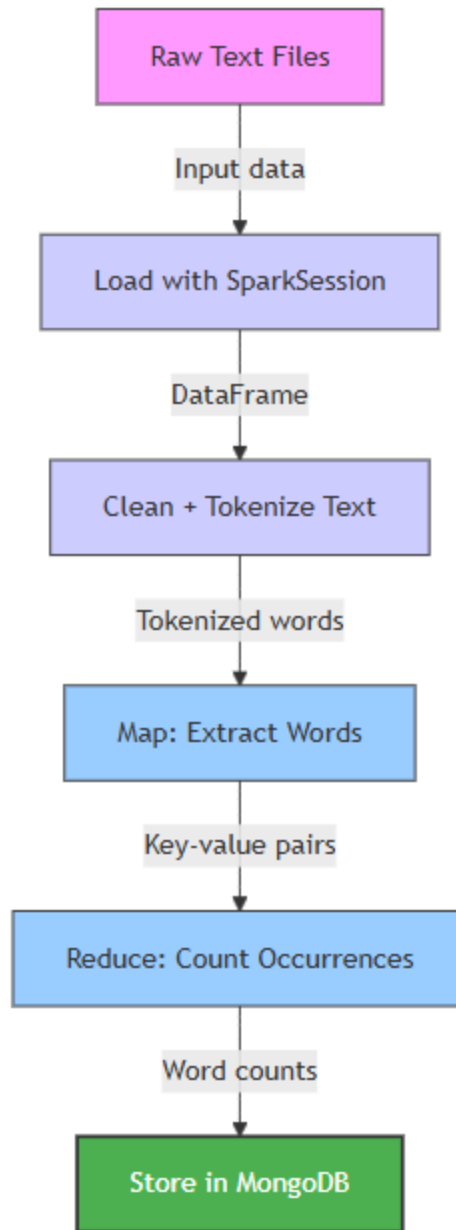


<https://github.com/kheder-hassoun/batch-processing-pipeline>

To make this easier, let me share with you a guide that outlines the steps, common problems, and their solutions.

First let's do with Simple architecture (without distributing)



First, we will forget the database let's just explore the spark and hdfs.

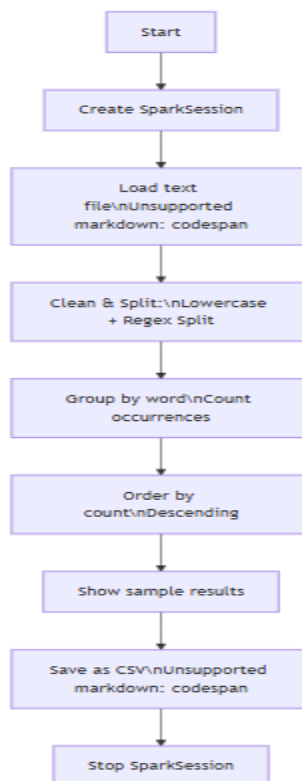
1. Add dependencies

```

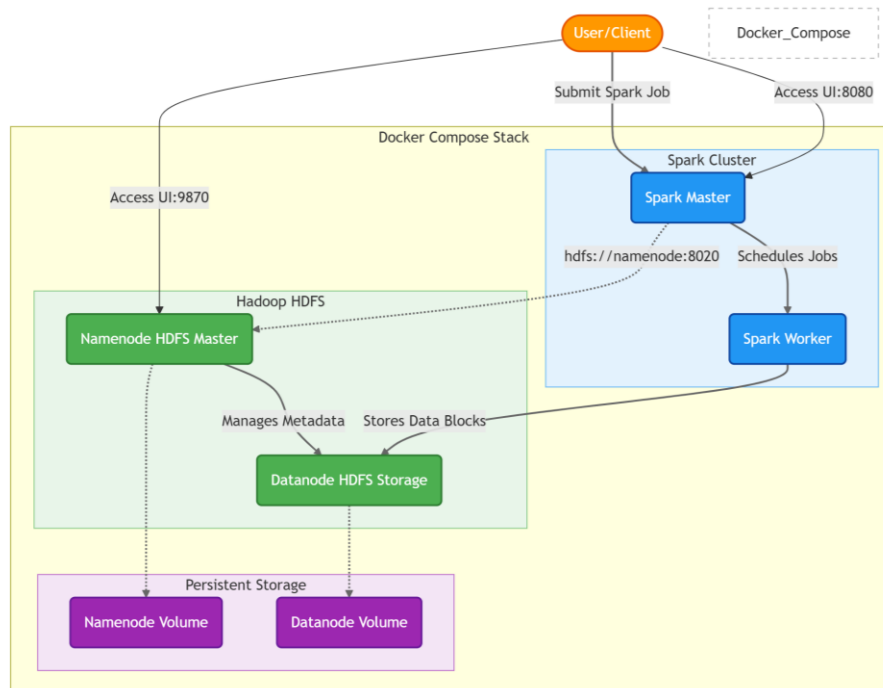
<dependencies>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.12</artifactId>
    <version>3.4.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.12</artifactId>
    <version>3.4.1</version>
    <scope>provided</scope>
  </dependency>
</dependencies>

