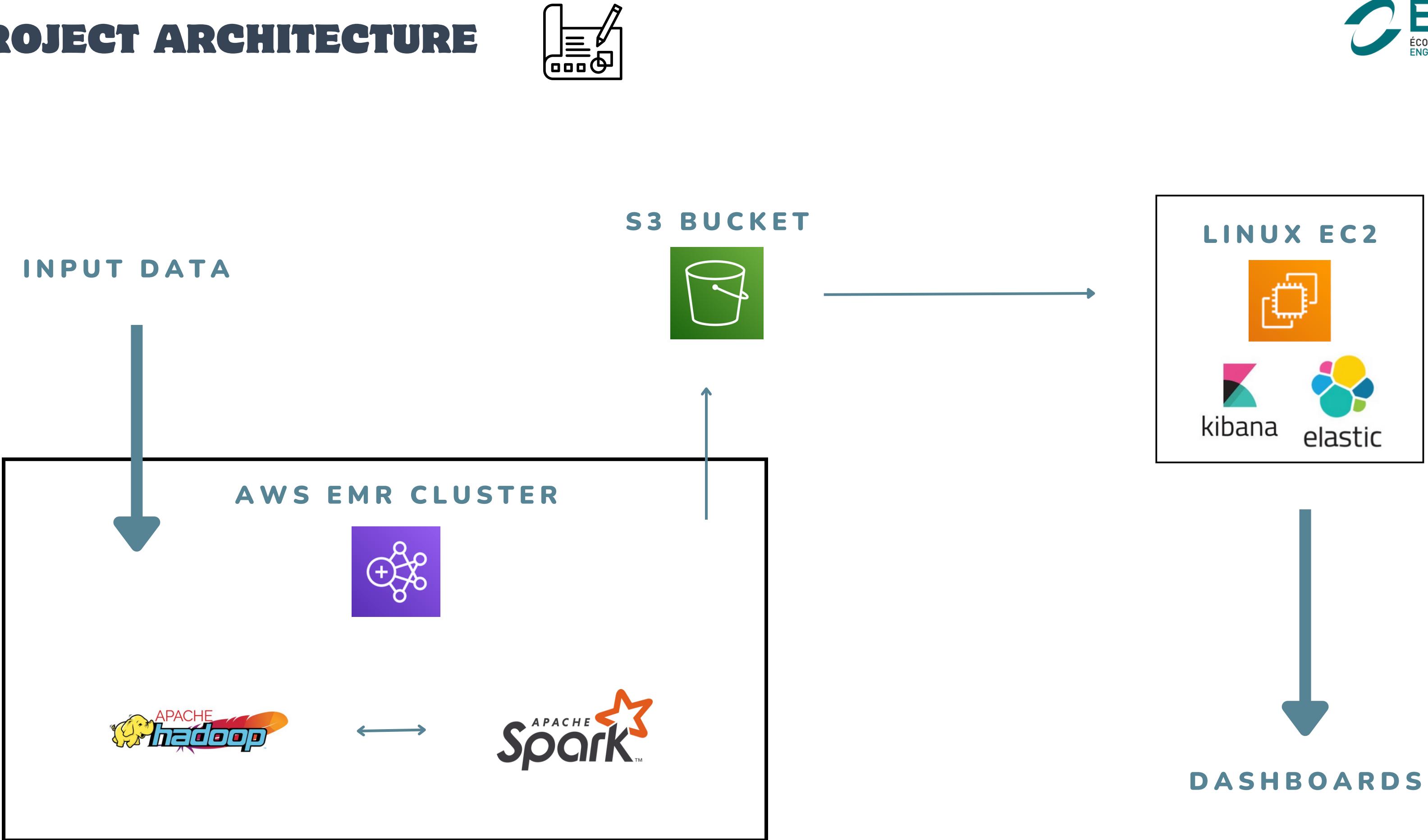
EMR closely mirrors the concepts of clusters and scalability that we study in BigData Ecosystem courses with you. It provides a managed and scalable environment to process large datasets, aligning well with our project's goals.

## CLOUD TECHNOLOGY WITH AWS

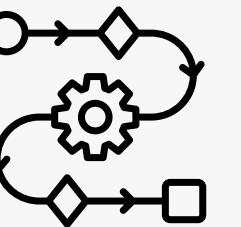
---

AWS EMR is a modern cloud-based solution increasingly adopted by companies. By working with it, we gained valuable experience with AWS services and the cloud provider ecosystem.

## 02 PROJECT ARCHITECTURE



# 03 PIPELINE WORKFLOW



## 00. OVERALL WORKFLOW

---

### Data Ingestion:

- Input CSV is uploaded to S3 Input Folder.
- Apache Spark reads data directly from S3.

