Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Саратовский государственный технический университет имени Ю. А. Гагарина»

Кафедра прикладных информационных технологий

**Практическая работа по курсу**

**Анализ больших данных на тему:**

**«Изучение основ и принципов работы Hadoop YARN»**

Выполнил студент б1-ИФСТ-41,

Мокляк Никита Вячеславович

Проверил преподаватель

Кузьмин Алексей Константинович

Саратов, 2022

Цель работы

* Научиться создавать MapReduce job
* Изучить и применить основные функции работы с YARN

Краткое описание технологий

**Apache Hadoop** is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

* Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
* Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
* MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
* Hadoop Common – Provides common Java libraries that can be used across all modules.

**Docker** is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

## Задание

Создать MapReduce job

Выполнить job при помощи YARN фреймворка для MapReduce

Шаг 1

Создал мавен проект. У меня получился следующий pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.4</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.moklyak.mapred</groupId>  
 <artifactId>mapredjob</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>mapredjob</name>  
 <description>Demo project for Spring Boot</description>  
 <repositories>  
 <repository>  
 <id>cloudera</id>  
 <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>  
 </repository>  
 </repositories>  
 <properties>  
 <java.version>8</java.version>  
 <hadoop-version>3.3.2</hadoop-version>  
 </properties>  
 <dependencies>  
 <!--dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency-->  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-annotations</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-auth</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-client</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-common</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-datajoin</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-hdfs</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-mapreduce-client-app</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-common</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-mapreduce-client-common</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-mapreduce-client-hs</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-mapreduce-client-jobclient</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-mapreduce-client-shuffle</artifactId>  
 <version>${hadoop-version}</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <version>3.3.0</version>  
 <type>maven-plugin</type>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-jar-plugin</artifactId>  
 <configuration>  
 <archive>  
 <manifest>  
 <addClasspath>true</addClasspath>  
 <mainClass>com.moklyak.mapred.mapredjob.MapredjobApplication</mainClass>  
  
 </manifest>  
 </archive>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Шаг 2

Создаём классы реализующие интерфейсы MapReduce

Map:

public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {  
 public void map(LongWritable key, Text value,Context context) throws IOException,InterruptedException{  
 String line = value.toString();  
 StringTokenizer tokenizer = new StringTokenizer(line);  
 while (tokenizer.hasMoreTokens()) {  
 value.set(tokenizer.nextToken());  
 context.write(value, new IntWritable(1));  
 }  
 }  
}

Reduce:

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable>{  
 public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException,InterruptedException {  
 int sum = StreamSupport.*stream*(values.spliterator(), false)  
 .mapToInt(x->x.get())  
 .sum();  
 context.write(key, new IntWritable(sum));  
 }  
}

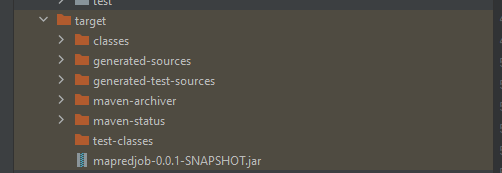
Шаг 3

Создаём главный метод, в котором конфигурируем job и запускаем её

public static void m(String[] args) throws Exception {  
 Configuration conf= new Configuration();  
 Job job = new Job(conf,"My Word Count Program");  
 job.setJarByClass(WordCount.class);  
 job.setMapperClass(Map.class);  
 job.setReducerClass(Reduce.class);  
 job.setOutputKeyClass(Text.class);  
 job.setOutputValueClass(IntWritable.class);  
 job.setInputFormatClass(TextInputFormat.class);  
 job.setOutputFormatClass(TextOutputFormat.class);  
 Path outputPath = new Path(args[1]);  
 //Configuring the input/output path from the filesystem into the job  
 FileInputFormat.*addInputPath*(job, new Path(args[0]));  
 FileOutputFormat.*setOutputPath*(job, new Path(args[1]));  
 //deleting the output path automatically from hdfs so that we don't have to delete it explicitly  
 outputPath.getFileSystem(conf).delete(outputPath);  
 //exiting the job only if the flag value becomes false  
 System.*exit*(job.waitForCompletion(true) ? 0 : 1);  
}