```

2. Spark Batch Job (java)



Now the Docker-Compose ❤️



```
version: '3.8'
services:
  namenode:
    image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
    container_name: namenode
    environment:
      - CLUSTER_NAME=test
      - CORE_CONF_fs_defaultFS=hdfs://namenode:8020
    ports:
      - "9870:9870"    # HDFS Web UI
      - "8020:8020"    # HDFS RPC
    volumes:
      - namenode_data:/hadoop/dfs/name

  datanode:
    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
    container_name: datanode
    environment:
      - CORE_CONF_fs_defaultFS=hdfs://namenode:8020
    depends_on:
      - namenode
    ports:
      - "9864:9864"    # Datanode Web UI
    volumes:
      - datanode_data:/hadoop/dfs/data

# from here i got the spark master and worker 🍷❤️
#https://github.com/big-data-europe/docker-spark
  spark-master:
    image: bde2020/spark-master:3.3.0-hadoop3.3
    container_name: spark-master
    environment:
      - ENABLE_INIT_DAEMON=false
```

```

    - SPARK_MODE=master
  ports:
    - "8080:8080"    # Spark Master UI
  volumes:
    - ./out/artifacts/spark_hdfs_jar/spark-hdfs.jar:/opt/spark-
apps/spark-pipeline-1.0.jar # this to map the local jar to the image
so we don't need to rebuild the image each time build the code
  depends_on:
    - namenode

spark-worker:
  image: bde2020/spark-worker:3.3.0-hadoop3.3
  container_name: spark-worker
  environment:
    - ENABLE_INIT_DAEMON=false
    - SPARK_MODE=worker
    - SPARK_MASTER_URL=spark://spark-master:7077
    - CORE_CONF_fs_defaultFS=hdfs://namenode:8020
  depends_on:
    - spark-master
    - datanode

volumes:
  namenode_data:
  datanode_data:

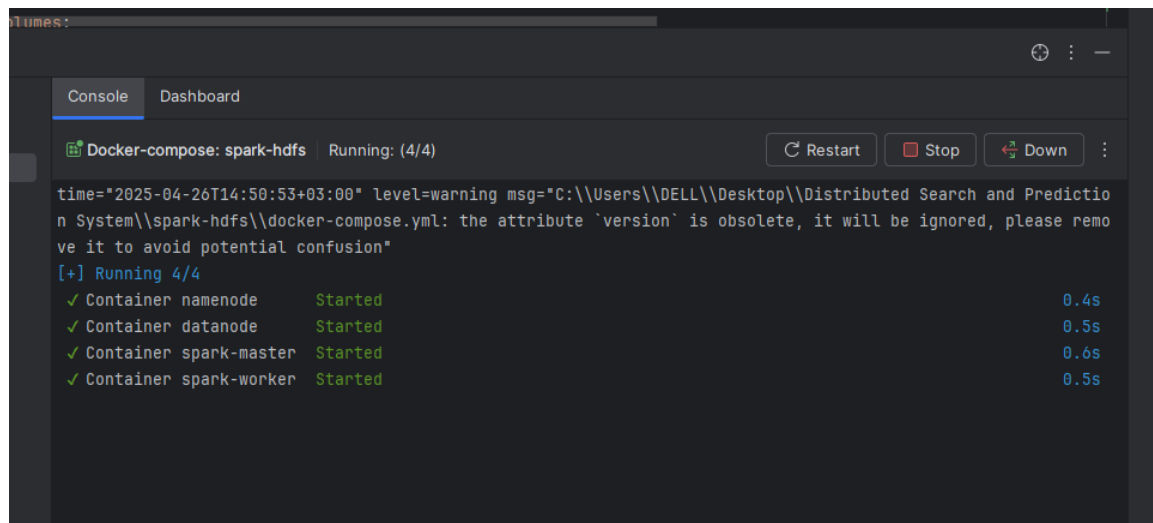
```