### Data Transformation with Spark:

- Transformations include column renaming, data type casting, and feature engineering.
- Transformed JSON is written to S3 Output Folder.

### Distributed Storage with Hadoop (HDFS):

- Intermediate storage for processing within the EMR cluster.

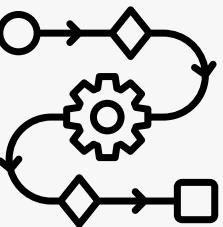
### Visualization with Kibana:

- Transformed JSON is indexed into Elasticsearch running on an EC2 instance.
- Visualizations created using Kibana dashboards.

We will see the details of each steps in the following slides



# 03 PIPELINE WORKFLOW



## 01. DATA INGESTION:

Sample of the content of the csv file (Downloaded from Kaggle):

```

START_DATE,END_DATE,CATEGORY,START,STOP,MILES,PURPOSE
01-01-2016 21:11,01-01-2016 21:17,Business,Fort Pierce,Fort Pierce,5.1,Meal/Entertain
01-02-2016 01:25,01-02-2016 01:37,Business,Fort Pierce,Fort Pierce,5,
01-02-2016 20:25,01-02-2016 20:38,Business,Fort Pierce,Fort Pierce,4.8,Errand/Supplies
01-05-2016 17:31,01-05-2016 17:45,Business,Fort Pierce,Fort Pierce,4.7,Meeting
01-06-2016 14:42,01-06-2016 15:49,Business,Fort Pierce,West Palm Beach,63.7,Customer Visit
01-06-2016 17:15,01-06-2016 17:19,Business,West Palm Beach,West Palm Beach,4.3,Meal/Entertain
01-06-2016 17:30,01-06-2016 17:35,Business,West Palm Beach,Palm Beach,7.1,Meeting
01-07-2016 13:27,01-07-2016 13:33,Business,Cary,Cary,0.8,Meeting
01-10-2016 08:05,01-10-2016 08:25,Business,Cary,Morrisville,8.3,Meeting
01-10-2016 12:17,01-10-2016 12:44,Business,Jamaica,New York,16.5,Customer Visit
    
```

LINK TO THE DATA HERE:

[HTTPS://WWW.KAGGLE.COM/DATASETS/BHANUPRATAPBISWAS/UBER-DATA-ANALYSIS?RESOURCE=DOWNLOAD](https://www.kaggle.com/datasets/bhanupratapbiswas/uber-data-analysis?resource=download)



Content of the S3:

- input folder with our data.csv file
- logs folder (linked with our EMR to instantly get the log here)
- output folder store the processed data from the cluster to .json format
- Process5.py is our currently used spark script
- Scripts folder contain other scripts that we tried for testing etc ..

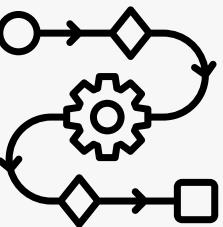
Objects (5) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.

[Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">input/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">logs/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">output/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">Process5.py</a>	py	January 3, 2025, 15:34:21 (UTC+01:00)	1.7 KB	Standard
<input type="checkbox"/>	<a href="#">Scripts/</a>	Folder	-	-	-

# 03 PIPELINE WORKFLOW



## 02. DATA TRANSFORMATION WITH SPARK

```

Users > nico > Desktop > Process5.py > transform_data
5
6  def transform_data(data_source: str, output_uri: str) -> None:
7
8      with SparkSession.builder.appName("Business Rides Data Transformation").getOrCreate() as spark:
9          # Step 1: Read CSV file from S3
10         df = spark.read.option("header", "true").csv(data_source)
11
12         # Step 2: Perform data transformations
13         # Convert columns to proper data types
14         df = df.withColumn("MILES", col("MILES").cast("double"))
15
16
17         df = df.withColumnRenamed("START", "start_location") \
18             .withColumnRenamed("CATEGORY", "ride_category") \
19             .withColumnRenamed("PURPOSE", "ride_purpose") \
20             .withColumnRenamed("START_DATE", "start_date") \
21             .withColumnRenamed("END_DATE", "end_date") \
22             .withColumnRenamed("STOP", "end_location")
23
24
25         df = df.withColumn("ride_length_category",
26                         when(col("MILES") > 10, lit("long"))
27                         .otherwise(lit("short")))
28
29         # Step 3: Write intermediate results to HDFS
30         hdfs_intermediate_path = "hdfs://intermediate_data"
31         df.write.mode("overwrite").parquet(hdfs_intermediate_path) # Save in Parquet format for efficient storage
32
33         print(f"Intermediate data written to HDFS at {hdfs_intermediate_path}")
34
35
36         # Step 4: Read intermediate data from HDFS
37         df_intermediate = spark.read.parquet(hdfs_intermediate_path)
38
39         # Step 5: Write the final transformed data to S3 as a JSON file
40         df_intermediate.write.mode("overwrite").json(f"{output_uri}/transformed_data.json")
41
42         print(f"Transformed data written to {output_uri}/transformed_data.json")
43
44     if __name__ == "__main__":
45         parser = argparse.ArgumentParser()
46         parser.add_argument("--data_source", help="Path to the input CSV file")
47         parser.add_argument("--output_uri", help="Path to save the transformed JSON file")

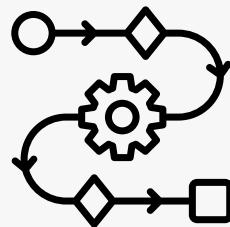
```

