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 HDFS»**

Выполнил студент б1-ИФСТ-41,

Мокляк Никита Вячеславович

Проверил преподаватель

Кузьмин Алексей Константинович

Саратов, 2022

Цель работы

* Научиться разворачивать HDFS кластер.
* Изучить и применить основные функции работы с файловой системой Hadoop как распределённой файловой системы.

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.

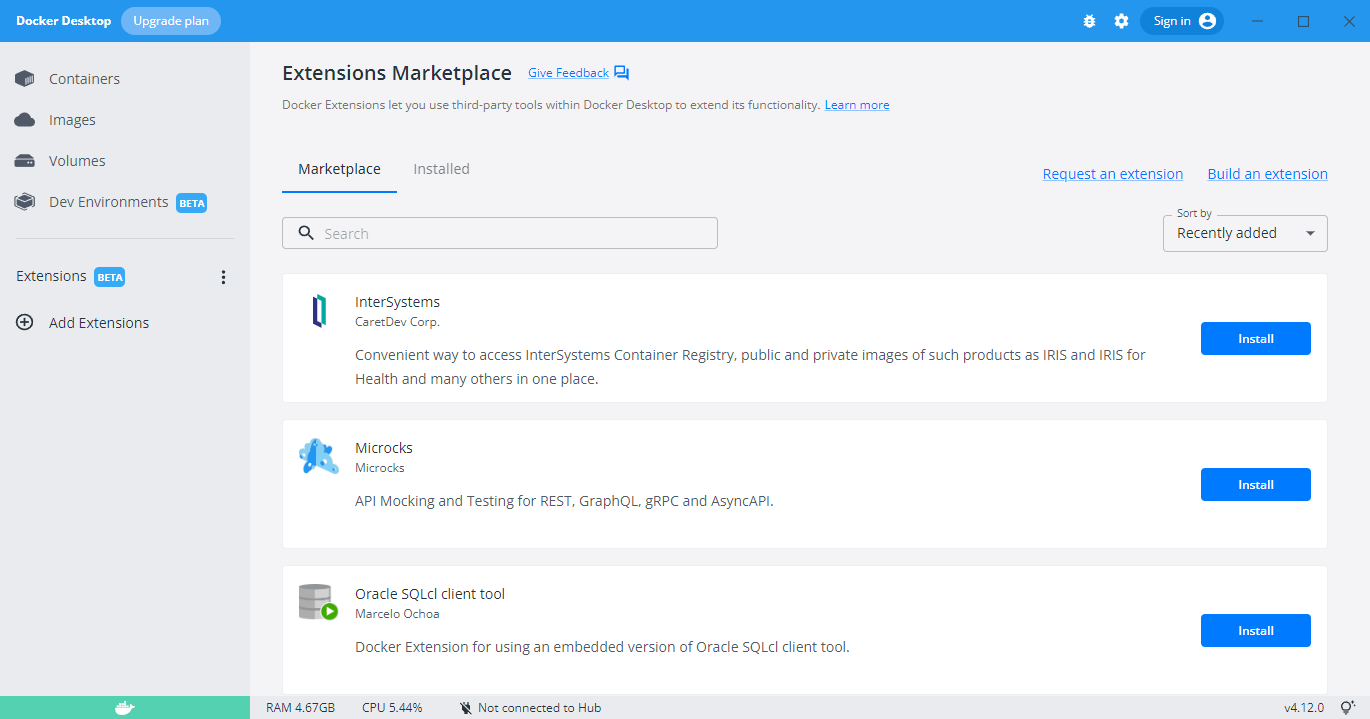
## Задание

Развернуть кластер Hadoop HDFS в Docker.

Опробовать основные команды HDFS.

Шаг 1

Установить Docker



Шаг 2

Найти образы Docker содержащие необходимые компоненты Hadoop и при помощи docker compose описать весь кластер и запустить его.

* Был найден готовый репозиторий <https://github.com/big-data-europe/docker-hadoop>
* В файле hadoop.env находиться конфигурация нашего кластера (оставил без изменений)

CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

CORE\_CONF\_hadoop\_http\_staticuser\_user=root

CORE\_CONF\_hadoop\_proxyuser\_hue\_hosts=\*

CORE\_CONF\_hadoop\_proxyuser\_hue\_groups=\*

CORE\_CONF\_io\_compression\_codecs=org.apache.hadoop.io.compress.SnappyCodec

HDFS\_CONF\_dfs\_webhdfs\_enabled=true

HDFS\_CONF\_dfs\_permissions\_enabled=false

HDFS\_CONF\_dfs\_namenode\_datanode\_registration\_ip\_\_\_hostname\_\_\_check=false

YARN\_CONF\_yarn\_log\_\_\_aggregation\_\_\_enable=true

YARN\_CONF\_yarn\_log\_server\_url=http://historyserver:8188/applicationhistory/logs/

YARN\_CONF\_yarn\_resourcemanager\_recovery\_enabled=true

YARN\_CONF\_yarn\_resourcemanager\_store\_class=org.apache.hadoop.yarn.server.resourcemanager.recovery.FileSystemRMStateStore

YARN\_CONF\_yarn\_resourcemanager\_scheduler\_class=org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler

YARN\_CONF\_yarn\_scheduler\_capacity\_root\_default\_maximum\_\_\_allocation\_\_\_mb=8192

YARN\_CONF\_yarn\_scheduler\_capacity\_root\_default\_maximum\_\_\_allocation\_\_\_vcores=4