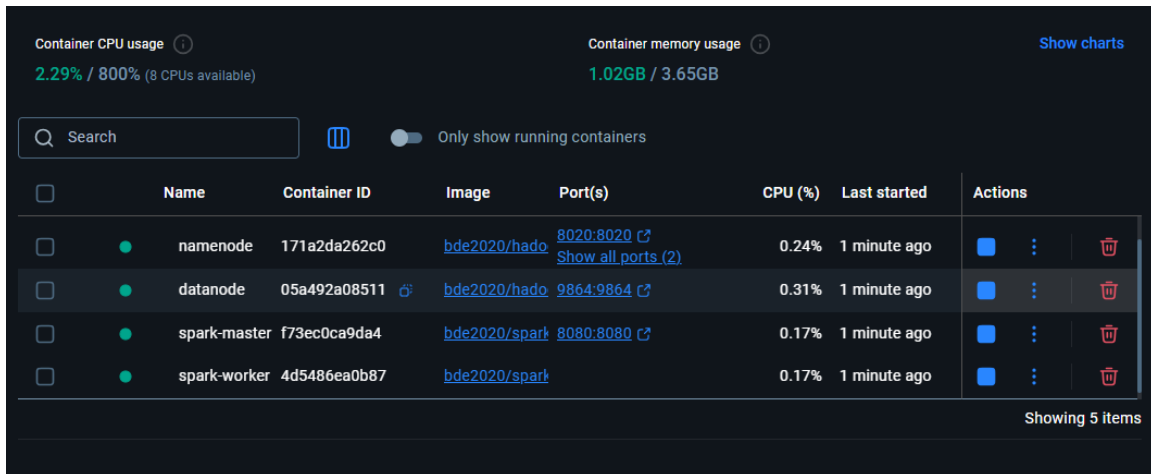
Testing

1 run all images and look at the containers running:

```
docker-compose up -d
```

Actually, I use the IntelliJ Docker plugin





Last way 🤔

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
4d5486ea0b87   bde2020/spark-worker:3.3.0-hadoop3.3  "/bin/bash /worker.sh"  14 minutes ago Up 2 minutes   8081/tcp
f73ec0ca9da4   bde2020/spark-master:3.3.0-hadoop3.3  "/bin/bash /master.sh"  35 minutes ago Up 2 minutes   6066/tcp, 7077/tcp, 0.0.0.0:8080->8080/tcp
05a492a08511   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8  "/entrypoint.sh /run_"  43 minutes ago Up 2 minutes (healthy)  0.0.0.0:9864->9864/tcp
171a2da262c0   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8  "/entrypoint.sh /run_"  45 minutes ago Up 2 minutes (healthy)  0.0.0.0:8020->8020/tcp, 0.0.0.0:9870->9870/tcp
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>
```

Copy sample.txt Into HDFS

```
docker cp input-data/sample.txt namenode:/sample.txt
```

```
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker cp input-data/sample.txt namenode:/sample.txt
Successfully copied 2.05kB to namenode:/sample.txt
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>
```

Then we put it into HDFS:

```
docker exec -it namenode hdfs dfs -mkdir -p /input
```

```
docker exec -it namenode hdfs dfs -put /sample.txt /input/
```

then verify

```
docker exec -it namenode hdfs dfs -ls /input
docker exec -it namenode hdfs dfs -cat /input/sample.txt
```

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker exec -it namenode hdfs dfs -ls /input
Found 1 items
-rw-r--r-- 3 root supergroup          41 2025-04-26 11:59 /input/sample.txt

C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker exec -it namenode hdfs dfs -cat /input/sample.txt
2025-04-26 12:03:16,431 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
test
test
kheder
kheder
goodmorning

C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>
```