CODE ON GITHUB

Step ID	Status	Name	Log files	Creation time (UTC+01:00)	Start time (UTC+01:00)
s-0217742Z2LNGJQFUGW	Completed	Try 4	controller   syslog   stderr   stdout	4 January 2025 at 12:02	4 January 2025 at 12:12
s-02590602EVQR4PSTVD27	Completed	Try 3	controller   syslog   stderr   stdout	4 January 2025 at 12:02	4 January 2025 at 12:11
s-080861036ACPNTRXBLIS	Failed	Try 3	controller   syslog   stderr   stdout	4 January 2025 at 12:02	4 January 2025 at 12:11
s-09404132F1AJGEG7HIJ3	Failed	Try 3	controller   syslog   stderr   stdout	4 January 2025 at 12:02	4 January 2025 at 12:10

EACH STEP ON EMR CORRESPOND TO THE EXECUTION OF A SCRIPT  
HERE "TRY 4" IS OUR STEP EXECUTING THE SPARK CODE

# 03 PIPELINE WORKFLOW



## 02. DATA TRANSFORMATION WITH SPARK

Amazon EMR > EMR on EC2: Clusters > Big-Data Ecosystem project cluster > Clone step s-0217742Z2LNGZJQFUGW

### Step settings

Type

- Custom JAR  
Adds a step that enables you to write a custom script to process your data using the Java programming language.
- Spark application  
Adds a step that submits work to the Spark framework on the cluster.
- Streaming program  
Adds a step that uses standard input to run mapper/reducer scripts and send results to standard output.
- Shell script  
Troubleshoot your cluster.

Name

Deploy mode

- Cluster mode  
Run your driver on a worker node.
- Client mode  
Run your driver on the primary node as an external client.

Application location

Path to a JAR with your application and dependencies (client deploy mode only supports a local path).

View
Browse S3

Spark-submit options - optional

Specify other options for spark-submit.

```
s3://nico-emr-bigdata-ecosystem/monthly_build/2023-09/Process5.py --data_source s3://nico-emr-bigdata-ecosystem/monthly_build/2023-09/input/UberDataset.csv --output_uri
```

Arguments - optional

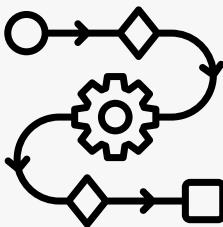
Info

Specify optional arguments for your script.

