TECHNICAL PROJECT

ECE BIG DATA ECOSYSTEM CLASS

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

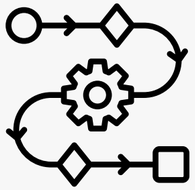
NICOLAS D'AVIAU DE TERNAY ET LOUIS CAGI NICOLAU

Summary

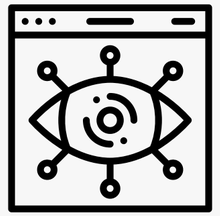
01 OVERVIEW

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:**

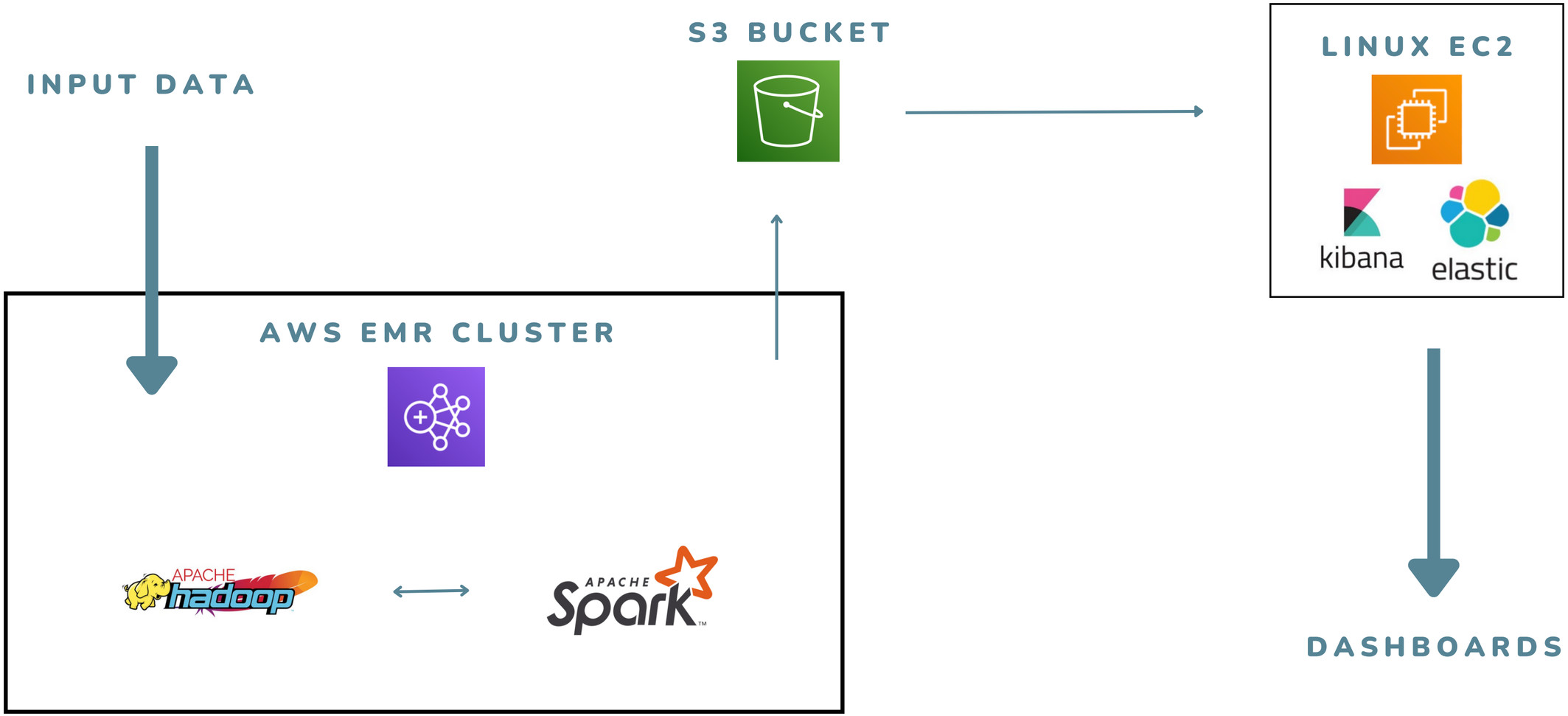
CONCEPTS STUDIED IN CLASS

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.