YARN\_CONF\_yarn\_resourcemanager\_fs\_state\_\_\_store\_uri=/rmstate

YARN\_CONF\_yarn\_resourcemanager\_system\_\_\_metrics\_\_\_publisher\_enabled=true

YARN\_CONF\_yarn\_resourcemanager\_hostname=resourcemanager

YARN\_CONF\_yarn\_resourcemanager\_address=resourcemanager:8032

YARN\_CONF\_yarn\_resourcemanager\_scheduler\_address=resourcemanager:8030

YARN\_CONF\_yarn\_resourcemanager\_resource\_\_tracker\_address=resourcemanager:8031

YARN\_CONF\_yarn\_timeline\_\_\_service\_enabled=true

YARN\_CONF\_yarn\_timeline\_\_\_service\_generic\_\_\_application\_\_\_history\_enabled=true

YARN\_CONF\_yarn\_timeline\_\_\_service\_hostname=historyserver

YARN\_CONF\_mapreduce\_map\_output\_compress=true

YARN\_CONF\_mapred\_map\_output\_compress\_codec=org.apache.hadoop.io.compress.SnappyCodec

YARN\_CONF\_yarn\_nodemanager\_resource\_memory\_\_\_mb=16384

YARN\_CONF\_yarn\_nodemanager\_resource\_cpu\_\_\_vcores=8

YARN\_CONF\_yarn\_nodemanager\_disk\_\_\_health\_\_\_checker\_max\_\_\_disk\_\_\_utilization\_\_\_per\_\_\_disk\_\_\_percentage=98.5

YARN\_CONF\_yarn\_nodemanager\_remote\_\_\_app\_\_\_log\_\_\_dir=/app-logs

YARN\_CONF\_yarn\_nodemanager\_aux\_\_\_services=mapreduce\_shuffle

MAPRED\_CONF\_mapreduce\_framework\_name=yarn

MAPRED\_CONF\_mapred\_child\_java\_opts=-Xmx4096m

MAPRED\_CONF\_mapreduce\_map\_memory\_mb=4096

MAPRED\_CONF\_mapreduce\_reduce\_memory\_mb=8192

MAPRED\_CONF\_mapreduce\_map\_java\_opts=-Xmx3072m

MAPRED\_CONF\_mapreduce\_reduce\_java\_opts=-Xmx6144m

MAPRED\_CONF\_yarn\_app\_mapreduce\_am\_env=HADOOP\_MAPRED\_HOME=/opt/hadoop-3.2.1/

MAPRED\_CONF\_mapreduce\_map\_env=HADOOP\_MAPRED\_HOME=/opt/hadoop-3.2.1/

MAPRED\_CONF\_mapreduce\_reduce\_env=HADOOP\_MAPRED\_HOME=/opt/hadoop-3.2.1/

* В файле docker-compose.yml находиться конфигурация docker контейнеров, изменил этот файл так, чтобы было 2 datanode и 2 nodemanager

version: "3"

services:

namenode:

image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8

container\_name: namenode

restart: always

ports:

- 9870:9870

- 9000:9000

volumes:

- hadoop\_namenode:/hadoop/dfs/name

environment:

- CLUSTER\_NAME=test

env\_file:

- ./hadoop.env

datanode1:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container\_name: datanode1

restart: always

volumes:

- hadoop\_datanode1:/hadoop/dfs/data

environment:

SERVICE\_PRECONDITION: "namenode:9870"

env\_file:

- ./hadoop.env

datanode2:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

container\_name: datanode2

restart: always

volumes:

- hadoop\_datanode2:/hadoop/dfs/data

environment:

SERVICE\_PRECONDITION: "namenode:9870"

env\_file:

- ./hadoop.env

resourcemanager:

image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8

container\_name: resourcemanager

restart: always

ports:

- 8088:8088

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 datanode2:9864"

env\_file:

- ./hadoop.env

nodemanager1:

image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8

container\_name: nodemanager1

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 resourcemanager:8088"

env\_file:

- ./hadoop.env

nodemanager2:

image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8

container\_name: nodemanager2

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode2:9864 resourcemanager:8088"

env\_file:

- ./hadoop.env

historyserver:

image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8

container\_name: historyserver

restart: always

environment:

SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode1:9864 datanode2:9864 resourcemanager:8088"

volumes:

- hadoop\_historyserver:/hadoop/yarn/timeline

env\_file:

- ./hadoop.env

volumes:

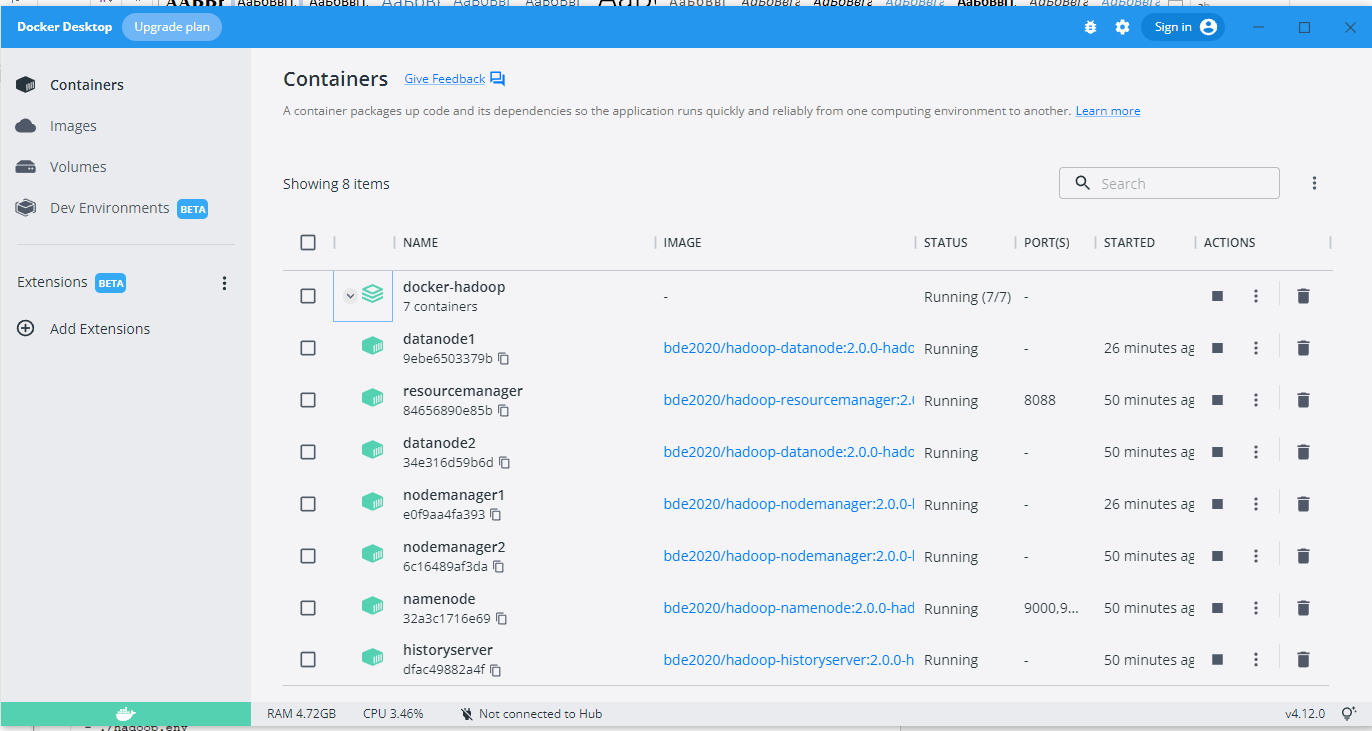
hadoop\_namenode:

hadoop\_datanode1:

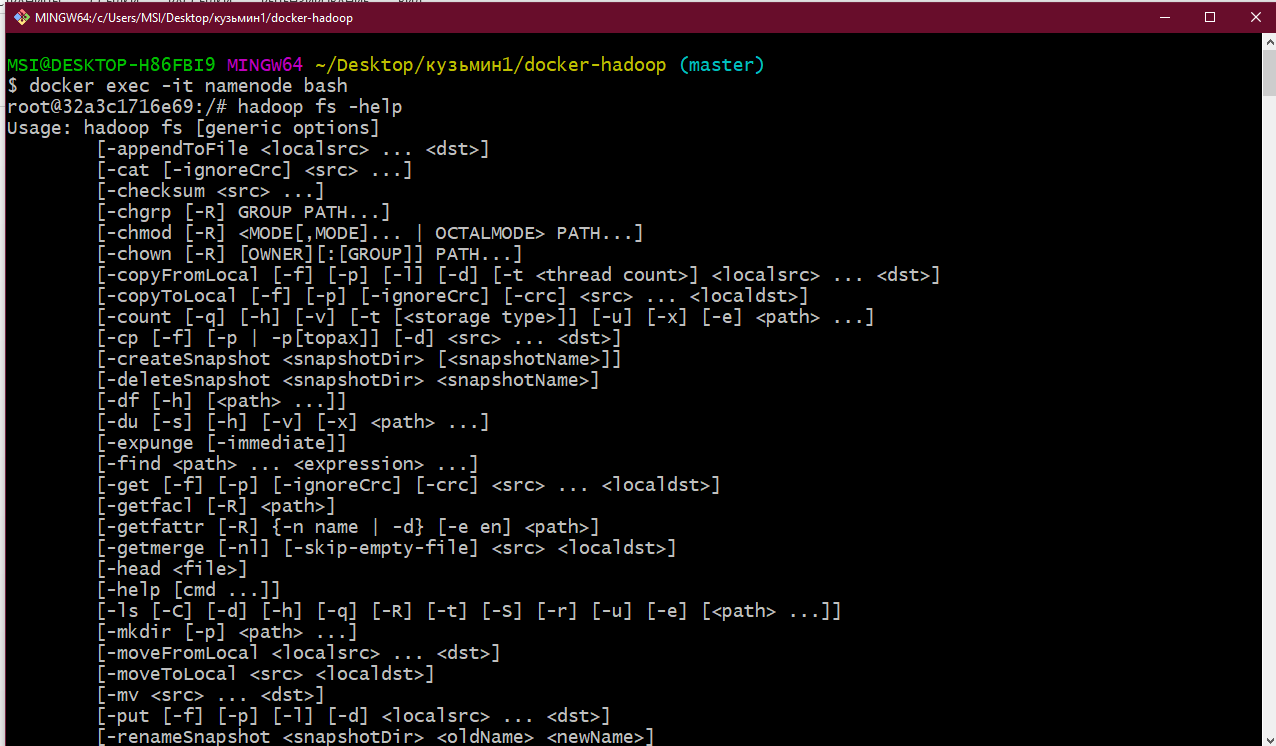
hadoop\_datanode2:

hadoop\_historyserver:

* Выполнил docker compose up –d чтобы запустить все сервисы



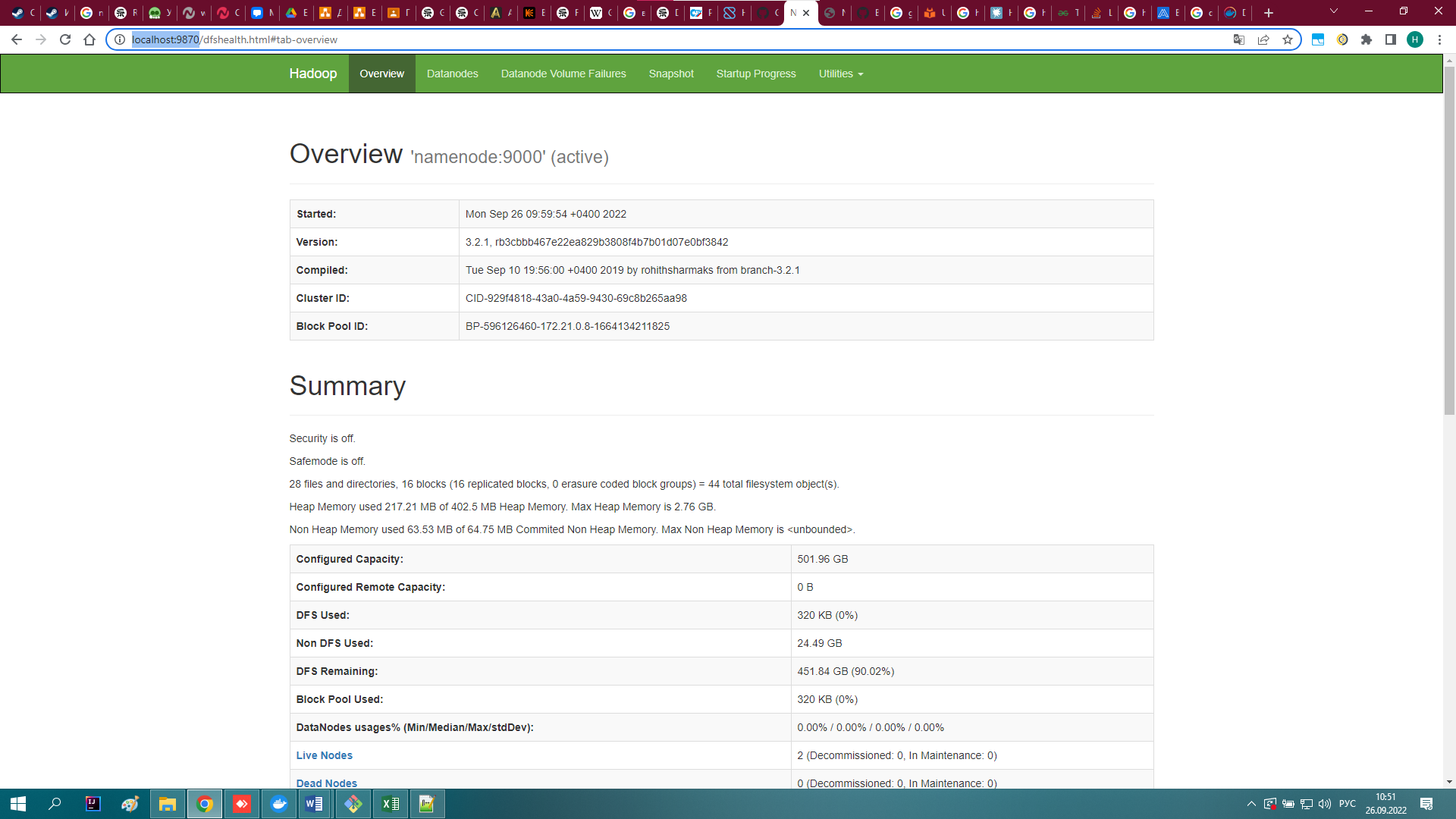
* Теперь выполнив команду docker exec –it namenode bash подключимся к терминалу NameNode