Шаг 4

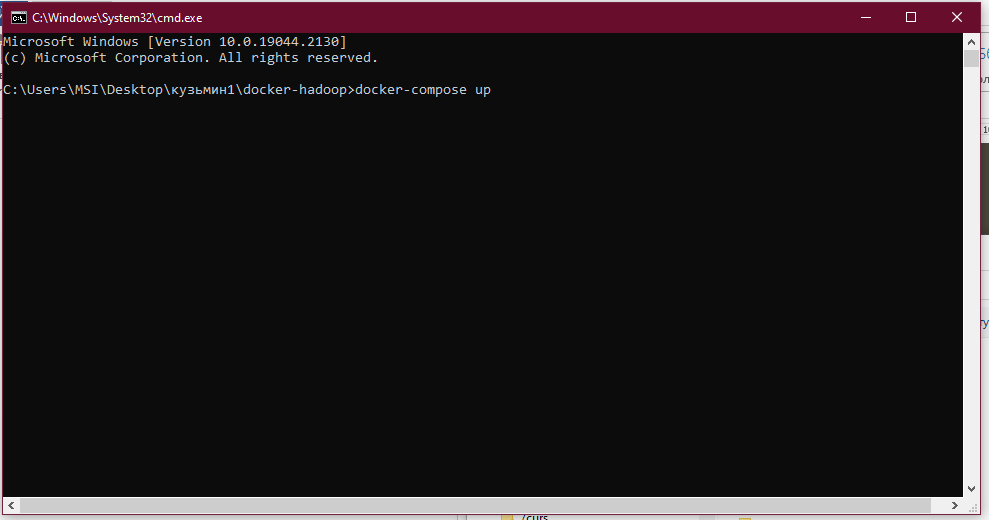
Билдим мавеном, получам jar



Шаг 5

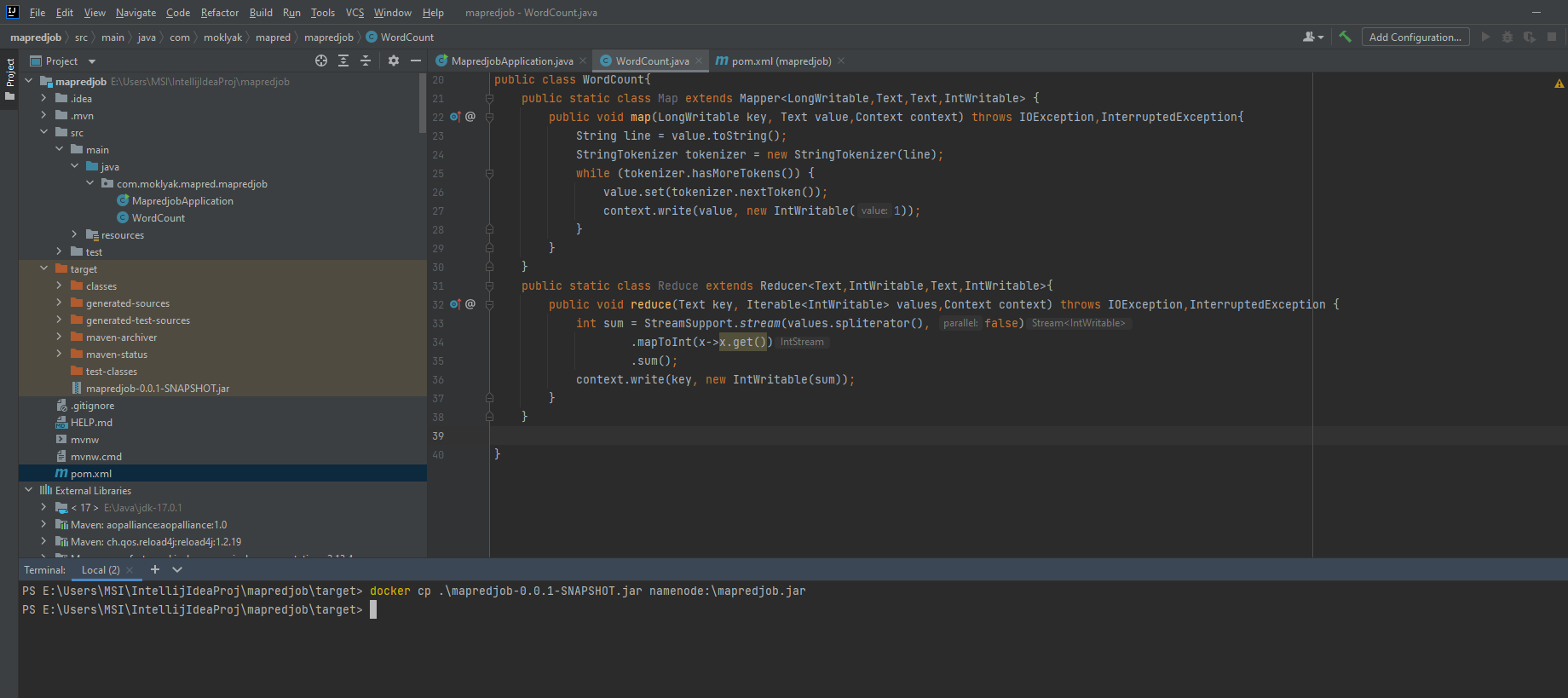
Теперь, когда джоба готова нужно её протестировать