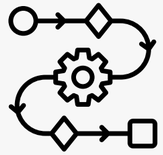
CLOUD TECHNOLOGY WITH AWS

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 Kaggel):

```
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

Content of the S3:

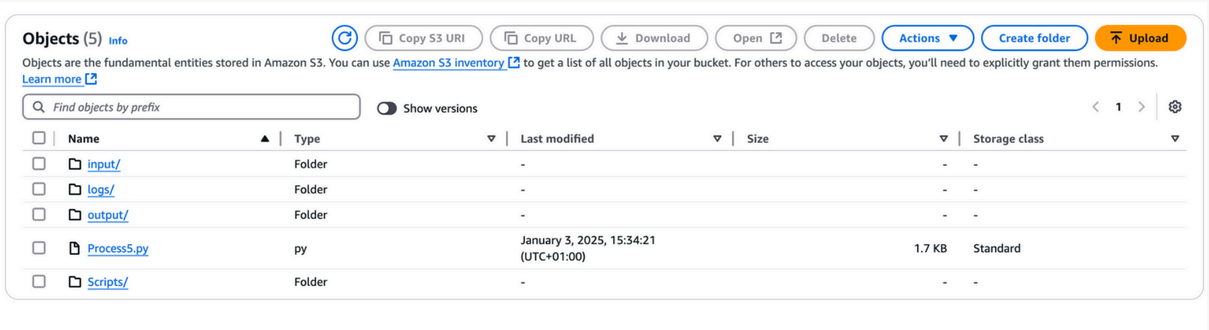
-input folder with our data.csv file

-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 ..



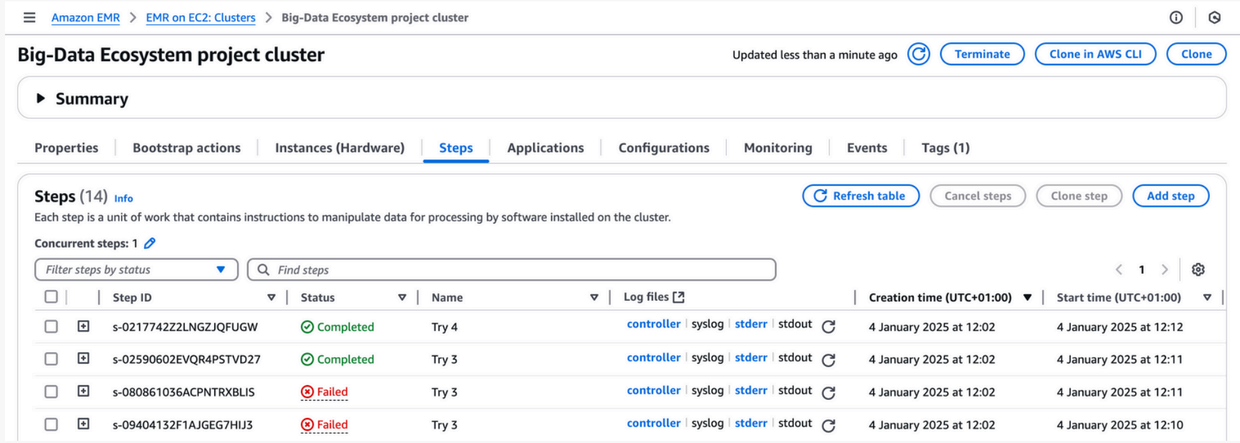
03 PIPELINE WORKFLOW

02. DATA TRANSFORMATION WITH SPARK

Users > nico > Desktop > Process5.py > transform_data

```python
def transform_data(data_source: str, output_uri: str) -> None:

    with SparkSession.builder.appName("Business Rides Data Transformation").getOrCreate() as spark:
        # Step 1: Read CSV file from S3
        df = spark.read.option("header", "true").csv(data_source)

