MINI PROJECT ON BIG DATA STACK

Hadoop HDFS

A distributed file system for storing large datasets

PySpark

A Python API for Apache Spark used for distributed data processing and analytics

Mongodb

NoSQL database for storing data, enabling fast reads and writes

WORKFLOW

Raw Data --> Hadoop HDFS --> PySpark --> MongoDB

1) Prepare a Hadoop HDFS filesystem
2) Store dataset in distributed file system
3) Data Processing with PySpark
4) Store processed data in MongoDB

INFRASTRUCTURE

OS: Ubuntu

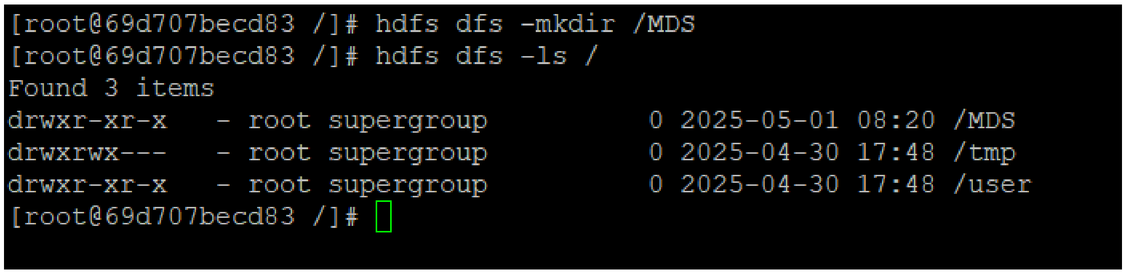
Platform: Docker

Docker Images:	westernscience/hadoop	(For Hadoop)
	pyspark	(For Data Processing)
	mongo:4.4.18	(For MongoDB)

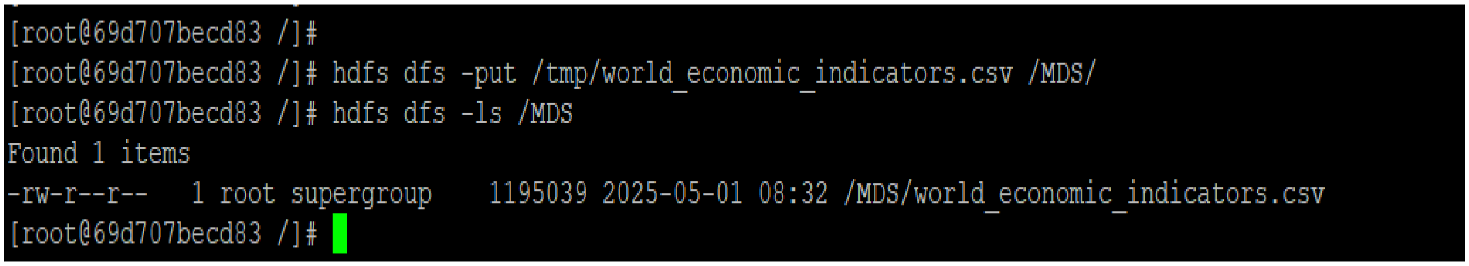
```
# docker run -d --name=hadoop -v /mini/data:/usr/local/hdfs/datastore -p 9870:9870 -p 9000:9000 -p 8088:8088 -p 9864:9864 westernscience/hadoop
```

```
# docker run -d --name=mongodb -v /mini/db:/data/db -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=password mongo:4.4.18
```

CREATE HDFS FILESYSTEM AND UPLOAD DATASET



```
[root@69d707becd83 /]# hdfs dfs -mkdir /MDS
[root@69d707becd83 /]# hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - root supergroup          0 2025-05-01 08:20 /MDS
drwxrwx---   - root supergroup          0 2025-04-30 17:48 /tmp
drwxr-xr-x   - root supergroup          0 2025-04-30 17:48 /user
[root@69d707becd83 /]#
```



```
[root@69d707becd83 /]#
[root@69d707becd83 /]# hdfs dfs -put /tmp/world_economic_indicators.csv /MDS/
[root@69d707becd83 /]# hdfs dfs -ls /MDS
Found 1 items
-rw-r--r--   1 root supergroup    1195039 2025-05-01 08:32 /MDS/world_economic_indicators.csv
[root@69d707becd83 /]#
```

ACCESS HDFS USING PYSPARK

```
from pyspark.sql import SparkSession
spark = SparkSession.builder .appName("Big Data Mini Project").getOrCreate()

# Read CSV from HDFS
df = spark.read.csv("hdfs://192.168.1.11:9000/MDS/world_economic_indicators.csv",
                    header=True,
                    inferSchema=True)
```

DATA PROCESSING

```
from pyspark.sql.functions import col
df_selected = df.select(
  "Country Name",
  "Year",
  "Unemployment_Rate"
)
# Pivot the DataFrame to have years as columns and the unemployment rate as values
df_unemployment_rate = df_selected.groupBy("Country Name").pivot("Year").agg(
  {"Unemployment_Rate":"first"}
)
# Rename columns for clarity
df_unemployment_rate = df_unemployment_rate.select(
  "Country Name",
  *[col(str(year)).alias(str(year)) for year in range(1960, 2021)])
```

STORE TRANFORMED DATA IN MONGODB

```
>use MDS
>db.createCollection(“unemployment”)
```

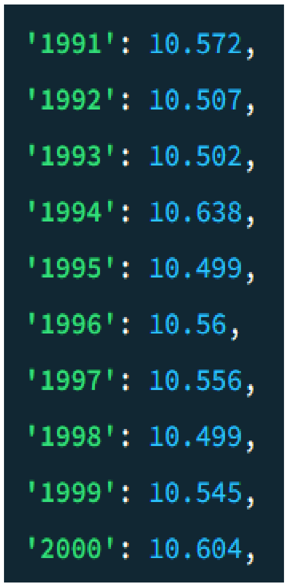
```
# Convert PySpark DataFrame to Pandas
pandas_df = df_unemployment_rate.toPandas()

# Convert each row into a dictionary
records = pandas_df.to_dict(orient='records')

client = MongoClient("mongodb://admin:password@192.168.1.11:27017/")
db = client["MDS"]
collection = db["unemployment"]

# Insert all records at once
if records:  # Check if there are records to insert
   result = collection.insert_many(records)
   if result.inserted_ids:  # Check if insert was successful
      print(f"{len(result.inserted_ids)} records successfully inserted into MongoDB!")
else:
   print("No records to insert.")
```

```
>db.unemployment.find({“Country Name”:”Nepal”)
```



```
'1991': 10.572,
'1992': 10.507,
'1993': 10.502,
'1994': 10.638,
'1995': 10.499,
'1996': 10.56,
'1997': 10.556,
'1998': 10.499,
'1999': 10.545,
'2000': 10.604,
```