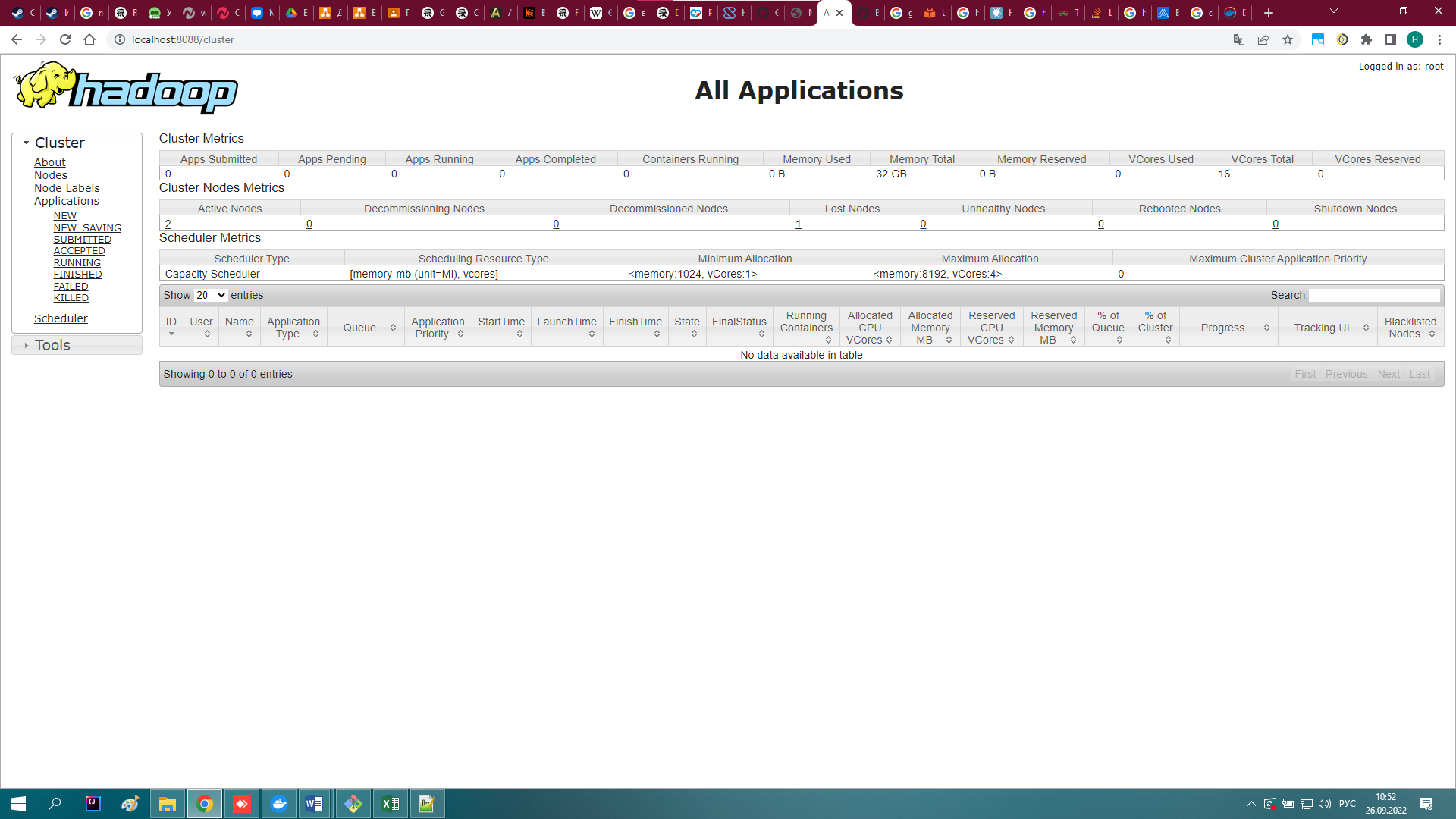
* Проверим, что сами сервисы работают нормально:

Зайдем на

<http://localhost:9870/>

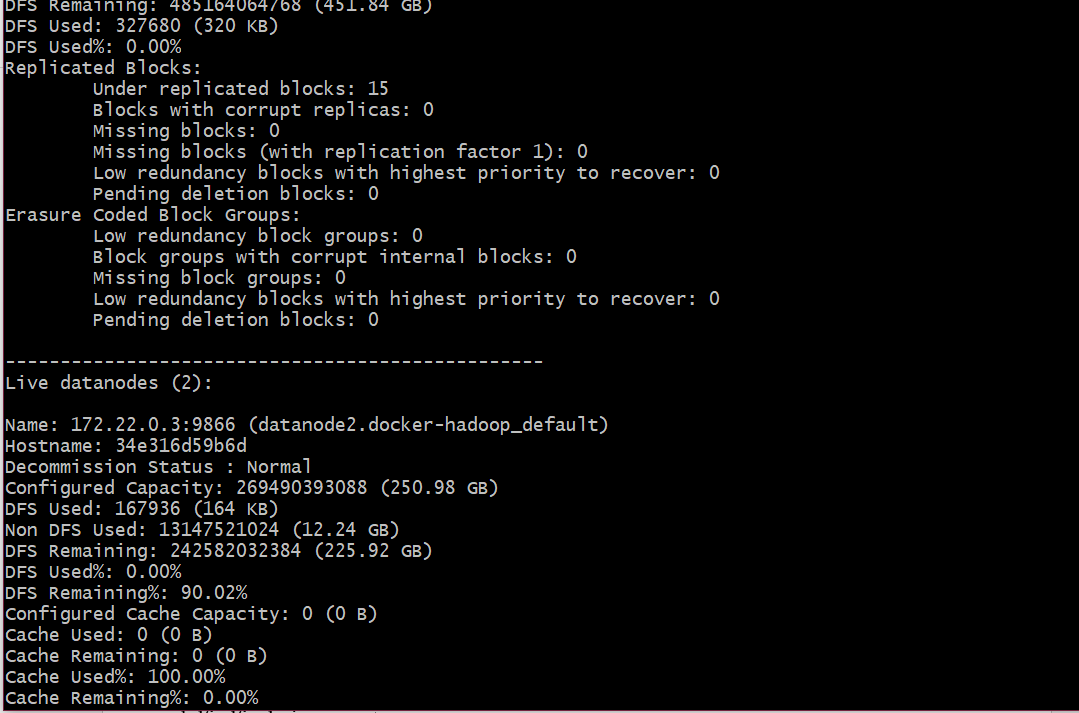


<http://localhost:8088/>

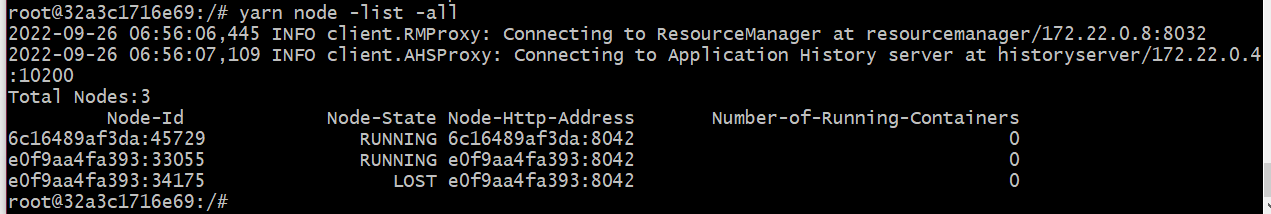


или выполнив команды:

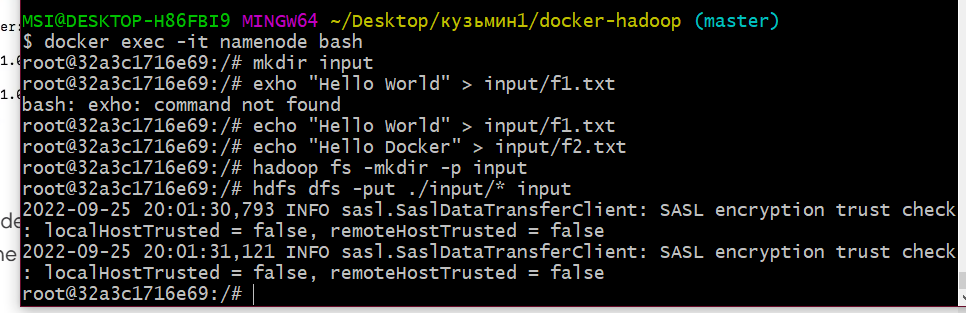
hdfs dfsadmin –report



и yarn node –list -all

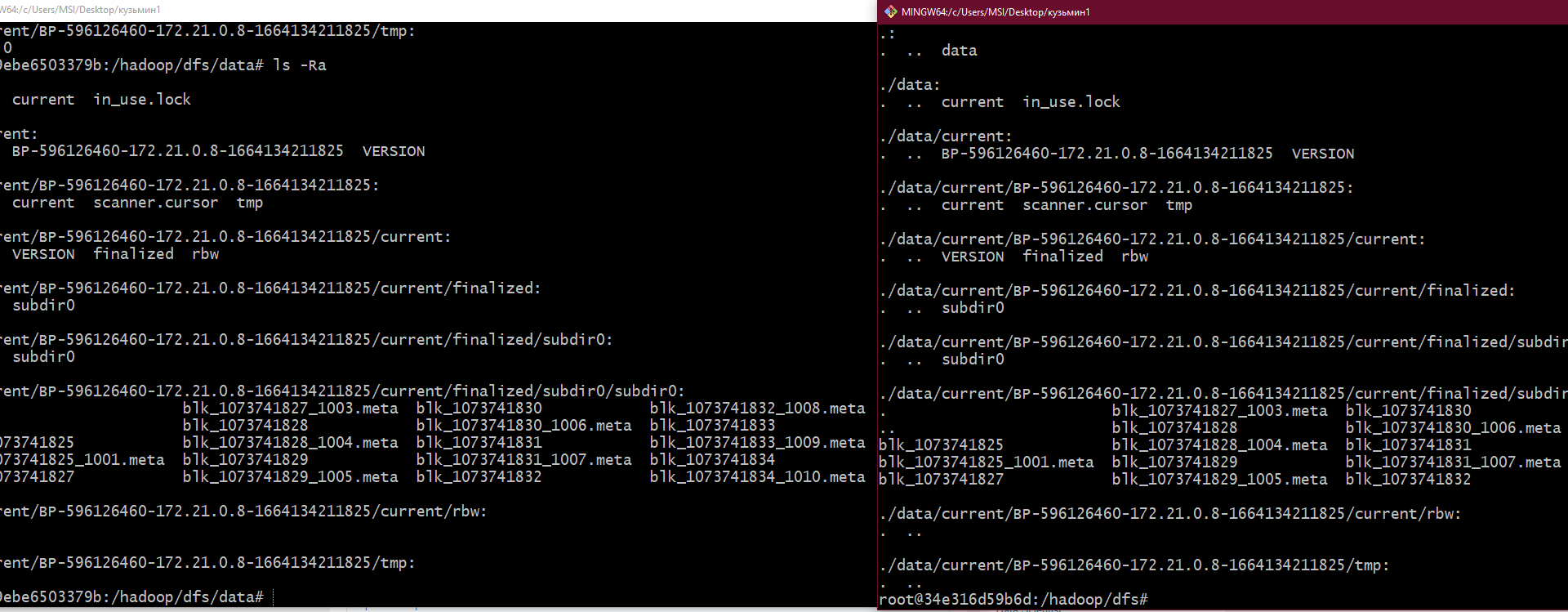


* Теперь скопируем несколько файлов в hdfs и изучим изменение структуры данных

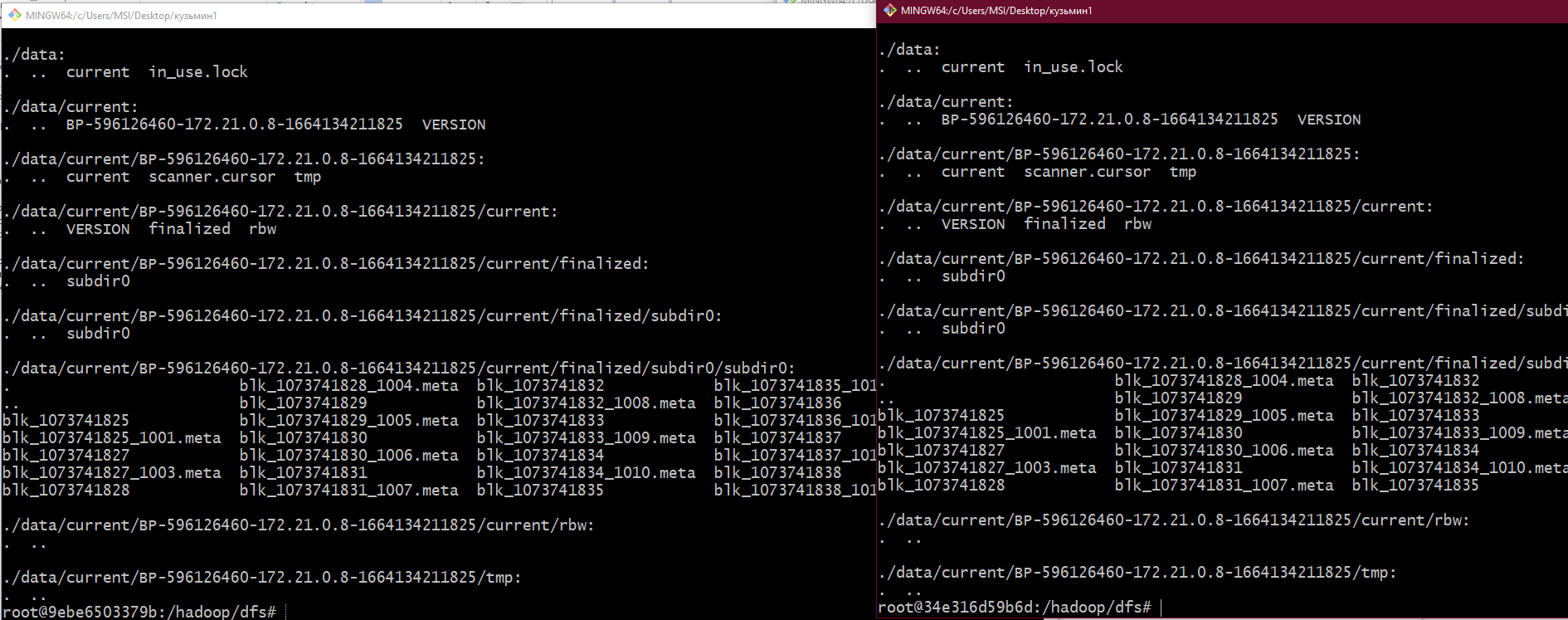


Посмотрим, как изменилась структура

До:

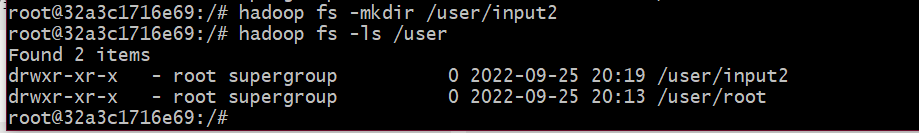