        # Step 2: Perform data transformations
        # Convert columns to proper data types
        df = df.withColumn("MILES", col("MILES").cast("double"))


        df = df.withColumnRenamed("START", "start_location") \
               .withColumnRenamed("CATEGORY", "ride_category") \
               .withColumnRenamed("PURPOSE", "ride_purpose") \
               .withColumnRenamed("START_DATE", "start_date") \
               .withColumnRenamed("END_DATE", "end_date") \
               .withColumnRenamed("STOP", "end_location")


        df = df.withColumn("ride_length_category",
                           when(col("MILES") > 10, lit("long"))
                           .otherwise(lit("short")))

        # Step 3: Write intermediate results to HDFS
        hdfs_intermediate_path = "hdfs:///intermediate_data"
        df.write.mode("overwrite").parquet(hdfs_intermediate_path)  # Save in Parquet format for efficient storage

        print(f"Intermediate data written to HDFS at {hdfs_intermediate_path}")

        # Step 4: Read intermediate data from HDFS
        df_intermediate = spark.read.parquet(hdfs_intermediate_path)

        # Step 5: Write the final transformed data to S3 as a JSON file
        df_intermediate.write.mode("overwrite").json(f"{output_uri}/transformed_data.json")

        print(f"Transformed data written to {output_uri}/transformed_data.json")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--data_source", help="Path to the input CSV file")
    parser.add_argument("--output_uri", help="Path to save the transformed JSON file")
```

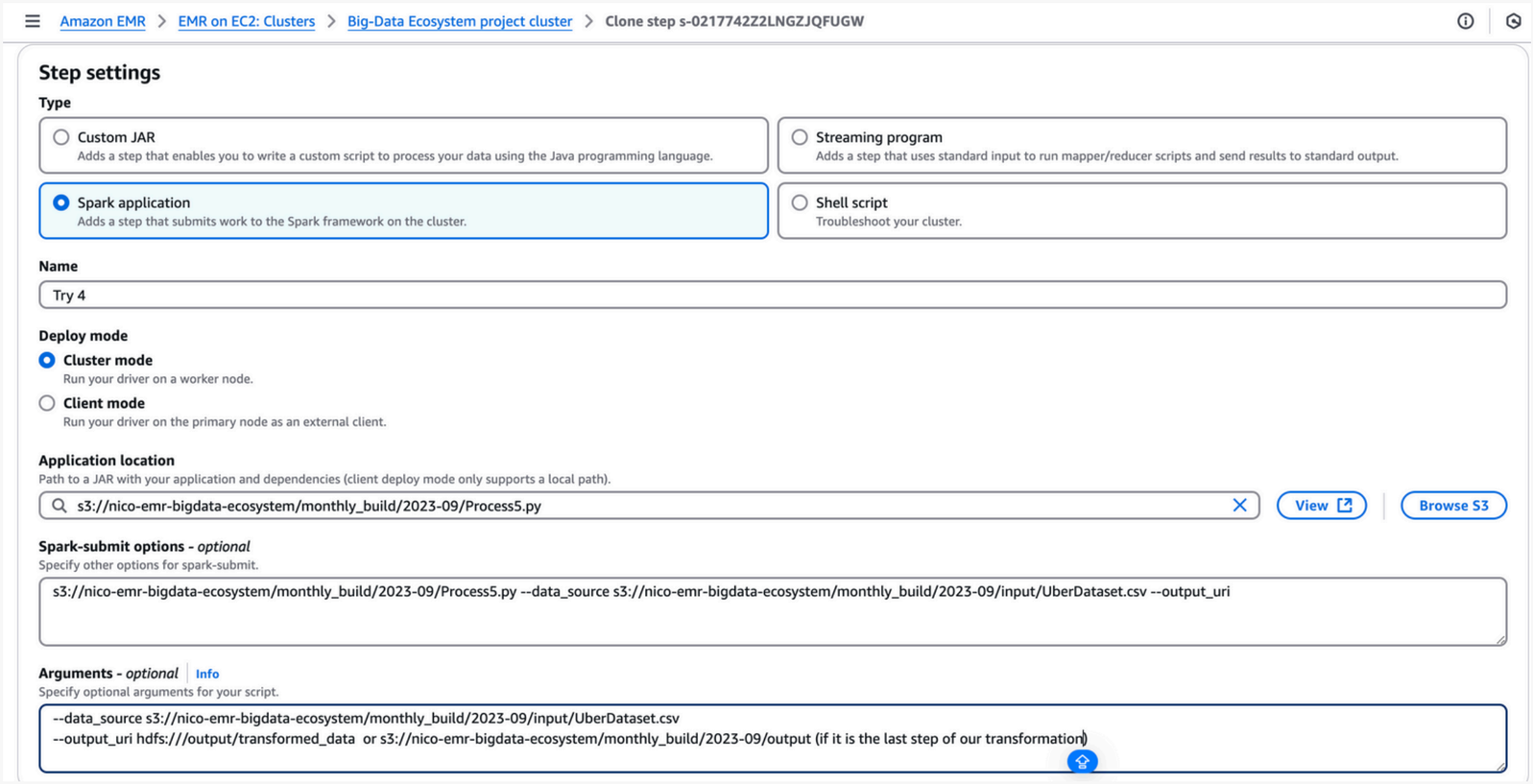
CODE ON GITHUB



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

03 PIPELINE WORKFLOW

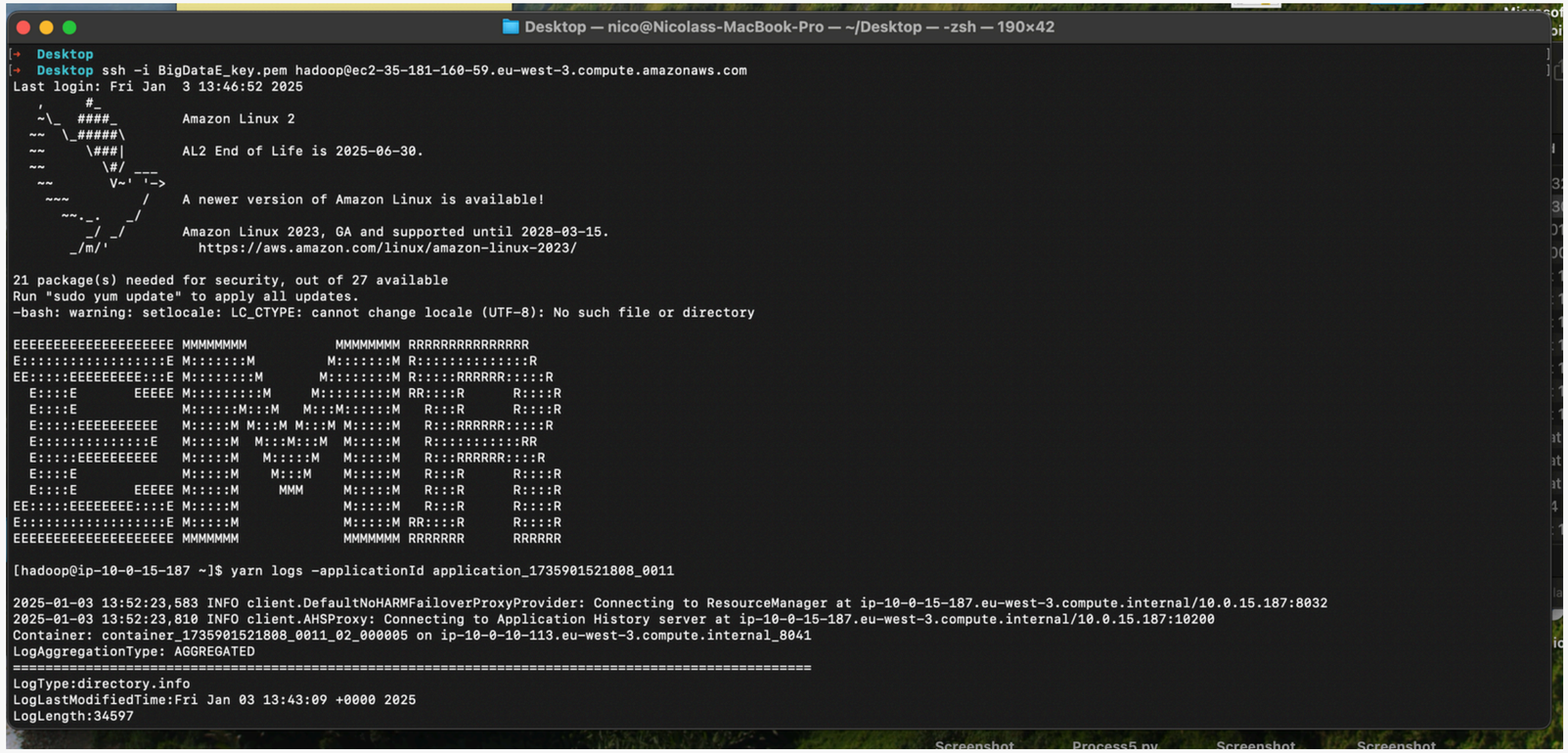
02. DATA TRANSFORMATION WITH SPARK



EXAMPLE OF STEP CONFIGURATION

03 PIPELINE WORKFLOW

03. DISTRIBUTED STORAGE WITH HADOOP



