**Лабораторная работа № 2**

**Лабораторная работа № 3**

**Группа № РИМ-201211**

**Кубарев А.Д.**

**Салтыш С.С.**

**Юткин Г.А.**

**Краткое описание лабораторной работы**

Установка Hadoop 3.2.2 происходила по инструкции, начиная с настройки операционной системы Ubuntu 18.

Ссылка на инструкцию:

<https://орел-57.рф/pages/aadress.php?page=468>

Подробное описание работы Hadoop:

<https://coderlessons.com/tutorials/bolshie-dannye-i-analitika/uchitsia-hadoop/hadoop-kratkoe-rukovodstvo>

Базовые команды:

https://crazythings.medium.com/hadoop-basic-commands-8a42f8041005

Для работы нам потребуется: OpenJDK (Среда выполнения Java служб, т.к. Hadoop написан на Java), SSH-сервер

Кроме того, потребуется создать отдельного пользователя для Hadoop, чтобы изолировать файловую систему Hadoop от файловой системы Unix.

Hadoop работает в **псевдораспределенном режиме** - это распределенное моделирование на одной машине. Каждый демон (в терминологии Linux - служба) Hadoop, такой как hdfs, yarn, MapReduce и т.д., Будет работать как отдельный процесс Java.

**Запуск установленного Hadoop:**

Переходим под созданный ранее hduser (похожа на отдельную учетную запись):

**sudo su hduser**

Запускаем файловую систему Hadoop: **start-dfs.sh**

Запускаем фреймворк YARN для управления ресурсами кластера и менеджмента задач, в том числе включает фреймворк MapReduce.: **start-yarn.sh**

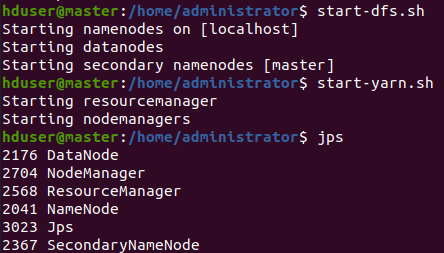
При помощи команды **jps** увидим, что запущены следующие службы:   
NodeManager, SecondaryNameNode, Jps, NameNode, ResourceManager и DataNode.

Для понимания, архитектура Hadoop:



**Задание 0**

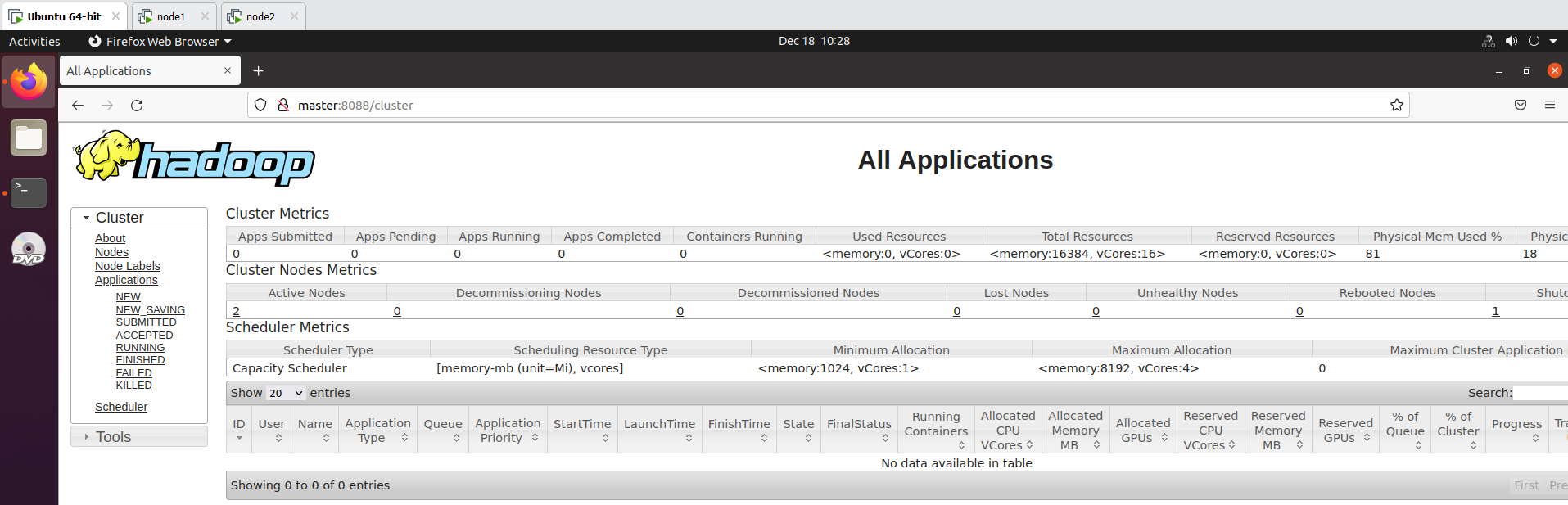
1. Установка и настройка кластера HDFS согласно инструкции. Запуск Hadoop:

