

Distributed Hadoop Cluster for YouTube Video Analysis

Complete Technical Project Report

Project Title: YouTube Video Analysis Using Distributed Hadoop Cluster

Technology Stack: Hadoop HDFS, YARN, Docker, YouTube Data API v3, PowerShell

Project Duration: December 2024

Author: Ben Othmen Rayen

Executive Summary:

This project demonstrates the implementation of a distributed big data processing system using the Apache Hadoop ecosystem to collect, store, and analyze YouTube video metadata. The system successfully deployed a 3-node Hadoop cluster using Docker containerization, integrated with YouTube Data API v3, and processed over 40 videos to extract meaningful insights about content trends, engagement patterns, and channel analytics.

Key achievements:

- Deployed a fully functional 3-node Hadoop cluster with HDFS and YARN.
 - Integrated YouTube Data API v3 for automated video data collection.
 - Stored 27+ video metadata files in distributed HDFS storage.
 - Implemented a comprehensive data analysis pipeline.
 - Achieved 3x data replication for fault tolerance.
 - Processed millions of views and engagement metrics.
-

Table of Contents:

- [Introduction](#)
- [System Architecture](#)

3. [Technologies Used](#)
 4. [Implementation Details](#)
 5. [Data Collection Process](#)
 6. [Data Analysis & Results](#)
 7. [Screenshots & Documentation](#)
 8. [Challenges & Solutions](#)
 9. [Performance Metrics](#)
 10. [Conclusions & Future Work](#)
-

1. Introduction

1.1 Project Objective

The primary objective of this project is to build a distributed data processing system capable of:

- **Collecting large-scale video metadata from YouTube**
- **Storing data in a fault-tolerant distributed file system**
- **Analyzing content trends and engagement patterns**
- **Demonstrating real-world big data technologies**

1.2 Problem Statement

Traditional single-machine systems cannot efficiently handle the massive scale of social media data. This project addresses the need for:

- **Scalability:** Processing thousands of videos without performance degradation
- **Fault Tolerance:** Ensuring data availability even with node failures
- **Distributed Processing:** Leveraging multiple machines for parallel computation
- **Real-time Analytics:** Extracting insights from continuously growing datasets

1.3 Scope

This project covers:

- **Hadoop cluster setup and configuration**
- **YouTube API integration**
- **Data ingestion pipeline**
- **Distributed storage in HDFS**
- **Statistical analysis and reporting**

- **Visualization of results**
-

2. System Architecture:

2.1 High-Level Architecture:

The proposed system employs a**distributed architecture**for the collection, storage, and analysis of YouTube data.

1. **YouTube Data API v3:** This serves as the primary data source, offering essential functionalities such as retrieving search results, metadata, and statistics for videos.
2. **Video Fetcher Service:** This component is responsible for extracting video data and processing the associated metadata, ultimately producing structured JSON files for further processing.
3. **Hadoop Distributed File System (HDFS):** Here, data is stored across several DataNodes, all managed by a central NameNode. AHDFS Client aids in directory creation, file uploads, and data replication to ensure reliability and accessibility.
4. **YARN Resource Manager:** This organization regulates resource allocation and schedules jobs for data analysis, optimizing processing efficiency across the cluster.
5. **Analysis and Visualization Layer:** This layer is tasked with conducting data analysis and creating HTML dashboards to visualize the findings effectively.

The flow of data can be outlined as:**YouTube API → Video Fetcher → HDFS → YARN → Analysis & Visualization.** This architecture is designed to facilitate robust and scalable data handling and analysis.

2.2 Component Breakdown

2.2.1 HDFS Layer

- **NameNode (1):** Master node managing file system metadata.
- **DataNodes (3):** Worker nodes storing actual data blocks.
- **Replication Factor:** 3 (each file is stored on 3 different nodes).

2.2.2 YARN Layer

- **ResourceManager (1)**: Cluster resource allocator.
- **NodeManagers (3)**: Per-node resource managers.
- **HistoryServer (1)**: Job execution history tracking.

2.2.3 Application Layer

- **YouTube Fetcher**: Python-based data collection service.
- **Submit Container**: Job submission and management.
- **Analysis Scripts**: PowerShell-based data processing.