Запускаем кластер



Шаг 6

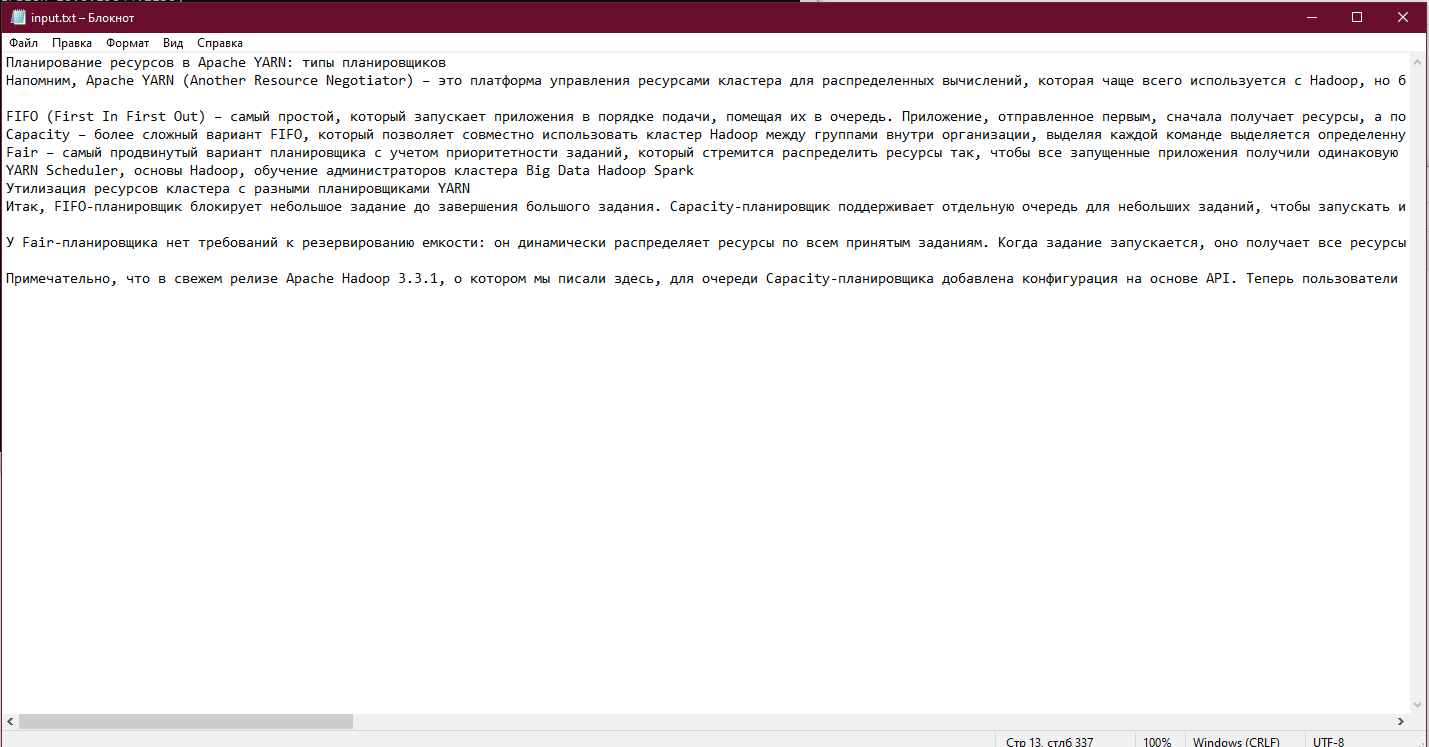
Отправляем jar на неймноду



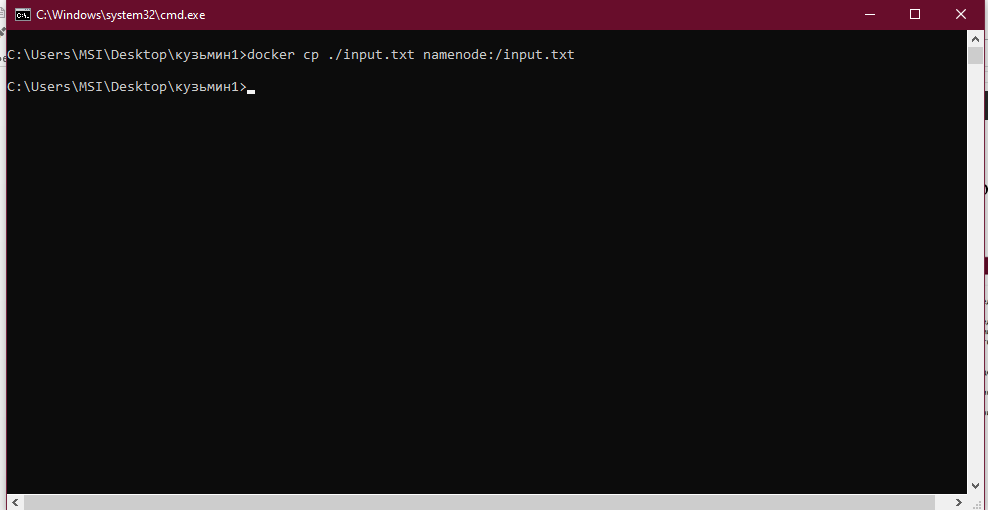