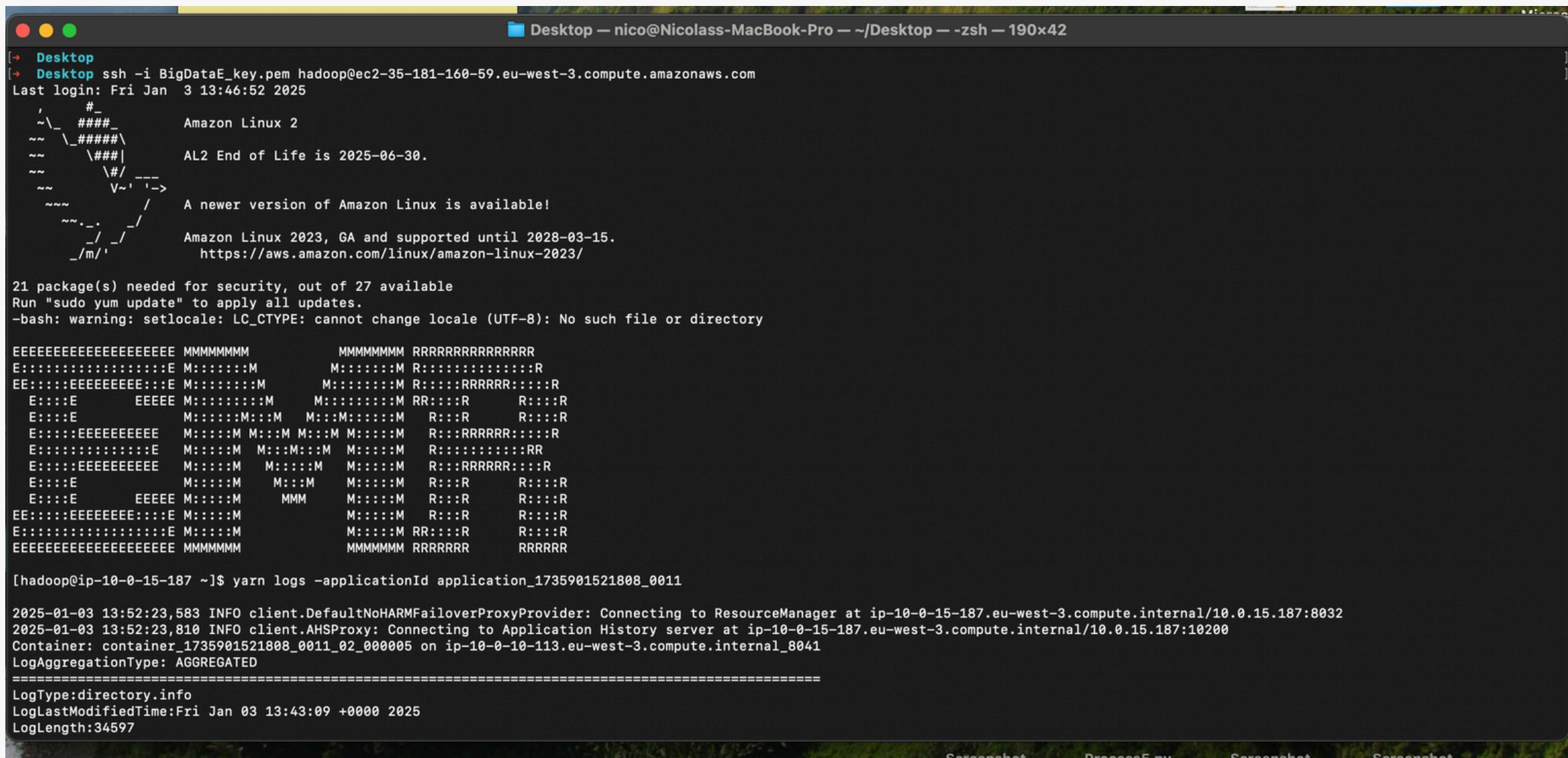
```
--data_source s3://nico-emr-bigdata-ecosystem/monthly_build/2023-09/input/UberDataset.csv
--output_uri hdfs://output/transformed_data or s3://nico-emr-bigdata-ecosystem/monthly_build/2023-09/output (if it is the last step of our transformation)
```

EXAMPLE OF STEP CONFIGURATION

# 03 PIPELINE WORKFLOW



## 03. DISTRIBUTED STORAGE WITH HADOOP



```

Desktop
Desktop ssh -i BigDataE_key.pem hadoop@ec2-35-181-160-59.eu-west-3.compute.amazonaws.com
Last login: Fri Jan  3 13:46:52 2025
      _#
     ~\_ ####_      Amazon Linux 2
     ~~ \####\`_
     ~~ \|##|_      AL2 End of Life is 2025-06-30.
     ~~ \|#/--->
     ~~ V~' '-->
     ~~ /       A newer version of Amazon Linux is available!
     ~~ .-./
     _/_. /      Amazon Linux 2023, GA and supported until 2028-03-15.
     _/m/'      https://aws.amazon.com/linux/amazon-linux-2023/

21 package(s) needed for security, out of 27 available
Run "sudo yum update" to apply all updates.
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory

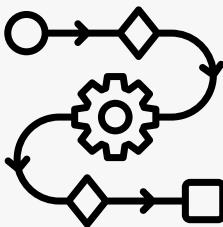
EEEEEEEEEEEEEE MMMMMMM MBBBBBBBBBBBBBB RRRRRRRRRRRRRR
E:::::::::::E M:::::M M:::::M R:::::::::::R
EE:::::EEEEEEE:::E M:::::M M:::::M R:::::RRRRR:::::R
   E:::E   EEEE M:::::M M:::::M RR:::::R   R:::::R
   E:::E   M:::::M::::M M:::::M::::M R:::R   R:::::R
   E:::::EEEEEEEEE M:::::M M::::M M:::::M R:::RRRRR:::::R
   E:::::::::::E M:::::M M:::::M M:::::M R:::::::::::RR
   E:::::EEEEEEE M:::::M M:::::M M:::::M R:::RRRRR:::::R
   E:::E   M:::::M M:::::M M:::::M R:::R   R:::::R
   E:::E   EEEE M:::::M M:::M M:::::M R:::R   R:::::R
EE:::::EEEEEEE:::E M:::::M M:::::M R:::R   R:::::R
E:::::::::::E M:::::M M:::::M RR:::::R   R:::::R
EEEEEEEEEEEEEE MMMMMMM MBBBBBBBBBBBBBB RRRRRRRR

[hadoop@ip-10-0-15-187 ~]$ yarn logs -applicationId application_1735901521808_0011
2025-01-03 13:52:23,583 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at ip-10-0-15-187.eu-west-3.compute.internal/10.0.15.187:8032
2025-01-03 13:52:23,810 INFO client.AHSProxy: Connecting to Application History server at ip-10-0-15-187.eu-west-3.compute.internal/10.0.15.187:10200
Container: container_1735901521808_0011_02_000005 on ip-10-0-10-113.eu-west-3.compute.internal_8041
LogAggregationType: AGGREGATED
=====
LogType:directory.info
LogLastModifiedTime:Fri Jan 03 13:43:09 +0000 2025
LogLength:34597