После:

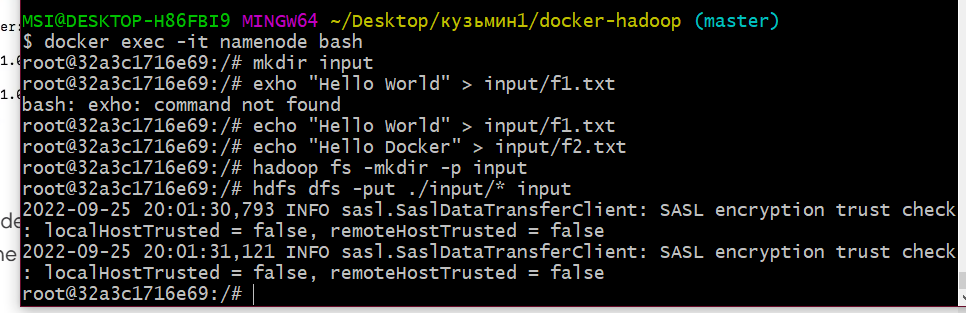


Поскольку я не менял конфигурацию Hadoop была использована конфигурация по умолчанию, в которой количество репликаций = 2, поэтому в обеих датанодах были добавлены файлы.

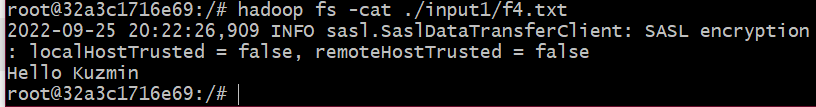
* Выполнить базовые операции работы с файлами в HDFS