Шаг 7

Теперь нам нужно чего-нибудь скормить джобе, добавим чего-нибудь в hdfs

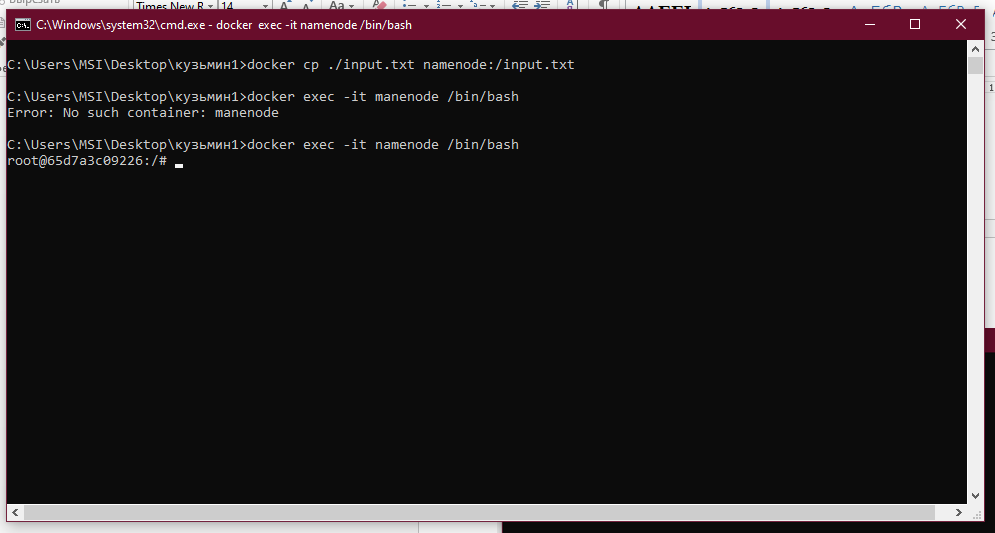
Файлик с чем-нибудь:



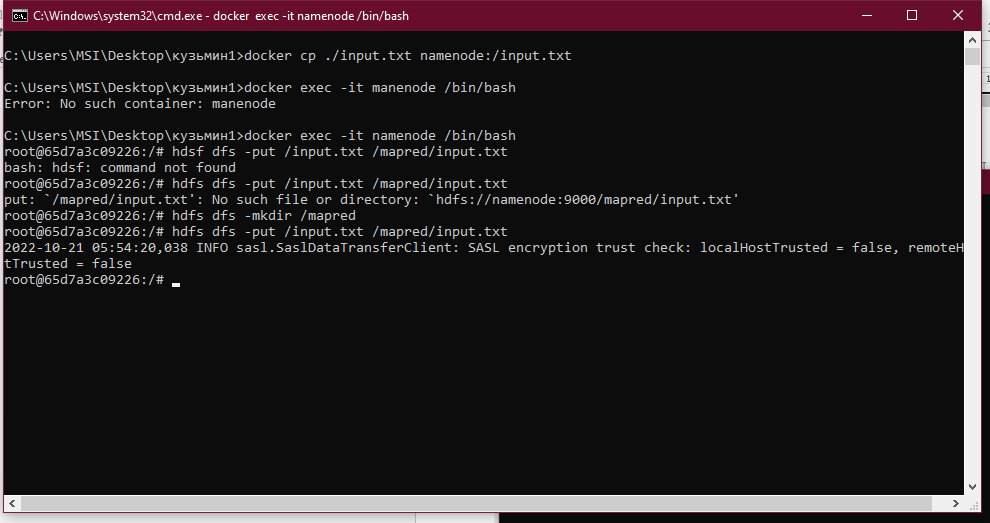
Копируем его на неймноду



Входим в неймноду

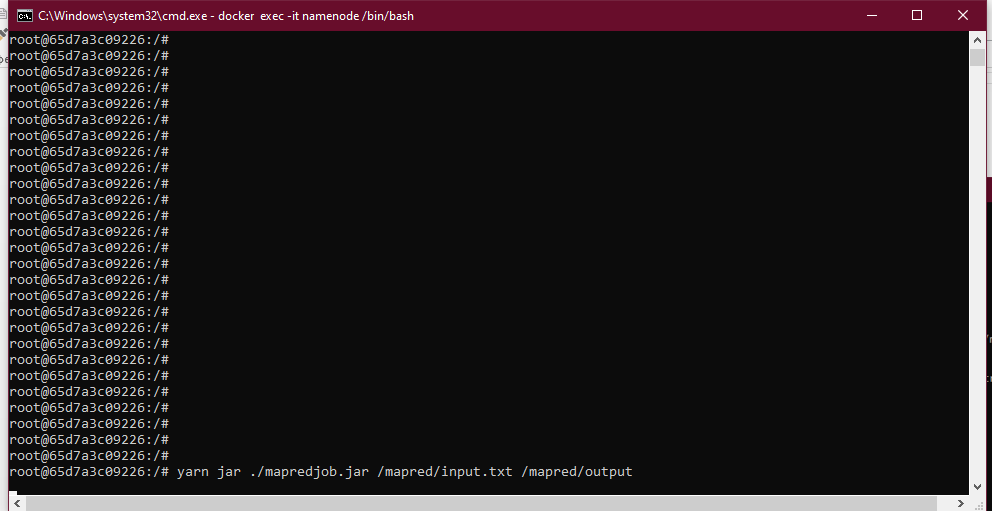


А теперь кладём в hdfs

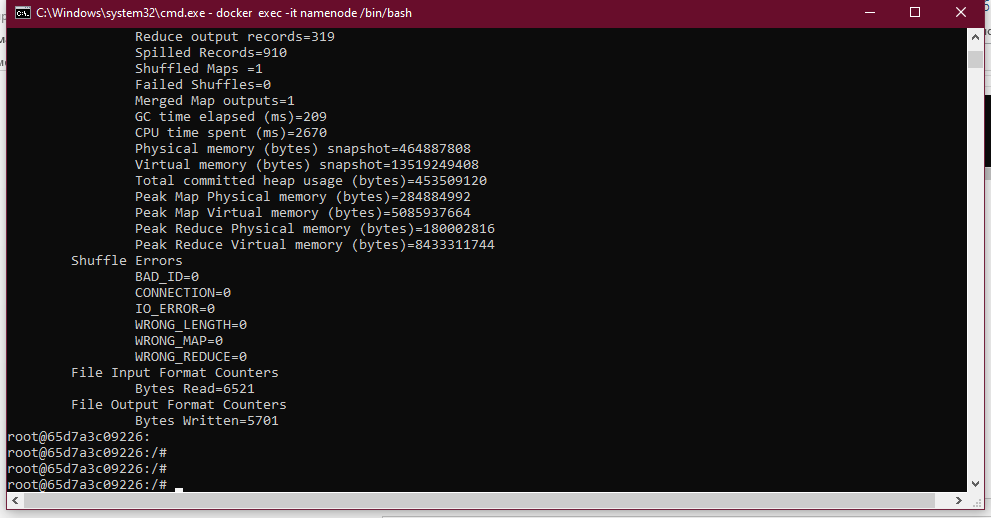


Шаг 8

Запускаем!