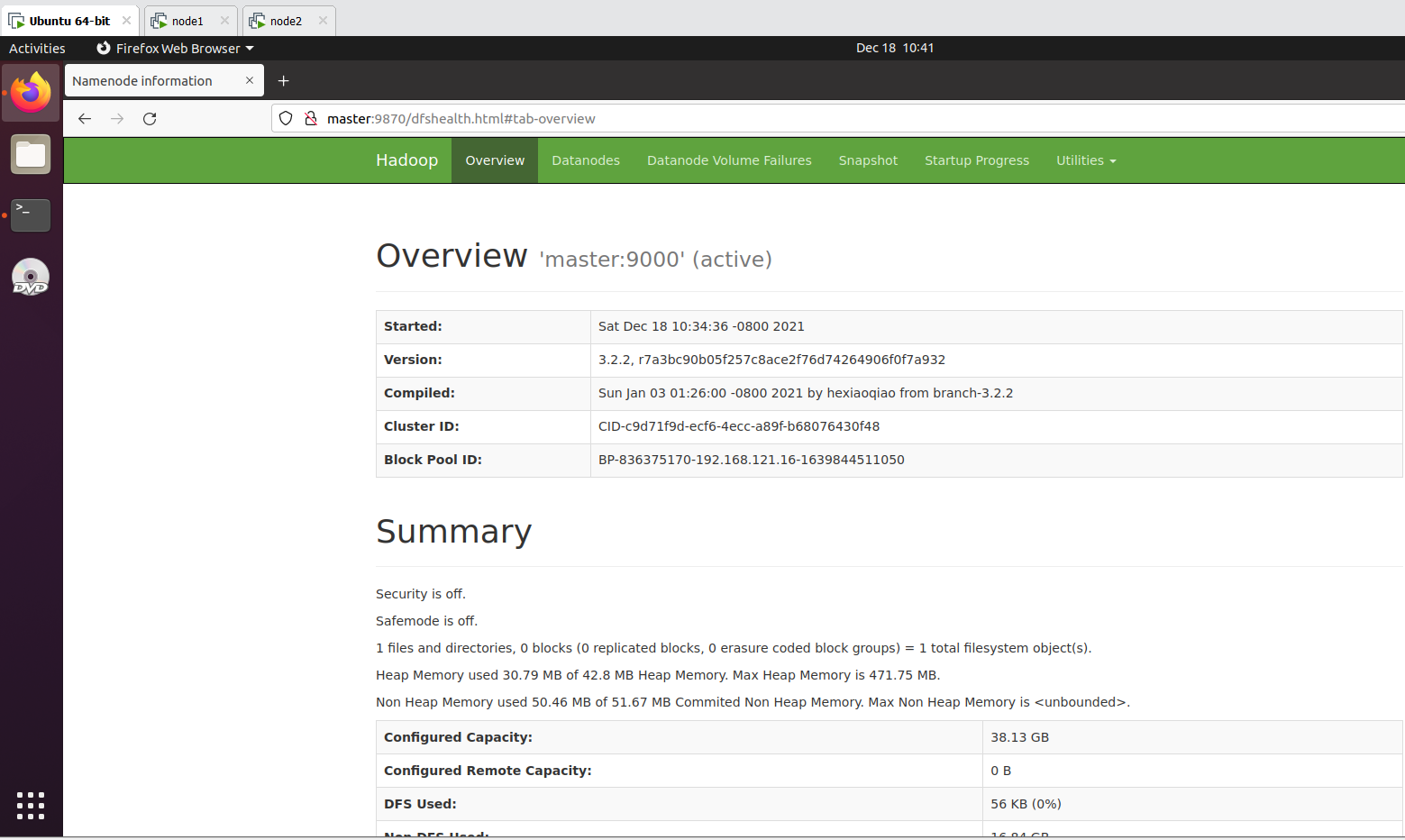


Данный вывод сообщений говорит нам о том, что Hadoop запущено на машине **master** без ошибок.

При успешной настройке также запускаются WEB-интерфейсы hdfs и задач MR в hadoop.

Интерфейс задач доступен по адресу http://Master:8088



Интерфейс состояния HDFS доступен по адресу [http://master:9870](http://master:9870)