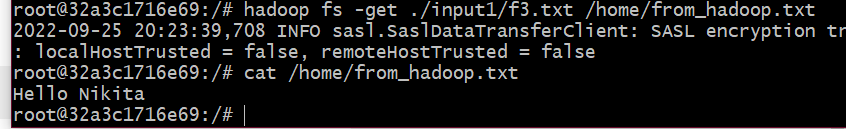
Mkdir + ls



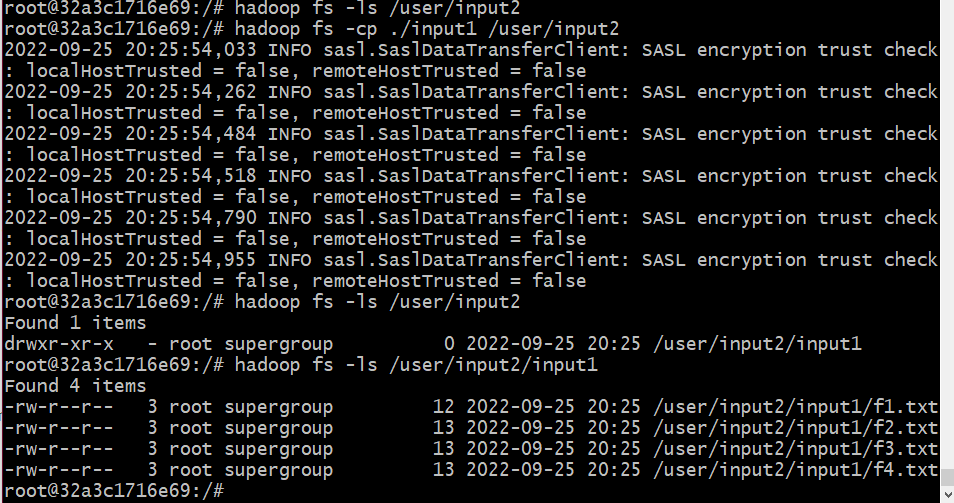
Put



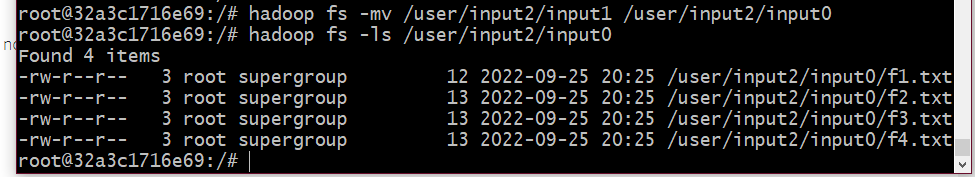
Cat



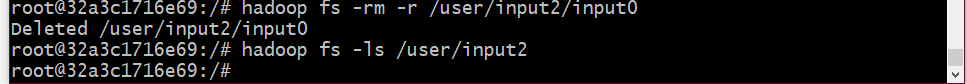
Get



Cp



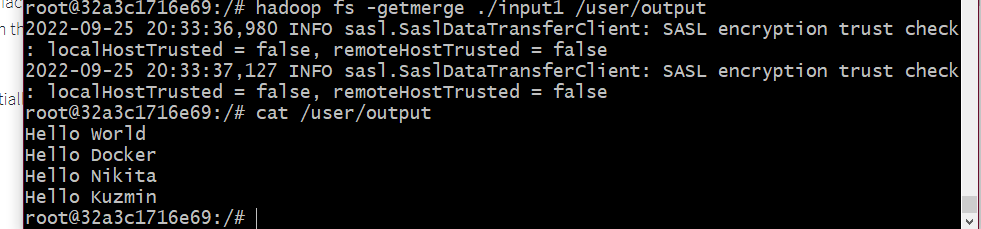
Mv



Rm



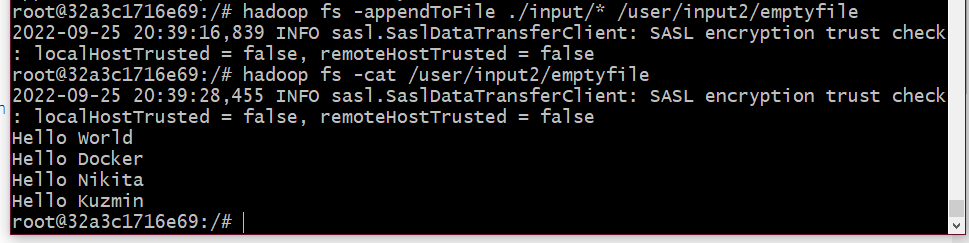
Touchz



Getmerge

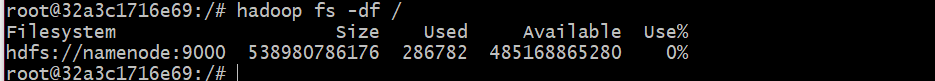


Setrep (replication of single file)

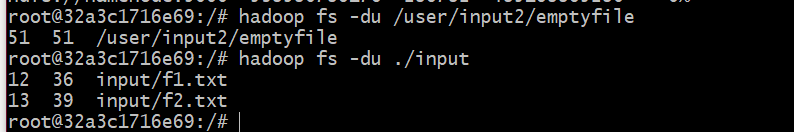


Append to file

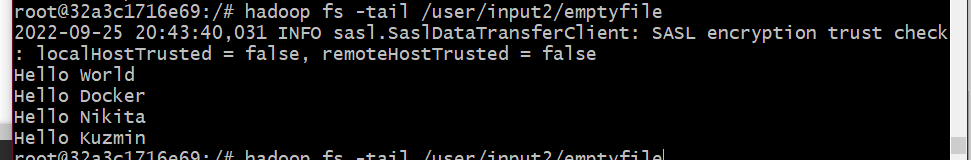
Сложно придумать нормальное изображение на chown chmod особено на chown но типа есть и меняет разрешения и владельца



Df



Du



Tail (последний 1кб файла, я знаю, надо было логи выводить, один фиг не заскриншочу)

Вывод

* Были получены навыки запуска кластера Hadoop, а также было выяснено, что при помощи докера разворачивать кластер hadoop очень просто, намного проще чем вручную с самого начала.
* Были изучены и применены основные функции работы с файловой системой Hadoop как распределённой файловой системы.