```

MANUALLY CONNECT TO OUR EMR TO SEE LOGS AND CHECK IF HDFS  
WELL GET OUR FILES

# 03 PIPELINE WORKFLOW



## 04. VISUALIZATION WITH KIBANA:

```

Last login: Thu Jan  2 14:42:24 on ttys008
↳ ~
↳ ~
↳ ~ cd Desktop
↳ Desktop ssh -i "BigDataE_key.pem" ubuntu@ec2-51-44-25-2.eu-west-3.compute.amazonaws.com
The authenticity of host 'ec2-51-44-25-2.eu-west-3.compute.amazonaws.com (51.44.25.2)' can't be established.
ED25519 key fingerprint is SHA256:PsAkk4zQ0lctJkxnd4x6oG1JeCQ/A+p2NH51Ys3/vyo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-51-44-25-2.eu-west-3.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Thu Jan  2 13:55:44 UTC 2025

System load: 0.93 Temperature: -273.1 C
Usage of /: 24.7% of 6.71GB Processes: 116
Memory usage: 6% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.9.138

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-9-138:~$ sudo yum install java-11
sudo: yum: command not found
ubuntu@ip-172-31-9-138:~$ sudo apt update
Hit:1 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://eu-west-3.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
  
```

After Processing our data and uploading it to the S3 we run an EC2 instance to host Elastic Search and Kibana.

Tutorial we use for the installation on EC2 :

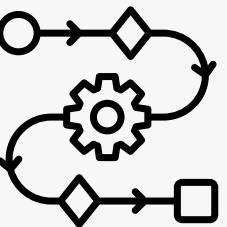
<https://balajidevops.hashnode.dev/guide-to-installing-elasticsearch-logstash-and-kibana-on-amazon-linux-2023>

We install the ELK stack without Logstash because it is useless in our case , for that we install java on our machine and we configure the EC2 policy to accept trafic on port number 5601 (Kibana) and 9200 (Elasticsearch)

▼ Inbound rules				
Filter rules				
Security group rule ID	Port range	Protocol	Source	Security groups
sgr-0e21ff4663797992f	22	TCP	0.0.0.0/0	<a href="#">launch-wizard-6</a> [2]
sgr-09f3c483de46f3a2f	9200	TCP	0.0.0.0/0	<a href="#">launch-wizard-6</a> [2]
sgr-09423a9d2a77d6a44	5601	TCP	0.0.0.0/0	<a href="#">launch-wizard-6</a> [2]