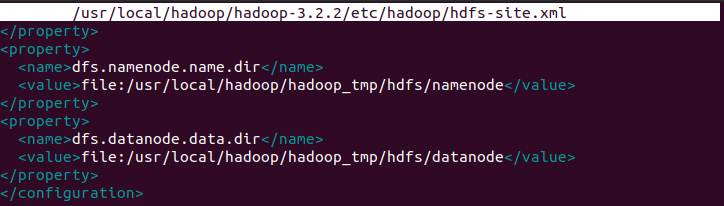
1. Воспроизвести схему кластера из презентации (Лабораторная №2)

Для воспроизведение схемы были созданы дополнительно 2 машины:



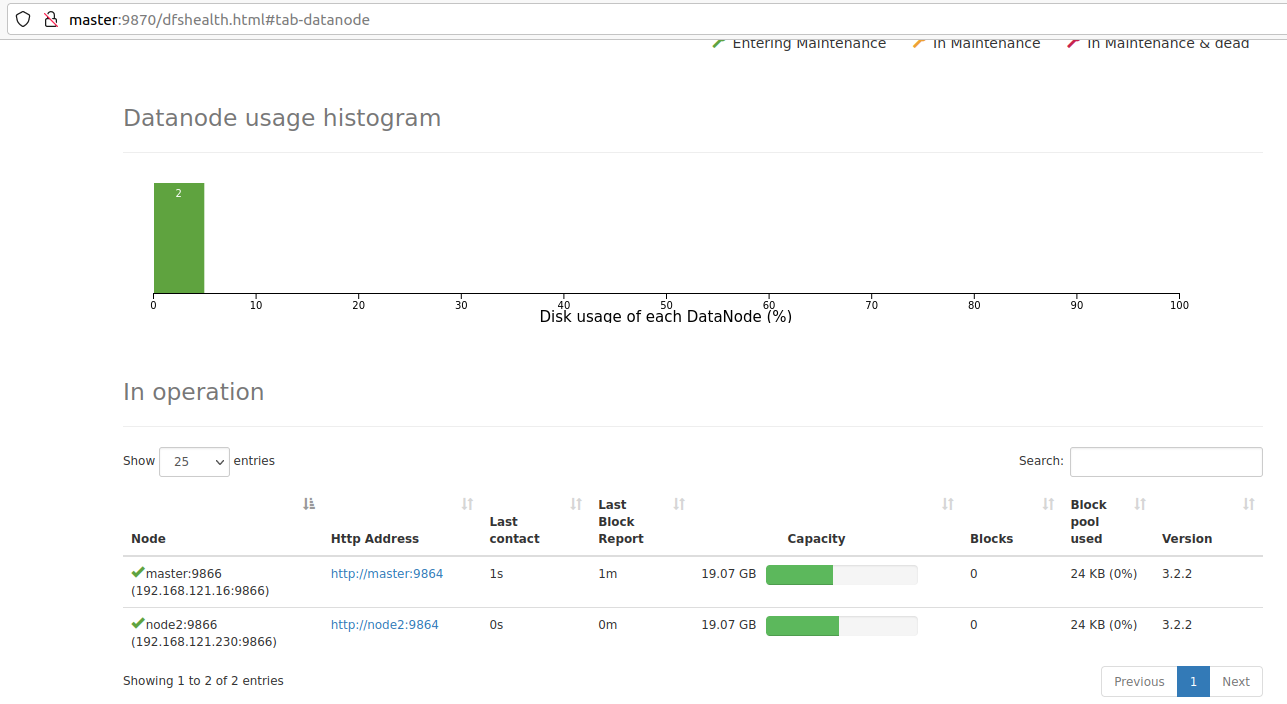
|  |  |
| --- | --- |
| Название машины | IP-адрес |
| master | 192.168.121.16/24 |
| node1 | 192.168.121.240/24 |
| node2 | 192.168.121.230/24 |

