**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

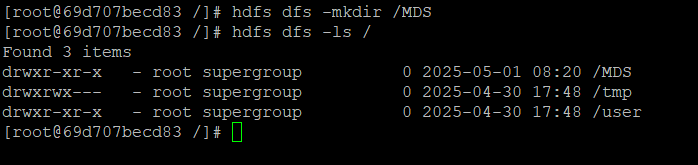
**Docker Images**: westernscience/hadoop (For Hadoop)

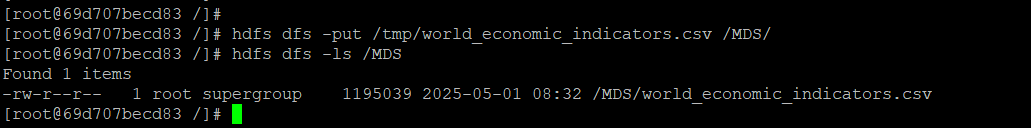
pyspark (For Data Processing)

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**

****

**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”)