Now, Let's run the Spark job from inside the spark-master container:

```
docker exec spark-master spark-submit --master spark://spark-master:7077 --class
me.spark.WordCount /opt/spark-apps/spark-pipeline-1.0.jar
hdfs://namenode:8020/input/sample.txt
hdfs://namenode:8020/output
```

But it does not work because **spark-submit** is not in the right path so **let's find it**

```
C:\Windows\System32\cmd.exe - docker exec -it spark-master /bin/bash
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker exec -it spark-master /bin/bash
bash-5.0#
bash-5.0# find / -name spark-submit
/spark/bin/spark-submit
bash-5.0#
```

Now let's run (inside the container)

```
/spark/bin/spark-submit --master spark://spark-master:7077 --class
me.spark.WordCount /opt/spark-apps/spark-hdfs.jar
hdfs://namenode:8020/input/sample.txt hdfs://namenode:8020/output
```

Still not working

Say that there is no class (cannot find it)

```
bash-5.0#
bash-5.0# /spark/bin/spark-submit --master spark://spark-master:7077 --class me.spark.WordCount /opt/spark-apps/spark-pipeline-1.0.jar dfs:/
/namenode:8020/input/sample.txt hdfs://namenode:8020/output
Error: Failed to load class me.spark.WordCount.
25/04/26 12:15:13 INFO ShutdownHookManager: Shutdown hook called
25/04/26 12:15:13 INFO ShutdownHookManager: Deleting directory /tmp/spark-a99f7552-ba45-463b-ae35-fc3f89593bc5
bash-5.0#
```

But The class and package are correct

```
WordCount.java pom.xml (spark-hdfs) docker-comp
Manifest-Version: 1.0
Main-Class: me.spark.WordCount
```

????!!!

So I guess there is a build problem

```
</dependencies>
<build>
  <plugins>
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <configuration>
        <archive>
          <manifest>
            <mainClass>me.spark.WordCount</mainClass>
          </manifest>
        </archive>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Then its worked but not exactly because there another problem in java version

(Compile with 17 but the spark support 8)

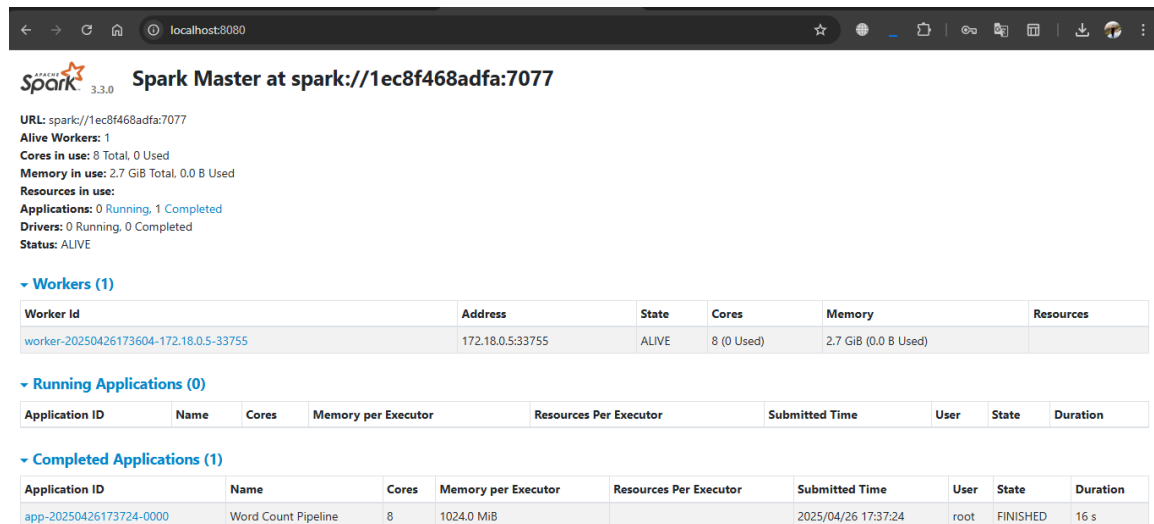
So