1. Настройка Namenode, Datanode:

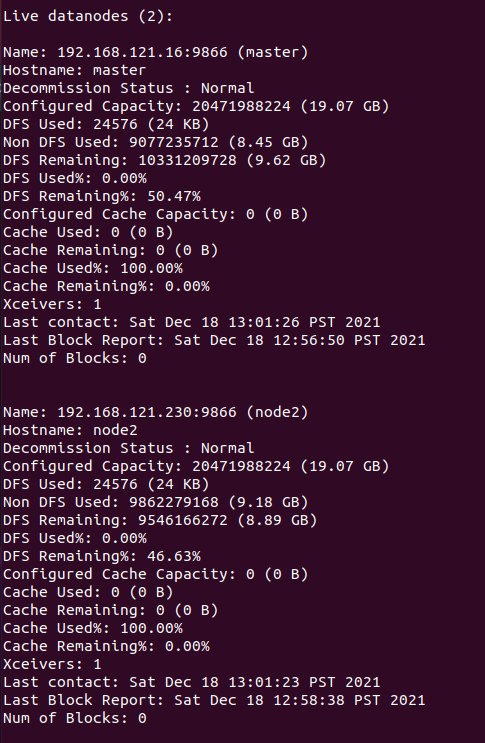


**Задание 1**

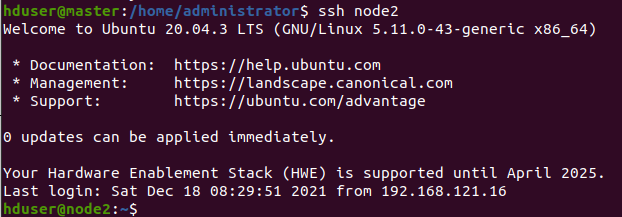
1,2. Произвели настройку node2 и запуск:



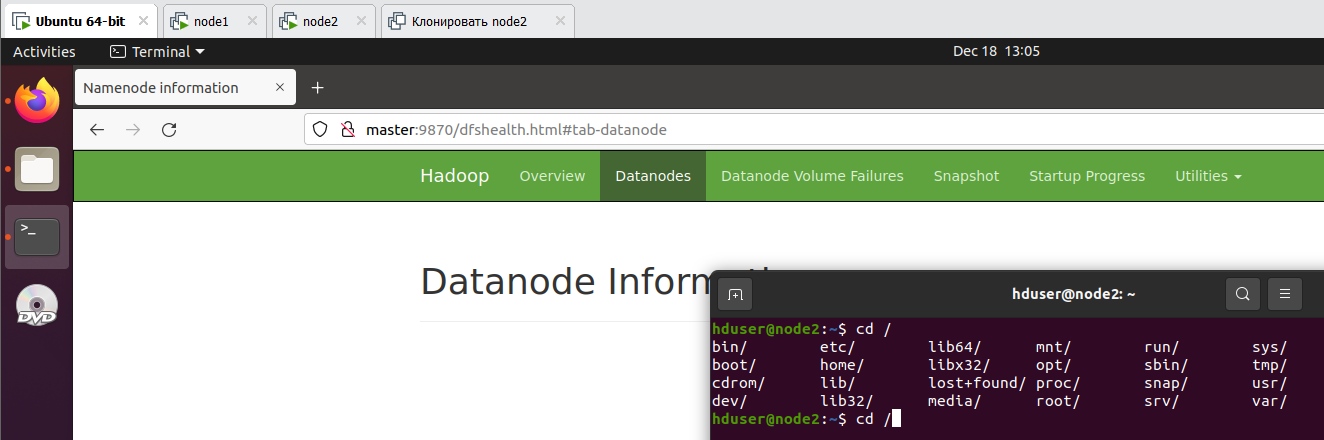
3. Отчет по рабочим узлам



1. С помощью SSH подключаемся на **node2:**



Корневой каталог на **master:**

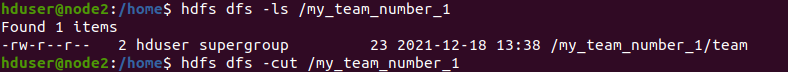


**Задание 2**

1. Создаем папку с идентификатором команды (**hdfs dfs -mkdir /my\_team\_number\_1**):



2. Помещаем файл в папку с идентификатором



Читаем содержимое файла:



**Лабораторная работа 3**

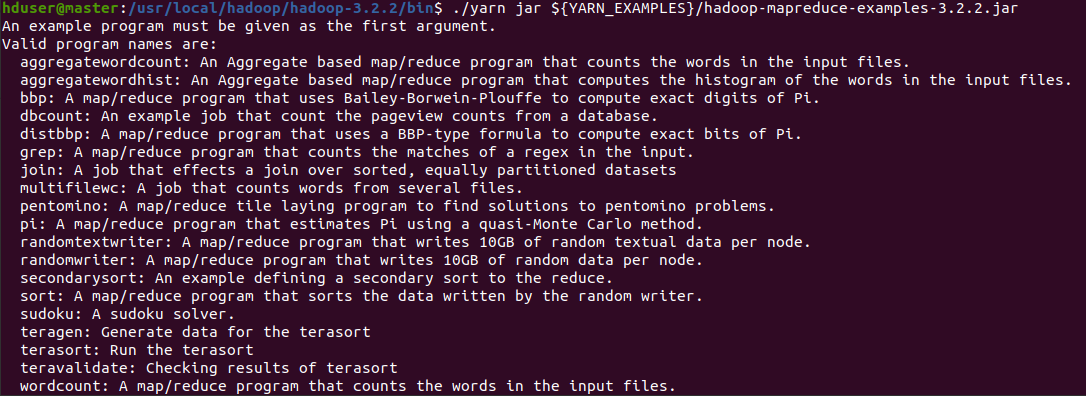
Исходное состояние:

|  |  |
| --- | --- |
| Live datanodes (3): |  |
| Name: 192.168.121.16:9866 (master)  Hostname: master  Decommission Status : Normal  Configured Capacity: 20471988224 (19.07 GB)  DFS Used: 28672 (28 KB)  Non DFS Used: 10507808768 (9.79 GB)  DFS Remaining: 8900632576 (8.29 GB)  DFS Used%: 0.00%  DFS Remaining%: 43.48%  Configured Cache Capacity: 0 (0 B)  Cache Used: 0 (0 B)  Cache Remaining: 0 (0 B)  Cache Used%: 100.00%  Cache Remaining%: 0.00%  Xceivers: 1  Last contact: Thu Dec 23 07:03:42 PST 2021  Last Block Report: Thu Dec 23 06:41:30 PST 2021  Num of Blocks: 0 | Name: 192.168.121.230:9866 (node2)  Hostname: node2  Decommission Status : Normal  Configured Capacity: 20471988224 (19.07 GB)  DFS Used: 28672 (28 KB)  Non DFS Used: 9866194944 (9.19 GB)  DFS Remaining: 9542246400 (8.89 GB)  DFS Used%: 0.00%  DFS Remaining%: 46.61%  Configured Cache Capacity: 0 (0 B)  Cache Used: 0 (0 B)  Cache Remaining: 0 (0 B)  Cache Used%: 100.00%  Cache Remaining%: 0.00%  Xceivers: 1  Last contact: Thu Dec 23 07:03:43 PST 2021  Last Block Report: Thu Dec 23 06:41:29 PST 2021  Num of Blocks: 0 |
| Name: 192.168.121.240:9866 (node1)  Hostname: node1  Decommission Status : Normal  Configured Capacity: 20471988224 (19.07 GB)  DFS Used: 28672 (28 KB)  Non DFS Used: 9809354752 (9.14 GB)  DFS Remaining: 9599086592 (8.94 GB)  DFS Used%: 0.00%  DFS Remaining%: 46.89%  Configured Cache Capacity: 0 (0 B)  Cache Used: 0 (0 B)  Cache Remaining: 0 (0 B)  Cache Used%: 100.00%  Cache Remaining%: 0.00%  Xceivers: 1  Last contact: Thu Dec 23 07:03:44 PST 2021  Last Block Report: Thu Dec 23 06:41:29 PST 2021  Num of Blocks: 0 |  |