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:PsAkk4zQOlctJkxnd4x6oG1JeCQ/A+p2NH5lYs3/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

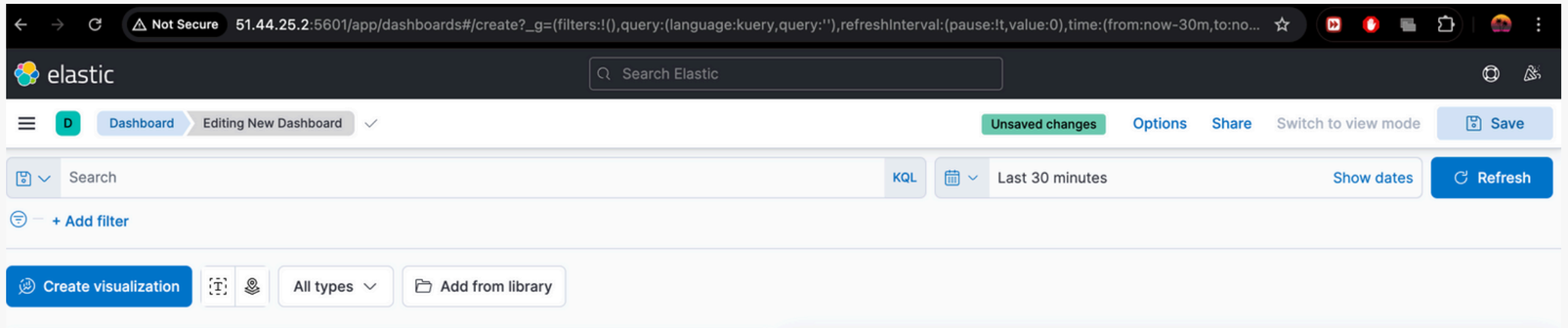
We install the ELK stack without Logstash because it is useless in our case , for that we install java on our machine and we configure the EC2 policy to accept trafic on port number 5601 (Kibana) and 9200 (Elastickseaerch)

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

03 PIPELINE WORKFLOW

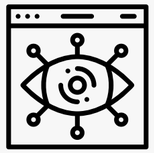
04. VISUALIZATION WITH KIBANA:

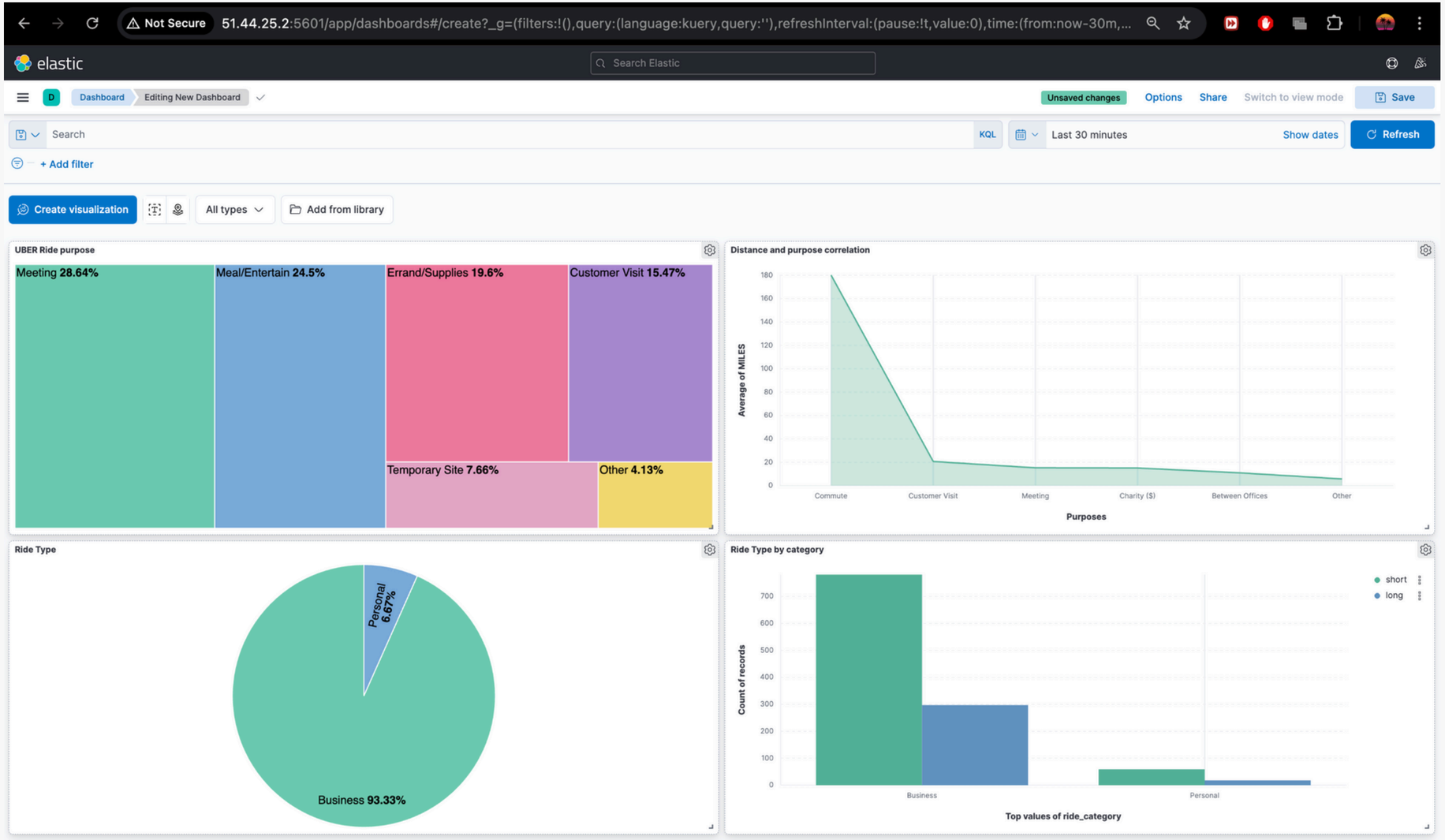
```
ubuntu@pod-elk:~$ sudo service kibana start
