CLUSTER SETUP DOCUMENTATION

1) Technologies Used

Technology	Purpose
Java (JDK 8+)	Required for running both Hadoop and Apache Spark
Apache Hadoop	Distributed file system (HDFS) to store large-scale datasets
Apache Spark	Distributed computing framework for fast, in-memory data processing
Jupyter Notebook	Interactive platform for developing and visualizing Spark-based ML models
HDFS	Hadoop Distributed File System used to store and retrieve datasets

2) Environment & Configuration Details

Component	Configuration
OS	Ubuntu 20.04 / Linux-based OS (Recommended)
Java Version	JDK 8 (Compatible with both Hadoop and Spark)
Hadoop Version	3.x
Spark Version	3.5.5
Spark Mode	Single Node Cluster Mode (Master + Worker on same machine)
Notebook	Jupyter Notebook (linked with PySpark using environment variables)

3) Cluster Setup Steps

Step 1: Java Installation

```
sudo apt update
sudo apt install openjdk-8-jdk
java -version
```

- Ensure `JAVA_HOME` is set:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Step 2: Hadoop Installation & Configuration

1. Download Hadoop:

```
Wget
https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz
tar -xvzf hadoop-3.4.1.tar.gz
```

2. Configure environment variables in `~/.bashrc`:

```
export HADOOP_HOME=~/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

3. Configure files:

   - `core-site.xml` → NameNode URI
   - `hdfs-site.xml` → Replication factor, storage directories

4. Format HDFS and start daemons:

```
hdfs namenode -format
start-dfs.sh
```

Step 3: Apache Spark Installation

```
wget
https://downloads.apache.org/spark/spark-3.5.5/spark-3.5.5-bin
-hadoop3.tgz
tar -xvzf spark-3.5.5-bin-hadoop3.tgz
```

- Add to `~/.bashrc`:

```
export SPARK_HOME=~/spark
export PATH=$PATH:$SPARK_HOME/bin
```

Step 4: Configure Spark for Single Node Cluster

- Edit `conf/spark-env.sh`:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export SPARK_MASTER_HOST='localhost'
export SPARK_LOCAL_IP='127.0.0.1'
```

- Start Spark services:

```
start-master.sh
start-worker.sh spark://localhost:7077
```

Step 5: Jupyter Notebook Integration with PySpark

- Install dependencies:

```
pip install findspark
pip install notebook
```

- Set environment variables inside the notebook:

```
import os
os.environ["SPARK_HOME"] = "/path/to/spark"
os.environ["PYSPARK_PYTHON"] = "/usr/bin/python3"
```

- Initialize Spark:

```
import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder \
      .master("spark://localhost:7077") \
      .appName("Lung_Cancer_prediction") \
      .getOrCreate()
```

HDFS Dataset Loading

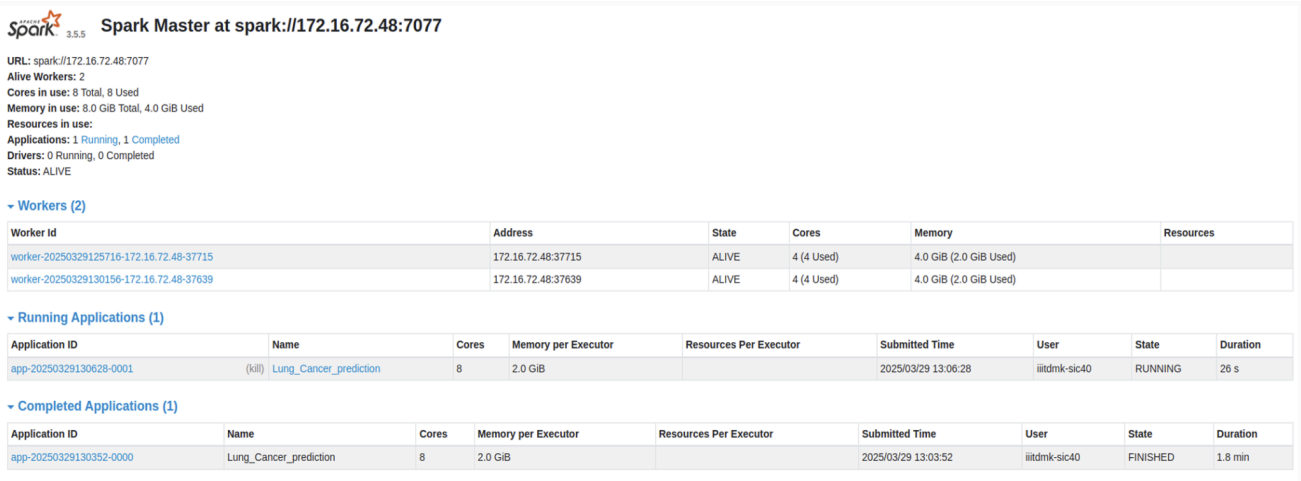
Place dataset in HDFS:

```
hdfs dfs -mkdir /inputdata
hdfs dfs -put survey_lung_cancer.csv /inputdata
```

Load it in Spark:

```
df =
spark.read.csv("hdfs://localhost:9000/user/iiitdmk-sic40/surve
y_lung_cancer.csv", header=True, inferSchema=True)
```

Master and Worker UI Screenshots:



Spark Master at spark://172.16.72.48:7077

URL: spark://172.16.72.48:7077
Alive Workers: 2
Cores in use: 8 Total, 8 Used
Memory in use: 8.0 GiB Total, 4.0 GiB Used
Resources in use:
Applications: 1 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

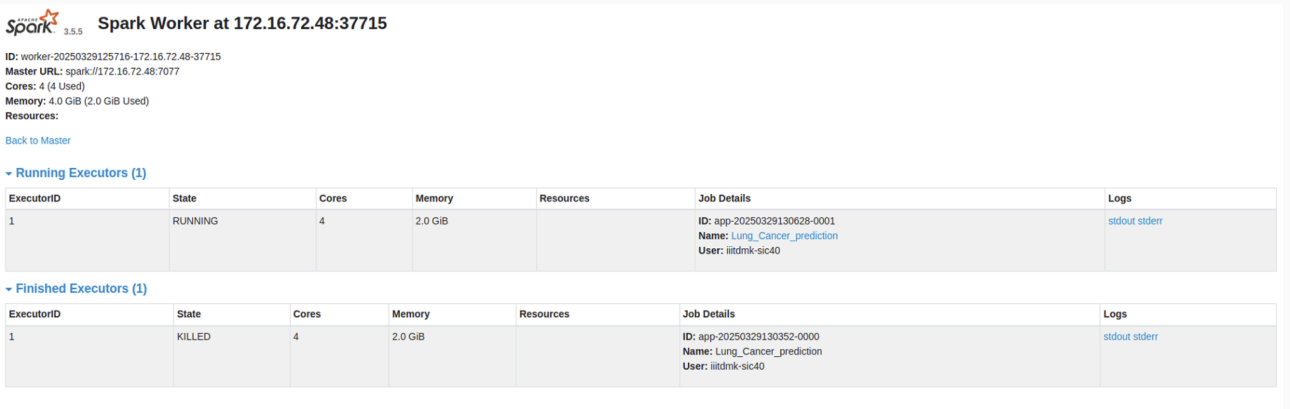
Worker Id	Address	State	Cores	Memory	Resources
worker-20250329125716-172.16.72.48-37715	172.16.72.48:37715	ALIVE	4 (4 Used)	4.0 GiB (2.0 GiB Used)	
worker-20250329130156-172.16.72.48-37639	172.16.72.48:37639	ALIVE	4 (4 Used)	4.0 GiB (2.0 GiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250329130628-0001 (kill)	Lung_Cancer_prediction	8	2.0 GiB		2025/03/29 13:06:28	iiitdmk-sic40	RUNNING	26 s

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250329130352-0000	Lung_Cancer_prediction	8	2.0 GiB		2025/03/29 13:03:52	iiitdmk-sic40	FINISHED	1.8 min



Spark 3.5.5 **Spark Worker at 172.16.72.48:37715**

ID: worker-20250329125716-172.16.72.48-37715
Master URL: spark://172.16.72.48:7077
Cores: 4 (4 Used)
Memory: 4.0 GiB (2.0 GiB Used)
Resources:

Back to Master

Running Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
1	RUNNING	4	2.0 GiB		**ID:** app-20250329130628-0001 **Name:** Lung_Cancer_prediction **User:** iiitdmk-sic40	stdout stderr

Finished Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
1	KILLED	4	2.0 GiB		**ID:** app-20250329130352-0000 **Name:** Lung_Cancer_prediction **User:** iiitdmk-sic40	stdout stderr

4) Issues Faced and Solutions

Issue 1: Worker not connecting to Master every time Spark starts

Problem:
Despite launching the worker, sometimes it does not register with the Spark master.

Possible Causes:

- Incorrect or missing IP/hostname configuration in `spark-env.sh`
- Firewall or port conflict (7077)
- Worker is not properly stopped in the previous session

Solution:

1. Ensure proper values in `spark-env.sh`:

```
export SPARK_MASTER_HOST='localhost'
export SPARK_LOCAL_IP='127.0.0.1'
```

2. Stop all previous Spark instances:

```
stop-all.sh
jps
```

3. Restart Spark in order:

```
start-master.sh
start-worker.sh spark://localhost:7077
```

Issue 2: Dataset not loading from HDFS consistently

Problem:
Sometimes the dataset doesn't load properly or gives a path not found error.

Possible Causes:

- Wrong HDFS path
- Hadoop services not running
- Permission issues

Solution:

1. Start Hadoop before using HDFS:
```
start-dfs.sh
```

2. Verify HDFS is accessible:
```
hdfs dfs -ls /user/yourname/
```

3. Ensure file is uploaded:
```
hdfs dfs -put dataset.csv /user/yourname/
```

Issue 3: Model accuracy is inconsistent on each run

Problem:
Accuracy varies each time the model is trained, even with the same data.

Possible Causes:

- Random splits in training/test data
- Randomness in algorithms (like MLP)
- Imbalance affecting SMOTE behavior

Solution:

1. Fix the random seed:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42)
```

2. Set random state in all models:

```
model = RandomForestClassifier(random_state=42)
```

3. Apply consistent SMOTE:

```
smote = SMOTE(random_state=42)
```

Conclusion

This documentation outlines a full setup of a single-node big data processing cluster using Hadoop and Spark, integrated with Jupyter Notebook for ML tasks. By addressing practical issues like worker connectivity, HDFS data loading, and ML consistency, the system is now stable and reproducible for lung cancer prediction tasks.