MANUALLY CONNECT TO OUR EMR TO SEE LOGS AND CHECK IF HDFS  
WELL GET OUR FILES

# 03 PIPELINE WORKFLOW

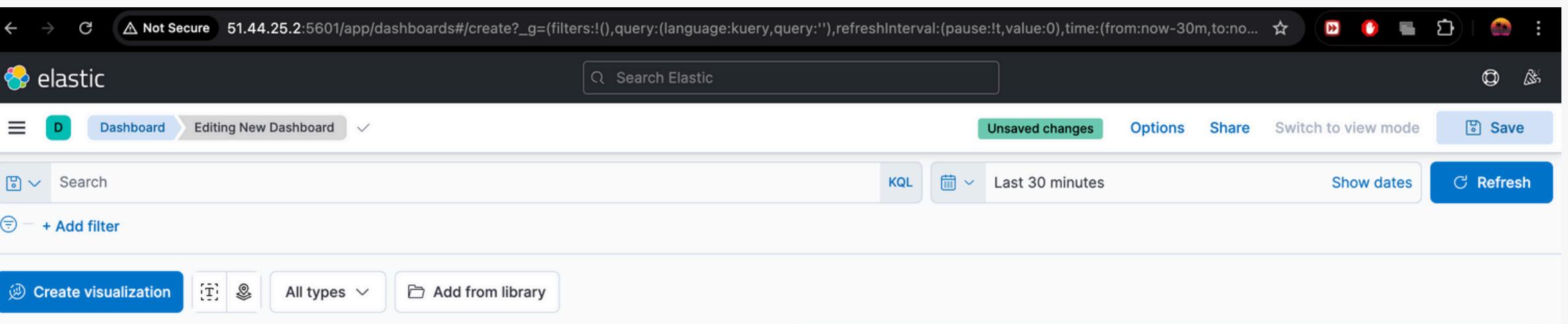


## 04. VISUALIZATION WITH KIBANA:

```

[ubuntu@pod-elk:~$ sudo service kibana start
[ubuntu@pod-elk:~$ sudo service kibana status
● kibana.service - Kibana
  Loaded: loaded (/etc/systemd/system/kibana.service; disabled; preset: enabled)
  Active: active (running) since Thu 2025-01-02 14:40:06 UTC; 4min 11s ago
    Docs: https://www.elastic.co
    Main PID: 1412 (node)
       Tasks: 11 (limit: 4586)
      Memory: 254.7M (peak: 350.0M)
        CPU: 30.367s
       CGroup: /system.slice/kibana.service
           └─1412 /usr/share/kibana/bin/.../node/bin/node /usr/share/kibana/bin/.../src/cli/dist --logging.dest=/var/log/kibana/kibana.log --pid.file=/run/kibana/kibana.pid "--deprecation.skip_depr

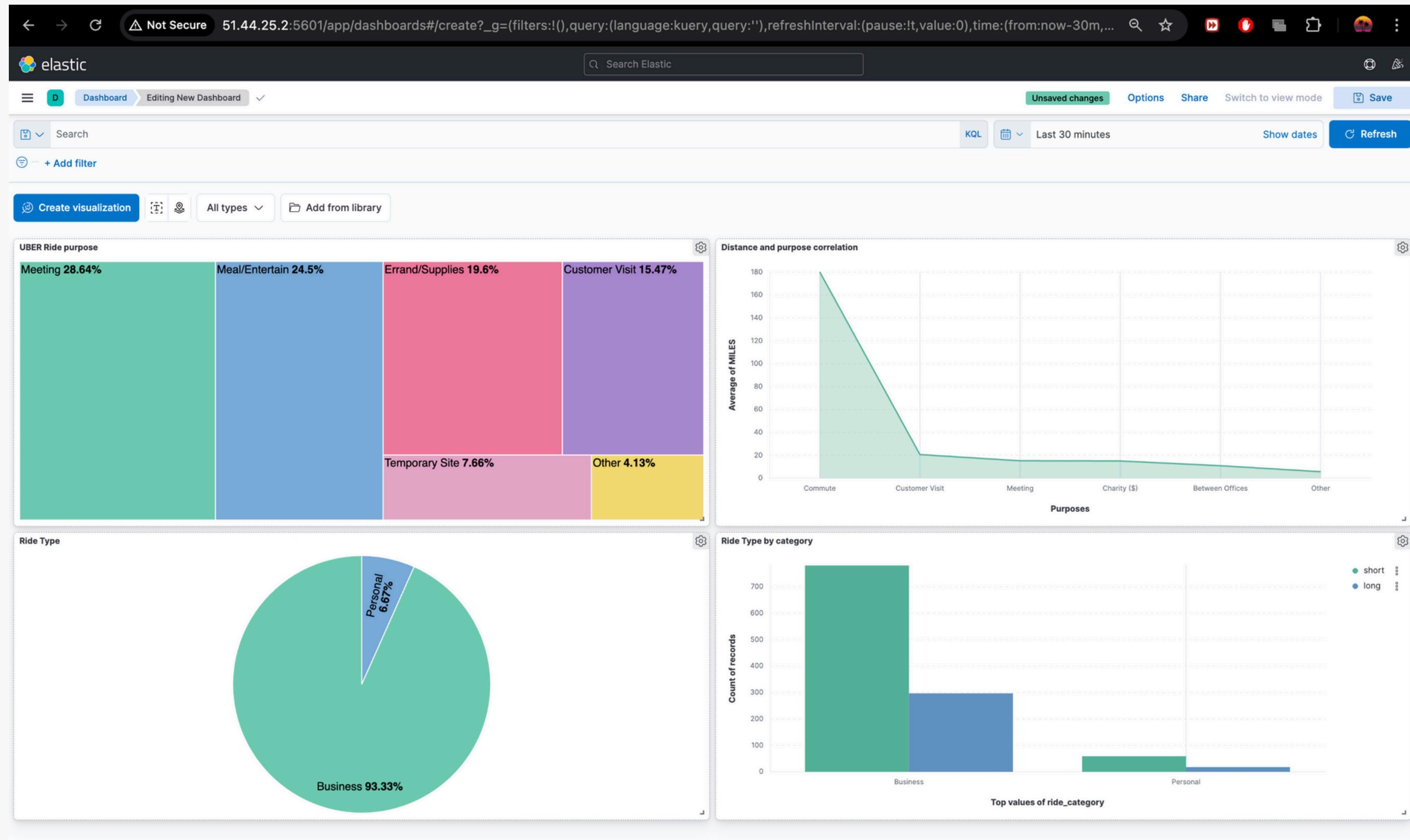
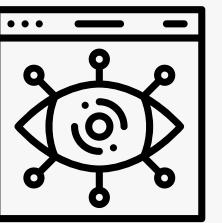
Jan 02 14:40:06 pod-elk systemd[1]: Started Kibana.service - Kibana.
Jan 02 14:40:06 pod-elk kibana[1412]: Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to disable see https://www.elastic.co/guide/en/kibana/7...
[ubuntu@pod-elk:~$ 
[ubuntu@pod-elk:~$ sudo nano /etc/kibana/kibana.yml
  