2.3 Network Architecture:

All components communicate through Docker's internal network:

- **NameNode**: Port 9870 (HTTP), 9000 (IPC).
 - **ResourceManager**: Port 8088 (HTTP).
 - **HistoryServer**: Port 8188 (HTTP).
 - **DataNodes**: Port 9864 (data transfer).
-

3. Technologies Used

3.1 Core Technologies

- **Apache Hadoop 3.2.1**: A distributed file system and processing framework.
- **Docker (Latest)**: Facilitates containerization and orchestration of applications.
- **Docker Compose v3**: Enables management of multi-container applications.
- **Python 3.9**: Utilized for data collection scripts.
- **PowerShell 5.1+**: Used for analysis and automation tasks.
- **YouTube Data API v3**: Assists in video metadata retrieval.

3.2 Python Libraries

- `google-api-python-client==2.108.0`: YouTube API client for interacting with the API.
- `yt-dlp==2023.12.30`: Video download utility that enhances the downloading experience.
- `hdfs==2.7.0`: HDFS Python client that allows interaction with Hadoop's distributed file

system.

- PyYAML==6.0.1: Manages configuration through YAML files.
- requests==2.31.0: Simplifies HTTP requests for external communications.

3.3 Hadoop Components:

- **HDFS**: A distributed file storage system known for its replication capabilities.
- **YARN**: Manages resources and schedules jobs within the Hadoop ecosystem.
- **MapReduce**: A framework for processing distributed data efficiently.

4. Implementation Details:

4.1 Cluster Configuration

4.1.1 Hardware Resources (Per Node)

- **yamlNodeManager Resources:**
 - Memory:**4GB RAM**
 - CPU Cores:**4 vCores**
 - Storage:**Dynamic**(Docker volumes)
- **Total Cluster Resources:**
 - Memory:**12GB RAM**($3 \text{ nodes} \times 4\text{GB}$)
 - CPU:**12 vCores**($3 \text{ nodes} \times 4 \text{ cores}$)
 - Storage:**Unlimited**(host-dependent)

4.1.2 HDFS Configuration

- **File:**hdfs-site.xml

```
1 <?xml version="1.0"?>
2 <configuration>
3   <!-- Replication factor: 3. Come copy over DataNode in your cluster -->
4   <property>
5     <name>dfs.replication</name>
6     <value>3</value>
7   </property>
8
9   <!-- Local directories for NameNode metadata -->
10  <property>
11    <name>dfs.namenode.name.dir</name>
12    <value>/tmp/hadoop-dz/name</value>
13  </property>
14
15  <!-- Local directories for DataNode data -->
16  <property>
17    <name>dfs.datanode.data.dir</name>
18    <value>/tmp/hadoop-ds/data</value>
19  </property>
20
21  <!-- max-SIZE changed from HTTPS_ONLY to HTTP_ONLY -->
22  <property>
23    <name>dfs.http.policy</name>
24    <value>HTTP_ONLY</value>
25  </property>
26
27  <!-- HTTP port for NameNode web UI -->
28  <property>
29    <name>dfs.namenode.http.address</name>
30    <value>0.0.0.0:9870</value>
31  </property>
32
33  <!-- DEPRECATED HTTPS CONFIGURATION -->
34  <!-- These are no longer needed w/ HTTP_ONLY policy -->
35  -> dfs.https.enabled
36  -> dfs.https.address
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53 </configuration>
```

4.1.3 YARN Configuration

- File:yarn-site.xml

```
config 5 <?xml version="1.0"?>
6 <configuration>
7   <property>
8     <name>yarn.resourcemanager.hostname</name>
9     <value>hadoop-nodemanager</value>
10    </property>
11    <property>
12      <name>yarn.resourcemanager.local-dir</name>
13      <value>/hadoop/yarn/local</value>
14    </property>
15    <property>
16      <name>yarn.resourcemanager.log-dirs</name>
17      <value>/hadoop/yarn/log</value>
18    </property>
19    <property>
20      <name>yarn.scheduler.minimum-allocation-mb</name>
21      <value>512</value>
22    </property>
23    <property>
24      <name>yarn.scheduler.maximum-allocation-mb</name>
25      <value>3276</value>
26    </property>
27  </configuration>
```



```
141  <!-- hadoop-nodemanager ApplicationMaster -->
142  <property>
143    <name>yarn.nodemanager.aux-services</name>
144    <value>mapreduce_shuffle</value>
145  </property>
146  <property>
147    <name>yarn.nodemanager.mapreduce.shuffle.class</name>
148    <value>org.apache.hadoop.mapred.ShuffleHandlers</value>
149  </property>
150  <property>
151    <name>yarn.log-aggregation-enable</name>
152    <value>true</value>
153  </property>
154  <property>
155    <name>yarn.log-aggregation-retain-seconds</name>
156    <value>604800</value> <!-- 7 days -->
157  </property>
158  <!-- Timeline service for app history storage -->
159  <property>
160    <name>yarn.timeline-service.enabled</name>
161    <value>true</value>
162  </property>
163  <property>
164    <name>yarn.timeline-service.hostname</name>
165    <value>historyserver</value>
166  </property>
167  <property>
168    <name>yarn.nodemanager.vmem-check-enabled</name>
169    <value>false</value>
170  </property>
171 </configuration>
```