ubuntu@pod-elk:~$ sudo service kibana status
● kibana.service - Kibana
     Loaded: loaded (/etc/systemd/system/kibana.service; disabled; preset: enabled)
     Active: active (running) since Thu 2025-01-02 14:40:06 UTC; 4min 11s ago
       Docs: https://www.elastic.co
   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
```



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

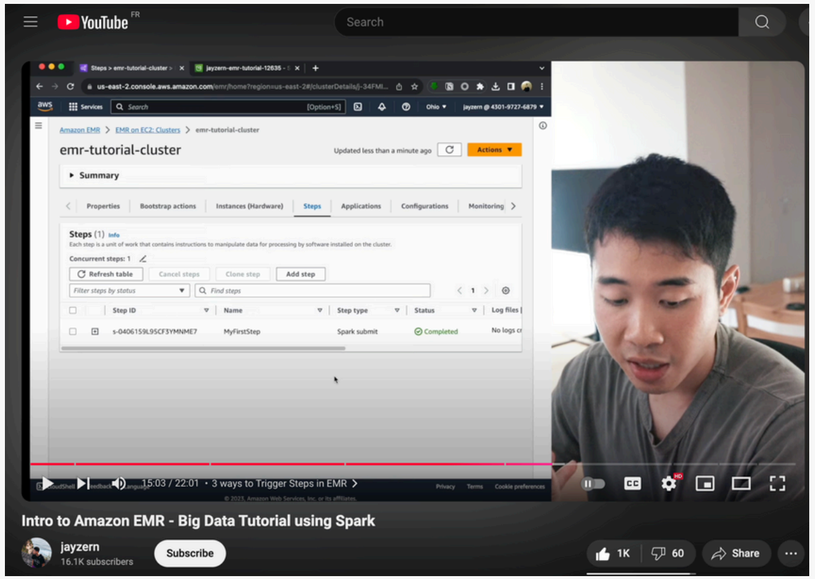
05 CHALLENGES ENCOUNTERED

01. GETTING STARTED WITH AWS EMR

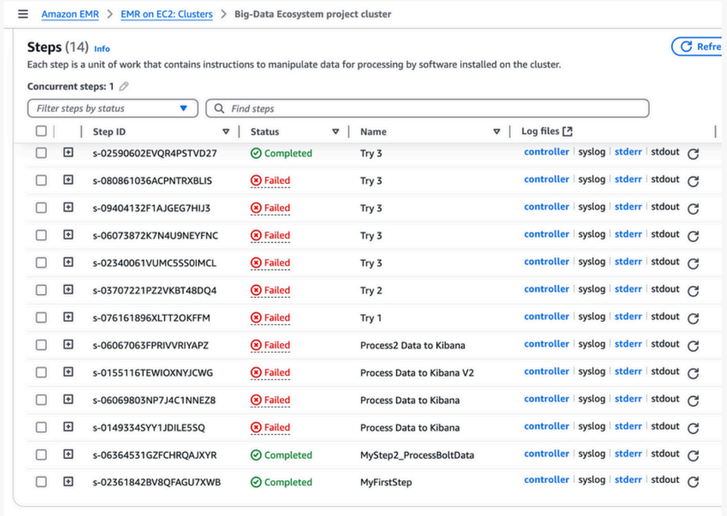