```



The screenshot shows the Kibana interface for creating a new dashboard. The URL in the browser is `51.44.25.2:5601/app/dashboards#/create?_g=(filters:!(),query:(language:kuery,query:'')),refreshInterval:(pause:!t,value:0),time:(from:now-30m,to:now)`. The main area is titled "Editing New Dashboard". It includes a search bar, date range selector (Last 30 minutes), and a "Create visualization" button. There are also tabs for "Search" and "KQL". At the top right, there are buttons for "Save", "Options", "Share", and "Switch to view mode". The overall theme is dark, with blue highlights for interactive elements.

LAUNCH KIBANA ON 5601

# 04 KIBANA DASHBOARDS



Here we did simple visualizations to create a Dashboard.

The main objectif is to show that Kibana works with our data on the EC2 instance. There are surely many more beautiful and relevant dashboards to create than this one

# 05 CHALLENGES ENCOUNTERED



## 01. GETTING STARTED WITH AWS EMR

Working with AWS EMR was challenging at first. We spent considerable time and effort ensuring the script could correctly access input files in S3 and upload outputs to the right location. Small misconfigurations caused repeated failures, making the process frustrating.

Configuring Virtual Private Cloud (VPC) for secure communication between EMR, S3, and other components was complex. It took time to understand subnets, gateways, and routing. Thankfully, online resources and a helpful tutorial video from an American instructor allowed us to overcome these hurdles.

Balancing new tools and the project timeline was tough, but it was a rewarding learning experience overall.

The screenshot shows a YouTube video player with a thumbnail of a man speaking. The video is titled "Intro to Amazon EMR - Big Data Tutorial using Spark" and is 15:03 long. The AWS EMR console is displayed in the video frame, specifically the "Steps" tab for a cluster named "emr-tutorial-cluster". A single step named "MyFirstStep" is listed, showing a status of "Completed". The console interface includes tabs for "Properties", "Bootstrap actions", "Instances (Hardware)", "Steps", "Applications", "Configurations", and "Monitoring".

[LINK HERE](#)

The screenshot shows the AWS EMR console under the "Steps" tab for a cluster named "Big-Data Ecosystem project cluster". There are 14 steps listed, all of which are in a failed state. A large black arrow points from this screenshot towards the text below.

Step ID	Status	Name	Log files
s-02590602EVQR4PSTVD27	Completed	Try 3	controller   syslog   stderr   stdout
s-080861036ACPNTRXBLIS	Failed	Try 3	controller   syslog   stderr   stdout
s-09404132F1AJGEG7HJ3	Failed	Try 3	controller   syslog   stderr   stdout
s-06073872K7N4U9NEYFNC	Failed	Try 3	controller   syslog   stderr   stdout
s-02340061VUMC5SS0IMCL	Failed	Try 3	controller   syslog   stderr   stdout
s-03707221PZ2VKBT48DQ4	Failed	Try 2	controller   syslog   stderr   stdout
s-076161896XLT20KFFM	Failed	Try 1	controller   syslog   stderr   stdout
s-06067063FPRIVVRUYAPZ	Failed	Process2 Data to Kibana	controller   syslog   stderr   stdout
s-0155116TEWIOXNYJCWG	Failed	Process Data to Kibana V2	controller   syslog   stderr   stdout
s-06069803NP7J4C1INNEZ8	Failed	Process Data to Kibana	controller   syslog   stderr   stdout
s-0149334SY1JDILE5SQ	Failed	Process Data to Kibana	controller   syslog   stderr   stdout
s-06364531GFCHRQAJXYR	Completed	MyStep2_ProcessBoltData	controller   syslog   stderr   stdout
s-02361842BV8QFAGU7XWB	Completed	MyFirstStep	controller   syslog   stderr   stdout