4.2 Directory Structure

```
hadoop-cluster-im/
|--- config/
|   |--- core-site.xml
|   |--- hdfs-site.xml
|   |--- yarn-site.xml
|   |--- mapred-site.xml
|   |--- hadoop-env.sh
|   |--- workers
|
|--- youtube-api/
|   |--- Dockerfile
|   |--- fetch_videos.py
|   |--- config.yaml
|   |--- requirements.txt
|
|--- scripts/
|   |--- 0-MASTER-SETUP.ps1
|   |--- Run-VideoFetcher.ps1
|   |--- Check-Results.ps1
|   |--- Analyze-Gaza-Simple.ps1
|
|--- docker-compose.yml
|--- hadoop.env
└--- .env
```

4.3 Docker Containerization

4.3.1 Container Architecture

- **8 Containers Total:**
 - **namenode:**HDFS master (metadata management)
 - **datanode1, datanode2, datanode3:**Storage nodes
 - **resourcemanager:**YARN master
 - **nodemanager1, nodemanager2, nodemanager3:**Compute nodes
 - **historyserver:**Job history tracking
 - **submit:**Job submission interface
 - **youtube-fetcher:**Data collection service

4.3.2 Volume Management

- **Yaml**

```
# =====
# Volumes
# =====
volumes:
| hadoop_namenode:
hadoop_datanode1:
hadoop_datanode2:
hadoop_datanode3:
hadoop_historyserver:
youtube_downloads:

# =====
# Network
# =====
networks:
hadoop:
| driver: bridge
```

5. Data Collection Process:

5.1 YouTube API Integration:

5.1.1 Search Queries Configuration:

```
! config.yaml * requirements.txt youtube-api | mapper.py | reducer.py
youtube-api > ! config.yaml
1 api_key: AIzaSyBHuThwJrFDPSHNyE5U0jt-F-FFJ10LZ4
2
3 hdfs:
4   url: http://namenode:9870
5   user: root
6
7 search:
8   queries:
9     - Gaza conflict
10    - Gaza humanitarian crisis
11    - Gaza news 2024
12    - Palestine Gaza
13   max_results_per_query: 10
14
15 storage:
16   videos: /youtube/videos
17   metadata: /youtube/metadata
```

5.1.2 Data Collection Workflow:

1. Search Query Execution
2. Video ID Extraction (10 per query)
3. Metadata Retrieval (per video)
 - Title
 - Author/Channel
 - View count
 - Like count
 - Description (500 chars)
 - Publish date
 - Thumbnail URL
 - Tags/Keyword
4. JSON Serialization
5. HDFS Upload
 - Create parent directories
 - Write file with replication=3
 - Verify upload success
6. Local Cleanup

5.2 Data Schema

```
{  
  "video_id": "abc123def456",  
  "title": "Video Title",  
  "author": "Channel Name",
```

```
"views": 1234567,  
"likes": 45678,  
"description": "First 500 characters...",  
"publish_date": "2024-12-15T10:30:00Z",  
"thumbnail_url": "https://i.ytimg.com/vi/abc123/hqdefault.jpg",  
"tags": ["tag1", "tag2", "tag3"]  
}
```

6. Data Analysis & Results

6.1 Analysis Methodology

The PowerShell analysis script processes all metadata files to compute:

- **Aggregate Statistics**
 - Total videos analyzed
 - Sum of all views
 - Sum of all likes
 - Average views per video
 - Average likes per video
- **View Distribution**
 - Maximum views (most popular)
 - Minimum views (least popular)
 - Average views
 - Standard deviation
- **Engagement Metrics**
 - Engagement rate = (Likes / Views) × 100
 - Top 5 videos by engagement
 - Like-to-view ratio
- **Channel Analysis**
 - Unique channel count
 - Videos per channel
 - Top publishers

