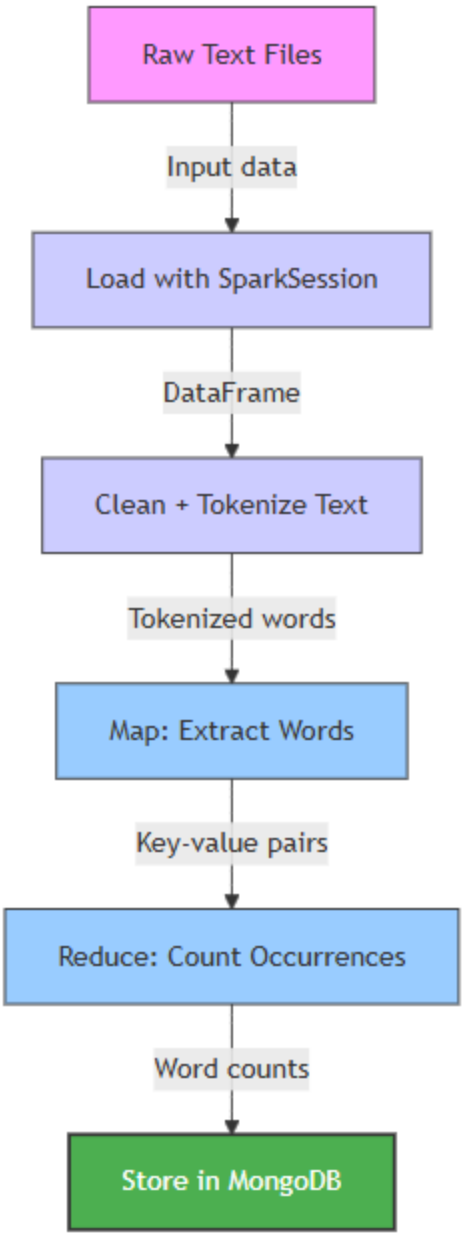
https://github.com/kheder-hassoun/batch-processing-pipeline

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)

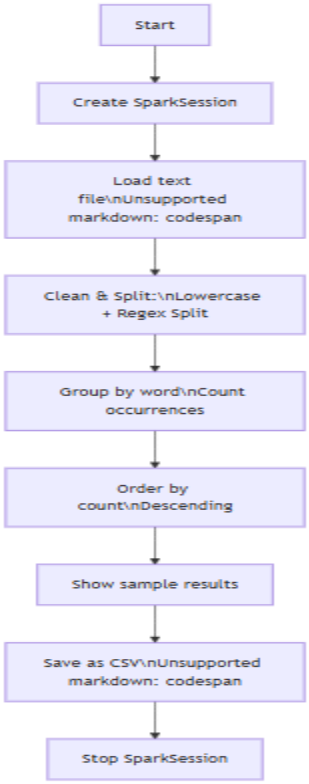


First, we will forget the database let's just explore the spark and hdfs.

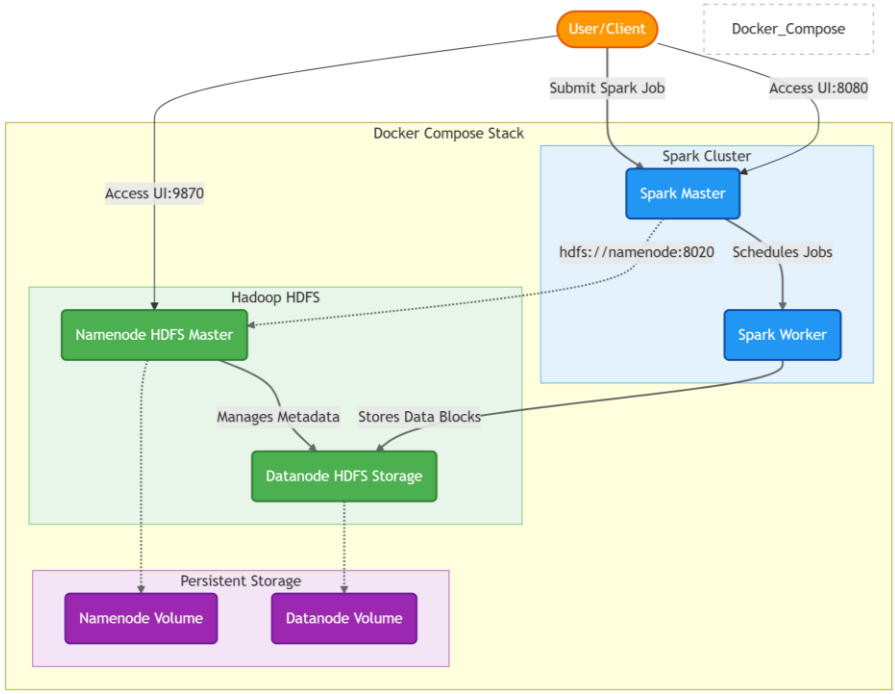
1. Add dependencies

```
<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 ❤️