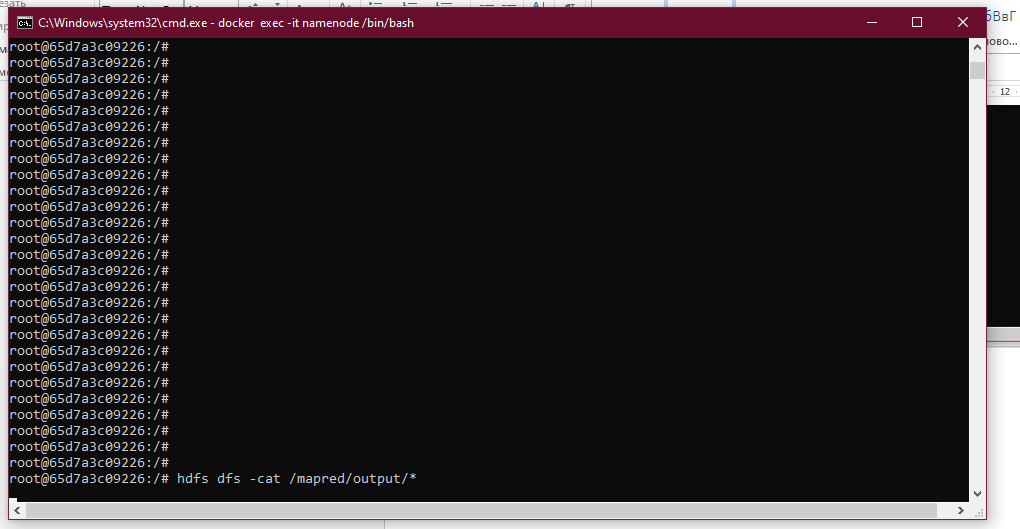
Ждём

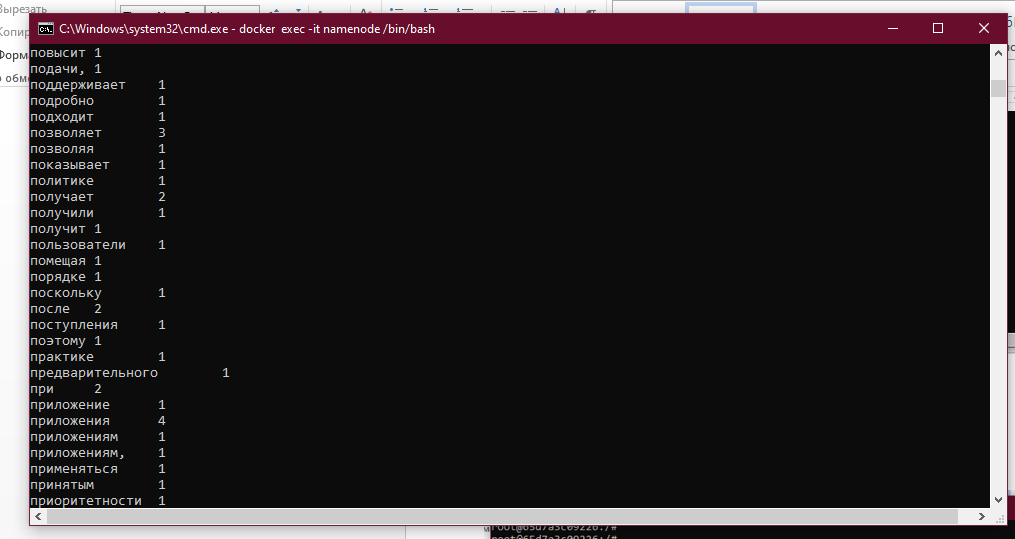


0 ошибок!!!

Шаг 9

Проверим результат





Вывод

* Были получены навыки создания MapReduce job.
* Были изучены и применены основные функции работы с YARN фреймворком MapReduce