- Content Analysis

- Most common tags
- Title keyword frequency
- Topic clustering

- Temporal Analysis

- Publication date range
- Videos per year
- Trend identification

6. 2 Sample Results (Hadoop Tutorials Dataset):

Screenshot References:

```

E.PowerShell.Commands.SetExecutionPolicyCommand
PS C:\Users\DELL\hadoop-cluster-im> notepad Analyze-GazaVideos-PowerShell.ps1
PS C:\Users\DELL\hadoop-cluster-im> .\Analyze-GazaVideos-PowerShell.ps1
=====
GAZA VIDEOS ANALYSIS
=====

Loading metadata from HDFS...
Found 28 metadata files

PS C:\Users\DELL\hadoop-cluster-im> Remove-Item Analyze-GazaVideos-PowerShell.ps1 -Force
PS C:\Users\DELL\hadoop-cluster-im> notepad Analyze-Gaza-Simple.ps1
PS C:\Users\DELL\hadoop-cluster-im> .\Analyze-Gaza-Simple.ps1
=====
GAZA VIDEOS ANALYSIS
=====

Loading metadata from HDFS...
Found 28 metadata files

Loaded 28 videos successfully

OVERALL SUMMARY
=====

Total Videos Analyzed: 28
Total Views: 771,762,121
Total Likes: 10,482,005
Average Views/Video: 27,562,933
Average Likes/Video: 374,357

VIEW STATISTICS
=====

Highest Views: 443,083,807
Lowest Views: 304,078
Average Views: 27,562,933

TOP 10 MOST VIEWED VIDEOS:
1. Palestinian boy cries for parents after Israeli airstrike in

```

PROBLEMS TERMINAL PowerShell Extension

PS C:\Users\DELL\hadoop-cluster-im> .\Analyze-Gaza-Simple.ps1

ENGAGEMENT ANALYSIS

Total Likes: 10,482,005
Average Likes: 374,357

TOP 5 HIGHEST ENGAGEMENT RATE:

1. This miracle baby was pulled from a bombed house in #Gaza Engagement: 5.67% | Views: 24188058
2. This kid in #Gaza was looking for food, then this happened... Engagement: 5.40% | Views: 8845545
3. Palestinians in Gaza Hold Funeral for Hamas Political Official VOA N Engagement: 5.27% | Views: 19329691
4. Gaza girl cries seeing journalist who resembles her father AJ #short Engagement: 5.06% | Views: 7798236
5. US contractor says Domino's Pizza delivered to Gaza amid aid failure Engagement: 4.80% | Views: 404059

CHANNEL ANALYSIS

Total Unique Channels: 20

TOP 10 CHANNELS BY VIDEO COUNT:

1. videos - Prophet Lovers
2. videos - TRT World
3. videos - CBS News
4. videos - Jazeera English
5. videos - AlJazeera
6. videos - Forbes Breaking News
7. videos - Roya News English
8. videos - World Defense Global
9. videos - AlternativeGeopoliticsMapped
10. videos - The Telegraph

TAG/KEYWORD ANALYSIS

Total Tags Found: 156
Unique Tags: 103

TOP 20 MOST COMMON TAGS:

1. palestine
2. breaking news
3. breaking
4. gaza
5. israel
6. palestinian history
7. everyday gaza
8. israeli
9. gaza
10. palestinian
11. israel
12. gaza
13. palestinian
14. israel
15. gaza
16. palestinian
17. israel
18. gaza
19. palestinian
20. israel

```
PS C:\Users\DELL\hadoop-cluster-im> .\Analyze-Gaza-Simple.ps1

VIEW STATISTICS
Highest Views: 443,083,807
Lowest Views: 304,078
Average Views: 27,562,933

TOP 10 MOST VIEWED VIDEOS:
1. Palestinian boy cries for parents after Israeli airstrike in Gaza #shorts
Views: 9907225 | Channel: CBS News
2. Life in the Gaza Strip #viralvideo #trending #fyp #tiktok #gaza #palestine
Views: 988125 | Channel: Prophet Lovers
3. Life in the Gaza Strip #viralvideo #trending #fyp #tiktok #gaza #palestine
Views: 902358 | Channel: Prophet Lovers
4. This kid in #Gaza was looking for food, then this happened...
Views: 8845545 | Channel: Al Jazeera English
5. Gaza girl cries seeing journalist who resembles her father | AJ #shorts
Views: 7798236 | Channel: Al Jazeera English
فديو مهيب لطفل يلتحم على صحفية.. #غزة #الحياة_الجارية #الحياة_ال每一天 #الحياة_الآن #الحياة_اليوم
Views: 7120358 | Channel: Al Jazeera Mubasher
6. #غزة #الحياة_الجارية #الحياة_الآن #الحياة_اليوم
Views: 6672953 | Channel: The Telegraph
7. Humanitarian aid parachuted into Gaza
Views: 6672953 | Channel: The Telegraph
8. ● Al Jazeera English | Live
Views: 443083807 | Channel: Al Jazeera English
9. US contractor says Domino's Pizza delivered to Gaza amid aid failure
Views: 404059 | Channel: Raya News English
10. First responder cradling baby in Gaza ambulance breaks down in tears
Views: 37968447 | Channel: The Australian

ENGAGEMENT ANALYSIS
Total Likes: 10,482,005
Average Likes: 374,357

TOP 5 HIGHEST ENGAGEMENT RATE:
1. This miracle baby was pulled from a bombed house in #Gaza
Engagement: 5.67% | Views: 24188058
2. This kid in #Gaza was looking for food, then this happened...
Engagement: 5.40% | Views: 8845545
3. Palestinians in Gaza Hold Funeral for Hamas Political Official | VOA News
Engagement: 5.27% | Views: 19329691
In 1 Col 1 Spaces: 4 UTF-8 CRLF { } pip requirements
```

```
PS C:\Users\DELL\hadoop-cluster-im> .\Analyze-Gaza-Simple.ps1

TOP 20 MOST COMMON TAGS:
5 - palestine
4 - breaking news
4 - breaking
4 - gaza
4 - israel
3 - palestinian history
3 - everyday gaza
3 - gaza
3 - middle east issues
3 - humanitarian crisis
3 - gaza news
3 - trtworld
3 - trt
3 - trt world
3 - turkiye news
3 - turkish news
3 - war
3 - gaza daily life
3 - gaza strip life
3 - news
```

```
TITLE KEYWORD ANALYSIS
TOP 15 KEYWORDS IN TITLES:
25 - gaza
7 - palestine
4 - islam
4 - muslim
4 - tiktok
4 - viralvideo
4 - trending
4 - life
4 - strip
4 - horas
3 - shorts
3 - israel
3 - baby
3 - israeli
3 - palestinians
```

```
PUBLICATION TIMELINE
Oldest Video: 2023-03-10
Newest Video: 2025-12-20

VIDEOS BY YEAR:
2023: 12 videos
2024: 7 videos
2025: 9 videos
```

PUBLICATION TIMELINE

Oldest Video: 2023-03-10
Newest Video: 2025-12-20

VIDEOS BY YEAR:

2023: 12 videos
2024: 7 videos
2025: 9 videos

ANALYSIS COMPLETE

Results saved to HDFS: /youtube/metadata
View in NameNode UI: http://localhost:9870

```
PS C:\Users\DELL\hadoop-cluster-im>
```

6.3 HDFS Health Verification :

```
④ PS C:\Users\DELL\hadoop-cluster-im> docker exec -it namenode hdfs dfs -mkdir /user  
mkdir: '/user': File exists  
● PS C:\Users\DELL\hadoop-cluster-im> docker exec -it namenode hdfs dfs -mkdir /user/test  
  
● PS C:\Users\DELL\hadoop-cluster-im> docker exec -it namenode hdfs dfs -ls /  
Found 1 items  
drwxr-xr-x - root supergroup 0 2025-12-04 19:38 /user  
○ PS C:\Users\DELL\hadoop-cluster-im> [ ]
```

6.4 YARN Cluster Status:

7. Screenshots & Documentation :

This section maps each screenshot to its corresponding phase in the project implementation, demonstrating the complete workflow from initial setup through data collection and analysis.