```yaml
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
```

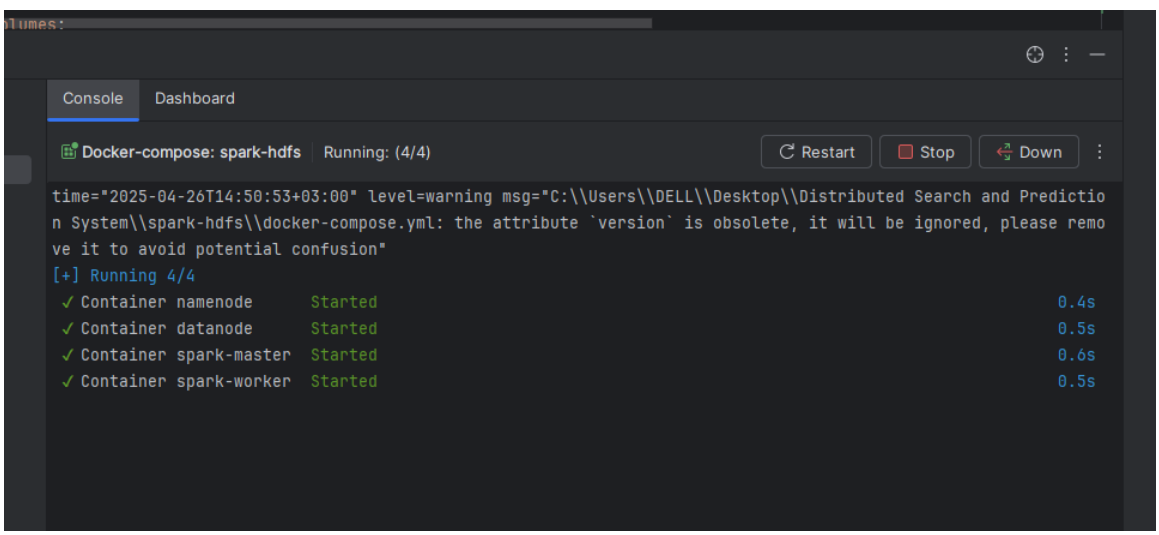
```
      - SPARK_MODE=master
    ports:
      - "8080:8080"    # Spark Master UI
    volumes:
      - ./out/artifacts/spark_hdfs_jar/spark-hdfs.jar:/opt/spark-
apps/spark-pipeline-1.0.jar # this to map the local jar to the image
so we don't need to rebuild the image each time build the code
    depends_on: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

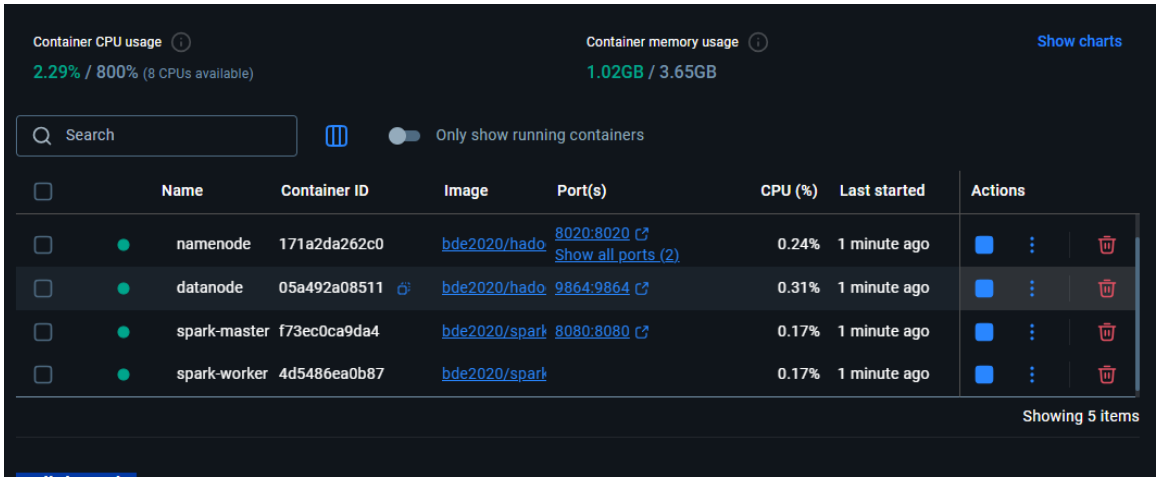
Testing

1 run all images and look at the containers running:

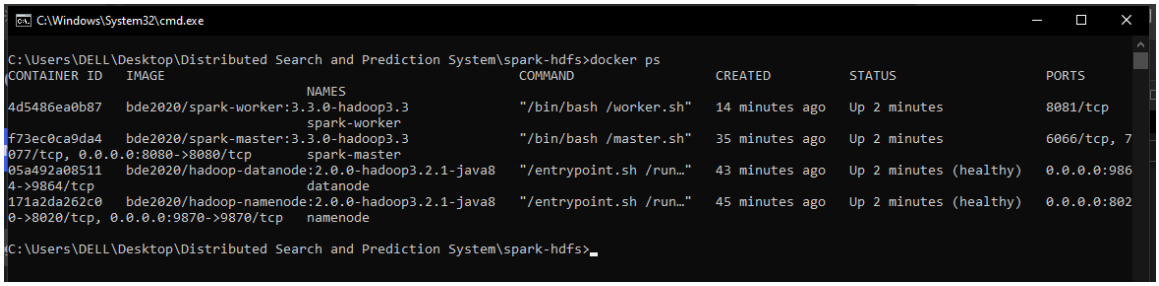
```
docker-compose up -d
```

Actually, I use the IntelliJ Docker plugin



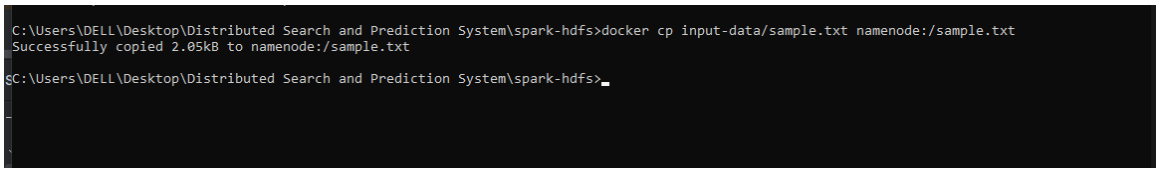


Last way 😁



Copy sample.txt Into HDFS

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



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        m pom.xml (spark-hdfs)        docker-comp

Manifest-Version: 1.0
Main-Class: me.spark.WordCount
```

????!!!

So I guess there is a build problem

```
    <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

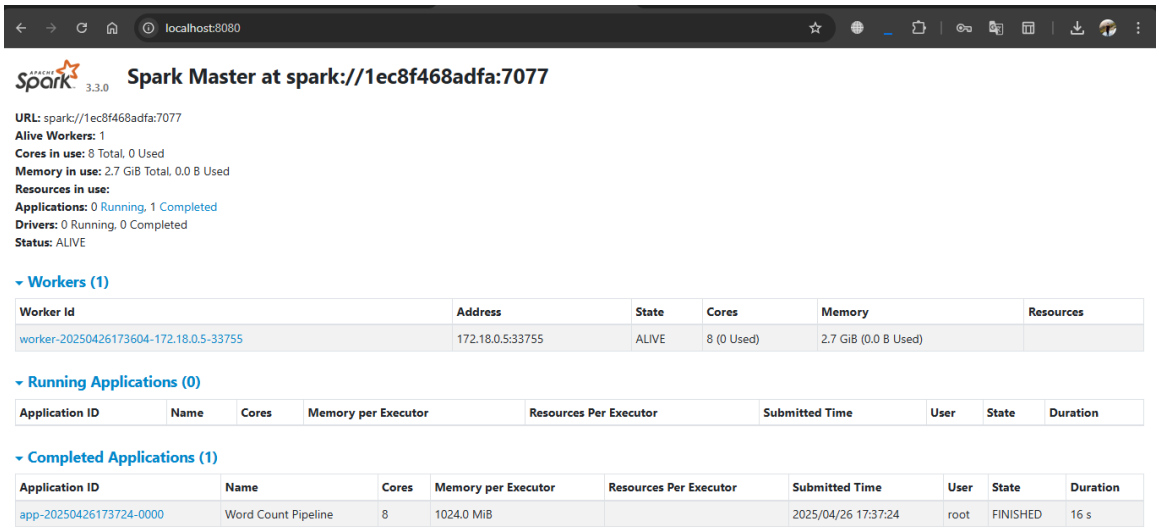
(Compile with 17 but the spark support 8)