The number of failed tasks before successfully executing our scripts in the EMR demonstrate well how we struggle to get all of this work



# 05 CHALLENGES ENCOUNTERED



Screenshot of the AWS Amazon EMR console showing the list of clusters. A red box highlights the top cluster, and a red arrow points from it to the text "it was very time-consuming overall".

Cluster ID	Cluster name	Status	Creation time (UTC+01:00)	Elapsed time
j-3LCMDKQ1BYS77	Big-Data Ecosystem project cluster	Terminated Auto-terminate	January 04, 2025, 12:02	3 hours, 13 minutes
j-IP1CXU8TL213	emr-tutorial-clusterV2	Terminated Auto-terminate	January 03, 2025, 11:49	6 hours, 49 minutes
j-3IJ3CRFVIYWQ6	emr-tutorial-clusterV2	Terminated User request	January 02, 2025, 09:51	45 minutes, 23 seconds
j-3NM32G7ICLKSW	emr-tutorial-clusterV2	Terminated Auto-terminate	January 01, 2025, 13:48	3 hours, 12 minutes
j-25Z6ZUG1ZO126	My cluster V2	Terminated with errors Validation error	January 01, 2025, 13:20	2 minutes, 9 seconds
j-2X7Z29HFVCLG2	V2-emr	Terminated with errors Validation error	January 01, 2025, 13:12	2 minutes, 9 seconds
j-1TYSG83FPU98Y	emr-tutorial-cluster	Terminated User request	January 01, 2025, 10:47	2 hours, 54 minutes
j-2ATB6UWZHQQCN	emr-tutorial-cluster	Terminated Auto-terminate	December 30, 2024, 19:30	2 hours, 9 minutes
j-1ZNPMVVKNIRF1	emr-tutorial-cluster	Terminated Auto-terminate	December 28, 2024, 10:39	3 hours, 41 minutes

it was very time-consuming overall

# 05 CHALLENGES ENCOUNTERED



To solve this issues we decided to do a step by step method by executing small scripts first before executing bigger scripts to make sure all our components works.

```
[[hadoop@ip-10-0-11-146 ~]$ echo "hello world hello" | python3 wordcount2.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/01/01 10:20:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes wh
icable
Word Count Results:
emr: 2
spark: 3
test: 1
is: 1
awesome: 1
example: 1
[hadoop@ip-10-0-11-146 ~]$
```

## 02. CREATE AN EC2 INSTANCE WITH ELK ON IT

We lost an enormous amount of time in order to install kibana and elastisearch, every time we tried the EC2 crashed, we learned after that the basic free tier machine was not big enough to handle the EK stack

Instances (2) <a href="#">Info</a>									
<a href="#">Find Instance by attribute or tag (case-sensitive)</a> <a href="#">All states</a>									
	Name <a href="#">Edit</a>	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv	
<input type="checkbox"/>	Big-Data-Proje...	i-0e00b807d9d69e2b5	<a href="#">Stopped</a> <a href="#">Q</a> <a href="#">Q</a>	t2.micro	-	<a href="#">View alarms +</a>	eu-west-3c	-	
<input type="checkbox"/>	ELK_V3	i-0e207353c29b24d3b	<a href="#">Running</a> <a href="#">Q</a> <a href="#">Q</a>	t3.medium	<a href="#">3/3 checks passed</a> <a href="#">View alarms +</a>	<a href="#">View alarms +</a>	eu-west-3a	ec2-51-44-	



# 06 CONCLUSION



This project was a valuable learning experience for us. We used what we've learned about distributed systems to build a small scalable data processing pipeline with Apache Spark and Hadoop on AWS EMR and the ELK stack with mainly Kibana.

We faced several challenges. At first, it was hard to understand how everything worked together, and it took us many tries to get it right.

Even though things didn't always go smoothly, we learned a lot by solving these problems step by step.

**This project also gave us an overview of how open-source tools and cloud technology can work together, and we feel more comfortable with these concepts now.**