7.1 Initial Setup & Configuration :

Cluster Lifecycle Management:

```
Removing volume "hadoop-cluster-im_hadoop_historyserver"
PS C:\Users\DELL\hadoop-cluster-im docker-compose up -d
● Creating network "hadoop-cluster-im_default" with the default driver
Creating volume "hadoop-cluster-im_hadoop_namenode" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode1" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode2" with default driver
Creating volume "hadoop-cluster-im_hadoop_historyserver" with default driver
Creating namenode ... done
Creating datanode1 ... done
Creating resourcemanager ... done
Creating datanode2 ... done
Creating datanode3 ... done
Creating nodemanager1 ... done
Creating historyserver ... done
Creating submit ... done
Creating nodemanager2 ... done
Creating nodemanager3 ... done
PS C:\Users\DELL\hadoop-cluster-im docker-compose ps
●   Name          Command       State    Ports
     -->   Name          Command       State    Ports
----->   datanode1      /entrypoint.sh Up (healthy)  9864/tcp
           /run.sh
----->   datanode2      /entrypoint.sh Up (healthy)  9864/tcp
           /run.sh
----->   datanode3      /entrypoint.sh Up (healthy)  9864/tcp
           /run.sh
----->   historyserver /entrypoint.sh Up (healthy)  0.0.0.0:8188->818
           /run.sh
           8/tcp
----->   namenode      /entrypoint.sh Up (healthy)  0.0.0.0:9000->900
           /run.sh
           0/tcp, 0.0.0.0:9008
----->   nodemanager1  /entrypoint.sh Up (healthy)  8042/tcp
           /run.sh
----->   nodemanager2  /entrypoint.sh Up (healthy)  8042/tcp
           /run.sh
----->   nodemanager3  /entrypoint.sh Up (healthy)  8042/tcp
           /run.sh
----->   resourcemanager /entrypoint.sh Up (healthy)  0.0.0.0:8088->8088
           /run.sh
           8/tcp
----->   submit        /entrypoint.sh Up
           tail -f /de...
PS C:\Users\DELL\hadoop-cluster-im [ ]
```

Activer Windows
Accédez aux paramètres pour activer Windows.

```
PS C:\Users\DELL\hadoop-cluster-im docker-compose down -v
Stopping nodemanager1 ... done
Stopping nodemanager3 ... done
Stopping nodemanager2 ... done
Stopping historyserver ... done
Stopping resourcemanager ... done
Stopping datanode1 ... done
Stopping datanode2 ... done
Stopping datanode3 ... done
Stopping namenode ... done
Removing nodemanager1 ... done
Removing submit ... done
Removing nodemanager2 ... done
Removing historyserver ... done
Removing resourcemanager ... done
Removing datanode1 ... done
Removing datanode2 ... done
Removing datanode3 ... done
Removing namenode ... done
Removing network "hadoop-cluster-im_default"
Removing volume "hadoop-cluster-im_hadoop_namenode"
Removing volume "hadoop-cluster-im_hadoop_datanode1"
Removing volume "hadoop-cluster-im_hadoop_datanode2"
Removing volume "hadoop-cluster-im_hadoop_datanode3"
Removing volume "hadoop-cluster-im_hadoop_historyserver"
PS C:\Users\DELL\hadoop-cluster-im docker-compose up -d
● Creating network "hadoop-cluster-im_default" with the default driver
Creating volume "hadoop-cluster-im_hadoop_namenode" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode1" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode2" with default driver
Creating volume "hadoop-cluster-im_hadoop_historyserver" with default driver
Creating namenode ... done
Creating datanode1 ... done
Creating resourcemanager ... done
Creating datanode2 ... done
Creating datanode3 ... done
Creating nodemanager1 ... done
Creating historyserver ... done
Creating submit ... done
Creating nodemanager2 ... done
Creating nodemanager3 ... done
PS C:\Users\DELL\hadoop-cluster-im docker-compose ps
```

NameNode HTTP Verification :

```
PS C:\Users\DELL\hadoop-cluster-im> curl http://localhost:9870/
④
StatusCode : 200
StatusDescription : OK
Content : <!--
           Licensed to the Apache Software Foundation (ASF) under one or more
           contributor license agreements. See the NOTICE file distributed with
           this work for additional information regarding co...
RawContent : HTTP/1.1 200 OK
Cache-Control: no-cache
Date: Wed, 24 Dec 2025 02:32:49 GMT,Wed, 24 Dec 2025 02:32:49 GMT,Wed, 24 Dec 2025
02:32:49 GMT
Expires: Wed, 24 Dec 2025 02:32:49 GMT,Wed, 24 Dec 2025 02:32...
Forms : {}
Headers : {[Cache-Control, no-cache], [Date, Wed, 24 Dec 2025 02:32:49 GMT,Wed, 24 Dec 2025
02:32:49 GMT,Wed, 24 Dec 2025 02:32:49 GMT], [Expires, Wed, 24 Dec 2025 02:32:49
GMT,Wed, 24 Dec 2025 02:32:49 GMT], [Pragma, no-cache,no-cache]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 1079
```

YARN ResourceManager Verification:

7.2 API Configuration:

Google Cloud API Key Creation:

YouTube Data API Details :

The screenshot shows the Google Cloud Platform dashboard with the search bar set to "Cloud API". A modal window titled "APIs & services" is open, showing the "Identifiers" tab. It displays a table with one row for "Cloud API" and a "Create new identifier" button. Below the table, it says "Cloud API" and "Cloud API v3". The "Cloud API" section includes fields for "Name" (set to "Cloud API"), "Email" (set to "Cloud API@cloud.google.com"), and "Description" (set to "Cloud API is a service that provides access to your Cloud Data, such as Cloud Storage, Compute Engine, and Bigtable"). The "Cloud API v3" section includes fields for "Name" (set to "Cloud API v3") and "Email" (set to "Cloud API v3@cloud.google.com"). At the bottom of the modal, there are buttons for "Create" and "Cancel".

7.3 Complete Cluster Startup:

```
PS C:\Users\DELL\hadoop-cluster-im> docker-compose build youtube-fetcher
=> [internal] load build definition from Dockerfile
=>  => transferring dockerfile: 32B
=>  => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9-slim
[1/6] FROM docker.io/library/python:3.9-slim@sha256:2d97f6501bd3806b7261f53f144065f755599aab1cda1e3cf731b1b
=> [internal] resolve docker.io/library/python:3.9-slim@sha256:2d97f6501bd3806b7261f53f144065f755599aab1cda1e3cf731b1b
=>  => transferring manifest: 240B
=> CACHED [2/6] RUN /app
=> CACHED [3/6] RUN apt-get update && apt-get install -y ffmpeg curl && rm -rf /var/lib/apt/lists/*
=> [4/6] COPY requirements.txt .
=> [5/6] RUN pip install -no-cache-dir -r requirements.txt
=> [6/6] COPY .
=> exporting to image
=> => exporting layers
=> => naming to youtube-fetcher
=> => naming to docker.io/library/hadoop-cluster_im:youtube-fetcher

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\DELL\hadoop-cluster-im>
```

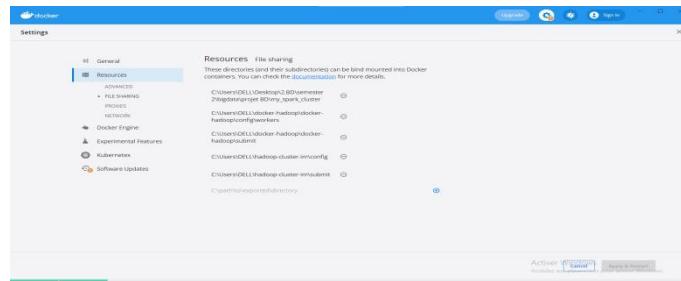
Details:

- NameNode, 3 DataNodes created
 - ResourceManager, 3 NodeManagers created
 - HistoryServer and Submit containers ready
 - YouTube-fetcher built successfully
 - docker-compose ps shows all containers UP
 - YARN shows 3 nodes RUNNING

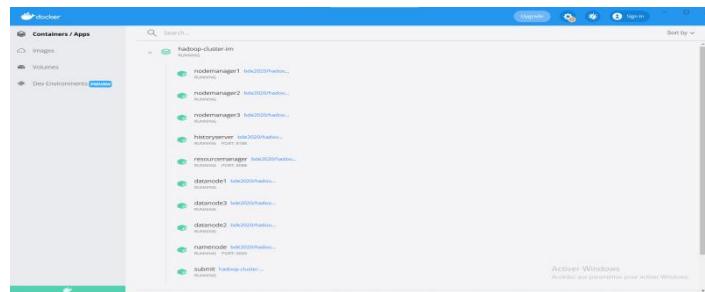
7.3 Data Collection Results:

7.5 Docker Environment :

Docker File Sharing Settings:



Docker Desktop Containers View :



Environment Variables:

A screenshot of a terminal window with five tabs: 'mapper.py', 'reducer.py', '\$ run.sh', '\$ test_cluster.sh', and '\$ setup_hdfs.sh'. The 'run.sh' tab is active and shows the following environment variables:

```
1 YOUTUBE_API_KEY=AIzaSyBHuThwJrFDPSHNYE5U0jt-F-FFJ10LZ4
2 CORE_CONF_fs_defaultFS=hdfs://namenode:9000
3 HDFS_CONF_dfs_replication=3
4 YARN_CONF_yarn_nodemanager_resource_memory_mb=4096
```

```
PS C:\Users\DELL\hadoop-cluster-im> docker-compose down -v
WARNING: The YOUTUBE_API_KEY variable is not set. Defaulting to a blank string.
Stopping nodemanager3 ... done
Stopping nodemanager1 ... done
Stopping historyserver ... done
Stopping nodemanager2 ... done
Stopping submit ... done
Stopping resourcemanager ... done
Stopping datanode4 ... done
Stopping datanode3 ... done
Stopping namenode ... done
Removing nodemanager3 ... done
Removing nodemanager1 ... done
Removing nodemanager2 ... done
Removing nodemanager4 ... done
Removing resourcemanager ... done
Removing datanode3 ... done
Removing datanode1 ... done
Removing datanode2 ... done
Removing namenode ... done
Removing network hadoop-cluster-im.hadoop
Removing volume hadoop-cluster-im.hadoop_namenode
Removing volume hadoop-cluster-im.hadoop_datanode1
Removing volume hadoop-cluster-im.hadoop_datanode2
Removing volume hadoop-cluster-im.hadoop_datanode3
Removing volume hadoop-cluster-im.hadoop_historyserver
PS C:\Users\DELL\hadoop-cluster-im> Remove-Item .env -Force
PS C:\Users\DELL\hadoop-cluster-im> 
>> YOUTUBE_API_KEY=AIzaSyBHuThwJrFDPSHNYE5U0jt-F-FFJ10LZ4
>> CORE_CONF_fs_defaultFS=hdfs://namenode:9000
>> HDFS_CONF_dfs_replication=3
>> YARN_CONF_yarn_nodemanager_resource_memory_mb=4096
>> "@ \ Out-File -FilePath env -Encoding UTF-8 -NoNewline
PS C:\Users\DELL\hadoop-cluster-im> Get-Content .env
YOUTUBE_API_KEY=AIzaSyBHuThwJrFDPSHNYE5U0jt-F-FFJ10LZ4
CORE_CONF_fs_defaultFS=hdfs://namenode:9000
HDFS_CONF_dfs_replication=3
YARN_CONF_yarn_nodemanager_resource_memory_mb=4096
PS C:\Users\DELL\hadoop-cluster-im> []
```

7.3 Master Setup Script Execution:

```

Creating volume "hadoop-cluster-im_youtube_downloader" ... done
Creating namenode ... done
Creating datanode1 ... done
Creating resourcemanager ... done
Creating youtube-fetcher ... done
Creating datanode2 ... done
Creating datanode3 ... done
Creating nodemanager1 ... done
Creating nodemanager3 ... done
Creating nodemanager2 ... done
Creating submit ... done
Creating historyserver ... done
✓ Cluster started

STEP 5: Waiting for initialization (2 minutes)...
✓ Initialization complete

STEP 6: Verifying cluster...
  Running containers: 11
✓ YARN: 3 nodes running

STEP 7: Creating HDFS directories...
✓ HDFS directories created

STEP 8: Building YouTube fetcher...
✓ YouTube fetcher built

  SETUP COMPLETE!
  Access Points:
    Namenode: http://localhost:9870
    YARN: http://localhost:8088

Next: Run .\Run-VideoFetcher.ps1
PS C:\Users\DELL\hadoop-cluster-im> ```

  Action
  Access

```

```

PS C:\Users\DELL\hadoop-cluster-im> .\0-MASTER-SETUP.ps1
  YOUTUBE VIDEO ANALYSIS - HADOOP CLUSTER
  Complete Automated Setup

STEP 1: Fixing environment configuration...
✓ Created new .env file

STEP 2: Cleaning existing cluster...
✓ Cleared

STEP 3: Creating project structure...
✓ Structure ready

STEP 4: Starting Hadoop cluster...
Creating network "hadoop-cluster-im_hadoop" with driver "bridge"
Creating volume "hadoop-cluster-im_hadoop_namenode" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode1" with default driver
Creating volume "hadoop-cluster-im_hadoop_datanode2" with default driver

  Initializing
    Time: 105 seconds
    [oooooooooooooooooooooo]

Creating resourcemanager ... done
Creating youtube-fetcher ... done
Creating datanode1 ... done
Creating datanode3 ... done
Creating nodemanager1 ... done
Creating nodemanager3 ... done
Creating nodemanager2 ... done
Creating submit ... done

  Active Wind
  Accédez aux para
  In 2nd Col 21

```