Working with AWS EMR was challenging at first. We spent considerable time and effort ensuring the script could correctly access input files in S3 and upload outputs to the right location. Small misconfigurations caused repeated failures, making the process frustrating.

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.



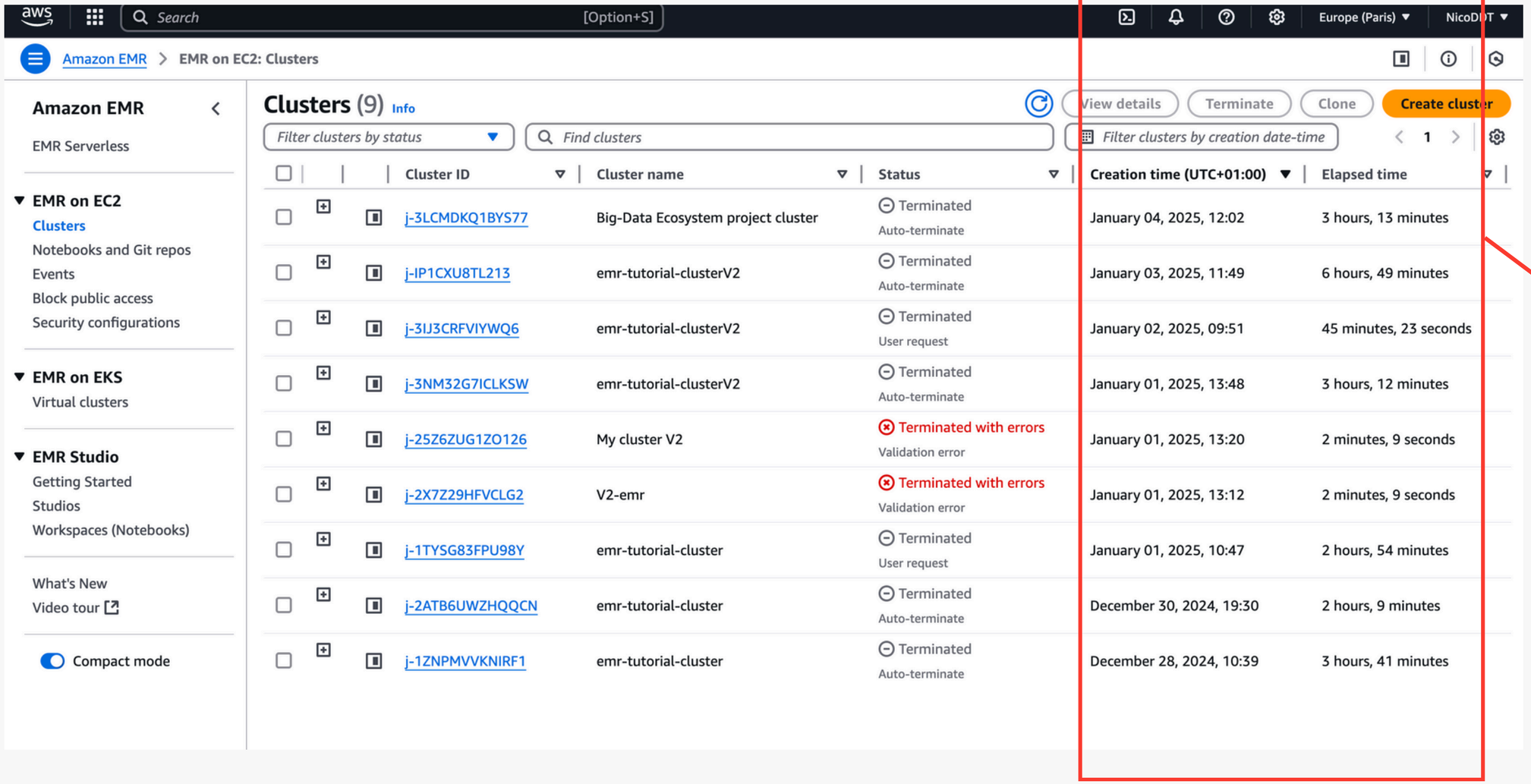
LINK HERE



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

05 CHALLENGES ENCOUNTERED

ECE
ÉCOLE D'INGÉNIEURS
ENGINEERING SCHOOL



it was very time-consuming overall

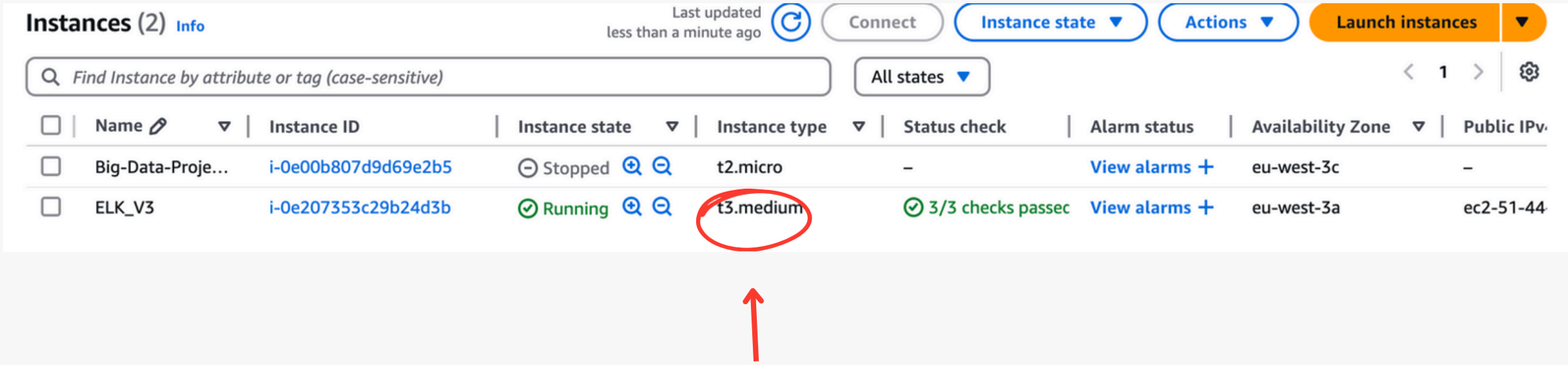
05 CHALLENGES ENCOUNTERED

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



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.