```
<properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```


Its works but there is a problem related to hard code (path) in the wordcount so I fix it with passed argument

We can Check Job Status

<http://localhost:8080>



Spark Master at spark://1ec8f468adfa:7077

URL: spark://1ec8f468adfa:7077

Alive Workers: 1

Cores in use: 8 Total, 0 Used

Memory in use: 2.7 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 1 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

▼ Workers (1)

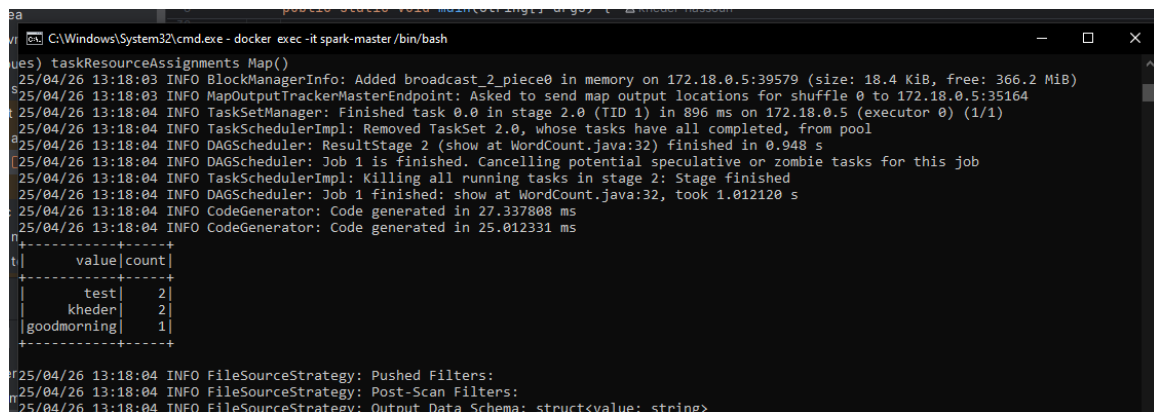
Worker Id	Address	State	Cores	Memory	Resources
worker-20250426173604-172.18.0.5-33755	172.18.0.5:33755	ALIVE	8 (0 Used)	2.7 GiB (0.0 B Used)	

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

▼ Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250426173724-0000	Word Count Pipeline	8	1024.0 MiB		2025/04/26 17:37:24	root	FINISHED	16 s



```
ba
C:\Windows\System32\cmd.exe - docker exec -it spark-master/bin/bash
ues) taskResourceAssignments Map()
25/04/26 13:18:03 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 172.18.0.5:39579 (size: 18.4 KiB, free: 366.2 MiB)
25/04/26 13:18:03 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 172.18.0.5:35164
25/04/26 13:18:04 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 1) in 896 ms on 172.18.0.5 (executor 0) (1/1)
25/04/26 13:18:04 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
25/04/26 13:18:04 INFO DAGScheduler: ResultStage 2 (show at WordCount.java:32) finished in 0.948 s
25/04/26 13:18:04 INFO DAGScheduler: Job 1 is finished. Cancelling potential speculative or zombie tasks for this job
25/04/26 13:18:04 INFO TaskSchedulerImpl: Killing all running tasks in stage 2: Stage finished
25/04/26 13:18:04 INFO DAGScheduler: Job 1 finished: show at WordCount.java:32, took 1.012120 s
25/04/26 13:18:04 INFO CodeGenerator: Code generated in 27.337808 ms
25/04/26 13:18:04 INFO CodeGenerator: Code generated in 25.012331 ms
+-----+
t|      value|count|
+-----+
|      test|      2|
|     kheder|      2|
|goodmorning|      1|
+-----+
25/04/26 13:18:04 INFO FileSourceStrategy: Pushed Filters:
25/04/26 13:18:04 INFO FileSourceStrategy: Post-Scan Filters:
25/04/26 13:18:04 INFO FileSourceStrategy: Output Data Schema: struct<value: string>
```