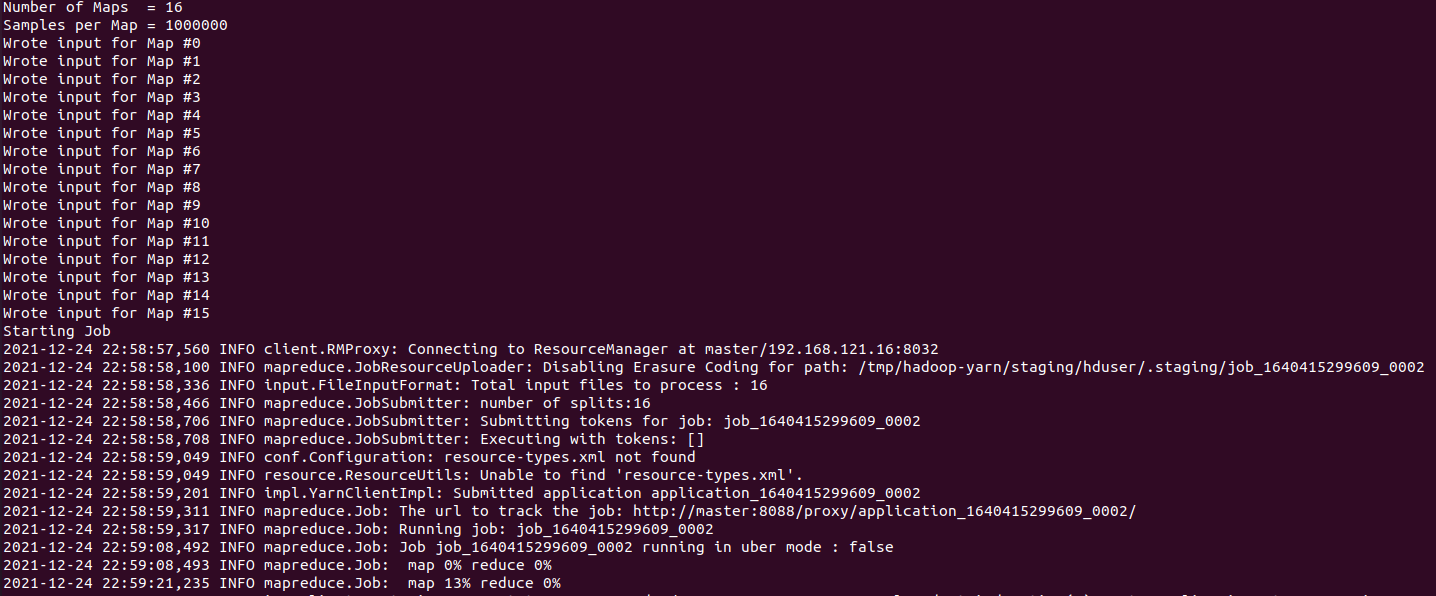
**Задание 0**

1. Следуем инструкции презентации, вводим следующие команды:

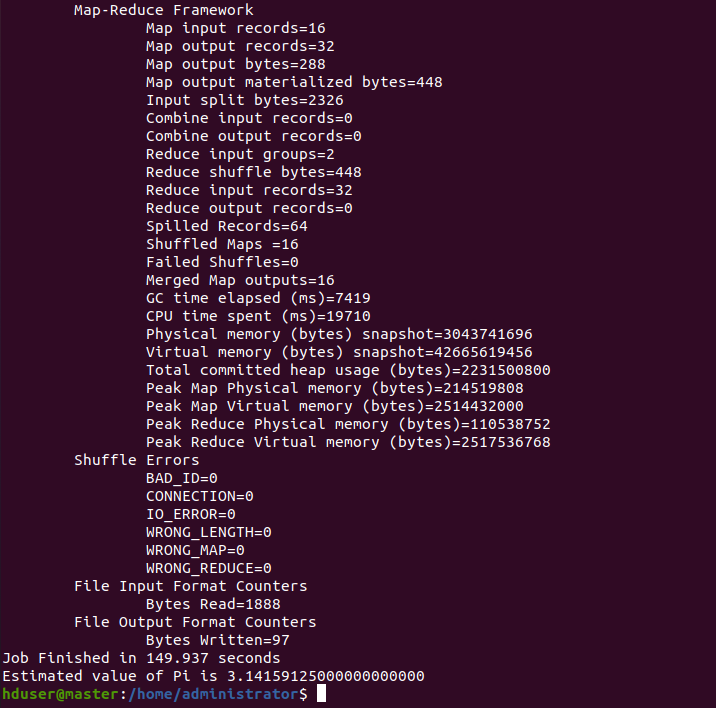




2. Запускаем выполнение расчетов



3. Отчет



**Задание 1**

Выполняем задание используя пример WordCount2 – <https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v2.0>

В файловой системе Hadoop создаем каталоги

hadoop fs -mkdir /user/hduser/wordcount

hadoop fs -mkdir /user/hduser/wordcount/input

В последнем каталоге создаем файл – my\_file

Помещаем туда текст – Hello Hadoop, Goodbye to hadoop.

Создаем .jar файл:

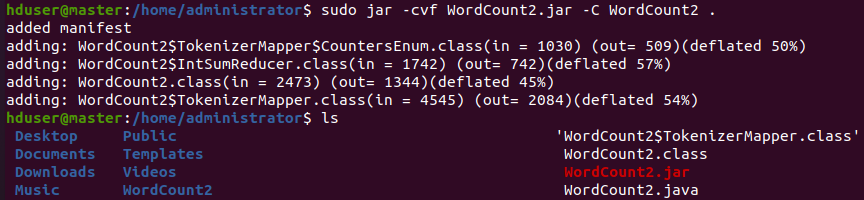
1. Создаем файл WordCount2.java и переносим туда код из примера, указанного на сайте.
2. Далее прописываем в консоли:

export HADOOP\_CLASSPATH=$($HADOOP\_HOME/bin/hadoop classpath)

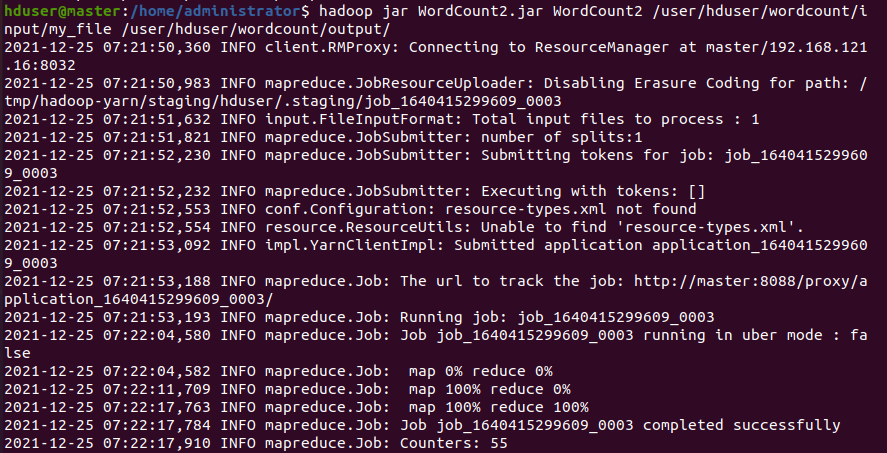
sudo javac -classpath $HADOOP\_CLASSPATH -d WordCount2/ WordCount2.java

sudo jar -cvf WordCount2.jar -C WordCount2 .