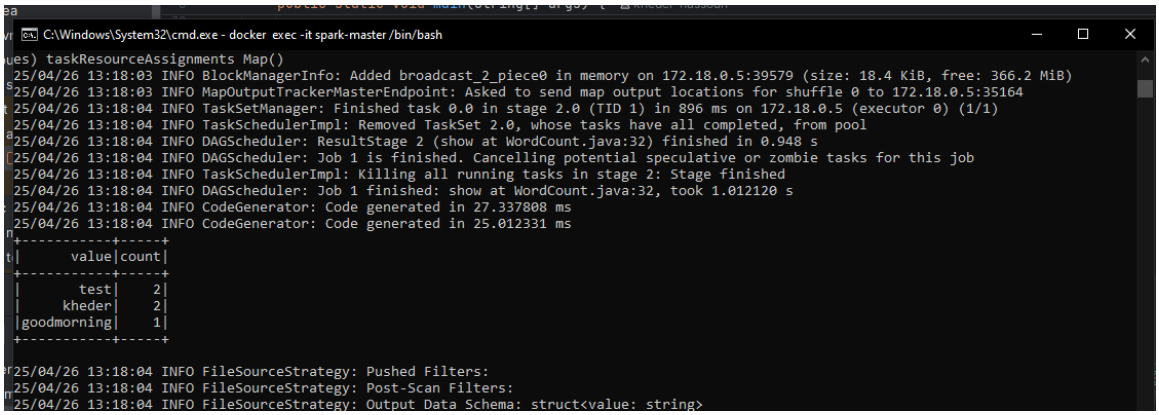
So

```
    <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





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)

```yaml
  mongodb:
    image: mongo:latest
    container_name: mongodb
    ports:
      - "27017:27017"
    volumes:
      - mongo_data:/data/db
```

```yaml
volumes:
  namenode_data:
  datanode_data:
  mongo_data:
```

Also the dependencies 😁

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

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

Solutinon

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

```yaml
  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-hdfs.jar #this
    depends_on:
      - namenode
      - mongodb

  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
      - 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 defult

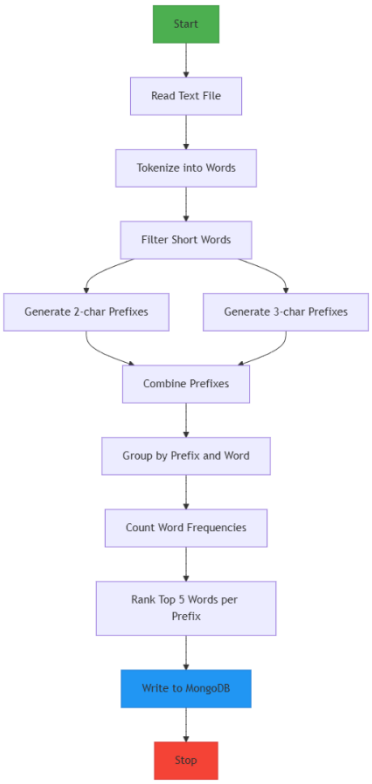
Parameter	Description	Default
spark.default.parallelism	How many tasks for RDD operations	2 × number of CPU cores

How to config it

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

Code steps

See the code in Github



Results

Terminal spark master namenode Local Local (2)

```
testcollection
mymongo> db.prefixcollection.find()
[
  {
    _id: ObjectId('680ff2cfc6a3d774a457928a'),
    prefix: 'go',
    word: 'goodmorning',
    count: Long('1')
  },
  {
    _id: ObjectId('680ff2cfc6a3d774a457928b'),
    prefix: 'goo',
    word: 'goodmorning',
    count: Long('1')
  },
  {
    _id: ObjectId('680ff2cfc6a3d774a457928c'),
    prefix: 'kh',
    word: 'kheder',
    count: Long('2')
  },
  {
```