How to see the results

First view the output file name then show it

```
C:\Windows\System32\cmd.exe
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker exec -it namenode hdfs dfs -ls /output
Found 2 items
-rw-r--r-- 3 root supergroup 0 2025-04-26 17:37 /output/_SUCCESS
-rw-r--r-- 3 root supergroup 30 2025-04-26 17:37 /output/part-00000-1a86a43c-0d8f-40d3-a542-0c13ebe4d6e9-c000.csv
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>docker exec -it namenode hdfs dfs -cat /output/part-00000-1a86a43c-0
d8f-40d3-a542-0c13ebe4d6e9-c000.csv
2025-04-26 17:43:41,344 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
kheder,2
test,2
goodmorning,1
C:\Users\DELL\Desktop\Distributed Search and Prediction System\spark-hdfs>
```

Store the results in MongoDB (inside Docker)

```
53  ▶  mongodb:
54      image: mongo:latest
55      container_name: mongodb
56      ports:
57      - "27017:27017"
58      volumes:
59      - mongo_data:/data/db
60
61
```

```
1
2  volumes:
3      namenode_data:
4      datanode_data:
5      mongo_data: ←
6
```

Also the dependencies 😊

```
</dependency>
<dependency>
  <groupId>org.mongodb.spark</groupId>
  <artifactId>mongo-spark-connector_2.12</artifactId>
  <version>10.1.1</version>
</dependency>
```

Problem one there is no connector in the docker (the connector exists in the dependencies but it not build with the jar)

Solution

Insert the connection in the runtime

```
/spark/bin/spark-submit --master spark://spark-master:7077 --
class me.spark.WordCount --packages
org.mongodb.spark:mongo-spark-connector_2.12:10.1.1
/opt/spark-apps/spark-hdfs.jar
```

Another problem is connection refused

The solution is

```

28 spark-master:
31 environment:
32   - ENABLE_INIT_DAEMON=false
33   - SPARK_MODE=master
34 ports:
35   - "8080:8080" # Spark Master UI
36 volumes:
37   - ./out/artifacts/spark_hdfs_jar/spark-hdfs.jar:/opt/spark-apps/spark-hdfs.jar #this
38 depends_on: ←
39   - namenode
40   - mongodb
41
42 spark-worker:
43   image: bde2020/spark-worker:3.3.0-hadoop3.3
44   container_name: spark-worker
45   environment:
46     - ENABLE_INIT_DAEMON=false
47     - SPARK_MODE=worker
48     - SPARK_MASTER_URL=spark://spark-master:7077
49     - CORE_CONF_fs_defaultFS=hdfs://namenode:8020
50   depends_on: ←
51     - spark-master
52     - datanode
53     - mongodb

```

Now lets implement our case (autocomplete)

Read input text file

Parse the words.

For each prefix of 2 and 3 characters, map → list of top 5 frequent full words.

Store prefix → list of 5 words in MongoDB.

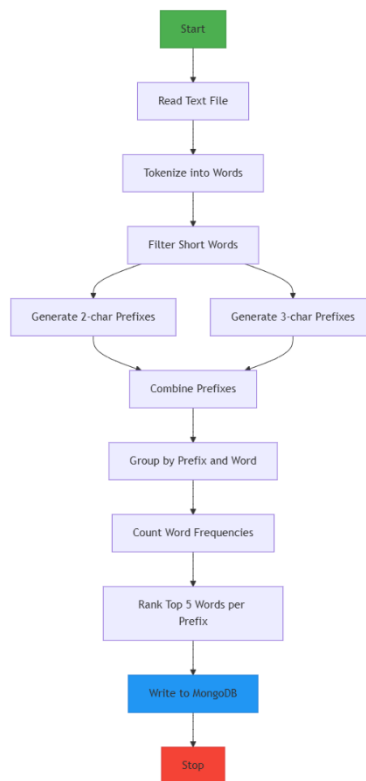
Parallel by default

Parameter	Description	Default
spark.default.parallelism	How many tasks for RDD operations	2 × number of CPU cores
How to config it		

```
SparkSession.builder()  
  .appName("PrefixAutocompletePipeline"  
  )  
  .config("spark.default.parallelism", "8") // example  
  .getOrCreate();
```

Code steps

See the code in Github



Results

```
testcollection
```