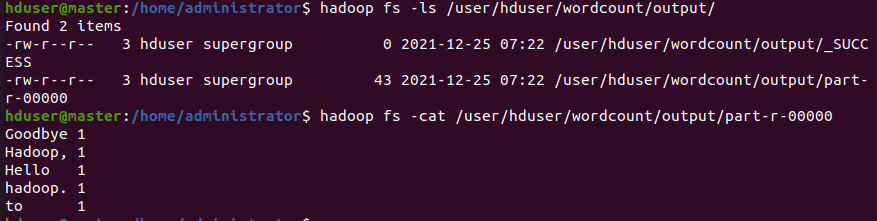
В итоге имеем созданный WordCount2.jar



Далее запускаем приложение



Смотрим наличие файла в каталоге и читаем файл – part-r-00000

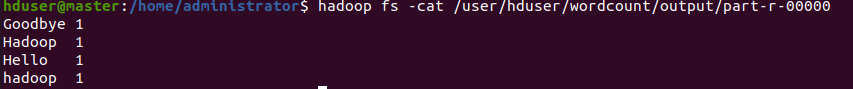


Далее совершенствуем подсчет слов путем добавления исключений, создаем файл – patterns.txt, в который заносим исключения (знаки препинания, символы и т.д.) и переносим его в файловую систему Hadoop

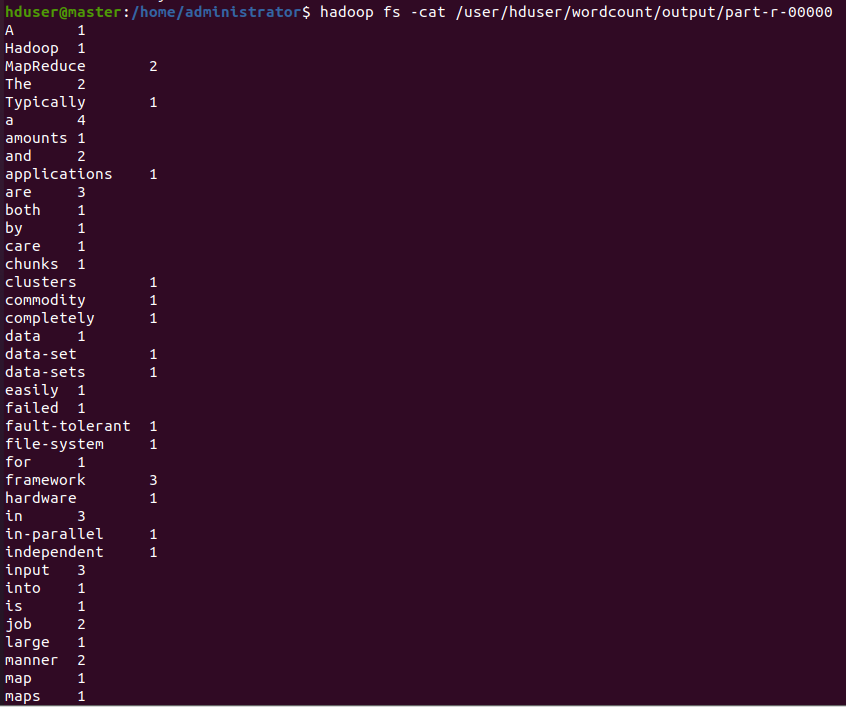
Далее запускаем работу программы длинной командой –

hadoop jar WordCount2.jar WordCount2 -Dwordcount.case.sensitive=true /user/hduser/wordcount/input/my\_file /user/hduser/wordcount/output/ -skip /user/hduser/wordcount/patterns.txt

Результат:



Заменим текст на более длинный. Запускаем программу и получаем результат:



Таким образом, мы создали файл с текстом на машине Hadoop, произвели обработку данного файла, используя код программы из примера. Результатом выполнения программы является подсчет отдельных слов.